Altova MapForce 2023 Enterprise Edition



Benutzer- und Referenzhandbuch

Altova MapForce 2023 Enterprise Edition
Benutzer- und Referenzhandbuch

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2022

© 2016-2022 Altova GmbH

Inhaltsverzeichnis

1 Einführung 19

1.1 Neue Funktionen ... 20
- 1.1.1 Version 2023 ... 20
- 1.1.2 Version 2022 ... 20
- 1.1.3 Version 2021 ... 21
- 1.1.4 Version 2020 ... 22
- 1.1.5 Version 2019 ... 24

1.2 Was ist MapForce? ... 26
- 1.2.1 Mapping: Quellen und Ziele ... 27
- 1.2.2 Transformationssprachen ... 28
- 1.2.3 Mapping-Szenarien ... 29
- 1.2.4 Integration mit Altova-Produkten ... 31

1.3 Übersicht über die Benutzeroberfläche ... 33
- 1.3.1 Leisten ... 34
- 1.3.2 Fenster ... 34
- 1.3.3 Fenster "Meldungen" ... 38
- 1.3.4 Bereiche ... 39

2 Tutorials 43

2.1 Eine Quellkomponente auf eine Zielkomponente ... 44
- 2.1.1 Erstellen und Speichern eines Designs ... 45
- 2.1.2 Hinzufügen der Quellkomponente ... 47
- 2.1.3 Hinzufügen der Zielkomponente ... 48
- 2.1.4 Verbinden von Quell- und Zielkomponente ... 49
- 2.1.5 Vorschau auf das Mapping-Ergebnis ... 52

2.2 Mehrere Quellkomponenten auf eine Zielkomponente ... 54
- 2.2.1 Vorbereiten der Quelldateien ... 55
- 2.2.2 Hinzufügen der Zielkomponente ... 56
- 2.2.3 Überprüfen und Definieren der Input/Output-Dateien ... 57

2.2.4 Verbinden von Quell- und Zielkomponenten ... 58
2.3 Verkettetes Mapping ... 61
2.3.1 Vorbereiten des Mapping-Designs ... 61
2.3.2 Konfigurieren der zweiten Zieldatei ... 62
2.3.3 Ziehen der Verbindungen ... 63
2.3.4 Filtern der Daten ... 64
2.3.5 Anzeige einer Vorschau und Speichern der Ausgabe ... 67
2.4 Mehrere Quellkomponenten auf mehrere Zielkomponenten ... 68
2.4.1 Vorbereiten des Mapping-Designs ... 70
2.4.2 Konfigurieren des Input ... 71
2.4.3 Konfigurieren des Ausgabeteils 1 ... 71
2.4.4 Konfigurieren des Ausgabeteils 2 ... 74

3 Mapping-Grundlagen 76

3.1 Komponenten ... 78
3.1.1 Hinzufügen von Komponenten ... 80
3.1.2 Komponentengrundlagen ... 82
3.1.3 Dateipfade ... 84
3.2 Verbindungen ... 89
3.2.1 Verbindungsarten ... 93
3.2.2 Verbindungseinstellungen ... 99
3.2.3 Kontextmenü für Verbindungen ... 101
3.2.4 Fehlerhafte Verbindungen ... 103
3.2.5 Beibehalten von Verbindungen nach Löschen von Komponenten ... 105
3.3 Allgemeine Verfahren und Funktionalitäten ... 107
3.3.1 Validierung ... 107
3.3.2 Codegenerierung ... 109
3.3.3 Funktionalitäten der Textansicht ... 110
3.3.4 Suchen in der Textansicht ... 113
3.3.5 Mapping-Einstellungen ... 116
3.4 Projekte ... 119
3.4.1 Grundlegendes zu Projekten ... 119
3.4.2 Projekteinstellungen ... 122
3.4.3 Projektordner ... 122

4 Strukturkomponenten 124

4.1 XML und XML-Schema ... 125
4.1.1 XML-Komponenteneinstellungen ... 126
4.1.2 Abgeleitete Typen ... 130
4.1.3 NULL-Werte ... 132
4.1.4 Kommentare und Processing Instructions ... 136
4.1.5 CDATA-Abschnitte ... 137
4.1.6 Wildcards - xs:any / xs:anyAttribute ... 138
4.1.7 Benutzerdefinierte Namespaces ... 141
4.1.8 Digitale Signaturen ... 143
4.1.9 Schema-Manager ... 149
4.2 Datenbanken ... 165
4.2.1 Herstellen einer Verbindung zu einer Datenquelle ... 167
4.2.2 Einführung in Datenbank-Mappings ... 255
4.2.3 Mappen von Daten auf Datenbanken ... 290
4.2.4 Verknüpfen von Datenbankdaten mittels Join ... 344
4.2.5 Filtern und Sortieren von Datenbankdaten (SQL WHERE/ORDER) ... 361
4.2.6 SQL SELECT-Anweisungen als virtuelle Tabellen ... 368
4.2.7 Mappen von XML-Daten von/auf Datenbankfelder ... 377
4.2.8 Anzeigen und Abfragen von Datenbanken ... 388
4.2.9 Gespeicherte Prozeduren ... 402
4.2.10 Behandlung von Datenbankausnahmen ... 429
4.2.11 Datenbankablaufverfolgung und Fehlerprotokollierung ... 443
4.2.12 NoSQL-Datenbanken ... 453
4.3 EDI ... 460
4.3.1 EDI-Terminologie ... 463
4.3.2 Hinzufügen von EDI-Komponenten zum Mapping ... 463
4.3.3 Ausführen von EDI-Mappings ... 465
4.3.4 EDI-Komponenteneinstellungen ... 466
4.3.5 Validierung von EDI-Komponenten ... 471
4.3.6 Generieren von XML anhand von EDI ... 475
4.3.7 ASC X12 ... 479
4.3.8 HIPAA X12 ... 484

4.3.9 HL7 Version 2 ... 490
4.3.10 HL7 Version 3 ... 492
4.3.11 IATA PADIS ... 494
4.3.12 NCPDP SCRIPT ... 498
4.3.13 SAP IDoc ... 501
4.3.14 SWIFT MT ... 504
4.3.15 SWIFT MX / ISO 20022 ... 512
4.3.16 TRADACOMS ... 512
4.3.17 UN/EDIFACT ... 527
4.3.18 Anpassen der EDI-Struktur ... 539
4.3.19 Anpassen der EDI-Validierung ... 573
4.4 CSV- und Textdateien ... 585
4.4.1 Beispiel: Mappen von CSV-Dateien auf XML ... 585
4.4.2 Beispiel: Iterieren durch Datenelemente ... 588
4.4.3 Beispiel: Erstellen von Hierarchien anhand von CSV- und FLF-Dateien ... 591
4.4.4 Definieren der CSV-Optionen ... 595
4.4.5 FLF auf Datenbank ... 599
4.4.6 Definieren der FLF-Optionen ... 603
4.5 Webservices ... 610
4.5.1 REST-Webservices ... 612
4.5.2 WSDL-basierte Webservices ... 630
4.5.3 Webservice-Call-Einstellungen ... 631
4.5.4 Einstellen der HTTP-Sicherheit ... 635
4.5.5 Einstellen der WS-Sicherheit ... 638
4.5.6 Beispiel: Aufrufen eines REST-Webservice ... 640
4.5.7 Beispiel: Mappen von Daten von einem RSS Feed ... 644
4.5.8 Beispiel: Aufruf eines SOAP-Webservice ... 648
4.5.9 Beispiel: OAuth 2.0-Autorisierung ... 652
4.5.10 Verwaltung digitaler Zertifikate ... 664
4.6 MapForce FlexText ... 684
4.6.1 Übersicht ... 684
4.6.2 FlexText Tutorial ... 687
4.6.3 FlexText-Komponenteneinstellungen ... 698
4.6.4 Verwenden von FlexText als Zielkomponente ... 700
4.6.5 FlexText Referenz ... 701

4.6.6 FlexText und Regular Expressions ... 730
4.7 Microsoft OOXML Excel 2007+ ... 738
4.7.1 Hinzufügen von Microsoft Excel-Dateien als Mapping-Komponenten ... 740
4.7.2 Informationen zur Excel 2007+-Komponente ... 741
4.7.3 Hinzufügen und Entfernen von Arbeitsblättern ... 744
4.7.4 Hinzufügen und Entfernen von Zeilenbereichen ... 745
4.7.5 Auswählen von Zellenbereichen ... 746
4.7.6 Einfügen von Spalten zwischen vorhandenen Spalten ... 750
4.7.7 Excel 2007+-Komponenteneinstellungen ... 752
4.7.8 Beispiel: Mappen von Excel 2007+ auf XML ... 755
4.7.9 Beispiel: Konvertieren von Excel-Zeilen in XML-Dateien ... 757
4.7.10 Beispiel: Mappen von Datenbankdaten auf Excel 2007+ ... 761
4.7.11 Beispiel: Aktualisieren bestehender Excel-Blätter ... 764
4.8 XBRL ... 768
4.8.1 Hinzufügen von XBRL-Dateien ... 769
4.8.2 Strukturansichten ... 770
4.8.3 XBRL-Komponentenelemente ... 772
4.8.4 Lesen von Daten aus Inline XBRL ... 775
4.8.5 Taxonomie-Manager ... 777
4.8.6 XBRL-Einstellungen ... 792
4.8.7 XBRL-Standardwerte ... 797
4.8.8 XBRL Hypercubes ... 801
4.8.9 XBRL-Tabellen ... 810
4.8.10 XBRL-Mapping-Beispiele ... 816
4.9 JSON ... 821
4.9.1 Hinzufügen von JSON-Dateien als Mapping-Komponenten ... 823
4.9.2 JSON-Komponenteneinstellungen ... 825
4.9.3 JSON5-Unterstützung ... 827
4.9.4 JSON Lines ... 829
4.9.5 Beispiel: Konvertieren von JSON in CSV ... 832
4.9.6 Beispiel: Konvertieren von Excel in JSON ... 837
4.10 Protocol Buffers ... 843
4.10.1 Hinzufügen von Binärdateien zum Mapping ... 844
4.10.2 Protocol Buffers-Komponenteneinstellungen ... 847
4.10.3 Beispiel: Auslesen von Daten aus Protol Buffers ... 849

4.10.4 Beispiel: Schreiben von Daten in Protocol Buffers ... 854
4.11 Binärdateien ... 860
4.11.1 Beispiel: Lesen von Binärdateien ... 862
4.11.2 Beispiel: Schreiben von Binärdateien ... 864

5 Transformationskomponenten 866

5.1 Einfache Input-Komponente ... 867
5.1.1 Hinzufügen von einfachen Input-Komponenten ... 868
5.1.2 Einstellungen für einfache Input-Komponenten ... 869
5.1.3 Erstellen eines Input-Standardwerts ... 871
5.1.4 Beispiel: Verwenden von Dateinamen als Mapping-Parameter ... 872
5.2 Einfache Output-Komponente ... 878
5.2.1 Hinzufügen einfacher Output-Komponenten ... 879
5.2.2 Beispiel: Vorschau auf die Funktionsausgabe ... 880
5.3 Variablen ... 882
5.3.1 Hinzufügen einer Variablen ... 884
5.3.2 Geltungsbereich und Kontext von Variablen ... 888
5.3.3 Beispiel: Zählen von Datenbanktabellenzeilen ... 891
5.3.4 Beispiel: Filtern und Nummerieren von Nodes ... 892
5.3.5 Beispiel: Unterteilen von Datensätzen in Gruppen und Untergruppen ... 893
5.4 Verknüpfen von Komponenten mittels Join ... 896
5.4.1 Hinzufügen von Join-Bedingungen ... 898
5.4.2 Verknüpfen von drei oder mehr Strukturen ... 901
5.4.3 Beispiel: Verknüpfen von XML-Strukturen ... 902
5.5 Sortieren von Komponenten ... 908
5.5.1 Sortieren nach mehreren Schlüsseln ... 910
5.5.2 Sortieren mit Variablen ... 912
5.6 Filter und Bedingungen ... 914
5.6.1 Beispiel: Filtern von Nodes ... 916
5.6.2 Beispiel: Rückgabe eines Werts auf Basis einer Bedingung ... 918
5.7 Wertezuordnungen ... 921
5.7.1 Beispiel: Ersetzen von Wochentagen ... 925
5.7.2 Beispiel: Ersetzen von Stellenbezeichnungen ... 927
5.8 Gruppierungsfunktionen ... 931

5.8.1 Beispiel: Gruppieren von Datensätzen nach Schlüssel ... 934
5.9 Ausnahmeereignisse ... 938
5.9.1 Beispiel: Ausnahme bei "größer als"-Bedingung ... 939
5.9.2 Beispiel: Ausnahme, wenn Node nicht vorhanden ist ... 940

6 Funktionen 942

6.1 Grundlegendes zu Funktionen ... 943
6.2 Verwalten von Funktionsbibliotheken ... 946
6.2.1 Lokale und globale Bibliotheken ... 948
6.2.2 Relative Bibliothekspfade ... 949
6.3 Standardwerte und Node-Funktionen ... 951
6.3.1 Konfiguration von Regeln ... 953
6.3.2 Anwendungsfallszenarien ... 957
6.3.3 Node-Metadaten in Node-Funktionen ... 963
6.4 Benutzerdefinierte Funktionen ... 967
6.4.1 Benutzerdefinierte Funktionen: Grundlagen ... 968
6.4.2 Parameter von benutzerdefinierten Funktionen ... 973
6.4.3 Rekursive benutzerdefinierte Funktionen ... 978
6.4.4 Look-up-Implementierung ... 980
6.5 Spezielle benutzerdefinierte Funktionen ... 984
6.5.1 Importieren benutzerdefinierter XSLT-Funktionen ... 984
6.5.2 Importieren benutzerdefinierter XQuery 1.0-Funktionen ... 992
6.5.3 Importieren benutzerdefinierter Java- und .NET-Bibliotheken ... 997
6.5.4 Manuelles Referenzieren von Java, C# und C++-Bibliotheken ... 1005
6.6 Regular Expressions ... 1020
6.7 Referenz Funktionsbibliothek ... 1024
6.7.1 core | aggregate functions (Aggregatfunktionen) ... 1026
6.7.2 core | conversion functions (Konvertierungsfunktionen) ... 1034
6.7.3 core | file path functions (Dateipfadfunktionen) ... 1050
6.7.4 core | generator functions (Generierungsfunktionen) ... 1054
6.7.5 core | logical functions (logische Funktionen) ... 1056
6.7.6 core | math functions (mathematische Funktionen) ... 1062
6.7.7 core | node functions (Node-Funktionen) ... 1068
6.7.8 core | QName functions (QName-Funktionen) ... 1074

6.7.9 core | sequence functions (Sequenzfunktionen) ... 1077
6.7.10 core | string functions (String-Funktionen) ... 1104
6.7.11 bson ... 1119
6.7.12 db ... 1122
6.7.13 edifact ... 1125
6.7.14 lang | datetime functions (Datums- und Uhrzeitfunktionen) ... 1129
6.7.15 lang | file functions (Dateifunktionen) ... 1149
6.7.16 lang | generator functions (Generierungsfunktionen) ... 1151
6.7.17 lang | logical functions (logische Funktionen) ... 1152
6.7.18 lang | math functions (mathematische Funktionen) ... 1154
6.7.19 lang | QName functions (QName-Funktionen) ... 1162
6.7.20 lang | string functions (String-Funktionen) ... 1163
6.7.21 mime ... 1180
6.7.22 xbrl ... 1184
6.7.23 xlsx ... 1186
6.7.24 xpath2 | accessors (Accessor-Funktionen) ... 1190
6.7.25 xpath2 | anyURI functions (anyURI-Funktionen) ... 1192
6.7.26 xpath2 | boolean functions (Boolesche Funktionen) ... 1193
6.7.27 xpath2 | constructors (Konstruktoren) ... 1194
6.7.28 xpath2 | context functions (Kontextfunktionen) ... 1195
6.7.29 xpath2 | durations, date and time functions (Zeitdauer-, Datums- und Uhrzeitfunktionen) ... 1198
6.7.30 xpath2 | node functions (Node-Funktionen) ... 1214
6.7.31 xpath2 | numeric functions ... 1221
6.7.32 xpath2 | string functions (String-Funktionen) ... 1222
6.7.33 xpath3 | external information functions ... 1233
6.7.34 xpath3 | formatting functions ... 1236
6.7.35 xpath3 | math functions ... 1240
6.7.36 xpath3 | URI functions ... 1246
6.7.37 xslt | xpath functions ... 1248
6.7.38 xslt | xslt function (XSLT-Funktionen) ... 1250

7 SOAP-Webservices 1255

7.1 Anmerkungen zur SOAP/WSDL-Unterstützung ... 1258

7.2 Erstellen von SOAP-Webservice-Projekten anhand von WSDL-Dateien ... 1261
7.3 Definieren von SOAP-Webservice-Fehlern ... 1270
7.4 Java SOAP-Webservice-Spezifisches ... 1272
7.5 Beispiel: Generieren von SOAP-Webservices (C#) ... 1275
7.6 Beispiel: Generieren von SOAP-Webservices (Java) ... 1290

8 Komplexe Mapping-Szenarien 1298

8.1 Verkettete Mappings ... 1299
8.1.1 Beispiel: Weiterleitung aktiv ... 1301
8.1.2 Beispiel: Weiterleitung deaktiviert ... 1306
8.2 Mappen von Node-Namen ... 1309
8.2.1 Zugriff auf Node-Namen ... 1310
8.2.2 Zugriff auf Nodes eines bestimmten Typs ... 1318
8.2.3 Beispiel: Mappen von Elementnamen auf Attributwerte ... 1322
8.2.4 Beispiel: Gruppieren und Filtern von Nodes nach Namen ... 1325
8.3 Mapping-Regeln und Strategien ... 1330
8.3.1 Sequenzen ... 1331
8.3.2 Der Mapping-Kontext ... 1333
8.3.3 Prioritätskontext ... 1344
8.3.4 Mehrere Zielkomponenten ... 1349
8.4 Ausführen von Mappings mit Authentifizierung ... 1353
8.4.1 Anmeldeinformationen ... 1353
8.4.2 Dynamische Authentifizierung ... 1366
8.4.3 Speicherung von Benutzername und Passwort ... 1369
8.5 Verarbeitung mehrerer Input- oder Output-Dateien ... 1371
8.5.1 Mappen mehrerer Input-Dateien auf eine einzige Output-Datei ... 1374
8.5.2 Mappen mehrerer Input-Dateien auf mehrere Output-Dateien ... 1375
8.5.3 Bereitstellen von Dateinamen als Mapping-Parameter ... 1376
8.5.4 Anzeigen einer Vorschau auf mehrere Output-Dateien ... 1377
8.5.5 Beispiel: Aufteilen einer XML-Datei in mehrere ... 1377
8.5.6 Beispiel: Aufteilen einer Datenbanktabelle in mehrere XML-Dateien ... 1379
8.6 Parsen und Serialisieren von Strings ... 1382
8.6.1 Die Parsen/Serialisieren-Komponente ... 1382
8.6.2 Beispiel: Parsen eines String (Text mit fester Länge auf Excel) ... 1385

8.6.3 Beispiel: Serialisieren in einen String (XML auf Datenbank) 1390
8.7 StyleVision-Ausgabefenster 1395
8.8 Generieren von Mapping-Dokumentation 1399
8.8.1 Vordefinierte StyleVision Power Stylesheets 1401
8.8.2 Benutzerdefinierte Stylesheets 1406

9 Debugger 1409

9.1 Debugger-Vorbereitung 1413
9.2 Informationen zum Debug-Modus 1414
9.3 Hinzufügen und Entfernen von Breakpoints 1418
9.4 Verwendung des Fensters "Werte" 1421
9.5 Verwendung des Fensters "Kontext" 1423
9.6 Verwendung des Fensters "Breakpoints" 1425
9.7 Vorschau auf teilweise generierte Ausgabe 1427
9.8 Anzeigen des aktuellen Werts eines Konnektors 1428
9.9 Zurücksteigen in die nahe Vergangenheit 1429
9.10 Anzeigen des Verlaufs der von einem Konnektor verarbeiteten Werte 1430
9.11 Setzen des Kontexts auf einen Wert 1431

10 Automatisieren mit Altova-Produkten 1432

10.1 Automatisierung mit RaptorXML Server 1433
10.2 Automatisierung mit MapForce Server 1434
10.3 Vorbereiten von Mappings für die Server-Ausführung 1435
10.4 Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442
10.5 Bereitstellen von Mappings auf FlowForce Server 1445
10.6 AS2-Integration 1450
10.7 MapForce-Befehlszeilenschnittstelle 1456

11 Globale Altova-Ressourcen 1462

11.1 Einrichten globaler Ressourcen Teil 1 1463
11.2 Einrichten globaler Ressourcen Teil 2 1465
11.3 XML-Dateien als globale Ressourcen 1469
11.4 Ordner als globale Ressourcen 1471

11.5 Datenbanken als globale Ressourcen ... 1473
11.6 Transformationsergebnisse als globale Ressource ... 1475
11.7 Globale Ressourcen in Ausführungsumgebungen ... 1479
11.7.1 Globale Ressourcen im generierten Code ... 1479
11.7.2 Globale Ressourcen in MapForce Server ... 1480
11.7.3 Globale Ressourcen auf FlowForce Server ... 1480

12 Kataloge in MapForce 1483

12.1 Funktionsweise von Katalogen ... 1484
12.2 Katalogstruktur in MapForce ... 1486
12.3 Anpassen Ihrer Kataloge ... 1488
12.4 Umgebungsvariablen ... 1490

13 MapForce Plug-in für Visual Studio 1492

14 MapForce Plug-in für Eclipse 1495

14.1 Installieren des MapForce Plug-in für Eclipse ... 1496
14.2 Die MapForce-Perspektive ... 1498
14.3 Aufrufen häufig verwendeter Menüs und Funktionen ... 1501
14.4 Arbeiten mit Mappings und Projekten ... 1504
14.4.1 Erstellen eines MapForce/Eclipse-Projekts ... 1504
14.4.2 Erstellen neuer Mappings ... 1506
14.4.3 Importieren bestehender Mappings in ein Eclipse-Projekt ... 1508
14.4.4 Konfigurieren eines automatischen Build und Generierung von MapForce Code ... 1511
14.5 Erweitern des MapForce Plug-in für Eclipse ... 1514

15 Menübefehle 1516

15.1 Datei ... 1517
15.2 Bearbeiten ... 1520
15.3 Einfügen ... 1521
15.4 Projekt ... 1524

15.5 Komponente ... 1526
15.6 Verbindung ... 1528
15.7 Funktion ... 1529
15.8 Ausgabe ... 1530
15.9 Debuggen ... 1532
15.10 Ansicht ... 1533
15.11 Extras ... 1535
15.11.1 Anpassen von Menüs ... 1536
15.11.2 Anpassen von Tastaturkürzeln ... 1537
15.11.3 Optionen ... 1539
15.12 Fenster ... 1556
15.13 Hilfe ... 1558

16 Code Generator 1563

16.1 Was ist neu 1565
16.2 Generieren von C++-Code ... 1567
16.2.1 Generieren von Code anhand eines Mappings ... 1568
16.2.2 Generieren von Code anhand eines Mapping-Projekts ... 1569
16.2.3 Erstellen des Projekts ... 1570
16.2.4 Ausführen der Applikation ... 1570
16.3 Generieren von C#-Code ... 1572
16.3.1 Auswahl der Zielplattform ... 1573
16.3.2 Generieren von Code anhand eines Mappings ... 1574
16.3.3 Generieren von Code anhand eines Mapping-Projekts ... 1575
16.3.4 Erstellen von .NET Framework-Projekten ... 1576
16.3.5 Erstellen von .NET- und .NET Core-Projekten ... 1577
16.4 Generieren von Java-Code ... 1580
16.4.1 Generieren von Code anhand eines Mappings ... 1581
16.4.2 Generieren von Code anhand eines Mapping-Projekts ... 1582
16.4.3 Erstellen des Projekts mit Ant ... 1583
16.4.4 Behandlung von JDBC-Referenzen ... 1584
16.4.5 Beispiel: Generieren und Ausführen von Java-Code ... 1586
16.5 Integrieren von mit MapForce generiertem Code ... 1594
16.5.1 Java-Beispiel ... 1595

16.5.2 C#-Beispiel ... 1597
16.5.3 C++-Beispiel ... 1599
16.5.4 Programmatisches Ändern von Input und Output ... 1600
16.6 Generieren von Code anhand von XML-Schemas oder DTDs ... 1607
16.6.1 Informationen zu Schema Wrapper-Bibliotheken (C++) ... 1610
16.6.2 Informationen zu Schema Wrapper-Bibliotheken (C#) ... 1612
16.6.3 Informationen zu Schema Wrapper-Bibliotheken (Java) ... 1615
16.6.4 Integrieren von Altova-Bibliotheken in vorhandene Projekte ... 1617
16.6.5 Beispiel: Verwendung der Schema Wrapper-Bibliotheken ... 1620
16.6.6 Beispiel: Bestellung ... 1644
16.7 Referenz zu generierten Klassen (C++) ... 1652
16.7.1 altova::DateTime ... 1652
16.7.2 altova::Duration ... 1655
16.7.3 altova::DayTimeDuration ... 1657
16.7.4 altova::YearMonthDuration ... 1658
16.7.5 altova::meta::Attribute ... 1659
16.7.6 altova::meta::ComplexType ... 1659
16.7.7 altova::meta::Element ... 1660
16.7.8 altova::meta::SimpleType ... 1661
16.7.9 [YourSchema]::[CDoc] ... 1662
16.7.10 [YourSchema]::[ElementType] ... 1664
16.7.11 [YourSchema]::MemberAttribute ... 1665
16.7.12 [YourSchema]::MemberElement ... 1666
16.8 Referenz zu generierten Klassen (C#) ... 1668
16.8.1 Altova.Types.DateTime ... 1668
16.8.2 Altova.Types.DateTimeFormat ... 1671
16.8.3 Altova.Types.Duration ... 1673
16.8.4 Altova.Xml.Meta.Attribute ... 1675
16.8.5 Altova.Xml.Meta.ComplexType ... 1676
16.8.6 Altova.Xml.Meta.Element ... 1677
16.8.7 Altova.Xml.Meta.SimpleType ... 1677
16.8.8 [YourSchema].[Doc] ... 1678
16.8.9 [YourSchema].[ElementType] ... 1680
16.8.10 [YourSchemaType].MemberAttribute ... 1681
16.8.11 [YourSchemaType].MemberElement ... 1682

16.9 Referenz zu generierten Klassen (Java) 1684
16.9.1 com.altova.types.DateTime 1684
16.9.2 com.altova.types.Duration 1688
16.9.3 com.altova.xml.meta.Attribute 1692
16.9.4 com.altova.xml.meta.ComplexType 1693
16.9.5 com.altova.xml.meta.Element 1693
16.9.6 com.altova.xml.meta.SimpleType 1694
16.9.7 com.[YourSchema].[Doc] 1695
16.9.8 com.[YourSchema].[ElementType] 1696
16.9.9 com.[YourSchema].[YourSchemaType].MemberAttribute 1698
16.9.10 com.[YourSchema].[YourSchemaType].MemberElement 1698
16.10 Codegenerierung Tipps 1700
16.11 Code Generator-Optionen 1701
16.12 SPL-Referenz 1703
16.12.1 Grundlegende SPL-Struktur 1703
16.12.2 Deklarationen 1704
16.12.3 Variablen 1706
16.12.4 Vordefinierte Variablen 1707
16.12.5 Erstellen von Ausgabedateien 1709
16.12.6 Operatoren 1710
16.12.7 Bedingungen 1711
16.12.8 Collections und foreach 1712
16.12.9 Subroutinen 1713
16.12.10 Vordefinierte Typen 1716

17 Die MapForce API 1722

17.1 Aufruf der API 1723
17.2 Das Objektmodell 1726
17.3 Behandlung von Fehlern 1727
17.4 C#-Beispielprojekt 1729
17.5 Java-Beispielprojekt 1734
17.6 JScript-Beispiele 1738
17.6.1 Applikation starten 1738
17.6.2 Einfacher Dokumentaufruf 1739

17.6.3 Code generieren ... 1740
17.6.4 Codegenerierung (alternative Methode) ... 1742
17.6.5 Ausführen eines Mappings ... 1744
17.6.6 Projektaufgaben ... 1747
17.7 Objektreferenz ... 1752
17.7.1 Schnittstellen ... 1752
17.7.2 Enumerationen ... 1912

18 ActiveX Integration 1919

18.1 Voraussetzungen ... 1920
18.2 Hinzufügen der ActiveX Controls zur Toolbox ... 1922
18.3 Integration auf Applikationsebene ... 1924
18.4 Integration auf Dokumentebene ... 1927
18.5 Beispiele zur ActiveX-Integration ... 1931
18.5.1 C# ... 1931
18.5.2 Java ... 1938
18.5.3 VB.NET ... 1947
18.6 Befehlsreferenz ... 1950
18.6.1 Menü "Datei" ... 1950
18.6.2 Menü "Bearbeiten" ... 1951
18.6.3 Menü "Einfügen" ... 1952
18.6.4 Menü "Projekt" ... 1952
18.6.5 Menü "Komponente" ... 1953
18.6.6 Menü "Verbindung" ... 1954
18.6.7 Menü "Funktion" ... 1955
18.6.8 Menü "Ausgabe" ... 1955
18.6.9 Menü "Debuggen" ... 1956
18.6.10 Menü "Ansicht" ... 1956
18.6.11 Menü "Extras" ... 1957
18.6.12 Menü "Fenster" ... 1958
18.6.13 Menü "Hilfe" ... 1958
18.7 Objektreferenz ... 1959
18.7.1 MapForceCommand ... 1959
18.7.2 MapForceCommands ... 1961

18.7.3 MapForceControl ... 1962
18.7.4 MapForceControlDocument ... 1970
18.7.5 MapForceControlPlaceHolder ... 1977
18.7.6 Enumerationen ... 1980

19 Anhänge 1981

19.1 Anmerkungen zur Unterstützung ... 1982
19.1.1 Unterstützte Quellen und Ziele ... 1982
19.1.2 Unterstützte Funktionalitäten im generierten Code ... 1984
19.2 Informationen zu den Prozessoren ... 1988
19.2.1 Informationen zum XSLT- und XQuery-Prozessor ... 1988
19.2.2 XSLT- und XPath/XQuery-Funktionen ... 1993
19.3 Technische Daten ... 2106
19.3.1 OS- und Arbeitsspeicheranforderungen ... 2106
19.3.2 Altova-Prozessoren ... 2106
19.3.3 Unicode-Unterstützung ... 2107
19.3.4 Internet-Verwendung ... 2107
19.4 Lizenzinformationen ... 2109
19.4.1 Electronic Software Distribution ... 2109
19.4.2 Software-Aktivierung und Lizenzüberwachung ... 2110
19.4.3 Altova Endbenutzer-Lizenzvereinbarung ... 2111

Index 2112

1 Einführung

MapForce® 2023 Enterprise Edition ist ein visuelles Mapping Tool für professionelle Datenintegrationsprojekte. MapForce® ist eine 32/64-Bit Windows Applikation, die auf Windows 7 SP1 mit Plattformupdate, Windows 8, Windows 10, Windows 11 und Windows Server 2008 R2 SP1 mit Plattformupdate oder höher läuft. 64-Bit-Unterstützung steht für die Enterprise und die Professional Edition zur Verfügung. MapForce lässt sich auch mit Visual Studio und Eclipse sowie Microsoft Office-Produkten integrieren. Nähere Informationen finden Sie unter Anmerkungen zur Unterstützung 1982.



Letzte Aktualisierung: 06.10.2022

1.1 Neue Funktionen

In diesem Abschnitt finden Sie eine Beschreibung der neuen Funktionalitäten in den einzelnen MapForce Release-Versionen. Nähere Informationen dazu finden Sie im jeweiligen Unterabschnitt.

1.1.1 Version 2023

Version 2023

- Es werden nun die folgenden Designs unterstützt: *Klassisch*, *Hell* und *Dunkel*. Nähere Informationen dazu finden Sie unter Fenster[1556].
- Interne Aktualisierungen und Verbesserungen
- Die Eclipse-Unterstützung wurde aktualisiert und auf die folgenden Versionen ausgeweitet: 2022-09, 2022-06, 2022-03, 2021-12 (*Professional und Enterprise Edition*). Nähere Informationen dazu finden Sie unter MapForce Plug-in für Eclipse[1495]).
- Unterstützung für ODETTE EDI-Nachrichten (*Enterprise Edition*). Nähere Informationen dazu finden Sie unter EDI[460].
- Unterstützung für die XII Transformation Registry 5 Specification (*Enterprise Edition*). Nähere Informationen dazu finden Sie unter XBRL[768].
- Es besteht nun die Möglichkeit, datenbankbasierte UDF-Parameter[975] und Variablen[883] mit einer hierarchischen Struktur damit in Zusammenhang stehender Tabellen zu erstellen (*Professional und Enterprise Edition*).
- Es ist nun möglich, eine `application/x-www-form-urlencoded` Request-Struktur an einen REST-Service zu senden (*Enterprise Edition*). Nähere Informationen dazu finden Sie unter REST Webservice-Parameter[618].
- Unterstützung für das UN/EDIFACT D.21B- und D.22A-Verzeichnis (*Enterprise Edition*). Nähere Informationen dazu finden Sie unter EDI[460].
- Unterstützung für SQLite 3.39.2, MariaDB 10.9.2 und PostgreSQL 14.5 (*Professional und Enterprise Edition*). Nähere Informationen über alle unterstützten Datenbanken finden Sie unter Datenbanken[165].
- Unterstützung für den XML-Schema-Manager[149], ein Tool, mit dem Sie XML-Schemas installieren und zentral verwalten können, um diese in allen XML-Schema-fähigen Applikationen von Altova verwenden zu können.
- Unterstützung für mapbare EDI-Trennzeichen (*Enterprise Edition*). Diese Funktion wird derzeit für die folgenden EDI-Standards unterstützt: EDIFACT, X12 und NCPDP SCRIPT. Nähere Informationen dazu finden Sie unter EDI-Komponenteneinstellungen[469].

1.1.2 Version 2022

Version 2022 Release 2

- Interne Aktualisierungen und Optimierungen
- Die Eclipse-Unterstützung wurde aktualisiert und auf die folgenden Versionen ausgeweitet: 2021-12; 2021-09; 2021-06; 2021-03 (*Professional und Enterprise Edition*). Nähere Informationen dazu finden Sie unter MapForce Plug-in für Eclipse[1495].

© 2016-2022 Altova GmbH

- Unterstützung für Visual Studio 2022 im MapForce Plug-in für Visual Studio und bei der Codegenerierung (*Professional und Enterprise Edition*). Nähere Informationen dazu finden Sie unter MapForce Plug-in für Visual Studio[1492] und Codegenerierung[1563].
- Unterstützung für NET 6.0 bei der Codegenerierung (*Professional und Enterprise Edition*). Nähere Informationen dazu finden Sie unter Codegenerierung[1563].
- Unterstützung neuer Datenbankversionen: PostgreSQL 14, SQLite 3.37.2, MariaDB 10.6.5, MySQL 8.0.28, IBM DB2 11.5.7 (*Professional und Enterprise Edition*). Nähere Informationen über alle unterstützten Datenbanken finden Sie unter Datenbanken[165].
- Von Bildern kann im Fenster **Projekt** nun eine Vorschau angezeigt werden (*Professional und Enterprise Edition*). Nähere Informationen dazu finden Sie unter Grundlegendes zu Projekten[121].
- Es können nun EBA-konforme Filing Indicators für XBRL-Zielkomponenten erstellt werden (*Enterprise Edition*). Nähere Informationen dazu finden Sie unter Filing Indicators[773].

Version 2022

- Interne Aktualisierungen und Optimierungen
- Die Eclipse-Unterstützung wurde aktualisiert und auf die folgenden Versionen ausgeweitet: 2021-09; 2021-06; 2021-03; 2020-12 (*Professional und Enterprise Edition*). Nähere Informationen dazu finden Sie unter MapForce Plug-in für Eclipse[1495].
- "Alles kopieren"-Verbindungen[97] unterstützen nun JSON. Diese Funktionalität steht nur für kompatible JSON-Typen zur Verfügung (*Enterprise Edition*).
- Es gibt ein neues StyleVision-Ausgabefenster namens *Text*. Wenn einer Komponente eine SPS-Datei zugewiesen wird, können Sie in MapForce eine Vorschau auf das neue reine Textausgabeformat anzeigen (*Professional und Enterprise Edition*). Nähere Informationen dazu finden Sie unter StyleVision-Ausgabefenster[1395].
- Unterstützung für JSON-Schema in Variablen[882] und Parametern von benutzerdefinierten Funktionen[973] (*Enterprise Edition*).
- Unterstützung für NoSQL-Datenbanken: MongoDB und CouchDB (*Enterprise Edition*). Nähere Informationen über alle unterstützten Datenbanken finden Sie unter Datenbanken[165].
- Es steht nun eine neue `bson`-Funktionsbibliothek zur Verfügung, mit Hilfe derer Sie einige der BSON-Typen erstellen und bearbeiten können (*Enterprise Edition*). Nähere Informationen dazu finden Sie unter `bson`[1119].
- Unterstützung für das EDIFACT D.20B- und D.21A-Verzeichnis. Nähere Informationen dazu finden Sie unter EDI[460] (*Enterprise Edition*).
- Unterstützung für SWIFT 2021. Nähere Informationen dazu finden Sie unter EDI[460] (*Enterprise Edition*).

1.1.3 Version 2021

Version 2021 Release 3

- Unterstützung für den neuen JSON Schema Draft 2019-09 und Draft 2020-12 (*nur Enterprise Edition*).

Version 2021 Release 2

- Unterstützung von XSLT 3.0 als Mapping-Sprache. Siehe Generieren von XSLT-Code[109]. Außerdem enthält MapForce nun vordefinierte Funktionen, die unterstützt werden, wenn als Mapping-Sprache XSLT 3.0 ausgewählt ist. Nähere Informationen dazu finden Sie unter Referenz Funktionsbibliothek[1024].

- Bei der Generierung von C#-Code kann in den Codegenerierungsoptionen[1701] als Ziel-Framework (zusätzlich zur bestehenden Unterstützung für .NET-Framework-Projekte) .NET Core 3.1 und .NET 5.0 ausgewählt werden. Nähere Informationen dazu finden Sie unter Generieren von C#-Code[1572].
- Interne Aktualisierungen und Verbesserungen

Version 2021

- Alle XBRL-fähigen Applikationen von Altova einschließlich MapForce profitieren nun von einer vereinfachten XBRL-Taxonomieverwaltung und releaseunabhängigen Taxonomieaktualisierungen. Zu diesem Zweck steht nun ein neuer XBRL-Taxonomie-Manager[777] zur Verfügung, mit dem Sie die aktuell installierten XBRL- PKG Taxonomien anzeigen, verwalten und aktualisieren können.
- Bei Aufruf eines REST-Webservice können Sie die URL des Webservice nun als Parameter für das Mapping angeben. Diese Option bietet mehr Flexibilität, wenn Sie komplette Kontrolle über die URL benötigen und diese zur Mapping-Laufzeit definieren möchten, siehe Hinzufügen eines (REST)-Webservice-Aufrufs[612].
- Unterstützung für das Mappen von Daten von oder auf das SWIFT MT[504]-Format.
- EDI-Collections können nun in einem Verzeichnis Ihrer Wahl oder im Verzeichnis Ihres Benutzerprofils gespeichert werden, siehe Erstellen einer benutzerdefinierten EDI-Collection[540].
- Wenn ein Mapping Daten in eine EDI-Zielkomponente schreibt, können Sie die Komponente ändern, sodass anstelle einer EDI-Ausgabe eine XML-Ausgabe generiert wird. Auf diese Art kann steht eine andere Art der Datenvorschau zur Verfügung. Außerdem können Sie EDI-Dateien auch mit einem Mausklick ins XML-Format konvertieren, ohne dass dazu ein XML-Schema benötigt wird oder Mapping-Verbindungen gezogen werden müssen. Nähere Informationen dazu finden Sie unter Konvertieren von EDI in XML[475].
- Ein MapForce Mapping kann BLOB (Binary Lage Object)-Daten aus Binärdateien auslesen und Binärdateien auf die Festplatte schreiben. Dadurch können BLOB-Felder z.B. aus einer Datenbank ausgelesen und als Bilddateien auf dem Rechner gespeichert werden oder es können Binärdateien wie PDF-Dateien von der Festplatte gelesen und als `xs:base64Binary`-Felder in einer XML-Datei gespeichert werden. Nähere Informationen dazu finden Sie unter Binärdateien[860].
- Unterstützung neuer Datenbankversionen: MariaDB 10.4, 10.5
- Unterstützung neuer Eclipse-Versionen: 2019.09, 2019.12, 2020.03, 2020.06
- Wenn Sie in einem Mapping mehrere Datenbanktabellen oder Ansichten mit Hilfe von SQL-Join-Komponenten verbinden, können Sie als Join-Modus entweder LINKER ÄUSSERER JOIN oder INNERER JOIN einstellen, siehe Ändern des Join-Modus[346].
- Interne Aktualisierungen und Verbesserungen

1.1.4 Version 2020

Version 2020 Release 2

- Es gibt ein neues Fenster Bibliotheken verwalten[36], über das Sie alle auf Dokument- oder Programmebene importierten Funktionsbibliotheken (darunter auch benutzerdefinierte MapForce-Funktionen und andere Arten von Bibliotheken) anzeigen und verwalten können. Dadurch können Sie etwa benutzerdefinierte Funktionen einfach von einem Mapping in ein anderes kopieren, siehe Kopieren und von benutzerdefinierten Funktionen in andere Mappings[972].
- Der Pfad von in eine Mapping-Datei importierten Bibliotheken ist standardmäßig relativ zur Mapping-Datei, siehe Relative Bibliothekspfade[949]. Sie können Mappings weiterhin wie in früheren Versionen auf Applikationsebene importieren, doch ist der Bibliothekspfad in diesem Fall immer absolut.

© 2016-2022 Altova GmbH

- Wenn XSLT- oder XQuery-Bibliotheken in eine Mapping-Datei importiert werden, können Sie XSLT- oder XQuery-Code generieren, in dem die importierten Bibliotheksdateien über einen relativen Pfad referenziert werden. Die neue Option steht im Dialogfeld Mapping-Einstellungen[116] zur Verfügung.
- Die MapForce API[1722] wurde durch neue Mitglieder, mit Hilfe derer Sie importierte Bibliotheken programmatisch verwalten (z.B. hinzufügen, oder entfernen) können, ergänzt.
- Für XML-Schema-Wrapper-Bibliotheken generierter Code erlaubt nun mehr Kontrolle über Element-Namespaces und -Präfixe. Zum Deklarieren oder Außerkraftsetzen von Namespaces für ein Element oder zum Anhängen eines Elements mit einem vorangestellten Namespace stehen neue Methoden zur Verfügung. Siehe Beispiel: Bestellung[1644].
- Unterstützung neuer Datenbankversionen: PostgreSQL 12.1 und Informix 14.10.
- Zusätzlich zur Generierung von Excel-Dateien anhand eines Mappings unterstützt MapForce nun die Aktualisierung vorhandener Excel-Dateien (wenn die Transformationssprache BUILT-IN ist). Siehe Beispiel: Aktualisieren bestehender Excel-Blätter[764].
- Unterstützung für das EDIFACT D.19A- und D19B-Verzeichnis (siehe EDI[460]).
- Bei Erstellung von Anmeldeinformationen vom Typ OAuth 2.0 können Sie konfigurieren, dass die Autorisierungsinformationen in den POST Request Body eingefügt werden. Dies ist eine optionale Methode zusätzlich zu der bereits unterstützten Standardmethode, bei der Autorisierungsinformationen im POST Request Header bereitgestellt werden, siehe OAuth 2.0-Anmeldeinformationen[1358].
- Interne Aktualisierungen und Optimierungen

Version 2020

- In Mappings können nun auch Webservices, für die eine OAuth 2.0-Autorisierung[1358] erforderlich ist, aufgerufen werden.
- HTTP- oder OAuth-Authentifizierungsinformationen können mit Hilfe von Anmeldeinformationsobjekten auf andere Ausführungsumgebungen übertragen werden, siehe Anmeldeinformationen[1353].
- Unterstützung für das Mappen von Daten von oder auf JSON-Dateien mit Hilfe des JSON Lines[829]-Textformats
- Unterstützung für Visual Studio 2019 im MapForce Plug-in für Visual Studio[1492] und bei der Codegenerierung[1563].
- Unterstützung für Eclipse 4.9 - 4.12, siehe MapForce Plug-in für Eclipse[1495]
- Wenn ein Oracle-Paket öffentliche gespeicherte Prozeduren oder Funktionen enthält, stehen diese auch für das Mapping zur Verfügung, siehe Hinzufügen von gespeicherten Prozeduren zum Mapping[405]
- Sie können eine Datenbankkomponente so konfigurieren, dass Datenbankobjektnamen als relativ zum Standardschema behandelt werden und nicht an ein bestimmtes Schema gebunden sind. Auf diese Art sparen Sie Zeit, wenn Sie in Zukunft zu einer anderen Datenbank wechseln müssen, siehe Wechseln von Datenbanken und Schemas[263].
- Sie können in MapForce erstellte globale Ressourcen auf FlowForce Server bereitstellen, siehe Bereitstellen von globalen Ressourcen auf FlowForce Server[1480]
- Bei der Ersetzung von Werten mit Hilfe einer Lookup-Tabelle können Tabellendaten (Wert-Schlüssel-Paare) aus externen Quellen wie CSV- oder Excel-Dateien in das Mapping eingefügt werden. Außerdem lassen sich Fälle, in denen ein Wert in der vordefinierten Lookup-Tabelle nicht gefunden wird, einfacher behandeln. Um solche Werte zu verarbeiten, wird die **`substitute-missing`**-Funktion nicht mehr benötigt. Siehe Verwendung von Wertezuordnungen[921].
- Interne Aktualisierungen und Optimierungen

1.1.5 Version 2019

Version 2019 Release 3

- Ein Großteil der grafischen Benutzeroberfläche wurde für Monitore mit hoher Pixeldichte (HiDPI) optimiert.
- Unterstützung für Datenbankablaufverfolgung und Fehlerprotokollierung[443]
- Verbesserte Unterstützung für die Behandlung von Datenbanktransaktionen in generiertem Programmcode und bei der Server-Ausführung. Wenn bei der Aktualisierung einer Datenbank durch ein Mapping ein Fehler auftritt, besteht die Möglichkeit, die Mapping-Ausführung fortzusetzen, auch wenn einige Operationen fehlgeschlagen sind, siehe Behandlung von Datenbankausnahmen[429].
- Webservice-Verbesserungen, darunter die folgenden:
  - Vom Webservice retournierte HTTP-Statuscodes können gemappt werden (gilt für RESTful Webservices).
  - Benutzername, Passwort und URL des Webservice können über das Mapping bereitgestellt werden (gilt sowohl für WSDL- als auch für RESTful-Webservices), siehe Webservice-Call-Einstellungen[631].
  - Möglichkeit, mit Hilfe der neuen mime[1180]-Funktionen den Raw Body (MIME Entity) des Webservice über das Mapping zu erstellen oder den vom Webservice-Aufruf retournierten Raw Body zu mappen. Gilt für RESTful Webservices.
  - Möglichkeit, die Response eines Webservice abhängig vom HTTP-Statuscode und (im Fall von RESTful Webservices) dem Inhaltstyp auf Basis von Bedingungen zu behandeln, siehe Behandlung der HTTP Response auf Basis von Bedingungen[628].
- Mit Hilfe der neuen Funktionen charset-encode[1166] und charset-decode[1164] können Base64-Daten in einem Mapping kodiert oder dekodiert werden.
- Unterstützung für UN/EDIFACT D.18B Directory, siehe EDI[460].
- Unterstützung neuer Datenbankversionen: PostgreSQL 11, Db2 für i 7.2 und 7.3
- Unterstützung für Eclipse 4.7 - 4.10, siehe MapForce Plug-in für Eclipse[1495]
- Unterstützung für die explizite Definition des Java Virtual Machine-Pfads von MapForce aus, siehe Java-Einstellungen[1542]
- Interne Aktualisierungen und Optimierungen

Version 2019

- Daten können nun mit MapForce (und MapForce Server) von oder auf Binärdateien im Protocol Buffers-Format gemappt werden, siehe Protocol Buffers[843].
- Unterstützung neuer Datenbankversionen: Firebird 3.0, Informix 12.10, MariaDB 10.2, MySQL 8.0, PostgreSQL 10, SQL Server 2017, SQL Server für Linux (zusätzlich zur Unterstützung für frühere Versionen)
- Sie können einen Standardwert oder eine Node-Funktion selektiv auf bestimmte Nodes, die Ihren benutzerdefinierten Kriterien entsprechen, anwenden. So können Sie z.B. eine Node-Funktion nur auf Nodes eines bestimmten Typs, deren Name mit einer Regular Expression übereinstimmt, anwenden, siehe Erstellen einer Node-Funktion und Anwendung dieser Funktion auf Basis von Bedingungen[961].
- Sie können Daten bei der Erstellung von Node-Funktionen bedingt auf Basis von Metadaten wie z.B. Node-Name oder Node-Annotation verarbeiten, siehe Bereitstellung von Node-Metadaten für Node-Funktionen[963].

© 2016-2022 Altova GmbH

- Mappings, die Gruppierungsfunktionen enthalten, werden im generierten C#-, C++- oder Java-Code viel schneller ausgeführt.
- Interne Aktualisierungen und Optimierungen

1.2 Was ist MapForce?

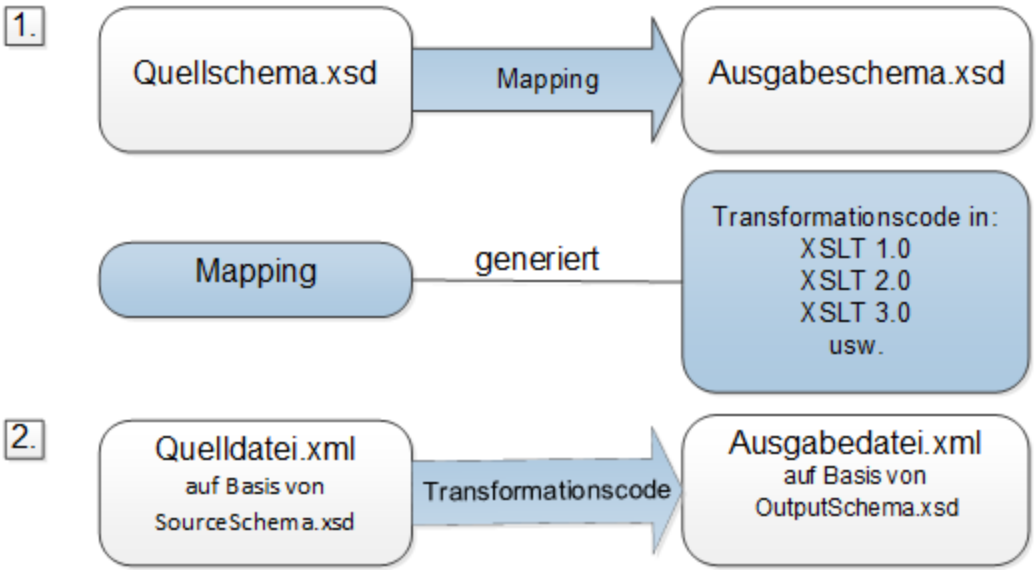
Altova Website: Datenmapping-Tool

MapForce ist ein leistungsstarkes, flexibles Tool zum grafischen Mappen beliebiger unterschiedlicher Datenformate aufeinander. Eine vollständige Liste der verfügbaren Datenformate finden Sie unter Mapping: Quellen und Ziele 27. Mit MapForce können Sie eine Quellkomponente auf eine Zielkomponente, eine Quellkomponente auf mehrere Zielkomponenten, mehrere Quellkomponenten auf eine Zielkomponente oder mehrere Quellkomponenten auf mehrere Zielkomponenten mappen. Nähere Informationen über Mapping-Szenarien finden Sie unter Mapping-Szenarien 29 und in den Tutorials 43. Außerdem bietet MapForce eine breite Palette an Datenverarbeitungs- und Filteroptionen, wie z.B. Funktionen 942, Variablen 882, Filter und Bedingungen 914, mit denen Sie Ihre Daten bearbeiten können.

Um ein Mapping durchführen zu können, müssen Sie eine Datenstruktur zur Beschreibung der Struktur der einzelnen Quell- und Zieldateien bereitstellen. So definiert etwa ein XML-Schema die Struktur eines XML-Dokuments. Das Mapping (der Quell- auf die Zieldaten) erfolgt über eine grafische Benutzeroberfläche mittels Drag-and-Drop. Für das Mapping muss kein Programmcode geschrieben werden, da der Code von MapForce für Sie generiert wird. Mit Hilfe dieses Codes können Sie Dokumente mit einer bestimmten Datenquellstruktur in Dokumente mit einer bestimmten Datenzielstruktur transformieren.

Abstraktes Modell

Im unten gezeigten abstrakten Modell wird eines der grundlegende Szenarien einer Datentransformation in MapForce dargestellt. Im ersten Schritt wird eine abstrakte Struktur namens **SourceSchema.xsd** auf eine andere abstrakte Struktur namens **Output Schema.xsd** gemappt. Das Mapping generiert Transformationscode in der ausgewählten Transformationssprache. Im zweiten Schritt wird der Inhalt der auf **SourceSchema.xsd** basierenden XML-Quelldatei auf die auf **OutputSchema.xsd** basierende XML-Zieldatei gemappt. Das Mapping des Inhalts von der Quell- auf die Zieldatei wird mit Hilfe des im vorherigen Schritt generierten Transformationscodes ausgeführt.



Alle Editionen von MapForce stehen als 32-Bit-Applikation zur Verfügung. Die MapForce Professional und die Enterprise Edition stehen zusätzlich dazu auch als 64-Bit-Applikation zur Verfügung.

© 2016-2022 Altova GmbH

Konventionen

Die in diesem Handbuch dargestellten oder referenzierten Mapping-Dateien befinden sich in den folgenden Ordnern:

- `C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForce Examples`
- `C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForceExamples\Tutorials.`
- `C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForceExamples\Tutorials\BasicTutorials`

In diesem Abschnitt

Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

- Mapping: Quellen und Ziele 27
- Transformationssprachen 28
- Mapping-Szenarien 29
- Integration mit Altova-Produkten 31

1.2.1 Mapping: Quellen und Ziele

Mit den wichtigen Begriffen *Quelle* und *Ziel* werden in MapForce Datenstrukturen bezeichnet, von bzw. auf die Daten gemappt werden. Unten finden Sie eine Liste der Technologien, die als Mapping-Quellen und -Ziele verwendet werden können.

MapForce Basic Edition

- XML und XML-Schema

MapForce Professional Edition

- XML und XML-Schema
- Flat Files, einschließlich der Formate CSV (Comma Separated Values = kommagetrennte Werte) und FLF (Fixed-Length Field = Felder mit fester Länge)
- Datenbanken (alle gebräuchlichen relationalen Datenbanken einschließlich Microsoft Access- und SQLite-Datenbanken)
- Binärdateien (BLOB-Rohinhalte)

MapForce Enterprise Edition

- XML und XML-Schema
- Flat Files, einschließlich der Formate CSV (Comma Separated Values = kommagetrennte Werte) und FLF (Fixed-Length Field = Felder mit fester Länge)
- Daten aus Altdatenbeständen aus Textdateien können mittels MapForce FlexText auf andere Formate gemappt und konvertiert werden;
- SQL-Datenbanken: alle gebräuchlichen relationalen Datenbanken einschließlich Microsoft Access- und SQLite-Datenbanken
- NoSQL-Datenbanken
- Binärdateien (BLOB-Rohinhalte)
- EDI-Formatfamilie. Dazu gehören UN/EDIFACT, ANSI X12, HL7, IATA PADIS, SAP IDoc, TRADACOMS

- JSON-Dateien
- Microsoft Excel-Dateien ab Version 2007
- XBRL-Instanzdateien und Taxonomien
- Protocol Buffer

1.2.2 Transformationssprachen

Mit Hilfe einer Transformationssprache wird in MapForce Transformationscode zur Ausführung von Mappings generiert. Sie können eine Transformationssprache jederzeit auswählen/wechseln. MapForce ermöglicht die Anzeige des Transformationscodes in der ausgewählten Sprache. Nähere Informationen dazu finden Sie unter Codegenerierung[109]. Sie können diesen Code auch mit dem Menübefehl **Datei | Code generieren in** generieren und damit jedes Datendokument, das gemäß dem Schema der Quellkomponente gültig ist, transformieren. Je nach MapForce Edition stehen die folgenden Sprachen für Ihre Datentransformationen zur Verfügung:

MapForce Basic Edition	MapForce Professional und Enterprise Edition
• XSLT 1.0 • XSLT 2.0 • XSLT 3.0	• XSLT 1.0 • XSLT 2.0 • XSLT 3.0 • BUILT-IN • XQuery • Java • C# • C++

Zur Auswahl der Transformationssprache haben Sie folgende Möglichkeiten:

- Klicken Sie im Menü **Ausgabe** auf den Namen der Sprache, die Sie für die Transformation verwenden möchten.
- Klicken Sie in der **Sprachauswahl**-Symbolleiste (*Abbildung unten*) auf den Namen der Sprache.



Wenn Sie die Transformationssprache des Mappings wechseln, kann es vorkommen, dass bestimmte MapForce-Funktionalitäten für diese Sprache nicht unterstützt werden. Nähere Informationen dazu finden Sie unter Anmerkungen zur Unterstützung[1982].

MapForce validiert bei der Erstellung von Mappings bzw. bei der Erstellung der Mapping-Vorschau die Gültigkeit Ihrer Schemas oder Transformationen. Wenn Validierungsfehler auftreten, werden diese von MapForce im Fenster "Meldungen"[38] angezeigt. Dies ist hilfreich, weil Sie die Fehler dadurch sofort überprüfen und korrigieren können.

Transformationssprachen in der MapForce Professional und Enterprise Edition
Wenn Sie Java, C# oder C++ als Transformationssprache auswählen, generiert MapForce die erforderlichen Projekte und Projektmappen, sodass Sie diese direkt in Visual Studio oder Eclipse öffnen können. Sie können das generierte Programm bei komplexeren Datenintegrationsszenarien mit Hilfe der Altova-Bibliotheken und der MapForce API auch durch eigenen Code ergänzen.

© 2016-2022 Altova GmbH

BUILT-IN
Bei Auswahl der Option BUILT-IN als Transformationssprache für Ihr Mapping wird zum Ausführen des Datenmappings der native MapForce-Transformationsprozessor verwendet. Bei Anzeige einer Vorschau auf die Mapping-Ausgabe verwendet MapForce diese Option auch implizit, wenn als Transformationssprache Java, C# oder C++ ausgewählt ist.

Der BUILT-IN-Prozessor verarbeitet Mappings, ohne dafür externe Prozessoren zu verwenden, was vor allem dann ratsam ist, wenn der Arbeitsspeicher knapp bemessen ist. Wenn kein Programmcode in einer bestimmten Sprache generiert werden muss, verwenden Sie BUILT-IN als die Standardoption, da diese im Vergleich zu anderen Sprachen die meisten MapForce-Funktionalitäten unterstützt (siehe Anmerkungen zur Unterstützung[1982]. Außerdem kann das Mapping bei Auswahl von BUILT-IN als Transformationssprache mit MapForce Server automatisiert werden. Nähere Informationen dazu finden Sie unter Automatisieren mit Altova-Produkten[1432].

1.2.3 Mapping-Szenarien

Die Szenarien können sich nach den folgenden Kriterien unterscheiden: (i) nach Quellen und Zielen und (ii) der Komplexität von Mappings. Als Quellen und Ziele können unterschiedliche Datenstrukturen verwendet werden, z.B. ein XML-Schema, eine XML-Datei mit einem dieser zugewiesenen Schema, Datenbanken usw. Nähere Informationen über unterstützte Formate von Quellen und Zielen finden Sie unter Mapping: Quellen und Ziele[27] .

Die folgenden Szenarien sind nur einige Beispiele für die Komplexität von Mapping-Designs:

- Mappen einer Quellkomponente auf eine Zielkomponente. Nähere Informationen zu dieser Art von Mapping finden Sie im Tutorial 1[44].
- Zusammenführen mehrerer Datenquellen in einer Zielstruktur. Nähere Informationen dazu finden Sie im Tutorial 2[54].
- Filtern der Daten auf eine Weise, dass nur eine Untergruppe dieser Daten auf die Zieldatei gemappt wird. Siehe Tutorial 3[61].
- Mappen von Struktur und Inhalt der Quell- auf die Zieldatei. Siehe Tutorial 4[68].

Unabhängig davon, mit welcher Technologie Sie arbeiten, ermittelt MapForce die Struktur Ihrer Daten normalerweise automatisch oder schlägt vor, dass Sie ein Schema für Ihre Daten bereitstellen. MapForce kann auch anhand einer Beispielinstanzdatei ein Schema generieren. Wenn Sie z.B. eine XML-Instanzdatei, aber keine Schemadefinition haben, kann MapForce eine für Sie generieren. Auf diese Art kann MapForce die Daten aus der XML-Datei für das Mappen auf andere Dateien oder Formate zur Verfügung stellen. Nähere Informationen über die Grundbegriffe und wichtigsten Funktionen von MapForce finden Sie unter Grundlegende Aufgaben[76] und Übersicht über die Benutzeroberfläche[33].

Um Ihre Datenmapping-Designs leichter aufrufen und verwalten zu können, können Sie diese In Mapping-Projekten organisieren. Diese Funktionalität steht in der MapForce Professional und der Enterprise Edition zur Verfügung. Code kann nicht nur für einzelne Mappings in Ihrem Projekt, sondern auch anhand eines gesamten Projekts generiert werden.

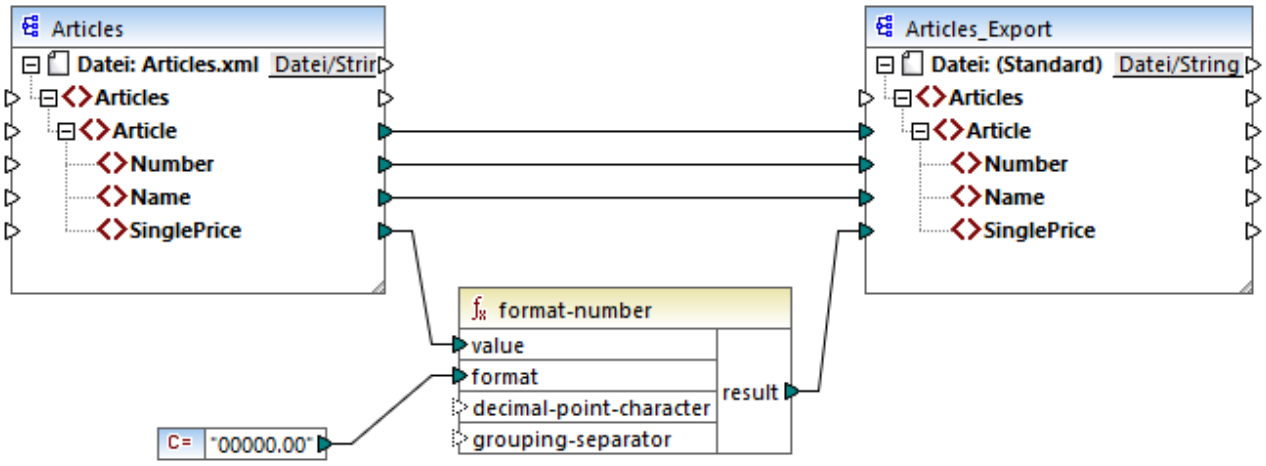
Nicht nur Layout und Darstellung der Entwicklungsumgebung, sondern auch verschiedene andere Einstellungen für die einzelnen Technologien und Mapping-Komponententypen können in MapForce je nach Bedarf angepasst werden. Zum Beispiel:

- Sie können einstellen, ob beim Mappen von oder auf XML (i) eine Schemareferenz inkludiert werden soll oder (ii) ob die XML-Deklaration in den XML-Ausgabedateien unterdrückt werden soll. Sie können auch die Kodierung der generierten Dateien auswählen (z.B. UTF-8).
- Beim Mappen von oder auf Datenbanken könne Sie Einstellungen wie den Timeout-Wert für die Ausführung von Datenbankanweisungen definieren. Außerdem können Sie auswählen, ob MapForce Datenbanktransaktionen verwenden soll oder ob der Datenbankschemaname beim Generieren von Code aus Tabellennamen entfernt werden soll.
- Bei XBRL können Sie die Strukturansichten auswählen, die in MapForce angezeigt werden sollen (z.B. die **Presentation und Definition Linkbases**-Ansicht, die **Tabellen-Linkbase**-Ansicht oder die **Alle Concepts**-Ansicht).

In den Beispielen unten werden Mapping-Designs gezeigt, in denen dieselbe (*Beispiel 1*) und unterschiedliche Arten (*Beispiel 2*) von Quell- und Zielstrukturen verwendet werden. Beide Mapping-Beispiele sind einfach gehalten, es wird jeweils nur eine Datenquelle und ein Datenziel verwendet. Nähere Informationen zu komplexen Mappings, Aufgaben und Verfahren, finden Sie unter Komplexe Mapping-Szenarien[1298].

Beispiel 1: XML-Mapping

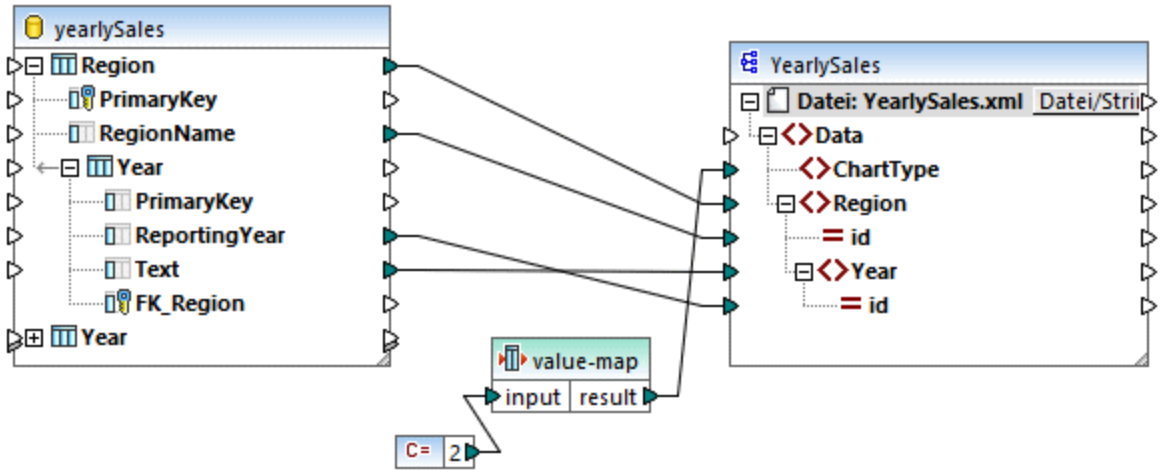
Das Design aller Mapping-Transformationen erfolgt in MapForce visuell. So können Sie etwa im Fall von XML jedes Element und Attribut in einer XML-Datei mit einem Element oder Attribut einer anderen XML-Datei verbinden. Dadurch wird MapForce angewiesen, Daten aus dem Quell-Node auszulesen und in den Ziel-Node zu schreiben. Unten sehen Sie die Transformation einer XML-Datei in eine andere XML-Datei.



Beispiel 2: Datenbank-Mapping

Beim Arbeiten mit Datenbanken in der MapForce Professional oder Enterprise Edition werden alle Datenbankspalten im MapForce Mapping-Bereich angezeigt. Hier können Sie Daten durch Ziehen von Verbindungslinien von oder auf diese Daten mappen. Wie bei anderen Altova MissionKit-Produkten können Sie beim Einrichten der Datenbankverbindung über MapForce den Datenbanktreiber und Verbindungstyp (ADO, ADO.NET, ODBC oder JDBC) ganz nach Bedarf und je nach Ihrer Infrastruktur flexibel auswählen. Zusätzlich dazu können Sie visuell SQL-Abfragen erstellen, gespeicherte Prozeduren verwenden oder eine Datenbank direkt abfragen (die Unterstützung variiert je nach Datenbanktyp, Edition und Treiber). Unten sehen Sie ein Beispiel für eine Datentransformation von einer Datenbank in eine XML-Datei:

© 2016-2022 Altova GmbH



1.2.4 Integration mit Altova-Produkten

Mit Hilfe der integrierten XSLT/XQuery-Prozessoren können Transformationen innerhalb von MapForce durchgeführt werden. MapForce kann auch in Kombination mit anderen Altova-Produkten verwendet werden (*siehe unten*).

XMLSpy
Wenn XMLSpy auf demselben Rechner installiert ist, können Sie alle unterstützten Dateitypen direkt vom jeweiligen MapForce-Kontext aus in XMLSpy öffnen und bearbeiten. So steht z.B. bei Klick auf eine XML-Komponente der Menübefehl **Komponente | Schema-Definition in XMLSpy bearbeiten** zur Verfügung.
RaptorXML Server
Sie können den generierten XSLT-Code direkt in MapForce ausführen und dort sofort eine Vorschau des Ergebnisses der Datentransformation anzeigen. Um die Verarbeitung schneller auszuführen, können Sie das Mapping auch mit RaptorXML Server, einem ultraschnellen XML-Transformationsprozessor, ausführen.

MapForce Server (Enterprise und Professional Edition)
Mit Hilfe von Altova MapForce Server, der auf Windows-, Linux- und macOS-Betriebssystemen installiert werden kann, lassen sich MapForce-Aufgaben automatisieren. Mit MapForce Server können Sie die in einem Mapping definierten Transformationen nicht nur über die Befehlszeile des jeweiligen Betriebssystems, sondern auch über API-Aufrufe (.NET, COM, Java) ausführen.
FlowForce Server (Enterprise und Professional Edition)
Mit Hilfe von Altova FlowForce Server, der auf Windows-, Linux- und macOS-Betriebssystemen installiert werden kann, lassen sich MapForce-Aufgaben ebenfalls automatisieren. Mit FlowForce Server können MapForce Server-Aufgaben nach einem Zeitplan ausgeführt werden.
StyleVision (Enterprise und Professional Edition)
Mit Hilfe von StyleVision können Sie StyleVision Power Stylesheets erstellen oder vorhandene wiederverwenden und eine Vorschau des Ergebnisses der Mapping-Transformationen als HTML-, RTF-, PDF- oder Word 2007+-Dokumente anzeigen.

MapForce als Plug-in
Die MapForce Professional und Enterprise Edition können als Plug-in der IDEs Visual Studio und Eclipse installiert werden. Auf diese Art können Sie Mappings erstellen und erhalten Zugriff auf die MapForce-Funktionalitäten, ohne die Entwicklungsumgebung Ihrer Wahl verlassen zu müssen.

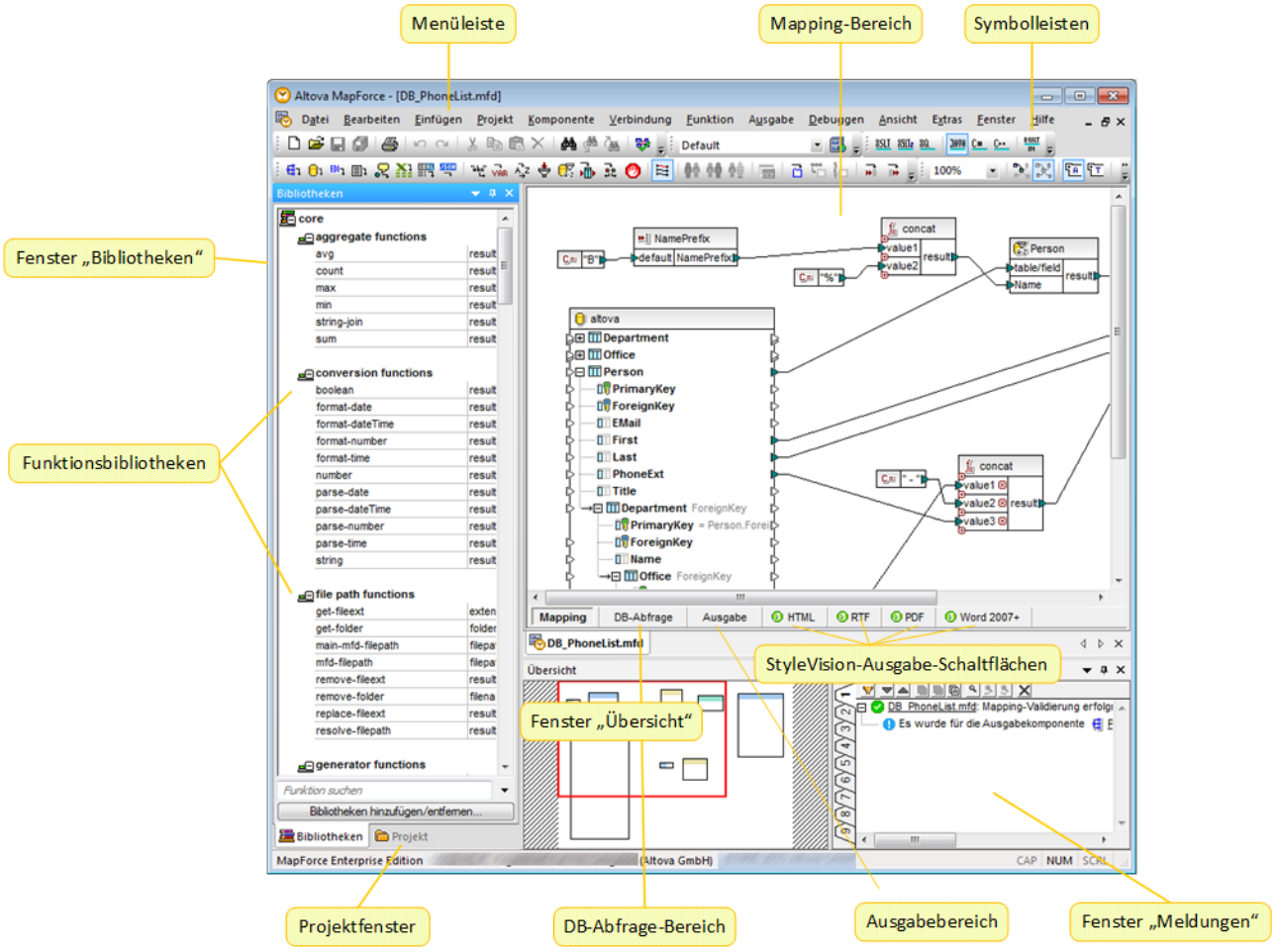
Nähere Informationen zur Automatisierung von Aufgaben finden Sie unter Automatisieren von MapForce-Aufgaben mit Altova-Produkten[1432]. Nähere Informationen über die Verwendung von MapForce als Plug-in finden Sie unter Plug-in für Visual Studio[1492] und Plug-in für Eclipse[1495].

© 2016-2022 Altova GmbH

1.3 Übersicht über die Benutzeroberfläche

Die grafische Benutzeroberfläche von MapForce ist als integrierte Entwicklungsumgebung konzipiert. In der Abbildung unten sehen Sie die wichtigsten Komponenten der Benutzeroberfläche. Mit Hilfe des Menübefehls **Extras | Anpassen** können Sie die Einstellungen der Benutzeroberfläche ändern. Mit Hilfe der Schaltflächen in der rechten oberen Ecke jedes Fensters können Sie das Fenster ein- und ausblenden sowie ab- und andocken. Um die Symbolleisten und Fenster wieder in Ihren ursprünglichen Zustand zurückzusetzen, verwenden Sie den Menübefehl **Extras | Symbolleisten und Fenster wiederherstellen**.

In der nachstehenden Abbildung sind die wichtigsten Bereiche der grafischen Benutzeroberfläche von MapForce dargestellt.



Nähere Informationen über die Features und Funktionen der einzelnen Bereiche finden Sie in den entsprechenden Kapiteln weiter unten.

In diesem Abschnitt

Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

- Leisten 34

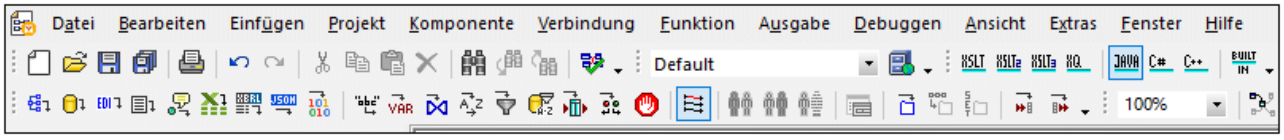
- Fenster 34
- Fenster "Meldungen" 38
- Bereiche 39

1.3.1 Leisten

In diesem Kapitel finden Sie eine Übersicht über die verfügbaren Leisten.

Menüleiste und Symbolleisten

In der **Menüleiste** werden die Menübefehle angezeigt. Jede Symbolleiste enthält eine Reihe von Schaltflächen für verschiedene MapForce-Befehle. Sie können die Symbolleisten an ihren Ziehpunkten mit der Maus an die gewünschte Stelle ziehen. In der Abbildung unten sehen Sie die **Menüleiste** und die Symbolleisten. Was genau auf der Benutzeroberfläche angezeigt wird, hängt von Ihrer MapForce Edition und den gewünschten Einstellungen ab.



Applikationsstatusleiste

Die Applikationsstatusleiste wird am unteren Rand des MapForce-Fensters angezeigt und enthält Informationen zur Applikation. Wenn Sie die Maus über eine Symbolleisten-Schaltflächen platzieren, werden Tooltips dazu angezeigt. Wenn Sie die 64-Bit-Version von MapForce verwenden, wird in der Statusleiste der Applikationsname mit dem Suffix x64 angezeigt. Die 32-Bit-Version hat kein Suffix.

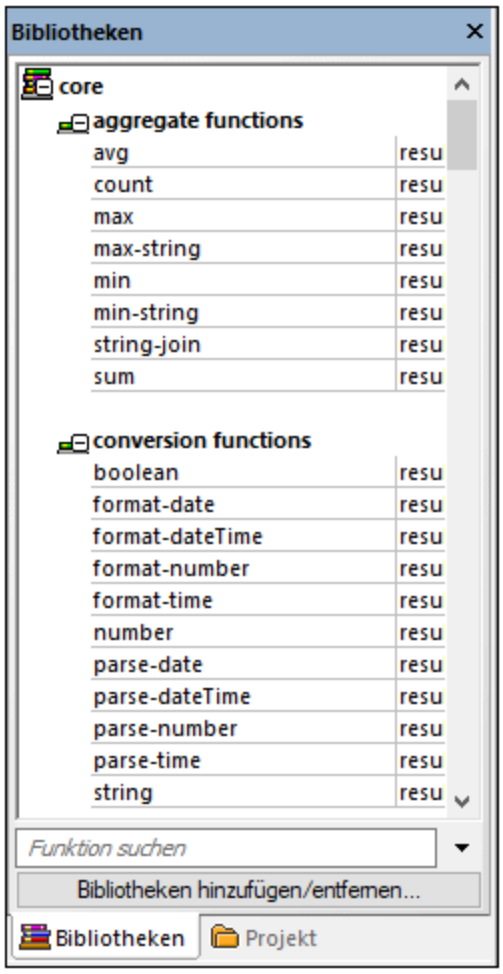
1.3.2 Fenster

In diesem Kapitel finden Sie eine Übersicht über die verfügbaren Fenster.

Fenster "Bibliotheken"

Im Fenster **Bibliotheken** wird eine Liste der vordefinierten MapForce-Funktionen, geordnet nach Bibliotheken, angezeigt. Je nachdem, welche Transformationssprache Sie entweder über das Menü **Ausgabe** oder über die **Sprachauswahl**-Symbolleiste auswählen, stehen unterschiedliche Funktionen zur Verfügung. Nähere Informationen dazu finden Sie unter Transformationssprachen 28. Wenn Sie benutzerdefinierte Funktionen erstellt oder externe Bibliotheken importiert haben, so werden auch diese im Fenster **Bibliotheken** angezeigt.

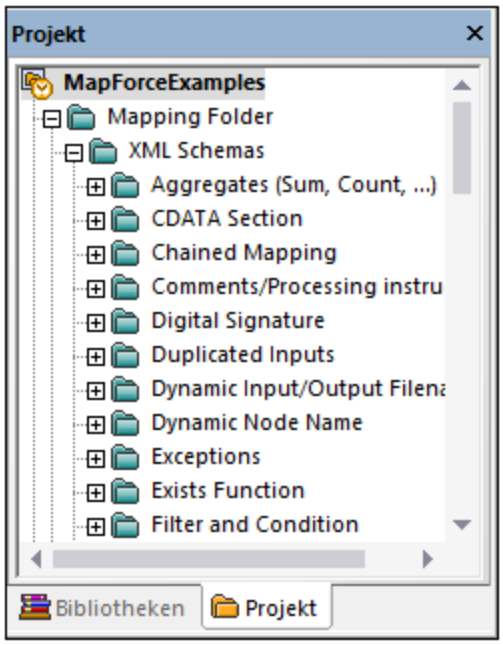
© 2016-2022 Altova GmbH



Um eine Funktion nach Name oder Beschreibung zu suchen, geben Sie den Suchwert in das Textfeld am unteren Rand des Fensters **Bibliotheken** ein. Um alle Instanzen einer Funktion (im gerade aktiven Mapping) zu suchen, klicken Sie mit der rechten Maustaste auf die Funktion und wählen Sie im Kontextmenü den Befehl **Alle Aufrufe suchen** aus. Sie können den Datentyp der Funktion und ihre Beschreibung auch direkt im Fenster **Bibliotheken** anzeigen. Nähere Informationen dazu finden Sie unter Funktionen[942].

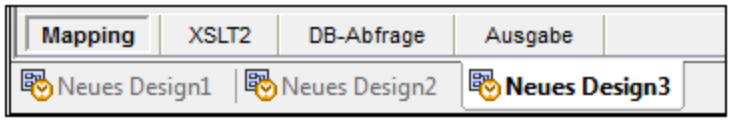
Projektfenster (Enterprise und Professional Edition)

MapForce unterstützt das Multiple Document Interface und gestattet Ihnen, Ihre Mappings in Mapping-Projekten zu gruppieren. Im **Projektfenster** sehen Sie alle Dateien und Ordner, die zum Projekt hinzugefügt wurden. Projektdateien haben die Dateierweiterung **`*.mfp`** (MapForce-Projekt). Um in Projekten nach Mappings zu suchen, klicken Sie an eine beliebige Stelle im **Projektfenster** und drücken Sie **Strg + F**. Nähere Informationen dazu finden Sie unter Projekte[119].



Mapping-Fenster

In MapForce wird ein Multiple Document Interface (MDI) verwendet. Jede in MapForce geöffnete Mapping-Datei hat ein eigenes Fenster. Auf diese Art können Sie mit mehreren Mapping-Fenstern arbeiten und diese auf verschiedene Arten im Hauptfenster von MapForce anordnen und in der Größe anpassen. Sie können alle offenen Fenster auch nach den Windows-Standard-Layouts "Horizontal anordnen", "Vertikal anordnen" und "Überlappend" anordnen. Wenn mehrere Mappings in MapForce geöffnet sind, können Sie über die im unteren Bereich des **Mapping**-Fensters angezeigten Register jederzeit zwischen den Mappings wechseln (*siehe Abbildung unten*).



Die Optionen zur Verwaltung der Fenster können über den Menübefehl **Fenster | Fenster** aufgerufen werden. Über das Dialogfeld **Fenster** können Sie an jedem bzw. allen der derzeit offenen Mapping-Fenstern Aktionen wie Speichern, Schließen oder Minimieren vornehmen. Um mehrere Fenster im Dialogfeld **Fenster** auszuwählen, klicken Sie auf die gewünschten Einträge, während Sie die **Strg**-Taste gedrückt halten.

Fenster "Bibliotheken verwalten"

Über dieses Fenster können Sie alle benutzerdefinierten Funktionen (UDFs = user-defined functions) und importierten benutzerdefinierten Bibliotheken (einschließlich kompilierter Java .class-Dateien und .NET DLL Assembly-Dateien), die in derzeit geöffneten Mappings verwendet werden, anzeigen und verwalten.

Standardmäßig wird das Fenster **Bibliotheken verwalten** nicht angezeigt. Um es anzuzeigen, wählen Sie eine der folgenden Methoden:

© 2016-2022 Altova GmbH

- Klicken Sie im Menü **Ansicht** auf **Bibliotheken verwalten**.
- Klicken Sie im unteren Bereich des Fensters **Bibliotheken** auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**.



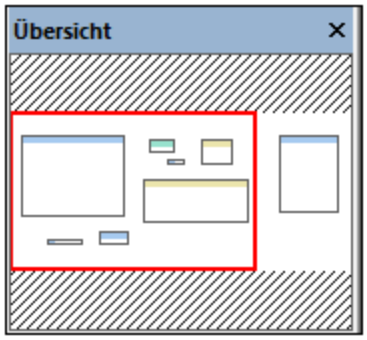
Sie können auswählen, ob benutzerdefinierte Funktionen (UDFs) und Bibliotheken nur für das gerade aktive Mapping-Dokument oder für alle geöffneten Mapping-Dokumente angezeigt werden sollen. Um die importierten Funktionen und Bibliotheken für alle gerade offenen Mapping-Dokumente anzuzeigen, klicken Sie mit der rechten Maustaste in das Fenster und wählen Sie im Kontextmenü den Befehl **Offene Dokumente anzeigen**.

Um anstelle des Namens den Pfad des geöffneten Mapping-Dokuments anzuzeigen, klicken Sie mit der rechten Maustaste in das Fenster und wählen Sie im Kontextmenü den Befehl **Dateipfade anzeigen**.

Nähere Informationen dazu finden Sie unter Verwalten von Funktionsbibliotheken. 946.

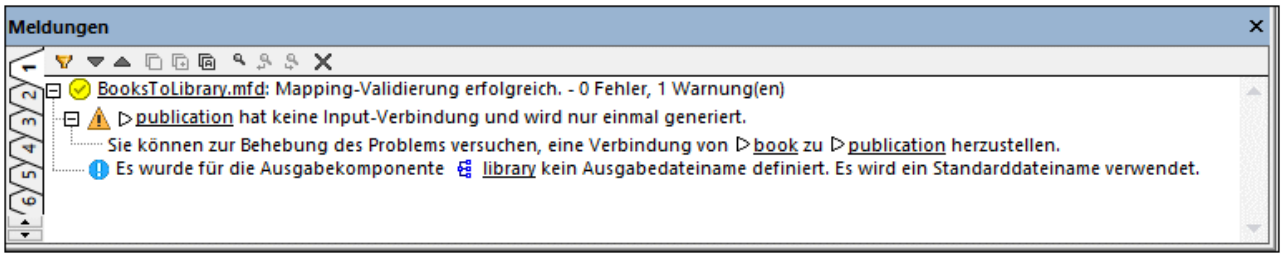
Fenster "Übersicht"

Im Fenster **Übersicht** sehen Sie eine Gesamtübersicht über den Mapping-Bereich 39. Wenn das Mapping sehr groß ist, können Sie hier schnell zu einer bestimmten Stelle im Mapping-Bereich navigieren. Klicken Sie dazu auf das rote Rechteck und ziehen Sie es an die gewünschte Stelle.



1.3.3 Fenster "Meldungen"

Im Fenster **Meldungen** (*siehe Abbildung unten*) werden bei der Mapping-Vorschau oder bei Durchführung einer Mapping-Validierung[107] der Validierungsstatus, Meldungen, Fehler und/oder Warnungen angezeigt. Klicken Sie auf den unterstrichenen Text im Fenster **Meldungen**, um zur Komponente bzw. Struktur zu gelangen, die die Information, Warnung oder Fehlermeldung verursacht hat.



Symbole für den Validierungsstatus

Bei der Validierung eines Mappings überprüft MapForce auf nicht unterstützte Komponentenarten, falsche oder fehlende Verbindungen. Das Validierungsergebnis wird im **Fenster Meldungen** mit einem der folgenden Statussymbole angezeigt:

Symbol	Bedeutung
	Die Validierung war erfolgreich.
	Die Validierung wurde abgeschlossen. Es wurden Warnungen ausgegeben.
	Die Validierung ist fehlgeschlagen.

Zusätzlich dazu werden im Fenster **Meldungen** eventuell die folgenden Arten von Meldungen angezeigt: Informationen, Warnungen und Fehler.

Symbol	Bedeutung
	Kennzeichnet eine Informationsmeldung. Bei Informationsmeldungen wird die Mapping-Ausführung nicht gestoppt.
	Kennzeichnet eine Warnmeldung. Bei Warnungen wird die Mapping-Ausführung nicht gestoppt. Sie werden z.B. angezeigt, wenn keine Verbindungen zu obligatorischen Input-Konnektoren erstellt wurden. In solchen Fällen wird für diejenigen Komponenten, die eine gültige Verbindung haben, dennoch eine Ausgabe generiert.
	Kennzeichnet einen Fehler. Bei einem Fehler schlägt die Mapping-Ausführung fehl und es wird keine Ausgabe generiert. Es kann keine Vorschau auf den XSLT- oder XQuery-Code generiert werden.

Um die Komponente oder Struktur, die die Informations-, Warn- oder Fehlermeldung verursacht hat, zu markieren, klicken Sie im Fenster **Meldungen** auf den unterstrichenen Text.

© 2016-2022 Altova GmbH

Aktionen im Zusammenhang mit Meldungen

Im Fenster **Meldungen** können die folgenden Aktionen durchgeführt werden:

Symbol	Beschreibung
	Filtert Meldungen nach ihrem Schweregrad (Informationsmeldungen, Fehler, und Warnungen). Wählen Sie **Alle aktivieren,** um alle Schweregrade zu inkludieren (dies ist die Standardeinstellung). Wählen Sie **Alle deaktivieren,** um alle Schweregrade aus dem Filter zu entfernen. In diesem Fall wird nur eine Meldung über den allgemeinen Ausführungs- oder Validierungsstatus angezeigt.
	Geht zur nächsten Zeile.
	Geht zur vorherigen Zeile.
	Kopiert die ausgewählte Zeile in die Zwischenablage.
	Kopiert die ausgewählte Zeile einschließlich aller untergeordneten Zeilen in die Zwischenablage.
	Kopiert den gesamten Inhalt des Fensters **Meldungen** in die Zwischenablage.
	Sucht im Fenster **Meldungen** nach einem bestimmten Text. Um optional nur nach Wörtern zu suchen, wählen Sie die Option **Ganzes Wort**. Um bei der Textsuche die Groß- und Kleinschreibung zu berücksichtigen, wählen Sie die Option **GROSS/klein beachten**.
	Sucht ab der aktuell ausgewählten Zeile bis zum Ende nach einem bestimmten Text.
	Sucht ab der aktuell ausgewählten Zeile bis zum Anfang nach einem bestimmten Text.
	Löscht die Meldungen im Fenster "Meldungen".

Wenn Sie gleichzeitig mit mehreren Mapping-Fenstern arbeiten, ist es sinnvoll, die Informationen, Warnungen und Fehlermeldungen für jedes Mapping auf einem eigenen Register anzuzeigen. Klicken Sie in diesem Fall auf die nummerierten Register auf der linken Seite des Fensters **Meldungen**, bevor Sie das Mapping validieren.

1.3.4 Bereiche

In diesem Kapitel finden Sie eine Übersicht über die verfügbaren Bereiche.

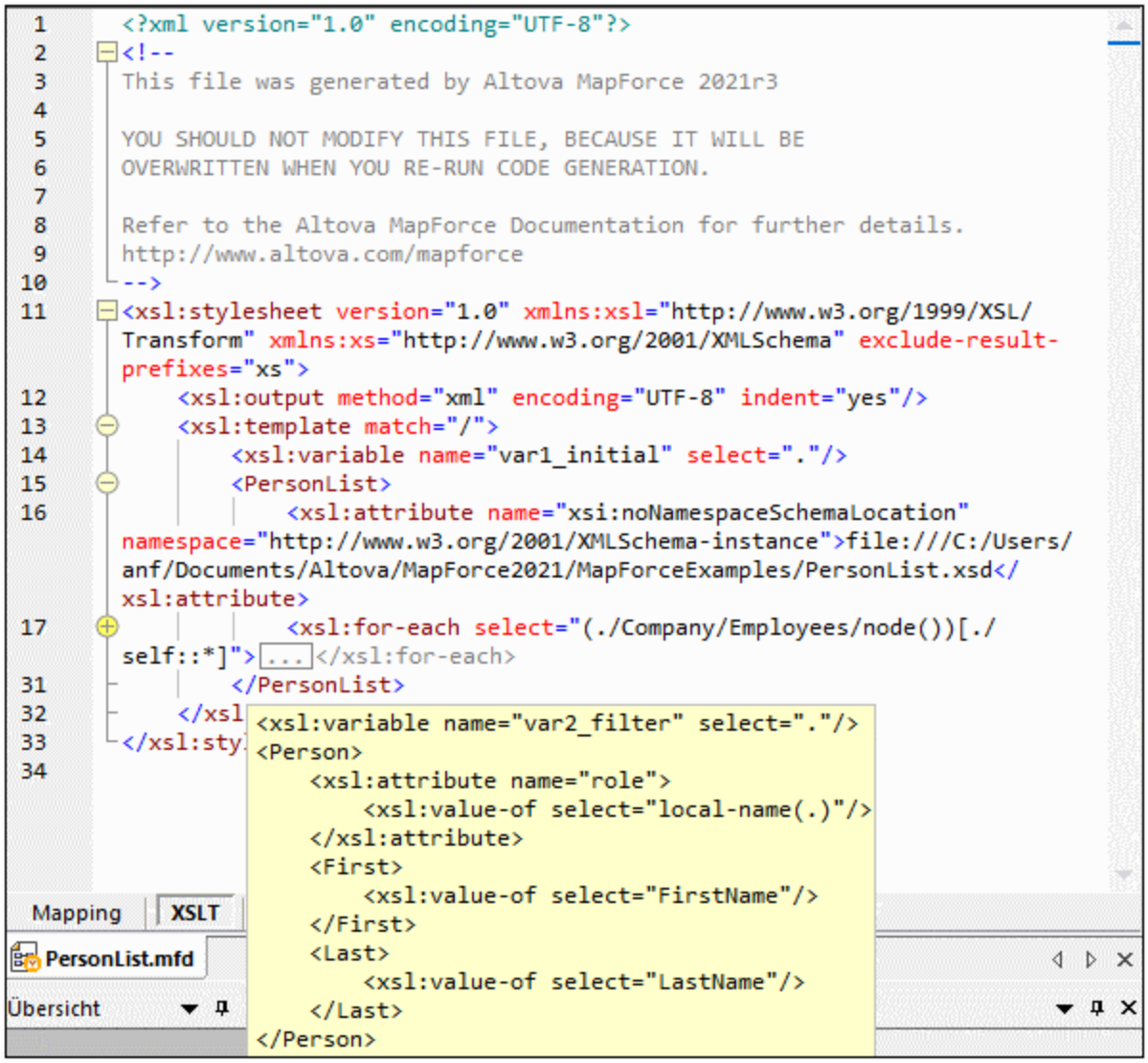
Mapping-Bereich

Der **Mapping**-Bereich ist der Arbeitsbereich, in dem Sie Mappings[107] erstellen. Über das Menü **Einfügen** können Sie Mapping-Komponenten (wie z.B. Dateien, Schemas, Konstanten, Variablen usw.) zum Mapping-Bereich hinzufügen. Nähere Informationen dazu finden Sie unter Hinzufügen von Komponenten zum Mapping[80]. Sie können auch Funktionen aus dem Fenster **Bibliotheken** in den **Mapping**-Bereich ziehen. Nähere Informationen dazu finden Sie unter Hinzufügen einer Funktion zum Mapping[943].

XSLT-Bereich

Im **XSLT**-Bereich wird der anhand Ihres Mappings generierte XSLT-Transformationscode angezeigt. Um zu diesem Fenster zu wechseln, wählen Sie als Transformationssprache[28] XSLT, XSLT2 oder XSLT3 aus und klicken Sie auf das Register mit dem entsprechenden Namen.

Dieses Fenster bietet Funktionalitäten zur Anzeige von Zeilennummern und zum Ein- und Ausklappen von Codeabschnitten. Um Codeabschnitte ein- oder auszuklappen, klicken Sie auf das "+" bzw."-"-Symbol am linken Fensterrand. Eingeklappte Codeabschnitte werden mittels Auslassungspunkten markiert. Um eine Vorschau des eingeklappten Abschnitts zu sehen, ohne diesen Abschnitt ausklappen zu müssen, platzieren Sie die Mauszeiger über die Auslassungspunkte. Daraufhin wird ein Tooltip mit der Codevorschau angezeigt, wie in der Abbildung unten gezeigt. Wenn der Abschnitt zu groß für die Vorschau ist, wird am Ende des Tooltips ein weiteres Auslassungssymbol angezeigt.



Um die Anzeigeeinstellungen einschließlich Einrückung, Zeilenendemarkierungen und anderen zu konfigurieren, klicken Sie mit der rechten Maustaste in den Bereich und wählen Sie im Kontextmenü den

© 2016-2022 Altova GmbH

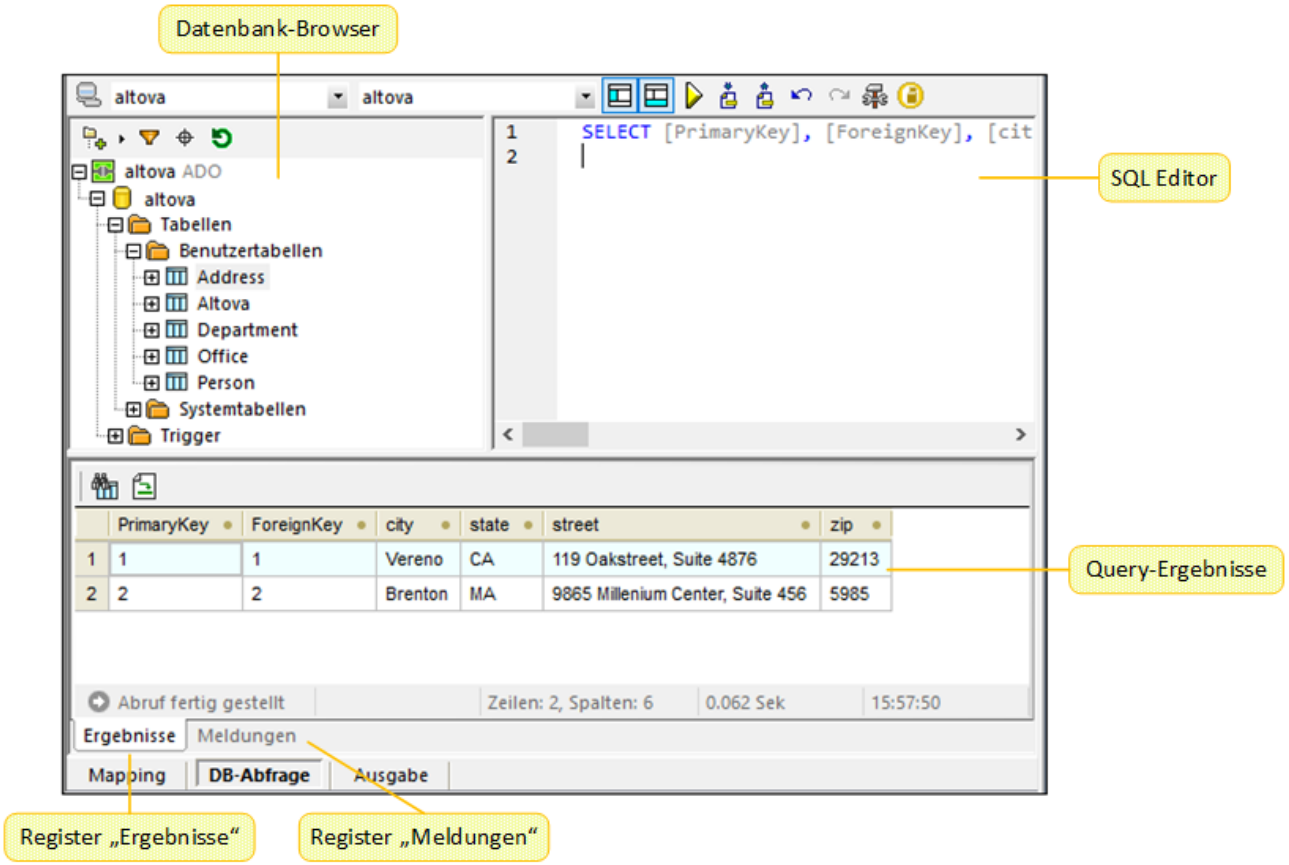
Befehl **Einstellungen für Textansicht** aus. Alternativ dazu können Sie auch auf die Symbolleisten-Schaltfläche (**Einstellungen für Textansicht**) klicken.

XQuery-Bereich (Enterprise und Professional Edition)

Im **XQuery**-Bereich wird der anhand Ihres Mappings generierte XQuery-Transformationscode angezeigt, wenn Sie auf die Schaltfläche **XQuery** klicken. Dieser Bereich steht zur Verfügung, wenn Sie XQuery als Transformationssprache auswählen. Dieser Bereich enthält auch eine Zeilennummerierung und eine Klappleiste, die ähnlich wie im Bereich "XSLT" (siehe oben) funktioniert.

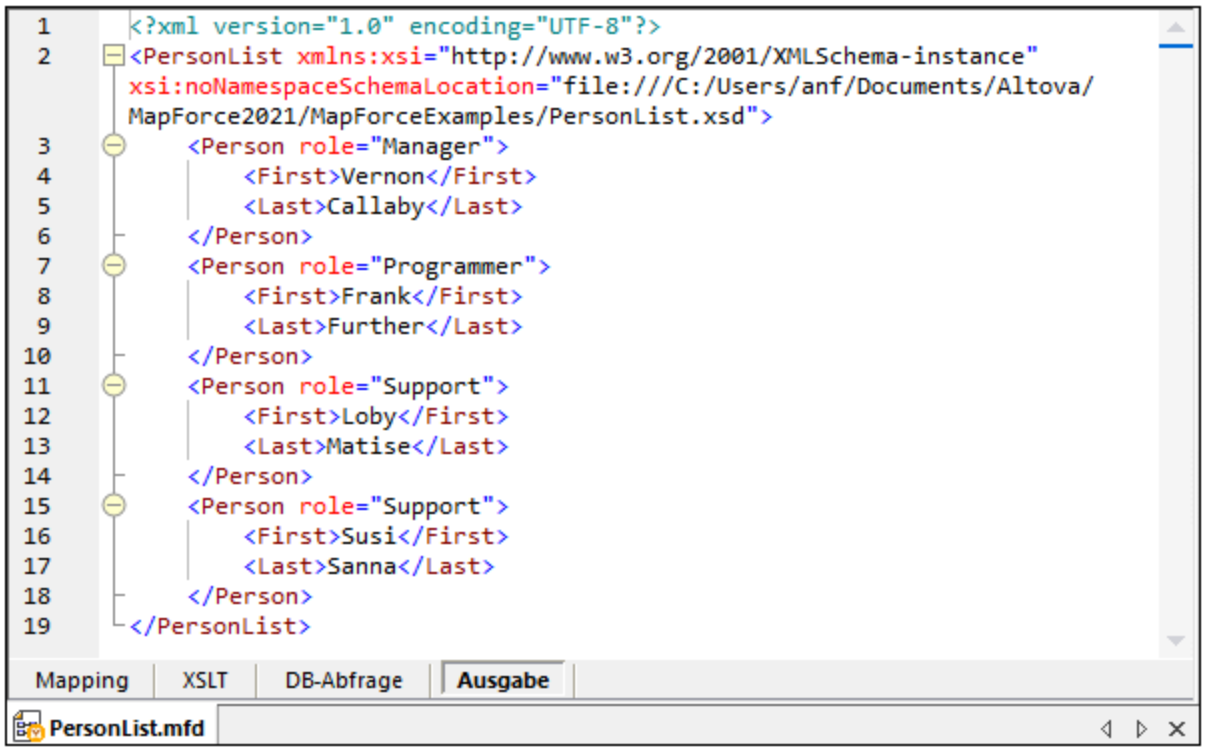
Bereich "DB-Abfrage" (Enterprise und Professional Edition)

Im Bereich **DB-Abfrage** können Sie Abfragen an allen gebräuchlichen Datenbanken durchführen. Sie können mit mehreren aktiven Verbindungen zu verschiedenen Datenbanken arbeiten. Nähere Informationen dazu finden Sie unter Anzeigen und Abfragen von Datenbanken 388.



Fenster "Ausgabe"

Auf dem Register **Ausgabe** sehen Sie das Ergebnis der Mapping-Transformation. Wenn beim Mapping mehrere Dateien generiert werden, können Sie der Reihe nach durch die einzelnen generierten Dateien navigieren.



Dieses Fenster enthält auch Zeilennummerierung und eine Klappleiste, die ähnlich wie im Fenster "XSLT" (siehe oben) funktioniert.

StyleVision-Ausgabebereiche (Enterprise und Professional Edition)

Wenn Sie Altova StyleVision installiert haben, stehen neben dem Fenster **Ausgabe** die StyleVision-Ausgabebereiche zur Verfügung. In den StyleVision-Ausgabebereichen können Sie eine Vorschau der Mapping-Ausgabe in HTML, RTF, PDF und Word 2007+ anzeigen und speichern. Dazu werden in StyleVision StyleVision Power Stylesheet (SPS)-Dateien erstellt und in MapForce einer Mapping-Komponente zugewiesen.

© 2016-2022 Altova GmbH

2 Tutorials

Die MapForce Tutorials sollen Ihnen dabei helfen, die grundlegenden Datentransformationsfunktionen von MapForce zu verstehen und verwenden zu können. Sie werden Schritt für Schritt durch die grundlegenden Funktionen geführt, wobei die Aufgaben stufenweise komplexer werden. Daher wird empfohlen, die Tutorials der Reihe nach durchzuarbeiten. Grundkenntnisse in XML und XML-Schema sind von Vorteil.

Beispieldateien
Die in diesen Tutorials gezeigten oder referenzierten Mapping-Dateien stehen im Ordner BasicTutorials 27 zur Verfügung. Wenn Sie sich nicht sicher sind, welche Auswirkungen Ihre Änderungen an den MapForce-Originaldateien haben werden, erstellen Sie Sicherungskopien der Originaldatei, bevor Sie damit zu arbeiten beginnen.

Liste der Tutorials

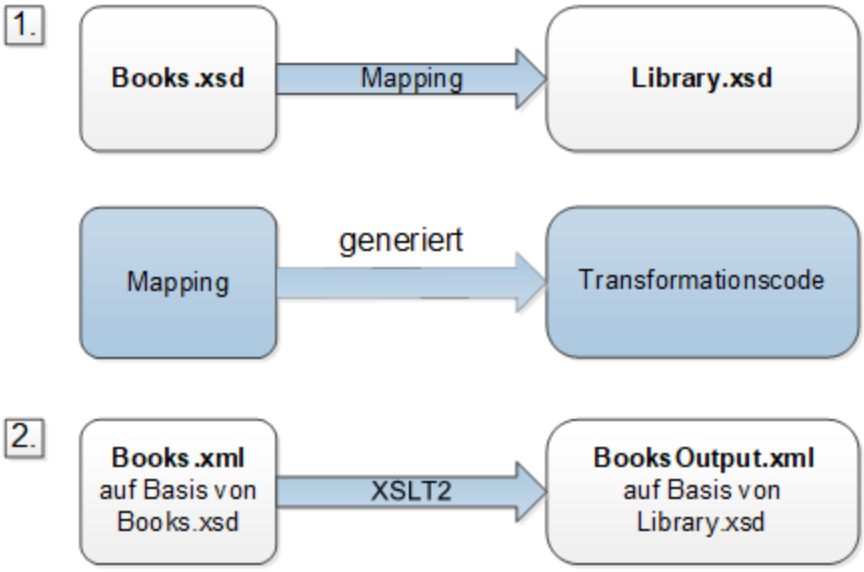
Eine Quellkomponente auf eine Zielkomponente
In diesem Tutorial 44 wird beschrieben, wie Sie die Nodes einer Quelldatei mit Hilfe grundlegender MapForce-Mechanismen auf die Nodes einer Zieldatei mappen. In diesem Tutorial wird anschließend erläutert, wie Sie eine durch ein bestimmtes XML-Schema definierte XML-Datei in eine durch ein anderes Schema definierte XML-Datei konvertieren.
Mehrere Quellkomponenten auf eine Zielkomponente
In diesem Tutorial 54 wird gezeigt, wie Sie Daten aus mehreren XML-Quelldateien in einer Zieldatei zusammenführen.
Verkettete Mappings
In diesem Tutorial 61 erstellen wir ein einfaches Mapping wie im ersten Tutorial, filtern anschließend die mit diesem Mapping erzeugten Daten und übergeben diese dann an die zweite Zielkomponente.
Mehrere Quellkomponenten auf mehrere Zielkomponenten
In diesem Tutorial 68 erfahren Sie, wie Sie Daten aus mehreren XML-Instanzdateien aus demselben Ordner auslesen und diese in mehrere on-the-fly generierte XML-Dateien schreiben.

2.1 Eine Quellkomponente auf eine Zielkomponente

In diesem Tutorial wird beschrieben, wie Sie ein Mapping für eines der einfachsten Szenarien erstellen. Ziel ist es, die Daten aus der XML-Datei A, der das XML-Schema A zugewiesen ist, in die XML-Datei B, der das XML-Schema B zugewiesen ist, zu schreiben. Unsere Zieldatei erhält somit dieselben Daten wie die Quelldatei, hat jedoch ein anderes Schema (eine andere Struktur). Dank des Transformationscodes können Sie sehen, wie sich die Struktur der Quelldatei geändert hat. Beachten Sie, dass auch nur die beiden Strukturen aufeinander gemappt werden könnten, jedoch könnte dann das Ergebnis der Transformation nicht in einer Vorschau im Ausgabefenster[41] angezeigt werden. Zu Illustrationszwecken wird daher eine XML-Datei namens `Books.xml` verwendet (*siehe Codefragment unten*) verwendet. Im Prinzip wird dabei folgendermaßen vorgegangen:

1. Da wir zwei Datenstrukturen verwenden, werden wir in unserem Mapping-Design zwei Komponenten (eine *Quell-* und eine *Zielkomponente*) erstellen.
2. Anschließend müssen wir Nodes mappen, indem wir einen Quell-Node mit dem gewünschten Ziel-Node verbinden. Diese Verbindungen bilden das Mapping und bestimmen, welcher Quell-Node auf welchen Ziel-Node gemappt wird.
3. Da die Transformation von einem Dokument in ein anderes mit Hilfe einer geeigneten Transformationssprache wie z.B. XSLT ausgeführt wird, wählen wir eine Transformationssprache aus.
4. Zum Transformieren des XML-Quelldokuments (`Books.xml`) in das gewünschte Zieldokument verwenden wir die integrierten Transformationsprozessoren von MapForce. Bei diesem Zieldokument handelt es sich um ein gemäß dem XSD-Zielschema (`Library.xsd`) gültiges XML-Dokument.
5. Schlussendlich können wir die XML-Ausgabedatei speichern.

In der Abbildung unten sehen Sie ein abstraktes Modell der in diesem Tutorial verwendeten Datentransformation:



Im oben gezeigten abstrakten Modell sehen Sie die zwei Schritte des Mapping-Vorgangs. Im ersten Schritt wird die Struktur von `Books.xsd` auf eine neue Struktur namens `Library.xsd` gemappt. Das Mapping wird mit Hilfe einer Transformationssprache[28] durchgeführt. Welche Transformationssprachen zur Auswahl stehen, hängt von Ihrer MapForce Edition ab. In unserem Fall wird als Transformationssprache XSLT2 gewählt. Im zweiten Schritt wird der Inhalt der Datei `Books.xml`, der `Books.xsd` zugewiesen wurde, auf die Zieldatei

© 2016-2022 Altova GmbH

(BooksOutput.xml) gemappt, der ein neues Schema (Library.xsd) zugewiesen wurde. Das nachstehende Codefragment enthält die Beispieldaten aus der Datei **Books.xml**, die als Datenquelle verwendet wird.

```
<books>
   <book id="1">
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <category>Fiction</category>
      <year>1876</year>
   </book>
   <book id="2">
      <author>Franz Kafka</author>
      <title>The Metamorphosis</title>
      <category>Fiction</category>
      <year>1912</year>
   </book>
</books>
```

So sollten die Daten in der Zieldatei namens **BooksOutput.xml** aussehen:

```
<library>
   <last_updated>2015-06-02T16:26:55+02:00</last_updated>
   <publication>
      <id>1</id>
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <genre>Fiction</genre>
      <publish_year>1876</publish_year>
   </publication>
   <publication>
      <id>2</id>
      <author>Franz Kafka</author>
      <title>The Metamorphosis</title>
      <genre>Fiction</genre>
      <publish_year>1912</publish_year>
   </publication>
</library>
```

Einige Elementnamen in der XML-Quell- und Zieldatei sind nicht identisch. Unser Ziel ist, die Elemente <author>, <title>, <genre> und <publish_year> der Zieldatei mit dem Inhalt aus den entsprechenden Elementen in der Quelldatei (<author>, <title>, <category>, <year>) zu befüllen. Das Attribut id in der Quelldatei muss auf das Element <id> in der Zieldatei gemappt werden. Schließlich müssen wir noch das Element <last_updated> der Zieldatei mit dem Datum und der Uhrzeit der letzten Aktualisierung der Datei befüllen.

Um die erforderliche Datentransformation durchzuführen, gehen Sie vor, wie in den nachfolgenden Unterabschnitten beschrieben.

2.1.1 Erstellen und Speichern eines Designs

In diesem Kapitel wird erläutert, wie Sie ein neues Design erstellen, eine Transformationssprache auswählen und Ihr Mapping validieren und speichern.

Erstellen eines neuen Designs

Um Daten transformieren zu können, müssen Sie ein neues Mapping-Design erstellen. Dies kann auf zwei Arten geschehen:

- Gehen Sie zum Menü **Datei** und klicken Sie auf **Neu**. Wählen Sie anschließend **Mapping** aus und klicken Sie auf **OK**.
- Klicken Sie in der Symbolleiste auf . Wählen Sie anschließend **Mapping** aus und klicken Sie auf **OK**.

Wählen Sie eine Transformationssprache aus.

Je nach MapForce Edition stehen verschiedene Transformationssprachen[28] zur Verfügung. Für dieses Tutorial haben wir XSLT2 ausgewählt. Sie können diese Transformationssprache auf die folgenden zwei Arten auswählen:

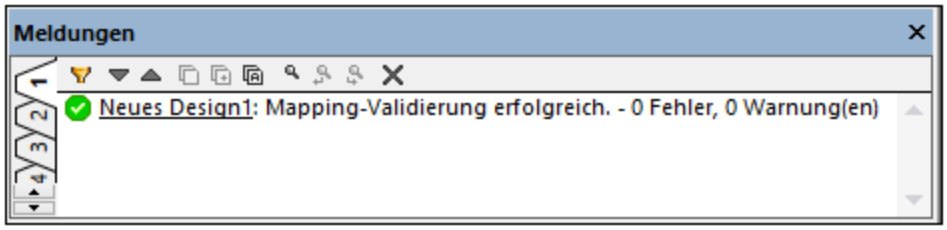
- Klicken Sie in der Symbolleiste auf  (**XSLT2**).
- Klicken Sie im Menü **Ausgabe** auf **XSLT 2.0**.

Validieren und Speichern des Designs

Die Validierung des Mappings ist ein optionaler Schritt, mit Hilfe dessen Sie potenzielle Mapping-Fehler- und Warnungen sehen und beheben können, bevor Sie das Mapping ausführen. Sie können Ihr Mapping jederzeit validieren. Um zu überprüfen, ob das Mapping gültig ist, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Datei** auf **Mapping validieren.**
- Klicken Sie in der Symbolleiste auf  (**Validieren**).

Die Validierungsergebnisse werden im Fenster **Meldungen** folgendermaßen angezeigt:



Wählen Sie dazu eine der folgenden Methoden:

- Klicken Sie im Menü **Datei** auf **Speichern**.
- Klicken Sie in der Symbolleiste auf  (**Speichern**).

Das Mapping aus diesem Tutorial wurde unter dem Namen **`Tut1_SchemaToSchema.mfd`** gespeichert.

© 2016-2022 Altova GmbH

2.1.2 Hinzufügen der Quellkomponente

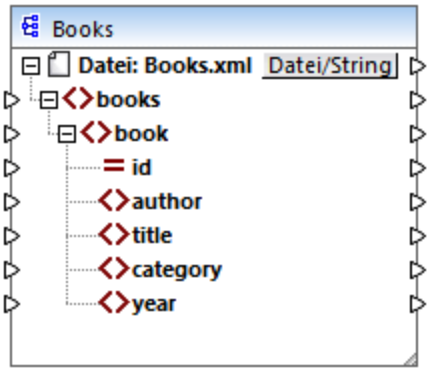
Wir möchten nun eine XSD-Datei, die die Struktur der ersten Komponente bildet und eine XML-Datei, die die Daten für diese Komponente enthält, hinzufügen. Die Quelldatei namens `Books.xsd` kann auf eine der folgenden Arten zum Mapping hinzugefügt werden:

- Klicken Sie in der Symbolleiste auf  (**XML-Schema/Datei einfügen**).
- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei**.
- Ziehen Sie `Books.xsd` aus dem Windows Explorer in den Mapping-Bereich.

Wenn Sie ein Schema hinzufügen, schlägt MapForce vor, eine XML-Beispieldatei hinzuzufügen. Klicken Sie auf **Durchsuchen** und navigieren Sie zur Datei `Books.xml` im selben Ordner. Unsere Quelldatei enthält nun sowohl ein Schema als auch Inhalt. In den Eigenschaften der einzelnen Quell- oder Zielkomponenten kann eine XSD- und eine XML-Datei definiert werden. Die XSD-Datei definiert die Struktur des Dokuments in dieser Komponente. Aus der XML-Datei stammen die Daten für diese Quell- oder Zielkomponente. Diese müssen gemäß dem Schema dieser Komponente gültig sein. Wenn eine Komponente anhand einer XSD-Datei erstellt wird, werden Sie aufgefordert, eine XML-Datei als Datendatei für diese Komponente anzugeben. Wenn eine Komponente anhand einer XML-Datei erstellt wird, wird die Struktur der Daten der Komponente anhand der von der XML-Datei referenzierten XSD-Datei definiert. Wenn keine Referenz auf eine XSD-Datei vorhanden ist, werden Sie gefragt, ob MapForce eine XSD-Datei für diese Komponente generieren soll.

Anzeigen der Struktur

Nachdem Sie die Quelldatei nun zum Mapping-Bereich hinzugefügt haben, sehen Sie ihre Struktur. Diese Struktur wird in MapForce als Mapping-Komponente oder einfach Komponente[76] bezeichnet. Durch Klick auf das ⊞-Symbol können Sie Elemente in der Komponente erweitern. Alternativ dazu können Sie die **+**-Taste des Ziffernblocks drücken. In der Abbildung unten sehen Sie die Quellkomponente:



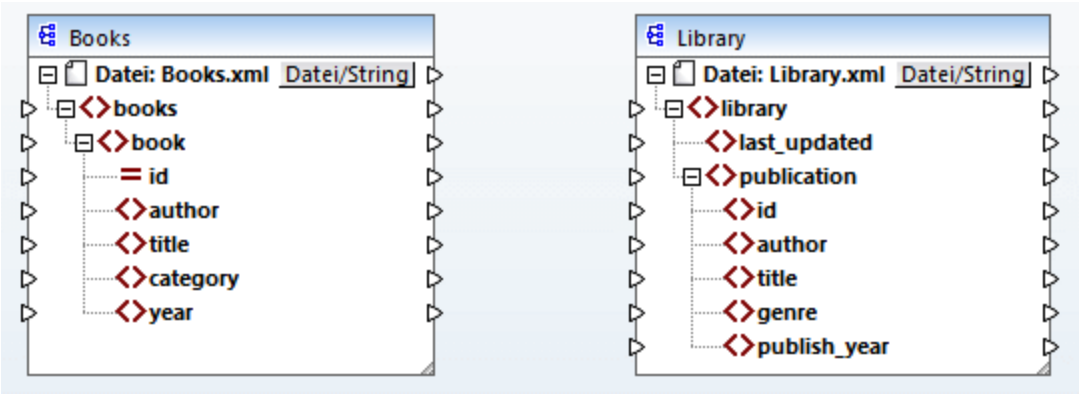
`Books` in der Titelleiste ist der Name der Komponente. Der oberste Node bildet den Namen der XML-Instanzdatei: `Datei: Books.xml`. Die XML-Elemente in der Struktur werden durch das Symbol  gekennzeichnet. XML-Attribute werden durch das Symbol  gekennzeichnet. Die kleinen Dreiecke an beiden Seiten der Komponente repräsentieren auf der linken Seite Daten-Inputs und auf der rechten Seite Daten-Outputs. Diese Dreiecke werden in MapForce als *Input*- bzw. *Output-Konnektoren* bezeichnet.

Verschieben von Komponenten und Anpassen der Größe von Komponenten

Um die Komponente im Mapping-Fenster zu verschieben, klicken Sie auf die Komponentenüberschrift und ziehen Sie sie mit der Maus an die neue Position. Um die Größe der Komponente anzupassen, ziehen Sie die rechte untere Ecke der Komponente mit der Maus an eine andere Position. Sie können auch auf diese Ecke doppelklicken, damit MapForce die Größe automatisch anpasst.

2.1.3 Hinzufügen der Zielkomponente

In nächsten Schritt wird nun eine Zielkomponente hinzugefügt und ihre Einstellungen definiert. Um die Zieldatei namens `Library.xsd` zum Mapping hinzuzufügen, klicken Sie auf  (**XML-Schema/Datei einfügen**). Klicken Sie auf **Überspringen**, wenn Sie von MapForce aufgefordert werden, eine Instanzdatei bereitzustellen. Zu diesem Zeitpunkt sieht das Mapping-Design folgendermaßen aus:

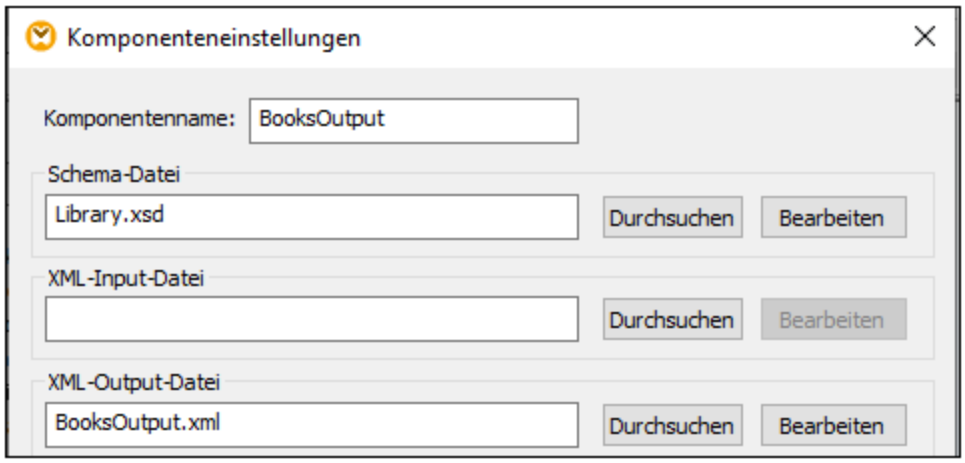


Wenn Sie `Library.xsd` öffnen, wird sie in der Komponente als XML-Datei angezeigt. Tatsächlich erzeugt MapForce nur eine Referenz auf die XML-Datei namens `Library.xml`, wobei diese XML-Datei selbst noch gar nicht existiert. Unsere Zielkomponente hat also ein Schema aber keinen Inhalt.

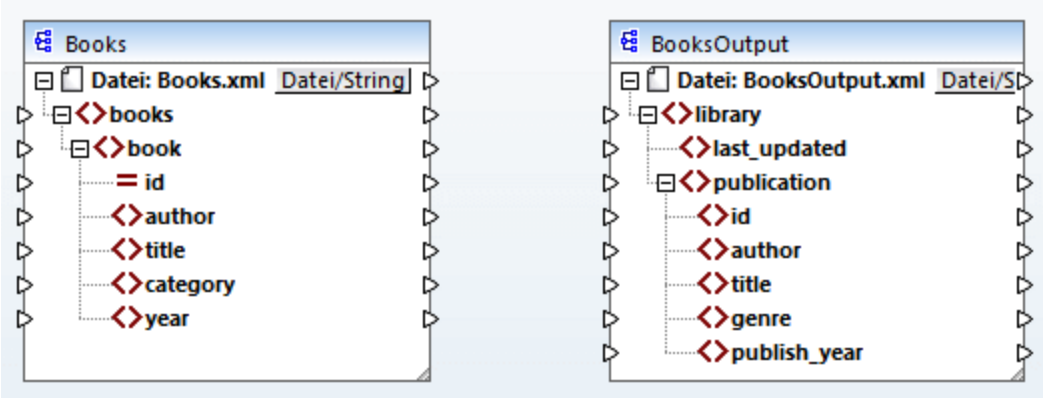
Komponenteneinstellungen

Wir müssen die Zielkomponente zu diesem Zeitpunkt außerdem in `BooksOutput.xml` umbenennen. Dadurch vermeiden, wir, dass es in den nächsten Tutorials zu Verwirrung kommt, da wir dort eine separate Datei namens `Library.xml` verwenden werden, die ihren eigenen Inhalt hat und auf demselben Schema `Library.xsd` basiert. Um die Zielkomponente umzubenennen, doppelklicken Sie auf die Überschrift der Zielkomponente. Daraufhin wird das Dialogfeld Komponenteneinstellungen 126 (*siehe Abbildung unten*) geöffnet, in dem wir den Namen der Zieldatei folgendermaßen ändern:

 © 2016-2022 Altova GmbH



Das Mapping-Design sieht nun folgendermaßen aus:



Anmerkung: Der Name der Überschrift bezieht sich nur auf den Namen der Komponente und nicht auf den Namen des Schemas, auf dem diese Datei basiert. Um den Namen des Schemas zu sehen, öffnen Sie das Dialogfeld **Komponenteneinstellungen**.

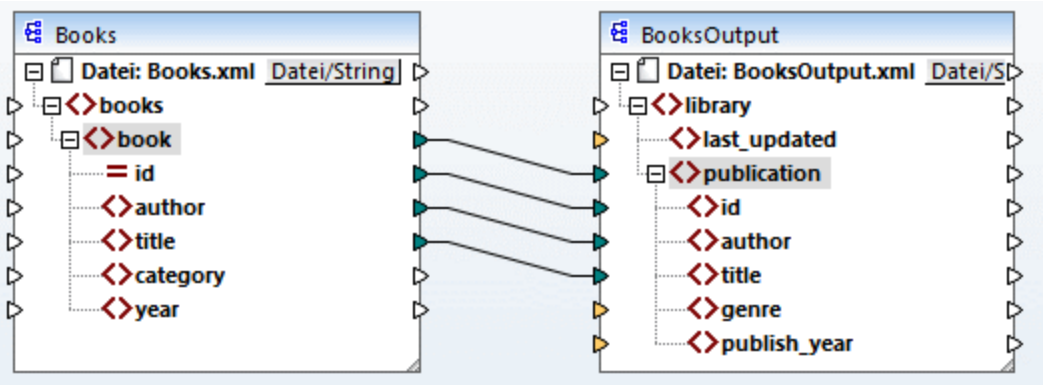
2.1.4 Verbinden von Quell- und Zielkomponente

In diesem Schritt werden wir die Daten in der Quelldatei auf die Zieldatei mappen. Es sollen zwei Arten von Input auf die Ziel-Nodes gemappt werden: (i) Quell-Nodes und (ii) Daten. Beide werden gleichzeitig gemappt. Die meisten Daten stammen aus **`Books.xml`**. Außerdem stellen wir in unserem Beispiel mit Hilfe der XPath-Funktion `current-dateTime` 1195 Informationen über das aktuelle Datum und die aktuelle Uhrzeit bereit.

Automatische Verbindungen

Wir werden nun eine Mapping-Verbindung zwischen dem <book>-Element in der Quellkomponente und dem <publication>-Element in der Zielkomponente erstellen. Klicken Sie dazu auf den Output-Konnektor (das kleine Dreieck) rechts vom Element <book> und ziehen Sie ihn auf den Input-Konnektor des Elements <publication> in der Zielkomponente. Bei Ziehen der Verbindungslinie verbindet MapForce unter Umständen automatisch alle Child-Elemente von <book> in der Quelldatei mit gleichnamigen Elementen in der Zieldatei. In

unserem Beispiel wurden vier Verbindungen gleichzeitig erstellt (*siehe Abbildung unten*). Dieses Funktion hat den Namen *Idente Sub-Einträge automatisch verbinden* und kann deaktiviert und gegebenenfalls angepasst werden.



Sie können **Idente Sub-Einträge automatisch verbinden** auf eine der folgenden Arten aktivieren bzw. deaktivieren:

- Klicken in der Symbolleiste auf (**Aktiviert/Deaktiviert die automatische Verbindung von Sub-Einträgen**.
- Klicken Sie im Menü **Verbindung** auf den Befehl **Idente Sub-Einträge automatisch verbinden**.

Verbinden obligatorischer Datenelemente

Beachten Sie, dass einige der Input-Konnektoren der Zielkomponente von MapForce orange markiert wurden. Dies bedeutet, dass diese Datenelemente zwingend erforderlich sind. Sie sind obligatorisch, da Sie im Schema der Datei als obligatorisch definiert wurden. Damit die XML-Zieldatei gültig ist, müssen Sie für die obligatorischen Datenelemente auf folgende Art Werte bereitstellen:

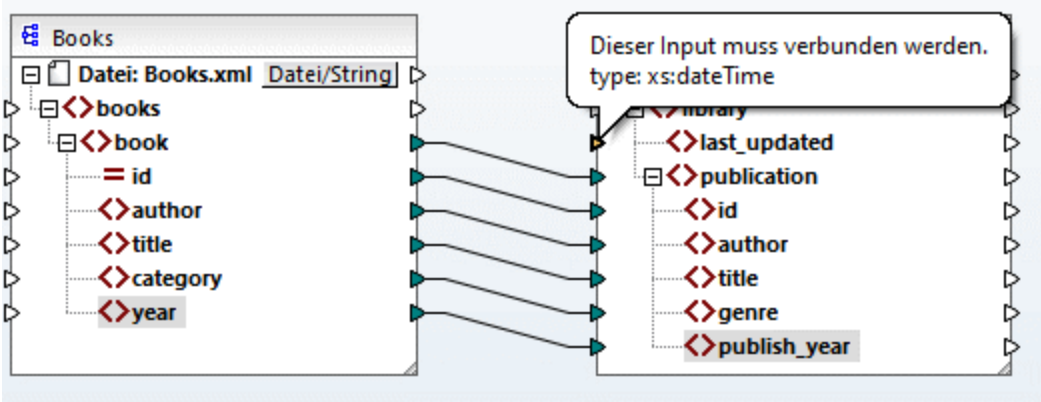
- Verbinden Sie das Datenelement `<category>`in der Quellkomponente mit dem Datenelement `<genre>` in der Zielkomponente.
- Verbinden Sie das Datenelement `<year>` in der Quellkomponente mit dem Datenelement `<publish_year>` in der Zielkomponente.

Hinzufügen des aktuellen Datums und der aktuellen Uhrzeit

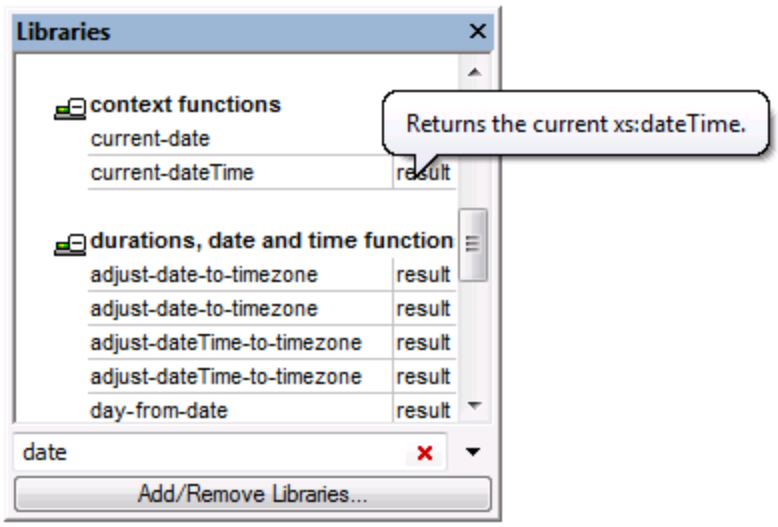
Sie müssen nun nur noch einen Wert für das Element `<last_updated>` bereitstellen. Wenn Sie die Maus über seinen Input-Konnektor platzieren, sehen Sie, dass das Element den Typ `xs:dateTime` hat (*siehe Abbildung unten*). Damit Tipps angezeigt werden, aktivieren Sie die Symbolleisten-Schaltfläche (**Tipps anzeigen**).

Durch Klick auf (**Datentypen anzeigen**) in der Symbolleiste können die Datentypen der einzelnen Datenelemente jederzeit sichtbar gemacht werden.

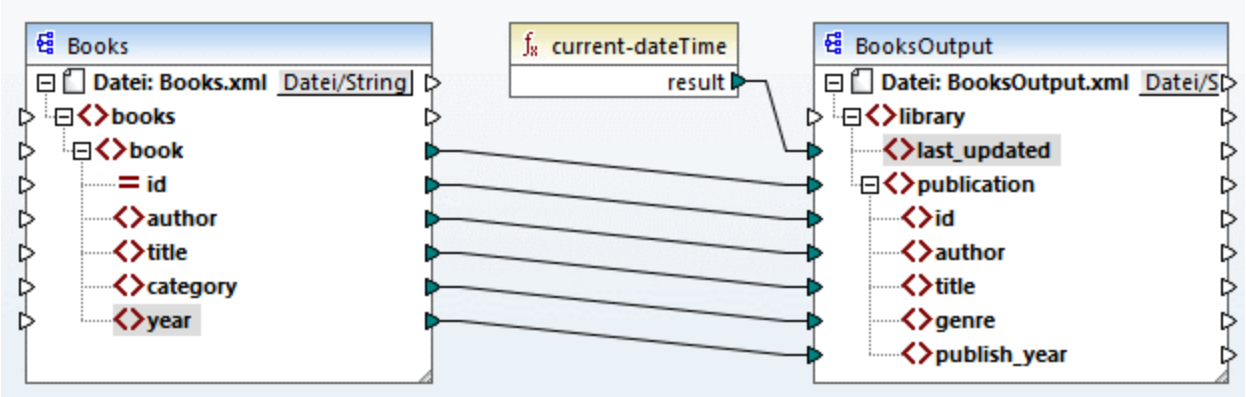
© 2016-2022 Altova GmbH



Sie können das aktuelle Datum und die aktuelle Uhrzeit mit Hilfe der XSLT2-Funktion **`current-dateTime`** abrufen. Um diese Funktion zu suchen, geben Sie den Namen der Funktion in das Textfeld im unteren Bereich des Fensters "Bibliotheken" 34 ein (*siehe Abbildung unten*). Doppelklicken Sie alternativ dazu auf einen leeren Bereich im **Mapping** und beginnen Sie mit der Eingabe von `current-date`.



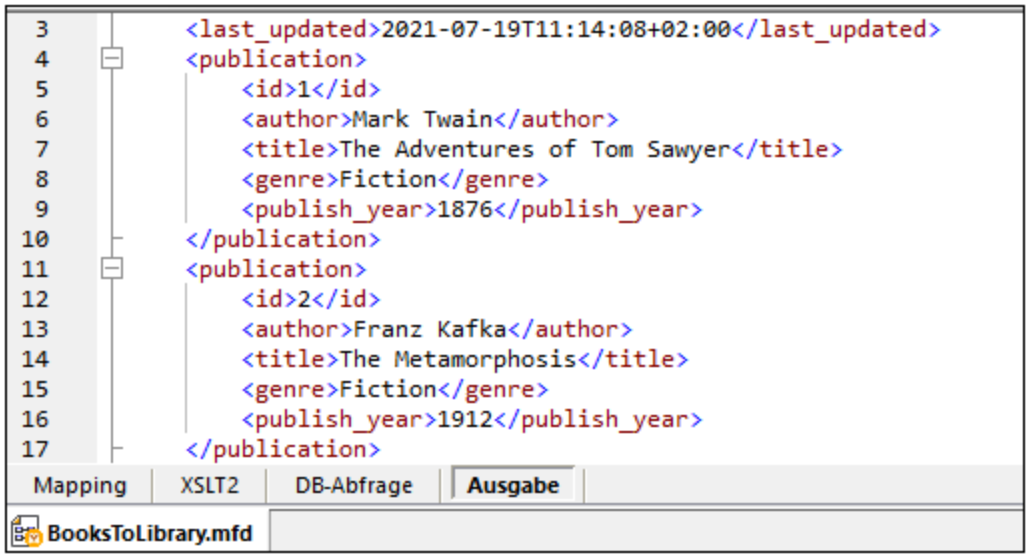
Um die Funktion zum Mapping hinzuzufügen, ziehen Sie die Funktion in den **Mapping**-Bereich und verbinden Sie ihren Output mit dem Input des Elements `<last_updated>` (*siehe Abbildung unten*).



Sie können Ihr Mapping jetzt, wie in Erstellen und Speichern eines Designs 45 beschrieben, validieren und speichern.

2.1.5 Vorschau auf das Mapping-Ergebnis

Mit Hilfe seiner integrierten Prozessoren können Sie in MapForce direkt im Bereich **Ausgabe** eine Vorschau auf das Mapping-Ergebnis anzeigen (*siehe Abbildung unten*).



Standardmäßig werden die im **Ausgabefenster** angezeigten Dateien nicht auf der Festplatte gespeichert. MapForce erstellt statt dessen temporäre Dateien. Um die Ausgabe zu speichern, öffnen Sie den **Ausgabebereich** und wählen Sie den Menübefehl **Ausgabe | Ausgabedatei speichern** oder klicken Sie in der Symbolleiste auf (**Generierte Ausgabe speichern**).

Um MapForce so zu konfigurieren, dass die Ausgabe, anstatt in einer temporären Datei, direkt in einer endgültigen Datei gespeichert wird, gehen Sie zu **Extras | Optionen | Allgemein** und aktivieren Sie das Kontrollkästchen **Direkt in die endgültigen Output-Dateien schreiben**. Beachten Sie, dass nicht empfohlen

 © 2016-2022 Altova GmbH

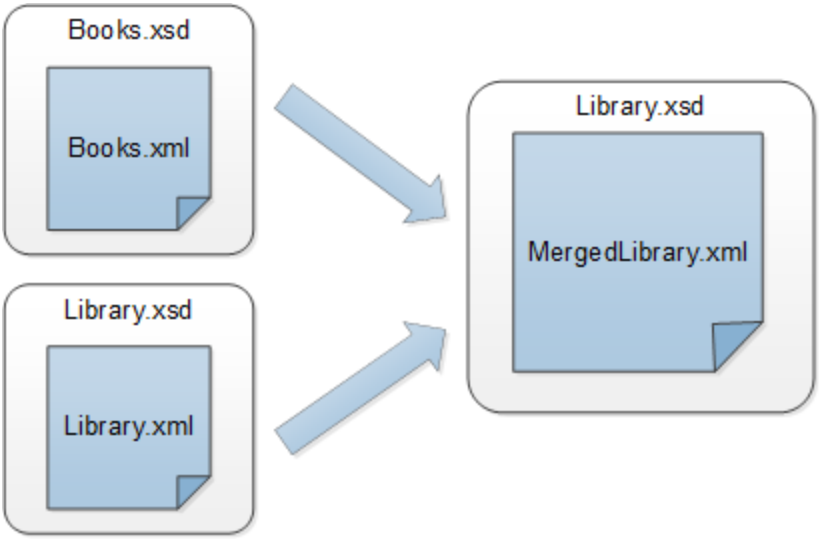
wird, diese Option zu aktivieren, während Sie das Tutorial durchnehmen, da Sie dadurch die Originaldateien eventuell unabsichtlich überschreiben könnten.

Sie können auch eine Vorschau des generierten XSLT-Codes, der die Transformation durchführt, anzeigen. Klicken Sie dazu im unteren Bereich des **Mapping**-Fensters auf die Schaltfläche **XSLT2**. Um den XSLT2-Code zu generieren und in einer Datei zu speichern, wählen Sie den Menübefehl **Datei | Code generieren in | XSLT 2.0**. Wenn Sie dazu aufgefordert werden, geben Sie einen Ordner an, in dem der generierte Code gespeichert werden soll. Nach Fertigstellung der Codegenerierung enthält der Zielordner die folgenden beiden Dateien:

1. Eine nach dem Zielschema benannte XSLT-Transformationsdatei. Diese Transformationsdatei hat das folgende Format: `MappingMapTo<TargetFileName>.xslt`.
2. Die Datei `DoTransform.bat`, mit der Sie die XSLT-Transformation mit Altova RaptorXML Server über die Befehlszeile ausführen können. Um den Befehl ausführen zu können, müssen Sie RaptorXML installieren.

2.2 Mehrere Quellkomponenten auf eine Zielkomponente

In diesem Tutorial erfahren Sie, wie Sie die Daten aus einer neuen Datei namens `Library.xml` mit den Daten aus `Books.xml` zusammenführen. Es wird eine Zieldatei namens `MergedLibrary.xml` erzeugt, die die Daten aus beiden Quelldateien enthält. Die Zieldatei basiert auf dem Schema `Library.xsd`. Beachten Sie, dass beide Quelldateien unterschiedliche Schemas haben. Wenn die Quelldateien dasselbe Schema hätten, könnten Sie ihre Daten auch auf andere Art miteinander zusammenführen (siehe Mehrere Quellkomponenten auf mehrere Zielkomponenten 68). In der Abbildung unten sehen Sie ein abstraktes Modell der in diesem Tutorial beschriebenen Datentransformation.



Das nachstehende Codefragment enthält einen Ausschnitt aus der Datei **`Books.xml`**, die als die erste Datenquelle verwendet wird.

```
<books>
   <book id="1">
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <category>Fiction</category>
      <year>1876</year>
   </book>
</books>
```

Das nachstehende Codefragment enthält einen Ausschnitt aus aus der Datei **`Library.xml`**, die als die zweite Datenquelle verwendet wird.

```
<library>
   <publication>
      <id>5</id>
      <author>Alexandre Dumas</author>
      <title>The Three Musketeers</title>
      <genre>Fiction</genre>
      <publish_year>1844</publish_year>
   </publication>
</library>
```


© 2016-2022 Altova GmbH

So sollten die zusammengeführten Daten in der Zieldatei namens **MergedLibrary.xml** aussehen:

```
<library>
   <publication>
      <id>1</id>
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <genre>Fiction</genre>
      <publish_year>1876</publish_year>
   </publication>
   <publication>
      <id>5</id>
      <author>Alexandre Dumas</author>
      <title>The Three Musketeers</title>
      <genre>Fiction</genre>
      <publish_year>1844</publish_year>
   </publication>
</library>
```

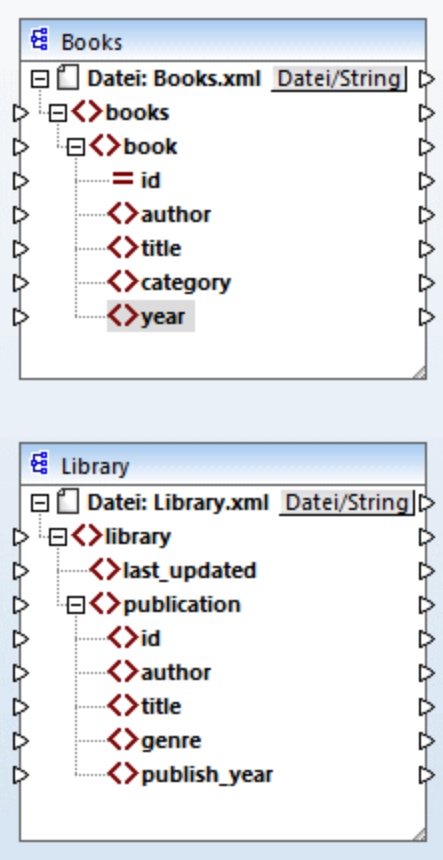
Um die Datentransformation durchzuführen, gehen Sie vor, wie in den nachfolgenden Unterabschnitten beschrieben.

2.2.1 Vorbereiten der Quelldateien

Den Ausgangspunkt dieses Tutorials bilden zwei Quelldateien, die jeweils ein Schema (**Books.xsd** and **Library.xsd**) und Daten (**Books.xml** and **Library.xml**) haben. Um die Quelldateien für das Mapping vorzubereiten, führen Sie die folgenden Schritte durch:

1. Öffnen Sie **Books.xsd**.
2. Wenn MapForce Sie auffordert, eine XML-Beispieldatei hinzuzufügen, klicken Sie auf **Durchsuchen** und öffnen Sie **Books.xml**.
3. Öffnen Sie **Library.xsd**.
4. Wenn MapForce Sie auffordert, eine XML-Beispieldatei hinzuzufügen, klicken Sie auf **Durchsuchen** und öffnen Sie **Library.xml**.
5. Platzieren Sie die Quelldateien aus Gründen der Übersichtlichkeit übereinander (*siehe Abbildung unten*).

Zu diesem Zeitpunkt sieht das Mapping-Design folgendermaßen aus:

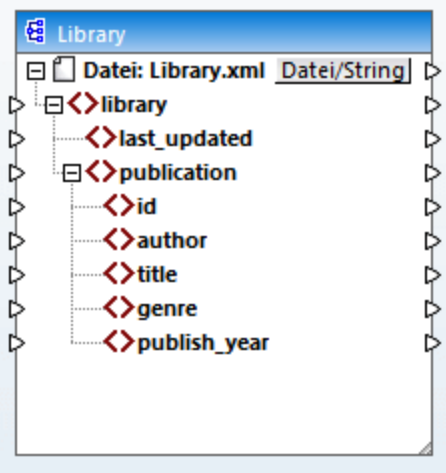


Da im neuen Mapping einige Dateien aus demselben Ordner referenziert werden, stellen Sie sicher, dass Sie dieses neue Mapping im Ordner **`BasicTutorials`** speichern.

2.2.2 Hinzufügen der Zielkomponente

In nächsten Schritt wird nun eine Zielkomponente namens `Library.xsd` hinzugefügt. Um die Zieldatei hinzuzufügen, klicken Sie auf  (**XML-Schema/Datei einfügen**). Klicken Sie auf **Überspringen**, wenn Sie von MapForce aufgefordert werden, eine Instanzdatei bereitzustellen. Klicken Sie anschließend auf die Überschrift der neuen Komponente und ziehen Sie sie auf die rechte Seite der beiden Quellkomponenten. Sie können Sie Mapping-Komponenten jederzeit beliebig verschieben. Wenn Sie eine Quellkomponente jedoch links von der Zielkomponente platzieren, wird das Mapping übersichtlicher und verständlicher. Dies ist auch die Konvention bei allen in dieser Dokumentation gezeigten Mappings und bei allen mit MapForce installierten Beispielmappingdateien.

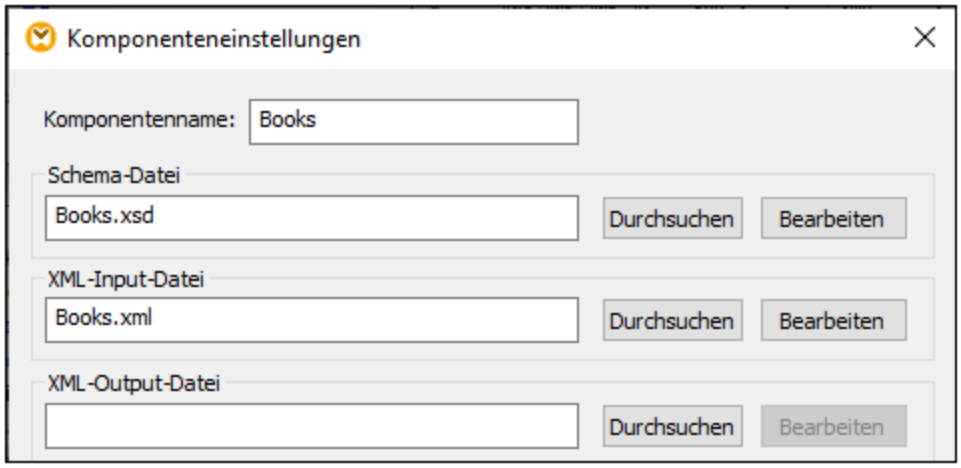
© 2016-2022 Altova GmbH

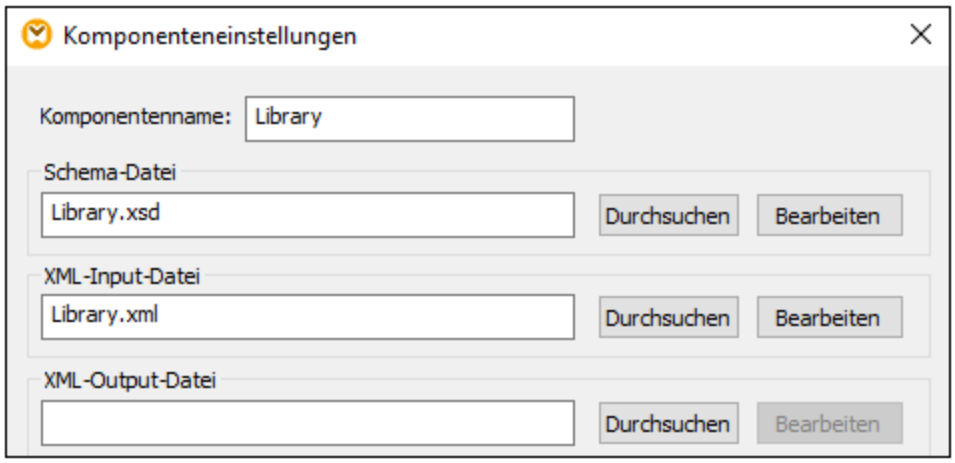


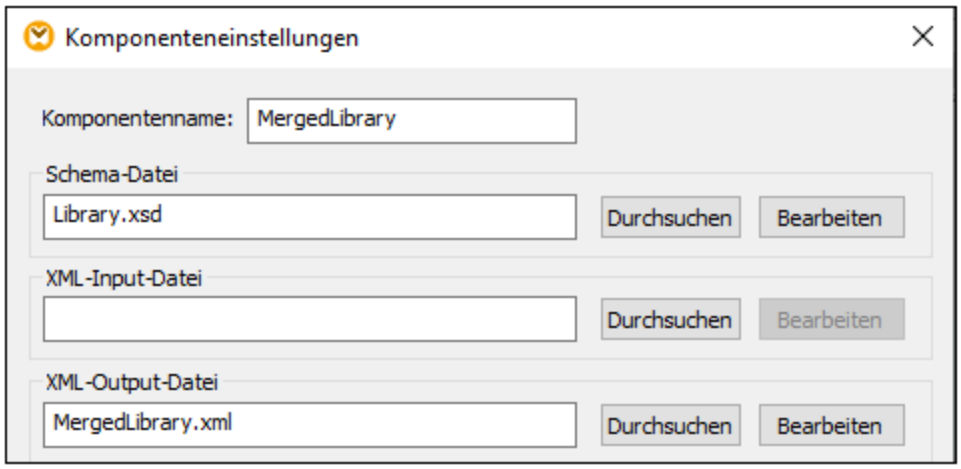
Die Zieldatei hat zu diesen Zeitpunkt nur eine Struktur, aber noch keine Daten. Die Daten werden später aus den beiden Quelldateien zusammengeführt und auf die Zieldatei gemappt.

2.2.3 Überprüfen und Definieren der Input/Output-Dateien

Das Mapping hat nun zwei Quellkomponenten: (`Books` und `Library`) und eine Zielkomponente (`Library`). Wir haben nun zwei Komponenten mit demselben Namen - `Library.xml`. Um Verwechslungen zu vermeiden, müssen wir die Einstellungen im Dialogfeld "Komponenteneinstellungen" 126 ändern. Doppelklicken Sie auf die Überschriften der einzelnen Komponenten. Überprüfen und ändern Sie die Namen und die Input/Output-Dateien der einzelnen Komponenten, wie unten gezeigt.







Wie oben gezeigt, liest die erste Quellkomponente Daten aus `Books.xml`. Die zweite Quellkomponente liest Daten aus `Library.xml`. Die Zielkomponente schließlich schreibt die Daten in eine Datei namens `MergedLibrary.xml`.

2.2.4 Verbinden von Quell- und Zielkomponenten

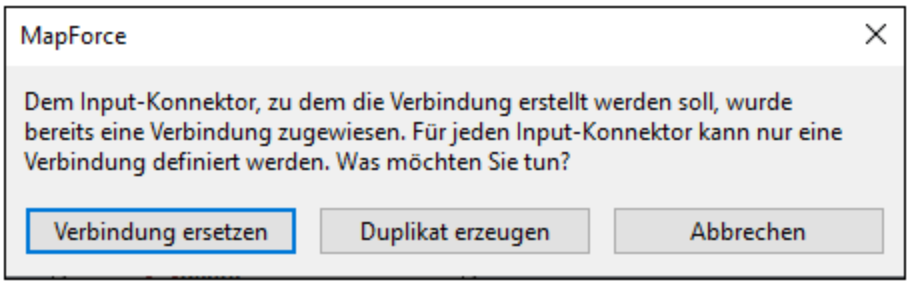
Wir wollen nun die Daten aus zwei Quelldateien (`Books.xml` und `Library.xml`) auf die Zieldatei (`MergedLibrary.xml`) mappen. Dazu gehen Sie folgendermaßen vor:

- Verbinden Sie das Element `<book>` in der ersten Quellkomponente mit dem Element `<publication>` in `MergedLibrary.xml`. Bei Ziehen der Verbindungslinie verbindet MapForce unter Umständen automatisch alle Child-Elemente von `<book>` in der Quelldatei mit gleichnamigen Elementen in der Zieldatei. In unserem Beispiel wurden die vier Verbindungen gleichzeitig erstellt. Nähere Informationen über die automatische Verbindung von Sub-Einträgen finden Sie im ersten Tutorial 49.
- Wenn Sie das Element `<book>` mit dem Element `<publication>` verbinden, werden Sie sehen, dass einige der Input-Konnektoren in der Zielkomponente orange markiert erscheinen. Dies bedeutet, dass diese Datenelemente obligatorisch sind. Um die Gültigkeit der XML-Zieldatei zu gewährleisten,

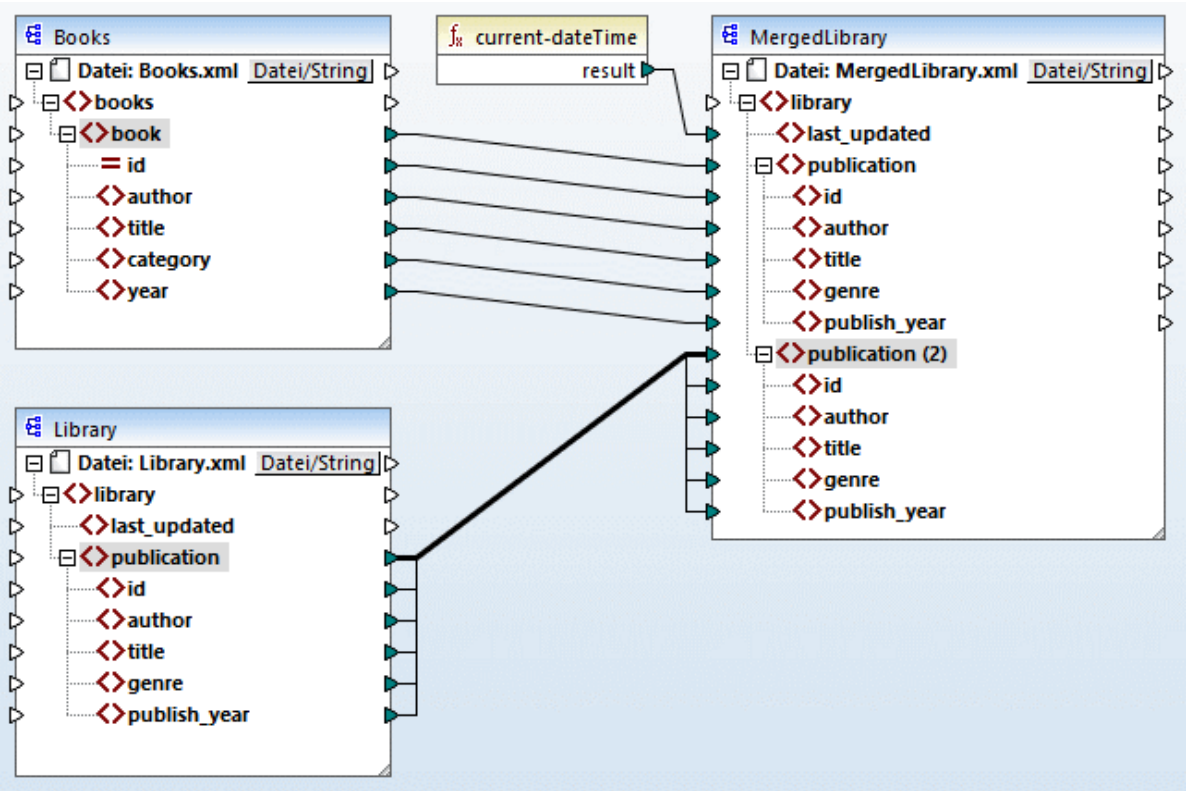
 © 2016-2022 Altova GmbH

verbinden Sie das Element <category> mit dem Element <genre> und das Element <year> mit dem Element <publish_year>.

- Um einen Wert für das Element <last_updated> bereitzustellen, suchen Sie in der XSLT2-Bibliothek nach der Funktion mit dem Namen current-dateTime. Ziehen Sie die Funktion in den Mapping-Bereich und verbinden Sie result mit dem Element <last_updated> in MergedLibrary.xml.
- Damit MapForce die Daten aus der zweiten Quellkomponente in die Zielkomponente schreibt, klicken Sie auf den Output-Konnektor des Elements <publication> in Library.xml und ziehen Sie ihn auf den Input-Konnektor des Elements <publication> in MergedLibrary.xml. Da der Input-Konnektor der Zielkomponente bereits eine Verbindung hat, wird die folgende Meldung angezeigt:



Unser Ziel in diesem Tutorial ist es, Daten aus zwei Quellkomponenten auf eine Zielkomponente zu mappen. Klicken Sie daher auf **Duplikat erzeugen**. Dadurch konfigurieren Sie die Zielkomponente so, dass sie auch Daten aus der neuen Quellkomponente akzeptiert. Das Mapping sieht nun folgendermaßen aus:



In der Abbildung oben sehen Sie, dass das Datenelement publication in der Zielkomponente nun dupliziert wurde. Der neue Node publication(2) akzeptiert Daten aus der Library-Quellkomponente. Beachten Sie

außerdem, dass als Name dieses Node im Mapping zwar `publication(2)` angezeigt wird, dass der Name in der XML-Zielkomponente aber, wie beabsichtigt, `publication` lautet.

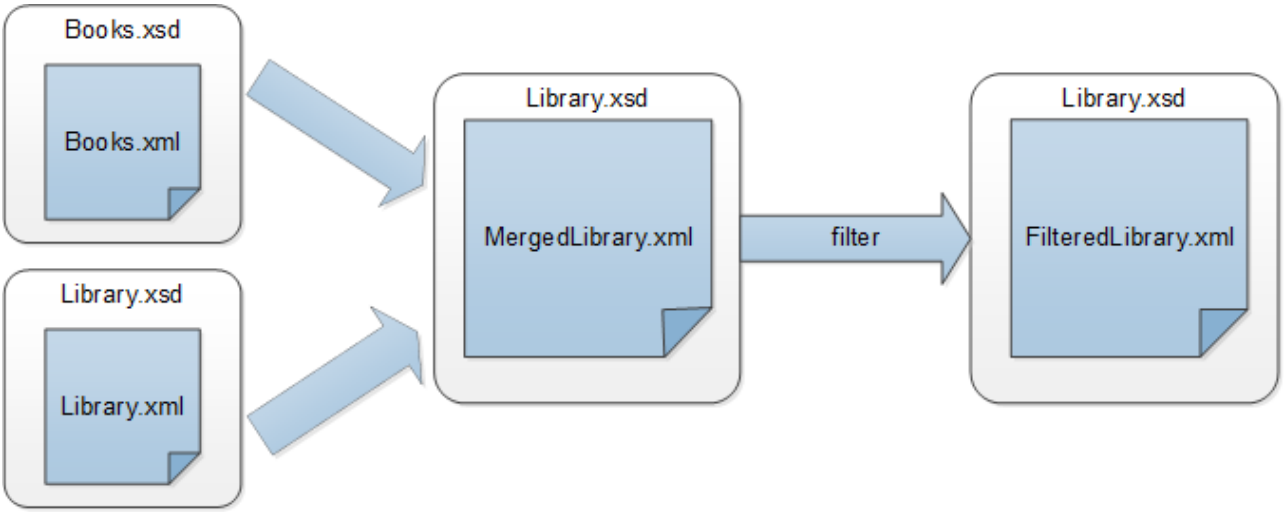
Klicken Sie nun am unteren Rand des **Mapping**-Bereichs auf die Schaltfläche **Ausgabe**, um das Ergebnis das Mappings anzuzeigen. Sie werden sehen, dass die Daten aus den beiden Dateien `Books.xml` und `Library.xml` nun in der neuen Datei `MergedLibrary.xml` zusammengeführt wurden. Um die Ausgabe zu speichern, öffnen Sie den **Ausgabebereich** und wählen Sie den Menübefehl **Ausgabe | Ausgabedatei speichern** oder klicken Sie in der Symbolleiste auf  (**Generierte Ausgabe speichern**).

Das Mapping-Design aus diesem Tutorial wurde unter dem Namen **`Tut2_MultipleToOne.mfd`** gespeichert. Dies ist nützlich, da dieses Mapping als Ausgangsbasis für das nächste Tutorial 61 dient.

 © 2016-2022 Altova GmbH

2.3 Verkettetes Mapping

Ziel dieses Tutorials ist es die Daten in der im vorherigen Tutorial[54] erstellten Datei `MergedLibrary.xml`, so zu filtern, dass nur eine Teilmenge dieser Daten in der Ausgabe angezeigt wird. Wir benötigen hier nur die nach 1900 veröffentlichten Bücher. In der Abbildung unten sehen Sie ein abstraktes Modell der in diesem Tutorial beschriebenen Datentransformation.

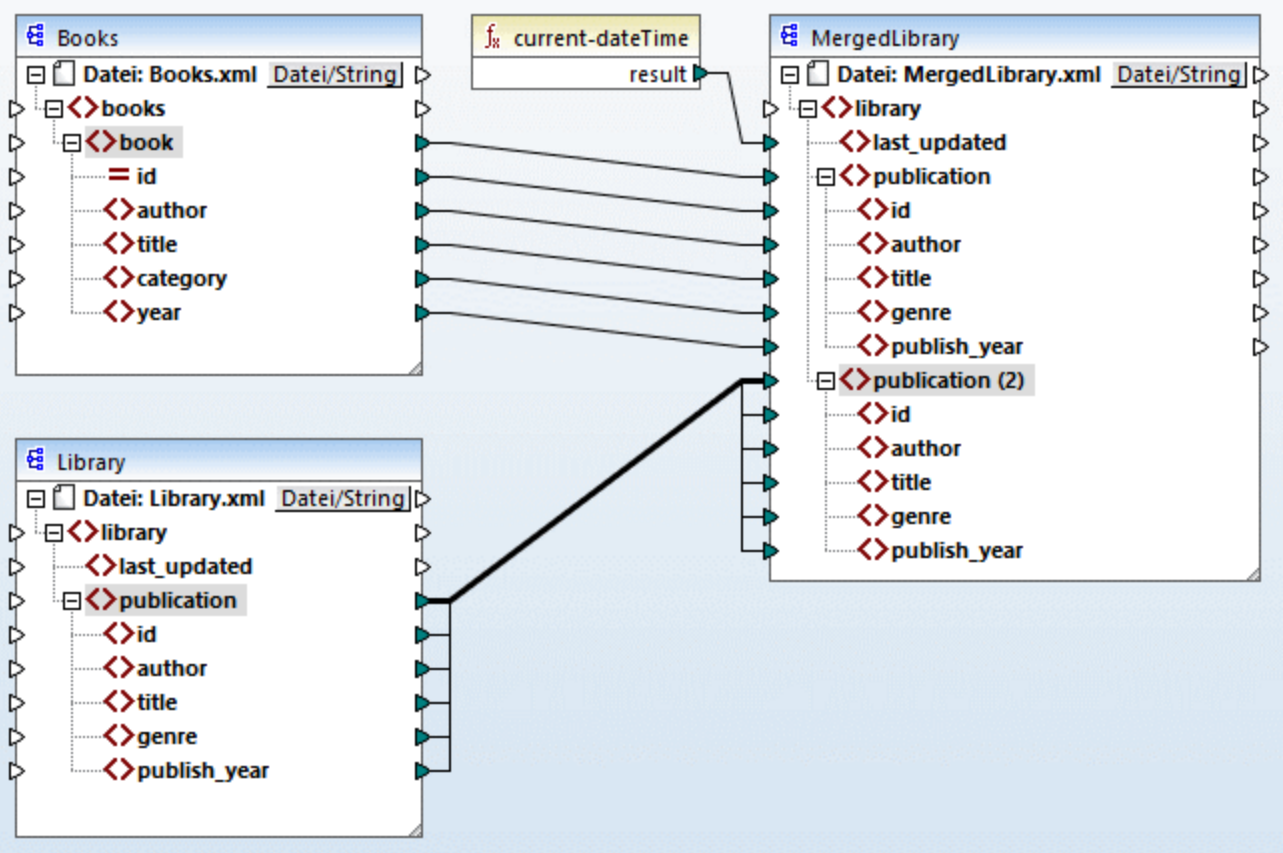


Im obigen Diagramm werden zuerst die Daten aus zwei Quelldateien (`Books.xml` und `Library.xml`) in einer einzigen Zieldatei namens `MergedLibrary.xml` zusammengeführt. Anschließend werden die Daten mit Hilfe einer Filterfunktion transformiert und an die nächste Komponente namens `FilteredLibrary.xml`übergeben. Beachten Sie, dass `FilteredLibrary.xml` auf dem Schema `Library.xsd` basiert. Die Zwischenkomponente dient sowohl als Datenziel als auch als Datenquelle. Dieses Verfahren wird in MapForce als *verkettetes Mapping*[1299] bezeichnet.

Um dias Mapping durchzuführen, gehen Sie vor, wie in den nachfolgenden Unterabschnitten beschrieben.

2.3.1 Vorbereiten des Mapping-Designs

Als Ausgangsbasis für dieses Tutorial wird das Mapping **`Tut2_MultipleToOne.mfd`** (*siehe Abbildung unten*) verwendet. Dieses Mapping wurde im vorherigen Tutorial[54] erstellt.



Da im neuen Mapping einige Dateien aus demselben Ordner referenziert werden, stellen Sie sicher, dass Sie dieses neue Mapping im Ordner **`BasicTutorials`** speichern.

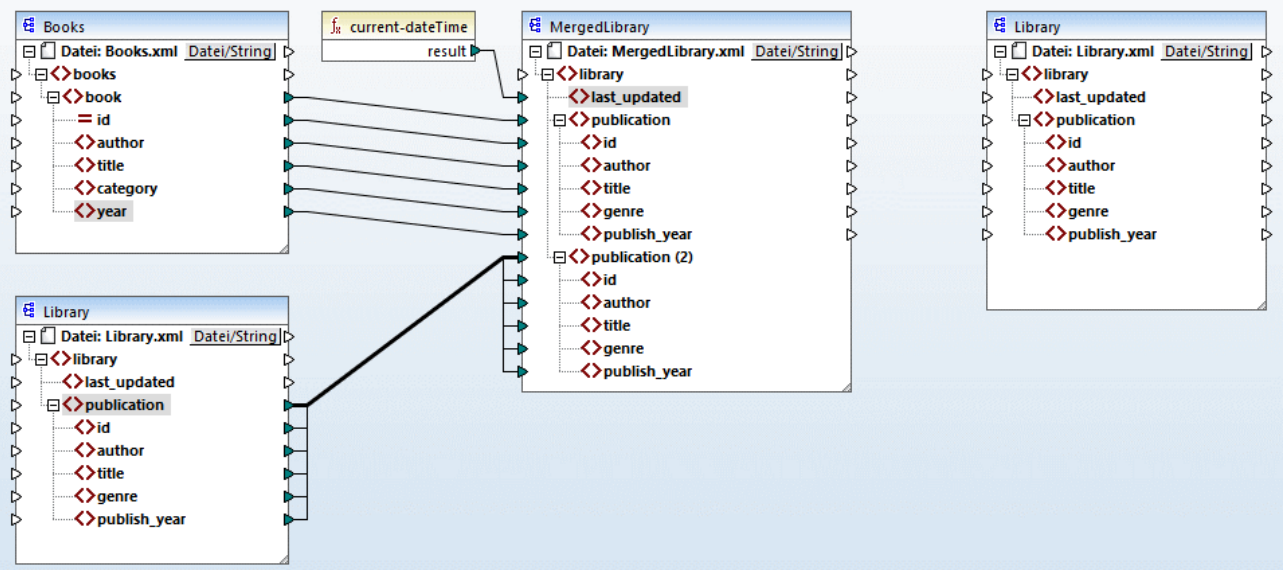
2.3.2 Konfigurieren der zweiten Zieldatei

Wir müssen nun die zweite Zieldatei (`FilteredLibrary.xml`), die nur eine Teilmenge der `<book>`-Elemente aus `MergedLibrary.xml` enthalten wird, hinzufügen und konfigurieren.

Hinzufügen der zweiten Zielkomponente

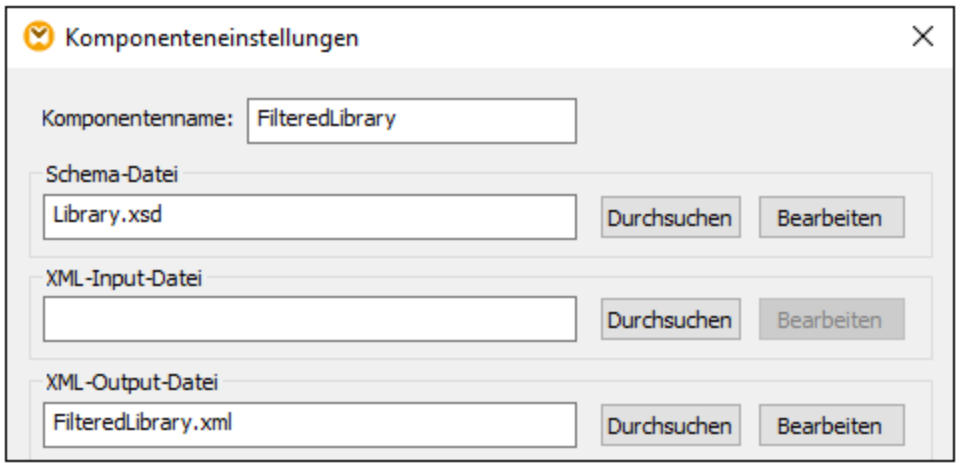
Um die zweite Zielkomponente hinzuzufügen, klicken Sie auf die Symbolleisten-Schaltfläche  (**XML-Schema/Datei einfügen**) und öffnen Sie die Datei `Library.xsd`. Klicken Sie auf **Überspringen**, wenn Sie von MapForce aufgefordert werden, eine Instanzdatei bereitzustellen. Die zweite Zielkomponente hat nur eine Struktur, aber keinen Inhalt. Wir werden die gefilterten Daten später auf diese Zieldatei mappen. Das Mapping-Design sieht nun folgendermaßen aus:

© 2016-2022 Altova GmbH



Konfigurieren der zweiten Zielkomponente

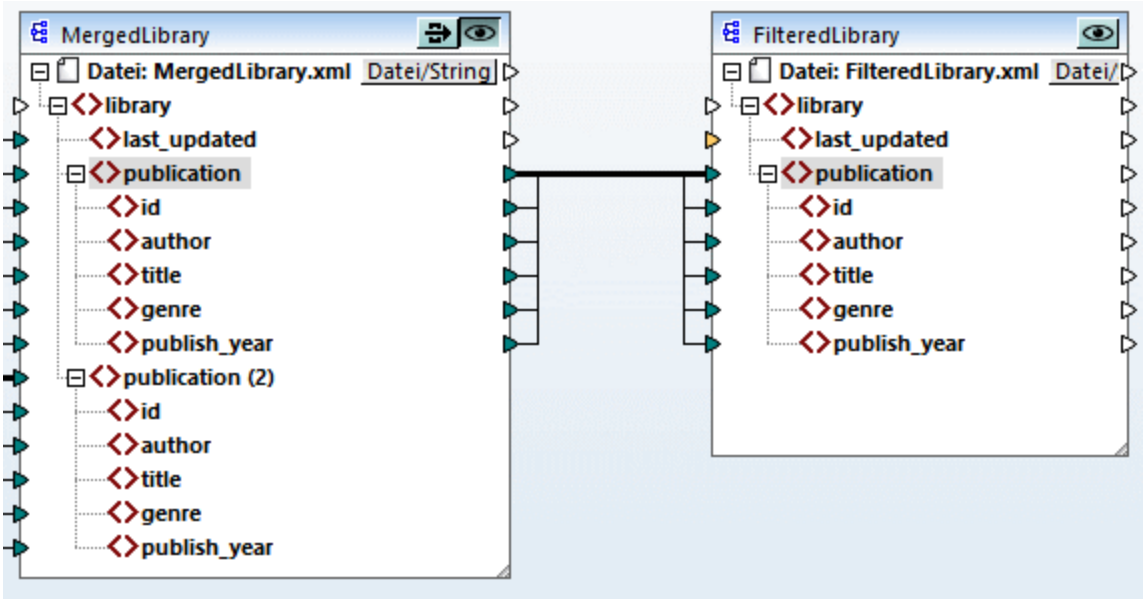
Wie oben gezeigt, hat das Mapping nun zwei Quellkomponenten (`Books` und `Library`) und zwei Zielkomponenten (`MergedLibrary` und `Library`). Aus Gründen der Klarheit benennen wir die neu hinzugefügte Komponente in `FilteredLibrary` *um.* Doppelklicken Sie dazu auf die Überschrift der Komponente ganz rechts und bearbeiten Sie die Komponenteneinstellungen[126] wie folgt:



2.3.3 Ziehen der Verbindungen

Im nächsten Schritt wird das Element `<publication>` in `MergedLibrary` auf das Element `<publication>` in `FilteredLibrary` gemappt. Wenn Sie den Output-Konnektor[77] von `MergedLibrary` mit dem Input-Konnektor von `FilteredLibrary` verbinden, wird die folgende Meldung angezeigt: "Sie haben im Mapping-Projekt mehrere

Zielkomponenten erstellt. Um eine Vorschau der Ausgabe einer bestimmten Zielkomponente anzuzeigen, klicken Sie in der Titelleiste der Komponente auf die Schaltfläche "Preview" und anschließend auf das Register XSLT oder Ausgabe-Vorschau, um das Ergebnis anzuzeigen." Klicken Sie auf **OK**. Beachten Sie, dass in der rechten oberen Ecke beider Zielkomponenten nun neue Schaltflächen zur Verfügung stehen: (**Vorschau**) und (**Weiterleitung**). Diese Schaltflächen werden in den folgenden Schritten verwendet und erklärt.



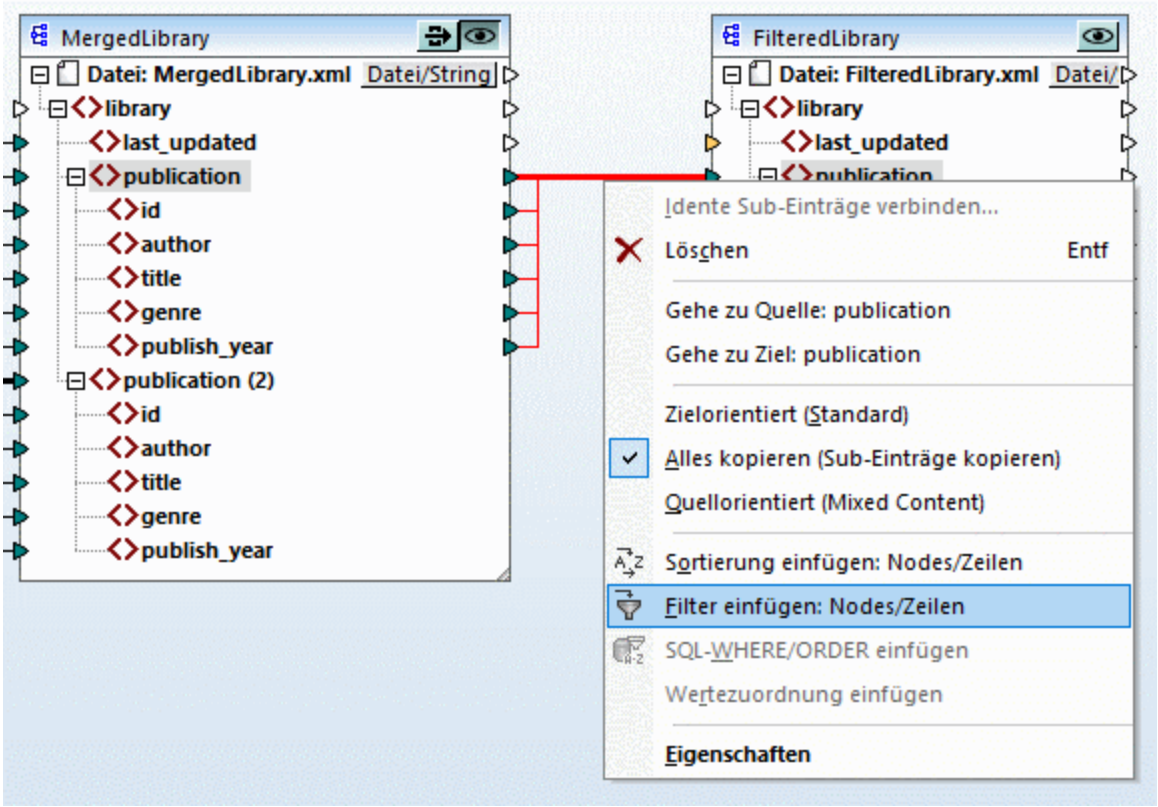
2.3.4 Filtern der Daten

In diesem Schritt werden wir die Daten aus `MergedLibrary` so filtern, dass nur die nach 1900 veröffentlichten Bücher an die Komponente `FilteredLibrary` übergeben werden. Zu diesem Zweck werden wir eine **Filter**-Komponente verwenden.

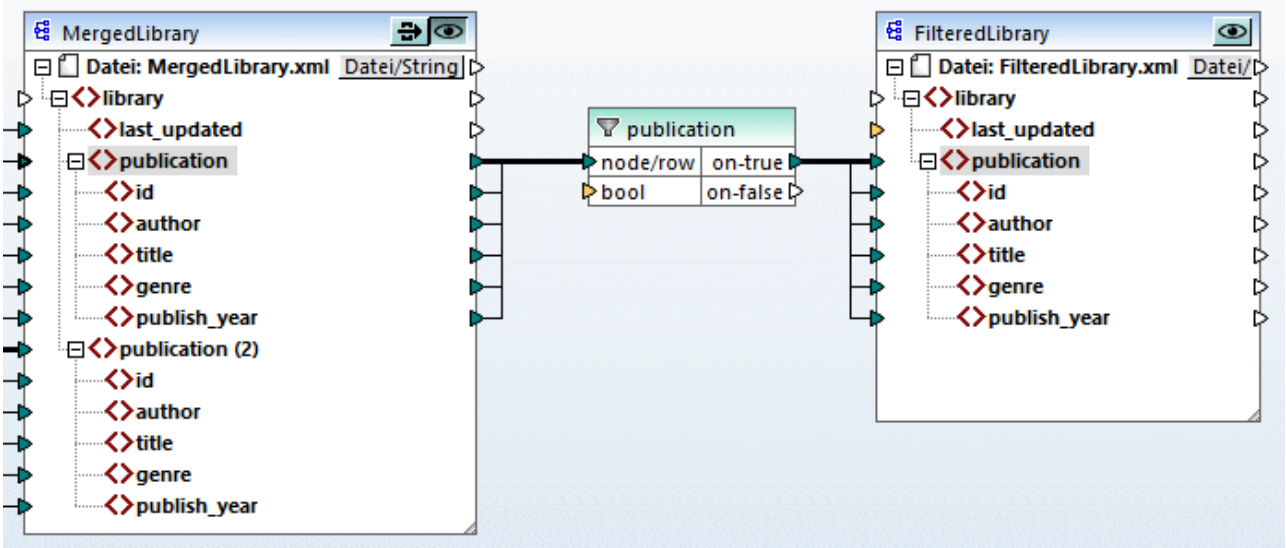
Hinzufügen eines Filters

Um einen Filter hinzuzufügen, klicken Sie mit der rechten Maustaste auf die Verbindung zwischen `MergedLibrary` und `FilteredLibrary` und wählen Sie im Kontextmenü den Befehl **Filter einfügen: Nodes/Zeilen**.

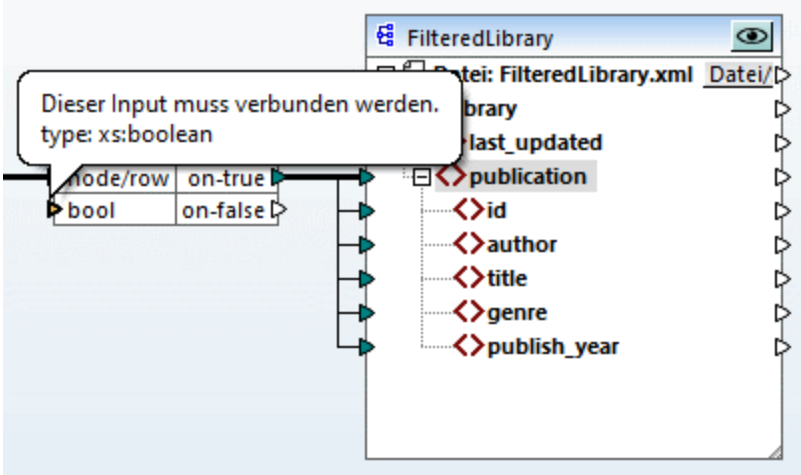
© 2016-2022 Altova GmbH



Daraufhin wird die Filterkomponente zum Mapping hinzugefügt (*siehe Abbildung unten*).



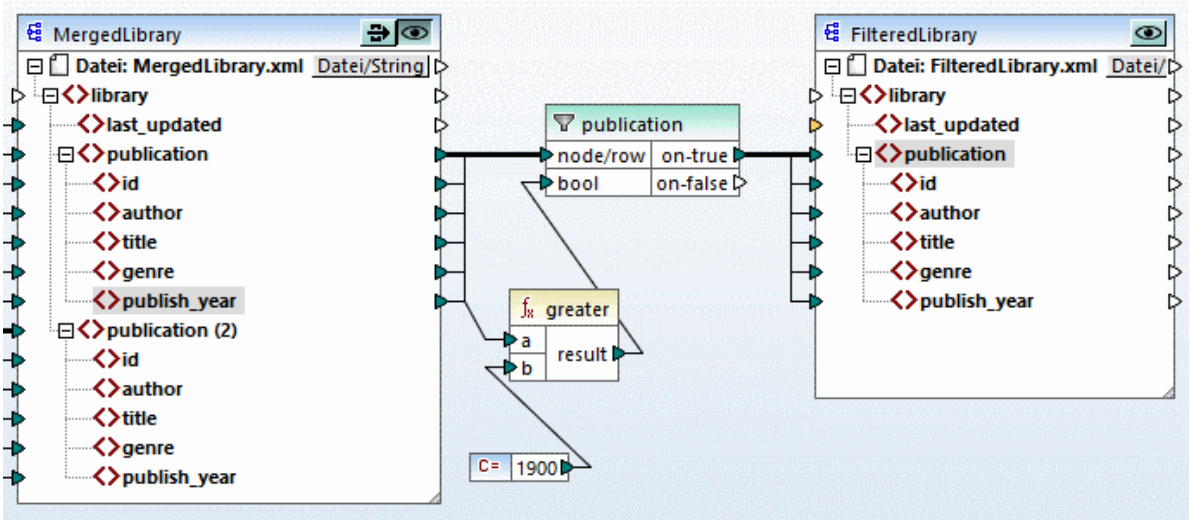
Wie oben gezeigt, erscheint der Input-Konnektor `bool` orange markiert, was bedeutet, dass ein Input zwingend erforderlich ist. Wenn Sie die Maus über den Konnektor platzieren, sehen Sie, dass ein Input vom Typ `xs:boolean` erforderlich ist (*siehe Abbildung unten*). Damit Tipps angezeigt werden, klicken Sie in der Symbolleiste auf  (**Tipps anzeigen**).



Nur Bücher nach 1900

Für die Filterkomponente wird eine Bedingung benötigt, die entweder `true` oder `false` zurückgibt. Wenn die Boolesche Bedingung `true` zurückgibt, werden die Daten der aktuellen `publication`-Sequenz in die Zielkomponente kopiert. Wenn die Bedingung `false` zurückgibt, werden die Daten nicht kopiert. In diesem Tutorial muss die Bedingung lauten, dass alle nach 1900 veröffentlichten Bücher gefiltert werden sollen. Gehen Sie folgendermaßen vor, um die Bedingung zu erstellen:

1. Klicken Sie im Menü **Einfügen** auf **Konstante**. Fügen Sie eine Konstante mit dem Wert *1900* hinzu. Wählen Sie als Typ **Zahl** aus.
2. Suchen Sie im Fenster **Bibliotheken** die Funktion `greater` und ziehen Sie sie in den Mapping-Bereich.
3. Erstellen Sie die Mapping-Verbindungen zu und von der Funktion `greater`, wie unten gezeigt. Dadurch geben Sie MapForce die Anweisung, das aktuelle Quellelement `<publication>` in das Element `<publication>` in der Zielkomponente zu kopieren, wenn `publish_year` größer ist als 1900.



© 2016-2022 Altova GmbH

2.3.5 Anzeige einer Vorschau und Speichern der Ausgabe

Sie können nun eine Vorschau der Ausgaben der einzelnen Zielkomponenten anzeigen und speichern. Wenn im selben Mapping mehrere Zielkomponenten vorhanden sind, können Sie auswählen, von welcher eine Vorschau angezeigt werden soll, indem Sie auf die Schaltfläche der jeweiligen Komponente klicken. Wenn die Schaltfläche **Vorschau** aktiviert ist, zeigt dies an, dass die jeweilige Komponente derzeit für die Vorschau aktiviert ist und dass diese Ausgabe im **Ausgabefenster** generiert wird. Die Vorschau kann immer nur für jeweils eine Komponente aktiviert werden.

Wenn Sie daher die Ausgabe der Zwischenkomponente `MergedLibrary` in der Vorschau sehen und speichern möchten, gehen Sie folgendermaßen vor:

1. Klicken Sie in der Komponente `MergedLibrary` **auf** .
2. Klicken Sie am unteren Rand des **Mapping**-Bereichs auf die Schaltfläche **Ausgabe**.
3. Klicken Sie im Menü **Ausgabe** auf **Ausgabedatei speichern,** wenn Sie die Ausgabe in einer Datei speichern möchten.

Wenn Sie die Ausgabe der Komponente `FilteredLibrary` in der Vorschau sehen und speichern wollen:

1. Klicken Sie in der Komponente `MergedLibrary` auf .
2. Klicken Sie in der Komponente `FilteredLibrary` auf .
3. Klicken Sie am unteren Rand des **Mapping**-Bereichs auf die Schaltfläche **Ausgabe**.
4. Klicken Sie im Menü **Ausgabe** auf **Ausgabedatei speichern,** wenn Sie die Ausgabe in einer Datei speichern möchten.

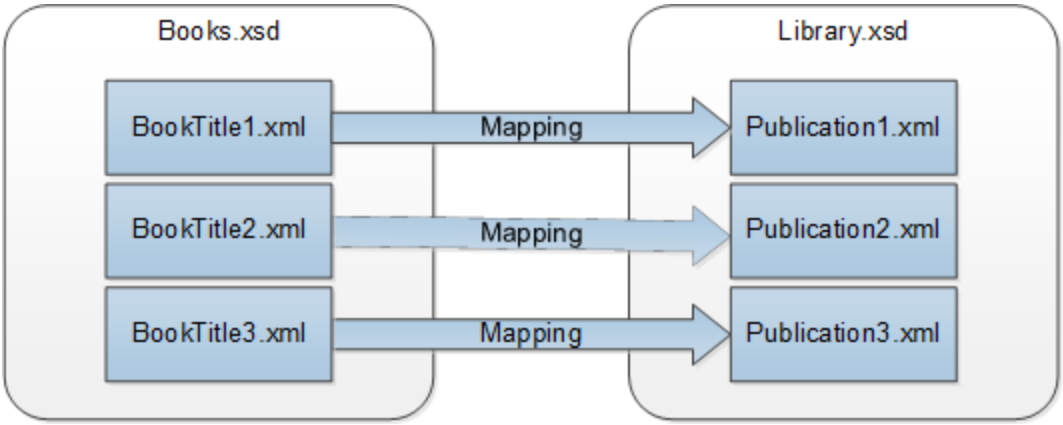
Sie haben nun ein Mapping erstellt, das zwei Zielkomponenten hat. Nähere Informationen zum Arbeiten mit Weiterleitungskomponenten finden Sie unter Verkettete Mappings 1299. Das Mapping-Design aus diesem Tutorial wurde unter dem Namen **`Tut3_ChainedMapping.mfd`** gespeichert.

2.4 Mehrere Quellkomponenten auf mehrere Zielkomponenten

In diesem Tutorial wird gezeigt, wie Sie mit derselben Transformation Daten aus mehreren Quelldateien auf mehrere Zieldateien mappen. Um dies zu veranschaulichen, werden wir nun ein Mapping erstellen, mit dem Folgendes bewerkstelligt werden soll:

1. Auslesen der Daten aus mehreren XML-Dateien im selben Verzeichnis.
2. Mappen der Schemas der einzelnen Dateien auf ein neues Schema.
3. Generieren einer neuen XML-Zieldatei mit dem neuen Schema für jede XML-Quelldatei.
4. Entfernen der XML- und Namespace-Deklaration aus den generierten Dateien.

In der Abbildung unten sehen Sie ein abstraktes Modell der in diesem Tutorial verwendeten Datentransformation:



Ausgangspunkt

Als XML-Quelldateien verwenden wir die drei Dateien `BookTitle1.xml`, `BookTitle2.xml` und `BookTitle3.xml`. Jede der drei Dateien basiert auf `Books.xsd` und es ist darin ein einziges Buch gespeichert (*siehe unten*).

`BookTitle1.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Books.xsd">
   <book id="1">
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <category>Fiction</category>
      <year>1876</year>
   </book>
</books>
```

`BookTitle2.xml`

© 2016-2022 Altova GmbH

```
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Books.xsd">
   <book id="2">
      <author>Franz Kafka</author>
      <title>The Metamorphosis</title>
      <category>Fiction</category>
      <year>1912</year>
   </book>
</books>
```

BookTitle3.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Books.xsd">
   <book id="3">
      <author>Herman Melville</author>
      <title>Moby Dick</title>
      <category>Fiction</category>
      <year>1851</year>
   </book>
</books>
```

Weitere Schritte

Wir gehen nun folgendermaßen vor:

1. Zuerst muss das Quellschema (`Books.xsd`) auf das Zielschema (`Library.xsd`) gemappt werden.
2. Nach der Transformation generiert das Mapping drei Dateien anhand dieses neuen Schemas (*siehe Codefragmente unten*).
3. Wir werden das Mapping so konfigurieren, dass die Namen der drei generierten Dateien `Publication1.xml`, `Publication2.xml` und `Publication3.xml` lauten. Beachten Sie, dass die XML-Deklaration und die Namespace-Deklaration entfernt werden müssen.

   Publication1.xml

   ```
   <library>
      <publication>
         <id>1</id>
         <author>Mark Twain</author>
         <title>The Adventures of Tom Sawyer</title>
         <genre>Fiction</genre>
         <publish_year>1876</publish_year>
      </publication>
   </library>
   ```

   Publication2.xml

   ```
   <library>
      <publication>
         <id>2</id>
         <author>Franz Kafka</author>
         <title>The Metamorphosis</title>
         <genre>Fiction</genre>
   ```

```
      <publish_year>1912</publish_year>
   </publication>
</library>
```

Publication3.xml

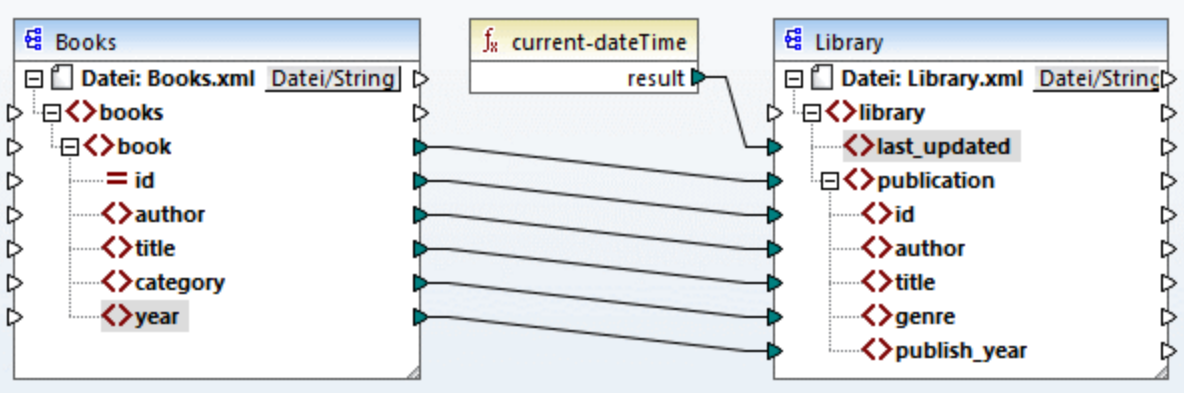
```
<library>
   <publication>
      <id>3</id>
      <author>Herman Melville</author>
      <title>Moby Dick</title>
      <genre>Fiction</genre>
      <publish_year>1851</publish_year>
   </publication>
</library>
```

Um die erforderliche Datentransformation durchzuführen, gehen Sie vor, wie in den nachfolgenden Unterabschnitten beschrieben.

2.4.1 Vorbereiten des Mapping-Designs

Die Ausgangsbasis in diesem Tutorial ähnelt dem Mapping `BooksToLibrary.mfd` aus dem ersten Tutorial [44], mit dem Unterschied, dass zu diesem Zeitpunkt keine XML-Dateien bereitgestellt werden. In diesem Schritt soll eine abstrakte Struktur (`Books.xsd`) auf eine andere abstrakte Struktur (`Library.xsd`) gemappt werden. Um das in der Abbildung unten dargestellte Szenario zu reproduzieren, gehen Sie folgendermaßen vor:

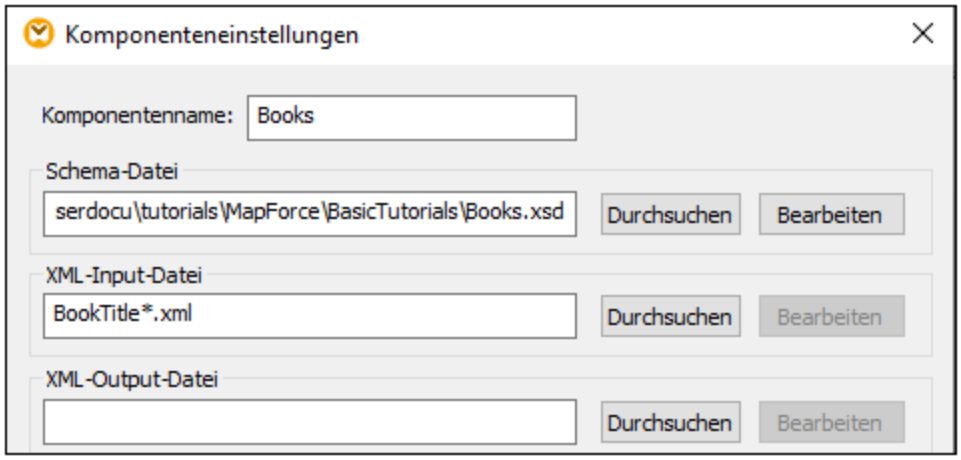
1. Öffnen Sie `Books.xsd`.
2. Wenn Sie aufgefordert werden, eine XML-Beispieldatei anzugeben, klicken Sie auf **Überspringen**.
3. Öffnen Sie `Library.xsd`.
4. Wenn Sie aufgefordert werden, eine XML-Beispieldatei anzugeben, klicken Sie auf **Überspringen**.
5. Verbinden Sie die entsprechenden Nodes, wie unten gezeigt.
6. Suchen Sie im Fenster **Bibliotheken** die Funktion `current-dateTime`.
7. Ziehen Sie die Funktion in den Mapping-Bereich und verbinden Sie `result` mit dem Element `last_updated`.



© 2016-2022 Altova GmbH

2.4.2 Konfigurieren des Input

Wir brauchen nun drei XML-Dateien, von denen jede ein Buch enthält, die alle auf demselben Schema **Books.xsd** basieren. Diese XML-Dateien mit den Namen **BookTitle1.xml**, **BookTitle2.xml** und **BookTitle3.xml** werden als Input-Dateien verwendet. Bevor Sie die Komponenteneinstellungen ändern, speichern Sie Ihr Mapping als **.mfd**-Datei im Ordner **BasicTutorials**. Damit in MapForce mehrere XML-Instanzdateien verarbeitet werden, doppelklicken Sie auf die Überschrift der Quellkomponente. Geben Sie im Dialogfeld Komponenteneinstellungen[126] **BookTitle*.xml**als Input-Datei ein. Aufgrund des Sternchens (*) im Dateinamen verwendet MapForce alle Dateien mit dem Präfix `BookTitle` als Mapping-Input. Da es sich um einen relativen Pfad handelt, sucht MapForce nach allen `BookTitle`-Dateien im selben Verzeichnis wie die Mapping-Datei. Sie können gegebenenfalls auch einen absoluten Pfad eingeben.

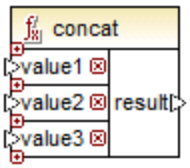


2.4.3 Konfigurieren des Ausgabeteils 1

In nächsten Schritt werden nun die Dateinamen für die einzelnen Ausgabedateien erstellt. Dazu verwenden wir die `concat`-Funktion, die alle als Argument gelieferten Werte miteinander verbindet. Wenn diese Werte miteinander verbunden werden, wird daraus ein Ausgabedateiname gebildet (z.B. `Publication1.xml`). Um die Dateinamen mit Hilfe der `concat`-Funktion zu generieren, gehen Sie folgendermaßen vor:

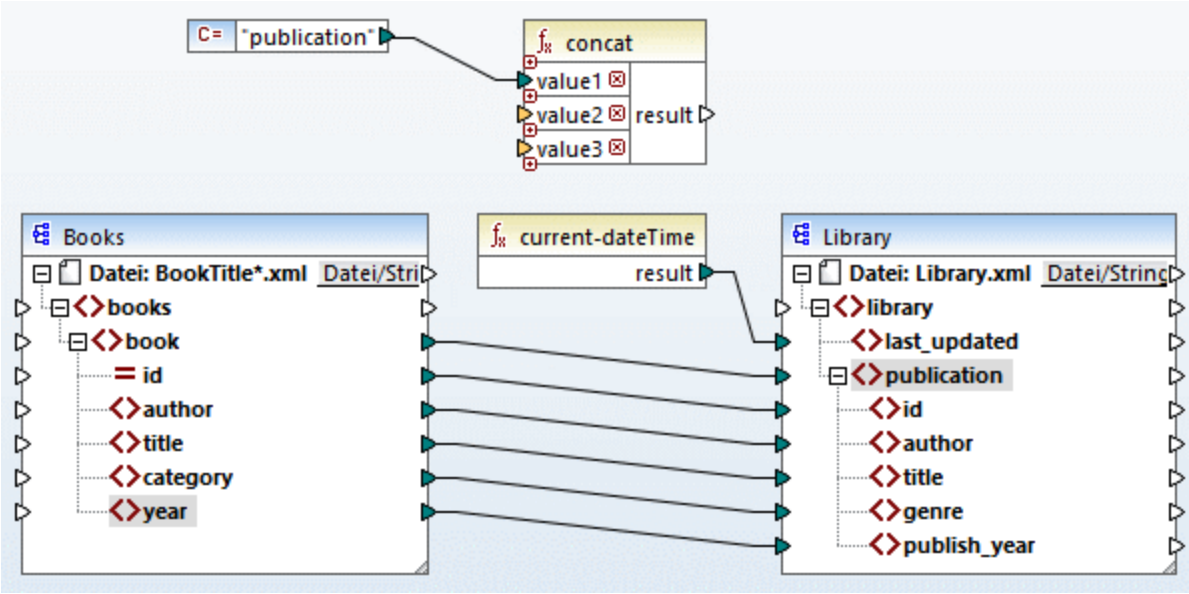
Hinzufügen der concat-Funktion

Suchen Sie die `concat`-Funktion im Fenster **Bibliotheken** und ziehen Sie sie in den Mapping-Bereich (*siehe Abbildung unten*). Diese Funktion hat standardmäßig zwei Parameter, wenn Sie zum Mapping hinzugefügt wird. Wir benötigen in unserem Beispiel drei Parameter. Klicken Sie in der Funktionskomponente auf ⊡ (**Parameter hinzufügen**) und fügen Sie einen dritten Parameter hinzu. Beachten Sie, dass Sie einen Parameter durch Klicken auf ⊠ (**Parameter löschen**) löschen können.



Einfügen einer Konstante

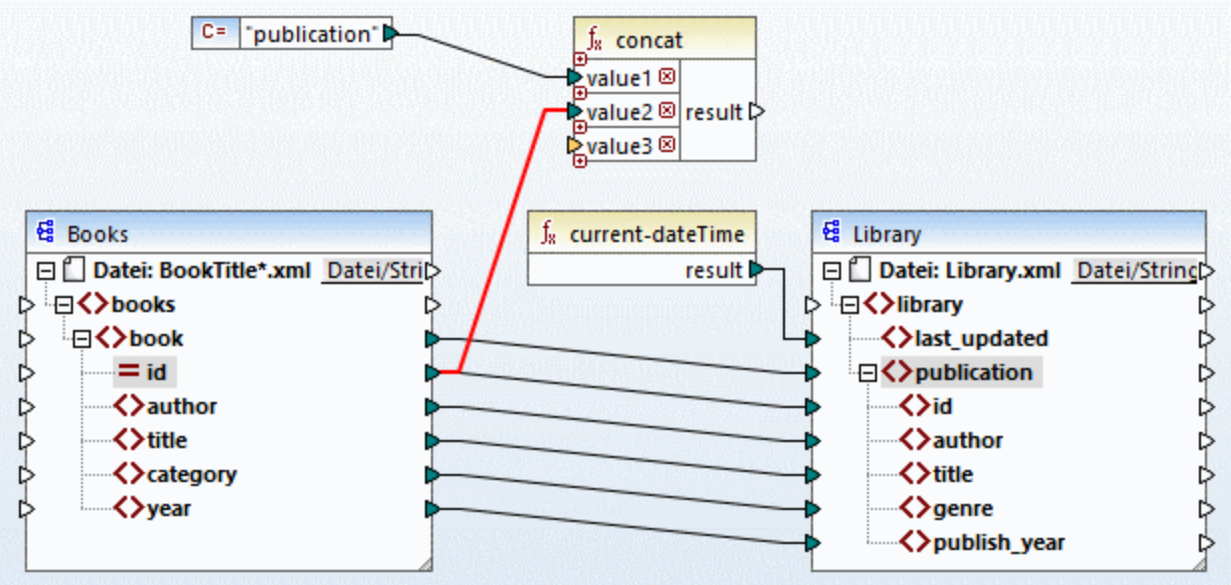
Um eine Konstante hinzuzufügen, klicken Sie im Menü **Einfügen** auf **Konstante**. Wenn Sie aufgefordert werden, einen Wert anzugeben, geben Sie `publication` ein und belassen Sie die Option **String** aktiviert. Die Konstante `publication` liefert den Konstanten-String-Wert `publication`. Verbinden Sie die Konstante mit `value1` der `concat`-Funktion, wie in der Abbildung unten gezeigt:



Angabe der ID

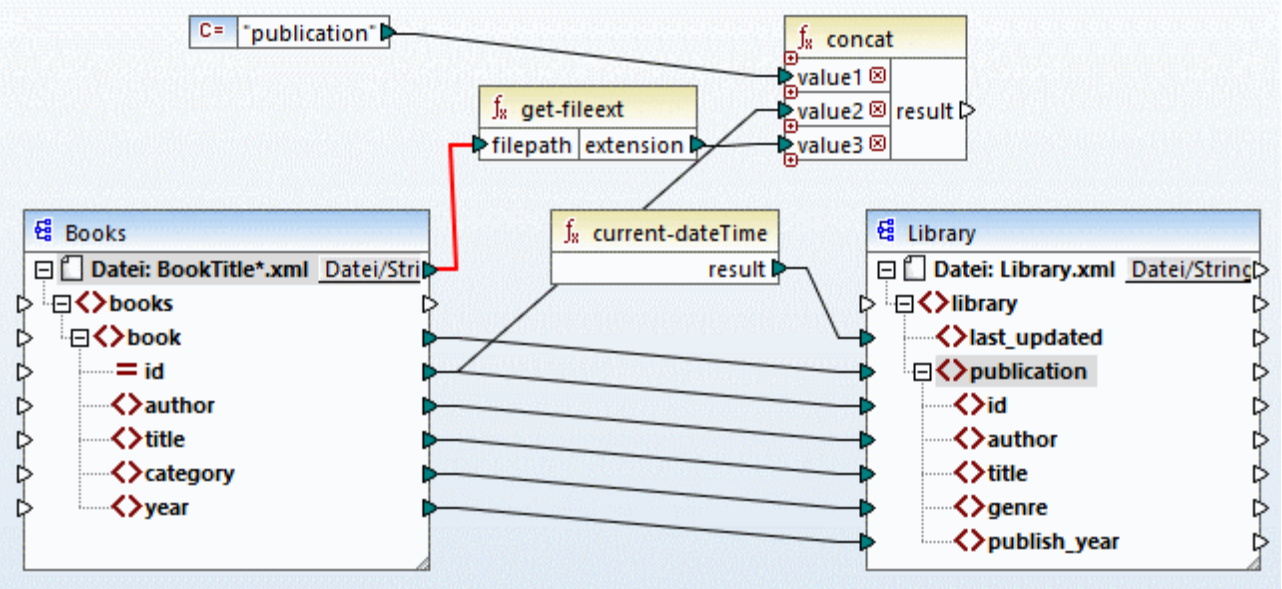
Verbinden Sie das Attribut `id` der Quellkomponente mit `value2` der `concat`-Funktion. Das Attribut `id` der XML-Quelldatei liefert den eindeutigen ID-Wert für die einzelnen Dateien. Dadurch wird verhindert, dass alle Dateien mit demselben Namen generiert werden. Die Verbindung wird rot angezeigt, wenn Sie darauf klicken.

© 2016-2022 Altova GmbH



Extraktion der Dateierweiterung

Suchen Sie im **Bibliotheksfenster** die Funktion `get-fileext` und ziehen Sie sie in den Mapping-Bereich. Erstellen Sie eine Verbindung vom obersten Node der Quellkomponente (**`Datei: BookTitle*.xml`**) zum Parameter `filepath` dieser Funktion. Verbinden Sie anschließend den Parameter `extension` der `get-fileext`-Funktion mit `value3` der `concat`-Funktion. Auf diese Art extrahieren Sie nur die Dateierweiterung (in diesem Fall `.xml`) aus dem Namen der Quelldatei und übergeben sie an den Namen der Ausgabedatei.

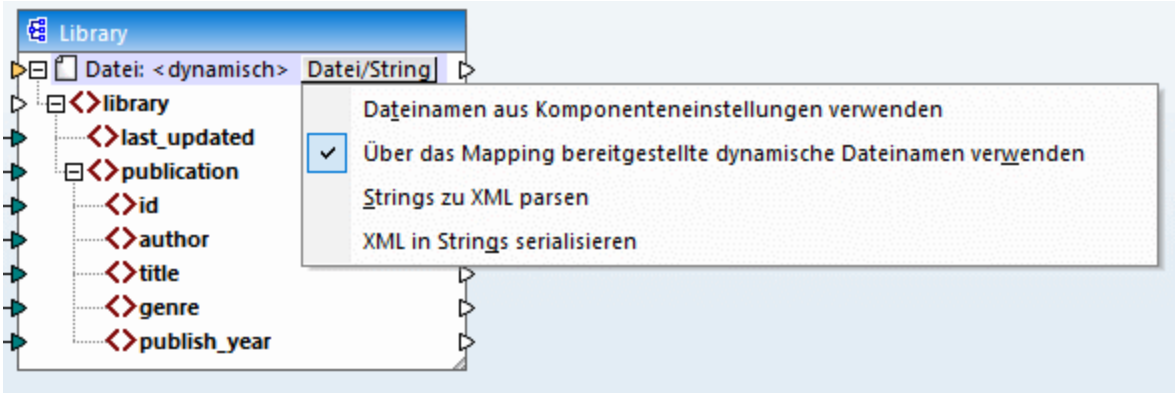


2.4.4 Konfigurieren des Ausgabeteils 2

Wir können MapForce nun anweisen, die Dateinamen bei der Ausführung des Mappings zu erstellen. Dazu verwenden wir dynamische Dateinamen (*siehe Unterabschnitte weiter unten*).

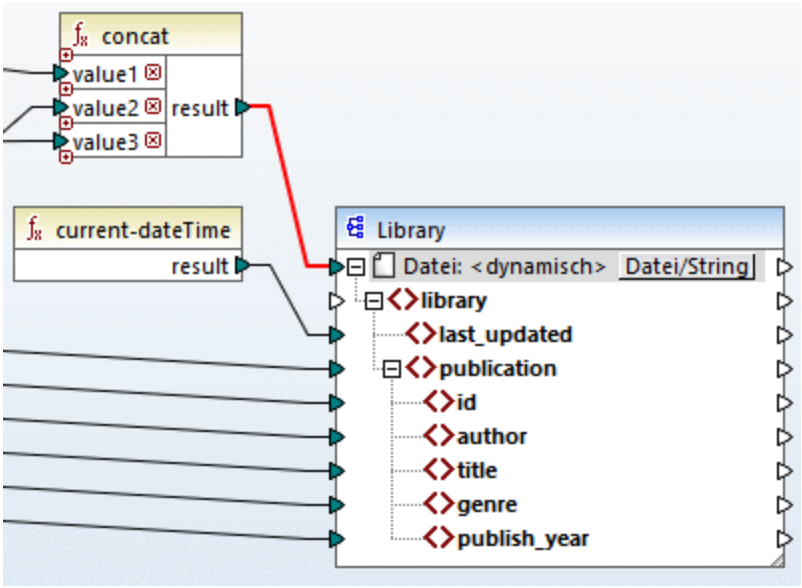
Dynamische Dateinamen

Wir wollen nun, dass MapForce die Instanzdateien dynamisch generiert, d.h. jede Ausgabedatei erhält ihren Namen auf Basis der durch die `concat`-Funktion bereitgestellten Argumente. Klicken Sie dazu auf die Schaltfläche Datei bzw. Datei/String der Zielkomponente und wählen Sie die Option **Über das Mapping bereitgestellte dynamische Dateinamen verwenden**.



Verbinden der Funktion mit dem dynamischen Node

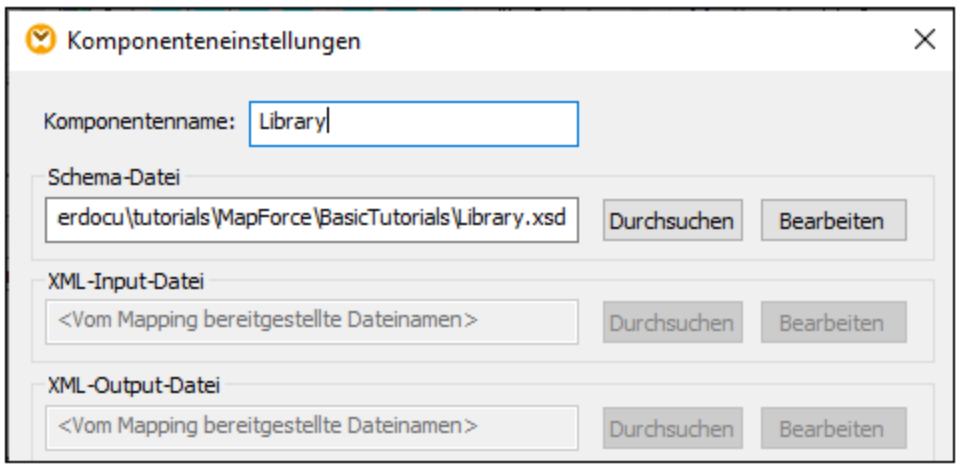
Im nächsten Schritt wird das Ergebnis der `concat`-Funktion mit dem Node `Datei: <dynamisch>` der Zielkomponente verbunden.



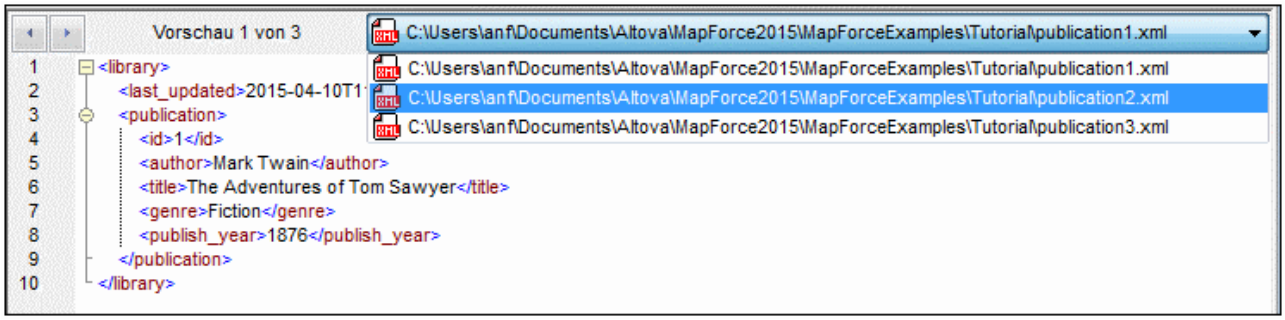
© 2016-2022 Altova GmbH

Überprüfen der Komponenteneinstellungen

Wenn Sie jetzt auf die Überschrift der Zielkomponente doppelklicken, sehen Sie, dass die Textfelder **XML-Input-Datei** und **XML-Output-Datei** deaktiviert sind und darin angezeigt wird *<Vom Mapping bereitgestellte Dateinamen>* (*siehe Abbildung unten*). Dies gibt an, dass die Instanzdateinamen dynamisch über ein Mapping bereitgestellt werden, daher müssen diese nicht mehr in den Komponenteneinstellungen definiert werden.



Sie können das Mapping nun ausführen und das Ergebnis sowie die Namen der generierten Dateien sehen. Mit diesem Mapping werden mehrere Ausgabedateien generiert. Um durch die Ausgabedateien zu navigieren, verwenden Sie die Pfeilschaltflächen in der linken oberen Ecke des **Ausgabefensters** oder wählen Sie eine Datei aus der daneben liegenden Dropdown-Liste aus (*siehe Abbildung unten*).

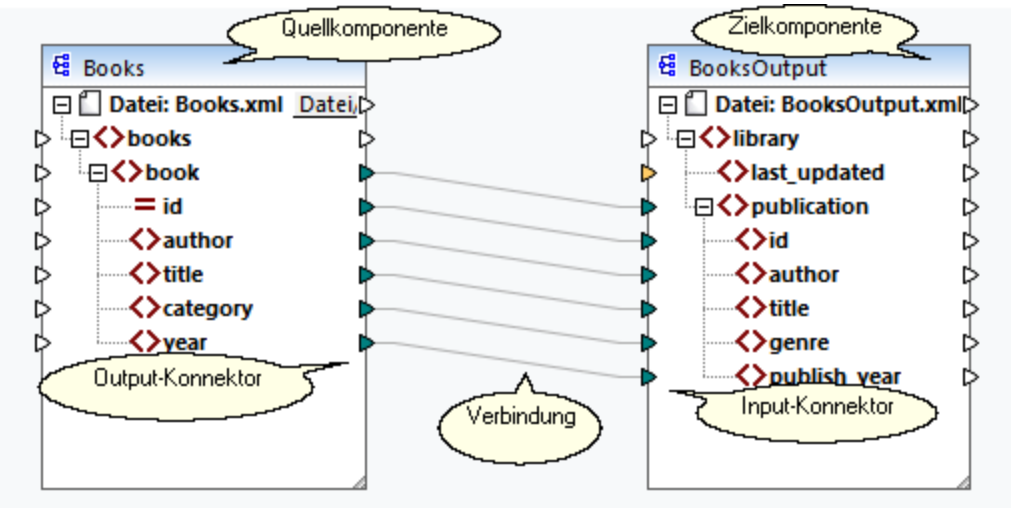


Das Mapping-Design aus diesem Tutorial wurde unter dem Namen **`Tut4_MultipleToMultiple.mfd`** gespeichert.

3 Mapping-Grundlagen

Ein MapForce Mapping-Design (oder "Mapping") ist eine visuelle Darstellung davon, wie Daten von einem Format in ein anderes transformiert werden sollen. Ein Mapping besteht aus *Komponenten*, die zum Mapping-Bereich hinzugefügt werden, um Datentransformationen zu erstellen. Ein gültiges Mapping besteht aus einer oder mehreren *Quellkomponenten*, die mit einer oder mehreren *Zielkomponenten* verbunden werden. Sie können ein Mapping ausführen und direkt in MapForce eine Vorschau auf das Ergebnis anzeigen. Sie können Code generieren und ihn extern ausführen. Sie können ein Mapping auch zu einer MapForce-Ausführungsdatei kompilieren und die Ausführung des Mappings mit MapForce Server oder FlowForce Server automatisieren. MapForce speichert Mappings als `.mfd`-Dateien.

In der Abbildung unten sehen Sie die grundlegende Struktur eines Mappings:



Neues Mapping

Um ein neues Mapping zu erstellen, klicken Sie in der Symbolleiste auf die Schaltfläche (**Neu**). Klicken Sie alternativ dazu im Menü **Datei** auf **Neu**. Wählen Sie anschließend **Mapping** aus und klicken Sie auf **OK**. Wir werden als nächstes Komponenten zum Mapping hinzufügen[80] und Verbindungen[89] zwischen den Komponenten erstellen.

Hauptbereiche eines Mappings

In den folgenden Unterabschnitten werden die Hauptbereiche eines Mapping-Designs beschrieben.

Komponente

Als *Komponente* wird in MapForce die visuelle Darstellung der Struktur Ihrer Daten bzw. die Art, wie die Daten transformiert werden sollen, bezeichnet. Komponenten bilden die zentralen Bestandteile eines jeden Mappings und werden in Form von rechteckigen Kästen dargestellt. Komponenten können in zwei große Gruppen unterteilt werden:

- Quell- und Zielkomponenten
- Struktur-[124] und Transformationskomponenten[866].

Beachten Sie, dass diese beiden Gruppen einander nicht gegenseitig ausschließen. In der ersten Gruppe werden die Beziehungen zwischen Komponenten dargestellt. So kann etwa eine Komponente eine Quellkomponente für eine Komponente und für eine andere eine Zielkomponente bilden. MapForce liest die

© 2016-2022 Altova GmbH

Daten aus einer Quellkomponente aus und schreibt sie in eine Zielkomponente. Bei der Ausführung des Mappings generiert MapForce anhand der Zielkomponente eine (oder mehrere Dateien) oder übergibt das Ergebnis als String-Wert für die weitere Verarbeitung an ein externes Programm. Unten sind die Komponentenarten aus der ersten Gruppe beschrieben:

- Eine *Quellkomponente* befindet sich auf der linken Seite des Mapping-Bereichs. Aus der Quellkomponente werden Daten ausgelesen.
- Eine *Zielkomponente* befindet sich rechts von der Quellkomponente. In die Zielkomponente werden Daten geschrieben.
- Eine *Weiterleitungskomponente* ist ein Subtyp von Quell- und Zielkomponenten. Eine Weiterleitungskomponente fungiert sowohl als Quell- als auch Zielkomponente. Nähere Informationen dazu finden Sie unter Verkettete Mappings 1299. Beachten Sie, dass nur Strukturkomponenten als Weiterleitungskomponenten fungieren können.

In der zweiten Gruppe wird dargestellt, ob eine Komponente eine Datenstruktur hat oder zum Transformieren von aus einer anderen Komponente gemappten Daten verwendet wird.

Nähere Informationen über Komponenten und Aktionen im Zusammenhang mit Komponenten finden Sie unter Komponenten 78.

Konnektor
Als Konnektor wird das kleine Dreieck auf der linken oder rechten Seite einer Komponente bezeichnet. Über die Input-Konnektoren auf der linken Seite einer Komponente werden Daten *in diese Komponente eingegeben*. Über die Output-Konnektoren auf der rechten Seite einer Komponente werden Daten *aus dieser Komponente* ausgegeben.

Verbindung
Eine Verbindung ist eine Linie, die Sie zwischen zwei MapForce-Konnektoren ziehen. Durch Erstellen von Verbindungen weisen Sie MapForce an, Daten auf eine bestimmte Art zu transformieren: z.B. Daten aus einem XML-Dokument zu lesen und in ein anderes XML-Dokument zu schreiben.

In diesem Abschnitt

In diesem Kapitel werden die häufigsten MapForce-Aufgaben und Konzepte beschrieben. Dieser Abschnitt ist in die folgenden Unterabschnitte gegliedert:

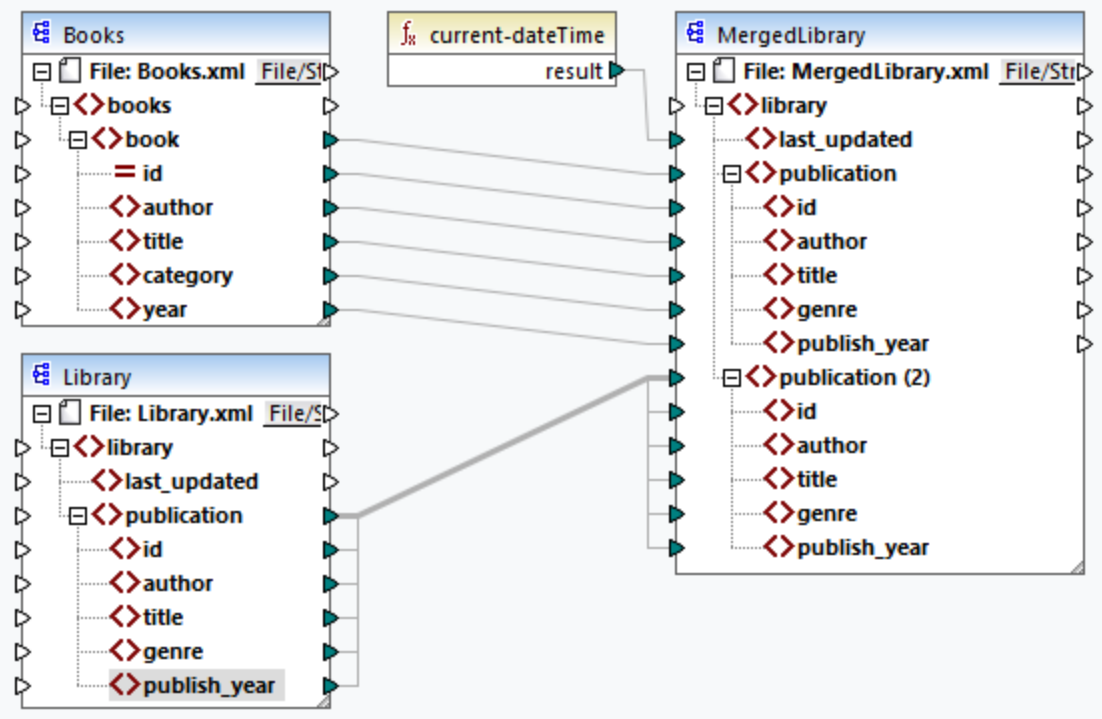
- Komponenten 78
- Verbindungen 89
- Allgemeine Verfahren und Funktionalitäten 107
- Projekte 119

3.1 Komponenten

Komponenten bilden das zentrale Element eines jeden Mapping-Designs in MapForce. Komponenten werden im Mapping-Bereich als rechteckige Kästen dargestellt. Dieses Kapitel enthält eine Übersicht über Struktur- und Transformationskomponenten (*siehe Beispiel unten*). Der Unterschied besteht darin, ob eine Komponente eine Datenstruktur hat oder zum Transformieren von Daten verwendet wird. Eine Beschreibung dieser beiden Arten finden Sie in den Unterabschnitten weiter unten. Siehe auch Mapping-Grundlagen 76.

Beispiel für Komponenten

Im unten gezeigten Beispiel-Mapping sehen Sie zwei Quellkomponenten (`Books` und `Library`), eine Zielkomponente (`MergedLibrary`) sowie eine Transformationskomponente (die Funktion **`current-dateTime`**).



Strukturkomponenten

Mit Strukturkomponenten wird eine abstrakte Struktur Ihrer Daten dargestellt (z.B. eine XML-Datei). Unter Strukturkomponenten 124 finden Sie eine Liste von Strukturkomponenten, die als Datenquellen und -ziele verwendet werden können. Mit Hilfe von Strukturkomponenten können Sie Daten aus Dateien oder anderen Quellen auslesen, Daten in Dateien oder andere Quellen schreiben und Daten in einer Zwischenphase des Mappings speichern (z.B. um eine Vorschau der Daten anzuzeigen). Die nachstehende Tabelle enthält eine Übersicht über Strukturkomponenten und die dazugehörigen Symbolleisten-Schaltflächen.

Symbol	Beschreibung
	XML-Komponente
	Textkomponente (*Professional und Enterprise Edition*)

© 2016-2022 Altova GmbH

Symbol	Beschreibung
	Datenbankkomponente (*Professional und Enterprise Edition* für SQL-Datenbanken; *Enterprise Edition* für NoSQL-Datenbanken)
	JSON-Komponente (*Enterprise Edition*)
	Microsoft Excel-Komponente (*Enterprise Edition*)
	WSDL-Komponente (*Enterprise Edition*)
	EDI-Komponente (*Enterprise Edition*)
	XBRL-Komponente (*Enterprise Edition*)
	Protocol Buffer (*Enterprise Edition*)

Transformationskomponenten

Mit Hilfe von Transformationskomponenten können Sie Daten transformieren[942], ein Mapping-Zwischenergebnis für die weitere Verarbeitung speichern[882], einen Wert durch einen anderen ersetzen[921] und Ihre Daten sortieren[908], gruppieren[931], miteinander verknüpfen[896] und filtern[914]. Des Weiteren können Sie eine Ausnahme[938] hinzufügen, wodurch das Mapping bei Erfüllung einer durch einen Filter definierten Bedingung gestoppt und eine Fehlermeldung angezeigt wird. Die nachstehende Tabelle enthält eine Übersicht über Transformationskomponenten und die dazugehörigen Symbolleisten-Schaltflächen.

Symbol	Beschreibung
	Einfache Input-Komponente
	Einfache Output-Komponente
	Filter-Komponente
	Sortierkomponente
	Vordefinierte Funktion
	Benutzerdefinierte Funktion
	SQL/NoSQL-WHERE/ORDER-Komponente (*Professional und Enterprise Edition*)
	Wertezuordnungskomponente
	Variable
	Webservice-Funktion (*Enterprise Edition*)
	Ausnahme (*Professional und Enterprise Edition*)
	Konstante

Symbol	Beschreibung
	If-Else-Bedingung
	Join-Komponente (*Professional und Enterprise Edition*)

In diesem Abschnitt

Dieser Abschnitt enthält eine Übersicht über Komponenten und ist in die folgenden Kapitel gegliedert:

- Hinzufügen von Komponenten 80
- Komponentengrundlagen 82
- Dateipfade 84

3.1.1 Hinzufügen von Komponenten

In diesem Kapitel wird erklärt, wie Sie Komponenten zu einem Mapping hinzufügen. Um eine Komponente hinzuzufügen, müssen Sie zuerst ein neues Mapping-Design erstellen 76. Wählen Sie anschließend eine der folgenden Methoden:

- Wählen Sie im Menü **Einfügen** einen Komponententyp aus (z.B. **XML-Schema/Datei**).
- Ziehen Sie eine Datei aus dem Windows Datei-Explorer in den Mapping-Bereich. Beachten Sie, dass dies nur bei kompatiblen dabeibasierten Komponenten möglich ist.
- Klicken Sie in der **Komponente einfügen**-Symbolleiste auf die entsprechende Schaltfläche (*siehe Abbildung unten*).



Jede Komponente hat einen bestimmten Zweck und weist ein bestimmtes Verhalten auf. Eine Übersicht über Komponenten finden Sie unter Komponenten 78. Nähere Informationen über Datenstrukturen, die als Quell- und Zielkomponenten verwendet werden können, finden Sie unter Strukturkomponenten 124. Informationen zu vordefinierten MapForce-Komponenten, mit denen Daten temporär gespeichert oder transformiert werden, finden Sie unter Transformationskomponenten 866.

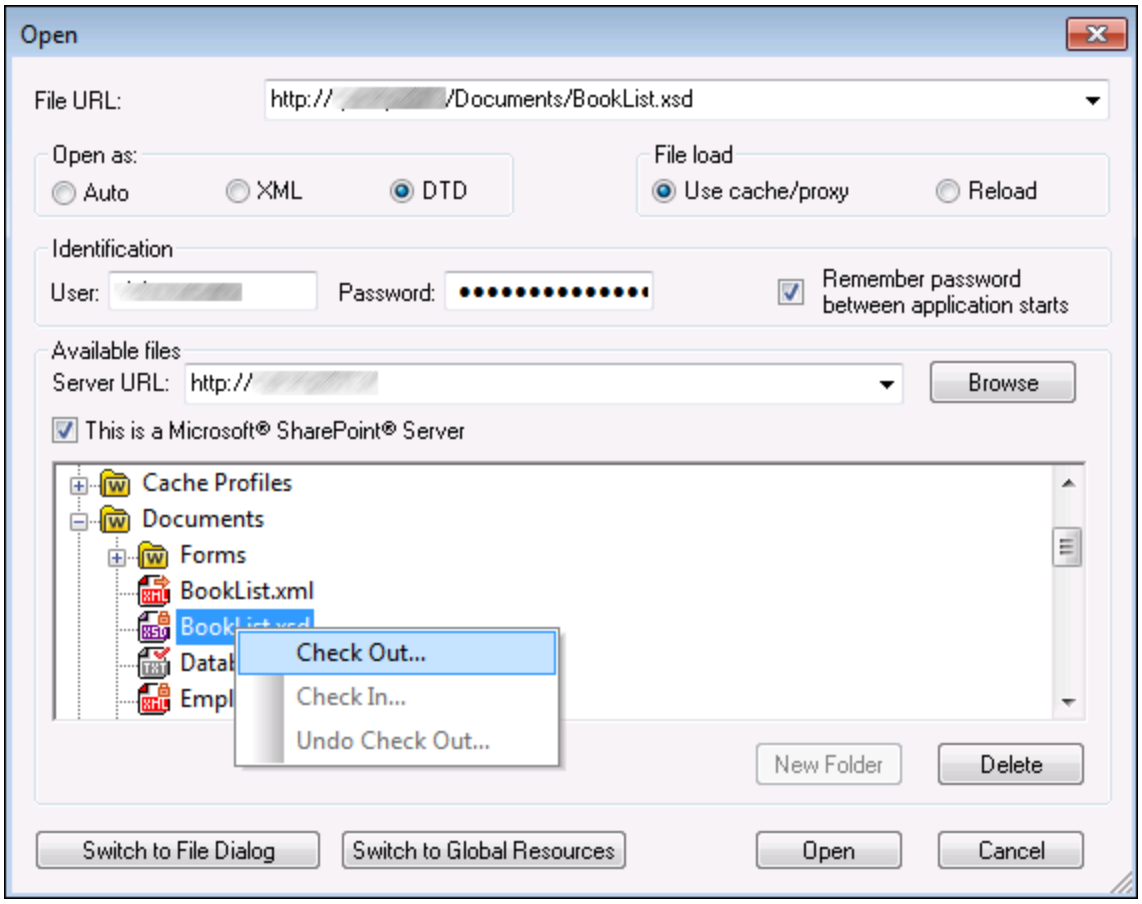
Eine vollständige Liste der Komponenten finden Sie in der Referenz der Komponentensymbole weiter unten.

Hinzufügen von Komponenten über eine URL

Sie können nicht nur lokale Dateien als Mapping-Komponenten hinzufügen, sondern Dateien auch über eine URL hinzufügen. Beachten Sie, dass dieser Vorgang unterstützt wird, wenn Sie eine Datei als Quellkomponente 76 hinzufügen. Es werden die Protokolle HTTP, HTTPS und FTP unterstützt. Um eine Komponente über eine URL hinzuzufügen, gehen Sie folgendermaßen vor:

1. Wählen Sie im Menü **Einfügen** den gewünschten Komponententyp aus (z.B. **XML-Schema/Datei**).
2. Klicken Sie im Dialogfeld **Öffnen** (*Abbildung unten*) auf **Zu URL wechseln**.
3. Geben Sie die URL der Datei in das Textfeld **Datei-URL** ein und klicken Sie auf **Öffnen**. Der Dateityp im Textfeld **Datei-URL** muss mit dem in Schritt 1 angegebenen Dateityp übereinstimmen.

© 2016-2022 Altova GmbH



In der folgenden Liste sind die verschiedenen Optionen des Dialogfelds **Öffnen** beschrieben.

- *Passwort speichern:* Falls für den Server eine Passwort-Authentifizierung erforderlich ist, werden Sie aufgefordert, den Benutzernamen und das Passwort einzugeben. Wenn sich MapForce Ihren Benutzernamen und Ihr Passwort für das nächste Mal merken soll, so geben Sie diese im Dialogfeld **Öffnen** ein und aktivieren Sie das Kontrollkästchen **Passwort speichern zwischen Applikationsstarts**.

- *Öffnen als:* Definiert die Grammatik für den Parser. Die Standardeinstellung, die auch die empfohlene Option ist, ist **Auto**.

- *Cache/proxy verwenden:* Wenn die Datei, die geladen wird, höchstwahrscheinlich nicht geändert wird, wählen Sie die Option **Cache/proxy verw.,** um Daten im Cache zu speichern und den Ladevorgang zu beschleunigen. Wenn Sie jedoch möchten, dass die Datei jedes Mal, wenn Sie das Mapping öffnen, neu geladen wird, wählen Sie die Option **Neu laden**.

- *Server URL:* Bei Servern mit WebDAV- (Web Distributed Authoring and Versioning)-Support können Sie Dateien nach Eingabe der Server-URL in das Textfeld **Server URL** und Klicken auf **Durchsuchen**, durchsuchen. In der Vorschau werden alle Dateitypen angezeigt, stellen Sie daher sicher, dass Sie denselben Dateityp wie in Schritt 1 auswählen. Andernfalls kommt es zu Fehlern.

- *Ein/Auschecken:* Wenn Sie einen Microsoft SharePoint Server verwenden, aktivieren Sie das Kontrollkästchen **Microsoft SharePoint Server**. Daraufhin wird im Vorschaubereich angezeigt, ob die

Datei eingecheckt oder ausgecheckt ist. Wenn Sie sicher stellen möchten, dass niemand anderer die Datei auf dem Server bearbeiten kann, während Sie diese verwenden, klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie den Befehl **Auschecken**(*siehe Abbildung oben*). Um eine von Ihnen vorher ausgecheckte Datei einzuchecken, klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie **Einchecken**.

3.1.2 Komponentengrundlagen

In diesem Kapitel wird beschrieben, wie Sie Strukturkomponenten[78] konfigurieren, durchsuchen und bearbeiten. Nähere Informationen dazu finden Sie in den Unterabschnitten weiter unten.

Ändern der Komponenteneinstellungen

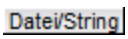
Nachdem Sie eine Komponente zum Mapping-Bereich hinzugefügt haben, können Sie die Einstellungen dafür über das Dialogfeld **Komponenteneinstellungen** konfigurieren. Sie können das Dialogfeld **Komponenteneinstellungen** auf eine der folgenden Arten öffnen:

- Doppelklicken Sie auf den Komponententitel.
- Wählen Sie die Komponente aus und klicken Sie im Menü **Komponente** auf **Eigenschaften**.
- Klicken Sie mit der rechten Maustaste auf den Komponententitel und wählen Sie **Eigenschaften**.

Beachten Sie, dass es von der Art der Komponente abhängt, welche Optionen zur Verfügung stehen. Im Folgenden finden die Liste der Einstellungen für die einzelnen Komponententypen.

- XML-Komponenteneinstellungen[126]
- Datenbank-Komponenteneinstellungen[286]
- CSV-Komponenteneinstellungen[595]
- Komponenteneinstellungen für Textdateien mit fester Länge[603]
- FlexText-Komponenteneinstellungen[698]
- JSON-Komponenteneinstellungen[825]
- Excel 2007+-Komponenteneinstellungen[752]
- XBRL-Komponenteneinstellungen[793]
- EDI-Komponenteneinstellungen[466]
- Protocol Buffer-Komponenteneinstellungen[847]

Bei allen dateibasierten Komponenten, wie z.B. einer XML-Datei wird neben dem Root-Node die Schaltfläche 

Datei (*Basic Edition*) bzw. 

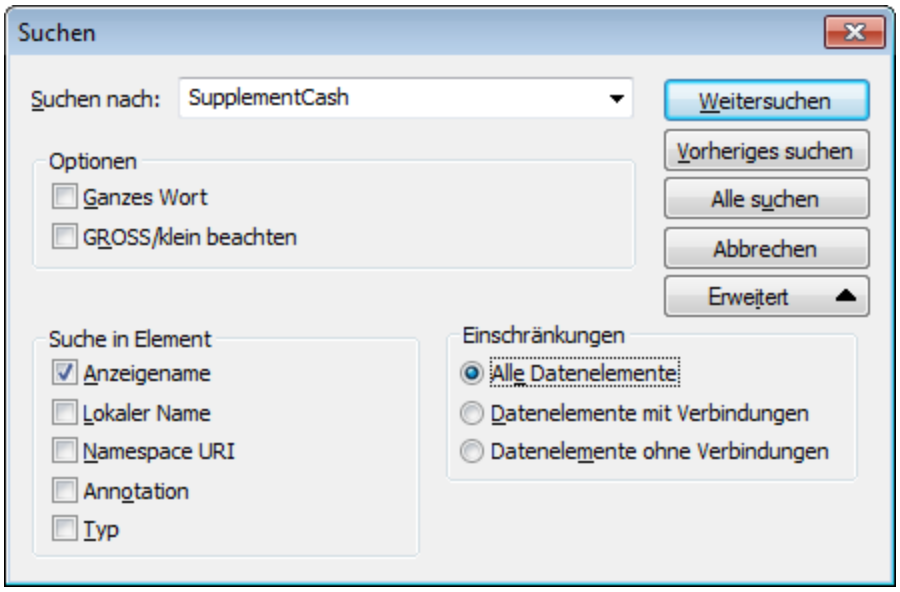
(*Professional und Enterprise Edition*) angezeigt. Über diese Schaltfläche werden erweiterte Optionen zum Verarbeiten oder Generieren mehrerer Dateien in einem einzigen Mapping angezeigt. Nähere Informationen dazu finden Sie unter Verarbeitung mehrerer Input- oder Output-Dateien[1371]. Zusätzlich dazu stehen Ihnen über diese Schaltfläche Optionen zum Parsen von Strings oder zum Serialisieren von Daten in Strings zur Verfügung[1382].

Durchsuchen einer Komponente

Um in einer Komponente nach einem bestimmten Node zu suchen, gehen Sie folgendermaßen vor:

1. Klicken Sie auf die gewünschte Komponente und drücken Sie die Tasten **Strg+F**.
2. Geben Sie einen Suchbegriff ein und klicken Sie auf **Weitersuchen/Vorheriges suchen/Alle suchen** (*siehe Abbildung unten*).

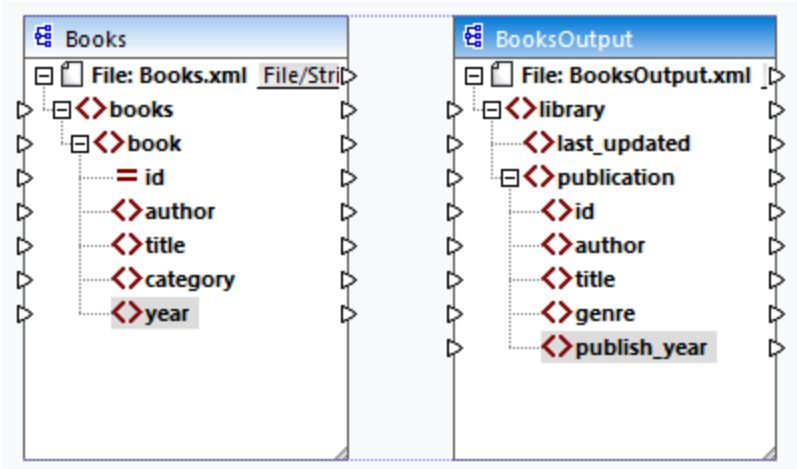
© 2016-2022 Altova GmbH



Über Option **Erweitert** können Sie festlegen, welche Datenelemente (Nodes) durchsucht werden sollen. Außerdem können Sie die Suchoptionen auf Basis bestimmter Verbindungen einschränken.

Ausrichten von Komponenten

Wenn Sie Komponenten im Mapping-Bereich verschieben, werden Hilfslinien (gepunktete Linien) zur Ausrichtung der Komponenten angezeigt (*siehe Abbildung unten*).



Um diese Option zu aktivieren, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Extras** auf **Optionen.**
2. Aktivieren Sie in der Gruppe **Bearbeiten** das Kontrollkästchen **Komponenten beim Ziehen mit der Maus aneinander ausrichten**.

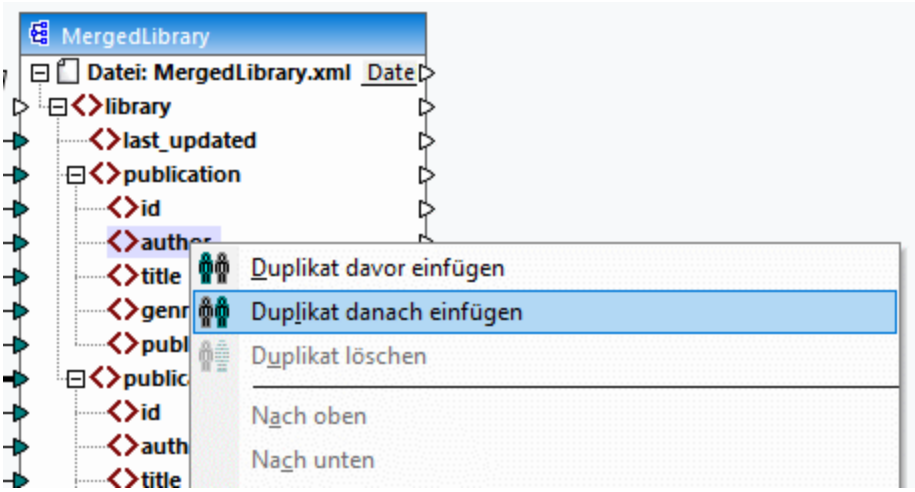
Input duplizieren

Manchmal muss eine Komponente so konfiguriert werden, dass sie Daten aus mehr als einer Datenquelle erhalten kann. Damit das Zielschema die Daten aus mehr als einem Quellschemas akzeptiert, können Sie jeden beliebigen Input-Node in Ihrer Zielkomponente duplizieren. Das Duplizieren von Input-Nodes ist nur bei Zielkomponenten sinnvoll. Duplizierte Nodes können Daten nur aufnehmen, Daten aus duplizierten Nodes können aber nicht gemappt werden. Sie können beliebig viele Nodes duplizieren.

Es gibt zwei Methoden, um einen Input zu duplizieren: (i) durch Auswahl des Befehls **Duplikat davor einfügen/Duplikat danach einfügen** aus dem Kontextmenü und (ii) durch Verbinden eines Quell-Node mit einem Ziel-Node, der bereits mit einem anderen Node verbunden ist. Die erste Option ist unten beschrieben. Informationen über die zweite Option finden Sie im zweiten Tutorial 54.

Duplikat davor/danach einfügen

Um einen bestimmten Input-Node zu duplizieren, klicken Sie mit der rechten Maustaste darauf und wählen Sie im Kontextmenü den Befehl **Duplikat davor einfügen/Duplikat danach einfügen** (*siehe Abbildung unten*). In der Abbildung oben wird der Node `author` dupliziert, damit Sie Daten von einem anderen Quell-Node darauf mappen können.



Anmerkung: Das Duplizieren von XML-Attributen ist unzulässig, da die XML-Instanz dadurch ungültig würde.

3.1.3 Dateipfade

Ein Mapping-Design (**`*.mfd`**) kann Referenzen zu mehreren Schema- und Instanzdateien enthalten. Anhand der Schemadateien ermittelt MapForce die Struktur der zu mappenden Daten und validiert diese. In der MapForce Professional und Enterprise Edition können Mappings auch Referenzen zu StyleVision Power Stylesheet-Dateien (**`*.sps`**) enthalten. Anhand dieser Dateien werden Daten für Ausgabeformate wie PDF, HTML und Word generiert. Mappings können auch Referenzen zu dateibasierten Datenbanken wie Microsoft Access oder SQLite haben.

Referenzen zu Dateien werden von MapForce erstellt, wenn Sie eine Komponente zum Mapping hinzufügen. Sie können solche Pfadreferenzen jedoch gegebenenfalls immer manuell einstellen oder ändern.

© 2016-2022 Altova GmbH

Dieser Abschnitt enthält eine Anleitung, wie Sie Pfade zu anderen von einem Mapping referenzierten Dateitypen konfigurieren oder ändern. Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

- Relative und absolute Pfade 85
- Pfade in Ausführungsumgebungen 87

3.1.3.1 Relative und absolute Pfade

In diesem Kapitel wird erläutert, wie Sie in den von Ihrer Komponente referenzierten Dateien absolute und relative Pfade verwenden. Ein absoluter Pfad enthält den vollständigen Pfad zu einer Datei, beginnend mit dem Root-Verzeichnis (*siehe Beispiel XML-Komponenten unten*). Ein relativer Pfad gibt den Pfad der Datei relativ zu aktuellen Arbeitsverzeichnis an, z.B. **`Books.xml`**.

Sie können im Dialogfeld **Komponenteneinstellungen** (*siehe Beispiel unten*) absolute oder relative Pfade für verschiedene Dateien, die von der Komponente referenziert werden können, definieren. Im Folgenden finden Sie eine Liste dieser Dateien:

- Input-Dateien, aus denen MapForce Daten ausliest.
- Output-Dateien, in die MapForce Daten schreibt.
- Schema-Dateien (gilt für Komponenten, die ein Schema haben)
- Strukturdateien, die als Input- oder Output-Parameter von benutzerdefinierten Funktionen oder Variablen verwendet werden.
- StyleVision Power Stylesheet (**`*.sps`**)-Dateien, anhand derer Daten für Ausgabeformate wie PDF, HTML und Word formatiert werden.
- Datenbankdateien bei Datenbankkomponenten (*Professional und Enterprise Edition*)

Kopieren-Einfügen und relative Pfade
Wenn Sie eine Komponente aus einem Mapping in ein anderes kopieren, wird überprüft, ob sich die relativen Pfade von Schema-Dateien anhand des Ordners des Ziel-Mappings auflösen lassen. Wenn die Pfade nicht aufgelöst werden können, werden Sie aufgefordert, die relativen Pfade zu absoluten zu machen.

Falsche Pfade
Wenn Sie eine Dateireferenz zu einem Mapping hinzufügen oder diese ändern und der Pfad nicht aufgelöst werden kann, zeigt MapForce eine Warnmeldung an. In den folgenden Fällen kann es zu falschen Pfadreferenzen kommen:

- Wenn Sie relative Pfade verwenden und die Mapping-Datei anschließend in ein neues Verzeichnis verschieben, ohne die Schema- und Instanzdatei ebenfalls zu verschieben.
- Wenn Sie absolute Pfade zu Dateien, die sich im selben Verzeichnis wie die Mapping-Datei befinden, verwenden und das Verzeichnis anschließend in einen anderen Ordner verschieben.

Wenn dies passiert, markiert MapForce die Komponente rot. Um das Problem zu beheben, doppelklicken Sie auf die Überschrift der Komponente und aktualisieren Sie die fehlerhaften Pfadreferenzen im Dialogfeld **Komponenteneinstellungen**. Siehe auch Ändern der Komponenteneinstellungen 82.

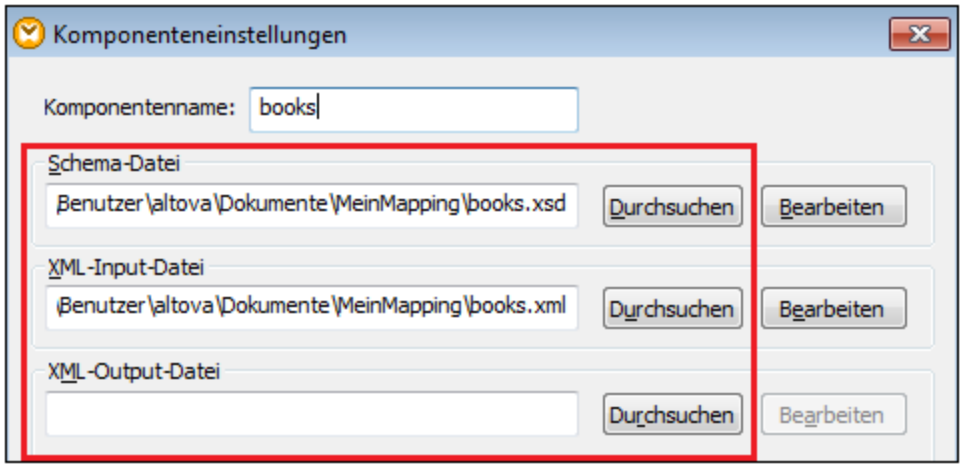
Beispiel: XML-Komponente

Im Beispiel unten wird gezeigt, wie Dateipfade in einer XML-Komponente verwendet werden. Wenn Sie alle mit dem Mapping in Zusammenhang stehenden Dateien relativ zur Mapping-Datei (**`.mfd`**) speichern möchten,

aktivieren Sie das Kontrollkästchen *Alle Dateipfade relativ zur MFD-Datei speichern* am unteren Rand des Dialogfelds **Komponenteneinstellungen**. Dies ist eine Standardoption, die empfohlen wird. Sie wirkt sich auf alle von der Komponente referenzierten Dateien (*in der Abbildung unten rot umrandet*) aus. Wenn Sie Ihr Mapping noch nicht gespeichert haben, werden im Dialogfeld **Komponenteneinstellungen** absolute Pfade zum Schema und/oder den Instanzdateien angezeigt. Damit im Dialogfeld **Komponenteneinstellungen** relative Pfade angezeigt werden, gehen Sie folgendermaßen vor:

1. Erstellen Sie ein neues Mapping[76] und fügen Sie eine Strukturkomponente[80], z.B. eine XML-Datei, der ein XML-Schema zugewiesen wurden, hinzu.
2. Doppelklicken Sie auf die Titelleiste der Komponente, um das Dialogfeld **Komponenteneinstellungen** zu öffnen.
3. Aktivieren Sie am unteren Rand des Dialogfelds **Komponenteneinstellungen** das Kontrollkästchen *Alle Dateipfade relativ zur MFD-Datei speichern*.
4. Speichern Sie Ihr Mapping.
5. Sie können die **Komponenteneinstellungen** nun erneut öffnen, um die relativen Pfade in den jeweiligen Textfeldern zu überprüfen.

Anmerkung: Pfade, die kein lokales Laufwerk referenzieren und Pfade, in denen eine URL verwendet wird, werden nicht relativ gemacht.



Wenn das Kontrollkästchen *Alle Dateipfade relativ zur MFD-Datei speichern* aktiviert ist, aktualisiert MapForce die von der Komponente referenzierten Dateien auch dann, wenn Sie das Mapping in einem anderen Ordner speichern. Wenn sich alle Dateien im selben Verzeichnis wie das Mapping befinden, funktionieren die Pfadreferenzen weiterhin, auch wenn Sie das gesamte Verzeichnis in einen anderen Ordner auf dem Rechner verschieben.

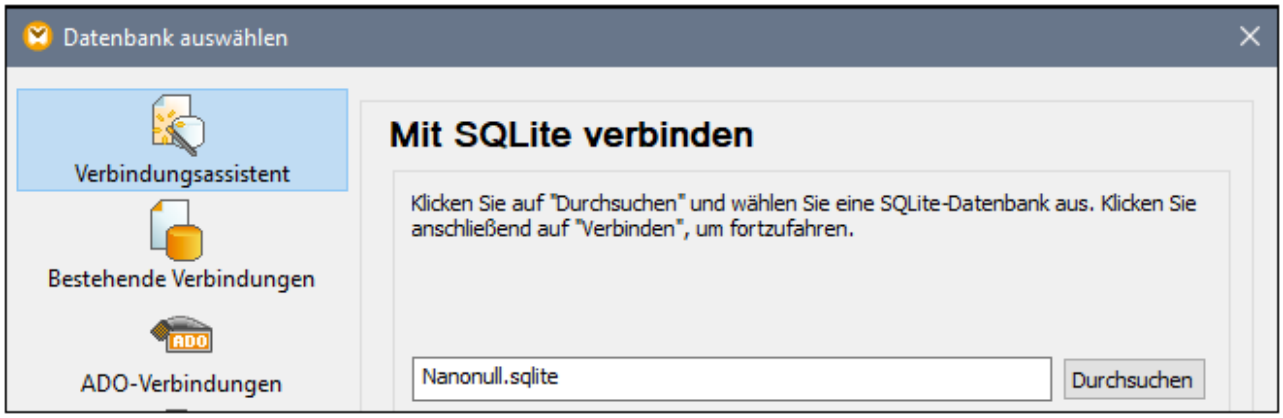
Die Einstellung *Alle Dateipfade relativ zur MFD-Datei speichern* wird auf die folgenden Dateien angewendet:

- Von komplexen Input- oder Output-Parametern benutzerdefinierter Funktionen verwendete Strukturdateien und Variablen vom Typ "complex"
- Input- oder Output-Dateien (*Professional and Enterprise Edition*).
- Von Datenbankkomponenten, die XML-Felder unterstützen, referenzierte Schema-Dateien (*Professional und Enterprise Edition*).
- Datenbankdateien (*Professional und Enterprise Edition*)
- XBRL-, FlexText-, EDI-, Excel 2007+-, JSON-Input- oder Output-Dateien (*nur Enterprise Edition*)

© 2016-2022 Altova GmbH

Beispiel: Datenbankkomponente (Professional und Enterprise Edition)

Wenn Sie eine Datenbankdatei wie Microsoft Access oder SQLite zum Mapping hinzufügen, können Sie im Dialogfeld **Datenbank auswählen** anstelle eines absoluten Pfads einen relativen Pfad eingeben (*siehe Abbildung unten*). Bevor Sie relative Dateipfade eingeben, stellen Sie sicher, dass Sie die **.mfd**-Datei zuerst speichern. Wenn Sie den Pfad einer Datenbankkomponente, die bereits im Mapping verwendet wird, ändern möchten, klicken Sie im Dialogfeld **Komponenteneinstellungen** auf **Ändern**. Nähere Informationen zum Herstellen einer Verbindung zu einer Datenbankquelle finden Sie unter Starten des Datenbankverbindungsassistenten 169.



Anmerkung: Wenn Sie Programmcode generieren oder MapForce Server-Ausführungsdateien (**.mfx**) kompilieren oder wenn Sie das Mapping auf FlowForce Server bereitstellen, wird ein relativer Pfad in einen absoluten Pfad konvertiert, wenn in den Mapping-Einstellunge 116n das Kontrollkästchen *Pfade im generierten Code absolut machen* aktiviert ist. Nähere Informationen dazu finden Sie unter Pfade im generierten Code 87.

3.1.3.2 Pfade in Ausführungsumgebungen

Wenn Sie Code anhand von Mappings generieren, Mappings zu MapForce Server-Ausführungsdateien (**.mfx**) kompilieren oder auf FlowForce Server bereitstellen, werden die generierten Dateien von der gewählten Zielumgebung, z.B. RaptorXML Server, MapForce Server oder einer C#-Applikation ausgeführt. Damit das Mapping erfolgreich ausgeführt werden kann, müssen alle relativen Pfade auch in der Umgebung, in der das Mapping ausgeführt wird, funktionieren. Im Folgenden sehen Sie die Basispfade für die einzelnen Zielsprachen.

Zielsprache	Basispfad
XSLT, XSLT2, XSLT3	Pfad der XSLT-Datei
XQuery*	Pfad der XQuery-Datei
C++, C#, Java*	Arbeitsverzeichnis der generierten Applikation
Built-In* (bei Anzeige einer Mapping-Vorschau in MapForce)	Pfad der Mapping-Datei (**.mfd**).

Zielsprache	Basispfad
Build-In* (bei Ausführung des Mappings mit MapForce Server)	Das aktuelle Arbeitsverzeichnis
Built-In* (bei Ausführung des Mappings mit MapForce Server durch FlowForce Server)	Das Arbeitsverzeichnis für den Auftrag oder das Arbeitsverzeichnis von FlowForce Server.

** In der MapForce Professional und Enterprise Edition verfügbare Sprachen*

Konvertierung relativer Pfade in absolute
Wenn Sie Programmcode generieren oder MapForce Server-Ausführungsdateien (**.mfx**) kompilieren oder wenn Sie das Mapping auf FlowForce Server bereitstellen, wird ein relativer Pfad in einen absoluten Pfad konvertiert, wenn in den Mapping-Einstellungen[116] das Kontrollkästchen **Pfade im generierten Code absolut machen** aktiviert ist.

Wenn Sie Code generieren und das Kontrollkästchen aktiviert ist, löst MapForce alle relativen Pfade anhand des Verzeichnisses der **.mfd**-Datei auf und macht sie im generierten Code zu absoluten Pfaden. Diese Einstellung wirkt sich auf die Pfade der folgenden Dateien aus:

- Input- und Output-Instanzdateien für alle dateibasierten Komponenten
- Access- und SQLite-Datenbankdateien, die als Mapping-Komponenten verwendet werden (*Professional und Enterprise Edition*).

Bibliothekspfade im generierten Code

Mapping-Dateien können optional Pfadreferenzen zu verschiedenen Bibliotheken enthalten. So können etwa benutzerdefinierte Funktionen aus anderen Mapping-Dateien oder Funktionen aus benutzerdefinierten XSLT-, XQuery-*, C#-* oder Java-Bibliotheken* aus **.mff***-Dateien (MapForce Funktionsdateien) importiert werden. Nähere Informationen dazu finden Sie unter Verwalten von Funktionsbibliotheken.[946].

** In der MapForce Professional und Enterprise Edition verfügbare Funktionalitäten*

Die Option **Pfade im generierten Code absolut machen** gilt nur für Mapping-Komponenten und wirkt sich nicht auf die Pfade zu externen Bibliotheken aus. Pfade zu allen anderen Bibliotheken als XSLT und XQuery werden im generierten Code in absolute Pfade konvertiert. Wenn Ihre Mapping-Datei z.B. Bibliotheksreferenzen wie .NET-, **.dll**- oder Java-Klassendateien enthält und der generierte Code in einer anderen Umgebung ausgeführt werden soll, so müssen die referenzierten Bibliotheken in der Zielumgebung unter demselben Pfad vorhanden sein.

Wenn Sie beabsichtigen, eine XSLT- oder XQuery-Datei anhand eines Mappings zu generieren, können Sie den Bibliothekspfad folgendermaßen relativ zur generierten XSLT- oder XQuery-Datei machen:

1. Öffnen Sie die Mapping-Einstellungen[116].
2. Aktivieren Sie das Kontrollkästchen **Bibliotheken relativ zu den generierten XSLT / XQuery-Dateien referenzieren**. Stellen Sie sicher, dass die XSLT- oder XQuery-Bibliotheksdatei unter diesem Pfad auch tatsächlich vorhanden ist.

© 2016-2022 Altova GmbH

3.2 Verbindungen

Eine Verbindung ist eine Linie, die eine Quellkomponente mit einer Zielkomponente verbindet. Verbindungen zeigen visuell an, wie Daten von einem Node auf einen anderen gemappt sind. In den Unterabschnitten weiter unten werden verschiedene Aktionen beschrieben, die Sie im Zusammenhang mit Verbindungen durchführen können.

Erstellen, Kopieren, Löschen einer Verbindung

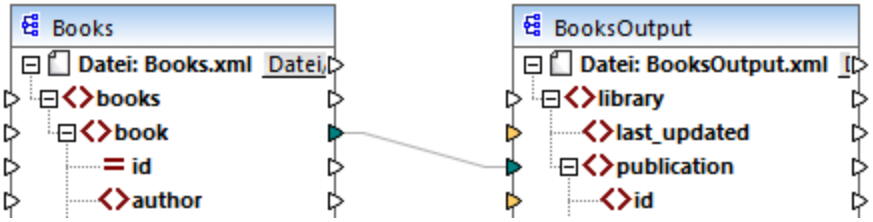
Um eine Verbindung zwischen zwei Datenelementen zu erstellen, klicken Sie auf den Output-Konnektor[77] eines Quell-Node und ziehen Sie ihn auf einen Ziel-Node. Ein Input-Konnektor kann *nur eine* eingehende Verbindung haben. Wenn Sie versuchen, eine zweite Verbindung zum selben Input hinzuzufügen, werden Sie gefragt, ob Sie die Verbindung durch eine neue ersetzen oder das Input-Datenelement duplizieren[84] möchten. Ein Output-Konnektor kann mehrere Verbindungen zu unterschiedlichen Inputs haben.

Um eine Verbindung auf ein anderes Datenelement zu kopieren, klicken Sie auf den dick angezeigten Abschnitt des Verbindungsendes (*siehe Abbildung in Verschieben einer Verbindung*) und ziehen Sie ihn bei gedrückter **Strg**-Taste auf das ausgewählte Zieldatenelement.

Um eine Verbindung zu löschen, klicken Sie darauf und drücken Sie die Taste **Löschen**. Klicken Sie alternativ dazu mit der rechten Maustaste auf die Verbindung und wählen Sie im Kontextmenü den Befehl **Löschen**.

Obligatorische Inputs

Als Hilfe beim Mapping werden obligatorische Inputs in Zielkomponenten orange markiert. Im Beispiel unten sehen Sie, dass bei Verbindung des Elements `book` der Komponente `Books` mit dem Element `publication` der Komponente `BookOutput` die Konnektoren der obligatorischen Nodes der Komponente `BooksOutput` markiert werden. Wenn obligatorische Inputs nicht verbunden werden, werden die entsprechenden Nodes nicht auf die Zielkomponente gemappt und das Mapping wird ungültig.



Fehlende Parent-Verbindungen

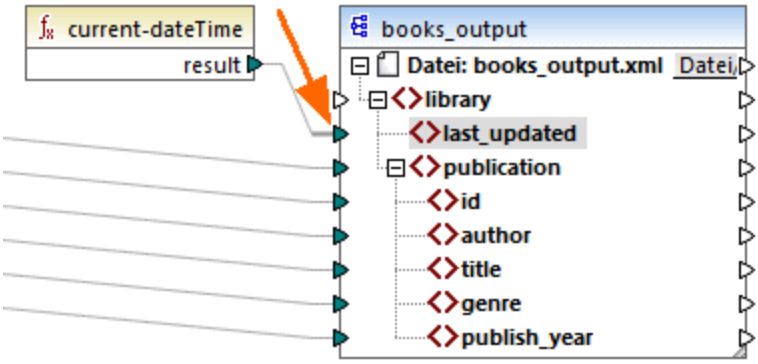
Wenn Sie manuell eine Verbindung zwischen einem Quell- und einem Ziel-Node erstellen, analysiert MapForce die möglichen Mapping-Ergebnisse. Wenn Sie zwei Child-Nodes verbinden, ohne ihre Parent-Nodes zu verbinden, wird eine Meldung angezeigt, in der vorgeschlagen wird, auch den Parent des Quell-Node mit dem Parent des Ziel-Node zu verbinden. Dadurch wird verhindert, dass eventuell nur ein einziger Child-Node im **Ausgabefenster** angezeigt wird.

Um die Anzeige solcher Meldungen zu deaktivieren, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Extras** auf **Optionen.**
2. Öffnen Sie die Gruppe **Meldungen**.
3. Deaktivieren Sie das Kontrollkästchen **Verbindung von übergeordneten Datenelementen beim Verbinden von Datenelementen vorschlagen**.

Verschieben einer Verbindung

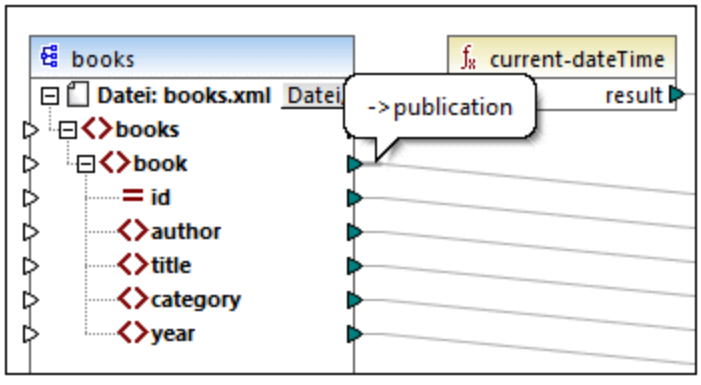
Um eine Verbindung auf einen anderen Node zu verschieben, klicken Sie auf den dick angezeigten Abschnitt des Verbindungsendes (*siehe Abbildung unten*) und ziehen Sie ihn auf den ausgewählten Ziel-Node.



Tooltips zu Verbindungen anzeigen

Mit Hilfe von Tooltips zu Verbindungen könne Sie die Namen von (i) Nodes, auf die Daten gemappt werden oder (ii) Nodes, von denen aus Daten gemappt werden, anzeigen. Damit Tipps angezeigt werden, aktivieren Sie die Symbolleisten-Schaltfläche  (**Tipps anzeigen**). Um die Namen von Nodes, auf die Daten gemappt werden, zu sehen, platzieren Sie den Cursor über den dicken Abschnitt einer Verbindung neben dem Output-Konnektor (*siehe Abbildung unten*). Um den Namen eines Node, von dem aus Daten gemappt werden, zu sehen, platzieren Sie den Cursor über den dicken Abschnitt einer Verbindung in der Nähe des Input-Konnektors. Wenn vom selben Output aus mehrere Verbindungen definiert wurden, werden im Tooltip maximal zehn Datenelementnamen angezeigt.

Im Beispiel unten hat der Ziel-Node, auf den die Daten aus dem Element `book` gemappt werden, den Namen `publication`.



Ändern der Verbindungseinstellungen

Um die Verbindungseinstellungen zu ändern, wählen Sie eine der folgenden Methoden:

© 2016-2022 Altova GmbH

- Wählen Sie eine Verbindung aus. Klicken Sie anschließend im Menü **Verbindung** auf **Eigenschaften**
- Doppelklicken Sie auf die Verbindung.
- Klicken Sie mit der rechten Maustaste auf die Verbindung und wählen Sie den Befehl **Eigenschaften**.

Nähere Informationen dazu finden Sie unter Verbindungseinstellungen 99.

Selektives Markieren von Verbindungen

Sie können in MapForce Verbindungen in einem Mapping selektiv hervorheben. Diese Funktionalität ist nützlich, wenn Ihr Mapping mehrere Komponenten mit mehreren Verbindungen aufweist. Durch selektives Hervorheben der Verbindungen können Sie besser überprüfen, ob die Nodes der ausgewählten Komponente korrekt gemappt wurden. Beachten Sie, dass mit dem für die Symbolleisten-Schaltflächen verwendeten Begriff *Konnektor* eine Verbindung gemeint ist, d.h. eine Verbindungslinie zwischen den Komponenten-Nodes. Siehe verfügbare Optionen unten.

	**Ausgewählte Komponentenkonnektoren anzeigen** (nur direkte Verbindungen)
	**Konnektoren zwischen Quelle und Ziel anzeigen** (direkte und indirekte Verbindungen)

Nur direkte Verbindungen

Wenn die Schaltfläche **Direkte Verbindungen** *nicht* aktiv ist, werden alle Verbindungen schwarz angezeigt. Wenn die Schaltfläche **Direkte Verbindungen** aktiv ist, werden nur Verbindungen, die mit der aktuell ausgewählten Komponente in Zusammenhang stehen, schwarz angezeigt. Andere Verbindungen erscheinen hellgrau.

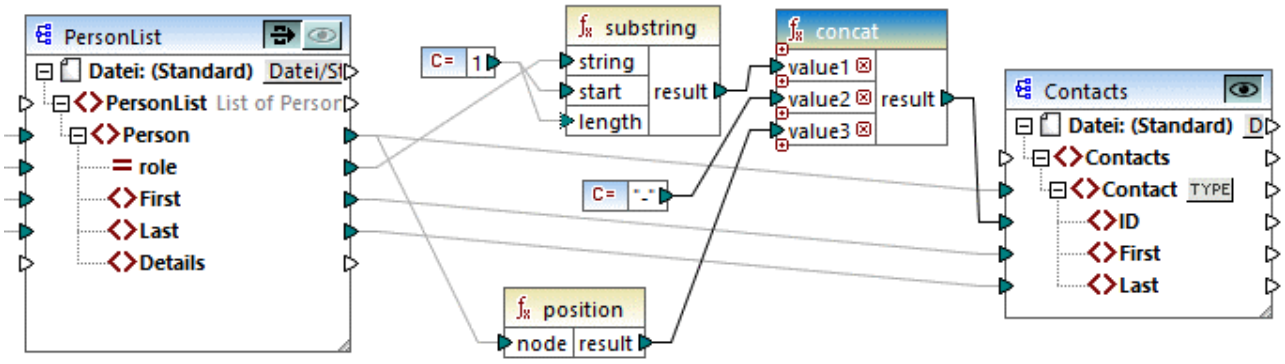
Direkte und indirekte Verbindungen

Die Schaltfläche **Quelle-Ziel-Verbindungen** steht nur zur Verfügung, wenn die Schaltfläche **Direkte Verbindungen** aktiv ist. Wenn die Schaltfläche **Quelle-Ziel-Verbindungen** gedrückt wird, können Sie Verbindungen der aktuell ausgewählten Komponente, einschließlich ihrer direkten Verbindungen und der Verbindungen der damit verbundenen Komponenten bis zu den Quell- und Zieldateien zurückverfolgen.

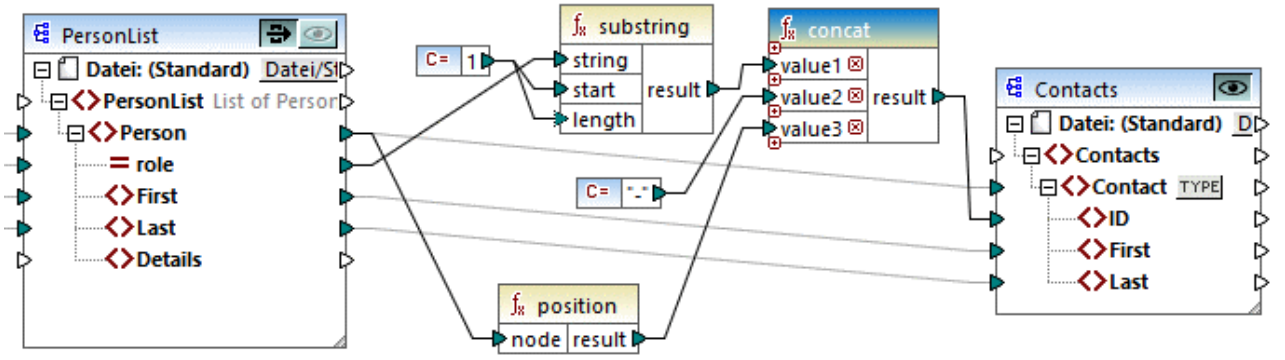
Ein Beispiel für die Funktionsweise dieser beiden Optionen finden Sie unten.

Beispiel

In der Abbildung unten sehen Sie einen Ausschnitt aus dem Mapping **`ChainedPersonList.mfd`** aus dem Ordner **`MapForceExamples`**. Im unten gezeigten Mapping haben wir die Schaltfläche **Direkte Verbindungen** gedrückt und auf die Titelleiste der Komponente **`concat`** geklickt, die Schaltfläche **Quelle-Ziel-Verbindungen** aber noch nicht gedrückt. Wir sehen daher nun, dass nur die direkten Verbindungen der **`concat`**-Funktion zur Konstante **`"-"`** und den Komponenten `substring`, `position` und `Contacts` schwarz angezeigt werden. Die anderen Verbindungen im Mapping sind hellgrau.



Als nächstes werden wir nun auf die Schaltfläche **Quelle-Ziel-Verbindungen** drücken. In der Abbildung unten sehen Sie, was sich geändert hat:



Wenn die Schaltfläche **Quelle-Ziel-Verbindungen** aktiv ist, werden einige weitere Verbindungen schwarz angezeigt: (i) die Verbindungen der **`substring`**-Funktion zur Konstante **`1`** und der Komponente `PersonList` und (ii) die Verbindung der **`position`**-Funktion mit der Komponente `PersonList`. Die Verbindungen zwischen der Komponente `PersonList` und der davor liegenden Komponente bleiben hingegen hellgrau. Wenn Sie also die Schaltfläche **Quelle-Ziel-Verbindungen** aktivieren und auf eine Komponente klicken, können Sie die direkten Verbindungen der Komponente zurückverfolgen. Wenn die ausgewählte Komponente mit Transformationskomponenten[79] (z.B. Funktionen, Konstanten, Filtern, Sortierkomponenten, SQL-NoSQL-WHERE/ORDER-Komponenten, if-else-Bedingungen, Wertezuordnungen) verbunden ist, sehen Sie auch deren Verbindungen bis zu den Strukturkomponenten[78] (wie z.B. `PersonList` oben), Variablen, Join-Komponenten oder Webservice-Funktion, mit denen diese Transformationskomponenten verbunden sind.

In diesem Abschnitt

Dieser Abschnitt enthält eine Übersicht über Verbindungen und ist in die folgenden Kapitel gegliedert:

- Verbindungsarten[93]
- Verbindungseinstellungen[99]
- Kontextmenü für Verbindungen[101]
- Fehlerhafte Verbindungen[103]
- Beibehalten von Verbindungen nach Löschen von Komponenten[105]

© 2016-2022 Altova GmbH

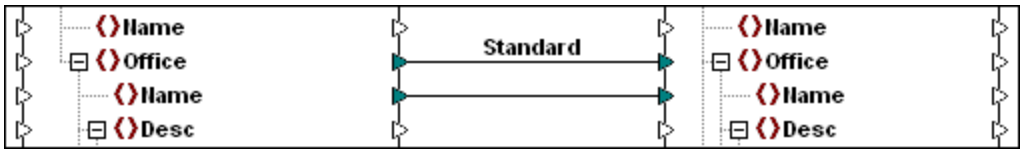
3.2.1 Verbindungsarten

In MapForce stehen die folgenden Verbindungsarten zur Verfügung:

- Zielorientierte Verbindungen 93 (Standard)
- Quellorientierte Verbindungen 93 (Mixed Content)
- Verbindungen mit identen Sub-Einträgen 96
- Alles kopieren-Verbindungen 97 (Sub-Einträge kopieren)

Zielorientierte im Vergleich zu quellorientierten Verbindungen

Zielorientierte und quellorientierte Verbindungen schließen einander gegenseitig aus. Welche dieser beiden Optionen Sie auswählen sollten, hängt von der Reihenfolge, in der Nodes gemappt werden müssen, ab. In zielorientierten Verbindungen wird die Reihenfolge der Nodes in der Ausgabe durch das *Ziel*-Schema vorgegeben. Diese Verbindungsart eignet sich für die meisten Mapping-Szenarien und ist der Standardverbindungstyp in MapForce. Zielorientierte Verbindungen werden in einem Mapping als durchgezogene Linien angezeigt (*siehe Abbildung unten*).



Wenn Sie XML-Nodes mit gemischtem Inhalt (mixed content, d.h. Child Nodes und Text) mappen möchten, sind zielorientierte Verbindungen manchmal nicht geeignet. In diesem Fall wird eine quellorientierte Verbindung 93 empfohlen. Dabei wird die Reihenfolge der Nodes in der Ausgabe durch das *Quell*-Schema vorgegeben.

Idente Sub-Einträge und "Alles kopieren"-Verbindungen

Idente Sub-Einträge und "Alles kopieren"-Verbindungen gehören zu einer Untergruppe von ziel- und quellorientierte Verbindungen. Damit werden Daten zwischen Nodes mit Child-Nodes, die in der Quell- und Zielkomponente ähnlich oder identisch sind, gemappt. "Alles kopieren"-Verbindungen sind Verbindungen mit identen Sub-Einträgen ähnlich, weisen aber anstelle mehrerer Verbindungen nur eine dicke Verbindung auf, wodurch das Mapping übersichtlicher wird.

In diesem Abschnitt finden Sie Informationen über die einzelnen Verbindungsarten und die Szenarien, in denen diese Verbindungsarten sich als nützlich erweisen.

3.2.1.1 Quellorientierte Verbindungen

Bei einer quellorientierten Verbindung, kann gemischter Inhalt (Mixed Content) (Text- und Child-Nodes) automatisch in derselben Reihenfolge, wie er in der XML-*Quelldatei* vorkommt, gemappt werden. Mixed Content-Verbindungen werden auf Ebene des Parent-Node als gepunktete Linien angezeigt (*siehe Mappen des <para>-Elements*). In diesem Kapitel wird erläutert, wie Sie Mixed Content mappen. Außerdem wird gezeigt, wie sich die Verwendung von Standardverbindungen (zielorientierten Verbindungen) mit Mixed Content auswirkt.

Anmerkung: Auch in Datenbankfeldern mit gemischtem Inhalt (Mixed Content) können quellorientierte Verbindungen verwendet werden (*Professional und Enterprise Edition*).

Anmerkung: Damit Mixed Content akzeptiert wird, müssen die Zielkomponenten Mixed Content Nodes haben.

Mappen von gemischtem Inhalt

In diesem Kapitel wird erläutert, wie Sie Mixed Content (gemischten Inhalt) mittels einer quellorientierten Verbindung mappen. Sie benötigen die folgenden Dateien: **Tut-OrgChart.xml**, **Tut-Orgchart.mfd, Tut-Person.xsd** und **Tut-OrgChart.xsd**, die sich im Ordner Tutorial[27] befinden.

XML-Quellinstanz

Unten sehen Sie einen Ausschnitt aus der Datei **Tut-OrgChart.xml**. In diesem Beispiel geht es um das Mixed Content-Element <para> mit seinen Child-Nodes <bold> und <italic>. Das Element para enthält auch eine Processing Instruction (<?sort alpha-ascending?>) sowie einen Kommentar (<!--Company details... -->), die, wie unten gezeigt, ebenfalls gemappt werden können. Beachten Sie die Reihenfolge der text-Nodes und der bold/italic Nodes in der XML-Instanzdatei.

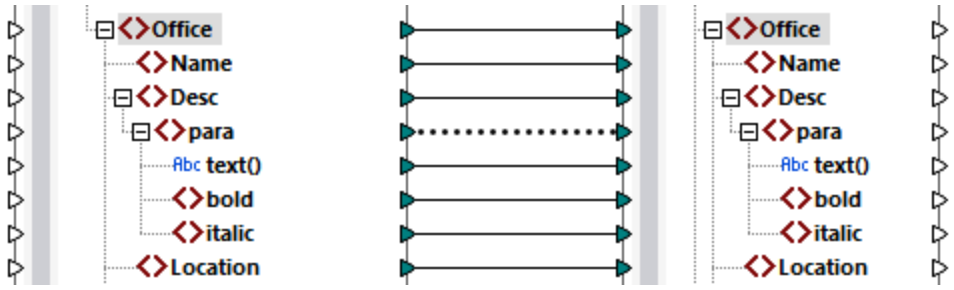
```
 8        <Desc>
 9          <para>The company was established in <bold>Vereno</bold> in 1995. Nanonull develops
    nanoelectronic technologies for <italic>multi-core processors.</italic> February 1999 saw the
    unveiling of the first prototype <bold>Nano-grid.</bold> The company hopes to expand its
    operations <italic>offshore</italic> to drive down operational costs.
10            <?sort alpha-ascending?>
11            <!--Company details: location and general company information.-->
12          </para>
13          <para>White papers and further information will be made available in the near future.
14          </para>
15        </Desc>
```

Mappen des <para>-Elements

In der Abbildung unten sehen Sie einen Ausschnitt aus **Tut-Orgchart.mfd**. Die gepunktete Linie im Beispiel unten zeigt, dass das Element <para> Mixed Content hat. Um manuell Mixed Content-Verbindungen zu erstellen, gehen Sie folgendermaßen vor:

1. Wählen Sie die Menüoption **Verbindung | Idente Sub-Einträge automatisch verbinden**, um idente Sub-Einträge[96] automatisch zu verbinden. Alternativ dazu können Sie den Node <para> mit seinen Sub-Einträgen manuell mappen.
2. Verbinden Sie das Datenelement <para> in der Quellkomponente mit dem Datenelement <para> in der Zielkomponente. Daraufhin wird eine Meldung angezeigt, in der Sie gefragt werden, ob die Verbindung als quellorientierte Verbindung definiert werden soll.
3. Klicken Sie auf **Ja**, um eine Mixed Content-Verbindung zu erstellen.
4. Klicken Sie auf **Ausgabe**, um das Ergebnis des Mappings zu sehen. Klicken Sie in der Symbolleiste im **Ausgabebereich** auf die Schaltfläche  (**Zeilenumbruch**), um das Codefragment zur Gänze (d.h. so dass es nicht über die Bildlaufleiste hinausragt) im **Ausgabebereich** zu sehen. Der gemischte Inhalt des <para>-Node wurde in derselben Reihenfolge, wie er in der XML-Quelldatei aufscheint, gemappt.

© 2016-2022 Altova GmbH



Processing Instructions und Kommentare
Um Processing Instructions und/oder Kommentare zu mappen, gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf die Mixed Content-Verbindung (gepunktete Linie) und wählen Sie **Eigenschaften**.
2. Aktivieren Sie unter **Quellorientiert (Mixed content)**, die Optionen **Processing Instructions mappen** und **Comments mappen**.

Zielorientierte Verbindungen mit Mixed Content
Wenn Sie für Mixed Content zielorientierte Verbindungen verwenden, erhalten Sie eventuell unerwünschte Ergebnisse. Um zu sehen, wie sich zielorientierte Verbindungen auf die Reihenfolge von Mixed Content Nodes auswirken, befolgen Sie die Anweisungen unten:

1. Öffnen Sie **`Tut-OrgChart.mfd`** aus dem Ordner **`Tutorial`**.
2. Aktivieren Sie die Symbolleisten-Schaltfläche  (Idente Sub-Einträge automatisch verbinden 96). Deaktivieren Sie in den Einstellungen für "Idente Sub-Einträge" 96 das Kontrollkästchen **Alles kopieren"-Verbindungen erstellen**. Dadurch wird verhindert, dass automatisch "Alles kopieren"-Verbindungen 97 erstellt werden.
3. Erstellen Sie eine Verbindung zwischen dem Node `para` in der Quellkomponente und dem Node `para` in der Zielkomponente. Daraufhin wird eine Meldung angezeigt, in der Sie gefragt werden, ob die Verbindung als quellorientierte Verbindung definiert werden soll. Klicken Sie auf **Nein**. Dadurch wird eine zielorientierte Verbindung erstellt.
4. Klicken Sie auf den **Ausgabe**-Bereich, um das Ergebnis des Mappings zu sehen (*siehe Abbildung unten*).

```
6        <Desc>
7            <para>The company was established in  in 1995. Nanonull develops nanoelectronic
technologies for  February 1999 saw the unveiling of the first prototype  The company hopes
to expand its operations  to drive down operational costs.
8                <bold>Vereno</bold><bold>Nano-grid.</bold><italic>multi-core processors.</
italic><italic>offshore</italic></para>
9            <para>White papers and further information will be made available in the near
future.
10           </para>
11       </Desc>
```

In der Abbildung oben sehen Sie, dass der Inhalt des Datenelements `text()` aus der Quellkomponente auf die Zielkomponente gemappt wurde. Die Reihenfolge der Child-Nodes (`bold` und `italic`) entspricht in der Ausgabe allerdings der im XML-Zielschema vorgegebenen Reihenfolge dieser Nodes, d.h. die Elemente `bold` und `italic` wurden nicht in den Text integriert, sondern separat gemappt.

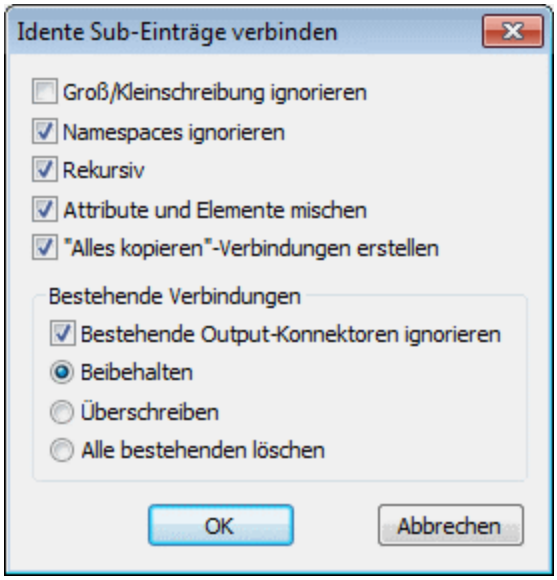
3.2.1.2 Verbindungen mit identen Sub-Einträgen

Bei Verbindungen mit identen Sub-Einträgen werden alle Sub-Einträge, die in der Quell- und Zieldatei denselben Namen haben, automatisch miteinander verbunden. Um diese Option zu aktivieren, wählen Sie eine der folgenden Methoden:

- Aktivieren Sie die Symbolleisten-Schaltfläche (**Idente Sub-Einträge automatisch verbinden**).
- Klicken Sie im Menü **Verbindung** auf den Befehl **Idente Sub-Einträge automatisch verbinden**.

Einstellungen für Verbindungen mit identen Sub-Einträgen

Um die Einstellungen für die Verbindung identischer Sub-Einträge zu ändern, klicken Sie mit der rechten Maustaste auf eine beliebige Verbindung und wählen Sie im Kontextmenü die Option **Idente Sub-Einträge verbinden** (*siehe Abbildung unten*). Klicken Sie alternativ dazu im Menü **Verbindung** auf den Befehl **Einstellungen für 'Idente Sub-Einträge verbinden'**.



In der nachstehenden Liste finden Sie eine Beschreibung der Optionen im Dialogfeld **Einstellungen für 'Idente Sub-Einträge verbinden'**. Die Einstellungen in diesem Dialogfeld werden nur wirksam, wenn die Symbolleisten-Schaltfläche (**Aktiviert/Deaktiviert die automatische Verbindung von Sub-Einträgen**) aktiv ist.

Allgemeine Einstellungen

- *Groß/Kleinschreibung ignorieren:* Idente Sub-Einträge werden unabhängig von der Groß- und Kleinschreibung der Child-Node-Namen verbunden.
- *Namespaces ignorieren:* Idente Sub-Einträge werden unabhängig von den Namespaces von Child-Nodes verbunden.
- *Rekursiv:* Erstellt zwischen identen Nodes neue Verbindungen, wenn Sie denselben Namen haben. Es spielt keine Rolle, wie tief die Nodes in der Struktur verschachtelt sind.
- *Attribute und Elemente mischen:* Damit können Verbindungen zwischen Attributen und Elementen desselben Namens erzeugt werden. So wird z.B. zwischen zwei Nodes mit dem Namen `Name` eine

© 2016-2022 Altova GmbH

Verbindung erstellt, selbst wenn es sich bei einem der Nodes um ein Element und beim anderen um ein Attribut handelt.
- *"Alles kopieren"-Verbindungen erstellen:* Diese Einstellung ist standardmäßigt aktiv. Damit wird (wenn möglich) eine "Alles kopieren"-Verbindung[97] erstellt, die Daten zwischen Nodes mit Child-Nodes, die einander ähnlich oder miteinander identisch sind, mappt.

Bestehende Verbindungen

- *Bestehende Output-Verbindungen ignorieren:* Mit dieser Option werden zusätzliche Verbindungen für alle identischen Nodes erstellt, selbst wenn diese bereits ausgehende Verbindungen aufweisen.
- *Beibehalten:* Mit dieser Option werden bestehende Verbindungen beibehalten.
- *Überschreiben:* Mit dieser Option werden bestehende Verbindungen überschrieben.
- *Alle bestehenden löschen:* Mit dieser Option werden alle bestehenden Verbindungen gelöscht, bevor neue erzeugt werden.

Löschen von Verbindungen als Gruppe

Um Verbindungen als Gruppe zu löschen, gehen Sie vor, wie unten beschrieben:

1. Klicken Sie mit der rechten Maustaste auf einen Node-Namen in der Komponente.
2. Wählen Sie im Kontextmenü den Befehl **Verbindungen löschen | Alle <...> Verbindungen löschen** (*siehe Abbildung unten*).



- *Alle direkten Verbindungen löschen:* Mit dieser Option löschen Sie alle Verbindungen, die direkt auf den oder vom ausgewählten Node aus gemappt sind.
- *Alle eingehenden Child-Verbindungen löschen:* Diese Option ist nur aktiv, wenn Sie mit der rechten Maustaste auf einen Parent Node in einer Zielkomponente geklickt haben. Mit dieser Option löschen Sie alle eingehenden Child-Verbindungen des ausgewählten Parent Node.
- *Alle hinausgehenden Child-Verbindungen löschen:* Diese Option ist nur aktiv, wenn Sie mit der rechten Maustaste auf einen Parent Node in einer Quellkomponente geklickt haben. Mit dieser Option löschen Sie alle ausgehenden Child-Verbindungen des ausgewählten Parent Node.

3.2.1.3 "Alles kopieren"-Verbindungen

Mit "Alles kopieren"-Verbindungen werden Daten zwischen Nodes mit Child-Nodes, die einander ähnlich oder miteinander identisch sind, gemappt. Alles kopieren"-Verbindungen sind nur für dasselbe Format (z.B. JSON auf JSON oder XML auf XML) möglich. Dies gilt auch für alle Textkomponenten: Flat Flat Files, FlexText- und EDI-Dateien. Da es sich bei allen diesen Formaten um Textdateien handelt, können Sie jedes davon miteinander kombinieren und z.B. zwischen EDI- und FlexText-Dateien eine "Alles kopieren"-Verbindung erstellen. Nähere Informationen zu den Formaten, die als Datenquell- und -zielkomponenten verwendet werden können, finden Sie unter Mapping: Quellen und Ziele[27].

Der Hauptvorteil von "Alles kopieren"-Verbindungen ist, dass der Mapping-Arbeitsbereich dadurch übersichtlicher wird: Anstelle mehrerer Verbindungen wird eine einzige "dicke" Verbindungslinie gezogen (*siehe*

Beispiel unter Manuelles Erstellen von "Alles kopieren"-Verbindungen). In den folgenden Unterabschnitten wird erläutert, wie Sie "Alles kopieren"-Verbindungen automatisch und manuell erstellen.

Erstellen einer "Alles kopieren"-Verbindung

Um eine "Alles kopieren"-Verbindung automatisch zu erstellen, gehen Sie folgendermaßen vor:

1. Gehen Sie zum Menü **Verbindung**.
2. Klicken Sie auf **Einstellungen für 'Idente Sub-Einträge verbinden'**.
3. Aktivieren Sie das Kontrollkästchen **"Alles kopieren"-Verbindungen erstellen** und klicken Sie auf **OK**.
4. Klicken Sie auf die Symbolleisten-Schaltfläche **Aktiviert/Deaktiviert die automatische Verbindung von Sub-Einträgen**. Klicken Sie alternativ dazu im Menü **Verbindung** auf den Befehl **Idente Sub-Einträge automatisch verbinden**.

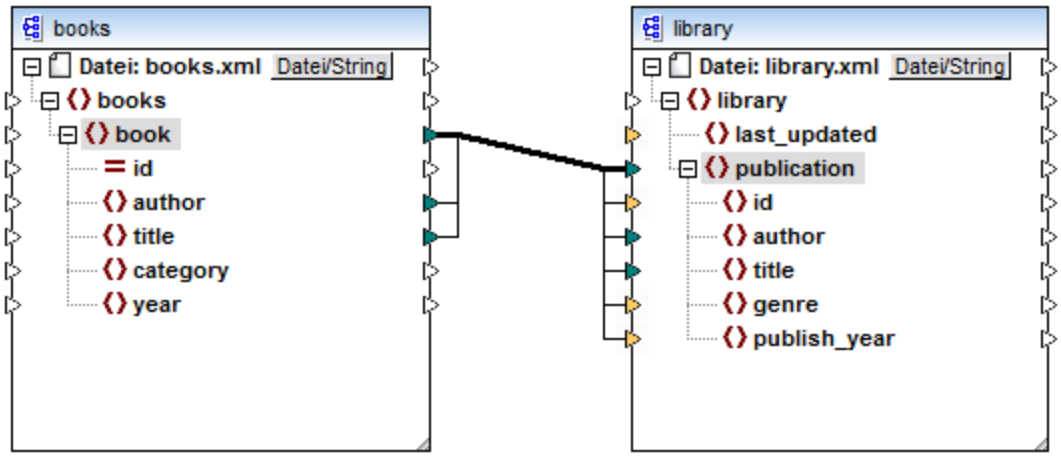
Wenn der Typ und/oder Name von Child-Nodes in der Quell- und der Zielkomponente nicht identisch ist, wird nicht automatisch eine "Alles kopieren"-Verbindung erstellt und Sie müssen diese manuell herstellen.

Manuelles Erstellen von "Alles kopieren"-Verbindungen

Um manuell eine "Alles kopieren"-Verbindung zu erstellen, gehen Sie folgendermaßen vor:

1. Fügen Sie eine Quelldatei hinzu: Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur Datei **`Books.xml`** im Ordner BasicTutorials 27.
2. Fügen Sie eine Zieldatei hinzu: Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur Datei **`Library.xsd`** im selben Ordner wie **`Books.xml`**. Klicken Sie auf **Überspringen**, wenn Sie von MapForce aufgefordert werden, eine XML-Beispieldatei hinzuzufügen.
3. Mappen Sie den Node <book> der Komponente **`Books`** auf den Node <publication> der Komponente **`Library`**. Da die Struktur der Elemente <book> und <publication> nicht vollständig identisch ist, wird keine "Alles kopieren"-Verbindung erstellt. Statt dessen werden mit Hilfe der Funktion **Idente Sub-Einträge automatisch verbinden** automatisch alle Child-Nodes desselben Namens miteinander verbunden, wie im Tutorial 1 49 beschrieben.
4. Um die automatische Verbindung in eine "Alles kopieren-Verbindung zu ändern, klicken Sie mit der rechten Maustaste auf die Verbindung zwischen <book> und <publication> und wählen Sie im Kontextmenü den Befehl **Alles kopieren (Sub-Einträge kopieren)**.
5. Daraufhin erscheint ein Fenster, in dem vorgeschlagen wird, die vorhandenen Verbindungen durch eine "Alles kopieren"-Verbindung zu ersetzen. Klicken Sie auf **OK.** Die Quell- und Zielkomponente haben nun eine "Alles kopieren"-Verbindung (*siehe Abbildung unten*).

© 2016-2022 Altova GmbH



Im obigen Mapping sind nur zwei Child-Nodes in den beiden Strukturen miteinander identisch: `<author>` und `<title>`. Daher bestehen keine "Alles kopieren"-Verbindungen zwischen diesen Nodes. Child-Nodes, die nicht identisch sind, können nicht miteinander verbunden werden. In der Abbildung sehen Sie, dass `id` nicht in die "Alles kopieren"-Verbindung miteinbezogen wird, da der Typ in der Quell- und Zielkomponente nicht derselbe ist: `id` ist in der Quellkomponente ein Attribut und in der Zielkomponente ein Element. Wenn Sie versuchen, zwischen nicht identischen Nodes wie z.B. zwischen `<category>` und `<genre>` eine Verbindung zu erstellen, fragt MapForce Sie, ob diese Verbindung ersetzt oder der Input dupliziert werden soll.

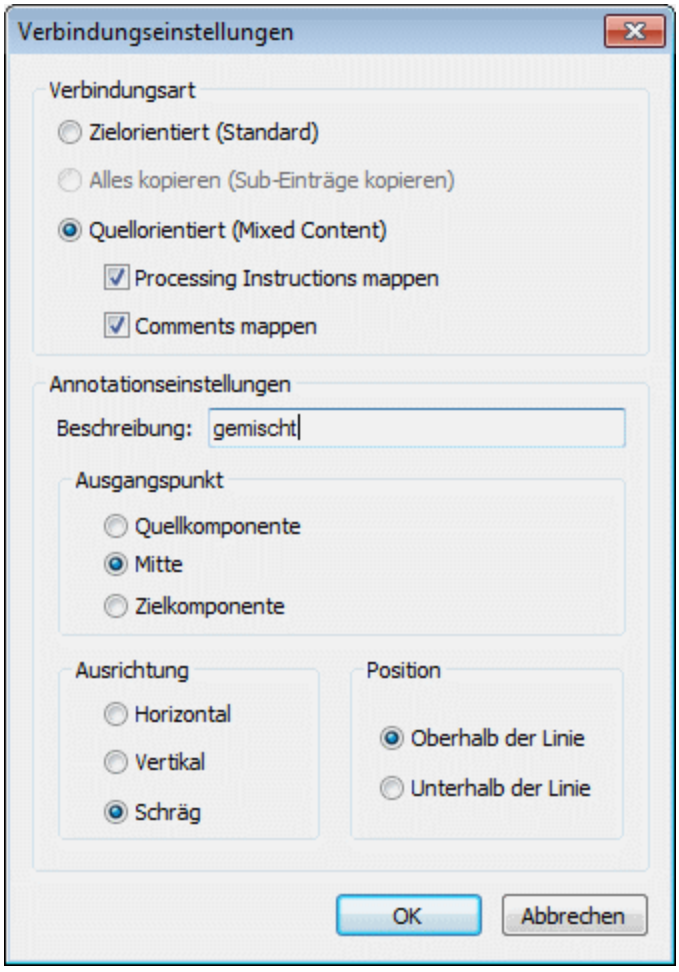
Eine Duplizierung des Input 84 ist nur dann sinnvoll, wenn die Zielkomponente Daten aus mehr als einem Input erhalten soll, was hier nicht erforderlich ist. Wenn Sie die "Alles kopieren"-Verbindung ersetzen, werden Sie in einem Meldungsfeld noch einmal aufgefordert, die "Alles kopieren"-Verbindung aufzulösen oder zu löschen. Klicken Sie auf **"Alles kopieren"-Verbindung auflösen**, wenn Sie die "Alles kopieren"-Verbindung durch einzelne zielorientierte Verbindungen 93 ersetzen möchten. Wenn Sie die "Alles kopieren"-Verbindung lieber vollständig entfernen möchten, klicken Sie auf **Child-Verbindungen löschen**.

Achtung:

Bei der Erstellung von "Alles kopieren"-Verbindungen zwischen einem Schema und einem Parameter einer benutzerdefinierten Funktion 967 muss den beiden Komponenten dasselbe Schema zugrunde liegen. Die beiden Komponenten müssen dabei aber nicht dasselbe Root-Element haben.

3.2.2 Verbindungseinstellungen

Im Dialogfeld **Verbindungseinstellungen** können Sie die Einstellungen einer Verbindung definieren. Um dieses Dialogfeld zu öffnen, doppelklicken Sie auf die Verbindung. Klicken Sie alternativ dazu mit der rechten Maustaste auf die Verbindung und wählen Sie im Kontextmenü den Befehl **Eigenschaften**. Die Einstellungen sind in zwei Bereiche unterteilt: "Verbindungsart" und "Annotationseinstellungen". Nähere Informationen dazu finden Sie in den Unterabschnitten weiter unten.



Verbindungsarten

Sie können eine der unten beschriebenen Verbindungsarten auswählen:

- Zielorientierte (Standard) 93-Verbindungen eignen sich für die meisten Mapping-Szenarien.
- Alles kopieren (Sub-Einträge kopieren) 97-Verbindungen: Wenn eine Quell- und Zielkomponente identische oder ähnliche Nodes mit identischen Child-Nodes hat, wird zwischen diesen identischen Nodes automatisch eine "Alles kopieren"-Verbindung erstellt.
- Bei einer quellorientierten (Mixed Content) 93-Verbindung, kann gemischter Inhalt (Mixed Content) (Text- und Child-Nodes) in derselben Reihenfolge, wie er in der XML-Quelldatei vorkommt, gemappt werden. Wenn Sie die Kontrollkästchen **Processing Instructions mappen** und/oder **Comments mappen** aktivieren, werden diese Datengruppen in die Ausgabedatei inkludiert (*siehe Abbildung unten*).

© 2016-2022 Altova GmbH



Annotationseinstellungen

Über den Abschnitt **Annotationseinstellungen** können Sie eine Verbindung beschriften. Diese Option steht für alle Verbindungsarten zur Verfügung. Um eine Verbindung zu beschriften, gehen Sie folgendermaßen vor:

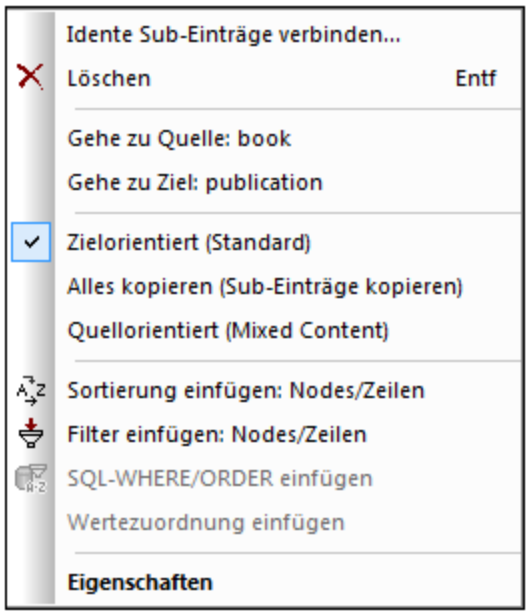
1. Klicken Sie mit der rechten Maustaste auf die Verbindung und wählen Sie im Kontextmenü den Befehl **Eigenschaften**. Doppelklicken Sie alternativ dazu auf die Verbindung.
2. Geben Sie den Namen der ausgewählten Verbindung in das Feld **Beschreibung** ein. Dadurch werden alle Optionen im Bereich **Annotationseinstellungen** aktiv.
2. In den restlichen Bereichen können Sie den **Ausgangspunkt**, die **Ausrichtung** und **Position** der Beschriftung definieren.
3. Aktivieren Sie die Symbolleisten-Schaltflächen  (**Annotationen anzeigen**). Wenn die Schaltfläche in der Symbolleiste noch nicht zu sehen ist, aktivieren Sie in der Symbolleiste die Schaltfläche **Optionen anzeigen**.



Anmerkung: Wenn die Schaltfläche **Annotationen anzeigen** deaktiviert ist, können Sie den Annotationstext dennoch sehen, wenn Sie den Mauszeiger über die Verbindung platzieren. Die Annotation wird als Tooltip angezeigt, wenn die Schaltfläche  (**Tipps anzeigen**) in der Symbolleiste **Anzeigeoptionen** aktiv ist.

3.2.3 Kontextmenü für Verbindungen

In diesem Kapitel werden Befehle aus dem Kontextmenü für Verbindungen beschrieben. Wenn Sie mit der rechen Maustaste auf eine Verbindung klicken, stehen die folgenden Kontextmenübefehle zur Verfügung:



Nähere Informationen dazu finden Sie in den Unterabschnitten weiter unten.

Allgemeine Einstellungen

- *Idente Sub-Einträge verbinden:* Öffnet das Dialogfeld Idente Sub-Einträge verbinden[96]. Dieser Befehl ist aktiv, wenn die Verbindung idente Sub-Einträge haben kann.
- *Löschen:* Löscht die markierte Verbindung.
- *Gehe zu Quelle: <Datenelementname>* Markiert den Output[77]-Konnektor der ausgewählten Verbindung.
- *Gehe zu Ziel: <Datenelementname>* Markiert den Input[77]-Konnektor der ausgewählten Verbindung.

Verbindungsarten

Nähere Informationen zu den Verbindungsarten finden Sie unter Verbindungsarten[93] und Verbindungseinstellungen[99].

Einfügebefehle

- *Sortierung einfügen: Nodes/Zeilen:* Fügt zwischen einem Quell- und Ziel-Node eine Sortierkomponente[908] ein.
- *Filter einfügen: Nodes/Zeilen:* Fügt zwischen einem Quell- und Ziel-Node eine Filter[914]-Komponente ein.
- *SQL/NoSQL-WHERE/ORDER einfügen:* Fügt zwischen einem Quell- und Ziel-Node eine SQL/NoSQL-WHERE/ORDER-Komponente ein (*Professional und Enterprise Edition*). Nähere Informationen dazu finden Sie unter Filtern und Sortieren von Datenbankdaten[361].
- *Wertezuordnung einfügen:* Fügt zwischen einem Quell- und Ziel-Node eine Wertezuordnung[921] ein.

Eigenschaften

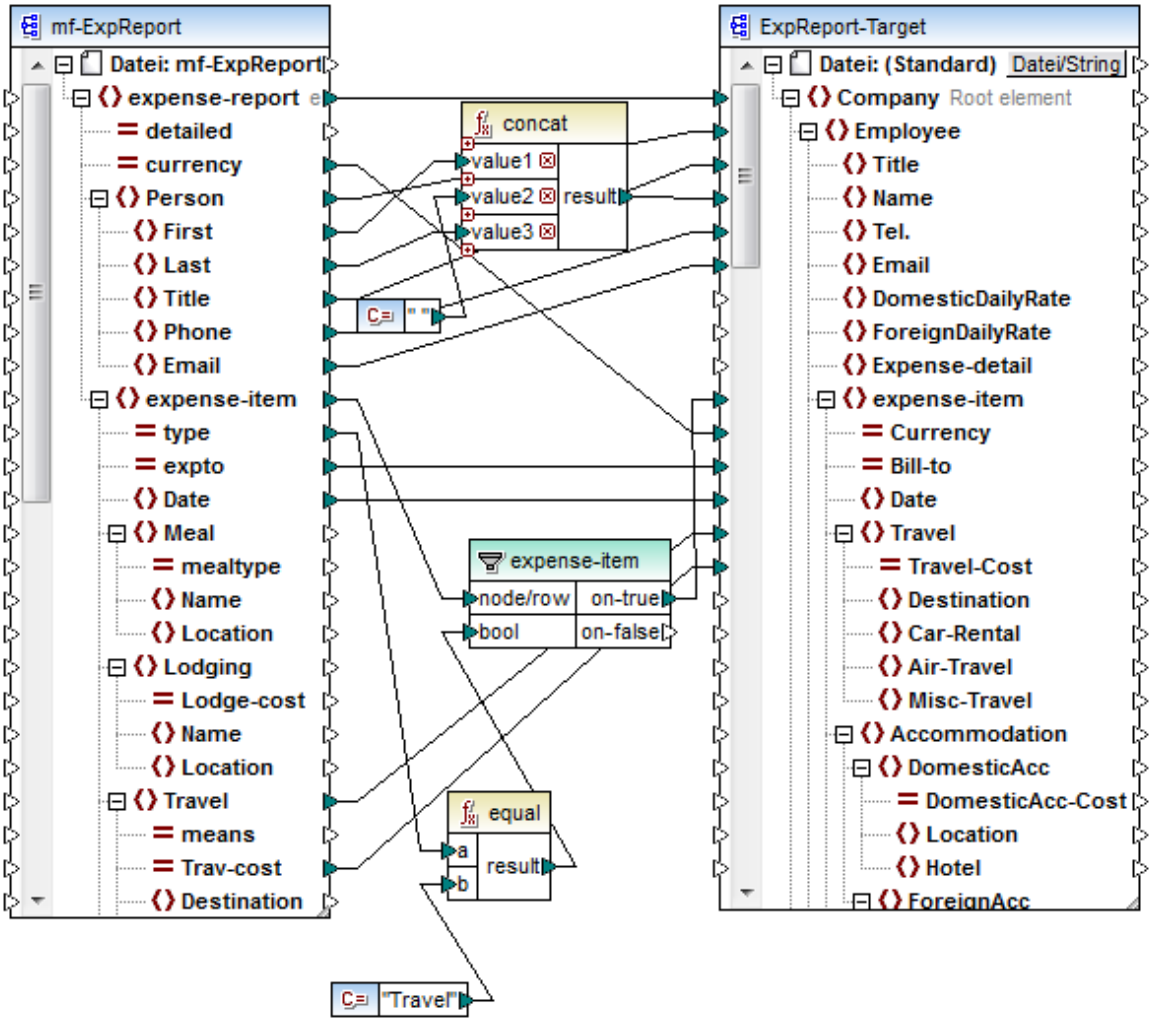
Öffnet das Dialogfeld Verbindungseinstellungen[99].

© 2016-2022 Altova GmbH

3.2.4 Fehlerhafte Verbindungen

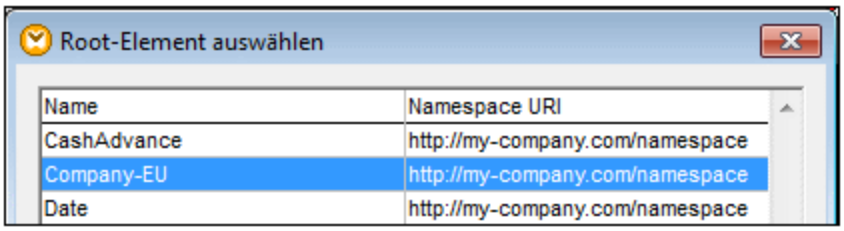
Manchmal muss ein Schema in einer Quell- oder Zielkomponente geändert werden. Änderungen an einem Schema können sich auf die Gültigkeit Ihres Mappings auswirken und zu einer Reihe fehlerhafter Verbindungen führen. In diesem Kapitel wird erläutert, wie Sie solche Verbindungen reparieren, nachdem Sie die Schema-Datei geändert haben. Befolgen Sie die Anleitung im Beispiel unten, um zu sehen, wie Sie fehlerhafte Verbindungen reparieren können.

1. Öffnen Sie **`Tut-ExpReport.mfd`** aus dem Tutorial-Ordner[27]. Unterhalb sehen Sie einen Ausschnitt aus diesem Mapping.

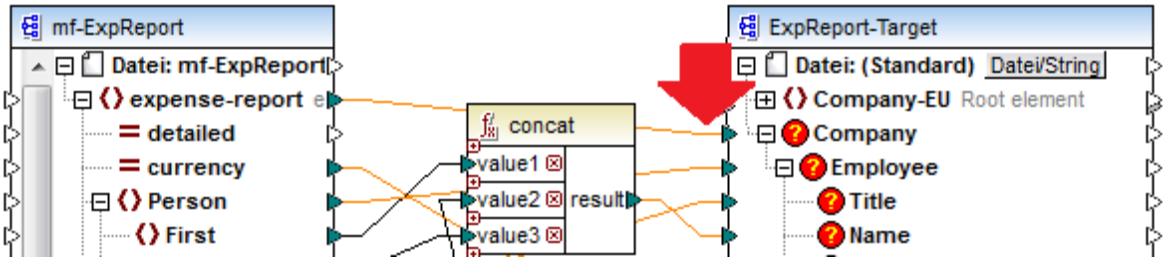


2. Öffnen Sie **`ExpReport-Target.xsd`** in einem Editor (z.B., Altova XMLSpy) und ändern Sie das Root-Element `Company` im Ziel-Schema in `Company-EU`. Sie müssen MapForce dazu nicht schließen.
3. Nachdem Sie das Root-Element des Ziel-Schemas bearbeitet haben**,** wird in MapForce eine **Meldung über geänderte Dateien** angezeigt. Klicken Sie auf die Schaltfläche **Neu laden**. Da das Root-Element geändert wurde, werden in der Komponente mehrere fehlende Nodes angezeigt.

4. Klicken Sie oben in der Komponente auf **Neues Root-Element auswählen** (*siehe Abbildung unten*). Sie können das Root-Element auch ändern, indem Sie mit der rechten Maustause auf die Titelleiste der Komponente klicken und im Kontextmenü den Befehl **Root-Element ändern** auswählen.



5. Wählen Sie als neues Root-Element `Company-EU` aus und klicken Sie auf **OK**. Das Root-Element `Company-EU` wird nun auf der obersten Ebene der Komponente angezeigt.
6. Sie müssen die Verbindung nun vom fehlenden Node `Company` auf das neue Root-Element verschieben. Klicken Sie auf den dicken Abschnitt der Verbindung von `Company` (*siehe roter Pfeil unten*). Ziehen Sie anschließend die Verbindung auf das Root-Element `Company-EU`.



Daraufhin wird ein Dialogfeld mit der Frage angezeigt, ob Sie alle übereinstimmenden verbundenen Child-Nodes verschieben möchten. Sie können wählen, ob Sie nur die ausgewählte Verbindung oder die ausgewählte Verbindung mit ihren Child-Nodes, die mit den Child-Nodes im neuen Root-Element übereinstimmen, verschoben werden soll. In unserem Beispiel haben wir die Option **Child-Verbindungen inkludieren** gewählt. Sobald Sie auf diese Schaltfläche klicken, verschwinden alle fehlerhaften Nodes aus der Komponente.

Anmerkung: Wenn der Node, auf den gemappt werden soll, zwar denselben Namen wie der Quell-Node, aber einen anderen Namespace hat, wird im Benachrichtigungsdialogfeld eine zusätzliche Schaltfläche **Child-Verbindungen inkludieren und Namespace mappen** angezeigt. Wenn Sie auf diese Schaltfläche klicken, werden Child-Verbindungen desselben Namespace wie die des Parent-Node der Quellkomponente auf dieselben Child-Nodes unter dem Node des anderen Namespace verschoben.

Alternative Lösung

Eine andere Lösung für das oben beschriebene Problem wäre, die in Ihrem Mapping nicht mehr benötigten fehlerhaften Nodes zu löschen. Wenn Sie z.B. die Verbindung zwischen der **`concat`**-Funktion und `Name` löschen, verschwindet der Node `Name` aus der `ExpReport-Target`-Komponente.

Fehlerhafte Verbindungen in Datenbanken (Professional und Enterprise Edition)

Wenn Ihre Datenbankkomponente fehlerhafte Verbindungen aufweist, müssen Sie die Komponenteneinstellungen ändern 82. Wenn Sie im Dialogfeld **Komponenteneinstellungen** auf die Schaltfläche **Ändern** klicken, können Sie eine andere Datenbank auswählen oder Tabellen in Ihrer Datenbankkomponente ändern. Alle gültigen/korrekten Verbindungen und relevanten Datenbankdaten bleiben erhalten, wenn Sie eine Datenbank mit derselben Struktur auswählen.

 © 2016-2022 Altova GmbH

3.2.5 Beibehalten von Verbindungen nach Löschen von Komponenten

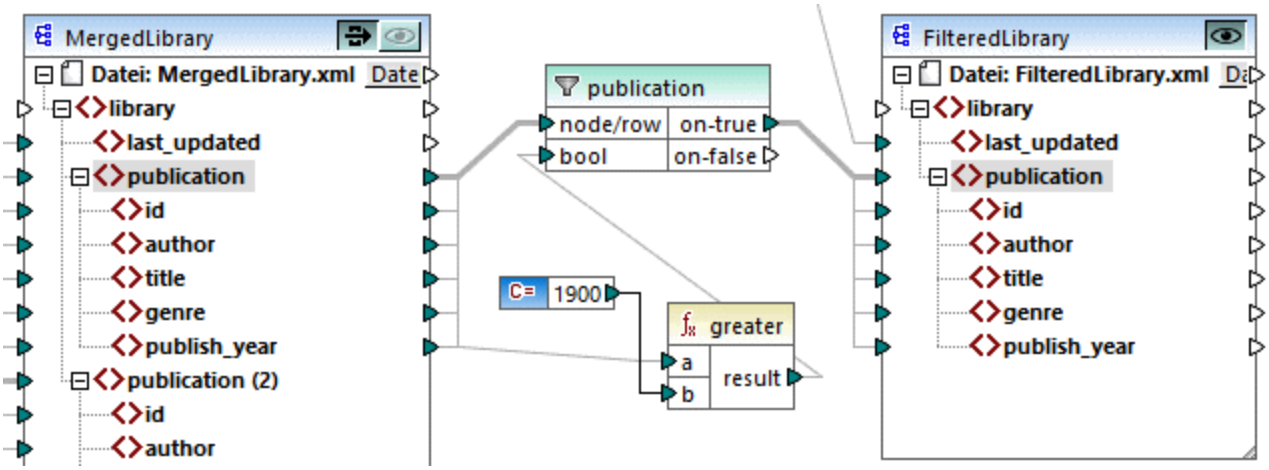
Sie können in MapForce Verbindungen auch nach Löschen einiger Transformationskomponenten[78] (z.B. Variablen, Sortier- und Filter-Komponenten, Wertezuordnungen, einfache Input-Komponenten, SQL/NoSQL-WHERE/ORDER-Komponenten) beibehalten. Bei Verbindungen kann es sich um Einfach- und Mehrfachverbindungen handeln. Vor allem die Beibehaltung von Verbindungen mit mehreren Child-Verbindungen kann sich als nützlich erweisen, da Sie dadurch nach Löschung einer Transformationskomponente nicht jede einzelne Child-Verbindung manuell wiederherstellen müssen. Um diese Option zu aktivieren, wählen Sie **Extras | Optionen | Bearbeiten** und aktivieren Sie **Intelligente Komponentenlöschung (nützliche Verbindungen beibehalten)**. Diese Option ist standardmäßig deaktiviert, sodass bei Löschung einer Transformationskomponente auch die direkten Verbindungen dazu gelöscht werden.

Beispiel

Zur Veranschaulichung einer intelligenten Komponentenlöschung werfen Sie einen Blick in die Beispieldatei **`Tut3-ChainedMapping`**. Die Beispieldatei steht im Ordner BasicTutorials[27] zur Verfügung.

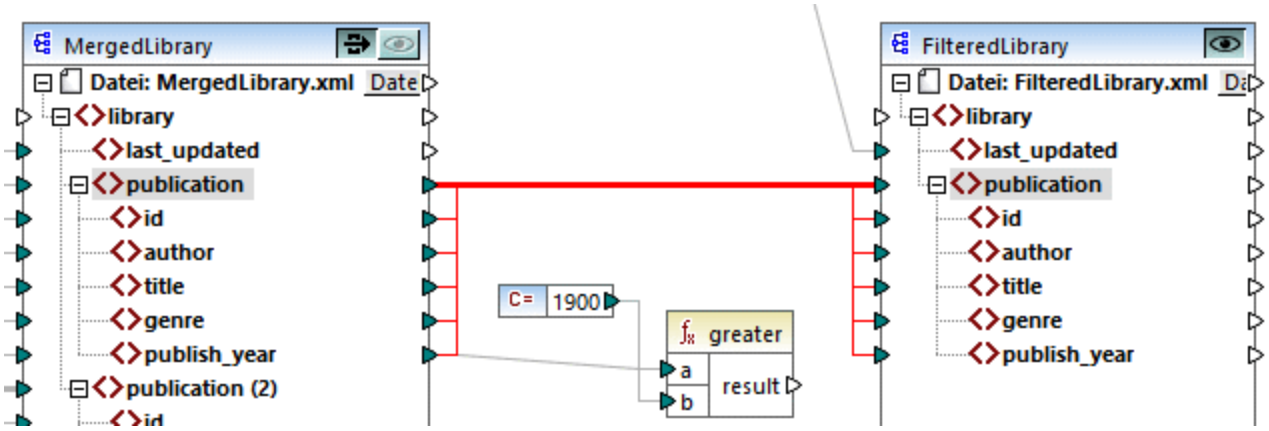
Vor der Löschung

In der Abbildung unten sehen Sie, dass zwischen der Komponente `MergedLibrary` und dem Filter `publication` und zwischen dem Filter `publication` und der Komponente `FilteredLibrary` Alles kopieren-Verbindungen[97] bestehen. Der Filter `publication` soll nun gelöscht werden, aber die "Alles kopieren"-Verbindungen sollen bestehen bleiben. Aktivieren Sie nun im Dialogfeld **Optionen** das Kontrollkästchen **Intelligente Komponentenlöschung** (*siehe oben*).



Nach der Löschung

Nachdem die `publication`-Funktion gelöscht wurde, wird die "Alles kopieren"-Verbindung direkt zwischen dem Node `publication` in `MergedLibrary` und dem Node `publication` in `FilteredLibrary` erstellt (*siehe Abbildung unten*).



Anmerkung: Wenn bei einer Filter-Komponente sowohl der `on-true` als auch der `on-false`-Output verbunden ist, bleiben die Verbindungen beider Outputs erhalten.

© 2016-2022 Altova GmbH

3.3 Allgemeine Verfahren und Funktionalitäten

Neben der Erstellung von Mappings stehen auch Funktionen zum Validieren Ihres Mappings und Ihrer Ausgabe, zum Generieren von Code, zur Verwendung der Textansicht und zum Definieren von Mapping-Einstellungen zur Verfügung. Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

- Validierung 107
- Codegenerierung 109
- Funktionalitäten der Textansicht 110
- Suchen in der Textansicht 113
- Mapping-Einstellungen 116

3.3.1 Validierung

In diesem Kapitel wird erläutert, wie Sie Mappings validieren. Außerdem erfahren Sie hier, wie Sie eine Vorschau auf die Ausgabe anzeigen und diese speichern und validieren.

Validieren von Mappings

MapForce validiert Mappings automatisch, wenn Sie auf das Register **Ausgabe** klicken. Sie können Ihr Mapping auch manuell validieren. Auf diese Art können Sie potenzielle Fehler und Warnungen ausfindig machen und korrigieren, bevor das Mapping ausgeführt wird. Um ein Mapping manuell zu validieren, klicken Sie in den **Mapping**-Bereich und wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Datei** auf **Mapping validieren**.
- Klicken Sie in der Symbolleiste auf  (**Validieren**).

Bei der Validierung eines Mappings überprüft MapForce das Mapping auf nicht unterstützte Komponentenarten, falsche oder fehlende Verbindungen. Nähere Informationen zum Validierungsstatus finden Sie unter Fenster "Meldungen" 38. Außerdem können Sie im Fenster **Meldungen** Aktionen im Zusammenhang mit Meldungen 39 durchführen. Um das Ergebnis jeder einzelnen Validierung auf einem eigenen Register anzuzeigen, klicken Sie auf der linken Seite des **Meldungsfensters** auf die nummerierten Register. Dies ist z.B. hilfreich, wenn Sie gleichzeitig mit mehreren Mappings arbeiten.

Validierung von Transformationskomponenten

Transformationskomponenten 79 werden folgendermaßen validiert:

- Wenn ein zwingend erforderlicher **Input-Konnektor** nicht verbunden ist, wird eine Fehlermeldung generiert und die Transformation wird gestoppt.
- Wenn ein **Output-Konnektor** nicht verbunden ist, wird eine Warnmeldung generiert und die Transformation wird fortgesetzt. Die Komponente, die die Warnung verursacht hat, und ihre Daten werden ignoriert und nicht auf die Zielkomponente gemappt.

Anzeige einer Vorschau und Validieren der Ausgabe

Sie können in MapForce eine Vorschau der Ausgabe anzeigen, ohne den generierten Code mit einem externen Prozessor oder Compiler ausführen und kompilieren zu müssen. Bevor Sie den generierten Code extern

verarbeiten, sollten Sie eine Vorschau der Transformationsausgabe in MapForce anzeigen. Bei einer Vorschau auf das Mapping-Ergebnis führt MapForce das Mapping aus und zeigt die Ausgabe im Ausgabefenster[41] an.

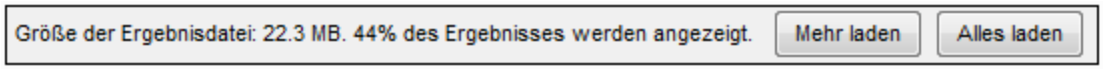
Sobald die Daten im **Ausgabefenster** zur Verfügung stehen, können Sie diese validieren und gegebenenfalls speichern. Außerdem können Sie mit dem Befehl **Suchen** (**Strg + F)** schnell nach einem bestimmten Textmuster in der Ausgabedatei suchen. Nähere Informationen dazu finden Sie unter Suchen in der Textansicht[113]. Alle Fehler, Warnungen oder Informationsmeldungen im Zusammenhang mit der Mapping-Ausführung werden im Fenster "Meldungen"[38] angezeigt.

Bei Auswahl von C++, C# oder Java (*Professional und Enterprise Edition*) als Transformationssprache[28], führt MapForce das Mapping mit dem Built-in-Transformationsprozessor aus und zeigt das Ergebnis im **Ausgabefenster** an.

Um die Transformationsausgabe zu speichern, klicken Sie in den **Mapping**-Bereich und wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Ausgabe** auf **Ausgabedatei speichern**.
- Klicken Sie in der Symbolleiste auf  **(Generierte Ausgabe speichern)**.

Optionen zum Laden
Bei großen Ausgabedateien beschränkt MapForce die Menge der im **Ausgabefenster** angezeigten Daten, d.h. es wird nur ein Teil der Daten im Ausgabefenster angezeigt. In diesem Fall erscheint im unteren Bereich eine Schaltfläche **Mehr laden** (*siehe Abbildung unten*). Wenn Sie darauf klicken, wird der nächste Teil angezeigt. Die Vorschaueinstellungen können im Dialogfeld **Optionen** auf dem Register **Allgemein** konfiguriert werden. Nähere Informationen dazu finden Sie unter MapForce Optionen[1539].

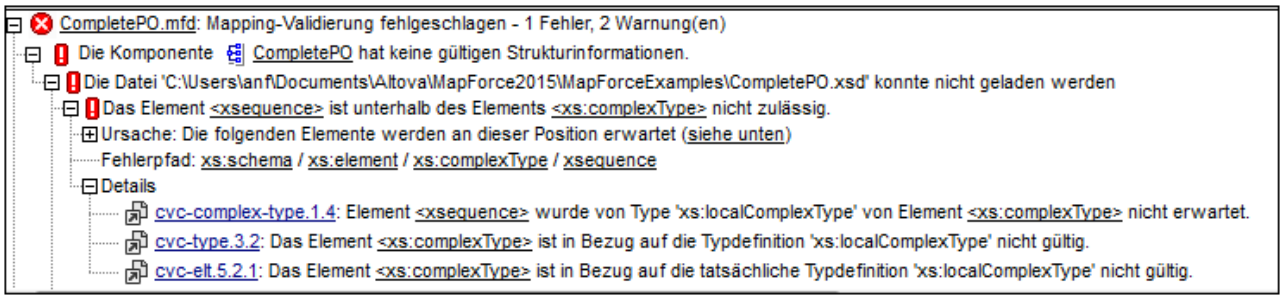


Validieren der Ausgabe
Sobald die Ausgabe im Fenster **Ausgabe** zur Verfügung steht, können Sie diese anhand des damit verknüpften Schemas validieren. Anmerkung: Die Schaltfläche **Ausgabe validieren** und der entsprechende Menübefehl (**Ausgabe | Ausgabedatei validieren**) sind nur aktiv, wenn die Ausgabedatei die Validierung anhand eines Schemas unterstützt. Das Ergebnis der Validierung wird im Fenster **Meldungen** angezeigt. Wählen Sie eine der folgenden Methoden, um die Ausgabe zu validieren:

- Öffnen Sie den **Ausgabebereich** und klicken Sie in der Symbolleiste auf  (**Ausgabe validieren**).
- Öffnen Sie den **Ausgabebereich** und klicken Sie im Menü **Ausgabe** auf **Ausgabedatei validieren**.

In der Abbildung unten sehen Sie eine nicht erfolgreiche Validierung. Das Fenster **Meldungen** enthält ausführliche Informationen über die Fehler. Wenn Sie z.B. auf den `<Name>`-Link klicken, wird dieses Element im **Ausgabebereich** markiert.

© 2016-2022 Altova GmbH



3.3.2 Codegenerierung

Je nachdem, welche Sprache Sie als Datentransformationssprache[28] ausgewählt haben, können Sie anhand eines Mappings Code generieren. Code kann in den folgenden Sprachen generiert werden:

- XSLT 1.0/XSLT 2.0/XSLT 3.0
- XQuery (*Professional und Enterprise Edition*)
- Java (*Professional und Enterprise Edition*)
- C# (*Professional und Enterprise Edition*)
- C++ (*Professional und Enterprise Edition*)

Beispiel
Um XSLT-Code zu generieren, gehen Sie folgendermaßen vor:

1. Wählen Sie die Menüoption **Datei | Code generieren in | XSLT**.
2. Geben Sie den Ordner an, in dem die generierte XSLT-Datei gespeichert werden soll und bestätigen Sie dies mit **OK**. MapForce generiert den Code und zeigt das Ergebnis der Operation im Fenster Meldungen[38] an.

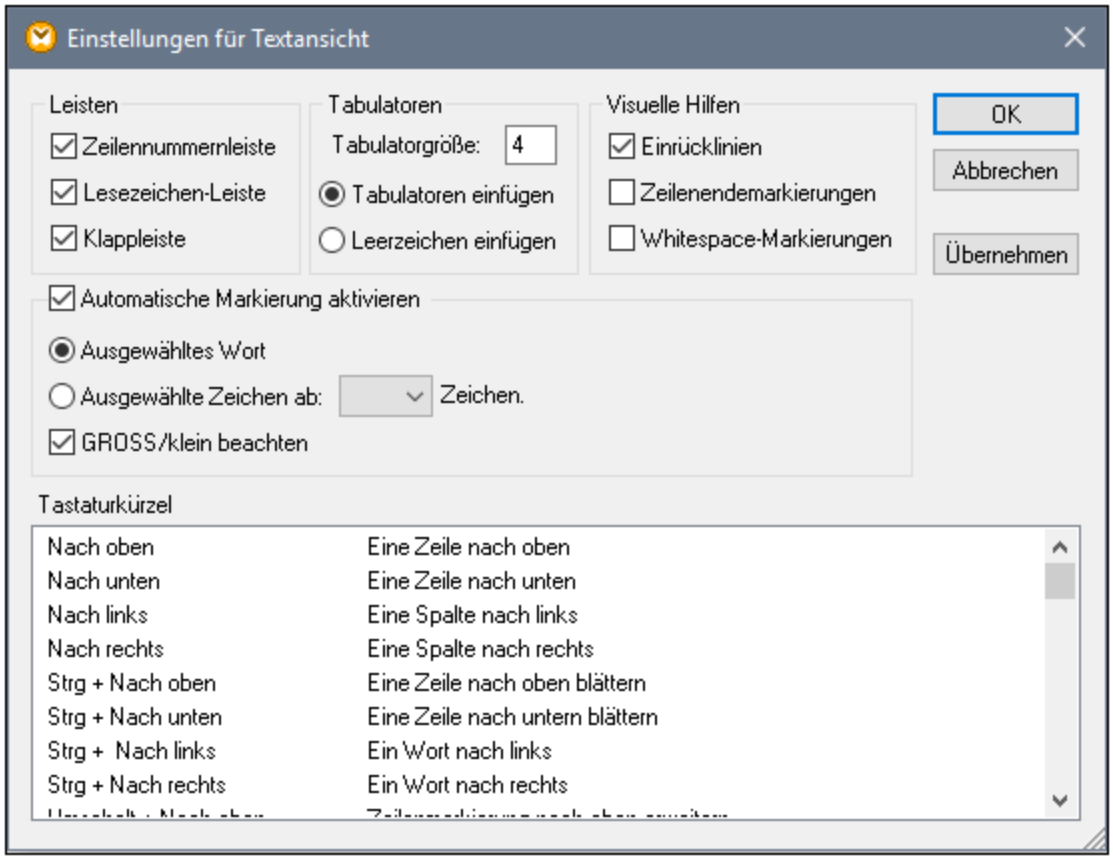
Zur Anzeige einer Vorschau auf den generierten XSLT-Code klicken Sie auf **XSLT** unterhalb des **Mapping**-Fensters. Dieselbe Anleitung lässt sich auch für jede andere der oben aufgelisteten Sprachen anwenden.

Nach Fertigstellung der Codegenerierung enthält der Zielordner die folgenden beiden Dateien:

1. Eine nach dem Zielschema benannte XSLT-Transformationsdatei. Diese Transformationsdatei hat das folgende Format: **`<Mapping>MapTo<Zieldateiname>.xslt`**. `<Mapping>` ist der Wert des Felds **Applikationsname** in den Mapping-Einstellungen[116]. `<Zieldateiname>` ist der Name der Mapping-Zielkomponente. Öffnen Sie die Einstellungen der Zielkomponente und bearbeiten Sie den Wert des Felds **Komponentenname**, um den Wert zu ändern. Nähere Informationen dazu finden Sie unter Ändern der Komponenteneinstellungen[82] und Bibliothekspfade im generierten Code[88].
2. Die Datei **`DoTransform.bat`**, mit der Sie die XSLT-Transformation mit Altova RaptorXML Server über die Befehlszeile ausführen können. Um den Befehl ausführen zu können, müssen Sie RaptorXML installieren.

3.3.3 Funktionalitäten der Textansicht

Die Fenster Ausgabe[41], XQuery[41] und XSLT[40] verfügen über eine Reihe von visuellen Hilfsmitteln für die Textanzeige, wie Leisten, Textmarkierung, Einrücklinien, Zeilenendezeichen und Whitespace-Markierungen. Sie können diese Funktionalitäten im Dialogfeld **Einstellungen für die Textansicht** anpassen **(***siehe Abbildung unten*). Die Einstellungen in diesem Dialogfeld gelten für die gesamte Applikation.



Um das Dialogfeld **Einstellungen für Textansicht** zu öffnen, wählen Sie eine der folgenden Methoden:

- Wählen Sie **Ausgabe** | **Einstellungen für Textansicht**.
- Klicken Sie in der Symbolleiste auf  (**Einstellungen für Textansicht**).
- Klicken Sie mit der rechten Maustaste in einen leeren Bereich des **Ausgabefensters** und wählen Sie im Kontextmenü den Befehl **Einstellungen für Textansicht**.

Einige der Navigationshilfen können auch über die Symbolleiste **Textansicht**, das Applikationsmenü oder Tastaturkürzel ein- und ausgeschaltet werden. Nähere Informationen zu Tastaturkürzeln finden Sie im Abschnitt **Tastaturkürzel** des oben gezeigten Dialogfelds **Einstellungen für Textansicht**.

Siehe die Liste der verfügbaren Einstellungen unten.

- Leisten

© 2016-2022 Altova GmbH

Zeilennummernleiste
Die Zeilennummern werden in der Zeilennummernleiste angezeigt, die über das Dialogfeld **Einstellungen für Textansicht** ein- und ausgeblendet werden können. Wenn ein Textabschnitt eingeklappt ist, werden auch die Zeilennummern der entsprechenden Textzeilen ausgeblendet.

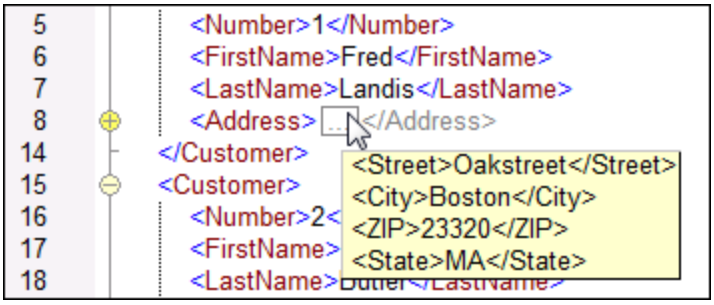
Lesezeichenleiste
Für den raschen Zugriff darauf können Zeilen in einem Dokument mit Lesezeichen versehen werden. Wenn im Dialogfeld **Einstellungen für Textansicht** das Kontrollkästchen **Lesezeichenleiste** aktiviert ist, werden die Lesezeichen in der Lesezeichenleiste angezeigt (*siehe Abbildung unten*). Wenn das Kontrollkästchen **Lesezeichenleiste** nicht aktiviert ist, werden mit Lesezeichen versehene Zeilen in Zyan markiert.



Die Bearbeitung von und Navigation zwischen Lesezeichen erfolgt über die Befehle in der Tabelle unten. Diese Befehle stehen im Menü **Ausgabe** zur Verfügung. Außerdem finden Sie die Lesezeichenbefehle im Kontextmenü, wenn Sie mit der rechten Maustaste in das Fenster **Ausgabe**, **XSLT** oder **XQuery** klicken.

	**Lesezeichen einfügen/löschen (Strg + F2)**
	**Nächstes Lesezeichen (F2)**
	**Vorhergehendes Lesezeichen (Umschalt + F2)**
	**Alle Lesezeichen löschen (Strg + Umschalt + F2)**

Klappleiste
Mit Hilfe der Klappleiste können Sie Nodes erweitern und reduzieren. Diese Funktionalität wird in der der Klappleiste angezeigt. Die Klappleiste kann über das Dialogfeld **Einstellungen für Textansicht** ein- und ausgeblendet werden. Um Textabschnitte ein- oder auszuklappen, klicken Sie auf das "+" bzw."-"-Symbol am linken Fensterrand. Eingeklappte Codeabschnitte werden mittels Auslassungspunkten markiert (*siehe Abbildung unten*). Um eine Vorschau des eingeklappten Abschnitts zu sehen, ohne diesen Abschnitt ausklappen zu müssen, platzieren Sie die Mauszeiger über die Auslassungspunkte. Daraufhin wird ein Tooltipp mit der Codevorschau angezeigt, wie in der Abbildung unten gezeigt. Wenn der Abschnitt zu groß für die Vorschau ist, wird am Ende des Tooltipps ein weiteres Auslassungssymbol angezeigt.



- Automatische Markierung aktivieren

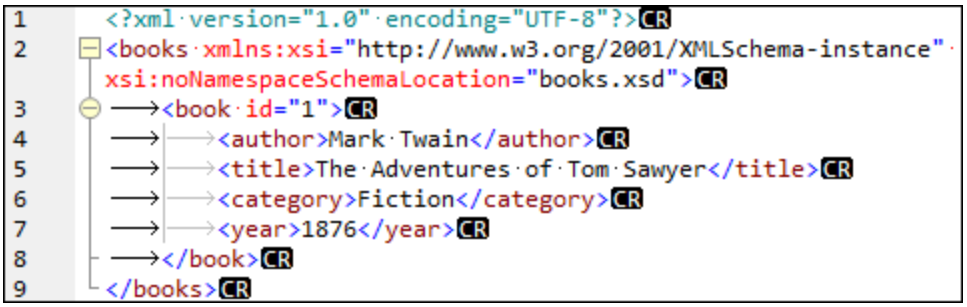
  Mit der Einstellung **Automatische Markierung aktivieren** können Sie alle Übereinstimmungen mit dem ausgewählten Textabschnitt anzeigen. Der ausgewählte Text wird hellblau markiert und die Übereinstimmungen erscheinen hellbraun markiert. Die Auswahl und die Übereinstimmungen werden in der Bildlaufleiste durch graue Markierungsquadrate gekennzeichnet. Die aktuelle Cursorposition wird in der Bildlaufleiste durch die blaue Cursormarkierung markiert. Als Auswahl kann ein ganzes Wort oder eine festgelegte Anzahl von Zeichen definiert werden. Sie können außerdem definieren, ob die Groß- und Kleinschreibung berücksichtigt werden soll.

  Bei Auswahl einer Anzahl von Zeichen können Sie die Mindestanzahl der Zeichen ab dem ersten Zeichen in der Auswahl, für die eine Übereinstimmung gefunden werden soll, definieren. So können Sie z.B. nach zwei oder mehr Zeichen suchen. Bei Wortsuchen werden folgende Einträge als separate Wörter behandelt: Elementnamen (ohne spitze Klammern), die spitzen Klammern von Element-Tags, Attributnamen und Attributwerte ohne Anführungszeichen.

- Visuelle Hilfen

  Einrücklinien
  Einrücklinien sind vertikal Linien, anhand derer Sie den Einrückungsgrad einer Zeile sehen können. Sie können im Dialogfeld **Einstellungen für Textansicht** ein- und ausgeblendet werden. Die Optionen **Tabulatoren einfügen** und **Leerzeichen einfügen** werden wirksam, wenn Sie die Option **Ausgabe | Pretty-Print** verwenden.

  Zeilenendezeichen und Whitespace-Markierungen
  Zeilenendezeichen und Whitespace-Markierungen (*siehe Abbildung unten*) können im Dialogfeld **Einstellungen für Textansicht** ein- und ausgeblendet werden. Die Pfeile stehen für Tabulatorzeichen. Die Abkürzung *CR* steht für Carriage Return (Wagenrücklauf). Die Punkte stehen für Leerzeichen.



- Weitere Einstellungen der Textansicht

 © 2016-2022 Altova GmbH

Syntaxfärbung
Eine weitere visuelle Hilfe ist die Syntaxfärbung. Sie macht Codefragmente übersichtlicher. Die Syntaxfärbung richtet sich nach dem semantischen Wert des Texts. So hat der Node-Name (und in einigen Fällen der Node-Inhalt) z.B. in XML-Dokumenten, je nachdem, ob es sich beim XML-Node um ein Element, Attribut, Inhalt, einen CDATA-Abschnitt, Kommentar oder eine Processing Instruction handelt, eine andere Farbe.

Vergrößern und Verkleinern
Durch Scrollen mit der Maus und gleichzeitiges Gedrückthalten der **Strg**-Taste können Sie in die Textansicht hinein- und daraus herauszoomen. Drücken Sie alternativ dazu die "-" bzw. "+"-Taste, während Sie die **Strg**-Taste gedrückt halten.

Pretty-Print-Anzeige
Mit dem Befehl **Pretty-Print** wird das aktive XML-Dokument in der **Textansicht** neu formatiert, um das Dokument strukturiert anzuzeigen. Standardmäßig wird jeder Child-Node wird um vier Leerzeichen vom Parent eingerückt. Diese Einstellung kann im Dialogfeld **Einstellungen für Textansicht** angepasst werden. Um ein XML-Dokument mit der Pretty-Print-Option anzuzeigen, wählen Sie den Menübefehl **Ausgabe | Pretty-Print** oder klicken Sie in der Symbolleiste auf  (**Pretty-Print**).

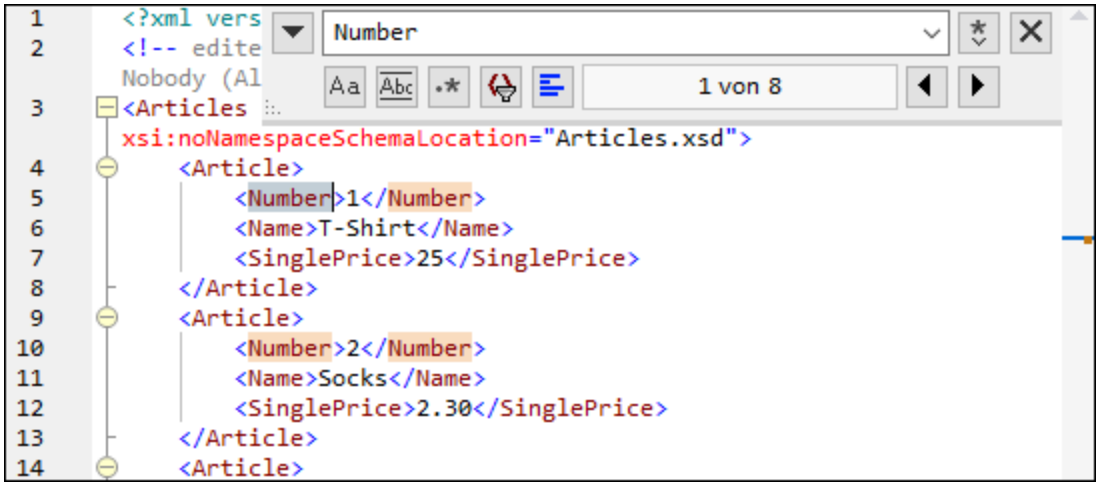
Zeilenumbruch
Mit Hilfe von Zeilenumbrüchen kann ein Codefragment innerhalb des Arbeitsbereichs angezeigt werden. Wenn Zeilenumbrüche nicht aktiviert sind, sind einige Textabschnitte im Arbeitsbereich eventuell nicht zu sehen. Um Zeilenumbrüche im gerade aktiven Dokument zu aktivieren bzw. zu deaktivieren, wählen Sie den Menübefehl **Ausgabe | Zeilenumbruch** oder klicken Sie in der Symbolleiste auf  (**Zeilenumbruch**).

3.3.4 Suchen in der Textansicht

Der Text in den Fenstern **Ausgabe**, **XQuery** und **XSLT** kann unter Verwendung einer Reihen von Optionen und visuellen Hilfsmitteln durchsucht werden.

Sie können im gesamten Dokument oder in einem ausgewählten Textbereich nach einem Begriff suchen. Drücken Sie **Strg+F** oder wählen Sie den Menübefehl **Bearbeiten | Suchen**, um eine Suche zu starten. Geben Sie den gewünschten String ein oder verwenden Sie die Auswahlliste, um einen String aus den letzten 10 Such-Strings auszuwählen. Nach Eingabe bzw. Auswahl des gewünschten String werden alle Übereinstimmungen markiert und die Position der Treffer wird durch orange Markierungen in der Bildlaufleiste gekennzeichnet (*siehe Abbildung unten*). Die Position der aktuell ausgewählten Übereinstimmung (grau markiert) hängt davon ab, wo sich der Cursor zuletzt befunden hat.

Sie sehen die Gesamtzahl der Treffer sowie die Indexposition des aktuell ausgewählten Treffers. Über die Schaltflächen  (**Gehe zu vorherigem Ergebnis**) und  (**Gehe zu nächstem Ergebnis**) gelangen Sie von einem Treffer zum nächsten.



Suchoptionen

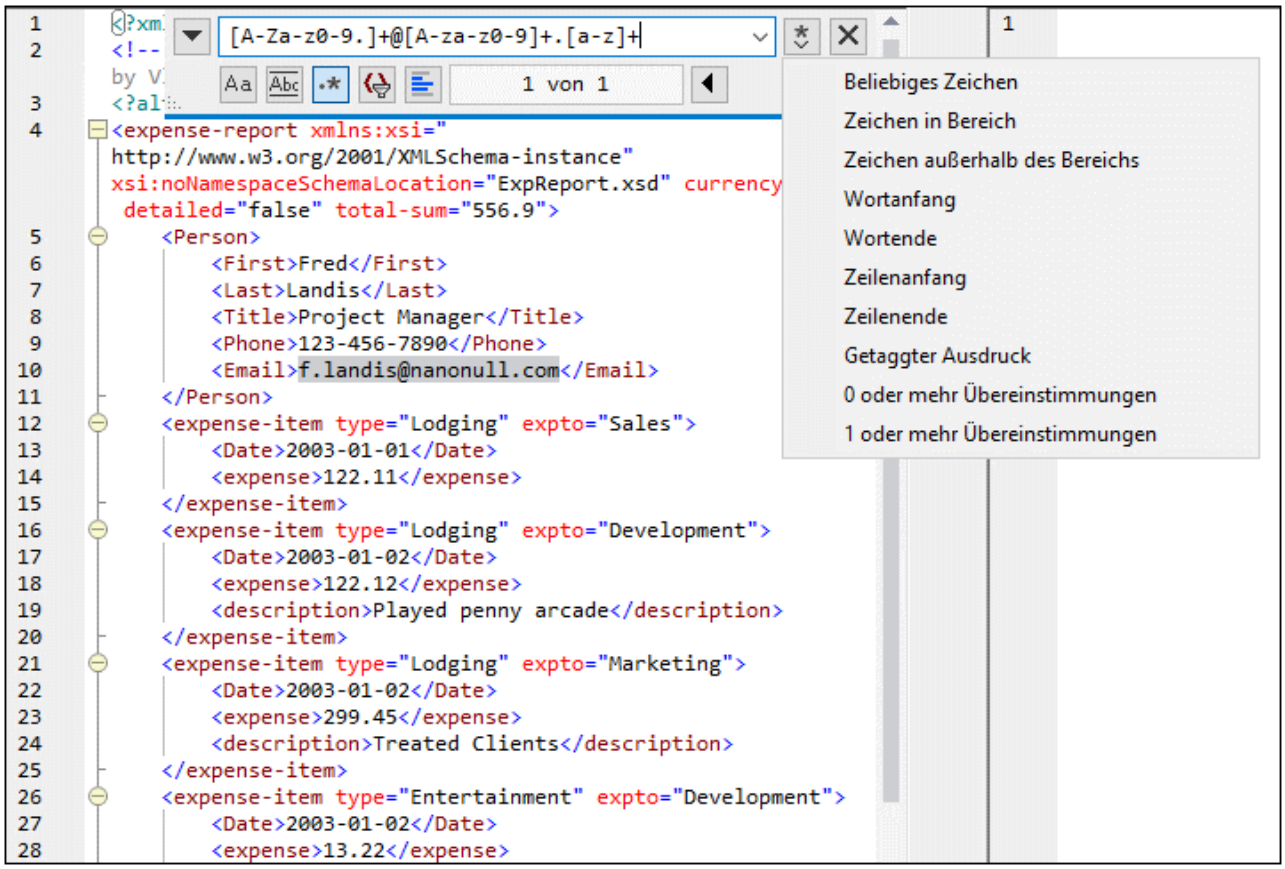
Über Schaltflächen unterhalb des Suchfelds können Sie Suchkriterien festlegen. In der folgenden Tabelle, finden Sie eine Liste der verfügbaren Optionen:

Option	Symbol	Beschreibung
GROSS/klein beachten	Aa	*Wenn die Schaltfläche aktiv ist, wird die Groß- und Kleinschreibung bei der Suche berücksichtigt (Address* ist nicht gleich *address*).
Ganzes Wort	Abc	Nur die exakte Wortentsprechung wird gefunden.
Regular Expression verwenden	.*	Wenn diese Option aktiv ist, wird der Suchbegriff als Regular Expression gelesen. Siehe *Regular Expressions* unten.
Anker suchen		Die Position des Ankers hängt von der Stelle ab, an der sich der Cursor zuletzt befunden hat. Die Position des Ankers ändert sich nicht durch Klicken auf **Gehe zu vorherigem Ergebnis**) und **Gehe zu nächstem Ergebnis**.
In Auswahl suchen		Bei einer Auswahl handelt es sich um einen markierten Textbereich. Um einen Begriff innerhalb einer Auswahl zu suchen, markieren Sie einen Textbereich, drücken Sie **Strg + F**, stellen Sie sicher, dass die Schaltfläche **In Auswahl suchen** aktiv ist und geben Sie den Suchbegriff in das Suchfeld ein.

Regular Expressions

Sie können zum Suchen eines Text-String Regular Expressions verwenden. Aktivieren Sie dazu zuerst die Option **Regular Expression** (*siehe Tabelle oben*). Geben Sie anschließend die Regular Expression in das Suchfeld ein. Bei Klick auf  (**Regular Expression Builder**) erhalten Sie eine Liste von Beispielausdrücken für Regular Expressions (*siehe unten*). In der Abbildung unten sehen Sie eine Regular Expression zum Suchen von E-Mail-Adressen.

© 2016-2022 Altova GmbH



Regular Expression-Metazeichen

Die unten stehende Tabelle enthält Metazeichen, die Sie zum Suchen und Ersetzen von Text verwenden können. Alle Metazeichen mit Ausnahme der beiden letzten entsprechen Menüeinträgen im **Regular Expression Builder** (*siehe oben*).

Menübefehl	Metazeichen	Beschreibung
Beliebiges Zeichen	`.`	Steht für jedes beliebige Zeichen. Dies ist ein Platzhalter für ein einzelnes Zeichen.
Zeichen im Bereich	`[...]`	Steht für jedes beliebige Zeichen in dieser Gruppe. **`[abc]`** z.B. steht für jedes der Zeichen `a`, `b` oder `c`. Sie können auch Bereiche angeben, z.B. **`[a-z]`** für alle klein geschriebenen Zeichen.
Zeichen nicht im Bereich	`[^...]`	Steht für jedes beliebige Zeichen in dieser Gruppe. **`[^A-Za-z]`** z.B. steht für jedes Zeichen mit Ausnahme alphabetischer Zeichen.
Wortanfang	`\<`	Steht für den Anfang eines Worts.
Wortende	`\>`	Steht für das Ende eines Worts.
Zeilenanfang	`^`	Steht für den Zeilenanfang, es sei denn dieses Zeichen wird innerhalb einer Menge verwendet (*siehe oben*).

Menübefehl	Metazeichen	Beschreibung
Zeilenende	$	Steht für das Zeilenende. Beispiel: **A+$** findet ein oder mehrere As am Ende der Zeile.
Getaggter Ausdruck	(abc)	Die Klammern markieren Beginn und Ende eines getaggten Ausdrucks. Getaggte (markierte) Ausdrücke eignen sich dazu, eine gesuchte Region zu markieren ("sich diese zu merken"), um diese später referenzieren zu können. Es können bis zu neun Unterausdrücke getaggt und später rückreferenziert werden. So wird etwa mit **(the) \1** der String the the gefunden. Diese Funktion bedeutet Folgendes: Suche den String the und merke ihn Dir als getaggte Region; der Ausdruck muss von einem Leerzeichen und einer Rückreferenz auf die zuvor gesuchte getaggte Region gefolgt sein.
0 oder mehr Übereinstimmungen	*	Steht für 0 oder mehr Übereinstimmungen mit dem vorhergehenden Ausdruck. Mit **Sa*m** werden z.B. Sm, Sam, Saam, Saaam usw. gefunden.
1 oder mehr Übereinstimmungen	+	Steht für 1 oder mehrere Instanzen des vorhergehenden Ausdrucks. Mit **Sa+m** werden z.B. Sam, Saam, Saaam usw. gefunden.
	\n	n steht für 1 bis 9 und bezieht sich auf die erste bis neunte getaggte Region (*siehe oben*).
	\x	Damit können Sie ein Zeichen x verwenden, das sonst eine spezielle Bedeutung hätte. So würde z.B. \[als [und nicht als der Beginn einer Zeichengruppe interpretiert werden.

Suchen von Sonderzeichen

Wenn die Option **Regular Expression verwenden** aktiv ist, können Sie im Text nach einem beliebigen der folgenden Sonderzeichen suchen.

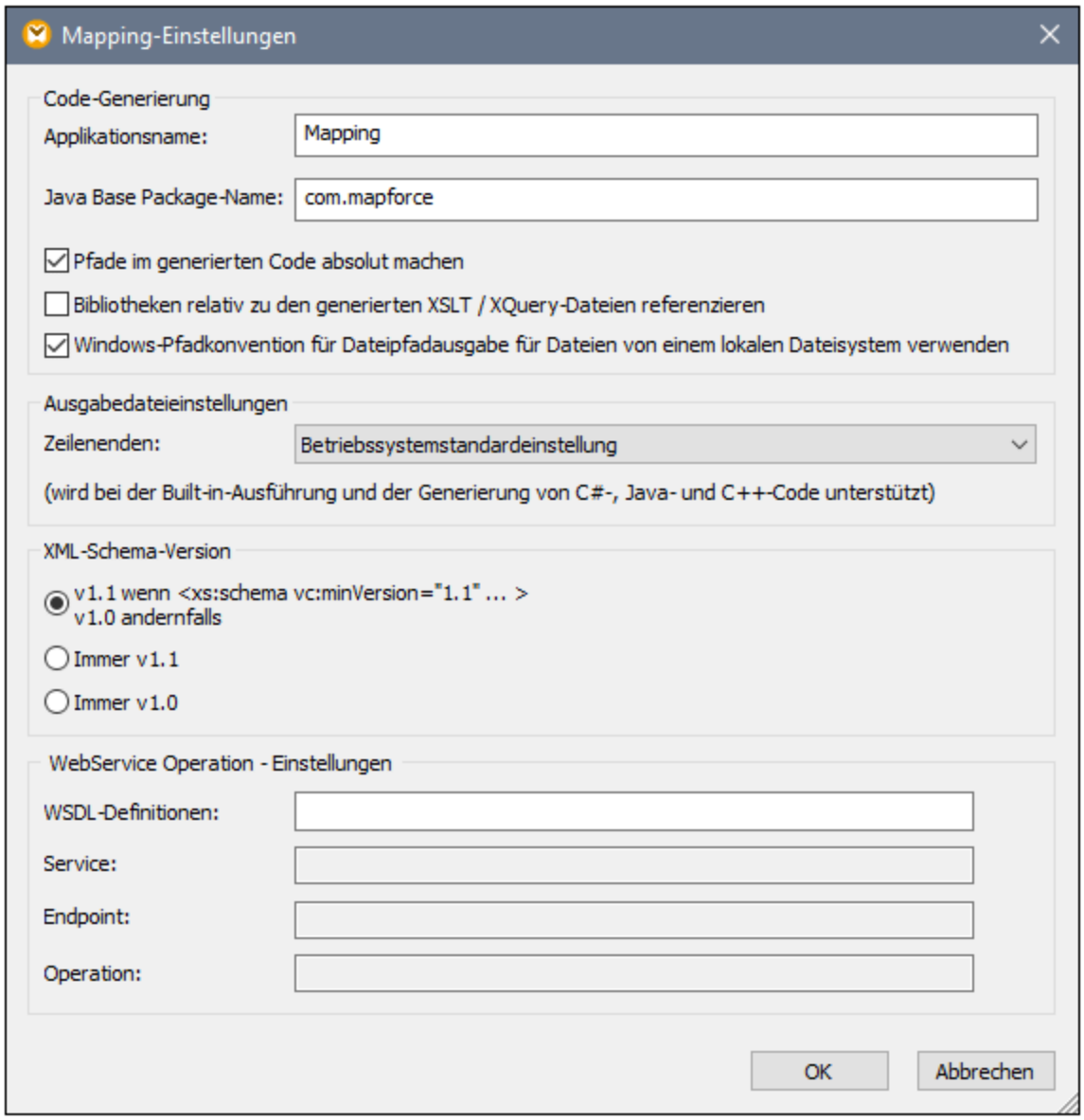
- \t (Tab)
- \r (Wagenrücklauf)
- \n (Neue Zeile)
- \\ (Umgekehrter Schrägstrich)

Um z.B. ein Tabulatorzeichen zu suchen, drücken Sie **Strg + F**, aktivieren Sie die Option **Regular Expression verwenden** und geben Sie in das **Suchfeld** **\t** ein.

3.3.5 Mapping-Einstellungen

Im Dialogfeld **Mapping-Einstellungen** (*siehe Abbildung unten*) können Sie dokumentspezifische Einstellungen definieren. Um dieses Dialogfeld zu öffnen, gehen Sie zum Menü **Datei** und klicken Sie auf **Mapping-Einstellungen**. Klicken Sie alternativ dazu mit der rechten Maustaste in einen leeren Bereich des Mappings und wählen Sie im Kontextmenü den Befehl **Mapping-Einstellungen**.

© 2016-2022 Altova GmbH



Weiter unten finden Sie eine Beschreibung der verfügbaren Einstellungen.

- Codegenerierung
  - *Applikationsname:* Definiert das Präfix der generierten XSLT-Datei oder den Namen der generierten Java-, C#- oder C++-Applikation (*Professional und Enterprise Edition*).
  - *Java Base Package-Name (Professional und Enterprise Edition)* Diese Option ist anwendbar, wenn als Transformationssprache Java ausgewählt ist. Sie definiert den Base Package-Namen für die Java-Ausgabe.
  - Pfade im generierten Code absolut machen: Dieses Kontrollkästchen wirkt sich auf alle Pfade in Mapping-Komponenten mit Ausnahme von Pfaden zu externen Bibliotheksdateien (wie z.B. XSLT-Bibliotheken) aus. Über das Kontrollkästchen wird definiert, ob die Dateipfade im generierten

Programmcode und in MapFlorce Server-Ausführungsdateien (`.mfx`) und auf FlowForce Server bereitgestellten Mapping-Funktionen relativ oder absolut sein sollen. Nähere Informationen dazu finden Sie unter Pfade in Ausführungsumgebungen 87.

- *Bibliotheken relativ zu den generierten XSLT / XQuery-Dateien referenzieren:* Dieses Kontrollkästchen wird angewendet, wenn die Mapping-Sprache entweder XQuery (*Professional und Enterprise Edition*) oder XSLT ist. Normalerweise ist diese Option nützlich, wenn in Ihrem Mapping eine XSLT- oder XQuery-Bibliothek referenziert wird und Sie beabsichtigen, anhand des Mappings XSLT- oder XQuery-Dateien zu generieren. Aktivieren Sie dieses Kontrollkästchen, wenn die Bibliothekspfade relativ zum Verzeichnis des generierten XSLT- oder XQuery-Code gemacht werden sollen. Wenn das Kontrollkästchen deaktiviert ist, werden die Bibliothekspfade im generierten Code absolut. Siehe auch Bibliothekspfade im generierten Code 88.

- *Windows-Pfadkonvention für Dateipfadausgaben von einem lokalen Dateisystem verwenden:* Dieses Kontrollkästchen wird angewendet, wenn die Mapping-Sprache entweder XQuery (*MapForce Professional und Enterprise Edition*), XSLT 2.0 oder XSLT 3.0 ist. Durch Aktivieren des Kontrollkästchens stellen Sie sicher, dass die Windows-Pfadkonventionen eingehalten werden. Bei der Ausgabe von XSLT 2.0-, XSLT 3.0- oder XQuery-Dokumenten wird die aktuell verarbeitete Datei intern über die **`document-uri`**-Funktion aufgerufen, die einen Pfad im Format `file://URI` für lokale Dateien zurückgibt. Wenn dieses Kontrollkästchen aktiviert ist, wird eine `file://URI`-Pfadspezifikation automatisch in einen vollständigen Windows-Dateipfad (z.B. `C:\...`) konvertiert, um die weitere Verarbeitung zu vereinfachen.

Ausgabedateieinstellungen (Professional und Enterprise Edition)

Über die Auswahlliste **Zeilenenden** können Sie die Zeilenenden der Ausgabedateien definieren. *Betriebssystemstandardeinstellung* ist die jeweilige Standardeinstellung für das Zielbetriebssystem, z.B. Windows (CR+LF), macOS (LF), oder Linux (LF). Sie können auch manuell ein bestimmtes Zeilenende auswählen. Die hier ausgewählten Einstellungen sind wichtig, wenn Sie ein Mapping zu einer MapForce Server-Ausführungsdatei (`.mfx`) kompilieren oder wenn Sie ein Mapping auf einem FlowForce Server bereitstellen, der auf einem anderen Betriebssystem installiert ist.

XML-Schema-Version

Diese Option dient zum Definieren der in der Mapping-Datei verwendeten XML-Schema-Version. Beachten Sie, dass nicht alle Version 1.1-spezifischen Funktionen derzeit unterstützt werden. Wenn die Deklaration `xs:schema vc:minVersion="1.1"` vorhanden ist, wird Version 1.1 verwendet; falls nicht, wird Version 1.0 verwendet.

Wenn das XSD-Dokument kein `vc:minVersion` Attribut hat oder der Wert des Attributs `vc:minVersion` nicht `1.0` oder `1.1` ist, wird XSD 1.0 als Standardmodus verwendet. Verwechseln Sie das `vc:minVersion` Attribut nicht mit dem `xsd:version` Attribut. Das erste Attribut hat die XSD-Versionsnummer, während das zweite die Dokument-Versionsnummer enthält. Wenn diese Einstellung in einem vorhandenen Mapping geändert wird, werden alle Schemas der ausgewählten XML-Schema-Version neu geladen, was sich auch auf die Gültigkeit auswirken kann.

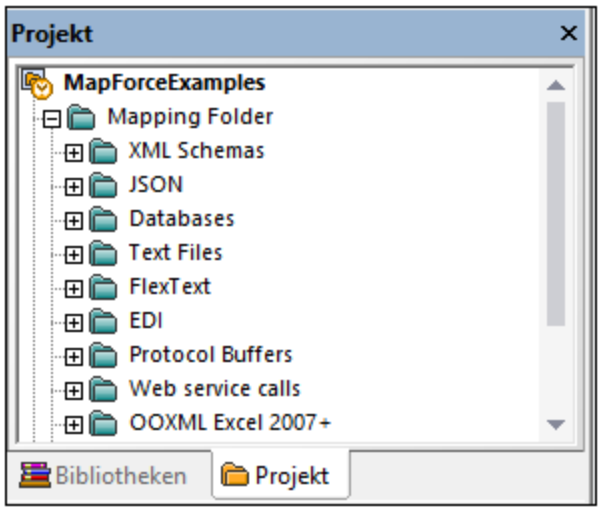
Webservice Operation - Einstellungen (Enterprise Edition)

Die Felder **WSDL-Definitionen**, **Service**, **Endpoint** und **Operation** werden automatisch ausgefüllt, wenn das Mapping-Dokument Teil einer Webservice-Implementierung ist.

 © 2016-2022 Altova GmbH

3.4 Projekte

Neben eigenständigen Mappings können Sie auch Mapping-Projekte erstellen, die mehrere Mappings enthalten. Zu einem Projekt hinzugefügte Mappings können über das **Projektfenster** aufgerufen werden (*siehe Abbildung unten*).



Der Hauptvorteil von Projekten ist, dass Sie für alle im Projekt inkludierten Mappings gemeinsame Codegenerierungseinstellungen (wie z.B. Zielsprache und Ausgabeverzeichnis) definieren können. Sie können auch Ordner innerhalb von Projekten erstellen und für die einzelnen Ordner eigene Codegenerierungseinstellungen definieren. Nähere Informationen zu von MapForce generiertem Programmcode (in C++, C#, and Java) finden Sie im Code Generator[1563]. MapForce-Projektdateien haben die Erweiterung **`.mfp`**.

In der MapForce Enterprise Edition können Sie zusätzlich dazu auch Webservice-Projekte erstellen. Mit Webservice-Projekten können Sie Java- oder C#-Programmcode, der SOAP-Webservices auf Basis vorhandener WSDL- (Web Services Description Language) -Dateien implementiert, generieren. Nähere Informationen zu Webservice-Projekten finden Sie unter Implementieren von Webservices[1255].

3.4.1 Grundlegendes zu Projekten

Die folgenden Unterabschnitte enthalten eine Einführung in Projekte. Das Arbeiten mit Projekten kann in die folgenden drei Kategorien eingeteilt werden: (i) Erstellen eines Projekts, (ii) Organisieren eines Projekts und (ii) Durchführen verschiedener Aktionen daran. Eine Beschreibung dazu finden Sie in den Unterabschnitten weiter unten.

Neues Projekt

Um ein neues Projekt zu erstellen, gehen Sie folgendermaßen vor:

1. Klicken Sie in der Symbolleiste auf die Schaltfläche . Gehen Sie alternativ dazu zum Menü **Datei** und klicken Sie auf **Neu**.

2. Wählen Sie **Projektdatei** und klicken Sie auf **OK**.
3. Geben Sie den Projektnamen im Dialogfeld **Projekt speichern unter** ein und klicken Sie auf **Speichern**. Das neue Projekt wird nun im **Projektfenster** angezeigt.

Um ein Projekt zu schließen, gehen Sie zum Menü **Projekt** und klicken Sie auf **Projekt schließen**.

Projektorganisation

Hinzufügen eines Mappings zu einem Projekt
Um das aktive Mapping zum Projekt hinzuzufügen, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Projekt** auf **Aktive Datei zu Projekt hinzufügen**.
- Klicken Sie im **Projektfenster** mit der rechten Maustaste auf das entsprechende Projekt und wählen Sie im Kontextmenü **Aktive Datei zu Projekt hinzufügen** .

Um vorhandene Mapping-Dateien zu einem Projekt hinzuzufügen, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Projekt** auf **Dateien zu Projekt hinzufügen** .
- Klicken Sie im **Projektfenster** mit der rechten Maustaste auf das entsprechende Projekt und wählen Sie im Kontextmenü **Dateien zu Projekt hinzufügen**.

Tipp: Um mehrere Dateien hinzuzufügen, halten Sie die **Strg**-Taste gedrückt, während Sie die Dateien im Dialogfeld **Öffnen** auswählen.

Löschen einer Datei aus einem Projekt
Um eine Datei oder einen Ordner aus einem Projekt zu entfernen, wählen Sie eine der folgenden Methoden:

- Wählen Sie die zu löschende Datei im **Projektfenster** aus. Klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie im Kontextmenü den Befehl **Löschen**.
- Wählen Sie die entsprechenden Dateien im **Projektfenster** aus und drücken Sie **Entf**.

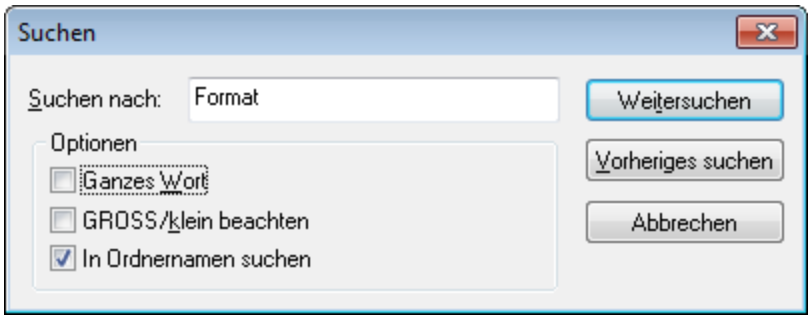
MapForce-Projektdateien haben die Erweiterung **`.mfp`**. Bestehende MapForce-Projekte können auf dieselbe Art wie Mappings geöffnet werden: Gehen Sie zum Menü **Datei** und klicken Sie auf **Öffnen**. Wenn Sie MapForce zum ersten Mal starten, wird im **Projektfenster** standardmäßig das Projekt **`MapForceExamples.mfp`** geöffnet.

Aktionen im Zusammenhang mit Projekten

Durchsuchen eines Projekts
Um in einem Projekt nach Dateien zu suchen, gehen Sie folgendermaßen vor:

1. Klicken Sie im **Projektfenster** auf das Projekt oder den Ordner, den Sie durchsuchen möchten.
2. Drücken Sie **Strg + F**. Daraufhin können Sie im Dialogfeld **Suchen** Ihre Suchoptionen definieren. Wenn Sie auch Ordnernamen in die Suche miteinbeziehen möchten, aktivieren Sie die Option **In Ordnernamen suchen** (*siehe Abbildung unten*).

© 2016-2022 Altova GmbH

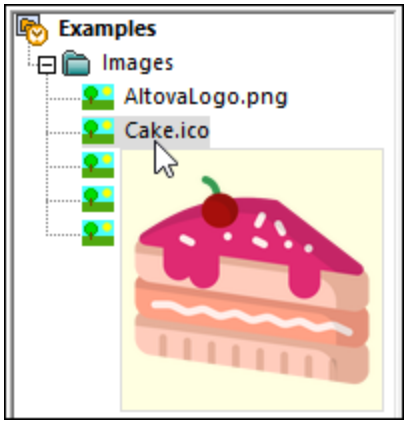


Code für das gesamte Projekt generieren
Sie können in Projekten Code für (i) einzelne Mappings, (ii) einen bestimmten Ordner oder (iii) das gesamte Projekt generieren.

Um Code für ein Mapping oder einen Ordner in Ihrem Projekt zu generieren, klicken Sie mit der rechten Maustaste auf das entsprechende Mapping oder den entsprechenden Ordner und wählen Sie **Code generieren** oder **Code generieren in**. Wenn Sie **Code generieren** wählen, wird der Code in der in den Projekteinstellungen[122] definierten Sprache generiert. Sie können Code wahlweise auch in einer der in Ihrer MapForce Edition verfügbaren Sprachen generieren. Nähere Informationen dazu finden Sie unter Codegenerierung[109].

Um Code für das gesamte Projekt zu generieren, klicken Sie im Menü **Projekt** auf **Code für das gesamte Projekt generieren**. Klicken Sie alternativ dazu mit der rechten Maustaste im **Projektfenster** auf den Namen des Projekts und wählen Sie den Befehl **Code generieren**. Daraufhin wird der Code in der in den Projekteinstellungen[122] definierten Sprache generiert. Sie können auch im Menü **Projekt** oder im Kontextmenü des Projekts eine Sprache für die Codegenerierung für das gesamte Projekt auswählen. Welche Sprachen zur Auswahl stehen, hängt von Ihrer MapForce Edition ab. Nähere Informationen dazu finden Sie unter Codegenerierung[109].

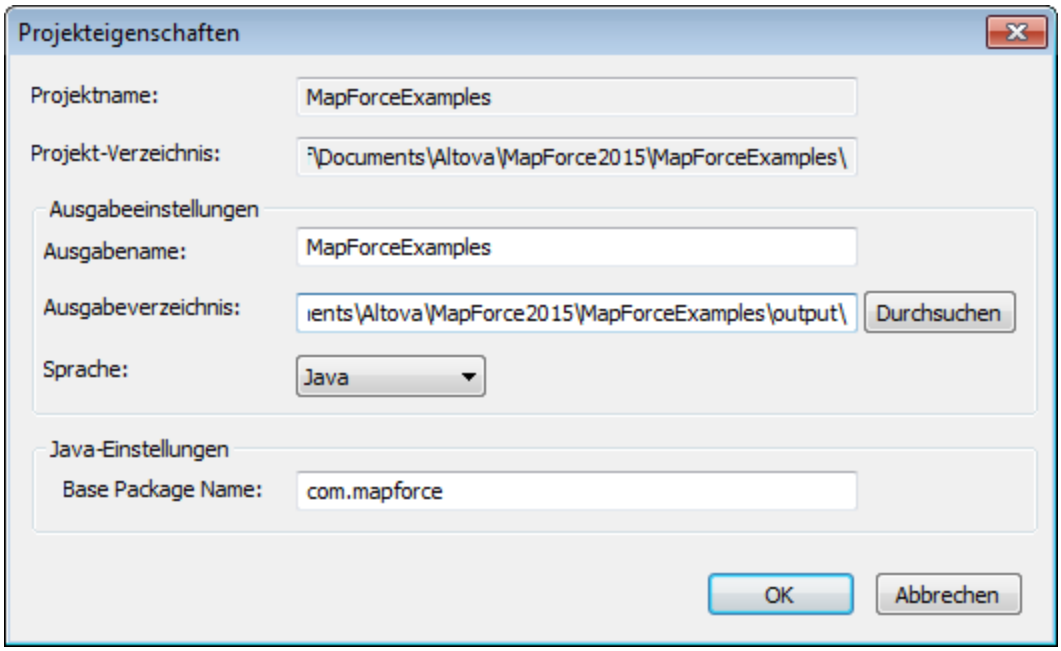
Bildvorschau
Sie können nun im Fenster **Projekt** eine Vorschau auf Bilder der folgenden Formate anzeigen: **`.png`**, **`.jpeg`**, **`.gif`**, **`.bmp`**, **`.tiff`** und **`.ico`** (*siehe Abbildung unten*). Durch Doppelklick auf eine Bilddatei wird diese in einer externen Applikation geöffnet (je nach Dateiverknüpfung in Windows).



3.4.2 Projekteinstellungen

Sie können für jedes Projekt Codegenerierungseinstellungen definieren, die sich auf alle Mappings in Ihrem Projekt auswirken. Um das Dialogfeld **Projekteinstellungen** (*siehe Abbildung unten*) zu öffnen, wählen Sie eine der folgenden Methoden:

- Klicken Sie mit der rechten Maustaste im Fenster **Projekt** auf den Projektnamen und wählen Sie im Kontextmenü den Befehl **Eigenschaften**.
- Klicken Sie im Menü **Projekt** auf **Eigenschaften**



Es stehen die folgenden Einstellungen zur Verfügung. Beachten Sie, dass der Projektname und das Projektverzeichnis nach Erstellung des Projekts nicht mehr geändert werden können.

- *Ausgabename:* Der Wert, den Sie in dieses Textfeld eingeben, legt den Namen des generierten Projekts bzw. der generierten Lösung sowie aller anderen Objektnamen im generierten Code fest.
- *Ausgabeverzeichnis:* Definiert den Windows-Ordner, in dem der generierte Code aus allen Mappings in diesem Projekt gespeichert wird. Standardmäßig wird die Ausgabe im Verzeichnis **`MapForceExamples\output\`** gespeichert.
- *Sprache:* Definiert die Codegenerierungssprache für alle Mapping-Dateien in diesem Projekt.
- *Base Package Name:* Diese Option ist anwendbar, wenn als Transformationssprache Java ausgewählt ist. Sie definiert den Namen des Basispakets im generierten Java-Projekt.

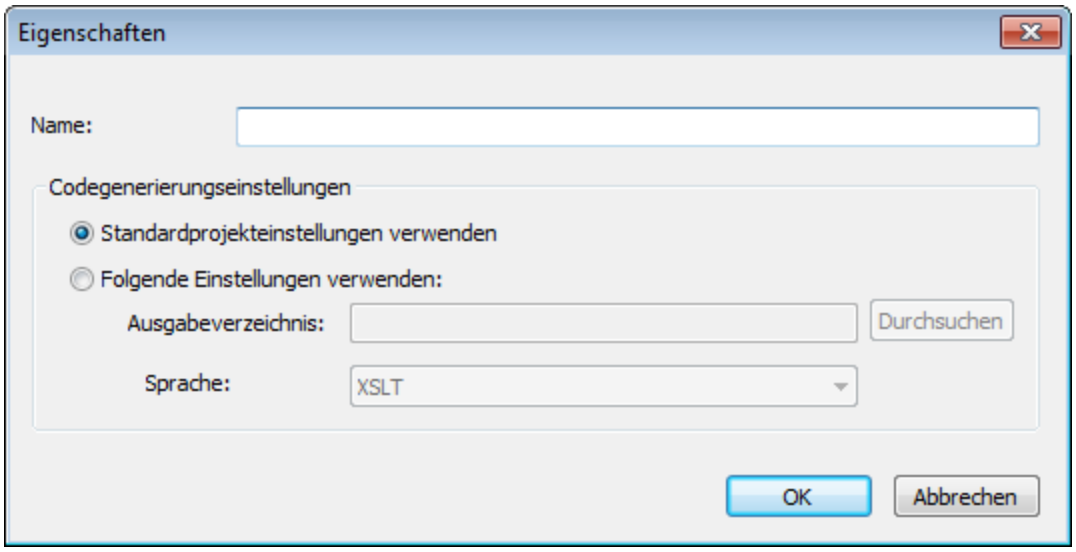
3.4.3 Projektordner

Sie können in MapForce Mappings in einem Projekt in Ordnern gliedern. Sie können so viele Ordner wie nötig erstellen und Mappings zu diesen hinzufügen. Es handelt sich hierbei um virtuelle Ordner, die nur im MapForce-Projekt vorhanden sind. Sie haben keine Entsprechung zu Ordnern auf Ihrem Rechner. Einer der

© 2016-2022 Altova GmbH

Vorteile von Projektordnern ist, dass Sie gemeinsame Codegenerierungseinstellungen für alle Mapping-Dateien in diesem Ordner definieren können. Um in einem MapForce-Projekt einen Ordner zu erstellen, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Projekt** auf **Ordner erstellen**. Klicken Sie alternativ dazu mit der rechten Maustaste im **Projektfenster** auf das Projekt und wählen Sie den Befehl **Ordner erstellen**.
2. Geben Sie im Dialogfeld **Eigenschaften** (*siehe Abbildung unten*) die gewünschten Codegenerierungeinstellungen ein und klicken Sie auf **OK**.



In der folgenden Liste sind die Einstellungen beschrieben, die Sie im Dialogfeld **Eigenschaften** definieren können.

- *Name:* Dies ist der Name des Ordners in Ihrem Projekt.
- *Standardprojekteinstellungen verwenden:* Diese Option bedeutet, dass die Codegenerierungseinstellungen im aktuellen Ordner mit denen des gesamten Projekts übereinstimmen. Wenn Sie daher Code anhand Ihres Projekts generieren, verwendet MapForce die in den Projekteinstellungen[122] definierten Codegenerierungseinstellungen. Wenn für Ihren Ordner eigene Codegenerierungseinstellungen vorgenommen werden müssen, wählen Sie die Option **Folgende Einstellungen verwenden** und definieren Sie das Codeausgabeverzeichnis und die benötigte Sprache.
- *Ausgabeverzeichnis:* Dies ist der Ordner, in dem der generierte Code aus allen Mappings in diesem Ordner gespeichert wird.
- *Sprache:* Definiert die Codegenerierungssprache für alle Mapping-Dateien in diesem Ordner.

4 Strukturkomponenten

Dieser Abschnitt enthält Informationen über verschiedene Datenformate, die als Quell- und Zielkomponenten verwendet werden können.

- XML und XML-Schema 125
- Datenbanken 165
- EDI 460 (Dazu gehören EDI-basierte Formate oder Formate im Zusammenhang mit EDI wie z.B. ASC X12, HIPAA X12, HL7 Version 2, IATA PADIS, NCPDP SCRIPT, SAP IDoc, TRADACOMS, UN/EDIFACT)
- CSV- und Textdateien 585
- Webservices 610
- MapForce Flextext 684
- Microsoft OOXML Excel 2007+ 738
- XBRL 768
- JSON 821
- Protocol Buffer 843
- Binärdateien 860

© 2016-2022 Altova GmbH

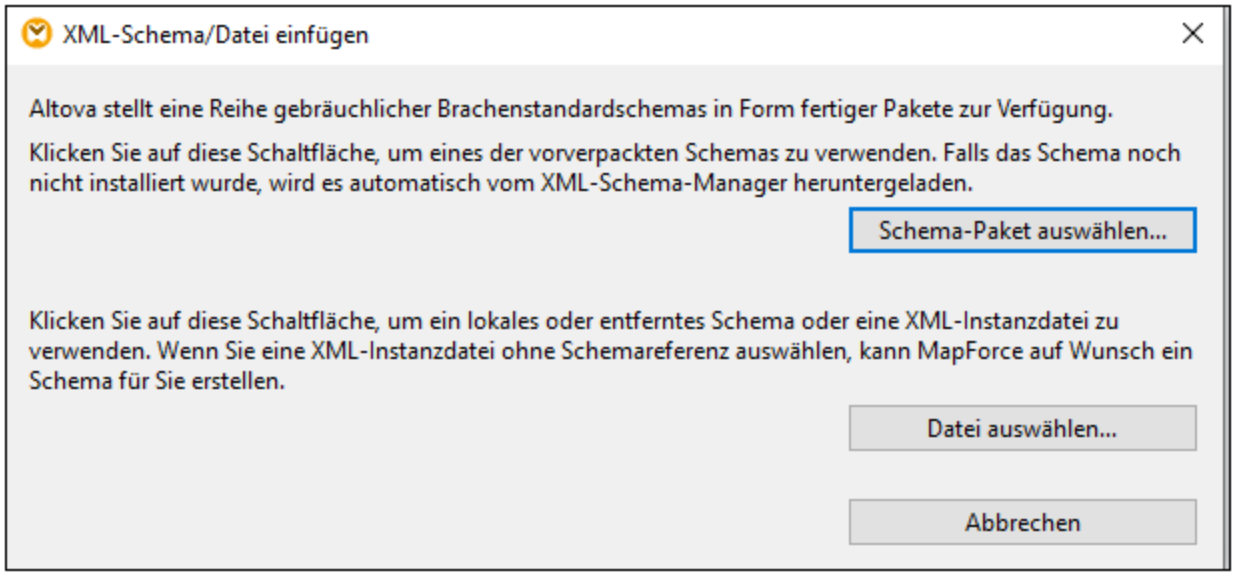
4.1 XML und XML-Schema

Altova Website: XML-Mapping

Dieser Abschnitt enthält Informationen über XML-Komponenten-spezifische Aspekte. Informationen zu grundlegenden Datentransformationsszenarien finden Sie in den Tutorials[43].

Einfügen eines XML-Schemas/einer XML-Datei

Mit dem Menübefehl **Einfügen | XML-Schema/Datei** oder über die Symbolleisten-Schaltfläche können Sie ein XML-Schema/eine XML-Datei einfügen. Im Dialogfeld (*siehe Abbildung unten*) werden Sie gefragt, ob Sie ein vorverpacktes Branchenstandardschema oder ein lokales bzw. eine entfernte Schema/Instanzdatei einfügen möchten. Wenn Sie sich für ein vorverpacktes Schema entscheiden, werden Sie aufgefordert, einen Eintrittspunkt auszuwählen. Wenn das gewünschte Schema noch nicht installiert ist, wird es automatisch vom XML-Schema-Manager[149] heruntergeladen.



Generieren eines XML-Schemas
Wenn Sie eine lokale oder entfernte XML-Datei ohne eine Schemareferenz hinzufügen, wird Ihnen von MapForce vorgeschlagen, ein XML-Schema zu generieren. Anschließend werden Sie gefragt, in welchem Verzeichnis das generierte Schema gespeichert werden soll.

Bei Generierung eines Schemas anhand einer XML-Datei müssen die Datentypen für Elemente/Attribute anhand des XML-Instanzdokuments abgeleitet werden und entsprechen eventuell nicht genau Ihren Erwartungen. Überprüfen Sie bitte, ob das generierte Schema tatsächlich die Instanzdaten genau darstellt.

Wenn Elemente oder Attribute in mehr als einem Namespace vorhanden sind, generiert MapForce für jeden Namespace ein separates XML-Schema, es können daher mehrere Dateien auf der Festplatte generiert werden.

DTD als Dokumentstruktur
Versionen ab MapForce 2006 SP2 unterstützen Namespace-fähige DTDs für Quell- und Zielkomponenten. Die Namespace URIs werden aus den DTD `xmlns`-Attributdeklarationen extrahiert, um Mappings zu ermöglichen. Einige DTDs enthalten allerdings `xmlns*`-Attributdeklarationen ohne Namespace-URIs (z.B. von StyleVision verwendete DTDs). Um solche DTDs in MapForce verwenden zu können, definieren Sie das `xmlns`-Attribut mit der Namespace-URI folgendermaßen:

```
<!ATTLIST fo:root
   xmlns:fo CDATA #FIXED 'http://www.w3.org/1999/XSL/Format'
   ...
>
```

In diesem Abschnitt

Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

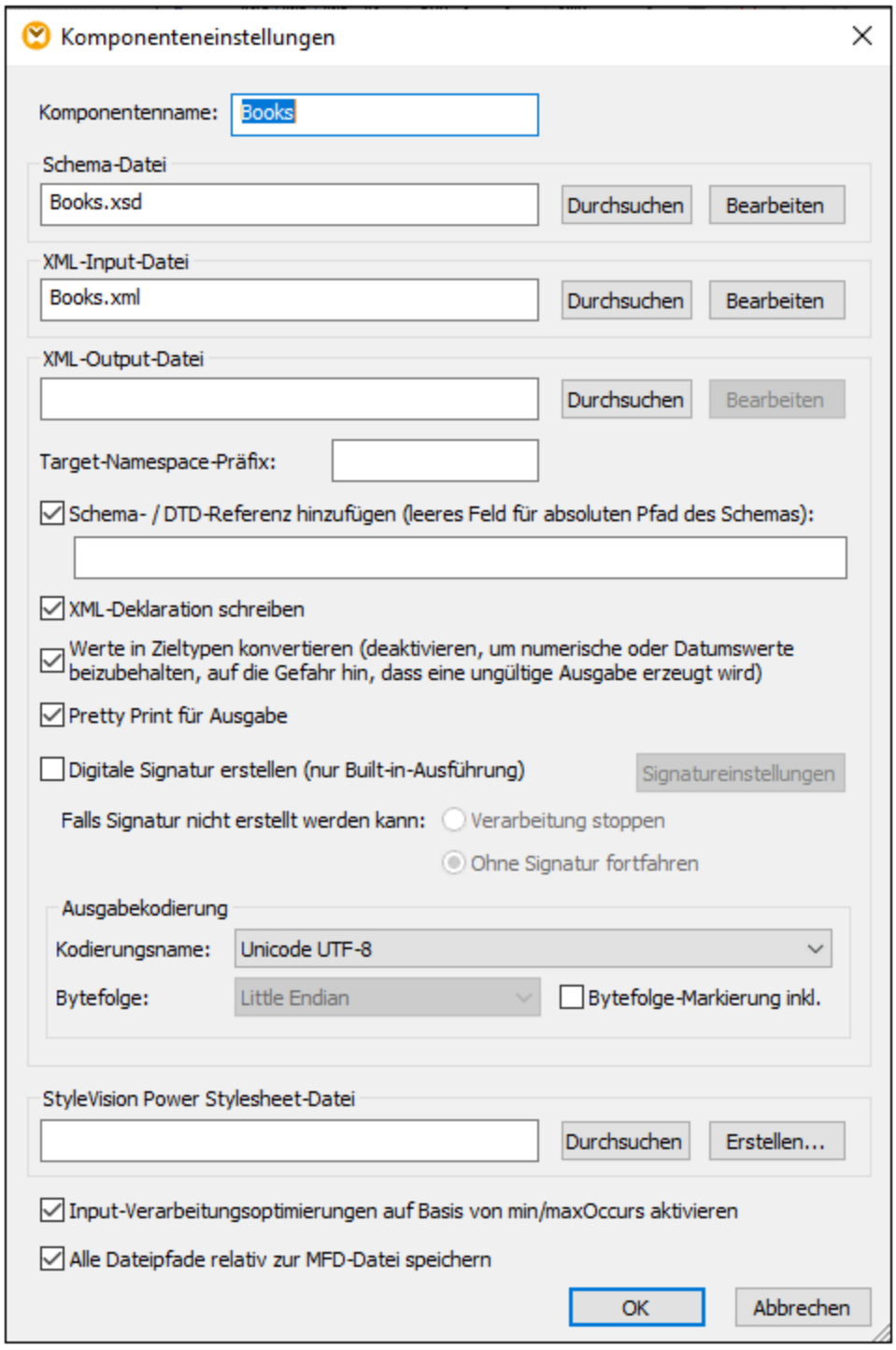
- XML-Komponenteneinstellungen 126
- Abgeleitete Typen 130
- NULL-Werte 132
- Kommentare und Processing Instructions 136
- CDATA-Abschnitte 137
- Wildcards: xs:any/xs:anyAttribute 138
- Benutzerdefinierte Namespaces 141
- Digitale Signaturen 143
- XML-Schema-Manager 149

4.1.1 XML-Komponenteneinstellungen

Nachdem Sie eine XML-Komponente zum Mapping-Bereich hinzugefügt haben, können Sie die Einstellungen dafür über das Dialogfeld **Komponenteneinstellungen** (*siehe Abbildung unten*) konfigurieren. Sie können das Dialogfeld **Komponenteneinstellungen** auf eine der folgenden Arten öffnen:

- durch Doppelklick auf den Komponententitel
- durch Klick mit der rechten Maustaste auf den Komponententitel und Auswahl von **Eigenschaften**
- durch Auswahl der Komponente im Mapping und Klick auf **Eigenschaften** im Menü **Komponente**

© 2016-2022 Altova GmbH



Weiter unten finden Sie eine Beschreibung der verfügbaren Einstellungen.

Allgemeine Einstellungen

- Komponentenname

Der Komponentenname wird bei der Erstellung der Komponente automatisch generiert. Sie können den Namen jedoch jederzeit ändern. Der Komponentenname kann Leerzeichen und Punkte enthalten, darf aber keine Schrägstriche, umgekehrten Schrägstriche, Doppelpunkte, doppelten Anführungszeichen und voran- oder nachgestellte Leerzeichen enthalten. Beachten Sie beim Ändern des Namens einer Komponente Folgendes:

- Wenn Sie das Mapping auf FlowForce Server bereitstellen möchten, muss der Komponentenname eindeutig sein.
- Es wird empfohlen, nur Zeichen zu verwenden, die über die Befehlszeile eingegeben werden können. Länderspezifische Sonderzeichen sind in Windows und der Befehlszeile eventuell unterschiedlich kodiert.

Schema-Datei

Definiert den Namen und Pfad der XML-Schema-Datei, die von MapForce zum Validieren und Mappen der Daten verwendet wird. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie eine neue Datei aus. Um die Datei in Altova XMLSpy zu bearbeiten, klicken Sie auf **Bearbeiten**.

XML-Input-Datei

Definiert die XML-Instanzdatei, aus der MapForce die Daten ausliest. Dieses Feld hat bei einer Quellkomponente eine Bedeutung und wird ausgefüllt, wenn Sie die Komponente erstellen und dieser Komponente eine XML-Instanzdatei zuweisen. In einer Quellkomponente werden anhand des Namens der Instanzdatei das XML-Root-Element und das referenzierte Schema, anhand dessen die Datei validiert werden soll, ermittelt. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie eine neue Datei aus. Um die Datei in Altova XMLSpy zu bearbeiten, klicken Sie auf **Bearbeiten**.

XML-Output-Datei

Definiert die XML-Instanzdatei, in die MapForce die Daten schreibt. Dieses Feld hat bei einer Zielkomponente eine Bedeutung. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie eine neue Datei aus. Um die Datei in Altova XMLSpy zu bearbeiten, klicken Sie auf **Bearbeiten**.

Target-Namespace-Präfix

Hier können Sie ein Präfix für den Target Namespace eingeben. Stellen Sie sicher, dass der Target Namespace im Zielschema definiert ist, bevor Sie ein Präfix zuweisen.

Schema-/DTD-Referenz hinzufügen

Fügt den Pfad der referenzierten XML-Schema-Datei zum Root-Element der XML-Ausgabe hinzu. Der in dieses Feld eingegebene Schemapfad wird in das Attribut `xsi:schemaLocation` oder (bei Verwendung einer DTD) die `DOCTYPE`-Deklaration der generierten Instanz-Zieldateien geschrieben.

MapForce Professional und Enterprise Edition: Bei der Generierung von Code in XQuery oder C++ wird das Hinzufügen einer DTD-Referenz nicht unterstützt.

Wenn Sie einen Pfad in dieses Feld eingeben, können Sie definieren, wo sich die von der XML-Instanzdatei referenzierte Schemadatei befindet. Damit stellen Sie sicher, dass die Ausgabeinstanz im Mapping-Ordner bei Ausführung des Mappings validiert werden kann. Sie können in dieses Feld eine **`http://`**-Adresse sowie einen absoluten oder relativen Pfad eingeben.

© 2016-2022 Altova GmbH

Wenn Sie diese Option deaktivieren, können Sie die XML-Instanz vom referenzierten XML-Schema oder der DTD entkoppeln. Verwenden Sie diese Option z.B., wenn Sie die erzeugte XML-Ausgabedatei an jemanden senden möchten, der keinen Zugriff auf das zugrunde liegende XML-Schema hat.

- XML-Deklaration schreiben

  Standardmäßig ist die Option aktiviert, d.h. die XML-Deklaration wird in die Ausgabe geschrieben. Diese Funktionalität wird in den folgenden MapForce-Zielsprachen und Ausführungsprozessoren unterstützt.

Zielsprache / Ausführungsprozessor	**Wenn die Ausgabe eine Datei ist**	**Wenn die Ausgabe ein String ist**
Built-in (*Professional und Enterprise Edition*)	Ja	Ja
MapForce Server (*Professional und Enterprise Edition*)	Ja	Ja
XSLT, XQuery	Ja	Nein
Code Generator (C++, C#, Java) (*Professional und Enterprise Edition*)	Ja	Ja

- Werte in Zieltypen konvertieren

  Damit können Sie festlegen, ob (i) im Mapping die XML-Zielschematypen verwendet werden sollen oder (ii) ob alle Daten, die auf die Zielkomponenten gemappt werden, als Stringwerte behandelt werden sollen. Standardmäßig ist diese Einstellung aktiviert. Wenn Sie diese Option deaktivieren, können Sie die exakte Formatierung der Werte beibehalten, z.B. eignet sich die Option, wenn ein Inhalt genau einem Pattern Facet in einem Schema entsprechen muss, demzufolge ein numerischer Wert eine bestimmte Anzahl an Dezimalstellen aufweisen muss. Sie können die Zahl mit Hilfe von Mapping-Funktionen als String im gewünschten Format formatieren und diesen String dann auf die Zielkomponente mappen.

  Wenn Sie diese Option deaktivieren, wird auch die Erkennung ungültiger Werte deaktiviert, z.B. das Schreiben von Buchstaben in numerische Felder.

- Pretty Print für Ausgabe

  Formatiert Ihr XML-Ausgabedokument neu, um eine strukturierte Anzeige des Dokuments zu generieren. Jedes Subelement ist einen Tabstopp vom übergeordneten Element eingerückt.

- Digitale Signatur erstellen (*Enterprise Edition*)

  Gestattet Ihnen, eine digitale Signatur zur XML-Ausgabeinstanzdatei hinzuzufügen. Digitale Signaturen können nur generiert werden, wenn als Transformationssprache Built-In gewählt wurde (siehe auch Digitale Signaturen 143).

Ausgabekodierung

Damit können Sie die folgenden Einstellungen der Output-Instanzdatei definieren:

- Kodierungsname
- Bytefolge

- Ob das Bytefolge-Markierungszeichen (BOM) inkludiert werden soll.

Standardmäßig haben alle neuen Komponenten die in der Option *Standardkodierung für neue Komponenten* definierte Kodierung. Sie können diese Option über **Extras | Optionen** (Abschnitt *Allgemein*) aufrufen.

Wenn mit dem Mapping XSLT 1.0/2.0-Code generiert wird, hat die Aktivierung des Kontrollkästchens *Bytefolge-Markierung* keine Auswirkung**,** da diese Sprachen Bytefolge-Markierungen nicht unterstützen.

StyleVision Power Stylesheet-Datei

Über diese Option können Sie eine Altova StyleVision Stylesheet-Datei auswählen oder erstellen. Mit Hilfe einer solchen Daten können Sie Daten aus der XML-Instanzdatei in die verschiedensten Berichtsformate wie z.B. HTML, RTF und andere transformieren. Siehe auch Verwenden relativer Pfade in einer Komponente 85.

Andere Einstellungen

- Input-Verarbeitungsoptimierungen auf Basis von min/maxOccurs aktivieren

  Diese Option ermöglicht die Sonderbehandlung bei Sequenzen, von denen bekannt ist, dass sie nur genau ein Datenelement enthalten, z.B. erforderliche Attribute oder Child-Elemente mit `minOccurs` und `maxOccurs="1"`. In diesem Fall wird das erste Datenelement der Sequenz extrahiert, anschließend wird das Datenelement direkt als atomarer Wert (und nicht als Sequenz) verarbeitet.

  Wenn die Input-Daten gemäß dem Schema **nicht gültig** sind, könnte eine leere Sequenz in einem Mapping vorkommen, sodass das Mapping mit einer Fehlermeldung abgebrochen wird. Damit auch ein solcher **ungültiger Input** verarbeitet werden kann, deaktivieren Sie dieses Kontrolllkästchen.

- Alle Dateipfade relativ zur MFD-Datei speichern

  Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld **Komponenteneinstellungen** angezeigten Dateipfade relativ zum Ordner, in dem sich die MapForce Design (**`.mfd`**)-Datei befindet. Siehe auch Relative und absolute Pfade 85.

4.1.2 Abgeleitete Typen

In diesem Kapitel wird beschrieben, wie Sie abgeleitete Typen in Mappings verwenden. Abgeleitete Typen sind in der W3C XML Schema-Spezifikation (Abschnitt 2.5.2) definiert. Eine kurze Übersicht über primitive und abgeleitete Typen finden Sie in der Microsoft-Dokumentation. Um abgeleitete Typen in einem Mapping verwenden zu können, müssen Sie das `xsi:type`-Attribut in Ihrer XML-Datei definieren (z.B., `<Address xsi:type="UK-Address">`).

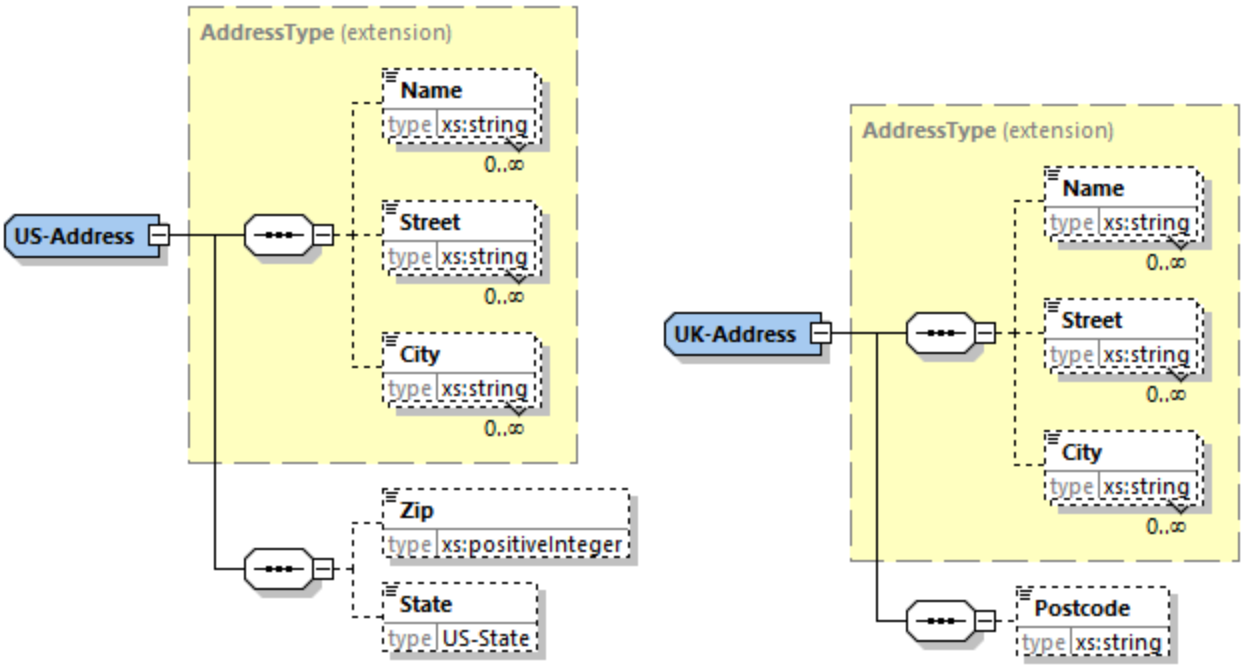
Mögliches Szenario

In diesem Unterabschnitt wird ein mögliches Szenario für die Verwendung eines abgeleiteten Typs beschrieben. Angenommen, wir haben eine Firma mit zwei Niederlassungen, einer in GB und der anderen in den USA. Wir hätten nun gerne zwei Listen (`UKCustomers` und `USCustomers`). Jede davon soll Informationen über die Adresse der jeweiligen Niederlassung und alle Kunden im Zusammenhang mit dieser Niederlassung enthalten.

© 2016-2022 Altova GmbH

Definition von abgeleiteten Typen

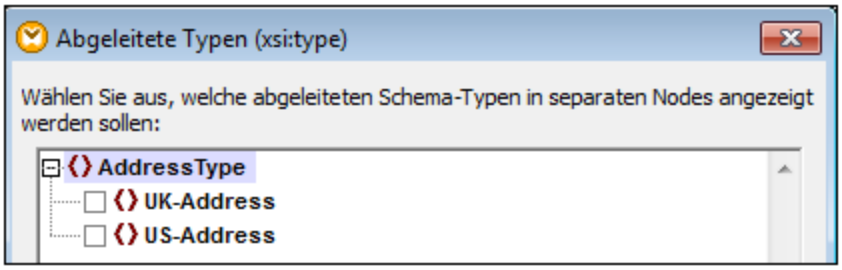
In der Abbildung unten sehen Sie die Definition von abgeleiteten Typen namens `US-Address` und `UK-Address` (*XMLSpy Schema-Ansicht*). Die Elemente `UK-Address` und `UK-Address` haben denselben Basistyp namens `AddressType`, der die Elemente `Name`, `Street` und `City` enthält. Im Element `US-Address` wurde der Basistyp erweitert und enthält `Zip` und `State`, während das Element `UK-Address` den Basistyp und das Element `Postcode` enthält. Wir werden nun zur Veranschaulichung nur das Element `UK-Address` auf die Zieldatei mappen.



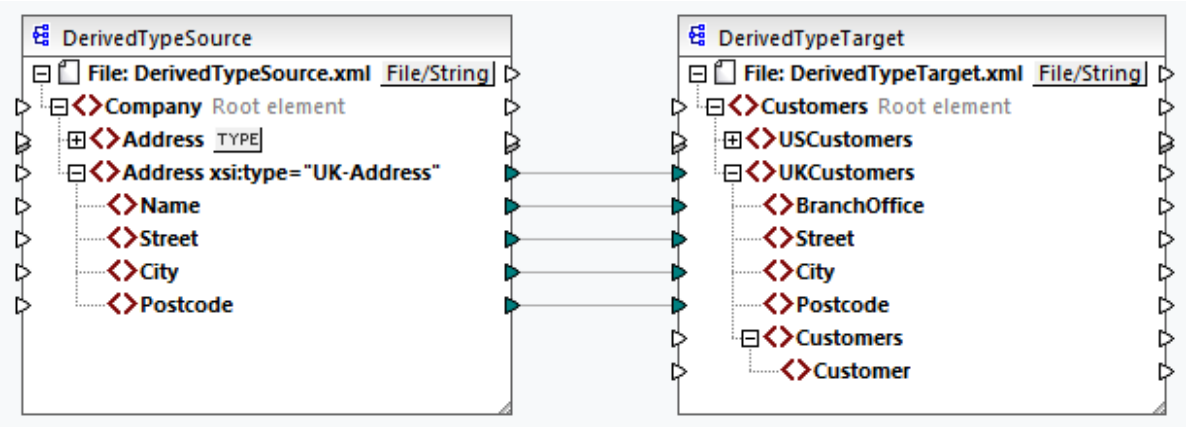
Abgeleiteter Typ in einem Mapping

In der Anleitung unten wird beschrieben, wie Sie Daten vom abgeleiteten Typ aus mappen. Ziel ist es, Informationen über das UK-Büro auf das Element `UKCustomers` zu mappen. Die Beispieldateien stehen im Ordner **`Tutorial`** zur Verfügung.

1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei**, und öffnen Sie die Datei **`DerivedTypeSource.xml`**. Diese XML-Datei basiert auf **`DerivedTypeSource.xsd`**.
2. Fügen Sie die Zieldatei **`DerivedTypeTarget.xsd`** ein. Beachten Sie, dass das Zielschema das Attribut `xsi:type` nicht enthalten muss.
3. Klicken Sie in der Quellkomponente neben dem Element `Address` auf die Schaltfläche TYPE. Diese Schaltfläche gibt an, dass es für dieses Element abgeleitete Typen im Schema gibt.
4. Über das Dialogfeld **Abgeleitete Typen** (*siehe Abbildung unten*) können Sie einen beliebigen abgeleiteten Typ für dieses bestimmte Element auswählen. In unserem Beispielmapping soll nur `UK-Address` gemappt werden.



5. Sobald Sie das Kontrollkästchen neben dem abgeleiteten `UK-Address`-Typ auswählen, steht in der Komponente ein neues Element namens `Address xsi:type="UK-Address"` zur Verfügung.
6. Verbinden Sie nun die Nodes, wie im Mapping unten gezeigt.



Ausgabe
Wenn Sie auf das **Ausgabe**-Fenster klicken, sehen Sie das folgende Resultat:

```
<UKCustomers>
   <BranchOffice>Sleuth Corp. UK</BranchOffice>
   <Street>222 Baker St</Street>
   <City>London</City>
   <Postcode>NW1 6XE</Postcode>
</UKCustomers>
```

Das Beispielmapping ist unter dem Namen **`Tutorial\DerivedType.mfd`** gespeichert. Sie können auch eine weitere XML-Quellkomponente hinzufügen, die Informationen über die Kunden in UK enthält, und diese Daten auf den Node `Customers` in der Zielkomponente mappen. Dadurch enthält das Element `UKCustomers` in der Folge Informationen über die UK-Adresse und alle mit dieser Niederlassung verknüpften Kunden.

4.1.3 NULL-Werte

In diesem Abschnitt wird erläutert, wie MapForce NULL-Werte in Quell- und Zielkomponenten behandelt. Um das Attribut `xsi:nil="true"` in Ihrer XML-Datei verwenden zu können, müssen Sie in Ihrer Schema-Datei für das/die relevante(n) Element(e) das Attribut `nillable="true"` definieren. Nähere Informationen über die Attribute `nillable` und `xsi:nil` finden Sie in der W3C-Spezifikation. Beachten Sie, dass das Attribut `xsi:nil` im **Mapping**-Fenster in der Struktur einer Komponente nicht sichtbar ist.

 © 2016-2022 Altova GmbH

In den folgenden Unterabschnitten werden einige mögliche Szenarien beim Mappen von NULL-Werten beschrieben.

NULL-Werte in XML-Komponenten

In diesem Unterabschnitt werden einige mögliche Szenarien beim Mappen von Elementen mit einem `xsi:nil="true"`-Attribut beschrieben.

Nur das Quellelement hat xsi:nil="true"/Sowohl Quell- als auch Zielelementen haben xsi:nil="true"
Für dieses Szenario gelten die folgenden Bedingungen:

- Die Verbindung ist zielorientiert[93].
- Das Quellelement hat ein `xsi:nil="true"`-Attribut. Das dazugehörige Zielelement hat dieses Attribut nicht.
- Alternativ dazu können sowohl Quell- als auch Zielelemente `xsi:nil="true"`-Attribute haben.
- Die `nillable="true"`-Attribute müssen im Quell- und im Zielschema definiert sein.
- Quell- und Zielelement haben den Typ simpleType.

In diesem Fall erhält das Zielelement in der Ausgabedatei das Attribut `xsi:nil="true"` , wie in der Beispiel-Ausgabedatei unten gezeigt (*gelb markiert*).

```
<book id="7">
	<author>Edgar Allan Poe</author>
	<title>The Murders in the Rue Morgue</title>
	<category xsi:nil="true"/>
	<year>1841</year>
	<OrderID id="213"/>
</book>
```

Anmerkung: Wenn das Attribut `nillable="true"` im Zielschema nicht definiert ist, bleibt das entsprechende Zielelement in der Ausgabe leer.

Nur das Zielelement hat xsi:nil="true"
Für dieses Szenario gelten die folgenden Bedingungen:

- Die Verbindung ist zielorientiert[93].
- Das Quellelement hat kein `xsi:nil="true"`-Attribut.
- Das dazugehörige Zielelement hat ein `xsi:nil="true"`-Attribut.
- Quell- und Zielelement können den Typ simpleType oder complexType haben.

In diesem Fall überschreibt das Quellelement das Zielelement, das das Attribut `xsi:nil="true"` enthält. Unten sehen Sie ein Beispiel für eine Ausgabedatei. Das Element `<genre>` enthält im Zielelement das Attribut `xsi:nil="true"`. Dieses Element wurde zur Mapping-Laufzeit jedoch überschrieben. Daher hat das Element `<genre>` (*gelb markiert*) in der Ausgabe `Fiction`.

```
<publication>
	<id>1</id>
	<author>Mark Twain</author>
	<title>The Adventures of Tom Sawyer</title>
	<genre>Fiction</genre>
	<year>1876</year>
	<OrderID id="124"/>
</publication>
```

Das Quellelement vom ComplexType/beide Elemente vom ComplexType haben xsi:nil="true"
Für dieses Szenario gelten die folgenden Bedingungen:

- Die Verbindung ist zielorientiert[93].
- Das Quellelement hat den Typ complexType. Das Quellelement hat in unserem Beispiel ein Attribut `id="213"` und ein Attribut `xsi:nil="true"`. Das dazugehörige Zielelement hat ebenfalls den Typ complexType und ein Attribut `id="124"`, hat aber kein `xsi:nil="true"`-Attribut.
- Alternativ dazu können das Quell- und das Zielelement, beide vom Typ complexType, `xsi:nil="true"`-Attribute haben.

In diesem Fall überschreibt das Quellelement das Zielelement (*unten gelb markiert*). Das Attribut `xsi:nil="true"` wird jedoch nicht automatisch in die Ausgabe geschrieben. Damit das Attribut `xsi:nil="true"` in der Ausgabedatei im Zielelement aufscheint, verwenden Sie eine Alles kopieren[97]-Verbindung.

```
<book id="7">
	<author>Edgar Allan Poe</author>
	<title>The Murders in the Rue Morgue</title>
	<year>1841</year>
	<OrderID id="213"/>
</book>
```

Nützliche Funktionen

Mit Hilfe der folgenden Funktionen können Sie NULL-Werte überprüfen, ersetzen und zuweisen:

- **is-xsi-nil**[1069]: Damit können Sie explizit überprüfen, ob in einem Quellelement das Attribut `xsi:nil` auf `true` gesetzt ist.
- **substitute-missing**[1103]: Ersetzt einen NULL-Wert im Quellelement durch etwas Bestimmtes.
- **set-xsi-nil**[1072]: Weist einem Zielelement das Attribut `xsi:nil="true"` zu. Dies funktioniert bei Zieldatenelementen vom Typ "simpleType" und "complexType".
- **substitute-missing-with-xsi-nil**[1073]: Wenn Inhalt vorhanden ist, wird dieser in das Zielelement geschrieben; wenn Werte fehlen, wird das Zielelement in der Ausgabe mit dieser Funktion mit dem Attribut `xsi:nil="true"` versehen.
- Wenn Sie die **exists**[1079]-Funktion mit einem Quellelement mit einem NULL-Wert verbinden, wird `true` zurückgegeben, selbst wenn das Element keinen Inhalt hat.

Beachten Sie, dass Funktionen, die `xsi:nil` generieren, nicht über Funktionen oder Komponenten übergeben werden können, die nur an Werten operieren (wie z.B. die **if-else**-Funktion).

NULL-Werte in Datenbankkomponenten

In diesem Unterabschnitt wird erläutert, wie NULL-Werte in Datenbankkomponenten behandelt werden.

Mappen von NULL-Datenbankfeldern auf NULL-Elemente
Zielelemente, die NULL-Werte aus Datenbankfeldern erhalten, werden in der Ausgabe nicht automatisch erstellt. Damit solche Elemente in der Ausgabe angezeigt werden, müssen Sie zu den entsprechenden Elementen in der Schema-Datei (i) `nillable="true"`-Attribute hinzufügen und (ii) im Mapping die Funktion **substitute-missing-with-xsi-nil**[1073] verwenden. Im Beispiel unten sehen Sie, wie NULL-Werte in Mappings mit einer Quelldatenbankkomponente behandelt werden.

© 2016-2022 Altova GmbH

Applications-Tabelle im DB-Abfrage-Fenster
Das Beispielmapping befindet sich unter dem folgenden Pfad: **`Tutorial\DBNullToXML.mfd`**. Wir haben für unser Beispiel nur eine einzige Tabelle (`Application`) aus der Datenbank `Accounts` (*siehe unten*) ausgewählt.

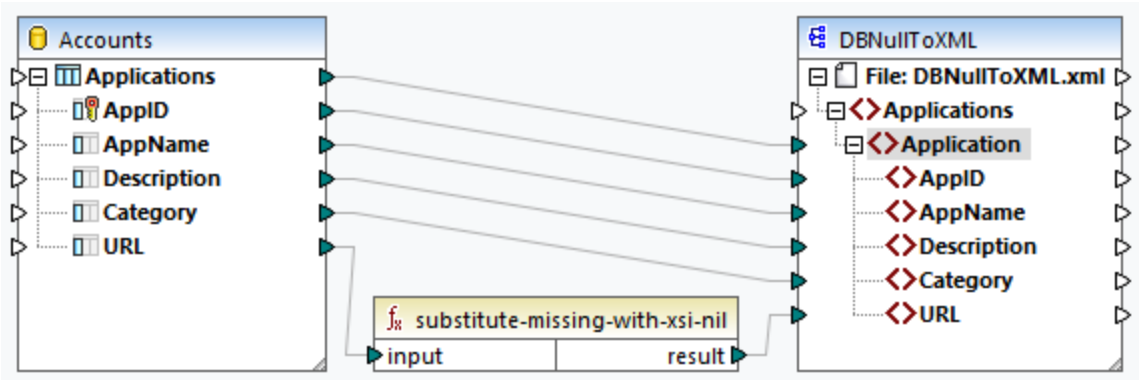
	AppID	AppName	Description	Category	URL
1	1	Altova MapForce	Best data mapping tool!	IDE	https://www.altova.com/mapforce
2	2	Notepad	*[NULL]*	*[NULL]*	*[NULL]*

Um die Tabelle `Application` zu sehen, gehen Sie folgendermaßen vor:

- Öffnen Sie das Fenster **DB-Abfrage**.
- Wählen Sie die Datenbank `Accounts` aus, um ihre Struktur im Datenbank Browser zu sehen.
- Klicken Sie mit der rechten Maustaste auf die Tabelle `Application` und wählen Sie den Befehl **In SQL Editor anzeigen | SELECT**.
- Klicken Sie auf die Schaltfläche  (**Abfrage ausführen**). Daraufhin wird die Tabelle `Application` auf dem Register **Ergebnisse** angezeigt.

Nähere Informationen über Datenbankabfragen finden Sie unter Anzeigen und Abfragen von Datenbanken 388.

Mapping
In der Tabelle `Application` oben sehen Sie, dass der zweite Datensatz in den Feldern `Description`, `Category` und `URL` NULL-Werte hat. Zur Veranschaulichung werden wir fast alle Spalten direkt auf die entsprechenden Zielelemente mappen. Für die Spalte `URL` verwenden wir die Funktion **`substitute-missing-with-xsi-nil`**, damit der NULL-Wert im Zielelement ein `xsi:nil="true"`-Attribut erhält (*siehe Mapping unten*).



Ausgabe
In der Ausgabedatei unten sehen Sie, dass der erste Datensatz aus der Tabelle vollständig in die Ausgabe geschrieben wurde, während der zweite Datensatz nur teilweise dorthin geschrieben wurde. Außer im Element `URL` fehlen die NULL-Datenbankwerte in der Ausgabe. Da das Element `URL` in der Schema-Datei das Attribut `nillable="true"` hat und wir die Funktion **`substitute-missing-with-xsi-nil`** verwenden, hat das Element `URL` nun in der Ausgabe das Attribut `xsi:nil="true"` (*gelb markiert*).

```
<Application>
	<AppID>1</AppID>
	<AppName>Altova MapForce</AppName>
```

```
        <Description>Best data mapping tool!</Description>
        <Category>IDE</Category>
        <URL>https://www.altova.com/mapforce</URL>
</Application>
<Application>
        <AppID>2</AppID>
        <AppName>Notepad</AppName>
        <URL xsi:nil="true"/>
</Application>
```

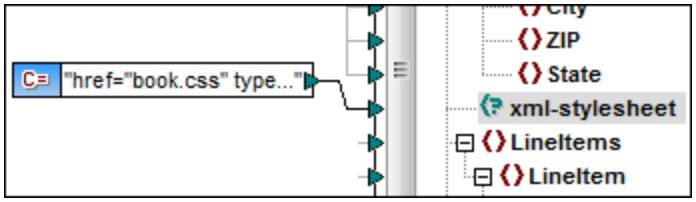
Mappen von NULL-Element auf NULL-Datenbankfelder
Wenn Sie ein XML-NULL-Element auf eine Datenbankspalte mappen, schreibt MapForce einen NULL-Wert in die entsprechende Datenbankspalte. Wenn Sie ein Datenbankfeld auf NULL setzen möchten, können Sie dies auch mit Hilfe der Funktion **set-null** 1123 tun. Nähere Informationen über Funktionen im Zusammenhang mit Datenbanken finden Sie in der DB-Bibliothek 1122.

4.1.4 Kommentare und Processing Instructions

Kommentare und Processing Instructions sind in der W3C-Spezifikation definiert. In diesem Kapitel wird erläutert, wie Sie Kommentare und Processing Instructions in XML-Zielkomponenten einfügen. Beachten Sie, dass Kommentare und Processing Instructions nur Input-Verbindungen haben. Kommentare und Processing Instructions können nicht für Nodes definiert werden, die Teil einer "Alles kopieren" 97-Verbindung sind.

Einfügen eines Kommentars/einer Processing Instruction
Um eine Processing Instruction oder einen Kommentar einzufügen, gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf ein Element in der Komponente und wählen Sie **Kommentar/Processing Instruction davor/danach einfügen**. Wenn Sie eine Processing Instruction einfügen, müssen Sie auch ihren Namen eingeben. Im Beispiel unten wurde nach dem Element `State` eine Processing Instruction namens `xml-stylesheet` eingefügt.



3. Den Wert eines Kommentars oder einer Processing Instruction können Sie mit Hilfe einer Konstanten angeben, z.B. (*siehe Abbildung oben*).

Anmerkung: Sie können vor oder nach jedem Element in der Zielkomponente mehrere Processing Instructions hinzufügen.

Anmerkung: Vor oder nach einem einzigen Ziel-Node kann nur ein einziger Kommentar hinzugefügt werden. Um mehrere Kommentare zu erstellen, verwenden Sie die Funktion Duplikat einfügen 84.

Löschen eines Kommentars/einer Processing Instruction
Um einen Kommentar/eine Processing Instruction zu löschen, klicken Sie mit der rechten Maustaste auf den entsprechenden Node, wählen Sie **Kommentar/Processing Instruction** und anschließend aus dem Untermenü den Befehl **Kommentar/Processing Instruction löschen**.

© 2016-2022 Altova GmbH

4.1.5 CDATA-Abschnitte

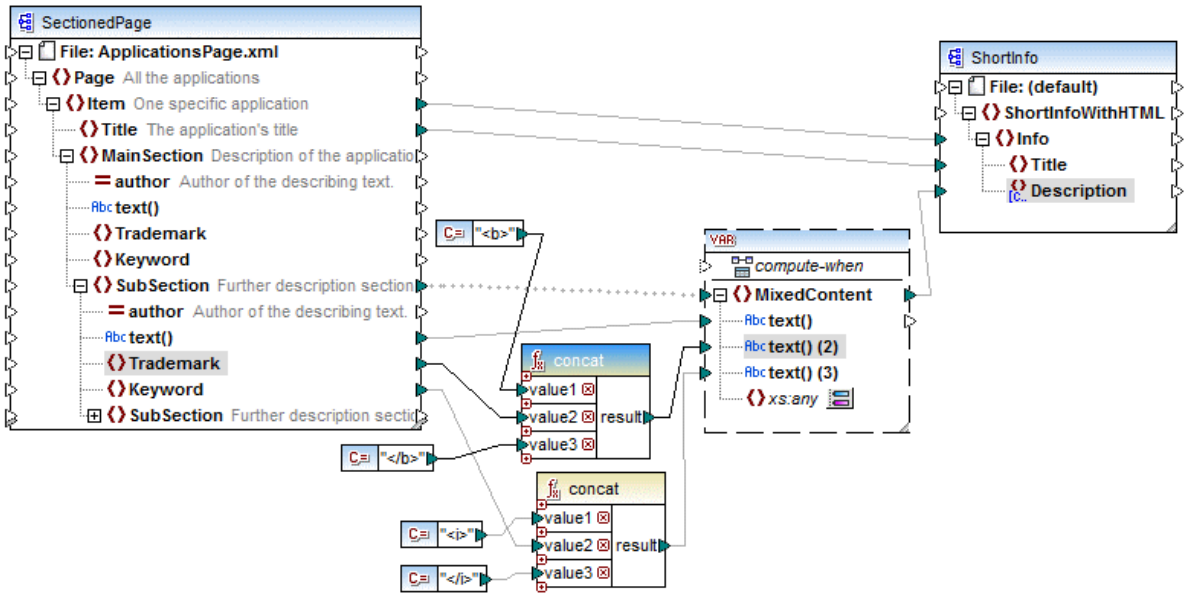
CDATA-Abschnitte dienen dazu, Abschnitte eines Dokuments, die normalerweise als Markup interpretiert würden, als Zeichendaten darzustellen. Nähere Informationen zu CDATA-Abschnitten finden Sie in der W3C-Spezifikation. Die folgenden Ziel-Nodes können CDATA-Abschnitte erhalten: XML-Daten, in Datenbankfelder eingebettete XML-Daten und XML-Child-Elemente von typisierten Dimensions in einer XBRL-Zielkomponente. CDATA-Abschnitte können auch in duplizierten Nodes und `xsi:type`-Nodes definiert werden.

Um einen CDATA-Abschnitt zu erstellen, klicken Sie mit der rechten Maustaste auf den entsprechenden Ziel-Node und wählen Sie die Befehl **Inhalt als CDATA-Abschnitt schreiben**. Daraufhin erscheint eine Meldung, in der Sie gewarnt werden, dass die Input-Daten das schließende CDATA-Abschnittstrennzeichen `]]>` nicht enthalten dürfen. Das Symbol `[C..` wird unterhalb des Element-Tags angezeigt und bedeutet, dass dieser Node nun als CDATA-Abschnitt definiert ist.

Beispiel

Im Beispiel unten sehen Sie ein Szenario, in dem sich ein CDATA-Abschnitt als nützlich erweisen könnte. Das Beispielmapping **`MapForceExamples\HTMLinCDATA.mfd`** (*siehe Abbildung unten*) hat die folgenden Aspekte:

- Das Element `SubSection` hat gemischten Inhalt. Nähere Informationen zu Nodes mit gemischtem Inhalt finden Sie unter Quellorientierte Verbindungen 93.
- Mit Hilfe der Funktion **`concat`** erhält der Inhalt des Elements `Trademark` die Tags `<b></b>`.
- Der Inhalt des Elements `Keyword` erhält die Tags `<i></i>`.
- Die Daten mit den neuen Tags werden in derselben Reihenfolge wie im Quelldokument an die duplizierten `text()`-Nodes übergeben.
- Die Ausgabe des `MixedContent` Node wird anschließend an den `Description` Node in der Zielkomponente `ShortInfo` übergeben. Der `Description`-Node wurde als CDATA-Abschnitt definiert.



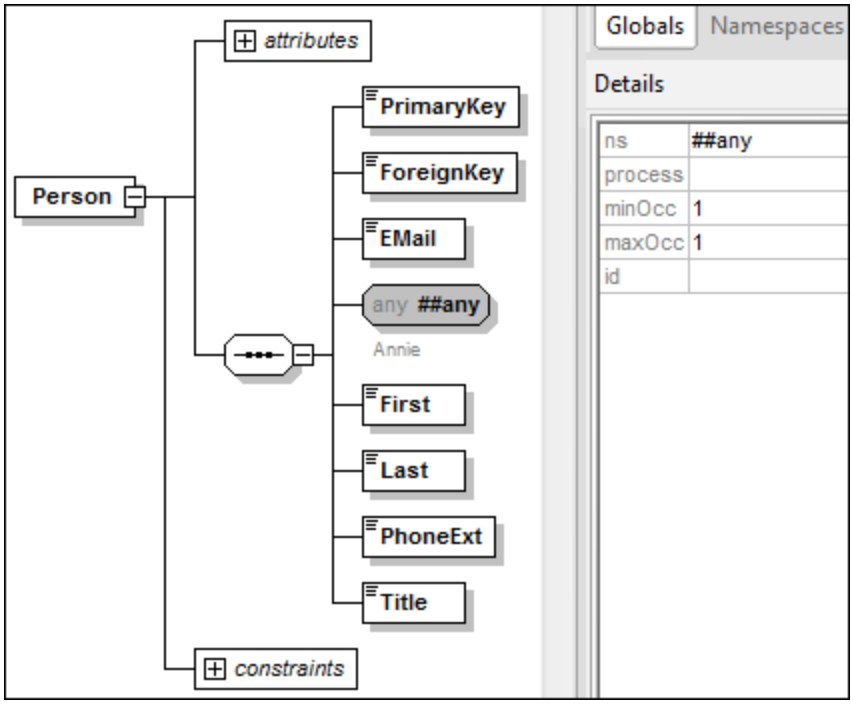
Ausgabe
Klicken Sie auf das Fenster **Ausgabe**, um den CDATA-Abschnitt im Node `Description` zu sehen (*Abbildung unten*).

```
7   <Info>
8     <Title>MapForce</Title>
9     <Description><![CDATA[Altova <b>MapForce</b> 2014 Enterprise Edition is the premier <i>XML</i>
      / <i>database</i> / <i>flat file</i> / <i>EDI</i> data mapping tool that auto-generates mapping code in
      <i>XSLT</i> 1.0/2.0, <i>XQuery</i>, <i>Java</i>, <i>C++</i> and <i>C#</i>. It is the definitive tool for
      data integration and information leverage.]]></Description>
10  </Info>
```

4.1.6 Wildcards - xs:any / xs:anyAttribute

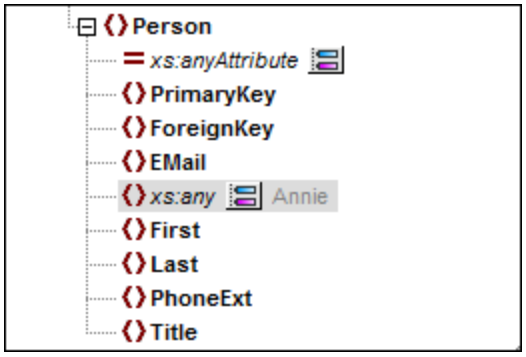
In diesem Kapitel wird beschrieben, wie Sie Wildcards in Mappings behandeln können. Mit Hilfe der Wildcards `xs:any` und `xs:anyAttribute` können Sie in Ihrer Schema-Datei definierte any-Elemente/Attribute verwenden. Nähere Informationen zu Wildcards finden Sie in der W3C-Spezifikation.

Wildcards in der Schema-Definition
In der Abbildung unten wurde ein `xs:any`-Element als Child-Element des `Person`-Elements definiert (*Schema-Ansicht in Altova XMLSpy*).



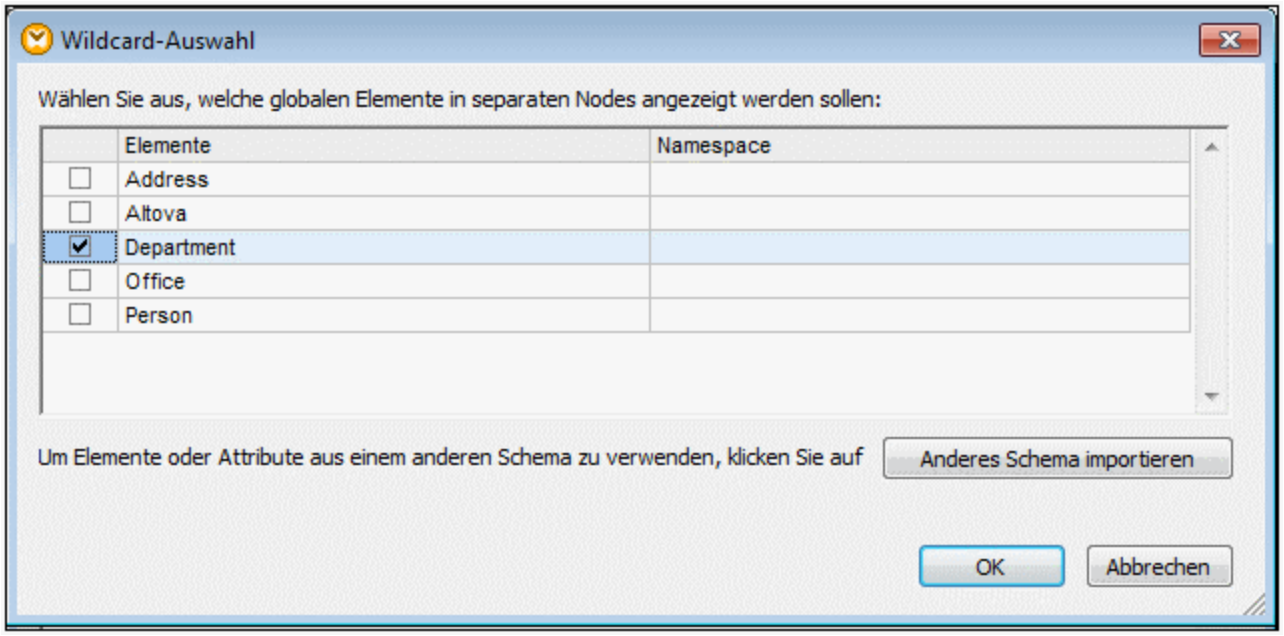
Wildcards in MapForce
Wenn für ein Element und/oder Attribut eine Wildcard definiert wurde, wird neben diesem Node eine -Schaltfläche (**Auswahl ändern**) angezeigt (*siehe Abbildung unten*).

© 2016-2022 Altova GmbH

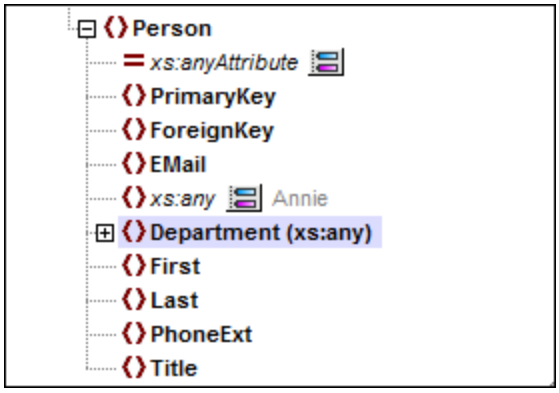


Wildcard-Auswahl

Wir möchten nun ein weiteres Element als separaten Node hinzufügen. Klicken Sie auf die Schaltfläche , um die Liste der Elemente, die zur Struktur hinzugefügt werden können, zu sehen. Beachten Sie, dass im Dialogfeld **Wildcard-Auswahl** nur Elemente und Attribute angezeigt werden, die in Ihrem Schema global deklariert wurden (*siehe Abbildung unten*).



Für dieses Beispiel haben wir `Department` ausgewählt. Beachten Sie, dass Wildcard-Elemente und -Attribute nach dem Node mit der Schaltfläche eingefügt werden. Unsere Komponente sieht nun folgendermaßen aus:



Sie können nun zwischen diesen Nodes wie gewöhnlich Mapping-Verbindungen erstellen. Wildcard-Elemente und -Attribute werden in einer Komponente mit `(xs:any)` bzw. (`xs:anyAttribute`) gekennzeichnet (*siehe Abbildung oben*).

Entfernen von Wildcards
Um einen Wildcard-Node zu entfernen, klicken Sie auf die und deaktivieren Sie das entsprechende Kontrollkästchen im Dialogfeld **Wildcard-Auswahl**.

Elemente/Attribute aus einem anderen Schema

Über das Dialogfeld **Wildcard-Auswahl** (*siehe oben*) können Elemente/Attribute aus einem anderen Schema verwendet werden. Wenn Sie auf die Schaltfläche **Anderes Schema importieren** klicken, haben Sie die folgenden Optionen: (i) Import einer Schema-Datei und (ii) Generierung eines Wrapper-Schemas (*siehe Beschreibung unten*).

Import eines Schemas
Mit der Option **Schema importieren** wird das externe Schema in das aktuelle, der Komponente zugewiesene Schema importiert. Beachten Sie, dass mit dieser Option das vorhandene Schema auf der Festplatte ersetzt wird. Wenn es sich beim aktuellen Schema nicht um ein entferntes über eine URL geöffnetes Schema (siehe Hinzufügen von Komponenten über eine URL 80) handelt und das Schema nicht von der lokalen Festplatte stammt, kann es nicht geändert werden. Verwenden Sie in diesem Fall die Option **Wrapper-Schema generieren**.

Generieren eines Wrapper-Schemas
Mit der Option **Wrapper-Schema generieren** wird eine neue als *Wrapper-Schema* bezeichnete Schema-Datei erstellt. Der Vorteil bei dieser Option ist, dass das bestehende Schema der Komponente nicht geändert wird. Stattdessen wird ein neues Schema erstellt, das sowohl das vorhandene Schema als auch das importierte Schema enthält. Wenn Sie diese Option auswählen, werden Sie gefragt, wo das Wrapper-Schema gespeichert werden soll. Standardmäßig hat das Wrapper-Schema das Format **`dateiname-wrapper.xsd`**.

Nachdem Sie das Wrapper-Schema gespeichert haben, wird es der Komponente standardmäßig automatisch zugewiesen. Außerdem werden Sie gefragt, ob Sie den Schemapfad anpassen möchten, um das vorherige Hauptschema referenzieren zu können. Klicken Sie auf **Ja**, um zum vorherigen Schema zurückzuwechseln oder auf **Nein**, damit das neu erstellte Wrapper-Schema der Komponente zugewiesen wird.

© 2016-2022 Altova GmbH

Wildcards oder dynamische Node-Namen

Es gibt Fälle, in denen eine Instanz zu viele Elemente und/oder Attribute enthält, als dass diese im Schema deklariert werden können. Betrachten Sie die folgende Beispieldatei:

```
<?xml version="1.0" encoding="UTF-8"?>
<message>
   <line1>1</line1>
   <line2>2</line2>
   <line3>3</line3>
   ...............
   <line999></line999>
</message>
```

In solchen Fällen wird empfohlen, anstelle von Wildcards dynamische Node-Namen zu verwenden. Nähere Informationen dazu finden Sie unter Mappen von Node-Namen[1309].

4.1.7 Benutzerdefinierte Namespaces

Wenn mit einem Mapping eine XML-Ausgabe erzeugt wird, so übernimmt MapForce den Namespace (oder eine Gruppe von Namespaces) für die einzelnen Elemente und Attribute automatisch aus dem Ziel-Schema. Dies ist das Standardverhalten, das für die meisten Szenarien, in denen eine XML-Ausgabe generiert werden soll, geeignet ist. In einigen Fällen müssen Sie den Namespace eines Elements jedoch direkt über das Mapping manuell deklarieren.

Die Deklaration benutzerdefinierter Namespaces ist nur bei XML-Zielkomponenten sinnvoll und gilt nur für Elemente. Der Befehl **Namespace hinzufügen** steht für Attribute, Wildcard Nodes und für Nodes, die Daten aus einer "Alles kopieren"-Verbindung[97] erhalten, nicht zur Verfügung.

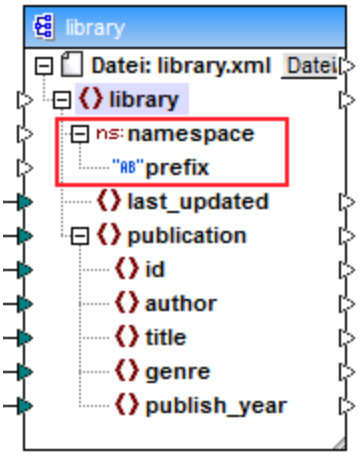
Um zu verstehen, wie benutzerdefinierte Namespaces funktionieren, folgen Sie der Anleitung in Unterabschnitt weiter unten.

Manuelle Deklaration eines Namespace

Sie benötigen für dieses Beispiel das folgende Mapping: **`BasicTutorials\Tut1-SchemaToSchema.mfd`**.

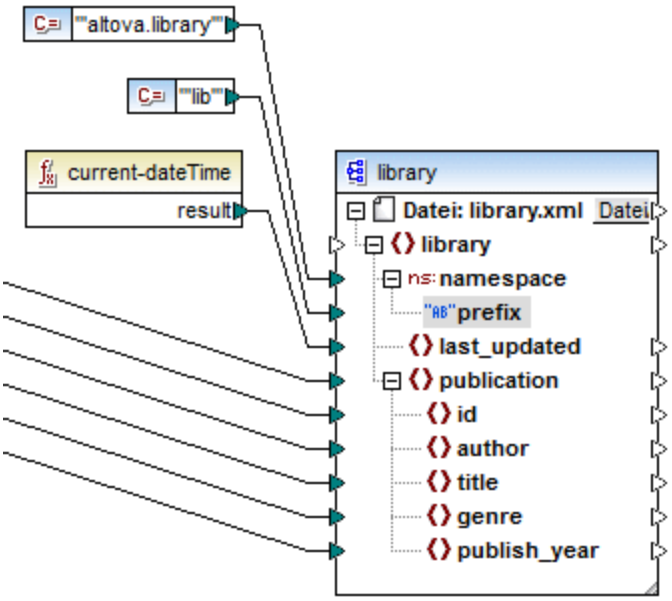
Hinzufügen eines Namespace

Öffnen Sie das Mapping, kllicken Sie in der Komponente `BooksOutput` mit der rechten Maustaste auf den Node `library` und wählen Sie im Kontextmenü den Befehl **Namespace hinzufügen**. Unter dem Element `library` stehen nun zwei neue Nodes zur Verfügung: `namespace` und `prefix` (*siehe Abbildung unten*).



Bereitstellen von Namespace-Werten

Im nächsten Schritt müssen nun Werte für die Nodes `namespace` und `prefix` angegeben werden. Zu diesem Zweck verwenden wir zwei Konstanten mit den folgenden String-Werten: `altova.library` und `lib` (*siehe Abbildung unten*).



Anmerkung: Sowohl der `namespace`- aus auch der `prefix`-Input-Konnektor muss gemappt werden, auch wenn Sie dafür leere Werte angeben.

Ausgabe

In der generieren Ausgabe wird zum Element ein `xmlns:<prefix>="<namespace>"`-Attribut hinzugefügt, wobei `<prefix>` und `<namespace>` Werte sind, die vom Mapping geliefert werden. Die Ausgabe sieht nun folgendermaßen aus (*Beachten Sie den markierten Teil*):

© 2016-2022 Altova GmbH

```
<?xml version="1.0" encoding="UTF-8"?>
<library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:lib="altova.library" xsi:noNamespaceSchemaLocation="Library.xsd">
...
```

Sie können für dasselbe Element auch mehrere Namespaces deklarieren, falls nötig. Klicken Sie dazu mit der rechten Maustaste auf den Node und wählen Sie im Kontextmenü den Befehl **Namespace hinzufügen**. Es steht nun ein neues Paar an "namespace" und "prefix"-Nodes zur Verfügung, mit dem Sie neue Präfix- und Namespace-Werte verbinden können.

Deklarieren eines Standard-Namespace
Wenn Sie einen Standard-Namespace deklarieren möchten, mappen Sie einen leeren String-Wert auf `prefix`. Die Ausgabe würden dann folgendermaßen aussehen (*Beachten Sie den markierten Teil*):

```
<?xml version="1.0" encoding="UTF-8"?>
<library xmlns="altova.library" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="Library.xsd">
...
```

Wenn Sie Präfixe für Attributnamen erstellen müssen, z.B. `<number prod:id="prod557">557</number>`, können Sie dies durch dynamischen Zugriff auf die Attribute eines Node (siehe Mappen von Node-Namen. 1309) oder durch Bearbeiten des Schemas tun, so dass es ein `prod:id`-Attribut für `<number>` hat.

Entfernen eines Namespace
Um eine zuvor hinzugefügte Namespace-Deklaration zu entfernen, klicken Sie mit der rechten Maustaste auf den Node `ns:namespace` und wählen Sie im Kontextmenü den Befehl **Namespace entfernen**.

4.1.8 Digitale Signaturen

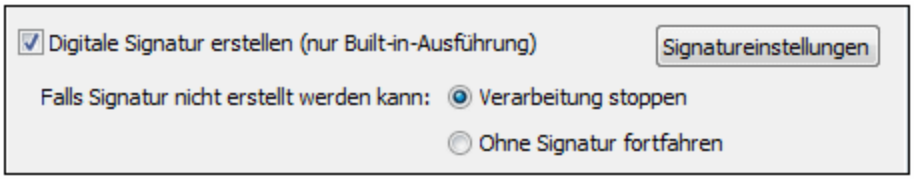
Digitale Signaturen dienen dazu, ein XML-Dokument mit einem verschlüsselten Code digital zu signieren. Anhand dieses Codes kann überprüft werden, ob das XML-Dokument nach der Signierung geändert wurde. Nähere Informationen zu XML-Signaturen finden Sie in der W3C-Spezifikation. MapForce unterstützt nur Zertifikate vom Typ RSA-SHA1 und DSA-SHA1.

MapForce unterstützt die Erstellung digitaler XML-Signaturen für XML- und XBRL-Ausgabedateien. Digitale Signaturen werden nur in Built-in und nur in der Vorschau unterstützt. Digitale Signaturen können als letztes Child-Element des Root-Elements in das Ausgabedokument eingebettet (*Enveloped*) oder in einer separaten Datei gespeichert werden (*Detached*). Nähere Informationen dazu finden Sie unter XML-Signatureinstellungen 144. Einen Vergleich zwischen den Optionen "Detached" und "Enveloped" finden Sie unter Detached" oder "Enveloped" 147.

Anmerkung: MapForce Server unterstützt digitale Signaturen nicht.

Aktivieren der Signaturgenerierung
Um digitale Signaturen generieren zu können, gehen Sie folgendermaßen vor:

1. Öffnen Sie das Dialogfeld **Komponenteneinstellungen** der Output-Komponente: Doppelklicken Sie auf die Komponentenüberschrift und wählen Sie **Komponente | Eigenschaften**.
2. Aktivieren Sie das Kontrollkästchen *Digitale Signatur erstellen* (*siehe Abbildung unten*).

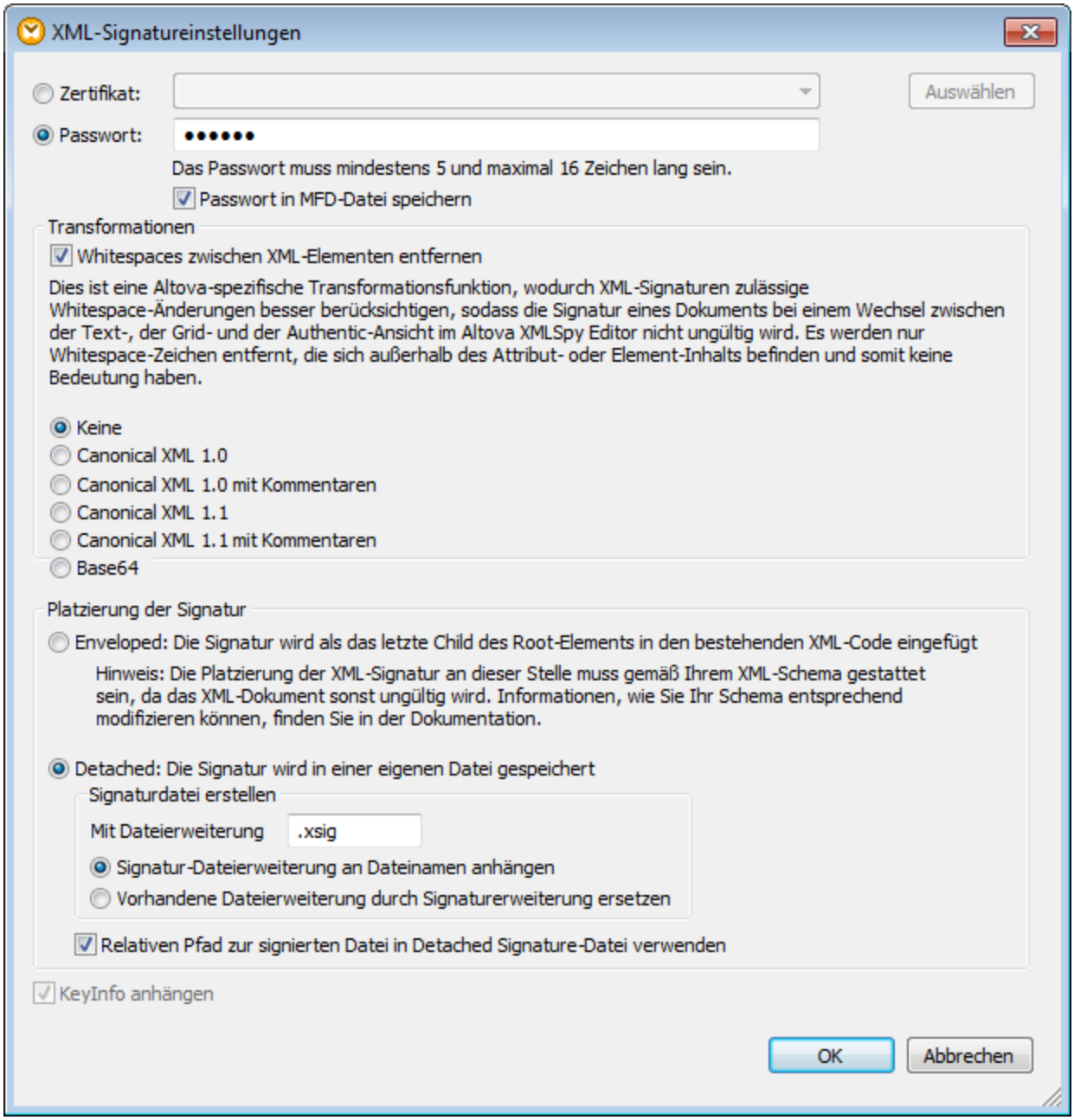


3. Daraufhin erscheint das Dialogfeld **XML-Signatureinstellungen**, in dem Sie die erforderlichen Einstellungen definieren können.

4.1.8.1 XML-Signatureinstellungen

Die Signatureinstellungen (*siehe Abbildung unten*) werden für jede Komponente einzeln gespeichert und können über das Dialogfeld Komponenteneinstellungen[126] aufgerufen werden. Alle Signatureinstellungen werden beim Speichern in der MFD-Datei gespeichert.

© 2016-2022 Altova GmbH



Weiter unten finden Sie eine Beschreibung der verfügbaren Einstellungen.

- Authentifizierungsmethode: Zertifikat oder Passwort

  Die Signatur kann auf einem Zertifikat oder einem Passwort basieren.

  - *Zertifikat:* Wenn Sie ein Zertifikat verwenden möchten, muss dieses einen privaten Schlüssel (private key) haben und sich in einem Zertifikatspeicher befinden, auf den Zugriff besteht. Die Signatur wird anhand des privaten Schlüssels des Zertifikats generiert. Zur Überprüfung der Signatur muss Zugriff auf das Zertifikat (oder eine Public-Key-Version davon) bestehen. Der

öffentliche Schlüssel (public key) des Zertifikats dient zur Überprüfung der Signatur. Klicken Sie zur Auswahl des Private-Public-Key-Zertifikats auf die Schaltfläche **Auswählen** und navigieren Sie zum gewünschten Zertifikat.

- *Passwort:* Geben Sie ein Passwort mit einer Länge von 5 bis 16 Zeichen ein. Anhand dieses Passworts wird die Signatur erstellt und dieses Passwort wird benötigt, um die Signatur zu überprüfen. Nur wenn diese Anforderung erfüllt wird, wird die Schaltfläche **OK** des Dialogfelds aktiv.

- *Passwort in MFD-Datei speichern:* Wenn diese Option aktiv ist, wird das in das Feld *Passwort* eingegebene Passwort verschlüsselt, so dass es für Menschen nicht lesbar ist. Beachten Sie, dass jeder Benutzer, der Zugriff auf die MFD-Datei hat, mit Hilfe dieses Passworts Signaturen erstellen kann.

Transformationen

Die XML-Daten werden transformiert und das Ergebnis der Transformation wird zur Erstellung der Signatur verwendet. Sie können den Algorithmus, der auf die XML-Daten der Datei (den `SignedInfo`-Inhalt) anzuwenden ist, definieren, bevor die Signatur berechnet wird. Es stehen die folgenden Optionen zur Verfügung:

- *Keine:* Es wird keine Transformation durchgeführt und die XML-Daten aus der auf der Festplatte gespeicherten Binärdatei werden direkt an die Signaturüberprüfung übergeben. Alle späteren Änderungen an den Daten führen dazu, dass die Signatur nicht verifiziert werden kann. Wenn allerdings das Kontrollkästchen *Whitespaces zwischen XML-Elementen entfernen* aktiviert ist, so werden alle Whitespace-Zeichen entfernt und Änderungen an Whitespace-Zeichen werden ignoriert.

- *Canonical XML mit oder ohne Kommentare:* Wenn Kommentare bei der Berechnung der Signatur einbezogen werden, so führt jede Änderung an den Kommentaren in den XML-Daten zu einem Fehler bei der Überprüfung der Signatur. Andernfalls dürfen Kommentare nach der Signierung des XML-Dokuments geändert oder hinzugefügt werden, ohne dass die Signatur dadurch ungültig wird.

  Beachten Sie, dass die Option *Canonical XML* mit Kommentaren nur für die Platzierung mit der Option *Detached* zur Verfügung steht.

- *Base64:* Das Root-Element des XML-Dokuments wird als Base64-kodiert betrachtet. Wenn das Root-Element nicht in Base64 kodiert ist, wird ein Fehler zurückgegeben oder das Element je nach Elementart als leeres gelesen.

Der Unterschied zwischen der Option *Keine* und *Canonical XML* ist, dass bei der zweiten Option ein XML-Datenstrom erzeugt wird, in dem einige Unterschiede wie z.B. die Reihenfolge der Attribute normalisiert wird. Beachten Sie allerdings, dass standardmäßig eine Kanonisierung erfolgt, wenn die Signatur (mittels enveloped) eingebettet wird. Die XML-Daten werden unverändert verwendet (d.h. es gibt keine Transformation), wenn eine der folgenden Bedingungen zutrifft: (i) die Signatur ist separat (detached), (ii) es wurde *Keine* ausgewählt und (iii) das Kontrollkästchen *Whitespaces entfernen* ist nicht ausgewählt.

Platzierung der Signatur

Es stehen die folgenden Signaturplatzierungsoptionen zur Verfügung:

© 2016-2022 Altova GmbH

- *Enveloped:* Das Signaturelement wird als das letzte Child-Element des Root (Dokument)-Elements erstellt. Damit die XML-Datei gültig ist, muss das mit der Datei verknüpfte XML-Schema die Elemente zur Definition der Signatur enthalten.

- *Detached:* Die XML-Signatur wird in Form einer separaten Datei erstellt. In diesem Fall können Sie die Dateierweiterung der Signatur definieren und festlegen ob: (i) die Erweiterung an den Namen der XML-Datei angehängt werden soll (z.B. `test.xml.xsig`) oder (ii) ob die Erweiterung die XML-Erweiterung der Datei ersetzen soll (z.B. `test.xsig`). Außerdem können Sie festlegen, ob die Referenz auf die XML-Datei in der Signaturdatei ein relativer oder absoluter Pfad sein soll.

Anmerkung: XML-Signaturen für XML-Schemadateien (`.xsd`) und XBRL-Dateien können nur als externe Signaturdateien erstellt werden. Signaturen für WSDL-Dateien können als externe Dateien erstellt werden und auch in die WSDL-Datei verpackt (enveloped) werden.

Anmerkung: Wenn die XML-Signatur als separate Datei erstellt wird, werden die XML-Datei und die Signaturdatei durch eine Referenz in der Signaturdatei miteinander verknüpft. Folglich muss, wenn sich die Signatur in einer externen Datei befindet, bei der Überprüfung der Signatur die Signaturdatei und nicht die XML-Datei aktiv sein.

Schlüsselinformationen anhängen

Die Option *KeyInfo anhängen* steht nur dann zur Verfügung, wenn die Signatur auf einem Zertifikat basiert. Wenn *Schlüsselinformationen anhängen* aktiviert ist, werden öffentliche Schlüsselinformationen in die Signatur gesetzt. Im anderen Fall werden die Schlüsselinformationen nicht in die Signatur inkludiert. Der Vorteil bei der Inkludierung der Schlüsselinformationen ist, dass das Zertifikat selbst (insbesondere die Daten des öffentlichen Schlüssels) für die Überprüfung nicht benötigt werden (da die Schlüsselinformationen ja in der Signatur vorhanden sind).

Ungültige Signatureinstellungen
MapForce kann eine Ausgabe nicht mit einer digitalen Signatur versehen, wenn die Signatureinstellungen ungültig sind. Die Signatureinstellungen sind ungültig wenn (i) kein Zugriff auf das ausgewählte Zertifikat besteht oder es sich nicht zum Signieren von XML-Dokumenten eignet oder (ii) kein Passwort definiert wurde.

Wenn kein Passwort oder Zertifikat ausgewählt wird, wird die Verarbeitung entweder gestoppt oder ohne Signatur fortgesetzt. Sie können dies im Dialogfeld **Komponenteneinstellungen** festlegen. Wenn das Mapping über die Befehlszeile ausgeführt wird, erscheint kein solches Dialogfeld. Die Mapping-Ausführung wird entweder mit einem Fehler abgebrochen oder ohne Signatur fortgesetzt..

4.1.8.2 "Detached" oder "Enveloped"

Im Beispiel unten sehen Sie die Verwendung von separaten und eingebetteten Signaturen.

Separate Signaturen (detached)
Für das Beispielmapping unten wurde die Option *Detached* (separat) ausgewählt. Es wurden daher zwei separate Dateien generiert: (i) die Mapping-Ausgabe-Datei **`MarketingExpenses.xml`** und (ii) eine temporäre Signaturdatei namens **`MarketingExpenses.xml.xsig`** (*siehe Abbildung unten*). Sie finden das Mapping **`MarketingExpenses_DetachedSignature.mfd`** im Ordner **`MapForceExamples`**.

```
<?xml version="1.0" encoding="UTF-8"?>
<marketing-expenses xsi:noNamespaceSchemaLocation="
C:/Users/altova/Documents/Altova/MapForce2015/MapForceExamples/MarketingExpenses.xsd" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance">
  <Person>
    <FullName>Fred Landis</FullName>
    <Title>Project Manager</Title>
    <Phone>123-456-7890</Phone>
    <Email>f.landis@nanonull.com</Email>
  </Person>
  <expense-item>
    <type>Meal</type>
    <Date>2003-01-01</Date>
    <expense>122.11</expense>
  </expense-item>
  <expense-item>
    <type>Lodging</type>
    <Date>2003-01-02</Date>
    <expense>299.45</expense>
  </expense-item>
</marketing-expenses>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsig:Signature xmlns:xsig="http://www.w3.org/2000/09/xmldsig#">
  <xsig:SignedInfo>
    <xsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"></xsig:CanonicalizationMethod>
    <xsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1"></xsig:SignatureMethod>
    <xsig:Reference URI="~mf92B5.tmp">
      <xsig:Transforms>
        <xsig:Transform Algorithm="http://www.altova.com/xml-signatures/strip-whitespaces"></xsig:Transform>
      </xsig:Transforms>
      <xsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></xsig:DigestMethod>
      <xsig:DigestValue>g23J8QKWoBQnz3YZbSH/19KmQEw=</xsig:DigestValue>
    </xsig:Reference>
  </xsig:SignedInfo>
  <xsig:SignatureValue>expKkRUNufQquwOr4+gcsk7bHHU=</xsig:SignatureValue>
</xsig:Signature>
```

Um **.xml**- und **.xsig**-Dateien im Ausgabeverzeichnis zu generieren, klicken Sie auf die Symbolleisten-Schaltfläche **Alle generierten Ausgaben speichern**.

Eingebettete Signaturen (Enveloped)
Wenn Sie die Option *Enveloped signature* (eingebettete Signatur) auswählen, wird das Element `<xsig:Signature>` nach `<expense-item>` in die Ausgabedatei integriert (*siehe Abbildungen oben*). Ein Beispiel für diese Funktion finden Sie in **MapForceExamples\MarketingExpenses_EnvelopedSignature.mfd**.

Gültigkeit des XML-Dokuments

Wenn eine XML-Signatur in das XML-Dokument eingebettet wird, so wird ein `Signature` Element aus dem Namespace `http://www.w3.org/2000/09/xmldsig#` zum XML-Dokument hinzugefügt. Damit das Dokument dem Schema gemäß gültig bleibt, muss das Schema die entsprechenden Elementdeklarationen enthalten. Wenn das Schema des XML-Dokuments nicht geändert werden soll, verwenden Sie die Option *Detached* (*siehe oben*).

In den Codefragmenten unten sehen Sie, wie das Element `Signature` einer eingebetteten Signatur zulässigerweise verwendet werden kann. Im ersten Codefragment wird das XML-Signatur-Schema in das Schema des Benutzers importiert (*gelb markiert*).

© 2016-2022 Altova GmbH

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:xsig="http://www.w3.org/2000/09/xmldsig#"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">
   <xs:import namespace="http://www.w3.org/2000/09/xmldsig#"
              schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
schema.xsd"/>
   <xs:element name="Root">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="FirstChildOfRoot"/>
            <xs:element ref="SecondChildOfRoot" minOccurs="0"/>
            <xs:element ref="xsig:Signature" minOccurs="0"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   ...
</xs:schema>
```

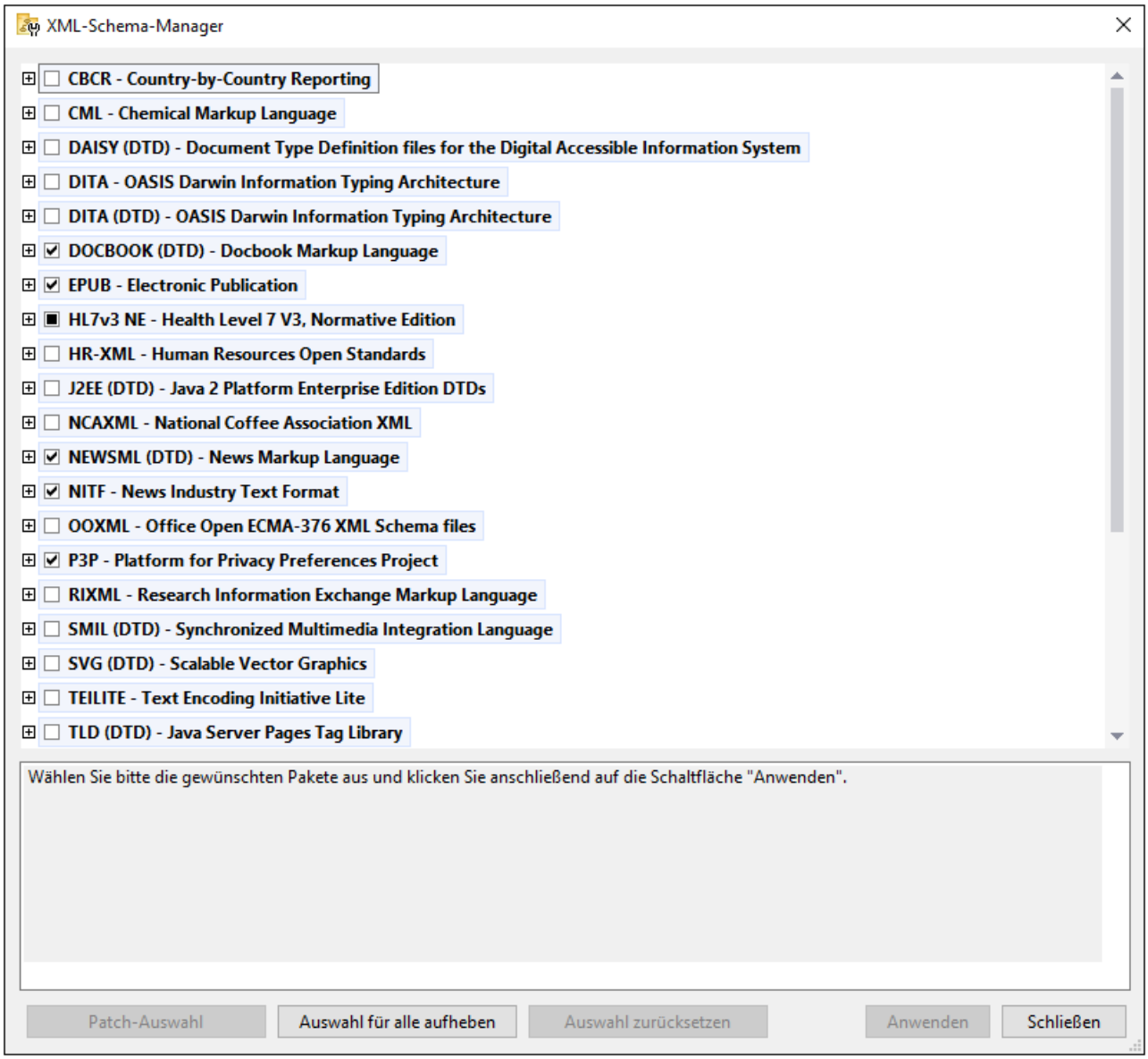
Die zweite Möglichkeit ist, ein allgemeines Wildcard-Element hinzuzufügen, das jedes Element aus anderen Namespaces repräsentiert (*unten gelb markiert*). Wenn Sie das Attribut `processContents` auf `lax` setzen, überspringt der Validator dieses Element, da keine übereinstimmende Elementdeklaration gefunden wird. Folglich muss der Benutzer das XML-Signatur-Schema nicht referenzieren. Der Nachteil diese Option ist allerdings, dass an der angegebenen Stelle im XML-Dokument jedes beliebige Element (und nicht nur das Element `Signature`) hinzugefügt werden kann, ohne dass das XML-Dokument ungültig wird.

```
<xs:element name="Root">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="selection"/>
         <xs:element ref="newsitems" minOccurs="0"/>
         <xs:element ref="team" minOccurs="0"/>
         <xs:any namespace="##other" minOccurs="0" processContents="lax"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

4.1.9 Schema-Manager

Der XML-Schema-Manager ist ein Altova-Tool, mit dem Sie XML-Schemas (DTDs für XML-Dateien und XML-Schemas) zentral installieren und verwalten können, um diese in allen XML-fähigen Applikationen von Altova einschließlich MapForce verwenden zu können.

- Unter Windows hat der Schema-Manager eine grafische Benutzeroberfläche (*siehe Abbildung unten*) und steht auch über die Befehlszeile zur Verfügung. (Die Desktop-Applikationen von Altova stehen nur unter Windows zur Verfügung; *siehe Liste unten*).
- Unter Linux und MacOS steht der Schema-Manager nur über die Befehlszeile zur Verfügung. (Die Server-Applikationen von Altova stehen unter Windows, Linux und macOS zur Verfügung; *siehe Liste unten*).



Altova-Applikationen, die mit Schema-Manager arbeiten:

Desktop-Applikationen (nur Windows)	Server-Applikationen (Windows, Linux, macOS)
XMLSpy (alle Editionen)	RaptorXML Server, RaptorXML+XBRL Server
MapForce (alle Editionen)	StyleVision Server
StyleVision (alle Editionen)	
Authentic Desktop Enterprise Edition	

© 2016-2022 Altova GmbH

Installation und Deinstallation des Schema-Managers

Der Schema-Manager wird bei der ersten Installation einer neuen Version der Altova Mission Kit Enterprise Edition oder einer der XML-Schema-fähigen Applikationen von Altova (*siehe Tabelle oben*) automatisch installiert.

Ebenso wird er auch automatisch entfernt, wenn Sie die letzte XML-Schema-fähige Applikation von Altova auf Ihrem Rechner deinstallieren.

Schema-Manager-Funktionalitäten

Im Schema-Manager stehen die folgenden Funktionalitäten zur Verfügung:

- Anzeigen der auf Ihrem Rechner installierten XML-Schemas und Überprüfung, ob neue Versionen zum Download zur Verfügung stehen.
- Download neuer Versionen von XML-Schemas unabhängig vom Altova Produkt-Release-Zyklus. (Die Schemas werden von Altova online bereitgestellt und können über den Schema-Manager heruntergeladen werden).
- Installation oder Deinstallation jeder beliebigen (oder ggf. aller) der zahlreichen Versionen eines bestimmten Schemas.
- Ein XML-Schema kann Abhängigkeiten von anderen Schemas aufweisen. Bei der Installation oder Deinstallation eines bestimmten Schemas informiert Sie der Schema-Manager über davon abhängige Schemas und installiert bzw. entfernt diese ebenfalls automatisch.
- Der Schema-Manager ordnet Schema-Referenzen mit Hilfe des XML-Katalogs lokalen Dateien zu. Dadurch lassen sich große XML-Schemas schneller verarbeiten, als wenn sie sich unter einem entfernten Pfad befinden.
- Alle wichtigen Schemas werden über den Schema-Manager bereitgestellt und regelmäßig auf die jeweils neuesten Version aktualisiert. Dadurch können alle Ihre Schemas zentral verwaltet werden und stehen allen XML-Schema-fähigen Applikationen von Altova jederzeit zur Verfügung.
- Im Schema-Manager vorgenommene Änderungen werden für alle auf dem Rechner installierten Altova-Produkte wirksam.

Funktionsweise

Alle in Altova-Produkten verwendeten XML-Schemas werden von Altova online bereitgestellt. Dieser Speicher wird bei Veröffentlichung neuer Versionen der Schemas aktualisiert. Im Schema-Manager werden sowohl bei Aufruf über die Benutzeroberfläche als auch über das CLI Informationen über die neuesten verfügbaren Schemas angezeigt. Sie können die gewünschten Schemas dann über den Schema-Manager installieren, aktualisieren oder deinstallieren.

Schemas können vom Schema-Manager auch auf eine weitere Art installiert werden. Sie können ein Schemas und die davon abhängigen Schemas auf der Altova Website (https://www.altova.com/de/schema-manager) auswählen. Daraufhin wird auf der Website eine Datei des Typs **`.altova_xmlschemas`** mit Informationen über Ihre ausgewählten Schemas zum Download vorbereitet. Bei Doppelklick auf diese Datei oder bei Übergabe an den Schema-Manager über das CLI als Argument des Befehls `install` 159 installiert der Schema-Manager die ausgewählten Schemas.

Lokaler Cache: Überprüfung Ihrer Schemas

Alle Informationen über installierte Schemas werden in einem zentralen Cache-Verzeichnis auf Ihrem Rechner aufgezeichnet. Das Verzeichnis befindet sich hier:

Windows	`C:\ProgramData\Altova\pkgs\.cache`

Linux	`/var/opt/Altova/pkgs\.cache`
macOS	`/var/Altova/pkgs`

Dieses Cache-Verzeichnis wird regelmäßig mit dem neuesten Status der Schemas aus dem Online-Speicher von Altova aktualisiert. Diese Aktualisierungen finden unter den folgenden Bedingungen statt:

- bei jedem Start von Schema-Manager.
- Wenn Sie MapForce zum ersten Mal an einem bestimmten Kalendertag starten.
- Wenn MapForce länger als 24 Stunden geöffnet ist, findet alle 24 Stunden eine Aktualisierung des Cache statt.
- Sie können den Cache auch durch Ausführung des `update`[163]-Befehls über die Befehlszeilenschnittstelle aktualisieren.

Der Schema-Manager kann somit Ihre installierten Schemas über den Cache ständig anhand der online verfügbaren Schemas auf der Altova Website überprüfen.

Nehmen Sie keine manuellen Änderungen am Cache vor!

Das lokale Cache-Verzeichnis wird automatisch auf Basis der installierten oder deinstallierten Schemas verwaltet; es sollte nicht manuell geändert oder gelöscht werden. Falls Sie den Schema-Manager je in seinen Originalzustand zurücksetzen möchten, (i) führen Sie den CLI-Befehl `reset`[161] der Befehlszeilenschnittstelle und (ii) anschließend den Befehl `initialize`[159] aus. (Führen Sie alternativ dazu den Befehl **`reset`** mit der Option **`-i`** aus).

4.1.9.1 Ausführen des Schema-Managers

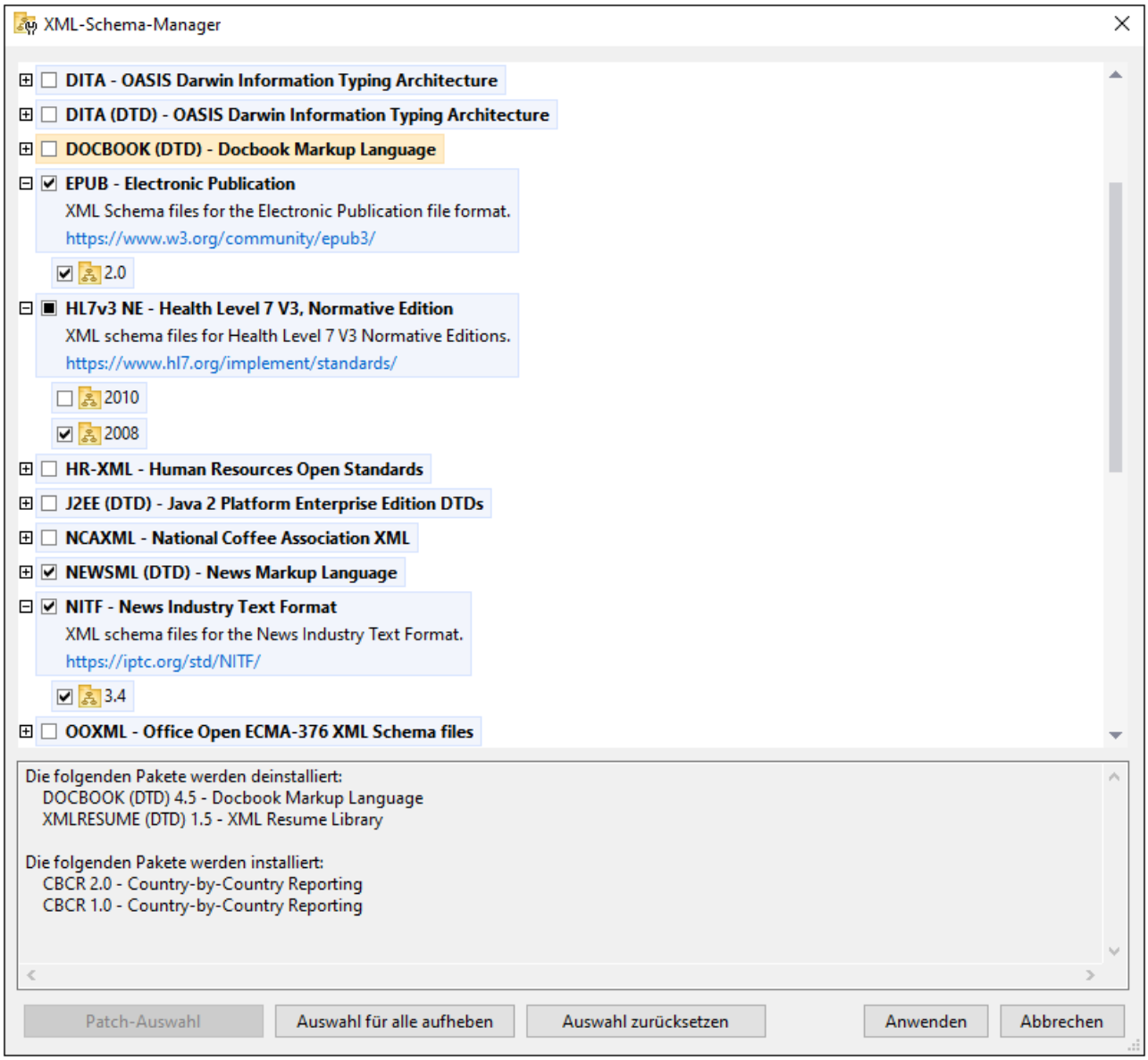
Grafische Benutzeroberfläche

Sie können die Benutzeroberfläche des Schema-Managers auf eine der folgenden Arten aufrufen:

- *Bei der Installationen von MapForce:* Aktivieren Sie gegen Ende der Installation das Kontrollkästchen *Altova-Schema-Manager aufrufen*, wodurch Sie die Benutzeroberfläche des Schema-Managers direkt aufrufen können. Auf diese Art können Sie Schemas während der Installation Ihrer Altova-Applikation installieren.
- *Nach der Installation von MapForce:* Nachdem die Applikation installiert wurde, können Sie die Benutzeroberfläche des Schema-Managers jederzeit über den Menübefehl **Tools | XML-Schemas-Manager** aufrufen.
- über die von der Altova Webseite heruntergeladene **`.altova_xmlschemas`**-Datei: Doppelklicken Sie auf die heruntergeladene Datei, um den Schema-Manager zu starten, der daraufhin die (auf der Website) ausgewählten Schemas installiert.

Nachdem die Benutzeroberfläche des Schema-Managers geöffnet wurde (*Abbildung unten*), werden bereits installierte Schemas markiert angezeigt. Wenn ein zusätzliches Schema installiert werden soll, aktivieren Sie dieses. Wenn ein bereits installiertes Schema deinstalliert werden soll, deaktivieren Sie dieses. Nachdem Sie Ihre Auswahl getroffen haben, können Ihre Änderungen angewendet werden. Die Schemas, die installiert bzw. deinstalliert werden, werden markiert und im Fenster "Meldungen" am unteren Rand des Schema-Manager-Fensters (*siehe Abbildung*) erscheint eine Meldung über die bevorstehenden Änderungen.

© 2016-2022 Altova GmbH



Befehlszeilenschnittstelle

Sie können den Schema-Manager über eine Befehlszeilenschnittstelle starten, indem Sie Befehle an die ausführbare Datei **`xmlschemamanager.exe`** senden.

Die Datei **`xmlschemamanager.exe`** steht im folgenden Ordner zur Verfügung:

- *unter Windows:* `C:\ProgramData\Altova\SharedBetweenVersions`
- *Unter Linux oder macOS (nur Server-Applikationen):* `%INSTALLDIR%/bin`, wobei `%INSTALLDIR%` das Installationsverzeichnis des Programms ist.

Anschließend können Sie jeden der im Abschnitt zur CLI-Befehlsreferenz 157 aufgelisteten Befehle verwenden.

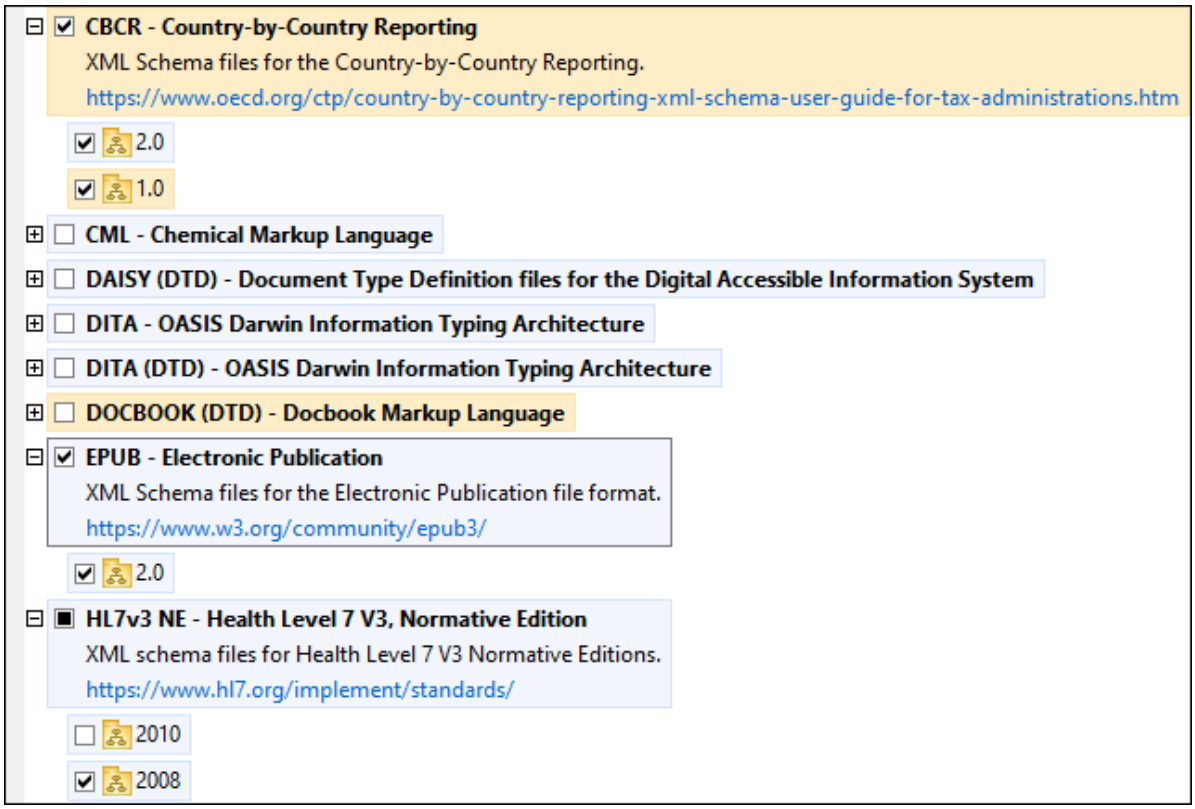
Um die Hilfe zu den Befehlen anzuzeigen, führen Sie den folgenden Befehl aus:

- *unter Windows:* `xmlschemamanager.exe --help`
- *Unter Linux oder macOS (nur Server-Applikationen):* `sudo ./xmlschemamanager --help`

4.1.9.2 Statuskategorien

Der Schema-Manager unterscheidet folgendermaßen zwischen den von ihm verwalteten Schemas:

- *installierte Schemas.* Diese werden auf der Benutzeroberfläche mit einem Häkchen angezeigt (*in der Abbildung unten sind die mit einem Häkchen versehenen und blau angezeigten Versionen der EPUB- und HL7v3 NE-Schemas installiert*). Wenn alle Versionen eines Schemas ausgewählt sind, wird ein Häkchen angezeigt. Wenn zumindest eine Version nicht ausgewählt ist, wird ein gefülltes Quadrat angezeigt. Sie können das Kontrollkästchen für ein installiertes Schema deaktivieren, um es zu **deinstallieren**; (*in der Abbildung unten ist die DocBook DTD installiert und wurde deaktiviert; sie wird daher für die Deinstallation vorbereitet*).
- *Nicht installierte verfügbare Schemas.* Diese werden auf der Benutzeroberfläche mit einem deaktivierten Kontrollkästchen angezeigt. Sie können die Schemas, die **installiert** werden sollen, auswählen.



- *Schemas, für die ein Upgrade zur Verfügung steht* sind diejenigen, die seit ihrer Installation vom Herausgeber überarbeitet wurden. Sie werden auf der Benutzeroberfläche durch ein -Symbol gekennzeichnet. Sie können für ein installiertes Schema ein **Patch** der verfügbaren überarbeiteten Version installieren.

© 2016-2022 Altova GmbH

Wichtige Punkte

- In der Abbildung oben sind beide CBCR-Schemas ausgewählt. Dasjenige mit einem blauen Hintergrund ist bereits installiert. Dasjenige mit dem gelben Hintergrund ist nicht installiert und wurde für die Installation ausgewählt. Beachten Sie, dass das Schema "HL7v3 NE 2010" nicht installiert ist und nicht für die Installation ausgewählt wurde.
- Ein gelber Hintergrund bedeutet, dass das Schema auf irgendeine Art geändert wird, wenn Sie auf die Schaltfläche **Anwenden** klicken. Wenn ein Schema deaktiviert ist und einen gelben Hintergrund hat, bedeutet dies, dass es bei Klick auf die Schaltfläche **Anwenden** deinstalliert wird. In der Abbildung oben ist dies bei der DocBook DTD der Fall.
- Bei Ausführung des Schema-Managers über die Befehlszeile wird der Befehl `list` 160 mit verschiedenen Optionen verwendet, um verschiedene Schemakategorien aufzulisten:

`xmlschemamanager.exe list`	Listet alle installierten und verfügbaren Schemas auf; auch verfügbare Upgrades werden angezeigt.
`xmlschemamanager.exe list -i`	Listet nur installierte Schemas auf; auch verfügbare Upgrades werden angezeigt.
`xmlschemamanager.exe list -u`	Listet Schemas auf, für die Upgrades zur Verfügung stehen

Anmerkung: Verwenden Sie unter Linux und macOS **`sudo ./xmlschemamanager list`**

4.1.9.3 Anwenden eines Patch oder Installation eines Schemas

Anwenden eines Patch auf ein installiertes Schema

Von Zeit zu Zeit werden von den Herausgebern der XML-Schemas Patches (Upgrades oder Überarbeitungen) veröffentlicht. Wenn der Schema-Manager erkennt, dass Patches zur Verfügung stehen, werden diese in der Schemaliste des Schema-Managers angezeigt und Sie können diese Patches schnell installieren.

Über die Benutzeroberfläche

Patches werden mit dem Symbol gekennzeichnet. (*Siehe auch vorhergehendes Kapitel über Statuskategorien* 154*).* Falls Patches zur Verfügung stehen, ist die Schaltfläche **Patch-Auswahl** aktiv. Klicken Sie darauf, um alle Patches für die Installation auszuwählen und vorzubereiten. Auf der Benutzeroberfläche ändert sich das Symbol von Schemas, für die ein Patch installiert wird von in , und im Fenster "Meldungen" am unteren Rand des Dialogfelds werden die Patches, die angewendet werden, aufgelistet. Sobald Sie mit der Auswahl fertig sind, klicken Sie auf **Anwenden**. Alle Patches werden gemeinsam angewendet. Beachten Sie, dass ein für die Installation eines Patch markiertes Schema deinstalliert wird, wenn Sie die Auswahl aufheben.

Über das CLI

So wenden Sie einen Patch über die Befehlszeilenschnittstelle an:

1. Führen Sie den Befehl `list -u` 160 aus. Daraufhin werden alle Schemas, für die Upgrades zur Verfügung stehen, aufgelistet.
2. Führen Sie den Befehl `upgrade` 163 aus, um alle Patches zu installieren.

Installieren eines verfügbaren Schemas

Sie können Schemas entweder über die Benutzeroberfläche des Schema-Managers oder durch Senden der Schema-Manager-Installationsbefehle über die Befehlszeile installieren.

Anmerkung: Wenn das aktuelle Schema andere Schemas referenziert, werden auch die referenzierten Schemas installiert.

Über die Benutzeroberfläche

Um Schemas über die Benutzeroberfläche des Schema-Managers zu installieren, wählen Sie die gewünschten Schemas aus und klicken Sie auf **Anwenden**.

Sie können die gewünschten Schemas auch auf der Altova Website auswählen und eine herunterladbare `.altova_xmlschemas`-Datei generieren. Bei Doppelklick auf diese Datei wird der Schema-Manager aufgerufen, in dem die gewünschten Schemas bereits vorausgewählt sind. Sie müssen nur mehr auf **Anwenden** klicken.

Über das CLI

Um Schemas über die Befehlszeile zu installieren, rufen Sie den Befehl `install`[159] auf:

```
xmlschemamanager.exe install [options] Schema+
```

wobei es sich bei **Schema** um das/die gewünschte(n) Schema(s) bzw. eine **.altova_xmlschemas**-Datei handelt. Ein Schema wird von einem Identifier im Format **<name>-<version>** referenziert. (Die Identifier von Schemas werden angezeigt, wenn Sie den Befehl `list`[160] ausführen.) Sie können beliebig viele Schemas eingeben. Nähere Informationen dazu finden Sie unter der Beschreibung des Befehls `install`[159].

Anmerkung: Verwenden Sie unter Linux oder macOS den Befehl **sudo ./xmlschemamanager**.

Installation eines benötigten Schemas

Wenn Sie einen XML-Schema-Befehl in MapForce ausführen und MapForce erkennt, dass ein zur Ausführung des Befehls erforderliches Schema nicht vorhanden oder unvollständig ist, wird der Schema-Manager mit Informationen über das/die fehlende(n) Schema(s) aufgerufen. Sie können die gewünschten Schemas dann über den Schema-Manager direkt installieren.

Alle bereits installierten Schemas können jederzeit durch Aufruf des Schema-Managers über **Extras | Schema-Manager** über die Benutzeroberfläche des Schema-Managers angezeigt werden.

4.1.9.4 Deinstallieren eines Schemas, Zurücksetzen

Deinstallieren eines Schemas

Sie können Schemas entweder über die Benutzeroberfläche des Schema-Managers oder durch Senden der Schema-Manager-Deinstallationsbefehle über die Befehlszeile deinstallieren.

Anmerkung: Wenn das gewünschte Schema andere Schemas referenziert, so werden auch die referenzierten Schemas deinstalliert.

© 2016-2022 Altova GmbH

Über die Benutzeroberfläche
Um Schemas über die Benutzeroberfläche des Schema-Managers zu deinstallieren, deaktivieren Sie die Kontrollkästchen der entsprechenden Schemas und klicken Sie auf **Anwenden**. Daraufhin werden die ausgewählten Schemas und die davon referenzierten Schemas deinstalliert.

Um alle Schemas zu deinstallieren, klicken Sie auf **Auswahl für alle aufheben** und anschließend auf **Anwenden**.

Über das CLI
Um Schemas über die Befehlszeile zu deinstallieren, rufen Sie den Befehl `uninstall`[162] auf:

```
xmlschemamanager.exe uninstall [options] Schema+
```

wobei es sich beim Argument **Schema** ein zu deinstallierendes Schema oder eine **.altova_xmlschemas**-Datei handelt. Ein Schema wird von einem Identifier im Format **<name>-<version>** definiert. (Die Identifier von Schemas werden angezeigt, wenn Sie den Befehl `list`[160] ausführen.) Sie können beliebig viele Schemas eingeben. Nähere Informationen dazu finden Sie unter der Beschreibung des Befehls `uninstall`[162].

Anmerkung: Verwenden Sie unter Linux oder macOS den Befehl **sudo ./xmlschemamanager**.

Zurücksetzen des Schema-Managers

Sie können den Schema-Manager zurücksetzen. Damit werden alle installierten Schemas und das Cache-Verzeichnis entfernt.

- Klicken Sie auf der Benutzeroberfläche auf **Auswahl zurücksetzen**.
- Führen Sie über die Benutzeroberfläche den Befehl `reset`[161] aus.

Nachdem Sie diesen Befehl ausgeführt haben, muss der Befehl `initialize`[159] ausgeführt werden, um das Cache-Verzeichnis neu zu erstellen. Führen Sie alternativ dazu den Befehl `reset`[161] mit der Option `-i` aus.

Beachten Sie, dass mit `reset-i`[161] die Originalinstallation des Produkts wiederhergestellt wird, daher wird empfohlen, nach dem Zurücksetzen auch den Befehl `update`[163] auszuführen. Führen Sie alternativ dazu den Befehl `reset`[161] mit den Optionen `-i` und `-u` aus.

4.1.9.5 Befehlszeilenschnittstelle (CLI)

Um den Schema-Manager über die Befehlszeile aufzurufen, müssen Sie den Pfad zur ausführbaren Datei kennen. Standardmäßig befindet sich die ausführbare Schema-Manager-Datei hier:

```
C:\ProgramData\Altova\SharedBetweenVersions\XMLSchemaManager.exe
```

Anmerkung: Nachdem Sie auf Linux- und macOS-Systemen das Verzeichnis in dasjenige, das die ausführbare Datei enthält, geändert haben, können Sie die ausführbare Datei mit **sudo ./xmlschemamanager** aufrufen. Das Präfix **./** gibt an, dass sich die ausführbare Datei im aktuellen

Verzeichnis befindet. Das Präfix **sudo** gibt an, dass der Befehl mit Root-Rechten ausgeführt werden muss.

Befehlszeilensyntax

Die allgemeine Syntax zur Verwendung der Befehlszeile lautet folgendermaßen:

```
<exec> -h | --help | --version | <command> [options] [arguments]
```

Der senkrechte Balken **|** im Codefragment oben trennt eine Gruppe einander gegenseitig ausschließender Elemente. Optionale Elemente stehen innerhalb von eckigen Klammern **[]**. Im Prinzip können Sie den Pfad zur ausführbaren Datei, gefolgt von entweder `-h`, `--help` oder `--version`-Optionen oder gefolgt von einem Befehl eingeben. Jeder Befehl kann Optionen und Argumente haben. Die Liste der Befehle wird in den folgenden Abschnitten beschrieben.

4.1.9.5.1 help

Mit diesem Befehl erhalten Sie Hilfe zu Befehlen zur ausführbaren Schema-Manager-Datei.

Syntax

```
<exec> help [Befehl]
```

`[Befehl]` ist hierbei ein optionales Argument zur Angabe jedes beliebigen gültigen Befehlsnamens.

Beachten Sie dazu Folgendes:

- Sie können die Hilfe zu einem Befehl auch durch Eingabe des Befehls, gefolgt von **-h** oder **--help** aufrufen, z.B: **<exec> list -h**
- Wenn Sie **-h** oder **--help** direkt nach dem Namen der ausführbaren Datei und vor einem Befehl eingeben, wird die allgemeine Hilfe (und nicht die Hilfe zu einem bestimmten Befehl) angezeigt, z.B: **<exec> -h list**

Beispiel

Mit dem folgenden Befehl wird Hilfe zum Befehl `list` angezeigt:

```
xmlschemamanager help list
```

4.1.9.5.2 info

Mit diesem Befehl werden ausführliche Informationen über die einzelnen als `Schema`-Argument angegebenen Schemas angezeigt. Darin enthalten sind Titel, Version, Beschreibung, Herausgeber der jeweils angegebenen Schemas und davon referenzierte Schemas sowie die Information, ob das Schema installiert ist oder nicht.

Syntax

```
<exec> info [options] Schema+
```


© 2016-2022 Altova GmbH

- Das Argument **Schema** ist der Name eines Schemas oder Teil eines Schemanamens. (Die Paket-ID eines Schemas und detaillierte Informationen über ihren Installationsstatus erhalten Sie mit dem Befehl list[160].)
- Mit **<exec> info -h** können Sie die Hilfe zum Befehl anzeigen.

Beispiel

Mit dem folgenden Befehl werden Informationen über das jeweils neueste **DocBook-DTD**- und **NITF**-Schemas angezeigt.

```
xmlschemamanager info doc nitf
```

4.1.9.5.3 initialize

Mit diesem Befehl wird die Schema-Manager-Umgebung initialisiert. Sie erstellen damit ein Cache-Verzeichnis, in dem Informationen über alle Schemas lokal gespeichert werden. Die Initialisierung erfolgt automatisch bei der ersten Installation einer Schema-fähigen Altova-Applikation. Normalerweise muss dieser Befehl nicht ausgeführt werden. Nach Ausführung des **reset**-Befehls ist dies allerdings erforderlich.

Syntax

```
<exec> initialize | init [options]
```

Optionen
Für den Befehl **initialize** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl wird der Schema-Manager initialisiert:

```
xmlschemamanager initialize
```

4.1.9.5.4 install

Mit diesem Befehl installieren Sie ein oder mehrere Schemas.

Syntax

```
<exec> install [options] Schema+
```

Um mehrere Schemas zu installieren, fügen Sie das Argument **Schema** mehrmals hinzu.

Als **Schema**-Argument kann eines der folgenden verwendet werden:

- Ein Schema-Identifier (im Format **<name>-<version>**, z.B: **cbcr-2.0**). Um die Schema-Identifier der gewünschten Schemas zu eruieren, führen Sie den Befehl `list` [160] aus. Sie können auch einen abgekürzten Identifier verwenden, sofern dieser eindeutig ist, z.B. **docbook**. Falls Sie einen abgekürzten Identifier verwenden, wird die neueste Version dieses Schemas installiert.
- Der Pfad zu einer von der Altova-Website heruntergeladenen **.altova_xmlschemas**-Datei. Informationen zu diesen Dateien finden Sie in der *Einführung zu Schema-Manager: Funktionsweise* [149].

Optionen
Für den Befehl **install** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl werden das CBCR 2.0 (Country-By-Country Reporting)-Schema und die neueste DocBook-DTD installiert:

```
xmlschemamanager install cbcr-2.0 docbook
```

4.1.9.5.5 list

Mit diesem Befehl werden vom Schema-Manager verwaltete Schemas aufgelistet. In der Liste wird eine der folgenden Informationen angezeigt:

- alle verfügbaren Schemas
- Schemas, die im Namen den im Argument **Schema** angegebenen String enthalten
- nur installierte Schemas
- Nur Schemas, für die ein Upgrade installiert werden kann

Syntax

```
<exec> list | ls [options] Schema?
```

Wenn kein **Schema**-Argument angegeben wird, werden alle verfügbaren Schemas aufgelistet. Andernfalls werden die durch die angegebenen Optionen definierten Schemas aufgelistet (*siehe Beispiel unten*). Beachten Sie, dass Sie das Argument **Schema** mehrfach angeben können.

Optionen
Für den Befehl **list** stehen die folgenden Optionen zur Verfügung:

© 2016-2022 Altova GmbH

`--installed, --i`	Auflisten nur der installierten Schemas. Der Standardwert ist **`false`**.
`--upgradeable, --u`	Auflisten nur derjenigen Schemas, für die Upgrades (Patches) zur Verfügung stehen. Der Standardwert ist **`false`**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiele

- Um alle verfügbaren Schemas aufzulisten, führen Sie den folgenden Befehl aus: **`xmlschemamanager list`**
- Um nur installierte Schemas aufzulisten, führen Sie **`xmlschemamanager list -i`** aus.
- Um Schemas, die in ihrem Namen entweder "doc" oder "nitf" enthalten, aufzulisten, führen Sie **`xmlschemamanager list doc nitf`** aus.

4.1.9.5.6 reset

Mit diesem Befehl werden alle installierten Schemas und das Cache-Verzeichnis entfernt. Ihre Schemaumgebung wird vollständig zurückgesetzt. Nachdem Sie diesen Befehl ausgeführt haben, muss der Befehl `initialize` 159 ausgeführt werden, um das Cache-Verzeichnis neu zu erstellen. Führen Sie alternativ dazu den Befehl **`reset`** mit der Option **`-i`** aus. Da mit **`reset-i`** die Originalinstallation des Produkts wiederhergestellt wird, wird empfohlen, nach dem Zurücksetzen und Initialisieren auch den Befehl `update` 163 auszuführen. Führen Sie alternativ dazu den Befehl **`reset`** mit den Optionen **`-i`** und **`-u`** aus.

Syntax

```
<exec> reset [Optionen]
```

Optionen
Für den Befehl **`reset`** stehen die folgenden Optionen zur Verfügung:

`--init, --i`	Initialisierung des Schema-Managers nach dem Zurücksetzen. Der Standardwert ist **`false`**.
`--update, --u`	Aktualisiert die Liste der verfügbaren Schemas im Cache. Der Standardwert ist **`false`**.
`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **`false`**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **`false`**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiele

- Um den Schema-Manager zurückzusetzen, führen Sie den folgenden Befehl aus: **`xmlschemamanager reset`**
- Um den Schema-Manager zurückzusetzen und ihn zu initialisieren, führen Sie **`xmlschemamanager reset -i`** aus.
- Um den Schema-Manager zurückzusetzen, ihn zu initialisieren und seine Schemaliste zu aktualisieren, führen Sie **`xmlschemamanager reset -i -u`** aus.

4.1.9.5.7 uninstall

Mit diesem Befehl deinstallieren Sie ein oder mehrere Schemas. Standardmäßig werden auch alle Schemas, die vom der aktuellen Schema referenziert werden, deinstalliert. Um nur das aktuelle Schema zu deinstallieren und die referenzierten Schemas beizubehalten, setzen Sie die Option `--k`.

Syntax

```
<exec> uninstall [options] Schema+
```

Um mehrere Schemas zu deinstallieren, fügen Sie das Argument **`Schema`** mehrmals hinzu.

Als **`Schema`**-Argument kann eines der folgenden verwendet werden:

- Ein Schema-Identifier (im Format **`<name>-<version>`**, z.B: **`cbcr-2.0`**). Um die Schema-Identifier der installierten Schemas zu eruieren, führen Sie den Befehl `list -i` 160 aus. Sie können auch einen abgekürzten Schemanamen verwenden, sofern dieser eindeutig ist, z.B. **`docbook`**. Falls Sie einen abgekürzten Namen verwenden, werden alle Schemas, die die Abkürzung in ihrem Namen enthalten, deinstalliert.
- Der Pfad zu einer von der Altova-Website heruntergeladenen **`.altova_xmlschemas`**-Datei. Informationen zu diesen Dateien finden Sie in der *Einführung zu Schema-Manager: Funktionsweise* 149.

Optionen
Für den Befehl **`uninstall`** stehen die folgenden Optionen zur Verfügung:

`--keep-references, --k`	Definieren Sie diese Option, um referenzierte Schemas beizubehalten. Der Standardwert ist **`false`**.
`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **`false`**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **`false`**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl werden die Schemas CBCR 2.0 und EPUB 2.0 und deren Abhängigkeiten deinstalliert:

```
xmlschemamanager uninstall cbcr-2.0 epub-2.0
```

 © 2016-2022 Altova GmbH

Mit dem folgenden Befehl wird das **eba-2.10**-Schema, nicht aber die davon referenzierten Schemas deinstalliert:

```
xmlschemamanager uninstall --k cbcr-2.0
```

4.1.9.5.8 update

Mit diesem Befehl wird die Liste der über den Online-Speicher verfügbaren Schemas abgefragt und das lokale Cache-Verzeichnis wird aktualisiert. Normalerweise muss dieser Befehl nur ausgeführt werden, wenn Sie reset[161] und initialize[159] ausgeführt haben.

Syntax

```
<exec> update [options]
```

Optionen

Für den Befehl **update** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl wird der lokale Cache mit der Liste der neuesten Schemas aktualisiert:

```
xmlschemamanager update
```

4.1.9.5.9 upgrade

Mit diesem Befehl werden alle installierten Schemas, für die ein Upgrade installiert werden kann, auf die neueste verfügbare *Patch*-Version aktualisiert. Um herauszufinden, welche Schemas aktualisiert werden können, starten Sie den Befehl list-u[160].

Anmerkung: Der Befehl **upgrade** entfernt ein veraltetes Schema, falls keine neuere Version zur Verfügung steht.

Syntax

```
<exec> upgrade [Optionen]
```

Optionen

Für den Befehl **upgrade** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.

`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **`false`**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

© 2016-2022 Altova GmbH

4.2 Datenbanken

Altova Website: Datenbank-Mapping

MapForce 2023 bietet umfangreiche Unterstützung für Datenbank-Mappings auf XML-Dateien, Flat Files und andere Datenbankformate. Mit der MapForce Enterprise Edition können Sie außerdem Datenbanken auf EDI-Formate, Excel 2007+, JSON, XBRL und Webservices mappen.

Die nachstehende Tabelle enthält eine Liste aller unterstützten Datenbanken. Altova ist bestrebt, auch Unterstützung für andere Datenbanken zu gewährleisten, doch wurde die erfolgreiche Herstellung der Verbindung und Verarbeitung der Daten nur für die aufgelisteten Datenbanken getestet. Wenn es sich bei Ihrer Altova-Applikation um eine 64-Bit-Version handelt, stellen Sie sicher, dass Sie Zugriff auf die 64-Bit-Datenbanktreiber haben, die für die entsprechenden Datenbank benötigt werden.

Datenbank	Anmerkungen
Apache CouchDB 3.x	NoSQL-Datenbank. Nähere Informationen dazu finden Sie in der CouchDB-Dokumentation. Um zu erfahren, wie NoSQL-Datenbanken in MapForce funktionieren, lesen Sie nach unter NoSQL-Datenbanken 453.
Firebird 2.x, 3.x	
IBM DB2 8.x, 9.x, 10.x, 11.x	
IBM Db2 for i 6.x, 7.4	Logische Dateien werden unterstützt und als Ansichten angezeigt.
IBM Informix 11.70 und höher	Informix unterstützt Verbindungen über ADO, JDBC und ODBC. Die Implementierung unterstützt in keiner der Codegenerierungssprachen große Objektdatentypen. Bei Verwendung eines dieser Datentypen erzeugt MapForce (bei der Codegenerierung) eine Fehlermeldung.
MariaDB 10 und höher	
Microsoft Access 2003 und höher	Zum Zeitpunkt der Verfassung dieser Dokumentation (Anfang September 2019) gibt es kein Microsoft Access Runtime für Access 2019. Sie können nur dann eine Verbindung von Altova-Produkten zu einer Access 2019-Datenbank herstellen, wenn Microsoft Access 2016 Runtime installiert ist und der Datentyp "Large Number" in der Datenbank nicht verwendet wird.
Microsoft Azure SQL-Datenbank	SQL Server 2016 Codebase
Microsoft SQL Server 2005 und höher Microsoft SQL Server unter Linux	
MongoDB 4.x	NoSQL-Datenbank. Nähere Informationen dazu finden Sie in der MongoDB-Dokumentation. Um zu erfahren, wie NoSQL-Datenbanken in MapForce funktionieren, lesen Sie nach unter NoSQL-Datenbanken 453.

Datenbank	Anmerkungen
MySQL 5 und höher	
Oracle 9i und höher	
PostgreSQL 8 und höher	PostgreSQL-Verbindungen werden sowohl als native Verbindungen als auch als treiberbasierte Verbindungen über Schnittstellen (Treiber) wie ODBC oder JDBC unterstützt. Für native Verbindungen werden keine Treiber benötigt.
Progress OpenEdge 11.6	
SQLite 3.x	SQLite-Verbindungen werden als native, direkte Verbindungen zur SQLite-Datenbankdatei unterstützt. Es sind keine separaten Treiber erforderlich.
Sybase ASE15, 16	
Teradata 16	Verbindungen werden über ADO.NET, JDBC und ODBC unterstützt. Wenn durch ein Mapping Daten in eine Datenbanktabelle eingefügt werden, werden datenbankgenerierte ID-Felder nicht unterstützt.

Datenbankmappings in verschiedenen Ausführungsumgebungen

Wenn Sie Programmcode anhand eines Mappings generieren oder wenn Sie ein Mapping zu einer MapForce Server-Ausführungsdatei kompilieren oder ein Mapping auf FlowForce Server bereitstellen, werden die mit den generierten Dateien gespeicherten Datenbankverbindungsinformationen, wie in der Tabelle unten gezeigt, an die jeweiligen Treiber bzw. an die für die ausgewählte Zielumgebung unterstützten Treiber angepasst. Wenn als Mapping-Transformationssprache z.B. Java ausgewählt ist, so werden ADO-Verbindungen in JDBC konvertiert, wenn anhand des Mappings Java-Code generiert wird.

Wenn das Mapping in einer anderen Umgebung als MapForce ausgeführt wird, müssen Sie sicherstellen, dass die Datenbankinformationen auf dem Rechner, auf dem das Mapping ausgeführt wird, sinnvoll sind (z.B. dass der Datenbanktreiber installiert und der Datenbankpfad korrekt ist und Zugriff auf den Datenbankserver besteht usw.).

Einige Datenbankverbindungen werden in einigen Zielumgebungen nicht unterstützt (siehe Tabelle unten).

Verbindungstyp/Ausführungsum-gebung	C#	C++	Java	MapForce Server auf Windows	MapForce Server auf Linux/Mac
ADO	ADO Bridge	Wird beibehalten	Wird in JDBC konvertiert	Wird beibehalten	Wird in JDBC konvertiert
ADO.NET	Wird beibehalten	Benutzerdefiniert	Wird in JDBC konvertiert	Wird beibehalten	Wird in JDBC konvertiert

© 2016-2022 Altova GmbH

Verbindungstyp/Ausführungsum-gebung	C#	C++	Java	MapForce Server auf Windows	MapForce Server auf Linux/Mac
JDBC	Benutzerdefiniert	Benutzerdefiniert	Wird beibehalten	Wird beibehalten	Wird beibehalten
ODBC	ODBC Bridge	ODBC Bridge	Wird in JDBC konvertiert	Wird beibehalten	Wird in JDBC konvertiert
Natives PostgreSQL	Nicht unterstützt	Nicht unterstützt	Nicht unterstützt	Wird beibehalten	Wird beibehalten
Natives SQLite	Nicht unterstützt	Nicht unterstützt	Nicht unterstützt	Wird beibehalten	Wird beibehalten

Tabellenlegende:

- "Wird beibehalten" bedeutet, dass der Datenbankverbindungstyp (z.B. JDBC, ODBC) bleibt, wie in MapForce definiert.
- "Wird in JDBC konvertiert" bedeutet, dass die Datenbankverbindung in eine Datenbankverbindungs-URL für JDBC konvertiert wird.
- "ADO Bridge" oder "ODBC Bridge" bedeutet, dass der Verbindungsstring übernommen wird, wie in MapForce definiert, dass aber im generierten Code eine passende Klasse verwendet wird, die als ADO Bridge (bzw. ODBC Bridge) fungiert, z.B. **System.Data.OleDb.OleDbConnection** oder **System.Data.Odbc.OdbcConnection**.
- "Benutzerdefiniert" bedeutet, dass Sie die Verbindungsinformationen manuell in das Dialogfeld Datenbankverbindungseinstellungen[286] eingeben müssen, damit die Verbindung im generierten Code funktioniert. Diese Verbindungsinformationen müssen je nach Anwendungsfall unter **ADO/OLE-DB-spezifische Einstellungen** oder unter **JDBC-spezifische Einstellungen** eingegeben werden.

Siehe auch:

- Vorbereiten von Mappings für die Server-Ausführung[1435]
- Kompilieren von Mappings zu MapForce Server-Ausführungsdateien[1442]
- Bereitstellen von Mappings auf FlowForce Server[1445]
- Code Generator[1563]

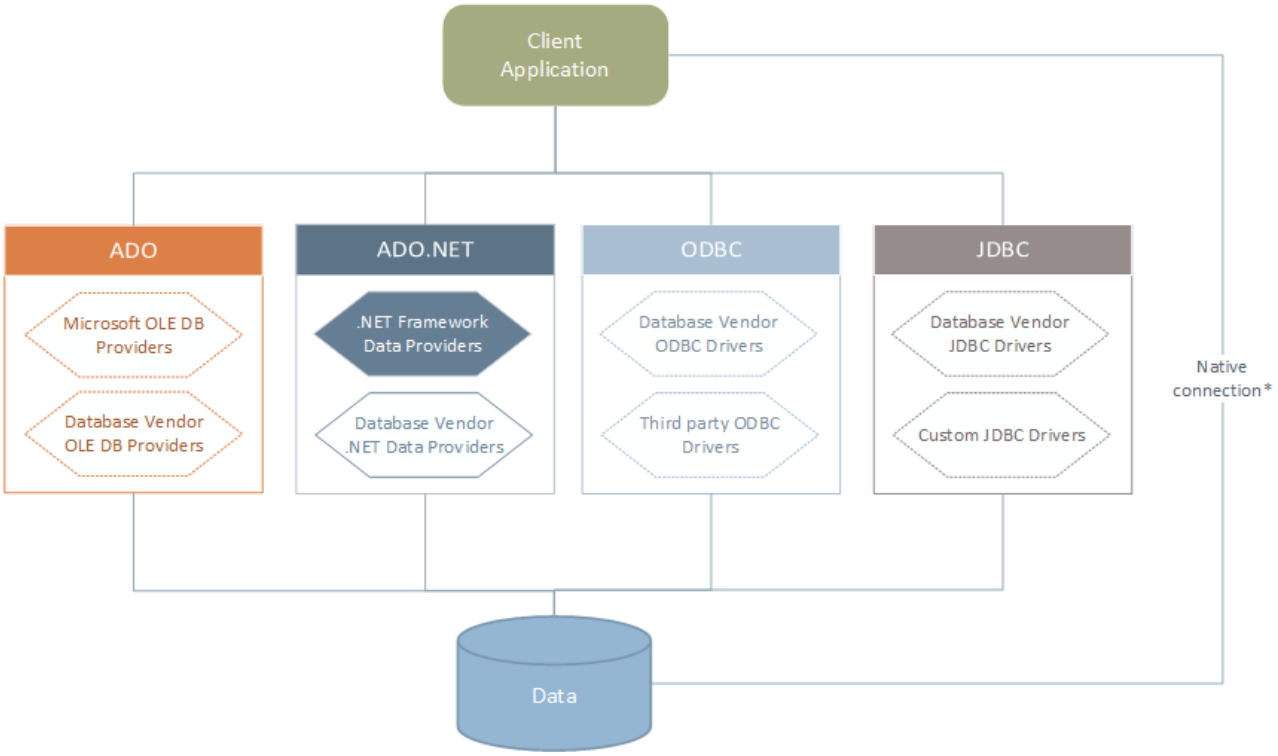
4.2.1 Herstellen einer Verbindung zu einer Datenquelle

Im einfachsten Fall kann es sich bei einer Datenbank um eine lokale Datei wie z.B. eine Microsoft Access- oder SQLite-Datenbankdatei handeln. In komplexeren Szenarien befindet sich die Datenbank manchmal auf einem entfernten Server oder eine Datenbank-Netzwerk-Server, auf dem nicht notwendigerweise dasselbe Betriebssystem wie das der damit verbundenen Applikation verwendet wird. Während z.B. MapForce auf einem Windows-System läuft, könnte die Datenbank, über die Sie die Daten aufrufen möchten (z.B. MySQL), auf einem Linux-Rechner installiert sein.

Für die Verbindung mit verschiedenen Datenbanktypen - sowohl entfernten und lokalen, werden in MapForce Datenverbindungsschnittstellen und Datenbanktreiber verwendet, die auf Ihrem Betriebssystem bereits vorhanden sind oder von denen regelmäßig aktualisierte Versionen von Anbietern gebräuchlicher Datenbanken

bereitgestellt werden. Aufgrund sich ständig weiterentwickelnder Datenbanktechnologien bietet diese Methode bessere plattformübergreifende Flexibilität und Interoperabilität.

Im folgenden Diagramm werden Datenbankverbindungsoptionen zwischen MapForce (als allgemeine Client-Applikation dargestellt) und einem Datenspeicher (einem Datenbank-Server oder einer Datenbankdatei) dargestellt.



** Für SQLite-, PostgreSQL-, CouchDB- und MongoDB-Datenbanken werden direkte native Verbindungen unterstützt. Für die Verbindung mit solchen Datenbanken müssen keine zusätzlichen Treiber auf Ihrem System installiert werden.*

Wie im Diagramm oben gezeigt, kann MapForce zu jeder der gebräuchlichen Datenbankarten über die folgenden Technologien eine Verbindung herstellen:

- ADO (Microsoft® ActiveX® Data Objects), wofür wiederum ein zugrunde liegender OLE DB (Object Linking and Embedding, Database) Provider verwendet wird
- ADO.NET (eine Gruppe von im Microsoft .NET Framework verfügbaren Bibliotheken, die die Interaktion mit Daten ermöglichen)
- JDBC (Java Database Connectivity)
- ODBC (Open Database Connectivity)

Anmerkung: Einige ADO.NET-Anbieter werden nicht oder nur eingeschränkt unterstützt. Siehe Anmerkungen zur Unterstützung von ADO.NET 186.

Datenzugriffstechnologien

Welche Datenverbindungsschnittstelle Sie verwenden sollten, hängt größtenteils von der vorhandenen Software-Infrastruktur ab. Normalerweise werden Sie die Datenzugriffstechnolgie und den Datenbanktreiber verwenden,

 © 2016-2022 Altova GmbH

die enger mit dem gewünschten Datenbanksystem integriert sind. Um z.B. eine Verbindung zu einer Microsoft Access 2013-Datenbank herzustellen, würden Sie einen ADO Connection String erstellen, der einen nativen Provider wie z.B. den **Microsoft Office Access Database Engine OLE DB Provider** verwendet. Um eine Verbindung zu Oracle herzustellen, sollten Sie eventuell die neueste JDBC-, ODBC- oder ADO.NET-Schnittstelle von der Oracle Webseite herunterladen.

Während die Treiber für Windows-Produkte (wie z.B. Microsoft Access oder SQL Server) wahrscheinlich bereits auf Ihrem Windows Betriebssystem vorhanden sind, ist dies bei anderen Datenbanktypen möglicherweise nicht der Fall. Die wichtigsten Datenbankanbieter bringen regelmäßig öffentlich verfügbare Datenbank Client-Software und Treiber heraus, die durch beliebige Kombinationen von OLE DB, ODBC oder JDBC plattformübergreifenden Zugriff auf die jeweilige Datenbank ermöglichen. Zusätzlich dazu steht für jede der oben angeführten Technologien eine Reihe von Treibern von Drittanbietern zur Verfügung. In den meisten Fällen gibt es mehrere Möglichkeiten, um von Ihrem Betriebssystem und somit von MapForce aus, eine Verbindung zur gewünschten Datenbank herzustellen. Welche Funktionalitäten und Performance-Parameter zur Verfügung stehen und welche bekannten Einschränkungen es gibt, hängt normalerweise von der Datenzugriffstechnologie oder den Treibern, die Sie verwenden, ab.

4.2.1.1 Starten des Verbindungsassistenten

MapForce bietet einen Datenbankverbindungsassistenten, der Sie Schritt für Schritt durch das Herstellen einer Verbindung zu einer Datenquelle führt. Bevor Sie den Assistenten aufrufen, denken Sie daran, dass bei einige Datenbanktypen vorher einige Dinge wie z.B. ein Datenbanktreiber oder Datenbank Client-Software separat installiert und konfiguriert werden müssen. Diese erhalten Sie normalerweise vom jeweiligen Datenbankanbieter. Darin enthalten ist die Dokumentation zu Ihrer jeweiligen Windows-Version. Unter Übersicht über Datenbanktreiber 171 finden Sie eine Liste von Datenbanktreibern, gruppiert nach Datenbanktyp.

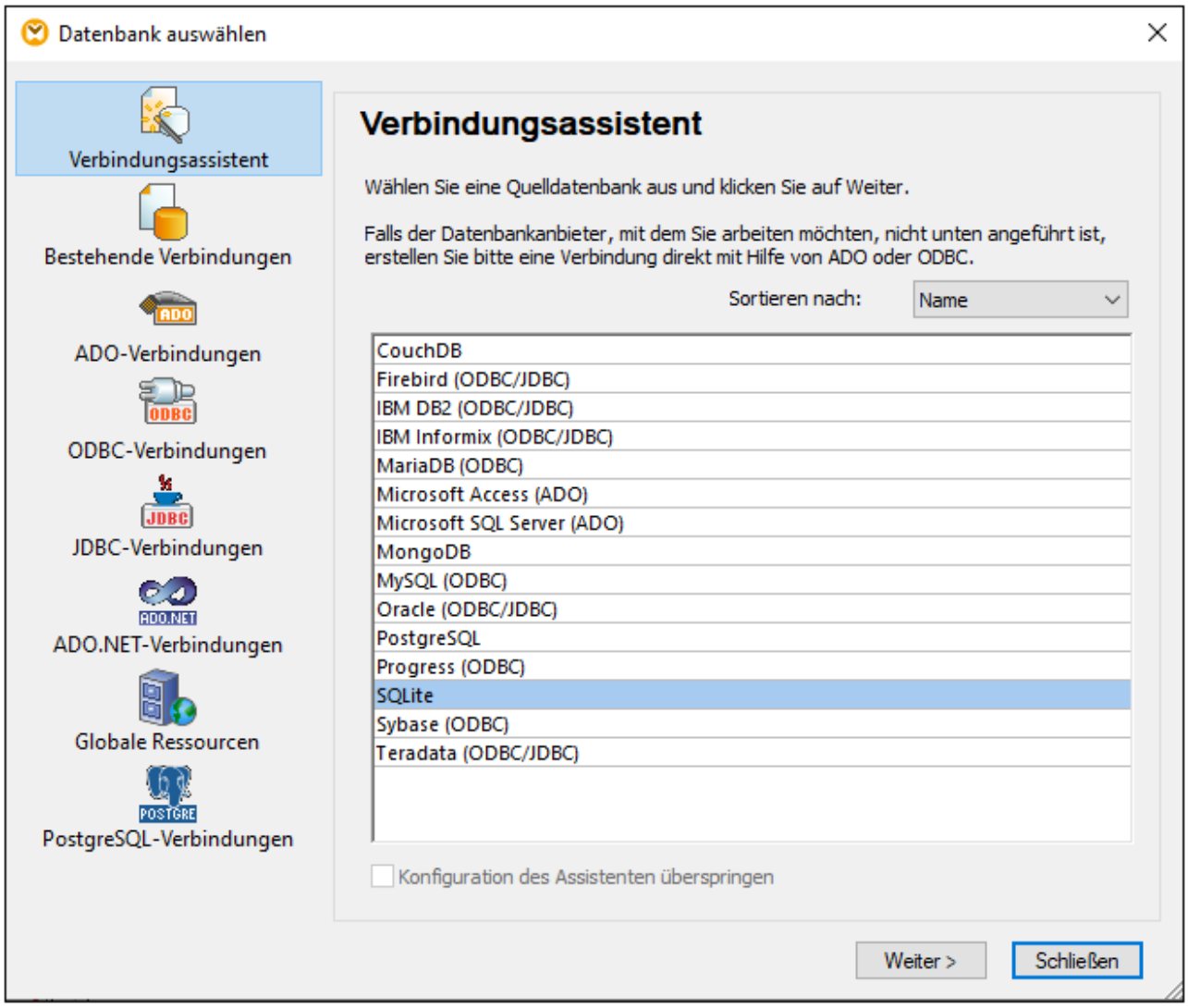
Um den Datenbankverbindungsassistenten (*siehe Abbildung unten*) zu starten, gehen Sie folgendermaßen vor:

- Klicken Sie im Menü **Einfügen** auf **Datenbank**.

Daraufhin wird der Datenbankverbindungsassistent gestartet (*Abbildung unten*). Auf der linken Seite des Fensters können Sie aus den folgenden Verbindungsarten die am besten geeignete auswählen:

- Verbindungsassistent: Hier werden Sie aufgefordert, Ihren Datenbanktyp auszuwählen. Anschließend werden Sie Schritt für Schritt durch den Vorgang zur Herstellung einer Verbindung mit einer Datenbank dieses Typs geführt.
- Auswahl einer bestehenden Verbindung
- Auswahl einer Datenzugriffstechnologie: ADO, ADO.NET, ODBC oder JDBC
- Auswahl einer globalen Altova-Ressource, in der die Datenbankverbindung gespeichert ist
- Native PostgreSQL-Verbindung

Die Datenbanken können im Verbindungsassistent-Fenster (*siehe Abbildung unten*) alphabetisch nach dem Namen des Datenbanktyps oder dem zuletzt verwendeten Datenbanktyp sortiert werden. Wählen Sie die gewünschte Option in der Auswahlliste *Sortieren nach* aus. Klicken Sie nach Auswahl des gewünschten Datenbanktyps auf **Weiter**.



Je nach gewähltem Datenbanktyp, nach gewählter Verbindungstechnologie (ADO, ADO.NET, ODBC, JDBC) und Treiber werden Sie vom Assistenten durch den Verbindungsvorgang geführt. Beispiele zum jeweiligen Datenbanktyp finden Sie im Abschnitt Beispiele für Datenbankverbindungen 200.

Anstelle des Verbindungsassistenten können Sie eine der folgenden Datenbankzugriffstechnologien verwenden:

- Einrichten einer ADO-Verbindung 174
- Einrichten einer ADO.NET-Verbindung 180
- Einrichten einer ODBC-Verbindung 187
- Einrichten einer JDBC-Verbindung 190

 © 2016-2022 Altova GmbH

4.2.1.2 Übersicht über Datenbanktreiber

Die folgende Tabelle enthält eine Liste gebräuchlicher Datenbanktreiber, über die Sie mit Hilfe einer bestimmten Datenzugriffstechnologie eine Verbindung zu einer bestimmten Datenbank herstellen können. Bitte beachten Sie, dass diese Liste keinen Anspruch auf Vollständigkeit erhebt und auch nicht zwingend befolgt werden muss; Sie können neben den unten angeführten Treibern auch andere native Treiber oder Produkte von Drittanbietern verwenden.

Standardmäßig stehen auf Windows Betriebssystemen zwar einige Datenbanktreiber bereits zur Verfügung, Sie müssen eventuell aber dennoch auch andere Treiber herunterladen und verwenden. Bei einigen Datenbanken empfiehlt es sich, anstelle des mit dem Betriebssystem mitgelieferten Treibers den neuesten Treiber des Datenbankanbieters zu verwenden.

Die meisten Datenbankanbieter bieten Treiber entweder als separat herunterladbare Pakete oder mit Datenbank-Client-Software gebündelt an. In letzterem Fall enthält die Datenbank Client-Software normalerweise alle erforderlichen Datenbanktreiber oder gibt Ihnen bei der Installation die Möglichkeit, die gewünschten Treiber und Komponenten auszuwählen. Datenbank Client-Software besteht normalerweise aus Verwaltungs- und Konfigurationstools zur einfacheren Verwaltung und Datenbankanbindung sowie der Dokumentation zum Installieren und Konfigurieren des Datenbank Client und seiner Komponenten.

Damit eine funktionierende Verbindung zur Datenbank hergestellt werden kann, muss der Datenbank Client unbedingt richtig konfiguriert werden. Bevor Sie die Datenbank Client-Software installieren, empfiehlt es sich, vorher die Installations- und Konfigurationsanleitungen zum Datenbank Client sorgfältig zu lesen, da diese normalerweise je nach Datenbank- und Windows-Version unterschiedlich ist.

Um die Möglichkeiten und Einschränkungen der einzelnen Datenzugriffstechnologien im Zusammenhang mit einzelnen Datenbanktypen zu verstehen, lesen Sie die Dokumentation zum jeweiligen Datenbankprodukt und testen Sie die Verbindung in Ihrer jeweiligen Rechnerumgebung. Beachten Sie die folgenden Hinweise und Empfehlungen, um häufige Verbindungsprobleme zu vermeiden:

- Einige ADO.NET-Anbieter werden nicht oder nur eingeschränkt unterstützt. Nähere Informationen dazu finden Sie unter Anmerkungen zur Unterstützung von ADO.NET 186.
- Bei Installation eines Datenbanktreibers wird empfohlen, dass dieser dieselbe Plattform wie die Altova-Applikation hat (32-Bit oder 64-Bit). Wenn Sie z.B. eine 32-Bit-Altova-Applikation auf einem 64-Bit-Betriebssystem verwenden, richten Sie ihre Datenbankverbindung mit der 32-Bit-Treiberversion ein, siehe auch Anzeigen der verfügbaren ODBC-Treiber 189.
- Es empfiehlt sich beim Einrichten einer ODBC-Datenquelle, im Allgemeinen den DSN (Data Source Name) als System-DSN und nicht als Benutzer-DSN zu erstellen. Nähere Informationen dazu finden Sie unter Einrichten einer ODBC-Verbindung 187.
- Wenn es sich bei der Zieldatenbank um eine über ODBC verbundene MySQL- oder MariaDB-Datenbank handelt, muss auf dem Register **Cursor/Results** von MySQL ODBC Connector die Option **Return matched rows instead of affected rows** aktiviert sein. Alternativ dazu können Sie bei manueller Eingabe des Strings über den Datenbankverbindungsassistenten die Option `Option=2` zum Connection String hinzufügen, z.B: `Dsn=mydsn;Option=2;`
- Stellen Sie beim Einrichten einer JDBC-Datenquelle sicher, dass JRE (Java Runtime Environment) installiert ist und dass die CLASSPATH-Umgebungsvariable des Betriebssystems konfiguriert ist. Nähere Informationen dazu finden Sie unter Einrichten einer JDBC-Verbindung 190.
- Nähere Informationen zur Installation und zu den Treibern oder der Datenbank Client-Software eines Datenbankanbieters finden Sie in der Dokumentation zum jeweiligen Installationspaket.

Datenbank	Schnittstelle	Treiber
Firebird	ADO.NET	Firebird ADO.NET-Datenanbieter (https://www.firebirdsql.org/en/additional-downloads/)
	JDBC	Firebird JDBC-Treiber (https://www.firebirdsql.org/en/jdbc-driver/)
	ODBC	Firebird ODBC-Treiber (https://www.firebirdsql.org/en/odbc-driver/)
IBM DB2	ADO	IBM OLE DB-Anbieter für DB2
	ADO.NET	IBM Datenserver-Anbieter für .NET
	JDBC	IBM Datenserver-Treiber für JDBC und SQLJ
	ODBC	IBM DB2 ODBC-Treiber
IBM DB2 for i	ADO	• IBM DB2 für i5/OS IBMDA400 OLE DB-Anbieter • IBM DB2 für i5/OS IBMDARLA OLE DB-Anbieter • IBM DB2 für i5/OS IBMDASQL OLE DB-Anbieter
	ADO.NET	.NET Framework Data Provider für IBM i
	JDBC	IBM Toolbox für Java JDBC-Treiber
	ODBC	iSeries Access ODBC-Treiber
IBM Informix	ADO	IBM Informix OLE DB-Treiber
	JDBC	IBM Informix JDBC-Treiber
	ODBC	IBM Informix ODBC-Treiber
Microsoft Access	ADO	• Microsoft Jet OLE DB-Anbieter • Microsoft Access Database Engine OLE DB-Anbieter
	ADO.NET	.NET Framework-Datenanbieter für OLE DB
	ODBC	• Microsoft Access-Treiber
MariaDB	ADO.NET	Falls kein eigener .NET-Konnektor für MariaDB vorhanden ist, verwenden Sie **Connector/NET** für MySQL (https://dev.mysql.com/downloads/connector/net/).
	JDBC	MariaDB Connector/J (https://downloads.mariadb.org/)
	ODBC	MariaDB Connector/ODBC (https://downloads.mariadb.org/)
Microsoft SQL Server	ADO	• Microsoft OLE DB-Treiber für SQL Server (MSOLEDBSQL) • Microsoft OLE DB-Anbieter für SQL Server (SQLOLEDB) • SQL Server Native Client (SQLNCLI)
	ADO.NET	• .NET Framework-Datenanbieter für SQL Server • .NET Framework-Datenanbieter für OLE DB

© 2016-2022 Altova GmbH

Datenbank	Schnittstelle	Treiber
	JDBC	• Microsoft JDBC-Treiber für SQL Server (https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server)
	ODBC	• ODBC-Treiber für Microsoft SQL Server (https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server)
MySQL	ADO.NET	• Connector/NET (https://dev.mysql.com/downloads/connector/net/)
	JDBC	Connector/J (https://dev.mysql.com/downloads/connector/j/)
	ODBC	Connector/ODBC (https://dev.mysql.com/downloads/connector/odbc/)
Oracle	ADO	• Oracle-Anbieter für OLE DB • Microsoft OLE DB-Anbieter für Oracle
	ADO.NET	Oracle-Datenanbieter für .NET (http://www.oracle.com/technetwork/topics/dotnet/index-085163.html)
	JDBC	• JDBC Thin-Treiber • JDBC Oracle Call Interface (OCI)-Treiber Diese Treiber werden normalerweise während der Installation Ihres Oracle-Datenbank-Client installiert. Stellen Sie die Verbindung über den OCI-Treiber (und nicht den Thin-Treiber) her, wenn Sie die Oracle XML DB-Komponente verwenden.
	ODBC	• Microsoft ODBC für Oracle • Oracle ODBC-Teiber (wird normalerweise während der Installation Ihres Oracle-Datenbank-Client installiert)
PostgreSQL	JDBC	PostgreSQL JDBC-Treiber (https://jdbc.postgresql.org/download.html)
	ODBC	psqlODBC (https://odbc.postgresql.org/)
	Native Verbindung	Verfügbar. Bei Verwendung der nativen Verbindung müssen keine Treiber installiert werden.
Progress OpenEdge	JDBC	JDBC Connector (https://www.progress.com/jdbc/openedge)
	ODBC	ODBC Connector (https://www.progress.com/odbc/openedge)
SQLite	Native Verbindung	Verfügbar. Bei Verwendung der nativen Verbindung müssen keine Treiber installiert werden.
Sybase	ADO	Sybase ASE OLE DB-Anbieter
	JDBC	jConnect™ für JDBC
	ODBC	Sybase ASE ODBC-Treiber
Teradata	ADO.NET	.NET-Datenanbieter für Teradata (https://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata)

Datenbank	Schnittstelle	Treiber
	JDBC	Teradata JDBC-Treiber (https://downloads.teradata.com/download/connectivity/jdbc-driver)
	ODBC	Teradata ODBC-Treiber für Windows (https://downloads.teradata.com/download/connectivity/odbc-driver/windows)

4.2.1.3 ADO-Verbindung

Microsoft ActiveX Data Objects (ADO) ist eine Datenzugriffstechnologie, mit der Sie über OLE DB eine Verbindung zu einer ganzen Reihe von Datenquellen herstellen können. OLE DB ist eine Alternativschnittstelle zu ODBC oder JDBC und ermöglicht einen einheitlichen Zugriff auf Daten in einer COM (Component Object Model)-Umgebung. ADO ist ein Vorläufer des neueren ADO.NET[180] und ist weiterhin eine der möglichen Methoden, um eine Verbindung zu nativen Microsoft-Datenbanken wie Microsoft Access oder SQL Server herzustellen, kann aber auch für andere Datenquellen eingesetzt werden.

Beachten Sie, dass Sie zwischen mehreren ADO-Anbietern wählen können. Einige davon müssen zuerst heruntergeladen und auf Ihrem Rechner installiert werden, bevor Sie sie verwenden können. Für die Verbindung mit SQL Server stehen z.B. die folgenden ADO-Anbieter zur Verfügung:

- Microsoft OLE DB-*Treiber* für SQL Server (MSOLEDBSQL)
- Microsoft OLE DB-*Anbieter* für SQL Server (SQLOLEDB)
- SQL Server Native Client (SQLNCLI)

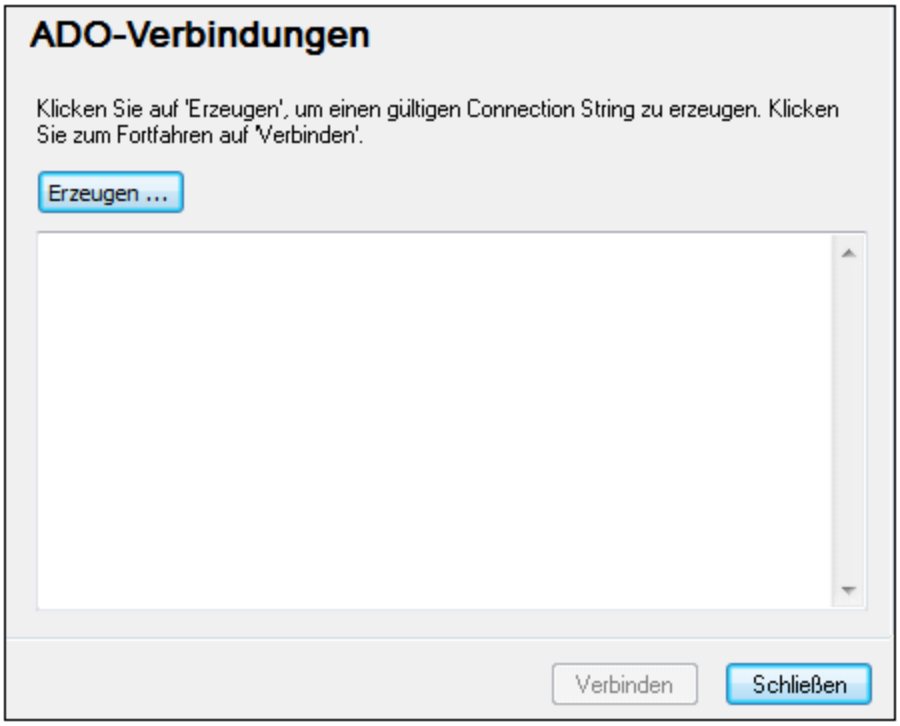
Von den oben aufgelisteten Anbietern wird MSOLEDBSQL empfohlen; Sie können diesen von https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 herunterladen. Beachten Sie, dass er mit der Plattform von MapForce (32-Bit oder 64-Bit) übereinstimmen muss. Die Anbieter SQLOLEDB und SQLNCLI gelten als veraltet und werden daher nicht empfohlen.

> Es ist bekannt, dass es beim **Microsoft OLE DB-Anbieter für SQL Server (SQLOLEDB)** zu Problemen mit der Parameterbindung komplexer Abfragen wie Common Table Expressions (CTE) und verschachtelten SELECT-Anweisungen kommt.

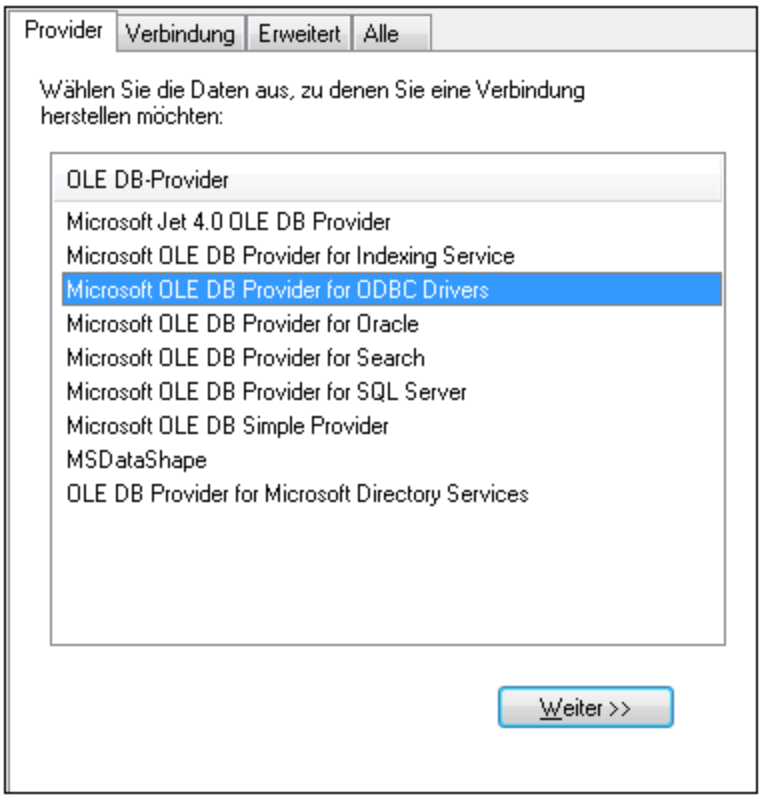
So richten Sie eine ADO-Verbindung ein:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **ADO-Verbindungen**.

© 2016-2022 Altova GmbH



3. Klicken Sie auf **Erzeugen**.



4. Wählen Sie den Daten-Provider, über den Sie die Verbindung erstellen möchten. In der unten stehenden Tabelle sind einige häufige Szenarien aufgelistet.

Zum Verbinden mit dieser Datenbank...	Verwenden Sie diesen Anbieter...
Microsoft Access	• **Microsoft Office Access Database Engine OLE DB-Anbieter** (empfohlen) • **Microsoft Jet OLE DB-Anbieter** Wenn der **Microsoft Office Access Database Engine OLE DB-Anbieter** in der Liste nicht vorhanden ist, überprüfen Sie, ob Sie entweder Microsoft Access oder die Microsoft Access Database Engine Redistributable (https://www.microsoft.com/en-us/download/details.aspx?id=54920) auf Ihrem Rechner installiert haben.
SQL Server	• **Microsoft OLE DB-Treiber für SQL Server (MSOLEDBSQL)** - dies ist der empfohlene OLE DB-Anbieter. Damit dieser Anbieter in der Liste angezeigt wird, muss er von https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 heruntergeladen und installiert werden. • **Microsoft OLE DB-Anbieter für SQL Server (OLEDBSQL)** • **SQL Server Native Client (SQLNCLI)**
Andere Datenbank	Wählen Sie den für Ihre Datenbank benötigten Anbieter aus. Wenn für Ihre Datenbank kein OLE DB-Anbieter zur Verfügung steht, installieren Sie den erforderlichen Treiber des Datenbankanbieters (siehe Übersicht über Datenbanktreiber 171). Alternativ dazu können Sie eine ADO.NET, ODBC- oder JDBC-Verbindung einrichten. Wenn das Betriebssystem über einen ODBC-Treiber für die gewünschte Datenbank verfügt, können Sie auch den **Microsoft OLE DB-Anbieter für ODBC-Treiber** verwenden oder sich vorzugsweise für eine ODBC-Verbindung 187 entscheiden.

5. Klicken Sie nach Auswahl des gewünschten Anbieters auf **Weiter** und stellen Sie den Assistenten fertig.

Die nächsten Schritte im Assistenten hängen vom verwendeten Anbieter ab. Bei SQL Server müssen Sie den Host-Namen des Datenbankservers angeben oder auswählen sowie die Authentifizierungsmethode, den Datenbanknamen und den Datenbank-Benutzernamen und das Passwort dafür. Ein Beispiel dafür finden Sie unter Verbinden mit Microsoft SQL Server (ADO) 222. Bei Microsoft Access müssen Sie zur Datenbankdatei navigieren bzw. den Pfad dafür angeben. Ein Beispiel dafür finden Sie unter Verbinden mit Microsoft Access (ADO) 220.

Auf dem Register **Alle** des Verbindungsdialogfelds finden Sie die vollständige Liste der Initialisierungseigenschaften (Verbindungsparameter). Diese Eigenschaften sind je nach gewähltem Anbieter

© 2016-2022 Altova GmbH

unterschiedlich und müssen eventuell explizit konfiguriert werden, um die Verbindung herstellen zu können. In den folgenden Abschnitten finden Sie eine Anleitung, wie Sie die grundlegenden Initialisierungseigenschaften für Microsoft Access- und SQL Server-Datenbanken konfigurieren:

- Einrichten der SQL Server-Datenverknüpfungseigenschaften 177
- Einrichten der Microsoft Access-Datenverknüpfungseigenschaften 178

4.2.1.3.1 Herstellen einer Verbindung zu einer vorhandenen Microsoft Access-Datenbank

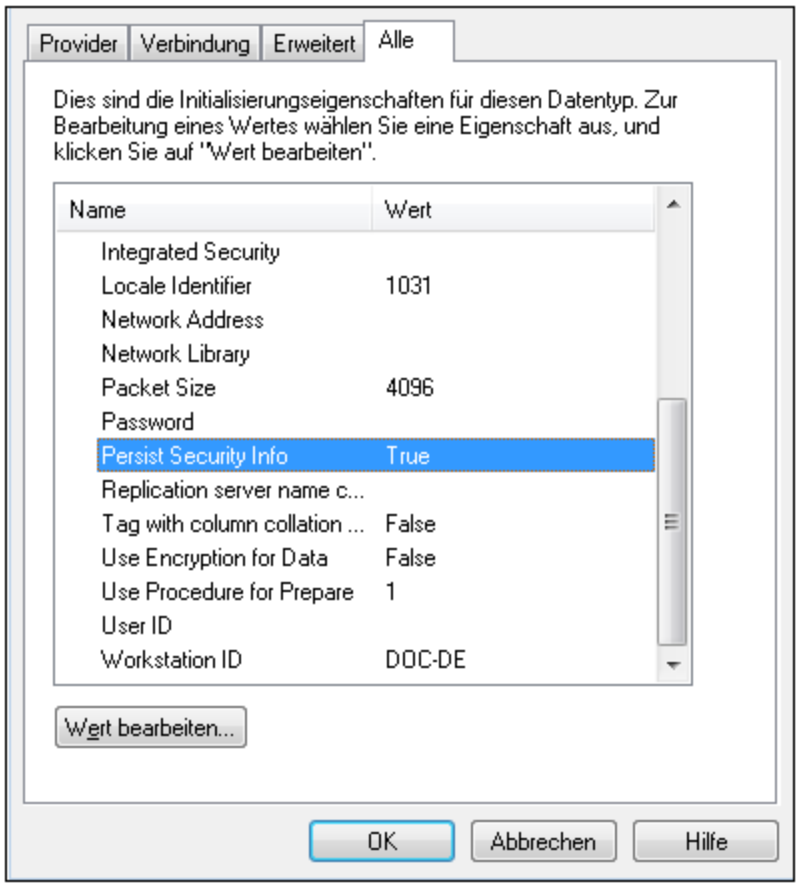
Gehen Sie auf diese Art und Weise vor, wenn Sie eine Verbindung zu einer nicht durch ein Passwort geschützten Microsoft Access-Datenbank herstellen möchten. Wenn die Datenbank passwortgeschützt ist, richten Sie das Datenbankpasswort, wie unter Verbinden mit Microsoft Access (ADO) 220 beschrieben, ein.

So stellen Sie eine Verbindung zu einer vorhandenen Microsoft Access-Datenbank her:

1. Starten Sie den Datenbankverbindungsassistenten (siehe Starten des Datenbankverbindungsassistenten 169).
2. Wählen Sie **Microsoft Access (ADO)** aus und klicken Sie auf **Weiter**.
3. Navigieren Sie zur Datenbankdatei oder geben Sie (entweder den relativen oder den absoluten) Pfad ein.
1. Klicken Sie auf **Verbinden**.

4.2.1.3.2 Einrichten der SQL Server-Datenverknüpfungseigenschaften

Wenn Sie über ADO 174 eine Verbindung zu einer Microsoft SQL Server-Datenbank herstellen, müssen Sie eventuell die folgenden Datenverknüpfungseigenschaften auf dem Register **Alle** des Dialogfelds "Datenverknüpfungseigenschaften" konfigurieren.



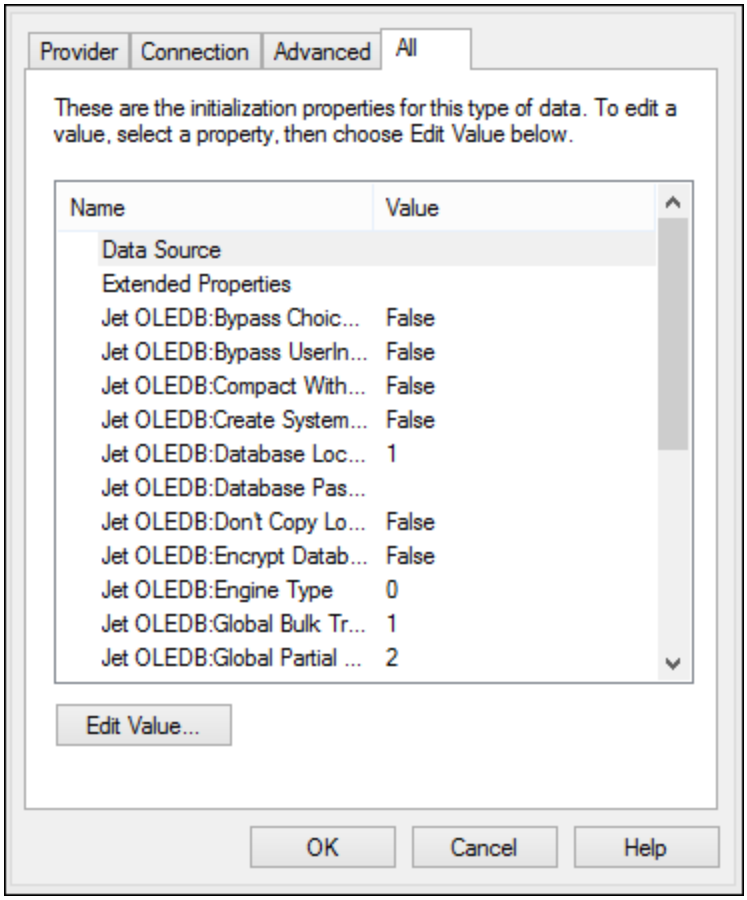
Dialogfeld "Datenverknüpfungseigenschaften"

Eigenschaft	Anmerkungen
Integrated Security	Wenn Sie den Daten-Provider **SQL Server Native Client** auf dem Register **Provider** auswählen, definieren Sie für diese Eigenschaft ein Leerzeichen.
Persist Security Info	Setzen Sie diese Eigenschaft auf **True**.

4.2.1.3.3 Einrichten der Microsoft Access-Datenverknüpfungseigenschaften

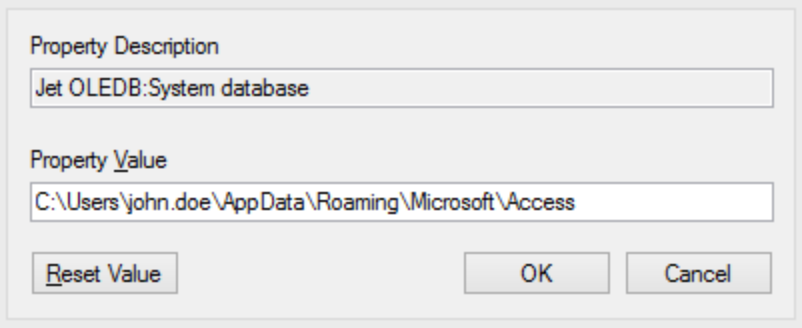
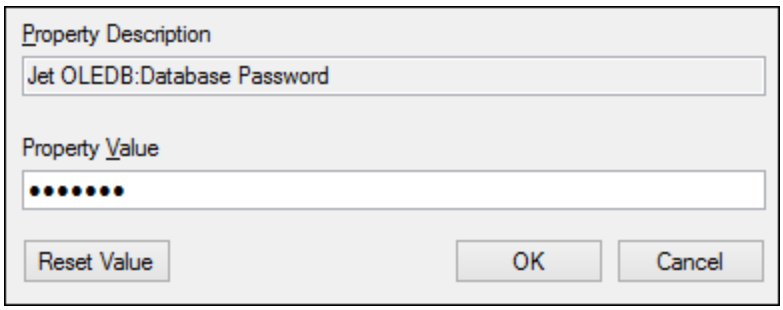
Wenn Sie über ADO[174] eine Verbindung zu einer Microsoft Access-Datenbank herstellen, müssen Sie eventuell die folgenden Datenverknüpfungseigenschaften auf dem Register **Alle** des Dialogfelds "Datenverknüpfungseigenschaften" konfigurieren.

© 2016-2022 Altova GmbH



Dialogfeld "Datenverknüpfungseigenschaften"

Eigenschaft	**Anmerkungen**
Datenquelle	In dieser Eigenschaft ist der Pfad zur Microsoft Access-Datenbankdatei gespeichert. Um Verbindungsprobleme zu vermeiden, wird empfohlen, das UNC (Universal Naming Convention)-Pfadformat zu verwenden, z.B.: `\\anyserver\share$\filepath`
Jet OLEDB:Systemdatenbank	In dieser Eigenschaft ist der Pfad zur Arbeitsgruppen-Informationsdatei gespeichert. Eventuell muss der Wert dieser Eigenschaft explizit definiert werden, bevor Sie eine Verbindung zu einer Microsoft Access-Datenbank herstellen können. Wenn die Verbindung aufgrund eines "Arbeitsgruppen-Informationsdatei"-Fehlers fehlschlägt, suchen Sie die Arbeitsgruppen-Informationsdatei (**System.MDW**) für Ihr Benutzerprofil und setzen Sie den Eigenschaftswert auf den Pfad der Datei **System.MDW**.

	
Jet OLEDB:Datenbankkennwort	Wenn die Datenbank durch ein Passwort geschützt ist, definieren Sie als Wert dieser Eigenschaft das Datenbank-Passwort. 

4.2.1.4 ADO.NET-Verbindung

ADO.NET ist eine Gruppe von Microsoft .NET Framework-Bibliotheken für die Interaktion mit Daten, darunter auch mit Daten aus Datenbanken. Für die Verbindung zu einer Datenbank von MapForce aus über ADO.NET wird Microsoft .NET Framework 4 oder höher benötigt. Wie unten gezeigt, erfolgt die Verbindung zu einer Datenbank über ADO.NET durch Auswahl eines .NET-Anbieters und Bereitstellung eines Connection String.

Ein .NET-Datenanbieter ist eine Sammlung von Klassen, mit Hilfe derer Sie eine Verbindung zu einem bestimmten Datenquellentyp (z.B. einem SQL Server oder einer Oracle-Datenbank) herstellen können, Befehle daran ausführen und Daten aus dieser Quelle abrufen können, d.h. mit Hilfe von ADO.NET kann eine Applikation wie MapForce über einen Datenanbieter mit einer Datenbank kommunizieren. Jeder Datenanbieter ist für den spezifischen Datenquellentyp, für den er entwickelt wurde, optimiert. Es gibt zwei Arten von .NET-Anbietern:

1. Solche, die standardmäßig mit Microsoft .NET Framework bereitgestellt werden.
2. Solche, die von Anbietern gebräuchlicher Datenbanken als Erweiterung zum .NET Framework bereitgestellt werden. ADO.NET-Anbieter dieser Art müssen separat installiert werden und können normalerweise von der Website des entsprechenden Datenbankanbieters heruntergeladen werden.

Anmerkung: Einige ADO.NET-Anbieter werden nicht oder nur eingeschränkt unterstützt. Siehe Anmerkungen zur Unterstützung von ADO.NET 186.

 © 2016-2022 Altova GmbH

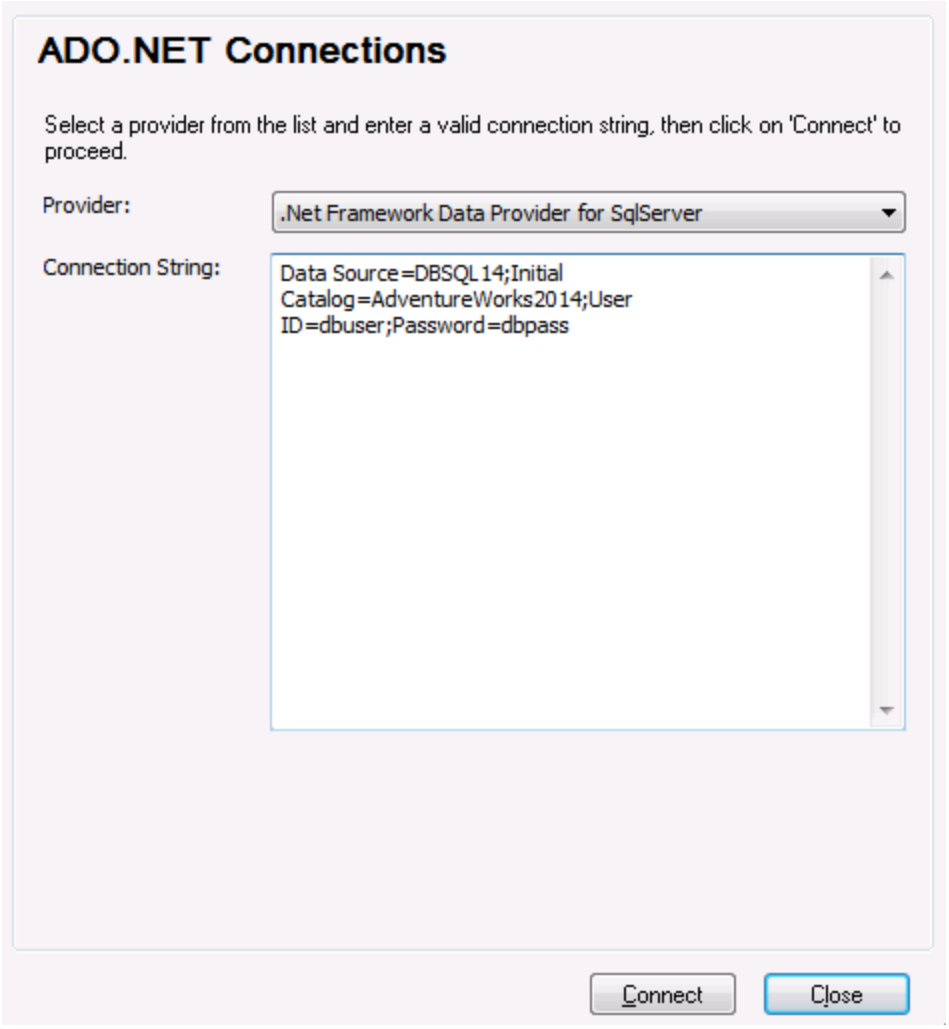
So richten Sie eine ADO.NET-Verbindung ein:

1. Starten Sie den Datenbankverbindungsassistenten[169].
2. Klicken Sie auf **ADO.NET-Verbindungen**.
3. Wählen Sie einen .NET-Datenanbieter aus der Liste aus.

   Die Liste der Anbieter, die standardmäßig mit dem .NET Framework zur Verfügung stehen, wird in der Liste der Anbieter angezeigt. Anbieterspezifische .NET-Datenanbieter stehen in der Liste nur zur Verfügung, wenn sie bereits auf Ihrem System installiert sind. Damit anbieterspezifische .NET-Anbieter zur Verfügung stehen, müssen diese durch Ausführung der vom Datenbankanbieter bereitgestellten .msi- oder .exe-Datei im GAC (Global Assembly Cache) installiert werden.

4. Geben Sie einen Datenbank-Connection String ein. Mit einem Connection String werden die Datenbankverbindungsinformationen in Form von durch Semikola getrennte Schlüssel/Wert-Paare von Verbindungsparametern definiert. Mit einem Connection String wie **`Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;Password=dbpass`** wird z.B. eine Verbindung zur SQL Server-Datenbank `ProductsDB` auf dem Server `DBSQLSERV` unter dem Benutzernamen `dbuser` und mit dem Passwort `dbpass` hergestellt. Sie können einen Connection String erstellen, indem Sie die Schlüssel/Wert-Paare direkt in das Dialogfeld "Connection String" eingeben. Eine weitere Methode ist die Erstellung über Visual Studio (siehe Erstellen eines Connection String in Visual Studio[182]).

   Die Syntax des Connection String ist von dem in der Liste "Anbieter" ausgewählten Anbieter abhängig. Beispiele dazu finden Sie unter Beispiele für ADO.NET Connection Strings[185].



5. Klicken Sie auf **Verbinden**.

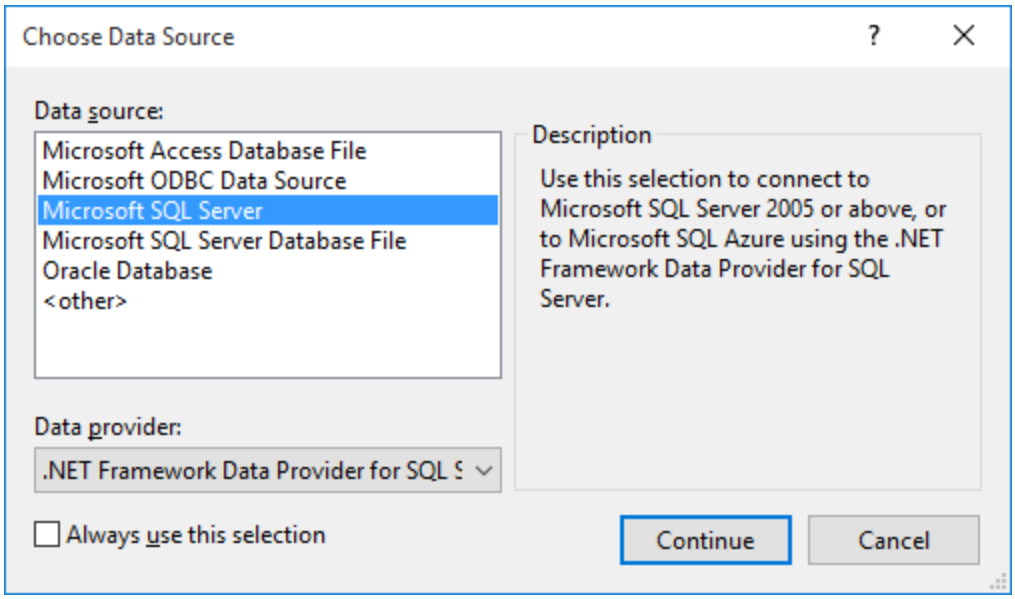
4.2.1.4.1 Erstellen eines Connection String in Visual Studio

Um über ADO.NET eine Verbindung zu einer Datenquelle herstellen zu können, wird ein gültiger Datenbank Connection String benötigt. Im Folgenden wird beschrieben, wie Sie über Visual Studio einen Connection String erstellen.

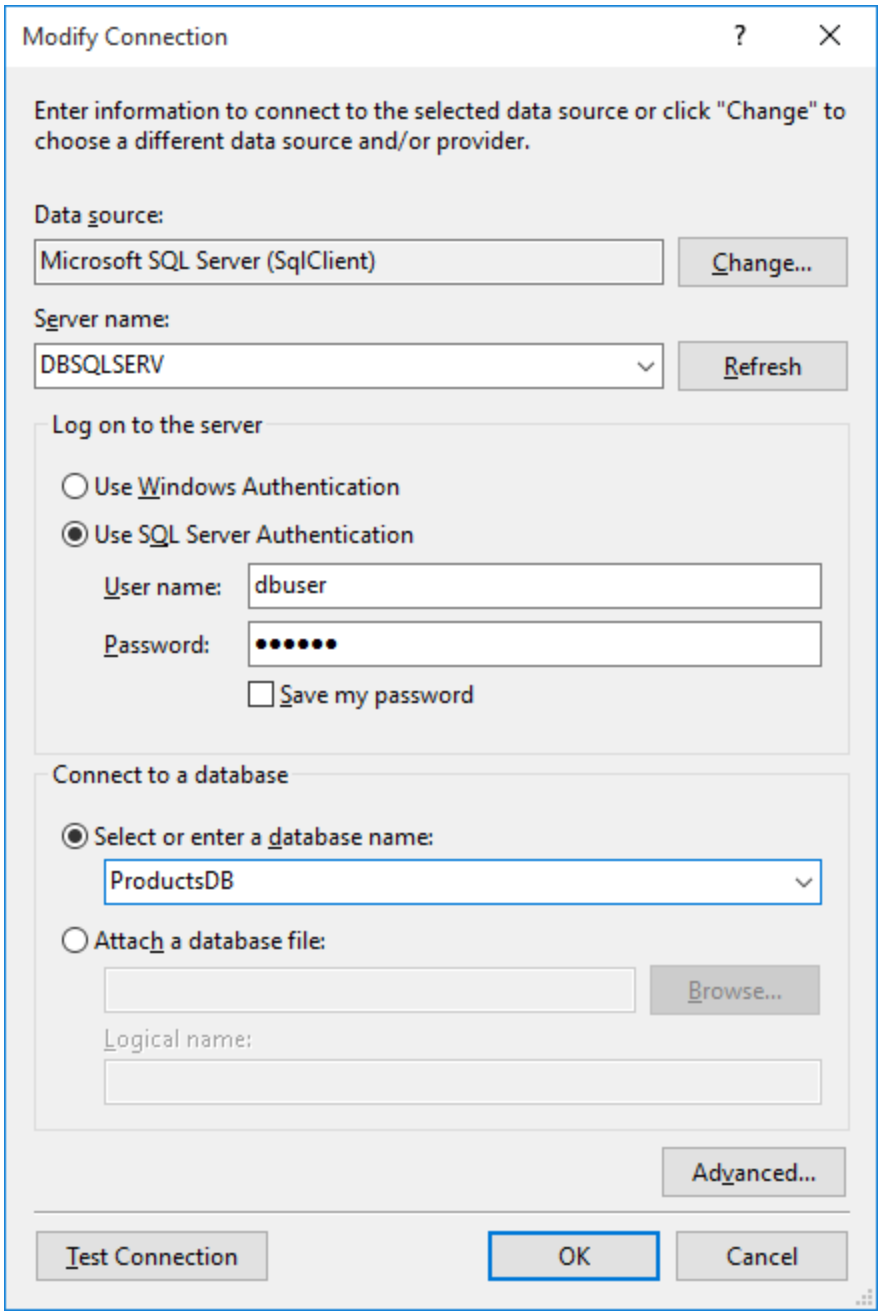
So erstellen Sie einen Connection String in Visual Studio:

1. Klicken Sie im Menü **Extras** auf **Mit Datenbank verbinden**.
2. Wählen Sie eine Datenquelle aus der Liste aus (in diesem Beispiel Microsoft SQL Server). Der Datenanbieter wird auf Basis Ihrer Auswahl automatisch ausgefüllt.

© 2016-2022 Altova GmbH



3. Klicken Sie auf **Weiter**.



4. Geben Sie den Server Host-Namen und den Benutzernamen und das Passwort für die Datenbank ein. In diesem Beispiel stellen wir unter Verwendung der SQL Server-Authentifizierung eine Verbindung zur Datenbank `ProductsDB` auf dem Server `DBSQLSERV` her.
5. Klicken Sie auf **OK**.

Wenn die Datenbankverbindung erfolgreich hergestellt wurde, wird sie im Server Explorer-Fenster angezeigt. Sie können das Server-Explorer-Fenster mit dem Menübefehl **Ansicht | Server-Explorer** aufrufen. Um den Datenbank Connection String anzuzeigen, klicken Sie im Server-Explorer-Fenster mit der rechten Maustaste auf die Verbindung und wählen Sie **Eigenschaften**. Der Connection String wird nun im Fenster "Eigenschaften"

© 2016-2022 Altova GmbH

von Visual Studio angezeigt. Beachten Sie, dass Sie das Sternchen (*) vor dem Einfügen des String in das Feld "Connection String" von MapForce durch das tatsächliche Passwort ersetzen müssen.

4.2.1.4.2 Beispiele für ADO.NET Connection Strings

Um eine ADO.NET-Verbindung einzurichten, müssen Sie einen ADO.NET-Anbieter aus dem Datenbankverbindungsdialogfeld auswählen und einen Connection String eingeben (siehe auch Einrichten einer ADO.NET-Verbindung 180). Beispiele für ADO.NET Connection Strings für verschiedene Datenbanken sind unter dem jeweiligen .NET-Anbieter aufgelistet.

.NET Data Provider für Teradata

Dieser Anbieter kann von der Teradata-Website (https://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata) heruntergeladen werden. Ein Beispiel-Connection String sieht folgendermaßen aus:

```
Data Source=ServerAddress;User Id=user;Password=password;
```

.NET Framework Data Provider für IBM i

Dieser Anbieter wird im Rahmen von *IBM i Access Client Solutions - Windows Application Package*. installiert. Ein Connection String sieht folgendermaßen aus:

```
DataSource=Serveradresse;UserID=Benutzer;Password=Passwort;DataCompression=True;
```

Nähere Informationen dazu finden Sie in der im obigen Installationspaket inkludierten Hilfedatei ".NET Provider Technical Reference".

.NET Framework Data Provider für MySQL

Dieser Anbieter kann von der MySQL Website (https://dev.mysql.com/downloads/connector/net/) heruntergeladen werden. Ein Beispiel-Connection String sieht folgendermaßen aus:

```
Server=127.0.0.1;Uid=root;Pwd=12345;Database=test;
```

Siehe auch: https://dev.mysql.com/doc/connector-net/en/connector-net-programming-connecting-connection-string.html

.NET Framework Data Provider für SQL Server

Ein Beispiel-Connection String sieht folgendermaßen aus:

```
Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;Password=dbpass
```

Siehe auch: https://msdn.microsoft.com/en-us/library/ms254500(v=vs.110).aspx

IBM DB2 Data Provider 10.1.2 für .NET Framework 4.0

```
Database=PRODUCTS;UID=user;Password=password;Server=localhost:50000;
```

Anmerkung: Dieser Anbieter wird normalerweise mit dem IBM DB2 Data Server Client-Paket installiert. Wenn der Anbieter nach Installation des IBM DB2 Date Server Client-Pakets nicht in der Liste der ADO.NET-Anbieter aufgelistet wird, lesen Sie nach unter: https://www-01.ibm.com/support/docview.wss?uid=swg21429586.

Siehe auch:
https://www.ibm.com/support/knowledgecenter/en/SSEPGG_10.1.0/com.ibm.swg.im.dbclient.adonet.ref.doc/doc/DB2ConnectionClassConnectionStringProperty.html

Oracle Data Provider für .NET (ODP.NET)

Das Installationspaket, das den ODP.NET-Anbieter enthält, kann von der Oracle Website heruntergeladen werden (siehe http://www.oracle.com/technetwork/topics/dotnet/downloads/index.html). Ein Beispiel-Connection String sieht folgendermaßen aus:

```
Data Source=DSORCL;User Id=user;Password=password;
```

`DSORCL` ist hierbei der Name der Datenquelle, der auf einen in der Datei **tnsnames.ora** definierten Oracle-Dienstnamen verweist. Eine Beschreibung dazu finden Sie unter Verbinden mit Oracle (ODBC) 235.

Um eine Verbindung herzustellen, ohne einen Dienstnamen in der Datei **tnsnames.ora** zu konfigurieren, verwenden Sie einen String wie den folgenden:

```
Data Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host)(PORT=port)))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=MyOracleSID)));User
Id=user;Password=password;
```

Siehe auch: https://docs.oracle.com/cd/B28359_01/win.111/b28375/featConnecting.htm

4.2.1.4.3 Anmerkungen zur Unterstützung von ADO.NET

In der folgenden Tabelle sind bekannte ADO.NET-Datenbanktreiber aufgelistet, die derzeit in MapForce nicht oder nur eingeschränkt unterstützt werden.

Datenbank	Treiber	Anmerkungen zur Unterstützung
Alle Datenbanken	**.Net Framework Data Provider for ODBC**	Eingeschränkte Unterstützung. Bekannte Probleme bei Microsoft Access-Verbindungen. Es wird empfohlen stattdessen direkte ODBC-Verbindungen zu verwenden.

© 2016-2022 Altova GmbH

Datenbank	Treiber	Anmerkungen zur Unterstützung
	.Net Framework Data Provider for OleDb	Eingeschränkte Unterstützung. Bekannte Probleme bei Microsoft Access-Verbindungen. Es wird empfohlen stattdessen direkte ADO-Verbindungen zu verwenden.
Firebird	**Firebird ADO.NET Data Provider**	Eingeschränkte Unterstützung. Es wird empfohlen stattdessen ODBC oder JDBC zu verwenden.
Informix	**IBM Informix Data Provider for .NET Framework 4.0**	Wird nicht unterstützt. Verwenden Sie stattdessen **DB2 Data Server Provider**.
IBM DB2 for i (iSeries)	**.Net Framework Data Provider for i5/OS**	Wird nicht unterstützt. Verwenden Sie stattdessen den im Rahmen des *IBM i Access Client Solutions - Windows Application-Pakets* bereitgestellten **.Net Framework Data Provider for IBM i-**Treiber.
Oracle	**.Net Framework Data Provider for Oracle**	Eingeschränkte Unterstützung. Der Treiber wird zwar mit dem .NET Framework zur Verfügung gestellt, doch wird von Microsoft von der Verwendung abgeraten, da er veraltet ist.
PostgreSQL	**-**	Es werden keine ADO.NET-Treiber für diesen Anbieter unterstützt. Verwenden Sie stattdessen eine native Verbindung.
Sybase	**-**	Es werden keine ADO.NET-Treiber für diesen Anbieter unterstützt.

4.2.1.5 ODBC-Verbindung

ODBC (Open Database Connectivity) ist eine häufig verwendete Datenzugriffstechnologie, mit der Sie von MapForce aus eine Verbindung zu einer Datenbank herstellen können. ODBC kann entweder als primäre Verbindungsmethode oder als Alternative zu nativen Verbindungen oder Verbindungen über OLE DB oder JDBC verwendet werden.

Um über ODBC eine Datenbankverbindung herzustellen, müssen Sie zuerst einen ODBC-Datenquellennamen (DSN = Data Source Name) auf dem Betriebssystem erstellen. Wenn bereits ein DSN erstellt wurde - möglicherweise von einem anderen Benutzer auf dem Betriebssystem - entfällt dieser Schritt. Der DSN bietet eine einheitliche Methode, um die Datenbankverbindung für jede ODBC-fähige Client-Applikation auf dem Betriebssystem einschließlich MapForce zu beschreiben. Es gibt folgende Arten von DSN:

- System-DSN
- Benutzer-DSN
- Datei-DSN

Eine *Systemdatenquelle* kann von allen Benutzern mit Rechten auf dem Betriebssystem aufgerufen werden. Eine *Benutzerdatenquelle* steht nur dem Benutzer, der sie erstellt hat, zur Verfügung. Wenn Sie einen *Datei-DSN* erstellen, wird die Datenquelle als Datei mit der Erweiterung .dsn erstellt, die Sie gemeinsam mit anderen Benutzern verwenden können, vorausgesetzt diese haben die für die Datenquelle erforderlichen Treiber installiert.

Alle auf Ihrem Rechner bereits verfügbaren DSNs werden im Dialogfeld "Datenbankverbindung" aufgelistet, wenn Sie im Dialogfeld "ODBC-Verbindungen" auf **ODBC-Verbindungen** klicken.



Dialogfeld "ODBC-Verbindungen"

Wenn zur gewünschten Datenbank kein DSN vorhanden ist, hilft Ihnen der MapForce Datenbank-Verbindungsassistent dabei, einen zu erstellen; Sie können den DSN aber auch direkt in Ihrem Windows-Betriebssystem erstellen. Stellen Sie in jedem Fall, bevor Sie fortfahren, sicher, dass der für die Datenbank erforderliche ODBC-Treiber in der Liste der ODBC-Treiber zur Verfügung steht (siehe Anzeigen der verfügbaren ODBC-Treiber 189).

So stellen Sie mit Hilfe eines neuen DSN eine Verbindung her:

1. Starten Sie den Datenbank-Verbindungsassistenten 169.
2. Klicken Sie im Dialogfeld "Datenbankverbindung" auf **ODBC-Verbindungen**.
3. Wählen Sie einen Datenquellentyp aus (Benutzer-DSN, System-DSN, Datei-DSN).

> Zur Erstellung eines System-DSN benötigen Sie Administratorrechte auf Ihrem Betriebssystem und MapForce muss von Ihnen als Adminstrator ausgeführt werden.

© 2016-2022 Altova GmbH

4. Klicken Sie auf **Hinzufügen** .
5. Wählen Sie einen Treiber aus und klicken Sie anschließend auf **Benutzer-DSN** oder **System-DSN** (je nachdem, welche Art von DSN Sie erstellen möchten). Wenn der Treiber für Ihre Datenbank nicht aufgelistet ist, laden Sie ihn vom Datenbankanbieter herunter und installieren Sie ihn (siehe Übersicht über Datenbanktreiber 171).
6. Füllen Sie im Dialogfeld, das daraufhin angezeigt wird, alle treiberspezifischen Informationen aus, um die Verbindung fertig zu konfigurieren.

Damit eine Verbindung hergestellt werden kann, müssen Sie den Host-Namen (oder die IP-Adresse) des Datenbankservers sowie den Datenbank-Benutzernamen und das Passwort dafür angeben. Eventuell gibt es weitere optionale je nach Anbieter unterschiedliche Verbindungsparameter. Nähere Informationen zu den Parametern für die einzelnen Verbindungsmethoden finden Sie in der Dokumentation des Treiberanbieters. Sobald der DSN erstellt wurde, steht er in der Liste der Datenquellennamen zur Verfügung. Auf diese Art können Sie die Datenbankverbindungsinformationen jedes Mal, wenn Sie eine Verbindung zur Datenbank herstellen, wiederverwenden. Beachten Sie, dass Benutzer-DSNs zur Liste der Benutzer-DSNs hinzugefügt werden, während System-DSNs zur Liste der System-DSNs hinzugefügt werden.

So stellen Sie über einen vorhandenen DSN eine Verbindung her:

1. Starten Sie den Datenbank-Verbindungsassistenten 169.
2. Klicken Sie auf **ODBC-Verbindungen**.
3. Wählen Sie einen Datenquellentyp aus (Benutzer-DSN, System-DSN, Datei-DSN).
4. Klicken Sie auf den vorhandenen DSN-Eintrag und anschließend auf **Verbinden.**

So erzeugen Sie auf Basis einer vorhandenen .dsn-Datei einen Connection String:

1. Starten Sie den Datenbank-Verbindungsassistenten 169.
2. Klicken Sie auf **ODBC-Verbindungen**.
3. Wählen Sie **Connection String erstellen** und klicken Sie anschließend auf **Erzeugen**.
4. Wenn Sie den Connection String mit Hilfe eines Datei-DSN erstellen möchten, klicken Sie auf das Register **Dateidatenquelle**. Klicken Sie andernfalls auf das Register **Computerdatenquelle**. (System-DSNs und Benutzer-DSNs werden als "Computerdatenquelle" bezeichnet.)
5. Wählen sie die benötigte .dsn-Datei aus und klicken Sie auf **OK.**

So stellen Sie die Verbindung mittels eines bereits definierten Connection String her:

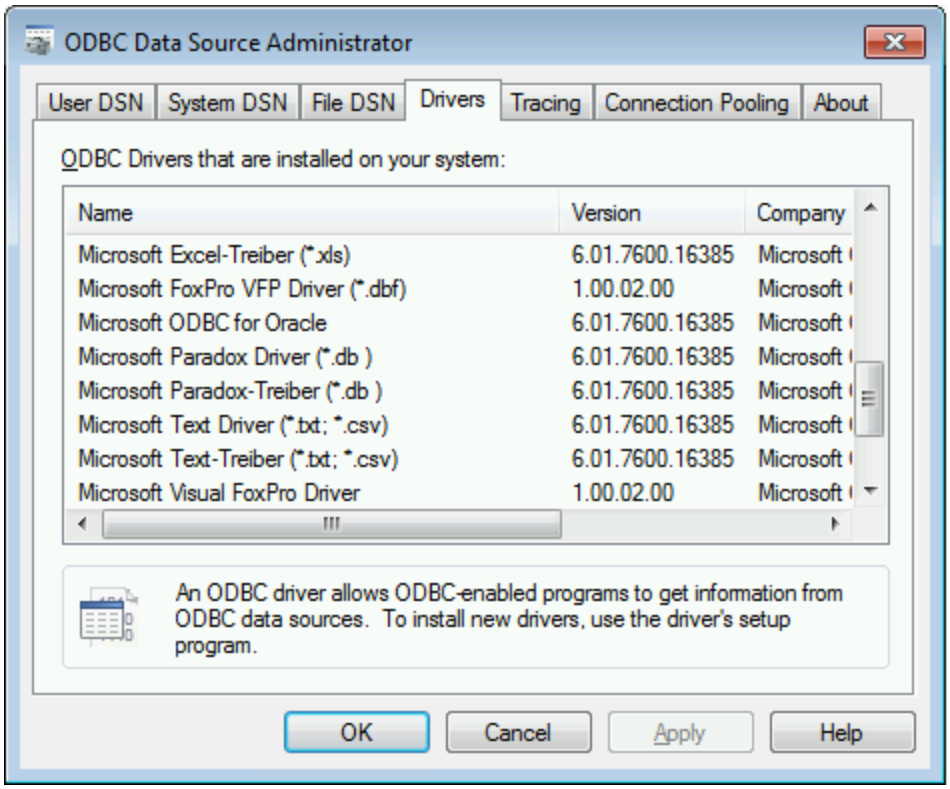
1. Starten Sie den Datenbank-Verbindungsassistenten 169.
2. Klicken Sie auf **ODBC-Verbindungen**.
3. Wählen Sie **Connection String erstellen-**
4. Kopieren Sie den Connection String in das entsprechende Feld ein und klicken Sie auf **Verbinden**.

4.2.1.5.1 Verfügbare ODBC-Treiber

Im ODBC-Datenquellen-Administrator können Sie die auf Ihrem Betriebssystem verfügbaren ODBC-Treiber anzeigen. Sie können den ODBC-Datenquellen-Administrator (**Odbcad32.exe**) über die Windows-Systemsteuerung unter **Verwaltung** aufrufen. Auf 64-Bit-Betriebssystemen gibt es zwei Versionen dieser ausführbaren Datei:

- Die 32-Bit-Version der Datei **Odbcad32.exe** befindet sich im Verzeichnis **C:\Windows\SysWoW64** (wenn **C:** Ihr Systemlaufwerk ist).
- Die 64-Bit-Version der Datei **Odbcad32.exe** befindet sich im Verzeichnis **C:\Windows\System32**.

Die installierten 32-Bit-Datenbanktreiber sind in der 32-Bit-Version des ODBC-Datenquellen-Administrators zu sehen, während die 64-Bit-Treiber in der 64-Bit-Version angezeigt werden. Vergewissern Sie sich daher, dass Sie die richtige Version des ODBC-Datenquellen-Administrators geöffnet haben, wenn Sie die Datenbanktreiber überprüfen.



ODBC-Datenquellen-Administrator

Wenn der Treiber für die gewünschte Datenbank in der Liste nicht vorhanden ist oder wenn Sie eine anderen Treiber hinzufügen möchten, müssen Sie diesen von der Webseite des Datenbankanbieters herunterladen (siehe Übersicht über Datenbanktreiber 171). Sobald der ODBC-Treiber auf Ihrem System verfügbar ist, können Sie damit ODBC-Verbindungen herstellen (siehe Einrichten der ODBC-Verbindung 187).

4.2.1.6 JDBC-Verbindung

JDBC (Java Database Connectivity) ist eine Datenbankzugriffsschnittstelle, die Teil der Java-Software-Plattform von Oracle ist. JDBC-Verbindungen beanspruchen im Allgemeinen mehr Ressourcen als ODBC-Verbindungen, bieten aber Funktionen, die über ODBC nicht zur Verfügung stehen.

Voraussetzungen

- JRE (Java Runtime Environment) oder Java Development Kit (JDK) muss installiert sein. Dabei muss es sich entweder um Oracle JDK oder einen Open Source Build wie Oracle OpenJDK

 © 2016-2022 Altova GmbH

handeln.MapForce ermittelt den Pfad zur Java Virtual Machine (JVM) anhand der folgenden Ordner und zwar in folgender Reihenfolge: a) anhand des benutzerdefinierten JVM-Pfads, den Sie eventuell in den **Applikationsoptionen** definiert haben, siehe Java-Einstellungen[1542]; b) anhand des JVM-Pfads in der Windows Registry; c) anhand der JAVA_HOME-Umgebungsvariablen.

- Stellen Sie sicher, dass die Plattform von MapForce (32-Bit, 64-Bit) mit der des JRE/JDK übereinstimmt.
- Die JDBC-Treiber des Datenbankanbieters müssen installiert sein. Dabei kann es sich um JDBC-Treiber, die im Rahmen der Datenbankclient-Installation installiert wurden oder um separat heruntergeladene JDBC-Bibliotheken (.jar-Dateien) (falls verfügbar und von der Datenbank unterstützt) handeln, siehe auch Beispiele für Datenbankverbindungen[200].
- Die CLASSPATH-Umgebungsvariable muss den Pfad zum JDBC-Treiber auf Ihrem Windows-Betriebssystem enthalten. Diese Variable wird unter Umständen bei der Installation einiger Datenbank Clients automatisch konfiguriert, siehe auch Konfigurieren des CLASSPATH[193].

Verbinden mit SQL Server über JDBC mit Windows-Anmeldeinformationen

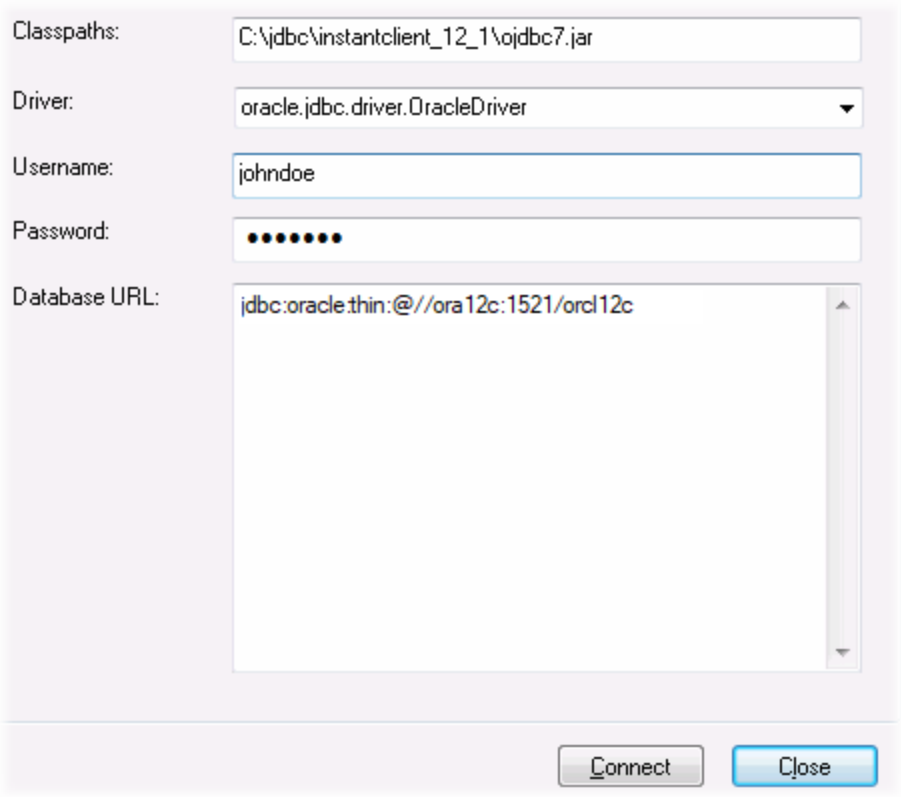
Wenn Sie über JDBC mit Windows-Anmeldeinformationen (integrierte Sicherheit) eine Verbindung zu SQL Server herstellen, beachten Sie die folgenden Punkte:

- Die im JDBC-Treiberpaket enthaltene Datei **sqljdbc_auth.dll** muss in ein Verzeichnis kopiert werden, das sich in der System PATH-Umgebungsvariablen befindet. Es gibt zwei solche Dateien, eine für die x86- und eine für die x64-Plattform. Vergewissern Sie sich, dass Sie die Ihrer JDK-Plattform entsprechende zum PATH hinzufügen. Außerdem muss MapForce (bzw. gegebenenfalls das Programm, mit dem das Mapping ausgeführt wird) nach Änderung der Umgebungsvariablen neu gestartet werden.
- Der JDBC Connection String muss die Eigenschaft **`integratedSecurity=true`** enthalten. Sie können diese Eigenschaft von verschiedenen Stellen aus hinzufügen:
  - über den Datenbankverbindungsassistenten, siehe unten
  - über die Datenbank-Komponenteneinstellungen[286]
  - ggf. durch Bearbeitung des Datenbank-Connection String im generierten Java-Code[1580].

Nähere Informationen dazu finden Sie in der *Microsoft-Dokumentation zu JDBC-Treibern für SQL Server* unter https://docs.microsoft.com/de-de/sql/connect/jdbc/building-the-connection-url?view=sql-server-ver15.

Einrichten einer JDBC-Verbindung

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **JDBC-Verbindungen**.
3. Geben Sie optional in das Textfeld "Classpaths" eine durch Semikola getrennte Liste von .jar-Dateipfaden ein. Die hier eingegebenen .jar-Bibliotheken werden zusätzlich zu den bereits in der Umgebungsvariablen CLASSPATH definierten in die Umgebung geladen. Nachdem Sie Ihre Eingaben ins Textfeld "Classpaths" beendet haben, werden alle in den .jar-Quellbibliotheken gefundenen JDBC-Treiber automatisch zur Liste "Treiber" (siehe nächster Schritt) hinzugefügt.



4. Wählen Sie neben "Treiber" einen JDBC-Treiber aus der Liste aus oder geben Sie einen Java-Klassennamen ein. Beachten Sie, dass diese Liste alle über die Umgebungsvariable CLASSPATH konfigurierten JDBC-Treiber (siehe Konfigurieren des CLASSPATH 193) sowie die im Textfeld "Classpaths" gefundenen JDBC-Treiber enthält.

> Die in der CLASSPATH-Variablen definierten JDBC-Treiberpfade sowie alle direkt in das Datenbankverbindungsdialogfeld eingegebenen Pfade zu .jar-Dateien werden alle der Java Virtual Machine (JVM) zur Verfügung gestellt. Die JVM entscheidet anschließend, welche Treiber zur Herstellung einer Verbindung verwendet werden sollen. Es wird empfohlen, die in die JVM geladenen Java-Klassen im Auge zu behalten, damit es zu keinen potenziellen JDBC-Treiberkonflikten und unerwarteten Ergebnissen bei der Herstellung der Datenbankverbindung kommt.

5. Geben Sie den Benutzernamen und das Passwort für die Datenbank in die entsprechenden Felder ein.
6. Geben Sie im Textfeld "Datenbank-URL" die JDBC Connection-URL (String) im datenbanktypspezifischen Format ein. In der folgenden Tabelle sehen Sie die Syntaxvorgaben für die JDBC Connection-URLs (Strings) für gebräuchliche Datenbanktypen.

Datenbank	JDBC-Verbindungs-URL
Firebird	`jdbc:firebirdsql://<host>[:<port>]/<database path or alias>`
IBM DB2	`jdbc:db2://hostName:port/databaseName`

© 2016-2022 Altova GmbH

IBM DB2 for i	`jdbc:as400://[host]`
IBM Informix	`jdbc:informix-sqli://hostName:port/databaseName:INFORMIXSERVER=myserver`
MariaDB	`jdbc:mariadb://hostName:port/databaseName`
Microsoft SQL Server	`jdbc:sqlserver://hostName:port;databaseName=name`
MySQL	`jdbc:mysql://hostName:port/databaseName`
Oracle	`jdbc:oracle:thin:@hostName:port:SID` `jdbc:oracle:thin:@//hostName:port/service`
Oracle XML DB	`jdbc:oracle:oci:@//hostName:port:service`
PostgreSQL	`jdbc:datadirect:openedge://host:port;databaseName=db_name`
Progress OpenEdge	`jdbc:datadirect:openedge://host:port;databaseName=db_name`
Sybase	`jdbc:sybase:Tds:hostName:port/databaseName`
Teradata	`jdbc:teradata://databaseServerName`

Anmerkung: Bei den oben aufgelisteten Formaten sind auch Syntaxvarianten möglich (die Datenbank-URL kann eventuell ohne Port oder einschließlich Benutzernamen und Datenbank-Passwort angegeben werden). Nähere Informationen dazu finden Sie in der Dokumentation des jeweiligen Datenbankanbieters.

7. Klicken Sie auf **Verbinden**.

4.2.1.6.1 Konfigurieren des CLASSPATH

Mit Hilfe der CLASSPATH-Umgebungsvariablen findet das Java Runtime Environment (JRE) bzw. der Java Development Kit (JDK) Java-Klassen und andere Ressourcendateien auf Ihrem Betriebssystem. Bei Herstellung einer Datenbankverbindung über JDBC muss diese Variable so konfiguriert werden, dass sie den Pfad zum JDBC-Treiber auf Ihrem Betriebssystem und in einigen Fällen den Pfad zur zusätzlichen datenbanktypspezifischen Bibliotheken enthält.

In der folgenden Tabelle sind typische Beispieldateipfade aufgelistet, die in der CLASSPATH-Variablen enthalten sein müssen. Sie müssen diese Informationen eventuell anhand des JDBC-Treibers auf Ihrem System, des JDBC-Treibernamens sowie der JRE/JDK-Version auf Ihrem Betriebssystem anpassen. Um Verbindungsprobleme zu vermeiden, lesen Sie die Installationsanleitung zu dem auf Ihrem Betriebssystem installierten JDBC-Treiber und führen Sie alle vor oder nach der Installation erforderlichen Schritte durch.

Datenbank	**CLASSPATH-Beispieleinträge**
Firebird	`C:\Programme\Firebird\Jaybird-2.2.8-JDK_1.8\jaybird-full-2.2.8.jar`

IBM DB2	`C:\Programme (x86)\IBM\SQLLIB\java\db2jcc.jar;C:\Programme (x86)\IBM\SQLLIB\java\db2jcc_license_cu.jar;`
IBM DB2 for i	`C:\jt400\jt400.jar;`
IBM Informix	`C:\Informix_JDBC_Driver\lib\ifxjdbc.jar;`
Microsoft SQL Server	`C:\Programme\Microsoft JDBC Driver 4.0 for SQL Server\sqljdbc_4.0\enu\sqljdbc.jar`
MariaDB	**`<installation directory>`**`\mariadb-java-client-2.2.0.jar`
MySQL	`mysql-connector-java-`*`version`*`-bin.jar;`
Oracle	**`ORACLE_HOME`**`\jdbc\lib\ojdbc6.jar;`
Oracle (mit XML DB)	**`ORACLE_HOME`**`\jdbc\lib\ojdbc6.jar;`**`ORACLE_HOME`**`\LIB\xmlparserv2.jar;` **`ORACLE_HOME`**`\RDBMS\jlib\xdb.jar;`
PostgreSQL	**`<installation directory>`**`\postgresql.jar`
Progress OpenEdge	`%DLC%\java\openedge.jar;%DLC%\java\pool.jar;` Anmerkung: Angenommen, Progress OpenEdge SDK ist auf dem Rechner installiert, so ist `%DLC%` das Verzeichnis, in dem OpenEdge installiert ist.
Sybase	`C:\sybase\jConnect-7_0\classes\jconn4.jar`
Teradata	**`<installation directory>`**`\tdgssconfig.jar;`**`<installation directory>`**`\terajdbc4.jar`

- Wenn Sie die CLASSPATH-Variable ändern, kann sich dies auf das Verhalten von Java-Applikationen auf Ihrem Rechner auswirken. Lesen Sie dazu die Java-Dokumentation.
- Umgebungsvariablen können benutzer- oder systemspezifisch sein. Um System-Umgebungsvariablen zu ändern, benötigen Sie Administratorrechte auf Ihrem Betriebssystem.
- Nachdem Sie die Umgebungsvariable geändert haben, starten Sie alle laufenden Programme neu, damit die Änderungen wirksam werden. Alternativ dazu können Sie sich am Betriebssystem auch ab- und wieder anmelden oder dieses neu starten.

So konfigurieren Sie den CLASSPATH unter Windows 7:

1. Öffnen Sie das **Startmenü** und klicken Sie mit der rechten Maustaste auf **Computer**.
2. Klicken Sie auf **Eigenschaften**.
3. Klicken Sie auf **Erweiterte Systemeinstellungen**.
4. Klicken Sie auf dem Register **Erweitert** auf **Umgebungsvariablen.**
5. Gehen Sie unter Benutzer- oder System-Umgebungsvariablen zur CLASSPATH-Variablen und klicken Sie anschließend auf **Bearbeiten**. Wenn die CLASSPATH-Variable nicht vorhanden ist, klicken Sie auf **Neu**, um sie zu erstellen.
6. Bearbeiten Sie den Wert der Variablen, damit sie den Pfad enthält, auf dem sich auf Ihrem Betriebssystem der JDBC-Treiber befindet. Verwenden Sie das Semikolon (;), um den JDBC-Treiberpfad von anderen in der CLASSPATH-Variablen bereits vorhandenen Pfaden zu trennen.

© 2016-2022 Altova GmbH

So konfigurieren Sie den CLASSPATH unter Windows 10:

1. Drücken Sie die Windows-Taste und beginnen Sie mit der Eingabe von "Umgebungsvariablen".
2. Klicken Sie auf den Vorschlag **Systemumgebungsvariablen bearbeiten**.
3. Klicken Sie auf **Umgebungsvariablen.**
4. Gehen Sie unter Benutzer- oder System-Umgebungsvariablen zur CLASSPATH-Variablen und klicken Sie anschließend auf **Bearbeiten**. Wenn die CLASSPATH-Variable nicht vorhanden ist, klicken Sie auf **Neu**, um sie zu erstellen.
5. Bearbeiten Sie den Wert der Variablen, damit sie den Pfad enthält, auf dem sich auf Ihrem Betriebssystem der JDBC-Treiber befindet. Verwenden Sie das Semikolon (;), um den JDBC-Treiberpfad von anderen in der CLASSPATH-Variablen bereits vorhandenen Pfaden zu trennen.

4.2.1.7 PostgreSQL-Verbindung

Verbindungen zu PostgreSQL-Datenbanken können entweder als native Verbindung oder über ODBC, JDBC und andere Treiber hergestellt werden. Der Vorteil einer nativen Verbindung ist, dass keine Treiber auf Ihrem System installiert werden müssen. Wenn Sie Dateien für die Ausführung auf einem Linux- oder macOS-Server bereitstellen möchten, müssen auch auf dem Zielserver keine Treiber installiert werden.

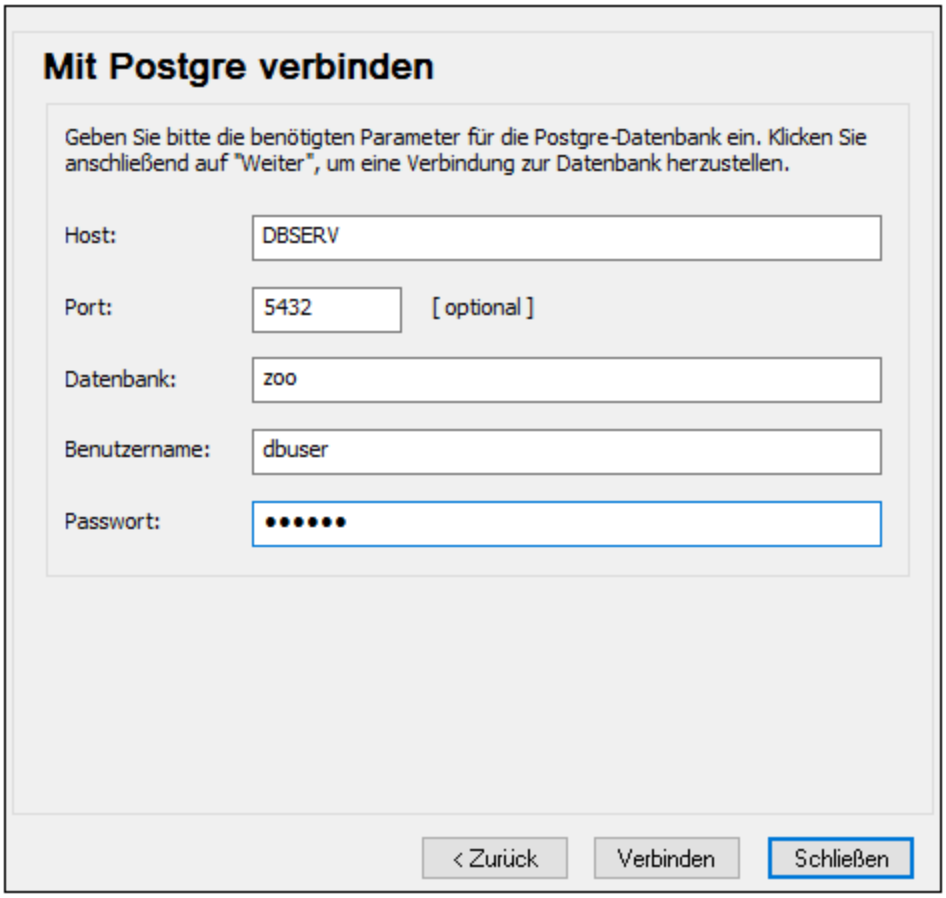
Wenn Sie die Verbindung lieber über einen nicht nativen Treiber herstellen möchten, lesen Sie in den folgenden Kapiteln nach:

- Einrichten einer JDBC-Verbindung 190
- Verbinden mit PostgreSQL (ODBC) 240

Wenn Sie eine native Verbindung zu PostgreSQL herstellen möchten, befolgen Sie die unten stehenden Anleitung. Um fortzufahren, benötigen Sie die folgenden Informationen: Host-Name, Port, Datenbankname, Benutzername und Passwort.

So richten Sie eine native PostgreSQL-Verbindung ein:

1. Starten Sie den Datenbank-Verbindungsassistenten 169.
2. Klicken Sie auf **PostgreSQL-Verbindungen**.
3. Geben Sie den Host (*localhost*, wenn PostgreSQL auf demselben Rechner ausgeführt wird), Port (normalerweise 5432, dies ist optional), den Datenbanknamen, den Benutzernamen und das Passwort in die entsprechenden Felder ein.



4. Klicken Sie auf **Verbinden**.

Wenn sich der PostgreSQL-Datenbankserver auf einem anderen Rechner befindet, beachten Sie die folgenden Punkte:

- Der PostgreSQL-Datenbankserver muss so konfiguriert sein, dass er Verbindungen von Clients zulässt. Insbesondere muss die Datei **pg_hba.conf** so konfiguriert werden, dass sie nicht lokale Verbindungen zulässt. Außerdem muss die Datei **postgresql.conf** so konfiguriert werden, dass sie eine listen-Verbindung zu bestimmten ID-Adressen und einem bestimmten Port hat. Nähere Informationen dazu finden Sie in der Dokumentation zu PostgreSQL (https://www.postgresql.org/docs/9.5/static/client-authentication-problems.html).
- Der Server-Rechner muss so konfiguriert sein, dass er Verbindungen am angegebenen Port (normalerweise 5432) durch die Firewall zulässt. So müssen Sie z.B. eventuell auf einem Windows-Datenbankserver eine Regel erstellen, damit Verbindungen über Port 5432 durch die Firewall zugelassen werden. Wählen Sie dazu **Systemsteuerung > Windows Firewall > Erweiterte Einstellungen > Eingehende Regeln**.

4.2.1.8 SQLite-Verbindung

SQLite ist ein dateibasierter, eigenständiger Datenbanktyp, der dadurch ideal für Szenarien geeignet ist, in denen Portabilität und einfache Konfiguration wichtig sind. Da SQLite-Datenbanken von MapForce nativ

 © 2016-2022 Altova GmbH

unterstützt werden, müssen zur Herstellung einer Verbindung mit SQLite-Datenbanken keine Treiber installiert werden.

Anmerkungen zur Unterstützung von SQLite-Datenbanken

- Für SQLite-Datenbanken kann auf Linux-Systemen kein Timeout für eine Anweisungsausführung definiert werden.
- Die Volltextsuche in Tabellen wird nicht unterstützt.
- Bei SQLite können in jeder Zeile einer Tabelle Werte unterschiedliche Datentypen verwendet werden. Alle verarbeiteten Werte müssen mit dem deklarierten Spaltentyp kompatibel sein; daher können unerwartete Werte abgerufen werden und Laufzeitfehler auftreten, wenn der Wert in der Zeile einer SQLite-Datenbank nicht mit dem deklarierten Spaltentyp übereinstimmt.
- Wenn bei Ihrem Mapping Daten in eine SQLite-Datenbank geschrieben werden sollen und die Zieldatenbank noch nicht vorhanden ist, müssen Sie diese separat erstellen. Verwenden Sie dazu ein Tool wie DatabaseSpy oder laden Sie die SQLite-Befehlszeilen-Shell von der offiziellen Website herunter und erstellen Sie die Datenbankdatei über die Befehlszeile (siehe auch Beispiel: Mappen von Daten von XML auf SQLite 339). Eine ausführliche Dokumentation zur SQLite-Befehlszeilensyntax finden Sie in der offiziellen SQLite-Dokumentation.
- SQLite-Datenbanken werden in der BUILT-IN-Transformationssprache von MapForce (bei der Mapping-Vorschau oder beim Ausführen einer MapForce Server-Ausführungsdatei) unterstützt.
- SQLite-Datenbanken werden in benutzerdefinierten Funktionen (UDF=user-defined functions) nicht unterstützt.

Achtung

Es wird empfohlen, Tabellen mit dem Schlüsselwort `STRICT` zu erstellen, um ein besser vorhersehbares Verhalten Ihrer Daten sicherzustellen. Andernfalls können Daten eventuell nicht korrekt gelesen oder geschrieben werden, wenn eine Spalte Werte unterschiedlichen Typs enthält. Nähere Informationen zu STRICT-Tabellen finden Sie in der SQLite-Dokumentation.

4.2.1.8.1 Herstellen einer Verbindung zu einer bestehenden SQLite-Datenbank

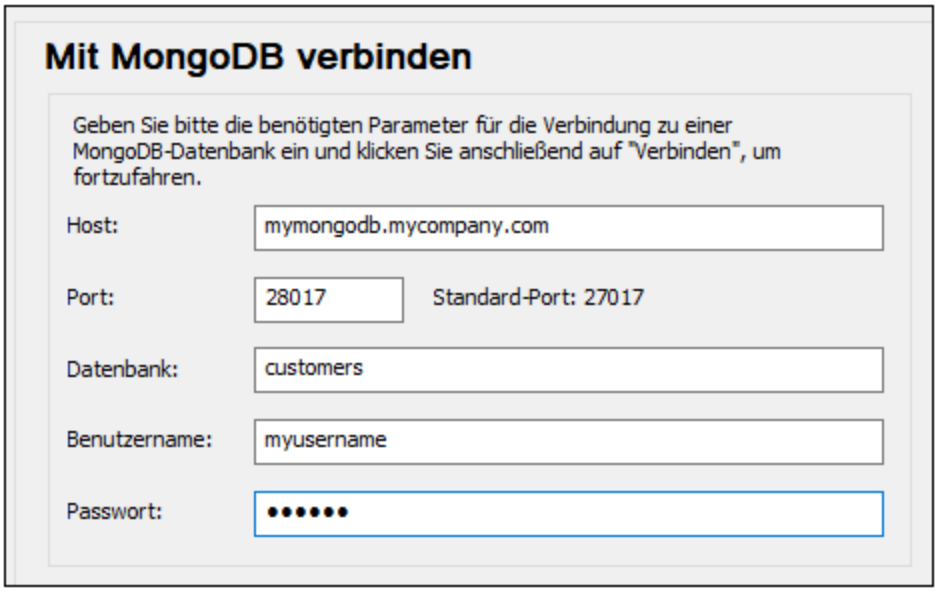
So stellen Sie eine Verbindung zu einer vorhandenen SQLite-Datenbank her:

1. Starten Sie den Datenbankverbindungsassistenten (siehe Starten des Datenbankverbindungsassistenten 169).
2. Wählen Sie **SQLite** aus und klicken Sie auf **Weiter**.
3. Navigieren Sie zur SQLite-Datenbankdatei oder geben Sie (entweder den relativen oder den absoluten) Pfad zur Datenbank ein. Die Schaltfläche **Verbinden** wird aktiv, sobald Sie den Pfad zur SQLite-Datenbankdatei eingegeben haben.
4. Klicken Sie auf **Verbinden**.

4.2.1.9 MongoDB-Verbindung

Um eine Verbindung zu einer vorhandenen MongoDB herzustellen, gehen Sie folgendermaßen vor.

1. Starten Sie den Datenbankverbindungsassistenten (*siehe Starten des Datenbankverbindungsassistenten*[169]).
2. Wählen Sie MongoDB aus und klicken Sie auf **Weiter**.
3. Geben Sie im daraufhin angezeigten Bildschirm *Mit MongoDB verbinden* (*Abbildung unten*) die Verbindungsinformationen für die MongoDB ein.



4. Klicken Sie auf **Verbinden**.
5. Daraufhin wird die Verbindung hergestellt und der Bildschirm *Datenbankobjekte einfügen*, in dem die Dokument-(Datensatz)-Collections der ausgewählten Datenbank aufgelistet werden, wird angezeigt.
6. Wählen Sie die gewünschte(n) Collection(s) aus und klicken Sie auf **OK**.

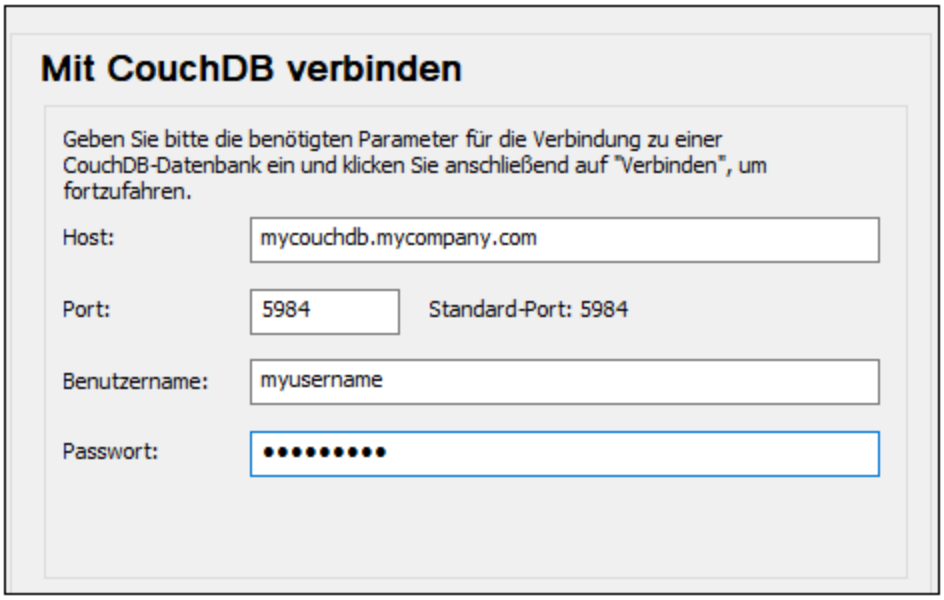
Anmerkung: Wenn Sie MongoDB Realm verwenden, können Sie Ihre Datenbank über Realm Sync aufrufen. Nähere Informationen darüber, wie Sie die Verbindungs-URL abrufen, finden Sie in der MongoDB Atlas-Dokumentation.

4.2.1.10 CouchDB-Verbindung

Um eine Verbindung zu einer vorhandenen CouchDB herzustellen, gehen Sie folgendermaßen vor.

1. Starten Sie den Datenbankverbindungsassistenten (*siehe Starten des Datenbankverbindungsassistenten*[169]).
2. Wählen Sie **CouchDB** aus und klicken Sie auf **Weiter**.
3. Geben Sie im daraufhin angezeigten Bildschirm *Mit CouchDB verbinden* (*Abbildung unten*) die Verbindungsinformationen für die CouchDB ein.

© 2016-2022 Altova GmbH



4. Klicken Sie auf **Verbinden**.
5. Daraufhin wird die Verbindung hergestellt und der Bildschirm *Datenbankobjekte einfügen*, in dem die Tabellen der Datenbank aufgelistet werden, wird angezeigt.
6. Wählen Sie die gewünschte(n) Tabelle(n) aus und klicken Sie auf **OK**.

4.2.1.11 Globale Ressourcen

Nachdem Sie eine Datenbank als globale Ressource erstellt haben, werden ihre Verbindungsinformationen gespeichert und können in allen auf Ihrem Rechner installierten Altova-Produkten verwendet werden.

Erstellen einer Datenbank als globale Ressource

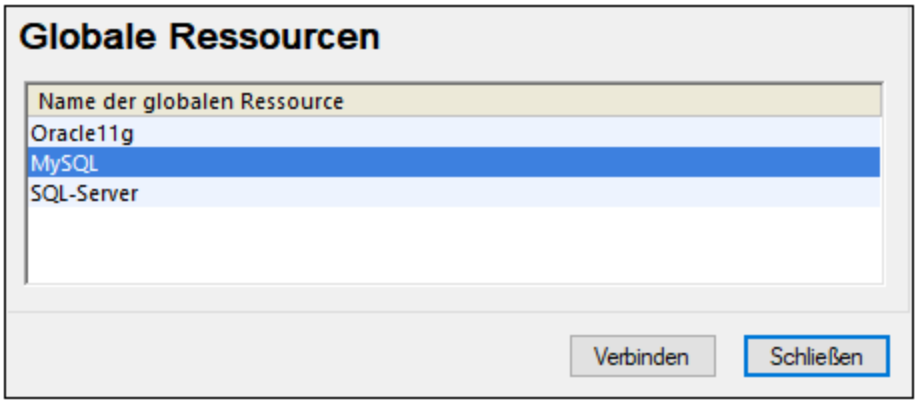
Um eine Datenbank als globale Ressource zu erstellen, gehen Sie folgendermaßen vor:

1. Wählen Sie im Menü **Extras** von MapForce den Befehl **Globale Ressourcen**.
2. Klicken Sie auf **Hinzufügen** und anschließend auf Datenbank.
3. Geben Sie in das Feld *Ressourcen-Alias* einen Namen für die globale Ressource ein.
4. Klicken Sie auf **Datenbank auswählen**. Daraufhin wird der Verbindungsassistent 169 angezeigt.
5. Fügen Sie mit Hilfe des Verbindungsassistenten, wie oben beschrieben, eine Datenbankverbindung hinzu.

Verwenden einer Datenbank aus den globalen Ressourcen

Um eine als globale Ressource erstellte Datenbank (*siehe oben*) zu verwenden, gehen Sie folgendermaßen vor:

1. Starten Sie den Verbindungsassistenten, wie oben beschrieben.
2. Wählen Sie Globale Ressourcen aus. Alle als globale Ressourcen erstellten Datenbanken werden im Fenster "Globale Ressourcen" nach ihrem Namen aufgelistet (*siehe Abbildung unten*).



3. Wählen Sie die gewünschte globale Ressource aus. Tipp: Wenn Sie den Mauszeiger über eine globale Ressource in der Liste platzieren, werden Informationen über die Datenbank angezeigt.

4.2.1.12 Beispiele für Datenbankverbindungen

In diesem Abschnitt sind Beispiele dafür beschrieben, wie Sie von MapForce aus über ADO, ODBC oder JDBC eine Verbindung zu einer Datenbank herstellen. Die Beispiele für ADO.NET-Verbindungen sind separat aufgelistet, siehe Beispiele für ADO.NET Connection Strings 185. Anleitungen dazu, wie Sie eine native Verbindung zu PostgreSQL und SQLite herstellen, finden Sie unter Einrichten einer PostgreSQL-Verbindung 195 bzw. Einrichten einer SQLite-Verbindung 196.

Beachten Sie die folgenden Punkte:

- Wenn Ihre Windows-Konfiguration, Ihre Netzwerkumgebung und Ihre Datenbank Client- oder Server-Software nicht genau der in den Beispielen beschriebenen Konfiguration entsprechen, weicht die Vorgangsweise eventuell etwas von der in den Beispielen beschriebenen ab.
- Bei den meisten Datenbanktypen kann die Verbindung über unterschiedliche Datenzugriffstechnologien (ADO, ADO.NET, ODBC, JDBC) oder Treiber hergestellt werden. Das Verhalten der Datenbankverbindung, die verfügbaren Funktionalitäten und Einschränkungen hängen vom ausgewählten Treiber, (gegebenenfalls) der Datenbank Client-Software und eventuellen zusätzlich außerhalb von MapForce konfigurierten Verbindungsparametern ab.

4.2.1.12.1 Firebird (JDBC)

Dieses Kapitel enthält eine Beispielanleitung für das Herstellen einer Verbindung zu einem Firebird Datenbankserver mittels JDBC.

Voraussetzungen:

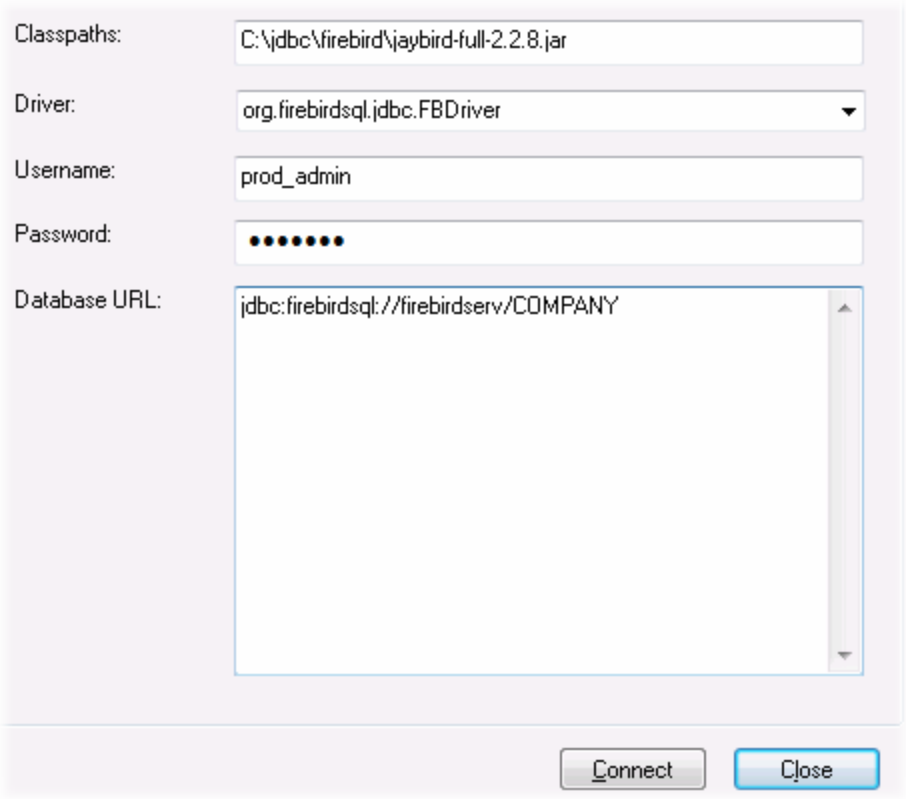
- Auf Ihrem Betriebssystem muss JRE (Java Runtime Enviroment) oder Java Development KIT (JDK) installiert sein. Dabei muss es sich entweder um Oracle JDK oder einen Open Source Build wie Oracle OpenJDK handeln.MapForce ermittelt den Pfad zur Java Virtual Machine (JVM) anhand der folgenden Ordner und zwar in folgender Reihenfolge: a) anhand des benutzerdefinierten JVM-Pfads, den Sie

 © 2016-2022 Altova GmbH

eventuell in den **Applikationsoptionen** definiert haben, siehe Java-Einstellungen[1542]; b) anhand des JVM-Pfads in der Windows Registry; c) anhand der JAVA_HOME-Umgebungsvariablen.
- Stellen Sie sicher, dass die Plattform von MapForce (32-Bit, 64-Bit) mit der des JRE/JDK übereinstimmt.
- Der Firebird JDBC-Treiber muss auf Ihrem Betriebssystem verfügbar sein (Er hat die Form einer .jar-Datei, die eine Verbindung zur Datenbank herstellt). Der Treiber kann von der Firebird Website (https://www.firebirdsql.org/) heruntergeladen werden. In diesem Beispiel wird *Jaybird 2.2.8* verwendet.
- Sie haben die folgenden Datenbankverbindungsinformationen zur Verfügung: Host, Datenbankpfad oder Alias, Benutzername und Passwort.

So stellen Sie über JDBC eine Verbindung zu Firebird her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **JDBC-Verbindungen**.
3. Geben Sie neben "Classpaths" den Pfad zur .jar-Datei , die die Verbindung zur Datenbank bereitstellt, ein. Falls nötig, können Sie auch eine durch Semikola getrennte Liste von .jar-Dateipfaden eingeben. Die benötigte .jar-Datei in diesem Beispiel befindet sich unter dem folgenden Pfad: **C:\jdbc\firebird\jaybird-full-2.2.8.jar**. Beachten Sie, dass Sie das Textfeld "Classpaths" leer lassen können, wenn Sie den/die .jar-Dateipfad(e) zur Umgebungsvariablen CLASSPATH des Betriebssystems hinzugefügt haben (siehe auch Konfigurieren des CLASSPATH[193]).
4. Wählen Sie im Feld "Treiber" **org.firebirdsql.jdbc.FBDriver** aus. Beachten Sie, dass dieser Eintrag zur Verfügung steht, wenn entweder im Textfeld "Classpaths" oder in der Umgebungsvariablen CLASSPATH des Betriebssystems eine gültige .jar-Datei gefunden wird (siehe vorheriger Schritt).



5. Geben Sie den Benutzernamen und das Passwort in die entsprechenden Textfelder ein.

6. Geben Sie in das Textfeld "Datenbank-URL" den Connection String zum Datenbankserver ein, indem Sie die hervorgehobenen Werte durch die entsprechenden Werte für Ihren Datenbankserver ersetzen.

```
jdbc:firebirdsql://<host>[:<port>]/<database path or alias>
```

7. Klicken Sie auf **Verbinden**.

4.2.1.12.2 Firebird (ODBC)

Dieses Kapitel enthält eine Beispielanleitung für das Herstellen einer Verbindung zu einer Firebird 2.5.4-Datenbank auf einem Linux Server.

Voraussetzungen:

- Der Firebird-Datenbankserver akzeptiert aufgrund seiner Konfiguration TCP/IP-Verbindungen von Clients.
- Der Firebird ODBC-Treiber muss auf Ihrem Betriebssystem installiert sein. In diesem Beispiel wird der Firebirg ODBC-Treiber Version 2.0.3.154 verwendet, der von der Firebird Website (https://www.firebirdsql.org/) heruntergeladen wurde.
- Der Firebird Client muss auf Ihrem Betriebssystem installiert sein. Beachten Sie, dass für den Firebird 2.5.4 Client kein eigenständiger Installer verfügbar ist; der Client ist Teil des Firebird Server-Installationspakets, welches Sie unter "Windows executable installer for full Superclassic/Classic or Superserver" von der Firebird Website (https://www.firebirdsql.org/) herunterladen können. Um nur die Client-Dateien zu installieren, wählen Sie im Zuge der Installation die Option "**Minimum client install - no server, no tools**" aus.

  Wichtig:

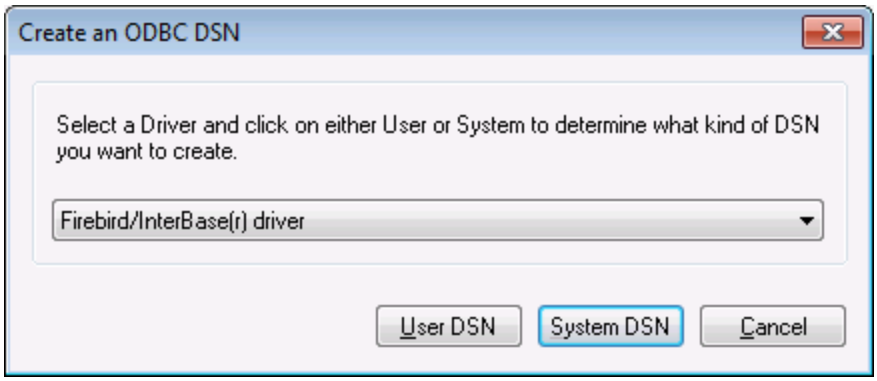
  - Die Plattform des Firebird ODBC-Treibers und Client (32-Bit oder 64-Bit) muss mit der von MapForce übereinstimmen.
  - Die Version des Firebird Client muss mit der des Firebird Servers, zu dem Sie die Verbindung herstellen, übereinstimmen.

- Sie verfügen über die folgenden Datenbankverbindungsinformationen: Name oder IP-Adresse des Server Host, Datenbankpfad (oder Alias) auf dem Server, Benutzername und Passwort.

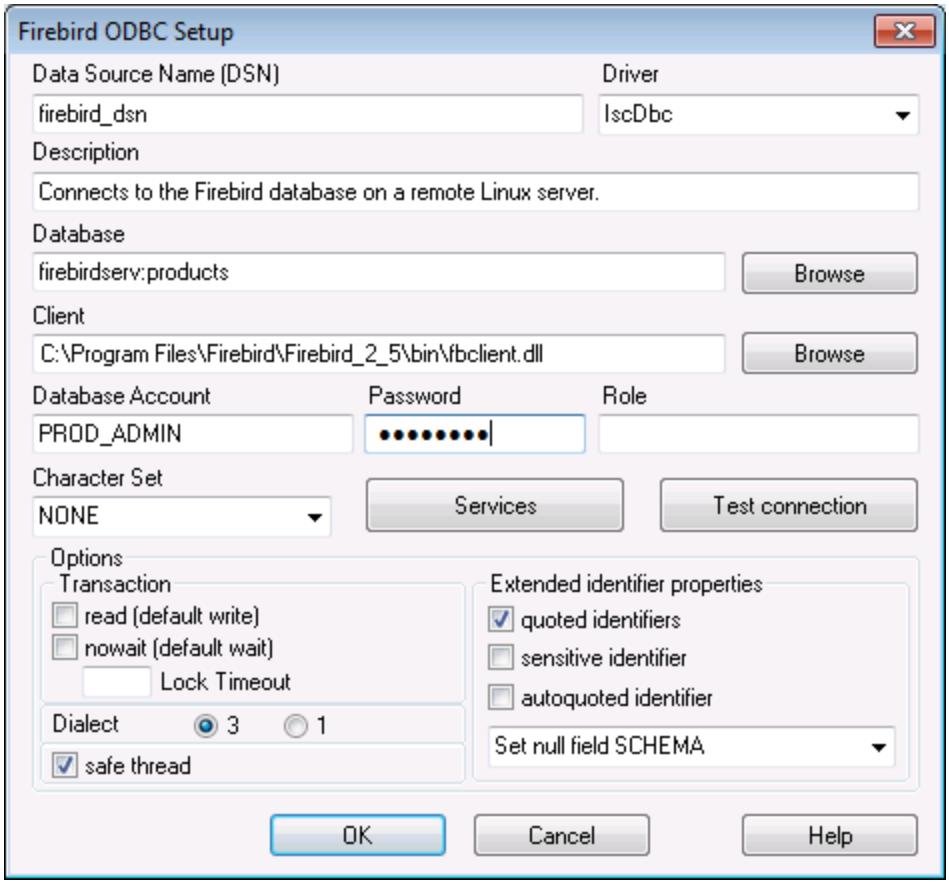
So stellen Sie über ODBC eine Verbindung zu Firebird her:

1. Starten Sie den Datenbank-Verbindungsassistenten 169.
2. Klicken Sie auf **ODBC-Verbindungen**.
3. Wählen Sie Benutzer-DSN (oder **System-DSN**, wenn Sie Administratorrechte haben) und klicken Sie anschließend auf **Hinzufügen** .

© 2016-2022 Altova GmbH



4. Wählen Sie den Firebird-Treiber aus und klicken Sie anschließend, je nachdem, was Sie im vorherigen Schritt ausgewählt haben, auf **Benutzer-DSN** oder **System-DSN**. Wenn der Firebird-Treiber in der Liste nicht zur Verfügung steht, stellen Sie sicher, dass er auf Ihrem Betriebssystem installiert ist (siehe auch Anzeigen der verfügbaren ODBC-Treiber 189).



5. Geben Sie die folgenden Datenbankverbindungsinformationen ein:

Data Source Name (DSN)	Geben Sie einen beschreibenden Namen für die zu erstellende Datenquelle ein.

Database	Geben Sie den Namen oder die IP-Adresse des Server Host, gefolgt von einem Doppelpunkt, gefolgt vom Datenbank-Alias (oder Pfad) ein. In diesem Beispiel lautet der Host-Name `firebirdserv` und der Datenbank-Alias `products`. Geben Sie daher den folgenden String ein:
	
	`firebirdserv:products`
	
	Wenn Sie einen Datenbank-Alias verwenden, wird davon ausgegangen, dass der Datenbankadministrator den Alias *products* auf dem Server so konfiguriert hat, dass er auf die tatsächliche Firebird (.fdb) Datenbankdatei verweist (nähere Informationen dazu siehe Firebird-Dokumentation).
	
	Sie können anstelle des Host-Namens auch die Server IP-Adresse und anstelle eines Pfads einen Alias verwenden; daher ist jeder der folgenden Beispiel Connection-Strings gültig:
	
	`firebirdserver:/var/Firebird/databases/butterflies.fdb`
	`127.0.0.1:D:\Misc\Lenders.fdb`
	
	Wenn sich die Datenbank auf dem lokalen Windows-Rechner befindet, klicken Sie auf **Durchsuchen** und wählen Sie die Firebird (.fdb) Datenbank direkt aus.

Client	Geben Sie den Pfad zur Datei **fbclient.dll** ein. Standardmäßig ist dies das Unterverzeichnis `bin` des Firebird-Installationsverzeichnisses.
Database Account	Geben Sie den vom Datenbankadministrator bereitgestellten Datenbank-Benutzernamen ein (in diesem Beispiel `PROD_ADMIN`).
Password	Geben Sie das vom Datenbankadministrator bereitgestellte Datenbank-Passwort ein

1. Klicken Sie auf **OK**.

4.2.1.12.3 IBM DB2 (JDBC)

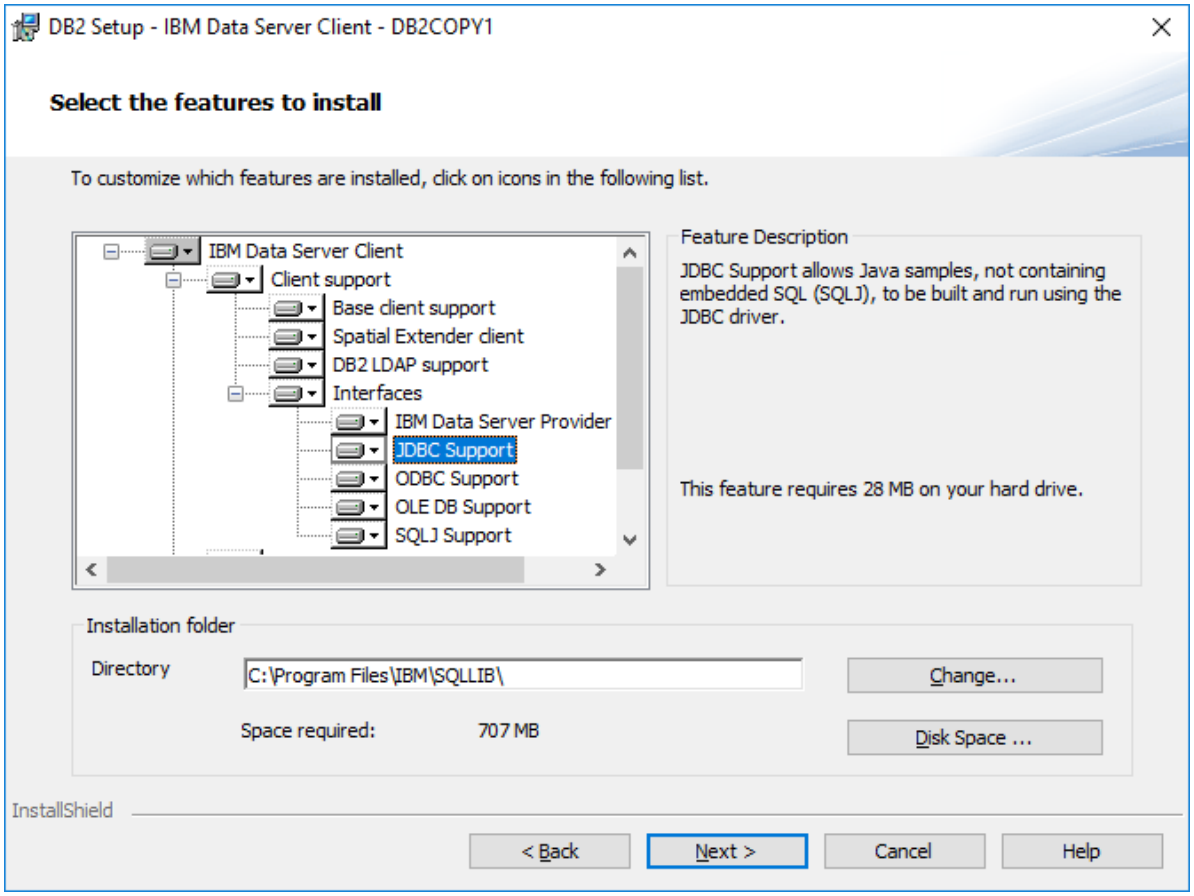
Dieses Kapitel enthält eine Beispielanleitung für das Herstellen einer Verbindung zu einem **IBM DB2** Datenbankserver über JDBC.

Voraussetzungen:

- Auf Ihrem Betriebssystem muss JRE (Java Runtime Enviroment) oder Java Development KIT (JDK) installiert sein. Dabei muss es sich entweder um Oracle JDK oder einen Open Source Build wie Oracle OpenJDK handeln.MapForce ermittelt den Pfad zur Java Virtual Machine (JVM) anhand der folgenden Ordner und zwar in folgender Reihenfolge: a) anhand des benutzerdefinierten JVM-Pfads, den Sie eventuell in den **Applikationsoptionen** definiert haben, siehe Java-Einstellungen[1542]; b) anhand des JVM-Pfads in der Windows Registry; c) anhand der `JAVA_HOME`-Umgebungsvariablen.

© 2016-2022 Altova GmbH

- Stellen Sie sicher, dass die Plattform von MapForce (32-Bit, 64-Bit) mit der des JRE/JDK übereinstimmt. In diesem Beispiel wird Oracle OpenJDK 11.0 64-Bit verwendet, folglich wird auch die 64-Bit-Version von MapForce verwendet.
- Der Firebird JDBC-Treiber (eine oder mehrere .jar-Dateien, die die Verbindung zur Datenbank herstellen) muss auf Ihrem Betriebssystem verfügbar sein. In diesem Beispiel wird der JDBC-Treiber verwendet, der zur Verfügung steht, nachdem Sie die **IBM Data Server Client** Version 10.1 (64-Bit) installiert haben. Um die JDBC-Treiber zu installieren, wählen Sie eine **Standardinstallation** oder wählen Sie diese Option explizit im Installationsassistenten aus.



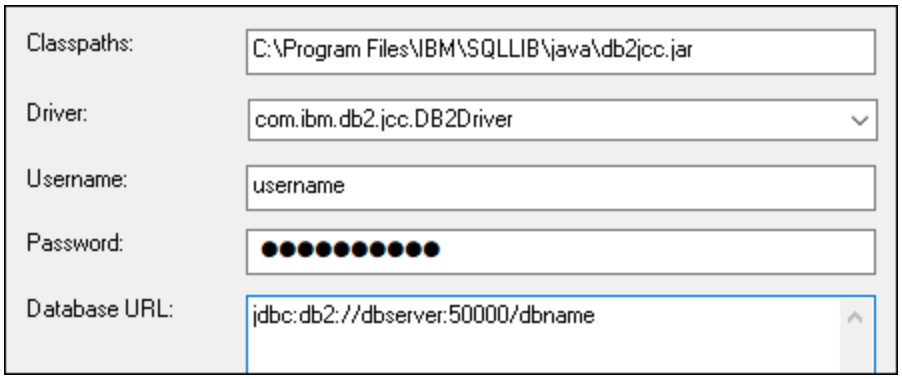
  Wenn Sie den Standardinstallationspfad nicht geändert haben, befinden sich die erforderlichen .jar-Dateien nach der Installation im Verzeichnis **C:\Programme\IBM\SQLLIB\java**.
- Sie benötigen die folgenden Datenbankverbindungsinformationen: Host, Port, Datenbankname, Benutzername und Passwort.

So stellen Sie über JDBC eine Verbindung zu IBM DB2 her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **JDBC-Verbindungen**.
3. Geben Sie neben "Classpaths" den Pfad zur .jar-Datei , die die Verbindung zur Datenbank bereitstellt, ein. In diesem Beispiel wird der Pfad **C:\Programme\IBM\SQLLIB\java\db2jcc.jar** referenziert. Je nach Datenbankversion müssen Sie eventuell den Treiber **db2jcc4.jar** referenzieren. Informationen zur Treiberkompatibilität finden Sie in der IBM-Dokumentation unter (http://www-01.ibm.com/support/docview.wss?uid=swg21363866). Beachten Sie, dass Sie das Textfeld

"Classpaths" leer lassen können, wenn Sie den/die .jar-Dateipfad(e) zur Umgebungsvariablen CLASSPATH des Betriebssystems hinzugefügt haben (siehe auch Konfigurieren des CLASSPATH 193).

4. Wählen Sie im Feld "Treiber" **com.ibm.db2.jcc.DB2Driver** aus. Beachten Sie, dass dieser Eintrag zur Verfügung steht, wenn entweder im Textfeld "Classpath" oder in der Umgebungsvariablen CLASSPATH des Betriebssystems eine gültige .jar-Datei gefunden wird (siehe vorheriger Schritt).



5. Geben Sie den Benutzernamen und das Passwort des Datenbankbenutzers in die entsprechenden Textfelder ein.
6. Geben Sie in das Textfeld **Datenbank-URL** den Connection String zum Datenbankserver ein. Dabei müssen Sie die Verbindungsinformationen durch die entsprechenden Werte für Ihren Datenbankserver ersetzen.

```
jdbc:db2://hostName:port/databaseName
```

7. Klicken Sie auf **Verbinden**.

4.2.1.12.4 IBM DB2 (ODBC)

Dieses Kapitel enthält eine Beispielanleitung für das Herstellen einer Verbindung zu einer IBM DB2-Datenbank über ODBC.

Voraussetzungen:

- Auf Ihrem Betriebssystem muss IBM Data Server Client installiert und konfiguriert sein (in diesem Beispiel wird IBM Data Server Client 9.7 verwendet). Eine Installationsanleitung dazu finden Sie in der Dokumentation zu Ihrer IBM DB2 Software. Nachdem Sie IBM Data Server Client installiert haben, überprüfen Sie, ob die ODBC-Treiber auf Ihrem Rechner verfügbar sind (siehe Anzeigen der verfügbaren ODBC-Treiber 189).
- Erstellen Sie einen Datenbank-Alias. Es gibt mehrere Methoden, dies zu tun:
  - über den IBM DB2-Konfigurationsassistenten
  - über den IBM DB2-Befehlszeilenprozessor
  - über den ODBC-Datenquellenassistenten (die Anleitung dazu finden Sie weiter unten)
- Sie haben die folgenden Datenbankverbindungsinformationen zur Verfügung: Host, Datenbank, Port, Benutzername und Passwort.

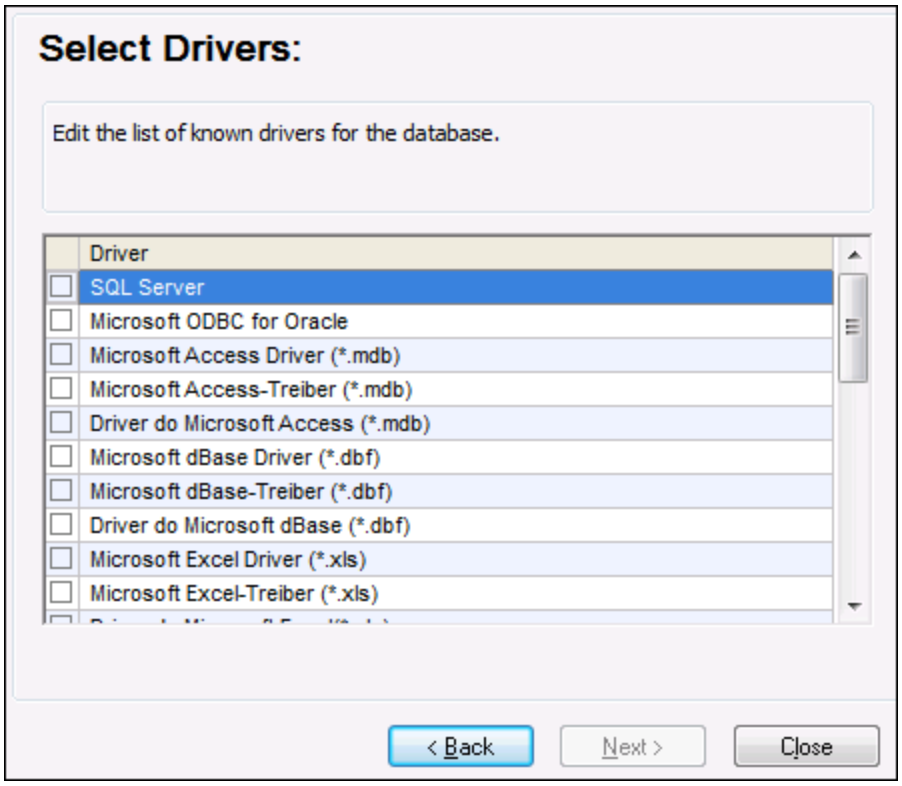
 © 2016-2022 Altova GmbH

So stellen Sie eine Verbindung zu IBM DB2 her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169] und wählen Sie **IBM DB2 (ODBC/JDBC)** aus.
2. Klicken Sie auf **Weiter**.



3. Wählen Sie **ODBC** aus und klicken Sie auf **Weiter.** Wenn Sie aufgefordert werden, die Liste der bekannten Treiber für die Datenbank zu bearbeiten, wählen Sie die Datenbanktreiber für IBM DB2 aus (siehe Voraussetzungen[206]) und klicken Sie auf **Weiter**.

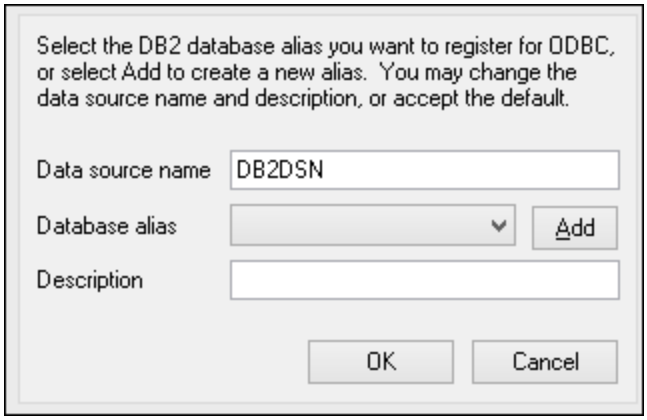


4. Wählen Sie den IBM DB2-Treiber aus der Liste aus und klicken Sie anschließend auf **Verbinden**. (Um die Liste der verfügbare Treiber zu bearbeiten, klicken Sie auf **Treiber bearbeiten** und aktivieren bzw. deaktivieren Sie anschließend die IBM DB2-Treiber, die Sie hinzufügen bzw. entfernen möchten.)

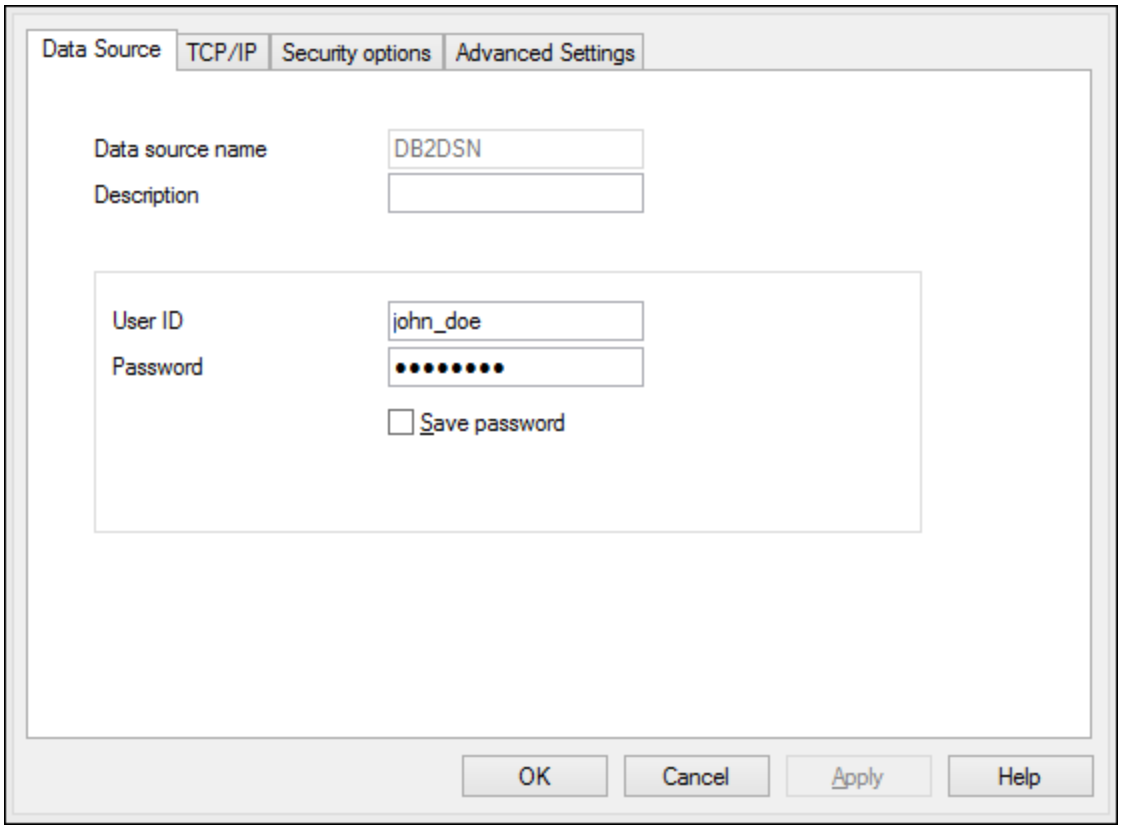
© 2016-2022 Altova GmbH



5. Geben Sie einen Datenquellennamen ein (in diesem Beispiel **DB2DSN**) und klicken Sie auf **Hinzufügen**.

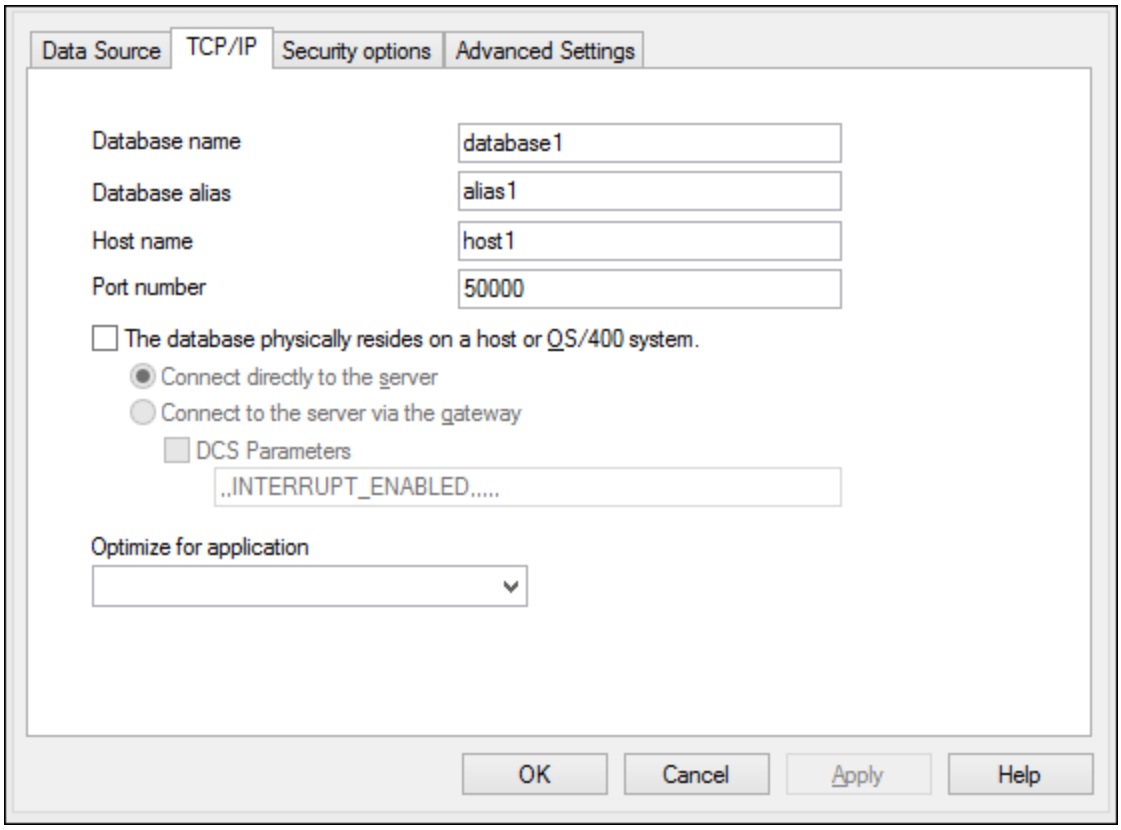


6. Geben Sie auf dem Register **Datenquelle** den Benutzernamen und das Passwort für die Datenbank ein.

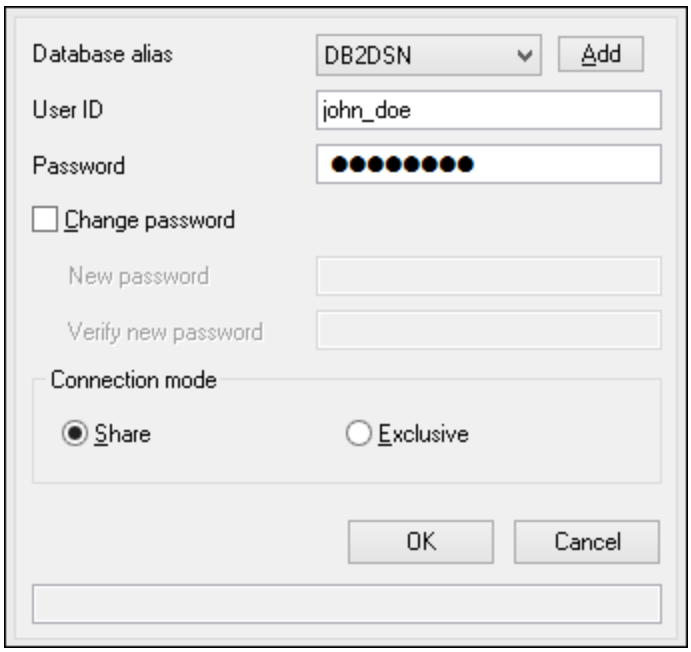


7. Geben Sie auf dem Register **TCP/IP** den Datenbanknamen, einen Namen für den Alias, den Host-Namen und die Port-Nummer ein und klicken Sie auf OK.

© 2016-2022 Altova GmbH



8. Geben Sie den Benutzernamen und das Passwort erneut ein und klicken Sie auf **OK**.



4.2.1.12.5 IBM DB2 für i (JDBC)

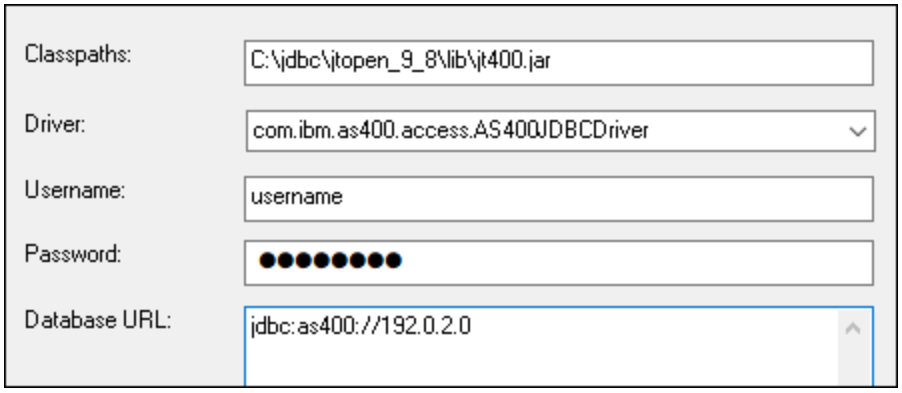
Dieses Kapitel enthält eine Beispielanleitung für das Herstellen einer Verbindung zu einem **IBM DB2 für i**-Datenbankserver über JDBC.

Voraussetzungen:

- Auf Ihrem Betriebssystem muss JRE (Java Runtime Enviroment) oder Java Development KIT (JDK) installiert sein. Dabei muss es sich entweder um Oracle JDK oder einen Open Source Build wie Oracle OpenJDK handeln.MapForce ermittelt den Pfad zur Java Virtual Machine (JVM) anhand der folgenden Ordner und zwar in folgender Reihenfolge: a) anhand des benutzerdefinierten JVM-Pfads, den Sie eventuell in den **Applikationsoptionen** definiert haben, siehe Java-Einstellungen[1542]; b) anhand des JVM-Pfads in der Windows Registry; c) anhand der `JAVA_HOME`-Umgebungsvariablen.
- Stellen Sie sicher, dass die Plattform von MapForce (32-Bit, 64-Bit) mit der des JRE/JDK übereinstimmt. In diesem Beispiel wird Oracle OpenJDK 11.0 64-Bit verwendet, folglich wird auch die 64-Bit-Version von MapForce verwendet.
- Der Firebird JDBC-Treiber (eine oder mehrere .jar-Dateien, die die Verbindung zur Datenbank herstellen) muss auf Ihrem Betriebssystem verfügbar sein. In diesem Beispiel wird die Open Source **Toolbox for Java/JTOpen** Version 9.8 (http://jt400.sourceforge.net/) verwendet. Nachdem Sie das Paket heruntergeladen und in ein lokales Verzeichnis entpackt haben, stehen die erforderlichen .jar-Dateien im Unterverzeichnis **lib** zur Verfügung.
- Sie benötigen die folgenden Datenbankverbindungsinformationen: Host, Benutzername und Passwort.

So stellen Sie über JDBC eine Verbindung zu IBM DB2 für i her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **JDBC-Verbindungen**.
3. Geben Sie neben "Classpaths" den Pfad zur .jar-Datei , die die Verbindung zur Datenbank bereitstellt, ein. In diesem Beispiel wird der Pfad **C:\jdbc\jtopen_9_8\jt400.jar** referenziert. Beachten Sie, dass Sie das Textfeld "Classpaths" leer lassen können, wenn Sie den/die .jar-Dateipfad(e) zur Umgebungsvariablen CLASSPATH des Betriebssystems hinzugefügt haben (siehe auch Konfigurieren des CLASSPATH[193]).
4. Wählen Sie im Feld "Treiber" **com.ibm.as400.access.AS400JDBCDriver** aus. Beachten Sie, dass dieser Eintrag zur Verfügung steht, wenn entweder im Textfeld "Classpath" oder in der Umgebungsvariablen `CLASSPATH` des Betriebssystems eine gültige .jar-Datei gefunden wird (siehe vorheriger Schritt).



 © 2016-2022 Altova GmbH

5. Geben Sie den Benutzernamen und das Passwort des Datenbankbenutzers in die entsprechenden Textfelder ein.
6. Geben Sie in das Textfeld **Datenbank-URL** den JDBC-Connection String zum Datenbankserver ein. Dabei müssen Sie **host** durch den Hostnamen oder die IP-Adresse Ihres Datenbankservers ersetzen.

```
jdbc:as400://host
```

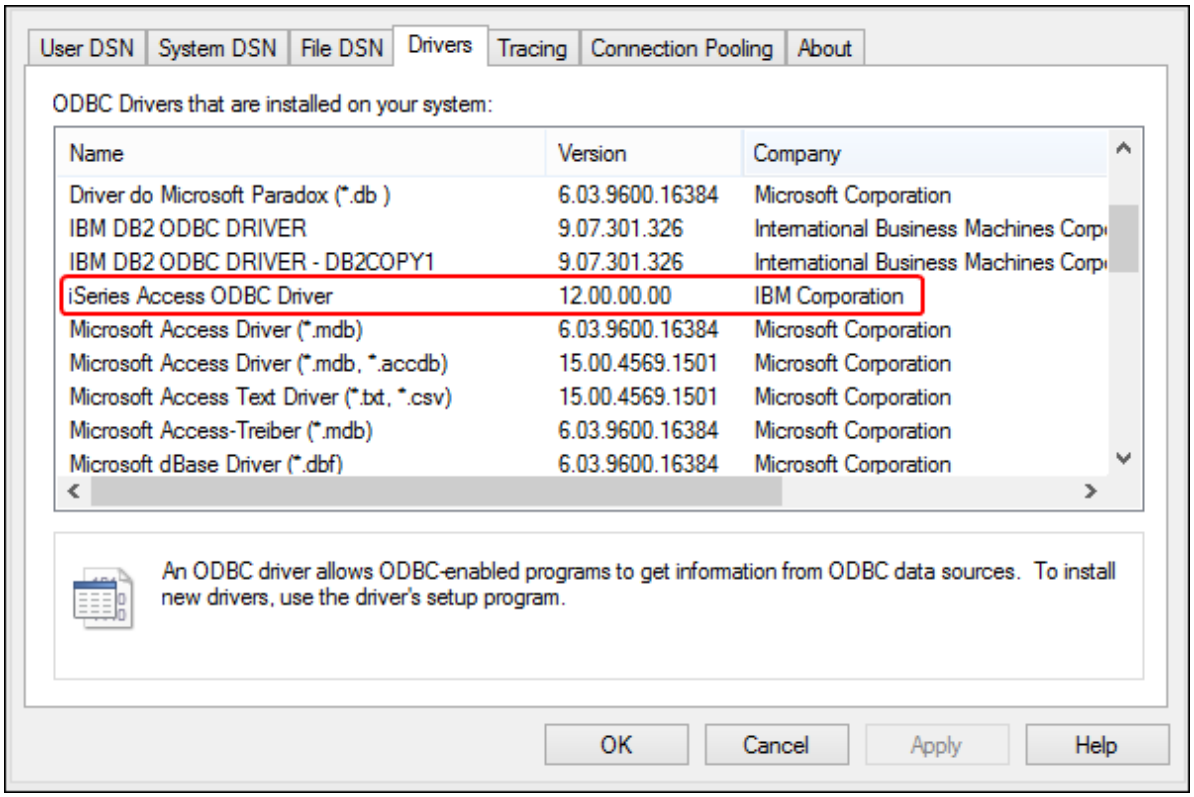
7. Klicken Sie auf **Verbinden**.

4.2.1.12.6 IBM DB2 für i (ODBC)

Dieses Kapitel enthält eine Beispielanleitung für das Herstellen einer Verbindung zu einer *IBM DB2 für i*-Datenbank über ODBC.

Voraussetzungen:

- Auf Ihrem Betriebssystem muss *IBM System i Access für Windows* installiert sein (in diesem Beispiel wird *IBM System i Access für Windows V6R1M0 verwendet*). Eine Installationsanleitung finden Sie in der Dokumentation zu Ihrer *IBM DB2 für i*-Software. Überprüfen Sie nach der Installation, ob der ODBC-Treiber auf Ihrem Rechner verfügbar ist (siehe Anzeigen der verfügbaren OBBC-Treiber 189).

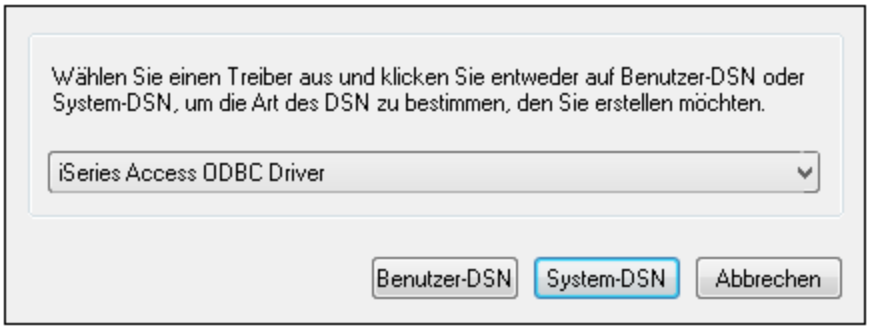


- Sie haben die folgenden Datenbankverbindungsinformationen zur Verfügung: die IP-Adresse des Datenbankservers, den Datenbank-Benutzernamen und das Passwort.
- Führen Sie den *System i Navigator* aus und befolgen Sie die Anweisungen des Assistenten, um eine neue Verbindung zu erstellen. Wenn Sie nach einem System gefragt werden, geben Sie die IP-

Adresse des Datenbankservers ein. Nachdem Sie die Verbindung hergestellt haben, sollten Sie diese überprüfen (Klicken Sie auf den Verbindung und wählen Sie **Datei** > **Diagnose** > **Verbindung überprüfen**). Wenden Sie sich an den Datenbankserver-Administrator, wenn Sie Verbindungsfehler erhalten.

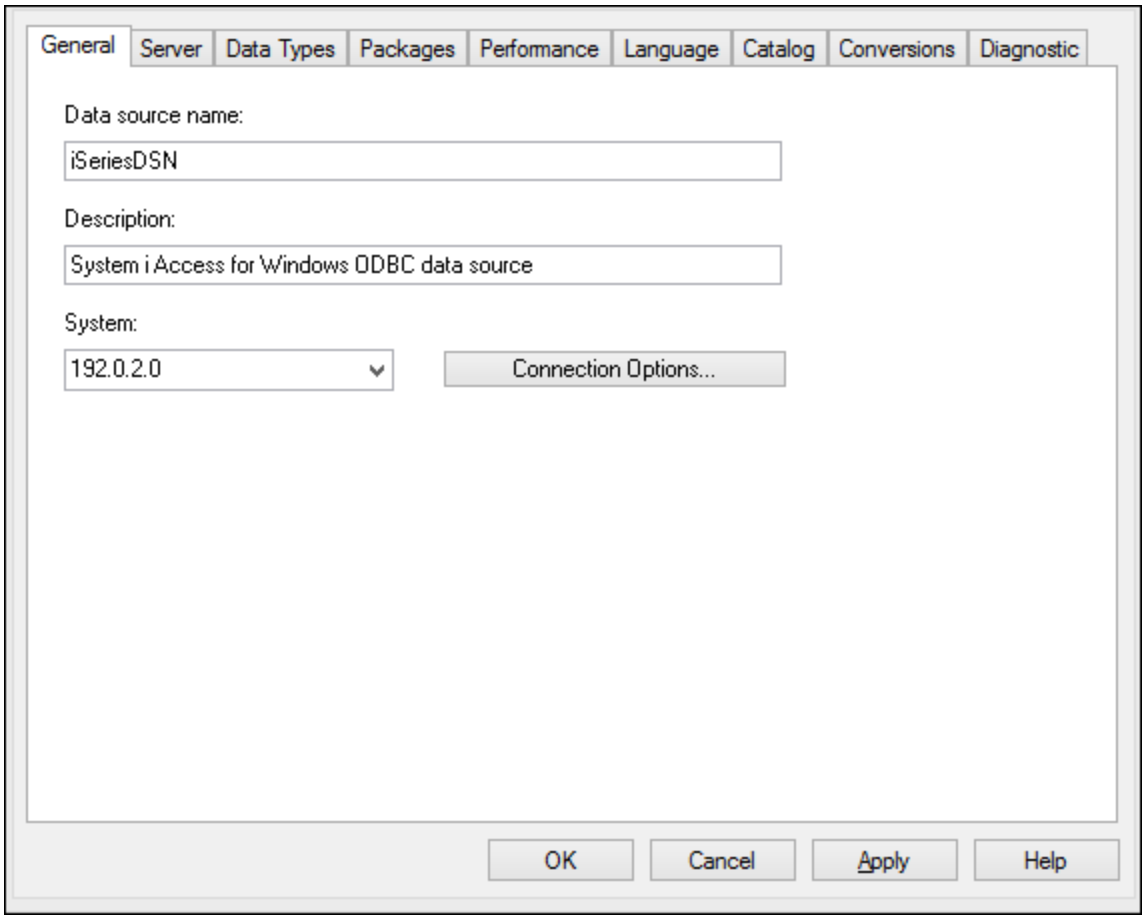
So stellen Sie eine Verbindung zu IBM DB2 für i ein:

1. Starten Sie den Datenbank-Verbindungsassistenten 169.
2. Klicken Sie auf **ODBC-Verbindungen**.
3. Klicken Sie auf **Benutzer-DSN** (Klicken Sie alternativ dazu auf **System-DSN** oder **Datei-DSN** - die darauf folgende Anleitung ist ähnlich).
4. Klicken Sie auf **Hinzufügen** .
5. Wählen Sie aus der Liste **iSeries Access ODBC Driver** aus und klicken Sie auf **Benutzer-DSN** (bzw. gegebenenfalls auf **System- DSN**).



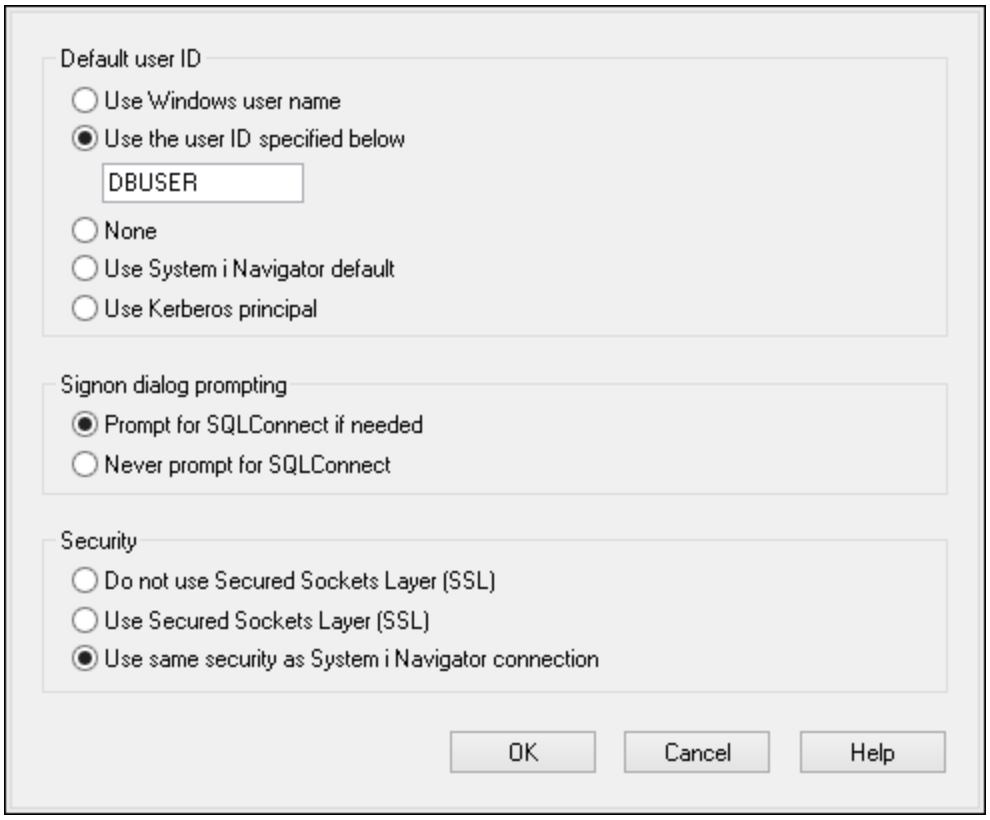
6. Geben Sie den Datenquellennamen ein und wählen Sie die Verbindung aus der System-Auswahlliste aus. In diesem Beispiel lautet der Datenquellenname **iSeriesDSN** und das System ist **192.0.2.0**.

© 2016-2022 Altova GmbH



Anmerkung: Wenn Sie eine ODBC-Datenquelle für eine *IBM DB2 for i*-Datenbank hinzufügen, wird ein Standard-Flag gesetzt, das Abfrage-Timeouts aktiviert. Diese Einstellung muss für MapForce deaktiviert werden, damit Mapping-Dateien korrekt geladen werden. Um die Einstellung zu deaktivieren, klicken Sie auf das Register **Performance,** anschließend auf **Advanced** und deaktivieren Sie das Kontrollkästchen **Allow query timeout**.

7. Klicken Sie auf **Verbindungsoptionen** und wählen Sie **unten angeführte Benutzer-ID verwenden** und geben Sie den Namen des Datenbankbenutzers ein (in diesem Beispiel **DBUSER**).



8. Klicken Sie auf **OK**. Die neue Datenquelle steht nun in der Liste der DSNs zur Verfügung.
9. Klicken Sie auf **Verbinden**.
10. Geben Sie den Benutzernamen und das Passwort für die Datenbank ein, wenn Sie dazu aufgefordert werden und klicken Sie auf **OK**.

4.2.1.12.7 IBM Informix (JDBC)

Dieses Kapitel enthält eine Beispielanleitung für das Herstellen einer Verbindung zu einer IBM Informix-Datenbank über JDBC.

Voraussetzungen:

- Auf Ihrem Betriebssystem muss JRE (Java Runtime Enviroment) oder Java Development KIT (JDK) installiert sein. Dabei muss es sich entweder um Oracle JDK oder einen Open Source Build wie Oracle OpenJDK handeln.MapForce ermittelt den Pfad zur Java Virtual Machine (JVM) anhand der folgenden Ordner und zwar in folgender Reihenfolge: a) anhand des benutzerdefinierten JVM-Pfads, den Sie eventuell in den **Applikationsoptionen** definiert haben, siehe Java-Einstellungen[1542]; b) anhand des JVM-Pfads in der Windows Registry; c) anhand der `JAVA_HOME`-Umgebungsvariablen.
- Stellen Sie sicher, dass die Plattform von MapForce (32-Bit, 64-Bit) mit der des JRE/JDK übereinstimmt.
- Der JDBC-Treiber (eine oder mehrere .jar-Dateien, die die Verbindung zur Datenbank herstellen) muss auf Ihrem Betriebssystem installiert sein. In diesem Beispiel wird die IBM Informix JDBC-Treiberversion

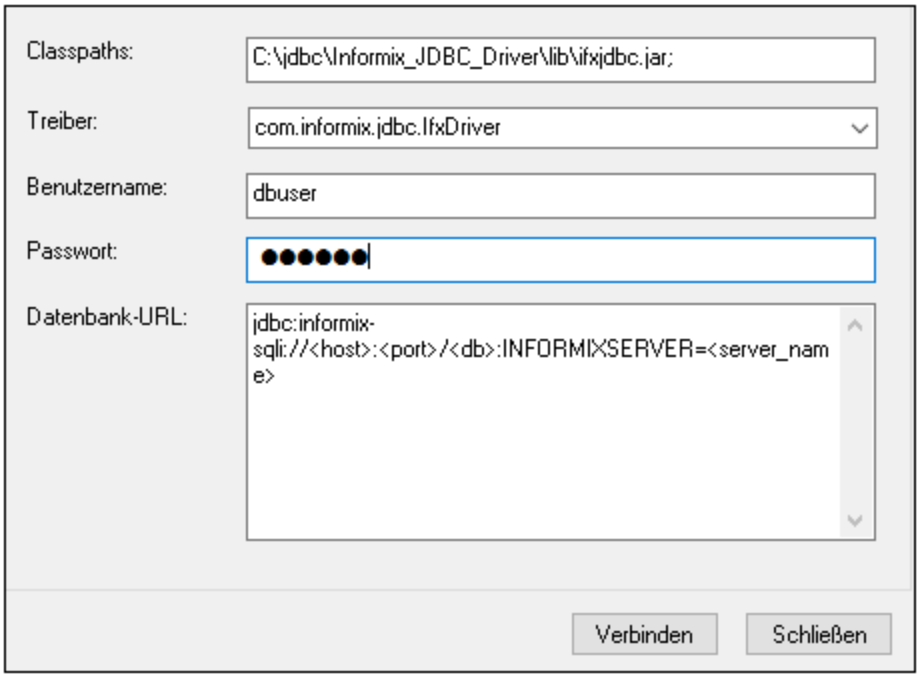
 © 2016-2022 Altova GmbH

3.70 verwendet. Die Installationsanleitung zum Treiber finden Sie in der dazugehörigen Dokumentation bzw. im "IBM Informix JDBC Driver Programmer's Guide").
- Sie haben die folgenden Datenbankinformationen zur Verfügung: Host, Name des Informix-Servers, Datenbank, Port. Benutzername und Passwort.

So stellen Sie über JDBC eine Verbindung zu IBM Informix her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **JDBC-Verbindungen**.
3. Geben Sie neben "Classpaths" den Pfad zur .jar-Datei , die die Verbindung zur Datenbank bereitstellt, ein. Falls nötig, können Sie auch eine durch Semikola getrennte Liste von .jar-Dateipfaden eingeben. Die benötigte .jar-Datei in diesem Beispiel befindet sich unter dem folgenden Pfad: **C:\Informix_JDBC_Driver\lib\ifxjdbc.jar**. Beachten Sie, dass Sie das Textfeld "Classpaths" leer lassen können, wenn Sie den/die .jar-Dateipfad(e) zur Umgebungsvariablen CLASSPATH des Betriebssystems hinzugefügt haben (siehe auch Konfigurieren des CLASSPATH[193]).

4. Wählen Sie im Feld "Treiber" **com.informix.jdbc.IfxDriver** aus. Beachten Sie, dass dieser Eintrag zur Verfügung steht, wenn entweder im Textfeld "Classpaths" oder in der Umgebungsvariablen CLASSPATH des Betriebssystems eine gültige .jar-Datei gefunden wird (siehe vorheriger Schritt).



5. Geben Sie den Benutzernamen und das Passwort für die Datenbank in die entsprechenden Textfelder ein.
6. Geben Sie den Connection String zum Datenbankserver in das Textfeld "Datenbank-URL" ein, indem Sie die hervorgehobenen Werte durch die entsprechenden Werte für Ihren Datenbankserver ersetzen.

```
jdbc:informix-sqli://hostName:port/databaseName:INFORMIXSERVER=myserver;
```

7. Klicken Sie auf **Verbinden.**

4.2.1.12.8 MariaDB (ODBC)

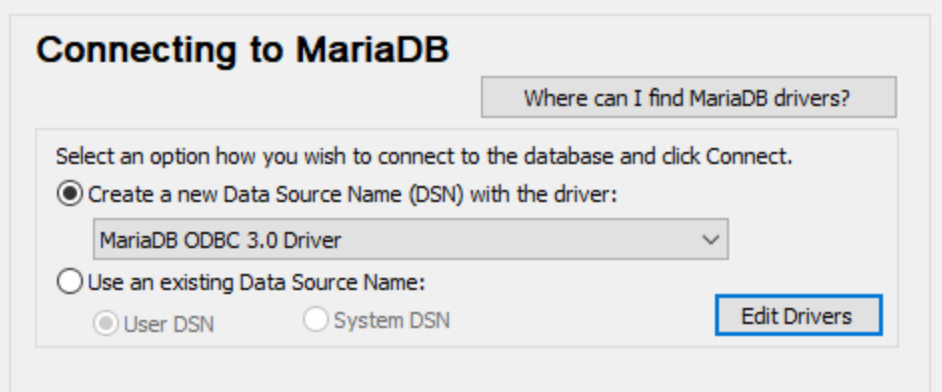
In diesem Beispiel wird gezeigt, wie Sie über ODBC eine Verbindung zu einem MariaDB-Datenbankserver herstellen.

Voraussetzungen:

- Der MariaDB Connector/ODBC (https://downloads.mariadb.org/connector-odbc/) muss installiert sein.
- Sie haben die folgenden Datenbankinformationen zur Verfügung: Host, Datenbank, Port. Benutzername und Passwort.

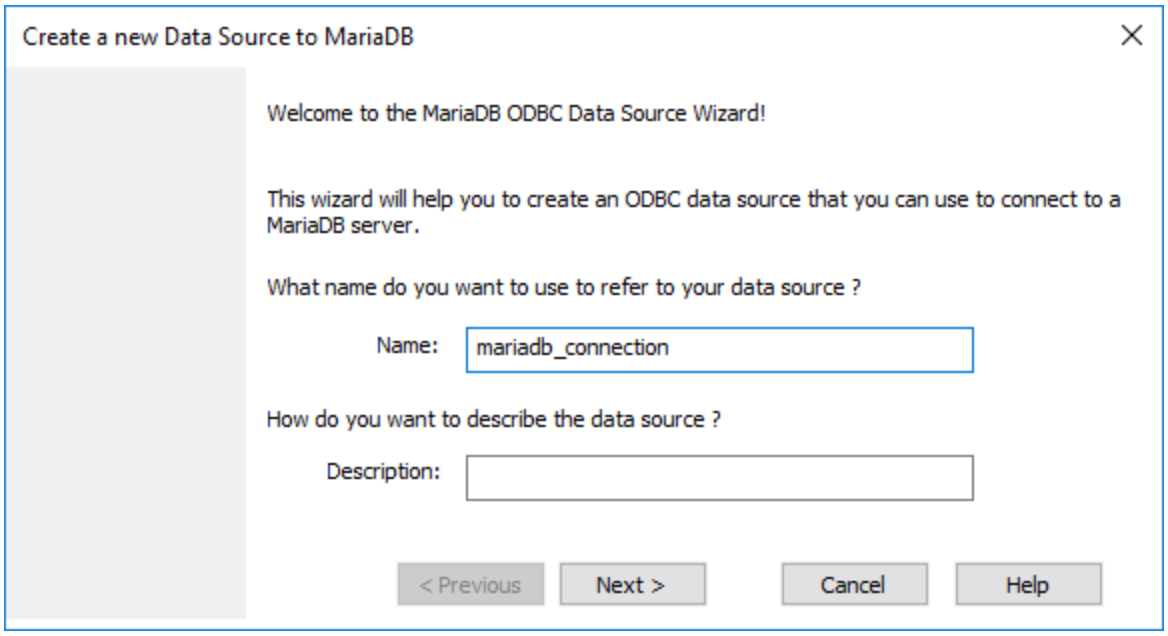
So stellen Sie über ODBC eine Verbindung zu MariaDB her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Wählen Sie **MariaDB (ODBC)** aus und klicken Sie auf **Weiter**.

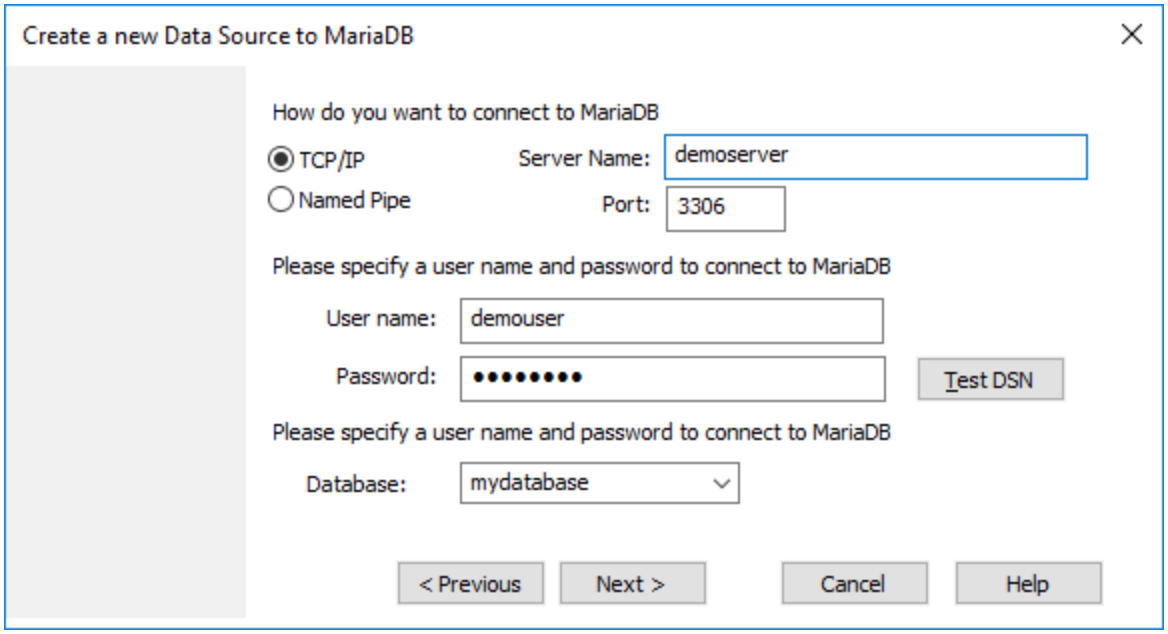


3. Wählen Sie **Erstelle neuen Data Source Name (DSN) mit dem Treiber** und wählen Sie **MariaDB ODBC 3.0 Driver** aus. Wenn in der Liste kein MySQL-Treiber verfügbar ist, klicken Sie auf **Treiber bearbeiten** und wählen Sie einen beliebigen verfügbaren MariaDB-Treiber aus (die Liste enthält alle auf Ihrem Betriebssystem installierten ODBC-Treiber).
4. Klicken Sie auf **Verbinden**.

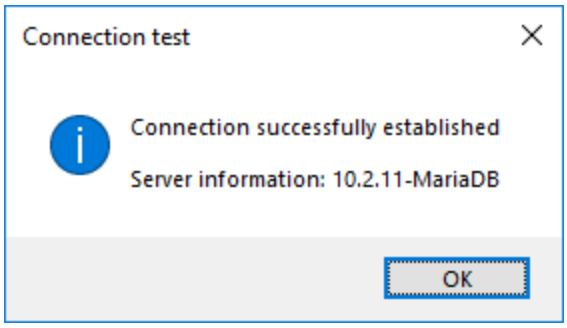
© 2016-2022 Altova GmbH



5. Geben Sie einen Namen und optional eine Beschreibung für diese ODBC-Datenquelle ein.



6. Füllen Sie die Anmeldeinformationen für die Datenbankverbindung aus (TCP/IP Server, Benutzer, Passwort), wählen Sie eine Datenbank aus und klicken Sie auf **DSN testen**. Wenn die Verbindung erfolgreich hergestellt werden konnte, wird eine Meldung angezeigt:



7. Klicken Sie auf **Weiter** und schließen Sie den Vorgang ab. Von Fall zu Fall werden eventuell auch andere Parameter benötigt (z.B. SSL-Zertifikate, wenn Sie eine sichere Verbindung zu MariaDB herstellen möchten).

Anmerkung: Wenn es sich um einen entfernten Datenbankserver handelt, muss er vom Server-Administrator so konfiguriert werden, dass er entfernte Verbindungen von der IP-Adresse Ihres Rechners aus zulässt.

4.2.1.12.9 Microsoft Access (ADO)

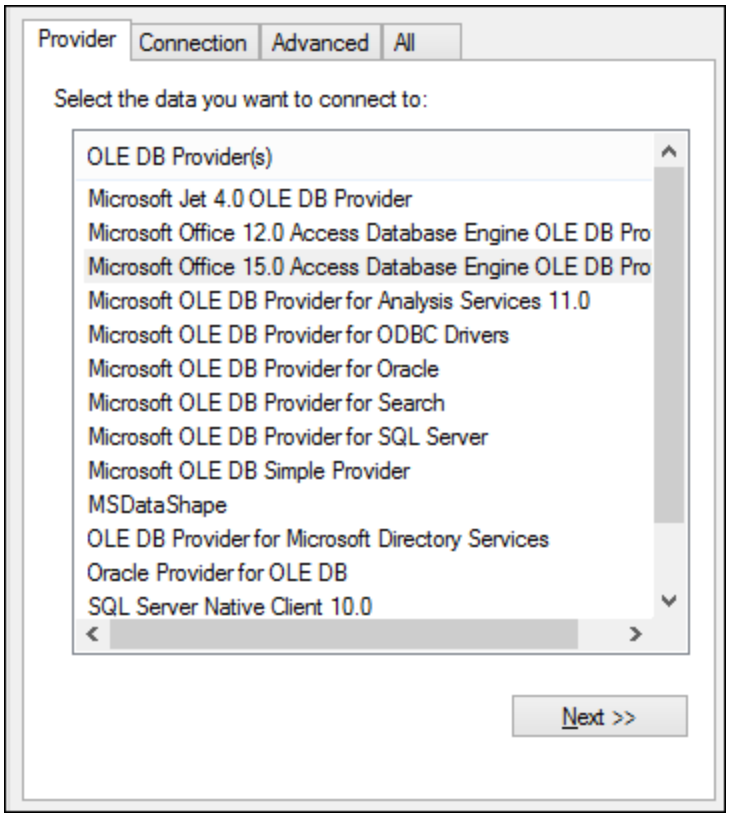
Eine einfache Methode, eine Verbindung zu einer Microsoft Access-Datenbank herzustellen, ist, den Anweisungen des Assistenten zu folgen und zur Datenbankdatei zu navigieren, wie unter Herstellen einer Verbindung zu einer vorhandenen Microsoft Access-Datenbank 177 beschrieben. Alternativ dazu können Sie explizit eine ADO-Verbindung definieren, wie im Folgenden gezeigt. Diese Methode empfiehlt sich, wenn Ihre Datenbank durch ein Passwort geschützt ist.

Sie können zwar auch über ODBC eine Verbindung zu Microsoft Access herstellen, doch sollte diese Methode vermieden werden, da sich dadurch einige Einschränkungen ergeben.

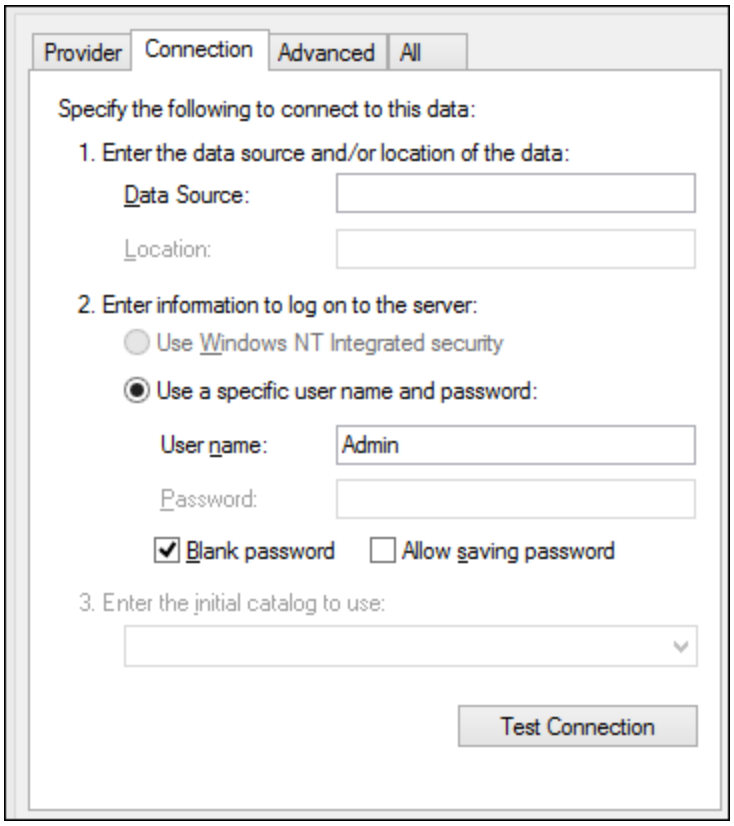
Herstellen einer Verbindung zu einer durch ein Passwort geschützten Microsoft Access-Datenbank:

1. Starten Sie den Datenbank-Verbindungsassistenten 169.
2. Klicken Sie auf **ADO-Verbindungen**.
3. Klicken Sie auf **Erzeugen**.

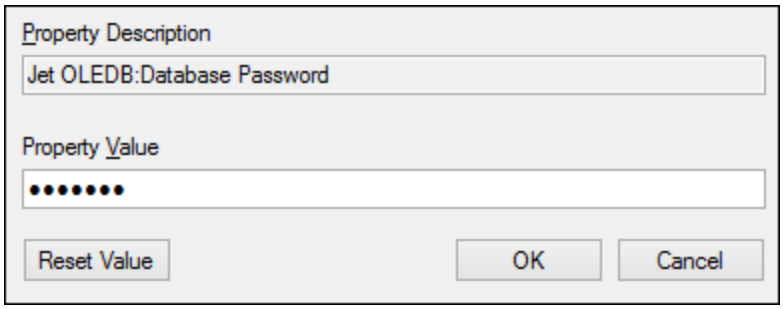
 © 2016-2022 Altova GmbH



4. Wählen Sie den **Microsoft Office 15.0 Access Database Engine OLE DB Provider** aus und klicken Sie auf **Weiter**.



5. Geben Sie in das Feld "Datenquelle" den Pfad zur Microsoft Access-Datei im UNC-Format ein, z.B. **\\\myserver\\mynetworkshare\Reports\Revenue.accdb**, wobei **myserver** der Name des Servers und **mynetworkshare** der Name des gemeinsamen Netzwerklaufwerks ist.
6. Doppelklicken Sie auf dem Register **Alle** auf die Eigenschaft **Jet OLEDB:Database Password** und geben Sie das Datenbank-Passwort als Eigenschaftswert ein.



Anmerkung: Wenn die Verbindung immer noch nicht hergestellt werden kann, suchen Sie die Arbeitsgruppen-Informationsdatei (**System.MDW**) für Ihr Benutzerprofil und setzen Sie den Wert der -Eigenschaft **Jet OLEDB: System** auf den Pfad der Datei **System.MDW**.

4.2.1.12.10 Microsoft SQL Server (ADO)

In diesem Beispiel wird gezeigt, wie Sie über ADO eine Verbindung zu einer SQL Server-Datenbank herstellen. Diese Anleitung gilt für die Verwendung des empfohlenen **Microsoft OLE DB-Treibers für SQL Server**

© 2016-2022 Altova GmbH

(MSOLEDBSQL), der von https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 heruntergeladen werden kann.

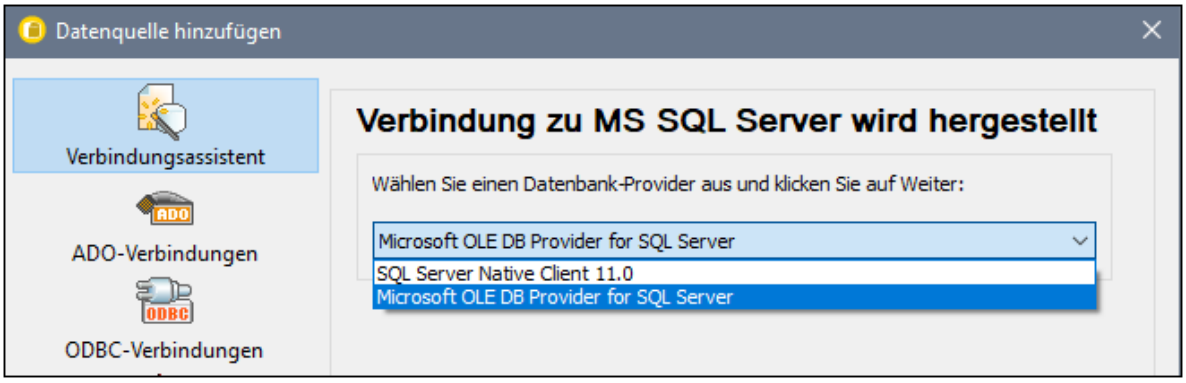
> Bevor Sie die Schritte aus dieser Anleitung durchführen, stellen Sie sicher, dass Sie den oben genannten Anbieter heruntergeladen und auf Ihrem Rechner installiert haben. Der ADO-Anbieter muss mit der Plattform von MapForce (32-Bit oder 64-Bit) übereinstimmen.

Wenn Sie andere ADO-Anbieter wie **SQL Server Native Client (SQLNCLI)** oder **Microsoft OLE DB-Anbieter für SQL Server (SQLOLEDB)** verwenden möchten, ist die Vorgangsweise ähnlich, doch sind diese Anbieter veraltet und werden daher nicht empfohlen. Damit die Verbindung zu einem veralteten Anbieter hergestellt werden kann, müssen Sie außerdem eventuell zusätzliche Verbindungseigenschaften konfigurieren, wie unter Einrichten der SQL Server-Datenverknüpfungseigenschaften 177 beschrieben.

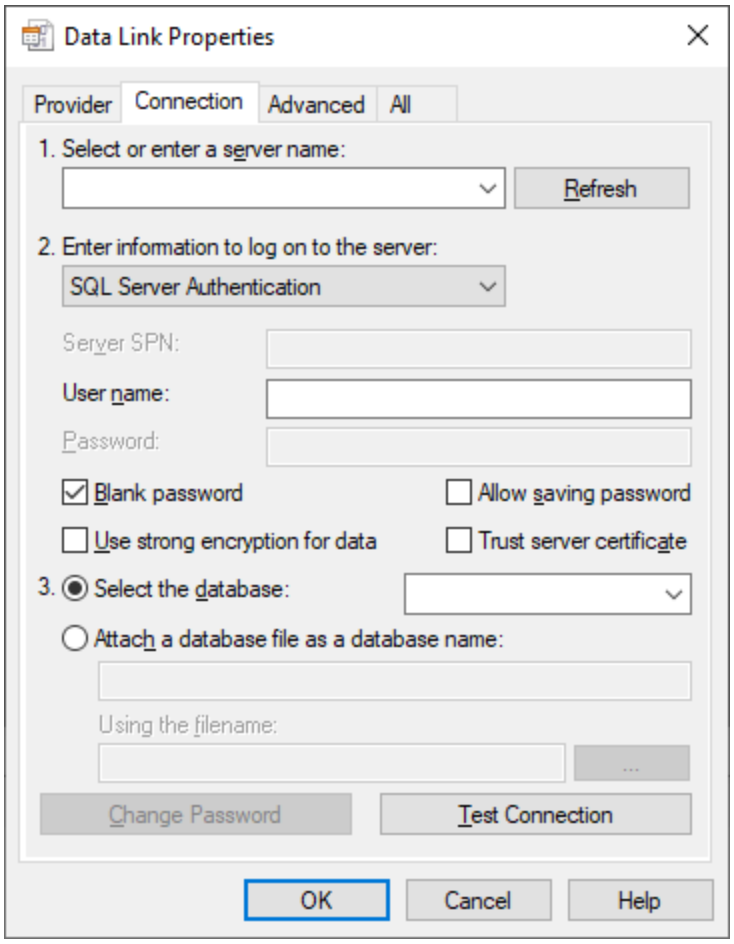
> Es ist bekannt, dass es beim **Microsoft OLE DB-Anbieter für SQL Server (SQLOLEDB)** zu Problemen mit der Parameterbindung komplexer Abfragen wie Common Table Expressions (CTE) und verschachtelten SELECT-Anweisungen kommt.

So stellen Sie eine Verbindung zu SQL Server her:

1. Starten Sie den Datenbank-Verbindungsassistenten 169.
2. Wählen Sie **Microsoft SQL Server (ADO)** und klicken Sie auf **Weiter**. Daraufhin wird die Liste der verfügbaren ADO-Anbieter angezeigt. In diesem Beispiel wird der **Microsoft OLE DB-Treiber für SQL Server** verwendet. Falls er in der Liste nicht enthalten ist, überprüfen Sie, ob er, wie oben erwähnt, auf Ihrem Rechner installiert ist.

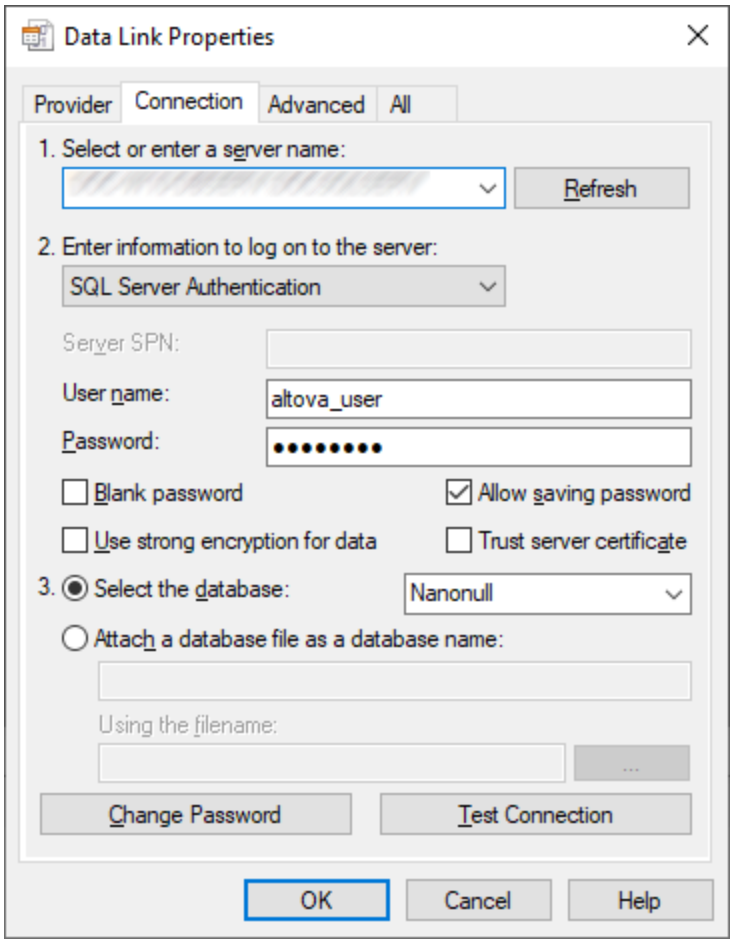


3. Klicken Sie auf **Weiter**. Daraufhin wird das Dialogfeld "Datenverknüpfungseigenschaften" angezeigt.



4. Wählen Sie den Namen des Datenbankservers aus oder geben Sie ihn ein (z.B. **SQLSERV01**). Wenn Sie eine Verbindung zu einer benannten SQL-Server-Instanz herstellen, sieht der Servername folgendermaßen aus: **SQLSERV01\INSTANZ**.
5. Wenn der Datenbankserver so konfiguriert ist, dass er Verbindungen von bei der Windows Domain angemeldeten Benutzern gestattet, wählen Sie **Windows-Authentifizierung**. Wählen Sie andernfalls **SQL Server-Authentifizierung**, deaktivieren Sie das Kontrollkästchen **Leeres Kennwort** und geben Sie die Anmeldeinformationen für die Datenbank in die entsprechenden Felder ein.
6. Aktivieren Sie das Kontrollkästchen **Passwort speichern zulassen** und wählen Sie die gewünschte Datenbank aus (in diesem Beispiel "Nanonull").

© 2016-2022 Altova GmbH



7. Um die Verbindung zu diesem Zeitpunkt zu überprüfen, klicken Sie auf **Verbindung testen**. Dieser Schritt ist optional, wird aber empfohlen.
8. Klicken Sie auf **OK.**

4.2.1.12.11 Microsoft SQL Server (ODBC)

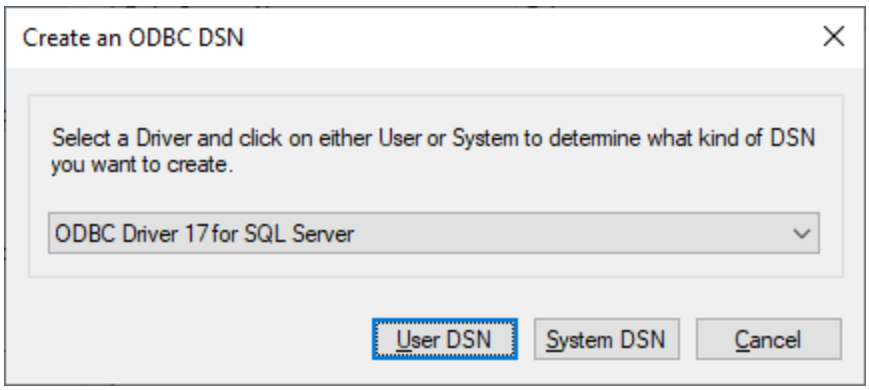
In diesem Beispiel wird gezeigt, wie Sie über ODBC eine Verbindung zu einer SQL Server-Datenbank herstellen.

Voraussetzungen:

- Laden Sie den **Microsoft ODBC-Treiber für SQL Server** von der Microsoft-Website herunter und installieren Sie ihn, siehe https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server. In diesem Beispiel wird für die Verbindung mit einer **SQL Server 2016**-Datenbank der **Microsoft ODBC-Treiber 17 für SQL Server** verwendet. Je nach gewünschter SQL-Server-Version müssen Sie eventuell eine andere ODBC-Treiberversion herunterladen. Informationen über von Ihrer SQL Server-Datenbank unterstützte ODBC-Treiberversionen schlagen Sie bitte unter den Systemanforderungen des Treibers nach.

So stellen Sie über ODBC eine Verbindung zu SQL Server her:

1. Starten Sie den Datenbank-Verbindungsassistenten 169.
2. Klicken Sie auf **ODBC-Verbindungen**.
3. Wählen Sie **Benutzer-DSN** (oder **System-DSN**, wenn Sie Administratorrechte haben) und klicken Sie auf **Hinzufügen** .
4. Wählen Sie den Treiber aus der Liste aus. Beachten Sie, dass der Treiber erst nach Installation auf der Liste angezeigt wird.

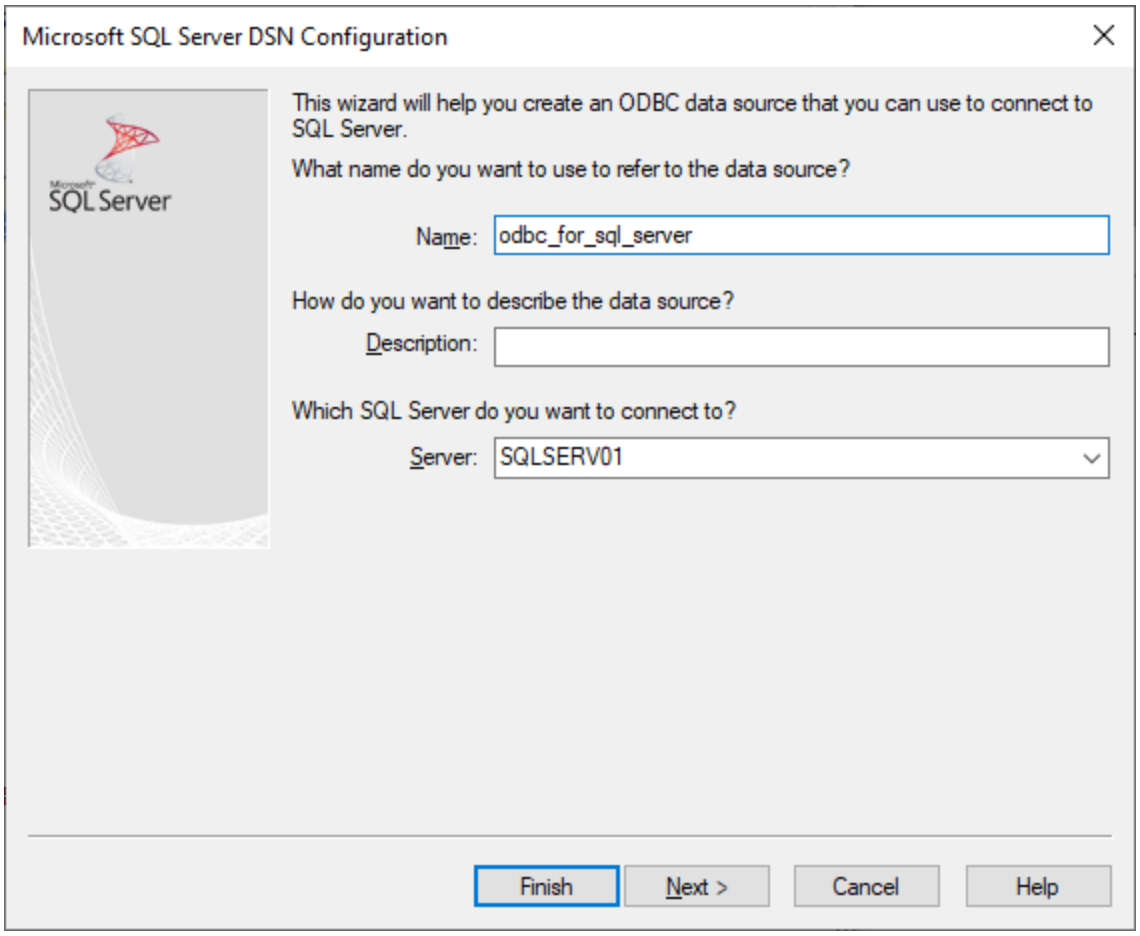


5. Wählen Sie **Benutzer-DSN** (oder **System-DSN**, wenn Sie einen System-DSN erstellen).

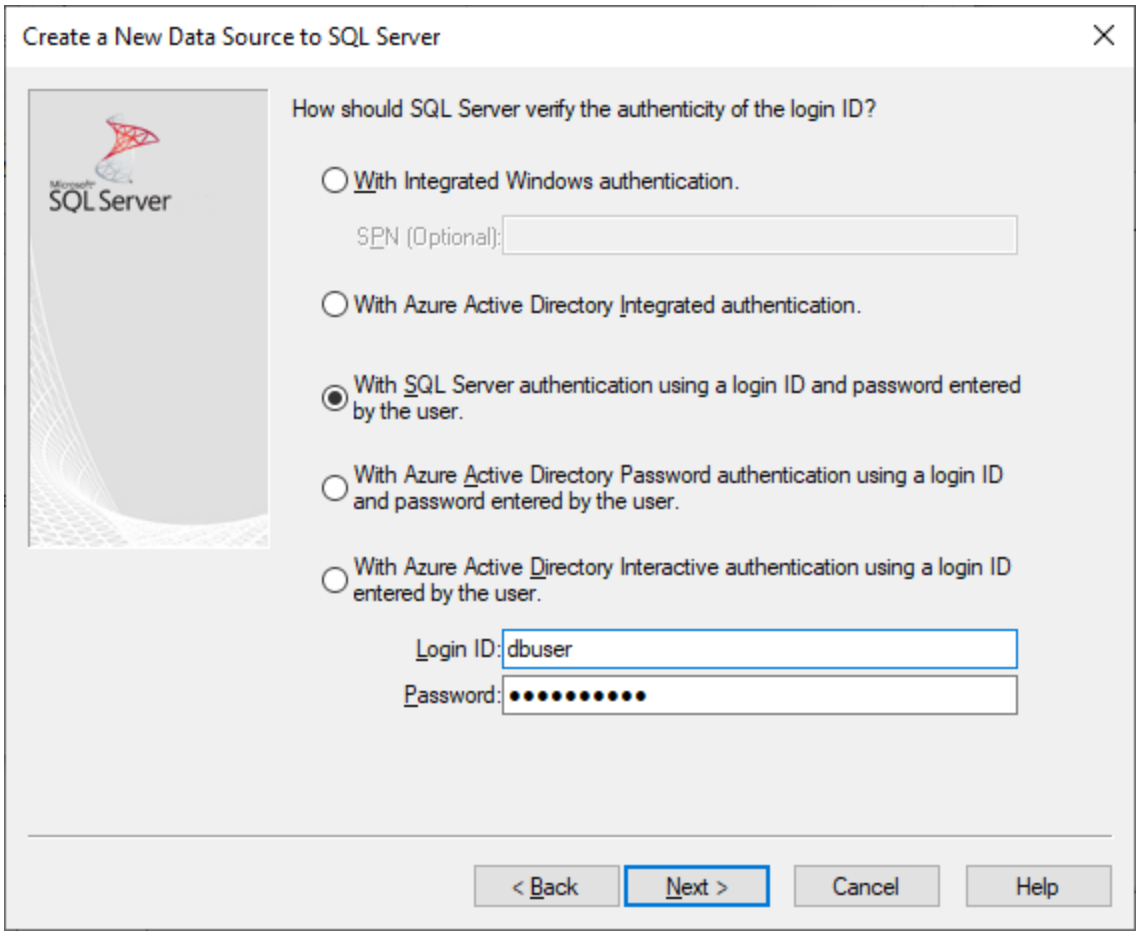
   Um einen **System-DSN** zu erstellen, müssen Sie MapForce als Administrator ausführen. Um daher einen **System-DSN** zu erstellen, brechen Sie den Assistenten ab, stellen Sie sicher, dass Sie MapForce als Administrator ausführen und führen Sie die obigen Schritte erneut durch.

6. Geben Sie einen Namen und optional eine Beschreibung für diese Verbindung ein und wählen Sie anschließend aus der Liste den gewünschten SQL Server aus (in diesem Beispiel **SQLSERV01**).

© 2016-2022 Altova GmbH

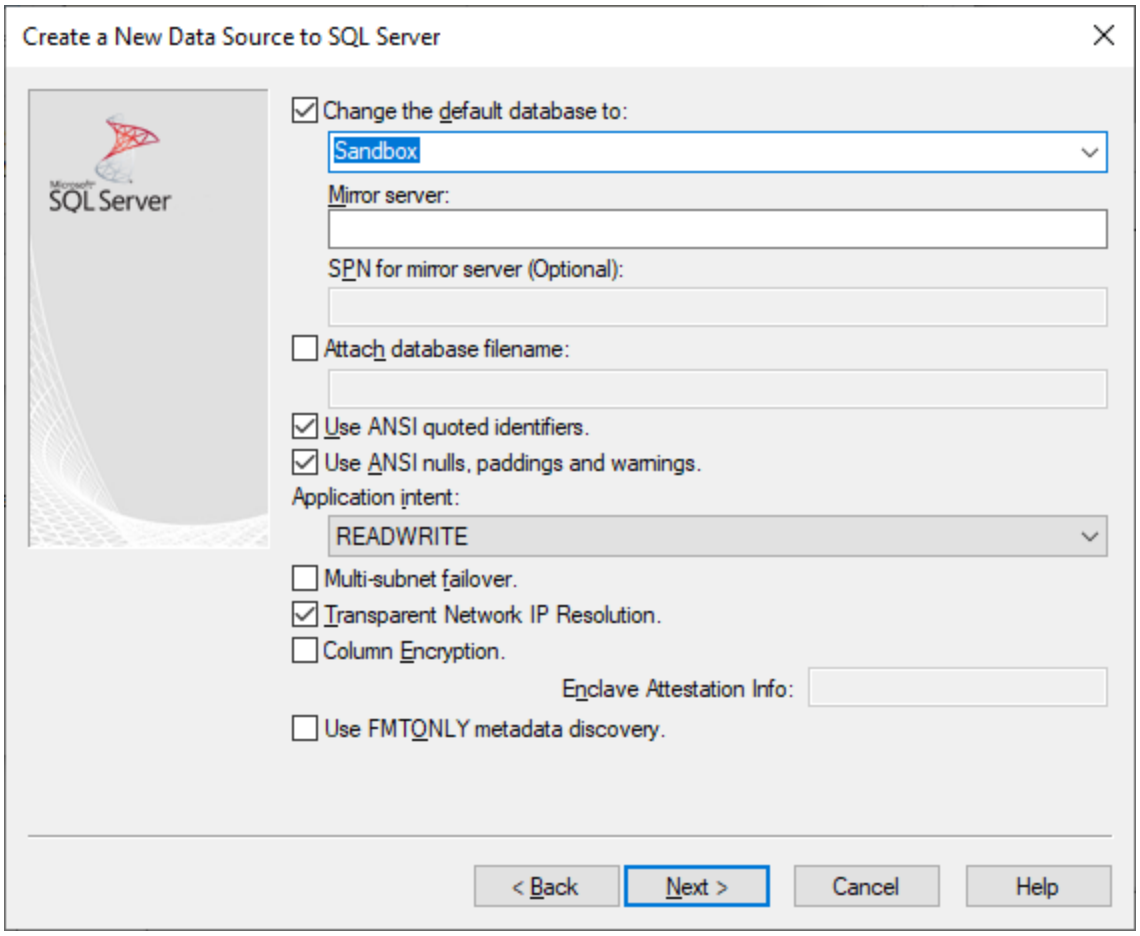


7. Wählen Sie die Option **Mit integrierter Windows NT-Authentifizierung**, wenn der Datenbankserver so konfiguriert ist, dass er Verbindungen von bei der Windows Domain angemeldeten Benutzern gestattet. Wählen Sie andernfalls je nach Bedarf eine der anderen Optionen aus. In diesem Beispiel wird **Mit SQL Server-Authentifizierung...** verwendet. In diesem Fall müssen Benutzername und Passwort in die entsprechenden Felder eingegeben werden.

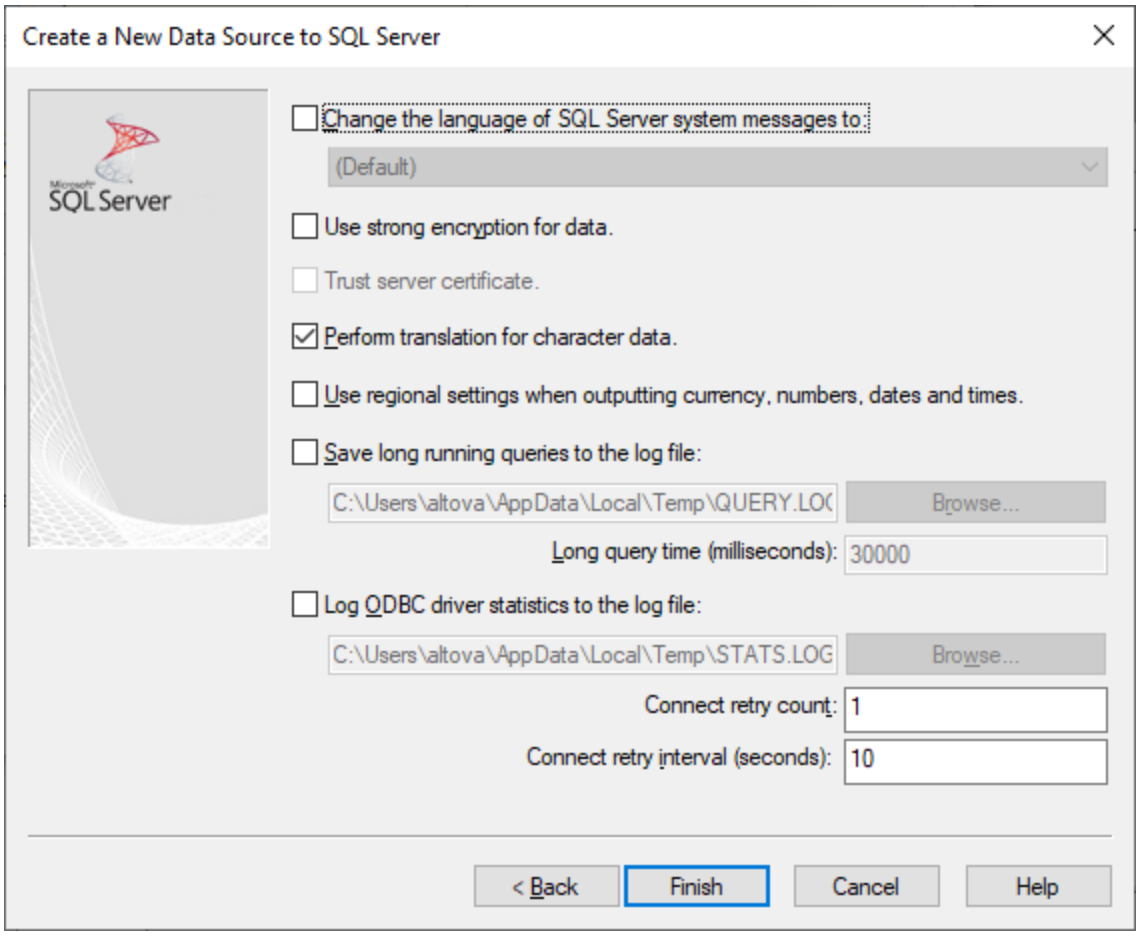


8. Aktivieren Sie optional das Kontrollkästchen **Standarddatenbank ändern in** und geben Sie den Namen der Datenbank, zu der Sie eine Verbindung herstellen (in diesem Beispiel **Sandbox**) ein.

© 2016-2022 Altova GmbH

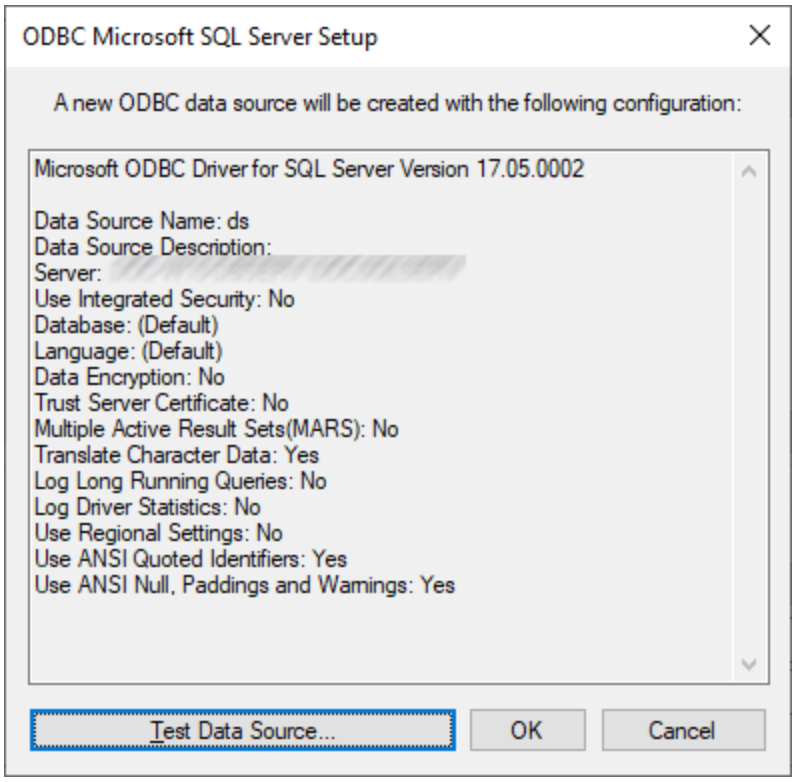


9. Klicken Sie auf **Weiter** und konfigurieren Sie optional weitere Parameter für diese Verbindung.

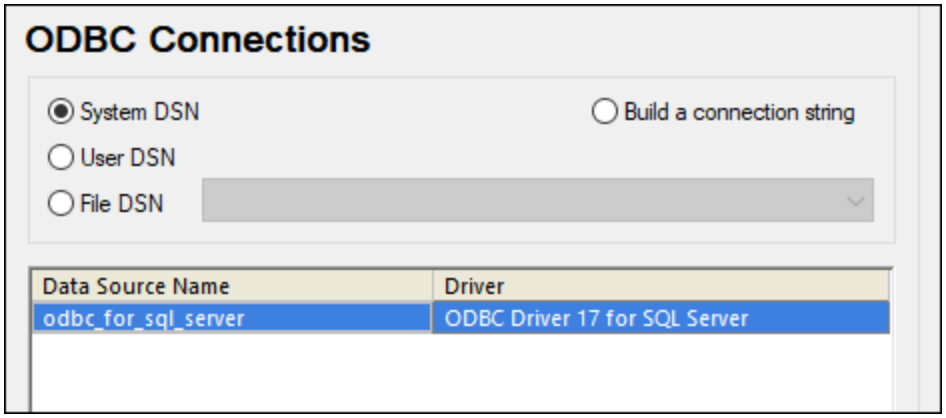


10. Klicken Sie auf **Fertig stellen**. Daraufhin wird ein Bestätigungsdialogfeld mit den Verbindungsinformationen angezeigt.

© 2016-2022 Altova GmbH



11. Klicken Sie auf **OK.** Daraufhin wird die Datenquelle je nach Konfiguration in der Liste der **Benutzer**- oder **System**-Datenquellen angezeigt, z.B:



4.2.1.12.12 MySQL (ODBC)

In diesem Kapitel wird gezeigt, wie Sie von einem Windows-Rechner aus über den ODBC-Treiber eine Verbindung zu einem MySQL-Datenbankserver herstellen. Der MySQL ODBC-Treiber steht auf Windows nicht zur Verfügung, daher müssen Sie ihn separat herunterladen und installieren. In diesem Beispiel wird die MySQL Connector/ODBC-Version 8.0 verwendet.

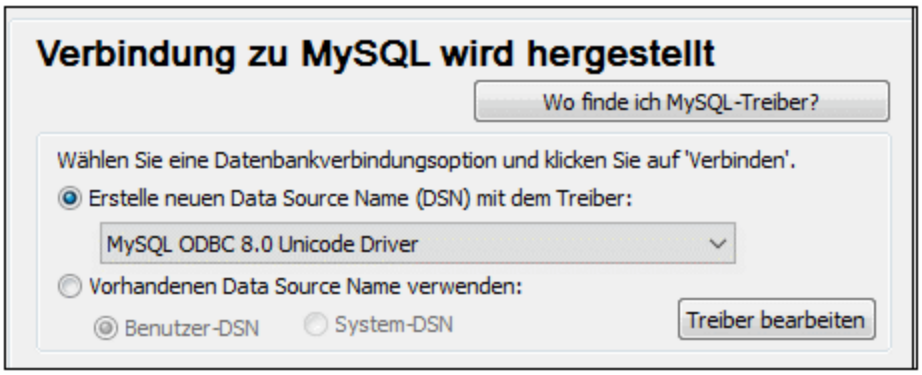
Voraussetzungen:

- Der MySQL ODBC-Treiber muss auf Ihrem Betriebssystem installiert sein. Lesen Sie nach in der Dokumentation zu MySQL, um zu ermittlen, welche Treiberversion für Ihre Datenbankserverversion empfohlen wird (siehe https://dev.mysql.com/downloads/connector/odbc/).
- Sie haben die folgenden Datenbankverbindungsinformationen zur Verfügung: Host, Datenbank, Port, Benutzername und Passwort.

Wenn Sie MySQL Connector/ODBC für die 64-Bit-Plattform installiert haben, stellen Sie sicher, dass auch MapForce für die 64-Bit-Plattform installiert ist.

So stellen Sie über ODBC eine Verbindung zu MySQL her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Wählen Sie **MySQL (ODBC)** und klicken Sie auf **Weiter**.

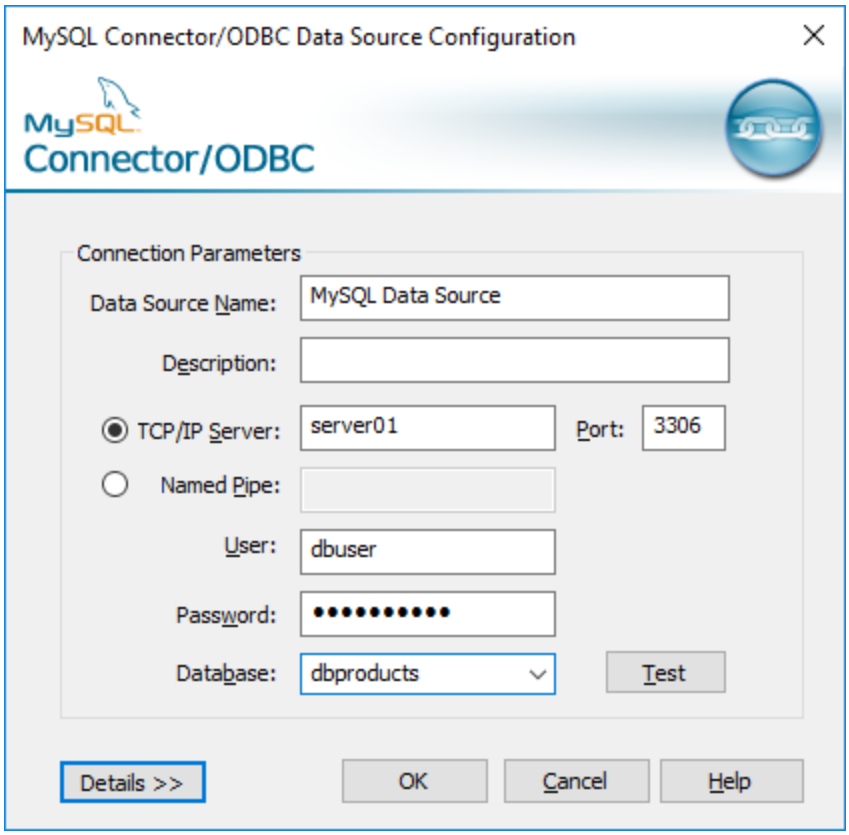


3. Wählen Sie **Erstelle neuen Data Source Name (DSN) mit dem Treiber** und wählen Sie einen MySQL-Treiber aus. Wenn in der Liste kein MySQL-Treiber verfügbar ist, klicken Sie auf **Treiber bearbeiten** und wählen Sie einen beliebigen verfügbaren MySQL-Treiber aus (die Liste enthält alle auf Ihrem Betriebssystem installierten ODBC-Treiber).

   Wenn Sie MapForce 64-Bit installiert haben, werden in der Liste die 64-Bit-ODBC-Treiber angezeigt. Andernfalls werden die 32-Bit-ODBC-Treiber angezeigt. Siehe auch Anzeigen der verfügbaren ODBC-Treiber[189].

4. Klicken Sie auf **Verbinden**.

© 2016-2022 Altova GmbH



5. Geben Sie in das Feld "Datenquellenname" einen Namen ein, anhand dessen Sie diese ODBC-Datenquelle in Zukunft identifizieren können.
6. Füllen Sie die Anmeldeinformationen für die Datenbankverbindung aus (TCP/IP Server, Benutzer, Passwort), wählen Sie eine Datenbank aus und klicken Sie auf **OK**.

Anmerkung: Wenn es sich um einen Remote-Datenbankserver handelt, muss er vom Server-Administrator so konfiguriert sein, dass er remote-Verbindungen von der IP-Adresse Ihres Rechners zulässt. Wenn Sie außerdem auf **Details>>** klicken, können Sie eine Reihe zusätzlicher Parameter konfigurieren. Lesen Sie die Dokumentation zum Treiber, bevor Sie die Standardwerte ändern.

4.2.1.12.13 Oracle (JDBC)

Dieses Kapitel enthält eine Beispielanleitung für das Herstellen einer Verbindung von einem Client-Rechner zu einem Oracle Datenbankserver mittels JDBC. Die Verbindung wird mit Hilfe des auf der Oracle Website verfügbaren **Oracle Instant Client Package (Basic)** als reine Java-Verbindung hergestellt. Der Vorteil dieser Verbindungsart ist, dass nur die Java-Umgebung und die vom Oracle Instant Client Package bereitgestellten .jar-Bibliotheken benötigt werden, sodass Sie keinen komplexeren Datenbank-Client installieren und konfigurieren müssen.

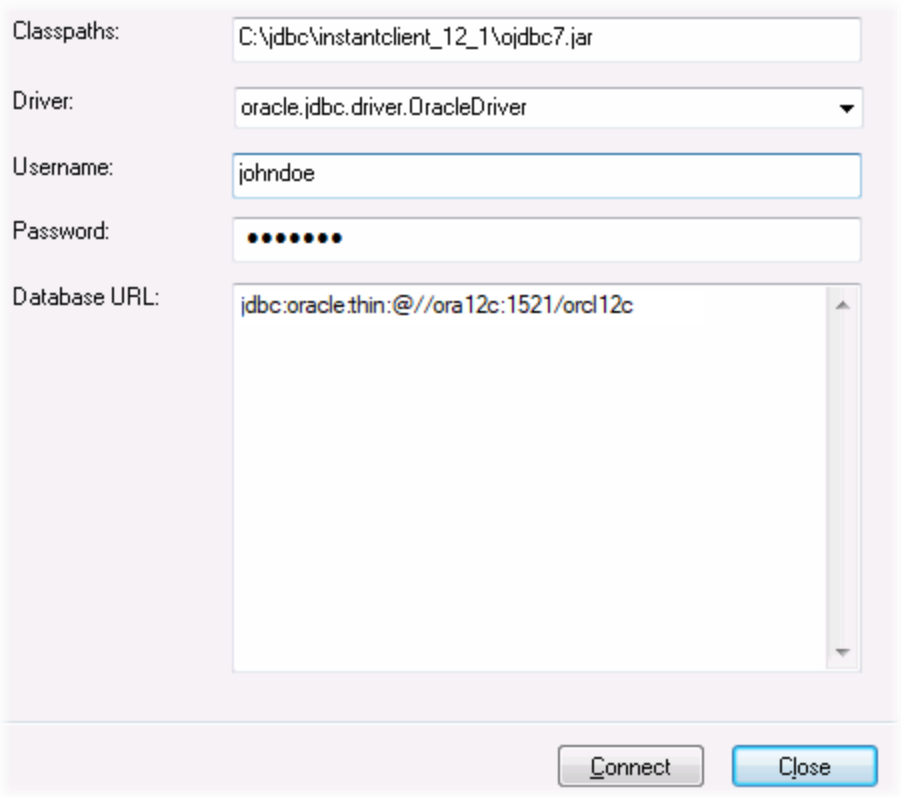
Voraussetzungen:

- Auf Ihrem Betriebssystem muss JRE (Java Runtime Enviroment) oder Java Development KIT (JDK) installiert sein. Dabei muss es sich entweder um Oracle JDK oder einen Open Source Build wie Oracle OpenJDK handeln.MapForce ermittelt den Pfad zur Java Virtual Machine (JVM) anhand der folgenden Ordner und zwar in folgender Reihenfolge: a) anhand des benutzerdefinierten JVM-Pfads, den Sie eventuell in den **Applikationsoptionen** definiert haben, siehe Java-Einstellungen[1542]; b) anhand des JVM-Pfads in der Windows Registry; c) anhand der `JAVA_HOME`-Umgebungsvariablen.
- Stellen Sie sicher, dass die Plattform von MapForce (32-Bit, 64-Bit) mit der des JRE/JDK übereinstimmt.
- **Oracle Instant Client Package (Basic)** muss auf Ihrem Betriebssystem verfügbar sein. Das Paket kann von der offiziellen Oracle Website heruntergeladen werden. In diesem Beispiel wird Oracle Instant Client Package Version 12.1.0.2.0 für Windows 32-Bit und folglich Oracle JDK 32-Bit verwendet.
- Sie haben die folgenden Datenbankverbindungsinformationen zur Verfügung: Host, Port, Servicename, Benutzername und Passwort.

So stellen Sie über das Instant Client Package eine Verbindung zu Oracle her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **JDBC-Verbindungen**.
3. Geben Sie neben "Classpaths" den Pfad zur .jar-Datei , die die Verbindung zur Datenbank bereitstellt, ein. Falls nötig, können Sie auch eine durch Semikola getrennte Liste von .jar-Dateipfaden eingeben. Die benötigte .jar-Datei in diesem Beispiel befindet sich unter dem folgenden Pfad: **C:\jdbc\instantclient_12_1\ojdbc7.jar**. Beachten Sie, dass Sie das Textfeld "Classpaths" leer lassen können, wenn Sie den/die .jar-Dateipfad(e) zur Umgebungsvariablen CLASSPATH des Betriebssystems hinzugefügt haben (siehe auch Konfigurieren des CLASSPATH[193]).
4. Wählen Sie im Feld "Treiber" **oracle.jdbc.driver.OracleDriver** oder **oracle.jdbc.driver.OracleDriver** aus. Beachten Sie, dass dieser Eintrag zur Verfügung steht, wenn entweder im Textfeld "Classpath" oder in der Umgebungsvariablen `CLASSPATH` des Betriebssystems eine gültige .jar-Datei gefunden wird (siehe vorheriger Schritt).
5. Geben Sie den Benutzernamen und das Passwort in die entsprechenden Textfelder ein.

© 2016-2022 Altova GmbH



6. Geben Sie in das Textfeld "Datenbank-URL" den Connection String zum Datenbankserver ein, indem Sie die hervorgehobenen Werte durch die entsprechenden Werte für Ihren Datenbankserver ersetzen.

```
jdbc:oracle:thin:@//host:port:service
```

7. Klicken Sie auf **Verbinden**.

4.2.1.12.14 Oracle (ODBC)

In diesem Beispiel wird ein häufig vorkommendes Szenario beschrieben: Sie stellen von MapForce aus über einen auf dem lokalen Betriebssystem installierten Oracle Datenbank Client eine Verbindung zu einer Oracle-Datenbank im Netzwerk her.

Das Beispiel enthält eine Anleitung, wie man mit Hilfe des Datenbankverbindungsassistenten in MapForce eine ODBC-Datenquelle (DSN) konfiguriert. Wenn Sie bereits einen DSN erstellt haben oder wenn Sie diesen lieber direkt über den ODBC-Datenquellen-Administrator in Windows erstellen, können Sie dies tun und den DSN dann auswählen, sobald Sie vom Assistenten dazu aufgefordert werden. Nähere Informationen zu ODBC-Datenquelle finden Sie unter Einrichten einer ODBC-Verbindung 187.

Voraussetzungen:

- Der Oracle Datenbank Client (der den Oracle-ODBC-Treiber enthält) muss auf Ihrem Betriebssystem installiert und konfiguriert sein. Eine Anleitung zum Installieren und Konfigurieren eines Oracle Datenbank Client finden Sie in der Dokumentation zur Oracle-Software.
- Die Datei **tnsnames.ora** im Oracle-Startverzeichnis enthält einen Eintrag, der die Datenbankverbindungsparameter in etwa wie folgt beschreibt:

```
ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = server01)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SID = orcl)
      (SERVER = DEDICATED)
    )
  )
```

Der Pfad zur Datei **tnsnames.ora** hängt davon ab, wo das Oracle-Startverzeichnis installiert wurde. Beim Oracle-Datenbank-Client 11.2.0 könnte der Standardpfad folgendermaßen lauten:

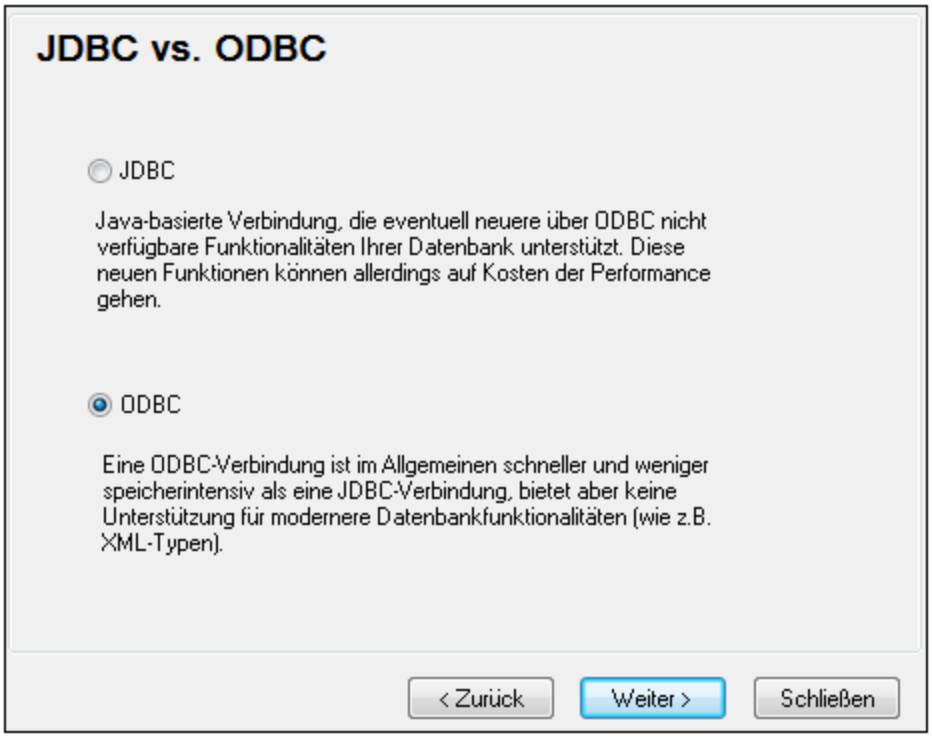
```
C:\app\username\product\11.2.0\client_1\network\admin\tnsnames.ora
```

Sie können neue Einträge zur Datei **tnsnames.ora** hinzufügen, indem Sie die Verbindungsdetails entweder hineinkopieren und die Datei speichern oder indem Sie den Oracle *Net-Konfigurationsassistenten* ausführen (falls vorhanden). Wenn diese Werte bei der Konfiguration in Dropdown-Listen aufscheinen sollen, müssen Sie den Pfad zum admin-Ordner eventuell als **TNS_ADMIN**-Umgebungsvariable hinzufügen.

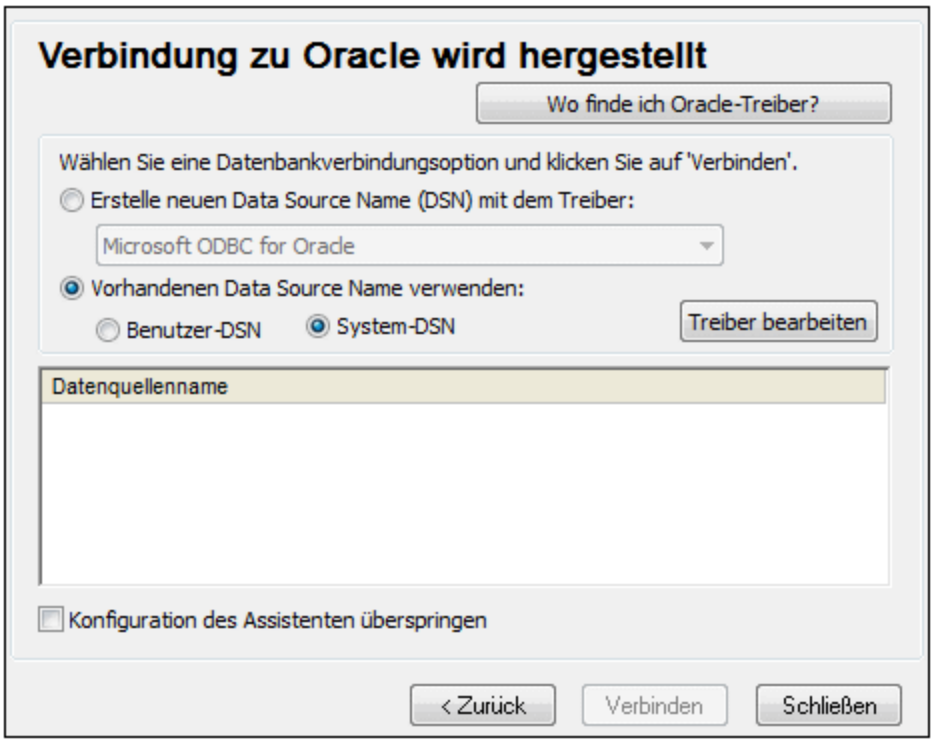
So stellen Sie über ODBC eine Verbindung zu Oracle her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Wählen Sie **Oracle (ODBC / JDBC)** und klicken Sie auf **Weiter**.

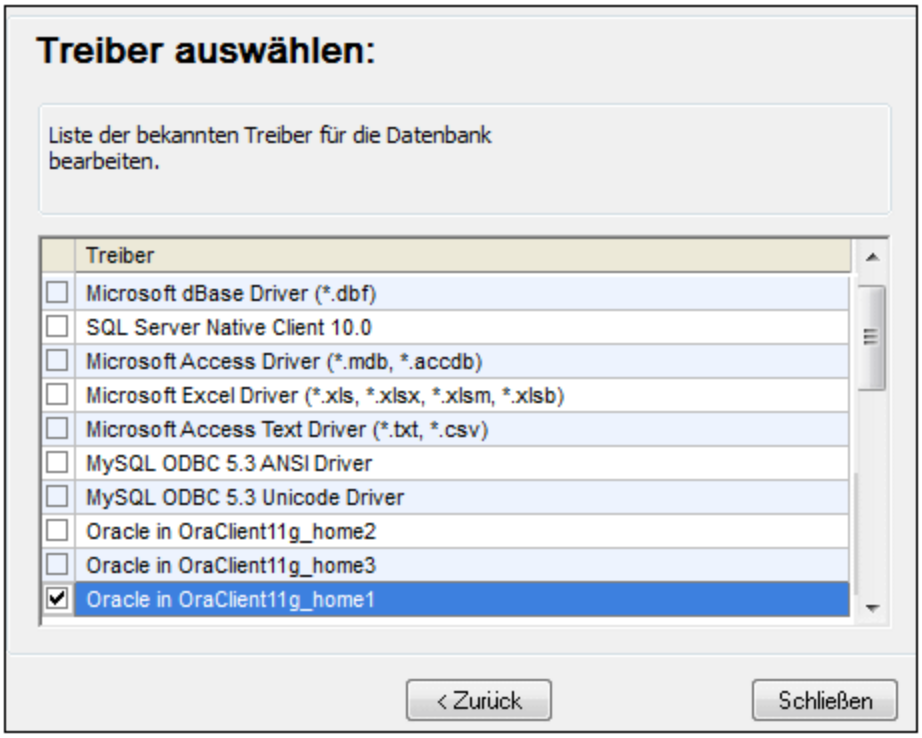
© 2016-2022 Altova GmbH



3. Wählen Sie **ODBC**.

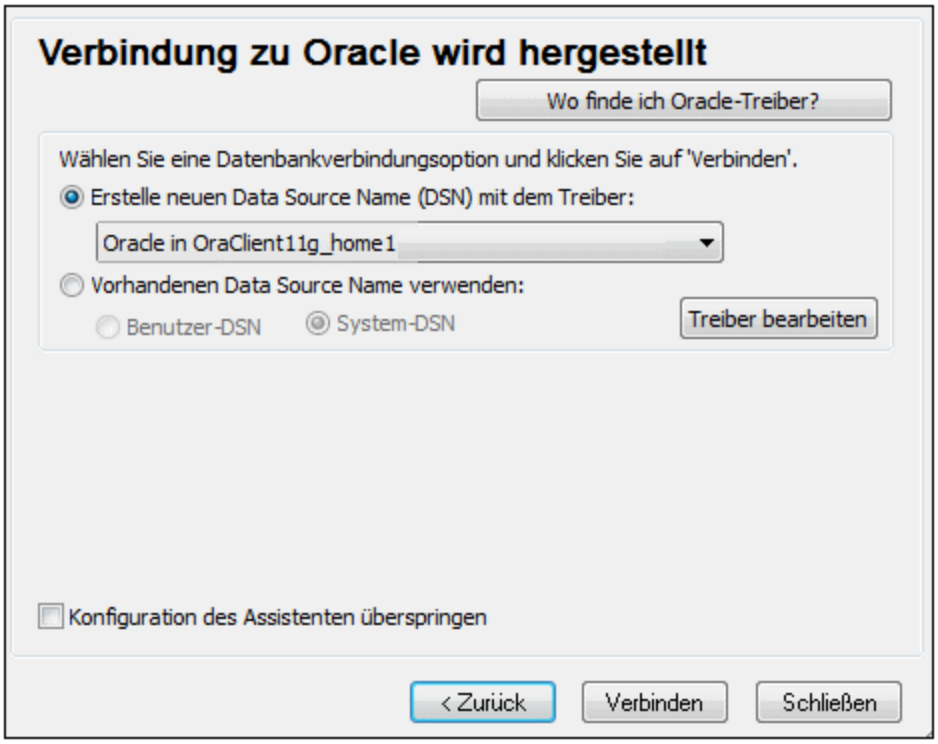


4. Klicken Sie auf **Treiber bearbeiten**.



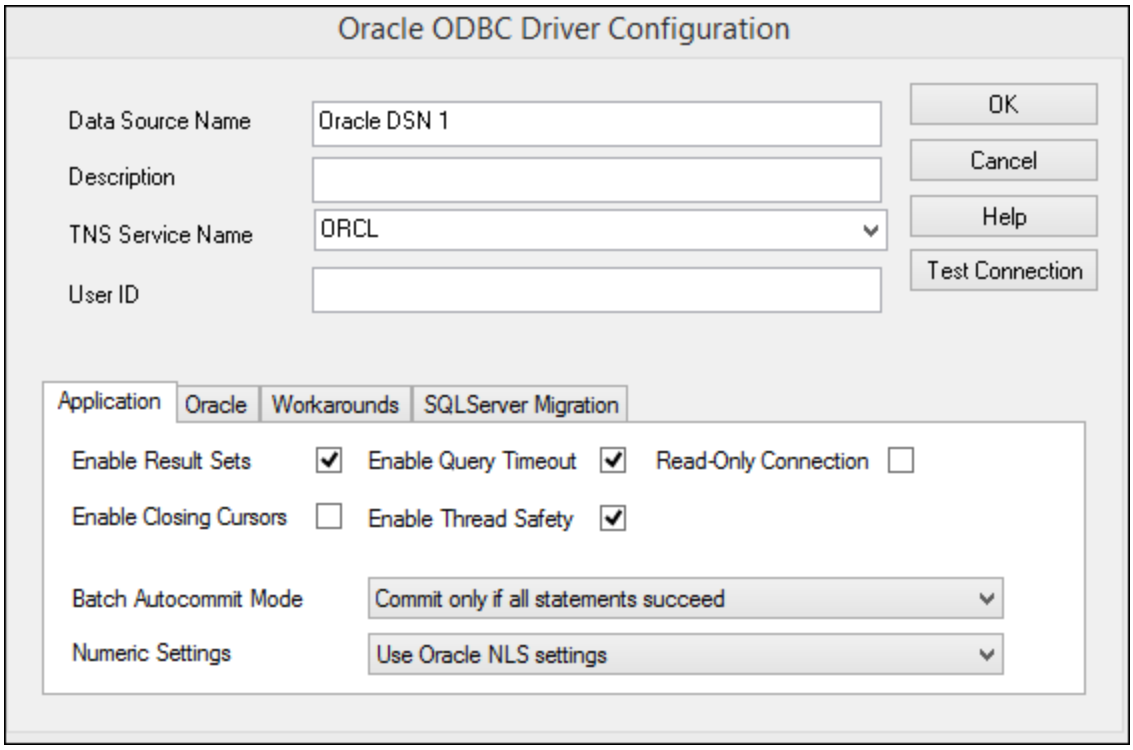
5. Wählen Sie die gewünschten Oracle-Treiber aus (in diesem Beispiel **Oracle in OraClient11g_home1**). In der Liste werden die nach der Installation des Oracle Client auf Ihrem System verfügbaren Oracle-Treiber angezeigt.
6. Klicken Sie auf **Zurück.**
7. Wählen Sie **Erstelle neuen Data Source Name (DSN) mit dem Treiber** und wählen Sie den in Schritt 4 ausgewählten Oracle-Treiber aus.

© 2016-2022 Altova GmbH

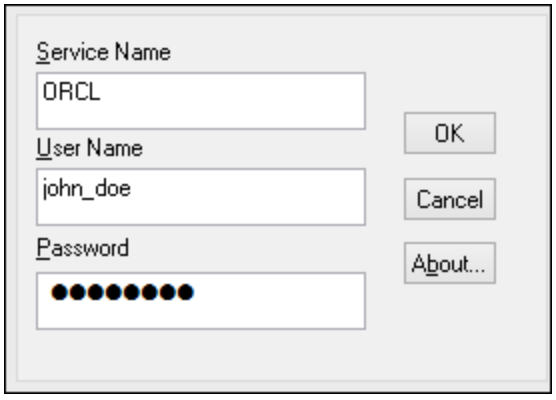


Verwenden Sie den von Microsoft bereitgestellten Treiber **Microsoft ODBC for Oracle** möglichst nicht. Microsoft empfiehlt, den von Oracle bereitgestellten ODBC-Treiber zu verwenden (siehe http://msdn.microsoft.com/en-us/library/ms714756%28v=vs.85%29.aspx)

8. Klicken Sie auf **Verbinden**.



9. Geben Sie im Textfeld "Datenquellenname" einen Namen für die Datenquelle ein (in diesem Beispiel **Oracle DSN 1**).
10. Geben Sie im Feld "TNS-Dienstname" den in der Datei **tnsnames.ora** definierten Verbindungsnamen ein (siehe Voraussetzungen[236]). In diesem Beispiel lautet der Verbindungsname **ORCL**. *Anmerkung:* Wenn die Dropdown-Liste der Auswahlliste mit den Werten der **tnsnames.ora**-Datei befüllt werden soll, müssen Sie den Pfad zum admin-Ordner als **TNS_ADMIN**-Umgebungsvariable hinzufügen.
11. Klicken Sie auf **OK**.



12. Geben sie den Benutzernamen und das Passwort für die Datenbank ein und klicken Sie auf OK.

4.2.1.12.15 PostgreSQL (ODBC)

Dieses Kapitel enthält eine Beispielanleitung für das Herstellen einer Verbindung über den ODBC-Treiber von einem Windows-Rechner zu einem PostgreSQL Datenbankserver. Der PostgreSQL ODBC-Treiber ist auf

© 2016-2022 Altova GmbH

Windows nicht verfügbar und muss separat heruntergeladen und installiert werden. In diesem Beispiel wird der von der offiziellen Website heruntergeladene psqlODBC-Treiber (Version 11.0) verwendet (siehe auch Übersicht über Datenbanktreiber[171]).

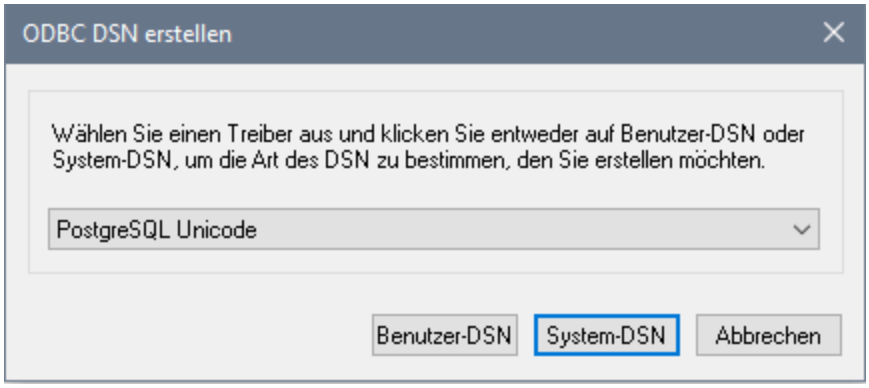
Anmerkung: Sie können die Verbindung zu einer PostgreSQL-Datenbank auch direkt (ohne ODBC-Treiber) herstellen, siehe Einrichten einer PostgreSQL-Verbindung[195].

Voraussetzungen:

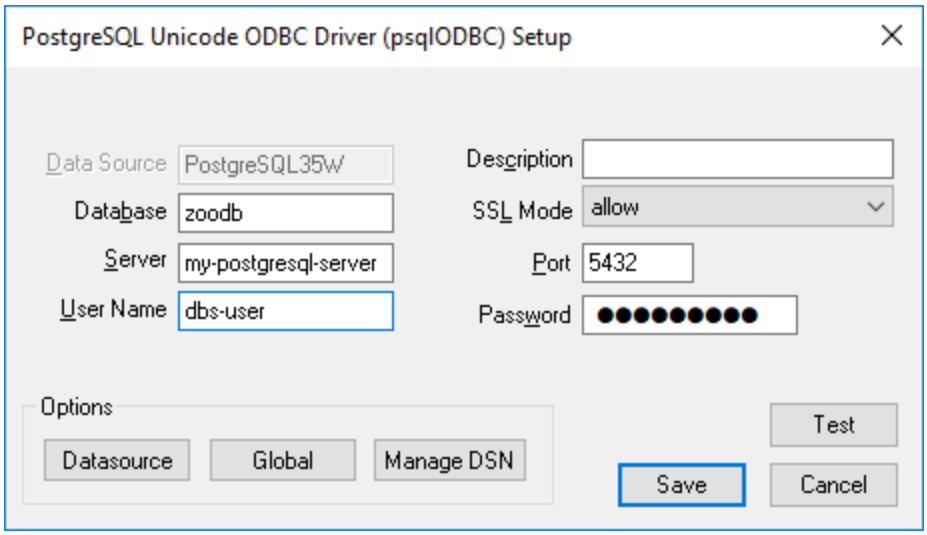
- Der *psqlODBC*-Treiber muss auf Ihrem Betriebssystem installiert sein.
- Sie haben die folgenden Datenbankverbindungsinformationen zur Verfügung: Server, Port, Datenbank, Benutzername und Passwort.

So richten Sie über ODBC eine Verbindung zu PostgreSQL ein:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **ODBC-Verbindungen**.
3. Aktivieren Sie die Option **Benutzer-DSN**.
4. Klicken Sie auf **Neuen DSN erstellen** und wählen Sie den Treiber aus der Dropdown-Liste aus. Wenn in der Liste kein PostgreSQL-Treiber zur Verfügung steht, stellen Sie sicher, dass der PostgreSQL ODBC-Treiber auf Ihrem Betriebssystem installiert ist, wie in den Voraussetzungen oben erwähnt.

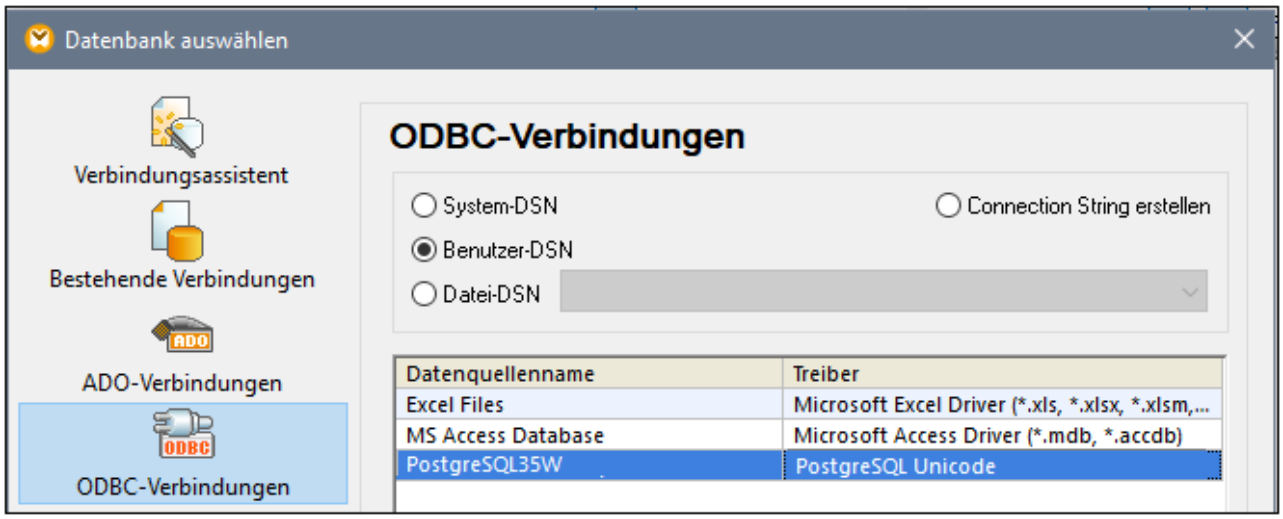


5. Klicken Sie auf **Benutzer-DSN**.



6. Füllen Sie die Anmeldeinformationen für die Datenbankverbindung aus (diese müssen vom Inhaber der Datenbank bereitgestellt werden) und klicken Sie anschließend auf **Speichern**.

Die Verbindung steht nun in der Liste der ODBC-Verbindungen zur Verfügung. Um eine Verbindung zur Datenbank herzustellen, können Sie entweder auf die Verbindung doppelklicken oder die Verbindung auswählen und auf **Verbinden** klicken.



4.2.1.12.16 Progress OpenEdge (JDBC)

Dieses Kapitel enthält eine Beispielanleitung für das Herstellen einer Verbindung zu einem Progress OpenEdge 11.6-Datenbankserver mittels JDBC.

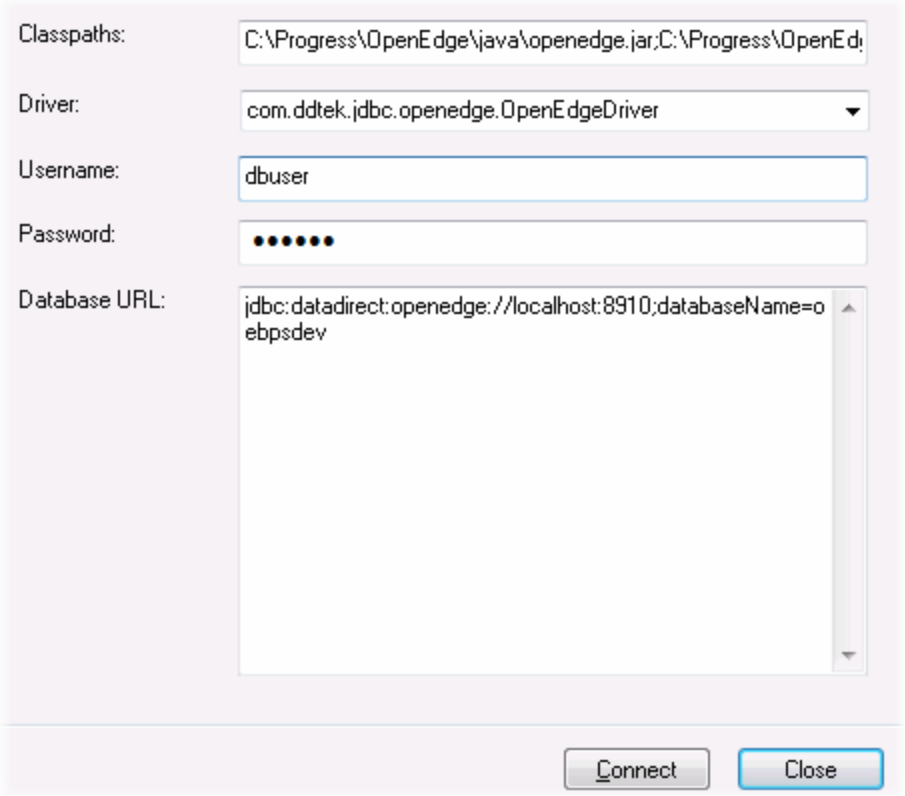
 © 2016-2022 Altova GmbH

Voraussetzungen:

- Auf Ihrem Betriebssystem muss JRE (Java Runtime Enviroment) oder Java Development KIT (JDK) installiert sein. Dabei muss es sich entweder um Oracle JDK oder einen Open Source Build wie Oracle OpenJDK handeln.MapForce ermittelt den Pfad zur Java Virtual Machine (JVM) anhand der folgenden Ordner und zwar in folgender Reihenfolge: a) anhand des benutzerdefinierten JVM-Pfads, den Sie eventuell in den **Applikationsoptionen** definiert haben, siehe Java-Einstellungen[1542]; b) anhand des JVM-Pfads in der Windows Registry; c) anhand der `JAVA_HOME`-Umgebungsvariablen.
- Stellen Sie sicher, dass die Plattform von MapForce (32-Bit, 64-Bit) mit der des JRE/JDK übereinstimmt.
- Die `PATH`-Umgebungsvariable des Betriebssystems muss den Pfad zum `bin`-Verzeichnis des JRE- bzw. JDK-Installationsverzeichnisses enthalten, z.B. `C:\Programe (x86)\Java\jre1.8.0_51\bin`.
- Der Progress OpenEdge JDBC-Treiber muss auf Ihrem Betriebssystem installiert sein. In diesem Beispiel erfolgt die JDBC-Verbindung über die Treiberkomponentendateien **openedge.jar** und **pool.jar,** die als Teil der OpenEdge SDK-Installation unter **C:\Progress\OpenEdge\java** zur Verfügung stehen.
- Sie haben die folgenden Datenbankverbindungsinformationen zur Verfügung: Host, Port, Datenbankname, Benutzername und Passwort.

So stellen Sie über JDBC eine Verbindung zu OpenEdge her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **JDBC-Verbindungen**.
3. Geben Sie neben "Classpaths" den Pfad zur .jar-Datei , die die Verbindung zur Datenbank bereitstellt, ein. Falls nötig, können Sie auch eine durch Semikola getrennte Liste von .jar-Dateipfaden eingeben. Die benötigte .jar-Datei in diesem Beispiel befindet sich unter dem folgenden Pfad: **`C:\Progress\OpenEdge\java\openedge.jar;C:\Progress\OpenEdge\java\pool.jar;`**. Beachten Sie, dass Sie das Textfeld "Classpaths" leer lassen können, wenn Sie den/die .jar-Dateipfad(e) zur Umgebungsvariablen CLASSPATH des Betriebssystems hinzugefügt haben (siehe auch Konfigurieren des CLASSPATH[193]).
4. Wählen Sie im Feld "Treiber" **com.ddtek.jdbc.openedge.OpenEdgeDriver** aus. Beachten Sie, dass dieser Eintrag zur Verfügung steht, wenn entweder im Textfeld "Classpaths" oder in der Umgebungsvariablen `CLASSPATH` des Betriebssystems eine gültige .jar-Datei gefunden wird (siehe vorheriger Schritt).



5. Geben Sie den Benutzernamen und das Passwort für die Datenbank in die entsprechenden Textfelder ein.
6. Geben Sie in das Textfeld "Datenbank-URL" den Connection String zum Datenbankserver ein, indem Sie die hervorgehobenen Werte durch die entsprechenden Werte für Ihren Datenbankserver ersetzen.

```
jdbc:datadirect:openedge://host:port;databaseName=db_name
```

7. Klicken Sie auf **Verbinden**.

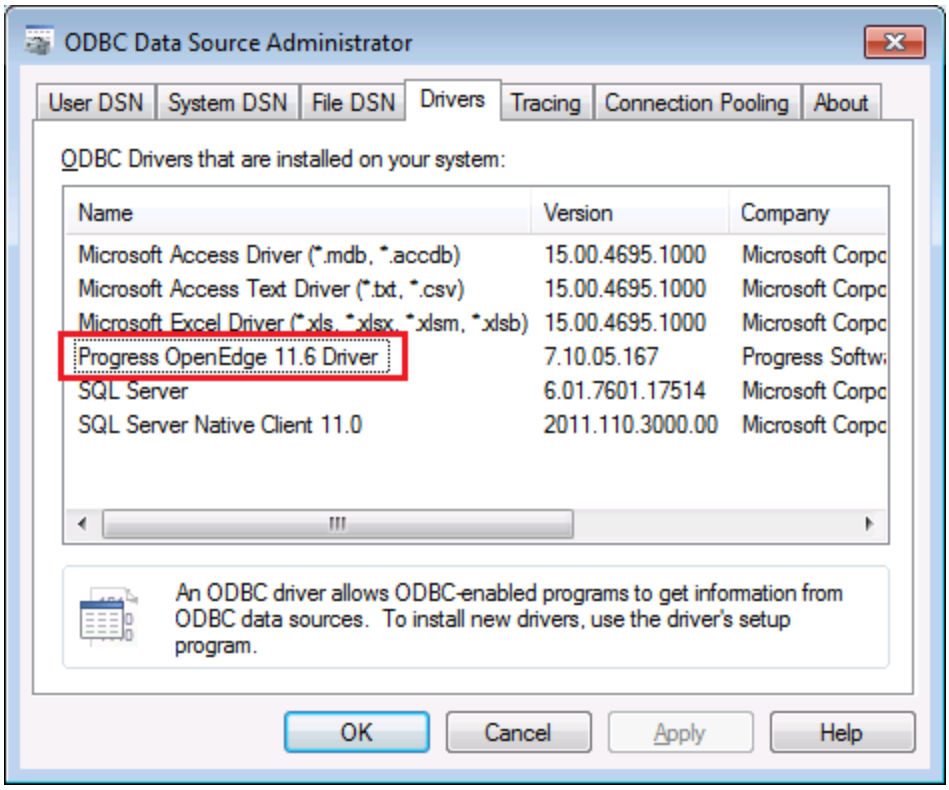
4.2.1.12.17 Progress OpenEdge (ODBC)

Dieses Kapitel enthält eine Beispielanleitung für das Herstellen einer Verbindung zu einer Progress OpenEdge-Datenbank über den Progress OpenEdge 11.6 ODBC-Treiber.

Voraussetzungen:

- Der *ODBC Connector for Progress OpenEdge*-Treiber muss auf Ihrem Betriebssystem installiert sein. Der Progress OpenEdge ODBC-Treiber kann von der Website des Anbieters heruntergeladen werden (siehe auch Übersicht über Datenbanktreiber[171]). Bei Ausführung der 32-Bit-Version von MapForce muss der 32-Bit-Treiber und bei Ausführung der 64-Bit-Version der 64-Bit-Treiber heruntergeladen werden. Überprüfen Sie nach Abschluss der Installation, ob der ODBC-Treiber auf Ihrem Rechner zur Verfügung steht (siehe auch Anzeigen der verfügbaren ODBC-Treiber[189]).

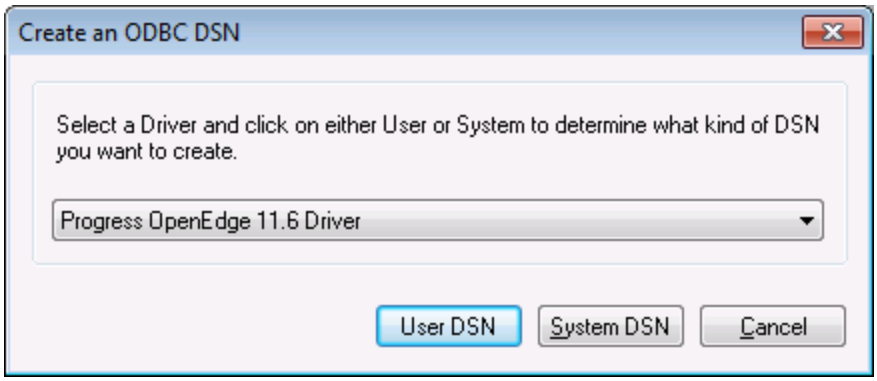
© 2016-2022 Altova GmbH



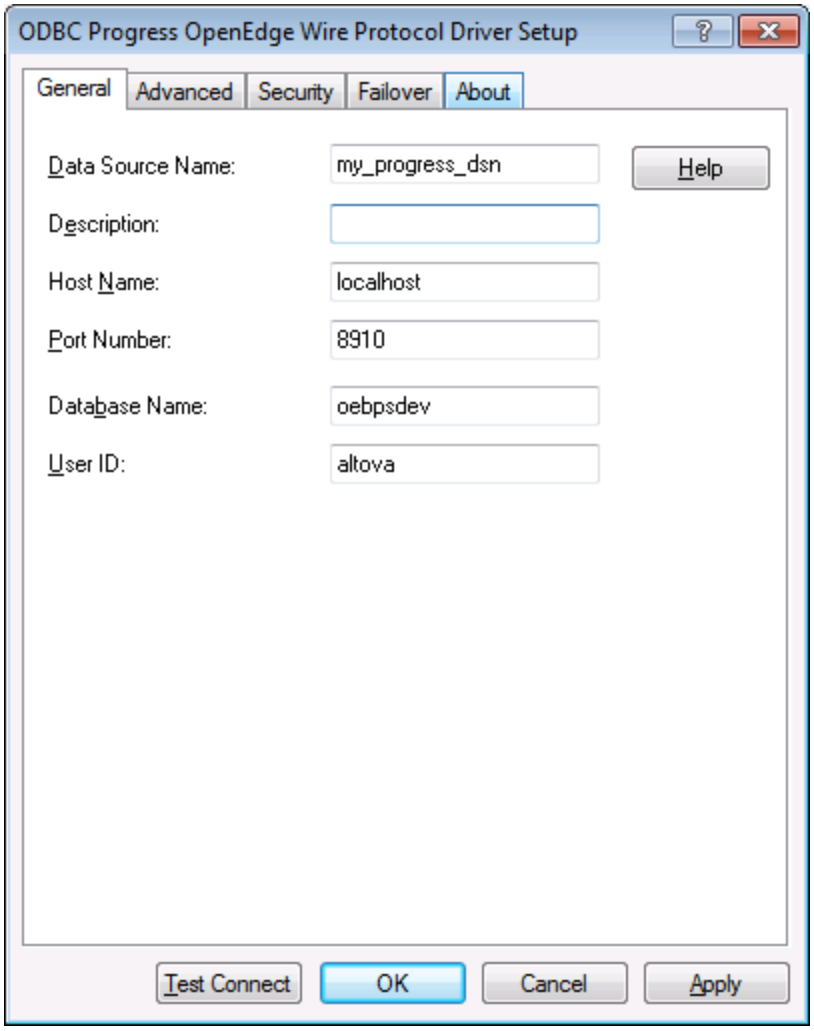
- Sie verfügen über die folgenden Datenbankverbindungsinformationen: Host-Name, Datenbankname, Benutzer-ID und Passwort.

So stellen Sie über ODBC eine Verbindung zu Progress OpenEdge her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **ODBC-Verbindungen**.
3. Wählen Sie Benutzer-DSN (oder alternativ dazu **System-DSN** oder **Datei-DSN** - in diesem Fall sind die nun folgenden Schritte ähnlich).
4. Klicken Sie auf **Hinzufügen** .
2. Wählen Sie aus der Liste **Progress OpenEdge Driver** aus und klicken Sie auf **Benutzer-DSN** (oder gegebenenfalls auf **System-DSN**).

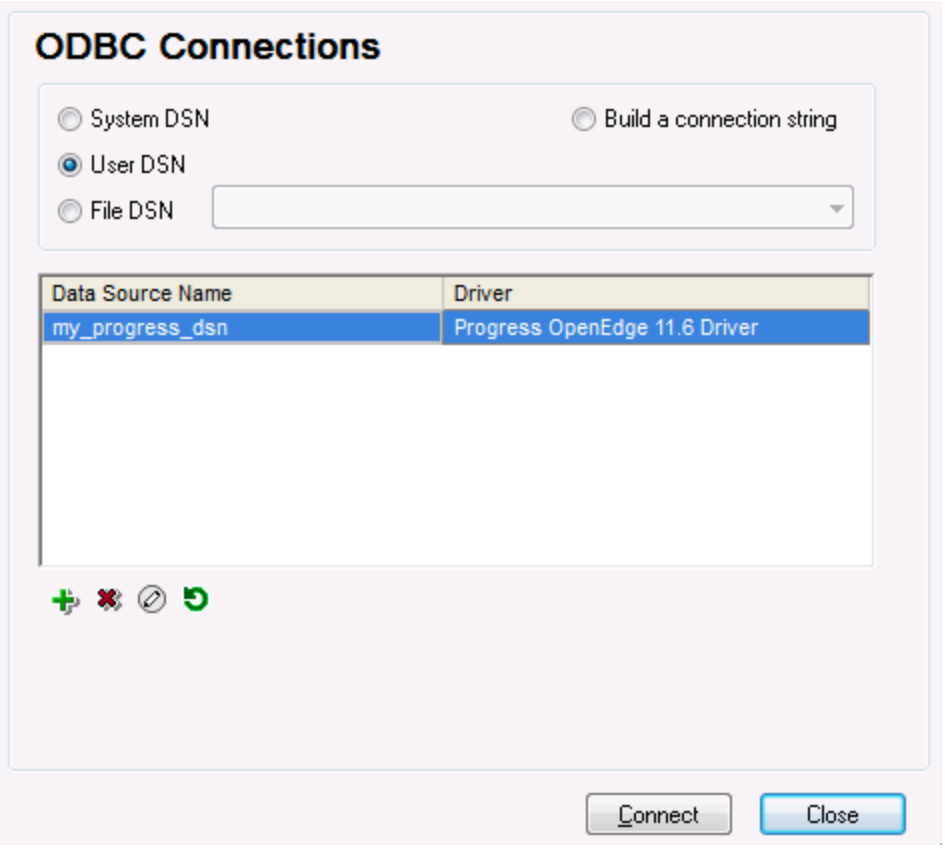


3. Füllen Sie die Anmeldeinformationen für die Datenbankverbindung aus (Datenbank, Server, Port, Benutzername und Passwort) und klicken Sie auf **OK**. Um die Verbindung zu überprüfen, bevor Sie die eingegebenen Daten speichern, klicken Sie auf **Verbindung testen**.



4. Klicken Sie auf OK. Die neue Datenquelle wird nun in der Liste der ODBC-Datenquellen angezeigt.

© 2016-2022 Altova GmbH



5. Klicken Sie auf **Verbinden**.

4.2.1.12.18 Sybase (JDBC)

Dieses Kapitel enthält eine Beispielanleitung, wie Sie über JDBC eine Verbindung zu einer Sybase-Datenbank herstellen.

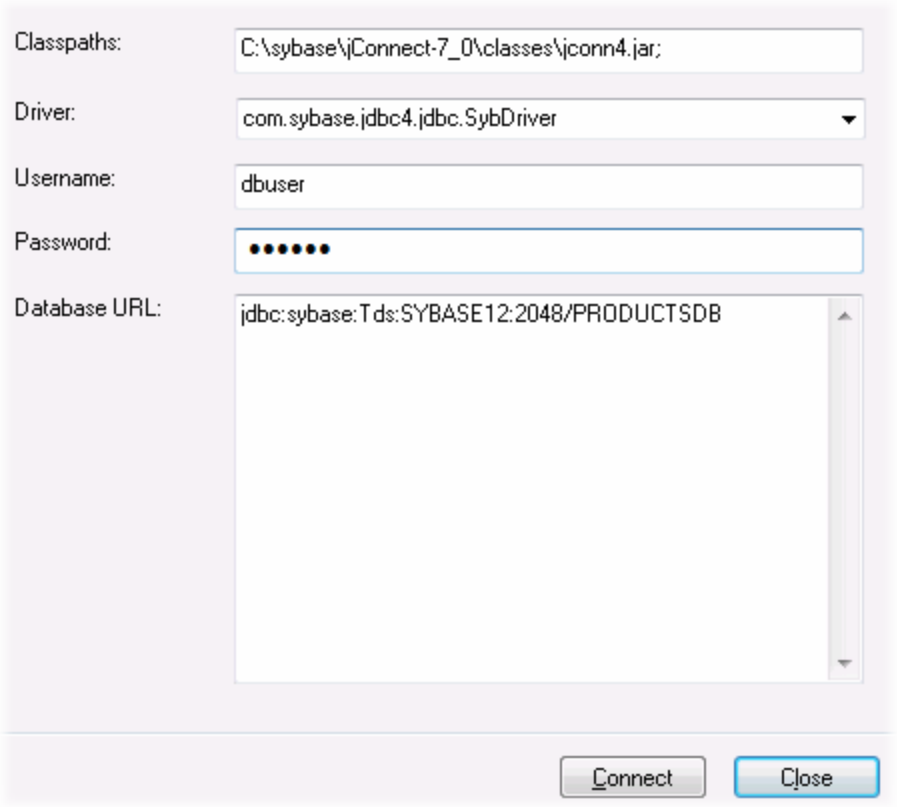
Voraussetzungen:

- Auf Ihrem Betriebssystem muss JRE (Java Runtime Enviroment) oder Java Development KIT (JDK) installiert sein. Dabei muss es sich entweder um Oracle JDK oder einen Open Source Build wie Oracle OpenJDK handeln.MapForce ermittelt den Pfad zur Java Virtual Machine (JVM) anhand der folgenden Ordner und zwar in folgender Reihenfolge: a) anhand des benutzerdefinierten JVM-Pfads, den Sie eventuell in den **Applikationsoptionen** definiert haben, siehe Java-Einstellungen[1542]; b) anhand des JVM-Pfads in der Windows Registry; c) anhand der `JAVA_HOME`-Umgebungsvariablen.
- Stellen Sie sicher, dass die Plattform von MapForce (32-Bit, 64-Bit) mit der des JRE/JDK übereinstimmt.
- Auf Ihrem Betriebssystem muss die Sybase *jConnect*-Komponente installiert sein (in diesem Beispiel wird *jConnect 7.0* verwendet. Sie wird als Teil des *Sybase Adaptive Server Enterprise PC Client* installiert). Eine Anleitung zur Installation des Datenbank Client finden Sie in der Sybase-Dokumentation.

- Sie haben die folgenden Datenbankverbindungsinformationen zur Verfügung: Host, Port, Datenbankname, Benutzername und Passwort.

So stellen Sie über JDBC eine Verbindung zu Sybase her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **JDBC-Verbindungen**.
3. Geben Sie neben "Classpaths" den Pfad zur .jar-Datei , die die Verbindung zur Datenbank bereitstellt, ein. Falls nötig, können Sie auch eine durch Semikola getrennte Liste von .jar-Dateipfaden eingeben. Die benötigte .jar-Datei in diesem Beispiel befindet sich unter dem folgenden Pfad: **C:\sybase\jConnect-7_0\classes\jconn4.jar**. Beachten Sie, dass Sie das Textfeld "Classpaths" leer lassen können, wenn Sie den/die .jar-Dateipfad(e) zur Umgebungsvariablen CLASSPATH des Betriebssystems hinzugefügt haben (siehe auch Konfigurieren des CLASSPATH[193]).
4. Wählen Sie aus der Liste der verfügbaren JDBC-Treiber den Sybase JDBC-Treiber aus (in diesem Beispiel **com.sybase.jdbc4.jdbc.SybDriver**). Beachten Sie, dass dieser Eintrag zur Verfügung steht, wenn entweder im Textfeld "Classpaths" oder in der Umgebungsvariablen CLASSPATH des Betriebssystems eine gültige .jar-Datei gefunden wird (siehe vorheriger Schritt).



5. Geben Sie den Benutzernamen und das Passwort in die entsprechenden Textfelder ein.
6. Geben Sie in das Textfeld "Datenbank-URL" den Connection String zum Datenbankserver ein, indem Sie die hervorgehobenen Werte durch die entsprechenden Werte für Ihren Datenbankserver ersetzen.

```
jdbc:sybase:Tds:hostName:port/databaseName
```

7. Klicken Sie auf **Verbinden**.

© 2016-2022 Altova GmbH

4.2.1.12.19 Teradata (JDBC)

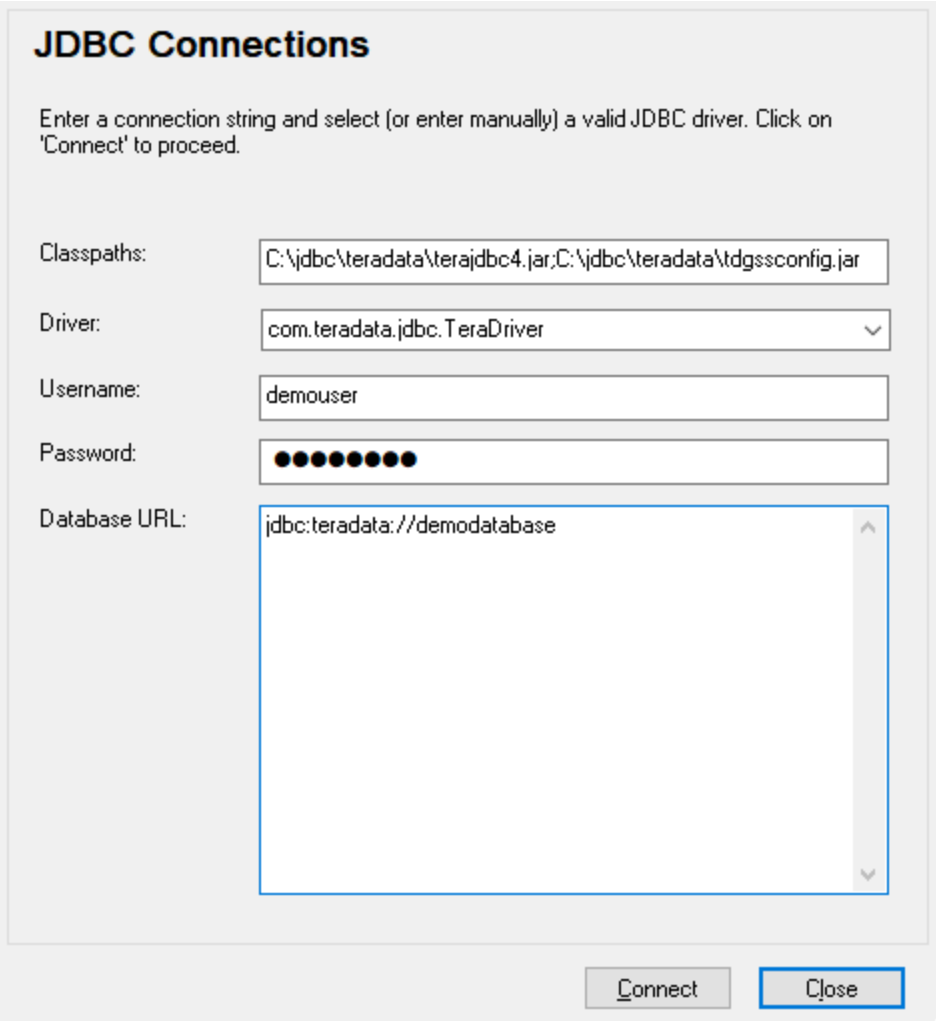
In diesem Beispiel wird gezeigt, wie Sie über JDBC eine Verbindung zu einem Teradata-Datenbankserver herstellen.

Voraussetzungen:

- Auf Ihrem Betriebssystem muss JRE (Java Runtime Enviroment) oder Java Development KIT (JDK) installiert sein. Dabei muss es sich entweder um Oracle JDK oder einen Open Source Build wie Oracle OpenJDK handeln.MapForce ermittelt den Pfad zur Java Virtual Machine (JVM) anhand der folgenden Ordner und zwar in folgender Reihenfolge: a) anhand des benutzerdefinierten JVM-Pfads, den Sie eventuell in den **Applikationsoptionen** definiert haben, siehe Java-Einstellungen[1542]; b) anhand des JVM-Pfads in der Windows Registry; c) anhand der `JAVA_HOME`-Umgebungsvariablen.
- Stellen Sie sicher, dass die Plattform von MapForce (32-Bit, 64-Bit) mit der des JRE/JDK übereinstimmt.
- Der JDBC-Treiber (eine oder mehrere .jar-Dateien, die die Verbindung zur Datenbank herstellen) muss auf Ihrem Betriebssystem installiert sein. In diesem Beispiel wird der Teradata JDBC-Treiber 16.20.00.02 verwendet. Nähere Informationen dazu finden Sie unter https://downloads.teradata.com/download/connectivity/jdbc-driver.
- Sie haben die folgenden Datenbankinformationen zur Verfügung: Host, Datenbank, Port, Benutzername und Passwort.

So stellen Sie über JDBC eine Verbindung zu Teradata her:

1. Starten Sie den Datenbank-Verbindungsassistenten[169].
2. Klicken Sie auf **JDBC-Verbindungen**.
3. Geben Sie neben "Classpaths" den Pfad zur .jar-Datei , die die Verbindung zur Datenbank bereitstellt, ein. Falls nötig, können Sie auch eine durch Semikola getrennte Liste von .jar-Dateipfaden eingeben. Die benötigte .jar-Datei in diesem Beispiel befindet sich unter dem folgenden Pfad:**C:\jdbc\teradata**. Beachten Sie, dass Sie das Textfeld "Classpaths" leer lassen können, wenn Sie den/die .jar-Dateipfad(e) zur Umgebungsvariablen CLASSPATH des Betriebssystems hinzugefügt haben (siehe auch Konfigurieren des CLASSPATH[193]).
4. Wählen Sie im Feld "Treiber" **com.teradata.jdbc.TeraDriver** aus. Beachten Sie, dass dieser Eintrag zur Verfügung steht, wenn entweder im Textfeld "Classpath" oder in der Umgebungsvariablen `CLASSPATH` des Betriebssystems eine gültige .jar-Datei gefunden wird (siehe vorheriger Schritt).



5. Geben Sie den Benutzernamen und das Passwort in die entsprechenden Textfelder ein.
6. Geben Sie in das Textfeld "Datenbank-URL" den Connection String zum Datenbankserver ein, indem Sie die hervorgehobenen Werte durch die entsprechenden Werte für Ihren Datenbankserver ersetzen.

```
jdbc:teradata://databaseServerName
```

7. Klicken Sie auf **Verbinden**.

4.2.1.12.20 Teradata (ODBC)

In diesem Beispiel wird gezeigt, wie Sie über ODBC eine Verbindung zu einem Teradata-Datenbankserver herstellen.

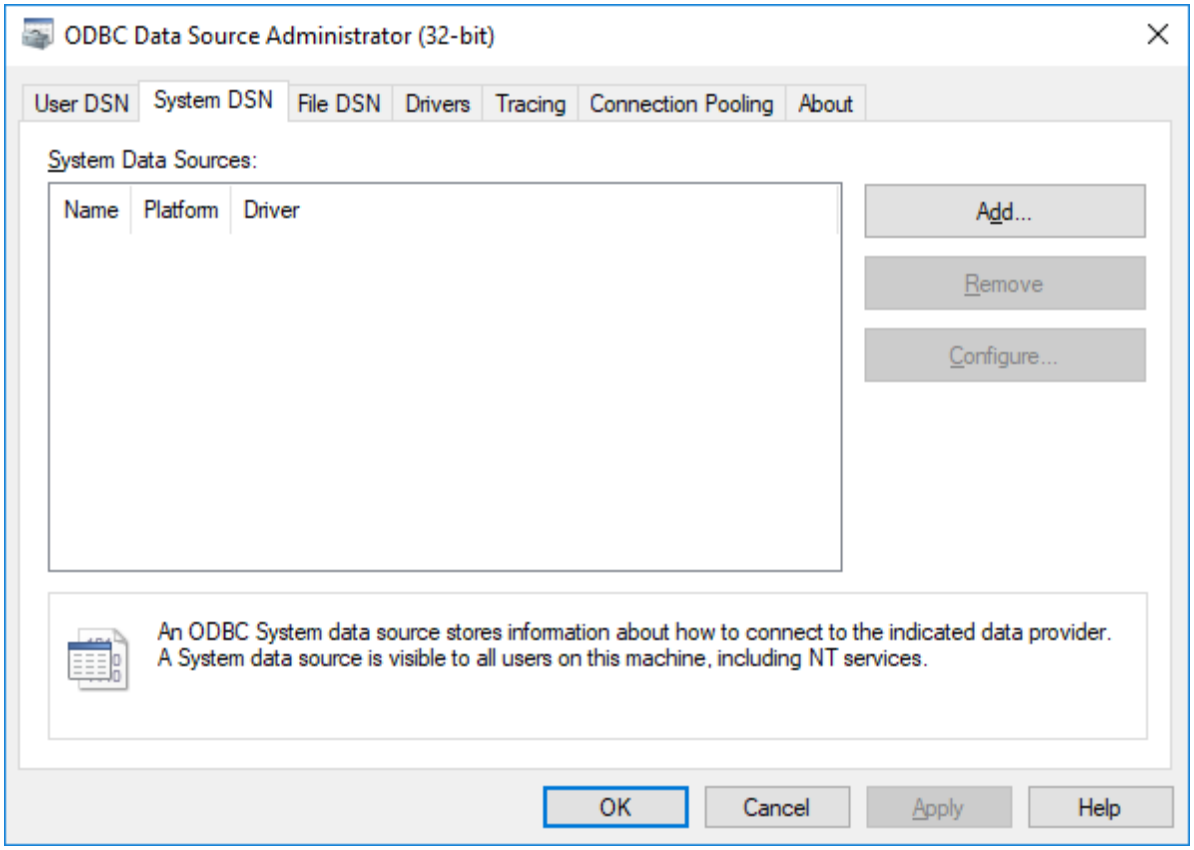
© 2016-2022 Altova GmbH

Voraussetzungen:

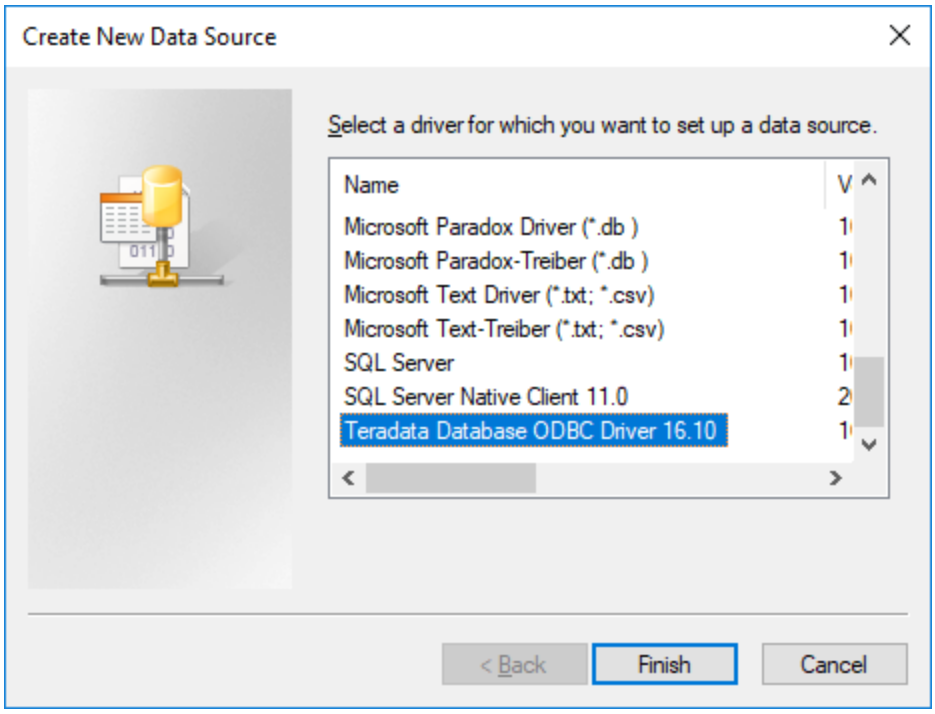
- Der Teradata ODBC-Treiber (https://downloads.teradata.com/download/connectivity/odbc-driver/windows) muss installiert sein. In diesem Beispiel wird der Teradata ODBC-Treiber für Windows Version 16.20.00 verwendet.
- Sie haben die folgenden Datenbankinformationen zur Verfügung: Host, Benutzername und Passwort.

So stellen Sie über ODBC eine Verbindung zu Teradata her:

1. Drücken Sie die **Windows**-Taste, beginnen Sie mit der Eingabe von "ODBC" und wählen Sie aus der Liste der Vorschläge **ODBC Datenquellen einrichten (32-Bit)** . Wenn Sie einen 64-Bit-ODBC-Treiber haben, wählen Sie **ODBC Datenquellen einrichten (64-Bit)** und verwenden Sie in den nachfolgenden Schritten die 64-Bit-Version von MapForce.

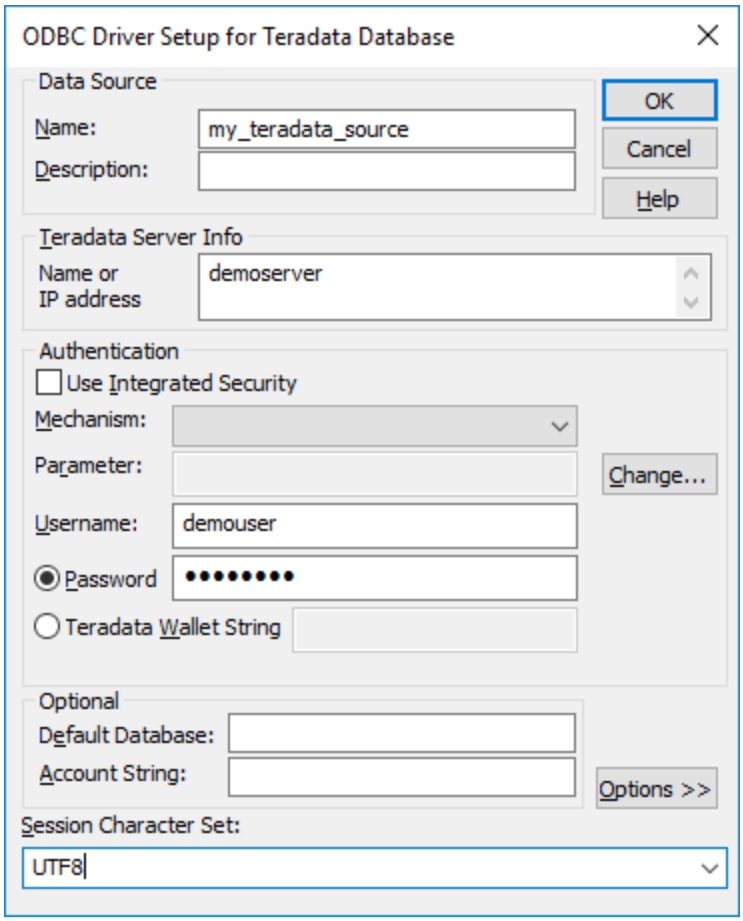


2. Klicken Sie auf das Register **System DSN** und anschließend auf **Hinzufügen**.

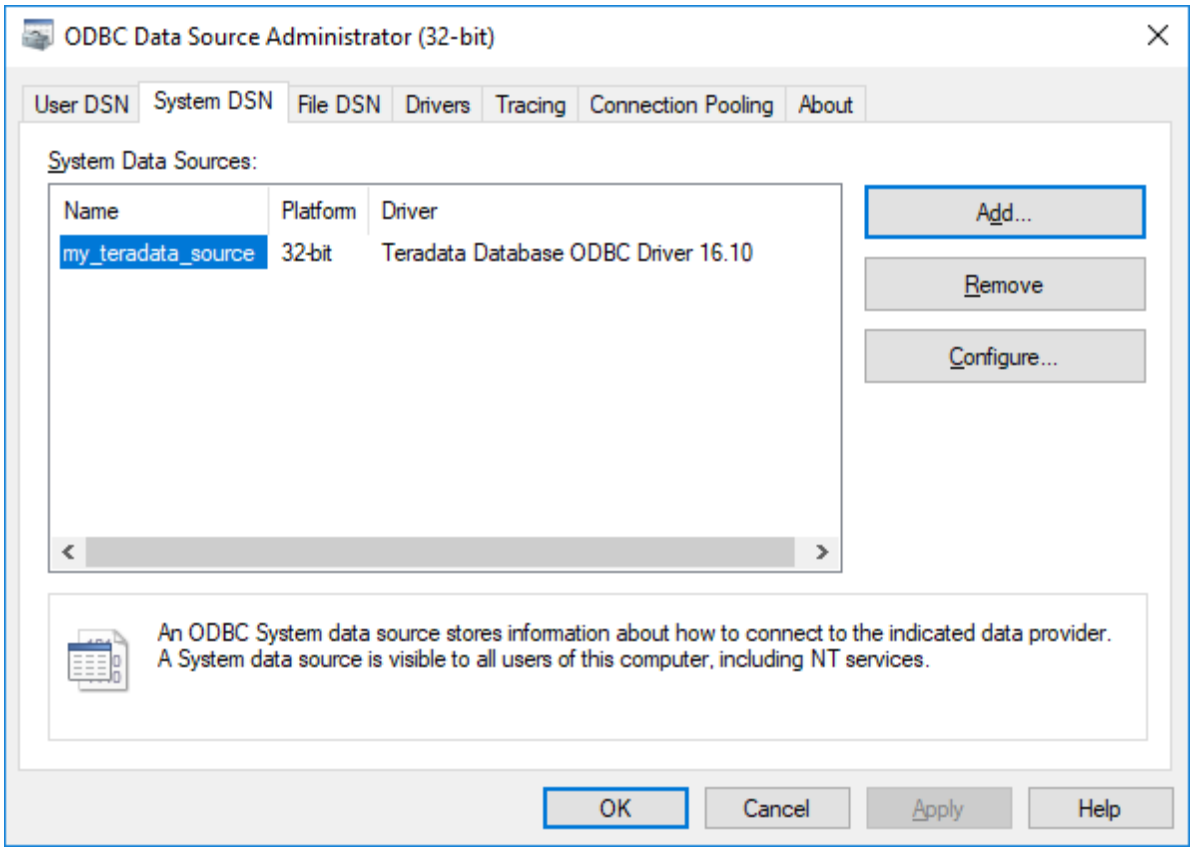


3. Wählen Sie **Teradata Database ODBC Driver** aus und klicken Sie auf **Fertig stellen**.

© 2016-2022 Altova GmbH



4. Geben Sie einen Namen und optional eine Beschreibung für diese ODBC-Datenquelle ein. Geben Sie außerdem die Anmeldeinformationen für die Datenbankverbindung ein (Datenbankserver, Benutzer, Passwort) und wählen Sie optional eine Datenbank aus.
5. Klicken Sie auf **OK**. Die Datenquelle wird nun in der Liste angezeigt.



6. Starten Sie MapForce und anschließend den Datenbankverbindungsassistenten[169].
7. Klicken Sie auf **ODBC-Verbindungen**.

© 2016-2022 Altova GmbH



8. Klicken Sie auf **System DSN,** wählen Sie die zuvor erstellte Datenquelle aus und klicken Sie auf **Verbinden**.

Anmerkung: Wenn Sie die folgende Fehlermeldung erhalten: "Der Treiber hat eine ungültige SQL_DRIVER_ODBC_VER: 03.80 zurückgegeben (oder ...konnte nicht zurückgegeben werden)", überprüfen Sie, ob der Pfad zum ODBC-Client (z.B. **C:\Programme\Teradata\Client\16.10\bin**, falls unter diesem Pfad installiert) in der PATH-Umgebungsvariablen Ihres Systems vorhanden ist. Fügen Sie den Pfad manuell hinzu, falls der Pfad fehlt.

4.2.2 Einführung in Datenbank-Mappings

Dieser Abschnitt enthält eine Einführung über das Arbeiten mit Datenbanken in MapForce. Es wird erläutert, wie Sie nach Herstellung einer Verbindung zur Datenbank (siehe Herstellen einer Verbindung zu einer Datenbank 167) damit arbeiten können. Es wird beschrieben, wie Sie Datenbankobjekte, die im Mapping aufscheinen sollen, auswählen, wie Sie Datenbankbeziehungen behandeln und die Datenbankeinstellungen für das Mapping konfigurieren. Des Weiteren finden Sie hier Beispiele, wie Sie durch das Mappen von Daten von oder auf Datenbankkomponenten bestimmte Ergebnisse erzielen.

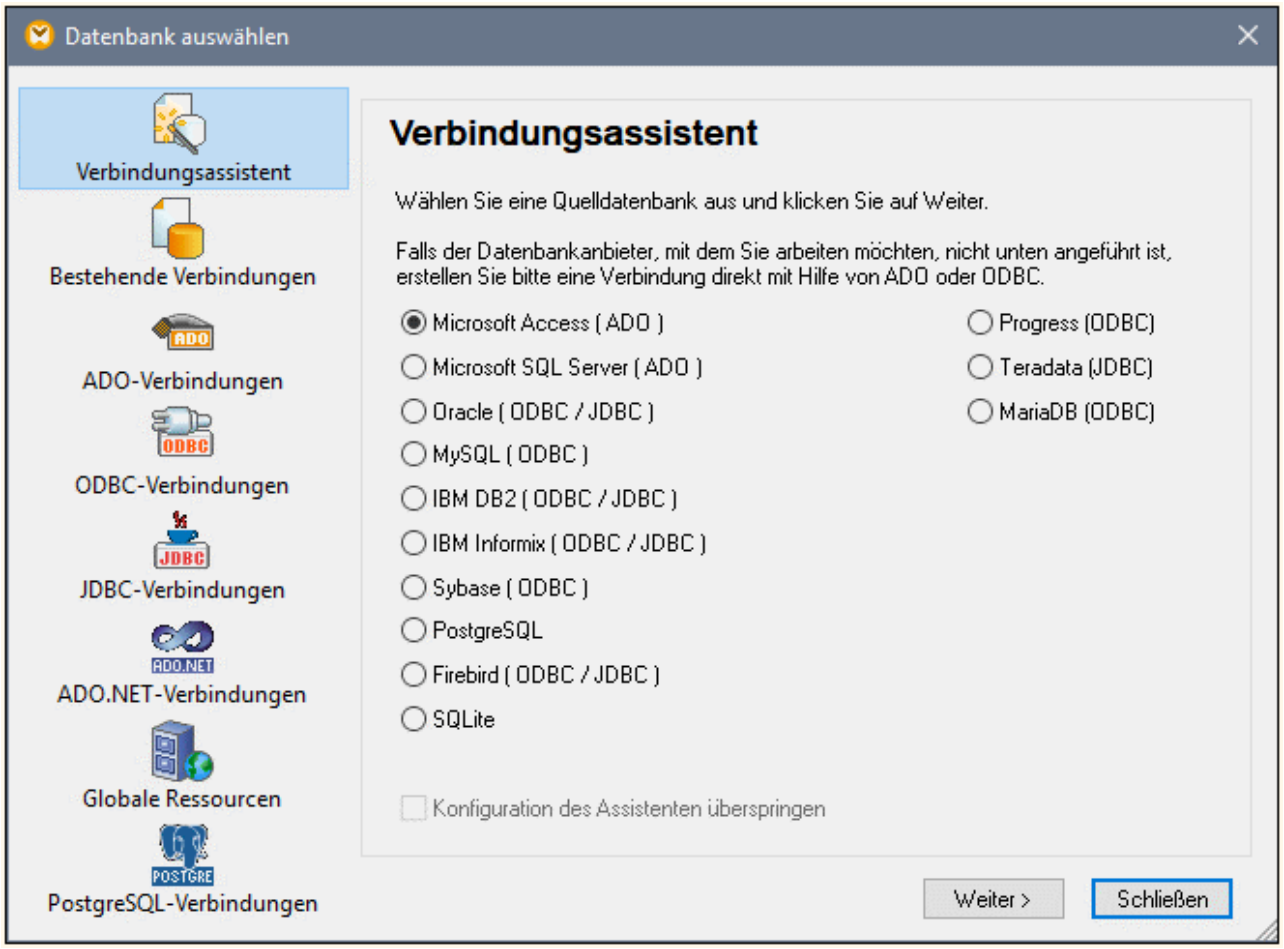
4.2.2.1 Hinzufügen von Datenbanken zum Mapping

Bevor Sie eine Datenbank zum Mapping hinzufügen, stellen Sie sicher, dass eine Transformationssprache ausgewählt ist, von der Datenbank-Mappings unterstützt werden, wie etwa die Transformationssprache BUILT-IN oder eine der folgenden Sprachen: C++, C#, Java (siehe auch Auswählen einer Transformationssprache 28). Wenn Sie vorhaben, das Mapping auf FlowForce Server bereitzustellen, es mit MapForce Server auszuführen oder Funktionen wie die Bulk-Einfügung zu verwenden, muss als Transformationssprache BUILT-IN ausgewählt sein.

Sobald die gewünschte Transformationssprache ausgewählt ist, können Sie auf eine der folgenden Arten eine Datenbank zum Mapping hinzufügen:

- Klicken Sie im Menü **Einfügen** auf **Datenbank**.
- Klicken Sie auf die Symbolleistenschaltfläche **Datenbank einfügen** ( ).

Bei Auswahl eines dieser Befehle wird ein Datenbankassistent angezeigt, der Sie Schritt für Schritt durch den Verbindungsvorgang leitet.



Datenbankverbindungsassistent

 © 2016-2022 Altova GmbH

Anmerkung: In einigen komplexen Szenarien können Datenbanken auch als Variablen zum Mapping hinzugefügt werden (siehe Verwendung von Variablen 882). Wenn Sie eine Datenbankstruktur in Form einer Variablen zum Mapping hinzufügen, wird derselbe oben erwähnte Datenbankverbindungsassistent aufgerufen.

Eine Anleitung zum Herstellen einer Verbindung zu einer der von MapForce unterstützten Datenbanken finden Sie im Abschnitt Herstellen einer Verbindung zu einer Datenbank 167.

Nach erfolgreicher Herstellung der Datenbankverbindung werden Sie aufgefordert, die Datenbankobjekte, die im Mapping angezeigt werden sollen, auszuwählen. Nähere Informationen dazu finden Sie unter Hinzufügen, Bearbeiten und Entfernen von Datenbankobjekten 260.

4.2.2.2 Beispiel: Hinzufügen der Datenbank "altova.mdb" zum Mapping

In diesem Beispiel wird gezeigt, wie Sie eine Microsoft Access-Datenbank zu einem Mapping hinzufügen. Die Beispieldatenbank hat den Namen **altova.mdb** und befindet sich im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Die Datenbank **altova.mdb** unterstützt verschiedene in dieser Dokumentation beschriebene Aktionen und Konzepte im Zusammenhang mit Datenbankaktionen.

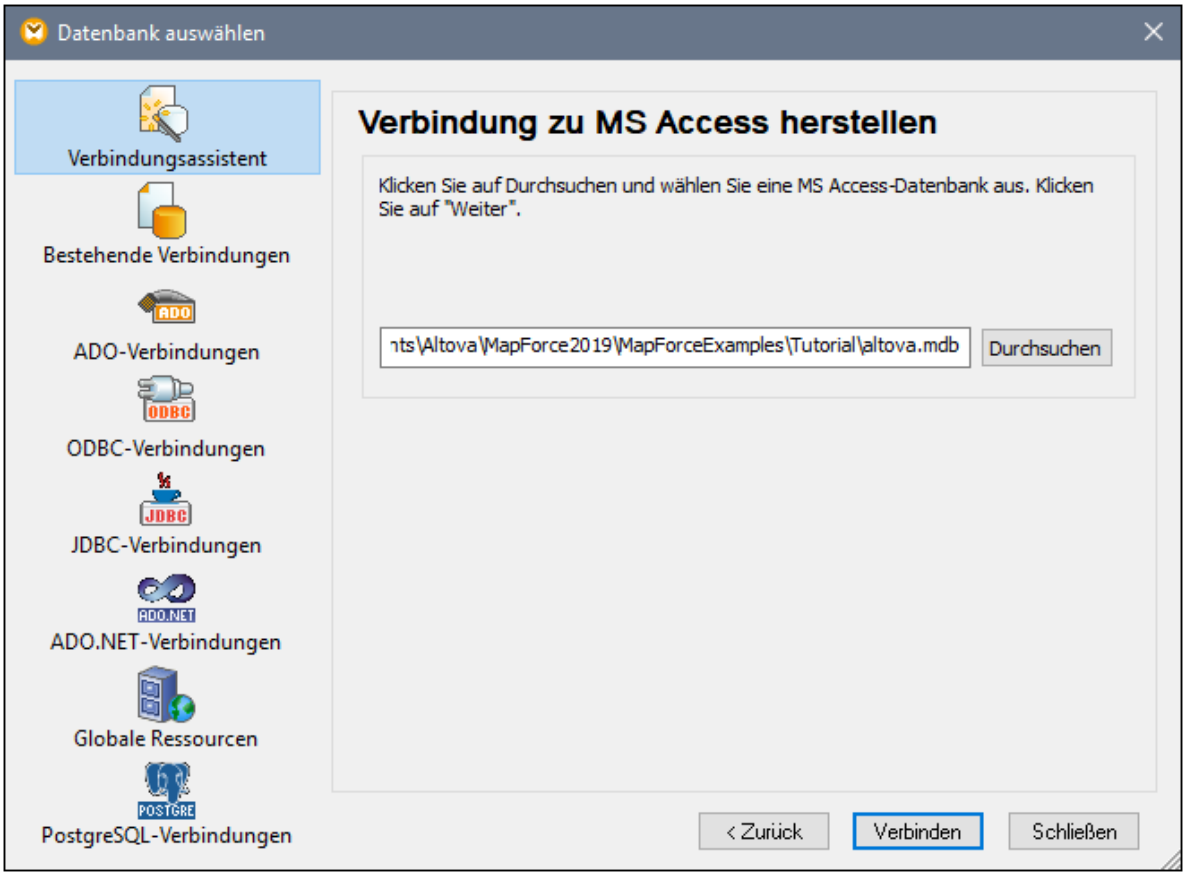
So fügen Sie die Datenbank altova.mdb zum Mapping hinzu:

1. Klicken Sie im Menü **Einfügen** auf **Datenbank**. Alternativ dazu können Sie auch in der Symbolleiste auf die Schaltfläche **Datenbank einfügen** ( ) klicken.

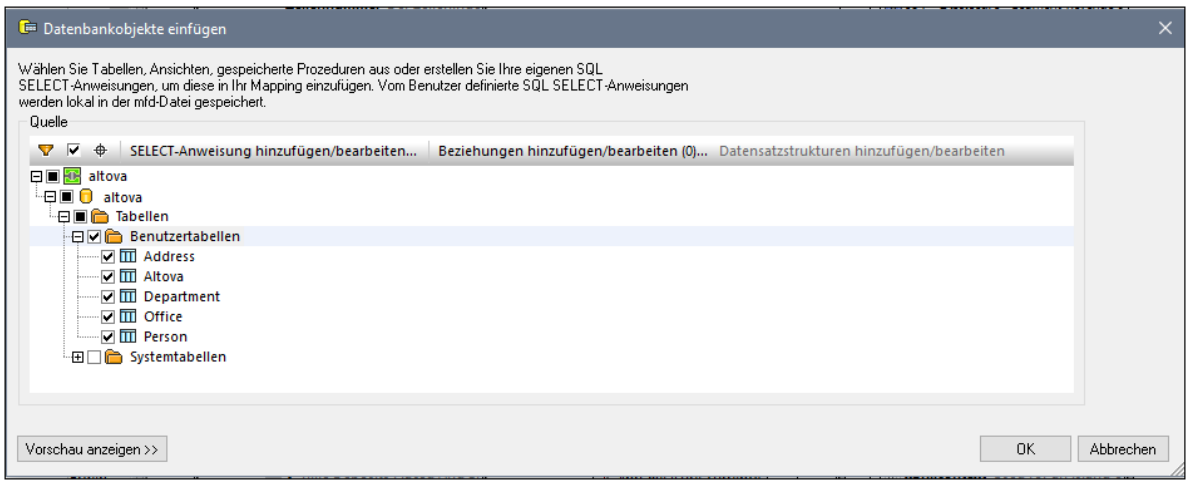


2. Klicken Sie auf **Verbindungsassistent**, wählen Sie **Microsoft Access (ADO)** und klicken Sie auf **Weiter**.

© 2016-2022 Altova GmbH



3. Navigieren Sie im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** zur Datei **altova.mdb** und klicken Sie anschließend auf **Verbinden**.
4. Wenn Sie aufgefordert werden, die Datenbankobjekte auszuwählen, wählen Sie **Benutzertabellen** aus.

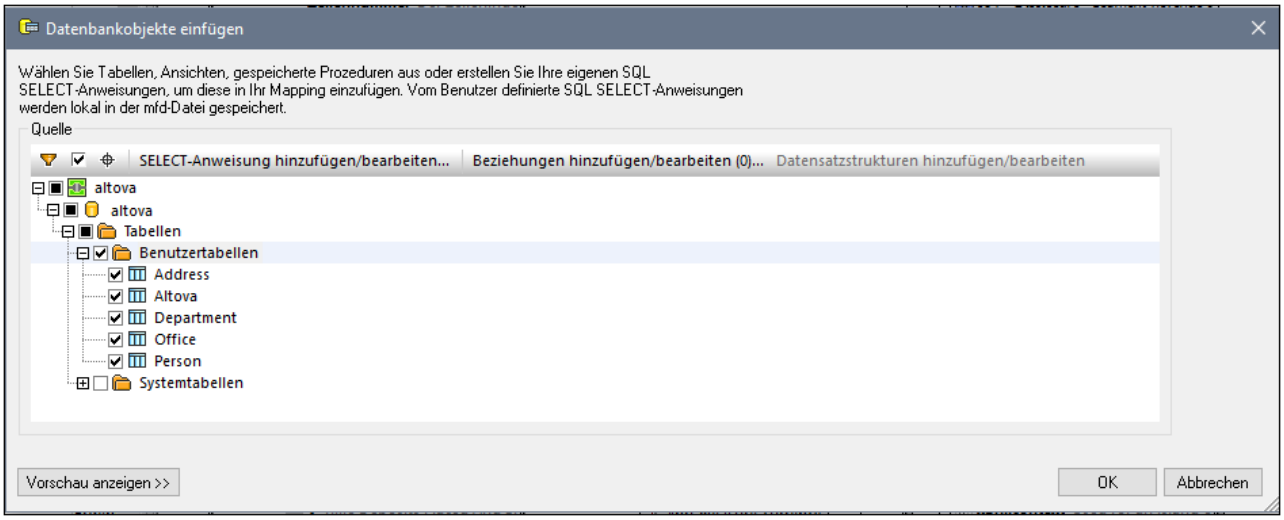


4.2.2.3 Hinzufügen, Bearbeiten und Entfernen von Datenbankobjekten

Einige Datenbanken können sehr viele Datenbankobjekte (wie z.B. Schemas, Tabellen, Ansichten, usw.) haben. In diesem Kapitel wird erläutert, wie Sie nur diejenigen Objekte ins Mapping inkludieren, die Sie dafür benötigen. Im Beispiel unten wird eine Access-Beispieldatenbank verwendet; die Vorgehensweise ist bei anderen Datenbankarten ähnlich.

1. Klicken Sie im Menü **Einfügen** auf **Datenbank**.
2. Klicken Sie auf **Verbindungsassistent** und anschließend auf **Microsoft Access (ADO)**.
3. Klicken Sie auf **Weiter** und navigieren Sie zu **altova.mdb** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**.

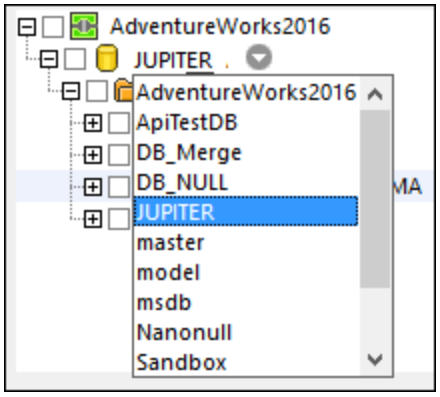
Daraufhin erscheint ein Dialogfeld, in dem Sie die Datenbankobjekte, die im Mapping enthalten sein sollen, auswählen können.



Im obersten Node in der Struktur wird die Datenbankverbindung angezeigt. Die Struktur darunter variiert je nach Datenbankart. So weisen etwa Oracle- und IBM DB2-Datenbanken unter dem Verbindungs-Node einen Schema-Node auf, während andere Datenbankarten einen Katalog (Datenbank) Node haben. Ein fett angezeigter Node kennzeichnet den Standardkatalog (Datenbank) bzw. das Standardschema.

Wenn Ihr Datenbank-Benutzerkonto Zugriff auf mehrere Datenbanken oder Schemas auf dem Server hat, können Sie durch Klick auf das Symbol zur gewünschten DB bzw. zum gewünschten Schema wechseln. In der Abbildung unten sehen Sie ein Beispiel für SQL Server. Die Struktur unterscheidet sich bei anderen Datenbankarten eventuell davon.

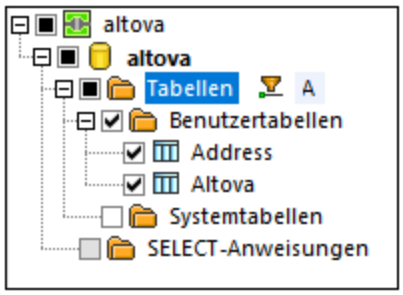
© 2016-2022 Altova GmbH



Um ein Datenbankobjekt (z.B. eine Tabelle) im Mapping zu inkludieren, aktivieren Sie das Kontrollkästchen neben dem Objekt. Aktivieren Sie in diesem Beispiel das Kontrollkästchen neben **Benutzertabellen**.

Mit Hilfe der Schaltfläche **Objektsuche**  können Sie bestimmte Datenbankelemente suchen. Wählen Sie in der Auswahlliste, die im unteren Bereich des Dialogfelds erscheint, ein bestimmten Objekt aus (oder geben Sie dessen Namen ein).

Mit Hilfe der **Filter**-Schaltfläche  können Sie die Objekte nach Namen filtern. Wenn Sie auf die Filterschaltfläche klicken, erscheint neben Objekten, die eine Filterung unterstützen (in diesem Beispiel "Tabellen"), ein Filtersymbol. Klicken Sie auf das Filtersymbol, um auszuwählen, ob der Objektname mit dem Suchtext beginnen oder enden soll, identisch mit dem Suchtext sein soll oder den Suchtext enthalten soll. Nun können Sie den Suchtext neben dem Filter eingeben (in diesem Beispiel "A"):



Bei Aktivierung der Schaltfläche **Nur ausgewählte Objekte anzeigen**  werden nur diejenigen Objekte angezeigt, deren Kontrollkästchen aktiviert ist.

Über die Schaltfläche **SELECT-Anweisung hinzufügen/bearbeiten** können Sie benutzerdefinierte SELECT-Anweisungen für die aktuelle Datenbank hinzufügen oder bearbeiten. Die durch solche Anweisungen zurückgegebenen Daten stehen anschließend als Mapping-Quelle zur Verfügung. Nähere Informationen finden Sie unter SELECT-Anweisungen als virtuelle Tabellen 368.

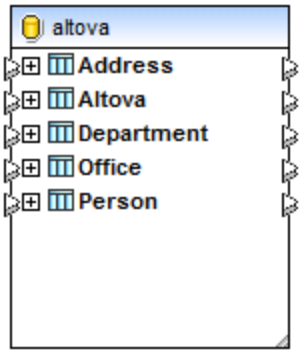
Über die Schaltfläche **Beziehungen hinzufügen/bearbeiten** können Sie zusätzlich zu den eventuell bereits vorhandenen Beziehungen lokale Primär- und Sekundärschlüsselbeziehungen zwischen Feldern in der Datenbank definieren. Nähere Informationen dazu finden Sie unter Definieren von lokalen Beziehungen 276.

Die Schaltfläche **Datensatzstrukturen hinzufügen/bearbeiten** gilt für Datenbanken, die gespeicherte Prozeduren unterstützen. Sie ist nur dann aktiv, wenn in der Datenbankstruktur gerade eine gespeicherte Prozedur aktiviert ist. Nähere Informationen dazu finden Sie unter Gespeicherte Prozeduren 402.

Mit Hilfe der Schaltfläche **Vorschau anzeigen** können Sie sofort eine Vorschau auf die Daten der aktuell ausgewählten Tabelle oder Ansicht anzeigen. Beachten Sie, dass Sie über den Datenbank Browser eine Datenbank auch unabhängig vom Mapping durchsuchen und abfragen können. Nähere Informationen dazu finden Sie unter Anzeigen und Abfragen von Datenbanken 388.

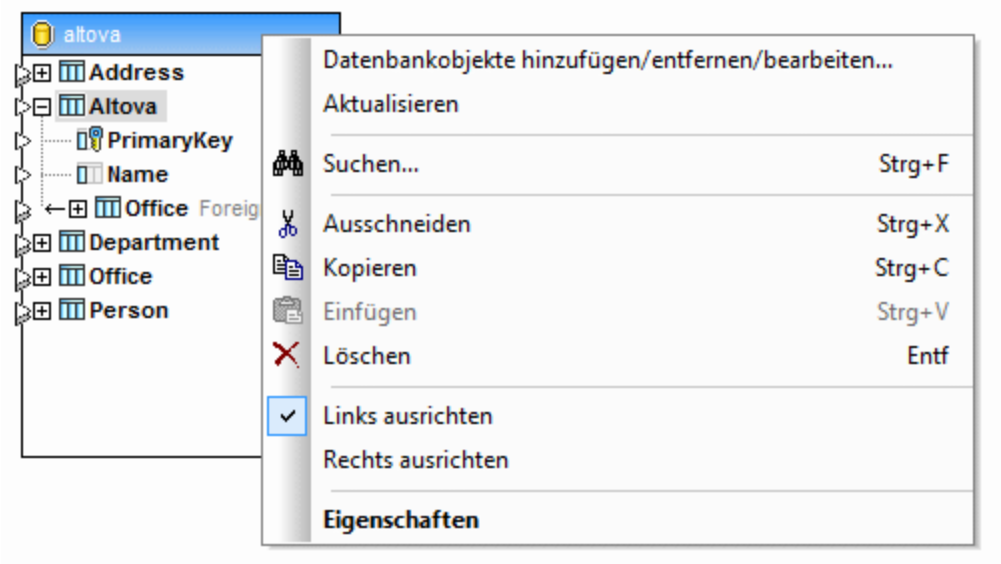
Das Kontrollkästchen **Schema aus Tabellennamen entfernen** ist nur bei Datenbanken, die die Qualifizierung von Objekten mit einem relativen Namen unterstützen, aktiv. Dies ist nützlich, wenn Sie beabsichtigen, später zu einer anderen Datenbank zu wechseln. Nähere Informationen dazu finden Sie unter Wechseln von Datenbanken und Schemas 263.

Sobald Sie bereit sind, die Datenbankobjekte zum Mapping hinzuzufügen, klicken Sie auf **OK**. In der Datenbankkomponente werden nur die ausgewählten Tabellen, Ansichten, usw. angezeigt. Sie können nun auf die übliche Art Mapping-Verbindungen von und zu diesen Objekten ziehen.



Sie können die ausgewählten Datenbankobjekte jederzeit ändern. Klicken Sie dazu mit der rechten Maustaste auf die Komponente und wählen Sie **Datenbankobjekte hinzufügen/entfernen/bearbeiten**.

© 2016-2022 Altova GmbH



4.2.2.4 Wechseln von Datenbanken und Schemas

Um eine Datenbankkomponente im Mapping zu ändern, sodass darin eine neue Datenbankverbindung verwendet wird, öffnen Sie die Datenbank-Komponenteneinstellungen[286], klicken Sie auf **Ändern** und befolgen Sie die Anweisungen des Assistenten, um eine Verbindung zu einer neuen Datenbank herzustellen. Wenn Sie nur die im Mapping angezeigten Datenbankobjekte wechseln möchten, klicken Sie mit der rechten Maustaste auf die Titelleiste der Datenbankkomponente und wählen Sie im Kontextmenü den Befehl **Datenbankobjekte hinzufügen/bearbeiten/entfernen,** siehe auch Hinzufügen, Bearbeiten und Entfernen von Datenbankobjekten[260].

Beachten, Sie, dass die bestehenden Mapping-Verbindungen zu oder von einer Datenbankkomponente bei einem Wechsel der Datenbank nicht alle neu gemappt werden. Bei den meisten Datenbankarten können die Mapping-Verbindungen nach dem Wechsel einer Datenbank jedoch beibehalten werden, wenn Sie einige Vorkehrungen treffen, wie in diesem Kapitel beschrieben.

Wenn Sie Datenbankobjekte, wie unter Hinzufügen, Bearbeiten und Entfernen von Datenbankobjekten[260] beschrieben, auswählen, können Sie optional auswählen, ob die Datenbankobjektnamen als relativ zu einem Standardschema behandelt werden sollen. Im Zusammenhang mit einer Datenbank ist ein "Schema" eine logische Gruppierung von Datenbankobjekten - normalerweise zu Sicherheitszwecken wie der Zuweisung von Berechtigungen. Schemas verhalten sich je nach Datenbankart etwas unterschiedlich.

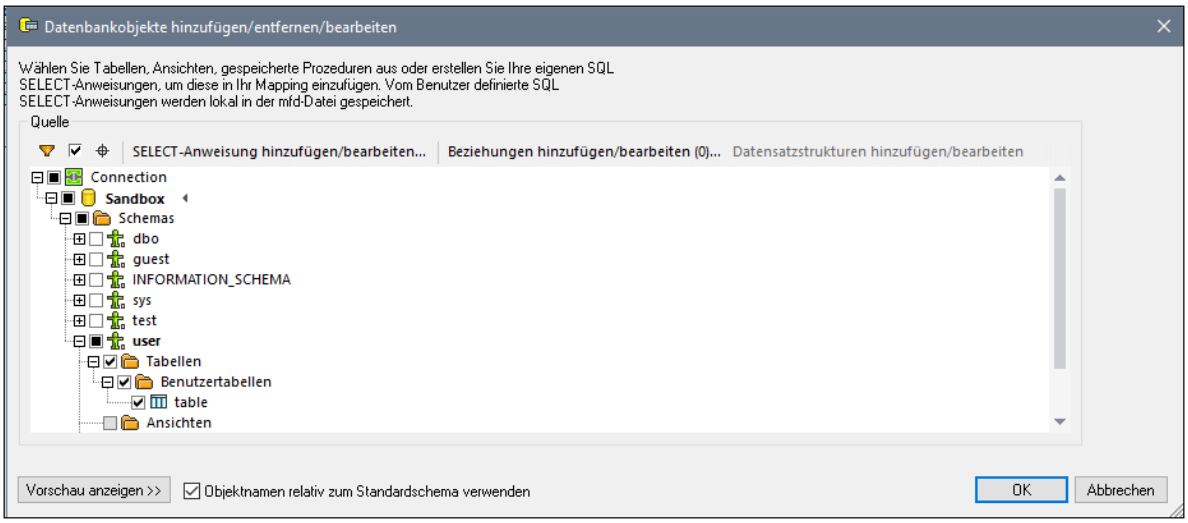
Datenbankobjektnamen in MapForce als relativ zu einem Schema zu behandeln, ist wichtig, wenn Sie beabsichtigen, im Mapping zu einem späteren Zeitpunkt zu einer anderen Datenbank (z.B. zu einer Produktionsdatenbank) zu wechseln. Dies ist auch nützlich, wenn das Datenbankschema auf dem Server irgendwann umbenannt wurde und Sie daher das Mapping entsprechend aktualisieren müssen. Wenn das neue Schema dieselbe Struktur wie das zum Zeitpunkt des Mapping-Designs verwendete hat, können Sie zu diesem Schema wechseln, ohne alle Mapping-Verbindungen manuell ändern zu müssen. Dies ist nur dann möglich, wenn Sie die Objeknamen, wie unten beschrieben, als relativ zu einem Standardschema konfigurieren.

Beachten Sie die folgenden Punkte:

- Die Verwendung von Objektnamen relativ zu einem Standardschema ist bei den Datenbanktypen möglich, die Schemas unterstützen: IBM DB2, IBM Informix, IBM Db2 für i (iSeries), Oracle, PostgreSQL, Progress OpenEdge, SQL Server und Sybase.
- Diese Funktionalität wird ab MapForce Version 2020 unterstützt.
- Wenn die Datenbankkomponente lokale Beziehungen[276] oder SELECT-Anweisungen als virtuelle Tabellen[368] enthält, können relative Namen nicht verwendet werden.

So behandeln Sie Datenbankobjektnamen als relativ zum Standardschema:

1. Klicken Sie mit der rechten Maustaste auf eine Datenbankkomponente im Mapping und wählen Sie im Kontextmenü den Befehl **Datenbankobjekte hinzufügen/entfernen/bearbeiten**.
2. Wählen Sie ein oder mehrere Objekte, die je nach Anwendung *zum Standardschema* oder zum *Standardkatalog (Datenbank) und Schema gehören,* aus. Die Standarddatenbank und das Standardschema werden fett angezeigt. Im Beispiel unten ist der Standardkatalog **Sandbox** und das Standardschema **user**. Diese Struktur ist SQL Server-Datenbank-spezifisch und kann bei anderen Datenbankarten anders aussehen. Nähere Informationen zur angezeigten Struktur finden Sie unter Hinzufügen, Bearbeiten und Entfernen von Datenbankobjekten[260].



3. Aktivieren Sie das Kontrollkästchen **Objektnamen relativ zum Standardschema verwenden**. Beachten Sie, dass dieses Kontrollkästchen ausgegraut ist, wenn die Datenbank relative Objektnamen nicht unterstützt.

Wenn sich die im Mapping benötigten Objekte in einem anderen Schema (und nicht dem Standardschema) befinden, haben Sie die folgenden Möglichkeiten:

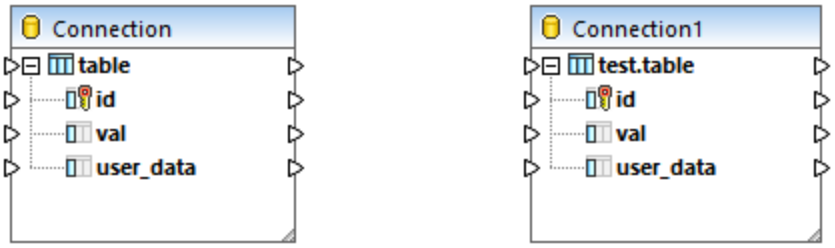
- Verbinden Sie sich als ein anderer Benutzer, der Zugriff auf das benötigte Standardschema hat, mit der Datenbank.
- Konfigurieren Sie den Datenbankserver um, um das Standardschema des bestehenden Datenbankbenutzers zu ändern, vorausgesetzt, Sie haben die nötigen Berechtigungen dazu.

Ein Datenbank-Administrator könnte das Standardschema eines Datenbankbenutzers z.B. bei einer SQL Server-Datenbank mit einer Anweisung wie der unten gezeigten ändern (dabei wird angenommen, dass der aktuelle Katalog "Sandbox" ist und dass sowohl der Benutzer als auch das Schema bereits vorhanden sind).

© 2016-2022 Altova GmbH

```
USE [Sandbox]
GO
ALTER USER [test_user] WITH DEFAULT_SCHEMA=[test_schema]
GO
```

Datenbankobjekte, die relativ zum Standardschema sind, werden im Mapping mit einem relativen Namen angezeigt. Beachten Sie die Unterschiede zwischen den folgenden beiden Datenbankkomponenten:



In der linken Komponente wurde das Kontrollkästchen **Objektnamen relativ zum Standardschema verwenden** aktiviert. Daher wird die Tabelle mit einem relativen Namen angezeigt. In der rechten Komponente wurde das Kontrollkästchen nicht aktiviert, daher enthält der Tabellenname den Schemanamen.

Wenn Datenbankobjekte relativ referenziert werden, können Sie im Mapping folgendermaßen zu einer neuen Datenbank oder einem neuen Schema wechseln, *ohne dass dabei die Mapping-Verbindungen verloren gehen*:

- Öffnen Sie die Datenbank-Komponenteneinstellungen[286] und klicken Sie auf **Ändern**. Befolgen Sie die Anweisungen des Assistenten, um *als neuer Benutzer* eine Verbindung zur neuen Datenbank herzustellen. Wenn Sie die Objekte mit einem relativen Namen qualifiziert haben und die neue Datenbank dieselbe Struktur wie die alte hat, werden alle Verbindungen im Mapping automatisch aktualisiert. Sie entsprechen nun dem *Standardkatalog und Schema des neuen Datenbankbenutzers*.
- Wenn Sie den Wechsel immer wieder durchführen müssen, wird empfohlen, die Datenbankverbindung als globale Ressource[1462] zu definieren. Die globale Ressource könnte zu.B. zwei Konfigurationen haben: eine Standardkonfiguration für die Entwicklungsdatenbank und eine Produktionskonfiguration. Durch Auswahl der gewünschten Konfiguration aus einer Dropdown-Liste können Sie zwischen den Datenbanken wechseln, vorausgesetzt beide Datenbanken haben dieselbe Struktur:

  Nähere Informationen zu diesem Szenario finden Sie unter Beispiel: Wechseln von Datenbanken[1473].

Wenn Datenbankobjekte nach dem Wechsel der Datenbank rot angezeigt werden, bedeutet dies, dass diese im neuen Datenbankschema nicht vorhanden sind.

Codegenerierung

Das Kontrollkästchen **Objektnamen relativ zum Standardschema verwenden** wirkt sich auf den generierten C#, C++ oder Java-Programmcode aus. Wenn dieses Kontrollkästchen aktiviert ist, werden auch im generierten Code alle Datenbankreferenzen relativ. Im Folgenden sehen Sie ein Beispiel für eine generierte SQL-Anweisung (in C#), wenn das Kontrollkästchen nicht aktiviert wurde. Beachten Sie, dass der Tabellenname den Schemanamen enthält.

```
var3_NewStatement = (Altova.Db.Dbs.NewStatement(closure.var1_Connection_Instance, 1,
"SELECT [id], [val], [user_data] FROM [user].[table]"));
```

Im Folgenden sehen Sie ein Beispiel für eine generierte SQL-Anweisung (in C#), wenn das Kontrollkästchen aktiviert wurde:

```
var3_NewStatement = (Altova.Db.Dbs.NewStatement(closure.var1_Connection_Instance, 1,
"SELECT [id], [val], [user_data] FROM [table]"));
```

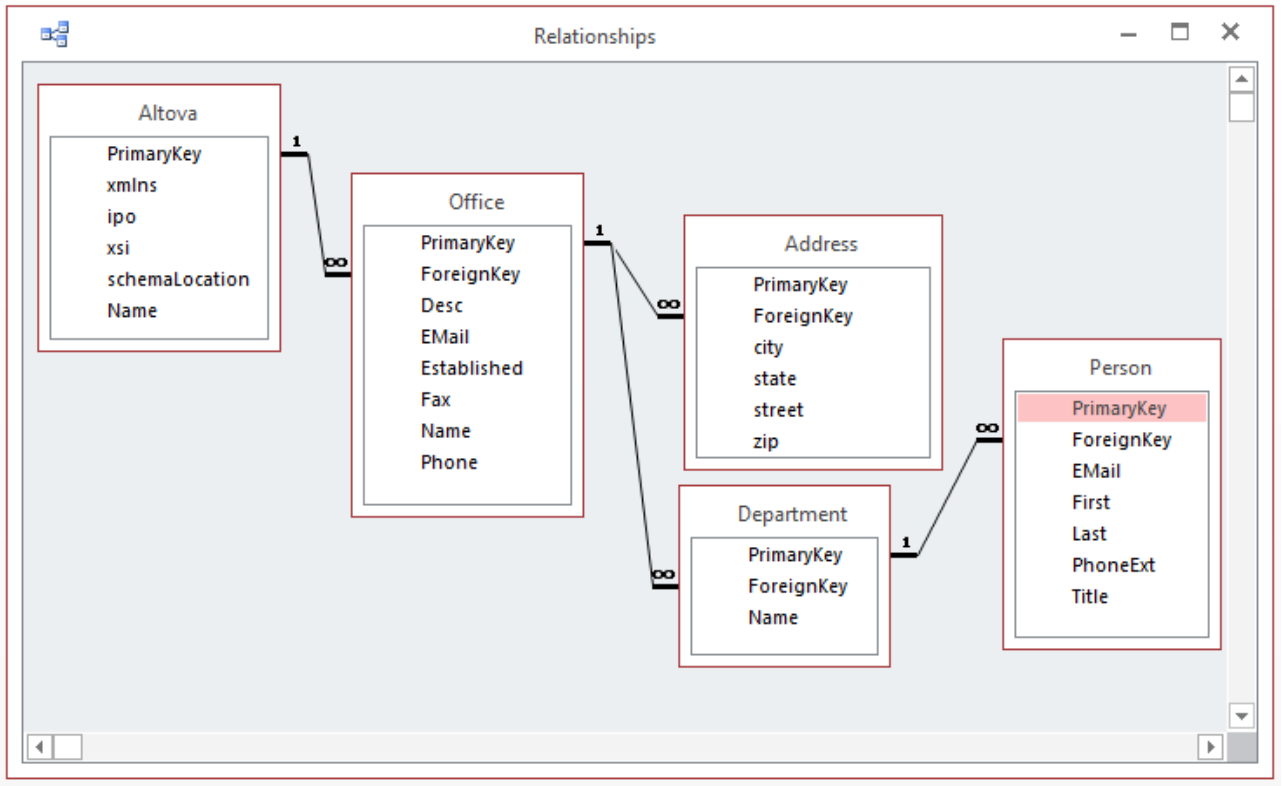
Anmerkung: Es gibt noch eine ältere Methode, um Schemanamen im generierten Code zu entfernen. Aktivieren Sie dazu in den Datenbank-Komponenteneinstellungen[286] das Kontrollkästchen **Schemanamen aus Tabellennamen entfernen**. Diese Methode wird nur aus Gründen der Rückwärtskompatibilität unterstützt und sollte vermieden werden.

4.2.2.5 Behandlung von Datenbankbeziehungen

Bei relationalen Datenbanken sind, Beziehungen zwischen deren Tabellen definiert. Wenn Sie z.B. einen Blick auf die Datenbank **altova.mdb** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** werfen, so gibt es darin eine Reihe von Beziehungen:

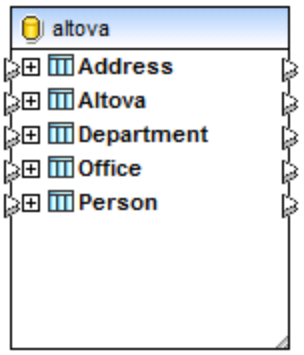
- Die Beispielfirma (entspricht der Tabelle "Altova") besteht aus einer oder mehreren Niederlassungen (Offices) (z.B. in Brenton und Vereno). In Datenbankbegriffen ausgedrückt besteht eine 1:n-Beziehung zwischen der Tabelle "Altova" und der Tabelle "Office". Anders ausgedrückt kann es für jeden **PrimaryKey**-Datensatz in der Tabelle "Altova" mehrere **ForeignKey**-Datensätze in der Tabelle "Office" geben. Jeder "Office"-Datensatz, in dem der **ForeignKey**-Wert dem **PrimaryKey**-Wert in "Altova" entspricht, wird daher als eine Niederlassung von "Altova" betrachtet.
- Jede Niederlassung besteht aus einer oder mehreren Abteilungen (departments) (z.B., "Marketing", "IT", "Development"). Auch hier besteht wieder eine 1:n-Beziehung zwischen den Tabellen "Office" und "Department".
- Schließlich besteht jede Abteilung aus einem oder mehreren Mitarbeitern (person). Daher besteht hier eine 1:n-Beziehung zwischen den Tabellen "Department" und "Person".

© 2016-2022 Altova GmbH



Tabellenbeziehungen in der Datenbank altova.mdb (Microsoft Access-Ansicht "Relationships")

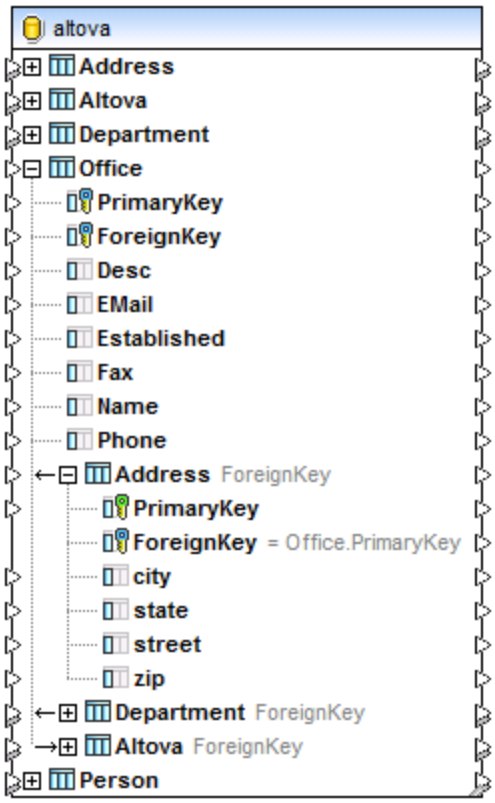
Beziehungen zwischen Datenbanktabellen sind wichtig für Mapping-Zwecke. MapForce berücksichtigt solche Datenbankbeziehungen, wenn Sie eine Datenbank zum Mapping hinzufügen. Auf diese Art können Sie beim Mappen von Daten von oder auf eine Datenbank die Datenbankbeziehungen beibehalten. Fügen Sie zum besseren Verständnis dieses Prinzips die Datenbank **altova.mdb** (mit dem Menübefehl **Einfügen | Datenbank**) zum Mapping hinzu. Nennen wir jeder der unten stehenden Tabellen eine "root"-Tabelle:



"Root"-Tabellen

Wenn Sie eine "root"-Tabelle erweitern, werden alle damit in Beziehung stehenden Tabellen in einer Baumstruktur angezeigt. Wenn Sie z.B. die Tabelle **Office erweitern**, sehen Sie, wie die Hierarchie der damit in Beziehung stehenden Tabellen angezeigt wird:

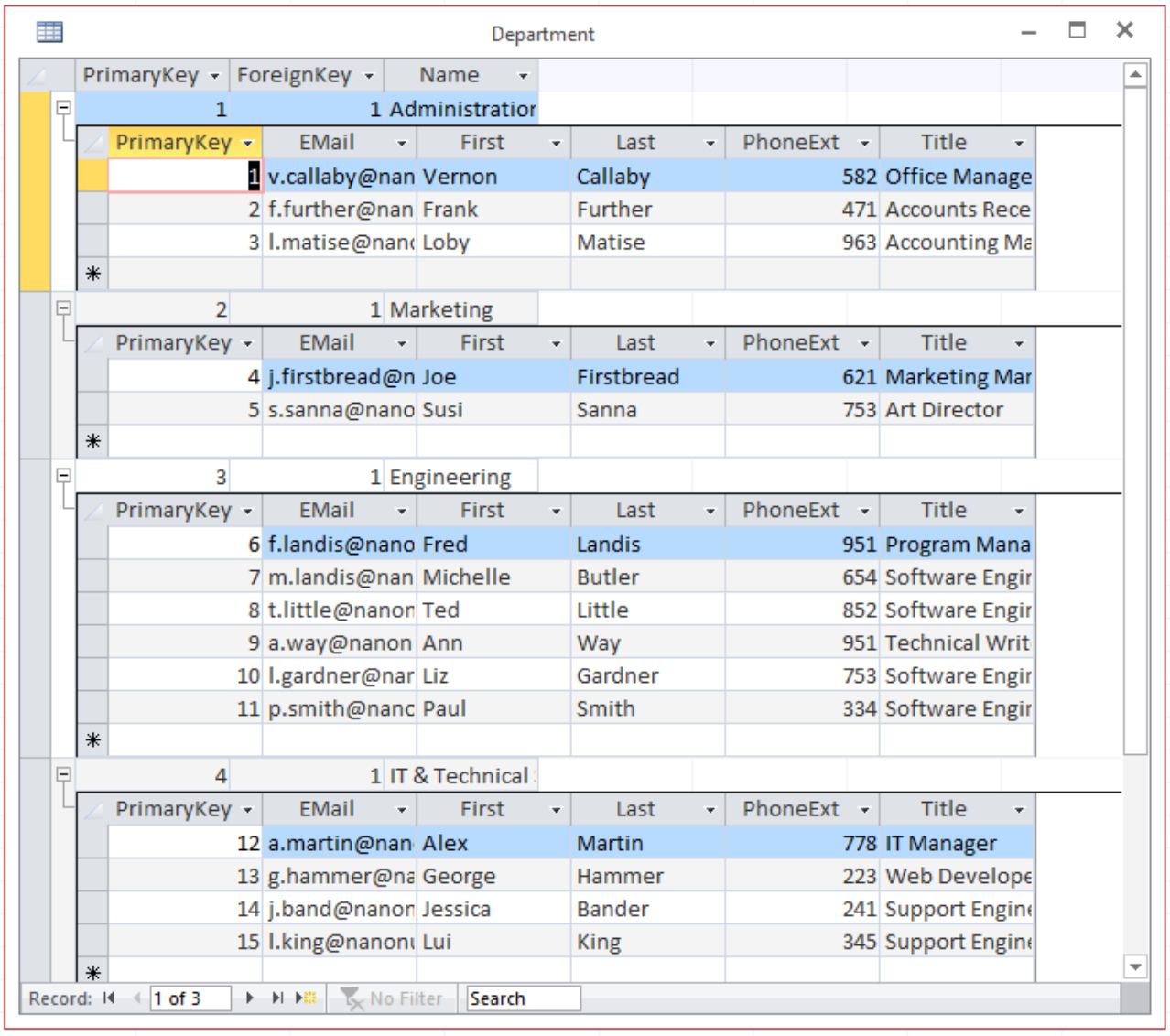
- ein Linkspfeil (←) vor einer Tabelle zeigt an, dass dies eine Child-Tabelle ist. So ist z.B. **Address** ein Child von **Office**. **Department** ist ebenfalls ein Child von **Office** und gleichzeitig eine gleichrangige Tabelle zu **Address**, daher befinden sich beide Tabelle auf derselben Einrückungsebene. Wie Sie sehen, entspricht die Beziehung im Mapping der im Beziehungsdiagramm oben.
- Ein Rechtspfeil (→) vor einer Tabelle kennzeichnet eine Parent-Tabelle. So ist z.B. **Altova** ein Parent von **Office**.



Tabellenbeziehungen in MapForce (altova.mdb Datenbank)

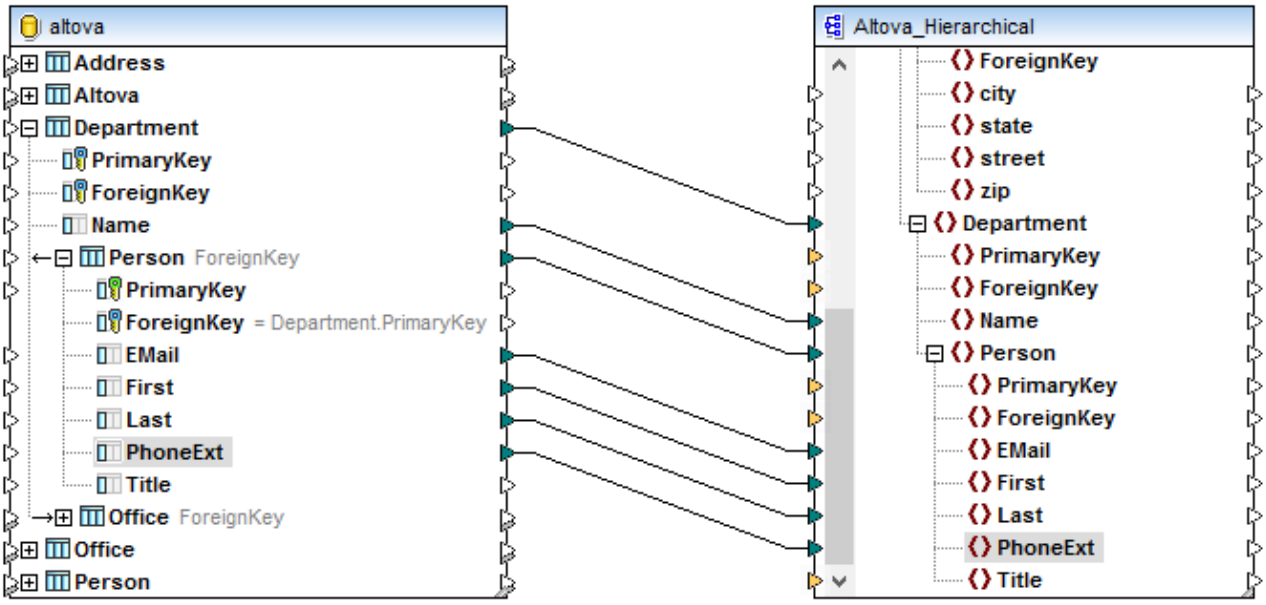
Diese hierarchische Tabellendarstellung hilft Ihnen, die bestehenden Datenbankbeziehungen zu erhalten, wenn Ihr Mapping Daten aus einer Datenbank liest oder in diese schreibt. Angenommen Sie möchten alle Datensätze aus der Tabelle **Person** gruppiert nach Abteilung (Department) in eine XML-Datei schreiben. Dabei sollte in Ihrer XML-Datei jeder Mitarbeiter, ähnlich wie in der in diesem Beispiel verwendeten Datenbank **altova.mdb**, mit einer Abteilung verknüpft sein:

© 2016-2022 Altova GmbH



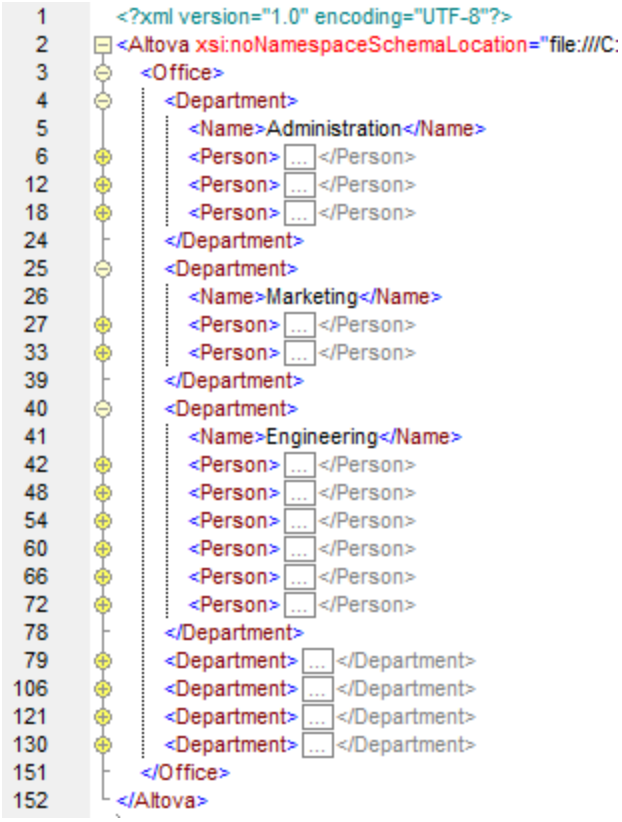
Wie Sie in der Abbildung oben sehen, hat die Abteilung "Administration" drei Mitarbeiter, "Marketing" zwei, "Engineering" sechs, usw.

Wenn Sie beim Mappen von Daten aus dieser Datenbank möchten, dass jeder Mitarbeiter der richtigen Abteilung zugeordnet wird, müssen Sie **Department** als "root"-Tabelle verwenden und die Daten anschließend von der Tabelle **Person,** die ein Child von **Department** ist, mappen.

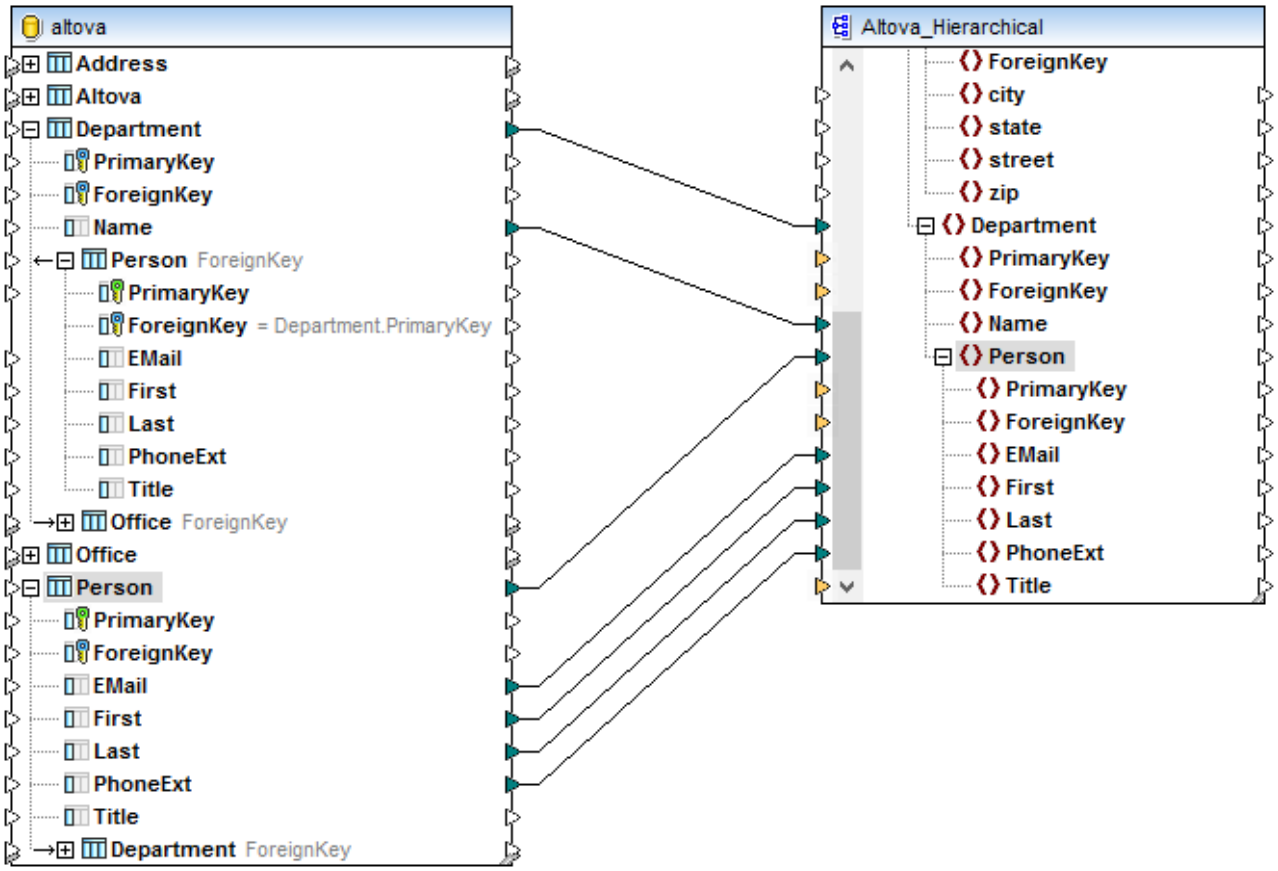


Das Mapping oben ist eine geänderte **DB_Altova_Hierarchical.mfd**-Datei aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Bei Anzeige einer Vorschau des Mappings wird als Ergebnis jeder Mitarbeiter nach Abteilung gruppiert, was Ziel dieses Mappings war. "Administration" hat also drei Mitarbeiter, "Marketing" zwei, "Engineering" sechs, usw.

© 2016-2022 Altova GmbH

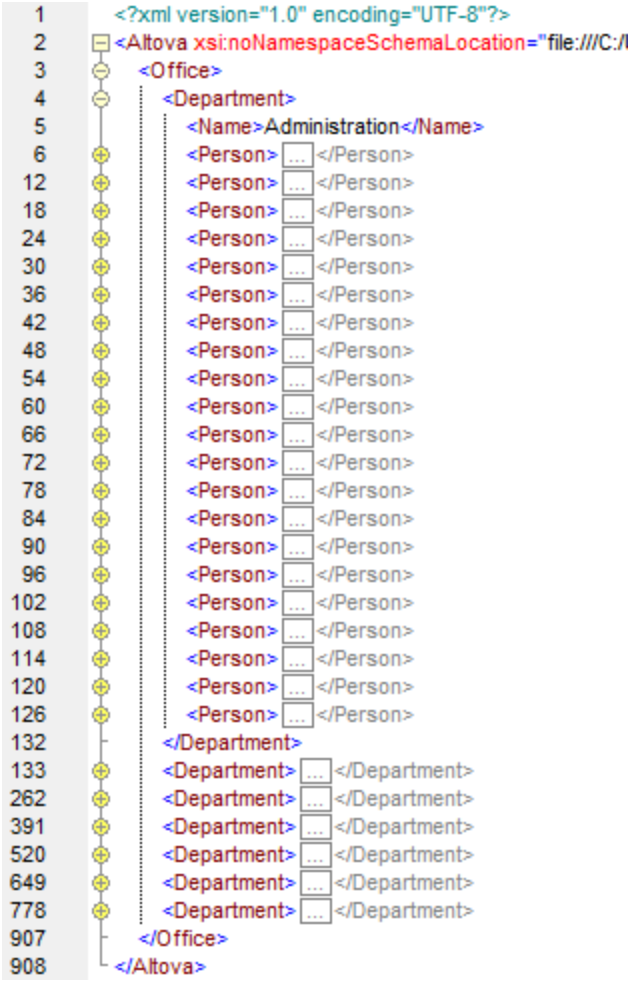


Werfen Sie nun einen Blick auf das geänderte Mapping unten, in dem die Verbindungen absichtlich so gezogen wurden, dass sowohl **Department** als auch **Person** "root"-Tabellen sind.



Wenn Sie diesmal eine Vorschau des Mappings anzeigen, werden alle Mitarbeiter (unabhängig von Ihrer eigenen Abteilung) unter jeder Ziel-Abteilung gruppiert, was nicht beabsichtigt war. Hier hat nun "Administration" 21 Mitarbeiter, ebenso wie "Marketing", "Engineering", usw.

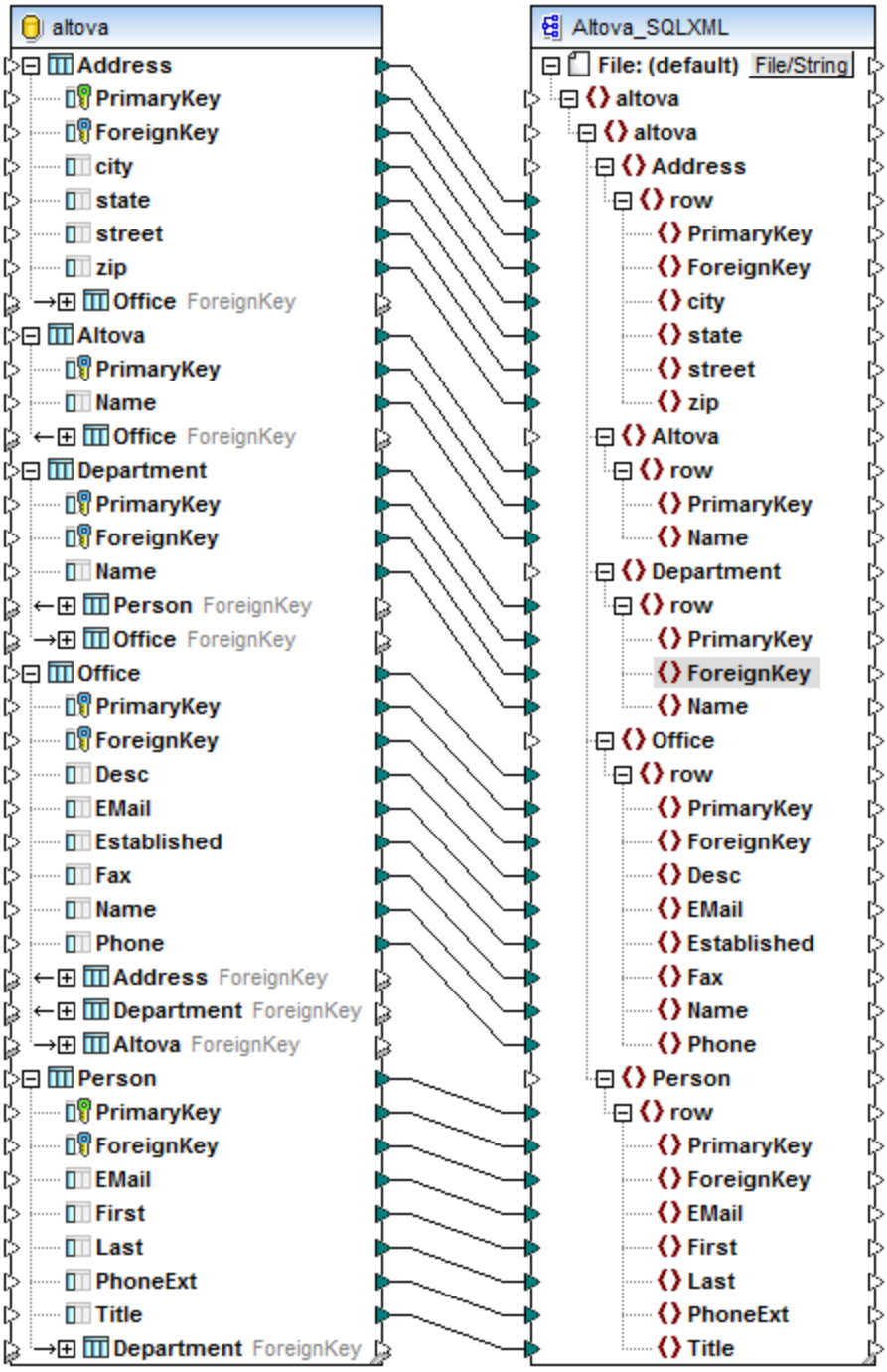
© 2016-2022 Altova GmbH



Im zweiten Beispiel werden die Datenbankbeziehungen aufgrund der Art, wie die Verbindung erstellt wurde, ignoriert.

Wenn Sie daher Datenbankbeziehungen erhalten möchten, stellen Sie sicher, dass die Verbindungen von oder auf dieselbe "root"-Tabelle, die die gewünschten Child-Verbindungen enthält, gezogen werden. Dies funktioniert sowohl bei der Quell- als auch der Zieldatenbank auf dieselbe Art. Beispiele für Datenbankmappings, in denen Beziehungen erhalten bleiben, finden Sie in den Dateien **DB_Altova_Hierarchical.mfd** und **Altova_Hierarchical_DB.mfd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** (siehe auch Einfügen von Daten in mehrere verknüpfte Tabellen 295)..

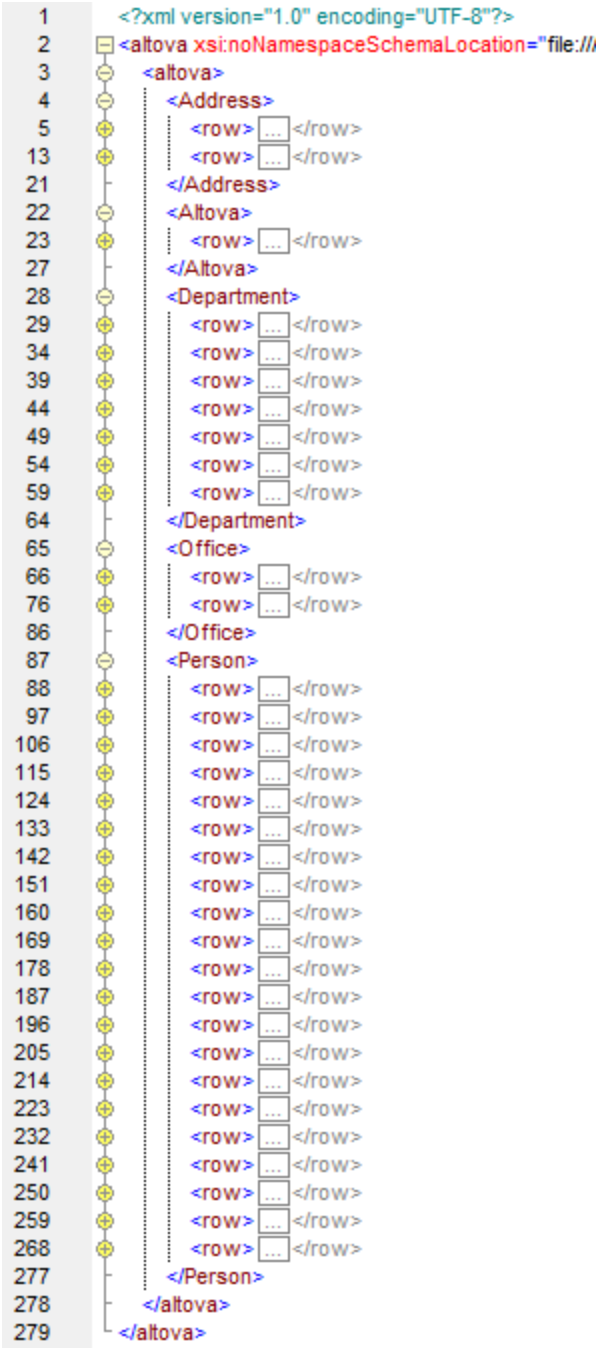
In manchen Fällen sollten Datenbankbeziehungen nicht beibehalten werden. Angenommen, Sie möchten alle Daten aus der Datenbank **altova.mdb** in eine flache, gemäß der SQL/XML-Spezifikation (Teil 14 der Structured Query Language (SQL)-Spezifikation) erstellte XML-Datei exportieren. Diese Art von Mapping sehen Sie in der Beispieldatei **DB_Altova_SQLXML.mfd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Ziel des Mappings ist, Datenbankdaten als flache XML-Datei anzulegen. Das SQL/XML-Zielschema wurde mit XMLSpy mit dem Menübefehl **Konvertieren | XML-Schema von DB-Struktur erzeugen** generiert.



DB_Altova_SQLXML.mfd

Wie oben gezeigt, hat jede Datenbanktabelle ein entsprechendes Element in der XML-Zieldatei. Bei Anzeige einer Vorschau auf das Mappingergebnis sehen Sie, dass die Datenbankzeilen aus den einzelnen Tabellen in die "row"-Elemente in der Zieldatei geschrieben werden.

© 2016-2022 Altova GmbH


<?xml version="1.0" encoding="UTF-8"?>
<altova xsi:noNamespaceSchemaLocation="file:///
<altova>
<Address>
<row> ... </row>
<row> ... </row>
</Address>
<Altova>
<row> ... </row>
</Altova>
<Department>
<row> ... </row>
<row> ... </row>
<row> ... </row>
<row> ... </row>
<row> ... </row>
<row> ... </row>
<row> ... </row>
</Department>
<Office>
<row> ... </row>
<row> ... </row>
</Office>
<Person>
<row> ... </row>
<row> ... </row>
<row> ... </row>
<row> ... </row>
<row> ... </row>
<row> ... </row>
<row> ... </row

4.2.2.6 Definieren von lokalen Beziehungen

Wenn es zwischen Datenbanktabellen keine explizit definierten Beziehungen gibt, können Sie solche Beziehungen lokal in MapForce definieren. So können Sie etwa von MapForce aus Primär- und Sekundärschlüsselbeziehungen zwischen Spalten unterschiedlicher Tabellen erstellen, ohne dass sich dies in irgendeiner Art auf die Datenbank auswirkt. Als Primär- oder Sekundärschlüssel kann jede beliebige Datenbankspalte verwendet werden. Außerdem können zusätzlich zu den bereits bestehenden Beziehungen in der Datenbank neue Beziehungen erstellt werden. Lokal definierte Beziehungen werden zusammen mit dem Mapping gespeichert.

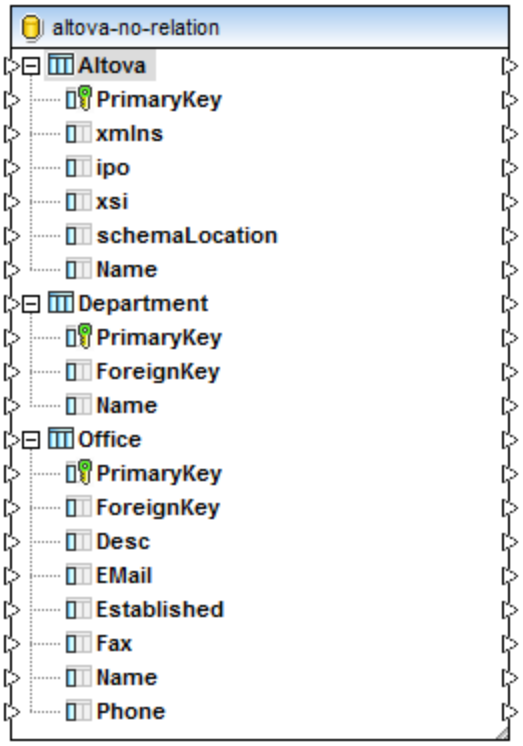
Diese "on-the-fly"-Beziehungen werden in MapForce als **lokale Beziehungen** bezeichnet. In der folgenden Tabelle sind alle möglichen Felder aufgelistet, zwischen denen lokale Beziehungen definiert werden können. Auch gemischte Beziehungen (z.B. Mappen der Ausgabe einer gespeicherten Prozedur auf eine Datenbankspalte) sind möglich. Die an einer Beziehung beteiligten Felder müssen denselben oder einen kompatiblen Datentyp haben.

Primärschlüssel/eindeutiger Schlüssel	Sekundärschlüssel
• Spalte einer Datenbanktabelle oder -ansicht • Output-Parameter oder Rückgabewert einer gespeicherten Prozedur, siehe auch Gespeicherte Prozeduren 402 • Spalte einer von einer gespeicherten Prozedur zurückgegebenen Datensatzstruktur* • Spalte einer benutzerdefinierten SELECT-Anweisung, siehe auch SELECT-Anweisungen als virtuelle Tabellen 368.	• Spalte einer Datenbanktabelle oder -ansicht • Input-Parameter einer gespeicherten Prozedur • Input-Parameter einer benutzerdefinierten SELECT-Anweisung

** Anwendbar, wenn die gespeicherte Prozedur entweder als Datenquelle (ohne Parameter) oder Funktion (mit Input- und Output-Parametern) aufgerufen wird. Damit die Datensatzstruktur für die Auswahl zur Verfügung steht, müssen Sie die gespeicherte Prozedur einmal ausführen, um die Datensatzstruktur abzurufen.*

Die in diesem Beispiel verwendete Datenbank **altova-no-relation.mdb** ist eine vereinfachte Version der mit MapForce mitgelieferten Datenbank **altova.mdb**. Die Tabellen "Person" und "Address" sowie alle Tabellenbeziehungen wurden in Microsoft Access entfernt. Wie unten gezeigt, hat keine der in der **altova-no-relation**-Struktur sichtbaren Tabellen Child-Tabellen; alle Tabellen befinden sich auf derselben "root"-Ebene. Der Inhalt der einzelnen Tabellen beschränkt sich auf die enthaltenen Felder.

© 2016-2022 Altova GmbH

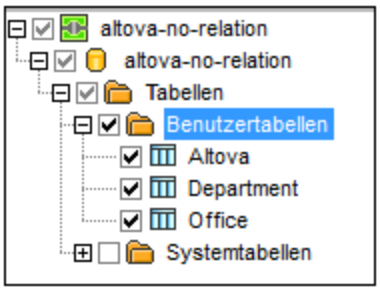


Datenbankstruktur ohne explizite Beziehungen

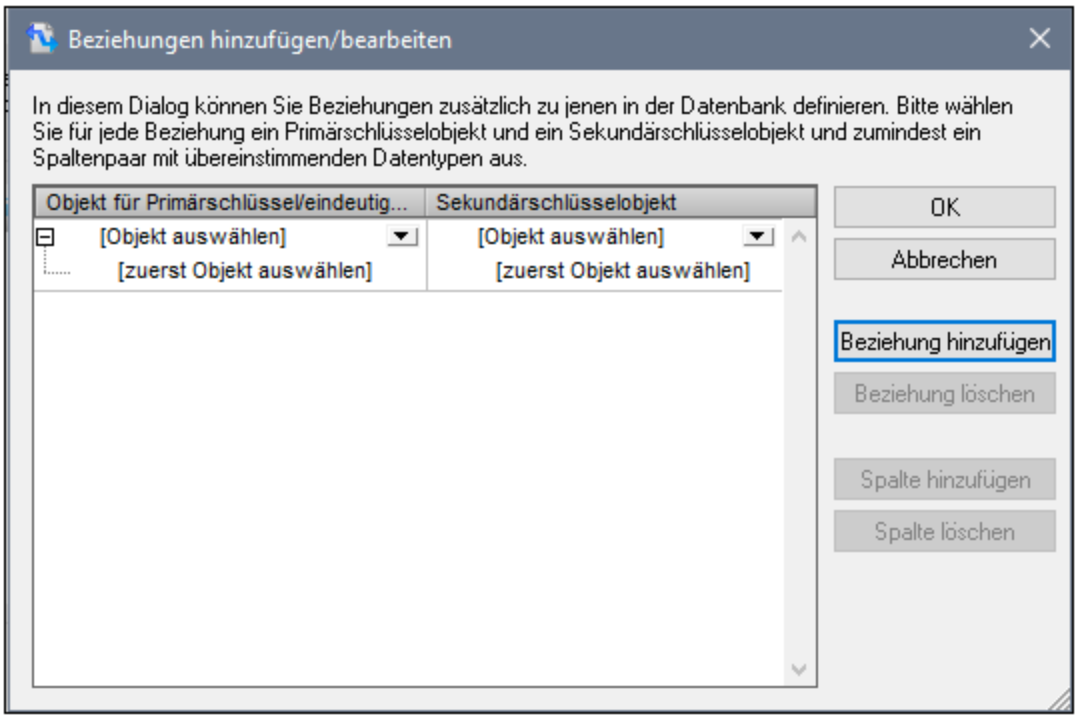
Ziel des Beispiels ist, die Niederlassungen (offices) von "Altova" sowie die Abteilungen in den einzelnen Niederlassungen anzuzeigen. Beachten Sie, dass die Primär- und Sekundärschlüsselbeziehungen in **altova-no-relation.mdb,** wie oben erwähnt, nicht explizit vorhanden sind. Sie sind nur logischerweise (implizit) vorhanden, daher werden wir diese zu diesem Zweck lokal in MapForce neu erstellen.

Lokale Beziehungen können definiert werden, wenn Sie eine Datenbank zum Mapping hinzufügen oder durch Rechtsklick auf die Datenbankkomponente und Auswahl des Kontextmenübefehls **Datenbankobjekte hinzufügen/entfernen/bearbeiten,** wie in den folgenden Schritten unten gezeigt.

1. Klicken Sie im Menü **Einfügen** auf **Datenbank**.
2. Klicken Sie im Verbindungsassistenten auf **Microsoft Access (ADO)** und anschließend auf **Weiter**.
3. Navigieren Sie zur Datenbank **altova-no-relation.mdb** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** und klicken Sie auf **Verbinden**.
4. Aktivieren Sie das Kontrollkästchen **Benutzertabellen**.

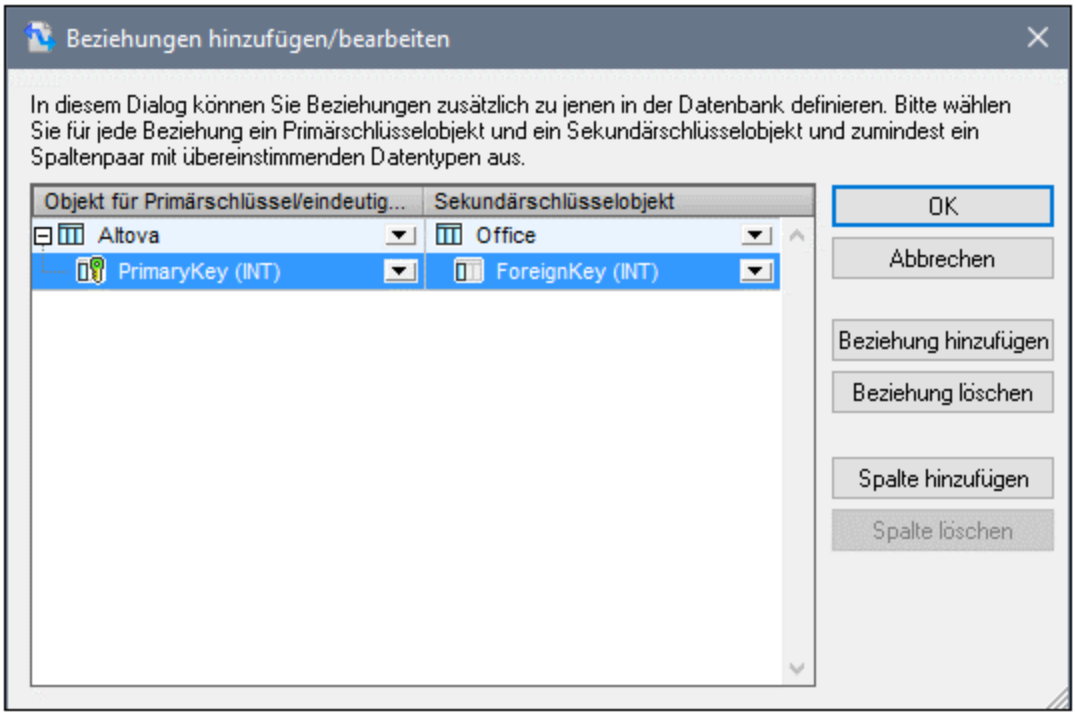


5. Klicken Sie in der Symbolleiste auf die Schaltfläche **Beziehungen hinzufügen/bearbeiten**.
6. Daraufhin wird das Dialogfeld "Tabellenbeziehungen hinzufügen/bearbeiten" geöffnet. Klicken Sie auf **Beziehung hinzufügen**.



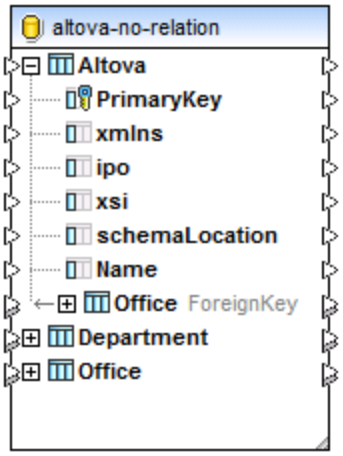
7. Wählen Sie Werte aus den beiden Dropdown-Listen aus, um, wie unten gezeigt, zwischen den Tabellen "Altova" und "Office" eine Primär- und Sekundärschlüsselbeziehung zu erstellen. Über die beiden Dropdown-Listen können Sie die Tabellen oder Datenbankobjekte, für die Sie Beziehungen erstellen möchten, auswählen. In der linken Liste ist das Objekt definiert, in dem der Primärschlüssel/eindeutige Schlüssel gespeichert ist, während in der rechten Liste das Sekundärschlüsselobjekt definiert ist. Das Objekt für den Primärschlüssel/eindeutigen Schlüssel-Objekt bildet das Parent-Objekt in MapForce, das Sekundärschlüsselobjekt wird in der Datenbankkomponente als Child angezeigt (siehe auch Behandlung von Datenbankbeziehungen 266).

 © 2016-2022 Altova GmbH

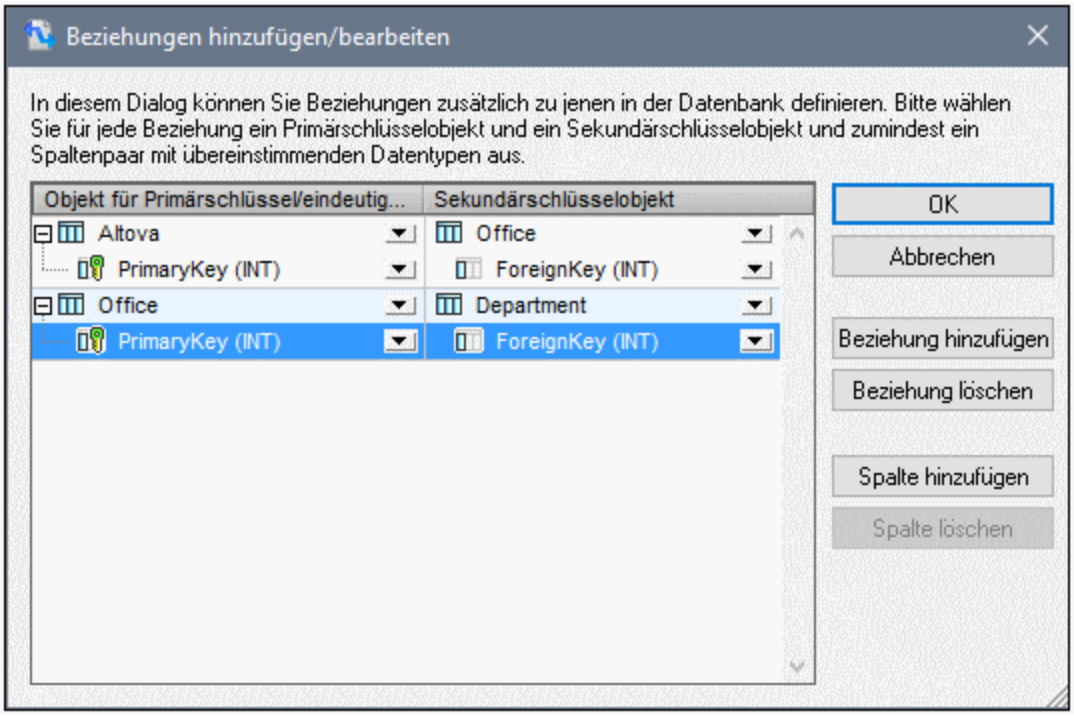


8. Klicken Sie auf **OK,** um die Definition der lokalen Beziehung fertig zu stellen und klicken Sie anschließend auf die Schaltfläche **Einfügen**, um die Datenbank in den Mapping-Bereich einzufügen.

Sie haben in diesem Schritt eine lokale Beziehung zwischen der Spalte **PrimaryKey** der Tabelle "Altova" und der Spalte **ForeignKey** der Tabelle "Office" erstellt. Die Root-Tabelle "Altova" ist nun, wie unten gezeigt, ein Parent der Tabelle "Office". Die Tabelle "Office" wird dabei mit eigenem Erweiterungssymbol unterhalb der Tabelle "Altova" als damit in Beziehung stehende Tabelle angezeigt.



Noch ist das Ziel des Mappings aber nicht erreicht. Gehen Sie nun auf dieselbe Art vor, um zwischen den Tabellen **Office** und **Department** ebenfalls eine Beziehung zu erstellen, wie unten gezeigt.



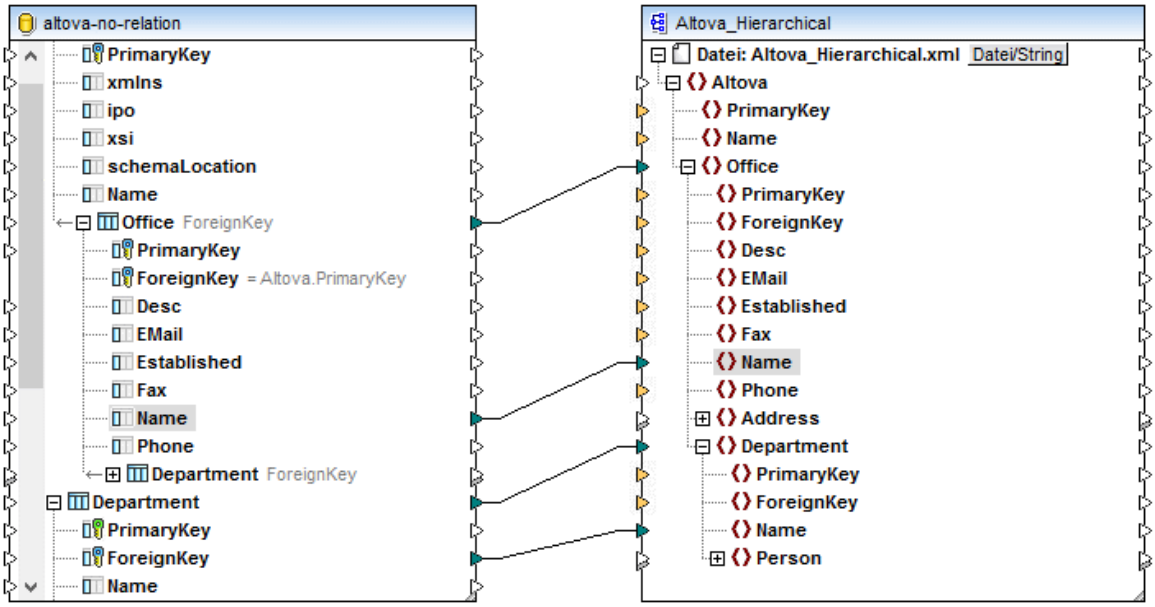
Um das Dialogfeld "Beziehungen hinzufügen/bearbeiten" erneut zu öffnen, klicken Sie mit der rechten Maustaste auf die Datenbankkomponente und wählen Sie im Kontextmenü den Befehl **Datenbankobjekte hinzufügen/entfernen/bearbeiten**.

Fügen Sie schlussendlich das Zielschema zum Mapping hinzu:

1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei einfügen**.
2. Navigieren Sie zur Datei **Altova_Hierarchical.xsd** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Wenn Sie aufgefordert werden, eine XML-Beispieldatei anzugeben, klicken Sie auf **Überspringen**. Wenn Sie aufgefordert werden, ein Root-Element auszuwählen, wählen Sie "Altova" aus.

Beachten Sie, dass alle Verbindungen von derselben "root"-Tabelle aus (in diesem Fall "Altova") hierarchisch gezogen wurden, um die Beziehungen zwischen den Tabellen in der XML-Zieldatei zu bewahren. Nähere Informationen dazu finden Sie unter Behandlung von Datenbankbeziehungen 266.

 © 2016-2022 Altova GmbH



Nachdem Sie das oben gezeigte Mapping definiert haben, klicken Sie auf das Register **Ausgabe**, um eine Vorschau auf das Ergebnis zu sehen. Im Mapping-Ergebnis sehen Sie die unter den jeweiligen Niederlassungen (office) verschachtelten Department-Elemente. Damit wurde das beabsichtigte Ergebnis des Mappings erzielt.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Altova xsi:noNamespaceSchemaLocation="file:///C:/U
3    <Office>
4      <Name>Nanonull, Inc.</Name>
5      <Department>
6        <Name>Administration</Name>
7      </Department>
8      <Department>
9        <Name>Marketing</Name>
10     </Department>
11     <Department>
12       <Name>Engineering</Name>
13     </Department>
14     <Department>
15       <Name>IT &amp;amp; Technical Support</Name>
16     </Department>
17   </Office>
18   <Office>
19     <Name>Nanonull Partners, Inc.</Name>
20     <Department>
21       <Name>Administration</Name>
22     </Department>
23     <Department>
24       <Name>Marketing</Name>
25     </Department>
26     <Department>
27       <Name>IT &amp;amp; Technical Support</Name>
28     </Department>
29   </Office>
30 </Altova>
```

4.2.2.7 Ausführen von Mappings, die Datenbanken modifizieren

Wenn ein Mapping Datenbankdaten auf irgendeine Art modifiziert (z.B. durch Einfügen, Aktualisieren oder Löschen von Datensätzen) , werden die Änderungen durch den Prozessor, der das Mapping ausführt, auf die Datenbank angewendet. Als ausführender Prozessor kann MapForce, MapForce Server (sowohl die Standalone Edition als auch unter FlowForce Server-Verwaltung) oder die Ausführungsumgebung, des für C++, C# oder Java generierten Codes sein.

Wenn Sie direkt in MapForce (durch Klicken auf das Register **Ausgabe**) eine Vorschau auf das Mapping-Ergebnis aufrufen, wird ein Aktualisierungs-Skript angezeigt. Das Skript wird erst dann tatsächlich auf die Datenbank angewendet, wenn Sie dies explizit anordnen; es steht nur zur Vorschau zur Verfügung.

```
1  /*
2  The following SQL statements are only for preview and may not be executed in another SQL query tool!
3  To execute these statements use function "Run SQL-script" from menu "Output".
4  Connect to database using the following connection-string:
5  Data Source=C:/Users/altova/Documents/Altova/MapForce2016/MapForceExamples/AltovaTarget.mdb;Provider=Microsoft.Jet.OLEDB.4.0
6  */
7
8  -- begin transaction
9
10 DELETE FROM [Address]
11
12 DELETE FROM [Person]
13
14 DELETE FROM [Department]
15
16 DELETE FROM [Office]
17
18 DELETE FROM [Altova]
19
20 SELECT IIF(MAX([Altova].[PrimaryKey]) IS NULL,0,MAX([Altova].[PrimaryKey]))+1 AS [PrimaryKey] FROM [Altova]
21 -->>> %PrimaryKey1%
22
23 INSERT INTO [Altova] ([Name], [PrimaryKey]) VALUES ('Organization Chart', '%PrimaryKey1%')
```

Ausgabevorschau eines Mappings, durch das eine Datenbank modifiziert wird (Altova_Hierarchical_DB.mfd)

Das Skript enthält SQL-Pseudoanweisungen, die nur zu Informationszwecken dienen. Dies entspricht nicht genau den auszuführenden Anweisungen. Wenn z.B. für eine Tabelle mehrere Aktionen definiert wurden (wie z.B. Aktualisiere, wenn.... Rest einfügen"), wird in der Vorschau nur die erste Aktion angezeigt, da die zweite Aktion auf Basis einer Bedinung ausgeführt wird.

Dieses SQL-Pseudoskript darf nicht manuell mit Hilfe anderer SQL-Tools als der oben genannten Ausführungsprozessoren auf die Datenbank angewendet werden. Das Skript enthält eventuell Werteformatierungen, die von externen SQL-Editoren nicht interpretiert werden können.

Wenn Sie die Mapping-Änderungen direkt von MapForce aus an der Datenbank ausführen möchten, klicken Sie im Menü **Ausgabe** auf den Befehl **SQL-Script ausführen**. Beachten Sie, dass die Datenbank mit dieser Aktion sofort geändert wird.

Wenn das Mapping mit MapForce Server (der Standalone Edition oder unter FlowForce Server-Verwaltung) ausgeführt wird, werden die Änderungen mit sofortiger Auswirkung an der Datenbank ausgeführt. Dasselbe ist auch bei generiertem Code der Fall; die Änderungen an der Datenbank werden sofort wirksam, wenn Sie den Code kompilieren und ausführen (z.B. durch Auswahl des Befehls **Run** in Visual Studio).

© 2016-2022 Altova GmbH

Ihre MapForce-Installation enthält im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** mehrere Beispieldatenbanken (Microsoft Access- oder SQLite-Dateien). Es empfiehlt sich, Datenbankänderungen nicht mit dem Befehl **SQL-Script ausführen** von MapForce aus an einer der mitgelieferten Datenbanken auszuführen, da die Beispieldatei dadurch unbrauchbar werden kann. Damit die Originaldaten nicht überschrieben werden, erstellen Sie am besten eine Sicherungskopie des gesamten Ordners **<Dokumente>\Altova\MapForce2023\MapForceExamples\,** bevor Sie Dateien darin aktualisieren.

Informationen zum Ausführen von Mappings in anderen Ausführungsumgebungen als MapForce finden Sie in den folgenden Kapiteln:

- Bereitstellen von Mappings auf FlowForce Server 1445
- Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442
- Code Generator 1563

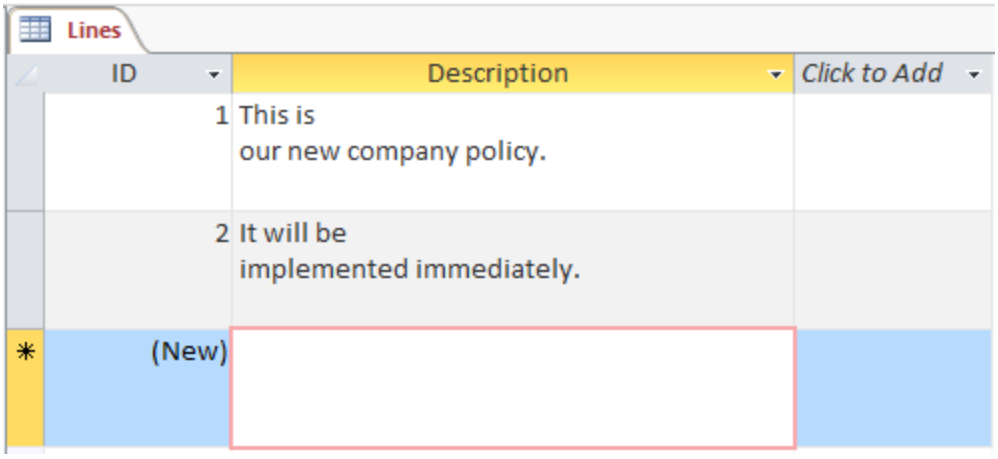
4.2.2.8 Ersetzen von Sonderzeichen

Beim Transformieren von Datenbankdaten müssen Sie unter Umständen bestimmte Sonderzeichen wie z.B. Wagenrücklauf (CR)/ Zeilenvorschub (LF) aus der Datenquelle entfernen. Dies können Sie mit Hilfe einer der folgenden Methoden bewerkstelligen:

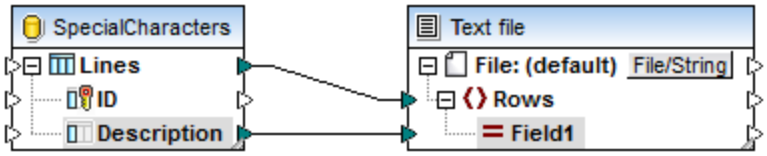
1. Definieren Sie eine Node-Funktion für das jeweilige Datenbankfeld (oder sogar für mehrere Felder), das bzw. die verarbeitet werden soll(en). Die Node-Funktion erhält als Input den Wert des Datenbankfelds. Wenden Sie eine Verarbeitung darauf an und geben Sie das Ergebnis an das Mapping zurück. Nähere Informationen zu dieser Methode finden Sie unter Standardwerte und Node-Funktionen 951.
2. Verarbeiten Sie Datenbankwerte mit Hilfe von vordefinierten MapForce-Funktionen. Bestimmte Zeichen, darunter auch Steuerelementzeichen, können Sie mit Hilfe der Funktion **char-from-code** 1104 aus der "core"-Bibliothek identifizieren. Um Werte zu ersetzen, verwenden Sie die Funktion **replace** 1175 aus der "lang"-Bibliothek.

In diesem Kapitel wird die zweite Methode erläutert. Als Datenbank wird in diesem Beispiel eine Microsoft Access-Datenbank verwendet, doch können Sie auch mit anderen Datenbanktypen auf dieselbe Art arbeiten.

Angenommen, Sie haben eine Microsoft Access-Datenbank bestehend aus einer Tabelle "Lines", die zwei Spalten enthält: "ID" und "Description".



Ziel ist es, die einzelnen Beschreibungen in eine CSV-Datei zu extrahieren (eine Beschreibung pro Zeile); ein Mapping, womit Sie dies erreichen, könnte folgendermaßen aussehen:



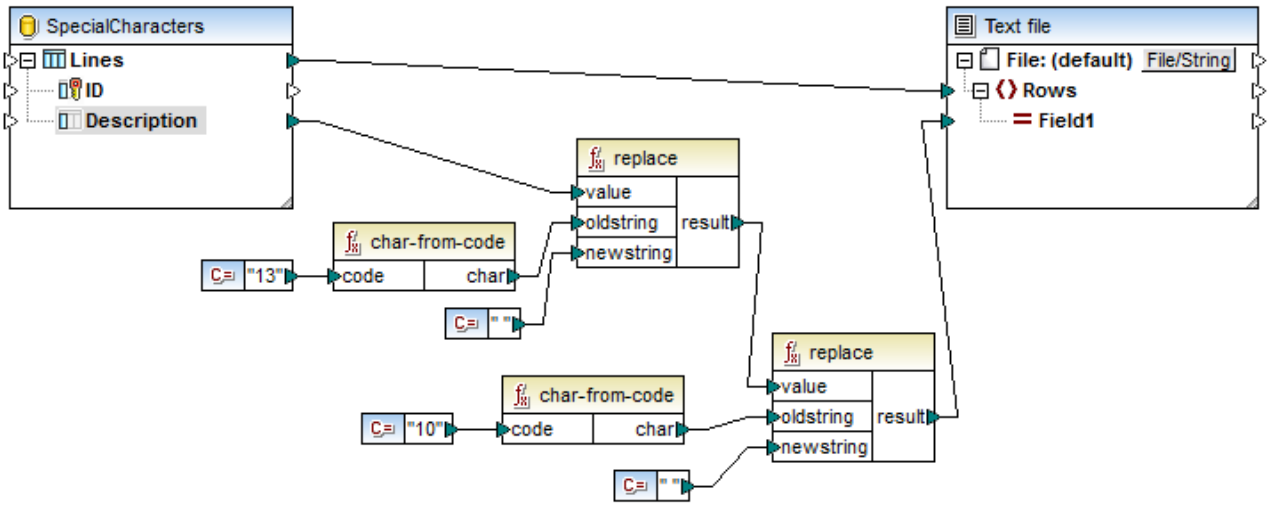
Da jedoch jede "Description"-Zeile in Access mehrere durch CR/LF-Zeichen getrennte Zeichen enthält, enthält auch das Mapping Zeilenumbrüche, was nicht erwünscht ist:

```
1   "This is
2   our new company policy."
3   "It will be
4   implemented immediately."
5
```

Um dieses Problem zu lösen, werden wir die Funktionen **`char-from-code`** und **`replace`** aus der Bibliothek der vordefinierten MapForce-Funktionen zum Mapping hinzufügen (siehe auch Hinzufügen einer Funktion zum Mapping 943). Jede Beschreibung muss verarbeitet werden, sodass die oben genannten Zeichen durch ein Leerzeichen ersetzt werden.

In der Unicode-Tabelle (http://www.unicode.org/charts/) entsprechen die Zeichen LF und CR den Hexadezimalzeichen **hex 0A | dec 10** bzw. **hex 0D | dec 13**. Das Mapping muss daher geändert werden, um die Unicode-Dezimalwerte 13 und 10 in einen String zu konvertieren, damit die Daten mit Hilfe der **`replace`** Funktion weiter bearbeitet werden können.

© 2016-2022 Altova GmbH



Bei Anzeige einer Vorschau auf das Mapping sehen Sie jetzt, dass die Zeichen CR/LF in den einzelnen Datenbankfeldern durch Leerzeichen ersetzt wurden.

```
1    This is  our new company policy.
2    It will be  implemented immediately.
3
```

4.2.2.9 Behandlung von Nullwerten

Um zur Mapping-Laufzeit zu überprüfen, ob ein Datenbankfeld den Wert Null hat, verwenden Sie die MapForce-Bibliotheksfunktionen **is-null**[1123] und **is-not-null**[1122]. Um von MapForce aus zu sehen, ob eine Tabelle Null-Felder hat, fragen Sie sie über den Datenbank Browser ab (siehe Anzeigen und Abfragen von Datenbanken[388]).

Um ein Datenbankfeld auf Null zu setzen, verwenden Sie die Funktion **set-null**[1123].

Um Null-Datenbankwerte durch einen String zu ersetzen, verwenden Sie die Funktion **substitute-null**[1124]. Ein Beispielmapping dazu ist **DB_ApplicationList.mfd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**.

Informationen zum Behandeln von NULLwertvergleichen in Mappings, in denen Datenbanken aktualisiert werden, finden Sie unter Behandlung von Nullwerten in Datenbankaktionen[333].

Informationen zum Behandeln von Nullwerten beim Mappen von Datenbanken von oder auf XML-Dokumente finden Sie unter Nullwerte / Nillable Werte[132].

4.2.2.10 Generieren von sequenziellen und eindeutigen Werten

Wenn Sie Daten in eine Datenbank einfügen oder einer Datenbank aktualisieren, müssen Sie für die Datenbankfelder, die keinen Input aus der gemappten Quellkomponente erhalten, manchmal "on-the-fly"

sequenzielle oder eindeutige Werte erstellen. Verwenden Sie zu diesem Zweck die folgenden vordefinierten MapForce-Funktionen:

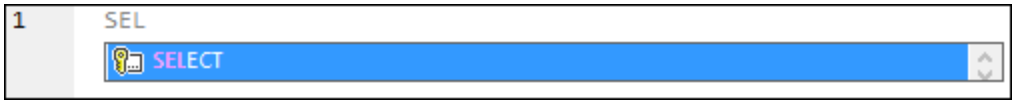
- **auto-number**[1054] (aus der Bibliothek "core | generator functions"). Diese Funktion dient im Allgemeinen dazu, Primärschlüsselwerte für ein numerisches Feld zu generieren.
- **create-guid**[1151] (aus der Bibliothek "lang | generator functions"). Mit Hilfe dieser Funktion wird ein global eindeutiger Identifier (als hexadezimal kodierter String) für das jeweilige Feld generiert.

Beachten Sie, dass Werte für Datenbankfelder auch mit Hilfe von Datenbank-generierten Werten geschrieben werden können. Diese Option steht im Dialogfeld "Datenbankaktionen" (siehe Einstellungen für Datenbankaktionen[336]) zur Verfügung und eignet sich besonders zum Generieren von Primärschlüsseln.

4.2.2.11 SQL-Autokomplettierungsvorschläge

Bei der Eingabe von SQL-Anweisungen in bestimmten Zusammenhängen kann MapForce automatisch Textvorschläge machen. Die Autokomplettierungsfunktion steht in folgenden Zusammenhängen zur Verfügung:

- SQL Editor (siehe Anzeigen und Abfragen von Datenbanken[388])
- Textfeld "Benutzerdefinierte SQL" im Dialogfeld "Datenbankaktionen" (siehe Einstellungen für Datenbankaktionen[336])
- Dialogfeld "SQL SELECT-Anweisung eingeben" (siehe Erstellen von SELECT-Anweisungen[369])



Autokomplettierung

Mit Hilfe der **Nach oben** und **Nach unten**-Tasten können Sie durch die Liste der Vorschläge navigieren. Um einen Vorschlag auszuwählen, klicken Sie darauf oder drücken Sie die **Eingabetaste**.

So deaktivieren Sie Autokomplettierungsvorschläge:

1. Klicken Sie im Menü **Extras** auf **Optionen** (oder drücken Sie **Strg+Alt+O**).
2. Klicken Sie unter **Datenbank** auf **SQL Editor**.
3. Deaktivieren Sie unter **Eingabehilfen** das Kontrollkästchen **Automatisch öffnen**.

So rufen Sie die Autokomplettierungsvorschläge bei Bedarf manuell auf:

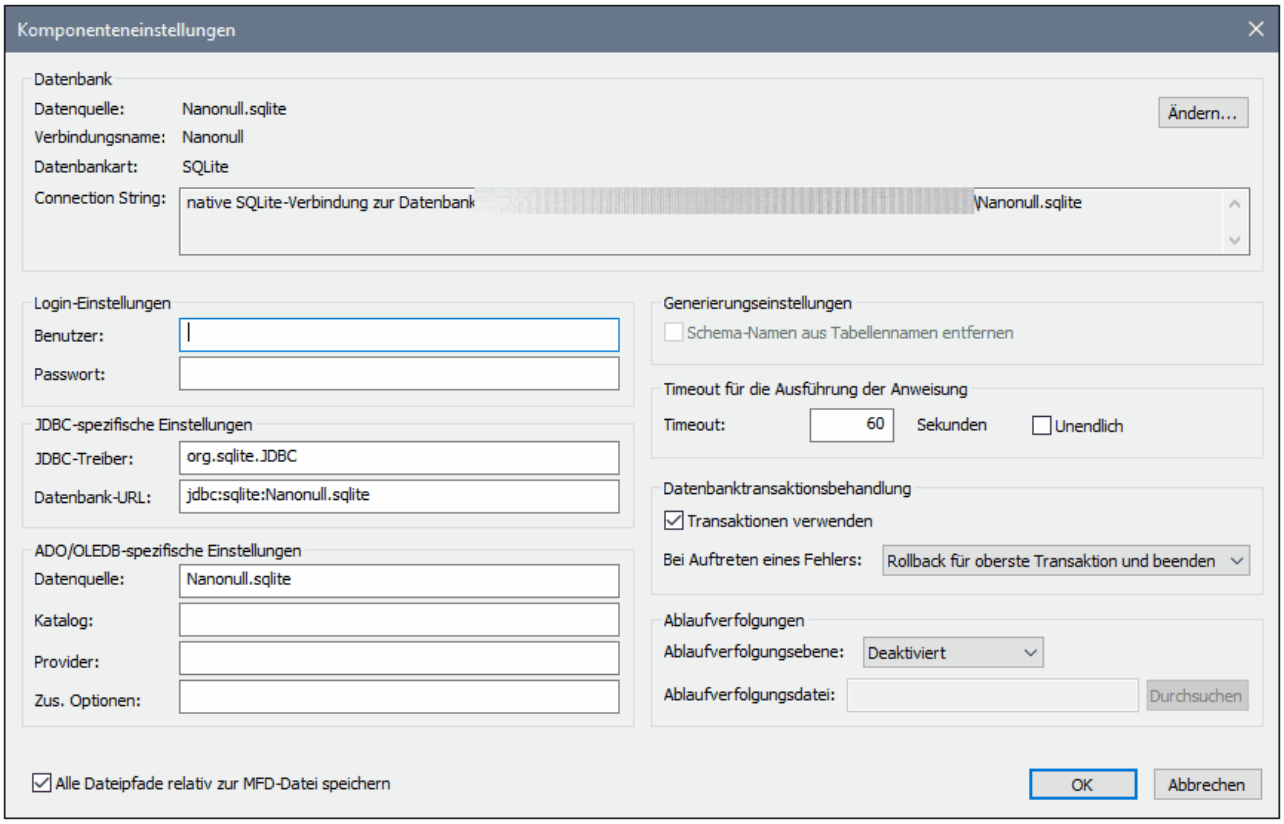
- Drücken Sie **Strg+Leertaste**.

4.2.2.12 Datenbank-Komponenteneinstellungen

Nachdem Sie eine Datenbankkomponente zum Mapping-Bereich hinzugefügt haben, können Sie die Einstellungen dafür über das Dialogfeld "Komponenteneinstellungen" konfigurieren. Das Dialogfeld "Komponenteneinstellungen" kann auf eine der folgenden Arten aufgerufen werden:

© 2016-2022 Altova GmbH

- Wählen Sie im Mapping die Komponente aus und klicken Sie im Menü **Komponente** auf **Eigenschaften.**
- Doppelklicken Sie auf die Titelleiste der Komponente.
- Klicken Sie mit der rechten Maustaste auf die Komponente und wählen Sie **Eigenschaften**.



Dialogfeld "Datenbank-Komponenteneinstellungen"

Es stehen die folgenden Einstellungen zur Verfügung:

Datenbank

In dieser Gruppe werden die Datenbankverbindungsinformationen angezeigt. Klicken Sie auf **Ändern,** um eine andere Datenbank auszuwählen oder die Datenbankobjekte in der bestehenden Datenbankkomponente neu zu definieren. Konnektoren zu Tabellen desselben Namens werden beibehalten. Sie können die Tabellen in der Komponente auch durch Rechtsklick auf eine Datenbankkomponente und Auswahl von **Datenbankobjekte hinzufügen/entfernen/bearbeiten** ändern.

Datenquelle	Gibt den Namen der aktuellen Datenquelle an. Bei dateibasierten Datenbanken kann es sich hierbei um einen Pfad im Dateisystem handeln. Über diese Einstellung können Sie feststellen, ob eine dateibasierte Datenbank über einen absoluten oder über einen relativen Pfad zum Mapping hinzugefügt wurde. Im Fall von

	relativen Pfaden enthält "Datenquelle" einen Pfad; im Fall von absoluten Pfaden enthält das Feld nur den Dateinamen der Datenbank.
Verbindungsname	Gibt den Namen der Verbindung an. Dieser Name wird von MapForce automatisch generiert. Normalerweise ist er mit dem der Datenquelle identisch, kann aber auch ein so genannter "Aliasname" sein, wenn Sie die Verbindung über globale Altova-Ressource[1462] herstellen. Wenn im Mapping mehrere Datenbankkomponenten mit demselben Verbindungsnamen vorhanden sind, erhält der Name die Form "<verbindung1>", "<verbindung2>", usw.
Datenbankart	Definiert die Art der Datenbank.
Connection String	Zeigt den aktuellen Datenbank Connection String an. Dieses schreibgeschützte Feld wird anhand der Informationen generiert, die Sie beim Erstellen oder Ändern der Datenbankverbindung eingegeben haben.

Login-Einstellungen

Die Login-Einstellungen werden für alle Codegenerierungsziele und den Built-in-Ausführungsprozessor verwendet.

Benutzer	Hier können Sie den Benutzernamen für die Datenbankverbindung ändern. Obligatorisch, wenn für die Verbindung mit der Datenbank ein Benutzername erforderlich ist.
Passwort	Hier können Sie das Passwort für die Datenbankverbindung ändern. Obligatorisch, wenn für die Verbindung mit der Datenbank ein Passwort erforderlich ist.

JDBC-spezifische Einstellungen

Diese Einstellungen werden für die Herstellung der Datenbankverbindung verwendet, wenn das Mapping eine JDBC-Verbindung enthält und von generiertem Java-Code oder MapForce Server ausgeführt wird.

Anmerkung: ADO-, ADO.NET- und ODBC-Verbindungen werden in JDBC konvertiert (und es werden die unten stehenden JDBC-Einstellungen angewendet), wenn das Mapping auf einem Linux- oder macOS-Rechner ausgeführt wird, siehe Datenbankmappings in verschiedenen Ausführungsumgebungen[166].

JDBC-Treiber	Zeigt den derzeit aktiven Treiber für die Datenbankkomponente an. Beim Definieren der Datenbankkomponente wird automatisch der Standardtreiber übernommen. Sie können den hier definierten Treiber Ihren Bedürfnissen entsprechend ändern. Achten Sie darauf, dass die Syntax des Eintrags im Feld "Datenbank-URL" mit dem von Ihnen gewählten Treiber konform ist.
Datenbank-URL	URL der aktuell ausgewählten Datenbank. Stellen Sie sicher, dass dieser Eintrag mit der JDBC-Treibersyntax des jeweils im

© 2016-2022 Altova GmbH

	Feld JDBC-Treiber definierten Treibers konform ist.

ADO/OLEDB-spezifische Einstellungen

Diese Einstellungen werden für die Herstellung der Datenbankverbindung verwendet, wenn das Mapping eine ADO-Verbindung enthält und von generiertem C#- oder C++-Code oder von einem unter Windows laufenden MapForce Server ausgeführt wird (siehe Datenbankmappings in verschiedenen Ausführungsumgebungen) 166. Die Einstellungen **Datenquelle** und **Katalog** werden nicht vom Built-in-Ausführungsprozessor verwendet.

Datenquelle	Zeigt den Namen der ADO-Datenquelle an.
Katalog	Zeigt den Namen des ADO-Katalogs an.
Provider	Zeigt den gerade aktiven Provider für die Datenbankkomponente an.
Zus. Optionen	Zeigt zusätzliche Datenbankoptionen an.

Generierungseinstellungen

Die Generierungseinstellungen gelten für alle Codegenerierungsziele sowie für den Built-In-Ausführungsprozessor.

Schemanamen aus Tabellennamen entfernen	Damit können Sie Datenbankschemanamen aus generiertem Code entfernen, sodass aus Gründen zusätzlicher Flexibilität nur die Tabellennamen beibehalten werden. Beachten Sie, dass diese Option nur bei SQL Select-Anweisungen, die von MapForce generiert wurden, funktioniert. Benutzerdefinierte SQL-Anweisungen werden beim Erstellen von virtuellen Tabellen 368 nicht geändert. Diese Option wird nur aus Gründen der Rückwärtskompatibilität unterstützt und sollte vermieden werden. Um Datenbankobjektnamen als relativ zum Standarschema zu behandeln, verwenden Sie die unter Wechseln von Datenbanken und Schemas 263 beschriebene Methode.

Timeout für die Ausführung der Anweisung

Bei Verwendung einer Datenbank als Zielkomponente kann es je nach Server-Verfügbarkeit, Datenverkehr, längerer Ausführung usw. zu Zeitüberschreitungen kommen. Mit dieser Einstellung können Sie festlegen, wie lange der Client wartet, bis die Datenbankverbindung abgebrochen wird. Die Einstellung wird bei Datenbankabfragen sowie beim Generieren von C#-, Java- und C++-Code wirksam.

Timeout	Definiert, wie lang der Ausführungsprozessor auf eine Antwort von der Datenbank warten soll, bevor er die Ausführung der Datenbankanweisung abbricht. Die Standardeinstellung für das Ausführungs-Timeout ist 60 Sekunden.

Unendlich	Wenn diese Option aktiviert ist, gibt es kein Timeout für den Ausführungsprozessor.

Anmerkung: Bei SQLite-Datenbanken kann kein Timeout für die Ausführung von Anweisungen definiert werden.

Datenbanktransaktionsbehandlung

Transaktionen verwenden	Aktiviert die Verarbeitung von Transaktionen, wenn als Ziel eine Datenbank verwendet wird. Bei Auswahl dieser Option wird die Transaktionsverarbeitung für alle Tabellen der Datenbankkomponente aktiviert. Nähere Informationen finden Sie unter Behandlung von Datenbankausnahmen 429.

Ablaufverfolgungen

Ablaufverfolgungsebene	Anwendbar auf Datenbankzielkomponenten. Wenn die Ablaufverfolgung aktiviert ist, werden die vom Mapping an der Datenbank ausgeführten Aktionen in einer Ablaufverfolgungsdatei protokolliert. Sie können festlegen, ob alle Aktionen oder nur Fehler protokolliert werden sollen oder ob die Ablaufverfolgung gänzlich deaktiviert werden soll. Nähere Informationen dazu finden Sie unter Datenbankablaufverfolgung und Fehlerprotokollierung 443.
Ablaufverfolgungsdatei	Definiert die Datei, in die die Datenbankablaufverfolgungsinformationen während der Ausführung des Mappings geschrieben werden sollen. Die Ablaufverfolgung ist im XML-Format, siehe Struktur der Ablaufverfolgungsdatei 445. Dieser Pfad kann entweder absolut oder relativ sein und ist davon abhängig, ob das Kontrollkästchen **Alle Dateipfad relativ zur MFD-Datei speichern** aktiviert ist.

Alle Dateipfad relativ zur MFD-Datei speichern

Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld "Komponenteneinstellungen" angezeigten Dateipfade relativ zum Ordner der MapForce Design (.mfd)-Datei. Verwenden Sie relative Pfade, wenn Sie vorhaben, das Mapping mit MapForce Server auf einem anderen Betriebssystem auszuführen. Siehe auch Verwenden relativer Pfade in einer Komponente 85.

4.2.3 Mappen von Daten auf Datenbanken

Dieser Abschnitt enthält Anleitungen und Beispiele zum Übertragen von Daten aus jeder von MapForce unterstützten Mapping-Quelle (z.B. einer XML-Datei) in eine Zieldatenbank. Einen Überblick über die verfügbaren Optionen finden Sie in der folgenden Tabelle.

© 2016-2022 Altova GmbH

Aufgabe	Kapitel
Einfügen von Daten in eine Zieldatenbank auf Basis der vom Mapping gelieferten Daten...	• Einfügen von Daten in eine Tabelle 292
Festlegen, wie Primärschlüsselwerte erstellt werden...	• Einfügen von Daten in eine Tabelle 292 • Einfügen von Daten in mehrere verknüpfte Tabellen 295
Ausführen einer "vorläufigen" SQL-Anweisung bevor eine Tabelle durch das Mapping geändert wird (z.B. Löschen aller Datensätze in einer Tabelle oder eine benutzerdefinierte SQL-Anweisung)...	• Einfügen von Daten in mehrere verknüpfte Tabellen 295 • Einstellungen für Datenbankaktionen 336
Beibehalten der hierarchischen Beziehungen von Datensätzen in durch Sekundärschlüssel verknüpften Tabellen...	• Einfügen von Daten in mehrere verknüpfte Tabellen 295
Aktualisieren einer Tabelle auf Basis von Bedingungen...	• Aktualisieren einer Tabelle 300
Zusammenführen von Datensätzen in einer Datenbanktabelle (Aktualisierung einiger Datensätze sowie Einfügung einiger anderer Datensätze in dieselbe Tabelle) auf Basis einer Bedingung...	• Die Aktion "Aktualisieren, wenn...Rest einfügen" 303 • MERGE-Anweisungen 310
Bewahrung der Datenbankintegrität beim Aktualisieren von Tabellen, die mit anderen Tabellen über Sekundärschlüsselbeziehungen verknüpft sind...	• Optionen für Child-Tabellen bei Aktualisierung einer Parent-Tabelle 312
Definieren mehrerer Aktionen an derselben Tabelle (z.B. Löschen eines Datensatzes, wenn eine Bedingung erfüllt wird, andernfalls Einfügen eines neuen Datensatzes)...	• Die Aktion "Löschen, wenn..." 320 • Die Aktion "Ignorieren, wenn..." 324
Bewahrung der Datenbankintegrität bei fehlgeschlagener Mapping-Ausführung...	• Behandlung von Datenbankausnahmen 429
Einfügen mehrerer Datensätze in eine Datenbanktabelle mittels einer Bulk-Operation (Kombinieren mehrerer INSERT-Anweisungen in einer Abfrage)...	• Bulk-Einfügungen (MapForce Server) 329
Vermeidung unerwünschter Ergebnisse beim Mappen von Daten auf Datenbankzieltabellen, die Nullwerte enthalten...	• Behandlung von Nullwerten in Datenbankaktionen 333
Anzeige eines Beispiels zur Erstellung eines Mappings, mit dem eine Datenbank aktualisiert wird.	• Beispiel: Mappen von Daten von XML auf SQLite 339

4.2.3.1 Einfügen von Daten in eine Tabelle

Mit einem Mapping können Daten aus jeder beliebigen von MapForce unterstützten Quellkomponente (auch aus anderen Datenbanken) in eine Datenbanktabelle eingefügt werden. Sie können flexibel konfigurieren, wie der Primärschlüssel neu eingefügter Datensätze erstellt werden soll. So kann der Primärschlüssel etwa aus dem Mapping stammen, von der Datenbank generiert werden oder anhand vorhandener Schlüsselwerte in der Datenbanktabelle berechnet werden.

In diesem Beispiel wird gezeigt, wie Sie neue Datensätze aus einer XML-Datei in eine vorhandene Datenbanktabelle einfügen. Dabei wird auch festgelegt, wie der Primärschlüssel generiert werden soll. Im Beispiel werden die folgenden Dateien verwendet:

- **altova-cmpy.xml** - enthält die Quelldaten, die in die Datenbank eingefügt werden sollen
- **Altova_Hierarchical.xsd** - das Schema, anhand dessen die obige Instanzdatei validiert wird
- **altova.mdb** - die Zieldatenbank, die aktualisiert werden soll

Sie finden all diese Dateien im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\.** Der vollständige Pfad wurde in der Folge aus Gründen der Einfachheit weggelassen.

> Im Mapping in diesem Beispiel wird eine Beispieldatenbankdatei geändert. Es wird empfohlen, eine Sicherungskopie der Originaldatenbank zu erstellen und mit einer neuen Kopie zu arbeiten, bevor Sie die unten stehende Anleitung befolgen. Dadurch stellen Sie sicher, dass die Originaldaten nicht überschrieben werden und Sie dasselbe Ergebnis, wie unten beschrieben, erhalten. Nähere Informationen dazu finden Sie unter Ausführen von Mappings, die Datenbanken modifizieren 282.

Ziel des Mappings ist es, Firmen aus der Datei **Altova-cmpy.xml** als neue Datensätze in die Tabelle "Altova" der Datenbank **altova.mdb** einzufügen. Wenn Sie die XML-Quelldatei öffnen, werden Sie sehen, dass sie nur eine Firma namens "Microtech OrgChart" enthält. Im Mapping muss daher ein neuer Datensatz mit dem Namen "Microtech OrgChart" zur Tabelle "Altova" hinzugefügt werden. Außerdem muss dafür ein neuer Primärschlüssel generiert werden.

Gehen Sie dazu folgendermaßen vor:

Schritt 1: Fügen Sie die XML-Quellkomponente ein

- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur Datei **Altova_Hierarchical.xsd**. Wenn Sie aufgefordert werden, eine Instanzdatei anzugeben, navigieren Sie zur Datei **Altova-cmpy.xml**.

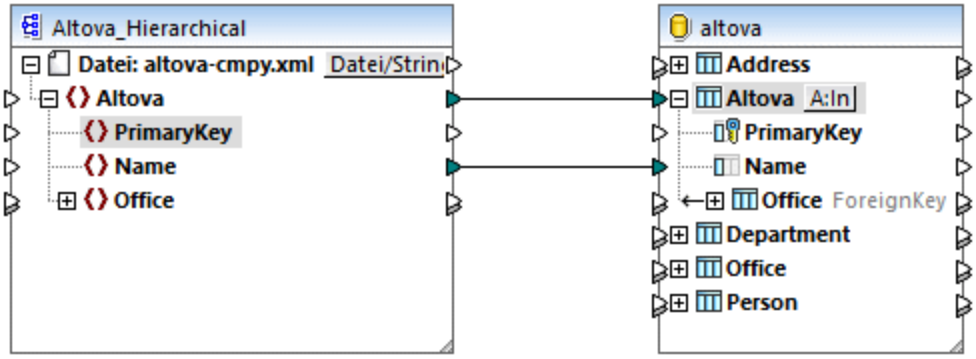
Schritt 2: Fügen Sie die Zieldatenbank ein

- Klicken Sie im Menü **Einfügen** auf **Datenbank** und befolgen Sie die Anweisungen des Assistenten, um eine Verbindung zu **altova.mdb** herzustellen (siehe Beispiel: Hinzufügen der Datenbank "altova.mdb" zum Mapping 257).

Schritt 3: Ziehen Sie die Verbindungen

- Ziehen Sie die Mapping-Verbindungen wie unten gezeigt.

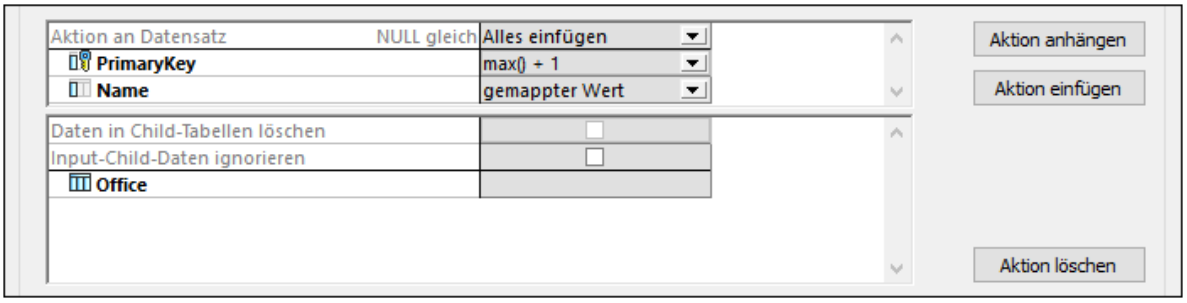
 © 2016-2022 Altova GmbH



Anmerkung: Wenn für untergeordnete Datenelemente automatisch unerwünschte Verbindungen gezogen werden, so ist die Option "Idente Sub-Einträge automatisch verbinden" aktiv. Falls dies der Fall ist, können Sie die letzte Aktion mit der Menüoption **Bearbeiten | Rückgängig** rückgängig machen. Um die Option zum automatischen Verbinden von Sub-Einträgen zu deaktivieren, klicken Sie auf die Menüoption **Verbindung | Idente Sub-Einträge automatisch verbinden**.

Schritt 4: Konfigurieren Sie die Einfügeaktion

1. Beachten Sie in der Zielkomponente die Schaltfläche **Aktion: Einfügen** (A:In). Diese Schaltfläche wird neben jeder Tabelle, die eine Verbindung vom Mapping aufweist, angezeigt (in diesem Fall die Tabelle "Altova"). Klicken Sie auf diese Schaltfläche, um die Einzelheiten der auszuführenden Datenbankaktion (in diesem Fall der Aktion "Einfügen") zu konfigurieren. Daraufhin wird das Dialogfeld "Datenbankaktionen" angezeigt.
2. Wählen Sie im Dialogfeld "Datenbankaktionen" unter **Alles einfügen** neben **PrimaryKey** die Option **max() + 1**.



Die Optionen in dieser Liste haben die folgenden Bedeutung:

Option	Beschreibung
gemappter Wert	Damit können Quelldaten direkt auf das Datenbankfeld gemappt werden. Dies ist die Standardeinstellung für alle Datenbankfelder. Sie können auch eine gespeicherte Prozedur verwenden, um durch Definition einer Beziehung einen Schlüsselwert bereitzustellen. Siehe dazu Generieren von Schlüsseln mit Hilfe von gespeicherten Prozeduren 424.

max() + 1	Generiert die Schlüsselwerte anhand der vorhandenen Schlüssel in der Datenbank. Wenn die Tabelle z.B. drei Datensätze mit den Primärschlüsseln 1, 2 und 3 hat, so ist **max() + 1** gleich 4. In diesem Beispiel hat die Tabelle "Altova" nur einen Datensatz mit dem Primärschlüssel 1, daher ist **max() + 1** gleich 2. Dies ist der erwartete Wert des neuen Primärschlüssels.
DB-generiert	Zur Generierung der Schlüsselwerte wird von der Datenbank die **Identity-Funktion** verwendet.

Die Option **gemappter Wert** neben "Name" bedeutet, dass diese Spalte den Wert direkt aus dem Mapping erhält. Informationen zu anderen Optionen im Dialogfeld "Datenbankaktionen" finden Sie unter Einstellungen für Datenbankaktionen 336.

Schritt 5: Vorschau auf das Mapping und Aktualisieren der Datenbank

Klicken Sie auf das Register **Ausgabe,** um eine Vorschau auf das Mapping zu sehen. Es wird ein SQL-Skript generiert, das Aktionen enthält, die an der Datenbank ausgeführt werden sollen. Das Skript hat die Datenbank noch nicht geändert; dies dient nur zur Vorschau.

```
1   /*
2   The following SQL statements are only for preview and may not be executed in another SQL query tool!
3   To execute these statements use function "Run SQL-script" from menu "Output".
4   Connect to database using the following connection-string:
5   Data Source=\\viepfs06\Documentation\Public\ExampleFiles\EN\MapForce\DB Insert\altova.mdb;Provider=Microsoft.Jet.OLEDB.4.0
6   */
7
8   SELECT IIF(MAX([Altova].[PrimaryKey]) IS NULL,0,MAX([Altova].[PrimaryKey]))+1 AS [PrimaryKey] FROM [Altova]
9   -->>> %PrimaryKey1%
10
11  INSERT INTO [Altova] ([Name], [PrimaryKey]) VALUES ('Microtech OrgChart', '%PrimaryKey1%')
12
```

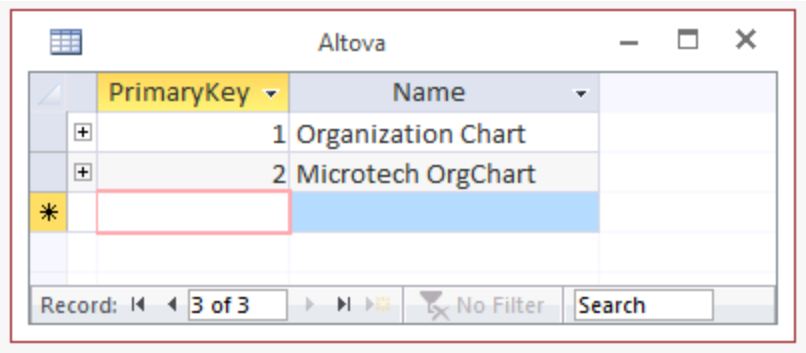
So führen Sie das Skript an der Datenbank aus:

- Klicken Sie im Menü **Ausgabe** auf **SQL-Script ausführen.**

Anmerkung: Das Ausführen des SQL-Skripts direkt von MapForce aus ist nur eine der Methoden, um eine Datenbank zu aktualisieren. Siehe auch Ausführen von Mappings, die Datenbanken modifizieren 282.

Um das Ergebnis zu sehen, öffnen Sie die Datenbank **altova.mdb** in DatabaseSpy oder Access. Beachten Sie, dass ein neuer Datensatz "Microtech OrgChart" mit dem Primärschlüssel 2 zur Tabelle "Altova" hinzugefügt wurde. Die Daten für diesen Datensatz stammen aus der XML-Input-Instanz.

© 2016-2022 Altova GmbH



Sie haben nun ein Mapping erstellt, das Daten in eine Datenbanktabelle einfügt. Ein Beispiel für ein Mapping, das Daten sowohl in die aktuelle Tabelle als auch in eine davon abhängige Child-Tabelle einfügt finden Sie unter Einfügen von Daten in mehrere verknüpfte Tabellen 295.

4.2.3.2 Einfügen von Daten in mehrere verknüpfte Tabellen

Bei einer Datenbanktabelle kann es sich um eine Parent-Tabelle, also eine übergeordnete Tabelle handeln. D.h. diese Tabelle kann über Sekundärschlüsselbeziehungen von anderen Tabellen in der Datenbank referenziert werden. Sie können das Mapping in solchen Szenarien so konfigurieren, dass nicht nur in die Parent-Tabelle Datensätze eingefügt werden, sondern auch in davon abhängige Child-Tabellen. Wenn z.B. ein neuer Firmendatensatz in eine Datenbanktabelle eingefügt wird, können auch Datensätze für Niederlassungen, die mit dieser Firma verknüpft sind, sowie deren untergeordnete Abteilungen (departments), Mitarbeiter (persons) usw. eingefügt werden.

In diesem Beispiel wird gezeigt, wie Sie Daten in mehrere Tabellen einfügen und dabei Datenbankbeziehungen beibehalten. Es handelt sich bei diesem Beispiel um eine etwas komplexere Version des vorherigen Beispiels Einfügen von Daten in eine Tabelle 292. Im Beispiel-Mapping werden die folgenden Dateien verwendet:

- **Altova_Hierarchical.mfd -** die eigentliche Mapping-Datei
- **Altova_Hierarchical.xml** - enthält die Quelldaten, die in die Datenbank eingefügt werden sollen
- **Altova_Hierarchical.xsd** - das Schema, anhand dessen die obige Instanzdatei validiert wird
- **AltovaTarget.mdb** - die Zieldatenbank, die aktualisiert werden soll

Sie finden all diese Dateien im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\.** Der vollständige Pfad wurde in der Folge aus Gründen der Einfachheit weggelassen.

> Im Mapping in diesem Beispiel wird eine Beispieldatenbankdatei geändert. Es wird empfohlen, eine Sicherungskopie der Originaldatenbank zu erstellen und mit einer neuen Kopie zu arbeiten, bevor Sie die unten stehende Anleitung befolgen. Dadurch stellen Sie sicher, dass die Originaldaten nicht überschrieben werden und Sie dasselbe Ergebnis, wie unten beschrieben, erhalten. Nähere Informationen dazu finden Sie unter Ausführen von Mappings, die Datenbanken modifizieren 282.

Ziel des Mappings ist es, Daten in der Zieldatenbank (**AltovaTarget.mdb**) durch Daten aus der XML-Quelldatei zu ersetzen. Die Struktur der XML-Datei entspricht ungefähr der hierarchischen Struktur der Tabellen in der Datenbank. Es handelt sich um ein Organigramm, des folgendermaßen strukturiert ist: Das oberste Element ist eine Firma, die zwei Niederlassungen (offices) hat. Jede Niederlassung enthält Abteilungen (departments) und jede Abteilung enthält Mitarbeiter (persons). Dieselbe Hierarchie finden Sie in der Zieldatenbank

AltovaTarget.mdb, in der die Tabelle "Altova" der Firma entspricht. Diese Tabelle ist über Sekundärschlüsselbeziehungen mit Datensätzen in der Tabelle "Office" verknüpft. Außerdem ist "Office" mit "Department" verknüpft und "Department" mit "Person". Um ein Beziehungsdiagramm der Datenbank **AltovaTarget.mdb** zu sehen, öffnen Sie sie in der Ansicht "Beziehungen" von Access (siehe auch Behandlung von Datenbankbeziehungen 266).

Gehen Sie dazu folgendermaßen vor:

Schritt 1: Fügen Sie die XML-Quellkomponente ein

- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur Datei **Altova_Hierarchical.xsd**. Wenn Sie aufgefordert werden, eine Instanzdatei anzugeben, navigieren Sie zur Datei **Altova_Hierarchical.xml**.

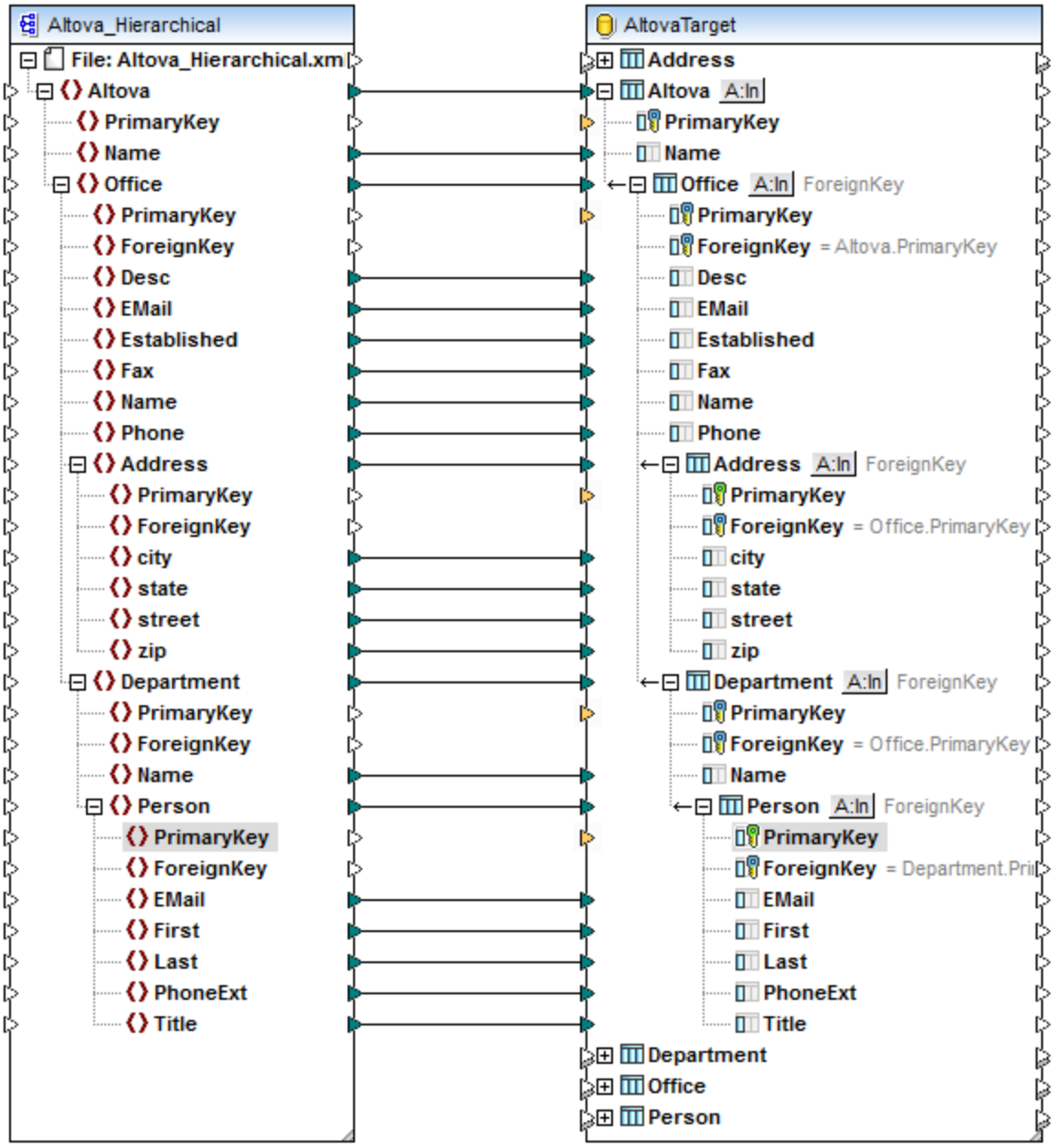
Schritt 2: Fügen Sie die Zieldatenbank ein

- Klicken Sie im Menü **Einfügen** auf **Datenbank** und befolgen Sie die Anweisungen des Assistenten, um eine Verbindung zu **AltovaTarget.mdb** herzustellen. die Anleitung zum Herstellen einer Verbindung zu dieser Datenbank ist dieselbe wie die Anleitung zum Verbinden mit altova.mdb (siehe Beispiel: Hinzufügen der Datenbank "altova.mdb" zum Mapping 257).

Schritt 3: Ziehen Sie die Verbindungen

- Ziehen Sie die Mapping-Verbindungen wie unten gezeigt. Beachten Sie, dass die Primär- und Sekundärschlüssel nicht gemappt sind; diese werden wie unten gezeigt on-the-fly generiert.

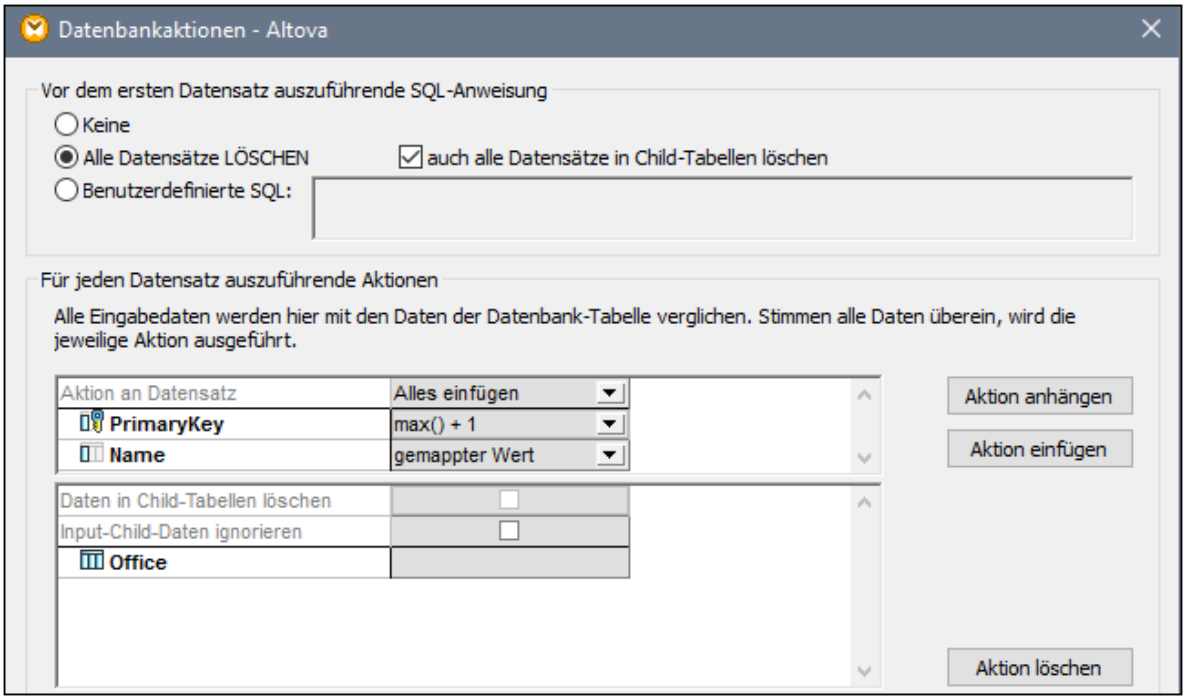
© 2016-2022 Altova GmbH



Anmerkung: Wenn für untergeordnete Datenelemente automatisch unerwünschte Verbindungen gezogen werden, so ist die Option "Idente Sub-Einträge automatisch verbinden" aktiv. Falls dies der Fall ist, können Sie die letzte Aktion mit der Menüoption **Bearbeiten | Rückgängig** rückgängig machen. Um die Option zum automatischen Verbinden von Sub-Einträgen zu deaktivieren, klicken Sie auf die Menüoption **Verbindung | Idente Sub-Einträge automatisch verbinden**.

Schritt 4: Konfigurieren Sie die Einfügeaktionen

1. Klicken Sie auf die Schaltfläche **Aktion: Einfügen** (A:In) neben der Tabelle "Altova" und konfigurieren Sie die Einstellung **max() + 1** des Primärschlüssels wie unten gezeigt. Die Einstellung wurde im vorherigen Beispiel Einfügen von Daten in eine Tabelle 292 näher erläutert.



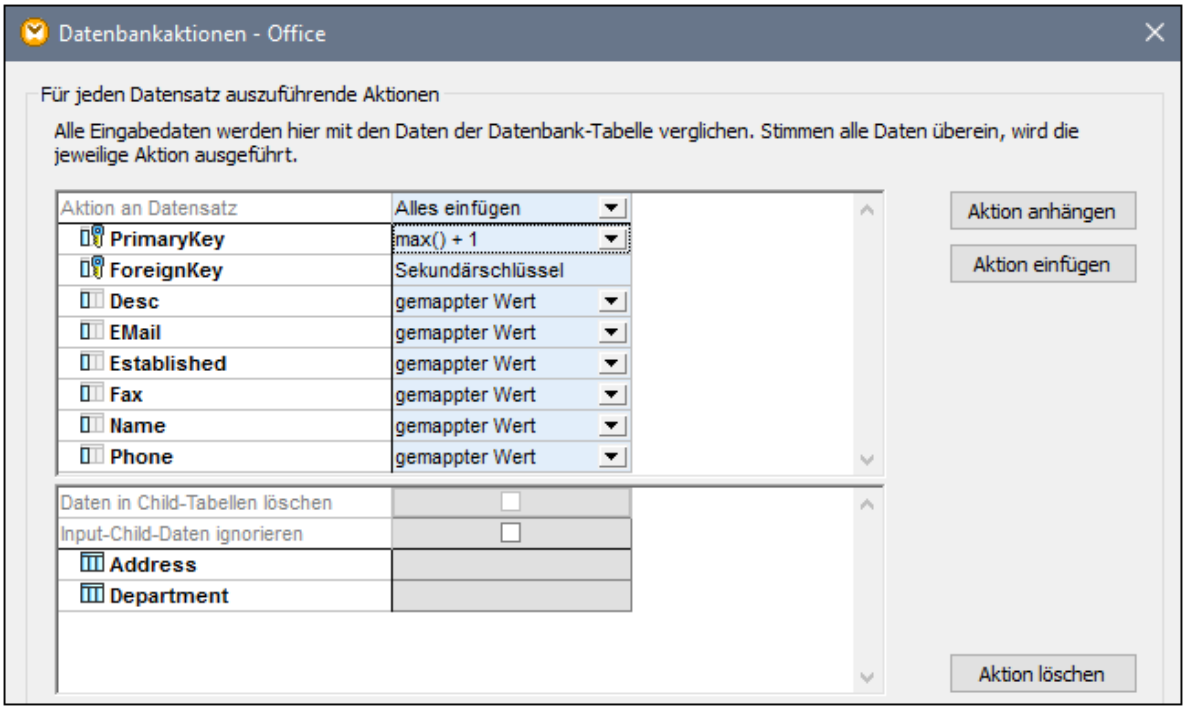
Beachten Sie außerdem, dass die Option **Alle Datensätze löschen** aktiviert ist. Dadurch werden plangemäß alle vorhandenen Datensätze aus der Tabelle gelöscht, bevor neue eingegeben werden. Wenn Sie diese Option deaktivieren, werden bei jeder Ausführung des Mappings neue Datensätze (mit einem neuen Primärschlüssel) zusätzlich zu den bereits vorhandenen zur Datenbank hinzugefügt, was in diesem Fall nicht erwünscht ist.

In diesem Beispiel ist auch die Option **auch alle Datensätze in Child-Tabellen löschen** aktiviert. Damit wird sichergestellt, dass nicht nur die Datensätze aus der Tabelle "Altova" gelöscht werden, sondern auch alle Datensätze in Tabellen, die über eine Sekundärschlüsselbeziehung mit der Tabelle "Altova" verknüpft sind. Wenn die Child-Tabellen selbst Child-Tabellen enthalten, so werden auch diese gelöscht. Dies wird über die gesamte Hierarchie der abhängigen Tabellen bis zur untersten Tabelle fortgesetzt. Würden Sie versuchen, nur Datensätze aus der Root-Tabelle "Altova" zu löschen, würde die Datenbankintegrität dadurch verletzt und die Mapping-Ausführung würde fehlschlagen.

Erläuterungen zu den anderen Optionen im Dialogfeld "Datenbankaktionen" finden Sie unter Einstellungen für Datenbankaktionen 336.

2. Klicken Sie auf **OK,** um das Dialogfeld zu schließen. Beachten Sie, dass im Mapping-Bereich die Schaltfläche nun als DEL, A:In angezeigt wird. Damit wird angezeigt, dass eine "Delete"-Anweisung vor der Einfügeaktion konfiguriert wurde.
3. Klicken Sie auf die Schaltfläche A:In neben der Tabelle "Office" und konfigurieren Sie die Einstellung **max() + 1** des Primärschlüssels.

© 2016-2022 Altova GmbH



4. Führen Sie für jede untergeordnete Tabelle von "Office", nämlich "Address", "Department" und "Person" Schritt 3 durch. Vergewissern Sie sich, dass all diese Tabellen unmittelbar untergeordnete Tabellen der Root-Tabelle "Altova" sind. Eine Erklärung, was eine Root-Tabelle ist und wofür diese benötigt wird, finden Sie unter Behandlung von Datenbankbeziehungen 266.

Schritt 5: Vorschau auf das Mapping und Aktualisieren der Datenbank

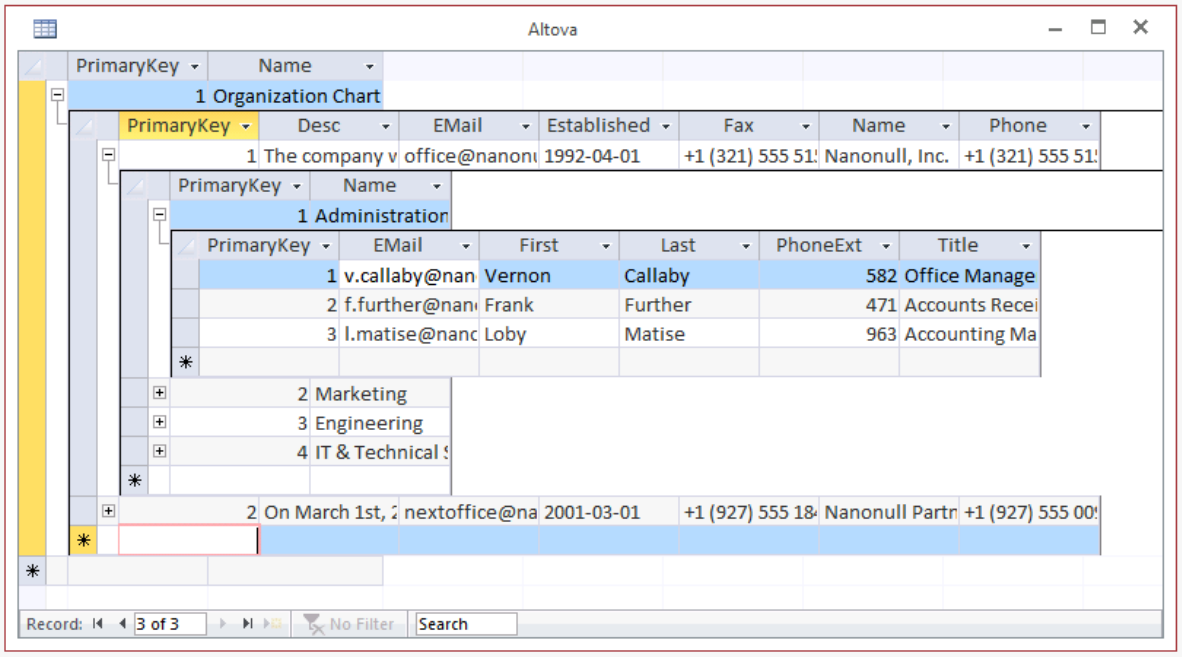
Klicken Sie auf das Register **Ausgabe,** um eine Vorschau auf das Mapping zu sehen. Es wird ein SQL-Skript generiert, das Aktionen enthält, die an der Datenbank ausgeführt werden sollen. Das Skript hat die Datenbank noch nicht geändert; dies dient nur zur Vorschau.

So führen Sie das Skript an der Datenbank aus:

- Klicken Sie im Menü **Ausgabe** auf **SQL-Script ausführen.**

Anmerkung: Das Ausführen des SQL-Skripts direkt von MapForce aus ist nur eine der Methoden, um eine Datenbank zu aktualisieren. Siehe auch Ausführen von Mappings, die Datenbanken modifizieren 282.

Um das Ergebnis zu sehen, öffnen Sie die Tabelle "Altova" in Microsoft Access. Sie sehen darin, wie die Beziehungen aus der XML-Datei, von der Tabelle "Altova" bis hinunter zur Tabelle "Person" nun in der Datenbank übernommen wurden.



Sie haben nun ein Mapping erstellt, das Daten in mehrere Datenbanken einfügt, wobei intakte Tabellenbeziehungen erhalten geblieben sind.

4.2.3.3 Aktualisieren einer Tabelle

In diesem Beispiel wird gezeigt, wie Sie Daten einer vorhandenen Datenbanktabelle mit Daten aus einer XML-Quelldatei aktualisieren. Im Beispiel werden die folgenden Dateien verwendet:

- **altova-cmpy.xml** - enthält die Quelldaten, die in die Datenbank eingefügt werden sollen
- **Altova_Hierarchical.xsd** - das Schema, anhand dessen die obige Instanzdatei validiert wird
- **Altova.mdb** - die Zieldatenbank, die aktualisiert werden soll

Sie finden all diese Dateien im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\.** Der vollständige Pfad wurde in der Folge aus Gründen der Einfachheit weggelassen.

Im Mapping in diesem Beispiel wird eine Beispieldatenbankdatei geändert. Es wird empfohlen, eine Sicherungskopie der Originaldatenbank zu erstellen und mit einer neuen Kopie zu arbeiten, bevor Sie die unten stehende Anleitung befolgen. Dadurch stellen Sie sicher, dass die Originaldaten nicht überschrieben werden und Sie dasselbe Ergebnis, wie unten beschrieben, erhalten. Nähere Informationen dazu finden Sie unter Ausführen von Mappings, die Datenbanken modifizieren 282.

Ziel des Mappings ist es, alle Datensätze in der Tabelle "Person" mit Instanzen von "Person" aus dem XML-Dokument zu aktualisieren. Jede Person in der XML-Datei hat ein `PrimaryKey`-Child-Element. Jede Person in der Tabelle "Person" hat eine `PrimaryKey`-Spalte. Nur die Datensätze, in denen der `PrimaryKey` einer Person in der XML-Datei mit dem `PrimaryKey` einer Person in der Datenbank übereinstimmt, müssen aktualisiert werden.

Gehen Sie dazu folgendermaßen vor:

 © 2016-2022 Altova GmbH

Schritt 1: Fügen Sie die XML-Quellkomponente ein

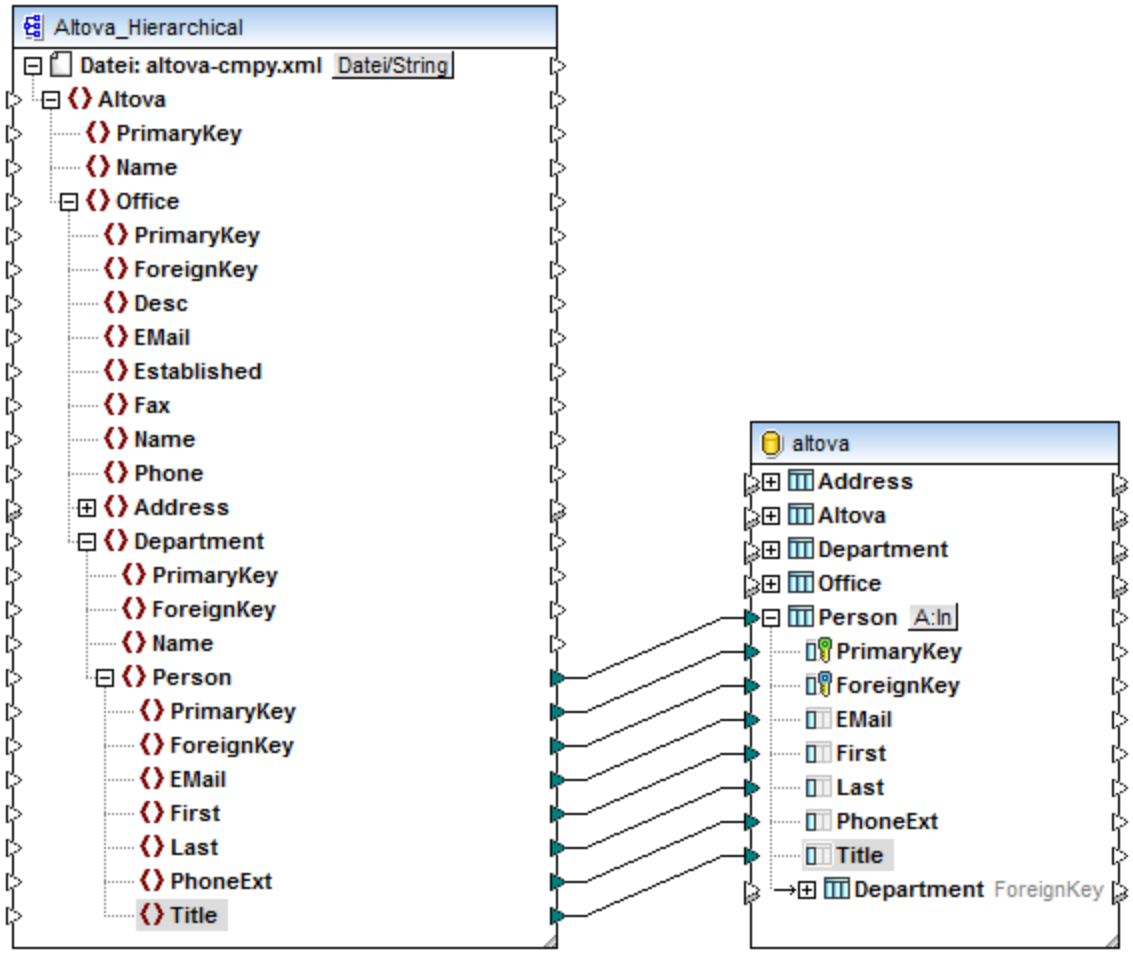
- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur Datei **Altova_Hierarchical.xsd**. Wenn Sie aufgefordert werden, eine Instanzdatei anzugeben, navigieren Sie zur Datei **altova-cmpy.xml**.

Schritt 2: Fügen Sie die Zieldatenbank ein

- Klicken Sie im Menü **Einfügen** auf **Datenbank** und befolgen Sie die Anweisungen des Assistenten, um eine Verbindung zu **Altova.mdb** herzustellen (siehe Beispiel: Hinzufügen der Datenbank "altova.mdb" zum Mapping 257).

Schritt 3: Ziehen Sie die Verbindungen

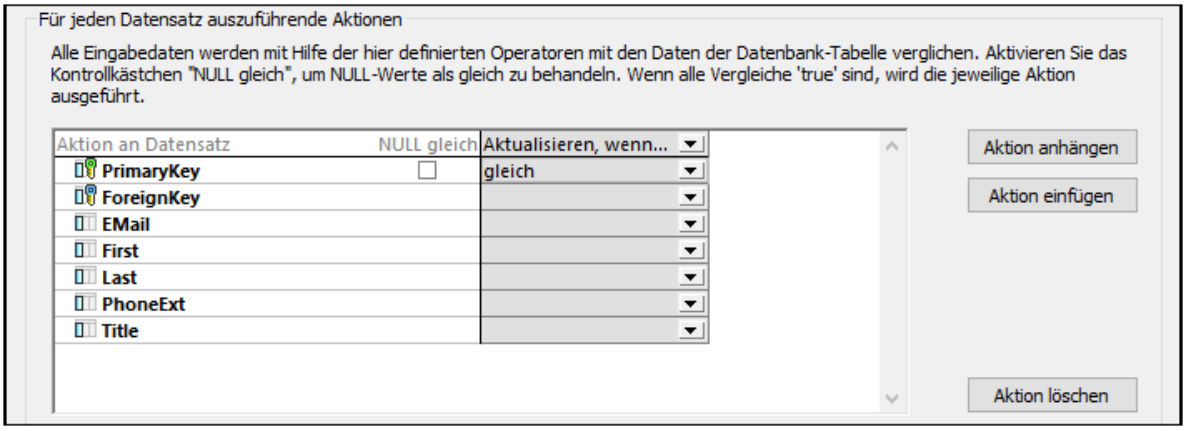
- Ziehen Sie die Mapping-Verbindungen wie unten gezeigt.



Schritt 4: Konfigurieren Sie die Aktualisierungsaktion

1. Klicken Sie auf die Schaltfläche **Aktion: Einfügen** (A:In) neben der Tabelle "Person".

2. Wählen Sie neben **Aktion an Datensatz** den Eintrag **Aktualisieren, wenn...** . Dadurch wird die Datenbankaktion in eine auf einer Bedingung basierende Aktualisierungsaktion geändert, d.h. der aktuelle Datensatz wird nur aktualisiert, wenn eine Bedingung erfüllt wird (siehe nächster Schritt).
3. Wählen Sie neben **PrimaryKey** den Wert **gleich** aus, wie unten gezeigt. Damit wird die Aktualisierungsbedingung definiert, d.h. der Datenbankdatensatz wird nur aktualisiert, wenn sein **PrimaryKey**-Wert gleich dem **PrimaryKey**-Wert aus dem Mapping ist.



In diesem Beispiel wird der Ist-gleich-Operator auf das Feld **PrimaryKey** angewendet, was beim Aktualisieren von Datenbanken ein häufiges Szenario ist. Beachten Sie, dass auch an anderen Feldern, die nicht unbedingt Primärschlüssel sein müssen, Bedingungen definiert werden können. Wenn Sie z.B. **gleich** neben den Feldern **First** und **Last** auswählen, würden nur die Datensätze aktualisiert, in denen der Vor- und Nachname (first und last name) gleich dem Vor- und Nachnamen in der XML-Quelldatei ist.

Aktivieren Sie optional das Kontrollkästchen **NULL gleich**. In diesem Beispiel spielt das Kontrollkästchen keine Rolle, da weder das Feld **PrimaryKey** in der XML-Quelldatei noch das Feld **PrimaryKey** in der Datenbank Nullwerte enthält. Wenn Ihre Daten jedoch Nullwerte enthalten und solche Nullwerte als identisch behandelt werden sollen, sollten Sie dieses Kontrollkästchen aktivieren, da Sie sonst unerwünschte Ergebnisse erhalten könnten. Nähere Informationen dazu finden Sie unter Behandlung von Nullwerten in Datenbankaktionen 333.

4. Klicken Sie auf **OK,** um das Dialogfeld zu schließen. Beachten Sie, dass im Mapping anstelle der Schaltfläche **Aktion: Einfügen** A:In nun eine Schaltfläche **Aktion:Aktualisieren** (A:Up) angezeigt wird. Dies zeigt an, dass für diese Tabelle eine Aktualisierungsaktion konfiguriert wurde.

Schritt 5: Vorschau auf das Mapping und Aktualisieren der Datenbank

Klicken Sie auf das Register **Ausgabe,** um eine Vorschau auf das Mapping zu sehen. Es wird ein SQL-Skript generiert, das Aktionen enthält, die an der Datenbank ausgeführt werden sollen. Das Skript hat die Datenbank noch nicht geändert; dies dient nur zur Vorschau.

© 2016-2022 Altova GmbH

```
1   /*
2   The following SQL statements were executed during "Generate output" function.
3   Every single result is written right to the "-->>>" string.
4   These statements are only for preview and may not be executed in another SQL query tool!
5   The database was connected using the following connection-string:
6   Data Source=\\viepfs06\Documentation\Public\ExampleFiles\EN\MapForce\DB Update
    if\altova.mdb;Provider=Microsoft.Jet.OLEDB.4.0
7   */
8
9   UPDATE [Person] SET [ForeignKey] = 1, [EMail] = 'A.Aldrich@microtech.com', [First] = 'Albert', [Last] = 'Aldrich',
    [PhoneExt] = 582, [Title] = 'Manager' WHERE ([Person].[PrimaryKey] = 1)
10  -->>> OK. 1 row(s).
11
12  UPDATE [Person] SET [ForeignKey] = 1, [EMail] = 'b.bander@microtech.com', [First] = 'Bert', [Last] = 'Bander',
    [PhoneExt] = 471, [Title] = 'Accounts Receivable' WHERE ([Person].[PrimaryKey] = 2)
13  -->>> OK. 1 row(s).
14
15  UPDATE [Person] SET [ForeignKey] = 1, [EMail] = 'c.clovis@microtech.com', [First] = 'Clive', [Last] = 'Clovis',
    [PhoneExt] = 963, [Title] = 'Accounting Manager' WHERE ([Person].[PrimaryKey] = 3)
16  -->>> OK. 1 row(s).
17
18  UPDATE [Person] SET [ForeignKey] = 2, [EMail] = 'd.Durnell@microtech.com', [First] = 'Dave', [Last] = 'Durnell',
    [PhoneExt] = 621, [Title] = 'Marketing Manager Europe' WHERE ([Person].[PrimaryKey] = 4)
19  -->>> OK. 1 row(s).
```

So führen Sie das Skript an der Datenbank aus:

- Klicken Sie im Menü **Ausgabe** auf **SQL-Script ausführen.**

Anmerkung: Das Ausführen des SQL-Skripts direkt von MapForce aus ist nur eine der Methoden, um eine Datenbank zu aktualisieren. Siehe auch Ausführen von Mappings, die Datenbanken modifizieren 282.

4.2.3.4 Die Aktion "Aktualisieren, wenn...Rest einfügen"

Manchmal müssen nicht nur vorhandene Datensätze aktualisiert, sondern auch neuen Datensätze in dieselbe Datenbanktabelle eingefügt werden. Zu diesem Zweck steht in MapForce die Aktion "Aktualisieren, wenn...Rest einfügen" zur Verfügung. Dabei geschieht Folgendes:

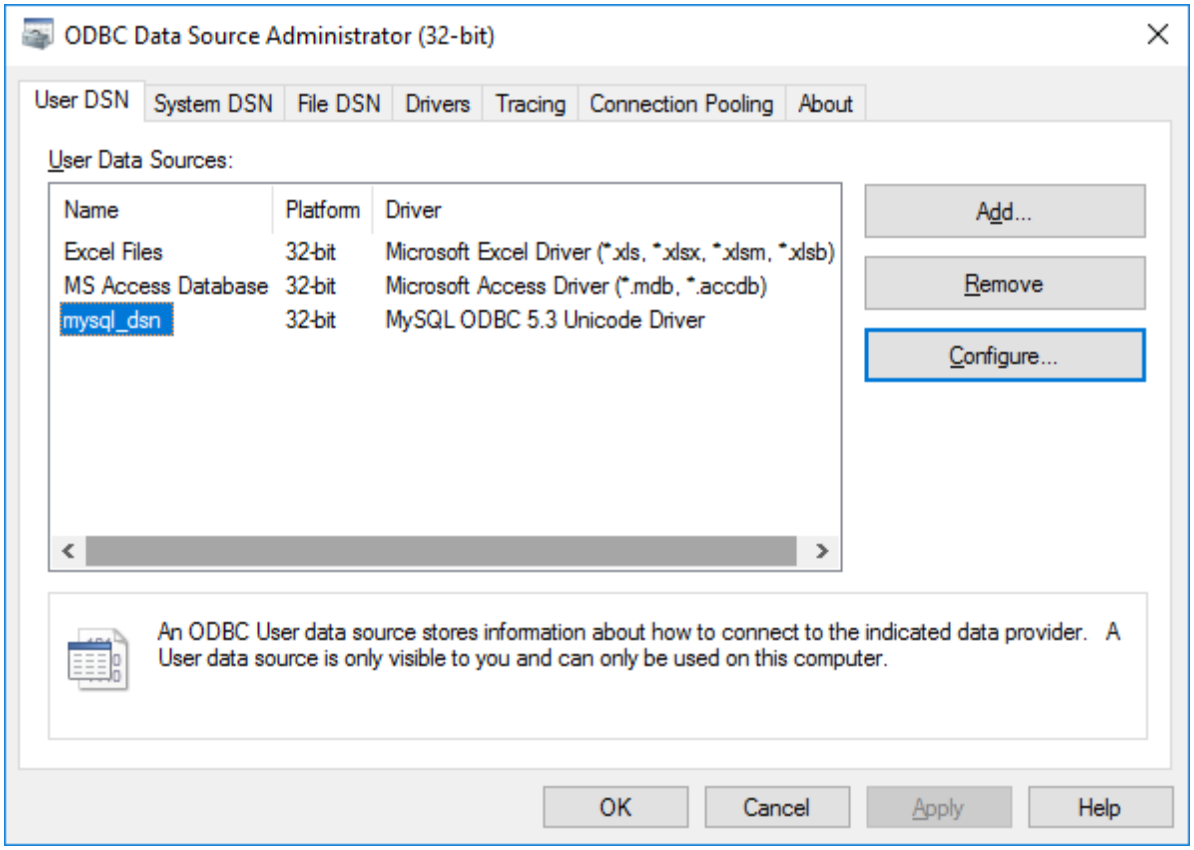
- Wenn die Bedingung **Aktualisieren, wenn** zutrifft, so wird der vorhandene Datenbankdatensatz mit den Daten aus dem Mapping aktualisiert.
- Wenn die Bedingung **Aktualisieren, wenn** nicht zutrifft und eine Bedingung **Rest einfügen** vorhanden ist, so wird ein neuer Datensatz eingefügt.
- Wenn in der Datenbank Datensätze vorhanden sind, zu denen es in der Quelldatei keinen entsprechenden Datensatz gibt, so bleiben diese Datensätze unverändert.

Anmerkung zu MySQL / MariaDB ODBC

Wenn es sich bei der Zieldatenbank um eine über ODBC verbundene MySQL- oder MariaDB-Datenbank handelt, muss auf dem Register **Cursor/Results** von MySQL ODBC Connector die Option **Return matched rows instead of affected rows** aktiviert sein. Alternativ dazu können Sie bei manueller Eingabe des Strings über den Datenbankverbindungsassistenten die Option `Option=2` zum Connection String hinzufügen, z.B: `Dsn=mydsn;Option=2;`

So aktivieren Sie diese Option über den MySQL ODBC Connector:

1. Drücken Sie die **Windows**-Taste und geben Sie "ODBC" ein.
2. Starten Sie den ODBC Data Sources Administrator (je nach Plattform des installierten MySQL ODBC Connectors entweder 32-Bit oder 64-Bit).
3. Klicken Sie auf den im MapForce Mapping verwendeten Data Source Name (DSN) und anschließend auf **Configure**.



4. Klicken Sie auf **Details >>** um zu den erweiterten Optionen zu gelangen.
5. Klicken Sie auf das Register **Cursors/Results** und aktivieren Sie anschließend das Kontrollkästchen **Return matched rows instead of affected rows**.

Beispiel

In diesem Beispiel wird gezeigt, wie Sie Daten aus einer XML-Quelldatei in einer Datenbanktabelle zusammenführen (Indem Daten sowohl aktualisiert als auch eingefügt werden). Im Beispiel werden die folgenden Dateien verwendet:

- **altova-cmpy-extra.xml** - enthält die Quelldaten, die in die Datenbank eingefügt werden sollen
- **Altova_Hierarchical.xsd** - das Schema, anhand dessen die obige Instanzdatei validiert wird
- **Altova.mdb** - die Zieldatenbank, die aktualisiert werden soll

Sie finden all diese Dateien im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\.** Der vollständige Pfad wurde in der Folge aus Gründen der Einfachheit weggelassen.

© 2016-2022 Altova GmbH

Im Mapping in diesem Beispiel wird eine Beispieldatenbankdatei geändert. Es wird empfohlen, eine Sicherungskopie der Originaldatenbank zu erstellen und mit einer neuen Kopie zu arbeiten, bevor Sie die unten stehende Anleitung befolgen. Dadurch stellen Sie sicher, dass die Originaldaten nicht überschrieben werden und Sie dasselbe Ergebnis, wie unten beschrieben, erhalten. Nähere Informationen dazu finden Sie unter Ausführen von Mappings, die Datenbanken modifizieren 282.

Ziel des Mappings ist es, alle Datensätze aus einem XML-Quelldokument in einer Zieltabelle namens "Person" zusammenzuführen. Dabei muss das Mapping für jeden Datensatz in der XML-Quelldatei die folgenden Aktionen ausführen:

- Wenn der Primärschlüssel `PrimaryKey` der Person in der XML-Datei mit dem Primärschlüssel `PrimaryKey` der Person in der Datenbank übereinstimmt, so muss der Datensatz aktualisiert werden.
- Vorhandene Datensätze in der Tabelle "Person", für die die obige Bedingung nicht zutrifft, dürfen nicht verändert werden.
- Wenn es für den `PrimaryKey` der Person in der XML-Datei in der Datenbankzieltabelle keine Entsprechung gibt, so wird ein neuer Datensatz zur Datenbanktabelle hinzugefügt.

Gehen Sie dazu folgendermaßen vor:

Schritt 1: Fügen Sie die XML-Quellkomponente ein

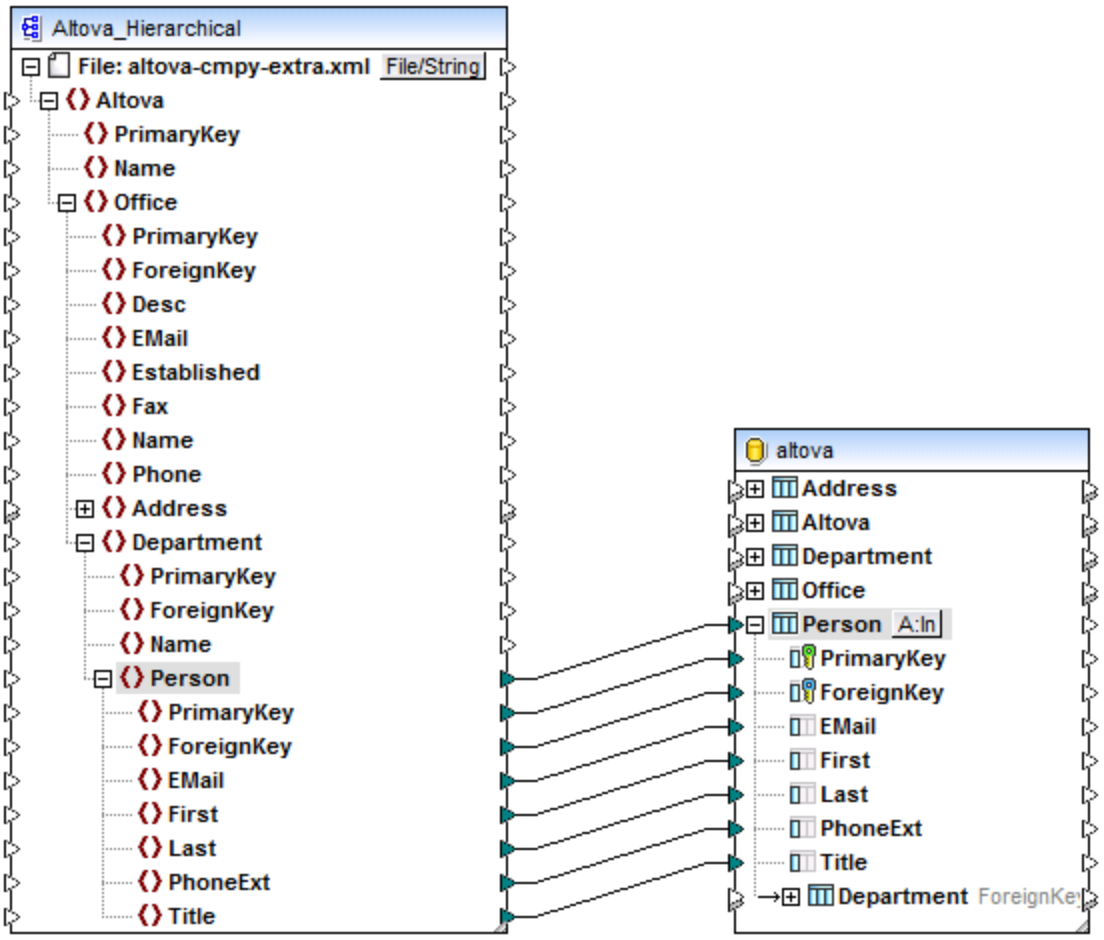
- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur Datei **Altova_Hierarchical.xsd**. Wenn Sie aufgefordert werden, eine Instanzdatei anzugeben, navigieren Sie zur Datei **altova-cmpy-extra.xml**.

Schritt 2: Fügen Sie die Zieldatenbank ein

- Klicken Sie im Menü **Einfügen** auf **Datenbank** und befolgen Sie die Anweisungen des Assistenten, um eine Verbindung zu **Altova.mdb** herzustellen (siehe Beispiel: Hinzufügen der Datenbank "altova.mdb" zum Mapping 257).

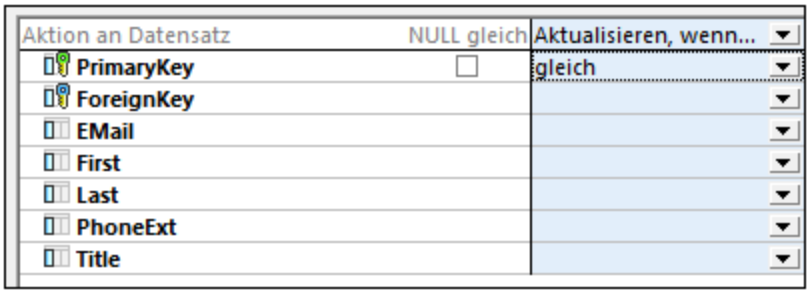
Schritt 3: Ziehen Sie die Verbindungen

- Ziehen Sie die Mapping-Verbindungen wie unten gezeigt.



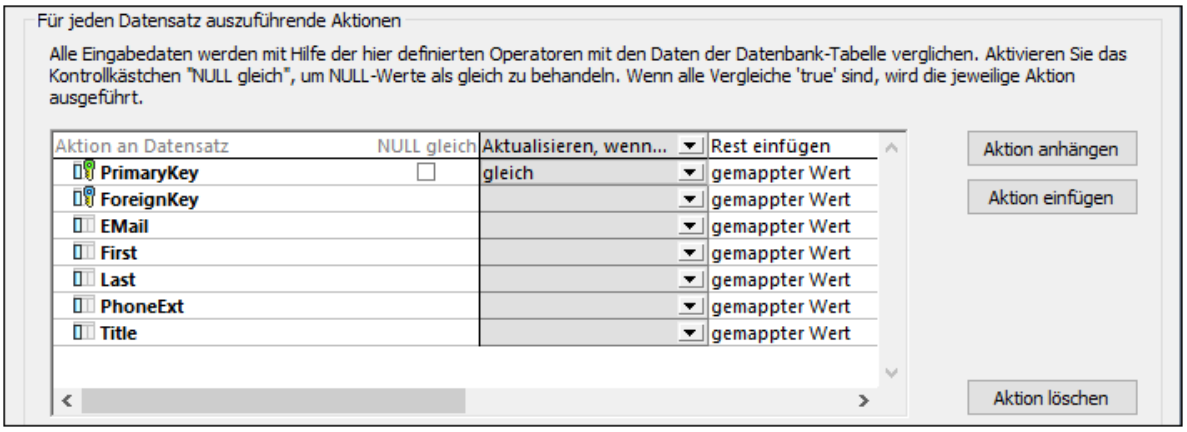
Schritt 4: Konfigurieren Sie die "Aktualisieren,wenn... Rest einfügen"-Aktionen

1. Klicken Sie auf die Schaltfläche **Aktion: Einfügen** (A:In) neben der Tabelle "Person".
2. Wählen Sie neben **Aktion an Datensatz** den Eintrag **Aktualisieren, wenn...** . Dadurch wird die Datenbankaktion in eine auf einer Bedingung basierende Aktualisierungsaktion geändert, d.h. der aktuelle Datensatz wird nur aktualisiert, wenn eine Bedingung erfüllt wird (siehe nächster Schritt).
3. Wählen Sie neben **PrimaryKey** den Wert **gleich** aus, wie unten gezeigt. Damit wird die Aktualisierungsbedingung definiert, d.h. der Datenbankdatensatz wird nur aktualisiert, wenn sein **PrimaryKey**-Wert gleich dem **PrimaryKey**-Wert aus dem Mapping ist.



 © 2016-2022 Altova GmbH

4. Aktivieren Sie optional das Kontrollkästchen **NULL gleich**. In diesem Beispiel spielt das Kontrollkästchen keine Rolle, da weder das Feld **PrimaryKey** in der XML-Quelldatei noch das Feld **PrimaryKey** in der Datenbank Nullwerte enthält. Wenn Ihre Daten jedoch Nullwerte enthalten und solche Nullwerte als identisch behandelt werden sollen, sollten Sie dieses Kontrollkästchen aktivieren, da Sie sonst unerwünschte Ergebnisse erhalten könnten. Nähere Informationen dazu finden Sie unter Behandlung von Nullwerten in Datenbankaktionen 333.
5. Klicken Sie auf **Aktion anhängen**. Daraufhin wird rechts von der vorhandenen **Aktualisieren, wenn**-Aktion eine neue Aktion hinzugefügt. Konfigurieren Sie die neue Aktion als **Rest einfügen**:



In der Abbildung oben wurden die Datenbankaktionen den Zielen des Mappings entsprechend konfiguriert, d.h. der Datensatz wird aktualisiert, wenn die **Aktualisieren, wenn..**-Bedingung zutrifft. Andernfalls wird der Datensatz eingefügt. Mit der Option "gemappter Wert" wird festgelegt, dass alle Felder des Datensatzes anhand der Werte aus dem Mapping befüllt werden.

Es können auch mehr als zwei Aktionen an derselben Datenbanktabelle definiert werden (in diesem Beispiel ist dies jedoch nicht notwendig). Die Aktionen werden zur Mapping-Laufzeit von links nach rechts ausgeführt. Die letzte **Einfügeaktion** wird als endgültig betrachtet; alle anderen danach hinzugefügten Aktionen werden ignoriert.

Beachten Sie, dass die neue Aktion mit der Schaltfläche **Aktion anhängen** *hinter* der ausgewählten Aktion im Dialogfeld angehängt wird. Mit der Schaltfläche **Aktion einfügen** wird die neue Aktion *vor* der ausgewählten hinzugefügt. Um eine vorhandene Aktion zu löschen, klicken Sie auf eine beliebige Stelle in der Aktion und anschließend auf **Aktion löschen**.

6. Klicken Sie auf **OK,** um das Dialogfeld zu schließen. Beachten Sie, dass im Mapping anstelle der Schaltfläche **Aktion: Einfügen** A:In nun eine Schaltfläche **Aktion: Aktualisieren; Einfügen** (A:Up,In) angezeigt wird. Dies zeigt an, dass für diese Tabelle sowohl eine Aktualisierungs- als auch eine Einfügeaktion konfiguriert wurde.

Schritt 5: Vorschau auf das Mapping und Aktualisieren der Datenbank

Klicken Sie auf das Register **Ausgabe,** um eine Vorschau auf das Mapping zu sehen. Es wird ein SQL-Skript generiert, das Aktionen enthält, die an der Datenbank ausgeführt werden sollen. Das Skript hat die Datenbank noch nicht geändert; dies dient nur zur Vorschau.

```
1   /*
2   The following SQL statements are only for preview and may not be executed in another SQL query tool!
3   To execute these statements use function "Run SQL-script" from menu "Output".
4   Connect to database using the following connection-string:
5   Data Source=\\viepfs06\Documentation\Public\ExampleFiles\EN\MapForce\DB Update if... Insert
    Rest\altova.mdb;Provider=Microsoft.Jet.OLEDB.4.0
6   */
7
8   UPDATE [Person] SET [ForeignKey] = 1, [EMail] = 'A.Aldrich@microtech.com', [First] = 'Albert', [Last] = 'Aldrich',
    [PhoneExt] = 582, [Title] = 'Manager' WHERE ([Person].[PrimaryKey] = 1)
9
10  UPDATE [Person] SET [ForeignKey] = 1, [EMail] = 'b.bander@microtech.com', [First] = 'Bert', [Last] = 'Bander',
    [PhoneExt] = 471, [Title] = 'Accounts Receivable' WHERE ([Person].[PrimaryKey] = 2)
11
12  UPDATE [Person] SET [ForeignKey] = 1, [EMail] = 'c.clovis@microtech.com', [First] = 'Clive', [Last] = 'Clovis',
    [PhoneExt] = 963, [Title] = 'Accounting Manager' WHERE ([Person].[PrimaryKey] = 3)
13
14  UPDATE [Person] SET [ForeignKey] = 1, [EMail] = 'c.Cicada@microtech.com', [First] = 'Camilla ', [Last] = 'Cicada',
    [PhoneExt] = 765, [Title] = 'HR' WHERE ([Person].[PrimaryKey] = 30)
```

SQL-Skript (Teilansicht) vor dem Aktualisieren der Datenbank

Wie Sie sehen, sind im Vorschauskript keine INSERT-Anweisungen zu sehen. Dies ist ein normales Verhalten, da Datensätze auf Basis einer Bedingung eingefügt werden und die INSERT-Anweisungen vom Ergebnis der **Aktualisieren, wenn**..-Aktion (die vor der Ausführung des Mappings nicht bekannt ist) abhängig sind.

Anmerkung: Für bestimmte Datenbanktypen erstellt MapForce anstelle von UPDATE-Anweisungen MERGE-Anweisungen. Nähere Informationen dazu finden Sie unter MERGE-Anweisungen 310.

So führen Sie das Skript an der Datenbank aus:

- Klicken Sie im Menü **Ausgabe** auf **SQL-Script ausführen.**

Jetzt, da das Mapping ausgeführt und das Skript auf die Datenbank angewendet wurde, werden INSERT-Anweisungen auf dem **Ausgaberegister** angezeigt.

© 2016-2022 Altova GmbH

```
1   /*
2   The following SQL statements were executed during "Generate output" function.
3   Every single result is written right to the "-->>>" string.
4   These statements are only for preview and may not be executed in another SQL query tool!
5   The database was connected using the following connection-string:
6   Data Source=\\viepfs06\Documentation\Public\ExampleFiles\EN\MapForce\DB Update if... Insert
    Rest\altova.mdb;Provider=Microsoft.Jet.OLEDB.4.0
7   */
8
9   UPDATE [Person] SET [ForeignKey] = 1, [EMail] = 'A.Aldrich@microtech.com', [First] = 'Albert', [Last] = 'Aldrich',
    [PhoneExt] = 582, [Title] = 'Manager' WHERE ([Person].[PrimaryKey] = 1)
10  -->>> OK. 1 row(s).
11
12  UPDATE [Person] SET [ForeignKey] = 1, [EMail] = 'b.bander@microtech.com', [First] = 'Bert', [Last] = 'Bander', [PhoneExt]
     = 471, [Title] = 'Accounts Receivable' WHERE ([Person].[PrimaryKey] = 2)
13  -->>> OK. 1 row(s).
14
15  UPDATE [Person] SET [ForeignKey] = 1, [EMail] = 'c.clovis@microtech.com', [First] = 'Clive', [Last] = 'Clovis', [PhoneExt] =
     963, [Title] = 'Accounting Manager' WHERE ([Person].[PrimaryKey] = 3)
16  -->>> OK. 1 row(s).
17
18  UPDATE [Person] SET [ForeignKey] = 1, [EMail] = 'c.Cicada@microtech.com', [First] = 'Camilla ', [Last] = 'Cicada',
    [PhoneExt] = 765, [Title] = 'HR' WHERE ([Person].[PrimaryKey] = 30)
19  -->>> OK. 0 row(s).
20
21  INSERT INTO [Person] ([PrimaryKey], [ForeignKey], [EMail], [First], [Last], [PhoneExt], [Title]) VALUES (30, 1,
    'c.Cicada@microtech.com', 'Camilla ', 'Cicada', 765, 'HR')
22  -->>> OK. 1 row(s).
```

SQL-Skript (Teilansicht) nach Aktualisierung der Datenbank

Anmerkung: Das Ausführen des SQL-Skripts direkt von MapForce aus ist nur eine der Methoden, um eine Datenbank zu aktualisieren. Siehe auch Ausführen von Mappings, die Datenbanken modifizieren 282.

Wenn Sie die Tabelle "Person" auf dem Register "DB-Abfrage" von MapForce öffnen (siehe Anzeigen und Abfragen von Datenbanken 388), sehen Sie als Ergebnis des Mappings Folgendes:

- Alle Datenbankdatensätze, für die es Primärschlüsselentsprechungen in der XML-Datei gab, wurden aktualisiert. Beispiele dafür sind die Datensätze mit den Primärschlüsseln 1, 2, 3, 4 und 5.
- Alle Datenbankdatensätze, für die es in der XML-Datei keine Primärschlüsselentsprechungen gab, sind unverändert geblieben. Beispiele dafür sind die Datensätze mit den Primärschlüsseln 6, 7, 8 und 9.
- Es wurden neue Datensätze in die Tabelle "Person" eingefügt (Datensätze, für die in der Datenbank noch kein Schlüssel vorhanden war). Beispiele dafür sind die Datensätze mit den Primärschlüsseln 30 und 31.

	PrimaryKey	ForeignKey	EMail	First	Last	PhoneExt	Title
1	1	1	A.Aldrich@microtech.com	Albert	Aldrich	582	Manager
2	2	1	b.bander@microtech.com	Bert	Bander	471	Accounts Receivable
3	3	1	c.clovis@microtech.com	Clive	Clovis	963	Accounting Manager
4	4	2	d.Durnell@microtech.com	Dave	Durnelll	621	Marketing Manager Europe
5	5	2	e.ellas@microtech.com	Eve	Ellas	753	Art Director
6	6	3	f.landis@nanonull.com	Fred	Landis	951	Program Manager
7	7	3	m.landis@nanonull.com	Michelle	Butler	654	Software Engineer
8	8	3	t.little@nanonull.com	Ted	Little	852	Software Engineer
9	9	3	a.way@nanonull.com	Ann	Way	951	Technical Writer
10	10	3	l.gardner@nanonull.com	Liz	Gardner	753	Software Engineer
11	11	3	p.smith@nanonull.com	Paul	Smith	334	Software Engineer
12	12	4	a.martin@nanonull.com	Alex	Martin	778	IT Manager
13	13	4	g.hammer@nanonull.com	George	Hammer	223	Web Developer
14	14	4	n.newbury@microtech.com	Nira	Newbury	241	Support Engineer
15	15	4	o.origone@microtech.com	Olanda	Origone	345	Support Engineer
16	16	5	s.meier@microtech.com	Paul	Ponetti	114	Office Manager
17	17	5	t.bone@microtech.com	Quang	Qui	331	Accounts Receivable
18	18	6	m.nafta@microtech.com	Rita	Rotter	122	PR & Marketing Manager US
19	19	7	v.bass@microtech.com	Silla	Semua	716	IT Manager
20	20	7	c.franken@microtech.com	Tom	Tinker	147	Support Engineer
21	21	7	m.redgreen@microtech.com	Uri	Ugoner	152	Support Engineer
22	30	1	c.Cicada@microtech.com	Camilla	Cicada	765	HR
23	31	1	c.corrigan@microtech.com	Carol	Corrigan	629	Admin

Die Tabelle "Person" nach Aktualisierung der Datenbank

4.2.3.5 MERGE-Anweisungen

Für bestimmte Mappings, die sowohl Daten in eine Datenbanktabelle einfügen als auch Daten darin aktualisieren (siehe auch Die Aktion "Aktualisieren, wenn...Rest einfügen" 303), generiert MapForce MERGE-Anweisungen, die zur Mapping-Laufzeit an der Datenbank ausgeführt werden. Als Ausführungsprozessor muss nicht notwendigerweise MapForce verwendet werden. Siehe Ausführen von Mappings, die Datenbanken modifizieren 282.

MERGE-Anweisungen werden für die folgenden Datenbanktypen unterstützt:

- SQL Server 2008 und höher
- Oracle
- DB2
- Firebird

 © 2016-2022 Altova GmbH

Mittels MERGE-Anweisungen kann die Anzahl der Datenbankserveraufrufe reduziert werden, da in diesen Anweisungen die INSERT- und die UPDATE-Anweisung in einer Anweisung kombiniert sind. Außerdem wird die Konsistenzprüfung bei MERGE-Anweisungen von der Datenbank durchgeführt.

Um zu sehen, ob das Mapping zur Laufzeit (anstelle einer Kombination von INSERT- und UPDATE-Anweisungen) MERGE-Datenbankanweisungen ausführt, gehen Sie folgendermaßen vor:

1. Erstellen Sie ein Mapping, in dem sowohl eine **Aktualisieren, wenn.. .**als auch eine **Rest einfügen**-Aktion verwendet wird. Ein Beispiel dazu finden Sie unter Die Aktion "Aktualisieren, wenn...Rest einfügen" 303.
2. Klicken Sie auf das Register **Ausgabe**, um eine Vorschau auf das Mapping zu sehen.

Wenn MERGE vom Datenbanktyp unterstützt wird, enthält das generierten SQL-Skript MERGE-Anweisungen, wie z.B. die folgende:

```
1   /*
2   The following SQL statements are only for preview and may not be executed in another SQL query tool!
3   To execute these statements use function "Run SQL-script" from menu "Output".
4   Connect to database using the following connection-string:
5   Data Source=                    ;Initial
    Catalog=DB MERGE;MultipleActiveResultSets=True;Password=•••••;Persist Security Info=True;User
    ID=
6   */
7
8   SET QUOTED_IDENTIFIER ON
9
10  UPDATE [dbo].[Users] SET [FirstName] = (CAST('Despine' AS nvarchar(50))), [LastName] =
    (CAST('Buttler' AS nvarchar(50))) WHERE ([dbo].[Users].[UserID] = 1)
11
12  SELECT [UserID] FROM [dbo].[Users] WHERE ([UserID] = 1)
13  -->>> %UserID1%
14
15  DELETE FROM [dbo].[Addresses] WHERE EXISTS(SELECT * FROM [dbo].[Users] WHERE [dbo].[Users].[UserID] =
    [dbo].[Addresses].[UserID] AND ([dbo].[Users].[UserID] = '%UserID1%'))
16
17  MERGE INTO [dbo].[Addresses] AS T USING ( VALUES ( '%UserID1%', 1, 1, (CAST('Home' AS nvarchar(20))),
    (CAST('Louisville' AS nvarchar(50))), (CAST('Elm Street' AS nvarchar(50))), (CAST('12' AS
    nvarchar(20))) ) ) AS S ( [UserID], [IsShipping], [IsBilling], [AddressType], [City], [Street],
    [Number] ) ON ( (S.[UserID] = T.[UserID]) ) WHEN MATCHED THEN UPDATE SET [IsShipping] = S.
    [IsShipping], [IsBilling] = S.[IsBilling], [AddressType] = S.[AddressType], [City] = S.[City],
    [Street] = S.[Street], [Number] = S.[Number] WHEN NOT MATCHED THEN INSERT ( [UserID], [IsShipping],
    [IsBilling], [AddressType], [City], [Street], [Number] ) VALUES ( S.[UserID], S.[IsShipping], S.
    [IsBilling], S.[AddressType], S.[City], S.[Street], S.[Number] );
18
```

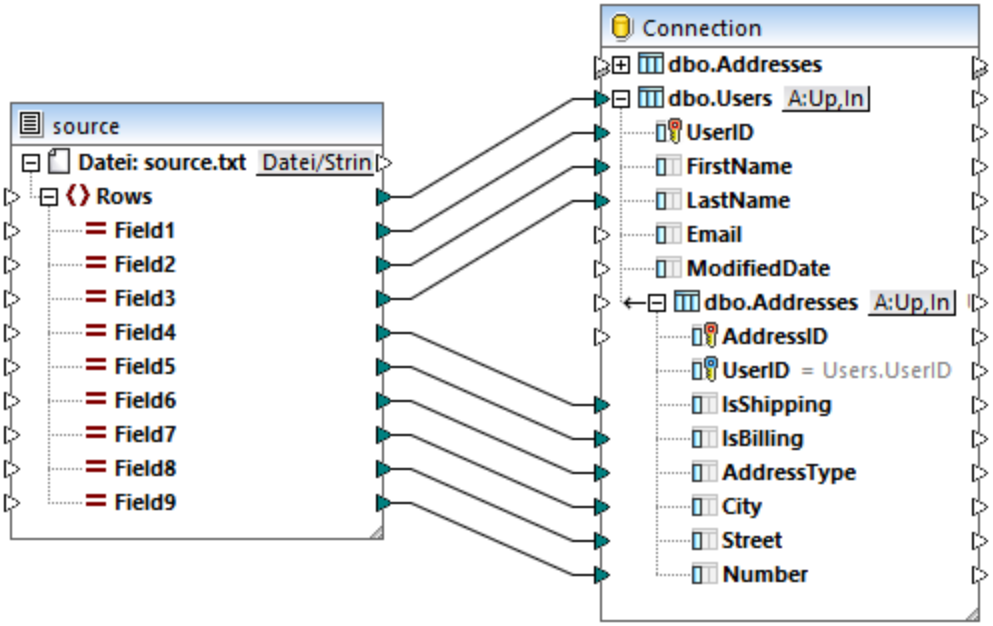
Mapping | DB-Abfrage | **Ausgabe**

Wenn MERGE vom Datenbanktyp nicht unterstützt wird, enthält das generierte SQL-Skript nur UPDATE-Anweisungen. In der Vorschau sind keine INSERT-Anweisungen sichtbar, da diese nur ausgeführt werden, wenn die **Aktualisieren, wenn...**-Bedingung erfüllt wird (und dies vor der Ausführung des Mappings noch nicht bekannt ist).

Anmerkungen

- MapForce erstellt automatisch MERGE-Anweisungen, wenn es einen unterstützten Datenbanktyp erkennt. Sie können manuell keinen Einfluss darauf nehmen, ob MapForce eine MERGE-Anweisung erstellen soll.

- Wenn Sie mehrere Tabellen mit Parent-Child-Beziehungen aktualisieren, werden Merges nur für "leaf"-Tabellen erstellt. Eine "leaf"-Tabelle ist die am tiefsten verschachtelte gemappte Child-Tabelle. So wurden z.B. im Mapping unten sowohl für die Tabelle `Users` als auch für `Addresses` **Aktualisieren, wenn... Rest einfügen...** Aktionen definiert. MERGE-Anweisungen werden jedoch nur für die leaf-Tabelle `Addresses` generiert. Die übergeordnete Tabelle `Users` erhält anstelle von MERGE-Anweisungen UPDATE- oder INSERT-Anweisungen.



- Bei MERGE-Anweisungen wird die "Bulk Transfer"-Option (siehe Bulk-Einfügungen (MapForce Server) 329 nur für ODBC- und JDBC-Datenbankverbindungen unterstützt.

4.2.3.6 Optionen für Child-Tabellen bei Aktualisierung einer Parent-Tabelle

Wenn mit dem Mapping eine Tabelle aktualisiert wird, bei der es sich um eine "Parent"-Tabelle handelt (d.h. die Tabelle hat Sekundärschlüsselbeziehungen zu anderen Tabellen), können Sie konfigurieren, wie die abhängigen Datensätze in der Quellkomponente und in der Zieltabelle behandelt werden sollen. Angenommen Sie möchten die Tabelle "Department" in der Datenbank **altova.mdb** aktualisieren. Da jede Person (person) über einen Sekundärschlüssel mit einer Abteilung (department) verknüpft ist, sollte auch an der Tabelle "Person" eine Aktion (z.B. Einfügen, Aktualisieren oder Löschen) ausgeführt werden. Dadurch bewahren Sie die Datenbankintegrität und vermeiden Mapping-Fehler.

In diesem Kapitel werden die Optionen behandelt, die für die Tabelle "Person" zur Verfügung stehen, wenn Sie die Parent-Tabelle "Department" aktualisieren. Im Beispiel werden die folgenden Dateien verwendet:

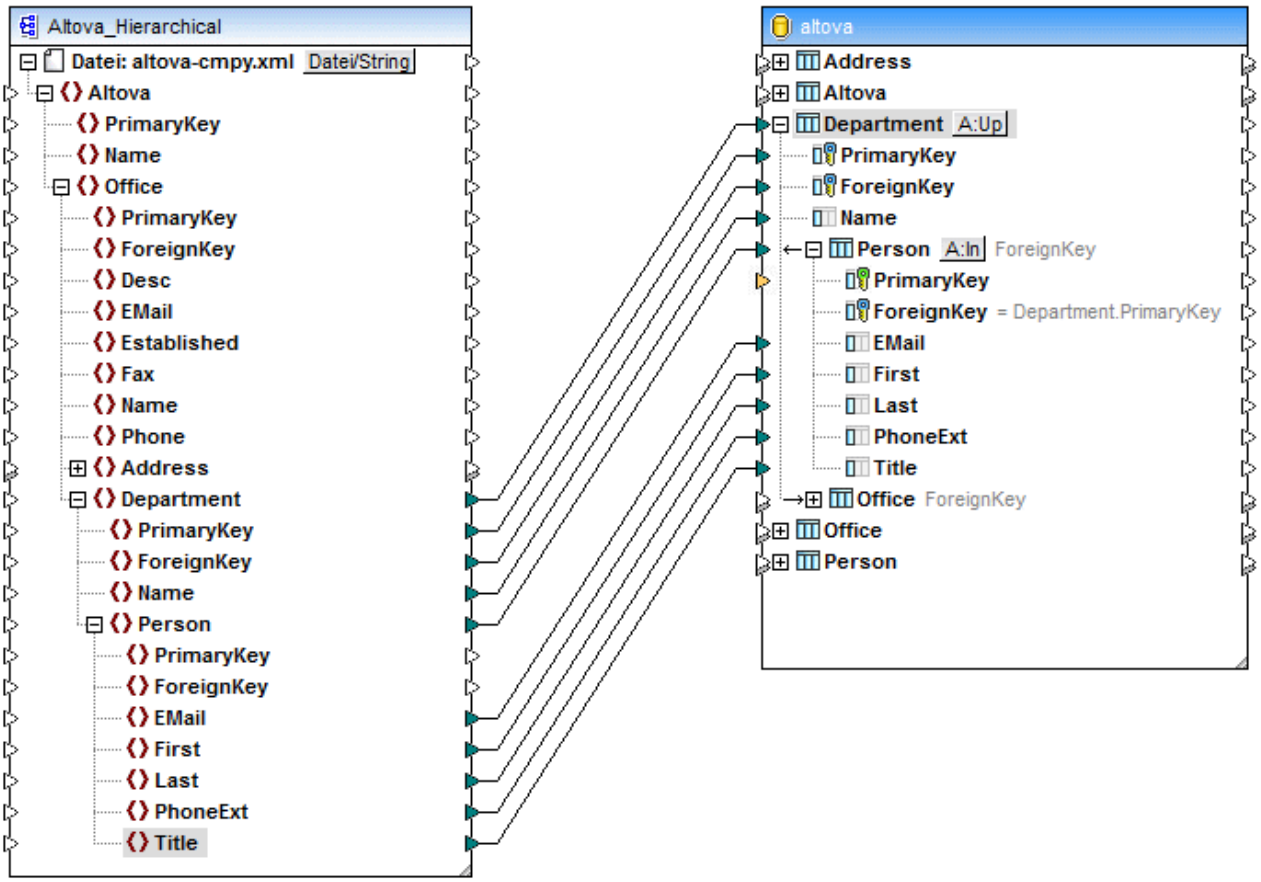
- **altova-cmpy-extra.xml** - enthält die Quelldaten, die in die Datenbank eingefügt werden sollen
- **Altova_Hierarchical.xsd** - das Schema, anhand dessen die obige Instanzdatei validiert wird
- **Altova.mdb** - die Zieldatenbank, die aktualisiert werden soll.

Sie finden all diese Dateien im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\.** Der vollständige Pfad wurde in der Folge aus Gründen der Einfachheit weggelassen.

© 2016-2022 Altova GmbH

Im Mapping in diesem Beispiel wird eine Beispieldatenbankdatei geändert. Es wird empfohlen, eine Sicherungskopie der Originaldatenbank zu erstellen und mit einer neuen Kopie zu arbeiten, bevor Sie die unten stehende Anleitung befolgen. Dadurch stellen Sie sicher, dass die Originaldaten nicht überschrieben werden und Sie dasselbe Ergebnis, wie unten beschrieben, erhalten. Nähere Informationen dazu finden Sie unter Ausführen von Mappings, die Datenbanken modifizieren 282.

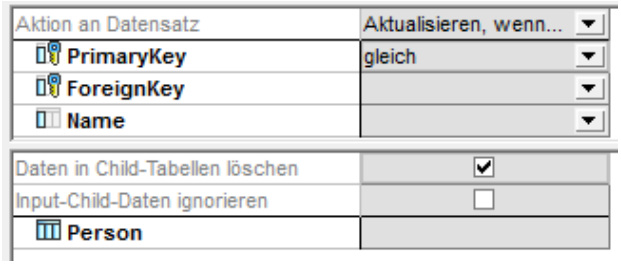
Fügen Sie zuerst das XML-Quellschema und die XML-Instanz sowie die Zieldatenbank zum Mapping hinzu (siehe Beispiel: Hinzufügen der Datenbank "altova.mdb" zum Mapping 257). Gehen Sie vor, wie im Kapitel Die Aktion "Aktualisieren, wenn...Rest einfügen" 303 beschrieben. Ziehen Sie anschließend die Mapping-Verbindungen, wie unten gezeigt:



Wie Sie in der Abbildung oben sehen, aktualisiert das Mapping die Tabelle "Department" in der Zieldatenbank. Die Tabelle "Department" wurde als "Root"-Tabelle ausgewählt. Nähere Informationen zu Root-Tabellen und deren Zweck finden Sie unter Behandlung von Datenbankbeziehungen 266. Thema des aktuellen Kapitels ist die Aktion, die an der Child-Tabelle "Person" ausgeführt wird.

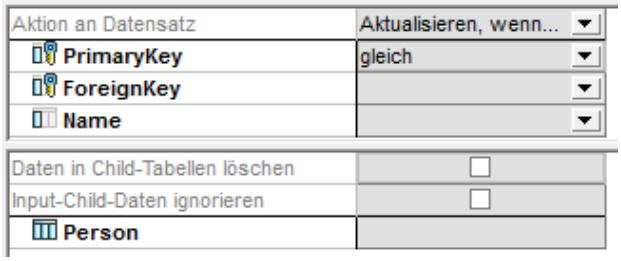
In den folgenden Tabellen sind verschiedene Konfigurationsoptionen und das entsprechende Mapping-Ergebnis beschrieben. Diese Optionen können im Dialogfeld "Datenbankaktionen" der Parent-Tabelle "Department" und der Child-Tabelle "Person" ausgewählt werden.

Konfiguration A

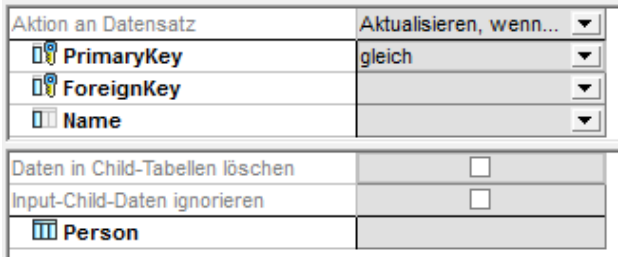
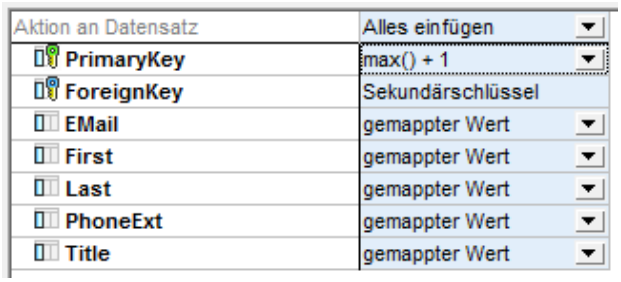
Einstellungen	Mapping-Ergebnis
 *Tabelle "Department"*  *Tabelle "Person"*	• Aktualisiert die Department-Datensätze, deren `PrimaryKey` in der XML-Quelldatei mit dem `PrimaryKey` in der Datenbanktabelle übereinstimmt. • Vorhandene Department-Datensätze, die in der XML-Input-Datei keine Entsprechung haben (in der Quelldatei ist kein solcher Schlüssel vorhanden), werden nicht aktualisiert. • Fügt alle Person-Datensätze aus der XML-Input-Instanz, die nicht bereits in der Datenbank vorhanden sind, in die Datenbank ein. • Löscht die Child-Daten (Person-Datensätze) derjenigen Department-Datensätze, die die **Aktualisieren, wenn...**-Bedingung erfüllen.

© 2016-2022 Altova GmbH

Konfiguration B

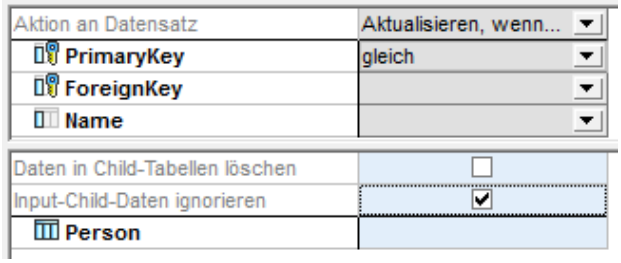
Einstellungen	Mapping-Ergebnis
 *Tabelle "Department"*  *Tabelle "Person"*	Das Mapping schlägt mit einem SQL-Fehler fehl. Grund dafür ist, dass versucht wird, neue Person-Datensätze einzufügen, die denselben Primärschlüssel wie die vorhandenen Person-Datensätze haben. Wenn Sie zusätzlich zu den bereits vorhandenen Datensätzen neue Datensätze aus der XML-Input-Datei einfügen möchten, dann verwenden Sie die nächste Option.

Konfiguration C

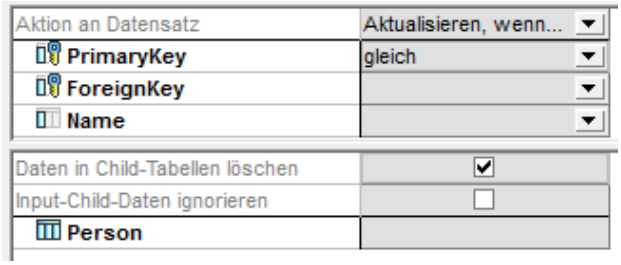
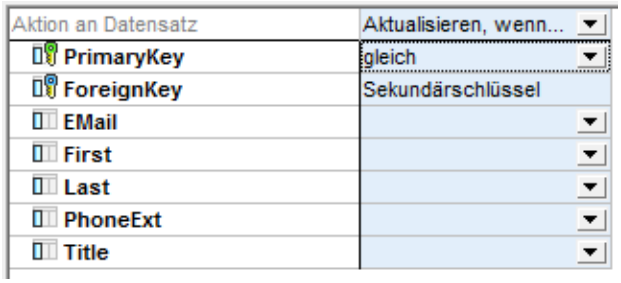
Einstellungen	Mapping-Ergebnis
 *Tabelle "Department"*  *Tabelle "Person"*	• Aktualisiert die Department-Datensätze, bei denen `PrimaryKey` in der XML-Quelldatei mit `PrimaryKey` in der Datenbanktabelle übereinstimmt. • Vorhandene Department-Datensätze, die in der XML-Input-Datei keine Entsprechung haben (in der Quelldatei ist kein solcher Schlüssel vorhanden), werden nicht aktualisiert. • Zusätzlich zu den vorhandenen Datensätzen werden neue Person-Datensätze (mit generierten Primärschlüsseln) in die Tabelle "Person" eingefügt.

© 2016-2022 Altova GmbH

Konfiguration D

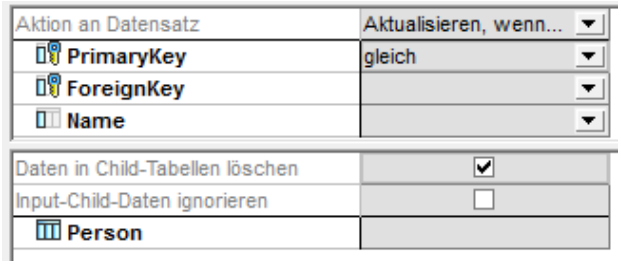
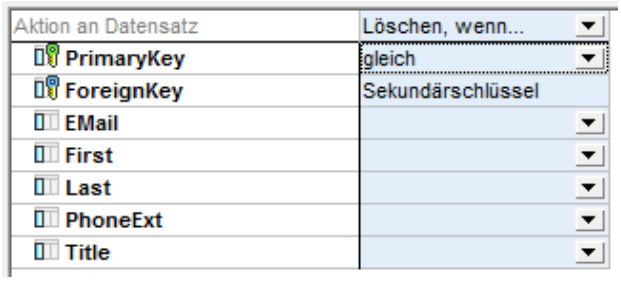
Einstellungen	Mapping-Ergebnis
 *Tabelle "Department"*  *Tabelle "Person"*	• Aktualisiert die Department-Datensätze, bei denen `PrimaryKey` in der XML-Quelldatei mit `PrimaryKey` in der Datenbanktabelle übereinstimmt. • Vorhandene Department-Datensätze, die in der XML-Input-Datei keine Entsprechung haben (in der Quelldatei ist kein solcher Schlüssel vorhanden), werden nicht aktualisiert. • In die Tabelle "Person" werden keine Datensätze eingefügt, weil die Option **Input-Child-Daten ignorieren** für die Parent-Tabelle "Department" aktiviert ist.

Konfiguration E

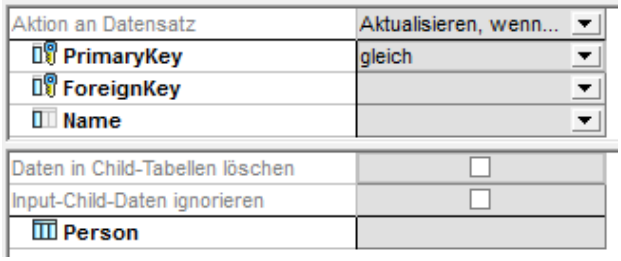
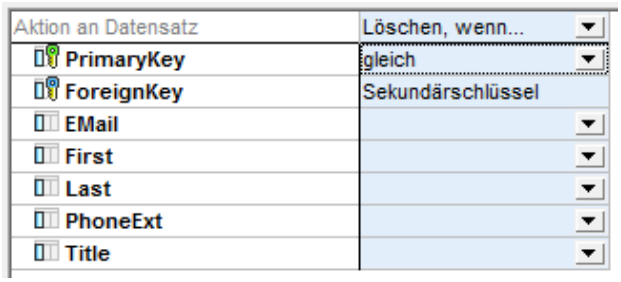
Einstellungen	Mapping-Ergebnis
 *Tabelle "Department"*  *Tabelle "Person"*	• Aktualisiert die Department-Datensätze, bei denen `PrimaryKey` in der XML-Quelldatei mit `PrimaryKey` in der Datenbanktabelle übereinstimmt. • Vorhandene Department-Datensätze, die in der XML-Input-Datei keine Entsprechung haben (in der Quelldatei ist kein solcher Schlüssel vorhanden), werden nicht aktualisiert. • Löscht alle Personen, die mit einer Abteilung (Department) verknüpft sind, für die es in der XML-Quelldatei eine Primärschlüsselentsprechung gibt. Der Grund dafür ist, dass die Option **Daten in Child-Tabellen löschen** für die Parent-Tabelle "Department" aktiviert ist. • Personen-Datensätze, die mit einer Abteilung verknüpft sind, die die **Aktualisieren, wenn...-** Bedingung nicht erfüllen, bleiben in der Datenbank. • Es werden keine Datensätze in der Tabelle "Person" aktualisiert.

© 2016-2022 Altova GmbH

Konfiguration F

Einstellungen	Mapping-Ergebnis
 *Tabelle "Department"*  *Tabelle "Person"*	• Aktualisiert die Department-Datensätze, bei denen `PrimaryKey` in der XML-Quelldatei mit `PrimaryKey` in der Datenbanktabelle übereinstimmt. • Vorhandene Department-Datensätze, die in der XML-Input-Datei keine Entsprechung haben (in der Quelldatei ist kein solcher Schlüssel vorhanden), werden nicht aktualisiert. • Löscht alle Personen, die mit einer Abteilung (Department) verknüpft sind, für die es in der XML-Quelldatei eine Primärschlüsselentsprechung gibt. Der Grund dafür ist, dass die Option **Daten in Child-Tabellen löschen** für die Parent-Tabelle "Department" aktiviert ist. • Personen-Datensätze, die mit einer Abteilung verknüpft sind, die die **Aktualisieren, wenn...-** Bedingung nicht erfüllen, bleiben in der Datenbank.

Konfiguration G

Einstellungen	Mapping-Ergebnis
 *Tabelle "Department"*  *Tabelle "Person"*	• Aktualisiert die Department-Datensätze, bei denen `PrimaryKey` in der XML-Quelldatei mit `PrimaryKey` in der Datenbanktabelle übereinstimmt. • Vorhandene Department-Datensätze, die in der XML-Input-Datei keine Entsprechung haben (in der Quelldatei ist kein solcher Schlüssel vorhanden), werden nicht aktualisiert. • Löscht alle Personen, die beide der folgenden Bedingungen erfüllen: a. Der Personen-Datensatz ist mit einer Abteilung verknüpft, die eine `PrimaryKey`-Entsprechung in der XML-Quelldatei hat und b. Der Personen-Datensatz hat in der XML-Quelldatei eine `PrimaryKey`-Entsprechung.

4.2.3.7 Die Aktion "Löschen, wenn..."

Mit Hilfe der Datenbankaktion **Löschen, wenn..**. können Daten auf Basis einer Bedingung aus einer Datenbanktabelle gelöscht werden. Sie können diese Aktion über das Dialogfeld "Datenbankaktionen" definieren (siehe Einstellungen für Datenbankaktionen 336). Wenn Sie z.B. Daten von einer XML-Quelldatei auf eine Zieldatenbank mappen, können Sie eine **Löschen, wenn...**-Bedingung definieren, um zu überprüfen, ob ein bestimmtes Feld (normalerweise ein Primärschlüsselwert) in der XML-Quelldatei mit einem Feld im Zieldatenbankdatensatz übereinstimmt . Wenn die **Löschen, wenn...**-Bedingung erfüllt wird (d.h. wenn die beiden Felder gleich sind), so wird der Datenbankdatensatz gelöscht, sobald das Mapping ausgeführt wird.

Anmerkung: Die Datenbankaktion **Löschen, wenn...** ist nicht mit der Option **Daten in Child-Tabellen löschen** aus dem Dialogfeld "Datenbankaktionen" zu verwechseln. Die Datenbankaktion **Löschen, wenn...** wirkt sich nur auf die Tabelle aus, für die die Aktion definiert ist; andere Tabellen sind davon nicht betroffen.

In diesem Beispiel wird gezeigt, wie Sie Daten auf Basis einer Bedingung aus einer Datenbanktabelle löschen und auf Basis des Ergebnisses der Lösch-Bedingung Datensätze in dieselbe Datenbanktabelle einfügen.

Im Beispiel werden die folgenden Dateien verwendet:

- **altova-cmpy-extra.xml** - enthält die Quelldaten, die in die Datenbank eingefügt werden sollen
- **Altova_Hierarchical.xsd** - das Schema, anhand dessen die obige Instanzdatei validiert wird
- **Altova.mdb** - die Zieldatenbank, die aktualisiert werden soll.

 © 2016-2022 Altova GmbH

Sie finden all diese Dateien im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\.** Der vollständige Pfad wurde in der Folge aus Gründen der Einfachheit weggelassen.

Im Mapping in diesem Beispiel wird eine Beispieldatenbankdatei geändert. Es wird empfohlen, eine Sicherungskopie der Originaldatenbank zu erstellen und mit einer neuen Kopie zu arbeiten, bevor Sie die unten stehende Anleitung befolgen. Dadurch stellen Sie sicher, dass die Originaldaten nicht überschrieben werden und Sie dasselbe Ergebnis, wie unten beschrieben, erhalten. Nähere Informationen dazu finden Sie unter Ausführen von Mappings, die Datenbanken modifizieren 282.

Ziel des Mappings ist das folgende:

- Wenn in der XML-Quelldatei und in der Zieltabelle "Person" Personendatensätze mit demselben `PrimaryKey` vorhanden sind, müssen diese aus der Tabelle "Person" gelöscht werden.
- Alle anderen Datensätze aus der XML-Quelldatei müssen in die Tabelle "Person" eingefügt werden.

Gehen Sie dazu folgendermaßen vor:

Schritt 1: Fügen Sie die XML-Quellkomponente ein

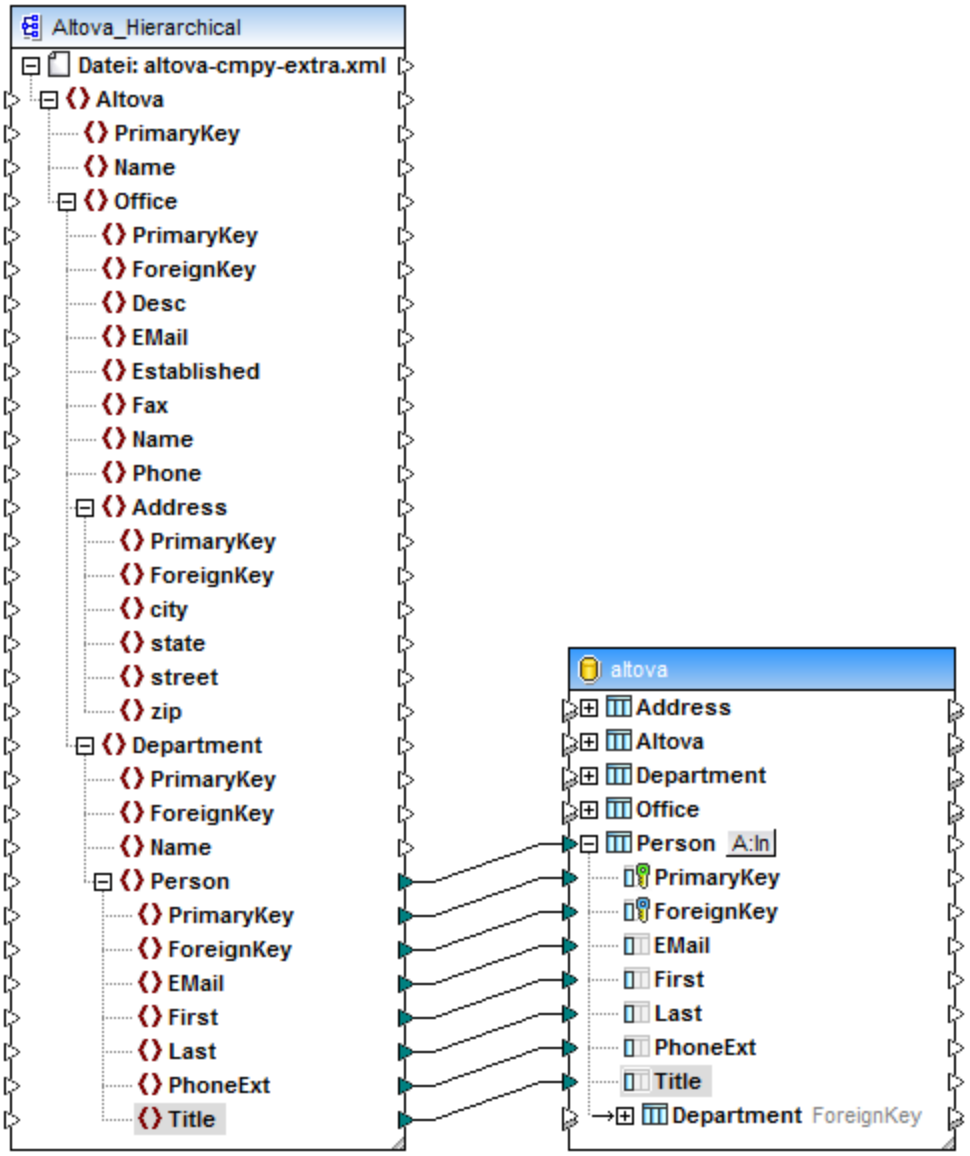
- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur Datei **Altova_Hierarchical.xsd**. Wenn Sie aufgefordert werden, eine Instanzdatei anzugeben, navigieren Sie zur Datei **altova-cmpy-extra.xml**.

Schritt 2: Fügen Sie die Zieldatenbank ein

- Klicken Sie im Menü **Einfügen** auf **Datenbank** und befolgen Sie die Anweisungen des Assistenten, um eine Verbindung zu **Altova.mdb** herzustellen (siehe Beispiel: Hinzufügen der Datenbank "altova.mdb" zum Mapping 257).

Schritt 3: Ziehen Sie die Verbindungen

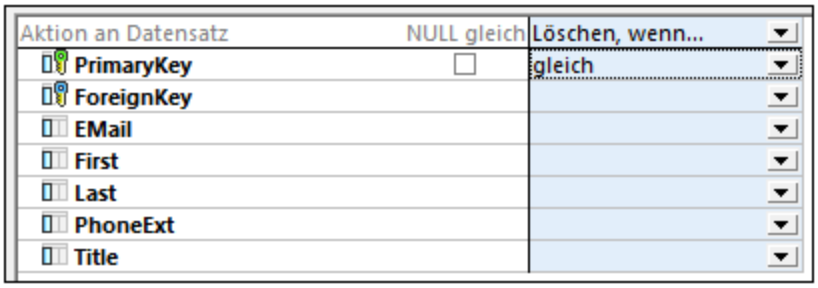
- Ziehen Sie die Mapping-Verbindungen wie unten gezeigt.



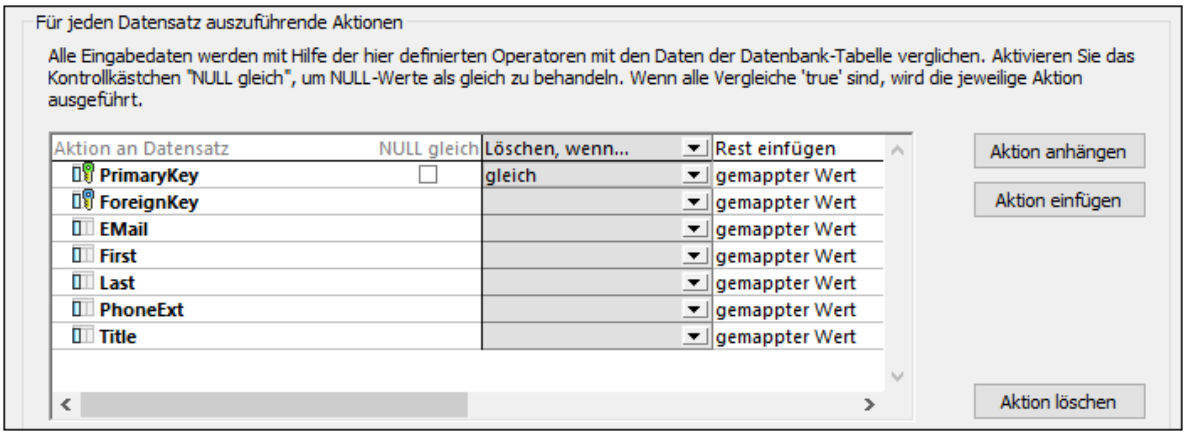
Schritt 4: Konfigurieren Sie die "Aktualisieren,wenn... Rest einfügen"-Aktionen

1. Klicken Sie in der Zielkomponente auf die Schaltfläche **Aktion: Einfügen** (A:In) neben der Tabelle "Person".
2. Wählen Sie neben **Aktion an Datensatz** den Eintrag **Löschen, wenn...** . Dadurch wird die Datenbankaktion in eine auf einer Bedingung basierende Löschaktion geändert, d.h. der aktuelle Datensatz wird nur gelöscht, wenn eine Bedingung erfüllt wird (siehe nächster Schritt).
3. Wählen Sie neben **PrimaryKey** den Wert **gleich** aus, wie unten gezeigt. Damit wird die Löschbedingung definiert, d.h. der Datenbankdatensatz wird nur gelöscht, wenn sein **PrimaryKey**-Wert gleich dem **PrimaryKey**-Wert aus dem Mapping ist.

© 2016-2022 Altova GmbH



4. Aktivieren Sie optional das Kontrollkästchen **NULL gleich**. In diesem Beispiel spielt das Kontrollkästchen keine Rolle, da weder das Feld **PrimaryKey** in der XML-Quelldatei noch das Feld **PrimaryKey** in der Datenbank Nullwerte enthält. Wenn Ihre Daten jedoch Nullwerte enthalten und solche Nullwerte als identisch behandelt werden sollen, sollten Sie dieses Kontrollkästchen aktivieren, da Sie sonst unerwünschte Ergebnisse erhalten könnten. Nähere Informationen dazu finden Sie unter Behandlung von Nullwerten in Datenbankaktionen 333.
5. Klicken Sie auf **Aktion anhängen**. Daraufhin wird rechts von der vorhandenen **Löschen, wenn**-Aktion eine neue Aktion hinzugefügt. Konfigurieren Sie die neue Aktion als **Rest einfügen**:



In der Abbildung oben wurden die Datenbankaktionen den Zielen des Mappings entsprechend konfiguriert, d.h. der Datensatz wird nur dann gelöscht, wenn die **Löschen, wenn..**-Bedingung zutrifft. Andernfalls wird der Datensatz eingefügt. Mit der Option "gemappter Wert" wird festgelegt, dass alle Felder des Datensatzes anhand der Werte aus dem Mapping befüllt werden.

6. Klicken Sie auf **OK,** um das Dialogfeld zu schließen. Beachten Sie, dass im Mapping anstelle der Schaltfläche **Aktion: Einfügen** A:In nun eine Schaltfläche **Aktion: Löschen; Einfügen** (A:Up,In) angezeigt wird. Dies zeigt an, dass für diese Tabelle sowohl eine Lösch- als auch eine Einfügeaktion konfiguriert wurde.

Schritt 5: Vorschau auf das Mapping und Aktualisieren der Datenbank

Klicken Sie auf das Register **Ausgabe,** um eine Vorschau auf das Mapping zu sehen. Es wird ein SQL-Skript generiert, das Aktionen enthält, die an der Datenbank ausgeführt werden sollen. Das Skript hat die Datenbank noch nicht geändert; dies dient nur zur Vorschau. So führen Sie das Skript an der Datenbank aus:

- Klicken Sie im Menü **Ausgabe** auf **SQL-Script ausführen.**

Anmerkung: Das Ausführen des SQL-Skripts direkt von MapForce aus ist nur eine der Methoden, um eine Datenbank zu aktualisieren. Siehe auch Ausführen von Mappings, die Datenbanken modifizieren 282.

Wenn Sie die Tabelle "Person" auf dem Register "DB-Abfrage" von MapForce öffnen (siehe Anzeigen und Abfragen von Datenbanken 388), sehen Sie als Ergebnis des Mappings Folgendes:

- Alle Datenbankdatensätze, für die es Primärschlüsselentsprechungen in der XML-Datei gab, wurden gelöscht. Beispiele dafür sind die Datensätze mit den Primärschlüsseln 1, 2, 3, 4 und 5.
- Alle Datenbankdatensätze, für die es in der XML-Datei keine Primärschlüsselentsprechungen gab, sind unverändert geblieben. Beispiele dafür sind die Datensätze mit den Primärschlüsseln 6, 7, 8, 9, 10, 11, 12 und 13.
- Es wurden neue Datensätze in die Tabelle "Person" eingefügt (Datensätze, für die in der Datenbank noch kein Schlüssel vorhanden war). Beispiele dafür sind die Datensätze mit den Primärschlüsseln 30 und 31.

	PrimaryKey	ForeignKey	EMail	First	Last	PhoneExt	Title
1	6	3	f.landis@nanonull.com	Fred	Landis	951	Program Manager
2	7	3	m.landis@nanonull.com	Michelle	Butler	654	Software Engineer
3	8	3	t.little@nanonull.com	Ted	Little	852	Software Engineer
4	9	3	a.way@nanonull.com	Ann	Way	951	Technical Writer
5	10	3	l.gardner@nanonull.com	Liz	Gardner	753	Software Engineer
6	11	3	p.smith@nanonull.com	Paul	Smith	334	Software Engineer
7	12	4	a.martin@nanonull.com	Alex	Martin	778	IT Manager
8	13	4	g.hammer@nanonull.com	George	Hammer	223	Web Developer
9	30	1	c.Cicada@microtech.com	Camilla	Cicada	765	HR
10	31	1	c.corrigan@microtech.com	Carol	Corrigan	629	Admin

Die Tabelle "Person" nach Aktualisierung der Datenbank

4.2.3.8 Die Aktion "Ignorieren, wenn..."

Mit Hilfe der Datenbankaktion **Ignorieren, wenn..**. wird auf Basis einer Bedingung verhindert, dass bestimmte Datensätze in einer Datenbanktabelle aktualisiert werden. Die Aktion **Ignorieren, wenn...** lässt sich nur in Verbindung mit einer anderen Datenbankaktion (z.B. **Rest einfügen**) sinnvoll einsetzen. Wenn Sie z.B. Daten von einer XML-Quelldatei auf eine Zieldatenbank mappen, können Sie eine **Ignorieren, wenn...**-Bedingung konfigurieren, um zu überprüfen, ob ein bestimmtes Feld in der XML-Quelldatei mit einem Feld (normalerweise ein Primärschlüsselwert) im Zieldatenbankdatensatz übereinstimmt. Wenn die **Ignorieren, wenn...**-Bedingung erfüllt wird (d.h. wenn die beiden Felder gleich sind), so wird der Datenbankdatensatz ignoriert, wenn das Mapping ausgeführt wird und es wird die nächste definierte Aktion (z.B. **Rest einfügen**) ausgeführt.

In diesem Beispiel wird gezeigt, wie Sie auf Basis des Ergebnisses der **Ignorieren, wenn...**-Bedingung Datensätze in eine Datenbanktabelle einfügen.

Im Beispiel werden die folgenden Dateien verwendet:

- **altova-cmpy-extra.xml** - enthält die Quelldaten, die in die Datenbank eingefügt werden sollen

 © 2016-2022 Altova GmbH

- **Altova_Hierarchical.xsd** - das Schema, anhand dessen die obige Instanzdatei validiert wird
- **Altova.mdb** - die Zieldatenbank, die aktualisiert werden soll.

Sie finden all diese Dateien im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\.** Der vollständige Pfad wurde in der Folge aus Gründen der Einfachheit weggelassen.

Im Mapping in diesem Beispiel wird eine Beispieldatenbankdatei geändert. Es wird empfohlen, eine Sicherungskopie der Originaldatenbank zu erstellen und mit einer neuen Kopie zu arbeiten, bevor Sie die unten stehende Anleitung befolgen. Dadurch stellen Sie sicher, dass die Originaldaten nicht überschrieben werden und Sie dasselbe Ergebnis, wie unten beschrieben, erhalten. Nähere Informationen dazu finden Sie unter Ausführen von Mappings, die Datenbanken modifizieren 282.

Ziel des Mappings ist das folgende:

- Wenn in der XML-Quelldatei und in der Zieltabelle "Person" Personendatensätze mit demselben `PrimaryKey` vorhanden sind, so wird an diesen keine Aktion ausgeführt (d.h sie werden ignoriert).
- Wenn in der Tabelle "Person" Personendatensätze vorhanden sind, die die obige Bedingung nicht erfüllen, darf auch an diesen keine Aktion durchgeführt werden.
- Datensätze aus der XML-Quelldatei, die keine Entsprechung (keinen Primärschlüssel) in der Tabelle "Person" haben, müssen als neu behandelt werden und werden mit einem neuen Primärschlüssel in die Tabelle "Person" eingefügt.

Gehen Sie dazu folgendermaßen vor:

Schritt 1: Fügen Sie die XML-Quellkomponente ein

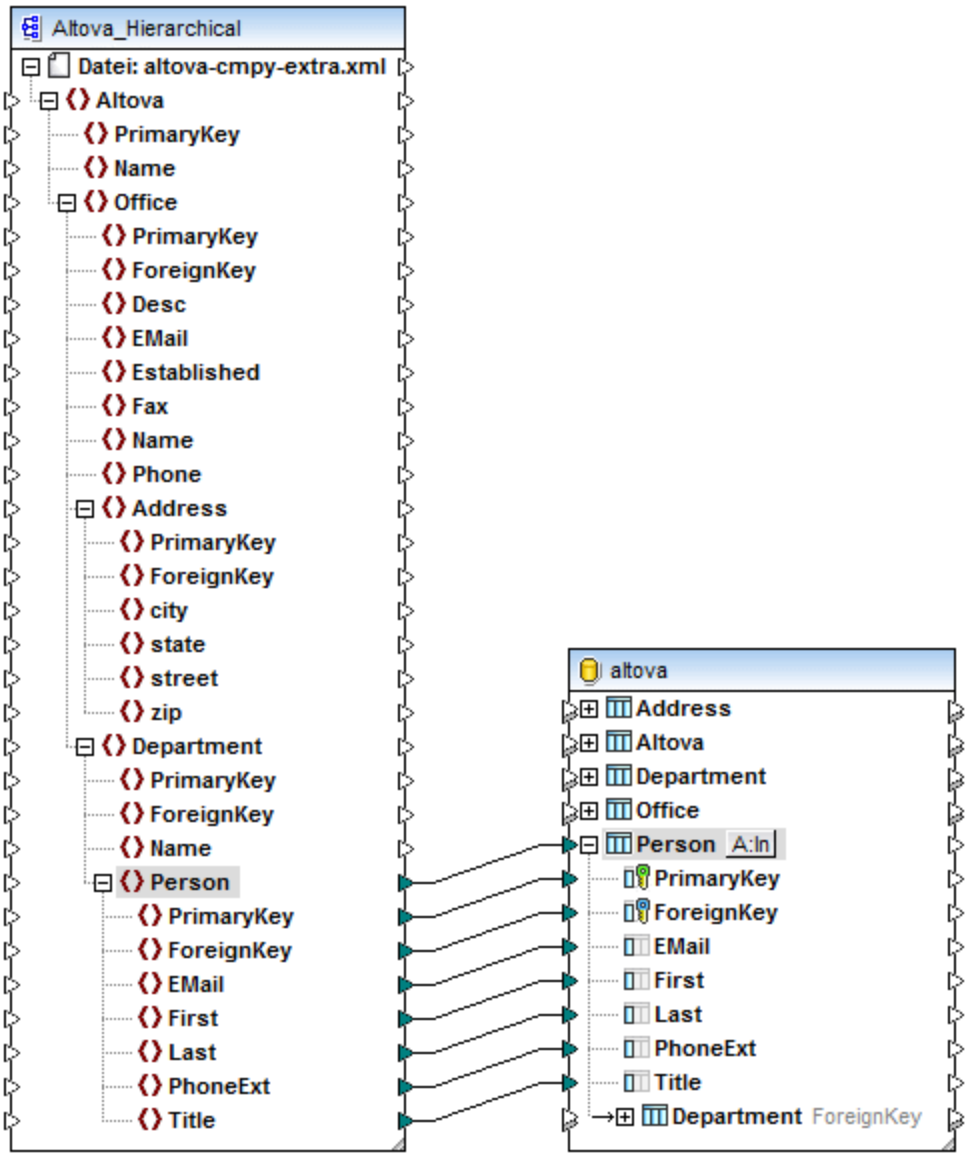
- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur Datei **Altova_Hierarchical.xsd**. Wenn Sie aufgefordert werden, eine Instanzdatei anzugeben, navigieren Sie zur Datei **altova-cmpy-extra.xml**.

Schritt 2: Fügen Sie die Zieldatenbank ein

- Klicken Sie im Menü **Einfügen** auf **Datenbank** und befolgen Sie die Anweisungen des Assistenten, um eine Verbindung zu **Altova.mdb** herzustellen (siehe Beispiel: Hinzufügen der Datenbank "altova.mdb" zum Mapping 257).

Schritt 3: Ziehen Sie die Verbindungen

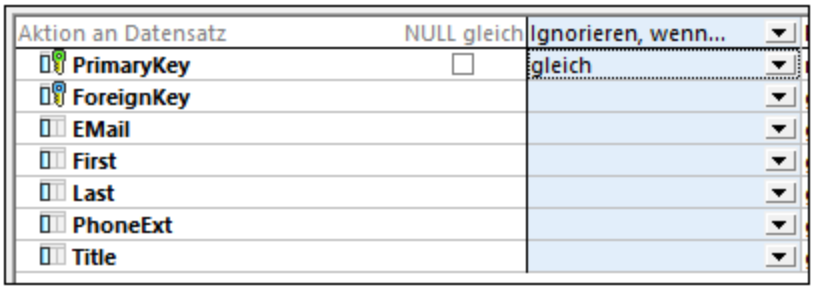
- Ziehen Sie die Mapping-Verbindungen wie unten gezeigt



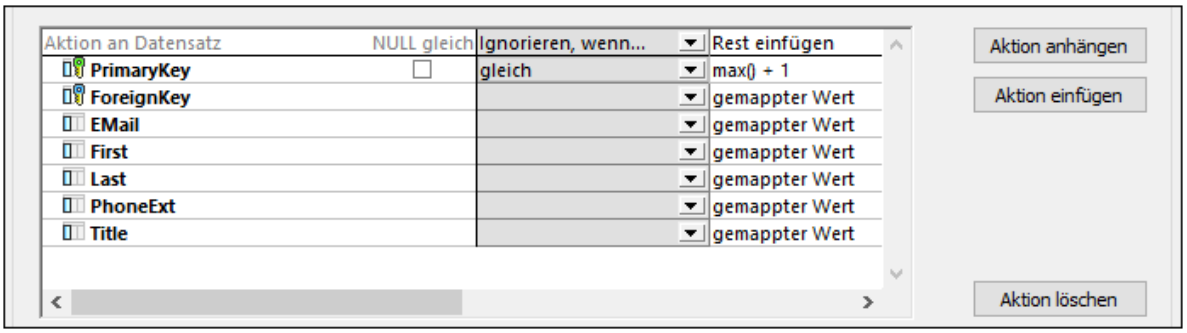
Schritt 4: Konfigurieren Sie die "Ignorieren, wenn... Rest einfügen"-Aktionen

1. Klicken Sie auf die Schaltfläche **Aktion: Einfügen** (A:In) neben der Tabelle "Person".
2. Wählen Sie neben **Aktion an Datensatz** den Eintrag **Ignorieren, wenn...** . Dadurch wird die Datenbankaktion in eine auf einer Bedingung basierende Ignorieren-Aktion geändert, d.h. der aktuelle Datensatz wird nur ignoriert, wenn eine Bedingung erfüllt wird (siehe nächster Schritt).
3. Wählen Sie neben **PrimaryKey** den Wert **gleich** aus, wie unten gezeigt. Damit wird die Ignorieren-Bedingung definiert, d.h. der Datenbankdatensatz wird nur ignoriert, wenn sein `PrimaryKey`-Wert gleich dem `PrimaryKey`-Wert aus dem Mapping ist.

© 2016-2022 Altova GmbH



4. Aktivieren Sie optional das Kontrollkästchen **NULL gleich**. In diesem Beispiel spielt das Kontrollkästchen keine Rolle, da weder das Feld **PrimaryKey** in der XML-Quelldatei noch das Feld **PrimaryKey** in der Datenbank Nullwerte enthält. Wenn Ihre Daten jedoch Nullwerte enthalten und solche Nullwerte als identisch behandelt werden sollen, sollten Sie dieses Kontrollkästchen aktivieren, da Sie sonst unerwünschte Ergebnisse erhalten könnten. Nähere Informationen dazu finden Sie unter Behandlung von Nullwerten in Datenbankaktionen 333.
5. Klicken Sie auf **Aktion anhängen**. Daraufhin wird rechts von der vorhandenen **Ignorieren, wenn**-Aktion eine neue Aktion hinzugefügt. Konfigurieren Sie die neue Aktion als **Rest einfügen**, wobei sie `PrimaryKey` wie unten gezeigt auf **max() + 1** setzen.



In der Abbildung oben wurden die Datenbankaktionen den Zielen des Mappings entsprechend konfiguriert, d.h. der Datensatz wird nur dann übersprungen, wenn die **Ignorieren, wenn..**-Bedingung zutrifft. Andernfalls wird der Datensatz eingefügt. Mit der Option "gemappter Wert" wird festgelegt, dass alle Felder des Datensatzes anhand der Werte aus dem Mapping befüllt werden. Mit der Option **max() + 1** wird ein neuer eindeutiger Primärschlüsselwert für den Datensatz generiert.

6. Klicken Sie auf **OK,** um das Dialogfeld zu schließen. Beachten Sie, dass im Mapping anstelle der Schaltfläche **Aktion: Einfügen** nun eine Schaltfläche **Aktion: Ignorieren; Einfügen** ( ) angezeigt wird. Dies zeigt an, dass für diese Tabelle sowohl eine Ignorieren- als auch eine Einfügeaktion konfiguriert wurde.

Schritt 5: Vorschau auf das Mapping und Aktualisieren der Datenbank

Klicken Sie auf das Register **Ausgabe,** um eine Vorschau auf das Mapping zu sehen. Es wird ein SQL-Skript generiert, das Aktionen enthält, die an der Datenbank ausgeführt werden sollen. Das Skript hat die Datenbank noch nicht geändert; dies dient nur zur Vorschau. So führen Sie das Skript an der Datenbank aus:

- Klicken Sie im Menü **Ausgabe** auf **SQL-Script ausführen.**

Anmerkung: Das Ausführen des SQL-Skripts direkt von MapForce aus ist nur eine der Methoden, um eine Datenbank zu aktualisieren. Siehe auch Ausführen von Mappings, die Datenbanken modifizieren 282.

Wenn Sie die Tabelle "Person" auf dem Register "DB-Abfrage" von MapForce öffnen (siehe Anzeigen und Abfragen von Datenbanken 388), sehen Sie als Ergebnis des Mappings Folgendes:

- Alle Datenbankdatensätze, für die es Primärschlüsselentsprechungen in der XML-Datei gab, haben die **Ignorieren, wenn..**-Bedingung erfüllt und sind unverändert geblieben. Beispiele dafür sind die Datensätze mit den Primärschlüsseln 1, 2, 3, 4 und 5.
- Alle Datenbankdatensätze, für die es in der XML-Datei keine Primärschlüsselentsprechungen gab, haben die **Ignorieren, wenn..**-Bedingung nicht erfüllt, sind aber unverändert geblieben. Beispiele dafür sind die Datensätze mit den Primärschlüsseln 6, 7, 8, 9, 10, 11, 12 und 13.
- Es wurden neue Datensätze in die Tabelle "Person" eingefügt (Datensätze, für die in der Datenbank noch kein Schlüssel vorhanden war). Beispiele dafür sind die Datensätze mit den Primärschlüsseln 30 und 31 in der XML-Quelldatei. Diese Datensätze wurden mit den neuen Primärschlüsseln 22 bzw. 23 in die Datenbank eingefügt.

	PrimaryKey	ForeignKey	EMail	First	Last	PhoneExt	Title
1	1	1	v.callaby@nanonull.com	Vernon	Callaby	582	Office Manager
2	2	1	f.further@nanonull.com	Frank	Further	471	Accounts Receivable
3	3	1	l.matise@nanonull.com	Loby	Matise	963	Accounting Manager
4	4	2	j.firstbread@nanonull.com	Joe	Firstbread	621	Marketing Manager Europe
5	5	2	s.sanna@nanonull.com	Susi	Sanna	753	Art Director
6	6	3	f.landis@nanonull.com	Fred	Landis	951	Program Manager
7	7	3	m.landis@nanonull.com	Michelle	Butler	654	Software Engineer
8	8	3	t.little@nanonull.com	Ted	Little	852	Software Engineer
9	9	3	a.way@nanonull.com	Ann	Way	951	Technical Writer
10	10	3	l.gardner@nanonull.com	Liz	Gardner	753	Software Engineer
11	11	3	p.smith@nanonull.com	Paul	Smith	334	Software Engineer
12	12	4	a.martin@nanonull.com	Alex	Martin	778	IT Manager
13	13	4	g.hammer@nanonull.com	George	Hammer	223	Web Developer
14	14	4	j.band@nanonull.com	Jessica	Bander	241	Support Engineer
15	15	4	l.king@nanonull.com	Lui	King	345	Support Engineer
16	16	5	s.meier@nanonull.com	Steve	Meier	114	Office Manager
17	17	5	t.bone@nanonull.com	Theo	Bone	331	Accounts Receivable
18	18	6	m.nafta@nanonull.com	Max	Nafta	122	PR & Marketing Manager US
19	19	7	v.bass@nanonull.com	Valentin	Bass	716	IT Manager
20	20	7	c.franken@nanonull.com	Carl	Franken	147	Support Engineer
21	21	7	m.redgreen@nanonull.com	Mark	Redgreen	152	Support Engineer
22	22	1	c.Cicada@microtech.com	Camilla	Cicada	765	HR
23	23	1	c.corrigan@microtech.com	Carol	Corrigan	629	Admin

Die Tabelle "Person" nach Aktualisierung der Datenbank

© 2016-2022 Altova GmbH

4.2.3.9 Bulk-Einfügungen (MapForce Server)

Mit der Option **Bulk Transfer verwenden** können Sie Daten aus einer MapForce-Komponente (TXT, CSV, DAT, usw.) sehr schnell in eine Datenbanktabelle einfügen. Auf diese Art können Sie das Einfügen von Daten enorm beschleunigen, da nur eine Anweisung anstatt mehrerer ausgeführt werden muss.

Die Option **Bulk Transfer verwenden** kann in MapForce, wie unten gezeigt, während des Mapping-Designs aktiviert werden. Ein Mapping, bei dem diese Option aktiviert ist, kann in MapForce ausgeführt werden, doch wird zu diesem Zeitpunkt keine Bulk-Einfügung angewendet. Der eigentliche Bulk-Transfer von Daten erfolgt, wenn das Mapping von MapForce Server ausgeführt wird.

Der Bulk Transfer wird unter folgenden Bedingungen unterstützt:

- Als Mapping-Transformationssprache muss BUILT-IN definiert sein. Nähere Informationen dazu finden Sie unter Transformationssprachen 28.
- Das Mapping wird von MapForce Server ausgeführt (entweder Standalone oder unter FlowForce Server-Verwaltung). Das bedeutet, dass das Mapping entweder zum .mfx-Format kompiliert oder auf FlowForce Server bereitgestellt werden muss. Nähere Informationen dazu finden Sie unter Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442 und Bereitstellen von Mappings auf FlowForce Server 1445.
- Die MapForce Server-Lizenz ist auf einem Mehrkernrechner nicht auf "Single Thread-Ausführung" eingeschränkt, d.h. das Kontrollkästchen **Limit to single thread execution** auf dem Register "Server Management" von Altova LicenseServer muss deaktiviert sein.
- Die Datenbankaktion ist "Alles einfügen", siehe auch Einstellungen für Datenbankaktionen 336.
- Es muss sich bei der Tabelle, in die die Daten im Bulk geladen werden, um ein Tabellenblatt, d.h. ein Blatt der untersten Hierarchieebene der Datenbank handeln. Es sollten keine Tabellen, Ansichten oder gespeicherten Prozeduren vorhanden sein, die die Tabelle im Mapping referenzieren.
- Der Datenbanktreiber unterstützt Bulk-Einfügungen bei WHERE-Bedingungen.

Die folgende Tabelle enthält einen Überblick über die Unterstützung von Bulk-Einfügungen, abhängig von der Datenbankart und dem verwendeten Treiber.

	ADO	ODBC	JDBC	ADO.NET	Nativ
Access	Nein	Nein	n/a	n/a	n/a
DB2	Nein	Ja	Ja	Ja	n/a
Firebird	n/a	Ja	Ja	Nein	n/a
Informix	Nein	Ja	Ja	Ja	n/a
iSeries	Nein	Ja	Ja	Ja	n/a
MariaDB	Nein	Ja	Ja	Nein	n/a
MySQL	n/a	Ja*	Ja	Nein	n/a
Oracle	Nein	Ja	Ja	Ja	n/a
PostgreSQL	n/a	Ja	Ja	n/a	Ja

	ADO	ODBC	JDBC	ADO.NET	Nativ
Progress	n/a	Ja	Ja	n/a	n/a
SQL Server	Ja	Ja	Ja	Ja	n/a
SQLite	n/a	n/a	n/a	n/a	Nein
Sybase	Nein	Ja	Ja	n/a	n/a
Teradata	n/a	Ja	Ja	n/a	n/a

** MySQL version 5 oder höher erforderlich*

Anmerkung: Die Unterstützung für die Bulk-Einfügung für MySQL und MariaDB über JDBC können Sie mit Hilfe der Verbindungsoption `rewriteBatchedStatements=true` aktivieren.

Beispiel für eine Bulk-Einfügung

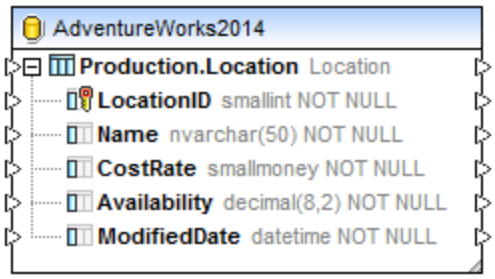
In diesem Beispiel wird gezeigt, wie Sie ein Mapping erstellen, das Daten aus der Beispieldatei **source.txt** im Bulk in eine Zieldatenbank lädt. Im Beispiel werden SQL Server 2014 und die AdventureWorks 2014-Datenbank verwendet. Letzere kann von der AdventureWorks-Beispielseite auf GitHub (https://github.com/Microsoft/sql-server-samples/releases/tag/adventureworks) heruntergeladen werden.

```
Location A,15.3,39
Location B,46,34
Location C,56.33,0
Location D,0,399
Location E,0,97.43
```

source.txt

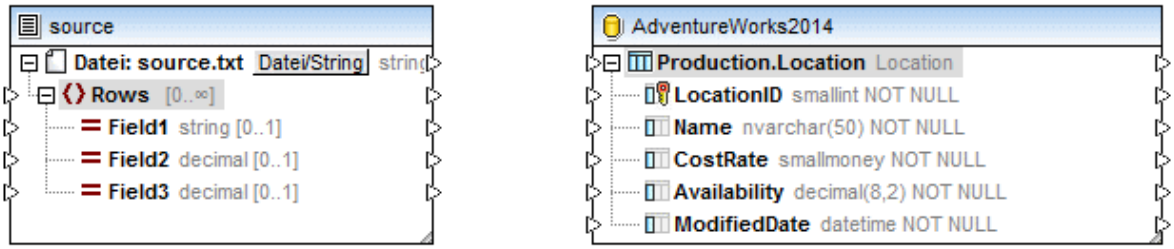
So definieren Sie eine Bulk-Einfügung:

1. Definieren Sie als Transformationssprache BUILT-IN ( ).
2. Stellen Sie eine Verbindung zur Datenbank AdventureWorks 2014 her und fügen Sie die Tabelle "Production.Location" zum Mapping hinzu. Nähere Informationen dazu finden Sie unter Hinzufügen von Datenbanken zum Mapping 256.

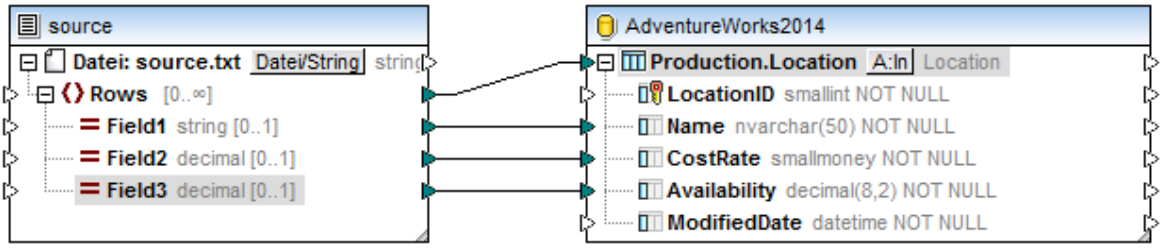


© 2016-2022 Altova GmbH

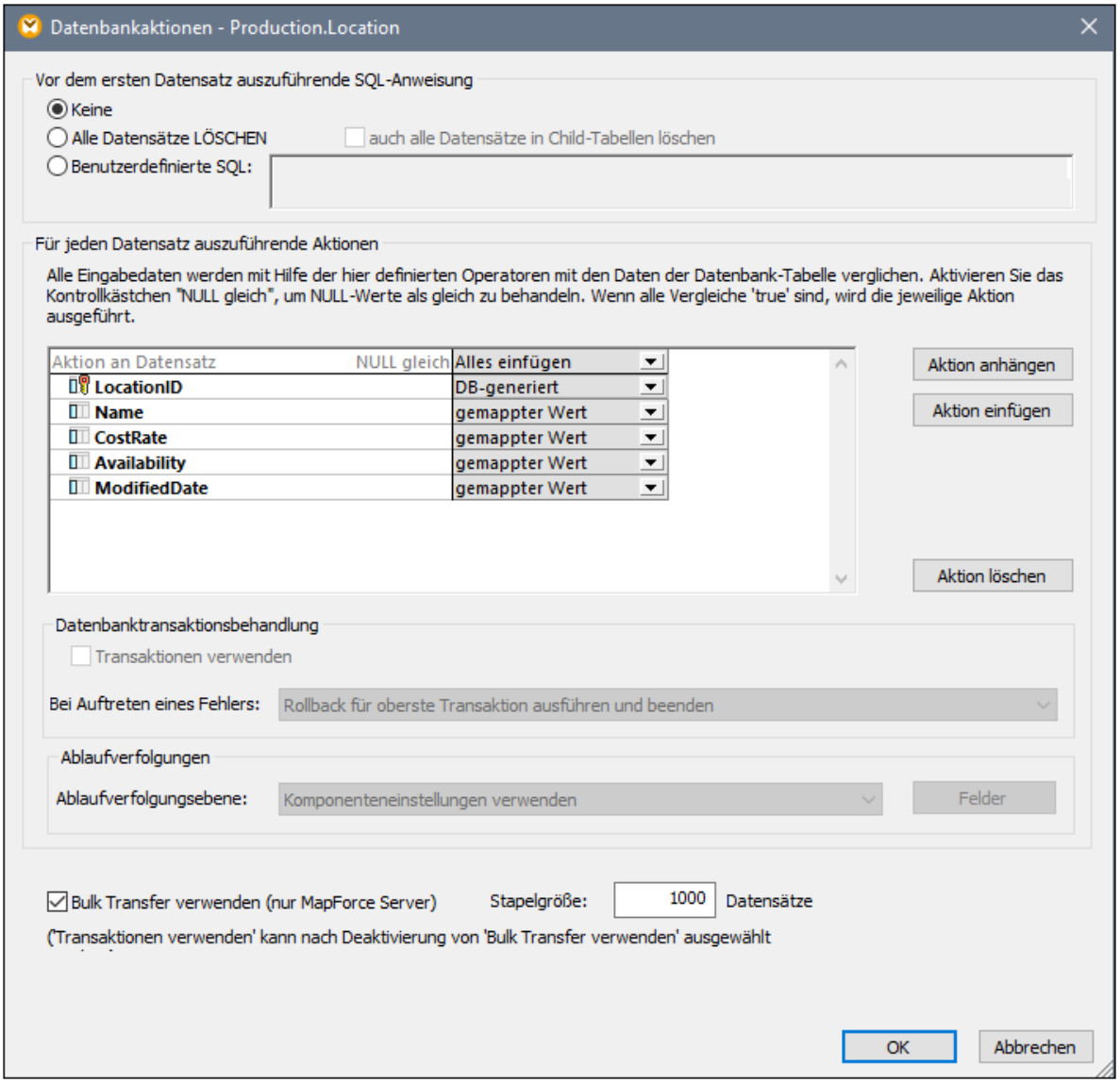
3. Klicken Sie im Menü **Einfügen** auf **Text** und fügen Sie als Quelle eine Textdatei (wie die Beispieldatei **source.txt** oben) zum Mapping hinzu. Nähere Informationen finden Sie unter CSV- und Textdateien 585. Stellen Sie sicher, dass die Datentypen von Quell- und Zielkomponente kompatibel sind. Die Datentypen sind in der Komponente sichtbar, wenn Sie die Symbolleisten-Schaltfläche **Datentypen anzeigen** () aktivieren.



4. Ziehen Sie die Mapping-Verbindungen wie unten gezeigt. Beachten Sie, dass nun die **Datenbankaktionsschaltfläche** A:In rechts vom Tabellennamen angezeigt wird.



5. Klicken Sie auf die **Datenbankaktionsschaltfläche** (A:In), aktivieren Sie das Kontrollkästchen **Bulk Transfer verwenden** und klicken Sie auf OK.



Beachten Sie, dass als "Aktion an Datensatz" im Dialogfeld oben "Alles einfügen" ausgewählt ist. Im Feld **Stapelgröße** wird die Anzahl der Datensätze definiert, die pro Aktion eingefügt werden.

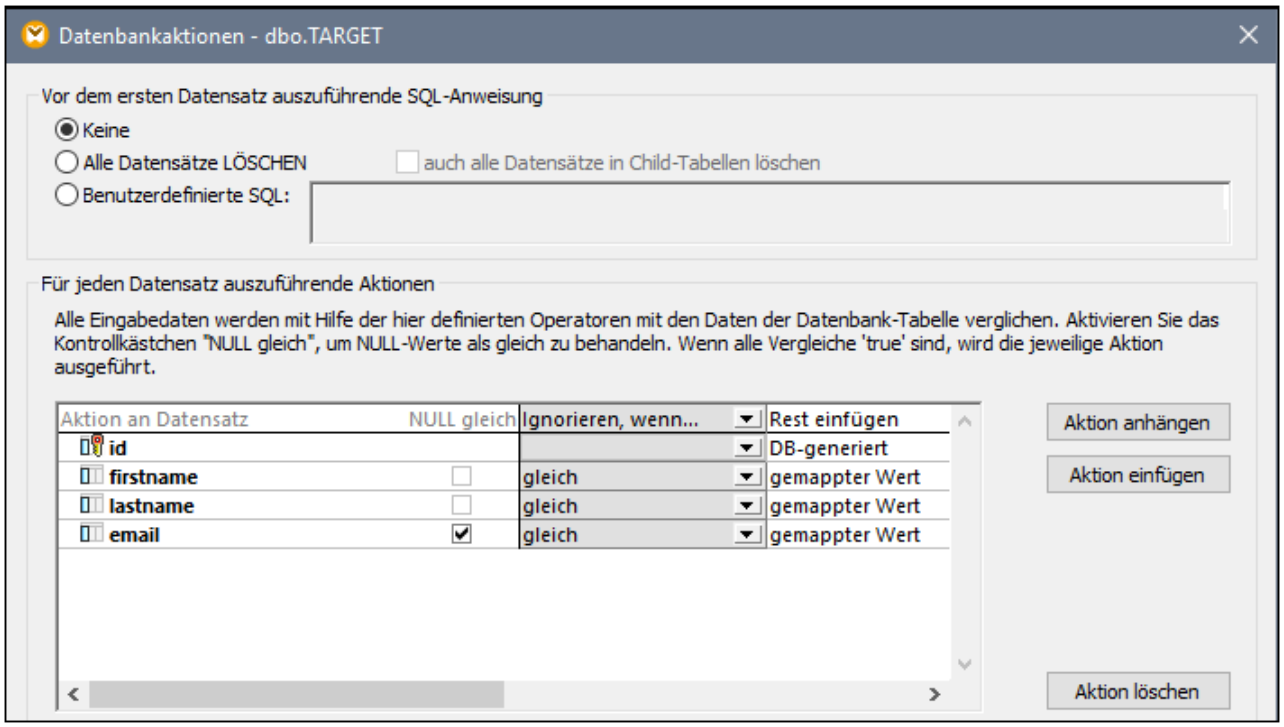
Anmerkung: Wenn die Option **Bulk Transfer verwenden** aktiv ist, wird die Option **Transaktionen verwenden** deaktiviert und umgekehrt. Wenn Sie die Transaktionsverarbeitung aktivieren möchten, klicken Sie auf das Kontrollkästchen **Bulk Transfer verwenden**, um es zu deaktivieren.

Nachdem Sie die Bulk-Einfügung nun aktiviert haben, ist der nächste Schritt die Ausführung des Mappings mit MapForce Server (entweder Standalone Edition oder unter Verwaltung von FlowForce Server). Nähere Informationen dazu finden Sie unter Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442 und Bereitstellen von Mappings auf FlowForce Server 1445.

 © 2016-2022 Altova GmbH

4.2.3.10 Behandlung von Nullwerten in Datenbankaktionen

Wenn in einem Mapping eine Zieldatenbank mit Hilfe von Tabellenaktionen wie "Ignorieren, wenn", "Aktualisieren, wenn", "Löschen, wenn", aktualisiert wird, so vergleicht MapForce die Quelldaten mit den Zieldaten und generiert als Ergebnis interne Datenbankaktualisierungsabfragen. (Eine Vorschau auf diese internen Abfragen steht im **Ausgabefenster** von MapForce zur Verfügung, siehe Ausführen von Mappings, die Datenbanken modifizieren 282). In den generierten Abfragen werden die Bedingungen, die im Dialogfeld "Datenbankaktionen" definiert wurden, berücksichtigt.



Dialogfeld "Datenbankaktionen"

Wenn Ihre Quell- oder Zieldaten Null-fähige Felder enthalten, haben Sie zwei Optionen, um Nullwerte aus der Quellkomponente mit denen in der Zielkomponente zu vergleichen:

1. Behandlung von Nullwerten als gleich
2. Behandlung von Nullwerte als nicht gleich

Zu diesem Zweck steht im oben gezeigten Dialogfeld neben einigen Feldern ein Kontrollkästchen **NULL gleich** zur Verfügung. Ob Sie dieses Kontrollkästchen aktivieren oder nicht aktivieren, kann sich auf das Mapping-Ergebnis auswirken und ist Thema dieses Kapitels. Beachten Sie, dass das Kontrollkästchen nur für nullfähige Felder und nur, wenn mindestens eine Tabellenaktion die Bedingung "gleich" oder "gleich (Groß/klein ignorieren)" aufweist.

Um unerwünschte Ergebnisse zu vermeiden, sollten Sie das Kontrollkästchen **NULL gleich** aktivieren, wenn alle der folgenden Bedingungen zutreffen:

1. Das Dialogfeld "Datenbankaktionen" enthält "Ignorieren, wenn", "Aktualisieren, wenn", "Löschen, wenn"-Aktionen und

2. Diese Aktionen werden an Datensätzen ausgeführt, die Nullwerte enthalten können und
3. Nullwerte in der Quellkomponente müssen in Ihrem Mapping als gleich Nullwerten in der Zielkomponente behandelt werden.

Standardmäßig ist das Kontrollkästchen **NULL gleich** deaktiviert. Wenn die obigen Bedingungen zutreffen und das Kontrollkästchen deaktiviert ist, wird die Datenbankzieltabelle eventuell nicht, wie erwartet, aktualisiert (z.B. werden mehr Zeilen eingefügt als nötig). Der Grund dafür ist, dass sich Nullwerte auf den Datenvergleich auswirken. Angenommen ein Datensatz in der Abbildung oben hat sowohl in der Datenquelle als auch im Datenziel eine null **email**. Wenn Sie das Kontrollkästchen **NULL gleich** aktivieren, erfüllt der Datensatz die Bedingung **Ignorieren, wenn...** und wird ignoriert (übersprungen). Wenn Sie das Kontrollkästchen **NULL gleich** aber nicht aktivieren, erfüllt der Datensatz die Bedingung **Ignorieren, wenn...** nicht mehr und wird in die Datenbank eingefügt.

Beispiel

Um die Nullfähigkeit in Mappings besser zu verstehen, werfen wir einen Blick auf ein Beispiel, in dem Nulldaten verglichen werden. In diesem Beispiel wird eine Microsoft SQL Server-Datenbank verwendet; es gilt jedoch auch für jeden anderen unterstützten Datenbanktyp. Wenn Sie Microsoft SQL Server installiert haben, haben Sie auch die Möglichkeit, die in diesem Beispiel verwendeten Tabellen und Daten durch Ausführung des folgenden Datenbankskripts zu erstellen:
<Dokumente>\Altova\MapForce2023\MapForceExamples\CreateNullableFields.sql.

Unten sind die verwendeten Datenbanktabellen abgebildet. In beiden Tabellen sind Personendaten gespeichert. Beide Tabellen haben die gleichen Spalten. Außerdem kann die Spalte `email` in beiden Tabellen Nullwerte (also keine Daten) enthalten.

```
+----+-----------+-----------+-------------------------+
| id | firstname | lastname  | email                   |
+----+-----------+-----------+-------------------------+
| 1  | Toby      | Hughey    | t.hughey@nanonull.com   |
| 2  | Mia       | Dahill    | NULL                    |
| 3  | Fred      | Weinstein | f.weinstein@nanonull.com |
+----+-----------+-----------+-------------------------+
```

Die Quelltabelle SOURCE

```
+----+-----------+-----------+-------------------------+
| id | firstname | lastname  | email                   |
+----+-----------+-----------+-------------------------+
| 1  | Mia       | Dahill    | NULL                    |
| 2  | Fred      | Weinstein | f.weinstein@nanonull.com |
+----+-----------+-----------+-------------------------+
```

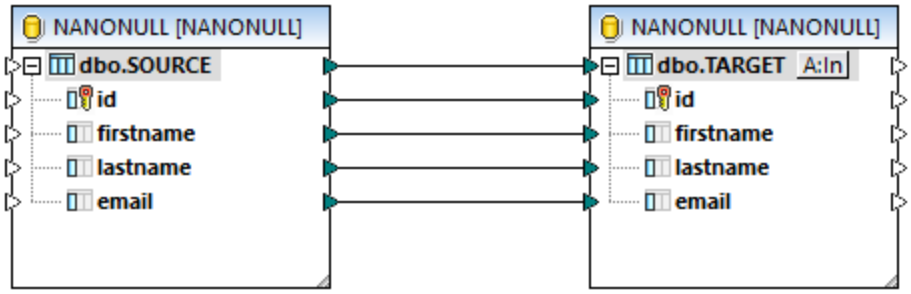
Die Zieltabelle TARGET

Angenommen Sie möchten Daten aus der Tabelle `SOURCE` in der Tabelle `TARGET` zusammenführen. Nur die neuen Datensätze (in diesem Beispiel "Tobie Hughey") dürfen in die Tabelle `TARGET` eingefügt werden. Datensätze, die in beiden Tabellen vorhanden sind ("Mia Dahill" und "Fred Weinstein") müssen ignoriert werden.

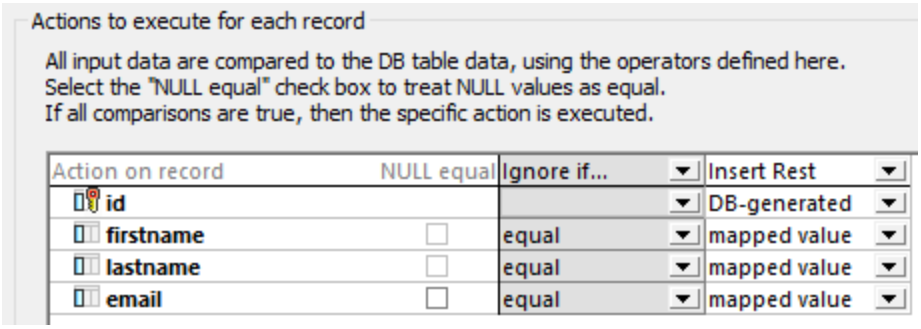
Gehen Sie dazu folgendermaßen vor:

© 2016-2022 Altova GmbH

1. Wählen Sie im Menü **Einfügen** den Befehl **Datenbank**. Befolgen Sie die Anweisungen des Assistenten, um die Datenbankverbindung herzustellen (siehe auch Herstellen einer Verbindung zu einer Datenbank 167). Wenn Sie aufgefordert werden, Datenbankobjekte hinzuzufügen, wählen Sie die Tabelle `SOURCE` aus.
2. Wählen Sie im Menü **Einfügen** den Befehl **Datenbank**. Stellen Sie erneut eine Verbindung zur Datenbank her und fügen Sie die Tabelle `TARGET` zum Mapping hinzu.
3. Ziehen Sie die Mapping-Verbindungen zwischen der Quell- und der Zielkomponente.



4. Klicken Sie auf die Schaltfläche **Aktion:Einfügen** A:In und konfigurieren Sie die Datenbankaktionen wie folgt:



Wie in der Abbildung gezeigt, ist eine Kombination aus "Ignorieren, wenn.. Rest einfügen"-Aktionen definiert. Dies bedeutet, dass im Mapping für jeden Datensatz überprüft wird, ob:

- `firstname` in der Quellkomponente gleich `firstname` in der Zielkomponente ist UND
- `lastname` in der Quellkomponente gleich `lastname` in der Zielkomponente ist UND
- `email` in der Quellkomponente gleich `email` in der Zielkomponente ist.

Wenn alle oben definierten Bedingungen zutreffen, so wird der Datensatz (wie erforderlich) ignoriert. Andernfalls wird ein neuer Datensatz in die Zieltabelle eingefügt. Die `id` des neuen Datensatzes wird von der Datenbank generiert, während die anderen Felder (`firstname`, `lastname`, `email`) mit den aus der Quellkomponente gemappten Werten befüllt werden.

Beachten Sie, dass das Kontrollkästchen neben `email` dazu dient, den NULL-fähigen Vergleich für dieses Feld zu aktivieren bzw. zu deaktivieren. Dieses Kontrollkästchen muss aktiviert sein, da `email` Nullwerte enthalten kann ("Mia Dahill" hat eine Null-E-Mail-Adresse). Um zu sehen, wie sich dieses Kontrollkästchen auf das Ergebnis auswirkt, versuchen Sie, die Datenbank zwei Mal zu aktualisieren: das erste Mal bei aktiviertem Kontrollkästchen und ein zweites Mal bei deaktiviertem Kontrollkästchen.

Um die Datenbank zu aktualisieren, klicken Sie auf das Register **Ausgabe** und führen Sie den Menübefehl **Ausgabe | SQL-Script ausführen** aus.

Wenn das Kontrollkästchen aktiviert ist, wurde MapForce explizit darüber informiert, dass die Nullwerte als identisch zu behandeln sind. Daher wird der Datensatz "Mia Dahill", wie gewünscht, nicht in die Zieltabelle eingefügt.

Wenn das Kontrollkästchen nicht aktiviert ist, wird der Datensatz "Mia Dahill" in die Zieltabelle eingefügt (obwohl er dort bereits vorhanden ist), was nicht das beabsichtigte Ergebnis ist. Der Grund dafür ist, dass keine explizite Angabe darüber gemacht wurde, dass Nullwerte als gleich zu behandeln sind. Eine ähnliche Situation würde auftreten, wenn Sie die folgende Abfrage an der Datenbank ausführen würden (mit dieser Abfrage werden keine Datensätze abgerufen, da der Nullwerte mit dem "=" Operator verglichen wird, daher ist sie nicht NULL-fähig):

```
SELECT firstname, lastname, email FROM TARGET WHERE firstname = 'Mia' AND lastname =
'Dahill' AND email = NULL;
```

Damit NULL-Werte berücksichtigt werden, müsste die Abfrage folgendermaßen lauten:

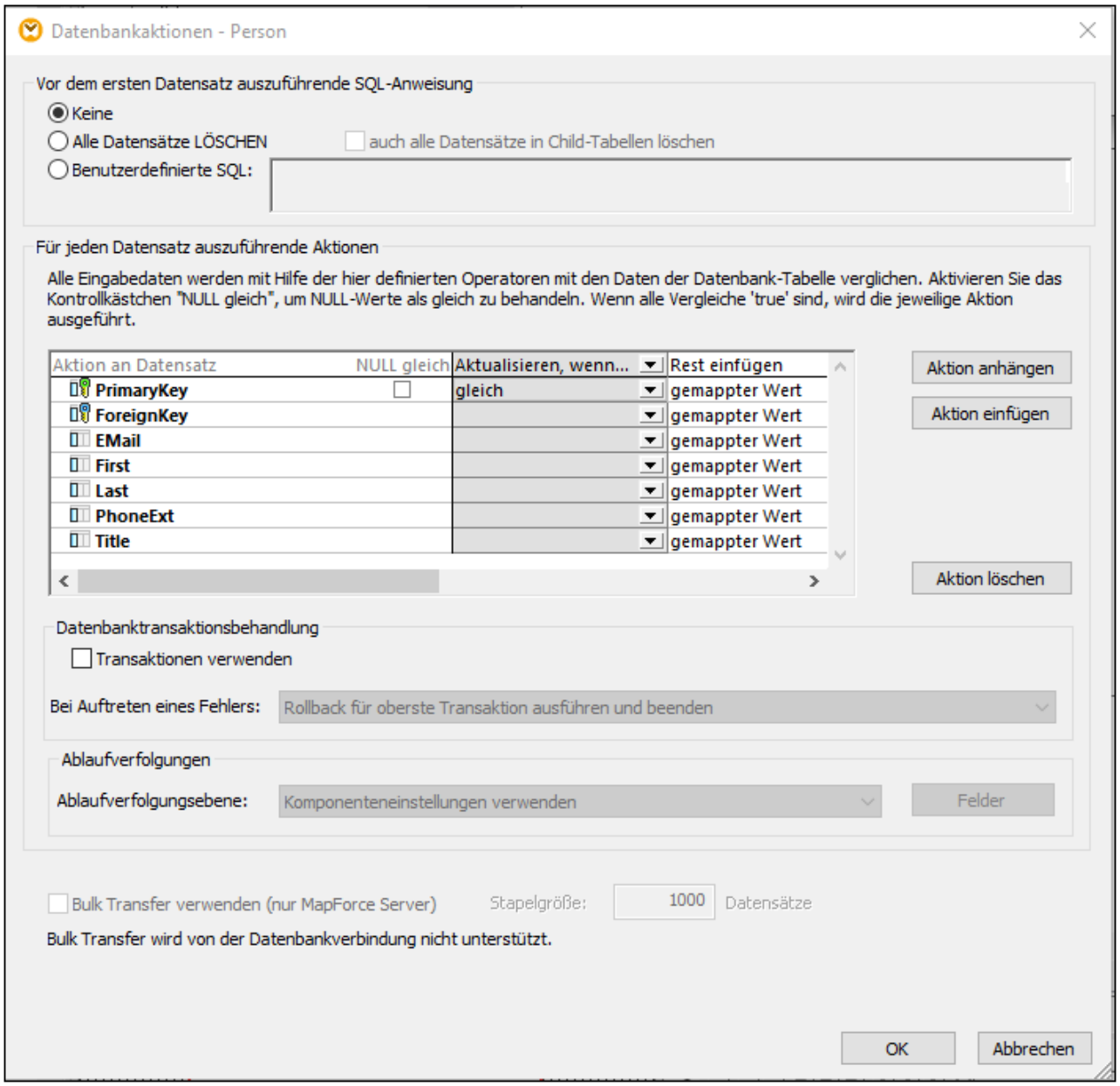
```
SELECT firstname, lastname, email FROM TARGET WHERE firstname = 'Mia' AND lastname =
'Dahill' AND email IS NULL;
```

Anmerkung: Die oben gezeigten Abfragen dienen nur als Beispiel und zeigen nicht die tatsächliche Syntax von internen, von MapForce generierten Abfragen. MapForce passt die Syntax der generierten Abfragen an den Datenbanktyp an (da verschiedene Datenbankanbieter unterschiedliche Methoden zur Behandlung von Null-Vergleichen verwenden).

4.2.3.11 Einstellungen für Datenbankaktionen

Wenn Sie eine Mapping-Verbindung zu einer Datenbanktabelle erstellen, wird neben der betroffenen Tabelle eine Datenbankaktionsschaltfläche angezeigt. Wenn Sie darauf klicken, wird das Dialogfeld "Datenbankaktionen" aufgerufen, in dem Sie die Einfüge-, Aktualisierungs- und Löschaktionen für die Datenbank sowie andere Optionen konfigurieren können.

© 2016-2022 Altova GmbH



Dialogfeld "Datenbankaktionen"

Im Folgenden finden Sie eine Beschreibung der Einstellungen, die Sie im Dialogfeld "Datenbankaktionen" vornehmen können.

Vor dem ersten Datensatz auszuführende SQL-Anweisung

In diesem Gruppenbereich können Sie SQL-Anweisungen definieren, die vor unter **Für jeden Datensatz auszuführende Aktionen** definierten Aktionen ausgeführt werden sollen. Aktivieren Sie das gewünschte Optionsfeld:

- **Keine** - Es wird keine Aktion ausgeführt. Dies ist die Standardeinstellung.

- **Alle Datensätze LÖSCHEN** - Alle Datensätze aus der ausgewählten Tabelle werden gelöscht, bevor eine im Gruppenbereich **Für jeden Datensatz auszuführende Aktionen** definierte Aktion ausgeführt wird. Aktivieren Sie das Kontrollkästchen **auch alle Datensätze in Child-Tabellen löschen,** wenn auch die in Child-Tabellen der ausgewählten Tabelle gespeicherten Daten entfernt werden sollen. Ein Beispiel dafür finden Sie unter Einfügen von Daten in mehrere verknüpfte Tabellen 295.
- **Benutzerdefinierte SQL** - Schreiben Sie eine benutzerdefinierte SQL-Anweisung, die sich auf die gesamte Tabelle auswirkt. Beachten Sie, dass die Unterstützung mehrerer SQL-Anweisungen in einer Abfrage von der Datenbank, der Verbindungsmethode und dem verwendeten Treiber abhängt.

Für jeden Datensatz auszuführende Aktionen

In dieser Gruppe von Einstellungen werden die Datenbankaktionen, die an dieser Tabelle bei Ausführung des Mappings ausgeführt werden sollen, definiert. Um Tabellenaktionen zu verwalten, klicken Sie auf die Schaltflächen **Aktion anhängen, Aktion einfügen** oder **Aktion löschen.** Falls erforderlich, können auch mehrere Aktionen definiert werden (z.B. eine "Aktualisieren, wenn..."-Aktion, gefolgt von einer "Rest einfügen"-Aktion).

Die definierten Tabellenaktionen werden von links nach rechts verarbeitet. Im oben gezeigten Beispiel wird zuerst die Aktion "Aktualisieren, wenn..." verarbeitet. Wenn die Aktualisierungsbedingung nicht erfüllt wird, wird die darauf folgende Aktion verarbeitet (in diesem Beispiel die Aktion "Rest einfügen"). Beachten Sie die folgenden Punkte:

- Alle definierten Bedingungen einer Aktion müssen erfüllt werden, damit die Datenbankaktion ausgeführt wird. Wenn dies der Fall ist, werden alle Felder, die im Mapping zwischen der Quell- und Zielkomponente miteinander verbunden sind, aktualisiert. Alle nachfolgenden Datenbankaktionen (rechts von einer Aktion, deren Bedingungen erfüllt wurden), werden für diesen Datensatz ignoriert.
- Wenn die definierte Bedingung nicht erfüllt wird, so wird die Datenbankaktion übersprungen und die nächste Aktion (rechts davon) wird verarbeitet.
- Wenn keine der Bedingungen erfüllt wird, werden keine Datenbankaktionen ausgeführt.

Datenbankaktionen, die nach "Alles einfügen" oder "Rest einfügen" definiert sind, werden nie ausgeführt, da es für Einfügeaktionen keine Spaltenbedingungen gibt. Wenn dies der Fall ist, wird ein Dialogfeld angezeigt, dass die nachfolgenden Spalten mit Datenbankaktionen gelöscht werden.

Mit dem Kontrollkästchen **NULL gleich,** das neben den einzelnen Datensätzen, auf die es anwendbar ist, angezeigt wird, weisen Sie MapForce an, Nullwerte im Quelldatensatz als gleich Nullwerten im Zieldatensatz zu behandeln. Wenn Sie dieses Kontrollkästchen nicht aktivieren, erhalten Sie unter Umständen unerwünschte Ergebnisse, siehe Behandlung von Nullwerten in Datenbankaktionen 333).

Wenn mit dem Mapping eine Tabelle aktualisiert wird, die Sekundärschlüsselbeziehungen zu anderen Tabellen hat, können die folgenden Optionen verwendet werden:

Daten in Child-Tabellen löschen	Diese Option ist sinnvoll, wenn Sie die Aktion "Aktualisieren, wenn.." für eine Parent-Tabelle auswählen. Sie ist eventuell erforderlich, wenn die Anzahl der Datensätze in der Quelldatei nicht mit der Anzahl der Datensätze in der Zieldatenbank übereinstimmt und Sie die Datenbank damit synchronisieren möchten (keine verwaisten Child-Tabellen). Siehe auch Optionen für Child-Tabellen bei Aktualisierung einer Parent-Tabelle 312.

 © 2016-2022 Altova GmbH

Input-Child-Daten ignorieren	Verwenden Sie diese Option, wenn Sie eine übergeordnete Zieltabelle aktualisieren möchten, ohne dass davon Child-Tabellen/Datensätze dieser Tabelle betroffen sein sollen. Siehe auch Optionen für Child-Tabellen bei Aktualisierung einer Parent-Tabelle 312.

Beispiele für die verschiedenen Kombinationen von Aktionen finden Sie in den folgenden Kapiteln:

- Einfügen von Daten in eine Tabelle 292
- Einfügen von Daten in mehrere verknüpfte Tabellen 295
- Aktualisieren einer Tabelle 300
- Optionen für Child-Tabellen bei Aktualisierung einer Parent-Tabelle 312
- Die Aktion "Aktualisieren, wenn...Rest einfügen" 303
- Die Aktion "Löschen, wenn..." 320
- Die Aktion "Ignorieren, wenn..." 324

Transaktionen verwenden

Aktiviert für diese spezielle Datenbankaktion Datenbanktransaktionen. Nähere Informationen dazu finden Sie unter Behandlung von Datenbankausnahmen 429.

Bulk Transfer verwenden

Aktiviert den Bulk Transfer (mehrere INSERT-Anweisungen in einer einzigen Abfrage). Bulk Transfer wird unterstützt, wenn das Mapping mit MapForce Server ausgeführt wird und wenn als Datenbankaktion "Alles einfügen" definiert ist. Nähere Informationen dazu finden Sie unter Bulk-Einfügungen (MapForce Server) 329.

4.2.3.12 Beispiel: Mappen von Daten von XML auf SQLite

In diesem Beispiel wird Schritt für Schritt beschrieben, wie Sie ein MapForce Mapping erstellen, das Daten aus einer XML-Datei liest und diese in eine SQLite-Datenbank schreibt. Zu diesem Beispiel gibt es eine Mapping-Beispieldesigndatei (.mfd-Datei). Wenn Sie die Datei vor Beginn dieses Beispiels öffnen möchten, finden Sie die Datei unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2023\MapForceExamples\XMLtoSQLite.mfd.

Ziel des Beispiels ist es, Daten aus einer XML-Datei in eine SQLite-Datenbank einzufügen. Dazu benötigen Sie eine leere SQLite-Datenbank, in die die Daten geschrieben werden. Wie unten gezeigt, können Sie die SQLite-Datenbank entweder mit Altova DatabaseSpy erstellen und anzeigen oder über die Befehlszeilen-Shell, die Sie von der offiziellen SQLite-Website herunterladen können.

So erstellen Sie die SQLite-Datenbank:

Wenn DatabaseSpy auf Ihrem Rechner (entweder als Standalone-Produkt oder als Teil des Altova MissionKit) installiert ist, können Sie die neue SQLite-Datenbank folgendermaßen erstellen:

1. Starten Sie DatabaseSpy.
2. Klicken Sie im Menü **Datei** auf **Datenbankverbindung erstellen**.
3. Klicken Sie auf **Verbindungsassistent** und anschließend auf**SQLite**.

4. Klicken Sie auf **Neue SQLite-Datenbank erstellen**, geben Sie als Pfad **c:\sqlite\articles.sqlite** ein und klicken Sie auf **Verbinden**.
5. Wenn Sie aufgefordert werden, einen Datenquellennamen zu definieren, lassen Sie den Standardnamen unverändert.
6. Öffnen Sie einen neuen SQL Editor (**Strg+N**) und führen Sie an der Datenbank die folgende Abfrage aus:

```
create table articles (number smallint, name varchar(10), singleprice real);
```

Gehen Sie andernfalls folgendermaßen vor, um die Datenbank zu erstellen:

1. Laden Sie die SQLite Command-Line Shell für Windows von der SQLite Download-Seite (http://www.sqlite.org/download.html) herunter und entpacken Sie das .zip-Archiv in ein Verzeichnis auf Ihrem lokalen Rechner (in diesem Beispiel in **c:\sqlite**).
2. Führen Sie **c:\sqlite\sqlite3.exe** aus und geben Sie die folgende Anweisung ein:

```
create table articles (number smallint, name varchar(10), singleprice real);
```

   Dadurch wird die Tabelle `articles` in der Arbeitsspeicherdatenbank erstellt. Die Tabelle `articles` besteht aus drei Spalten: *number*, *name* und *singleprice*. In diesen Spalten sollen die Daten aus den gleichnamigen Elementen aus dem XML-Quellschema gespeichert werden. Für jede Spalte ist ein für die jeweiligen Daten in der Spalte geeigneter Datentyp deklariert.
3. Führen Sie den folgenden Befehl aus:

```
.save articles.sqlite
```

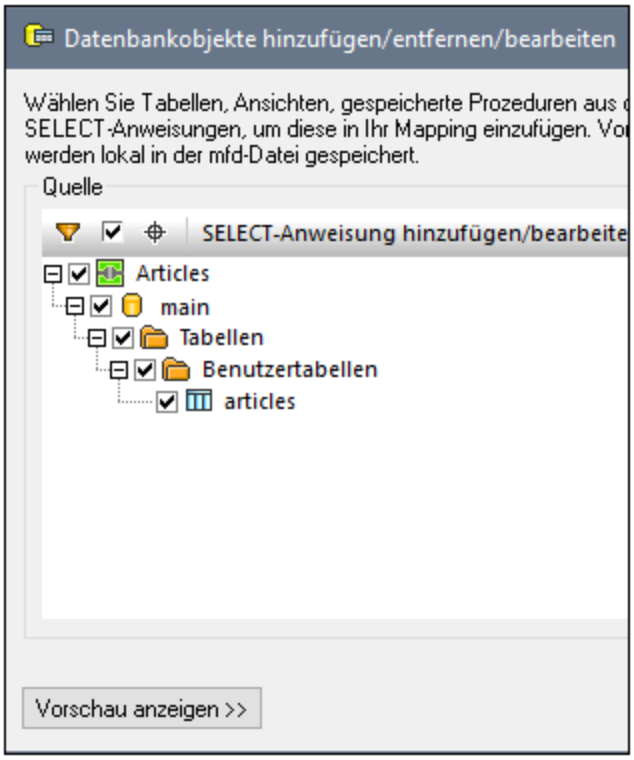
   Daraufhin wird die Datenbank im Arbeitsspeicher unter dem aktuellen Arbeitspfad **c:\sqlite\Articles.db** gespeichert. Beachten Sie, dass Sie diesen Pfad später referenzieren müssen.

Sie haben die für dieses Beispiel benötigte SQLite-Beispieldatenbank nun fertig erstellt.

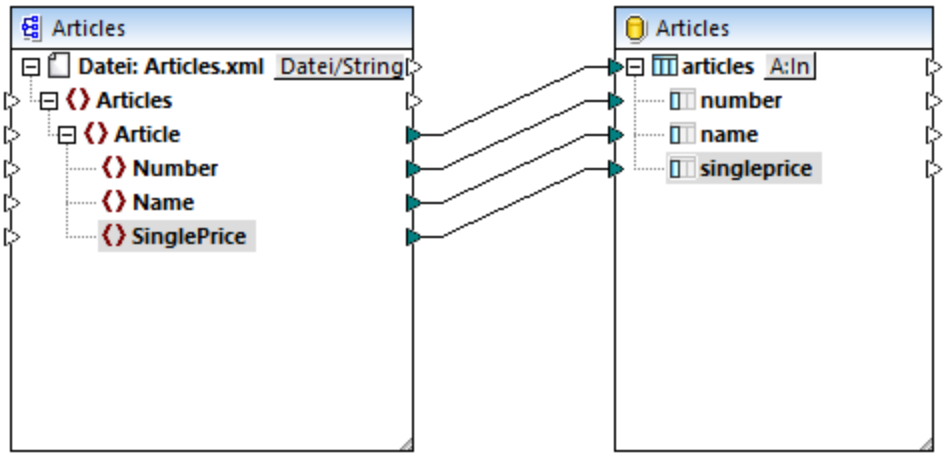
So erstellen Sie das XML-auf-SQLite-Mapping-Design:

1. Starten Sie MapForce und stellen Sie sicher, dass als Transformationssprache BUILT-IN (Menübefehl **Ausgabe | Built-in Ausführungsprozessor**) ausgewählt ist.
2. Fügen Sie die Datei **Articles.xml** aus dem Ordner **<Documents>\Altova\MapForce2023\MapForceExamples\Tutorial** zum Mapping-Bereich hinzu (Menübefehl **Einfügen | XML-Schema/Datei einfügen**).
3. Fügen Sie die zuvor erstellte Datenbank **Articles.sqlite** aus dem Ordner zum Mapping-Bereich hinzu (Menübefehl **Einfügen | Datenbank einfügen**) und wählen Sie anschließend **SQLite** aus.
4. Klicken Sie auf **Verbinden**. Wenn Sie aufgefordert werden, die Datenbankobjekte auszuwählen, wählen Sie die Tabelle `articles` aus.

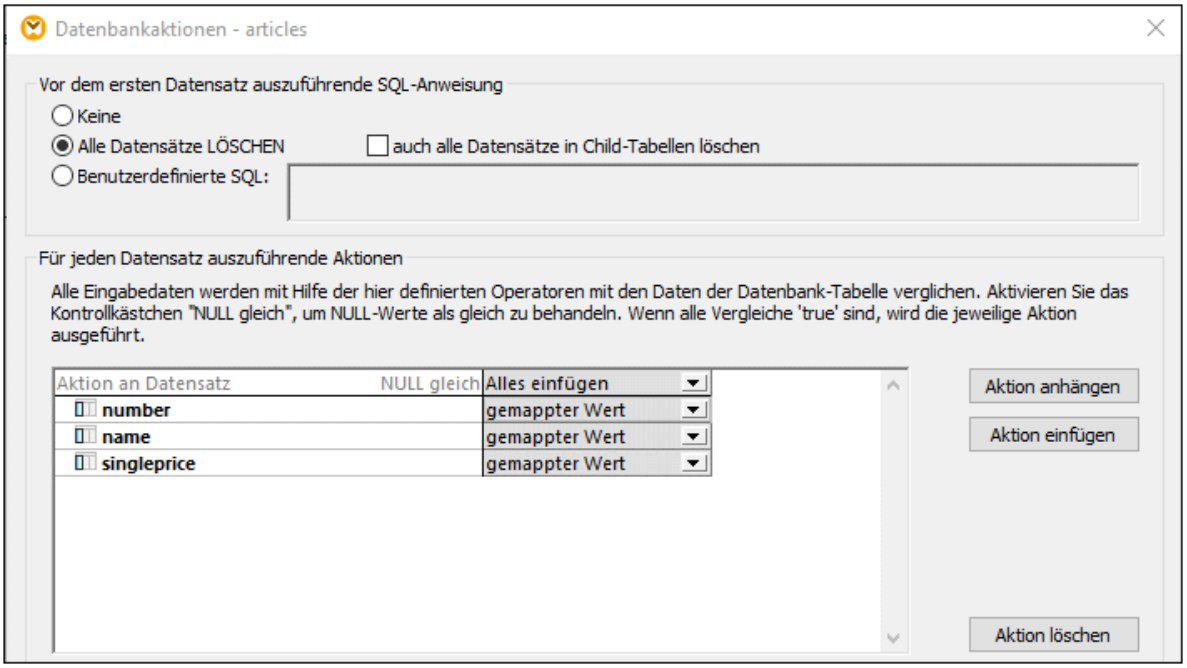
© 2016-2022 Altova GmbH



5. Erstellen Sie die folgenden Verbindungen:



6. Klicken Sie auf die **A:In**-Schaltfläche der Datenbankkomponente und wählen Sie die Option **Alle Datensätze löschen** aus. Damit stellen Sie sicher, dass bei jeder Ausführung des Mappings zuerst alle vorhandenen Datenbankzeilen gelöscht werden, damit es zu keinen doppelt vorhandenen Zeilen kommt.



7. Klicken Sie im Hauptmappingfenster auf das Register **Ausgabe**. MapForce führt das Mapping mit dem Built-in-Ausführungsprozessor aus und zeigt die erstellte SQL-Abfrage im Ausgabefenster an.

```
1   /*
2   The following SQL statements are only for preview and may not be executed in another SQL query tool!
3   To execute these statements use function "Run SQL-script" from menu "Output".
4   Connect to database using the following connection-string:
5   C:/sqlite/articles.sqlite
6   */
7
8   PRAGMA foreign_keys = ON;
9
10  DELETE FROM "articles"
11
12  INSERT INTO "articles" ("number", "name", "singleprice") VALUES (1, 'T-Shirt', 25)
13
14  INSERT INTO "articles" ("number", "name", "singleprice") VALUES (2, 'Socks', 2.3)
15
16  INSERT INTO "articles" ("number", "name", "singleprice") VALUES (3, 'Pants', 34)
17
18  INSERT INTO "articles" ("number", "name", "singleprice") VALUES (4, 'Jacket', 57.5)
19
```

8. Führen Sie das SQL-Script aus, um die Datenbank zu befüllen (Menübefehl **Ausgabe | SQL-Script ausführen)**. Falls keine Laufzeitfehler auftreten, werden die Datensätze in die SQLite-Datenbank eingefügt.

© 2016-2022 Altova GmbH

```
/*
The following SQL statements were executed during "Generate output" function.
Every single result is written right to the "-->>>" string.
These statements are only for preview and may not be executed in another SQL query tool!
The database was connected using the following connection-string:
C:/sqlite/articles.sqlite
*/

PRAGMA foreign_keys = ON;
-->>> OK. 0 row(s).

DELETE FROM "articles"
-->>> OK. 0 row(s).

INSERT INTO "articles" ("number", "name", "singleprice") VALUES (1, 'T-Shirt', 25)
-->>> OK. 1 row(s).

INSERT INTO "articles" ("number", "name", "singleprice") VALUES (2, 'Socks', 2.3)
-->>> OK. 1 row(s).

INSERT INTO "articles" ("number", "name", "singleprice") VALUES (3, 'Pants', 34)
-->>> OK. 1 row(s).

INSERT INTO "articles" ("number", "name", "singleprice") VALUES (4, 'Jacket', 57.5)
-->>> OK. 1 row(s).
```

So überprüfen Sie, ob die Daten korrekt in die SQLite-Datenbank eingefügt wurden:

1. Klicken Sie in MapForce auf das Register **DB-Abfrage**.
2. Wählen Sie aus der Dropdown-Liste im oberen Bereich des Fensters **SQL-Abfrage** den Eintrag **articles** aus.
3. Geben Sie im SQL Editor die folgende Abfrage ein:

```
select * from articles;
```

Führen Sie alternativ dazu die folgenden Schritte durch:

1. Führen Sie die Datei **c:\sqlite\sqlite3.exe** aus und öffnen Sie die Datenbank mit dem Befehl:

```
.open Articles.sqlite
```

2. Führen Sie die folgenden SELECT-Anweisung aus:

```
select * from articles;
```

Unabhängig davon, welche Methode Sie zur Auswahl der Tabellendaten verwenden (MapForce oder SQLite Shell), sollte das Ergebnis der Abfrage nun vier Zeilen sein. Dies entspricht der Anzahl der Datensätze in der XML-Datei.

4.2.4 Verknüpfen von Datenbankdaten mittels Join

In Mappings, in denen Daten aus Datenbanken gelesen werden, können Sie Datenbankobjekte wie Tabellen oder Ansichten durch Hinzufügen einer Join-Komponente zum Mapping miteinander verknüpfen. So könnten Sie etwa Daten aus zwei oder mehr Tabellen, die, wie beim Speichern von Daten in relationalen Datenbanken üblich, durch Sekundärschlüsselbeziehungen gebunden sind, miteinander kombinieren. Das Ergebnis wäre dasselbe wie bei der Ausführung einer SQL-Abfrage an der Datenbank, bei der zwei oder mehr Tabellen mit Hilfe einer INNERER JOIN (oder ggf. LINKER JOIN)-Operation miteinander verknüpft werden.

Je nach Art der mit der Join-Komponente verbundenen Daten kann die Join-Operation entweder im Standard-Modus (nicht SQL) oder im SQL-Modus erfolgen. Join-Verknüpfungen im Nicht-SQL-Modus werden von MapForce durchgeführt, während Join-Verbindungen im SQL-Modus von der Datenbank, aus der die Daten ausgelesen werden, durchgeführt werden.

Join-Verknüpfungen im Nicht-SQL-Modus sind flexibler, da sie mehr Komponententypen als Input unterstützen (So kann die Join-Verknüpfung etwa zwischen Tabellen aus unterschiedlichen Datenbanken oder zwischen XML-Strukturen und Datenbanktabellen erfolgen). Ein Beispiel für eine Nicht-SQL-Verknüpfung finden Sie unter Beispiel: Verknüpfen von XML-Strukturen 902. Andernfalls müssen bei einer Nicht-SQL-Join-Verknüpfung speicherintensive Vergleiche durchgeführt werden (da die Gesamtanzahl der Vergleiche das Kreuzprodukt oder Kartesische Produkt aller verknüpften Strukturen bildet). Normalerweise geht dies sehr schnell und ist in Mappings mit wenigen Daten vernachlässigbar. Wenn die miteinander verknüpften Datenquellen jedoch eine sehr große Menge an Datensätzen enthalten, so wird die Ausführung des Mappings entsprechend lange dauern. Wenn in Ihren Mappings eine sehr große Anzahl an Datensätzen verarbeitet werden muss, sollten Sie eventuell die MapForce Server Advanced Edition verwenden, die spezielle Join-Optimierungen zur Beschleunigung der Mapping-Ausführung enthält.

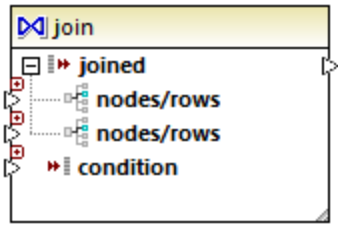
Für eine Join-Verknüpfung im SQL-Modus werden nur geeignete Datenbankobjekte (wie z.B. Tabellen oder Ansichten) als Input akzeptiert, daher ist dieser Modus nicht so flexibel wie eine Nicht-SQL-Join-Verknüpfung. Dafür ermöglicht dieser Modus eine bessere Mapping Performance, da der Join nativ von der Datenbank ausgeführt wird. Nähere Informationen dazu finden Sie unter Join-Verknüpfungen im SQL-Modus 345.

Anmerkung: Datenbanktabellen oder -ansichten können nicht nur mittels einer Join-Komponente miteinander verknüpft werden. Join-Verknüpfungen bei Datenbanken können auch mit Hilfe von SQL SELECT-Anweisungen durchgeführt werden. Siehe SELECT-Anweisungen als virtuelle Tabellen 368. Ein bedeutender Unterschied zwischen SQL SELECT-Anweisungen und Join-Komponenten ist, dass erstere von Hand geschrieben werden, so dass sie mehr Flexibilität bieten. Join-Komponenten sind eine einfachere Alternative, wenn Sie mit dem Schreiben von SQL-Anweisungen nicht so vertraut sind.

So fügen Sie eine Join-Komponente hinzu:

1. Definieren Sie BUILT-IN als Mapping-Transformationssprache (Klicken Sie dazu entweder auf die Symbolleisten-Schaltfläche  oder verwenden Sie den Menübefehl **Ausgabe | Built-In Ausführungsprozessor**).
2. Wählen Sie im Menü **Einfügen** den Befehl **Verbinden**. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche **Verbinden** (). Daraufhin wird die Join-Komponente im Mapping angezeigt. Standardmäßig akzeptiert sie Daten aus zwei Strukturen, daher hat sie zwei `nodes/rows`-Inputs. Falls nötig, können Sie neue Inputs durch Klicken auf die Schaltfläche **Input hinzufügen** () hinzufügen, siehe Verknüpfen von drei oder mehr Strukturen 901.

 © 2016-2022 Altova GmbH



3. Verbinden Sie die gewünschten Strukturen mit den `nodes/rows`-Datenelementen der Join-Komponente.
4. Fügen Sie eine (oder mehrere) Bedingungen für die Verknüpfung hinzu. Klicken Sie dazu mit der rechten Maustaste auf die Join-Komponente und wählen Sie **Eigenschaften**. Join-Bedingungen können auch direkt über das Mappen hinzugefügt werden, indem Sie das Boolesche Ergebnis einer Funktion mit dem `condition`-Datenelement der Join-Komponente verbinden. In bestimmten Fällen, beim Verknüpfen von Datenbanktabellen mittels Join, kann die Join-Bedingung (oder die Join-Bedingungen) von MapForce automatisch erstellt werden. Nähere Informationen dazu finden Sie unter Hinzufügen von Join-Bedingungen 898.

Anmerkungen:

- Join-Verbindungen werden unterstützt, wenn als Zielsprache des Mappings BUILT-IN definiert ist. Die Codegenerierung in C#, C++ oder Java wird nicht unterstützt.
- Wenn eine Struktur keine gültige oder unterstützte Input-Quelle für die Join-Verknüpfung ist, wird entweder direkt im Mapping oder im Fenster "Meldungen" eine entsprechende Information angezeigt, wenn Sie das Mapping validieren (siehe Validieren von Mappings 107).
- Join-Komponenten sollten nicht mit Input-Parametern oder Ergebnissen von inline gesetzten benutzerdefinierten Funktionen verbunden werden. Falls solche Verbindungen vorhanden sind, kommt es bei der Mapping-Validierung zu Validierungsfehlern.
- Wenn Sie verknüpfbare Datenbankkomponenten (wie z.B. Tabellen oder Ansichten) direkt mit einer Join-Komponente verknüpfen, wird in der rechten oberen Ecke der Join-Komponente automatisch eine **SQL-Modus** (SQL) Schaltfläche angezeigt. Wenn diese Schaltfläche aktiviert ist, stehen spezielle SQL-Funktionen für die Join-Operation zur Verfügung (siehe Join-Verknüpfungen im SQL-Modus 345).
- Die Ausgabe des mit `Join` verknüpften Datenelements kann mit einer anderen Join-Komponente verbunden werden. Gegebenenfalls können Sie jedoch ein Teilergebnis der einen Join-Verknüpfung mit einer anderen verbinden.

4.2.4.1 Join-Verknüpfungen im SQL-Modus

Wenn Sie geeignete Datenbankkomponenten (wie z.B. Tabellen oder Ansichten) direkt mit einer Join-Komponente verbinden, wird in der rechten oberen Ecke der Join-Komponente eine **SQL-Modus** (SQL) Schaltfläche angezeigt. Wenn der SQL-Modus aktiviert ist, wird die Join-Operation von der Datenbank, aus der die Daten gelesen werden, ausgeführt. Anders ausgedrückt sendet MapForce intern eine Abfrage in der entsprechenden SQL-Syntax an die Datenbank, um Daten aus allen an der Join-Verknüpfung beteiligten Tabellen auszuwählen und zu kombinieren. Vor allem müssen Sie dazu keine SQL-Anweisung schreiben; die erforderliche Abfrage wird auf Basis Ihres visuellen Designs der Join-Komponente im Mapping erstellt, wie in den nachfolgenden Beispielen gezeigt.

Damit der SQL-Modus verwendet werden kann, müssen die folgenden Voraussetzungen erfüllt werden:

1. Beide zu verknüpfende Objekte (Tabellen oder Ansichten) müssen aus derselben Datenbank stammen.

2. Beide zu verknüpfende Objekte müssen aus derselben MapForce-Komponente stammen. (Beachten Sie, dass Sie Objekte in einer Komponente schnell hinzufügen/entfernen können: Klicken Sie mit der rechten Maustaste auf die Datenbankkomponente und wählen Sie im Kontextmenü den Befehl **Datenbankobjekte hinzufügen/entfernen/bearbeiten**.)
3. Die Join-Bedingung (oder Bedingungen) darf/dürfen ausschließlich über die Komponenteneigenschaften (durch Rechtsklick auf die Überschrift der Join-Komponente und Auswahl von Eigenschaften) und nicht im Mapping definiert sein (siehe auch Hinzufügen von Join-Bedingungen 898).

Anmerkung: Wenn Datenbanktabellen im SQL-Modus mittels Join verknüpft werden, erstellt MapForce die Join-Bedingung (oder Bedingungen) automatisch auf Basis von zwischen Tabellen gefundenen Sekundärschlüsselbeziehungen. Damit dies möglich ist, müssen sich die Datenbanktabellen in der MapForce-Komponente in einer Child-Parent-Beziehung befinden (d.h. eine Tabelle muss "Parent" oder "Child" einer anderen in der Komponente sein), siehe Beispiel: Verknüpfen von Tabellen mittels Join im SQL-Modus 348.

4. Alle Datenbanktabellen dürfen sich noch nicht im aktuellen Ziel-Kontext befinden. Wenn das Join-Ergebnis in einer Zielkomponente verwendet wird, darf keiner der verbundenen Tabellen direkt oder indirekt mit Parent-Ziel-Nodes verbunden sein. Nähere Informationen zur Ausführung eines Mappings finden Sie unter Mapping-Regeln und -Strategien 1330.

Sie können den SQL-Modus über die Schaltfläche **SQL** (SQL) in der rechten oberen Ecke der Join-Komponente anzeigen und steuern:

SQL SQL-Modus ist deaktiviert (Die Join-Verknüpfung wird von MapForce (oder ggf. von MapForce Server) durchgeführt.

SQL SQL-Modus ist aktiviert (Die Join-Verknüpfung wird von der Datenbank durchgeführt).

Wenn die SQL Schaltfläche fehlt, bedeutet dies, dass der SQL-Modus hier nicht sinnvoll ist oder für die verknüpften Daten nicht unterstützt wird.

In bestimmten Fällen muss der SQL-Modus explizit deaktiviert werden (SQL), z.B.:

- Wenn für Ihr Mapping Join-Bedingungen außerhalb der Join-Komponenteneigenschaften benötigt werden (d.h. im Mapping definierte Bedingungen, die mit dem `condition` Input der Join-Komponente verbunden sind).
- Wenn Sie Tabellen aus unterschiedlichen Datenbanken miteinander verbinden möchten. Verwenden Sie eine Standard-Join-Verknüpfung (nicht SQL), wenn Sie Tabellen aus unterschiedlichen Datenbanken verknüpfen müssen.

Ändern des Join-Modus

Wenn sich die Join-Komponente im **SQL-Modus** SQL befindet, können Sie Datenbanktabellen oder -ansichten auf eine der folgenden Arten verknüpfen:

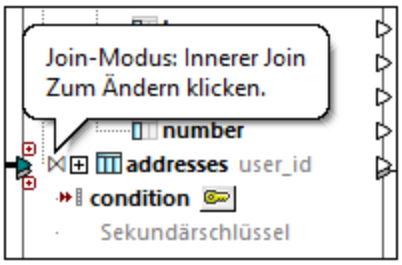
- INNERER JOIN - von der Join-Komponente werden nur Datensätze, die die Bedingung in beiden Input-Gruppen erfüllen, zurückgegeben.
- LINKER ÄUSSERER JOIN - die Join-Komponente inkludiert alle Datensätze aus der Tabelle ganz links (in MapForce ist dies die oberste Tabelle einer Join-Komponente), plus diejenigen Datensätze aus der in der Folge verknüpften Tabelle, die die Join-Bedingung erfüllen.

 © 2016-2022 Altova GmbH

Der Join-Modus einer Tabelle oder Ansicht in der Join-Komponente wird aus dem Symbol, das vor der verknüpften Tabelle oder Ansicht steht, ersichtlich. Für jede verknüpfte Tabelle oder Ansicht mit Ausnahme der ersten kann eines der folgenden Symbole angezeigt werden:

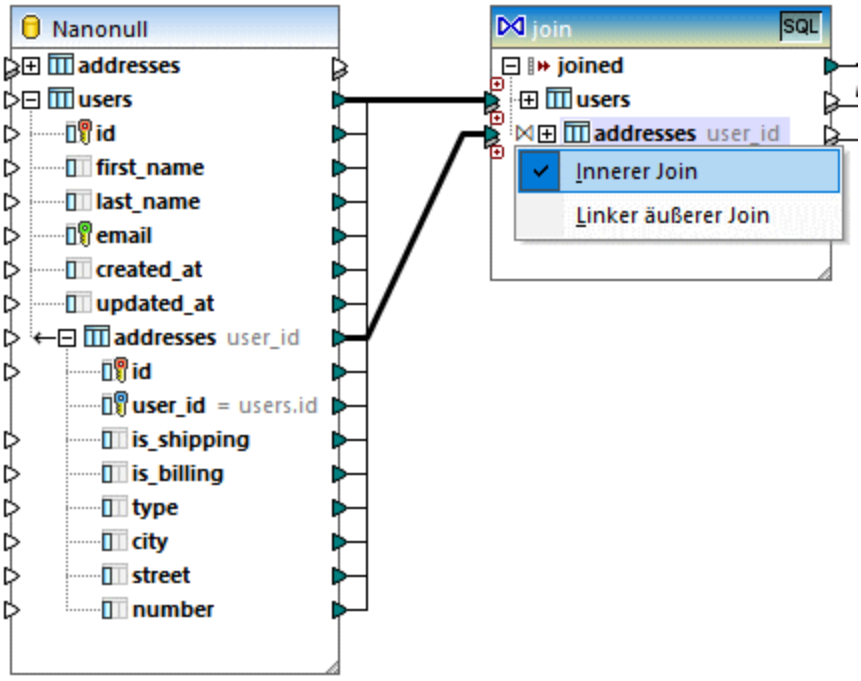
- Innerer Join ⋈
- Linker Join ⟕

Um einen Tooltipp mit näheren Informationen zum Join anzuzeigen, platzieren Sie den Mauszeiger über das Symbol:



Um den Join-Modus zu ändern, wählen Sie eine der folgenden Methoden:

- Klicken Sie auf das **Innerer Join** ⋈ oder **Linker Join** ⟕ Symbol vor der jeweiligen verknüpften Tabelle oder Ansicht und wählen Sie im Kontextmenü den Befehl **Innerer Join** oder **Linker äußerer Join**.
- Klicken Sie mit der rechten Maustaste in der Join-Komponente auf die zweite (oder dritte, vierte, usw.) verknüpfte Tabelle oder Ansicht und wählen Sie im Kontextmenü den Befehl **Join-Typ | Innerer Join** oder **Join-Typ | Linker äußerer Join**.



Beachten Sie dazu Folgendes:

- Wenn Sie den Join-Modus in LINKER ÄUSSERER JOIN ändern, bildet die obere Tabelle oder Ansicht die "linke" Seite des Join.

- Eine Änderung des Join-Modus wirkt sich auf die von der Join-Komponente zurückgegebenen Daten auf dieselbe Weise aus, wie sich ein INNERER JOIN oder LINKER JOIN auf das Ergebnis einer SQL-Abfrage in einer Datenbank auswirkt.

Aliasnamen

Oft kommt es vor, dass mit Join verknüpfte Datenbanktabellen oder -ansichten in beiden verknüpften Strukturen identische Feldnamen enthalten. Wenn der SQL-Modus aktiviert ist, wird solchen Datenelementen in der Komponente das Präfix "AS" vorangestellt. Wenn z.B. zwei verknüpfte Tabellen ein Feld "id" enthalten, wird das Feld in der ersten verknüpften Tabelle als "id" und in der zweiten verknüpften Tabelle als "id AS id2" angezeigt. Auch für verknüpfte Tabellen können Aliasnamen erzeugt werden, z.B. wenn eine Tabelle mit sich selbst verknüpft wird.

Die Alias-Feld- oder Tabellennamen sind wichtig, wenn Sie diese später in einem Mapping referenzieren möchten. Stellen Sie sich z.B. einen Fall vor, in dem Sie das Ergebnis der Join-Verknüpfung filtern oder sortieren möchten. Zu diesem Zweck kann der Output der Join-Komponente mit einer SQL WHERE/ORDER-Komponente, in die Sie die SQL WHERE-Klausel eingeben würden, verbunden werden.

Um ein Feld über eine WHERE-Klausel zu referenzieren, schreiben Sie den Tabellennamen, gefolgt von einem Punkt (.), gefolgt vom Feldnamen. Um einen Tabellenaliasnamen zu referenzieren, referenzieren Sie den Namen, so wie er in der Join-Komponente angezeigt wird. In der ORDER BY-Klausel können Sie entweder dieselbe Methode verwenden (`table.field`) oder nur den Aliasfeldnamen (den Namen, der nach "AS" steht) verwenden.

Ein Beispielmapping, in dem SQL WHERE/ORDER-Klauseln verwendet werden, finden Sie unter Beispiel: Join-Tabellen im SQL-Modus 348.

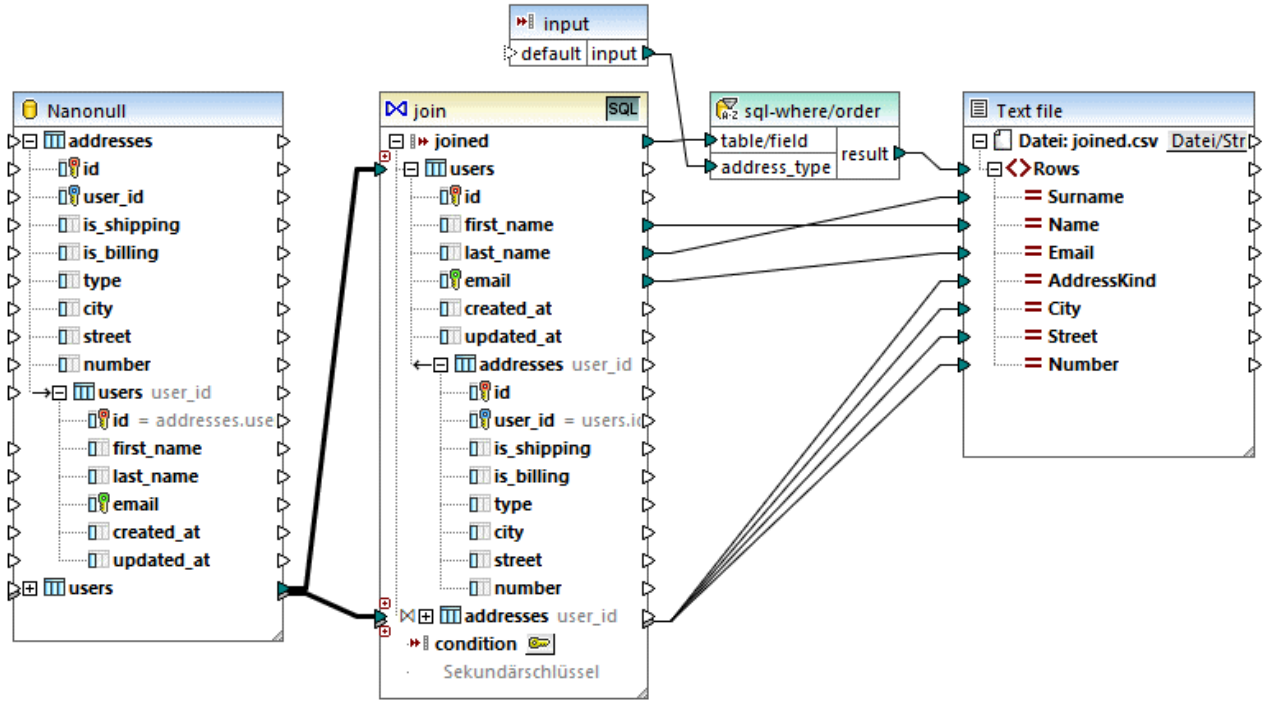
Anmerkung: SQL WHERE/ORDER-Komponenten sind zwischen einer Datenbanktabelle und der Join-Komponente nicht zulässig; Sie können nur hinter (nicht aber vor) einer Join-Komponente hinzugefügt werden. Nähere Informationen zu SQL WHERE/ORDER-Komponenten finden Sie unter Filtern und Sortieren von Datenbankdaten (SQL WHERE/ORDER) 361.

4.2.4.2 Beispiel: Verknüpfen von Tabellen mittels Join im SQL-Modus

In diesem Beispiel wird gezeigt, wie Sie Daten aus zwei Datenbanktabellen mit Hilfe einer MapForce Join-Komponente miteinander verknüpfen. Die Join-Operation wird im SQL-Modus durchgeführt, wie in Join-Verknüpfungen im SQL-Modus 345 beschrieben. Beachten Sie, dass das Verknüpfen von drei oder mehr Tabellen ähnlich funktioniert, siehe auch Beispiel: Erstellen eines CSV-Berichts anhand mehrerer Tabellen 357.

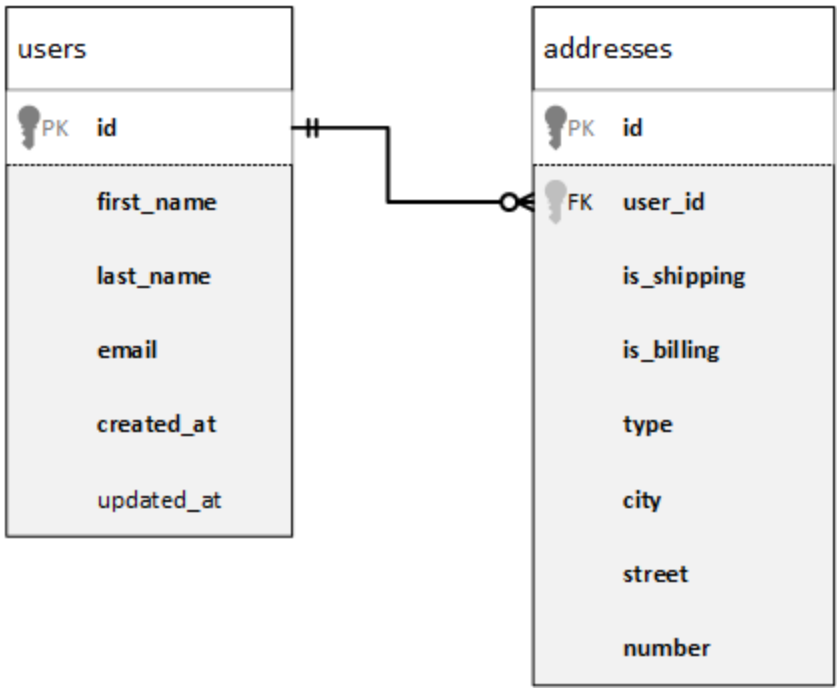
Das Mapping-Beispiel hierzu finden Sie unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2023\MapForceExamples\JoinDatabaseTables.mfd.

© 2016-2022 Altova GmbH



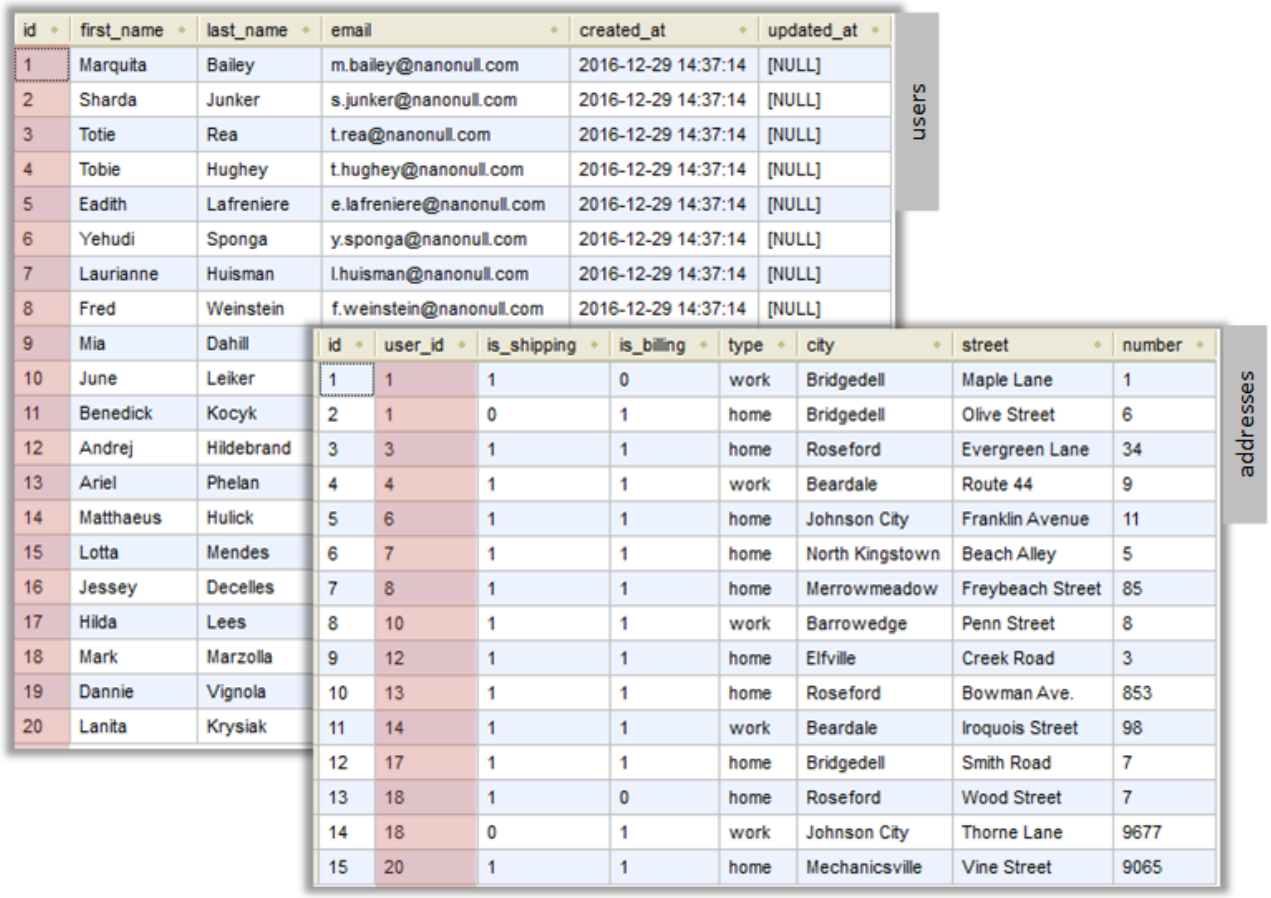
JoinDatabaseTables.mfd

Ziel des obigen Mappings ist es, Daten aus zwei Datenbankquelltabellen in einer einzigen CSV-Zieldatei zu kombinieren. Wie im Datenbankdiagramm unten gezeigt, sind in der ersten Tabelle (`users`) Personenadressen und in der zweiten Tabelle (`addresses`) Personennamen und E-Mail-Adressen gespeichert. Die beiden Tabellen sind über ein gemeinsames Feld miteinander verknüpft (`id` in `users` entspricht `user_id` in `addresses`). In der Datenbankterminologie bezeichnet man diese Art von Beziehung auch als "Sekundärschlüsselbeziehung".



In der Abbildung unten sehen Sie die tatsächlichen Daten in beiden Tabellen.

© 2016-2022 Altova GmbH



users

id	first_name	last_name	email	created_at	updated_at
1	Marquita	Bailey	m.bailey@nanonull.com	2016-12-29 14:37:14	[NULL]
2	Sharda	Junker	s.junker@nanonull.com	2016-12-29 14:37:14	[NULL]
3	Totie	Rea	t.rea@nanonull.com	2016-12-29 14:37:14	[NULL]
4	Tobie	Hughey	t.hughey@nanonull.com	2016-12-29 14:37:14	[NULL]
5	Eadith	Lafreniere	e.lafreniere@nanonull.com	2016-12-29 14:37:14	[NULL]
6	Yehudi	Sponga	y.sponga@nanonull.com	2016-12-29 14:37:14	[NULL]
7	Laurianne	Huisman	l.huisman@nanonull.com	2016-12-29 14:37:14	[NULL]
8	Fred	Weinstein	f.weinstein@nanonull.com	2016-12-29 14:37:14	[NULL]
9	Mia	Dahill			
10	June	Leiker			
11	Benedick	Kocyk			
12	Andrej	Hildebrand			
13	Ariel	Phelan			
14	Matthaeus	Hulick			
15	Lotta	Mendes			
16	Jessey	Decelles			
17	Hilda	Lees			
18	Mark	Marzolla			
19	Dannie	Vignola			
20	Lanita	Krysiak			

addresses

id	user_id	is_shipping	is_billing	type	city	street	number
1	1	1	0	work	Bridgedell	Maple Lane	1
2	1	0	1	home	Bridgedell	Olive Street	6
3	3	1	1	home	Roseford	Evergreen Lane	34
4	4	1	1	work	Beardale	Route 44	9
5	6	1	1	home	Johnson City	Franklin Avenue	11
6	7	1	1	home	North Kingstown	Beach Alley	5
7	8	1	1	home	Merrowmeadow	Freybeach Street	85
8	10	1	1	work	Barrowedge	Penn Street	8
9	12	1	1	home	Elfville	Creek Road	3
10	13	1	1	home	Roseford	Bowman Ave.	853
11	14	1	1	work	Beardale	Iroquois Street	98
12	17	1	1	home	Bridgedell	Smith Road	7
13	18	1	0	home	Roseford	Wood Street	7
14	18	0	1	work	Johnson City	Thorne Lane	9677
15	20	1	1	home	Mechanicsville	Vine Street	9065

Jeder Benutzerdatensatz in der Tabelle `users` kann null oder mehr Adressen in der Tabelle `addresses` haben. So kann ein Benutzer etwa eine Adresse vom Typ "home" oder zwei Adressen (eine vom Typ "home" und eine weitere vom Typ "work") oder gar keine Adresse haben.

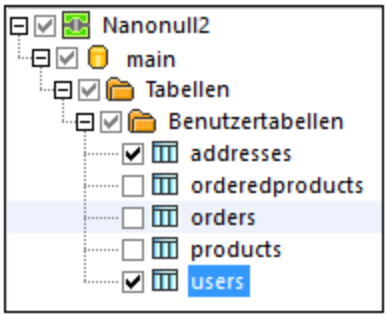
Ziel des Mappings ist es, die vollständigen Daten (Namen, Familienname, E-Mail, Stadt, Straße, Hausnummer) aller Benutzer, die in der Tabelle "addresses" mindestens eine Adresse haben, abzurufen. Es sollte auch problemlos möglich sein, nur Adressen einer bestimmten Art (z.B. nur Home-Adressen oder nur Work-Adressen) abzurufen. Die Art der abzurufenden Adressen ("home" oder "work") sollte in Form eines Parameters an das Mapping übergeben werden. Die abgerufenen Personendatensätze müssen nach dem Nachnamen (last name) alphabetisch sortiert werden.

Dies wird mit Hilfe einer Join-Komponente erreicht, wie in den Schritten unten beschrieben.

Anmerkung: Datenbanktabellen oder -ansichten können nicht nur mittels einer Join-Komponente miteinander verknüpft werden. Join-Verknüpfungen bei Datenbanken können auch mit Hilfe von SQL SELECT-Anweisungen durchgeführt werden. Siehe SELECT-Anweisungen als virtuelle Tabellen 368. Ein bedeutender Unterschied zwischen SQL SELECT-Anweisungen und Join-Komponenten ist, dass erstere von Hand geschrieben werden, so dass sie mehr Flexibilität bieten. Join-Komponenten sind eine einfachere Alternative, wenn Sie mit dem Schreiben von SQL-Anweisungen nicht so vertraut sind.

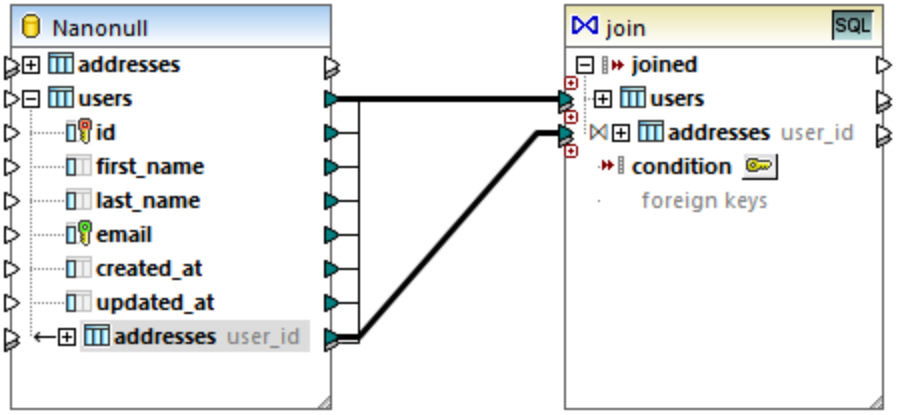
Schritt 1: Hinzufügen der Quelldatenbank

1. Klicken Sie im Menü **Einfügen** auf **Datenbank**. (Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche **Datenbank einfügen**).
2. Wählen Sie als Datenbankart "SQLite" aus und klicken Sie auf **Weiter**.
3. Navigieren Sie zur Datei **Nanonull.sqlite** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** und klicken Sie auf **Verbinden**.
4. Wenn Sie dazu aufgefordert werden, wählen Sie die Tabellen `addresses` und `users` aus.



Schritt 2: Hinzufügen der Join-Komponente

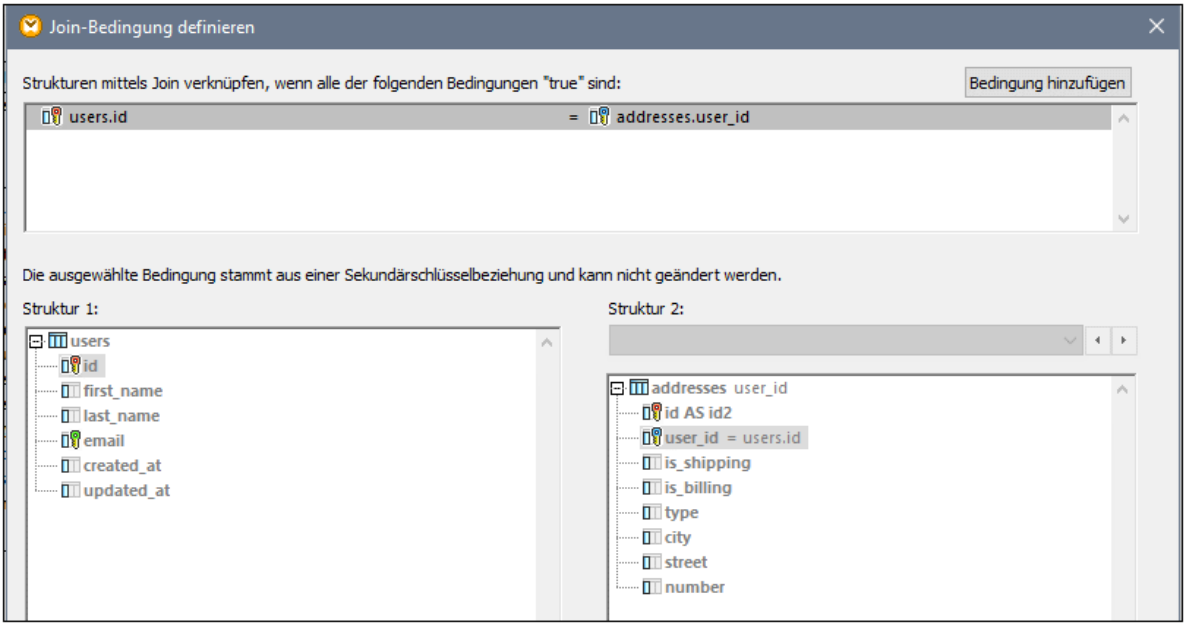
1. Klicken Sie im Menü **Einfügen** auf **Join**. (Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche **Join**).
2. Ziehen Sie eine Verbindung von der Tabelle `users` zum ersten Input der Join-Komponente.
3. Erweitern Sie die Tabelle `users` und ziehen Sie eine Verbindung von der Tabelle `addresses` (Child von `users`) zum zweiten Input der Join-Komponente. Über die Schaltfläche ⊡ können Sie, falls nötig, weitere Tabellen hinzufügen; in diesem Beispiel werden jedoch nur zwei Tabellen miteinander verknüpft.



Anmerkung: Die Verbindung kann auch direkt von der Tabelle `addresses` aus hinzugefügt werden (die Tabelle, die kein Child von `users` ist); in diesem Fall müssten die Bedingungen jedoch manuell definiert werden, wie unter Hinzufügen von Join-Bedingungen 898 beschrieben. In diesem Beispiel sollten Sie die Verbindungen jedoch wie oben gezeigt ziehen. Dadurch stellen Sie sicher, dass die benötigte Join-Verbindung automatisch erstellt wird.

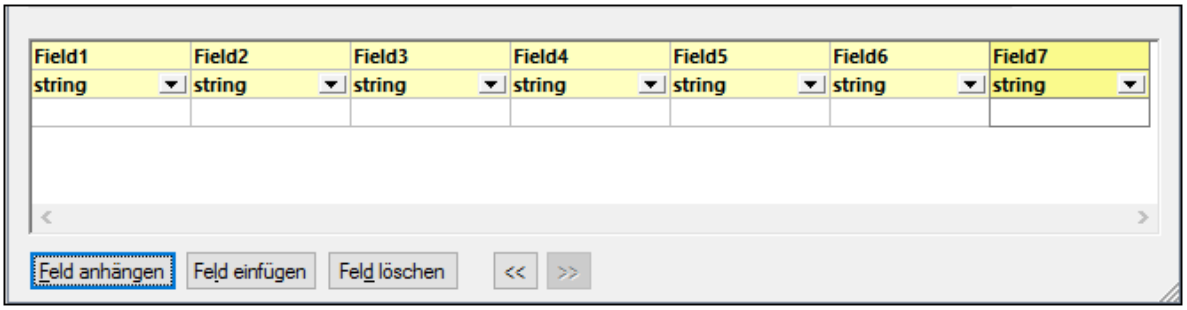
4. Klicken Sie in der Join-Komponente auf die Schaltfläche **Join-Bedingung definieren** . Beachten Sie, dass die Join-Bedingung automatisch erstellt wurde (`users.id = addresses.user_id`).

© 2016-2022 Altova GmbH

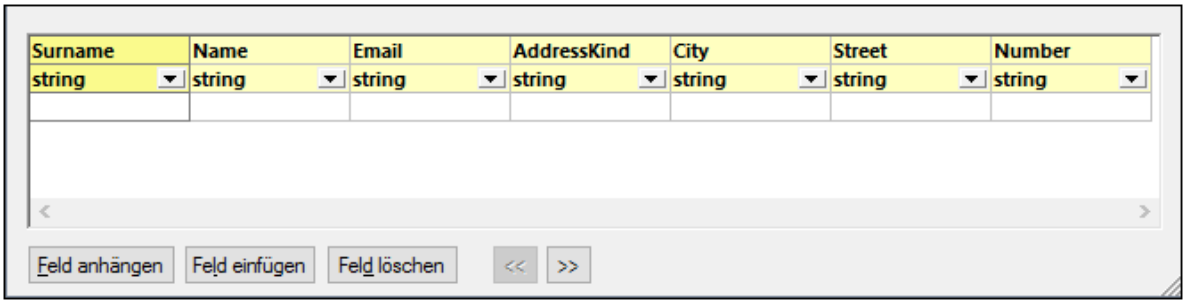


Schritt 3: Hinzufügen der CSV-Zielkomponente

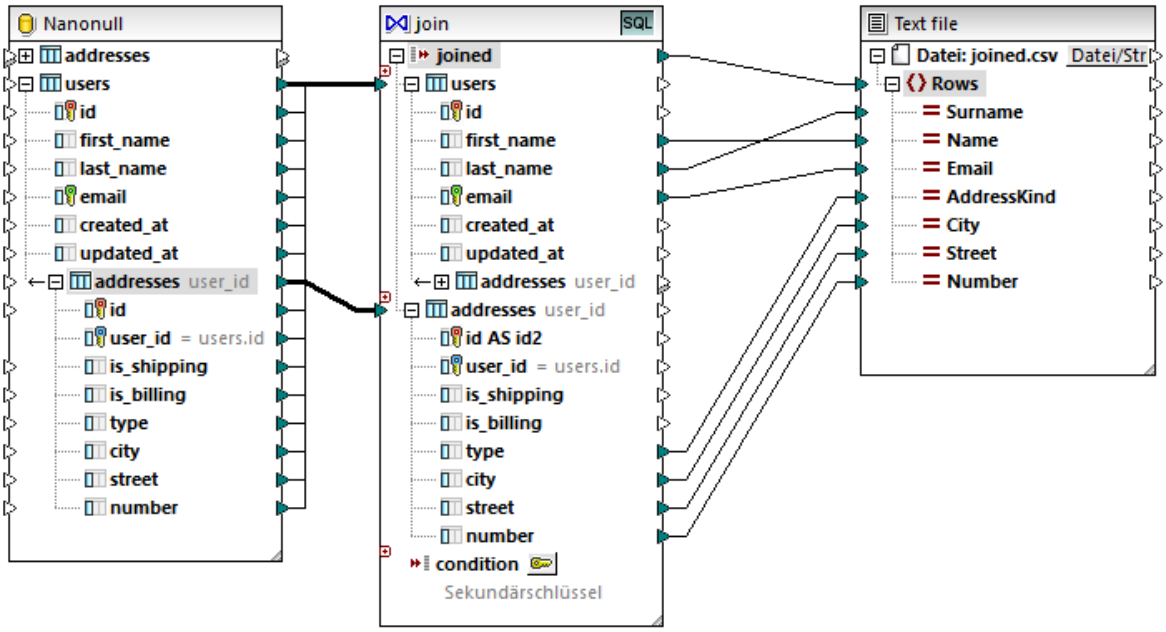
1. Klicken Sie im Menü **Einfügen** auf **Textdatei**. (Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche **Textdatei einfügen**).
2. Wenn Sie aufgefordert werden, einen Textverarbeitungsmodus auszuwählen, wählen Sie **Einfache Verarbeitung für Standard-CSV...** .
3. Klicken Sie mehrmals auf **Feld anhängen,** um sieben CSV-Felder zu erstellen. Belassen Sie alle anderen Einstellungen unverändert.



4. Doppelklicken Sie auf die Titelzellen der einzelnen Felder, um ihnen einen beschreibenden Namen zu geben (dadurch wird Ihr Mapping leichter lesbar).



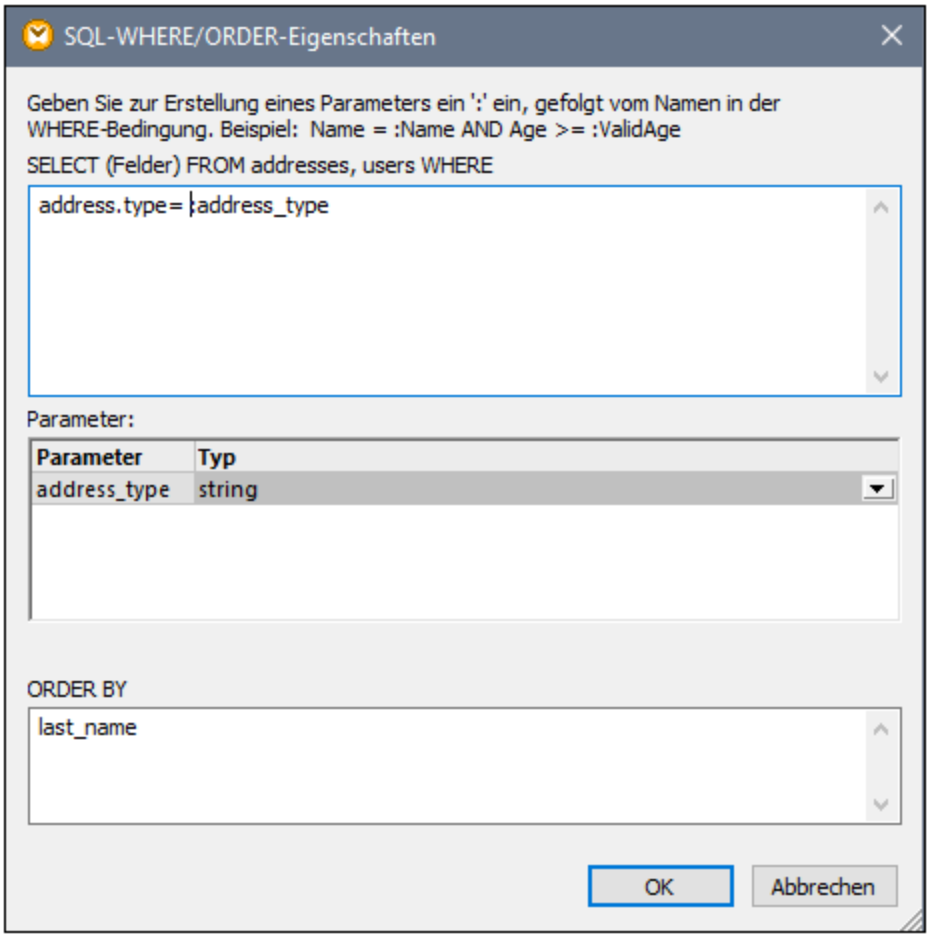
5. Ziehen Sie die Mapping-Verbindungen zwischen der Join-Komponente und der CSV-Komponente, wie unten gezeigt. Die Verbindung zwischen dem Datenelement `joined` der Join-Komponente und dem Datenelement `Rows` der Zielkomponente bedeutet "Erstelle in der Zielkomponente so viele Datensätze (Zeilen) wie Datensätze vorhanden sind, die die Join-Bedingung erfüllen".



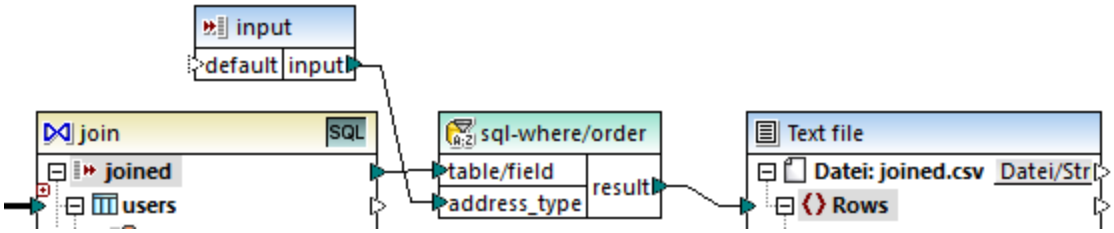
Schritt 4: Hinzufügen der SQL WHERE/ORDER-Bedingung und der Input-Parameter

1. Klicken Sie mit der rechten Maustaste auf die Verbindung zwischen dem Datenelement `joined` der Join-Komponente und dem Datenelement `Rows` der CSV-Zielkomponente und wählen Sie **SQL-WHERE/ORDER einfügen.**
2. Geben Sie die unten gezeigten WHERE- und ORDER BY-Klauseln ein.

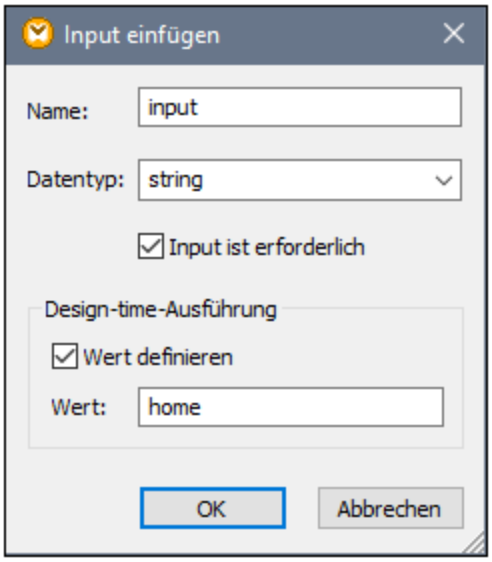
© 2016-2022 Altova GmbH



3. Fügen Sie im Mapping (mit dem Befehl **Einfügen | Input-Komponente einfügen**) eine Input-Komponente hinzu und verbinden Sie ihren Output mit dem im vorherigen Schritt erstellten Parameter `address_type`.



4. Doppelklicken Sie auf die Input-Komponente und konfigurieren Sie sie, wie unten gezeigt. Um in MapForce eine Vorschau auf die Mapping-Ausgabe anzuzeigen, wird ein Design-time-Wert benötigt (in diesem Fall "home"). Wenn in der Vorschau Arbeitsadressen abgerufen werden sollen, ersetzen Sie den Wert durch "work".



Erläuterung des Mappings

Mit der in Schritt 2 automatisch erstellten Join-Bedingung wird sichergestellt, dass nur Datensätze, die die Join-Bedingung `users.id = addresses.user_id` erfüllen, in die Zielkomponente kopiert werden. Die Join-Bedingung wurde automatisch hinzugefügt, weil die zwei Tabellen durch eine Sekundärschlüsselbeziehung verbunden sind und die Mapping-Verbindungen wurden entsprechend gezeichnet Nähere Informationen zu Tabellenbeziehungen finden Sie unter Behandlung von Datenbankbeziehungen[266]. Da in diesem Beispiel die bereits existierenden Tabellenbeziehungen verwendet wurden, mussten keine Join-Bedingungen manuell definiert werden. Ein Beispiel, in dem gezeigt wird, wie Join-Bedingungen manuell definiert werden, finden Sie unter Beispiel: Erstellen eines CSV-Berichts anhand mehrerer Tabellen[357].

Die beiden Quelltabellen stammen aus derselben Datenbank und derselben Komponente, daher können Sie für diese Join-Verknüpfung den SQL-Modus SQL nutzen. Da der SQL-Modus aktiviert ist, wird die Join-Operation von der Datenbank und nicht von MapForce ausgeführt. Anders ausgedrückt, wird von MapForce intern eine INNERER JOIN-Anweisung generiert und für die Ausführung an die Datenbank gesendet. Die Art des Join (INNERER JOIN) wird in der Join-Komponente durch das **Innerer Join** ⋈ Symbol vor der Tabelle **addresses** angezeigt. Sie können den Join-Typ auch in LINKER ÄUSSERER JOIN ⋈ ändern, wie unter Ändern des Join-Modus[346] beschrieben. Beachten Sie jedoch, dass eine Änderung des Join-Modus keine Auswirkung auf den Output dieses Beispiels hat.

Mit Hilfe der in Schritt 4 hinzugefügten SQL WHERE/ORDER-Komponente können die Datensätze gefiltert (so dass entweder Heimat- oder Arbeitsadressen abgerufen werden) und sortiert werden. Beachten Sie, dass mit der WHERE-Klausel ein Parameter **`:address_type`** vom Typ `string` erstellt wurde. Über diesen Parameter kann die Art der Adresse (home oder work) aus dem Mapping bereitgestellt werden. Nähere Informationen zu SQL WHERE/ORDER finden Sie unter Filtern und Sortieren von Datenbankdaten (SQL WHERE/ORDER)[361].

Über die Input-Komponente kann schließlich während der Ausführung des Mappings der tatsächliche Parameterwert bereitgestellt werden. Beachten Sie, dass der Input bei Ausführung des Mappings außerhalb von MapForce (z.B. wenn das Mapping von MapForce Server auf einem anderen Rechner ausgeführt wird) zur Mapping-Laufzeit als Befehlszeilenparameter bereitgestellt werden muss, d.h. der oben erwähnte Design-time-Wert wird ignoriert. Nähere Informationen finden Sie unter Bereitstellen von Parametern für das Mapping[867].

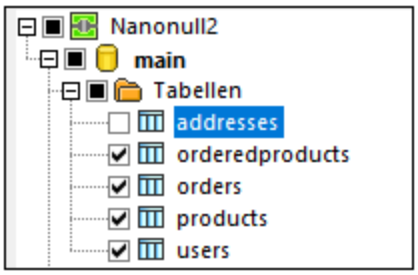
© 2016-2022 Altova GmbH

4.2.4.3 Beispiel: Erstellen eines CSV-Berichts anhand mehrerer Tabellen

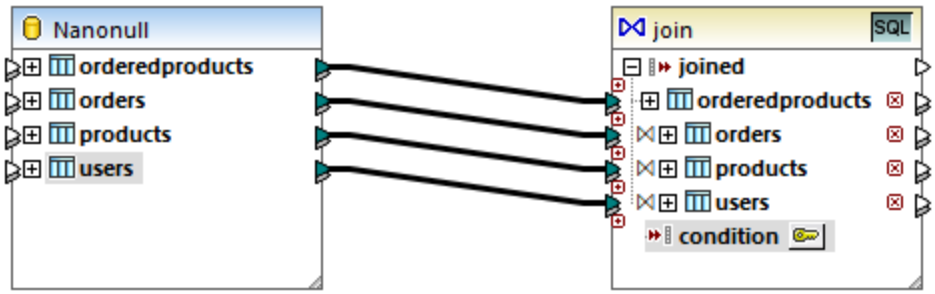
In diesem Beispiel wird gezeigt, wie Sie mehrere Datenbanktabellen mittels Join verknüpfen, um Daten in einen einzigen Bericht im CSV-Format zu extrahieren. Die in diesem Beispiel verwendete Datenbank heißt **Nanonull.sqlite** und befindet sich unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples**. In dieser Datenbank sind Informationen über ein fiktives Unternehmen gespeichert (dazu gehören Bestellungen, Produkte, Benutzer und deren Adressen). Wie bei relationalen Datenbanken meist der Fall, sind die Informationen normalisiert und über mehrere Tabellen verteilt. So sind etwa in der Tabelle `users` persönliche Benutzerdaten (wie Vorname, Nachname und E-Mail-Adresse) gespeichert. Außerdem sind in der Datenbank in zwei verschiedenen Tabellen Informationen über von Benutzern bestellte Produkte gespeichert: `orders` (welche die eindeutige ID der Bestellung und die Uhrzeit der Bestellung enthält) und `orderedproducts` (welche eine Liste der bestellten Produkte und deren Anzahl enthält). Weiters sind die Namen der Produkte selbst in einer separaten Tabellen namens `products` gespeichert.

Ziel dieses Mappings ist die Erstellung eines Berichts auf Basis der aus verschiedenen Tabellen extrahierten Daten, um zu sehen, wer, wann bestimmte Produkte in welcher Menge bestellt hat. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie im Menü **Einfügen** auf **Datenbank**.
2. Wählen Sie als Datenbankart **SQLite** aus und klicken Sie auf **Weiter**.
3. Navigieren Sie zur oben erwähnten Datenbank **Nanonull.sqlite** und klicken Sie auf **Verbinden**.
4. Wenn Sie dazu aufgefordert werden, wählen Sie die Tabellen `orderedproducts`, `orders`, `products` und `users` aus und klicken Sie auf **OK**.



1. Fügen Sie eine Join-Komponente zum Mapping hinzu und erstellen Sie durch Klicken auf die Schaltfläche **Input hinzufügen** (⊡) vier `nodes/rows`-Einträge.
2. Verbinden Sie die vier Tabellen aus der Datenbankkomponente mit den entsprechenden Input-Einträgen der Join-Komponente.



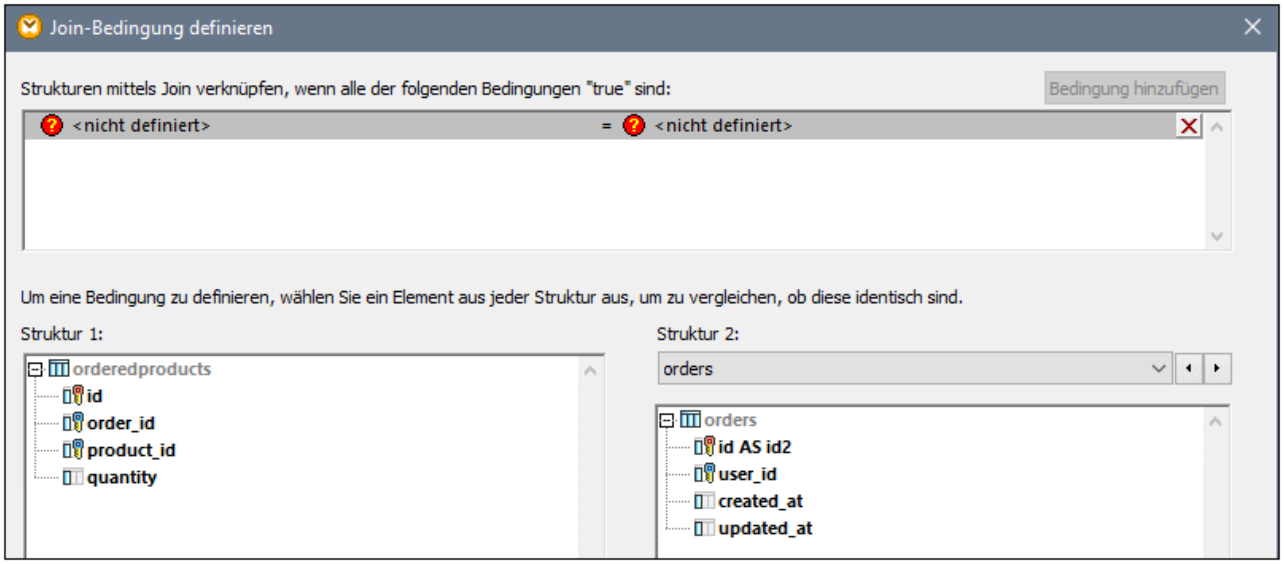
Anmerkung: In einem alternativen Szenario könnten Sie die Tabelle `orderedproducts` mit der Join-Komponente verbinden, anschließend die Tabelle `orders` (die Tabelle die darunter verschachtelt ist,

nicht die Tabelle auf derselben Ebene) usw., sodass alle verknüpften Tabellen unter derselben "Root"-Tabelle verschachtelt sind, siehe auch Behandlung von Datenbankbeziehungen 266. Das Mapping-Ergebnis wäre dasselbe, wenn Sie die Tabellen auf diese Art miteinander verknüpfen würden. Der Unterschied ist, dass die Join-Bedingungen in diesem Beispiel manuell erstellt werden, während die Join-Bedingungen im alternativen Szenario von MapForce automatisch erstellt würden. Ein Beispiel für das Verknüpfen von Tabellen mittels Join, bei dem manuell keine Join-Bedingungen definiert werden müssen, finden Sie unter Beispiel: Verknüpfen von Tabellen mittels Join im SQL-Modus 348. Ein weiteres Mapping, in dem sich alle verknüpften Tabellen unter derselben "Root"-Tabelle befinden, finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\DB_Denormalize.mfd**.

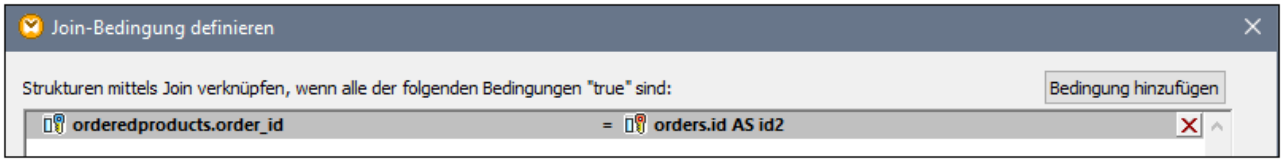
In diesem Beispiel haben die mit der Join-Komponente verbundenen Tabellen die folgenden Reihenfolge:

1. `orderedproducts`
2. `orders`
3. `products`
4. `users`

Diese Reihenfolge wirkt sich darauf aus, wie die entsprechenden Strukturen im Dialogfeld "Join-Bedingung definieren" angezeigt werden, wenn Sie auf die Schaltfläche **Join-Bedingung definieren** ( ) klicken. Die erste Tabelle (`orderedproducts`) wird hier standardmäßig unter **Struktur 1** und die Tabelle unmittelbar danach (`orders`) unter **Struktur 2** angezeigt.

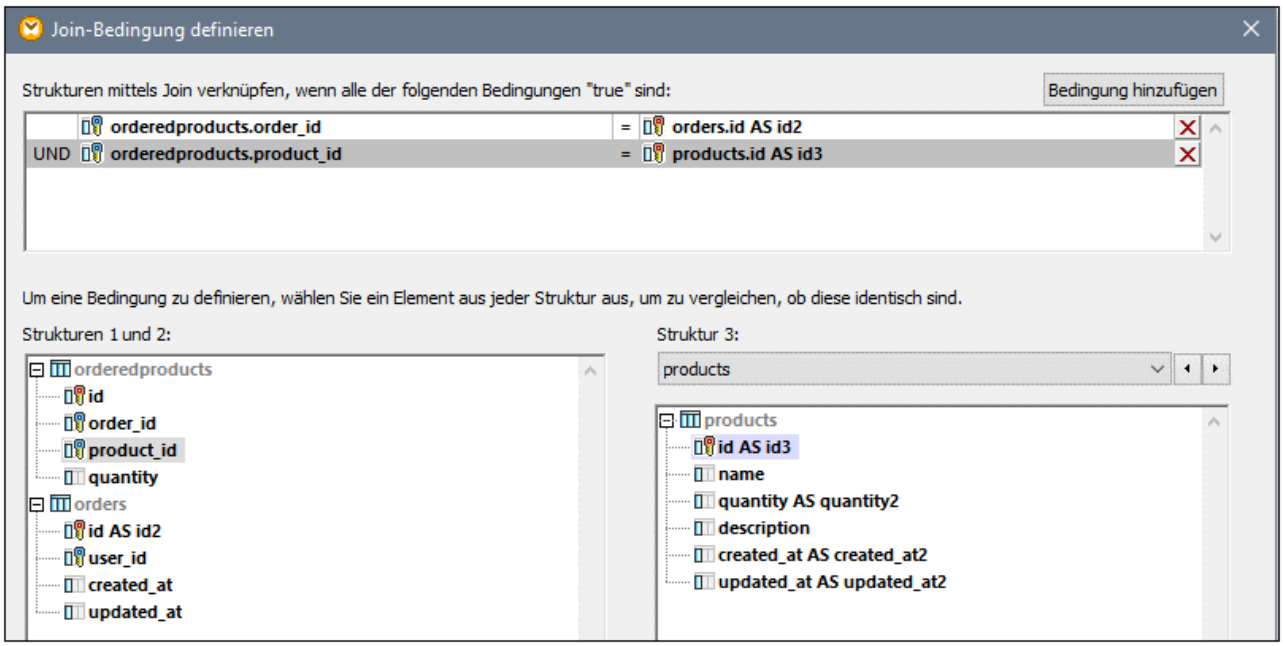


Um die erste Join-Bedingung zu definieren, klicken Sie im linken Bereich auf das Datenelement `order_id` und im rechten Bereich auf das Datenelement `id`. Die Felder `orderedproducts.order_id` und `orders.id`. bilden nun ein Paar:

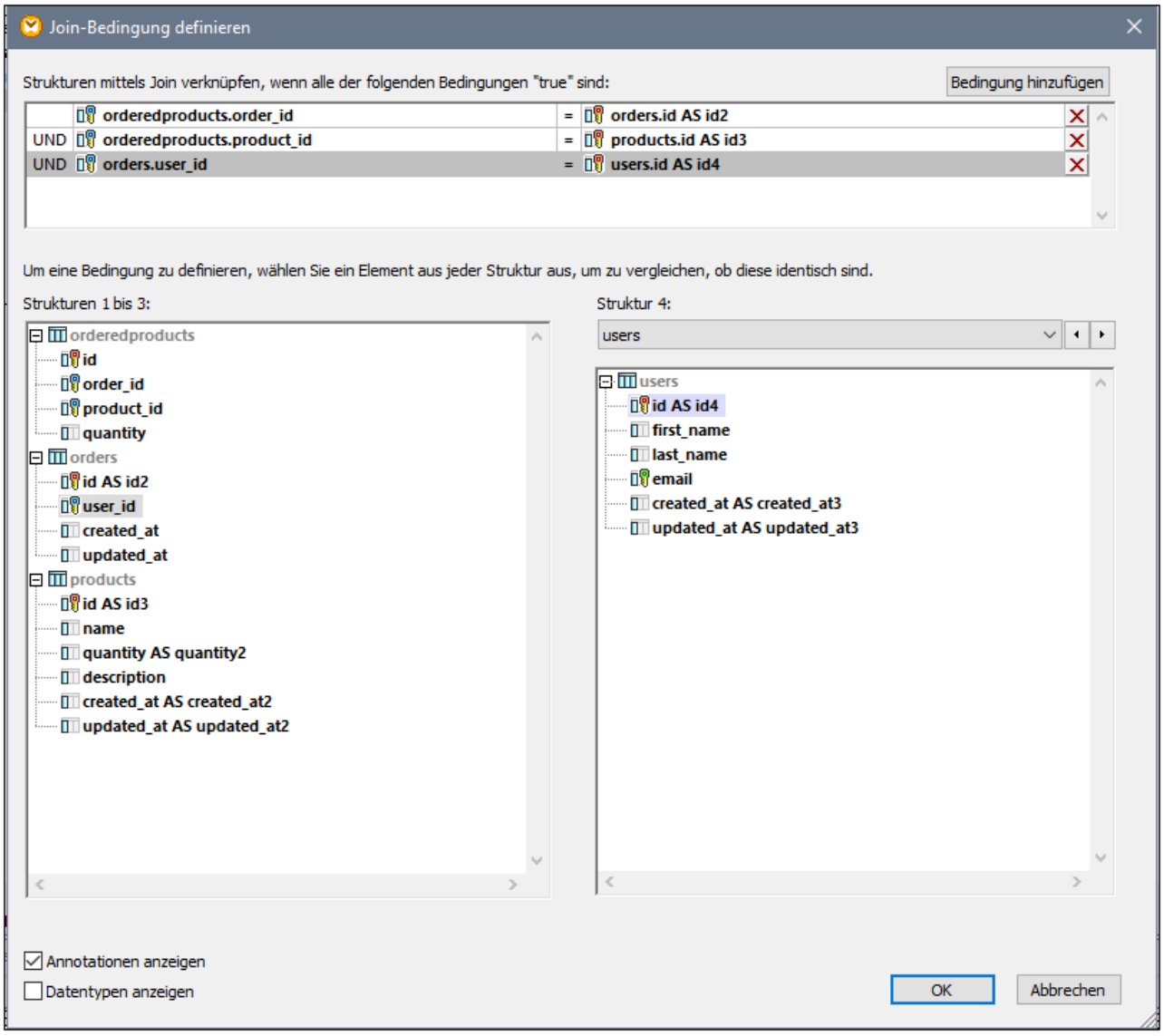


 © 2016-2022 Altova GmbH

Bisher wurden erst zwei Tabellen mittels Join verknüpft. Um Join-Bedingungen zu definieren, an denen eine dritte Tabelle beteiligt ist, wählen Sie die gewünschte Tabelle aus der Dropdown-Liste oberhalb des rechten Bereichs aus. Im linken Bereich werden in diesem Fall alle Tabellen, die *davor* vorkommen, angezeigt. Wenn Sie `products` auf der rechten Seite auswählen, werden auf der linken Seite `orderedproducts` und `orders` angezeigt (da diese Tabellen in der Join-Komponente vor `products` vorkommen). Sie können nun Paare aus den Feldern der Tabelle `products` und den Feldern der davor vorkommenden Tabellen (in diesem Fall `orderedproducts.product_id` und `products.id`) bilden.

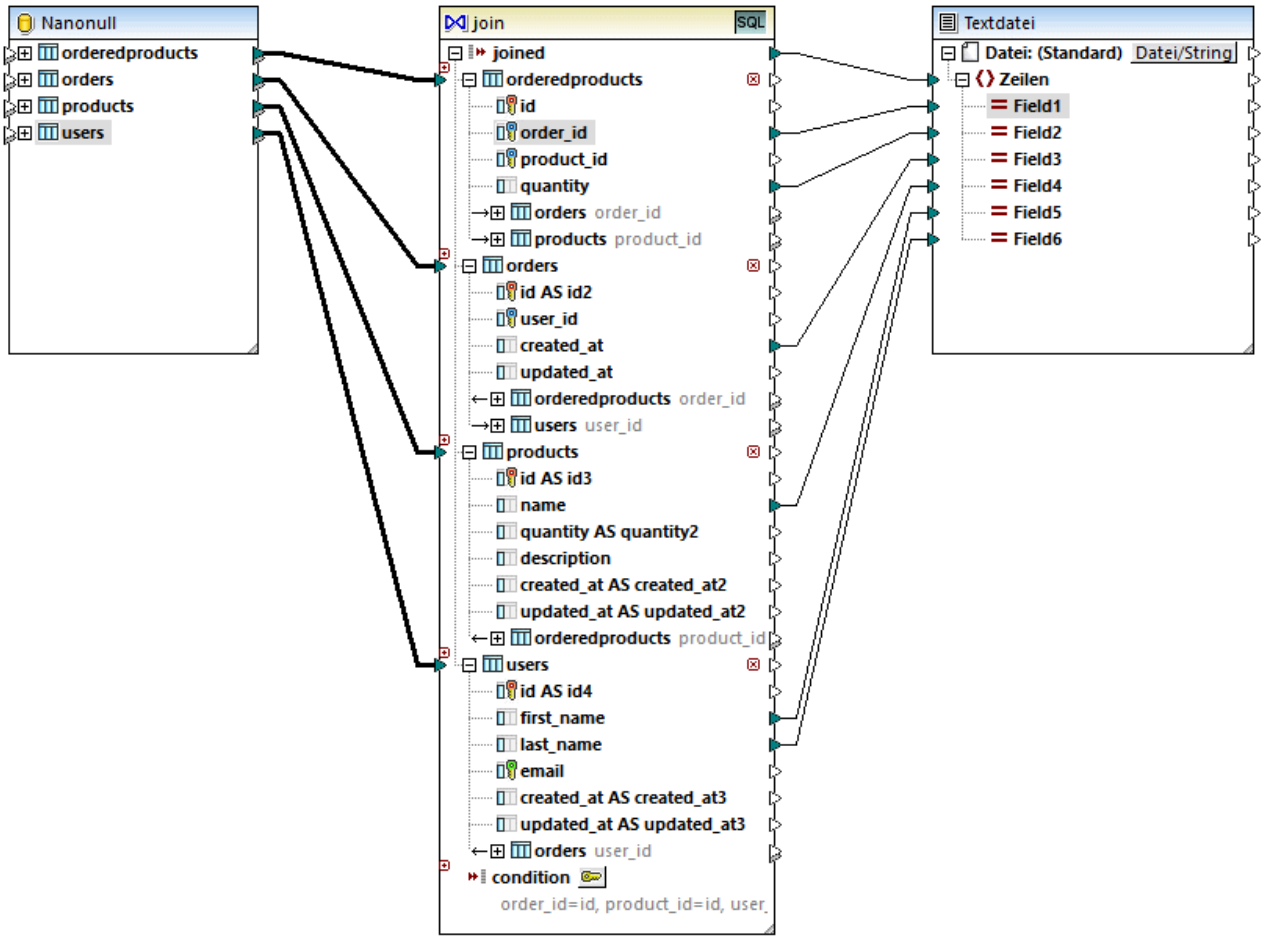


Um eine vierte Tabelle (`users`) mittels Join zu verknüpfen, wählen Sie die Tabelle `users` aus der Dropdown-Liste aus. Sie können nun Paare aus den Feldern `orders.user_id` und `users.id` bilden.



Jetzt, da alle erforderlichen Join-Bedingungen definiert wurden, können Datenelemente der Join-Komponente weiter auf eine Zielkomponente gemappt werden. Fügen Sie nun zum Abschluss des Mappings eine CSV-Komponente hinzu (siehe CSV- und Textdateien 585) und verbinden Sie Datenelemente aus der Join-Komponente mit der CSV-Zielkomponente, wie unten gezeigt.

© 2016-2022 Altova GmbH



Mit dem oben gezeigten Mapping wird ein Bericht (im CSV-Format) erstellt, der folgendermaßen aus allen vier am Join beteiligten Tabellen kompiliert wird:

- ID der Bestellung (aus der Tabelle `orderedproducts`)
- Anzahl der bestellten Produkte (aus der Tabelle `orderedproducts`)
- Uhrzeit der Bestellung (aus der Tabelle `orders`)
- Name des bestellten Produkts (aus der Tabelle `products`)
- Vor- und Nachname des Benutzers, der das Produkt bestellt hat (aus der Tabelle `users`).

Alle Tabellen in diesem Beispiel werden mit der INNERER JOIN-Methode miteinander verknüpft. Informationen darüber, wie Sie den Join-Modus in LINKER ÄUSSERER JOIN ändern, finden Sie unter Ändern des Join-Modus 346.

4.2.5 Filtern und Sortieren von Datenbankdaten (SQL WHERE/ORDER)

Um Datenbankdaten zu filtern und zu sortieren, verwenden Sie eine **SQL/NoSQL-WHERE/ORDER**-Komponente. Auf diese Art können Sie manuell eine SQL WHERE-Klausel eingeben, die die Daten filtert.

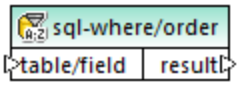
Optional dazu können Sie auch eine ORDER BY-Klausel definieren, wenn Sie die Datensätze nach einem bestimmten Datenbankfeld in auf- oder absteigender Ordnung sortieren möchten.

Die **SQL/NoSQL-WHERE/ORDER**-Komponente muss mit einer Tabelle oder einem Feld einer Datenbank-Mapping-Komponente verbunden werden. Sie können eine **SQL/NoSQL-WHERE/ORDER**-Komponente auch mit einer **Join**-Komponente verbinden, wenn die verknüpfte Gruppe bzw. die verknüpften Datensätze gefiltert werden sollen (siehe Verknüpfen von Datenbankdaten mittels Join 344).

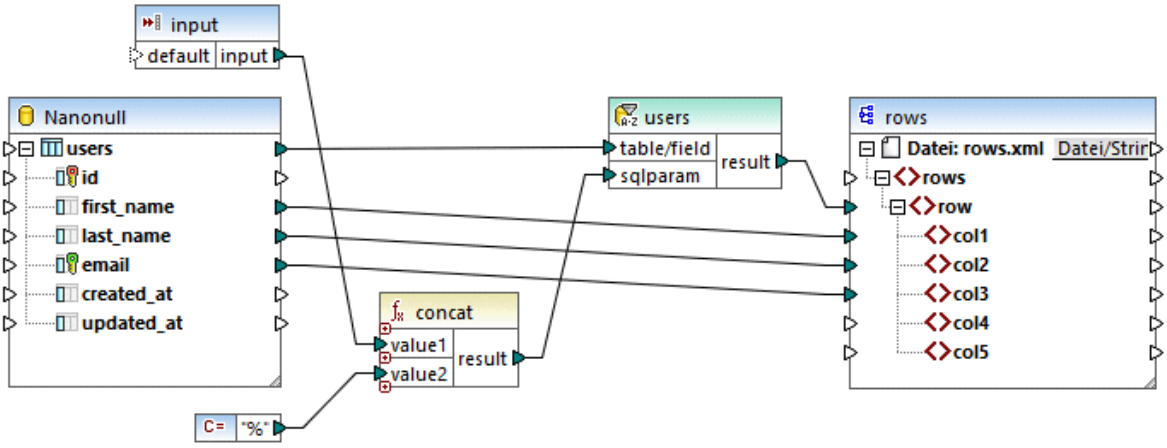
Hinzufügen einer SQL/NoSQL-WHERE ORDER-Komponente

Um eine **SQL/NoSQL-WHERE/ORDER**-Komponente zum Mapping hinzuzufügen, gehen Sie folgendermaßen vor:

1. Gehen Sie zum Menü **Einfügen** und klicken Sie auf **SQL/NoSQL-WHERE/ORDER**. Standardmäßig hat diese Komponente die folgende Struktur:

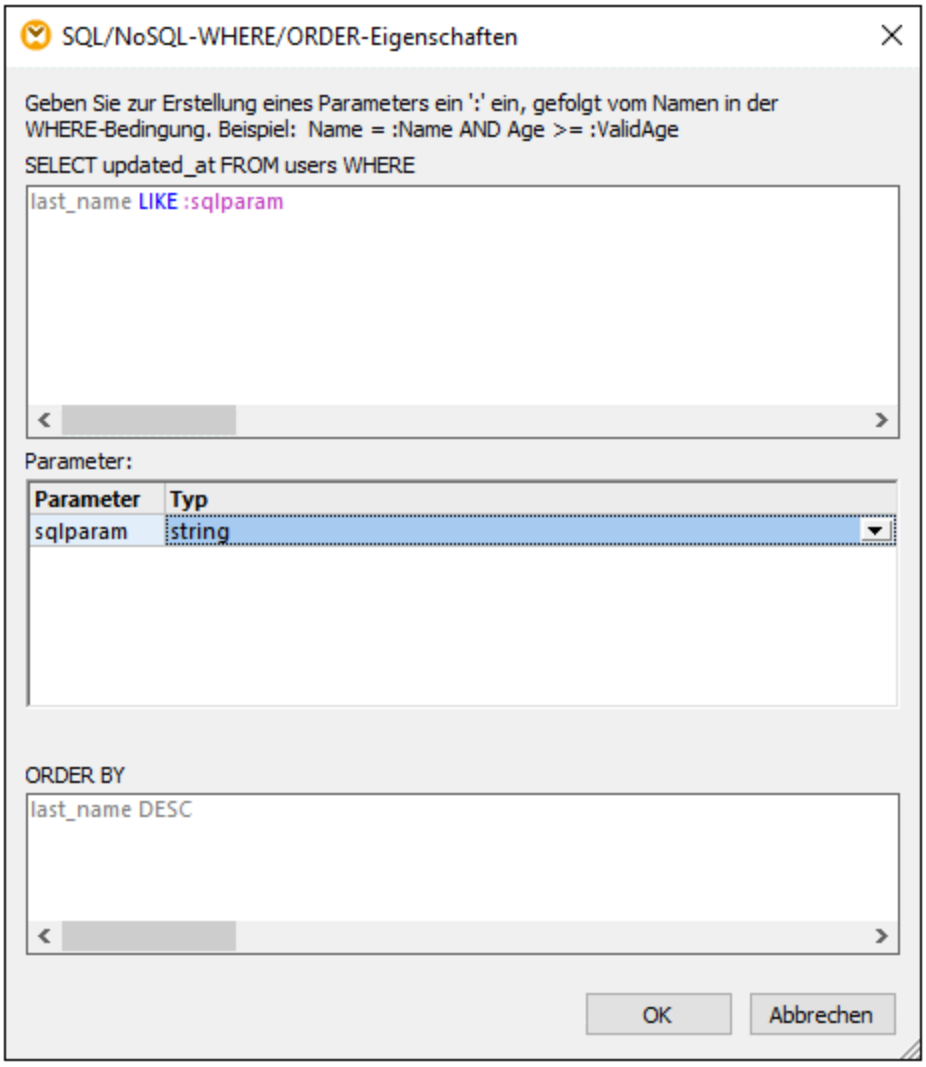


2. Verbinden Sie eine Tabelle oder ein Feld der Quelldatenbank mit dem Datenelement `table/field` der **SQL/NoSQL-WHERE/ORDER**-Komponente. Das Beispielmapping **`FilterDatabaseRecords.mfd`** (*siehe Abbildung unten*) dazu finden Sie im folgenden Ordner: **`<Dokumente>\Altova\MapForce2023\MapForceExamples\`**. In diesem Mapping nimmt die **SQL/NoSQL-WHERE/ORDER**-Komponente die Daten aus der Quelltabelle `users`, filtert alle Datensätze und wählt nur diejenigen aus, deren Nachname mit dem Buchstaben "M" beginnt (*siehe Erläuterung im Unterabschnitt weiter unten*).



3. Doppelklicken Sie auf die Titelleiste der **SQL/NoSQL-WHERE/ORDER**-Komponente oder klicken Sie mit der rechten Maustaste auf die Komponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften**. Daraufhin wird das Dialogfeld **SQL/NoSQL-WHERE/ORDER-Eigenschaften** geöffnet.

© 2016-2022 Altova GmbH



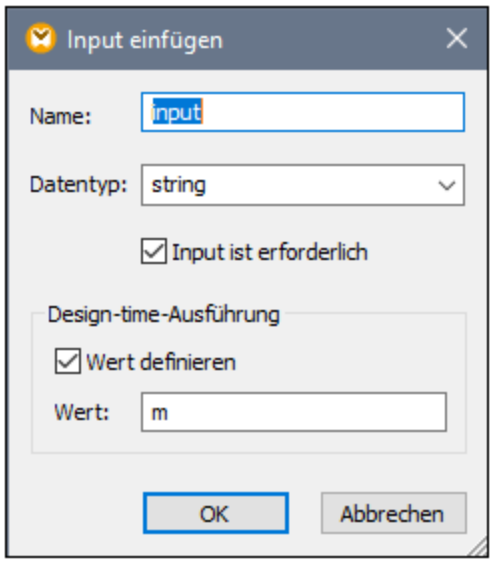
4. Geben Sie eine SQL WHERE-Klausel in das Textfeld oben ein. In unserem Beispiel lautet die SQL Where-Klausel folgendermaßen: **`last_name LIKE :sqlparam`**. Geben Sie optional eine ORDER BY-Klausel ein. In der Abbildung oben sehen Sie die im Mapping **`FilterDatabaseRecords.mfd`** definierte WHERE- und ORDER BY-Klausel (dieses Einstellungen werden weiter unten beschrieben). Weitere Beispiele finden Sie unter Erstellen von WHERE- und ORDER BY-Klauseln 365.

Parameter in SQL/NoSQL-WHERE/ORDER-Komponenten

In der **SQL/NoSQL-WHERE/ORDER**-Komponente aus dem Mapping **`FilterDatabaseRecords.mfd`** ist die folgende WHERE-Klausel definiert: **`last_name LIKE :sqlparam`**, wobei sich `last_name` auf den Namen des Datenbankfelds in der verbundenen Tabelle bezieht und `LIKE` ein SQL-Operator ist; `:sqlparam` erstellt im Mapping einen Parameter namens `sqlparam`.

Parameter in der **SQL/NoSQL-WHERE/ORDER**-Komponente sind optional. Sie sind nützlich, wenn Sie über das Mapping einen Wert an die WHERE-Klausel übergeben möchten. Ohne Parameter hätte die obige WHERE-Klausel folgendermaßen gelautet: `Last LIKE "M%"`. Damit würden alle Personen, deren Nachname mit dem Buchstaben "M" beginnt, abgerufen. Um diese Abfrage noch flexibler zu gestalten, haben wir anstelle von `"M%"` einen Parameter hinzugefügt. Dadurch kann auch jeder andere Buchstabe aus dem Mapping

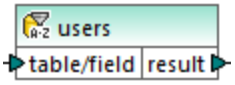
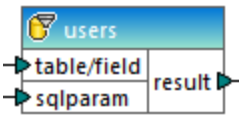
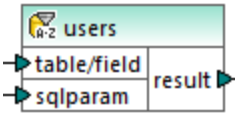
bereitgestellt werden, z.B. "D", um durch Änderung einer Konstante oder eines Mapping-Input-Parameters Personen, deren Nachname mit "D" beginnt, abzurufen. Der Input-Buchstabe im obigen Mapping stammt aus einer Input-Komponente namens `input`. Wenn Sie auf die Titelleiste dieser Komponente doppelklicken und ihre Eigenschaften öffnen, sehen Sie, dass `m` zum Zeitpunkt des Designs als Ausführungswert angegeben wird (*siehe Abbildung unten*).



Das SQL-Platzhalterzeichen % wird im Mapping von einer Konstante bereitgestellt und anschließend mit Hilfe der **concat**-Funktion mit dem Parameterwert verkettet. Dies hat den Vorteil, dass man keine SQL-Platzhalterzeichen in die Befehlszeile eingeben muss, wenn dieses Mapping in einer andere Umgebung (z.B. MapForce Server) ausgeführt wird.

Aussehen von SQL/NoSQL-WHERE/ORDER-Komponenten

Das Aussehen von **SQL/NoSQL-WHERE/ORDER**-Komponenten ändert sich, je nachdem welche Einstellungen darin definiert sind. Auf diese Art sehen Sie gleich direkt im Mapping, was die **SQL/NoSQL-WHERE/ORDER**-Komponente tut (*siehe Tabelle unten*).

	Es wurde eine WHERE-Klausel definiert.
	Es wurde eine WHERE-Klausel mit einem Parameter definiert. Der Parametername ist unterhalb des Datenelements `table/field` zu sehen.
	Es wurde eine WHERE-Klausel mit einem Parameter definiert. Zusätzlich dazu wurde eine ORDER BY-Klausel definiert. Die Sortierreihenfolge wird durch das Sortiersymbol A-Z angezeigt.

Wenn Sie den Mauszeiger über die Titelleiste von **SQL/NoSQL-WHERE/ORDER** platzieren, wird ein Tooltip mit den verschiedenen definierten Klauseln angezeigt.

© 2016-2022 Altova GmbH

4.2.5.1 Erstellen von WHERE- und ORDER BY-Klauseln

Nachdem Sie eine **SQL/NoSQL-WHERE/ORDER**-Komponente zum Mapping hinzugefügt haben, benötigen Sie dafür eine WHERE-Bedingung (Klausel), mit der Sie festlegen können, wie genau die mit der Komponente verbundenen Daten gefiltert werden sollen. Die WHERE-Bedingung muss in das Dialogfeld **SQL/NoSQL-WHERE/ORDER-Eigenschaften** (*siehe vorhergehender Abschnitt*) eingegeben werden.

WHERE-Bedingungen in MapForce werden ähnlich wie dieselbe SQL-Klausel außerhalb von MapForce geschrieben. Verwenden Sie die Syntax für die SQL-Variante für die entsprechende Datenbank. Verwenden Sie z.B. Operatoren, Platzhalter, Sub-Select-Klauseln oder Aggregierungsfunktionen. Um Parameter zu erstellen, die über das Mapping übergeben werden können, geben Sie ein Semikolon-Zeichen (:), gefolgt von einem Parameternamen ein.

Anmerkung: Sobald Sie die WHERE-Klausel fertig erstellt und auf OK geklickt haben, validiert MapForce die endgültige SQL-Anweisung. Falls Syntaxfehler vorhanden sind, werden Sie darüber in einem Dialogfeld informiert.

Die nachstehende Tabelle enthält eine Liste einiger typischer Operatoren, die in der WHERE-Klausel verwendet werden können:

Operator	Beschreibung
=	Ist gleich
<>	Ist nicht gleich
<	Kleiner als
>	Größer als
>=	Größer oder gleich
<=	Kleiner oder gleich
IN	Ruft einen bekannten Wert einer Spalte ab
LIKE	Sucht nach einem bestimmten Muster
BETWEEN	Sucht innerhalb eines Bereichs

Verwenden Sie das Platzhalterzeichen % (Prozent), um eine beliebige Anzahl von Zeichen in einem Muster darzustellen. Verwenden Sie z.B. den folgenden Ausdruck, um alle Datensätze, die mit "r" enden, aus einem Feld namens `lastname` abzurufen:

```
lastname = "%r"
```

Bei Abfragen an Datenbanken, die das Speichern und Abfragen von XML-Datenbankdaten unterstützen (z.B. IBM DB2, Oracle, SQL Server) können Sie XML-Funktionen und Schlüsselwörter für die jeweilige Datenbank verwenden, z.B.:

```
xmlexists('$c/Client/Address[zip>"55116"]' passing USER.CLIENTS.CONTACTINFO AS "c")
```

Siehe auch Beispiel: Extrahieren von Daten aus IBM DB2-Spalten vom Typ XML 384.

Wenn Sie die abgerufenen Datensätze optional nach einem bestimmten Feld sortieren möchten, fügen Sie im entsprechenden Textfeld des Dialogfelds **SQL/NoSQL-WHERE/ORDER-Eigenschaften** eine `ORDER BY`-Klausel hinzu. Um nach mehreren Feldern zu sortieren, trennen Sie die Felder durch Kommas. Sie können die Sortierreihenfolge mit Hilfe der Schlüsselwörter `ASC` und `DESC` ändern. So werden z.B. mit der folgenden `ORDER BY`-Klausel Datensätze in absteigender Reihenfolge zuerst nach `lastname` und anschließend nach `firstname` sortiert:

```
lastname, firstname DESC
```

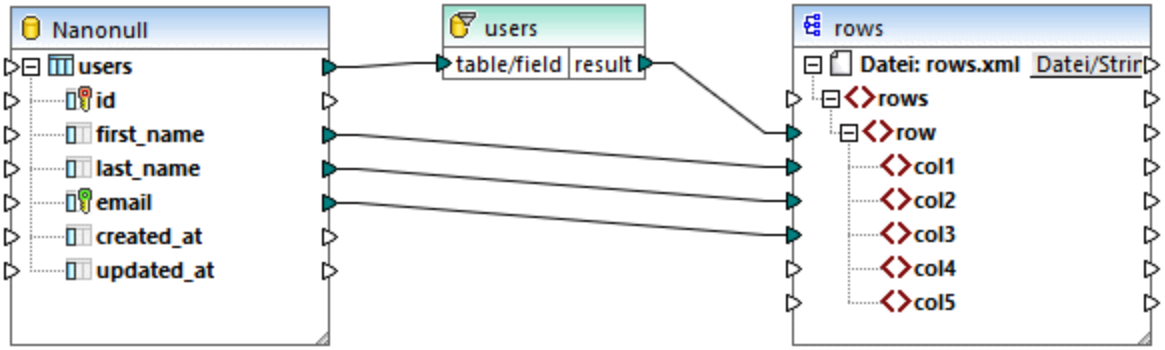
Beispiel 1

Die folgende WHERE-Bedingung wird in der Datenbankkomponente **`Nanonull.sqlite`** an die Tabelle `users` angehängt. Sie ruft diejenigen Datensätze ab, in denen `last_name` größer als der Buchstabe "M" ist, d.h. es werden alle Namen ab dem Benutzer "Marzolla" abgerufen.

```
last_name > "M"
```

Beachten Sie, wie die Verbindungen platziert wurden:

- Die Verbindung zum Parameter `table/field` ist mit der Quelltabelle verbunden, an der Sie die Abfrage durchführen möchten (in diesem Fall mit `users`).
- Der Parameter `result` ist mit einem übergeordneten Datenelement der Felder verbunden, die abgefragt/gefiltert werden, in diesem Fall mit dem Datenelement `row`.

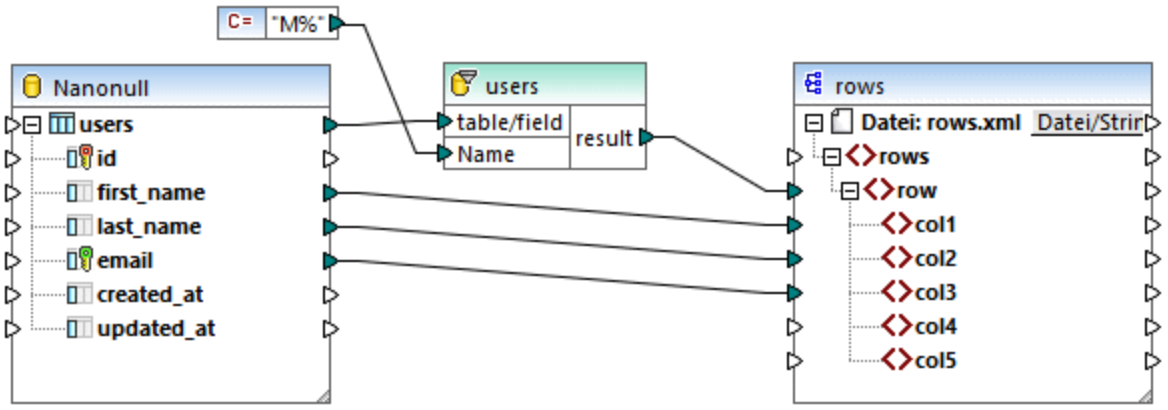


Beispiel 2

Mit der folgenden WHERE-Klausel wird ein Parameter `param` erstellt, der anschließend im Mapping in der **SQL/NoSQL-WHERE/ORDER**-Komponente aufscheint.

```
last_name LIKE :param
```


© 2016-2022 Altova GmbH

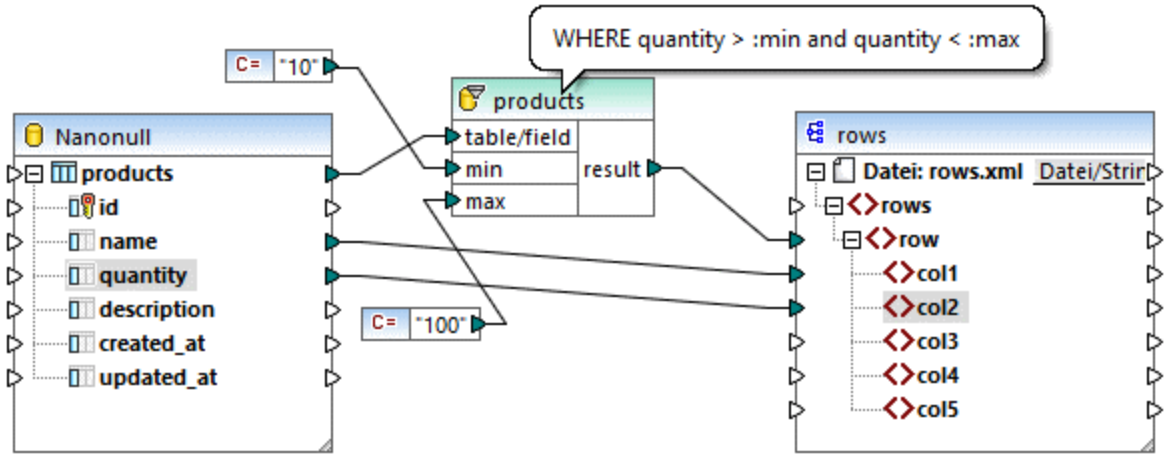


Die Konstantenkomponente `%M` liefert den Wert von `param`. Der Platzhalter % steht für beliebig viele Zeichen. Dadurch wird im Mapping nach einem Muster in der Spalte `last_name` gesucht (alle Nachnamen, die mit "M" beginnen).

Beispiel 3

Mit der folgenden WHERE-Anweisung werden zwei Parameter erstellt, `min` und `max`, mit denen die aktuellen Werte von `quantity` verglichen werden. Die Werte min und max stammen aus den zwei Konstantenkomponenten aus dem Mapping.

```
quantity > :min and quantity < :max
```



Die WHERE-Bedingung in diesem Beispiel könnte auch mit Hilfe des Operators BETWEEN erstellt werden.

```
quantity BETWEEN :min and :max
```

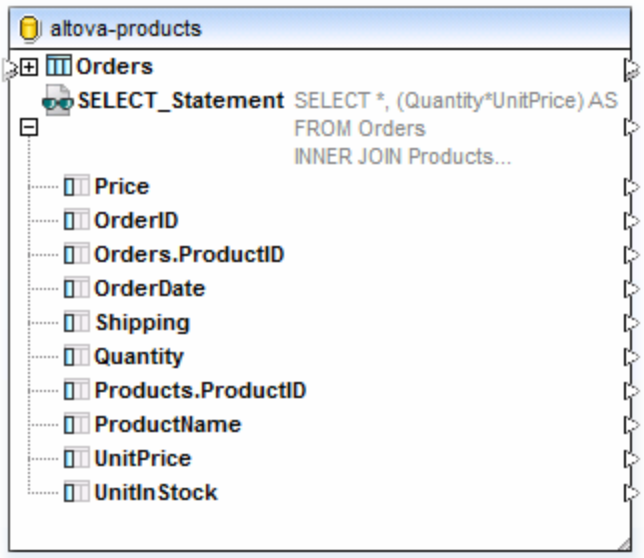
4.2.6 SQL SELECT-Anweisungen als virtuelle Tabellen

MapForce unterstützt nun auch die Erstellung von SQL SELECT-Anweisungen in Datenbankkomponenten. Es handelt sich hierbei um tabellenartige Strukturen, die die Felder der mit der SELECT-Anweisung generierten Tabellen enthalten. Diese Strukturen können anschließend wie jede andere in der Datenbank definierte Tabelle oder Ansicht als Datenquelle für das Mapping verwendet werden.

- Wenn Sie die inneren/äußeren **Joins** in der Select-Anweisung verwenden, werden Felder aller Tabellen in die Komponente inkludiert.
- Ausdrücke mit Entsprechungsnamen werden ebenfalls (unter Verwendung des SQL-Schlüsselworts "AS") als mapbare Datenelemente in der Komponente angezeigt,
- Auch Datenbankansichten können in der FROM-Klausel verwendet werden.
- SELECT-Anweisungen können Parameter enthalten, die dieselbe Syntax wie die SQL WHERE/ORDER[361]-Komponente verwenden.

SQL SELECT-Anweisungen werden in den Sprachen C++, C#, Java und BUILT-IN unterstützt, solange sie keine Input-Parameter enthalten. SQL SELECT-Anweisungen mit Input-Parametern werden nur in der Transformationssprache BUILT-IN unterstützt.

Sobald die SELECT-Anweisung zur Datenbankkomponente hinzugefügt wurde, stehen die von der Anweisung zurückgegebenen Felder für das Mapping zur Verfügung, z.B.:



Die Anzahl der sichtbaren Zeilen der SELECT-Anweisung ist konfigurierbar. Um die Anzahl der Zeilen, die in der Komponente angezeigt werden sollen, zu definieren, wählen Sie die Menüoption **Extras | Optionen**, klicken Sie auf das Register "**Allgemein**" und geben Sie in der Gruppe "Mapping-Ansicht" die Anzahl der Zeilen ein.

In diesem Abschnitt wird die Datei **altova-products.mdb** im Ordner **...\MapForceExamples\Tutorial** verwendet. Auch das Mapping **select-component.mfd,** das ein Beispiel enthält, befindet sich im selben Ordner.

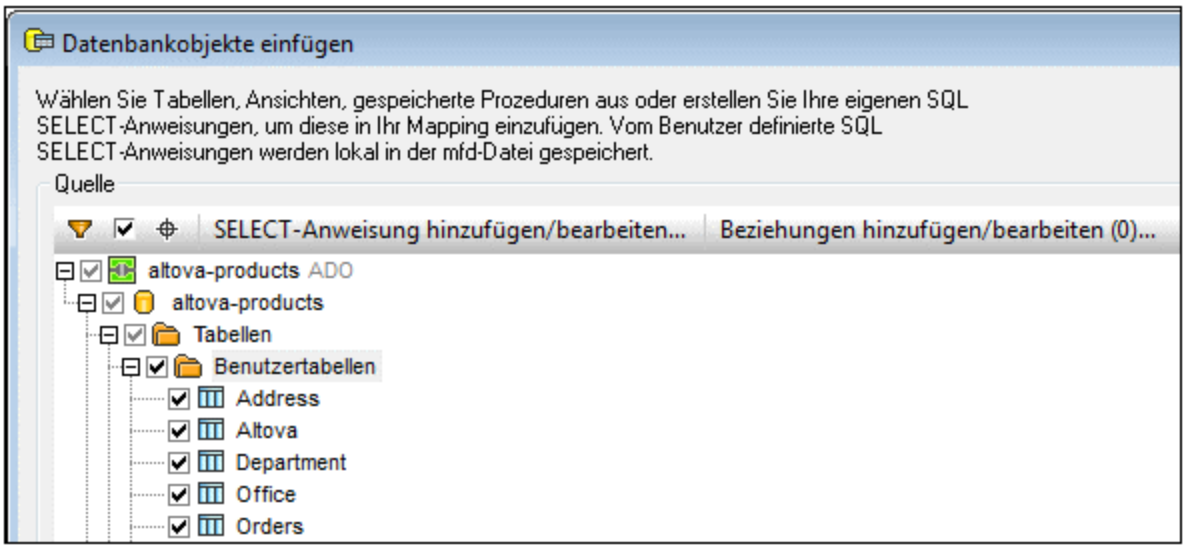
© 2016-2022 Altova GmbH

4.2.6.1 Erstellen von SELECT-Anweisungen

Sie können in jedem Mapping, das eine Datenbankkomponente enthält, SELECT-Anweisungen erstellen. Wenn Ihr Mapping noch keine Datenbank enthält, fügen Sie zuerst eine hinzu (siehe Herstellen einer Verbindung zu einer Datenbank 167). Wählen Sie den Menübefehl **Einfügen | Datenbank einfügen,** befolgen Sie die Anweisungen des Datenbankassistenten und stellen Sie eine Verbindung zu der in diesem Beispiel verwendeten Datei **altova-products.mdb** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** her.

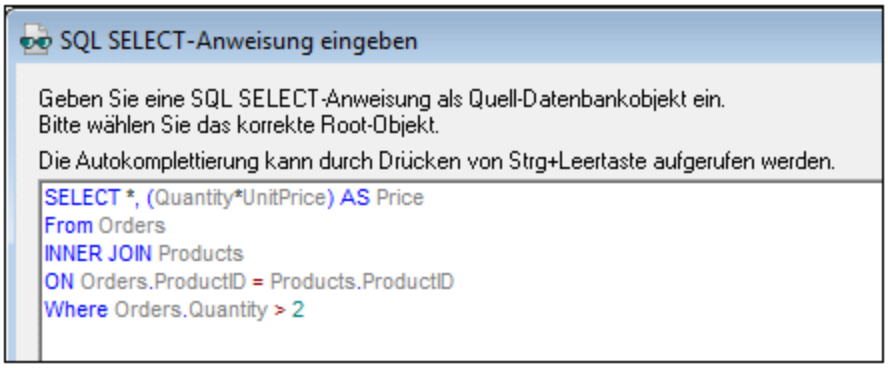
So erstellen Sie eine SELECT-Anweisung:

1. Klicken Sie mit der rechten Maustaste auf den Titel der Datenbankkomponente und wählen Sie den Befehl **Datenbankobjekte hinzufügen/entfernen/bearbeiten** aus. (Alternativ dazu können Sie auch die Datenbankkomponente auswählen und anschließend den Menübefehl **Komponente | Datenbankobjekte hinzufügen/entfernen/bearbeiten** auswählen).

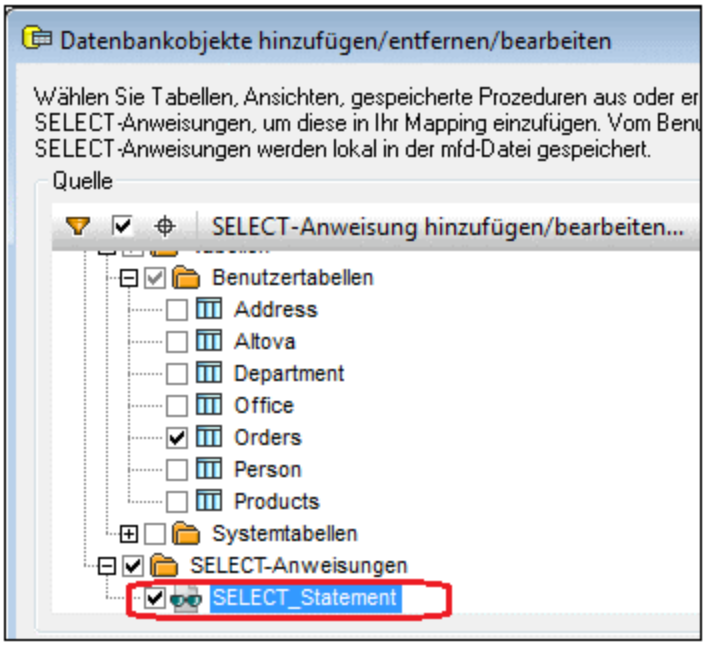


2. Wählen Sie eine der folgenden Methoden:
   - Um anhand einer vorhandenen Tabelle eine SELECT-Anweisung zu generieren, klicken Sie mit der rechten Maustaste auf eine Tabelle und wählen Sie im Kontextmenü den Befehl **SELECT-Anweisung erzeugen und hinzufügen....** Nach Auswahl dieses Befehls können Sie die generierte Anweisung anschließend bearbeiten.
   - Um eine benutzerdefinierte SELECT-Anweisung zu schreiben, klicken Sie auf die Schaltfläche **SELECT-Anweisung hinzufügen/bearbeiten.**
3. Bearbeiten bzw. erstellen Sie die Anweisung, wie benötigt. Die unten stehende SELECT-Anweisung ist z.B. für die Datei **altova-products.mdb** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** gültig. Das Feld **Price** ist das Produkt der beiden Felder **Quantity** und **UnitPrice** und wird als Korrelationsname (`AS Price`) deklariert.

```
SELECT *, (Quantity*UnitPrice) AS Price
From Orders
INNER JOIN Products
ON Orders.ProductID = Products.ProductID
Where Orders.Quantity > 2
```

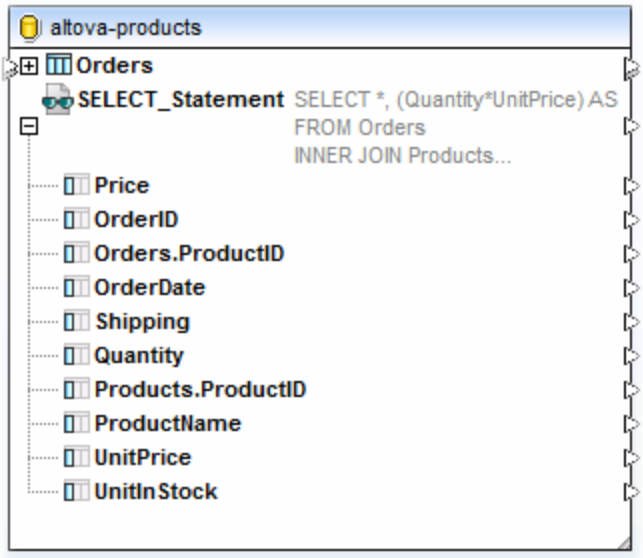


4. Klicken Sie auf **SELECT-Anweisung hinzufügen**. Beachten Sie, dass die SELECT-Anweisung nun als Datenbankobjekt sichtbar ist, genauso, wie Tabellen, Ansichten und Prozeduren sichtbar sind.



5. Klicken Sie auf OK. Die SELECT-Anweisung wird auch in der Datenbankkomponente angezeigt und Sie können von jedem der von der SELECT-Abfrage zurückgegebenen Felder aus Daten mappen.

© 2016-2022 Altova GmbH



Wichtige Hinweise:

- Alle berechneten Ausdrücke in der SELECT-Anweisung müssen einen eindeutigen Korrelationsnamen haben (wie in diesem Beispiel "AS Price"), um als mapbares Element zur Verfügung zu stehen.
- Wenn Sie über JDBC eine Verbindung zu einer Oracle- oder IBM DB2-Datenbank herstellen, darf die SELECT-Anweisung kein Semikolon am Ende aufweisen.

So entfernen Sie eine zuvor hinzugefügte SELECT-Anweisung:

1. Klicken Sie mit der rechten Maustaste auf den Titel der Datenbankkomponente und wählen Sie den Befehl **Datenbankobjekte hinzufügen/entfernen/bearbeiten**.
2. Klicken Sie mit der rechten Maustaste auf die zu löschende SELECT-Anweisung und wählen Sie den Befehl **SELECT-Anweisung entfernen.**

4.2.6.2 Beispiel: SELECT mit Parametern

In diesem Beispiel wird gezeigt, wie Sie ein MapForce-Mapping erstellen, das Daten aus einer Microsoft Access-Datenbank ausliest und in eine CSV-Datei schreibt. In dem hier beschriebenen Mapping wird eine benutzerdefinierte SELECT-Datenbankabfrage mit einem Parameter verwendet. Die SELECT-Anweisung kombiniert Daten aus mehreren Tabellen miteinander. Die Ergebnisse davon werden dem Mapping anschließend für die weitere Verarbeitung zur Verfügung gestellt.

Die Mapping-Datei (.mfd) zu diesem Beispiel finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\select-component.mfd**. Sie können die Beispieldatei zuerst öffnen und einen Blick darauf werfen oder sie mit Hilfe der unten beschriebenen Schritte von Grund auf neu erstellen.

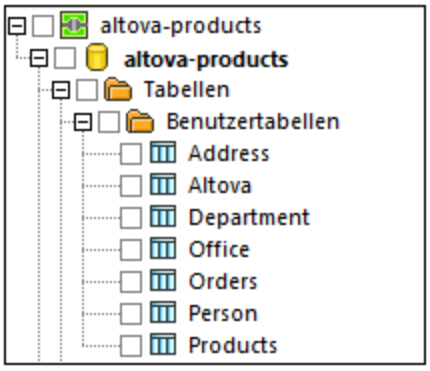
Zwar wird in diesem Beispiel eine Microsoft Access-Datenbank verwendet, doch ist die Vorgangsweise auch bei anderen Datenbanktypen dieselbe. Informationen zum Herstellen einer Verbindung zu anderen Datenbanken finden Sie unter Herstellen einer Verbindung zu einer Datenbank 167.

Mit dem Mapping möchten wir folgendes Ergebnis erzielen:

1. Wir müssen aus der Datenbank nur diejenigen Bestellungen auswählen, bei denen die Anzahl der bestellten Posten einen benutzerdefinierten Wert übersteigt. Dieser benutzerdefinierte Wert sollte dem Mapping als Parameter bereitgestellt werden. Zu diesem Zweck wollen wir eine benutzerdefinierte SELECT-Anweisung mit einem Input-Parameter erstellen.
2. Das Datumsformat in der Access-Datenbank ist `JJJJ-MM-TT HH-MI-SS`. Der Uhrzeitteil muss in der CSV-Datei entfernt werden, sodass das Format `JJJJ-MM-TT` lautet. Wir verwenden dazu die MapForce-Funktion date-from-datetime[1136].
3. Die erzeugte CSV-Datei muss den Namen **OrdersReport.csv** erhalten.

Schritt 1: Hinzufügen der SELECT-Struktur

1. Klicken Sie im Menü **Einfügen** auf **Datenbank**.
2. Wählen Sie **Microsoft Access (ADO)** und befolgen Sie die Anweisungen des Assistenten, um eine Verbindung zur Datei **altova-products.mdb** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** herzustellen.

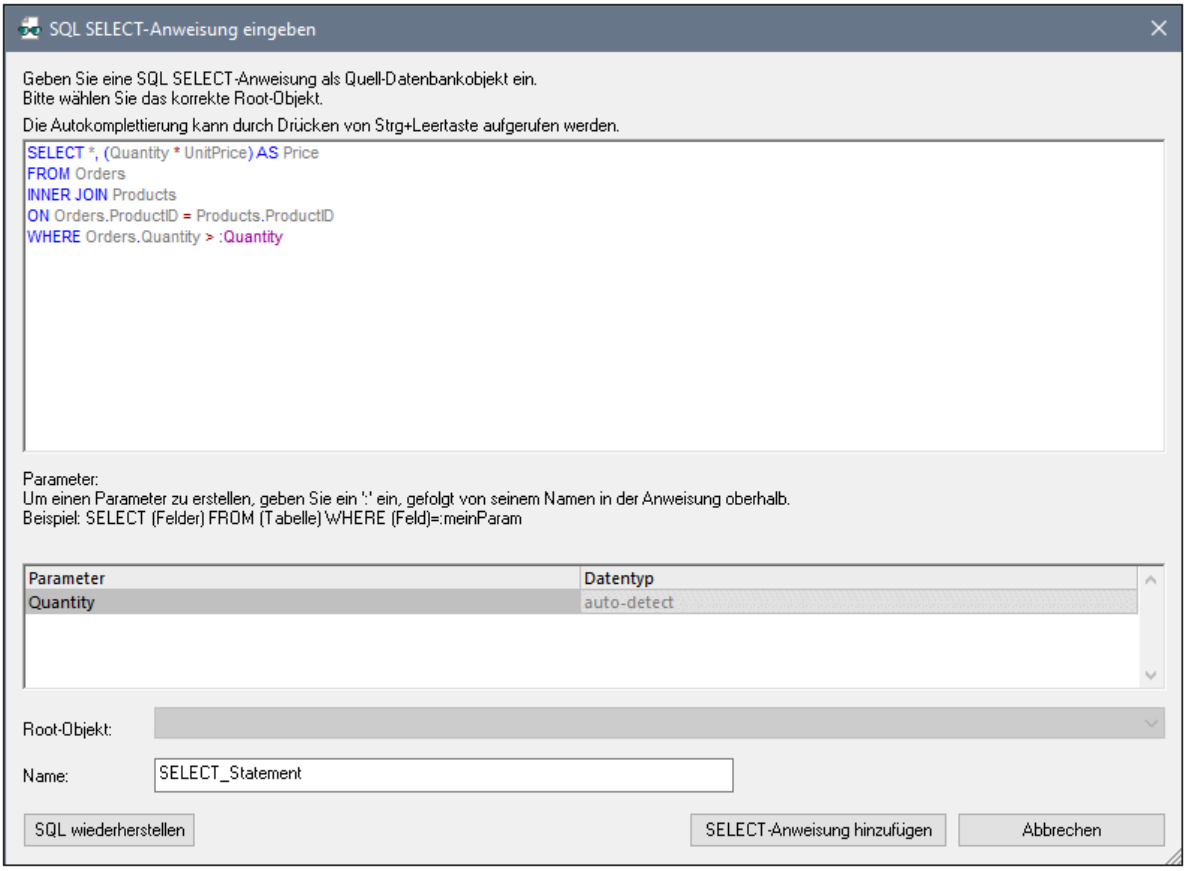


3. Klicken Sie im Dialogfeld "Datenbankobjekte einfügen" auf **SELECT-Anweisung hinzufügen/bearbeiten** und geben Sie die folgende Abfrage ein:

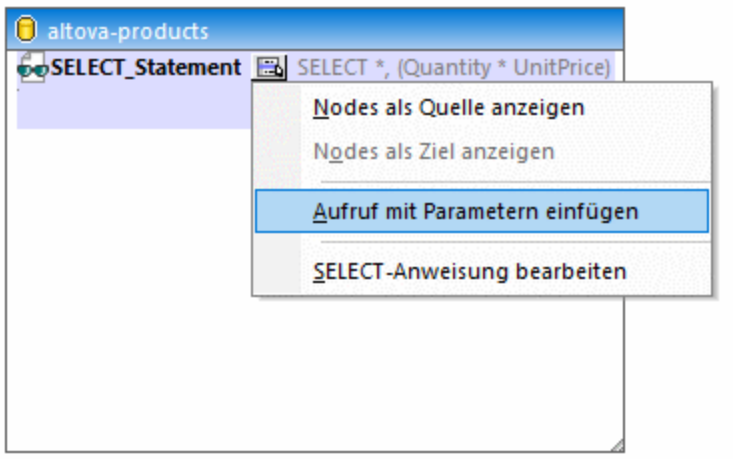
```
SELECT *, (Quantity * UnitPrice) AS Price
FROM Orders
INNER JOIN Products
ON Orders.ProductID = Products.ProductID
WHERE Orders.Quantity > :Quantity
```

In dieser Abfrage wird eine JOIN-Anweisung zwischen den Tabellen Orders und Products verwendet. Sie ruft alle Felder (`*`) sowie einen berechneten Wert (`AS Price`) ab. Außerdem ist in der Abfrage in der WHERE-Klausel der Parameter **:Quantity** definiert.

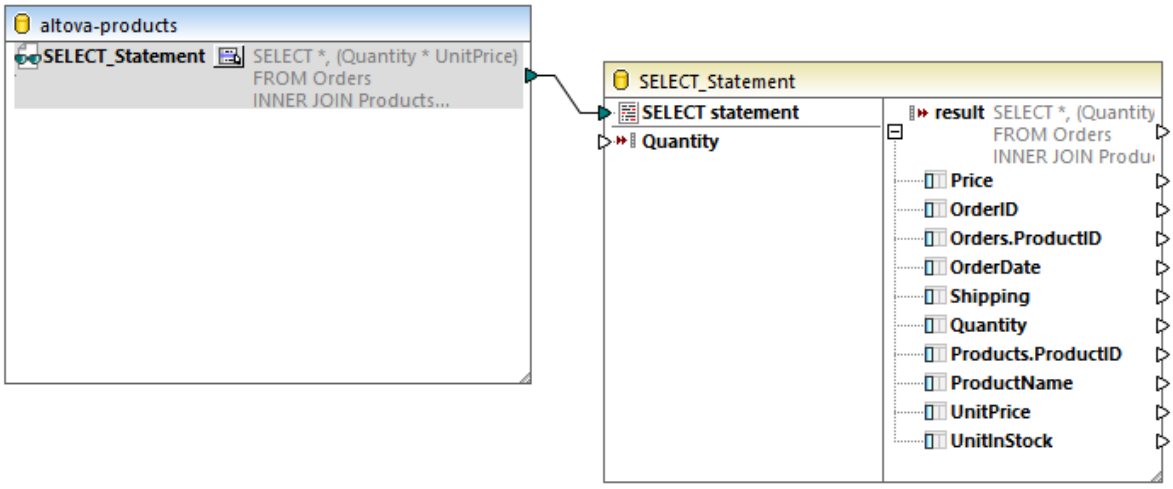
© 2016-2022 Altova GmbH



4. Klicken Sie auf **SELECT-Anweisung hinzufügen**.
5. Klicken Sie auf OK. Die Komponente **altova-products** wurde nun zum Mapping-Bereich hinzugefügt.
6. Klicken Sie in der Komponente **altova-products** auf  und wählen Sie den Befehl **Aufruf mit Parametern einfügen**.

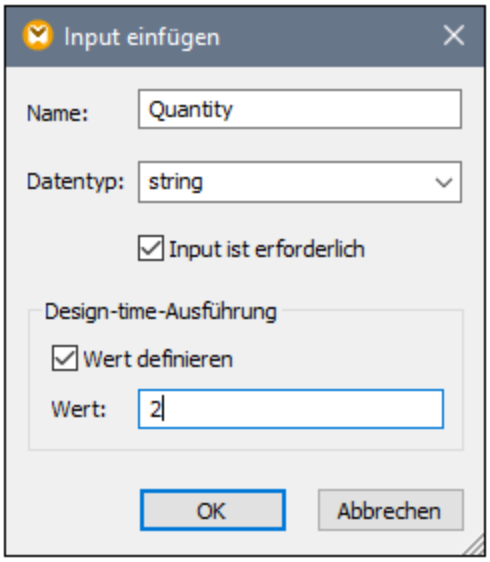


Im Mapping steht nun eine neue Struktur (SELECT_Statement) zur Verfügung. Sie besteht aus zwei Teilen: der linke Teil liefert Input-Konnektoren und der rechte Teil liefert Output-Konnektoren. Beachten Sie, dass der linke Teil auch den zuvor definierten Parameter **Quantity** enthält.



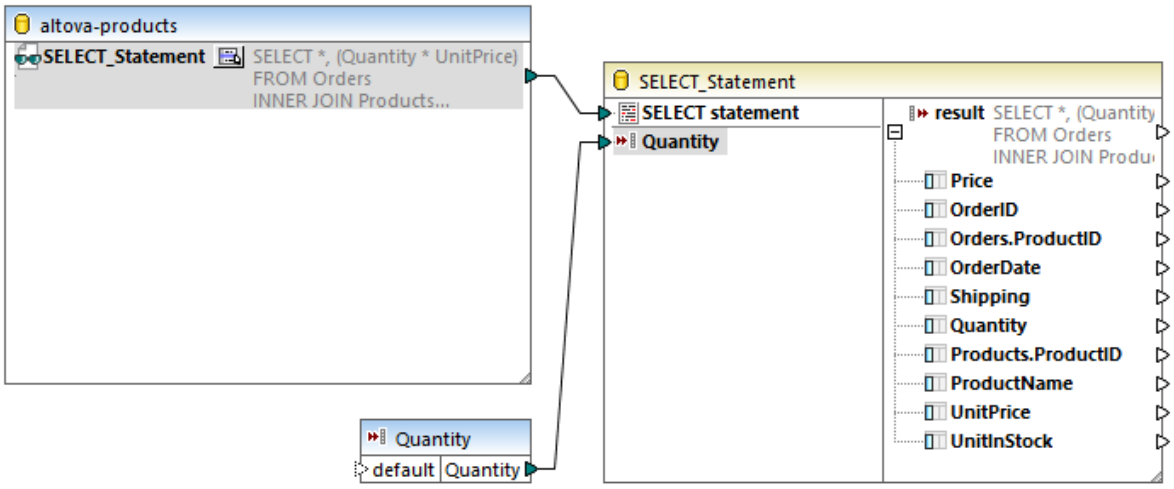
Schritt 2: Hinzufügen des Input-Parameters

1. Klicken Sie im Menü **Einfügen** auf **Input-Komponente einfügen**.
2. Geben Sie als Name "Quantity" ein.
3. Geben Sie unter **Design-time-Ausführung** einen Parameterwert ein, der für die Ausführung des Mappings während der Designphase verwendet werden soll (in diesem Beispiel, "2"). Nähere Informationen dazu finden Sie unter Bereitstellen von Parametern für das Mapping 867.



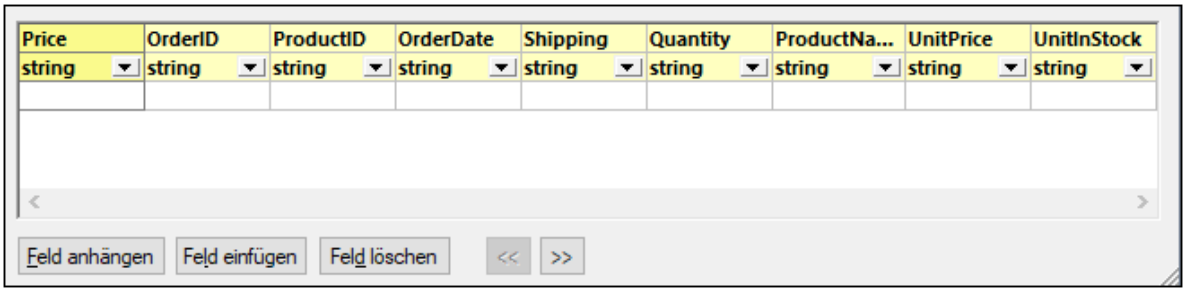
Sie können den Input-Parameter nun, wie unten gezeigt, mit der Datenbankaufrufstruktur verbinden.

© 2016-2022 Altova GmbH

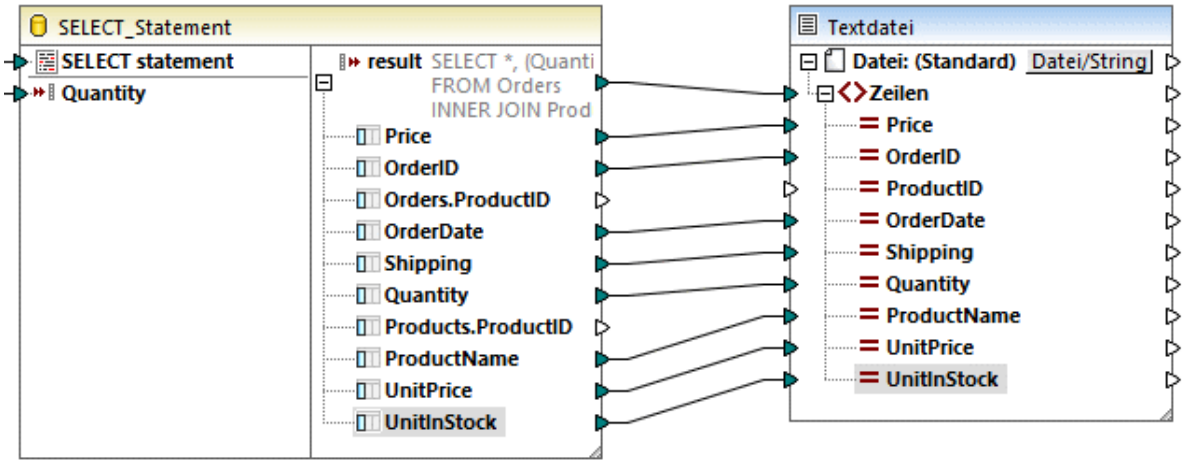


Schritt 3: Hinzufügen der CSV-Zielkomponente

1. Klicken Sie im Menü **Einfügen** auf **Textdatei**.
2. Wählen Sie die Option **Einfache Verarbeitung für Standard-CSV....verwenden** und klicken Sie auf **Weiter**.
3. Klicken Sie im Dialogfeld "Komponenteneinstellungen" auf **Feld anhängen** und fügen Sie neun neue Felder hinzu. Es wird empfohlen, den Feldern der CSV-Datei die Namen der Felder aus der Datenbank zu geben (siehe unten). Dadurch sparen Sie später beim Ziehen der Mapping-Verbindungen Zeit. Nähere Informationen zu diesen Einstellungen finden Sie unter Definieren der CSV-Optionen 595.



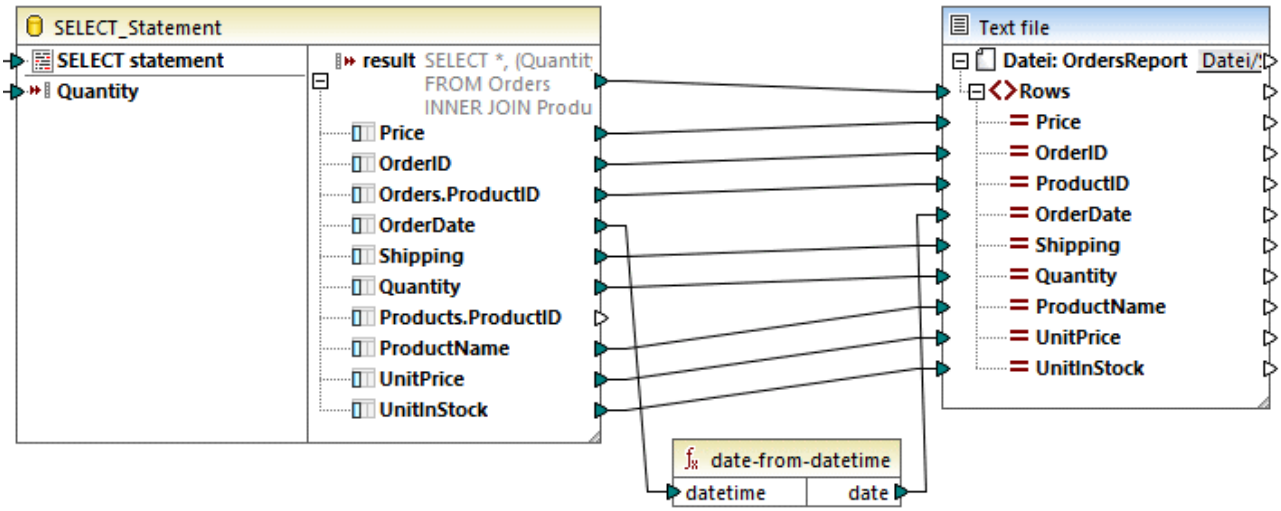
4. Erstellen Sie eine Verbindung zwischen dem **result**-Node der SELECT-Struktur und dem **Zeilen**-Node der CSV-Komponente.



Da die meisten Felder in der CSV-Komponente denselben Namen wie ihr Datenbankäquivalent haben, werden die Mapping-Verbindungen wahrscheinlich automatisch erstellt, wenn Sie **result** mit **Rows** verbinden. Stellen Sie andernfalls sicher, dass im Menü Verbindung die Option **Idente Sub-Einträge automatisch verbinden** aktiv ist. Das einzige Mapping-Datenelement, das Sie manuell verbinden müssen, ist **ProductID**, da es in der SELECT-Struktur kein Feld mit diesem Namen gibt.

Schritt 4: Datumskonvertierung

Suchen Sie im Fenster "Bibliotheken" nach der Funktion **`date-from-datetime`** und ziehen Sie diese in den Mapping-Bereich. Verbinden Sie anschließend ihren Input und Output wie unten gezeigt.



Schritt 5: Definieren des Namens der Ausgabedatei

Um als Namen der Ausgabedatei **OrdersReport.csv** zu definieren, doppelklicken Sie auf die Titelleiste der CSV-Komponente. Daraufhin wird das Dialogfeld "Komponenteneinstellungen" angezeigt, in dem Sie den Wert in das Feld **Ausgabedatei** eingeben können.

© 2016-2022 Altova GmbH

4.2.7 Mappen von XML-Daten von/auf Datenbankfelder

MapForce unterstützt das Mappen von Daten von oder auf Datenbankfelder (Spalten), in denen XML-Inhalte gespeichert sind. Dies bedeutet, dass im Datenbankfeld (Spalte) gespeicherte XML-Daten extrahiert und in jede andere von MapForce unterstützte Struktur geschrieben werden können (und umgekehrt). Die folgenden Datenmappings sind möglich:

1. Das Mappen von Daten von oder auf Felder eines ausgewiesenen XML-Typs (z.B. `Xml` in SQL Server, `XMLType` in Oracle). Das Lesen aus oder Schreiben von XML-Daten von/in spezielle XML-Felder ist bei Datenbanken mit nativer Unterstützung für XML möglich (z.B. IBM DB2, Oracle und SQL Server).
2. Das Mappen von oder auf Textfelder mit XML-Inhalt (z.B. `Text, Varchar`). Dies gilt für alle Datenbanken, deren Textfelder lang genug sind, um ein XML-Dokument speichern zu können.

In beiden Fällen muss für jede Datenbankspalte, von oder auf die Sie Daten mappen möchten, ein gültiges XML-Schema vorhanden sein. Wenn in einer Datenbankspalte XML-Inhalte gespeichert sind, haben Sie in MapForce die Wahl, entweder direkt aus der Datenbank ein XML-Schema zuzuweisen (falls dies von der Datenbank unterstützt wird) oder ein Schema aus einer externen Datei auszuwählen. Sie können ein XML-Schema pro Datenbankspalte zuweisen. Wenn das Schema mehrere Root-Elemente hat, können Sie ein einzelnes Root-Elemente dieses Schemas auswählen.

Wenn XML-Daten in einer Datenbank als String-Feld gespeichert sind, so ist die Zeichenkodierung des XML-Dokuments die des zugrunde liegenden String-Felds. Wenn Text im Datenbankfeld nicht als Unicode gespeichert ist, können einige Zeichen nicht dargestellt werden.

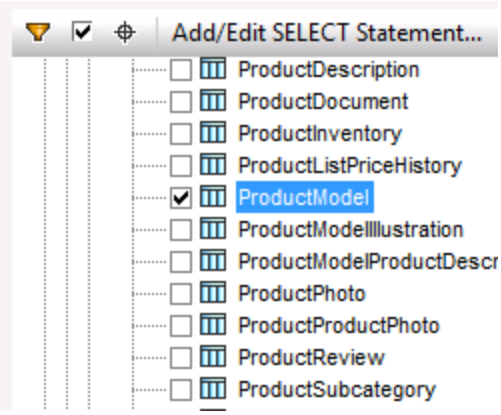
Einige Datenbanken unterstützen XML-Kodierung für XML-Felder (welche nicht notwendigerweise mit der des Datenbankzeichensatzes übereinstimmen muss). Wenn dies von der Datenbank unterstützt wird, wird als XML-Dokumentkodierungsdeklaration die im XML-Feld deklarierte angenommen. Informationen zur Unterstützung für XML-Kodierung in den verschiedenen Datenbanken finden Sie in der jeweiligen Dokumentation.

4.2.7.1 Zuweisen eines XML-Schemas zu einem Datenbankfeld

In diesem Kapitel wird beschrieben, wie Sie einem nativ als XML-Typ definierten Feld in der Datenbank ein Schema zuweisen. In der Anleitung unten werden SQL Server 2014 und die Adventure Works 2014-Datenbank verwendet. Zweitere kann von der AdventureWorks-Beispielseite auf GitHub (https://github.com/Microsoft/sql-server-samples/releases/tag/adventureworks) heruntergeladen werden. Beachten Sie, dass das Mappen von Daten von oder auf XML-Felder auch bei anderen Datenbanktypen, die XML-Felder unterstützen, auf dieselbe Art und Weise funktioniert.

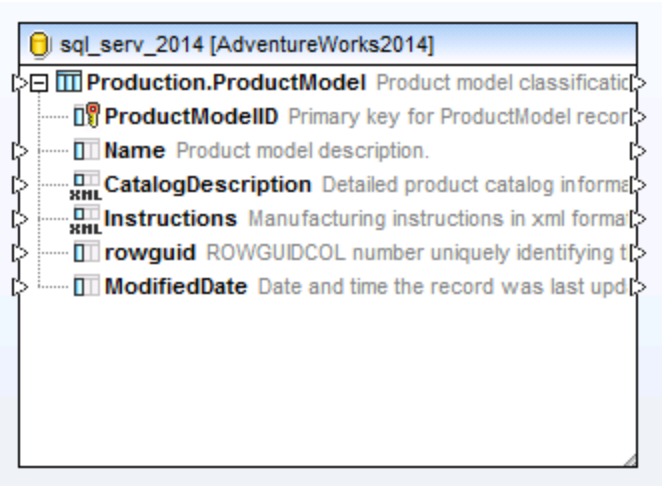
So fügen Sie die Adventure Works 2014-Datenbank als Mapping-Komponente hinzu:

1. Klicken Sie im Menü **Einfügen** auf **Datenbank** und befolgen Sie die Anweisungen des Assistenten, um die Verbindung zur Datenbank mit der gewünschten Methode (ADO oder ODBC) herzustellen. Nähere Informationen dazu finden Sie unter ADO-Verbindungn 174 and ODBC-Verbindung 187. ANMERKUNG: Wenn Sie den **SQL Server Native Client**-Treiber verwenden, müssen Sie die Eigenschaft **Integrated Security** eventuell auf ein Leerzeichen setzen (siehe Einrichten der SQL Server-Datenverknüpfungseigenschaften 177).
2. Erweitern Sie im Dialogfeld **Insert Database Object** das Schema **Production** und wählen Sie anschließend die Tabelle **ProductModel** aus.



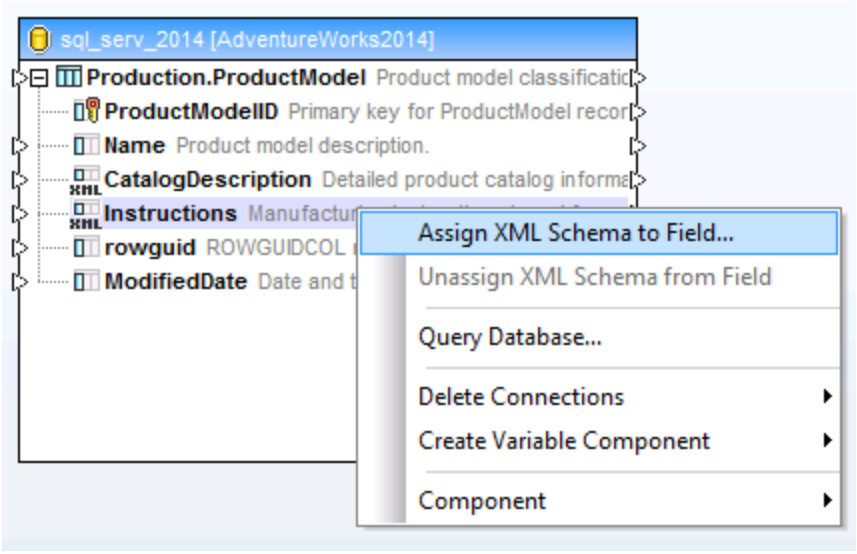
3. Klicken Sie auf OK.

Die Datenbanktabelle wurde nun zum Mapping-Bereich hinzugefügt. Beachten Sie, dass diese Tabelle zwei Felder vom Typ XML hat: **CatalogDescription** und **Instructions**:

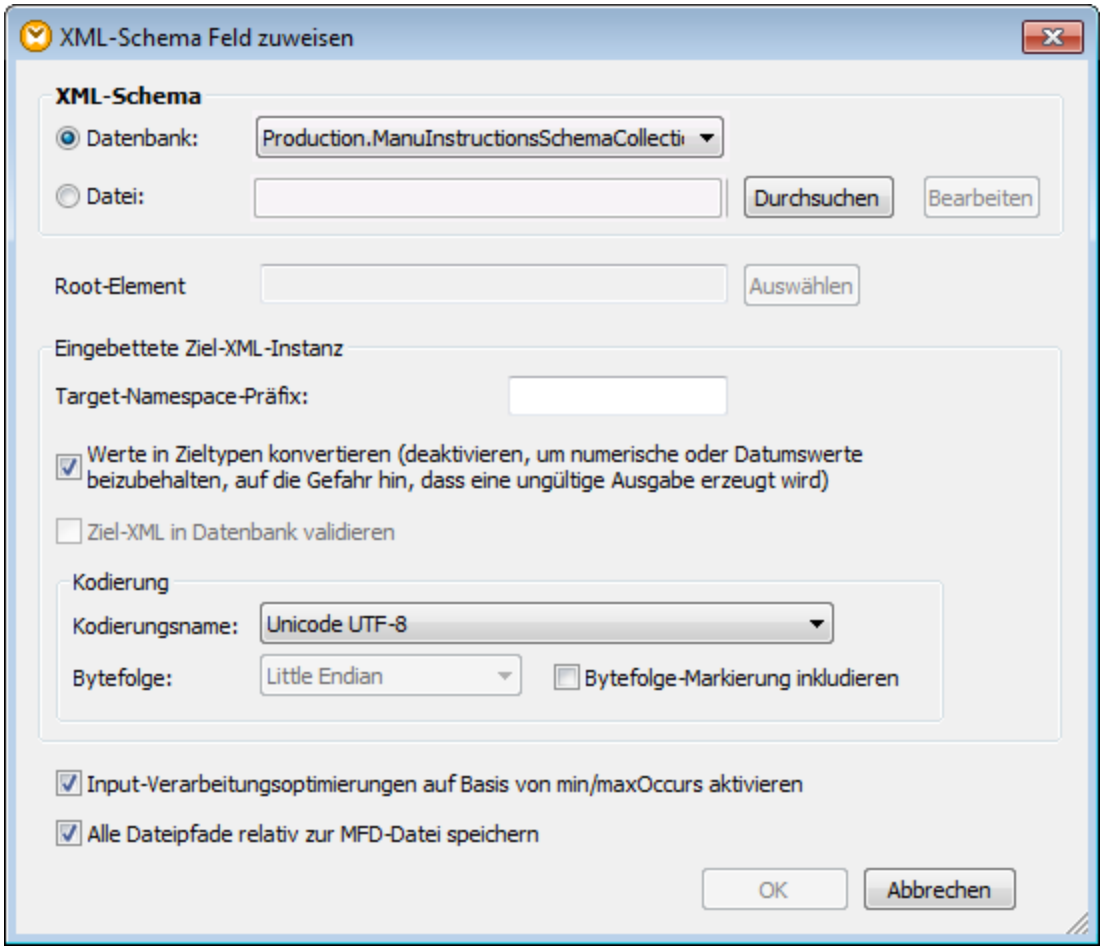


Damit die Struktur der XML-Felder im Mapping angezeigt wird, wird das XML-Schema des Feldinhalts benötigt. Klicken Sie mit der rechten Maustaste auf das Feld **Instructions** und wählen Sie im Kontextmenü den Befehl **XML-Schema Feld zuweisen.**

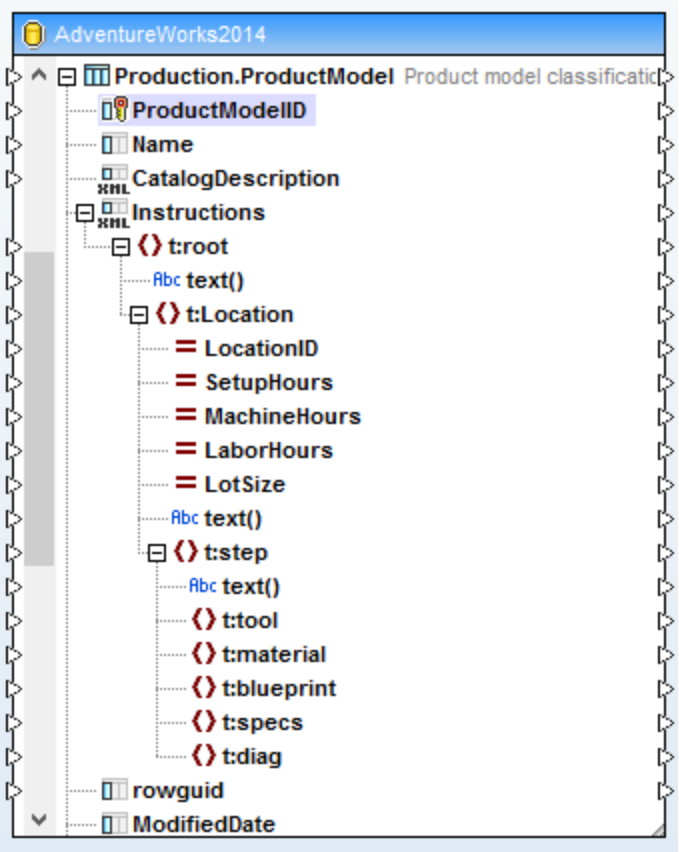
© 2016-2022 Altova GmbH



In diesem Beispiel werden wir dem Feld **Instructions** das Schema direkt über die Datenbank zuweisen. Wählen Sie dazu neben der Option **Datenbank** das Datenelement **Production.ManuInstructionsSchemaCollection** aus und klicken Sie auf OK.



Die Struktur des XML-Felds wird nun in der Komponente angezeigt. Sie können nun Verbindungen von oder zu diesem Feld ziehen (und Daten mappen).



4.2.7.2 Beispiel: Schreiben von XML-Daten in ein SQLite-Feld

In diesem Beispiel wird Schritt für Schritt beschrieben, wie Sie ein MapForce-Mapping erstellen, in dem Daten aus mehreren XML-Dateien gelesen und in eine SQLite-Datenbank geschrieben werden. Ziel des Mappings ist die Erstellung eines neuen Datenbankdatensatzes in der SQLite-Datenbank für jede XML-Quelldatei. Das XML-Dokument wird in den einzelnen Datensätzen als TEXT-Feld gespeichert.

Alle in diesem Beispiel verwendeten Dateien stehen unter dem folgenden Pfad zur Verfügung: **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Es werden die folgenden Dateien verwendet:

Die Mapping-Design-Datei	• **XmlToSqliteField.mfd**
Die XML-Quelldateien	• **bookentry1.xml** • **bookentry2.xml** • **bookentry3.xml**
Das für die Validierung verwendete XML-Schema	• **books.xsd**
Die SQLite-Zieldatenbank	• **Library.sqlite**

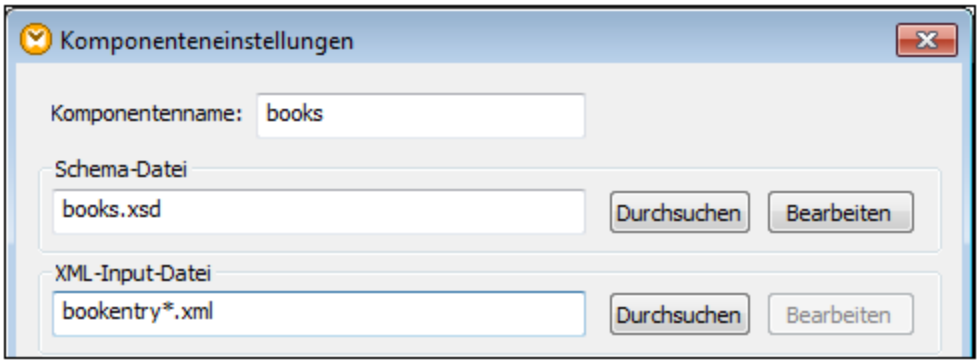
© 2016-2022 Altova GmbH

Die folgenden Schritte müssen durchgeführt werden, um das gewünschte Resultat zu erzielen:

1. Fügen Sie die XML-Komponente hinzu und konfigurieren Sie sie so, dass sie den Inhalt aus mehreren Dateien liest.
2. Fügen Sie die SQLite-Datenbankkomponente hinzu und weisen Sie dem TEXT-Zielfeld ein XML-Schema zu.
3. Erstellen Sie die Mapping-Verbindungen und konfigurieren Sie die Datenbank-INSERT-Aktion.

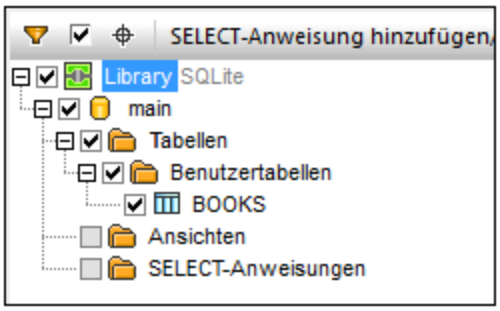
Schritt 1: Hinzufügen der XML-Komponente

1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie im Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples** zum Schema **books.xsd**. Wenn Sie aufgefordert werden, eine XML-Beispieldatei anzugeben, klicken sie auf **Überspringen**. Wenn Sie aufgefordert werden, ein Root-Element auszuwählen, wählen Sie **Books** aus.
2. Doppelklicken Sie auf die Komponentenüberschrift und geben Sie in das Feld **XML-Input-Datei bookentry*.xml** ein. Dadurch liest MapForce alle XML-Dateien, deren Name im Quellverzeichnis mit "bookentry-" beginnt. Nähere Informationen zu diesem Verfahren finden Sie unter Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 1371.

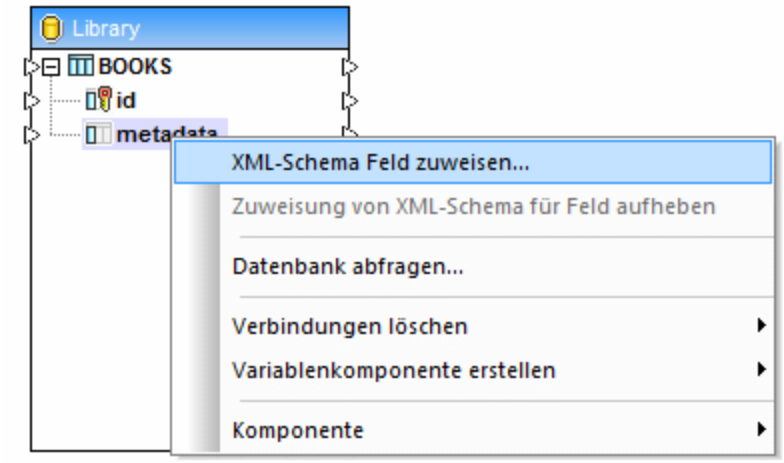


Schritt 2: Hinzufügen der SQLite-Komponente

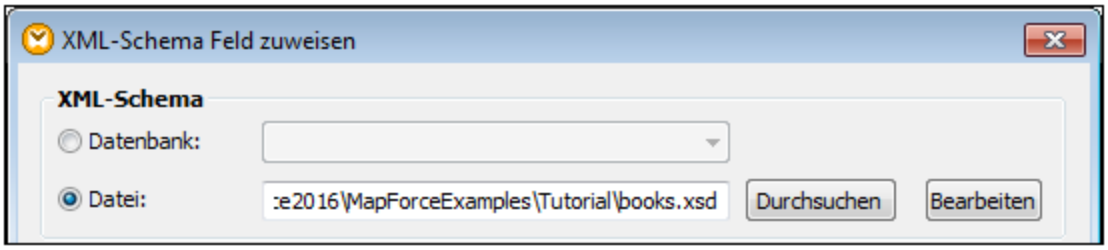
Klicken Sie im Menü **Einfügen** auf **Datenbank** und befolgen Sie die Anweisungen des Assistenten, um eine Verbindung zur Datenbankdatei **Library.sqlite** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples** herzustellen (siehe auch Herstellen einer Verbindung zu einer bestehenden SQLite-Datenbank 197). Wenn Sie aufgefordert werden, die Datenbankobjekte auszuwählen, wählen Sie die Tabelle BOOKS aus.



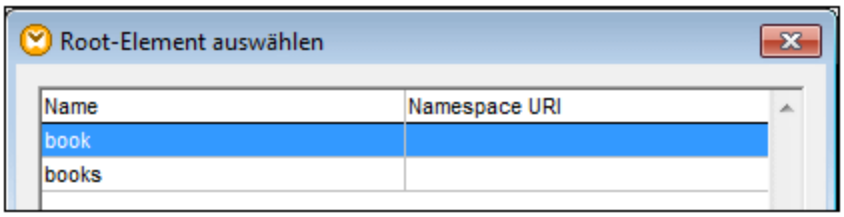
Das Datenbankfeld, in das XML-Inhalt geschrieben werden soll, heißt `metadata`. Um diesem Feld ein XML-Schema zuzuweisen, klicken Sie mit der rechten Maustaste darauf und wählen Sie im Kontextmenü den Befehl **XML-Schema Feld zuweisen**.



Das dem Feld `metadata` zugewiesene Feld in diesem Tutorial ist dasselbe, das auch zum Validieren der XML-Quelldateien verwendet wird. Klicken Sie auf **Durchsuchen** und wählen Sie das Schema **books.xsd** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples** aus:



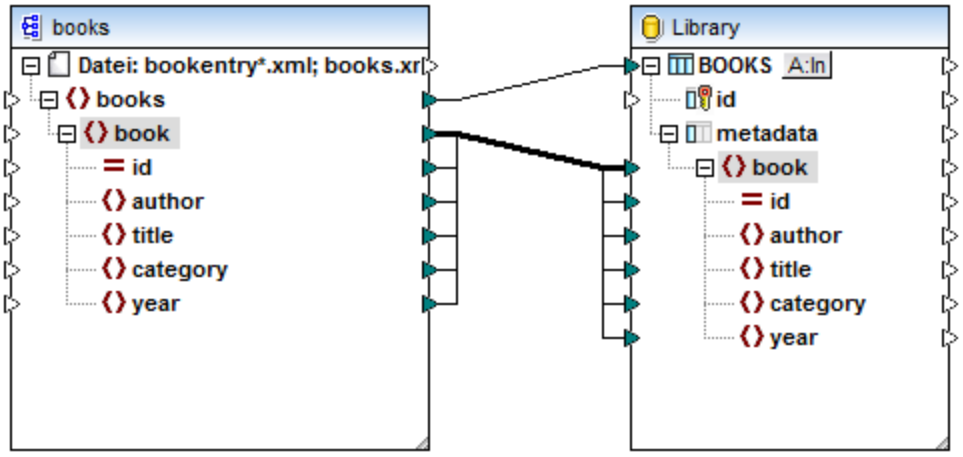
Das Schema **books.xsd** hat zwei Elemente mit einer globalen Deklaration: `book` und `books`. Wir definieren `book` in diesem Beispiel als das Root-Element der XML-Daten, die in das Datenbankfeld geschrieben werden. Klicken Sie auf **Auswählen** und wählen Sie `book` als Root-Element aus:



© 2016-2022 Altova GmbH

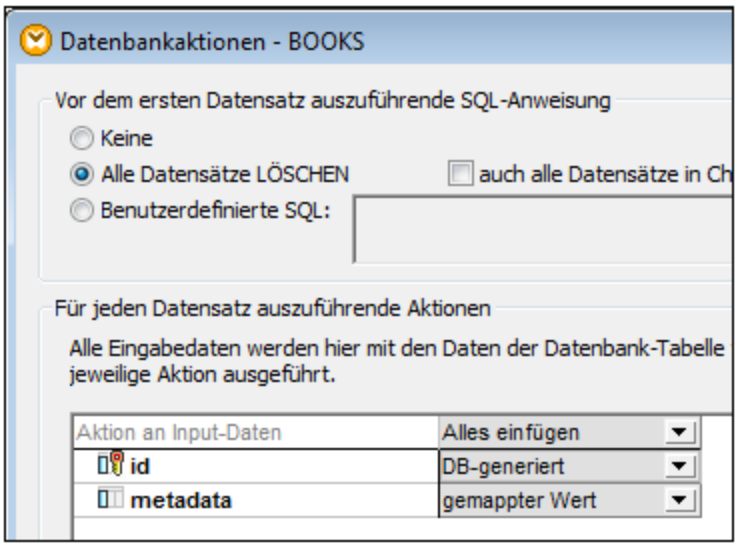
Schritt 3: Erstellen der Mapping-Verbindungen und Konfigurieren der Datenbank-INSERT-Aktion

Erstellen Sie die folgenden Mapping-Verbindungen:



Wie Sie in der Abbildung oben sehen, handelt es sich bei der Verbindung von `book` zu `book` um eine "Alles kopieren"-Verbindung, da im Quell- und Zielschema dasselbe Schema verwendet wird und die Namen der Sub-Einträge dieselben sind. Nähere Informationen zu solchen Verbindungen finden Sie unter "Alles kopieren"-Verbindungen 97.

Die oberste Verbindung (`books` zu `BOOKS`) iteriert durch die einzelnen book-Elemente in der Quelldatei und schreibt neue Datensätze in die Tabelle BOOKS. Klicken Sie auf die `A:In` -Schaltfläche der Datenbankkomponente und konfigurieren Sie die folgenden Datenbankaktualisierungseinstellungen:



Mit der Option **Alle Datensätze LÖSCHEN** wird MapForce angewiesen, den Inhalt der Tabelle `BOOKS` zu löschen, bevor neue Datensätze eingefügt werden.

Mit den **Alle einfügen**-Aktionen wird festgelegt, dass eine Datenbank `INSERT`-Abfrage durchgeführt wird. Das Feld `id` wird anhand der Datenbank selbst generiert, während das Feld `metadata` mit dem Wert aus dem Mapping befüllt wird.

Vergessen Sie nicht, das Mapping vor der Ausführung zu speichern.

Um das Mapping auszuführen und die generierte Ausgabe anzuzeigen, klicken Sie auf das Register **Ausgabe**. Beachten Sie, dass die Datenbank dadurch noch nicht sofort aktualisiert wird. Wenn Sie bereit sind, das generierte Datenbank-Skript auszuführen, wählen Sie den Menübefehl **Ausgabe | SQL-Script ausführen** (oder klicken Sie auf die Symbolleisten-Schaltfläche ).

4.2.7.3 Beispiel: Extrahieren von Daten aus IBM DB2-Spalten vom Typ XML

In diesem Beispiel wird gezeigt, wie Sie Daten aus IBM DB2-Datenbankspalten vom Typ XML extrahieren und in eine CSV-Zieldatei schreiben. Außerdem wird hier gezeigt, wie Sie XML-Inhalte mit Hilfe von in SQL eingebetteten XQuery-Anweisungen auf Basis von Bedingungen abrufen. Sie benötigen für dieses Beispiel Zugriff auf eine IBM DB2-Datenbank, in der Sie die Berechtigung zum Erstellen und Befüllen von Tabellen haben.

Bereiten wir zuerst die Datenbank vor, sodass sie auch tatsächlich XML-Daten enthält. Sie können dies entweder mit einem für Ihre Datenbank spezifischen Datenbankverwaltungstool oder direkt in MapForce tun. Um dies direkt in MapForce zu tun, befolgen Sie die unten stehende Anweisung:

1. Erstellen Sie ein neues Mapping und klicken Sie auf das Register **DB-Abfrage**.
2. Klicken Sie auf **Schnellverbindung** () und befolgen Sie die Anweisungen des Assistenten, um eine neue Datenbankverbindung herzustellen (siehe auch Beispiele für Datenbankverbindungen 200).
3. Fügen Sie den folgenden Text in den SQL-Editor ein. Mit dieser SQL-Abfrage wird eine Datenbanktabelle namens `ARTICLES` erstellt und mit Daten befüllt.

© 2016-2022 Altova GmbH

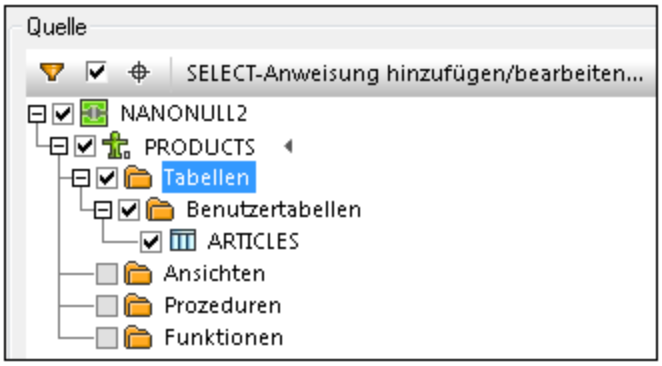
```
-- Create the table
CREATE TABLE
    ARTICLES (
        id INTEGER NOT NULL,
        article XML ) ;
-- Populate the table
INSERT INTO ARTICLES VALUES
   (1, '<Article>
      <Number>1</Number>
      <Name>T-Shirt</Name>
      <SinglePrice>25</SinglePrice>
   </Article>'),
(2, '<Article>
      <Number>2</Number>
      <Name>Socks</Name>
      <SinglePrice>230</SinglePrice>
   </Article>'),
(3, '<Article>
      <Number>3</Number>
      <Name>Pants</Name>
      <SinglePrice>34</SinglePrice>
   </Article>'),
 (4, '<Article>
      <Number>4</Number>
      <Name>Jacket</Name>
      <SinglePrice>5750</SinglePrice>
   </Article>');
```

4. Klicken Sie auf die Schaltfläche **Ausführen** (). Das Ergebnis der Abfrage wird im Abfrageergebnis-Fenster angezeigt. Wenn die Abfrage erfolgreich ausgeführt wurde, werden vier Zeilen zur neu erstellten Tabelle hinzugefügt.

Als nächstes erstellen wir ein Mapping, das auf Basis von Bedingungen XML-Daten aus der oben erstellten Tabelle `ARTICLES` abruft. Es sollen nur Artikel aus der Spalte `ARTICLES` abgerufen werden, deren Preis größer als 100 ist.

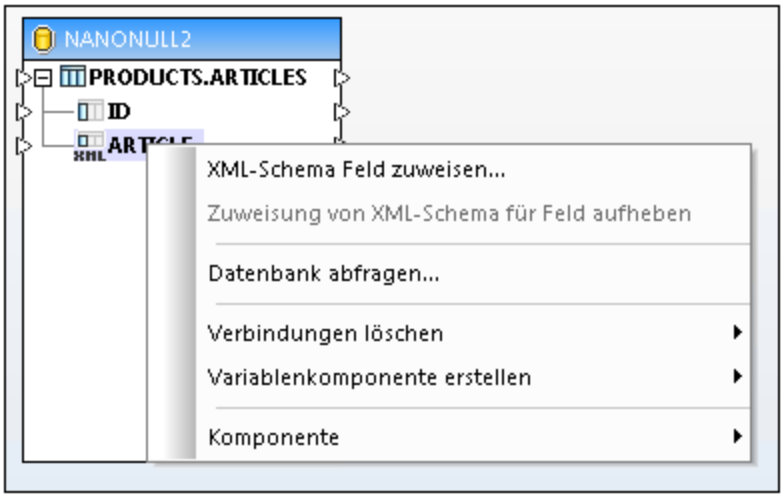
Schritt 1: Hinzufügen der Datenbank

1. Klicken Sie auf das Register **Mapping,** um wieder zurück ins Mapping-Fenster zu wechseln.
2. Klicken Sie im Menü **Einfügen** auf **Datenbank** und befolgen Sie die Anweisungen des Assistenten, um eine Verbindung zur Datenbank herzustellen.
3. Wenn Sie aufgefordert werden, die Datenbankobjekte auszuwählen, wählen Sie die zuvor erstellte Tabelle `ARTICLES` aus.



Schritt 2: Zuweisen des Schemas zum Feld vom Typ XML

1. Klicken Sie mit der rechten Maustaste auf das Datenelement `ARTICLE` der Komponente und wählen Sie im Kontextmenü den Befehl **XML-Schema Feld zuweisen**.

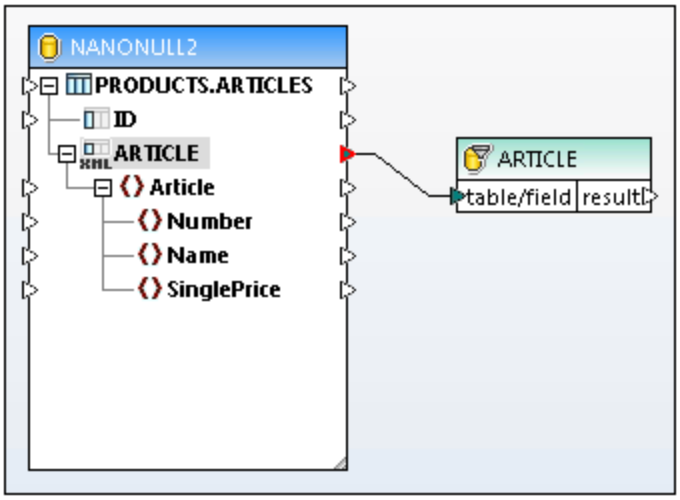


2. Wählen Sie **Datei** und navigieren Sie zum folgenden Schema: **<Dokumente>\Altova\MapForce2023\MapForceExamples\DB2xsd.xsd**.

Schritt 3: Hinzufügen der SQL WHERE/ORDER-Komponente

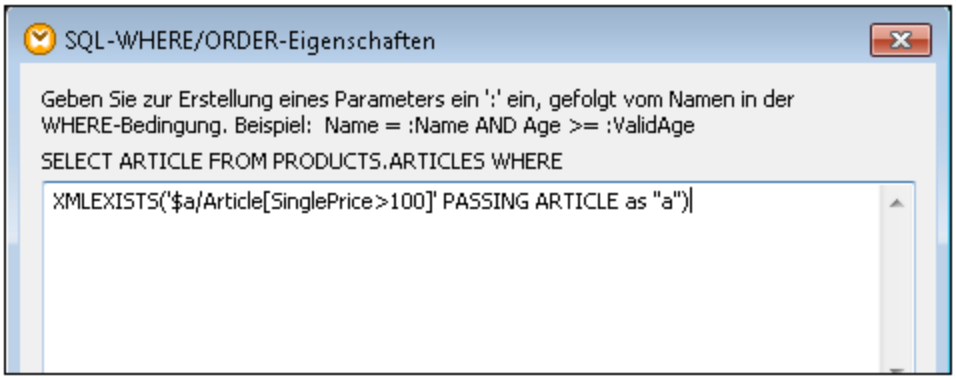
1. Klicken Sie im Menü **Einfügen** auf **SQL WHERE/ORDER**.
2. Verbinden Sie die Spalte `ARTICLE` vom Typ XML mit dem Input von SQL WHERE/ORDER.

© 2016-2022 Altova GmbH



3. Geben Sie im Dialogfeld "Eigenschaften" der SQL-WHERE/ORDER-Komponente den folgenden Text ein:

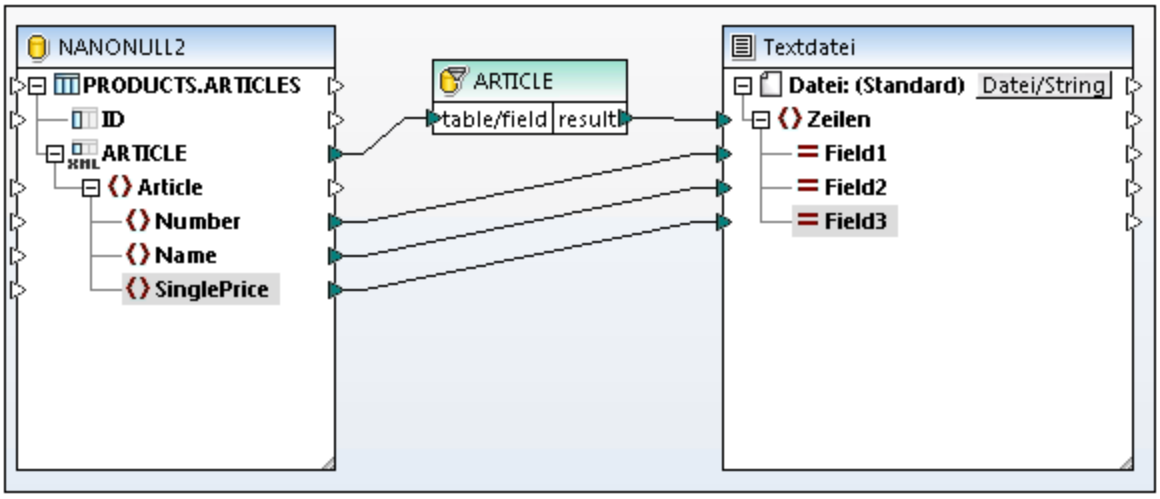
```
XMLEXISTS('$a/Article[SinglePrice>100]' PASSING ARTICLE as "a")
```



Der obige Text bildet den "WHERE"-Teil der SQL-Abfrage. Zur Mapping-Laufzeit wird er mit dem im Dialogfeld angezeigten "SELECT"-Teil kombiniert. In dieser Anweisung werden die Funktion `XMLEXISTS` und die für IBM DB2-Datenbank spezifische Syntax verwendet.

Schritt 4: Hinzufügen der CSV-Zieldatei

1. Klicken Sie im Menü **Einfügen** auf **Textdatei**.
2. Wenn Sie dazu aufgefordert werden, wählen Sie **Einfache Verarbeitung für Standard-CSV... verwenden** und klicken Sie auf **Weiter**.
3. Klicken Sie drei Mal auf **Feld anhängen,** um drei Felder hinzuzufügen, in denen die Nummer, der Name und der Preis des Artikels gespeichert werden sollen. Belassen Sie alle anderen Einstellungen unverändert.
4. Ziehen Sie die Mapping-Verbindungen wie unten gezeigt.



Sie können nun durch Klicken auf das Register **Ausgabe** eine Vorschau auf das Mapping-Ergebnis anzeigen. Wie erwartet, werden nur Artikel mit einem Preis über 100 in der Ausgabe angezeigt.

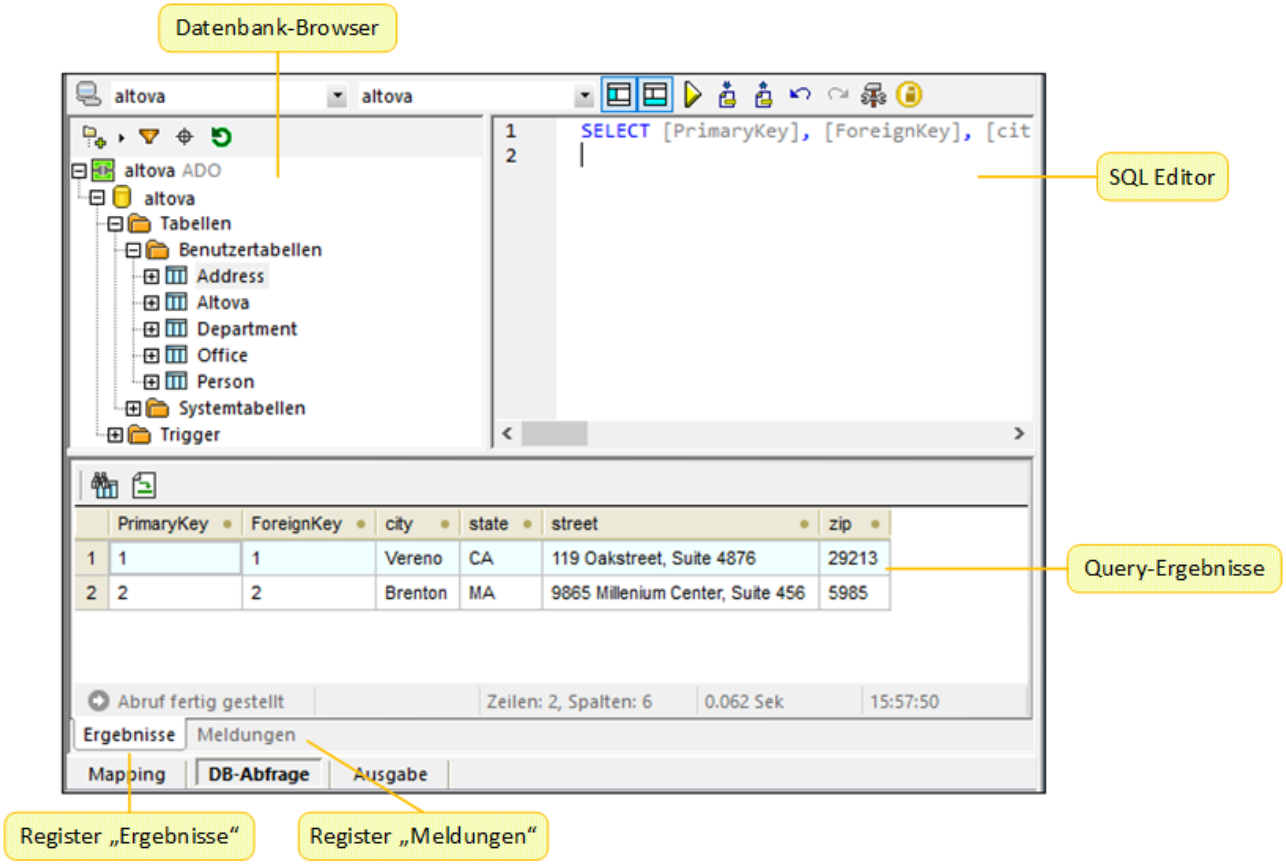
```
1    2,Socks,230
2    4,Jacket,5750
```

4.2.8 Anzeigen und Abfragen von Datenbanken

MapForce hat ein eigenes Fenster für Datenbankabfragen (auch DB-Abfragen genannt), über das Sie - unabhängig vom Mapping-Vorgang - eine Datenbankabfrage durchführen können. Solche direkten Abfragen werden nicht zusammen mit der Mapping-Datei (*.mfd) gespeichert, doch bieten Sie eine praktische Methode, um den Inhalt einer Datenbank direkt von MapForce aus abzufragen bzw. anzuzeigen.

Zu jedem gerade aktiven Mapping gibt es ein separates **DB-Abfrage-Fenster**. Sie können innerhalb von jedem **DB-Abfragefenster** mehrere aktive Verbindungen zu unterschiedlichen Datenbanken herstellen. Beachten Sie, dass die über das **DB-Abfragefenster** erstellen Verbindungen nicht Teil des Mappings bilden und daher beim Schließen des Mappings verloren gehen, es sei denn, Sie definieren diese als globale Ressourcen[1462].

© 2016-2022 Altova GmbH



Das Fenster "Datenbankabfrage" enthält die folgenden Bereiche:

- den **Datenbank-Browser**, in dem die Verbindungsinformationen und die Datenbanktabellen angezeigt werden
- den **SQL Editor,** in dem Sie Ihre SQL-Abfragen verfassen können
- das Register **Ergebnisse**, auf dem die Ergebnisse der Abfrage tabellarisch angezeigt werden
- das Register **Meldungen**, in dem Warn- und Fehlermeldungen angezeigt werden

Der oberste Bereich des Fensters "Datenbankabfrage" enthält die Steuerelemente zum Definieren der Arbeitsdatenbanken. Mit Hilfe dieser Schaltflächen können Sie die Arbeitsdatenbanken sowie die Verbindung und die Datenbankschemas definieren.

Datenbankabfrageeinstellungen können Sie im Abschnitt *Datenbank* des Dialogfelds Optionen konfigurieren. Nähere Informationen dazu finden Sie unter Datenbankabfrageeinstellungen[1547].

4.2.8.1 Auswahl bzw. Herstellung einer Datenbankverbindung

Für jede Datenbank, die Sie abfragen möchten, muss eine Datenbankverbindung hergestellt werden. Wenn Ihr Mapping bereits eine Datenbankkomponente enthält, können Sie die vorhandene Datenbankverbindung aus dem oberen Bereich des **DB-Abfragefensters** auswählen (die Verbindung ist standardmäßig offline) und die Datenbankobjekte anzeigen und Abfragen durchführen.

Wenn Ihr Mapping keine Datenbankkomponente enthält oder wenn Sie eine Verbindung zu einer anderen Datenbank herstellen möchten, klicken Sie auf **Schnellverbindung** ( ) und befolgen Sie die Anweisungen des Assistenten, um eine neue Datenbankverbindung zu erstellen (siehe Beispiele 200). Sie können auch aus den globalen Ressourcen eine bestehende Datenbankverbindung auswählen, wenn eine als globale Ressource definiert wurde (siehe Globale Resourcen 1462).

Nach Herstellung der Datenbankverbindung können Sie mit einer der folgenden Methoden Datenbankabfragen erstellen:

- Importieren der SQL-Abfrage aus einer vorhandenen SQL-Datei in das SQL Editor-Fenster.
- Erstellen der Abfrage im SQL Editor-Fenster.
- Rechtsklick auf ein Objekt im Datenbank-Browser-Fenster und Generieren einer Abfrage (normalerweise SELECT).

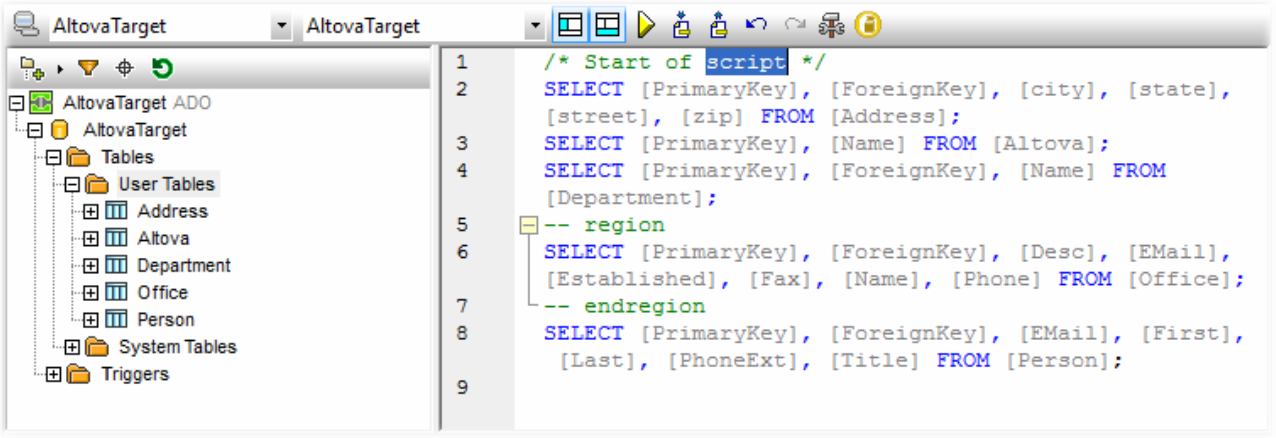
Wenn die im SQL Editor-Fenster angezeigte Abfrage fertig ist, klicken Sie auf die Schaltfläche **Ausführen** . Daraufhin werden die Datenbankdaten abgerufen und auf dem Register "Ergebnisse" in Tabellenform angezeigt.

Beachten Sie, dass in der Statusleiste die Meldung "Abruf fertig gestellt" (Abruf fertig gestellt) sowie andere Informationen zu den Abfrageergebnissen angezeigt werden.

Sobald Sie die Meldung "Abruf fertig gestellt" sehen, können Sie die Ergebnisse durchsuchen, sortieren oder in die Zwischenablage kopieren (siehe Kopieren, Sortieren und Durchsuchen der Abfrageergebnisse 400).

4.2.8.2 Erstellen und Bearbeiten von SQL-Anweisungen

Der **SQL Editor** dient zum Schreiben und Ausführen von SQL-Anweisungen. Darin werden alle SQL-Anweisungen, die Sie automatisch generiert, aus vorhandenen SQL-Skripts geladen oder manuell erstellt haben, angezeigt. Der SQL-Editor unterstützt die Autokomplettierungsfunktion (siehe SQL-Autokomplettierungsvorschläge 286), Regionen und Zeilen- oder Blockkommentare.



Die SQL-Symbolleiste enthält die folgenden Schaltflächen:

	Browser ein/aus: Blendet das Browser-Fenster ein und aus.

 © 2016-2022 Altova GmbH

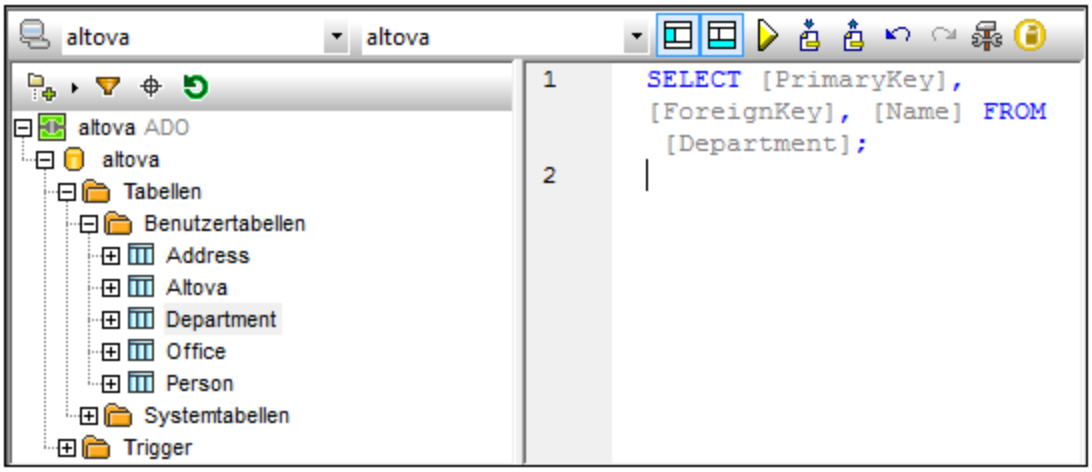
	Ergebnisse ein/aus: Blendet das Ergebnisfenster ein und aus.
	Ausführen (F5): Wenn Sie auf diese Schaltfläche klicken, werden die gerade ausgewählten SQL-Anweisungen ausgeführt. Wenn es mehrere Anweisungen gibt und keine ausgewählt ist, werden alle ausgeführt.
	Rückgängig: Dient zum Rückgängigmachen einer unbegrenzten Anzahl von Bearbeitungen im SQL-Fenster.
	Wiederherstellen: Dient zum Wiederherstellen von zuvor rückgängig gemachten Befehlen. Mit Hilfe dieser beiden Befehle können Sie sich im Verlauf vor und zurückbewegen.
	SQL-Datei importieren: Öffnet eine SQL-Datei im SQL-Editor, die dann ausgeführt werden kann.
	SQL-Datei exportieren: Speichert SQL-Abfragen zur späteren Wiederverwendung
	SQL Script in DatabaseSpy öffnen: Startet DatabaseSpy und öffnet das Script im SQL Editor-Fenster.
	Optionen: Öffnet das Dialogfeld "Optionen", in dem Sie sowohl allgemeine als auch SQL-Editor-Einstellungen definieren können.

4.2.8.2.1 Generieren von SQL-Anweisungen

SQL-Anweisungen können über den Datenbank-Browser automatisch generiert werden, aus Skripts geladen oder manuell erstellt werden.

Um über den Datenbank-Browser SQL-SELECT-Anweisungen zu generieren, wählen Sie eine der folgenden Methoden:

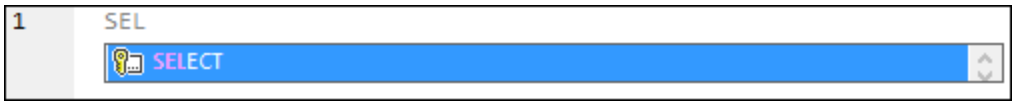
- Klicken Sie im Datenbank-Browser auf ein Datenbankobjekt (z.B. eine Tabelle oder eine Ansicht) oder auf einen Ordner und ziehen Sie es in den SQL Editor.



- Rechtsklicken Sie im Datenbank-Browser auf ein Datenbankobjekt und wählen Sie den Befehl **In neuem SQL Editor anzeigen | SELECT**.

So erstellen Sie SQL-Anweisungen manuell:

1. Beginnen Sie im SQL Editor mit der Eingabe der SQL-Anweisung. Wenn die Autokomplettierungsfunktion aktiviert ist, erscheint während der Eingabe eine Dropdown-Liste mit Eingabevorschlägen.



2. Mit Hilfe der **Nach oben** und **Nach unten**-Pfeiltasten können Sie einen Vorschlag auswählen und den markierten Eintrag dann durch Drücken der Eingabetaste einfügen (siehe auch SQL-Autokomplettierungsvorschläge 286).

4.2.8.2.2 Ausführen von SQL-Anweisungen

Die SQL-Anweisungen, die im SQL Editor angezeigt werden, können an der Datenbank ausgeführt werden, sodass das Ergebnis sofort angezeigt wird. Das Ergebnis der SQL-Abfrage sowie die Anzahl der betroffenen Zeilen werden im Fenster **Meldungen** des **DB-Abfragefensters** angezeigt.

Wenn im SQL-Editor mehrere SQL-Anweisungen angezeigt werden, werden nur die ausgewählten Anweisungen ausgeführt. Sie können einzelnen SQL-Anweisungen folgendermaßen auswählen:

- Halten Sie die linke Maustaste gedrückt und ziehen Sie den Cursor über eine bestimmte Anweisung.
- Klicken Sie im SQL Editor auf eine Zeilennummer.
- Klicken Sie drei Mal auf eine bestimmte Anweisung.

So führen Sie eine SQL-Anweisung aus:

1. Geben Sie die SQL-Anweisung in den SQL Editor ein oder wählen Sie sie aus (siehe Generieren von SQL-Anweisungen 391).
2. Klicken Sie auf die Schaltfläche **Ausführen** ().

4.2.8.2.3 Importieren und Exportieren von SQL-Skripts

Sie können jede SQL-Anweisung, die in einem SQL Editor-Fenster angezeigt wird, in einer Datei speichern und das Skript später wiederverwenden.

So exportieren Sie den Inhalt des SQL Editor-Fensters in eine Datei:

1. Klicken sie auf die Schaltfläche **SQL-Datei exportieren**  und geben Sie einen Namen für das SQL-Skript ein.

© 2016-2022 Altova GmbH

So importieren Sie eine zuvor gespeicherte SQL-Datei:

1. Klicken Sie auf die Schaltfläche **SQL-Datei importieren** und wählen Sie die gewünschte SQL-Datei aus.

4.2.8.2.4 Hinzufügen und Entfernen von SQL-Kommentaren

Sie können im SQL Editor Anweisungen, Teile von Anweisungen oder Gruppen von Anweisungen auskommentieren. Diese Anweisungen bzw. die entsprechenden Teile davon werden bei der Ausführung des SQL Scripts übersprungen.

So kommentieren Sie einen Textabschnitt aus:

1. Wählen Sie eine Anweisung oder den Teil einer Anweisung aus.
2. Rechtsklicken Sie auf die ausgewählte Anweisung und wählen Sie den Befehl **Einfügen / Blockkommentar entfernen**.

```
1
2   /*
3   SELECT [PrimaryKey], [ForeignKey], [city], [state],
    [street], [zip] FROM [Address];
4   SELECT [PrimaryKey], [Name] FROM [Altova];
5   SELECT [PrimaryKey], [ForeignKey], [Name] FROM
    [Department];*/
6
7   SELECT [PrimaryKey], [ForeignKey], [Desc], [EMail],
    [Established], [Fax], [Name], [Phone] FROM [Office];
8   SELECT [PrimaryKey], [ForeignKey], [EMail], [First],
    [Last], [PhoneExt], [Title] FROM [Person];
9
```

So kommentieren Sie Text Zeile für Zeile aus:

- Rechtsklicken Sie auf die gewünschte Stelle, an der Sie den Text auskommentieren möchten und wählen Sie den Befehl **Einfügen / Zeilenkommentar entfernen**. Die Anweisung wird ab der aktuellen Cursorposition bis zum Ende der Anweisung auskommentiert.

So entfernen Sie einen Block- oder Zeilenkommentar:

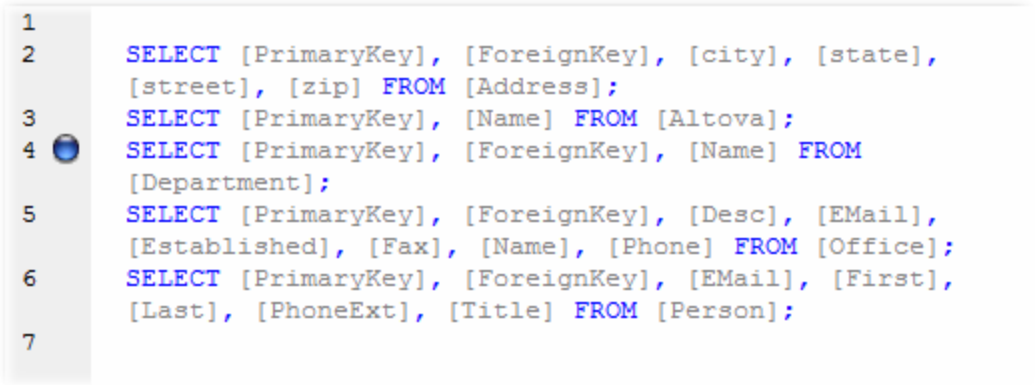
1. Wählen Sie den auskommentierten Teil der Anweisung aus.
   Um einen Zeilenkommentar zu entfernen, genügt es, nur die Kommentarmarkierungen -- vor dem Kommentar auszuwählen.
2. Klicken Sie auf die rechte Maustaste und wählen Sie den Befehl **Einfügen / Block- (oder Zeilen) Kommentar entfernen.**

4.2.8.2.5 Verwenden von Lesezeichen

Lesezeichen dienen zum Markieren von Elementen in langen Scripts.

So fügen Sie ein Lesezeichen hinzu:

- Rechtsklicken Sie auf die Zeile, die Sie mit einem Lesezeichen versehen wollen, und wählen Sie im Kontextmenü den Befehl **Lesezeichen einfügen/entfernen.**



Daraufhin wird am Rand am Anfang der markierten Zeile ein Lesezeichensymbol angezeigt.

So entfernen Sie ein Lesezeichen:

- Rechtsklicken Sie auf die Zeile, aus der Sie das Lesezeichen entfernen möchten und wählen Sie im Kontextmenü den Befehl **Lesezeichen einfügen/entfernen.**

So navigieren Sie zwischen Lesezeichen:

- Um mit dem Cursor zum nächsten Lesezeichen zu gehen, rechtsklicken Sie und wählen Sie den Befehl **Zum nächsten Lesezeichen**.
- Um den Cursor zum vorhergehenden Lesezeichen zu verschieben, rechtsklicken Sie und wählen Sie den Befehl **Zum vorhergehenden Lesezeichen**.

So entfernen Sie alle Lesezeichen:

- Rechtsklicken Sie und wählen Sie den Befehl **Alle Lesezeichen entfernen**.

© 2016-2022 Altova GmbH

4.2.8.2.6 Einfügen von Regionen

Regionen sind Textabschnitte, die zur Strukturierung Ihrer SQL Scripts markiert und als Einheit definiert werden. Regionen können reduziert und erweitert werden, um Teile von SQL Scripts ein- und auszublenden. Regionen können auch innerhalb anderer Regionen verschachtelt werden.

Wenn Sie eine Region einfügen, werden ein Erweitern/Reduzieren Symbol und ein `--region` Kommentar oberhalb des ausgewählten Texts eingefügt.

Anmerkung: Sie können den Namen einer Region ändern, indem Sie an den --region Kommentar einen beschreibenden Text anhängen. Das Wort "region" darf nicht gelöscht werden, z.B. --region DB2query.

So erstellen Sie eine Region:

1. Wählen Sie im SQL Editor die Anweisungen aus, die als Region definiert werden sollen.
2. Rechtsklicken Sie und wählen Sie im Kontextmenü den Befehl **Region einfügen**.
   Die ausgewählten Anweisungen werden zu einer Region und können erweitert und reduziert werden.

```
1
2     SELECT [PrimaryKey], [ForeignKey], [city], [state],
      [street], [zip] FROM [Address];
3     -- region
4     SELECT [PrimaryKey], [Name] FROM [Altova];
5     SELECT [PrimaryKey], [ForeignKey], [Name] FROM
      [Department];
6     -- endregion
7     SELECT [PrimaryKey], [ForeignKey], [Desc], [EMail],
      [Established], [Fax], [Name], [Phone] FROM [Office];
8     SELECT [PrimaryKey], [ForeignKey], [EMail], [First],
      [Last], [PhoneExt], [Title] FROM [Person];
9
```

3. Klicken Sie auf das + oder - Feld, um die Region zu erweitern bzw. zu reduzieren.

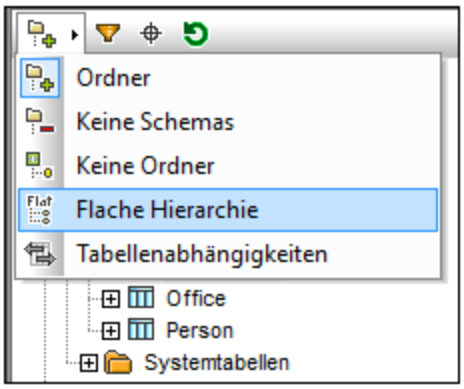
So entfernen Sie eine Region:

- Löschen Sie die Kommentare `-- region` und `-- endregion`.

4.2.8.3 Anzeigen von Datenbankobjekten

Wenn Sie eine Verbindung zu einer oder mehreren Datenbanken hergestellt haben, sehen Sie im **Datenbank-Browser**-Fenster eine vollständige Übersicht über die Objekte in der jeweiligen Datenbank. Die Tabellen, Ansichten, Prozeduren usw. werden bis auf die letzte Ebene hinunter angezeigt. Bei Datenbanken mit XML-Unterstützung werden die XML-Schemas im **Datenbank-Browser** in einem eigenen Ordner angezeigt.

Der **Datenbank-Browser** enthält für die benutzerdefinierte Navigation durch die Datenbankobjekte eine Reihe vordefinierter Layouts für die Datenbank-Anzeige. Diese vordefinierten Layouts können im oberen Bereich des Datenbank-Browsers ausgewählt werden.



Um ein Layout auszuwählen, klicken Sie im Browser auf die Schaltfläche **Layout "Ordner"**  und wählen Sie aus der Dropdown-Liste einen Eintrag aus. Beachten Sie, dass sich die Schaltfläche ändert, je nachdem welches Layout ausgewählt ist.

- Im Layout **Ordner** werden Datenbankobjekte auf Basis des Objekttyps in einer hierarchischen Struktur in Ordner unterteilt. Dies ist die Standardeinstellung.
- Das Layout **Keine Schemas** ähnelt dem Ordner-Layout, mit der Ausnahme, dass es keine Datenbankschema-Ordner gibt; Tabellen werden daher nicht nach Datenbankschemas gegliedert.
- Im Layout **Keine Ordner** werden Datenbankobjekte in einer Hierarchie ohne Verwendung von Ordnern angezeigt.
- Im Layout **Flach** werden Datenbankobjekte nach ihrem Typ auf der ersten hierarchischen Ebene geordnet. So werden Spalten z.B. anstatt in der entsprechenden Tabelle in einem separaten Ordner für Spalten angezeigt.
- Im Layout **Tabellenabhängigkeiten** werden Tabellen nach ihren Beziehungen zu anderen Tabellen gegliedert. Es gibt Kategorien für Tabellen mit Sekundärschlüsseln, durch Sekundärschlüssel referenzierte Tabellen und Tabellen, die in keiner Beziehung zu anderen Tabellen stehen.

Neben der Layout-Navigation können Sie im **Datenbank-Browser** die folgenden Aufgaben durchführen:

- Generieren von SQL-Anweisungen (siehe Generieren von SQL-Anweisungen 391)
- Filtern und Durchsuchen der angezeigten Datenbankobjekte (siehe Filtern und Suchen von Datenbankobjekten 397)
- Sortieren der Tabellen nach System- und Benutzertabellen
- Aktualisieren des Root-Objekts der aktiven Datenquelle

So sortieren Sie Tabellen nach Benutzer- und Systemtabellen:

1. Rechtsklicken Sie im Browser auf den Ordner "Tabellen" und wählen Sie den Befehl **Nach Benutzer- und Systemtabellen sortieren**.

   Anmerkung: Diese Funktion steht nur zur Verfügung, wenn eines der folgenden Layouts ausgewählt ist: **Ordner, Keine Schemas** oder **Flach**.

© 2016-2022 Altova GmbH

So aktualisieren Sie das Root-Objekt der aktiven Datenquelle:

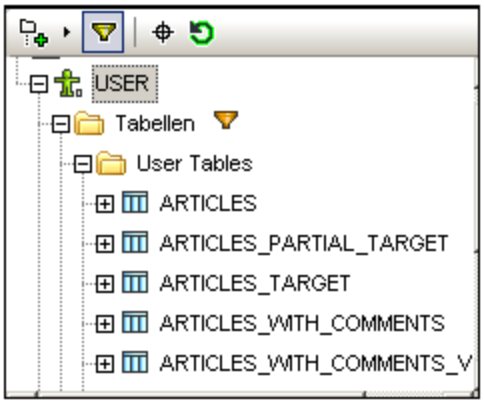
- Klicken Sie am oberen Rand des **Datenbank-Browsers** auf die Schaltfläche **Aktualisieren** ( ).

4.2.8.3.1 Filtern und Suchen von Datenbankobjekten

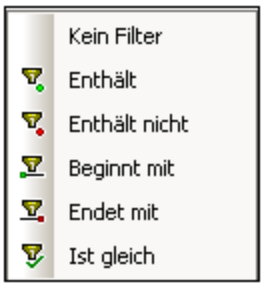
Im **Datenbank-Browser** können Sie alle Datenbankobjekte (Schemas, Tabellen, Ansichten, usw.) nach dem Namen oder einem Teil des Namens filtern. Die Objekte werden während der Eingabe gefiltert. Standardmäßig wird die Groß- und Kleinschreibung bei der Filterung nicht berücksichtigt. Die Filterung wird nicht unterstützt, wenn Sie das Layout "Keine Ordner" verwenden.

Filtern von Datenbankobjekten

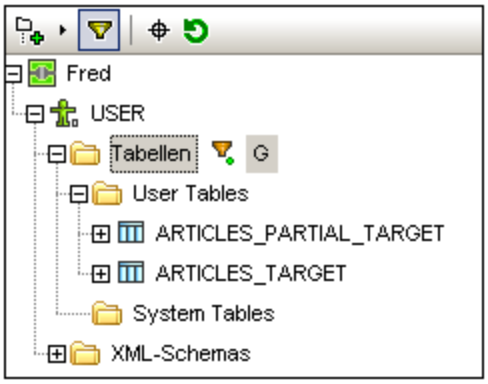
1. Klicken Sie am oberen Rand des Datenbank-Browsers auf **Ordnerinhalt filtern** .
   Neben allen Ordnern in derzeit ausgewählten Layout erscheinen Filtersymbole.



2. Klicken Sie auf das Filtersymbol neben dem gewünschten Ordner und wählen Sie im Kontextmenü die Filteroptionen (z.B. **Enthält**).



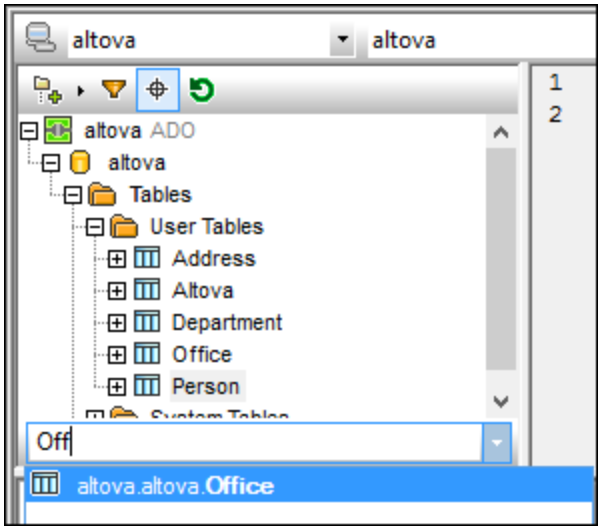
3. Geben Sie in das leere Feld neben dem Filtersymbol den Suchtext ein, z.B "G". Das Ergebnis wird während der Eingabe angepasst.



Suchen von Datenbankobjekten

Um ein bestimmtes Datenbankobjekt nach seinem Namen zu suchen, verwenden Sie entweder Filterfunktionen oder die **Objektsuche**. So suchen Sie Datenbankelemente über die Objektsuche:

1. Klicken Sie im **Datenbank-Browser** auf die Schaltfläche **Objektsuche** .
2. Geben Sie in der daraufhin angezeigten Dropdown-Liste den gewünschten Suchstring ein, z.B. "Off".



3. Klicken Sie in der Liste auf das Objekt, um es im **Datenbank-Browser** anzuzeigen.

4.2.8.3.2 Kontextoptionen im Datenbank-Browser

Welche Kontextmenüoptionen im Datenbank-Browser zur Verfügung stehen, hängt vom ausgewählten Objekt ab, z.B.:

- bei Rechtsklick auf das "Root"-Objekt können Sie die Datenbank **aktualisieren**.
- bei Rechtsklick auf einen Ordner stehen immer die selben Auswahlmöglichkeiten zur Verfügung: **Erweitern** | **Gleichrangige | Untergeordnete** und **Reduzieren** | **Gleichrangige | Untergeordnete**
- bei Rechtsklick auf jede Art von Datenbankobjekt wird der Befehl **In neuem SQL Editor anzeigen** sowie die weiter unten beschriebenen Einträge des Untermenüs angezeigt

© 2016-2022 Altova GmbH

Um mehrere Datenbankobjekte auszuwählen, drücken Sie entweder **Umschalt + Klick** oder **Strg + Klick**.

Anmerkung: Die Syntax der Anweisungen kann variieren, je nachdem, welche Datenbank Sie verwenden. In den Beschreibungen unten wird Microsoft SQL Server 2014 als Beispiel verwendet.

Die folgenden Optionen stehen im **In neuem SQL Editor anzeigen**-Kontextmenü für das Root-Objekt zur Verfügung:

- **CREATE**: Erstellt eine CREATE-Anweisung für das ausgewählte Datenbank-Root-Objekt, z.B:
  ```
  CREATE DATABASE [MYDB]
  ```
- **DROP**: Erstellt eine DROP-Anweisung für das ausgewählte Datenbank-Root-Objekt, z.B:
  ```
  DROP DATABASE [MYDB]
  ```

Die folgenden Optionen stehen im **In neuem SQL Editor anzeigen**-Kontextmenü für Tabellen und Ansichten zur Verfügung:

- **SELECT**: Erstellt eine SELECT-Anweisung, mit der Daten aus allen Spalten der Quelltabelle abgerufen werden, z.B.
  ```
  SELECT [DepartmentID], [Name], [GroupName], [ModifiedDate] FROM [MYDB].
  [HumanResources].[Department]
  ```
- **Name**: Gibt den Namen der Tabelle zurück.
- **Pfad**: Gibt den vollständigen Pfad zu der/den Tabellen im folgenden Format zurück: `Datenquellenname.Datenbankname.Schemaname.Tabellenname`.

Wenn Sie mehrere Tabellen ausgewählt haben, werden die Namen oder Pfade auf durch Kommas getrennten separaten Zeilen ausgegeben.

Die folgenden Optionen stehen im **In neuem SQL Editor anzeigen**-Kontextmenü für Spalten zur Verfügung:

- **SELECT**: Erstellt eine SELECT-Anweisung, mit der Daten aus der/den ausgewählten Spalte(n) der übergeordneten Tabelle abgerufen werden, z. B.:
  ```
  SELECT [DepartmentID] FROM [MYDB].[HumanResources].[Department]
  ```
- **Name**: Gibt den Namen der ausgewählten Spalte zurück.
- **Pfad**: Gibt den vollständigen Pfad zur Spalte im folgenden Format zurück: `Datenquellenname.Datenbankname.Schemaname.Tabellenname.Spaltenname`.

Wenn Sie mehrere Spalten ausgewählt haben, werden die Namen oder Pfade auf durch Kommas getrennten separaten Zeilen ausgegeben.

Die folgenden Optionen stehen im **In neuem SQL Editor anzeigen**-Kontextmenü für **Constraints** zur Verfügung:

- **Name**: Gibt den Namen des ausgewählten Constraint zurück.
- **Pfad**: Gibt den vollständigen Pfad zum Constraint im folgenden Format zurück: `Datenquellenname.Datenbankname.Schemaname.Tabellenname.Constraintname`.

Wenn Sie mehrere Constraints ausgewählt haben, werden die Namen oder Pfade auf durch Kommas getrennten separaten Zeilen ausgegeben.

Die folgenden Optionen stehen im **In neuem SQL Editor anzeigen**-Kontextmenü für **Indizes** zur Verfügung:

- **Name**: Gibt den Namen des ausgewählten Indexes zurück.

- **Pfad**: Gibt den vollständigen Pfad zum Index im folgenden Format zurück: `Datenquellenname.Datenbankname.Schemaname.Tabellenname.Indexname.`

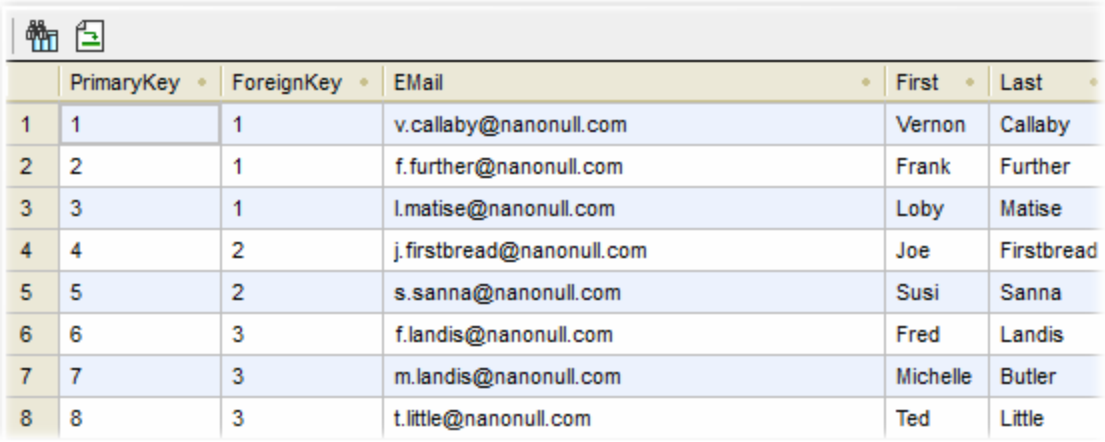
Wenn Sie mehrere Indizes ausgewählt haben, werden die Namen oder Pfade auf durch Kommas getrennten separaten Zeilen ausgegeben.

Wenn die Datenbank Unterstützung für XML-Schemas bietet, stehen zu jedem im Ordner "XML-Schemas" angezeigten Schema die folgenden Optionen zur Verfügung:

- **In XMLSpy anzeigen**: Öffnet das Datenbankschema in XMLSpy, vorausgesetzt dieses Programm ist installiert.
- **XML-Schemas verwalten**: Öffnet ein Dialogfeld, in dem Sie neue XML-Datenbankschemas hinzufügen oder vorhandene löschen können.

4.2.8.4 Kopieren, Sortieren und Durchsuchen der Abfrageergebnisse

Auf dem **Ergebnisregister** des **DB-Abfragefensters** sehen Sie die als Ergebnis der Datenbankabfrage abgerufenen Datensätze.



	PrimaryKey	ForeignKey	EMail	First	Last
1	1	1	v.callaby@nanonull.com	Vernon	Callaby
2	2	1	f.further@nanonull.com	Frank	Further
3	3	1	l.matise@nanonull.com	Loby	Matise
4	4	2	j.firstbread@nanonull.com	Joe	Firstbread
5	5	2	s.sanna@nanonull.com	Susi	Sanna
6	6	3	f.landis@nanonull.com	Fred	Landis
7	7	3	m.landis@nanonull.com	Michelle	Butler
8	8	3	t.little@nanonull.com	Ted	Little

Über die Symbolleisten-Schaltflächen können Sie zwischen den Ergebnissen und den SQL-Anweisungen navigieren und in den Abfrageergebnissen suchen.

	**Suchen:** Sucht in den angezeigten Ergebnissen nach einem bestimmten Text. Drücken Sie **F3,** zu zur nächsten Instanz des gesuchten Begriffs zu gelangen.
	**Gehe zu Anweisung**: Springt zum SQL Editor und markiert die Anweisung, die das aktuelle Ergebnis erzeugt hat. Sie kann besonders dann hilfreich sein, wenn der SQL Editor mehrere Anweisungen enthält.

So wählen Sie Zellen aus den Abfrageergebnissen aus:

- Klicken Sie auf eine Spaltenüberschrift, um die gesamte Spalte auszuwählen
- Klicken Sie auf eine Zeilennummer, um die gesamte Zeile auszuwählen

© 2016-2022 Altova GmbH

- Klicken Sie auf einzelne Zellen. Wenn Sie beim Klicken die **Strg**-Taste gedrückt halten, können Sie mehrere Elemente auswählen. Wenn eine Spalte oder Zelle XML-Daten enthält, können diese Daten ebenfalls kopiert werden.

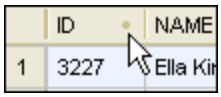
  Anmerkung: Sie können über das Kontextmenü Daten auch auswählen, **Auswahl | Zeile | Spalte | Alle.**

So kopieren Sie die ausgewählten Zellen in die Zwischenablage:

- Klicken Sie auf die rechte Maustaste und wählen Sie im Kontextmenü den Befehl **Ausgewählte Zellen kopieren.**

So sortieren Sie Daten:

- Rechtsklicken Sie auf eine beliebige Stelle in der Spalte, die sortiert werden soll, und wählen Sie **Sortierung | Aufsteigend** oder **Absteigend.**
- Klicken Sie auf das Sortiersymbol in der Spalteüberschrift



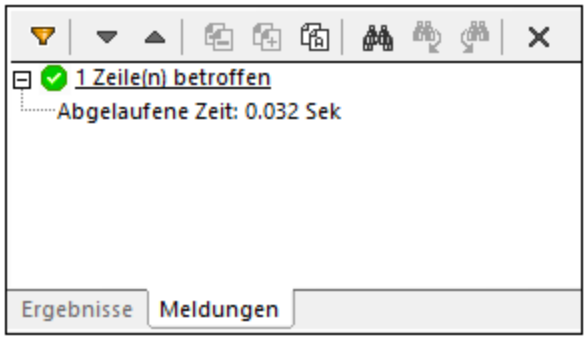
  Die Daten werden nach dem Inhalt der sortierten Spalte sortiert.

So stellen Sie die Standard-Sortierreihenfolge wieder her:

- Rechtsklicken Sie auf eine beliebige Stelle in der Tabelle und wählen Sie im Kontextmenü den Befehl **Sortierung | Standardeinstellung wiederherstellen.**

4.2.8.5 Anzeige des Status von ausgeführten Abfragen

Auf dem Register **Meldungen** des **DB-Abfragefensters** finden Sie Informationen zur letzten ausgeführten SQL-Anweisung und Fehlerberichte oder Warnmeldungen.



Sie können zum Anpassen der Ansicht des Registers **Meldungen** verschiedene Filter verwenden oder das Fenster Zeile für Zeile mit Hilfe der Schaltflächen **Nächste** und **Vorherige** durchsuchen. Mit Hilfe der Schaltflächen am oberen Rand können Sie durch die Meldungen navigieren, Text in die Zwischenablage kopieren und bestimmte Teile der Meldung ausblenden. Diese Optionen stehen auch im Kontextmenü zur Verfügung, wenn Sie auf dem Register **Meldungen** auf die rechte Maustaste klicken.

	**Filter**: Ruft ein Popup-Menü auf, in dem Sie einzelne Meldungstypen herausfiltern können (**Übersicht, Erfolg, Warnung, Fehler, Auto-Einfügung, Fortschritt**). "Auto-Einfügung" bezieht sich auf Meldungen, die ausgelöst werden können, wenn SQL-Anweisungen oder -Konstrukte automatisch im den SQL Editor eingefügt werden. "Fortschrittsmeldungen" geben Auskunft über das Ergebnis der Datenbankverbindung sowie das des SQL-Parsing und des Ladens der Datenstruktur). Sie können alle oder keiner der Optionen mit einem einzigen Mausklick aktivieren bzw. deaktivieren. Wählen Sie dazu einfach aus dem Popup-Menü eine der Optionen **Alle aktivieren** oder **Alle deaktivieren**.
	**Nächste**: Springt zur nächsten Meldung und markiert sie.
	**Vorherige**: Springt zur vorherigen Meldung und markiert sie.
	**Ausgewählte Meldung in die Zwischenablage kopieren**
	**Ausgewählte Meldung einschließlich ihrer Subeinträge in die Zwischenablage kopieren**
	**Alle Meldungen in die Zwischenablage kopieren**
	**Suchen**: Ruft das Dialogfeld **Suchen** auf.
	**Vorheriges suchen**: Springt zur vorhergehenden Instanz des im Dialogfelds **Suchen** definierten String.
	**Weitersuchen**: Springt zur nächsten Instanz des im Dialogfelds **Suchen** definierten String.
	**Löschen**: Entfernt alle Meldungen vom Meldungsregister des SQL Editor-Fensters.

4.2.9 Gespeicherte Prozeduren

Gespeicherte Prozeduren sind Programme, die auf einem Datenbankserver gehostet und ausgeführt werden. Gespeicherte Prozeduren können von Client-Applikationen aufgerufen werden. Oft werden sie in einem erweiterten SQL-Dialekt geschrieben. Einige Datenbanken unterstützen auch Implementierungen in Java, .NET CLP oder anderen Programmiersprachen.

Normalerweise dienen gespeicherte Prozeduren zur Abrage von Datenbanken und der Rückgabe der Daten an den aufrufenden Client. Oder es werden damit, nachdem Input-Parameter zusätzlich validiert wurden, Änderungen an der Datenbank vorgenommen. Mit gespeicherten Prozeduren können aber auch andere Aktionen außerhalb von Datenbanken ausgeführt werden - z.B. können damit E-Mails gesendet werden.

© 2016-2022 Altova GmbH

Eine gespeicherte Prozedur kann null oder mehr Input- und Output-Parameter haben und zusätzlich zum Standardrückgabewert optional null oder mehr Datensatzstrukturen zurückgeben. Folglich können Sie eine gespeicherte Prozedur in MapForce auf verschiedene Arten aufrufen:

- Sie können eine gespeicherte Prozedur aufrufen, um Daten abzurufen, als wären diese eine Quellkomponente im Mapping. Dies gilt für Prozeduren, die keine Input-Parameter haben. Die Prozedur wird bei der Ausführung des Mappings aufgerufen und gibt eine Datensatzstruktur oder Output-Parameter zurück. Sie können die Datensatzstruktur oder die Output-Parameter oder beides auf jeden anderen von MapForce unterstützten Datentyp mappen. Ein Beispiel dazu finden Sie unter Gespeicherte Prozeduren als Datenquelle 408.
- Sie können eine gespeicherte Prozedur als funktionsähnlichen Aufruf mit Parametern verwenden. In diesem Fall stellen Sie alle erforderlichen Input-Parameter über das Mapping bereit und können auch die zurückgegebene Datensatzstruktur oder die Output-Parameter oder beides auf eine andere von MapForce unterstützte Zielkomponente mappen. Ein Beispiel dazu finden Sie unter Gespeicherte Prozeduren mit Input und Output 411.
- Sie können eine gespeicherte Prozedur aufrufen, als würde es sich dabei um eine Zielkomponente im Mapping handeln. Ein typisches Anwendungsbeispiel wäre der Aufruf einer gespeicherten Prozedur mit Parametern, um die Datenbank zu bearbeiten (z.B. um einen Datensatz einzufügen). Diese Methode eignet sich dann, wenn Sie keinen Output aus der gespeicherten Prozedur benötigen. Bei dieser Methode können Sie die gespeicherte Prozedur auch innerhalb einer Datenbanktransaktion, die im Fall eines Fehlers mittels Rollback rückgängig gemacht werden kann, ausführen. Ein Beispiel dazu finden Sie unter Gespeicherte Prozeduren in Zielkomponeten 415.

In manchen Fällen müssen gespeicherte Prozeduren oder Aktionen an Datenbanktabellen in einer bestimmten Reihenfolge (zuerst einfügen, dann aktualisieren, usw.) ausgeführt werden. So muss z.B. eventuell der Output-Parameter einer gespeicherten Prozedur an eine andere gespeicherte Prozedur übergeben werden oder von einer gespeicherten Prozedur zurückgegebene Daten müssen mit Daten aus einer Tabelle kombiniert werden. Solche Aktionen lassen sich mit Hilfe lokaler, in MapForce definierter Beziehungen durchführen, selbst wenn die zugrunde liegende Datenbank Primär-/Sekundärschlüsselbeziehungen zwischen Tabellen nicht erzwingt. Nähere Informationen dazu finden Sie unter Gespeicherte Prozeduren und lokale Beziehungen 418.

Anmerkung: Zur Veranschaulichung der Implementierung gespeicherter Prozeduren in MapForce werden in diesem Abschnitt Microsoft SQL Server 2016 und die "AdventureWorks 2016"-Datenbank verwendet. Zweitere kann von der CodePlex Website (https://github.com/Microsoft/sql-server-samples/releases/tag/adventureworks) heruntergeladen werden.

Anmerkungen zur Unterstützung

- Gespeicherte Prozeduren können nur im BUILT-IN-Ausführungsprozessor verwendet werden. Die Codegenerierung in C++, C# oder Java wird nicht unterstützt.
- Benutzerdefinierte Typen, Cursor-Typen, Variante-Typen und viele "exotische" datenbankspezifische Datentypen (z.B. Arrays, Geometrieparameter, CLR-Typen) werden im Allgemeinen als Input- oder Output-Parametertypen nicht unterstützt.
- Das Überladen von Prozeduren und Funktionen (mehrere Definitionen von Routinen mit demselben Namen und unterschiedlichen Parametern) wird nicht unterstützt.
- Einige Datenbanken unterstützen Standardwerte für Input-Parameter. Diese Funktion wird derzeit nicht unterstützt. Sie können Input-Parameter im Mapping nicht weglassen und stattdessen den Standardwert verwenden.
- Gespeicherte Prozeduren, die mehrere Datensatzstrukturen zurückgeben, werden je nach Treiber- und Datenbank-API-Kombination (ODBC/ADO/ADO.NET/JDBC) unterstützt. Es werden nur Prozeduren unterstützt, die bei einer fixen Spaltenstruktur dieselbe Anzahl an Datensatzstrukturen zurückgeben.

- Verwenden Sie, wo immer dies möglich ist, die neueste Version des nativen vom Datenbankanbieter zur Verfügung gestellten Treibers. Verwenden Sie nach Möglichkeit keine Brückentreiber wie die ODBC zu ADO Bridge oder ODBC zu JDBC Bridge.
- Sie können Datenbanktransaktionen optional für gespeicherte Prozeduren, die als Datenzielkomponente aufgerufen werden, aktivieren, siehe Gespeicherte Prozeduren in Zielkomponeten 415. Für gespeicherte Prozeduren, die als Datenquelle (ohne Input-Parameter) oder solche, die wie eine Funktion aufgerufen werden (sowohl mit Input als auch Output), werden Transaktionen nicht unterstützt.

Die folgenden Tabelle enthält Anmerkungen zur Unterstützung der verschiedenen Datenbanken.

Datenbank	Anmerkungen zur Unterstützung
Access	• Gespeicherte Prozeduren haben in Microsoft Access-Datenbanken nur sehr eingeschränkte Funktionalitäten und werden in MapForce nicht unterstützt.
DB2	• Unterstützt in MapForce: gespeicherte Prozeduren, Skalarfunktionen und Tabellenwertfunktionen. • Die Rückgabewerte aus gespeicherten D2-Prozeduren werden nicht unterstützt, da sie über die in MapForce verwendeten Datenbank-APIs nicht gelesen werden können. • Zeilenwertfunktionen (RETURNS ROW) werden nicht unterstützt. • Es wird empfohlen, zumindest das "IBM_DB2 9.7 Fix Pack 3a" zu installieren, um beim Lesen von Fehlermeldungen/Warnungen nach der Ausführung ein bekanntes Problem mit dem JDBC-Treiber zu vermeiden. Dadurch wird auch ein ADO-Problem behoben, das eine fehlende Ergebniszeile verursacht.
Firebird	• Wird in MapForce unterstützt: gespeicherte Prozeduren, Tabellenwertfunktionen
Informix	• Wird in MapForce unterstützt: gespeicherte Prozeduren, Tabellenwertfunktionen
MariaDB	• Wird in MapForce unterstützt: gespeicherte Prozeduren, Skalarfunktionen
MySQL	• Wird in MapForce unterstützt: gespeicherte Prozeduren, Skalarfunktionen. • MySQL bietet ab Version 5.5 vollständige Unterstützung für gespeicherte Prozeduren und Funktionen. Bei Verwendung einer früheren Version sind die MapForce-Funktionen eingeschränkt.
Oracle	• Unterstützt in MapForce: gespeicherte Prozeduren, Skalarfunktionen und Tabellenwertfunktionen. Dazu gehören auch allein stehende gespeicherte Prozeduren sowie innerhalb eines Oracle-Pakets definierte gespeicherte Prozeduren. • Es wird empfohlen, anstelle des **Microsoft OLE DB Provider für Oracle** einen nativen Oracle-Treiber zu verwenden. • Oracle hat eine eigene Methode, um Ergebnisse über Output-Parameter vom Typ REF CURSOR an den Client zurückzugeben. Dies wird von MapForce für gespeicherte Prozeduren unterstützt, nicht aber für Funktionen. Die Namen und die Anzahl der Datensatzstrukturen sind daher für gespeicherte Oracle-Prozeduren immer festgelegt.

© 2016-2022 Altova GmbH

Datenbank	Anmerkungen zur Unterstützung
PostgreSQL	• Unterstützt in MapForce: Skalarfunktionen, Zeilenwertfunktionen, Tabellenwertfunktionen. • In PostgreSQL beschreibt jeder beliebige in einer Funktion definierte Output-Parameter die Spalten der Ergebnisdatensätze. Diese Funktion wird automatisch von MapForce verwendet - die Datensätze müssen nicht durch die Ausführung oder die manuelle Eingabe von Datensatzstrukturen ermittelt werden. Parameter vom Typ "refcursor" werden nicht unterstützt.
Progress OpenEdge	• In MapForce unterstützt: Gespeicherte Prozeduren.
SQL Server	• Unterstützt in MapForce: gespeicherte Prozeduren, Skalarfunktionen und Tabellenwertfunktionen. • Es wird empfohlen, anstelle von **Microsoft OLE DB Provider for SQL Server** den neuesten **SQL Server Native Client-**Treiber zu verwenden. • Die ADO API bietet nur eingeschränkte Unterstützung für einige in SQL Server 2008 hinzugekommenen Datentypen (`datetime2`, `datetimeoffset`). Wenn es mit den neuen temporären Typen bei der Verwendung von ADO mit dem nativen SQL Server Client zu Datenkürzungen kommt, können Sie das `DataTypeCompatibility=80` setzen oder ODBC verwenden. • SQL Server-Prozeduren haben einen impliziten Rückgabeparameter vom Typ `int null`, der für das Mapping zur Verfügung steht. Wenn in der Prozedur eine RETURN-Anweisung weggelassen wird, ist der Ergebniswert 0.
SQLite	• In SQLite werden gespeicherte Prozeduren nicht verwendet.
Teradata	• In MapForce unterstützt: gespeicherte Prozeduren, Makros • Skalarfunktionen: Aggregatfunktion und Tabellenfunktionen werden nicht unterstützt. • Bekanntes Problem: Output-Parameter werden nach dem Aufruf einer Prozedur vom Teradata ODBC-Treiber nicht befüllt.

4.2.9.1 Hinzufügen von gespeicherten Prozeduren zum Mapping

Im Mapping-Bereich werden gespeicherte Prozeduren als Teil der Datenbankkomponente, zu der sie gehören, angezeigt. Damit gespeicherte Prozeduren in der Datenbankkomponente sichtbar sind, müssen Sie diese explizit auswählen, wenn Sie die Datenbankkomponente, wie unten gezeigt, zum Mapping hinzufügen. In diesem Beispiel stellen wir eine Verbindung zu der auf SQL Server ausgeführten Datenbank "AdventureWorks" her. Die Vorgangsweise ist auch bei anderen Datenbanktypen ähnlich.

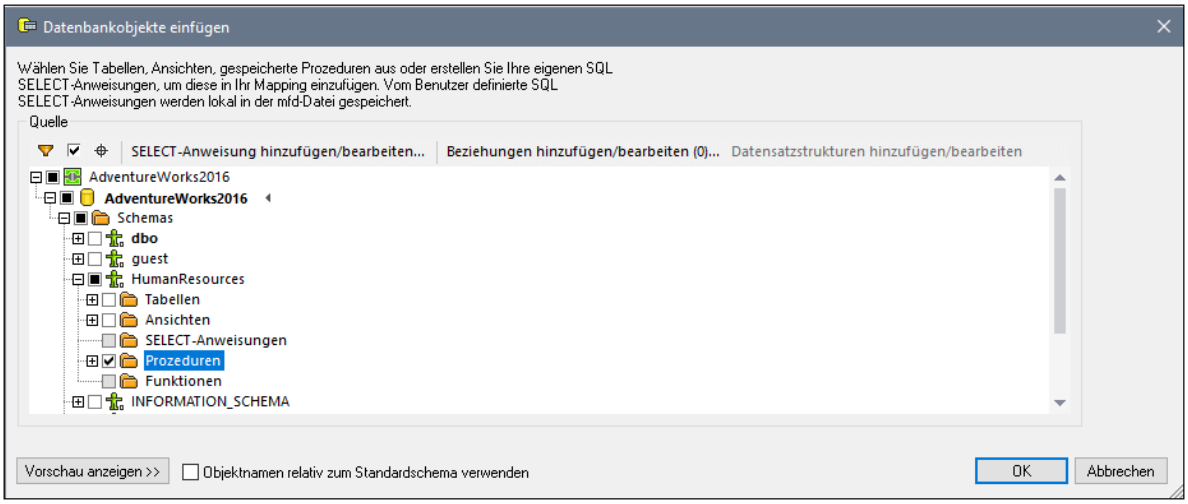
Im Fall von Oracle-Datenbanken kann es sich um eigenständige gespeicherte Prozeduren bzw. Funktionen oder um solche, die Teil eines Oracle-Pakets bilden, handeln. Beide Kategorien können zum Mapping hinzugefügt werden. Gespeicherte Prozeduren oder Funktionen, die zu einem Paket gehören, werden, wie unten gezeigt, im Dialogfeld "Datenbankobjekte einfügen" unter dem Namen des jeweiligen Pakets angezeigt.

So fügen Sie gespeicherte Prozeduren zum Mapping hinzu:

1. Wählen Sie eine der folgenden Methoden:

   - Klicken Sie im Menü **Einfügen** auf **Datenbank**.
   - Klicken Sie auf die Symbolleisten-Schaltfläche **Datenbank einfügen** ( ) .

2. Befolgen Sie die Anweisungen des Datenbankassistenten, bis Sie zum Dialogfeld "Datenbankobjekte einfügen" gelangen. Eine genaue Anleitung zu den einzelnen Datenbanktypen finden Sie unter Beispiele für Datenbankverbindungen 200.



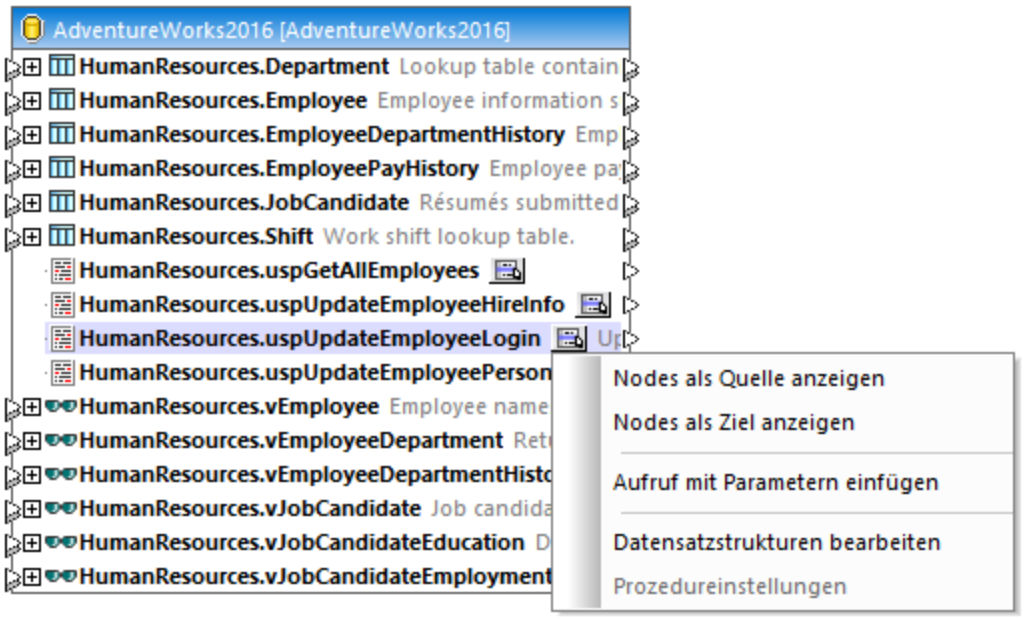
3. Aktivieren Sie die Kontrollkästchen neben den Datenbankobjekten, die im Mapping angezeigt werden sollen, und klicken Sie auf **OK**. Wir haben in diesem Beispiel alle Tabellen, Ansichten und gespeicherten Prozeduren im Schema "HumanResources" ausgewählt.

Anmerkungen

- Sie können die ausgewählten Objekte später jederzeit ändern, indem Sie mit der rechten Maustaste auf die Titelleiste einer Datenbankkomponente klicken und im Kontextmenü den Befehl **Datenbankobjekte hinzufügen/entfernen/bearbeiten** auswählen.
- Ihr Datenbankbenutzerkonto muss die entsprechenden Rechte haben, um gespeicherte Prozeduren in der Datenbank anzeigen und ausführen zu können.

Die Datenbankkomponente wird nun zum Mapping hinzugefügt. Beachten Sie, dass gespeicherte Prozeduren durch das Symbol gekennzeichnet sind. Außerdem werden Tabellen, Ansichten und Prozeduren in der Datenbankkomponente alphabetisch sortiert.

© 2016-2022 Altova GmbH



Über die Schaltfläche **Kontextmenü anzeigen** neben den einzelnen gespeicherten Prozeduren können Sie die Art des Aufrufs der gespeicherten Prozedur sowie andere Einstellungen im Zusammenhang mit der Prozedur konfigurieren:

Option	Verwendung
Nodes als Quelle anzeigen	Wählen Sie diese Option, wenn Sie eine gespeicherte Prozedur *ohne Parameter* aufrufen möchten, um Daten aus einer Datenbank abzurufen und auf eine andere von MapForce unterstützte Komponente (XML, Text, EDI, usw.) zu mappen. Ein Beispiel dazu finden Sie unter Gespeicherte Prozeduren als Datenquelle[408].
Nodes als Ziel anzeigen	Wählen Sie diese Option, wenn Sie eine gespeicherte Prozedur aufrufen möchten, um die Datenbank zu bearbeiten oder eine andere spezielle Aktion, bei der Sie die Ausgabe der gespeicherten Prozedur nicht benötigen, auszuführen. Ein Beispiel dazu finden Sie unter Gespeicherte Prozeduren in Zielkomponenten[415].
Aufruf mit Parametern einfügen	Wählen Sie diese Option, wenn Sie eine gespeicherte Prozedur *mit Parametern* aufrufen und die zurückgegebenen Daten auf eine andere von MapForce unterstützte Komponente mappen möchten. Ein Beispiel dazu finden Sie unter Gespeicherte Prozeduren mit Input und Output[411].
Datensatzstrukturen bearbeiten	Anwendbar auf gespeicherte Prozeduren, die Datensatzstrukturen zurückgeben. Wählen Sie diese Option, um die gespeicherte Prozedur einmal auszuführen, damit MapForce die Struktur der zurückgegebenen Datensatzgruppe ermitteln und im Mapping anzeigen kann. Wenn die gespeicherte Prozedur nicht zum Design-Zeitpunkt ausgeführt werden soll, können Sie die Datensatzstruktur alternativ dazu manuell definieren.

Prozedureinstellungen	Nur anwendbar auf gespeicherte Prozeduren, die als "Ziel" (d.h. diejenigen, die die Datenbank aktualisieren) konfiguriert wurden. Wählen Sie diese Option, um zusätzliche Einstellungen im Zusammenhang mit Prozeduren, wie z.B. die Ausführung einer benutzerdefinierten SQL-Abfrage vor Aufruf der Prozedur oder die Ausführung von Datenbanktransaktionen, zu konfigurieren.

4.2.9.2 Gespeicherte Prozeduren als Datenquelle

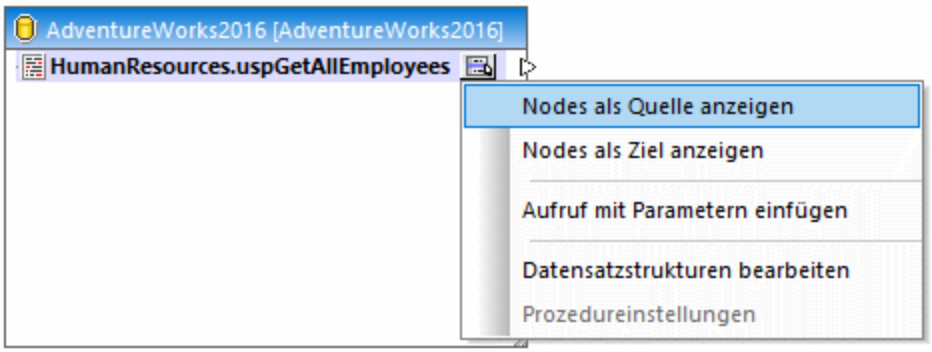
In diesem Beispiel wird gezeigt, wie Sie eine Prozedur, die keine Input-Parameter hat und nur Daten aus der Datenbank abruft, aufrufen. Die gespeicherte Prozedur fungiert in diesem Szenario als Quellkomponente für das Mapping. Sie können damit abgerufene Daten auf jede andere von MapForce unterstützte Zielkomponente mappen. Wenn Sie eine gespeicherte Prozedur mit Input-Parametern aufrufen müssen, lesen Sie nach unter Gespeicherte Prozeduren mit Input und Output 411.

Erstellen wir zuerst die gespeicherte Demo-Prozedur in der Datenbank "AdventureWorks". Führen Sie zu diesem Zweck das unten stehende Skript an der Datenbank aus. Sie können dies entweder von einem Abfragefenster von **Microsoft SQL Server Management Studio** aus oder direkt auf dem Register **DB-Abfrage** von MapForce (siehe Anzeigen und Abfragen von Datenbanken 388) tun. Vergewissern Sie sich in jedem Fall, dass Ihr Datenbankbenutzerkonto Rechte zum Erstellen von gespeicherten Prozeduren hat.

```
CREATE PROCEDURE HumanResources.uspGetAllEmployees
AS
    SELECT LastName, FirstName, JobTitle, Department
    FROM HumanResources.vEmployeeDepartment
```

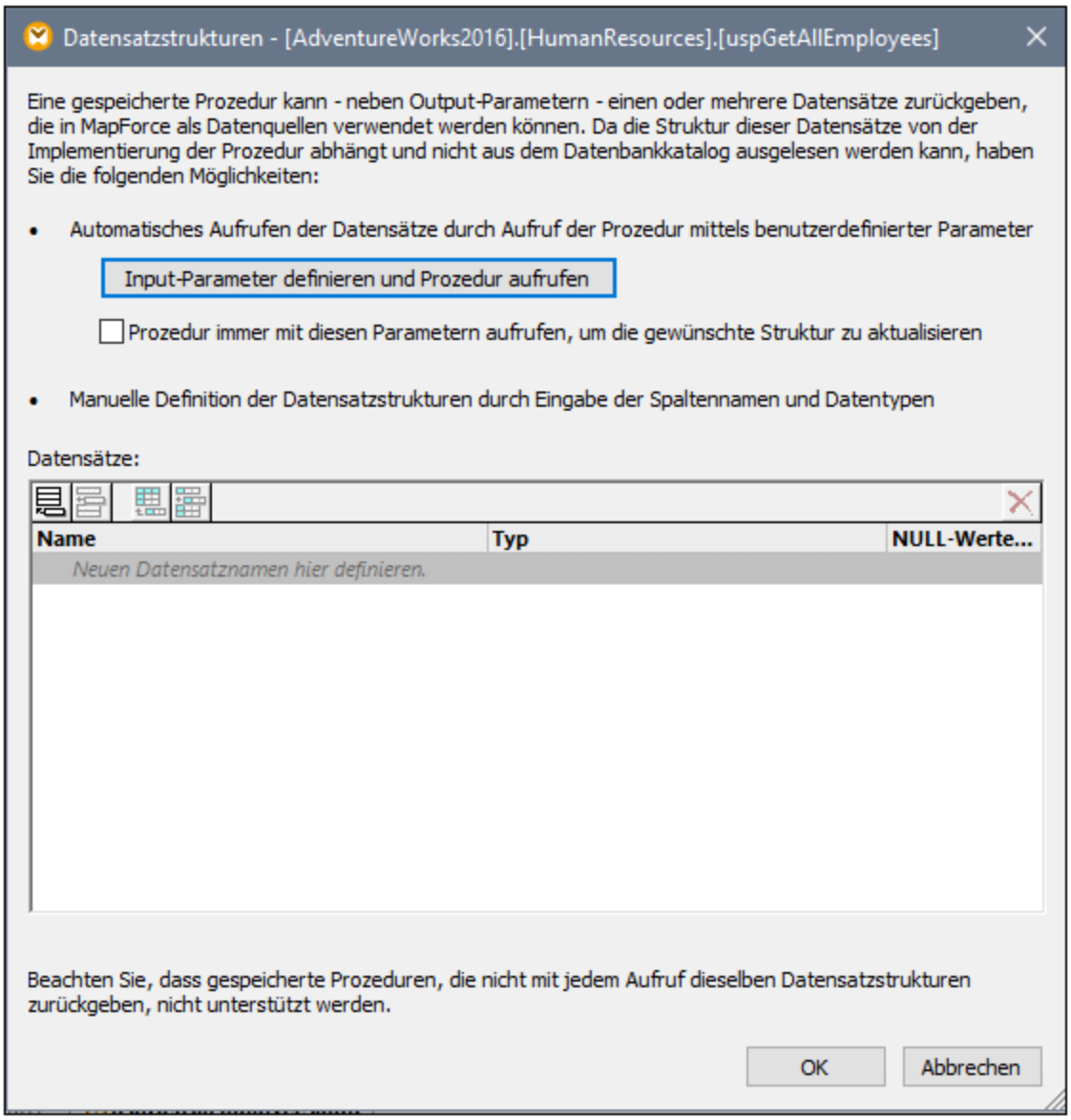
Mit der obigen gespeicherten Prozedur werden Mitarbeiterdaten aus der Ansicht **vEmployeeDepartment** zurückgegeben. In der folgenden Anleitung wird beschrieben, wie Sie ein Mapping erstellen, das die von dieser Prozedur zurückgegebenen Daten verarbeitet.

1. Stellen Sie in MapForce eine Verbindung zur Datenbank "AdventureWorks" her und fügen Sie die gespeicherte Prozedur zum Mapping hinzu, wie unter Hinzufügen von gespeicherten Prozeduren zum Mapping 405 beschrieben. Vergewissern Sie sich, dass Ihr Datenbankbenutzerkonto Rechte zum Anzeigen und Ausführen von gespeicherten Prozeduren hat.
2. Klicken Sie neben der gespeicherten Prozedur auf die Schaltfläche **Kontextmenü anzeigen** und wählen Sie den Befehl **Nodes als Quelle anzeigen**.



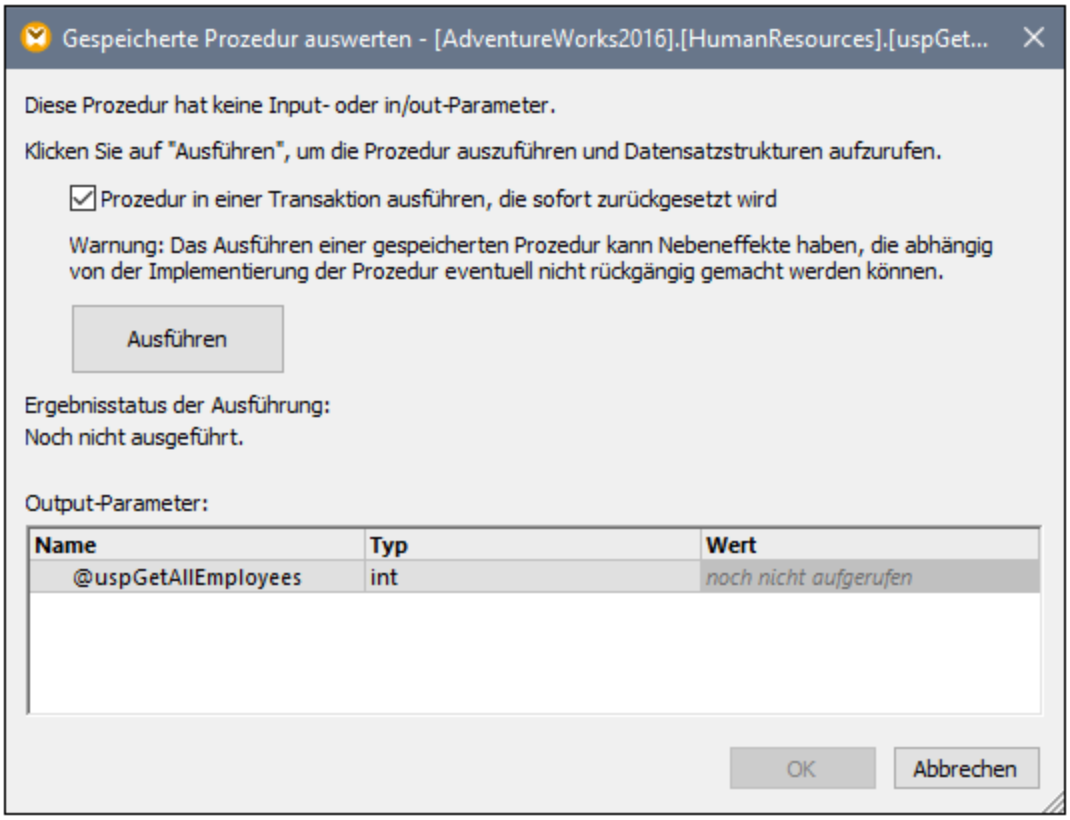
 © 2016-2022 Altova GmbH

3. Klicken Sie nochmals auf die Schaltfläche **Kontextmenü anzeigen** und wählen Sie **Datensatzstrukturen bearbeiten**. Daraufhin wird das Dialogfeld "Datensatzstrukturen" angezeigt.

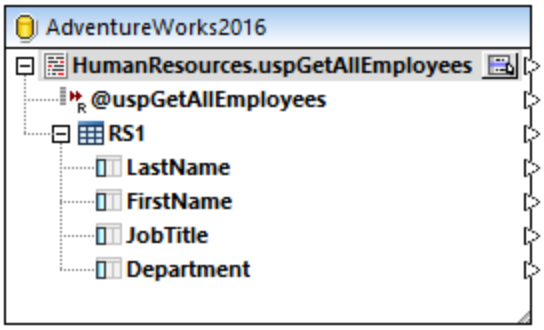


Wenn eine gespeicherte Prozedur während des Designs aufgerufen wird, kann dies (je nach Implementierung der Prozedur) Nebenwirkungen haben. Wenn die gespeicherte Prozedur zum Zeitpunkt des Designs nicht ausgeführt werden soll, klicken Sie nicht, wie in den folgenden Schritten beschrieben, auf **Ausführen**, sondern definieren Sie die erwartete Datensatzstruktur im Dialogfeld "Datensatzstrukturen", indem Sie Datensatzstrukturen und deren dazugehörige Spalten manuell hinzufügen. Verwenden Sie dazu im Dialogfeld "Datensatzstrukturen" die Schaltflächen **Datensatz hinzufügen** oder **Spalte hinzufügen**.

4. Klicken Sie auf **Input-Parameter definieren und Prozedur aufrufen**. Daraufhin wird das Dialogfeld "Gespeicherte Prozedur auswerten" angezeigt.

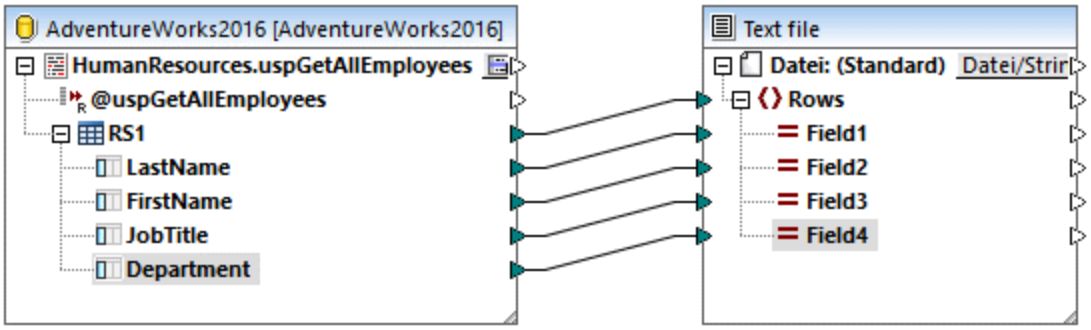


5. Klicken Sie auf **Ausführen** und anschließend auf **OK**. Die Datensatzstruktur ("RS1") ist nun sowohl im Dialogfeld "Datensatzstrukturen" als auch im Mapping sichtbar.

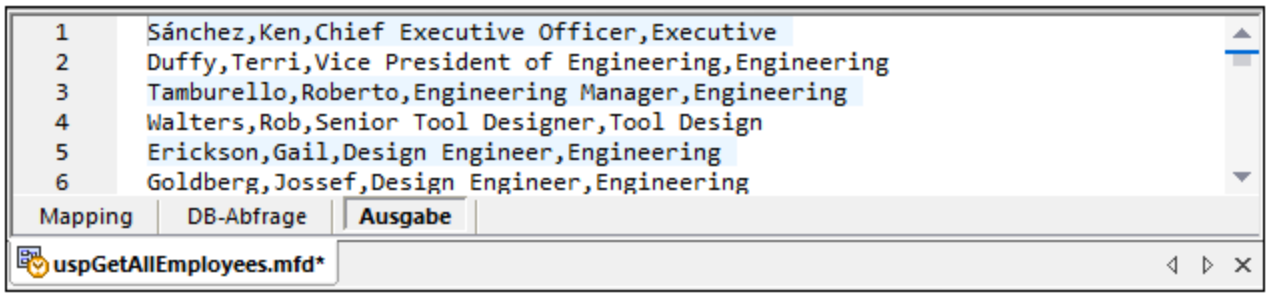


6. Sie können zu diesem Zeitpunkt eine Zielkomponente, in die die abgerufenen Daten geschrieben werden sollen, hinzufügen. In diesem Beispiel werden die Daten in eine CSV-Datei geschrieben. Klicken Sie im Menü **Einfügen** auf **Textdatei** und fügen Sie eine CSV-Komponente zum Mapping hinzu. Nähere Informationen dazu finden Sie unter CSV und Textdateien 585.

 © 2016-2022 Altova GmbH



Sie können nun eine Vorschau auf das Mapping anzeigen. Klicken Sie auf die Schaltfläche **Ausgabe,** um das Ergebnis des Mappings im Fenster **Ausgabe** zu sehen, z.B.:



4.2.9.3 Gespeicherte Prozeduren mit Input und Output

In diesem Beispiel wird gezeigt, wie Sie eine Prozedur, die Input-Parameter hat und auch eine Ausgabe aus der Datenbank abruft, aufrufen. Die gespeicherte Prozedur wird in diesem Szenario ähnlich wie ein Webservice oder eine Funktion aufgerufen. Sie können damit abgerufene Daten auf jede andere von MapForce unterstützte Zielkomponente mappen.

Erstellen wir zuerst die gespeicherte Demo-Prozedur in der Datenbank "AdventureWorks". Führen Sie zu diesem Zweck das unten stehende Skript an der Datenbank aus. Sie können dies entweder von einem Abfragefenster von **Microsoft SQL Server Management Studio** aus oder direkt auf dem Register **DB-Abfrage** von MapForce (siehe Anzeigen und Abfragen von Datenbanken 388) tun. Vergewissern Sie sich in jedem Fall, dass Ihr Datenbankbenutzerkonto Rechte zum Erstellen von gespeicherten Prozeduren hat.

```
CREATE PROCEDURE Production.uspSearchProducts
   @SearchString nvarchar(50)
   ,@MaxPrice money
   ,@ComparePrice money OUTPUT
AS
BEGIN
   SET NOCOUNT ON
   SELECT pr.[Name], pr.ListPrice FROM [Production].[Product] pr
   WHERE pr.[Name] like @SearchString AND  pr.ListPrice < @MaxPrice

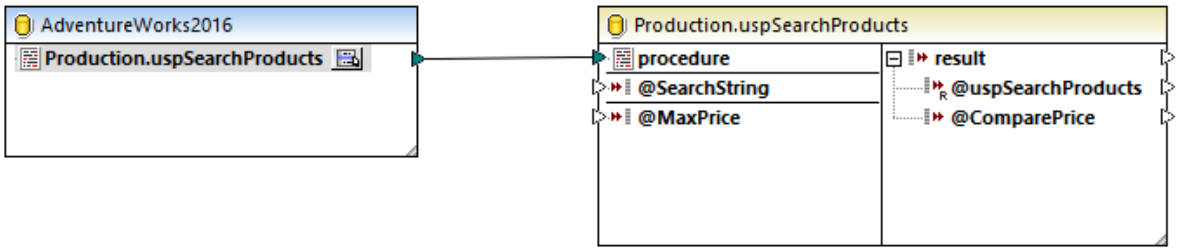
   SET @ComparePrice = @MaxPrice
```

```
   RETURN @ComparePrice
END
```

Mit der obigen gespeicherten Prozedur wird eine Datensatzstruktur mit Produktinformationen abgerufen. Die Prozedur hat als Input zwei Parameter: einen String mit dem Produktnamen (`@SearchString`) und den maximalen Produktpreis (`@MaxPrice`). Zusätzlich zur Datensatzstruktur und dem Standardrückgabeparameter, ruft sie auch einen Output-Parameter (`@ComparePrice`) ab.

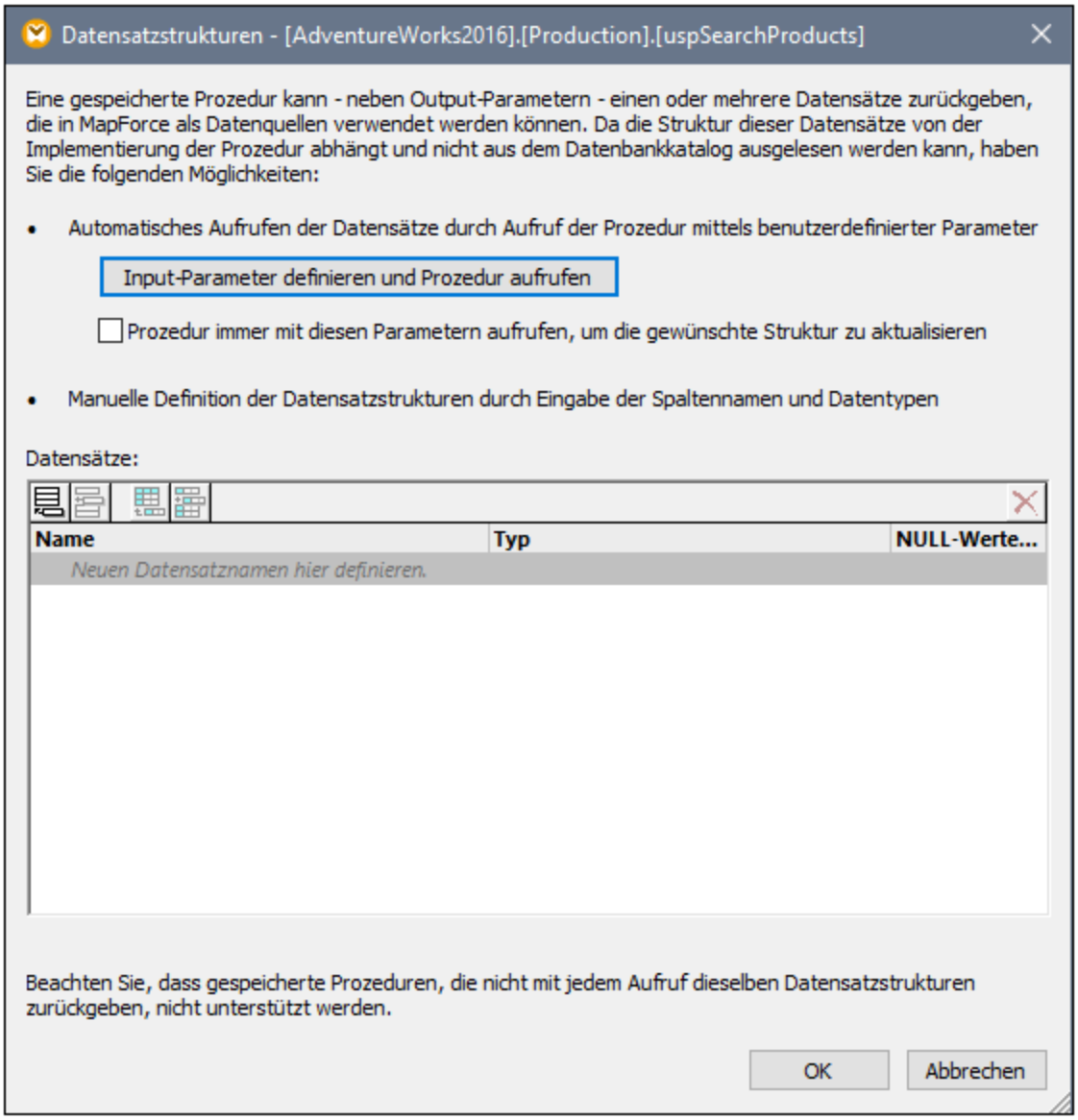
In den folgenden Schritten wird beschrieben, wie Sie ein Mapping erstellen, das von dieser Prozedur zurückgegebene Daten verarbeitet.

1. Stellen Sie in MapForce eine Verbindung zur Datenbank "AdventureWorks" her und fügen Sie die gespeicherte Prozedur zum Mapping hinzu, wie unter Hinzufügen von gespeicherten Prozeduren zum Mapping 405 beschrieben. Vergewissern Sie sich, dass Ihr Datenbankbenutzerkonto Rechte zum Anzeigen und Ausführen von gespeicherten Prozeduren hat.
2. Klicken Sie neben der gespeicherten Prozedur auf die Schaltfläche **Kontextmenü anzeigen** und wählen Sie den Befehl **Aufruf mit Parametern einfügen**. Die gespeicherte Prozedur wird nun in einer separaten Komponente im Mapping angezeigt, wobei auf der linken Seite die Input-Parameter aufgelistet sind und die rechte Seite den Rückgabe- und die Output-Parameter enthält.



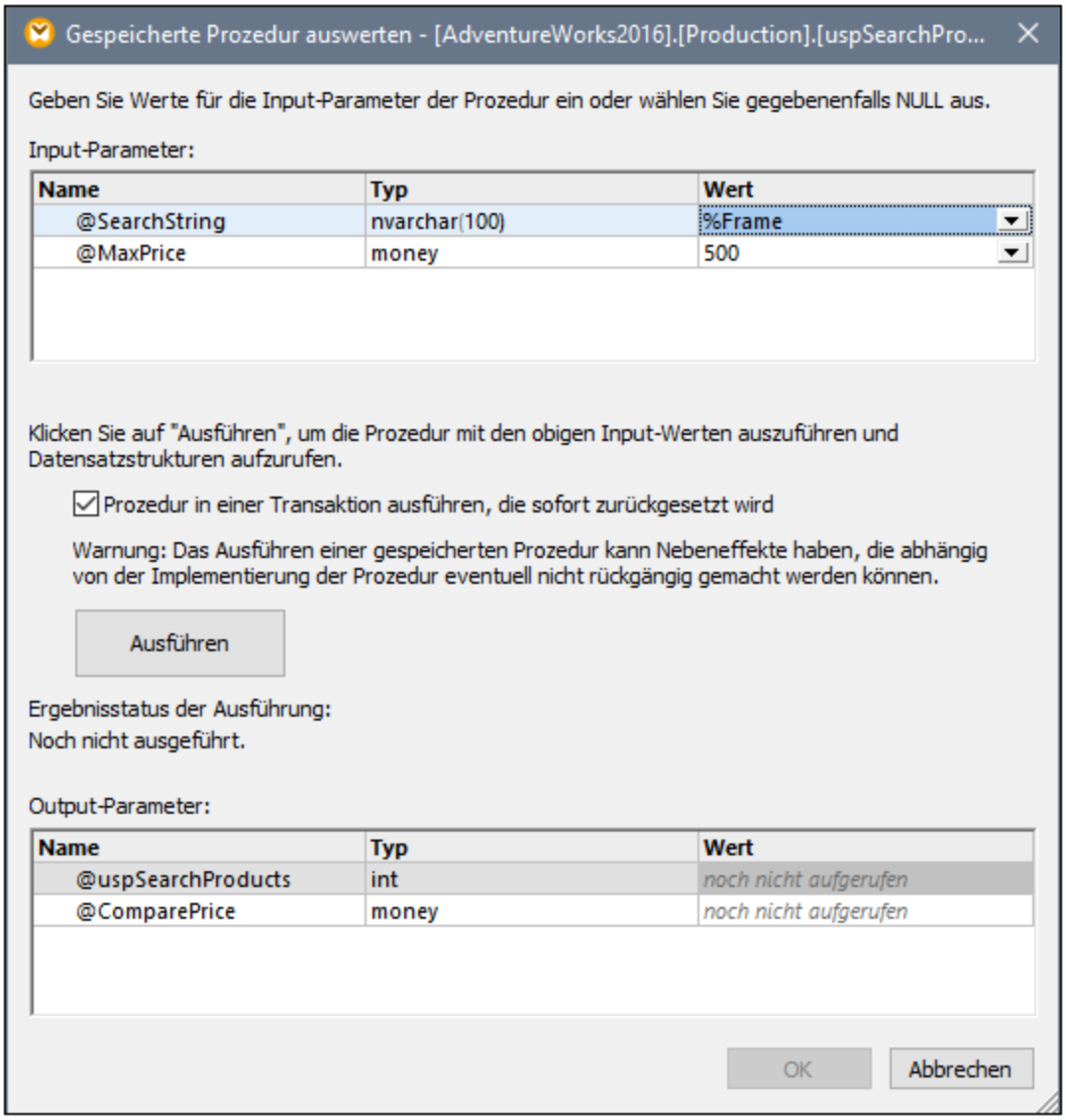
3. Klicken Sie nochmals auf die Schaltfläche **Kontextmenü anzeigen** und wählen Sie **Datensatzstrukturen bearbeiten,** um MapForce Informationen über die Struktur der von der Prozedur zurückgegebenen Datensatzgruppe zu liefern. Daraufhin wird das Dialogfeld "Datensatzstrukturen" angezeigt.

© 2016-2022 Altova GmbH



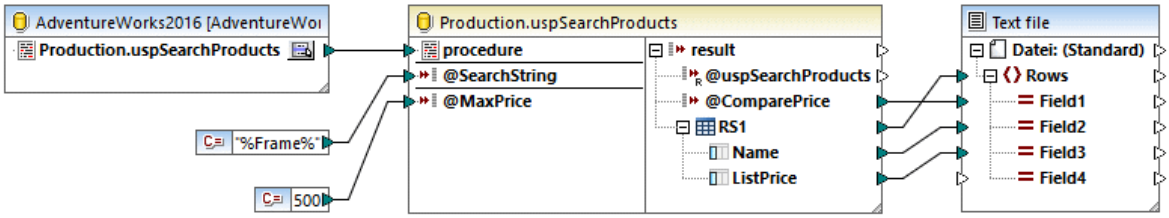
Wenn eine gespeicherte Prozedur während des Designs aufgerufen wird, kann dies (je nach Implementierung der Prozedur) Nebenwirkungen haben. Wenn die gespeicherte Prozedur zum Zeitpunkt des Designs nicht ausgeführt werden soll, klicken Sie nicht, wie in den folgenden Schritten beschrieben, auf **Ausführen**, sondern definieren Sie die erwartete Datensatzstruktur im Dialogfeld "Datensatzstrukturen", indem Sie Datensatzstrukturen und deren dazugehörige Spalten manuell hinzufügen. Verwenden Sie dazu im Dialogfeld "Datensatzstrukturen" die Schaltflächen **Datensatz hinzufügen** oder **Spalte hinzufügen**.

4. Klicken Sie auf **Input-Parameter definieren und Prozedur aufrufen.** Daraufhin wird das Dialogfeld "Gespeicherte Prozedur auswerten" angezeigt.

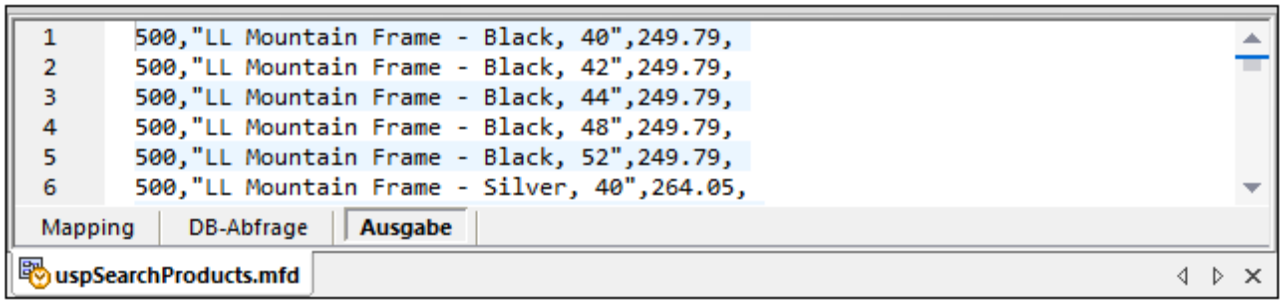


5. Füllen Sie die Parameterwerte, wie oben gezeigt, aus und klicken Sie auf **Ausführen**.
6. Klicken Sie auf **OK**. Die Datensatzstruktur ("RS1") ist nun sowohl im Dialogfeld "Datensatzstrukturen" als auch im Mapping sichtbar.
7. Sie können zu diesem Zeitpunkt eine Zielkomponente, in die die abgerufenen Daten geschrieben werden sollen, hinzufügen. In diesem Beispiel werden die Daten in eine CSV-Datei geschrieben. Klicken Sie im Menü **Einfügen** auf **Textdatei** und fügen Sie eine CSV-Komponente zum Mapping hinzu. Nähere Informationen dazu finden Sie unter CSV und Textdateien[585]. Ziehen Sie als nächstes die Mapping-Verbindungen, wie unten gezeigt. Beachten Sie, dass die Input-Parameter der Prozedur mittels Konstanten bereitgestellt werden. Nähere Informationen zu Konstanten finden Sie unter Hinzufügen einer Konstante zum Mapping[943].

© 2016-2022 Altova GmbH



Sie können nun eine Vorschau auf das Mapping anzeigen. Klicken Sie auf die Schaltfläche **Ausgabe,** um das Ergebnis des Mappings im Fenster **Ausgabe** zu sehen, z.B.:



4.2.9.4 Gespeicherte Prozeduren in Zielkomponeten

In diesem Beispiel wird gezeigt, wie Sie eine Prozedur aufrufen, die Input-Parameter hat und eine Datenbank aktualisiert. Bei dieser Art des Prozeduraufrufs können Transaktionen durchgeführt und die Aktion im Fall eines Fehlers mittels Rollback rückgängig gemacht werden. Oder Sie können eine benutzerdefinierte SQL-Anweisung hinzufügen, die vor dem Aufruf der Prozedur ausgeführt wird. In diesem Szenario wird vorausgesetzt, dass die gespeicherte Prozedur wie eine Zielkomponente in MapForce fungiert und Sie nicht an deren Ergebnis interessiert sind. Ein Beispiel, in dem gezeigt wird, wie Parameter übergeben und von einer gespeicherten Prozedur zurückgegebene Daten gemappt werden, finden Sie unter Gespeicherte Prozeduren mit Input und Output 411.

Erstellen wir zuerst die gespeicherte Demo-Prozedur in der Datenbank "AdventureWorks". Führen Sie zu diesem Zweck das unten stehende Skript an der Datenbank aus. Sie können dies entweder von einem Abfragefenster von **Microsoft SQL Server Management Studio** aus oder direkt auf dem Register **DB-Abfrage** von MapForce (siehe Anzeigen und Abfragen von Datenbanken 388) tun. Vergewissern Sie sich in jedem Fall, dass Ihr Datenbankbenutzerkonto Rechte zum Erstellen von gespeicherten Prozeduren hat.

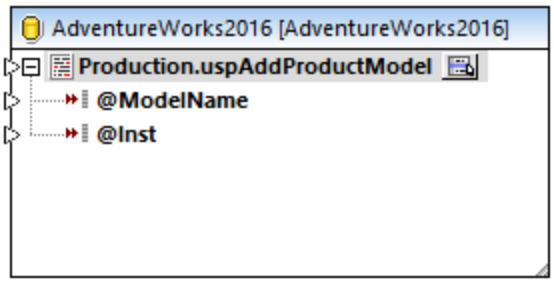
```
CREATE PROCEDURE Production.uspAddProductModel
   @ModelName nvarchar(50)
   ,@Inst xml
AS
BEGIN
INSERT INTO [Production].[ProductModel]
           ([Name]
           ,[Instructions]
           ,[rowguid]
           ,[ModifiedDate])
```

```
    VALUES
          (@ModelName
          ,@Inst
          ,NEWID()
          ,GETDATE())
END
```

Die obige gespeicherte Prozedur hat zwei Parameter (`@ModelName`, `@Inst`) als Input und fügt die entsprechenden Werte zusammen mit einigen datenbankgenerierten Daten in die Tabelle `ProductModel` ein.

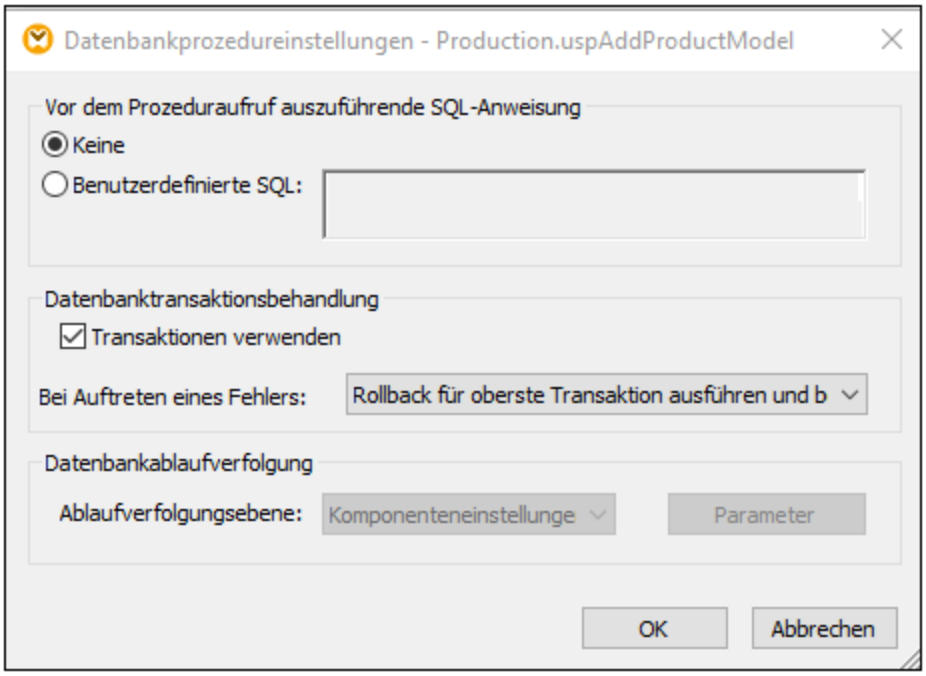
In den folgenden Schritten wird gezeigt, wie Sie ein Mapping erstellen, in dem von dieser Prozedur zurückgegebene Daten verarbeitet werden.

1. Stellen Sie in MapForce eine Verbindung zur Datenbank "AdventureWorks" her und fügen Sie die gespeicherte Prozedur zum Mapping hinzu, wie unter Hinzufügen von gespeicherten Prozeduren zum Mapping 405 beschrieben. Vergewissern Sie sich, dass Ihr Datenbankbenutzerkonto Rechte zum Anzeigen und Ausführen von gespeicherten Prozeduren hat.
2. Klicken Sie neben der gespeicherten Prozedur auf die Schaltfläche **Kontextmenü anzeigen** und wählen Sie den Befehl **Nodes als Ziel anzeigen**. Die gespeicherte Prozedur wird nun als Zielkomponente im Mapping angezeigt, wobei auf der linken Seite die Input-Parameter aufgelistet sind.



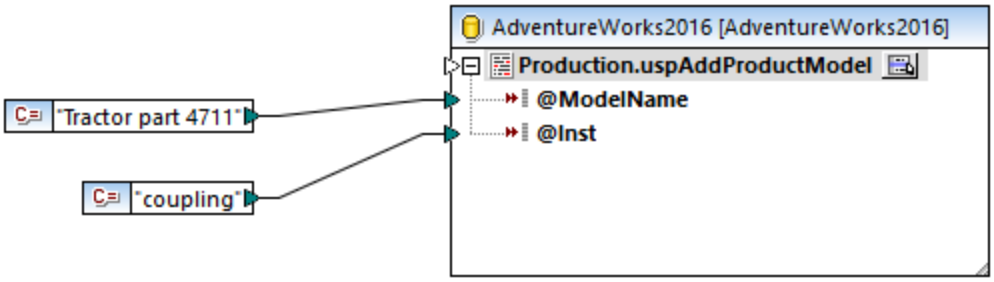
3. Klicken Sie nochmals auf die Schaltfläche **Kontextmenü anzeigen** und wählen Sie **Prozedureinstellungen**. Mit diesem optionalen Schritt können Sie die gespeicherte Prozedur innerhalb einer mit Rollback rückgängig machbaren Transaktion ausführen. Sie können auch eine benutzerdefinierte SQL-Anweisung hinzufügen, die vor dem Aufruf der Prozedur ausgeführt werden soll.
4. Aktivieren Sie das Kontrollkästchen **Transaktionen verwenden**.

© 2016-2022 Altova GmbH

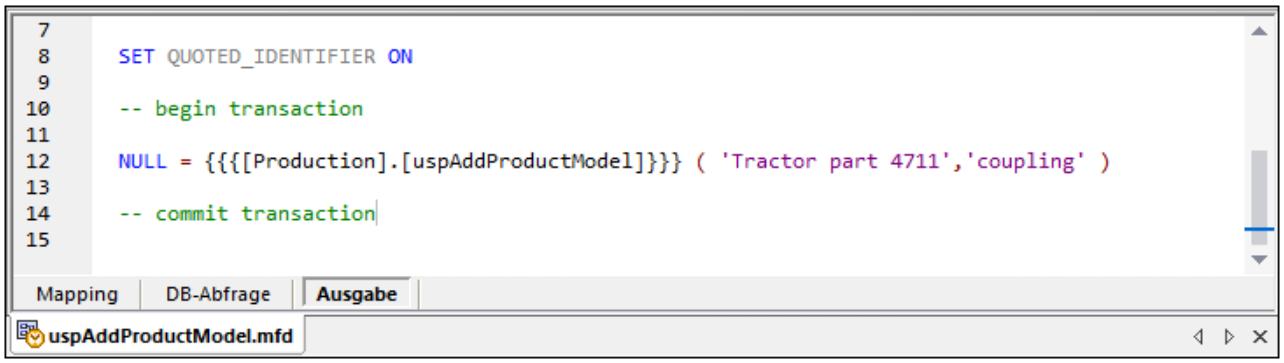


Anmerkung: In diesem Beispiel ist die Datenbankablaufverfolgung auf Ebene der Datenbankkomponente deaktiviert. Es findet keine Ablaufverfolgung statt. Sie können die Datenbankablaufverfolgung jedoch bei Bedarf aktivieren. Nähere Informationen dazu finden Sie unter Datenbankablaufverfolgung und Fehlerprotokollierung[443].

5. Fügen Sie die Quellkomponente, aus der die Daten, die in die Datenbank eingefügt werden sollen, stammen, hinzu. In diesem Beispiel werden die Quelldaten durch Konstanten bereitgestellt, es kann jedoch als Input auch jede andere von MapForce unterstützte Quellkomponente verwendet werden. Nähere Informationen zu Konstanten finden Sie unter Hinzufügen einer Konstante zum Mapping[943].



Da mit diesem Mapping eine Datenbank aktualisiert wird, können Sie die Vorschau auf die Ausgabe nicht wie in anderen Mappings direkt anzeigen. Klicken Sie stattdessen auf die Schaltfläche **Ausgabe**, um die Pseudo-SQL-Anweisung mit Angaben, wie die Datenbank bearbeitet wird, anzuzeigen. Wenn Sie Transaktionen aktiviert haben, werden diese, wie in den Kommentaren angegeben, ausgeführt.



In der im Fenster **Ausgabe** angezeigten Pseudo-SQL-Anweisung werden die eigentlichen Transaktionsbefehle nicht angezeigt. Es werden stattdessen nur Hinweise (in Form von Kommentaren) angezeigt. Die eigentlichen SQL-Befehle werden jedoch an die zugrunde liegende Datenbank-API gesendet.

Um das Mapping an der Datenbank auszuführen, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Ausgabe** auf **SQL-Script ausführen**.
- Klicken Sie in der Symbolleiste auf die Schaltfläche **SQL-Script ausführen** .

Gespeicherte Prozeduren und duplizierte Inputs

Wenn Sie Daten aus mehreren Quellen im Mapping auf dieselbe gespeicherte Prozedur mappen müssen, können Sie die gespeicherte Prozedur duplizieren, sodass sie mehrere Inputs akzeptiert. Klicken Sie dazu mit der rechten Maustaste in der Komponente auf den Eintrag für die gespeicherte Prozedur und wählen Sie im Kontextmenü den Befehl **Duplikat einfügen**, siehe auch Input duplizieren 84. Bei Ausführung des Mappings werden solche duplizierten gespeicherten Prozeduren einmal für jeden duplizierten Input aufgerufen.

Beachten Sie, dass der Befehl **Duplikat einfügen** für die Parameter der gespeicherten Prozedur deaktiviert ist, da jeder Parameter ein atomarer Wert ist (und daher auch auf Null gesetzt werden könnte).

4.2.9.5 Gespeicherte Prozeduren und lokale Beziehungen

Lokale Beziehungen sind logische Beziehungen zwischen Datenbankfeldern, die Sie in MapForce erstellen können, ohne die zugrunde liegende Datenbank bearbeiten zu müssen, siehe auch Definieren von lokalen Beziehungen 276. Lokale Beziehungen können nicht nur für Datenbankfelder, sondern auch für gespeicherte Prozeduren sowohl in Quell- als auch Zielkomponenten definiert werden.

Mit Hilfe von lokalen Beziehungen in Quellkomponenten können Daten aus miteinander in Zusammenhang stehenden Objekten ausgelesen werden. So können z.B. IDs aus einer Datenbanktabelle ausgelesen werden und mit jeder dieser IDs kann eine gespeicherte Prozedur aufgerufen werden, um dazugehörige Informationen abzurufen. Außerdem kann eine gespeicherte Prozedur auch mit aus einer anderen Prozedur abgerufenen Daten aufgerufen werden.

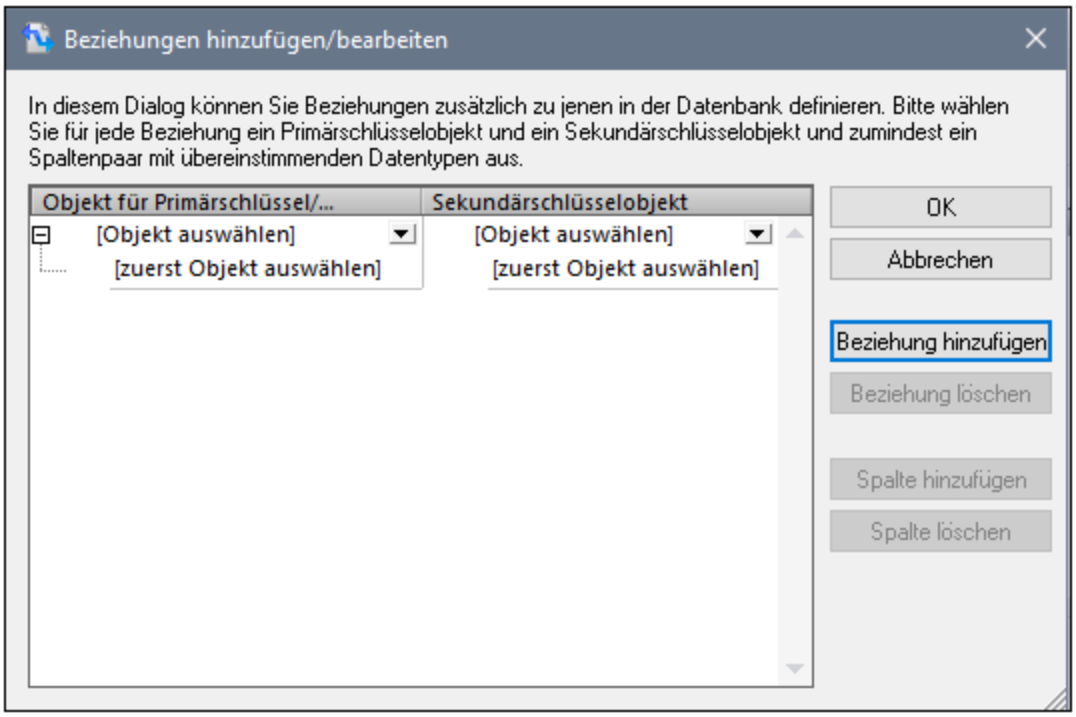
In Zielkomponenten können Sie mit Hilfe von lokalen Beziehungen eine hierarchische Reihung definieren, in der mehrere miteinander in Zusammenhang stehende Prozeduren aufgerufen werden sollen. So können Sie etwa

 © 2016-2022 Altova GmbH

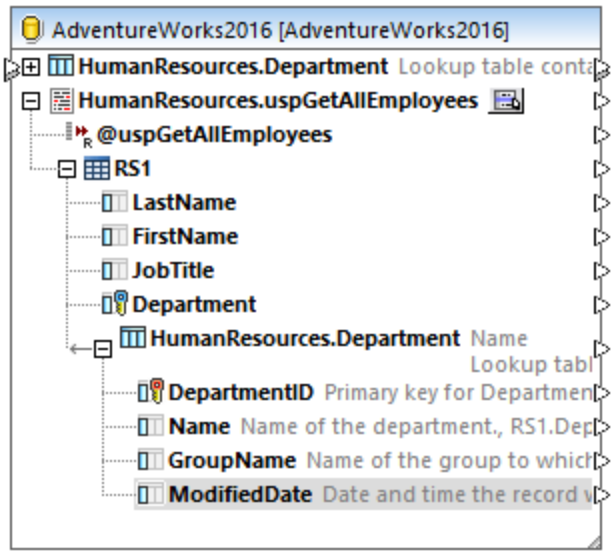
zuerst eine gespeicherte Prozedur aufrufen, die einen ID-Wert erstellt und eine weitere, die dazugehörige Informationen in eine Tabelle einfügt. Gespeicherte Prozeduren und Tabellen können in lokalen Beziehungen auch gemeinsam verwendet werden. So können Sie die Einfügung z.B. direkt in einer damit in Zusammenhang stehenden Tabelle durchführen, anstatt eine weitere Prozedur aufzurufen, siehe Generieren von Schlüsseln mit Hilfe von gespeicherten Prozeduren 424.

So erstellen Sie eine lokale Beziehung:

1. Klicken Sie mit der rechten Maustaste auf die Titelleiste einer Datenbankkomponente und wählen Sie im Kontextmenü den Befehl **Datenbankobjekte hinzufügen/entfernen/bearbeiten**. Daraufhin wird das Dialogfeld "Datenbankobjekte hinzufügen/entfernen/bearbeiten" aufgerufen.
2. Klicken Sie auf **Beziehungen hinzufügen/bearbeiten**.
3. Klicken Sie auf **Beziehung hinzufügen** und wählen Sie die Objekte aus, zwischen denen die Beziehung erstellt werden soll.



Wie oben gezeigt, besteht eine lokale Beziehung aus einem **Objekt für Primärschlüssel/eindeutigen Schlüssel** und einem **Sekundärschlüsselobjekt**. Betrachten Sie die Beziehung als Parent-Child-Beziehung. In der Mapping-Komponente wird das Objekt (Tabelle, Ansicht, Prozedur, usw.), in dem sich der Primärschlüssel/eindeutige Schlüssel befindet, als Parent angezeigt, während das Objekt, in dem der Sekundärschlüssel aufscheint, darunter verschachtelt angezeigt wird. So wurde etwa in der unten gezeigten Datenbankkomponente eine lokale Beziehung zwischen einer Datensatzspalte (**RS1.Department**) und einer Tabellenspalte (**Department.Name**) definiert. Daher wird die Tabelle **Department** als Child der gespeicherten Prozedur im Mapping angezeigt. Eine genauere Beschreibung dieses Beispiels finden Sie unter Lokale Beziehungen in Quellkomponenten 420.



In der folgenden Tabelle sind alle möglichen Felder aufgelistet, zwischen denen lokale Beziehungen definiert werden können. Auch gemischte Beziehungen (z.B. Mappen der Ausgabe einer gespeicherten Prozedur auf eine Datenbankspalte) sind möglich. Die an einer Beziehung beteiligten Felder müssen denselben oder einen kompatiblen Datentyp haben.

Primärschlüssel/eindeutiger Schlüssel	Sekundärschlüssel
• Spalte einer Datenbanktabelle oder -ansicht • Output-Parameter oder Rückgabewert einer gespeicherten Prozedur, siehe auch Gespeicherte Prozeduren 402 • Spalte einer von einer gespeicherten Prozedur zurückgegebenen Datensatzstruktur* • Spalte einer benutzerdefinierten SELECT-Anweisung, siehe auch SELECT-Anweisungen als virtuelle Tabellen 368.	• Spalte einer Datenbanktabelle oder -ansicht • Input-Parameter einer gespeicherten Prozedur • Input-Parameter einer benutzerdefinierten SELECT-Anweisung

** Anwendbar, wenn die gespeicherte Prozedur entweder als Datenquelle (ohne Parameter) oder Funktion (mit Input- und Output-Parametern) aufgerufen wird. Damit die Datensatzstruktur für die Auswahl zur Verfügung steht, müssen Sie die gespeicherte Prozedur einmal ausführen, um die Datensatzstruktur abzurufen.*

4.2.9.6 Lokale Beziehungen in Quellkomponenten

In diesem Beispiel wird gezeigt, wie Sie von einer gespeicherten Prozedur zurückgegebene Daten mit Hilfe lokaler Beziehungen mit Daten aus einer Tabelle in derselben Datenbank kombinieren.

Starten Sie das folgende Skript, um die gespeicherte Demo-Prozedur in der Datenbank "AdventureWorks" zu erstellen, falls dies noch nicht geschehen ist. Sie können dies entweder von einem Abfragefenster von **Microsoft SQL Server Management Studio** aus oder direkt auf dem Register **DB-Abfrage** von MapForce

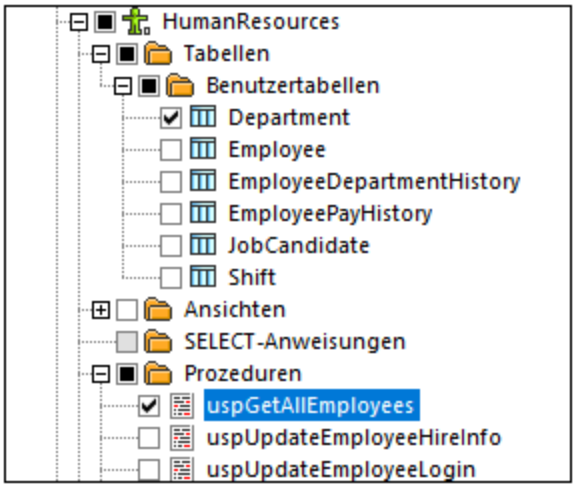
 © 2016-2022 Altova GmbH

(siehe Anzeigen und Abfragen von Datenbanken[388]) tun. Vergewissern Sie sich in jedem Fall, dass Ihr Datenbankbenutzerkonto Rechte zum Erstellen von gespeicherten Prozeduren hat.

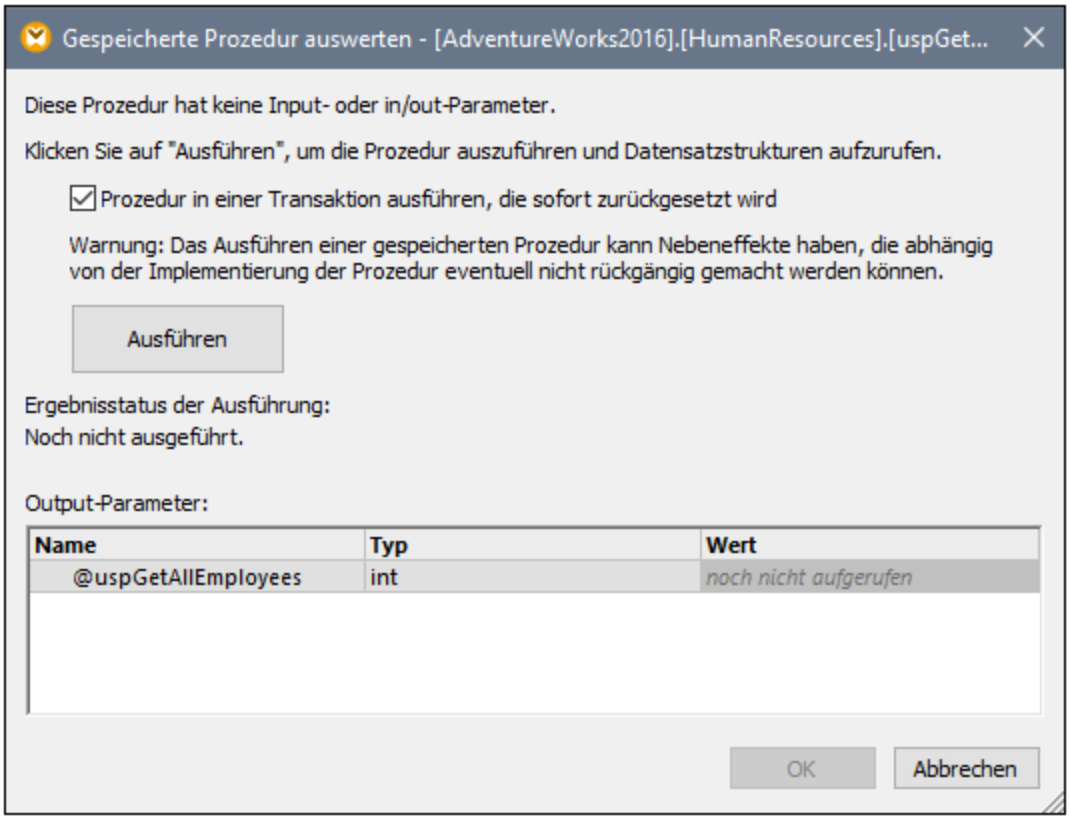
```
CREATE PROCEDURE HumanResources.uspGetAllEmployees
AS
    SELECT LastName, FirstName, JobTitle, Department
    FROM HumanResources.vEmployeeDepartment
```

Die obige gespeicherte Prozedur gibt Mitarbeiterdaten aus der Ansicht **vEmployeeDepartment** zurück. In der folgenden Anleitung wird beschrieben, wie Sie ein Mapping erstellen, das die von dieser Prozedur zurückgegebenen Daten verarbeitet.

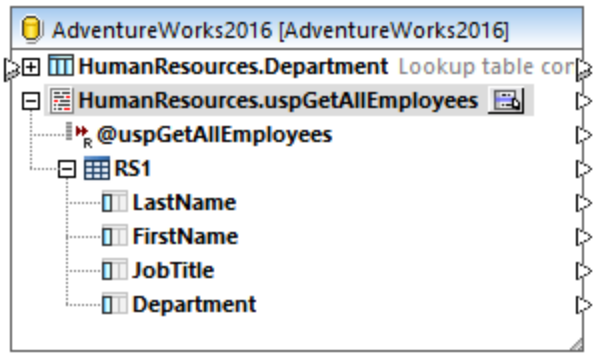
1. Stellen Sie in MapForce eine Verbindung zur Datenbank "AdventureWorks" her, wie unter Hinzufügen von gespeicherten Prozeduren zum Mapping[405] beschrieben. Vergewissern Sie sich, dass Ihr Datenbankbenutzerkonto Rechte zum Anzeigen und Ausführen von gespeicherten Prozeduren hat.
2. Wenn Sie aufgefordert werden, Datenbankobjekte auszuwählen, wählen Sie die Tabelle **Department** und die gespeicherte Prozedur **uspGetAllEmployees** aus.



3. Klicken Sie neben der gespeicherten Prozedur auf die Schaltfläche **Kontextmenü anzeigen** und wählen Sie den Befehl **Nodes als Quelle anzeigen**.
4. Klicken Sie nochmals auf die Schaltfläche **Kontextmenü anzeigen** und wählen Sie **Datensatzstrukturen bearbeiten**. Daraufhin wird das Dialogfeld "Datensatzstrukturen" angezeigt.
5. Klicken Sie auf **Input-Parameter definieren und Prozedur aufrufen.** Daraufhin wird das Dialogfeld "Gespeicherte Prozedur auswerten" angezeigt.



6. Klicken Sie auf **Ausführen** und anschließend auf **OK**. Die Datensatzstruktur ("RS1") ist nun sowohl im Dialogfeld "Datensatzstrukturen" als auch im Mapping sichtbar.

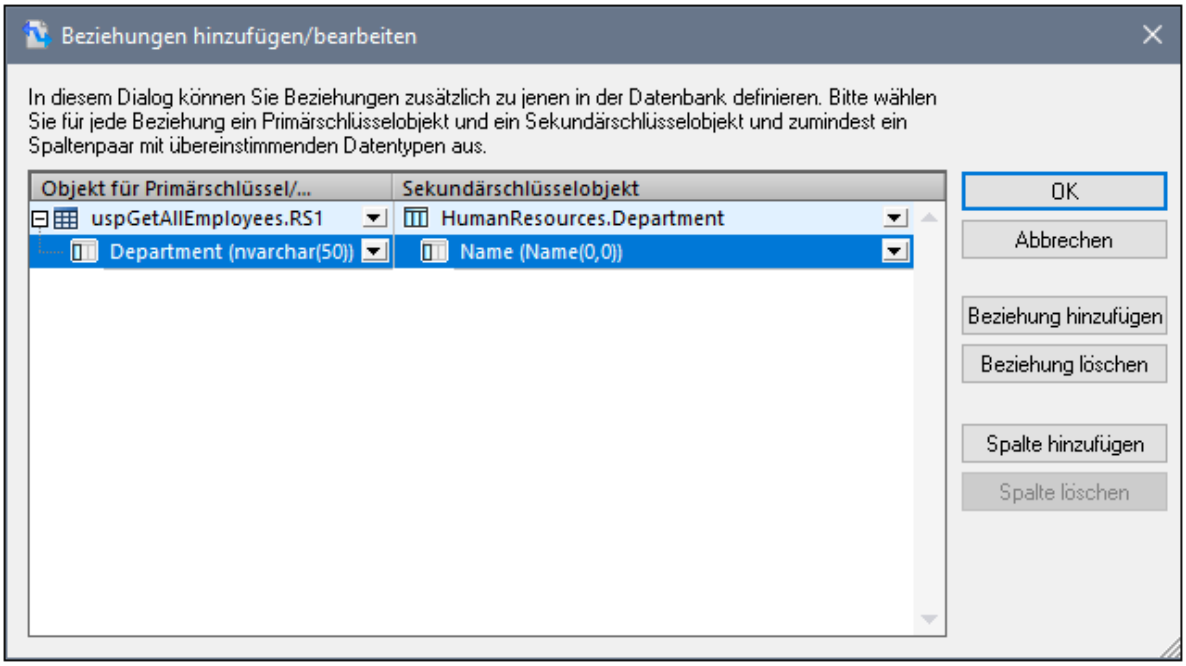


Definieren der lokalen Beziehungen

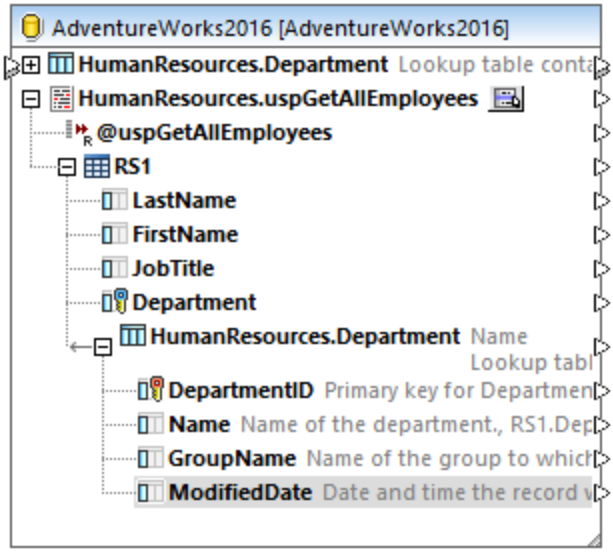
Wir wollen nun eine lokale Beziehung zwischen der Spalte **Department** der zurückgegebenen Datensatzstruktur und der Spalte **Name** der Tabelle **Department** definieren.

1. Klicken Sie mit der rechten Maustaste auf die Titelleiste einer Datenbankkomponente und wählen Sie im Kontextmenü den Befehl **Datenbankobjekte hinzufügen/entfernen/bearbeiten**.
2. Klicken Sie auf **Beziehungen hinzufügen/bearbeiten** und anschließend auf **Beziehung hinzufügen.** Definieren Sie die Beziehung, wie unten gezeigt.

© 2016-2022 Altova GmbH



3. Klicken Sie auf **OK,** um das Dialogfeld zu schließen. Beachten Sie, dass die Tabelle **Department** nun zu einem Child der Datensatzstruktur **RS1** geworden ist.

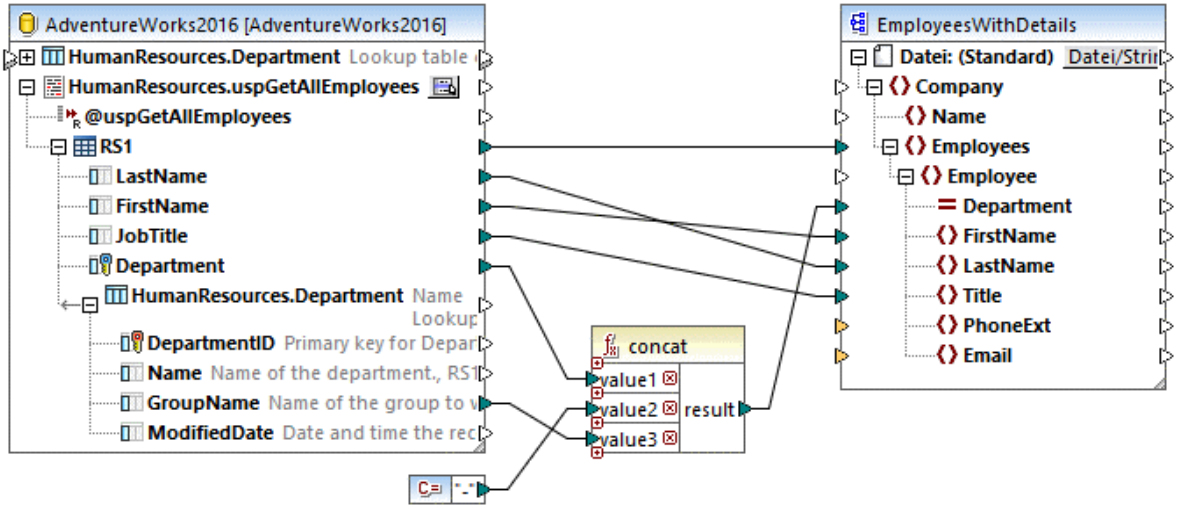


Fertigstellung des Mappings

Dank der soeben erstellen Beziehung können nun Daten aus der Datensatzstruktur, die mit Daten aus der Tabelle kombiniert wurden, gemappt werden. In diesem Beispiel werden Daten folgendermaßen in eine XML-Zieldatei geschrieben.

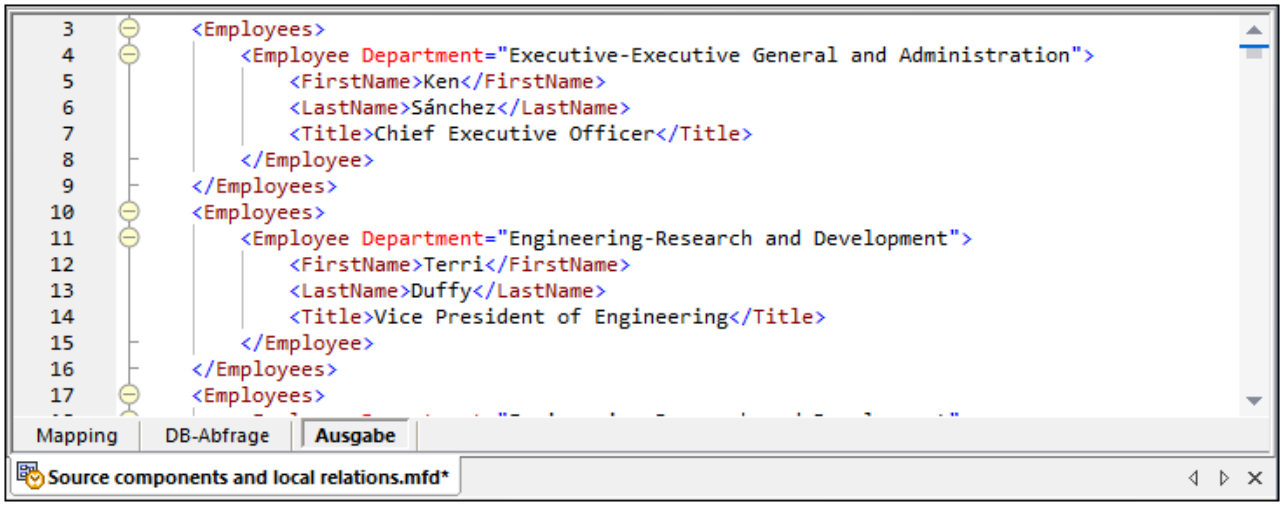
1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und wählen Sie die folgende Datei aus: **<Dokumente>\Altova\MapForce2023\MapForceExamples\EmployeesWithDetails.xsd**.

2. Wenn Sie aufgefordert werden, eine XML-Beispieldatei bereitzustellen, klicken Sie auf **Überspringen**.
3. Ziehen Sie die Mapping-Verbindungen, wie unten gezeigt.



Im oben gezeigten Mapping werden Daten aus der Datenbank in eine XML-Zieldatei geschrieben. Bei den Quelldaten handelt es sich um Daten, die durch die gespeicherte Prozedur extrahiert wurden und mit direkt aus einer Tabelle extrahierten Daten kombiniert wurden. Mit Hilfe der **concat**[1105]-Funktion wird ein String erzeugt, der den Abteilungsnamen, gefolgt von einem Bindestrich, gefolgt vom Gruppennamen enthält.

Sie können nun eine Vorschau auf das Mapping anzeigen. Klicken Sie auf die Schaltfläche **Ausgabe,** um das Ergebnis des Mappings im Fenster **Ausgabe** zu sehen, z.B.:



4.2.9.7 Generieren von Schlüsseln mit Hilfe von gespeicherten Prozeduren

In diesem Beispiel wird gezeigt, wie Sie einen durch eine gespeicherte Prozedur generierten Schlüssel (ID) mit Hilfe lokaler Beziehungen in eine andere Tabelle einfügen.

© 2016-2022 Altova GmbH

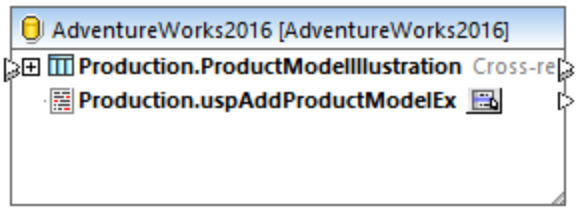
Erstellen wir zuerst die gespeicherte Demo-Prozedur in der Datenbank "AdventureWorks". Führen Sie zu diesem Zweck das unten stehende Skript an der Datenbank aus. Sie können dies entweder von einem Abfragefenster von **Microsoft SQL Server Management Studio** aus oder direkt auf dem Register **DB-Abfrage** von MapForce (siehe Anzeigen und Abfragen von Datenbanken 388) tun. Vergewissern Sie sich in jedem Fall, dass Ihr Datenbankbenutzerkonto Rechte zum Erstellen von gespeicherten Prozeduren hat.

```
CREATE PROCEDURE Production.uspAddProductModelEx
  @ModelName nvarchar(50)
  ,@Inst xml
  ,@ProductModelID int OUTPUT
AS
BEGIN
INSERT INTO [Production].[ProductModel]
           ([Name]
           ,[Instructions]
           ,[rowguid]
           ,[ModifiedDate])
     VALUES
           (@ModelName
           ,@Inst
           ,NEWID()
           ,GETDATE())
   SELECT @ProductModelID = SCOPE_IDENTITY()
END
```

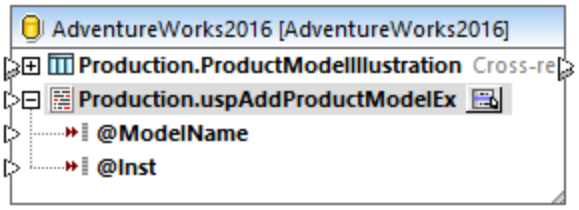
Die obige gespeicherte Prozedur hat zwei Parameter (`@ModelName`, `@Inst`) als Input und führt an der Tabelle **ProductModel** eine `INSERT`-Operation durch. Anschließend gibt sie die generierte Produktmodell-ID `@ProductModelID` als Output-Parameter zurück. Die von der gespeicherten Prozedur zurückgegebene `@ProductModelID` soll in die Tabelle **ProductModelIllustration** eingefügt werden.

In der folgenden Anleitung wird beschrieben, wie Sie ein Mapping für die obige Anforderung erstellen.

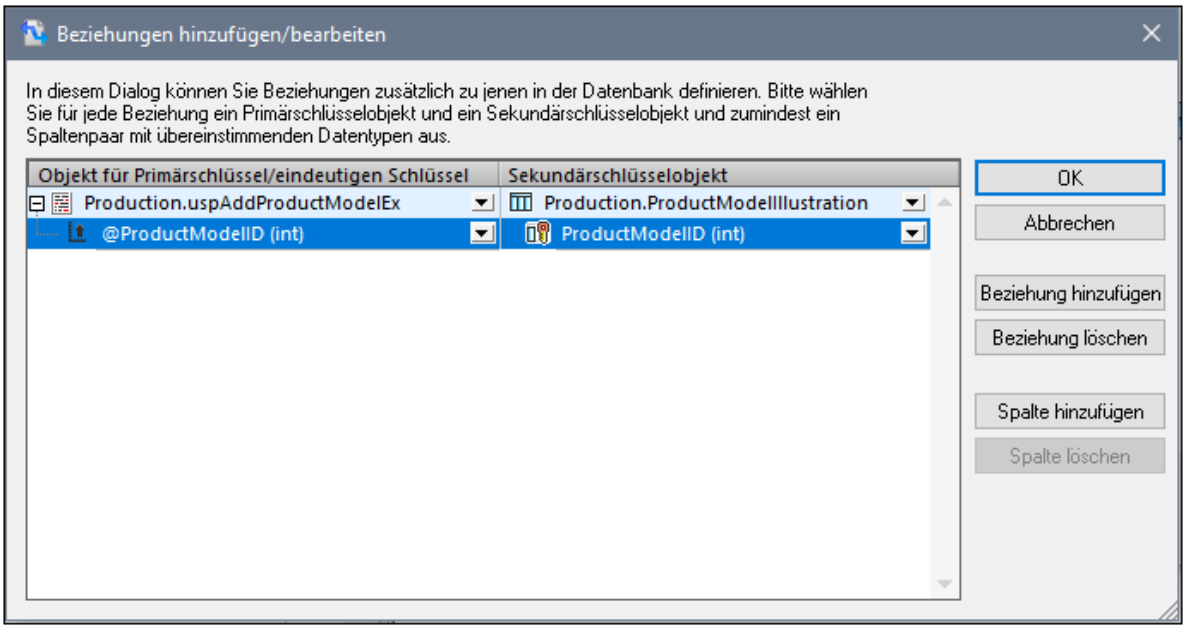
1. Stellen Sie in MapForce eine Verbindung zur Datenbank "AdventureWorks" her, wie unter Hinzufügen von gespeicherten Prozeduren zum Mapping 405 beschrieben. Vergewissern Sie sich, dass Ihr Datenbankbenutzerkonto Rechte zum Anzeigen und Ausführen von gespeicherten Prozeduren hat.
2. Wenn Sie aufgefordert werden, Datenbankobjekte auszuwählen, wählen Sie die Tabelle **ProductModelIllustration** und die gespeicherte Prozedur **uspAddProductModelEx** aus.



3. Klicken Sie neben der gespeicherten Prozedur auf die Schaltfläche **Kontextmenü anzeigen** und wählen Sie den Befehl **Nodes als Ziel anzeigen**. Die gespeicherte Prozedur wird nun als Zielkomponente im Mapping angezeigt, wobei auf der linken Seite die Input-Parameter aufgelistet sind.

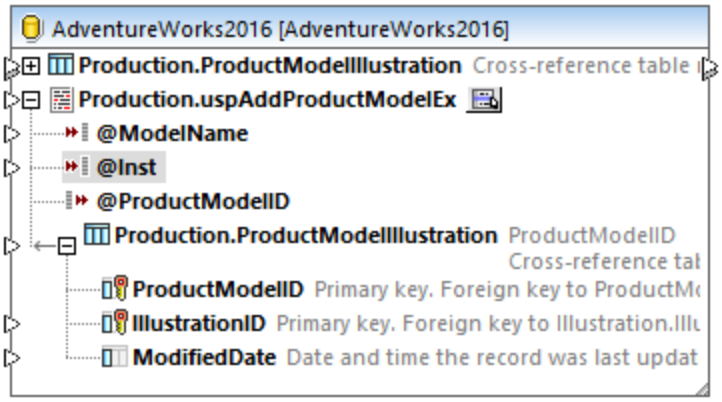


4. Klicken Sie optional nochmals auf die Schaltfläche **Kontextmenü anzeigen**, wählen Sie **Prozedureinstellungen** und aktivieren Sie anschließend das Kontrollkästchen **Transaktionen verwenden,** wenn die gespeicherte Prozedur innerhalb einer Transaktion ausgeführt werden soll. Durch Definieren der Transaktion für die gespeicherte Prozedur stellen Sie sicher, dass der Abruf des Schlüssels und das Einfügen des Datensatzes während derselben Transaktion erfolgen.
5. Klicken Sie mit der rechten Maustaste auf die Titelleiste der Datenbankkomponente und wählen Sie im Kontextmenü den Befehl **Datenbankobjekte hinzufügen/entfernen/bearbeiten**.
6. Klicken Sie auf **Beziehungen hinzufügen/bearbeiten** und anschließend auf **Beziehung hinzufügen.** Definieren Sie die Beziehungen, wie unten gezeigt.

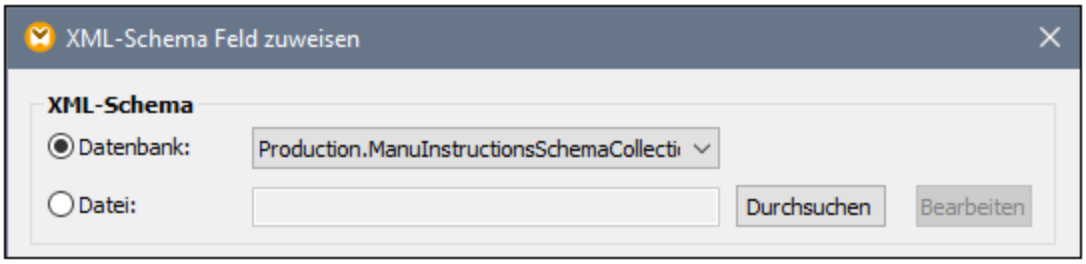


7. Klicken Sie auf **OK,** um das Dialogfeld zu schließen. Beachten Sie, dass die Tabelle **ProductModelIllustration** nun als Child der gespeicherten Prozedur angezeigt wird. Der Output-Parameter (`@ProductModelID`) der gespeicherten Prozedur wird als Indikator angezeigt, dass er in der lokalen Beziehung verwendet wird. Er hat jedoch keine Input- oder Output-Konnektoren.

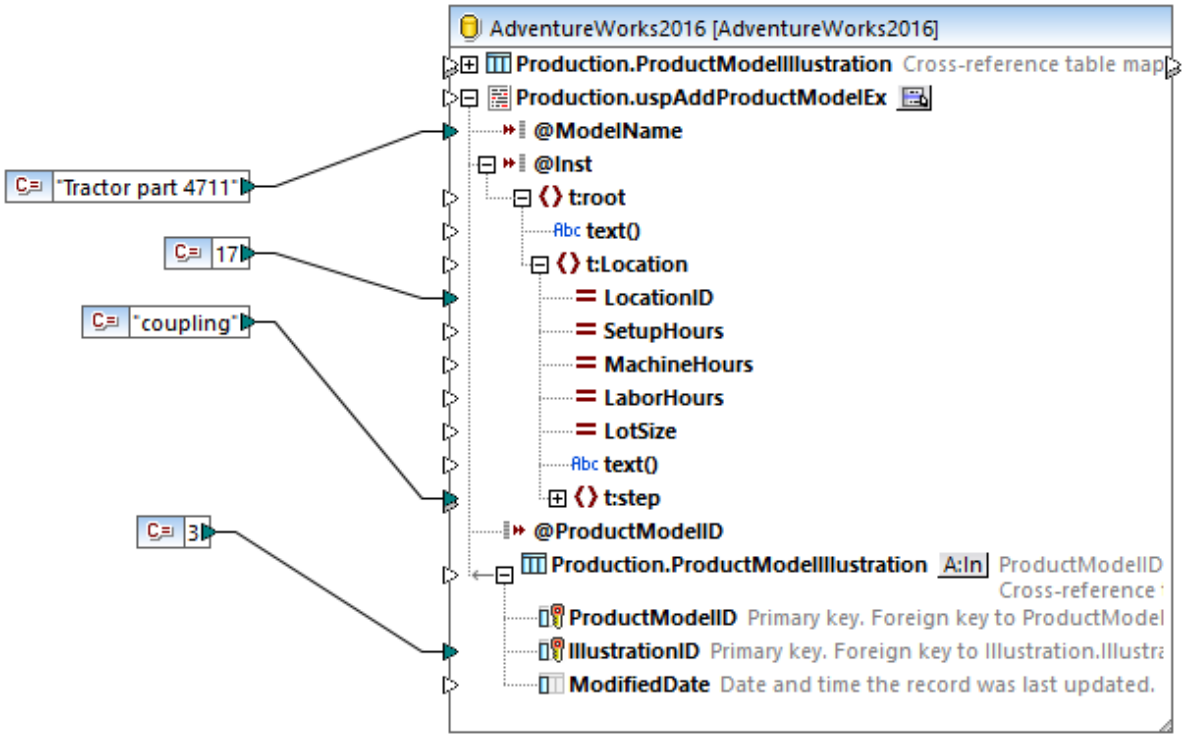
© 2016-2022 Altova GmbH



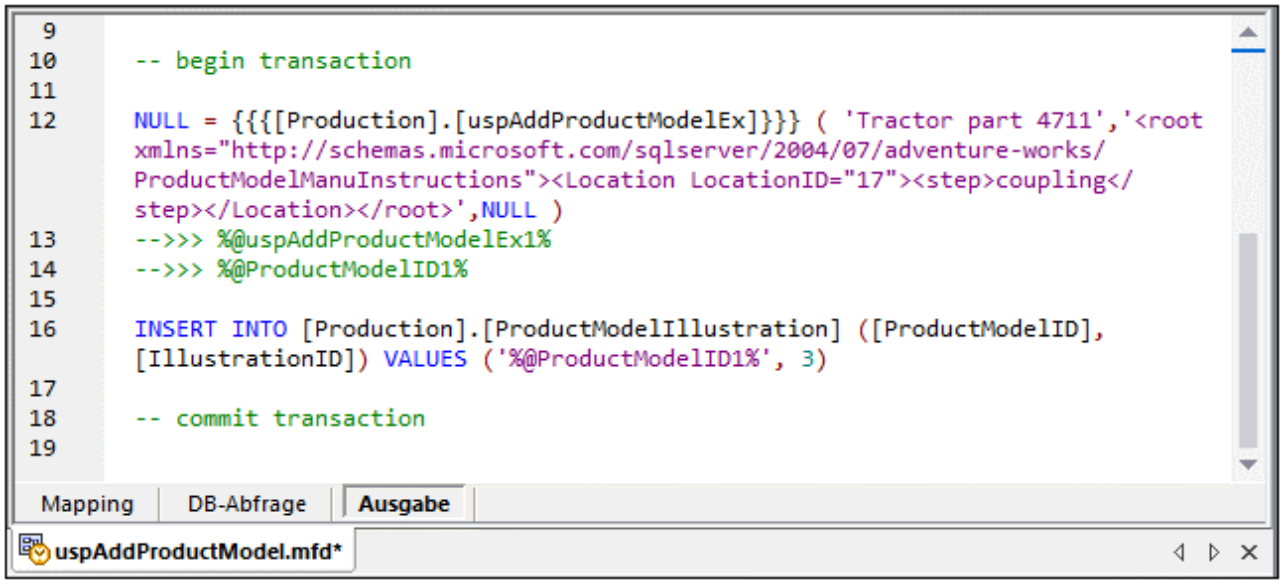
8. Der Parameter `@Inst` in diesem Beispiel hat den Typ XML. Klicken Sie mit der rechten Maustaste in der Komponente auf den Parameter `@Inst` und wählen Sie im Kontextmenü den Befehl **XML-Schema Feld zuweisen**. Wählen Sie als nächstes das Schema **Production.ManuInstructionsSchemaCollection** aus der Datenbank aus. Wenn Sie aufgefordert werden, ein Root-Element auszuwählen, übernehmen Sie den Standardwert unverändert und klicken Sie auf **OK**. Nähere Informationen zum Mappen von Daten auf XML-Datenbankfelder finden Sie unter Mappen von XML-Daten von/auf Datenbankfelder 377.



9. Fügen Sie die Quellkomponenten, aus denen die Daten, die in die Datenbank eingefügt werden sollen, stammen, hinzu. In diesem Beispiel werden die Quelldaten durch Konstanten bereitgestellt, es kann jedoch als Input auch jede andere von MapForce unterstützte Quellkomponente verwendet werden. Nähere Informationen zu Konstanten finden Sie unter Hinzufügen einer Konstante zum Mapping 943.



Da mit diesem Mapping eine Datenbank aktualisiert wird, können Sie die Vorschau auf die Ausgabe nicht wie in anderen Mappings direkt anzeigen. Klicken Sie stattdessen auf die Schaltfläche **Ausgabe**, um die Pseudo-SQL-Anweisung mit Angaben, wie die Datenbank bearbeitet wird, anzuzeigen. Wenn Sie Transaktionen aktiviert haben, werden diese, wie in den Kommentaren angegeben, ausgeführt.



In der im Fenster **Ausgabe** angezeigten Pseudo-SQL-Anweisung werden die eigentlichen

© 2016-2022 Altova GmbH

Transaktionsbefehle nicht angezeigt. Es werden stattdessen nur Hinweise (in Form von Kommentaren) angezeigt. Die eigentlichen SQL-Befehle werden jedoch an die zugrunde liegende Datenbank-API gesendet.

Um das Mapping an der Datenbank auszuführen, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Ausgabe** auf **SQL-Script ausführen**.
- Klicken Sie in der Symbolleiste auf die Schaltfläche **SQL-Script ausführen** .

4.2.10 Behandlung von Datenbankausnahmen

Bei der Ausführung von Mappings, bei denen Änderungen in einer Datenbank vorgenommen werden, können in MapForce (oder MapForce Server oder einem mit MapForce generierten Programm) unter Umständen im Zusammenhang mit der Datenbank verschiedene Fehler auftreten. So können für das Datenbankkonto eventuell nicht genügend Rechte vorhanden sein, um eine bestimmte Datenbankaktion auszuführen oder es können Datenbank-Constraints vorhanden sein, durch die verhindert wird, dass ungültige Daten durch das Mapping eingefügt oder aktualisiert werden. Letzteres kann z.B. vorkommen, wenn eine Tabellenspalte obligatorisch ist, aber vom Mapping kein Wert dafür bereitgestellt wird.

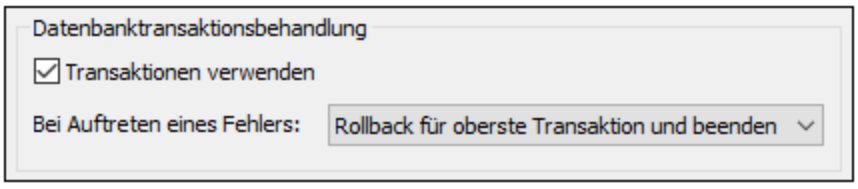
Damit bei Datenbankausnahmen ein Rollback durchgeführt werden kann, können Sie für das Mapping Datenbanktransaktionen konfigurieren. Sie können die Verwendung von Transaktionen, wie unten beschrieben, auf Ebene der Datenbankkomponente und auf Ebene der Tabellenaktionen aktivieren.

Die in MapForce konfigurierten Transaktions-Rollback-Einstellungen werden beibehalten, wenn Sie das Mapping entweder mit MapForce Server oder einem mit MapForce generierten Programm ausführen.

Transaktionen auf Datenbankkomponentenebene

Um Transaktionen auf Datenbankkomponentenebene zu aktivieren, gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf die Datenbankkomponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften** aus. (Oder doppelklicken Sie alternativ dazu auf die Titelleiste der Datenbankkomponente). Daraufhin wird das Dialogfeld "Komponenteneinstellungen" der Datenbank geöffnet.



2. Aktivieren Sie das Kontrollkästchen **Transaktionen verwenden**.

Dadurch werden alle von der Datenbankkomponente vorgenommenen Änderungen in eine Transaktion gesetzt, die bei Auftreten eines Fehlers mit Rollback rückgängig gemacht wird. Wenn während der Ausführung eines Mappings ein Fehler auftritt, hängt das Ergebnis von der Option ab, die Sie aus der Dropdown-Liste auswählen:

- **Rollback für oberste Transaktion und beenden** - Die Transaktion, innerhalb der alle an der Datenbankkomponente vorgenommenen Änderungen stehen, wird mit Rollback rückgängig gemacht und die Ausführung des Mappings wird abgebrochen.
- **Rollback für oberste Transaktion und fortfahren** - Wie oben, doch wird die Ausführung des Mappings nach dem Rollback fortgesetzt (z.B., um eine zweite Zielkomponente, falls vorhanden, zu verarbeiten).

Einige Mappings können mehrere Datenbankkomponenten enthalten. Für all diese können wiederum entweder dieselbe Datenbankverbindung oder auch Verbindungen zu komplett anderen Datenbanken verwendet werden. Das Ergebnis solcher Mappings hängt im Fall eines Datenbankfehlers vom Ausführungsprozessor ab:

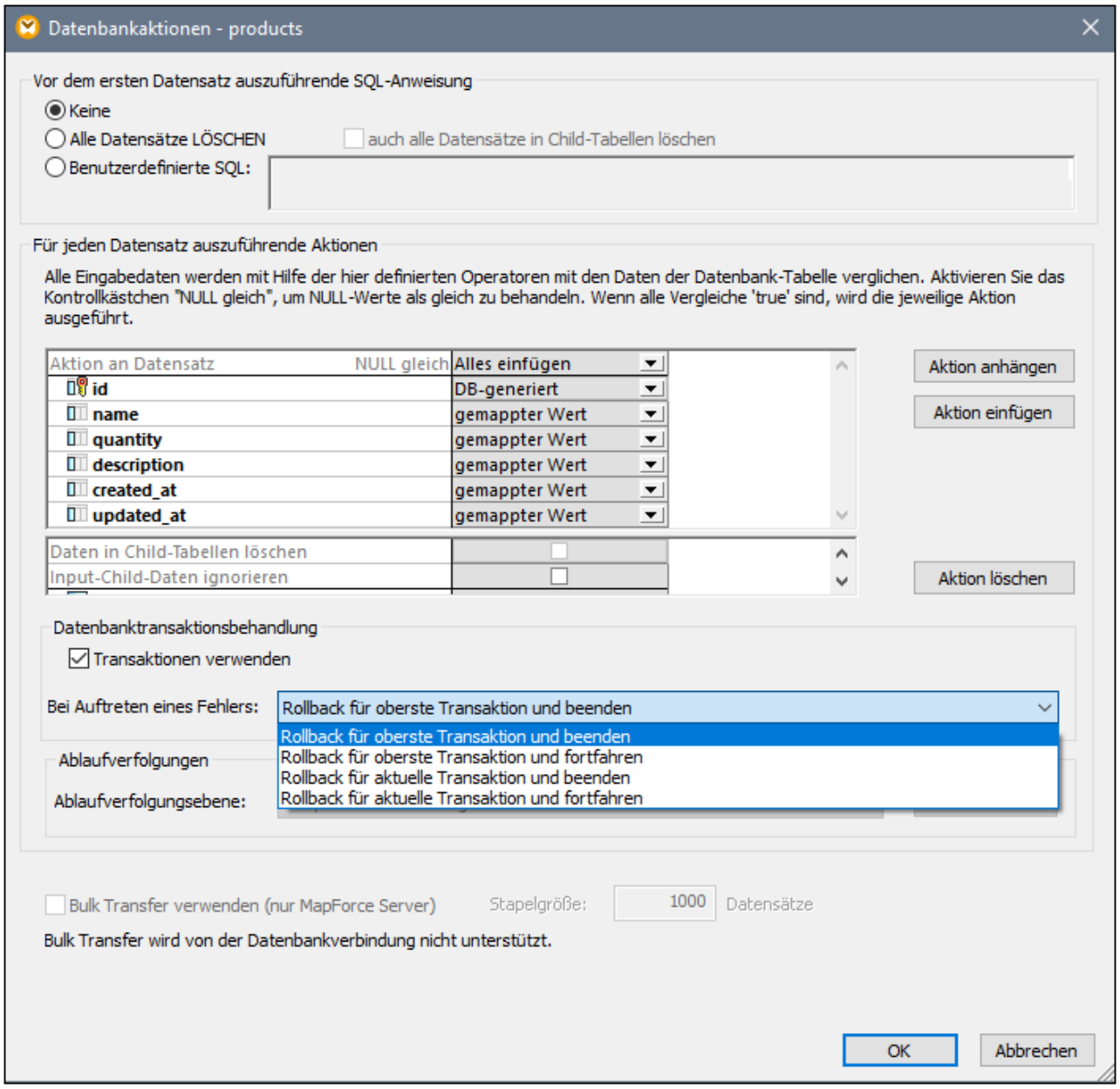
- Wenn das Mapping mit MapForce ausgeführt wird, kann während der Ausführung des Mappings nur eine Zielkomponente verarbeitet werden. Dies ist die Komponente, für die die Schaltfläche **Vorschau**  aktiviert ist. Wenn in dieser Komponente ein Datenbankfehler auftritt und das Kontrollkästchen **Transaktionen verwenden** aktiviert ist, so werden alle Änderungen an der Komponente mit einem Rollback rückgängig gemacht.
- Wenn das Mapping mit MapForce Server oder einem mit MapForce generierten Programm ausgeführt wird, werden alle Zielkomponenten der Reihe nach verarbeitet. In diesem Fall wird bei einem Datenbankfehler für die Datenbankkomponente, bei der der Fehler aufgetreten ist, ein Rollback durchgeführt. Je nachdem, welchen Eintrag Sie in der Dropdown-Liste **Bei Auftreten eines Fehlers** ausgewählt haben, wird das Mapping nun entweder beendet oder es wird mit der Verarbeitung der nächsten Zielkomponente fortgefahren.

Nähere Informationen zu allen Einstellungen des Dialogfelds "Komponenteneinstellungen" für Datenbanken finden Sie unter Datenbank-Komponenteneinstellungen 286.

Transaktionen auf Tabellenaktionsebene

Bei Transaktionen auf Ebene von Tabellenaktionen wird jede einzelne Tabellenaktion (INSERT, UPDATE, usw.) in eine Transaktion eingeschlossen. Wenn eine Einfüge- oder Aktualisierungsaktion bei dieser Konfiguration fehlschlägt, kann nur die jeweilige fehlgeschlagene Aktion mit Rollback rückgängig gemacht werden und andere Datensätze können weiterhin verarbeitet werden. Um Transaktionen auf Tabellenaktionsebene zu aktivieren, aktivieren Sie im Dialogfeld "Datenbankaktionen" das Kontrollkästchen **Transaktionen verwenden** (siehe auch Einstellungen für Datenbankaktionen 336).

 © 2016-2022 Altova GmbH



Es stehen die folgenden Optionen zur Verfügung:

- **Rollback für oberste Transaktion und beenden** - Wenn im Mapping mehrere Tabellen mit einer Parent-Child-Beziehung aktualisiert werden sollen, können Transaktionen nicht nur für die Parent-Tabelle, sondern auch für die Child-Tabelle aktiviert werden. Da Transaktionen auf mehreren Ebenen aktiviert werden können, bezieht sich "oberste" auf die Transaktion auf der obersten Ebene in der Hierarchie. Dies ist die Transaktion ab Ebene der Datenbankkomponente, wenn Sie Transaktionen auf dieser Ebene aktiviert haben. Andernfalls ist es die auf der obersten Tabellenaktion definierte Transaktion, wenn die Komponente mehrere verschachtelte Tabellenaktionen hat. Um diese Option zu verstehen, bedenken Sie, dass in einer Komponente mehrere hierarchisch verschachtelte Tabellen aktualisiert werden können, wie unter Einfügen von Daten in mehrere verknüpfte Tabellen 295

beschrieben. So kann eine Komponente z.B. eine übergeordnete "INSERT"-Aktion haben, die Personendatensätze einfügt, und eine darunter verschachtelte "INSERT"-Aktion, die für jeden Personendatensatz eine oder mehrere Adressen in eine separate Tabelle einfügt. Da Sie für beide INSERT-Aktionen Transaktionen aktivieren können, bezieht sich "oberste" auf die oberste Ebene in der Hierarchie (in diesem Beispiel auf die Einfügeaktion für Personen). Wenn daher beim Einfügen einer Adresse eine Ausnahme auftritt, wird die oberste Transaktion (in diesem Fall auf Ebene der person-Tabelle) rückgängig gemacht und die Ausführung wird beendet. Anders ausgedrückt, wird kein Personendatensatz eingefügt, wenn die Adresse der Person nicht eingefügt werden konnte.

- **Rollback für oberste Transaktion und fortfahren** - Wie oben, doch wird die Ausführung des Mappings nach dem Rollback fortgesetzt (z.B. um, eine zweite Zielkomponente, falls vorhanden, zu verarbeiten).
- **Rollback für aktuelle Transaktion und beenden** - Bei einer Ausnahme werden nur die in die aktuelle Transaktion eingeschlossenen Änderungen rückgängig gemacht; zuvor außerhalb der aktuellen Transaktion durchgeführte Änderungen werden in die Datenbank übernommen.
- **Rollback für aktuelle Transaktion und fortfahren** - Wie oben, doch wird das Mapping nach Ausführung des Rollbacks weiter ausgeführt.

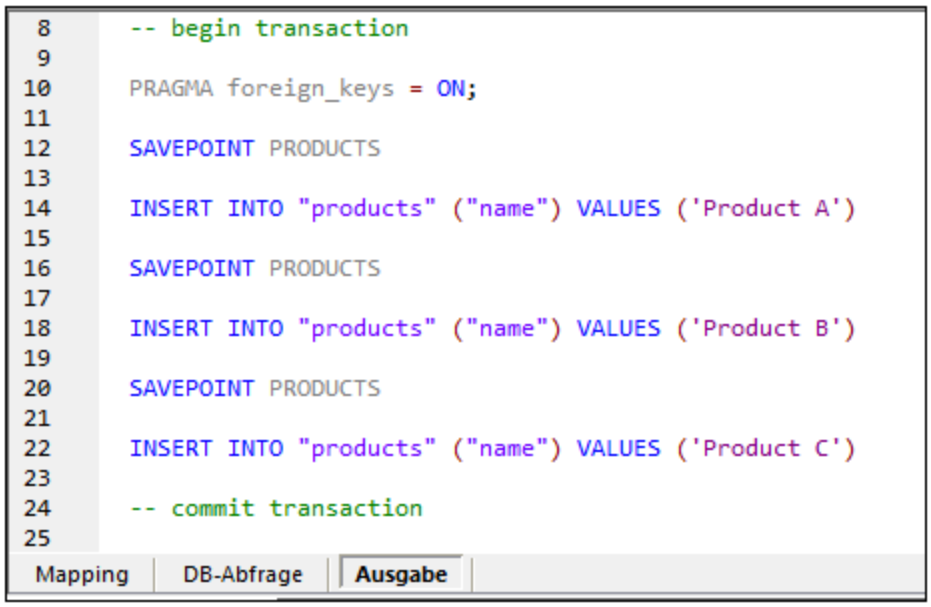
Transaktionen auf Ebene von gespeicherten Prozeduren

Sie können Datenbanktransaktionen auch für gespeicherte Prozeduren, die als Datenzielkomponente aufgerufen werden, aktivieren. Ein Beispiel dafür finden Sie unter Gespeicherte Prozeduren in Zielkomponenten 415. Bei gespeicherten Transaktionen, die als Datenquelle (ohne Input-Parameter) oder solchen, die wie eine Funktion (mit Input- und Output) aufgerufen werden, werden Transaktionen nicht unterstützt.

Rollback

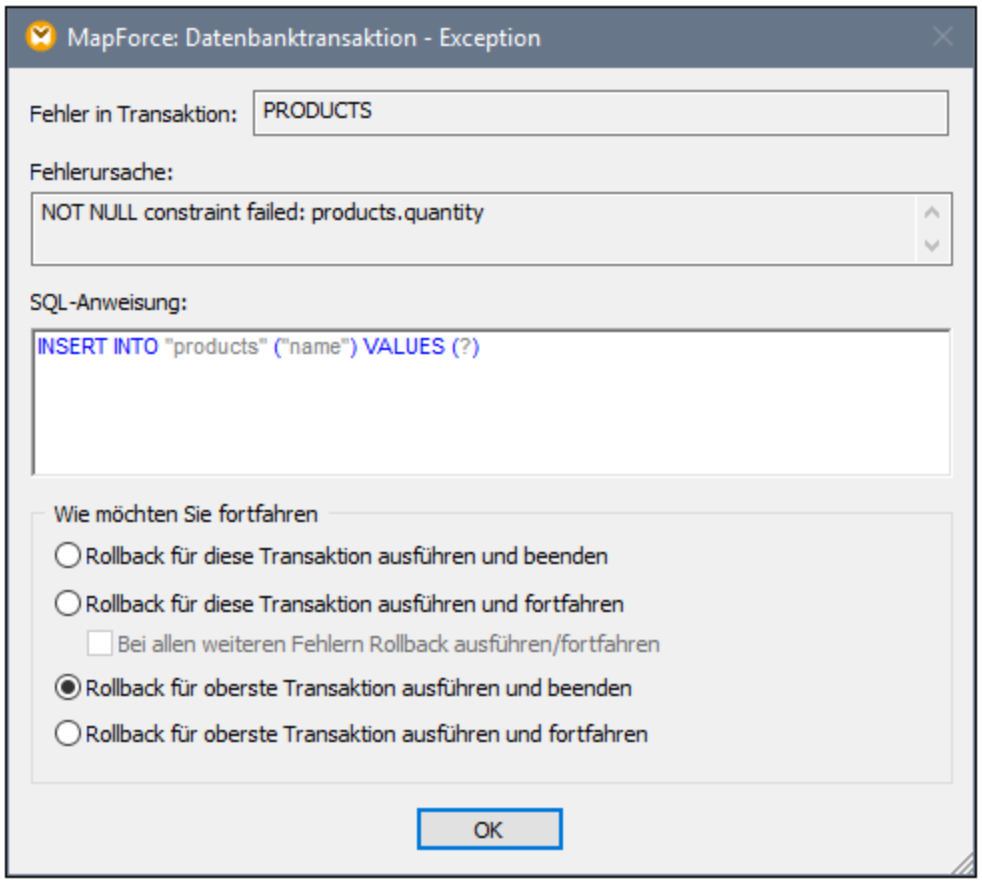
Um zu sehen, ob zur Mapping-Laufzeit Transaktionen ausgeführt werden, klicken Sie auf das Register **Ausgabe** und betrachten Sie den von MapForce generierten Pseudo-SQL-Code. In der Abbildung unten sehen Sie die Vorschau auf ein Mapping, in dem Transaktionen auf Komponenten- und Tabellenaktionsebene aktiviert sind. Wie Sie unten sehen, steht die gesamte Operation innerhalb einer Haupttransaktion (die durch die Kommentare `--begin transaction` und `--commit transaction` ausgewiesen ist). Außerdem werden für jede INSERT-Operation mehrere kleinere Transaktionen durchgeführt (diese sind durch den Befehl `SAVEPOINT` gekennzeichnet).

© 2016-2022 Altova GmbH



Wenn ein Mapping in MapForce Server oder über ein mit MapForce generiertes Programm ausgeführt wird, erfolgt der Rollback auf Basis der Optionen, die Sie, wie oben beschrieben, beim Erstellen des Mappings konfiguriert haben. Wenn Sie z.B. die Option **Rollback für oberste Transaktion und beenden** ausgewählt haben, werden bei einer fehlgeschlagenen INSERT-Operation alle Änderungen rückgängig gemacht, auch wenn einige Einfügungen erfolgreich durchgeführt werden konnten. Wenn Sie jedoch **Rollback für aktuelle Transaktion und fortfahren** ausgewählt haben, wird nur die fehlgeschlagene INSERT-Aktion bis zum nächstgelegenen Savepoint (Punkt der Speicherung) rückgängig gemacht, das Mapping wird fortgesetzt und es wird versucht weitere Datensätze einzufügen.

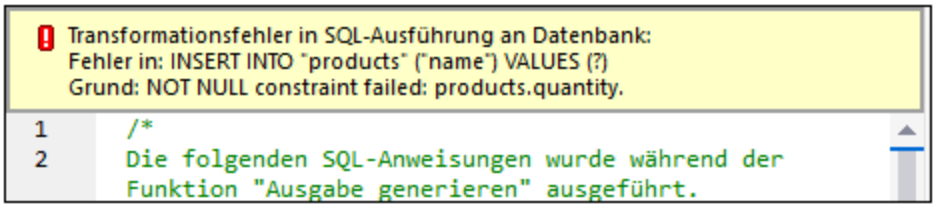
Wenn Sie ein Mapping, bei dem die Verwendung von Transaktionen aktiviert wurde, mit dem Menübefehl **Ausgabe | SQL-Script ausführen** direkt in MapForce ausführen, erscheint bei einer Datenbankausnahme ein Dialogfeld, in dem Sie darüber informiert werden, z.B.:



Sie können daraufhin festlegen, wie danach fortgefahren werden soll:

Rollback für diese Transaktion ausführen und beenden

- Die aktuelle Transaktion wird bereits mit Rollback rückgängig gemacht, wenn das Dialogfeld angezeigt wird.
- Am oberen Rand des Vorschaufensters und im Fenster "Meldungen" wird ein Fehler angezeigt.



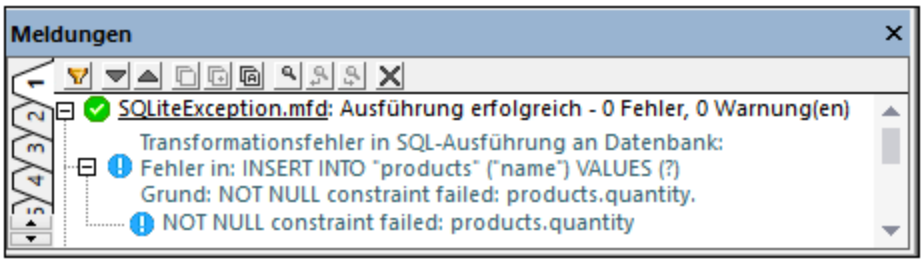
- Übergeordnete Transaktionen werden in die Datenbank übernommen.
- Die Mapping-Ausführung wird mit einem Fehlercode beendet.

Rollback für diese Transaktion ausführen und fortfahren

- Die aktuelle Transaktion wird bereits mit Rollback rückgängig gemacht, wenn das Dialogfeld angezeigt wird.

© 2016-2022 Altova GmbH

- Der Fehler wird als Warnung behandelt und wird im Fenster "Meldungen" angezeigt, sobald die Mapping-Ausführung fertig gestellt wurde.



- Übergeordnete Transaktionen sind nicht betroffen.
- Die Mapping-Ausführung wird mit dem nächsten Datensatz (wenn diese Transaktion sich nicht auf Komponentenebene befindet) oder mit der nächsten Komponente (nur MapForce Server) fortgesetzt.

Rollback für oberste Transaktion ausführen und beenden

- Die aktuelle Transaktion wird bereits mit Rollback rückgängig gemacht, wenn das Dialogfeld angezeigt wird.
- Am oberen Rand des Vorschaufensters und im Fenster "Meldungen" wird ein Fehler angezeigt.
- Übergeordnete Transaktionen werden mit Rollback rückgängig gemacht.
- Die Mapping-Ausführung wird mit einem Fehlercode beendet.

Anmerkung: Die Option **Rollback für oberste Transaktion ausführen und beenden** steht im Dialogfeld nur dann zur Verfügung, wenn es verschachtelte Transaktionen (savepoints) gibt.

Rollback für oberste Transaktion ausführen und fortfahren

- Die aktuelle Transaktion wird bereits mit Rollback rückgängig gemacht, wenn das Dialogfeld angezeigt wird.
- Der Fehler wird als Warnung behandelt und wird im Fenster "Meldungen" angezeigt, sobald die Mapping-Ausführung fertig gestellt wurde.
- Übergeordnete Transaktionen werden mit Rollback rückgängig gemacht.
- Die Mapping-Ausführung wird mit dem nächsten Datensatz (wenn diese Transaktion sich nicht auf Komponentenebene befindet) oder mit der nächsten Komponente (nur MapForce Server) fortgesetzt.

Anmerkung: Die Option **Rollback für oberste Transaktion ausführen und fortfahren** steht im Dialogfeld nur dann zur Verfügung, wenn es verschachtelte Transaktionen (savepoints) gibt.

4.2.10.1 Beispiel: Transaktions-Rollback

In diesem Beispiel werden verschiedene Möglichkeiten gezeigt, wie Sie bei Ausführung eines Mappings, bei dem eine Datenbank geändert wird, Datenbankausnahmen behandeln können. In der Datenbank in diesem Beispiel sind Benutzerdatensätze (Tabelle `users`) und Adressen (Tabelle `addresses`) gespeichert. Jeder Benutzer kann null, eine oder zwei Adressen haben (z.B. eine Wohnadresse und eine Arbeitsadresse). Jede Adresse in der Tabelle "`addresses`" enthält ein `user_id`-Feld, das auf das Feld `id` in der Tabelle `users` verweist. Die Tabelle `users` ist daher eine übergeordnete Tabelle der Tabelle `addresses`. Im Codefragment unten sehen Sie die SQL-Erstellungs-Skripts für beide Tabellen (die Syntax gilt für eine SQLite-Datenbank):

```
CREATE TABLE
       users (id INTEGER NOT NULL PRIMARY KEY,
       first_name TEXT NOT NULL,
       last_name TEXT NOT NULL,
       email TEXT UNIQUE NOT NULL);

CREATE TABLE
       addresses (id INTEGER NOT NULL PRIMARY KEY,
       user_id INTEGER NOT NULL,
       is_shipping INTEGER,
       is_billing INTEGER,
       type TEXT NOT NULL,
       city TEXT NOT NULL,
       street TEXT NOT NULL,
       number INTEGER NOT NULL,
       FOREIGN KEY (user_id) REFERENCES users (id) ) ;
```

Quelltabellen

Es sollen nun alle Daten aus den Tabellen `users` und `addresses` in neue Tabellen, nämlich `new_users` und `new_addresses` kopiert werden. Diese Tabellen sind fast identisch mit den Tabellen `users` und `addresses`. Der einzige Unterschied besteht darin, dass in der Tabelle `new_addresses` für die Spalten `is_shipping` und `is_billing` keine Nullwerte zulässig sind, wie Sie im Codefragment unten sehen. Dies ist wichtig, da dies bedeutet, das bei Nullwerten Ausnahmen auftreten.

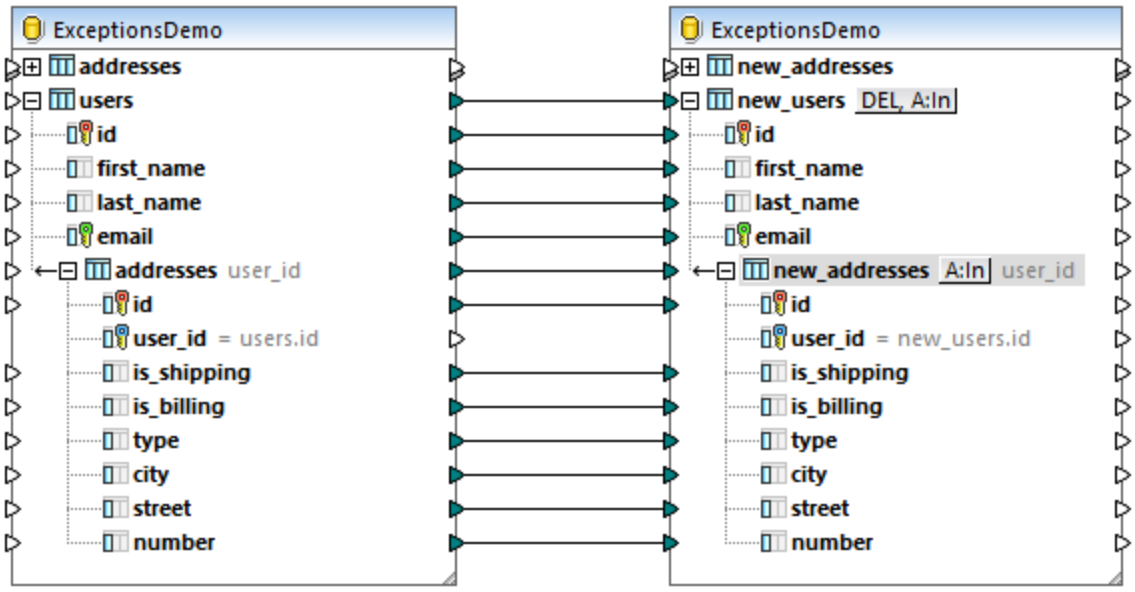
```
CREATE TABLE
       new_users (id INTEGER NOT NULL PRIMARY KEY,
       first_name TEXT NOT NULL,
       last_name TEXT NOT NULL,
       email TEXT UNIQUE NOT NULL);

CREATE TABLE
       new_addresses (id INTEGER NOT NULL PRIMARY KEY,
       user_id INTEGER NOT NULL,
       is_shipping INTEGER NOT NULL,
       is_billing INTEGER NOT NULL,
       type TEXT NOT NULL,
       city TEXT NOT NULL,
       street TEXT NOT NULL,
       number INTEGER NOT NULL,
       FOREIGN KEY (user_id) REFERENCES new_users (id) ) ;
```

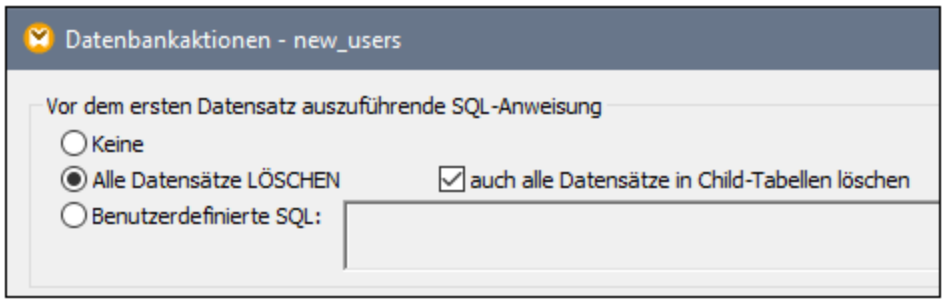
Zieltabellen

Unten sehen Sie das Mapping, in dem alle Daten aus den alten in neue Tabellen kopiert werden. Sie finden dieses Mapping unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2023\MapForceExamples\DatabaseExceptions.mfd.

© 2016-2022 Altova GmbH

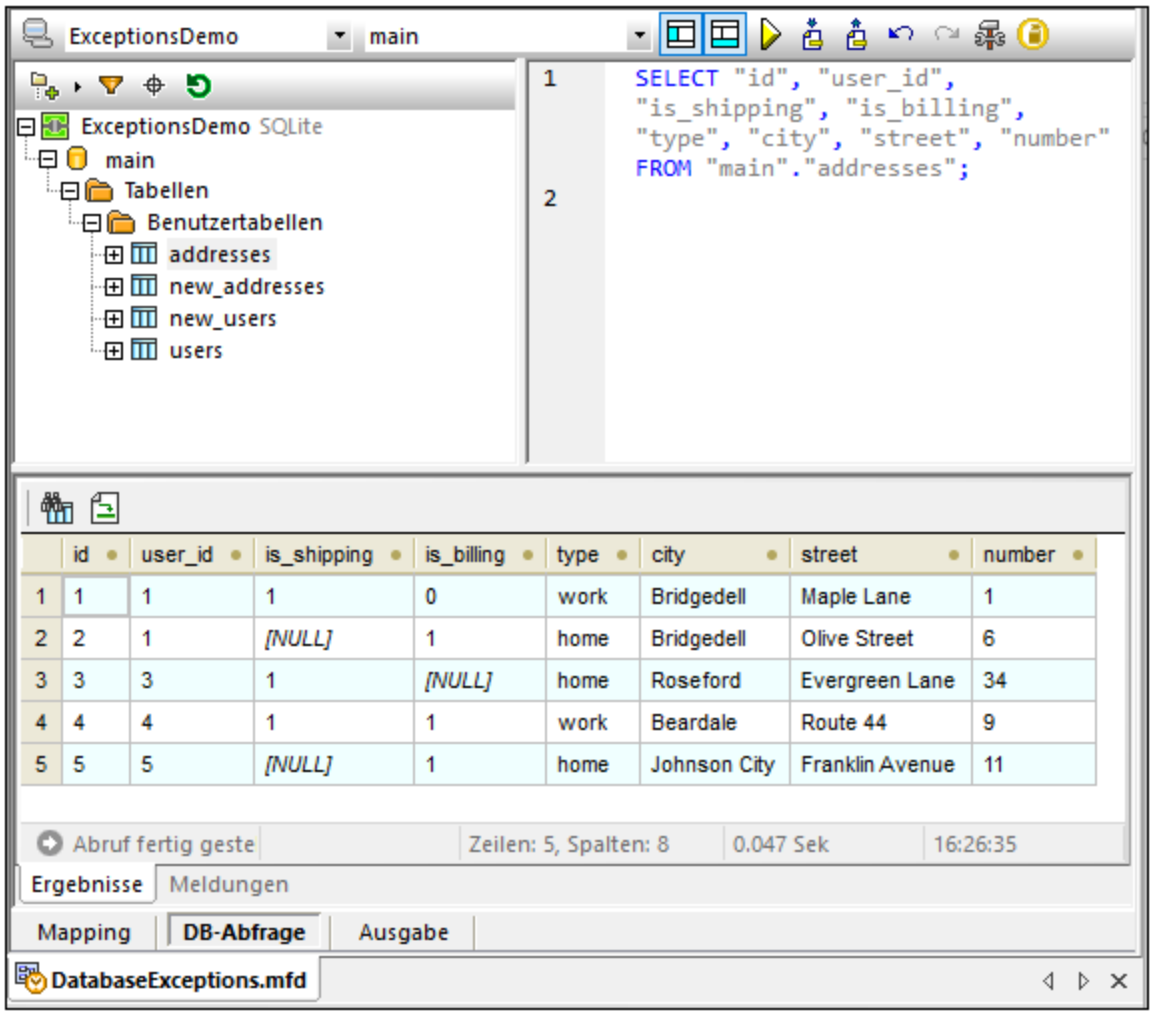


Wie Sie oben sehen, werden Daten in diesem Mapping unverändert aus einer Datenbankquellkomponente in eine Datenbankzielkomponente kopiert. Aus Gründen der Einfachheit werden keine weiteren Transformationen durchgeführt. Sowohl die Quell- als auch die Zielkomponente sind in diesem Beispiel Teil derselben Datenbank. Es unterscheiden sich nur die Quell- und Zieltabelle. Klicken Sie auf die Schaltfläche DEL, A:In, um zu sehen, dass laut Konfiguration vor jeder Einfügeaktion alle Datensätze gelöscht werden. Damit stellen Sie sicher, dass die Zieltabellen vor jeder Einfügung immer leer sind.



Wie bereits erwähnt, werden Ausnahmen auftreten, wenn die Quelltabelle `addresses` Nullwerte enthält. Sie können folgendermaßen überprüfen, ob die Quelltabelle Nullwerte enthält:

1. Klicken Sie auf das Register **DB-Abfrage**.
2. Klicken Sie mit der rechten Maustaste auf die Tabelle `addresses` und wählen Sie den Befehl **In neuem SQL-Editor anzeigen | SELECT**.
3. Klicken Sie auf **Abfrage ausführen** .



Wie Sie oben sehen, enthält die Quelltabelle `addresses` eine Reihe von `NULL` -Feldern. Sie haben verschiedene Möglichkeiten, die dadurch verursachten Ausnahmen bei der Ausführung des Mappings zu behandeln, z.B.:

- A) Rückgängigmachen aller Änderungen mit Rollback, wenn eine Ausnahme auftritt, d.h. es sollen keine Datensätze eingefügt werden, wenn die Quelldaten ungültige Datensätze enthalten.
- B) Überspringen der Datensätze, bei denen Ausnahmen auftreten, wobei gültige Datensätze weiterhin eingefügt werden.
- C) Wie Sie im Mapping oben sehen, fügt das Mapping Daten in zwei hierarchisch verknüpfte Tabellen ein, nämlich `new_users` und `new_addresses`. Sie können MapForce so konfigurieren, dass, wenn eine Adresse nicht eingefügt werden konnte, auch der übergeordnete Benutzerdatensatz nicht eingefügt werden soll.

Szenario A: Rückgängigmachung aller Änderungen mit Rollback bei Auftreten einer Ausnahme

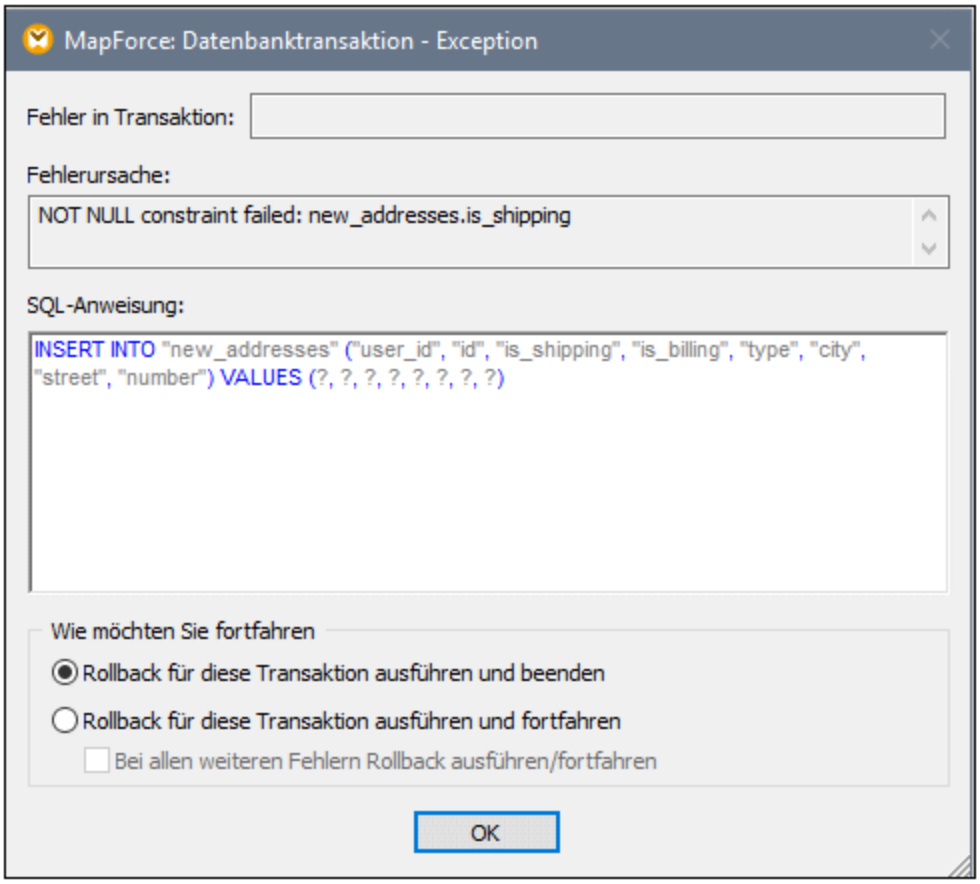
Aufgabenstellung: Wenn eine Ausnahme auftritt, sollen alle Änderungen an der Datenbank mit Rollback rückgängig gemacht werden. Konfigurieren Sie das Mapping dazu folgendermaßen:

© 2016-2022 Altova GmbH

1. Klicken Sie mit der rechten Maustaste auf die Titelleiste der Datenbankzielkomponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften**. Doppelklicken Sie alternativ dazu auf die Titelleiste. Daraufhin wird das Dialogfeld "Komponenteneinstellungen" für Datenbanken angezeigt.
2. Aktivieren Sie das Kontrollkästchen **Transaktionen verwenden** und wählen Sie **Rollback für oberste Transaktion und beenden**.

Wenn das Mapping, wie oben gezeigt, konfiguriert ist, wird die Transaktion auf der obersten Ebene der Datenbankkomponente bei Auftreten einer Ausnahme mit Rollback rückgängig gemacht. So führen Sie das Mapping mit diesen Einstellungen aus:

1. Klicken Sie auf das Register **Ausgabe**.
2. Klicken Sie im Menü **Ausgabe** auf **SQL-Script ausführen**. Daraufhin kommt es aufgrund eines Nullwerts in der Quelltabelle `addresses` zu einer Ausnahme. Daraufhin wird das Dialogfeld "Datenbanktransaktion - Exception" angezeigt.



3. Belassen Sie die vorausgewählte Standardoption unverändert und klicken Sie auf **OK**.

Dies führt zu folgendem Ergebnis:

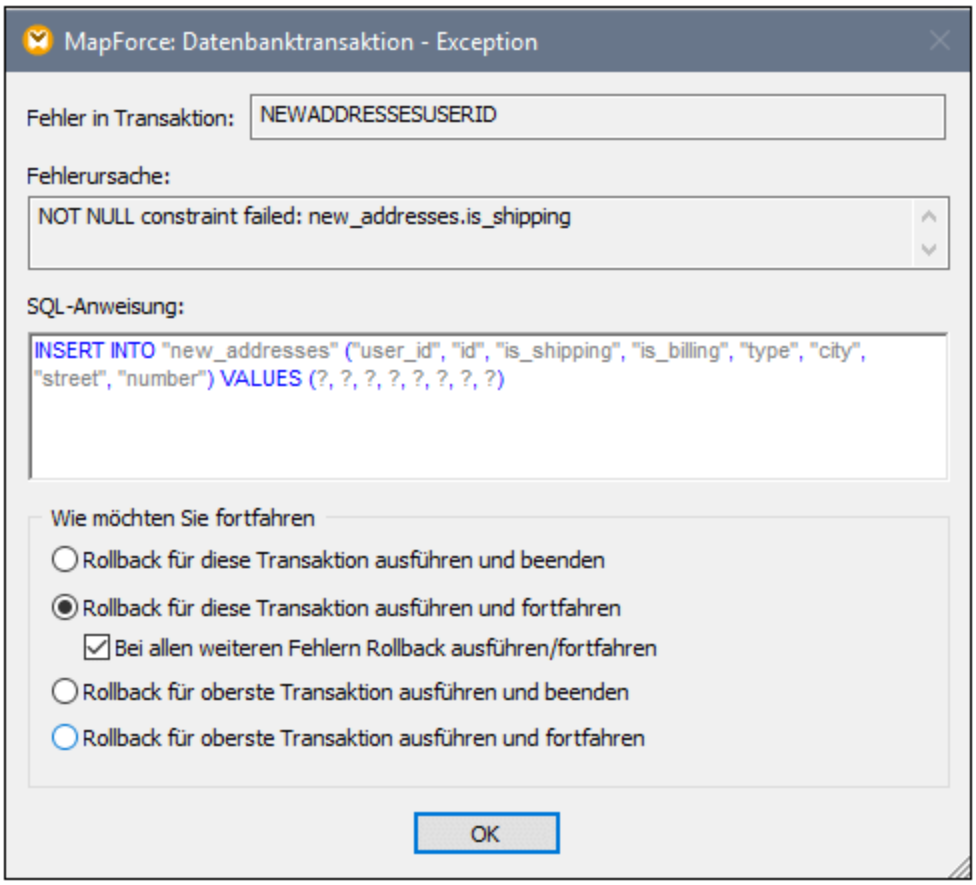
- Alle Änderungen werden mit Rollback rückgängig gemacht.
- Es werden keine Datensätze in die Tabelle `new_users` eingefügt.
- Es werden keine Datensätze in die Tabelle `new_addresses` eingefügt.

Szenario B) Rückgängigmachung der aktuellen Transaktion mit Rollback und Fortsetzung der Ausführung

Aufgabenstellung: Alle Datensätze, bei denen eine Ausnahme generiert wird, sollen übersprungen werden, gültige Datensätze sollen weiterhin eingefügt werden. Konfigurieren Sie das Mapping dazu folgendermaßen:

1. Klicken Sie auf die Schaltfläche DEL, A:In neben der Tabelle `new_users`, aktivieren Sie das Kontrollkästchen **Transaktionen verwenden** und wählen Sie **Rollback für aktuelle Transaktion und fortfahren.**
2. Klicken Sie auf die Schaltfläche A:In neben der Tabelle `new_addresses`, aktivieren Sie das Kontrollkästchen **Transaktionen verwenden** und wählen Sie **Rollback für aktuelle Transaktion und fortfahren.**

Wenn das Mapping mit diesen Einstellungen ausgeführt wird, wird das Dialogfeld "Datenbanktransaktion - Ausnahme" angezeigt. Aktivieren Sie die Optionen, wie unten gezeigt, um nur die aktuelle Transaktion mit Rollback rückgängig zu machen und das Mapping weiter auszuführen:



Dies führt zu folgendem Ergebnis:

- Alle fehlerhaften Transaktionen werden mit Rollback rückgängig gemacht.
- 5 von 5 Benutzern werden eingefügt (da es bei keinem user-Datensatz zu einer Datenbankausnahme kam)

© 2016-2022 Altova GmbH

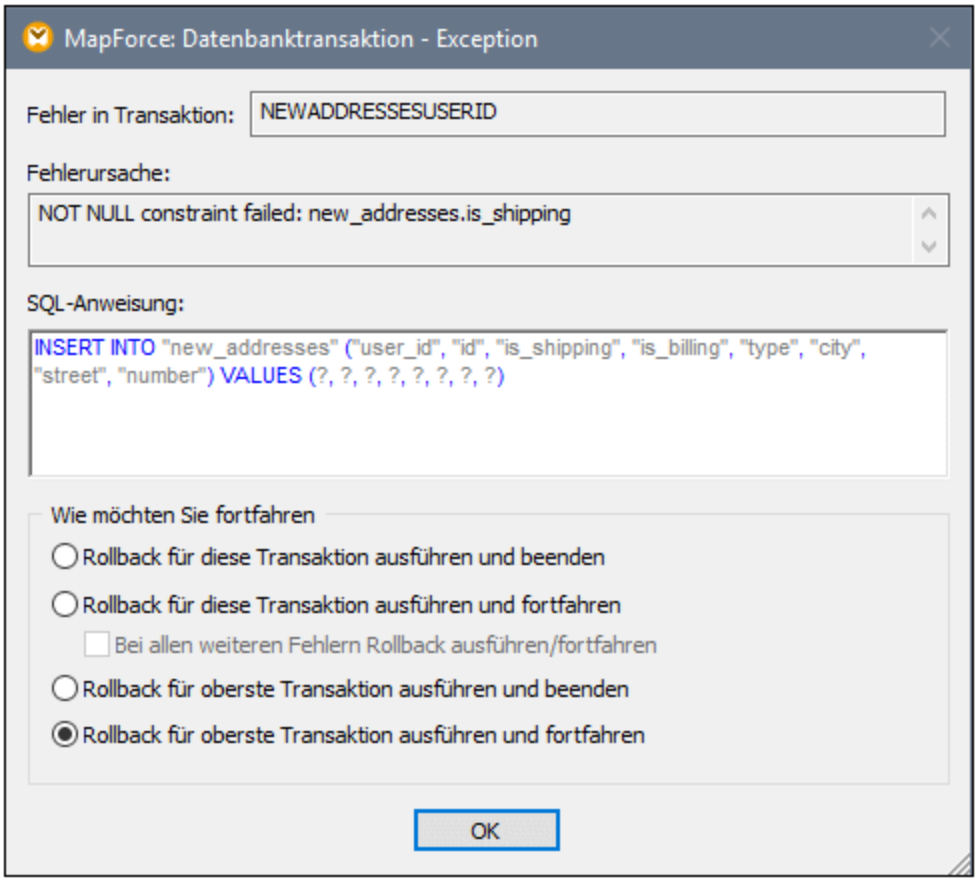
- 2 von 5 Adressen werden eingefügt (da 3 Adressen Nullwerte enthalten haben und es dadurch zu einer Ausnahme kam)

Szenario C) Rückgängigmachung der obersten Transaktion mit Rollback und Fortsetzung der Ausführung

Aufgabenstellung: Wenn eine Adresse nicht eingefügt werden kann, soll auch der übergeordnete Benutzerdatensatz nicht eingefügt werden. Konfigurieren Sie das Mapping dazu folgendermaßen:

1. Doppelklicken sie auf die Titelleiste der Datenbankkomponente und deaktivieren Sie das Kontrollkästchen **Transaktionen verwenden**.
2. Klicken Sie auf die Schaltfläche DEL, A:In neben der Tabelle `new_users`, aktivieren Sie das Kontrollkästchen **Transaktionen verwenden** und wählen Sie die Option **Rollback für aktuelle Transaktion und fortfahren.**
3. Klicken Sie auf die Schaltfläche A:In neben der Tabelle `new_addresses`, aktivieren Sie das Kontrollkästchen **Transaktionen verwenden** und wählen Sie die Option **Rollback für oberste Transaktion und fortfahren.**

Wenn Sie das Mapping mit dieser Konfiguration ausführen, wird das Dialogfeld "Datenbanktransaktion - Exception" angezeigt. Wählen Sie die Optionen wie unten gezeigt aus und wiederholen Sie diesen Vorgang, sooft Sie gefragt werden:



Sie erhalten das folgende Ergebnis:

- 2 Benutzer (Sharda Junker, Tobie Hughey) werden eingefügt.
- 1 Adresse wird eingefügt.

Erläuterung: Der Benutzer "Sharda Junker" ist der einzige Benutzer, der keine Adresse hat. Es gab bei diesem Datensatz keine Ausnahmen, daher wird er eingefügt. Der zweite Benutzer, "Tobie Hughey", wurde eingefügt, da er der einzige Benutzer ist, bei dem es auf Adressebene keine Ausnahmen gab. Alle anderen Benutzerdatensätze wurden nicht eingefügt, da sie mindestens eine Adresse hatten, bei der eine Ausnahme aufgetreten ist und bei denen die Transaktionen mit Rollback rückgängig gemacht wurden.

Es gibt zwei Adressen, die keine Nullen aufweisen und die daher keine Ausnahmen auslösen. Es sind dies die Adressen mit `id=1` und `id=4`. Die Einfügetransaktion für die erste Adresse wurde jedoch mit Rollback rückgängig gemacht, da die übergeordnete Benutzertransaktion rückgängig gemacht werden musste, daher wurde nur die Adresse mit `id=4` was eingefügt.

Beachten Sie, dass Sie dasselbe Ergebnis auch folgendermaßen erzielen:

4. Klicken Sie auf die Schaltfläche DEL, A:In neben der Tabelle `new_users`, aktivieren Sie das Kontrollkästchen **Transaktionen verwenden** und wählen Sie die Option **Rollback für aktuelle Transaktion und fortfahren.**
5. Klicken Sie auf die Schaltfläche A:In neben der Tabelle `new_addresses` und deaktivieren Sie das Kontrollkästchen **Transaktionen verwenden.**

Ausführen des Mappings mit MapForce Server

Wenn Sie MapForce Server lizenziert haben, können Sie das Mapping auch über die Befehlszeile auf einem Linux-, macOS- oder Windows-Rechner ausführen:

1. Kompilieren Sie das Mapping mit dem Menübefehl **Datei | Zu MapForce Server-Ausführungsdatei kompilieren** zu einer MapForce Server-Ausführungsdatei (.mfx), siehe auch Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442.
2. Kopieren Sie die .mfx-Datei und die Datenbank **ExceptionsDemo.sqlite** in ein Verzeichnis auf den Server-Rechner. Nennen wir es das "Arbeitsverzeichnis".
3. Ändern Sie die Befehlszeile in das Arbeitsverzeichnis und führen Sie MapForce mit dem unten stehenden Befehl aus.

```
mapforceserver run DatabaseExceptions.mfx.mfx
```

Anmerkungen:

- **mapforceserver** ist der betriebssystemspezifische Pfad zur ausführbaren MapForce Server-Datei. Ändern Sie den Pfad zur .mfx-Datei nach Bedarf. Auf einem Linux-Rechner lautet der Pfad z.B.: `/opt/Altova/MapForceServer2023/mapforceserver`.

Sie können Mappings bei der Server-Ausführung als API-Aufruf oder als FlowForce Server-Auftrag (entweder bei Bedarf oder in regelmäßigen Abständen) ausführen. Nähere Informationen dazu finden Sie unter Automatisierung mit MapForce Server 1434.

 © 2016-2022 Altova GmbH

4.2.11 Datenbankablaufverfolgung und Fehlerprotokollierung

Wenn bei einem Mapping Daten in eine Datenbank geschrieben werden, haben Sie die Möglichkeit, die Datenbankablaufverfolgung und Fehlerprotokollierung zu aktivieren. Die Ablaufverfolgung ist nützlich, wenn Sie aufzeichnen möchten, welche Änderungen während der Ausführung des Mappings vom Mapping an der Datenbank vorgenommen werden. Wenn die Ablaufverfolgung aktiviert ist, werden Ereignisse wie Einfügungen in die Datenbank oder Aktualisierungsaktionen in einer XML-Datei, die Sie später analysieren oder eventuell auf automatisierte Weise weiterverarbeiten können, protokolliert. Wenn Sie die Log-Datei in einem anderen Format als XML haben möchten, können Sie Daten daraus optional auf eine andere, von MapForce unterstützte Komponentenart (z.B. eine Textdatei oder sogar eine andere Datenbank) mappen.

Anmerkung: Damit die Ablaufverfolgung und Fehlerprotokollierung von einem Mapping unterstützt wird, muss als Mapping-Sprache **Built-in** ausgewählt sein. Die Datenbankablaufverfolgung und Fehlerprotokollierung wird in C#, C++ oder Java nicht unterstützt.

Die Datenbankablaufverfolgung ist konfigurierbar. Sie können alle Meldungen oder nur Fehler aufzeichnen lassen oder die Ablaufverfolgung komplett deaktivieren. Zusätzlich dazu können Sie die Ablaufverfolgung auf folgenden Ebenen verwenden:

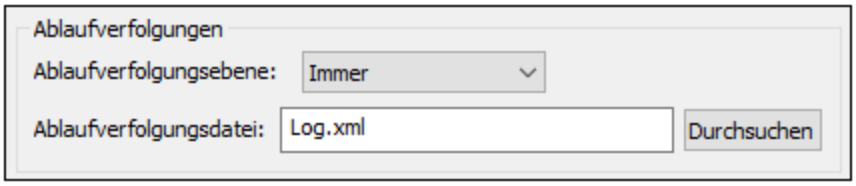
- Auf Ebene der Datenbankkomponente. Auf dieser Ebene schalten Sie die Ablaufverfolgung für diese Datenbankkomponente effektiv ein oder aus. Dies eignet sich für Mappings mit mehreren Zielkomponenten, bei denen die Ablaufverfolgung nur für einige davon benötigt wird.
- Auf Ebene von Tabellen oder gespeicherten Prozeduren. Sie können für eine Datenbankkomponente festlegen, ob die Ablaufverfolgung für jede Tabelle oder gespeicherte Prozedur aktiviert werden soll. Wenn die Ablaufverfolgung aktiviert ist, betrifft dies auf dieser Ebene Ereignisse im Zusammenhang mit Tabellenaktionen wie "Alle einfügen", "Aktualisieren, wenn", "Ignorieren, wenn", "Löschen, wenn". Im Fall von gespeicherten Prozeduren werden Ereignisse im Zusammenhang mit dem Aufruf der gespeicherten Prozedur aufgezeichnet.
- Auf Ebene von Datenbankfeldern. Standardmäßig ist die Ablaufverfolgung für alle Felder aktiviert, Sie können jedoch bestimmte Felder von der Ablaufverfolgung ausnehmen oder Sie können festlegen, dass diese nur bei einem Fehler in die Ablaufverfolgungsdatei aufgenommen werden sollen.

Beachten Sie, dass die drei Ebenen hierarchisch sind und die Ablaufverfolgungseinstellungen von oben nach unten weitergegeben werden. Wenn Sie die Ablaufverfolgung daher auf Ebene der Datenbankkomponente deaktivieren, kann sie nicht auf der Ebene von Tabellen oder gespeicherten Prozeduren aktiviert werden. So ist es auch nicht möglich, die Ablaufverfolgung auf der Ebene von Datenbankfeldern (Spalten) zu aktivieren, wenn Sie diese auf Tabellenebene deaktiviert haben. Dasselbe gilt, wenn Sie die Ebene der Ablaufverfolgung einschränken. Wenn Sie die Ablaufverfolgung auf Ebene der Datenbankkomponente auf Fehler eingeschränkt haben, können Sie keine uneingeschränkte Ablaufverfolgung auf Ebene von Tabellen oder gespeicherten Prozeduren durchführen.

Ablaufverfolgung auf Ebene der Datenbankkomponente

Um die Ablaufverfolgung auf Ebene der Datenbankkomponente zu aktivieren, gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf die Datenbankkomponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften** (Oder doppelklicken Sie alternativ dazu auf die Titelleiste der Datenbankkomponente). Daraufhin wird das Dialogfeld "Komponenteneinstellungen" für die Datenbank geöffnet.



2. Wählen Sie die gewünschte Ablaufverfolgungsebene aus (**Immer, Fehler, Deaktiviert**). Bei Auswahl von **Fehler** wird nur dann eine Ablaufverfolgungsdatei erstellt, wenn während der Mapping-Ausführung Fehler auftreten.
3. Geben Sie neben **Ablaufverfolgungsdatei** den Namen oder Pfad der XML-Datei ein, in die alle Ablaufverfolgungsinformationen geschrieben werden sollen (z.B. **Log.xml**). Wenn Sie anstelle eines vollständigen Pfads einen Dateinamen eingeben, wird die Ablaufverfolgungsdatei im selben Verzeichnis erstellt, in dem das Mapping ausgeführt wird.

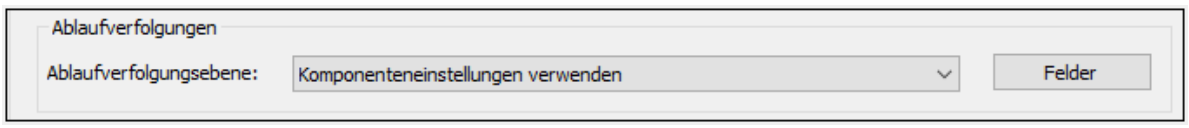
Nachdem Sie die Ablaufverfolgung aktiviert haben, ändert sich die Struktur der Datenbankkomponente im Mapping. Zusätzlich zur Datenbankstruktur wird eine neue Node-Hierarchie von Datenelementen angezeigt, bei denen das oberste Datenelement in der Hierarchie die Ablaufverfolgungsdatei ist. Nähere Informationen dazu finden Sie unter Struktur der Ablaufverfolgungsdatei 445.

Wenn Sie die Ablaufverfolgung auf Ebene der Datenbankkomponente aktivieren, wird sie automatisch für alle Tabellen oder gespeicherten Prozeduren in dieser Komponente, zu denen eine Mapping-Verbindung vorhanden ist, aktiviert. Falls nötig, können Sie die Ablaufverfolgung auf der Ebene von Tabellen oder gespeicherten Prozeduren noch genauer anpassen, wie weiter unten beschrieben. Beachten Sie, dass Tabellen oder gespeicherte Prozeduren, die in der Komponente angezeigt werden, aber nicht Teil des Mappings bilden (d.h. zu denen keine Mapping-Verbindungen bestehen), nicht in die Ablaufverfolgung eingeschlossen werden.

Ablaufverfolgung auf Ebene von gespeicherten Prozeduren

Um die Ablaufverfolgung auf Ebene von Tabellen oder gespeicherten Prozeduren zu aktivieren, gehen Sie folgendermaßen vor:

1. Stellen Sie sicher, dass auf Ebene der Datenbankkomponente als Ablaufverfolgungsebene entweder **Immer** oder **Fehler** ausgewählt ist (siehe oben).
2. Wählen Sie eine der folgenden Methoden:
   - Klicken Sie neben der Tabelle, für die Sie die Ablaufverfolgung aktivieren möchten, auf die Schaltfläche für die Tabellenaktion (z.B. ) oder
   - Wenn es sich um eine gespeicherte Prozedur handelt, klicken Sie auf die Schaltfläche **Kontextmenü anzeigen**  und wählen Sie anschließend im Kontextmenü den Befehl **Prozedureinstellungen** aus.

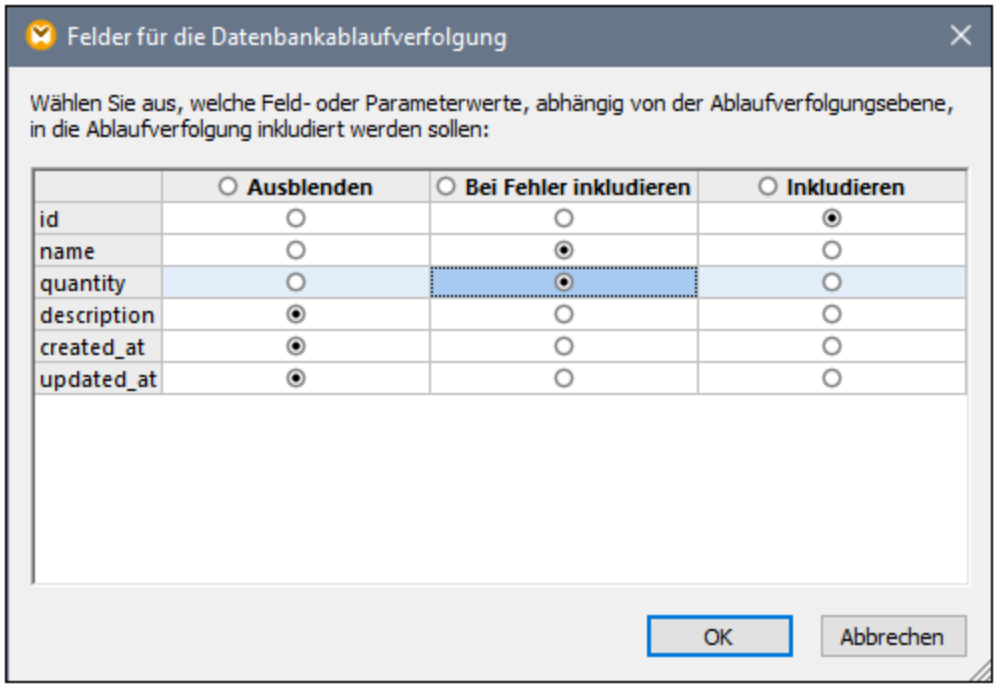


3. Wählen Sie die Ablaufverfolgungsebene aus. Bei Auswahl der Option **Komponenteneinstellungen verwenden** werden die auf Komponentenebene definierten Einstellungen übernommen. Mit der Option **Auf Fehler einschränken** wird die Ablaufverfolgung auf Fehlerereignisse eingeschränkt. Mit **Immer deaktiviert** wird für diese Tabelle oder gespeicherte Prozedur keine Ablaufverfolgung durchgeführt.

 © 2016-2022 Altova GmbH

Ablaufverfolgung auf Ebene von Datenbankfeldern

Wenn Sie die Ablaufverfolgung auf der Ebene von Datenbankkomponenten und Tabellen oder gespeicherten Prozeduren aktivieren, werden standardmäßig alle Felder (Datenbankspalten) in den Ablaufverfogungsbericht inkludiert. Sie können die Ablaufverfolgung optional auf bestimmte Datenbankfelder oder Parameter von gespeicherten Prozeduren einschränken. Klicken Sie dazu im Dialogfeld "Datenbankaktionen" oder "Einstellungen für Datenbankprozeduren" (siehe oben) auf die Schaltfläche **Felder**.

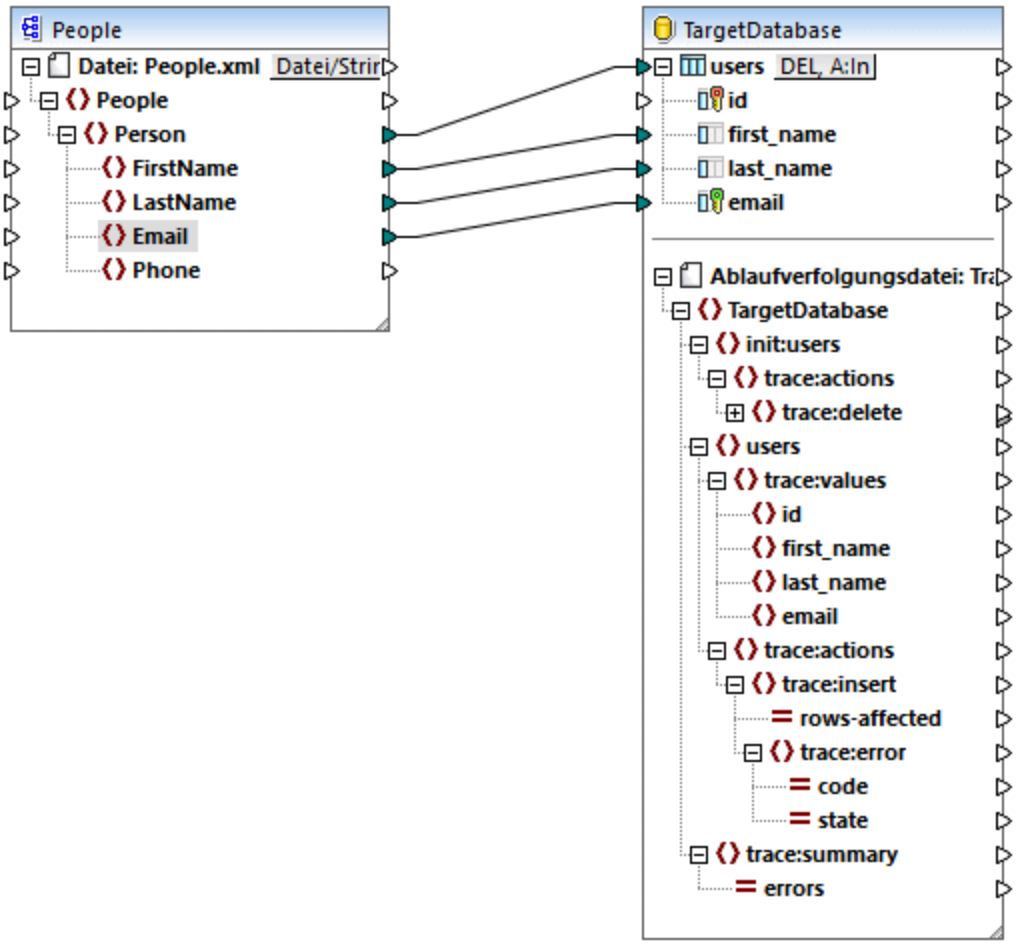


Eine in diesem Dialogfeld definierte Auswahl wirkt sich auf die Struktur der Ablaufverfolgungsdatei aus. So scheinen etwa Felder, die Sie ausgeblendet haben, in der Ablaufverfolgungsdatei nicht mehr auf. Dies ist unter Umständen nützlich, um die Größe der Log-Datei zu reduzieren.

Um mehrere Felder auf einmal auszublenden oder zu inkludieren, aktivieren Sie die entsprechende Option in der Überschrift.

4.2.11.1 Struktur der Ablaufverfolgungsdatei

Wenn die Ablaufverfolgung für eine Datenbankkomponente aktiviert wurde, wird in der unteren Hälfte der Komponente eine Ablaufverfolgungsstruktur angezeigt, z.B:



Wie Sie oben sehen, gibt der oberste Node der Ablaufverfolgungsstruktur den Namen der Ablaufverfolgungsdatei, die bei Ausführung des Mappings geschrieben wird, an. Der Rest der Ablaufverfolgungsstruktur wird nach der Struktur der Datenbanktabellen oder gespeicherten Prozeduren, die Teil des Mappings bilden, modelliert. In diesem Beispiel ist der Name der Datenbank "TargetDatabase", daher ist in der Ablaufverfolgungsstruktur ein ähnliches Datenelement vorhanden.

Das Child-Datenelement von "TargetDatabase" ist **users.** Es entspricht der Datenbanktabelle "users". Die Datenbankkomponente in diesem Beispiel hat nur eine Tabellen, nämlich "users". Bei einer Komponente mit vielen Tabellen enthält die Ablaufverfolgungsstruktur alle in der Komponente vorhandenen Tabellen. Wenn Sie Tabellen oder gespeicherte Strukturen zur Datenbankkomponente hinzufügen oder daraus entfernen, wird die Ablaufverfolgungsstruktur entsprechend angepasst. Eine Anleitung dazu finden Sie unter Hinzufügen, Bearbeiten und Entfernen von Datenbankobjekten 260.

In diesem Beispiel wird auch eine **init:users**-Struktur angezeigt, da das Mapping so konfiguriert ist, dass noch vor der Einfüge-, Aktualisierungs- oder einer anderen Aktion eine Anfangsaktion ausgeführt wird. So ist die **init**-Struktur z.B. dann vorhanden, wenn laut Konfiguration vor dem Einfügen neuer Datensätze alle vorhandenen Datensätze gelöscht werden. Die Ablaufverfolgungsinformationen über diese Anfangsaktion (wie z.B. Löschen oder eine benutzerdefinierte SQL-Abfrage) und etwaige Fehler, die zu diesem Zeitpunkt auftreten, werden gesammelt.

 © 2016-2022 Altova GmbH

In der **trace:values**-Struktur werden alle Spalten der Datenbanktabelle (in diesem Fall die Spalten der Tabelle "users") angezeigt. Bei gespeicherten Prozeduren werden in dieser Struktur die Parameter der gespeicherten Prozedur angezeigt. Standardmäßig ist die Ablaufverfolgung für alle Spalten aktiviert, Sie können jedoch, wie zuvor beschrieben, einstellen, für welche Spalten die Ablaufverfolgung durchgeführt werden soll.

Die **trace:actions**-Datenelemente geben an, welche Aktionen aktuell für die Datenbanktabelle definiert sind. In diesem Beispiel ist eine Datenbank aktualisieren-Aktion 

definiert, daher steht das Datenelement **trace:update** zur Verfügung. Wenn Sie die Komponente ändern und stattdessen eine Datenbankeinfügeaktion 

definieren, ändert sich die Ablaufverfolgungsstruktur in **trace:insert**, usw. für andere Aktionsarten. Für gespeicherte Prozeduren wird stattdessen ein **trace:execute**-Aktion angezeigt.

Jede Ablaufverfolgungsaktion hat ein **rows-affected**-Attribut, das definiert, wie viele Zeilen von der jeweiligen Datenbankaktion betroffen sind.

Das Datenelement **trace:error** wird nur dann befüllt, wenn bei der Mapping-Ausführung ein Datenbankfehler auftritt. Es hat zwei Attribute, **code** und **state**. Der Text der error- und state-Werte wird vom Datenbanktreiber bereitgestellt und unterscheidet sich daher je nach Datenbankart.

Das Datenelement **trace:summary** enthält ein **errors**-Attribut, das die Anzahl der aufgetretenen Fehler enthält. Wenn keine Fehler gefunden aufgetreten sind, ist der Attributwert **0**.

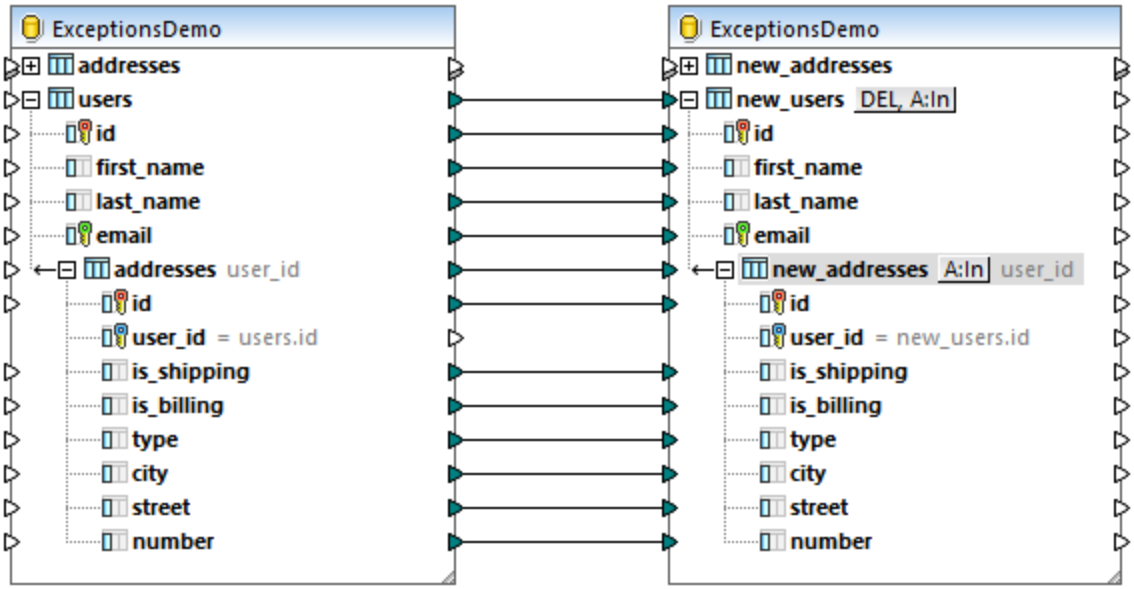
Anzeigen einer Vorschau auf Ablaufverfolgungsdateien

Wenn für eine Datenbankzielkomponente die Ablaufverfolgung aktivert ist, wird die Ablaufverfolgungsdatei, wenn Sie auf das Register **Ausgabe** klicken, auch in der Vorschau angezeigt. Dies ist hilfreich, wenn Sie wissen möchten, wie die Datei ungefähr aussehen wird. Beachten Sie, dass die tatsächlich generierte auf dem Ergebnis der Mapping-Ausführung basierende Ablaufverfolgungsdatei anders aussehen wird.

4.2.11.2 Beispiel: Protokollieren von Datenbankfehlern

In diesem Beispiel wird gezeigt, wie Sie die Datenbankablaufverfolgung und Fehlerprotokollierung für ein Mapping, in dem Daten in eine SQLite-Datenbank eingefügt werden, konfigurieren. Die Anleitung ist für alle von MapForce unterstützten Datenbanktypen sowie für andere Arten von Datenbankaktionen (wie aktualisieren, ignorieren, löschen usw.) ähnlich.

Sie finden das Demo-Mapping zu diesem Beispiel unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\DatabaseExceptions.mfd**. In diesem Mapping werden Daten aus zwei Tabellen einer Datenbank in ein neues Tabellenpaar derselben Datenbank eingefügt. Das Mapping wird unter Beispiel: Transaktions-Rollback 435 näher beschrieben. Beachten Sie, dass einige der Datensätze in der Datenbank absichtlich auf Null gesetzt sind, damit Datenbankausnahmen generiert werden.



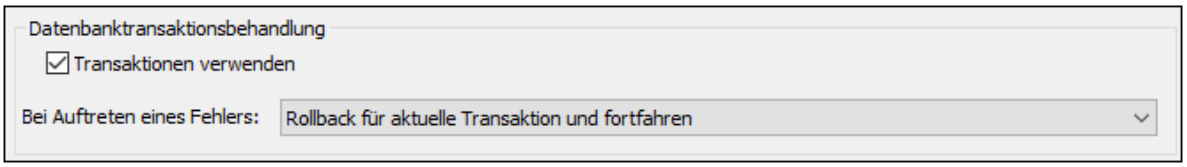
DatabaseExceptions.mfd

Aufgabe in diesem Beispiel ist es, die Ablaufverfolgung für die Datenbankzielkomponente zu aktivieren, um festzustellen, wie viele Fehler generiert wurden und wo genau die Fehler aufgetreten sind. In der schrittweisen Anleitung unten wird beschrieben, wie Sie dies erreichen.

Schritt 1: Aktivieren des Transaktions-Rollback

In diesem Demo-Mapping wird der Transaktions-Rollback laut Konfiguration auf Ebene der Datenbankkomponente verwendet, d.h. wenn die erste Ausnahme auftritt, wird die Mapping-Ausführung beendet und alle Daten werden in den Originalzustand zurückversetzt. Wir wollen zuerst die Transaktions-Rollback-Optionen ändern, sodass die Mapping-Ausführung auch bei Auftreten von Ausnahmen bei einigen Datensätzen fortgesetzt wird:

1. Klicken Sie neben der Tabelle `new_users` auf die Schaltfläche DEL, A:In, aktivieren Sie das Kontrollkästchen **Transaktionen verwenden** und wählen Sie **Rollback für aktuelle Transaktion und fortfahren**.



2. Klicken Sie neben der Tabelle `new_addresses` auf die Schaltfläche A:In, aktivieren Sie das Kontrollkästchen **Transaktionen verwenden** und wählen Sie **Rollback für aktuelle Transaktion und fortfahren**.

Sie könnten den Transaktions-Rollback komplett deaktivieren. In diesem Fall würde die Mapping-Ausführung jedoch beim ersten Fehler beendet und die Änderungen würden *nicht* mit Rollback rückgängig gemacht, daher wird dies nicht empfohlen. In der Ablaufverfolgung und Protokollierung würde somit nur die Ausführung bis zur

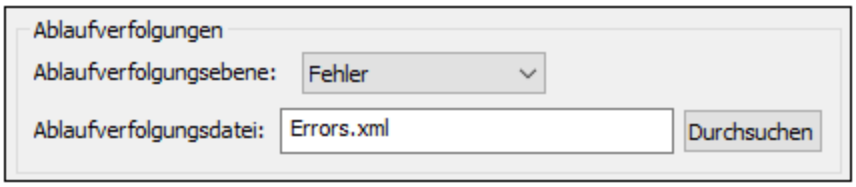
© 2016-2022 Altova GmbH

ersten Ausnahme berücksichtigt. Nähere Informationen zu Transaktions-Rollbacks finden Sie unter Behandlung von Datenbankausnahmen 429.

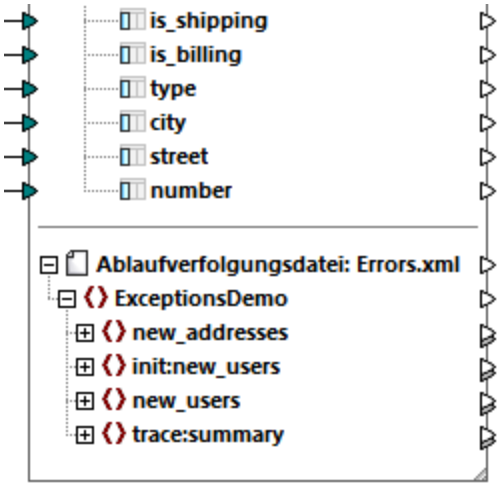
Schritt 2: Konfigurieren der Fehlerprotokollierung

Um die Ablaufverfolgung zu aktivieren, gehen Sie folgendermaßen vor:

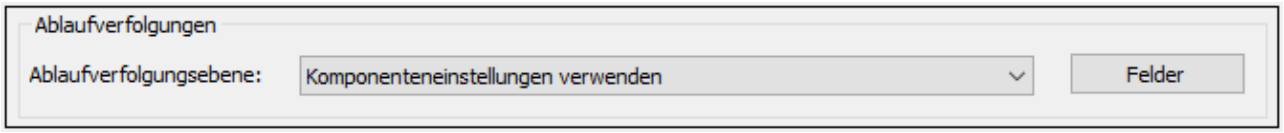
1. Klicken Sie mit der rechten Maustaste auf die Titelleiste der Datenbankkomponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften. (**Doppelklicken Sie alternativ dazu auf die Titelleiste der Datenbankkomponente). Daraufhin wird das Dialogfeld "Komponenteneinstellungen" für Datenbanken angezeigt.
2. Wählen Sie als Ablaufverfolgungsebene **Fehler** aus.
3. Geben Sie neben **Ablaufverfolgungsdatei** den Namen der Ablaufverfolgungsdatei ein (in diesem Beispiel **Errors.xml**). Geben Sie keinen absoluten Pfad ein, wenn Sie vorhaben, das Mapping mit MapForce Server auf anderen Rechnern oder Betriebssystemen auszuführen.



Die Struktur der Datenbankzielkomponente hat sich nun geändert und es wird ein zusätzlicher Bereich mit der Struktur der Ablaufverfolgungsdatei angezeigt:



Sie haben bis jetzt die Ablaufverfolgung auf Ebene der Datenbankkomponente aktiviert. Folglich werden die Ablaufverfolgungseinstellungen von allen Tabellenaktionen in dieser Datenbankkomponente übernommen. In diesem Beispiel werden die Einstellungen von den Einfügeaktionen der beiden Tabellen `new_users` und `new_addresses` geerbt. Wenn Sie für jede dieser Tabellen auf die Aktionsschaltfläche A:In klicken, sehen Sie die Einstellungen:



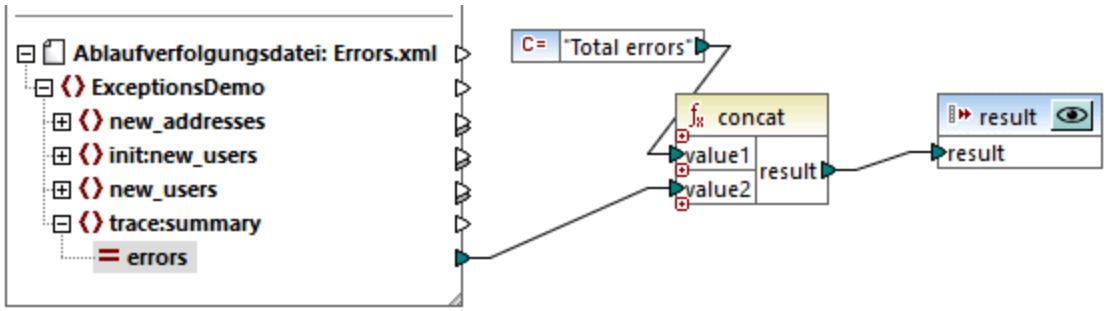
Klicken Sie optional auf die Schaltfläche **Felder,** um zu konfigurieren, welche Felder in der Ablaufverfolgungsdatei berücksichtigt oder daraus ausgenommen werden sollen. In diesem Beispiel wurden alle Felder inkludiert.

Schritt 3: Konfigurieren der Mapping-Ausgabe

In diesem Schritt wird gezeigt, wie Sie de Werte aus der Ablaufverfolgungsdatei auf eine andere Art von Komponente (in diesem Fall eine einfache Ausgabekomponente) mappen. Beachten Sie, dass dieser Schritt optional ist. Auch wenn Sie die Ablaufverfolgungsdatei nicht auf eine andere Komponente mappen, wird eine Fehlerprotokolldatei erzeugt. In diesem Beispiel möchten wir das Mapping jedoch explizit so konfigurieren, dass in der Ausgabe die Gesamtanzahl der gefundenen Fehler zurückgegeben wird. Gehen Sie dazu folgendermaßen vor:

1. Fügen Sie eine einfache Ausgabekomponente hinzu. Klicken Sie dazu auf den Menübefehl **Einfügen | Output-Komponente einfügen**. Nähere Informationen zu einfachen Input-Komponenten finden Sie unter Rückgabe von String-Werten aus einem Mapping 878.
2. Fügen Sie eine Konstante hinzu. Klicken Sie dazu auf den Menübefehl **Einfügen | Konstante**. Zur näheren Beschreibung der Mapping-Ausgabe liefert die Konstante den statischen Text "Total errors: " (Fehlergesamtzahl).
3. Ziehen Sie die **concat**-Funktion aus dem Fenster "Bibliotheken" in das Mapping. Nähere Informationen zu vordefinierten Funktionen finden Sie unter Hinzufügen einer Funktion zum Mapping 943.

Die **concat**-Funktion verkettet, wie unten gezeigt, den Text der Konstante mit dem **errors**-Wert aus der Ablaufverfolgungsdatei.



Mit der obigen Konfiguration wird in der Ausgabe bei Ausführung des Mappings mit MapForce Server der Text: "Total errors: {value}", angezeigt, wobei **value** die Anzahl der zur Laufzeit aufgetretenen Fehler darstellt.

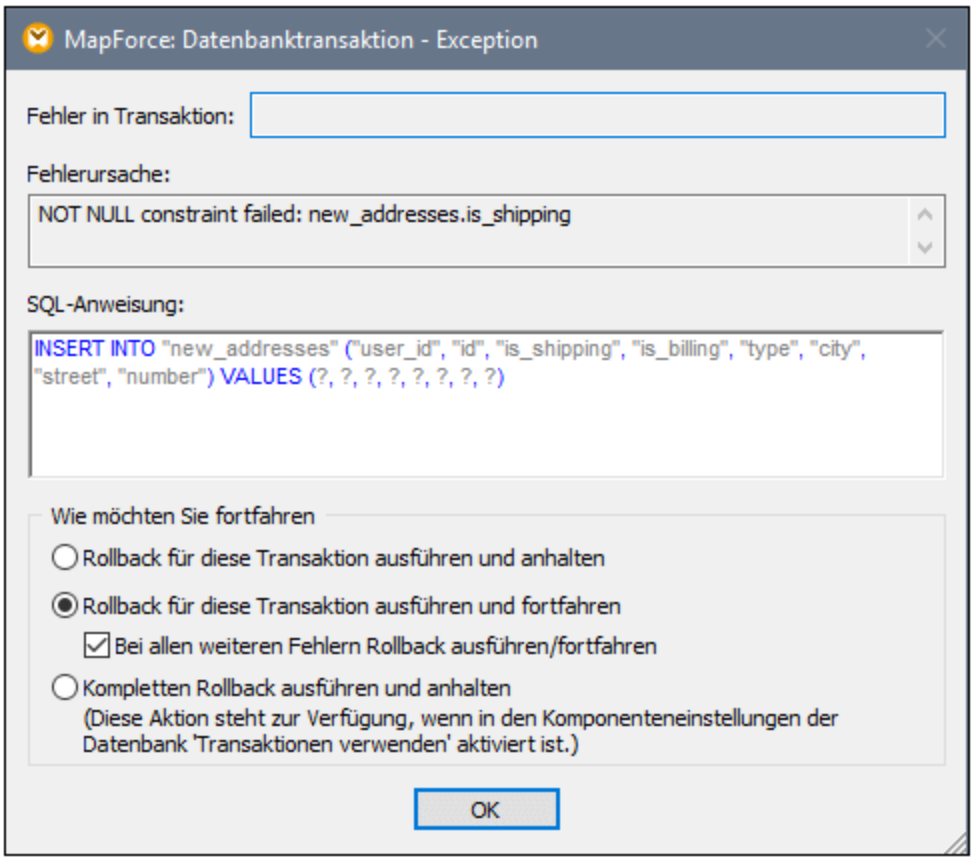
Eine Ausführung des Mappings zu diesem Zeitpunkt in MapForce über die Schaltfläche **Vorschau** der Ergebniskomponente (result) ist nicht möglich und führt zu einem Fehler, da zuerst eine Ablaufverfolgungsdatei vorhanden sein muss, da die einfache Ausgabekomponente Daten aus der Ablaufverfolgungsdatei ausliest. Im nächsten Schritt wird beschrieben, wie Sie die Ablaufverfolgungsdatei erzeugen.

© 2016-2022 Altova GmbH

Schritt 4: Ausführen des Mappings

Um dieses Mapping mit MapForce auszuführen, gehen Sie folgendermaßen vor:

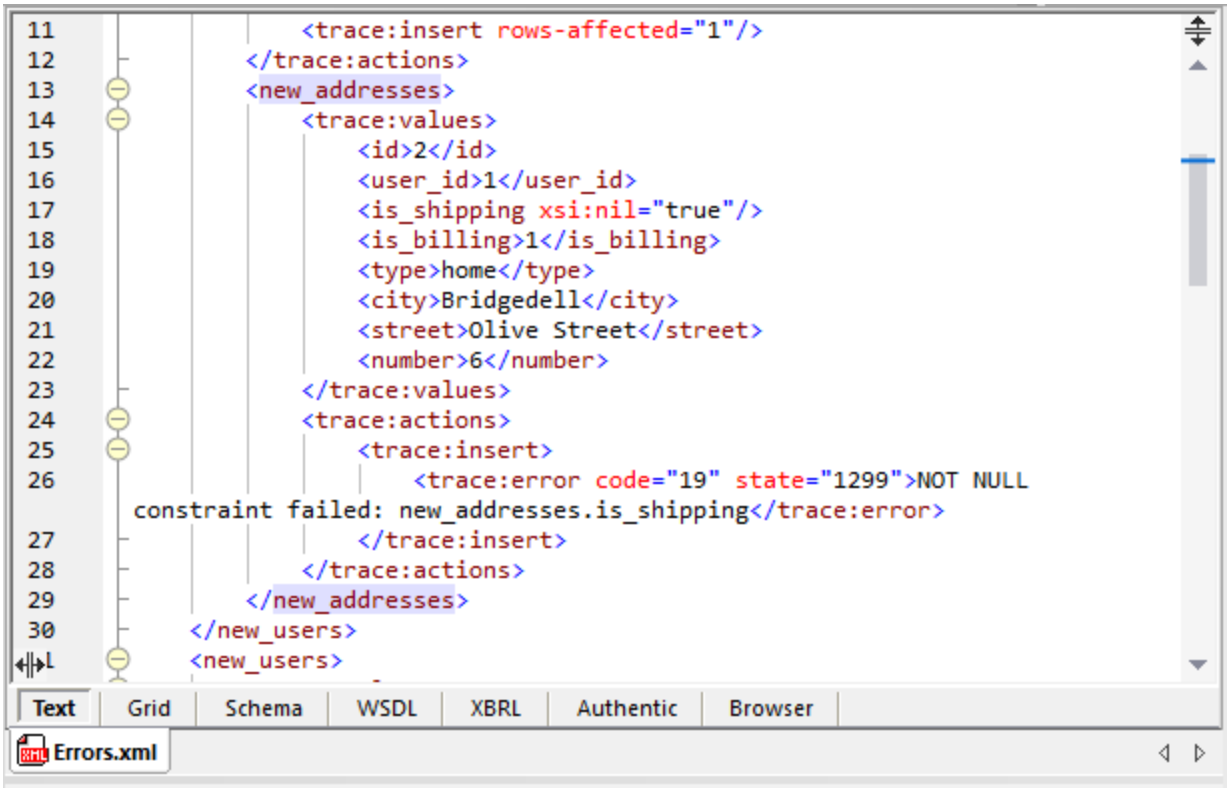
1. Klicken Sie in der Datenbankkomponente auf die Schaltfläche **Vorschau** .
2. Klicken Sie auf das Register **Ausgabe**.
3. Klicken Sie im Menü **Ausgabe** auf **SQL-Script ausführen**. Zu diesem Zeitpunkt tritt im Mapping die erste Ausnahme auf. Dies war beabsichtigt. Daraufhin wird das Dialogfeld "Datenbanktransaktion - Exception" angezeigt.



4. Wählen Sie die Rollback-Optionen, wie oben gezeigt, aus und klicken Sie auf **OK**.

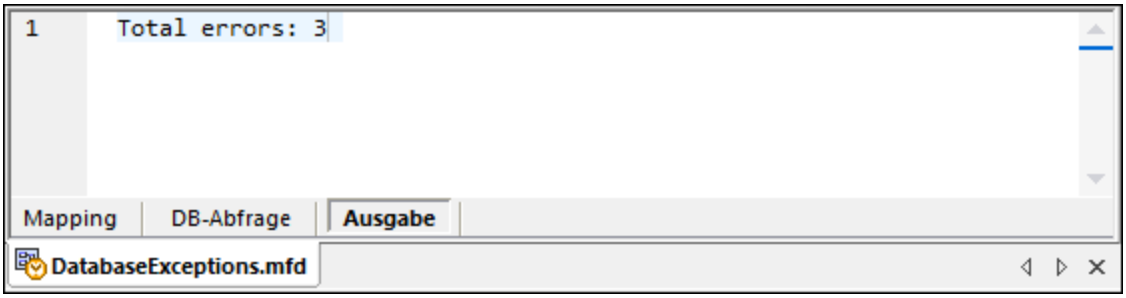
Anmerkung: Wenn ein Mapping von MapForce Server ausgeführt wird, erscheint kein Dialogfeld und die fehlerhaften Transaktionen werden automatisch entsprechend der in MapForce konfigurierten Rollback-Einstellungen rückgängig gemacht.

In diesem Beispiel werden alle fehlgeschlagenen Transaktionen nach Abschluss der Mapping-Ausführung mit Rollback rückgängig gemacht. Nur die erfolgreichen Transaktionen werden in die Datenbank übernommen. Die aufgetretenen Fehler werden in die Datei **Errors.xml,** von der Sie unten ein kleines Fragment sehen, geschrieben.



Jetzt, da die Log-Datei erzeugt wurde, wird bei einer Vorschau auf die Ausgabe die Gesamtanzahl der Fehler in der MapForce-Ausgabe angezeigt:

1. Klicken Sie zurück im Mapping auf die Schaltfläche **Vorschau** der "result"-Komponente.
2. Klicken Sie auf das Register **Ausgabe,** um das Ergebnis zu sehen.



In diesem Teil des Mappings werden Daten aus der zuvor erzeugten Log-Datei ausgelesen. Falls die Log-Datei nicht vorhanden ist, wird ein Fehler generiert.

© 2016-2022 Altova GmbH

4.2.12 NoSQL-Datenbanken

Die MapForce Enterprise Edition und die MapForce Server Advanced Edition unterstützen NoSQL-Datenbanken als Mapping-Quell- und Zielkomponenten. Derzeit werden nur zwei Dokument-Datenbanken unterstützt. MongoDB und CouchDB. Nähere Informationen über die Besonderheiten von NoSQL-Datenbanken können Sie in den folgenden Artikeln nachlesen: Was ist NoSQL? und NoSQL vs. SQL Databases. NoSQL-Datenbanken unterstützen keine Beziehungen[266] und benutzerdefinierten SQL-Anweisungen.

Dieser Abschnitt enthält die folgenden Kapitel:

- Informationen zu NoSQL-Datenbanken[453]
- NoSQL-Datenbankkonfiguration[454].

4.2.12.1 Informationen zu NoSQL-Datenbanken

In diesem Unterabschnitt werden die wichtigsten Punkte zu NoSQL-Datenbanken behandelt.

Primärschlüssel

Der Primärschlüssel ist immer `_id`.

JSON Schema

In MapForce wird die Struktur von NoSQL-Datenbankkomponenten mit Hilfe von JSON-Schema beschrieben. Beachten Sie, dass für das Datenmapping in MapForce eine Datenstruktur erforderlich ist. Anstelle von Feldern in einer Datenbanktabelle muss der Benutzer ein JSON-Schema einbetten, um eine mapbare Struktur zu erhalten. Als Schema kann ein benutzergeneriertes oder ein automatisch anhand des Datenbankinhalts generiertes Schema verwendet werden. Nähere Informationen dazu finden Sie unter Zuweisen von JSON-Schemas[456] MongoDB unterstützt die Speicherung eines Schemas zur Validierung von Dokumenten in der Datenbank.

Daten in JSON und BSON

In MongoDB werden Dokumente in BSON (Binary JSON) gespeichert. In CouchDB werden Daten in JSON gespeichert. Nähere Informationen über die Unterschiede finden Sie unter JSON und BSON.

BSON-Funktionsbibliothek

Das Fenster "Bibliotheken"[34] enthält eine neue BSON-Funktionsbibliothek[1119], mit deren Hilfe Sie BSON-Typen erstellen und bearbeiten können. Um die `bson`-Bibliothek zu finden, geben Sie in das Textfeld am unteren Rand des Fensters **Bibliotheken** `bson` ein.

Komponenteneinstellungen

In Dialogfeld **Komponenteneinstellungen** stehen für NoSQL-Datenbanken nur die folgenden Abschnitte des Dialogfelds zur Verfügung: **Datenbank**, **Login-Einstellungen**, **Timeout für die Ausführung der Anweisung** und **Datenbanktransaktionsbehandlung**. Die anderen Einstellungen sind nur für SQL-Datenbanken relevant. Nähere Informationen über die Komponenteneinstellungen finden Sie in diesem Kapitel[286].

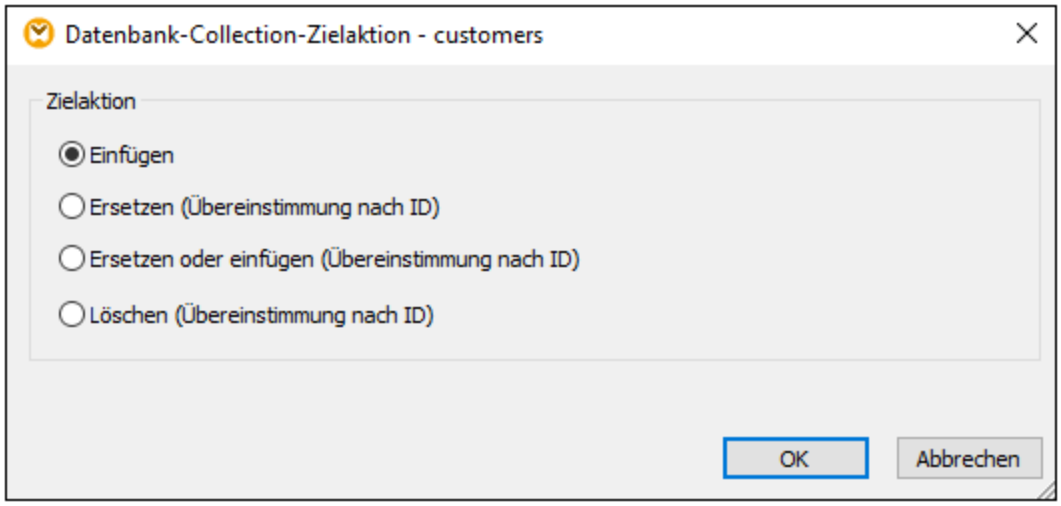
DB-Abfrage

Derzeit wird das DB-Abfrage-Fenster[39] nur eingeschränkt unterstützt. Sie sehen im Fenster **DB-Abfrage** die Liste der verfügbaren Collections in Ihrer Datenbank. Die Ausführung von Abfragen wird jedoch nicht unterstützt.

SQL/NoSQL-WHERE/ Order-Komponente
Die SQL/NoSQL-WHERE/ORDER[361]-Komponente Zum Filtern und Sortieren von Datenbankdaten wird eine JSON-basierte Syntax verwendet; Anstelle von `":"` wird vor einem Parameter im Abfragefilter das Zeichen `"@"` verwendet. Siehe dazu das Beispiel im Dialogfeld **SQL/NoSQL-WHERE/ORDER-Parameter**. Nähere Informationen über die Syntax zum Abfragen von Dokumenten in einer Collection finden Sie in der Dokumentation zu MongoDB.

Join-Komponente
Die Join-Komponente[344] funktioniert nur im Nicht-SQL-Modus.

Dialogfeld "Datenbank-Collection-Zielaktion"
NoSQL-Datenbanken unterstützen vier Aktionen für Ziel-Collections. Wenn als Zielkomponente eine NoSQL-Datenbank verwendet wird, wird neben dem Collection-Namen die Schaltfläche [In] (die **Aktionsschaltfläche**) angezeigt. Die **Aktionsschaltfläche** ändert sich je nach gewählter Aktion. Bei Klick auf diese Schaltfläche wird das Dialogfeld **Datenbank-Collection-Zielaktion** aufgerufen, in dem eine von vier Zielaktionen ausgewählt werden kann: **Einfügen**, **Ersetzen**, **Einfügen oder ersetzen** und **Löschen** (*siehe Abbildung unten*).



Die nachstehende Liste enthält eine kurze Beschreibung der Zielaktionen:

- Die Aktion **Einfügen**: Die gemappten Daten werden zusätzlich zu den bereits in der Zieldatenbank enthaltenen Daten (falls vorhanden) eingefügt.
- Mit der Aktion **Ersetzen** wird das gesamte Dokument in der Zieldatenbank ersetzt, falls seine `_id` mit derjenigen in den gemappten Daten übereinstimmt.
- Mit der Einfügen Aktion **Ersetzen oder einfügen** wird das gesamte Dokument in der Zieldatenbank ersetzt, falls seine `_id` mit derjenigen in den gemappten Daten übereinstimmt. Andernfalls wird die Aktion **Einfügen** ausgeführt.
- Mit der Aktion **Löschen** wird das gesamte Dokument in der Zieldatenbank gelöscht, falls seine `_id` mit derjenigen in den gemappten Daten übereinstimmt.

4.2.12.2 NoSQL-Datenbankkonfiguration

In der folgenden Anleitung erfahren Sie, wie Sie eine NoSQL-Datenbank als Mapping-Komponente hinzufügen und ihr ein Schema zuweisen.

© 2016-2022 Altova GmbH

Hinzufügen einer NoSQL-Datenbank

Bevor Sie eine Datenbank zum Mapping hinzufügen, stellen Sie sicher, dass Sie eine Transformationssprache ausgewählt haben, von der Datenbank-Mappings unterstützt werden. Für NoSQL-Datenbanken unterstützt MapForce nur Built-in[29] als Transformationssprache. Unten wird beschrieben, wie Sie Collections aus einer MongoDB-Datenbank zu einem Mapping hinzufügen. Unsere Beispieldatenbank hat den Namen **doc**.

Um eine **doc**-Datenbank zum Mapping hinzuzufügen, gehen Sie folgendermaßen vor:

1. Im ersten Schritt muss eine Quelldatenbank ausgewählt werden (in unserem Fall MongoDB). Gehen Sie zum Menü **Einfügen** und klicken Sie auf **Datenbank**. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche (**Datenbank einfügen**).
2. Wählen Sie im **Verbindungsassistenten MongoDB** aus und klicken Sie auf **Weiter**.
3. Im Dialogfeld **Datenbank auswählen** müssen Sie die folgenden Parameter angeben: **Host**, **Port**, **Datenbank**, **Benutzername** und **Passwort**. Geben Sie die erforderlichen Parameter ein und klicken Sie auf **Verbinden**. Nähere Informationen zum Herstellen einer Verbindung mit einer Datenbankquelle finden Sie unter MongoDB-Verbindung[197] und CouchDB-Verbindung[198].
4. Das Dialogfeld **Datenbankobjekte einfügen** wird geöffnet und Sie werden aufgefordert, Collections, die Sie in Ihrem Mapping verwenden möchten, auszuwählen (*siehe Abbildung unten*). Aktivieren Sie die entsprechenden Kästchen und klicken Sie auf **OK**.



5. Ihre NoSQL-Datenbankkomponente enthält nun eine Collection namens `customers` (*siehe Abbildung unten*).



Im nächsten Schritt wird der Collection nun ein JSON-Schema zugewiesen, das ihre Struktur definiert.

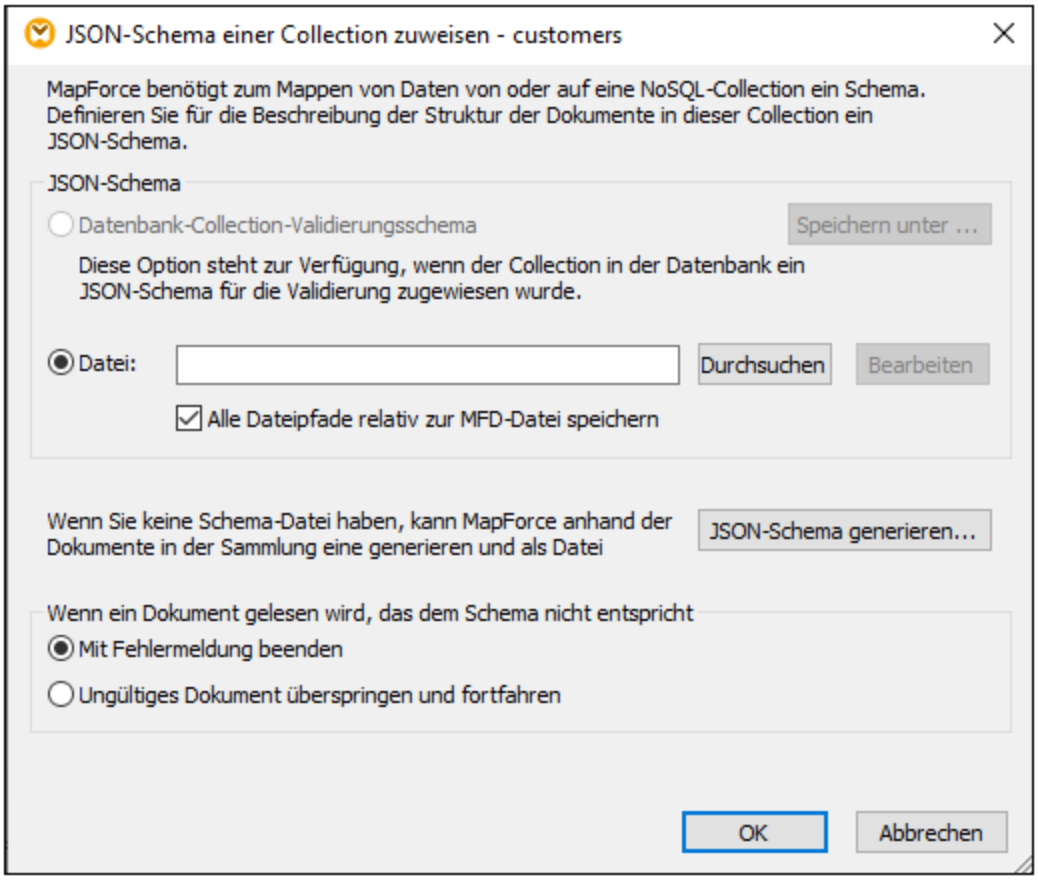
Anmerkung: Wenn Sie eine Verbindung zu einem MongoDB Cluster herstellen, müssen Sie eventuell Ihre IT-Abteilung kontaktieren, um Firewall-Zugriff auf die IP-Adressen Ihres Clusters und Port zu erhalten.

Zuweisen eines JSON-Schemas

Nachdem wir nun die entsprechenden Dokumente aus unserer **doc**-Collection ausgewählt haben, müssen wir der NoSQL-Komponente ein JSON-Schema zuweisen. Beachten Sie, dass die Zuweisung eines JSON-Schemas für MapForce eine Voraussetzung ist, damit das Mapping durchgeführt werden kann. Um der NoSQL-Komponente ein JSON-Schema zuzuweisen, gehen Sie vor, wie unten beschrieben.

1. Klicken Sie auf die Schaltfläche (**JSON-Schema zuweisen**) rechts vom Namen der Collection (`customers`). Daraufhin wird das Dialogfeld **JSON-Schema einer Collection zuweisen** (*siehe Abbildung unten*) aufgerufen.

© 2016-2022 Altova GmbH



2. Sie haben die Wahl zwischen drei Optionen, um ein JSON-Schema zuzuweisen: (i) Auswahl eines in Ihrer Datenbank zugewiesenen Schemas, (ii) Suche nach dem vorhandenen JSON-Schema auf Ihrem lokalen Rechner, (iii) Generierung eines neuen Schemas durch MapForce.

   i. Wenn Sie die erste Option wählen, können Sie das JSON-Schema auch aus der Datenbank exportieren (Klicken Sie auf **Speichern unter**).
   ii. Wenn Sie für Ihre Quelldatei bereits ein JSON-Schema haben, klicken Sie auf **Durchsuchen** und navigieren Sie zum gewünschten Schema. Wenn Sie sich für das Hochladen Ihrer Schema-Datei entscheiden, ist die Schaltfläche **Bearbeiten** aktiv. Wenn Sie darauf klicken, wird Ihre Schema-Datei in Altova XMLSpy geöffnet, falls das Programm auf Ihrem Rechner installiert ist.
   iii. Falls Sie kein Schema zur Verfügung haben, kann MapForce eines für Sie generieren. Klicken Sie in diesem Fall auf **JSON-Schema generieren**.

   Außerdem müssen Sie angeben, ob die Dokumentverarbeitung beendet oder fortgesetzt werden soll, wenn Dokumente vorhanden sind, die nicht dem Schema entsprechen (*siehe letzter Abschnitt des Dialogfelds oben*). Beachten Sie, dass diese beiden Optionen zur Mapping-Laufzeit angewendet werden.

3. Wenn Sie auf **JSON-Schema generieren** klicken, wird das Dialogfeld **JSON-Schema generieren** geöffnet. Lassen Sie die Textfelder **FILTER** und **SORT** leer, wenn Sie keine Filter- und Sortierkriterien definierten möchten. Klicken Sie anschließend auf **OK**. Beachten Sie, dass Sie auch angeben können, wie viele Dokumente der Parser analysieren soll. Mit der Option **ÜBERSPRINGEN** weisen Sie den Parser an, die ersten *N* Dokumente in der Collection zu überspringen. Mit der Option **LIMIT** weisen Sie den Parser an, nur die ersten *N* Dokumente in der Collection zu analysieren. Dem Schema der

Datenbankkomponente wird dann die Struktur der durch die Kombination dieser Kriterien definierten Dokumente zugrunde gelegt. Nähere Informationen zu **FILTER** und **SORT** finden Sie im Unterabschnitt weiter unten.

4. Wenn Sie auf **OK** klicken, müssen Sie Ihrer Schema-Datei einen Namen geben und den gewünschten Ordner auswählen, in dem sie gespeichert werden soll. Klicken Sie anschließend wieder auf **OK**.

Ihrer Datenbankkomponente hat nun eine Baumstruktur und ist bereit für das Mapping.

FILTER und SORT
Ohne Abfrageoptionen in den Textfeldern **FILTER** und **SORT** (des Dialogfelds **JSON-Schema generieren**) liest MapForce alle Dokumente in der Collection, um ein Schema zu generieren. Collections können allerdings Tausende von Dokumenten enthalten, was lange Verarbeitungszeiten verursachen kann. In diesem Fall empfiehlt es sich, eine Untergruppe von Dokumenten auszuwählen, die alle Strukturvarianten, die in Ihrem Mapping verarbeitet werden sollen, abdecken. Mit dem Begriff *Strukturvarianten* wird in diesem Kontext festgelegt, welche Eigenschaften im Dokument vorkommen und welche Typen diese Eigenschaften haben können.

Es ist unwahrscheinlich, dass in derselben Collection Dokumente vollkommen unterschiedlicher Struktur gespeichert sind, unter Umständen gibt es jedoch viele zusätzliche Felder, die in nur einigen Dokumenten vorkommen. Folgende Szenarien sind möglich:

- Wenn mit dem Filter nur Dokumente ausgewählt werden, die ein Feld, das in einigen anderen Dokumenten vorkommt, nicht haben, so wird das Schema dieses Feld nicht enthalten. Wenn später zur Laufzeit ein solches Feld gefunden wird, kommt es zu einem Validierungsfehler oder es gibt in der Zielkomponente keinen Node, auf den dieses Feld gemappt werden kann.
- Wenn alle Beispieldokumente eine bestimmte Eigenschaft enthalten, wird diese Eigenschaft im Schema als "obligatorisch" gekennzeichnet, wodurch ein Dokument (das eventuell später zu dieser Collection hinzugefügt wurde), das diese Eigenschaft nicht hat, anhand des Schemas als ungültig validiert wird.

In der Praxis ist ein anhand einer eingeschränkten Auswahl an Dokumenten generiertes Schema eventuell unvollständig und muss manuell bearbeitet werden, z.B. um zusätzliche Eigenschaften hinzuzufügen, einige Eigenschaften zu optionalen zu machen oder für einige Eigenschaften zusätzliche Datentypen zur Verfügung zu stellen.

FILTER-Beispiel
Im Unterabschnitt Hinzufügen einer NoSQL-Datenbank 455 haben wir eine NoSQL-Datenbankkomponente erstellt, die ein Dokument namens `customers` enthält. MapForce soll nun bei der Generierung eines neuen Schemas die Filterkriterien berücksichtigen. Es gibt in unserer `customers` Collection Dokumente zweier Arten (*siehe unten*).

doc.customers1

```
"CustomerID": "ALFKI",
"CompanyName": "Alfreds Futterkiste",
"ContactName": "Maria Anders",
"ContactTitle": "Sales Representative",
"Address": "Obere Str. 57",
"City": "Berlin",
"Region": "NULL",
"PostalCode": "12209",
"Country": "Germany",
```

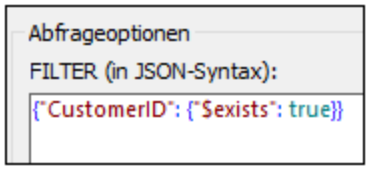
 © 2016-2022 Altova GmbH

```
    "Phone": "030-0074321",
    "Fax": "030-0076545"
```

doc.customers2

```
    "0": "BOTTM",
    "1": "Bottom-Dollar Markets",
    "2": "Elizabeth Lincoln",
    "3": "Accounting Manager",
    "4": "23 Tsawassen Blvd.",
    "5": "Tsawassen",
    "6": "BC",
    "7": "T2F 8M4",
    "8": "Canada",
    "9": "(604) 555-4729",
    "10": "(604) 555-3745"
```

Unserer Schema-Datei soll nun die Struktur mit den Namensfeldern (`doc.customers1`) zugrunde gelegt werden. In der Abbildung unten sehen Sie unser Filterkriterium, das folgendermaßen lautet: Wähle nur diejenigen Dokumente aus, in denen das Feld `CustomerID` vorhanden ist (*siehe Abbildung unten*). Der Parser geht die Liste aller Dokumente in der Collection durch und wählt nur diejenigen aus, die dieses Kriterium erfüllen. Nähere Informationen über die Syntax zum Abfragen von Dokumenten in einer Collection finden Sie in der Dokumentation zu MongoDB.



Nachdem Sie Ihre Daten gefiltert haben, können Sie diese auch sortieren, z.B. nach dem Feld `City`. Anschließend könnten Sie angeben, wie viele Dokumente aus der sortierten Auswahl bei der Schemagenerierung übersprungen (**ÜBERSPRINGEN**) und/oder berücksichtigt (**LIMIT**) werden sollen.

Achtung:

Wenn Sie Ihre Daten filtern und/oder sortieren, wird eine Untergruppe Ihrer Dokumente wie z.B. diejenigen in `doc.customers2` dem generierten Schema nicht entsprechen und Validierungsfehler verursachen. Um solche Fehler zu vermeiden, aktivieren Sie im Dialogfeld **JSON-Schema eine Collection zuweisen** (*siehe Abbildung oben*) die Option **Ungültige Dokumente überspringen und fortfahren**.

4.3 EDI

Altova Website: EDI-Mapping

EDI (Electronic Data Interchange) ist eine Gruppe von Standards für den elektronischen Datenaustausch zwischen Organisationen oder Unternehmen. MapForce unterstützt das Mappen von Daten von oder in EDI-Formate in jeder der folgenden Transformationssprachen: Built-In, C#, C++, Java. Die EDI-Daten können von oder auf jedes der von MapForce unterstützten Formate gemappt werden. Dazu gehören Datenbanken, Flat Files, XML-Dokumente, und andere. Inwieweit die Funktionalitäten für die einzelnen Sprachen und Formate unterstützt werden, finden Sie in den Anmerkungen zur Unterstützung 1982.

Die Tabelle unten enhält eine Übersicht zur Unterstützung von Standards im Zusammenhang mit EDI in MapForce.

Standard	Beschreibung
ASC X12	ASC X12 ist ein Industriestandard für den Dokumentenaustausch und MapForce unterstützt die folgenden Versionen: 3040, 3050, 3060, 3070, 4010, 4020, 4030, 4040, 4041, 4042, 4050, 4051, 4052, 4060, 5010, 5011, 5012, 5020, 5030, 5040, 5050, 6010, 6020, 6030 und 6040. **Als Standard-ASC X12-Version wird in MapForce 6040 verwendet.** ANSI X12-Komponenten haben "virtuelle" Nodes, in die, je nachdem, welche Einstellungen Sie im Dialogfeld "EDI-Validierungseinstellungen" (siehe Validierung von EDI-Komponenten 471) auswählen, EDI-Parser-Fehlerinformationen/Daten geschrieben werden. Ein X12 EDI 997 Functional Acknowledgement kann von jedem X12-Dokument generiert werden. Informationen zu ASC X12-Publikationen finden Sie unter http://www.x12.org/.
HIPAA X12	HIPAA basiert auf dem X12 EDI 5010-Standard, hat aber spezielle eigene Versionen, die nativ von MapForce 2011 Release 3 oder höher unterstützt werden. **Als Standard-HIPAA X12-Version wird in MapForce die Release A2 der HIPAA Implementierungs-Spezifikationen (TR3) verwendet.** Frühere Versionen können von der MapForce-Komponentenseite unter (https://www.altova.com/de/components_mapforce.html) heruntergeladen werden.
HL7	HL7 ist ein Industriestandard für den Austausch von Daten zwischen medizinischen Applikationen und steht für "Health Level Seven". MapForce unterstützt die Versionen 2.2 bis 2.6. **Als Standard-HL7-Version wird in MapForce 2.6 verwendet.** Auf der Seite MapForce-Komponenten der Altova-Website (https://www.altova.com/de/mapforce/download/libraries) steht ein eigener Installer für die zusätzlichen HL7 V2.2 - V2.5.1-XML-Schemas und Konfigurationsdateien zur Verfügung.

© 2016-2022 Altova GmbH

Standard	Beschreibung
	Die XML-basierte Version HL7 Version 3.x wird in MapForce 2023 auf Basis von XML-Schemakomponenten unterstützt. Das **MapForceExamples**-Projekt enthält eine Beispieldatei, in der eine HL7 V2.6 Datei auf eine HL7 V3 XML-Datei gemappt wurde (HL7V260_to_HL7V3.mfd). Nähere Informationen zu HL7 finden Sie unter https://www.hl7.org/.
IATA PADIS	PADIS (Passenger and Airport Data Interchange Standards) ist ein Industriestandard für den Austausch von Passagier- und Flughafendaten mittels EDI-Dokumenten. MapForce unterstützt die Versionen 00.1 bis 08.1. **Als Standard-PADIS-Version wird in MapForce 08.1 verwendet.** Nähere Informationen zu PADIS finden Sie unter http://www.iata.org/Pages/default.aspx.
NCPDP SCRIPT	SCRIPT ist ein Standard für den elektronischen Datenaustausch im amerikanischen Gesundheitssektor. Er wurde vom NCPDP (National Council for Prescription Drug Programs) entwickelt, siehe http://www.ncpdp.org. MapForce unterstützt das Mappen von Daten zwischen Dateien im SCRIPT-Format und allen anderen von MapForce unterstützten Formaten, einschließlich XML, CSV, Microsoft Excel, Datenbanken und anderen. Unterstützt werden die NCPDP SCRIPT-Versionen 8.1, 10.6, 10.10. **Die NCPDP SCRIPT-Standardversion in MapForce ist 10.10.**
SAP IDocs	SAP IDocs (intermediate documents)-Dokumente dienen zum Austausch von Geschäftsdaten zwischen SAP R/3-Applikationen und Nicht-SAP-Applikationen. Bei den Dokumenten handelt es sich um ein Zwischenformat, das zum Austausch von Daten zwischen verschiedenen Systemen dient. Nähere Informationen zu SAP IDocs finden Sie im SAP Help Portal.
SWIFT	SWIFT (Society for Worldwide Interbank Financial Telecommunication) ist ein internationaler Anbieter von Finanznachrichtendiensten. SWIFT MT (Message Text) ist ein im SWIFT-Netz verwendeter Nachrichtentyp. MapForce unterstützt das Mappen von SWIFT MT-Daten von und auf jedes andere von MapForce unterstützte Format,darunter XML, CSV, JSON, Datenbanken und andere. MapForce unterstützt die folgenden SWIFT MT-Release-Versionen: Versionen ab 2019. Beachten Sie, dass die installierte Standardversion SWIFT 2022 ist. Die Unterstützung für das Mappen von Daten von oder auf die Formate SWIFT MX / ISO 20022 ist implizit, da diese Formate XML-basiert sind.
TRADACOMS	TRADACOMS (Trading Data Communications) ist ein britischer Standard für den elektronischen Datenaustausch im Handel. In MapForce ist die TRADACOMS-Basisspezifikation aus dem von der Article Numbering Association (ANA) UK, jetzt bekannt als GS1 UK im Jänner 1993 veröffentlichten "TRADACOMS Manual of Standards for Electronic Data Interchange" implementiert. MapForce kann mit Hilfe von Konfigurationsdateien für andere TRADACOMS-Versionen

Standard	Beschreibung
	angepasst werden, um neue Nachrichtentypen, Datenelemente und Codewerte verarbeiten zu können.
UN/EDIFACT	UN/EDIFACT (United Nations rules for Electronic Data Interchange for Administration, Commerce and Transport) ist eine Gruppe von Standards für den elektronischen Austausch von Dokumenten. MapForce unterstützt die folgenden UN/EDIFACT-Verzeichnisse: D.93A oder später. Die neueste UN/EDIFACT-Version ist die in MapForce installierte Standardversion. In den Altova MapForce-Bibliotheken finden Sie separate Installer für alle früheren Versionen bis zurück zu Version D.93A, einschließlich der für interaktive EDI-Mappings erforderlichen Konfigurationsdateien.
ODETTE	ODETTE EDI ist eine Gruppe von Standards, die zum Austausch von Nachrichten in der Automobilbranche verwendet werden. Nähere Informationen zu ODETTE EDI-Nachrichtentypen finden Sie unter *Messaging Resources*.

Die standardmäßig mit MapForce installierten EDI-Konfigurationsversionen (in der Tabelle oben fett formatiert) ändern sich normalerweise mit jeder neuen Release. Wenn z.B. eine neuere Version des UN/EDIFACT-Standard erscheint, wird diese in der nächsten MapForce-Release zur Standardversion. Beachten Sie, dass die EDI-Standardversionen hauptsächlich für Test- und Demozwecke, nicht aber für die Produktion geeignet sind, da sie sich ständig ändern.

Um ein EDI-Mapping für die Produktion zu erstellen, stellen Sie sicher, dass Sie die genau die richtige erforderliche Version der EDI-Konfigurationsdateien von der Altova Webseite (https://www.altova.com/de/mapforce/download/libraries) heruntergeladen und installiert haben. Nach der Installation stehen die installierten EDI-Versionen zusätzlich zu den Standardversionen in MapForce zur Verfügung, siehe Hinzufügen von EDI-Komponenten zum Mapping 463.

MapForce kann für eine einzige zum Mapping-Bereich hinzugefügte EDI-Komponente mehrere Nachrichtentypen jeder beliebigen einzelnen EDI-Version verarbeiten. Dadurch kann MapForce z.B. Daten aus einer Datenaustauschdatei lesen, die mehrere Nachrichtentypen enthält, bzw. kann MapForce als Mapping-Ausgabe eine Datenaustauschdatei generieren, die mehrere Nachrichtentypen enthält. Sie können auch Daten aus mehreren Instanzdateien mit unterschiedlichen Nachrichtentypen verarbeiten. Beachten Sie, dass Operationen wie die oben beschriebene nur dann sinnvoll sind, wenn sie von der zugrunde liegenden Spezifikation unterstützt werden.

Falls Sie in Ihrem Unternehmen EDI-Spezifikationen verwenden, bei denen es sich um adaptierte Spezifikationen von bereits von MapForce unterstützten Spezifikationen handelt, so können Sie MapForce mit Hilfe von Konfigurationsdateien auch an Ihre benutzerdefinierte EDI-Spezifikation anpassen. So können Sie z.B. benutzerdefinierte Nachrichten, Segmente, Datenelemente, Codelisten usw. hinzufügen (oder die vorhandenen ändern) und auf diese Weise eine benutzerdefinierte EDI-Collection erstellen, die einem der bereits von MapForce unterstützten EDI-Verzeichnisse ähnlich ist. Nähere Informationen dazu finden Sie unter Anpassen der EDI-Struktur 539 und Anpassen der EDI-Validierung 573.

 © 2016-2022 Altova GmbH

4.3.1 EDI-Terminologie

In der folgenden Liste werden gebräuchliche Im Zusammenhang mit EDI-Standards beschrieben.

Message (Nachricht)

Ein Message stellt ein Geschäftsdokument dar, In ASC X12 werden Messages als "Transaction Sets" bezeichnet. Eine Message besteht aus mehreren Blöcken, den sogenannten Segmenten. MapForce unterstützt einen oder mehrere Message Types (Nachrichtentypen) in einer einzigen EDI-Komponente. (siehe Hinzufügen von EDI-Komponenten zum Mapping 463).

Segment

Ein einzelner "Datensatz" in einer Message bildet ein Segment. Segmente werden durch eine aus zwei oder drei Zeichen bestehende ID am Anfang des Segments identifiziert. Eine Gruppe von verwandten Elementen wird durch einen Segment-Tag (bzw. Segment ID in ASC X12) zusammengefasst. Segmente einer Transaktion können als Muss-Angabe (engl. mandatory) oder Kann-Angabe (engl. optional) definiert werden.

Datenelement

Ein Datenelement ist ein einzelnes Datenfeld innerhalb eines Segments und ist als Feld zu verstehen, da es normalerweise einen einzigen Datentyp enthält (z.B. einen Namen oder eine Adresse). Datenelemente lassen sich weiter in Composite Elements bestehend aus Component Elements oder Unterelementen unterteilen.

Trennzeichen

Datenelemente werden durch "Trennzeichen" voneinander getrennt. Bei Trennzeichen handelt es sich normalerweise um ASCII-Zeichen, die in Textfeldern selten vorkommen (z.B. Doppelpunkte, Sternchen, senkrechter Strich, Balken, kaufmännisches Und, usw). Die Trennzeichen variieren je nach EDI-Komponententyp. MapForce erkennt per Voreinstellung die Standardtrennzeichen für die einzelnen EDI-Versionen. Bei Bedarf können auch benutzerdefinierte Trennzeichen definiert werden (siehe EDI-Komponenteneinstellungen 466).

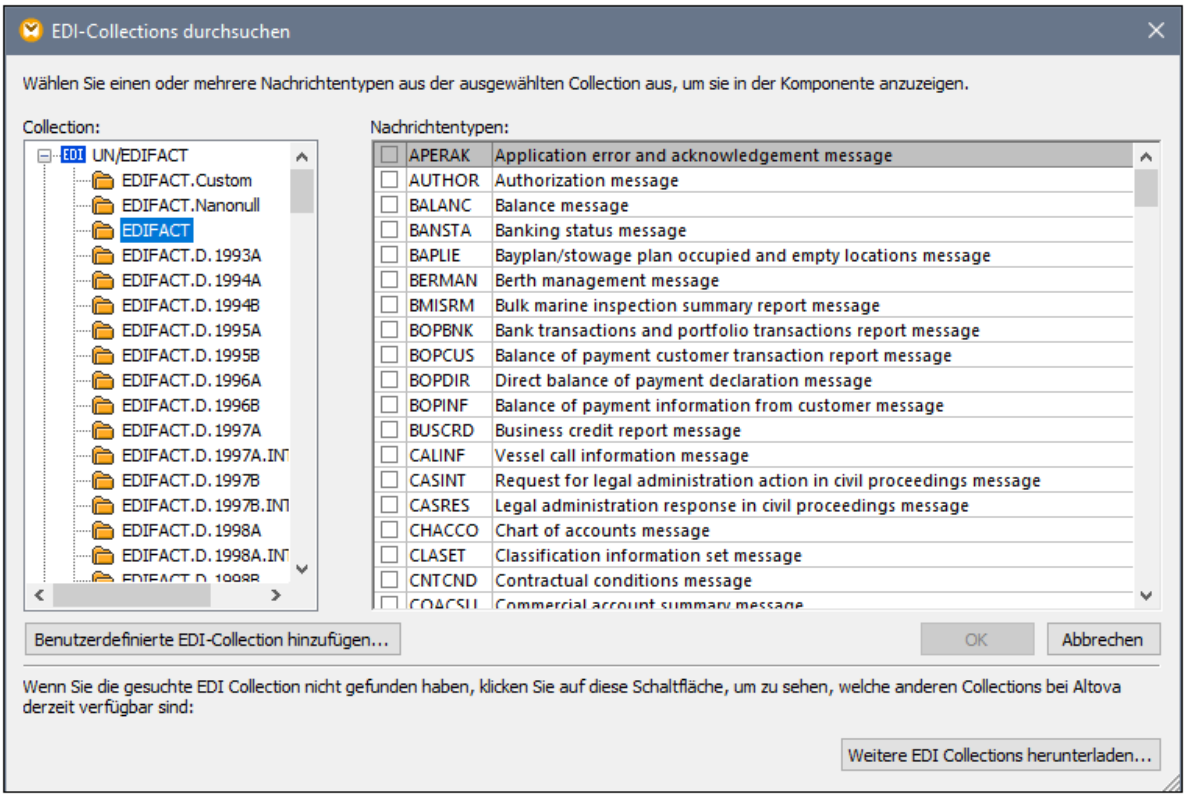
4.3.2 Hinzufügen von EDI-Komponenten zum Mapping

Um Daten von oder auf EDI-Dateien zu mappen, muss zuerst die EDI-Komponente zum Mapping hinzugefügt werden. Die EDI-Komponente stellt die Struktur der EDI-Nachricht (oder Nachrichten) im Mapping dar. Nachdem Sie die EDI-Komponente zum Mapping hinzugefügt haben, können Sie ähnlich wie bei anderen von MapForce unterstützten Komponententypen (siehe Arbeiten mit Komponenten 78) Mapping-Verbindungen zu oder von dieser Komponente aus ziehen. Falls nötig, können Sie die EDI-Komponente auch so konfigurieren, dass sie mehrere Nachrichten enthält. Auf diese Art können Daten aus mehreren EDI-Dateien (eine Nachricht pro Datei) gelesen werden (siehe Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 1371) bzw. können dadurch unterschiedliche Nachrichtentypen in derselben Datei verarbeitet werden.

So fügen Sie die EDI-Komponente zum Mapping hinzu:

1. Klicken Sie im Menü **Einfügen** auf **EDI**.
2. Klicken Sie auf die EDI Collection und aktivieren Sie das Kontrollkästchen neben dem gewünschten Nachrichtentyp. Aktivieren Sie die entsprechenden Kontrollkästchen, um mehrere Nachrichten

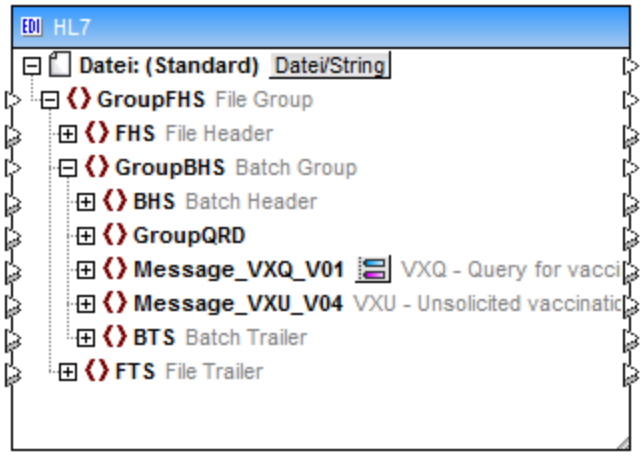
inkludieren. In der Abbildung unten werden z.B. zwei Nachrichten der HL7 Collection zum Mapping hinzugefügt.



3. Klicken Sie auf OK. Sie werden nun aufgefordert, eine EDI-Beispieldatei anzugeben.
4. Wenn Daten aus der EDI-Komponente gelesen werden sollen, navigieren Sie zur EDI-Beispieldatei. Klicken Sie andernfalls auf **Überspringen**.

Die Komponente wird nun im Mapping angezeigt und enthält die Struktur der beiden zuvor ausgewählten Nachrichten. Um weitere Nachrichten hinzuzufügen oder vorhandene zu entfernen, klicken Sie auf die Schaltfläche  neben der ersten Nachricht.

© 2016-2022 Altova GmbH



HL7-Mapping-Beispielkomponente, die zwei Nachrichten enthält

Sie können nun Mapping-Verbindungen von dieser EDI-Komponente zu einer Zielkomponente bzw. umgekehrt ziehen. Im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** finden Sie als Beispiele für das Mappen von Daten von oder auf verschiedene EDI-Varianten eine Reihe von Mapping-Design-Dateien (.mfd):

Format	Mapping-Beispiel
ASC X12	• **X12_To_Xml_Order.mfd** • **Xml_To_X12_Order.mfd**
HIPAA X12	• **HIPAA_837D.mfd**
HL7	• **HL7_MultiMessageTypes.mfd** • **HL7V260_To_HL7V3.mfd**
IATA	• **IATA_FlightInformationReport.mfd**
SAP IDoc	• **IDOC_Order.mfd**
UN/EDIFACT	• **EDI_Order.mfd** • **EDI_Order_DB.mfd** • **EDI_OrderReverse.mfd**

4.3.3 Ausführen von EDI-Mappings

Mappings, die EDI-Komponenten enthalten, werden im Allgemeinen unterstützt, wenn für das Mapping die Transformationssprache BUILT-IN[29] ausgewählt ist. Dadurch kann nicht nur direkt in MapForce eine Vorschau auf das Mapping angezeigt werden, sondern es kann auch in einer Server-Umgebung für die Ausführung durch MapForce Server bereitgestellt werden. Auch für Mappings, in denen eine *schnelle* (nicht-Standard) EDI-Konvertierung in XML aktiviert wurde wird der BUILT-IN-Prozessor benötigt, siehe Generieren von XML anhand von EDI[475].

Bei den meisten EDI-Formaten kann zusätzlich dazu auch Mapping-Code in C++, C# und Java generiert werden. Nähere Informationen dazu finden Sie in den Anmerkungen zur Unterstützung[1982].

Nachdem in MapForce ein EDI-Mapping erstellt wurde, kann es auf folgende Arten ausgeführt werden:

- Sie können direkt in MapForce eine Vorschau auf das Ergebnis der Datentransformation anzeigen, indem Sie auf das Register **Ausgabe** klicken. Die vom Mapping generierte Ausgabe wird je nach Konfiguration von MapForce und der Zielkomponente mit dem gewünschten Namen unter dem gewünschten Pfad gespeichert.
- Das Mapping kann über einen API-Aufruf oder die Befehlszeilenschnittstelle mit MapForce Server auf einem Server-Rechner (Windows, Linux, macOS) ausgeführt werden. Nähere Informationen dazu finden Sie unter Kompilieren von Mappings zu MapForce Server-Ausführungsdateien[1442].
- Das Mapping kann über HTTP (oder HTTPS) auf einem Server-Rechner (Windows, Linux, macOS), auf dem MapForce Server unter Verwaltung von FlowForce Server ausgeführt wird, bereitgestellt werden. In diesem Fall kann ein Mapping als geplanter Auftrag, bei Bedarf oder als Webservice ausgeführt werden, siehe Bereitstellen von Mappings auf FlowForce Server[1445].
- Für Mappings, die die Sprachen C++, C# oder Java unterstützen, können Sie anhand des Mappings Programmcode generieren. Das generierte Programm erzeugt dasselbe Ergebnis, wie bei Vorschau auf das Mapping in MapForce; das Programm kann als eigenständiges Programm oder integriert in Ihren vorhandenen Code ausgeführt werden, siehe Code Generator[1563].

Um nur die für eine EDI-Komponente (entweder Quell- oder Zielkomponente) geltenden Einstellungen zu konfigurieren, klicken Sie mit der rechten Maustaste auf die Titelleiste der Komponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften**, siehe EDI-Komponenteneinstellungen[466]. Informationen zu Einstellungen zu anderen Komponententypen finden Sie unter Ändern der Komponenteneinstellungen[82].

Um die Mapping-Einstellungen für das gesamte Mapping zu konfigurieren, klicken Sie mit der rechten Maustaste auf das Mapping und wählen Sie im Kontextmenü den Befehl **Mapping-Einstellungen**, siehe Ändern der Mapping-Einstellungen[116].

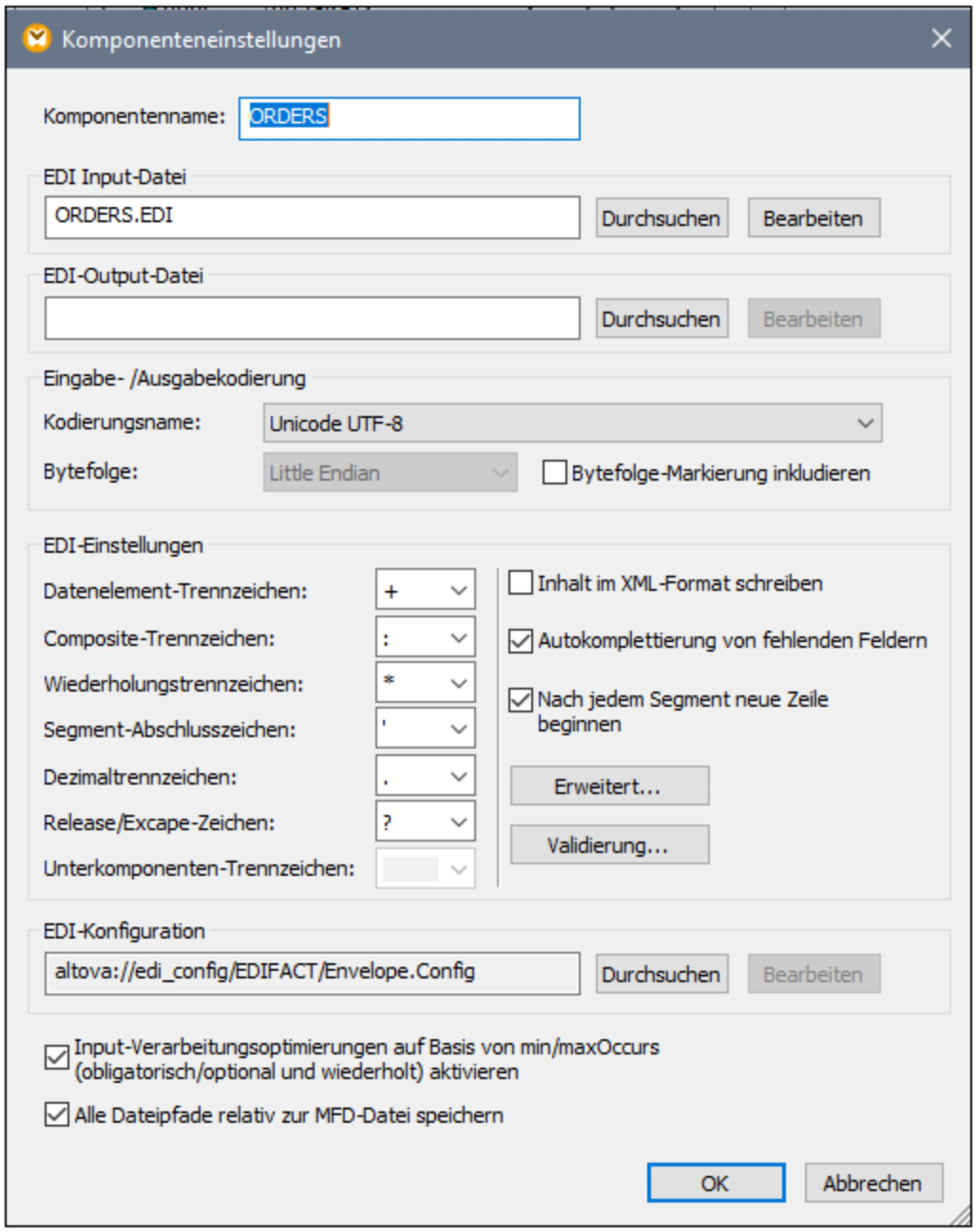
Um die Codegenerierungseinstellungen aufzurufen, gehen Sie zu **Extras | Optionen** und klicken Sie anschließend auf das Register **Code-Generierung**.

4.3.4 EDI-Komponenteneinstellungen

In diesem Kapitel erfahren Sie, wie EDI-Komponenteneinstellungen definiert und mapbare EDI-Trennzeichen in einem Mapping verwendet werden. Nachdem Sie eine EDI-Komponente zum Mapping-Bereich hinzugefügt haben, können Sie die Einstellungen dafür über das Dialogfeld **Komponenteneinstellungen** (*siehe Abbildung unten*) konfigurieren. Sie können das Dialogfeld **Komponenteneinstellungen** auf eine der folgenden Arten öffnen:

- Doppelklicken Sie im Mapping auf die Titelleiste der EDI-Komponente.
- Wählen Sie eine EDI-Komponente im Mapping aus und klicken Sie auf den Menübefehl **Komponente | Eigenschaften**.
- Klicken Sie mit der rechten Maustaste auf die Titelleiste der EDI-Komponente und wählen Sie **Eigenschaften**.

© 2016-2022 Altova GmbH



Es stehen die folgenden Einstellungen zur Verfügung. Beachten Sie, dass einige Einstellungen nicht zur Verfügung stehen, wenn sie von der aktuell ausgewählten Komponente nicht unterstützt werden.

- Komponentenname

  Der Komponentenname wird bei der Erstellung der Komponente automatisch generiert. Sie können den Namen jedoch jederzeit ändern. Der Komponentenname kann Leerzeichen und Punkte enthalten, darf aber keine Schrägstriche, umgekehrten Schrägstriche, Doppelpunkte, doppelten Anführungszeichen und voran- oder nachgestellte Leerzeichen enthalten. Beachten Sie beim Ändern des Namens einer Komponente Folgendes:

- Wenn Sie das Mapping auf FlowForce Server bereitstellen möchten, muss der Komponentenname eindeutig sein.
- Es wird empfohlen, nur Zeichen zu verwenden, die über die Befehlszeile eingegeben werden können. Länderspezifische Sonderzeichen sind in Windows und der Befehlszeile eventuell unterschiedlich kodiert.

EDI-Input-Datei

Definiert die EDI-Quelldatei, aus der MapForce die Daten ausliest.

EDI-Output-Datei

Definiert die EDI-Zieldatei, in die MapForce die Daten schreibt.

Eingabe- / Ausgabekodierung

In diesem Abschnitt können Sie die folgenden Einstellungen der Output-Instanzdatei definieren: die Kodierung, die Bytefolge und ob das Bytefolge-Markierungszeichen (BOM) inkludiert werden soll. Standardmäßig haben alle neuen Komponenten die in der Option *Standardkodierung für neue Komponenten* definierte Kodierung. Sie können diese Option über **Extras | Optionen | Allgemein** aufrufen. Nähere Informationen dazu finden Sie unter Optionen 1539.

EDI-Einstellungen

In dieser Einstellungsgruppe können Sie EDI-Trennzeichen (z.B. Segment-Abschlusszeichen, Datenelementtrennzeichen) definieren. Beachten Sie, dass die Einstellungen nur zur Verfügung stehen, wenn diese vom verwendeten EDI-Format unterstützt werden. Die EDI-Trennzeichen, die Sie in diesem Dialogfeld definieren, werden immer, wenn MapForce EDI-Dateien schreibt, angewendet. Beim Lesen von EDI-Dateien werden diese Trennzeichen nur angewendet, wenn in der Input-Datei keine eigenen Trennzeichen definiert/enthalten sind (z.B. in EDIFACT-Dateien ohne das UNA-Segment).

Wenn eine EDI-Input-Komponente/Datei Trennzeichendefinitionen enthält (z.B. eine X12-Datei mit einem ISA-Segment), so setzen die vorhandenen Trennzeichen alle im Dialogfeld **Komponenteneinstellungen** definierten Trennzeichen für diese Datei außer Kraft. Sie können als Trennzeichen nicht druckbare Zeichen verwenden. Geben Sie dazu in eine der Auswahllisten **x** gefolgt vom hexadezimalen ASCII-Zeichencode ein, z.B. **x1e** für das RS-Steuerzeichen (ASCII-Datensatztrennzeichen, Dezimalcode 30).

Das *Unterkomponenten-Trennzeichen* gilt für den HL7-Standard. Der Standardwert für dieses Trennzeichen ist das kaufmännische Und (**&**).

Inhalt im XML-Format schreiben

Die Aktivierung dieses Kontrollkästchens ist nur bei einer EDI-Zielkomponente sinnvoll. Mit dieser Option wird eine XML-Ausgabe generiert. Nähere Informationen dazu finden Sie unter Beispiel: Wechseln zwischen der XML- und EDI-Ausgabe 478.

Autokomplettierung von fehlenden Feldern

Diese Option wird auf EDI-Zielkomponenten angewendet. Wenn dieses Kontrollkästchen aktiviert ist, füllt MapForce die Werte einiger Datenfelder automatisch aus. Dies gilt nur für die Felder, wo diese Operation nicht im Widerspruch zur Spezifikation des aktuell ausgewählten EDI-Formats steht. Nähere Informationen zur Autokomplettierungsoption finden Sie in den folgenden Kapiteln:

© 2016-2022 Altova GmbH

- ASC X12-Regeln zur automatischen Ergänzung von Daten 480
- UN/EDIFACT-Regeln zur automatischen Ergänzung von Daten 529

Nach jedem Segment neue Zeile beginnen

Diese Option wird auf EDI-Zielkomponenten angewendet. Wenn das Kontrollkästchen aktiviert ist, fügt MapForce ein CR/LF-Zeichen (Carriage Return / Line Feed) nach jedem EDI-Segment hinzu. Gemäß dem EDI-Standard werden derartige Zeilen in einer Nachricht ignoriert.

Erweitert

Ruft ein Dialogfeld auf, in dem Sie zusätzliche Einstellungen für die aktuelle EDI-Komponente definieren können. Je nach Art der EDI-Komponente stehen unterschiedliche Einstellungen zur Verfügung.

Validierung

Ruft ein Dialogfeld auf, in dem Sie die Validierungseinstellungen für die aktuelle EDI-Komponente definieren können. Nähere Informationen dazu finden Sie unter Validierung von EDI-Komponenten 471.

EDI-Konfiguration

Liefert Informationen über die von der aktuellen Komponente verwendete EDI-Konfiguration. Nähere Informationen dazu finden Sie unter Wie MapForce EDI-Konfigurationen auflöst 541.

Input-Verarbeitungsoptimierungen auf Basis von min/maxOccurs aktivieren

Diese Option ermöglicht die Sonderbehandlung von Sequenzen, die nur ein einziges Datenelement enthalten (z.B. erforderliche Attribute und Child-Elemente mit `minOccurs` und `maxOccurs="1"`). In diesem Fall wird das erste Datenelement der Sequenz extrahiert, anschließend wird das Datenelement dann direkt als atomarer Wert (und nicht als Sequenz) verarbeitet.

Wenn die Input-Daten gemäß dem Schema *nicht gültig* sind, kommt es im Mapping zu einem Fehler. Damit auch ein ungültiger Input verarbeitet werden kann, deaktivieren Sie dieses Kontrollkästchen.

Alle Dateipfade relativ zur MFD-Datei speichern

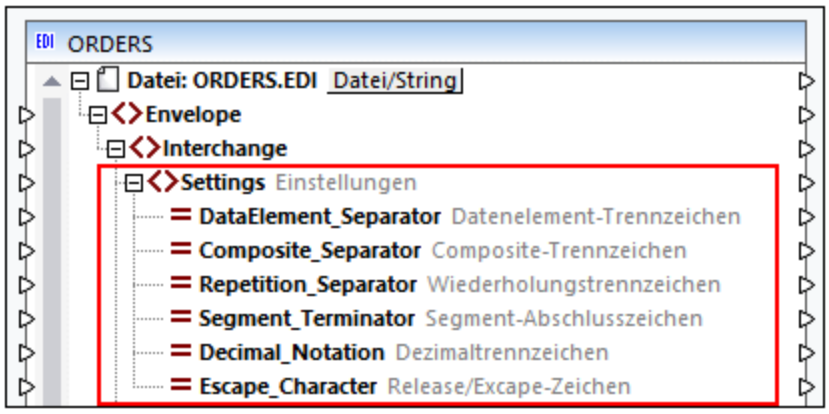
Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld **Komponenteneinstellungen** angezeigten Dateipfade relativ zum Ordner, in dem sich die MapForce Design (**`.mfd`**)-Datei befindet. Diese Einstellung wirkt sich auf die folgenden Dateien aus:

- die EDI-Input-Datei (falls vorhanden)
- die EDI-Output-Datei (falls vorhanden)

Siehe auch Relative und absolute Pfade 85.

Mapbare EDI-Trennzeichen

Sie können in MapForce EDI-Trennzeichen (z.B. Datenelement-Trennzeichen, Segment-Abschlusszeichen, usw.) mappen. Diese Funktion wird derzeit für die folgenden EDI-Standards unterstützt: EDIFACT, X12 und NCPDP SCRIPT. Die mapbaren EDI-Trennzeichen in einer EDI-Komponente befinden sich im Element `Settings` (Child-Element des Elements `Interchange`) (*Abbildung unten*).



In der Abbildung oben sehen Sie einen Ausschnitt aus einer EDIFACT-Nachricht, in der jedes Trennzeichen durch einen einzigen Child-Node dargestellt wird. Alle diese Trennzeichen können im Dialogfeld **Komponenteneinstellungen** konfiguriert werden (*Unterabschnitt oben*). Beachten Sie, dass im Element `Settings` nur diejenigen Trennzeichen angezeigt werden, die für den jeweiligen EDI-Standard konfiguriert werden können.

Anmerkung zu 997- und 999-Nachrichten
Wenn Sie ein Mapping für die X12/HIPAA.X12 997- und 999-Nachricht erstellen, müssen Sie zwischen dem Node `Settings` der Input-Instanz und dem Node `Settings` der 997- und 999-Nachricht der Ausgabe eine "Alles kopieren"-Verbindung[97] erstellen.

EDI-Quell- und -Zielkomponente
In Quellkomponenten stellen die Trennzeichen-Nodes die tatsächlich verwendeten Trennzeichen aus der Quellinstanz bereit. Diese unterschieden sich eventuell von den im Dialogfeld **Komponenteneinstellungen** definierten Trennzeichen. Bei Zielkomponenten können die Trennzeichen in den Trennzeichen-Nodes zur Laufzeit definiert werden, d.h. gemappte Trennzeichen setzen die im Dialogfeld **Komponenteneinstellungen** definierten außer Kraft.

Validierungsüberprüfungen
Beachten Sie die folgenden Punkte im Zusammenhang mit Validierungen:

- Wenn ein Trennzeichen-Node nicht verbunden wird, ändert sich das aktuelle Verhalten nicht.
- Wenn eine leere Sequenz oder eine Sequenz bestehend aus mehr als einem String gemappt wird, kommt es zu einem Laufzeitfehler.
- Wiederholungs-Trennzeichen in EDIFACT: Wenn ein Leerzeichen auf ein Trennzeichen gemappt wird, wird das Trennzeichen als nicht verwendet markiert. Das UNA-Segment hat ein Leerzeichen anstatt des Trennzeichens. Wenn Sie ein Leerzeichen auf ein tatsächlich benötigtes Trennzeichen mappen, kommt es zu einem Fehler, weil das Trennzeichen nicht definiert wurde.
- Wenn ein String gemappt wird, der nicht genau ein Zeichen enthält, kommt es zu einem Laufzeitfehler.
- Wenn dasselbe Zeichen auf zwei oder mehr Trennzeichen gemappt wird, kommt es zu einem Laufzeitfehler.

© 2016-2022 Altova GmbH

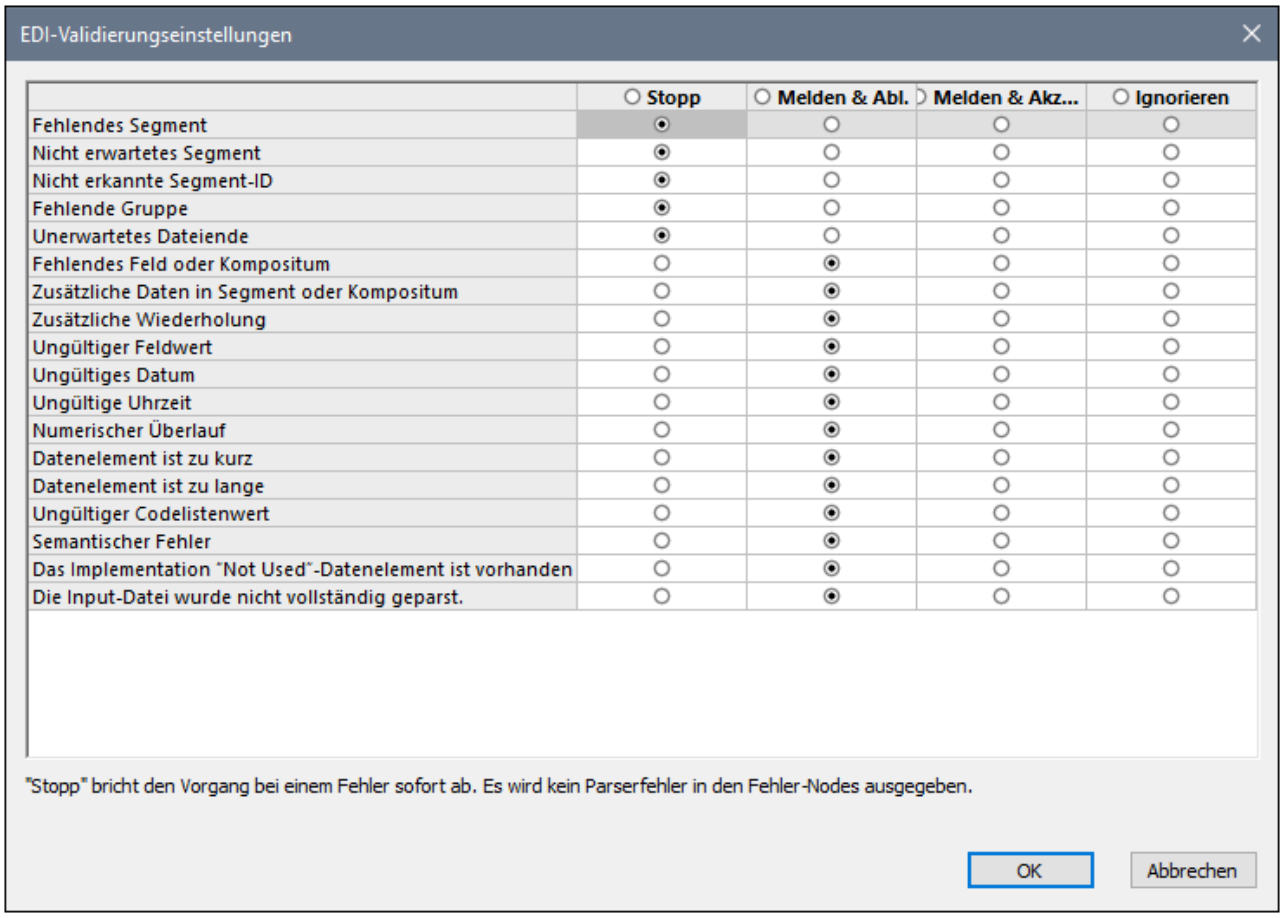
4.3.5 Validierung von EDI-Komponenten

MapForce validiert alle unterstützten EDI-**Quell-** und -**Zieldokumente**, wenn das Mapping ausgeführt wird. Dies geschieht entweder, wenn Sie in MapForce eine Vorschau auf das Mapping anzeigen, indem Sie auf die Schaltfläche **Ausgabe** klicken oder wenn das Mapping durch einen externen Ausführungsprozessor wie z.B. MapForce Server oder ein C++, C# oder Java-Programm ausgeführt wird. Sie können auch explizit die vom Mapping erzeugte EDI-Ausgabe validieren. Klicken Sie dazu zuerst auf das Register **Ausgabe** und anschließend auf die Symbolleisten-Schaltfläche **Ausgabe validieren**  (siehe auch Validieren der Mapping-Ausgabe 107).

MapForce validiert die in dieser Dokumentation angegebenen EDI-Formate. Dabei werden jedoch nicht die gesamte EDI-Syntax oder -Semantik validiert. Es kann z.B. in einigen Fällen vorkommen, dass die möglichen Werte eines Felds auf EDI-externen Standards basieren und nicht einfach als eine endliche und vollständige Gruppe von Werten, für die alle Verarbeitungsmöglichkeiten im Vorhinein bekannt sind, ausgedrückt werden können. In solchen Fällen können die Validierungsregeln mit Hilfe von Konfigurationsdateien an den jeweiligen Einzelfall angepasst werden (siehe Anpassen der EDI-Validierung 573).

Auf einer sehr einfachen Ebene können die Validierungsparameter direkt über die Benutzeroberfläche von MapForce konfiguriert werden. So können Sie festlegen, dass Fehler ignoriert werden oder die Mapping-Ausführung bei einem bestimmten Fehler gestoppt wird.

Um die Validierungseinstellungen anzupassen, doppelklicken Sie auf eine EDI-Komponente und klicken Sie im Dialogfeld "Komponenteneinstellungen" auf die Schaltfläche **Validierung**... (siehe auch EDI-Komponenteneinstellungen 466).In der Abbildung unten sehen Sie die Standardeinstellungen.



Dialogfeld "EDI-Validierungseinstellungen"

Für jedes oben aufgelistete Validierungsereignis können beliebige der folgenden Aktionen ausgewählt werden: **Stopp**, **Melden & Ablehnen**, **Melden & Akzeptieren**, **Ignorieren**.

Bei **Stopp** werden schwerwiegende Fehler ausfindig gemacht und die Ausführung des Mappings abgebrochen.

Melden & Ablehnen liefert Informationen über den Fehler und weist die aktuelle Nachricht zurück. **Melden & Akzeptieren** liefert Informationen über den Fehler und akzeptiert die aktuelle Nachricht. Die eigentliche Fehlermeldung wird im Fenster "Meldungen" von MapForce angezeigt. Bei X12- und HIPAA-Quellkomponenten werden die Validierungsfehler auch in der Mapping-Komponente protokolliert, wodurch X12 997- oder X12 999-Berichte generiert werden können (siehe Validierungsergebnisse und X12 Acknowledgement 474).

Bei **Ignorieren** wird das Validierungsereignis ignoriert. Wenn Sie diese Aktion auswählen, werden keine Informationen über den Fehler bereitgestellt. Verwenden Sie die Aktion daher mit Vorsicht.

© 2016-2022 Altova GmbH

Wenn bei der Verarbeitung eines EDI-Mappings unerwartete Segmente gefunden werden, können nur MapForce und MapForce Server den Fehler beheben und den Parsing-Vorgang fortsetzen. Aufgrund unerwarteter Segmente entstandene Fehler im generierten C++-, C#- oder Java-Code können nicht behoben werden und der Vorgang wird abgebrochen. Zu den unerwarteten Segmenten gehören die folgenden Fehler: falsche Segmentreihenfolge, fehlender obligatorischer Segmentname, zusätzliche Segmente, die in den Konfigurationsdateien nicht vorhanden sind.

Für die folgenden EDI-Validierungsereignisse können benutzerdefinierte Aktionen (**Stopp, Ignorieren**, usw.) konfiguriert werden:

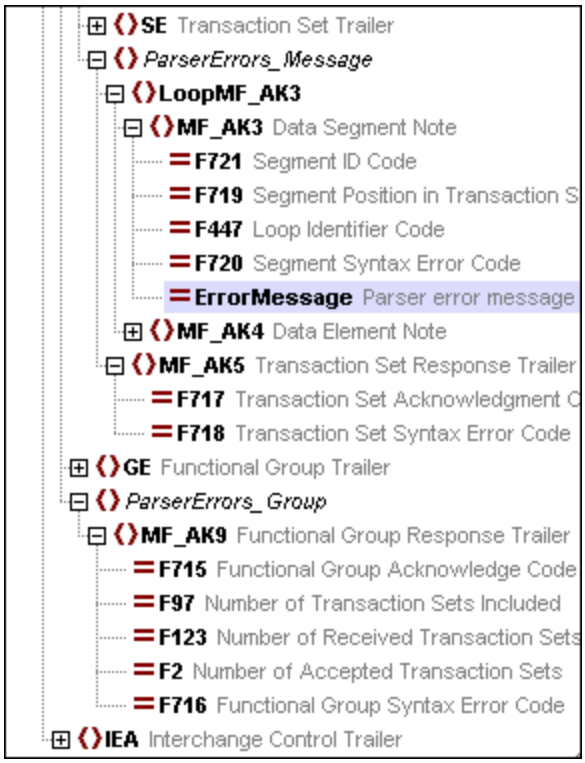
Ereignis	**Beschreibung**	**Fehler kann auftreten bei einer**
Fehlendes Segment	Ein obligatorisches Segment fehlt oder kommt weniger oft als im Mindestwert definiert vor.	Quellkomponente
Nicht erwartetes Segment	Ein Segment ist zwar in der Spezifikation, nicht aber in dieser Nachricht definiert.	Quellkomponente
Nicht erkannte Segment-ID	Es wurde ein Segment gefunden, das in der Spezifikation nicht definiert ist.	Quellkomponente
Fehlende Gruppe	Eine obligatorische Gruppe fehlt oder kommt weniger oft als im Mindestwert definiert vor.	Quellkomponente
Unerwartetes Dateiende	Die Instanz kann nicht geparst werden, da einige Daten fehlen.	Quellkomponente
Fehlendes Feld oder Kompositum	Ein obligatorisches Feld oder Kompositum fehlt oder kommt weniger oft als im Mindestwert definiert vor.	Quell- oder Zielkomponente
Zusätzliche Daten in Segment oder Kompositum	Die Input-Instanz enthält zusätzliche Daten, die von der Syntaxbeschreibung nicht erwartet werden.	Quellkomponente
Zusätzliche Wiederholung	Die tatsächliche Anzahl der Felder in einem Segment/Kompositum übersteigt die definierte Maximalanzahl	Quell- oder Zielkomponente
Ungültiger Feldwert	Ein numerisches Feld enthält ein ungültiges Zeichen.	Quellkomponente
Ungültiges Datum	Ein Datumsfeld enthält entweder ein ungültiges Zeichen oder die Werte für den Monat oder Tag sind ungültig.	Quellkomponente
Ungültige Uhrzeit	Ein Uhrzeitfeld enthält entweder ein ungültiges Zeichen oder die Werte für die Stunden oder Minuten sind ungültig.	Quellkomponente

Numerischer Überlauf	Ein numerischer Wert geht über die definierte Domain hinaus. Dieser Fehler wird nur innerhalb des generierten Codes unterstützt.	Quellkomponente
Datenelement ist zu kurz	Die Länge eines Datenelemente ist geringer als der definierte Mindestwert.	Quell- oder Zielkomponente
Datenelement ist zu lange	Die Länge eines Datenelemente ist größer als der definierte Maximalwert.	Quell- oder Zielkomponente
Ungültiger Codelistenwert	Der Wert des Datenelements stimmt mit keinem der vorkonfigurierten Codewerte überein. Siehe auch Ändern der möglichen Codewerte von Datenelementen 556.	Quell- oder Zielkomponente
Semantischer Fehler	Es ist ein semantischer Fehler aufgetreten.	Quell- oder Zielkomponente
Das Implementation “Not Used”-Datenelement ist vorhanden	In der Input-Datei ist ein Element vorhanden, das laut HIPAA-Konfigurationsdatei nicht zulässig ist (`maxOccurs="0"`).	Quellkomponente
Die Input-Datei wurde nicht vollständig geparst	Die Input-Datei wurde nicht zur Gänze geparst.	Quellkomponente

Validierungsergebnisse und X12 Acknowledgment

Wenn MapForce eine X12- oder HIPAA-Komponente validiert, werden die Validierungsergebnisse in "virtuelle" Datenelemente am unteren Rand der EDI-Komponente (unter "ParserErrors_Message" und "ParserErrors_Group", siehe Abb. unten) platziert. Anhand dieser Informationen können X12 997- oder X12 999-Dateien mit Informationen über den Status des Datenaustauschs generiert werden (siehe Generieren eines X12 997 Functional Acknowledgment 481 und Generieren eines X12 999 Implementation Acknowledgment 483).

© 2016-2022 Altova GmbH



Bei X12- oder HIPAA.X12-Komponenten haben die im Dialogfeld "EDI-Validierungseinstellungen" (Beschreibung siehe oben) verfügbaren Validierungsaktionen die folgende Bedeutung:

Stopp beendet die Ausführung des Mappings, ohne dass ein Fehlerbericht generiert wird.

Melden & Ablehnen und **Melden & Akzeptieren** stellt in den Datenelementen "ParserErrors_Message" und "ParserErrors_Group" der EDI-Komponente Informationen bereit, die weiter gemappt werden können.

Je nachdem, ob **Ablehnen** oder **Akzeptieren** ausgewählt ist, enthalten die Datenelemente "Functional Group Acknowledge Code F715" und "Transaction Set Acknowledgment Code F717" entweder:

den Wert 'R' für *'Rejected'* (Abgelehnt) oder

den Wert 'E' für 'Akzeptiert, doch wurden Fehler gefunden'. Die Fehler werden auch im Fenster "Meldungen" angezeigt.

Ignorieren ignoriert den jeweiligen Fehler. In den Datenelementen "ParserErrors_Message" und "ParserErrors_Group" werden keine Informationen zur Verfügung gestellt.

4.3.6 Generieren von XML anhand von EDI

Wie bereits erwähnt, können Sie EDI-Daten von und auf jedes andere von MapForce unterstützte Datenformat mappen. EDI-Mappings unterscheiden sich kaum von Nicht-EDI-Mappings und ermöglichen denselben Grad an Kontrolle über die generierten Daten.

Zusätzlich zu regulären Mappings, für die ein Design erstellt werden muss, bietet MapForce auch die Möglichkeit, EDI-Dateien mit wenigen Klicks *schnell* in das XML-Format zu konvertieren. Dazu muss weder ein

XML-Schema erstellt werden, noch müssen manuell Mapping-Verbindungen gezogen werden. Dies ist eine praktische Funktion, mit der Sie EDI-Dateien in ein - bedeutend einfacher zu lesendes - Alternativformat exportieren können. Die XML-Ausgabe ist wohlgeformt und kann wie die meisten anderen von MapForce generierten Ausgabedokumente in einer Datei gespeichert werden.

Für diese Funktion wird weder ein XML-Schema benötigt noch wird dadurch eines bereitgestellt. In der generierten XML-Datei werden einfach alle Daten aus der EDI-Quellkomponente repliziert. Dabei werden die Datenelementnamen und deren Werte ausgegeben und die Elementhierarchie bleibt erhalten. Felder, die keine Daten enthalten, werden weggelassen. Beachten Sie, dass die genaue Struktur der generierten XML-Datei von den EDI-Quelldaten abhängig ist und nicht geändert werden kann. Wenn Sie vollständige Kontrolle über die generierten XML-Daten benötigen, erstellen Sie stattdessen ein Standard-EDI-auf-XML-Mapping, über das Sie jedes einzelne Mapping-Datenelement mittels Filtern, Funktionen, usw. bearbeiten können. Beispiele zu Standard-EDI-auf-XML-Mappings finden Sie unter Beispiel: Mappen von UN/EDIFACT auf XML[531], Beispiel: Mappen von HIPAA X12 auf XML[485] und Beispiel: Mappen von HL7 auf XML[490].

Voraussetzungen

Um EDI-Dateien *schnell* mit Hilfe von MapForce in XML zu konvertieren, muss als Transformationssprache des Mappings BUILT-IN ausgewählt sein; es werden keine anderen Transformationssprachen unterstützt. Diese Voraussetzung gilt spezifisch für die *schnelle* Konvertierung von EDI in XML. Reguläre EDI-auf-XML-Mappings unterstützen zusätzliche Sprachen, siehe EDI[460].

Schnelle Konvertierung von EDI in XML

Nachdem Sie eine EDI-Komponente zum Mapping hinzugefügt haben, gibt es zwei Arten, um anhand dieser Komponente XML zu generieren:

1. Wenn die EDI-Komponente Daten aus einer EDI-Instanzdatei ausliest (d.h. wenn es sich um eine Quellkomponente handelt), klicken Sie mit der rechten Maustaste auf ihre Titelleiste und wählen Sie im Kontextmenü den Befehl **Mapping auf XML erstellen**. Ein Schritt-für-Schritt-Beispiel dazu finden Sie unter Beispiel: Schnelle Konvertierung von EDI in[476].

Anmerkung: Sie können dies auch für Zielkomponenten durchführen, wenn Sie zusätzlich zur bestehenden EDI-Ausgabe eine neue Ausgabe erstellen möchten und das Mapping dadurch in ein "verkettetes" Mapping umwandeln. Nähere Informationen dazu finden Sie unter Verkettete Mappings[1299].

2. Wenn es sich bei der EDI-Komponente um eine Zielkomponente handelt, doppelklicken Sie zuerst auf ihre Titelleiste, um die Komponenteneinstellungen zu öffen und aktivieren Sie anschließend das Kontrollkästchen **Inhalt im XML-Format schreiben**. Ein Schritt-für-Schritt-Beispiel dazu finden Sie unter Beispiel: Wechseln zwischen der XML- und EDI-Ausgabe[478].

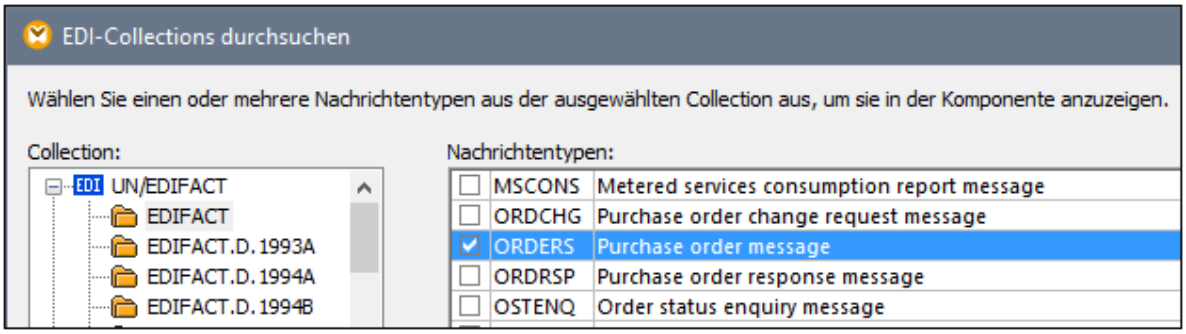
4.3.6.1 Beispiel: Schnelle Konvertierung von EDI in XML

In diesem Beispiel wird gezeigt, wie Sie Daten mit möglichst wenig Aufwand aus einer EDI-Datei ins XML-Format konvertieren. In diesem Beispiel wird die *schnelle* Konvertierung erläutert. Dabei müssen Mapping-Verbindungen nicht manuell gezogen werden. Ein Beispiel zu einer regulären Konvertierung von EDI in XML finden Sie unter: Beispiel: Mappen von UN/EDIFACT auf XML[531].

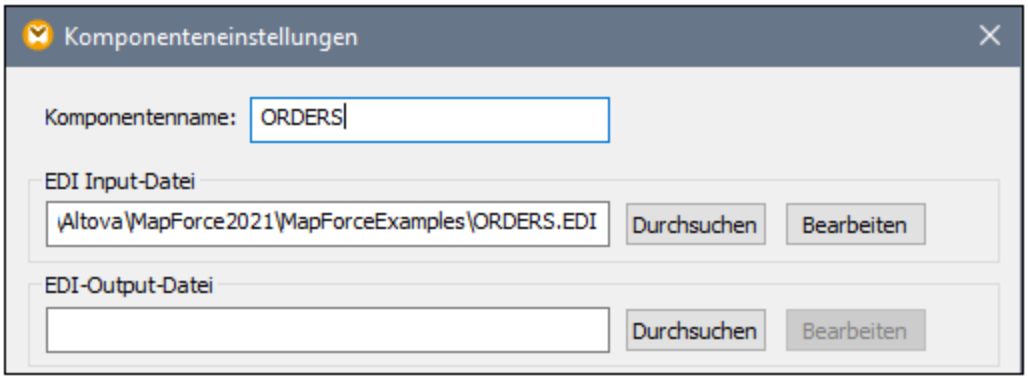
1. Erstellen Sie ein neues Mapping und wählen Sie als Transformationssprache BUILT-IN aus. Siehe auch Auswählen einer Transformationssprache[28].
2. Klicken Sie im Menü **Einfügen** auf **EDI**.

© 2016-2022 Altova GmbH

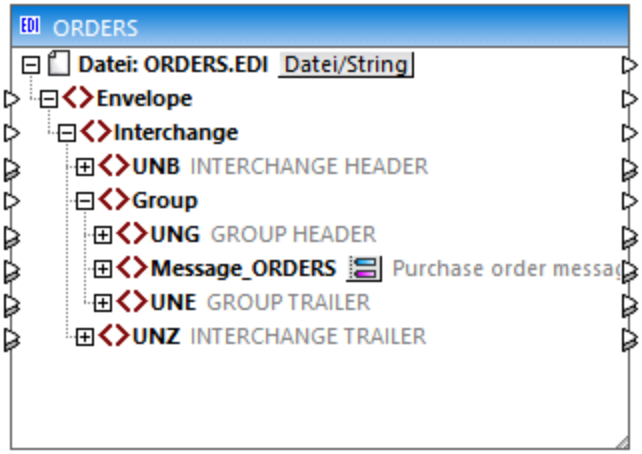
3. Wählen Sie die **EDIFACT** Collection und den Message-Typ **ORDERS** aus und klicken Sie auf **OK**.



4. Wenn Sie aufgefordert werden, eine EDI-Beispieldatei bereitzustellen, klicken Sie auf **Durchsuchen** und öffnen Sie die Datei **ORDERS.EDI** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Nach dem Öffnen der Datei wird das Dialogfeld "Komponenteneinstellungen" angezeigt. Auf diese Art können Sie die Einstellungen der EDI-Komponente überprüfen, bevor Sie diese zum Mapping hinzufügen. Sie können diese Einstellungen 466 später bei Bedarf jederzeit ändern. Beachten Sie, dass die Datei **ORDERS.EDI** als **EDI-Input-Datei** angezeigt wird.

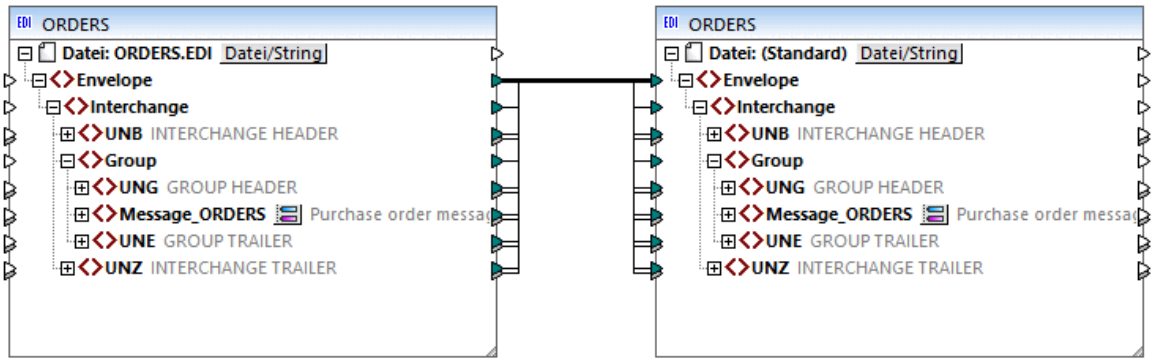


5. Klicken Sie auf **OK.** Die EDI-Komponente wird nun im Mapping-Bereich angezeigt.

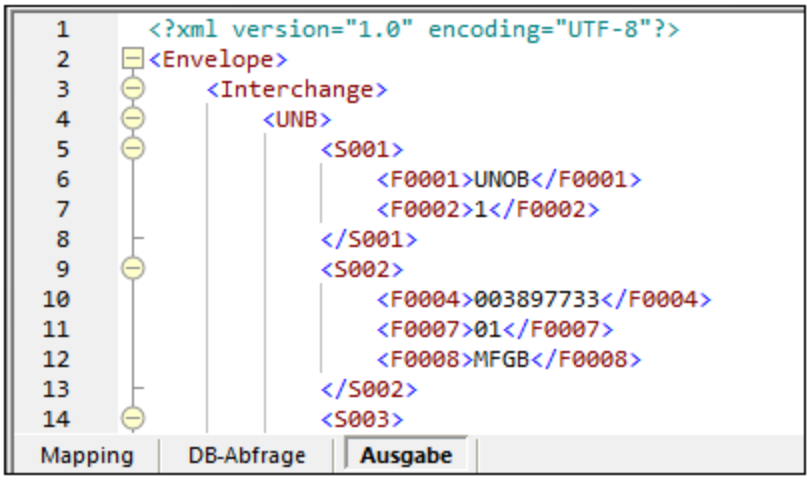


6. Klicken Sie mit der rechten Maustaste auf die Titelleiste der ORDERS-Komponente und aktivieren Sie im Kontextmenü die Option **Mapping auf XML erstellen**. Daraufhin wird eine XML-Zielkomponente

erstellt und alle Verbindungen dazu werden automatisch mittels einer Alles kopieren[97]-Verbindung hergestellt.



7. Klicken Sie auf das Register **Ausgabe**, um eine Vorschau auf die generierte XML-Datei zu sehen.



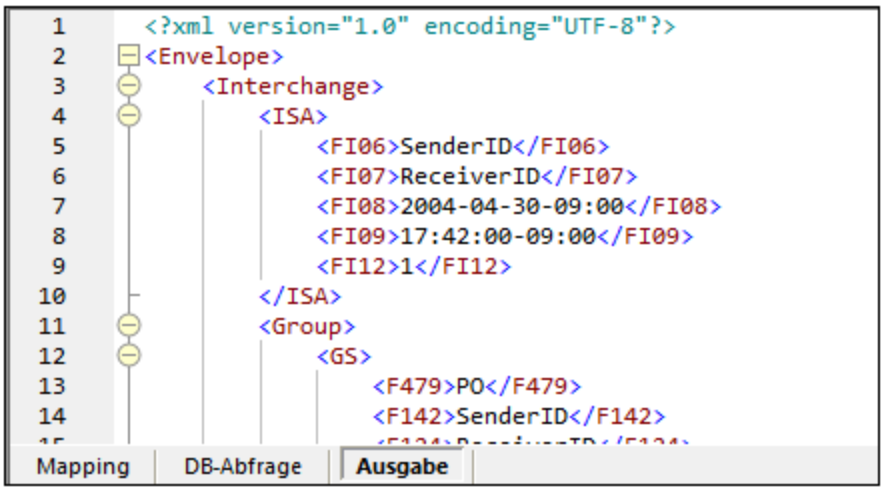
4.3.6.2 Beispiel: Wechsel zwischen der XML- und EDI-Ausgabe

In diesem Beispiel wird gezeigt, wie Sie eine EDI-Zielkomponente so konfigurieren, dass sie eine XML-Ausgabe anstelle der EDI-Ausgabe erzeugt. Die Konfiguration ist minimal. Es muss dabei nur, wie unten gezeigt, eine einzige Eigenschaft geändert werden.

1. Öffnen Sie das folgende Mapping: **<Dokumente>\Altova\MapForce2023\MapForceExamples\Xml_To_X12_Order.mfd**.
2. Ändern Sie die Transformationssprache in BUILT-IN (Wählen Sie im Menü **Ausgabe** den Befehl **Built-In Ausführungsprozessor**).
3. Doppelklicken Sie auf die Titelleiste der Zielkomponente **850**. Daraufhin wird das Dialogfeld "EDI-Komponenteneinstellungen" geöffnet.
4. Aktivieren Sie das Kontrollkästchen **Inhalt im XML-Format** schreiben.

Durch Klicken auf das Register **Ausgabe** können Sie nun eine Vorschau auf das Mapping anzeigen. Wie unten gezeigt, generiert die EDI-Komponente nun anstelle von EDI eine XML-Ausgabe.

© 2016-2022 Altova GmbH



4.3.7 ASC X12

ASC X12 ist ein Industriestandard für den Dokumentenaustausch und MapForce unterstützt die folgenden Versionen: 3040, 3050, 3060, 3070, 4010, 4020, 4030, 4040, 4041, 4042, 4050, 4051, 4052, 4060, 5010, 5011, 5012, 5020, 5030, 5040, 5050, 6010, 6020, 6030 und 6040.

Als Standard-ASC X12-Version wird in MapForce 6040 verwendet.

ANSI X12-Komponenten haben "virtuelle" Nodes, in die, je nachdem, welche Einstellungen Sie im Dialogfeld "EDI-Validierungseinstellungen" (siehe Validierung von EDI-Komponenten 471) auswählen, EDI-Parser-Fehlerinformationen/Daten geschrieben werden. Ein X12 EDI 997 Functional Acknowledgement kann von jedem X12-Dokument generiert werden.

Informationen zu ASC X12-Publikationen finden Sie unter http://www.x12.org/.

4.3.7.1 ASC X12-Validierungsregeln

Bei der Validierung eines ASC X12-Dokuments werden die folgenden Punkte überprüft:

- ob ein ISA- und ein IEA-Segment vorhanden sind
- ob ISA/I01 einen zulässigen Authorization Information Qualifier enthält.
- ob ISA/I03 einen zulässigen Security Information Qualifier enthält.
- ob die beiden ISA/I05 Segmente zulässige Interchange ID Qualifier enthält.
- ob ISA/I08 einen wohlgeformten Datumswert enthält.
- ob ISA/I09 einen wohlgeformten Uhrzeitwert enthält.
- ob ISA/I13 einen zulässigen Boolschen Wert enthält.
- ob ISA/I14 einen zulässigen Interchange Usage Indicator enthält.
- ob ISA/I12 und IEA/I12 denselben Wert enthalten.
- ob IEA/I16 die richtige Anzahl von Funktionsgruppen im Interchange enthält.

Jede Funktionsgruppe wird auf folgende Punkte überprüft:

- ob sie ein übereinstimmendes GS- und GE-Paar enthält.
- ob GS/373 einen wohlgeformten Datumswert enthält.
- ob GS/337 einen wohlgeformten Uhrzeitwert enthält.
- ob GS/28 und GE/28 denselben Wert enthalten.
- ob GE/97 die richtige Anzahl Nachrichten in der Funktionsgruppe enthält.

Jede Message wird auf folgende Punkte überprüft:

- ob es ein übereinstimmendes ST- und SE-Paar gibt.
- ob ST/143 den richtigen Message Identifier enthält.
- ob ST/329 und SE/329 denselben Wert enthalten.
- ob SE/96 die richtige Anzahl von Segmenten in der Message enthält.

4.3.7.2 ASC X12-Regeln zur automatischen Ergänzung von Daten

MapForce kann bei der Generierung einer ASC X12-Datei anhand der unten aufgelisteten Regeln (die in der Folge als _"Regeln zur automatischen Ergänzung von Daten" bzw. "Autokomplettierungsregeln" bezeichnet werden) automatisch bestimmte Felder oder Werte erstellen. Um dieses Verhalten zu deaktivieren, entfernen Sie in den EDI-Komponenteneinstellungen[466] das Häkchen aus dem Kontrollkästchen **Autokomplettierung von fehlenden Feldern.**

Die automatische Datenkomplettierung für EDI/X12 stellt sicher:

- dass auf dem Interchange Level ein ISA- und IEA-Paar vorhanden sind.
- dass bei Vorhandensein von entweder GS oder GE auch die andere ID vorhanden ist.
- dass es auf dem Message Level mindestens ein ST/SE-Paar gibt.
- dass ISA/I01 und ISA/I03 vorhanden sind. Wenn sie keine Daten enthalten, wird 00 eingesetzt.
- dass ISA/I02 und ISA/I04 vorhanden sind. Wenn sie keine Daten enthalten, werden zehn Leerstellen eingesetzt.
- dass beide ISA/I05 Segmente vorhanden sind. Wenn sie keine Daten enthalten, wird ZZ eingesetzt.
- dass ISA/I08 vorhanden ist. Wenn es keine Daten enthält, wird das aktuelle Datum im EDI-Format eingesetzt.
- dass ISA/I09 vorhanden ist. Wenn es keine Daten enthält, wird die aktuelle Uhrzeit im EDI-Format eingesetzt.
- dass ISA/I65 vorhanden ist. Wenn es keine Daten enthält, wird das Wiederholungstrennzeichen eingesetzt.
- dass ISA/I11 vorhanden ist. Wenn es keine Daten enthält, wird die Interchange Control Version Number aus den benutzerdefinierten Einstellungen eingesetzt. (siehe Einstellung **Erweitert | Interchange Control Versionsnummer** der EDI-Komponenteneinstellungen[466]).
- dass ISA/I12 vorhanden ist.
- dass ISA/I13 vorhanden ist. Wenn es keine Daten enthält, wird die Request Acknowledgement-Einstellung verwendet. (siehe Einstellung **Erweitert | Empfangsbetätigung anfordern** der EDI-Komponenteneinstellungen[466]).
- dass ISA/I14 vorhanden ist. Wenn es keine Daten enthält, wird P eingesetzt.
- dass ISA/I15 vorhanden ist. Wenn es keine Daten enthält, wird das Composite-Trennzeichen aus den benutzerdefinierten Einstellungen eingesetzt (siehe Einstellung **Composite-Trennzeichen** der EDI-Komponenteneinstellungen[466]).
- dass IEA/I16 vorhanden ist. Wenn es keine Daten enthält, wird die Anzahl der Funktionsgruppen im Interchange berechnet und eingesetzt.
- dass IEA/I12 vorhanden ist. Wenn es keine Daten enthält, wird der Wert aus ISA/I12 kopiert.

© 2016-2022 Altova GmbH

Bitte beachten Sie:

Felder, die hier nicht erwähnt sind, werden nicht eingefügt oder erstellt. Die korrekten Werte können nicht automatisch ermittelt werden. Bei einem Parent-(Ziel)-Element A (in der EDI-Zielkomponente) mit den Child-Elementen x, y und z - wobei y obligatorisch ist - wird das Parent-Element A in der Ausgabedatei nur dann erstellt, wenn das obligatorische Child-Element "y" in der Zielkomponente gemappt wurde.

Potentiell existierende Funktionsgruppen werden auf folgenden Punkte überprüft:

- dass GS/373 vorhanden ist. Wenn es keine Daten enthält, wird das aktuelle Datum im EDI-Format eingefügt.
- dass GS/337 vorhanden ist. Wenn es keine Daten enthält, wird die aktuelle Uhrzeit im EDI-Format eingesetzt.
- dass GE/97 vorhanden ist. Wenn es keine Daten enthält, wird die Anzahl der Messages in der Funktionsgruppe berechnet und eingesetzt.
- dass GE/28 vorhanden ist. Wenn es keine Daten enthält, wird der Wert aus GS/28 kopiert.

Bei der Überprüfung von Messages wird sichergestellt:

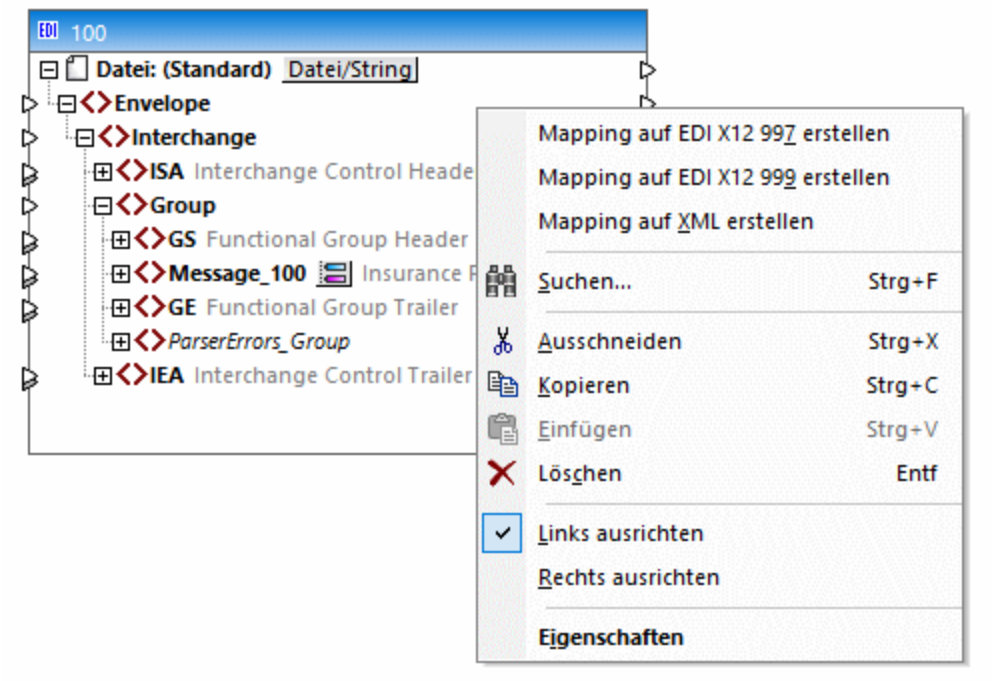
- dass ST/143 vorhanden ist. Wenn es keine Daten enthält, wird der Name der Message eingefügt.
- dass SE/96 vorhanden ist. Wenn es keine Daten enthält, wird die Anzahl der Segmente in der Message berechnet und eingesetzt.
- dass ST/329 und SE/329 vorhanden sind. Wenn SE/329 keine Daten enthält, wird der Wert aus ST/329 kopiert.

4.3.7.3 Generieren eines X12 997 Functional Acknowledgment

Das X12 997 Functional Acknowledgement gibt Auskunft über den Status des EDI-Datenaustauschs. Es enthält eine Liste aller Fehler, die bei der Verarbeitung des Dokuments aufgetreten sind. MapForce kann automatisch eine X12 997-Komponente im Hauptmappingbereich erstellen und die nötigen Konnektoren automatisch verbinden.

So generieren Sie das EDI 997 Functional Acknowledgment:

1. Klicken Sie im Menü **Einfügen** auf **EDI** und fügen Sie die X12-Komponente zum Mapping hinzu. Wenn Sie aufgefordert werden, eine Instanzdatei auszuwählen, navigieren Sie zu der Datei, für die Sie das X12 997 Acknowledgement erstellen möchten.
2. Klicken Sie mit der rechten Maustaste auf die Titelleiste der EDI-Quellkomponente und wählen Sie den Befehl "**Mapping auf EDI X12 997 erstellen**" (Beachten Sie, dass dieser Befehl nur bei X12-Komponenten aktiv ist).



Dadurch wird eine EDI 997-Mapping-Komponente erstellt und die zur Generierung des erforderlichen X12 997 Acknowledgement erforderlichen Datenelemente werden automatisch verbunden. Ein Beispiel dazu finden Sie in der Mapping-Design-Datei **X12_To_XML_Order.mfd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Um die mit diesem Mapping erstelle Acknowledgment-Datei zu sehen, klicken Sie auf die **Vorschauschaltfläche** ( ) der EDI 997-Komponente und anschließend auf das Register **Ausgabe**.

```
1  ISA+00+          +00+          +ZZ+ReceiverID    +ZZ+SenderID      +100128+1113+!+00505+000000000+1+P+:'
2  GS+01+ReceiverID+SenderID+20100128+11130544+1+X+05012'
3  ST+997+0001'
4  AK1+PO+1+05012'
5  AK2+850+12345'
6  AK5+A'
7  AK9+A+1+1+1'
8  SE+6+0001'
9  GE+1+1'
10 IEA+1+000000000'
11
```

So speichern Sie die X12 997 Acknowledgement-Datei:

- Klicken Sie auf das Register **Ausgabe** und anschließend auf die Symbolleisten-Schaltfläche **Generierte Ausgabe speichern** ( ). Wenn Sie vorhaben, anhand des Mappings Programmcode zu generieren, geben Sie einen Dateinamen in das Feld "EDI-Output-Datei" des Dialogfelds "Komponenteneinstellungen" ein (siehe EDI-Komponenteneinstellungen 466).

 © 2016-2022 Altova GmbH

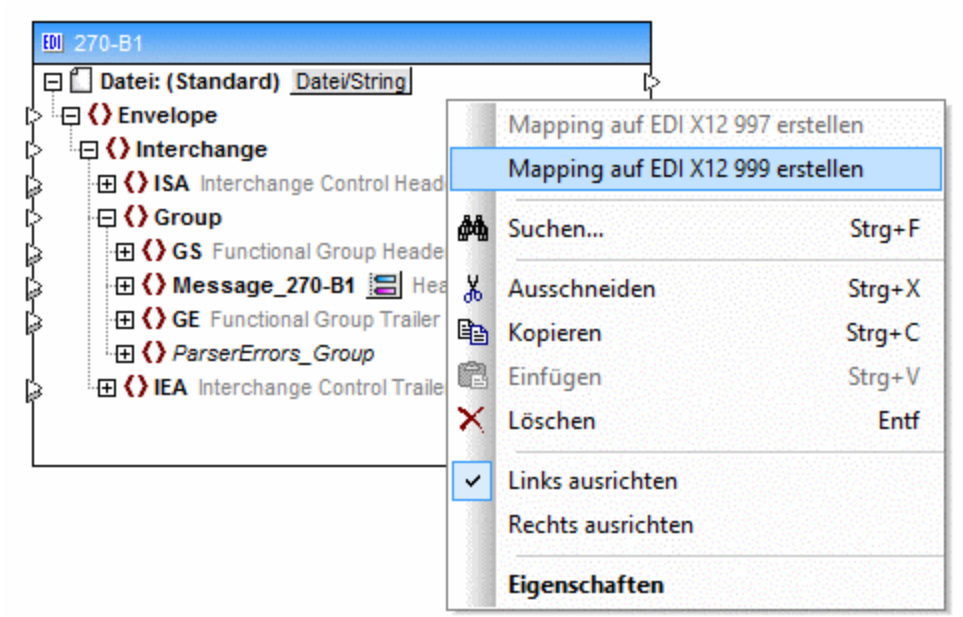
4.3.7.4 Generieren eines X12 999 Implementation Acknowledgment

Das X12 999 Implementation Acknowledgment meldet die Nichterfüllung des X12 Implementation Guide oder Applikationsfehler. Da es sich hierbei um eine Obermenge des 997 Functional Acknowledgement handelt, kann 999 anstelle von 997 verwendet werden, um Transaction Sets auf Basis entweder der Syntaxanforderungen der X12- oder der HIPAA-Implementierungsrichtlinien zu akzeptieren oder abzulehnen.

Alle Fehler, die bei der Verarbeitung des Dokuments auftreten, werden darin ausgegeben (z.B. "Required Segment Missing", "Required Data Element Missing", "Code Value Not Used in Implementation", usw.). MapForce kann automatisch eine X12 999-Komponente im Bereich des Hauptmappings generieren und die nötigen Mapping-Verbindungen automatisch erstellen.

So generieren Sie das EDI 999 Implementation Acknowledgment:

1. Klicken Sie im Menü **Einfügen** auf **EDI** und fügen Sie die X12- oder HIPAA-Komponente zum Mapping hinzu. Wenn Sie aufgefordert werden, eine Instanzdatei auszuwählen, navigieren Sie zu der Datei, für die Sie das X12 999 Acknowledgement erstellen möchten.
1. Klicken Sie mit der rechten Maustaste auf die EDI-Quellkomponente und wählen Sie den Befehl "**Mapping auf EDI X12 999 erstellen**". (Beachten Sie, dass dieser Befehl nur bei X12- und HIPAA-Komponenten aktiv ist).



Daraufhin wird eine EDI 999-Komponente erstellt und die zum Generieren des X12 999 Implementation Acknowledgment benötigten Datenelemente werden automatisch verbunden. Um eine Vorschau auf die generierte Acknowledgment-Datei zu sehen, klicken Sie auf das Register **Ausgabe**. Wenn das Mapping mehrere Zielkomponenten hat, klicken Sie zuerst auf die **Vorschau**-Schaltfläche ( ) und anschließend auf das **Ausgaberegister**.

```
1   ISA+00+       +00+       +ZZ+ReceiverID    +ZZ+SenderID      +110502+1533+!+00505+000000000+1+P+:'
2   GS+FA+ReceiverID+SenderID+20110502+15333751+1+X+006020'
3   ST+999+0001'
4   AK1+PO+1+006020'
5   AK2+850+12345'
6   IK5+A'
7   AK9+A+1+1+1'
8   SE+6+0001'
9   GE+1+1'
10  IEA+1+000000000'
11
```

So speichern Sie die 999 Acknowledgement-Datei:

- Klicken Sie auf das Register **Ausgabe** und anschließend auf die Symbolleisten-Schaltfläche **Generierte Ausgabe speichern** ( ). Wenn Sie vorhaben, anhand des Mappings Programmcode zu generieren, geben Sie einen Dateinamen in das Feld "EDI-Output-Datei" des Dialogfelds "Komponenteneinstellungen" ein (siehe EDI-Komponenteneinstellungen 466).

4.3.8 HIPAA X12

HIPAA X12 ist die neueste Version des Standards für elektronische Gesundheitsdaten, der vom US Department of Health and Human Services basierend auf EDI X12 Version 5010 zum Zweck der elektronischen Übertragung medizinischer Daten zwischen Versicherern, Gesundheitsdienstleistern und Arbeitgebern ins Leben gerufen wurde.

MapForce unterstützt die neueste Version, A2, der HIPAA-Implementierungsspezifikation (TR3). Ältere Versionen können in Form separater ZIP-Dateien von der Altova Website heruntergeladen werden.

HIPAA-Komponenten ähneln gewöhnlichen ANSI X12-Komponenten und MapForce unterstützt die folgenden Transactions:

X12-Name	Nachrichtenname
X279A1	"Health Care Eligibility Benefit Inquiry (270)"
X279A1	"Health Care Eligibility Benefit Response (271)"
X212	"Health Care Claim Status Request (276)"
X212	"Health Care Information Status Notification (277)"
X214	"Health Care Claim Acknowledgment (277)"
X217	"Health Care Services Review - Request for Review (278)"
X217	"Health Care Services Review - Response (278)"
X218	"Payroll Deducted and Other Group Premium Payment for Insurance Products (820)"

© 2016-2022 Altova GmbH

X220A1	"Benefit Enrollment and Maintenance (834)"
X221A1	"Health Care Claim Payment/Advice (835)"
X222A1	"Health Care Claim: Professional (837)"
X224A2	"Health Care Claim: Dental (837)"
X223A2	"Health Care Claim: Institutional (837)"
X231A1	"Implementation Acknowledgment For Health Care Insurance (999)"

Unterschiede zur Behandlung von Standard X12-Nachrichten in MapForce:

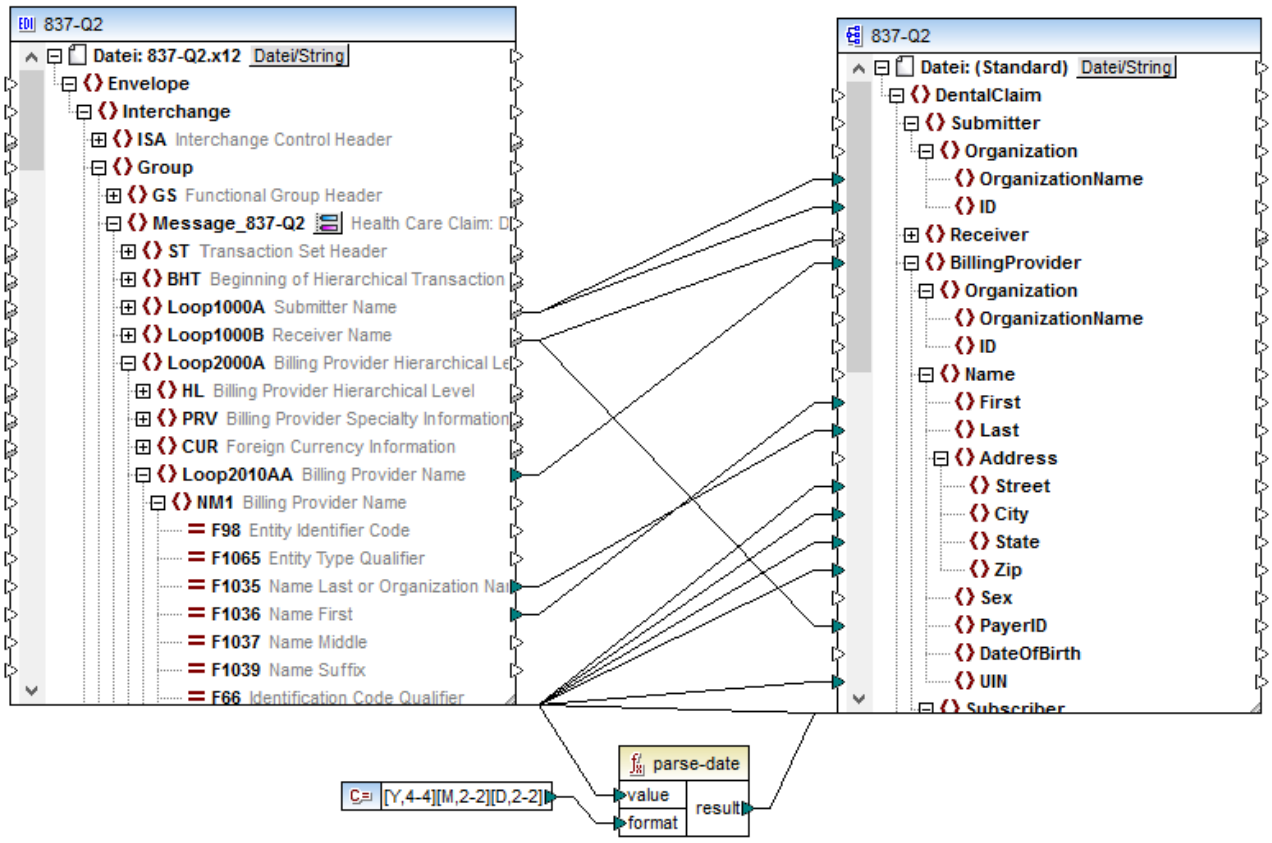
- Die Hierarchie von HL-Segmenten wird beibehalten.
- So genannte "floating structures" (in 837-Nachrichten) werden unterstützt.
- MapForce komplettiert und validiert automatisch mehrere Felder.

MapForce unterstützt außerdem die automatische Generierung des X12 999 Implementation Acknowledgment, das dem bestehenden 999-Implementation Acknowledgement des X12-Standards ähnlich ist (siehe Generieren eines X12 999 Implementation Acknowledgment 483).

Außerdem unterstützt MapForce bei HIPAA-Komponenten mehrere Nachrichten pro Nachrichtenaustausch (d.h. EDI-Komponente).

4.3.8.1 Beispiel: Mappen von HIPAA X12 auf XML

In diesem Beispiel wird gezeigt, wie Sie eine Datei, die eine "837-Q2 Health Care Claim: Dental" Transaktion enthält, auf eine XML-Zieldatei mappen. Das Mapping dazu finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\HIPAA_837D.mfd**.



HIPAA_837D.mfd

Das Mapping wird, wie unten beschrieben, erstellt.

Schritt 1: Hinzufügen der HIPAA X12-Quelldatei zum Mapping

1. Wählen Sie die Menüoption **Einfügen | EDI**.
1. Wählen Sie die Collection "HIPAA.X12" und den Nachrichtentyp "837-Q2 Health Care Claim: Dental".
2. Wenn Sie aufgefordert werden, eine Instanzdatei bereitzustellen, öffnen Sie die Quelldatei aus dem folgenden Ordner: **<Dokumente>\Altova\MapForce2023\MapForceExamples\837-Q2.x12**.
3. Passen Sie die Einstellungen der neu hinzugefügten HIPAA X12-Komponente gegebenenfalls an (Nähere Informationen dazu finden Sie unter EDI-Komponenteneinstellungen 466).

Schritt 2: Hinzufügen des Zielschemas zum Mapping

1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und öffnen Sie die Datei **837-Q2.xsd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**.
2. Wenn Sie aufgefordert werden, eine XML-Beispieldatei anzugeben, klicken Sie auf **Überspringen** und wählen Sie als Root des Zieldokuments **DentalClaim** aus.

 © 2016-2022 Altova GmbH

Schritt 3: Hinzufügen einer Funktion zum Formatieren des Datums und Ziehen der Mapping-Verbindungen

In der HIPAA-Quelldatei enthält das DMG-Segment ein Datum im Format **19490501**. Um dieses Daten in ein passendes Format für die XML-Zieldatei **1949-05-01** zu konvertieren, fügen wir eine Funktion zur Konvertierung des Datums hinzu:

1. Ziehen Sie die Funktion **parse-date**[1044] aus dem Fenster "Bibliotheken" in das Mapping. Nähere Informationen zu Funktionen finden Sie unter Hinzufügen einer Funktion zum Mapping[943].
2. Klicken Sie im Menü **Einfügen** auf **Konstante** und geben Sie als Konstantenwert **[Y,4-4][M,2-2][D,2-2]** ein.
3. Ziehen Sie die folgenden Verbindungen:

Von	Zu	Aufgabe
Konstante	`format` Input-Element der Funktion **parse-date**	Stellt der Funktion das Format bereit, anhand dessen das Datum geparst werden soll.
`F1251`	`value` Input-Element der Funktion **parse-date**	Liefert den tatsächlich zu formatierenden Wert an die Funktion.
`result` Output-Element der Funktion **parse-date**	`DateOfBirth` Input-Element der XML-Zieldatei	Liefert den von der Funktion zurückgegebenen Wert an den XML-Node `DateOfBirth`.

4. Ziehen Sie die Mapping-Verbindungen, wie im Beispiel-Mapping gezeigt (Dies sind zu viele, um hier alle aufgelistet zu werden; im Prinzip werden für jede Verbindung Werte aus der HIPAA-Quelldatei in die XML-Zieldatei kopiert. Grundlegende Informationen zu Mapping-Verbindungen finden Sie unter Arbeiten mit Verbindungen[89].)

4.3.8.2 Beim Mappen von HIPAA-Transaktionen zu beachten

HL-Segmenthierarchie

HL-Segmente werden automatisch generiert und müssen vom Benutzer nicht manuell gemappt werden. Allerdings können bei Bedarf Werte gemappt werden.

Autokomplettierung

Die Konfigurationsdateien für HIPAA-Transactions enthalten zahlreiche Codes und werden, wo möglich, zur Autokomplettierung von Feldern verwendet. Die Autokomplettierung ist kontextsensitiv, daher können Felder an verschiedenen Stellen unterschiedliche Werte haben. Die Autokomplettierungsfunktion kann durch Deaktivieren des Kontrollkästchens **Autokomplettierung von fehlenden Feldern** im Dialogfeld "Komponenteneinstellungen" ausgeschaltet werden (siehe EDI-Komponenteneinstellungen[466]).

Validierung

Die Validierung ist der Autokomplettierung insofern ähnlich, als sie ebenfalls kontextsensitiv erfolgt. Validierungsfehler werden auf dieselbe Art wie bei der X12-Validierung behandelt (siehe ASC X12-Validierungsregeln 479). .

Bearbeiten von Konfigurationsdateien

HIPAA-Konfigurationsdateien basieren auf einer neueren Schema-Version als X12-Konfigurationsdateien (siehe EDI-Konfigurationsschemas 543). HIPAA-Konfigurationsdateien ähneln daher X12-Konfigurationsdateien, weisen aber zusätzlich folgende Unterschiede auf:

- Codewerte auf Feldebene
- Bedingungen
- Komplettierungs-Flags
- “Not Used” Datenelemente

Codewerte

Feldinstanzen können eine integrierte Codeliste haben, die zur Validierung und/oder Autokomplettierung herangezogen wird. Werte können manuell hinzugefügt oder entfernt werden (siehe Ändern der möglichen Codewerte von Datenelementen 556)..

```
<Data ref="F365" info="Communication Number Qualifier" nodeName="F365_1">
<Values>
        <Value Code="EM" />
        <Value Code="FX" />
        <Value Code="TE" />
</Values>
</Data>
```

Bedingungen

Bedingungen werden in Verbindung mit Werten angewendet. Da ein Segment oder eine Gruppe mehrmals vorkommen kann (Schleife), werden diese Instanzen durch eine spezielle Bedingung identifiziert. Damit eine Bedingung erfüllt wird, muss ein Wert vorhanden sein, sonst wird die spezifische Gruppe nicht gefunden.

So kommt z.B. Loop1000 mehrmals vor, doch hat jede Instanz davon eine andere semantische Bedeutung.

Um zu erkennen, um welche Schleife es sich handelt, definiert MapForce mit Hilfe einer Bedingung, dass Loop1000A im Feld F98 von Segment NM1 einen Wertcode “41” haben muss. Wenn dies nicht der Fall ist, wird der Gruppenname Loop1000A nicht gefunden.

```
<Group name="Loop1000A" info="Submitter Name">
  <Segment name="NM1" info="Submitter Name">
    <Condition path="F98" />
    <Data ref="F98" info="Entity Identifier Code">
        <Values>
          <Value Code="41" />
        </Values>
    </Data>
(…)
```


© 2016-2022 Altova GmbH

Bedingungscodes werden in der Zielkomponente automatisch generiert, wenn sie einen einzigen Wert enthalten.

Konfigurationsdateien werden mit einer Methode generiert, die gewährleistet, dass Bedingungswerte innerhalb der Sequenz sich wiederholender Segmente/Schleifen garantiert eindeutig sind.

Wenn Bedingungen oder ihre Werte bearbeitet werden müssen, muss diese Voraussetzung der Eindeutigkeit unbedingt berücksichtigt werden.

Komplettierungs-Flags

Die Autokomplettierung kann auf Ebene der Konfigurationsdatei angepasst werden. Ein neues Element "Completion" mit drei Attributen wurde zur Definition der folgenden Dinge hinzugefügt:

- singleConditions – Einzelbedingungen für alle Felder in der Zielkomponente automatisch komplettieren
- singleValues – Einzelwerte in der Zielkomponente für alle Felder automatisch komplettieren
- HL – entsprechende Felder (für alle HL-Segmente) in der Zielkomponente im HL7-Segment generieren

  (wobei "1" für "true" steht)

```
<Message>
  <MessageType>837-Q2</MessageType>
  <Completion singleConditions="1" singleValues="1" HL="1" />
  <Description>Health Care Claim: Dental</Description>
(...)
```

"Not Used"-Datenelemente

In HIPAA wird eine Reihe optionaler Elemente, die in Standard X12 Transactions vorhanden sind, weggelassen. Diese optionalen Elemente werden in MapForce-Komponenten versteckt.

Der Validierungsprozessor überprüft diese nicht verwendeten Felder, um sicher zu stellen, dass diese in Quelldateien nicht vorhanden sind. Da diese Felder nicht auf die Zielkomponente gemappt werden können (da sie versteckt werden), können sie auch nicht in Output-Zieldateien aufscheinen.

Für diese Felder wird in den Konfigurationsdateien das maxOccurs-Attribut "0" definiert. Falls nötig, können Sie mit Hilfe des maxOccurs-Attributs bestimmte Felder manuell verbergen (oder wieder einblenden).

```
<Data ref="F1037" minOccurs="0" info="Submitter Middle Name or Initial" />
<Data ref="F1038" minOccurs="0" maxOccurs="0" />
<Data ref="F1039" minOccurs="0" maxOccurs="0" />
```

Geltungsbereich der Validierung

Eine semantische Validierung wird von MapForce nicht unterstützt. Dies bedeutet, dass "Situationsfelder" einfach als optional behandelt werden.

4.3.9 HL7 Version 2

HL7 ist ein Industriestandard für den Austausch von Daten zwischen medizinischen Applikationen und steht für "Health Level Seven". MapForce unterstützt die Versionen 2.2 bis 2.6.

Als Standard-HL7-Version wird in MapForce 2.6 verwendet.

Auf der Seite MapForce-Komponenten der Altova-Website (https://www.altova.com/de/mapforce/download/libraries) steht ein eigener Installer für die zusätzlichen HL7 V2.2 - V2.5.1-XML-Schemas und Konfigurationsdateien zur Verfügung.

Die XML-basierte Version HL7 Version 3.x wird in MapForce 2023 auf Basis von XML-Schemakomponenten unterstützt.

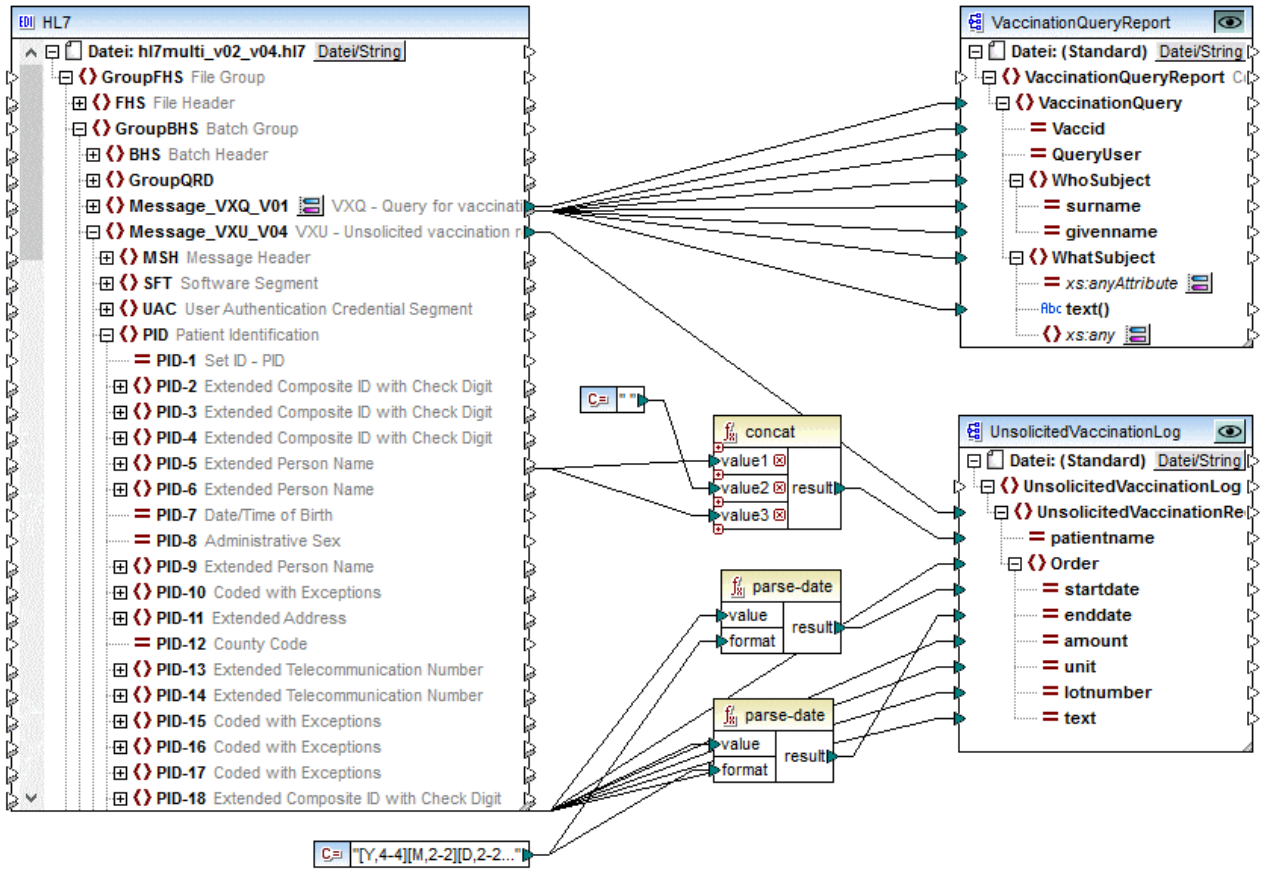
Das **MapForceExamples**-Projekt enthält eine Beispieldatei, in der eine HL7 V2.6 Datei auf eine HL7 V3 XML-Datei gemappt wurde (HL7V260_to_HL7V3.mfd).

Nähere Informationen zu HL7 finden Sie unter https://www.hl7.org/.

4.3.9.1 Beispiel: Mappen von HL7 auf XML

In diesem Beispiel wird gezeigt, wie Sie eine Datei, die zwei HL7-Nachrichten enthält ("Query for vaccination record" und "Unsolicited vaccination record update"), auf zwei separate XML-Zieldateien mappen. Das Mapping dazu finden Sie unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2023\MapForceExamples\HL7_MultiMessageTypes.mfd.

© 2016-2022 Altova GmbH



HL7_MultiMessageTypes.mfd

Das Mapping kann folgendermaßen erstellt werden:

Schritt 1: Hinzufügen der HL7-Datei zum Mapping

1. Klicken Sie im Menü **Einfügen** auf **EDI**.
2. Wählen Sie die Collection "HL7" aus und aktivieren Sie anschließend die Kontrollkästchen neben dem Nachrichtentypen "VXQ - Query for vaccination record" und "VXU - Unsolicited vaccination record update".
3. Wenn Sie aufgefordert werden, eine Instanzdatei bereitzustellen, öffnen Sie die folgende Quelldatei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\hl7multi_v02_v04.hl7**.
4. Wenn Sie dazu aufgefordert werden, passen Sie die Einstellungen der neu hinzugefügten HL7-Komponente an, falls nötig (Nähere Informationen dazu finden Sie unter EDI-Komponenteneinstellungen 466).

Schritt 2: Hinzufügen des Zielschemas zum Mapping

1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und öffnen Sie die Datei **VaccinationQueryReport.xsd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**.
2. Wenn Sie aufgefordert werden, eine XML-Beispieldatei anzugeben, klicken Sie auf **Überspringen.**

Gehen Sie vor, wie oben beschrieben, um die Datei **UnsolicitedVaccinationLog.xsd** als Mapping-Komponente hinzuzufügen.

Schritt 3: Hinzufügen von Funktionen und Ziehen der Mapping-Verbindungen

In der HL7-Quelldatei haben Datumsangaben das Format `JJJJMMTT`. Um solche Datumswerte in ein passendes Format `JJJJ-MM-TT` für die XML-Zieldatei zu konvertieren, fügen wir eine Datumskonvertierungsfunktion hinzu, wie folgt:

1. Ziehen Sie die Funktion **parse-date**[1044] aus dem Fenster "Bibliotheken" in das Mapping. Nähere Informationen zu Funktionen finden Sie unter Hinzufügen einer Funktion zum Mapping[943].
2. Klicken Sie im Menü **Einfügen** auf **Konstante** und geben Sie als Konstantenwert `[Y,4-4][M,2-2][D,2-2]` ein.
3. Ziehen Sie die folgenden Verbindungen:

Von	Zu	Aufgabe
Konstante	`format` Input-Element der Funktion **parse-date**	Stellt der Funktion das Format bereit, anhand dessen das Datum geparst werden soll.
`RXA-3`	`value` Input-Element der Funktion **parse-date**	Liefert den tatsächlich zu formatierenden Wert an die Funktion.
`result` Output-Element der Funktion **parse-date**	`startdate` Input-Element der XML-Zieldatei	Liefert den von der Funktion zurückgegebenen Wert an den XML-Node `startdate`.

Befolgen Sie dieselben Schritte, um eine neue **parse-date** Funktion hinzuzufügen, um den Wert von `RXA-4` zu konvertieren und in das Datenelement `enddate` der XML-Zieldatei zu kopieren.

Das Datenelement `patientname` im der zweiten XML-Zieldatei wird durch Verkettung zweier Felder der HL7-Quelldatei erstellt: `FN-1` (Nachname) und `XPN-2` (Vorname). Ziehen Sie dazu die Funktion **concat** aus dem Fenster "Bibliotheken" in das Mapping. Fügen Sie außerdem eine Konstante hinzu, die das Leerzeichen zum Trennen der beiden Namen liefert.

Ziehen Sie die verbleibenden Mapping-Verbindungen, wie im Beispielmapping gezeigt (Dies sind zu viele, um hier alle aufgelistet zu werden; im Prinzip werden für jede Verbindung Werte aus der HL7-Quelldatei in die XML-Zieldatei kopiert. Grundlegende Informationen zu Mapping-Verbindungen finden Sie unter Arbeiten mit Verbindungen[89].)

Um eine Vorschau auf die Ausgabe, die von einer beliebigen der XML-Zielkomponenten generiert wird, zu sehen, klicken Sie zuerst auf die Schaltfläche **Vorschau** () der Komponente und anschließend auf das Register **Ausgabe**.

4.3.10 HL7 Version 3

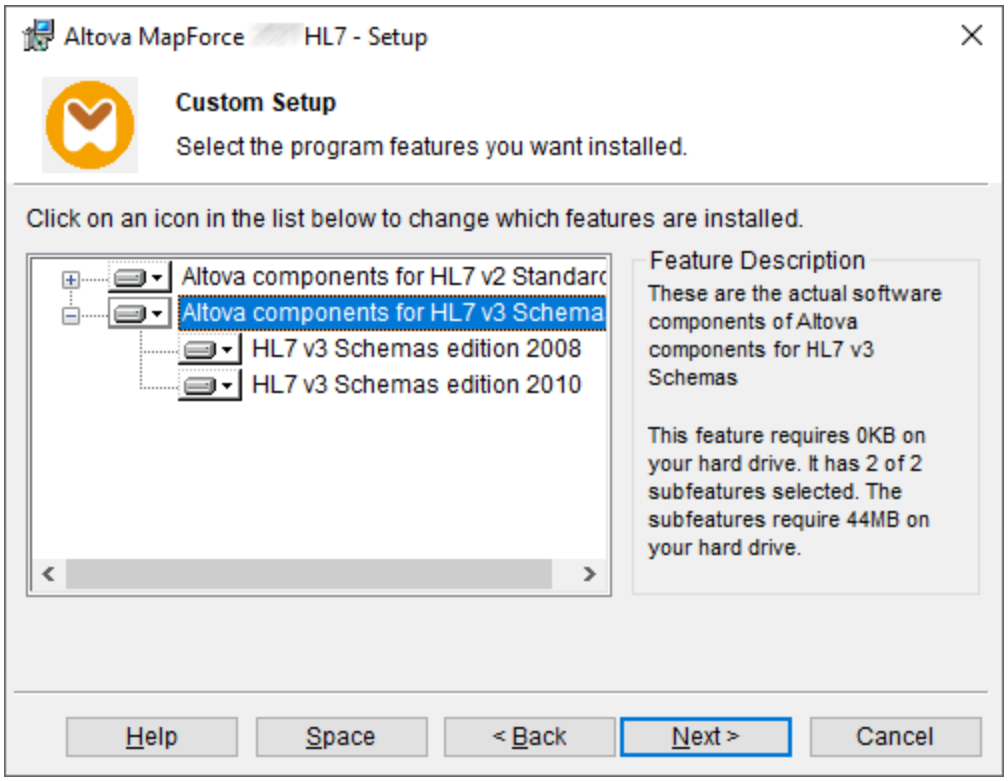
MapForce bietet Unterstützung für das Mappen von Daten von oder auf HL7 Version 3-Dokumente. Dabei können Sie Daten von jedem beliebigen von Ihrer MapForce Edition unterstützten Komponententyp auf HL7 3.x-Instanzdateien mappen oder umgekehrt.

© 2016-2022 Altova GmbH

Der HL7 Version 3-Standard basiert im Gegensatz zur HL7 Version 2 auf XML. Sie können daher HL7 Version 3-Komponenten auf dieselbe Art wie jedes andere XML-Schema zum Mapping hinzufügen. Beachten Sie, dass HL7 Version 3-Schemas standardmäßig nicht in MapForce installiert sind. Sie können die HL7 Version 3-Schemas jedoch separat herunterladen und installieren, wie weiter unten beschrieben.

So installieren Sie die HL7 Version 3-XML-Schemas:

1. Laden Sie das Paket namens "HL7 EDI Configuration Files" von der Seite "MapForce-Bibliotheken" aus dem Altova Download-Center herunter: https://www.altova.com/de/mapforce/download#mapforce_libraries. Beachten Sie, dass dieses Paket sowohl (EDI-basierte) HL7 Version 2-Konfigurationsdateien als auch HL7 Version 3-XML-Schemas enthält.
2. Doppelklicken Sie auf die heruntergeladene ausführbare Datei und befolgen Sie die Anweisungen des Installationsassistenten.
3. Wenn Sie aufgefordert werden, einen Setup-Typ auszuwählen, wählen Sie **Custom** und klicken Sie dann auf **Next**.
4. Wählen Sie z.B. die HL7 Version 3-Schemas aus:



Nachdem Sie die HL7 Version 3-XML-Schemas installiert haben, stehen diese im folgenden Verzeichnis zur Verfügung: **C:\Programme\Altova\Common2023\Schemas\hl7v3**.

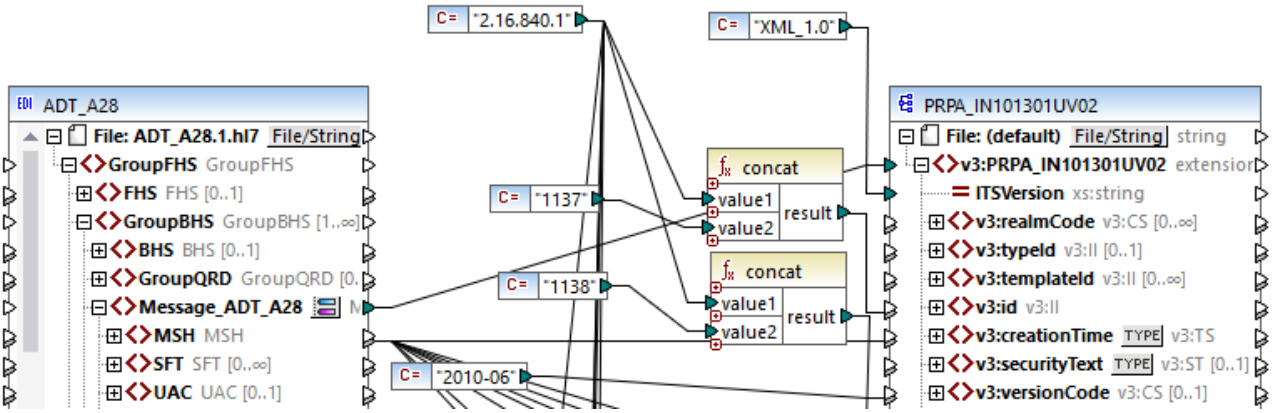
Anmerkung: Ändern Sie den obigen Pfad bei Verwendung einer 32-Bit-MapForce-Version auf einem 64-Bit-Betriebssystemen in **C:\Programme (x86)\Altova\Common2023\Schemas\hl7v3**.

Nachdem Sie die HL7 Version 3-XML-Schemas installiert haben, können Sie ein Mapping folgendermaßen beginnen:

1. Starten Sie MapForce.
2. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei**.
3. Navigieren Sie zur Schema-Datei. Sie können z.B. die **Datei "PRPA_IN101301UV02.xsd"** aus dem oben genannten Installationsverzeichnis auswählen.
4. Wenn Sie aufgefordert werden, die Instanzdatei anzugeben, wählen Sie die HL7 Version 3-Instanzdatei für den zuvor ausgewählten Nachrichtentyp aus, wenn Daten daraus ausgelesen und auf ein anderes Format gemappt werden sollen. Wenn Sie hingegen beabsichtigen, die Instanzdatei zu generieren (anstatt Daten daraus auszulesen), klicken Sie auf **Überspringen**.

Sie haben die HL7 Version 3-Komponente nun zum Mapping hinzugefügt und können nun die gewünschten Mapping-Verbindungen ziehen. Eine Einführung in die Erstellung von Mappings mit MapForce finden Sie in den Tutorials 43.

Unter dem folgenden Pfad finden Sie ein Beispielmapping, in dem Daten aus einer HL7 Version 2.6-Datei in eine HL7 Version 3-Datei konvertiert werden:
<Dokumente>\Altova\MapForce2023\MapForceExamples\HL7V260_To_HL7V3.mfd. Unten sehen Sie einen Ausschnitt aus diesem Mapping.



Im oben gezeigten Mapping ist die Quellkomponente eine HL7 Version 2.6-Instanzdatei. Um solche Komponenten zum Mapping hinzuzufügen, wählen Sie den Menübefehl **Einfügen | EDI** und wählen Sie anschließend die erforderlichen Nachrichtentypen au der Gruppe "HL7" aus. Es handelt sich hierbei um eine EDI-Komponente; Sie können die Instanzdatei and andere Einstellungen für diese Komponente durch Doppelklick auf die Titelleiste der Komponente anzeigen oder ändern, siehe auch EDI-Komponenteneinstellungen 466.

Die Zielkomponente ist eine HL7 Version 3-Instanzdatei. Es handelt sich hierbei, wie bereits erwähnt, um eine XML-Komponente. Sie können solche Komponenten über den Menübefehl **Einfügen | XML-Schema/Datei** zum Mapping hinzufügen. Informationen über alle Einstellungen, die Sie für XML-Komponenten vornehmen können, finden Sie unter XML-Komponenteneinstellungen 126.

4.3.11 IATA PADIS

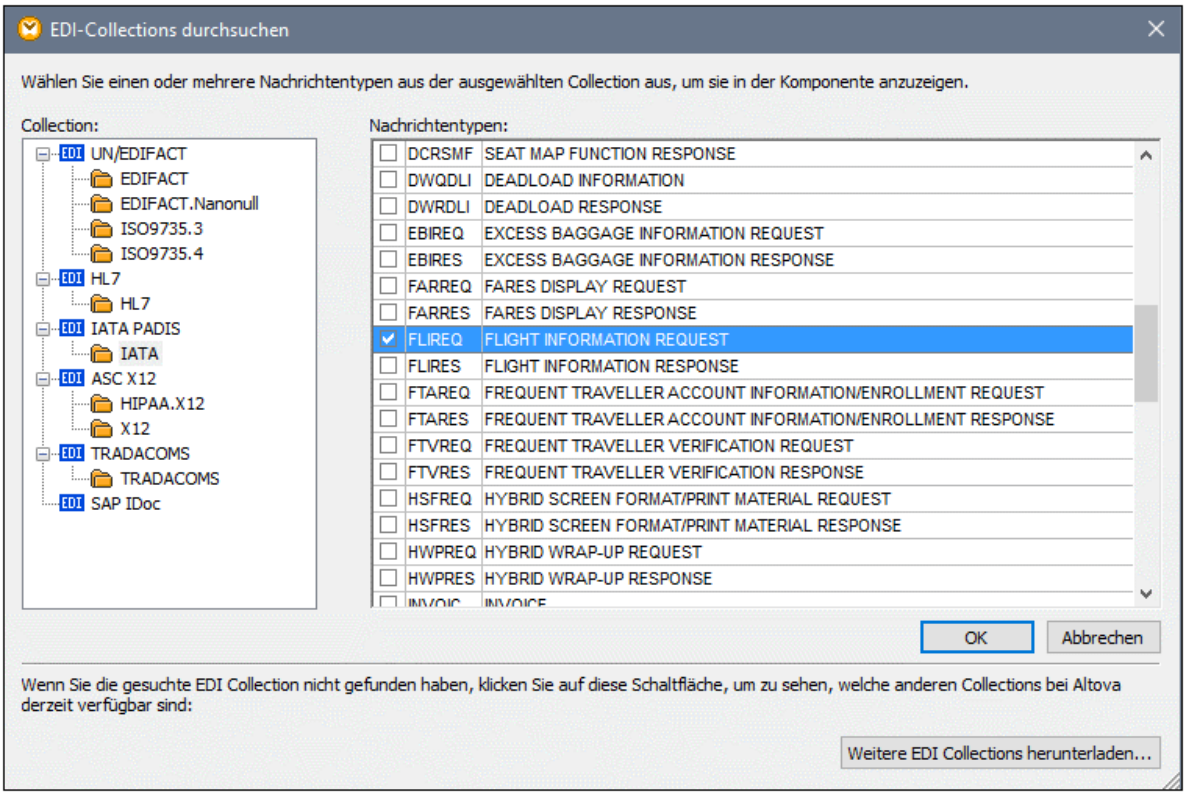
PADIS (Passenger and Airport Data Interchange Standards) ist eine Gruppe von Nachrichten, für die die EDIFACT (ISO 9735) Syntax verwendet wird.

© 2016-2022 Altova GmbH

- MapForce unterstützt derzeit nur Nachrichten, in denen die UNH/UNT Message Header und Trailer-Segmente verwendet werden.
- MapForce unterstützt die Collections IATA.00.1 bis IATA.08.1.

So fügen Sie eine IATA PADIS-Datei als Mapping-Komponente hinzu:

1. Klicken Sie im Menü **Einfügen** auf **EDI**.
2. Klicken Sie auf die "IATA" Collection unter "IATA PADIS" und aktivieren Sie das Kontrollkästchen neben dem/den gewünschten Nachrichtentyp(en).

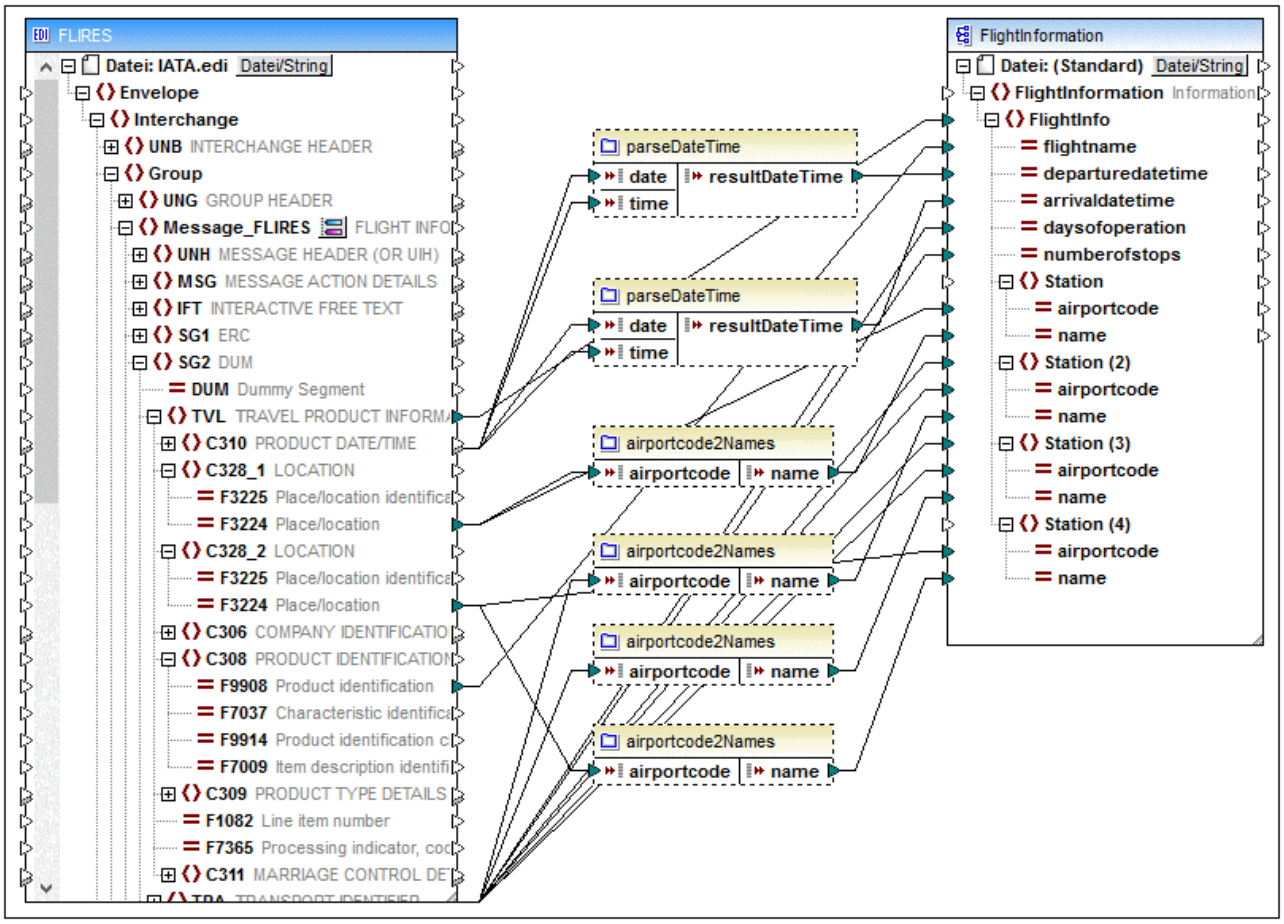


3. Wenn Sie aufgefordert werden, eine Instanzdatei anzugeben, klicken Sie auf **Durchsuchen** und öffnen Sie die gewünschte IATA-Quelldatei für das Mapping. Wenn Sie ein Mapping auf eine IATA PADIS-Komponente durchführen, klicken Sie auf **Überspringen**.

Die IATA-Komponente wird nun zum Mapping hinzugefügt und Sie können Mapping-Verbindungen von und zu dieser Komponente ziehen. Um die Einstellungen der IATA-Komponente zu ändern, klicken Sie mit der rechten Maustaste darauf und wählen Sie im Kontextmenü den Befehl **Eigenschaften** (siehe auch EDI-Komponenteneinstellungen 466).

4.3.11.1 Beispiel: Mappen von IATA auf XML

In der Datei **IATA_FlightInformationReport.mfd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** sehen Sie ein Beispiel für ein Mapping einer IATA PADIS-Datei auf eine XML-Zieldatei.



IATA_FlightInformationReport.mfd

Dieses Mapping besteht aus einer IATA-Quellkomponente, die eine FLIRES -Nachricht enthält, einer XML-Zielkomponente und zwei benutzerdefinierten Funktionen, **`parseDateTime`** und **`airportcode2Names`** (die mehrfach verwendet werden).

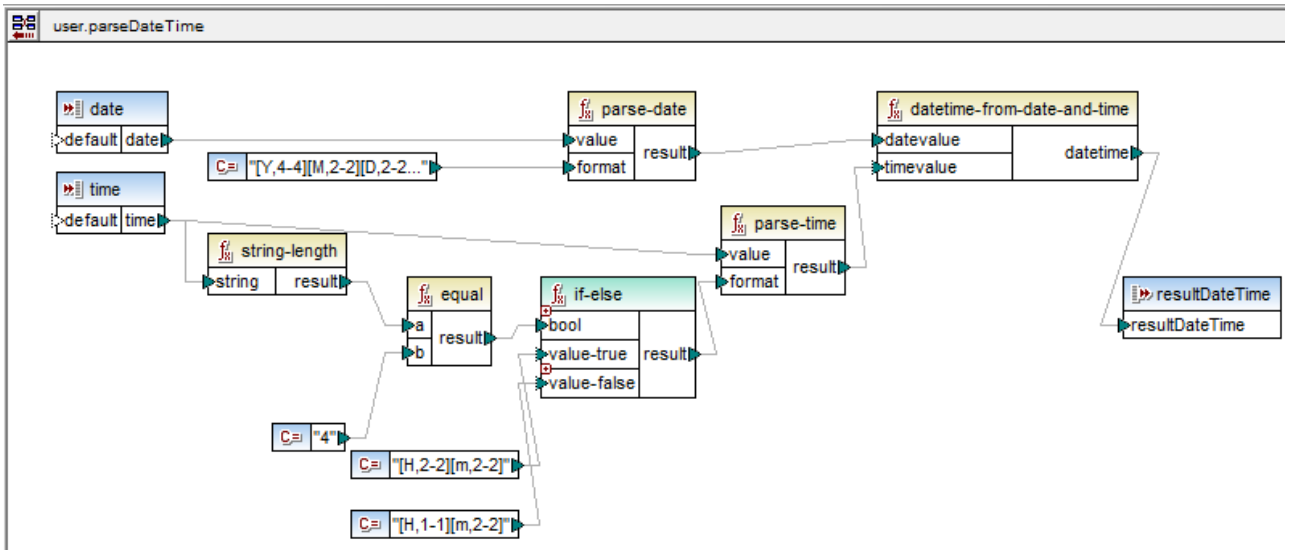
Um die von der Quell- oder Zielkomponente gelesene bzw. geschriebene Instanzdatei sowie andere Einstellungen zu sehen, klicken Sie mit der rechten Maustaste auf die Komponente und wählen Sie den Befehl **Eigenschaften** (siehe auch EDI-Komponenteneinstellungen 466).

Um die Struktur der einzelnen benutzerdefinierten Funktionen zu sehen, doppelklicken Sie auf die Funktion. Um zum Hauptmapping zurückzukehren, klicken Sie in der linken oberen Ecke des Mapping-Bereichs auf die Schaltfläche **Zurück zum Haupt-Mapping** ( ) Nähere Informationen dazu finden Sie unter Benutzerdefinierte Funktionen 967.

Die benutzerdefinierten Funktionen in diesem Mapping haben die folgenden Aufgaben:

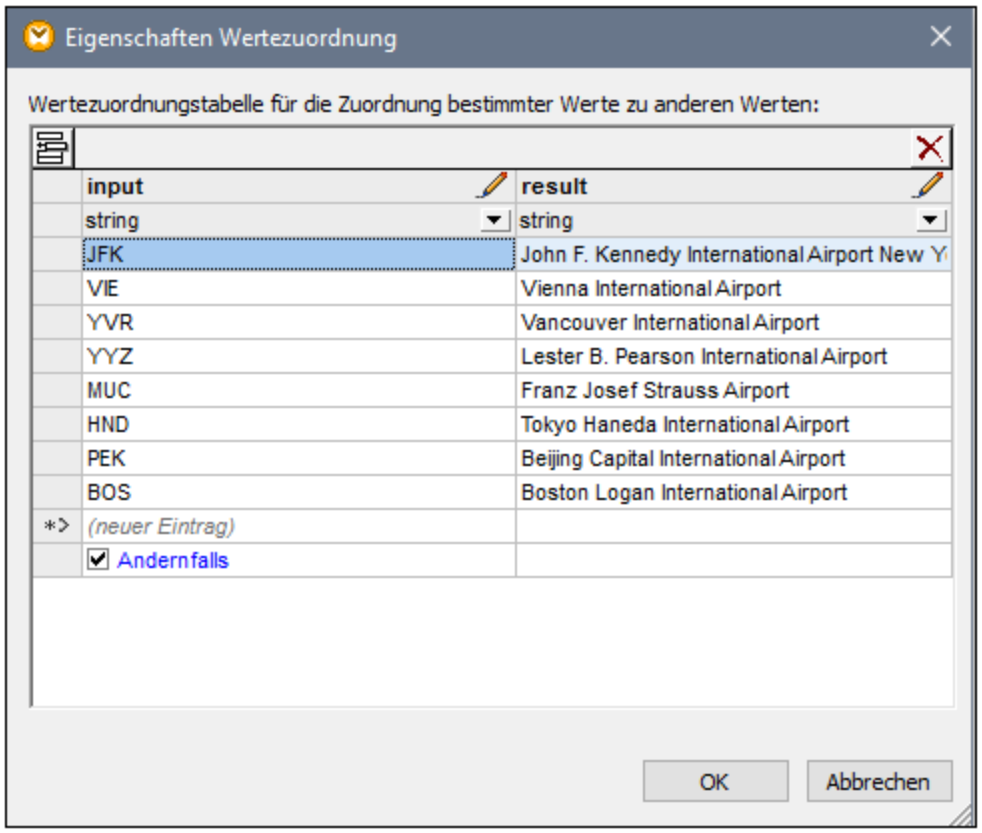
1) Die Funktion **`parseDateTime`** erhält als Argumente das Datum und die Uhrzeit aus den IATA-Datums- und Uhrzeitelementen und erzeugt anhand dieser Werte einen in der XML-Zieldatei erforderlichen `xs:dateTime`-Wert. Zu diesem Zweck werden in dieser Funktion verschiedene Funktionen aus der MapForce core-Bibliothek verwendet, wie die folgenden:

 © 2016-2022 Altova GmbH



Struktur der parseDateTime-Funktion

2) Die Funktion **`airportcode2Names`** erhält als Argument die Feldwerte F3223 oder F3224 (den Flughafencode) und gibt den vollständigen Namen des Flughafens zurück. Die Funktion enthält zu diesem Zweck eine Wertezuordnungskomponente, die die einzelnen Flughafencodes auf Flughafennamen mappt, wie in der folgenden Abbildung gezeigt:



Struktur der Wertezuordnungskomponente

Nähere Informationen zu Wertezuordnungskomponenten finden Sie unter Verwendung von Wertezuordnungen[921].

In der XML-Zielkomponente wird ein Verfahren namens Duplizierung von Nodes verwendet. In diesem Fall wird der Node `Station` in der Mapping-Komponente mehrmals dupliziert. Dadurch können Werte aus mehreren Quelldatenelementen auf dasselbe Zieldatenelement gemappt werden (siehe Input duplizieren[84]).

Um eine Vorschau auf die vom Mapping erzeugte Ausgabe zu sehen, klicken Sie auf das Register **Ausgabe**.

4.3.12 NCPDP SCRIPT

SCRIPT ist ein Standard für den elektronischen Datenaustausch im amerikanischen Gesundheitssektor. Er wurde vom NCPDP (National Council for Prescription Drug Programs) entwickelt, siehe http://www.ncpdp.org. MapForce unterstützt das Mappen von Daten zwischen Dateien im SCRIPT-Format und allen anderen von MapForce unterstützten Formaten, einschließlich XML, CSV, Microsoft Excel, Datenbanken und anderen. Unterstützt werden die NCPDP SCRIPT-Versionen 8.1, 10.6, 10.10.

Die NCPDP SCRIPT-Standardversion in MapForce ist 10.10.

© 2016-2022 Altova GmbH

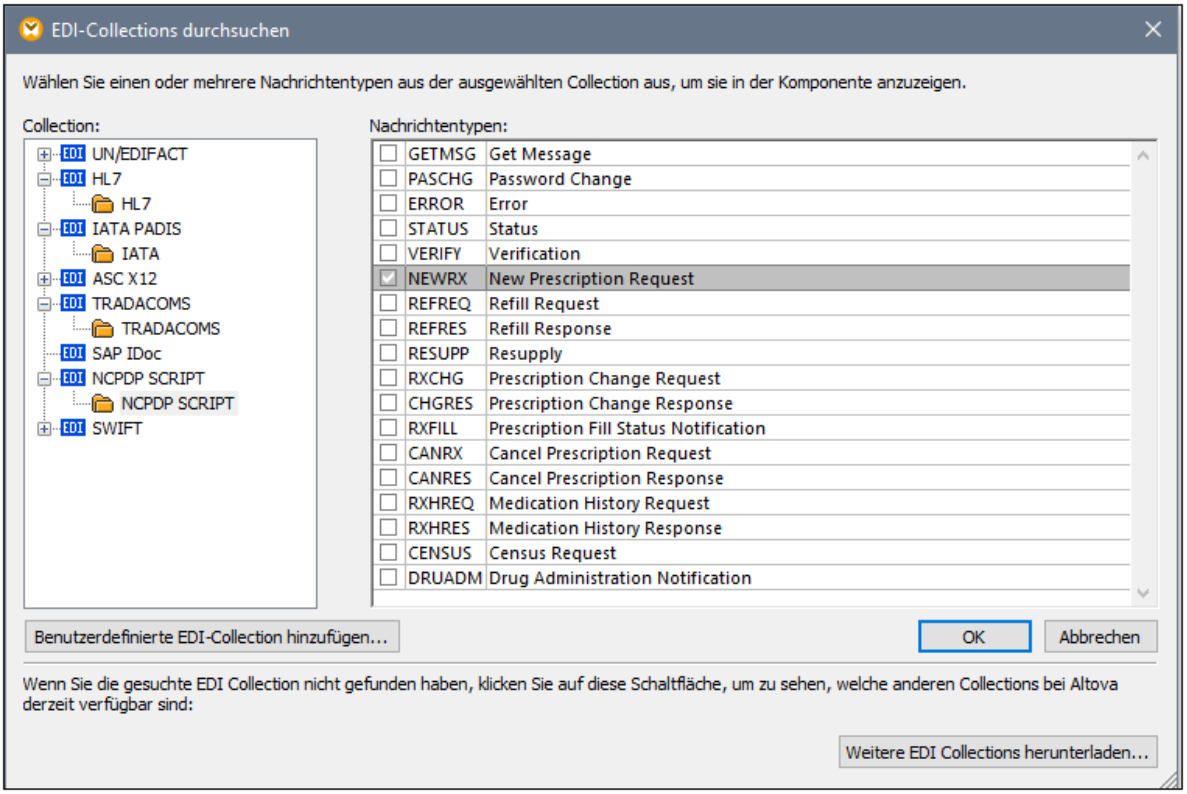
Anmerkung: Um die Unterstützung für eine andere NCPDP SCRIPT-Version als die Standardversion zu aktivieren, laden Sie das "NCPDP SCRIPT-Konfigurationsdateien"-Paket vom Altova-Download Center herunter (https://www.altova.com/de/mapforce/download/libraries) und installieren Sie es.

Für NCPDP SCRIPT kann entweder das XML- oder das UN/EDIFACT-Format verwendet werden. Diese Dokumentation gilt für die UN/EDIFACT-Version von NCPDP SCRIPT. Wenn Sie die NCPDP XML-Schemas haben, unterstützt MapForce unter Verwendung der XML-Standard-Mapping-Befehle und Validierung das Mappen von Daten von oder auf SCRIPT implizit (XML-Dateien können Sie über den Menübefehl **Einfügen | XML-Schema/Datei**) zum Mapping hinzufügen.

Beachten Sie, dass MapForce das Senden oder Empfangen von SCRIPT-Nachrichten als Teil der Datenkommunikation zwischen SCRIPT-Parteien nicht unterstützt. Die Aufgabe von MapForce ist es, Daten im SCRIPT-Format auszulesen und auf andere Formate zu mappen oder Daten aus anderen Formaten in das SCRIPT-Format zu konvertieren.

NCPDP SCRIPT-Dateien im UN/EDIFACT-Format können folgendermaßen zum Mapping hinzugefügt werden:

1. Klicken Sie im Menü **Einfügen** auf **EDI**.
2. Wählen Sie die "NCPDP SCRIPT" Collection aus und aktivieren Sie anschließend das Kontrollkästchen neben dem/den gewünschten Nachrichtentyp(en).



3. Klicken Sie auf OK. Sie werden nun aufgefordert, eine Instanzdatei anzugeben.
4. Wählen Sie eine der folgenden Methoden:
   - Wenn Sie Daten von einer NCPDP SCRIPT-Nachricht auf ein anderes Format mappen möchten, klicken Sie auf **Durchsuchen** und wählen Sie die SCRIPT-Quelldatei aus. Sie können Dateien auch über eine URL öffnen, siehe Hinzufügen von Komponenten über eine URL 80.

- Wenn Sie Daten auf eine NCPDP SCRIPT-Nachricht mappen möchten oder die Quelldatei zu einem späteren Zeitpunkt hinzufügen möchten, klicken Sie auf **Überspringen**.

Die Struktur der ausgewählten NCPDP SCRIPT-Nachricht(en) wird zu diesem Zeitpunkt im Mapping angezeigt (in der MapForce-Terminologie als eine "EDI-Komponente"). Sie können nun entweder Mapping-Verbindungen zu dieser Komponente ziehen (um Daten darauf zu mappen) oder von dieser Komponente aus Mapping-Verbindungen ziehen, um Daten von dieser aus zu mappen. Siehe auch:

- Hinzufügen von EDI-Komponenten zum Mapping 463
- Ausführen von EDI-Mappings 465
- EDI-Komponenteneinstellungen 466
- Validierung von EDI-Komponenten 471

4.3.12.1 NCPDP SCRIPT-Validierungsregeln

MapForce führt bei NCPDP SCRIPT-Nachrichten die folgenden Validierungsüberprüfungen durch:

- `UIB/S001/F0001` sollte "UNOA" sein.
- `UIB/S001/F0002` sollte "0" sein.
- `UIH/S306/F0329` sollte den Nachrichtentyp "SCRIPT" enthalten.
- `UIH/S306/F0316` sollte die in den ausgewählten Konfigurationsdateien definierte Nachrichtenversionsnummer enthalten.
- `UIH/S306/F0318` sollte die in den ausgewählten Konfigurationsdateien definierte Nachrichten-Release-Nummer enthalten.
- `UIH/S306/F0326` sollte die Nachrichtenfunktion (oder in der MapForce-Terminologie den Nachrichtentyp) enthalten
- `UIT/F0062` sollte mit dem entsprechenden `UIH/F0062`-Segment (Nachrichten-Header/Trailer) übereinstimmen oder nicht vorhanden sein (optionales Feld).
- `UIT/F0074` sollte die Segmentanzahl enthalten oder nicht vorhanden sein (optionales Feld).
- `UIZ/F0036` sollte die Nachrichtenanzahl enthalten oder nicht vorhanden sein (optionales Feld).

Siehe auch Validierung von EDI-Komponenten 471.

4.3.12.2 NCPDP SCRIPT-Autokomplettierungsregeln

MapForce erzeugt beim Generieren einer NCPDP SCRIPT-Datei gemäß den unten stehenden Regeln automatisch bestimmte Felder oder Werte (Diese Regeln werden als "Automatische Datenkomplettierung" oder "Autokomplettierungsregeln" bezeichnet). Um dieses Verhalten zu deaktivieren, deaktivieren Sie in den EDI-Komponenteneinstellungen 466 das Kontrollkästchen **Autokomplettierung von fehlenden Feldern**.

Die folgenden Felder werden automatisch vervollständigt:

Interchange Header

- `UIB/S001/F0001` = Kontrollagentur + Syntaxversionsnummer = "UNOA"
- `UIB/S001/F0002` = Syntaxversionsnummer = "0"
- `UIB/S002/F0004` = "Absender"
- `UIB/S003/F0010` = "Empfänger"

© 2016-2022 Altova GmbH

- `UIB/S300/F0017` = aktuelles Datum
- `UIB/S300/F0114` = aktuelle Uhrzeit

Message Header

- `UIH/S306/F0329` = "SCRIPT", Nachrichtentyp
- `UIH/S306/F0316` = "010" für 10.6, Nachrichtenversionsnummer aus den Konfigurationsdateien.
- `UIH/S306/F0318` = "006" für 10.6, Nachrichten-Release-Nummer aus den Konfigurationsdateien.
- `UIH/S306/F0326` = Nachrichtenfunktion (oder - in MapForce-Terminologie - Nachrichtentyp)
- `UIH/F0062` = "0", Nachrichtenreferenznummer

Message Trailer

- `UIT/F0062` = `UIH/F0062`
- `UIT/F0074` = Anzahl der Segmente in der Nachricht

Interchange Trailer

- `UIZ/F0036` = Nachrichtenanzahl

4.3.13 SAP IDoc

SAP IDoc (intermediate documents) Dokumente dienen zum Austauschen von Geschäftsdaten zwischen SAP und Nicht-SAP-Applikationen. Bei den Dokumenten handelt es sich um eine Form von Datenzwischenspeicher, das zwischen unterschiedlichen Systemen ausgetauscht werden kann.

Ein IDoc hat die folgende Struktur:

- **Kontrollsatz:** Enthält Kontrollinformationen über das IDoc: Absender, Empfänger, Nachrichtentyp und IDoc-Typ. Das Kontrollsatzformat ist für alle IDoc-Typen ähnlich.
- **Datensegment:** Enthält die eigentlichen Daten des Segments sowie andere Metadaten: Überschrift, Segmentnummer und Typ sowie die Felder, die die Daten enthalten.
- **Statussätze:** Enthalten Informationen zum aktuellen Status des Dokuments, d.h. die derzeit verarbeiteten Schritte und die Schritte, die noch verarbeitet werden müssen. Das Statusformat ist für alle IDoc-Typen identisch.

Die Versionsnummer in der Port-Definition definiert die Systeme, mit denen kommuniziert wird. Die Hauptunterschiede zwischen den Versionen sind die verschiedenen Namenslängen, die in verschiedenen Elementen verwendet werden, und die Verwendung von Erweiterungen. SAP R3 Version 4.X unterstützt lange Namen (sowie Erweiterungen), während die früheren Versionen dies nicht tun.

Port Version 1: Releases 2.1. und 2.2.
Port Version 2: Releases 3.0, 3.1 und R/2 Systeme.
Port Version 3: Release 4.x (Standardwert)

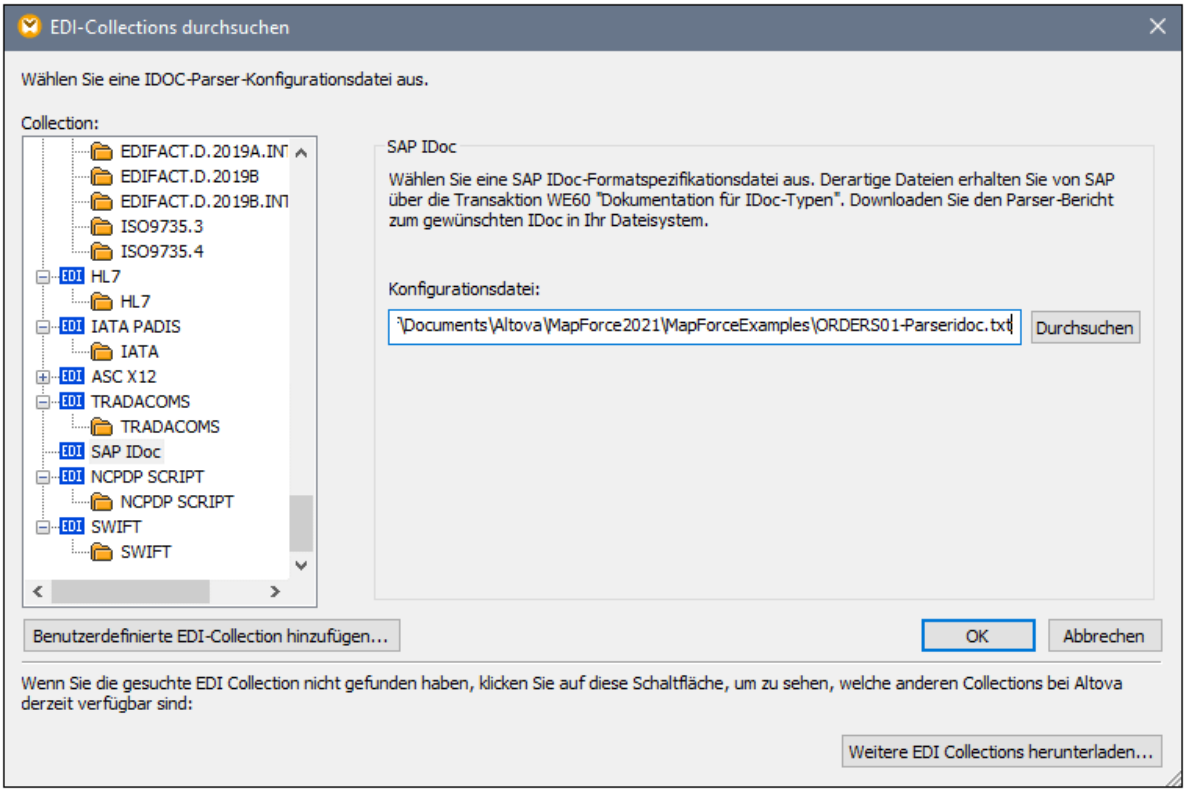
MapForce behandelt IDoc-Komponenten als Dateien mit einer festen Länge von 30 Zeichen für den Message Typ, 30 für den IDoc Typ und 27 für Segmentfelder.

In der Datei **<Dokumente>\Altova\MapForce2023\MapForceExamples\IDoc_Order.mfd** sehen Sie ein Beispiel für das Mapping einer SAP IDoc-Datei auf eine XML-Datei.

So fügen Sie ein SAP IDoc-Dokument als Mapping-Komponente ein:

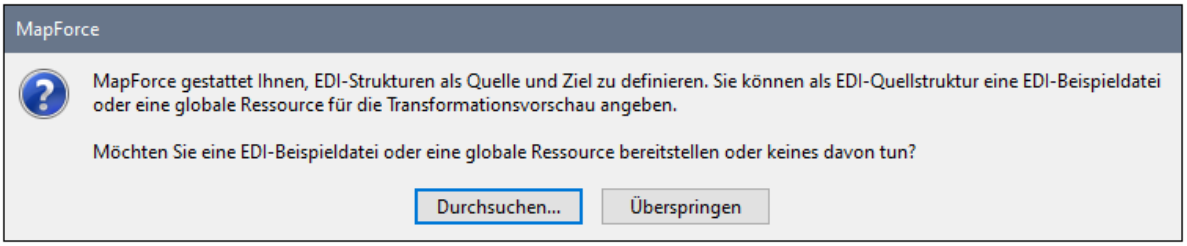
1. Wählen Sie die Menüoption **Einfügen | EDI**.
   Daraufhin wird das Dialogfeld "EDI-Collections durchsuchen" geöffnet.
2. Klicken Sie im Listenfeld auf den Eintrag **SAP IDoc**.
3. Klicken Sie auf die Schaltfläche **Durchsuchen** und wählen Sie den IDoc-Parser-Bericht (auch IDoc-Definitionsdatei genannt) aus.

> Der IDoc-Parser-Bericht wird anhand der Transaktion WE60 ("Documentation for IDoc types") aus dem SAP-System erstellt. Beachten Sie, dass die Datei im unkomprimierten Format aus SAP exportiert werden muss. Informationen darüber, wie Sie die Transaktion WE60 aufrufen und den Parser-Bericht generieren, finden Sie in der SAP IDoc-Dokumentation. Der Ordner MapForce Examples enthält eine Beispieldatei für einen Parser-Bericht (siehe **<Dokumente>\Altova\MapForce2023\MapForceExamples\ORDERS01-Parseridoc.txt**).

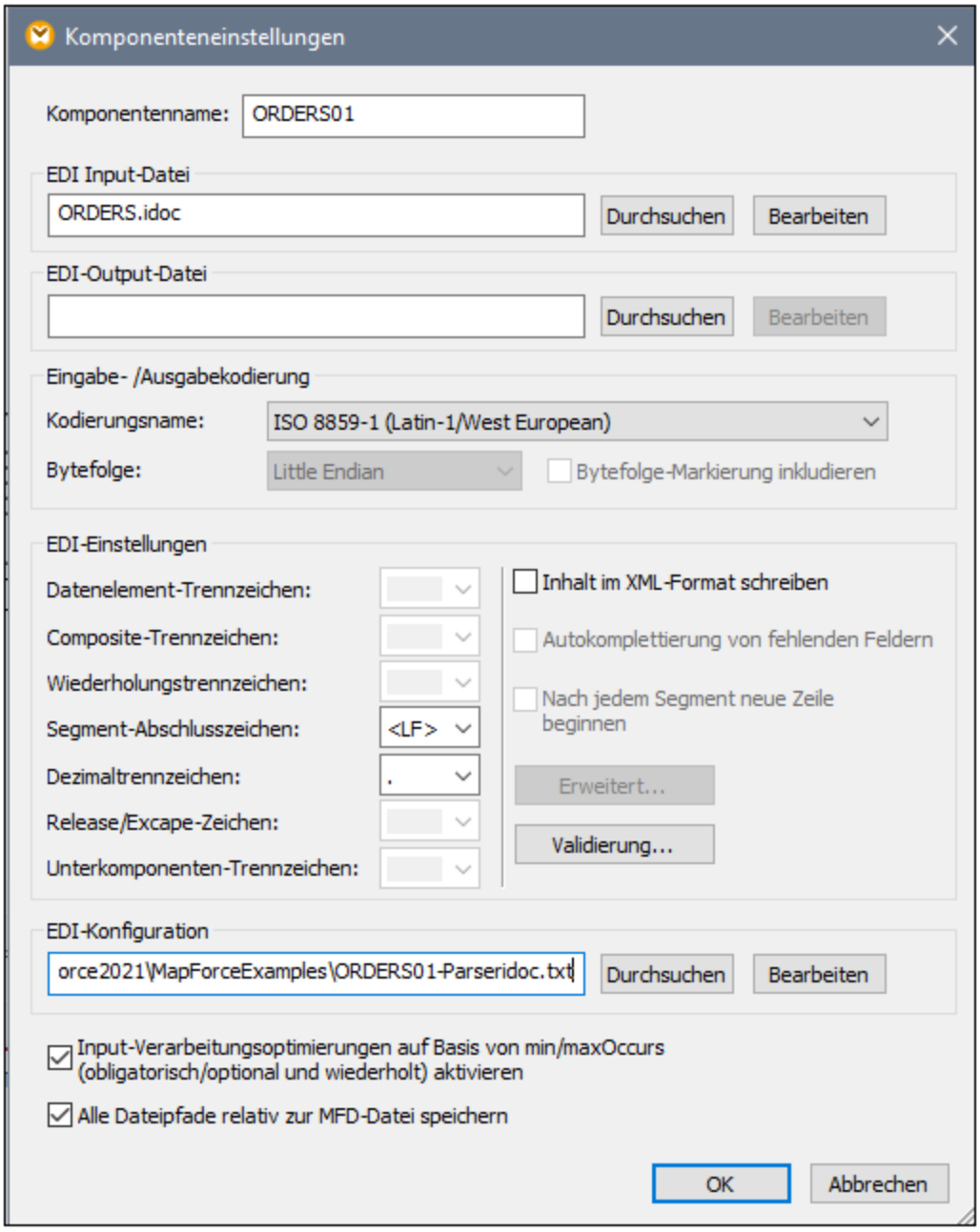


4. Klicken Sie auf **OK**. MapForce fordert Sie auf, optional eine EDI-Beispieldatei auszuwählen.

© 2016-2022 Altova GmbH



5. Wenn Sie Daten *von* einer IDoc-Datei mappen möchten, klicken Sie auf **Durchsuchen** und wählen Sie die IDoc-Datei (*.idoc), aus der die Daten stammen, aus. Eine IDoc-Beispieldatei finden Sie im MapForce-Ordner Examples (siehe **<Dokumente>\Altova\MapForce2023\MapForceExamples\ORDERS.idoc**). Klicken Sie andernfalls auf **Überspringen**.
6. Daraufhin wird das Dialogfeld "Komponenteneinstellungen" angezeigt. Hier können Sie die Einstellungen für die IDoc-Komponente überprüfen, bevor Sie sie zum Mapping hinzufügen. Sie können diese Einstellungen später bei Bedarf jederzeit ändern, siehe EDI-Komponenteneinstellungen 466.



7. Klicken Sie auf OK, um das Dialogfeld "Komponenteneinstellungen" zu schließen und die IDoc-Komponente zum Mapping-Bereich hinzuzufügen.

4.3.14 SWIFT MT

SWIFT (Society for Worldwide Interbank Financial Telecommunication) ist ein internationaler Anbieter von Finanznachrichtendiensten. SWIFT MT (Message Text) ist ein im SWIFT-Netz verwendeter Nachrichtentyp. MapForce unterstützt das Mappen von SWIFT MT-Daten von und auf jedes andere von MapForce unterstützte

© 2016-2022 Altova GmbH

Format,darunter XML, CSV, JSON, Datenbanken und andere.

MapForce unterstützt die folgenden SWIFT MT-Release-Versionen: Versionen ab 2019. Beachten Sie, dass die installierte Standardversion SWIFT 2022 ist.

Anmerkung: Um Daten von oder auf SWIFT MT mappen zu können, muss als Transformationssprache BUILT-IN[29] verwendet werden. Sie können in MapForce eine Vorschau auf das Mapping anzeigen, Ausgabedateien, falls vorhanden, speichern oder das Mapping mit (einem separat lizenzierten) MapForce Server auf einem anderen Rechner oder einer anderen Plattform ausführen. Die Generierung eines ausführbaren C#-, C++- oder Java-Programms anhand von Mappings, die SWIFT M lesen oder schreiben, wird nicht unterstützt.

Damit das Mappen von Daten von oder auf verschiedene Formate im Zusammenhang mit EDI, darunter SWIFT MT, unterstützt werden kann, verwendet MapForce Release-spezifische Konfigurationsdateien, auch als "EDI-Collections" bekannt. Eine EDI-Collection ist ein Verzeichnis, das maschinenlesbare Konfigurationsdateien im XML-Format enthält. Standardmäßig werden EDI-Collections im Verzeichnis **C:\Programme\Altova\MapForce2023\MapForceEDI** installiert. Der Name von Collections (Verzeichnissen) für SWIFT beginnt mit "SWIFT".

Beachten Sie, dass standardmäßig nur die neueste unterstützte SWIFT-Collection installiert wird - zusätzliche Versionen können von der Altova Webseite heruntergeladen und installiert werden.

Einschränkungen

Beim Mappen von Daten von und auf SWIFT MT-Komponenten gelten die folgenden Einschränkungen. Diese Einschränkungen gelten unabhängig davon, ob das Mapping von MapForce oder MapForce Server ausgeführt wird.

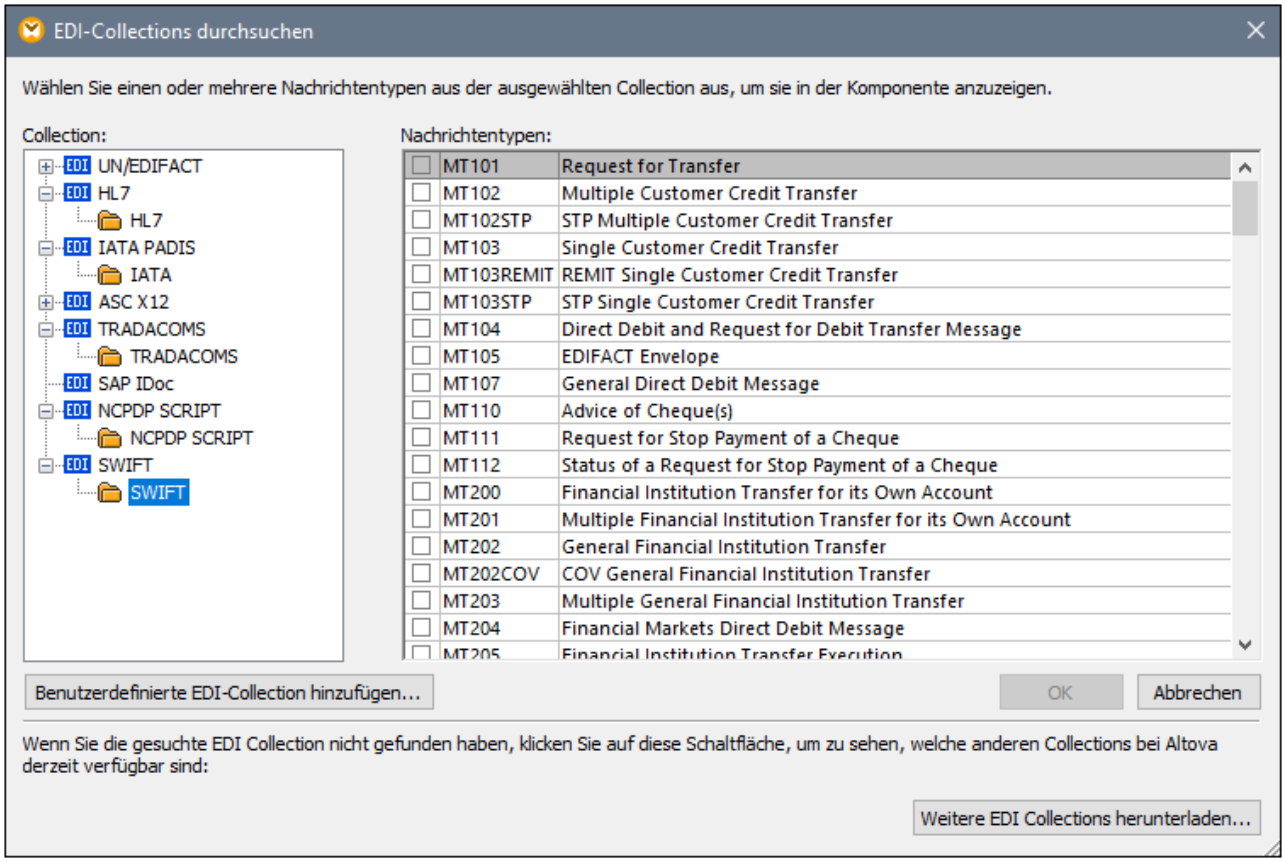
- Die folgenden Nachrichtentypen werden nicht unterstützt:
  - Nicht-FIN-Nachrichten
  - Kategorie 0-Nachrichten (SWIFT-Systemnachrichten).
  - Kategorie n-Nachrichten (MTx9x).
- Um eine SWIFT MT-Nachricht zu generieren, müssen Sie im Application Header des Mappings den richtigen Nachrichtentyp (Message Type) angeben, da dieser nicht automatisch anhand von Inhaltsdaten erstellt wird. Stellen Sie sicher, dass nur Daten des korrekten Nachrichtentyps auf den Nachrichtentextkörper (block 4) gemappt werden.
- MapForce bietet keine Validierung von SWIFT MT-Inhalt. Es werden nur einige grundlegende zum Parsen der Nachricht erforderliche Überprüfungen durchgeführt.
- Es werden keine Überprüfungen zur maximalen Nachrichtenlänge durchgeführt.
- Wenn der Input ungültig ist, stoppt der Parser mit einer Fehlermeldung - es stehen keine weiteren Wiederherstellungsoptionen zur Verfügung.

Hinzufügen von SWIFT MT-Komponenten zum Mapping

SWIFT MT-Nachrichten haben eine Struktur, die sich erheblich von anderen EDI-Nachrichten unterscheidet. Dennoch können Sie in MapForce SWIFT MT-Strukturen auf eine allen EDI-basierten Formaten gemeinsame Art mappen. So können Sie SWIFT MT-Komponenten (entweder als Quelle oder Ziel) auf eine der folgenden Arten zum Mapping hinzufügen:

- Wählen Sie im Menü **Einfügen** den Befehl **EDI**.
- Klicken Sie auf die Symbolleisten-Schaltfläche **EDI-Datei einfügen** .

Wählen Sie im Dialogfeld **EDI-Collections durchsuchen** die gewünschte SWIFT-Release aus und aktivieren Sie anschließend die Nachrichtentypen, die Sie zum Mapping hinzufügen möchten (*siehe Abbildung unten*).



Anmerkung: Standardmäßig steht im oben gezeigten Dialogfeld **EDI-Collections durchsuchen** nur eine generische Collection namens "SWIFT" zur Verfügung. Diese Standard-SWIFT-Collection eignet sich nicht für die Produktion, da ihre Version sich wie alle anderen EDI-Standards in neueren Versionen von MapForce ändern kann. Es wird empfohlen für die Produktion die Release-spezifischen SWIFT-Konfigurationen von der Altova Website herunterzuladen und zu installieren. Diese Release-spezfischen Konfigurationen werden nach der Installation zusätzlich zur Standard-SWIFT-Konfiguration im Dialogfeld **EDI-Collections durchsuchen** angezeigt.

Siehe auch Hinzufügen von EDI-Komponenten zum Mapping 463. Ein Mapping-Beispiel dazu finden Sie unter Beispiel: Konvertieren von SWIFT MT in SWIFT MX 507.

SWIFT MT-Einstellungen

Sie können die Input-Dateien (oder den Pfad der generierten Datei) und anderen Einstellungen von SWIFT MT-Komponenten über das Dialogfeld **Komponenteneinstellungen** ändern. Um das Dialogfeld **Komponenteneinstellungen** zu öffnen, doppelklicken Sie in MapForce auf die Titelleiste der Komponente oder klicken Sie mit der rechten Maustaste auf die Titelleiste der Komponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften**.

© 2016-2022 Altova GmbH

Informationen zu allen Optionen, die in diesem Dialogfeld zur Verfügung stehen, finden Sie unter EDI-Komponenteneinstellungen[466]. Beachten Sie dass die Validierungseinstellungen bei SWIFT MT-Komponenten deaktiviert (ausgegraut) sind, da sie, wie oben erwähnt, nicht unterstützt werden.

Behandlung von SWIFT MT-Datentypen

MapForce behandelt SWIFT MT-spezifische einfache Datentypen folgendermaßen:

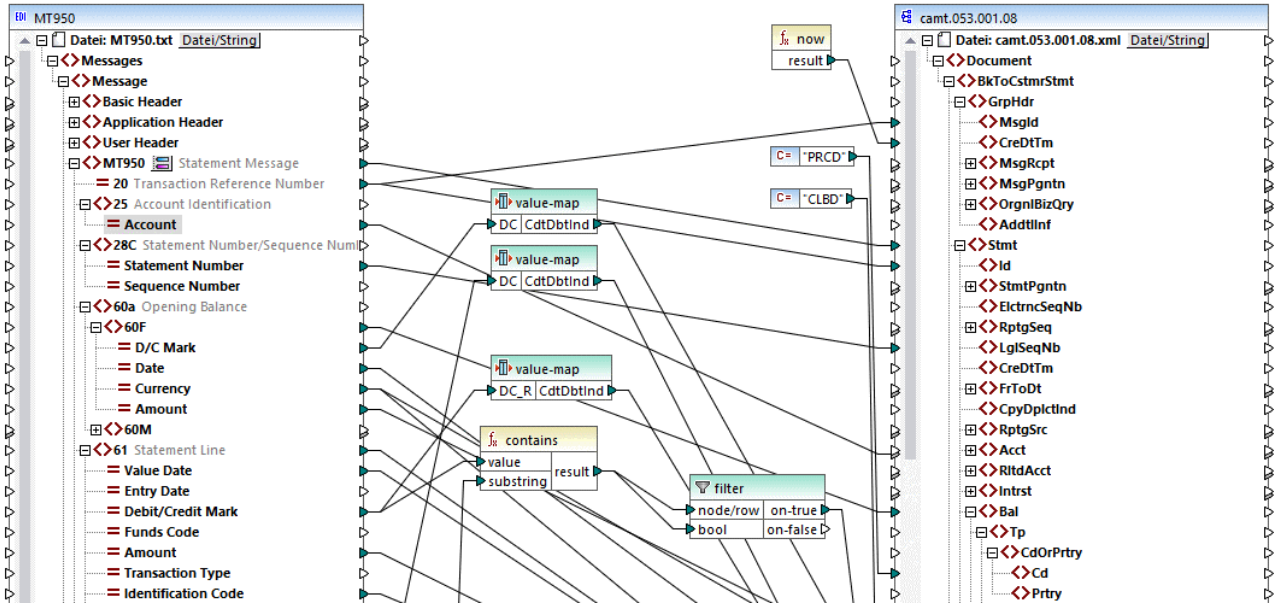
- Sie können SWIFT MT-Datentypen auf ein Datenelement vom Typ `xs:date` oder auf eine Funktion, die `xs:date` als Input erhält, mappen und dieses Mapping wird korrekt verarbeitet, ohne dass eine spezielle Behandlung erforderlich wäre. Wenn Sie z.B. ein Datenelement vom Typ `swift:date6` auf einen String mappen, lautet der Ergebnisstring z.B: "`20200922`". Wenn Sie denselben Typ jedoch auf einen `xs:date` Typ mappen, ist das Ergebnis **`2020-09-22`** (vom Typ `xs:date`).
- Wenn das Zieldatenelement ein String ist und das Datum im Format "`2020-09-22`" dargestellt werden soll, fügen Sie eine Variable[882] vom Typ `xs:date` zum Mapping hinzu und mappen Sie das Ergebnis auf das String-Zieldatenelement.
- Wenn Sie den Wert `swift:decimal` **`42,66`** auf ein Datenelement vom Typ `xs:decimal` mappen, ist das Ergebnis **`42.66`**. Wenn Sie denselben Typ jedoch auf einen `xs:string` Typ mappen, ist das Ergebnis "**`42,66`**".
- Dasselbe Verhalten gilt auch beim Schreiben von Daten in eine Zielkomponente. Wenn Sie z.B. den `xs:decimal`-Wert **`15`** auf ein Datenelement vom Typ `swift:decimal` mappen, ist das Ergebnis **`15`** (vom Typ `swift.decimal`).

Schnelle Konvertierung in XML

Wie bei allen anderen EDI-Formaten können Sie SWIFT MT-Dateien ganz einfach in XML konvertieren, ohne Mapping-Verbindungen ziehen zu müssen. Nähere Informationen dazu finden Sie unter Generieren von XML anhand von EDI[475].

4.3.14.1 Beispiel: Konvertieren von SWIFT MT in SWIFT MX

In diesem Beispiel wird gezeigt,wie Sie eine Datei, die eine SWIFT MT-Nachricht enthält, in eine XML-Zieldatei im SWIFT MX-Format konvertieren. Das Mapping dazu finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\SWIFT_MT950_camt053.mfd**. Aufgrund der Größe dieses Mappings sehen Sie in der Abbildung nur einen Ausschnitt.



SWIFT_MT950_camt053.mfd

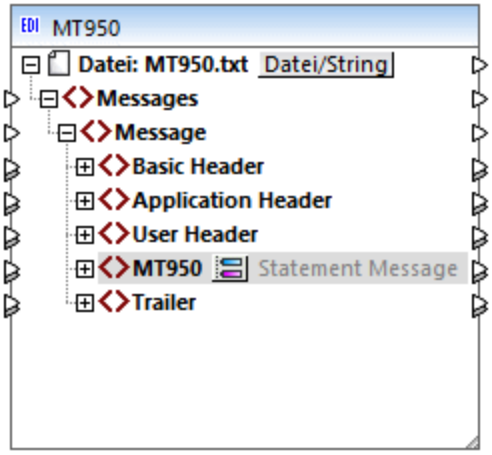
Ein Mapping, wie das oben gezeigte, kann auf die folgende Art erstellt werden:

Schritt 1: Hinzufügen der SWIFT MT-Datei zum Mapping

1. Klicken Sie im Menü **Einfügen** auf **EDI**.
2. Wählen Sie die Collection "SWIFT" und den Nachrichtentyp "MT950 - Statement Message".
3. Wenn Sie aufgefordert werden, eine Instanzdatei bereitzustellen, öffnen Sie die folgende Quelldatei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\MT950.txt**. Daraufhin wird das Dialogfeld "Komponenteneinstellungen" aufgerufen. Sie können die Einstellungen bei diesem Beispiel unverändert lassen. Klicken Sie auf **OK**. Sie können diese Einstellungen später bei Bedarf jederzeit ändern, siehe EDI-Komponenteneinstellungen 466.

Daraufhin wird die SWIFT MT-Komponente zum Mapping hinzugefügt.

 © 2016-2022 Altova GmbH

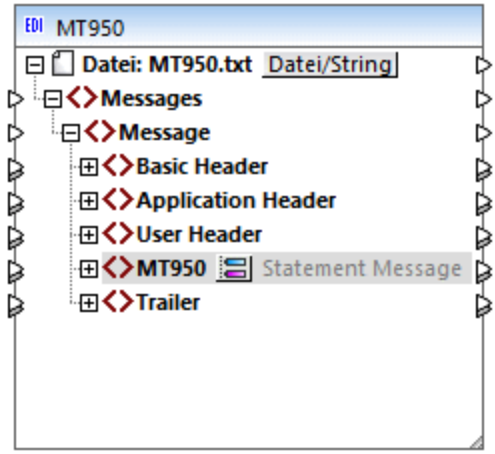


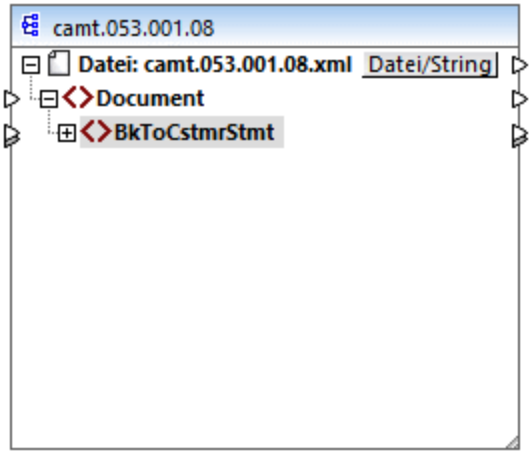
Falls Sie das noch nicht getan haben, sollten Sie das Mapping nun speichern. Es wird empfohlen, die Mapping-Design-Datei im selben Verzeichnis wie die Quell- und Zieldateien zu speichern, da Sie dadurch alle relativen Pfade zu den Quell- und Zieldateien einfach verwalten können. Sie könnten das Mapping in diesem Beispiel unter **<Dokumente>\Altova\MapForce2023\MapForceExamples** speichern, da dies das Verzeichnis ist, in dem sich sowohl die Quelldatei als auch das Zielschema befindet.

Schritt 2: Hinzufügen des Zielschemas zum Mapping

1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und öffnen Sie die Datei **camt.053.001.08.xsd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**.
2. Wenn Sie aufgefordert werden, eine XML-Beispieldatei anzugeben, klicken sie auf **Überspringen**.

Daraufhin wird auch die XML-Zielkomponente zum Mapping hinzugefügt:

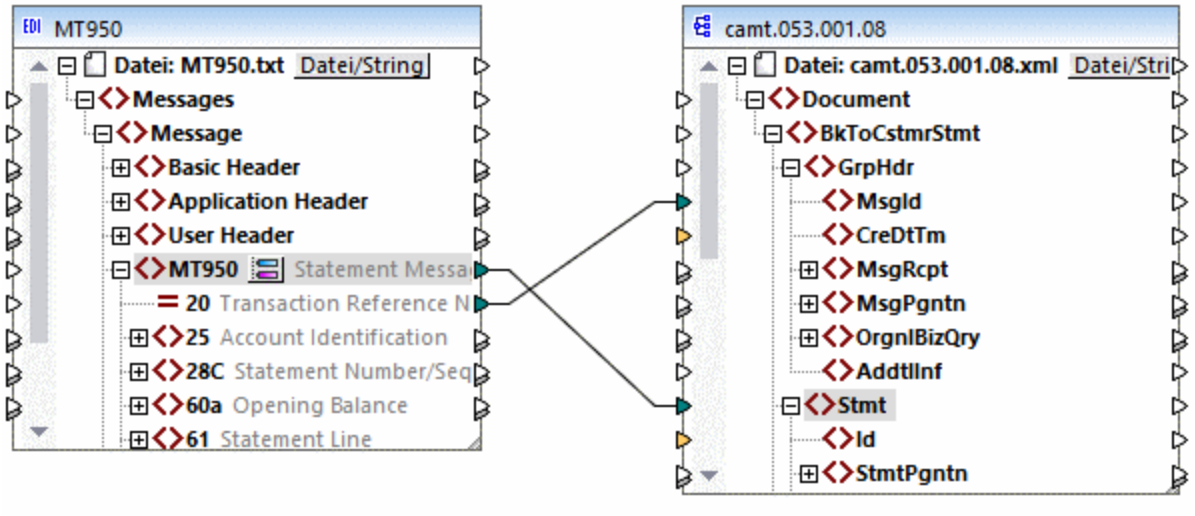




Schritt 3: Hinzufügen der Zwischenkomponenten

Es müssen nun nur noch Verbindungen zwischen der Quell- und der Zielkomponente gezogen werden. So wird z.B., wie unten gezeigt, das Datenelement **MT950** aus der Quellkomponente auf das Datenelement **Stmt** in der

Zielkomponente gemappt, während das Datenelement **20 (Transaction Reference Number)** auf **MsgId** gemappt wird. Wenn Sie noch nicht mit MapForce vertraut sind, finden Sie unter Arbeiten mit Verbindungen[89] und in den Tutorials[43] nähere Informationen über das Ziehen von Verbindungen.



Wenn zwischen Quell- und Zielkomponente mindestens eine Verbindung besteht, können Sie durch Klick auf das Register **Ausgabe** bereits eine Vorschau auf die Mapping-Ausgabe anzeigen. Bedenken Sie jedoch, dass die erzeugte XML-Ausgabe gemäß dem Schema erst dann gültig ist, wenn alle obligatorischen Konnektoren einen Input haben. Die obligatorischen Konnektoren sind orange markiert (z.B. **CreDtTm** in der Abbildung oben).

Neben direkten Verbindungen von der Quell- zur Zielkomponente enthält das vollständige Beispielmapping (**SWIFT_MT950_camt053.mfd**) einige Zwischenkomponenten:

1. Wertezuordnungskomponenten (Look-up-Tabellen). Diese sind nützlich, wenn Werte zwischen zwei Sets konvertiert werden müssen (z.B. "C" in "CRDT", "D" in "DBIT", usw.). Nähere Informationen dazu finden Sie unter Verwendung von Wertezuordnungen[921].
2. Die Funktionen **now** und **contains**. Die Funktion **now** gibt das aktuelle Datum und die aktuelle Uhrzeit zurück, während die Funktion **contains** den Booleschen Wert "true" zurückgibt, wenn das erste String-Argument das zweite String-Argument enthält. Nähere Informationen dazu finden Sie unter Hinzufügen einer Funktion zum Mapping[943].
3. Verschiedene String-Konstanten (z.B. "PRCD", "CLBD"), siehe auch Hinzufügen einer Konstante zum Mapping[943].

In der folgenden Tabelle finden Sie nur einige Beispiele dafür, wie einige der Konnektoren der Zielkomponente ihren Input erhalten.

Ziel-Konnektor	Beschreibung
`GrpHdr`	Zwar erhält dieses Zieldatenelement nicht explizit einen Input, doch erhält sein Child-Element **MsgId** einen Input (siehe unten); daher wird bei Ausführen des Mappings auch das Parent-Element **GrpHdr** erstellt.

© 2016-2022 Altova GmbH

Ziel-Konnektor	Beschreibung
`MsgId`	Erhält eine direkte Verbindung von **Transaction Reference Number** in der Quellkomponente (Datenelement **20** unter **MT950**).
`CreDtTm`	Erhält den direkten Output der Funktion now[1144], welche eine vordefinierte MapForce-Funktion ist, die das aktuelle Datum und die aktuelle Uhrzeit zurückgibt.
`Stmt`	Erhält eine direkte Verbindung von Quelldatenelement **MT950**.
`Stmt/Bal/Tp/CdOrPrtry/Cd`	Erhält Input aus einer Textkonstante, "PRCD".

Ausführen des Mappings

Um eine Vorschau auf die generierte Datei zu sehen, klicken Sie in MapForce auf das Register **Ausgabe**. In der Abbildung unten sehen Sie nur einen Ausschnitt der erzeugten XML-Datei.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Document xsi:schemaLocation="urn:iso:std:iso:20022:tech:xsd:camt.053.001.08
3      <BkToCstmrStmt>
4          <GrpHdr>
5              <MsgId>MF12345</MsgId>
6              <CreDtTm>2020-09-28T14:09:53.277+02:00</CreDtTm>
7          </GrpHdr>
8          <Stmt>
9              <Id>MF12345</Id>
10             <LglSeqNb>42</LglSeqNb>
11             <Acct>
12                 <Id>
13                     <IBAN>NL58NANO3141592658</IBAN>
14                 </Id>
15             </Acct>
16             <Bal>
17                 <Tp>
18                     <CdOrPrtry>
19                         <Cd>PRCD</Cd>
20                     </CdOrPrtry>
21                 </Tp>
22                 <Amt Ccy="EUR">3723495</Amt>
23                 <CdtDbtInd>CRDT</CdtDbtInd>
```

Mapping-Ausgabe

Um die Mapping-Ausgabe in einer Datei zu speichern, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Ausgabe** auf **Ausgabedatei speichern**.
- Klicken Sie auf die Symbolleisten-Schaltfläche **Generierte Ausgabe speichern**.

Automatisierung mit MapForce Server

Wenn Sie eine lizenzierte Version von MapForce Server haben, können Sie das Mapping auch über die Befehlszeile auf einem Linux-, macOS- oder Windows-Rechner ausführen. Beachten Sie, dass normalerweise einige zusätzliche Schritte durchgeführt werden müssen, um Mappings auf einem Server auszuführen, siehe

Vorbereiten von Mappings für die Server-Ausführung[1435]. Dies sind die Schritte, die durchgeführt werden müssen, um dieses konkrete Mapping auf einem Server auszuführen:

1. Klicken Sie mit der rechten Maustaste in einen leeren Bereich des Mappings und wählen Sie im Kontextmenü den Befehl **Mapping-Einstellungen**.
2. Deaktivieren Sie das Kontrollkästchen **Pfade im generierten Code absolut machen** und speichern Sie das Mapping im selben Verzeichnis.
3. Klicken Sie im Menü **Datei** auf den Befehl **Zu MapForce Server-Ausführungsdatei kompilieren** und speichern Sie die .mfx-Datei im Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**.
4. Kopieren Sie die generierte .mfx-Datei in ein Verzeichnis auf dem Server-Rechner (nennen wir es das "Arbeitsverzeichnis").
5. Kopieren Sie die Datei **<Dokumente>\Altova\MapForce2023\MapForceExamples\MT950.txt** in das Arbeitsverzeichnis auf dem Server-Rechner.
6. Öffnen Sie einen Terminal und ändern Sie das aktuelle Verzeichnis in das Arbeitsverzeichnis.
7. Führen Sie MapForce Server mit dem unten stehenden Befehl aus. Passen Sie für Ihr jeweiliges Betriebssystem **mapforceserver_exec** an den entsprechenden Pfad zur ausführbaren MapForce Server Datei an.

```
mapforceserver_exec run SWIFT_MT950_camt053.mfx
```

Sie können Mappings bei der Server-Ausführung als API-Aufruf oder als FlowForce Server-Auftrag (entweder bei Bedarf oder in regelmäßigen Abständen) ausführen. Nähere Informationen dazu finden Sie unter Kompilieren von Mappings zu MapForce Server-Ausführungsdateien[1442].

4.3.15 SWIFT MX / ISO 20022

Die SWIFT MX / ISO 20022-Standards sind XML-basiert. Sie können solche Komponenten daher auf dieselbe Art wie jedes andere XML-Schema[125] zum Mapping hinzufügen. Um solche Daten zu mappen, gibt es keine besonderen anderen Voraussetzungen.

Die Standard SWIFT MX-Schemas können von https://www2.swift.com/knowledgecentre/products/Standards%20MX heruntergeladen werden.

Die Standard ISO 20022 -Schemas können von https://www.iso20022.org/iso-20022-message-definitions heruntergeladen werden.

Es gibt viele nicht standardgemäße Varianten der oben genannten Schemas wie z.B. länderspezifische Varianten. Sie können in MapForce jedes beliebige XML-Schema als Mapping-Komponente verwenden, solange dieses ein gültiges XML-Schema ist.

Ein Mapping-Beispiel dazu finden Sie unter Beispiel: Konvertieren von SWIFT MT in SWIFT MX[507].

4.3.16 TRADACOMS

TRADACOMS (Trading Data Communications) ist ein Standard für den elektronischen Austausch von Einzelhandelsdaten, der v.a. in Großbritannien verwendet wird und 1982 als erster EDI-Standard für den englischen Einzelhandel und die englische Industrie eingeführt wurde. TRADACOMS weist zwar viele

© 2016-2022 Altova GmbH

Ähnlichkeiten mit dem UN/EDIFACT-Standard auf, ist aber einer der Vorläuferstandards von EDIFACT und verwendet innerhalb von Nachrichten andere Strukturen.

In MapForce ist die TRADACOMS-Basisspezifikation aus dem "TRADACOMS Manual of Standards for Electronic Data Interchange", der von der Article Numbering Association (ANA) UK, heute als GS1 UK bekannt (https://www.gs1uk.org), im Jänner 1993 veröffentlicht wurde, implementiert. MapForce kann mit Hilfe von Konfigurationsdateien für andere TRADACOMS-Versionen angepasst werden, um neue Nachrichtentypen, Datenelemente und Codewerte verarbeiten zu können.

Sie können in MapForce mit dem TRADACOMS-Format auf dieselbe Art wie mit anderen unterstützten EDI-Formaten arbeiten. Sie haben folgende Möglichkeiten:

- Sie können TRADACOMS-Dateien auf jedes andere von MapForce unterstützte Datenformat bidirektional mappen, d.h. als Datenquelle oder -ziel.
- Das Mappen von Daten von oder auf das TRADACOMS-Format wird in der BUILT-IN-Sprache (die bei der Anzeige der Mapping-Vorschau oder bei der Ausführung mit MapForce Server und FlowForce Server verwendet wird) und in den Codegenerierungssprachen (C++, C#, Java) unterstützt. Wenn Sie das Mapping auf einem FlowForce Server bereitstellen, der auf einem anderen Rechner ausgeführt wird, so enthält das bereitgestellte Paket automatisch die Konfigurationen der ausgewählten TRADACOMS-Nachrichtentypen sowie alle von ihren Datenelementen verwendeten Codelisten. Ebenso generiert MapForce im generierten Code Klassen für die Konfigurationsgruppen, -segmente und Datenelemente.
- Sie können flexibel definieren, welche Validierungsereignisse einen Abbruch der Datenkonvertierung zur Folge haben sollen, bei welchen Ereignissen Datensätzen abgewiesen (oder akzeptiert) werden sollen und welche Validierungsereignisse ignoriert werden sollen.
- Sie können die Autokomplettierung von Daten aktivieren oder deaktivieren. Wenn "Automatische Datenkomplettierung" (oder "Autokomplettierung") aktiviert ist, füllt MapForce bei der Generierung von TRADACOMS-Dateien einige der Werte automatisch aus.
- Sie können die Nachrichtentypen, Datenelemente und Codelisten entweder global oder lokal mit Hilfe von Konfigurationsdateien anpassen (siehe Konfigurationsdateien 522).
- Sie können die Kodierung von geparsten oder generierten Dateien festlegen.

Standardmäßig stehen in MapForce die folgenden TRADACOMS-Datei- und Nachrichtentypen zur Verfügung:

Dateityp	Nachrichtentyp	Version
THE PRODUCT INFORMATION FILE	PROHDR (Product File Header)	8
	PROINF (Product Details)	8
	PROTLR (Product File Trailer)	8
THE PRICE INFORMATION FILE	PRIHDR (Price File Header)	8
	PRIINF (Price Details)	8
	PRITLR (Price File Trailer)	8
THE CUSTOMER INFORMATION FILE	CUSHDR (Customer Information Header)	8
	CUSINF (Customer Information Details)	8
	CUSTLR (Customer Information Trailer)	8

Dateityp	Nachrichtentyp	Version
THE ORDER FILE	ORDHDR (Order File Header)	9
	ORDERS (Order Details)	9
	ORDTLR (Order File Trailer)	9
THE PICKING INSTRUCTIONS FILE	PICHDR (Picking Instructions File Header)	4
	PICKER (Picking Instructions File Details)	4
	PICTLR (Picking Instructions File Trailer)	4
THE DELIVERY NOTIFICATION FILE	DELHDR (Delivery File Header)	9
	DELIVR (Delivery Details)	9
	DELTLR (Delivery File Trailer)	9
DELIVERY CONFIRMATION FILE	DLCHDR (Delivery Confirmation Header)	5
	DLCDET (Delivery Confirmation Details)	5
	DLCTLR (Delivery Confirmation Trailer)	5
THE INVOICE FILE	INVFIL (Invoice File Header)	9
	INVOIC (Invoice Details)	9
	VATTLR (VAT Trailer)	9
	INVTLR (Invoice File Trailer)	9
THE CREDIT NOTE FILE	CREHDR (Credit Note File Header)	9
	CREDIT (Credit Note Details)	9
	VATTLR (File VAT Trailer)	9
	CRETLR (Credit Note File Trailer)	9
STATEMENT/REMITTANCE FILE	SRMHDR (Statement/Remittance Details File Header)	9
	SRMINF (Statement/Remittance Line Details)	9
	SRMTLR (Statement/Remittance Details File Trailer)	9
UPLIFT INSTRUCTION FILE	UPLHDR (Uplift File Header)	4
	UPLIFT (Uplift File Details)	4
	UPLTLR (Uplift File Trailer)	4
UPLIFT CONFIRMATION FILE	UCNHDR (Uplift Confirmation Header)	3

© 2016-2022 Altova GmbH

Dateityp	Nachrichtentyp	Version
	UCNDET (Uplift Confirmation Details)	3
	UCNTLR (Uplift Confirmation Trailer)	3
THE STOCK SNAPSHOT FILE	SNPHDR (Stock Snapshot Header)	3
	SNPSTS (Stock Snapshot Details)	3
	SNPTLR (Stock Snapshot Trailer)	3
THE STOCK ADJUSTMENT FILE	SADHDR (Stock Adjustment Header)	3
	SADDET (Stock Adjustment Details)	3
	SADTLR (Stock Adjustment Trailer)	3
AVAILABILITY REPORT FILE	AVLHDR (Availability File Header)	4
	AVLDET (Availability Report Details)	4
	AVLTLR (Availability File Trailer)	4
GENERAL COMMUNICATIONS FILE	GENHDR (General Communications File Header)	3
	GENRAL (General Communications Text)	3
	GENTLR (General Communications Trailer)	3
COMPLEX ORDER FILE	CORHDR (Complex Order File Header)	6
	CORDER (Complex Order Details)	6
	CORTLR (Complex Order Trailer)	6
THE ACKNOWLEDGEMENT OF ORDER FILE	ACKHDR (Acknowledgement File Header)	4
	ACKMNT (Acknowledgement Details)	4
	ACKTLR (Acknowledgement Trailer)	4
PRODUCT PLANNING REPORT FILE	PPRHDR (Product Planning Report Header)	2
	PPRDET (Product Planning Report Details)	2
	PPRTLR (Product Planning Report Trailer)	2
THE PAYMENT ORDER FILE	PAYHDR (Payment Order File Header)	3
	PAYINF (Payment Order Line Details)	3
	PAYTLR (Payment Order File Trailer)	3
THE DEBIT ADVICE FILE	DRAHDR (Debit Advice File Header)	3

Dateityp	Nachrichtentyp	Version
	DRAINF (Debit Advice Line Details)	3
	DRATLR (Debit Advice File Trailer)	3
THE CREDIT ADVICE FILE	CRAHDR (Credit Advice File Header)	3
	CRAINF (Credit Advice Line Details)	3
	CRATLR (Credit Advice File Trailer)	3
THE EXCEPTION CONDITION FILE	EXCHDR (Exception Condition File Header)	3
	EXCINF (Exception Condition Line Details)	3
	EXCTLR (Exception Condition File Trailer)	3
LOCATION PLANNING REPORT FILE	LPRHDR (Location Planning Report Header)	2
	LPRDET (Location Planning Report Details)	2
	LPRTLR (Location Planning Report Trailer)	2
THE UTILITY BILL FILE	UTLHDR (Utility Bill File Header)	3
	UTLBIL (Utility Bill File Details)	3
	UVATLR (Utility Bill VAT Trailer)	3
	UTLTLR (Utility Bill File Trailer)	3

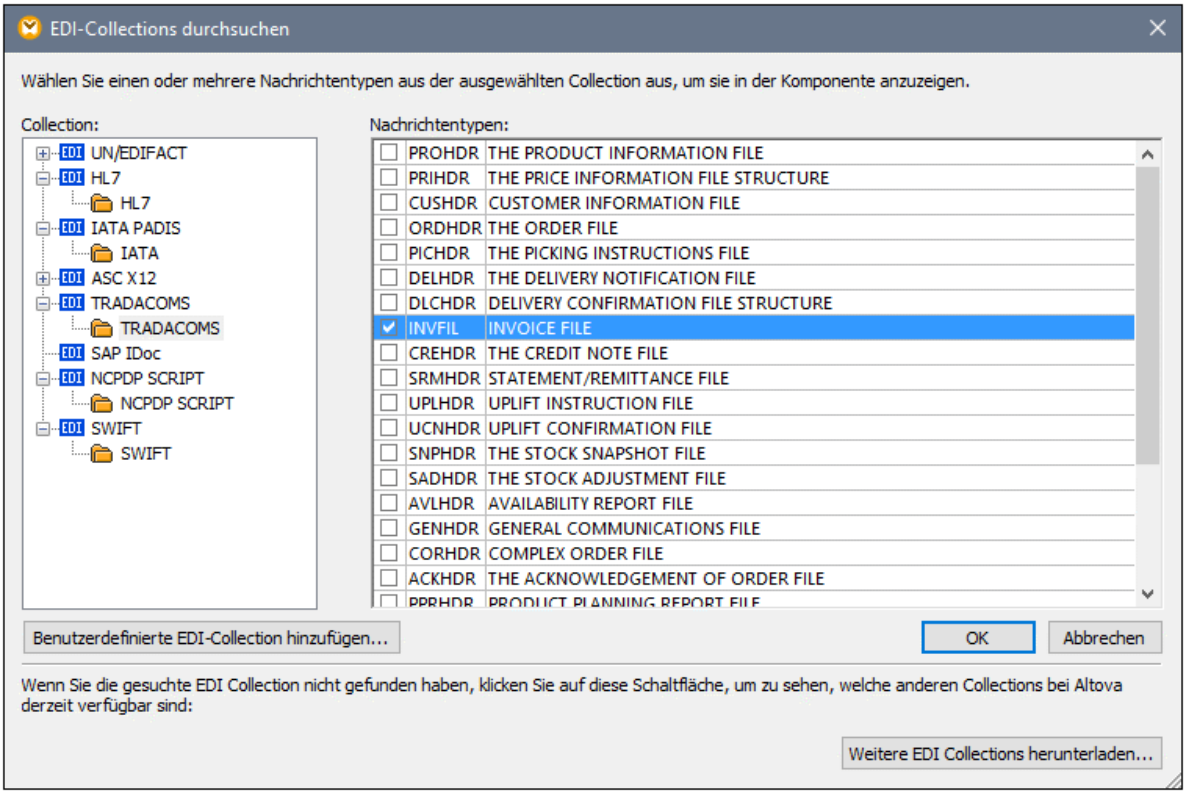
Anmerkung: Auch RSGRSG (Reconciliation Message) Version 2 wird unterstützt. Dieser Nachrichtentyp kann in allen Dateitypen vorkommen.

4.3.16.1 Hinzufügen von TRADACOMS-Dateien als Mapping-Komponenten

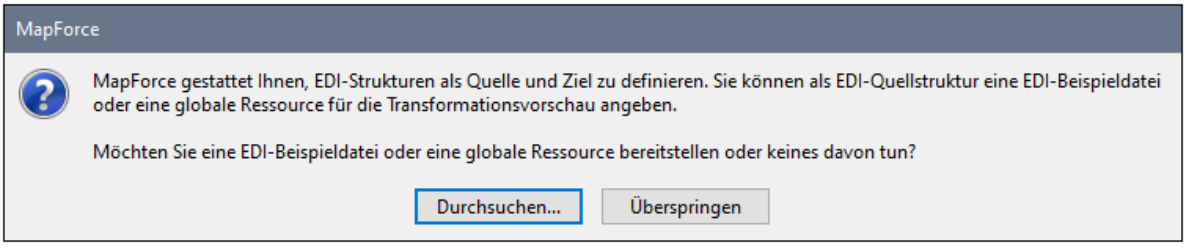
So fügen sie eine TRADACOMS-Datei als Datenmappingquelle hinzu:

1. Wählen Sie eine der folgenden Methoden:
   - Klicken Sie im Menü **Einfügen** auf **EDI**.
   - Klicken Sie auf die Symbolleisten-Schaltfläche **EDI-Datei einfügen** ( ).
2. Klicken Sie im Dialogfeld "EDI-Collections durchsuchen" auf **TRADACOMS** und wählen Sie den/die Nachrichtentypen aus, der/die in die Nachricht inkludiert werden soll(en).

© 2016-2022 Altova GmbH



3. Klicken Sie auf **OK**. Sie werden nun aufgefordert, eine Instanzdatei anzugeben.



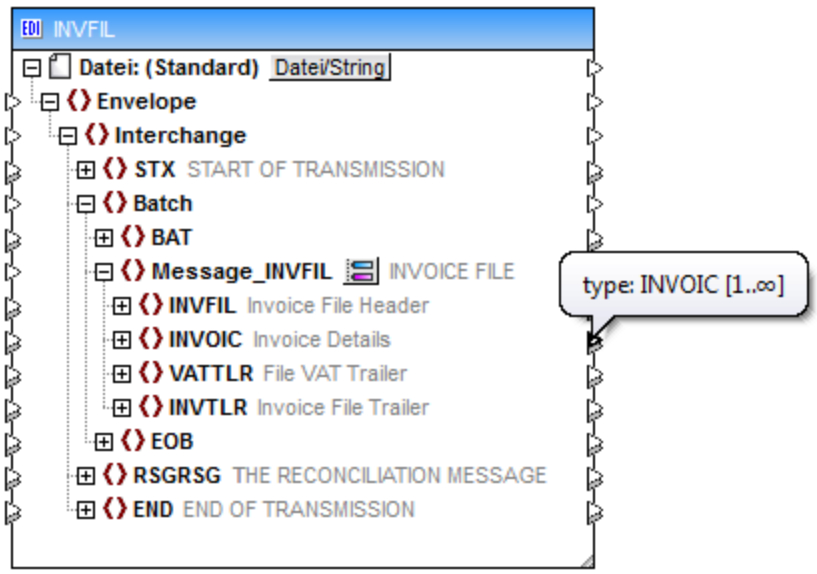
4. Wenn Sie die TRADACOMS-Quelldatei später definieren möchten, klicken Sie auf **Überspringen**. Klicken Sie andernfalls auf **Durchsuchen** und wählen Sie aus, wie Sie die TRADACOMS-Datei öffnen möchten:
   - Wenn Sie die Datei von Ihrem lokalen Verzeichnis aus oder über das Netzwerk öffnen wollen, navigieren Sie zur Datei und klicken Sie anschließend auf **Öffnen**.
   - Wenn Sie die Datei über eine URL öffnen möchten, klicken Sie auf **Zu URL wechseln** (Nähere Informationen dazu finden Sie unter Hinzufügen von Komponenten über eine URL 80)
   - Wenn Sie die Datei zuvor als globale Ressource definiert haben und Sie die Datei als globale Ressource öffnen möchten, klicken Sie auf **Globale Ressourcen** (siehe Globale Altova-Ressourcen 1462).
5. Definieren oder ändern Sie gegebenenfalls im Dialogfeld "Komponenteneinstellungen" die Einstellungen (siehe EDI-Komponenteneinstellungen 466).

So fügen Sie eine TRADACOMS-Datei als Zielkomponente für ein Datenmapping hinzu:

1. Befolgen Sie die oben beschriebenen Schritte 1 bis 2.
2. Klicken Sie auf **Überspringen**, wenn Sie aufgefordert werden, eine Instanzdatei anzugeben.

4.3.16.2 Die TRADACOMS-Komponente in MapForce

Die TRADACOMS-Komponente repliziert in MapForce die Struktur einer allgemeinen TRADACOMS-Übertragung (mit einer optionalen Batch-Funktion), wobei alle Funktionalitäten, die die Komponente mit anderen MapForce-Komponenten, die im Zusammenhang mit EDI stehen, beibehalten werden.



Eine TRADACOMS-Beispielkomponente in MapForce

Im obersten Datenelement der Komponente, nämlich **Datei,** wird der Name der TRADACOMS-Austauschdatei, die gerade verarbeitet oder von der Komponente generiert wird, angezeigt. "Standard" zeigt an, dass noch keine Datei zugewiesen wurde. Über die Schaltfläche **Datei/String** (Datei/String) werden erweiterte Funktionalitäten angezeigt, die allen dateibasierten MapForce-Komponenten gemeinsam sind. Unter anderem haben Sie über diese Schaltfläche die Möglichkeit, mehrere Dateien dynamisch zu lesen oder zu verarbeiten (Nähere Informationen dazu finden Sie unter Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 1371).

Die "Envelope-" und "Interchange"-Struktur sind allgemeine Strukturen, die bei allen mit EDI in Zusammenhang stehenden MapForce-Komponenten vorhanden sind. Wo dies erforderlich ist, bieten sie die Möglichkeit, mehrere Datenaustauschvorgänge in derselben MapForce-Komponente zu verarbeiten.

Bei der `STX`- und `END`-Struktur handelt es sich um ein TRADACOMS-spezifisches Format. Sie kennzeichnen die Segmente für "Start of Transmission" (Beginn der Übertragung) bzw. "End of Transmission" (Ende der Übertragung).

Bei der `BAT`- und `EOB` -Struktur handelt es sich um ein TRADACOMS-spezifisches Format. Sie kennzeichnen die Segmente für "Start of Batch" (Beginn des Batch) bzw. "End of Batch" (Ende des Batch). Solche

 © 2016-2022 Altova GmbH

Segmente werden verwendet, wenn Sie Daten aus einer TRADACOMS-Austauschdatei mit Batches lesen oder wenn Sie eine Datei mit Batches generieren möchten. Da in einer einzigen Übertragung mehrere Batches vorhanden sein können, steht eine Batch-Sequenz als übergeordnetes Element der `BAT`-Sequenz zur Verfügung.

Die **Message_*Code***-Struktur (im Beispiel oben **Message_INVFIL**) entspricht der TRADACOMS-"Datei". Über die Schaltfläche **EDI-Nachrichtentypen auswählen** ( ) wird ein Dialogfeld aufgerufen, in dem Sie den Typ der Datei ändern können (z.B. "Invoice File", "Orders File", usw.). Aufgrund der vorhandenen EDI-Funktionalität von MapForce können Sie auch mehrere Dateitypen in derselben Komponente inkludieren. Beachten Sie jedoch, dass in der TRADACOMS-Spezifikation pro Datenaustausch nur eine Datei und ein Dateityp empfohlen werden.

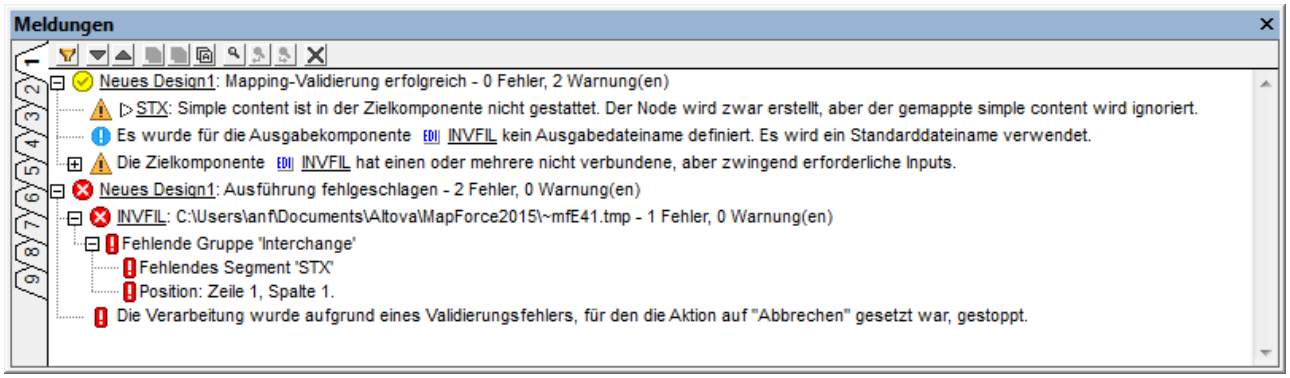
Jede TRADACOMS-Datei (In diesem Beispiel "Invoice File") enthält eine Header-Nachricht, eine oder mehrere Details-Nachrichten und eine Trailer-Nachricht. In der Komponente oben sind diese die `INVFIL-` (Invoice File Header), `INVTLR-` (Invoice File Trailer), `VATTLR-` (File VAT Trailer) sowie mehrere `INVOIC-` (Invoice Details)-Nachrichten.

Wie bei jeder anderen MapForce-Komponente haben Sie über die Input- und Output-Konnektoren (kleine Dreiecke) auf jeder Seite der Komponente die Möglichkeit, einzelne Datenelemente oder -segmente von oder auf andere von MapForce unterstützte Datentypen oder -formate zu mappen. So können Sie auch zusätzliche Informationen zu einem Datenelement anzeigen (z.B. die minimale oder maximal zulässige Zahl der Instanzen), wenn Sie den Mauszeiger über ein Datenelement der Komponente platzieren, wenn die Anzeige von Tipps über das Menü **Ansicht | Tipps** anzeigen aktiviert wurde.

Sie können diese und andere Einstellungen auch ändern, indem Sie auf die Komponentenüberschrift doppelklicken und das Dialogfeld "Komponenteneinstellungen" öffnen (siehe EDI-Komponenteneinstellungen 466). Beachten Sie, dass dies ein allgemeines Dialogfeld für alle mit EDI in Zusammenhang stehenden Komponententypen ist und dass daher einige der Optionen nicht auf TRADACOMS anwendbar sind.

4.3.16.3 Validierung und automatische Datenkomplettierung

Wenn Sie ein Mapping ausführen, das Daten aus einer TRADACOMS-Struktur liest oder in diese schreibt, führt MapForce eine strukturelle Datenvalidierung gemäß der TRADACOMS-Spezifikation durch und zeigt etwaige Validierungsfehler im Fenster "Meldungen" an.

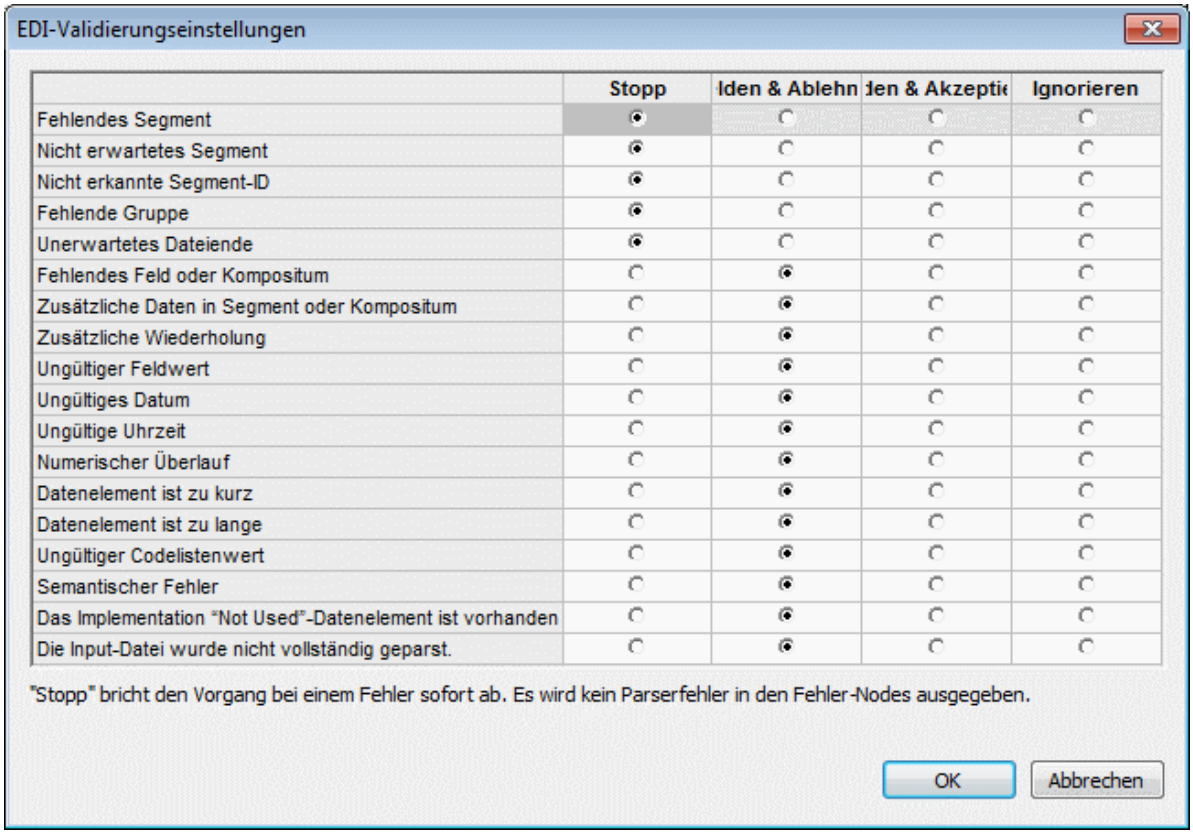


Fenster "Meldungen" mit Validierungsfehlern

Die TRADACOMS-spezifischen Valdierungsmeldungen (Informationen, Warnungen, Fehler) werden, wie im Beispiel oben gezeigt, zusätzlich zu allgemeinen, allen Mappings gemeinsamen Meldungen angezeigt. Nähere Informationen zur MapForce-Validierung im Allgemeinen finden Sie unter Validieren von Mappings[107].

Die folgenden Faktoren sind ausschlaggebend dafür, wie MapForce die geparsten bzw. generierten TRADACOMS-Dateien validiert:

- die in den Konfigurationsdateien im MapForce-Installationsordner (Unterordner **MapForceEDI\TRADACOMS**) definierten Validierungseinschränkungen. Diese Konfigurationsdateien enthalten einerseits die Standardvalidierungsregeln der TRADACOMS-Spezifikation. Andererseits bieten sie aber auch die Möglichkeit, das TRADACOMS-Format an spezielle Vorgaben anzupassen. So können etwa auch die in den Konfigurationsdateien definierten Datenelemente, Segmente oder Codewerte geändert werden, was sich auch auf das Resultat der Validierung und der Mapping-Ausführung auswirkt. Nähere Informationen zu den Konfigurationsdateien finden Sie unter Konfigurationsdateien[522].
- Die in MapForce vordefinierte Validierungslogik. Dazu gehört die MapForce-interne Überprüfung der Vollständigkeit der Daten, die möglicherweise nicht durch die Konfigurationsdateien erzwungen wird.
- Benutzerdefinierte Validierungseinstellungen, die Sie über die grafische Benutzeroberfläche von MapForce definiert haben (siehe Validierung von EDI-Komponenten[471]). Um die EDI-Validierungseinstellungen einer beliebigen EDI-Komponente (auch einer TRADACOMS-Komponente) anzuzeigen oder zu ändern, doppelklicken Sie auf die Überschrift der Komponente und klicken Sie anschließend im Dialogfeld "Komponenteneinstellungen" auf **Validierung**.



Beim Schreiben von Daten in eine TRADACOMS-Struktur füllt MapForce den Inhalt der Datenelemente, für die der Wert berechnet oder vordefiniert werden kann, automatisch aus. Dies wird als automatische Komplettierung

 © 2016-2022 Altova GmbH

oder "Autokomplettierung" bezeichnet. Um dieses Verhalten zu verhindern, deaktivieren Sie im Dialogfeld "Komponenteneinstellungen" das Kontrollkästchen **Autokomplettierung von fehlenden Feldern** (siehe EDI-Komponenteneinstellungen 466).

Aufgrund der folgenden TRADACOMS-Validierungsregeln, gibt MapForce entweder (beim Parsen oder Generieren der Datei) Validierungsfehler aus oder füllt fehlende Felder (bei der Dateigenerierung) automatisch aus:

1. Die Segmente STX (Start of Transmission) und END (End of Transmission) müssen vorhanden sein.
2. Wenn STDS-1 den Wert 'ANAA' hat, muss eine Reconciliation Message (RSGRSG) vor dem Ende der Übertragung (END) vorhanden sein. Andernfalls muss keine Reconciliation Message (RSGRSG) vorhanden sein.
3. Wenn STDS-1 den Wert 'ANAA' hat:
   a. so muss der Wert von RSGA in der Reconciliation Message gleich dem Wert von SNRF im Segment STX sein.
   b. so muss der Wert von RSGB in der Reconciliation Message gleich dem Wert von UNTO-1 im Segment STX sein.
4. TRDT-1 muss das Datum (JJMMTT) und TRDT-2 muss die Uhrzeit (HHMMSS) der Übertragung enthalten (aktuelles Datum und aktuelle Uhrzeit).
5. Wenn der Batch Header (BAT) vorhanden ist, so muss auch der Batch Trailer (EOB) vorhanden sein. Die Anzahl der Nachrichten im Batch muss im Datenelement NOLI (Number of Messages in Batch) vorhanden sein.
6. Das Datenelement MSRF (Message Reference) im Message Header (MHD) muss die fortlaufende Nummerierung der Nachrichten in der Übertragung (beginnend mit 1) enthalten.
7. Das Datenelement NOSG (Number of Segments in Message) im Message Trailer (MTR) muss die Anzahl der Segmente einschließlich MHD und MTR enthalten.
8. Falls vorhanden, muss die Reconciliation Message (RSGRSG) abgesehen vom Message Header und Message Trailer aus einem Segment (RSG) bestehen.
9. Das Datenelement NMST (Number of Messages in Transmission) im END-Segment muss die Anzahl der Nachrichten (Messages) im Austausch enthalten (Anzahl der MHD-Segmente).
10. Im Allgemeinen erwartet MapForce beim Lesen einer TRADACOMS-Struktur, dass es sich bei der Austauschumgebung um den Typ "Computer an Computer" (oder in der TRADACOMS-Terminologie "intelligenter Terminal an intelligenten Terminal") handelt. Ein Segment wie z.B. `MHD = 12 + ORDHDR :3` würde daher einen Validierungsfehler verursachen, da es zusätzliche vorangestellte und nachgestellte Leerzeichen enthält.
11. String-Daten müssen in Großbuchstaben geschrieben sein. Bei der Generierung von TRADACOMS-Daten konvertiert MapForce die String-Daten in Großbuchstaben.

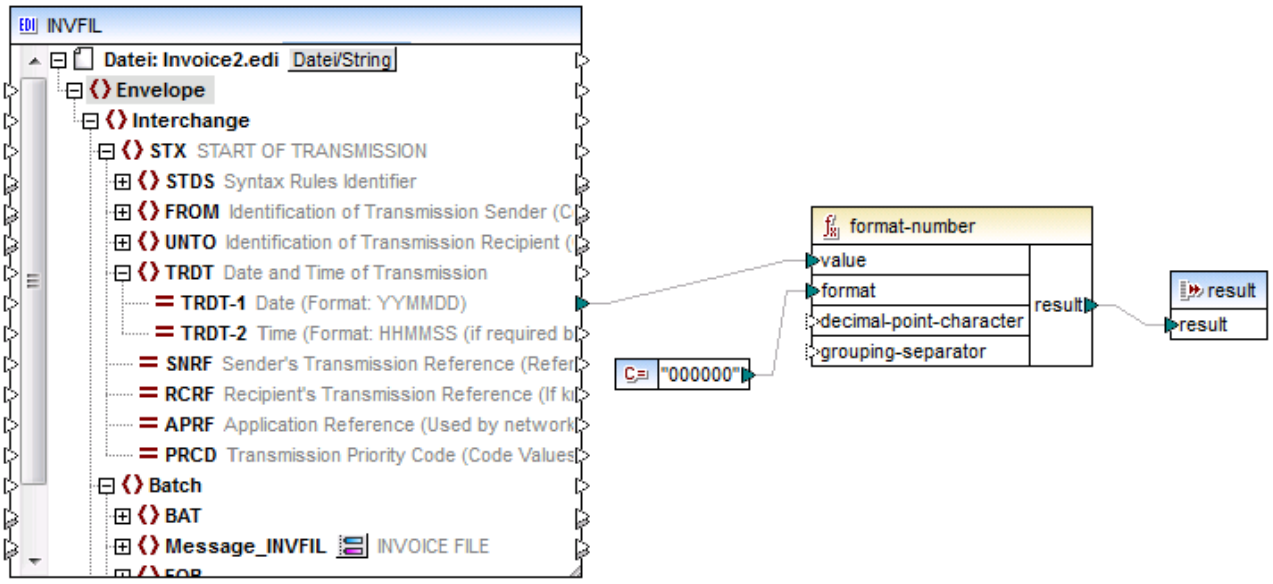
Wie bereits erwähnt, wirken sich zusätzlich dazu alle in den Konfigurationsdateien definierten Validierungsregeln ebenfalls auf das Parsen und die Generierung von TRADACOMS-Dateien aus.

4.3.16.4 Beibehalten vorangestellter Nullen bei der Konvertierung

In der TRADACOMS-Spezifikation ist der Datentyp der Datenelemente TRDT-1 und TRDT-2 (Datum und Uhrzeit der Übertragung) als "decimal" definiert. Das bedeutet, dass alle vorangestellten Nullen in diesen Feldern beim Mappen der Daten von den Feldern TRDT-1 oder TRDT-2 standardmäßig bei der Konvertierung entfernt würden, was zu unerwünschten Ergebnissen führen würde. Dies kann auch bei anderen Datenelementen, die als "decimal" definiert sind, die aber Werte enthalten, die als Strings zu behandeln sind, vorkommen.

Sie können in MapForce einstellen, dass numerische Felder als Strings behandelt werden (sodass vorangestellte Nullen erhalten bleiben), indem Sie die MapForce-Funktion **`core | format-number`** 1039

anwenden. Im folgenden Beispiel ist der Wert des Datenelements TRDT-1 in der Quelldatei "020312". Normalerweise würde der Wert dieses Datenelements (aufgrund der Konvertierung in eine Dezimalzahl) als "20312" ausgegeben, was nicht das gewünschte Ergebnis ist. Um die vorangestellte Null beizubehalten, wurde die **format-number** Funktion zum Mapping hinzugefügt.



Beibehalten vorangestellter Nullen mit Hilfe der format-number-Funktion

Wie oben gezeigt, hat die Funktion die folgenden zwei Input-Argumente:

1. den zu formatierenden Wert (in diesem Fall "020312").
2. die Formatmaske "000000".

Um das Ergebnis der Funktion zu testen, wird in diesem Beispiel eine einfache Output-Komponente verwendet (siehe Rückgabe von String-Werten aus einem Mapping 878). Wenn Sie auf die Schaltfläche **Ausgabe** klicken, ist die Ausgabe des Mappings (welches auch das Ergebnis der Funktion ist), wie gewünscht, "020312".

4.3.16.5 Konfigurationsdateien

Standardmäßig entsprechen die MapForce-Konfigurationsdateien und die zugrunde liegende Verarbeitungslogik der Basisversion des TRADACOMS-Standards (931). Wenn Sie folglich diese TRADACOMS-Version verwenden, müssen die TRADACOMS-Konfigurationsdateien nicht angepasst werden. Wenn Sie allerdings aus irgendeinem Grund Änderungen an den bestehenden Konfigurationsdateien vornehmen möchten, so ist dies möglich. Beachten Sie, dass alle Änderungen an den TRADACOMS-Standardkonfigurationsdateien sofort wirksam werden und sich auf die Art, wie MapForce TRADACOMS-Strukturen parst oder schreibt, auswirken.

Die MapForce-Konfigurationsdateien für TRADACOMS sind im folgenden Verzeichnis gespeichert: **C:\Programme\Altova\MapForce2023\MapForceEDI\TRADACOMS**. Falls Sie Änderungen an den TRADACOMS-Konfigurationsdateien vornehmen müssen, erstellen Sie zuerst eine gleichrangige Kopie des TRADACOMS-Verzeichnisses, z.B. "TRADACOMS1", ändern Sie die Rechte in Schreibrechte um und bearbeiten Sie die Dateien Ihren Vorgaben entsprechend. Alle Konfigurationsdateien entsprechend der XML-Syntax, daher wird empfohlen, für die Bearbeitung der Dateien einen XML-Editor zu verwenden.

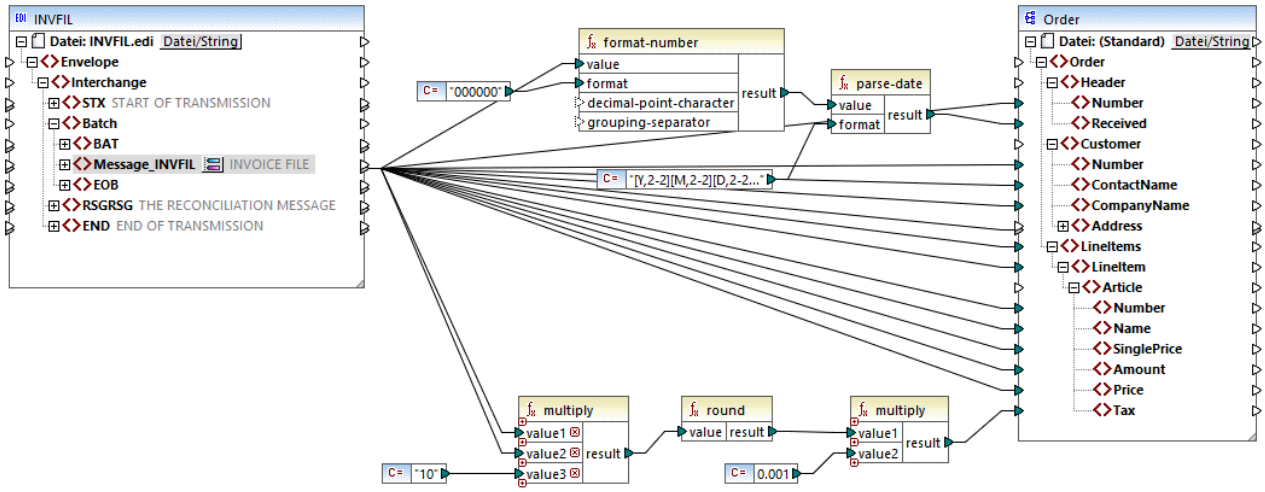
© 2016-2022 Altova GmbH

Datei	Aufgabe
EDI.Collection	Die Datei **EDI.Collection** enthält eine Liste aller Messages (Nachrichten) im aktuellen TRADACOMS-Konfigurationsverzeichnis (z.B. **C:\Programme\Altova\MapForce2023\MapForceEDI\TRADACOMS**). Alle in der Collection-Datei aufgelisteten Nachrichten stehen im Dialogfeld "EDI-Collections durchsuchen" zur Verfügung, wenn Sie eine TRADACOMS-Komponente zum Mapping hinzufügen **(Einfügen | EDI)**. Wenn ein bestimmter Nachrichtentyp daher in diesem Dialogfeld nicht angezeigt werden soll, kommentieren Sie das entsprechende XML-Element aus der EDI.Collection-Datei aus oder entfernen Sie es. Wenn Sie z.B. die folgende Zeile auskommentieren, steht der Nachrichtentyp LPRHDR nicht mehr zur Verfügung: `<!--<Message Type="LPRHDR" File="LPRHDR.Config" Description="LOCATION PLANNING REPORT FILE"/>-->`
Envelope.Config	Definiert die Struktur der TRADACOMS-Komponente in MapForce (d.h. Elemente, die in die **Envelope**-Sequenz eingeschlossen sind). Diese Datei wird anhand eines Schemas validiert, das von MapForce intern gehandhabt wird und sollte nicht bearbeitet werden.
Tradacoms.Codelist	In dieser Datei sind die TRADACOMS-Codes und die Werte, die diese enthalten können, definiert. Sie wird zur Validierung von TRADACOMS-Input/Output-Dateien verwendet. Sie können bei Bedarf benutzerdefinierte Codelisten zu dieser Datei hinzufügen.
Tradacoms.Segment	In dieser Datei sind die Segment-, Composite- und Feld-Namen der TRADACOMS-Dateien definiert. Sie wird beim Parsen einer TRADACOMS-Datei verwendet. Änderungen, die an dieser Datei vorgenommen werden, sind globale Anpassungen und werden auf alle Segmente und Nachrichten angewendet.
[Message].Config	Definiert die Struktur einer einzelnen TRADACOMS-Nachricht. Änderungen, die an dieser Datei vorgenommen werden, werden als lokale (Inline)-Anpassungen behandelt.

Nähere Informationen finden Sie unter Anpassen der EDI-Struktur 539 und Anpassen der EDI-Validierung 573.

4.3.16.6 Beispiel: Konvertieren einer TRADACOMS-Rechnung in XML

Dieses Beispiel bietet eine Anleitung, wie Sie ein Mapping-Design erstellen, das Daten aus einer TRADACOMS-Rechnung in ein XML-Format konvertiert. Die in diesem Beispiel erstellte Mapping-Design-Datei steht auch unter den folgenden Pfad zur Verfügung:
<Dokumente>\Altova\MapForce2023\MapForceExamples\TRADACOMS_Invoice.mfd.



TRADACOMS_Invoice.mfd

Das Mapping soll die folgenden Schritte durchführen:

1. Konvertieren einer TRADACOMS-Quelldatei (**INVFIL.edi**) in eine XML-Datei, die anhand eines vorhandenen Schemas (**Order.xsd**) validiert wird. Die TRADACOMS-Quelldatei und die Schema-Datei (**Order.xsd**) stehen beide im Ordner MapForce Examples (**<Dokumente>\Altova\MapForce2023\MapForceExamples**) zur Verfügung.
2. Die Daten in der TRADACOMS-Datei haben das Format `decimal` und müssen daher für die XML-Zieldatei in das Format `xs:dateTime` konvertiert werden.
3. Für jeden Zeileneintrag in der Bestellung muss der Umsatzsteuerbetrag als Dezimalwert, gerundet auf drei Nachkommastellen (Tausendstel), berechnet werden. Für dieses Beispiel nehmen wir an, dass für die Berechnung der Umsatzsteuer die folgende Formel verwendet wird:

   ```
   Tax = round(LEXC * VATP * 10) * 0.001
   ```

   round rundet den Wert auf die nächste Ganzzahl auf bzw. ab.

Gehen Sie beim Mapping folgendermaßen vor:

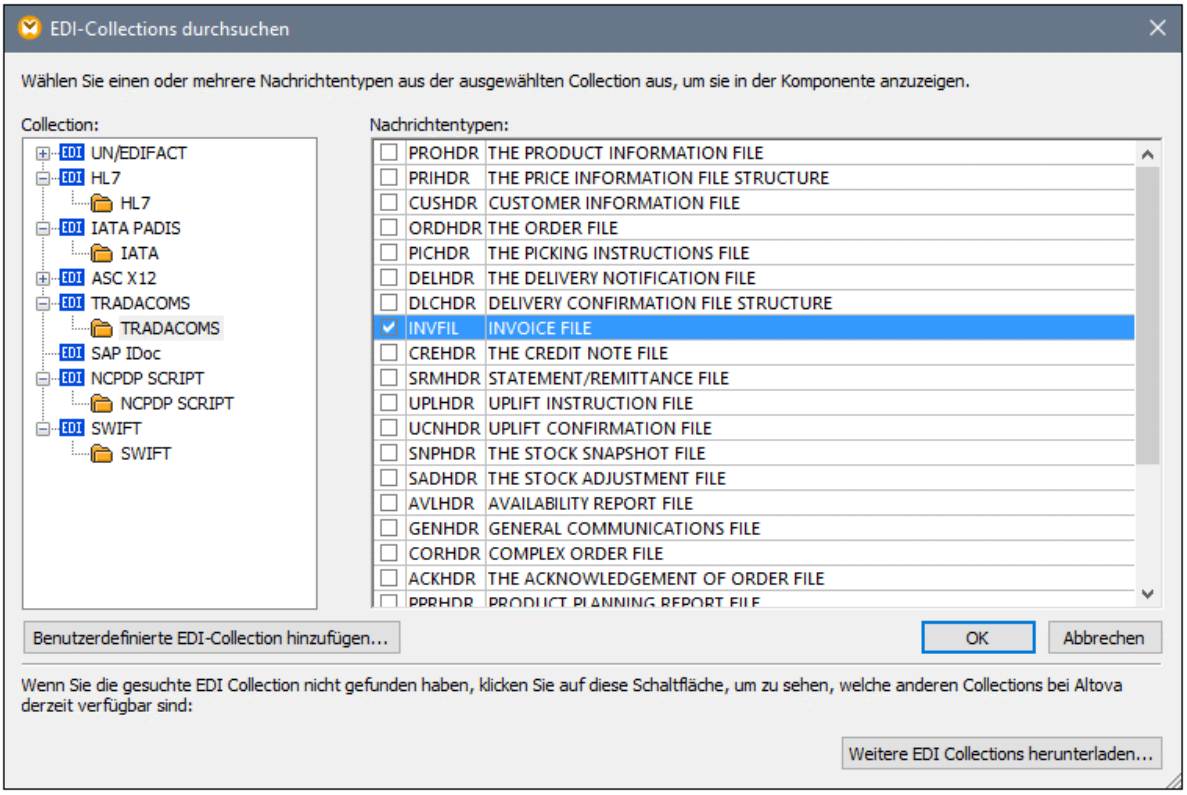
1. Fügen Sie die Quell- und Zielkomponente zum Mapping-Bereich hinzu und ziehen Sie Verbindungslinien zwischen den beiden Komponenten.
2. Verwenden Sie Datumsfunktionen, um das Datum in das gewünschte Format zu konvertieren.
3. Verwenden Sie Math-Funktionen, um den Wert für die Umsatzsteuer zu berechnen.

Schritt 1: Hinzufügen der Komponenten zum Mapping-Bereich

So fügen Sie die TRADACOMS-Komponente hinzu:

1. Klicken Sie im Menü **Einfügen** auf **EDI**.
2. Wählen Sie im TRADACOMS-Collection-Ordner unter Nachrichtentypen den Eintrag **INVOICE FILE** aus.

© 2016-2022 Altova GmbH



3. Wenn Sie aufgefordert werden, eine Beispieldatei anzugeben, navigieren Sie zur Datei **INVFIL.edi** im Ordner MapForce Examples: (**<Dokumente>\Altova\MapForce2023\MapForceExamples**).
4. Wenn Sie aufgefordert werden, die Komponenteneinstellungen zu definieren, belassen Sie die Standardwerte unverändert und klicken Sie auf OK.

So fügen Sie die XML-Zielkomponente hinzu:

1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Dat**ei.
2. Wenn Sie aufgefordert werden, eine Beispieldatei anzugeben, klicken Sie auf **Überspringen**. (Da es sich um eine Zielkomponente handelt, wird die Datei generiert, daher muss keine vorhandene Beispieldatei angegeben werden.)
3. Wenn Sie aufgefordert werden, ein Root-Element auszuwählen, wählen Sie "Order".

Die Daten werden in diesem Beispiel folgendermaßen von der TRADACOMS-Quelldatei auf die XML-Zieldatei gemappt:

Quelle (TRADACOMS)	**Ziel (XML)**
INVN	/Order/Header/Number
IVDT*	/Order/Header/Received
CIDN-2	/Order/Customer/Number
CADD-1	/Order/Customer/ContactName

Quelle (TRADACOMS)	Ziel (XML)
CNAM	/Order/Customer/CompanyName
CADD-2	/Order/Customer/Address/Street
CADD-3	/Order/Customer/Address/City
CADD-5	/Order/Customer/Address/ZIP
CADD-4	/Order/Customer/Address/State
Group_IRF**	/Order/LineItems
Group_ILD**	/Order/LineItems/LineItem
SPRO-1	/Order/LineItems/LineItem/Article/Number
TDES-1	/Order/LineItems/LineItem/Article/Name
AUCT-1	/Order/LineItems/LineItem/Article/SinglePrice
QTYI-1	/Order/LineItems/LineItem/Article/Amount
LEXC	/Order/LineItems/LineItem/Article/Price
LEXC*, VATP*	/Order/LineItems/LineItem/Article/Tax

* Die Quelldaten werden vor dem Schreiben in die Zielkomponente mit Hilfe von Funktionen verarbeitet
** Diese Felder gestatten die Iteration durch eine Gruppe von Feldern desselben Typs

Anhand der obigen Tabelle können Sie nun von der Quellkomponente Mapping-Verbindungen von allen oben aufgelisteten Einträgen zu Einträgen in der Zielkomponente ziehen. Lassen Sie vorerst diejenigen Felder, für die Werte berechnet werden müssen (d.h. diejenigen, die in der Tabelle mit * gekennzeichnet sind) aus; diese werden später in den nächsten Schritten behandelt. Allgemeine Informationen zum Erstellen von Verbindungen in MapForce, finden Sie unter Arbeiten mit Verbindungen 89.

Tipp: Um in der MapForce-Komponente nach einem Feld zu suchen, drücken Sie **Strg + F** und geben Sie den Namen des gesuchten Felds ein.

Schritt 2: Hinzufügen der Funktionen zur Konvertierung des Datums

Um den Wert des Datenelements `IVDT` von einem Dezimalwert in den Typ `xs:dateTime` zu konvertieren, wird im Mapping die Funktion **`format-number`** aus der **`core | conversion functions`** 1034-Bibliothek verwendet. Ein Beispiel dazu finden Sie unter Beibehalten vorangestellter Nullen bei der Konvertierung 521.

Die Aufgabe der zweiten Funktion, **`parse-date`**, ist, den von der **`format-number`**-Funktion zurückgegebenen YYMMDD-String-Wert in den Typ `xs:dateTime` zu konvertieren. Die Funktion hat zwei Input-Argumente: (a) den zu formatierenden Wert und (b) die Formatmaske. Anhand der Formatmaske liest MapForce das Jahr, Monat und Datum als zwei Zeichen für jeden dieser Bestandteile.

Um die Funktionen oberhalb des Mappings einzufügen, ziehen Sie diese aus dem Fenster "Bibliotheken" in den Mapping-Bereich. Um ein Argument für eine Funktion zu definieren, fügen Sie eine Konstante ein

 © 2016-2022 Altova GmbH

(Menübefehl **Einfügen | Konstante**). Allgemeine Informationen zum Arbeiten mit Funktionen in MapForce finden Sie unter Funktionen[942].

Schritt 3: Hinzufügen der Funktionen zur Berechnung der Umsatzsteuer

Zur Berechnung der Umsatzsteuer entsprechend der Formel werden die Funktionen **multiply** und **round** aus der Bibliothek **core | math functions**[1062] verwendet. Die multiply-Funktion kann erweitert werden (d.h. kann beliebig viele Argumente erhalten). In diesem Beispiel erhält die Funktion LEXC, VATP und die Ganzzahl 10 als Argumente. Die ersten beiden Werte werden von der TRADACOMS-Quellkomponente aus verbunden, während die Ganzzahl von einer Konstante geliefert wird. Das Ergebnis der **multiply**-Funktion wird anschließend mit Hilfe der round-Funktion auf die nächste Ganzzahl gerundet. Das Ergebnis der **round**-Funktion wird schließlich mit Hilfe der zweiten multiply-Funktion mit 0,001 multipliziert. Dieses Ergebnis wird anschließend mit dem gewünschten Datenelement der XML-Zielkomponente verbunden. (/Order/LineItems/LineItem/Article/Tax).

Eine Vorschau der Ausgabe des Mapping-Designs sehen Sie, wenn Sie im unteren Bereich des Mapping-Fensters auf das Register **Ausgabe** klicken.

Schritt 4 (optional): Darstellen der Mapping-Ausgabe mit Hilfe von StyleVision

Sie können die XML-Zielkomponente optional mit einer StyleVision Power StyleSheet (.sps)-Datei verknüpfen. (Nähere Informationen zu StyleVision finden Sie unter https://www.altova.com/de/stylevision.html.) Dadurch können Sie die Mapping-Ausgabe im HTML-, RTF-, PDF- oder 2007+-Format darstellen (Voraussetzung ist, dass StyleVision installiert ist und die für die einzelnen Formate erforderlichen Voraussetzungen gegeben sind, siehe Darstellen der Mapping-Ausgabe mit StyleVision[1395]). Um diesen optionalen Verarbeitungsschritt zu definieren, doppelklicken Sie auf die Überschrift der XML-Komponente und navigieren Sie neben dem Eintrag "StyleVision Power StyleSheet-Datei", zur Datei **Order.sps** im MapForce Examples-Ordner. Im Mapping-Bereich können Sie nun auf die Register HTML, PDF, RTF und Word 2007+ klicken, um die Mapping-Ausgabe im entsprechenden Format zu sehen.

4.3.17 UN/EDIFACT

UN/EDIFACT (United Nations rules for Electronic Data Interchange for Administration, Commerce and Transport) ist eine Gruppe von Standards für den elektronischen Austausch von Dokumenten. MapForce unterstützt die folgenden UN/EDIFACT-Verzeichnisse: D.93A oder später. Die neueste UN/EDIFACT-Version ist die in MapForce installierte Standardversion.

In den Altova MapForce-Bibliotheken finden Sie separate Installer für alle früheren Versionen bis zurück zu Version D.93A, einschließlich der für interaktive EDI-Mappings erforderlichen Konfigurationsdateien.

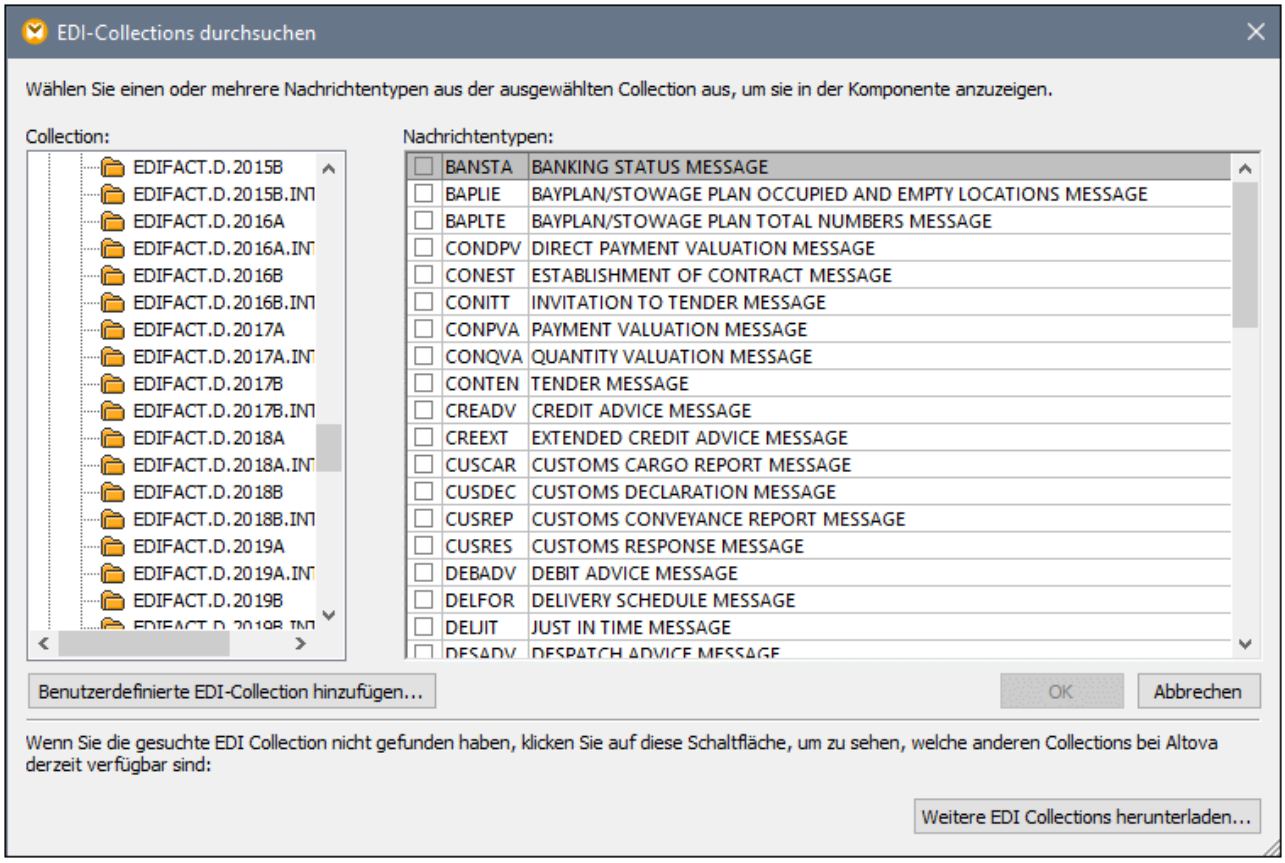
4.3.17.1 Batch im Gegensatz zu Interactive EDIFACT Messages

MapForce unterstützt das Mappen von Daten von und auf Batch-Nachrichten und Interactive UN/EDIFACT Messages, d.h. die UN/EDIFACT-Komponente in MapForce stellt die Struktur zur Verfügung, die zum Lesen und Schreiben syntaktische gültiger UN/EDIFACT-Formate (sowohl für den Batch-Austausch als auch für den interaktiven Datenaustausch) benötigt werden. Außerdem werden beim Lesen oder Schreiben von Dateien im UN/EDIFACT-Format einige grundlegende Validierungs- und Autokomplettierungsregeln angewendet, wie weiter unten in dieser Dokumentation beschrieben. Beachten Sie, dass MapForce die Dateien zwischen den am UN/EDIFACT-Datenaustausch beteiligten Parteien nicht überträgt. Die Aufgabe von MapForce ist lediglich die Generierung solcher Dateien oder die Konvertierung dieser Dateien in andere Formate.

Es gibt keine speziellen Anforderungen in MapForce, um Daten von oder auf EDIFACT-Batch-Dateien zu mappen. Dennoch müssen für das Mappen von Daten von oder auf Interactive EDIFACT Messages die folgenden Voraussetzungen erfüllt sein:

1. Es wird MapForce 2018 oder höher benötigt.
2. Die für die Unterstützung von Interactive EDI Messages benötigten MapForce-Konfigurationsdateien müssen separat installiert werden. Die Installationsdatei kann von der Altova MapForce-Bibliotheken-Seite kostenlos heruntergeladen und installiert werden. Die erforderlichen Collections und Nachrichtentypen stehen anschließend im unten gezeigten Dialogfeld **EDI-Collections durchsuchen** zur Verfügung (*siehe Abbildung unten*).

Um entweder eine Batch-Komponente oder eine Interactive UN/EDIFACT-Komponente zum Mapping hinzuzufügen, wählen Sie die entsprechende Collection im Dialogfeld **EDI-Collections durchsuchen** aus. Dieses Dialogfeld können Sie über den Menübefehl **Einfügen | EDI** oder über die Symbolleisten-Schaltfläche **EDI-Datei einfügen** aufrufen. Nähere Informationen dazu finden Sie unter Hinzufügen von EDI Komponenten zum Mapping 463.



Anmerkung: Die Dateien im Dialogfeld oben mit dem Suffix "INTERACTIVE" bezeichnen eine Collection, die für eine Interactive EDIFACT-Struktur verwendet werden kann. Collections die für Batch-Strukturen verwendet werden können, enthalten in ihrem Titel nur die UN/EDIFACT-Verzeichnisversion (z.B. EDIFACT.D.2016A). Dieselbe Konvention gilt für die Konfigurationsverzeichnisse (Collections) im EDI-Konfigurationsordner von MapForce: `C:\Programme\Altova\MapForce2023\MapForceEDI`.

 © 2016-2022 Altova GmbH

Die Struktur von UN/EDIFACT-Batch-Dateien unterscheidet sich von der von Dateien, die für den interaktiven Datenaustausch gedacht sind. Eine EDIFACT-Komponente in MapForce kann daher entweder eine Batch- ODER eine Interactive EDIFACT-Komponente (und daher entweder Batch- ODER Interactive Messages enthalten), nicht aber beides gleichzeitig sein. Auch die Validierungs- und Autokomplettierungsregeln für die beiden Szenarien unterscheiden sich, wie in den folgenden Kapiteln näher erläutert:

- UN/EDIFACT-Validierungsregeln (Batch) 529
- UN/EDIFACT-Autokomplettierungsregeln (Batch) 529
- UN/EDIFACT-Validierungsregeln (Interactive) 530
- UN/EDIFACT-Autokomplettierungsregeln (Interactive) 531

4.3.17.2 UN/EDIFACT-Validierungsregeln (Batch)

Die folgenden Punkte werden bei der Validierung eines UN/EDIFACT-Dokuments überprüft.

- ob ein UNB- und ein UNZ-Segment existieren.
- ob UNB/S004 eine gültige Datums-/Uhrzeit-Angabe enthält.
- ob UNB/0020 und UNZ/0020 denselben Wert enthalten.
- ob UNZ/0036 die richtige Zahl, die der Anzahl der Funktionsgruppen, falls vorhanden, bzw. der Anzahl der Messages entspricht. Wenn Funktionsgruppen vorhanden sind, sollte dies die Anzahl der Funktionsgruppen sein, andernfalls sollte dies die Anzahl der im Datenaustausch (Interchange) enthaltenen Messages sein.

Jede Funktionsgruppe wird auf folgende Punkte überprüft:

- ob sie ein übereinstimmendes UNG- und UNE-Paar enthält.
- ob UNG/S004 eine gültige Datums-/Uhrzeit-Angabe enthält.
- ob UNE/0060 die richtige Zahl enthält, die der Anzahl der Messages in der Funktionsgruppe entsprechen muss.

Jede Message wird auf folgende Punkte überprüft:

- ob sie ein übereinstimmendes UNH- und UNT-Paar enthält.
- ob UNH/S009/0052 denselben Wert enthält wie UNG/S008/0052 der einschließenden Funktionsgruppe.
- ob UNH/0062 und UNT/0062 denselben Wert enthalten.
- ob UNH/S009/0065 den korrekten Message Type Specifier enthält.
- ob UNT/0074 die korrekte Anzahl der in der Message enthaltenen Segmente enthält.

4.3.17.3 UN/EDIFACT-Autokomplettierungsregeln (Batch)

MapForce kann bei der Generierung von UN/EDIFACT-Dateien anhand der unten aufgelisteten Regeln (die in der Folge als _"Regeln zur automatischen Ergänzung von Daten" bzw. "Autokomplettierungsregeln" bezeichnet werden) automatisch bestimmte Felder oder Werte erstellen. Um dieses Verhalten zu deaktivieren, entfernen Sie in den EDI-Komponenteneinstellungen 466 das Häkchen aus dem Kontrollkästchen **Autokomplettierung von fehlenden Feldern.**

Die automatische Datenkomplettierung für EDIFACT stellt sicher:

- dass ein UNB- und ein UNZ-Segment existieren.
- dass, wenn entweder UNG oder UNE vorhanden ist, die andere ID ebenfalls vorhanden ist.

- dass ein UNH- und ein UNT-Segment vorhanden sind.
- dass UNB/S001 existiert. Wenn es keine Daten enthält, werden Syntax Level und Syntax Versionsnummer aus den benutzerdefinierten Einstellungen verwendet. Siehe **Erweitert | Syntax Version Number** der EDI-Komponenteneinstellungen[466]).
- dass UNB/S002 und UNB/S003 vorhanden sind.
- dass UNB/S004 vorhanden ist. Wenn es keine Daten enthält, wird das aktuelle Datum/die aktuelle Uhrzeit im EDI-Format eingefügt.
- dass UNZ/0036 vorhanden ist. Wenn es keine Daten enthält, wird die Anzahl der Funktionsgruppen und Messages berechnet und eingefügt.
- dass UNZ/0020 vorhanden ist. Wenn es keine Daten enthält, wird der Wert aus UNB/0020 kopiert.

Bitte beachten Sie:

Felder, die in diesem Abschnitt (Automatische Datenkomplettierung) nicht erwähnt sind, werden nicht eingefügt oder erstellt. Die korrekten Werte können nicht automatisch ermittelt werden. Bei einem Parent-(Ziel)-Element A (in der EDI-Zielkomponente) mit den Child-Elementen x, y und z - wobei y obligatorisch ist - wird das Parent-Element A in der Ausgabedatei nur dann erstellt, wenn das obligatorische Child-Element "y" in der Zielkomponente gemappt wurde!

Bei der Überprüfung der Funktionsgruppen wird sichergestellt:

- dass UNG/0038 vorhanden ist. Wenn es keine Daten enthält, wird der Name der Message eingefügt.
- dass UNG/S006 und UNG/S007 vorhanden sind.
- dass UNG/S004 vorhanden ist. Wenn es keine Daten enthält, wird das aktuelle Datum/die aktuelle Uhrzeit im EDI-Format eingefügt.
- dass UNG/0051 vorhanden ist. Wenn es keine Daten enthält, werden aus den benutzerdefinierten Einstellungen die ersten beiden Zeichen oder Controlling Agency eingefügt. (Siehe **Einstellungen | Controlling agency** der EDI-Komponenteneinstellungen[466])
- dass UNE/0060 vorhanden ist. Wenn es keine Daten enthält, wird die Anzahl der Messages in der Gruppe berechnet und eingefügt.
- dass UNE/0048 vorhanden ist. Wenn es keinen Wert enthält, wird der Wert aus UNG/0048 kopiert.

Bei der Überprüfung der Messages wird sichergestellt:

- dass UNH/S009/0065 vorhanden ist. Wenn es keine Daten enthält, wird der Name der Message eingefügt.
- dass UNH/S009/0052 und UNH/S009/0054 vorhanden sind.
- dass UNH/S009/0051 vorhanden ist. Wenn es keine Daten enthält, werden aus den benutzerdefinierten Einstellungen die ersten beiden Zeichen oder Controlling Agency eingefügt. Siehe **Einstellungen | Controlling agency** der EDI-Komponenteneinstellungen[466]).
- dass UNT/0074 vorhanden ist. Wenn es keine Daten enthält, wird die Anzahl der Segmente in der Message berechnet und eingefügt.
- dass UNT/0062 vorhanden ist. Wenn es keinen Wert enthält, wird der Wert aus UNH/0062 kopiert.
- Dass UNH/0062 vorhanden ist. Wenn es keinen Wert enthält, wird der Wert aus UNT/0062 kopiert. (Wenn nur die Nummer des Trailer Segments gemappt wird, erhält das entsprechende Feld im Header-Segment denselben Wert)

4.3.17.4 UN/EDIFACT-Validierungsregeln (Interactive)

Die folgenden Validierungsregeln werden auf MapForce-Komponenten, die Interactive EDIFACT Messages enthalten, angewendet:

© 2016-2022 Altova GmbH

- Wenn `UIB` vorhanden ist, müssen alle folgenden `UIH/S302`-Segmente mit `UIB/S302` übereinstimmen.
- `UIH/S306/F0065` sollte den Nachrichtentyp enthalten (validiert vom Parser).
- `UIH/S306/F0052` sollte die in den ausgewählten Konfigurationsdateien definierte Nachrichtenversionsnummer enthalten.
- `UIH/S306/F0054` sollte die in den ausgewählten Konfigurationsdateien definierte Nachrichten-Release-Nummer enthalten.
- `UIT/F0340` sollte mit dem entsprechenden `UIH/F0340`-Segment (Nachrichten-Header/Trailer) übereinstimmen oder nicht vorhanden sein (optionales Feld).
- `UIT/F0074` sollte die Segmentanzahl enthalten oder nicht vorhanden sein (optionales Feld).
- `UIZ/S302` sollte mit `UIB/S302` übereinstimmen oder nicht vorhanden sein (optionales Komposit).
- `UIZ/F0036` sollte die Nachrichtenanzahl enthalten oder nicht vorhanden sein (optionales Feld).

4.3.17.5 UN/EDIFACT-Autokomplettierungsregeln (Interactive)

Die folgenden Autokomplettierungsregeln werden auf MapForce-Komponenten, die Interactive EDIFACT Messages enthalten, angewendet:

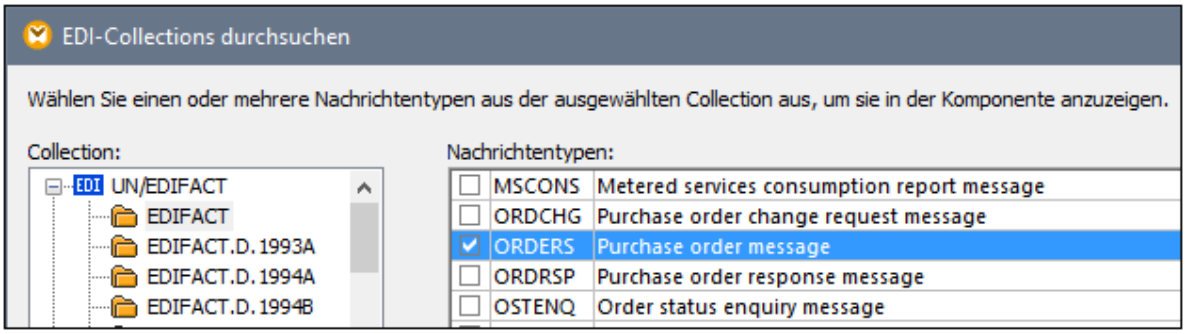
- `UIB/S001/F0001` wird mit der Kontrollagentur + Syntaxversionsnummer befüllt.
- `UIB/S001/F0002` wird mit der Syntaxversionsnummer befüllt.
- `UIB/S002/F0004` wird mit dem "Absender" befüllt.
- `UIB/S003/F0010` wird mit dem "Empfänger" befüllt.
- `UIZ/S302/*` wird mit `UIB/S302/*` befüllt.
- `UIZ/F0036` wird mit der Nachrichtenanzahl befüllt.
- `UIH/S302/*` wird mit `UIB/S302/*` befüllt (falls `UIB` und `UIB/S302` vorhanden sind).
- `UIH/S306/F0065` wird mit dem Nachrichtentyp befüllt.
- `UIH/S306/F0052` wird mit der Nachrichtenversionsnummer aus den Konfigurationsdateien befüllt.
- `UIH/S306/F0054` wird mit der Nachrichten-Release-Nummer aus den Konfigurationsdateien befüllt.
- `UIT/F0340` wird mit `UIH/F0340` befüllt (Dies ist ein optionales Feld, das dennoch ausgefüllt wird).
- `UIT/F0074` wird mit der Anzahl der Segmente in der Nachricht befüllt (Dies ist ein optionales Feld, das dennoch ausgefüllt wird).

4.3.17.6 Beispiel: Mappen von UN/EDIFACT auf XML

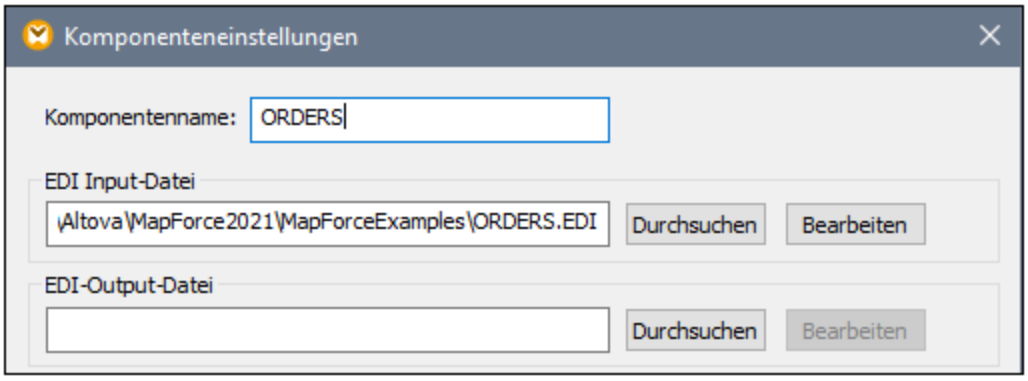
In diesem Beispiel wird beschrieben, wie Daten von UN/EDIFACT-Nachrichten auf ein XML-Schema gemappt werden, um eine XML-Instanzdatei zur weiteren Verarbeitung zu erzeugen. Sie finden die Beispieldatei dazu unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\EDI_Order.mfd**.

Schritt 1: Hinzufügen der UN/EDIFact-Komponenten zum Mapping

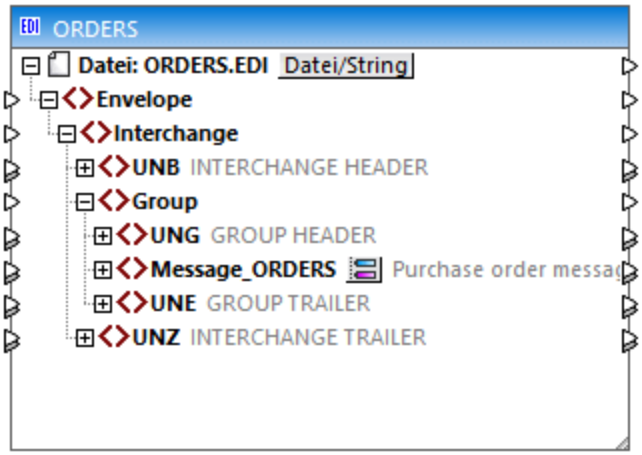
1. Erstellen Sie ein neues Mapping und wählen Sie eine der folgenden Transformationssprachen aus: Java, C#, C++ oder BUILT-IN. In diesem Beispiel wurde Java als Transformationssprache ausgewählt.
2. Wählen Sie im Menü **Einfügen** den Befehl **EDI.**
3. Wählen Sie im Dialogfeld "EDI-Collections" die EDIFACT Collection und anschließend den Nachrichtentyp **ORDERS** aus und klicken Sie auf **OK**.



4. Wenn Sie aufgefordert werden, eine EDI-Beispieldatei anzugeben, klicken Sie auf **Durchsuchen** und öffnen Sie die Datei **ORDERS.EDI** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Nach dem Öffnen der Datei wird das Dialogfeld "Komponenteneinstellungen" angezeigt. Auf diese Art können Sie die Einstellungen der EDI-Komponente überprüfen, bevor Sie diese zum Mapping hinzufügen. Sie können diese Einstellungen später bei Bedarf jederzeit ändern (siehe EDI-Komponenteneinstellungen 466). Beachten Sie, dass die Datei **ORDERS.EDI** als Input-Datei angezeigt wird.



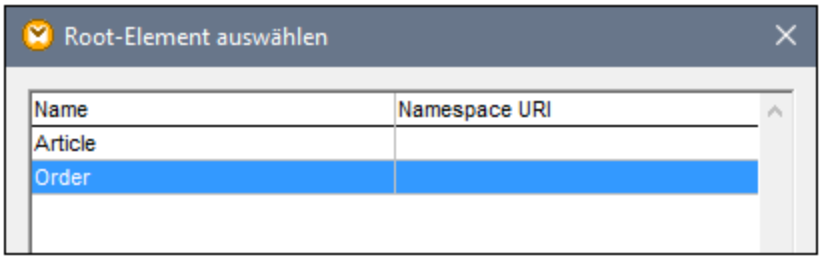
5. Klicken Sie auf **OK**. Die EDI-Komponente wird nun im Mapping-Bereich angezeigt. Doppelklicken Sie auf den Node **Message_ORDERS,** um die Subeinträge zu sehen. Um die Größe der Komponente anzupassen, ziehen Sie die rechte untere Ecke des Komponentenfensters mit der Maus.



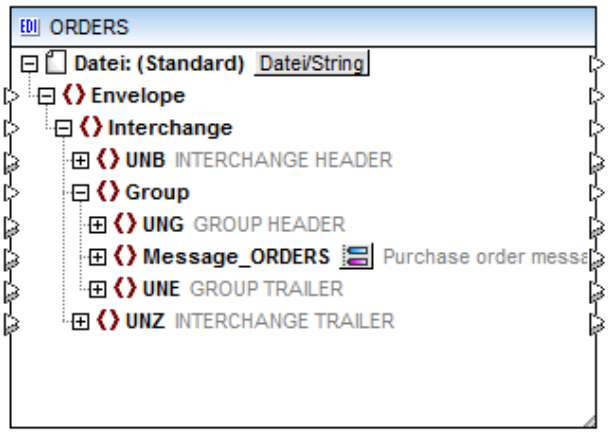
 © 2016-2022 Altova GmbH

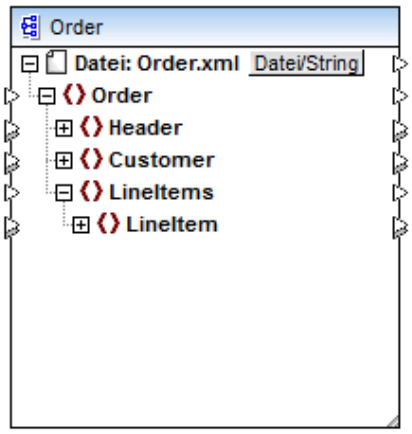
Schritt 2: Hinzufügen der Zielschemakomponente zum Mapping

1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und öffnen Sie die Datei **Order.xsd** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**.
2. Wenn Sie aufgefordert werden, eine XML-Beispieldatei anzugeben, klicken Sie auf **Überspringen** und wählen Sie als Root des Zieldokuments den Eintrag **Order** aus.



Zu diesem Zeitpunkt befindet sich nun sowohl die EDI-Quellkomponente als auch die XML-Schema-Zielkomponente im Mapping-Bereich, so dass Sie die Mapping-Verbindungslinien ziehen können.





Schritt 3: Mappen der EDI-Datenelemente

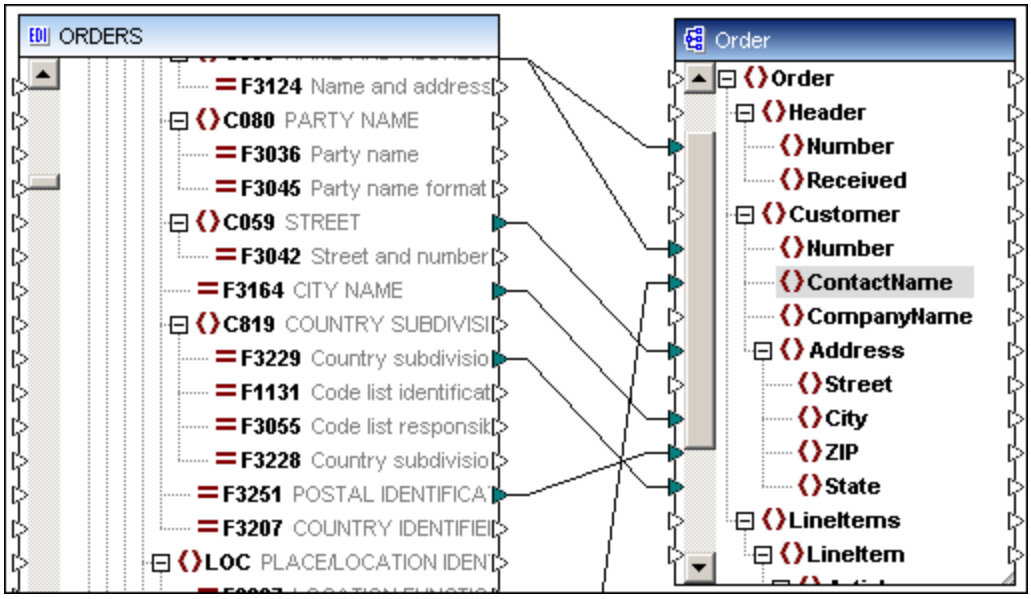
In der EDI-Komponente wird nun die Struktur einer Nachricht angezeigt. Diese Struktur basiert auf der soeben ausgewählten Collection (ORDERS). Normalerweise enthalten nicht alle Nodes Daten, daher müssen Sie das jeweils verarbeitete EDI-Dokument gut genug kennen, um die relevanten Nodes zu finden.

In MapForce erfolgt das Mapping eines Quell-Datenelements auf ein Ziel-Datenelement durch Ziehen einer Verbindung zwischen den beiden Elementen. Eine schrittweise Anleitung zum Erstellen von Mapping-Verbindungen in MapForce finden Sie unter Arbeiten mit Verbindungen 89.

In diesem Beispiel müssen die folgenden Nodes (beginnend mit dem **Group/Message_ORDERS** Node) gemappt werden:

Quelle	Ziel
BGM/C106/F1004	Order/Header/Number
SG2/NAD/C082/F3039	Order/Customer/Number
SG2/NAD/C080/F3036	Order/Customer/CompanyName
SG2/NAD/C059/F3042	Order/Customer/Address/Street
SG2/NAD/F3164	Order/Customer/Address/City
SG2/NAD/C819/F3229	Order/Customer/Address/State
SG2/NAD/F3251	Order/Customer/Address/ZIP
SG2/SG5/CTA/C056/F3412	Order/Customer/ContactName

In dieser Phase sollte das Mapping ähnlich wie in der unten gezeigten Abbildung aussehen:



Fahren Sie mit dem Mapping fort und mappen Sie:

Quelle	Ziel
SG29	Order/LineItems
SG29/LIN/C212/F7140	Order/LineItems/LineItem/Article/Number
SG29/IMD/C273/F7008	Order/LineItems/LineItem/Article/Name

© 2016-2022 Altova GmbH

Quelle	Ziel
SG29/QTY/C186/F6060	Order/LineItems/LineItem/Article/Amount
SG29/SG33/PRI/C509/F5118	Order/LineItems/LineItem/Article/SinglePrice

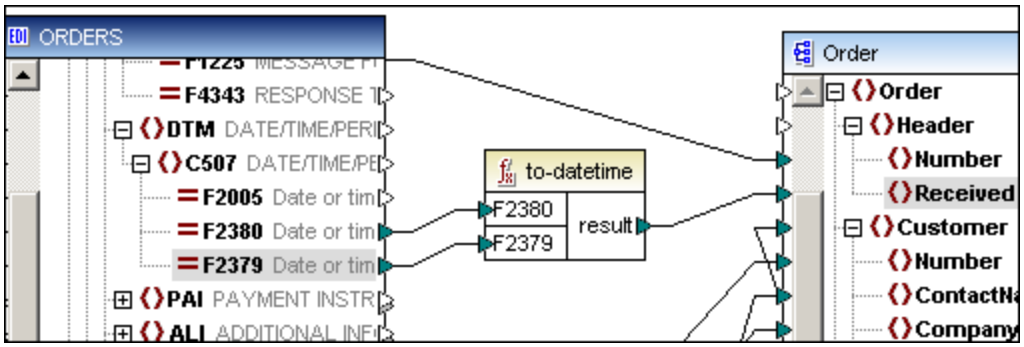
Schritt 4: Formatieren des Datums

Ziehen Sie die **to-datetime**-Funktion aus der **edifact**-Bibliothek in das Mapping-Fenster. Eine Anleitung zum Arbeiten mit Funktionen in MapForce finden Sie unter Hinzufügen einer Funktion zum Mapping 943.

Durch Bereitstellung der Komponenten F2380 und F2379 des Elements DTM/C507 als Argumente lässt sich ein ordnungsgemäß formatiertes datetime-Element **Received** erzeugen.

Wir mappen daher die folgenden Felder:

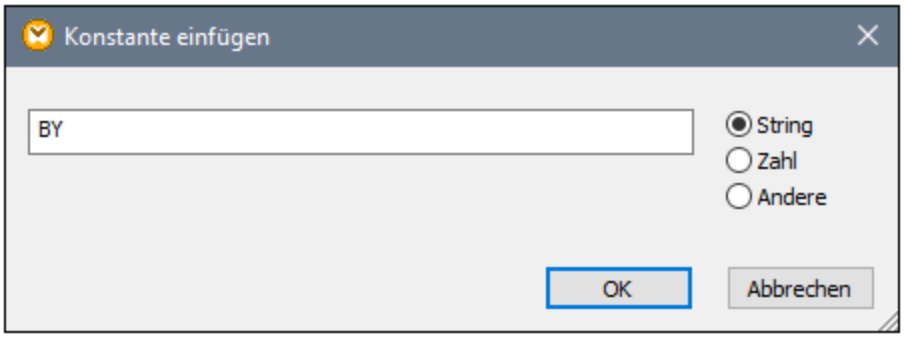
Quelle	Ziel
DTM/C507/F2380	Der F2380-Input der **to-datetime**-Funktion
DTM/C507/F2379	Der F2379-Input der **to-datetime**-Funktion
Das Ergebnis (**result)** der **to-datetime**-Funktion	Order/Header/Received



Schritt 5: Filtern der Bestellungen des Käufers (Buyer):

Wir wollen nun die Bestellungen des Käufers (Buyer) filtern. Diese lassen sich anhand des "party function code qualifier" des NAD (Name and Address)-Segments identifizieren. In diesem Fall bedeutet der Wert 'BY' einen Buyer, also Käufer (Partei an den eine Ware und/oder Dienstleistung verkauft wird).

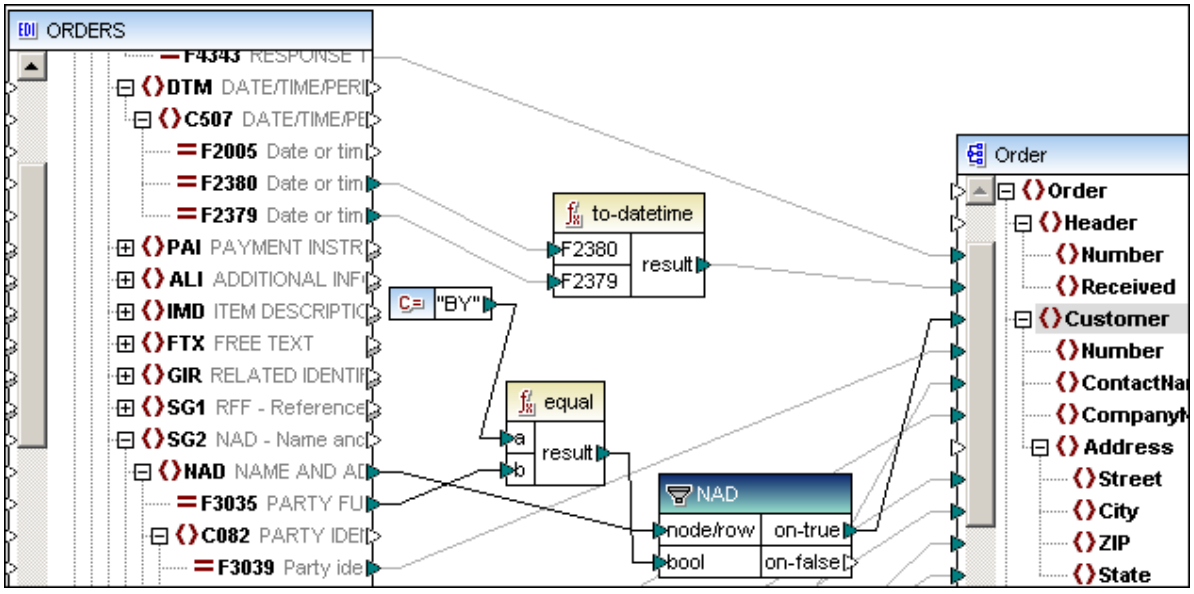
1. Ziehen Sie die equal-Funktion aus der **core-**Bibliothek in das Mapping-Fenster.
2. Fügen Sie einen Filter zum Mapping hinzu. (Wählen Sie die Menüoption **Einfügen | Filter: Nodes/Zeilen)**.
3. Fügen Sie eine Konstante zum Mapping hinzu. (Wählen Sie dazu die Menüoption **Einfügen | Konstante).** Weisen Sie der Konstante den Wert "BY" zu, indem Sie in das Textfeld "BY" eingeben.



Mappen Sie die folgenden Datenelemente:

Quelle	Ziel
`SG2/NAD/F3035`	Der Input **b** der `equal`-Funktion
Die Konstante "BY"	Der Input **a** der `equal`-Funktion
Das Ergebnis der `equal`-Funktion	Der Input **bool** der **filter**-Komponente
`SG2/NAD`	Der **node/row**-Input der Filterkomponente
Das **on-true** Ergebnis der **filter**-Komponente	`Order/Customer` im Schema

Ziel ist, nur Daten zu mappen, deren NAD auf einen 'Buyer' referenziert, wie vom "party function code qualifier" 'BY' gekennzeichnet.



 © 2016-2022 Altova GmbH

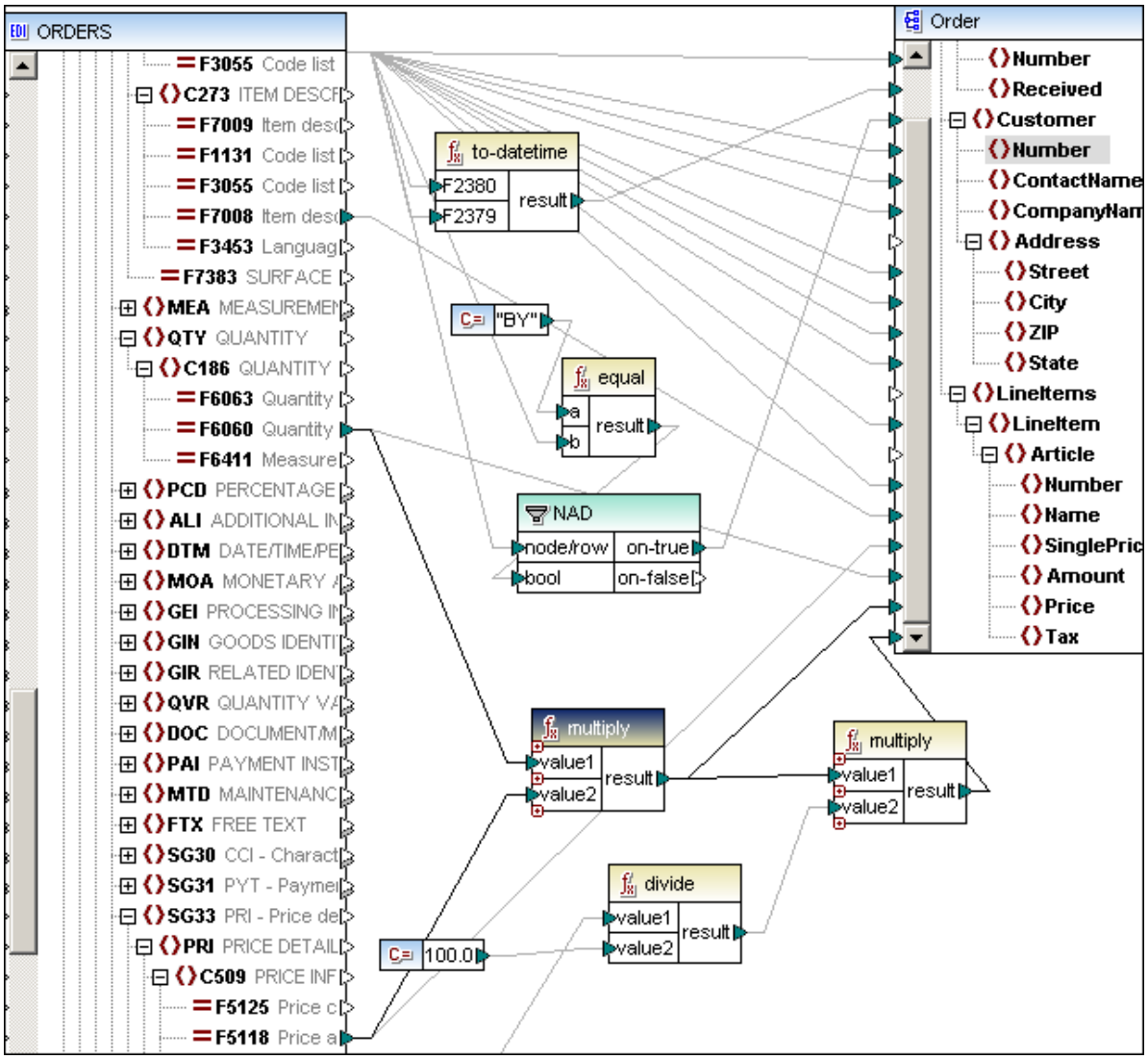
Schritt 6: Berechnen von Preis und Steuer

Der letzte Schritt in diesem Vorgang ist nun die Berechnung des Preises und der Steuern.

1. Ziehen Sie aus dem Fenster "Bibliotheken" zwei **multiply**- und eine **divide**-Funktion (oder der `core`-Bibliothek) in das Mapping-Fenster.
2. Fügen Sie eine Konstantenkomponente ein (**Einfügen | Konstante**). Stellen Sie sicher, dass als Typ "Numerisch" ausgewählt ist und geben Sie im Textfeld 100.0 ein.
3. Mappen Sie die folgenden Elemente:

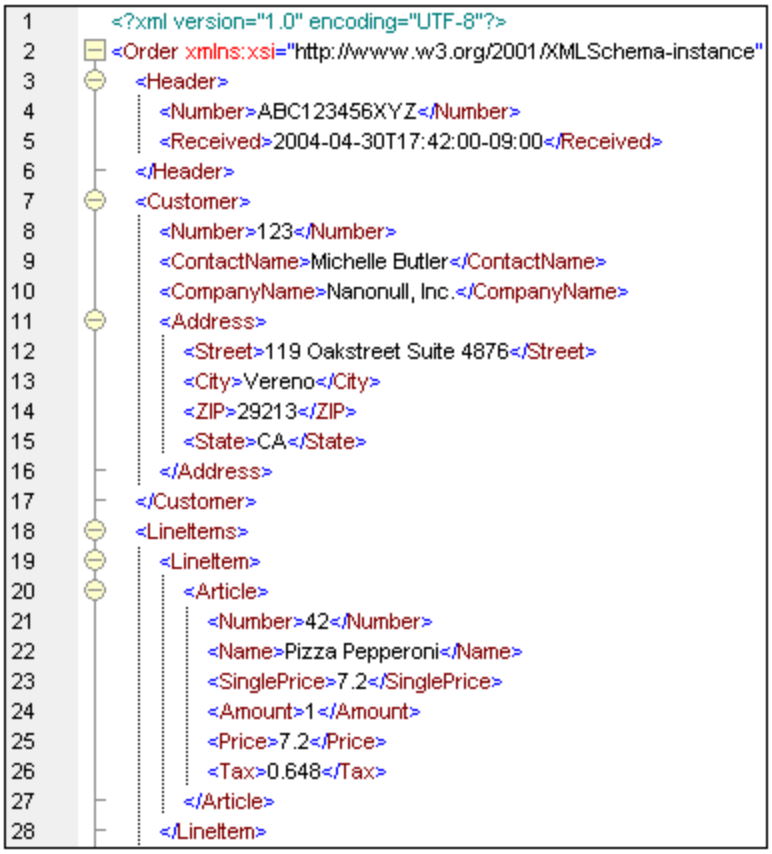
Quelle	Ziel
`SG29/QTY/C186/F6060`	**value1** der ersten **`multiply`**-Funktion
`SG29/SG33/PRI/C509/F5118`	**value2** der ersten **`multiply`**-Funktion
Das Ergebnis der ersten **`multiply`**-Funktion	`Order/LineItems/LineItem/Article/Price`
`SG29/SG40/TAX/C243/F5278`	**value1** der **`divide`**-Funktion
Die Konstante "100.0"	**value2** der **`divide`**-Funktion
Das Ergebnis der ersten **`multiply`**-Funktion	**value1** der zweiten **`multiply`**-Funktion
Das Ergebnis der **`divide`**-Funktion	**value2** der zweiten **`multiply`**-Funktion
Das Ergebnis der zweiten **`multiply`**-Funktion	`Order/LineItems/LineItem/Article/Tax`

Ihr Mapping sollte nun folgendermaßen aussehen:



Wenn Sie auf die Schaltfläche Ausgabe klicken, wird eine on-the-fly-Transformation durchgeführt und das erzeugte XML-Dokument angezeigt:

© 2016-2022 Altova GmbH



```
<?xml version="1.0" encoding="UTF-8"?>
<Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <Header>
    <Number>ABC123456XYZ</Number>
    <Received>2004-04-30T17:42:00-09:00</Received>
  </Header>
  <Customer>
    <Number>123</Number>
    <ContactName>Michelle Butler</ContactName>
    <CompanyName>Nanonull, Inc.</CompanyName>
    <Address>
      <Street>119 Oakstreet Suite 4876</Street>
      <City>Vereno</City>
      <ZIP>29213</ZIP>
      <State>CA</State>
    </Address>
  </Customer>
  <LineItems>
    <LineItem>
      <Article>
        <Number>42</Number>
        <Name>Pizza Pepperoni</Name>
        <SinglePrice>7.2</SinglePrice>
        <Amount>1</Amount>
        <Price>7.2</Price>
        <Tax>0.648</Tax>
      </Article>
    </LineItem>
```

4.3.18 Anpassen der EDI-Struktur

Standardmäßig wird die Struktur von EDI-Dateien in MapForce Mappings gemäß ihren formalen Spezifikationen dargestellt. Meist ist es nicht notwendig, die EDI-Standardkonfiguration manuell anzupassen. Wenn Sie jedoch verschiedene Nicht-Standard-EDI-Formate und -Varianten verarbeiten müssen, so ist eine Anpassung erforderlich. So müssen Sie möglicherweise die Struktur von Validierungsregeln von bestimmten Datenelementen eventuell etwas anpassen. Fügen Sie zu diesem Zweck eine benutzerdefinierte EDI-Collection zu MapForce hinzu und konfigurieren Sie diese anschließend Ihren Bedürfnissen entsprechend.

Sie können für die folgenden von MapForce bereits unterstützten Standards EDI-Varianten (benutzerdefiniere EDI-Collections) erstellen: EDIFACT (ISO 9735), ASC X12, HIPAA X12, HL7, NCPDP SCRIPT, SWIFT MT oder TRADACOMS. Die Erstellung eines neuen EDI-Standards von Grund auf wird nicht unterstützt.

4.3.18.1 Erstellen einer benutzerdefinierten EDI-Collection

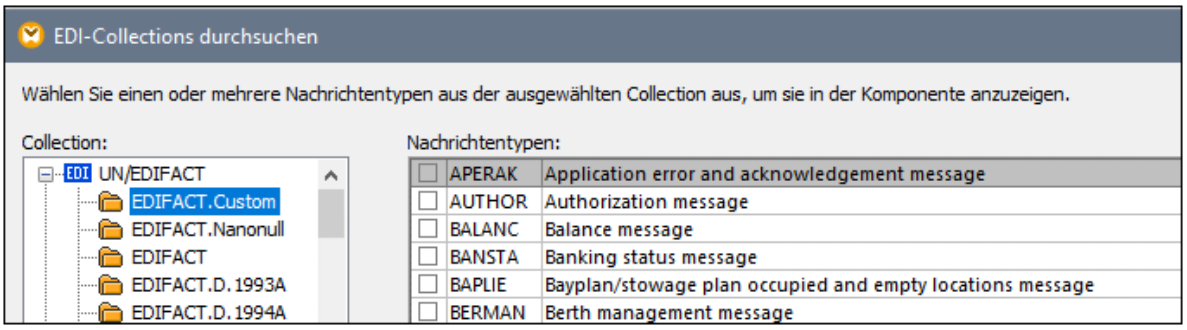
Eine "EDI-Collection" ist ein Verzeichnis, das alle Konfigurationsdateien zu einem bestimmten EDI-Standard enthält. Jede EDI-Collection enthält eine Datei namens "EDI.Collection" und verschiedene andere davon entweder direkt oder indirekt referenzierte Dateien. Das letzte Verzeichnis im Pfad ist der Name der EDI-Collection, wie er auf der grafische Benutzeroberfläche von MapForce angezeigt wird, wenn Sie eine EDI-Komponente zum Mapping hinzufügen[463] (z.B. "UN/EDIFACT", "HL7, "ASC X12", usw.).

Nach Installation von MapForce steht unter dem folgenden Pfad eine Reihe von vordefinierten EDI-Collections zur Verfügung: **C:\Programme\Altova\MapForce2023\MapForceEDI**. Wenn Sie eine 32-Bit-MapForce-Version auf einem 64-Bit-Betriebssystem ausführen, ändern Sie den Verzeichnispfad in **C:\Programme (x86)\Altova\MapForce2023\MapForceEDI**.

> Standardmäßig enthält das obige Verzeichnis nur die vordefinierten EDI-Collections. Sie können alle unterstützte Versionen der jeweiligen EDI-Collection vom Altova Download Center (https://www.altova.com/de/mapforce/download#mapforce_libraries) herunterladen.

EDI-Collections können in MapForce auf drei Arten erstellt werden:

1. Erstellen Sie eine benutzerdefinierte EDI-Collection in folgenden Verzeichnis: **C:\Benutzer\<Name>\Dokumente\Altova\MapForce2023\MapForceEDI** Wenn Sie z.B. **C:\Programme\Altova\MapForce2023\MapForceEDI\EDIFACT** in **C:\Benutzer\<Name>\Dokumente\Altova\MapForce2023\MapForceEDI\EDIFACT.Custom** kopieren, scheint die "EDIFACT.Custom" Collection in der Liste der EDI-Collections auf, wenn Sie eine EDI-Komponente zum Mapping hinzufügen[463].



   Normalerweise sollte man Ihre benutzerdefinierten EDI-Collections von den vordefinierten EDI-Collections unterscheiden können. Aus diesem Grund sollten sie nicht denselben Namen wir die vordefinierten Collections haben. So sollte z.B. das Verzeichnis Ihrer benutzerdefinierten EDI-Collection nicht den Namen "EDIFACT" haben, da dieser Name bereits von der vordefinierten Collection unter **C:\Programme\Altova\MapForce2023\MapForceEDI\EDIFACT** verwendet wird.

   Wenn Ihre benutzerdefinierte EDI-Collection jedoch Vorrang vor einer Standard-EDI-Collection aus **C:\Programme\Altova\MapForce2023\MapForceEDI** erhalten soll, dann benennen Sie Ihre benutzerdefinierte EDI-Collection auf dieselbe Art wie die vordefinierte EDI-Collection.

2. Geben Sie ein benutzerdefiniertes Verzeichnis an, in dem Ihre EDI-Collection gespeichert werden soll. So können Sie z.B. **C:\Programme\Altova\MapForce2023\MapForceEDI\EDIFACT** in **C:\EDIFACT.Custom** kopieren. Klicken Sie als nächstes auf **Benutzerdefinierte EDI-Collection**

 © 2016-2022 Altova GmbH

hinzufügen, und navigieren Sie zu **C:\EDIFACT.Custom**. Um den Pfad der EDI-Collection relativ zur Mapping-(.mfd)-Datei zu machen, aktivieren Sie das Kontrollkästchen **Alle Dateipfad relativ zur MFD-Datei speichern**.

Auf diese Art hinzugefügte benutzerdefinierte EDI-Collections wirken sich nur auf die aktuelle EDI-Komponente im Mapping aus.

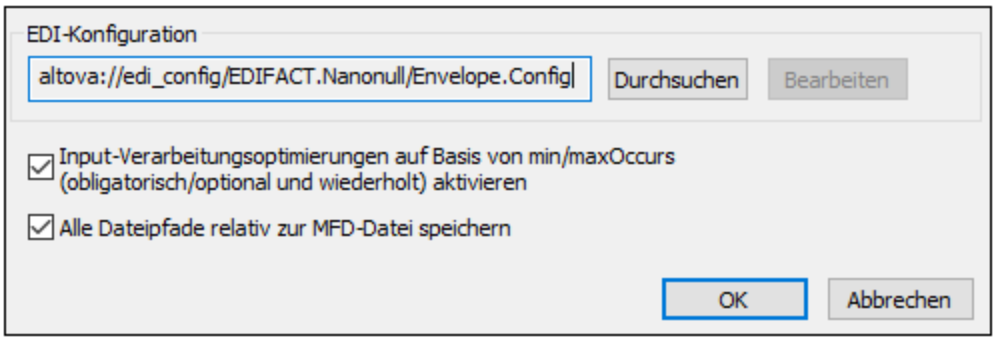
3. (Nicht empfohlen) Erstellen Sie Ihre benutzerdefinierte EDI-Collection im Verzeichnis **C:\Programme\Altova\MapForce2023\MapForceEDI**. Beachten Sie, dass Sie Administratorrechte benötigen, um an diesem Verzeichnis Änderungen vorzunehmen. Diese Methode wird nur aus Gründen der Rückwärts-Kompatibilität unterstützt und wird nicht empfohlen.

> Es wird nicht empfohlen, den Inhalt der Originaldateien und -ordner im Verzeichnis **C:\Programme\Altova\MapForce2023\MapForceEDI** zu ändern, es sei denn, Sie möchten die EDI-Standardvalidierung und -verarbeitung in MapForce auf eigene Verantwortung außer Kraft setzen. Verwenden Sie stattdessen entweder die erste oder zweite oben beschriebene Methode. Auf diese Art lässt sich Ihre benutzerdefinierte Konfiguration auch einfacher in eine neuere Version von MapForce migrieren.

Wie MapForce EDI-Konfigurationen auflöst

So zeigen Sie die von einer EDI-Komponente verwendete Konfiguration (EDI-Collection) an:

- Doppelklicken Sie im Mapping auf die Titelleiste der Komponente. (Klicken Sie alternativ dazu mit der rechten Maustaste auf die Titelleiste der Komponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften** aus). Daraufhin wird das Dialogfeld EDI-Komponenteneinstellungen 466 geöffnet.



Die Eigenschaft **EDI-Konfiguration** gibt den Pfad zu der von der Komponente verwendeten Konfiguration (EDI-Collection) an. Wenn dieser Pfad mit `altova://edi_config/` beginnt, sucht MapForce im folgenden Verzeichnis nach der entsprechenden Konfiguration (die Reihenfolge ist von Bedeutung):

1. Im Verzeichnis Ihres Benutzerprofils unter Windows (**C:\Benutzer\<Name>\Dokumente\Altova\MapForce2023\MapForceEDI**\). Wenn die Konfiguration unter diesem Pfad gefunden wird, erhält sie Vorrang vor einer aus **C:\Programme\...** (siehe nächster Punkt).
2. Im Verzeichnis **C:\Programme\Altova\MapForce2023\MapForceEDI**.

Wenn die Eigenschaft **EDI-Konfiguration** einen Standard-Windows-Pfad angibt, wird von der Komponente die EDI-Collection unter dem entsprechenden Pfad verwendet.

Beispiele

Schritt-für-Schritt-Beispiele für benutzerdefinierte EDI-Collections finden Sie unter:

- Beispiel: Hinzufügen eines TA1-Segments 558
- Beispiel: Anpassen einer EDIFACT Message 559
- Beispiel: Anpassen einer ANSI X12-Transaktion 566

4.3.18.2 EDI-Konfigurationsdateien

Eine EDI-Collection (Verzeichnis) kann jede der folgenden Konfigurationsdateien enthalten:

Datei	Aufgabe
EDI.Collection	In der **EDI.Collection**-Datei sind alle Nachrichtentypen aus der benutzerdefinierten EDI-Collection, die Sie implementieren, definiert. Sie können diese Datei bearbeiten, um Nachrichtentypen, die für die Auswahl in MapForce (im Dialogfeld "EDI-Collections durchsuchen") zur Verfügung stehen sollten, hinzuzufügen. Siehe auch Hinzufügen oder Entfernen von Nachrichtentypen 547.
Envelope.Config	In der **Envelope.config**-Datei ist die Struktur der EDI-Mapping-Komponente in MapForce (d.h. Elemente, die innerhalb der **Envelope**-Sequenz stehen) definiert. Normalerweise sollte diese Datei nicht manuell bearbeitet werden.
<Message>.Config	Das EDI-Collection-Verzeichnis muss eine oder mehrere **<Message>.Config-**Dateien enthalten. Eine solche Datei entspricht einem EDI-Nachrichtentyp und enthält die in dieser Nachricht verwendeten Gruppen- und Segmentdefinitionen. Die **<Message>.Config**-Dateien werden von der Datei **EDI.Collection** referenziert. Bei Bedarf können Sie die **<Message>.Config-Dateien** modifizieren, wenn Sie die Struktur oder die Validierungsregeln für die Felder einer bestimmten Nachricht ändern möchten. Beachten Sie, dass Änderungen an diesen Dateien als lokale Anpassungen betrachtet werden (d.h. sie gelten nur für diese spezifische Nachricht). Beispiele finden Sie unter Anpassen einer EDIFACT Message 559 und Anpassen einer ANSI X12-Transaktion 566.
<Dateiname>.Segment	In dieser Datei sind die Datenelemente, Segmente, Composites und Subcomposites der EDI-Dateien definiert. Sie dient zum Parsen der EDI-Datei. Sie können die .segment-Dateien modifizieren, wenn Sie die Struktur oder die Validierungsregeln für ein Feld (Datenelement) in allen Nachrichten, in denen diese vorkommen kann/können, ändern müssen. Änderungen an dieser Datei sind globale Anpassungen.
<Dateiname>.Codelist	In dieser Datei sind die von MapForce zum Validieren der EDI-Input- und Output-Dateien verwendeten EDI-Codewerte definiert. Im Allgemeinen entsprechen die hier definierten Codewerte den im jeweiligen EDI-Standard

© 2016-2022 Altova GmbH

Datei	Aufgabe
	definierten. Eine Ausnahme zu dieser Regel bilden einige Grenzfälle, in denen der Standard von externen Codewerten abhängig ist und die Liste der möglichen Werte nicht zur Gänze umsetzt (siehe auch Validierung von EDI-Komponenten[471]). Sie können .codelist-Dateien modifizieren, wenn Sie die Standardvalidierungsregeln ändern müssen (z.B. um eine weniger strenge Validierung durchzuführen oder um neue, in der EDI-Codeliste noch nicht definierte Codes hinzuzufügen). Nähere Informationen dazu finden Sie unter Gelockerte Validierungsregeln für Felder[574] und Anpassen von Codewerten aus externen Quellen[577].
ParserErrors.Config	Diese Datei wird nur für den X12- und den HIPAA X12-Standard benötigt. Darin ist die Struktur der Datenelemente "ParserErrors_Message" und "ParserErrors_Group" in der EDI-Komponente in MapForce definiert. Diese Datenelemente enthalten Informationen über bei der Verarbeitung der Datei aufgetretene Parsing-Fehler und können zur Erzeugung von X12 997- (Functional Acknowledgement) und X12 999- (Implementation Acknowledgement)-Nachrichten verwendet werden.

Alle der oben angeführten Dateien sind XML-Dateien und können in einem XML-Editor wie XMLSpy bearbeitet werden. Sie beziehen sich alle, mit Ausnahme der Datei **EDI.Collection**, auf ein XML-Schema (siehe EDI-Konfigurationsschemas[543]). Für die Datei **EDI.Collection** ist kein explizites Schema definiert, sondern sie wird intern von MapForce validiert.

4.3.18.3 EDI-Konfigurationsschemas

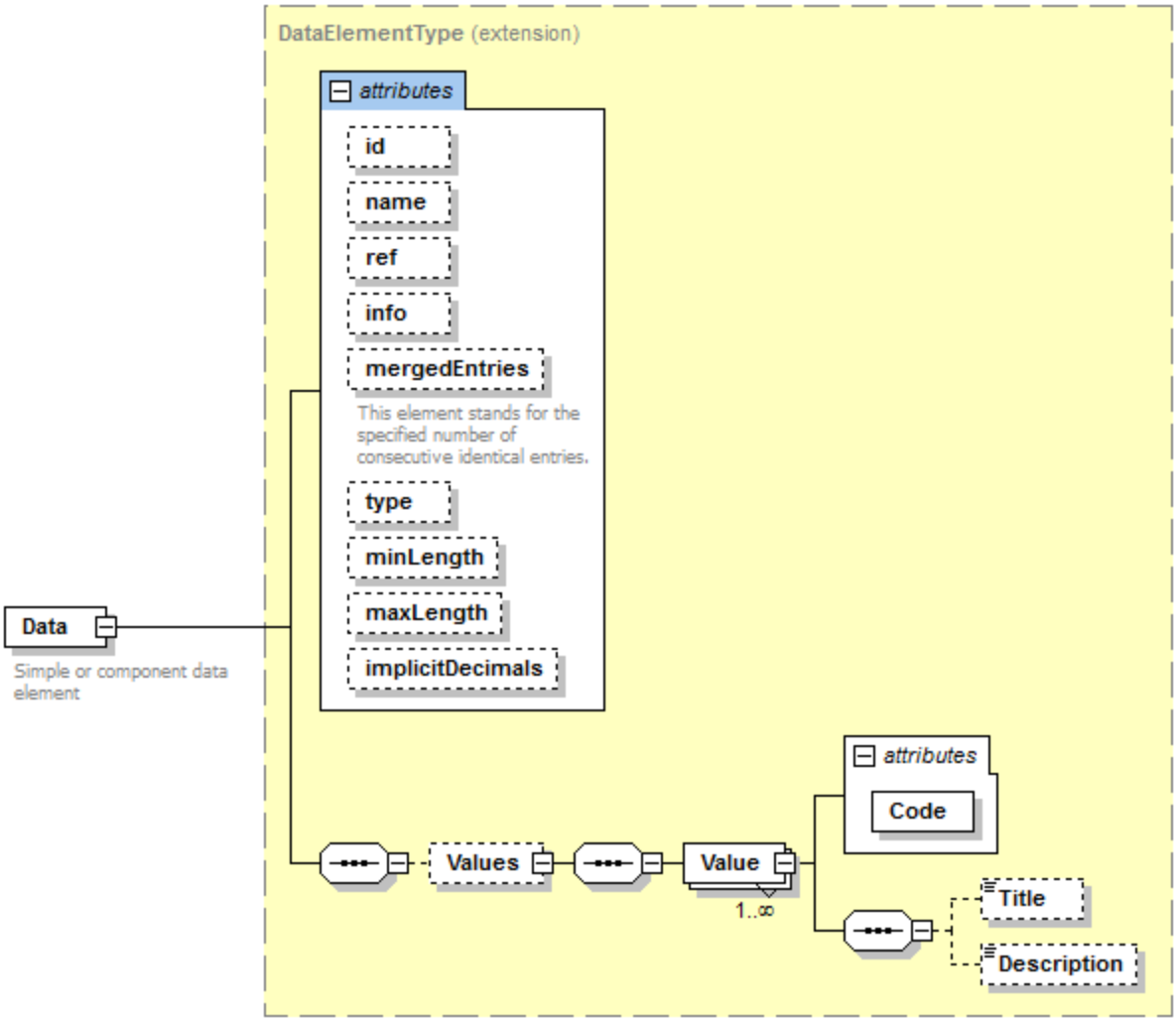
Wenn Sie eine Änderung an einer Konfigurationsdatei vornehmen, muss diese unbedingt anhand des dazugehörigen XML-Schemas validiert werden. Das Schema befindet sich im Verzeichnis **C:\Programme\Altova\MapForce2023\MapForceEDI** und kann eines der folgenden sein:

Datei	Aufgabe
EDIConfig.xsd (Version "3")	Dieses Konfigurationsschema ist ab MapForce 2011, Release 2 gültig. Wenn Sie Konfigurationsdateien haben, für die ein älteres Schema als Version 3 verwendet wird, müssen diese zu diesem Schema (oder einem neueren) aktualisiert werden. Nähere Informationen dazu finden Sie unter Aktualisieren älterer Konfigurationsdateien[545].
EDIConfig4.xsd	Dieses Konfigurationsschema ist ab MapForce 2011, Release 3 gültig. Es enthält mehrere Verbesserungen am ursprünglichen **EDIConfig.xsd-**Schema, darunter die Möglichkeit benutzerdefinierte rekursive Hierarchien zu definieren, die eventuell für das Mappen von Daten von oder auf X12- oder HL7-Formate benötigt werden.

Datei	Aufgabe
EDIConfig5.xsd	Dieses Konfigurationsschema ist ab MapForce 2017 gültig. Es kann darin nun ein Datenelement definiert werden, das unvollständige Validierungsregeln aufweist. Mit diesem Schema können komplexe Feldvalidierungsregeln auf Zeichenebene definiert werden (siehe Konfigurieren der Validierung auf Zeichenebene[581]).
EDIConfig6.xsd	Dieses Konfigurationsschema ist ab MapForce 2018 gültig. Es bietet die Möglichkeit, Daten von und auf das NCPDP SCRIPT-Format zu mappen, siehe NCPDP SCRIPT[498] .
EDIConfig7.xsd	Dieses Konfigurationsschema ist ab MapForce 2021 gültig. Es bietet die Möglichkeit, Daten von und auf SWIFT MT[504] zu mappen.

Das EDI-Konfigurationsschema enthält auch Informationen über die Struktur von Elementen, die in den Konfigurationsdateien vorkommen dürfen. Unten sehen Sie z.B. eine Darstellung einer EDI-Datenelementstruktur, wie aus dem Schema **EDIConfig4.xsd**:

© 2016-2022 Altova GmbH



Struktur des Datenelements (Schemaansicht von XMLSpy)

4.3.18.4 Aktualisieren älterer Konfigurationsdateien

EDI-Konfigurationsdateien enthalten eine Referenz auf eines der XML-Schemas im **MapForceEDI**-Verzeichnis und werden anhand dieses Schemas validiert (siehe EDI-Konfigurationsschemas 543). Um das Schema und die Version einer Konfigurationsdatei sehen zu können, öffnen Sie die Datei in einem Texteditor und überprüfen Sie die folgende Zeile am Anfang der Datei:

```
<?xml version="1.0" encoding="UTF-8"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../EDIConfig5.xsd" Version="5">
```

MapForce 2023 erkennt benutzerdefinierte EDI-Konfigurationen, die auf einer **EDIConfig.xsd** Datei ab Schemaversion 3 basieren und in MapForce 2011, Release 3 eingeführt wurden. Im Vergleich zu früheren

Versionen bietet dieses Schema nun die Möglichkeit, Daten von oder auf mehrere Nachrichtentypen in derselben EDI-Komponente zu mappen. Wenn Sie benutzerdefinierte EDI-Konfigurationsdateien verwendet haben, die auf eine ältere Schemaversion als Version 3 verweisen, müssen Ihre Konfigurationsdateien aktualisiert werden, da Sie im Dialogfeld "EDI-Collections durchsuchen" sonst die folgende Fehlermeldung erhalten, wenn Sie versuchen, die benutzerdefinierte EDI-Collection zum Mapping hinzuzufügen:

> "Bei der ausgewählten EDI-Collection wird eine alte Konfigurationsdateiversion verwendet, die nur einen einzigen Nachrichtentyp pro Komponente unterstützt."

In der folgenden Anleitung wird beschrieben, wie Sie das Schema einer EDI-Konfigurationsdatei von **EDIConfig.xsd** Version 2 auf **EDIConfig.xsd** Version 3 aktualisieren.

So aktualisieren Sie die Konfigurationsdateien:

1. Kopieren Sie **Envelope.Config** aus dem Originalkonfigurationsordner (z.B. EDIFACT) in den Ordner, der Ihre benutzerdefinierten EDI-Collections enthält (z.B. "EDIFACT.Custom").
2. Bearbeiten Sie **EDI.Collection**, ändern Sie das Version-Attribut des Root-Elements von "2" in "3".
3. Fügen Sie den Tag "**<Root File="Envelope.Config"/>**" hinter dem </Meta>-Tag hinzu:

```
<?xml version="1.0" encoding="UTF-8"?>
<Messages xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Version="3">
  <Meta>
    <Version>D</Version>
    <Release>04B</Release>
    <Agency>UN</Agency>
    </Meta>
    <Root File="Envelope.Config"/>
    <Message Type="ORDERS" File="ORDERS.Config" Description="Purchase order
message"/>
</Messages>
```

4. Bearbeiten Sie **ORDERS.Config**, ändern Sie das Version-Attribut des Root-Elements von "2" in "3".
5. Fügen Sie "**<Format standard="EDIFACT"/>**" (oder, je nach Fall X12 oder HL7) hinter dem </Meta>-Tag hinzu.
6. Benennen Sie "**<Group name="Message"...>**" in "**<Group name="Message_ORDERS"...>**" um (oder einen beliebigen benutzerdefinierten Nachrichtentyp) und entfernen Sie die äußeren Gruppenebenen ("Envelope" und "Interchange" und deren Segmente), wie unten gezeigt:

© 2016-2022 Altova GmbH

```
<?xml version="1.0" encoding="UTF-8"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Version="3">
   <Meta>
      <Version>D</Version>
      <Release>04B</Release>
      <Agency>UN</Agency>
   </Meta>
   <Format standard="EDIFACT"/>
   <Include href="Admin.Segment"/>
   <Include href="EDSD.Segment"/>
   <Include href="UNCL.Codelist"/>
   <Message>
      <MessageType>ORDERS</MessageType>
      <Description>Purchase order message</Description>
      <Revision>14</Revision>
      <Date>2004-11-23</Date>

      <!-- DELETED TEXT -->

               <Group name="Message_ORDERS" maxOccurs="unbounded" info="UNH -
Message header">
                  <Segment ref="UNH"/>
                  <Segment ref="BGM"/>
                        <Segment ref="UNT"/>
               </Group>

     <!-- DELETED TEXT -->

   </Message>
</Config>
```

Wenn während der Bearbeitung der Konfigurationsdatei ein Mapping geöffnet war, sollte es neu geladen werden. Die Verbindungen werden automatisch vom Datenelement "Message" auf das Datenelement "Message_ORDERS" neu gemappt.

4.3.18.5 Hinzufügen oder Entfernen von Nachrichtentypen

Um benutzerdefinierte EDI-Nachrichtentypen (oder, in der X12-Terminologie, Transaktionstypen) hinzuzufügen, gehen Sie folgendermaßen vor:

1. Erstellen Sie in dem Verzeichnis, in dem Ihre EDI-Collection gespeichert ist, eine **<Message>.Config**-Datei für jeden Nachrichtentyp (bzw. Transaktionstyp), der zur Collection hinzugefügt werden soll.
2. Modifizieren Sie die Datei **EDI.Collection,** um eine Referenz zu jedem verfügbaren Nachrichtentyp (Transaktionstyp) zu erstellen.

So entfernen Sie vorhandene Nachrichtentypen:

- Kommentieren Sie die entsprechenden `Message`-Elemente in der Datei **EDI.Collection** aus.

Hinzufügen und Konfigurieren der <Message>.Config-Datei

Jede **<Message>.config**-Datei, die Sie erstellen, muss gemäß ihrem Konfigurationsschema (siehe EDI-Konfigurationsdateien[542]) gültig sein. Um Probleme zu vermeiden, achten Sie v.a. auf die folgenden Elemente im Body der einzelnen Nachrichtenkonfigurationsdateien:

- Meta
- Format
- Include
- Message

Das `Meta`-Element enthält die Spezifikationsversion, die Release und die Agentur, z.B.:

```
<Meta>
   <Version>D</Version>
   <Release>14B</Release>
   <Agency>UN</Agency>
</Meta>
```

Das `Format`-Element enthält den Namen des Basisstandards, z.B.:

```
<Format standard="EDIFACT"/>
```

Die folgenden Werte sind gültig:

- Fixed
- EDIFACT
- X12
- HL7
- TRADACOMS (für diesen Wert wird ein Schema ab **EDIConfig4.xsd** benötigt, siehe EDI-Konfigurationsdateien[542])

Im `Include`-Element in einer **<Message>.Config**-Datei sind Referenzen auf etwaige zusätzliche Konfigurationsdateien enthalten. So enthält etwa das unten gezeigte Codefragment eine Referenz auf zwei .segment-Dateien und eine .codelist-Datei. In einer .segment-Datei sind alle Segment-, Composite- und Datenelementdefinitionen auf einer globalen Ebene gespeichert (damit diese Informationen nicht in den einzelnen Nachrichten, in denen sie vorkommen können, wiederholt werden müssen). In einer .codelist-Datei sind Validierungsregeln für die einzelnen Felder auf einer globalen Ebene gespeichert.

© 2016-2022 Altova GmbH

```
<Include href="Admin.Segment"/>
<Include href="EDSD.Segment"/>
<Include href="UNCL.Codelist"/>
```

Alle vom `Include`-Element referenzierten Dateien müssen sich im selben Verzeichnis wie die **<Message>.Config**-Datei befinden.

Im `Message`-Element jeder **<Message>.Config**-Datei sind die Struktur der Nachricht (d.h. die Hierarchie von Gruppen, Segmenten und Datenelementen) sowie etwaige für die Validierung verwendete Codewerte (falls nötig) definiert. Nähere Informationen zur Struktur einer Nachricht finden Sie unter Ändern der Nachrichtenstruktur 549. Beispiele, wie man vorhandene Nachrichten ändert, finden Sie in den folgenden Kapiteln:

- Beispiel: Anpassen einer EDIFACT Message 559
- Beispiel: Anpassen einer ANSI X12-Transaktion 566
- Definieren von Codewerten auf Message-Ebene (inline) 580

Ändern der EDI.Collection-Datei

Wenn Sie eine benutzerdefinierte **<Message>.Config**-Datei hinzufügen, müssen Sie auch ein `Message`-Element zur **EDI.Collection**-Datei hinzufügen. Jedes `Message`-Element muss die folgenden Attribute haben:

Type	Definiert den offiziellen Code des Nachrichtentyps (z.B. "BALANC", "999").
File	Definiert den Namen der Datei, die die Spezifikation des jeweiligen Nachrichtentyps enthält. Die Datei selbst muss sich im selben Verzeichnis befinden.
Description	Enthält eine Beschreibung dieses Nachrichtentyps, wie sie auf der grafischen Benutzeroberfläche von MapForce angezeigt wird.

Beispiele für gültige **EDI.Collection**-Dateien finden Sie in den vordefinierten MapForce EDI-Collections im Verzeichnis **C:\Programme\Altova\MapForce2023\MapForceEDI**.

4.3.18.6 Ändern der Nachrichtenstruktur

In jeder **<Message>.config**-Datei wird die Struktur eines `Message`-Elements durch das XML-Schema des EDI-Standards, den Sie anpassen, vorgegeben. Dieses Schema hat den Namen **EDIConfig.xsd** (oder einen ähnlichen Namen, siehe EDI-Konfigurationsschemas 543). Um die Struktur der Nachricht zu erstellen oder zu ändern, müssen Sie daher sowohl die Dokumentation zu Ihrem benutzerdefinierten EDI-Standard als auch das XML-Schema konsultieren.

Hinzufügen von Segmenten, Datenelementen und Composites

Um Datenelemente, Segmente, Composites und Subcomposites hinzuzufügen, erstellen Sie die Elemente `<Data>`, `<Segment>`, `<Composite>` und `<SubComposite>`. Gehen Sie dabei folgendermaßen vor:

- Die .segment-Datei muss die Liste aller wiederverwendbaren, global definierten Segmente und Datenelemente enthalten.

- Die .config-Datei für den jeweiligen Nachrichtentyp (oder Transaktionstyp) muss die eigentliche Struktur dieser Nachricht enthalten (die Hierarchie von Segmenten, Composites und Datenelementen).

Auf diese Art können Sie ein Datenelement einmal (in der .segment-Datei) definieren und es in mehreren Nachrichten (in .config-Dateien) wiederverwenden.

Um ein Datenelement zu referenzieren, verwenden Sie das Attribut `ref`. So sehen Sie etwa im Codefragment unten, wie das Datenelement F100 in der **X12.Segment**-Datei der HIPAA.X12 Collection definiert ist.

```
<Data name="F100" type="string" maxLength="3" minLength="3" info="Currency Code" />
```

X12.Segment-Datei

Nun können alle Segmente in allen Transaktionstypen, in denen dieses Datenelement benötigt wird, dieses referenzieren. Im Codefragment unten sehen Sie, wie das "CUR"-Segment der Transaktion "Health Care: Professional" (837P) das zuvor definierte F100-Datenelement referenziert:

```
<Include href="X12.Segment" />

...

<Segment name="CUR" minOccurs="0" info="Foreign Currency Information">
    <Data ref="F100" info="Currency Code" />
</Segment>
```

837P.Config-Datei (X12)

Eine Referenz kann sich in derselben Datei oder in einer externen Datei befinden. Die .config-Datei im oben gezeigten Codefragment enthält eine Referenz auf eine externe .segment-Datei. Aus diesem Grund finden Sie am Anfang der **837P.Config**-Datei eine **`<Include href="X12.Segment" />`** -Anweisung, um die Referenz zur externen **X12.Segment**-Datei hinzuzufügen.

Ein weiteres Beispiel für das Hinzufügen eines benutzerdefinierten Segments finden Sie unter Beispiel: Hinzufügen des TA1-Segments 558.

Erstellen von Gruppen und Schleifen

Bei der Erstellung oder Änderung von EDI-Strukturen müssen Segmente oder Datenelemente oft unter einem übergeordneten Node (oder einer übergeordneten Gruppe) gruppiert werden. Selbst die Nachricht in der EDI-Komponente ist eine Gruppe, da sie all die Segmente, aus denen diese Nachricht besteht, in einer Gruppe vereint. Gruppen können auch ineinander verschachtelt sein (sie können untergeordnete Gruppen enthalten). Außerdem können sie wiederholt vorkommen (in diesem Fall werden sie zu Schleifen). Gruppen und Schleifen können durch Hinzufügen eines `<Group>`-Elements erstellt werden. Mit Hilfe des Attributs `maxOccurs` kann die Anzahl der zulässigen Instanzen einer Schleife festgelegt werden. Im folgenden Codefragment sehen Sie eine Schleife, die bis zu 25 Mal vorkommen kann, optional ist und aus zwei Segmenten besteht:

```
<Group name="LoopSAC" maxOccurs="25" minOccurs="0">
   <Segment ref="SAC"/>
   <Segment ref="CUR" minOccurs="0"/>
</Group>
```

 © 2016-2022 Altova GmbH

850.Config-Datei (X12)

Definieren des Typs und der Größe von Datenelementen

Mit Hilfe des Attributs `type` eines Datenelements oder Segments können Sie seinen Typ (String, Dezimal, usw.) definieren. Bei den möglichen Datentypen (Attribut `type`) handelt es sich im Allgemeinen um beliebige XML-Schematypen, die in den standardmäßig bereitgestellten Konfigurationsdateien verwendet werden. Eventuell können auch andere XML-Schema simpleTypes verwendet werden, deren Unterstützung aber nicht garantiert ist. Die folgenden simpleTypes werden nicht unterstützt: "anyType", "ENTITIES" und "QName".

Mit Hilfe der Attribute `minLength` und `maxLength` eines Datenelements kann dessen minimale und maximale Länge definiert werden.

Definieren obligatorischer und optionaler Datenelemente

Um festzulegen, dass ein Datenelement obligatorisch ist, definieren Sie keinen Wert für das Attribut `minOccurs` (Dieses Attribut hat als Standardwert "1", was bedeutet, dass das Feld standardmäßig obligatorisch ist). Um ein Datenelement als optional zu definieren, definieren Sie als Wert des Attributs `minOccurs` den Wert "0". Im Codefragment unten sehen Sie z.B. zwei obligatorische Datenelemente (F1331, F1332) und ein optionales Datenelement (F1325).

```
<Composite name="C023" info="Health Care Service Location Information">
   <Data ref="F1331"/>
   <Data ref="F1332"/>
   <Data ref="F1325" minOccurs="0"/>
</Composite>
```

4.3.18.7 Behandlung mehrerer aufeinander folgender Datenelemente

In EDI-Segmenten können oft mehrere aufeinander folgende Elemente mit demselben Namen vorkommen. Dies gilt z.B. für das N2-Segment des "850 - PURCHASE ORDER" Transaction Set des X12-Standards. Im N2-Segment dürfen der Spezifikation gemäß zwei aufeinander folgende alphanumerische "Name"-Felder vorkommen (eines davon ist obligatorisch, eines optional):

REF	ELE ID	NAME	RPT	ATTRIBUTES
01	93	Name	M	AN 1/60
02	93	Name	O	AN 1/60

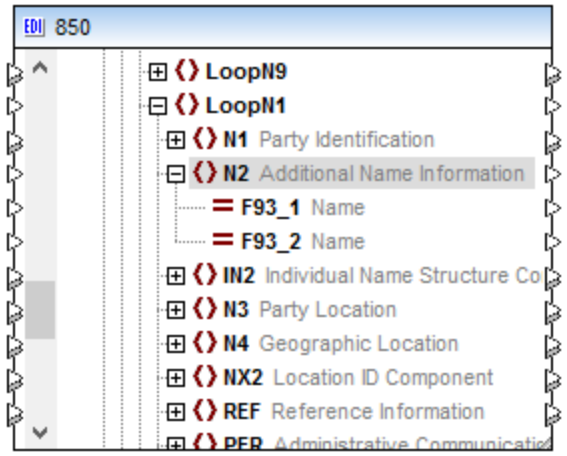
Wenn Sie eine EDI-Konfiguration erstellen oder ändern, haben Sie die zwei unten beschriebenen Möglichkeiten, wie Sie mit solchen Datenelementen verfahren können.

Methode 1: Erstellen separater Mapping-Nodes für jedes aufeinander folgende Datenelement

Bei dieser Methode erstellen Sie in der Segment-Datei mehrere Instanzen des Datenelements. In diesem Fall wird für jede Instanz des Datenelements im Mapping ein neuer Node angezeigt. Werfen Sie einen Blick auf diese Konfiguration:

```
<Segment name="N2" info="Additional Name Information">
   <Data ref="F93" nodeName="F93_1"/>
   <Data ref="F93" minOccurs="0" nodeName="F93_2"/>
</Segment>
```

Mit dieser Konfiguration werden im Mapping zwei separate F93 Nodes erstellt, einer für jede Instanz von F93:



X12 EDI-Komponente mit separaten F93-Nodes

Diese Methode ist sinnvoll, wenn jede Instanz des Datenelements auch alleinstehend Sinn ergibt, sodass Sie jedes im Mapping in einem separaten Node anzeigen möchten.

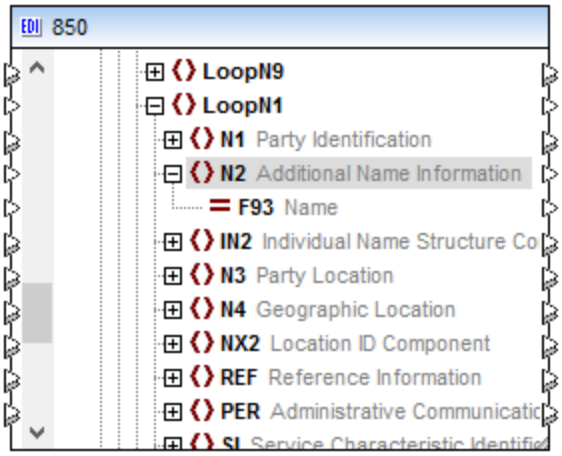
Methode 2: Erstellen eines einzigen Node für alle Instanzen des Datenelements

Bei dieser Methode behandeln Sie alle Instanzen des Datenelements als einen einzigen Node. Fügen Sie dazu das Attribut `mergedEntries` zum entsprechenden Datenelement hinzu. Im unten gezeigten Codefragment hat das Attribut `mergedEntries` des Datenelements F93 den Wert "2". Dadurch sind zwei aufeinander folgende Instanzen dieses Felds in der EDI-Instanzdatei zulässig.

```
<Segment name="N2" info="Additional Name Information">
   <Data ref="F93" mergedEntries="2"/>
</Segment>
```

Der Node für das F93-Datenelement wird im Mapping als ein einziger Node angezeigt:

© 2016-2022 Altova GmbH



X12 EDI-Komponente mit separaten F93-Nodes

Beim Mappen von Daten aus einem Datenelement mit zusammengeführten Einträgen werden in der Zielkomponente mehrere duplizierte Elemente erstellt, wenn die EDI-Quelldatei mehrere (nicht leere) Instanzen dieses Elements enthält. Angenommen, Sie haben eine EDI-Quelldatei, in der das N2-Segment zwei aufeinander folgende Instanzen von F93 enthält. Die erste Instanz ist "Michelle Butler" und die zweite ist "Mrs".

```
N2+Michelle Butler+Mrs+'
```

Wenn Sie Daten aus einer solchen EDI-Datei mappen, würde in der Zielkomponente die folgende Ausgabe erzeugt (beachten Sie, dass `<ContactName>` zwei Mal vorkommt):

```
<Customer>
      <Number>123</Number>
      <ContactName>Michelle Butler</ContactName>
      <ContactName>Mrs</ContactName>
      <CompanyName>Nanonull, Inc.</CompanyName>
</Customer>
```

Die Verwendung dieser Methode ist dann sinnvoll, wenn die einzelnen Instanzen des Datenelements für sich selbst keinen Sinn ergeben und Sie im Mapping keinen separaten Node für jede einzelne Instanz erzeugen möchten. So können etwa in einem Datenelement mit mehreren Instanzen mehrere Zeileneinträge, aus denen eine Adresse besteht, gespeichert sein. In solchen Fällen eignet sich die Methode der "zusammengeführten Einträge" eventuell besser als die Verwendung einzelner Nodes für jede Adresszeile.

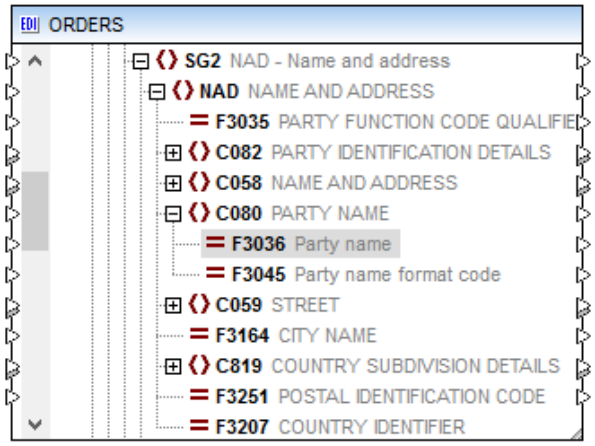
In einigen standardmäßig mit MapForce bereitgestellten Konfigurationsdateien werden zusammengeführte Einträge verwendet, wodurch im Mapping für mehrere aufeinander folgende Instanzen eines Datenelements nur ein Node angezeigt wird. Wenn Sie separate Nodes für die einzelnen Instanzen des Datenelements benötigen, bearbeiten Sie dazu die EDI-Konfigurationsdatei. Ein Beispiel dafür finden Sie unter Aufgliedern zusammengeführter Einträge in separate Nodes 554.

4.3.18.8 Aufgliedern zusammengeführter Einträge in separate Nodes

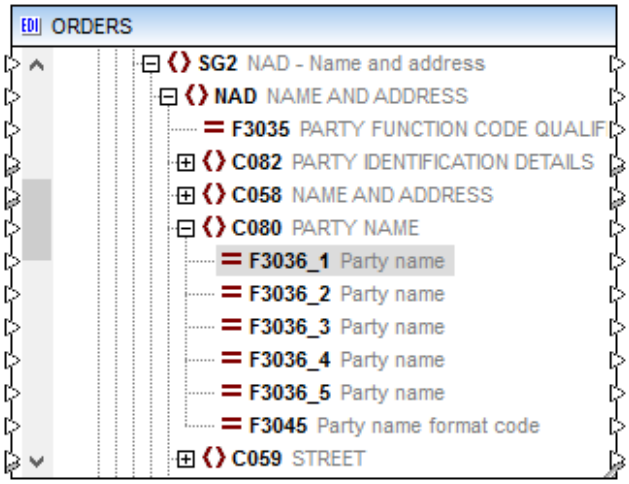
Einige Datenelemente in den standardmäßig mit MapForce bereitgestellten EDI-Konfigurationsdateien sind so konfiguriert, dass nur ein Node im Mapping angezeigt wird, selbst wenn das entsprechende Datenelement mehrmals in der EDI-Quelldatei vorkommt (siehe auch Behandlung mehrerer aufeinander folgender Datenelemente 551). Ein Beispiel für ein solches Datenelement ist F3036 des UN/EDIFACT-Standards. Der Spezifikation gemäß kann F3036 im C080 ("Party name") Composite bis zu fünf Mal hintereinander vorkommen (siehe https://www.unece.org/fileadmin/DAM/trade/untdid/d15b/trcd/trcdc080.htm).

In der UN/EDIFACT-Standard-Collection von MapForce wird das Datenelement F3036 durch einen einzigen Node im Mapping dargestellt. Wenn Sie von diesem Node eine Mapping-Verbindung zu einem Ziel-Node ziehen, so würden mit dieser Verbindung in der Zielkomponente so viele Datenelemente erstellt, wie Instanzen von F3036 in der Quellkomponente vorkommen. (Dies entspricht der Grundregel von MapForce-Verbindungen, dass für jedes Datenelement in der Quellkomponente ein Datenelement in der Zielkomponente erstellt wird). Es gibt jedoch Fälle, in denen dies nicht das gewünschte Ergebnis ist, sondern in denen Sie für jede Instanz des Datenelements separate Nodes benötigen.

Ziel dieses Beispiels ist es daher, eine benutzerdefinierte UN/EDIFACT-Konfiguration zu erstellen, gemäß welcher für jede Instanz von F3036 separate Mapping-Nodes angezeigt werden. Auf diese Art können Sie jede einzelne Instanz von F3036 auf einen separate Ziel-Node mappen. In den Abbildungen unten sehen Sie die EDIFACT Mapping-Komponente vor und nach der Anpassung.



Vor der Anpassung



Nach der Anpassung

Um das F3036-Datenelement in mehrere Mapping-Nodes aufzugliedern, gehen Sie folgendermaßen vor:

1. Erstellen Sie anhand der vorhandenen UN/EDIFACT-Collection eine benutzerdefinierte EDI-Collection (siehe Erstellen einer benutzerdefinierten EDI-Collection 540). Dadurch stellen Sie sicher, dass die Änderungen sich in keiner Weise auf die EDI-Standardkonfiguration auswirken.
2. Öffnen Sie die Datei **EDSD.Segment** Ihrer benutzerdefinierten Collection in einem XML-Editor. (Wenn Sie eine andere EDI-Variante anpassen möchten, bearbeiten Sie die entsprechende .segment-Datei dieser EDI-Variante. Bei X12 wäre dies die Datei **X12.Segment**.)
3. Suchen Sie nach dem Composite C080 und entfernen Sie das Attribut `mergedEntries="5"` des Datenelements F3036.

© 2016-2022 Altova GmbH

```
<Composite name="C080" info="PARTY NAME">
   <Data ref="F3036" mergedEntries="5"/>
   <Data ref="F3045" minOccurs="0"/>
</Composite>
```

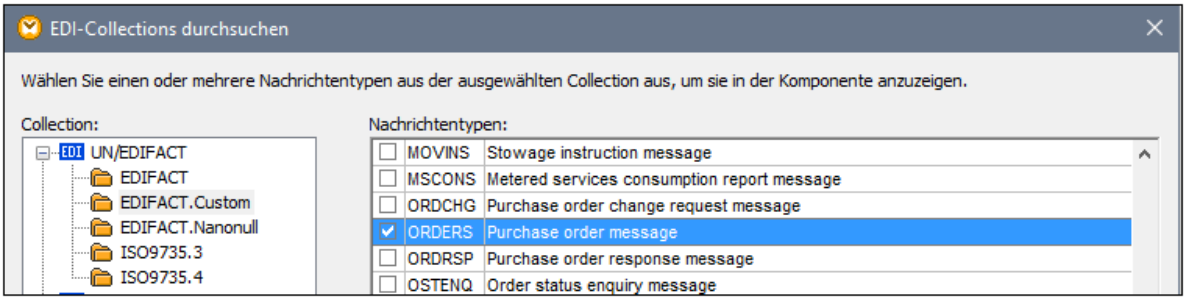
4. Geben Sie für jedes der Felder einen eindeutigen Node-Namen ein (z.B. F3036_1, F3036_2, usw.). Die Konfiguration sollte zu diesem Zeitpunkt folgendermaßen aussehen:

```
<Composite name="C080" info="PARTY NAME">
   <Data ref="F3036" nodeName="F3036_1"/>
   <Data ref="F3036" nodeName="F3036_2" minOccurs="0"/>
   <Data ref="F3036" nodeName="F3036_3" minOccurs="0"/>
   <Data ref="F3036" nodeName="F3036_4" minOccurs="0"/>
   <Data ref="F3036" nodeName="F3036_5" minOccurs="0"/>
   <Data ref="F3045" minOccurs="0"/>
</Composite>
```

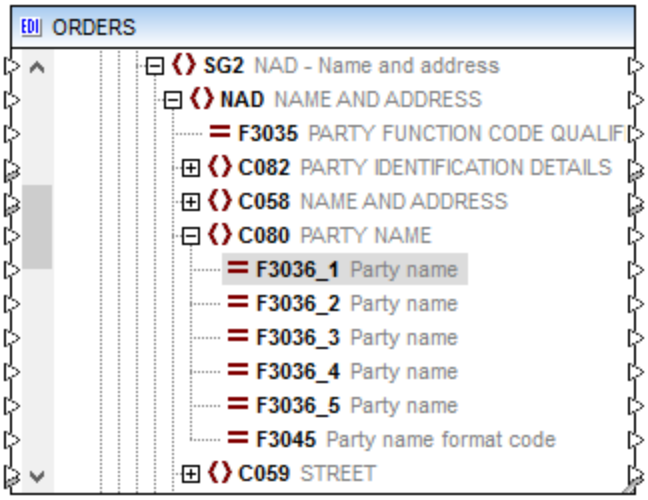
5. Fügen Sie zu allen Feldern mit Ausnahme des ersten das Attribut `minOccurs="0"` hinzu. Der Grund dafür ist, dass gemäß dem UN/EDIFACT-Standard nur die erste Instanz des Felds obligatorisch ist; die anderen hängen von Bedingungen ab.
6. Speichern Sie die bearbeitete **EDSD.Segment**-Datei.

Sie können die geänderte EDI-Collection nun folgendermaßen in MapForce anzeigen und damit arbeiten:

1. Klicken Sie im Menü **Einfügen** auf **EDI**.
2. Sobald Sie dazu aufgefordert werden, wählen Sie die zuvor erstellte benutzerdefinierte Collection aus (in diesem Beispiel "EDIFACT.Custom").
3. Wählen Sie einen beliebigen Nachrichtentyp, in dem das C080-Segment verwendet wird, aus (in diesem Beispiel "ORDERS - Purchase Order Message").



Die EDI-Komponente wird nun geändert in MapForce angezeigt (jede Instanz des F3036-Datenelements hat im Mapping einen separaten Node).



Mit dem in diesem Beispiel beschriebenen Verfahren können Sie, unabhängig von der EDI-Variante, auch zusammengeführte Einträge für andere Datenelemente aufgliedern.

4.3.18.9 Ändern der möglichen Codewerte von Datenelementen

Welche Werte ein Datenelement haben kann, kann mit Hilfe von Konfigurationsdateien geändert werden. Ähnlich anderen Konfigurationsinformationen können Sie die möglichen Codewerte für jede einzelne Nachricht (lokale Anpassung) oder global (für alle Nachrichten, in denen diese bestimmte Datenelement verwendet werden kann) konfigurieren.

> Wie bei jeder anderen benutzerdefinierten Konfiguration wird empfohlen, zur Anpassung von Codewerten eine benutzerdefinierte EDI-Collection zu erstellen (siehe Erstellen einer benutzerdefinierten EDI-Collection 540) und darin alle Anpassungen vorzunehmen, anstatt die MapForce-Standardkonfiguration zu ändern.

Welche Anpassungsoptionen zur Verfügung stehen, hängt von der für Ihre benutzerdefinierte EDI-Collection verwendeten Schemaversion ab (siehe EDI-Konfigurationsschemas 543). Einfach ausgedrückt, bedeutet eine neuere Schemaversion, dass mehr Konfigurationsoptionen zur Verfügung stehen. Die Schemas unterstützen jeweils die folgenden benutzerdefinierten Codewerte:

- Bei **EDIConfig.xsd** kann eine vorhandene .codelist-Datei geändert werden, um benutzerdefinierte Codewerte auf globaler Ebene hinzuzufügen.
- Bei **EDIConfig4.xsd** können benutzerdefinierte Codewerte nicht nur auf globaler Ebene, sondern auch auf Nachrichtenebene hinzugefügt werden.
- Bei **EDIConfig5.xsd** stehen alle oben beschriebenen Optionen zur Verfügung sowie zusätzlich die Möglichkeit, komplexe Konfigurationsregeln auf Zeichenebene hinzuzufügen (siehe Konfigurieren der Validierung auf Zeichenebene 581). Außerdem können Datenelemente als Datenelemente mit unvollständiger Validierung definiert werden (siehe Gelockerte Validierungsregeln für Felder 574).

© 2016-2022 Altova GmbH

Globale Anpassung

Um benutzerdefinierte Codewerte für ein Datenelement global hinzuzufügen, bearbeiten Sie die .codelist-Datei Ihrer benutzerdefinierten EDI-Collection. Suchen Sie zuerst nach dem Datenelement (Feld), dessen Werte Sie ändern müssen, und fügen Sie anschließend, wie unten gezeigt, alle erforderlichen benutzerdefinierten Werte als neue <Value> Elemente hinzu:

```
<?xml version="1.0" encoding="UTF-8"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Version="5"
xsi:noNamespaceSchemaLocation="../EDIConfig5.xsd">
   <Include href="ISO6346.Codelist"/>
   <Codelist>
      <Field Name="F1001" Length="3" Type="Alphanumeric">
         <Title>Document name code</Title>
         <Description>Code specifying the document name.</Description>
         <Values>
            <Value Code="1">
               <Title>Certificate of analysis</Title>
               <Description>Certificate providing the values of an
analysis.</Description>
            </Value>

>>> add your custom values here...

        </Values>
      </Field>
   </Codelist>
</Config>
```

UNCL.Codelist-Datei (UN/EDIFACT Collection)

Wie Sie im Codefragment oben sehen, hat jedes <Value>-Element ein Code-Attribut, in dem der eigentliche für dieses Datenelement erwartete Wertcode definiert ist. Die Elemente <Title> und <Description> liefern zusätzliche beschreibende Informationen zu diesem Wertcode.

Das Codefragment oben enthält auch eine Include-Anweisung, aufgrund welcher MapForce nach zusätzlichen Codewerten sucht, die in einer externen Datei (**ISO6346.Codelist**) definiert sind, die auf demselben Schema wie die Datei **UNCL.Codelist** basiert. Verwenden Sie diese Methode, wenn Sie verschiedene externe Dateien, die benutzerdefinierte Codewerte enthalten, anpassen müssen.

Ein Beispiel dazu finden Sie unter Anpassen von Codewerten aus externen Quellen 577.

Lokale Anpassung

Um benutzerdefinierte Codewerte auf Nachrichtenebene hinzuzufügen, müssen Sie die .config-Datei für den Nachrichtentyp bearbeiten. Nähere Informationen dazu finden Sie unter Definieren von Codewerten auf Message-Ebene (inline) 580.

4.3.18.10 Beispiel: Hinzufügen eines TA1-Segments

Das TA1-Segment ist ein optionales Segment, mit dem der Empfang des Interchange-Elements und die syntaktische Korrektheit der darin enthaltenen Envelope-Segmente bestätigt wird. Das Segment kann durch Änderung der Konfigurationsdateien zu EDI X12- und EDI HIPAA-Komponenten in MapForce hinzugefügt werden.

So inkludieren Sie das TA1-Segment in eine EDI-Komponente:

1. Erstellen Sie anhand der HIPAA.X12-Collection eine benutzerdefinierte EDI-Collection (siehe Erstellen einer benutzerdefinierten EDI-Collection 540). Damit stellen Sie sicher, dass Ihre Änderungen keine Auswirkung auf die Originalkonfiguration haben.
2. Öffnen Sie die Datei **Envelope.config** Ihrer benutzerdefinierten EDI-Collection in einem XML-Editor.
3. Fügen Sie den Text **** zwischen das ISA- und das Group-Segment ein und speichern Sie die Datei. Die Datei sollte nun folgendermaßen aussehen:

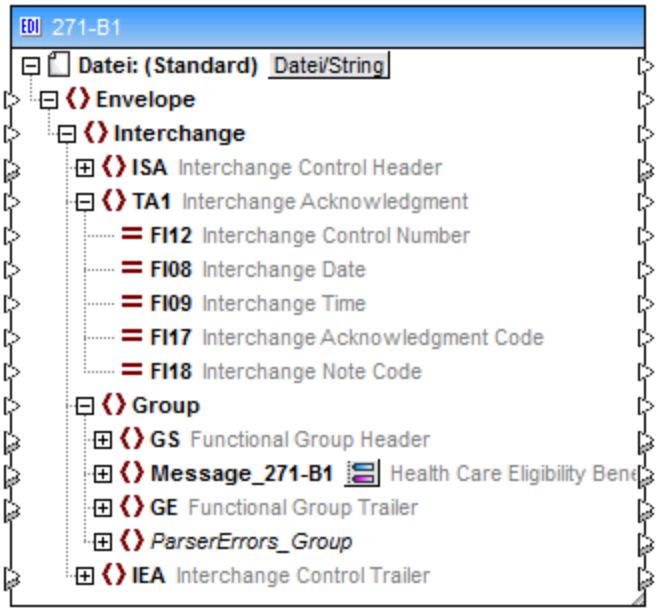
```
<?xml version="1.0" encoding="UTF-8"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../EDIConfig.xsd" Version="3">
   <Meta>
      <Release>6020</Release>
      <Agency>X12</Agency>
   </Meta>
   <Format standard="X12"/>
   <Include href="X12.Segment"/>
   <Include href="X12.Codelist"/>
   <Include collection="EDI.Collection"/>
   <Group name="Envelope">
      <Group name="Interchange" maxOccurs="unbounded">
         <Segment ref="ISA" minOccurs="0"/>
         <Segment ref="TA1" minOccurs="0" maxOccurs="unbounded"/>
         <Group name="Group" maxOccurs="unbounded">
            ...
         </Group>
         <Segment ref="IEA" minOccurs="0"/>
      </Group>
   </Group>
</Config>
```

Mit der oben markierten Zeile wird eine Referenz auf das bereits in der Datei **X12.Segment** definierte TA1-Segment hinzugefügt. Sie können die Struktur des TA1-Segments nun folgendermaßen in MapForce anzeigen:

1. Klicken Sie im Menü **Einfügen** auf **EDI**.
2. Wählen Sie im Dialogfeld "EDI-Collections durchsuchen" die oben erstellte benutzerdefinierte Collection und einen Nachrichtentyp (z.B. "850 - Purchase Order") aus.

Beachten Sie, dass die benutzerdefinierte Komponente nun das TA1-Segment enthält. Sie können nun Mapping-Verbindungen von und zu dieser benutzerdefinierten Komponente ziehen.

© 2016-2022 Altova GmbH



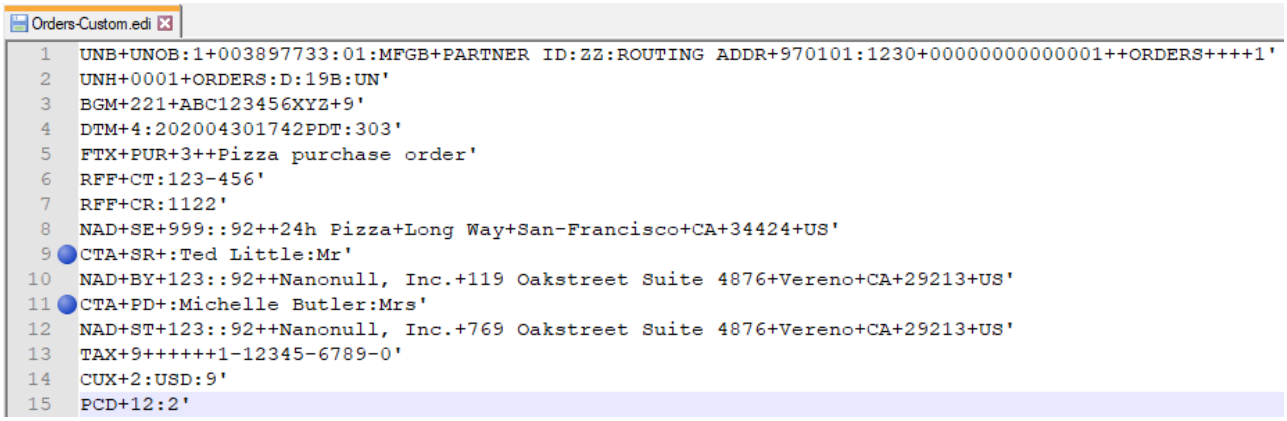
Benutzerdefinierte X12 EDI-Komponente mit optionalem TA1-Segment

4.3.18.11 Beispiel: Anpassen einer EDIFACT Message

In diesem Beispiel wird gezeigt, wie Sie EDIFACT-Nachrichten anpassen können, damit MapForce nicht standardkonforme oder geänderte EDIFACT-Formate verarbeiten kann. Beachten Sie, dass dieses Beispiel zur Anpassung spezifisch auf Version **19B** von EDIFACT basiert. Für andere EDIFACT-Versionen kann sich die Vorgangsweise etwas unterscheiden.

Sie finden ein Beispielmapping, in dem eine benutzerdefinierte EDIFACT-Datei mit einem etwas erweiterten CTA-(Contact Information)-Segment der ORDERS Message verarbeitet wird, unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\ExtractCustomEDIFACT.mfd**. Beachten Sie, dass MapForce dieses Mapping nicht erfolgreich validieren kann, falls Sie das Mapping öffnen, ohne vorher die Anleitungen zur benutzerdefinierten Anpassung befolgt zu haben. Der Grund hierfür ist, dass das Mapping eine benutzerdefinierte EDIFACT-Konfiguration referenziert, die auf Ihrem Rechner noch nicht zur Verfügung steht (Sie wird in diesem Beispiel erstellt).

Im Beispielmapping werden Daten aus der Datei **Orders-Custom.EDI** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples** gelesen. Wenn Sie diese Datei mit einem Text-Editor öffnen, sehen Sie, dass darin ein benutzerdefiniertes CTA-Segment verwendet wird: Zeile 9 enthält den Eintrag **Mr** und Zeile 11 enthält den Eintrag **Mrs**:



Bevor Sie Daten aus benutzerdefinierten EDI-Dateien wie der oben gezeigten mappen können, muss eine benutzerdefinierte EDIFACT-Konfiguration erstellt werden. In diesem Beispiel werden wir die EDIFACT-Standardkonfiguration anpassen, indem wir das CTA-Segment durch ein neues Feld für die Anrede der Person ergänzen. Sie finden das angepasste Endergebnis im Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples** als ZIP-Archiv (**EDIFACT.Nanonull.zip**).

Voraussetzungen

Um dieses Beispiel durchführen zu können, müssen Sie zuerst die ausführbare Datei "EDIFACT Configuration Files"vom Altova Download Center (https://www.altova.com/de/mapforce/download) herunterladen. Stellen Sie sicher, dass Sie das jeweilige Paket für Ihre Version und Plattform (32-Bit, 64-Bit) von MapForce auswählen,

Das Installationspaket enthält alle von MapForce unterstützten EDIFACT-Versionen einschließlich der in diesem Beispiel verwendeten Version "19B". Um die Installation auszuführen, doppelklicken Sie auf die heruntergeladene ausführbare Date und befolgen Sie die Anweisungen auf dem Bildschirm. Nach der Installation werden für die entsprechenden installierten EDIFACT-Konfigurationsversionen im Ordner **C:\Programme\Altova\MapForce2023\MapForceEDI** neue Verzeichnisser erstellt.

Wenn Sie eine 32-Bit-MapForce-Version auf einem 64-Bit-Betriebssystem verwenden, ändern Sie den obigen Verzeichnispfad in **C:\Programme(x86)**-

Erstellen der benutzerdefinierten EDI-Collection

1. Erstellen Sie ein neues Verzeichnis, in dem Ihre benutzerdefinierte EDI-Collection unter dem folgenden Pfad gespeichert wird: **C:\Benutzer\<Name>\Dokumente\Altova\MapForce2023\MapForceEDI\EDIFACT.Nanonull**. Der Name "EDIFACT-Nanonull" hilft Ihnen, diese benutzerdefinierte EDI-Collection von der mit MapForce installierten Standard-Collection zu unterscheiden.
2. Kopieren Sie die folgenden Dateien aus **C:\Programme\Altova\MapForce2023\MapForceEDI\EDIFACT.2019B** in **C:\Benutzer\<Name>\Dokumente\Altova\MapForce2023\MapForceEDI\EDIFACT.Nanonull**:

   - Admin.Segment
   - EDI.Collection
   - EDSD.Segment
   - Envelope.Config
   - ISO6346.Codelist

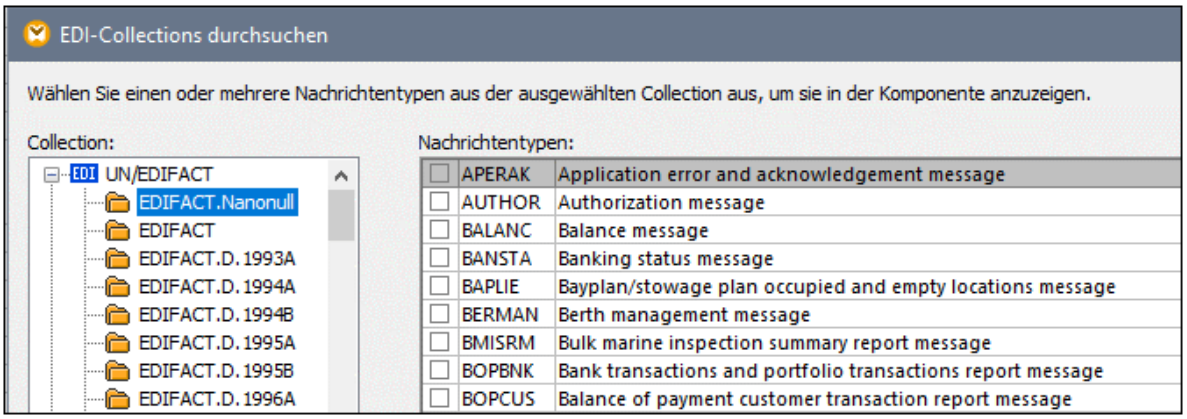
© 2016-2022 Altova GmbH

- ORDERS.Config
- UNCL.Codelist

Mit Ausnahme von **ORDERS.Config** werden alle diese Dateien von der Datei **Envelope.Config** referenziert und müssen daher in der EDI-Collection vorhanden sein. Wozu die einzelnen Dateien dienen, ist unter EDI-Konfigurationsdateien 542 beschrieben.

Die neue **EDI.Collection**-Datei sollte nun von MapForce aufgerufen werden können. Sie können dies folgendermaßen testen:

1. Starten Sie MapForce (oder starten Sie das Programm neu, wenn es bereits ausgeführt wird).
2. Klicken Sie im Menü **Einfügen** auf **EDI**. Alternativ dazu können Sie auch in der Symbolleiste auf die Schaltfläche **EDI einfügen** klicken. Daraufhin wird ein Dialogfeld geöffnet, das eine neue Collection mit dem Namen "EDIFACT.Nanonull" enthält.



Anmerkung: Sie können Ihre benutzerdefinierte EDI-Collection auch, wie unter Erstellen einer benutzerdefinierten EDI-Collection 540 beschrieben, in einem anderen Verzeichnis erstellen.

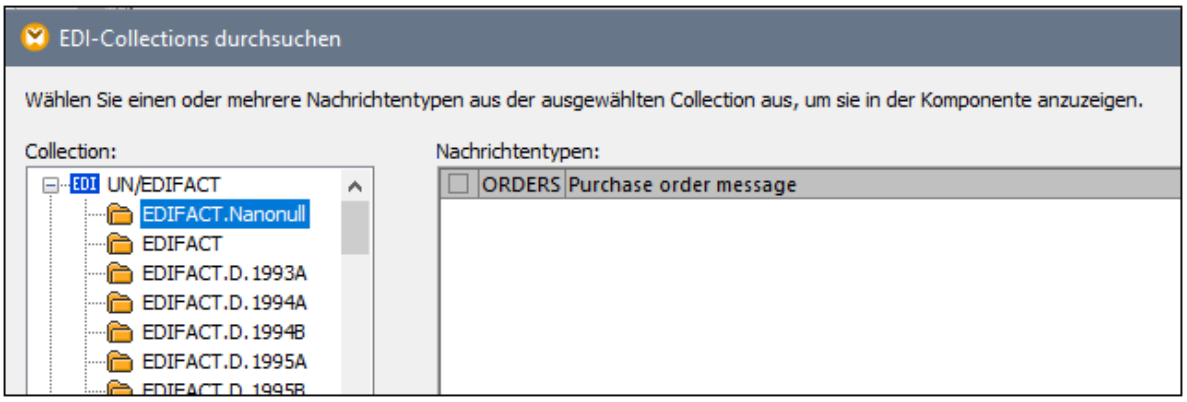
Konfigurieren der EDI.Collection-Datei

1. Öffnen Sie die Datei **EDI.Collection** aus Ihrer benutzerdefinierten EDI-Collection in einem XML-Editor (z.B. XMLSpy).
2. Entfernen Sie alle "Message"-Elemente mit Ausnahme der ORDERS Message. Die Datei sollte nun folgendermaßen aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<Messages Version="3">
   <Meta>
      <Version>D</Version>
      <Release>19B</Release>
      <Agency>UN</Agency>
   </Meta>
   <Root File="Envelope.Config"/>
   <Message Type="ORDERS" File="ORDERS.Config" Description="Purchase order
message"/>
</Messages>
```

3. Speichern Sie die Datei.

Wenn Sie die Menübefehl **Einfügen | EDI** wählen, wird in der Collection nun nur ein Nachrichtentyp angezeigt: "ORDERS".



Globale Anpassung im Vergleich zu Inline-Anpassung

Ziel dieses Beispiels ist es, das CTA (Contact Information) Segment neu zu definieren. Das CTA-Segment besteht aus einem einzigen Feld (F3139) und einem Composite (C056). Für den Inhalt der Anrede (z.B. "Mr" oder "Mrs") fügen wir ein neues Feld namens X1000 zum Composite C056 hinzu. Es gibt verschiedene Methoden, um die Anpassung vorzunehmen:

- **Global** durch Anpassen der Datei **EDSD.segment**. Alle Segmente in allen Messages, in denen das Composite C056 verwendet wird, enthalten daraufhin das neue Element bzw. eine Referenz darauf.
- **Inline** durch Anpassen der Datei **ORDERS.Config**. Nur das angepasste Segment (CTA) in der aktuellen Message enthält daraufhin das neue Element.

Globale Anpassung

Um den Zugriff auf das neue Feld X1000 *global* zu machen, müssen nur an der Datei **EDSD.Segment** Änderungen vorgenommen werden. Alle Segmente in allen Messages, in denen das Composite C056 verwendet wird, enthalten daraufhin das neue Element bzw. eine Referenz darauf.

So definieren Sie das Composite C056 global neu:

1. Öffnen Sie die Datei **EDSD.Segment** in einem Text- oder XML-Editor und navigieren Sie zu **Config | Elements | Composite | C056**.

```
<Composite name="C056" info="CONTACT DETAILS">
  <Data ref="F3413" minOccurs="0"/>
  <Data ref="F3412" minOccurs="0"/>
</Composite>
```

2. Fügen Sie im Segment C056 unter "F3412" die folgende Zeile hinzu:

```
<Data name="X1000" type="string" maxLength="35" minOccurs="0" info="New Element"/>
```

Die `Composite`-Definition sollte nun folgendermaßen aussehen:

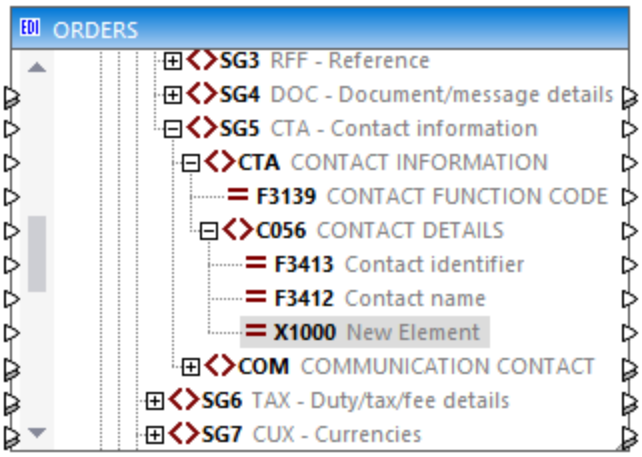
© 2016-2022 Altova GmbH

```
<Composite name="C056" info="CONTACT DETAILS">
  <Data ref="F3413" minOccurs="0"/>
  <Data ref="F3412" minOccurs="0"/>
  <Data name="X1000" type="string" maxLength="35" minOccurs="0" info="New
Element"/>
</Composite>
```

Im oben gezeigten Codefragment wird das neue Feld X1000 im Gegensatz zu anderen Feldern, für die das "ref" Attribut verwendet wird, mit Hilfe des "name" Attributs definiert. Die beiden anderen Felder werden am Beginn der Datei definiert und hier nur referenziert.

Das neue Feld X1000 steht nun für alle Messages, in denen Composite C056 verwendet wird, zur Verfügung. Sie können in MapForce folgendermaßen eine Vorschau auf das neue Feld anzeigen:

1. Klicken Sie im Menü **Einfügen** auf **EDI**. Alternativ dazu können Sie auch in der Symbolleiste auf die Schaltfläche **EDI einfügen** klicken.
2. Klicken Sie auf die Collection "EDIFACT.Nanonull" und wählen Sie die ORDERS Message aus.
3. Wenn Sie aufgefordert werden, eine EDI-Quelldatei auszuwählen, klicken Sie auf die Schaltfläche **Überspringen**. Daraufhin wird die ORDERS-Komponente im Mapping-Fenster angezeigt.
4. Klicken Sie auf die Komponentenüberschrift, drücken Sie **Strg + F** und geben Sie als Suchtext "X1000" ein. Klicken Sie auf **Weitersuchen**, um zur nächsten Instanz des neuen Elements "X1000" zu gelangen. (es sollte sich unter **Envelope > Interchange > Group > Message_ORDERS > SG2 > SG5 > CTA > C056** befinden).



Inline Anpassung

Um den Zugriff auf das neue Feld X1000 *lokal (oder inline)* zu machen, müssen nur an der Datei **ORDERS.config** Änderungen vorgenommen werden. In diesem Fall enthält/referenziert nur das umdefinierte CTA-Segment in der **aktuellen** Message das neue Element, d.h. das CTA-Segment wird lokal umdefiniert, sodass es ein umdefiniertes Composite C056 mit einem benutzerdefinierten neuen Feld X1000 enthält.

So definieren Sie das Composite C056 lokal um:

1. Öffnen Sie die Datei **ORDERS.Config** in einem Text- oder XML-Editor und navigieren Sie zu `Group name="SG5"` (oder suchen Sie nach **SG5**).

```
<Group name="SG5" minOccurs="0" maxOccurs="5" info="CTA - Contact information">
   <Segment ref="CTA"/>
   <Segment ref="COM" minOccurs="0" maxOccurs="5"/>
</Group>
```

2. Ersetzen Sie die Zeile `<Segment ref="CTA"/>` durch die folgenden Zeilen:

```
<Segment name="CTA" id="CTA_ORDERS_SG5" info="CONTACT INFORMATION">
    <Data ref="F3139" minOccurs="0"/>
    <Composite name="C056" minOccurs="0" info="DEPARTMENT OR EMPLOYEE DETAILS">
           <Data ref="F3413" minOccurs="0"/>
           <Data ref="F3412" minOccurs="0"/>
           <Data name="X1000" type="string" maxLength="35" minOccurs="0" info="New
Element"/>
    </Composite>
</Segment>
```

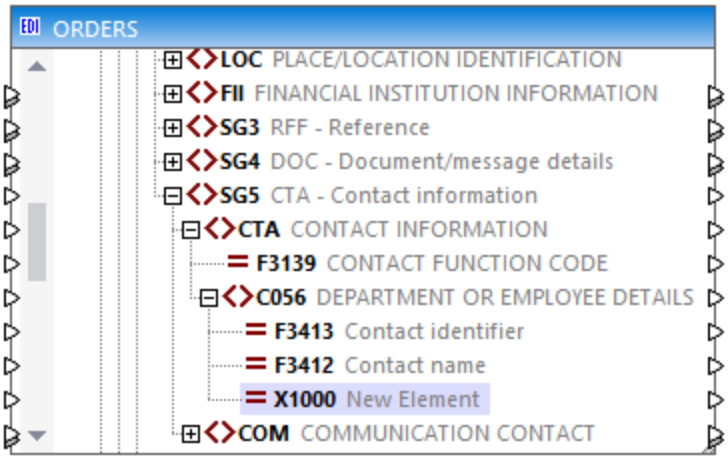
Die `Group`-Definition sollte nun wie unten gezeigt aussehen:

```
<Group name="SG5" minOccurs="0" maxOccurs="5" info="CTA - Contact information">
   <Segment name="CTA" id="CTA_ORDERS_SG5" info="CONTACT INFORMATION">
    <Data ref="F3139" minOccurs="0"/>
    <Composite name="C056" minOccurs="0" info="DEPARTMENT OR EMPLOYEE DETAILS">
           <Data ref="F3413" minOccurs="0"/>
           <Data ref="F3412" minOccurs="0"/>
           <Data name="X1000" type="string" maxLength="35" minOccurs="0" info="New
Element"/>
    </Composite>
   </Segment>
   <Segment ref="COM" minOccurs="0" maxOccurs="5"/>
</Group>
```

Sie können in MapForce folgendermaßen eine Vorschau auf das neue Feld anzeigen:

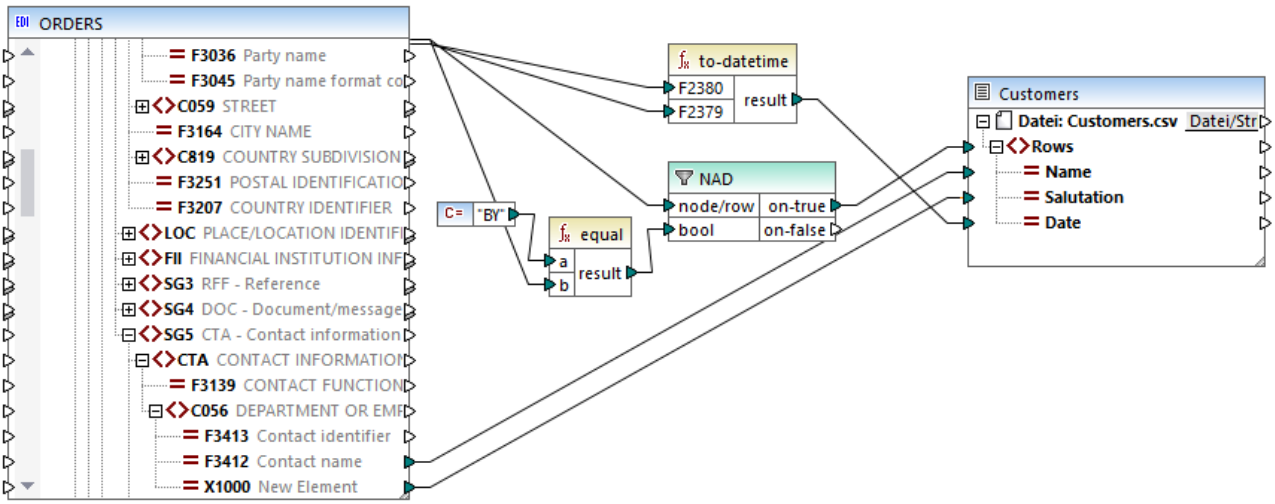
1. Klicken Sie im Menü **Einfügen** auf **EDI**. Alternativ dazu können Sie auch in der Symbolleiste auf die Schaltfläche **EDI einfügen**  klicken.
2. Klicken Sie auf die Collection "EDIFACT.Nanonull" und wählen Sie die ORDERS Message aus.
3. Wenn Sie aufgefordert werden, eine EDI-Quelldatei auszuwählen, klicken Sie auf die Schaltfläche **Überspringen**. Daraufhin wird die ORDERS-Komponente im Mapping-Fenster angezeigt.
4. Navigieren Sie zu **Envelope > Interchange > Group > Message_ORDERS > SG2 > SG5 > CTA > C056**, um das neue Element X1000 zu sehen.

© 2016-2022 Altova GmbH

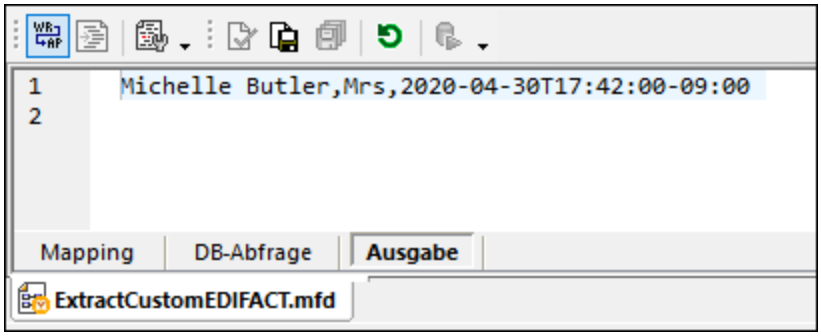


Verwendung der benutzerdefinierten Message in Mappings

Wenn Sie alle Anleitungen oben befolgt haben, können Sie nun die benutzerdefinierte EDIFACT-Konfiguration in Ihren Mappings verwenden. Ein Beispielmapping, in dem die oben erstellte benutzerdefinierte EDIFACT-Konfiguration verwendet wird, ist **ExtractCustomEDIFACT.mfd** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. In dem Beispiel wird die Datei **ORDERS-Custom.EDI** auf eine CSV-Datei gemappt. Das benutzerdefinierte Feld `X1000`, das zur EDI-Struktur hinzugefügt wurde, wird in der Zieldatei auf das Datenelement `Salutation` gemappt.



Um eine Vorschau auf das Mapping-Ergebnis zu sehen, klicken Sie auf das Register **Ausgabe**. Wie erwartet, wird der aus dem benutzerdefinierten X1000 Datenelement der EDIFACT-Datei extrahierte Wert "Mrs" im zweiten Feld der CSV-Datei ("Salutation") angezeigt.



Wie andere Mapping-Design-Dateien in MapForce können Sie auch EDI-Mappings außerhalb von MapForce ausführen. So können Sie diese etwa auf einem anderen Server für die Ausführung mit MapForce Server bereitstellen oder Programmcode generieren, um das Mapping auszuführen. Wenn Sie dieses Mapping-Beispiel durch Generierung von C++-Code ausführen möchten, so beachten Sie, dass eine Klasse namens "CX1000Type" generiert wird, die über die Klasse "CC056Type" aufgerufen werden kann.

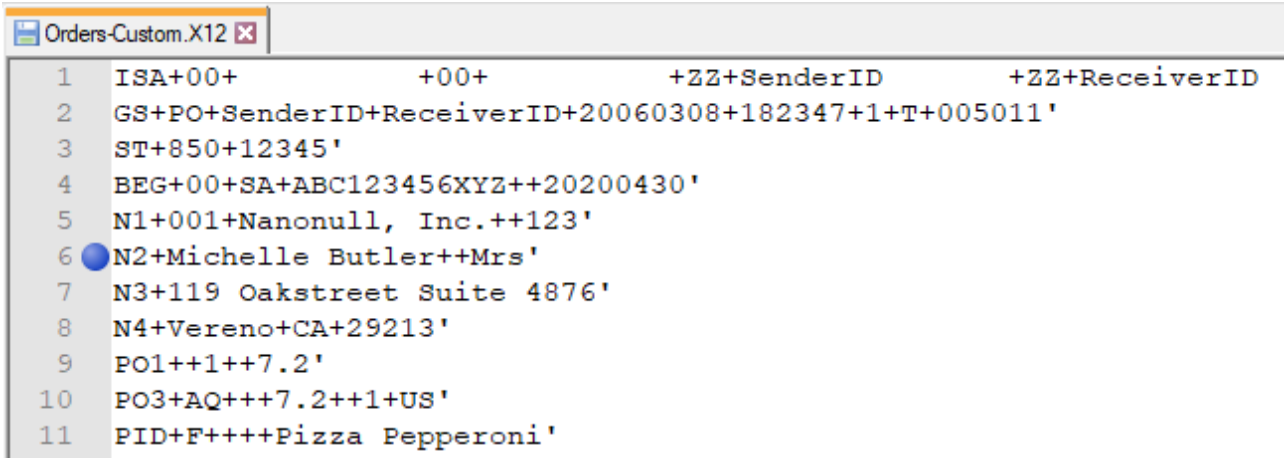
4.3.18.12 Beispiel: Anpassen einer ANSI X12-Transaktion

In diesem Beispiel wird gezeigt, wie Sie MapForce anpassen können, damit es nicht standardkonforme oder geänderte X12-Formate verarbeiten kann. Beachten Sie, dass dieses Beispiel zur Anpassung spezifisch auf Version **6020** von ANSI X12 basiert. Für andere X12-Versionen kann sich die Vorgangsweise etwas unterscheiden.

Das Mapping zu diesem Beispiel finden Sie im folgenden Ordner:
<Dokumente>\Altova\MapForce2023\MapForceExamples\ExtractCustomX12.mfd. Beachten Sie, dass MapForce dieses Mapping nicht erfolgreich validieren kann, falls Sie das Mapping öffnen, ohne vorher die Anleitungen zur benutzerdefinierten Anpassung befolgt zu haben. Der Grund hierfür ist, dass das Mapping eine benutzerdefinierte X12-Collection referenziert, die auf Ihrem Rechner noch nicht zur Verfügung steht (Sie wird in diesem Beispiel erstellt).

Im Beispielmapping werden Daten aus der Datei **Orders-Custom.X12** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples** gelesen. Wenn Sie diese Datei mit einem Text-Editor öffnen, sehen Sie, dass darin ein benutzerdefiniertes N2-Segment verwendet wird: Zeile 6 enthält den zusätzlichen Eintrag ++**Mrs**:

© 2016-2022 Altova GmbH



Bevor Sie Daten aus benutzerdefinierten X12-Dateien wie der oben gezeigten mappen können, muss eine benutzerdefinierte X12-Konfiguration erstellt werden. In diesem Beispiel werden wir die X12-Standardkonfiguration anpassen, indem wir das N2-Segment durch ein neues Datenelement für die Anrede der Person erweitern. Sie finden das angepasste Endergebnis im Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples** als ZIP-Archiv (**X12.Nanonull.zip**).

Voraussetzungen

Um dieses Beispiel durchführen zu können, müssen Sie zuerst die ausführbare Datei "X12 Configuration Files"vom Altova Download Center (https://www.altova.com/de/mapforce/download) herunterladen. Stellen Sie sicher, dass Sie das jeweilige Paket für Ihre Version und Plattform (32-Bit, 64-Bit) von MapForce auswählen,

Das Installationspaket enthält alle von MapForce unterstützten X12-Versionen einschließlich der in diesem Beispiel verwendeten Version "6020". Um die Installation auszuführen, doppelklicken Sie auf die heruntergeladene ausführbare Datei und befolgen Sie die Anweisungen auf dem Bildschirm. Nach der Installation werden für die entsprechenden installierten X12-Konfigurationsversionen im Ordner **C:\Programme\Altova\MapForce2023\MapForceEDI** neue Verzeichnisser erstellt.

Wenn Sie eine 32-Bit-MapForce-Version auf einem 64-Bit-Betriebssystem verwenden, ändern Sie den obigen Verzeichnispfad in **C:\Programme(x86)**-

Einrichten der benutzerdefinierten Dateien

1. Erstellen Sie ein neues Verzeichnis, in dem Ihre benutzerdefinierte EDI-Collection unter dem folgenden Pfad gespeichert wird: **C:\Benutzer\<Name>\Dokumente\Altova\MapForce2023\MapForceEDI\X12.Nanonull**. Der Name "X12.Nanonull" hilft Ihnen, diese benutzerdefinierte EDI-Collection von der mit MapForce installierten Standard-Collection zu unterscheiden.
2. Kopieren Sie die folgenden Dateien aus **C:\Programme\Altova\MapForce2023\MapForceEDI\X12.6020** in **C:\Benutzer\<Name>\Dokumente\Altova\MapForce2023\MapForceEDI\X12.Nanonull**:

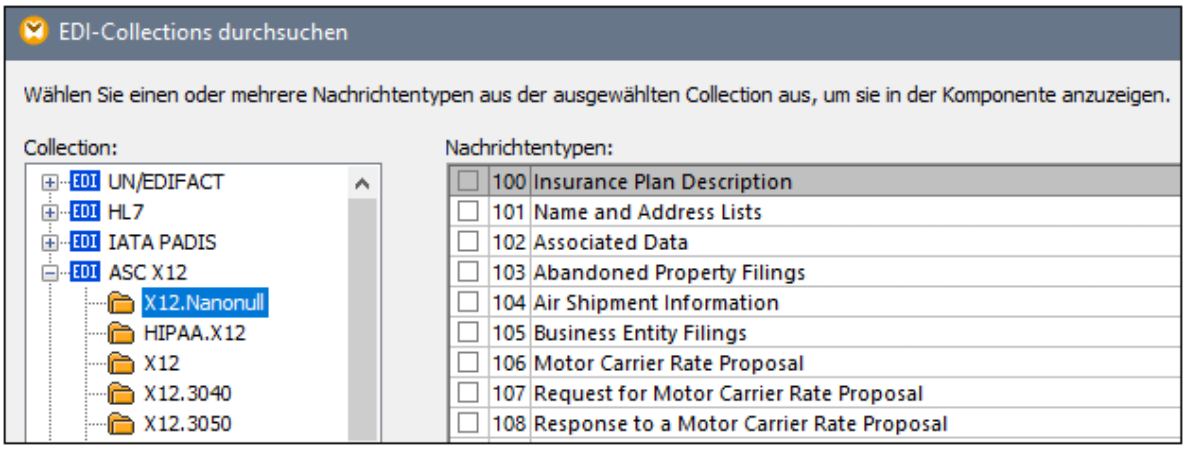
   - 850.Config
   - EDI.Collection
   - Envelope.Config
   - ParserErrors.Config

- X12.Codelist
- X12.Segment

Wozu die einzelnen Dateien dienen, ist unter EDI-Konfigurationsdateien[542] beschrieben.

Die neue **EDI.Collection**-Datei sollte nun von MapForce aufgerufen werden können. Sie können dies folgendermaßen testen:

1. Starten Sie MapForce.
2. Klicken Sie im Menü **Einfügen** auf **EDI**. Alternativ dazu können Sie auch in der Symbolleiste auf die Schaltfläche **EDI einfügen** klicken. Daraufhin wird ein Dialogfeld wie das unten gezeigte geöffnet, das eine neue Collection mit dem Namen "X12.Nanonull" enthält.



Anmerkung: Sie können Ihre benutzerdefinierte EDI-Collection auch, wie unter Erstellen einer benutzerdefinierten EDI-Collection[540] beschrieben, in einem anderen Verzeichnis erstellen.

Konfigurieren der EDI.Collection-Datei

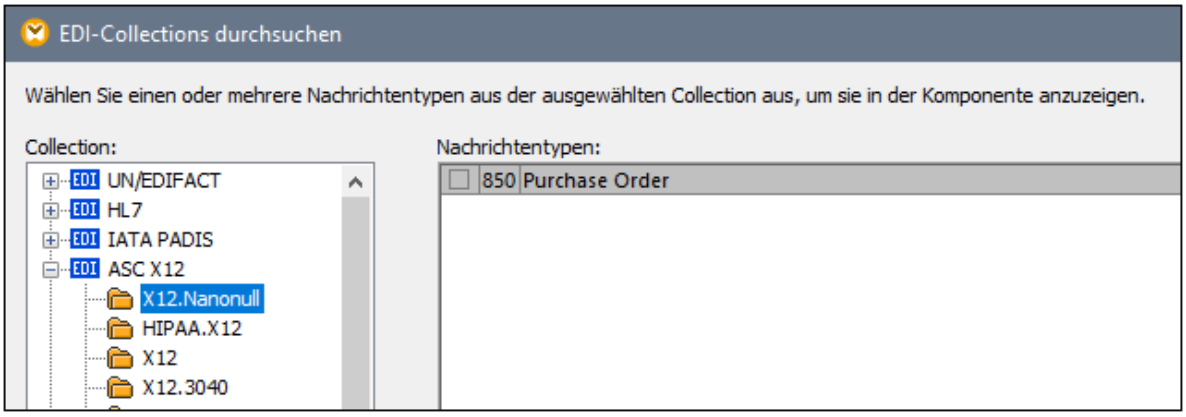
1. Öffnen Sie die Datei **EDI.Collection** in einem XML-Editor (z.B. XMLSpy).
2. Entfernen Sie alle "Message"-Elemente mit Ausnahme der "Purchase Order" Message. Stellen Sie sicher, dass Sie die <Messages> Tags beibehalten.

```
<?xml version="1.0" encoding="UTF-8"?>
<Messages Version="3">
   <Meta>
      <Release>6020</Release>
      <Agency>X12</Agency>
   </Meta>
   <Root File="Envelope.Config"/>
   <Message Type="850" File="850.Config" Description="Purchase Order"/>
</Messages>
```

3. Speichern Sie die Datei.

Wenn Sie die Menübefehl **Einfügen | EDI** wählen, wird in der Collection nun nur ein Nachrichtentyp angezeigt: "850 - Purchase Order".

© 2016-2022 Altova GmbH



Globale Anpassung im Vergleich zu Inline-Anpassung

Ziel dieses Beispiels ist es, das N2 "Additional Name Information"-Segment neu zu definieren, sodass es ein benutzerdefiniertes X1000-Datenelement enthält, in dem die Anrede einer Person (z.B. "Mrs") gespeichert wird. Vor der Anpassung besteht N2 aus nur einem Datenelement "F93 Name". Es gibt verschiedene Methoden, um die Anpassung vorzunehmen:

- **Global** durch Anpassen der Datei **X12.Segment**. "Global" bedeutet, dass alle Segmente in allen Transaktionen, in denen N2 verwendet wird, das neue Element bzw. eine Referenz darauf enthalten.
- **Inline** durch Anpassen der Datei **850.Config**. Lokal bedeutet, dass nur das angepasste Segment (N2) in der aktuellen Message das neue Element enthält.

Globale Anpassung

Um den Zugriff auf das neue Feld X1000 *global* zu machen, müssen nur an der Datei **X12.Segment** Änderungen vorgenommen werden. Alle Segmente in allen Transaktionen, in denen N2 verwendet wird, enthalten/referenzieren dann das neue Element.

So definieren Sie das N2-Segment global neu:

1. Öffnen Sie die Datei **X12.Segment** in einem XML-Editor und navigieren Sie zu **Config | Elements | Segment name="N2"**.

```
<Segment name="N2" info="Additional Name Information">
   <Data ref="F93" mergedEntries="2"/>
</Segment>
```

2. Fügen Sie unter "F93" die folgende Zeile hinzu und speichern Sie die Datei:

```
<Data name="X1000" type="string" maxLength="35" minOccurs="0" info="New Element"/>
```

Die Segment-Definition sollte nun folgendermaßen aussehen:

```
<Segment name="N2" info="Additional Name Information">
   <Data ref="F93" mergedEntries="2"/>
```

```
   <Data name="X1000" type="string" maxLength="35" minOccurs="0" info="New
Element"/>
</Segment>
```

Anmerkungen:

- Das neue X1000-Feld wurde im Gegensatz zu anderen Feldern des Segments, für die das `ref`-Attribut verwendet wurde, mit Hilfe des Attributs `name` definiert. Das Feld F93 ist am Anfang der Datei **X12.Segment** definiert und wird hier nur referenziert.
- Das Attribut `mergedEntries` von F93 legt fest, dass es zwei aufeinander folgende Instanzen dieses Elements mit demselben Namen geben darf. Dies entspricht der Definition dieses Felds in der ASC X12-Spezifikation. Aus diesem Grund sieht das N2-Segment in der EDI-Quelldatei, die hier angepasst wird, folgendermaßen aus:

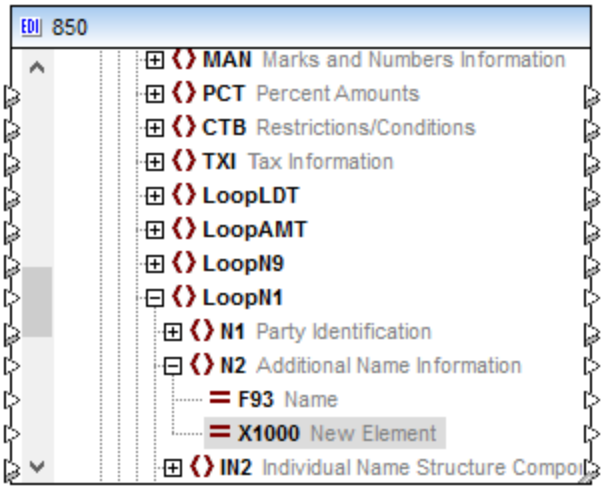
  ```
  N2+Michelle Butler++Mrs'
  ```

  Beachten Sie in der Zeile oben die beiden aufeinander folgenden "+"-Trennzeichen zwischen "Michelle Butler" und "Mrs". Der Grund hierfür ist, dass das N2-Segment aus einem "F93"-Datenelement ("Michelle Butler"), gefolgt von einem "+"-Trennzeichen, gefolgt vom zweiten "F93" Element (welches leer ist), gefolgt von einem "+"-Trennzeichen, gefolgt vom benutzerdefinierten Datenelement "X1000" ("Mrs") besteht. Nähere Informationen zum Attribut `mergedEntries` finden Sie unter Behandlung mehrerer aufeinander folgender Datenelemente 551.

Das neue Feld X1000 steht nun für alle Transaktionen, in denen das N2-Segment verwendet wird, zur Verfügung. Sie können in MapForce folgendermaßen eine Vorschau auf das neue Feld anzeigen:

1. Klicken Sie im Menü **Einfügen** auf **EDI**. Alternativ dazu können Sie auch in der Symbolleiste auf die Schaltfläche **EDI einfügen**  klicken.
2. Klicken Sie auf die Collection "X12.Nanonull" und wählen Sie die "Purchase Order" Message aus.
3. Wenn Sie aufgefordert werden, eine EDI-Quelldatei auszuwählen, klicken Sie auf die Schaltfläche **Überspringen**. Daraufhin wird die 850 -Komponente im Mapping-Fenster angezeigt.
4. Klicken Sie auf die Komponentenüberschrift, drücken Sie **Strg + F** und geben Sie als Suchtext "X1000" ein. Klicken Sie auf **Weitersuchen**, um zur nächsten Instanz des neuen Elements "X1000" zu gelangen. (es sollte sich unter **Envelope > Interchange > Group > Message_850 > LoopN1 > N2** befinden).

© 2016-2022 Altova GmbH



Lokale Anpassung

Um den Zugriff auf das neue Feld X1000 lokal zu machen, müssen nur an der Datei **850.Config** Änderungen vorgenommen werden. In diesem Fall enthält/referenziert nur das Segment N2 in der Transaktion "850 - Purchase Order" das neue Feld X1000.

So definieren Sie das N2-Segment lokal um:

1. Öffnen Sie die Datei **850.Config** in einem Text- oder XML-Editor und navigieren Sie zu `Group name="LoopN1"` (oder suchen Sie nach **LoopN1**).

```
<Group name="LoopN1" maxOccurs="10" minOccurs="0">
   <Segment ref="N1"/>
   <Segment ref="N2" minOccurs="0" maxOccurs="2"/>

   ... ...

</Group>
```

2. Ersetzen Sie die Zeile `<Segment ref="N2" minOccurs="0" maxOccurs="2"/>` durch die folgenden Zeilen:

```
<Segment name="N2" info="Additional Name Information">
   <Data ref="F93" mergedEntries="2"/>
   <Data name="X1000" type="string" maxLength="35" minOccurs="0" info="New
Element"/>
</Segment>
```

Die `Group`-Definition sollte nun wie unten gezeigt aussehen:

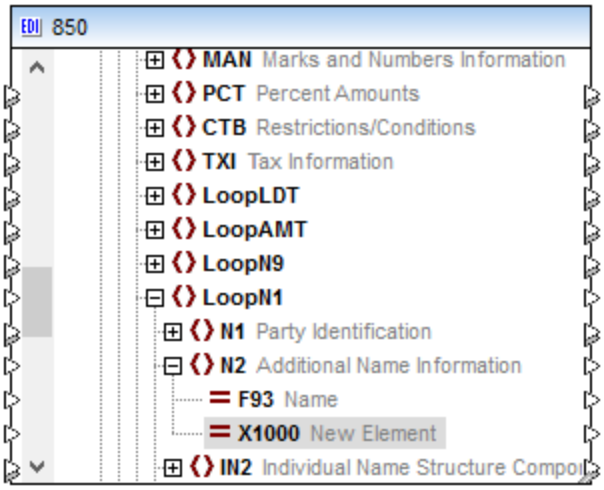
```
<Group name="LoopN1" maxOccurs="10" minOccurs="0">
   <Segment ref="N1"/>
```

```
   <Segment name="N2" info="Additional Name Information">
      <Data ref="F93" mergedEntries="2"/>
      <Data name="X1000" type="string" maxLength="35" minOccurs="0" info="New
Element"/>
   </Segment>

   ... ...

</Group>
```

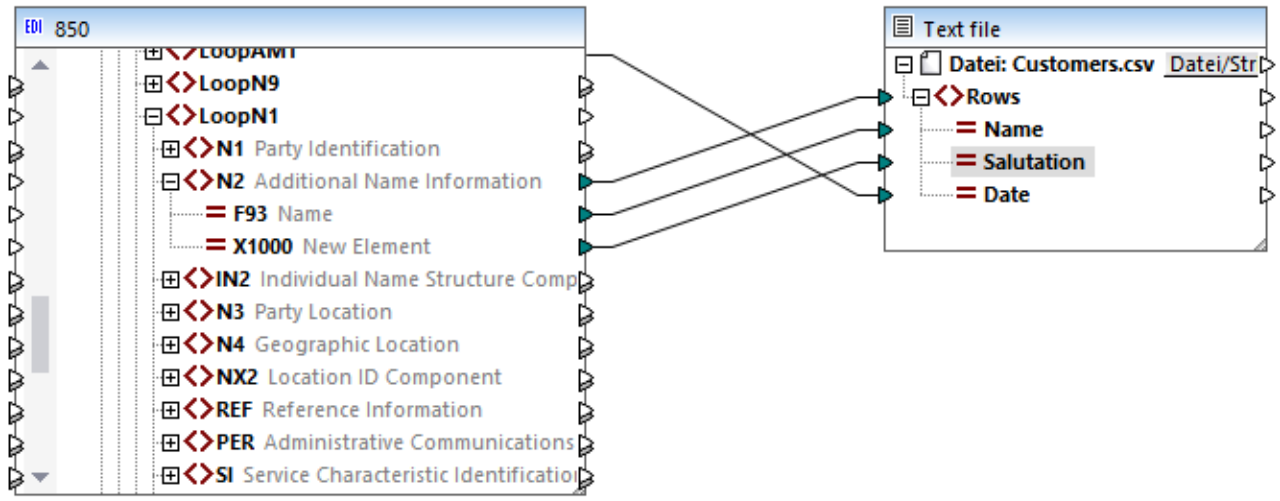
Sie können nun in MapForce eine Vorschau auf das neue Feld anzeigen. Gehen Sie dabei vor, wie oben für globale Anpassungen beschrieben.



Verwendung der benutzerdefinierten Message in Mappings

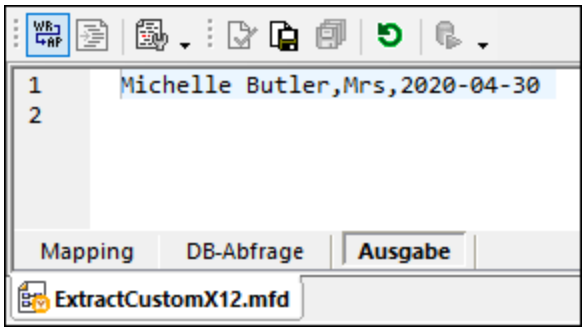
Wenn Sie alle Anleitungen oben befolgt haben, können Sie nun die benutzerdefinierte X12-Konfiguration in Ihren Mappings verwenden. Ein Beispielmapping, in dem die oben erstellte benutzerdefinierte EDIFACT-Konfiguration verwendet wird, ist **ExtractCustomX12.mfd** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. In dem Beispiel wird die Datei **ORDERS-Custom.X12** auf eine CSV-Datei gemappt. Das benutzerdefinierte Feld `X1000`, das in der Anleitung oben hinzugefügt wurde, wird in der Zieldatei auf das Datenelement `Salutation` gemappt.

© 2016-2022 Altova GmbH



ExtractCustomX12.mfd

Um eine Vorschau auf das Mapping-Ergebnis zu sehen, klicken Sie auf das Register **Ausgabe**. Wie erwartet, wird das Zieldatenelement `Salutation` mit dem aus dem benutzerdefinierten X1000-Datenelement der X12-Datei extrahierten Wert "Mrs" befüllt.



4.3.19 Anpassen der EDI-Validierung

In diesem Abschnitt wird beschrieben, wie die Validierung von EDI Messages in MapForce zu folgenden Zwecken angepasst werden kann:

- Lockerung der Validierungsregeln bei Feldern, auf die die Validierungsregeln durch den Standard nicht zur Gänze angewendet werden müssen und bei denen die Validierungsregeln mit Hilfe externer Standards erweitert werden können.

  So gibt es z.B. EDI-Datenelemente, für die zusätzlich zu den vordefinierten Codewerten in externen ISO-Standards oder UN/ECE-Recommendations definierte Codewerte zulässig sind. In diesem Fall können Sie bei der Verarbeitung solcher EDI-Felder MapForce so konfigurieren, dass bei einem externen Code anstatt eines Validierungsfehlers eine Warnung zurückgegeben und das Mapping

fortgesetzt wird. Nähere Informationen dazu finden Sie unter Gelockerte Validierungsregeln für Felder 574.

- Ausführung vollständiger Validierung von Feldern wie den oben beschriebenen durch Anpassen benutzerdefinierter Konfigurationsdateien an MapForce.

  Um z.B. das Datenelement 4405 des UN/EDIFACT D.16A-Standards zu validieren, können Sie eine benutzerdefinierte Konfigurationsdatei hinzufügen, die die in der UN/ECE Recommendation 24 ("Trade and Transport Status Codes") definierten Validierungsregeln enthält. Die in der UN/ECE Recommendation 24 definierten Codewerte sind nicht im EDIFACT-Standard enthalten (und daher nicht Teil der Standardgruppe der mit MapForce bereitgestellten EDIFACT-Konfigurationsdateien). Nähere Informationen dazu finden Sie unter Anpassen von Codewerten aus externen Quellen 577.

- Globale Durchführung einer Validierung für alle Messages, in denen ein Feld verwendet wird, oder lokale Durchführung einer Validierung für eine bestimmte Message.

  Sie können die Validierung so konfigurieren, dass sie entweder global oder lokal angewendet wird. Bei der globalen Validierung werden alle Instanzen eines bestimmten Felds in allen Messages validiert. Eine lokale (oder "inline") Validierung wird nur auf eine bestimmte Message angewendet. Nähere Informationen dazu finden Sie unter Definieren von Codewerten auf Message-Ebene (inline) 580.

- Definieren komplexer Validierungsregeln für Felder, die nicht nur auf Feld-, sondern auf Zeichenebene validiert werden müssen.

  Bestimmte Messages wie z.B. die VERMAS (Verification of Mass) Message des UN/EDIFACT D.16A-Standards, enthalten Felder, auf die komplexe Validierungsregeln angewendet werden müssen (z.B. das Datenelement 8155). Für dieses Feld sind in ISO 6346 definierte Codewerte zulässig. Um ein Feld gemäß ISO 6346 zu validieren, kann MapForce mit Hilfe von Konfigurationsdateien angepasst werden, um zu überprüfen, ob bestimmte Zeichen an bestimmten Positionen im Feld vorkommen. Nähere Informationen dazu finden Sie unter Konfigurieren der Validierung auf Zeichenebene 581. (Beachten Sie, dass ISO 6346 hier nur als Beispiel erwähnt wird. Die MapForce-Konfigurationsdateien enthalten bereits Validierungsregeln für ISO 6346.)

4.3.19.1 Gelockerte Validierungsregeln für Felder

Es gibt Fälle, in denen die in einem Feld zulässigen Codewerte nicht zur Gänze in einem Standard vorgegeben sind, sodass durch externe Standards oder Recommendations definierte Codewerte verwendet werden können. So sind z.B. laut EDIFACT D16A in einigen Feldern in externen ISO-Standards oder UN/ECE Recommendations definierte Codewerte zulässig. So sind etwa für das Datenelement 4405 146 Codewerte definiert, es können jedoch laut externer UN/ECE Recommendation 24 auch 346 Codewerte verwendet werden.

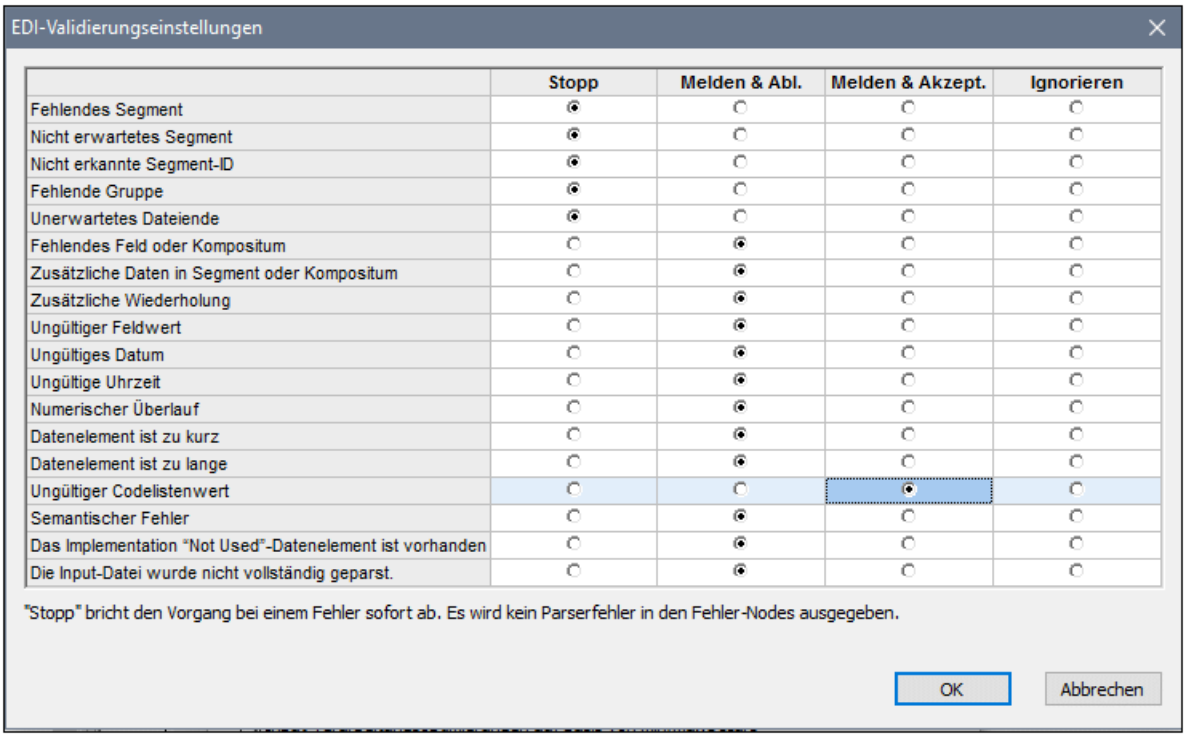
Die möglichen Codewerte für ein Feld werden in MapForce anhand einer Konfigurationsdatei mit der Erweiterung .codelist (im Folgenden als ".codelist"-Datei bezeichnet) validiert. Wenn MapForce eine EDI-Datei verarbeitet, erzeugt es normalerweise einen Validierungsfehler, wenn ein Feld keinen der in der .codelist-Datei für diese EDI-Variante definierten Werte hat (vorausgesetzt Sie haben die Standardvalidierungsoptionen nicht dahingehend geändert, dass ungültige Feldwerte stillschweigend ignoriert werden, siehe Validierung von EDI-Komponenten 471.) Die .codelist-Dateien für die einzelnen EDI-Varianten befinden sich im Verzeichnis **C:\Programme\Altova\MapForce2023\MapForceEDI**. So sind etwa in der Datei **UNCL.codelist** im Verzeichnis **C:\Programme\Altova\MapForce2023\MapForceEDI\EDIFACT** mögliche Codewerte für Werte in UN/EDIFACT-Feldern gespeichert.

© 2016-2022 Altova GmbH

Die .codelist-Standarddatei enthält alle möglichen Codewerte für ein Feld gemäß dem jeweiligen EDI-Standard. Wenn Codewerte vom Standard nicht explizit vorgegeben werden, sondern in externen Standards definiert sind, wird die Validierung als unvollständig betrachtet. Wenn die Liste der Codewerte für ein bestimmtes Feld (z.B. Datenelement 4405 von UN/EDIFACT D16A) laut EDI-Standard nicht zur Gänze im EDI-Standard vorgegeben ist, würden bei der Verarbeitung des Felds während der Mapping-Ausführung normalerweise Validierungsfehler ausgegeben werden. Dabei gibt MapForce (oder der Ausführungsprozessor, mit dem das EDI-Mapping ausgeführt wird, wie z.B. MapForce Server oder ein C#, C++ oder Java-Programm) einen Fehler im folgenden Format aus: **{Wert} ist kein gültiger Wert für das Feld {Feld}**. Das bedeutet, dass der Feldwert in der Liste der in der .codelist-Datei definierten vorhandenen Codelistenwerte nicht gefunden wurde.

Damit das Mapping bei Fehlern wie dem oben beschriebenen nicht abgebrochen wird, können Sie Folgendes tun:

1. Ändern Sie das Validierungsverhalten über die EDI-Validierungseinstellungen (Aktivieren Sie z.B. im Dialogfeld "EDI-Validierungseinstellungen" das Kontrollkästchen **Melden und Akzeptieren** neben "Ungültiger Codelistenwert". (Um die EDI-Validierungseinstellungen zu konfigurieren, klicken Sie im Dialogfeld EDI-Komponenteneinstellungen 466 auf die Schaltfläche **Validierung**.)



2. Lockern Sie die Feldvalidierungsregeln in den Konfigurationsdateien, wie unten gezeigt.

MapForce bietet eine Möglichkeit, um ein Feld zu kennzeichnen, das "unvollständige" Validierungsregeln aufweist. Dadurch stellen Sie sicher, dass MapForce (oder der Ausführungsprozessor, mit dem das EDI-Mapping ausgeführt wird, wie z.B. MapForce Server oder ein C#, C++ oder Java-Programm) keine Fehlermeldung erzeugt, wenn das Feld verarbeitet wird; stattdessen wird eine Warnmeldung angezeigt und das Mapping wird fortgesetzt. "Unvollständig" bedeutet also, dass nur die in der MapForce .codelist-Standarddatei definierten Codewerte bei der Validierung berücksichtigt werden sollen, nicht aber die extern definierten.

Zur einfacheren und flexibleren Handhabung werden in den mit MapForce bereitgestellten .codelist-Standardkonfigurationsdateien bestimmte Felder als Felder behandelt, die eine unvollständige Validierung aufweisen.

Standardmäßig erzeugt MapForce Warnungen (und nicht Fehler), wenn eines der folgenden Fehler Codewerte enthält, die in externen Spezifikationen definiert sind:

- Datenelement 4053 von UN/EDIFACT D.16A
- Datenelement 4277 von UN/EDIFACT D.16A
- Datenelement 4405 von UN/EDIFACT D.16A
- Datenelement 8179 von UN/EDIFACT D.16A

Die Warnung wird ausgegeben, um Sie darüber zu informieren, dass die Validierung dieses Felds unvollständig ist. Durch eine unvollständige Validierung können unter Umständen vom Mapping fehlerhafte Daten erzeugt werden. Um dies zu verhindern, wird empfohlen, mittels benutzerdefinierter .codelist-Dateien für MapForce eine vollständige Feldvalidierung durchzuführen. Mittels benutzerdefinierter .codelist-Dateien können Sie externe Codewerte in einem Format definieren, das von MapForce verstanden wird, sodass Sie eine vollständige Validierung des Felds durchführen können. Nähere Informationen hierzu finden Sie unter Anpassen von Codewerten aus externen Quellen 577.

Anmerkung: Das Datenelement 8155 wird ebenfalls auf Basis eines externen Standards (ISO 6346 Standard) validiert; für dieses Feld stellt MapForce jedoch eine vollständige Validierung mit Hilfe einer separaten .codelist-Datei zur Verfügung. Siehe Anpassen von Codewerten aus externen Quellen 577).

Gegebenenfalls können Sie weitere Felder anpassen, damit diese als Felder behandelt werden, die unvollständige Validierungsregeln aufweisen. Gehen Sie dazu folgendermaßen vor:

1. Erstellen Sie eine benutzerdefinierte EDI-Collection, in der Ihre Änderungen an den Validierungsregeln gespeichert werden. Es wird nicht empfohlen, die standardmäßig mit MapForce installierten EDI-Collections zu ändern. Nähere Informationen dazu finden Sie unter Erstellen einer benutzerdefinierten EDI-Collection 540.
2. Navigieren Sie zur .codelist-Datei in Ihrer benutzerdefinierten EDI-Collection und öffnen Sie sie mit einem XML-Editor wie z.B. XMLSpy.
3. Suchen Sie das `<Field>`-Element, das dem gewünschten Datenelement entspricht, und fügen Sie das Attribut `Incomplete="1"` (unten markiert) zu diesem Element hinzu, z.B.:

```
<Field Name="F8179" Length="8" Type="Alphanumeric" Incomplete="1">
   <Title>Transport means description code</Title>
   <Description>Code specifying the means of transport.</Description>
   <Note>1 Use UN/ECE recommendation 28.
</Note>
   <Values>
      <Value Code="1">
         <Title>Box tricycle</Title>
         <Description>Tricycle with a basket for delivering goods.</Description>
      </Value>
   </Values>
</Field>
```


© 2016-2022 Altova GmbH

Falls es für die Verarbeitung Ihrer Datei notwendig ist, dass für Felder, die bereits als "unvollständig" (incomplete) gekennzeichnet wurden, ein Fehler anstelle einer Warnung ausgegeben werden soll, so ändern Sie den Attributwert von `Incomplete` in den Wert "0" anstelle von "1". Dadurch wird eine strenge Feldvalidierung durchgeführt und es werden Fehler (und nicht Warnungen) ausgegeben, wenn der Feldwert nicht Teil der in der .codelist-Datei definierten Gruppe von Werten ist.

4.3.19.2 Anpassen von Codewerten aus externen Quellen

In den standardmäßig mit MapForce zur Verfügung gestellten EDI-Konfigurationsdateien wurden einige Datenelemente als unvollständig validiert konfiguriert (siehe Gelockerte Validierungsregeln für Felder 574). Der Grund dafür ist, dass die möglichen Codewerte für dies Datenelemente auf EDI-Standard-externen Spezifikationen basieren.

Wenn eine unvollständige (oder weniger strenge) Validierung eines Felds nicht in Frage kommt oder unerwünscht ist, können Sie MapForce auch so konfigurieren, dass eine vollständige Feldvalidierung durchgeführt wird. Zu diesem Zweck müssen Sie zusätzlich zu den bereits in MapForce verfügbaren .codelist-Standarddateien benutzerdefinierte .codelist-Dateien hinzufügen. Mit Hilfe der benutzerdefinierten .codelist-Dateien können Sie Codewerte, die standardmäßig nicht in den EDI-Standards definiert sind und somit in den MapForce-Konfigurationsdateien nicht vorhanden sind, hinzufügen. Dies gilt normalerweise für Codewerte, die in externen Standards oder Recommendations definiert sind.

Beachten Sie, dass für jeden möglichen Codewert, der validiert werden soll (und dabei kann es sich um Hunderte von Datensätzen handeln), ein Eintrag in der benutzerdefinierten .codelist-Datei erforderlich ist. Ein Beispiel dafür ist das Datenelement 4405 des UN/EDIFACT D16A-Standards. Zusätzlich zu den im UN/EDIFACT-Standard definierten Codewerten sind in diesem Feld auch Codewerte zulässig, die in der UN/ECE Recommendation 24 ("Trade and Transport Status Codes", siehe http://www.unece.org/fileadmin/DAM/cefact/recommendations/rec24/rec24_ecetrd258e.pdf) definiert sind. Es handelt sich hierbei um 346 Codes. Für all diese externen Codewerte muss eine benutzerdefinierte .codelist-Datei erstellt werden, die für jeden der 346 Codes aus der Recommendation einen Eintrag enthält. Wenn es daher Ihr konkreter Fall gestattet, ist es einfacher, die unvollständige Validierung zu verwenden (siehe Gelockerte Validierungsregeln für Felder 574). Bedenken Sie jedoch, dass bei einer unvollständigen Validierung das Risiko besteht, dass ungültige Daten in der Mapping-Ausgabe erzeugt werden, daher wird stattdessen eine vollständige Feldvalidierung empfohlen.

Um bei Feldern, deren Codewerte in einem externen Standard definiert sind, eine vollständige Validierung durchzuführen, gehen Sie folgendermaßen vor:

1. Erstellen Sie eine benutzerdefinierte EDI-Collection, in der Ihre Änderungen an den Standardvalidierungsregeln gespeichert werden. Nähere Informationen dazu finden Sie unter Erstellen einer benutzerdefinierten EDI-Collection 540.

   Sie können externe Codelisten zwar auch durch Bearbeitung vorhandener mit MapForce bereitgestellter .codelist-Dateien (in diesem Beispiel **UNCL.codelist**) definieren, doch wird dies nicht empfohlen. Es empfiehlt sich, externe Codelisten immer in benutzerdefinierten Dateien zu definieren und Änderungen an den MapForce-Standardkonfiguarationsdateien möglichst zu vermeiden.

2. Suchen Sie in Ihrem Verzeichnis für die benutzerdefinierte EDI-Collection nach der als Vorlage verwendeten .codelist-Datei. Wenn Sie z.B. eine benutzerdefinierte .codelist-Datei zur Validierung von UN/EDIFACT Messages hinzufügen, verwenden Sie als Vorlage die **UNCL.Codelist**-Datei.
3. Benennen Sie die .codelist-Vorlagendatei um, um eine neue .codelist-Datei mit einem kurzen beschreibenden Namen zu erhalten. Diese Datei wird von anderen MapForce-Konfigurationsdateien aus referenziert. Nennen wir die benutzerdefinierte .codelist-Datei in diesem Beispiel **UNECE24.codelist**.
4. Löschen Sie mit Hilfe eines XML-Editors wie XMLSpy den gesamten Inhalt zwischen den Tags `<Codelist>` und `</Codelist>` aus der neuen Datei. Die Datei sollte nun folgendermaßen aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Version="5"
xsi:noNamespaceSchemaLocation="../EDIConfig5.xsd">
   <Codelist>

...

   </Codelist>
</Config>
```

5. Fügen Sie unter `<Codelist>` eine neue `<Field>`-Struktur wie unten gezeigt hinzu. In diesem Beispiel wird angenommen, dass wir externe Codewerte für das Datenelement 4405 des UN/EDIFACT-Standards hinzufügen. Ersetzen Sie die Feldattribute andernfalls bedarfsgemäß. Den korrekten Namen, die korrekte Länge und den korrekten Typ der einzelnen Felder finden Sie in der von MapForce bereitgestellten .codelist-Standarddatei (in diesem Beispiel **UNCL.Codelist**).

```
<?xml version="1.0" encoding="UTF-8"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Version="5"
xsi:noNamespaceSchemaLocation="../EDIConfig5.xsd">
   <Codelist>
      <Field Name="F4405" Length="3" Type="Alphanumeric">
         <Title>Trade and transport status codes</Title>
         <Description>Code specifying a status.</Description>
         <Values>

...

         <Values>
      </Field>
   </Codelist>
</Config>
```

6. Fügen Sie unter `<Values>` für jeden externen Code ein neues `<Value>`-Element hinzu, wie unten gezeigt. In diesem Beispiel fügen wir die ersten beiden in der UN/ECE Recommendation 24 ("Trade and Transport Status Codes", siehe http://www.unece.org/fileadmin/DAM/cefact/recommendations/rec24/rec24_ecetrd258e.pdf) definierten Codes hinzu.

```
<?xml version="1.0" encoding="UTF-8"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Version="5"
xsi:noNamespaceSchemaLocation="../EDIConfig5.xsd">
   <Codelist>
```


© 2016-2022 Altova GmbH

```
      <Field Name="F4405" Length="3" Type="Alphanumeric">
         <Title>Trade and transport status codes</Title>
         <Description>Code specifying a status.</Description>
         <Values>
            <Value Code="1">
               <Title>Arrival, completed</Title>
               <Description>The goods/consignment/equipment/means of transport has
arrived.</Description>
            </Value>
            <Value Code="2">
               <Title>Loading, authorized</Title>
               <Description>Authorization to load has been given.</Description>
            </Value>
         <Values>
      </Field>
   </Codelist>
</Config>
```

Die benutzerdefinierte .codelist-Datei ist jetzt fertig definiert, aber noch nicht an MapForce gebunden. Beachten Sie, dass wir in diesem Beispiel nur zwei externe Codes ("1" und "2") hinzugefügt haben, dass die benutzerdefinierte .codelist-Datei jedoch in realen Einsatzszenarien alle im externen Standard definierten Codewerte enthalten sollte.

Bevor die benutzerdefinierte .codelist-Datei von MapForce erkannt werden kann, muss noch ein letzter Schritt erledigt werden:

1. Da das Datenelement 4405 nun keine unvollständige Validierung mehr aufweist, müssen Sie MapForce anweisen, eine vollständige Validierung durchzuführen. Dazu muss das Attribut `Incomplete="1"` für dieses Feld in der Datei **UNCL.codelist** geändert werden (das Attribut muss entweder entfernt oder auf `Incomplete="0"` gesetzt werden). Ändern Sie die Datei **UNCL.codelist** daher, wie unten gezeigt:

```
...

<Field Name="F4405" Length="3" Type="Alphanumeric" Incomplete="0">
         <Title>Status description code</Title>
         <Description>Code specifying a status.</Description>
         <Note>1 For transport status, use UN/ECE Recommendation 24.</Note>

...
```

2. Fügen Sie in der Datei **UNCL.codelist** eine Referenz zur neu erstellten Datei **UNECE24.codelist** hinzu, wie unten gezeigt:

```
<?xml version="1.0" encoding="UTF-8"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Version="5"
xsi:noNamespaceSchemaLocation="../EDIConfig5.xsd">
   <Include href="UNECE24.codelist"/>
   <Codelist>

....
```

Sie haben die MapForce-Konfigurationsdateien nun so angepasst, dass das Feld 4405 anhand von in einer benutzerdefinierten Datei gespeicherten Codewerten validiert wird. Diese Methode kann auch auf andere EDI-Varianten angewendet werden, um eine vollständige Feldvalidierung anhand eines externen Standards durchzuführen.

Beachten Sie, dass die Feldvalidierung mit den obigen Schritten global für jedes Segment, in dem Feld 4405 verwendet wird, definiert wird. In einigen Fällen sollen die benutzerdefinierten Codelisten nur auf bestimmte Messages angewendet werden (d.h. die Konfiguration soll nicht global auf alle Messages, sondern inline oder lokal auf eine bestimmte Message angewendet werden). Nähere Informationen zur Anwendung der inline-Methode finden Sie unter Definieren von Codewerten auf Message-Ebene (inline) 580.

4.3.19.3 Definieren von Codewerten auf Message-Ebene (inline)

In einem früheren Kapitel, Anpassen von Codewerten aus externen Quellen 577, wurde beschrieben, wie Sie die Feldvalidierung mittels benutzerdefinierter Codelisten *global* definieren (die Validierung wird auf jede Message, in der ein bestimmtes Feld verwendet werden kann, angewendet). Falls die Validierung nur auf eine bestimmte Message angewendet werden soll, besteht die Möglichkeit, benutzerdefinierte Codelisten inline (lokal für eine bestimmte Message) hinzuzufügen. Gehen Sie in diesem Fall folgendermaßen vor:

1. Erstellen Sie eine benutzerdefinierte EDI-Collection, falls Sie dies noch nicht getan haben, siehe Erstellen einer benutzerdefinierten EDI-Collection 540.
2. Suchen Sie in Ihrem Verzeichnis für die benutzerdefinierte EDI-Collection nach der **{Message}.config**-Datei für die entsprechende Message und öffnen Sie sie in einem XML-Editor.
3. Suchen Sie nach dem Element `<Data>` unter dem gewünschten `<Segment>` und fügen Sie die benutzerdefinierten Codewerte als Child-Elemente von `<Values>` hinzu, z.B.:

```
<Group name="Message_270-B1" info="Health Care Eligibility Benefit Inquiry"
maxOccurs="unbounded">
      <Segment name="ST" info="Transaction Set Header">
        <Condition path="F1705" value="005010X279A1" />
        <Data ref="F143" info="Transaction Set Identifier Code">
          <Values>
            <Value Code="270" />

[>>>> custom values can be added here...]

          </Values>
        </Data>
        <Data ref="F329" info="Transaction Set Control Number" />
        <Data ref="F1705" info="Implementation Convention Reference" />
      </Segment>
```

Beachten Sie die folgenden Punkte:

- Wenn ein Feld sowohl lokal definierte Werte (wie die im Codefragment oben definierten) als auch global (in einer .codelist-Datei, wie unter Anpassen von Codewerten aus externen Quellen 577 beschrieben) definierte Codewerte hat, stellen Sie sicher, dass lokale Codewerte auch als globale Codewerte definiert werden. Andernfalls wird ein Validierungsfehler*** ausgegeben: **{Wert} ist kein zulässiger Wert für das Feld {Feld}**.

 © 2016-2022 Altova GmbH

- Die Inline Validierung (lokale Validierung) kann verwendet werden, wenn eine Validierung auf Feldebene erforderlich ist (d.h. die möglichen Werte, die ein Feld erhalten kann, können in Form einer Liste ausgedrückt werden). Die Inline Validierung kann nicht für diejenigen Felder verwendet werden, für die komplexe Validierungsregeln auf Zeichenebene erforderlich sind. Wenn komplexe Validierungsregeln auf Zeichenebene (wie z.B. die in ISO 6346 für das Datenelement 8155 des UN/EDIFACT-Standards definierten) auf ein Feld angewendet werden müssen, fügen Sie die Codewerte global hinzu, wie unter Anpassen von Codewerten aus externen Quellen 577 beschrieben. Anschließend können Sie die Validierung wie unter Konfigurieren der Validierung auf Zeichenebene 581 beschrieben konfigurieren.

*** Beachten Sie, dass beim tatsächlichen Validierungsergebnis die im Dialogfeld "EDI-Validierungseinstellungen" für die Regel "Ungültiger Codelistenwert" konfigurierten Einstellungen berücksichtigt werden. (Um dieses Dialogfeld aufzurufen, klicken Sie im Dialogfeld EDI-Komponenteneinstellungen 466 auf die Schaltfläche **Validierung**).

4.3.19.4 Konfigurieren der Validierung auf Zeichenebene

In den meisten Fällen bedeutet eine Validierung von EDI-Feldern, dass der Feldwert zur Mapping-Laufzeit mit einer Liste von in einer MapForce-Konfigurationsdatei definierten möglichen Werten verglichen wird. Wenn der Feldwert mit einem der möglichen in der Konfigurationsdatei definierten Werte übereinstimmt, gilt er als gültig.

Für einige Felder werden jedoch manchmal komplexere Validierungsregeln benötigt, sodass ihre Validierungslogik nicht in Form einer einfachen Liste möglicher Werte ausgedrückt werden kann. Ein Beispiel für ein Feld, für das komplexe Validierungsregeln erforderlich sind, ist das in der VERMAS (Verification of Mass) Message des UN/EDIFACT D16A-Standards verwendete Datenelement 8155. In diesem Feld sind nicht nur im UN/EDIFACT-Standard definierte Codewerte, sondern auch externe, durch den ISO 6346-Standard definierte Codes zulässig. Im ISO 6346-Standard sind Validierungsregeln auf Zeichenebene definiert, nämlich:

- Das erste Zeichen kann eines der folgenden sein: 1, 2, 3, 4, A, B, C, D, E, F, G, H, K, L, M, N, P
- Das zweite Zeichen kann eines der folgenden sein: 0, 2, 4, 5, 6, 8, 9, C, D, E, F, L, M, N, P
- Das dritte und vierte Zeichen kann eines der folgenden sein: G0, G1, G2, G3, usw. (zu viele Werte, um diese hier aufzulisten).

Auch eine Validierungslogik wie die oben gezeigte kann für MapForce angepasst werden. In diesem Fall validiert MapForce das Datenelement 8155 nicht nur gemäß den UN/EDIFACT-Codewerten, sondern auch unter Berücksichtigung der obigen Regeln aus dem ISO 6346-Standard.

Aus Sicht einer MapForce-Konfiguration können die drei oben aufgezählten Regeln als drei separate Wertelisten betrachtet werden. Die erste Liste ist ein Zeichen lang, die zweite Liste ist ein Zeichen lang und die dritte Liste ist zwei Zeichen lang. Alle drei Listen sind finit und deren Werte sind bekannt. Aufgrund dieser Informationen kann der Wert des Datenelements mit Hilfe der MapForce-Konfiguration validiert werden.

Ein Beispiel dafür, wie die Validierung auf Zeichenebene konfiguriert werden muss, finden Sie in der Datei **ISO6346.Codelist** aus dem Verzeichnis **C:\Programme\Altova\MapForce2023\MapForceEDI\EDIFACT**. Diese mit MapForce bereitgestellte Datei ist so konfiguriert, dass sie die Validierung des Datenelements 8155 gemäß den oben angeführten ISO 6346-Regeln durchführt.

```
<?xml version="1.0" encoding="UTF-8"?>
<Config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Version="5"
xsi:noNamespaceSchemaLocation="../EDIConfig5.xsd">
   <Codelist>
      <Field Name="F8155" Type="Alphanumeric" ListName="ISO6346" Length="1" Offset="1">
         <Title>ISO 6346</Title>
         <Description>Code specifying seagoing container type.</Description>
         <Values>
            <Value Code="1"/>

            ..............

            <Value Code="N"/>
         </Values>
      </Field>
      <Field Name="F8155" Type="Alphanumeric" ListName="ISO6346" Length="1" Offset="2">
         <Title>ISO 6346</Title>
         <Description>Code specifying seagoing container type.</Description>
         <Values>
            <Value Code="0"/>

            ..............

            <Value Code="F"/>
         </Values>
      </Field>
      <Field Name="F8155" Type="Alphanumeric" ListName="ISO6346" Length="2" Offset="3">
         <Title>ISO 6346</Title>
         <Description>Code specifying seagoing container type.</Description>
         <Values>
            <Value Code="G0"/>

            ..............

            <Value Code="A0"/>
         </Values>
      </Field>
   </Codelist>
</Config>
```

ISO6346.Codelist-Konfigurationsdatei

Anmerkung: Einige der Werte aus dem obigen Beispielcodefragment wurden aus Gründen der Einfachheit entfernt.

Beachten Sie, dass das Element `<Field>` im Codefragment oben drei Mal definiert wurde. Dadurch kann der Feldinhalt auf Zeichenebene validiert werden. Jede Instanz von `<Field>` wird jeweils anhand einer der in der Aufzählung oben angegebenen Regeln validiert. Zu diesem Zweck hat jedes `<Field>`-Element mehrere Attribute (im Codefragment oben markiert):

- `Name` - Dieses Attribut gibt an, dass die Liste der validierten Werte zum selben Feld gehört, d.h. es bindet alle drei Listen an dasselbe Feld (in diesem Fall "F8155").

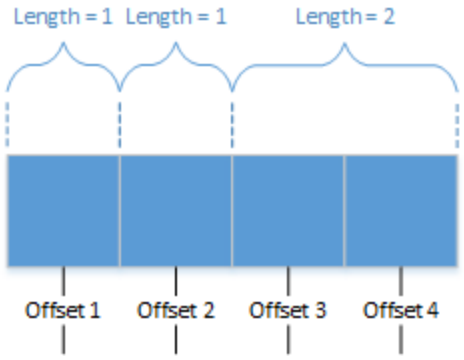
© 2016-2022 Altova GmbH

- `ListName` - Dieses Attribut gibt einer Liste von Werten einen Namen. Wertelisten mit demselben Namen werden in einer Codeliste gruppiert. Beachten Sie, dass alle drei Listen im Beispiel oben denselben Listennamen `ListName="ISO6346"` haben. Dies wird speziell für die Validierung auf Zeichenebene benötigt. Die eigentlichen zu validierenden Zeichen werden, wie unten erläutert, durch die Attribute `Length` und `Offset` der einzelnen Listen in der Gruppe definiert.
- `Length` - Mit diesem Attribut wird die Anzahl der zu validierenden Zeichen definiert (die Anfangsposition von Zeichen im Feld wird durch das Attribut `Offset` definiert). Beachten Sie, dass die Zeichenlänge für jedes `<Field>`-Element unterschiedlich ist: Die erste und die zweite Werteliste ist ein Zeichen lang, während die dritte Liste zwei Zeichen enthält.
- `Offset` - Dieses Attribut definiert die Position, ab der MapForce (oder der externe Ausführungsprozessor) die Zeichen aus dem zu validierenden Datenfeld liest. Das erste Zeichen hat den Abstand (offset) 1, das zweite Zeichen den Abstand 2, usw. Die tatsächlich Anzahl der Zeichen wird durch das Attribut `Length` definiert.

Wenn das Attribut `Length` definiert ist, so muss auch das Attribut `Offset` definiert sein. Außerdem muss der Wert von `Length` und `Offset` größer als Null sein.

Im Codefragment oben sind die folgenden Längen-Abstands- (Length-Offset)-Kombinationen definiert:

- `Length="1"`, `Offset="1"` - mit dieser Kombination wird ein einziges Zeichen des Felds beginnend mit dem Abstand 1 validiert.
- `Length="1"`, `Offset="2"` - mit dieser Kombination wird ein einziges Zeichen des Felds beginnend mit dem Abstand 2 validiert.
- `Length="2"`, `Offset="3"` - mit dieser Kombination werden zwei Zeichen des Felds beginnend mit dem Abstand 3 validiert



Jede Length-Offset-Kombination stellt, kombiniert mit den unter `<Values>` definierten Codewerten eine Validierungsbedingung auf Zeichenebene dar. Bei Ausführung des Mappings müssen alle Bedingungen auf Zeichenebene erfüllt werden (da alle denselben `ListName`-Attributwert haben), damit die Feldvalidierung erfolgreich ist, d.h. MapForce wendet bei der Validierung einer Message, die das Datenelement 8155 enthält, die folgende Logik an:

- Die Feldzeichen werden anhand der in der Konfigurationsdatei (in diesem Fall **ISO6346.Codelist**) definierten Codewertliste anhand einer übereinstimmenden Length-Offset-Kombination validiert. So muss z.B. das erste Zeichen des Felds eines der folgenden sein: 1, 2, 3, 4, B, C, G, H, L, M, N.
- Der Feldwert muss alle Bedingungen auf Zeichenebene erfüllen (in diesem Fall müssen drei Bedingungen erfüllt werden), damit die Validierung erfolgreich ist.

- Wenn eine der Bedingungen nicht erfüllt wird, ist der Feldwert nicht gültig und MapForce (oder der Ausführungsprozessor, der das Mapping verarbeitet, wie z.B. MapForce Server oder ein C#, C++ oder Java-Programm) erzeugt einen Validierungsfehler.
- Ein Feld kann mehr als eine Liste möglicher Codewerte haben. Um als gültig zu gelten, muss der Feldwert in mindestens einer dieser Listen vorhanden sein.

© 2016-2022 Altova GmbH

4.4 CSV- und Textdateien

MapForce unterstützt das Mappen von Daten von und auf textbasierte Dateiformate wie CSV (Komma-Separated Values = kommagetrennte Werte) und FLF- (Fixed-Length Field = Felder mit fester Länge) -Textdateien. Sie können z.B. die folgenden Datentransformationen durchführen:

- XML-Schema auf Flat File-Formate und umgekehrt
- Datenbank auf Flat File-Formate und umgekehrt
- UN/EDIFACT und ANSI X12 von/auf Flat File-Formate oder Datenbanken

FLF ist ein gebräuchliches Textformat, in dem Daten in Felder mit einer festen Länge aufgeteilt sind (So repräsentieren z.B. die ersten 5 Zeichen jeder Zeile eine Transaktions-ID, die nächsten 20 Zeichen enthalten die Transaktionsbeschreibung).

Beachten Sie, dass Ihre Dateien im Fall von CSV-Dateien als Trennzeichen nicht nur Kommas, sondern auch Tabulatorzeichen, Semikola, Leerzeichen oder jeden anderen benutzerdefinierten Wert haben können.

Neben CSV- und FLF-Dateien können Sie mit Hilfe von MapForce FlexText (dieses Modul steht in der MapForce Enterprise Edition zur Verfügung) auch Textdateien mit komplexeren oder benutzerdefinierten Strukturen mappen. Mit Hilfe von FlexText können Sie im Grunde die Struktur Ihrer benutzerdefinierten Textdaten (mittels einer so genannten "FlexText-Vorlage") definieren, um diese auf andere Formate mappen zu können. Nähere Informationen dazu finden Sie unter MapForce FlexText 684.

Das Mappen von Daten von oder auf Textdateien wird in jeder der folgenden Programmiersprachen unterstützt: Java, C#, C++ oder BUILT-IN.

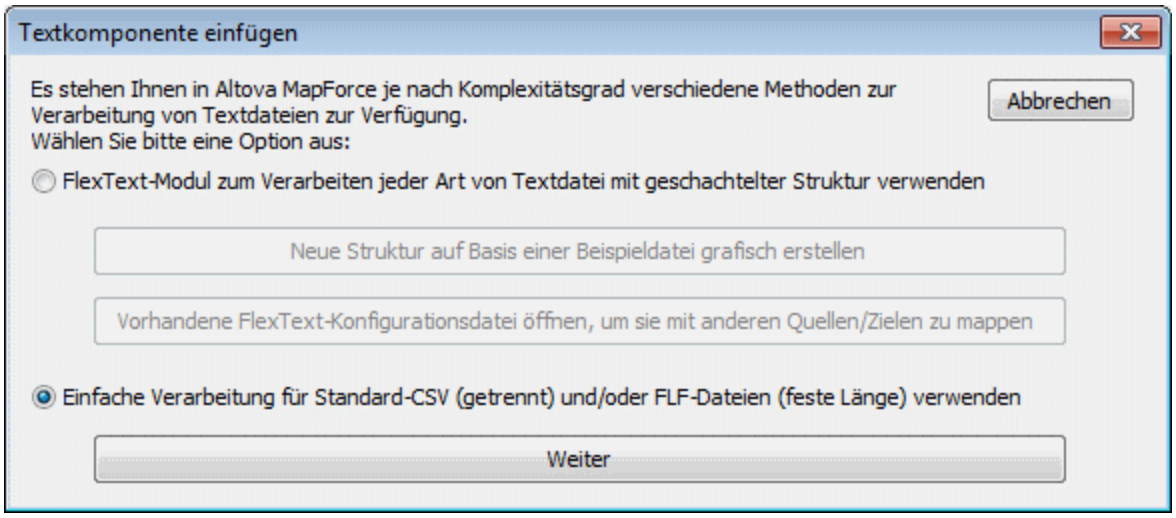
Es gibt zwei Methoden, um gemappte Flat File-Daten zu generieren:

- Durch Klicken auf die Schaltfläche "Ausgabe", woraufhin mit Hilfe des Built-In-Ausführungsprozessors eine Vorschau generiert wird. Oder wählen Sie die Menüoption **Ausgabe | Ausgabedatei speichern** oder klicken Sie auf das Symbol , um das Mapping-Ergebnis zu speichern.
- Durch Auswahl von **Datei | Code generieren in | Java, C# oder C++** und anschließendes Kompilieren und Ausführen des generierten Codes.

4.4.1 Beispiel: Mappen von CSV-Dateien auf XML

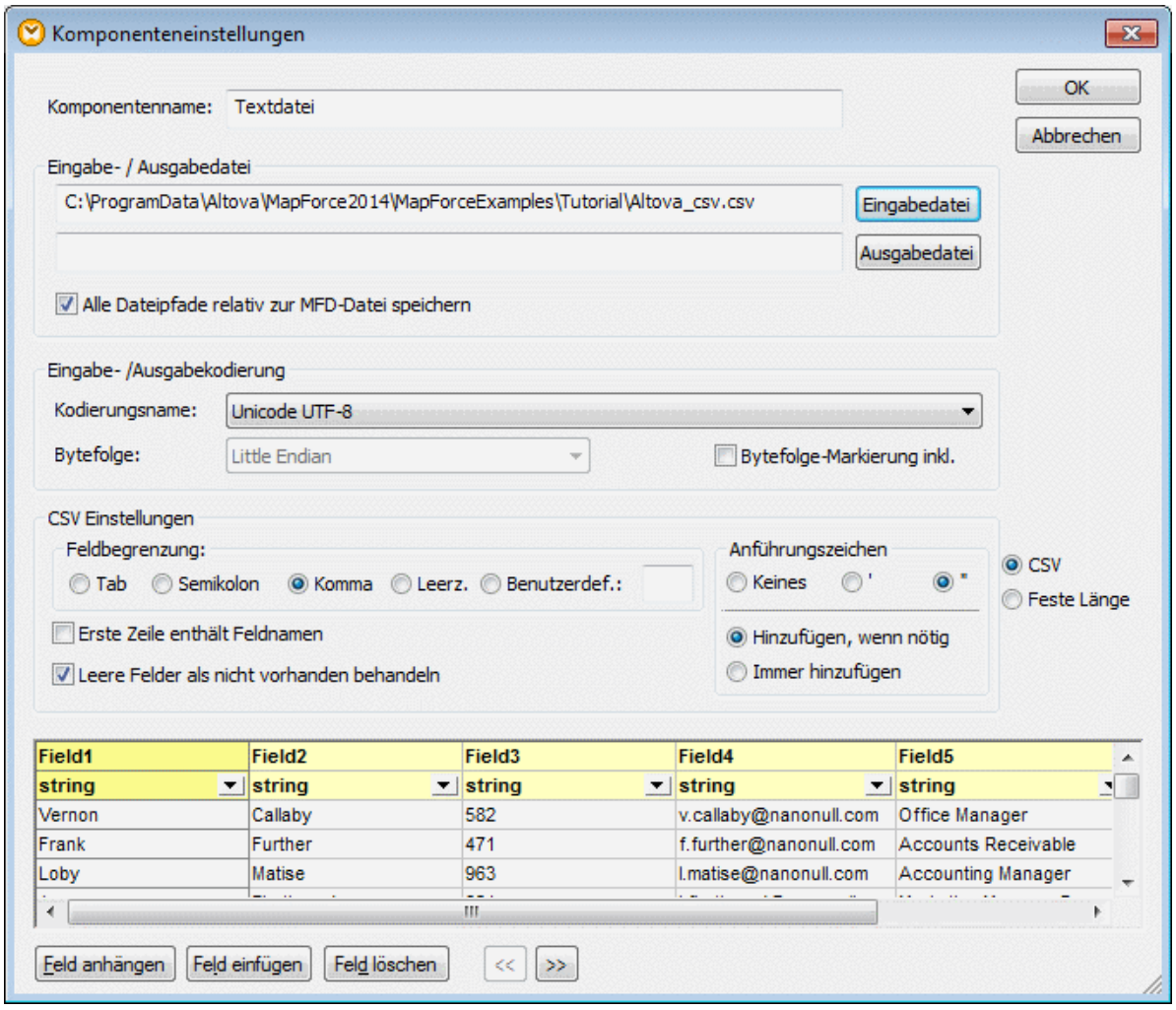
In diesem Beispiel soll ein Mapping erstellt werden, das Daten aus einer einfachen CSV-Datei liest und in eine XML-Datei schreibt. Die in diesem Beispiel verwendeten Dateien befinden sich im Ordner **TUTFOLDER%>.**

1. Wählen Sie als Transformationssprache eine der folgenden Sprachen aus: Java, C#, C++ oder BUILT-IN.
2. Fügen Sie eine Textdateikomponente zum Mapping-Bereich hinzu (Klicken Sie im Menü **Einfügen** auf **Textdatei** oder klicken Sie auf die Symbolleisten-Schaltfläche **Textdatei** einfügen ().

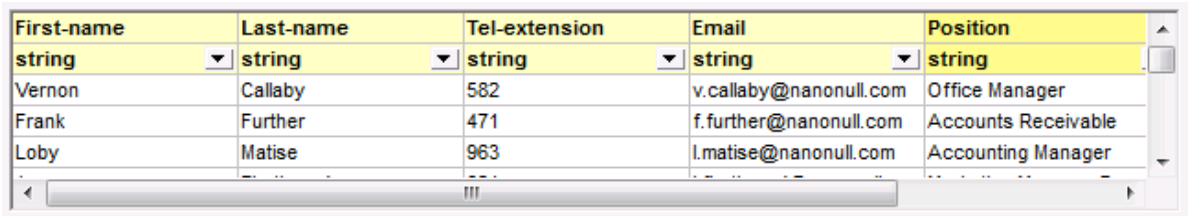


3. Aktivieren Sie das Optionsfeld **Einfache Verarbeitung für** ... und klicken Sie auf die Schaltfläche **Weiter**.
4. Klicken Sie im Dialogfeld "Komponenteneinstellungen" auf **Einbagedatei** und navigieren Sie zur Datei **Altova_csv.csv**. Der Inhalt der Datei wird nun im unteren Bereich des Dialogfelds angezeigt. Beachten Sie, dass nur die ersten 20 Zeilen der Textdatei im Vorschaufenster angezeigt werden.

© 2016-2022 Altova GmbH



5. Klicken Sie in die Spaltenüberschrift **Field1** und ändern Sie den Text in First-name. Ändern Sie auf die gleiche Weise die anderen Überschriften: Field 2 => Last-name, Field 3 =>Tel-extension, Field 4 => Email, Field 5 => Position. TIPP: Durch Drücken der Tabulatortaste können Sie der Reihe nach durch alle Felder navigieren: header1, header2 usw.

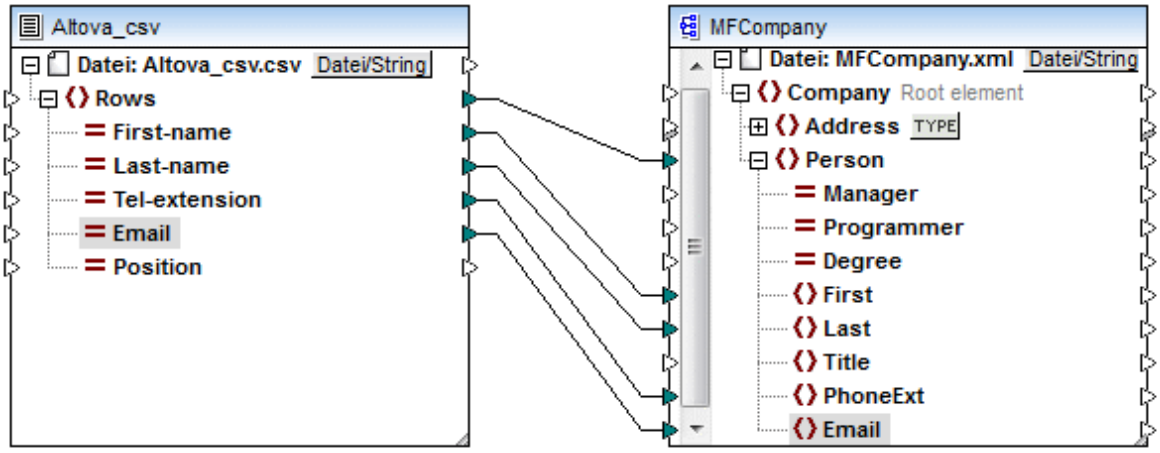


First-name	Last-name	Tel-extension	Email	Position
string	string	string	string	string
Vernon	Callaby	582	v.callaby@nanonull.com	Office Manager
Frank	Further	471	f.further@nanonull.com	Accounts Receivable
Loby	Matise	963	l.matise@nanonull.com	Accounting Manager

6. Klicken Sie auf OK.
7. Wenn Sie aufgefordert werden, den Komponentennamen zu ändern, klicken Sie auf "Komponentennamen ändern". Daraufhin wird die CSV-Komponente im Mapping angezeigt.
8. Fügen Sie **MFCompany.xsd** als XML-Zielkomponente des Mappings hinzu (Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei**).

9. Klicken Sie auf **Überspringen**, wenn Sie aufgefordert werden, eine XML-Beispieldatei anzugeben und wählen Sie `Company` als Root-Element aus.
10. Mappen Sie die Komponentenentsprechungen und stellen Sie dabei sicher, dass das Datenelement **Rows** auf das Datenelement **Person** in der Schema-Zieldatei gemappt wird.

> Der Konnektor vom Datenlement `Rows` in der CSV-Komponente zum Datenelement `Person` im Schema ist von essenzieller Bedeutung, da er festlegt, durch welche Elemente iteriert werden soll. Dadurch wird für jede Zeile in der CSV-Datei ein neues `Person`-Element in der XML-Ausgabedatei erstellt.



11. Klicken Sie auf das Register "Ausgabe", um das Ergebnis zu sehen.

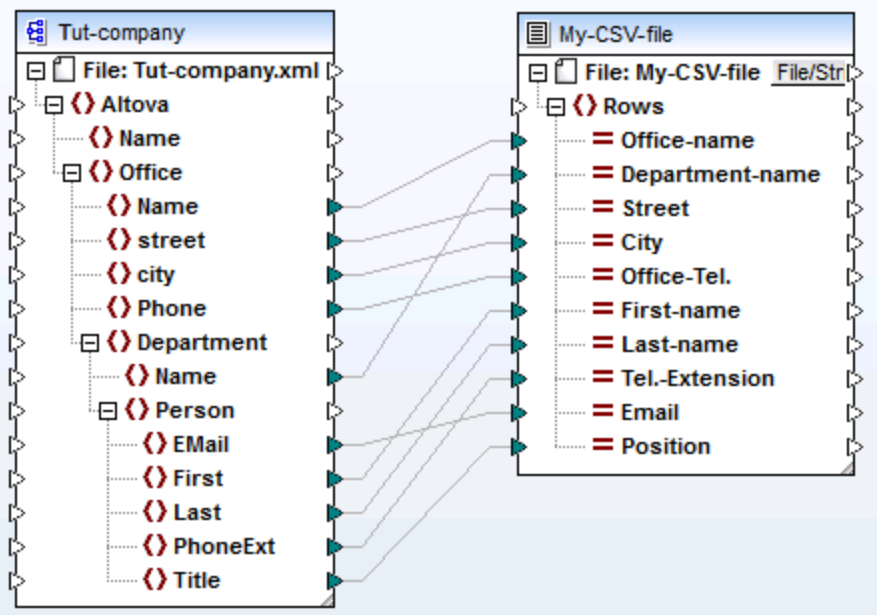
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Company xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
3    <Person Manager="true">
4      <First>Vernon</First>
5      <Last>Callaby</Last>
6      <PhoneExt>582</PhoneExt>
7      <Email>v.callaby@nanonull.com</Email>
8    </Person>
9    <Person Manager="true">
10     <First>Frank</First>
11     <Last>Further</Last>
12     <PhoneExt>471</PhoneExt>
13     <Email>f.further@nanonull.com</Email>
14   </Person>
15   <Person Manager="true">
```

Die Daten aus der CSV-Datei wurden nun erfolgreich auf eine XML-Datei gemappt.

4.4.2 Beispiel: Iterieren durch Datenelemente

In diesem Beispiel wird gezeigt, wie Sie Iterationen (mehrere Zeilen) in einer CSV-Zieldatei erstellen. Die Mapping-Design-Datei zu diesem Beispiel finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\Tut-xml2csv.mfd**.

© 2016-2022 Altova GmbH

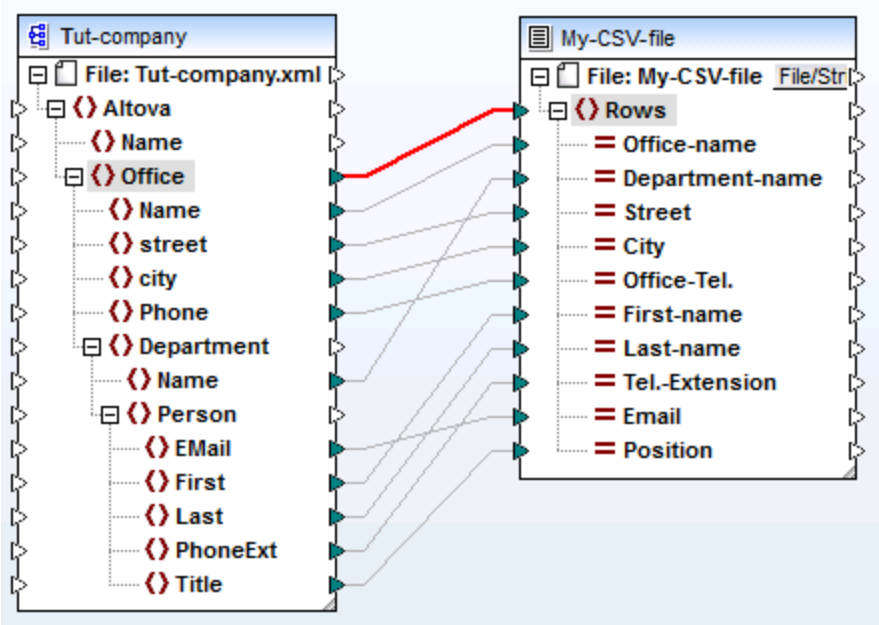


Tut-xml2csv.mfd

Dieses Mapping wurde absichtlich als unvollständiges Mapping erstellt. Wenn Sie versuchen, die Beispieldatei mit dem Menübefehl **Datei | Mapping validieren** zu validieren, erhalten Sie eine Validierungswarnung. Wenn Sie außerdem eine Vorschau der Mapping-Ausgabe anzeigen, wird nur eine einige Zeile erzeugt, was eventuell nicht das beabsichtigte Ergebnis ist.

Angenommen, Sie möchten anhand einer Sequenz von Datenelementen aus der XML-Datei mehrere Zeilen in der CSV-Datei erstellen. Sie können dazu auch eine Verbindung zum Datenelement `Rows` der CSV-Zieldatei ziehen.

Um z.B. durch alle offices (Büros) zu iterieren, sodass die Ausgabe in der CSV-Datei aufscheint, müssen Sie `Office` mit `Rows` verbinden. Dadurch erstellt MapForce für jedes `Office`-Datenelement der XML-Quelldatei eine Zeile (row) in der CSV-Zieldatei.

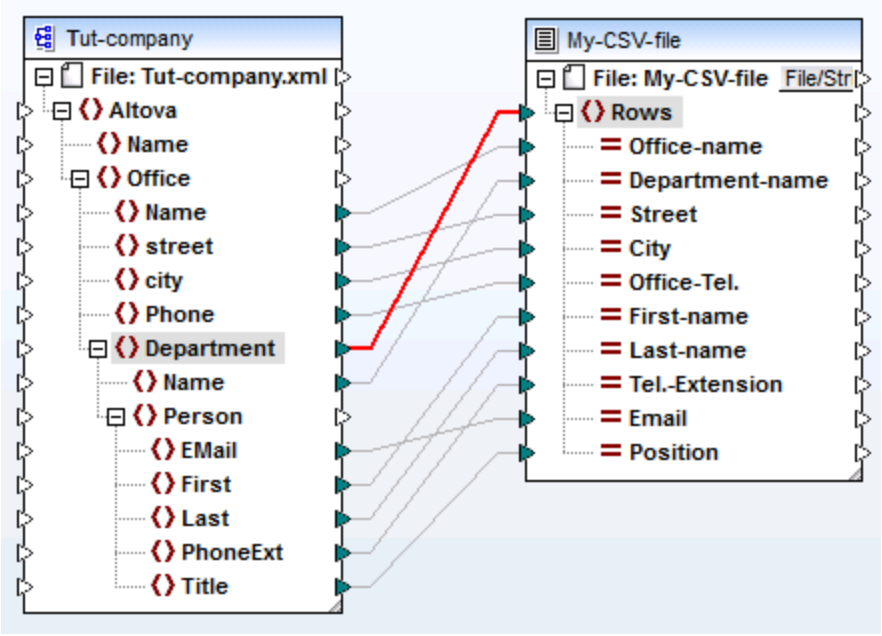


Das Datenelement `Rows` in der CSV-Komponente fungiert als Iterator für die damit verbundene Datenelementsequenz. Wenn Sie daher das Datenelement "Office" verbinden, wird in der Ausgabe für jedes in der XML-Quelldatei gefundene office-Element eine Zeile (row) erstellt.

```
1   "Microtech, Inc.",Level 1 support,Major Ave 1,Vancouver,558833
2   "Microtech Partners, Inc.",Level 2 support,Perro Bvd 1324,Otto
3
```

Ähnlich dazu wird bei Verbindung von **Department** mit dem Datenelement **Rows** für jede Abteilung (department) der XML-Quelldatei eine Zeile erzeugt.

© 2016-2022 Altova GmbH



Die Ausgabe würde folgendermaßen aussehen:

```
1  "Microtech, Inc.",Admin,Major Ave 1,Vancouver,5588339,Clive,Clo
2  "Microtech, Inc.",Sales and Marketing,Major Ave 1,Vancouver,558
3  "Microtech, Inc.",Manufacturing,Major Ave 1,Vancouver,5588339,K
4  "Microtech, Inc.",Level 1 support,Major Ave 1,Vancouver,5588339
5  "Microtech Partners, Inc.",Admin,Perro Bvd 1324,Ottowa,3549202,
6  "Microtech Partners, Inc.",Sales and Marketing,Perro Bvd 1324,O
7  "Microtech Partners, Inc.",Level 2 support,Perro Bvd 1324,Ottow
8
```

Wenn Sie schließlich `Person` auf das Datenelement `Rows` mappen, werden alle Personen (persons) ausgegeben. In diesem Fall iteriert MapForce folgendermaßen durch die Datensätze: jede Person in jeder Abteilung (department) in jedem Büro (Office).

4.4.3 Beispiel: Erstellen von Hierarchien anhand von CSV- und FLF-Dateien

Sie finden dieses Beispiel im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** unter **Tut-headerDetail.mfd**. In diesem Beispiel wurde eine CSV-Datei (Orders.csv) mit dem folgenden Format verwendet:

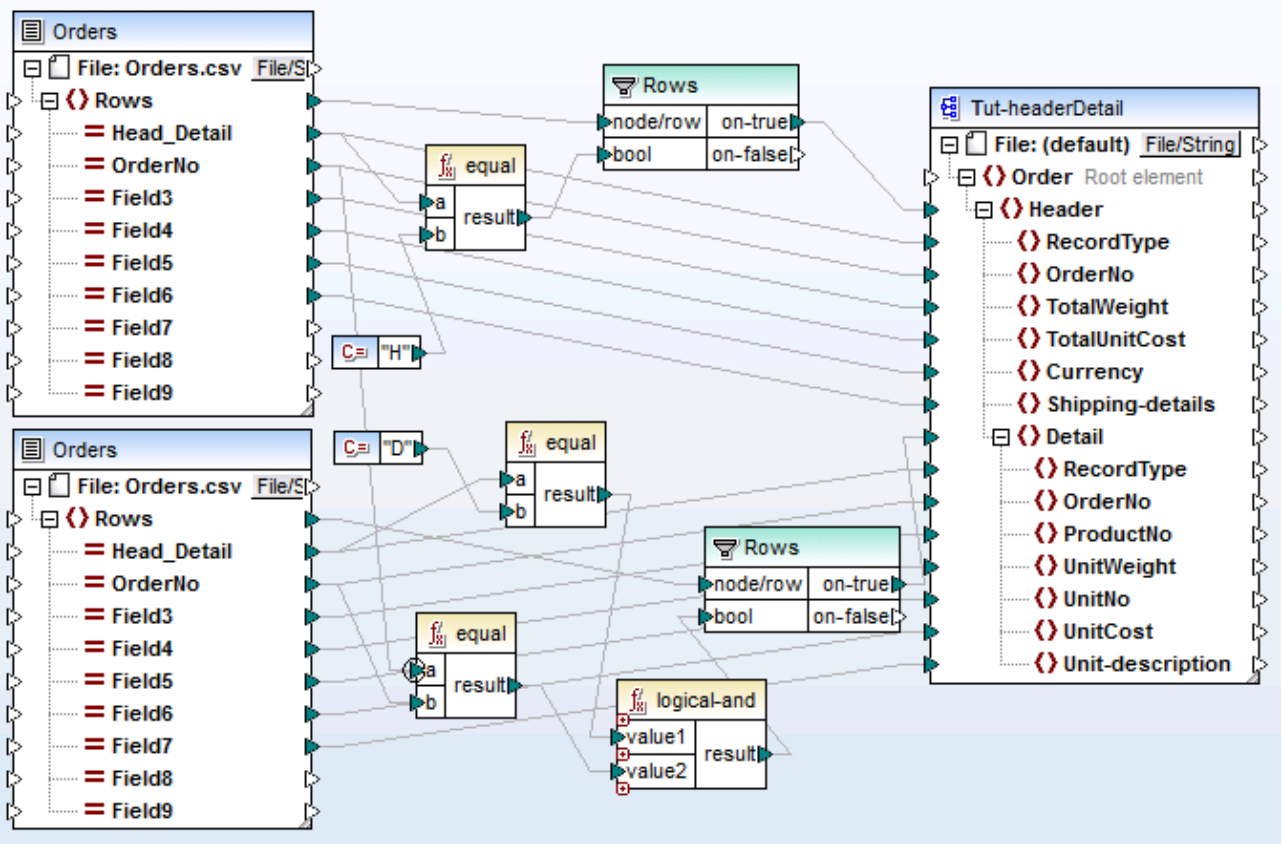
- Field 1: H definiert einen Header-Datensatz und D einen Detail-Datensatz.
- Field 2: Ein gemeinsamer Schlüssel sowohl für Header- als auch Detail-Datensätze.
- Jeder Header- oder Detail-Datensatz befindet sich in einer eigenen Zeile.

Nachfolgend sehen Sie den Inhalt der Datei Orders.csv.

```
H,111,332.1,22537.7,,Container ship,,,
D,111,A-1579-227,10,3,400,Microtome,,
D,111,B-152-427,7,6,1200,Miscellaneous,,
H,222,978.4,7563.1,,Air freight,,,
D,222,ZZ-AW56-1,10,5,10000,Gas Chromatograph,,
```

Ziel dieses Mapping ist:

- die Flat File CSV-Datei auf eine hierarchische XML-Datei zu mappen
- die mit einem H gekennzeichneten Header-Datensätze zu filtern
- die entsprechenden Detail-Datensätze, die mit einem D gekennzeichnet sind, mit den einzelnen Header-Datensätzen zu verknüpfen.



tut-headerDetail.mfd

Damit dies möglich ist, müssen die Header- und Detail-Datensätze ein gemeinsames Feld haben. In diesem Fall handelt es sich beim gemeinsamen Feld/Schlüssel um das zweite Feld in der CSV-Datei, nämlich **OrderNo**. In der CSV-Datei enthalten sowohl der erste Header-Datensatz als auch die folgenden beiden Detail-Datensätze den gemeinsamen Wert 111.

Die Datei Orders.csv wurde zweimal eingefügt, um das Mapping intuitiver zu machen.

Die Schema-Datei **Tut-headerDetail.xsd** hat eine hierarchische Struktur: "Order" ist das Root-Element, das das Child-Element "Header" enthält, das wiederum das Child-Element "Detail" enthält.

© 2016-2022 Altova GmbH

Die erste Orders.csv-Datei liefert die **Header**-Datensätze (und alle gemappten Felder) für das Header-Datenelement in der Schema-Zieldatei. Mit Hilfe der filter-Komponente werden alle Datensätze mit Ausnahme derer, die mit H beginnen herausgefiltert. Das **Rows**-Datenelement liefert die gefilterten Datensätze für das Header-Datenelement in der Schema-Datei.

Die zweite Orders.csv-Datei liefert die **Detail**-Datensätze (und alle gemappten Felder), indem die Detail-Datensätze, die mit dem OrderNo-Schlüssel des Header-Datensatzes übereinstimmen, herausgefiltert werden. Dies geschieht folgendermaßen:

- Das **OrderNo**-Feld des Header-Datensatzes wird mit Hilfe der **equal**-Funktion (der Prioritätskontext[1344] wird aus Gründen der Effizienz für den Parameter **a** festgelegt) mit demselben Feld der Details-Datensätze verglichen.
- Die **Logical-and**-Funktion wird verwendet, um nur diejenigen Detail-Datensätze weiterzugeben, die dasselbe OrderNo-Feld enthalten, wie der Header-Datensatz.

Über den on-true-Parameter der Filter-Komponente liefert das Datenelement **Rows** diese gefilterten Datensätze an die Datenelemente "Header" und "Detail" in der Schema-Datei.

Wenn Sie auf die Schaltfläche "Ausgabe" klicken, wird die unten gezeigte XML-Datei erzeugt. Jeder Header-Datensatz enthält die dazugehörigen Daten sowie alle damit verknüpften Detail-Datensätze mit derselben Bestellnummer (OderNo).

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation
3    <Header>
4      <RecordType>H</RecordType>
5      <OrderNo>111</OrderNo>
6      <TotalWeight>332.1</TotalWeight>
7      <TotalUnitCost>22537.7</TotalUnitCost>
8      <Currency/>
9      <Shipping-details>Container ship</Shipping-details>
10     <Detail>
11       <RecordType>D</RecordType>
12       <OrderNo>111</OrderNo>
13       <ProductNo>A-1579-227</ProductNo>
14       <UnitWeight>10</UnitWeight>
15       <UnitNo>3</UnitNo>
16       <UnitCost>400</UnitCost>
17       <Unit-description>Microtome</Unit-description>
18     </Detail>
19     <Detail>
20       <RecordType>D</RecordType>
21       <OrderNo>111</OrderNo>
22       <ProductNo>B-152-427</ProductNo>
23       <UnitWeight>7</UnitWeight>
24       <UnitNo>6</UnitNo>
25       <UnitCost>1200</UnitCost>
26       <Unit-description>Miscellaneous</Unit-description>
27     </Detail>
28   </Header>
```

Werfen wir kurz einen Blick auf eine andere Beispieldatei, in der eine etwas andere CSV-Datei verwendet wird. Die Datei befindet sich unter dem Namen **Head-detail-inline.mfd** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\.** Dies sind die Unterschiede:

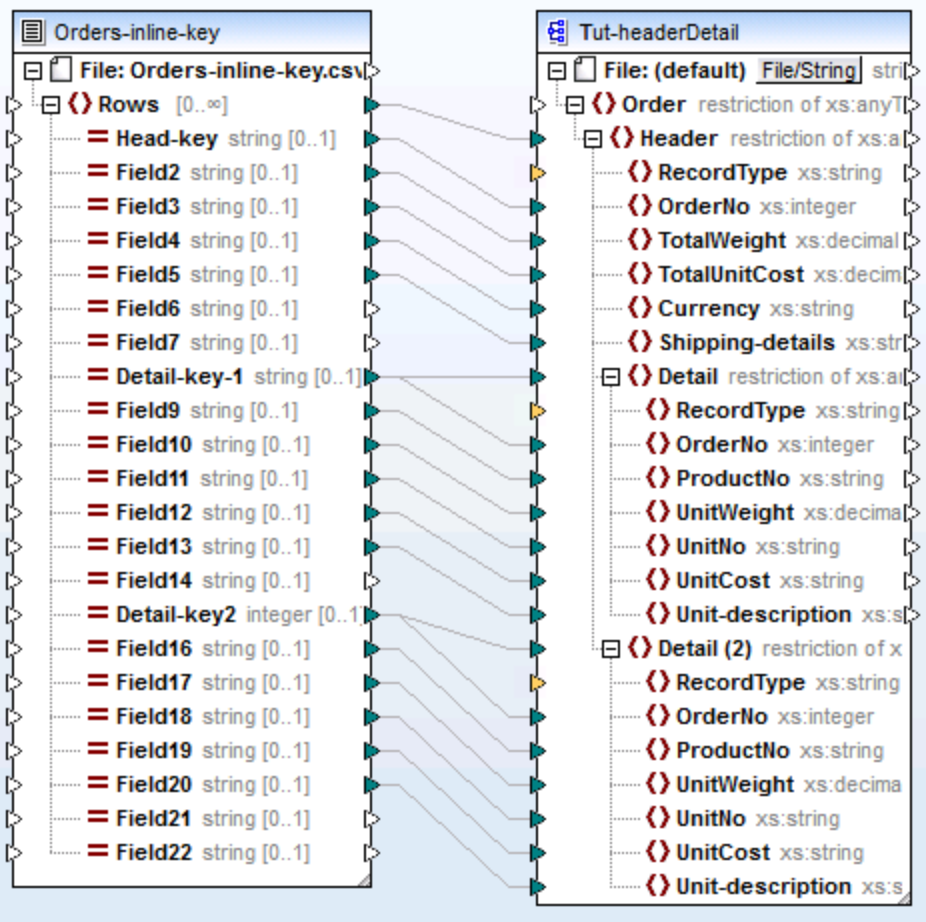
- Es ist hier keine Datensatzkennung (H, or D) vorhanden

- Es gibt jedoch weiterhin ein gemeinsames Schlüsselfeld für den Header- und die Detail-Datensätze (Head-key, Detail-key...), nämlich das erste Feld in der CSV-Datei. Das Feld wird auf OrderNo im Zielschema gemappt.
- "Header" und alle entsprechenden "Detail"-Felder befinden sich alle in derselben Zeile.

```
111,332.1,22537.7,,Container ship,,,111,A-1579-227,10,3,400,Microtome,,111,B-15
222,978.4,7563.1,,Air freight,,,222,ZZ-AW56-1,10,5,10000,Gas Chromatograph,,
```

Das Mapping wurde folgendermaßen erstellt:

- Die Schlüsselfelder werden auf die entsprechenden OrderNo-Datenelemente im Zielschema gemappt.
- Das Datenelement "Detail" in der Schema-Zieldatei wurde dupliziert und wird als **Detail (2)** angezeigt, damit Sie den zweiten Satz von Detail-Datensätzen auf das richtige Datenelement mappen können.
- Als Ergebnis dieses Mappings erhalten Sie in etwa dieselbe XML-Datei, wie im ersten Beispiel.



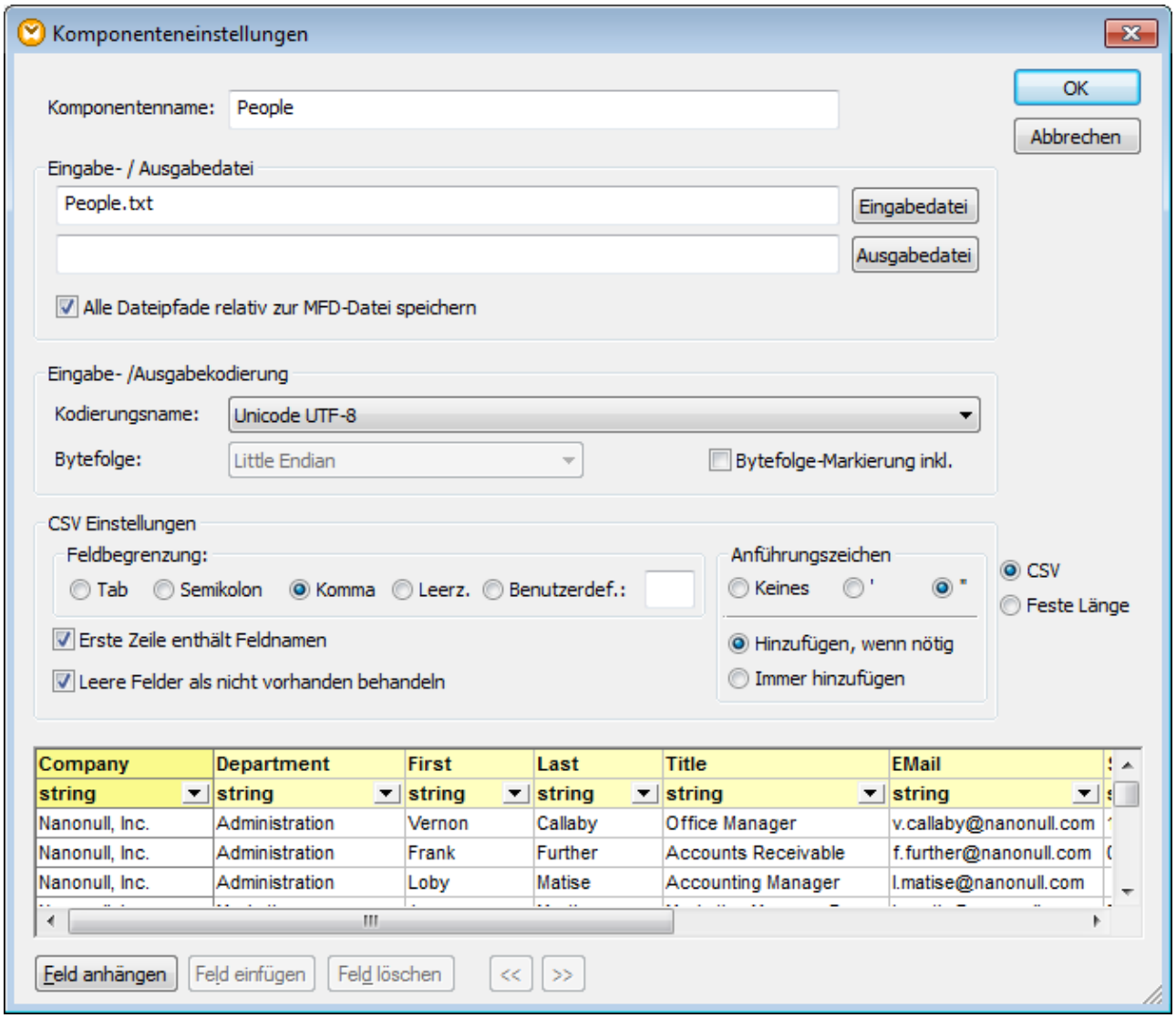
Head-detail-inline.mfd

© 2016-2022 Altova GmbH

4.4.4 Definieren der CSV-Optionen

Nachdem Sie eine Textkomponente zum Mapping-Bereich hinzugefügt haben, können Sie die Einstellungen dafür im Dialogfeld "Komponenteneinstellungen" definieren. Sie können das Dialogfeld "Komponenteneinstellungen" auf eine der folgenden Arten öffnen:

- Wählen Sie die Komponente aus und klicken Sie im Menü **Komponente** auf **Eigenschaften**.
- Doppelklicken Sie auf die Komponentenüberschrift.
- Klicken Sie mit der rechten Maustaste auf die Komponentenüberschrift und wählen Sie den Befehl **Eigenschaften**.

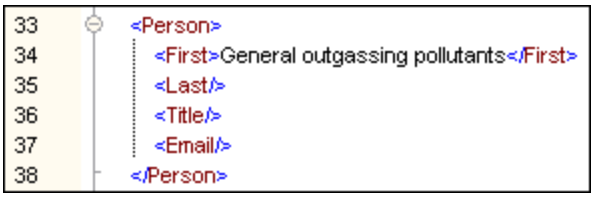
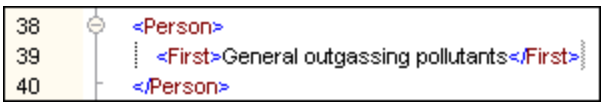


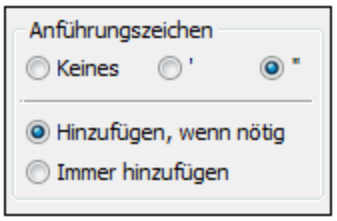
Dialogfeld "Komponenteneinstellungen" für eine Textkomponente (im CSV-Modus)

Es stehen die folgenden Einstellungen zur Verfügung.

Komponentenname	Der Komponentenname wird bei der Erstellung der Komponente automatisch generiert. Sie können den Namen jedoch jederzeit ändern. Der Komponentenname kann Leerzeichen und Punkte enthalten, darf aber keine Schrägstriche, umgekehrten Schrägstriche, Doppelpunkte, doppelten Anführungszeichen und voran- oder nachgestellte Leerzeichen enthalten. Beachten Sie beim Ändern des Namens einer Komponente Folgendes: • Wenn Sie das Mapping auf FlowForce Server bereitstellen möchten, muss der Komponentenname eindeutig sein. • Es wird empfohlen, nur Zeichen zu verwenden, die über die Befehlszeile eingegeben werden können. Länderspezifische Sonderzeichen sind in Windows und der Befehlszeile eventuell unterschiedlich kodiert.
Eingabedatei	Definiert die Datei, aus der MapForce die Daten ausliest. Dieses Feld ist bei einer Quellkomponente von Bedeutung und wird ausgefüllt, wenn Sie die Komponente erstellen und ihr eine Textdatei zuweisen. Das Feld kann leer bleiben, wenn Sie die Textdateikomponente als Ziel für Ihr Mapping verwenden. Anhand des Werts in diesem Feld werden bei einer Quellkomponente Spaltennamen gelesen. Außerdem wird anhand dieses Werts eine Vorschau des Inhalts der Instanztextdatei angezeigt. Um eine neue Datei auszuwählen, klicken Sie auf **Eingabedatei**.
Ausgabedatei	Definiert die Datei, in die MapForce die Daten schreibt. Dieses Feld hat bei einer Zielkomponente eine Bedeutung. Um eine neue Datei auszuwählen, klicken Sie auf **Ausgabedatei**.
Alle Dateipfade relativ zur MFD-Datei speichern	Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld "Komponenteneinstellungen" angezeigten Dateipfade relativ zum Ordner, in dem sich die MapForce Design (.mfd)-Datei befindet. Diese Einstellung wirkt sich auf die von der Textkomponente verwendeten Eingabe- und Ausgabedateien aus. Siehe auch Verwenden relativer Pfade in einer Komponente 85.
Eingabe- / Ausgabekodierung	Damit können Sie die folgenden Einstellungen der Output-Instanzdatei definieren: • Kodierungsname • Bytefolge • Ob das Bytefolgemarkierungszeichen (BOM) inkludiert werden soll. Standardmäßig haben alle neuen Komponenten die in der Option **Standardkodierung für neue Komponenten** definierte Kodierung. Sie können diese Option über **Extras | Optionen,** Register "Allgemein" aufrufen.

© 2016-2022 Altova GmbH

Feldbegrenzung	In CSV-Dateien werden standardmäßig Kommas "," als Trennzeichen verwendet. Mit Hilfe dieser Option können Sie die Zeichen **Tab**, **Semikolon** oder **Leerzeichen** als Trennzeichen verwenden. In das Feld **Benutzerdef.** können Sie ein benutzerdefiniertes Trennzeichen eingeben.
Erste Zeile enthält Feldnamen	Wählen Sie diese Option aus, damit MapForce die Werte im ersten Datensatz der Textdatei als Spaltenüberschriften verwendet. Die Spaltenüberschriften werden im Mapping dann als Namen der Datenelemente angezeigt.
Leere Felder als nicht vorhanden behandeln	Wenn diese Option aktiviert ist, werden in der Zieldatei keine entsprechenden leeren Datenelemente (Elemente oder Attribute) anhand leerer Felder in der Quelldatei erzeugt. So besteht z.B. der CSV-Datensatz "`General outgassing pollutants,,,,`" aus vier Feldern, von denen die letzten drei leer sind. Angenommen, es handelt sich bei der Ausgabedatei um eine XML-Datei, so werden die leeren Felder (in diesem Beispiel die Elemente `Last`, `Title` und `Email`) in der Ausgabe mit einem leeren Wert erstellt, wenn diese Option deaktiviert ist. :  Wenn diese Option aktiviert ist, so werden die leeren Felder in der Ausgabe nicht erstellt: 
Anführungszeichen	Wenn Ihr Eingabefeld rund um die Feldwerte Anführungszeichen enthält, wählen Sie die Art des in der Quelldatei vorhandenen Anführungszeichens aus. Dieselbe Einstellung wird auch für die Ausgabedateien verwendet.



Sie können für Ausgabedateien zusätzliche Einstellungen definieren:

Hinzufügen, wenn nötig	Fügt das ausgewählte Anführungszeichen nur den Feldern hinzu, in denen der Text das Feldbegrenzungszeichen oder Zeilenumbrüche enthält.
Immer hinzufügen	Fügt das ausgewählte Anführungszeichen zu a Feldern der generierten CSV-Datei hinzu.

CSV / Feste Länge

Ändert den Komponententyp entweder in CSV oder FLF (Felder mit fester Länge).

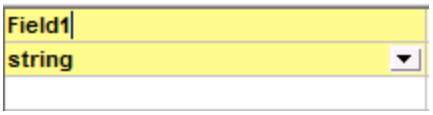
Vorschaubereich

Im unteren Teil des Dialogfelds wird eine Vorschau von bis zu 20 Zeilen der als Ein- oder Ausgabedatei ausgewählten Datei angezeigt.

Falls nötig, können Sie die Struktur der Datei folgendermaßen erstellen (bzw. die Struktur einer vorhandenen entsprechend ändern):

Feld anhängen	Erstellt nach dem letzten CSV-Datensatz ein neues Feld.
Feld einfügen	Erstellt unmittelbar vor dem aktuell ausgewählt CSV-Datensatz ein neues Feld.
Feld löschen	Löscht das aktuell ausgewählte Feld.
<<	Verschiebt das aktuell ausgewählte Feld um e Position nach links.
>>	Verschiebt das aktuell ausgewählte Feld um e Position nach rechts.

Um den Namen eines Felds zu ändern, klicken Sie auf die Überschrift (z.B. **Field1**) und geben Sie den neuen Wert ein. Beachten Sie, dass die Feldnamen nicht editierbar sind, wenn die Option **Erste Zeile enthält Feldnamen** aktiv ist.



© 2016-2022 Altova GmbH

	Um den Datentyp eines Felds zu ändern, wählen Sie den gewünschten Wert aus der Dropdown-Liste aus. MapForce überprüft den Datentyp. Wenn die Input-Daten daher nicht mit dem Feldformat übereinstimmen, so werden die Daten rot markiert.  Die Feldtypen basieren auf den XML-Standard-Schemadatentypen. Der Typ Datum hat z.B. den Form **JJJJ-MM-TT**.

4.4.5 FLF auf Datenbank

In diesem Beispiel erfahren Sie, wie Sie Daten aus einer Textdatei mit Feldern fester Länger (FLF-Datei) auf eine Microsoft Access-Datenbank mappen. Die in diesem Beispiel verwendeten Dateien stehen im Ordner **`Tutorial`** zur Verfügung. Sowohl in der Text-Quelldatei als auch in der Zieldatenbank ist eine Liste von Mitarbeitern gespeichert. Die Datensätze in der Quelldatei sind durch Ihre Größe getrennt:

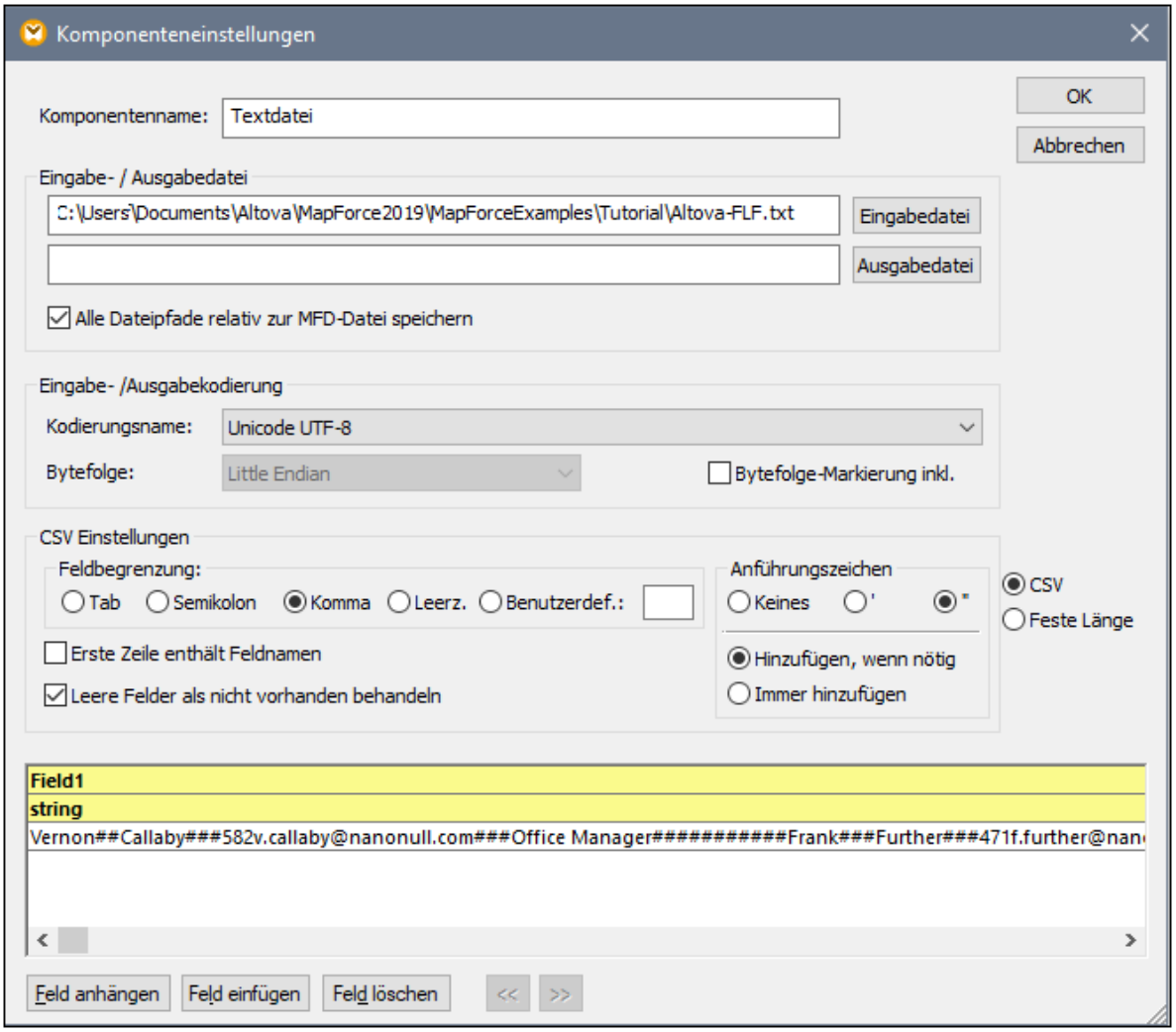
Feldposition und -name	Größe (in Zeichen)
Field 1 (First name)	8
Field 2 (Last name)	10
Field 3 (Phone extension)	3
Field 4 (Email)	25
Field 5 (Position)	25

In diesem Mapping sollen die FLF-Daten auf die Datenbankkomponente gemappt werden. Außerdem soll zu jeder Durchwahl ein neues Präfix hinzugefügt werden. Gehen Sie dazu folgendermaßen vor:

Schritt 1: Einfügen einer Textkomponente

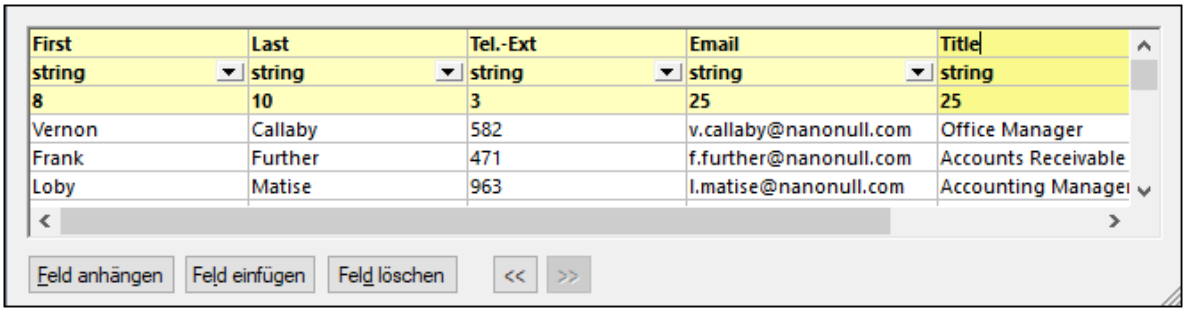
Fügen Sie zuerst eine Textkomponente hinzu und konfigurieren Sie sie. Gehen Sie folgendermaßen vor:

1. Wählen Sie den Menübefehl **Einfügen | Textdatei** oder klicken Sie auf die Symbolleisten-Schaltfläche  (**Textdatei einfügen**). Daraufhin wird das Dialogfeld **Textkomponente einfügen** geöffnet. Wählen Sie die Option **Einfache Verarbeitung....verwenden** und klicken Sie auf **Weiter**.
2. Klicken Sie im Dialogfeld **Komponenteneinstellungen** auf die Schaltfläche **Eingabedatei** (*siehe unten*) und wählen Sie die Datei **`Altova-FLF.txt`** aus.



3. Aktivieren Sie die Option **Feste Länge**.
4. Deaktivieren Sie das Kontrollkästchen **Datensatz-Trennzeichen als vorhanden annehmen**.
5. Die gelb markierten Zeilen können bearbeitet werden. Sie können hier i) den Feldnamen, ii) den Datentyp und iii) die Feldgröße angeben. Geben als Feldgröße 8 ein (dritte gelbe Zeile von oben) und drücken Sie die **Eingabetaste**. In der ersten Spalte, die nun 8 Zeichen breit ist, sind nun mehr Daten zu sehen.
6. Klicken Sie auf **Feld anhängen**, um ein neues Feld hinzuzufügen und definieren Sie als Länge für das zweite Feld 10 Zeichen.
7. Erstellen Sie auf diese Art und Weise drei weitere Felder mit der folgenden Länge: 3, 25 und 25 Zeichen und ändern Sie die Feldüberschriften, wie in der Abbildung unten gezeigt.

© 2016-2022 Altova GmbH

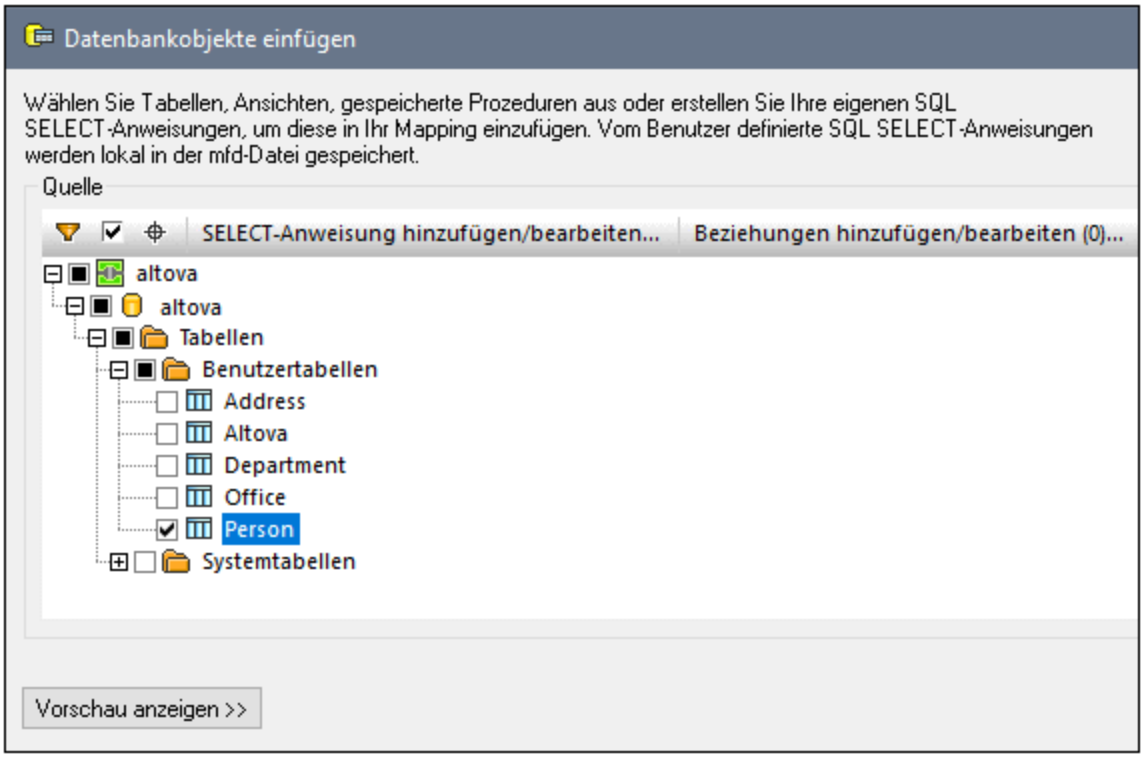


8. Wählen Sie in der Gruppe **Felder mit fester Länge - Einstellungen** die Einstellung **Benutzerdef.** aus und geben Sie die Raute (#) als Zeichen ein. Dadurch behandelt MapForce das #-Zeichen als Füllzeichen.
9. Klicken Sie auf **OK.**
10. Sie werden gefragt, ob der Komponentenname gemäß den Instanzdateien geändert werden soll. Klicken Sie auf **Komponentennamen ändern**. Daraufhin wird die Komponente `Altova-FLF` im Mapping-Fenster angezeigt.

Schritt 2: Einfügen einer Datenbankkomponente

Im nächsten Schritt wird nun eine Datenbankkomponente hinzugefügt. Gehen Sie folgendermaßen vor:

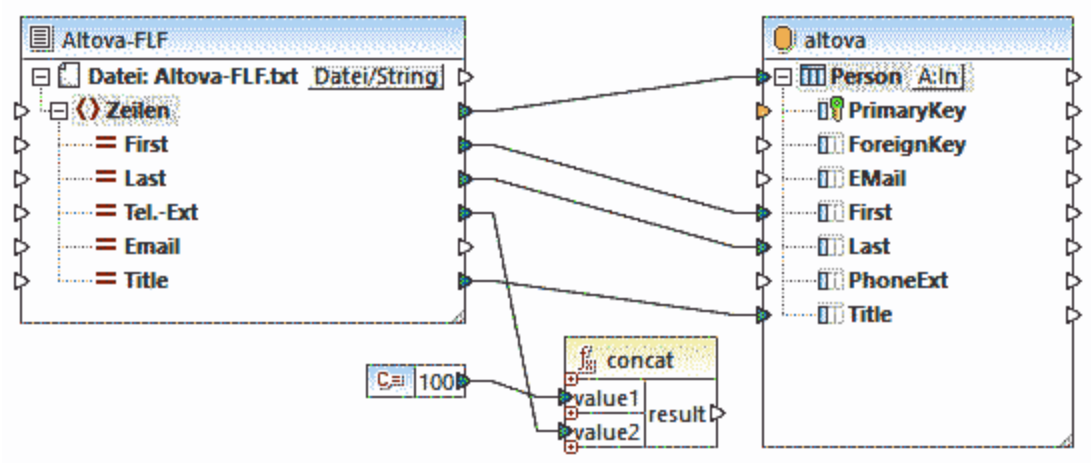
1. Wählen Sie den Menübefehl **Einfügen | Datenbank**, wählen Sie **Microsoft Access** aus und klicken Sie auf **Weiter**.
2. Wählen Sie die Datenbank **`altova.mdb`** aus und klicken Sie auf **Verbinden**.
3. Wählen Sie die Tabelle **Person** aus (*siehe unten*) und klicken Sie auf **OK**.



Schritt 3: Erstellen des Mappings

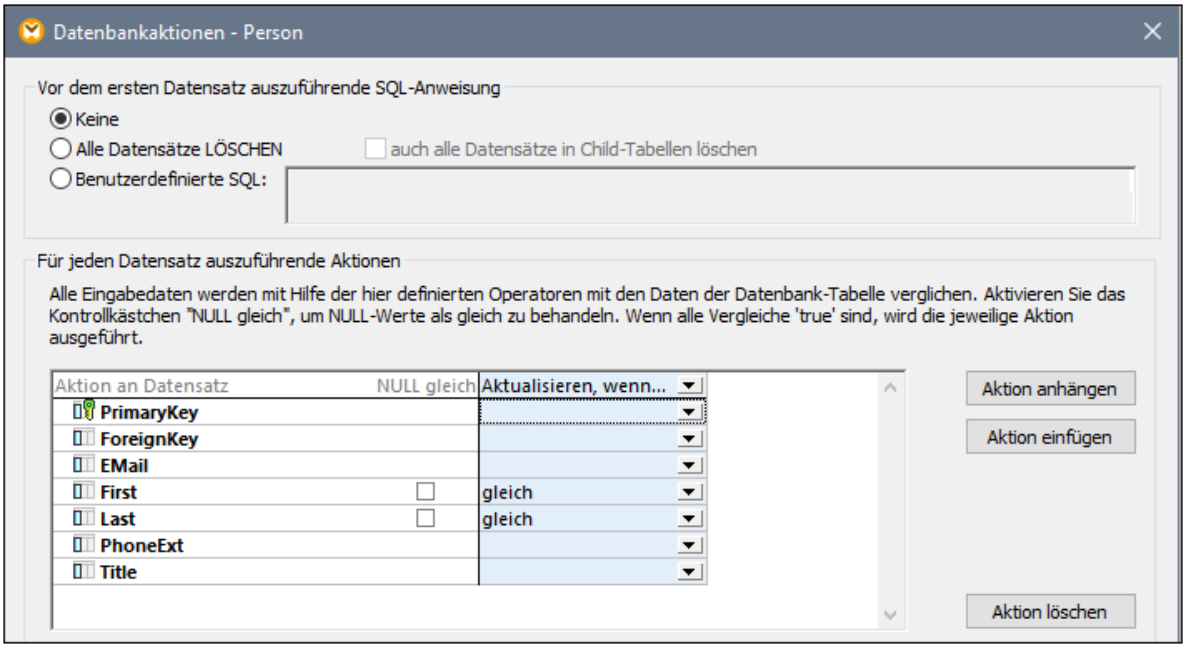
Als nächstes werden wir nun ein Mapping erstellen:

1. Ziehen Sie die **core | concat**[1105]-Funktion aus dem Fenster **Bibliotheken** in das Mapping.
2. Wählen Sie den Menübefehl **Einfügen | Konstante**, wählen Sie als Typ *Zahl* aus und gebe Sie in das Textfeld den Wert "100" ein. In dieser Konstante wird das Präfix für die neue Telefondurchwahl gespeichert.
3. Erstellen Sie die Verbindungen wie oben gezeigt.



4. Klicken Sie in der Datenbankkomponente auf die **Tabellenaktionsschaltfläche** neben **Person**. Daraufhin wird das Dialogfeld **Datenbankaktionen** (*siehe Abbildung unten*) angezeigt.
5. Wählen Sie neben *Aktion an Datensatz*, die Option **Aktualisieren, wenn**. Setzen Sie für die Felder **First** und **Last** die Aktion `gleich`. Klicken Sie auf **OK.** Dadurch aktualisiert MapForce die Tabelle `Person` nur, wenn der Vor- und Nachname (First und Last) in der Quelldatei mit dem entsprechenden Datenbankfeld übereinstimmt. Wenn diese Bedingung zutrifft, wird der Telefondurchwahl die Zahl 100 vorangestellt und in das Feld `PhoneExt` der Tabelle `Person` kopiert.

© 2016-2022 Altova GmbH

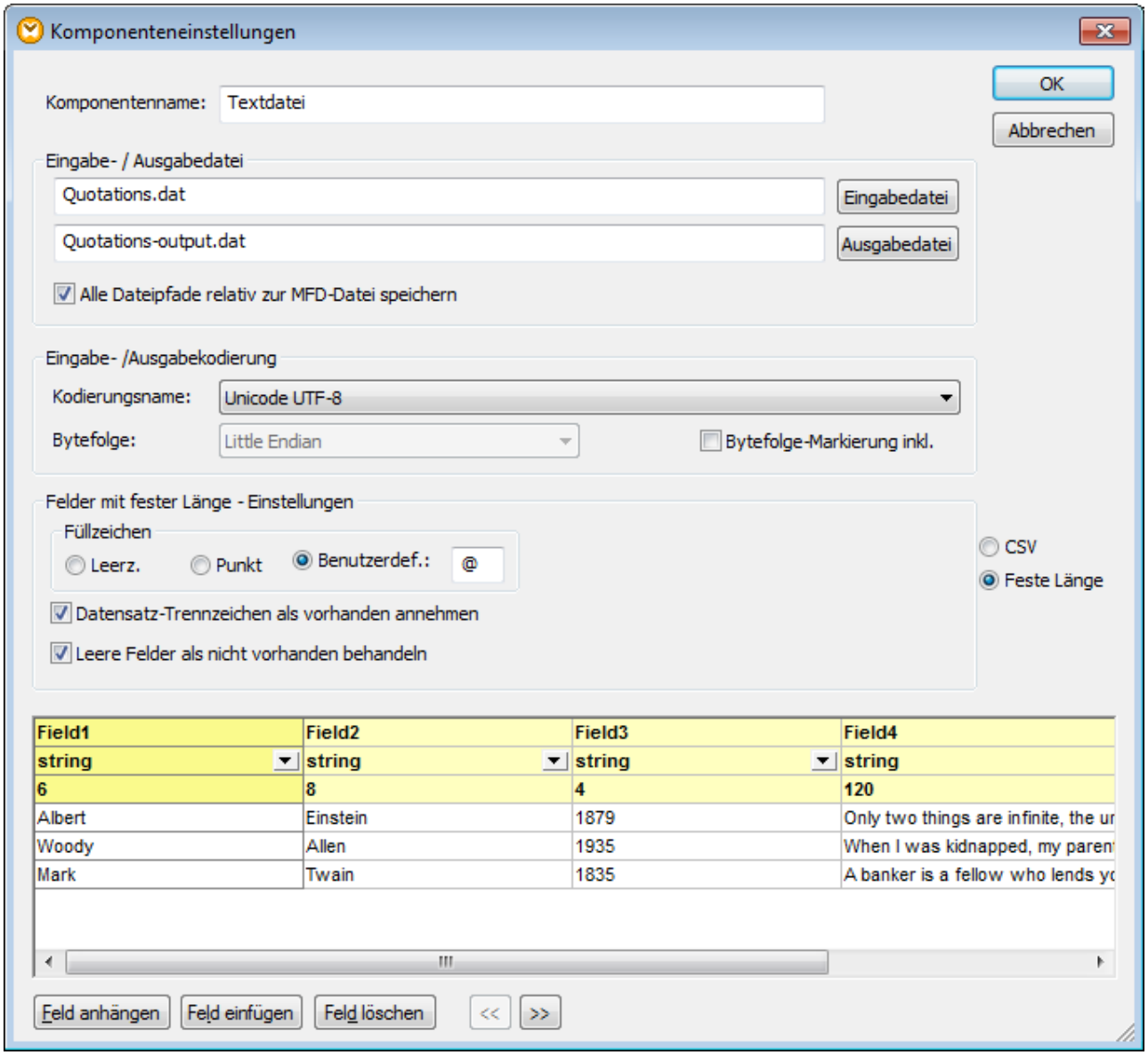


6. Um die SQL-Anweisung (für die Vorschau in MapForce) zu generieren, klicken Sie auf das Register **Ausgabe**. Um die SQL-Anweisungen an der Datenbank auszuführen, klicken Sie auf die Schaltfläche **SQL-Script ausführen** .

4.4.6 Definieren der FLF-Optionen

Nachdem Sie eine Textkomponente zum Mapping-Bereich hinzugefügt haben, können Sie die Einstellungen dafür im Dialogfeld "Komponenteneinstellungen" definieren. Sie können das Dialogfeld "Komponenteneinstellungen" auf eine der folgenden Arten öffnen:

- Wählen Sie die Komponente aus und klicken Sie im Menü **Komponente** auf **Eigenschaften**.
- Doppelklicken Sie auf die Komponentenüberschrift.
- Klicken Sie mit der rechten Maustaste auf die Komponentenüberschrift und wählen Sie den Befehl **Eigenschaften**.



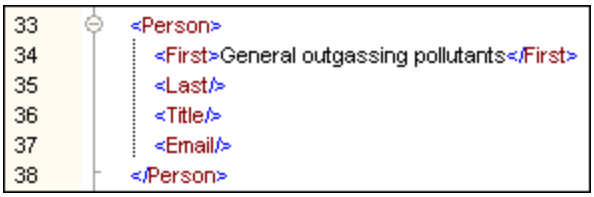
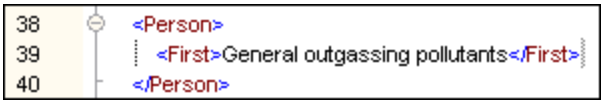
Dialogfeld "Komponenteneinstellungen" für eine Textkomponente (im Modus "feste Länge")

Es stehen die folgenden Einstellungen zur Verfügung.

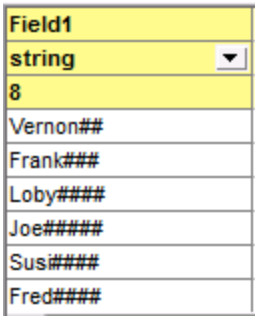
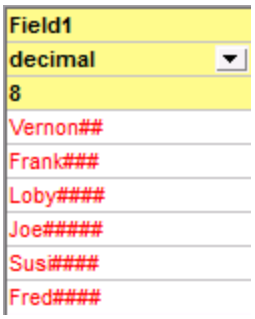
Komponentenname	Der Komponentenname wird bei der Erstellung der Komponente automatisch generiert. Sie können den Namen jedoch jederzeit ändern. Der Komponentenname kann Leerzeichen und Punkte enthalten, darf aber keine Schrägstriche, umgekehrten Schrägstriche, Doppelpunkte, doppelten Anführungszeichen und voran- oder nachgestellte Leerzeichen enthalten. Beachten Sie beim Ändern des Namens einer Komponente Folgendes: • Wenn Sie das Mapping auf FlowForce Server bereitstellen

© 2016-2022 Altova GmbH

	möchten, muss der Komponentenname eindeutig sein. • Es wird empfohlen, nur Zeichen zu verwenden, die über die Befehlszeile eingegeben werden können. Länderspezifische Sonderzeichen sind in Windows und der Befehlszeile eventuell unterschiedlich kodiert.
Eingabedatei	Definiert die Datei, aus der MapForce die Daten ausliest. Dieses Feld ist bei einer Quellkomponente von Bedeutung und wird ausgefüllt, wenn Sie die Komponente erstellen und ihr eine Textdatei zuweisen. Das Feld kann leer bleiben, wenn Sie die Textdateikomponente als Ziel für Ihr Mapping verwenden. Anhand des Werts in diesem Feld werden bei einer Quellkomponente Spaltennamen gelesen. Außerdem wird anhand dieses Werts eine Vorschau des Inhalts der Instanztextdatei angezeigt. Um eine neue Datei auszuwählen, klicken Sie auf **Eingabedatei**.
Ausgabedatei	Definiert die Datei, in die MapForce die Daten schreibt. Dieses Feld hat bei einer Zielkomponente eine Bedeutung. Um eine neue Datei auszuwählen, klicken Sie auf **Ausgabedatei**.
Alle Dateipfade relativ zur MFD-Datei speichern	Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld "Komponenteneinstellungen" angezeigten Dateipfade relativ zum Ordner, in dem sich die MapForce Design (.mfd)-Datei befindet. Diese Einstellung wirkt sich auf die von der Textkomponente verwendeten Eingabe- und Ausgabedateien aus. Siehe auch Verwenden relativer Pfade in einer Komponente 85.
Eingabe- / Ausgabekodierung	Damit können Sie die folgenden Einstellungen der Output-Instanzdatei definieren: • Kodierungsname • Bytefolge • Ob das Bytefolgemarkierungszeichen (BOM) inkludiert werden soll. Standardmäßig haben alle neuen Komponenten die in der Option **Standardkodierung für neue Komponenten** definierte Kodierung. Sie können diese Option über **Extras | Optionen,** Register "Allgemein" aufrufen.
Füllzeichen	Mit Hilfe dieser Option können Sie definieren, welches Zeichen verwendet werden soll, um den Rest von Feldern mit fester Länge auszufüllen, wenn die eingehenden Daten kürzer sind als die jeweils definierte Feldlänge. Im Feld "Benutzerdef." können Sie Ihr eigenes Füllzeichen definieren. Wenn die eingehenden Daten bereits spezielle Füllzeichen enthalten, können Sie dieses Zeichen in das Feld "Benutzerdefiniert" eingeben, sodass diese Füllzeichen aus den eingehenden Daten entfernt werden.

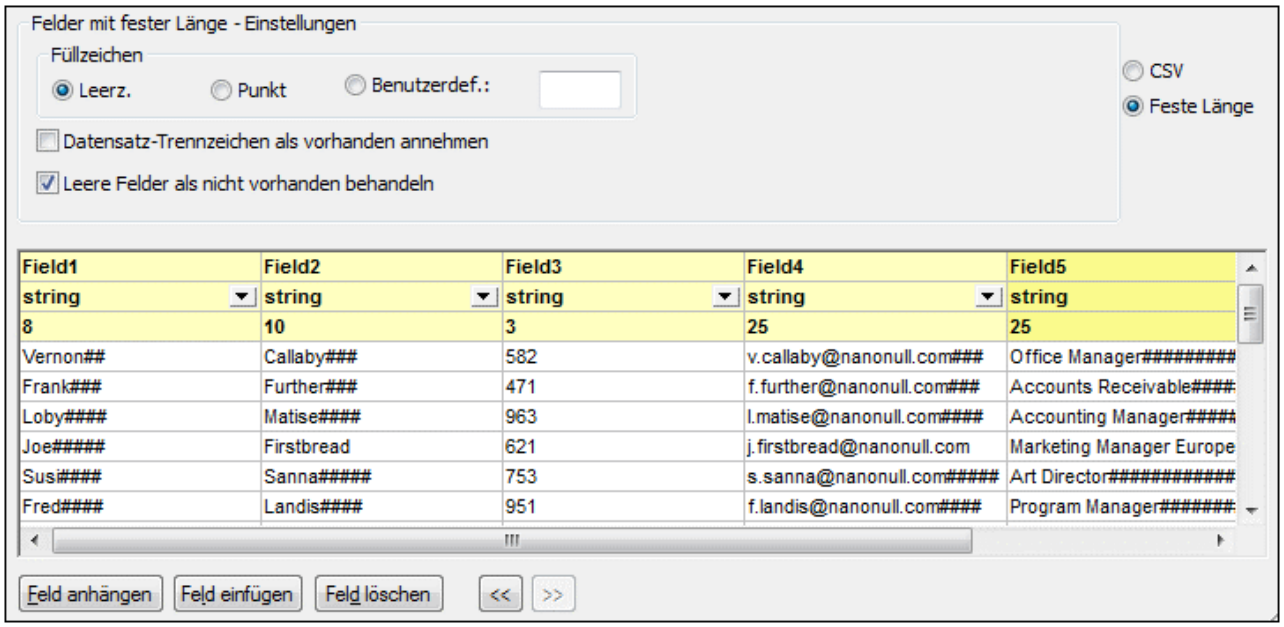
Datensatz-Trennzeichen als vorhanden annehmen	Diese Option ist nützlich, wenn Sie Daten aus einer Flat File-Quelldatei lesen möchten, die keine Datensatz-Trennzeichen wie z.B. CR/LF enthält, oder wenn Sie eine FLF-Zieldatei ohne Datensatz-Trennzeichen generieren möchten. Nähere Informationen dazu finden Sie im weiter unten im Abschnitt Die Option "Datensatz-Trennzeichen als vorhanden annehmen" 607.
Leere Felder als nicht vorhanden behandeln	Wenn diese Option aktiviert ist, werden in der Zieldatei keine entsprechenden leeren Datenelemente (Elemente oder Attribute) anhand leerer Felder in der Quelldatei erzeugt. Angenommen, es handelt sich bei der Ausgabedatei um eine XML-Datei, so werden die leeren Felder (in diesem Beispiel die Elemente `Last`, `Title` und `Email`) in der Ausgabe mit einem leeren Wert erstellt, wenn diese Option deaktiviert ist:  Wenn diese Option aktiviert ist, so werden die leeren Felder in der Ausgabe nicht erstellt: 
CSV / Feste Länge	Ändert den Komponententyp entweder in CSV oder FLF (Felder mit fester Länge).
Vorschaubereich	Im unteren Teil des Dialogfelds wird eine Vorschau von bis zu 20 Zeilen der als Ein- oder Ausgabedatei ausgewählten Datei angezeigt. Falls nötig, können Sie die Struktur der Datei folgendermaßen erstellen (bzw. die Struktur einer vorhandenen entsprechend ändern): **Feld anhängen** — Erstellt nach dem letzten CSV-Datensatz ein neues Feld. **Feld einfügen** — Erstellt unmittelbar vor dem aktuell ausgewählt CSV-Datensatz ein neues Feld. **Feld löschen** — Löscht das aktuell ausgewählte Feld. **<<** — Verschiebt das aktuell ausgewählte Feld um e Position nach links.

© 2016-2022 Altova GmbH

	>> Verschiebt das aktuell ausgewählte Feld um e Position nach rechts. Um den Namen eines Felds zu ändern, klicken Sie auf die Überschrift (in diesem Beispiel **Field1**) und geben Sie den neuen Wert ein.  Um den Datentyp eines Felds zu ändern, wählen Sie den gewünschten Wert aus der Dropdown-Liste aus. MapForce überprüft den Datentyp. Wenn die Input-Daten daher nicht mit dem Feldformat übereinstimmen, so werden die Daten rot markiert.  Um die Größe des Felds in Zeichen zu definieren, geben Sie die Anzahl in die dritte Zeile von oben aus ein..

Die Option "Datensatz-Trennzeichen als vorhanden annehmen"

Um diese Option besser zu verstehen, öffnen Sie die Datei **Altova-FLF.txt** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Beachten Sie, dass die Datei aus 71 Zeichen langen Datensätzen ohne Trennzeichen wie CR/LF besteht. Um Daten aus dieser Datei zu lesen, müssten Sie die Datei zuerst in Datensätze aufgliedern, d.h. Sie müssten mehrere Felder erstellen, deren Gesamtgröße insgesamt 71 Zeichen ausmacht (wie unten gezeigt). Anschließend müssen Sie die Option Datensatz-Trennzeichen als vorhanden annehmen deaktivieren. Eine schrittweise Anleitung dazu finden Sie im Beispiel: Mappen von Textdateien mit fester Länge auf Datenbanken 599.



Wenn Sie Daten von dieser Datei in eine Zieldatei mit derselben Struktur schreiben wollten, so würde bei Aktivierung der Option **Datensatz-Trennzeichen als vorhanden annehmen** nach jeweils 71 Zeichen ein neuer Datensatz erstellt.

```
 1  Vernon##Callaby###582v.callaby@nanonull.com###Office Manager##########
 2  Frank###Further###471f.further@nanonull.com###Accounts Receivable######
 3  Loby####Matise####963l.matise@nanonull.com####Accounting Manager#######
 4  Joe#####Firstbread621j.firstbread@nanonull.comMarketing Manager Europe#
 5  Susi####Sanna#####753s.sanna@nanonull.com#####Art Director############
 6  Fred####Landis####951f.landis@nanonull.com####Program Manager#########
 7  MichelleButler####654m.landis@nanonull.com####Software Engineer########
 8  Ted#####Little####852t.little@nanonull.com####Software Engineer########
 9  Ann#####Way#######951a.way@nanonull.com#######Technical Writer#########
10  Liz#####Gardner###753l.gardner@nanonull.com###Software Engineer########
11  Paul####Smith#####334p.smith@nanonull.com#####Software Engineer########
12  Alex####Martin####778a.martin@nanonull.com####IT Manager###############
13  George##Hammer####223g.hammer@nanonull.com####Web Developer############
14  Jessica#Bander####241j.band@nanonull.com######Support Engineer#########
15  Lui#####King######345l.king@nanonull.com######Support Engineer#########
16  Steve###Meier#####114s.meier@nanonull.com#####Office Manager##########
17  Theo####Bone######331t.bone@nanonull.com######Accounts Receivable######
18  Max#####Nafta#####122m.nafta@nanonull.com#####PR & Marketing Manager US
19  ValentinBass######716v.bass@nanonull.com######IT Manager###############
20  Carl####Franken###147c.franken@nanonull.com###Support Engineer#########
21  Mark####Redgreen##152m.redgreen@nanonull.com##Support Engineer#########
22
```

Das Mapping-Ergebnis bei Aktivierung der Option "Datensatz-Trennzeichen als vorhanden annehmen"

Wenn die Option **Datensatz-Trennzeichen als vorhanden annehmen** deaktiviert ist, so wird das Mapping als ein einziger langer String angezeigt.

© 2016-2022 Altova GmbH

```
1       Vernon##Callaby###582v.callaby@nanonull.com###Office
        Manager##########Frank###Further###471f.further@nanonull.com###Accounts
        Receivable######Loby####Matise####963l.matise@nanonull.com####Accounting
        Manager#######Joe#####Firstbread621j.firstbread@nanonull.comMarketing Manager
        Europe#Susi####Sanna#####753s.sanna@nanonull.com#####Art
        Director############Fred####Landis####951f.landis@nanonull.com####Program
        Manager##########MichelleButler####654m.landis@nanonull.com####Software
        Engineer########Ted#####Little####852t.little@nanonull.com####Software
        Engineer########Ann#####Way#######951a.way@nanonull.com#######Technical
        Writer#########Liz#####Gardner###753l.gardner@nanonull.com###Software
        Engineer########Paul####Smith#####334p.smith@nanonull.com#####Software
        Engineer########Alex####Martin####778a.martin@nanonull.com####IT
        Manager##############George##Hammer####223g.hammer@nanonull.com####Web
        Developer###########Jessica#Bander####241j.band@nanonull.com######Support
        Engineer#########Lui#####King######345l.king@nanonull.com######Support
        Engineer#########Steve###Meier#####114s.meier@nanonull.com#####Office
        Manager##########Theo####Bone######331t.bone@nanonull.com######Accounts
        Receivable######Max#####Nafta#####122m.nafta@nanonull.com#####PR & Marketing Manager
        USValentinBass######716v.bass@nanonull.com######IT
        Manager##############Carl####Franken###147c.franken@nanonull.com###Support
        Engineer#########Mark####Redgreen##152m.redgreen@nanonull.com##Support
        Engineer#########
```

Das Mapping-Ergebnis bei Deaktivierung der Option "Datensatz-Trennzeichen als vorhanden annehmen"

4.5 Webservices

Sie können ein Mapping in MapForce so designen, dass während der Ausführung ein Webservice aufgerufen wird. Sie können nämlich einen Webservice-Aufruf (ähnlich wie eine Funktion) in ein Mapping einfügen, Input- und Output-Komponenten damit verbinden und das Ergebnis nach Bedarf verarbeiten (Sie können z.B. eine Vorschau im MapForce-Ausgabefenster anzeigen, es an eine andere Komponente weiterleiten oder in eine Datei schreiben).

Ein mit MapForce erstellter Webservice-Aufruf kann entweder von MapForce selbst oder auf einem anderen Rechner oder sogar auf einer anderen Plattform mit Hilfe von MapForce Server ausgeführt werden. Mappings können von MapForce Server über die Befehlszeile, als API-Aufruf oder als FlowForce Server-Auftrag ausgeführt werden, siehe Automatisierung mit MapForce Server 1434.

Unterstützte Webservices

MapForce unterstützt den Aufruf von WSDL-basierten Webservices und REST-Webservices (*nähere Informationen siehe unten*).

- Bei WSDL-basierten Webservices müssen Sie die WSDL-Datei des Webservice bereitstellen. Mit Hilfe der WSDL-Datei kann MapForce mit dem Webserver kommunizieren. Wenn in der WSDL-Datei mehrere Services, Endpoints und Operationen implementiert sind, können Sie diese direkt in MapForce auswählen oder aktualisieren.
- Bei REST-Webservices müssen Sie die Webservice-Informationen (i) manuell oder (ii) automatisch (durch Import einer WADL-Datei oder einer URL) eingeben. Dazu gehören die URL, eine Request-Methode (z.B. GET, POST, PUT), die Request- und Response-Struktur (wie z.B. XML, JSON oder benutzerdefinierte MIME-Types) sowie Parameter.

REST-Services im Vergleich zu WSDL-basierten Webservices

Die folgende Tabelle enthält eine Übersicht darüber, was in Bezug auf REST- und WSDL-Webservices in MapForce unterstützt wird.

Funktion	REST	WSDL-basiert
Mapping-Sprache	Built-In	Built-In, C#, Java
Automatisierung mit MapForce Server	Ja	Ja, wenn die Sprache "BUILT- in" ist
Protokolle	HTTP (GET, POST, PUT, DELETE, benutzerdefinierte Verben)	• SOAP 1.1, 1.2 über HTTP • Nicht-SOAP-Dienste über HTTP
Request/Response-Strukturen	• XML • JSON • Protocol Buffer • benutzerdefinierte MIME-Types	SOAP-Nachricht
Sicherheit	• HTTP/HTTPS • Server-Zertifikate	• HTTP/HTTPS • Server-Zertifikate

© 2016-2022 Altova GmbH

Funktion	REST	WSDL-basiert
	• Client-Zertifikate • HTTP-Authentifizierung • Authentifizierung im Vorhinein • OAuth-Autorisierung	• Client-Zertifikate* • HTTP-Authentifizierung • Authentifizierung im Vorhinein* • WS-Security* • OAuth-Autorisierung* * *wird nur in Built-in unterstützt.*
Import der Service-Definition	Aus WADL-Datei (optional)	Aus WSDL-Datei (erforderlich)
Import von Parametern aus der URL	Ja	Nicht anwendbar
Dynamische Webservice-URL	Sie können die vollständige URL des Webservice als Parameter für das Mapping bereitstellen oder diese im Mapping definieren. Außerdem können Sie bestimmte URL-Teile als Parameter für das Mapping bereitstellen, während die Basis-URL im Mapping definiert wird.	Sie können die URL des Webservice als Parameter für das Mapping bereitstellen oder diese als fix (mit dem Mapping gespeichert) definieren.
Timeout (das Intervall, nach dem der Webservice-Aufruf aufgrund einer Zeitüberschreitung abgebrochen wird, falls keine Antwort vom Webservice eintrifft).	Ja	Ja
Dynamische Authentifizierung (die Authentifizierungsinformationen werden vom Mapping oder als Parameter für das Mapping bereitgestellt. Nähere Informationen finden Sie unter Dynamische Authentifizierung 1366).	Ja	Ja

WSDL-basierte Webservices

Bei Auswahl eines WSDL-basierten Webservice können Sie die folgenden Protokolle verwenden:

- SOAP 1.1, 1.2 über HTTP: Es wird sowohl der Stil `RPC/Encoded` als auch der Stil `Document/Literal` unterstützt. Wenn der Webserver einen WSDL-Fehler zurückgibt, wird die Ausführung des Mappings gestoppt. In solchen Fällen können Sie optional eine Ausnahmekomponente für die Fehlerbehandlung in den Mapping-Bereich einfügen. Wenn der Webservice einen nicht-WSDL-Fehler zurückgibt, wird die Mapping-Ausführung gestoppt und eine Fehlermeldung zurückgegeben (oder auf dem Bildschirm angezeigt, wenn Sie eine Mapping-Vorschau in MapForce anzeigen).
- Nicht-SOAP über HTTP: Dies bezieht sich auf weniger gebräuchliche Nicht-SOAP HTTP-Services. Bei HTTP GET wird der Stil `url-encoded` unterstützt. Bei HTTP POST werden die Stile `url-encoded` und `text/xml` unterstützt.

4.5.1 REST-Webservices

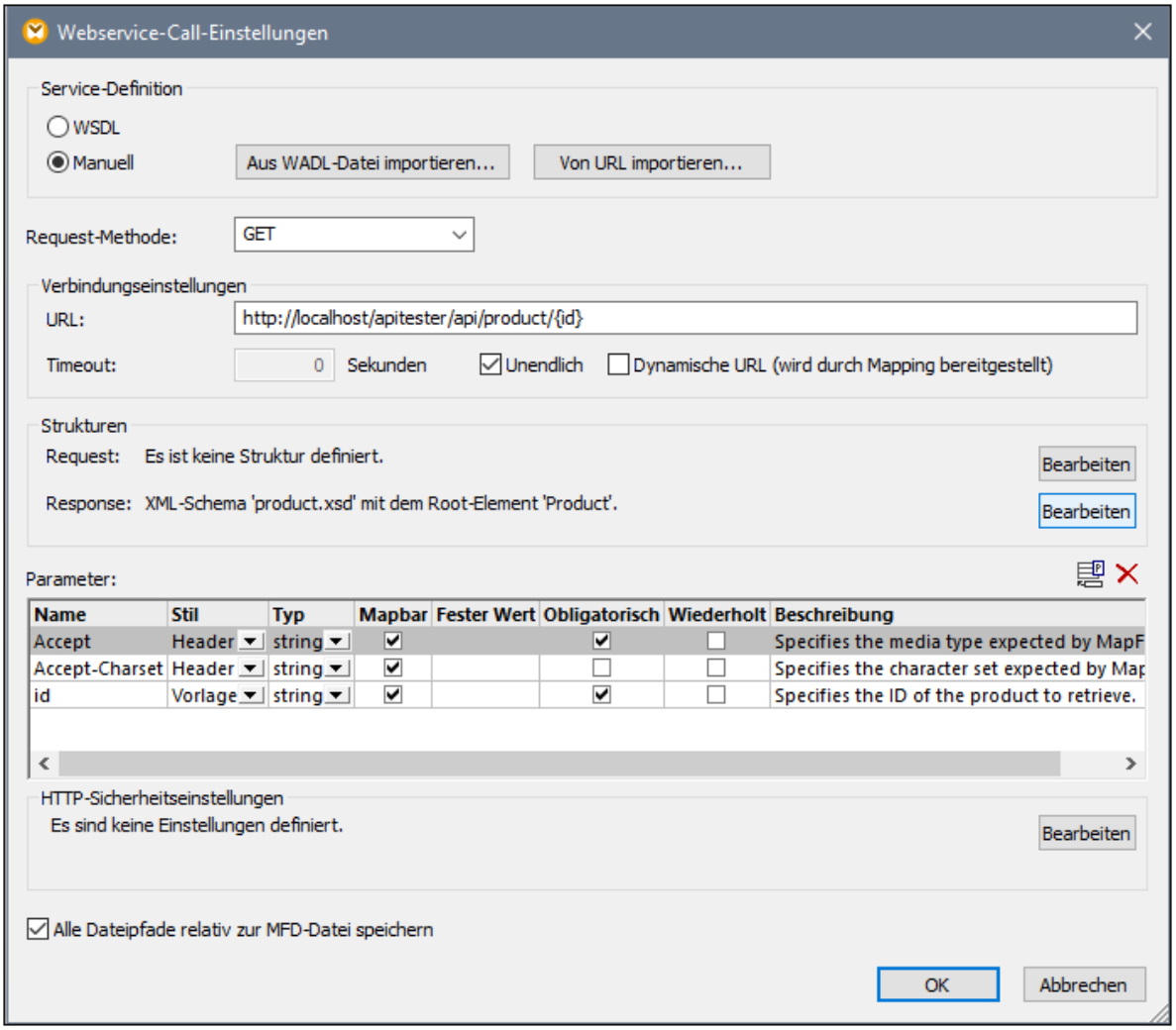
Zu den generischen Webservices (nicht WSDL-Webservices) gehört eine große Kategorie von Webservices, die von ihrer Architektur her einem Stil namens "REST" entsprechen bzw. teilweise entsprechen. Diese Art von Webservices wird normalerweise als RESTful oder REST-Webservices bezeichnet. Generische HTTP-Webservices übertragen normalerweise in ihrem Message Body benutzerdefinierte Request- oder Response-Strukturen. MapForce unterstützt die folgenden Arten von Request- oder Response-Bodies: JSON, XML, Protocol Buffers und unstrukturierte Bodys, die benutzerdefinierte MIME Types enthalten. Aus der Sicht von MapForce hängen die Voraussetzungen, um einen Webservice aufrufen zu können vom Request-Typ ab:

- Im Fall von XML- oder JSON-Requests oder Responses benötigt MapForce die Request- oder Response-Struktur als JSON-, XML-, oder DTD-Schema. MapForce akzeptiert als Struktur auch eine XML-Datei mit einer gültigen Schemareferenz. Die Struktur des Webservice kann vom Anbieter des Webservice in Form eines XML- oder JSON-Schemas oder in einer formalen Sprache wie WADL oder auch in Form einer vom Menschen lesbaren Spezifikation zur Verfügung gestellt werden. Wenn Sie das XML- oder JSON-Schema des Request/Response bereits zur Verfügung haben, lässt sich der Webservice-Aufruf am leichtesten erstellen. Wenn Sie die Webservice-Definition als WADL-Datei haben, können Sie die benötigten Informationen aus der WADL-Datei importieren und etwaige Anpassungen manuell vornehmen. Beachten Sie jedoch, dass WADL keine Standardmethode zum Definieren von JSON-Strukturen, sondern nur zum Definieren von XML-Strukturen bietet. Wenn Sie schließlich eine XML- oder JSON-Beispielinstanzdatei, nicht aber die Schema-Datei haben, können Sie das Schema mit XMLSpy entweder erstellen oder generieren (https://www.altova.com/de/xmlspy.html). Falls nötig, kann XMLSpy Ihre Instanzdatei auch von XML in JSON konvertieren oder umgekehrt.
- Im Fall von Protocol Buffers Requests oder Responses benötigen Sie die .proto-Datei, die die Protocol Buffers-Binärdatei beschreibt. In diesem Szenario kann der Body des Webservice von oder auf eine Protocol Buffers-Komponente gemappt werden. Nähere Informationen dazu finden Sie unter Beispiel: Auslesen von Daten aus Protol Buffers 849 und Beispiel: Schreiben von Daten in Protocol Buffers 854.
- Sie können auch Webservices aufrufen, bei denen die Request- oder Response-Struktur flexibel und nicht an ein bestimmtes Schema gebunden ist. ie können für solche Fälle die vordefinierten MapForce mime 1180-Funktionen verwenden, um entweder den an den Webservice gesendeten Raw Message Body (die MIME Entity) zu erstellen oder die vom Webservice retournierte MIME Entity über das Mapping zu verarbeiten.

Hinzufügen eines Aufrufs zu einem generischen Webservice

1. Klicken Sie im Menü **Einfügen** auf **Webservice-Funktion**. (Klicken Sie alternativ dazu in der Symbolleiste auf die Schaltfläche **Webservice-Funktion einfügen** ).
2. Klicken Sie unter **Service-Definition** auf **Manuell**.

© 2016-2022 Altova GmbH



3. Wenn Sie die WADL-Datei, in der der Service beschrieben ist, zur Verfügung haben, haben Sie auch die Möglichkeit, auf die Schaltfläche **Aus WADL-Datei importieren** zu klicken und die Datei auszuwählen (siehe Importieren von Webservice-Informationen aus einer WADL-Datei 623).
4. Wählen Sie die HTTP Request-Methode aus, mit der MapForce den Webservice aufrufen soll. Sie können entweder einen Wert aus der vorhandenen Liste auswählen oder den Namen der Request-Methode eingeben. Beachten Sie bei den Namen der HTTP-Methoden die Groß- und Kleinschreibung.
5. Wählen Sie eine der folgenden Methoden:

   a. Geben Sie die URL des Webservice in das Textfeld **URL** ein. Bei dieser Methode haben Sie die Möglichkeit, bestimmte Teile der URL in Parameter zu verwandeln und diese über das Hauptmapping bereitzustellen. Beachten Sie bei auf diese Art definierten URLs Folgendes:
      i. Wenn Sie einen Webservice mit "Vorlagen-" oder "Matrix"-Parametern aufrufen, setzen Sie die Parameter in geschweifte Klammern, z.B.: `http://example.org/api/products/{id}`. Definieren Sie anschließend die eigentlichen Einstellungen der einzelnen Parameter in der Tabelle "Parameter". MapForce verarbeitet die Parameternamen innerhalb geschweifter Klammern zur Laufzeit und erzeugt die endgültige URL, die die tatsächlichen Werte enthält.
      ii. Wenn Sie einen Webservice mit URL-"Abfrageparametern" (z.B. `http://example.org/api/products?sort=asc&category=1&page=1`) aufrufen, geben Sie

den Abfrageteil nicht in das URL-Textfeld ein, sondern definieren die Parameter nur in der Tabelle "Parameter" und stellen Sie sicher, dass diese als "Abfrage"-Parameter definiert sind.

iii. Ein Beispiel dazu finden Sie unter Definieren von Webservice-Parametern[617].

b. Wenn Sie die *vollständige* URL des Webservice über das Hauptmapping (oder eventuell als Parameter für das Mapping) bereitstellen möchten, aktivieren Sie das Kontrollkästchen **Dynamische URL (wird durch Mapping bereitgestellt)**. Dadurch wird das Textfeld **URL** deaktiviert. In diesem Fall müssen Sie die vollständige URL des Webservice (einschließlich aller URL-Parameter) über das Hauptmapping erstellen und diese mit dem entsprechenden für die Webservice-Komponente angezeigten Input verbinden. Beachten Sie, dass es bei dieser Methode außer für Header-Parameter nicht mehr sinnvoll ist, Parameter in der Tabelle "Parameter" zu definieren. Wenn Sie dies versuchen, wird ein Dialogfeld angezeigt, in dem Sie darauf hingewiesen werden.

Dynamische URLs
Sie können die URL mit beiden oben beschriebenen Methoden (vollständig oder teilweise dynamische URLs) flexibel, wie benötigt, anpassen. So könnten Sie das Mapping etwa in der Entwicklungsphase mit einer bestimmten URL ausführen und in der Produktion eine andere URL verwenden, ohne das Mapping ändern zu müssen. Bei einer URL wie `https://{host}/some/path/to/service` wäre dies möglich, vorausgesetzt, der Host-Name ist der einzige Unterschied zwischen der Produktions- und der Test-URL und Sie stellen diese als Parameter für das Mapping bereit. Beachten Sie, dass durch teilweise dynamische URLs eine strengere Validierung erzwungen wird, da nur die designierten URL-Teile durch mapbare Werte oder Laufzeitwerte ersetzt werden.

Bei vollständig dynamischen URLs ist die gesamte URL mapbar und Sie haben vollständige Kontrolle darüber. Die einzige Voraussetzung ist, dass die URL mit **http://** oder **https://** beginnt und es sich um eine gültige URI handeln muss. Bei dynamischen URLs kann außerdem die von einem Webservice-Aufruf retournierte URL als Input für eine andere Komponente, die denselben (oder einen anderen) Webservice aufrufen kann, bereitgestellt werden.

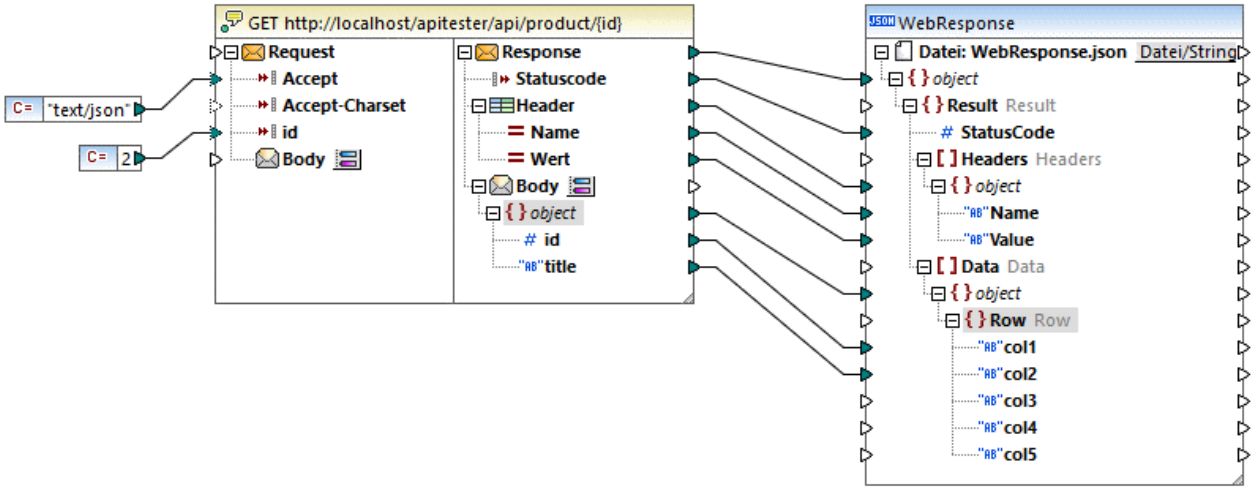
6. Geben Sie unter **Timeout** optional eine Zeitspanne in Sekunden ein, nach der die Verbindung beendet werden soll, wenn der Server nicht antwortet.
7. Wenn für die HTTP-Methode ein Body-Teil (wie XML, JSON und andere) benötigt wird, klicken Sie unter **Struktur** auf die Schaltfläche **Bearbeiten** und navigieren Sie zum Schema für den Body-Teil. Nähere Informationen dazu finden Sie unter Definieren der Request-Struktur[624] und Definieren der Response-Struktur[626].
8. Definieren Sie unter **Parameter** die Parameter des Webservice. Klicken Sie optional auf **Von URL importieren**, um die Parameter aus einer Beispiel-URL des Webservice zu importieren und die Tabelle "Parameter" automatisch zu befüllen (siehe Importieren von Webservice-Parametern aus einer URL[622]). Nachdem Sie die Parameter aus einer URL importiert haben, können Sie den Inhalt der Tabelle "Parameter" bei Bedarf bearbeiten.

Anmerkung: Um benutzerdefinierte Request Header zu definieren, fügen Sie einen Parameter mit dem Stil "Header" hinzu, wobei der Parametername dem Header-Namen und der Parameterwert dem Header-Wert entspricht. Falls der Wert des Request Header aus dem Mapping selbst abgerufen werden soll, setzen Sie den Parametertyp auf "Mapbar". Nähere Informationen dazu finden Sie unter Definieren von Webservice-Parametern[617].

9. Wenn vom Webservice eine einfache HTTP-Authentifizierung, eine OAuth 2.0-Autorisierung oder Sicherheitseinstellungen auf Basis eines Zertifikats verlangt werden, klicken Sie unter **HTTP-Sicherheitseinstellungen** auf die Schaltfläche **Bearbeiten** und füllen Sie die erforderlichen Felder aus, siehe auch Einstellen der HTTP-Sicherheit[635].

© 2016-2022 Altova GmbH

Nachdem Sie auf **OK** geklickt haben, wird eine neue Webservice-Komponente zum Mapping-Bereich hinzugefügt. Im unten gezeigten Mapping wird ein Webservice aufgerufen, um mit Hilfe eines GET Request ein Produkt anhand seiner ID abzurufen. In diesem konkreten Beispiel hat die ID im HTTP Request den Konstantenwert "2". Sie könnte aber auch ein Parameter im Mapping sein oder von einer von MapForce unterstützten Komponente bereitgestellt werden. Neben dem Parameter `id` enthält der Request den Header `Accept: text/json`. Sie können die Request Header mit Hilfe der "header"-Parameter definieren, wie in Definieren von Webservice-Parametern 617 beschrieben.



Beachten Sie, dass die Webservice-Komponente aus zwei Teilen besteht: **Request** und **Response**. Über den **Request**-Teil können Sie Daten aus dem Mapping für den Webservice bereitstellen, während Sie über den **Response**-Teil die vom Webservice retournierten Daten aufrufen und auf andere Formate mappen können. Die Struktur des Request und Response hängt von den Parametern sowie der Request- oder Response-Struktur ab, die Sie durch Klick auf die Schaltfläche definiert haben. Im obigen Webservice-Aufruf ist die Struktur des Request Body nicht definiert, während die Struktur des Response Body als JSON definiert ist. Dadurch kann das Ergebnis des Webservice auf eine JSON-Datei gemappt werden.

Um den Webservice mit Request-Parametern (falls anwendbar) aufzurufen, ziehen Sie Mapping-Verbindungen zwischen einer beliebigen, von MapForce unterstützten Komponente (z.B. einer XML- oder JSON-Datei) und dem **Request**-Teil. Auf die gleiche Art können Sie auch vom Webservice stammende Daten auf ein anderes Format mappen, indem Sie Mapping-Verbindungen zwischen dem **Response**-Teil und einem beliebigen anderen, von MapForce unterstützten Komponententyp ziehen. Wenn Sie ein MapForce-Neuling sind und eine Anleitung zum Ziehen von Mapping-Verbindungen benötigen, schlagen Sie nach unter Arbeiten mit Verbindungen 89.

Auch die vom Webservice zurückgegebenen Response Header können gemappt werden, wenn es sich um zusätzliche Header handelt (solche, die nicht mit "Content" beginnen). Die Header-Werte stehen in der Webservice-Komponente über ein Datenelement namens "Headers" ( ), das die zwei Child-Elemente "Name" und "Value" enthält, zur Verfügung. Diese Struktur fungiert als Sequenz und ermöglicht Ihnen, Daten von beliebig vielen vom Webservice zurückgegebenen Headern zu mappen. Um die Daten aus den Response Headern auf ein anderes von MapForce unterstütztes Format zu mappen, verbinden Sie den "Headers"-Struktur-Node mit der Zielsequenz im Mapping. Wenn Sie z.B. die **Headers**-Sequenz (und ihre untergeordneten Datenelemente) mit einer **Zeilen**-Sequenz (und deren untergeordneten Datenelementen) einer CSV-Komponente verbinden, würde ein Header einer Zeile in einer CSV-Datei entsprechen.

Der **Body** repräsentiert den Entity Body der HTTP-Nachricht. Daten auf dieser Ebene sind binär kodiert, daher werden für eine direkte Interaktion mit diesen Daten MapForce mime[1180]-Funktionen benötigt. Beachten Sie, dass normalerweise keinerlei Interaktion mit dem **Body** erforderlich ist, wenn vom Webservice strukturierte Daten wie z.B. XML oder JSON erwartet werden. Das Mappen von Daten direkt von oder auf den **Body** ist nur beim Aufruf von Webservices, die unstrukturierten Inhalt erwarten oder zurückgeben, notwendig.

> Standardmäßig parst das Datenelement **Body** per Konfiguration das Ergebnis, wenn der HTTP-Statuscode zwischen 200 und 299 liegt, d.h. das Mapping gibt für Statuscodes über 299 einen Fehler zurück. Außerdem wird auch dann ein Fehler zurückgegeben, wenn die Response nicht geparst werden kann oder der Webservice aufgrund einer nicht zustande gekommenen Verbindung oder DNS-Auflösungsproblemen nicht aufgerufen werden kann.

In einigen Fällen soll jedoch auch bei HTTP-Statuscodes, die über 299 liegen, kein Fehler ausgegeben werden. Klicken Sie dazu auf die Schaltfläche neben dem Response **Body** und ändern Sie den HTTP-Statuscodebereich. Alternativ dazu können Sie auch mehrere Response **Body**-Datenelemente erstellen. Dies ist nützlich, wenn Sie das Mapping auf Basis von Bedingungen behandeln möchten, je nachdem, welcher HTTP-Statuscode vom Webservice retourniert wird. Klicken Sie dazu mit der rechten Maustaste auf den **Body** des Response-Teils und wählen Sie im Kontextmenü den Befehl **Body-Node hinzufügen vor/nach**. So könnten Sie z.B. zwei **Body**-Datenelemente erstellen:

1. einen **Body** für alle Statuscodes, die sich innerhalb des Bereichs "erfolgreich" (200 bis 299) befinden. Durch Klick auf die Schaltfläche können Sie den Statuscodebereich konfigurieren. Normalerweise sollte die **Body**-Struktur bei diesem Szenario auf einen "success" (erfolgreich) Output, wie z.B. eine JSON- oder XML-Datei gemappt werden.
2. **einen** für alle anderen Statuscodes im Bereich zwischen 300 und 599. Einige Webservices stellen eventuell zusätzliche Informationen über Statuscodes im Body bereit. **Um den Grund für den Fehler festzustellen, können Sie daher den "erroneous" Body (fehlerhaft) auf einen anderen Output als den "success"** Body mappen (zB. einen String)

Wenn mehrere **Body** -Datenelemente vorhanden sind, wertet MapForce diese von oben nach unten aus. Um die Priorität zu ändern, klicken Sie mit der rechten Maustaste auf das **Body**-Datenelement und wählen Sie im Kontextmenü den Befehl **Nach oben** oder **Nach unten**. Nähere Informationen zur Behandlung der HTTP Response auf Basis von Bedingungen finden Sie unter Behandlung der HTTP Response auf Basis von Bedingungen[628].

Mappings, die generische HTTP-Webservice-Aufrufe enthalten, können wie die meisten anderen Mappings ausgeführt werden. Es gibt die folgenden Ausführungsmöglichkeiten:

- Manuell mit MapForce durch Klicken auf das Register **Ausgabe**. In diesem Fall steht das Ergebnis des Mapping-Aufrufs sofort im Fenster **Ausgabe** zur Verfügung. Wenn das Mapping wie im Beispiel oben mehrere Output-Komponenten hat, klicken Sie vor Ausführung des Mappings auf die Schaltfläche **Vorschau** der gewünschten Komponente.
- Mit MapForce Server über die Befehlszeile oder über API-Aufrufe (https://www.altova.com/de/mapforce/mapforce-server.html). Dazu muss das Mapping zuerst zu einer Mapping-Ausführungsdatei kompiliert werden (siehe Kompilieren von Mappings zu MapForce Server-Ausführungsdateien[1442]).
- Als wiederholt ausgeführter Auftrag, bei dem MapForce Server von FlowForce Server ausgeführt wird (https://www.altova.com/de/flowforce.html). Siehe auch Bereitstellen von Mappings auf FlowForce Server[1445].

Schrittweise Beispiele zur Erstellung eines generischen Webservice-Aufrufs finden Sie unter:

© 2016-2022 Altova GmbH

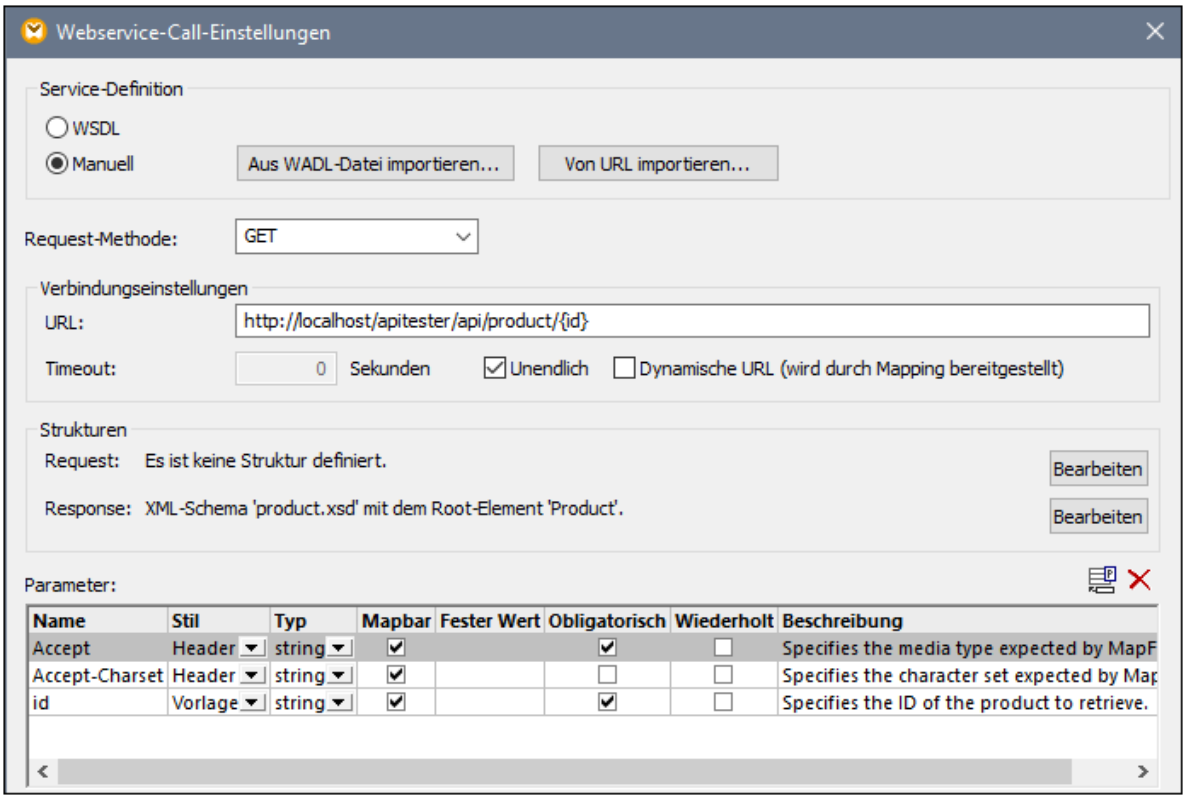
- Beispiel: Aufrufen eines REST-Webservice 640
- Beispiel: Mappen von Daten von einem RSS Feed 644

4.5.1.1 REST-Webservice-Parameter

Wenn Sie einen Webservice mit URL-Parametern aufrufen müssen, so müssen die Parameter im Dialogfeld **Webservice-Call-Einstellungen** definiert werden. In diesem Dialogfeld können Sie auch den Webservice Request mit benutzerdefinierten Headern definieren. Sie können die Parameterwerte als *mapbar* (über das Mapping bereitgestellt) oder *festgelegt* (im Dialogfeld definiert) konfigurieren.

Um Webservice-Parameter zu definieren, gehen Sie folgendermaßen vor:

1. Doppelklicken Sie im Mapping auf die Titelleiste der Webservice-Komponente, oder klicken Sie mit der rechten Maustaste darauf und wählen Sie den Befehl **Eigenschaften**. Daraufhin wird das Dialogfeld **Webservice-Call-Einstellungen** aufgerufen.



2. Klicken Sie unter **Service-Definition** auf **Manuell**. Sie können die Parameter entweder manuell oder automatisch (durch Import aus einer vorhandenen URL 622 oder einer WADL-Datei 623) eingeben. Die importierten Parameter stehen anschließend in der Tabelle **Parameter** zur Verfügung (*siehe Unterabschnitt unten*), wo Sie diese gegebenenfalls weiter bearbeiten können.
3. Verwenden Sie dazu die Schaltflächen **Parameter hinzufügen** ( ) bzw. **Parameter löschen** ( ).

Parameter

Die folgenden Unterabschnitte enthalten einen Überblick über die im Dialogfeld **WebService-Call-Einstellungen** verfügbaren Parameter.

Name

Definiert den Namen des URL-Parameters. Der Parametername muss eindeutig sein und darf aus Buchstaben, Ziffern, Punkten (.), Bindestrichen (-) und Unterstrichen (_) bestehen. Im Parameternamen sind keine Leerzeichen gestattet.

Stil

Definiert die Syntax (den Stil) des URL-Parameters.

- Verwenden Sie den Stil *Header*, um beim Aufruf des Webservice einen Parameter zum HTTP Header hinzuzufügen. Wenn Sie z.B. einen Parameter `Accept` mit dem Wert `text/xml` hinzufügen, ist dies dasselbe, wie wenn Sie den `Accept: text/xml` Header definieren, welcher den Webserver informiert, dass MapForce eine Response im XML-Format erwartet. Nähere Informationen zu HTTP Headern finden Sie in der Message Header-Spezifikation.
- Verwenden Sie für URL-Parameter, die Wert-Schlüssel-Paare im Format **`?key=value&key=value`** definieren (z.B. `http://example.org/api/products?sort=asc&category=1&page=1`) den Stil *Abfrage*.
- Verwenden Sie für in geschweifte Klammern gesetzte URL-Parameter (z.B.: **`http://example.org/api/products/{id}`**) den Stil *Vorlage*. Werte für derartige Parameter werden von MapForce entsprechend den https://tools.ietf.org/html/rfc6570 RFC 6570-Regeln mit Escape-Zeichen versehen.
- Verwenden Sie für URL-Parameter, die Schlüssel-Wert-Paare im Format **`;key=value;key=value;`** (z.B., `http://example.org/api/products;sort=asc;category=1;page=1;`) definieren, den Stil *Matrix*. Um Boolesche Matrixparameter zu verwenden, wählen Sie als Stil *Matrix* aus und setzen Sie den Typ auf `boolean`.
- Wählen Sie *Body*, um eine `application/x-www-form-urlencoded` Request-Struktur an einen REST-Service zu senden (Die Request-Methode muss POST sein). Nähere Informationen zum Format `application/x-www-form-urlencoded` finden Sie in der URL-Spezifikation.

Typ

Definiert den Datentyp des Parameters (string, integer, date, usw.). Dabei kann es sich um jeden beliebigen XML-Schematyp handeln. Beachten Sie, dass jeder Wert, der kein String ist, während des Webservice-Aufrufs in einen String konvertiert wird. Es ist sinnvoll, einen Typ zu definieren, wenn MapForce Konvertierungsfehlermeldungen anzeigen soll, falls versucht wird, einen Webservice mit falschen Werten aufzurufen.

Mapbar

Aktivieren Sie dieses Kontrollkästchen, wenn Sie Werte aus dem Mapping an diesen Parameter übergeben möchten. Diese Option und die Option *Fester Wert* schließen einander gegenseitig aus.

Fester Wert

Definiert den Wert des Parameters. Gilt nur, wenn der Parameter einen konstanten Wert hat. Kann nicht angewendet werden, wenn der Parameter mapbar ist (*siehe vorige Option*).

© 2016-2022 Altova GmbH

- Obligatorisch

  Aktivieren Sie dieses Kontrollkästchen, wenn der Parameter vom Webservice zwingend verlangt wird. MapForce erzwingt bei Parametern, die obligatorisch und außerdem mapbar sind, Validierungsüberprüfungen: Es wird eine Fehlermeldung angezeigt, wenn der Parameter keinen Wert hat.

- Wiederholt

  Definiert, ob der Parameter einen einzigen Wert hat oder mehrere Werte haben kann. Diese Einstellung gilt nur für mapbare Parameter. Mit dieser Option können Sie mehrere Werte im selben Webservice-Aufruf über einen einzigen Parameter übergeben. Wenn Sie dieses Kontrollkästchen aktivieren, können Sie eine Sequenz von Werten mit dem Parameterstruktur-Node im Mapping verbinden. MapForce behandelt die Wertesequenz dann ja nach Parameterstil folgendermaßen:

  - Bei *Vorlagenparametern* werden die Werte durch Kommas getrennt an den Webservice übergeben (z.B: `http://example.org/api/products/1,2,3`).
  - Bei *Abfrageparametern* wird der Parametername für jeden Wert wiederholt (z.B: `http://example.org/api/products?color=red&color=green&color=blue`).
  - Wenn als Stil *Matrix* ausgewählt ist, werden mehrere Werte durch Kommas getrennt (z.B.: `http://example.org/api/products;color=red;color=green;color=blue;size=big;size=small;`).
  - Bei Auswahl des Stils *Header* wird der HTTP Header für jeden Wert wiederholt.

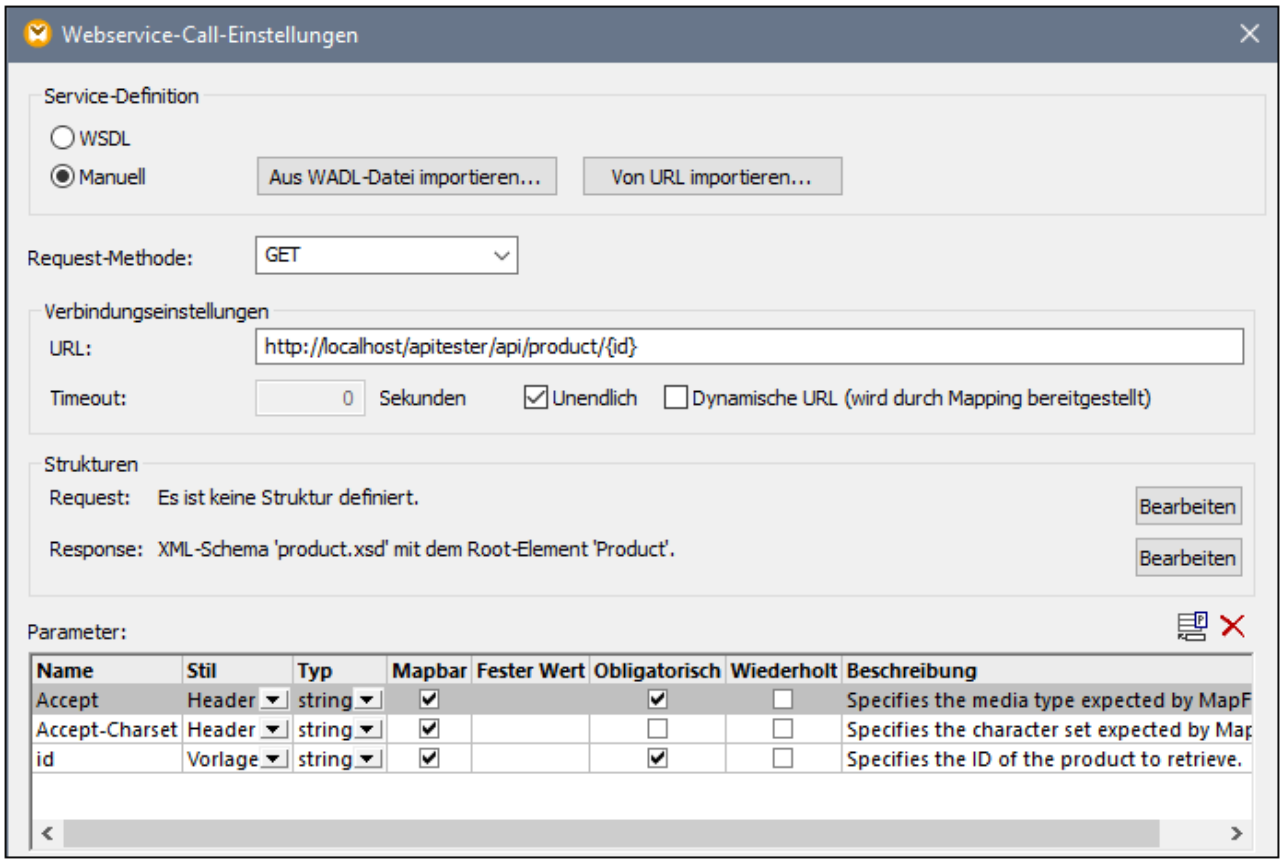
- Beschreibung

  Enthält die optionale Beschreibung des Parameters. Wenn der Parameter mapbar ist, wird die hier eingegebene Beschreibung in der Mapping-Komponente als Annotation neben dem Mapping-Datenelement angezeigt.

Beispiel 1: Header- und Vorlagenparameter

Mit dem unten gezeigten Webservice wird ein Produkt anhand seines Identifiers (`id`) mit Hilfe der HTTP GET-Methode abgerufen. In der URL des Webservice steht in den geschweiften Klammern der Parameter `id`. Beachten Sie, dass der Parameter `id` in der Tabelle "Parameter" vorhanden ist und den Stil *Vorlage* hat. Der Parameter ist auch **mapbar**. Dadurch wird der Parameter im Mapping als Struktur-Node, mit dem Sie den Wert von id verbinden können, angezeigt. `id` könnte z.B. aus einer Datenbank, einer Datei oder einer Konstante stammen. Dieser Parameter würde während der Mapping-Ausführung zur Laufzeit durch den aktuellen Wert ersetzt. Wenn also der Wert "1" ist, wird die URL `http://example.org/api/products/1` verwendet.

Um einen `id`-Wert in Form einer Konstante anstatt über das Mapping bereitzustellen, deaktivieren Sie die Option "Mapbar" und geben Sie den Wert in die Spalte "Fester Wert" ein.



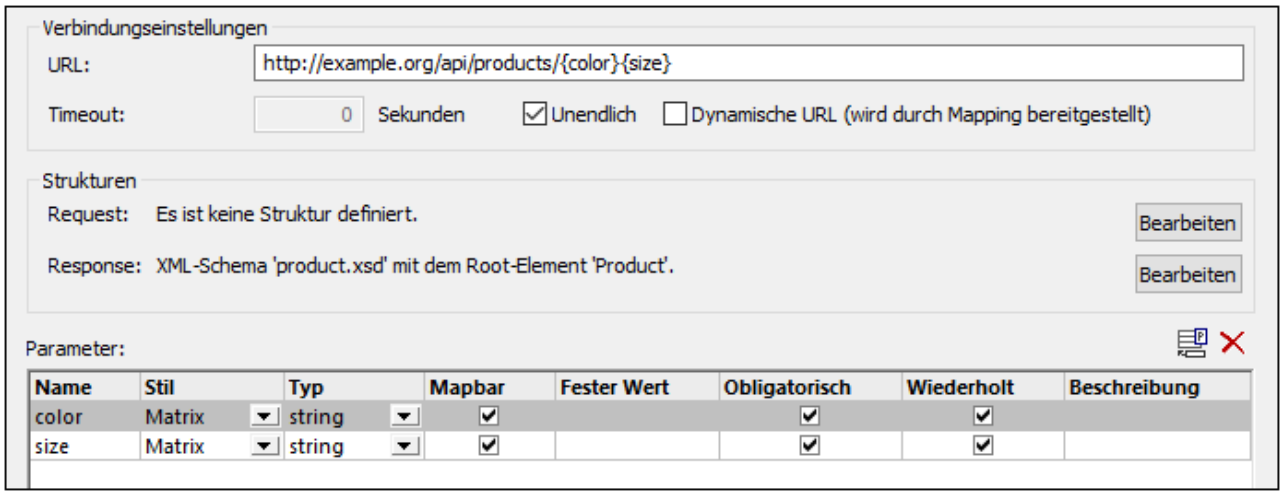
Die Parameter `Accept` und `Accept-Charset` haben den Stil *Header*. Mit Hilfe dieser Parameter können Sie den Webservice mit benutzerdefinierten Request Headern aufrufen. Es gibt zwei Methoden zur Bereitstellung des Header-Werts:

- Lassen Sie die Option **Mapbar** aktiviert und stellen Sie den benutzerdefinierten Header-Wert über das Mapping bereit oder
- Deaktivieren Sie die Option **Mapbar** und geben Sie den Wert direkt in die Spalte "Fester Wert" ein.

Beispiel 2: Matrix-Parameter

Im unten gezeigten Webservice wird eine Liste der Produkte abgerufen, die mit den als Argumente bereitgestellten Farb- und Größenwerten übereinstimmen. Die Stil der Parameter ist "Matrix", d.h. sie sind sowohl als Platzhalter in der URL als auch in der Mapping-Tabelle definiert. Beachten Sie, dass die Parameter mapbar sind und die Option "Wiederholt" aktiviert ist. Dadurch wird ihr Wert aus einer Sequenz von Werten im Mapping (z.B. aus einer Liste von Zeilen in einer Textdatei, einem XML-Node oder einer Datenbankspalte) ausgelesen und zur Laufzeit an den Webservice übertragen. Aus einer URL wie der unten gezeigten würde daher die URL `http://example.org/api/products/;color=red;color=blue;size=big;size=small`, falls vom Mapping die Farben `red` und `blue` und die Größen `big` und `small` bereitgestellt werden.

© 2016-2022 Altova GmbH



Wenn Ihre URL `http://example.org/api/products;color=red,blue;size=big,small` lauten soll, gehen Sie folgendermaßen vor:

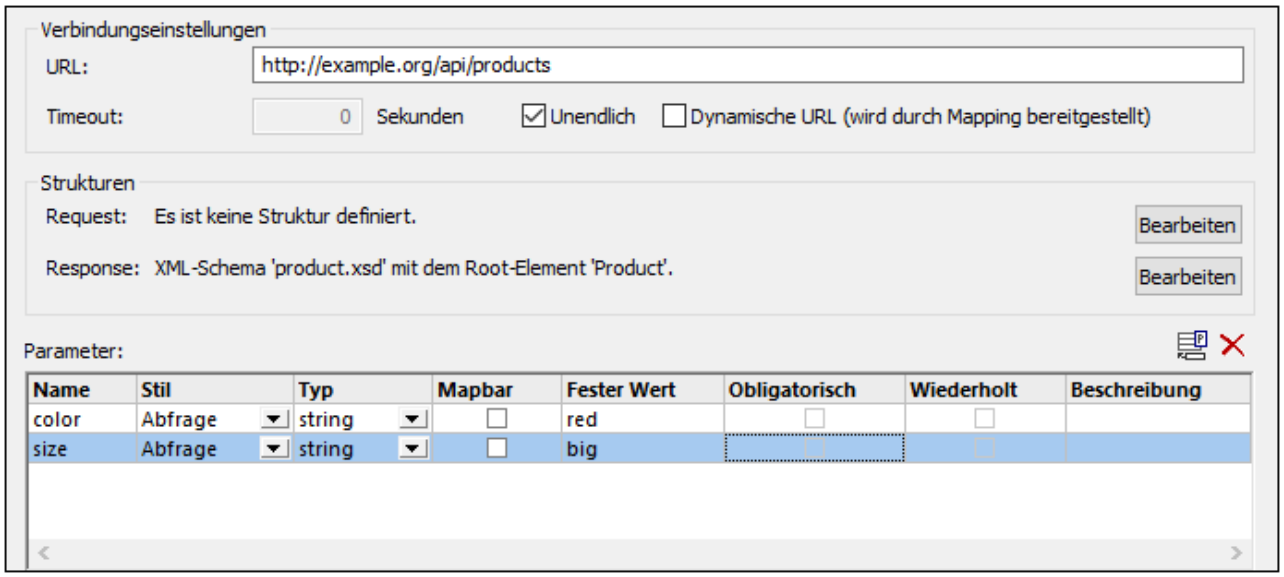
1. Geben Sie die folgende URL ein: `http://example.org/api/products;color={color};size={size}`
2. Ändern Sie den Parameterstil in der Tabelle "Parameter" in *Matrix*.

Wenn Ihre URL `http://example.org/api/products?color=red,blue&size=big,small` lauten soll, gehen Sie folgendermaßen vor:

1. Geben Sie die folgende URL ein: `http://example.org/api/products?color={color}&size={size}`
2. Ändern Sie den Parameterstil in der Tabelle "Parameter" in *Matrix*.

Beispiel 3: Abfrageparameter

Im unten gezeigten Webservice wird ebenfalls eine Liste von Produkten abgerufen, die mit der als Argumente bereitgestellten Farbe und Größe übereinstimmen. Dieses Mal wird der Stil "Abfrage" verwendet. Bei diesem Stil müssen die Parameter nicht als Platzhalter in der URL definiert werden, daher werden sie nur in der Tabelle "Parameter" definiert. Nehmen wir in diesem Beispiel an, dass die Parameterwerte fix sind und nicht über das Mapping bereitgestellt werden dürfen. Die Parameterwerte werden zu diesem Zweck unter "Feste Werte" eingegeben und die Option "Mapbar" ist deaktiviert. Daher wird die URL zur Mapping-Laufzeit zu **`http://example.org/api/products?color=red&size=big`**.



4.5.1.2 Importieren von Webservice-Parametern aus einer URL

Wenn in der URL zum Webservice Parameter verwendet werden, können Sie MapForce anweisen, die URL zu parsen und Parameter automatisch zu importieren. Wenn Sie diese Option verwenden, stehen anschließend alle aus der URL extrahierten Vorlagen-, Matrix- oder Abfrageparameter in der Tabelle "Parameter" des Dialogfelds "Webservice-Call-Einstellungen" zur Verfügung. Sie können diese dort weiter bearbeiten, falls nötig.

In der folgenden Tabelle finden Sie Beispiele für verschiedene URL-Stile:

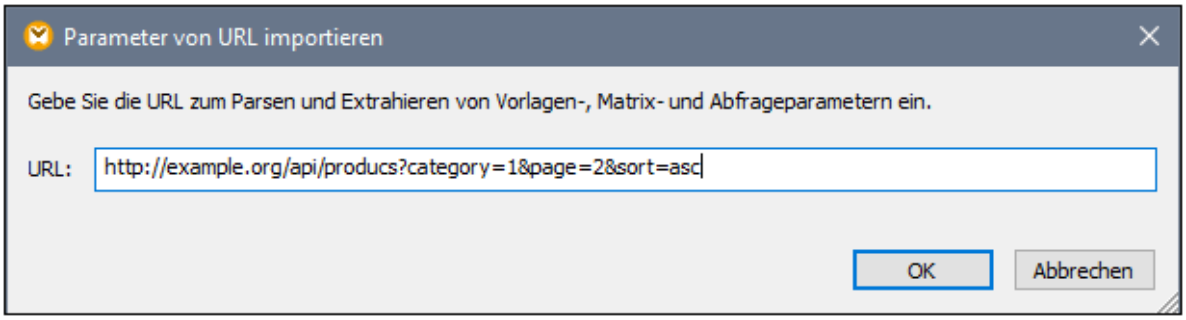
URL mit Vorlagenparametern	`http://example.org/api/products/{id}`
URL mit Matrixparametern	`http://example.org/api/products;sort=asc;category=1;page=1`
URL mit Abfrageparametern	`http://example.org/api/products?category=1&page=2&sort=asc`

Anmerkung: Eine URL muss entweder mit "http://" oder "https://" beginnen, um geparst werden zu können.

So importieren Sie Webservice-Parameter von einer URL:

1. Öffnen Sie das Dialogfeld "Webservice-Call-Einstellungen" (siehe Webservice-Call-Einstellungen 631).
2. Klicken Sie unter **Service-Definition** auf **Manuell**.
3. Klicken Sie auf **Von URL importieren**.

© 2016-2022 Altova GmbH



4. Geben Sie die URL zum Webservice in das Textfeld ein oder fügen Sie diese ein und klicken Sie auf OK.

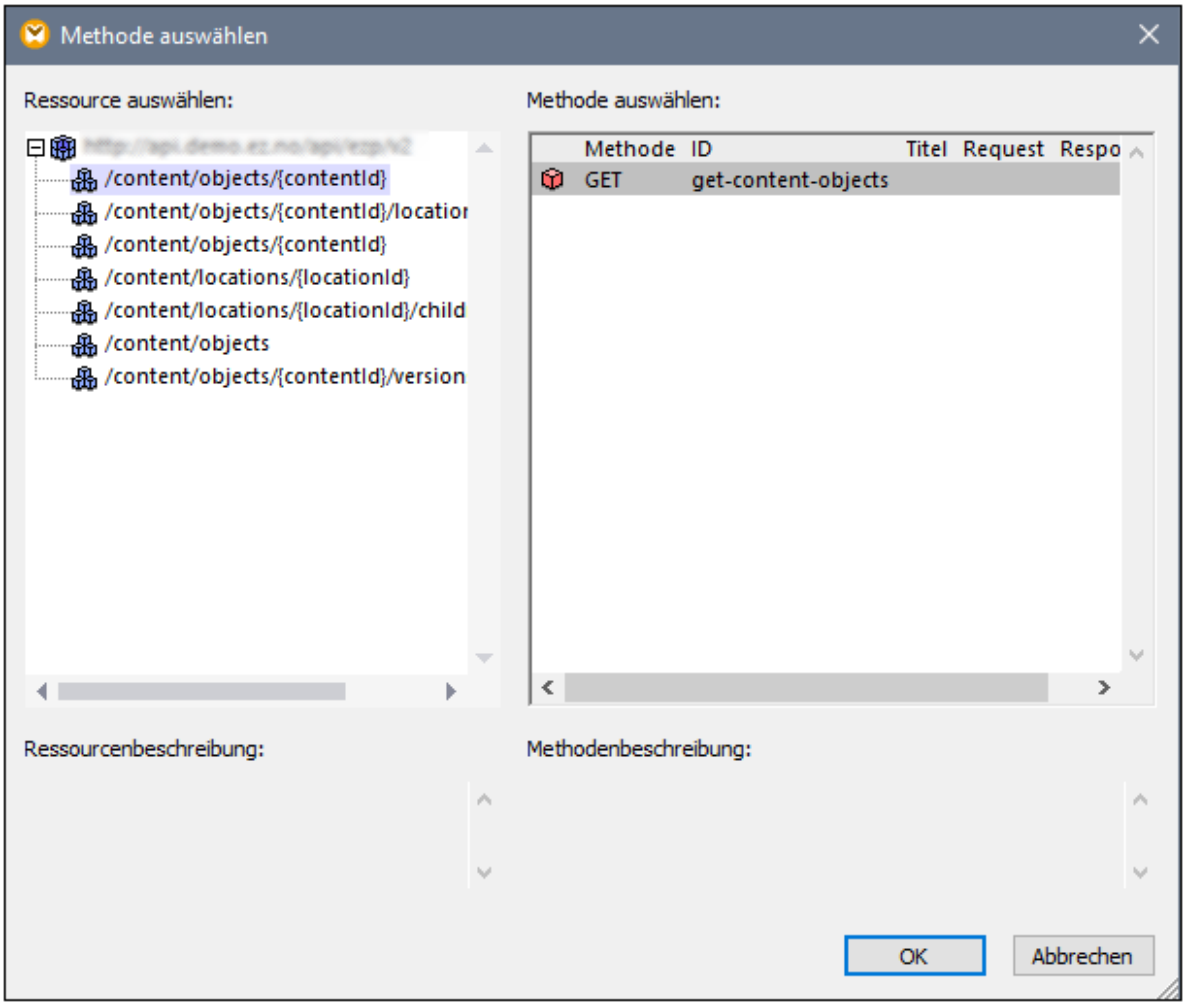
4.5.1.3 Importieren von Webservice-Informationen aus einer WADL-Datei

Web Application Description Language (WADL) ist eine der Methoden, um den Inhalt von in einer Web-Applikation (einschließlich RESTful Webservices) verwendeten Responses und Requests darzustellen. MapForce unterstützt den Import von Webservice-Informationen aus WADL Version 2009 (http://www.w3.org/Submission/2009/SUBM-wadl-20090831/).

Wenn ein Webservice eine WADL-Datei hat, können Sie die Webservice-Definition aus der WADL-Datei importieren, anstatt sie manuell einzugeben. Wenn Sie Informationen aus einer WADL-Datei importieren, werden die Parameter und die XML Message Body-Struktur im Dialogfeld "Webservice Call-Einstellungen" automatisch ausgefüllt. Sie können die Parameter und die Response/Request-Struktur nach dem Import gegebenenfalls manuell ändern. Dies hat keinen Einfluss auf die zugrunde liegende WADL-Datei.

So importieren Sie Webservice-Informationen aus einer WADL-Datei:

1. Öffnen Sie das Dialogfeld Webservice-Call-Einstellungen 631.
2. Wählen Sie unter Service-Definition die Option **Manuell** aus.
3. Klicken Sie auf **Aus WADL-Datei importieren** und navigieren Sie zur WADL-Datei.



4. Wählen Sie die Ressource und Methode im Dialogfeld aus und klicken Sie auf **OK**.

4.5.1.4 Definieren der Request-Struktur

Bei Aufruf von Webservices, die Inhalt im Body-Teil der Request Message erwarten, können Sie die Request-Struktur als eine der folgenden definieren:

- Keine Struktur
- XML
- JSON
- Protocol Buffers-Dateien, siehe Protocol Buffers 843

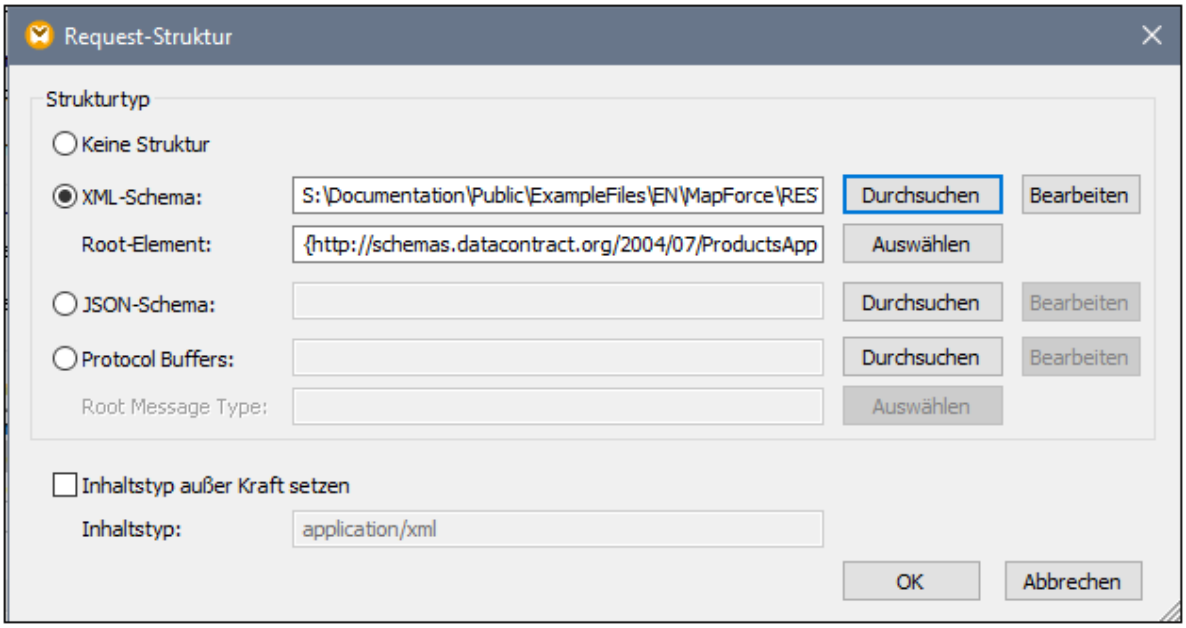
Anmerkung: In mit MapForce erstellten Webservice-Aufrufen können Sie als Request- oder Response-Struktur Standard-JSON verwenden; JSON Lines- oder JSON5-Strukturen werden nicht unterstützt.

Die hier ausgewählte Struktur bestimmt, welche mapbaren Datenelemente im Mapping als Children des Datenelements **Body** angezeigt werden sollen.

© 2016-2022 Altova GmbH

So definieren Sie die Request-Struktur:

1. Öffnen Sie das Dialogfeld "Webservice-Call-Einstellungen" (siehe Webservice-Call-Einstellungen 631).
2. Wählen Sie unter Service-Definition die Option **Manuell** aus.
3. Klicken Sie unter **Strukturen** neben **Request** auf **Bearbeiten**.



Tipp: Ein Shortcut zum Anzeigen des Dialogfelds ist die Schaltfläche im Request-Teil der Webservice-Komponente des Mappings.

4. Wählen Sie eine der folgenden Optionen aus:

   a) Wenn im Body-Teil kein Inhalt für den Webservice erwartet wird oder wenn Sie den Raw Body (MIME Entity) direkt über das Mapping bereitstellen möchten, wählen Sie die Option **Keine Struktur**.

   b) Holen Sie sich bei XML-Dateien das XML- oder DTD-Schema für den Request vom Webservice-Anbieter. Wenn Sie eine XML-Instanzdatei auswählen, muss diese eine gültige Schemareferenz haben. In einigen XML-Schemas sind Elemente mittels einer globalen Deklaration definiert (d.h. Elemente, deren übergeordnetes Element das `schema` Element ist). Bei solchen Schemas können Sie auswählen, welches Element im Schema das Root-Element der Mapping-Struktur in MapForce bilden soll. Klicken Sie dazu auf **Auswählen** und wählen Sie im anschließend angezeigten Dialogfeld das gewünschte Root-Element aus.

   c) Holen Sie sich bei JSON-Dateien das JSON-Schema für den Request vom Webservice-Anbieter oder generieren Sie es mit einem Tool wie XMLSpy anhand einer JSON-Beispieldatei.

Tipp: Wenn Sie eine XML-Beispielinstanzdatei oder eine JSON Request-Datei, aber keine Schema-Datei zur Verfügung haben, können Sie die Schema-Datei mit Hilfe von XMLSpy generieren. Falls nötig, kann XMLSpy Ihre Instanzdatei auch von XML in JSON konvertieren oder umgekehrt.

   b) Holen Sie sich bei Protocol Buffers-Datei die `.proto` -Datei, die die Binärdatei beschreibt, vom

Webservice-Anbieter und klicken Sie anschließend auf **Durchsuchen**, um die Datei auszuwählen. Das Feld **Root Message Type** ist anwendbar, wenn die Protocol Buffers-Datei mehrere Message Types enthält. Klicken Sie auf **Auswählen,** um den Message Type an der Root der Struktur auszuwählen.

5. Aktivieren Sie optional das Kontrollkästchen **Inhaltstyp außer Kraft setzen,** wenn Sie den implizit von MapForce definierten **Content-Type** Header außer Kraft setzen möchten. Sie können nach dem Inhaltstyp auch Parameter angeben, z.B.: `Content-Type: text/html; charset=utf-8`.

Wenn Sie das Mapping speichern, werden alle absoluten Pfade in diesem Dialogfeld relativ zur Mapping Design-Datei (.mfd) gespeichert, wenn in den Komponenteneinstellungen das Kontrollkästchen **Alle Dateipfade relativ zur MFD-Datei speichern** aktiviert wurde, siehe Webservice-Call-Einstellungen 631.

4.5.1.5 Definieren der Response-Struktur

Bei Aufruf von Webservices, die Inhalt im Body-Teil der Response Message zurückgeben, können Sie die Request-Struktur als eine der folgenden definieren:

- Keine Struktur
- XML
- JSON
- Protocol Buffers-Dateien, siehe Protocol Buffers 843

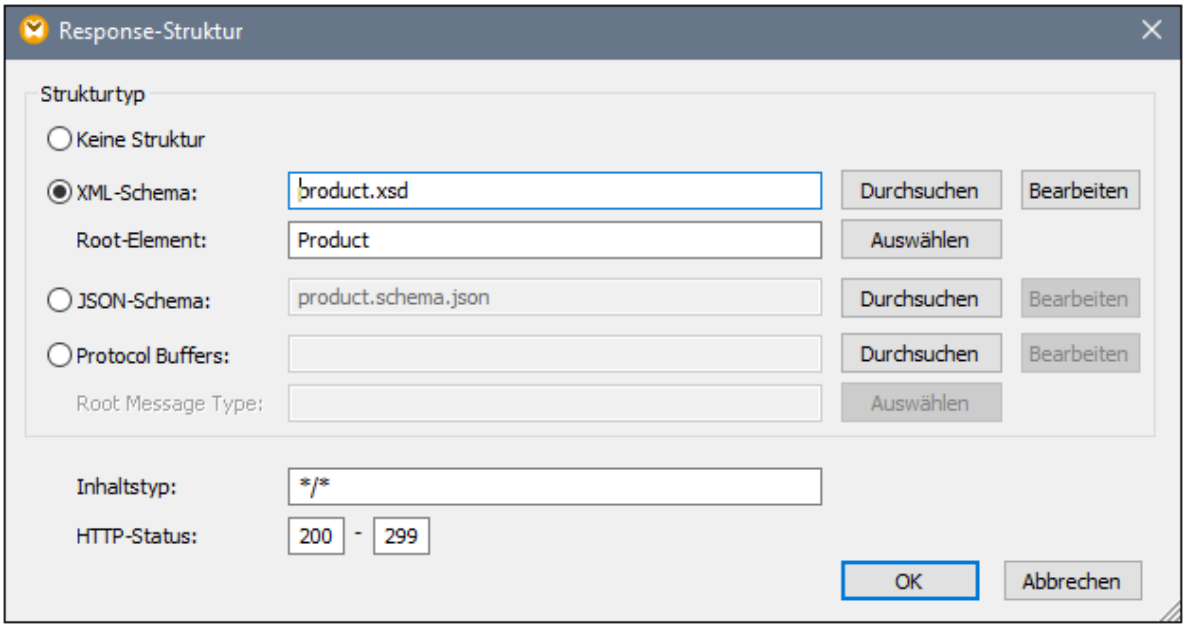
Anmerkung: In mit MapForce erstellten Webservice-Aufrufen können Sie als Request- oder Response-Struktur Standard-JSON verwenden; JSON Lines- oder JSON5-Strukturen werden nicht unterstützt.

Die hier ausgewählte Struktur bestimmt, welche mapbaren Datenelemente im Mapping als Children des Datenelements **Body** angezeigt werden sollen.

So definieren Sie die Response-Struktur:

1. Öffnen Sie das Dialogfeld "Webservice-Call-Einstellungen" (siehe Webservice-Call-Einstellungen 631).
2. Wählen Sie unter Service-Definition die Option **Manuell** aus.
3. Klicken Sie unter **Strukturen** neben **Response** auf **Bearbeiten**.

© 2016-2022 Altova GmbH



Tipp: Ein Shortcut zum Anzeigen des Dialogfelds ist die Schaltfläche im Request-Teil der Webservice-Komponente des Mappings.

1. Wählen Sie eine der folgenden Optionen aus:

   a) Wenn der Webservice im Body-Teil keinen Inhalt retourniert oder wenn Sie den Raw Body (MIME Entity) direkt über das Mapping bereitstellen möchten, wählen Sie die Option **Keine Struktur**.

   b) Holen Sie sich bei XML-Dateien das XML- oder DTD-Schema für den Request vom Webservice-Anbieter. Wenn Sie eine XML-Instanzdatei auswählen, muss diese eine gültige Schemareferenz haben. In einigen XML-Schemas sind Elemente mittels einer globalen Deklaration definiert (d.h. Elemente, deren übergeordnetes Element das `schema` Element ist). Bei solchen Schemas können Sie auswählen, welches Element im Schema das Root-Element der Mapping-Struktur in MapForce bilden soll. Klicken Sie dazu auf **Auswählen** und wählen Sie im anschließend angezeigten Dialogfeld das gewünschte Root-Element aus.

   c) Holen Sie sich bei JSON-Dateien das JSON-Schema für den Request vom Webservice-Anbieter oder generieren Sie es mit einem Tool wie XMLSpy anhand einer JSON-Beispieldatei.

Tipp: Wenn Sie eine XML-Beispielinstanzdatei oder eine JSON Request-Datei, aber keine Schema-Datei zur Verfügung haben, können Sie die Schema-Datei mit Hilfe von XMLSpy generieren. Falls nötig, kann XMLSpy Ihre Instanzdatei auch von XML in JSON konvertieren oder umgekehrt.

   b) Holen Sie sich bei Protocol Buffers-Datei die `.proto` -Datei, die die Binärdatei beschreibt, vom Webservice-Anbieter und klicken Sie anschließend auf **Durchsuchen**, um die Datei auszuwählen. Das Feld **Root Message Type** ist anwendbar, wenn die Protocol Buffers-Datei mehrere Message Types enthält. Klicken Sie auf **Auswählen,** um den Message Type an der Root der Struktur auszuwählen.

Füllen Sie optional die Textfelder **Inhaltstyp** und **HTTP-Status** aus, wenn Sie eine MIME Entity-Response nach Medien-(Inhalts)typ und HTTP-Statuscode filtern müssen. Dies ist nützlich, wenn Sie abhängig vom HTTP-Statuscode und Inhaltstyp, der vom Webservice-Aufruf zurückgegeben wird, eine Behandlung auf Basis

von Bedingungen definieren möchten. Ein Beispiel dazu finden Sie unter Behandlung der HTTP Response auf Basis von Bedingungen 628.

Standardmäßig ist für den Filter **Inhaltstyp** `*/*` definiert, wodurch alle Medientypen inkludiert werden. Die Standard-HTTP-Statuscodes sind **200** bis **299**. Wenn vom Webservice-Aufruf ein anderer nicht explizit behandelter Statuscode retourniert wird, schlägt die Mapping-Ausführung aufgrund eines Fehlers fehl.

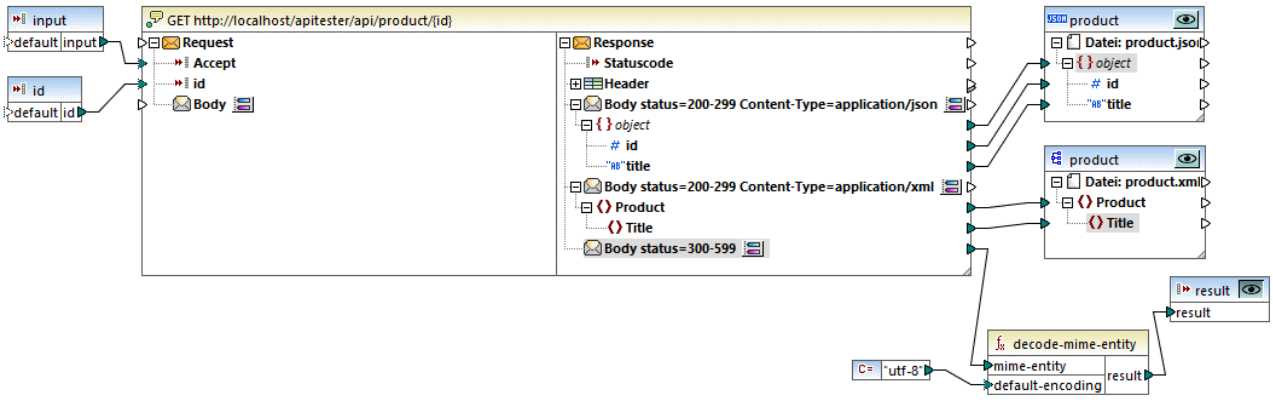
> Wenn Sie das Mapping speichern, werden alle absoluten Pfade in diesem Dialogfeld relativ zur Mapping Design-Datei (.mfd) gespeichert, wenn in den Komponenteneinstellungen das Kontrollkästchen **Alle Dateipfade relativ zur MFD-Datei speichern** aktiviert wurde, siehe Webservice-Call-Einstellungen 631.

4.5.1.6 Behandlung der HTTP Response auf Basis von Bedingungen

Angenommen, ein Webservice gibt im Response Body abhängig vom Wert des **Accept** HTTP Request Headers entweder einen JSON- oder einen XML-Inhaltstyp zurück. Zusätzlich dazu gibt der Webservice im Fall eines Fehlers (Fehlercode > 299) eventuell einen nicht definierten Inhaltstyp zurück. In einem solchen Fall ist es unter Umständen sinnvoll, die Response anhand des HTTP-Statuscodes und Inhaltstyps auf Basis von Bedingungen zu behandeln.

Um im selben Mapping sowohl die JSON- als auch die XML-Response des Webservice zu behandeln, könnten Sie mehrere Body-Datenelemente erstellen. Klicken Sie dazu mit der rechten Maustaste in der Komponente auf das Body-Datenelement und wählen Sie im Kontextmenü den Befehl **Body-Node hinzufügen nach**.

Eine Webservice-Komponente mit mehreren für das obige Szenario konfigurierten **Body**-Datenelementen könnte im Mapping folgendermaßen aussehen:



Das oben gezeigte Mapping erhält zwei Input-Parameter. Er erste Parameter liefert den erwarteten Inhaltstyp. Sein Wert kann `application/json` oder `application/xml` sein. Der zweite Parameter liefert die numerische ID des abzurufenden Produkts.

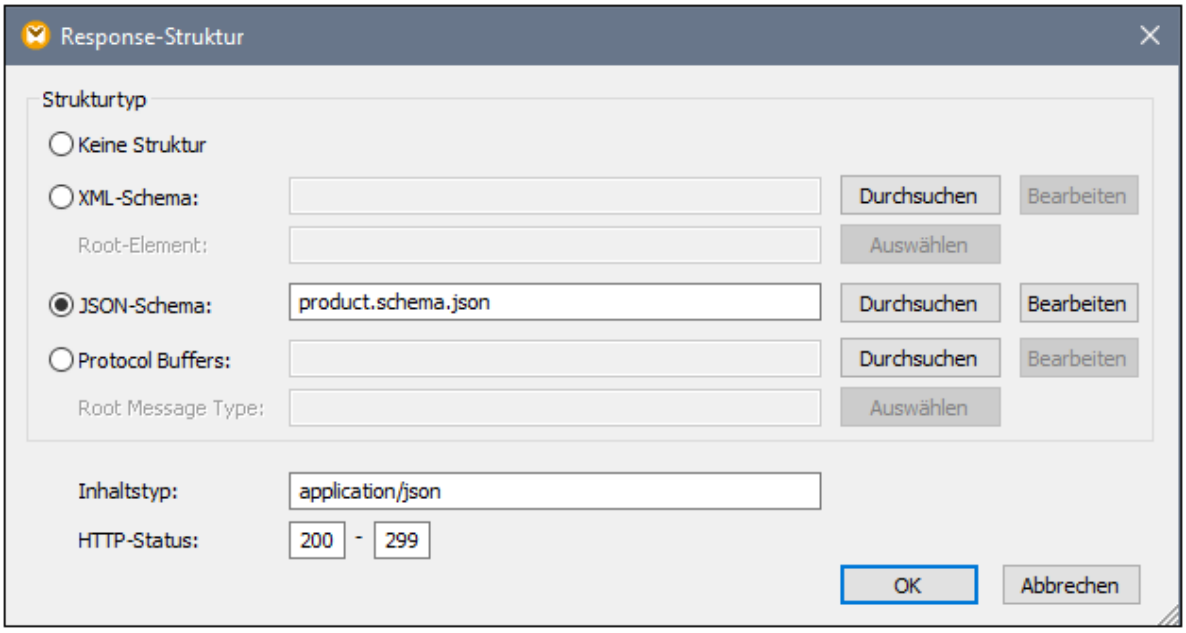
Wenn Sie den Wert `application/json` als Input-Parameter bereitstellen, klicken Sie auf die Schaltfläche **Vorschau** der JSON-Zielkomponente. Wenn der Wert des Input-Parameters hingegen `application/xml` ist, klicken Sie auf die Schaltfläche **Vorschau** der XML-Zielkomponente. Die jeweilige Ausgabedatei (JSON oder XML) wird generiert, wenn der Webservice-Aufruf erfolgreich war.

© 2016-2022 Altova GmbH

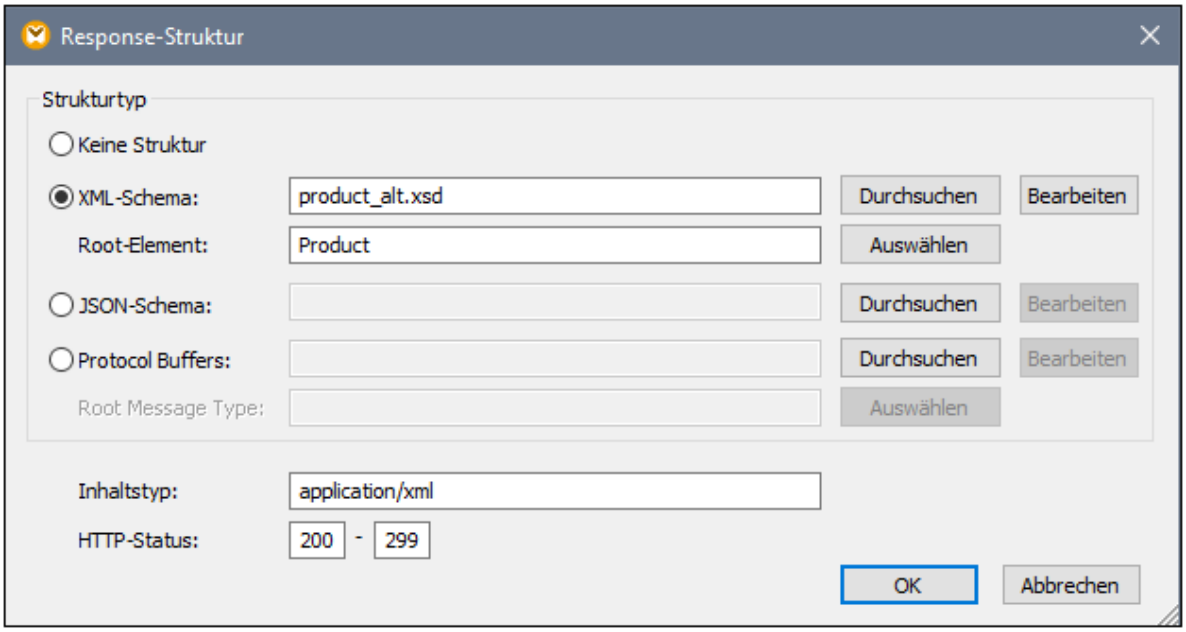
Wenn der Webservice einen Statuscode großer als 300 (Fehler) zurückgibt, gelangt die HTTP Response zum dritten **Body**-Datenelement. Da wir den genauen Inhaltstyp nicht kennen, werden wir ihn mit Hilfe der Funktion decode-mime-entity[1182] dekodieren. Der Wert des Zeichensatzes wird vom Webservice im Response Header nicht bereitgestellt, daher wurde der Konstantenwert **utf-8** mit der Dekodierungsfunktion verbunden.

Jedes der drei Response Body-Datenelemente in der obigen Komponente wird durch Klick auf die Schaltfläche neben dem jeweiligen Body-Element konfiguriert.

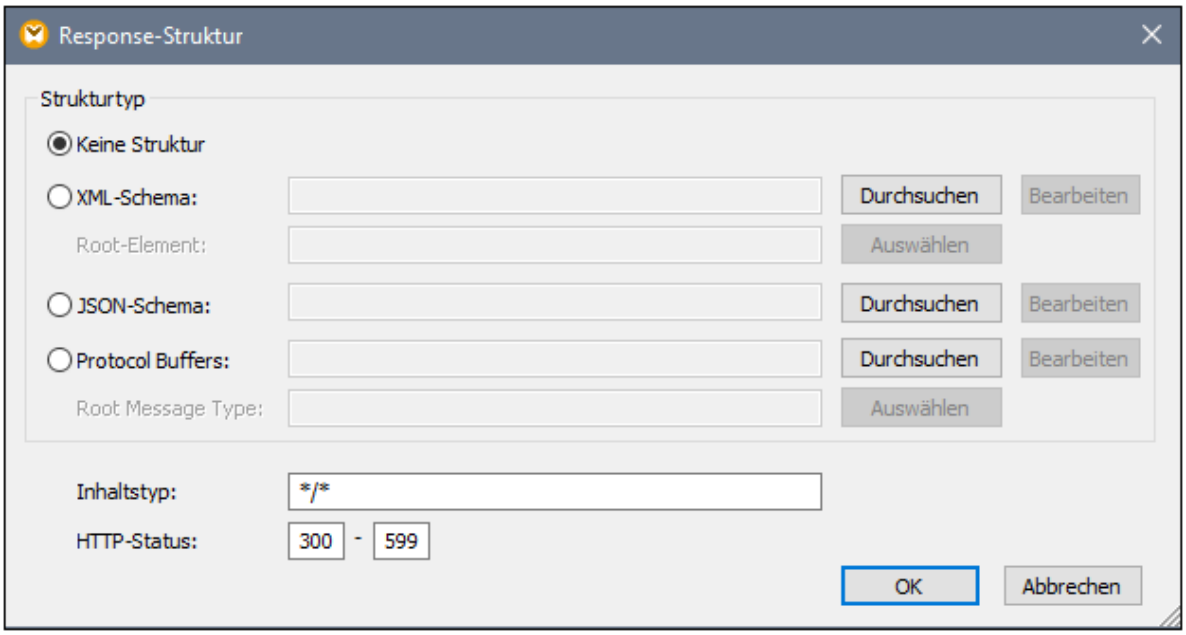
1. Im ersten Body-Element wird als **`application/json`** zurückgegebener Inhalt behandelt, wenn der Statuscode zwischen 200 und 299 liegt. Außerdem wurde das Schema der erwarteten JSON Response-Struktur definiert.



2. Im zweiten Body-Element wird als **`application/xml`** zurückgegebener Inhalt behandelt, wenn der Statuscode zwischen 200 und 299 liegt. Außerdem wurde das Schema der erwarteten XML Response-Struktur definiert.



3. Im dritten Body-Element wird als `*/*` zurückgegebener Inhalt behandelt, wenn der Webservice einen Fehler-Statuscode (zwischen 300 und 599) zurückgibt.



4.5.2 WSDL-basierte Webservices

In diesem Kapitel wird beschrieben, wie Sie einen Aufruf eines WSDL-Webservice zu einem Mapping hinzufügen. Eine Anleitung, wie Sie einen generischen Webservice zum Mapping hinzufügen, finden Sie unter Hinzufügen eines Webservice-Aufrufs (REST) 612.

 © 2016-2022 Altova GmbH

Bevor Sie einen WSDL-Webservice-Aufruf zum Mapping-Bereich hinzufügen, stellen Sie sicher, dass Sie die Web Service Definition Language (.wsdl), die den gewünschten Webservice, zu dem Sie eine Verbindung herstellen möchten, beschreibt, zur Verfügung haben.

So fügen Sie einen WSDL-Webservice-Aufruf zum Mapping hinzu:

1. Klicken Sie im Menü **Einfügen** auf **Webservice-Funktion**. (Klicken Sie alternativ dazu in der Symbolleiste auf die Schaltfläche **Webservice-Funktion einfügen** ).
2. Klicken Sie unter "Service-Definition" auf **WSDL**, falls diese Option noch nicht ausgewählt ist.
3. Klicken Sie unter "WSDL-Einstellungen" auf "Durchsuchen" und wählen Sie die Web Service Definition Language-Datei (.wsdl) aus. Im MapForce-Ordner "Examples" finden Sie zwei WSDL-Beispieldateien - eine für jede WSDL-Version (1.0 und 2.0). Die entsprechenden Pfade sind die folgenden: <Dokumente>\Altova\MapForce2023\MapForceExamples\TimeService\TimeService.wsdl and <Dokumente>\Altova\MapForce2023\MapForceExamples\TimeServiceWsdl2\TimeService20.wsdl.
4. Wenn mehrere Services, Operationen, Endpoints in der .wsdl-Datei definiert sind, klicken Sie auf **Auswählen** und wählen Sie die gewünschten Optionen aus.
5. Definieren Sie gegebenenfalls andere Einstellungen (siehe Webservice Call-Einstellungen 631).
6. Klicken Sie auf **OK**.

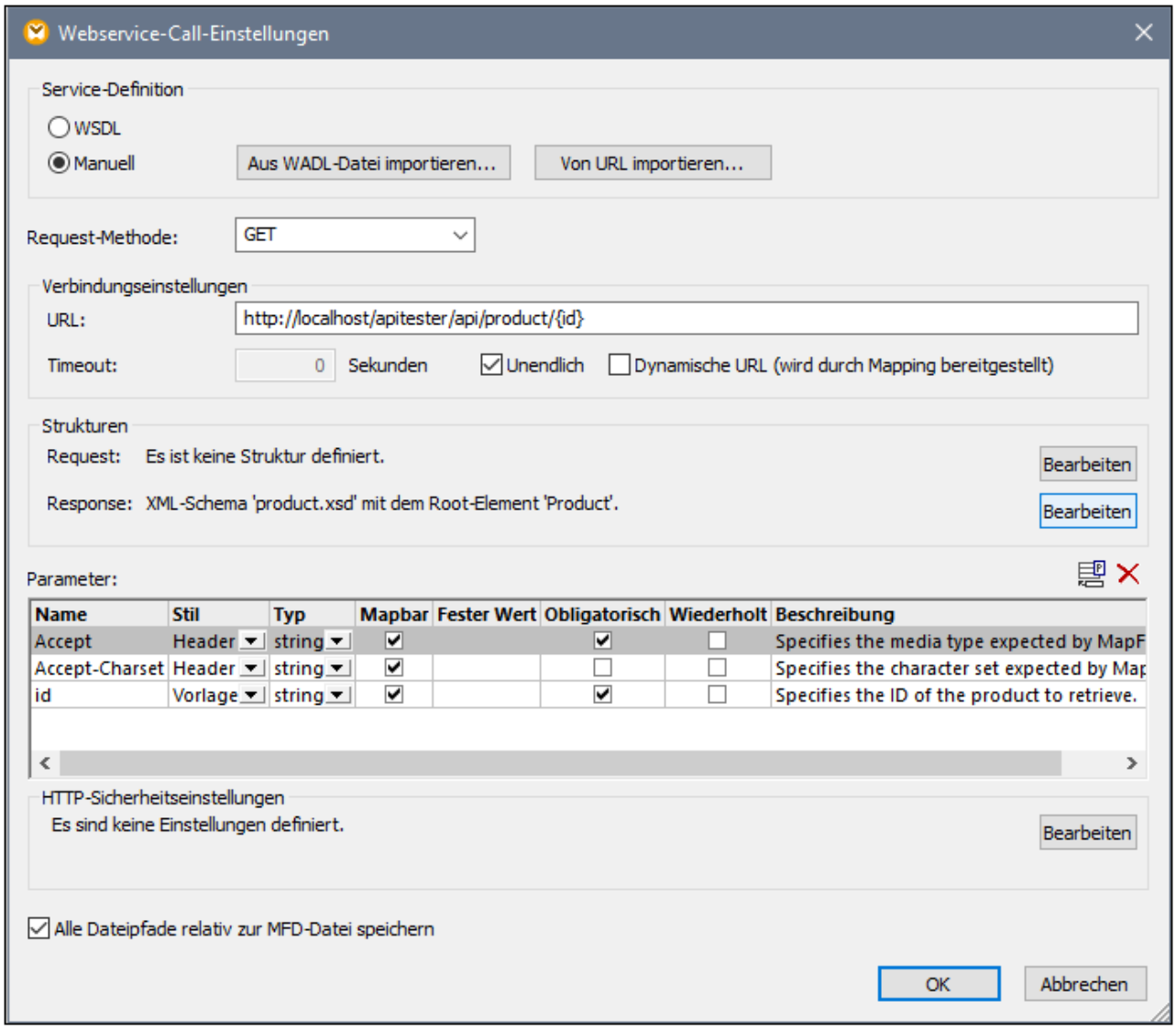
Siehe auch Beispiel: Aufruf eines SOAP-Webservice 648.

4.5.3 Webservice-Call-Einstellungen

Sie können die Einstellungen für Webservice-Aufrufe (=Webservice Calls) über das Dialogfeld "Webservice-Call-Einstellungen" konfigurieren. Um dieses Dialogfeld zu öffnen, wählen Sie eine der folgenden Methoden:

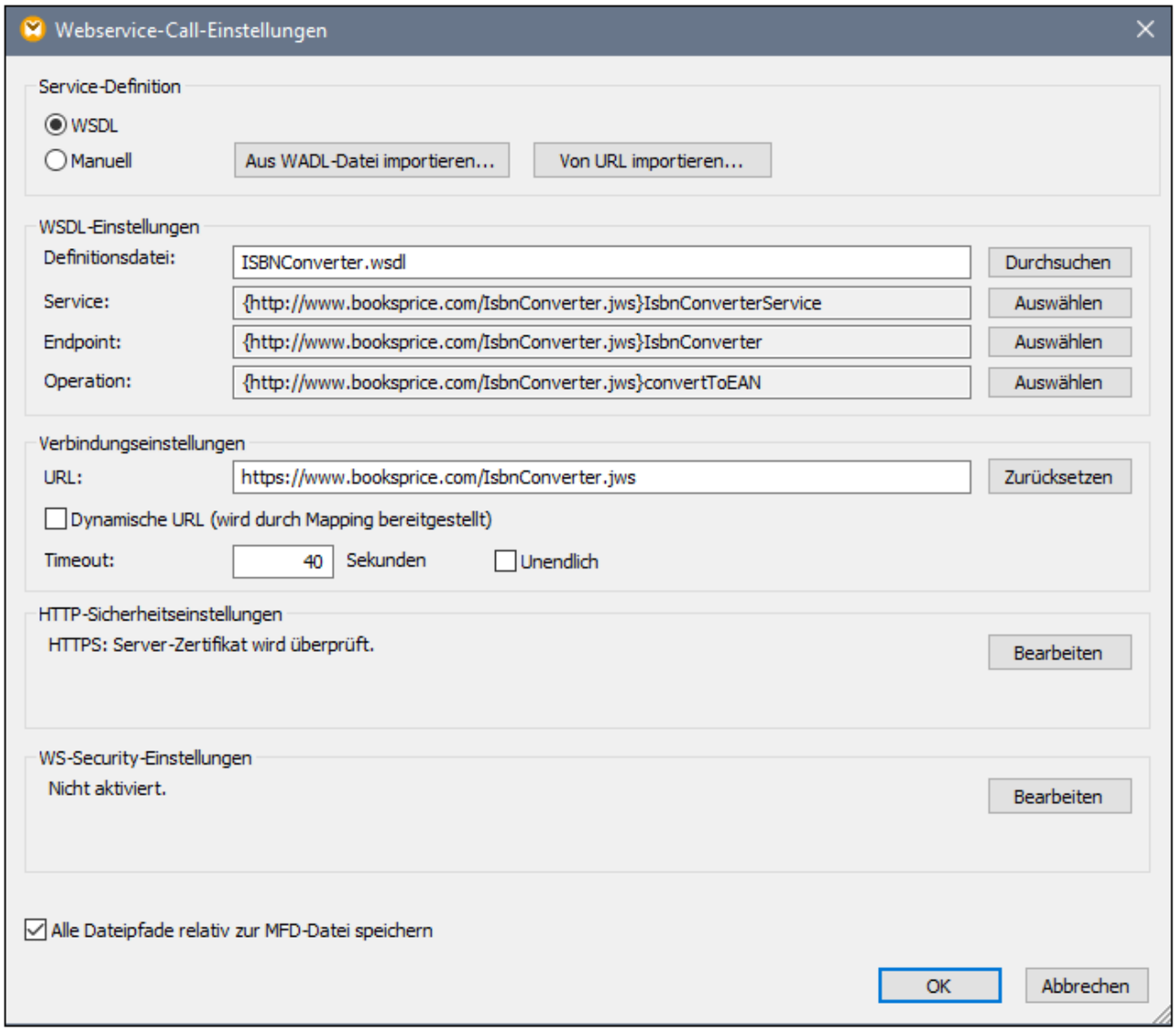
- Klicken Sie im Mapping auf die Webservice-Komponente und wählen Sie anschließend im Menü **Komponente** den Befehl **Eigenschaften**.
- Doppelklicken Sie auf den Komponententitel.
- Klicken Sie mit der rechten Maustaste auf den Komponententitel und wählen Sie **Eigenschaften**.

Die Liste der verfügbaren Einstellungen hängt davon ab, ob für die **Service-Definition** ganz oben im Dialogfeld die Option **WSDL** oder **Manuell** ausgewählt ist. Im folgenden Bild sehen Sie die Einstellungen für einen generischen HTTP-Webservice.



Im folgenden Bild sehen Sie die Einstellungen für einen WSDL-Webservice:

© 2016-2022 Altova GmbH



Weiter unten finden Sie eine Beschreibung der verfügbaren Einstellungen:

Service-Definition

Wenn die zu definierenden Einstellungen für einen WSDL-Webservice gelten, wählen Sie **WSDL**. aus. Wählen Sie andernfalls die Option **Manuell**.

Bei Auswahl von **Manuell** müssen Sie die Webservice-Einstellungen manuell in das Dialogfeld eingeben. Optional haben Sie die Möglichkeit, die Webservice-Einstellungen durch Klick auf die Schaltfläche **Aus WADL importieren** zu importieren, falls Sie die WADL-Datei des Webservice zur Verfügung haben (siehe Importieren von Webservice-Informationen aus einer WADL-Datei 623). Wenn SIe die Webservice-Parameter aus einer URL extrahieren möchten, können Sie auf **Von URL importieren** klicken (siehe Importieren von Webservice-Parametern aus einer URL 622).

WSDL-Einstellungen

Die folgenden Einstellung stehen nur für WSDL-Webservices zur Verfügung.

WSDL-Definitionen	Definiert die Web Service Definition Language (WSDL)-Datei des aufzurufenden Webservice.
Dienst	Definiert den Namen des aufzurufenden Webservice. Wenn in der WSDL-Datei mehrere Webservices definiert sind, klicken Sie auf **Auswählen**, um den gewünschten auszuwählen.
Endpoint	Definiert den Endpoint (Endpunkt) (oder den Port) des aufzurufenden Webservice. Wenn im Webservice mehrere Webservice-Endpunkte definiert sind, klicken Sie auf **Auswählen**, um den gewünschten auszuwählen.
Operation	Definiert die Operation des aufzurufenden Webservice. Wenn im Webservice mehrere Webservice-Operationen definiert sind, klicken Sie auf **Auswählen**, um die gewünschte auszuwählen.

Request-Methode

Diese Einstellung gilt nur für generische (Nicht-WSDL)-Webservices. Sie definiert, welche Request-Methode vom Service verwendet wird: z.B. GET, POST oder PUT.

Verbindungseinstellungen

Im Textfeld **URL** wird die Adresse (URL) des Webservice definiert. **Timeout** definiert das Intervall, nach dem für den Webservice-Aufruf eine Zeitüberschreitung festgestellt wird, falls vom Server keine Antwort kommt. Wählen Sie **Unendlich**, falls beim Aufruf unbeschränkt lange auf eine Antwort gewartet werden soll.

Wenn Sie die URL des Webservice über das Mapping bereitstellen möchten, aktivieren Sie das Kontrollkästchen **Dynamische URL (wird durch Mapping bereitgestellt)**. Dadurch wird in der Webservice-Konponente im Mapping ein zusätzlicher Input-Konnektor erstellt, mit dem Sie Input verbinden können, der die URL des Webservice bereitstellt.

Anmerkung: Wenn das Kontrollkästchen **Dynamische URL (wird durch Mapping bereitgestellt)** aktiviert ist, wird das Textfeld **URL** deaktiviert (ist ausgegraut) und Sie müssen die URL über das Hauptmapping bereitstellen. Für REST-Webservices bedeutet dies außerdem, dass Sie in der Tabelle "Parameter" nur Header-Parameter definieren können.

Parameter

Diese Einstellungen gelten nur für generische (nicht-WSDL)-Webservices. In der Tabelle **Parameter** werden die URL-Parameter, mit denen MapForce den Webservice aufruft, definiert (siehe Definieren von Webservice-Parametern 617).

Strukturen

Diese Einstellungen gelten nur für generische (nicht-WSDL)-Webservices. Nähere Informationen dazu finden Sie unter Definieren der Request-Struktur 624 und Definieren der Response-Struktur 626.

© 2016-2022 Altova GmbH

HTTP-Sicherheitseinstellungen

Falls eine Authentifizierung (entweder HTTP oder HTTPS) von Webservice verlangt wird, so klicken Sie auf **Bearbeiten**, um die erforderlichen Authentifizierungseinstellungen zu definieren (siehe Einstellen der HTTP-Sicherheit 635).

WS-Sicherheitseinstellungen

Diese Einstellungen gelten nur für WSDL-Webservices. Wenn für den Webservice benutzerdefinierte SOAP-Sicherheitseinstellungen erforderlich sind, klicken Sie auf **Bearbeiten**, um weitere Einstellungen zu definieren (siehe Einstellen der WS-Sicherheit 638).

Alle Dateipfade relativ zur MFD-Datei speichern

Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld "Komponenteneinstellungen" angezeigten Dateipfade relativ zum Ordner, in dem sich die MapForce Design (.mfd)-Datei befindet. Siehe auch Verwendung relativer und absoluter Pfade 84.

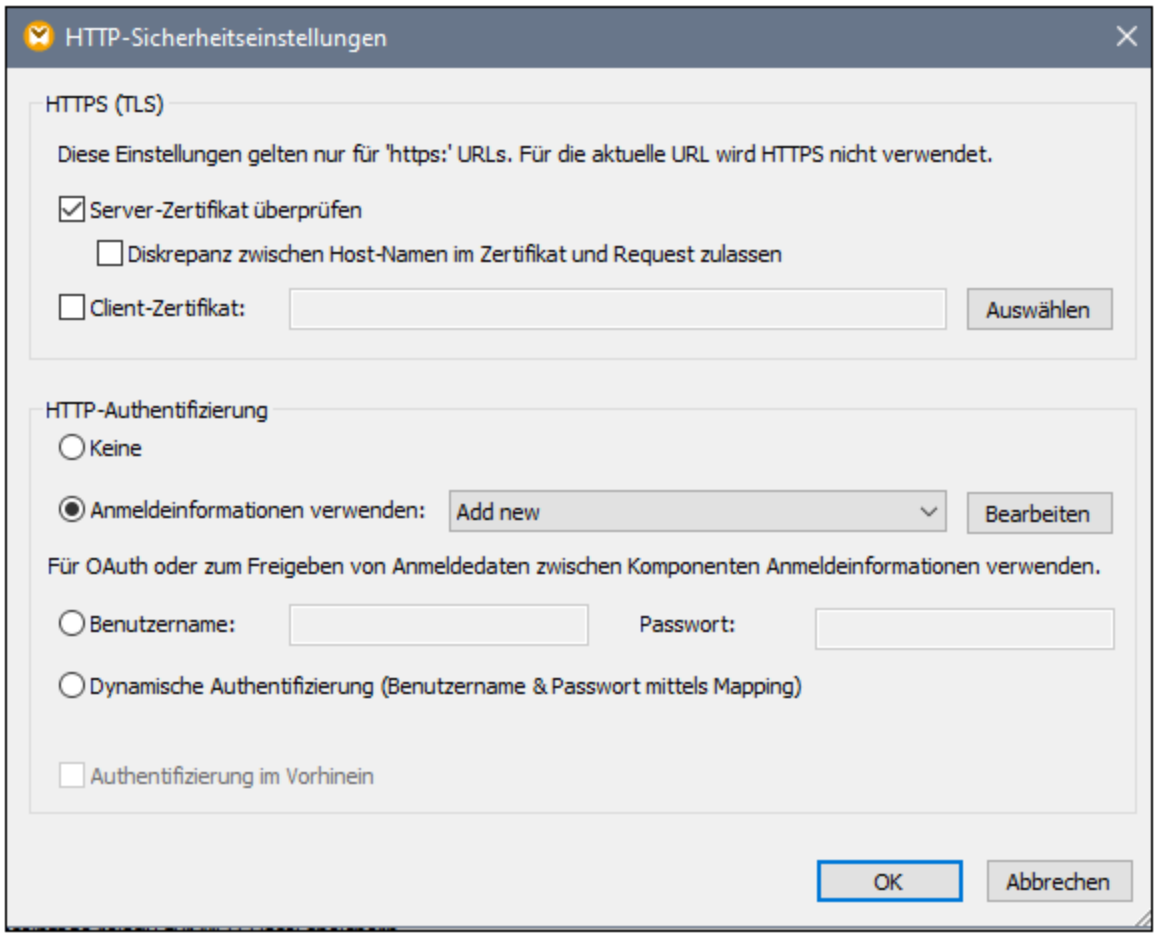
4.5.4 Einstellen der HTTP-Sicherheit

Jede Webservice-Komponente kann hinsichtlich der HTTP-Sicherheitseinstellungen für verschiedene Szenarien konfiguriert werden, z.B:

- Der Webservice wird über HTTPS aufgerufen. Dafür ist ein Client-Zertifikat erforderlich.
- Vom Server wird ein falsches Zertifikat verwendet. Sie möchten zulassen, dass der Host-Name zwischen Zertifikat und Requst nicht übereinstimmt.
- Für den Webservice ist eine einfache HTTP-Authentifizierung oder eine OAuth 2.0-Autorisierung erforderlich.

So stellen Sie die HTTP-Sicherheitseinstellungen ein:

1. Öffnen Sie das Dialogfeld "Webservice-Call-Einstellungen" (siehe Webservice-Call-Einstellungen 631).
2. Klicken Sie auf die Schaltfläche **Bearbeiten** neben den HTTP-Sicherheitseinstellungen.



Dialogfeld "HTTP-Sicherheitseinstellungen"

Die Optionsgruppe "HTTPS (TLS)" wird auf Webservices angewendet, die über HTTPS aufgerufen werden.

Server-Zertifikat überprüfen	Dieses Kontrollkästchen ist standardmäßig aktiviert, d.h. MapForce ist so konfiguriert, dass es das Zertifikat des Servers überprüft, bevor mit dem Request fortgefahren wird. Wenn diese Option aktiviert ist, schlagen der Webservice-Request (und das Mapping) fehl, wenn der Server nicht vertrauenswürdig ist oder wenn Ihr Betriebssystem nicht so konfiguriert ist, dass es den Webserver als vertrauenswürdig einstuft. Diese Option sollte nur deaktiviert werden, falls es einen gute Grund dafür gibt. Siehe auch Verwaltung digitaler Zertifikate 664.
Diskrepanz zwischen Host-Namen im Zertifikat und Request zulassen	Manchmal ist ein Zertifikat für einen bestimmten Host-Namen (z.B. *www.example.com*) auf einem anderen Host-Namen installiert (z.B. *example.com*). Aktivieren Sie dieses Kontrollkästchen, um mit der Authentifizierung fortzufahren, selbst wenn der Host-Name des

 © 2016-2022 Altova GmbH

	Zertifikats nicht mit dem vom Webservice aufgerufenen Host-Namen übereinstimmt.
Client-Zertifikat	Klicken Sie auf **Auswählen**, um ein Client-Zertifikat aus dem Zertifikatspeicher "**Aktueller Benutzer\Persönlich**" auszuwählen. Dabei wird vorausgesetzt, dass das Client-Zertifikat bereits im Zertifikatspeicher "**Aktueller Benutzer\Persönlich**" vorhanden ist; andernfalls können Sie es über das Zertifikat-Snap-In importieren (siehe Aufrufen des Zertifikatspeichers unter Windows 671). Damit das Mapping unter dem Zielpfad erfolgreich durchgeführt werden kann, muss dasselbe Zertifikat auch auf dem Zielbetriebssystem installiert sein. Nähere Informationen finden Sie unter Verwaltung digitaler Zertifikate 664.

Die Optionsgruppe "HTTP-Authentifizierung" wird angewendet, wenn für die Webservice eine HTTP-Authentifizierung erforderlich ist. Nähere Informationen zu diesen Optionen finden Sie unter Ausführen von Mappings mit Authentifizierung 1353.

Keine	Aktivieren Sie diese Option, wenn für den Webservice keinerlei Authentifizierung erforderlich ist.
Anmeldeinformationen verwenden	Aktivieren Sie diese Option, um HTTP- oder OAuth 2.0-Anmeldeinformationen zu verwenden. Nähere Informationen dazu finden Sie unter Anmeldeinformationen 1353.
Benutzername, Passwort	Aktivieren Sie diese Option, wenn der Benutzername und das Passwort in verschlüsselter Form in der Mapping-Datei gespeichert werden sollen. Diese Option wird nur aus Gründen der Rückwärtskompatibilität unterstützt, siehe Speicherung von Benutzername und Passwort 1369.
Dynamische Authentifizierung	Aktivieren Sie dieses Kontrollkästchen, wenn der Benutzername und das Passwort über das Mapping bereitgestellt werden sollen, anstatt dass Sie diese in dieses Dialogfeld eingeben. Nähere Informationen dazu finden Sie unter Dynamische Authentifizierung 1366.
Authentifizierung im Vorhinein	Aktivieren Sie dieses Kontrollkästchen, wenn der Webservice so konfiguriert ist, dass er im ersten Aufruf die Authentifizierungsdaten erwartet. Andernfalls, versucht MapForce den Webservice ohne Benutzernamen und Passwort aufzurufen und verwendet diese erst, wenn der Server die Authentifizierung anfordert (HTTP-Status 401).

4.5.5 Einstellen der WS-Sicherheit

Die WS-Sicherheitseinstellungen müssen konfiguriert werden, wenn der Webservice durch WS-Sicherheitseinstellungen geschützt ist, sodass Sie das **Benutzernamentoken** Sicherheitstoken angeben müssen.

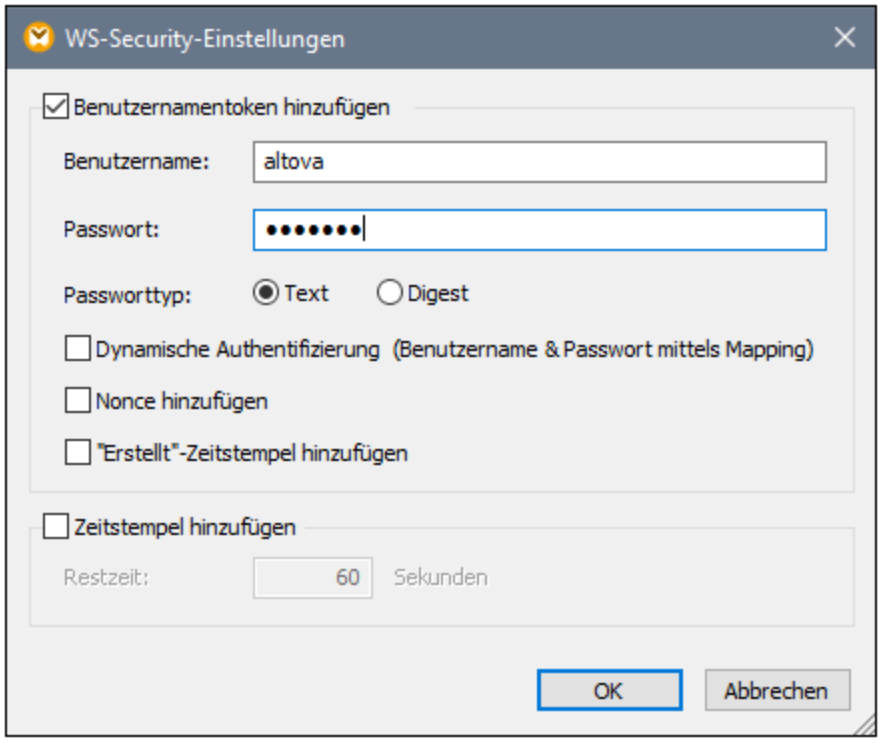
Konventionen

In diesem Kapitel werden die folgenden Abkürzungen für Webservice-Namespaces verwendet:

Präfix	Namespace
wsse	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd`
wsu	`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd`

So stellen Sie die WS-Sicherheit ein:

1. Öffnen Sie das Dialogfeld "Webservice-Call-Einstellungen" (siehe Webservice-Call-Einstellungen 631).
2. Klicken Sie auf die Schaltfläche **Bearbeiten** neben den WS-Sicherheitseinstellungen.



Dialogfeld "WS-Sicherheitseinstellungen"

© 2016-2022 Altova GmbH

Benutzernamentoken hinzufügen	Ein Benutzernamentoken ist ein optionales WS-Sicherheitselement im Header der SOAP-Nachricht. Anhand des Benutzernamentokens authentifiziert der Webserver, wer den Webservice aufruft. In der **wsse**-Spezifikation (siehe Konventionen[638]) entspricht dieses Feld: `/wsse:UsernameToken`
Benutzername	Geben Sie den im Benutzernamentoken enthaltenen Benutzernamen ein. In der **wsse**-Spezifikation (siehe Konventionen[638]) entspricht dieses Feld: `/wsse:UsernameToken/wsse:Username`
Passwort	Geben Sie den Passworttext für das Benutzernamentoken ein. In der **wsse**-Spezifikation (siehe Konventionen[638]) entspricht dieses Feld: `/wsse:UsernameToken/wsse:Password`
Passworttyp	Wählen Sie den Passworttyp für das Benutzernamentoken ein. Wählen Sie **Digest,** wenn der Webserver das Passwort in diesem Modus erwartet; wählen Sie andernfalls **Text** aus. In der **wsse**-Spezifikation (siehe Konventionen[638]) entspricht dieses Feld: `/wsse:UsernameToken/wsse:Password/@Type`
Dynamische Authentifizierung	Aktivieren Sie dieses Kontrollkästchen, wenn der Benutzername und das Passwort über das Mapping bereitgestellt werden sollen, anstatt dass Sie diese in dieses Dialogfeld eingeben.
Nonce hinzufügen	Aktivieren Sie dieses Kontrollkästchen, wenn Sie ein Nonce zum Benutzernamentoken hinzufügen möchten. Ein Nonce ist ein Zufallswert, anhand dessen jedes einzelne Benutzernamentoken eindeutig identifiziert wird. Dies ist eine zusätzliche Sicherheitsvorkehrung. Wenn Sie diese Option aktivieren, empfiehlt es sich auch, die Option **"Erstellt"-Zeitstempel hinzufügen** zu aktivieren. In der **wsse**-Spezifikation (siehe Konventionen[638]) entspricht dieses Feld: `/wsse:UsernameToken/wsse:Nonce`
"Erstellt"-Zeitstempel hinzufügen	Aktivieren Sie diese Option, um zu jedem Nonce einen Zeitstempel hinzuzufügen.

	In der **wsse**-Spezifikation (siehe Konventionen[638]) entspricht dieses Feld: `/wsse:UsernameToken/wsu:Created`
Zeitstempel hinzufügen	Aktivieren Sie dieses Kontrollkästchen, wenn Sie den TTL (time-to-live)-Wert für die SOAP-Meldung aktivieren möchten. In der **wsse**-Spezifikation (siehe Konventionen[638]) entspricht dieses Feld: `/wsu:Timestamp`
Restzeit	Geben Sie den TTL (time-to-live)-Wert, also die Restzeit für die SOAP-Nachricht ein, um das Risiko, dass jemand die Nachricht abfängt und sie erneut sendet, zu verringern.

4.5.6 Beispiel: Aufrufen eines REST-Webservice

In diesem Beispiel wird gezeigt, wie Sie einen generischen HTTP-Webservice über MapForce aufrufen. Die Beschreibung des in diesem Beispiel aufgerufenen Webservice finden Sie unter https://app.swaggerhub.com/apis/FAA/ASWS/1.1.0. Dieser Webservice gibt den aktuellen Status jedes größeren US-Flughafens als XML- oder JSON-Struktur zurück, wobei er den aus drei Buchstaben bestehenden Flughafencode als Argument erhält (z.B. "SFO", "IAD", "ABE", "DFW", usw). Die Mapping-Datei zu diesem Beispiel finden Sie unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2023\MapForceExamples\AirportStatus.mfd .

Die Response des Webservice ist in diesem Beispiel im JSON-Format und wird auf eine JSON-Datei gemappt. Daher wird das JSON-Schema der Webservice Response-Struktur benötigt. Aus Gründen der Einfachheit wurde das JSON-Schema mit XMLSpy anhand einer vom Webservice-Anbieter bereitgestellten Beispiel-Response generiert. Es steht unter dem folgenden Pfad zur Verfügung:
<Dokumente>\Altova\MapForce2023\MapForceExamples\AirportStatus.schema.json.

Für den Request-Teil rufen wir den Webservice mit einem Parameter auf, der im Request den Wert "SFO" bereitstellt. Für den Response-Teil mappen wir die Response-Daten auf eine JSON-Datei. Das Mapping besteht folglich aus drei Hauptkomponenten: dem Webservice-Aufruf, dem Input-Parameter und der JSON-Ausgabedatei.

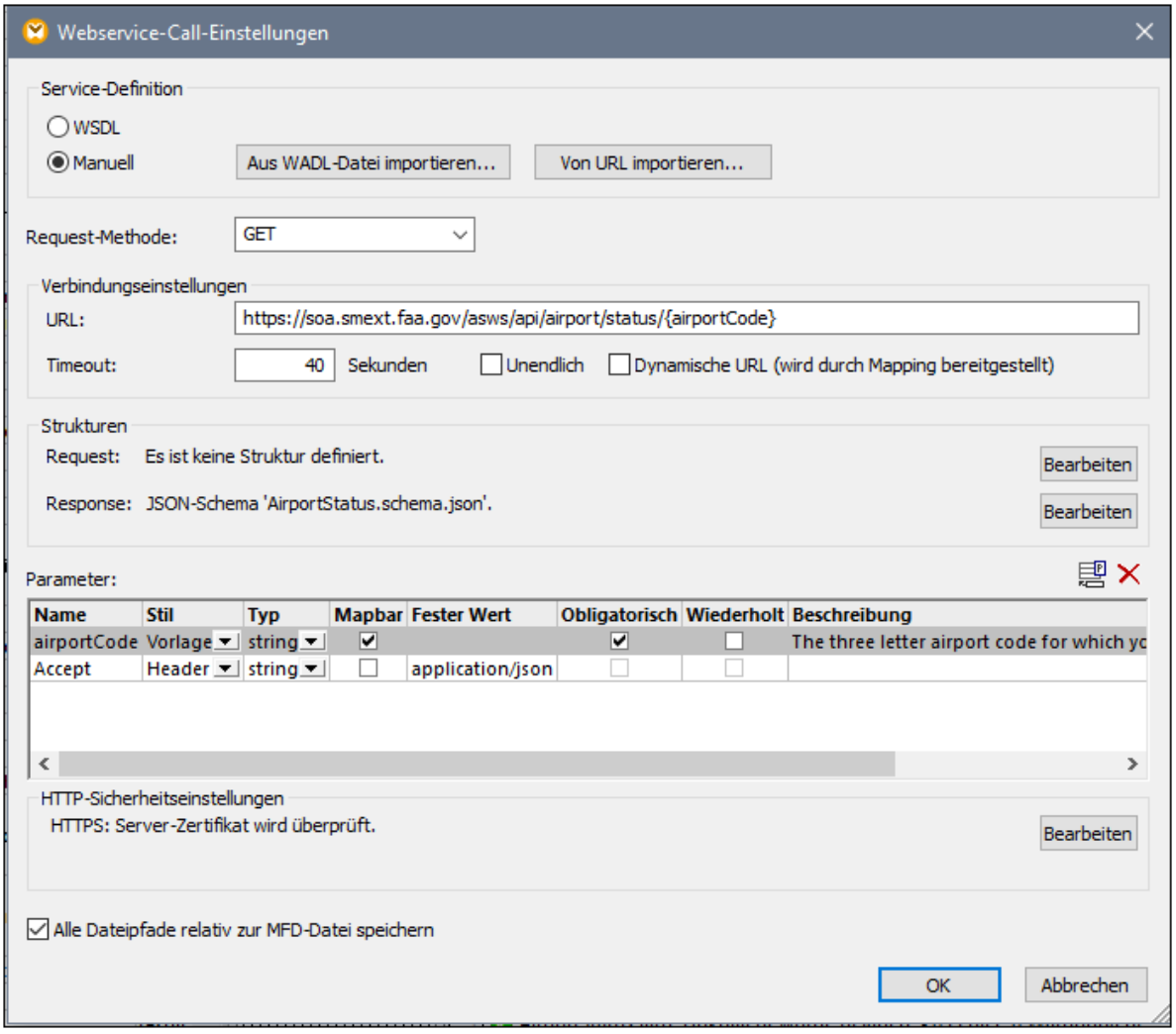
Schritt 1: Hinzufügen der Webservice-Komponente

1. Stellen Sie sicher, dass die Transformationssprache des Mappings BUILT-IN[29] ist.
2. Klicken Sie im Menü **Einfügen** auf **Webservice-Funktion**.
3. Klicken Sie unter **Service-Definition** auf **Manuell**.
4. Setzen Sie die Request-Methode auf GET und geben Sie als URL `https://soa.smext.faa.gov/asws/api/airport/status/{airportCode}` ein. Der Wert innerhalb von geschweiften Klammern ist ein Vorlagenparameter, der zur Laufzeit durch den tatsächlichen Flughafencode ersetzt wird (siehe Hinzufügen eines Webservice-Aufrufs (REST)[612]).
5. Klicken Sie auf die Schaltfläche **Parameter hinzufügen** ( ) und fügen Sie zur Tabelle "Parameter" einen neuen Parameter hinzu. Beachten Sie, dass der Name des Parameters mit dem Parameternamen in geschweiften Klammern in der URL identisch sein muss. Wählen Sie als Stil

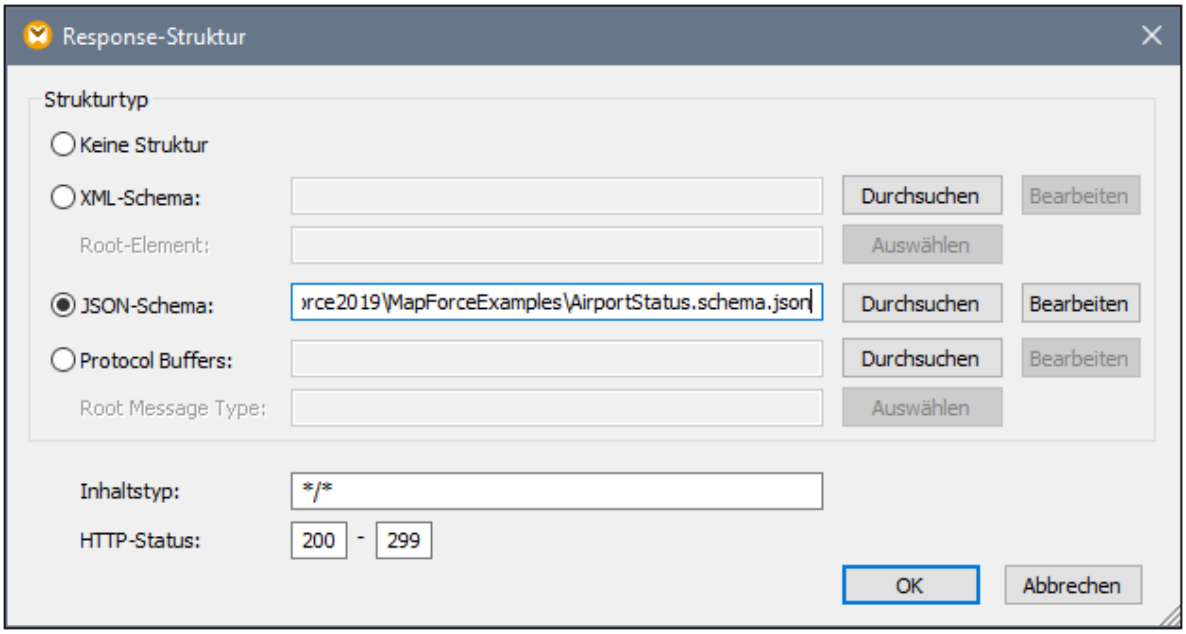
© 2016-2022 Altova GmbH

"Vorlage" und als Typ "String" aus und aktivieren Sie die Optionen "Mapbar" und "Obligatorisch". Das Feld "Beschreibung" ist optional.

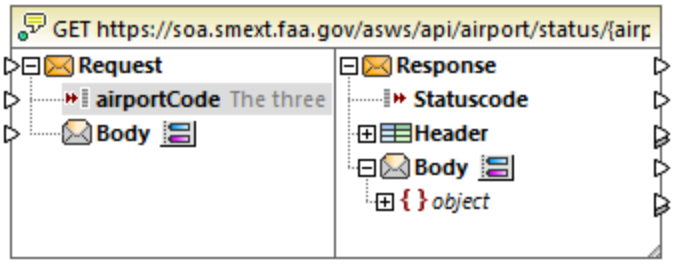
6. Fügen Sie einen Header hinzu, um dem Webserver mitzuteilen, dass der Client (d.h. MapForce) JSON-Daten in der Response erwartet. Klicken Sie dazu erneut auf die Schaltfläche **Parameter hinzufügen** ( ), nennen Sie den Parameter "Accept", wählen Sie als Stil "Header" aus und geben Sie als festen Wert "**application/json"** ein.



7. Klicken Sie unter **Response** auf die Schaltfläche **Bearbeiten** und navigieren Sie zum Schema der Webservice Response. Das Schema befindet sich unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\AirportStatus.schema.json**.



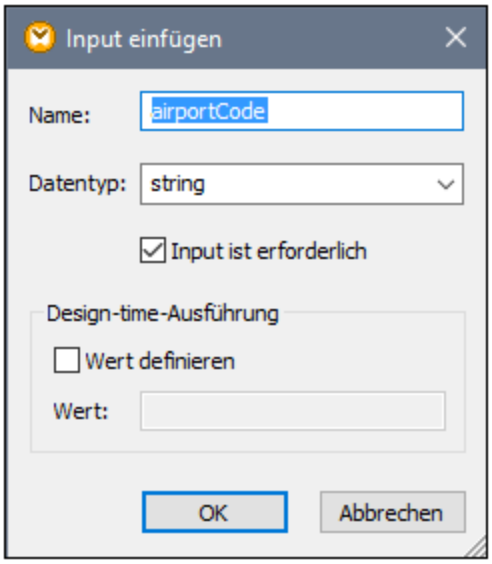
8. Klicken Sie auf **OK**. Der Webservice steht nun im Mapping-Bereich zur Verfügung.



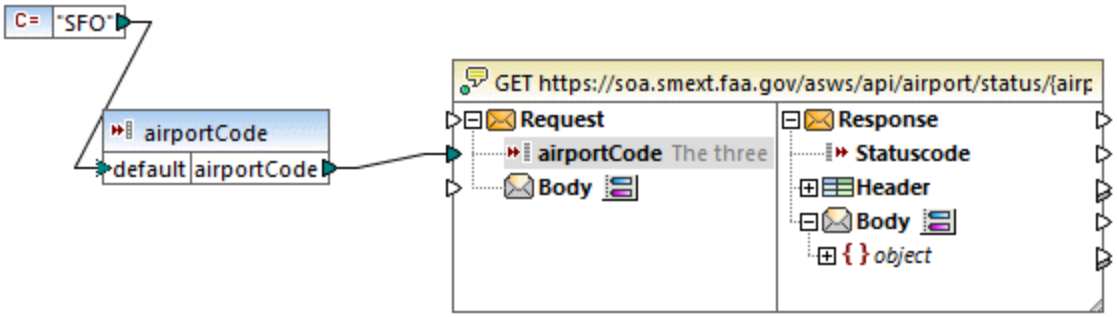
Schritt 2: Hinzufügen des Input-Parameters

1. Klicken Sie im Menü **Einfügen** auf **Input-Komponente einfügen**.
2. Geben Sie als Parameternamen "airportCode" ein und deaktivieren Sie die Option **Input ist erforderlich** (da die Input-Komponente, wie im nächsten Schritt gezeigt, von einer Konstanten bereitgestellt wird).

© 2016-2022 Altova GmbH



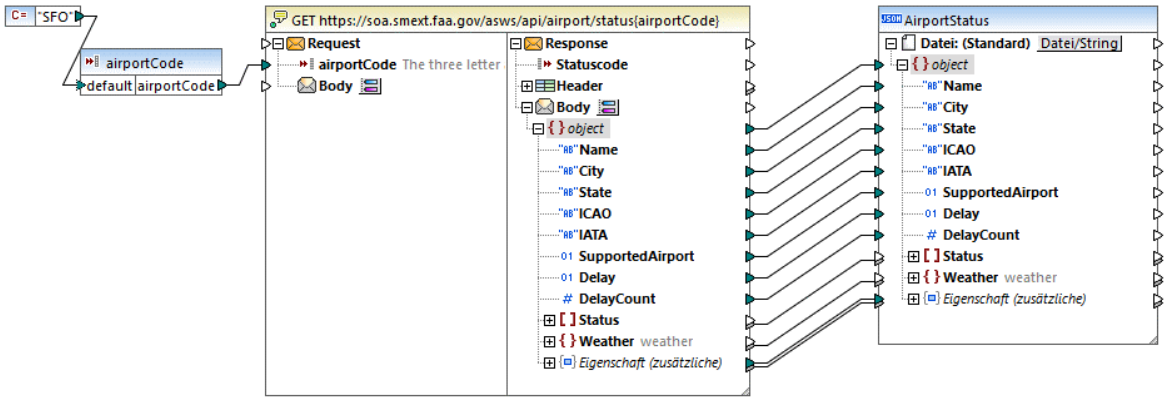
3. Fügen Sie (über den Menübefehl **Einfügen | Konstante**) eine Konstante mit dem Wert "SFO" hinzu und verbinden Sie sie mit dem Parameter-Input.
4. Verbinden Sie den Output des Parameters "airportCode" mit dem Input des Request, wie unten gezeigt.



Zu diesem Zeitpunkt ist der Requst-Teil des Webservice fertig. Zur Mapping-Laufzeit wird der Webservice mit dem von der Konstanten bereitgestellten Wert (in diesem Fall "SFO") aufgerufen.

Schritt 3: Hinzufügen der JSON-Ausgabekomponente

1. Klicken Sie im Menü **Einfügen** auf **JSON-Schema/Datei**.
2. Navigieren Sie zu **<Dokumente>\Altova\MapForce2023\MapForceExamples\AirportStatus.schema.json** und klicken Sie auf **Öffnen**.
3. Sobald Sie aufgefordert werden, eine JSON-Beispieldatei anzugeben, klicken Sie auf **Überspringen** (eine JSON-Beispieldatei wird nicht benötigt, da diese generiert wird).
4. Stellen Sie sicher, dass im Menü **Verbindung** der Menübefehl **Idente Sub-Einträge automatisch verbinden** ausgewählt ist. Wenn Sie die Option aktivieren, sparen Sie im nächsten Schritt Zeit.
5. Verbinden Sie den Node `object` des Response Body des Webservice mit dem Node `object` der JSON-Komponente, wie unten gezeigt. Da die Option **Idente Sub-Einträge automatisch verbinden** im vorigen Schritt aktiviert wurde, werden alle untergeordneten Nodes automatisch verbunden, sodass Sie die Verbindungen nicht einzeln manuell ziehen müssen.



Schritt 4: Ausführen des Mappings

Sie können den Webservice nun aufrufen. Klicken Sie auf die Schaltfläche **Ausgabe** um das Mapping auszuführen und eine Vorschau der generierten Ausgabe anzuzeigen. Wenn der Webservice-Aufruf erfolgreich ausgeführt werden konnte, wird auf dem Register **Ausgabe** die zurückgegebene JSON-Struktur angezeigt. Wenn der Aufruf nicht erfolgreich ausgeführt werden konnte, gibt MapForce den entsprechenden Fehler zurück. Wie in der Beschreibung des Webservice angegeben, wird vom Webservice der HTTP-Code 500 zurückgegeben, falls es zu einem Fehler bei der Response kommt. Wenn keine Informationen gefunden wurde, erhalten Sie unter Umständen den HTTP-Code 404.

Wenn Sie MapForce Server installiert haben, können Sie das Mapping auch zu einer Mapping-Ausführungsdatei (*.mfx)-Datei kompilieren und es über die Befehlszeile oder über die MapForce Server API auf dem Server-Rechner, auf dem MapForce Server ausgeführt wird, ausführen (siehe Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442).

Sie haben nun einen generischen HTTP-Webservice-Aufruf erstellt, in dem der Flughafenstatus mit Hilfe einer GET-Methode in Echtzeit abgerufen wird. Nähere Informationen zum Arbeiten mit generischen Webservices finden Sie unter Hinzufügen eines Webservice-Aufrufs (REST) 612.

4.5.7 Beispiel: Mappen von Daten von einem RSS Feed

In diesem Beispiel wird gezeigt, wie Sie in MapForce einen generischen HTTP-Webservice aufrufen, um Daten von einem RSS (Rich Site Summary) Feed auf Microsoft Excel zu mappen. Die Design-Datei zu diesem Beispiel finden Sie unter dem folgenden Pfad **<Dokumente>\Altova\MapForce2023\MapForceExamples\RssReader.mfd** .

Die Response des Webservice wird in diesem Beispiel vom RSS Feed des Altova Blog (https://blog.altova.com/feed) gemappt. Dazu wird das Schema der Webservice Response-Struktur benötigt, damit MapForce die Struktur der vom Feed zurückgegebenen Daten erstellen kann. Sie finden das erforderliche Schema unter dem folgenden Pfad relativ zu Ihrem Ordner "(Meine) Dokumente":**<Dokumente>\Altova\MapForce2023\MapForceExamples\rss-2-0.xsd**.

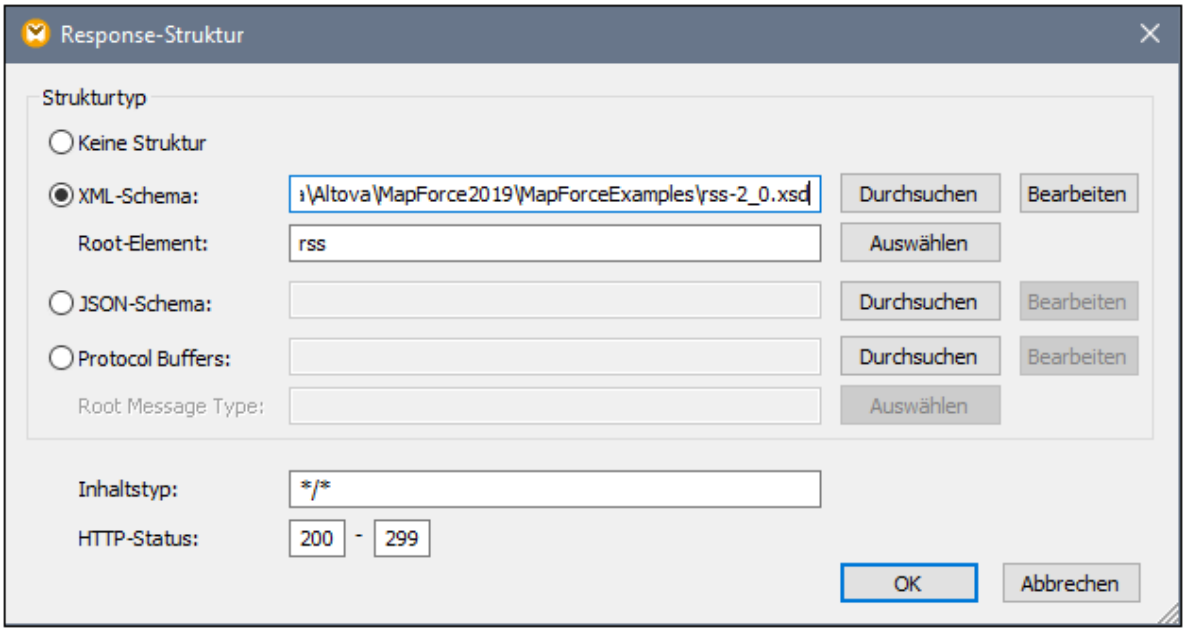
Um den RSS Feed aufzurufen, werden keine Request-Parameter benötigt, daher bleibt der Request-Teil des Webservice-Aufrufs leer. Die Response wird auf eine Microsoft Excel-Datei gemappt. Um das Datum der Veröffentlichung der einzelnen RSS-Einträge leichter lesbar zu machen, wird es in der Excel-Datei als JJJJ-MM-TT formatiert. Zu diesem Zweck werden Datenverarbeitungsfunktionen verwendet.

 © 2016-2022 Altova GmbH

Das Mapping besteht daher aus drei Hauptteilen: dem Webservice-Aufruf, der Excel-Zielkomponente und den Datenverarbeitungsfunktionen.

Schritt 1: Hinzufügen der Webservice-Komponente

1. Stellen Sie sicher, dass als Transformationssprache des Mappings BUILT-IN[29] ist.
2. Klicken Sie im Menü **Einfügen** auf **Webservice-Funktion**.
3. Klicken Sie unter **Service-Definition** auf **Manuell**.
4. Setzen Sie die Request-Methode auf GET und geben Sie als URL `http://blog.altova.com/feed` ein.
5. Klicken Sie unter **Response** auf die Schaltfläche **Bearbeiten** und navigieren Sie zum Schema der Webservice Response. Das Schema befindet sich unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\rss-2-0.xsd**.



6. Klicken Sie auf **OK**.

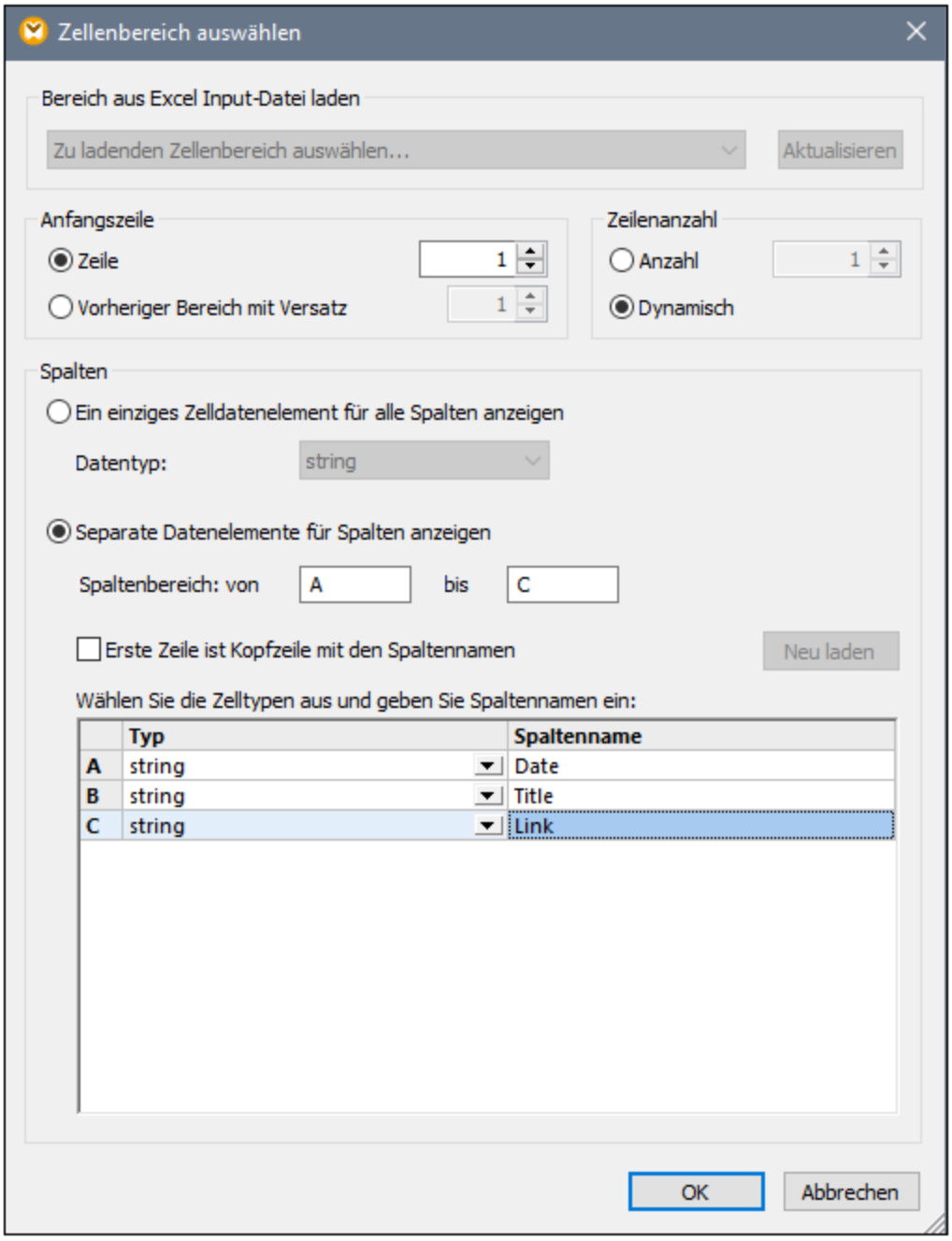
Der Webservice steht nun im Mapping-Bereich zur Verfügung und Sie können nun die Daten vom Response-Teil mappen.



Schritt 2: Hinzufügen der Excel-Zielkomponente

1. Klicken Sie im Menü **Einfügen** auf **Excel 2007+-Datei**.

2. Wenn Sie aufgefordert werden, eine Beispieldatei bereitzustellen, klicken Sie auf **Überspringen**. Die Excel-Komponente steht nun im Mapping-Bereich zur Verfügung.
3. Klicken Sie neben **Blatt 1** auf die Schaltfläche und deaktivieren Sie die Option **Arbeitsblätter nach Namen anzeigen**.
4. Klicken Sie neben **Zeile 1** auf die Schaltfläche und ändern Sie die Optionen wie unten gezeigt.



5. Verbinden Sie die folgenden Datenelemente des Response Body mit den folgenden Datenelementen der Excel-Komponente:

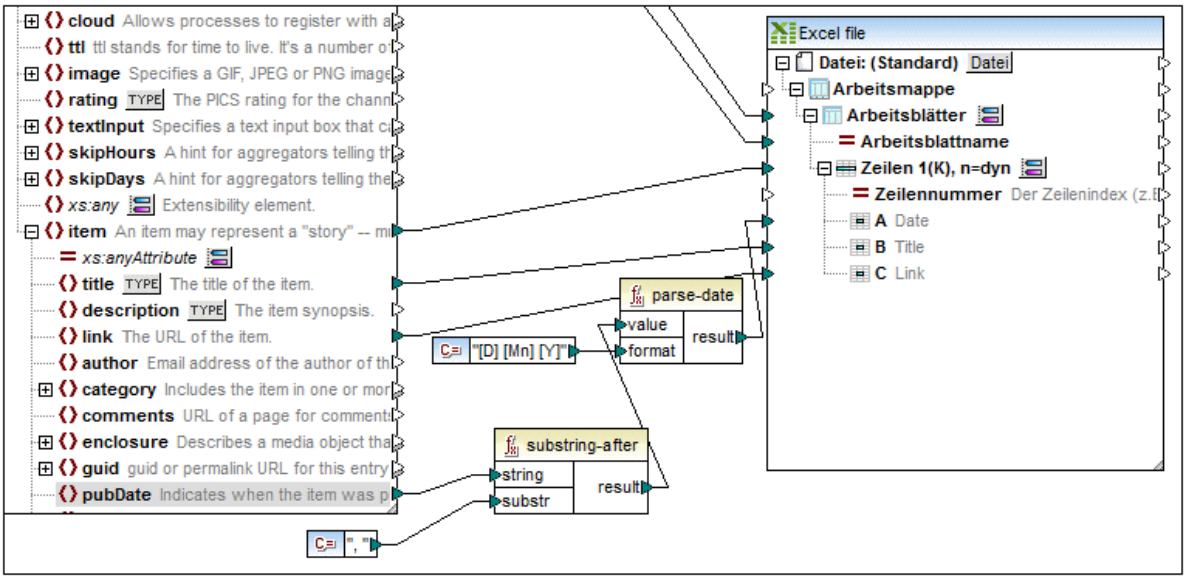
© 2016-2022 Altova GmbH

Response Body	Excel-Datei
channel	Worksheets
title	Worksheet Name
item	Rows
item/title	Row B (Title)
item/link	Row C (Link)

Zu diesem Zeitpunkt fehlt in **Zeile A (Date)** ein Input-Wert. Dieser Wert wird im nächsten Schritt hinzugefügt.

Schritt 3: Hinzufügen der Datumsverarbeitungsfunktionen

Das Datum in der Response des RSS Feed wird im RFC-822-Format angezeigt, z.B.: `Wed, 20 Jan 2016 14:49:35 +0000`. Das Datum soll in der Excel-Datei aber als `JJJJ-MM-TT` angezeigt werden. Fügen Sie zu diesem Zweck die Funktionen **`substring-after`**[1110] und **`parse-date`**[1044] aus der MapForce code-Bibliothek zum Mapping hinzu, siehe auch Hinzufügen einer Funktion zum Mapping[943].



Wie oben gezeigt, erhält die Funktion **`substring-after`**[1110] ihren Wert aus dem Node **pubDate** und gibt nur den Text nach `,` zurück. Der erzeugte Wert wird anschließend an eine weitere MapForce-Funktion, nämlich **`parse-date`**[1044] übergeben, welche ihn anhand der Maske **`[T] [Mn] [J]`** parst und den Wert als `xs:date` zurückgibt. Die Formatmaske bedeutet "Zahl für den Tag des Monats, gefolgt von einem Leerzeichen, gefolgt vom Monatsnamen beginnend mit einem Großbuchstaben, gefolgt von einem Leerzeichen, gefolgt von der vierstelligen Jahreszahl". Beispiele zur Formatierung von Datums- und Uhrzeitwerten finden Sie in der Referenz zur Funktion `parse-dateTime`[1045].

Der geparste Wert wird am Ende an die Zeile **Date** der Excel-Komponente übergeben.

Schritt 4: Ausführen des Mappings

Sie können den Webservice nun aufrufen. Klicken Sie auf die Schaltfläche **Ausgabe** um das Mapping auszuführen und eine Vorschau der generierten Ausgabe anzuzeigen. Wenn der Webservice-Aufruf erfolgreich ausgeführt werden konnte, wird auf dem Register **Ausgabe** die zurückgegebene Datenstruktur angezeigt. Wenn der Aufruf nicht erfolgreich ausgeführt werden konnte, gibt MapForce den entsprechenden Fehler zurück.

Wenn Sie MapForce Server (https://www.altova.com/de/mapforce/mapforce-server.html) installiert haben, können Sie das Mapping auch zu einer Mapping-Ausführungsdatei (*.mfx)-Datei kompilieren und es über die Befehlszeile oder über die MapForce Server API auf dem Server-Rechner, auf dem MapForce Server ausgeführt wird, ausführen (siehe Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442).

Sie haben nun einen generischen HTTP-Webservice-Aufruf erstellt, in dem Daten aus einem RSS Feed mit Hilfe einer GET-Methode abgerufen werden. Nähere Informationen zum Arbeiten mit generischen Webservices finden Sie unter Hinzufügen eines Webservice-Aufrufs (REST) 612.

4.5.8 Beispiel: Aufruf eines SOAP-Webservice

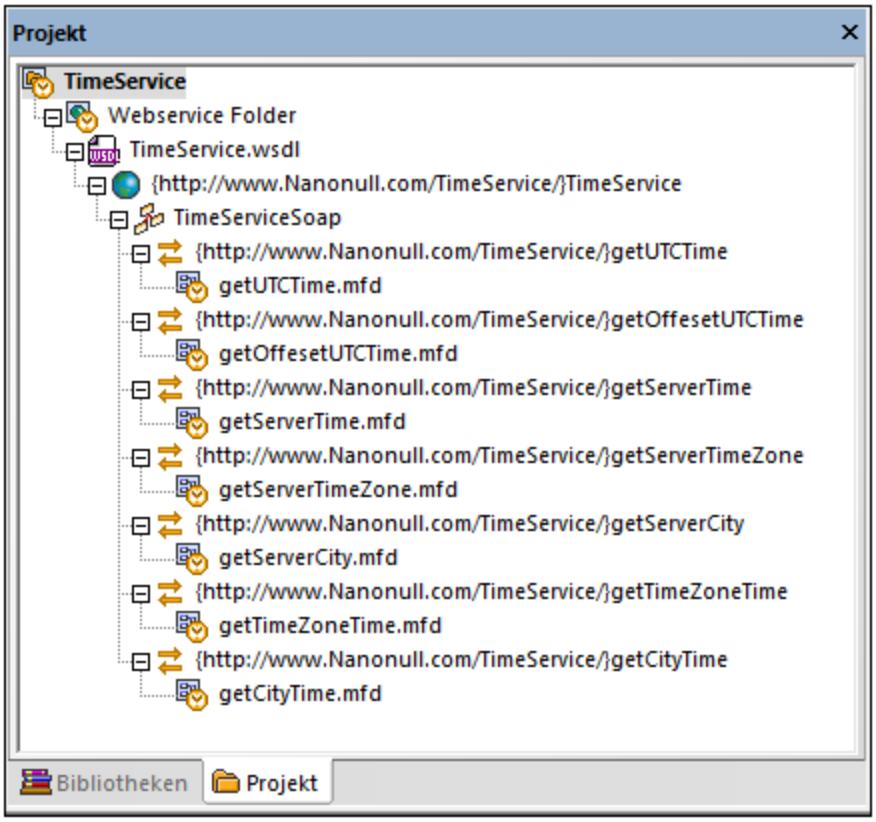
In diesem Beispiel sehen Sie, wie einen Webservice, anhand einer Konstante als Input abfragen können. Der Webservice selbst wurde mit Hilfe von MapForce implementiert. Dies dient nur zu Demonstrationszwecken - der Webservice könnte auch mit jeder anderen Technologie implementiert werden, die ein kompatibles Protokoll unterstützt.

Das unten gezeigte Mapping ist Teil des Mapping-Projekts **TimeService.mfp,** das im Ordner **<%EXFOLDER% TimeService** zur Verfügung steht. Die Projektdatei **TimeService2.mfp** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\TimeserviceWsdl2** unterstützt WSDL 2.0.

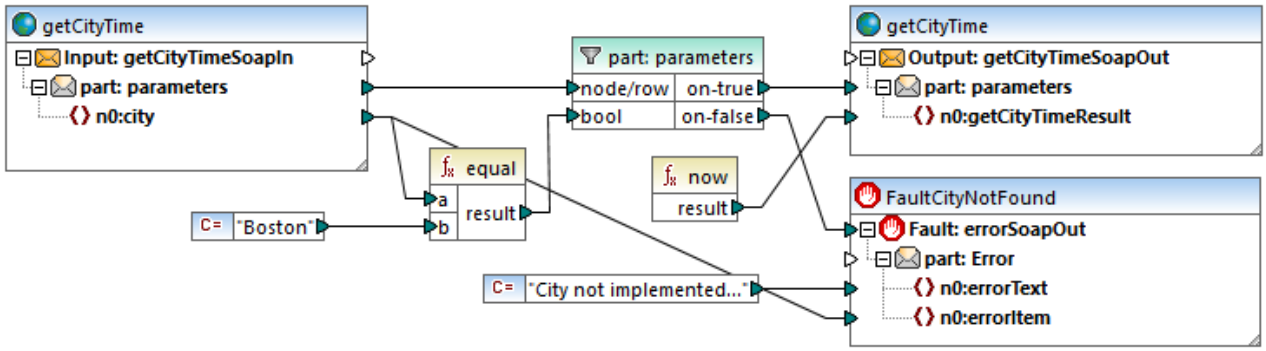
So zeigen Sie an, wie der Webservice implementiert ist:

1. Wählen Sie **Datei | Öffnen** und anschließend die Datei **TimeService.mfp** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\Timeservice**. Die mit dem Projekt verknüpften Dateien werden daraufhin im Projektfenster geladen.

 © 2016-2022 Altova GmbH



2. Doppelklicken Sie im Projektfenster auf den Eintrag **getCityTime.mfd**.

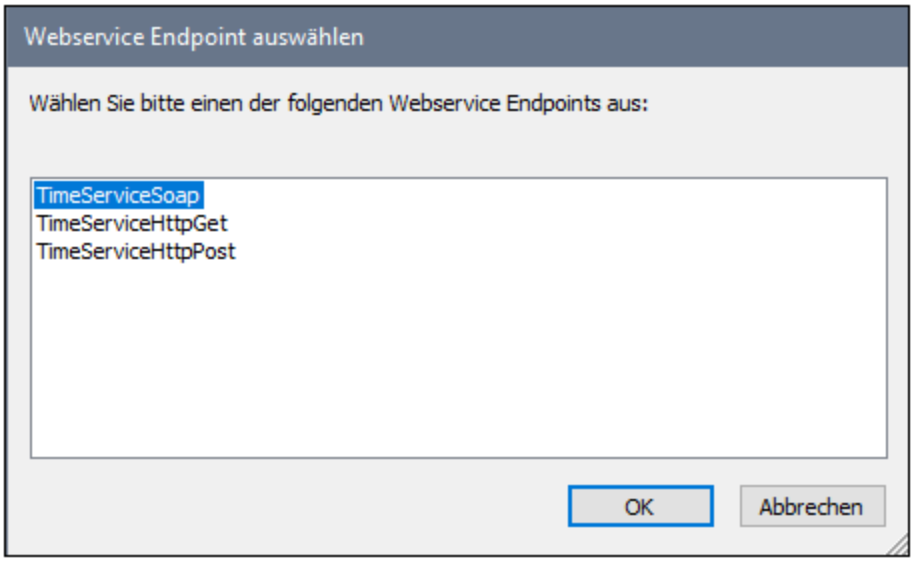


Das Mapping **getCityTime.mfd** erhält einen Stadtnamen (city name) als Input und gibt im Output die aktuelle Uhrzeit zurück. Wenn die Stadt nicht "Boston" ist, wird ein WSDL-Fehler zurückgegeben. Das Mapping erhält die Input-Daten aus der Datei **getCityTimeRequest.xml** im Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples\TimeService**. Wenn Sie auf die Komponente **getCityTimeSoapIn** doppelklicken, sehen Sie, dass diese Datei als Datenquelle definiert ist.

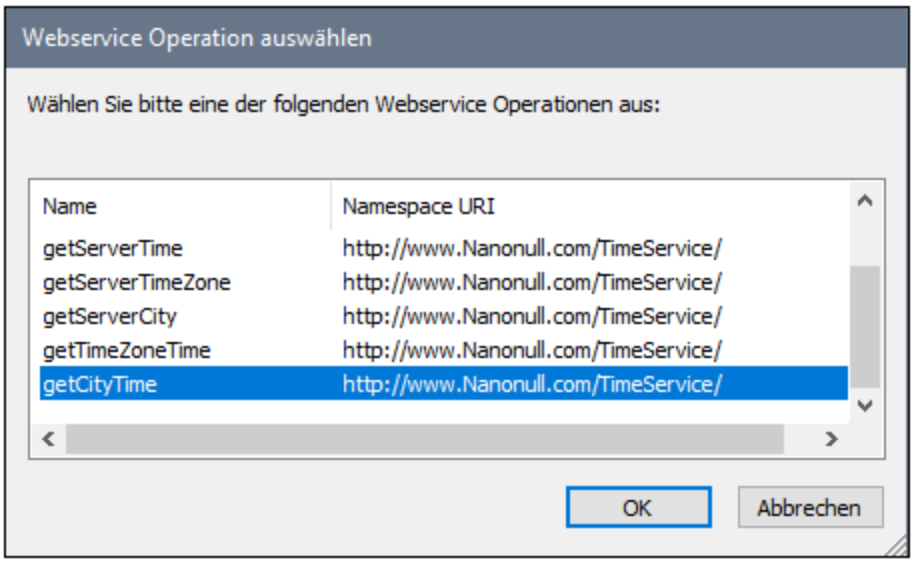
In diesem Beispiel gehen wir davon aus, dass dieses Webservice-Projekt bereits kompiliert und auf einem Webserver bereitgestellt wurde, den Sie nun in diesem Beispiel in den folgenden Schritten aufrufen werden. Sie finden dazu unter http://nanonull.com/TimeService/TimeService.asmx einen Demo Online Service. Beispiele, wie Sie solche Webservices mit MapForce erstellen finden Sie unter Beispiel: Generieren von SOAP-Webservices (C#) 1275 und Beispiel: Generieren von SOAP-Webservices (Java) 1290.

So fügen Sie den Webservice zum Mapping hinzu:

1. Wählen Sie **Datei | Neu**, klicken Sie auf das Mapping-Symbol und bestätigen Sie mit OK.
2. Wählen Sie die Menüoption **Einfügen | Webservice-Funktion...** oder klicken Sie in der Symbolleiste auf das  Symbol.
3. Klicken Sie auf die Schaltfläche **Durchsuchen**, um die WSDL-Definitionsdatei auszuwählen; wählen Sie im TimeService-Verzeichnis **TimeService.wsdl** aus und klicken Sie auf die Schaltfläche "Öffnen".
4. Wenn Sie aufgefordert werden, einen Webservice Endpoint auszuwählen, klicken Sie auf **TimeServiceSoap**.



5. Wenn Sie aufgefordert werden, eine Webservice-Operation auszuwählen, klicken Sie auf **getCityTime**.



© 2016-2022 Altova GmbH

6. Belassen Sie die Einstellungen unverändert und klicken Sie auf OK. Nähere Informationen finden Sie unter Webservice-Call-Einstellungen[631].

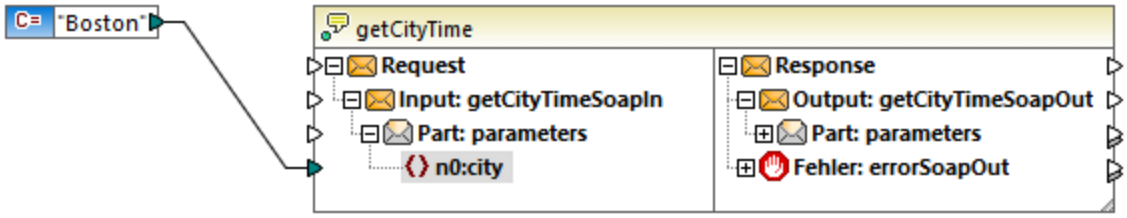
   Die Webservice-Funktion **getCityTime** wird als einzelne Komponente eingefügt. Beachten Sie: Diese Komponente steht für alle acht Komponenten, aus denen die im WSDL-Projekt gespeicherte Datei **getCityTime**.**mfd** besteht.



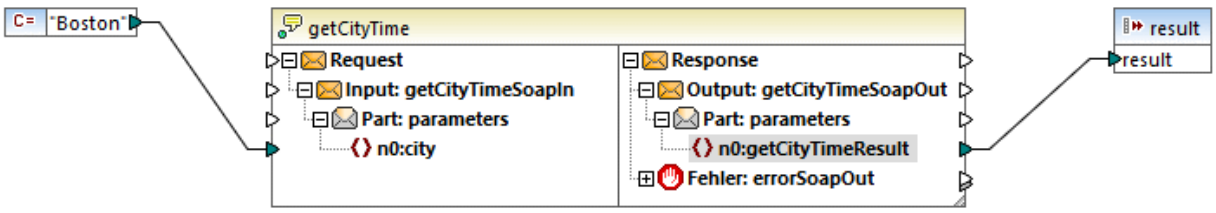
   Im linken Abschnitt der Komponente ist die Dateneingabe (SoapIn) definiert, während auf der rechten Seite die Datenausgabe (SoapOut) definiert ist. Dieser Abschnitt kann auch einen Fehler-Abschnitt enthalten, wenn in der WSDL-Datei einer definiert wurde.

So rufen Sie den Webservice auf:

1. Fügen Sie die Komponente ein, die die Input-Daten bereitstellen soll, z.B. eine Konstante, Text oder eine Schemakomponente. Fügen Sie in diesem Fall eine Konstantenkomponente ein und geben Sie als Input String "Boston" ein.



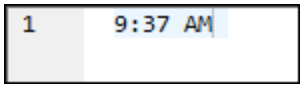
2. Verbinden Sie die Konstante mit dem Datenelement **n0:city**.
3. Fügen Sie eine einfache Textkomponente ein (Klicken Sie im Menü **Funktion** auf **Output-Komponente einfügen).**
4. Verbinden Sie **n0:getCityTimeResult** mit der Output-Komponente.



5. Klicken Sie mit der rechten Maustaste im Mapping auf die Komponente "getCityTime" und wählen Sie den Befehl **Eigenschaften**. Daraufhin wird das Dialogfeld "Webservice-Call-Einstellungen" geöffnet.

Geben Sie ins Feld **URL** `http://nanonull.com/TimeService/TimeService.asmx` ein (es handelt sich hierbei um einen mit MapForce erstellten Demo-Online-Webservice, siehe Implementieren von SOAP-Webservices 1255).

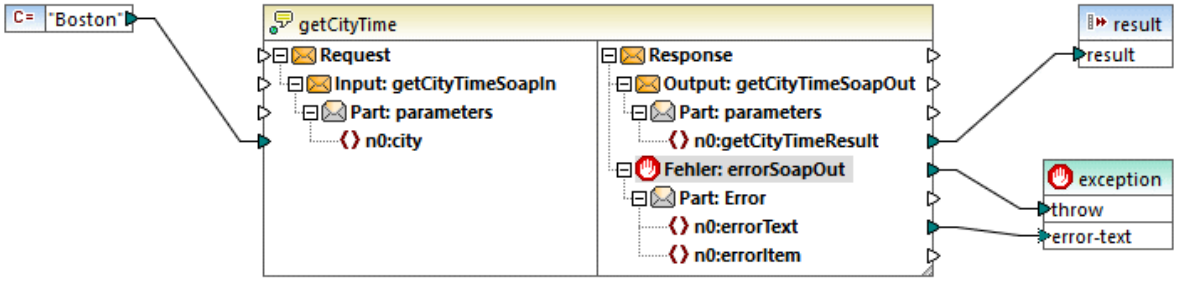
6. Klicken Sie auf das Register **Ausgabe**. Wie unten gezeigt, wird im Fenster **Ausgabe** die aktuelle Uhrzeit in Boston angezeigt.



Anmerkung: Der Eingabewert der Webservice-Funktion hat Vorrang vor der Datenquelle des ursprünglichen Mappings. So hat z.B. der Wert der Konstante "Boston" Vorrang vor der Datenquelle **getCityTimeRequest.xml** im Originalmapping.

So mappen Sie Webservice-Fehler:

1. Wählen Sie den Befehl **Einfügen | Ausnahme** oder klicken Sie in der Symbolleiste auf das Symbol "Ausnahmeereignis" .
2. Mappen Sie das Datenelement **Fault:** auf das Datenelement **throw** der Ausnahmeereigniskomponente.
3. Mappen Sie das Datenelement **n0:errorText** auf das Datenelement **error-text** der Ausnahmeereigniskomponente.



4.5.9 Beispiel: OAuth 2.0-Autorisierung

In diesem Beispiel wird gezeigt, wie Sie einen REST-Webservice, für den eine OAuth 2.0-Autorisierung erforderlich ist, aufrufen. Die Client-Applikation ist ein MapForce Mapping. Darin werden über die Google Calendar API (https://developers.google.com/calendar/) Kalenderereignisse abgerufen. Aus Gründen der Einfachheit ruft das Mapping die Kalenderinformationen im Istzustand ab und gibt nur das rohe JSON-Result aus, ohne dieses weiter zu verarbeiten.

Voraussetzungen:

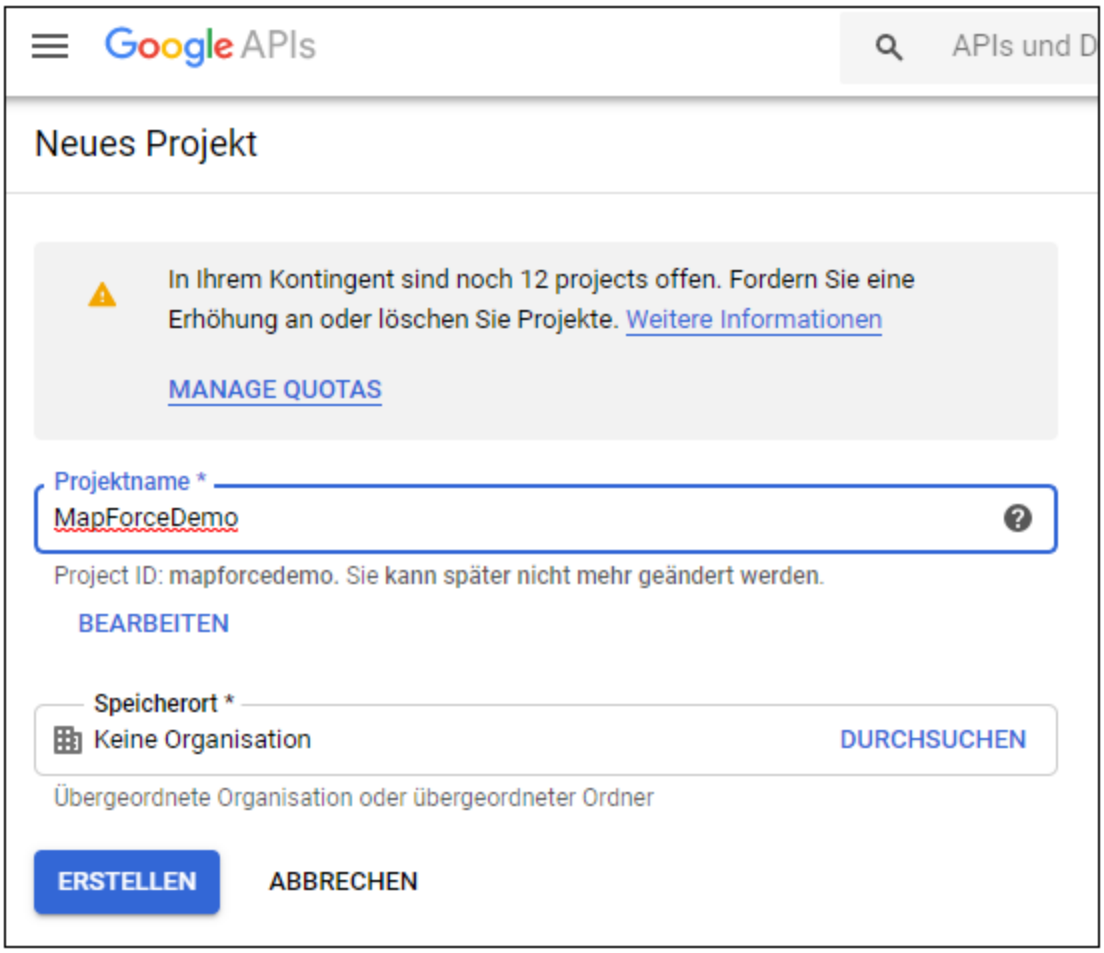
- MapForce Enterprise Edition
- Um dieses Beispiel durchspielen zu können, benötigen Sie ein Google-Konto. Wenn Sie einen anderen Webservice aufrufen möchten, fordern Sie OAuth 2.0-Anmeldeinformationen von Ihrem Webservice-Anbieter an und verwenden Sie diese in der unten stehenden Anleitung.

 © 2016-2022 Altova GmbH

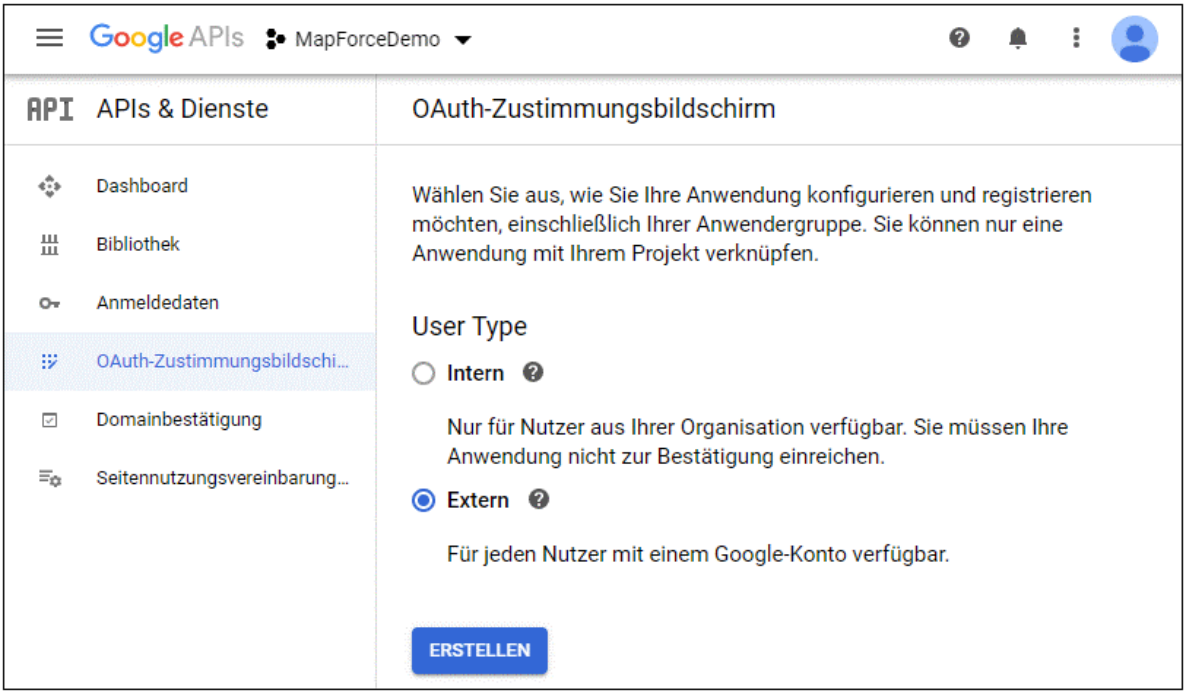
Anforderung der OAuth 2.0-Anmeldeinformationen

Wenn Sie die erforderlichen OAuth 2.0-Anmeldeinformationen für den Aufruf des Webservice bereits zur Verfügung haben, können Sie diesen Schritt überspringen. Andernfalls ist der genaue Ablauf vom Anbieter des Webservice, der von Ihrem Mapping aufgerufen wird, abhängig. Um die Google Calender API, wie in diesem Beispiel gezeigt, aufzurufen, gehen Sie folgendermaßen vor:

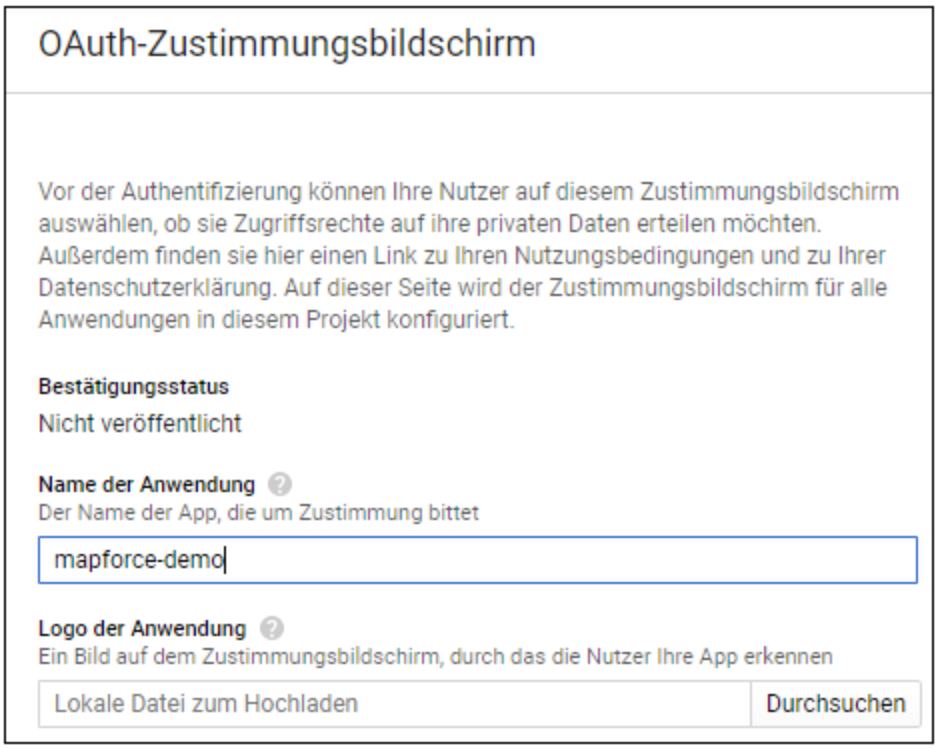
1. Melden Sie sich bei der Google API-Konsole (https://console.developers.google.com/) an.
2. Erstellen Sie ein neues Projekt.



3. Klicken Sie auf **OAuth-Zustimmungsbildschirm**.
4. Wählen Sie als Benutzertyp **Extern**, es sei denn, Sie haben ein G Suite-Konto, über das Sie nur Benutzern in Ihrem Unternehmen API-Zugriff einräumen können.

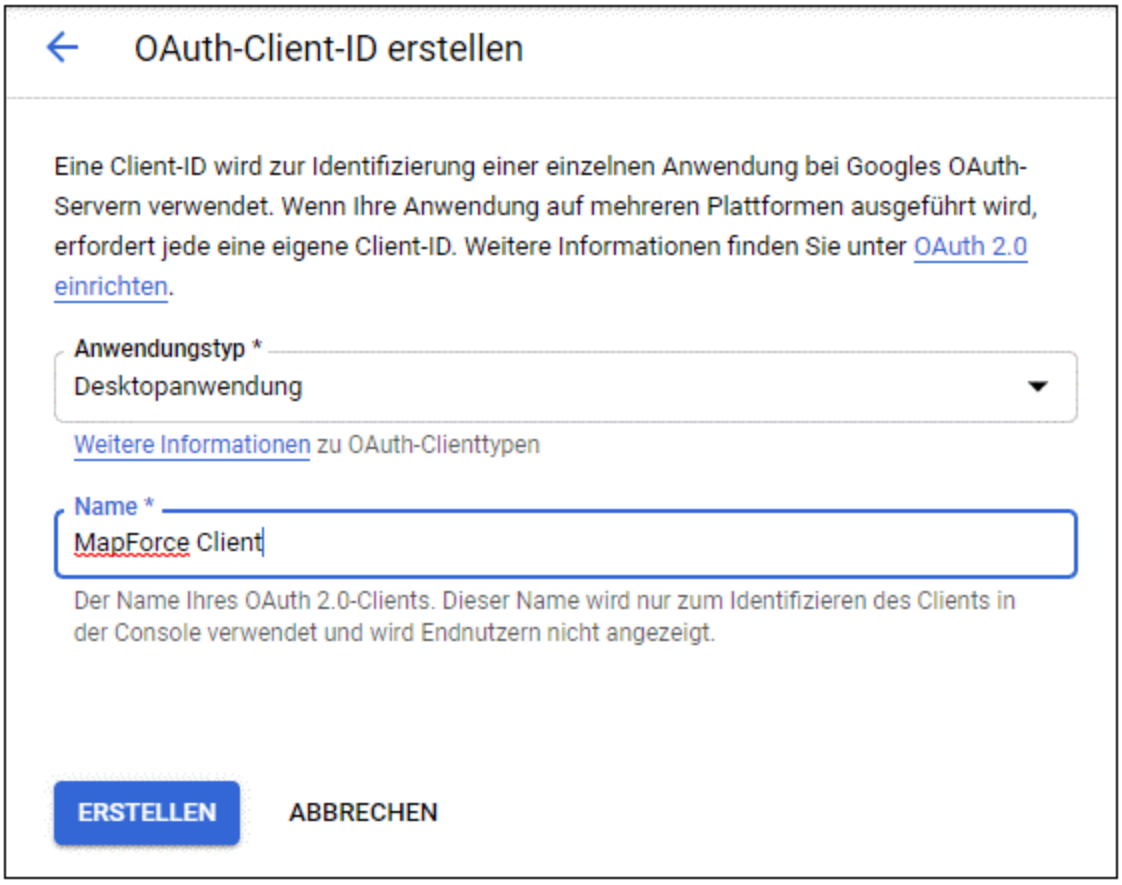


5. Geben Sie "mapforce-demo" als Applikationsnamen ein und speichern Sie die Einstellungen.

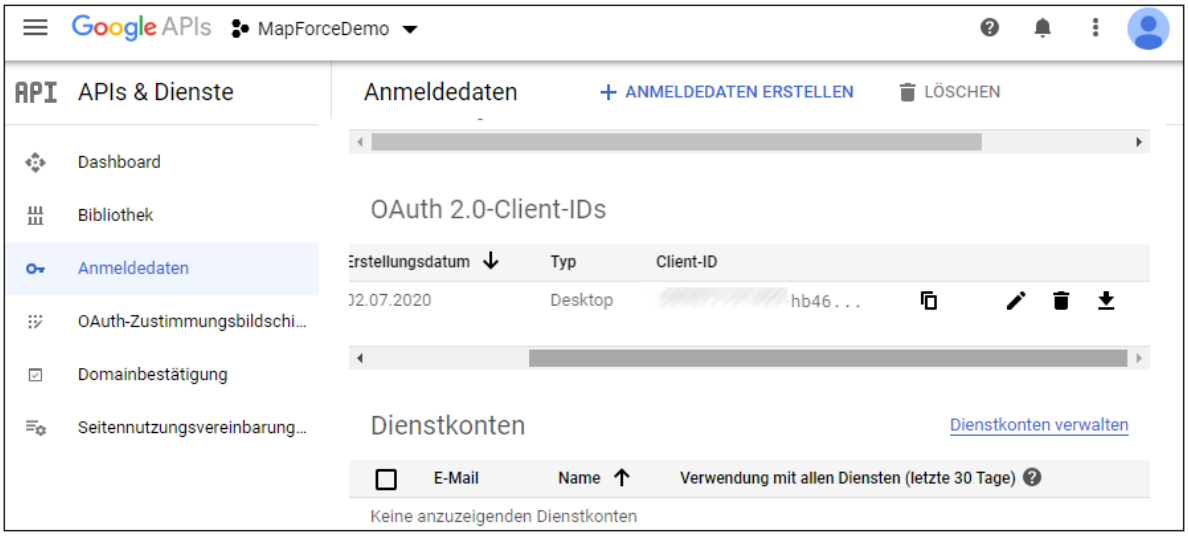


6. Klicken Sie auf **Anmeldedaten erstellen** wählen Sie anschließend **OAuth-Client-ID** aus.
7. Geben Sie als Applikationstyp **Desktopanwendung** und als Client-Namen "MapForce Client" ein.

© 2016-2022 Altova GmbH



8. Klicken Sie auf **Erstellen**. Daraufhin wird die Client-ID erstellt und steht auf der Seite **Anmeldedaten** zur Verfügung.



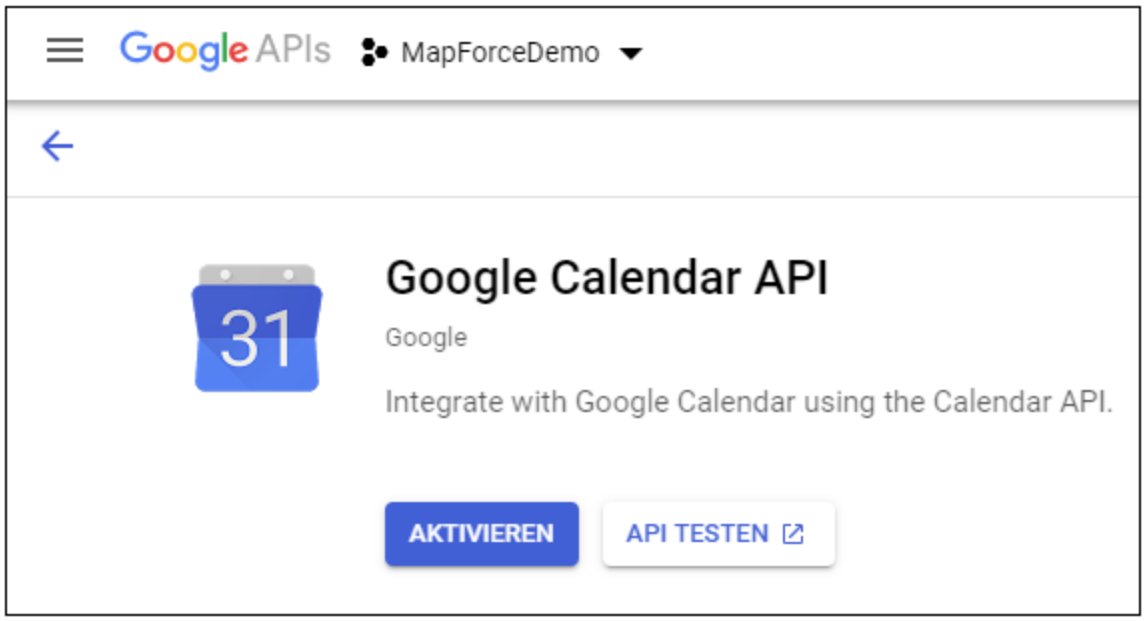
9. Klicken Sie auf , um die OAuth 2.0-Autorisierungsinformationen als JSON-Datei herunterzuladen.

Sie haben nun die folgenden OAuth 2.0-Autorisierungsinformationen von der Google Console API erhalten:

1. Autorisierungsendpunkt
2. Token-Endpunkt
3. Client ID
4. Client Secret

Aktivieren der Google Calender API

Um Aufrufe von Clients zu akzeptieren, muss die in diesem Beispiel verwendete Google Calendar API aktiviert werden. Klicken Sie in der Google API-Konsole auf **Bibliothek**, suchen Sie die Google Calendar API und aktivieren Sie sie:



In diesem Beispiel werden wir die **list**-Methode der **Events** Entity aufrufen. Ausführliche Informationen über diese API-Methode finden Sie unter https://developers.google.com/calendar/v3/reference/events/list. Beachten Sie die folgenden wichtigen Punkte:

1. Wie in der Dokumentation beschrieben, muss die Methode durch Senden eines GET Request an `https://www.googleapis.com/calendar/v3/calendars/calendarId/events` aufgerufen werden, wobei `calendarId` der Identifier eines Google Calendar ist. Der Request-Parameter **calendarId** wird in der Folge später über MapForce konfiguriert.
2. Für den Aufruf der API-Methode ist mindestens einer der folgenden Geltungsbereiche erforderlich:

   - `https://www.googleapis.com/auth/calendar.readonly`
   - `https://www.googleapis.com/auth/calendar`
   - `https://www.googleapis.com/auth/calendar.events.readonly`
   - `https://www.googleapis.com/auth/calendar.events`

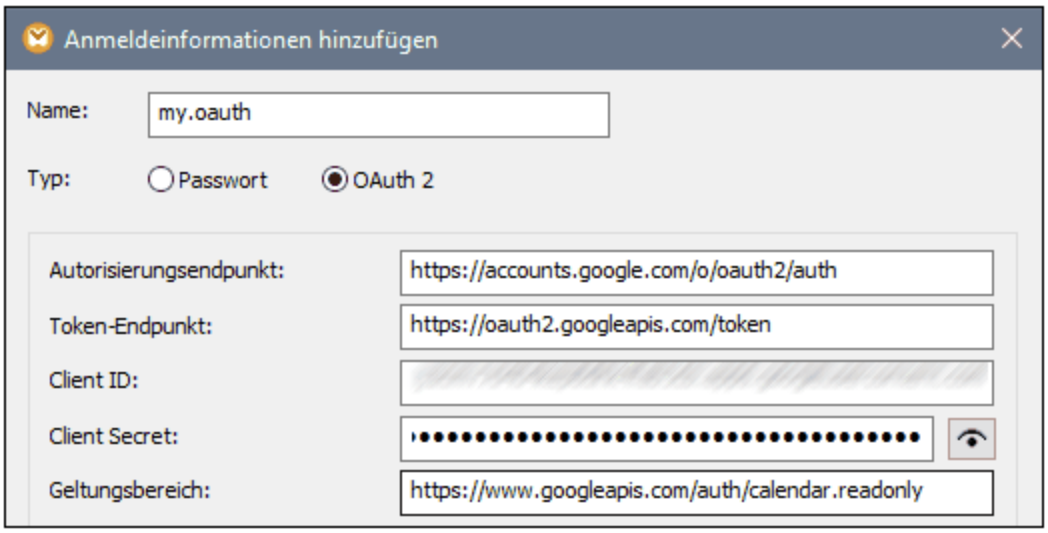
   Ihr Mapping muss während der OAuth 2-Autorisierung einen dieser Geltungsbereiche angeben. Auch dieser wird in einem späteren Schritt konfiguriert. Für dieses Beispiel ist der erste Geltungsbereich "read-only" ausreichend.

© 2016-2022 Altova GmbH

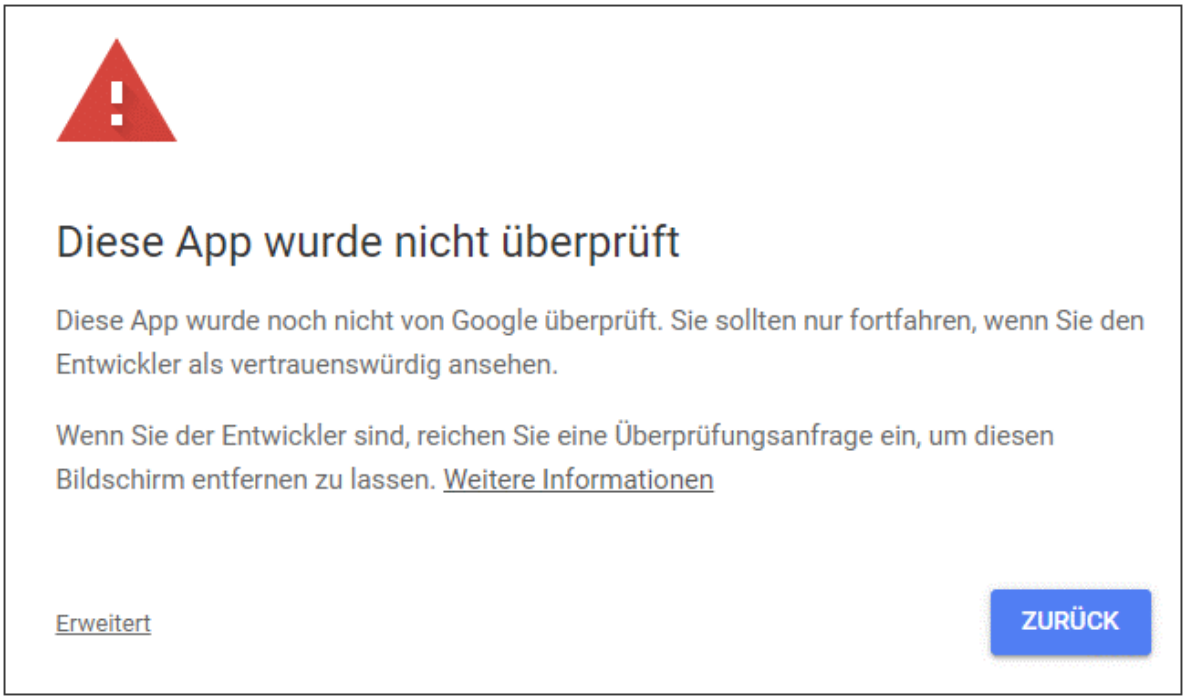
Anforderung eines Autorisierungs-Token

Um in MapForce eine Vorschau auf das Mapping anzeigen zu können, benötigen Sie die OAuth 2.0-Autorisierungsinformationen für das Mapping und müssen, wie unten gezeigt, ein Autorisierungs-Token anfordern.

1. Klicken Sie in MapForce mit der rechten Maustaste in einen leeren Bereich des Mappings und wählen Sie im Kontextmenü den Befehl **Anmeldeinformationen-Manager öffnen**.
2. Klicken Sie auf **Anmeldeinformationen hinzufügen**.
3. Geben Sie einen Namen ein (in diesem Beispiel "my.oauth") und wählen Sie als Typ **OAuth 2** aus.
4. Füllen Sie die Textfelder **Autorisierungsendpunkt**, **Token-Endpunkt**, **Client ID**, **Client Secret** mit den entsprechenden Werten aus der zuvor heruntergeladenen JSON-Datei aus.
5. Geben Sie `https://www.googleapis.com/auth/calendar.readonly` in das Textfeld **Geltungsbereich** ein.
6. Belassen Sie alle anderen Einstellungen unverändert.

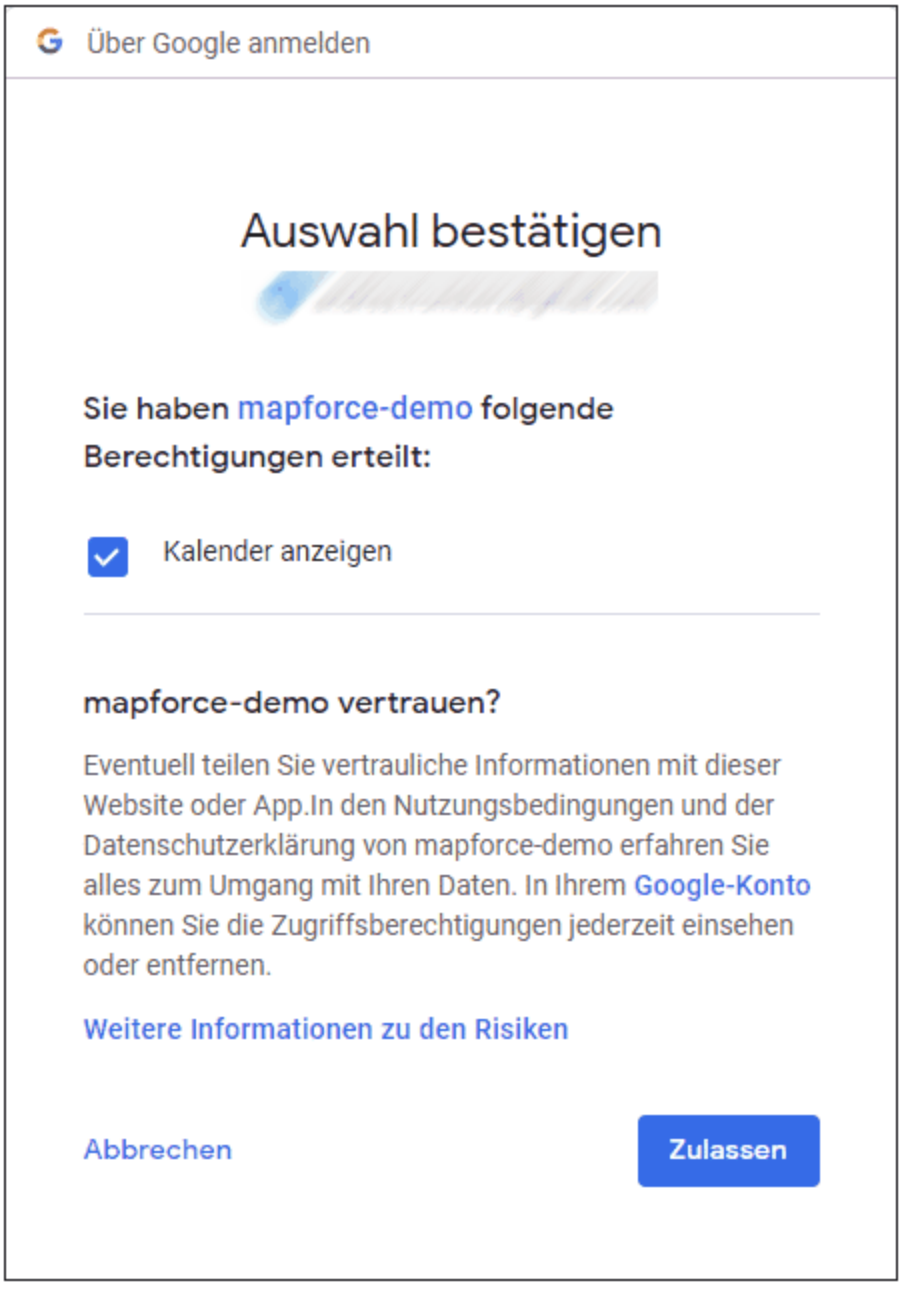


7. Klicken Sie auf **Access Token anfordern**, um das Token vom Autorisierungsserver (in diesem Beispiel Google) anzufordern. Daraufhin wird ein Browser-Fenster geöffnet, in dem Sie gebeten werden, sich mit Ihrem Google-Konto zu verbinden.
8. Melden Sie sich bei Ihrem Google-Konto an. Da Sie noch keine App-Überprüfungsanforderungen an Google gesendet haben, wird die folgende Seite angezeigt.



9. Klicken Sie auf **Erweitert** und anschließend auf **mapforce-demo öffnen (unsicher)**.

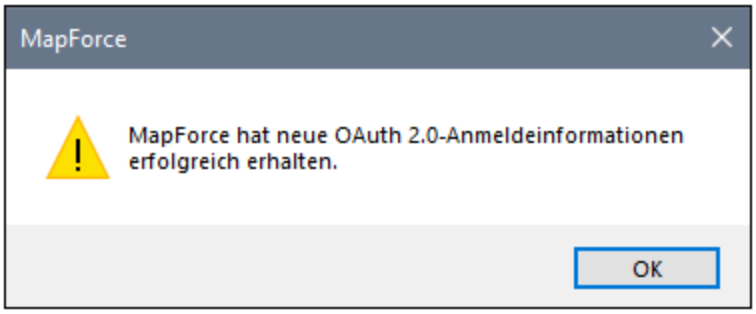
© 2016-2022 Altova GmbH



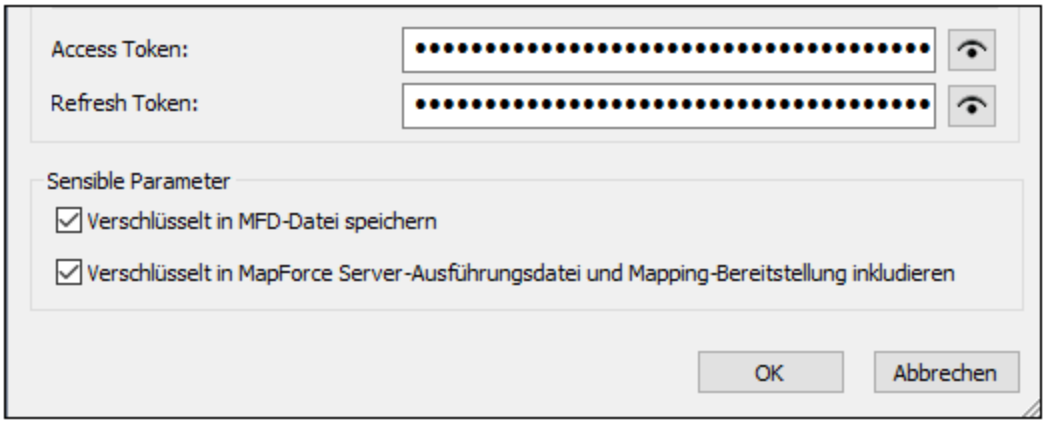
10. Klicken Sie auf **Zulassen**. Daraufhin wird eine Bestätigung im Browser angezeigt.



MapForce zeigt ebenfalls eine Meldung an, dass der OAuth 2.0-Autorisierungscode erfolgreich abgerufen wurde.



11. Klicken Sie auf **OK.** Beachten Sie, dass die Felder **Access Token** und **Refresh Token** nun ausgefüllt wurden.



12. Speichern Sie das Mapping unter dem Namen **GetCalendarEvents.mfd**.

> In diesem Tutorial wurde das Kontrollkästchen **Verschlüsselt in MFD-Datei speichern** im Dialogfeld "Anmeldeinformationen bearbeiten" aktiviert. Daher werden die sensiblen Daten in den Feldern **Client Secret**, **Autorisierungs-Token** und **Refresh Token** beim Speichern des Mappings in verschlüsselter Form in der Mapping-Design-Datei (.mfd) gespeichert.

Beachten Sie, dass das Autorisierungs-Token nach einem gewissen Zeitraum abläuft. Wenn dies passiert, können Sie das Mapping nicht mehr ausführen (derzeit wurde noch kein Mapping erstellt, dies geschieht in einem späteren Schritt). Wenn Sie manuell einen neuen Autorisierungscode anfordern möchten, klicken Sie auf **Access Token anfordern** gehen Sie vor, wie unten beschrieben.

Erstellen des Webservice-Aufrufs

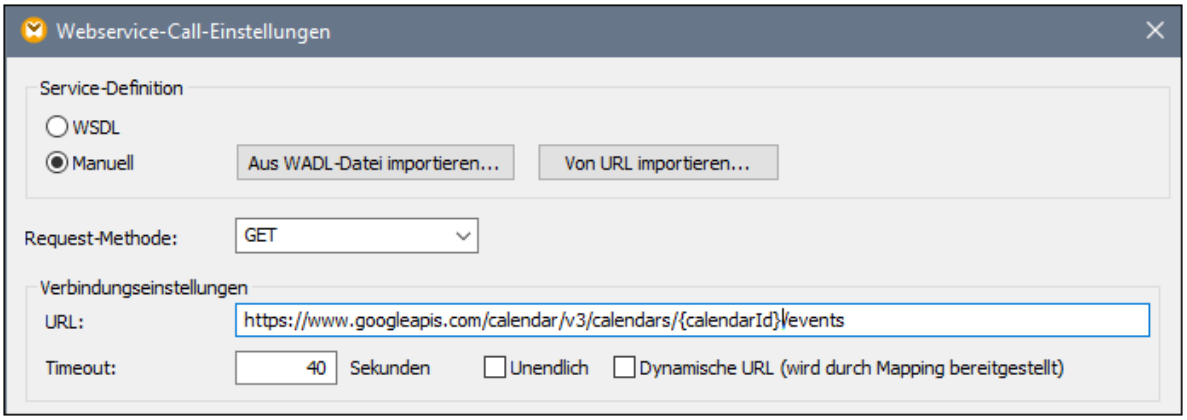
Das bisher erstellte Mapping **GetCalendarEvents.mfd** hat noch keine Funktionen. Es enthält bisher nur OAuth 2.0-Anmeldeinformationen, die den Zugriff auf die Google Calender API aktivieren.

Wir werden nun den Webservice-Aufruf in MapForce folgendermaßen erstellen:

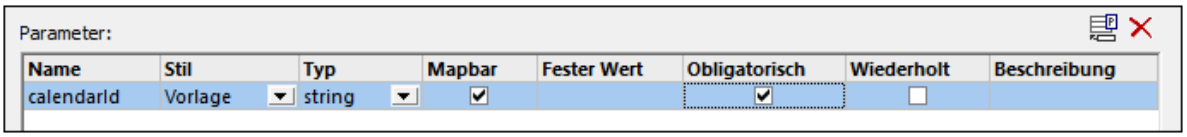
1. Öffnen Sie das Mapping **GetCalendarEvents.mfd**.
2. Klicken Sie im Menü **Einfügen** auf **Webservice-Funktion**. Daraufhin wird das Dialogfeld "Webservice-Call-Einstellungen" aufgerufen.
3. Klicken Sie auf **Manuell**.

© 2016-2022 Altova GmbH

4. Wählen Sie als Request-Methode **GET** aus und geben Sie die URL zu dem im vorigen Schritt erwähnten Webservice ein:
   `https://www.googleapis.com/calendar/v3/calendars/calendarId/events`.
5. Da **calendarId** ein Platzhalter ist, der in Form eines Parameters angegeben werden muss, setzen Sie ihn in geschweifte Klammern, wie unten gezeigt.

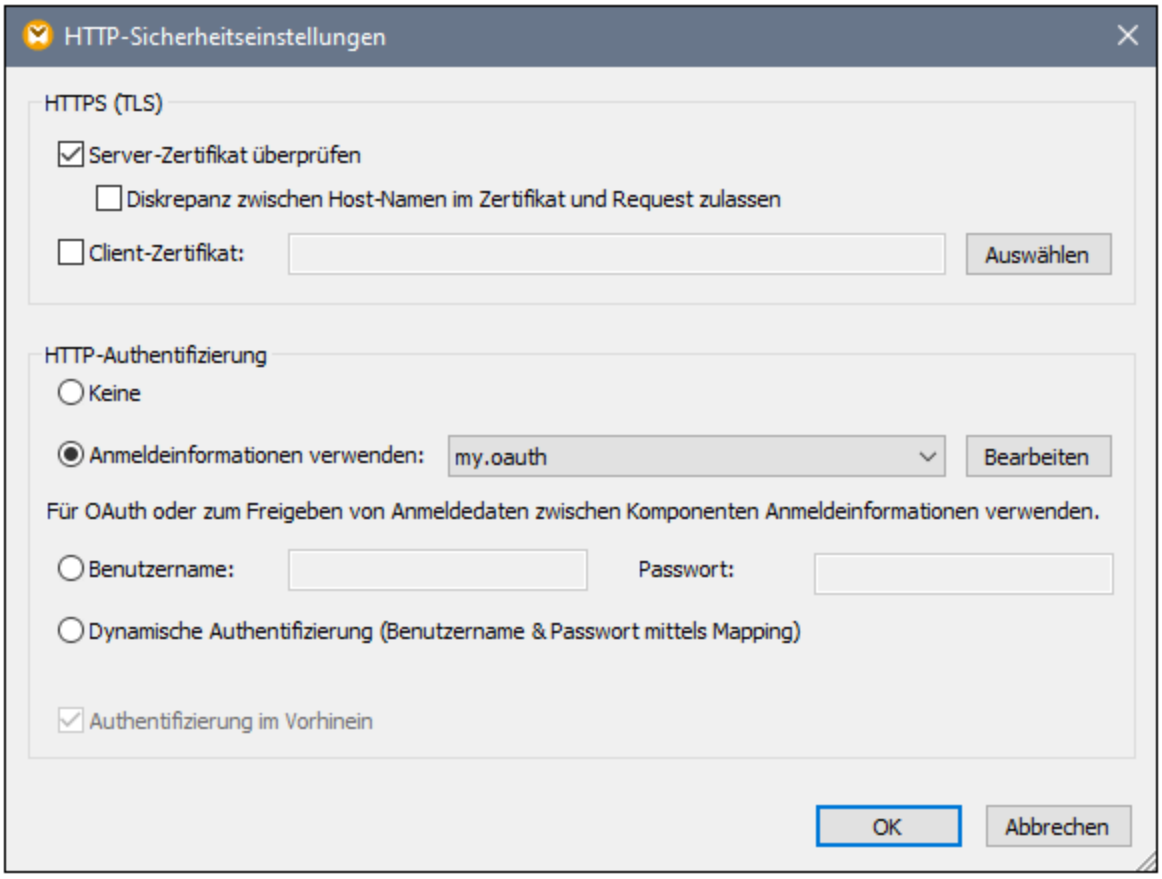


6. Klicken Sie auf die Schaltfläche **Parameter hinzufügen** und definieren Sie die folgenden Parameterinformationen:

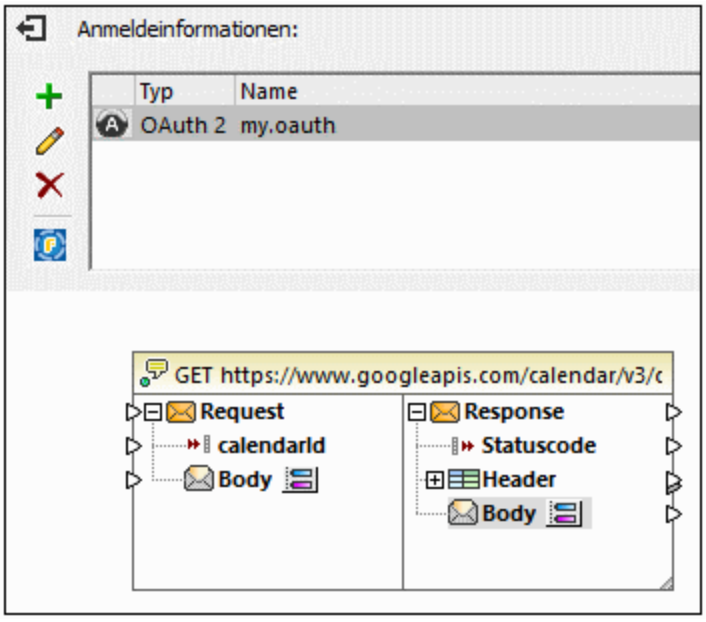


Aufgrund des Stils "Vorlage" in der Konfiguration oben kann der in geschweifte Klammern gesetzte Teil der URL zur Laufzeit durch den Parameterwert ersetzt werden. "Mapbar" bedeutet, dass der Wert über das Mapping bereitgestellt werden kann (z.B. über eine Konstante oder eventuell einen Input-Parameter). Außerdem wurde der Parameter als "Obligatorisch" gekennzeichnet, da der API-Aufruf nicht ohne diesen Parameter durchgeführt werden kann. Nähere Informationen dazu finden Sie unter Definieren von Webservice-Parametern 617.

7. Klicken Sie neben **HTTP-Sicherheitseinstellungen** auf **Bearbeiten**.
8. Aktivieren Sie im Dialogfeld "HTTP-Sicherheitseinstellungen" die Option **Anmeldeinformationen verwenden** und wählen Sie den zuvor konfigurierten Datensatz "my.oauth" aus.



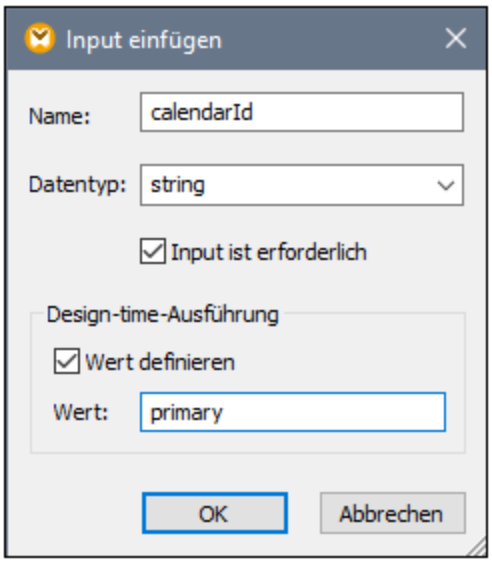
Der bisher konfigurierte Webservice wird im Mapping folgendermaßen angezeigt:



Sie können das Design nun folgendermaßen fertig stellen:

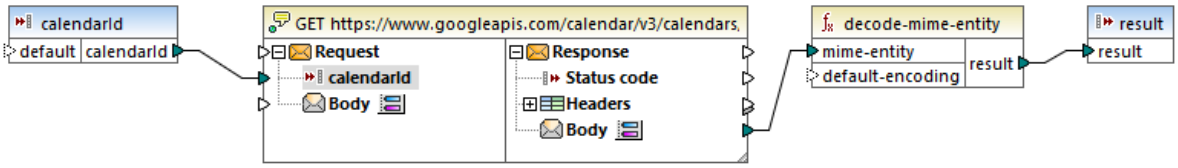
© 2016-2022 Altova GmbH

1. Klicken Sie im Menü **Einfügen** auf **Input-Komponente einfügen** und konfigurieren Sie die Komponente folgendermaßen:



   Wie oben gezeigt, hat die Input-Komponente den Design-Zeit-Wert "primary". Laut API-Dokumentation bewirkt der Wert "primary", dass der API-Server den primären Google-Kalender des aktuell angemeldeten Benutzers aufruft. Beachten Sie dass dieser Wert ein Design-Zeit-Wert ist und nur gilt, wenn Sie in MapForce eine Vorschau auf das Mapping anzeigen. Wenn das Mapping in einer Server-Umgebung ausgeführt wird, muss der gewünschte Wert zur Laufzeit bereitgestellt werden.

2. Ziehen Sie die Funktion decode-mime-entity[1182] aus dem Fenster "Bibliotheken" in den Mapping-Bereich. Diese Funktion konvertiert den vom Server erhaltenen rohen MIME-Body in einen String.
3. Klicken Sie im Menü **Einfügen** auf **Output-Komponente einfügen** und fügen Sie eine einfache Output-Komponente hinzu, deren Aufgabe es ist, das Ergebnis als einfachen String auszugeben.
4. Verbinden Sie die Komponenten als nächstes wie unten gezeigt.



Damit ist der Design-Teil in MapForce abgeschlossen.

Testen der Mapping-Ausführung

Um die Ausführung des Mappings in MapForce zu testen, klicken Sie auf das Register **Ausgabe**, um das Ergebnis im Fenster "Meldungen" zu sehen.

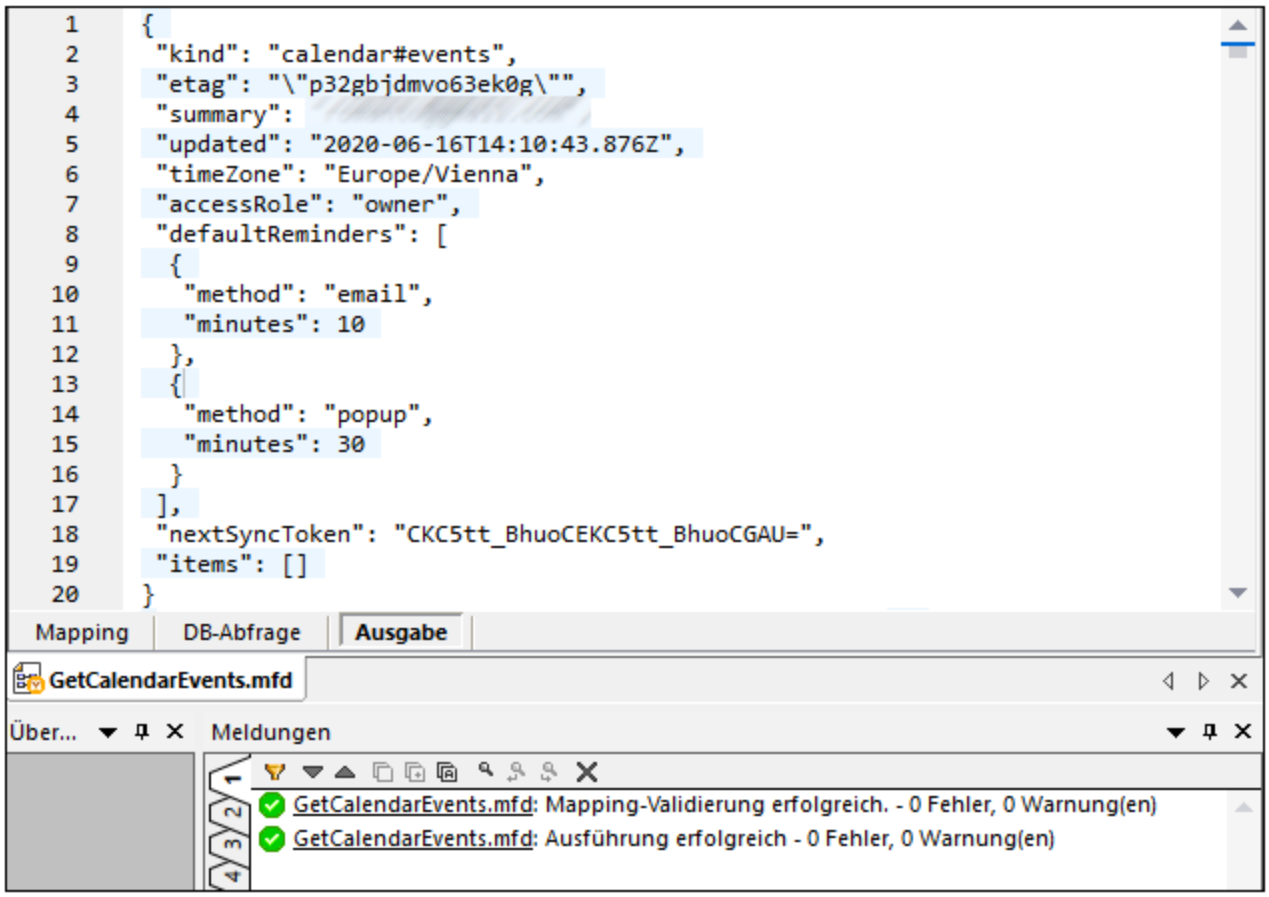
Wenn Sie einen Autorisierungsfehler wie z.B. "Unauthorized (401)" erhalten, beachten Sie die folgenden Tipps zur Fehlerbehebung:

1. Stellen Sie sicher, dass die Google Calendar API aktiviert ist, siehe Aktivieren der Google Calender API[656].
2. Falls das zuvor erhaltene Access Token bereits abgelaufen ist, fordern Sie ein neues Autorisierungs-

Token an[657].

3. Überprüfen Sie, ob alle OAuth 2.0-Informationen korrekt in MapForce eingegeben wurden.

Wenn das Mapping erfolgreich ausgeführt werden konnte und MapForce die OAuth 2.0-Autorisierung abrufen konnte, sollte die Mapping-Ausgabe ähnlich wie unten gezeigt aussehen:



Wenn Sie ein Google-Konto verwendet haben, das wie in diesem Beispiel keine Kalenderereignisse enthält, ist das Array "items" in der Antwort leer. Wenn Sie jedoch ein Ereignis zu Ihrem Google-Kalender hinzufügen und das Mapping erneut ausführen, wird dies in der Ausgabe entsprechend angezeigt. Sie können übrigens auch Ereignisse aus einem anderen Kalender, als dem Standardkalender abrufen. So könnten Sie z.B. Daten aus einem öffentlichen Kalender wie z.B. "Holidays in United States" abrufen. Setzen Sie zu diesem Zweck den Wert des Parameters **calendarId** auf `en.usa#holiday@group.v.calendar.google.com` anstelle von `primary`.

Informationen zu anderen Parametern, die Sie zum API-Aufruf hinzufügen können, finden Sie in der Dokumentation zur API-Methode unter https://developers.google.com/calendar/v3/reference/events/list.

4.5.10 Verwaltung digitaler Zertifikate

Die Verwaltung digitaler Zertifikate spielt eine wichtige Rolle beim sicheren Austausch von Daten zwischen einem Client-Rechner und einem Webserver. Da Mappings nicht nur durch MapForce unter Windows ausgeführt werden können, sondern auch durch MapForce Server (entweder als Standalone-Version oder in

© 2016-2022 Altova GmbH

einer FlowForce Server-Ausführung) auf einem Windows-, Linux- oder macOS-Server, wird in diesem Abschnitt die Verwaltung von HTTPS-Zertifikaten auf verschiedenen Plattformen behandelt.

Im Zusammenhang mit dem sicheren HyperText Transport Protocol (HTTPS) muss zwischen Server- und Client-Zertifikaten unterschieden werden.

Server-Zertifikate

Ein Server-Zertifikat dient dazu, einen Server für eine Client-Applikation wie z.B. MapForce als vertrauenswürdig zu erkennen. Das Server-Zertifikat kann von einer kommerziellen Zertifizierungsstelle oder von Ihrem Unternehmen selbst digital signiert sein. In beiden Fällen können Sie beim Erstellen eines Mappings in Mapforce die folgenden Einstellungen definieren:

- ob das Server-Zertifikat überprüft werden muss,
- ob mit dem Request fortgefahren werden soll, wenn das Namenszertifikat nicht mit dem Namen des Host übereinstimmt.

Diese Einstellungen werden über das MapForce-Dialogfeld "HTTP-Sicherheitseinstellungen" vorgenommen (siehe Einstellen der HTTP-Sicherheit 635). Wenn Sie die Überprüfung der Server-Zertifikate aktivieren, beachten Sie die folgenden Punkte:

- Wenn Sie einen Webserver aufrufen, dessen Zertifikat von einer vertrauenswürdigen Zertifizierungsstelle signiert ist, erkennt Ihr Betriebssystem das Server-Zertifikat von seiner Konfiguration her meist bereits als vertrauenswürdig, sodass keine zusätzliche Konfiguration erforderlich ist.
- Wenn Sie einen Webserver aufrufen, der ein selbst signiertes Zertifikat bereitstellt (z.B. einen lokalen Netzwerkserver in Ihrem Unternehmen), so müssen Sie auch Ihr Betriebssystem so konfigurieren, dass es dieses Zertifikat als vertrauenswürdig erkennt.

In den meisten Fällen können Sie die Konfiguration der Vertrauenswürdigkeit überprüfen, indem Sie die URL des Webservice in die Adressleiste Ihres Browsers eingeben. Wenn dem Server nicht vertraut wird oder wenn Ihr Betriebssystem den Server laut Konfiguration nicht als vertrauenswürdig einstuft, so wird in Ihrem Browser eine Meldung ungefähr folgenden Inhalts angezeigt "Diese Verbindung ist nicht vertrauenswürdig" oder "Es gibt ein Problem mit dem Zertifikat der Website". Beachten Sie, dass Sie die Einstellung für die Vertrauenswürdigkeit eines Webservers nicht überprüfen können, wenn der Browser eine andere Zertifikat-Datenbank als die des Betriebssystems verwendet (z.B. Firefox auf Ubuntu).

Unter Windows können Sie die Vertrauenswürdigkeit einer Verbindung durch Befolgen der Anweisungen im Browser und Importieren oder Installieren des benötigten Zertifikats im Speicher der vertrauenswürdigen Stammzertifikate bestätigen (siehe Vertrauenswürdige Server-Zertifikate unter Windows 670). Auf macOS-Systemen können Sie die entsprechende Operation in Keychain Access durchführen (siehe Vertrauenswürdige Server-Zertifikate auf macOS 669). Eine Anleitung zu Linux-Systemen finden Sie unter Vertrauenswürdige Server-Zertifikate unter Linux 667.

Client-Zertifikate

Während Server-Zertifikate dazu dienen, einen Server als vertrauenswürdig zu erkennen, werden Client-Zertifikate in erste Linie dazu verwendet, den aufrufenden Client auf dem Webserver zu identifizieren. Wenn Sie vorhaben, auf einen Webserver zuzugreifen, für den Client-Zertifikate erforderlich sind, müssen Sie möglicherweise den Administrator des Webservers bezüglich einer Client-Konfigurationsanweisung kontaktieren. So kann ein Webserver z.B. bei IIS (Internet Information Services) so konfiguriert werden, dass er HTTPS- und Client-Zertifikate auf eine der folgenden Arten behandelt:

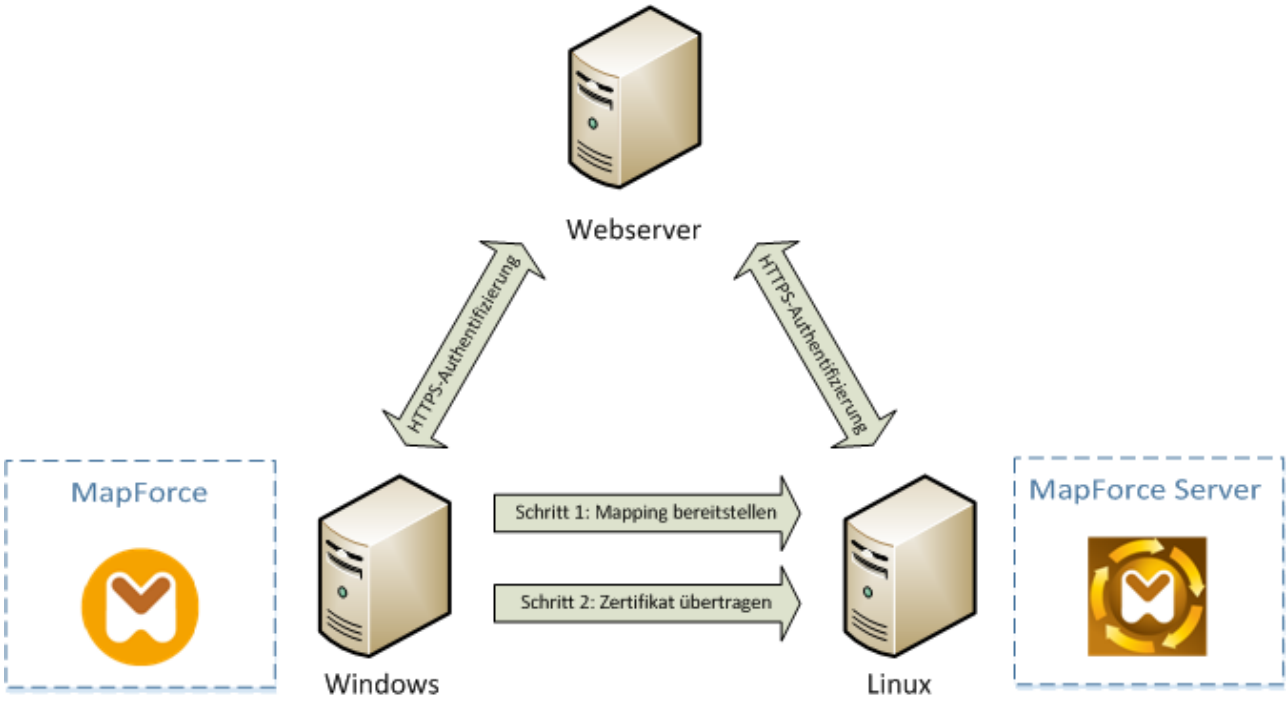
- HTTPS ist erforderlich und Client-Zertifikat wird ignoriert

- HTTPS ist erforderlich und Client-Zertifikat wird akzeptiert
- HTTPS und Client-Zertifikat sind erforderlich

Der Erfolg eines Webservice-Request hängt von der Konfiguration des Webservers und der Client-Applikation ab. Wenn der Webserver z.B. so konfiguriert ist, dass er ein Client-Zertifikat verlangt, so muss die aufrufende Applikation ein gültiges Client-Zertifikat zur Verfügung stellen, damit der Aufruf erfolgreich ist.

Aus der MapForce-Perspektive gesehen, gilt dasselbe für Mappings, die Webservice-Aufrufe über HTTPS enthalten. Damit solche Mappings erfolgreich ausgeführt werden können, geht MapForce davon aus, dass der Webserver so konfiguriert ist, dass er das Client-Zertifikat akzeptiert oder zwingend verlangt und dass das Betriebssystem, auf dem das Mapping ausgeführt wird, das richtige Client-Zertifikat für den Webserver zur Verfügung stellt.

Im unten gezeigten Diagramm sehen Sie ein Szenario, in dem ein in MapForce verwendetes Client-Zertifikat auf einen Linux-Server transferiert wird, auf dem MapForce Server ausgeführt wird. Sobald das Zertifikat auf das Zielbetriebssystem transferiert wurde, kann MapForce Server sich mit Hilfe dieses Zertifikats beim Webserver authentifizieren und das Mapping erfolgreich ausführen.



Bereitstellen von Mappings mit Client-Zertifikaten auf einem anderen Rechner

Bei einer HTTPS-Authentifizierung in Webservice-Aufrufen kann MapForce zusätzlich zu HTTP die Transport Layer Security (TLS) verwendenon. TLS ist der Nachfolger des SSL (Secure Sockets Layer)-Protokolls. Beachten Sie dass der Client bzw. Server auf SSL zurückgreift, falls die Client- bzw. Server-Implementierung TLS nicht unterstützt.

Die jeweilige Implementierung der Zertifikatverwaltung auf den einzelnen Plattformen ist dafür verantwortlich, dass Webservice-Aufrufe mit Client-Authentifizierung durch MapForce unterstützt werden. Daher muss die Zertifikatverwaltung immer im Rahmen des zugrunde liegenden Betriebssystems erfolgen. Jedes Betriebssystem bietet unterschiedliche Unterstützung für Zertifikatverwaltung - siehe Tabelle unten.

© 2016-2022 Altova GmbH

Plattform	Zertifikatverwaltung und -implementierung
Windows	Unter Windows können Sie Zertifikate mit Hilfe des Zertifikat-Snap-in verwalten (siehe Aufrufen der Zertifikatspeicher unter Windows 671). TLS-Unterstützung steht über den *Sicheren Kanal* (auch als SKanal bezeichnet) zur Verfügung.
Linux	Unter Linux können Sie Zertifikate mit Hilfe des OpenSSL (`openssl`) Befehlszeilentools und der dazugehörigen Bibliothek verwalten. Falls auf dem Linux-Recher, auf dem MapForce Server installiert ist, noch keine OpenSSL-Unterstützung zur Verfügung steht, müssen Sie diese Funktion herunterladen und sie installieren, bevor Sie Zertifikate verwalten können. TLS-Unterstützung steht über die OpenSSL-Bibliothek zur Verfügung (https://www.openssl.org/).
macOS	Auf macOS-Rechnern können Sie Zertifikate mit dem *Keychain Access Manager* unter **Finder > Applications > Utilities** verwalten. TLS-Unterstützung steht über die native *Secure Transport*-Bibliothek zur Verfügung

Wenn Sie das Mapping auf einem Windows-Betriebssystem ausführen, auf dem Sie den Webservice, den Sie aufrufen möchten, bereits erfolgreich verwendet haben, ist normalerweise keine zusätzliche Zertifikatkonfiguration erforderlich (Welche Bedingungen erfüllt werden müssen, damit das Mapping erfolgreich unter Windows ausgeführt werden kann, finden Sie unter Client-Zertifikate unter Windows 682). Wenn Sie allerdings auf einem Windows-Rechner Mappings mit MapForce erstellen und diese dann auf einem anderen Computer bereitstellen (der möglicherweise unter einem anderen Betriebssystem läuft), so wird das Client-Zertifikat nicht zusammen mit dem bereitgestellten Paket gespeichert bzw. kopiert. Damit der Webservice-Aufrufe (und das Mapping) erfolgreich ausgeführt werden können, muss das Client-Zertifikat auch auf dem Zielbetriebssystem vorhanden sein.

Um ein Zertifikat von einem Windows-System auf einen anderen Windows-basierten Rechner zu transferieren, exportieren Sie das erforderliche Zertifikat (mit dem privaten Schlüssel) aus dem Ausgangssystem (siehe Exportieren von Zertifikaten aus Windows 674). Importieren Sie anschließend dasselbe Zertifikat in den Speicher **Aktueller Benutzer\Persönlich** auf dem Zielbetriebssystem (siehe Client-Zertifikate unter Windows 682).

Eine Anleitung zum Transferieren von Client-Zertifikaten auf die Linux- und macOS-Plattform finden Sie unter Client-Zertifikate unter Linux 680 bzw. Client-Zertifikate unter macOS 681.

4.5.10.1 Definieren vertrauenswürdiger Zertifikate unter Linux

Auf Linux-Systemen gehen Sie vor, wie unten gezeigt, um ein vertrauenswürdiges Zertifikat in den Zertifikatspeicher des Systems zu importieren:

> Führen Sie die folgenden Schritte nur durch, wenn Sie sicher sind, dass das betreffende Zertifikat authentisch ist.

Befolgen Sie auf Debian und Ubuntu die folgende Anleitung:

1. Kopieren Sie die Zertifikatdatei des Webservers in das folgende Verzeichnis.

```
sudo cp /home/downloads/server_cert.crt /usr/local/share/ca-certificates/
```

2. Aktualisieren Sie den Zertifikatspeicher folgendermaßen:

```
sudo update-ca-certificates
```

Befolgen Sie auf CentOS die folgende Anleitung:

1. Installieren Sie das Paket `ca-certificates`:

```
yum install ca-certificates
```

2. Aktivieren Sie die Funktion zur dynamischen Konfiguration der Zertifizierungsstelle:

```
update-ca-trust enable
```

3. Kopieren Sie das Server-Zertifikat in das folgenden Verzeichnis:

```
cp server_cert.crt /etc/pki/ca-trust/source/anchors/
```

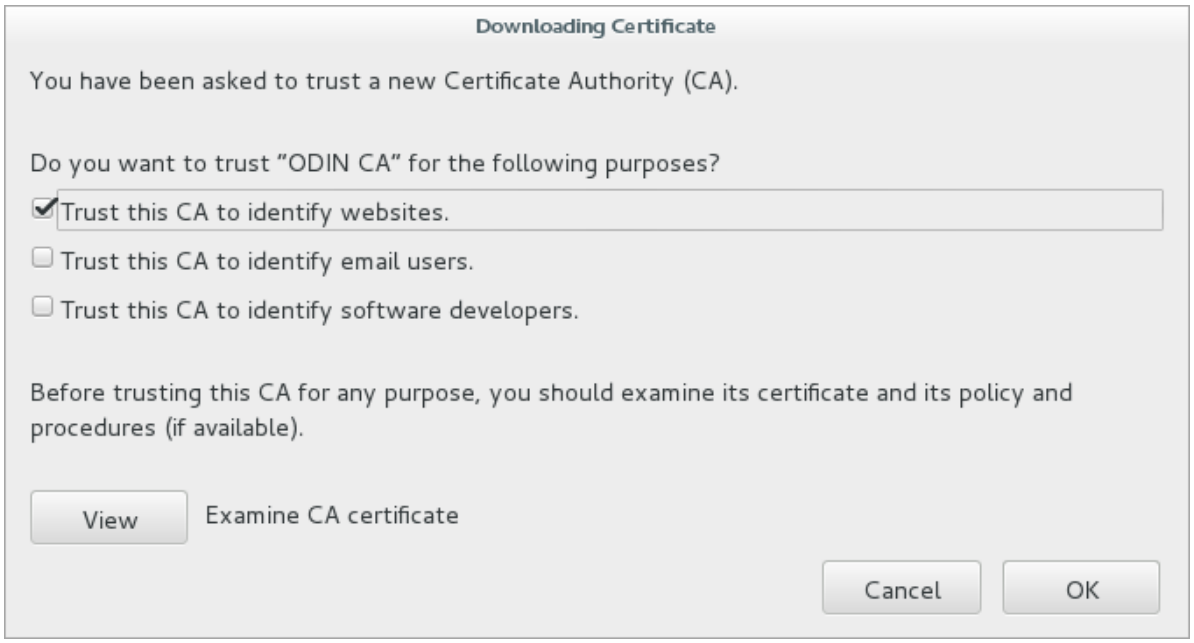
4. Verwenden Sie den folgenden Befehl:

```
update-ca-trust extract
```

In Fällen, in denen Sie nur über den Browser auf den Server zugreifen, genügt es, das Zertifikat in den Zertifikatspeicher des Browsers zu importieren. Die genaue Vorgehensweise variiert je nach Browser. In Firefox 59.0.2 können Sie z.B. folgendermaßen vorgehen:

1. Klicken Sie unter **Options | Privacy & Security** auf **View Certificates**.
2. Klicken Sie auf dem Register **Authorities** auf **Import** und navigieren Sie zur zuvor erstellen Root-Zertifikatdatei.
3. Aktivieren Sie die Option **Trust this CA to identify websites**, wenn Sie gefragt werden**.**

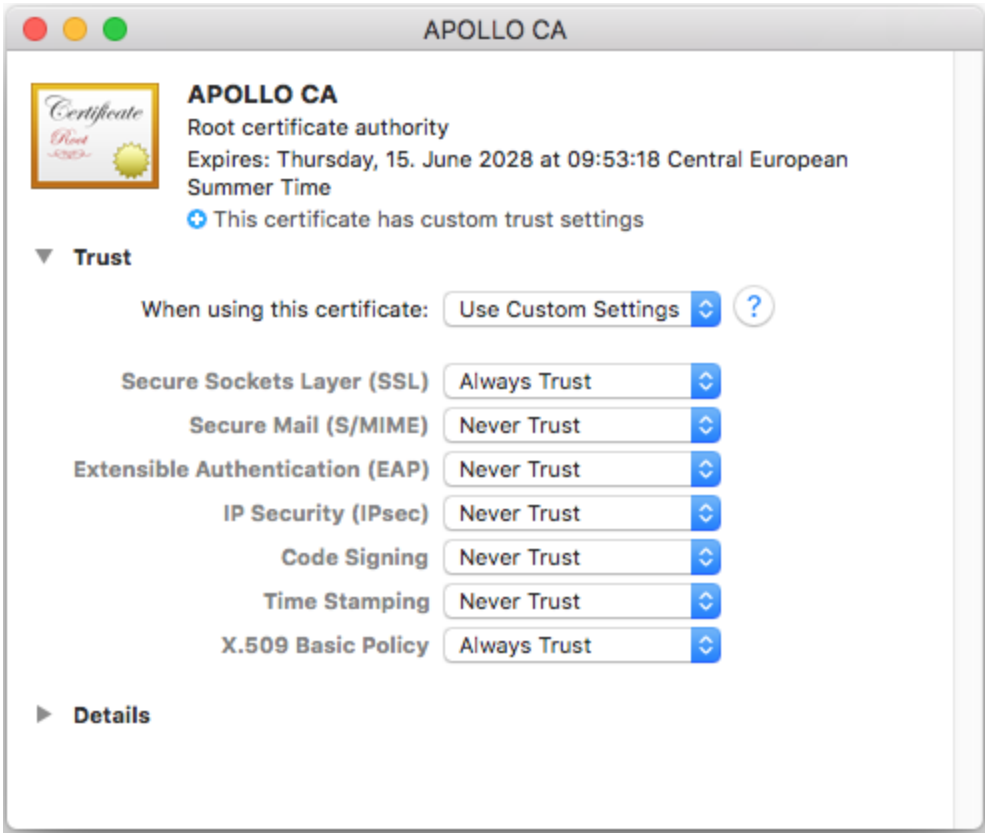
© 2016-2022 Altova GmbH



4.5.10.2 Definieren vertrauenswürdiger Zertifkate auf macOS-Systemen

Auf macOS-Systemen importieren Sie ein vertrauenswürdiges Zertifikat auf folgende Weise in den Keychain Access.

1. Starten Sie Keychain Access.
2. Klicken Sie auf **System** und anschließend auf **Certificates**.
3. Klicken Sie im Menü **File** auf **Import Items**.
4. Navigieren Sie zum gewünschten vertrauenswürdigen Zertifikat und klicken Sie auf **Open**.
5. Geben Sie das Keychain Access-Passwort ein, wenn Sie dazu aufgefordert werden und klicken Sie anschließend auf **Modify Keychain**.
6. Doppelklicken Sie auf das Zertifikat, erweitern Sie den Abschnitt **Trust** und aktivieren Sie die Option **Always Trust**.



4.5.10.3 Definieren vertrauenswürdiger Zertifikate unter Windows

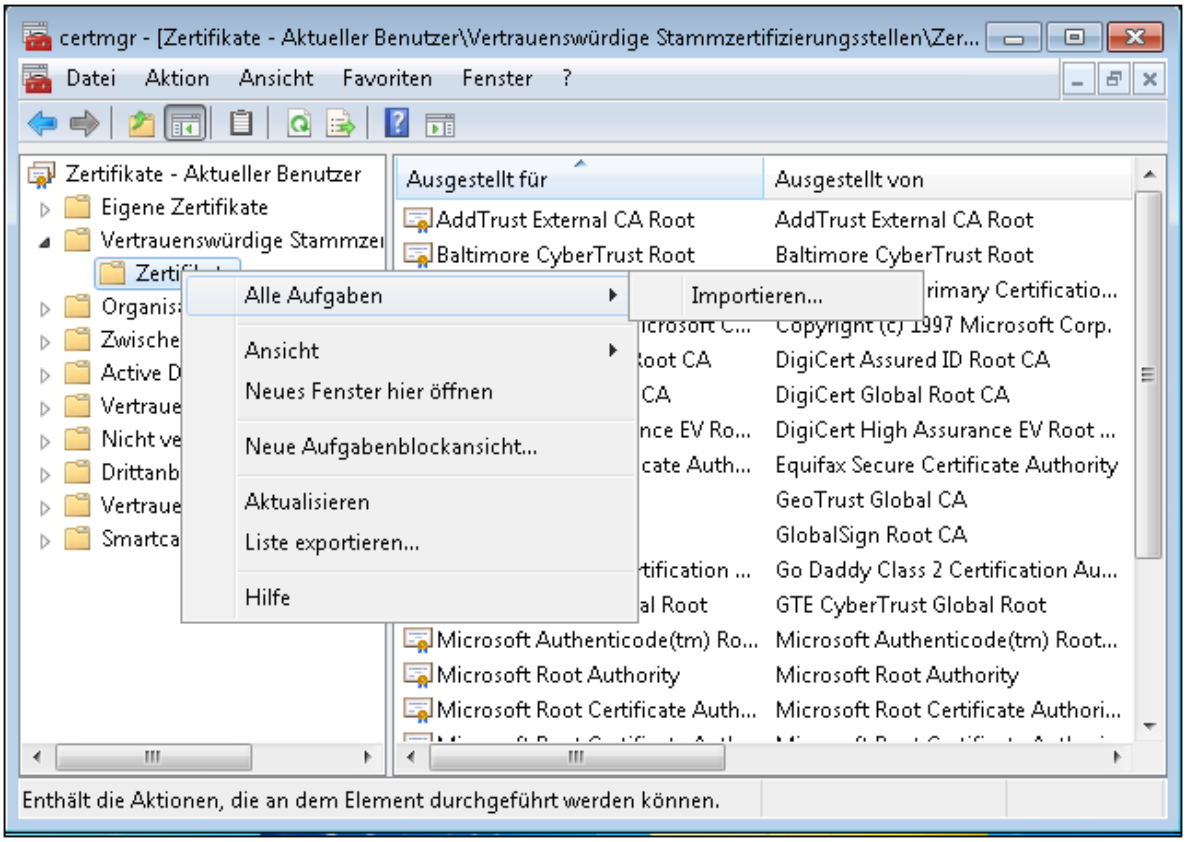
Auf Windows-Systemen importieren Sie ein vertrauenswürdiges Zertifikat auf folgende Weise in den Zertifikatspeicher des Systems:

1. Öffnen Sie den Windows-Zertifikatspeicher für *das Computer-Konto*, siehe Aufrufen des Windows-Zertifikatspeichers 671 .

   Führen Sie die folgenden Schritte nur durch, wenn Sie sicher sind, dass das Webserver-Zertifikat authentisch ist.

1. Klicken Sie unter "Vertrauenswürdige Stammzertifizierungsstellen" mit der rechten Maustaste auf **Zertifikate** wählen Sie **Alle Aufgaben | Importieren** und befolgen Sie die Anweisungen des Zertifikatimportassistenten.

© 2016-2022 Altova GmbH



Nähere Informationen dazu finden Sie unter https://technet.microsoft.com/en-us/library/cc754489(v=ws.11).aspx.

4.5.10.4 Aufrufen der Zertifikatspeicher unter Windows

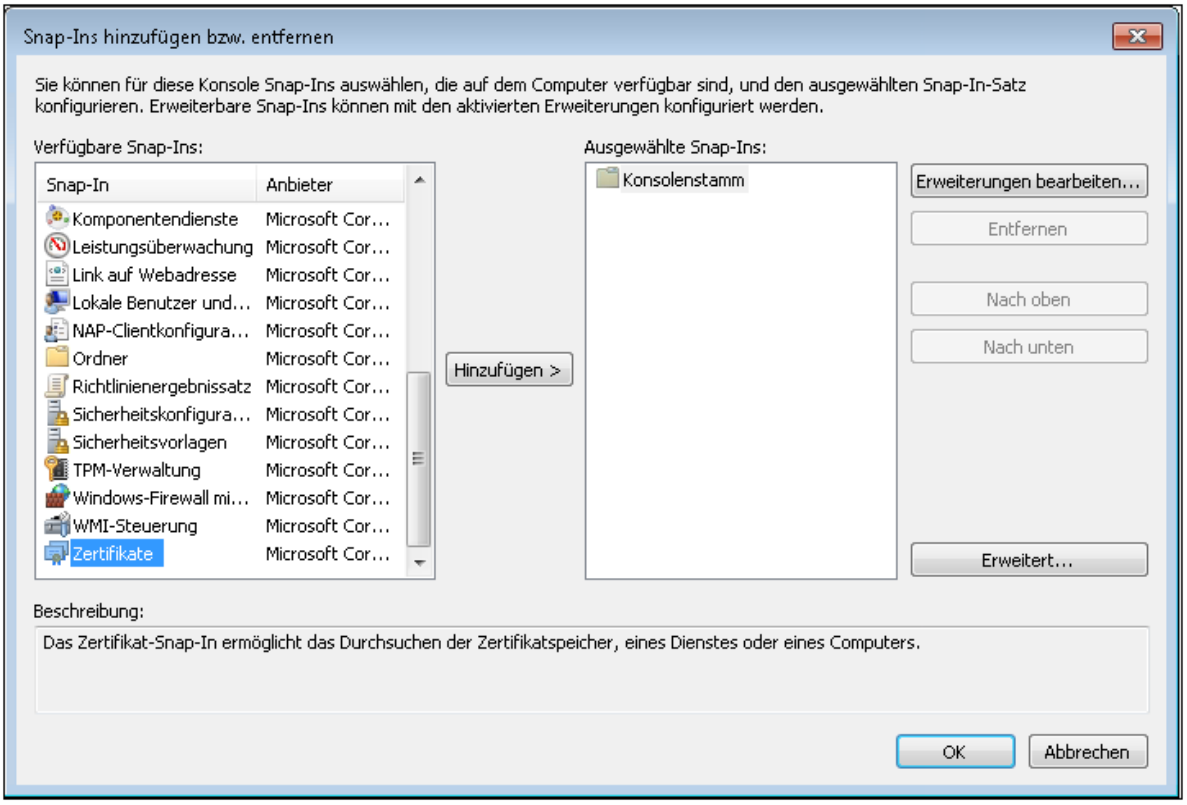
Auf Windows-Systemen können Sie Zertifikate entweder über das Microsoft Management Console (MMC) Snap-in entweder für Ihr Benutzerkonto oder für das Computerkonto verwalten.

So öffnen Sie das Snap-in "Zertifikate" (für den aktuellen Windows-Benutzer):

- Führen Sie `certmgr.msc` in der Befehlszeile aus.

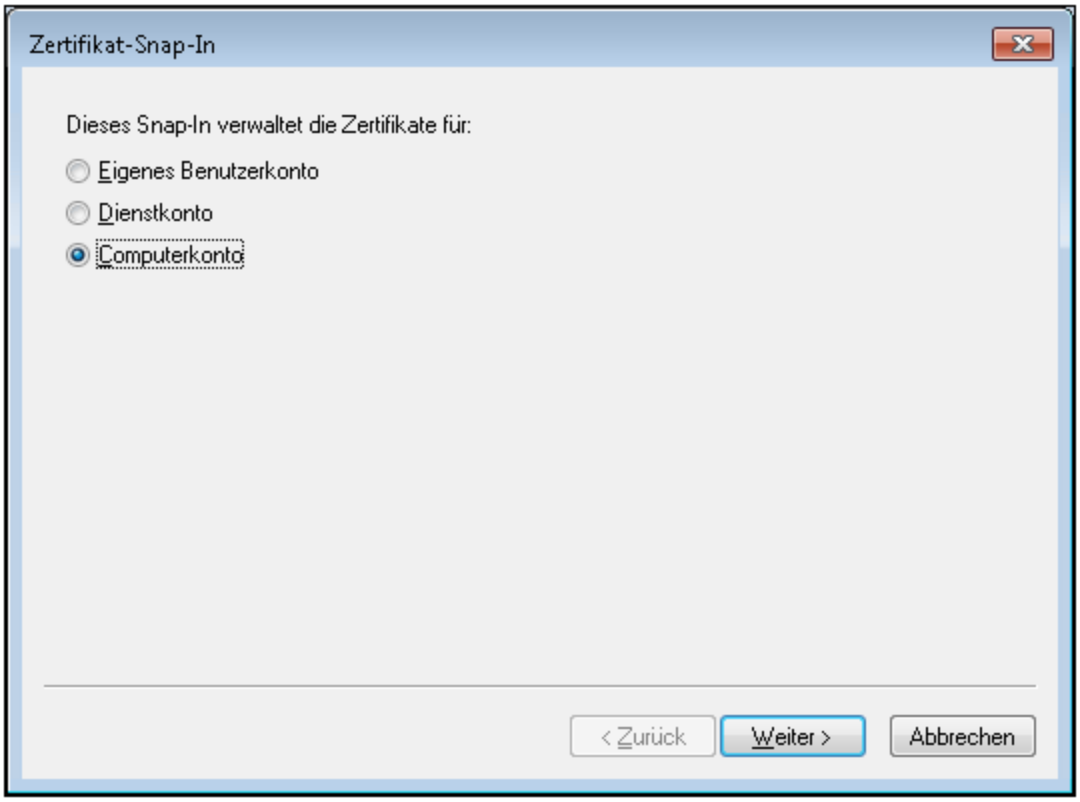
So öffnen Sie das Snap-in "Zertifikate" (für das Computerkonto):

1. Führen Sie in der Befehlszeile den Befehl `mmc` aus.
2. Klicken Sie im Menü **Datei** auf **Snap-in hinzufügen/entfernen**.
3. Klicken Sie auf **Zertifikate** und anschließend auf **Hinzufügen**.

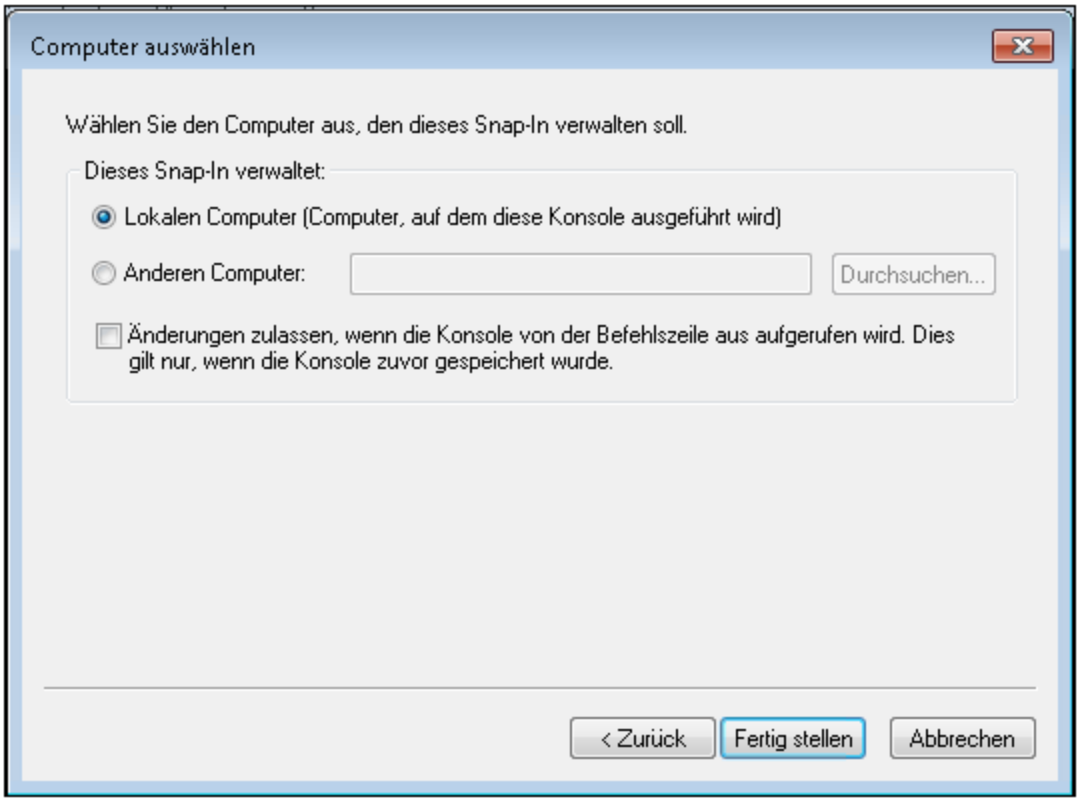


4. Klicken Sie auf **Computerkonto** und anschließend auf **Weiter**.

© 2016-2022 Altova GmbH



5. Klicken Sie auf **Lokalen Computer** und anschließend auf **Fertig stellen**.



4.5.10.5 Exportieren von Zertifikaten aus Windows

Bei Mappings, in denen Webservices über HTTPS aufgerufen werden und die auf einem macOS- oder Linux-Server, auf dem MapForce Server oder FlowForce Server ausgeführt wird, bereitgestellt werden, muss auf dem Nicht-Windows-Betriebssystem dasselbe Client-Zertifikat zur Verfügung stehen, das auch beim Design und Testen des Mappings unter Windows verwendet wurde. Um solche Mappings auf einem anderen Betriebssystem als Windows mit MapForce Server ausführen zu können, exportieren Sie das erforderliche Zertifikat mit dem privaten Schlüssel aus Windows und importieren Sie es anschließend in das Zielbetriebssystem.

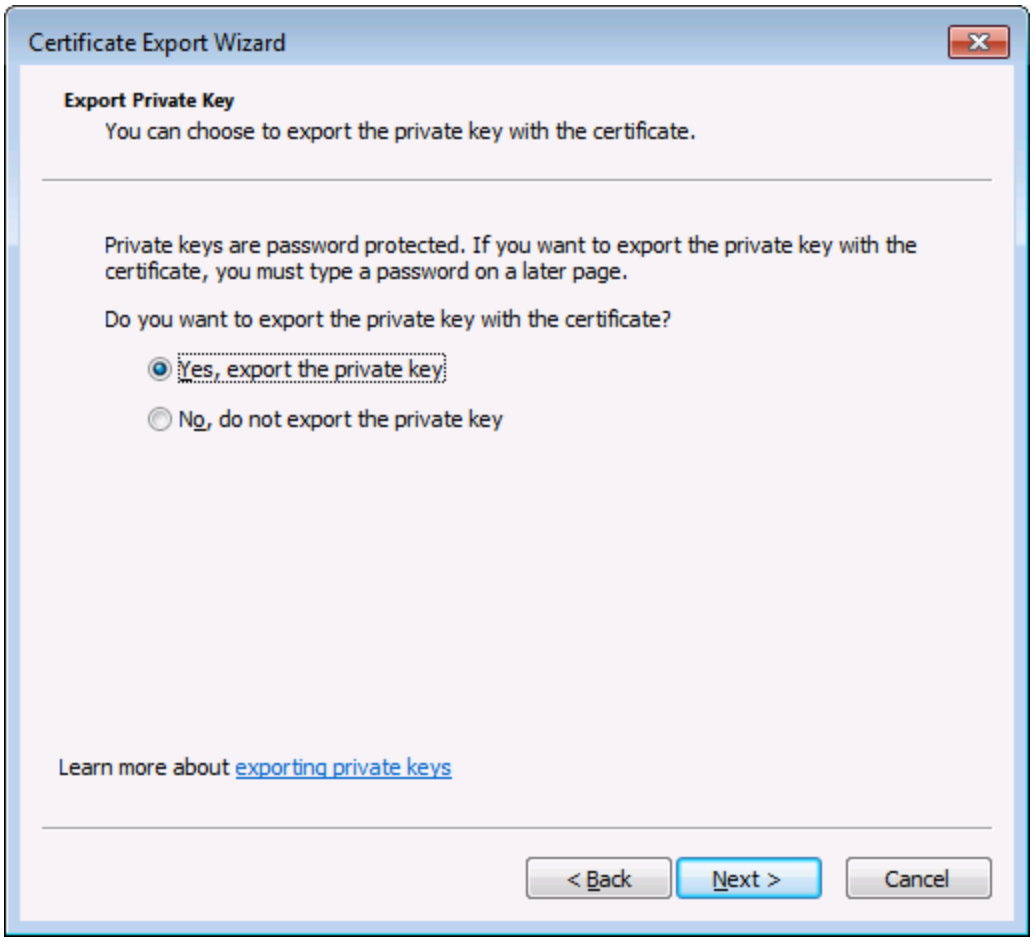
So exportieren Sie ein Zertifikat mit einem privaten Schlüssel aus Windows:

1. Öffen Sie in Windows das Zertifikate-Snap-in (siehe Aufrufen der Zertifikatspeicher unter Windows 671).
2. Klicken Sie mit der rechten Maustaste auf das gewünschte Zertifikat, platzieren Sie den Mauszeiger über **Alle Aufgaben** und klicken Sie auf **Exportieren**.
3. Klicken Sie auf **Weiter**.

© 2016-2022 Altova GmbH

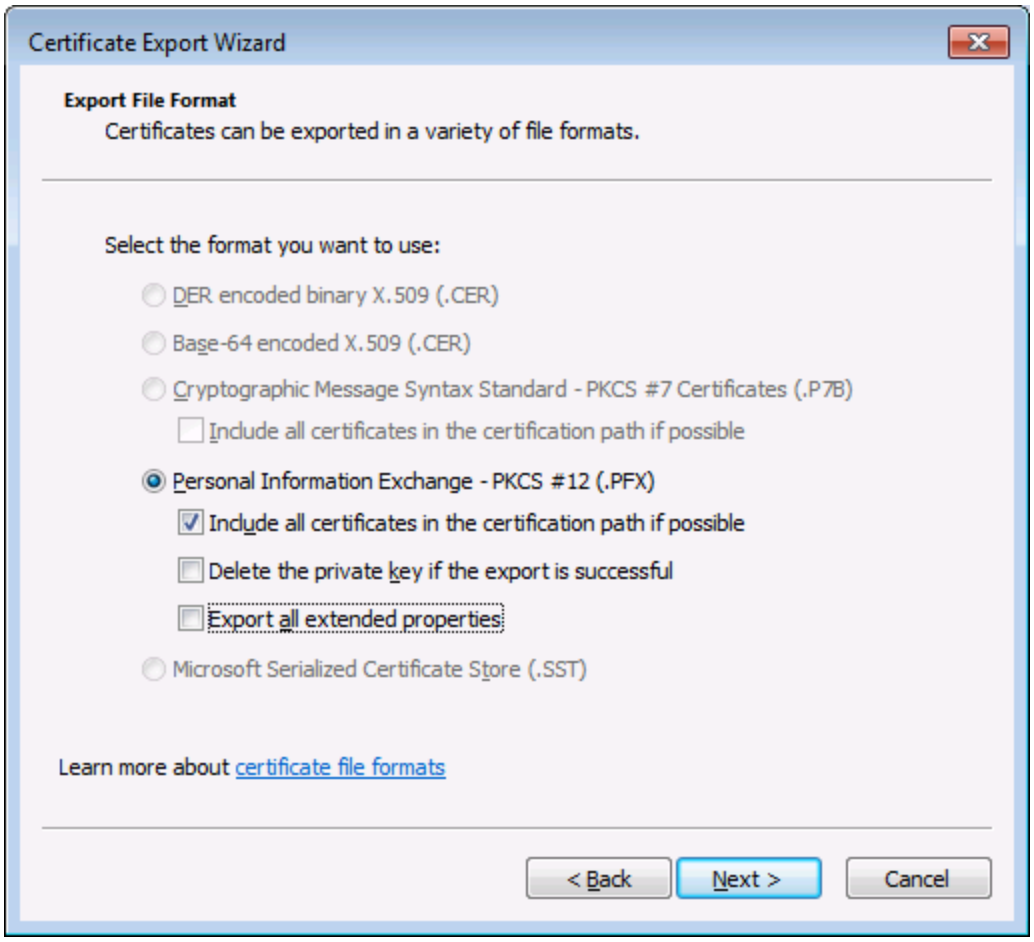


4. Wählen Sie die Option, dass das Zertifikat zusammen mit seinem privaten Schlüssel aus Windows exportiert werden soll und klicken Sie anschließend auf **Weiter**.



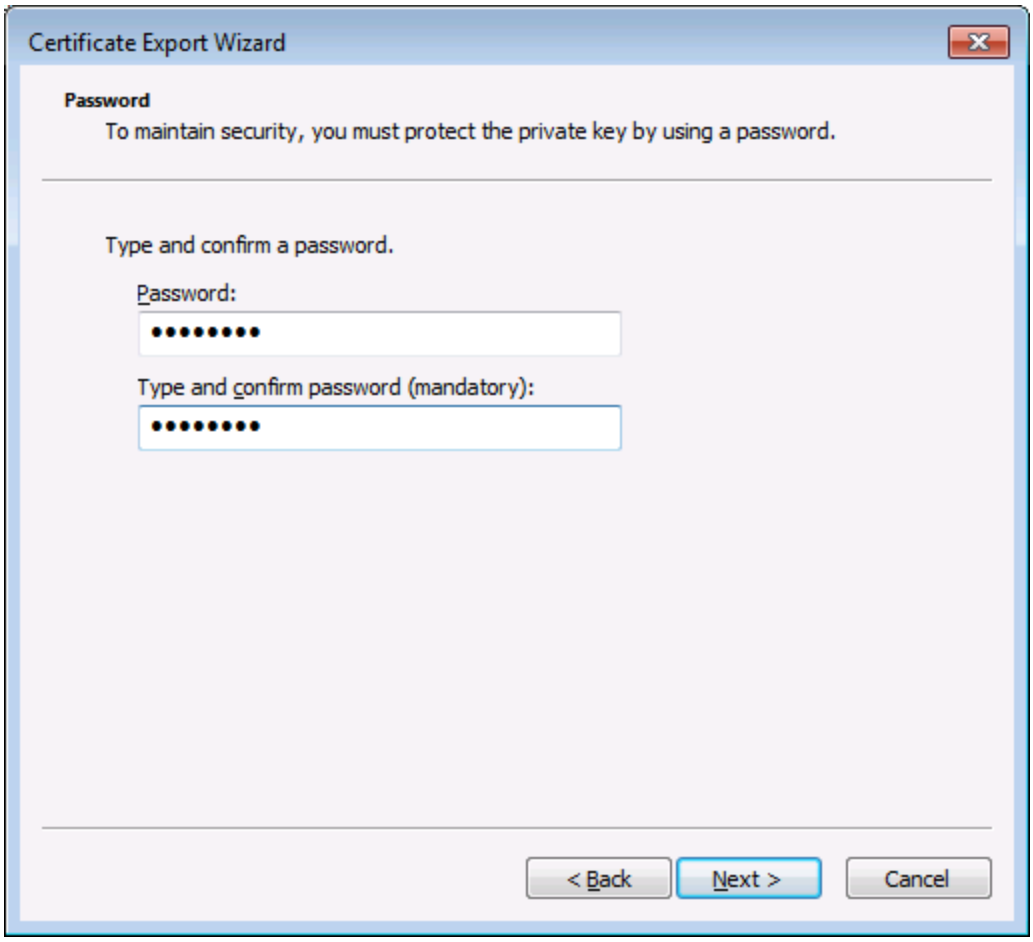
5. Wählen Sie das Dateiformat *Privater Informationsaustausch - PKCS #12 (.pfx)* aus und klicken Sie auf **Weiter**.

© 2016-2022 Altova GmbH



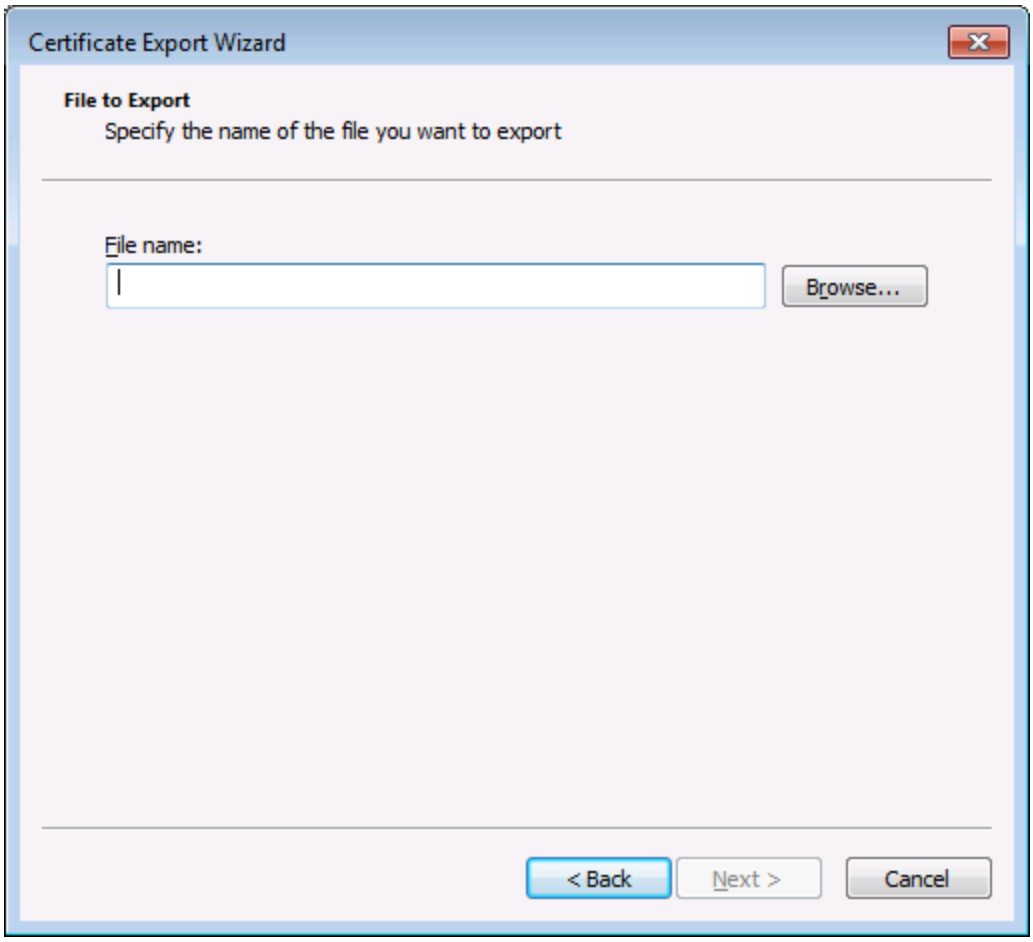
Anmerkung: Stellen Sie sicher, dass Sie nicht die Option **Privaten Schlüssel nach erfolgreichem Export löschen** aktiviert haben, da Sie das Zertifikat nach dem Export sonst nicht mehr verwenden können.

6. Geben Sie ein Passwort ein und klicken Sie auf **Weiter**. Sie benötigen dieses Passwort nachdem Sie das Zertifikat in das Zielbetriebssystem kopiert haben.

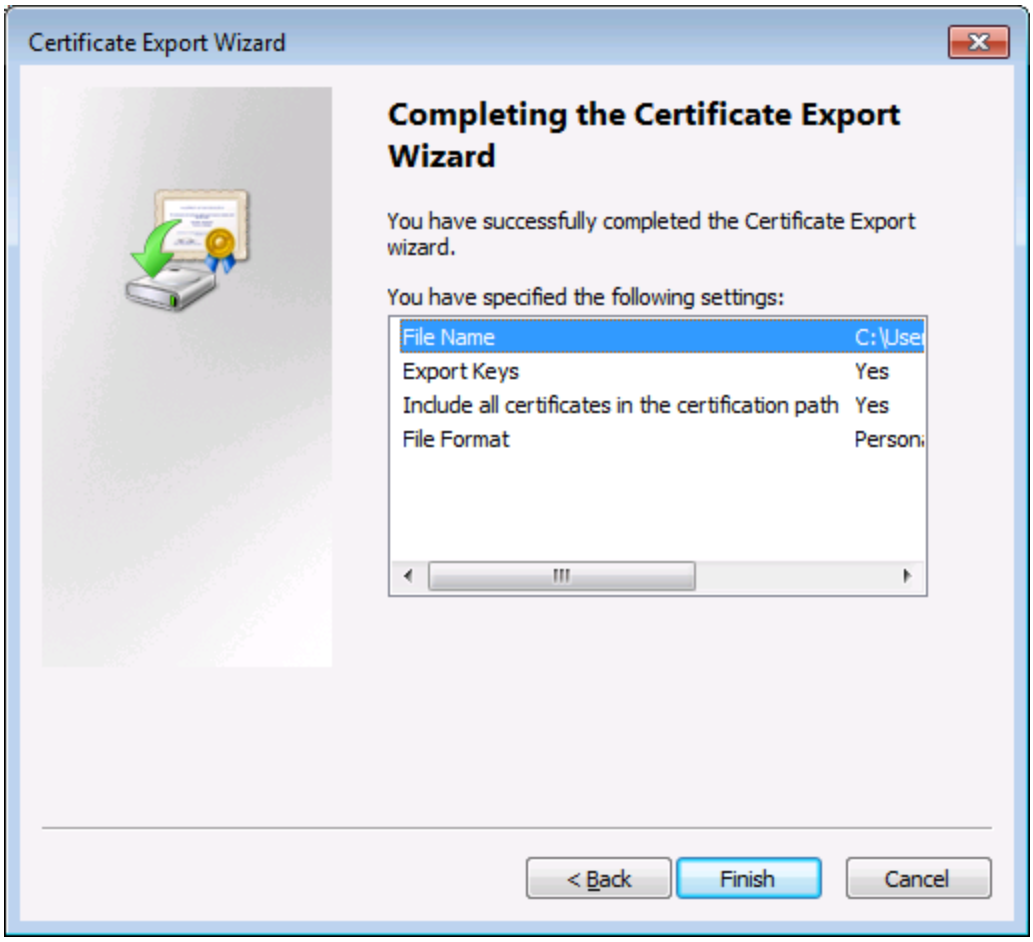


7. Navigieren Sie zur Datei, die Sie exportieren möchten und klicken Sie auf **Weiter**.

© 2016-2022 Altova GmbH



8. Klicken Sie auf **fertig stellen**.



4.5.10.6 Client-Zertifikate unter Linux

Wenn Ihre Mappings eine Webservice-Authentifizierung mittels Client-Zertifikaten über HTTPS enthalten, befolgen Sie die folgende Anleitung, um solche Mappings auf einem Linux-Rechner, auf dem MapForce Server läuft, bereitzustellen:

1. Öffnen Sie das Mapping, das den Webservice aufruft, .
2. Doppelklicken Sie auf die Überschrift der Webservice-Komponente. Daraufhin wird das Dialogfeld "Komponenteneinsetllungen" aufgerufen.
3. Klicken Sie neben "HTTP-Sicherheitseinstellungen" auf **Bearbeiten**.
4. Klicken Sie im Dialogfeld "HTTP-Sicherheitseinstellungen" auf **Client-Zertifikat** und wählen Sie anschließend in Windows das gewünschte Zertifikat aus dem Speicher **Aktueller Benutzer\Persönlich** aus (siehe Einstellen der HTTP-Sicherheit 635).
5. Speichern Sie das Mapping und kompilieren Sie es zu einerm Mapping-Ausführungsdatei oder stellen Sie es auf FlowForce bereit (siehe Kompilieren von Mappings to Server-Ausführungsdateien 1442 und Bereistellen von Mappings auf FlowForce Server 1445).
6. Übertragen Sie die für den Webservice-Aufruf erforderlichen Client-Zertifikate auf das Zielbetriebssystem. Stellen Sie sicher, dass das Zertifikat einen privaten Schlüssel hat und dass die Eigenschaft **Erweiterte Schlüsselverwendung** des Zertifikats "Clientauthentifizierung" als Zweck enthält.

© 2016-2022 Altova GmbH

So transferieren Sie das Client-Zertifikat auf Linux:

1. Exportieren Sie das Client-Zertifikat mit dem privaten Schlüssel im Dateiformat *Privater Informationsaustausch - PKCS #12 (.pfx)* (siehe Exportieren von Zertifikaten aus Windows 674).
2. Kopieren Sie die Zertifikatdatei auf den Linux-Rechner.
3. Konvertieren Sie die .pfx-Datei mit dem folgenden Befehl in das .pem-Format.

   ```
   openssl pkcs12 -in cert.pfx -out "John Doe.pem" -nodes
   ```

   Dieser Befehl parst die .pfx-Datei und erzeugt eine .pem-Datei, ohne dabei den privaten Schlüssel zu verschlüsseln. Bei Zertifikaten mit einem verschlüsselten privaten Schlüssel wird nach einem Passwort gefragt, was bei der Server-Ausführung nicht unterstützt wird.

Ausführen des Mappings

Damit MapForce Server die .pem-Datei als Client-Zertifikat verwendet, definieren Sie beim Ausführen des Mappings den Parameter `--certificatespath`. Der Parameter `--certificatespath` definiert den Pfad zu dem Verzeichnis, in dem alle vom aktuellen Mapping benötigten Zertifikate gespeichert sind. Wenn der Pfad zur Zertifikatdatei z.B. `/home/John/John Doe.pem` lautet, so muss `--certificatespath` auf `/home/John` gesetzt werden.

Wenn der Parameter `--certificatespath` nicht angegeben wird, sucht MapForce Server standardmäßig im Verzeichnis `$HOME/.config/altova/certificates` des aktuellen Benutzers nach Zertifikaten.

Damit das Mapping erfolgreich ausgeführt werden kann, muss die Zertifikatdatei die Erweiterung .pem haben und der Name der Datei muss mit dem Common Name (CN) des Zertifikats einschließlich der Leerzeichen übereinstimmen (z.B. **John Doe.pem**).

Wenn Sie vorhaben, das Mapping als FlowForce Server-Auftrag auszuführen, kopieren Sie das Zertifikat in das Verzeichnis `$HOME/.config/altova/certificates`. FlowForce Server sucht bei Ausführung des Auftrags in diesem Verzeichnis nach allen für das Mapping erforderlichen Zertifikatdateien.

Stellen Sie aus Sicherheitsgründen sicher, dass Zertifikatdateien nicht von anderen Benutzern gelesen werden können, da sie sensible Informationen enthalten.

4.5.10.7 Client-Zertifikate auf macOS

Wenn Ihre Mappings eine Webservice-Authentifizierung mittels Client-Zertifikaten über HTTPS enthalten, befolgen Sie die folgende Anleitung, um solche Mappings auf einem macOS-System, auf dem MapForce Server läuft, bereitzustellen:

1. Öffnen Sie das Mapping, das den Webservice aufruft, .
2. Doppelklicken Sie auf die Überschrift der Webservice-Komponente. Daraufhin wird das Dialogfeld "Komponenteneinsetllungen" aufgerufen.
3. Klicken Sie neben "HTTP-Sicherheitseinstellungen" auf **Bearbeiten**.
4. Klicken Sie im Dialogfeld "HTTP-Sicherheitseinstellungen" auf **Client-Zertifikat** und wählen Sie

anschließend das gewünschte Zertifikat aus (siehe Einstellen der HTTP-Sicherheit[635]).
5. Wenn der Zertifikatname nicht genau mit dem Host-Namen des Servers übereinstimmt, aktivieren Sie die Option **Diskrepanz zwischen Host-Namen im Zertifikat und Request zulassen**.
6. Speichern Sie das Mapping und stellen Sie es auf dem Zielbetriebssystem bereit (siehe Bereitstellen eines MapForce-Mappings[1445]).
7. Übertragen Sie die für den Webservice-Aufruf erforderlichen Client-Zertifikate auf das Zielbetriebssystem. Stellen Sie sicher, dass das Zertifikat einen privaten Schlüssel hat und dass die Eigenschaft **Erweiterte Schlüsselverwendung** des Zertifikats "Clientauthentifizierung" als Zweck enthält.

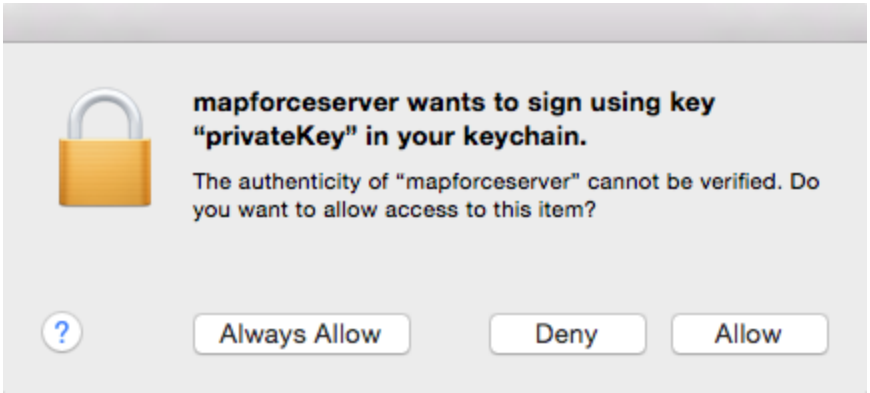
So übertragen Sie das Client-Zertifikat auf das macOS-System:

1. Exportieren Sie das Client-Zertifikat mit dem privaten Schlüssel im Dateiformat *Privater Informationsaustausch - PKCS #12 (.pfx)* (siehe Exportieren von Zertifikaten aus Windows[674]) aus Windows und kopieren Sie die .pfx-Datei auf den macOS.
2. Stellen Sie sicher, dass das Betriebssystem das Server-Zertifikat als vertrauenswürdig einstuft (siehe Definieren vertrauenswürdiger Zertifkate auf Mac-Systemen[669].
3. Führen Sie über **Finder > Applications > Utilities** Keychain Access aus.
4. Klicken Sie im Menü **File** auf **Import Items**.
5. Navigieren Sie zu dem in Schritt 1 exportierten Client-Zertifikat und wählen Sie eine Ziel-Keychain aus.
6. Klicken Sie auf **Open** und geben Sie das Passwort ein, mit dem das Zertifikat verschlüsselt wurde.

Ausführen des Mappings

Sie können jetzt das Mapping mit dem MapForce Server-Befehl `run` ausführen. Beachten Sie dazu Folgendes:

- Wenn Sie das Mapping über einen entfernten Rechner mittels SSH ausführen, entsperren Sie zuerst die Keychain mit dem Befehl `security unlock-keychain`.
- Wenn Sie das Mapping über die grafische Benutzeroberfläche von macOS ausführen, klicken Sie auf **Allow,** wenn Sie gefragt werden, ob MapForce Server Zugriff auf die Keychain erhalten soll.



4.5.10.8 Client-Zertifikate unter Windows

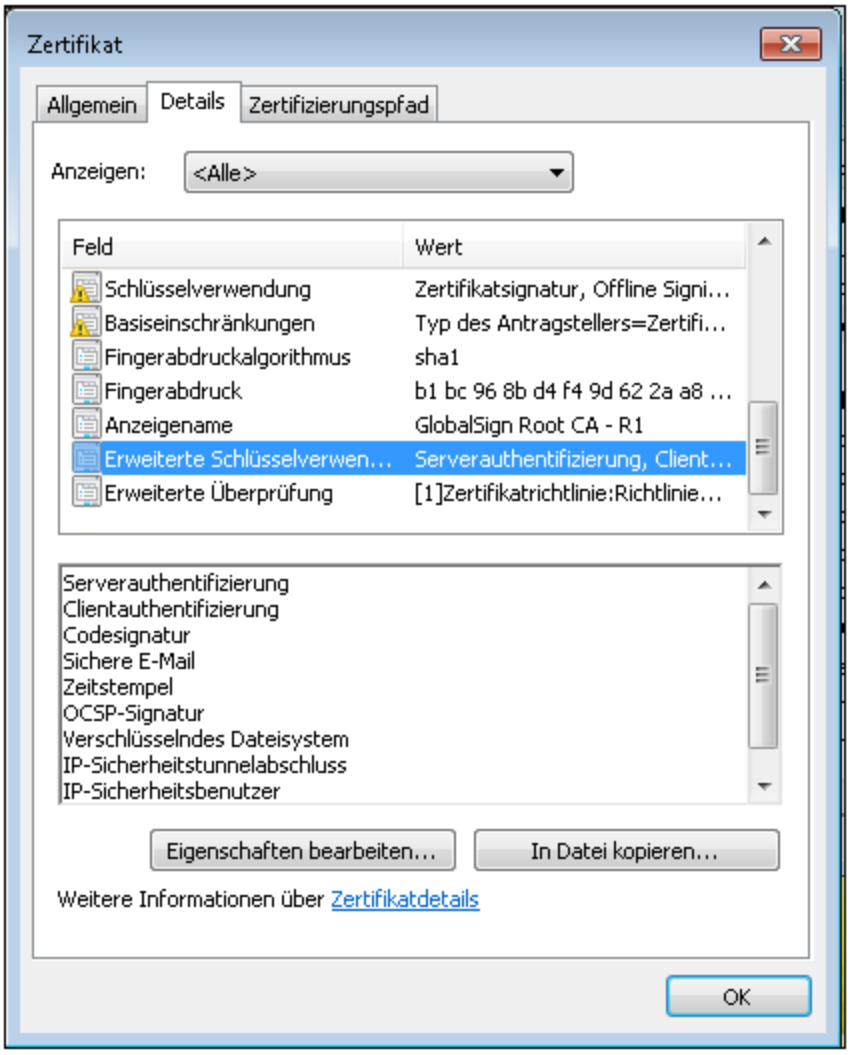
Wenn Sie unter Windows ein Mapping ausführen, für das Client-Zertifikate erforderlich sind, müssen die folgenden Voraussetzungen erfüllt werden, damit das Mapping erfolgreich ausgeführt werden kann:

- Das Client-Zertifikat muss im Zertifikatspeicher **Aktueller Benutzer\Persönlich** (auch als **Mein**

© 2016-2022 Altova GmbH

Speicher bezeichnet) vorhanden sein. Dazu muss es mit dem Zertifikatimport-Assistenten importiert werden. Eine Anleitung dazu finden Sie unter https://technet.microsoft.com/de-de/library/cc754489(v=ws.11).aspx.

- Das Zertifikat muss einen privaten Schlüssel haben.
- Die Eigenschaft **Erweiterte Schlüsselverwendung** des Zertifikats muss als Zweck "Clientauthentifizierung" enthalten.



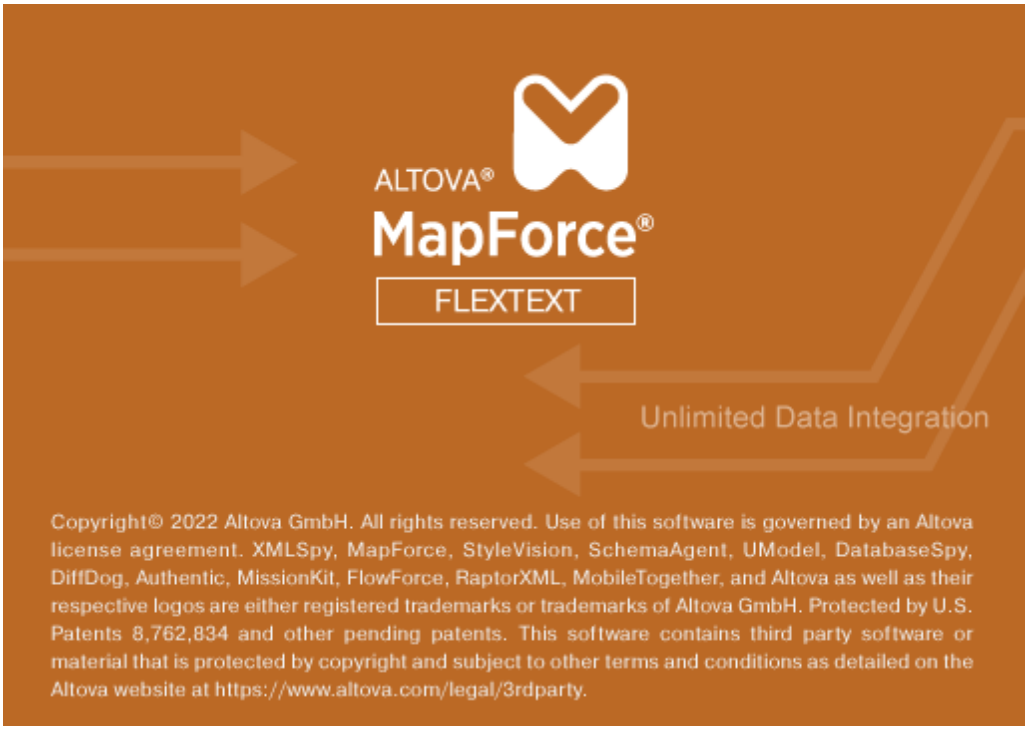
In der aktuellen Version von MapForce wählt Windows das erforderliche Zertifikat aufgrund einer Einschränkung der von MapForce verwendeten Bibliothek automatisch aus dem Zertifikatspeicher aus, wenn Sie das Mapping ausführen. Das Mapping wird erfolgreich ausgeführt, wenn der Server nach dem Filtern des Zertifikatspeichers **Aktueller Benutzer\Persönlich** ein passendes Zertifikat findet. Beachten Sie, dass die HTTPS-Authentifizierung (und die Auswahl des Zertifikats) von Windows und nicht von MapForce oder MapForce Server gesteuert wird. Es kann vorkommen, dass im Zertifikatspeicher **Aktueller Benutzer\Persönlich** mehrere Zertifikate vorhanden sind und dass das Betriebssystem automatisch ein unpassendes Zertifikat auswählt, wodurch das Mapping fehlschlägt. Diese Situation lässt sich vermeiden, indem Sie die Anzahl der im Speicher **Aktueller Benutzer\Persönlich** verfügbaren Zertifikate einschränken.

4.6 MapForce FlexText

Altova Website: Mappen komplexer Textdateien - FlexText

FlexText ist ein MapForce-Modul, mit dem Sie Daten aus nicht standardgemäßen Dateien oder alten Textdateien mit hohem Komplexitätsgrad in andere von MapForce unterstützte Formate konvertieren können. Während XML-Dateien ein Schema oder eine Struktur haben, anhand derer MapForce das Schema ableiten kann, ist dies bei Textdateien nicht der Fall, vor allem, wenn diese eine komplexe und ungewöhnliche Struktur haben, die nicht genau mit CSV-Standards oder Feldern mit fester Länge übereinstimmt. Außerdem müssen manchmal nur Teile aus einer Textdatei extrahiert werden. FlexText löst dieses Problem, indem es Ihnen erlaubt, mit visuellen Mitteln Regeln bzw. eine Vorlage für die Unterteilung und Trennung von Textdaten in mapbare Datenelemente zu definieren und diese in Echtzeit zu testen. Mit einer FlexText-Vorlage wird im Grunde anhand festgelegter Kriterien ein Strukturmodell für benutzerdefinierte Textdaten definiert, um diese auf andere Formate mappen zu können.

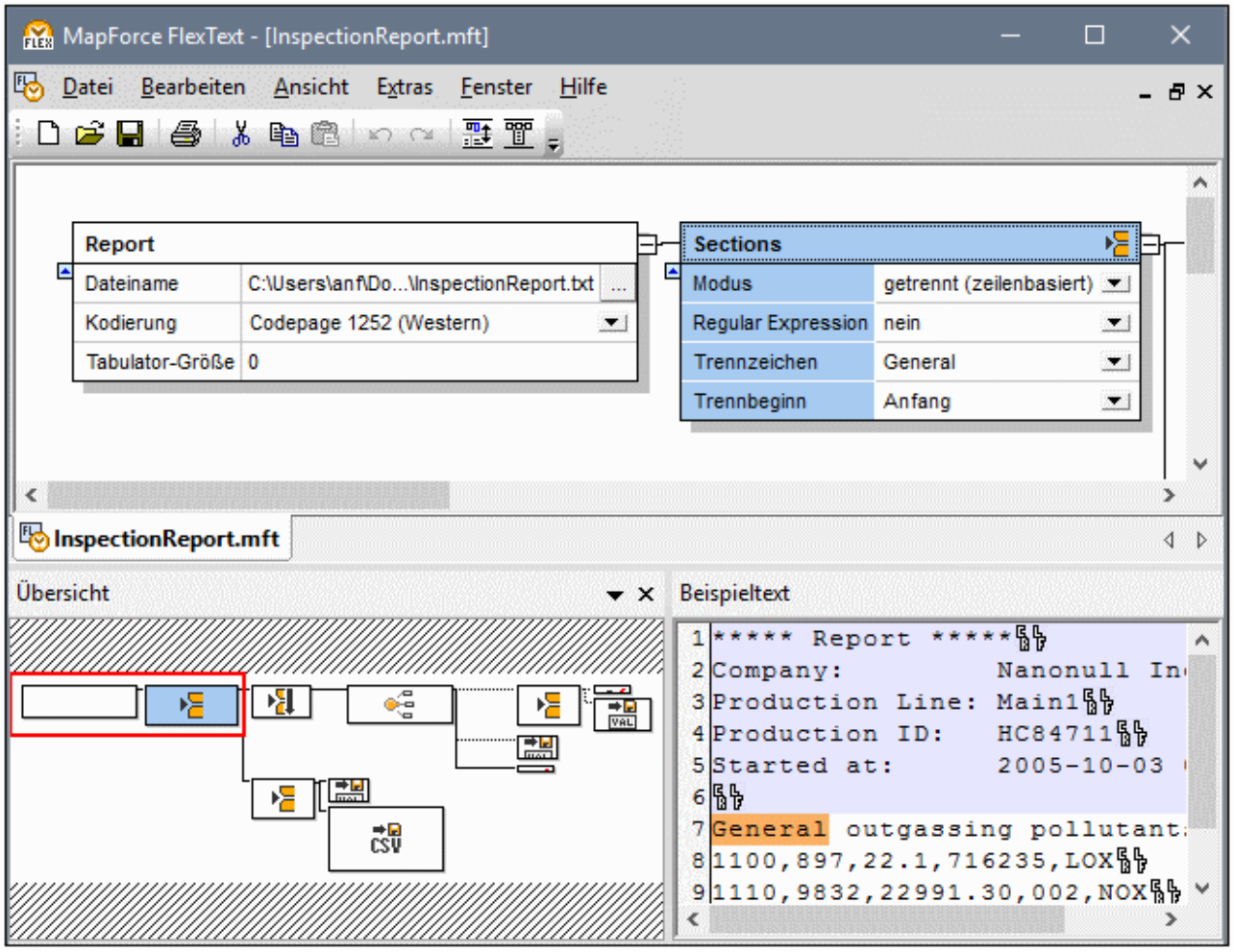
Wenn Sie eine FlexText-Vorlage definieren, können Sie diese als Quell- oder Zielkomponente zum MapForce Mapping hinzufügen und damit komplexe, nicht standardgemäße Textdaten auf andere von MapForce unterstützte Formate mappen. FlexText-Vorlagendateien haben die Erweiterung .mft (MapForce FlexText). Sie können ein und dieselben FlexText-Vorlage in mehreren Mappings verwenden, um damit beliebig viele Textdateien zu verarbeiten.



4.6.1 Übersicht

Sie können FlexText direkt von MapForce aus starten, wenn Sie Textdateien als Mapping-Komponenten hinzufügen (Befehl **Einfügen | Textdatei**). Sie können FlexText auch als eigenständiges Programm starten, indem Sie die ausführbare Datei *Altova MapForce FlexText* aus dem MapForce-Installationsverzeichnis starten.

In der Abbildung unten sehen Sie eine FlexText-Beispielvorlage (**InspectionReport.mft**), die zusammen mit weiteren FlexText-Beispieldateien im Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples** zur Verfügung steht.



Die grafische Benutzeroberfläche von FlexText

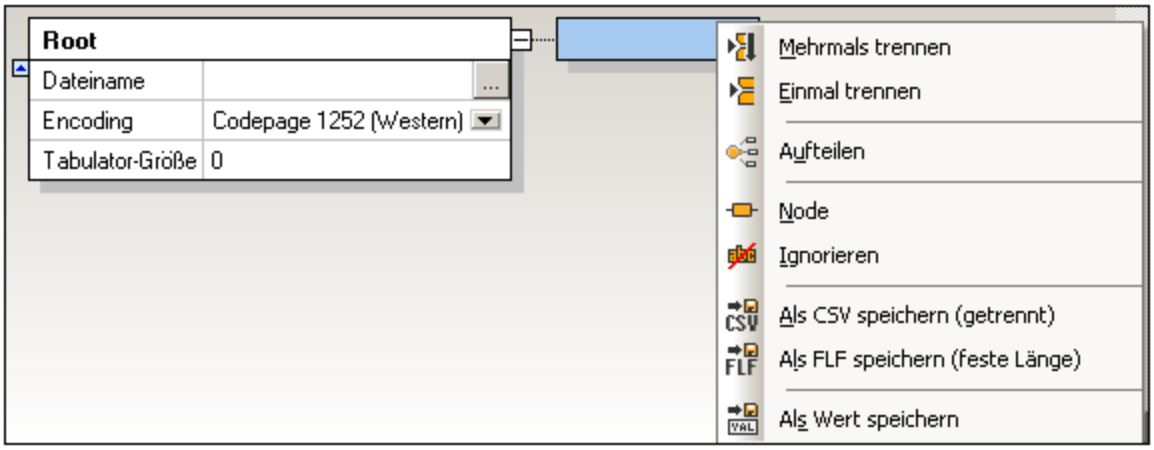
Die Benutzeroberfläche von FlexText besteht aus den folgenden Fenstern: dem Design-Fenster, dem Übersichtsfenster und dem Fenster "Beispieltext". Die Position und das Aussehen dieser Fenster können ganz nach Belieben angepasst werden (siehe unten).

Design-Fenster

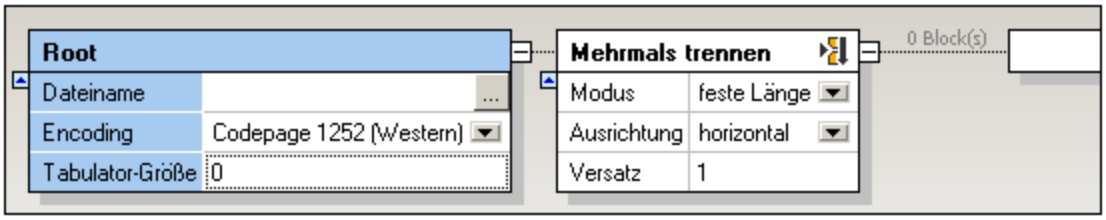
Das Design-Fenster ist der Arbeitsbereich, in dem Sie die Struktur Ihres Texts mit Hilfe von Containern definieren. Container enthalten die Regeln für die Behandlung von Text nach den von Ihnen definierten Kriterien. Der erste Container jeder FlexText-Vorlage ist der *Root*-Container. Er repräsentiert die gesamte Textdatei. Alle anderen Container enthalten aufeinander folgende Anweisungen zur Verarbeitung des Texts (normalerweise wird Text dadurch in sinnvolle Einheiten, von denen aus Daten gemappt werden können, unterteilt). So trennt etwa der Container *Einmal trennen* ein Textfragment in genau zwei Fragmente. Standardmäßig enthält der Name des Containers die Beschreibung seiner Funktion (z.B. *Mehrmals trennen*), Sie können den Namen bei Bedarf jedoch ändern. Normalerweise definieren Sie so viele Container, wie es die Struktur der jeweiligen Textdatei erfordert.

Im Abschnitt FlexText-Referenz[701] finden Sie eine ausführlichere Beschreibung zu Containern. Beachten Sie bei Containern die folgenden Punkte:

- Wenn Sie auf das Symbol in der rechten oberen Ecke eines Containers klicken, wird ein Popup-Menü geöffnet, über das Sie den Container-Typ auswählen können.



- Jeder Container enthält eine Reihe von Optionen, mit denen Sie festlegen, wie Textdaten an dieser Stelle der Struktur behandelt werden sollen. Mit diesen Optionen wird der Inhalt des Containers festgelegt und Sie können den Inhalt damit bearbeiten, bevor Sie ihn für die MapForce-Komponente zur Verfügung stellen.



- Um den Inhalt von Containern ein- bzw. auszublenden klicken Sie auf das Symbol **Erweitern** bzw. **Reduzieren**.
- Um Container als Gruppe zu reduzieren, drücken Sie die **Umschalttaste**. Daraufhin wird ein Symbol mit zwei Pfeilen angezeigt . Wenn Sie darauf klicken, wird der Abschnitt der Containerstruktur rechts von diesem Bereich reduziert.
- Um direkt im Design-Fenster eine Vorschau des durch einen bestimmten Container erzeugten Texts zu sehen, aktivieren Sie die Symbolleisten-Schaltfläche **Node-Text in Design-Ansicht** .
- Wenn die Symbolleisten-Schaltfläche **Node-Text für nicht ausgewählte automatisch reduzieren** aktiviert ist, wird der gesamte Inhalt eines Containers erweitert angezeigt, wenn Sie den Container auswählen. Alle anderen Container werden reduziert angezeigt.

Übersichtsfenster

Im Übersichtsfenster sehen Sie eine Gesamtansicht des Design-Fensters. Ziehen Sie des rote Rechteck mit der Maus, um im Fenster zu navigieren. Um das Übersichtsfenster abgedockt anzuzeigen und es an einer anderen Stelle der Benutzeroberfläche anzuzeigen, klicken Sie auf seine Titelleiste und ziehen Sie das Fenster an die gewünschte Position.

© 2016-2022 Altova GmbH

Fenster "Beispieltext"

Im Fenster "Beispieltext" wird der Inhalt des gerade ausgewählten Containers angezeigt. (Beachten Sie, dass dieses Fenster nicht zu sehen ist, wenn die Symbolleisten-Schaltfläche **Node-Text in Design-Ansicht** aktiv ist. In diesem Fall wird der Inhalt des ausgewählten Containers im Design-Fenster angezeigt.

Damit die Zeichen **Tab** und **Wagenrücklauf/Zeilenvorschub** im Vorschautext besser zu sehen sind, werden diese Zeichen in FlexText mit den Symbolen ◦ bzw. ¶ angezeigt.

4.6.2 FlexText Tutorial

Im Tutorial wird erläutert, wie Sie die wichtigsten Funktionen von FlexText verwenden, um eine Textdatei zu verarbeiten und das Resultat auf verschiedene Arten in MapForce zu mappen.

Im Beispiel wird die Datei **Flex-tutorial.txt** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** verwendet. Die .txt-Datei hat das folgende Format:

```
111,332.1,22537.7,5,Container ship,Mega,
111,A1579227,10,3,400,Microtome,
111,B152427,7,6,1200,Miscellaneous,
222,978.4,7563.1,69,Air freight,Mini,
222,ZZAW561,10,5,10000,Gas Chromatograph,,

General outgassing pollutants
1100,897,22.1,716235,LOX
1110,9832,22991.30,002,NOX
1120,1213,33.01,008,SOX
```

Ziel dieses Tutorials:

- die Datensätze mit den Schlüsseln, die die Zahlen 111 und 222 enthalten, herauszutrennen und in Elementen abzulegen, die separat gemappt werden können,
- die reinen Textstellen zu entfernen,
- eine CSV-Datei mit den verbleibenden Datensätzen zu erstellen.

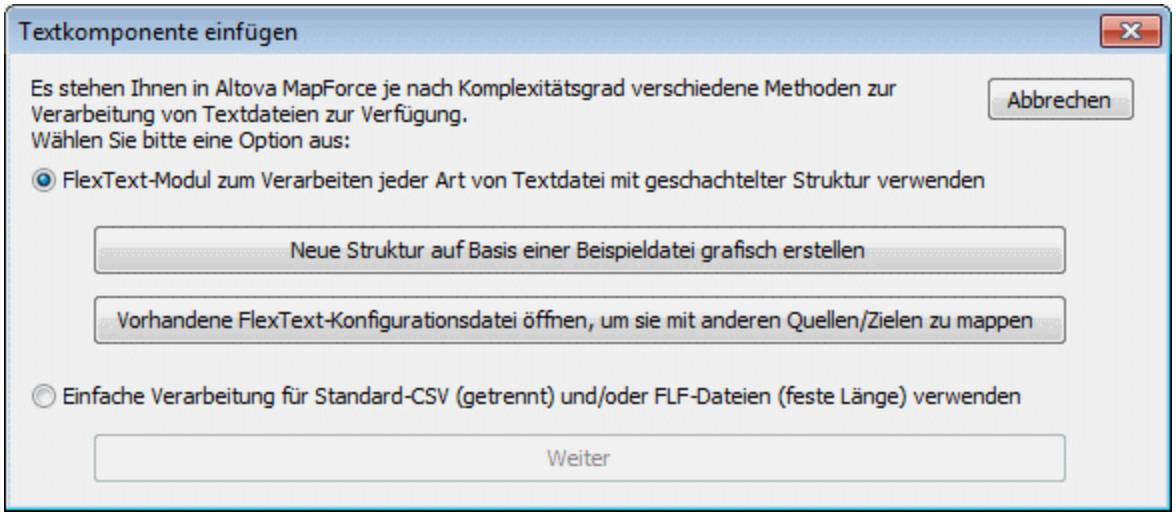
Das Tutorial ist in die folgenden Abschnitte gegliedert:

- Schritt 1: Erstellen der FlexText-Vorlage 687
- Schritt 2: Definieren der Trennbedingungen 690
- Schritt 3: Definieren mehrerer Bedingungen pro Container 692
- Schritt 4: Erstellen der MapForce-Zielkomponente 696
- Schritt 5: Verwenden der FlexText-Vorlage in MapForce 697

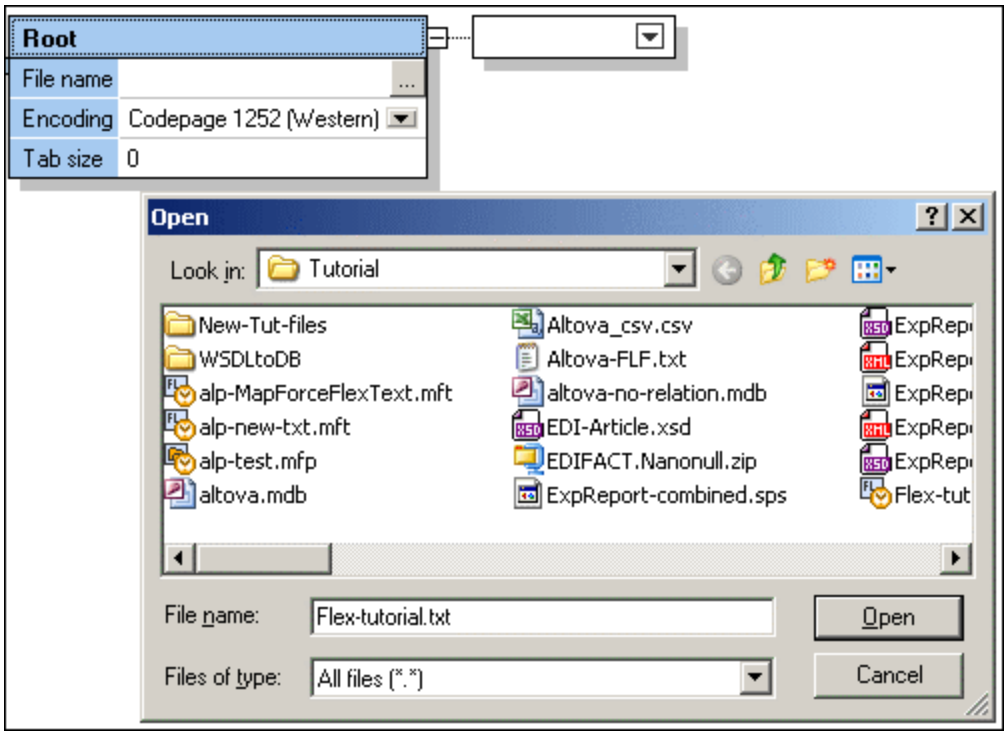
4.6.2.1 Schritt 1: Erstellen der FlexText-Vorlage

Erstellen der FlexText-Vorlage:

1. Starten Sie MapForce und öffnen Sie eine neue Mapping-Datei.
2. Wählen Sie **Einfügen | Textdatei** oder klicken Sie auf das Symbol "Textdatei einfügen" .
3. Klicken Sie auf die Schaltfläche "**Neue Struktur auf Basis einer Beispieldatei grafisch erstellen**".

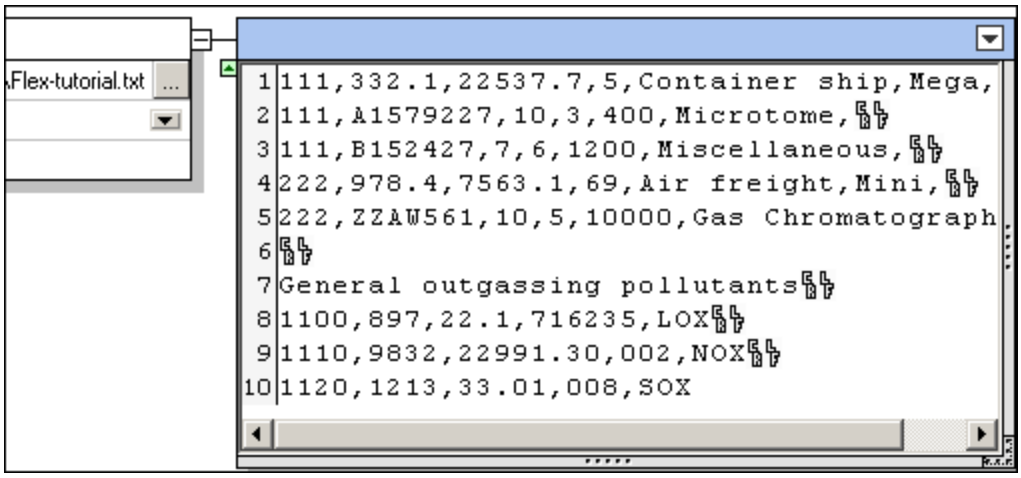


4. Geben Sie einen Namen für Ihre FlexText Vorlage ein und klicken Sie auf "Speichern" (z.B. Flex-tutorial.mft) um fortzufahren. Zusammen mit dem Dialogfeld "Öffnen" wird ein leeres Design angezeigt.



5. Wählen Sie im Ordner **...\MapForceExamples\Tutorial** die Datei **Flex-tutorial.txt** aus und bestätigen Sie den Vorgang durch Klicken auf "Öffnen".

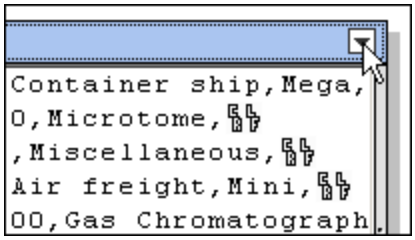
© 2016-2022 Altova GmbH



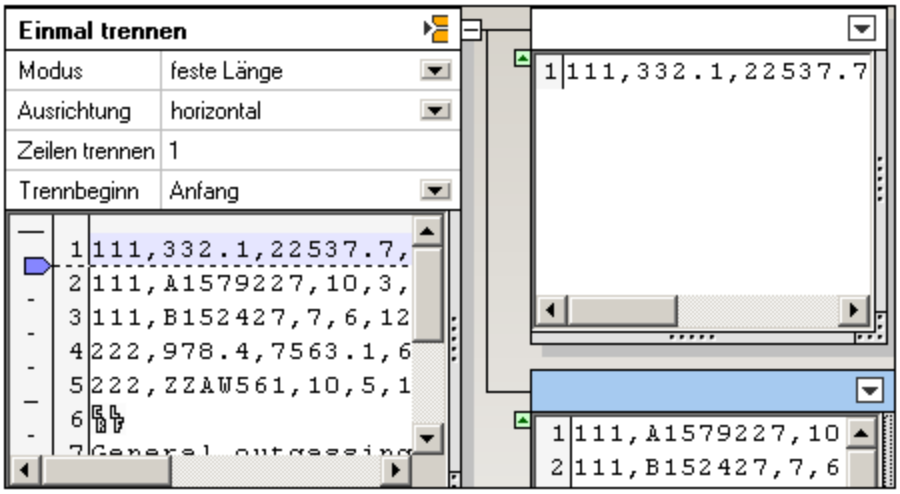
Sie sehen nun den Inhalt der Textdatei.

Wenn Sie auf das Symbol "Node-Text in Design-Ansicht" klicken, wird der Inhalt des aktiven Containers im Beispieltextbereich angezeigt.

Wenn Sie das Symbol "Node-Text für nicht ausgewählte automatisch reduzieren" klicken, wird der Inhalt im aktiven Container angezeigt. Alle anderen Container, die Inhalt enthalten, werden reduziert angezeigt.



6. Klicken Sie auf das Container-Symbol in der rechten oberen Ecke und wählen Sie aus dem Popup-Menü den Befehl **Einmal trennen**.
Neben dem "Einmal trennen"-Container werden zwei neue Container angezeigt. Nähere Informationen zur Bedingung "Einmal trennen" finden Sie unter: Einmal trennen[709].



Sie sehen die Standardeinstellungen des "Einmal trennen"-Containers: feste Länge, horizontal und Zeilen trennen=1.

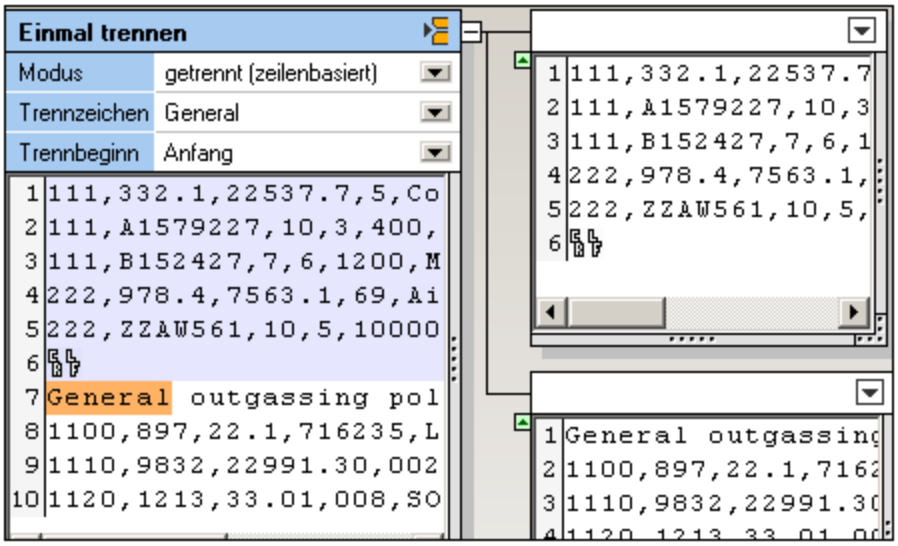
Außerdem sehen Sie auch das Ergebnis dieser Standardeinstellung:

- Der obere Container enthält die erste Zeile der Textdatei, die im Container "**Einmal trennen**" markiert erscheint.
- Der untere Container enthält den Rest der Textdatei.

4.6.2.2 Schritt 2: Definieren von Trennbedingungen

Mit Hilfe von Trennbedingungen können Sie Textfragmente auf verschiedene Arten fragmentieren. So definieren Sie Trennbedingungen:

1. Klicken Sie auf das Listenfeld **Modus** und wählen Sie "getrennt (zeilenbasiert)".
2. Doppelklicken sie auf das Feld "Trennzeichen" und geben Sie "General" ein.



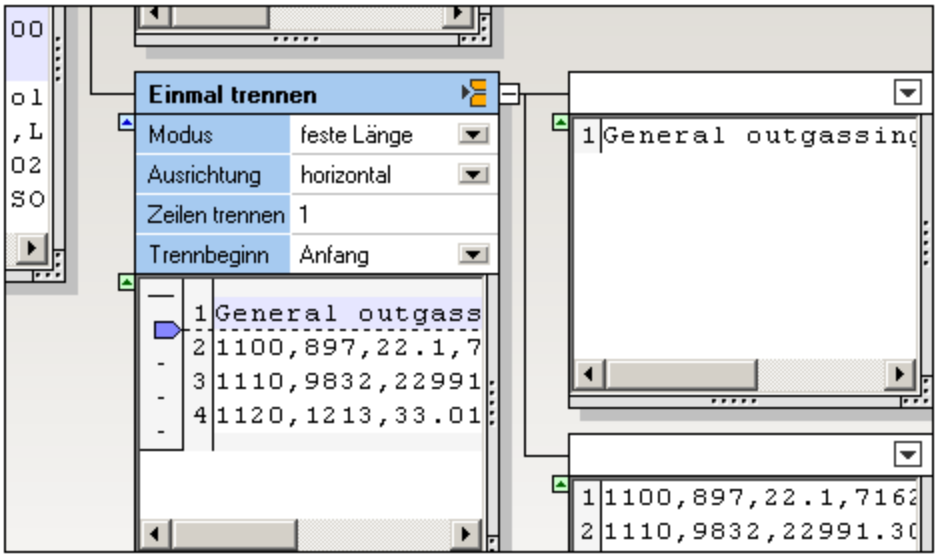
Die Textfragmente in den jeweiligen Containern haben sich nun geändert.

Durch Eingabe von "General" und Verwendung der Option "getrennt (zeilenbasiert)", können Sie den Textabschnitt, der die Zeichenfolge "General" enthält, abtrennen und in den unteren Container platzieren. Das Textfragment bis zum Trennzeichen wird in den oberen Container platziert.

Wir möchten nun den unteren Container bearbeiten, um eine CSV-Datei zu erzeugen, die die Datensätze mit 1100 und darüber enthält.

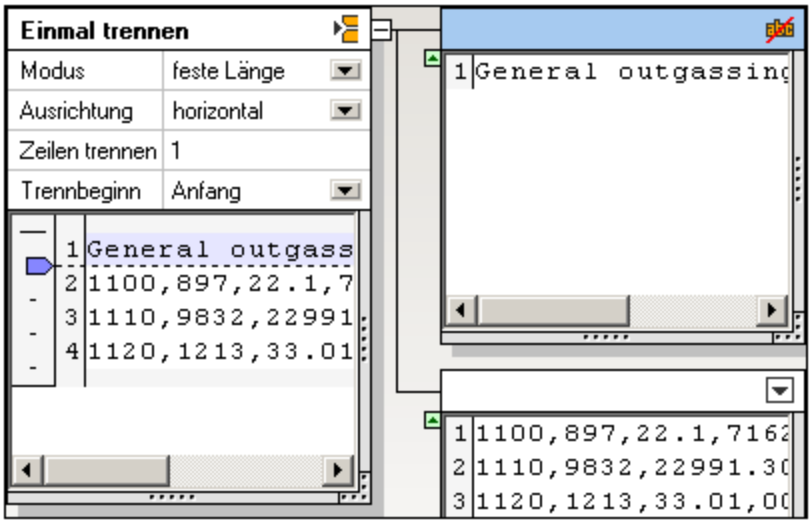
3. Klicken Sie auf den unteren Container und ändern Sie ihn in **Einmal trennen**.

© 2016-2022 Altova GmbH



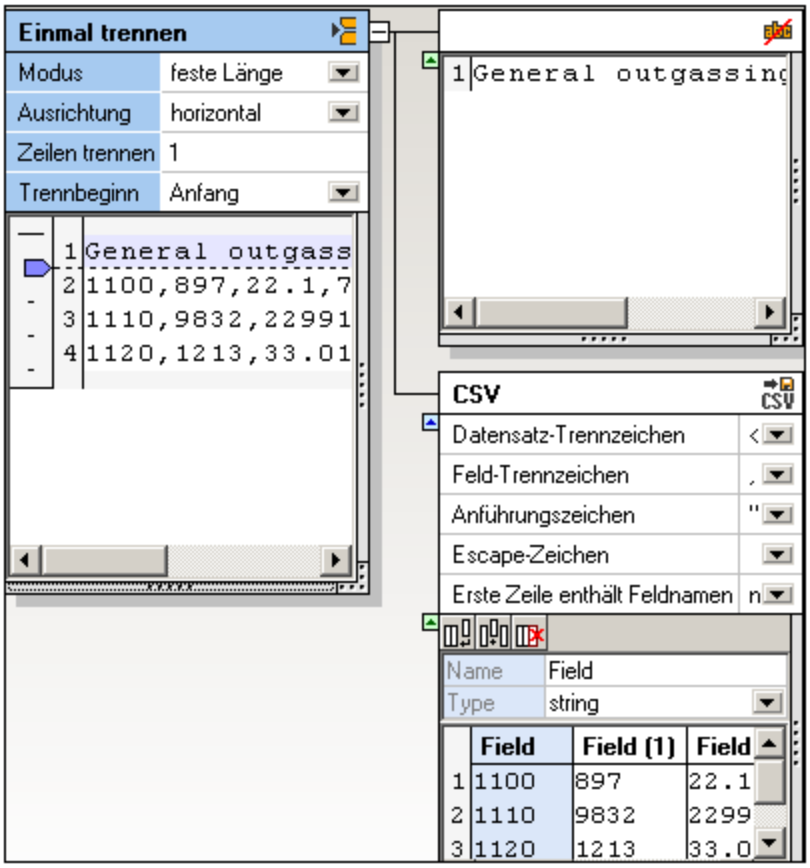
Daraufhin werden zwei neue Container erstellt. Die Standardeinstellungen bleiben unverändert, da wir nur die erste Zeile dieses Textfragments abtrennen und ignorieren möchten. Aus dem restlichen Fragment im unteren Container wird eine CSV-Datei erzeugt.

4. Klicken Sie auf den oberen Container und ändern Sie ihn in **Ignorieren**.



Das Textfragment und somit Mapping-Element dieses Containers steht nun für das Mappen in MapForce nicht mehr zur Verfügung.

5. Klicken Sie auf den unteren Container und ändern Sie ihn in **Als CSV speichern**.



Der Container zeigt nun das Textfragment in Tabellenform an. Die Standardeinstellungen können beibehalten werden.

Konfigurieren der CSV-Datei

Wenn Sie die Feldnamen ändern möchten, klicken Sie in der Tabelle in das Feld und ändern Sie den Eintrag im Feld **Name**. Des Weiteren können in diesem Container Spalten angehängt, eingefügt und gelöscht werden. Nähere Informationen dazu finden Sie unter "Als CSV speichern 721".

Wir können nun mit der Definition des restlichen Textfragments fortfahren.

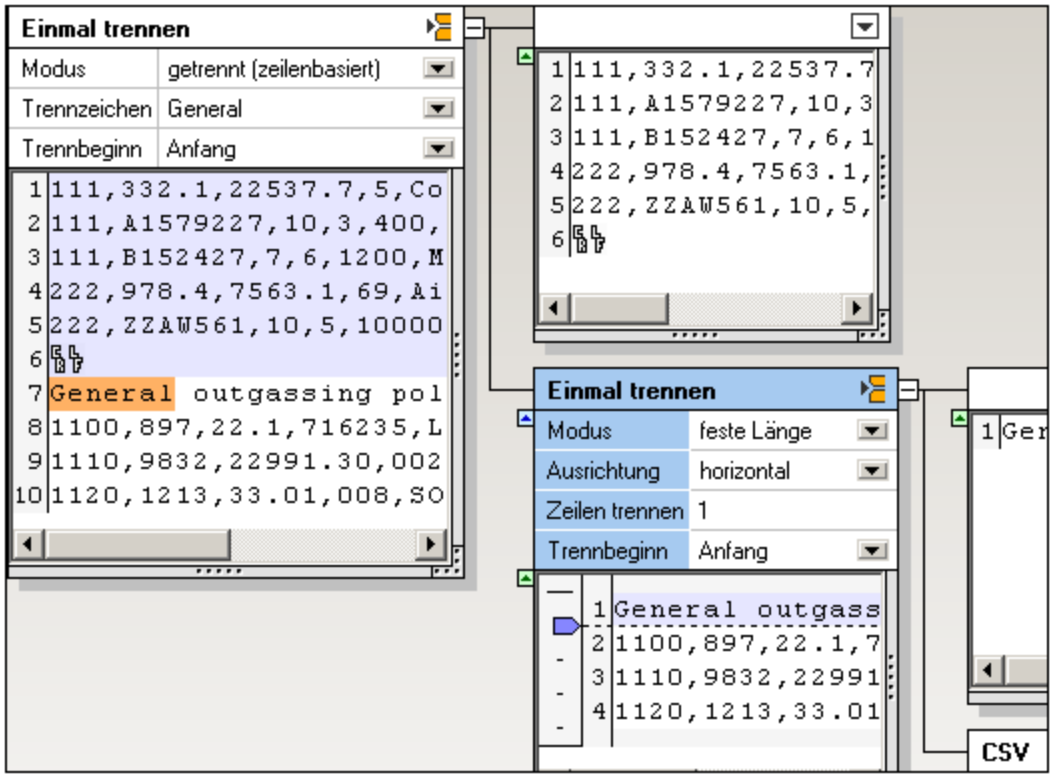
4.6.2.3 Schritt 3: Definieren mehrerer Bedingungen pro Container

Sie können in FlexText mit Hilfe des Containers "Aufteilen" mehrere Bedingungen pro Textfragment definieren. Jede Bedingung, die Sie definieren, erhält automatisch einen dazugehörigen Container.

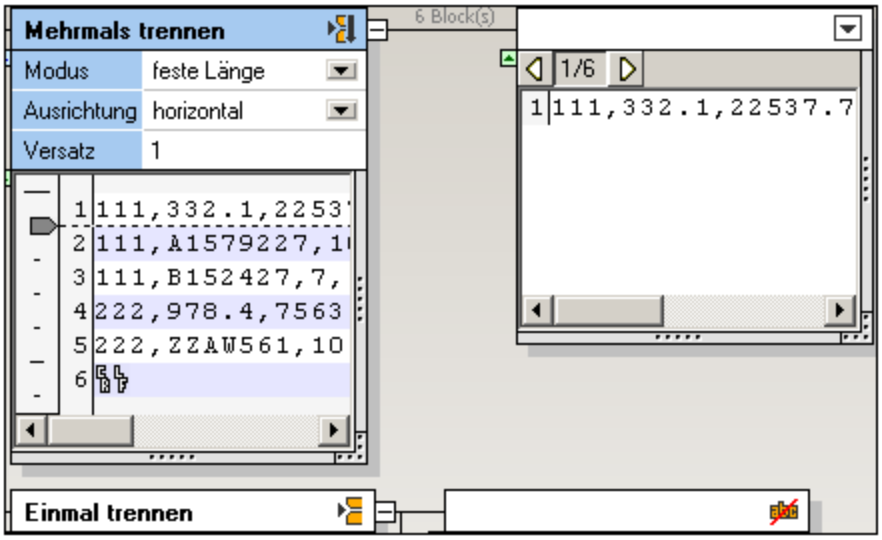
Bis zu diesem Zeitpunkt haben wir in diesem Tutorial das untere Textfragment des ersten **Einmal trennen**-Containers definiert:

- Mit einem "Einmal trennen"-Container wird die erste Zeile abgetrennt und in einen "Ignorieren"-Container platziert.
- Das restliche Segment wird als CSV-Datei definiert/gespeichert.

© 2016-2022 Altova GmbH



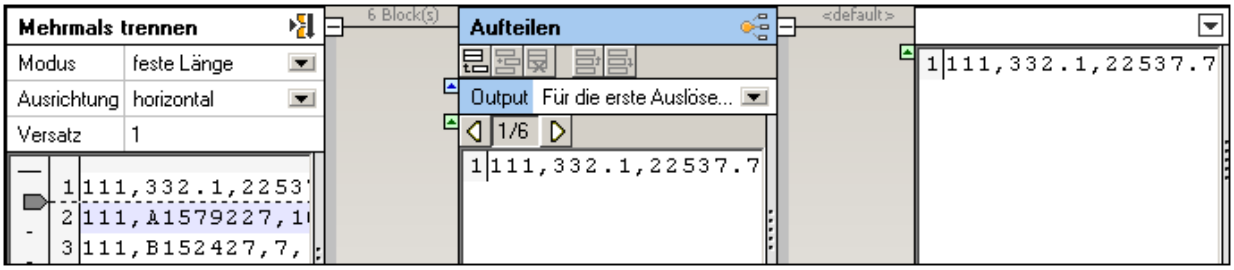
1. Klicken Sie auf das Symbol des oberen Containers und ändern Sie es in **Mehrmals trennen**.



Wir benötigen hier die Standardeinstellungen. Das Textfragment wird in mehrere einzeilige Textblocke aufgeteilt. Im dazugehörigen Container sehen Sie ein Vorschau der einzelnen Textblöcke.

Wenn Sie auf das Symbol "Nächster Textblock" klicken, können Sie der Reihe nach alle 6 Textfragmente anzeigen.

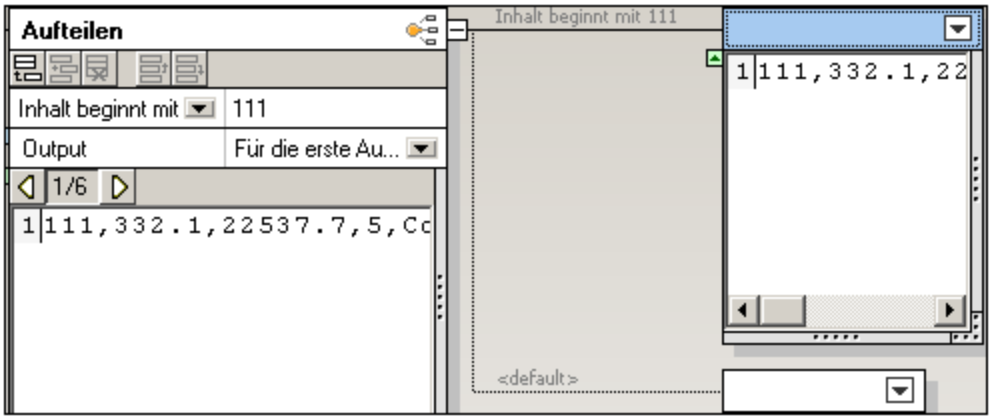
2. Klicken Sie auf den neuen Container und ändern Sie ihn in **Aufteilen**.



In der Abbildung oben sehen Sie den Anfangszustand des "Aufteilen"-Containers.

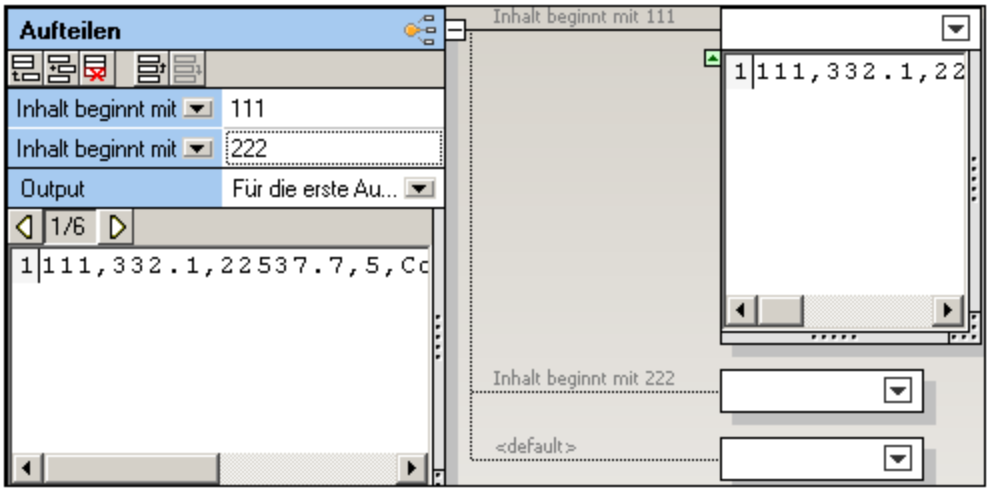
- Es wurde ein dazugehöriger Container "Default" hinzugefügt.
- Der Inhalt des ersten Datensatzes 1/6 wird im Standardcontainer angezeigt.

3. Klicken Sie in der Titelleiste "Aufteilen" auf das Symbol "Bedingung anhängen" , um eine neue Bedingung hinzuzufügen.
4. Doppelklicken Sie in das Feld "Inhalt beginnt mit" und geben Sie die Zahl 111ein.



Damit wird die erste Bedingung definiert. Oberhalb des "Standardcontainers" wurde ein dazugehöriger Container (Inhalt beginnt mit 111) hinzugefügt.

5. Klicken Sie nochmals auf das "Anhängen"-Symbol und geben Sie in das Feld "Inhalt beginnt mit" die Zahl "222" ein.



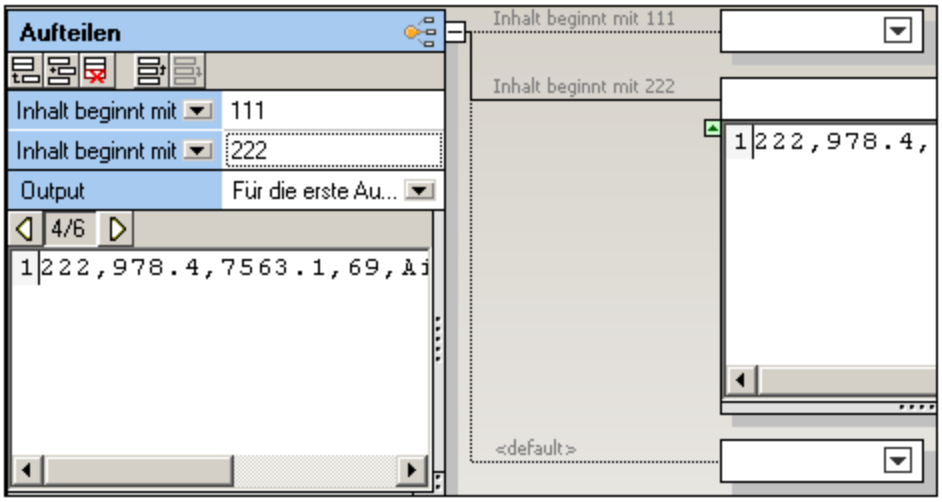
© 2016-2022 Altova GmbH

Es wurde ein dritter Container hinzugefügt (Inhalt beginnt mit 222).

Anmerkung

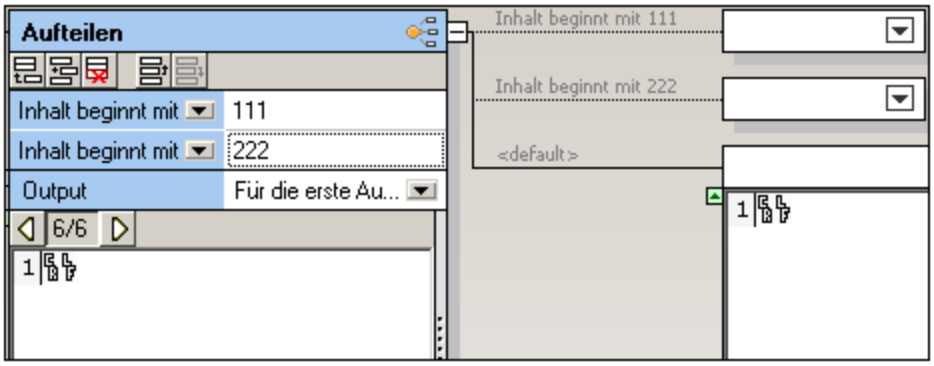
Wenn Sie auf das Listenfeld "Inhalt beginnt mit" klicken, können Sie die Option "Inhalt enthält" auswählen und dadurch eine Zeichenfolge definieren, die an einer beliebigen Stelle im Textfragment vorkommen kann.

6. Klicken Sie mehrmals hintereinander auf das Symbol "Nächster Textblock" , um die Auswirkung zu sehen.



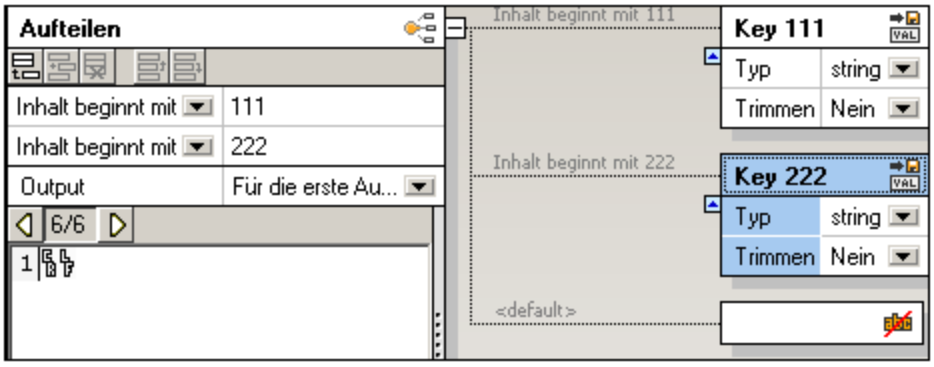
Sobald Sie Datensatz 4 von 6 erreichen, wird der Container 222 geöffnet und dessen Inhalt angezeigt.

7. Klicken Sie weiter, bis Sie Datensatz 6 von 6 erreichen. Im Standardcontainer wird ein einziges CR / LF-Zeichen angezeigt.



Wenn ein Datenfragment im aktuellen Block eine Bedingung erfüllt, so werden **alle Daten** dieses Blocks an den dazugehörigen Container übergeben. Die Daten werden nicht getrennt, sondern nur in den dazugehörigen Container platziert bzw. in den Standardcontainer, wenn keine der definierten Bedingungen erfüllt wird.

8. Klicken Sie auf die beiden ersten Container und ändern Sie sie in **Als Wert speichern**. Klicken Sie auf den letzten Container und ändern Sie ihn in **Ignorieren**.
9. Doppelklicken Sie auf den Text "Speichern" und fügen Sie beschreibenden Text hinzu, z.B. Key 111 und Key 222.

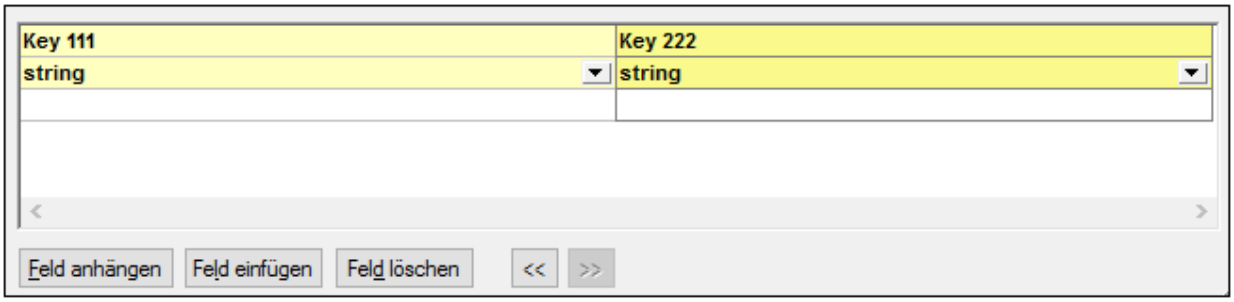


10. Speichern Sie die FlexText-Vorlage, z. B. als Flex-Tutorial.mft.

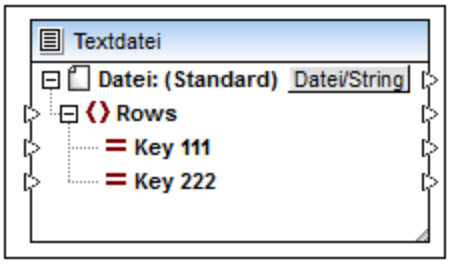
4.6.2.4 Schritt 4: Erstellen der MapForce-Zielkomponente

So erstellen Sie die Zielkomponente für die Verwendung mit der FlexText-Quelle:

1. Wählen Sie den Befehl **Einfügen | Textdatei** oder klicken Sie auf das Symbol "Textdatei" .
2. Klicken Sie auf das Optionsfeld "**Einfache Verarbeitung...verwenden**" und klicken Sie auf **Weiter**. Daraufhin wird das Dialogfeld "Komponenteneinstellungen" geöffnet.
3. Klicken Sie auf die Schaltfläche **Feld anhängen,** um ein neuen Feld anzuhängen.
4. Doppelklicken Sie auf den Feldnamen Field1 und ändern Sie ihn in **Key 111**.

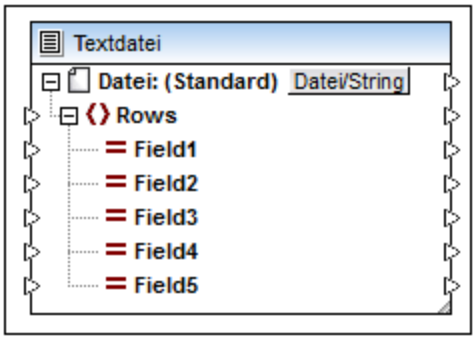


5. Gehen Sie mit dem anderen Feld ebenso vor, nennen Sie es **Key 222** und klicken Sie zur Bestätigung auf OK.
   Es wurde nun eine Textkomponente mit zwei Feldern erstellt.



6. Erstellen Sie auf dieselbe Art eine zweite Textkomponente, die aus fünf Feldern besteht.

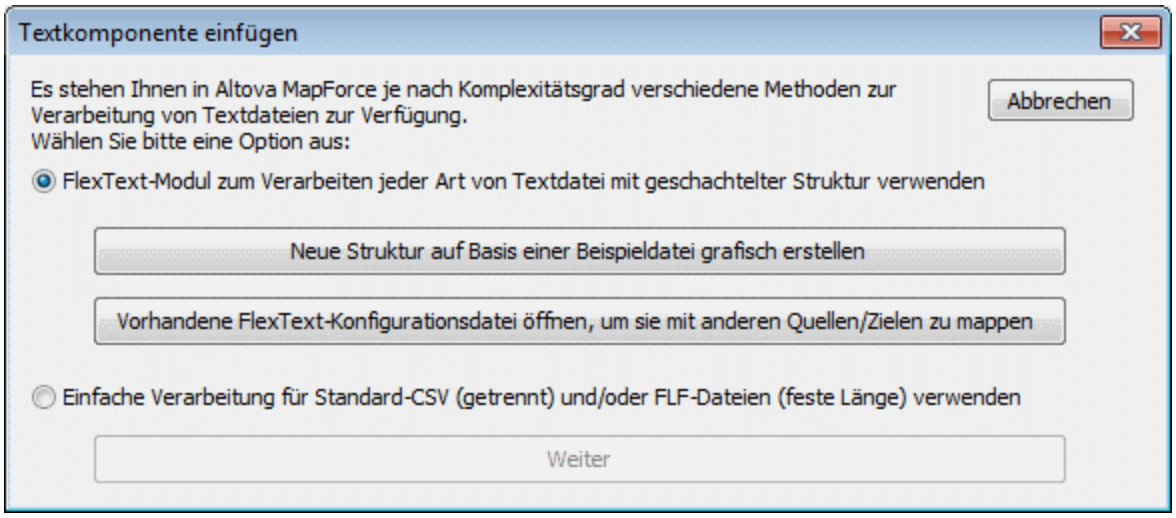
© 2016-2022 Altova GmbH



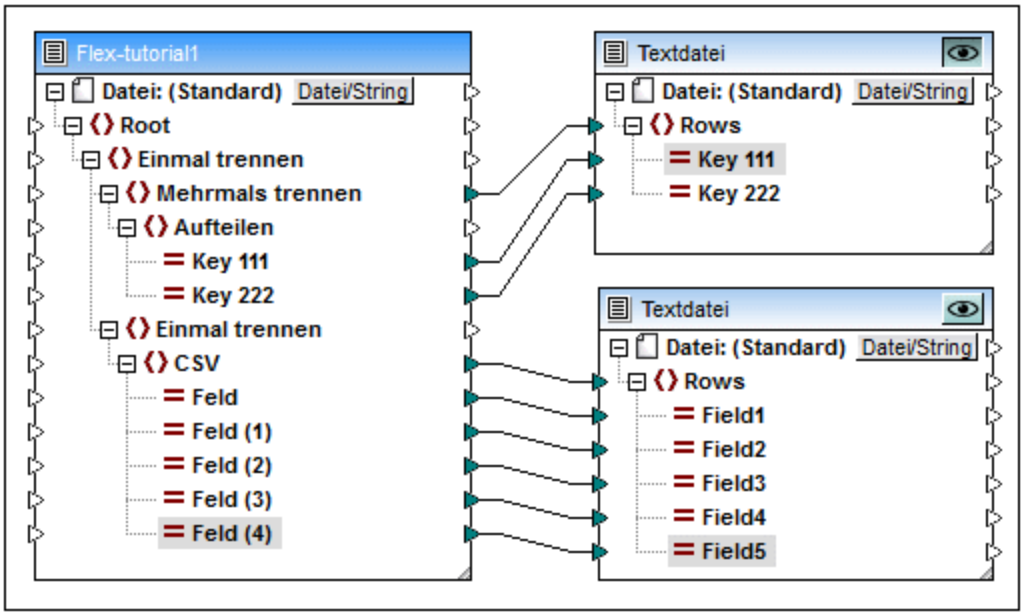
4.6.2.5 Schritt 5: Verwendung der FlexText-Vorlagen in MapForce

So verwenden Sie die FlexText-Vorlage in MapForce:

1. Starten Sie MapForce bzw. wechseln Sie zurück zu MapForce und wählen Sie **Einfügen | Textdatei**.



2. Klicken Sie auf die Schaltfläche **Vorhandene FlexText-Konfigurationsdatei öffnen,...,** und wählen Sie die zuvor definierte FlexText-Vorlage (**Flex-tutorial.mft**) aus.
   Die Struktur der MapForce-Komponente spiegelt die der Container in der Design-Ansicht in FlexText wider.



3. Mappen Sie die verschiedenen Elemente auf die zuvor definierten Zielkomponenten und klicken Sie auf das Register "Ausgabe-Vorschau", um eine Vorschau der Ergebnisse zu sehen.

Mapping-Vorschau der oberen Textkomponente:

```
"111,332.1,22537.7,5,Container ship,Mega,
",
"111,A1579227,10,3,400,Microtome,
",
"111,B152427,7,6,1200,Miscellaneous,
",
,"222,978.4,7563.1,69,Air freight,Mini,
"
,"222,ZZAW561,10,5,10000,Gas Chromatograph,,
"
```

Mapping-Vorschau der unteren Textkomponente:

```
1100,897,22.1,716235,LOX
1110,9832,22991.30,002,NOX
1120,1213,33.01,008,SOX
```

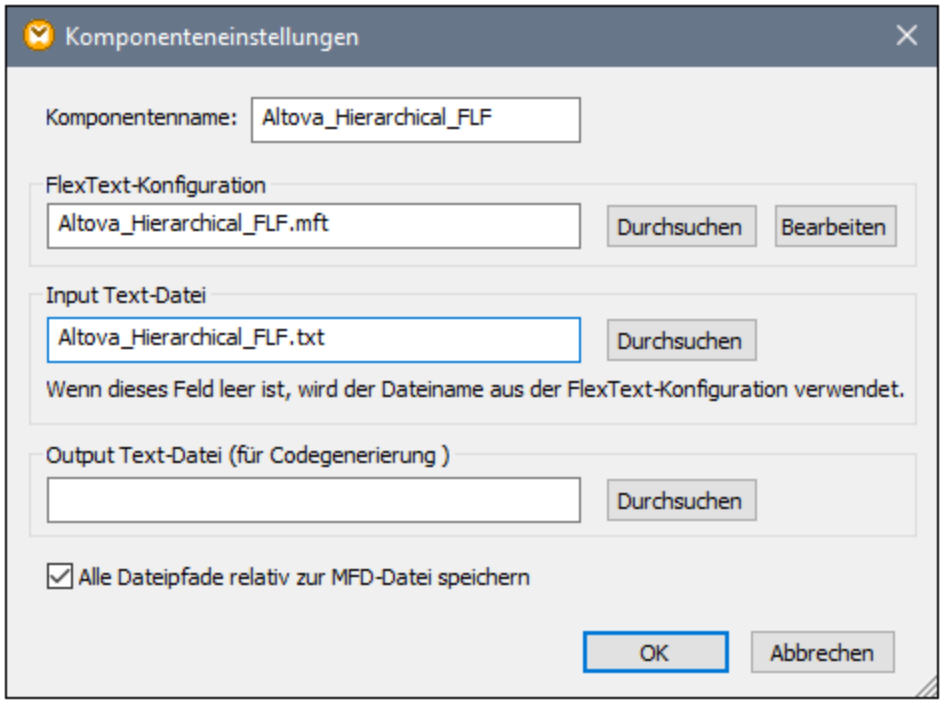
4.6.3 FlexText-Komponenteneinstellungen

Nachdem Sie eine FlexText-Komponente zum Mapping-Bereich hinzugefügt haben, können Sie die Einstellungen dafür im Dialogfeld "Komponenteneinstellungen" konfigurieren. Sie können das Dialogfeld "Komponenteneinstellungen" auf eine der folgenden Arten öffnen:

- Klicken Sie im Menü **Komponente** auf den Befehl **Eigenschaften** (dieses Menü wird aktiv, wenn Sie eine Komponente auswählen).
- Doppelklicken Sie auf den Komponententitel.

© 2016-2022 Altova GmbH

- Klicken Sie mit der rechten Maustaste auf den Komponententitel und wählen Sie anschließend den Befehl **Eigenschaften.**



FlexText-Dialogfeld "Komponenteneinstellungen"
Es stehen die folgenden Einstellungen zur Verfügung.

Komponentenname	Der Komponentenname wird bei der Erstellung der Komponente automatisch generiert. Sie können den Namen jedoch jederzeit ändern. Wenn der Komponentenname automatisch generiert wurde und Sie danach eine Instanzdatei auswählen, fragt Sie MapForce, ob der Komponentenname ebenfalls entsprechend aktualisiert werden soll. Der Komponentenname kann Leerzeichen und Punkte enthalten, darf aber keine Schrägstriche, umgekehrten Schrägstriche, Doppelpunkte, doppelten Anführungszeichen und voran- oder nachgestellte Leerzeichen enthalten. Beachten Sie beim Ändern des Namens einer Komponente Folgendes: • Wenn Sie das Mapping auf FlowForce Server bereitstellen möchten, muss der Komponentenname eindeutig sein. • Es wird empfohlen, nur Zeichen zu verwenden, die über die Befehlszeile eingegeben werden können. Länderspezifische Sonderzeichen sind in Windows und der Befehlszeile eventuell unterschiedlich kodiert.

FlexText-Konfigurationsdatei	Definiert den Namen und Pfad der FlexText-Vorlagendatei (.mft), die von der Komponente verwendet wird. Um eine neue Vorlage auszuwählen, klicken Sie auf **Durchsuchen**. Um die Vorlage in FlexText zu bearbeiten, klicken Sie auf **Bearbeiten.**
Input-Textdatei	Definiert die Textdatei, aus der MapForce die Daten ausliest. Dieses Feld hat bei einer Quellkomponente eine Bedeutung und wird ausgefüllt, wenn Sie die Komponente erstellen und ihr eine Textdatei zuweisen. Die hier definierte Text-Quelldatei hat Vorrang vor der in der FlexText-Vorlage definierten Datei. Wenn dieses Feld leer ist, wird der in der FlexText-Vorlage definierte Dateiname als Quelle verwendet.
Output-Textdatei (für die Codegenerierung)	Definiert den Namen und Pfad der Textdatei, in die MapForce die Daten schreibt. Dieses Feld hat bei einer Zielkomponente eine Bedeutung, wenn Code generiert wird. Wenn Sie den vollständigen Pfad eingeben, können Sie das Zielverzeichnis, z.B. **c:\myfiles\sequence.txt,** definieren.
Alle Dateipfade relativ zur MFD-Datei speichern	Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld "Komponenteneinstellungen" angezeigten Dateipfade relativ zum Ordner, in dem sich die MapForce Design (.mfd)-Datei befindet.

4.6.4 Verwenden von FlexText als Zielkomponente

Hauptsächlich werden FlexText-Komponenten verwendet, um Textdateien zu reduzieren oder Abschnitte daraus herauszutrennen und die relevanten Einträge der FlexText Komponente auf andere Zielkomponenten zu mappen. FlexText kann jedoch auch als Zielkomponente verwendet werden, um separate Dateien zu einer einzigen Datei zusammenzusetzen bzw. neu zusammenzusetzen. Oft sind jedoch einige Versuche erforderlich, um das gewünschte Ergebnis zu erzielen.

Wenn Sie eine FlexText Komponente als **Zielkomponente** verwenden, werden die zuvor darin definierten Operationen umgekehrt. Anstatt eine Datei in verschiedene Unterabschnitte aufzuteilen, wird sie zu einer einzigen Datei zusammengesetzt bzw. erneut zusammengesetzt.

Im Allgemeinen werden die einzelnen in der FlexText Vorlage definierten Operationen (von unten nach oben) umgekehrt, wenn sie als Ziel verwendet werden:

- eine Trennung wird zu einer Zusammenführung, so wird z.B. ein Mapping auf wiederholt durch "," getrennte Dateien zu einer Zusammenführung von Datenelementen, die durch "," getrennt sind.
- aus "Speichern" wird "Laden"
- aus "Aufteilen" wird "Wähle die erste Übereinstimmung"

Beachten Sie beim Verwenden von FlexText als Zielkomponente die folgenden Punkte:

© 2016-2022 Altova GmbH

- Sobald eine Verbindung zwischen einer Datenquell-Komponente und einem der Input-Datenelemente einer FlexText Komponente hergestellt wurde, wird die Datenquelle der FlexText Komponente ignoriert. Vorrang haben nun die Daten aus der neu gemappten Quellkomponente.
- Wenn Text auf einen "Speichern als..."-Container gemappt wird (Als CSV und FLF speichern), wird das Trennzeichen beibehalten. Es kann jedoch vorkommen, dass Text abgeschnitten wird, wenn in einem Node oberhalb des "Speichern als..." Node eine Trennung mit fester Länge vorkommt.
- Bei Trennungen mit fester Breite wird der linke/obere Abschnitt auf die vordefinierte Länge abgeschnitten, wenn Trennbeginn=Anfang, bzw. der rechte/untere Abschnitt bei Trennbeginn=Ende. Der abgeschnittene Abschnitt entspricht der in Zeichen definierten Länge. Wenn der Text zu kurz ist, werden Leerzeichen als Füllzeichen eingefügt.
- FlexText würde normalerweise Trennzeichen (bzw. bei Trennungen mit fester Länge Leerzeichen) zwischen den Elementen einer Trennungsoperation einfügen. Dies ist allerdings bei Trennungen mit der Operation 'getrennt (zeilenbasiert)' nicht der Fall. Die Operation 'getrennt (zeilenbasiert)' kann nicht wieder vollständig rückgängig gemacht werden. Der "getrennte" Text kann an jeder beliebigen Stelle in der ersten Zeile vorkommen und ist im Text inkludiert, daher kann der Text durch einen automatischen Vorgang nicht zuverlässig hinzugefügt werden.
  - mit "getrennt (zeilenbasiert)" wird kein Trennzeichen zur ersten Zeile hinzugefügt, wenn es fehlt.
  - mit "getrennt (beliebig)" wird ein Trennzeichen zwischen zwei Abschnitten hinzugefügt.
- Die Operation "Aufteilen" kann - außer in einfachen Fällen - nicht auf sinnvolle Weise umgekehrt werden. Dabei werden die Verzweigungen nach der ersten Verzweigung durchsucht, die Daten enthält, und diese Daten werden verwendet/einfügt.
  Nur die erste Verbindung einer Aufteilungsoperation wird gemappt. Um Daten in die restlichen "Aufteilen"-Container zu transferieren, müssen Filter für die restlichen Konnektoren definiert werden und das übergeordnete Datenelement der Operation "Aufteilen" muss dupliziert werden, sodass jedes "Aufteilen"-Datenelement ein einziges Datenelement zurückgibt, welches anschließend in ein wiederholtes "Trenn"-Datenelement eingespeist wird, um alle Elemente zusammenzuführen.
- Wenn auf einen Subeintrag eines einzigen Trenn-Containers gemappt wird, werden alle Mapping-Ergebnisse mit Ausnahme des letzten Datenelements verworfen. Nur ein einziges Ergebnis wird beibehalten, selbst wenn mehrere Ergebnisse generiert wurden.

Die folgende Analogie zum XML-Schema Content Model zeigt, wie FlexText sich verhält, wenn es als Zielkomponente verwendet wird:

- eine wiederholte Trennung ist ein wiederholbares Element.
- eine einzige Trennung bildet Teil einer 'sequence' Content Model-Gruppe.
- Bei einer Aufteilung wird eine 'choice' Content Model-Gruppe gebildet, wobei es sich bei jedem Fall um ein mögliches Subelement handelt.
- Mit "Speichern" wird ein Element vom Typ simpleType erstellt.

4.6.5 FlexText Referenz

Im Referenzabschnitt werden die verschiedenen Funktionen von FlexText beschrieben und Sie erfahren wie Sie sie verwenden, um bestimmte Ergebnisse zu erzielen.

4.6.5.1 Mehrmals trennen

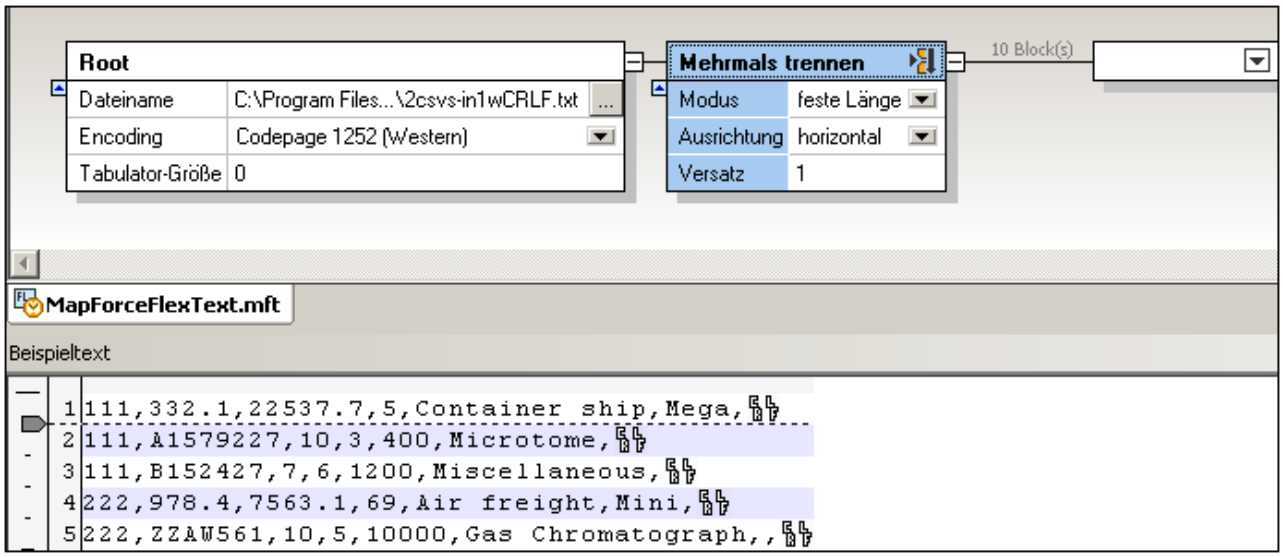


Wenn Sie diese Option am Anfang verwenden, wird ein einzelner Container erstellt. Der Container enthält den Text, der durch die Bedingung definiert wird, die in **Mehrmals trennen** eingestellt ist. Es gibt verschiedene Versionen der Option "Mehrmals trennen": Feste[703] Länge, getrennt (beliebig)[704], getrennt zeilenbasiert[706] und getrennt Zeile beginnt mit...[707].

Wenn Sie diese Option zum ersten Mal verwenden, werden automatisch Standardparameter verwendet und die erzeugten Fragmente werden im dazugehörigen Container angezeigt. Beachten Sie, dass der "Mehrmals trennen"-Container derzeit aktiv ist, wobei im Beispieltextbereich eine Vorschau aller aktuellen Datensätze/Zeilen angezeigt wird.

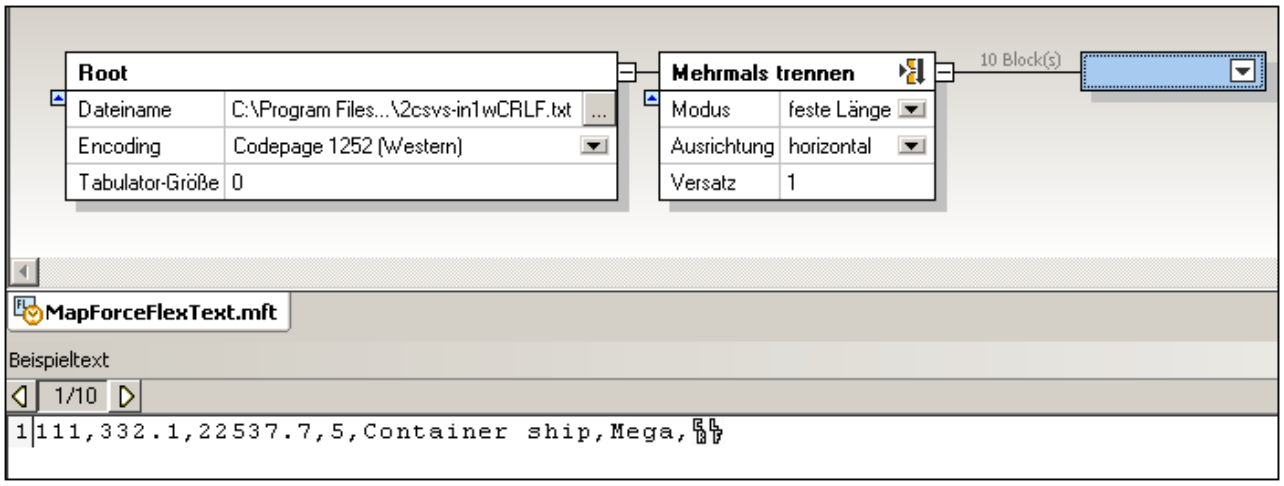
Die **Standardeinstellungen** für den Container sind:

Modus	feste Länge
Ausrichtung	horizontal
Versatz	1



Standardergebnis:
Jede Textzeile erscheint als Zeile/Datensatz im neuen Container, da als Versatz der Wert 1 angenommen wird. Klicken Sie auf den neuen Container, um eine Vorschau des Inhalts zu sehen. Mit Hilfe der **Bildlaufpfeile** im Beispieltextbereich können Sie durch jeden/jedes der 10 mit diesen Einstellungen erzeugten Blöcke/Fragmente scrollen.

© 2016-2022 Altova GmbH



4.6.5.1.1 Modus - Feste Länge

Verwenden Sie den Modus **Mehrmals trennen (feste Länge),** um den Text in mehrere horizontale oder vertikale Fragmente mit einer festen Länge zu teilen. Im Folgenden sind die Einstellungen für den Modus **Mehrmals trennen (feste Länge)** beschrieben.

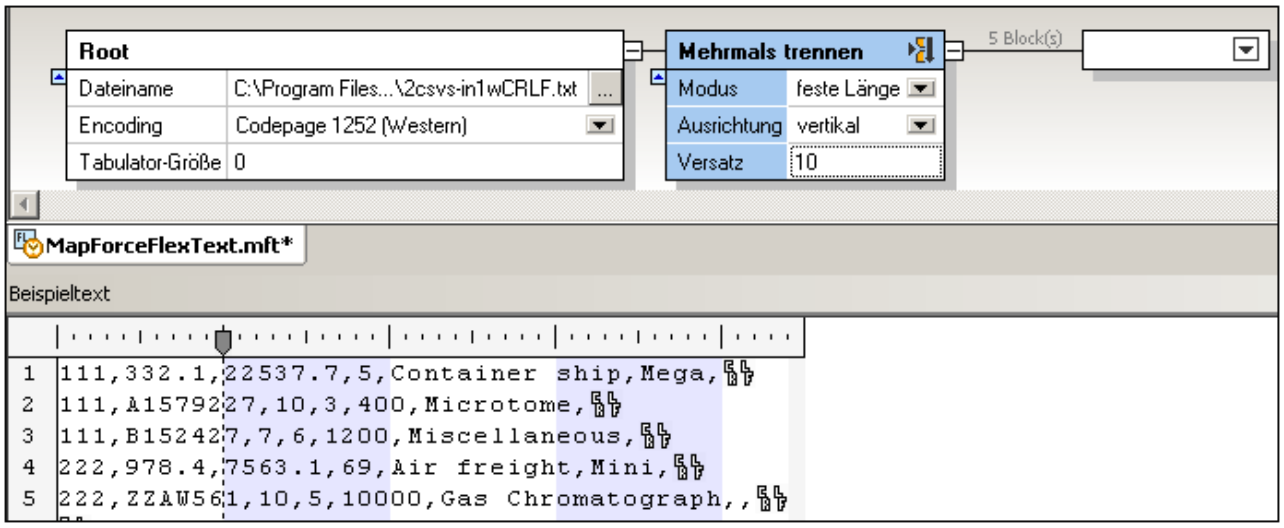
Ausrichtung

Dient dazu festzulegen, ob das Textfragment horizontal oder vertikal getrennt werden soll. Wählen Sie "Horizontal", um das Fragment in mehrere horizontale Abschnitte zu trennen. Wählen Sie "Vertikal", um das Fragment in mehrere vertikale Spalten zu trennen. Die Standardausrichtung ist "Horizontal".

Versatz

Wenn als Ausrichtung "Horizontal" ausgewählt ist, wird mit dieser Einstellung die Anzahl der Zeilen definiert, die jedes Fragment enthalten soll. Wenn als Ausrichtung "Vertikal" ausgewählt ist, wird mit dieser Einstellung die Breite (in Zeichen) der einzelnen Fragmente definiert. Der Standardversatz ist "1". Um den Versatz zu ändern, wählen Sie eine der folgenden Methoden:

- Geben Sie einen Wert in das Feld "Versatz" ein
- Ziehen Sie den Tabulator am vertikalen bzw. horizontalen Lineal.



4.6.5.1.2 Modus - Getrennt (beliebig)

Verwenden Sie den Modus **Mehrmals trennen - getrennt (beliebig)** in den folgenden Situationen:

- um Text zu trennen, bei dem die von Ihnen definierten Trennzeichen aus den erzeugten Fragmenten entfernt werden müssen
- um Text zu trennen, bei dem sich die Trennzeichen innerhalb der Zeile befinden (z.B. Text, der keine CR/LF-Zeichen enthält)

Anmerkung: Ein Fragment ist definiert als der Text zwischen dem ersten Zeichen **nach** dem Trennzeichen bis zum letzten Zeichen vor der nächsten Instanz desselben Trennzeichens. Eine Ausnahme für diese Regel bilden das erste und das letzte Fragment (siehe Beispiel unten).

Im Folgenden sind die Einstellungen zum Modus **Mehrmals trennen - getrennt (beliebig)** beschrieben.

Regular Expression

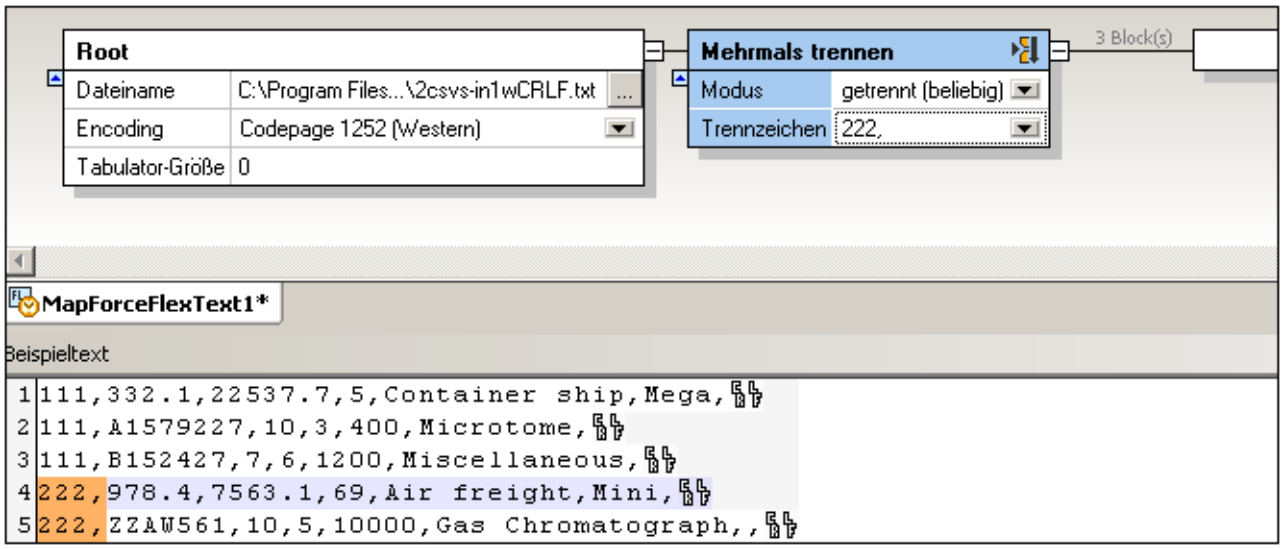
Dies ist eine optionale Einstellung, mit der Text an den Stellen, an denen eine Übereinstimmung mit einer Regular Expression, die definiert wurde, gefunden wird, getrennt wird (siehe Trennen von Text mit Hilfe von Regular Expressions 731). Der Standardwert ist "nein".

Trennzeichen

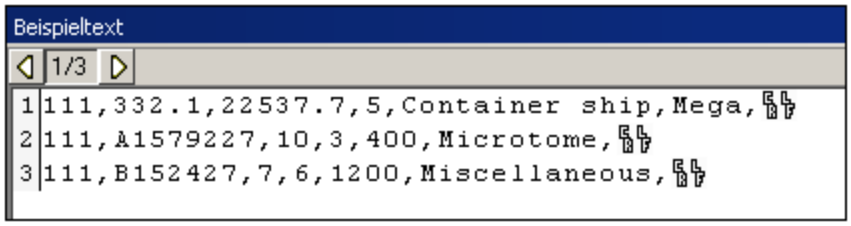
Definiert welche(s) Zeichen als Trennzeichen verwendet werden soll(en). Der Standardwert ist "keines" (kein Trennzeichen).

Wenn Sie im unten gezeigten Text als Trennzeichen "**222,**" verwenden, werden drei separate Fragmente erzeugt:

© 2016-2022 Altova GmbH



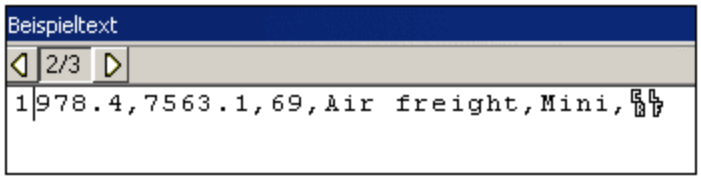
Das erste Fragment enthält alle Zeichen vom **Beginn** des Fragments bis zum Beginn des **ersten Trennzeichens** ("222,"), also von "111"... bis "Miscellaneous,".



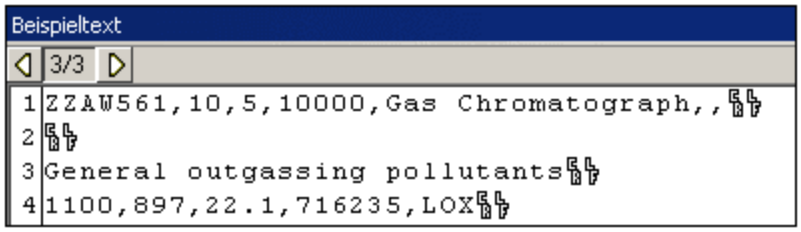
Wenn das Trennzeichen, wie in diesem Beispiel, nicht die erste Zeichengruppe der ersten Zeile im Fragment ist, enthält das erste Fragment den gesamten Text **bis zur** ersten Instanz des Trennzeichens, also bis "222".

Wäre "111" das Trennzeichen, dann wäre das erste Fragment ein String mit der Länge Null, da das Trennzeichen am Anfang der ersten Zeile des Quellfragments aufscheint.

Das zweite Fragment enthält die erste Zeile, die das Trennzeichen 222 enthält, ohne das Trennzeichen.



Das dritte Fragment enthält die **nächste** Zeile, die das Trennzeichen 222 enthält, ohne das Trennzeichen selbst, und zwar bis zum Ende der Textdatei/des Textfragments.



4.6.5.1.3 Modus - Getrennt (zeilenbasiert)

Verwenden Sie den Modus **Mehrmals trennen - getrennt (zeilenbasiert),** um Text in mehrere Fragmente mit dem folgenden Verhalten zu trennen:

- Mit dieser Option werden mehrere Fragmente erstellt, die durch Trennzeichen definiert werden, die Sie in das Feld "Trennzeichen" eingeben.
- Die Trennzeichen werden in das Fragment inkludiert.
- Ein Fragment wird definiert als die gesamte Zeile, die das Trennzeichen enthält, bis zur nächsten Zeile, die dasselbe Trennzeichen enthält.
- Wenn das Trennzeichen in der ersten Zeile nicht vorkommt, so enthält das erste Fragment die Zeile(n) bis zur ersten Zeile, die das Trennzeichen enthält.

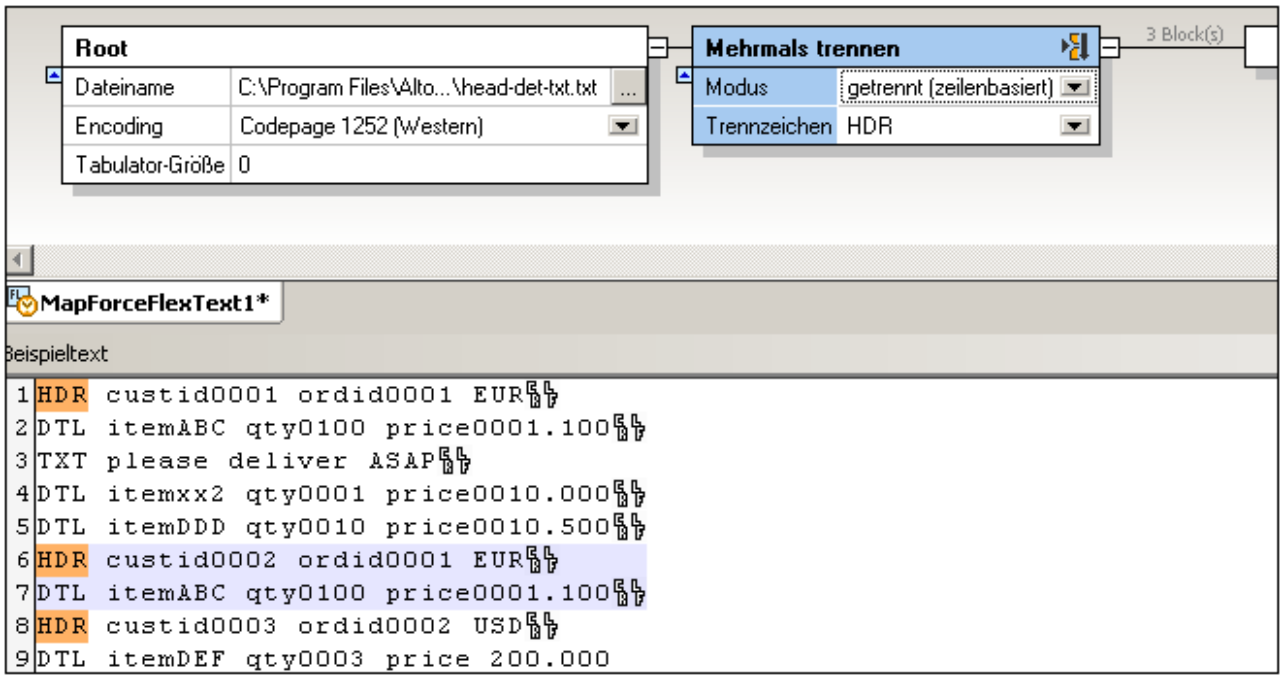
Im Folgenden sind die Einstellungen zum Modus **Mehrmals trennen - getrennt (zeilenbasiert)** beschrieben.

Regular Expression

Dies ist eine optionale Einstellung, mit der Text an den Stellen, an denen eine Übereinstimmung mit einer Regular Expression, die definiert wurde, gefunden wird, getrennt wird (siehe Trennen von Text mit Hilfe von Regular Expressions 731). Der Standardwert ist "nein".

Trennzeichen

Definiert welche(s) Zeichen als Trennzeichen verwendet werden soll(en). Der Standardwert ist "keines" (kein Trennzeichen).
Bei Verwendung des Trennzeichens "**HDR**" für den unten gezeigten Text, werden drei separate Fragmente erzeugt.



 © 2016-2022 Altova GmbH

Das erste Fragment enthält alle Zeichen ab dem Anfang der Datei/des Fragments einschließlich aller Zeilen bis zur nächsten Zeile, die dasselbe Trennzeichen enthält.

```
1 HDR custid0001 ordid0001 EUR
2 DTL itemABC qty0100 price0001.100
3 TXT please deliver ASAP
4 DTL itemxx2 qty0001 price0010.000
5 DTL itemDDD qty0010 price0010.500
```

Beachten Sie, dass Sie mit dieser Option jede beliebige Anzahl von Zeilen zwischen zwei Trennzeichen abrufen können. Dies eignet sich für Dateien mit Datensatztypen, die optional sind oder nicht hintereinander vorkommen (z.B. "DTL" oder "TXT" in diesem Beispiel)

Das zweite Fragment enthält den gesamten Text ab der zweiten Instanz von "HDR" bis zur nächsten Instanz von "HDR".

```
1 HDR custid0002 ordid0001 EUR
2 DTL itemABC qty0100 price0001.100
```

Das dritte Fragment enthält den gesamten Text ab der dritten Instanz von "HDR" bis zum Ende.

```
1 HDR custid0003 ordid0002 USD
2 DTL itemDEF qty0003 price 200.000
```

4.6.5.1.4 Modus - Getrennt (Zeile beginnt mit)

Verwenden Sie den Modus **Mehrmals trennen - getrennt (Zeile beginnt mit),** um Text in mehrere Fragmente mit dem folgenden Verhalten zu trennen:

- In diesem Modus werden mehrere Fragmente erstellt, die Sie durch Trennzeichen definieren, die Sie in das Feld "Trennzeichen" eingeben.
- Die Trennzeichen werden in das Fragment inkludiert.
- Ein Fragment wird definiert als die gesamte Zeile, beginnend mit dem Trennzeichen bis zur nächsten Zeile, die das gleiche Trennzeichen am Beginn der Zeile enthält.
- Wenn in der ersten Zeile kein Trennzeichen vorkommt, so enthält das erste Fragment die Zeile(n) bis zur ersten Zeile, die das Trennzeichen enthält.

Im Folgenden sind die Einstellungen zum Modus **Mehrmals trennen - getrennt (Zeile beginnt mit)** beschrieben.

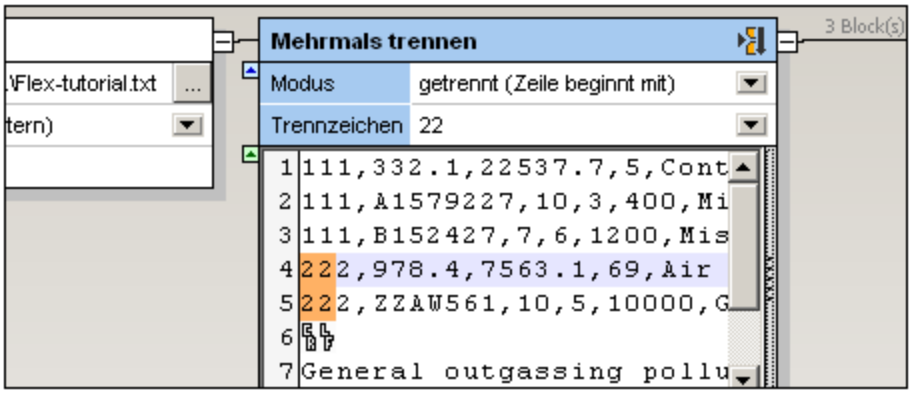
Regular Expression

Dies ist eine optionale Einstellung, mit der Text an den Stellen, an denen eine Übereinstimmung mit einer Regular Expression, die definiert wurde, gefunden wird, getrennt wird (siehe Trennen von Text mit Hilfe von Regular Expressions 731). Der Standardwert ist "nein".

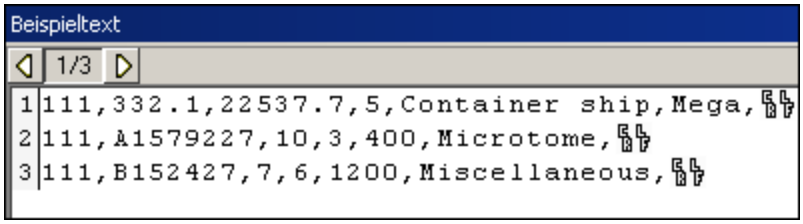
Trennzeichen

Definiert welche(s) Zeichen als Trennzeichen verwendet werden soll(en). Der Standardwert ist "keines" (kein Trennzeichen).

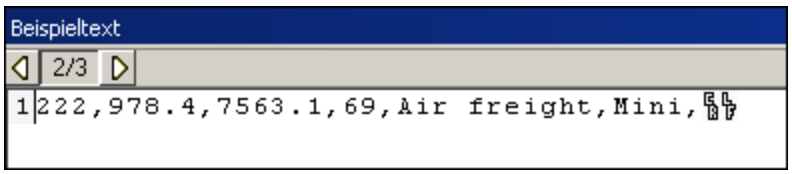
Bei Verwendung des Trennzeichens "**22**" erhalten Sie mit dem unten gezeigten Text drei separate Fragmente.



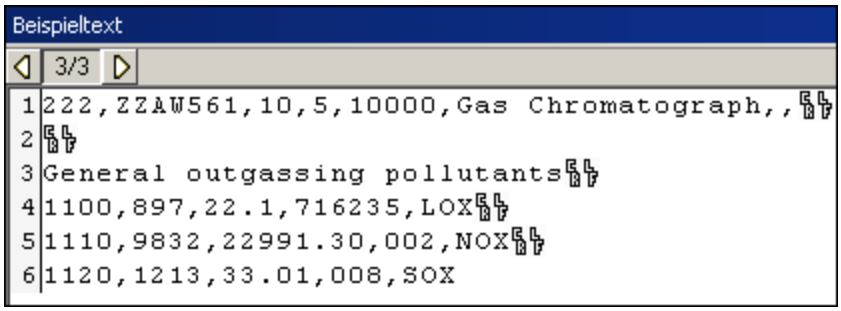
Das erste Fragment enthält alle Zeichen vom Beginn der Zeile/des Fragments bis zu der Zeile, die das Trennzeichen "22" enthält.



Das zweite Fragment enthält alle Zeichen/Zeilen ab der Stelle, an der das Trennzeichen "22" das erste Mal vorkommt, bis zu der Stelle, an der "22" das nächste Mal vorkommt. Dies ist in diesem Fall nur eine Zeile.

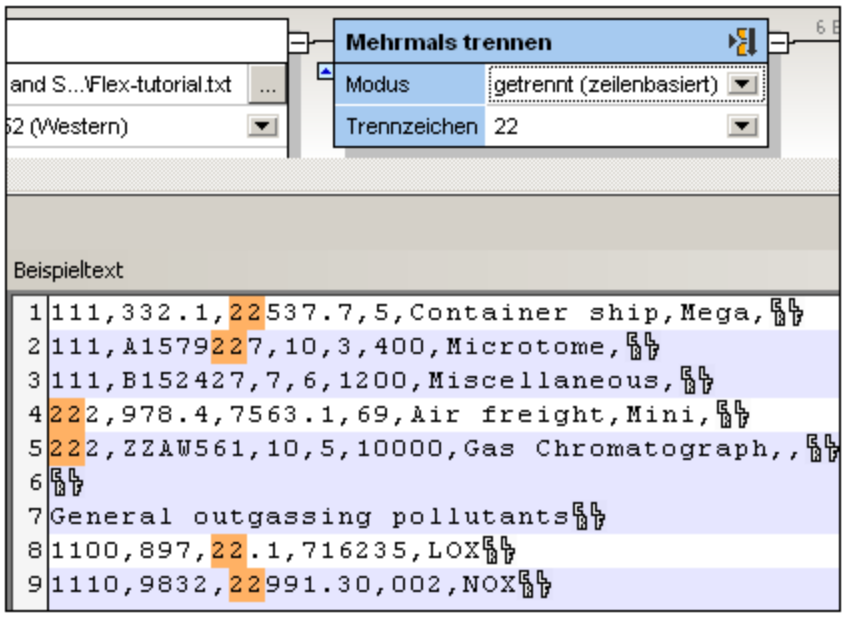


Das dritte Fragment enthält alle Zeichen/Zeilen ab der Stelle, an der "22" das zweite Mal vorkommt bis zum Ende.



© 2016-2022 Altova GmbH

Vergleichen Sie das obige Beispiel mit dem Ergebnis, das wir erhalten würden, wenn wir die Option **getrennt (zeilenbasiert)** und das Trennzeichen "**22"** verwenden würden.



In diesem Fall gäbe es sechs Fragmente bestehend aus Zeilen, die die Zahl 22 irgendwo in der Zeile enthalten.

4.6.5.2 Einmal trennen

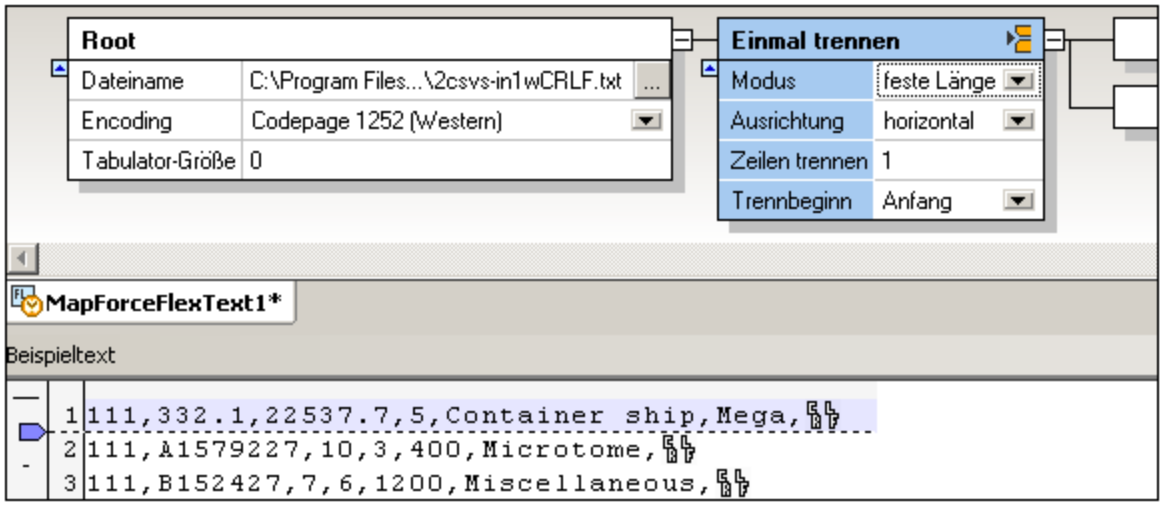


Mit dieser Option werden zwei vertikal übereinander ausgerichtete Container angelegt. Der obere Container enthält den Text, der durch die Bedingung definiert wird, die im **Einmal trennen**-Container eingestellt wurde. Der untere Container enthält den Rest der Textdatei/des Fragments. Es gibt mehrere Versionen der Option "Einmal trennen"; Feste 710 Länge, getrennt (beliebig) 711 und getrennt zeilenbasiert 712.

Wenn Sie diese Option zum ersten Mal auswählen, werden automatisch Standardparameter verwendet und die erzeugten Fragmente werden in beiden Containern angezeigt. Beachten Sie, dass der **Einmal trennen-**Container derzeit aktiv ist und im Beispieltextbereich eine Vorschau aller aktuellen Datensätze/Zeilen angezeigt wird.

Die **Standardeinstellungen** für den Container sind:

Modus	feste Länge
Ausrichtung	horizontal
Zeilen trennen	1
Trennbeginn	Anfang



Standardergebnis:
Die erste Zeile des Texts erscheint im oberen Container. Der untere Container enthält den Rest der Textdatei/des Fragments.

4.6.5.2.1 Modus - Feste Länge

Verwenden Sie den Modus **Einmal trennen (feste Länge),** um Text an einer bestimmten Zeile oder Spalte relativ zum Anfang oder Ende des Texts in zwei horizontale bzw. vertikale Fragmente zu teilen. Im Folgenden sind die Einstellungen zum Modus **Einmal trennen (feste Länge)** beschrieben.

Ausrichtung

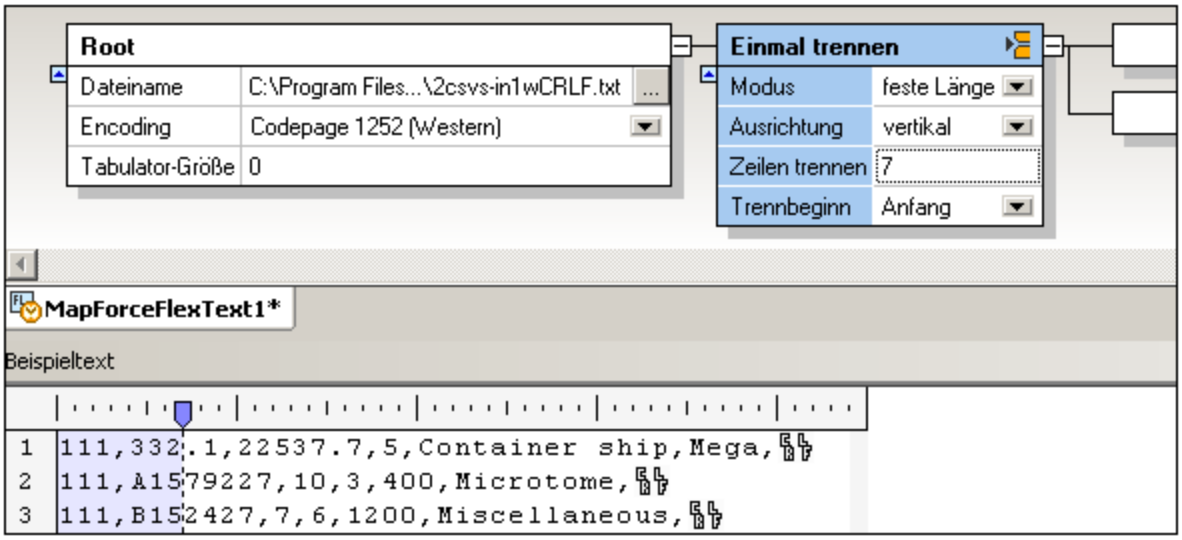
Damit können Sie definieren, ob das Textfragment horizontal oder vertikal getrennt werden soll. Wählen Sie "Horizontal", um das Fragment in zwei horizontale Abschnitte zu teilen. Wählen Sie "Vertikal", um das Fragment in zwei vertikale Abschnitte zu teilen. Die Standardausrichtung ist "horizontal".

Zeilen trennen

Definiert die Anzahl der Zeilen (oder Spalten) nach der das Fragment geteilt werden soll. Die Standardeinstellung ist "1". Um diese Einstellungen zu ändern, wählen Sie eine der folgenden Methoden:

- Geben Sie einen Wert in das Feld "**Zeilen trennen**" ein
- Ziehen Sie den Tabulator am vertikalen Lineal

© 2016-2022 Altova GmbH



Trennbeginn

Anfang	Sucht nach der ersten Instanz des Trennzeichens beginnend am Textanfang.
Ende	Sucht nach der ersten Instanz des Trennzeichens beginnend am Textende.

4.6.5.2.2 Modus - Getrennt (beliebig)

Verwenden Sie den Modus **Einmal trennen - getrennt (beliebig),** um Text mittels eines an einer beliebigen Stelle im Text befindlichen benutzerdefinierten Trennzeichens zu trennen. Dies eignet sich im Allgemeinen für Dateien, die kein CR- oder LF-Zeichen enthalten und die Sie an einer bestimmten Stelle in der Zeile in zwei Fragmente teilen möchten. Beachten Sie bitte Folgendes:

- Mit diesem Modus werden zwei Fragmente erzeugt, die durch die Trennzeichen definiert werden, die Sie in das Feld "Trennzeichen" eingegeben haben.
- Die Trennzeichen werden nicht in das Fragment inkludiert.
- Das erste Fragment ist definiert als der Text zwischen dem ersten Zeichen der Datei/des Fragments bis zum letzten Zeichen vor dem Trennzeichen.
- Das zweite Fragment ist definiert als das erste Zeichen nach dem "Trennzeichen" bis zum letzten Zeichen in der Datei/im Fragment.
- Wenn das Trennzeichen an der ersten/letzten Position der Datei/des Fragments vorkommt, so bleibt der erste der beiden erzeugten Container leer.

Im Folgenden sind die Einstellungen zum Modus **Einmal trennen - getrennt (beliebig)** beschrieben.

Regular Expression

Dies ist eine optionale Einstellung, mit der Text an den Stellen, an denen eine Übereinstimmung mit einer Regular Expression, die definiert wurde, gefunden wird, getrennt wird (siehe Trennen von Text mit Hilfe von Regular Expressions 731). Der Standardwert ist "nein".

Trennzeichen

Definiert welche(s) Zeichen als Trennzeichen verwendet werden soll(en). Der Standardwert ist "keines" (kein Trennzeichen).

Trennbeginn

Anfang	Sucht nach der ersten Instanz des Trennzeichens beginnend am Textanfang.
Ende	Sucht nach der ersten Instanz des Trennzeichens beginnend am Textende.

4.6.5.2.3 Modus - Getrennt (zeilenbasiert)

Verwenden Sie den Modus **Einmal trennen - getrennt (zeilenbasiert)**, um Text, in dem sich das Trennzeichen an einer beliebigen Stelle in einer der Zeilen befindet, in zwei Fragmente zu teilen. Die Zeile, die das Trennzeichen enthält, wird nicht aufgeteilt, sondern bleibt ganz. Dieser Modus eignet sich im Allgemeinen für Dateien, die Datensatztrennzeichen (CR/LF) enthalten, wenn Sie den Text in zwei separate Fragmente aufteilen möchten. Beachten Sie bitte Folgendes:

- Mit diesem Modus werden zwei Fragmente erzeugt, die durch die Trennzeichen, die Sie in das Feld "Trennzeichen" eingeben, definiert werden.
- Die Trennzeichen werden im Fragment inkludiert**.**
- Das erste Fragment ist definiert als der gesamte Text bis zur Zeile, die das Trennzeichen enthält.
- Das zweite Fragment ist definiert als der Text und die Zeile einschließlich des Trennzeichens bis zum Ende der Datei/des Fragments.
- Wenn das Trennzeichen in der ersten/letzten Zeile der Datei/des Fragments vorkommt, so bleibt der obere Container leer.

Im Folgenden sind die Einstellungen zum Modus **Einmal trennen - getrennt (zeilenbasiert)** beschrieben.

Regular Expression

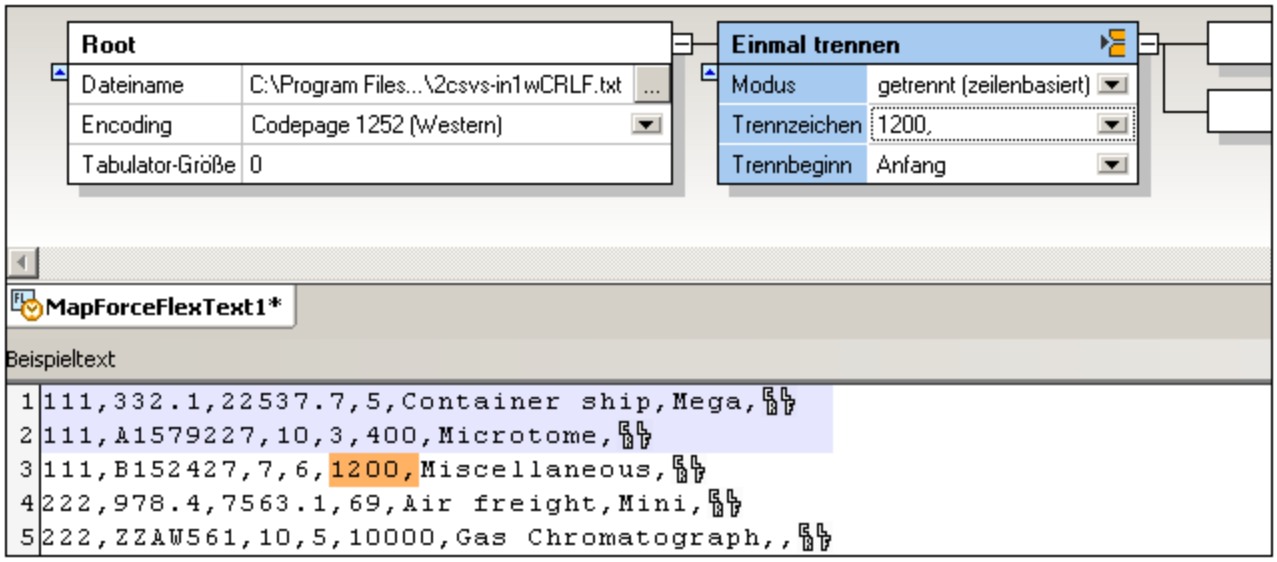
Dies ist eine optionale Einstellung, mit der Text an den Stellen, an denen eine Übereinstimmung mit einer Regular Expression, die definiert wurde, gefunden wird, getrennt wird (siehe Trennen von Text mit Hilfe von Regular Expressions 731). Der Standardwert ist "nein".

Trennzeichen

Definiert welche(s) Zeichen als Trennzeichen verwendet werden soll(en). Der Standardwert ist "keines" (kein Trennzeichen).

© 2016-2022 Altova GmbH

Wenn Sie z.B. das Trennzeichen "1200" für den unten gezeigten Text verwenden, so erhalten Sie zwei Fragmente.



Das erste Fragment enthält den Text bis zur Zeile, die das Trennzeichen enthält.

```
1 111,332.1,22537.7,5,Container ship,Mega,
2 111,A1579227,10,3,400,Microtome,
```

Das zweite Fragment enthält die gesamte Zeile, die das Trennzeichen enthält sowie alle restlichen Zeilen bis zum Ende.

```
1 111,B152427,7,6,1200,Miscellaneous,
2 222,978.4,7563.1,69,Air freight,Mini,
3 222,ZZAW561,10,5,10000,Gas Chromatograph,,
```

Trennbeginn

Anfang	Sucht nach der ersten Instanz des Trennzeichens beginnend am Textanfang.
Ende	Sucht nach der ersten Instanz des Trennzeichens beginnend am Textende.

4.6.5.2.4 Modus - Getrennt (Zeile beginnt mit)

Wählen Sie den Modus **Einmal trennen - Getrennt (Zeile beginnt mit)** aus, wenn die Trennung in der ersten Zeile, die mit den definierten Zeichen beginnt, erfolgen soll. Bei Auswahl dieses Modus werden die folgenden zwei Textfragmente erstellt:

- Das erste Fragment enthält den gesamten Text bis zur Zeile, in der sich das Trennzeichen befindet.
- Das zweite Fragment enthält den restlichen Text einschließlich der Zeile, in der sich das Trennzeichen befindet.

- Wenn sich das Trennzeichen am Anfang der ersten Zeile befindet, erfolgt keine Trennung (d.h. das erste erzeugte Fragment bleibt leer).

Regular Expression

Dies ist eine optionale Einstellung, mit der Text an den Stellen, an denen eine Übereinstimmung mit einer Regular Expression, die definiert wurde, gefunden wird, getrennt wird (siehe Trennen von Text mit Hilfe von Regular Expressions 731). Der Standardwert ist "nein".

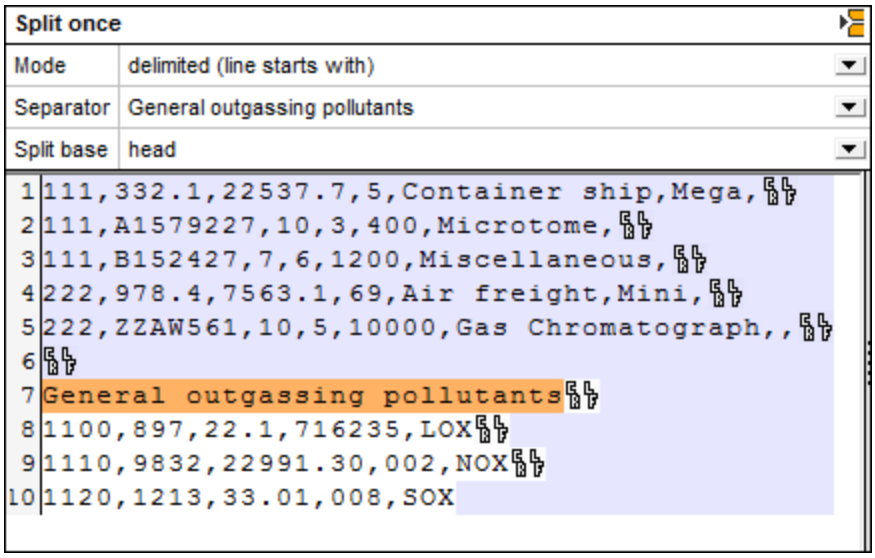
Trennzeichen

Definiert welche(s) Zeichen als Trennzeichen verwendet werden soll(en). Der Standardwert ist "keines" (kein Trennzeichen).

Trennbeginn

Anfang	Sucht nach der ersten Instanz des Trennzeichens beginnend am Textanfang.
Ende	Sucht nach der ersten Instanz des Trennzeichens beginnend am Textende.

In der folgenden Abbildung sehen Sie, wie eine Textdatei in der ersten Zeile, die mit den definierten Trennzeichen ("General outgassing pollutants") beginnt, getrennt wird. Wenn die Option "Trennbeginn" auf **Ende** gesetzt wäre, würde die Trennung in der ersten Zeile, die von Ende aus gesehen, mit dem definierten Zeichen beginnt, erfolgen. In diesem Beispiel ist dies dieselbe Zeile.



Das erste erzeugte Fragment ist das folgende:

 © 2016-2022 Altova GmbH

```
111,332.1,22537.7,5,Container ship,Mega,
111,A1579227,10,3,400,Microtome,
111,B152427,7,6,1200,Miscellaneous,
222,978.4,7563.1,69,Air freight,Mini,
222,ZZAW561,10,5,10000,Gas Chromatograph,,
```

Als zweites Fragment erhalten Sie dieses:

```
General outgassing pollutants
1100,897,22.1,716235,LOX
1110,9832,22991.30,002,NOX
1120,1213,33.01,008,SOX
```

4.6.5.3 Aufteilen

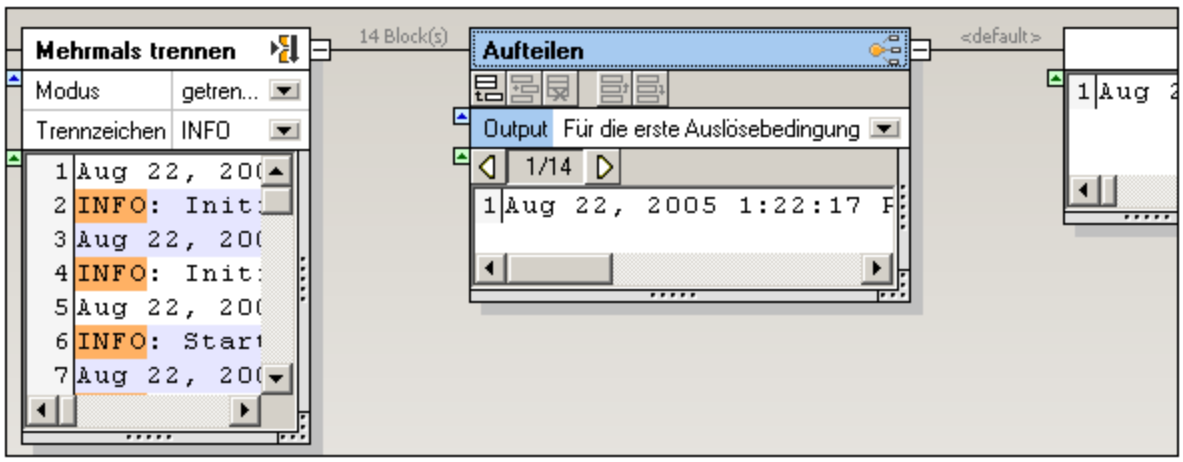


Mit Hilfe der Option "Aufteilen" können Sie für ein einzelnes Textfragment mehrere Schlüsselwörter oder Bedingungen definieren. Für jedes Schlüsselwort, das Sie definieren, gibt es einen eigenen Container, in dem die Daten nur dann abgelegt werden, wenn eine bestimmte Bedingung zutrifft, also "true" ist. Wenn keiner der Bedingungen zutrifft, so wird das jeweilige Fragment in einem "Standardcontainer" abgelegt.

Die **Standardeinstellungen** für den Container sind:

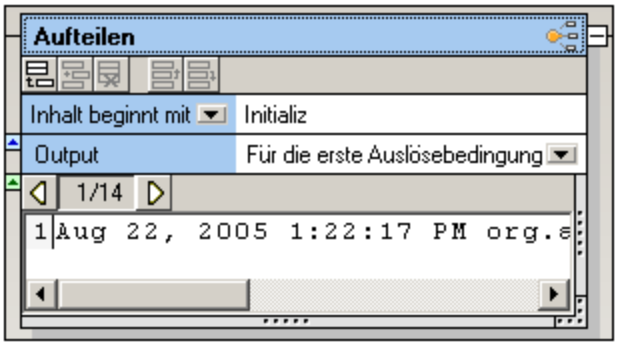
Output: für die erste Auslösebedingung.

Im Beispiel unten wird eine Tomcat-Log-Datei verarbeitet, in der die einzelnen Prozesse herausgetrennt werden sollen, damit sie gemappt werden können. Wenn Sie einen "Aufteilen"-Container zum ersten Mal definieren, erscheint rechts neben dem "Aufteilen"-Container nur der Standardcontainer. Alle Daten werden automatisch an diesen Container übergeben.



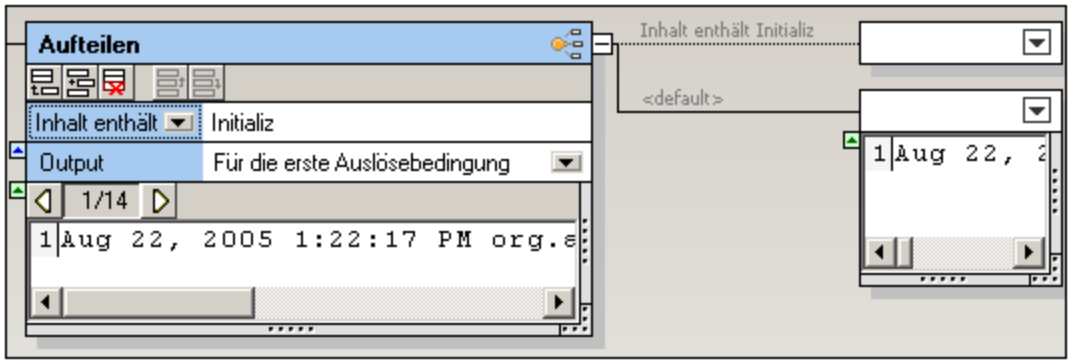
Im "Mehrmals trennen"-Container werden unter Verwendung der Option "getrennt (zeilenbasiert)" alle INFO-Abschnitte aus der Log-Datei herausgetrennt und an den "Aufteilen"-Container übergeben.

1. Klicken Sie auf das Symbol "Anhängen" , um eine neue Bedingung zum "Aufteilen"-Container hinzuzufügen.
2. Doppelklicken Sie auf das Feld rechts neben "**Inhalt beginnt mit**" geben Sie "**Initializ**" ein und drücken Sie die Eingabetaste.

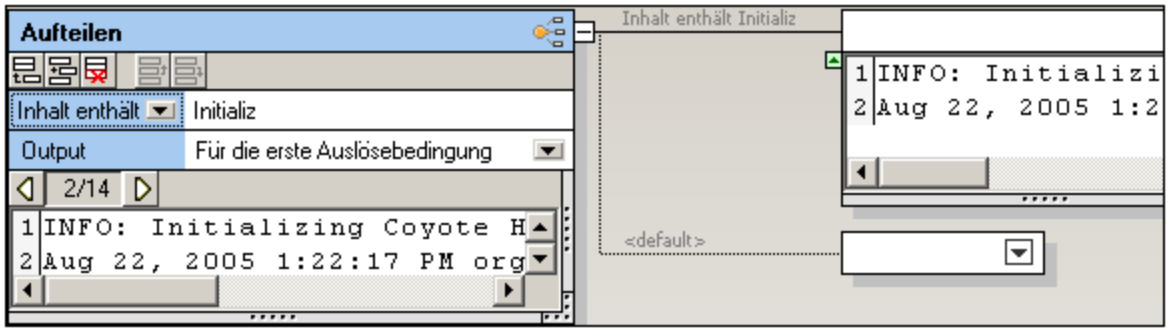


Es wird ein neuer Container hinzugefügt. Wenn die Bedingung zutrifft, also "true" ist, werden die Daten an diesen Container übergeben. Wenn nicht, werden sie an den Standard-Container übergeben.

3. Klicken Sie auf das Listenfeld "**Inhalt beginnt mit**" und ändern Sie den Eintrag in "Inhalt enthält". Die erste Bedingung wurde nun definiert und Sie sehen das Resultat unten.
Das erste Fragment enthält "Initializ" nicht, daher wird der Inhalt an den **Standard-**Container übergeben.



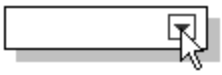
4. Klicken Sie auf das Symbol "Nächsten Block anzeigen" , um das nächste Textfragment zu sehen.



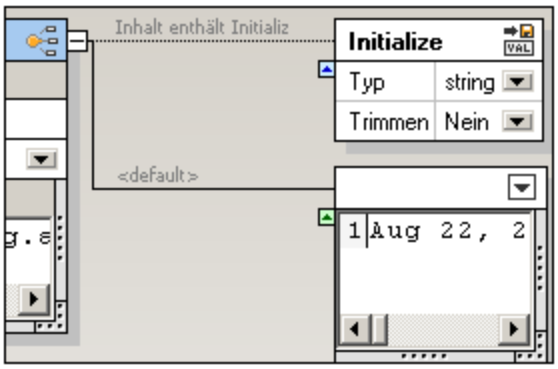
Das Fragment "Initializing..." erscheint nun im dazugehörigen Container und der Standardcontainer ist leer. Wenn Sie die Fragmente der Reihe nach anzeigen, sehen Sie, was die einzelnen Container enthalten werden.

5. Klicken Sie auf das Containersymbol und wählen Sie den Befehl **Als Wert speichern**.

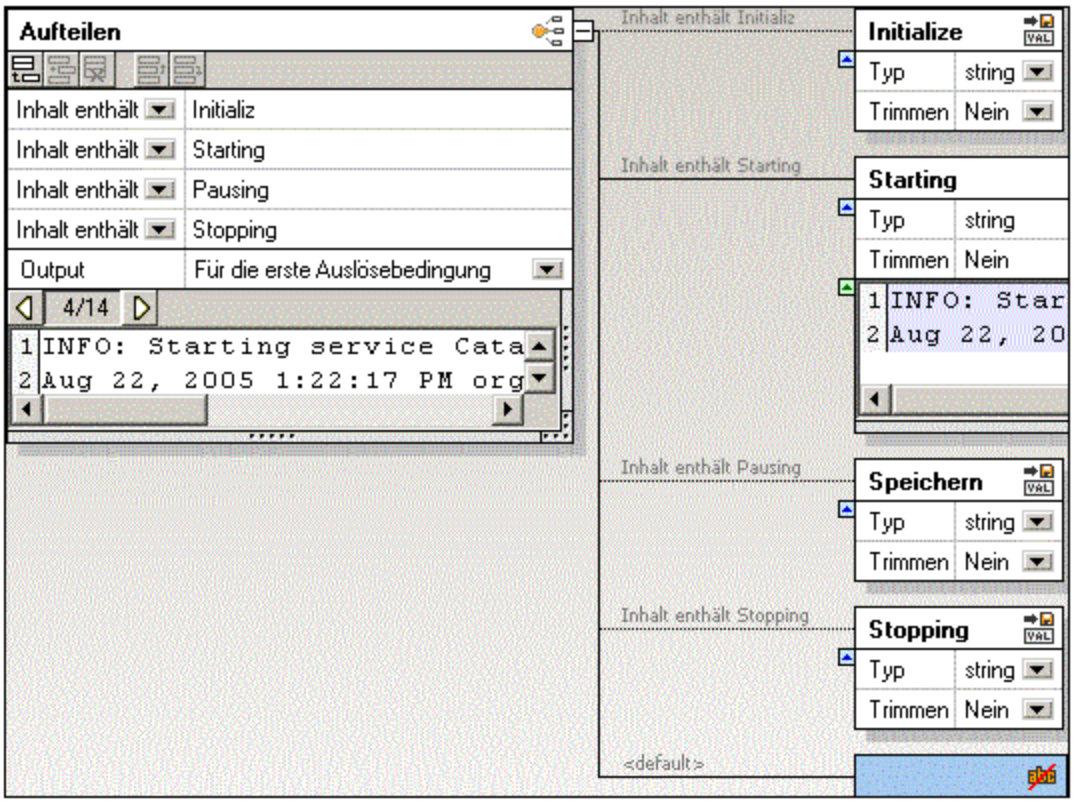
© 2016-2022 Altova GmbH



6. Doppelklicken Sie auf die Titelleiste "Speichern" und ändern Sie den Text z.B. Initialize.



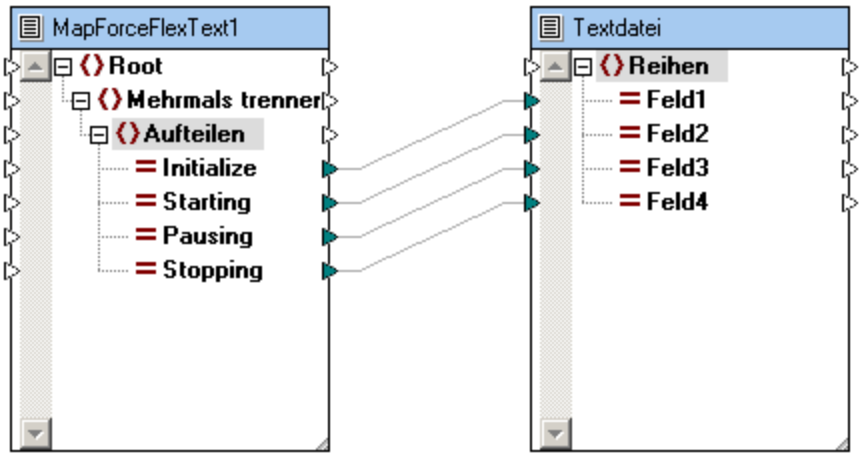
7. Klicken Sie auf das Symbol "Anhängen" , um eine neue Bedingung zum "Aufteilen"-Container hinzuzufügen.
8. Doppelklicken Sie in das Feld neben "**Inhalt beginnt mit**", geben Sie den Text "Starting" ein und drücken Sie die Eingabetaste.
   Sie können beliebig viele Bedingungen hinzufügen, z.B. Pausing und Stopping.
   Geben Sie jedem der dazugehörigen Container einen Namen, um sie in MapForce leichter unterscheiden zu können.



Im obigen Screenshot sehen Sie alle vier Bedingungen und den Inhalt des "Starting"-Containers bei Block/Fragment Nr. 6. Die entsprechenden Container wurden alle umbenannt, um sie in MapForce besser unterscheiden zu können.

Beachten Sie, dass die Bedingungen in der Liste der Bedingungen mit Hilfe der entsprechenden Schaltflächen (Nach oben/Nach unten bzw.) nach oben oder nach unten verschoben werden können.

9. Speichern Sie die Vorlage und fügen Sie sie in MapForce ein.



Anmerkung

Wenn ein Textfragment im aktuellen Fragment eine Bedingung erfüllt, so werden **alle Daten** dieses Fragments an den dazugehörigen Container übergeben. Die Daten werden nicht getrennt, sondern nur in den entsprechenden Container bzw. - wenn keine der definierten Bedingungen zutrifft - in den Standardcontainer geroutet.

Die mit "Aufteilen" erzeugten Container können für die weitere Verarbeitung verwendet werden. Sie können einen solchen Container in "Einmal trennen", "Mehrmals trennen" oder eine beliebige andere Option ändern.

Inhalt beginnt mit:
Die Daten werden nur dann an den dazugehörigen Container übergeben, wenn die in der Bedingung definierte Zeichenfolge am Anfang des Textfragments vorkommt.

Inhalt enthält:
Die Daten werden an den dazugehörigen Container übergeben, wenn die in der Bedingung definierte Zeichenfolge an einer beliebigen Stelle im Textfragment vorkommt.

Für die erste Auslösebedingung:
Die Daten werden übergeben, wenn **eine** der **Bedingungen** in der Bedingungsliste **true** ist. Alle anderen Bedingungen, die "true" sind, werden ignoriert und es werden keine Daten an einen der dazugehörigen Container übergeben.

Für alle Auslösebedingungen:
Es werden für **jede** Bedingung in der Bedingungsliste, die "true" ist, Daten übergeben. Auf diese Art wird es möglich, dass mehrere Instanzen derselben Daten/desselben Fragments in mehreren verknüpften Containern gleichzeitig vorkommen. Dies kann vorkommen, wenn ein Textfragment Text enthält, der gleichzeitig zwei Bedingungen erfüllt z.B. "initializing starting sequence" im obigen Beispiel.

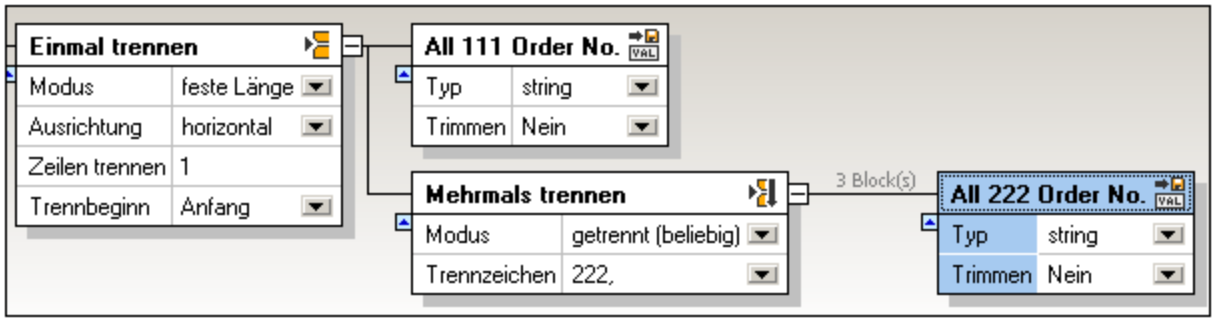
© 2016-2022 Altova GmbH

4.6.5.4 Node

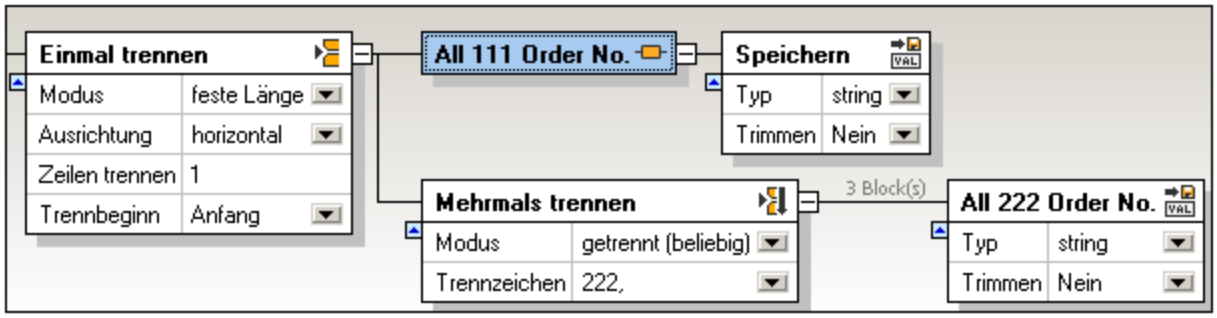


Damit können Sie eine neue hierarchische Ebene zur FlexText- und MapForce-Baumstruktur hinzufügen. Die Daten, die der folgende Node/Container enthält, werden so übergeben, wie sie sind.

Im unten gezeigten Screenshot ist der Container "All 111 Order No." der letzte Container im oberen Zweig.



Klicken Sie rechts oben im Container auf das Symbol dieses Containers und wählen Sie die **Node**-Option aus dem Kontextmenü aus.

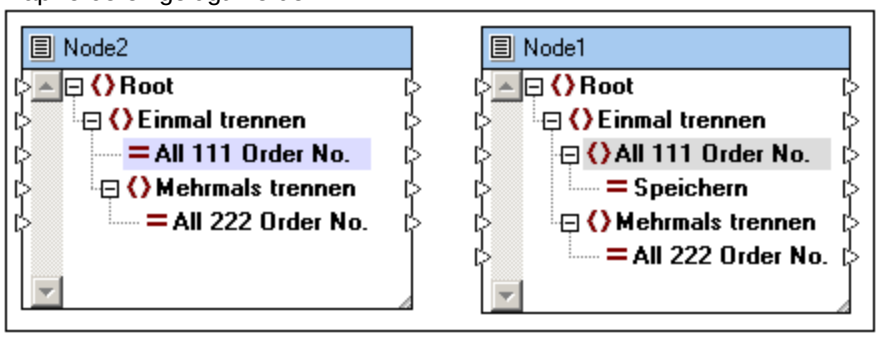


Es wurde nun ein neuer Container rechts vom aktuellen hinzugefügt.

Anmerkung

Der automatisch hinzugefügt Container wurde anschließend manuell als "**Als Wert speichern**" definiert.

Im unten gezeigten Screenshot sehen Sie beide Vorlagenstrukturen so, wie sie aussehen, wenn sie in MapForce eingefügt werden.



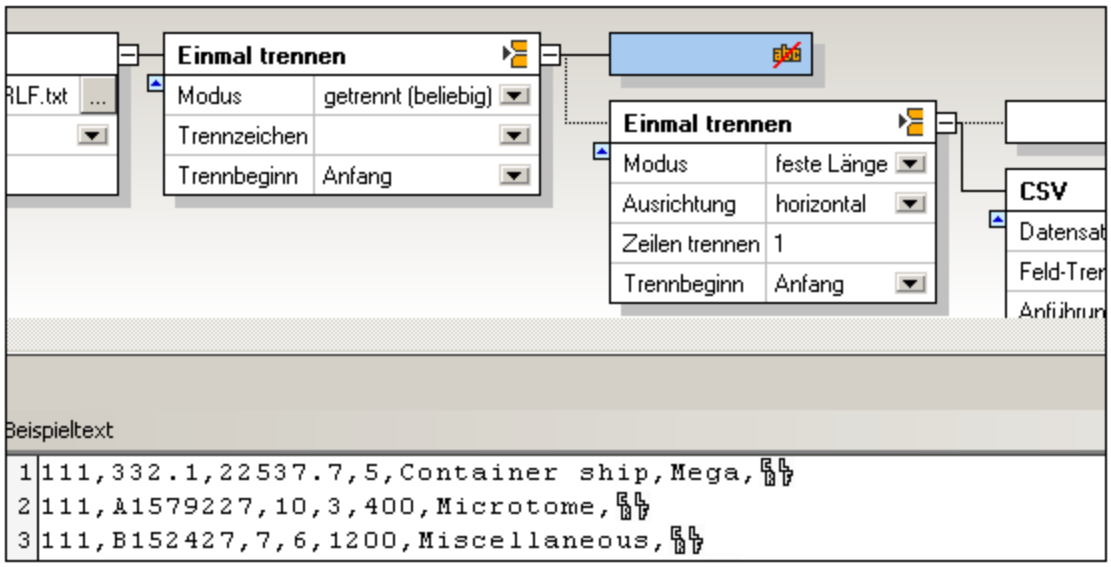
In der linken Komponente sehen Sie die anfängliche Struktur, bevor der neue Node hinzugefügt wurde.

In der rechten Komponente sehen, Sie wie sich die Komponentenstruktur geändert hat. "All 111..." ist nun ein Parent-Element. Unterhalb davon wurde ein neues Child-Element "Speichern" hinzugefügt.

4.6.5.5 Ignorieren

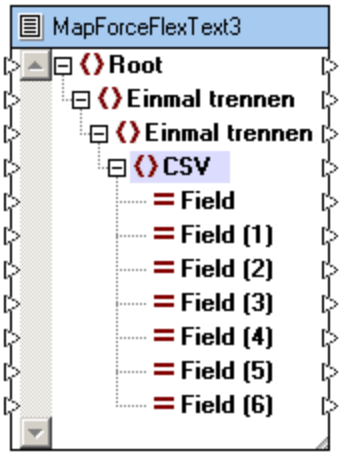


Mit dieser Option können Sie die Ausgabe eines bestimmten Textfragments unterdrücken. D.h. dieser Container und alle Daten, die er enthält, stehen in der FlexText-Komponente in MapForce nicht für das Mapping zur Verfügung.



Im obigen Beispiel wurde der aktive Container auf "Ignorieren" eingestellt. Daher steht der darin enthaltene Beispieltext in MapForce nicht für das Mapping zur Verfügung.

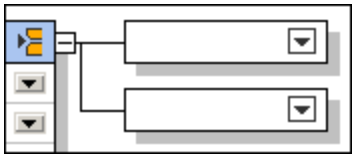
Wenn Sie die Textvorlage in MapForce einfügen, hat sie die unten gezeigte Struktur. Zwischen den beiden "Einmal trennen"-Elementen gibt es kein Element, das gemappt werden kann.



Anmerkung

© 2016-2022 Altova GmbH

Es gibt auch Standard "Ignorieren"-Container. Dies sind die neuen Container, die automatisch angehängt werden, wenn Sie "Einmal trennen" oder "Mehrmals trennen" usw. auswählen.



Der Inhalt dieser Container kann anfangs in MapForce nicht gemappt werden, wenn die Vorlage eingefügt wird. Sie müssen zuerst in FlexText eine der Container-Optionen auswählen: Als Wert speichern, Als CSV speichern usw., um die Container mappen zu können.

4.6.5.6 Als CSV speichern (getrennt)



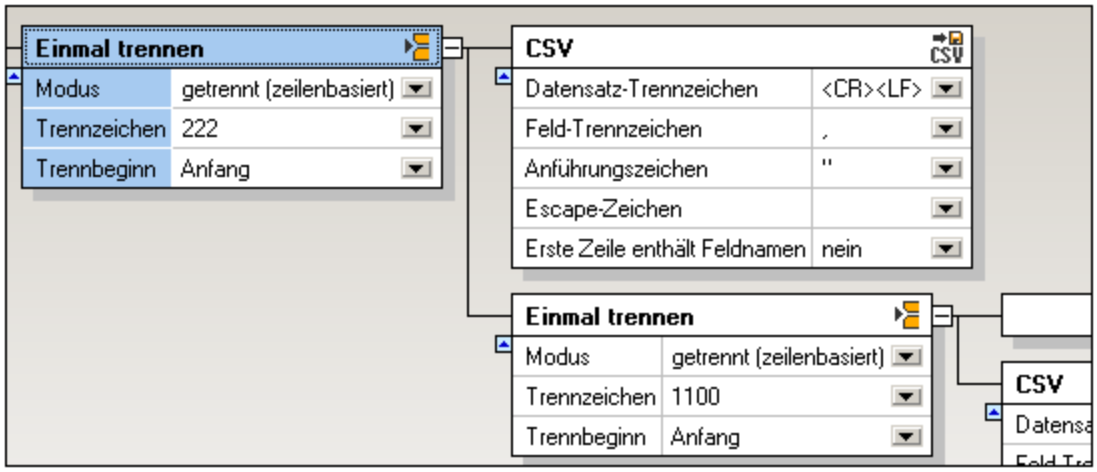
Mit dieser Option können Sie Textfragmente als CSV-Text speichern und einzelne Spalten in MapForce mappen. Sie können in FlexText beliebig viele CSV-Container/Komponenten erstellen, wobei jeder der CSV-Container andere Trennzeichen enthalten kann.

Der Beispieltextbereich bietet eine Übersicht über das aktuelle CSV-Fragment und gestattet Ihnen, einzelne Feldnamen und Feldtypen zu definieren. Jede Spalte wird in der FlexText-Komponente in MapForce als mapbares Element dargestellt.

Die **Standardeinstellungen** des Containers sind:

Datensatz-Trennzeichen	CR LF
Feld-Trennzeichen	,
Anführungszeichen	"
Anführungszeichen immer hinzufügen	nein
Escape-Zeichen	(keines)
Erste Zeile enthält Feldnamen	nein
Leere Felder als nicht vorhanden behandeln	ja

Im folgenden Beispiel sehen Sie, wie Daten in einer kleinen Textdatei in zwei CSV-Dateien aufgeteilt und in MapForce auf separate XML-Dateien gemappt werden.



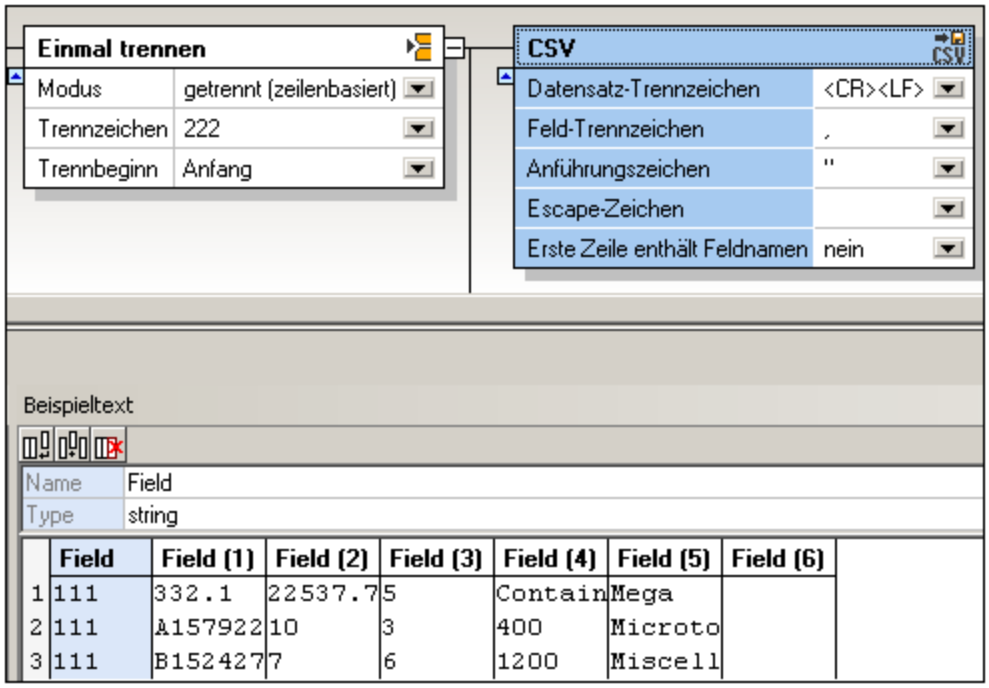
Der oben gezeigte **Einmal trennen**-Container wird verwendet, um zwei Container zu erzeugen. Dies wird mit der Option **getrennt (zeilenbasiert)** und unter Verwendung des Trennzeichens 222 erzielt. Alle Datensätze bis zur ersten Instanz von 222 werden an den CSV-Container übergeben. Der erste Container, der alle Datensätze mit der Zahl 111 enthält, wird anschließend als ein CSV-Container definiert. Im Beispieltextbereich sehen Sie den Inhalt des derzeit aktiven "Einmal trennen"-Containers.

Beispieltext

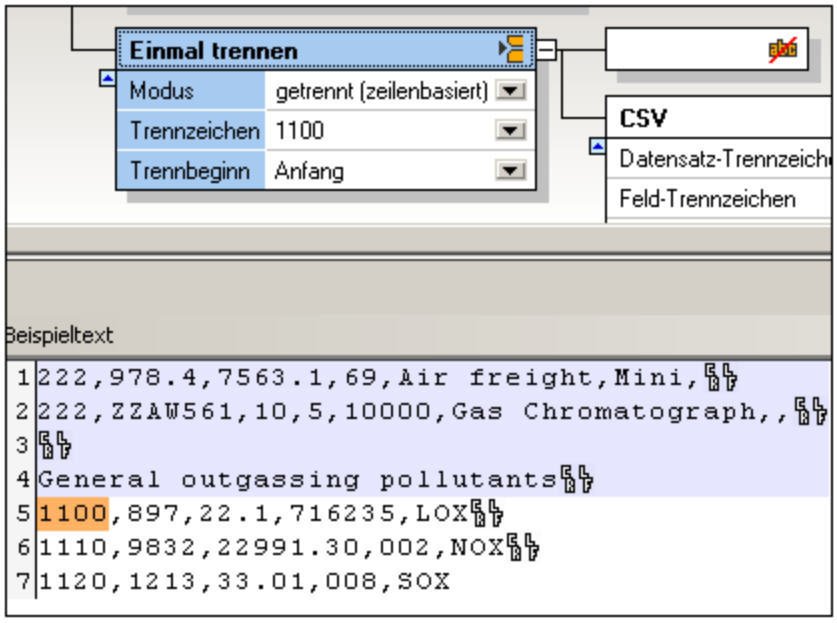
```
1  111,332.1,22537.7,5,Container ship,Mega,
2  111,A1579227,10,3,400,Microtome,
3  111,B152427,7,6,1200,Miscellaneous,
4  222,978.4,7563.1,69,Air freight,Mini,
5  222,ZZAW561,10,5,10000,Gas Chromatograph,,
6
7  General outgassing pollutants
8  1100,897,22.1,716235,LOX
9  1110,9832,22991.30,002,NOX
10 1120,1213,33.01,008,SOX
```

Die Standard-CSV-Einstellungen wurden nicht geändert. Wenn Sie auf den CSV-Container klicken, sehen Sie seinen Inhalt in Tabellenform.

© 2016-2022 Altova GmbH

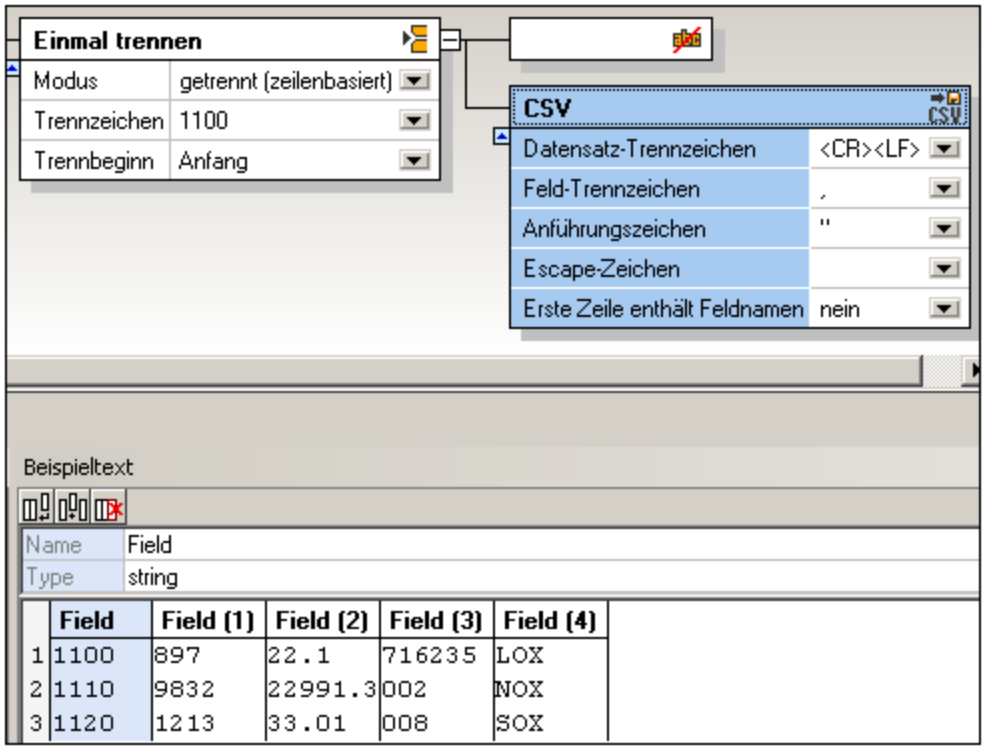


Der zweite Container enthält die restlichen Daten und wird als ein weiterer **Einmal trennen**-Container definiert. Dadurch werden zwei weitere Container erzeugt, von denen einer der zweite CSV-Container sein wird. Wenn Sie auf den "Einmal trennen"-Container klicken, sehen Sie den aktuellen Inhalt.

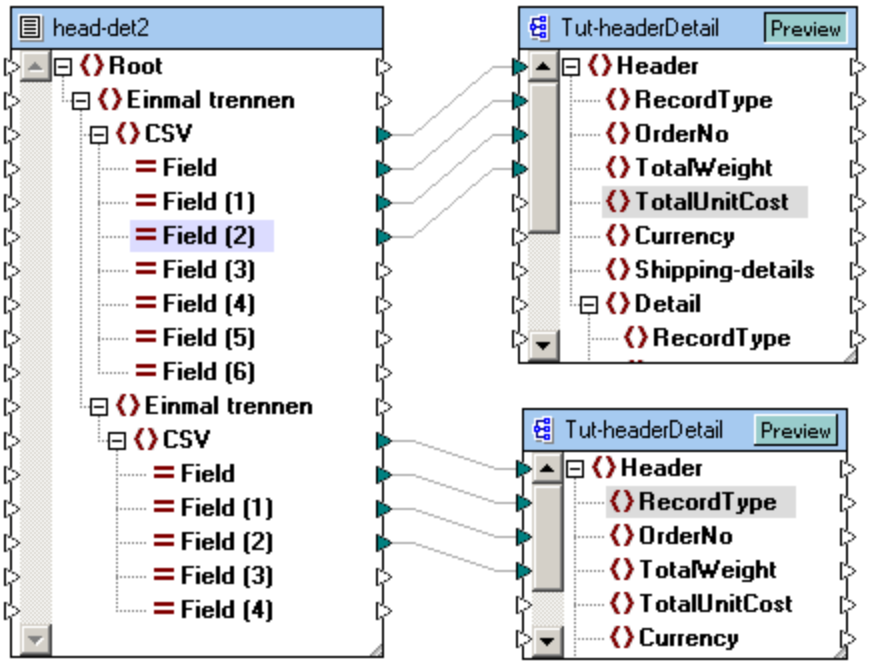


Mit Hilfe der Funktion "getrennt (zeilenbasiert)" und dem Trennzeichen "1100" werden die restlichen Daten in zwei Abschnitte aufgeteilt.

- Alle Datensätze bis zur ersten Instanz von 1100 werden an den ersten Container übergeben, der mit "Ignorieren" als nicht mapbar definiert wird.
- Anschließend wird der zweite Container als CSV definiert. Die Standardeinstellungen wurde nicht geändert. Wenn Sie auf den CSV-Container klicken, wird dessen Inhalt in Tabellenform angezeigt.



Wenn Sie die FlexText-Vorlage in MapForce einfügen, können Sie die Daten auf jede der unterstützten Zieldateien mappen. In diesem Beispiel wird jedes der CSV-Elemente auf zwei separate XML-Dateien gemappt.



Beachten Sie, dass nicht alle der Elemente in den CSV-Abschnitten auf die Zieldateien gemappt werden.

© 2016-2022 Altova GmbH

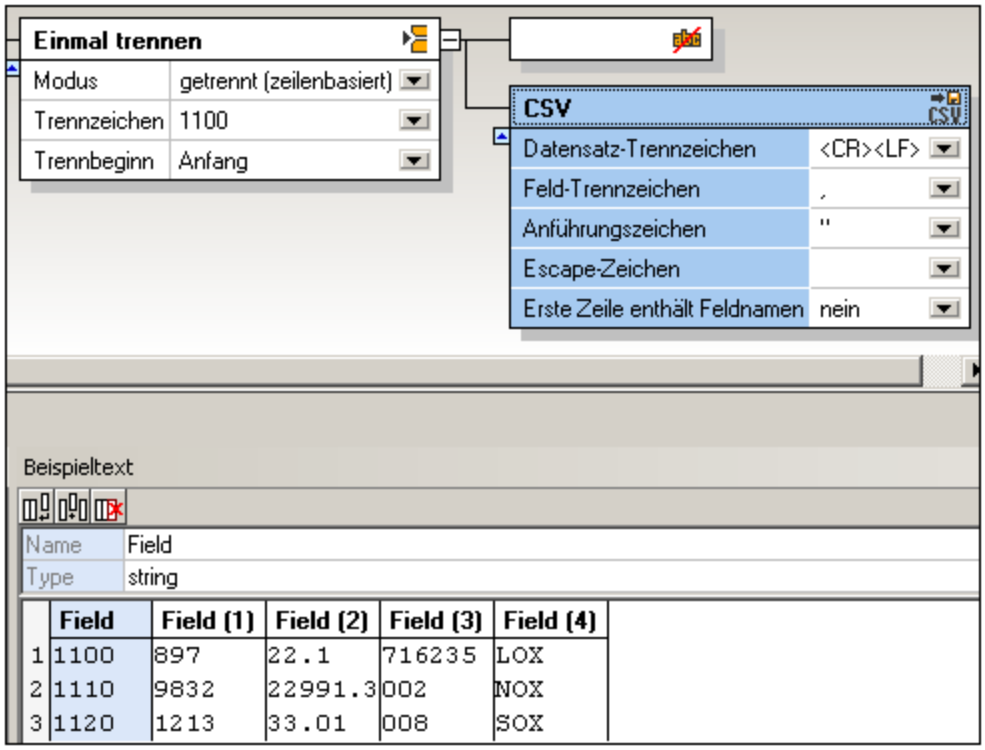
Die erste XML-Datei enthält alle 111 Datensatztypen.

```
<?xml version="1.0" encoding="UTF-8"?>
<Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <Header>
    <RecordType>111</RecordType>
    <OrderNo>332</OrderNo>
    <TotalWeight>22537.7</TotalWeight>
  </Header>
  <Header>
    <RecordType>111</RecordType>
    <OrderNo>0</OrderNo>
    <TotalWeight>10</TotalWeight>
  </Header>
  <Header>
    <RecordType>111</RecordType>
    <OrderNo>0</OrderNo>
    <TotalWeight>7</TotalWeight>
  </Header>
</Order>
```

Die zweite XML-Datei enthält alle Datensätze beginnend mit 1100.

```
<?xml version="1.0" encoding="UTF-8"?>
<Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <Header>
    <RecordType>1100</RecordType>
    <OrderNo>897</OrderNo>
    <TotalWeight>22.1</TotalWeight>
    <TotalUnitCost>716235</TotalUnitCost>
  </Header>
  <Header>
    <RecordType>1110</RecordType>
    <OrderNo>9832</OrderNo>
    <TotalWeight>22991.3</TotalWeight>
    <TotalUnitCost>2</TotalUnitCost>
  </Header>
  <Header>
    <RecordType>1120</RecordType>
    <OrderNo>1213</OrderNo>
    <TotalWeight>33.01</TotalWeight>
    <TotalUnitCost>8</TotalUnitCost>
  </Header>
</Order>
```

Konfigurieren der CSV Container/Daten



Wenn Sie im Beispieltextbereich auf ein Feld klicken, wird dieses markiert, sodass Sie es näher konfigurieren können.

- Klicken Sie in das Feld **Name,** um den hier erscheinenden Standardtext zu editieren.
- Klicken Sie in das Feld **Type,** um die Felddatentypen zu definieren: string, boolean, decimal usw.
- Klicken Sie auf das Symbol "Anhängen" , um ein neues Feld anzuhängen.
- Klicken Sie auf das Symbol "Einfügen" , um vor dem derzeit aktiven Feld ein Feld einzufügen.
- Klicken Sie auf das Symbol "Löschen" , um das derzeit aktive Feld zu löschen.

Anmerkung

Sie können die Ränder des Felds mit der Maus ziehen, um die Daten zu sehen.

Anführungszeichen immer hinzufügen

Damit können Sie festlegen, ob das definierte Anführungszeichen zu allen Feldern der generierten CSV-Datei hinzugefügt werden soll.

Leere Felder als nicht vorhanden behandeln

Damit können Sie definieren, dass für leere Felder in der Quelldatei keine entsprechenden leeren Datenelemente (Element oder Attribut) in der Zieldatei erzeugt werden.

Beachten Sie, dass dennoch Trennzeichen für die leeren Felder in der Quelldatei vorhanden sein müssen, z.B. "General outgassing pollutants**,,,,**".

© 2016-2022 Altova GmbH

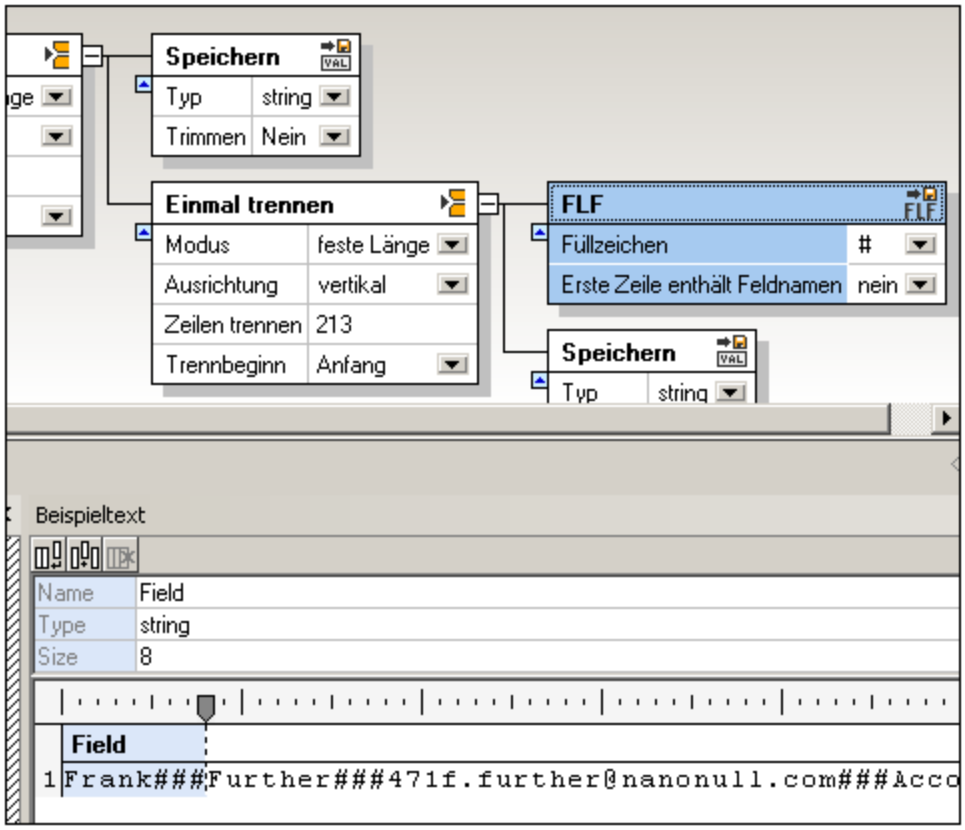
4.6.5.7 Als FLF speichern (feste Länge)

Mit dieser Option können Sie Textfragmente als Text mit fester Länge speichern und einzelne Spalten auf MapForce mappen. Sie können in FlexText beliebig viele FLF-Container/Komponenten anlegen, wobei jeder der FLF-Container andere Füllzeichen haben kann.

Der Beispieltextbereich bietet eine Übersicht über das aktuelle FLF-Fragment und gestattet Ihnen, einzelne Feldnamen, Feldlängen und Breiten zu definieren. Jede Spalte wird in der Textkomponente in MapForce als mapbares Element dargestellt.

Die **Standardeinstellungen** des Containers sind:

Füllzeichen	(keines)
Erste Zeile enthält Feldnamen	nein
Leere Felder als nicht vorhanden behandeln	ja

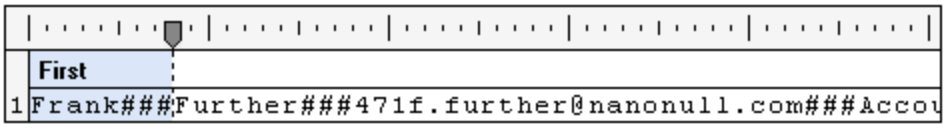


Konfigurieren des FLF-Containers/der Daten

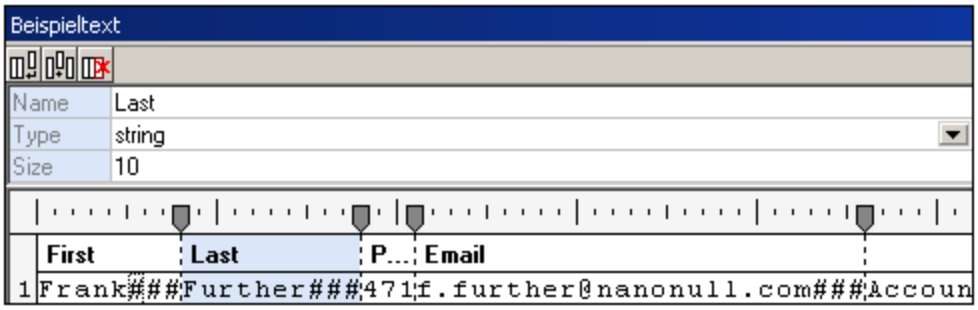
Nachdem Sie einen Container als "Speichern FLF" definiert haben, wird der Beispieltextbereich so wie im oben gezeigten Screenshot angezeigt. Es wird automatisch ein Standardfeld mit der Breite 10 eingefügt.

- Klicken Sie im Lineal auf das Tabulatorsymbol und ziehen Sie es, um es neu zu positionieren. Es erscheint ein Popup-Fenster mit der aktuellen Position.

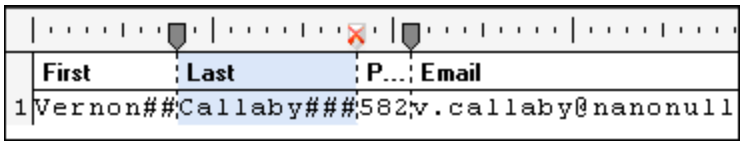
- Wenn Sie den Cursor über das Lineal platzieren, wird ein "Punkt" angezeigt. Wenn Sie darauf klicken, wird an dieser Position ein neues Tabulatorzeichen eingefügt.



- Nachdem Sie die neue Position definiert haben, klicken Sie auf das Feld, um es auszuwählen und bearbeiten Sie den Namen im Feld "Name".



- Um ein Feld zu **entfernen**, klicken Sie auf das Tabulatorsymbol und ziehen Sie es aus dem Lineal. Nach erfolgreicher Ausführung dieser Aktion ändert sich das Tabulatorsymbol.



Anmerkung:

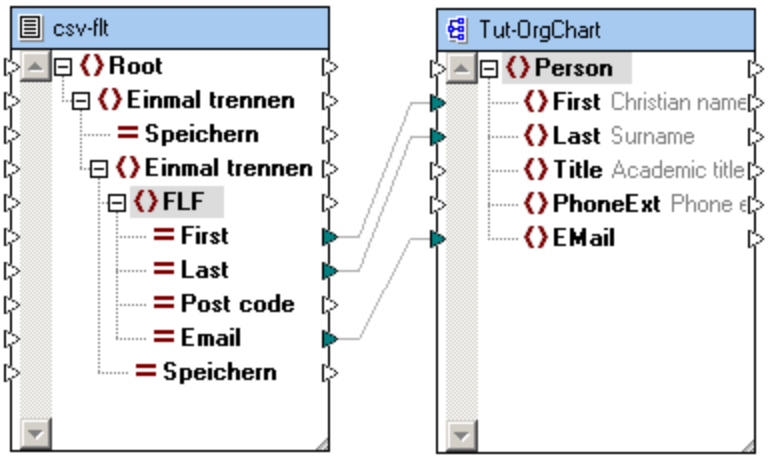
Wenn Sie einen Tabulator auf dem Lineal verschieben, werden automatisch alle Tabulatoren rechts davon verschoben. Um die anderen Tabulatorpositionen beizubehalten, halten Sie die **Umschalttaste** gedrückt, bevor Sie den Tabulator verschieben.

Wenn Sie im Beispieltext auf ein Feld klicken, erscheint dieses markiert, sodass Sie es näher konfigurieren können.

- Klicken Sie auf das Symbol "Anhängen" , um ein neues Feld mit der Länge 10 anzuhängen.
- Klicken Sie auf das Symbol "Einfügen" , um vor dem derzeit aktiven Feld ein Feld mit der Länge 10 einzufügen.
- Klicken Sie auf das Symbol "Löschen" , um das derzeit aktive Feld zu löschen.
- Klicken Sie in das Feld **Name,** um den hier erscheinenden Standardtext zu editieren.
- Klicken Sie in das Feld **Type,** um die Felddatentypen zu definieren: string, boolean, decimal usw.

Wenn Sie die FlexText-Vorlage in MapForce einfügen, können Sie die Daten auf jede der unterstützten Zieldateien mappen. In diesem Beispiel werden FLF-Elemente auf XML-Elemente gemappt.

© 2016-2022 Altova GmbH



Wenn die Option **Leere Felder als nicht vorhanden behandeln** auf **ja** gesetzt ist, werden für leere Felder in der Quelldatei keine entsprechenden leeren Datenelemente (Element oder Attribut) in der Zieldatei erzeugt werden. Ein Feld wird als nicht vorhanden betrachtet, wenn zwischen zwei aufeinanderfolgenden Füllzeichen keine Daten vorhanden sind.

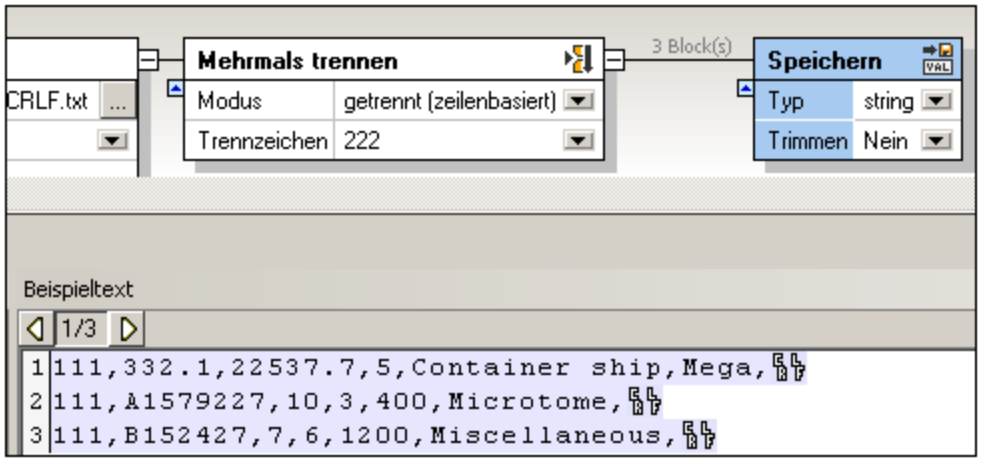
4.6.5.8 Als Wert speichern

Mit dieser Option können Sie einen Container definieren, der die darin enthaltenen Daten in MapForce als zu mappendes Element zur Verfügung stellt. Wenn Sie den Namen des Containers in FlexText nicht ändern, wird das Mapping-Element mit dem Namen "Speichern" angezeigt.

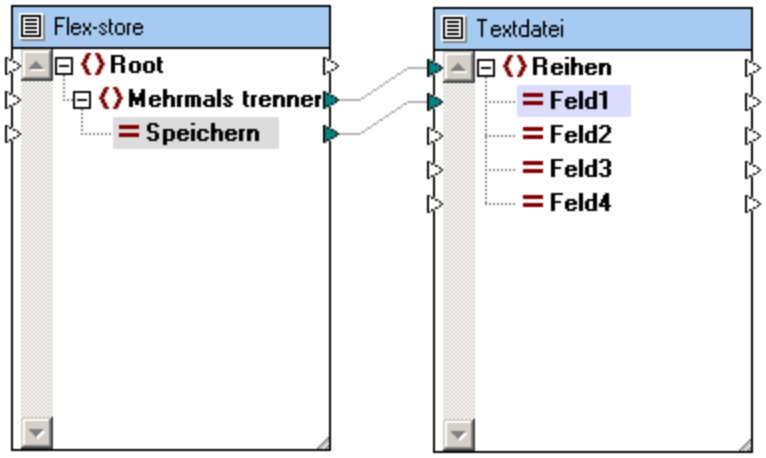
Die **Standardeinstellungen** des Containers sind:

Type	string
Trimmen	nein

Im unten gezeigten Screenshot sehen Sie den "Speichern"-Container sowie dessen Inhalt, der im Beispieltextbereich angezeigt wird.



Wenn Sie diese Vorlage speichern und in MapForce öffnen, können Sie das Element "Speichern" auf andere Elemente in einer Zielkomponente mappen.



Anmerkung

Das Element Feld1 in der Zieltextdatei wird 3 Fragmente enthalten, die aus dem Element "Speichern" stammen, wenn Sie auf das Register "Ausgabe-Vorschau" klicken, um eine Vorschau des Ergebnisses zu sehen.

Type

Gestattet Ihnen, den Datentyp der Textfragmente zu definieren.

Trimmen

Definiert die Seite, von der die Zeichen getrimmt, also gekürzt werden sollen: links, rechts, oder beide. Bei Auswahl von links, rechts oder beide wird die Option "Zeichenfolge" aktiviert.

Zeichenfolge

Definieren Sie die Zeichen, die Sie von diesem Textfragment trimmen möchten. Sie können beliebig viele Zeichen eingeben, indem Sie in das Feld doppelklicken. Die Zeichen, die Sie hier eingeben, werden von der Trimm-Seite/den Trimm-Seiten des Fragments entfernt.

4.6.6 FlexText und Regular Expressions

Sie können Regular Expressions in MapForce FlexText folgendermaßen verwenden:

1. Zum Aufteilen von Containern, wo eine Übereinstimmung gefunden wird (der übereinstimmende Text dient als Trennzeichen). In diesem Fall sind Regular Expressions implizit markiert, daher werden hier das Caret- (^) und das Dollar-Zeichen ($) nicht verwendet. Um z.B. drei aufeinander folgende Ziffern zu finden, verwenden Sie `[0-9]{3}` anstelle von `^[0-9]{3}$`.
2. Zum Umleiten von Text aus einem Aufteilen-Container, wenn der Text eine Übereinstimmung mit einer Regular Expression enthält.

Sie können in jeder der folgenden MapForce-Zielsprachen Regular Expressions in FlexText-Komponenten verwenden:

- Built-in (bei der Mapping-Vorschau)
- Built-in (beim Ausführen der MapForce Server-Ausführungsdatei)

© 2016-2022 Altova GmbH

- Codegenerierungssprachen (C++, C#, Java). Beachten Sie, dass einige komplexere Funktionen von Regular Expressions in diesen Sprachen von der Implementierung der Regular Expression in dieser spezifischen Sprache abhängen.

Die Syntax und Semantik von Regular Expressions in FlexText basiert, ähnlich der MapForce core-Funktion **`tokenize-regexp`**, auf http://www.w3.org/TR/xmlschema-2/#regexs. Beachten Sie Folgendes:

- Wenn die Trennbedingung mit zwei aufeinander folgenden Sequenzen übereinstimmt, erstellt FlexText dazwischen ein leeres Ergebnis (dasselbe Verhalten tritt auf, wenn Sie keine Regular Expressions verwenden).
- Regular Expression Flags (http://www.w3.org/TR/xquery-operators/#flags) werden nicht unterstützt.

Dieser Abschnitt enthält die folgenden Kapitel:

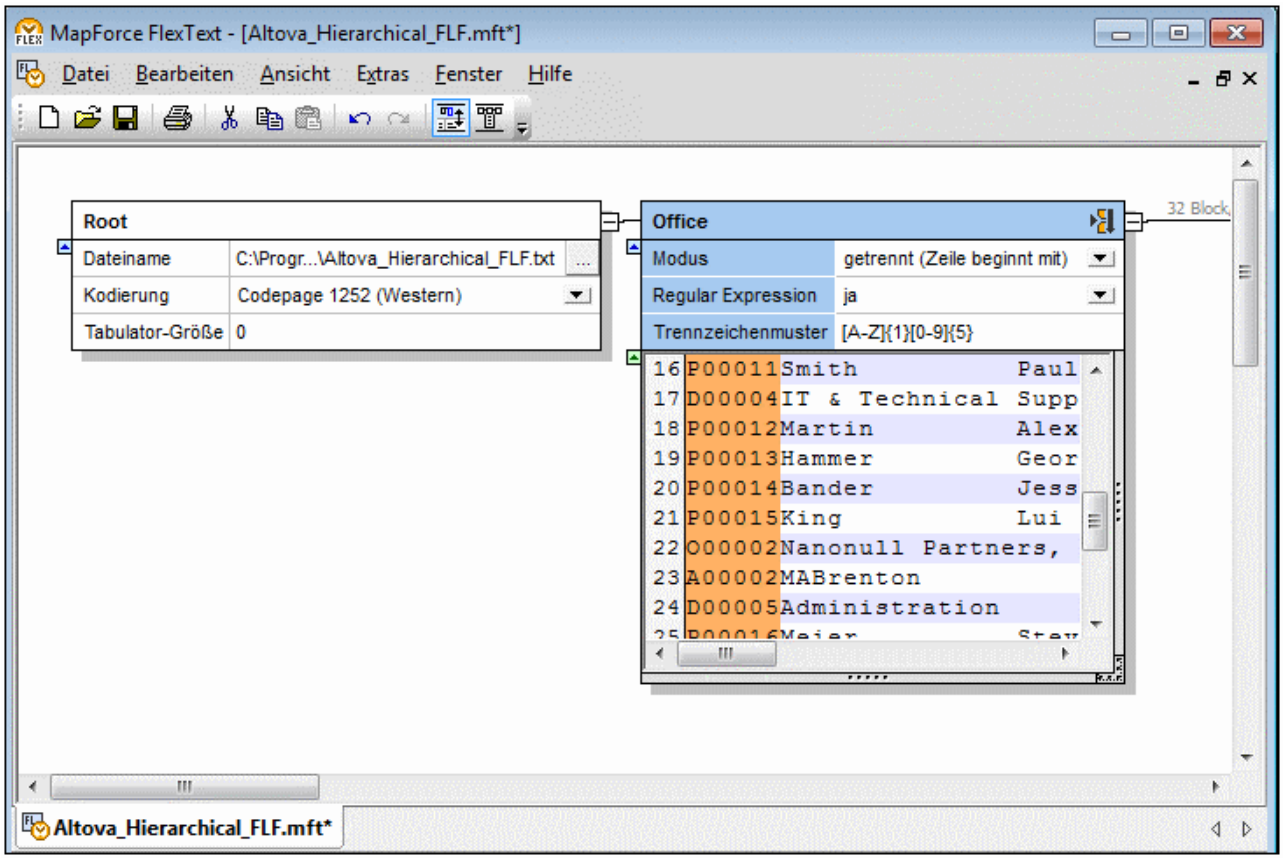
- Trennen von Text mit Hilfe von Regular Expressions 731
- Verwendung von Regular Expressions in Aufteilen-Bedingungen 733

4.6.6.1 Trennen von Text mit Hilfe von Regular Expressions

Wenn Sie Text in zwei oder mehr Fragmente trennen müssen, können Sie optional eine Regular Expression als Trennzeichen verwenden. Dies ist eine weitergehende Alternative zum Trennen von Text mit Hilfe eines einzigen oder mehrerer aufeinander folgender Zeichen.

Die Option zum Trennen von Text mit Hilfe von Regular Expressions steht unter den folgenden Bedingungen in FlexText zur Verfügung:

- Der Container ist vom Typ **Einmal trennen** 709 oder **Mehrmals trennen** 702.
- Als *Modus* ist **getrennt (beliebig)**, **getrennt (zeilenbasiert)** oder **getrennt (Zeile beginnt mit)** ausgewählt.



FlexText-Beispielvorlage, in der zum Trennen von Text Regular Expressions verwendet werden

Für Regular Expressions stehen die folgenden Optionen zur Verfügung.

Regular Expression	Um eine Regular Expression als Texttrennzeichen verwenden zu können, setzen Sie diese Option auf **ja**. Um einfachen Text als Trennzeichen zu verwenden, setzen Sie diese Option auf **nein**. Der Standardwert ist **nein**.
Trennzeichenmuster	Wenn die Option *Regular Expression* auf **ja** gesetzt wurde, steht ein Textfeld *Trennzeichenmuster* zur Verfügung, in das Sie die Regular Expression, die als Trennzeichen dienen soll, eingeben können. In der oben gezeigten FlexText-Vorlage findet die Regular Expression `[A-Z]{1}[0-9]{5}` genau ein alphanumerisches Zeichen, gefolgt von genau fünf numerischen Zeichen (z.B. "P00011"). Alle diese Instanzen werden in der Vorschau markiert und dienen als Text-Trennzeichen.
Trennzeichen für Schreiben	Die Option *Trennzeichen für Schreiben* kann dann verwendet werden, wenn die folgenden Bedingungen zutreffen: • Die FlexText-Komponente ist eine Zielkomponente • Als *Modus* ist **getrennt (beliebig)** gewählt.

© 2016-2022 Altova GmbH

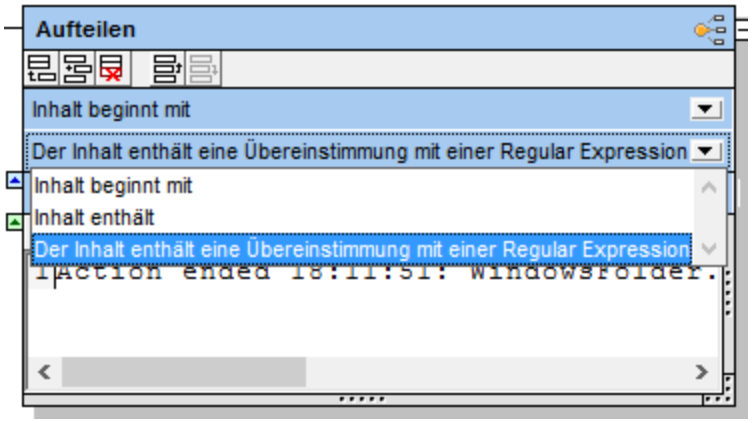
	Geben Sie in dieses Feld den String ein, der an der Stelle, an der eine Übereinstimmung mit der Regular Expression gefunden wurde, in die Zielkomponente geschrieben werden soll.

4.6.6.2 Verwendung von Regular Expressions in Aufteilen-Bedingungen

Wenn Sie mit Aufteilen-Containern arbeiten (siehe auch Aufteilen[715]), können Sie innerhalb des Aufteilen-Containers optional auch eine Bedingung erstellen, um das Textfragment weiterzuleiten, wenn es eine Übereinstimmung mit einer Regular Expression enthält. In diesen Fällen können Sie den Anfang oder das Ende des zu suchenden Texts mit dem Caret- (^) und dem Dollar-Zeichen ($) markieren, es sei denn C++ ist als Zieltransformationssprache eingestellt. (In C++ werden das Caret- und das Dollar-Zeichen als Zeilenbeginn bzw. -ende und nicht als der Beginn bzw. das Ende des gesamten Texts interpretiert).

So verwenden Sie eine Regular Expression in einer Aufteilen-Bedingung:

1. Definieren Sie den Container-Typ als Aufteilen[715] (klicken Sie auf die rechte obere Ecke des Containers und wählen Sie anschließend **Aufteilen** aus).
2. Klicken Sie auf die Schaltfläche **Bedingung anhängen** ( ), um eine neue Bedingung anzuhängen.
3. Definieren Sie als Bedingungstyp **Der Inhalt enthält eine Übereinstimmung mit einer Regular Expression** und geben Sie die Regular Expression in das Textfeld daneben ein.



Beispiel

Angenommen, Sie müssen Daten aus der unten dargestellten Datenbank-Log-Datei mappen (Sie finden diese Datei unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2023\MapForceExamples\SampleDatabaseLog.txt).

```
Action 18:11:51: INSTALL.
Action start 18:11:51: INSTALL.
Action 18:11:51: WindowsFolder.68B7C6D9_1DF2_54C1_FF1F_C8B3B9A1E18E.
Action start 18:11:51: WindowsFolder.68B7C6D9_1DF2_54C1_FF1F_C8B3B9A1E18E.
Action ended 18:11:51: WindowsFolder.68B7C6D9_1DF2_54C1_FF1F_C8B3B9A1E18E. Return value
1.
```

```
Action 18:11:51: SystemFolder.68B7C6D9_1DF2_54C1_FF1F_C8B3B9A1E18E.
Action start 18:11:51: SystemFolder.68B7C6D9_1DF2_54C1_FF1F_C8B3B9A1E18E.
Action ended 18:11:51: SystemFolder.68B7C6D9_1DF2_54C1_FF1F_C8B3B9A1E18E. Return value 0.
Action 18:11:51: WindowsFolder.9BAE13A2_E7AF_D6C3_FF1F_C8B3B9A1E18E.
Action start 18:11:51: WindowsFolder.9BAE13A2_E7AF_D6C3_FF1F_C8B3B9A1E18E.
Action ended 18:11:51: WindowsFolder.9BAE13A2_E7AF_D6C3_FF1F_C8B3B9A1E18E. Return value
0.
Action 18:11:51: SystemFolder.9BAE13A2_E7AF_D6C3_FF1F_C8B3B9A1E18E.
Action start 18:11:51: SystemFolder.9BAE13A2_E7AF_D6C3_FF1F_C8B3B9A1E18E.
Action ended 18:11:51: SystemFolder.9BAE13A2_E7AF_D6C3_FF1F_C8B3B9A1E18E. Return value 1.
Action 18:11:51: WindowsFolder.63E949F6_03BC_5C40_FF1F_C8B3B9A1E18E.
Action start 18:11:51: WindowsFolder.63E949F6_03BC_5C40_FF1F_C8B3B9A1E18E.
Action ended 18:11:51: WindowsFolder.63E949F6_03BC_5C40_FF1F_C8B3B9A1E18E. Return value
1.
```

Sie möchten Folgendes erreichen:

1. Sammlung aller Zeilen, die einen Rückgabewert enthalten, in einer Liste, d.h. die Liste muss alle Zeilen enthalten, die den Wert "Return value 1" oder "Return value 0" oder irgendeinen anderen in Form einer Ziffer ausgedrückten Rückgabewert enthalten.
2. Sammlung aller Zeilen, deren Text den Wert "Action start" enthält, in einer anderen Liste.

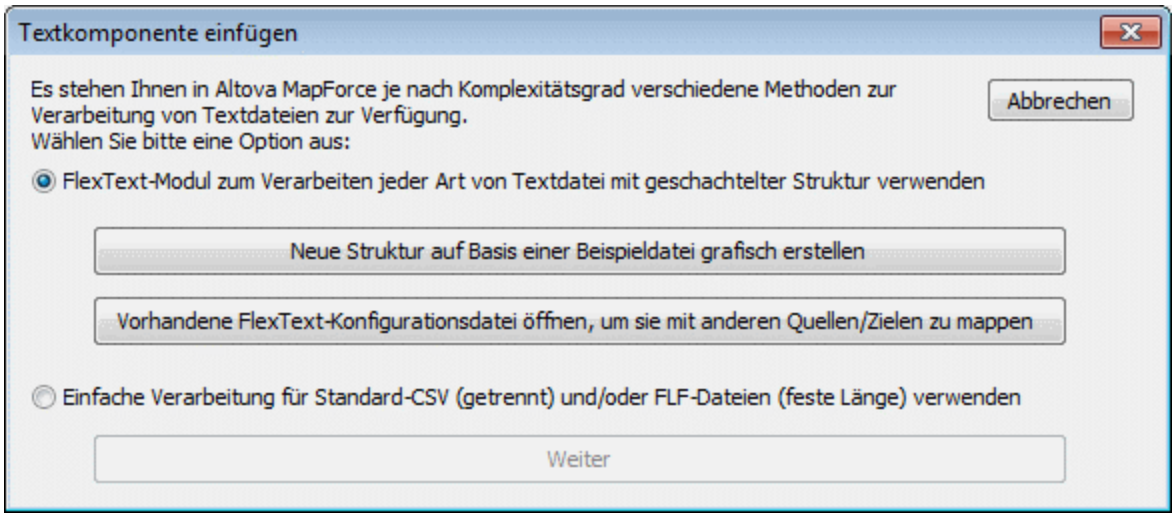
Zu diesem Zweck können Sie den Text mit Hilfe eines *Mehrmals trennen*-Containers in einzelne Zeilen aufgliedern. Anschließend können Sie die einzelnen Zeilen wie erforderlich mit Hilfe eines *Aufteilen*-Containers weiterleiten. Der *Aufteilen*-Container besteht aus den folgenden drei Bedingungen:

1. Weiterleiten der aktuellen Zeile an Output A, falls die Zeile den Wert "Action started" enthält. Sie können solche Zeilen mit Hilfe einer Bedingung vom Typ "Inhalt beginnt mit" und durch Eingabe des Werts "Action started" suchen.
2. Weiterleiten der aktuellen Zeilen an Output B, falls die Zeile einen Rückgabewert enthält. Sie können solchen Zeilen mit Hilfe der Regular Expression `Return value [0-9]\.` suchen. Diese Regular Expression findet eine Übereinstimmung, wenn die Zeile den Text "Return value" gefolgt von einer einzigen Ziffer, gefolgt von einem Punkt enthält. Der umgekehrte Schrägstrich (\ ) vor dem Punkt dient als Escape-Zeichen. Es gibt an, dass der Punkt als normales Zeichen und nicht als Metazeichen zu behandeln ist.
3. Weiterleiten der aktuellen Zeile an Output C (<default>), wenn keine der oben definierten Bedingungen auf die Zeile zutrifft.

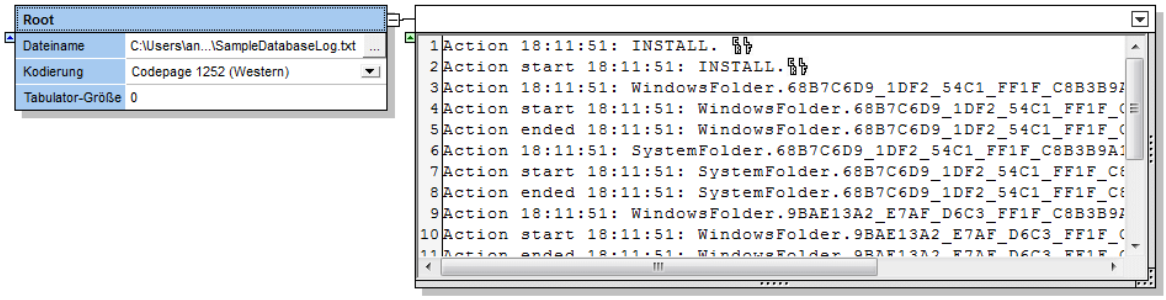
So erstellen Sie die FlexText-Vorlage, die die obigen Aufgaben durchführt:

1. Klicken Sie im Menü **Einfügen** auf **Textdatei**.

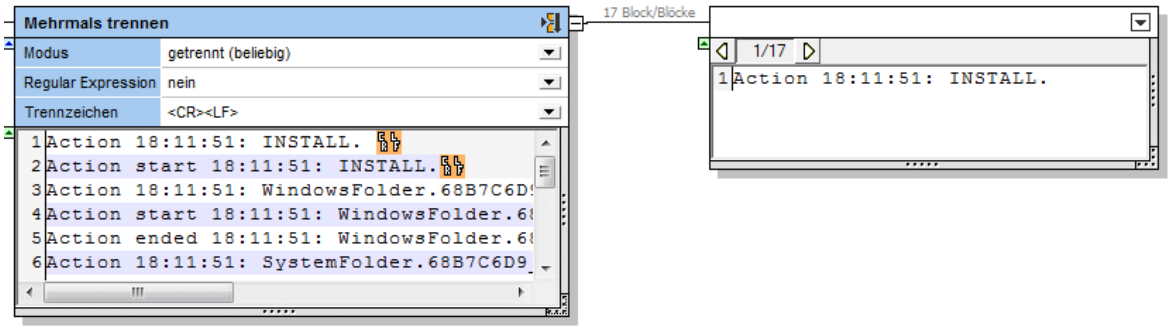
© 2016-2022 Altova GmbH



2. Klicken Sie auf **Neue Struktur auf Basis einer Beispieldatei grafisch erstellen** und speichern Sie die FlexText .mft-Vorlage in einem Verzeichnis Ihrer Wahl.
3. Wenn Sie aufgefordert werden, eine Textdatei zu öffnen, navigieren Sie zur Datei **<Dokumente>\Altova\MapForce2023\MapForceExamples\SampleDatabaseLog.txt**.

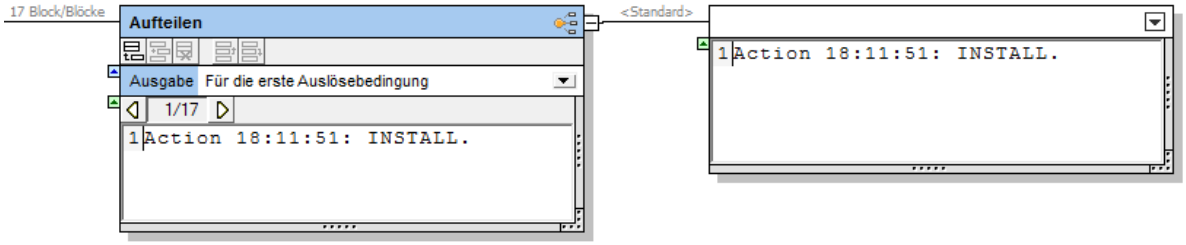


4. Klicken Sie auf die rechte obere Ecke des Ausgabe-Containers und wählen Sie die Option **Mehrmals trennen**. Da wir Carriage return (Wagenrücklauf) als Trennzeichen verwenden, wählen Sie die Modus **getrennt (beliebig)** und als Trennzeichen **<CR><LF>**. Daraufhin wird ein neuer Ausgabe-Container erstellt, der aus 17 Blöcken (einem für jede Zeile) besteht.

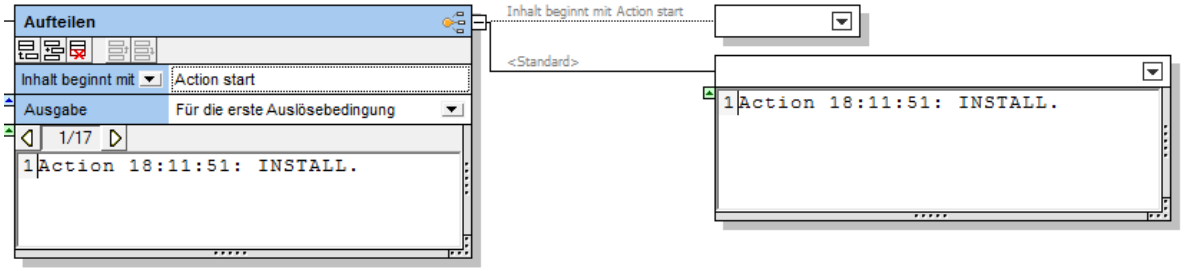


5. Klicken Sie auf die rechte obere Ecke des neuen Ausgabe-Containers und wählen Sie die Option **Aufteilen**. FlexText behandelt nun den Inhalt des Containers als Aufteilungsoption. Wie Sie unten sehen, wurde eine <Standard> Aufteilbedingung automatisch erstellt. Diese Bedingung leitet jeden

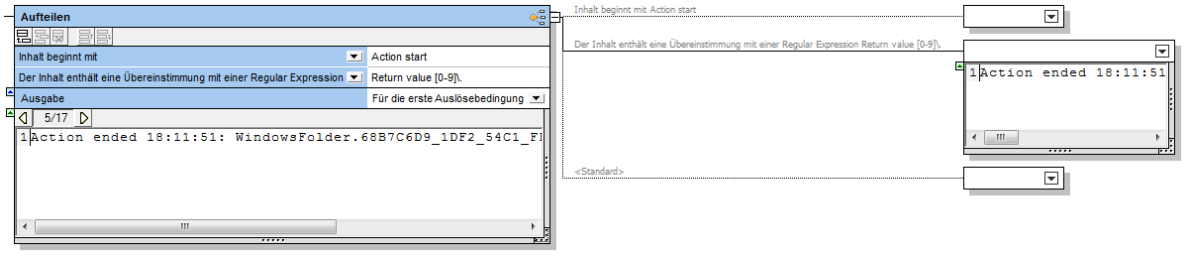
Text, der mit keiner anderen Bedingung übereinstimmt, an einen neuen Container weiter. Zu diesem Zeitpunkt sind keine anderen Bedingungen definiert, daher wird derzeit der gesamte Text an die Standardausgabe **<Standard>** weitergeleitet.



6. Klicken Sie auf die Schaltfläche **Bedingung anhängen** ( ) und fügen Sie eine Bedingung vom Typ **Inhalt beginnt mit** mit dem Wert "Action start" hinzu (siehe unten). Diese Bedingung leitet Text, der mit "Action start" beginnt, an einen neuen Container weiter.



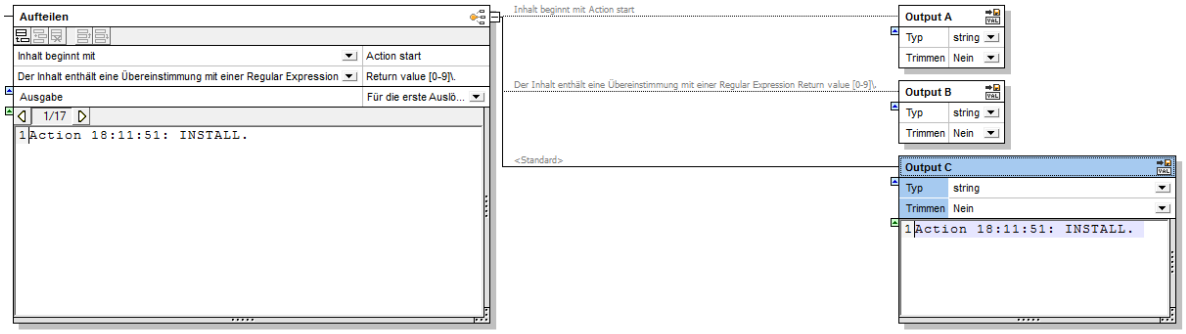
7. Klicken Sie auf die Schaltfläche **Bedingung anhängen** ( ) und fügen Sie eine Bedingung vom Typ **Der Inhalt enthält eine Übereinstimmung mit einer Regular Expression** mit dem Wert `Return value [0-9]\.` hinzu. Wenn Sie jetzt zum Block 5 von 17 navigieren, sehen Sie, dass diese Bedingung diesen Block an einen neuen Container weiterleitet, da der Block eine Übereinstimmung mit der Regular Expression enthält.



Sie haben den Aufteilen-Container nun so konfiguriert, dass er Text auf Basis von Bedingungen an eine andere Ausgabe weiterleitet. Es gibt drei Aufteilen-Bedingungen und drei mögliche Ausgaben (für jede Bedingung eine). In den restlichen Schritten dieses Tutorials wird beschrieben, wie Sie jede der Ausgaben in eine separate Textdatei schreiben können.

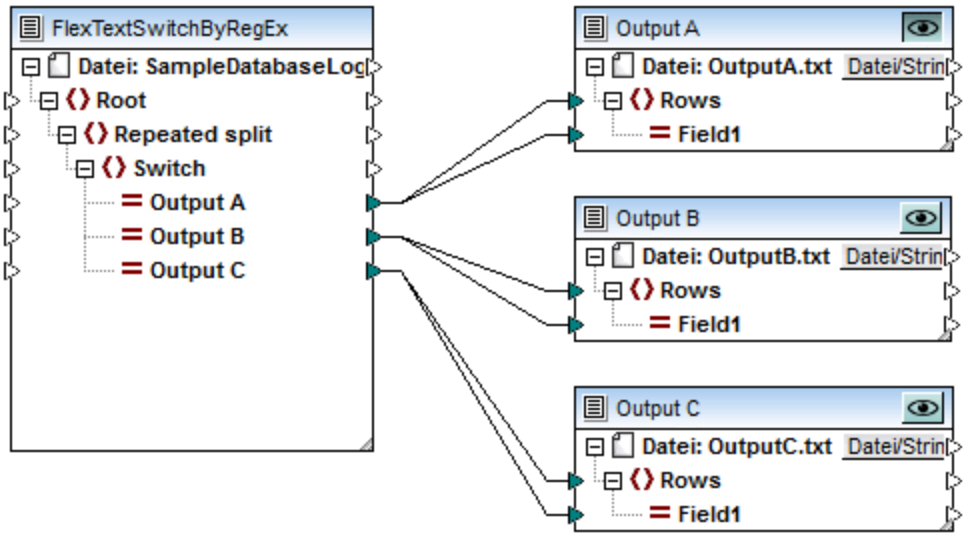
8. Klicken Sie auf die rechte obere Ecke der einzelnen Ausgabe-Container und wählen Sie die Option **Als Wert speichern**. Doppelklicken Sie außerdem auf die Titelleiste der einzelnen Ausgabe-Container und geben Sie einen beschreibenden Namen ein: **Output A**, **Output B** und **Output C**.

© 2016-2022 Altova GmbH



9. Speichern und schließen Sie die FlexText-Vorlage.

Die erforderliche FlexText-Vorlage wurde somit jetzt erstellt. Um zu sehen, wie diese Vorlage funktioniert, öffnen Sie die folgende Tutorial-Datei:
<Dokumente>\Altova\MapForce2023\MapForceExamples\FlexTextSwitchByRegEx.mfd.



FlexTextSwitchByRegEx.mfd

In dem in der Abbildung gezeigten Mapping **FlexTextSwitchByRegEx.mfd** werden Textdaten aus der Datei **<Dokumente>\Altova\MapForce2023\MapForceExamples\SampleDatabaseLog.txt** gelesen und in drei separate Textdateien konvertiert: Output A, Output B und Output C. Jede der Zieltextdateien enthält nur die Zeilen, auf die eine der drei zuvor in der FlexText-Vorlage definierten Bedingungen zutrifft. Um die Ausgabe einer bestimmten Komponente zu sehen, klicken Sie in der rechten oberen Ecke der Komponente auf die Schaltfläche **Vorschaukomponente** ( ) und klicken Sie anschließend auf das Register **Ausgabe**.

4.7 Microsoft OOXML Excel 2007+

MapForce kann im Standardformat *Office Open XML (OOXML)* Daten aus Microsoft Excel 2007+-Arbeitsmappen lesen und in diese schreiben. Dieses Format wurde in Microsoft Office 2007 erstmals verwendet und ist bei Excel-Arbeitsmappen mit der Erweiterung .xlsx verknüpft. Auf der Benutzeroberfläche von MapForce und in dieser Dokumentation werden Microsoft Excel-Datei ab der Excel Version 2007 als Excel 2007+-Dateien bezeichnet.

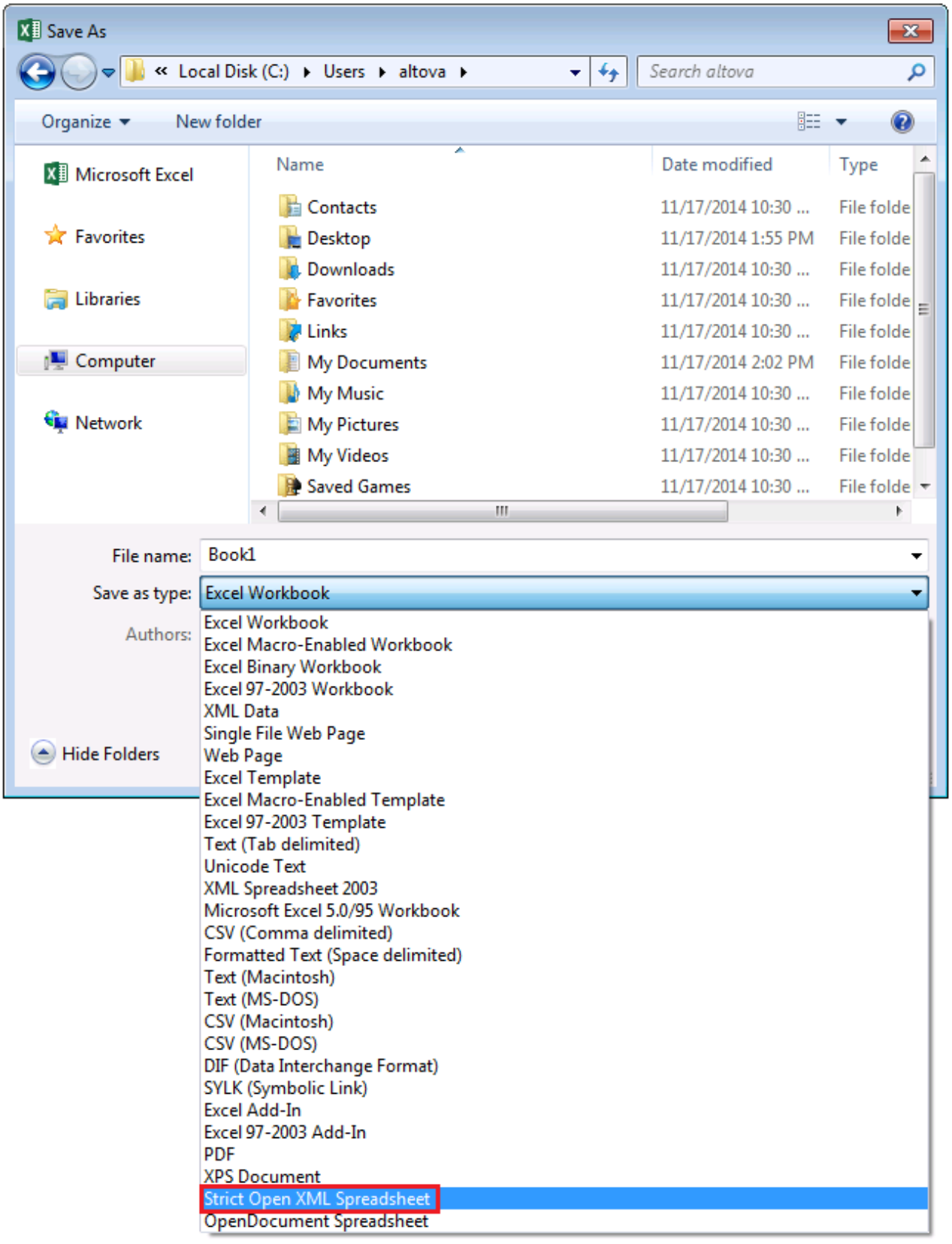
Ab Microsoft Office 2013 wird das Format *Strict Open XML Spreadsheet* ((ISO/IEC 29500 Strict) unterstützt. Das Format *Strict Open XML Spreadsheet* hat ebenfalls die Erweiterung .xlsx. Technisch gesehen, handelt es sich hierbei jedoch um ein eigenes Format, für das strengere Validierungsregeln gelten.

Die folgende Tabelle enthält eine Zusammenfassung, wie das Schreiben von Daten in Excel 2007+-Arbeitsmappen bzw. das Lesen daraus von den verschiedenen MapForce-Transformationssprachen unterstützt wird. Beachten Sie die Unterschiede zwischen dem *Office Open XML*- und dem *Strict Open XML Spreadsheet*-Format.

Microsoft Excel-Format	MapForce-Sprache	Unterstützung für das Lesen	Unterstützung für das Schreiben
Office Open XML	BUILT-IN	Ja	Ja
	C#	Ja	Ja
	Java	Ja	Ja
	XSLT 2.0, XSLT 3.0	Ja	Nein
Strict Open XML Spreadsheet	BUILT-IN	Ja	Nein
	C#	Ja	Nein
	Java	Ja	Nein
	XSLT 2.0. XSLT 3.0	Ja	Nein

Wenn Sie eine mit MapForce generierte *Office Open XML*-Datei ins *Strict Open XML Spreadsheet*-Format konvertieren müssen, öffnen Sie die Arbeitsmappe in Excel 2013 und speichern Sie sie dann als *Strict Open XML-Arbeitsmappe*.

© 2016-2022 Altova GmbH


Save As
« Local Disk (C:) ▸ Users ▸ altova ▸
Search altova
Organize
New folder
Microsoft Excel
Favorites
Libraries
Computer
Network
Name
Date modified
Type
Contacts 11/17/2014 10:30 ... File folde
Desktop 11/17/2014 1:55 PM File folde
Downloads 11/17/2014 10:30 ... File folde
Favorites 11/17/2014 10:30 ... File folde
Links 11/17/2014 10:30 ... File folde
My Documents 11/17/2014 2:02 PM File folde
My Music 11/17/2014 10:30 ... File folde
My Pictures 11/17/2014 10:30 ... File folde
My Videos 11/17/2014 10:30 ... File folde
Saved Games 11/17/2014 10:30 ... File folde
File name: Book1
Save as type: Excel Workbook
Authors:
Hide Folders
Excel Workbook
Excel Macro-Enabled Workbook
Excel Binary Workbook
Excel 97-2003 Workbook
XML Data
Single File Web Page
Web Page
Excel Template
Excel Macro-Enabled Template
Excel 97-2003 Template
Text (Tab delimited)
Unicode Text
XML Spreadsheet 2003
Microsoft Excel 5.0/95 Workbook
CSV (Comma delimited)
Formatted Text (Space delimited)
Text (Macintosh)
Text (MS-DOS)
CSV (Macintosh)
CSV (MS-DOS)
DIF (Data Interchange Format)
SYLK (Symbolic Link)
Excel Add-In
Excel 97-2003 Add-In
PDF
XPS Document
Strict Open XML Spreadsheet
OpenDocument Spreadsheet

Speichern im Format "Strict Open XML-Arbeitsmappe"

Excel 2007+-Dateien weisen in MapForce als Mapping-Komponenten das folgende allgemeine Verhalten auf:

- Sie können Daten von 2007+ von und auf jede andere in MapForce unterstützte Komponente, einschließlich XBRL-Taxonomiedateien mappen (siehe Beispiel "Microsoft Excel auf XBRL" 818).
- Wenn auf Ihrem Rechner Microsoft Excel 2007 oder höher installiert ist, können Sie die Transformationsausgabe sofort auf dem **Ausgabe**-Register des Mapping-Fensters ansehen und diese in einer Datei speichern. Wenn Sie Excel 2007 oder höher nicht installiert haben, können Sie Dateien trotzdem von/auf Excel 2007+-Dateien mappen. In diesem Fall können Sie zwar keine Vorschau des Ergebnisses auf dem Register "**Ausgabe**" anzeigen, das Ergebnis aber dennoch über das Menü **Ausgabe** mit dem Menübefehl **Ausgabedatei speichern** speichern.

  Wenn Sie mehrere HDPI-Monitore mit unterschiedlicher DPI-Awareness verwenden, kann die Vorschau auf die Excel 2007+-Ausgabe nicht direkt in MapForce angezeigt werden. Stattdessen wird eine Meldung ungefähr folgenden Wortlauts angezeigt: "Mit der aktuellen Systemkonfiguration ist eine Einbettung von Excel-Dateien nicht möglich".

- Sie können in den Komponenteneinstellungen 752 auswählen, ob anhand eines Mappings neue Excel-Dateien erstellt oder vorhandene Excel-Dateien aktualisiert werden sollen.
- Wenn Sie die Option zum Erstellen neuer Excel-Dateien auswählen, enthält die generierte Excel 2007+-Ausgabe keine Formatierung, sondern nur Daten.
- Wenn Sie die Option zum Aktualisieren vorhandener Excel-Dateien auswählen, werden die Formatierungsinformationen Ihrer vorhandenen Excel-Datei unverändert beibehalten. Beachten Sie, dass die Aktualisierung von Zellen, die Formeln enthalten, nicht unterstützt wird. Sie können in den Komponenteneinstellungen 752 allerdings festlegen, dass die Mapping-Ausführung bei solchen Zellen mit einem Fehler abgebrochen werden soll oder dass diese ignoriert werden sollen (während im Fenster "Meldungen" weiterhin Warnungen angezeigt werden).
- Als Alternative zum manuellen Generieren und Speichern der Ausgabe können Sie das Mapping-Design in einer MapForce Server-Ausführungsdatei speichern oder es auf einem FlowForce Server bereitstellen und als FlowForce-Server Auftrag bzw. auch mit Hilfe von FlowForce Server-Aufträgen ausführen. Nähere Informationen dazu finden Sie unter Kompilieren eines MapForce Mappings 1442 und Bereitstellen eines MapForce Mappings 1445.
- Falls dies von den Mapping-Komponenten unterstützt wird, können Sie anhand eines Mappings Programmcode generieren (siehe unterstützte Sprache oben) und diesen extern von einer benutzerdefinierten Applikation ausführen lassen. Nähere Informationen dazu finden Sie unter Code Generator 1563. Beachten Sie, dass die Funktion **Bestehende Excel-Dateien aktualisieren**, nur unterstützt wird, wenn die Transformationssprache BUILT-IN ist.

4.7.1 Hinzufügen von Microsoft Excel-Dateien als Mapping-Komponenten

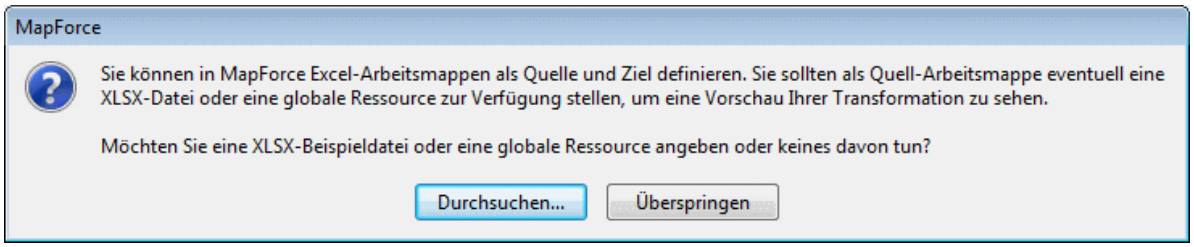
So fügen Sie eine Microsoft Excel 2007+ (*.xlsx)-Komponente zum Mapping-Bereich hinzu:

1. Wählen Sie eine der folgenden Methoden:

   o Klicken Sie im Menü **Einfügen** auf **Excel 2007+-Datei**.

© 2016-2022 Altova GmbH

- Klicken Sie auf die Symbolleisten-Schaltfläche **Excel 2007+-Datei einfügen** ( ) .

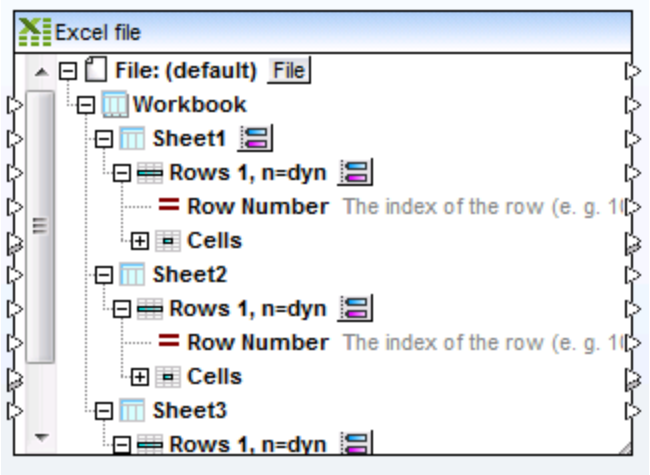


2. Wählen Sie eine der folgenden Methoden:

   - Wenn Sie Daten *von* einer Microsoft Excel-Arbeitsmappen aus mappen möchten, klicken Sie auf **Durchsuchen**, um die gewünschte .xlsx-Datei auszuwählen. MapForce liest die Arbeitsblattnamen und Zellenbereiche aus der Excel-Beispieldatei. Wenn Sie Excel-Dateien als globale Ressourcen definiert haben, können Sie diese über das Dialogfeld "Globale Ressourcen" auswählen (siehe Globale Ressourcen[1462]). Klicken Sie auf **Überspringen,** wenn Sie die Beispieldatei erst später angeben möchten.
   - Wenn Sie Daten *auf* eine Microsoft Excel-Arbeitsmappen mappen wollen, klicken Sie auf **Überspringen**. Standardmäßig generiert MapForce bei der Ausführung der Mapping-Transformation im Mapping-Ordner eine Excel-Ausgabedatei namens **xlsx-mapforce.xlsx**. Gegebenenfalls können Sie den Namen der Ausgabedatei über die Excel 2007+-Komponenteneinstellungen ändern (siehe Informationen zur Excel 2007+-KomponenteAbout the Excel 2007+ Component[741]).

4.7.2 Informationen zur Excel 2007+-Komponente

Wenn Sie eine Excel 2007+-Datei zum Mapping-Bereich hinzufügen, ohne eine Beispieldatei anzugeben (siehe Hinzufügen von Microsoft Excel-Dateien als Mapping-Komponenten[740]), erstellt MapForce eine Standardkomponente, die drei Arbeitsblätter enthält (siehe Abbildung unten). Wenn Sie eine Beispieldatei angeben, liest MapForce diese Datei und erstellt nur die benötigten Arbeitsblätter.



Excel-Standardkomponente

Die Struktur der Excel 2007+-Komponente in MapForce entspricht der Struktur der Daten in der Excel-Arbeitsmappe, mit dem Unterschied, dass die Struktur in MapForce in Form einer Baumstruktur angezeigt wird, über die Sie jede einzelne Zelle mappen können.

Bevor Sie die Excel 2007+-Komponente mit einem anderen Komponententyp verbinden können, müssen Sie in MapForce genau definieren, welche Spalten und Zeilen im Mapping verwendet werden sollen. Im Gegensatz zu anderen MapForce-Komponenten wie XML oder JSON, haben Excel 2007+-Dateien kein explizites Schema, anhand dessen MapForce die Struktur Ihrer Daten ableiten kann. Statt dessen können Sie folgende Einstellungen über MapForce vornehmen:

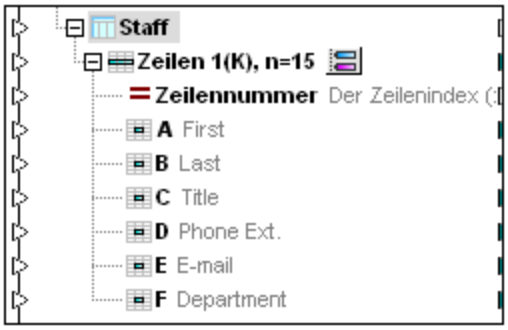
- Welche Daten genau in Ihrer Arbeitsmappe ausgewählt werden sollen (z.B. Arbeitsblätter, benannte Bereiche und Tabellen, Spalten, Zeilen) (Wenn Sie Daten aus einer Arbeitsmappe lesen);
- In welche Arbeitsblätter, Spalten und Zeilen MapForce Daten schreiben soll (wenn Sie Daten in eine Arbeitsmappe schreiben).

Sie können die Komponente auch so konfigurieren, dass die Daten bei derselben Mapping-Operation aus mehreren Stellen in einer Arbeitsmappe gelesen bzw. an mehrere Stellen geschrieben werden sollen.

Die erforderlichen Konfigurationseinstellungen für die Komponente stehen direkt in der Komponente zur Verfügung. Die Informationen dazu finden Sie in der folgenden Tabelle.

Datenelement der Komponente	**Symbol**	**Beschreibung**
Arbeitsmappe		Repräsentiert eine Excel-Arbeitsmappe
Arbeitsblatt		Repräsentiert ein Excel-Arbeitsblatt. Über die Schaltfläche  neben dem ersten Arbeitsblatt können Sie Einstellungen zum Arbeitsblatt definieren, siehe Hinzufügen und Entfernen von Arbeitsblättern[744].
Zeilen		Steht für einen Zeilenbereich in Excel. Sie können mehrere Zeilenbereiche in einem Arbeitsblatt hinzufügen, sodass Sie Daten in derselben Mapping-Operation aus mehreren Zeilenbereichen lesen bzw. in mehrere schreiben können, siehe Hinzufügen und Entfernen von Zeilenbereichen[745]. Für jeden definierten Bereich können Sie eigene Optionen zur Auswahl der Daten definieren. So kann z.B. ein Bereich bei Zeile 1 beginnen und alle Spalten dieser Zeile enthalten, während ein anderer Bereich bei Zeile 3 beginnt und je nach Menge der Daten in der Excel-Quelldatei aus einer dynamischen Anzahl von Zeilen besteht. Um alle Bereichseinstellungen auf einen Blick kenntlich zu machen, wird die Komponente mit den folgenden visuellen Markierungen gekennzeichnet. **Zeilen *n*** Kennzeichnet einen Bereich, der bei der Zeile ***n*** beginnt.

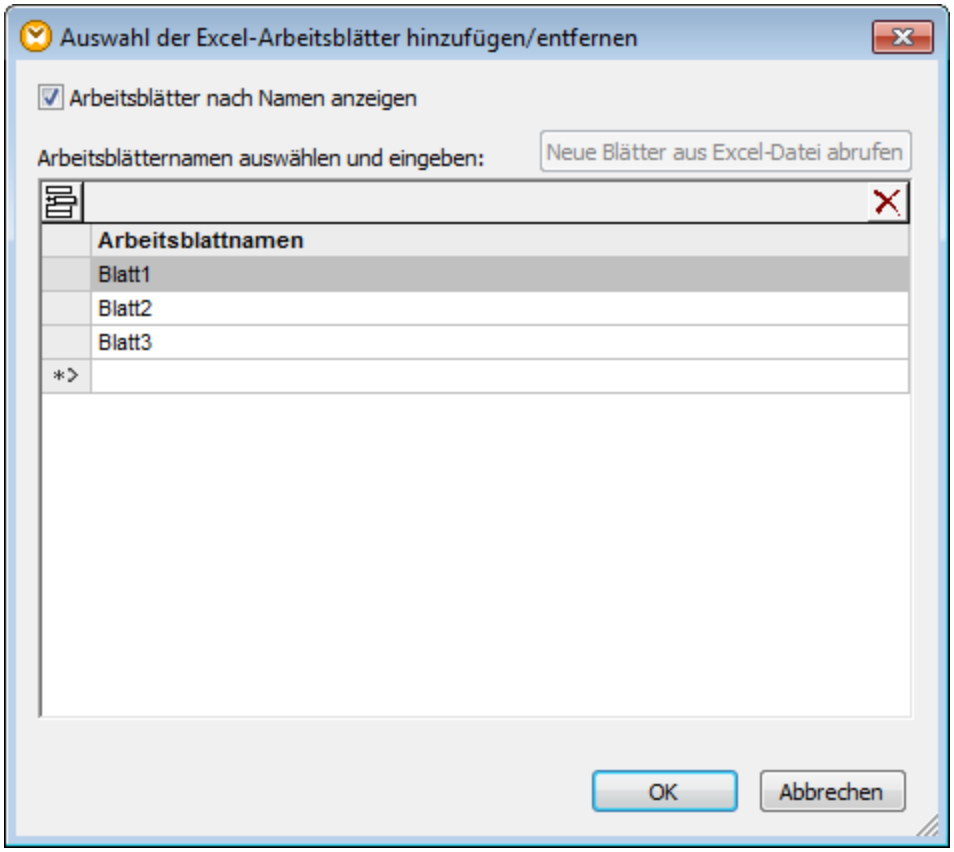
© 2016-2022 Altova GmbH

Datenelement der Komponente	Symbol	Beschreibung
		Zeile *n* Kennzeichnet einen aus einer einzigen Zeile bestehenden Bereich der Zeile *n*. **Zeilen vorher+*n*** Kennzeichnet einen Bereich, der *n* Zeilen nach dem vorherigen Bereich beginnt. **Zeilen *n*(h)** Kennzeichnet einen Bereich, der an der Zeile *n* beginnt und dessen erste Zeile als Kopfzeile definiert ist. **n=*n*** Kennzeichnet einen Bereich, der aus genau *n* Zeilen besteht. **n=dyn** Kennzeichnet einen dynamischen Bereich. Dynamische Bereiche können unbegrenzt viele Zeilen haben. Über die Schaltfläche neben jedem Zeilenbereich können Sie erweiterte Datenauswahleinstellungen für diesen Bereich definieren.
Zellen		Repräsentiert alle Zellen (Spalten) einer bestimmten Zeile. Dieses Element wird angezeigt, wenn für die Komponente eine einzige Zelle für alle Spalten angezeigt wird (dies ist das MapForce-Standardverhalten). Alternativ dazu können Sie eine Komponente so konfigurieren, dass jede Spalte separat angezeigt wird. Diese würde aussehen, wie in der Abbildung unten gezeigt. 
Auswahl ändern		Über die Schaltfläche neben jedem Arbeitsblatt bzw. jeder Zeile können Sie weitere Einstellungen für diesen Kontext vornehmen. Über diese Schaltfläche können Sie die Mapping-Struktur der Excel-Komponente nach Bedarf ändern, siehe Excel 2007+-Komponenteneinstellungen 752. So können Sie z.B. beim Lesen von Daten aus einer Excel-Datei das Arbeitsblatt, die Zeile und Spalte, aus der MapForce

Datenelement der Komponente	Symbol	Beschreibung
		die Daten lesen soll definieren. Wenn Sie Daten in eine Excel-Datei schreiben, können Sie das Arbeitsblatt, die Zeile und Spalte, in die MapForce die Daten schreiben soll, definieren.

4.7.3 Hinzufügen und Entfernen von Arbeitsblättern

Sie können Arbeitsblätter der 2007+-Komponente entweder manuell oder durch Neuladen aus einer .xslx-Input-Datei hinzufügen oder entfernen. Klicken Sie dazu auf die Schaltfläche neben einem Arbeitsblatt-Node.



Dialogfeld Auswahl der Excel-Arbeitsblätter hinzufügen/entfernen

Für Arbeitsblätter stehen die folgenden Optionen zur Verfügung.

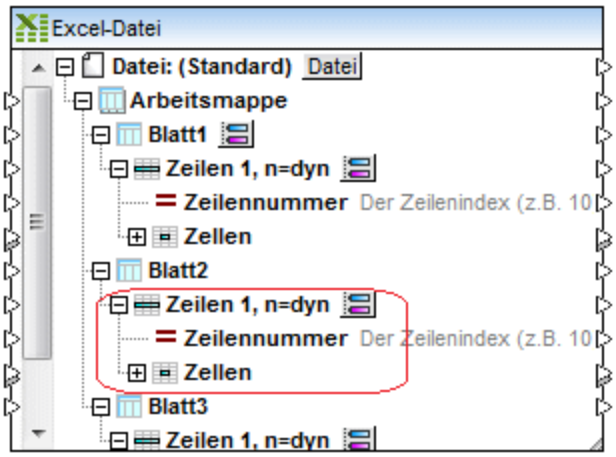
Arbeitsblätter nach Namen anzeigen	Diese Option muss ausgewählt werden, wenn jedes Arbeitsblatt in Ihrer Arbeitsmappe ein anderes Layout hat und daher in MapForce als separates Datenelement angezeigt werden muss. Wenn alle Arbeitsblätter in Ihrer Arbeitsmappe eine identische Struktur haben, können Sie diese Option deaktivieren. MapForce

© 2016-2022 Altova GmbH

	reduziert in diesem Fall die Arbeitsblatt-Datenelemente zu einem einzigen Datenelement, das die geordnete Sammlung aller Arbeitsblätter repräsentiert. Wenn eine Arbeitsmappe mehrere Arbeitsblätter hat und die Option **Arbeitsblätter nach Namen anzeigen** deaktiviert ist, behandelt MapForce die Arbeitsmappe wie ein einziges Arbeitsblatt. Dadurch können im Mapping beliebig viele Arbeitsblätter auf einmal verarbeitet werden, doch müssen alle Arbeitsblätter dieselbe Struktur haben. Siehe Beispiel: Konvertieren von Excel-Zeilen in XML-Dateien 757.
Einfügen ( )	Fügt vor dem ausgewählten Arbeitsblatt ein neues ein.
Anhängen ( )	Hängt ein neues Arbeitsblatt an. Geben Sie in das Textfeld rechts vom Symbol einen Namen ein.
Löschen ( )	Löscht das aktuell ausgewählte Arbeitsblatt.
Neue Blätter aus Excel-Datei abrufen	Integriert die neuen in der Excel-Input-Datei gefundenen Arbeitsblätter in die Komponente. Diese Einstellung ist bei Quellkomponenten, die eine Excel-Input-Datei haben, relevant und aktiviert.

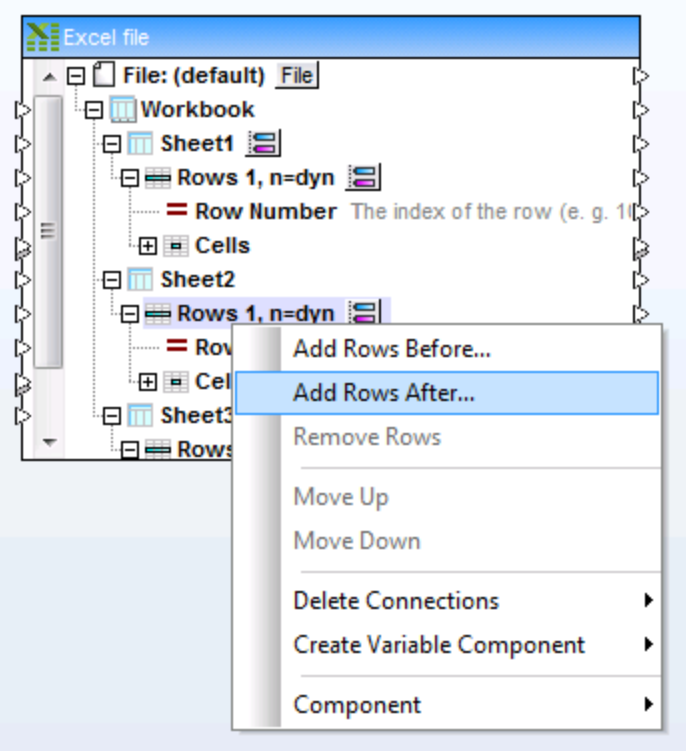
4.7.4 Hinzufügen und Entfernen von Zeilenbereichen

Sie können MapForce anweisen, einen bestimmten Zellenbereich an einer bestimmten Stelle in einem Arbeitsblatt zu lesen (oder zu schreiben). In der folgenden Beispielkomponente sehen Sie einen Bereich, der in Blatt2 der Arbeitsmappe zur Verfügung steht.



Standardmäßig beginnt jeder Bereich bei Zeile 1 und iteriert dynamisch durch n Zeilen (**n=dyn**). Sie können diese und andere Einstellungen jedoch bei Bedarf ändern (siehe Auswählen von Zellenbereichen 746).

Sie können im selben Arbeitsblatt so viele Zeilenbereiche wie notwendig erstellen und die nicht benötigten Bereiche entfernen. Um einen neuen Zeilenbereich hinzuzufügen, klicken Sie mit der rechten Maustaste auf einen beliebigen Zeilen-Node ( ) und wählen Sie den Befehl **Zeilen davor hinzufügen** bzw. **Zeilen danach hinzufügen**.



Hinzufügen von Zellenbereichen

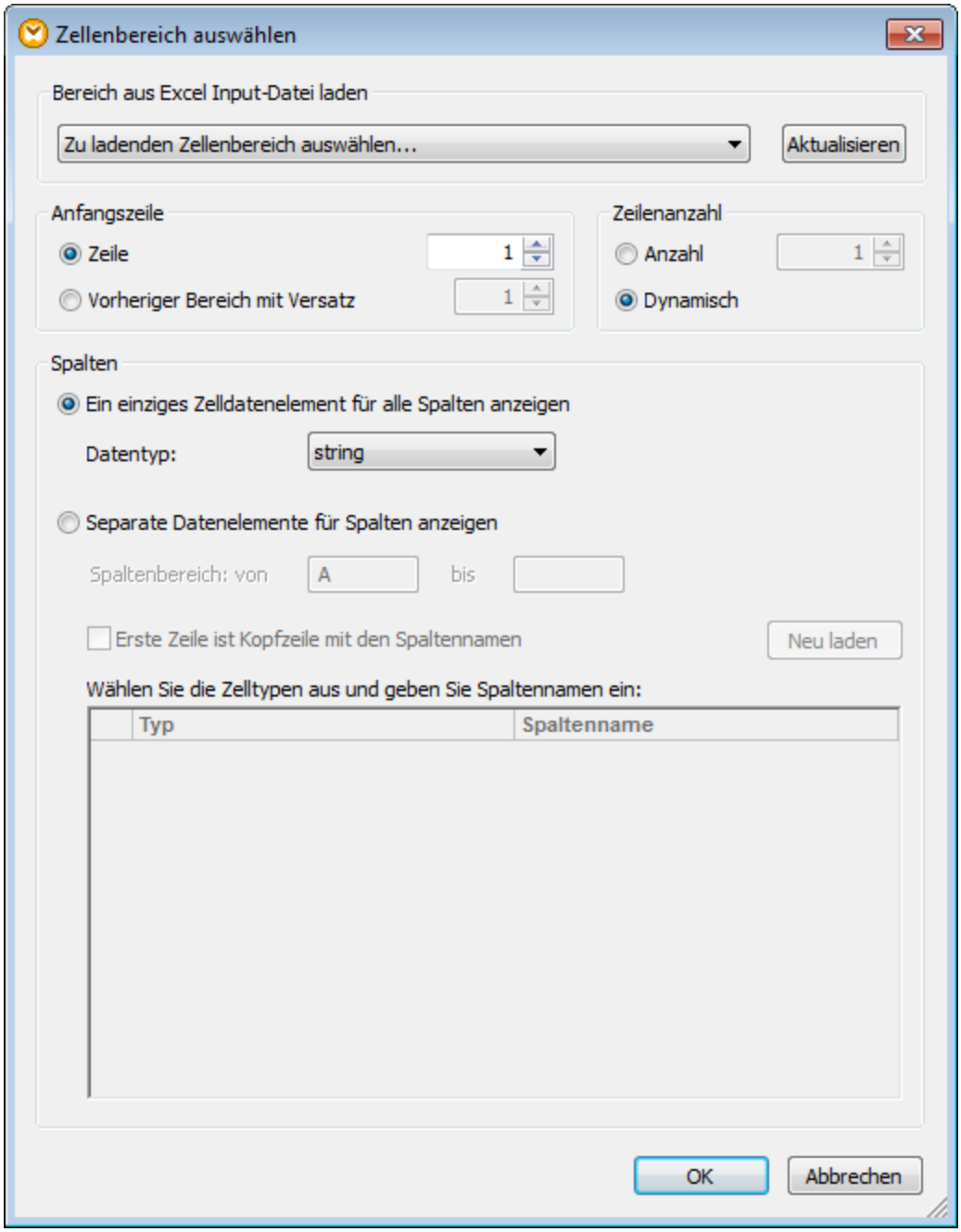
Um einen Bereich zu löschen, klicken Sie mit der rechten Maustaste darauf und wählen Sie den Befehl **Zeilen entfernen**.

Um einen Bereich in einer Komponente nach oben bzw. nach unten zu verschieben, klicken Sie mit der rechten Maustaste darauf und wählen Sie den Befehl **Nach oben** bzw. **Nach unten**.

4.7.5 Auswählen von Zellenbereichen

Über das Dialogfeld "Zellenbereich auswählen" können Sie definieren, welcher Zellenbereich (beim Lesen von Daten aus einer Arbeitsmappe) von MapForce gelesen werden soll, bzw. in welchen Zellenbereich (beim Schreiben in eine Arbeitsmappe) geschrieben werden soll. Um dieses Dialogfeld aufzurufen, klicken Sie neben einem Zellenbereich der Komponente auf die Schaltfläche .

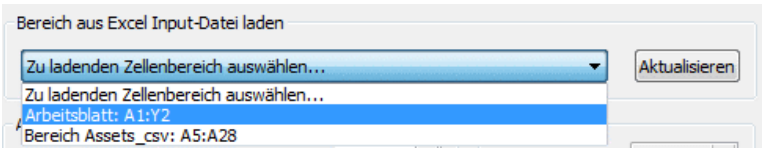
© 2016-2022 Altova GmbH



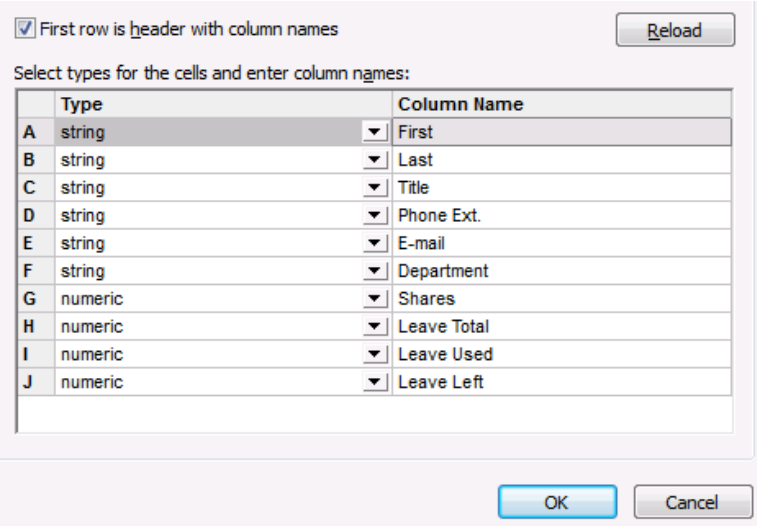
Dialogfeld "Zellenbereich auswählen"

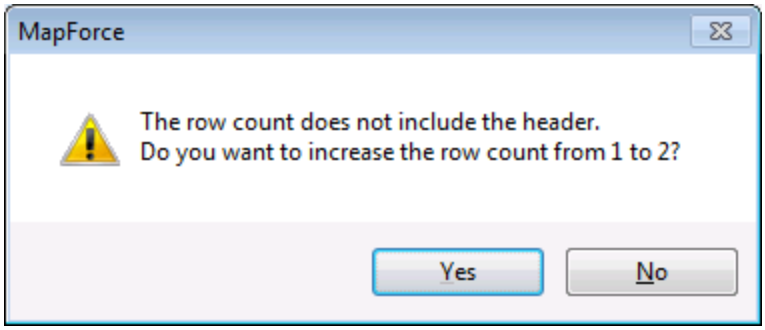
Es stehen die folgenden Einstellungen zur Verfügung.

Bereich aus Excel Input-Datei laden	Wenn Daten aus einer Arbeitsmappe gelesen werden, verwenden Sie diese Option, um einen bestimmten Bereich des Arbeitsblatts, einen benannten Bereich oder eine Tabelle auszuwählen.

	 Wenn die Option **Arbeitsblätter nach Namen anzeigen** deaktiviert ist (siehe Hinzufügen und Entfernen von Arbeitsblättern 744), sind in der Liste die Daten aus allen Arbeitsblättern zu sehen. Mit der Schaltfläche **Aktualisieren** werden die Zellenbereiche anhand der Excel-Input-Datei aktualisiert. Beachten Sie, dass derzeit nur rechteckige Bereiche unterstützt werden. Wenn Daten in eine Arbeitsmappe geschrieben werden, steht diese Option nicht zur Verfügung.
Anfangszeile	Mit Hilfe der Option **Zeile** können Sie die erste Datenzeile für den jeweiligen Bereich definieren. Wenn Sie z.B. "5" als Anfangszeile eingeben, so wird als erste Zeile aus dem Blatt die Zeile 5 verwendet. Die Option **Vorheriger Bereich mit Versatz** eignet sich dann, wenn es im selben Arbeitsblatt einen weiteren Bereich gibt. MapForce verschiebt daraufhin den aktuellen Bereich um *N* Zeilen ab dem zuvor definierten Bereich nach unten. Der Mindestversatzwert ist 1.
Zeilenanzahl	*Wenn Daten aus einer Arbeitsmappe gelesen werden* **Anzahl** definiert die genaue Anzahl der Zeilen, aus denen Daten beginnend mit der in **Anfangszeile** definierten Position (siehe vorherige Option) gelesen werden sollen. Dieser Wert wird automatisch ausgefüllt, wenn Sie einen benannten Excel-Bereich bzw. eine Tabelle auswählen. Mit der Option **Dynamisch** wird MapForce angewiesen, alle in den Quelldaten gefundenen Zeilen beginnend mit der **Anfangszeile** zu lesen. Verwenden Sie diese Option nur, wenn Ihr Bereich der letzte definierte Bereich des Arbeitsblatts ist, da sonst Daten aus darauf folgenden Bereichen nicht aus der Excel-Quelldatei ausgewählt werden. *Wenn Daten in eine Arbeitsmappe geschrieben werden* **Anzahl** definiert die genaue Anzahl der Zeilen, in die Daten beginnend mit der in **Anfangszeile** definierten Position (siehe vorherige Option) geschrieben werden sollen. Beachten Sie Folgendes: Wenn Ihre Input-Instanz mehr als die durch **Anzahl** definierte Anzahl von Zeilen enthält, so schreibt MapForce nur die durch **Anzahl** definierte Anzahl von Zeilen in die Komponente und

© 2016-2022 Altova GmbH

	ignoriert die restlichen Daten, ohne eine Warnmeldung auszugeben. Mit der Option **Dynamisch** wird MapForce angewiesen, alle in der Input-Instanz gefundenen Zeilen beginnend mit der **Anfangszeile** zu schreiben. Wenn Sie mit Hilfe der Option **Erste Zeile ist Kopfzeile mit den Spaltennamen** eine Kopfzeile definiert haben, so wird die Kopfzeile bei **Anzahl** nicht berücksichtigt (siehe Beispieldatei **Excel_Company_to_XML.mfd**). 
Ein einziges Zelldatenelement für alle Spalten anzeigen	Reduziert alle Zelldatenelemente zu einem einzigen mapbaren *Zelldatenelement*, wie unten gezeigt.  *Wenn Daten aus einer Arbeitsmappe gelesen werden* Verwenden Sie diese Option, wenn Sie alle Zellen einer bestimmten Zeile auslesen möchten. Ein Beispiel dazu finden Sie in den Dateien **ExcelColumnsToRecords.mfd** und **ExcelWith2Dimensions.mfd** aus dem **MapForceExamples**-Projekt im Ordner **OOXML Excel 2007+**.

	Wenn Daten in eine Arbeitsmappe geschrieben werden Verwenden Sie diese Option, um Daten in eine oder mehrere Zellen in derselben Zeile zu schreiben. Ein Beispiel dazu finden Sie in der Datei **Altova_Hierarchical_Excel.mfd.**
Separate Datenelemente für Spalten anzeigen	Mit Hilfe dieser Option können einzelne Spalten des angegebenen Zeilenbereichs aufgerufen werden. Wenn Sie einen Arbeitsblattbereich, einen benannten Bereich oder eine Tabelle ausgewählt haben, werden die Spaltennamen automatisch ausgefüllt. Andernfalls können Sie bestimmte Spaltennamen auswählen, indem Sie die entsprechenden Buchstaben dafür in die Textfelder **von** und **bis** eintragen. Wenn sich die Bereiche in der Excel-Input-Datei geändert haben, klicken Sie auf **Neu laden**, um die Änderungen in der Komponente zu aktualisieren. Um MapForce anzuweisen, dass die erste Zeile eines Bereichs als Spaltenüberschrift für diesen Bereich verwendet werden soll, aktivieren Sie die Option **Erste Zeile ist Kopfzeile mit den Spaltennamen**. Wenn Sie diese Option aktivieren oder deaktivieren und die **Zeilenanzahl** definiert wurde, fordert Sie MapForce auf, den Wert **Zeilenanzahl** optional anzupassen, damit dieser Wert nicht eine Zeile zu viel oder zu wenig anzeigt.  Beachten Sie, dass die Kopfzeile bei der Einstellung **Zeilenanzahl** nicht berücksichtigt wird.

4.7.6 Einfügen von Spalten zwischen vorhandenen Spalten

Wenn Sie Daten auf ein Excel-Arbeitsblatt mappen, können Sie ganz einfach neue Spalten zwischen vorhandene Spalten einfügen, ohne die bestehenden Mapping-Verbindungen neu erstellen zu müssen. Klicken Sie dazu in der Excel-Komponente mit der rechten Maustaste auf eine Zelle und wählen Sie im Kontextmenü den Befehl **Zelle davor hinzufügen**.

Anmerkung: Der Befehl **Zelle davor hinzufügen** steht zur Verfügung, wenn die Zelle zu einem Zeilenbereich gehört, d.h. sie muss ein Child eines "Zeile"- oder "Zeilen"-Elements einer Excel-

 © 2016-2022 Altova GmbH

Komponente sein). Außerdem muss im Dialogfeld "Zellenbereich auswählen" die Option **Separate Datenelemente für Spalten anzeigen** aktiv sein (siehe Auswählen von Zellenbereichen 746).

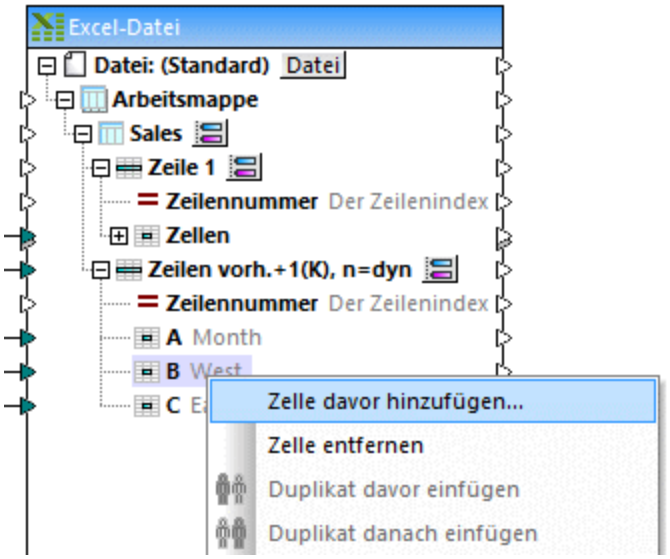
Angenommen, Sie haben ein Mapping erstellt, das Daten aus einer XML-Datei in ein Excel-Arbeitsblatt (wie z.B. **Sales_to_Excel.mfd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**) schreibt. Das Ergebnis dieses Mappings ist ein Excel-Arbeitsblatt bestehend aus drei Spalten (Month, West und East), wie unten gezeigt.

	A	B	C	D
2	Month	West	East	
3	Jan	110.4	75.3	
4	Feb	114.3	65.2	
5	Mar	134.2	86.1	
6	Apr	107.3	112.1	
7	May	114.4	93.8	
8	Jun	113.9	72.4	
9	Jul	89.4	67.4	
10	Aug	95.3	84.9	
11	Sep	107.2	99.5	
12	Oct	129.7	82.5	
13	Nov	137.1	101.9	
14	Dec	152.6	120.6	
15				

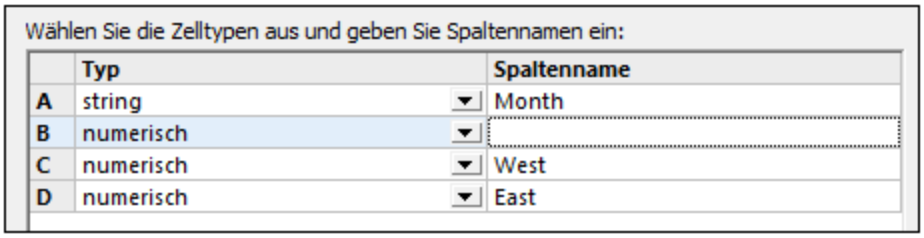
Ausgabe des Mappings Sales_to_Excel.mfd

Angenommen, Sie müssen unmittelbar vor "West" eine neue Spalte einfügen (z.B. "North"). Dazu müssen Sie keine der bestehenden Verbindungen neu ziehen. Gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste im Mapping auf die Zelle B und wählen Sie im Kontextmenü den Befehl **Zelle davor hinzufügen**.

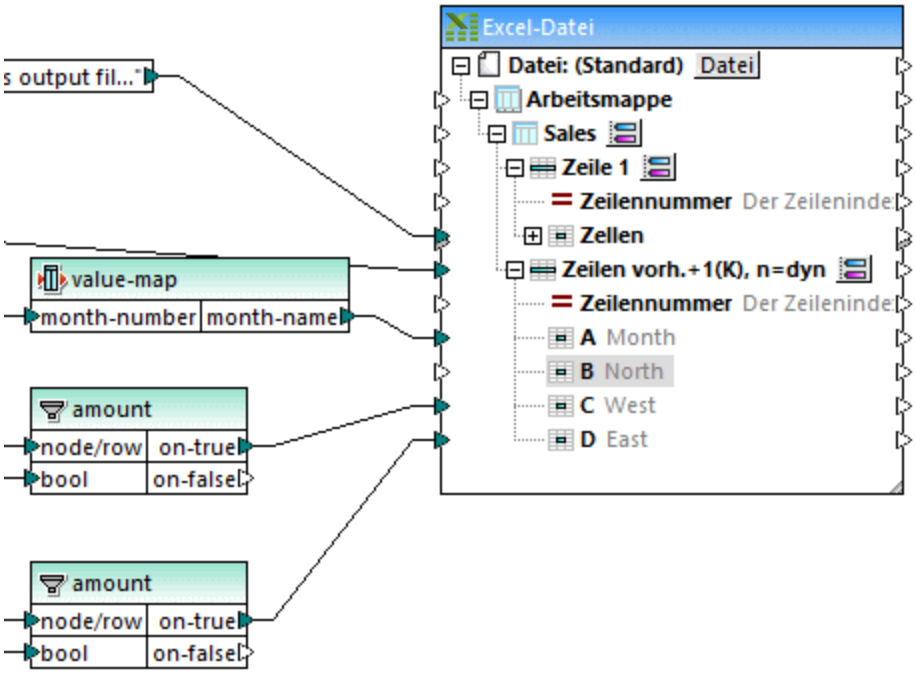


2. Sie können optional einen Spaltennamen (z.B. "North") eingeben.



3. Klicken Sie auf OK.

Die neue Spalte (in diesem Beispiel "North") wurde nun zum Mapping hinzugefügt und Sie können nun eine Mapping-Verbindung zu dieser Spalte ziehen. Dies hat keine Auswirkung auf die bereits bestehenden Verbindungen.



Sales_to_Excel.mfd Mapping (geändert)

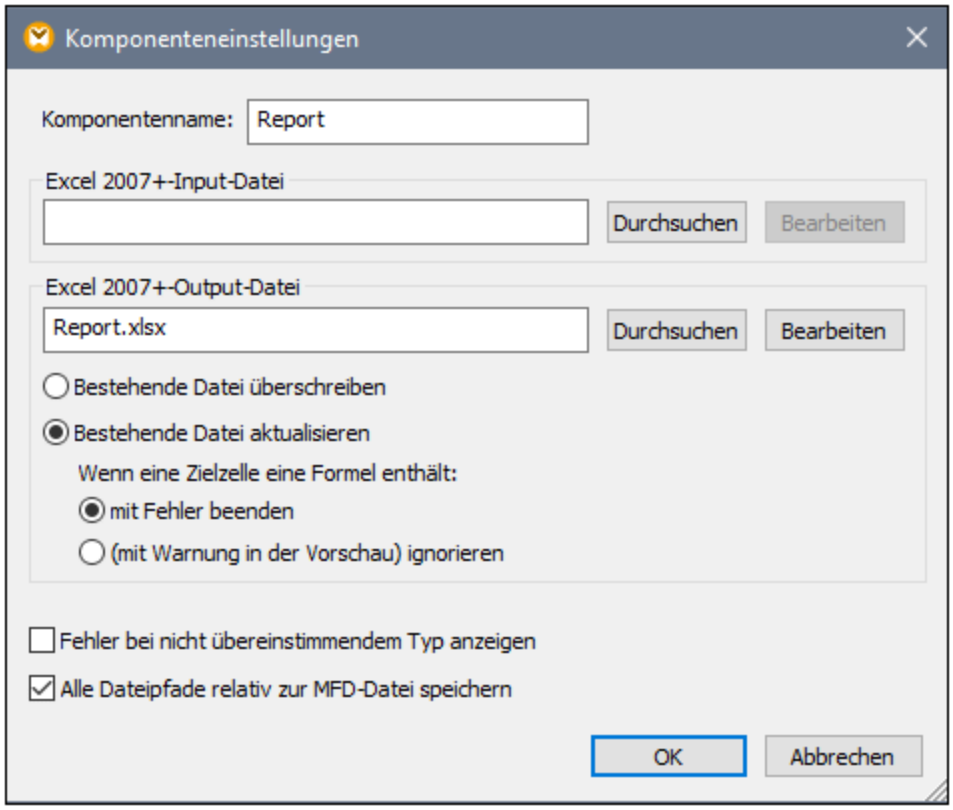
Auf ähnliche Weise können vorhandene Spalten auch entfernt werden: Klicken Sie mit der rechten Maustaste auf eine Zelle im Mapping und wählen Sie im Kontextmenü den Befehl **Zelle entfernen**.

4.7.7 Excel 2007+-Komponenteneinstellungen

Nachdem Sie eine "Excel 2007+"-Komponente zum Mapping-Bereich hinzugefügt haben, können Sie die Einstellungen dafür über das Dialogfeld "Komponenteneinstellungen" vornehmen. Sie können das Dialogfeld "Komponenteneinstellungen" auf eine der folgenden Arten öffnen:

© 2016-2022 Altova GmbH

- Klicken Sie im Menü **Komponente** auf den Befehl **Eigenschaften** (dieses Menü wird aktiv, wenn Sie eine Komponente auswählen).
- Doppelklicken Sie auf den Komponententitel.
- Klicken Sie mit der rechten Maustaste auf den Komponententitel und wählen Sie **Eigenschaften**.



Das Dialogfeld "Komponenteneinstellungen"

Es stehen die folgenden Einstellungen zur Verfügung.

Komponentenname	Der Komponentenname wird bei der Erstellung der Komponente automatisch generiert. Sie können den Namen jedoch jederzeit ändern. Der Komponentenname kann Leerzeichen und Punkte enthalten, darf aber keine Schrägstriche, umgekehrten Schrägstriche, Doppelpunkte, doppelten Anführungszeichen und voran- oder nachgestellte Leerzeichen enthalten. Beachten Sie beim Ändern des Namens einer Komponente Folgendes: • Wenn Sie das Mapping auf FlowForce Server bereitstellen möchten, muss der Komponentenname eindeutig sein. • Es wird empfohlen, nur Zeichen zu verwenden, die über die Befehlszeile eingegeben werden können. Länderspezifische Sonderzeichen sind in Windows und der Befehlszeile eventuell unterschiedlich kodiert.
Excel 2007+-Input-Datei	Definiert den Namen und Pfad der Microsoft Excel-Datei, aus der MapForce die Daten liest. Dieses Feld ist in einer

	Quellkomponente von Bedeutung, wenn MapForce die Excel-Arbeitsblattnamen und -spalten daraus liest. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie die neue Datei aus. Um die Daten in Microsoft Excel zu bearbeiten, klicken Sie auf **Bearbeiten**.
Excel 2007+-Output-Datei	Definiert den Namen oder Pfad der Microsoft Excel-Datei, in die MapForce die Daten schreibt. Dieses Feld hat für eine Zielkomponente eine Bedeutung. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie die neue Datei aus. Um die Daten in Microsoft Excel zu bearbeiten, klicken Sie auf **Bearbeiten**. Wenn Sie keine Ausgabedatei definieren, generiert MapForce die Ausgabe in einer Datei namens **xlsx-mapfore.xlsx**. Standardmäßig wird diese Datei im selben Ordner generiert, in dem sich die .mfd-Datei befindet, es sei denn, Sie haben einen anderen Pfad konfiguriert.
Bestehende Datei überschreiben	Wenn diese Option ausgewählt ist, wird bei Ausführung des Mappings die im Textfeld **Excel 2007+-Output-Datei** (siehe oben) definierte Datei generiert. Die generierte Excel-Datei enthält keine Formatierung. Wenn im Ausgabeverzeichnis bereits eine Datei desselben Namens vorhanden ist, wird diese bei Ausführung der Mapping-Transformation überschrieben. Diese Option wird unterstützt, wenn als Transformationssprache C#, Java oder BUILT-IN ausgewählt ist.
Bestehende Datei aktualisieren	Wenn diese Option ausgewählt ist, wird bei Ausführung des Mappings die im Textfeld **Excel 2007+-Output-Datei** (siehe oben) definierte Datei selektiv aktualisiert (und nicht überschrieben). Dabei werden alle designierten Arbeitsblätter[744], Zeilen[745] und Zellen[746] aus der vorhandenen Datei durch Daten aus dem Mapping ersetzt. Alle Formatierungen aus der vorhandenen Datei werden unverändert beibehalten. Beachten Sie, dass vorhandene Dateien nur dann aktualisiert werden können, wenn die Zellen keine Formeln enthalten. Aktivieren Sie die Option **mit Fehler beenden**, damit die Mapping-Ausführung beendet wird, wenn eine Zelle mit Formeln vorkommt. Aktivieren Sie die Option **(mit Warnung in der Vorschau) ignorieren**, damit solche Zellen ignoriert werden und im Fenster "Meldungen" eine Warnung angezeigt wird. Diese Option wird nur unterstützt, wenn als Transformationssprache BUILT-IN ausgewählt ist.

© 2016-2022 Altova GmbH

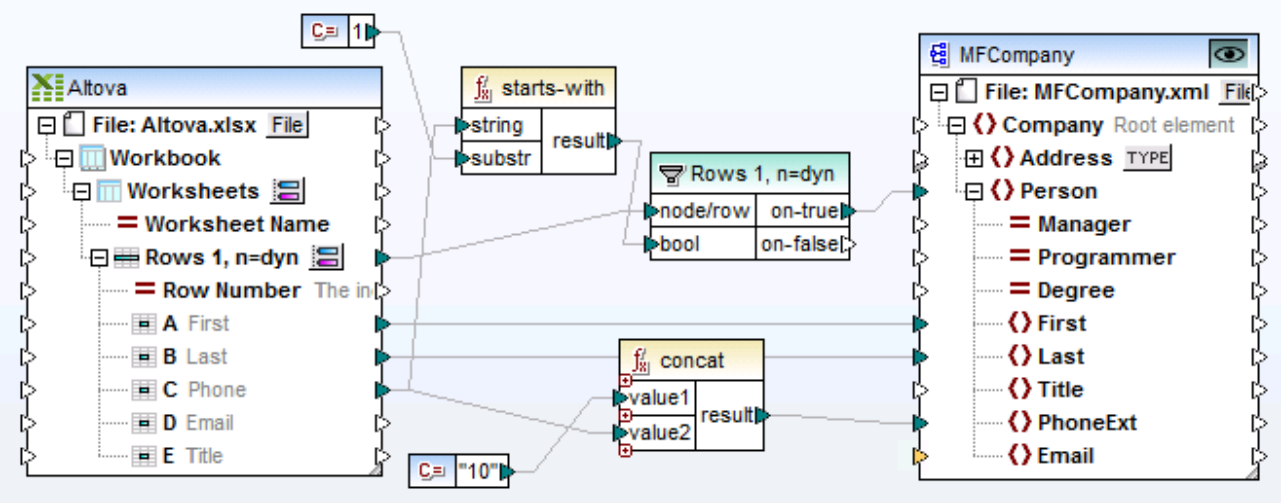
Fehler bei nicht übereinstimmendem Typ anzeigen	Aktiviert die Anzeige von Fehlermeldungen im Fenster "Meldungen", wenn ein in der Komponente deklarierter Datentyp nicht mit den Daten in der .xlsx-Input-Datei übereinstimmt. Angenommen, diese Einstellung ist aktiviert und Sie haben eine Spalte im Dialogfeld *Zellenbereich auswählen* als numerisch deklariert. Wenn die Excel-Datei Textdaten in dieser Spalte enthält, wird im Fenster "Meldungen" eine entsprechende Fehlermeldung angezeigt, nachdem Sie diese Spalte mit einem Output verbunden haben und eine Vorschau der Ausgabe angezeigt haben.
Alle Dateipfade relativ zur MFD-Datei speichern	Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld "Komponenteneinstellungen" angezeigten Dateipfade relativ zum Ordner, in dem sich die MapForce Design (.mfd)-Datei befindet. Siehe auch Verwenden relativer Pfade in einer Komponente 85.

4.7.8 Beispiel: Mappen von Excel 2007+ auf XML

Die im folgenden Beispiel verwendete Mapping-Datei steht unter dem Pfad **<Dokumente>\Altova\MapForce2023\MapForceExamples\Excel-mapping.mfd** zur Verfügung. Wenn Sie die Beispieldatei öffnen, sehen Sie, dass sie drei verschiedene Mapping-Transformationen enthält. In diesem Abschnitt werden die beiden oberen Transformationen erläutert.

Ziel des Mappings ist Folgendes:

- Auswahl nur der Personen aus der Quellkomponente, deren Durchwahl mit "1" beginnt (Spalte C in der Arbeitsmappe).
- Hinzufügen des Präfix "10" zur ursprünglichen Zahl und Schreiben der Daten in die XML-Datei - zusammen mit den Vor- und Zunamen der entsprechenden Personen.



Excel-mapping.mfd (Beispiel 1)

Das Mapping ist folgendermaßen konfiguriert:

- **Altova.xlsx** ist die Excel 2007+-Arbeitsmappe, die als Quellkomponente verwendet wird. Die Spalten A und B stellen die Vor- bzw. Zunamen zur Verfügung. Spalte C enthält die Telefondurchwahl.
- Beide Arbeitsblätter der Arbeitsmappe werden in der Komponente als ein einziger Node angezeigt, d.h. die Option **Arbeitsblätter nach Namen anzeigen** ist deaktiviert. Sie sehen dies am allgemeinen Titel "Arbeitsblätter" unter dem Eintrag Arbeitsmappe.
- Die Funktion `starts-with` überprüft, ob die Durchwahl (Spalte C) mit "1" beginnt. Wenn das Ergebnis "true" ist, werden diese Datensätze von der Filterkomponente übergeben.
- Der **concat**-Filter stellt den einzelnen Durchwahlen eine "10" voran und schreibt sie in das Datenelement PhoneExt.
- MFCompany.xsd ist die Zielkomponente und enthält bei Ausgabe der Daten die gefilterten Personendaten.

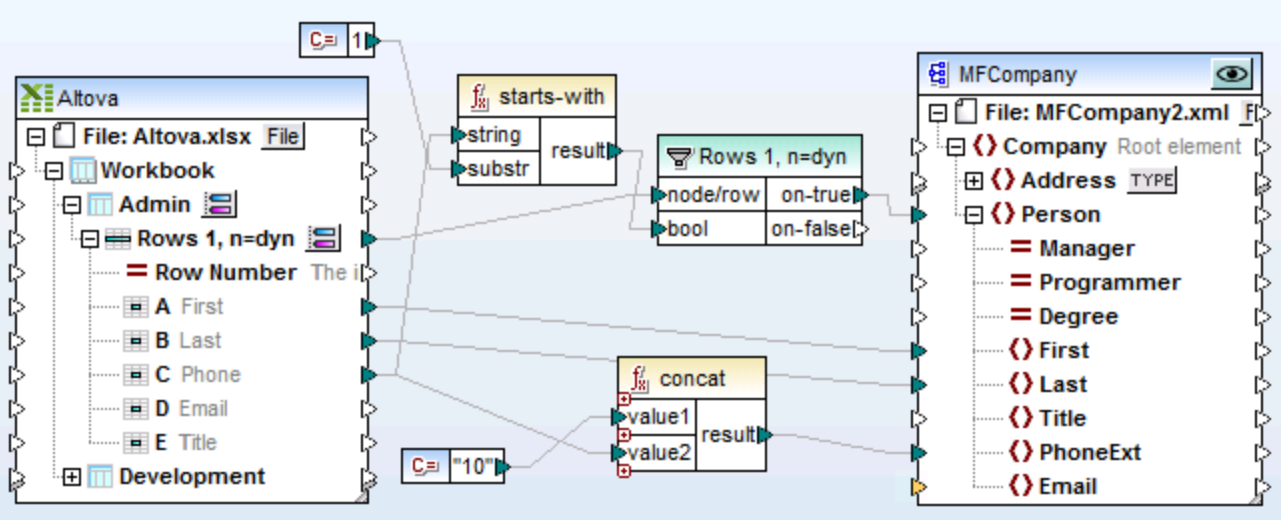
Das Ergebnis des Mappings ist, dass vier Personen mit ihren Daten auf die XML-Datei gemappt wurden.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Company xsi:schemaLocation="http://my-company.com/namespace
3    <Person>
4      <First>Steve</First>
5      <Last>Meier</Last>
6      <PhoneExt>10114</PhoneExt>
7    </Person>
8    <Person>
9      <First>Max</First>
10     <Last>Nafta</Last>
11     <PhoneExt>10122</PhoneExt>
12   </Person>
13   <Person>
14     <First>Carl</First>
15     <Last>Franken</Last>
16     <PhoneExt>10147</PhoneExt>
17   </Person>
18   <Person>
19     <First>Mark</First>
20     <Last>Redgreen</Last>
21     <PhoneExt>10152</PhoneExt>
22   </Person>
23 </Company>
```

Das zweite Mapping ist mit dem ersten bei auf einen Punkt identisch: Im zweiten Mapping wurden die Arbeitsblätter mit Hilfe der Option **Arbeitsblätter nach Namen anzeigen** einzeln aktiviert. Dieses Mapping ist folgendermaßen konfiguriert:

- Die Arbeitsblätter **Admin** und **Development** werden beide unterhalb des Eintrags Arbeitsmappe angezeigt.
- Es wurden nur Verbindungen vom Arbeitsblatt **Admin** zur Zielkomponente definiert.

© 2016-2022 Altova GmbH



Excel-mapping.mfd (Beispiel 2)

Das Ergebnis des Mappings: Es wurden nur zwei Personen mit ihren Daten auf die XML-Datei gemappt.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Company xsi:schemaLocation="http://my-company.com/namespace
3    <Person>
4      <First>Steve</First>
5      <Last>Meier</Last>
6      <PhoneExt>10114</PhoneExt>
7    </Person>
8    <Person>
9      <First>Max</First>
10     <Last>Nafta</Last>
11     <PhoneExt>10122</PhoneExt>
12   </Person>
13 </Company>
```

4.7.9 Beispiel: Konvertieren von Excel-Zeilen in XML-Dateien

Excel-Dateien können aus mehreren Arbeitsblättern mit jeweils unterschiedlicher Struktur bestehen, was bedeutet, dass die Anzahl und Reihenfolge von Spalten sich voneinander unterscheiden kann. Es gibt allerdings auch Excel-Dateien, bei denen alle Arbeitsblätter dieselbe Struktur haben und bei denen sich nur der Name und die Daten des Arbeitsblatts voneinander unterscheiden.

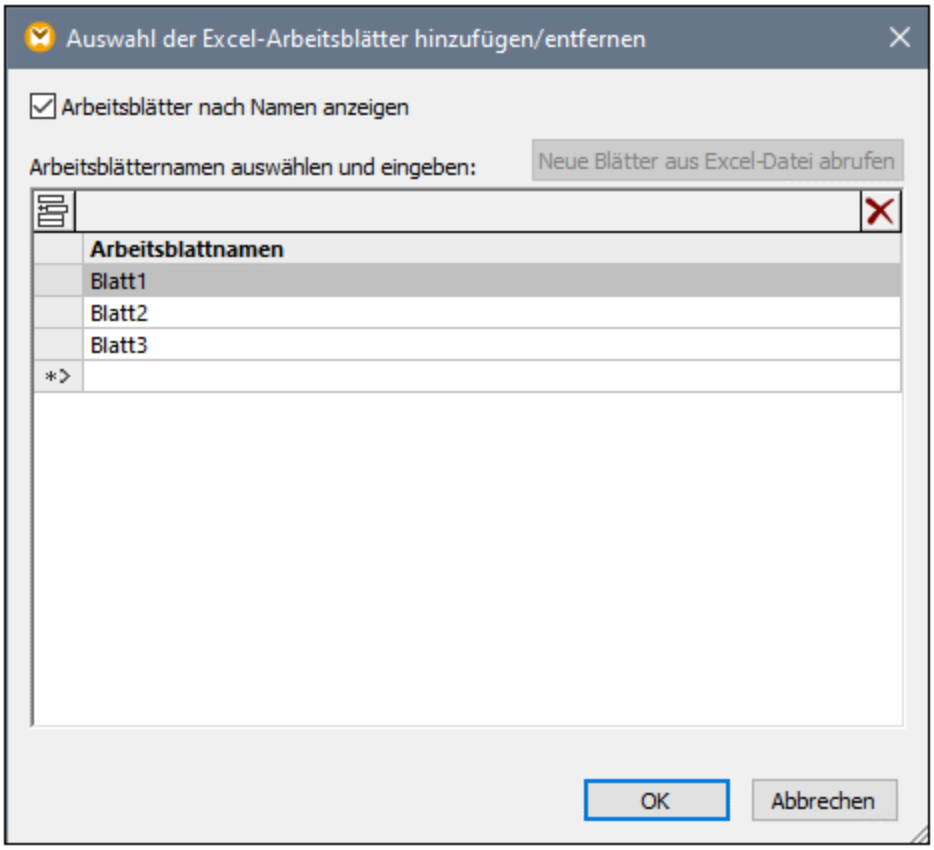
Sie können mit MapForce ein Mapping so konfigurieren, dass es Daten in jedem oder oben erwähnten Fälle flexible aus Arbeitsblättern extrahiert:

1. Sie können im Mapping jedes einzelne Excel-Arbeitsblattt als separate Struktur anzeigen. Diese Methode eignet sich, wenn die einzelnen Excel-Arbeitsblätter eine unterschiedliche Struktur aufweisen. Sie haben dadurch die Flexibilität, Daten aus jeder beliebigen Zelle jedes Arbeitsblatts in Ihrer Excel-Quelldatei zu mappen.

2. Sie können alle Excel-Arbeitsblätter in der Datei so behandeln, als würde es sich dabei um ein einziges Arbeitsblatt handeln. Diese Methode eignet sich nur dann, wenn alle Arbeitsblätter dieselbe Struktur haben; sie hat den Vorteil, dass Sie im Mapping weniger Verbindungen ziehen müssen. Da die Struktur der einzelnen Arbeitsblätter ja als identisch angenommen wird, ziehen Sie, unabhängig davon, in welchem Arbeitsblatt die Spalten vorkommen, nur einmal Verbindungen von den einzelnen Spalten aus.

So konfigurieren Sie eine Excel-Komponente für eine der oben beschriebenen beiden Methoden:

1. Fügen Sie die Excel-Komponente mit dem Menübefehl **Einfügen | Excel 2007+-Datei** hinzu, siehe auch Hinzufügen von Microsoft Excel-Dateien als Mapping-Komponenten 740).
2. Klicken Sie neben dem ersten Arbeitsblatt auf die Schaltfläche . Daraufhin erscheint das folgende Dialogfeld:



3. Wählen Sie eine der folgenden Methoden:

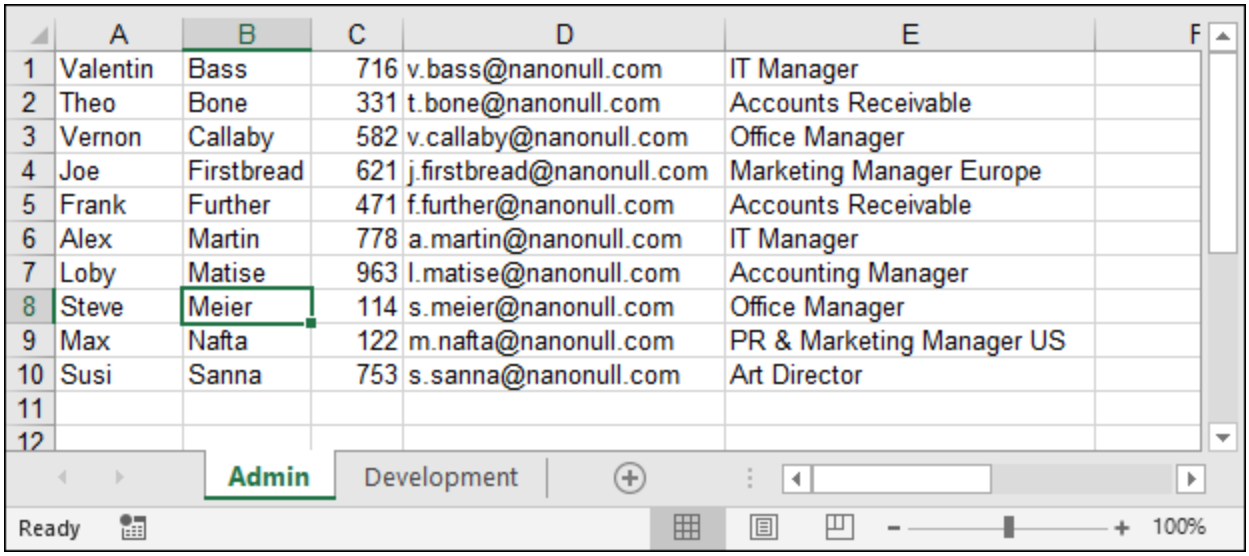
   a. Um die Struktur der einzelnen Arbeitsblätter separat anzuzeigen, stellen Sie sicher, dass das Kontrollkästchen **Arbeitsblätter nach Namen anzeigen** aktiviert ist. Dies ist die Standardoption.
   b. Um alle Arbeitblätter als eine einzige mapbare Komponente anzuzeigen, deaktivieren Sie das Kontrollkästchen **Arbeitsblätter nach Namen anzeigen**.

Unter dem folgenden Pfad finden Sie eine Demo-Mapping-Design-Datei zur Veranschaulichung der beiden Methoden: **<Dokumente>\Altova\MapForce2023\MapForceExamples\ConvertExcelRows.mfd**. In diesem Mapping werden zwei verschiedene Strategien, Daten aus einer Excel-Datei auszulesen, gezeigt. Da mit diesem Mapping mehrere nicht miteinander in Zusammenhang stehende Ausgaben erzeugt werden, wird in der

 © 2016-2022 Altova GmbH

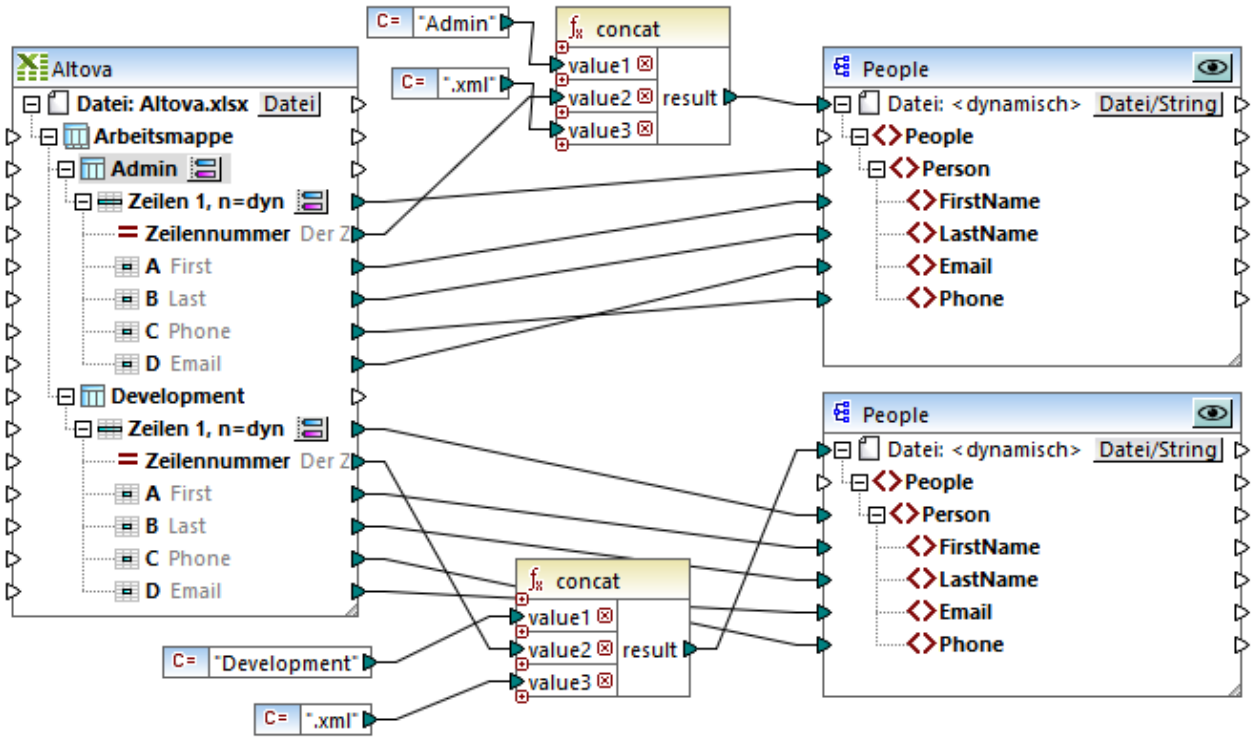
Titelleiste jeder Zielkomponente die Schaltfläche **Vorschau** angezeigt. Denken Sie daran, zuerst auf diese Schaltfläche zu klicken, bevor Sie auf das Register **Ausgabe** klicken, um eine Vorschau auf die entsprechende Ausgabedatei zu sehen.

Die Excel-Quelldatei hat den Namen **Altova.xlsx** und besteht aus zwei Arbeitsblättern: "Admin" und "Development". Beachten Sie, dass die Struktur der einzelnen Arbeitsblätter identisch ist. Nur die Daten in den Arbeitsblättern unterscheiden sich (aus diesem Grund kann der Aufruf auf verschiedene Arten erfolgen).



	A	B	C	D	E	F
1	Valentin	Bass	716	v.bass@nanonull.com	IT Manager	
2	Theo	Bone	331	t.bone@nanonull.com	Accounts Receivable	
3	Vernon	Callaby	582	v.callaby@nanonull.com	Office Manager	
4	Joe	Firstbread	621	j.firstbread@nanonull.com	Marketing Manager Europe	
5	Frank	Further	471	f.further@nanonull.com	Accounts Receivable	
6	Alex	Martin	778	a.martin@nanonull.com	IT Manager	
7	Loby	Matise	963	l.matise@nanonull.com	Accounting Manager	
8	Steve	Meier	114	s.meier@nanonull.com	Office Manager	
9	Max	Nafta	122	m.nafta@nanonull.com	PR & Marketing Manager US	
10	Susi	Sanna	753	s.sanna@nanonull.com	Art Director	
11						
12						

Im oberen Teil des Mappings sehen Sie, wie Daten aus der Excel-Datei ausgelesen werden, wobei jedes Arbeitsblatt als separate Struktur behandelt wird. Wenn Sie neben dem Datenelement **Admin** auf die Schaltfläche klicken, sehen Sie dass das Kontrollkästchen **Arbeitsblätter nach Namen anzeigen** ausgewählt wurde. Dadurch werden die beiden Arbeitsblätter "Admin" und "Development" in der Quellkomponente explizit als Child-Elemente von **Arbeitsmappe** angezeigt.



ConvertExcelRows.mfd (Teil 1)

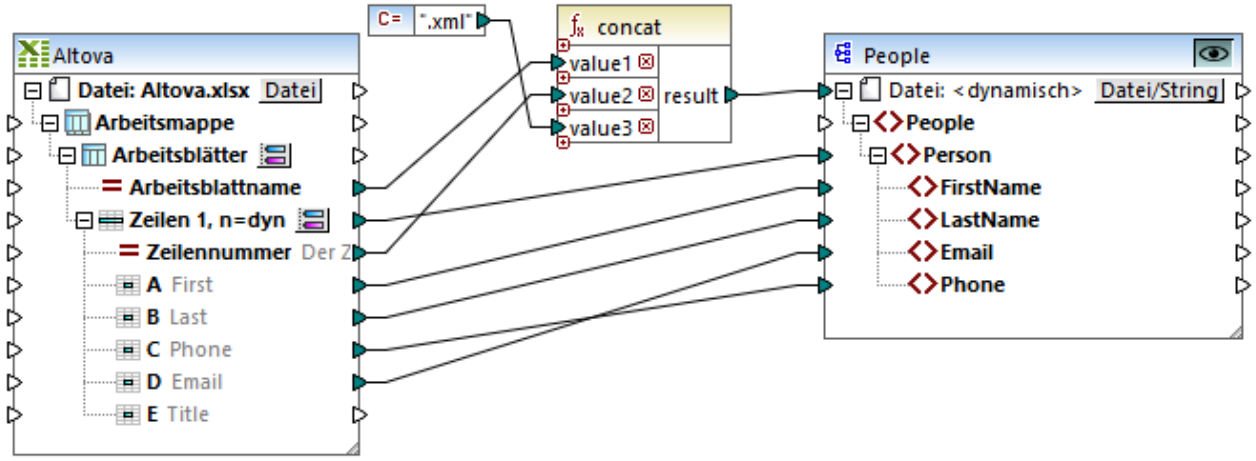
Im obigen Mapping werden Daten aus beiden Excel-Arbeitsblättern in XML-Dateien konvertiert. Für jeden Person-Datensatz aus jeder Zeile in den einzelnen Quell-Arbeitsblättern muss eine separate XML-Datei generiert werden. Der Dateiname muss den Abteilungsnamen von Department identifizieren und eine numerische ID haben. So werden z.B. in der Datei **Admin1.xml** die Informationen zur ersten Person (Zeile) im Arbeitsblatt "Admin" gespeichert, usw. Dies wurde folgendermaßen bewerkstelligt:

1. Die Funktion **concat** erzeugt die Dateinamen für die einzelnen XML-Zieldateien. Sie verbindet die folgenden Elemente miteinander (in der folgenden Reihenfolge): den Namen der Abteilung (Department), die Excel-Zeilennummer und den String ".xml".
2. Aufgrund der Verbindung vom Ergebnis der **concat**-Funktion mit dem Ziel-Datenelement **Datei:<dynamisch>** wird zur Mapping-Laufzeit für jedes Datenelement in der eingehenden Datenelementsequenz eine neue Datei generiert. Die eingehende Sequenz in diesem Beispiel enthält alle Zeilen des Excel-Quell-Arbeitsblatts; daher wird für jede Excel-Zeile eine neue XML-Datei generiert.

Um eine Zielkomponente so zu konfigurieren, dass Dateinamen dynamisch generiert werden, klicken Sie auf die Schaltfläche **Datei/String** und aktivieren Sie die Option **Über das Mapping bereitgestellte dynamische Dateinamen verwenden**. Nähere Informationen dazu finden Sie unter Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 1371.

Unten sehen Sie den zweiten Teil des Mappings **ConvertExcelRows.mfd**. Mit diesem Teil des Mappings erzielen Sie genau das gleiche Ergebnis wie mit Teil 1, doch ist dieser Teil kürzer. Beachten Sie, dass dieser Teil erheblich weniger Mapping-Verbindungen aufweist.

 © 2016-2022 Altova GmbH



ConvertExcelRows.mfd (Teil 2)

Die Mapping-Logik ist beinahe dieselbe wie die des zuvor beschriebenen Mappings:

1. Für jede Zeile in jedem Quell-Arbeitsblatt wird in der Zielkomponente ein einziges **Person**-Datenelement erstellt.
2. Die **concat**-Funktion generiert die Namen der einzelnen Zieldateien.
3. Für die Zielkomponente wurde eine dynamische Generierung von Dateinamen ausgewählt.

Der einzige Unterschied ist, dass implizit über alle Arbeitsblätter in der Excel-Quelldatei iteriert wird. Es besteht keine Notwendigkeit, die Struktur der einzelnen Arbeitsblätter separat anzuzeigen und dafür separate Mapping-Verbindungen zu ziehen. Der Grund dafür ist, dass die Excel-Quellkomponente so konfiguriert wurde, dass sie alle Arbeitsblätter als ein einziges behandelt. Wenn Sie neben dem Datenelement **Arbeitsblätter** auf die Schaltfläche klicken, sehen Sie, dass das Kontrollkästchen **Arbeitsblätter nach Namen anzeigen** nicht aktiviert ist.

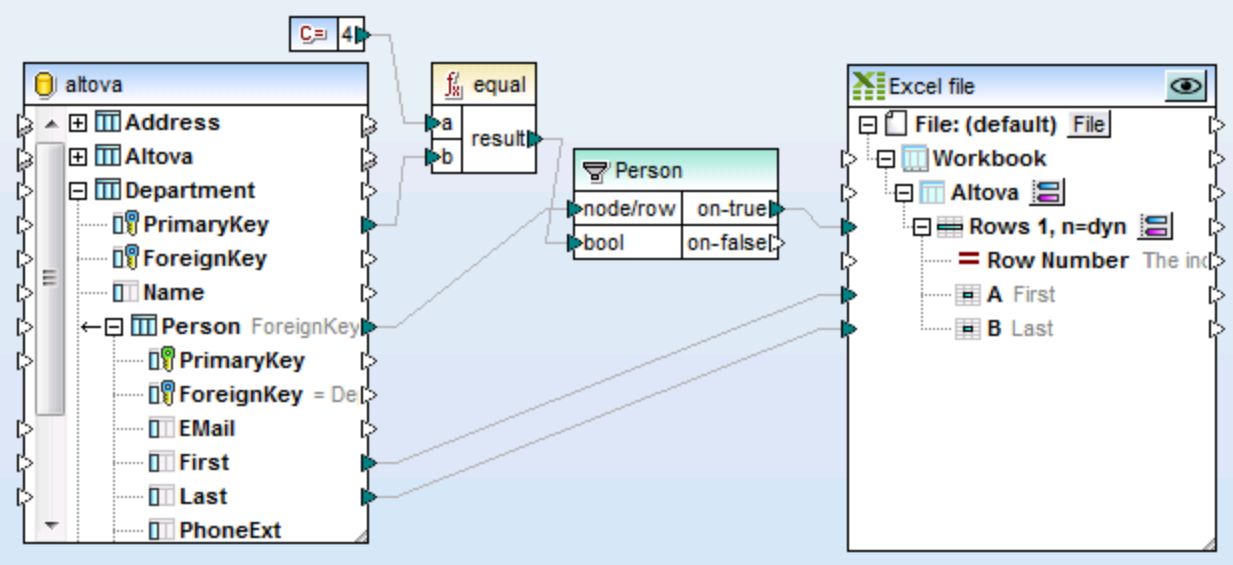
Sie haben in diesem Kapitel nun die Unterschiede zwischen dem Mapping von Daten aus einzelnen Excel-Arbeitsblättern im Gegensatz zu einem Mapping, in dem alle Arbeitsblätter als ein einziges behandelt werden, gesehen. Beachten Sie, dass die zweite Methode voraussetzt, dass die Struktur aller Arbeitsblätter dieselbe ist. In diesem Beispiel wird angenommen, dass in der ersten Spalte jedes Arbeitsblatts der Vorname der Person (First Name), in der zweiten der Nachname (Last Name) der Person, usw. gespeichert sind.

4.7.10 Beispiel: Mappen von Datenbankdaten auf Excel 2007+

Die im folgenden Beispiel verwendete Mapping-Datei steht unter dem Pfad **<Dokumente>\Altova\MapForce2023\MapForceExamples\Excel-mapping.mfd** zur Verfügung. In diesem Beispiel wird das dritte von den drei Mappings erläutert.

Ziel des Mappings ist folgendes:

- Extrahieren nur derjenigen Personen aus der Datenbank "altova", deren Primärschlüssel für die Abteilung "4" ist, d.h. die der IT-Abteilung angehören.
- Schreiben der extrahierten Datensätze in eine Excel 2007+-Standardkomponente.

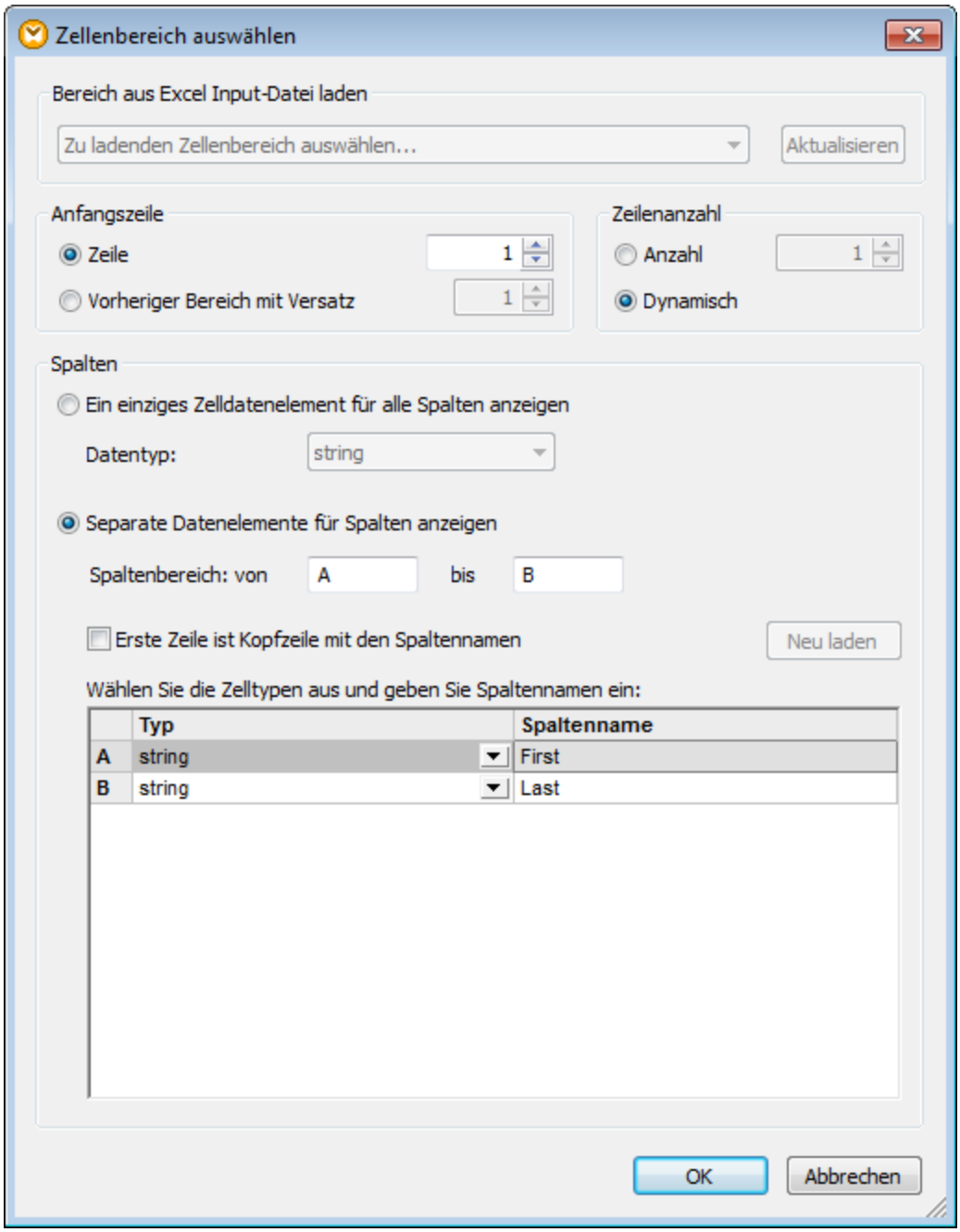


Excel-mapping.mfd (Beispiel 3)

Das Mapping ist folgendermaßen konfiguriert:

1. Aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** wurde die Datenbank "altova" mit dem Menübefehl **Einfügen | Datenbank** und über den anschließend angezeigten Assistenten zum Verbinden mit einer Microsoft Access-Datenbank zum Mapping-Bereich hinzugefügt.
2. Die Excel 2007+-Standardkomponente wurde mit dem Menübefehl **Einfügen | Excel 2007+-Datei** hinzugefügt, wobei die Option zur Angabe einer Beispieldatei übersprungen wurde.
3. Das erste Arbeitsblatt (Blatt1) wurde durch Klicken auf die daneben liegende Schaltfläche und Eingabe von "Altova" als Arbeitsblattname umbenannt.
4. Der Bereich **Zeilen 1, n=dyn** wurde durch Klicken auf die daneben liegende Schaltfläche konfiguriert. Die Optionen für den Zellenbereich wurden folgendermaßen definiert:

© 2016-2022 Altova GmbH



5. Weiters wurden die folgenden Optionen definiert:

- Der Wert von **PrimaryKey** wird mit Hilfe der `equal`-Funktion mit dem Wert 4 aus der Konstantenkomponente verglichen.
- Die **Filterkomponente** übergibt die First- und Last-Felder, bei denen die Boolesche Bedingung "true" ergibt, d.h. wenn der Primärschlüssel von Department "4" ist.
- Das Datenelement **on-true** ist mit dem Datenelement **Zeilen 1, n=dyn** in der Excel-Datei verbunden ist.

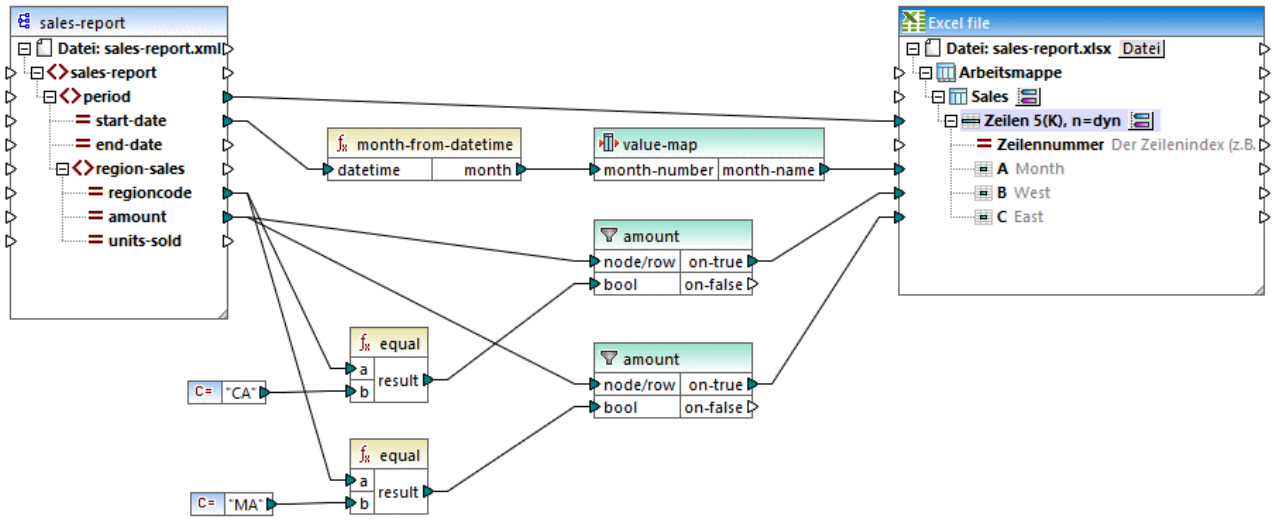
Ergebnis des Mappings: Die vier Personen der IT-Abteilung werden in der Excel-Arbeitsmappe angezeigt.

	A	B	C	D
1	Alex	Martin		
2	George	Hammer		
3	Jessica	Bander		
4	Lui	King		
5				
6				

Altova

4.7.11 Beispiel: Aktualisieren bestehender Excel-Blätter

Wenn Sie Excel-Arbeitsblätter mit MapForce *generieren*, enthalten die generierten Arbeitsblätter keine Formatierung, sondern nur Daten. Sie können jedoch auch bestehende Excel-Dateien *aktualisieren* und deren Formatierung beibehalten (anstatt die Blätter zu *generieren*). Diese Aufgabe wird in diesem Beispiel beschrieben. Sie finden das in diesem Beispiel beschriebene Mapping unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\Sales_to_Excel.mfd**. Wie Sie in der Abbildung unten sehen, werden in diesem Mapping Daten aus einer Datei im XML-Format ausgelesen und in eine vorhandene Excel-Datei (**sales-report.xslx**) geschrieben. Dabei wird ein Arbeitsblatt namens "Sales" ab der fünften Zeile in den Spalten A, B und C aktualisiert.

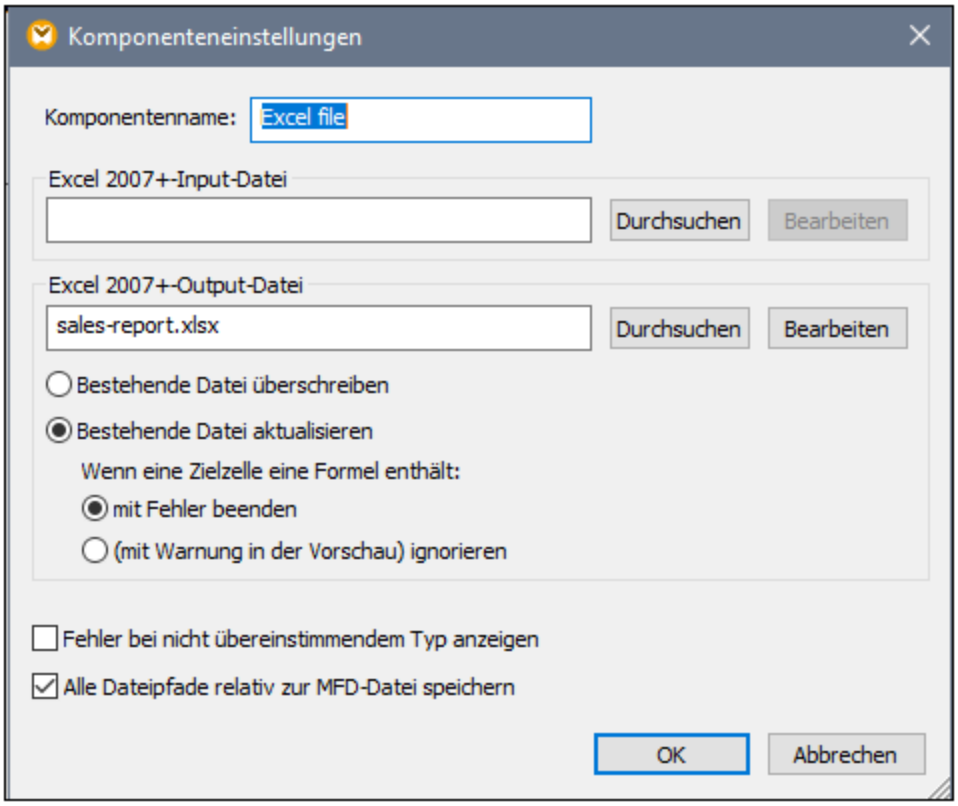


Sales_to_Excel.mfd

Die Zielkomponente wurde folgendermaßen konfiguriert, um die bestehende Datei zu aktualisieren:

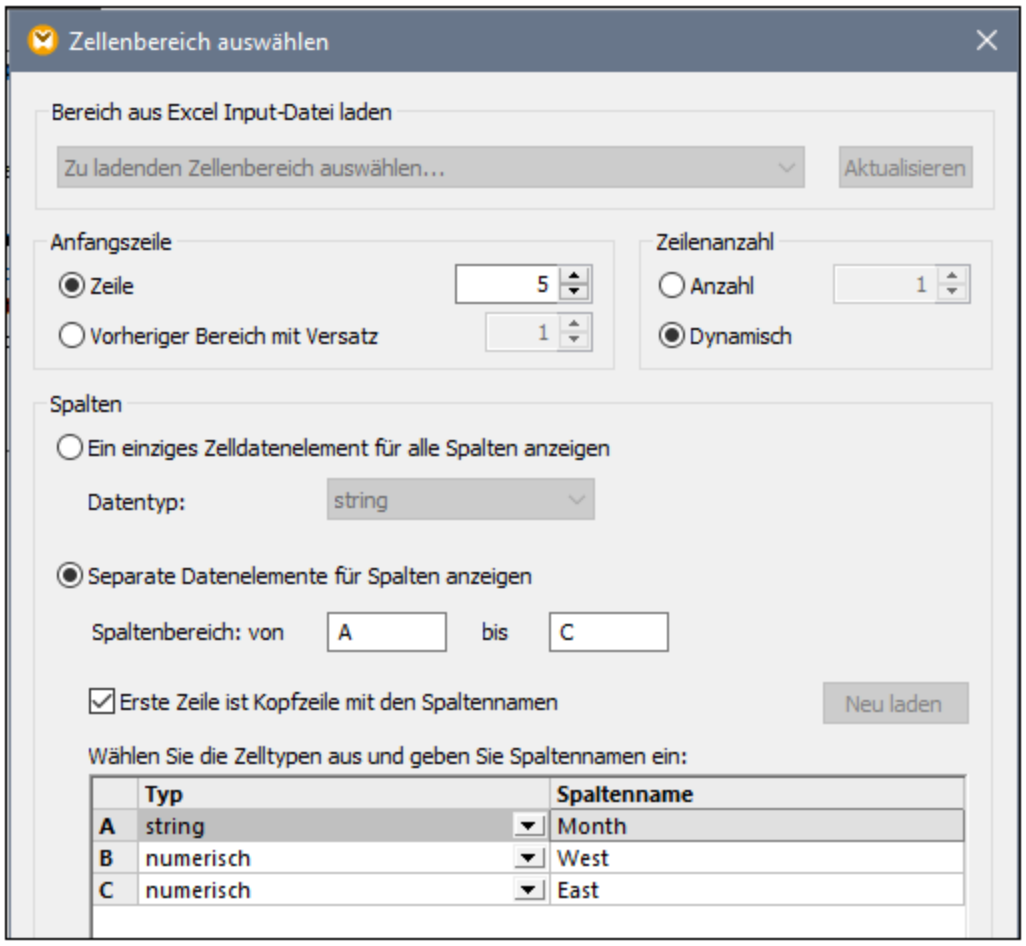
1. Definieren Sie als Transformationssprache BUILT-IN.
2. Klicken Sie mit der rechten Maustaste auf die Titelleiste der Zielkomponente und wählen Sie den Befehl **Eigenschaften**. Daraufhin wird das Dialogfeld **Komponenteneinsetllungen** aufgerufen.
3. Geben Sie im Textfeld **Excel-Output-Datei** den Pfad zur vorhandenen Excel-Datei ein. Die Datei befindet sich in diesem Beispiel im selben Ordner, daher ist der Pfad relativ und das Kontrollkästchen **Alle Dateipfade relativ zur MFD-Datei speichern** ist aktiviert.
4. Aktivieren Sie die Option **Bestehende Datei aktualisieren**.

© 2016-2022 Altova GmbH



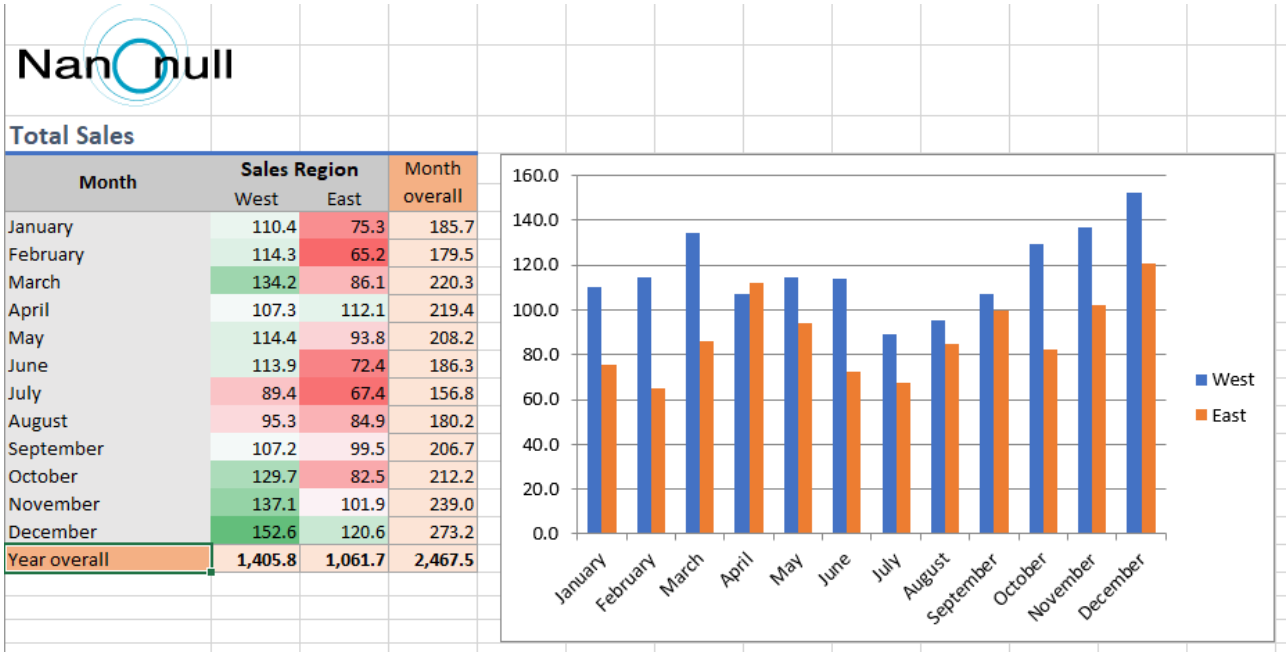
Das Ziel-Arbeitsblatt wurde durch Klick auf die Schaltfläche neben dem Node "'Sales" ausgewählt ("Sales" ist hier der Name des Arbeitsblatts).

Sie können nun den Zellen-/Zeilenbereich, in den die Daten geschrieben werden sollen, auswählen, indem Sie auf die Schaltfläche neben dem Node "Zeilen" klicken. Laut unten gezeigter Konfiguration wird das Arbeitsblatt ab Zeile 5 aktualisiert, wobei so viele Zeilen erstellt werden, wie die gemappten Quelldaten Einträge enthalten. Die Spalten A, B und C haben spezifische Namen erhalten, um das Mapping zu erleichtern.



Wenn Sie auf das Register **Ausgabe** klicken, um eine Vorschau auf das Mapping-Ergebnis zu sehen, wird das vorhandene Arbeitsblatt aktualisiert, wobei die gesamte vorhandene Formatierung beibehalten wird. Das vorhandene Diagramm wird in der Folge ebenfalls aktualisiert (da sich die Werte in den im Diagramm verwendeten Quellzellen geändert haben).

© 2016-2022 Altova GmbH



Ergebnis des Mappings

Beachten Sie, dass die Änderungen an der Excel-Datei bei der Anzeige der Vorschau auf das Mapping noch nicht gespeichert werden. Um die im **Ausgabefenster** angezeigte Datei auf der Festplatte zu speichern, wählen Sie den Menübefehl **Ausgabe | Ausgabedatei speichern** oder klicken Sie auf die Symbolleisten-Schaltfläche **Generierte Ausgabe speichern**. Alternativ dazu können Sie unter **Extras | Optionen**[1539] | **Allgemein** auch die Option **Direkt in die endgültige Output-Datei schreiben** aktivieren. Seien Sie vorsichtig, wenn Sie diese Option verwenden, da etwaige vorhandene Dateien dadurch ohne vorherige Bestätigung überschrieben werden.

Um die Ausführung von Mappings wie diesem zu automatisieren, sodass der Benutzer auf keine Schaltflächen mehr klicken muss, können Sie das Mapping auch auf MapForce Server bereitstellen, wie unter Automatisieren von Mappings und MapForce[1432] beschrieben.

4.8 XBRL

XBRL (Extensible Business Reporting Language) ist ein offener internationaler Standard zum Austausch von Geschäftsdaten. XBRL wird in erster Linie von staatlichen Stellen, Finanzinstituten und Unternehmen für die Finanzberichterstattung verwendet und von XBRL International verwaltet.

Unternehmen oder Organisationen setzen XBRL oft auf unterschiedliche Art ein (z.B. XBRL-Validierung, Erstellung von XBRL-Daten, Erstellung neuer XBRL-Taxonomien, Export von XBRL-Daten in andere Formate). Für Aufgaben im Zusammenhang mit XBRL steht eine Reihe verschiedener Altova-Produkte zur Verfügung:

- *XMLSpy* dient zum Erstellen oder Bearbeiten neuer Taxonomien und generiert XBRL-Berichte anhand von XSLT-Transformationsdateien (XSLT-Transformationsdateien können in StyleVision erstellt werden)
- *StyleVision* dient zum Erstellen von Taxonomie Stylesheets/Vorlagen, mit Hilfe derer Sie XBRL-Berichte erstellen können.
- *MapForce* dient zum Mappen von Daten von oder auf XBRL-Instanzdateien. Sie können XBRL-Daten von oder auf andere von MapForce unterstützte Formate (z.B. Microsoft Excel 2007-Dateien (oder höher), Datenbanken oder CSV-Dateien) mappen. So können Sie z.B. bestimmte Daten aus vorhandenen XBRL-Instanzdokumenten filtern oder XBRL-Instanzen anhand von aus einer Datenbank extrahierten Daten erstellen.
- *RaptorXML+XBRL Server* ist ein plattformübergreifender XBRL-Prozessor, der XBRL-Instanzen und -Taxonomien validiert.
- *Altova XBRL Add-ins für Excel*: Damit können Unternehmen XBRL-Berichte mit Hilfe von Microsoft Excel vorbereiten und validieren.

Eine weitere Einführung in XBRL würde den Rahmen dieser Dokumentation sprengen. Die folgenden Begriff werden im Zusammenhang mit XBRL häufig verwendet:

- *XBRL-Taxonomie*: Eine Taxonomie ist wie ein Wörterbuch aller Concepts, die in einem XBRL-Bericht vorkommen können. Außerdem sind darin Regeln zur hierarchischen Strukturierung dieser Concepts definiert.
- *XBRL-Instanz*: Die XBRL-Instanz ist ein Dokument, das die sogenannten "Facts", über die ein Bericht erstellt wird, enthält. Die eigentliche Definition von Concepts und hierarchischen Beziehungen zwischen diesen erfolgt in der Taxonomie.

Um in MapForce Daten von oder auf XBRL-Instanzdateien mappen zu können, wird die zugrunde gelegte XBRL-Taxonomie benötigt. Die Taxonomie besteht aus einem (normalerweise gezippten) Dateiarchiv, die von der Organisation oder der Stelle, bei der die XBRL-Daten eingereicht werden, zur Verfügung gestellt wird. Die folgenden Taxonomien sind Beispiele für Taxonomien, die für Datenmappings mit MapForce verwendet werden können:

- US-GAAP (Generally Accepted Accounting Principles): wird vom Financial Accounting Standards Board (http://www.fasb.org) herausgegeben.
- IFRS (International Financial Reporting Standards): wird von der IFRS (International Financial Reporting Standards) Foundation (http://www.ifrs.org) herausgegeben.
- COREP / FINREP (Common Reporting und Financial Reporting Taxonomies, veröffentlicht von der European Banking Authority): wird von der European Banking Authority (http://www.eba.europa.eu) herausgegeben.

Sie können XBRL-Taxonomien in allen XBRL-fähigen Altova-Produkten mit Hilfe des XBRL-Taxonomie-Managers 777 installieren bzw. deinstallieren.

© 2016-2022 Altova GmbH

Um URIs in XBRL-Dokumenten aufzulösen, wird in MapForce ein Katalogmechanismus verwendet, bei dem URIs auf Dateien auf dem lokalen System gemappt werden - größtenteils aufgrund der Größe der Taxonomien und der Tatsache, dass diese Tausende von Dateien enthalten. Ein Aufruf der Taxonomiedateien über das Internet würde das System extrem verlangsamen, selbst wenn die Herausgeber der Taxonomien diese auf diese Art zur Verfügung stellen würden. Beachten Sie, dass für diesen Katalogmechanismus XBRL-Taxonomiepakete lokal auf Ihrem Rechner installiert sein müssen. Standardmäßig sind keine XBRL-Taxonomiepakete installiert. Wenn Sie daher ein Mapping, das XBRL-Komponenten enthält, öffnen, wird in MapForce eventuell ein Dialogfeld angezeigt, in dem Sie aufgefordert werden, die erforderlichen Taxonomiepakete zu installieren. Zusätzlich dazu können Sie alle installierten XBRL-Taxonomien immer über den XBRL-Taxonomie-Manager 777 anzeigen und verwalten.

Unterstützte XBRL-Standards

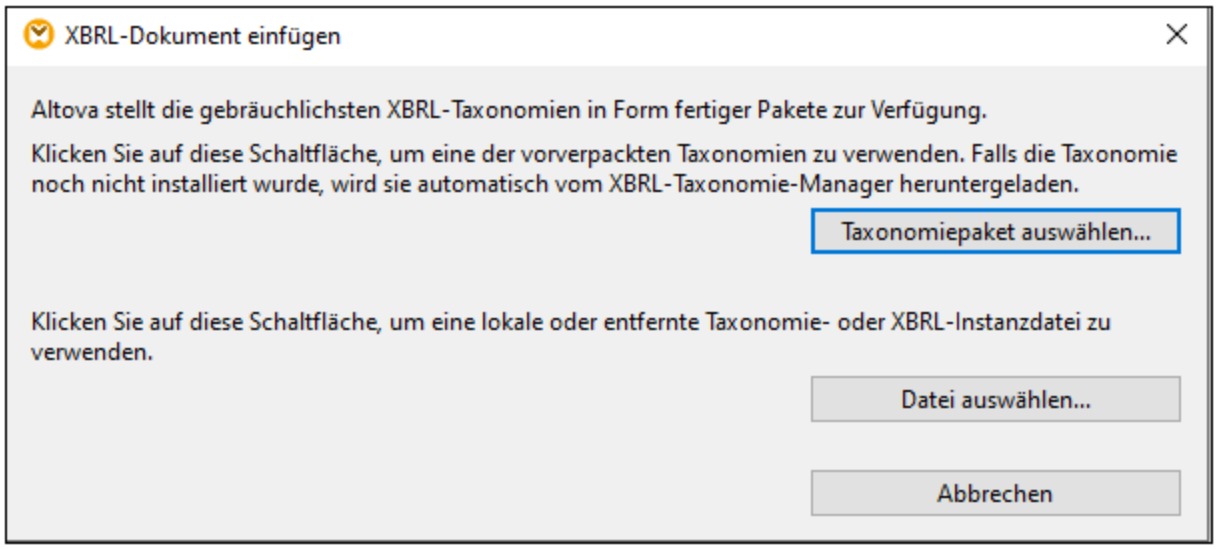
MapForce unterstützt im Zusammenhang mit XBRL die folgenden Standards:

- XBRL Specification 2.1
- XBRL Dimensions Specification 1.0
- Table Linkbase Spezifikation 1.0 (siehe Arbeiten mit XBRL-Tabellen 810)
- Taxonomy Packages 1.0 (siehe XBRL-Taxonomiepakete 1545)
- Inline XBRL 1.0 und 1.1, einschließlich Transformation Registry 3 (nur zum Lesen von Daten). Siehe auch Lesen von Daten aus Inline XBRL 775.
- XII Transformation Registry 5-Spezifikation

4.8.1 Hinzufügen von XBRL-Dateien

In diesem Kapitel wird erklärt, wie Sie ein XBRL-Dokument zu einem Mapping hinzufügen. Mit dem Menübefehl **Einfügen | XBRL-Dokument** oder über die Symbolleisten-Schaltfläche  können Sie ein XBRL-Dokument einfügen. Im Dialogfeld (*siehe Abbildung unten*) werden Sie gefragt, ob Sie eine vorverpackte Taxonomie oder eine lokales bzw. entfernte Datei einfügen möchten.

Wenn Sie sich für eine vorverpackte Taxonomie entscheiden, werden Sie aufgefordert, einen Eintrittspunkt auszuwählen. Wenn die gewünschte Taxonomie noch nicht installiert ist, wird sie automatisch vom XBRL-Taxonomie-Manager 777 heruntergeladen. Wenn Sie eine lokale oder entfernte Datei importieren möchten, können Sie eine Taxonomiedatei oder eine XBRL-Instanzdatei auswählen.



Informationen dazu, welche Elemente in einer XBRL-Komponente vorkommen können und wie diese grafisch dargestellt werden, finden Sie unter XBRL-Komponentenelemente 772.

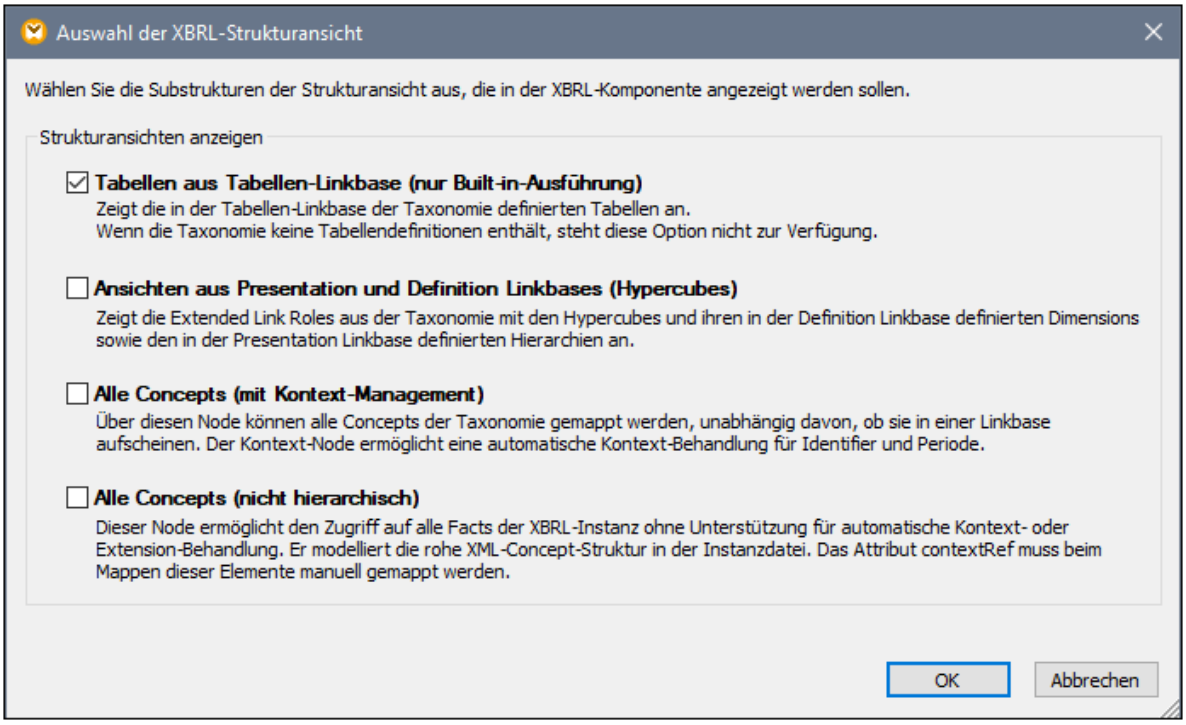
4.8.2 Strukturansichten

MapForce bietet mit Hilfe von so genannten "Strukturansichten" verschiedene Möglichkeiten, um die XBRL-Struktur innerhalb einer Komponente anzuzeigen. Sie können entweder beim Hinzufügen eines XBRL-Dokuments zum Mapping-Bereich (siehe Hinzufügen von XBRL-Dateien als Mapping-Komponenten 769) oder zu einem späteren Zeitpunkt auswählen, welche XBRL-Strukturansichten in der Komponente angezeigt werden sollen. Es stehen die folgenden Strukturansichten zur Verfügung.

So wählen Sie die Strukturansichten aus, die in der XBRL-Komponente angezeigt werden sollen

1. Klicken Sie auf dem Root-Element der XBRL-Komponente auf die Schaltfläche **Kontextmenü anzeigen** ( ) und wählen Sie den Menübefehl **Strukturansichten auswählen**. Daraufhin wird das folgende Dialogfeld geöffnet.

© 2016-2022 Altova GmbH



2. Wählen Sie eine oder mehrere Strukturansichten aus und klicken Sie auf **OK**.

Strukturansicht	**Beschreibung**
Tabellen aus Tabellen-Linkbase	Zeigt die in der Tabellen-Linkbase der Taxonomie definierten Tabellen an. Diese Option ist deaktiviert, wenn die Taxonomie keine Tabellendefinitionen enthält. Diese Ansicht ist bedingt. Um fortzufahren, aktivieren Sie entweder dieses Kontrollkästchen oder das Kontrollkästchen **Ansichten aus Presentation und Definition Linkbases**.
Ansichten aus Presentation und Definition Linkbases	Zeigt Daten aus der Presentation und Definition Linkbase an. Die Presentation Linkbase enthält Hierarchien und die Definition Linkbase enthält Extended Link Roles mit Hypercubes und Dimensions. Diese Ansicht ist bedingt. Um fortzufahren, aktivieren Sie entweder dieses Kontrollkästchen oder das Kontrollkästchen **Tabellen aus Tabellen-Linkbase**.
Alle Concepts (mit Kontext-Management)	Zeigt in der hierarchischen Sturktur den Zusatz-Node "Alle Concepts" an. Dieser Node enthält den Hypercube "Dimensionless", mit Hilfe dessen alle Concepts der Taxonomie gemappt werden können, unabhängig davon, ob diese in Hypercubes über die zwei Standard-Dimensions **Identifier** und **Zeitraum** aufgelistet werden. Der **Kontext-**Node, der alle XBRL-Concepts der Taxonomie als Children enthält, dient zur automatischen Kontext-Behandlung. Abstract Items werden nicht angezeigt.

	Die Anzeige dieser Ansicht ist optional.
Alle Concepts (nicht hierarchisch)	Zeigt den zusätzlichen Node "Alle Concepts (nicht hierarchisch)" an, über den Sie Zugriff auf alle Facts der XBRL-Instanz haben, und zwar ohne Unterstützung für automatische Kontext- oder Dimension-Behandlung. Er enthält die nicht hierarchische XML-Struktur der Concepts in der Instanzdatei, Das `contextRef` Attribut muss beim Mappen dieser Datenelemente manuell gemappt werden. Die Anzeige dieser Ansicht ist optional.

Anmerkungen

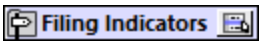
- Beim Auslesen von Daten aus einer XBRL-Komponente (d.h., wenn eine XBRL-Komponente die Quellkomponente ist) können Sie über das Kontextmenü jede beliebige Kombination von Strukturansichten auswählen. Wenn jedoch beim Schreiben in eine XBRL-Komponente (d.h. wenn die XBRL-Komponente die Zielkomponente ist) die **Tabellen aus Tabellen-Linkbase**-Ansicht ausgewählt ist, können aus technischen Gründen Daten nur in diese Ansicht und in keine andere geschrieben werden.
- Für die **Tabellen aus Tabellen-Linkbase**-Ansicht muss BUILT-IN[29] als Transformationssprache ausgewählt sein.

4.8.3 XBRL-Komponentenelemente

Die folgende Tabelle enthält einen Überblick über die Komponentenelemente (Nodes), die in MapForce normalerweise Teil einer XBRL-Komponente bilden. Beachten Sie, dass einige der Nodes nur verfügbar sind, wenn Sie die entsprechende Strukturansicht ausgewählt haben (siehe Auswählen der Strukturansichten[770]). So sind z.B. die für Tabellen-Linkbases spezifischen Nodes nicht sichtbar, wenn Sie für die Anzeige nur XBRL-Daten aus der Presentation und Definition Linkbase ausgewählt haben.

Komponentenelement	Symbol	Beschreibung
Abstract Item		Abstract Items dienen zum Strukturieren miteinander in Beziehung stehender Facts. Sie sind entweder innerhalb der Presentation Linkbase oder innerhalb von Domain Member-Netzwerken von Definitions Linkbases definiert. Sie sind nicht tatsächlich in der XBRL-Instanzdatei vorhanden. Mit Hilfe dieser Abstract Items können Facts auf intuitive Art gruppiert werden.
Breakdown		Siehe Definition in der Table Linkbase Specification 1.0, §5.4.
Context		Der Context Node ist ein Behälter für alle damit in Zusammenhang stehenden Business Facts. Context Nodes innerhalb von Hypercubes verwalten ihre Context-Elemente und Dimensions automatisch. Der `xbrli:context`

© 2016-2022 Altova GmbH

Komponentenelement	Symbol	Beschreibung
		Node, welcher ein Child des XBRL-Root-Elements ist, wird für die manuelle Behandlung von Dimensions verwendet.
Dimension		Dimensions (XBRL Dimensions Specification 1.0, §2.5) dienen zum Gliedern von Context-Informationen für Business Facts. Dimensions sind in der Taxonomie innerhalb eines Hypercube definiert.
Explicit member		Ein Explicit Member ist ein Mitglied einer Dimension, das durch eine Enumeration von `QName`-Werten definiert ist (siehe auch Anzeigen von Dimensions in einer Komponente 803).
Explicit Member-Wert		Der Wert eines Explicit Member.
Fact		Facts (XBRL Specification 2.1, §1.4) sind die Werte von XBRL-Items (Elementen). Sie können folgenden Typ haben: • Monetary Items () • String Items () • Numeric Items () • General Items () • Shares Items ()
Filing Indicators		Der Node `Filing Indicators` wird nur in XBRL-Zielkomponenten mit Taxonomien, die mit EBA in Zusammenhang stehen, angezeigt. Mit Hilfe der Schaltfläche **Filing Indicators** können Sie auswählen, wie Filing Indicators in der Ausgabe generiert werden sollen. Nähere Informationen dazu finden Sie unter XBRL-Tabellen 810.
Fußnote		Mit Hilfe von Footnotes (Fußnoten) können Sie Facts zusätzliche Textinformationen zuweisen
Hypercube		Hypercubes (XBRL Dimensions Specification 1.0, §2.2) gliedern Dimensions, Contexts und damit in Zusammenhang stehende XBRL-Concepts anhand von Informationen aus der Definition Linkbase und der Presentation Linkbase hierarchisch. Siehe auch Arbeiten mit XBLR Hypercubes 801.
Root-Element		Das Element `xbrli:xbrl` ist das Instanz-Root-Element jeder XBRL-Komponente.

Komponentenelement	Symbol	Beschreibung
Regel-Node		Siehe Definition "Rule Node" in der Table Linkbase Specification 1.0, §6.6.
Struktur-Node		Siehe Definition "Structural Node" in der Table Linkbase Specification 1.0, §5.5.
Tabelle		Siehe Definition "Table Set" in der Table Linkbase Specification 1.0, §5.1. MapForce zeigt die Tabellenstruktur in einer Hierarchie an (siehe Arbeiten mit XBRL-Tabellen 810).
Tabellengruppe		Siehe Definition "Table Set" in der Table Linkbase Specification 1.0, §5.2.
Tuple		Tuples (XBRL Specification 2.1, §1.4) sind komplexe Elemente, die als Mitglieder Facts oder andere Tuples enthalten.
Typed Member		Typed Members sind Members, die durch ein XML-Schema-Element definiert sind (siehe https://www.xbrl.org/specification/dimensions/rec-2012-01-25/dimensions-rec-2006-09-18+corrected-errata-2012-01-25-clean.html#sec-typed-dimensions).
Unit		Das Unit-Element  enthält Units, auf die sich XBRL Items beziehen. Beim Mappen von Daten muss ein Wert für das `xbrl:unit` Element definiert sein. Z.B. ist die `UnitID` die Bezeichnung `usd` und `Measure` ist `iso4217:USD`. Siehe auch Arbeiten mit XBRL-Standardwerten 797.
Ansicht		Ansichten stellen erweiterte Link Roles aus der Definition und Presentation Linkbase einer XBRL-Taxonomie dar.

Beachten Sie, dass es sich beim Symbol  um ein allgemeines Symbol handelt, das mit den verschiedensten Node-Typen einschließlich dem XBRL-Root-Element verwendet wird.

Zusätzlich dazu können in XBRL-Komponenten je nach Kontext die folgenden grafischen Benutzeroberflächenelemente angezeigt werden.

GUI-Element	Symbol	Beschreibung
Zusätzliche Informationen		Dieses Symbol wird zu Elementen angezeigt, mit denen eine Meldung verknüpft ist (z.B. Fehler). Klicken Sie auf das Symbol, um die Zusatzinformationen im Fenster "Meldungen" anzuzeigen.

© 2016-2022 Altova GmbH

GUI-Element	Symbol	Beschreibung
Fehler		Dieses Symbol zeigt eine Fehlermeldung an.
Kontextmenü anzeigen		Dieses Symbol wird zu Elementen angezeigt, für die zusätzliche Kontextmenüoptionen zur Auswahl bereit stehen.
Unbekannt		Dieses Symbol wird zu Elementen angezeigt, die von MapForce aufgrund ungültiger Namespace-Referenzen oder, weil das referenzierte Datenelement in der Taxonomie nicht vorhanden ist, nicht aufgelöst werden konnten.

4.8.4 Lesen von Daten aus Inline XBRL

Einige öffentliche Stellen, die mit XBRL-Daten arbeiten, stellen manchmal als Alternative (oder zusätzlich) zu simplen XBRL-Dokumenten Informationen im Inline XBRL-Format zur Verfügung (bzw. akzeptieren Einreichungen in diesem Format). Als Inline XBRL oder iXBRL werden in HTML-Dokumente eingebettete maschinenlesbare XBRL-Inhalte bezeichnet. Dadurch können XBRL-Daten in einem Web Browser in einem vom Menschen lesbaren Format angezeigt werden. Eine Liste der Spezifikationen im Zusammenhang mit iXBRL finden Sie unter https://specifications.xbrl.org/spec-group-index-inline-xbrl.html.

Dateien, die Inline XBRL (wie z .htm, .html oder .xhtml-Dateien) enthalten, können in MapForce als Datenquelle zum Mapping hinzugefügt werden. MapForce kann die Daten also aus Inline XBRL-Instanzdateien auslesen, um sie auf jedes beliebige unterstützte Zielformat zu mappen. Beachten Sie, dass das Schreiben von Daten in Inline XBRL-Instanzen nicht unterstützt wird.

Voraussetzungen

- Als Mapping-Transformationssprache muss BUILT-IN[29] ausgewählt sein; andere Sprachen werden nicht unterstützt. Bei Auswahl der BUILT-IN Transformationssprache kann entweder direkt in MapForce eine Vorschau auf das Mapping angezeigt werden oder das Mapping kann zu Server-Ausführungsdateien kompiliert und von MapForce Server ausgeführt werden (siehe Kompilieren von Mappings zu MapForce Server-Ausführungsdateien[1442]).
- Bei der Taxonomie der Inline XBRL-Datei kann es sich entweder um eine URL-Adresse oder eine Referenz auf eine lokale Datei handeln. Beachten Sie in letzterem Fall die folgenden zusätzlichen Voraussetzungen, wenn das Mapping mit MapForce Server ausgeführt wird:
  - Die Taxonomiedatei muss sich im selben Verzeichnis wie die .mfx-Datei (Mapping-Ausführungsdatei) befinden.
  - Die Inline XBRL-Datei muss mit Hilfe eines relativen Pfads auf die Taxonomiedatei verweisen, z.B. `<link:schemaRef xlink:type="simple" xlink:href="taxonomy.xsd" />`.
  - Auch im Dialogfeld "Komponenteneinstellungen" (siehe XBRL-Komponenteneinstellungen[793]) muss die Taxonomiedatei über einen relativen Pfad referenziert werden (z.B. "taxonomy.xsd").
- Die Inline XBRL-Quelldatei muss eine einzige XBRL-Instanz enthalten; mehrere Instanzen werden nicht unterstützt.

Hinzufügen von Inline XBRL-Instanzdateien zum Mapping

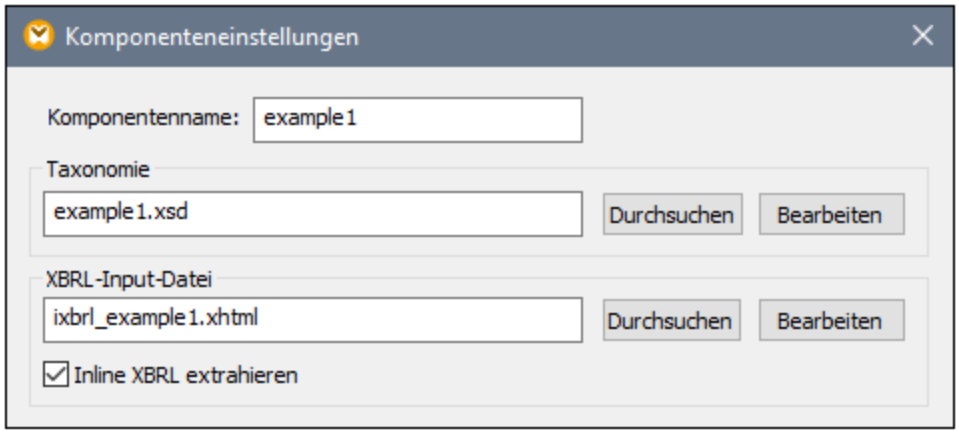
1. Klicken Sie im Menü **Einfügen** auf **XBRL-Dokument**.

1. Wählen Sie eine der folgenden Methoden:
   a. Wenn Sie die XBRL-Taxonomie zur Verfügung haben, navigieren Sie zur XBRL-Taxonomie (normalerweise einer Datei mit der Erweiterung .xsd). Es muss sich hierbei um eine gültige Taxonomie für die Inline XBRL-Instanzdatei handeln. Sie können auch eine URL als Taxonomie definieren. Klicken Sie dazu im Dialogfeld "Durchsuchen" auf die Schaltfläche **Zu URL wechseln** (siehe auch Hinzufügen von Komponenten über eine URL 80).
   b. Wenn Sie die XBRL-Taxonomiedatei oder die URL nicht zur Verfügung haben, navigieren Sie stattdessen zur Inline XBRL-Instanzdatei. MapForce versucht in diesem Fall, die dazugehörige Taxonomie automatisch zu ermitteln und zu laden.
2. Wenn Sie aufgefordert werden, eine Instanz auszuwählen, navigieren Sie zur HTML-Datei, die die Inline XBRL-Daten enthält (normalerweise eine Datei mit der Erweiterung .html, .htm oder .xhtml).
3. Wenn Sie aufgefordert werden, eine Strukturansicht auszuwählen,behalten Sie die Standardoption unverändert bei. Sie können die Strukturansicht(en) später jederzeit ändern (siehe Auswählen von Strukturansichten 770).
4. Klicken Sie auf OK.

Die Struktur des Inline XBRL-Inhalts steht nun in der Komponente zur Verfügung. Falls Validierungsfehler aufgetreten sind, werden diese im Fenster "Meldungen" angezeigt. Andernfalls können Sie wie bei XBRL-Standardkomponenten Verbindungen von der XBRL-Komponente zu einer Zielstruktur wie z.B. XML, Text, einer Datenbank usw. ziehen.

Das Kontrollkästchen "Inline XBRL extrahieren"

Wenn das Kontrollkästchen **Inline XBRL extrahieren** im Dialogfeld XBRL-Komponenteneinstellungen 793 aktiviert ist, wird die als Input-Datei definierte .html- oder .xhtml-Datei als Inline XBRL-Datei behandelt. MapForce extrahiert daraufhin den Inline XBRL-Inhalt aus der Input-Datei und validiert ihn, wenn Sie das Dialogfeld schließen. Dieses Kontrollkästchen ist normalerweise automatisch ausgewählt, wenn Sie eine Inline XBRL-Instanzdatei zum Mapping hinzufügen. Um zu überprüfen, ob diese Option aktiviert ist, doppelklicken Sie im Mapping auf die Überschrift der XBRL-Komponente (oder klicken Sie alternativ dazu mit der rechten Maustaste auf die Überschrift der XBRL-Komponente und wählen Sie **Eigenschaften**).

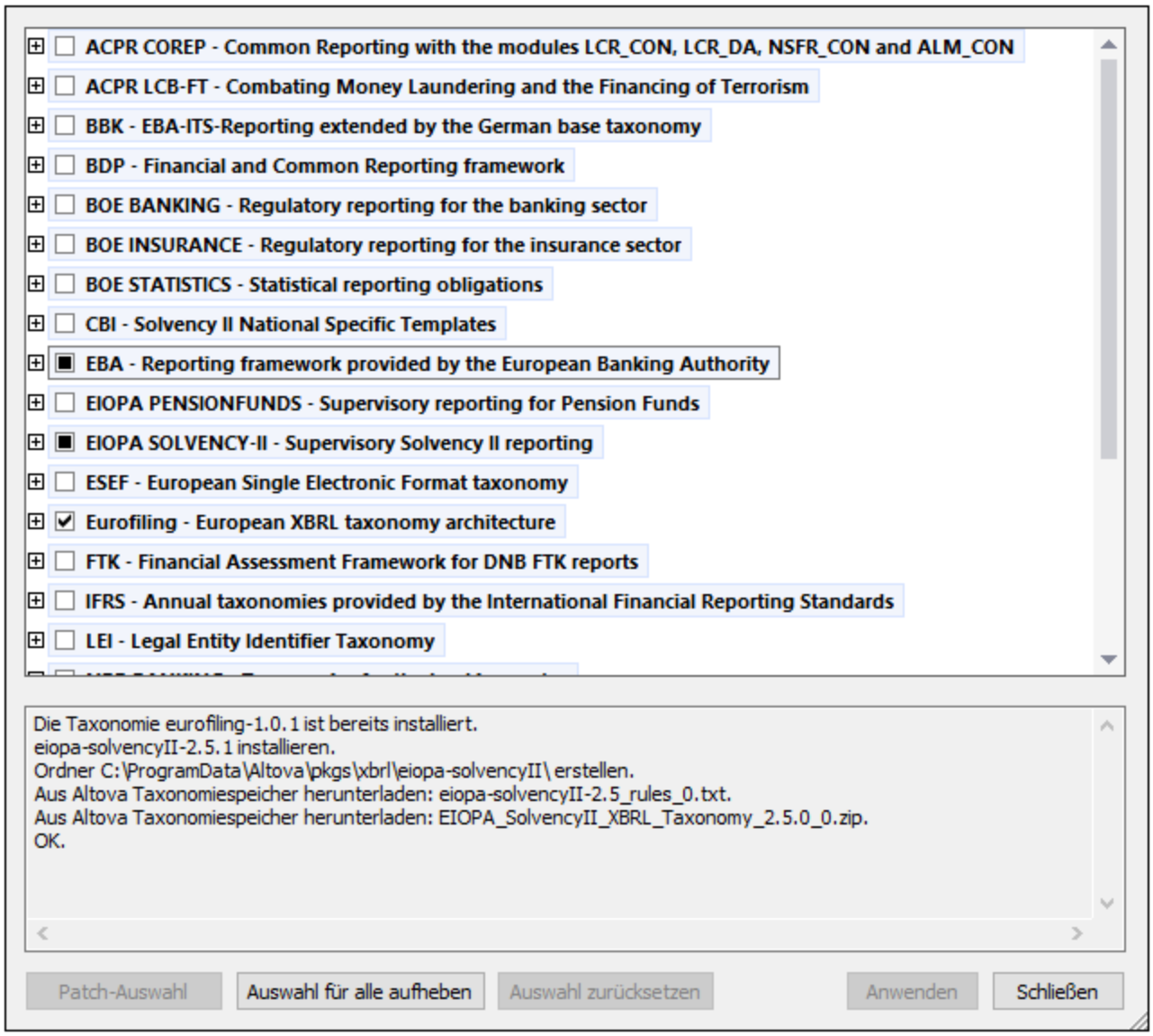


 © 2016-2022 Altova GmbH

4.8.5 Taxonomie-Manager

Der XBRL-Taxonomie-Manager ist ein Altova-Tool, mit dem Sie XBRL-Taxonomien zentral installieren und verwalten können, um diese in allen XBRL-fähigen Applikationen von Altova einschließlich MapForce verwenden zu können.

- Unter Windows hat der Taxonomie-Manager eine grafische Benutzeroberfläche (*siehe Abbildung unten*) und steht auch über die Befehlszeile zur Verfügung. (Die Desktop-Applikationen von Altova stehen nur unter Windows zur Verfügung; *siehe Liste unten*).
- Unter Linux und MacOS steht der Taxonomie-Manager nur über die Befehlszeile zur Verfügung. (Die Server-Applikationen von Altova stehen unter Windows, Linux und macOS zur Verfügung; *siehe Liste unten*).



XBRL-fähige Applikationen von Altova

Desktop-Applikationen (nur Windows)	Server-Applikationen (Windows, Linux, macOS)
Altova XBRL Add-ins für Excel (EBA, ESEF, Solvency II)	MapForce Server (Standard und Advanced Edition)
MapForce Enterprise Edition	RaptorXML+XBRL Server
StyleVision Enterprise Edition	StyleVision Server
XMLSpy Enterprise Edition	

Installation und Deinstallation des Taxonomie-Managers

Der Taxonomie-Manager wird bei der ersten Installation einer neuen Version der Altova Mission Kit Enterprise Edition oder einer der XBRL-fähigen Applikationen von Altova (*siehe Tabelle oben*) automatisch installiert.

Ebenso wird er auch automatisch entfernt, wenn Sie die letzte XBRL-fähige Applikation von Altova auf Ihrem Rechner deinstallieren.

Taxonomie-Manager-Funktionalitäten

Im Taxonomie-Manager stehen die folgenden Funktionalitäten zur Verfügung:

- Anzeigen der auf Ihrem Rechner installierten XBRL-Taxonomien und Überprüfung, ob neue Versionen zum Download zur Verfügung stehen.
- Download neuer Versionen von XBRL-Taxonomien unabhängig vom Altova Produkt-Release-Zyklus. (Die Taxonomien werden von Altova online bereitgestellt und können über den Taxonomie-Manager heruntergeladen werden).
- Installation oder Deinstallation jeder beliebigen (oder ggf. aller) der zahlreichen Versionen einer bestimmten Taxonomie.
- Eine XBRL-Taxonomie kann Abhängigkeiten von anderen Taxonomien aufweisen. Bei der Installation oder Deinstallation einer bestimmten Taxonomie informiert Sie der Taxonomie-Manager über davon abhängige Taxonomien und installiert bzw. entfernt diese ebenfalls automatisch.
- Der Taxonomie-Manager ordnet Schema-Referenzen mit Hilfe des XML-Katalogs lokalen Dateien zu. Dadurch lassen sich große XBRL-Taxonomien schneller verarbeiten, als wenn sie sich unter einem entfernten Pfad befinden.
- Alle wichtigen Taxonomien werden über den Taxonomie-Manager bereitgestellt und regelmäßig auf die jeweils neuesten Version aktualisiert. Dadurch können alle Ihre Taxonomien zentral verwaltet werden und stehen allen XBRL-fähigen Applikationen von Altova jederzeit zur Verfügung.
- Im Taxonomie-Manager vorgenommene Änderungen werden für alle auf dem Rechner installierten Altova-Produkte wirksam.

Benutzerdefinierte XBRL-Taxonomien

Wenn Sie mit benutzerdefinierten XBRL-Taxonomien, die nicht Teil des Taxonomie-Manager sind, arbeiten müssen, können Sie diese folgendermaßen zum Taxonomie-Manager hinzufügen, um diese verwalten zu können:

- *In Altova-Desktop-Applikationen:* Wählen Sie den Menübefehl **Extras | Optionen** und gehen Sie anschließend zum Abschnitt *XBRL | Taxonomiepakete*. Navigieren Sie zum ZIP-Paket Ihrer

© 2016-2022 Altova GmbH

benutzerdefinierten XBRL-Taxonomie. Nähere Informationen dazu finden Sie in Ihrer Desktop-Produktdokumentation unter der Beschreibung dieses Befehls.

- *In Altova-Server-Applikationen:* Geben Sie bei der Befehlszeilenausführung von Befehlen, die benutzerdefinierte Taxonomien unterstützen, die Option **`--taxonomy-package`** oder **`--taxonomy-package-config-file`** ein. Diese Optionen werden z.B. in RaptorXML+XBRL Server von XBRL-Validierungsbefehlen wie valxbrl oder valxbrltaxonomy oder in MapForce vom Befehl run unterstützt.

Funktionsweise

Alle in Altova-Produkten verwendeten XBRL-Taxonomien werden von Altova online bereitgestellt. Dieser Speicher wird bei Veröffentlichung neuer Versionen der Taxonomien aktualisiert. Im Taxonomie-Manager werden sowohl bei Aufruf über die Benutzeroberfläche als auch über das CLI Informationen über die neuesten verfügbare Taxonomien angezeigt. Sie können die gewünschten Taxonomien dann über den Taxonomie-Manager installieren, aktualisieren oder deinstallieren.

Taxonomien können vom Taxonomie-Manager auch auf eine weitere Art installiert werden. Sie können eine Taxonomie und die davon abhängigen Taxonomien auf der Altova Website (https://www.altova.com/de/taxonomy-manager) auswählen. Daraufhin wird auf der Website eine Datei des Typs **`.altova_taxonomies`** mit Informationen über Ihre ausgewählte Taxonomie zum Download vorbereitet. Bei Doppelklick auf diese Datei oder bei Übergabe an den Taxonomie-Manager über das CLI als Argument des Befehls **`install`**[787] installiert der Taxonomie-Manager die ausgewählten Taxonomien.

Lokaler Cache: Überprüfung Ihrer Taxonomien

Alle Informationen über installierte Taxonomien werden in einem zentralen Cache-Verzeichnis auf Ihrem Rechner aufgezeichnet. Das Verzeichnis befindet sich hier:

Windows	`C:\ProgramData\Altova\pkgs\.cache`
Linux	`/var/opt/Altova/pkgs\.cache`
macOS	`/var/Altova/pkgs`

Dieses Cache-Verzeichnis wird regelmäßig mit dem neuesten Status der Taxonomien im Online-Speicher von Altova aktualisiert. Diese Aktualisierungen finden unter den folgenden Bedingungen statt:

- bei jedem Start von Taxonomie-Manager.
- Wenn Sie MapForce zum ersten Mal an einem bestimmten Kalendertag starten.
- Wenn MapForce länger als 24 Stunden geöffnet ist, findet alle 24 Stunden eine Aktualisierung des Cache statt.
- Sie können den Cache auch durch Ausführung des **`update`**[791]-Befehls über die Befehlszeilenschnittstelle aktualisieren.

Der Taxonomie-Manager kann somit Ihre installierten Taxonomien über den Cache ständig anhand der online verfügbaren Taxonomien auf der Altova Website überprüfen.

Nehmen Sie keine manuellen Änderungen am Cache vor!

Das lokale Cache-Verzeichnis wird automatisch auf Basis der installierten oder deinstallierten Taxonomien verwaltet; es sollte nicht manuell geändert oder gelöscht werden. Falls Sie den Taxonomie-Manager je in seinen Originalzustand zurücksetzen möchten, (i) führen Sie den CLI-Befehl **`reset`** der Befehlszeilenschnittstelle und (ii) anschließend den Befehl `initialize`[787] aus. (Führen Sie alternativ dazu den Befehl `reset`[789] mit der Option **`-i`** aus).

4.8.5.1 Ausführen des Taxonomie-Managers

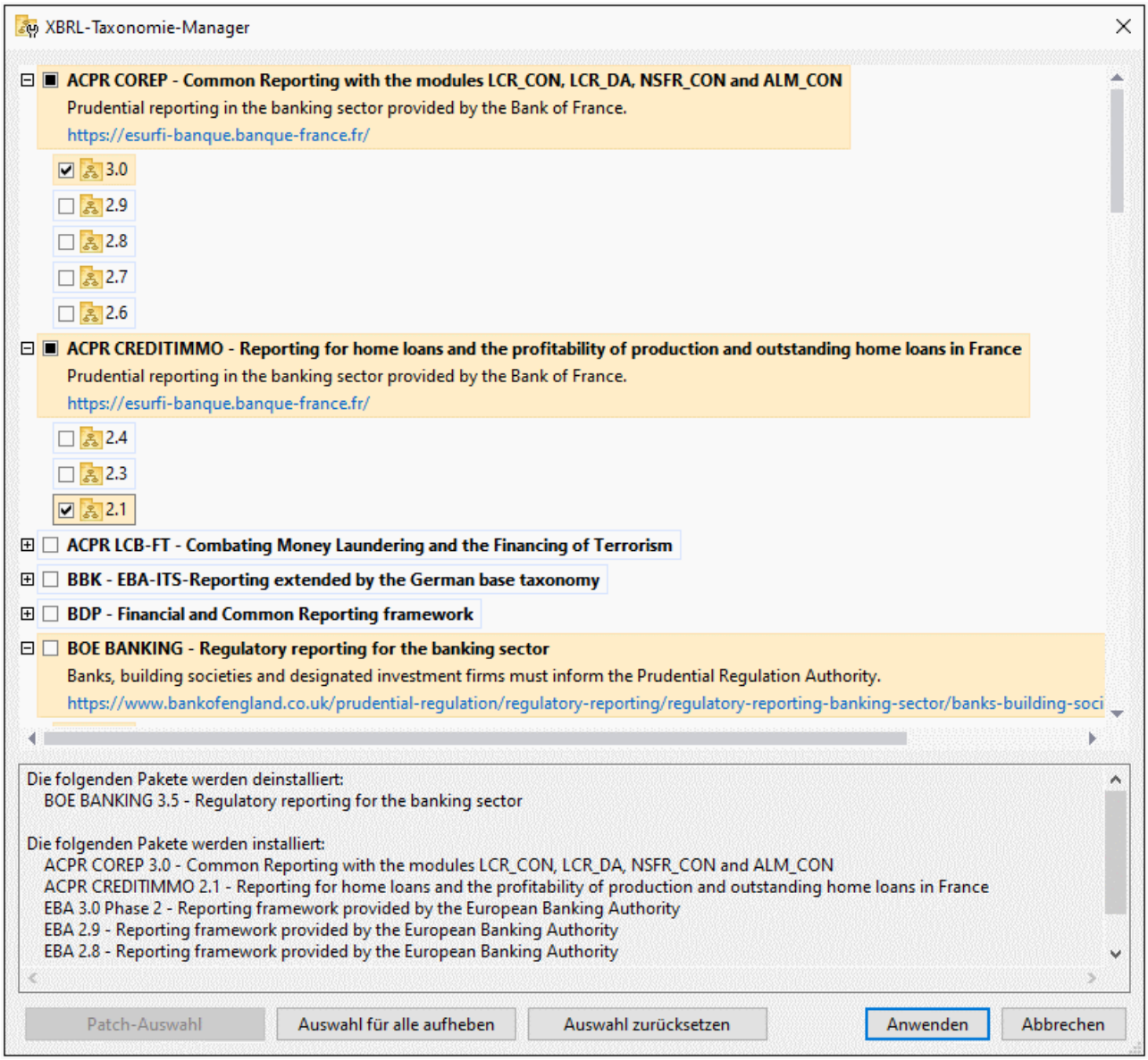
Grafische Benutzeroberfläche

Sie können die Benutzeroberfläche des Taxonomie-Managers auf eine der folgenden Arten aufrufen:

- *Bei der Installationen von MapForce:* Aktivieren Sie gegen Ende der Installation das Kontrollkästchen *Altova-Taxonomie-Manager aufrufen*, wodurch Sie die Benutzeroberfläche des XBRL-Taxonomie-Managers direkt aufrufen können. Auf diese Art können Sie Taxonomien während der Installation Ihrer Altova-Applikation installieren.
- *Nach der Installation von MapForce:* Nachdem die Applikation installiert wurde, können Sie die Benutzeroberfläche des Taxonomie-Managers jederzeit über den Menübefehl **Tools | XBRL-Taxonomie-Manager** aufrufen .
- über die vom XBRL-Taxonomie-Download-Center von Altova heruntergeladene `.altova_taxonomies`-Datei: Doppelklicken Sie auf die heruntergeladene Datei, um den Taxonomie-Manager zu starten, der daraufhin die (auf der Website) ausgewählten Taxonomien installiert.

Nachdem die Benutzeroberfläche des Taxonomie-Managers geöffnet wurde (*Abbildung unten*), werden bereits installierte Taxonomien markiert angezeigt. Wenn eine zusätzliche Taxonomie installiert werden soll, aktivieren Sie diese. Wenn eine bereits installierte Taxonomie deinstalliert werden soll, deaktivieren Sie diese. Nachdem Sie Ihre Auswahl getroffen haben, können Ihre Änderungen angewendet werden. Die Taxonomien, die installiert bzw. deinstalliert werden, werden markiert und im Fenster "Meldungen" am unteren Rand des Taxonomie-Manager-Fensters (*siehe Abbildung*) erscheint eine Meldung über die bevorstehenden Änderungen.

© 2016-2022 Altova GmbH



Befehlszeilenschnittstelle

Sie können den Taxonomie-Manager über eine Befehlszeilenschnittstelle starten, indem Sie Befehle an die ausführbare Datei **`taxonomymanager.exe`** senden.

Die Datei **`taxonomymanager.exe`** steht im folgenden Ordner zur Verfügung:

- *unter Windows:* `C:\ProgramData\Altova\SharedBetweenVersions`
- *Unter Linux oder macOS (nur Server-Applikationen):* `%INSTALLDIR%/bin`, wobei `%INSTALLDIR%` das Installationsverzeichnis des Programms ist.

Anschließend können Sie jeden der im Abschnitt zur CLI-Befehlsreferenz aufgelisteten Befehle verwenden.

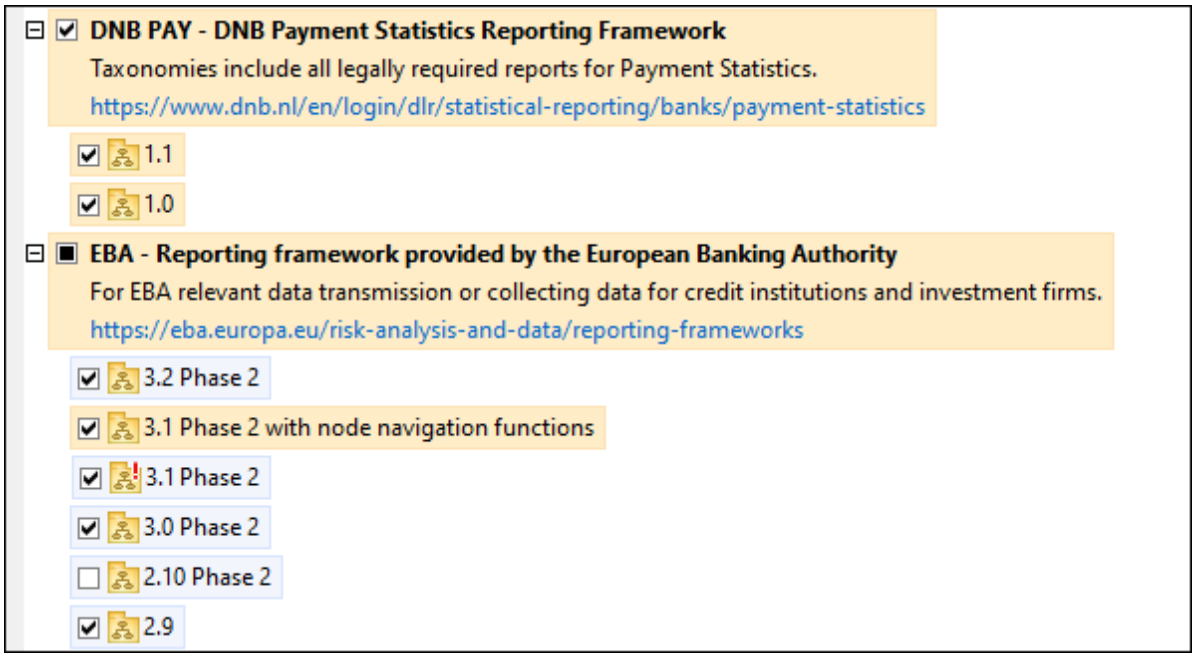
Um die Hilfe zu den Befehlen anzuzeigen, führen Sie den folgenden Befehl aus:

- *unter Windows:* `taxonomymanager.exe --help`
- *Unter Linux oder macOS (nur Server-Applikationen):* `sudo ./taxonomymanager --help`

4.8.5.2 Statuskategorien

Der Taxonomie-Manager unterscheidet folgendermaßen zwischen den von ihm verwalteten Taxonomien:

- *Installierte Taxonomien* Diese werden auf der Benutzeroberfläche mit einem Häkchen angezeigt (*in der Abbildung unten sind die mit einem Häkchen versehenen Versionen der DNB- und EBA-Taxonomie installiert*). Wenn alle Versionen einer Taxonomie ausgewählt sind, wird ein Häkchen angezeigt. Wenn zumindest eine Version nicht ausgewählt ist, wird ein gefülltes Quadrat angezeigt. Sie können die Auswahl für eine installierte Taxonomie aufheben, um sie zu **deinstallieren**.
- *Nicht installierte verfügbare Taxonomien.* Diese werden auf der Benutzeroberfläche mit einem deaktivierten Kontrollkästchen angezeigt. Sie können die Taxonomien, die **installiert** werden sollen, auswählen.



- *Taxonomien, für die ein Upgrade zur Verfügung steht* sind diejenigen, die seit ihrer Installation vom Herausgeber überarbeitet wurden. Sie werden auf der Benutzeroberfläche durch ein Symbol gekennzeichnet (*siehe Abbildung oben*). Sie können für eine installierte Taxonomie ein **Patch** der verfügbaren überarbeiteten Version installieren.

Wichtige Punkte

- In der Abbildung oben sind beide DNB-Taxonomien und einige der EBA-Taxonomien ausgewählt. Diejenigen mit einem blauen Hintergrund sind bereits installiert. Diejenige mit gelbem Hintergrund sind nicht installiert und wurden für die Installation ausgewählt. Beachten Sie, dass (i) die EBA 2.10 Phase

© 2016-2022 Altova GmbH

2-Taxonomie nicht installiert wurde und nicht für die Installation ausgewählt wurde, (ii) die EBA 3.1 Phase 2-Taxonomie installiert wurde, dafür aber seit ihrer Installation ein Patch vom Herausgeber zur Verfügung gestellt wurde, das noch nicht installiert wurde.

- Bei Ausführung des Taxonomie-Managers über die Befehlszeile wird der Befehl `list` (788) mit verschiedenen Optionen verwendet, um verschiedene Taxonomiekategorien aufzulisten:

`taxonomymanager.exe list`	Listet alle installierten und verfügbaren Taxonomien auf; auch verfügbare Upgrades werden angezeigt.
`taxonomymanager.exe list -i`	Listet nur installierte Taxonomien auf; auch verfügbare Upgrades werden angezeigt.
`taxonomymanager.exe list -u`	Listet für Taxonomien auf, für die Upgrades zur Verfügung stehen

Anmerkung: Verwenden Sie unter Linux und macOS, **`sudo ./taxonomymanager list`**

4.8.5.3 Anwenden eines Patch oder Installation einer Taxonomie

Anwenden eines Patch auf eine installierte Taxonomie

Von Zeit zu Zeit werden von den Herausgebern der XBRL-Taxonomien Patches (Upgrades oder Überarbeitungen) veröffentlicht. Wenn der Taxonomie-Manager erkennt, dass Patches zur Verfügung stehen, werden diese in der Taxonomieliste des Taxonomie-Managers angezeigt und Sie können diese Patches schnell installieren.

Über die Benutzeroberfläche

Patches werden mit dem Symbol gekennzeichnet. (*Siehe auch vorhergehendes Kapitel über Statuskategorien* (782)*).* Falls Patches zur Verfügung stehen, ist die Schaltfläche **Patch-Auswahl** aktiv. Klicken Sie darauf, um alle Patches für die Installation auszuwählen und vorzubereiten. Auf der Benutzeroberfläche ändert sich das Symbol von Taxonomien, für die ein Patch installiert wird von in , und im Fenster "Meldungen" am unteren Rand des Dialogfelds werden die Patches, die angewendet werden, aufgelistet. Sobald Sie mit der Auswahl fertig sind, klicken Sie auf **Anwenden**. Alle Patches werden gemeinsam angewendet. Beachten Sie, dass eine für die Installation eines Patch markierte Taxonomie deinstalliert wird, wenn Sie die Auswahl für diese Taxonomie aufheben.

Über das CLI

So wenden Sie einen Patch über die Befehlszeilenschnittstelle an:

1. Führen Sie den Befehl `list -u` (788) aus. Daraufhin werden alle Taxonomien, für die Patch Upgrades zur Verfügung stehen, aufgelistet.
2. Führen Sie den Befehl `upgrade` (791) aus, um alle Patches zu installieren.

Installieren einer verfügbaren Taxonomie

Sie können Taxonomien entweder über die Benutzeroberfläche des Taxonomie-Managers oder durch Senden der Taxonomie-Manager-Installationsbefehle über die Befehlszeile installieren.

Anmerkung: Wenn die aktuelle Taxonomie andere Taxonomien referenziert, werden auch die referenzierten Taxonomien installiert.

Über die Benutzeroberfläche
Um Taxonomien über die Benutzeroberfläche des Taxonomie-Managers zu installieren, wählen Sie die gewünschten Taxonomien aus und klicken Sie auf **Anwenden**.

Sie können die gewünschten Taxonomien auch auf der Altova Website auswählen und eine herunterladbare **.altova_taxonomies**-Datei generieren. Bei Doppelklick auf diese Datei wird der Taxonomie-Manager aufgerufen, in dem die gewünschten Taxonomien bereits vorausgewählt sind. Sie müssen nur mehr auf **Anwenden** klicken.

Über das CLI
Um Taxonomien über die Befehlszeile zu installieren, rufen Sie den Befehl `install` 787 auf:

```
taxonomymanager.exe install [options] Taxonomy+
```

wobei es sich bei **Taxonomy** um die gewünschte(n) Taxonomie(n) bzw. eine **.altova_taxonomies**-Datei handelt. Eine Taxonomie wird von einem Identifier im Format **<name>-<version>** referenziert. (Die Identifier von Taxonomien werden angezeigt, wenn Sie den Befehl **list** 788 ausführen.) Sie können beliebig viele Taxonomien eingeben. Nähere Informationen dazu finden Sie unter der Beschreibung des Befehls `install` 787.

Anmerkung: Verwenden Sie unter Linux oder macOS den Befehl **sudo ./taxonomymanager**.

Installation einer benötigten Taxonomie
Wenn Sie einen XBRL-Befehl in MapForce ausführen und MapForce erkennt, dass eine zur Ausführung des Befehls erforderliche Taxonomie nicht vorhanden oder unvollständig ist, wird der Taxonomie-Manager mit Informationen über die fehlende Taxonomie aufgerufen. Sie können die gewünschten Taxonomien dann über den Taxonomie-Manager direkt installieren.

Alle bereits installierten Taxonomien können jederzeit durch Aufruf des Taxonomie-Managers über **Extras | Taxonomie-Manager** über die Benutzeroberfläche des Taxonomie-Managers angezeigt werden.

4.8.5.4 Deinstallieren einer Taxonomie, Zurücksetzen

Deinstallieren einer Taxonomie

Sie können Taxonomien entweder über die Benutzeroberfläche des Taxonomie-Managers oder durch Senden der Taxonomie-Manager-Deinstallationsanweisungen über die Befehlszeile deinstallieren.

Anmerkung: Wenn die gewünschte Taxonomie andere Taxonomien referenziert, so werden auch die referenzierten Taxonomien deinstalliert.

Über die Benutzeroberfläche
Um Taxonomien über die Benutzeroberfläche des Taxonomie-Managers zu deinstallieren, deaktivieren Sie die Kontrollkästchen der entsprechenden Taxonomien und klicken Sie auf **Anwenden**. Daraufhin werden die ausgewählten Taxonomien und die davon referenzierten Taxonomien deinstalliert.

© 2016-2022 Altova GmbH

Um alle Taxonomien zu deinstallieren, klicken Sie auf **Auswahl für alle aufheben** und anschließend auf **Anwenden**.

Über das CLI
Um Taxonomien über die Befehlszeile zu deinstallieren, rufen Sie den Befehl **uninstall** auf:

```
taxonomymanager.exe uninstall [options] Taxonomy+
```

wobei es sich beim Argument **Taxonomy** eine zu deinstallierende Taxonomie oder eine **.altova_taxonomies**-Datei handelt. Eine Taxonomie wird von einem Identifier im Format **<name>-<version>** definiert. (Die Identifier von Taxonomien werden angezeigt, wenn Sie den Befehl **list** 788 ausführen.) Sie können beliebig viele Taxonomien eingeben. Nähere Informationen dazu finden Sie unter der Beschreibung des Befehls uninstall 790.

Anmerkung: Verwenden Sie unter Linux oder macOS den Befehl **sudo ./taxonomymanager**.

Zurücksetzen des Taxonomie-Managers

Sie können den Taxonomie-Manager zurücksetzen. Damit werden alle installierten Taxonomien und das Cache-Verzeichnis entfernt.

- Klicken Sie auf der Benutzeroberfläche auf **Auswahl zurücksetzen**.
- Führen Sie über die Benutzeroberfläche den Befehl reset 789 aus.

Nachdem Sie diesen Befehl ausgeführt haben, muss der Befehl initialize 787 ausgeführt werden, um das Cache-Verzeichnis neu zu erstellen. Führen Sie alternativ dazu den Befehl reset 789 mit der Option -i aus.

Beachten Sie, dass mit reset-i 789 die Originalinstallation des Produkts wiederhergestellt wird, daher wird empfohlen, nach dem Zurücksetzen auch den Befehl update 791 auszuführen. Führen Sie alternativ dazu den Befehl reset 789 mit den Optionen -i und -u aus.

4.8.5.5 Befehlszeilenschnittstelle (CLI)

Um den Taxonomie-Manager über die Befehlszeile aufzurufen, müssen Sie den Pfad zur ausführbaren Datei kennen. Standardmäßig befindet sich die ausführbare Taxonomie-Manager-Datei hier:

```
C:\ProgramData\Altova\SharedBetweenVersions\TaxonomyManager.exe
```

Anmerkung: Nachdem Sie auf Linux- und macOS-Systemen das Verzeichnis in dasjenige, das die ausführbare Datei enthält, geändert haben, können Sie die ausführbare Datei mit **sudo ./taxonomymanager** aufrufen. Das Präfix **./** gibt an, dass sich die ausführbare Datei im aktuellen Verzeichnis befindet. Das Präfix **sudo** gibt an, dass der Befehl mit Root-Rechten ausgeführt werden muss.

Befehlszeilensyntax

Die allgemeine Syntax zur Verwendung der Befehlszeile lautet folgendermaßen:

```
<exec> -h | --help | --version | <command> [options] [arguments]
```

Der senkrechte Balken **|** im Codefragment oben trennt eine Gruppe einander gegenseitig ausschließender Elemente. Optionale Elemente stehen innerhalb von eckigen Klammern **[]**. Im Prinzip können Sie den Pfad zur ausführbaren Datei, gefolgt von entweder `-h`, `--help` oder `--version`-Optionen oder gefolgt von einem Befehl eingeben. Jeder Befehl kann Optionen und Argumente haben. Die Liste der Befehle wird in den folgenden Abschnitten beschrieben.

4.8.5.5.1 help

Mit diesem Befehl erhalten Sie Hilfe zu Befehlen zur ausführbaren Taxonomie-Manager-Datei.

Syntax

```
<exec> help [Befehl]
```

`[Befehl]` ist hierbei ein optionales Argument zur Angabe jedes beliebigen gültigen Befehlsnamens.

Beachten Sie dazu Folgendes:

- Sie können die Hilfe zu einem Befehl auch durch Eingabe des Befehls, gefolgt von **-h** oder **--help** aufrufen, z.B: **<exec> list -h**
- Wenn Sie **-h** oder **--help** direkt nach dem Namen der ausführbaren Datei und vor einem Befehl eingeben, wird die allgemeine Hilfe (und nicht die Hilfe zu einem bestimmten Befehl) angezeigt, z.B: **<exec> -h list**

Beispiel

Mit dem folgenden Befehl wird Hilfe zum Befehl `list` angezeigt:

```
taxonomymanager help list
```

4.8.5.5.2 info

Mit diesem Befehl werden ausführliche Informationen über die einzelnen als `Taxonomy`-Argument angegebenen Taxonomien angezeigt. Darin enthalten sind Titel, Version, Beschreibung, Herausgeber der jeweils angegebenen Taxonomie und davon abhängige Taxonomien sowie die Information, ob die Taxonomie installiert ist oder nicht.

Syntax

```
<exec> info [options] Taxonomy+
```

- Das Argument **Taxonomy** ist der Name einer Taxonomie oder Teil eines Taxonomienamens. (Die Paket-ID einer Taxonomie und detaillierte Informationen über ihren Installationsstatus erhalten Sie mit dem Befehl `list`788.)
- Mit **<exec> info -h** können Sie die Hilfe zum Befehl anzeigen.

 © 2016-2022 Altova GmbH

Beispiel

Mit dem folgenden Befehl werden Informationen über die Taxonomien **eba-2.10** und **us-gaap-2020.0** angezeigt:

```
taxonomymanager info eba-2.1.0 us-gaap-2020.0
```

4.8.5.5.3 initialize

Mit diesem Befehl wird die Taxonomie-Manager-Umgebung initialisiert. Sie erstellen damit ein Cache-Verzeichnis, in dem Informationen über alle Taxonomien lokal gespeichert werden. Die Initialisierung erfolgt automatisch bei der ersten Installation einer XBRL-fähigen Altova-Applikation. Normalerweise muss dieser Befehl nicht ausgeführt werden. Nach Ausführung des **reset**-Befehls ist dies allerdings erforderlich.

Syntax

```
<exec> initialize | init [options]
```

Optionen

Für den Befehl **initialize** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl wird der Taxonomie-Manager initialisiert:

```
taxonomymanager initialize
```

4.8.5.5.4 install

Mit diesem Befehl installieren Sie eine oder mehrere Taxonomien.

Syntax

```
<exec> install [options] Taxonomy+
```

Um mehrere Taxonomien zu installieren, fügen Sie das Argument **Taxonomy** mehrmals hinzu.

Als **Taxonomy**-Argument kann eines der folgenden verwendet werden:

- Ein Taxonomie-Identifier (im Format **<name>-<version>**, z.B: **eba-2.10**). Um die Taxonomie-Identifier der gewünschten Taxonomien zu eruieren, führen Sie den Befehl `list` 788 aus. Sie können auch einen abgekürzten Identifier verwenden, sofern dieser eindeutig ist, z.B. **eba**. Falls Sie einen abgekürzten Identifier verwenden, wird die neueste Version dieser Taxonomie installiert.
- Der Pfad zu einer von der Altova-Website heruntergeladenen **.altova_taxonomies**-Datei. Informationen zu diesen Dateien finden Sie in *der Einführung zum Taxonomie-Manager.: Funktionsweise* 777.

Optionen
Für den Befehl **install** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl werden die neuesten **eba** (European Banking Authority) und **us-gaap** (US Generally Accepted Accounting Principles) Taxonomien installiert:

```
taxonomymanager install eba us-gaap
```

4.8.5.5.5 list

Mit diesem Befehl werden vom Taxonomie-Manager verwaltete Taxonomien aufgelistet. In der Liste wird eine der folgenden Informationen angezeigt:

- Alle verfügbaren Taxonomien
- Taxonomien, die im Namen den im Argument **Taxonomy** angegebenen String enthalten
- Nur installierte Taxonomien
- Nur Taxonomien, für die ein Upgrade installiert werden kann

Syntax

```
<exec> list | ls [options] Taxonomy?
```

Wenn kein **Taxonomy**-Argument angegeben wird, werden alle verfügbaren Taxonomien aufgelistet. Andernfalls werden die durch die angegebenen Optionen definierten Taxonomien aufgelistet (*siehe Beispiel unten*). Beachten Sie, dass Sie das Argument **Taxonomy** mehrfach angeben können.

Optionen
Für den Befehl **list** stehen die folgenden Optionen zur Verfügung:

`--installed, --i`	Auflisten nur der installierten Taxonomien. Der Standardwert ist **false**.
`--upgradeable, --u`	Auflisten nur derjenigen Taxonomien, für die Upgrades (Patches) zur Verfügung stehen. Der Standardwert ist **false**.

© 2016-2022 Altova GmbH

`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiele

- Um alle verfügbaren Taxonomien aufzulisten, führen Sie den folgenden Befehl aus: **`taxonomymanager list`**
- Um installierte Taxonomien aufzulisten, führen Sie **`taxonomymanager list -i`** aus.
- Um Taxonomien, die in ihrem Namen entweder "eba" oder "us-gaap" enthalten, aufzulisten, führen Sie **`taxonomymanager list eba us-gaap`** aus.

4.8.5.5.6 reset

Mit diesem Befehl werden alle installierten Taxonomien und das Cache-Verzeichnis entfernt. Ihre Taxonomieumgebung wird vollständig zurückgesetzt. Nachdem Sie diesen Befehl ausgeführt haben, muss der Befehl `initialize` 787 ausgeführt werden, um das Cache-Verzeichnis neu zu erstellen. Führen Sie alternativ dazu den Befehl `reset` mit der Option `-i` aus. Da mit `reset-i` die Originalinstallation des Produkts wiederhergestellt wird, wird empfohlen, nach dem Zurücksetzen und Initialisieren auch den Befehl `update` 791 auszuführen. Führen Sie alternativ dazu den Befehl reset mit den Optionen `-i` und `-u` aus.

Syntax

```
<exec> reset [Optionen]
```

Optionen
Für den Befehl **`reset`** stehen die folgenden Optionen zur Verfügung:

`--init, --i`	Initialisierung des Taxonomie-Managers nach dem Zurücksetzen. Der Standardwert ist **`false`**.
`--update, --u`	Aktualisiert die Liste der verfügbaren Taxonomien im Cache. Der Standardwert ist **`false`**.
`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **`false`**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **`false`**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiele

- Um den Taxonomie-Manager zurückzusetzen, führen Sie den folgenden Befehl aus: **`taxonomymanager reset`**
- Um den Taxonomie-Manager zurückzusetzen und ihn zu initialisieren, führen Sie **`taxonomymanager reset -i`** aus.
- Um den Taxonomie-Manager zurückzusetzen, ihn zu initialisieren und seine Taxonomieliste zu aktualisieren, führen Sie **`taxonomymanager reset -i -u`** aus.

4.8.5.5.7 uninstall

Mit diesem Befehl deinstallieren Sie eine oder mehrere Taxonomien. Standardmäßig werden auch alle Taxonomien, die von der aktuellen Taxonomie referenziert werden, deinstalliert. Um nur die aktuelle Taxonomie zu deinstallieren und die referenzierten Taxonomien beizubehalten, setzen Sie die Option `--k`.

Syntax

```
<exec> uninstall [options] Taxonomy+
```

Um mehrere Taxonomien zu deinstallieren, fügen Sie das Argument **`Taxonomy`** mehrmals hinzu.

Als **`Taxonomy`**-Argument kann eines der folgenden verwendet werden:

- Ein Taxonomie-Identifier (im Format **`<name>-<version>`**, z.B: **`eba-2.10`**). Um die Taxonomie-Identifier der installierten Taxonomien zu eruieren, führen Sie den Befehl `list -i`[788] aus. Sie können auch einen abgekürzten Taxonomienamen verwenden, sofern dieser eindeutig ist, z.B. **`eba`**. Falls Sie einen abgekürzten Namen verwenden, werden alle Taxonomien, die die Abkürzung in ihrem Namen enthalten, deinstalliert.
- Der Pfad zu einer von der Altova-Website heruntergeladenen **`.altova_taxonomies`**-Datei. Informationen zu diesen Dateien finden Sie *der Einführung zum Taxonomie-Manager.: Funktionsweise*[777].

Optionen
Für den Befehl **`uninstall`** stehen die folgenden Optionen zur Verfügung:

`--keep-references, --k`	Definieren Sie diese Option, um referenzierte Taxonomien beizubehalten. Der Standardwert ist **`false`**.
`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **`false`**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **`false`**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl werden die Taxonomien **`eba-2.10`** und **`us-gaap-2020.0`** und deren Abhängigkeiten deinstalliert:

```
taxonomymanager uninstall eba-2.10 us-gaap-2020.0
```

Mit dem folgenden Befehl wird die **`eba-2.10`**-Taxonomie, nicht aber die davon referenzierten Taxonomien deinstalliert:

```
taxonomymanager uninstall --k eba-2.10
```


© 2016-2022 Altova GmbH

4.8.5.5.8 update

Mit diesem Befehl wird die Liste der über den Online-Speicher verfügbaren Taxonomien abgefragt und das lokale Cache-Verzeichnis wird aktualisiert. Normalerweise muss dieser Befehl nur ausgeführt werden, wenn Sie `reset` 789 und `initialize` 787 ausgeführt haben.

Syntax

```
<exec> update [options]
```

Optionen
Für den Befehl **update** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl wird der lokale Cache mit der Liste der neuesten Taxonomien aktualisiert:

```
taxonomymanager update
```

4.8.5.5.9 upgrade

Mit diesem Befehl werden alle installierten Taxonomien, für die ein Upgrade installiert werden kann, auf die neueste verfügbare *Patch*-Version aktualisiert. Um herauszufinden, welche Taxonomien aktualisiert werden können, starten Sie den Befehl `list -u` 788 .

Anmerkung: Der Befehl **upgrade** würde eine veraltete Taxonomie entfernen, falls keine neuere Version zur Verfügung steht.

Syntax

```
<exec> upgrade [Optionen]
```

Optionen
Für den Befehl **upgrade** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

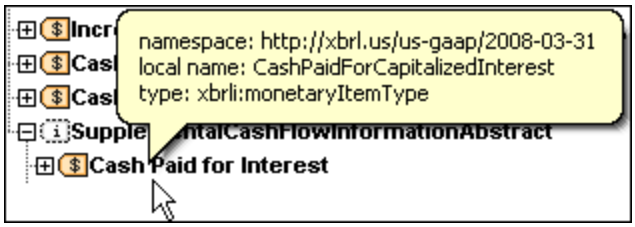
4.8.6 XBRL-Einstellungen

Sie können in MapForce die XBRL-spezifischen Einstellungen folgendermaßen konfigurieren:

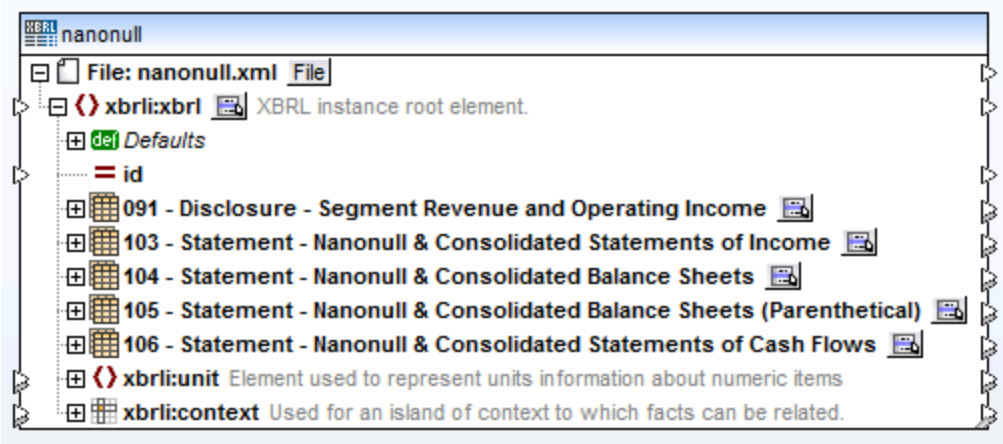
- Auf Mapping-Ebene können Sie zusätzliche Informationen zu Komponenten-Datenelementen anzeigen, siehe Aktivieren von Tipps und Annotationen 792.
- Sie können die jeweiligen Einstellungen für jede XBRL-Komponente im Mapping ändern, siehe XBRL-Komponenteneinstellungen 793.
- Sie können auch applikationsweite XBRL-Einstellungen 1543 konfigurieren.

4.8.6.1 Aktivieren von Tipps und Annotationen

Wenn Tipps aktiviert sind, können Sie zusätzliche Informationen zu den einzelnen Komponentenelementen anzeigen, wenn Sie den Mauszeiger, wie unten gezeigt, darüber platzieren.



Sie können den Annotationstext (falls vorhanden) ein- oder ausblenden. Wenn der Befehl aktiv ist, werden die Annotationen in der Komponente rechts vom Datenelement angezeigt. So lautet z.B. in der Abbildung unten die Annotation des `xbrli:xbrl` Root-Elements "XBRL instance root element".



Um Tipps zu aktivieren bzw. zu deaktivieren, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Ansicht** auf **Tipps anzeigen**.
- Klicken Sie auf die Symbolleisten-Schaltfläche **Tipps anzeigen** ( ).

© 2016-2022 Altova GmbH

Um Annotationen zu aktivieren bzw. zu deaktivieren, wählen Sie eine der folgenden Methoden:

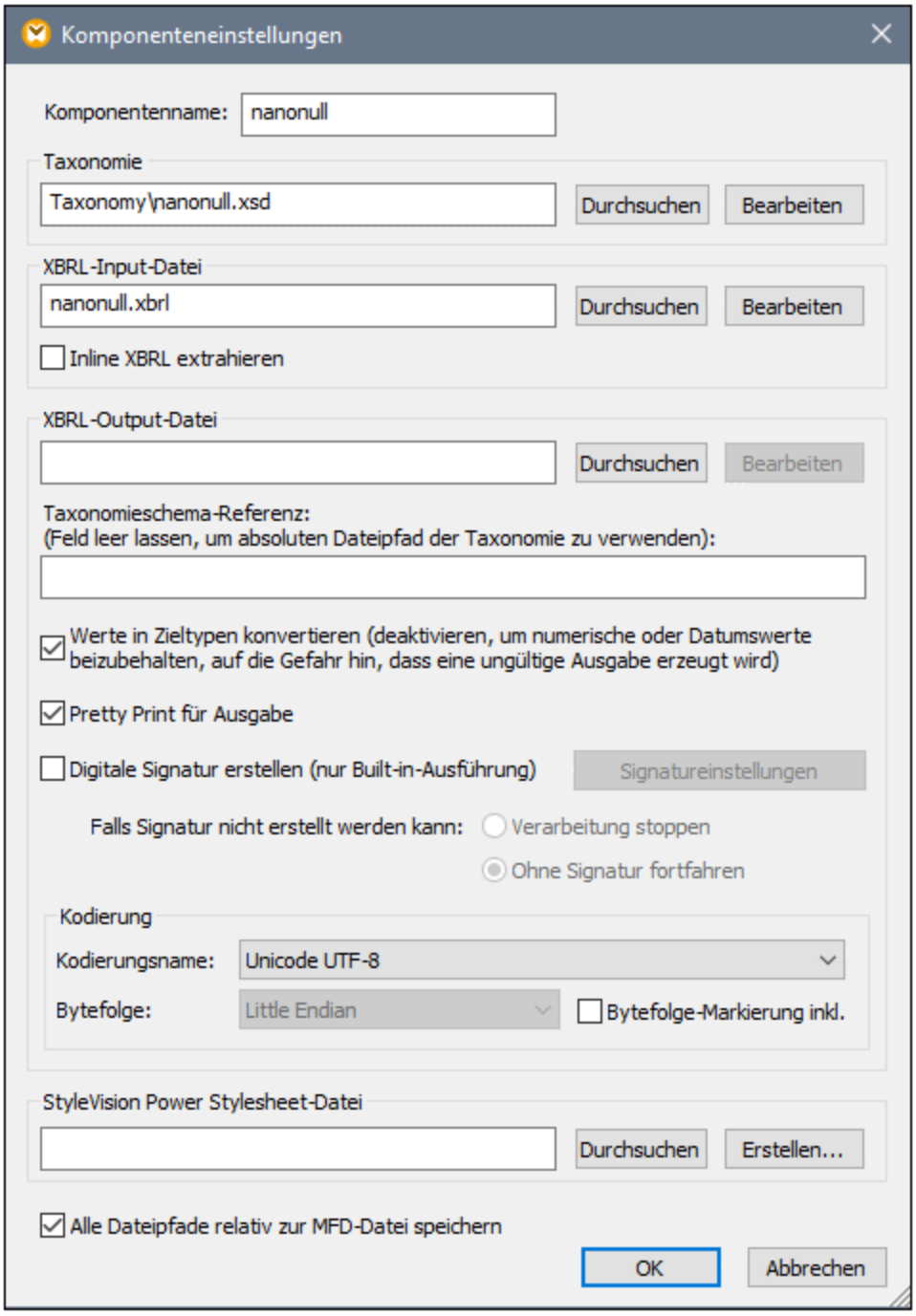
- Klicken Sie im Menü **Ansicht** auf **Annotationen anzeigen**.
- Klicken Sie auf die Symbolleisten-Schaltfläche **Annotationen anzeigen** ().

4.8.6.2 XBRL-Komponenteneinstellungen

Nachdem Sie eine XBRL-Komponente zum Mapping-Bereich hinzugefügt haben, können Sie über das Dialogfeld "Komponenteneinstellungen" die Einstellungen für die Komponente konfigurieren.

So öffnen Sie das Dialogfeld "Komponenteneinstellungen"

- Doppelklicken Sie auf die Titelleiste der XBRL-Komponente oder klicken Sie mit der rechten Maustaste darauf und wählen Sie den Befehl **Eigenschaften**.



Das Dialogfeld "Komponenteneinstellungen" enthält die folgenden Einstellungen.

Komponentenname	Der Komponentenname wird bei der Erstellung der Komponente automatisch generiert. Sie können den Namen jedoch jederzeit ändern. Der Komponentenname kann Leerzeichen und Punkte enthalten, darf aber keine Schrägstriche, umgekehrten

© 2016-2022 Altova GmbH

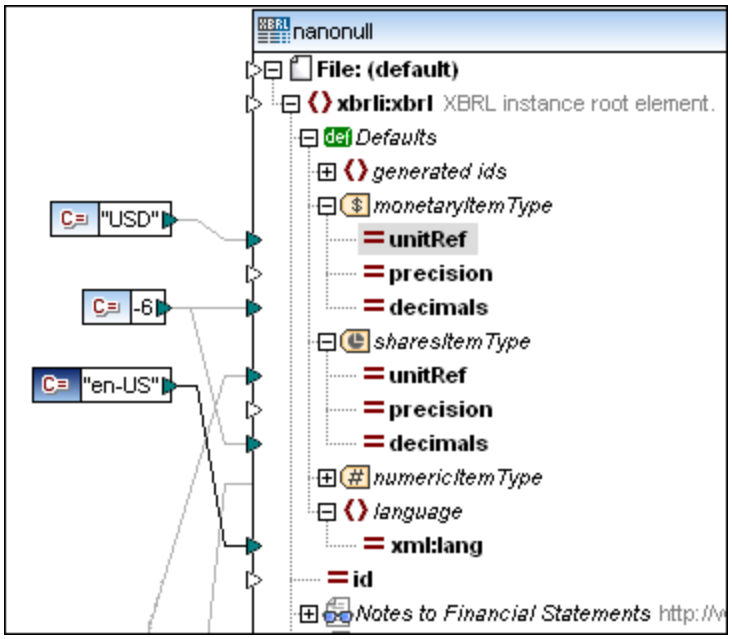
	Schrägstriche, Doppelpunkte, doppelten Anführungszeichen und voran- oder nachgestellte Leerzeichen enthalten. Beachten Sie beim Ändern des Namens einer Komponente Folgendes: • Wenn Sie das Mapping auf FlowForce Server bereitstellen möchten, muss der Komponentenname eindeutig sein. • Es wird empfohlen, nur Zeichen zu verwenden, die über die Befehlszeile eingegeben werden können. Länderspezifische Sonderzeichen sind in Windows und der Befehlszeile eventuell unterschiedlich kodiert.
Taxonomie	Zeigt den Dateinamen und Pfad der Taxonomie-Hauptdatei an. Um den Dateipfad zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie die neue Datei aus. Um die Datei in Ihrem XBRL-Editor (z.B. in XMLSpy) zu bearbeiten, klicken Sie auf **Bearbeiten**.
XBRL-Input-Datei	Definiert den Namen der XBRL-Input-Instanz für die aktuell ausgewählte XBRL-Komponente. Dieses Feld wird automatisch ausgefüllt, wenn Sie die XBRL-Komponente zum ersten Mal einfügen und ihr eine XBRL-Instanzdatei zuweisen. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie die neue Datei aus. Um die Datei in Ihrem XBRL-Editor (z.B. in XMLSpy) zu bearbeiten, klicken Sie auf **Bearbeiten**.
Inline XBRL extrahieren	Wenn dieses Kontrollkästchen aktiviert ist, wird der Inline XBRL-Inhalt aus der als Input-Datei definierten .html-, .htm- oder .xhtml-Datei extrahiert.
XBRL-Output-Datei	Dies ist der Dateiname und -pfad, in den die XBRL-Zielinstanzdatei platziert wird, wenn die Komponente als Zielkomponente des Mappings verwendet wird. Der Eintrag aus dem Feld "XBRL-Input-Datei" wird automatisch in dieses Feld kopiert, wenn Sie die XBRL-Input-Instanzdatei zuweisen. Wenn Sie der Komponente keine XBRL-Input-Instanzdatei zuweisen, enthält dieses Feld den Dateinamen und Pfad der Taxonomiedatei und die Erweiterung "xml". Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie die neue Datei aus. Um die Datei in Ihrem XBRL-Editor (z.B. in XMLSpy) zu bearbeiten, klicken Sie auf **Bearbeiten**.
Taxonomieschema-Referenz	The path of the referenced/associated taxonomy schema file relative to the MFD file. Use this field if you want to specify a

	different taxonomy location for validation. The taxonomy reference is written into the `href` attribute of the `link:schemaRef` element, for example: Der Pfad des referenzierten/verknüpften Taxonomieschemas relativ zur MFD-Datei. Verwenden Sie dieses Feld, um einen anderen Taxonomiepfad für die Validierung zu definieren. Die Taxonomiereferenz wird in das `href`-Attribut des `link:schemaRef` Elements geschrieben. z.B.: `<link:schemaRef xlink:type="simple" xlink:href="..\..\nanonull.xsd"/>`
Werte in Zieltypen konvertieren	Hier können Sie definieren, ob beim Mapping die XML-Schemazieltypen verwendet werden sollen oder ob alle auf die Zielkomponente gemappten Daten als **Stringwerte** behandelt werden sollen. Standardmäßig ist diese Option aktiviert.
Pretty Print für Ausgabe	Formatiert das XBRL-Ausgabedokument, um eine übersichtliche Struktur zu erhalten. Jeder Child Node wird durch einen einzigen Tabulator von seinem übergeordneten Node eingerückt
Digitale Signatur erstellen	Damit können Sie eine digitale Signatur zur XBRL-Outputinstanzdatei hinzufügen. Digitale Signaturen können bei Auswahl der Transformationssprache "Built-in" hinzugefügt werden (siehe auch Digitale Signaturen 143).
Kodierung	Damit können Sie die folgenden Einstellungen der Output-Instanzdatei definieren: • Kodierungsname • Bytefolge • Ob das Bytefolgemarkierungszeichen (BOM) inkludiert werden soll.
StyleVision Power Stylesheet-Datei	Mit Hilfe dieser Option können Sie eine Altova StyleVision Stylesheet-Datei auswählen oder erstellen. Damit können Sie Daten aus der XBRL-Instanzdatei in den verschiedensten Berichtsformaten wie z.B. HTML, RTF usw. ausgeben.
Alle Dateipfade relativ zur MFD-Datei speichern	Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld "Komponenteneinstellungen" angezeigten Pfade relativ zum Ordner der MapForce Design (.mfd) Datei. Diese Einstellung wirkt sich auf die folgenden Dateien aus: • die XBRL-Taxonomiedatei • die XBRL-Input-Datei • die XBRL-Output-Datei • die StyleVision Stylesheet-Datei Siehe auch Verwenden relativer Pfade in einer Komponente 85.

© 2016-2022 Altova GmbH

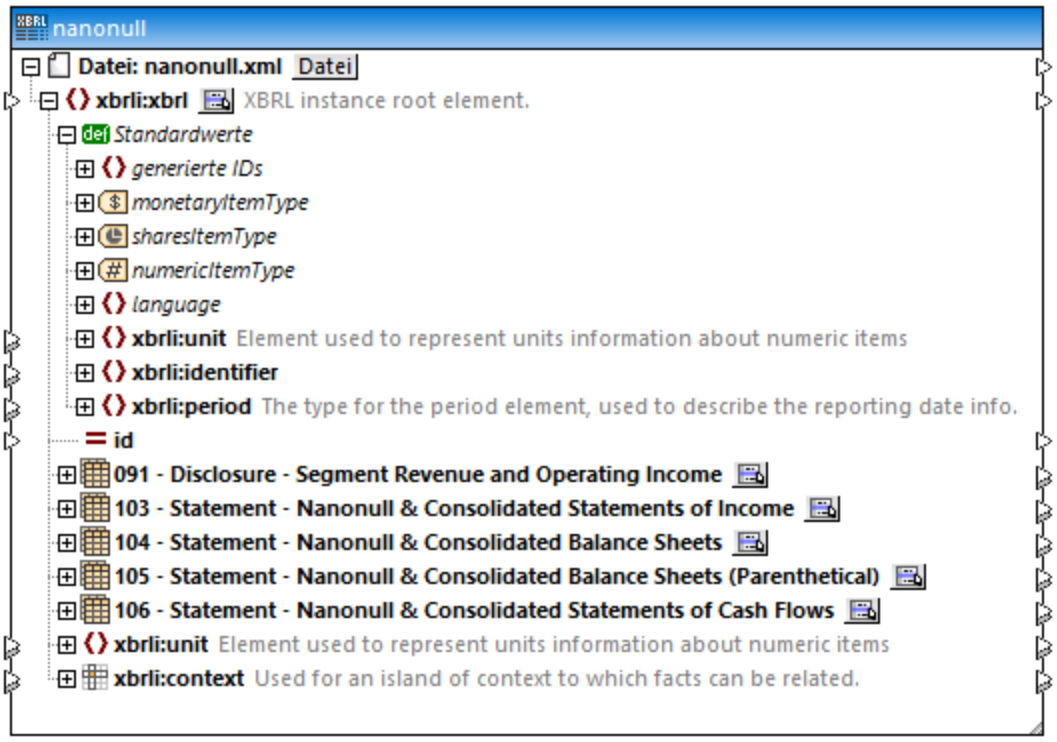
4.8.7 XBRL-Standardwerte

Standardwerte sind eine gute Methode, um Attributen in der XBRL-Komponente Werte zuzuweisen, ohne dass zu allen diesen Attributen explizite Mapping-Verbindungen vorhanden sind. Wenn Sie z.B. dem Attribut `unitRef` des Item `monetaryItemType` in der Hierarchie "Standardwerte" (siehe folgende Abbildung) einen Constant Unit Identifier zuweisen, so wird der Standardwert jedem Monetary XBRL Item Unit Identifier zugewiesen, außer dort, wo dessen Input explizit auf einen anderen Wert gemappt ist. In der Abbildung unten sehen Sie auch die Verwendung des Standardwerts für das Attribut `xml:lang`, mit dem die Sprache einer Fußnote definiert wird.



Standardmäpig ist der Node "Defaults" in der XBRL-Komponente sichtbar; Sie können ihn aber ausblenden, wenn Sie keine Daten auf Standardwerte mappen müssen. Sie können den Node "Standardwerte" auf Root-Ebene (für das gesamte XBRL-Dokument) oder für einzelne Nodes auf jeder beliebigen hierarchischen Ebene der XBRL-Komponente anzeigen. Beachten Sie, dass unterschiedliche Substrukturen unterschiedliche Standardwerte haben können, da die Standardwerte auf jeder beliebigen Ebene der XBRL-Struktur definiert werden können.

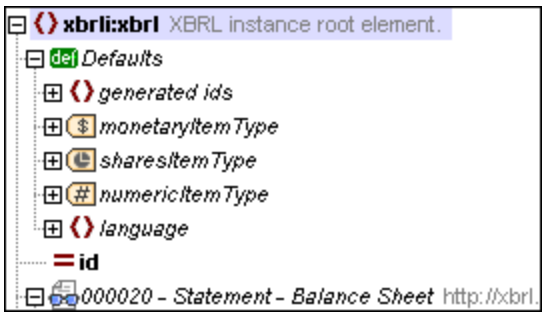
Wenn Ihre XBRL-Komponente anstelle der Presentation/Definition Linkbase-Ansicht die Tabellen-Linkbase-Ansicht verwendet, können Sie auch auf Standard-Aspect-Werte aus der Tabellen-Linkbase mappen. In der Abbildung unten sind die Elemente `xbrli:unit`, `xbrli:identifier` und `xbrli:period` Standard-Aspect-Werte, die Sie beim Mappen auf XBRL-Tabellen verwendet können.



So zeigen Sie den Node "Standardwerte" für ein bestimmtes Item an:

1. Klicken Sie mit der rechten Maustaste auf den Eintrag, für den Sie die Standardwerte definieren möchten und wählen Sie den Befehl **XBRL | Standardwerte anzeigen**. (Alternativ dazu klicken Sie auf die Schaltfläche **Kontextmenü anzeigen** ( ), falls diese für den Node zur Verfügung steht und anschließend auf **Standardwerte anzeigen**.)

   Daraufhin wird ein Standardwerteintrag eingefügt, mit dem Sie Ihre eigenen Standardwerte für die verschiedenen Item-Typen verbinden können.

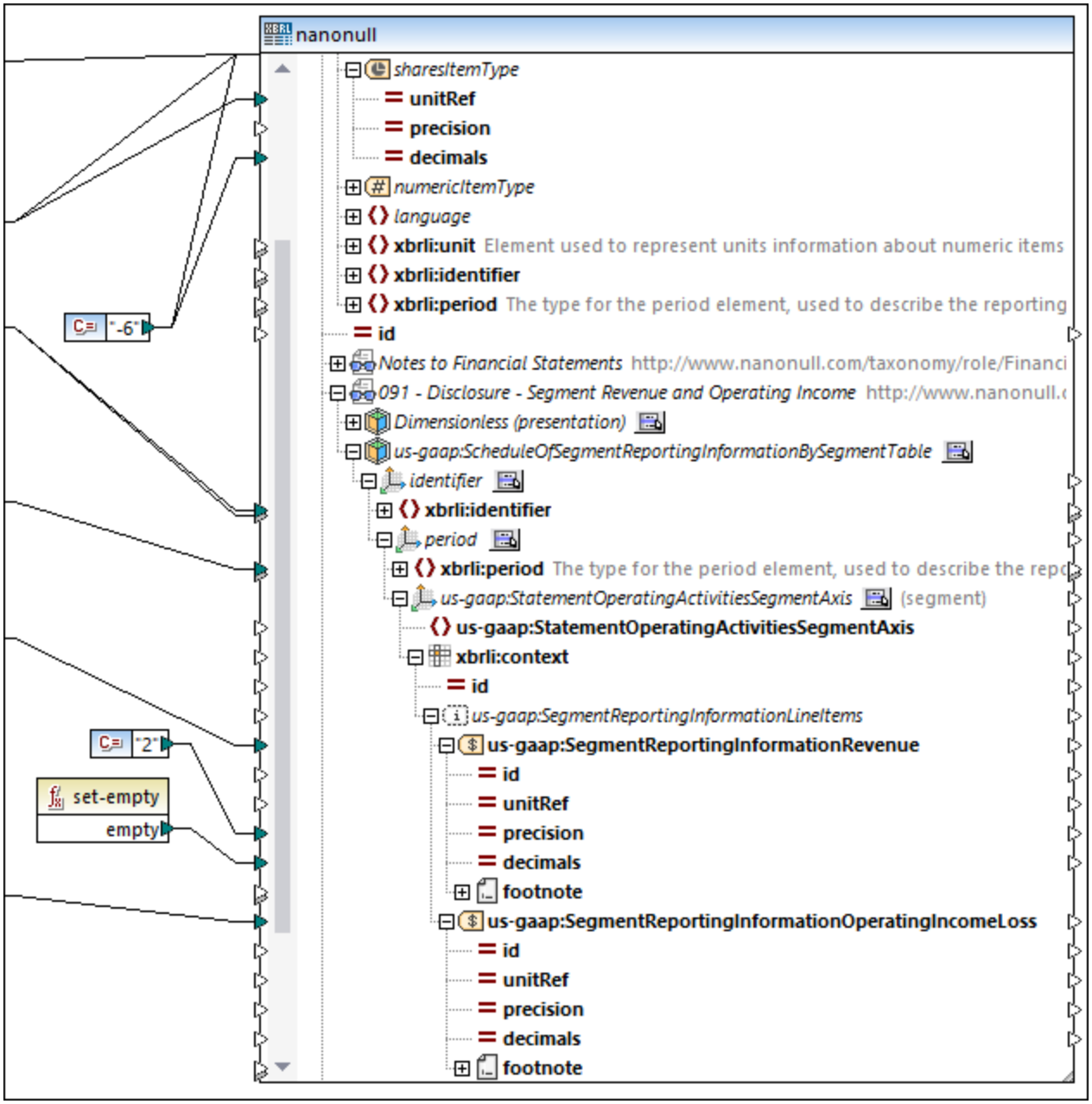


 © 2016-2022 Altova GmbH

Ersetzen oder Deaktivieren eines Standardwerts

Wenn für einige Concept-Attribute, z.B. **decimals**, ein Standardwert definiert wurde, so kann diese Einstellung mittels der Funktion **set-empty** lokal für einzelne Concepts entfernt werden. Nähere Informationen zu Funktionen finden Sie unter Funktionen 942.

In unten gezeigten Beispiel-Mapping deaktiviert die **set-empty** Funktion den Standardwert "-6" für das Monetary Item "Segment Reporting Information, Revenue". Dieses Datenelement wird nun im Bericht mit dem Präzisionswert "2", der auf das precision-Attribut gemappt wurde, ausgegeben, während das andere Datenelement "Segment Reporting Information, Operating Income" mit dem Standarddezimalwert "-6" ausgegeben wird.



In der Mapping-Ausgabe würden diese beiden Datenelemente in der Folge aussehen wie unten gezeigt:

```
<us-gaap:SegmentReportingInformationRevenue contextRef="ctx1" unitRef="USD"
precision="2">23000988</us-gaap:SegmentReportingInformationRevenue>
<us-gaap:SegmentReportingInformationOperatingIncomeLoss contextRef="ctx1" unitRef="USD"
decimals="-6">5605900</us-gaap:SegmentReportingInformationOperatingIncomeLoss>
```


© 2016-2022 Altova GmbH

Behandlung von Contexts

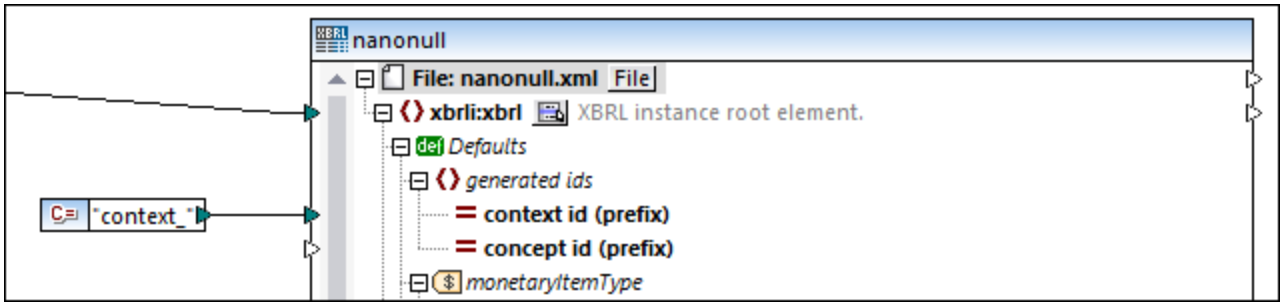
Die hierarchische Struktur in XBRL-Komponenten ermöglicht die automatische Behandlung von Contexts. Die Generierung des `xbrli:context` in XBRL-Ausgabeinstanzen erfolgt automatisch bei der Ausgabe damit in Zusammenhang stehender Concepts.

Der Wert des Attributs `id` eines bestimmten Context in einer XBRL-Instanz ist der Wert des Attributs `contextRef` in den einzelnen damit in Zusammenhang stehenden XBRL Concepts. MapForce nummeriert in einer Output-Instanz automatisch alle erstellten Contexts.

Contexts lassen sich anpassen, indem Sie im Node "context id (prefix)" unter "generated ids" Text als ein Präfix zuweisen.

Wenn Sie z.B. den Konstantenwert "**context**_" als Standardpräfix mappen, werden in der Ausgabeinstanz nacheinander Context-IDs mit den folgenden Werten erstellt: "context_1", "context_2", "context_3", usw.

Wenn kein Standardwert definiert ist, erstellt MapForce alle Context-IDs mit dem Präfix "**ctx**_".



Wenn die XBRL-Output-Instanz **Fußnoten** enthält, müssen die damit in Zusammenhang stehenden Concepts Concept-IDs haben, über die diese mit den automatisch generierten Fußnoten-Links verknüpft werden. Diese Attribute werden automatisch generiert. Das Datenelement "**concept id (prefix)**" kann gemappt werden, um ein solches Präfix zu definieren.

Wenn das Präfix nicht in der XBRL-Komponente gemappt ist, erstellt MapForce alle Concept-IDs mit dem Präfix "**fact**_".

4.8.8 XBRL Hypercubes

XBRL-Hypercubes ermöglichen die automatische Behandlung von Contexts. Nähere Informationen zu XBRL-Hypercubes finden Sie in der XBRL Dimensions 1.0 Spezifikation. MapForce zeigt alle Hypercubes an, die Concepts haben, die in Berichten ausgegeben werden können. Wenn eine der damit verknüpften Hypercube Dimensions keine Domain hat, wird sie in der XBRL-Komponente nicht angezeigt.

Arten von XBRL-Hypercubes
Es gibt folgende Arten von Hypercubes:

- durch die Taxonomie definierte Hypercubes, (z.B. *Statement [Table]*)
- von MapForce generierte Hypercubes. Sie dienen zur Vereinfachung des Standard-Dimensions Identifier und der Periode aus der `Presentation` Linkbase (z.B. *Dimensionless (presentation)*)

- von MapForce innerhalb des **Alle Concepts** Node generierte Hypercubes *Dimensionless*.

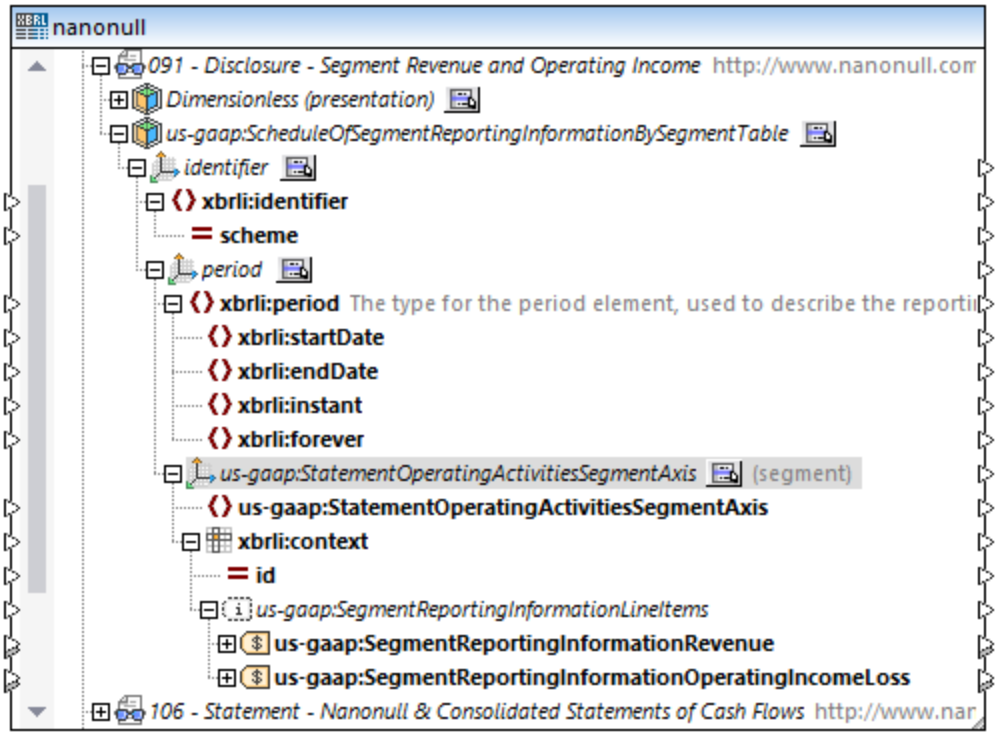
Jeder Hypercube enthält zwei Standard-Dimensions, Identifier und Periode, die das einfache Lesen/Schreiben dieser beiden Elemente für die einzelnen Contexts unterstützen. Zusätzliche in der Taxonomie definierte Dimensions werden automatisch mit den Context-Elementen `xbrli:segment` und `xbrli:scenario` in Beziehung gesetzt.

In als `Dimensionless (presentation)` gekennzeichneten Hypercubes werden beide Standard-Dimensions verwendet. Die hierarchische Reihenfolge der Concepts, die in ihrem Context-Node angezeigt werden, stammt aus der Presentation Linkbase.

In `Dimensionless` Hypercube-Elementen werden ebenfalls beide Standard-Dimensions verwendet, es gibt jedoch keine hierarchische Concept-Reihenfolge und es wird nur die nicht hierarchische Liste aller in der Taxonomie definierten Concepts angezeigt.

Alle anderen Hypercubes sind in der Taxonomie definiert und werden entsprechend dem in der Label Linkbase der Taxonomie definierten Namen angegeben.

Hypercubes sowie ihre Dimensions (oder Achsen) verfügen jeweils über ein kleines -Symbol, über das Sie die Darstellung (Presentation) für die einzelnen Dimensions in der Komponente definieren können. In der unten gezeigten Abbildung sehen Sie eine Beispiel-Taxonomiedatei, die sowohl generierte als auch von der Taxonomie definierte Hypercubes enthält.



Neben Concepts werden alle Tuples, die nur Elemente desselben Hypercube enthalten, als Children des Kontext-Node angezeigt.

© 2016-2022 Altova GmbH

4.8.8.1 Anzeigen von Dimensions in einer Komponente

Dimension-Elemente in XBRL beziehen sich entweder auf **explizite (explicit)** oder **typed** Dimension-Werte in der Instanz. Die Annotationen der einzelnen Dimension-Elemente zeigen in Klammern an, ob die Dimension in den Context-Elementen `xbrli:segment` oder `xbrli:scenario` angegeben wird.

Bei **Typed** Dimension-Elementen werden die Elemente ihres XML-Schema-Typs als Child-Elemente angezeigt. Ihre Werte können direkt gemappt werden.

Explicit Dimensions in einer XBRL-Taxonomie haben einen Wert vom Typ `xs:QName` aus einer bestimmten Domain. Diese umfasst die Werte der Mitglieder der XBRL Domain und die Werte des XBRL Domain-Elements selbst. Explicit Dimensions können, je nach Mapping-Anforderungen und abhängig von der anderen Komponente/Struktur, auf die von/auf XBRL gemappt wird, auf zwei verschiedene Arten angezeigt werden.

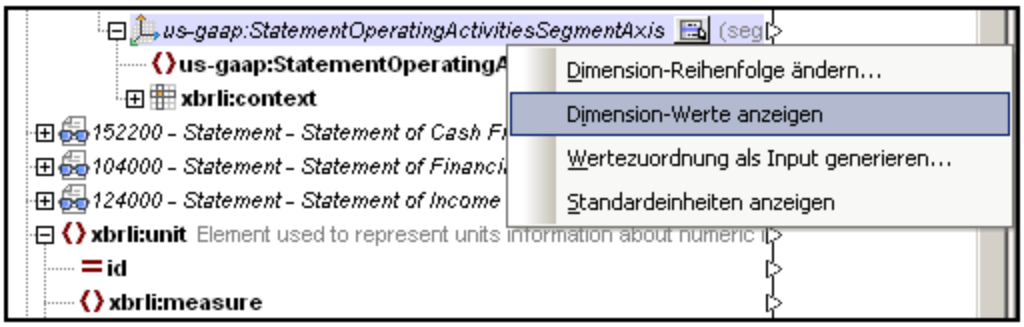
Anfangs wird die explicit Dimension mit einem **einzigen** Child Node angezeigt und kann direkt über dieses Child-Element gemappt werden, z.B. "Statement, Equity Components [Axis]".

Dies ist (für eine XBRL-Zielkomponente) nützlich, wenn die Dimension-Werte von einem Feld der Quelldaten z.B. einem Datenbankfeld oder einer Spalte in einer Excel-Tabelle ermittelt werden können. Da die Quelldaten im Allgemeinen den erforderlichen `QName`-Datentyp nicht enthalten, kann MapForce sie automatisch mit Hilfe der Wertezuordnungsfunktion erstellen (siehe Generieren von Wertezuordnungen für Hypercube Dimensions 807).

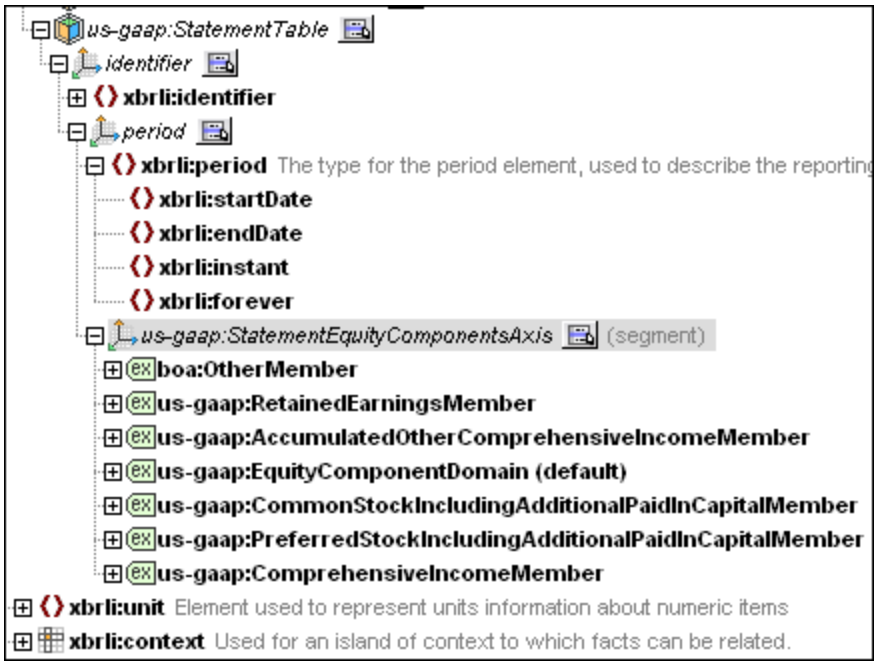
Um für die Facts unterschiedliche Mappings zu ermöglichen, können separate Nodes für jeden einzelnen Wert der Dimension Domain angezeigt werden.

So zeigen Sie Dimension-Werte in der Komponente an:

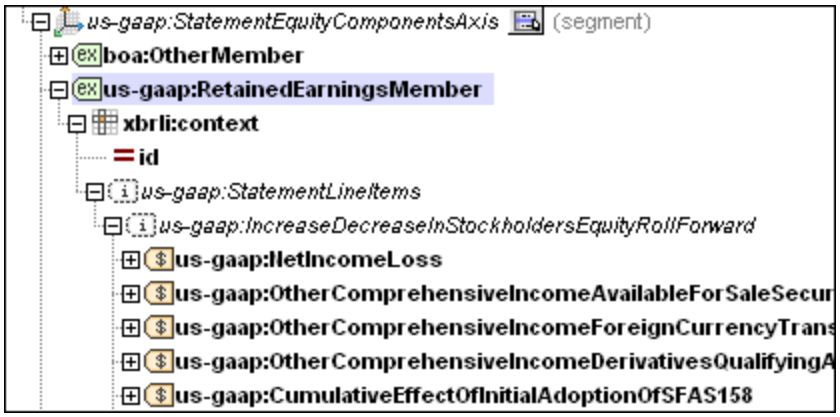
1. Klicken Sie auf das Symbol der Dimension, für die die Werte angezeigt werden sollen, und wählen Sie den Befehl "Dimension-Werte anzeigen".



Daraufhin ändern sich die Einträge, die Sie unterhalb des Dimension-Namens sehen. Die **Dimension**-Domain und die **Member** Items sind nun sichtbar, wobei jeder dieser Einträge mit einem hellgrünen Symbol gekennzeichnet ist. Alle diese Einträge sind explizite Mitglieder (explicit members) einer Domain, was durch das Präfix "ex" im Symbol für den Eintrag gekennzeichnet ist.



Jedes explicit Member enthält nun dieselbe Substruktur, sodass für jedes andere Mappings möglich sind.



Wenn der Output eines Concept gemappt wird, werden nur jene Werte verwendet, für die das dazugehörige Context-Element den entsprechenden Dimension-Wert hat. So wird z.B. der Wert "Net Income (Loss)" in der Instanz nur für Contexts gemappt, die den Dimension-Wert "Comprehensive Income [Member]" für die Dimension "Statement, Equity Components [Axis]" enthalten. Es ist keine zusätzliche Filterung erforderlich.

Beim Schreiben von XBRL-Output-Instanzen wird die automatische Generierung ordnungsgemäßer Dimension-Werte im Context unterstützt. So erhält z.B. der Context Node **xbrli:context** in seinem Kontext-Element (**xbrli:segment**) für jedes im Bericht ausgegebene Monetary Item "Net Income (Loss)" ein Element für die explizite Dimension "Statement, Equity Components [Axis]", das den Wert "Comprehensive Income [Member]" enthält.

© 2016-2022 Altova GmbH

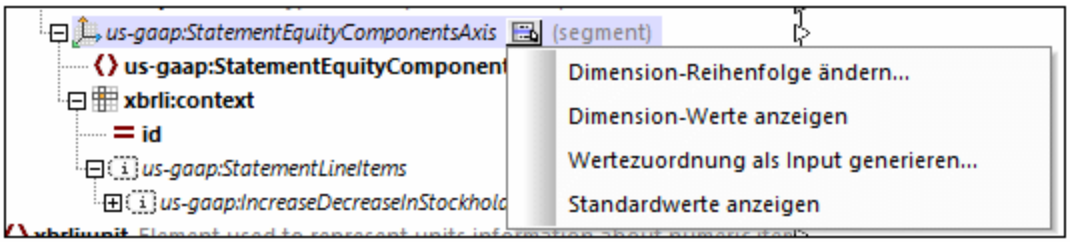
4.8.8.2 Ändern der Reihenfolge von Dimensions

Anfangs werden alle Dimensions eines Hypercube in MapForce als verschachtelte Child Nodes angezeigt, wodurch automatisch eine Hierarchie entsteht. Die hierarchische Reihenfolge innerhalb des Hypercube kann geändert werden, sodass sie der anderen Komponente des Mappings (der Nicht-XBRL-Komponente) entspricht.

Wo Dimension-Werte für einige Concepts spezifisch definiert werden müssen, kann MapForce eine Dimension ohne Hierarchie anzeigen und diese als Child-Element des Context Node anzeigen.

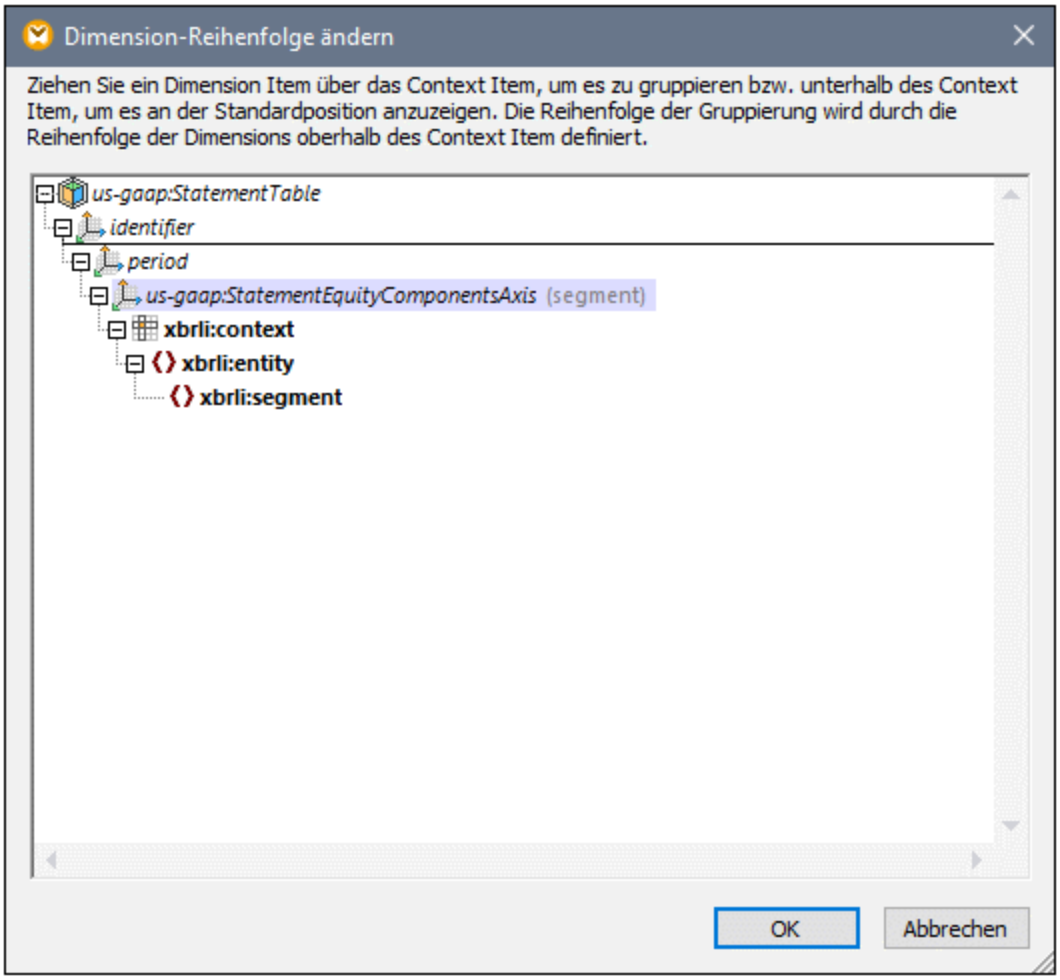
So ändern Sie die Reihenfolge von Dimension Items:

1. Klicken Sie auf ein beliebiges -Symbol des entsprechenden Hypercube und wählen Sie im Popup-Fenster den Befehl "Dimension-Reihenfolge ändern".



   Daraufhin wird ein Dialogfeld geöffnet, in dem Sie die verschiedenen Dimensions eines Hypercube neu positionieren können. Beachten Sie, dass ein Hypercube zwei **Standard-Dimensions** hat, nämlich **Identifier** und **Zeitraum**, deren Reihenfolge im Hypercube ebenfalls geändert werden kann.

2. Klicken Sie auf die Hypercube-Dimension und ziehen Sie sie mit der Maus an die gewünschte Stelle im Dialogfeld. An Stellen, an die Sie die Dimension ziehen können, erscheint eine Linie.

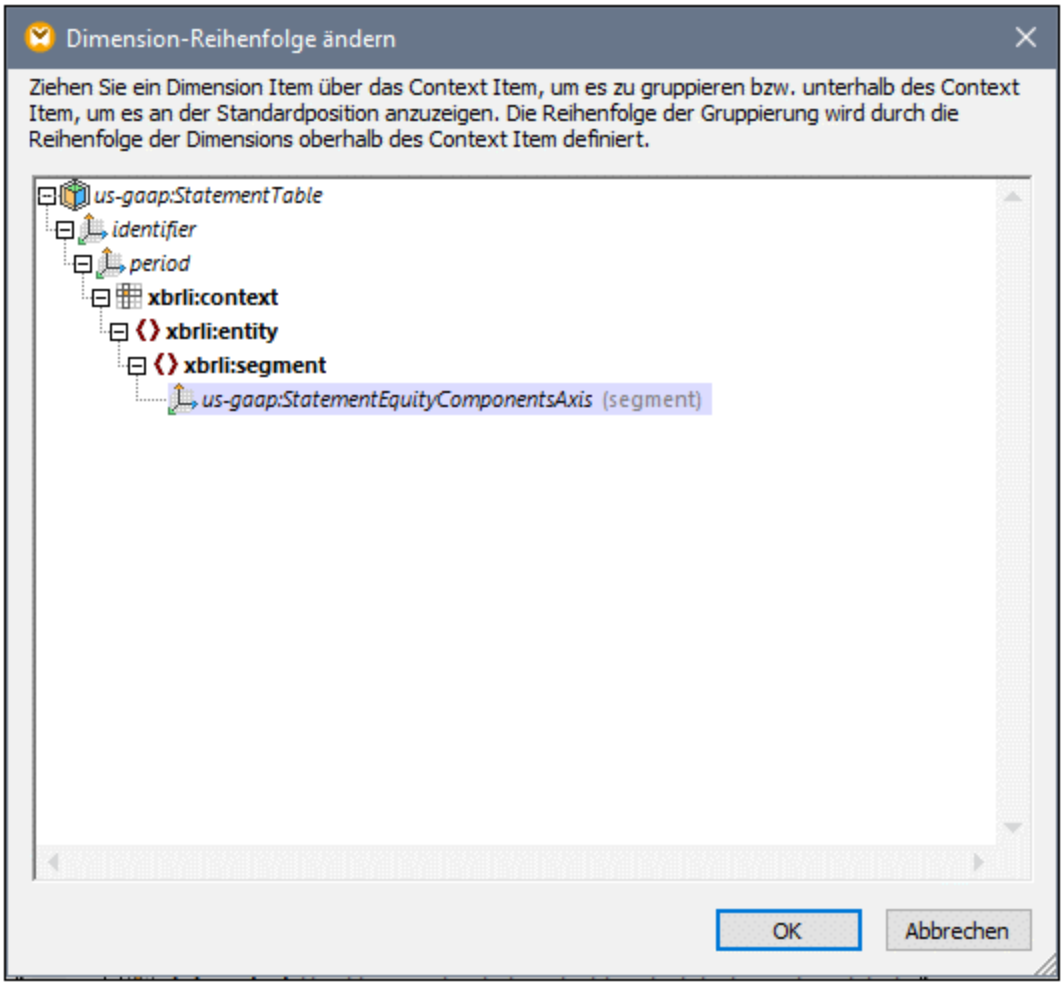


3. Klicken Sie auf OK, um das Dialogfeld zu schließen, damit die Dimension im Dokument an die neue Position verschoben wird.

So exkludieren Sie eine Dimension aus einer Hierarchie:

- Ziehen Sie die Dimension unter die Zeile xbrli:context. Daraufhin wird sie in dieses Context Item eingefügt.

© 2016-2022 Altova GmbH

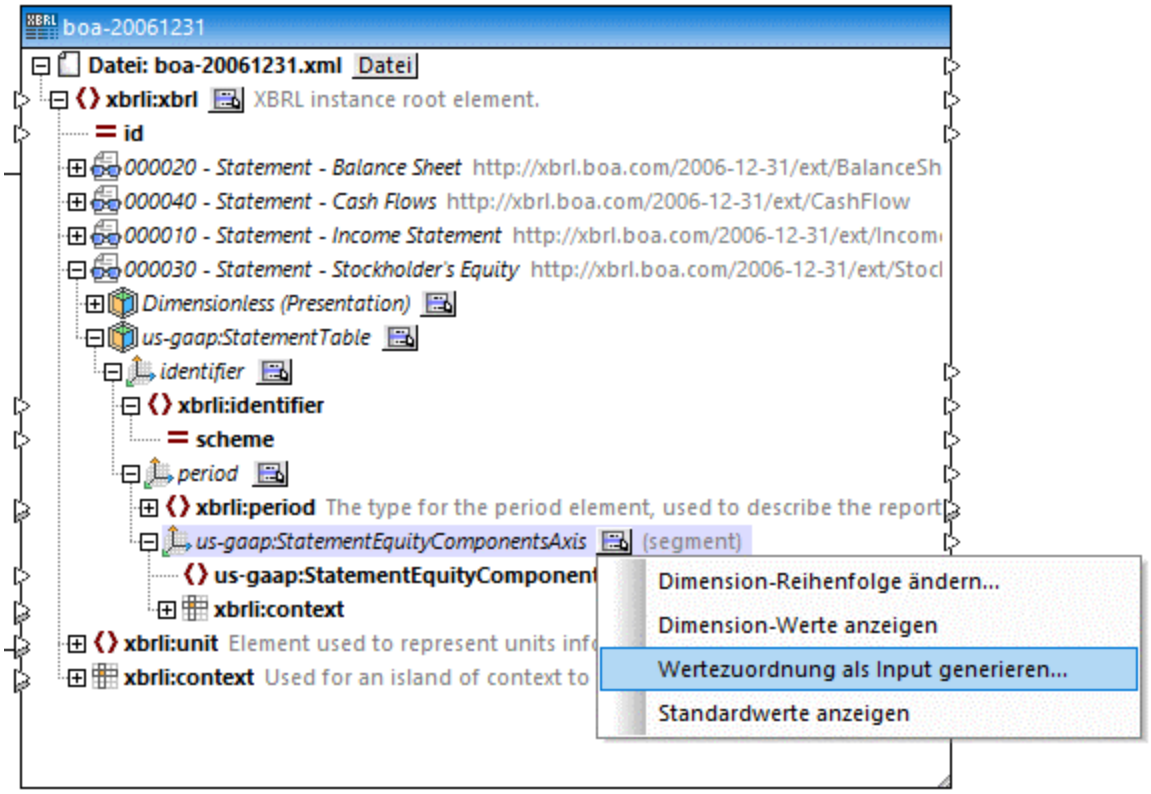


4.8.8.3 Generieren von Wertezuordnungen für Hypercube Explicit Dimensions

Mit dieser Option werden Input-Daten jedes Typs in der Zielkomponente in einen gültigen `QName` transformiert, d.h. der Input-String wird in den Namen mit Präfix (`QName`) umgewandelt.

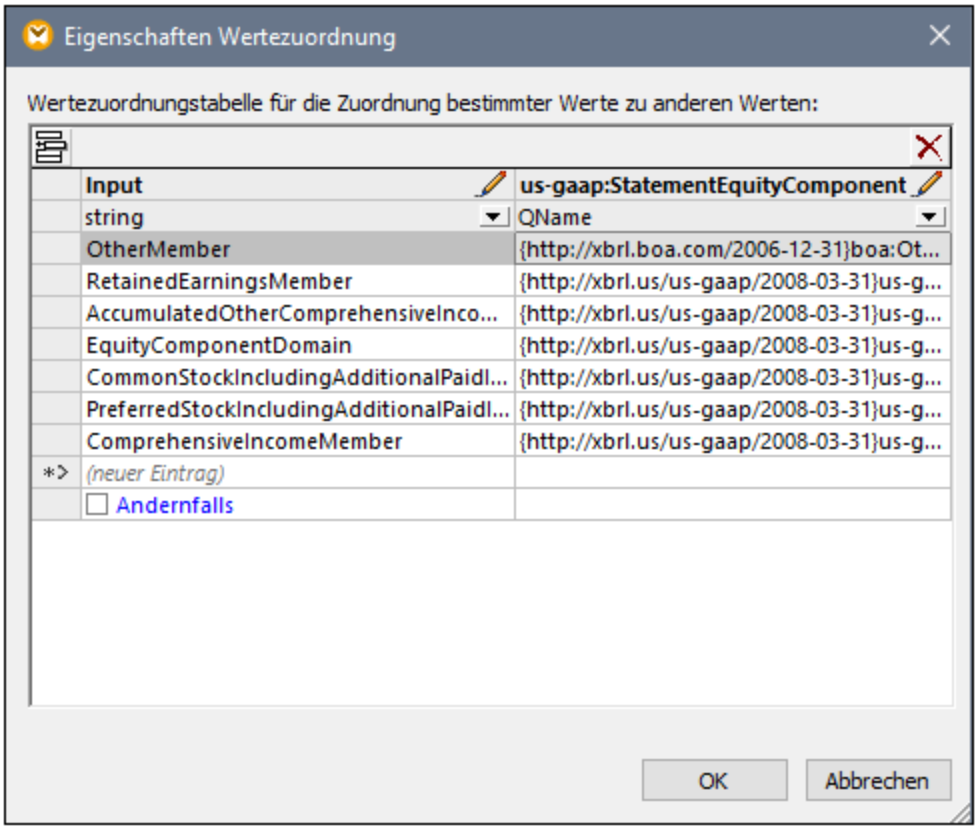
So generieren Sie eine Wertezuordnungstabelle für die ausgewählte Hypercube-Dimension:

1. Stellen Sie sicher, dass die Dimension-Werte für die entsprechende Hypercube-Dimension nicht sichtbar sind.
2. Klicken Sie auf das Symbol der entsprechenden **Dimension** und wählen Sie im Popup-Fenster die Option "Wertezuordnung als Input generieren".



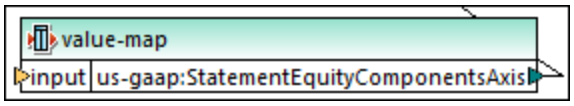
Daraufhin wird das Dialogfeld "Eigenschaften Wertezuordnung" geöffnet, das automatisch anhand der Standard-Dimension Domain und der laut Taxonomie definierten Domain Members generierte Input- und Output-Werte enthält.

© 2016-2022 Altova GmbH



3. Bearbeiten Sie die Input-Werte, falls erforderlich, und klicken Sie auf OK, um die Wertezuordnungskomponente einzufügen. Beachten Sie, dass Sie den Text der Spaltenüberschriften durch Doppelklick auf die Überschrift bearbeiten können, um ihn zu kürzen oder eine bessere Beschreibung einzugeben.

   Daraufhin wird eine Wertezuordnungsfunktion eingefügt, in der die Namen des Input- und Output-Parameters angezeigt werden. Der Output-Konnektor wird automatisch mit dem Domain-Element der Zielkomponente verbunden.



4. Verbinden Sie das Quelldatenelement, das die zu transformierenden Input-Daten enthält, mit dem Input-Parameter der Funktion.

Ein allgemeines Beispiel zur Verwendung der Wertezuordnungsfunktion finden Sie unter Verwendung von Wertezuordnungen 921.

4.8.9 XBRL-Tabellen

Wenn Ihre XBRL-Taxonomie die Table Linkbase referenziert, können Sie MapForce so konfigurieren, dass die Table Linkbase Node-Typen (wie z.B. Tabellen oder Aufschlüsselungen (Breakdowns)) angezeigt werden. Dadurch können Sie Daten von und auf in dargestellten Tabellen angezeigte Facts mappen. Um die Ansicht *Tabellen aus Tabellen-Linkbase* in der XBRL-Komponente anzuzeigen, gehen Sie folgendermaßen vor:

1. Gehen Sie zum Menü **Einfügen** und klicken Sie auf **XBRL-Dokument**.
2. Navigieren Sie zur XBRL-Instanzdatei. Siehe Hinzufügen von XBRL-Dateien als Mapping-Komponenten 769.
3. Klicken Sie neben dem Root-Element auf die Schaltfläche  (**Kontextmenü anzeigen**). Klicken Sie anschließend auf **Strukturansichten auswählen** 770.
4. Aktivieren Sie das Kontrollkästchen *Tabellen aus Tabellen-Linkbase* und klicken Sie auf **OK**.

Informationen zu anderen Ansichten finden Sie unter Auswählen von Strukturansichten 770.

Anmerkungen

- Beim Auslesen von Daten aus einer XBRL-Komponente (d.h., wenn eine XBRL-Komponente die Quellkomponente ist) können Sie über das Kontextmenü  jede beliebige Kombination von Strukturansichten auswählen. Wenn jedoch beim Schreiben in eine XBRL-Komponente (d.h. wenn die XBRL-Komponente die Zielkomponente ist) die **Tabellen aus Tabellen-Linkbase**-Ansicht ausgewählt ist, können aus technischen Gründen Daten nur in diese Ansicht und in keine andere geschrieben werden.
- Für die **Tabellen aus Tabellen-Linkbase**-Ansicht muss BUILT-IN 29 als Transformationssprache ausgewählt sein.

Zur Darstellung von Informationen aus der Table Linkbase wird in MapForce normalerweise das Strukturmodell verwendet. Es gelten die folgenden Ausnahmen:

- Zusammengeführte Regel-Nodes sind sichtbar.
- Roll-up Nodes ohne gleichrangige Nodes bilden keine separate Hierarchieebene.

Das Strukturmodell ist eines von drei in der Table Linkbase Specification 1.0, §4 definierten Datenmodellen. Zum Identifizieren von Teilen des durch das Mapping referenzierten Strukturmodells verwendet MapForce auch Informationen aus dem Definitionsmodell..

Da eine MapForce-Komponentenstruktur eindimensional ist, während Tabellen bis zu drei Dimensionen haben können (x, y, z), zeigt MapForce die Tabellendimensionen durch Ineinanderschachteln aller Aufschlüsselungen (Breakdowns) in der folgenden Standardreihenfolge an: z, y, x. Der Zweck von Aufschlüsselungsnodes in MapForm ist, anzugeben, wo die einzelnen Aufschlüsselungen beginnen.

Filing Indicators (EBA)

Gemäß den EBA Filing Rules wird mit Hilfe von Filing Indicators angezeigt, ob der Benutzer beabsichtigt, ein bestimmtes Berichtselement in einem Bericht auszugeben oder nicht. Filing Indicators werden in EBA-Taxonomien verwendet, wenn Tabellen ein Label mit der Linkrole `http://www.eurofiling.info/xbrl/role/filing-indicator-code` haben (EBA Filing Rules §1.6.3). Zusätzlich dazu muss das Concept für die Filing Indicator Facts vorhanden sein: `find:fIndicators` (EBA-kompatibler Stil).

© 2016-2022 Altova GmbH

Der Node `Filing Indicators` wird in der XBRL-Zielkomponente angezeigt, wenn (i) die Ansicht *Tabellen aus Tabellen-Linkbase* in einer XBRL-Komponente aktiv ist und (ii) in der Taxonomie die oben angegebenen Filing Indicators verwendet werden. Der Node `Filing Indicators` wird als Child des Root-Elements `xbrli:xbrl` zwischen dem Attribut `id` und dem ersten Tabellen-Node angezeigt.

Sie können in MapForce automatisch Filing Indicators für XBRL-Zielkomponenten generieren. Der Node `Filing Indicators` hat keine mapbaren Datenpunkte, sondern nur eine Schaltfläche zum Ändern der Einstellung[772]. Wenn Sie auf die Schaltfläche klicken, können Sie auswählen, wie Filing Indicators generiert werden sollen: (i) *Keiner*, d.h. in der Ausgabe werden keine Filing Indicators generiert (Standardoption) und (ii) *Automatisch generieren*, d.h. es werden für alle relevanten Tabellen in der Komponente Filing Indicators generiert. Anhand von Tabellen, die Input-Verbindungen in der untergeordneten Struktur aufweisen, wird ein positiver Filing Indicator `filed="true"` generiert; anhand von nicht verbundenen Nodes wird ein negativer Filing Indicator `filed="false"` generiert.

In diesem Abschnitt

In diesem Abschnitt wird erläutert, wie Sie mit Entities aus der Tabellen-Linkbase arbeiten. Er ist in die folgenden Kapitel gegliedert:

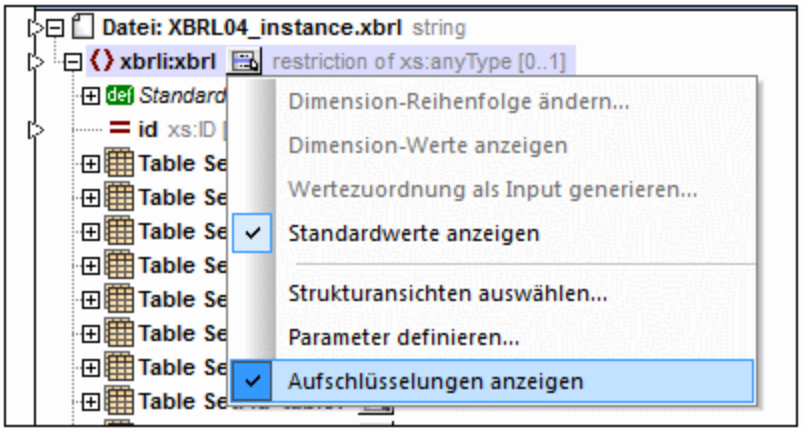
- Anzeigen oder Ausblenden von Aufschlüsselungen[811]
- Ändern der Reihenfolge von Aufschlüsselungen[812]
- Arbeiten mit Parametern[813]

4.8.9.1 Anzeigen oder Ausblenden von Aufschlüsselungen

Sie können in XBRL-Dokumenten, die die Table Linkbase referenzieren, auswählen, ob Aufschlüsselungen (Breakdowns) in der Komponente angezeigt oder ausgeblendet werden sollen.

So zeigen Sie Aufschlüsselungen an oder blenden diese aus:

1. Suchen Sie in der XBRL-Komponente einen Node, der die Schaltfläche **Kontextmenü anzeigen** ( ) rechts vom Node enthält (z.B. den Root-Node).



2. Klicken Sie auf die Schaltfläche **Kontextmenü anzeigen** ( ) und klicken Sie anschließend auf **Aufschlüsselungen anzeigen**.

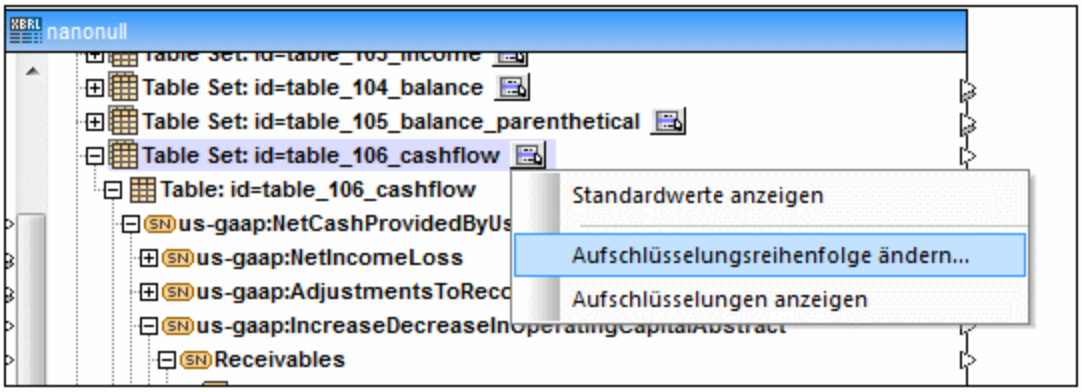
4.8.9.2 Ändern der Reihenfolgen von Aufschlüsselungen

Sie können in XBRL-Dokumenten, die die Table Linkbase referenzieren, die Reihenfolge von Aufschlüsselungen ändern.

So ändern Sie die Reihenfolge von Aufschlüsselungen

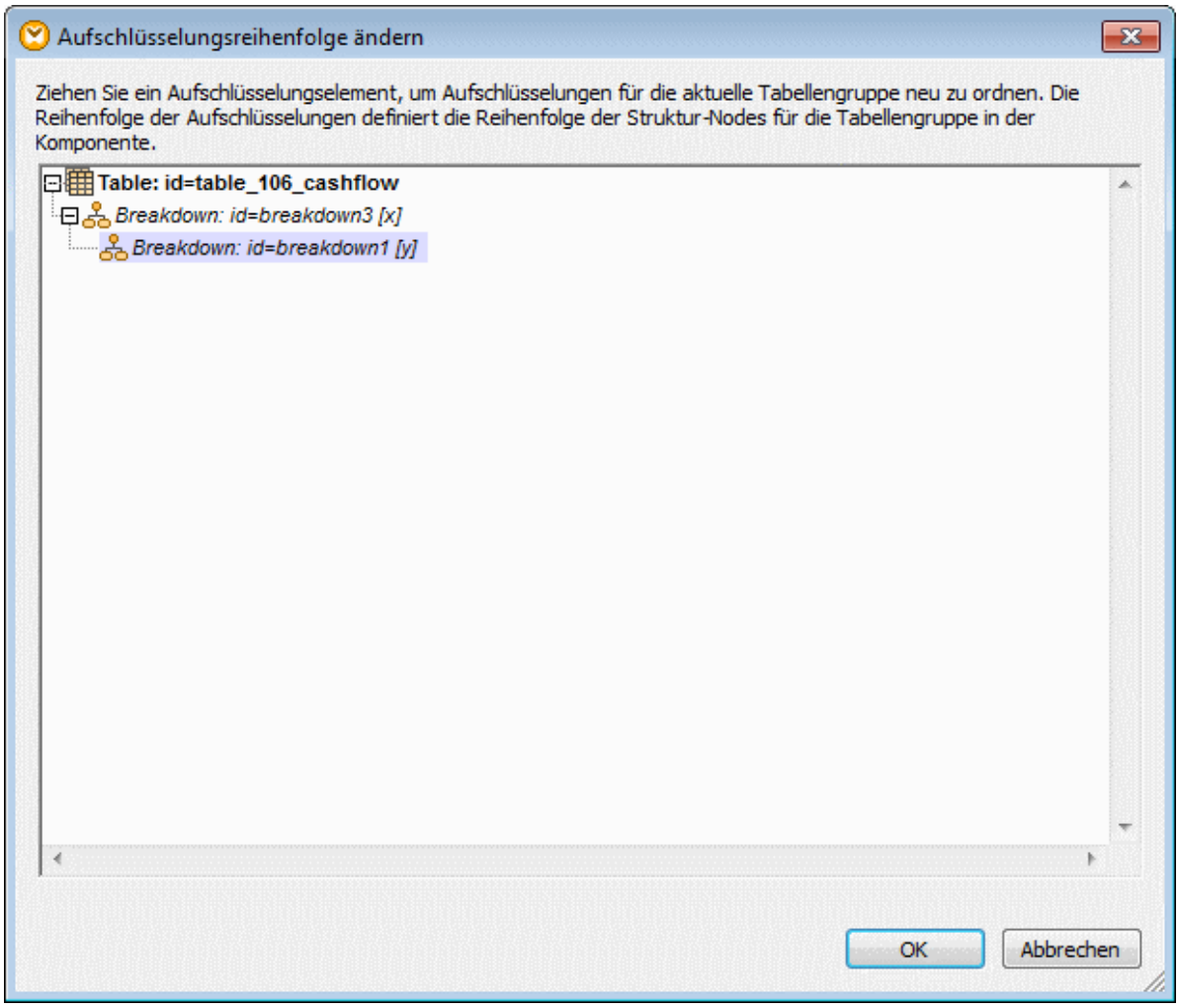
1. Suchen Sie in der XBRL-Komponente den Node, für den Sie die Aufschlüsselungsreihenfolge ändern möchten.

2. Klicken Sie auf die Schaltfläche **Kontextmenü anzeigen** ( ) rechts vom Node und anschließend auf den Befehl **Aufschlüsselungsreihenfolge ändern**. (Wenn die Menüoption **Aufschlüsselungsreihenfolge ändern** nicht verfügbar ist, bedeutet dies, dass die Anwendung dieser Option für den ausgewählten Node nicht sinnvoll ist.)



3. Ziehen Sie die Aufschlüsselungen an die gewünschte Stelle (in der folgenden Abbildung können Sie z.B. die Aufschlüsselung *y* über die Aufschlüsselung *x* ziehen).

© 2016-2022 Altova GmbH



4. Klicken Sie auf **OK**.

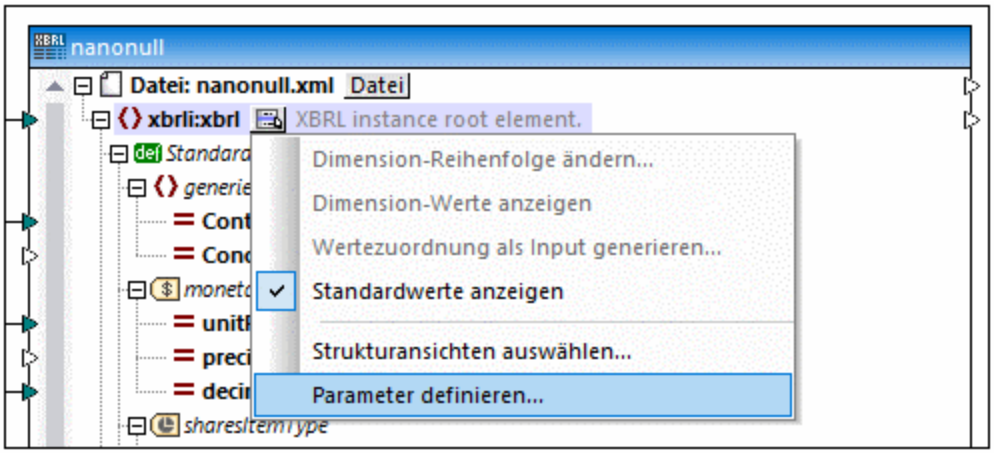
4.8.9.3 Arbeiten mit Parametern

Sie können in XBRL-Dokumenten, die die Table Linkbase referenzieren und Parameter enthalten (Table Linkbase Specification 1.0, §5.3), die Parameterwerte ändern. Falls dies der Kontext zulässt, können Sie auch neue Parameterwerte an vorhandene Parameter anhängen.

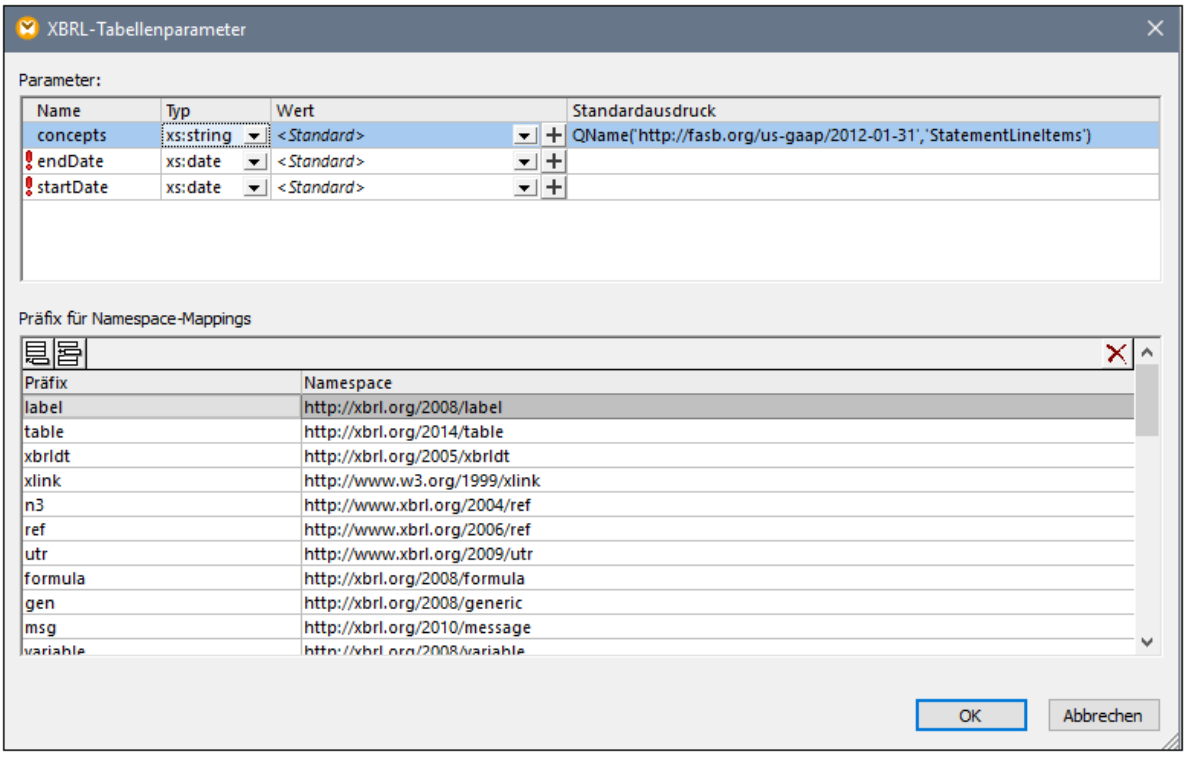
Die Parameterwerte können über das Dialogfeld "XBRL-Tabellenparameter" geändert werden (siehe Anleitung unten). In diesem Dialogfeld werden sowohl Parameter angezeigt, die auf globaler Ebene (an einer beliebigen Stelle im Discoverable Taxonomy Set) als auch in einzelnen Tabellen definiert sind. Wenn Sie einen Parameterwert ändern und das Dialogfeld schließen, versucht MapForce, alle Tabellen in der Komponente neu zu generieren und die Nodes der Struktur zusammenzuführen. Wenn die Struktur aufgrund fehlender Parameter nicht neu generiert werden kann, wird im entsprechenden Node der Komponente ein Fehler angezeigt. Nähere Informationen zum Fehler finden Sie im Fenster Meldungen[33].

So ändern Sie Parameterwerte

1. Klicken Sie auf die Schaltfläche **Kontextmenü anzeigen** ( ) rechts vom Root-Node und wählen Sie anschließend den Befehl **Parameter definieren**. (Wenn die Menüoption **Parameter definieren** nicht zur Verfügung steht, enthält die Tabellen Linkbase keine Parameter.)



2. Wählen Sie den Parametereintrag aus und geben Sie den neuen Wert in die Spalte "Wert" ein. Obligatorische Parameter haben als Symbol ein rotes Ausrufezeichen ( ).



 © 2016-2022 Altova GmbH

So hängen Sie Parameterwerte an vorhandene Parameter an:

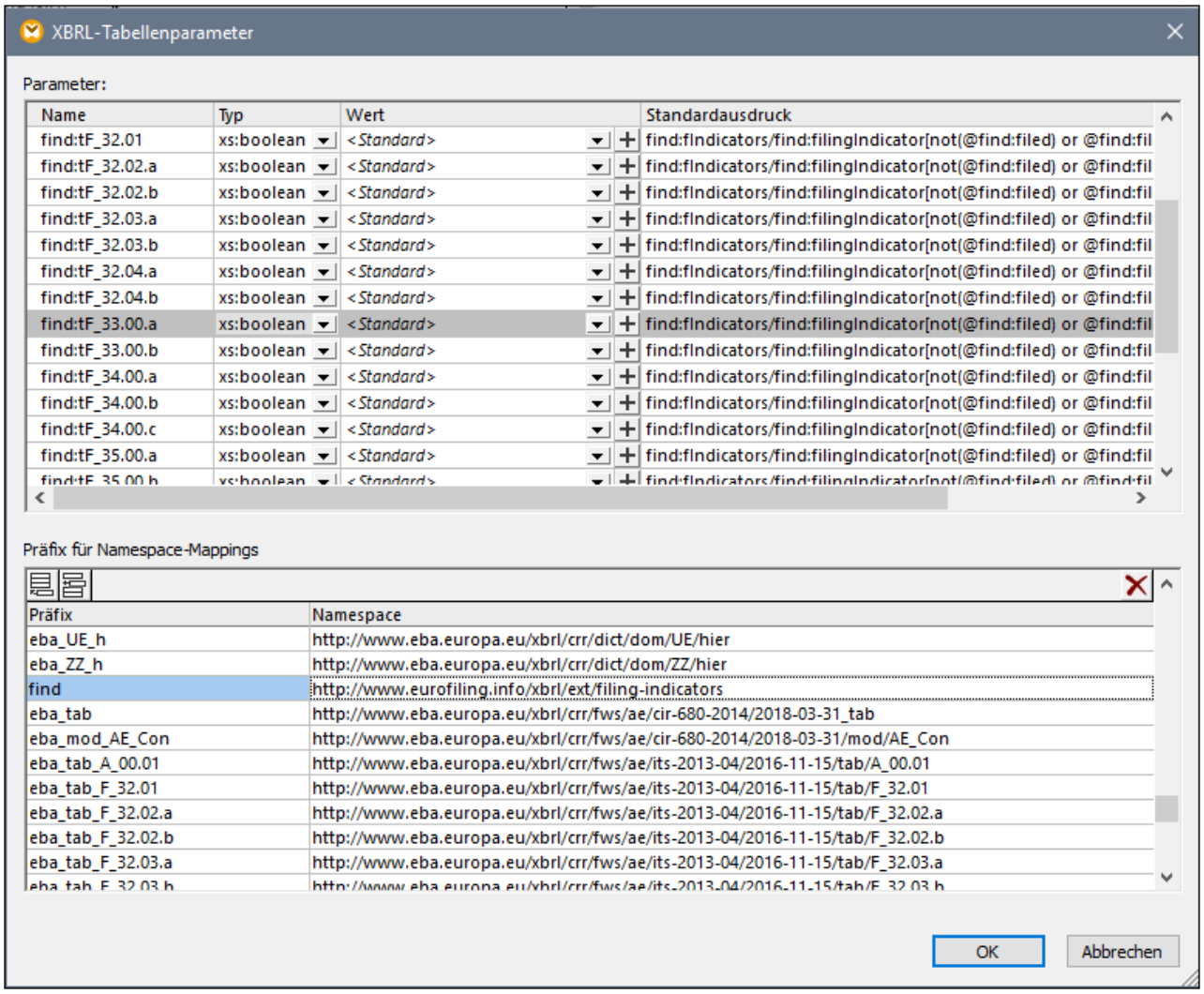
- Wählen Sie den Parametereintrag aus und klicken Sie anschließend in der Spalte "Wert" auf die Schaltfläche **Parameterwert hinzufügen** ( ). Beachten Sie, dass nur dann ein Wert angehängt werden kann, wenn der Parametertyp das Hinzufügen mehrerer Werte unterstützt.

So entfernen Sie einen angehängten Wert:

Klicken sie neben dem Wert, den Sie entfernen möchten, auf die Schaltfläche **Wert entfernen** ( ).

Präfix für Namespace Mappings

Die Namen von Tabellenparametern bestehen aus einem optionalen Namespace und einen lokalen Namen (d.h. es handelt sich um QNames, die die Form *namespace_präfix:localername*) haben. MapForce erkennt solche Mappings automatisch und zeigt diese im unteren Bereich des Dialogfelds "XBRL-Tabellenparameter" an. So ist z.B. in der Abbildung unten das Präfix **find** auf **http://eurofiling.info/xbrl/ext/filling-indicators** gemappt.



Wenn MapForce ein Parameterpräfix einem benutzerdefinierten Namespace entsprechend parsen soll, können Sie das vorhandene Präfixpaar ändern oder ein benutzerdefiniertes hinzufügen. Klicken Sie auf die Schaltfläche **Einfügen** oder **Anhängen** , um Name-Präfix-Zuordnungen zu erstellen. Beachten Sie, dass dies keinen Einfluss auf Präfixnamen von XBRL-Instanzelementen hat, wenn in Ihrem Mapping Daten in eine XBRL-Instanz geschrieben werden. Dies dient einzig und allein als Information für MapForce, wie die XBRL-Struktur geparst und neu generiert werden soll, nachdem Sie das Dialogfeld geschlossen haben.

4.8.10 XBRL-Mapping-Beispiele

Dieser Abschnitt enthält die folgenden XBRL-Mapping-Beispiele:

- Microsoft Access auf XBRL 817 (zeigt, wie Daten aus einer Microsoft Access-Datenbank auf eine XBRL-Taxonomie gemappt werden, sodass eine gültige XBRL-Instanzdatei erzeugt wird)
- Microsoft Excel auf XBRL 818 (zeigt, wie Daten aus einem Microsoft Excel-Arbeitsblatt auf eine Taxonomie gemappt werden, sodass eine gültige XBRL-Instanzdatei erzeugt wird).

© 2016-2022 Altova GmbH

4.8.10.1 Microsoft Access auf XBRL

Dieses Beispiel steht unter dem Namen **DB_to_XBRL.mfd** im Ordner **C:\Users\<username>\Documents\Altova\MapForce2023\MapForceExamples** zur Verfügung. Zum Extrahieren der Datenbankdaten werden darin verschiedene Filter und Funktionen verwendet.

Die Taxonomie **nanonull.xsd** wurde von US:GAAP abgeleitet. Im Mapping wird eine XBRL-Output-Instanz erstellt, die alle Contexts, Concepts, Units und Fußnoten für eine "Disclosure" und drei "Statements" (Berichte).

Im Bericht **"091 - Disclosure - Segment Revenue and Operating Income"** sehen Sie, wie Dimension-Werte in MapForce gemappt werden können. Der Hypercube "us-gaap:ScheduleOfSegmentReportingInformationBySegmentTable" enthält eine explizite Dimension "us-gaap:StatementOperatingActivitiesSegmentAxis".

Ihre Domain wurde in der Taxonomie durch drei Dimensionwerte, nämlich "nanonull:USA", "nanonull:Europe" und "nanonull:Asia" erweitert. Im Mapping sehen Sie, wie die Werte der Datenbankspalte "Name" der Tabelle "Region" mittels einer Wertezuordnungsfunktion auf die erforderlichen Dimension-Werte vom Typ QName gemappt werden.

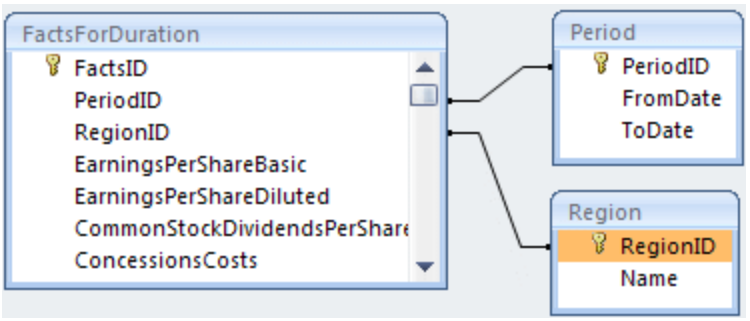
Im Bericht **"106 - Statement - Nanonull and Consolidated Statement of Cash Flows"** sehen Sie, wie mit Hilfe von MapForce Facts in die Output-Instanz, die sowohl mit Duration als auch Instant Periods verknüpft ist, geschrieben werden können.

Wie im Mapping gezeigt, können Facts wie z.B. "Cash and cash equivalents at beggining (end) of period" durch Duplizieren des Period Item in der Hierarchiestruktur ordnungsgemäß in einem Bericht ausgegeben werden.

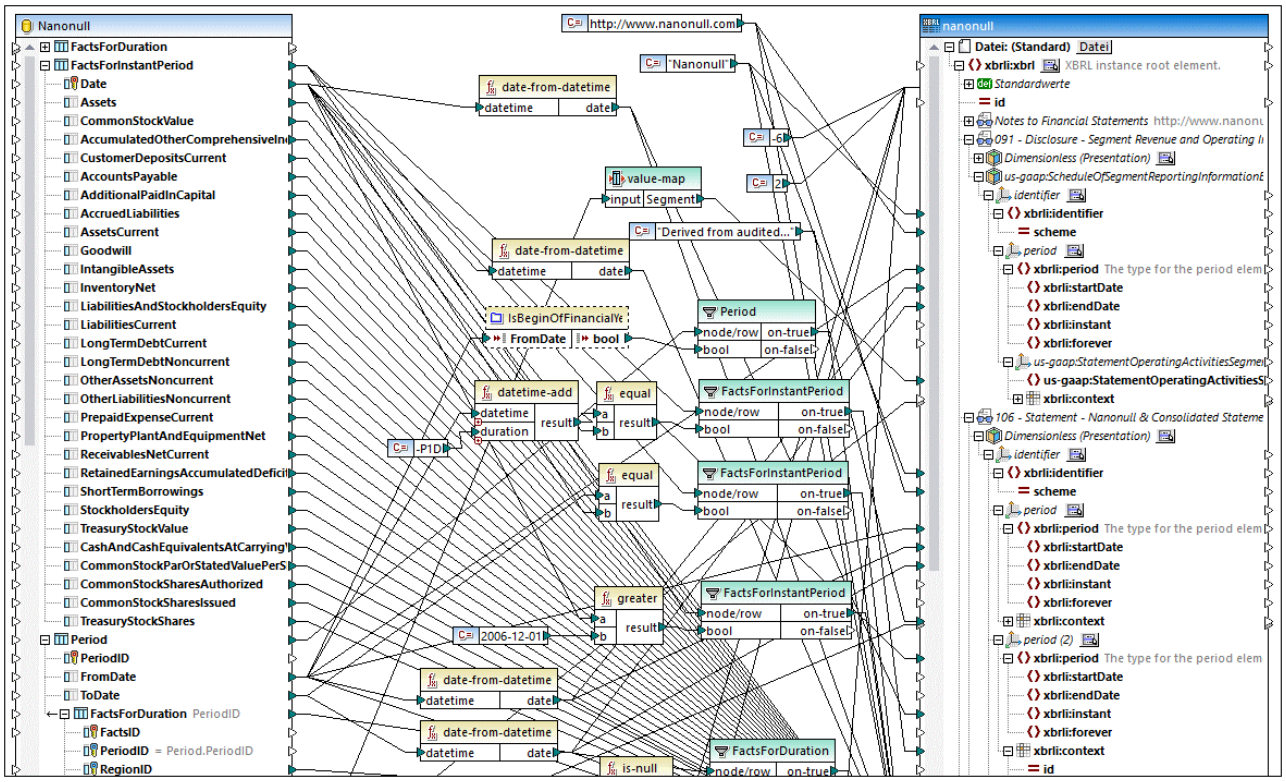
Im Mapping sind in der Output-Instanz zwei Units definiert: "USD" und "perShareItemType". Das xbrli:unit Element muss zu diesem Zweck dupliziert werden. Die dazugehörigen measure-Elemente werden mit Hilfe der Funktionen "xbrl-measure-currency" und "xbrl-measure-shares" anhand der XBRL-Bibliothek erstellt.

Die Facts in den Datenbanktabellen wurden aufgeteilt, je nachdem, ob sie sich auf eine Instant oder Duration Period beziehen.

- Die Tabelle **FactsForInstantPeriod** ist eine flache (nicht hierarchische) Wertetabelle.
- Die Tabelle **FactsForDuration** ist hierarchisch und jedes Fact, das darin enthalten ist, bezieht sich auf eine bestimmte PeriodID sowie eine RegionID.



In der Period-Tabelle werden die Daten der Start und End Period anhand der Felder FromDate und ToDate definiert, während sich in der Region-Tabelle jedes der Facts auf eine bestimmte Region bezieht, also Asia, Europe oder USA.



Mapping DB_to_XBRL.mfd

Obligatorische XBRL Items, die in einer XBRL-Instanzdatei benötigt werden:

- **unitRef** und entweder **decimals** oder **precision** in Monetary Concepts.
- xbrli:**identifier** und **scheme** der Identifier Dimension
- xbrl:**period** und entweder xbrli:**instant** oder die Elemente xbrli:**startDate**/xbrli:**endDate**
- xbrli:**id** und xbrli:**measure** im Element xbrli:**unit**

Siehe auch Microsoft Excel auf XBRL [818].

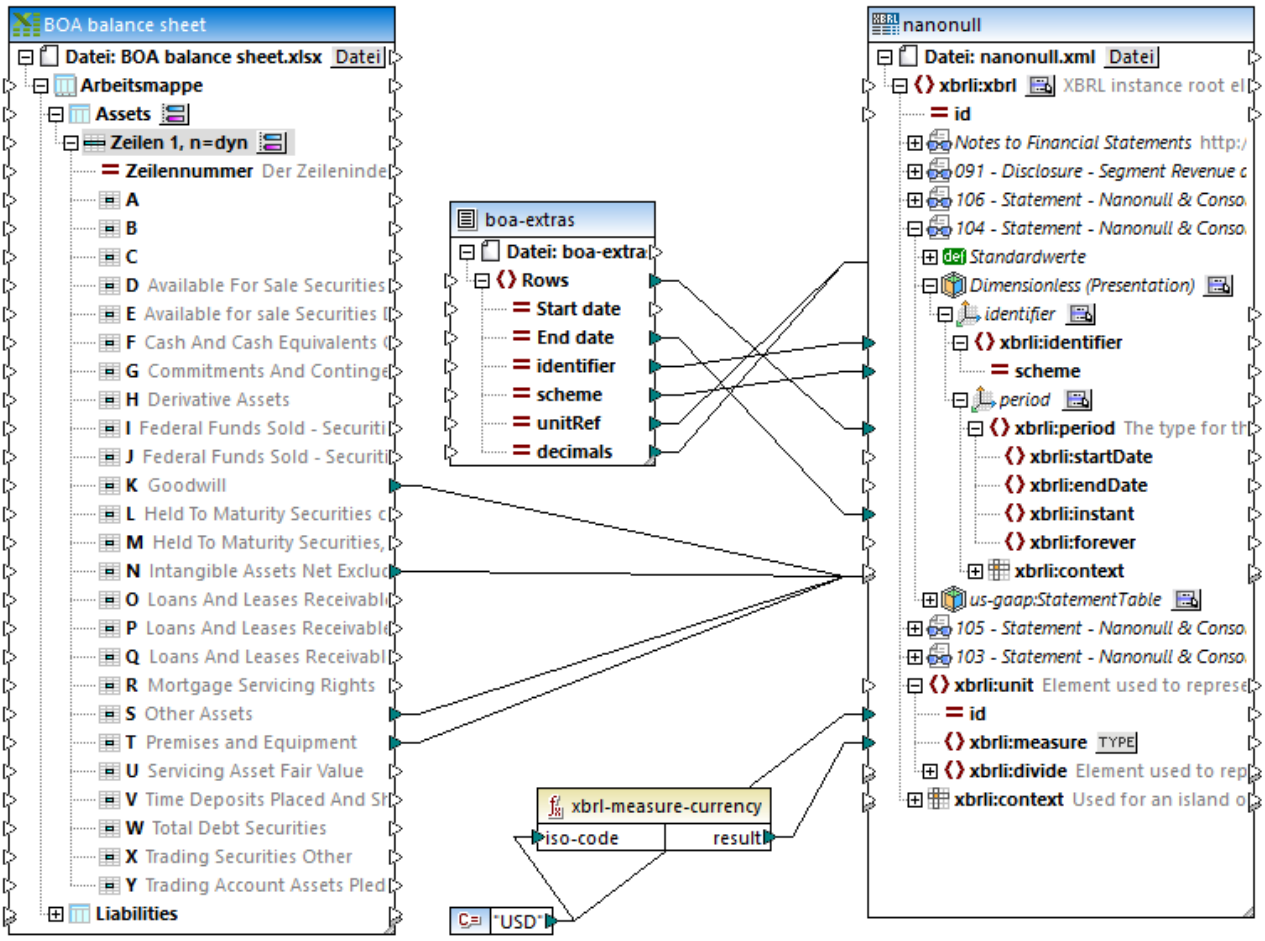
4.8.10.2 Microsoft Excel auf XBRL

In diesem Beispiel sehen Sie, wie Daten aus einem Excel-Arbeitsblatt auf eine XBRL-Taxonomie gemappt werden, um eine XBRL-Instanzdatei zu generieren. Die verwendete Mapping-Datei finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\boa-balance-sheet.mfd**.

Um das Beispiel einfach zu halten, wurden nur bestimmte Spalten aus dem Arbeitsblatt "Assets" gemappt. Das Ergebnis des Mappings ist ein gültiges XBRL-Instanzdokument, das einige Assets-Daten für eine bestimmte Instanz, den 31. Dezember 2012 enthält.

© 2016-2022 Altova GmbH

Die XBRL-Taxonomie für die Zielkomponente steht unter dem folgenden Pfad zur Verfügung: **<Dokumente>\Altova\MapForce2023\MapForceExamples\\Tutorial\Taxonomy\Nanonull.xsd**. Es handelt sich hierbei um eine absichtlich vereinfachte Demo-Taxonomie, die im Gegensatz zu einer echten Taxonomie nur wenige mapbare XBRL-Datenelemente enthält. Für die XBRL-Zielkomponente wurden für die Ansicht die Presentation und Definition Linkbases (Hypercubes) definiert, siehe Auswählen von Strukturansichten 770.



boa-balance-sheet.mfd Mapping

Die folgenden Excel-Spalten wurden direkt auf die XBRL Items/Facts in der Taxonomiekomponente gemappt: "Goodwill", "Intangible Assets Net Excluding Goodwill", "Other Assets" und "Premises and Equipment".

Beachten Sie, dass die Namen der Datenelemente in der Quell- und Zielkomponente nicht identisch sind und auch nicht identisch sein müssen. In der erzeugten XBRL-Instanzdatei wird als Name der Name der Zieltaxonomie verwendet. Bei identischen Namen des Quell- und Zieldatenelements haben Sie allerdings den Vorteil, dass Sie die Funktion **Idente Sub-Einträge automatisch verbinden** verwenden können (siehe Idente Sub-Einträge verbinden 96).

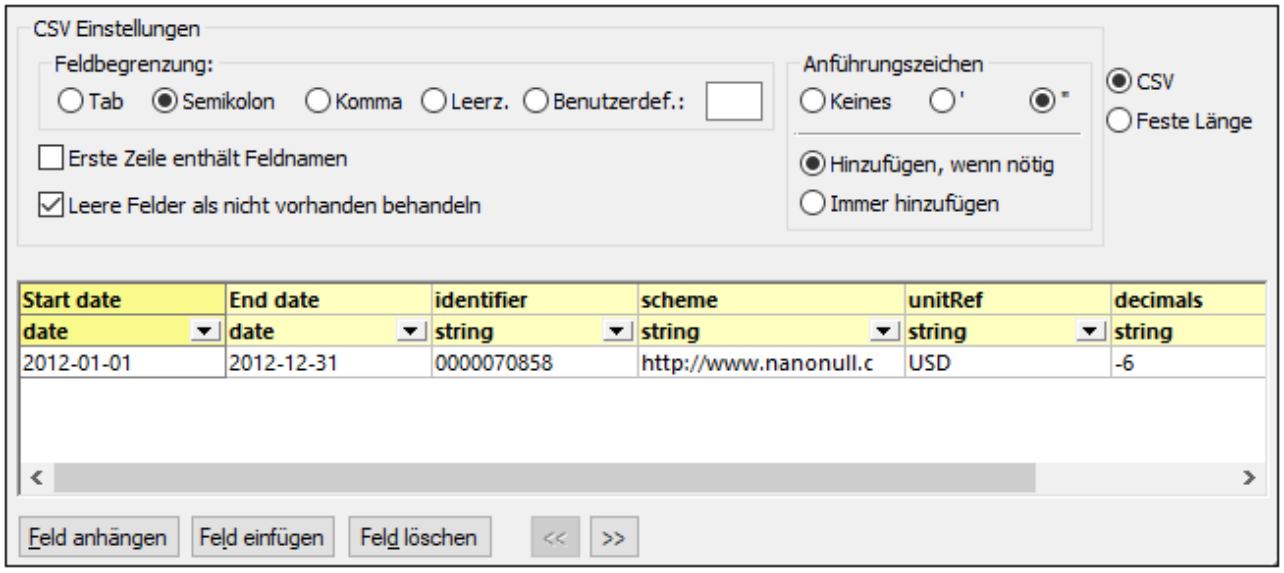
Um eine gültige XBRL-Instanzdatei generieren zu können, müssen einige obligatorische Datenelemente gemappt werden. Einige obligatorische Datenelemente (Items) stammen aus einer Textdatei (boa-extras.txt, die mittlere Komponente des Mappings), deren Felder auf Sub-Elemente des Datenelements "**Standardwerte**" im **Dimensionless (Presentation) Hypercube** gemappt werden.

Das Datenelement "**Standardwerte**" wird durch Rechtsklick auf eine Hypercube Dimension und Auswahl des Befehls **XBRL | Standardwerte anzeigen** oder durch Klicken auf das Symbol und Auswahl der entsprechenden Option eingefügt. Nähere Informationen dazu finden Sie unter Arbeiten mit XBRL-Standardwerten 797.

Die folgenden XBRL-Items werden gemappt, um eine gültige XBRL-Instanzdatei zu generieren:

- **unitRef** und entweder **decimals** oder **precision** in Monetary Concepts.
- xbrli:**identifier** und **scheme** der Identifier Dimension
- xbrl:**period** und entweder xbrli:**instant** oder die Elemente xbrli:**startDate**/xbrli:**endDate**
- xbrli:**id** und xbrli:**measure** im Element xbrli:**unit**

Aus der Textdatei "boa-extras" stammen die Daten für einige der obligatorischen Datenelemente. Durch Rechtsklick auf die Komponente "boa-extras" und Auswahl des Befehls **Eigenschaften** aus dem Kontextmenü können Sie diese Werte anzeigen oder ändern.



Die Daten für die Datenelemente "**id**" und "**measure**", die sich an der Basis der Taxonomiekomponente befinden, werden durch eine Konstantenkomponente geliefert. Die Funktion `xbrl-measure-currency` konvertierte den Währungscode in einen für das Feld **xbrli:measure** geeigneten Wert. Nähere Informationen zu MapForce-Funktionen finden Sie im Kapitel Funktionen 942.

Klicken Sie auf die Schaltfläche **Ausgabe** klicken, um das Mapping auszuführen und eine Vorschau auf die erzeugte XBRL-Instanzdatei zu sehen.

- Durch Klicken auf die Schaltfläche **Ausgabe validieren** in der Ausgabesymbolleiste, können Sie die Gültigkeit der XBRL-Instanz überprüfen. Meldungen oder Warnungen werden im Fenster "Meldungen" angezeigt.
- Durch Klicken auf die Schaltfläche **Einstellungen für Textansicht** können Sie verschiedene Einstellungen für die generierte Ausgabe definieren.

© 2016-2022 Altova GmbH

4.9 JSON

Altova Website: JSON-Mapping

JSON (JavaScript Object Notation) ist ein beliebtes Datenformat zum Speichern und Transportieren von Daten. In der in MapForce Enterprise Edition können Sie Daten von jedem von MapForce unterstützten Form von und auf JSON mappen. JSON-Daten können auch als String serialisiert werden (z.B. um diese in eine Datenbank zu schreiben). Außerdem kann ein JSON-String mit Hilfe einer Parse/Serialise-Komponente zu einer JSON-Struktur geparst werden. Nähere Informationen dazu finden Sie unter Parsen und Serialisieren von Strings 1382.

Um JSON-Dateien in Mappings verwenden zu können, muss als Mapping-Sprache eine der folgenden eingestellt sein: Built-in, C#, Java. Nähere Informationen dazu finden Sie unter Auswählen einer Transformationssprache 28.

Ähnlich wie bei anderen von MapForce unterstützten Komponententypen wird eine Schemadatei benötigt, damit MapForce JSON-Daten auslesen oder schreiben kann. Wenn Sie keine Schemadatei zur Verfügung haben, kann MapForce automatisch eine generieren., siehe Hinzufügen von JSON-Dateien als Mapping-Komponenten 823. Wenn Sie eine nicht mit MapForce generierte JSON-Schemadatei bereitstellen, sollte diese eine Draft 04-, Draft 06-, Draft 07-, Draft 2019-09, Draft 2020-12-Version sein oder gar keine Versionsnummer haben. Nähere Informationen zu JSON-Schema-Versionen finden Sie in der JSON-Schema-Spezifikation.

Im Folgenden sehen Sie ein Beispiel dafür, wie eine JSON-Struktur in MapForce angezeigt wird:



JSON-Komponente in MapForce

Wie Sie unten sehen, entspricht die Struktur der Nodes in der JSON-Komponente der Struktur der JSON-Instanzdatei. Die JSON-Basistypen werden in MapForce normalerweise folgendermaßen dargestellt:

Typ	MapForce-Darstellung
Array	[]
Boolean	01
Null	∅
Zahl oder Ganzzahl	#

Typ	MapForce-Darstellung
Objekt	{ }
String	"AB"

Außerdem gibt es die folgenden Sonderfälle:

Fall	Beschreibung
zusätzliche Eigenschaften, Pattern-Eigenschaften	In einer JSON-Komponente erscheint der Node unter jedem Objekt, dessen Eigenschaft `additionalProperties` `true` lautet oder im Schema nicht vorhanden ist. Damit können Sie von oder auf im Schema nicht explizit aufgelistete Eigenschaften mappen (siehe auch Beispielmapping von JSON auf CSV 832). Dieser Node kann auch für Objekte mit der Eigenschaft `patternProperties` angezeigt werden.
Subtypen	Laut dem JSON-Schema dürfen für Objekte und Arrays (`anyOf`, `allOf`, `oneOf`) Subtypen definiert werden. In MapForce werden solche Subtypen mit Hilfe spezieller Struktur-Nodes ( ), die in der JSON-Instanzdatei keine direkte Entsprechung haben, angezeigt. Wenn in Ihrem Mapping Daten aus einer JSON-Datei gelesen werden, stellen solche Subtypen-Nodes nur dann einen Wert zur Verfügung, wenn der aktuelle Input-Wert entsprechend dem Subtyp-Schema gültig ist. Wenn Ihr Mapping Daten in eine JSON-Datei schreibt, stellen Sie sicher, dass Sie den richtigen Subtyp ausgewählt haben. Wenn mehrere Subtypen befüllt werden, können Objekteigenschaften doppelt vorhanden sein, wodurch ungültige JSON-Ausgabedateien erzeugt werden können.
Mehrere Typen unter demselben Pfad	Gemäß einem JSON-Schema dürfen mehrere Typen unter demselben Pfad vorkommen. In solchen Fällen werden in der MapForce-Komponente für alle Basistypen, die unter dem Pfad vorkommen, separate Struktur-Nodes angezeigt.
Typnamen	In MapForce werden die Eigenschaften `title` und `description` von Typen im JSON-Schema (falls vorhanden) in den Feldern "type" und "annotation" angezeigt. Wenn `title` fehlt, kann MapForce auch einen Teil der URI aus der Eigenschaft `$ref` als Typnamen verwenden.
Arrays, die gemischte Elementtypen enthalten	Wenn ein Array im JSON-Schema unterschiedliche Datenelementtypen hat (z.B. Strings und Zahlen), zeigt MapForce für jedes Datenelement einen "Item"-Node an. Wenn die Daten in eine JSON-Datei geschrieben werden, können Sie dadurch Arrays erstellen, die Datenelemente unterschiedlicher Typen enthalten.
Als Tuples definierte Arrays	Wenn ein Array Datenelemente hat, deren Typ nach der Position im JSON-Schema zugewiesen wird, zeigt MapForce den nullbasierten Index das Datenelements als separate Struktur-Nodes an (z.B. **item[0]**, **item[1]** usw.

© 2016-2022 Altova GmbH

Fall	Beschreibung
	Beim Schreiben in eine JSON-Datei können Sie dadurch den Typ der einzelnen Datenelemente im Array anhand ihres nullbasierten Index definieren.

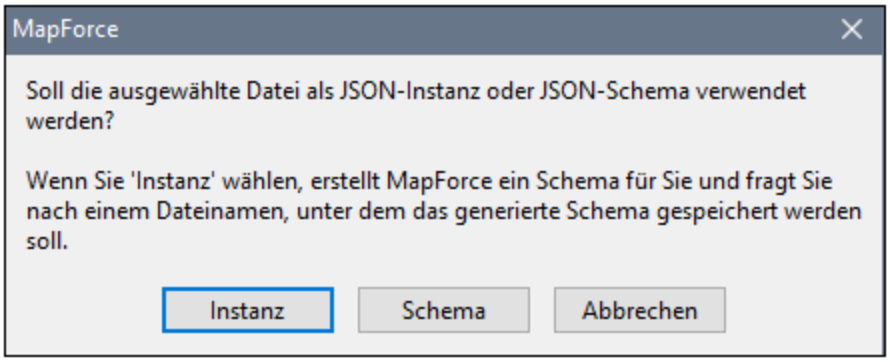
4.9.1 Hinzufügen von JSON-Dateien als Mapping-Komponenten

Um JSON-Instanzdateien mit MapForce auslesen oder schreiben zu können, wird eine JSON-Schemadatei benötigt. Sie können die JSON-Schemadatei entweder mit MapForce generieren oder, wie unten gezeigt, in Form einer benutzerdefinierten Datei bereitstellen.

Zum Zeitpunkt der Verfassung dieser Dokumentation gibt es keine offizielle Konvention zur Benennen von JSON-Schemadateien. Um daher JSON-Schemadateien von Instanzdateien zu unterscheiden, können Sie für Ihre JSON-Schemadateien die Erweiterung **schema.json** verwenden (z.B. **Beispiel.schema.json**) und MapForce erkennt diese als solche.

So fügen Sie JSON-Dateien als Mapping-Komponenten hinzu

1. Wählen Sie im Menü **Einfügen** die Option **JSON-Schema/Datei** aus. Daraufhin wird eventuell ein Dialogfeld geöffnet, in dem Sie den Typ der JSON-Datei (Schema oder Instanz) auswählen können. Dieses Dialogfeld wird nicht angezeigt, wenn MapForce die Datei definitiv als JSON-Schema identifiziert.



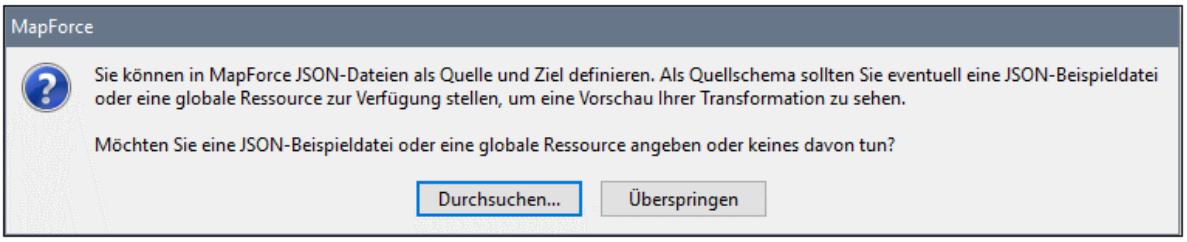
2. Wählen Sie je nach Bedarf **Schema** oder **Instanz** aus.

   Die Größe der JSON-Instanzdatei, darf den in Ihrem System verfügbaren Arbeitsspeicher nicht übersteigen.

   Bei Auswahl von **Instanz** generiert MapForce automatisch anhand der Instanzdatei ein Schema und fordert Sie auf, einen Pfad anzugeben, unter dem das Schema gespeichert werden soll.

   Wenn Schema auswählen, werden Sie aufgefordert, eine JSON-Beispieldatei oder eine globale Ressource (siehe Globale Ressourcen 1462) anzugeben. Sie benötigen diese, um eine Vorschau der Transformation anzeigen zu können. Dies ist nur dann sinnvoll, wenn Daten aus einer JSON-Datei

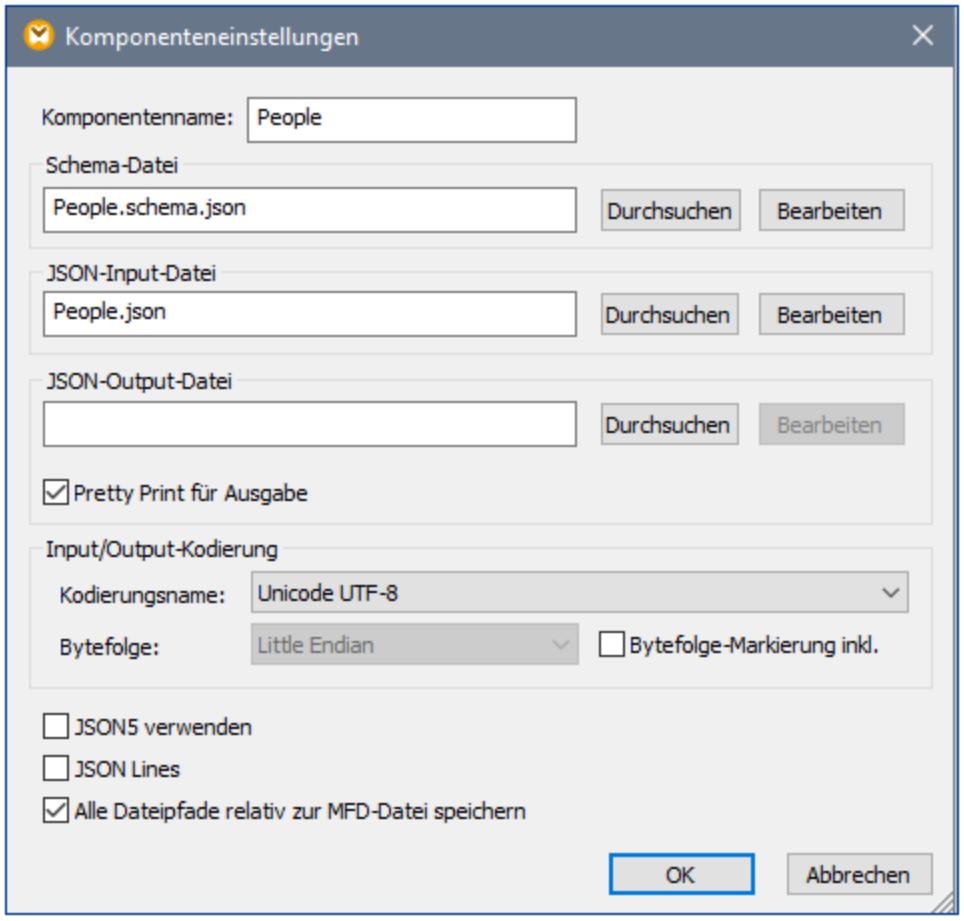
ausgelesen werden sollen. Klicken Sie auf **Überspringen**, wenn in Ihrem Mapping Daten in JSON geschrieben werden sollen.



© 2016-2022 Altova GmbH

4.9.2 JSON-Komponenteneinstellungen

Um die Einstellungen einer JSON-Komponente zu ändern, klicken Sie mit der rechten Maustaste auf die Überschrift der JSON-Komponente und klicken Sie anschließend auf **Eigenschaften** (Doppelklicken Sie alternativ dazu auf die Überschrift der JSON-Komponente).



Dialogfeld für JSON-Komponenteneinstellungen

Es stehen die folgenden Einstellungen zur Verfügung:

Komponentenname	Der Komponentenname wird bei der Erstellung der Komponente automatisch generiert. Sie können den Namen jedoch jederzeit ändern. Der Komponentenname kann Leerzeichen und Punkte enthalten, darf aber keine Schrägstriche, umgekehrten Schrägstriche, Doppelpunkte, doppelten Anführungszeichen und voran- oder nachgestellte Leerzeichen

	enthalten. Beachten Sie beim Ändern des Namens einer Komponente Folgendes: • Wenn Sie das Mapping auf FlowForce Server bereitstellen möchten, muss der Komponentenname eindeutig sein. • Es wird empfohlen, nur Zeichen zu verwenden, die über die Befehlszeile eingegeben werden können. Länderspezifische Sonderzeichen sind in Windows und der Befehlszeile eventuell unterschiedlich kodiert.
Schema-Datei	Definiert den Namen und Pfad der Schema-Datei. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie die neue Datei aus. Um die Datei in Ihrem JSON-Editor (z.B. XMLSpy) zu bearbeiten, klicken sie auf **Bearbeiten.**
JSON-Input-Datei	Dieses Textfeld ist nur dann von Bedeutung, wenn Daten aus einer JSON-Datei ausgelesen werden. Darin wird der Name und Pfad der JSON-Input-Instanz definiert. Um den Pfad der Datei zu definieren oder zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie die neue Datei aus. Sie können auch einen relativen Dateipfad eingeben, wenn das Kontrollkästchen **Alle Dateipfade relativ zur MFD-Datei speichern** aktiviert ist (siehe unten). Um die Datei in Ihrem JSON-Editor (z.B. XMLSpy) zu bearbeiten, klicken sie auf **Bearbeiten.**
JSON-Output-Datei	Dieses Textfeld ist nur dann von Bedeutung, wenn Daten in eine JSON-Datei geschrieben werden. Darin wird der Name und Pfad, unter dem die JSON-Instanzzieldatei gespeichert wird, wenn das Mapping von MapForce Server ausgeführt wird, definiert. Dies ist auch der Standardpfad, unter dem die Ausgabe vom Register "Ausgabe" gespeichert wird. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie die neue Datei aus. Sie können auch einen relativen Dateipfad eingeben, wenn das Kontrollkästchen **Alle Dateipfade relativ zur MFD-Datei speichern** aktiviert ist (siehe unten). Um die Datei in Ihrem JSON-Editor (z.B. XMLSpy) zu bearbeiten, klicken sie auf **Bearbeiten.**
Pretty Print für Ausgabe	Formatiert das JSON-Dokument bei Ausführung des Mappings neu, um das Dokument in strukturierter Form anzuzeigen. Jeder Child Node wird unterhalb seines Parent-Node um einen Tabulatorabstand eingerückt.
Digitale Signatur erstellen	Damit können Sie eine digitale Signatur zur XBRL-Outputinstanzdatei hinzufügen. Digitale Signaturen können bei Auswahl der

© 2016-2022 Altova GmbH

	Transformationssprache "Built-in" hinzugefügt werden (siehe auch Digitale Signaturen[143]).
Kodierung	Damit können Sie die folgenden Einstellungen der Output-Instanzdatei definieren: • Kodierungsname • Bytefolge • Ob das Bytefolgemarkierungszeichen (BOM) inkludiert werden soll. JSON-Dateien haben standardmäßig die UTF-Kodierung (siehe https://tools.ietf.org/html/rfc7159#section-8.1). Andere Kodierungen werden als nicht-standardmäßig erachtet.
JSON5 verwenden	Aktivieren Sie dieses Kontrollkästchen, wenn in der JSON-Input-Datei, aus der die Daten ausgelesen werden, die JSON5-Syntax verwendet wird. Wenn Sie das Kontrollkästchen in einer Zielkomponente aktivieren, erstellt MapForce eine .json5-Datei, in der Objektschlüssel nicht innerhalb von Anführungszeichen stehen, wenn es sich um gültige ECMAScript5-Identifier handelt. Siehe auch JSON5-Unterstützung[827].
JSON Lines	Aktivieren Sie dieses Kontrollkästchen, wenn Sie Daten aus einer Quelldatei im JSON Lines-Format auslesen oder wenn Sie Daten in eine Zieldatei im JSON Lines-Format speichern möchten. Nähere Informationen dazu finden Sie unter JSON Lines[829].
Alle Dateipfade relativ zur MFD-Datei speichern	Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld "Komponenteneinstellungen" angezeigten Pfade relativ zum Ordner der MapForce Design (.mfd) Datei. Diese Einstellung wirkt sich auf die folgenden Dateien aus: • die JSON-Schemadatei • die JSON-Input-Datei • die JSON-Output-Datei Siehe auch Verwenden relativer Pfade in einer Komponente[85].

4.9.3 JSON5-Unterstützung

JSON5 ist eine vorgeschlagene Erweiterung für JSON, die das händische Verfassen von JSON-Dateien erleichtern soll. Beachten Sie, dass JSON5 nicht der offizielle Nachfolgestandard von JSON ist (Nähere Informationen dazu siehe https://json5.org/).

Für JSON5-Dateien wird die Erweiterung .json5 verwendet; zum Zeitpunkt der Verfassung dieser Dokumentation gibt es jedoch keinen offiziellen MIME-Typ für .json5-Dateien. MapForce erkennt die Erweiterung .json5 jedoch, wenn Sie solche Dateien zum Mapping hinzufügen. Wenn Sie das Mapping auf FlowForce Server bereitstellen (siehe Bereitstellen von Mappings auf FlowForce Server[1445]), wird der MIME-Typ von .json5-Dateien als `"application/json5"` angegeben.

Voraussetzungen

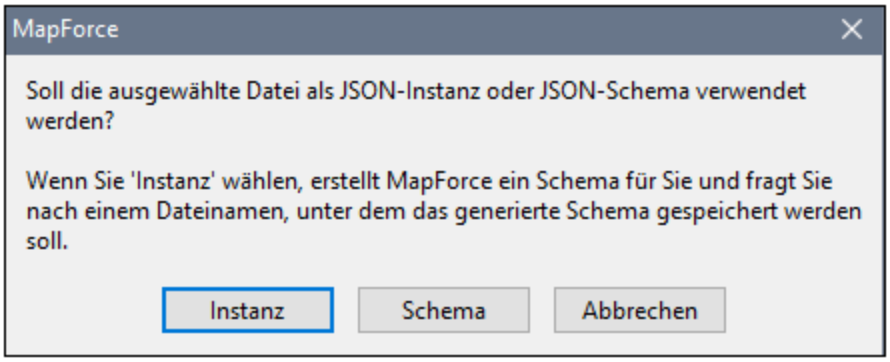
- Um Daten von/auf JSON5-Dateien mappen zu können, muss ein gültiges JSON-Schema (in dem keine JSON5-Syntax verwendet werden sollte) in MapForce bereitgestellt werden. Wenn Sie kein JSON-Schema für Ihre Datei haben, generiert MapForce automatisch eines, wie unten gezeigt. Beim von MapForce automatisch generierten Schema handelt es sich um ein JSON-Schema (nicht ein JSON5-Schema). Auch mit XMLSpy kann ein JSON-Schema für eine .json5-Datei generiert werden kann.

Einschränkungen

- Das Mappen von Kommentaren von und auf JSON5-Dateien wird nicht unterstützt.
- Da MapForce mit strengen Datentypen arbeitet, können Werte wie `Infinity`, `-Infinity`, `NaN` oder `-NaN`. nicht qualifiziert oder gemappt werden.

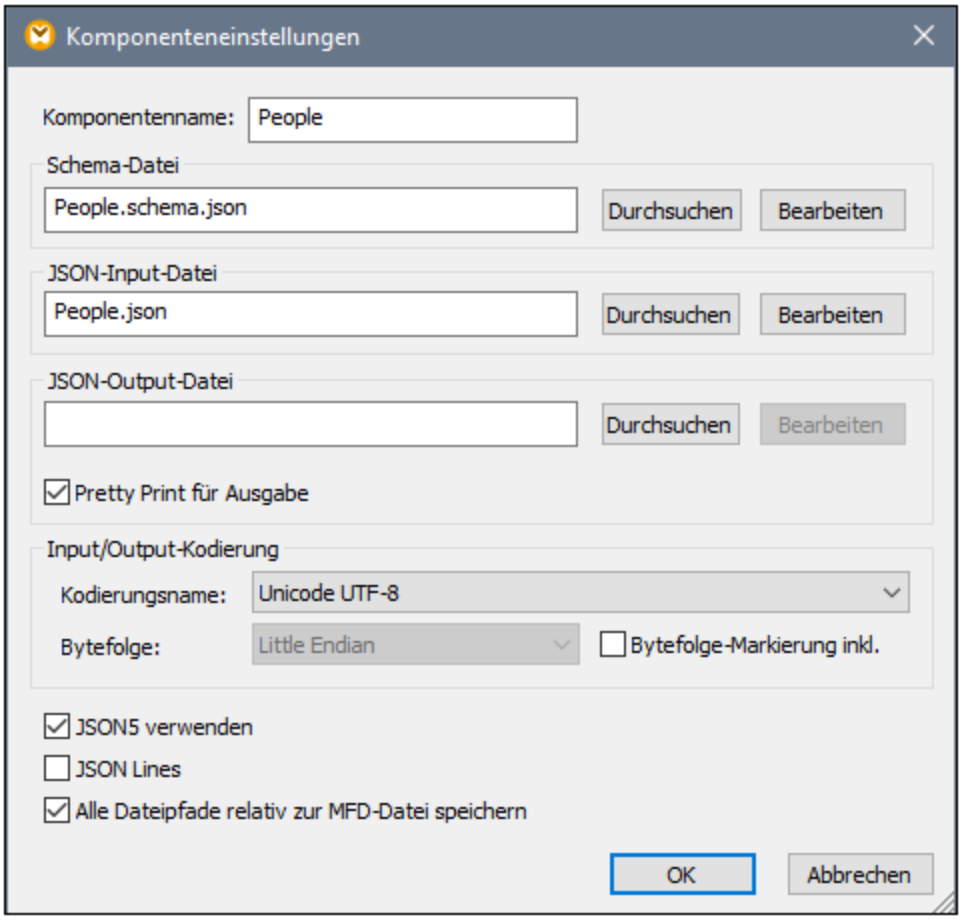
So mappen Sie Daten von oder auf .json5-Dateien

1. Klicken Sie im Menü **Einfügen** auf **JSON** und navigieren Sie zur .json5-Instanzdatei. (Klicken sie alternativ dazu auf die Symbolleisten-Schaltfläche ). Wenn Sie eine JSON-Schema-Datei haben und die .json5-Instanzdatei erst später angeben möchten, navigieren Sie zu einem gültigen JSON-Schema (nicht JSON5-Schema). Gehen Sie dabei vor, wie bei einfachen JSON-Dateien (siehe Hinzufügen von JSON-Dateien als Mapping-Komponenten 823).



2. Klicken Sie nach dem Hinzufügen der JSON-Komponente mit der rechten Maustaste auf ihre Titelleiste, wählen Sie **Eigenschaften** und aktivieren Sie anschließend in den Komponenteneigenschaften das Kontrollkästchen **JSON5 verwenden.** (siehe auch JSON-Komponenteneinstellungen 825). Damit weisen Sie MapForce explizit an, die Instanzdaten als JSON5-Daten zu behandeln.

© 2016-2022 Altova GmbH



Um eine JSON5-Datei mit MapForce zu generieren, aktivieren Sie das Kontrollkästchen **JSON5 verwenden** (siehe Abb. oben) und verbinden Sie die erforderlichen Input-Daten mit der JSON-Komponente im Mapping. Wenn das Kontrollkästchen **JSON5 verwenden** aktiviert ist, setzt MapForce die Objektschlüssel mit Ausnahme von Fällen, in denen die generierte JSON5-Instanz sonst ungültig würde, nicht in Anführungszeichen. Welche Dateierweiterung verwendet werden soll (z.B. "json5") kann ebenfalls über das oben gezeigte Dialogfeld definiert werden, und zwar durch Eingabe des entsprechenden Werts in das Feld **JSON-Output-Datei**.

4.9.4 JSON Lines

JSON Lines (http://jsonlines.org/) ist ein Textformat, das eine Erweiterung zum JSON-Format darstellt. Es eignet sich zum Speichern und Verarbeiten von JSON-Daten, bei denen "Neue Zeile"-Trennzeichen von Bedeutung sind, wie z.B. Log-Dateien. In jeder Zeile einer JSON Lines-Datei ist ein JSON-Wert gespeichert, z.B:

```
{"name":"Alethia Alonso","email":"alethia@example.org"}
{"name":"Klaus Mauer","email":"klaus@example.org"}
{"name":"Natsuo Shinohara","email":"natsuo@example.org"}
```

Mit der MapForce Enterprise Edition können Sie Daten im JSON Lines-Format auslesen und in jedes beliebige andere von MapForce unterstützte Format konvertieren. Ebenso können Sie Daten aus jedem beliebigen von MapForce unterstützten Format in das JSON Lines-Format konvertieren.

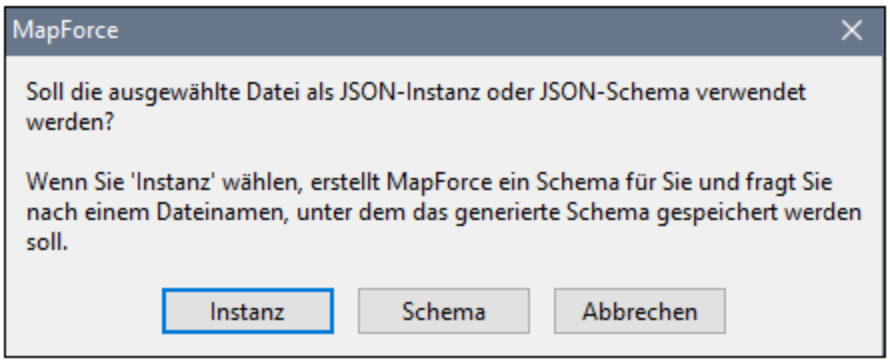
Anmerkung: In mit MapForce erstellten Webservice-Aufrufen können Sie als Request- oder Response-Struktur Standard-JSON verwenden; JSON Lines- oder JSON5-Strukturen werden nicht unterstützt.

Um Daten im JSON Lines-Format auszulesen oder diese in dieses Format zu schreiben, wird in MapForce wie bei normalen JSON-Dateien ein JSON-Schema benötigt. Der Unterschied besteht darin, dass bei JSON Lines jede Zeile zu Validierungszwecken als separate JSON-Datei behandelt wird. Das Schema, das Sie in MapForce bereitstellen, validiert daher jede einzelne Zeile als JSON-Dokument.

Die Entwickler des JSON Lines-Formats schlagen als Dateierweiterung dafür **.jsonl** vor.

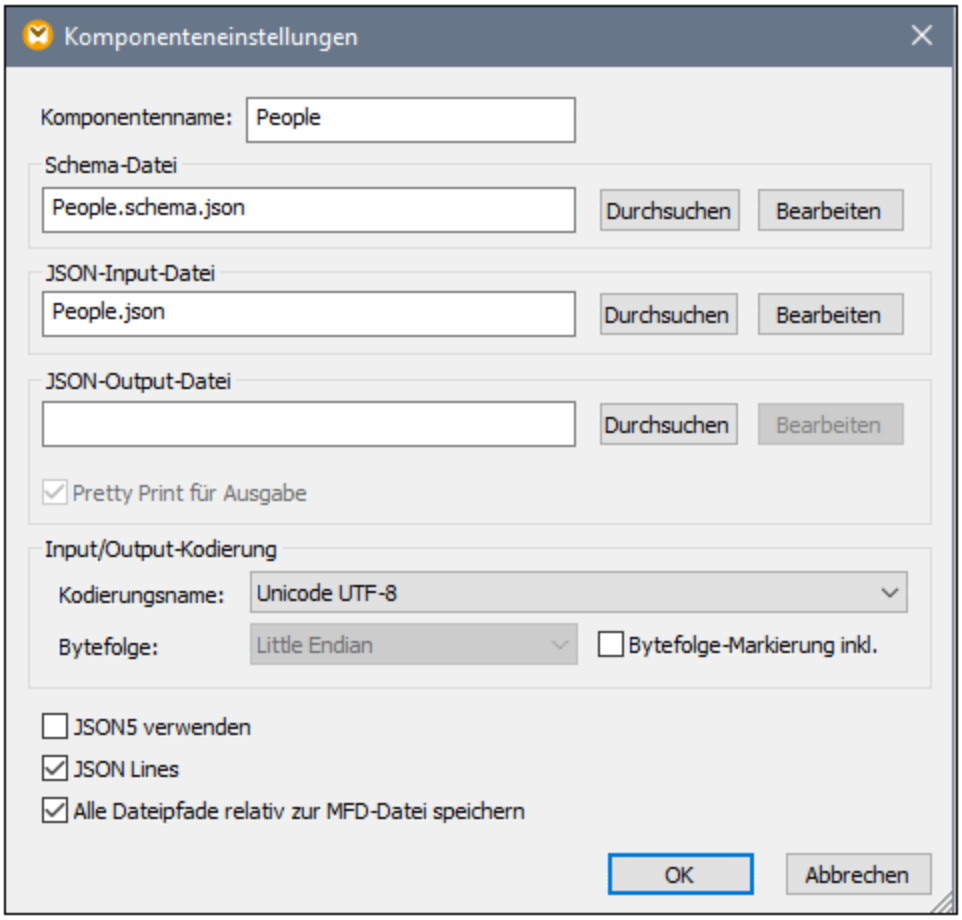
So mappen Sie Daten von oder auf das JSON Lines-Format

1. Klicken Sie im Menü **Einfügen** auf **JSON** und navigieren Sie zur **.jsonl**-Instanzdatei. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche . Wenn Sie bereits eine JSON-Schema-Datei haben und die Instanzdatei erst später angeben möchten, navigieren Sie zu einem gültigen JSON-Schema. Dies funktioniert auf dieselbe Art und Weise wie bei einfachen JSON-Dateien (siehe Hinzufügen von JSON-Dateien als Mapping-Komponenten 823). Denken Sie daran, dass im JSON-Schema für das JSON Lines-Format der Inhalt jeder einzelnen Zeile in der Instanzdatei und nicht die Datei als ganze beschrieben sein muss.



2. Nachdem Sie die JSON-Komponente zum Mapping hinzugefügt haben, klicken Sie mit der rechten Maustaste auf ihre Titelleiste, wählen Sie **Eigenschaften** und aktivieren Sie anschließend das Kontrollkästchen **JSON Lines**.

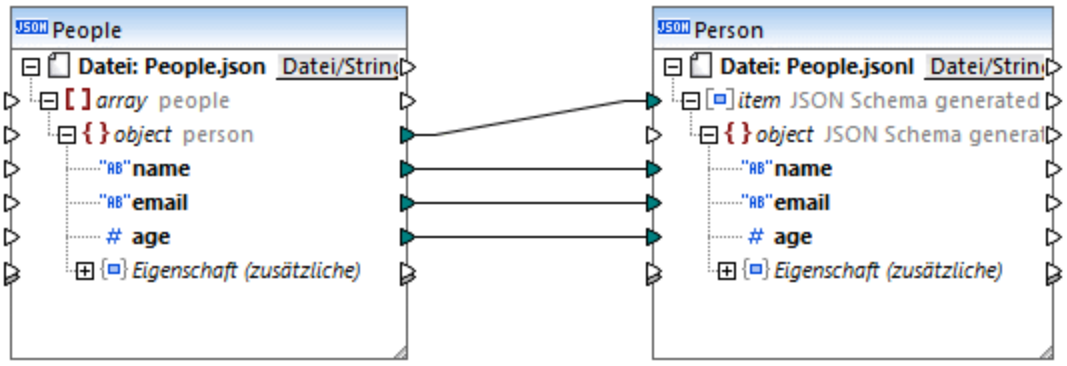
© 2016-2022 Altova GmbH



Um die JSON-Instanzdatei, aus der die Daten ausgelesen werden, zu definieren oder zu wechseln, geben Sie den Pfad in das Textfeld **JSON-Input-Datei** ein. Dieses Textfeld muss nicht ausgefüllt werden, wenn Daten im Mapping in eine JSON-Datei geschrieben werden.

Um festzulegen, welcher Dateiname generiert wird, geben Sie in das Textfeld **JSON-Output-Datei** des obigen Dialogfelds den gewünschten Dateinamen (z.B. "Dateiname.jsonl") ein. Dieses Textfeld muss nicht ausgefüllt werden, wenn Daten im Mapping aus einer JSON-Datei ausgelesen werden. Nähere Informationen zu weiteren Einstellungen finden Sie unter JSON-Komponenteneinstellungen 825.

Wenn das Kontrollkästchen **JSON Lines** aktiviert ist, hat die JSON-Komponente im Mapping einen neuen item-Node. Durch Verbinden einer beliebigen Reihe von Werten damit (z.B. die **Rows**-Datenelemente einer CSV-Quellkomponente) können Sie in der JSON Lines-Datei mehrere Zeilen erstellen (für jedes Datenelement in der Sequenz eine). Die Quelldatei im unten gezeigten Mapping ist eine JSON-Standarddatei und die Zielkomponente ist eine JSON Lines-Datei. Aufgrund der obersten Verbindung wird bei der Ausführung des Mappings für jedes Objekt in der Quelldatei **People.json** in der Zieldatei **People.jsonl** eine neue JSON-Zeile erstellt.



Wenn in Ihrem Mapping Daten aus einer JSON Lines-Datei ausgelesen werden sollen, können Sie den item-Node mit dem gewünschten Datenelement in der Zielkomponente verbinden (z.B. einem XML-Datenelement). Dadurch würden in der Zieldatei so viele XML-Elemente erstellt, wie in der Quelldatei JSON-Zeilen vorhanden sind.

4.9.5 Beispiel: Konvertieren von JSON in CSV

In diesem Beispiel wird gezeigt, wie Sie in MapForce Daten aus einer JSON-Datei auslesen und in eine kommagetrennte Textdatei konvertieren. Außerdem erfahren Sie darin, wie Sie Daten aus zusätzlichen in der JSON-Instanzdatei eventuell vorhandenen Eigenschaften, wie der in diesem Beispiel gezeigten, mappen. Mit zusätzlichen Eigenschaften sind Daten gemeint, die in der JSON-Instanzdatei vorhanden sind, obwohl Sie nicht durch das Schema definiert sind. In der Regel sollten keine zusätzlichen Eigenschaften verwendet werden. Stattdessen sollten alle Eigenschaften in Ihrem JSON-Schema definiert werden. Falls die JSON-Instanzdatei jedoch zusätzliche Eigenschaften enthält, so gehen Sie vor, wie in diesem Beispiel beschrieben.

Die Demo-Mapping-Datei zu diesem Beispiel finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\ReadJSON.mfd**. Wenn Sie vorgehen, wie unten beschrieben, lernen Sie, wie Sie eine solche Mapping-Datei von Grund auf neu erstellen.

Die JSON-Quelldatei enthält Personendatensätze, die in CSV konvertiert werden sollen. Beachten Sie den markierten Text: Die erste und zweite Person haben eine zusätzliche Eigenschaft, die im Schema nicht definiert ist, nämlich `birthday`.

```
[
	{
		"name": "Alethia Alonso",
		"email": "altethia@example.com",
		"age": 35,
		"birthday": "4 July"
	}, {
		"name": "Klaus Mauer",
		"email": "klaus@example.com",
		"age": 57,
		"birthday": "31 August"
	}, {
		"name": "Natsuo Shinohara",
		"email": "natsuo@example.com",
```


© 2016-2022 Altova GmbH

```
            "age": 29
      }
]
```

People.json

Unten sehen Sie das im Beispiel verwendete JSON-Schema. Wie durch das Schlüsselwort `$schema` angegeben, validiert das Schema JSON-Instanzen anhand des Draft 04 JSON Schemas. Das Schema enthält einen Array `people`. der aus mehreren `person`-Objekten besteht. Das `people`-Objekt muss mindestens ein `person`-Objekt enthalten, um gültig zu sein. Jedes `person`-Objekt hat die Eigenschaften `name`, `age` und `email`. Beachten Sie, dass `name` und `email` vom Typ `string` sind, während `age` vom Typ `integer` ist. Außerdem sind die Eigenschaften `name` und `email` obligatorisch, während `age` optional ist.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "people",
  "type": "array",
  "items": {
       "title": "person",
       "type": "object",
    "required": [
      "name",
      "email"
    ],
    "properties": {
      "name": {
        "type": "string"
      },
      "email": {
        "type": "string",
        "format": "email"
      },
      "age": {
        "type": "integer"
      }
    }
  },
  "minItems": 1
}
```

People.schema.json

Beachten Sie, dass das obige JSON-Schema für das Objekt `person` keine Eigenschaft `additionalProperties` enthält. Wenn die Eigenschaft `additionalProperties` im JSON-Schema nicht vorhanden ist, hat sie den Standardwert **true**, d.h. das Objekt in der JSON-Instanzdatei kann beliebig viele zusätzliche Eigenschaften haben und ist dennoch gültig. Eine JSON-Instanzdatei wie **People.json** kann daher anhand dieses Schemas erfolgreich validiert werden.

Um das in diesem Beispiel gewünschte Mapping-Ergebnis zu erzielen, gehen Sie folgendermaßen vor:

Schritt 1: Hinzufügen der JSON-Quelldatei zum Mapping

1. Klicken Sie im Menü **Einfügen** auf **JSON Schema/Datei** und navigieren Sie im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** zur Datei

<Dokumente>\Altova\MapForce2023\MapForceExamples\People.schema.json. Wenn Sie nach einer Instanz gefragt werden, wählen Sie die Datei **People.json** im selben Ordner aus. Die MapForce-Komponente sieht zu diesem Zeitpunkt folgendermaßen aus:

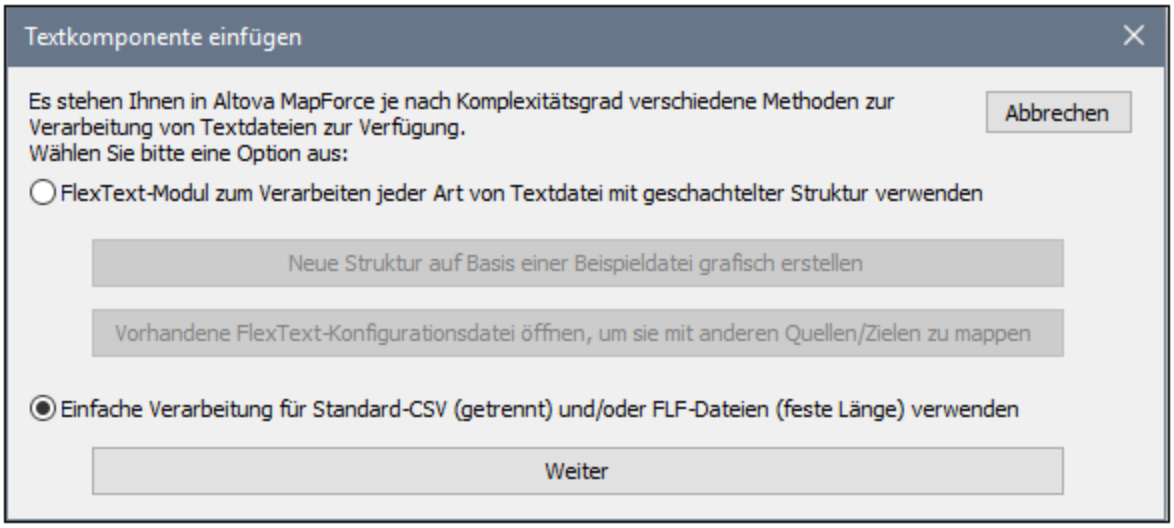


Die Struktur der MapForce-Komponente ähnelt - mit Ausnahme des Node **Eigenschaft (zusätzliche)** - der JSON-Datei selbst. Dieser Node zeigt an, dass die Eigenschaft `additionalProperties` des Objekts `person` entweder fehlt oder im Schema auf **true** gesetzt ist. Dadurch kann das Schema zusätzliche benutzerdefinierte Eigenschaften enthalten, so dass der Node in MapForce angezeigt wird, falls Sie von einigen der zusätzlichen Eigenschaften des Objekts mappen möchten (eine Beschreibung dazu finden Sie im nächsten Schritt).

Sie können die JSON-Quelldatei oder die JSON-Schemadatei jederzeit ändern, indem Sie auf die Titelleiste der JSON-Komponente doppelklicken und die Einstellungen anpassen, siehe JSON-Komponenteneinstellungen 825
.

Schritt 2. Hinzufügen der CSV-Zieldatei zum Mapping

1. Klicken Sie im Menü **Einfügen** auf **Textdatei**.



2. Wählen Sie **Einfache Verarbeitung für Standard-CSV ... verwenden** und klicken Sie auf "Weiter".

© 2016-2022 Altova GmbH

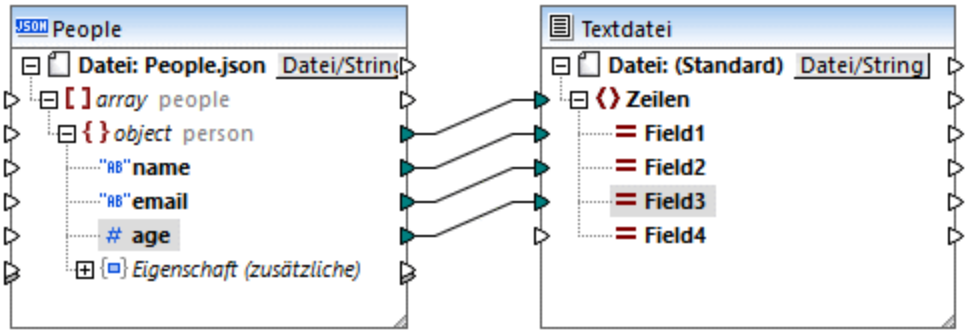


3. Klicken Sie mehrmals auf **Feld anhängen,** um vier CSV-Felder hinzuzufügen und klicken Sie anschließend auf **OK**.

Nähere Informationen zu Textkomponenten finden Sie unter CSV und Textdateien 585.

Schritt 3: Ziehen der Mapping-Verbindungen

4. Ziehen Sie im Mapping, wie unten gezeigt, die Verbindungen zwischen der JSON-Komponente und der CSV-Komponente.



Nähere Informationen dazu finden Sie unter Arbeiten mit Verbindungen 89.

Schritt 4: Vorschau auf das Mapping

Bei Anzeige einer Vorschau auf die Transformationsausgabe auf dem Register **Ausgabe** sehen Sie das folgende Ergebnis:

```
Alethia Alonso,altethia@example.com,35,
Klaus Mauer,klaus@example.com,57,
Natsuo Shinohara,natsuo@example.com,29,
```

In der Ausgabe sehen Sie, dass für jedes person-Objekt in der JSON-Datei eine neue Zeile erstellt wird und dass die Objekteigenschaften, wie gewünscht, durch Kommas voneinander getrennt werden.

Wir haben das Feld `birthday` aus der JSON-Instanzdatei, das im Schema nicht vorhanden ist, allerdings noch nicht gemappt.

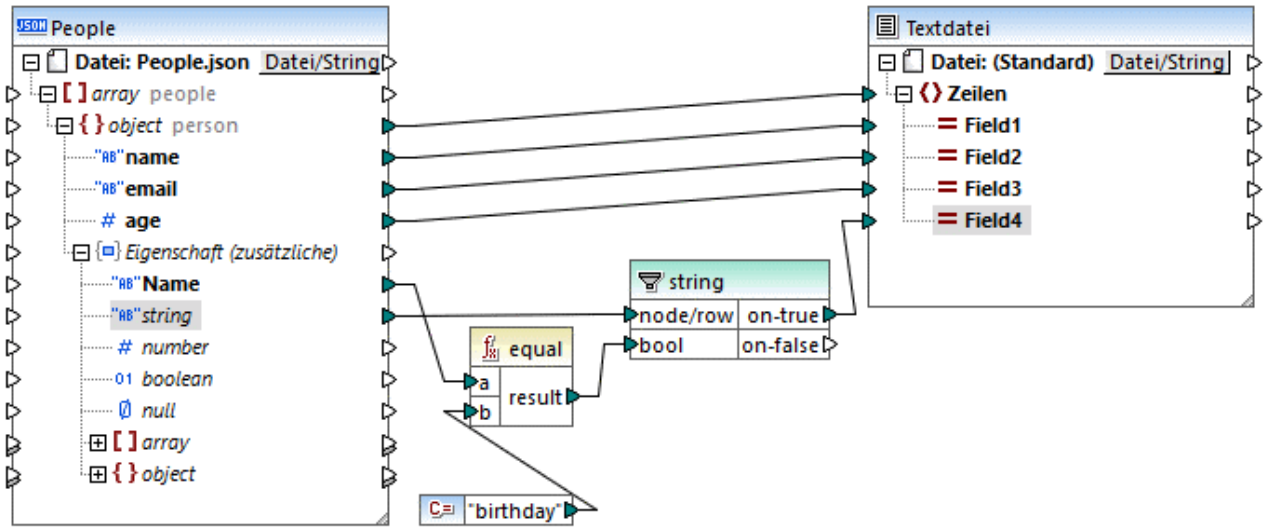
Schritt 5: Mappen zusätzlicher Eigenschaften

Erinnern Sie sich, dass das Feld `birthday` eine zusätzliche Eigenschaft ist, die nicht im Schema definiert ist. Um Daten aus solchen Feldern auslesen zu können, bietet MapForce in der JSON-Komponente zusätzliche Datenelemente.

Um in diesem Beispiel Daten aus dem Feld `birthday` auszulesen, fügen Sie folgende Datenelemente zum Mapping hinzu:

1. Eine Konstante mit dem Wert "birthday", siehe Hinzufügen einer Konstante zum Mapping 943.
2. eine "Filter: Nodes/Zeilen" Komponente, siehe Filter und Bedingungen 914.
3. die logische Funktion **`equal`**, siehe Hinzufügen einer Funktion zum Mapping 943.

Verbinden Sie die Komponenten als nächstes wie unten gezeigt. Wenn Sie die Komponenten auf diese Art verbinden, sucht MapForce nach einer zusätzlichen Eigenschaft mit dem Namen **birthday** und kopiert den String-Wert, falls einer gefunden wird, in das Feld **Field4** der Textkomponente.



Über das Datenelement **name** unter **property (additional)** haben Sie im Mapping oben anhand des Namens der zusätzlichen Eigenschaft Zugriff darauf. Die folgenden Datenelemente (**string**, **number**, **boolean**, usw.) entsprechen den JSON-Datentypen. Anhand dieser Datenelemente haben Sie Zugriff auf den tatsächlichen JSON-Wert dieses Typs. Der Name der zusätzlichen Eigenschaft in diesem Beispiel lautet "birthday" und der Typ ist "string". Daher wurden diese beiden Datenelemente mit dem Filter verbunden.

Wenn Sie zu diesem Zeitpunkt eine Vorschau auf die Transformation anzeigen, sehen Sie das folgende Ergebnis:

```
Alethia Alonso,altethia@example.com,35,4 July
Klaus Mauer,klaus@example.com,57,31 August
Natsuo Shinohara,natsuo@example.com,29,
```


© 2016-2022 Altova GmbH

Wie Sie in der Ausgabe sehen, enthält das vierte CSV-Feld nun den Wert der einzigen zusätzlichen Eigenschaft vom Typ "string", in diesem Fall von `birthday`. Da die dritte Person keine solche Eigenschaft `birthday` hat, steht in der CSV-Komponente an der entsprechenden Position kein Wert zur Verfügung.

4.9.6 Beispiel: Konvertieren von Excel in JSON

In diesem Beispiel wird gezeigt, wie Sie mit MapForce anhand einer Excel-Datei ein JSON-Dokument erstellen. Die Excel-Quellarbeitsmappe besteht aus zwei Arbeitsblättern: **Office** und **Staff**. Das Arbeitsblatt **Office** enthält Informationen über die Firmenadresse, während das Arbeitsblatt **Staff** eine Liste aller Angestellten enthält.

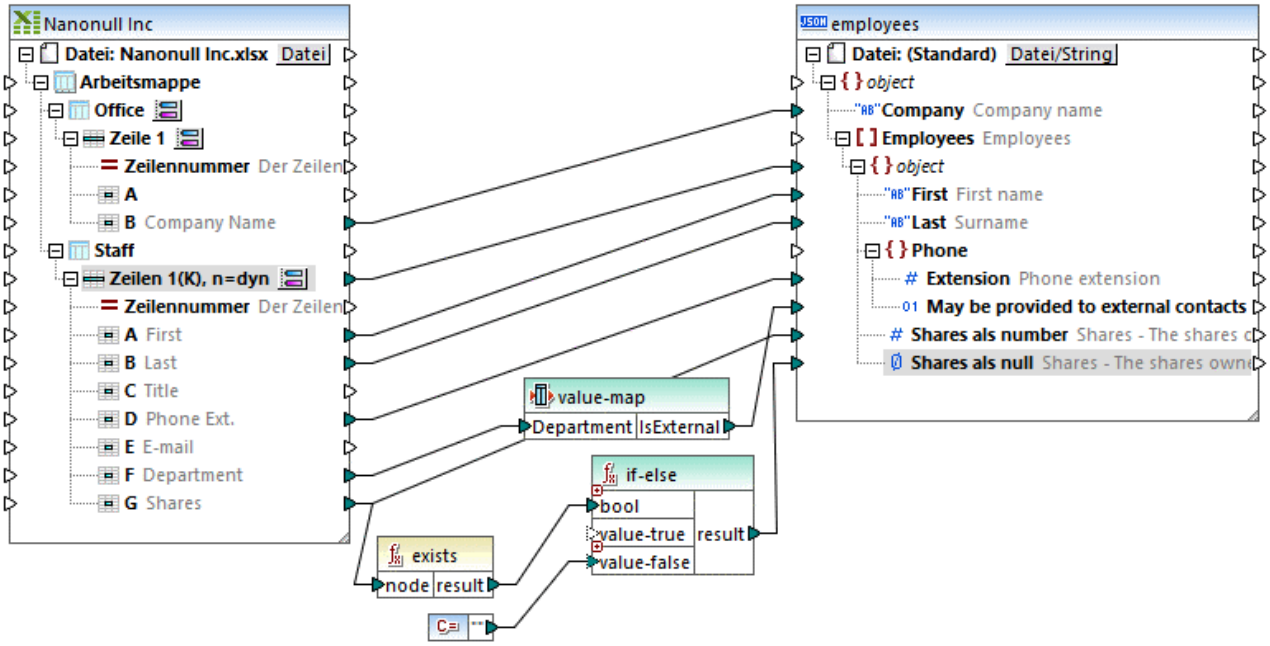
	A	B	C	D	E	F	G
1	First	Last	Title	Phone Ext.	E-mail	Department	Shares
2	Jessica	Bander	Support Engineer	241	j.band@nanonull.com	IT & Technical Support	0
3	Michelle	Butler	Software Engineer	654	m.butler@nanonull.com	Engineering	1500
4	Vernon	Callaby	Office Manager	582	v.callaby@nanonull.com	Administration	1500
5	Frank	Further	Accounts Receivable	471	f.further@nanonull.com	Administration	0
6	Liz	Gardner	Software Engineer	753	l.gardner@nanonull.com	Engineering	
7	George	Hammer	Web Developer	223	g.hammer@nanonull.com	IT & Technical Support	0

Arbeitsblatt "Staff"

Die Aufgabenstellung ist die folgende:

1. Extraktion der Daten aus der Excel-Datei und Konvertierung der Daten in ein JSON-Dokument.
2. In der JSON-Datei soll für jeden Angestellten festgelegt werden, ob seine Durchwahl (Phone Ext.) auch externen Kontakten zugänglich sein soll. Nur Durchwahlen von Mitarbeitern aus den Abteilungen Marketing und Verwaltung können von außerhalb zugänglich sein.
3. Die Freigaben der einzelnen Angestellten(durchwahlen) müssen in der JSON-Datei in Form eines numerischen Werts, der auch Null enthalten kann, angezeigt werden. Wenn nicht bekannt, müssen die Freigaben in der JSON-Datei als `null` angegeben werden.

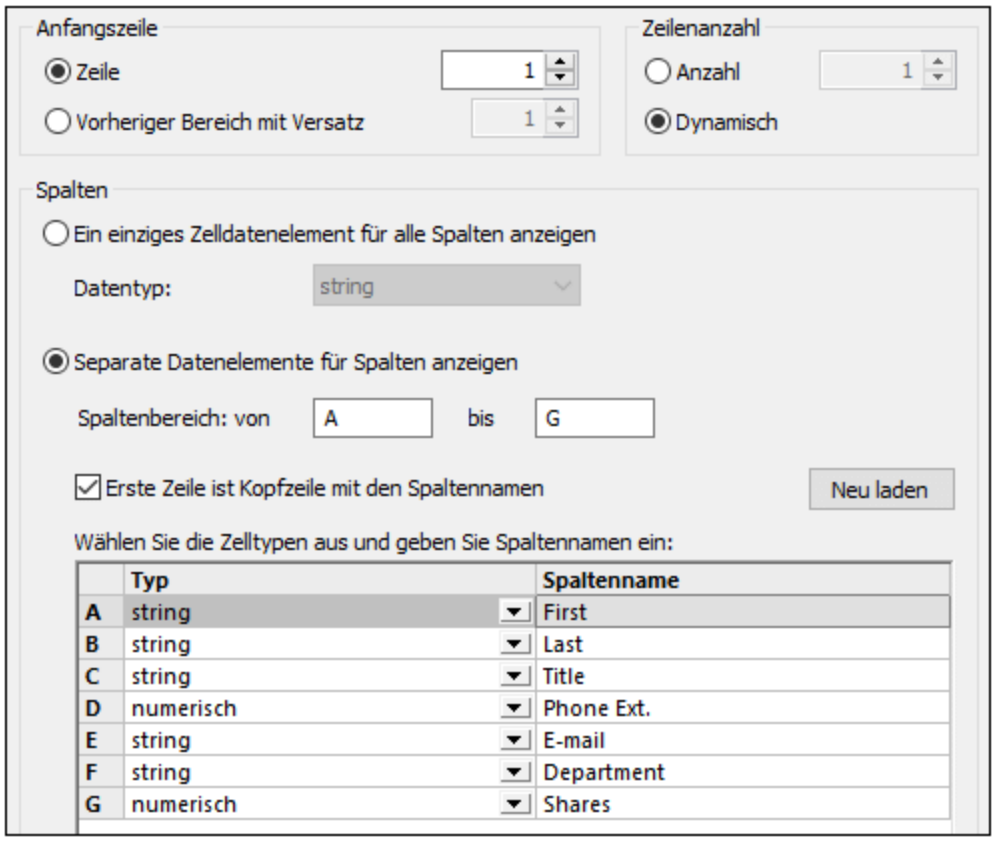
Unten sehen Sie das Mapping dazu. Sie finden es unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2023\MapForceExamples\\Excel_Company_To_JSON.mfd.



Excel_Company_To_JSON.mfd

Wie oben gezeigt, besteht das Mapping aus einer Excel-Quellkomponente und einer JSON-Zielkomponente. Beachten Sie, dass die beiden Arbeitsblätter der Excel-Datei wie in der Excel-Quellkomponente aufscheinen. Der Firmenname wird aus dem Arbeitsblatt **Office** gemappt, während der Rest der Felder aus dem Arbeitsblatt **Staff** gemappt wird. Wenn Sie sehen möchten, wie das Mapping zum Auslesen von Zeilendaten aus einzelnen Arbeitsblättern konfiguriert ist, klicken Sie auf die entsprechende -Schaltfläche. In der Abbildung unten sehen Sie die Einstellungen für das Arbeitsblatt **Staff**.

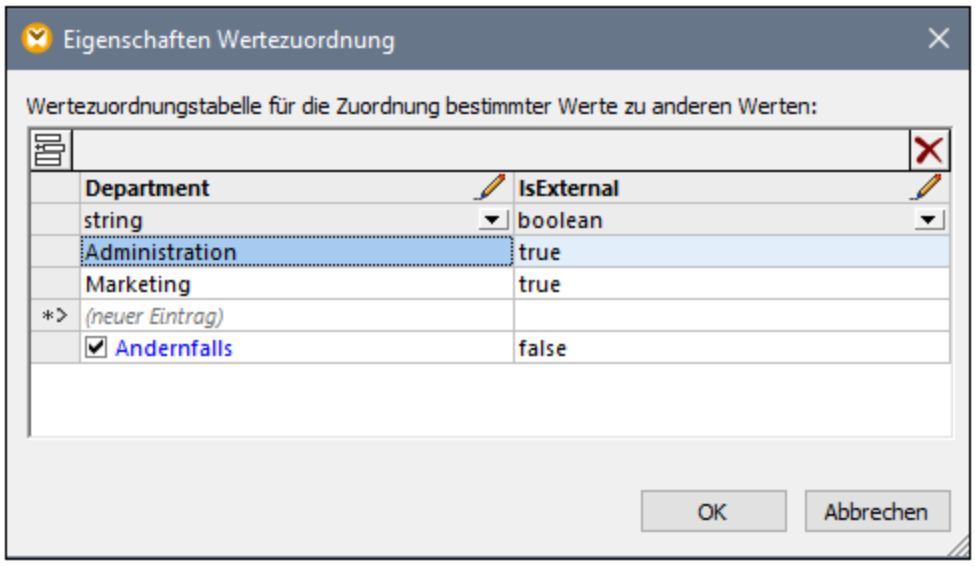
© 2016-2022 Altova GmbH



Gemäß der oben gezeigten Einstellung werden Zeilen ab Zeile 1 bis zum Ende (dynamisch) ausgelesen. Die erste Zeile ist eine Überschrift mit Spaltennamen, daher wurde das Kontrollkästchen **Erste Zeile ist Kopfzeile...** aktiviert. Außerdem ist im Dialogfeld die Art der Spalten A bis G sowie der Name, unter dem die jeweilige Spalte im Mapping angezeigt wird, definiert. Nähere Informationen zu Excel-Komponenten finden Sie unter Microsoft OOXML Excel 2007+ 738.

Die Zielkomponente ist eine JSON-Struktur. Sie können JSON-Komponenten wie unter Hinzufügen von JSON-Dateien als Mapping-Komponenten 823 beschrieben, zum Mapping hinzufügen. Die Struktur entspricht dem JSON-Schema **employees.schema.json**, welches sich im selben Ordner wie das Mapping befindet. Nähere Informationen zum Konfigurieren der JSON-Komponente finden Sie unter JSON-Komponenteneinstellungen 825.

Außerdem enthält das Mapping eine Reihe von Zwischenkomponenten. Bei der ersten handelt es sich um eine Wertezuordnungskomponente. Mit Hilfe dieser Komponente wird anhand der Abteilung entschieden, ob die Telefonnummer gemäß Punkt 2 oben extern freigegeben werden darf. Sie können die Einstellungen der Wertezuordnungskomponente durch Doppelklick auf deren Titelleiste anzeigen:



Die obige Konfiguration kann folgendermaßen zusammengefasst werden: Wenn **Department** = "Administration" oder "Marketing", gib `true` zurück. Andernfalls gib `false` zurück. Der auf diese Weise ermittelte Wert `true` oder `false` wird anschließend in die JSON-Zieleigenschaft geschrieben. Nähere Informationen zu Wertezuordnungen finden Sie unter Verwendung von Wertezuordnungen[921].

Zuletzt wird schließlich mit Hilfe der Funktion exists[1079] ermittelt, ob in der Excel-Quelldatei im Kontext der einzelnen Zeilen ein **Shares (Freigaben)**-Wert vorhanden ist. Diese Funktion gibt einen Booleschen Wert `true` zurück, wenn der Wert vorhanden ist. Andernfalls wird `false` zurückgegeben. Der Ergebniswert `true` oder `false` wird anschließend als Input an eine **If-Else**-Bedingung geliefert. Bei `false` wird ein von einer Konstanten bereitgestellter leerer String an das Zieldatenelement **Shares as null** übergeben. Dieser leere Wert wird in der erzeugten JSON-Datei zu einer `null`, da dies, wie durch das Symbol ∅ angezeigt, der Datentyp des Datenelements **Shares as null** ist.

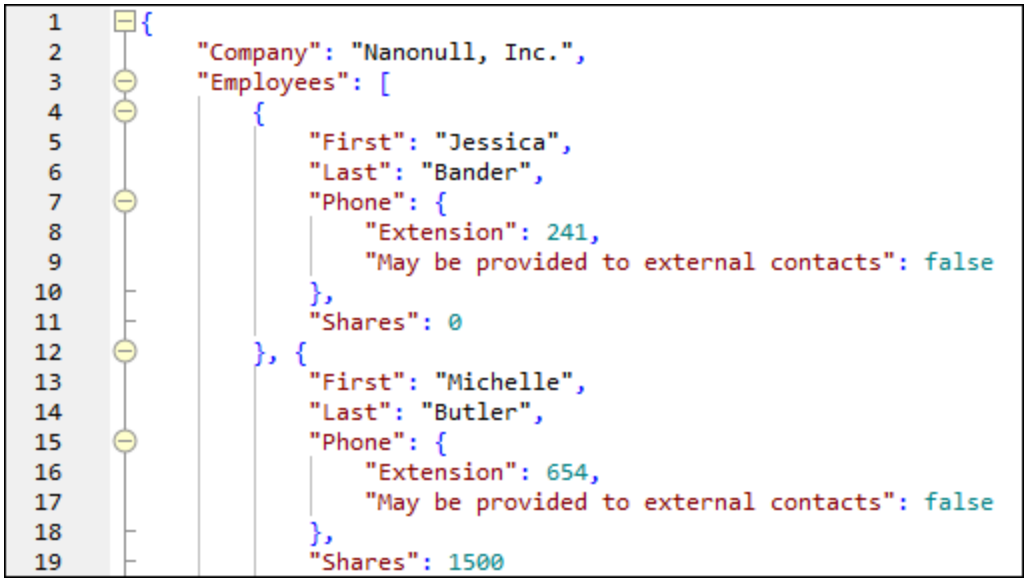
Wenn der Share-Wert des Angestellten numerisch ist, erfüllt er die **If-Else**-Bedingung nicht mehr und wird stattdessen über eine separate Verbindung in das Zieldatenelement **Shares as number** (vom numerischen Typ #) geschrieben.

Nähere Informationen zum Hinzufügen von Funktionen zu einem Mapping finden Sie unter Hinzufügen einer Funktion zum Mapping[943]. **If-Else**-Bedingungen werden unter Filter und Bedingungen[914] nähere beschrieben.

Ausführen des Mappings

Um in MapForce eine Vorschau auf die generierte Datei zu sehen, klicken Sie auf das Register **Ausgabe**. In der Abbildung unten sehen Sie nur ein Fragment der erzeugten JSON-Datei.

© 2016-2022 Altova GmbH



```
{
    "Company": "Nanonull, Inc.",
    "Employees": [
        {
            "First": "Jessica",
            "Last": "Bander",
            "Phone": {
                "Extension": 241,
                "May be provided to external contacts": false
            },
            "Shares": 0
        }, {
            "First": "Michelle",
            "Last": "Butler",
            "Phone": {
                "Extension": 654,
                "May be provided to external contacts": false
            },
            "Shares": 1500
```

Mapping-Ausgabe

Um die Mapping-Ausgabe in einer Datei zu speichern, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Ausgabe** auf **Ausgabedatei speichern**.
- Klicken Sie auf die Symbolleisten-Schaltfläche **Generierte Ausgabe speichern** .

Automatisierung mit MapForce Server

Wenn Sie MapForce Server lizenziert haben, können Sie das Mapping auch über die Befehlszeile auf einem Linux-, macOS- oder Windows-Rechner ausführen. Beachten Sie, dass bei Ausführung von Mappings auf einem Server normalerweise einige zusätzliche Schritte durchgeführt werden müssen, wie unter Vorbereiten von Mappings für die Server-Ausführung 1435 beschrieben. Im Folgenden wird beschrieben, was Sie tun müssen, um dieses Mapping auf einem Server auszuführen:

1. Klicken Sie mit der rechten Maustaste auf das Mapping und wählen Sie im Kontextmenü den Befehl **Mapping-Einstellungen**. Achten Sie darauf, dass das Kontrollkästchen **Pfade im generierten Code absolut machen** deaktiviert ist und speichern Sie das Mapping.
2. Klicken Sie im Menü **Datei** auf **Zu MapForce Server-Ausführungsdatei kompilieren.** Dadurch wird im Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples** eine MapForce Server-Ausführungsdatei generiert.
3. Kopieren Sie die generierte .mfx-Datei in ein Verzeichnis auf den Server-Rechner (nennen wir es das "Arbeitsverzeichnis").
4. Kopieren Sie die Excel-Datei aus **<Dokumente>\Altova\MapForce2023\MapForceExamples\ \Nanonull Inc.xlsx** in das Arbeitsverzeichnis auf dem Server-Rechner.
5. Öffnen Sie einen Terminal und ändern Sie das aktuelle Verzeichnis in das Arbeitsverzeichnis.
6. Führen Sie MapForce Server mit dem unten gezeigten Befehl aus. Passen Sie **mapforceserver_exec** an den Pfad der ausführbaren MapForce Server-Datei auf Ihrem Betriebssystem an.

```
mapforceserver_exec run Excel_Company_To_JSON.mfx
```

Sie können Mappings bei der Server-Ausführung auch als API-Aufruf oder als FlowForce Server-Auftrag (entweder bei Bedarf oder in regelmäßigen Abständen) ausführen. Nähere Informationen dazu finden Sie unter Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442.

© 2016-2022 Altova GmbH

4.10 Protocol Buffers

MapForce unterstützt das Mappen von Daten von oder auf im Protocol Buffers-Format kodierte Binärdateien (https://developers.google.com/protocol-buffers/). MapForce (oder MapForce Server) kann Daten aus solchen Binärdateien auslesen und in jedes beliebige andere von MapForce unterstützte Format konvertieren (z.B. XML, CSV, Datenbank, usw.). Ebenso kann MapForce Daten aus jedem von MapForce unterstützten Format auslesen und als Protocol Buffers-Binärdatei schreiben. Zusätzlich dazu können Sie Protocol Buffers-Dateien in Webservice Requests und Responses verwenden, siehe Hinzufügen eines Webservice-Aufrufs (REST) 612.

Beachten Sie die folgenden Punkte:

- Damit Daten aus Binärdateien im Protocal Buffers-Format gelesen oder in dieses geschrieben werden können, muss als Transformationssprache des Mappings "Built-In" ausgewählt werden. Die Codegenerierung in C#, C++, Java, XSLT oder XQuery wird nicht unterstützt. Nähere Informationen dazu finden Sie unter Transformationssprachen 28.
- Damit ein Datenaustausch möglich wird, gibt es zu Binärdateien im Protocol Buffers-Format so genannte `.proto`-Dateien. MapForce unterstützt die Protocol Buffers `.proto`-Dateiversionen 2 und 3.

In einer `.proto`-Datei ist, ähnlich wie bei einer XML-Schema-Datei, bei der die Struktur einer XML-Instanzdatei beschrieben ist, die Struktur der kodierten Binärdateien definiert. Im folgenden Codefragment finden Sie ein Beispiel für eine `.proto` -Datei, in der ein Person-Objekt beschrieben ist:

```
syntax="proto3";

message Person {
  string name = 1;
  int32 id = 2;
  string email = 3;
}
```

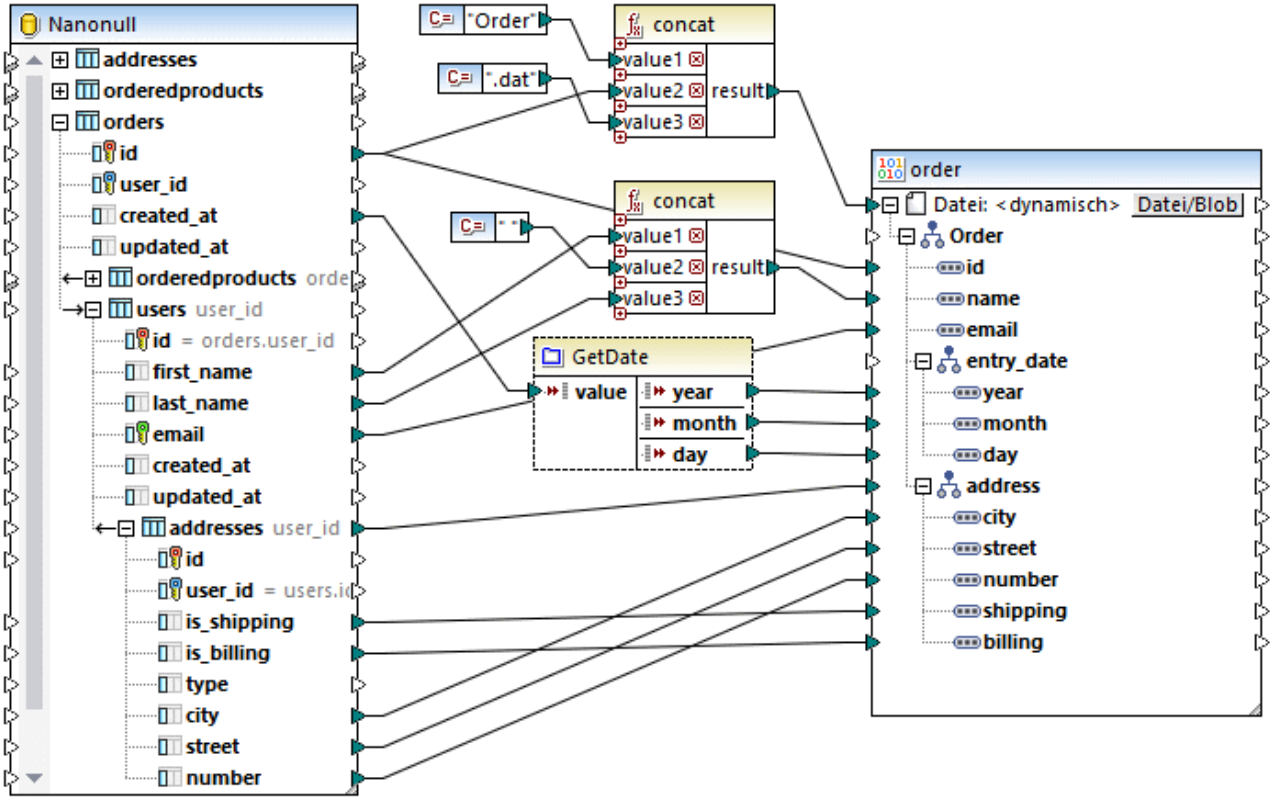
Beispiel .proto-Datei

Mit Hilfe der `.proto`-Datei kann die dazugehörige Binärdatei interpretiert und verarbeitet werden. Damit MapForce daher die Daten im Protocol Buffers-Format auslesen kann, benötigen Sie die folgenden Dateien:

1. die Quell-Binärdatei. Diese Datei kann eine Erweiterung wie .bin oder .dat oder eine beliebige andere Erweiterung haben.
2. die `.proto`-Datei.

Um mit MapForce Daten im Protocol Buffers-Format schreiben zu können, benötigen Sie nur die `.proto`-Datei. Sie können Ihre .proto-Dateien entweder manuell erstellen oder Sie von demjenigen anfordern, für den Sie Daten im entsprechenden Format erstellen sollen. Jedenfalls generiert MapForce (oder MapForce Server) auf Basis der `.proto`-Datei die Binärdatei.

Damit Daten aus Protocol Buffers-Dateien außerhalb von MapForce erstellt oder gelesen werden können, müssen Sie Programmcode generieren und schreiben. Mit MapForce müssen Sie jedoch keinen Code schreiben oder anhand von `.proto`-Dateien generieren. Fügen Sie einfach die `.proto`-Datei zum Mapping hinzu und ziehen Sie die erforderlichen Verbindungen visuelle. Wenn Sie das Mapping ausführen, liest MapForce (oder ggf. MapForce Server) je nach Mapping-Design Daten aus der/den Quell-Binärdatei(en) aus oder generiert Binärdateien. In der Abbildung unten sehen Sie z.B. ein Mapping, in dem Daten aus einer Datenbank ausgelesen und als mehrere Binärdateien im Protocol Buffers-Format geschrieben werden.



GenerateOrders.mfd

Eine ausführlichere Beschreibung zum Mapping finden Sie unter Beispiel: Schreiben von Daten in Protocol Buffers 854.

4.10.1 Hinzufügen von Binärdateien zum Mapping

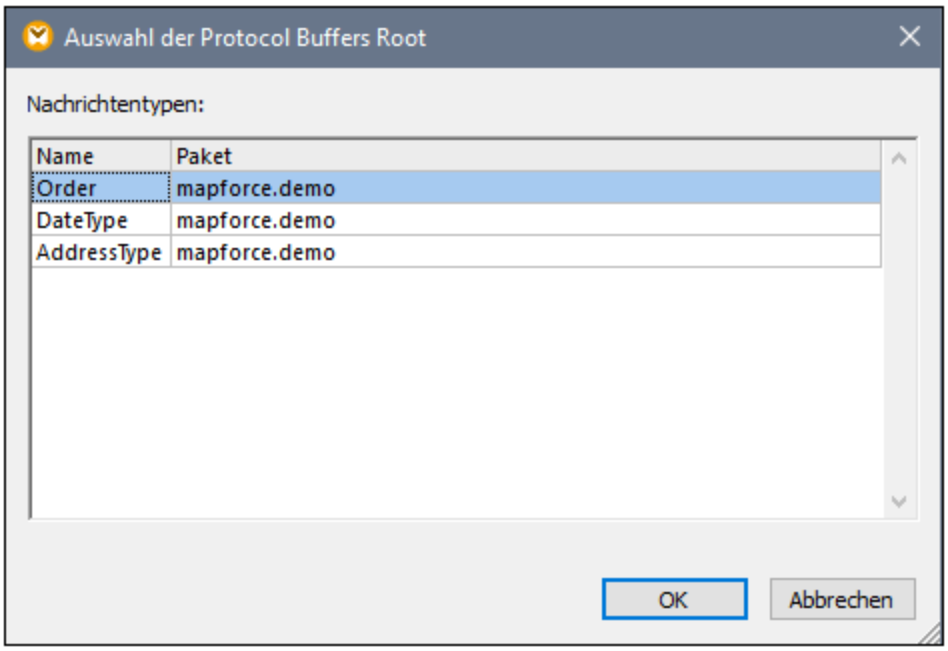
Um Daten von oder auf im Protocol Buffers-Format kodierte Binärdateien mappen zu können, müssen Sie zuerst die dazugehörige .proto-Datei, wie unten beschrieben, zum MapForce-Mapping hinzufügen.

So fügen Sie eine .proto-Datei als Mapping-Komponente hinzu:

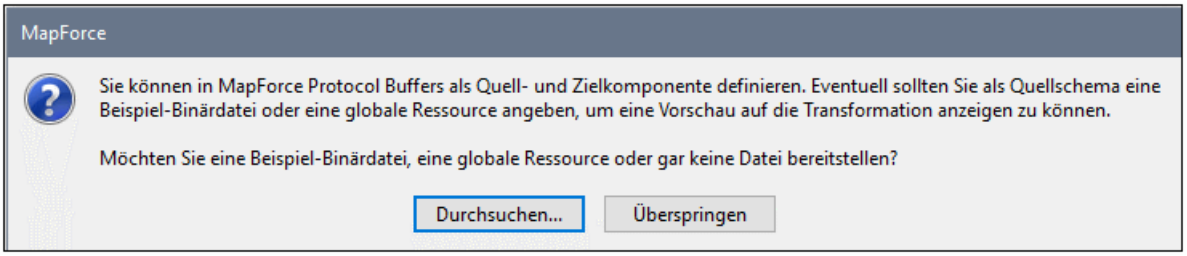
1. Klicken Sie im Menü **Einfügen** auf **Protocol Buffers-Datei** oder klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche **Protocol Buffers-Datei einfügen** .
2. Wählen Sie eine der folgenden Methoden:
   a. Um eine auf der Festplatte gespeicherte .proto-Datei zu öffnen, navigieren Sie zur Datei und klicken Sie auf **Öffnen**.
   b. Um eine .proto-Datei über eine URL zu öffnen, klicken Sie auf **Zu URL wechseln** und geben Sie in das Dialogfeld die URL der Datei ein, siehe Hinzufügen von Komponenten über eine URL 80.
   c. Um eine zuvor als globale Ressource definierte .proto-Datei zu öffnen, klicken Sie auf **Zu globalen Ressourcen**. Nähere Informationen zu globalen Ressourcen finden Sie unter Globale Altova-Ressourcen 1462.

 © 2016-2022 Altova GmbH

3. Eine .proto-Datei enthält normalerweise mehrere Nachrichtentypen. Wählen Sie, wenn Sie dazu aufgefordert werden, den Nachrichtentyp aus, der das oberste Objekt in der Hierarchie ist (die so genannte "Root").

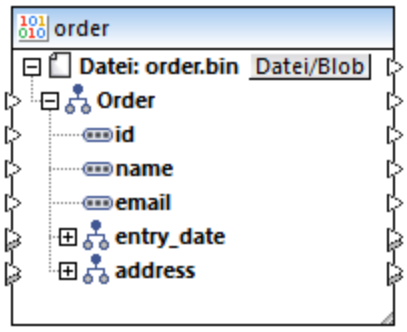


4. Klicken Sie auf **OK**. Daraufhin wird ein Dialogfeld angezeigt.



1. Wenn Daten aus einer Protocol Buffers-Binärdatei ausgelesen werden sollen, klicken Sie auf **Durchsuchen** und wählen Sie die gewünschte Binärdateiinstanz aus. Wenn Sie beabsichtigen, Daten in eine Protocol Buffers-Datei zu schreiben, wählen Sie zu einem späteren Zeitpunkt eine Instanz aus und klicken Sie auf **Überspringen**.

Wenn die .proto.Datei erfolgreich gelesen werden kann, zeigt MapForce, ähnlich wie bei anderen Komponententypen, ihre Struktur im Mapping an, z.B.:



Sie haben bisher eine Binärkomponente zum Mapping hinzugefügt, doch ist das Mapping noch nicht vollständig, da keine Daten gelesen oder geschrieben werden. Beispiele für vollständige Mappings finden Sie unter:

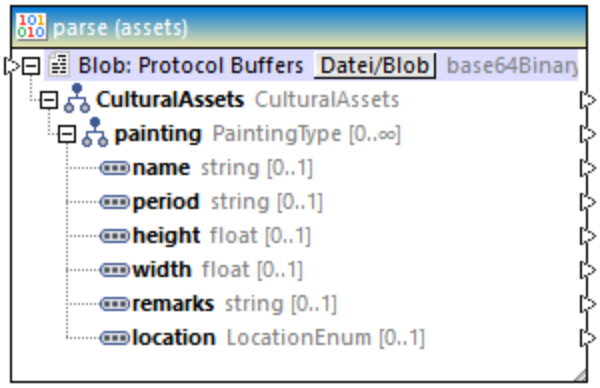
- Beispiel: Auslesen von Daten aus Protol Buffers 849
- Beispiel: Schreiben von Daten in Protocol Buffers 854

Schaltfläche "Datei/Blob"

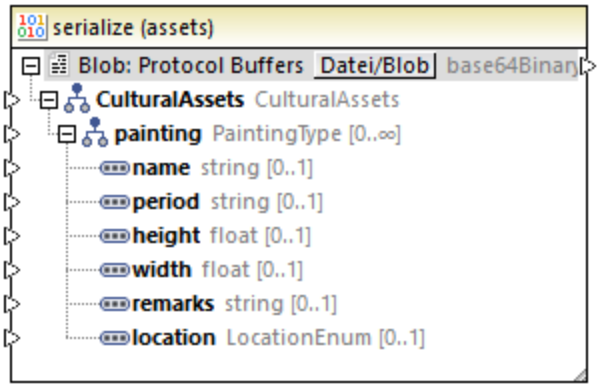
Mit der Schaltfläche **Datei/Blob** der Binärkomponente wird ein Kontextmenü mit einer Reihe von Befehlen für die folgenden Situationen geöffnet:

- **Dateinamen aus Komponenteneinstellungen verwenden** - Dies ist die standardmäßig ausgewählte Option. Sie sollte verwendet werden, wenn in Ihrem Mapping eine einzige Binärdatei ausgelesen oder generiert wird. Wenn diese Option definiert ist, können Sie den Namen der Datei über das Dialogfeld "Komponenteneinstellungen" ändern, siehe Binärdatei-Komponenteneinstellungen 847.
- **Über das Mapping bereitgestellte dynamische Dateinamen verwenden** - Dieser Befehl ist sinnvoll, wenn mit Ihrem Mapping mehrere Binärdateien ausgelesen oder generiert werden sollen. Wenn diese Option definiert ist, können Sie den Dateinamen nicht mehr über das Dialogfeld "Komponenteneinstellungen" ändern. Stattdessen definieren Sie die Dateinamen auf Basis von Informationen, die aus dem Mapping selbst stammen. Ein Beispiel dafür finden Sie unter dem Beispiel: Schreiben von Daten in Protocol Buffers 854.
- **Blobs parsen zu Protocol Buffers** - Mit diesem Befehl wird die Komponente in eine Parser-Komponente umgewandelt. Auf diese Art können Sie einen Binärwert zum Protocol Buffers-Format parsen, um eine mapbare Baumstruktur zu erhalten. Dies funktioniert ähnlich wie beim Parsen und Serialisieren von Strings, siehe Parsen und Serialisieren von Strings 1382, mit dem Unterschied, dass hier keine Zeichenstrings, sondern Binär-Byte-Strings zum Einsatz kommen. So können Sie etwa in der Komponente unten ein Datenelement vom Typ Binärdaten mit dem "Blob" Input auf der linken Seite verbinden. Zur Mapping-Laufzeit wird dieser Binärwert geparst und es werden Daten an die Outputs der mapbaren Struktur auf der rechten Seite übergeben.

© 2016-2022 Altova GmbH



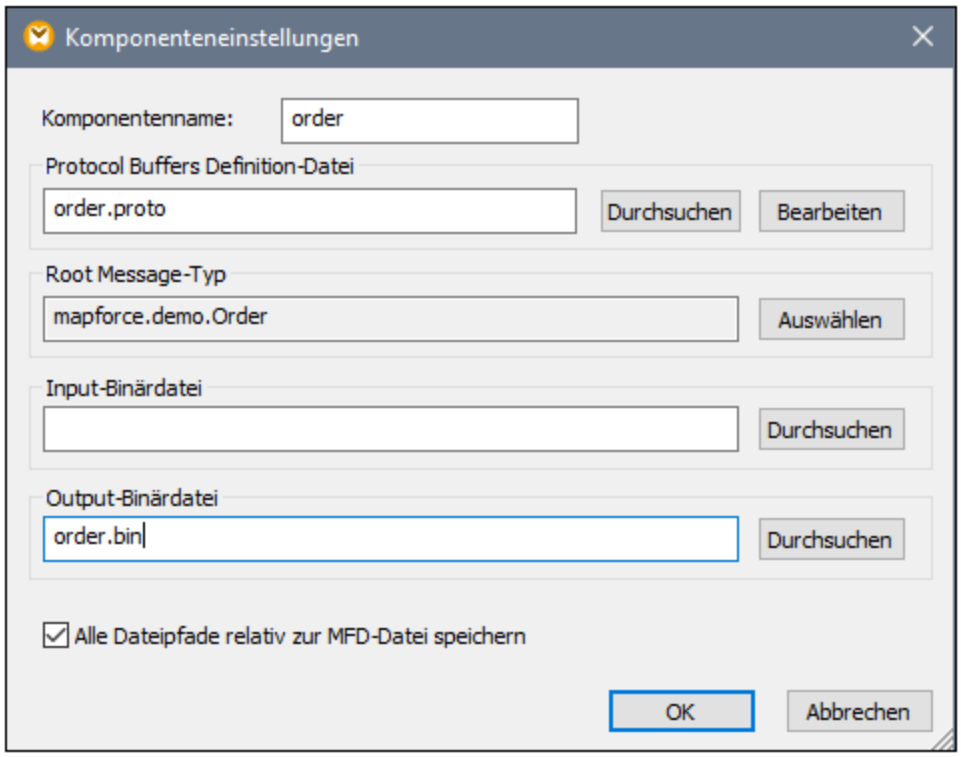
- **Protocol Buffers in Blobs serialisieren** - Mit diesem Befehl wird die Komponente in eine Serialisierungskomponente umgewandelt. Dadurch können Sie eine mapbare Baumstruktur in einen Binärwert im Protocol Buffers-Format serialisieren. So können Sie etwa in der Komponente unten Datenelemente der entsprechenden Typen mit der mapbaren Struktur auf der linken Seite verbinden. Zur Mapping-Laufzeit wird die Struktur serialisiert und der einzige Output (vom Typ Binärdatei) auf der rechten Seite wird befüllt, sodass Sie diesen weiter mit einer Zielkomponente verbinden können. Dies ist z.B. nützlich, wenn in Ihrem Mapping Daten als BLOB-Typ (Binary Large Object) in eine Zieldatenbank geschrieben werden sollen. Eine weitere Verwendungsmöglichkeit ist, Daten in Binärdatenfelder (vom Typ "Bytes") einer Protocol Buffers-Zielkomponente zu schreiben.



4.10.2 Protocol Buffers-Komponenteneinstellungen

Nachdem Sie eine Binärkomponente zum Mappingbereich hinzugefügt haben, können Sie die entsprechenden Einstellungen dafür über das Dialogfeld "Komponenteneinstellungen" vornehmen. Sie können dieses Dialogfeld auf eine der folgenden Arten aufrufen:

- Wählen Sie die Komponente im Mapping aus und klicken Sie im Menü **Komponente** auf **Eigenschaften**.
- Doppelklicken Sie auf die Komponentenüberschrift.
- Klicken Sie mit der rechten Maustaste auf die Komponentenüberschrift und wählen Sie den Befehl **Eigenschaften**.



Dialogfeld "Komponenteneinstellungen"

Es stehen die folgenden Einstellungen zur Verfügung:

Komponentenname	Der Komponentenname wird bei der Erstellung der Komponente automatisch generiert. Sie können den Namen jedoch jederzeit ändern. Der Komponentenname kann Leerzeichen und Punkte enthalten, darf aber keine Schrägstriche, umgekehrten Schrägstriche, Doppelpunkte, doppelten Anführungszeichen und voran- oder nachgestellte Leerzeichen enthalten. Beachten Sie beim Ändern des Namens einer Komponente Folgendes: • Wenn Sie das Mapping auf FlowForce Server bereitstellen möchten, muss der Komponentenname eindeutig sein. • Es wird empfohlen, nur Zeichen zu verwenden, die über die Befehlszeile eingegeben werden können. Länderspezifische Sonderzeichen sind in Windows und der Befehlszeile eventuell unterschiedlich kodiert.
Protocol Buffers Definition-Datei	Definiert den Namen oder Pfad der Strukturdefinitionsdatei (.proto-Datei), anhand der MapForce die eigentlichen Binärdaten mappt. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie die neue Datei aus.

© 2016-2022 Altova GmbH

Root Message-Typ	Eine .proto-Datei enthält normalerweise mehrere Nachrichtentypen. Klicken Sie auf **Auswählen**, um den Nachrichtentyp, der als oberstes Element der Hierarchie ("Root") angezeigt werden soll, auszuwählen.
Input-Binärdatei	Definiert die Input-Binärdatei, aus der MapForce die Daten liest. Dieses Feld ist bei einer Quellkomponente von Bedeutung. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie die neue Datei aus.
Output-Binärdatei	Definiert die Output-Binärdatei, in die MapForce die Daten schreibt. Dieses Feld ist bei einer Zielkomponente von Bedeutung. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie die neue Datei aus.
Alle Dateipfade relativ zur MFD-Datei speichern	Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld "Komponenteneinstellungen" angezeigten Dateipfade relativ zum Ordner, in dem sich die MapForce Design (.mfd)-Datei befindet. Siehe auch Verwenden relativer Pfade in einer Komponente 85.

4.10.3 Beispiel: Auslesen von Daten aus Protol Buffers

In diesem Beispiel wird gezeigt, wie Daten aus einer im Protocol Buffers-Format kodierten Binärdatei ausgelesen werden. Die Binär-Quelldatei befindet sich unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\assets.bin**. Die `.proto`-Datei, die die Binärdatei beschreibt, befindet sich im selben Verzeichnis und sieht folgendermaßen aus:

```
syntax="proto3";

package mapforce.demo;

message CulturalAssets {
  repeated PaintingType painting = 1;
}

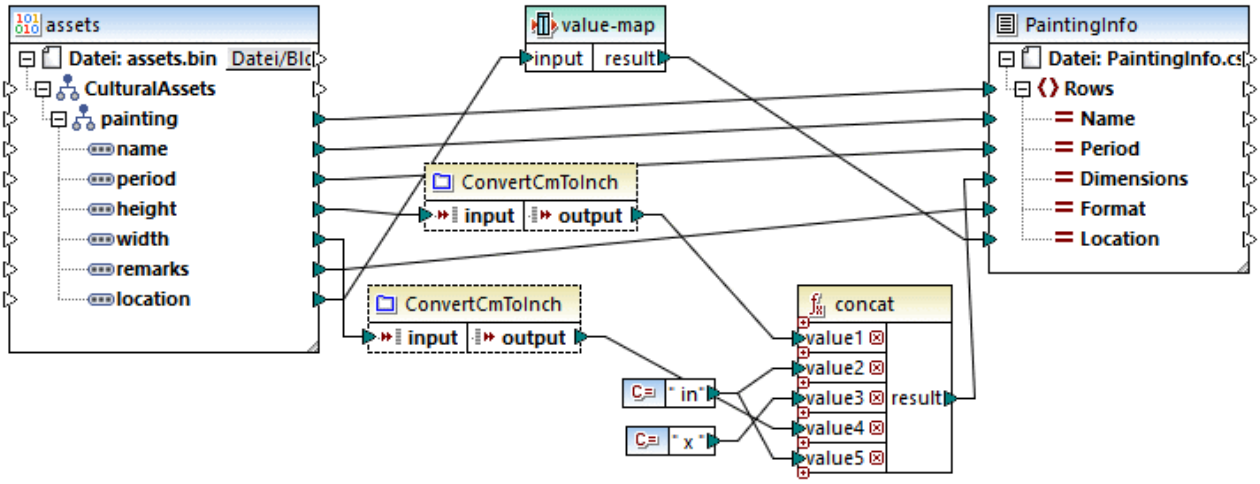
message PaintingType {
       string name = 1;
       string period = 2;
       float height = 3; // in cm
       float width = 4; // in cm
       string remarks = 5;
       LocationEnum location = 6;

       enum LocationEnum {
              UNKNOWN = 0;
              MUSEUM = 1;
              TEMPLE = 2;
              PRIVATE = 3;
       }
}
```

assets.proto

Die Binär-Quelldatei enthält Informationen über verschiedene Kulturschätze (in diesem Beispiel eine Gemäldesammlung). Wie Sie aus der obigen `.proto` -Datei ersehen, hat jedes Gemälde eine in Zentimetern definierte Höhe (**height)** und Breite (**width)**. Es gibt eine Reihe weiterer Felder zur Beschreibung eines Gemäldes, darunter eine Enumeration, in der Aufbewahrungsort angegeben ist (Museum, private Sammlung oder Tempel). Diese Informationen sollen nun in eine kommagetrennte Datei (CSV-Datei) extrahiert werden. Außerdem sollen alle Abmessungen von Zentimetern in Zoll umgerechnet werden und in einem einzigen String in einem Format wie "24 in x 56.8 in" dargestellt werden.

Unten sehen Sie das Mapping, mit dem die Daten den Anforderungen gemäß extrahiert werden und das gewünschte Ergebnis erzeugt wird. Sie finden dieses Mapping im folgenden Verzeichnis: **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Sie können das Mapping öffnen und direkt ausführen oder die schrittweise Anleitung unten befolgen, um es von Grund auf neu zu erstellen.



ReadProtocolBuffers.mfd

Wie oben gezeigt, besteht das Mapping aus einer Binär-Quellkomponente (**assets**) und CSV-Zielkomponente (**PaintingInfo**). Des Weiteren enthält es die folgenden Zwischenkomponenten:

- eine Wertezuordnungstabelle zur Konvertierung von Enumerationswerten wie 0, 1, 2, 3 in eine für Menschen lesbare Stringdarstellung (z.B., `0 = UNKNOWN`, `1 = MUSEUM`, usw.).
- die Funktion **`ConvertCmToInch`**. Wie der Name vermuten lässt, handelt es sich um eine benutzerdefinierte Funktion, die Zentimeter in Zoll konvertiert. Diese Funktion wird im Mapping zwei Mal aufgerufen (für die **Höhe** und die **Breite**).
- die **`concat`**-Funktion. Diese Funktion verkettet mehrere Werte und gibt einen String im Format "# in x # in" zurück, wobei # die Höhe und Breite des Bildes in Zoll repräsentiert.

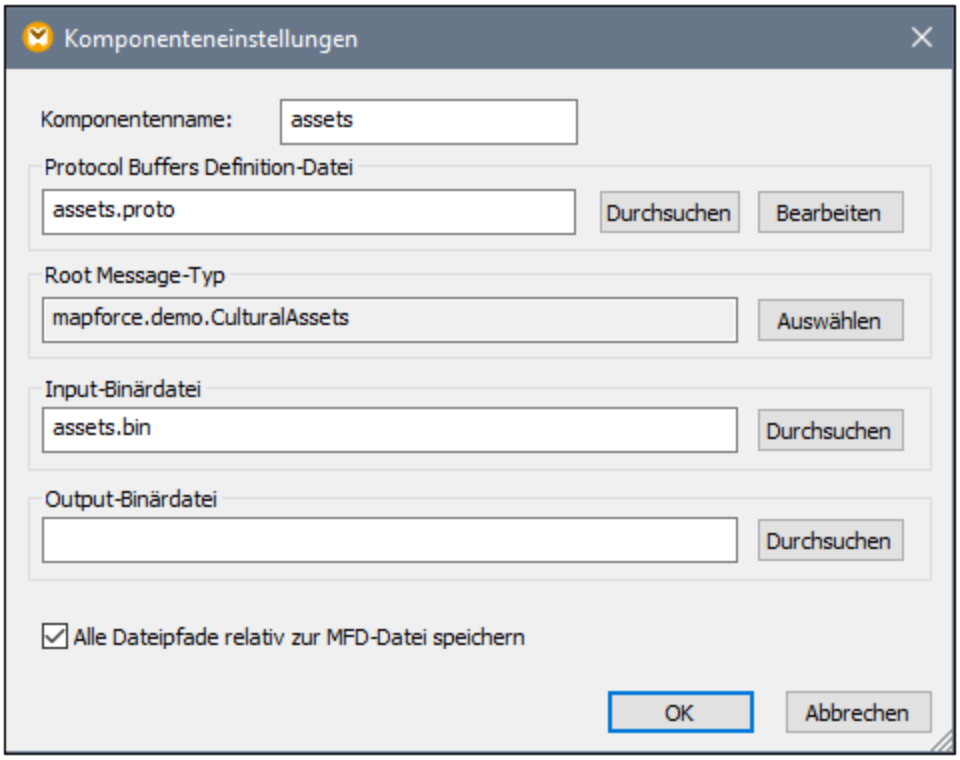
In der folgenden Anleitung wird beschrieben, wie Sie das obige Mapping von Grund auf neu erstellen.

Schritt 1: Hinzufügen der Quell-Binärdatei

1. Klicken Sie im Menü **Einfügen** auf **Protocol Buffers-Datei** und navigieren Sie zur folgenden Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\assets.proto**.

 © 2016-2022 Altova GmbH

2. Wenn Sie dazu aufgefordert werden, wählen Sie "CulturalAssets" als Root Message aus und klicken Sie auf **OK**.
3. Doppelklicken Sie auf die Titelleiste der neuen Komponente. Daraufhin wird das Dialogfeld "Komponenteneinstellungen" geöffnet.
4. Klicken Sie neben **Input-Binärdatei** auf **Durchsuchen** und wählen Sie die folgenden Datei aus: **<Dokumente>\Altova\MapForce2023\MapForceExamples\assets.bin**.



Siehe auch Hinzufügen von Binärdateien zum Mapping[844].

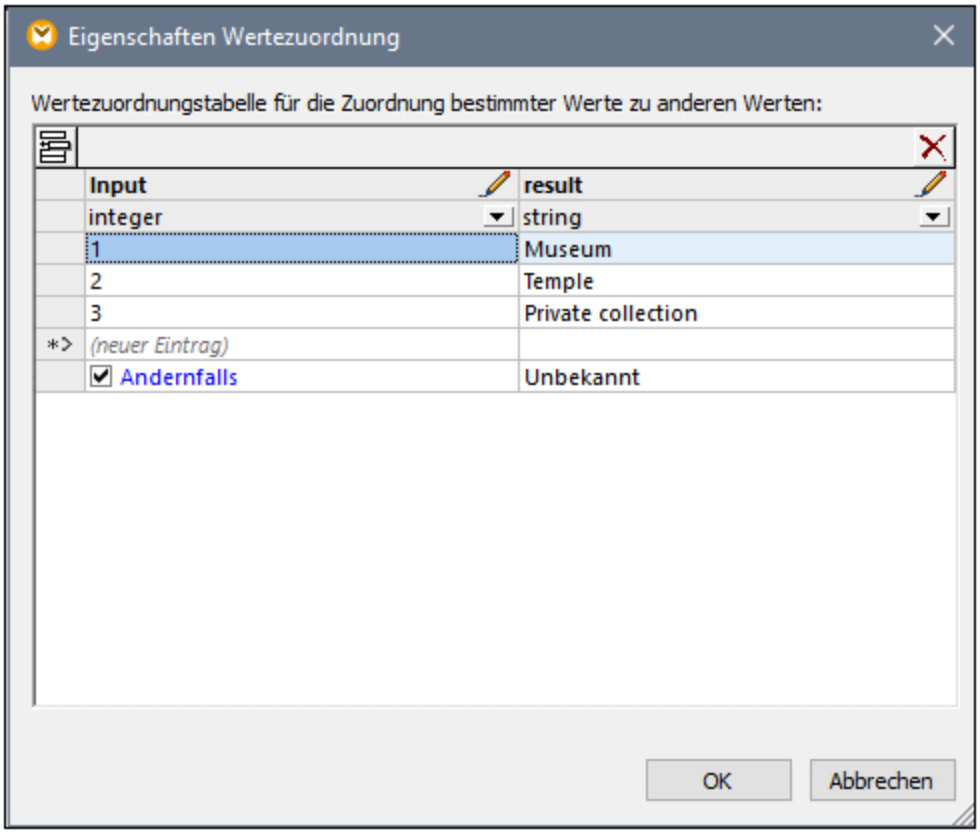
Schritt 2: Hinzufügen der CSV-Zielkomponente

1. Klicken Sie im Menü **Einfügen** auf **Textdatei**.
2. Aktivieren Sie das Kontrollkästchen **Einfache Verarbeitung für Standard-CSV...wählen** und klicken Sie auf **Weiter**. Daraufhin wird das Dialogfeld "CSV-Komponenteneinstellungen" geöffnet.
3. Klicken Sie mehrmals auf die Schaltfläche **Feld anhängen**, um fünf Zielfelder für die erwarteten Daten zu erstellen.
4. Doppelklicken Sie optional auf die Feldüberschriften, um ihnen beschreibende Namen zu geben.

Nähere Informationen dazu finden Sie unter Definieren der CSV-Optionen[595].

Schritt 3: Hinzufügen der Wertezuordnung

1. Klicken Sie im Menü **Einfügen** auf **Wertezuordnung**.
2. Erstellen Sie für jeden gemappten Datensatz, wie unten gezeigt, einen Eintrag.



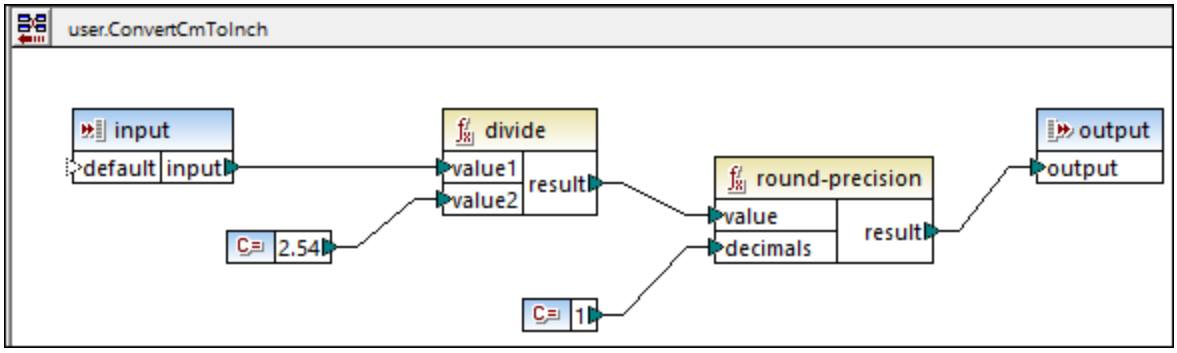
Mit Hilfe der "Andernfalls"-Bedingung können Sie festlegen, was geschieht, wenn die Binärdatei keinen Wert für das Feld `Location` enthält. In diesem Fall wird der Wert "Unbekannt" in die Zielkomponente geschrieben, anstatt diese leer zu lassen. Nähere Informationen dazu finden Sie unter Verwendung von Wertezuordnungen[921].

Schritt 4: Erstellen der benutzerdefinierten Funktion

1. Klicken Sie im Menü **Funktion** auf **Benutzerdefinierte Funktion erstellen** und geben Sie der Funktion den Namen "ConvertCmToInch". Belassen Sie alle anderen Optionen unverändert.
2. Ziehen Sie die vordefinierten Funktionen **`divide`**[1064] und **`round-precision`**[1067] aus dem Fenster "Bibliotheken" in das Funktionsmappingfenster. Siehe auch Hinzufügen einer vordefinierten Funktion zum Mapping[943].
3. Fügen Sie zwei Konstanten hinzu und ziehen Sie alle Verbindungen, wie unten gezeigt, siehe auch Hinzufügen einer Konstante zum Mapping[943]. Die erste Konstante liefert die Dezimalkonstante zum Konvertieren von Zentimetern in Zoll gemäß der Formel und die zweite liefert die Rundungsgenauigkeit.

Im Prinzip dividiert die Funktion den Input-Wert durch 2,54 und rundet das Ergebnis dann auf eine Dezimalstelle, wie unten gezeigt:

© 2016-2022 Altova GmbH

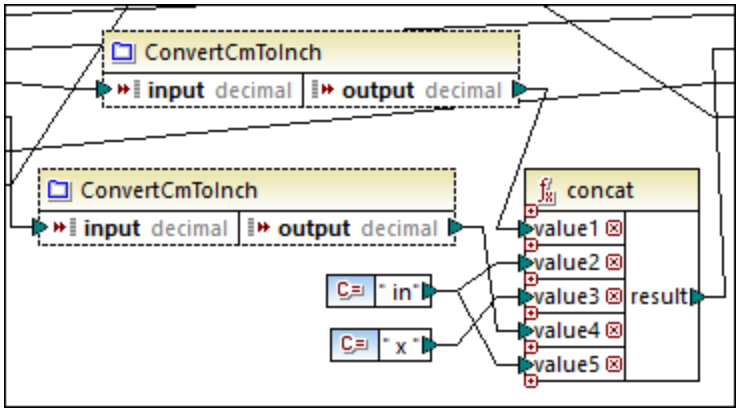


Um etwaige Fehler zu vermeiden, setzen Sie den Datentyp der Input- und Output-Komponente auf "decimal". Um den Datentyp anzuzeigen oder zu ändern, doppelklicken Sie auf die Titelleiste der Komponente oder klicken Sie mit der rechten Maustaste auf die Titelleiste und wählen Sie im Kontextmenü den Befehl **Eigenschaften**.

Nach Definition der benutzerdefinierten Funktion klicken Sie auf **Zurück zum Haupt-Mapping** und ziehen Sie die Funktion aus dem Fenster "Bibliotheken" in das Mapping. In diesem Beispiel wird die Funktion zwei Mal aufgerufen. Nähere Informationen finden Sie unter Benutzerdefinierte Funktionen 967.

Schritt 5: Hinzufügen der Verkettungsfunktion

Geben Sie im Fenster "Bibliotheken" die ersten Buchstaben von "concat" ein und ziehen Sie die **concat** 1105-Funktion anschließend in das Mapping. Um alle erforderlichen Input-Argumente hinzuzufügen, klicken Sie mehrmals auf **Parameter hinzufügen** ( ) , siehe auch Hinzufügen oder Löschen von Funktionsargumenten 944. Die Konstanten " in" und " x " sind String-Werte. Das Ergebnis der Funktion ist die Höhe (**height**) des Bilds, gefolgt vom String " in ", gefolgt von " x ", gefolgt von der Breite (**width**) des Bilds, gefolgt von "in".



Ausführen des Mappings

Um eine Vorschau auf die generierte Datei zu sehen, klicken Sie in MapForce auf das Register **Ausgabe**. Beachten Sie, dass die dritte CSV-Spalte die Abmessungen des Gemäldes im gewünschten Format enthält.

```
1   Red and White Plum Blossoms,18th century,61.4 in x 67.8 in,Ink and color on paper,Museum
2   Red and white hibiscuses,1197,10 in x 9.9 in,Color on silk,Museum
3   Pigeon on a peach branch,1108,11.2 in x 10.3 in,Color on silk,Private collection
4   Pine trees in snow,1773,142.5 in x 61.2 in,Light color on paper,Museum
5   Fishing village in the evening glow,13th century,13 in x 44.3 in,Ink on paper,Museum
6   Five Guardian Kings,Kamakura period,76.3 in x 49.7 in,Color on silk,Temple
7   Snowclad houses in the night,Edo period,11 in x 51.2 in,Ink and light color on paper,Private
```

Mapping-Ausgabe

So speichern Sie die Mapping-Ausgabe auf der Festplatte:

- Klicken Sie im Menü **Ausgabe** auf **Ausgabedatei speichern**.

Automatisierung mit MapForce Server

Wenn Sie MapForce Server lizenziert haben, können Sie das Mapping auch über die Befehlszeile auf einem Linux-, macOS- oder Windows-Rechner ausführen:

1. Kompilieren Sie das Mapping mit dem Menübefehl **Datei | Zu MapForce Server-Ausführungsdatei kompilieren** zu einer MapForce Server-Ausführungsdatei (.mfx), siehe auch Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442.
2. Kopieren Sie die .mfx-Datei auf den Server-Rechner.
3. Führen Sie MapForce Server mit dem unten stehenden Befehl aus.

```
mapforceserver run ReadProtocolBuffers.mfx
```

Anmerkungen:

- **mapforceserver** ist der Pfad zur ausführbaren MapForce Server Datei für Ihr jeweiliges Betriebssystem.
- Ändern Sie den Pfad in die entsprechende .mfx-Datei oder kopieren Sie die .mfx-Datei in denselben Ordner wie die ausführbare Datei.
- Die .mfx-Datei enthält alle Informationen; die .proto-Datei wird für die Server-Ausführung nicht benötigt.

Sie können Mappings bei der Server-Ausführung als API-Aufruf oder als FlowForce Server-Auftrag (entweder bei Bedarf oder in regelmäßigen Abständen) ausführen. Nähere Informationen dazu finden Sie unter Automatisierung mit MapForce Server 1434.

4.10.4 Beispiel: Schreiben von Daten in Protocol Buffers

In diesem Beispiel wird gezeigt, wie Daten in im Protocol Buffers-Format kodierte Binärdateien geschrieben werden. Die Quelldaten in dieser Demo stammen aus einer SQLite-Datenbank (**Nanonull.sqlite**); Sie können Daten jedoch auf ähnliche Weise aus verschiedenen anderen von MapForce unterstützten Formaten, wie XML, EDI, JSON und anderen extrahieren.

© 2016-2022 Altova GmbH

In der Demo-Quelldatenbank sind Daten über Personen, deren Adressen und die Produkte, die sie in einem Geschäft bestellt haben, gespeichert. Aus der Datenbank sollen nun Bestellinformationen extrahiert werden und es sollen mehrere Binärdateien im Protocol Buffers-Format generiert werden. Für jede Bestellung soll eine Binärdatei mit der Erweiterung `.dat` generiert werden. Der Name der generierten Datei soll die eindeutige Bestell-ID, z.B. **Order1.dat**, **Order2.dat**, usw., enthalten. Außerdem soll jede Binärdatei die Bestell-ID, den Namen und die E-Mail Adresse der Person, das Datum und die Lieferadresse enthalten. Die `.proto` -Datei, die die Struktur einer solchen Bestellung beschreibt, sieht folgendermaßen aus:

```
syntax="proto3";

package mapforce.demo;

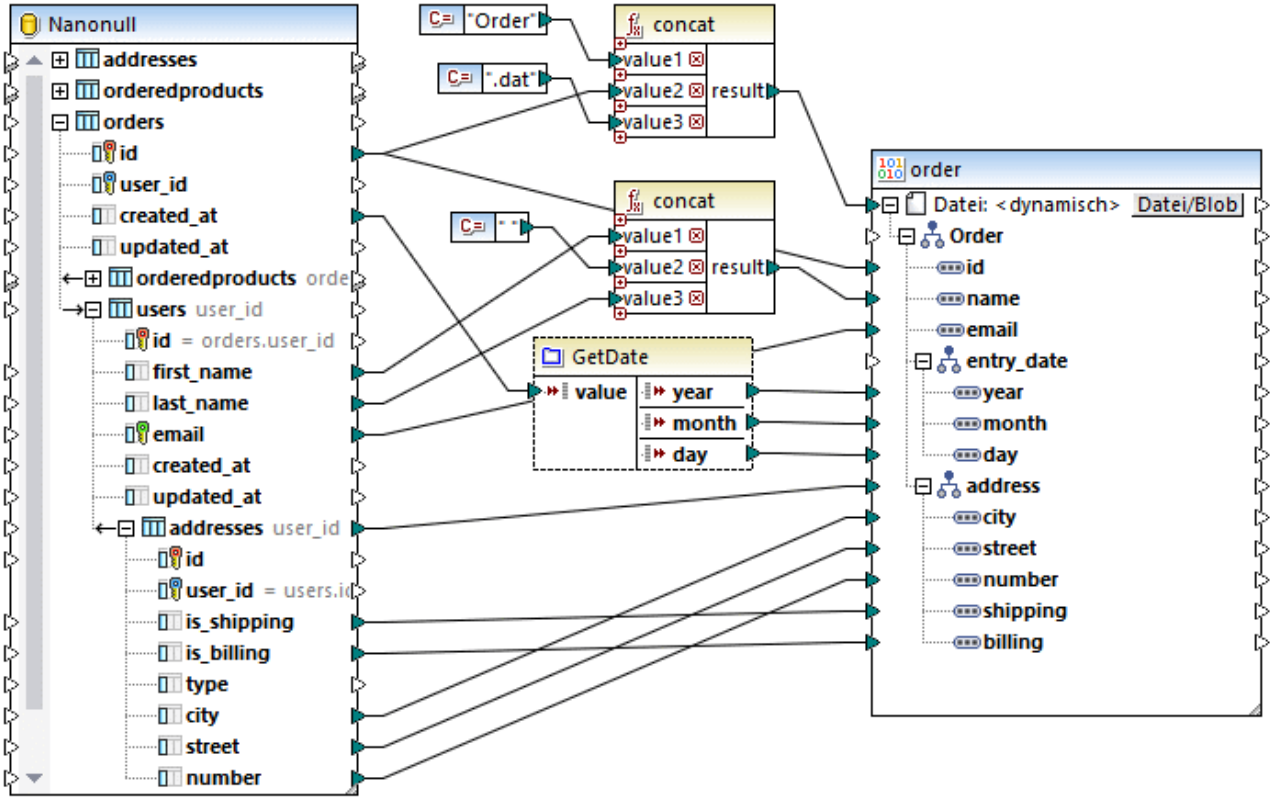
message Order {
  int32 id = 1;
  string name = 2;
  string email = 3;
  DateType entry_date = 4;
  // An order can have multiple addresses (shipping, billing)
  repeated AddressType address = 5;
}

message DateType {
  // Must be from 1 to 9999
  int32 year = 1;
  // Must be from 1 to 12
  int32 month = 2;
  // Must be from 1 to 31, and valid for the year and month
  int32 day = 3;
}

message AddressType {
  string city = 1;
  string street = 2;
  int32 number = 3;
  bool shipping = 5;
  bool billing = 4;
}
```

orders.proto

Sowohl die Datenbank **Nanonull.sqlite** als auch die Datei `orders.proto` steht im folgenden Verzeichnis zur Verfügung: **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Die Mapping-Design-Datei zur Durchführung der Transformation hat den Namen **GenerateOrders.mfd** und befindet sich im selben Verzeichnis. Sie können das Mapping öffnen und direkt ausführen oder die unten stehende Anleitung befolgen, um es von Grund auf neu zu erstellen.



GenerateOrders.mfd

Wie Sie oben sehen, besteht das Mapping aus einer Quellkomponente (der SQLite-Datenbank) und einer Zielkomponente (der Binärkomponente).

Beachten Sie, dass die Tabelle **users** unterhalb der Tabelle **orders** verschachtelt ist, da MapForce automatisch eine Sekundärschlüsselbeziehung zwischen diesen beiden Tabellen erkannt hat. Ebenso ist die Tabelle **addresses** unter **users** verschachtelt. Bei allen Verbindungen, die von solchen verschachtelten Tabellen gezogen werden, wird sichergestellt, dass die Schlüsselbeziehungen im Mapping beibehalten werden. Nähere Informationen dazu finden Sie unter Behandlung von Datenbankbeziehungen 266.

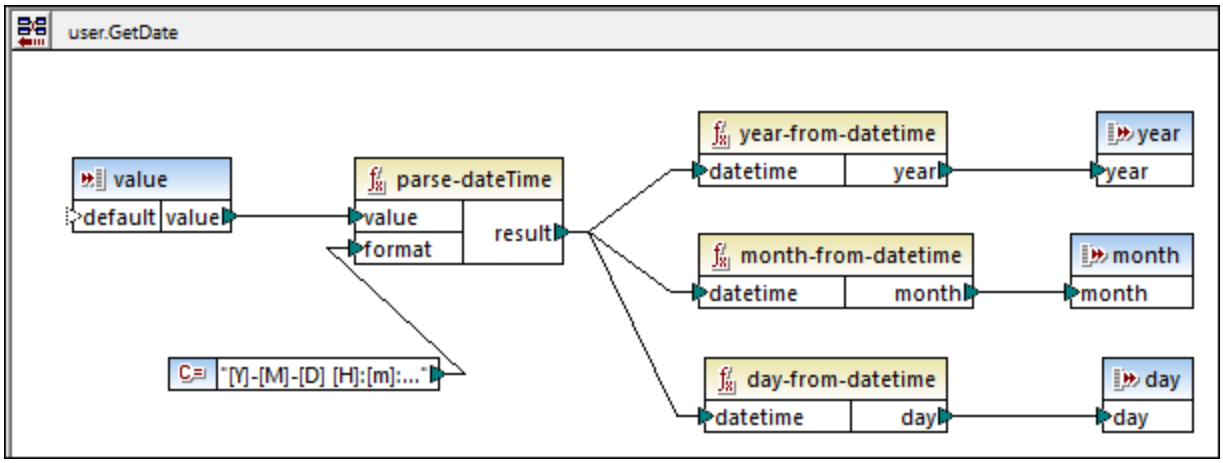
In der Zielkomponente sehen Sie am Datenelement **Datei<dynamisch>,** dass die Binärkomponente auf Basis der aus dem Mapping erhaltenen Informationen Instanzdateien dynamisch erstellt. Damit die Dateien der Zielkomponente dynamisch generiert werden, klicken Sie auf die Schaltfläche Datei/Blob und aktivieren Sie im Kontextmenü die Option **Über das Mapping bereitgestellte dynamische Dateinamen verwenden**. Nähere Informationen zu dynamischen Dateinamen finden Sie unter Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 1371.

In diesem Mapping wird für jede **id** in der Datenbanktabelle **orders** eine neue Datei generiert, wie Sie an der Verbindung zum Datenelement **Datei: <dynamisch>** sehen. Die Namen der einzelnen generierten Dateien werden mit Hilfe der ersten **concat** 1105-Funktion generiert. Diese Funktion verbindet den String "Order" mit der eindeutigen ID der Bestellung aus der Datenbank und der Dateierweiterung ".dat". Die generierten Dateien erhalten dadurch Namen wie **Order1.dat**, **Order2.dat,** usw.

 © 2016-2022 Altova GmbH

Die **concat**-Funktion wird im Mapping auch ein zweites Mal aufgerufen. Dieses Mal ist ihr Ergebnis der Name der Person, der durch Verkettung des Vor- und Zunamens und Einfügen eines Leerzeichens dazwischen erstellt wird.

Die "GetDate"-Komponente in der Mitte schließlich ist eine benutzerdefinierte Funktion, die die Datenfelder in der Binärdatei befüllt. Diese Funktion erhält als Input ein Datum in String-Form, parst es und gibt Jahr, Monat und Tag separat in Form von Ganzzahlwerten zurück. Diese Konvertierung ist erforderlich, um das in der `.proto`-Datei definierte Datumsformat zu erhalten. Wenn Sie im Mapping auf die Titelleiste der Funktion doppelklicken, können Sie die Definition dieser Funktion anzeigen oder ändern.



Benutzerdefinierte Funktion "GetDate"

Wie oben gezeigt, parst die benutzerdefinierte Funktion "GetDate" den Input-Wert mit Hilfe der vordefinierten Funktion **parse-dateTime**[1045]. Die Maske **[Y]-[M]-[D] [H]:[m]:[s]** entspricht dem in der Datenbank gespeicherten Datenformat, z.B. **2017-10-19 08:09:54**. Nähere Informationen zu Formatmasken finden Sie unter der Beschreibung der Funktion **parse-dateTime**[1045]. Als nächstes werden verschiedene vordefinierte Funktionen aufgerufen, um die relevanten Datumsteile aus den geparsten Daten zu extrahieren und zurückzugeben.

Erstellen des Mappings

Um das Mapping (oder ein ähnliches) von Grund auf neu zu erstellen, gehen Sie folgendermaßen vor:

1. Fügen Sie die Quelldatenbank, wie unter Hinzufügen von Datenbanken zum Mapping[256] beschrieben, hinzu.
2. Fügen Sie die `.proto`-Zieldatei ein, siehe Hinzufügen von Binärdateien zum Mapping[844].
3. Konfigurieren Sie die Zielkomponente so, dass die Dateinamen bei der Ausführung des Mappings dynamisch generiert werden. Klicken Sie dazu auf die Schaltfläche Datei/Blob und wählen Sie im Kontextmenü die Option **Über das Mapping bereitgestellte dynamische Dateinamen verwenden**. Verbinden Sie als nächstes das Datenelement **Datei<dynamisch>** mit einem Input, der den Dateinamen bereitstellt. In diesem Beispiel wird der Dateiname, wie oben beschrieben, mit Hilfe der **concat**-Funktion erstellt. Nähere Informationen zu dynamischen Dateinamen finden Sie unter Dynamische Verarbeitung mehrerer Input- oder Output-Dateien[1371].
4. Fügen Sie optional vordefinierte MapForce-Funktionen hinzu und verbinden Sie deren Inputs. In diesem Beispiel wird die **concat**-Funktion zwei Mal verwendet. Nähere Informationen dazu finden Sie unter Hinzufügen einer Funktion zum Mapping[943]. Beachten Sie, dass bei Aufruf von vordefinierten Funktionen wie **concat** einige der Input-Daten wahrscheinlich aus String-Konstanten stammen.

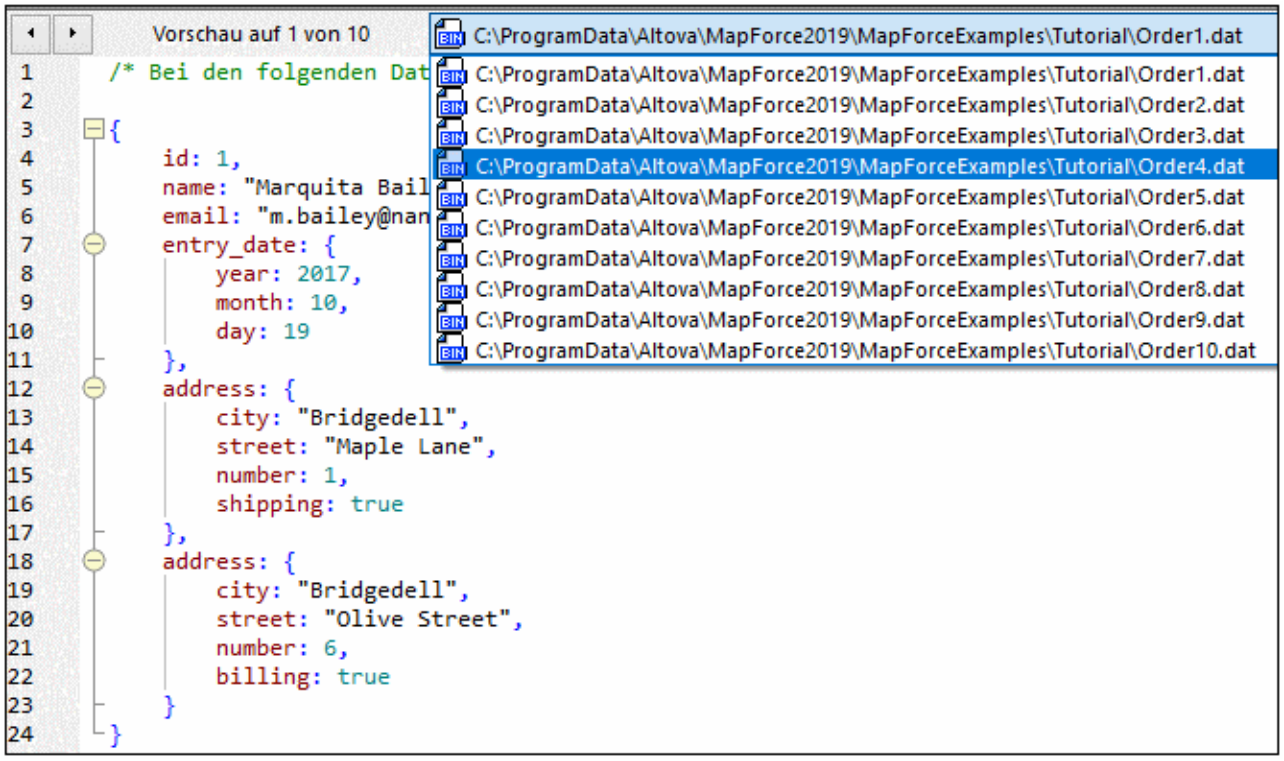
Informationen dazu, wie Sie solche Konstanten hinzufügen, finden Sie unter Hinzufügen einer Konstante zum Mapping[943].

5. Fügen Sie optional eventuell erforderliche benutzerdefinierte Funktionen hinzu. Mit Hilfe dieser Funktionen können einige der Mapping-Details wegabstrahiert werden, sodass Sie die Übersicht besser behalten. So konvertiert etwa die benutzerdefinierte Funktion "GetDate" in diesem Beispiel, wie erwähnt, einen String in Datumsteile, wobei ihre Implementierungslogik im Hauptmapping ausgeblendet wird. Nähere Informationen zum Erstellen von benutzerdefinierten Funktionen finden Sie unter Benutzerdefinierte Funktionen[967].

Wenn Sie mit MapForce noch gar nicht vertraut sind, finden Sie im Kapitel Tutorials[43] einige einfache Schritt-für-Schritt-Beispiele.

Ausführen des Mappings

Wie zu Beginn dieses Beispiels erläutert, sollen mit dem Mapping mehrere .dat-Dateien erzeugt werden. Um eine Vorschau auf die generierten Dateien zu sehen, klicken Sie auf das Register **Ausgabe**.



Wie oben gezeigt, werden im Fenster **Ausgabe** alle generierten Dateien angezeigt. Durch Klicken auf die Richtungspfeile oder auf einen Eintrag aus der Liste können Sie darin navigieren. Um die generierte Ausgabe zu speichern, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Ausgabe** auf **Alle Ausgabedateien speichern**. ( ).
- Klicken Sie auf die Symbolleisten-Schaltfläche **Alle generierten Ausgaben speichern** ( ).

Wenn Sie mit MapForce eine Vorschau auf die Ausgabe-Binärdateien generieren, wird ihre Struktur im JSON-Format angezeigt, da dieses Format im Gegensatz zum Binärformat für Menschen lesbar ist. Durch

© 2016-2022 Altova GmbH

Anzeige der Ausgabe im JSON-Format können Sie das Mapping testen. Wenn Sie die Ausgabedatei speichern, wird diese jedoch als Binärdatei gespeichert.

Automatisierung mit MapForce Server

Wenn Sie MapForce Server lizenziert haben, können Sie das Mapping auch über die Befehlszeile auf einem Linux-, macOS- oder Windows-Rechner ausführen:

1. Kompilieren Sie das Mapping mit dem Menübefehl **Datei | Zu MapForce Server-Ausführungsdatei kompilieren** zu einer MapForce Server-Ausführungsdatei (.mfx), siehe auch Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442.
2. Kopieren Sie die .mfx-Datei auf den Server-Rechner.
3. Da in diesem Mapping Daten aus einer SQLite-Datenbankdatei ausgelesen werden, kopieren Sie die Datenbankdatei **Nanonull.sqlite** auf dem Server in dasselbe Verzeichnis wie die .mfx-Datei. Informationen zu anderen Datenbankarten finden Sie unter Datenbankmappings in verschiedenen Ausführungsumgebungen 166.
4. Führen Sie MapForce Server mit dem unten stehenden Befehl aus.

```
mapforceserver run GenerateOrders.mfx
```

Anmerkungen:

- **mapforceserver** ist der betriebssystemspezifische Pfad zur ausführbaren MapForce Server-Datei.
- Ändern Sie den Pfad zur .mfx-Datei nach Bedarf oder kopieren Sie die .mfx-Datei in denselben Ordner wie die ausführbare Datei.

Sie können Mappings bei der Server-Ausführung als API-Aufruf oder als FlowForce Server-Auftrag (entweder bei Bedarf oder in regelmäßigen Abständen) ausführen. Nähere Informationen dazu finden Sie unter Automatisierung mit MapForce Server 1434.

4.11 Binärdateien

MapForce bietet die Möglichkeit, BLOB (Binary Large Objects)-Daten aus Binärdateien in ein Mapping einzulesen und dieses anschließend zu verarbeiten, ohne die interne Struktur der Binärdaten (Rohdaten) zu verändern. So können Sie Binärdaten etwa in einem BLOB-Datenbankfeld, einem Feld vom Typ `xs:base64Binary` in einer XML-Datei speichern oder an einen Webservice senden.

* *Webservice-Aufrufe werden nur in der MapForce Enterprise Edition unterstützt.*

Sie können auch Mappings erstellen, in denen Binärdaten aus einer Quelle (wie z.B. einem BLOB-Feld in einer Datenbank, einem Feld vom Typ `xs:base64Binary` in einer XML-Datei oder einem Webservice) gelesen und anschließend als Binärdateien auf den Datenträger geschrieben werden.

Im Folgenden werden nur einige Beispielszenarien beschrieben, in denen Sie die Funktion zum Lesen oder Schreiben von Binärdateien nutzen können:

- Extraktion von als base-64-Daten kodiertem Binärdateninhalt aus einer XML-Datei und Speicherung auf dem Datenträger (z.B. als PDF-Datei)
- Verarbeitung von auf dem Datenträger gespeicherten Bilddateien und Senden dieser Dateien als base-64-kodiertem Binärdateninhalt an einen Webservice
- Extraktion von BLOB-Inhalt aus einer Datenbanktabelle und Speichern des Inhalts als Bilddateien auf dem Datenträger (eine Bilddatei pro Zeile in der Datenbanktabelle)
- Lesen von Bilddateien vom Datenträger und Speichern des Inhalts als BLOB-Datenfelder in einer Datenbanktabelle.

Anmerkung: Um Daten von oder auf Binärdateien mappen zu können, muss als Transformationssprache BUILT-IN[29] verwendet werden. Sie können in MapForce eine Vorschau auf das Mapping anzeigen (und Ausgabedateien, falls vorhanden, speichern) oder das Mapping mit (einem separat lizenzierten) MapForce Server auf einem anderen Rechner oder einer anderen Plattform ausführen. Die Generierung eines ausführbaren C#-, C++- oder Java-Programms anhand von Mappings, die Binärdateien lesen oder schreiben, wird nicht unterstützt.

Hinzufügen von Binärdateien zum Mapping

Mit Binärdateien ist in MapForce an sich keine Komponentenart verknüpft, wie dies sonst z.B. bei XML-Text oder JSON-Dateien der Fall ist. Damit Sie Aufgabe, wie die oben erwähnten durchführen können, stehen stattdessen die folgenden vordefinierten MapForce-Funktionen zur Verfügung.

- read-binary-file[1149]
- write-binary-file[1150]

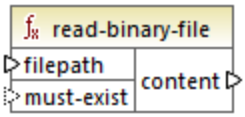
Sie finden diese Funktionen im Fenster "Bibliotheken" (in der Bibliothek "lang"). Um eine dieser Funktionen in einem Mapping zu verwenden, ziehen Sie sie aus dem Fenster "Bibliotheken" in das Mapping oder doppelklicken Sie in einen leeren Bereich des Mappings und geben Sie die ersten Buchstaben des Funktionsnamens ein, um die Funktion auszuwählen. Nähere Informationen dazu finden Sie unter Hinzufügen von Funktionen zum Mapping[943].

read-binary-file

Diese Funktion gibt den Inhalt der angegebenen Datei als BLOB (Binary Large Object) vom Typ `xs:base64Binary` zurück. Zwar handelt es sich beim Datentyp um "base64Binary", doch wird er intern einfach

© 2016-2022 Altova GmbH

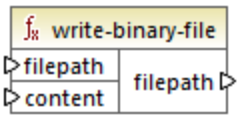
als BLOB dargestellt. Erst wenn Sie das Ergebnis der Funktion auf einen XML-Node vom Typ `xs:base64Binary` mappen, wird er tatsächlich base64-kodiert. Sie könnten das Ergebnis der Funktion auch auf `xs:hexBinary`, einen Datenbank-Blob oder ein Binärdatenfeld in einer Protocol Buffer-Struktur mappen.



Um eine Binärdatei in ein Mapping einzulesen, geben Sie als Input den Pfad zum Argument **filepath** an. Wenn der Dateipfad (**filepath**) relativ ist, sucht MapForce im selben Verzeichnis wie dem Mapping-Verzeichnis nach der Datei. Das Argument **must-exist** ist optional; wenn die Datei nicht geöffnet werden kann und dieser Parameter **true** ist, gibt das Mapping einen Fehler aus. Wenn die Datei nicht geöffnet werden kann und dieser Parameter **false** ist, wird eine leere Binärdatei zurückgegeben.

write-binary-file

Diese Funktion schreibt den Inhalt einer Binärdatei in den angegebenen Dateipfad und gibt den Pfad der geschriebenen Datei zurück. Wenn eine Binärdatei das einzige gewünschte Ergebnis ist, verbinden Sie das Ergebnis der Funktion mit einer einfachen Output[879]-Komponente. Da diese Funktion immer, wenn deren Ausgabe im Mapping verwendet wird, eine Datei schreibt, wird empfohlen, das Ergebnis der Funktion direkt mit einer Zielkomponente zu verbinden, ohne weiterer Verarbeitungen dazwischen durchzuführen.



Um Binärdateien in ein Mapping zu schreiben, geben Sie als Input für das Argument **filepath** ihren Pfad an. Wenn der Dateipfad (**filepath**) relativ ist, generiert MapForce die Datei im selben Verzeichnis wie dem Mapping-Verzeichnis. Das Argument **content** muss mit dem eigentlichen Binärdateninhalt (z.B. einem BLOB-Feld in einer Datenbank) verbunden werden.

Wenn Sie in MapForce eine Vorschau auf das Mapping anzeigen, generiert die Funktion standardmäßig temporäre Dateien, anstatt die Dateien direkt auf dem Datenträger zu speichern. Um die temporären Dateien zu speichern, klicken Sie zuerst auf das Register **Ausgabe** und anschließend je nach Bedarf auf eine der Symbolleisten-Schaltflächen **Generierte Ausgabe speichern** oder **Alle generierten Ausgaben speichern**.

Um MapForce so zu konfigurieren, dass die Ausgabe, anstatt in einer temporären Datei, direkt in einer endgültigen Datei gespeichert wird, wählen Sie den Menübefehl **Extras | Optionen**, klicken Sie auf **Allgemein** und aktivieren Sie die Option **Direkt in die endgültigen Output-Dateien schreiben**. Beachten Sie, dass mit dieser Option alle vorhandenen Dateien desselben Namens überschrieben werden.

Die Funktion gibt immer den endgültigen (und nicht den temporären) Dateinamen zurück, selbst wenn die endgültige Datei noch nicht auf dem Datenträger gespeichert wurde (Dies ist der Fall, wenn Sie eine Vorschau auf das Mapping anzeigen und die Option **Direkt in die endgültige Output-Datei schreiben** deaktiviert ist).

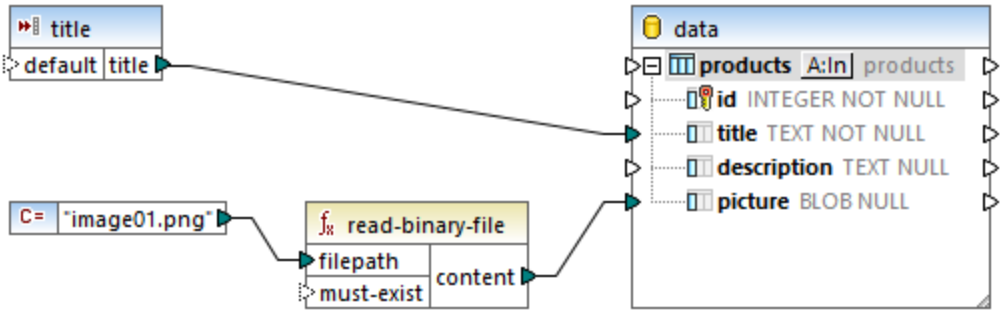
> Beachten Sie, dass dies bei einem Mapping, das seine eigene Ausgabedatei wieder ausliest, nicht unterstützt wird.

Beispiele

Siehe Beispiel: Lesen von Binärdateien[862] und Beispiel: Schreiben von Binärdateien[864].

4.11.1 Beispiel: Lesen von Binärdateien

Im unten gezeigten Mapping werden Daten aus einer Bilddatei gelesen und in eine Datenbanktabelle geschrieben. Die Zieldatenbank ist SQLite. Beachten Sie, dass der Datentyp des Datenbankfelds **picture** BLOB ist.

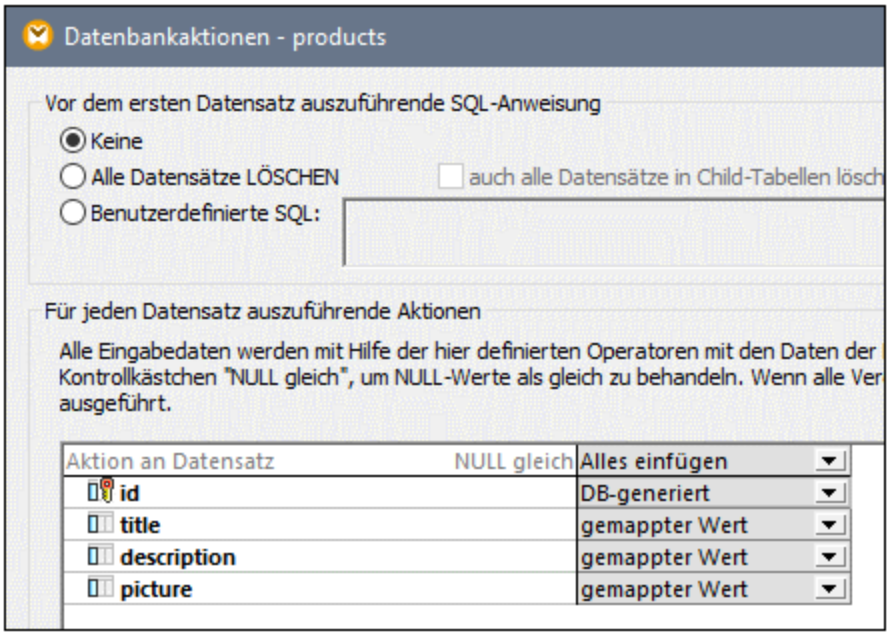


Zur Extrahierung von binärem Inhalt aus der Datei wurde die Funktion `read-binary-file` verwendet. Das erste Argument **filepath** in diesem Beispiel wird von einer Konstante bereitgestellt. Da der Pfad relativ ist, sucht MapForce im selben Verzeichnis wie dem Mapping-Verzeichnis nach der Bilddatei.

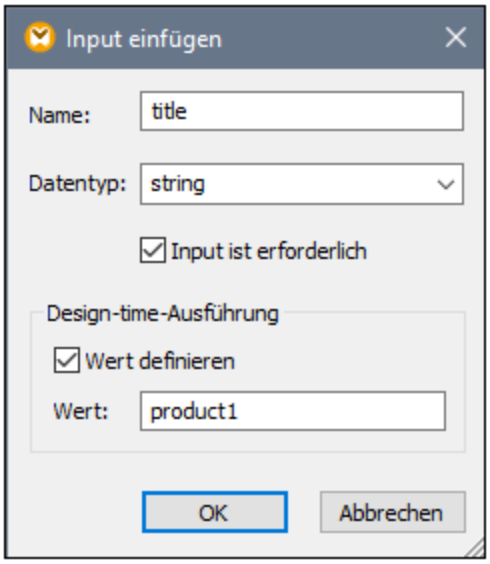
Durch das Mapping werden die folgenden Felder in der Zieldatenbank befüllt:

- **id** - in diesem Beispiel wird die Datenbankkomponente so konfiguriert, dass **id** von der Datenbank generiert und nicht über das Mapping bereitgestellt wird. Nähere Informationen finden Sie unter Einfügen von Daten in eine Tabelle[292].

© 2016-2022 Altova GmbH



- **title** - Dieser Wert stammt aus einer einfachen Input-Komponente desselben Namens. Beachten Sie, dass ein Design-Zeit-Ausführungswert gesetzt wird ("product1"), damit Sie eine Vorschau auf das Mapping anzeigen können. Nähere Informationen dazu finden Sie unter Bereitstellen von Parametern für das Mapping 867.



- **picture** - dieses Feld erhält seinen Ausgabewert direkt aus der `read-binary-file`-Funktion.

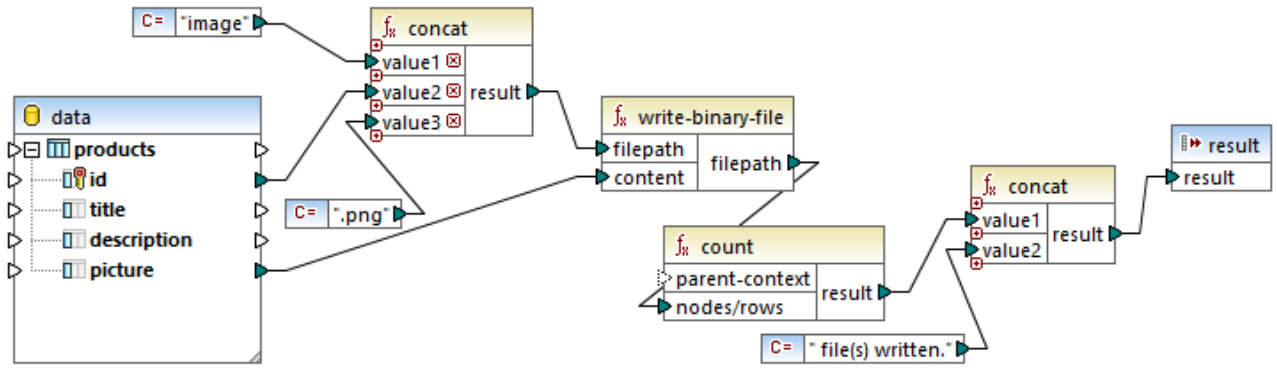
Da es sich bei der Zielkomponente um eine Datenbank handelt, wird bei der Anzeige der Mapping-Vorschau ein Pseudo-SQL-Script generiert, das Sie anzeigen, aber dazu nicht verwenden können, um Änderungen in die Datenbank zu schreiben. Um das eigentliche Script an der Datenbank auszuführen, wählen Sie den Menübefehl **Ausgabe | SQL-Script ausführen.**

4.11.2 Beispiel: Schreiben von Binärdateien

Im unten gezeigten Mapping werden BLOB-Werte aus einer SQLite-Datenbank ausgelesen und es werden Bilddateien auf den Datenträger geschrieben. Die Datenbank hat eine Tabelle namens **products**, welche die folgenden Spalten (Felder) hat:

- **id** (*Integer*, die eindeutige Seriennummer des Datensatzes)
- **title** (*text*, der Titel des Produkts)
- **description** (*text*, die Produktbeschreibung)
- **picture** (*blob*, das Bild des Produkts)

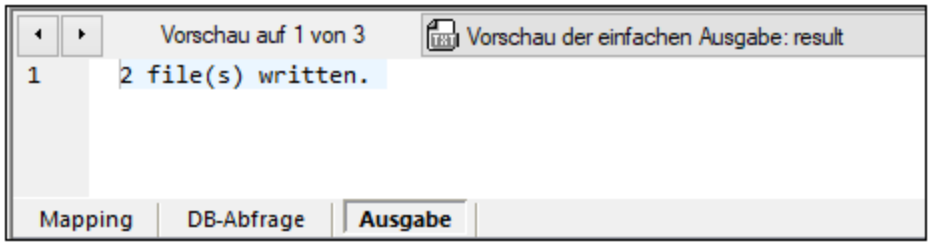
Für den Zweck dieses Beispiels sind nur die Felder **id** und **picture** relevant.



Ziel des Mappings ist es, alle Bilder aus der Tabelle **products** zu extrahieren und als Dateien auf dem Rechner zu speichern. Wie oben gezeigt, wird dazu die Funktion `write-binary-file` verwendet. Das erste Argument, **filepath**, erhält den Dateipfad für jedes einzelne Bild. Der Pfad muss eindeutig sein, damit keine Dateien überschrieben werden, daher wird er durch Verkettung der eindeutigen Datenbank-**id** der einzelnen Datensätze mit dem Wort "image" und Ergänzung der Dateierweiterung ".png" generiert (es wird angenommen, dass alle Bilder im PNG-Format sind). Beachten Sie, dass der Pfad relativ ist, d.h. die Dateien werden in dasselbe Verzeichnis, in dem sich das Mapping befindet, geschrieben.

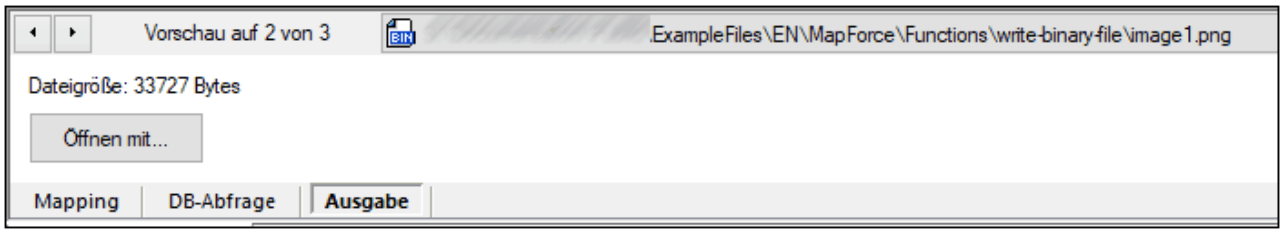
Das zweite Argument, **content**, erhält den in der Datenbank gespeicherten binären Inhalt.

Die `count`-Funktion gibt die Anzahl aller generierten Dateien zurück und kombiniert diese Zahl mit dem String "file(s) written". Dadurch erhalten Sie einen Bericht darüber, wie viele Dateien vom Mapping tatsächlich generiert wurden. In diesem Beispiel enthält die Datenbank nur zwei Produktdatensätze, daher die folgende Mapping-Ausgabe:



 © 2016-2022 Altova GmbH

Wie zuvor erwähnt, generiert diese Funktion bei Ausführung der Mapping-Vorschau temporäre Dateien. Um eine Vorschau auf die einzelnen Dateien anzuzeigen, klicken Sie auf die Pfeilschaltflächen, um zum gewünschten Datensatz zu navigieren, klicken Sie auf die Schaltfläche **Öffnen mit** und wählen Sie ein Bildbearbeitungsprogramm aus.



Um alle Dateien zu speichern, klicken Sie je nach Bedarf auf eine der Symbolleisten-Schaltflächen **Generierte Ausgabe speichern** oder **Alle generierten Ausgaben speichern**.

5 Transformationskomponenten

In diesem Abschnitt werden Transformationskomponenten zur Transformation von Daten oder zum temporären Speichern von Daten für die weitere Verarbeitung beschrieben. Im Folgenden finden Sie eine Liste von Transformationskomponenten:

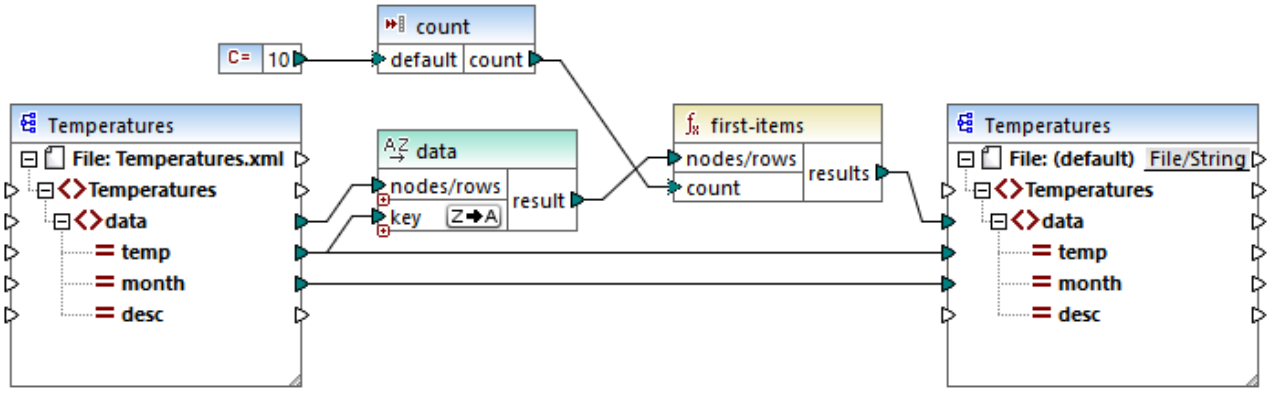
- Einfache Input-Komponente 867
- Einfache Output-Komponente 878
- Variablen 882
- Verknüpfen von Komponenten mittels Join 896
- Sortieren von Komponenten 908
- Filter und Bedingungen 914
- Wertezuordnungen 921
- Gruppierungsfunktionen 931
- Ausnahmeereignisse 938

Beachten Sie, dass auch Funktionen zu den Transformationskomponenten gehören. Funktionen 942 werden jedoch in einem eigenen Abschnitt beschrieben.

© 2016-2022 Altova GmbH

5.1 Einfache Input-Komponente

Wenn Sie ein Mapping erstellen müssen, das als Input Parameter erhält, können Sie das durch Hinzufügen einer speziellen Input-Komponente namens "einfache Input-Komponente" bewerkstelligen. Einfache Input-Komponenten haben immer einen einfachen Datentyp (z.B. String, Ganzzahl usw.) anstelle einer Struktur von Datenelementen und Sequenzen. So enthält etwa das Mapping unten die einfache Input-Komponente **count**. Sie hat die Aufgabe, die maximale Zeilenanzahl, die aus der XML-Quelldatei abgerufen werden soll (mit dem Wert **10** als Standardeinstellung), in Form eines Parameters bereitzustellen. Beachten Sie, dass die als Input für die Funktion first-items[1081] bereitgestellten Nodes mit Hilfe einer Sortierkomponente sortiert werden, sodass im Mapping nur die höchsten *N* Temperaturen ausgegeben werden, wobei *N* der Wert des Parameters ist.



FindHighestTemperatures.mfd

Eine weitere häufige Einsatzmöglichkeit für einfache Input-Komponenten ist die Bereitstellung eines Dateinamens für das Mapping. Dies ist in Mappings, in denen Daten dynamisch aus Input-Dateien ausgelesen oder in Output-Dateien geschrieben werden sollen, hilfreich, siehe Dynamische Verarbeitung mehrerer Input- oder Output-Dateien[1371].

Sie können einfache Input-Komponenten in jeder der folgenden MapForce-Transformationssprachen verwenden:

- BUILT-IN (wenn Sie direkt in MapForce über das Register **Vorschau** eine Vorschau der Mapping-Transformation anzeigen)
- BUILT-IN (bei Ausführung der MapForce Server-Ausführungsdatei)
- SLT 1.0, XSLT 2.0, XSLT 3.0
- XQuery
- C++
- C#
- Java

In Fall von mit MapForce Server oder über generierten Code ausgeführten Mappings werden einfache Input-Komponenten zu Befehlszeilenparametern. Bei Mappings, die als XSLT-Transformationen generiert werden, entsprechen einfache Input-Komponenten Stylesheet-Parametern in der generierten XSLT-Datei.

Sie können jede einzelne Input-Komponente (oder jeden Parameter) als optional oder obligatorisch erstellen (siehe Hinzufügen von einfachen Input-Komponenten[868]). Bei Bedarf können Sie auch Standardwerte für die Input-Mapping-Parameter erstellen (siehe Erstellen eines Input-Standardwerts[871]). Dadurch können Sie das Mapping ohne Probleme ausführen, selbst wenn Sie während der Ausführung des Mappings nicht explizit einen

Parameterwert bereitstellen. Ein Beispiel dazu finden Sie unter Beispiel: Verwenden von Dateinamen als Mapping-Parameter[872].

Input-Parameter, die zum Mapping-Bereich hinzugefügt werden, sind nicht mit Input-Parametern in benutzerdefinierten Funktionen[967] zu verwechseln. Die beiden Parameterarten weisen die folgenden Ähnlichkeiten und Unterschiede auf:

Input-Parameter im Mapping-Bereich	Input-Parameter von benutzerdefinierten Funktionen
Werden über das Menü **Funktion | Input einfügen** hinzugefügt.	Werden über das Menü **Funktion | Input einfügen** hinzugefügt.
Können einfache Datentypen haben (String, Ganzzahl, usw.).	Können sowohl einfache als auch komplexe Datentypen haben.
Anwendbar auf das gesamte Mapping.	Können nur im Kontext der Funktion, in der sie definiert wurden, angewendet werden.

Wenn Sie (mit dem Menübefehl **Extras | Umgekehrtes Mapping erstellen**) ein umgekehrtes Mapping erstellen, wird eine einfache Input-Komponente zu einer einfachen Output-Komponente.

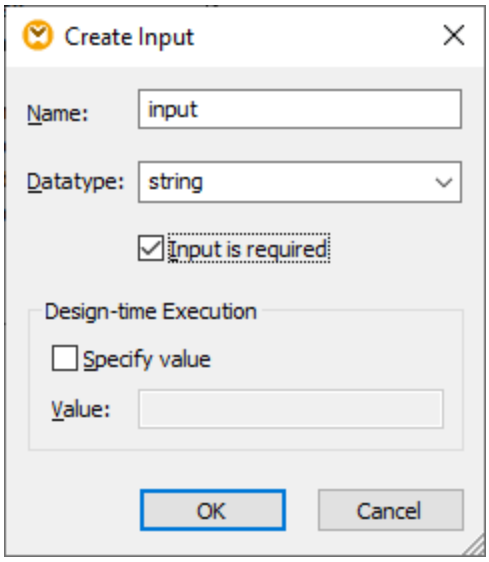
5.1.1 Hinzufügen von einfachen Input-Komponenten

So fügen Sie eine einfache Input-Komponente zum Mapping hinzu:

1. Stellen Sie sicher, dass im Mapping-Fenster das Hauptmapping (und keine benutzerdefinierte Funktion) angezeigt wird.
2. Wählen Sie eine der folgendem Methoden:

   - Klicken Sie im Menü **Funktion** auf **Input-Komponente einfügen**.
   - Klicken Sie im Menü **Einfügen** auf **Input-Komponente einfügen**.
   - Klicken Sie auf die Symbolleisten-Schaltfläche **Input-Komponente einfügen** .

© 2016-2022 Altova GmbH



3. Geben Sie einen Namen ein und wählen Sie den gewünschten Datentyp für diesen Input aus. Wenn der Input als zwingend erforderlicher Mapping-Parameter behandelt werden soll, aktivieren Sie das Kontrollkästchen **Input ist erforderlich**. Eine vollständige Liste der Einstellungen finden Sie unter Einstellungen für einfache Input-Komponenten 869.

Anmerkung: Der Parametername darf nur Buchstaben, Zahlen und Unterstrichzeichen enthalten. Andere Zeichen sind nicht zulässig. Dadurch funktioniert das Mapping in allen Codegenerierungssprachen.

4. Klicken Sie auf **OK.**

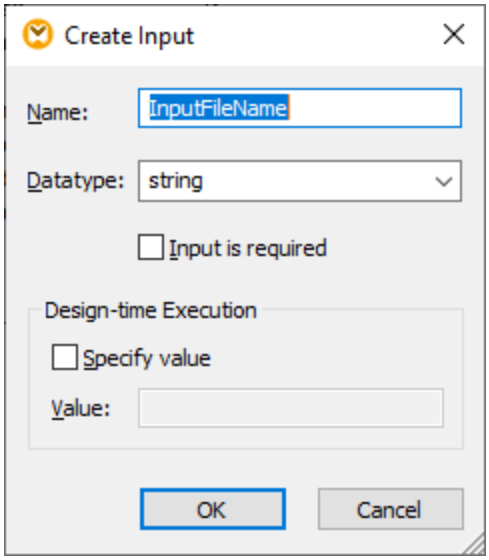
Sie können jede der hier definierten Einstellungen später ändern (siehe Einstellungen für einfache Input-Komponenten 869).

5.1.2 Einstellungen für einfache Input-Komponenten

Sie können die Einstellungen für eine einfache Input-Komponente beim Hinzufügen der Komponente zum Mapping-Bereich definieren. Sie können die Einstellungen später über das Dialogfeld "Input bearbeiten" jederzeit ändern.

Um das Dialogfeld "Input bearbeiten" zu öffnen, wählen Sie eine der folgenden Methoden:

- Wählen Sie die Komponente aus und klicken Sie im Menü **Komponente** auf **Eigenschaften**.
- Doppelklicken Sie auf die Komponente.
- Klicken Sie mit der rechten Maustaste auf die Komponente und wählen Sie den Befehl **Eigenschaften**.



Dialogfeld "Input bearbeiten"

Es stehen die folgenden Einstellungen zur Verfügung.

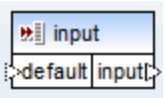
Name	Geben Sie einen beschreibenden Namen für den Input-Parameter, der dieser Komponente entspricht, ein. Zum Zeitpunkt der Mapping-Ausführung wird der in dieses Textfeld eingegebene Wert der Name des für das Mapping bereitgestellten Parameters; es sind daher keine Leer- oder Sonderzeichen zulässig.
Datentyp	Standardmäßig wird der Input-Parameter als String-Datentyp behandelt. Wenn der Parameter einen anderen Datentyp haben soll, wählen Sie den entsprechenden Wert aus der Liste aus. MapForce konvertiert den Input-Parameter bei der Ausführung des Mappings in den hier ausgewählten Datentyp.
Input ist erforderlich	Wenn dieses Kontrollkästchen aktiviert ist, wird der Input-Parameter zu einem zwingend erforderlichen Parameter, d.h. das Mapping kann nur dann ausgeführt werden, wenn Sie einen Parameterwert angeben. Deaktivieren Sie dieses Kontrollkästchen, wenn Sie einen Standardwert für den Input-Parameter definieren möchten (siehe Erstellen eines Input-Standardwerts 871).
Wert definieren	Diese Einstellung wird nur angewendet, wenn Sie das Mapping während des Designs durch Klicken auf das Register **Vorschau** ausführen. Über diese Einstellung können Sie den als Mapping-Input zu verwendenden Wert direkt in der Komponente eingeben.
Wert	Diese Einstellung wird nur angewendet, wenn Sie das Mapping während des Designs durch Klicken auf das Register **Vorschau** ausführen. Um den gewünschten Wert für MapForce einzugeben, aktivieren Sie das Kontrollkästchen **Wert definieren** und geben Sie anschließend den gewünschten Wert ein. **Anmerkung:** Wenn Sie das Kontrollkästchen **Wert definieren** aktivieren und in das benachbarte Feld einen Wert eingeben, hat der eingegebene Wert Vorrang vor dem Standardwert, wenn Sie eine Vorschau auf das Mapping

© 2016-2022 Altova GmbH

	anzeigen (d.h. bei der Ausführung während des Designs). Der Design-Zeit-Wert hat jedoch im generierten XSLT-, XQuery- oder Programmcode, bei der Ausführung durch MapForce Server oder bei Bereitstellung auf FlowForce Server keine Auswirkung.

5.1.3 Erstellen eines Input-Standardwerts

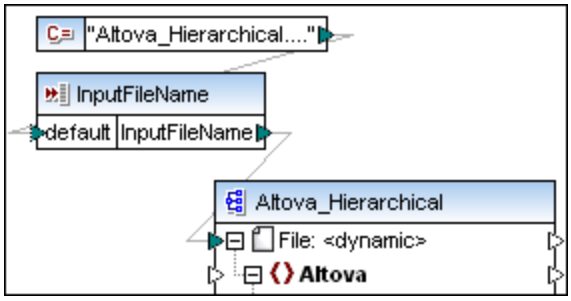
Beachten Sie nach Hinzufügen einer Input-Komponente zum Mapping-Bereich das Datenelement **default** links von der Komponente.



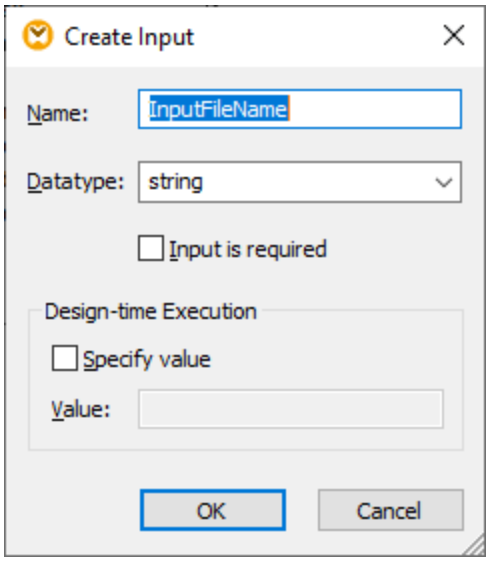
Einfache Input-Komponente

Über das Datenelement **default** können Sie wie folgt einen optionalen Standardwert zu dieser Input-Komponente hinzufügen:

1. Fügen Sie eine Konstantenkomponente hinzu (Menü **Einfügen | Konstante)** und verbinden Sie diese mit dem Datenelement **default** der Input-Komponente.



2. Doppelklicken Sie auf die Input-Komponente und deaktivieren Sie das Kontrollkästchen **Input ist erforderlich**. Wenn Sie einen Input-Standardwert erstellen, hat diese Einstellung keine Bedeutung und verursacht Mapping-Validierungswarnungen.



3. Klicken Sie auf **OK.**

Anmerkung: Wenn Sie das Kontrollkästchen **Wert definieren** aktivieren und in das benachbarte Feld einen Wert eingeben, hat der eingegebene Wert Vorrang vor dem Standardwert, wenn Sie eine Vorschau auf das Mapping anzeigen (d.h. bei der Ausführung während des Designs). Der Design-Zeit-Wert hat jedoch im generierten XSLT-, XQuery- oder Programmcode, bei der Ausführung durch MapForce Server oder bei Bereitstellung auf FlowForce Server keine Auswirkung.

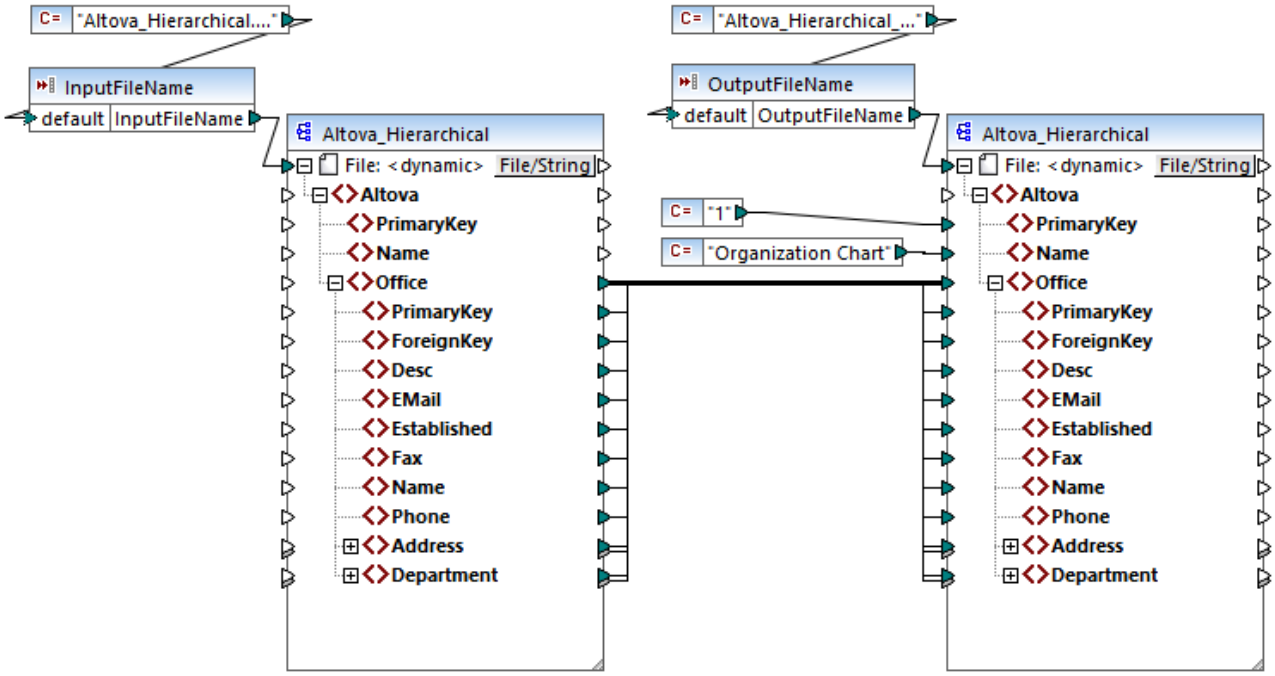
5.1.4 Beispiel: Verwenden von Dateinamen als Mapping-Parameter

In diesem Beispiel wird Schritt für Schritt beschrieben, wie Sie ein Mapping ausführen, das zur Laufzeit Input-Parameter erhält. Sie finden die in diesem Beispiel verwendete Mapping-Design-Datei unter dem Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\FileNamesAsParameters.mfd**.

In diesem Mapping werden Daten aus einer XML-Quelldatei ausgelesen und in eine XML-Zieldatei geschrieben. Die Daten werden beinahe unverändert in die Zieldatei geschrieben; nur die Attribute **PrimaryKey** und **Name** werden mit Konstantenwerten aus dem Mapping befüllt. Hauptaufgabe des Mappings ist, dem Aufrufenden eine Möglichkeit zu geben, die Namen der Input- und Output-Datei zur Mapping-Laufzeit in Form von Mapping-Parametern anzugeben.

Das Mapping hat zu diesem Zweck zwei Input-Komponenten: **InputFileName** und **OutputFileName**. Diese Komponenten stellen den Input-Dateinamen (bzw. den Output-Dateinamen) der XML-Quell- und der XML-Zieldatei bereit. Aus diesem Grund wurden Sie mit dem Datenelement **Datei:<dynamisch>** verbunden. Durch Klicken auf die Schaltfläche **Datei/String** (Datei/String) und Auswahl der Option **Über das Mapping bereitgestellte dynamische Dateinamen verwenden** können Sie eine Komponente in diesen Modus schalten.

© 2016-2022 Altova GmbH



FileNamesAsParameters.mfd (MapForce Enterprise Edition)

Wenn Sie auf die Titelleiste einer der Komponenten (**InputFileName** oder **OutputFileName**) doppelklicken, sehen Sie deren Eigenschaften. So können Sie etwa den Datentyp des Input-Parameters definieren oder den Namen des Input-Parameters ändern, wie unter Einstellungen für einfache Input-Komponenten[869] beschrieben. In diesem Beispiel sind die Input- und Output-Parameter folgendermaßen konfiguriert:

- Der Parameter **InputFileName** hat den Typ "string" und hat einen Standardwert, der von einer im selben Mapping definierten Konstante stammt. Die Konstante hat den Typ "string" und den Wert "Altova_Hierarchical.xml". Bei Ausführung dieses Mappings wird daher versucht, Daten aus einer Datei namens "Altova_Hierarchical.xml" zu lesen, vorausgesetzt Sie geben keinen anderen Wert als Parameter an.
- Der Parameter **OutputFileName** hat den Typ "string" und hat einen Standardwert, der von einer im selben Mapping definierten Konstante stammt. Die Konstante hat den Typ "string" und den Wert "Altova_Hierarchical_output.xml". Das Mapping erstellt daher bei seiner Ausführung eine XML-Ausgabedatei namens "Altova_Hierarchical_output.xml", vorausgesetzt Sie geben keinen anderen Wert als Parameter an.

In den folgenden Abschnitten wird beschrieben, wie Sie das Mapping ausführen und in den folgenden Transformationssprachen Parameter bereitstellen:

- XSLT 2.0[874] mit Hilfe von RaptorXML Server
- Built-in (MapForce Server-Ausführungsdatei)[874] mit Hilfe von MapForce Server
- Java[875]
- C#[876]
- C++[876]

XSLT 2.0

Wenn Sie Code in XSLT 1.0, XSLT 2.0 oder XSLT 3.0 generieren, wird im gewählten Zielverzeichnis zusätzlich zur XSLT-Datei die Batch-Datei **DoTransform.bat** generiert. Mit Hilfe von **DoTransform.bat** können Sie das Mapping mit RaptorXML Server ausführen, siehe Automatisierung mit RaptorXML Server 1433.

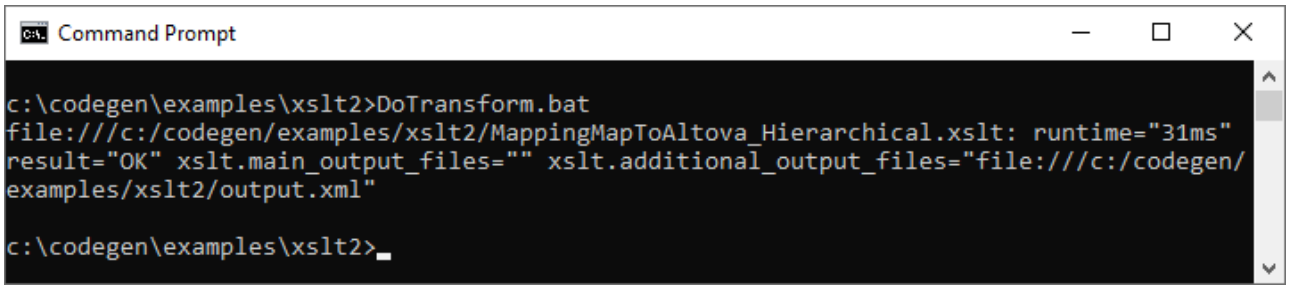
Um eine andere Input (oder Output)-Datei zu verwenden, bearbeiten Sie die Datei **DoTransform.bat**, damit Sie die erforderlichen Parameter enthält. Gehen Sie dazu folgendermaßen vor:

1. Generieren Sie zuerst den XSLT-Code (Um z.B. XSLT 2.0 zu generieren, klicken Sie im Menü **Datei** auf **Code generieren in | XSLT 2.0**).
2. Kopieren Sie die Datei **Altova_Hierarchical.xml** aus **<Dokumente>\Altova\MapForce2023\MapForceExamples** in das Verzeichnis, in dem Sie XSLT 2.0-Code generiert haben (in diesem Beispiel **c:\codegen\examples\xslt2**). Wie zuvor erwähnt, versucht das Mapping diese Datei zu lesen, wenn Sie keinen benutzerdefinierten Wert für den Parameter **InputFileName** bereitstellen.
3. Bearbeiten Sie **DoTransform.bat,** sodass diese den benutzerdefinierten Input-Parameter entweder vor oder nach **%*** enthält. Beachten Sie, dass der Parameterwert innerhalb von einfache Anführungszeichen gesetzt wird. Die verfügbaren Input-Parameter werden im Abschnitt **rem** (Remark) aufgelistet. Angenommen, Sie möchten eine Output-Datei mit dem Namen **output.xml** generieren. Ändern Sie dazu die Datei **DoTransform.bat** folgendermaßen:

```
@echo off

RaptorXML xslt --xslt-version=2
  --input="MappingMapToAltova_Hierarchical.xslt"
  --param=OutputFileName:'output.xml' %* "MappingMapToAltova_Hierarchical.xslt"
rem --param=InputFileName:
rem --param=OutputFileName:
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

Wenn Sie die Datei **DoTransform.bat** ausführen, stellt RaptorXML Server die Transformation mit Hilfe von **Altova_Hierarchical.xml** als Input fertig. Wenn Sie die obige Anleitung befolgt haben, erhält die generierte Ausgabedatei den Namen **output.xml**.



MapForce Server-Ausführungsdatei

So stellen Sie für eine MapForce Server-Ausführungsdatei benutzerdefinierte Input-Parameter zur Verfügung:

1. Öffnen Sie das Beispiel **FileNamesAsParameters.mfd** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**, falls Sie das noch nicht getan haben.

© 2016-2022 Altova GmbH

2. Klicken Sie im Menü **Datei** auf **Zu MapForce Server-Ausführungsdatei kompilieren**, siehe Kompilieren von Mappings zu MapForce Server-Ausführungsdateien[1442]. Wenn Sie dazu aufgefordert werden, speichern Sie die .mfx-Ausführungsdatei in einem Verzeichnis auf Ihrem Rechner (in diesem Beispiel unter **c:\codegen\examples\mfx**).
3. Kopieren Sie die Datei **Altova_Hierarchical.xml** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples** in das Verzeichnis, in dem Sie die .mfx-Datei gespeichert haben.
4. Führen Sie MapForce Server mit dem folgenden Befehl aus:

```
MapForceServer.exe run "C:\codegen\examples\mfx\FileNamesAsParameters.mfx"
  -p=InputFileName:"C:\codegen\examples\mfx\Altova_Hierarchical.xml"
  -p=OutputFileName:"C:\codegen\examples\mfx\OutputFile.xml"
```

In obigen MapForce Server-Befehl sind `-p=InputFileName` und `-p=OutputFileName` die Input-Parameter für das Mapping. Sie können als Wert von **-OutputFileName** jeden beliebigen Dateinamen verwenden. Der im Parameter **-InputFileName** bereitgestellte Dateiname muss jedoch als physische Datei vorhanden sein, da das Mapping sonst fehlschlägt.

Anmerkung: Wenn Sie die Meldung "MapForceServer.exe wird nicht als interner oder externer Befehl, Programm oder Batch-Datei erkannt", ändern Sie das aktuelle Verzeichnis in das Verzeichnis, in dem die ausführbare MapForce Server-Datei installiert ist. Damit Sie den Pfad nicht bei jeder Mapping-Ausführung ändern müssen, fügen Sie den Pfad zum Verzeichnis, in dem die ausführbare MapForce Server-Datei installiert ist (z.B. **C:\Programme (x86)\Altova\MapForceServer2023\bin**), zur PATH-Umgebungsvariablen Ihres Betriebssystems hinzu.

Mit MapForce Server kann ein Mapping auch durch Aufruf der MapForce Server API (die von Sparchen wie C++, C# oder Java aus aufgerufen werden kann) ausgeführt werden. Nähere Informationen zu diesem Szenario finden Sie in der MapForce Server-Dokumentation (https://www.altova.com/de/documentation).

Java

So stellen Sie für eine Java .jar-Applikation einen benutzerdefinierten Input-Parameter zur Verfügung:

1. Öffnen Sie das Beispiel **FileNamesAsParameters.mfd** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**, falls Sie das noch nicht getan haben.
2. Klicken Sie im Menü **Datei** auf den Befehl **Code generieren in | Java**.
3. Kompilieren Sie den Java-Code zu einer ausführbaren JAR-Datei. Ein Beispiel dafür in Eclipse finden Sie unter Beispiel: Generieren und Ausführen von Java-Code.[1586])
4. Kopieren Sie die Datei **Altova_Hierarchical.xml** aus **<Dokumente>\Altova\MapForce2023\MapForceExamples** in das Verzeichnis, in dem sich die .jar-Datei befindet. Wie zuvor erwähnt, versucht das Mapping diese Datei zu lesen, wenn Sie keinen benutzerdefinierten Wert für den Parameter **InputFileName** bereitstellen.
5. Führen Sie die Java-Applikation mit dem folgenden Befehl aus:

```
java -jar Mapping.jar /OutputFileName "output.xml"
```

Im obigen Befehl stellt der Parameter `/OutputFileName` den Namen der zu generierenden Output-Datei bereit.

Anmerkung: Wenn Sie bei der Übergabe der Parameter an .jar-Dateien Platzhalterzeichen verwenden, setzen Sie diese Platzhalterzeichen bitte in Anführungszeichen, z.B:

```
java -jar Mapping.jar /InputFileName "altova-*.xml"
```

C#

So stellen Sie für eine von MapForce generierte C#-Befehlszeilenapplikation einen benutzerdefinierten Input-Parameter zur Verfügung:

1. Öffnen Sie das Beispiel **FileNamesAsParameters.mfd** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**, falls Sie das noch nicht getan haben.
2. Klicken Sie im Menü **Datei** auf **Code generieren in | C#** und wählen Sie ein Zielverzeichnis aus (in diesem Beispiel **C:\codegen\examples\cs**).
3. Öffnen Sie die Projektmappe in Visual Studio und erstellen Sie sie mit Build (**Strg + Umschalt + B**).
4. Kopieren Sie die Datei **Altova_Hierarchical.xml** aus **<Dokumente>\Altova\MapForce2023\MapForceExamples** in das Verzeichnis, in dem **Mapping.exe** generiert wurde (in diesem Beispiel **C:\codegen\examples\cs\Mapping\bin\Debug**). Wie zuvor erwähnt, versucht das Mapping diese Datei zu lesen, wenn Sie keinen benutzerdefinierten Wert für den Parameter **InputFileName** bereitstellen.
5. Öffnen Sie ein Eingabeaufforderungsfenster.und wechseln Sie in das Verzeichnis, in dem sich **Mapping.exe** befindet.

   ```
   cd C:\codegen\examples\cs\Mapping\bin\Debug
   ```

6. Führen Sie die Applikation mit dem folgenden Befehl aus:

   ```
   Mapping.exe /OutputFileName output.xml
   ```

Im obigen Befehl stellt der Parameter `/OutputFileName` den Namen der zu generierenden Output-Datei bereit.

C++

So stellen Sie für eine von MapForce generierte C++-Befehlszeilenapplikation einen benutzerdefinierten Input-Parameter zur Verfügung:

1. Öffnen Sie das Beispiel **FileNamesAsParameters.mfd** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**, falls Sie das noch nicht getan haben.
2. Klicken Sie im Menü **Datei** auf **Code generieren in | C++** und wählen Sie ein Zielverzeichnis aus (in diesem Beispiel **C:\codegen\examples\cpp**).
3. Öffnen Sie die Projektmappe in Visual Studio und erstellen Sie sie mit Build (**Strg + Umschalt + B**).
4. Kopieren Sie die Datei **Altova_Hierarchical.xml** aus **<Dokumente>\Altova\MapForce2023\MapForceExamples** in das Verzeichnis, in dem **Mapping.exe** generiert wurde (in diesem Beispiel **C:\codegen\examples\cpp\Mapping\Debug**). Wie zuvor erwähnt, versucht das Mapping diese Datei zu lesen, wenn Sie keinen benutzerdefinierten Wert für den Parameter **InputFileName** bereitstellen.
5. Öffnen Sie ein Eingabeaufforderungsfenster.und wechseln Sie in das Verzeichnis, in dem sich **Mapping.exe** befindet.

   ```
   cd C:\codegen\examples\cpp\Mapping\Debug
   ```

6. Führen Sie die Applikation mit dem folgenden Befehl aus:

© 2016-2022 Altova GmbH

```
Mapping.exe /OutputFileName output.xml
```

Im obigen Befehl stellt der Parameter `/OutputFileName` den Namen der zu generierenden Output-Datei bereit.

5.2 Einfache Output-Komponente

Eine Output-Komponente (oder "einfacher Output") ist eine MapForce Komponente, über die ein String-Wert aus dem Mapping zurückgegeben wird. Output-Komponenten sind nur eine mögliche Art von Zielkomponenten[76], sind aber nicht damit zu verwechseln. Verwenden Sie eine einfache Output-Komponente, wenn Sie einen String-Wert anhand des Mappings zurückgeben möchten. Einfache Output-Komponenten spielen im Mapping-Bereich die Rolle einer Zielkomponente mit einem String-Datentyp anstelle einer Struktur von Datenelementen und Sequenzen. Infolgedessen können Sie anstelle (oder zusätzlich zu) einer dateibasierten Zielkomponente eine einfache Output-Komponente erstellen. So können Sie eine einfache Output-Komponente z.B. verwenden, um die Ausgabe einer Funktion schnell zu testen und eine Vorschau des Ergebnisses anzuzeigen (siehe Beispiel: Testen der Funktionsausgabe[880]). Dieses Verfahren eignet sich auch für Mappings, in denen String-Serialisierung verwendet wird (siehe Parsen und Serialisieren von Strings[1382]). Der Hauptzweck einer einfachen Output-Komponente ist jedoch, beim Aufruf der MapForce Server API einen String zu erhalten, ohne Dateien schreiben zu müssen.

Einfache Output-Komponenten sollten nicht mit Output-Parametern in benutzerdefinierten Funktionen verwechselt werden (siehe Benutzerdefinierte Funktionen[967]). Die beiden Parameterarten weisen die folgenden Ähnlichkeiten und Unterschiede auf:

Output-Komponenten	Output-Parameter von benutzerdefinierten Funktionen
Werden über das Menü **Funktion | Output-Komponente einfügen** hinzugefügt.	Werden über das Menü **Funktion | Output-Komponente einfügen** hinzugefügt.
Haben "string" als Datentyp.	Können sowohl einfache als auch komplexe Datentypen haben.
Anwendbar auf das gesamte Mapping.	Können nur im Kontext der Funktion, in der sie definiert wurden, angewendet werden.

Bei Bedarf können Sie mehrere einfache Output-Komponenten zu einem Mapping hinzufügen. Sie können einfache Output-Komponenten auch in Kombination mit dateibasierten und datenbankbasierten Zielkomponenten verwenden. Wenn Ihr Mapping mehrere Zielkomponenten enthält, können Sie eine Vorschau der von einer bestimmten Komponenten zurückgegebenen Daten anzeigen, indem Sie in der Titelleiste der Komponente auf die Schaltfläche **Vorschau** () klicken und anschließend im Mapping-Fenster auf das Register **Ausgabe** klicken.

Sie können einfache Output-Komponenten folgendermaßen in MapForce-Transformationssprachen verwenden:

Sprache	Funktionsweise
BUILT-IN (bei Anzeige einer Vorschau auf die Mapping-Transformation)	Sie können von Output-Komponenten genau wie bei einer dateibasierten Mapping-Ausgabe eine Vorschau anzeigen, indem Sie im Mapping-Fenster auf das Register **Ausgabe** klicken.
BUILT-IN (bei Ausführung der MapForce Server-Ausführungsdatei)	Wenn Sie eine kompilierte MapForce Server-Ausführungsdatei ausführen (siehe Kompilieren eines MapForce Mappings[1442]), wird die Mapping-Ausgabe im Output-Standardstream (stdout) zurückgegeben, den Sie anzeigen oder in eine Datei weiterleiten können. Angenommen, der Name

© 2016-2022 Altova GmbH

Sprache	Funktionsweise
	der MapForce Server-Ausführungsdatei lautet **MyMapping.mfx**, so verwenden Sie die folgende Syntax, um die Mapping-Ausgabe an die Datei **output.txt** und etwaige Fehler an die Datei **log.txt** weiterzuleiten: `MapForceServer.exe run MyMapping.mfx >output.txt 2>log.txt`
XSLT 1.0, XSLT 2.0, XSLT 3.0	Wenn Sie XSLT-Dateien generieren, wird eine im Mapping definierte einfache Output-Komponente zur Ausgabe der XSLT-Transformation. Wenn Sie RaptorXML Server verwenden, können Sie RaptorXML Server anweisen, die Mapping-Ausgabe in die Datei zu schreiben, die als Wert an den Parameter `--output` übergeben wird. Um die Ausgabe in eine Datei zu schreiben, fügen Sie in der Datei **DoTransform.bat** den Parameter `--output` hinzu oder bearbeiten Sie ihn. Die folgende Datei **DoTransform.bat** wurde z.B. so bearbeitet, dass sie die Mapping-Ausgabe in die Datei **Output.txt** schreibt (siehe markierter Text). `RaptorXML xslt --xslt-version=2 --input="MappingMapToResult1.xslt" --output="Output.txt" %* "MappingMapToResult1.xslt"` Wenn kein `--output` Parameter definiert ist, wird die Mapping-Ausgabe bei Ausführung des Mappings in den Output-Standard-Stream (stdout) geschrieben.
C++, C#, Java	Die Mapping-Ausgabe im generierten C++-, C#- und Java-Code wird in die Standardausgabe der generierten Applikation geschrieben. Wenn das Mapping mehrere Zielkomponenten enthält, so verkettet die generierte Applikation die Standardausgabe jeder einzelnen Zielkomponente und gibt sie als eine einzige Standardausgabe zurück.

Wenn Sie (mit dem Menübefehl **Extras | Umgekehrtes Mapping erstellen**) ein umgekehrtes Mapping erstellen, wird eine einfache Output-Komponente zu einer einfachen Input-Komponente.

5.2.1 Hinzufügen einfacher Output-Komponenten

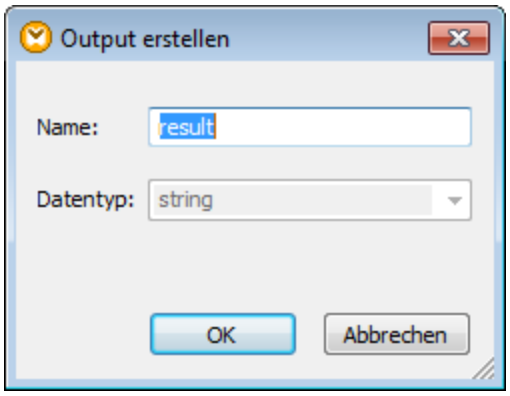
So fügen Sie eine Output-Komponente zum Mapping-Bereich hinzu:

1. Stellen Sie sicher, dass im Mapping-Fenster das Hauptmapping (und keine benutzerdefinierte Funktion) angezeigt wird.
2. Wählen Sie eine der folgenden Methoden:

a. Klicken Sie im Menü **Funktion** auf **Output-Komponente einfügen**.

b. Klicken Sie auf die Symbolleisten-Schaltfläche **Output-Komponente einfügen** .

3. Geben Sie einen Namen für die Komponente ein.
4. Klicken Sie auf **OK.**



Dialogfeld "Output erstellen"

Sie können den Komponentennamen später auf eine der folgenden Arten ändern:

- Wählen Sie die Komponente aus und klicken Sie im Menü **Komponente** auf **Eigenschaften**.
- Doppelklicken Sie auf den Komponententitel.
- Klicken Sie mit der rechten Maustaste auf den Komponententitel und wählen Sie **Eigenschaften**.

5.2.2 Beispiel: Vorschau auf die Funktionsausgabe

In diesem Beispiel wird gezeigt, wie Sie mit Hilfe einfacher Output-Komponenten eine Vorschau der von MapForce-Funktionen zurückgegebenen Ausgabe anzeigen können. Sie sollten bereits über ein grundlegendes Verständnis über Funktionen im Allgemeinen und MapForce-Funktionen im Besonderen verfügen. Wenn MapForce-Funktionen neu für Sie sind, lesen Sie bitte zuerst den Abschnitt Verwendung von Funktionen 942, bevor Sie fortfahren..

Wir wollen in diesem Beispiel eine Reihe von Funktionen zum Mapping-Bereich hinzufügen und lernen, wie man eine Vorschau ihrer Ausgabe mit Hilfe von einfachen Output-Komponenten anzeigt. In diesem Beispiel werden einige einfache Funktionen aus der core-Bibliothek verwendet. Hier sehen Sie eine Zusammenfassung ihrer Verwendung:

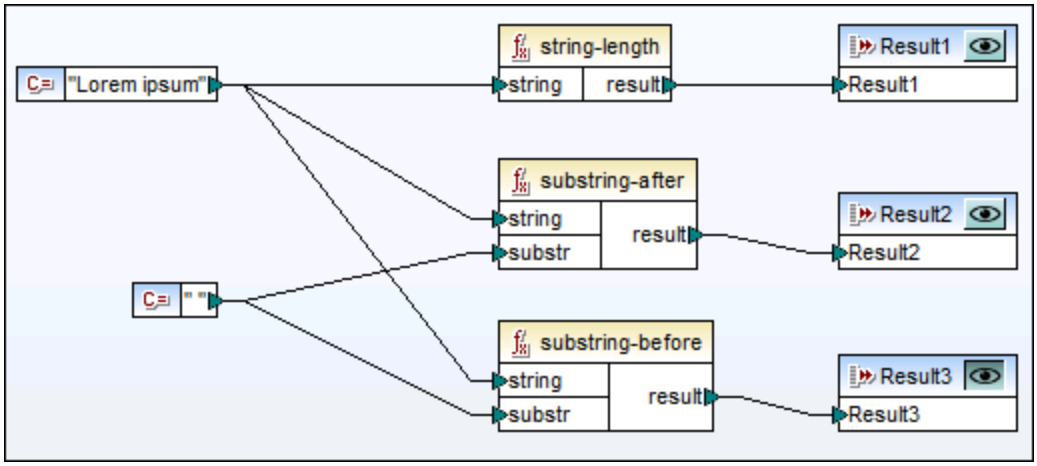
`string-length` 1109	Gibt die Anzahl der Zeichen in dem als Argument angegebenen String zurück. Wenn Sie z.B. den Wert "Lorem ipsum" an diese Funktion übergeben, so ist das Ergebnis "11", da dies die Anzahl der Zeichen im Text "Lorem ipsum" ist.
`substring-after` 1110	Gibt den Teil des String zurück, der hinter dem als Argument angegebenen Trennzeichen steht. Wenn Sie z.B. den Wert "Lorem ipsum" und das Trennzeichen (" ") an diese Funktion übergeben, ist das Ergebnis "ipsum".
`substring-before` 1111	Gibt den Teil des String zurück, der vor dem als Argument angegebenen Trennzeichen steht. Wenn Sie z.B. den Wert "Lorem ipsum" und das Trennzeichen ("

© 2016-2022 Altova GmbH

") an diese Funktion übergeben, ist das Ergebnis "Lorem".

Um jede dieser Funktionen anhand eines benutzerdefinierten Textwerts (in diesem Beispiel "Lorem ipsum") zu testen, gehen Sie folgendermaßen vor:

1. Fügen Sie (mit dem Menübefehl **Einfügen | Konstante**) eine Konstante mit dem Wert "Lorem ipsum" zum Mapping-Bereich hinzu. Die Konstante bildet den Input-Parameter für jede der zu testenden Funktionen.
2. Fügen Sie die Funktionen **`string-length`**, **`substring-after`** und **`substring-before`** zum Mapping-Bereich hinzu, indem Sie sie aus der core-Bibliothek, Abschnitt **string functions,** in den Mapping-Bereich ziehen.
3. Fügen Sie eine Konstante mit einem Leerzeichen (" ") als Wert hinzu. Dieses Leerzeichen dient als Trennzeichenparameter für die Funktionen **`substring-after`** und **`substring-before`**.
4. Fügen Sie (mit dem Menübefehl **Funktion | Output-Komponente einfügen)** drei einfache Output-Komponenten hinzu. In diesem Beispiel haben wir die Komponenten *Result1*, *Result2* und *Result3* genannt, Sie können aber auch einen anderen Namen wählen.
5. Verbinden Sie die Komponenten wie unten gezeigt.



Testen der Funktionsausgabe mit Hilfe von einfachen Output-Komponenten

Wie im Beispiel oben gezeigt, wird der String "Lorem ipsum" als Input-Parameter für die Funktionen **`string-length`**, **`substring-after`** und **`substring-before`** verwendet. Zusätzlich dazu erhalten die Funktionen **`substring-after`** und **`substring-before`** einen Leerzeichenwert als zweiten Input-Parameter. Mit Hilfe der Komponenten *Result1*, *Result2* und *Result3* können Sie eine Vorschau des Ergebnisses jeder Funktion anzeigen.

So zeigen Sie eine Vorschau der Ausgabe einer Funktion an:

- Klicken Sie in der Titelleiste der Komponente auf die Schaltfläche **Vorschau** () und klicken Sie anschließend im Mapping-Fenster auf das Register **Ausgabe**.

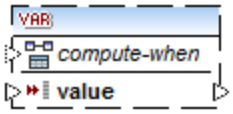
5.3 Variablen

Bei Variablen handelt es sich um eine spezielle Art von Komponente, in der ein Mapping-Zwischenergebnis für die weitere Verarbeitung gespeichert wird. Variablen können einen simpleType (z.B. String, Ganzzahl, Boolsche Werte, usw.) oder complexType (eine Baumstruktur) haben. Beispiele für diese beiden Arten finden Sie in den Unterabschnitten weiter unten.

Einer der wichtigsten Aspekte von Variablen ist, dass es sich dabei um Sequenzen handelt und dass sie zum Erstellen von Sequenzen verwendet werden können. Mit dem Begriff *Sequenz* wird eine Liste von null oder mehr Datenelementen bezeichnet. Dadurch können mit Hilfe einer Variable mehrere Datenelemente im Laufe des Mappings verarbeitet werden. Nähere Informationen dazu finden Sie auch unter Mapping-Regeln und Strategien 1330. Sie können einer Variablen auch ein Mal einen Wert zuweisen und diesen Wert im restlichen Mapping beibehalten. Nähere Informationen dazu finden Sie unter Ändern von Kontext und Geltungsbereich von Variablen 888.

Einfache Variablen

Eine einfache Variable dient zur Darstellung eines atomaren Typs wie z.B. Strings, Zahlen und Booleschen Werten (*siehe Abbildung unten*).



Komplexe Variablen

Eine komplexe Variable hat eine Baumstruktur. In der Liste unten sehen Sie, auf welchen Strukturen eine komplexe Variable basieren kann.

MapForce Basic Edition:

- XML-Schema-Struktur

MapForce Professional Edition:

- XML-Schema-Struktur
- Datenbankstruktur

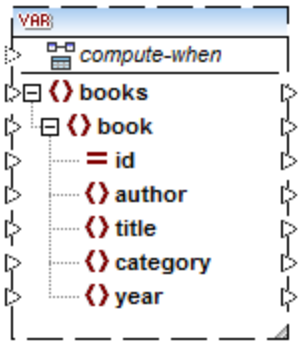
MapForce Enterprise Edition:

- XML-Schema-Struktur
- Datenbankstruktur
- EDI-Struktur
- FlexText Structure
- JSON-Schema-Struktur

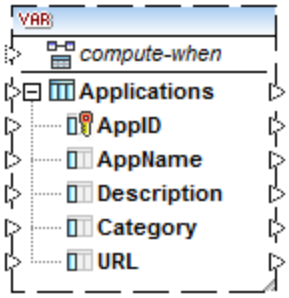
Beispiel 1: Auf einem XML-Schema basierende Variable

Sie können eine Variable vom Typ complexType erstellen, indem Sie ein XML-Schema bereitstellen, das die Struktur der Variablen beschreibt (*siehe Abbildung unten*). Wenn das Schema globale Elemente enthält, können Sie auswählen, welches davon den Root-Node der Variablenstruktur bilden soll. Beachten Sie, dass mit einer Variablen keine XML-Instanzdatei verknüpft ist. Die Daten der Variablen werden zur Mapping-Laufzeit berechnet.

© 2016-2022 Altova GmbH



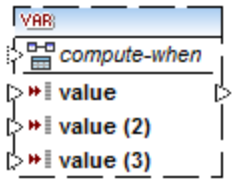
Beispiel 2: Auf einer Datenbank basierende Variable (MapForce Professional und Enterprise Edition)
Wenn Sie eine Datenbankstruktur für Ihre Variable auswählen (*siehe Abbildung unten*), können Sie eine bestimmte Datenbanktabelle als Root-Datenelement für die Variablenstruktur auswählen. Sie können in MapForce datenbankbasierte Variablen mit einer Struktur damit in Zusammenhang stehender Tabellen erstellen. Die Struktur dieser Tabellen bildet eine speicherresidente Struktur, die keine Verbindung zur Datenbank zur Laufzeit hat. Das heißt auch, dass es keine automatische Behandlung von Sekundärschlüsseln und keine Tabellenaktionen in Parametern oder Variablen gibt.



Compute-when
In beiden Beispielen oben hat jede Variable ein Datenelement namens `compute-when`. Die Verbindung mit diesem Datenelement ist optional. Dadurch können Sie festlegen, wie der Variablenwert im Mapping berechnet werden soll. Nähere Informationen dazu finden Sie unter Ändern von Kontext und Geltungsbereich von Variablen 888.

Variablen mit duplizierten Inputs

Falls nötig, können Datenelemente einer Variablenstruktur dupliziert werden, damit Daten aus mehr als einer Quellverbindung verbunden werden können. Dies ist ähnlich dem Duplizieren von Inputs 84 in Standardverbindungen. Dies gilt jedoch nicht für Variablen, die anhand von Datenbanktabellen erstellt wurden. In der Abbildung unten sehen Sie eine einfache Variable mit duplizierten Inputs.



Verkettete Mappings im Gegensatz zu Variablen

Variablen können mit Zwischenkomponenten eines verketteten Mappings[1299] verglichen werden. Sie sind jedoch flexibler und praktischer, wenn in den einzelnen Phasen des Mappings keine Zwischendateien erzeugt werden müssen. In der folgenden Tabelle wurden die Unterschiede zwischen Variablen und verketteten Mappings gegenübergestellt.

Verkettete Mappings	Variablen
Verkettete Mappings bestehen aus zwei voneinander unabhängigen Schritten. So kann ein Mapping z.B. aus den drei Komponenten A, B, und C bestehen. Schritt 1: Mappen der Daten A auf die Daten B. Schritt 2: Mappen der Daten von B auf C.	Sie können steuern, wann und wie oft der Variablenwert bei der Ausführung des Mappings berechnet wird. Nähere Informationen dazu finden Sie unter Ändern von Kontext und Geltungsbereich von Variablen[888].
Wenn das Mapping ausgeführt wird, werden die Zwischenergebnisse extern in Dateien gespeichert.	Wenn das Mapping ausgeführt wird, werden die Zwischenergebnisse intern gespeichert. Es werden keine externen Dateien, die die Ergebnisse einer Variablen enthalten, erzeugt.
Das Zwischenergebnis kann über die Vorschau-Schaltfläche in einer Vorschau angezeigt werden.	Sie können keine Vorschau auf das Ergebnis einer Variablen anzeigen, da diese zur Mapping-Laufzeit berechnet wird. Sie können die durch die Variable erzeugten Daten jedoch mit Hilfe des Debuggers[1409] anzeigen.

Anmerkung: Variablen werden nicht unterstützt, wenn als Mapping-Transformationssprache XSLT 1.0 ausgewählt ist.

5.3.1 Hinzufügen einer Variablen

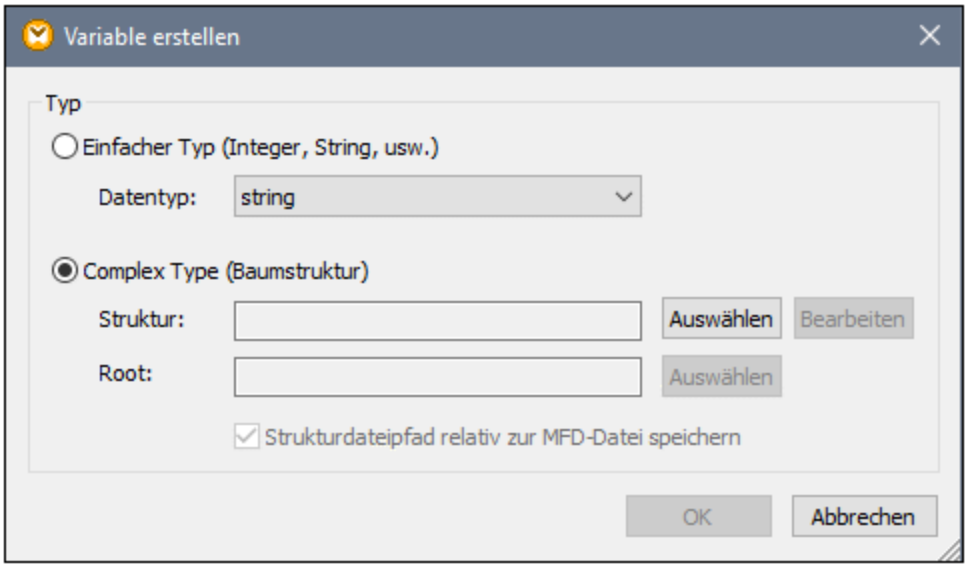
In diesem Kapitel wird erklärt, wie Sie eine Variable zu einem Mapping hinzufügen. Die erste Option ist, eine Variable über das Menü oder einen Symbolleistenbefehl hinzuzufügen. Bei der zweiten Option können Sie eine Variable über das Kontextmenü hinzufügen.

Option 1: über das Menü oder einen Symbolleistenbefehl

Mit Hilfe dieser Option können Sie eine Variable über das Menü oder einen Symbolleistenbefehl hinzuzufügen. Gehen Sie folgendermaßen vor:

© 2016-2022 Altova GmbH

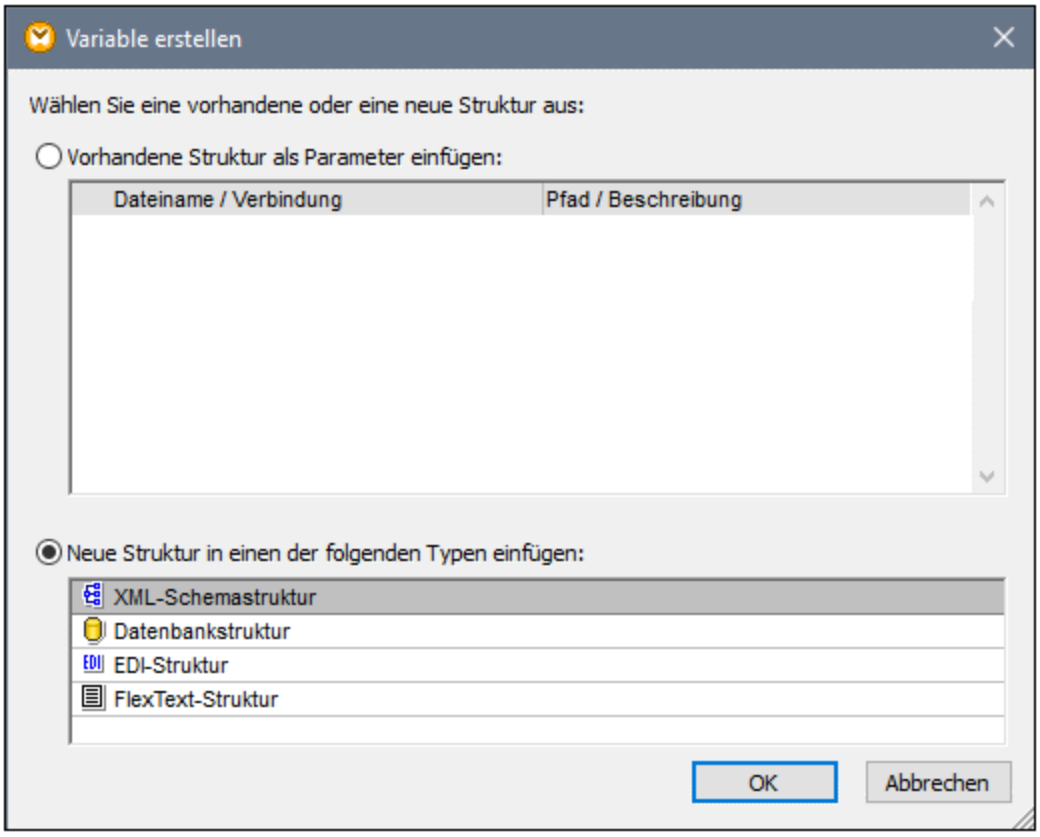
1. Gehen Sie zum Menü **Datei** und klicken Sie auf **Variable**. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche  (**Variable**).



2. Wählen Sie den Typ der gewünschten Variable aus (einfacher Typ oder complex Type).

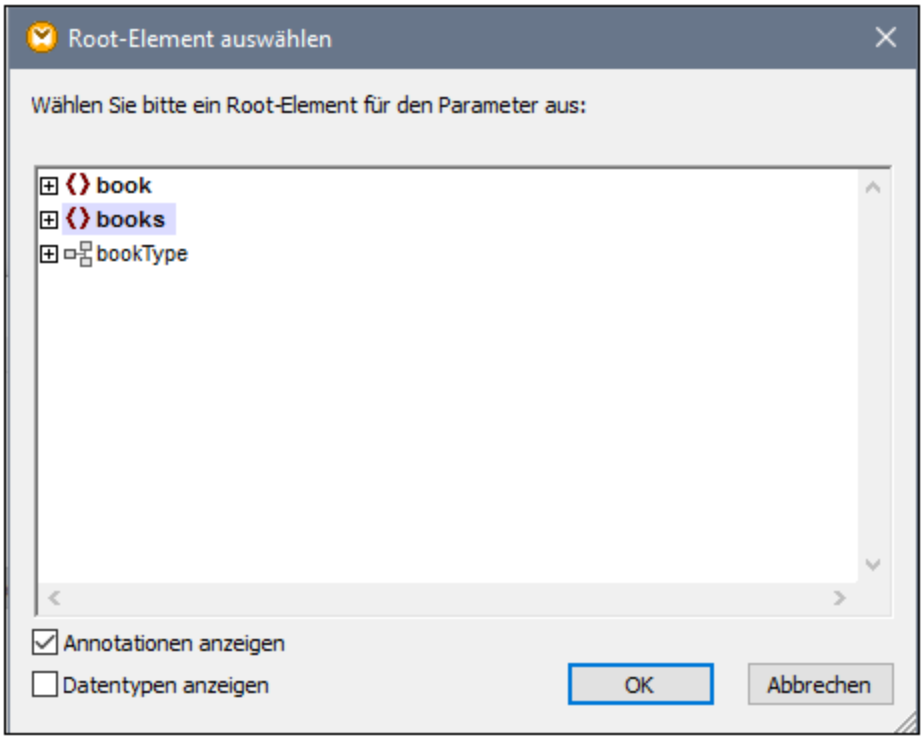
Bei Auswahl von **Complex Type** müssen einige zusätzliche Schritte durchgeführt werden:

3. Klicken Sie auf **Auswählen**, um die Quelldatei für die Struktur der Variablen[882] auszuwählen. Die Strukturen in der Abbildung unten beziehen sich nur auf die MapForce Enterprise Edition. Eine Liste der für andere MapForce Editionen relevanten Strukturen finden Sie im vorhergehenden Kapitel[882].



4. Definieren Sie das Root-Datenelement für die Struktur der Variablen, wenn Sie danach gefragt werden. So können Sie z.B. in XML-Schemas jedes beliebige Element oder jeden beliebigen Typ aus der ausgewählten Quelldatei auswählen (*siehe Abbildung unten*).

© 2016-2022 Altova GmbH

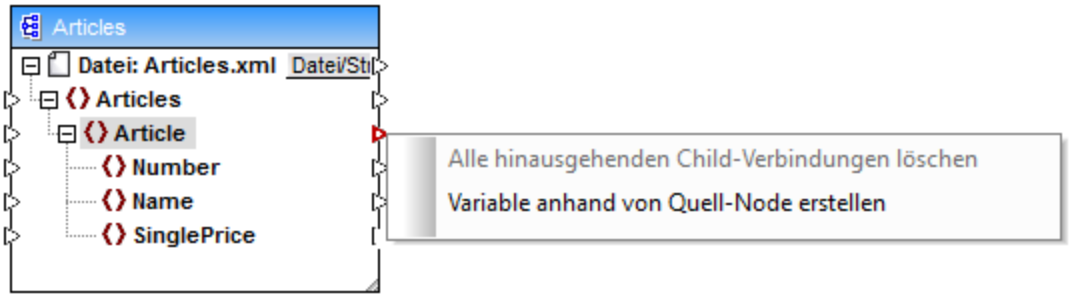


Option 2: über das Kontextmenü

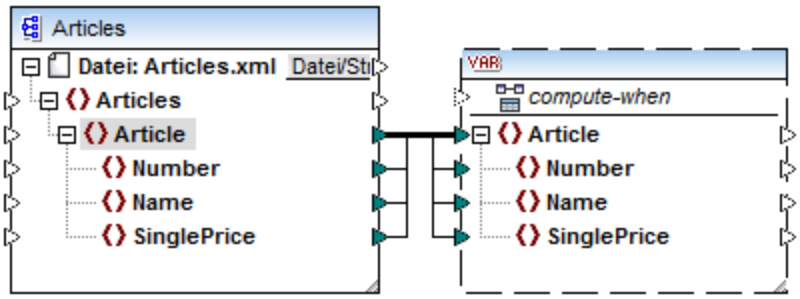
Bei der zweiten Option können Sie eine Variable über das Kontextmenü erstellen. Unten finden Sie eine Liste der möglichen Optionen.

Variable anhand eines Quell-Node

Um anhand eines Quell-Node eine Variable zu erstellen, klicken Sie mit der rechten Maustaste auf den Output-Konnektor einer Komponente (in diesem Beispiel den Output-Konnektor des Elements `<Article>`) und wählen Sie den Befehl **Variable anhand von Quell-Node erstellen** (*siehe Abbildung unten*).



Daraufhin wird eine komplexe Variable mit dem Quellschema der Komponente `Articles` erstellt. Alle Datenelemente werden automatisch mittels einer "Alles kopieren"-Verbindung[97] verbunden (*siehe Abbildung unten*).

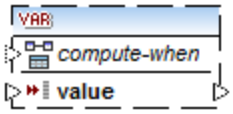


Variable anhand eines Ziel-Node
Um eine Variable anhand eines Ziel-Node zu erstellen, klicken Sie mit der rechten Maustaste auf den Input-Konnektor einer Zielkomponente und wählen Sie **Variable für Ziel-Node erstellen**. Daraufhin wird eine komplexe Variable mit demselben Schema, das auch in der Zielkomponente verwendet wird, erstellt. Alle Datenelemente werden automatisch mittels einer "Alles kopieren"-Verbindung verbunden.

Variable anhand eines Filters:
Um eine Variable mit Hilfe eines Filters zu erstellen, klicken Sie mit der rechten Maustaste auf den Output-Konnektor einer Filterkomponente (`on-true/on-false`) und wählen Sie den Befehl **Variable anhand von Quell-Node erstellen**. Daraufhin wird anhand des Quellschemas eine komplexe Komponente erstellt, wobei das mit dem Filter-Input verbundene Datenelement automatisch als Root-Element der Zwischenkomponente verwendet wird.

5.3.2 Geltungsbereich und Kontext von Variablen

Jede Variable hat ein `compute-when` Input-Datenelement (*siehe Abbildung unten*), mit dem Sie den Geltungsbereich der Variablen festlegen können, d.h. Sie können damit festlegen, wann und wie oft der Wert der Variablen bei der Ausführung des Mappings berechnet werden soll. Sie müssen diesen Input in vielen Fällen nicht verbinden, doch ist er manchmal wichtig, um den Standardkontext außer Kraft zu setzen oder die Mapping-Leistung zu optimieren.



In der Erläuterung von Geltungsbereich und Kontext von Variablen werden die folgenden Begriffe verwendet: *Substruktur* und *Variablenwert*. Als Substruktur wird eine Gruppe eines Datenelements/Node in einer Zielkomponente und aller seiner Nachfahren bezeichnet, z.B. ein Element `<Person>` mit seinen Child-Elementen `<FirstName>` und `<LastName>`.

Als Variablenwert werden die auf der Output-Seite der Variablenkomponente verfügbaren Daten bezeichnet.

- Bei einfachen Variablen handelt es sich um eine Sequenz atomarer Werte, die den in den Komponenteneigenschaften definierten Datentyp haben.

 © 2016-2022 Altova GmbH

- Bei komplexen Variablen handelt es sich um eine Sequenz von Root Nodes (des in den Komponenteneigenschaften definierten Typs), einschließlich aller Nachfahren-Nodes der einzelnen Root Nodes.

Die Sequenz der atomaren Werte (oder Nodes) kann ein oder auch null Elemente enthalten. Dies ist abhängig davon, was mit der Input-Seite der Variablen und etwaigen übergeordneten Datenelementen in der Quell- und Zielkomponente verbunden ist.

"Compute-when" ist nicht verbunden (Standardeinstellung)

Wenn das `compute-when`-Input-Datenelement nicht (mit einem Output Node einer Quellkomponente) verbunden ist, wird der Variablenwert berechnet, *sobald er das erste Mal in einer Zielsubstruktur* (direkt über einen Konnektor von der Variablenkomponente zu einem Node in der Zielkomponente oder indirekt über Funktionen) verwendet wird. Derselbe Variablenwert wird auch für alle Child-Ziel-Nodes innerhalb der Substruktur verwendet.

Der tatsächliche Wert der Variablen hängt von etwaigen Verbindungen zwischen übergeordneten Datenelementen der Quell- und Zielkomponente ab. Dieses Standardverhalten ist dasselbe wie das von "complex" Outputs von regulären benutzerdefinierten Funktionen[970] und Webservice-Funktionsaufrufen. Wenn die Variable mit mehreren nicht miteinander in Zusammenhang stehenden Ziel-Nodes verbunden ist, wird der Wert der Variablen *für jeden davon separat berechnet*. Dabei können in jedem Fall unterschiedliche Ergebnisse erzeugt werden, da unterschiedliche Parent-Verbindungen sich auf den Kontext auswirken, in dem der Wert der Variablen ausgewertet wird.

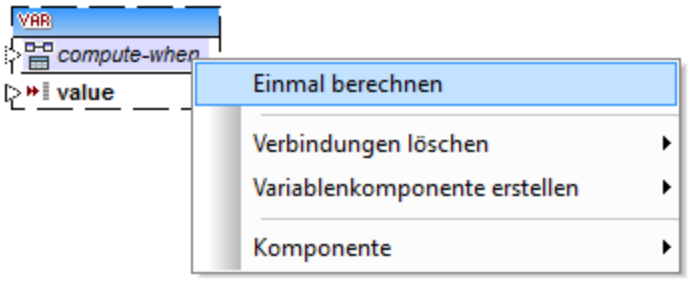
"Compute-when" ist verbunden

Wenn ein Output Node einer Quellkomponente mit `compute-when` verbunden ist, wird die Variable immer dann berechnet, *wenn das Quelldatenelement in der Zielsubstruktur zum ersten Mal verwendet wird*.

Die Variable verhält sich so, als wäre sie ein Child-Element des mit `compute-when` verbundenen Datenelements. Auf diese Art kann die Variable an ein bestimmtes Quelldatenelement gebunden werden, d.h. die Variable wird zur Laufzeit immer dann neu ausgewertet, wenn ein neues Datenelement aus der Sequenz in der Quellkomponente gelesen wird. Dies steht im Zusammenhang mit der allgemeinen Regel zu Verbindungen in MapForce: Erstelle für jedes Quelldatenelement ein Zieldatenelement. In diesem Fall berechnet MapForce aufgrund von `compute-when` den Variablenwert für jedes Quelldatenelement. Nähere Informationen dazu finden Sie unter Mapping-Regeln und Strategien[1330].

Compute-once

Falls nötig, können Sie festlegen, dass der Variablenwert *einmal vor jedem Wert der Zielkomponenten berechnet wird*, sodass die Variable praktisch zu einer globalen Konstante für das restliche Mapping wird. Klicken Sie dazu mit der rechten Maustaste auf den Eintrag `compute-when` und wählen Sie im Kontextmenü den Befehl **Einmal berechnen**:

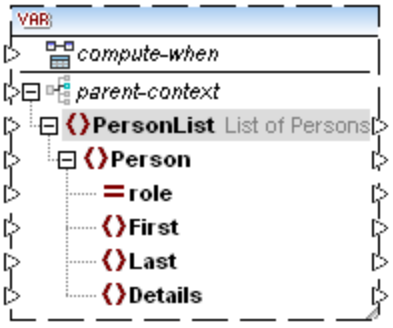


Wenn Sie den Geltungsbereich der Variablen in `compute-when=once` ändern, wird der Input-Konnektor aus dem `compute-when`-Datenelement entfernt, da eine solche Variable nur einmal ausgewertet wird. In einer benutzerdefinierten Funktion wird die Variable `compute-when=once` bei jedem Funktionsaufruf ausgewertet, bevor das tatsächliche Funktionsergebnis ausgewertet wird.

Parent-context

Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.

Ein `parent-context`-Datenelement muss z.B. dann hinzugefügt werden, wenn in Ihrem Mapping mehrere Filter verwendet werden, und Sie einen zusätzlichen Parent-Node benötigen, über den iteriert werden kann. Nähere Informationen dazu finden Sie unter Beispiel: Ändern des Parent-Kontexts 1340. Um einen parent-context zu einer Variablen hinzuzufügen, klicken Sie mit der rechten Maustaste auf den Root Node, (in diesem Beispiel `PersonList`) und wählen Sie im Kontextmenü den Befehl **parent-context hinzufügen**. Daraufhin wird ein neuer Node `parent-context` zur bestehenden Hierarchie hinzugefügt.



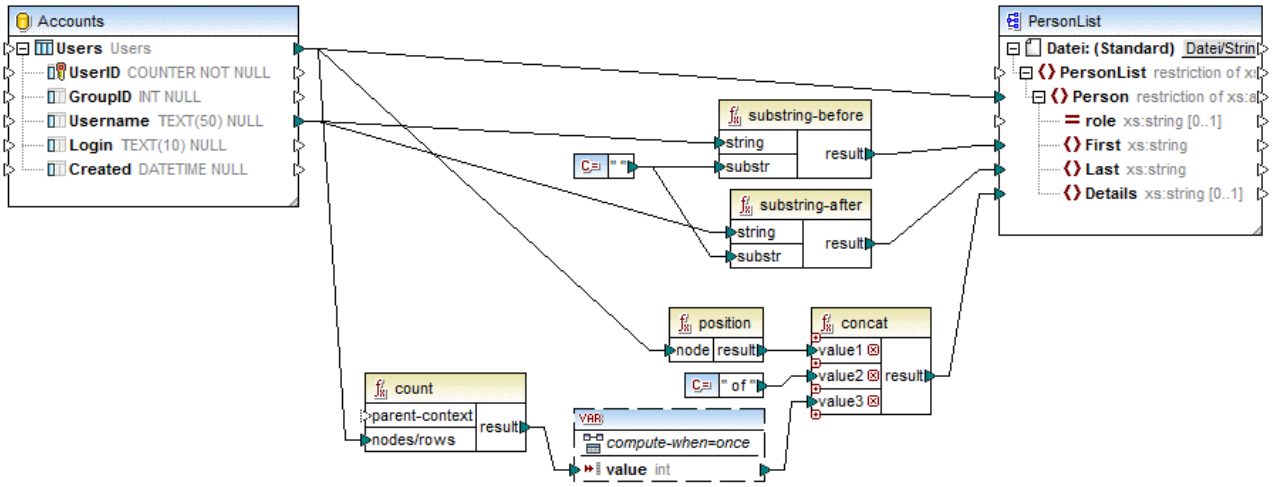
Durch den parent-context wird innerhalb der Komponente ein virtueller übergeordneter (Parent) Node zur Hierarchie hinzugefügt. Auf diese Art können Sie über einen zusätzlichen Node in derselben oder einer anderen Quellkomponente iterieren.

 © 2016-2022 Altova GmbH

5.3.3 Beispiel: Zählen von Datenbanktabellenzeilen

Das in diesem Beispiel beschriebene Mapping finden Sie unter dem Namen **DB_UserList.mfd** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. In diesem Mapping werden Benutzerdatensätze aus einer Datenbanktabelle namens "Users" extrahiert und in eine XML-Datei geschrieben. Die Datenbankspalte "Username" enthält den Vor- als Zunamen einer Person (z.B. "Vernon Callaby"). Dieses Mapping hat folgende Ziele:

1. Für jeden Datensatz in der Tabelle "Users" soll ein neues `Person`-Element in der XML-Datei erstellt werden.
2. Der aus dem Datenbankfeld "Username" extrahierte Wert soll in der XML-Datei auf zwei separate Felder aufgeteilt werden ("First" und "Last").
3. Für jeden Datensatz soll die fortlaufende Nummer aus der Gesamtanzahl aller Datensätze in der Datenbank extrahiert werden (z.B. "Datensatz 1 von 4") und in das Datenelement `Details` geschrieben werden.



DB_UserList.mfd

Wie oben gezeigt, wird eine Verbindung zwischen der Quelltabelle "Users" und dem Element `Person` der XML-Zieldatei gezogen. Auf diese Art wird für jeden Datensatz in der Quelltabelle ein neues `Person`-Element in der Zielkomponente erstellt (erster Punkt der Mapping-Ziele).

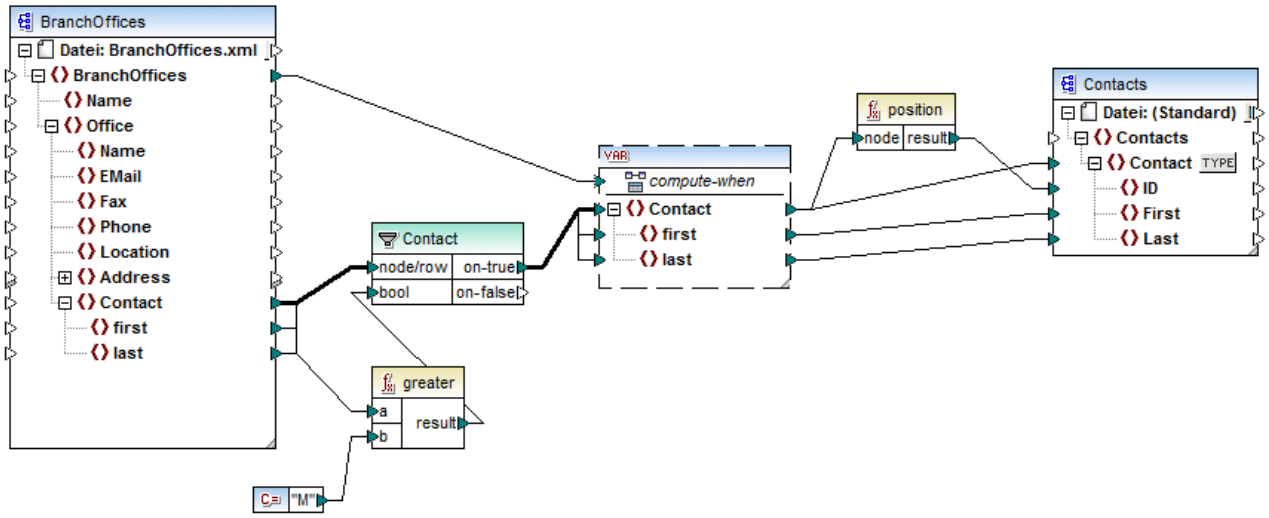
Der Wert des Felds "Username" wird durch die Funktionen **`substring-before`**[1111] und **`substring-after`**[1110] bereitgestellt. Diese beiden Funktionen extrahieren den Text vor und nach dem Leerzeichen (" "). Damit wird der zweite Punkt der Mapping-Ziele erfüllt.

Zur Erreichung des dritten Ziels wird die **`count`**-Funktion verwendet. Das Ergebnis der count-Funktion wird an eine Variable übergeben. Mit der Variablen wird sichergestellt, dass das Ergebnis im Mapping gespeichert wird und zur Verfügung steht, wenn die "Details"-Elemente der einzelnen Personen in die XML-Zieldatei geschrieben werden. Beachten Sie, dass Datenbankdatensätze aus Gründen der Effizienz nur einmal gezählt werden sollten, daher ist der Geltungsbereich der Variablen auf `compute-when=once` gesetzt (siehe Ändern von Kontext und Geltungsbereich von Variablen[888])

5.3.4 Beispiel: Filtern und Nummerieren von Nodes

Das in diesem Beispiel beschriebene Mapping finden Sie unter dem Namen **PositionInFilteredSequence.mfd** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**.

Dieses Mapping liest eine XML-Datei mit Kontaktdaten mehrerer Personen, filtert die Daten und schreibt sie in eine XML-Zieldatei. Ziel des Mappings ist es, nur die Personen aus der XML-Quelldatei zu filtern, deren Nachname mit dem Buchstaben "M" oder einem nachfolgenden Buchstaben beginnt. Außerdem müssen die extrahierten Kontakte nummeriert werden. Die Nummerierung dient als eindeutiger Identifier für die einzelnen Kontakte in der XML-Zieldatei.



PositionInFilteredSequence.mfd

Um das obige Ziel zu erreichen, wurden die folgenden Komponententypen zum Mapping hinzugefügt:

- ein Filter (siehe Filter und Bedingungen 914)
- eine komplexe Variable (siehe Hinzufügen von Variablen 884)
- die Funktionen **greater** 1059 und **position** 1096 (siehe Hinzufügen einer Funktion zum Mapping 943)
- eine Konstante (Um eine Konstante hinzuzufügen, wählen Sie den Menübefehl **Einfügen | Konstante**).

Für die Variable wird dasselbe Schema wie für die Quellkomponente verwendet. Wenn Sie mit der rechten Maustaste auf die Variable klicken und im Kontextmenü den Befehl **Eigenschaften** wählen, sehen Sie, dass der Node **BranchOffices/Office/Contact** als Root-Node für diese Variablenstruktur ausgewählt ist.

Zuerst werden die Daten der Quellkomponente an den Filter übergeben. Der Filter übergibt nur die Datensätze an die Variable, die die Filterbedingungen erfüllen. In diesem Fall wurde der Filter so konfiguriert, dass nur die `Contact` Nodes abgerufen werden, deren Nachname gleich oder größer "M" ist. Zu diesem Zweck vergleicht die Funktion **greater** 1059 jedes `last`-Datenelement mit dem Konstantenwert "M".

In der Variable wurde der `compute-when` Input mit dem Root-Datenelement der Quellkomponente (`BranchOffices`) verbunden. Dadurch wird die Variable zur Laufzeit jedes Mal, wenn ein neues Datenelement aus der Sequenz in der Quellkomponente gelesen wird, erneut ausgewertet. In diesem Mapping macht es jedoch keinen Unterschied, ob das `compute-when`-Datenelement verbunden ist oder nicht, da die Variable

© 2016-2022 Altova GmbH

(indirekt über den Filter) mit dem Quelldatenelement `Contact` verbunden ist und so oft, wie Instanzen von `Contact`, die die Filterbedingungen erfüllen, vorhanden sind, berechnet wird.

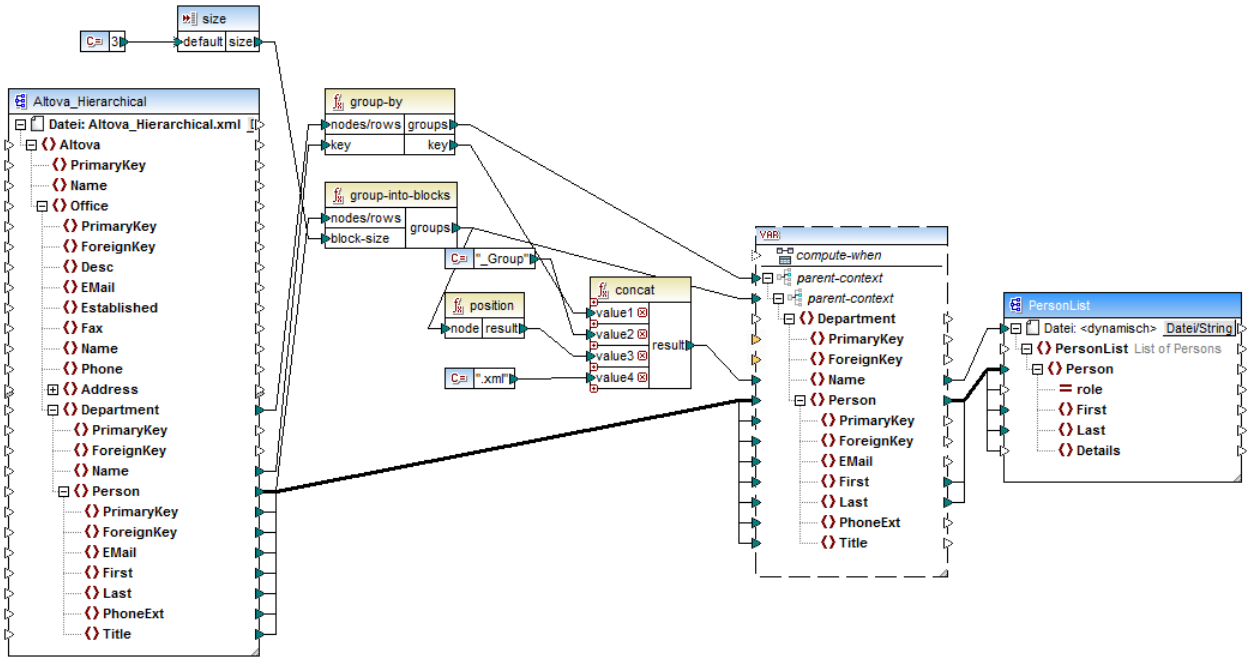
Die Funktion **`position`**[1096] gibt für jede Iteration der Variablen die Nummer der aktuellen Sequenz zurück. Nur acht Kontakte erfüllen die Filterbedingungen; wenn Sie eine Vorschau des Mappings anzeigen und sich die Ausgabe ansehen, sehen Sie daher, dass die IDs 1 bis 8 in das `ID`-Element der Zielkomponente geschrieben wurden.

Falls Sie sich fragen, wozu die Variable überhaupt benötigt wurde: Der Grund dafür ist, dass alle Datensätze nummeriert werden müssen. Hätten wir das Filterergebnis direkt mit der Zielkomponente verbunden, hätte es keine Möglichkeit gegeben, die einzelnen Instanzen von `Contact` durchzunummerieren. Die Aufgabe der Variablen in diesem Mapping ist somit, die einzelnen Instanzen von `Contact` temporär im Mapping zu speichern, damit diese nummeriert werden können, bevor sie in die Zielkomponente geschrieben werden.

5.3.5 Beispiel: Unterteilen von Datensätzen in Gruppen und Untergruppen

Das in diesem Beispiel beschriebene Mapping finden Sie unter dem Namen **DividePersonsByDepartmentIntoGroups.mfd** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**.

Das Mapping verarbeitet eine XML-Datei, die Mitarbeiterdatensätze einer fiktiven Firma enthält. Die Firma hat zwei Niederlassungen: "Nanonull, Inc." und "Nanonull Partners, Inc". Jede Niederlassung hat mehrere Abteilungen (departments) (z.B. "IT", "Marketing" usw.) und jede Abteilung hat einen oder mehrere Mitarbeiter. Ziel des Mappings ist es, unabhängig von der Niederlassung, aus jeder Abteilung Gruppen von maximal drei Mitarbeitern zu erstellen. Die Größe jeder Gruppe beträgt standardmäßig drei; dies sollte man jedoch bei Bedarf leicht ändern können. Jede Gruppe muss als separate XML-Datei mit dem Namen im Format "<Abteilungsname>_GruppeN" (z.B. **Marketing_Group1.xml**, **Marketing_Group2.xml**, usw.) gespeichert werden.



DividePersonsByDepartmentIntoGroups.mfd

Wie in der Abbildung oben gezeigt, wurden eine komplexe Variable sowie einige andere Komponententypen (hauptsächlich Funktionen) zum Mapping hinzugefügt. Die Variable hat dieselbe Struktur wie ein `Department`-Datenelement in der XML-Quelldatei. Wenn Sie mit der rechten Maustaste auf die Variable klicken, um ihre Eigenschaften anzuzeigen, sehen Sie, dass dafür dasselbe XML-Schema wie für die Quellkomponente verwendet wird und dass ihr Root Element `Department` ist. Wichtig ist vor allem, dass die Variable zwei ineinander verschachtelte `parent-context`-Datenelemente hat, mit denen sichergestellt wird, dass die Variable zuerst im Kontext jeder einzelnen Abteilung und anschließend im Kontext der einzelnen Gruppen in diesen Abteilungen berechnet wird (siehe auch Ändern von Kontext und Geltungsbereich von Variablen 888).

Das Mapping iteriert zuerst durch alle Abteilungen, um die Namen der einzelnen Abteilungen zu erhalten (diese Namen werden anschließend benötigt, um die Dateinamen für die einzelnen Gruppen zu erstellen). Zu diesem Zweck wird die **`group-by`** 1085-Funktion mit dem Quelldatenelement `Department` verbunden und der Abteilungsname wird als Gruppierungsschlüssel bereitgestellt.

Als nächstes findet innerhalb des Kontexts der einzelnen Abteilungen eine zweite Gruppierung statt. Dabei ruft das Mapping die Funktion **`group-into-blocks`** 1089 auf, um die gewünschten Mitarbeitergruppen zu erstellen. Die Größe der Gruppen wird durch eine einfache Input-Komponente mit dem Standardwert "3" angegeben. Der Standardwert stammt aus einer Konstante. Um die Gruppengröße in diesem Beispiel zu ändern, muss man nur die Konstantenwert nach Bedarf anpassen. Sie können allerdings auch die "size" Input-Komponente ändern, sodass die Größe jeder Gruppe dem Mapping einfach als Parameter bereitgestellt wird, wenn das Mapping durch generierten Code oder mit MapForce Server ausgeführt wird. Nähere Informationen dazu finden Sie unter Bereitstellen von Parametern für das Mapping 867.

Als nächstes wird der Wert der Variablen an die XML-Zielkomponente PersonList geliefert. Die Dateinamen der einzelnen erstellten Gruppen wurden durch Verkettung der folgenden Teile mit Hilfe der **`concat`** 1105-Funktion berechnet:

 © 2016-2022 Altova GmbH

1. Der Name der jeweiligen Abteilung
2. Der String "_Group"
3. Die Nummer der Gruppe in der aktuellen Sequenz (z.B. "1", wenn es sich um die erste Gruppe in dieser Abteilung handelt)
4. Der String ".xml"

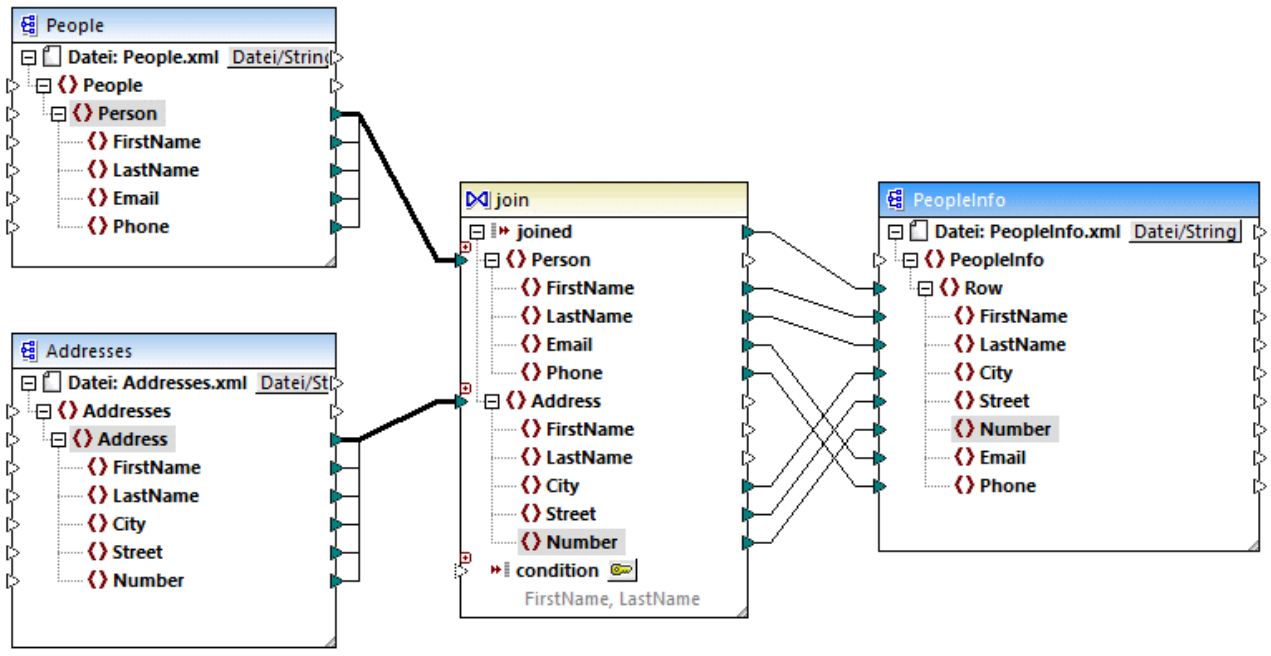
Das Ergebnis dieser Verkettung wird im Datenelement `Name` der Variablen gespeichert und anschließend als dynamischer Dateiname an die Zielkomponente weitergeleitet. Dadurch wird für jeden erhaltenen Wert ein neuer Dateiname erstellt. In diesem Beispiel werden anhand der Variablen insgesamt acht Gruppen erstellt, d.h. es werden bei Ausführung des Mappings, wie gewünscht, acht Ausgabedateien erstellt. Nähere Informationen zu diesem Methode finden Sie unter Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 1371.

5.4 Verknüpfen von Komponenten mittels Join

Manchmal müssen Daten aus zwei oder mehr Strukturen auf Basis einiger Bedingungen (z.B. wenn Feld A in der ersten Struktur denselben Wert wie Feld B in der zweiten Struktur hat) miteinander kombiniert werden. Zu diesem Zweck eignet sich die Verwendung einer Join-Komponente.

Eine Join-Komponente ist eine MapForce-Komponente, mit der zwei oder mehr Strukturen im Mapping auf Basis benutzerdefinierter Bedingungen miteinander verknüpft werden können. Sie gibt die miteinander verknüpften Datenelemente (die verknüpfte Gruppe) zurück, die die Bedingung erfüllen. Join-Verknüpfungen eignen sich besonders, um Daten aus zwei Datenstrukturen mit einem gemeinsamen Feld (z.B. einer ID) miteinander zu kombinieren.

Im unten gezeigten Mapping ist die mittlere Komponente z.B. eine "Join"-Komponente. In diesem Mapping werden zwei XML-Strukturen (eine Personen- und eine Adressliste) miteinander verknüpft. Ziel ist es, alle Informationen zu jeder Person in eine XML-Zieldatei zu übertragen. Die Felder `FirstName` und `LastName` werden als Verknüpfungsschlüssel verwendet. Wenn der Wert von `FirstName` und `LastName` (unter `Person`) mit dem von `FirstName` und `LastName` (unter `Address`) identisch ist, gehören die Adressdaten zu ein und derselben Person und werden verknüpft. Alle Datenelemente aus der miteinander verknüpften Struktur können anschließend auf eine Zielkomponente (in diesem Fall eine XML-Datei) gemappt werden. Die Join-Bedingung selbst wird durch Klicken auf die Schaltfläche **Join-Bedingung definieren** ( ) in den Eigenschaften der Join-Komponente definiert. Zu dieser Komponenten gibt es ein Mapping-Beispiel, das unter Beispiel: Verknüpfen von XML-Strukturen 902 ausführlich erläutert wird.



JoinPeopleInfo.mfd

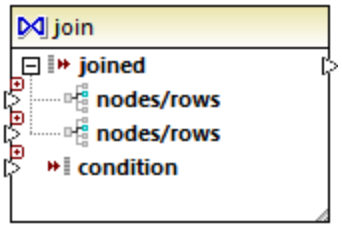
Wie oben gezeigt, werden die Quellstrukturen und die Join-Komponente mittels einer "Alles kopieren"-Verbindung verbunden, wodurch die Verbindungen übersichtlicher angezeigt werden. Im Allgemeinen werden solche Verbindungen automatisch von MapForce erstellt, wenn der Kontext relevant ist (Nähere Informationen dazu siehe "Alles kopieren"-Verbindungen 97).

© 2016-2022 Altova GmbH

Die zu verknüpfenden Strukturen können entweder (wie im obigen Mapping) aus separaten Komponenten oder aus derselben Komponente stammen. Es kann sich dabei auch um unterschiedliche Arten handeln (z.B. eine XML-Struktur und eine Datenbanktabelle). Nähere Informationen zu Join-Verknüpfungen im Zusammenhang mit Datenbanken finden Sie unter Verknüpfen von Datenbankdaten mittels "Join" 344.

So fügen Sie eine Join-Komponente hinzu:

1. Definieren Sie BUILT-IN als Mapping-Transformationssprache (Klicken Sie dazu entweder auf die Symbolleisten-Schaltfläche BUILT IN oder verwenden Sie den Menübefehl **Ausgabe | Built-In Ausführungsprozessor**).
2. Wählen Sie im Menü **Einfügen** den Befehl **Verbinden**. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche **Verbinden** ( ). Daraufhin wird die Join-Komponente im Mapping angezeigt. Standardmäßig akzeptiert sie Daten aus zwei Strukturen, daher hat sie zwei `nodes/rows`-Inputs. Falls nötig, können Sie neue Inputs durch Klicken auf die Schaltfläche **Input hinzufügen** ( ) hinzufügen, siehe Verknüpfen von drei oder mehr Strukturen 901.



3. Verbinden Sie die gewünschten Strukturen mit den `nodes/rows`-Datenelementen der Join-Komponente.
4. Fügen Sie eine (oder mehrere) Bedingungen für die Verknüpfung hinzu. Klicken Sie dazu mit der rechten Maustaste auf die Join-Komponente und wählen Sie **Eigenschaften**. Join-Bedingungen können auch direkt über das Mappen hinzugefügt werden, indem Sie das Boolesche Ergebnis einer Funktion mit dem `condition`-Datenelement der Join-Komponente verbinden. In bestimmten Fällen, beim Verknüpfen von Datenbanktabellen mittels Join, kann die Join-Bedingung (oder die Join-Bedingungen) von MapForce automatisch erstellt werden. Nähere Informationen dazu finden Sie unter Hinzufügen von Join-Bedingungen 898.

Anmerkungen:

- Join-Verbindungen werden unterstützt, wenn als Zielsprache des Mappings BUILT-IN definiert ist. Die Codegenerierung in C#, C++ oder Java wird nicht unterstützt.
- Wenn eine Struktur keine gültige oder unterstützte Input-Quelle für die Join-Verknüpfung ist, wird entweder direkt im Mapping oder im Fenster "Meldungen" eine entsprechende Information angezeigt, wenn Sie das Mapping validieren (siehe Validieren von Mappings 107).
- Join-Komponenten sollten nicht mit Input-Parametern oder Ergebnissen von inline gesetzten benutzerdefinierten Funktionen verbunden werden. Falls solche Verbindungen vorhanden sind, kommt es bei der Mapping-Validierung zu Validierungsfehlern.
- Wenn Sie verknüpfbare Datenbankkomponenten (wie z.B. Tabellen oder Ansichten) direkt mit einer Join-Komponente verknüpfen, wird in der rechten oberen Ecke der Join-Komponente automatisch eine **SQL-Modus** (SQL) Schaltfläche angezeigt. Wenn diese Schaltfläche aktiviert ist, stehen spezielle SQL-Funktionen für die Join-Operation zur Verfügung (siehe Join-Verknüpfungen im SQL-Modus 345).
- Die Ausgabe des mit `Join` verknüpften Datenelements kann mit einer anderen Join-Komponente verbunden werden. Gegebenenfalls können Sie jedoch ein Teilergebnis der einen Join-Verknüpfung mit

einer anderen verbinden.

Join-Komponenten im Vergleich zu anderen Komponententypen

In einigen Fällen können dieselben Ergebnisse auch mit Hilfe komplexer Variablen oder Filter anstelle von Join-Komponenten erzielt werden (siehe Verwendung von Variablen 882 bzw. Filter und Bedingungen 914). Im Gegensatz zu anderen Komponententypen machen Join-Komponenten das Mapping jedoch übersichtlicher, da Sie auf einen Blick sehen, welche Daten miteinander verknüpft wurden. Außerdem wird die Mapping Performanz erheblich verbessert, wenn der SQL-Modus der Join-Komponente aktiviert ist (Dies gilt für Datenbankverbindungen, siehe Verknüpfen von Datenbanktabellen mittels Join 344).

Hinzufügen eines Parent-Kontexts

In speziellen Fällen können Sie zur Erzielung eines bestimmten Mapping-Ergebnisses explizit einen Mapping-Kontext (einen so genannten "Parent-Kontext") für mit der Join-Komponente verbundene Daten bereitstellen. Um einen Parent-Kontext hinzuzufügen, klicken Sie mit der rechten Maustaste auf das `verknüpfte` Datenelement der Join-Komponente und wählen Sie im Kontextmenü den Befehl **Parent-Kontext hinzufügen**. Daraufhin wird in der Join-Komponente ein zusätzlicher `parent-context`-Input, mit dem Sie die erforderliche Datenquelle verbinden können, angezeigt. Nähere Informationen dazu finden Sie unter Beispiel: Ändern des Parent-Kontexts 1340.

Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.

5.4.1 Hinzufügen von Join-Bedingungen

Bei einer Join-Verknüpfung müssen Datenelemente aus zwei oder mehr Strukturen auf Basis einer Bedingung miteinander kombiniert werden, daher ist für eine Join-Verknüpfung immer mindestens eine Bedingung erforderlich. Es gibt verschiedene Möglichkeiten, um Join-Bedingungen hinzuzufügen. Siehe unten.

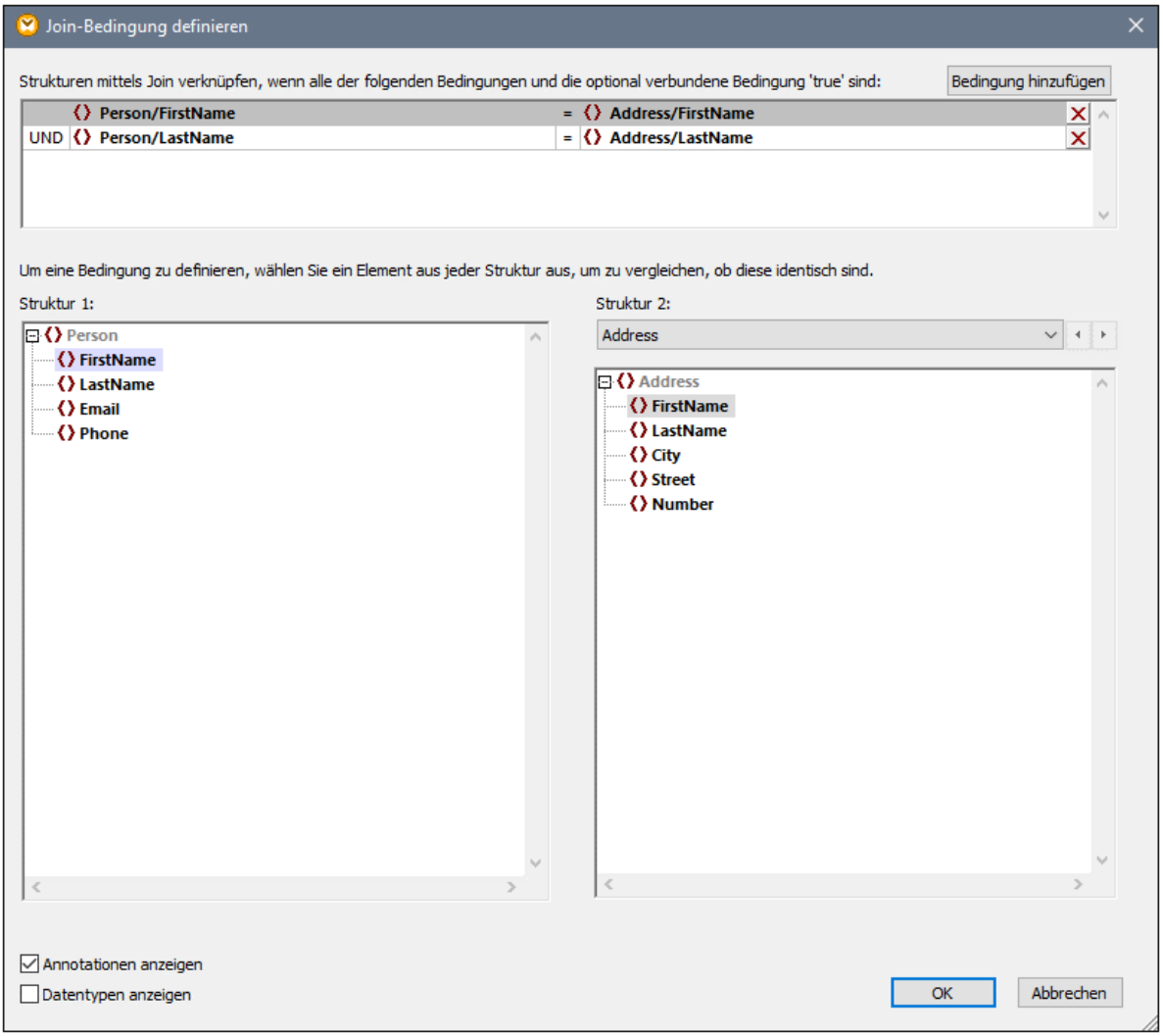
Anmerkung: Wenn Datenbanktabellen im SQL-Modus mittels Join verknüpft werden, erstellt MapForce die Join-Bedingung (oder Bedingungen) automatisch auf Basis von zwischen Tabellen gefundenen Sekundärschlüsselbeziehungen. Damit dies möglich ist, müssen sich die Datenbanktabellen in der MapForce-Komponente in einer Child-Parent-Beziehung befinden (d.h. eine Tabelle muss "Parent" oder "Child" einer anderen in der Komponente sein), siehe Beispiel: Verknüpfen von Tabellen mittels Join im SQL-Modus 348.

Methode 1: Hinzufügen einer Join-Bedingung über die Komponenteneigenschaften

1. Stellen Sie im Mapping sicher, dass mindestens zwei Strukturen (oder Datenbanktabellen) mit der Join-Komponente verbunden sind. Die in diesem Beispiel gezeigte Join-Komponente ist Teil des **JoinPeopleInfo.mfd** Mappings aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Dieses Mapping wird im Beispiel: Verknüpfen von XML-Strukturen 902 näher erläutert.
2. Klicken Sie in der Join-Komponente auf die Schaltfläche **Join-Bedingung definieren** () (oder klicken Sie mit der rechten Maustaste auf die Überschrift der Komponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften**).

 © 2016-2022 Altova GmbH

3. Wählen Sie ein Datenelement aus der linken Struktur sowie ein weiteres aus der rechten Struktur aus (Immer wenn das Ergebnis des Vergleichs dieses Paars "true" ergibt, werden die linke und die rechte Struktur miteinander verknüpft).



Um mehrere Bedingungen hinzuzufügen, klicken Sie auf **Bedingung hinzufügen** und wählen Sie ein neues Datenelementpaar aus. So wurden etwa in der Abbildung oben zwei Join-Bedingungen definiert:

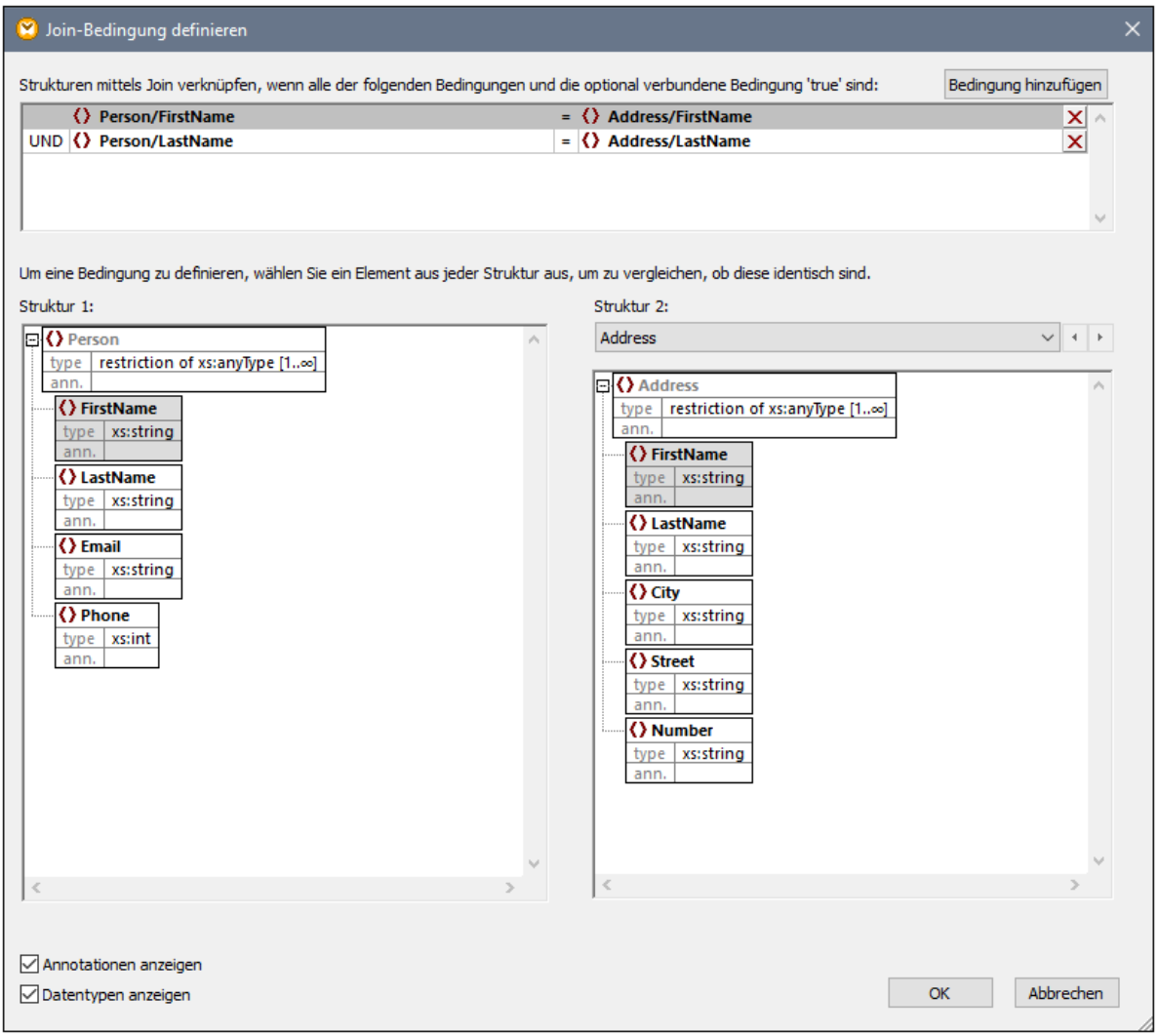
1. `FirstName` in der Struktur 1 muss gleich `FirstName` in Struktur 2 sein und
2. `LastName` in Struktur 1 muss gleich `LastName` in Struktur 2 sein.

Um eine Bedingung zu entfernen, klicken Sie auf die Schaltfläche **Löschen** neben der jeweiligen Bedingung.

Anmerkungen:

- Wenn mehrere Join-Bedingungen vorhanden sind, müssen alle davon erfüllt werden, damit die beiden Strukturen miteinander verknüpft werden. Anders ausgedrückt, werden mehrere Bedingungen mittels einer logischen UND-Operation miteinander verknüpft. Darin eingeschlossen sind auch optionale über das Mapping hinzugefügte Bedingungen (siehe Methode 2 weiter unten).

- Wenn mehr als zwei Strukturen mit der Join-Komponente verbunden sind, werden diese optionalen Strukturen in der Dropdown-Liste unterhalb von "Struktur 2" angezeigt. Wenn Sie eine solche zusätzliche Struktur aus der Dropdown-Liste auswählen, werden im linken Bereich alle Strukturen, die *vor* der Join-Komponente vorkommen, angezeigt. Auf diese Art können Sie zwischen jeder der vielen Strukturen Join-Bedingungen definieren. Ein Beispiel dazu finden Sie unter Beispiel: Erstellen eines CSV-Berichts anhand mehrerer Tabellen 357.
- Um den Datentyp von Datenelementen in den einzelnen Strukturen anzuzeigen, aktivieren Sie das Kontrollkästchen **Datentypen anzeigen**. Mit der Option **Annotationen anzeigen** werden zusätzliche Informationen zu Datenelementen angezeigt, vorausgesetzt solche Informationen sind im zugrunde liegenden Schema (bzw. der Datenbank) vorhanden. Wenn beide Kontrollkästchen aktiviert sind, ändert sich das Layout und es werden sowohl Annotationen als auch Datentypen angezeigt, z.B.:

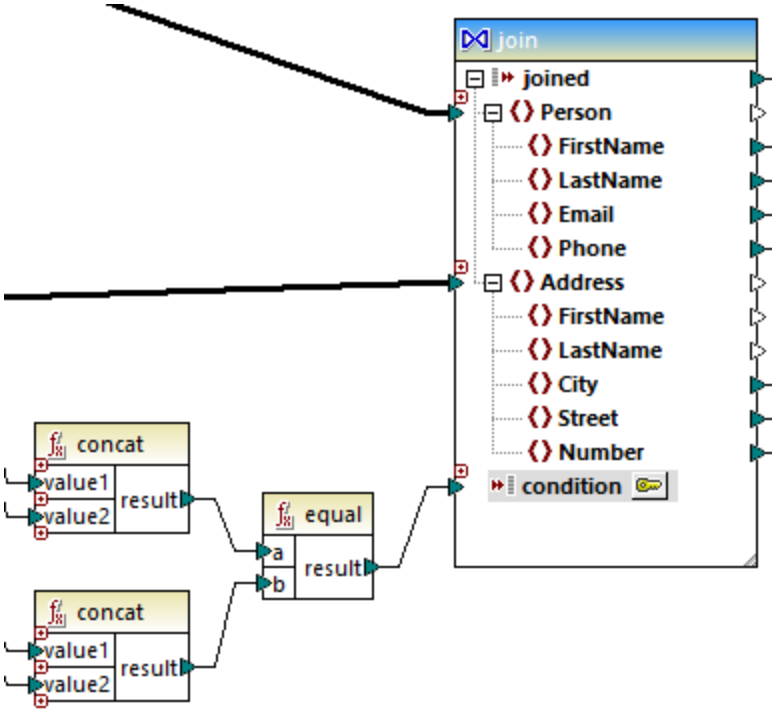


Methode 2: Hinzufügen einer Join-Bedingung über das Mapping

- Fügen Sie im Mapping Komponenten hinzu, deren Ergebnis ein Boolescher Wert ist und verbinden Sie dann den Booleschen Output mit dem Input des Datenelements `condition`. So kann z.B. mit Hilfe der **`equal`**-Funktion ein Wert mit einem Datenelement aus dem Mapping verglichen werden und das

© 2016-2022 Altova GmbH

Boolesche Ergebnis kann als Input an das `condition`-Datenelement der Join-Komponente übergeben werden.



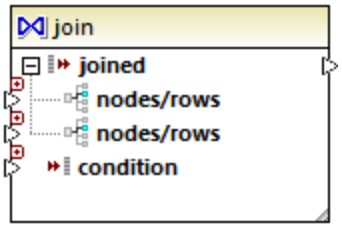
Anmerkung: Wenn über die Eigenschaften der Join-Komponente keine Bedingung definiert ist (Methode 1), muss das Datenelement `condition` der Join-Komponente verbunden werden (Methode 2).

Methode 3: Gemischte Methode

Sie können im selben Mapping einige Join-Bedingungen in den Komponenteneigenschaften (Methode 1) definieren und diese mit einer aus dem Mapping (Methode 2) kombinieren. Wenn Sie jedoch beabsichtigen, Datenbanktabellen im SQL-Modus zu verknüpfen, müssen die Bedingungen unbedingt mit Methode 1 definiert werden (siehe Join-Verknüpfungen im SQL-Modus 345).

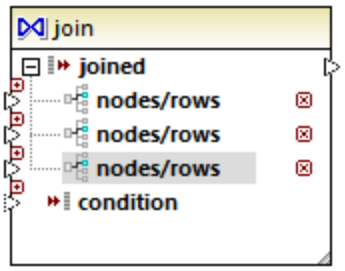
5.4.2 Verknüpfen von drei oder mehr Strukturen

Wenn Sie eine Join-Komponente über den Menübefehl **Einfügen | Join** zum Mapping hinzufügen, so bietet diese standardmäßig Platz für zwei Strukturen (d.h. die Komponente enthält nur zwei `nodes/rows`-Inputs).



Wenn Sie mehr als zwei Strukturen verknüpfen müssen, klicken Sie auf die Schaltfläche **Input hinzufügen** ( ) und erstellen Sie so viele `nodes/rows`-Inputs, wie nötig. Wenn Sie einen `nodes/rows`-Input entfernen

müssen, klicken Sie auf die Schaltfläche **Input löschen** (⊠). Beachten Sie, dass mindestens zwei Strukturen für eine Join-Verknüpfung benötigt werden, daher steht die Schaltfläche ⊠ nur zur Verfügung, wenn mehr als zwei Inputs vorhanden sind.



Wenn eine Join-Komponente mehrere Inputs hat, muss in den Join-Bedingungen ebenfalls jeder der zu verknüpfenden Inputs berücksichtigt werden, siehe Hinzufügen von Join-Bedingungen 898. Ein Schritt-für-Schritt-Beispiel für die Verknüpfung mehrerer Datenbanktabellen mittels Join finden Sie im Beispiel: Erstellen eines CSV-Berichts anhand mehrerer Tabellen 357.

5.4.3 Beispiel: Verknüpfen von XML-Strukturen

In diesem Beispiel wird gezeigt, wie Sie Daten aus zwei XML-Strukturen auf Basis von Bedingungen mittels einer Join-Komponente miteinander verknüpfen. Das Mapping-Beispiel dazu finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\JoinPeopleInfo.mfd**.

Ziel dieses Mappings ist es, Personendaten (Vorname, Familienname, Adresse, E-Mail und Telefonnummer) aus zwei XML-Quelldateien in einer einzigen XML-Zieldatei zu sammeln.

In der ersten XML-Datei sind Vor- und Zunamen der einzelnen Personen sowie ihre E-Mail-Adresse und Telefonnummer gespeichert (siehe Beispielcodefragment unten. Beachten Sie, dass die XML-Deklaration, der Namespace und einige Datensätze aus Gründen der Einfachheit weggelassen wurden):

```
<People>
   <Person>
      <FirstName>Marquita</FirstName>
      <LastName>Bailey</LastName>
      <Email>m.bailey@nanonull.com</Email>
      <Phone>555323698</Phone>
   </Person>
   <Person>
      <FirstName>Totie</FirstName>
      <LastName>Rea</LastName>
      <Email>t.rea@nanonull.com</Email>
      <Phone>555598653</Phone>
   </Person>
</People>
```

People.xml

Die zweite XML-Datei enthält die Vor- und Zunamen der einzelnen Personen sowie deren Adressdaten:

 © 2016-2022 Altova GmbH

```
<Addresses>
   <Address>
      <FirstName>Marquita</FirstName>
      <LastName>Bailey</LastName>
      <City>Bridgedell</City>
      <Street>Olive Street</Street>
      <Number>4</Number>
   </Address>
   <Address>
      <FirstName>Totie</FirstName>
      <LastName>Rea</LastName>
      <City>Roseford</City>
      <Street>Evergreen Lane</Street>
      <Number>34</Number>
   </Address>
</Addresses>
```

Addresses.xml

Ziel dieses Mappings ist es, die `<Person>`-Daten aus der ersten Datei mit den `<Address>`-Daten aus der zweiten Datei für die Datensätze, in denen der Vor- und Zuname (FirstName und LastName) übereinstimmt, zu kombinieren. Dazu müssen für jedes `<Person>`-Element in der ersten Datei und für jedes `<Address>`-Element in der zweiten Datei die Elemente `FirstName` und `LastName` verglichen werden. Wenn beide Werte gleich sind, so beziehen sich die entsprechenden `<Person>` und `<Address>`-Datensätze auf dieselbe Person und müssen miteinander verknüpft werden. Die XML-Zielstruktur sollte folgendermaßen aussehen:

```
<PeopleInfo>
   <Row>
      <FirstName>Marquita</FirstName>
      <LastName>Bailey</LastName>
      <City>Bridgedell</City>
      <Street>Olive Street</Street>
      <Number>4</Number>
      <Email>m.bailey@nanonull.com</Email>
      <Phone>555323698</Phone>
   </Row>
   <Row>
      <FirstName>Totie</FirstName>
      <LastName>Rea</LastName>
      <City>Roseford</City>
      <Street>Evergreen Lane</Street>
      <Number>34</Number>
      <Email>t.rea@nanonull.com</Email>
      <Phone>555598653</Phone>
   </Row>
</PeopleInfo>
```

PeopleInfo.xml

Das gewünschte Ergebnis kann ganz einfach durch Hinzufügen einer Join-Komponente zum Mapping erzielt werden. Beachten Sie, dass Sie dasselbe Ergebnis auch mit Hilfe anderer Komponententypen erzielen; in der Anleitung unten wird hierfür jedoch eine Join-Komponente verwendet, was das Thema dieses Beispiels ist.

Um das erforderliche Mapping zu erstellen, gehen Sie folgendermaßen vor:

Schritt 1: Hinzufügen der XML-Quelldateien zum Mapping

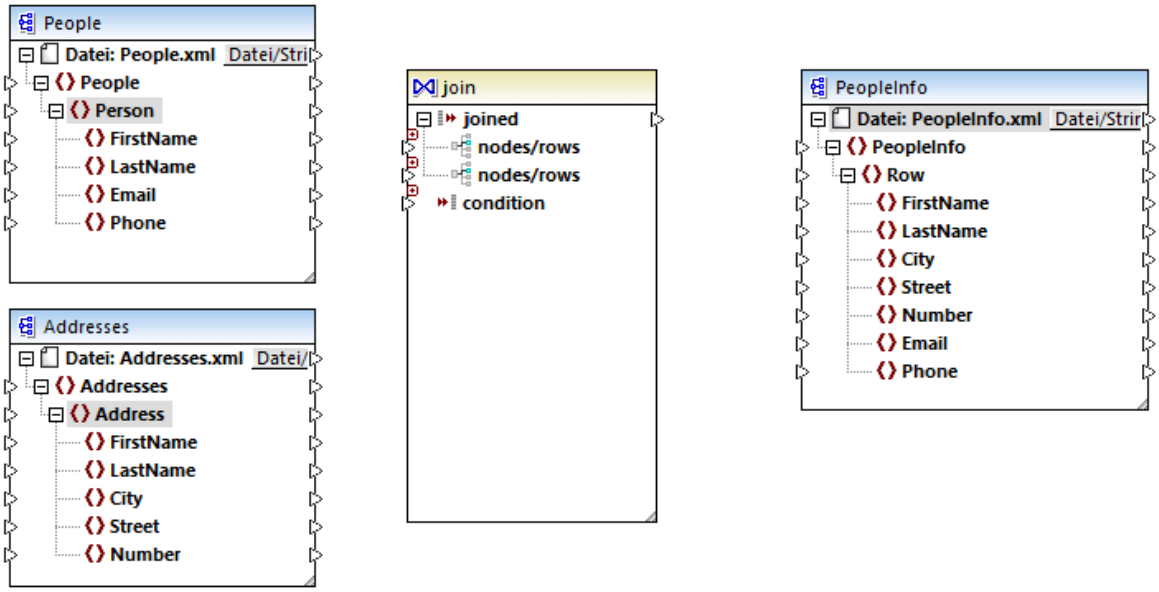
1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur folgenden Quelldatei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\People.xml**.
2. Wiederholen Sie den obigen Schritt für die Datei **Addresses.xml** (zweite Quelldatei).

Schritt 2: Hinzufügen der Schema-Zieldatei zum Mapping

- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur Datei **<Dokumente>\Altova\MapForce2023\MapForceExamples\People.xsd** (XSD-Schema-Zieldatei). Wenn Sie aufgefordert werden, eine XML-Beispieldatei bereitzustellen, klicken Sie auf **Überspringen**. Wenn Sie aufgefordert werden, ein Root-Element auszuwählen, wählen Sie `PeopleInfo` als Root-Element aus.

Schritt 3: Hinzufügen der Join-Komponente

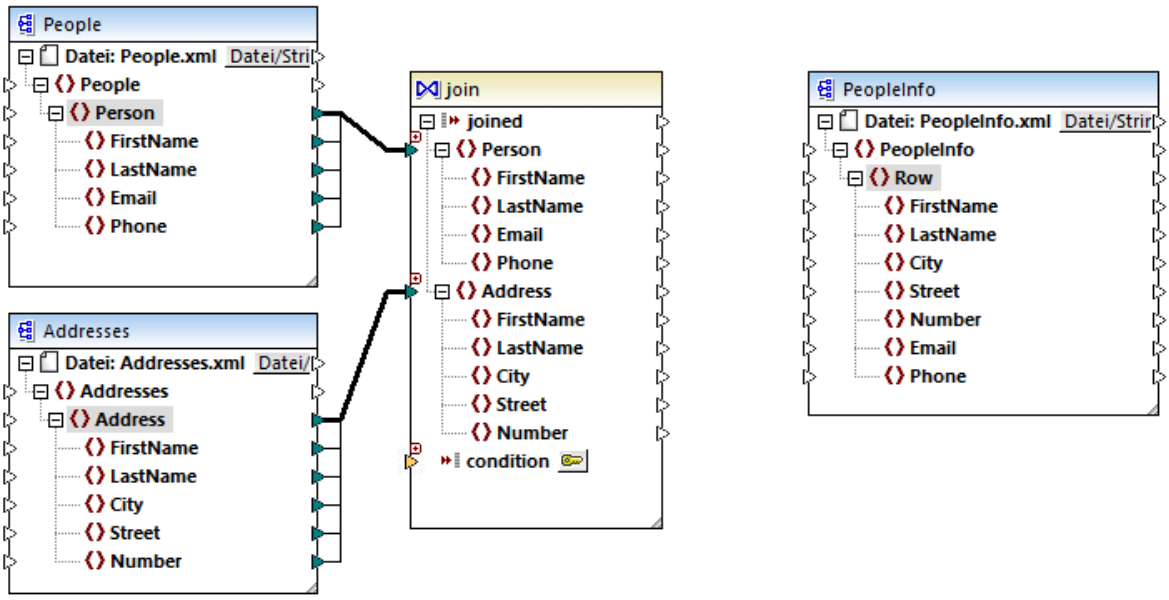
1. Klicken Sie im Menü **Einfügen** auf **Join**. (Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche **Join**). Das Mapping sollte zu diesem Zeitpunkt folgendermaßen aussehen (Sie müssen die Komponenten mit der Maus ziehen und deren Größe anpassen, damit sie aussehen, wie unten gezeigt):



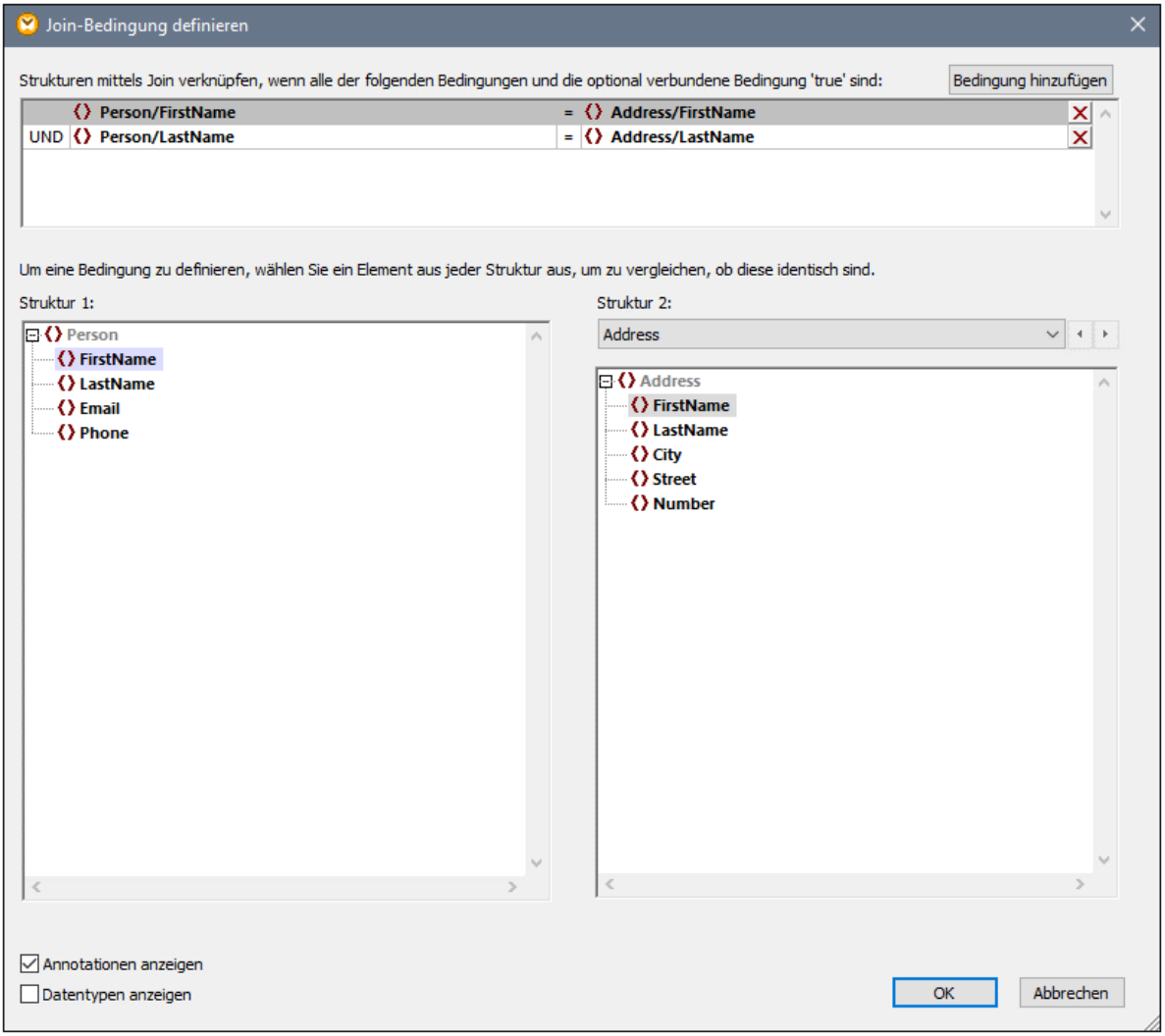
Wenn Sie einen Blick auf die Join-Komponente werfen, sehen Sie, dass sie zwei `nodes/rows`-Datenelemente hat, über die sie mit den zwei zu vergleichenden Strukturen verbunden werden kann (in diesem Fall mit den Strukturen `Person` und `Address`).

© 2016-2022 Altova GmbH

2. Ziehen Sie eine Verbindung von `Person` zum ersten `nodes/rows`-Datenelement der Join-Komponente. Verbinden Sie auf die gleiche Weise `Address` mit dem zweiten `nodes/rows`-Datenelement.



3. Wie bereits erwähnt, kann die Verknüpfung nur durchgeführt werden, wenn die Werte von `FirstName` und `LastName` in beiden Strukturen identisch sind. Klicken Sie auf die Schaltfläche **Join-Bedingung definieren** , um diese Bedingung zu definieren.
4. Wählen Sie das Datenelementpaar zur Definition der ersten Join-Bedingung aus (`FirstName` unter Struktur 1 und `FirstName` unter Struktur 2).
5. Klicken Sie auf **Bedingung hinzufügen** und wiederholen Sie den Schritt für `LastName`.



In einigen Mappings genügt eventuell eine aus einem einzigen Vergleich bestehende Bedingung für die Join-Verknüpfung. In diesem Beispiel müssen jedoch zwei Vergleiche erstellt werden:

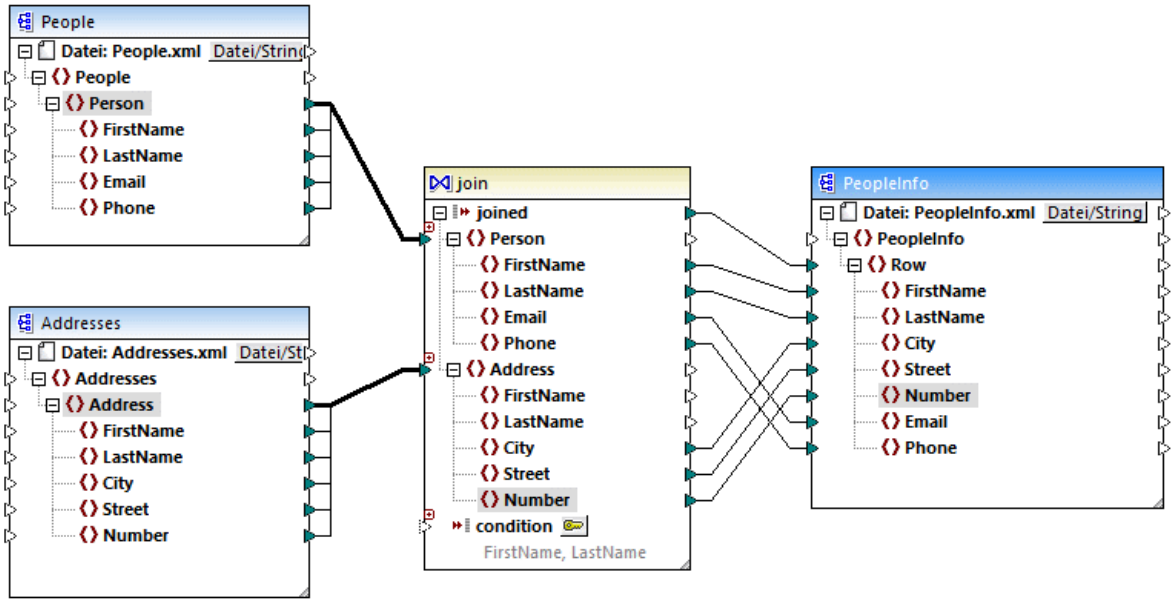
1) `FirstName` in Struktur 1 = `FirstName` in Struktur 2
2) `LastName` in Struktur 1 = `LastName` in Struktur 2.

Wenn mehrere Bedingungen definiert werden, *müssen alle davon "true" ergeben, damit die Join-Verknüpfung erstellt wird*. In diesem Beispiel wird die Join-Verknüpfung daher nur erstellt, wenn beide Vergleiche "true" ergeben. Wenn nur eine der obigen Bedingungen definiert würde, würden auch Personen-Datensätze verknüpft, die denselben Vornamen (FirstName) aber einen anderen Nachnamen (LastName) haben.

Schritt 4: Mappen der Join-Komponente auf das Zielschema

Nachdem nun die beiden Strukturen miteinander verknüpft wurden, können Sie definieren, welche Datenelemente der miteinander verknüpften Struktur auf die Zielkomponente gemappt werden sollen. Ziehen Sie dazu, wie unten gezeigt, Verbindungen von Datenelementen beider verknüpften Strukturen auf die Zielkomponente. Die Verbindung zwischen `joined` und `Row` hat den folgenden Zweck: Immer wenn die Join-Bedingung erfüllt wird, wird in der Zielkomponente ein neues `Row`-Datenelement erstellt.

 © 2016-2022 Altova GmbH



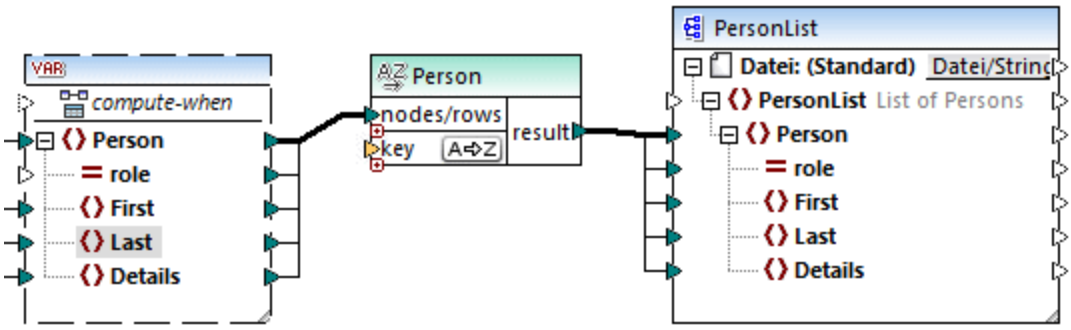
Klicken Sie auf das Register **Ausgabe**, um eine Vorschau auf das Mapping-Ergebnis zu sehen. Wie erwartet enthält jeder Personendatensatz (<Row>) nun die vollständigen, aus zwei verschiedenen Quelle miteinander verbundenen Adressdaten.

5.5 Sortieren von Komponenten

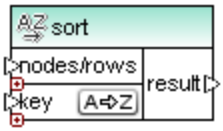
Um Input-Daten auf Basis eines bestimmten Sortierschlüssels zu sortieren, verwenden Sie bitte die Sortierkomponente ("sort"). Die Sortierkomponente unterstützt die folgenden Zielsprachen: XSLT2, XQuery und den Built-in-Ausführungsprozessor. Wenn die Transformationssprache "Built-in" ist, können Datenbanktabellendaten mit Hilfe der Sortierkomponente sortiert werden. Besser lässt sich dies jedoch mit Hilfe einer SQL-WHERE/ORDER-Komponente bewerkstelligen. Nähere Informationen dazu finden Sie unter Filtern und Sortieren von Datenbankdaten (SQL WHERE/ORDER) 361.

Um eine Sortierkomponente zum Mapping hinzuzufügen, gehen Sie folgendermaßen vor:

- Klicken Sie mit der rechten Maustaste auf eine vorhandene Verbindung und wählen Sie im Kontextmenü den Befehl **Sortierung einfügen: Nodes/Zeilen**. Daraufhin wird die Sortierkomponente eingefügt und automatisch mit der Quell- und Zielkomponente verbunden. Im unten gezeigten Mapping wurde die Sortierkomponente z.B. zwischen einer Variablen und einer XML-Komponente eingefügt. Jetzt muss nur noch der Sortierschlüssel (das Feld, anhand dessen sortiert werden soll) manuell verbunden werden.



- Klicken Sie im Menü **Einfügen** auf **Sortieren** (oder alternativ dazu auf die Symbolleisten-Schaltfläche **Sortieren**). Daraufhin wird die Sortierkomponente in ihrer nicht verbundenen Form eingefügt.

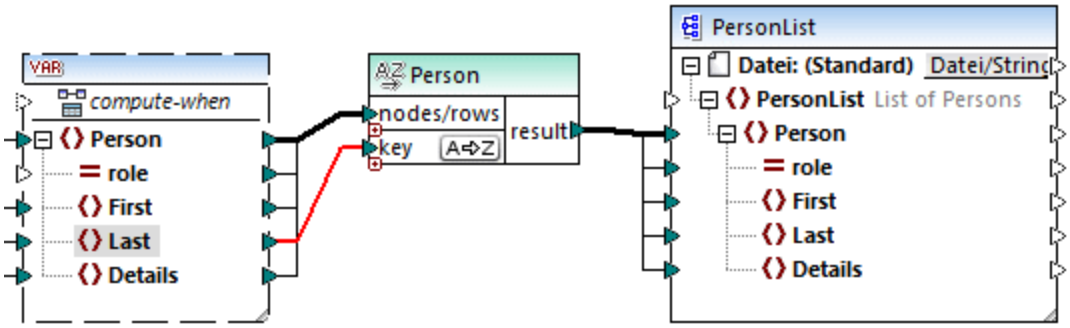


  Sobald eine Verbindung zur Quellkomponente hergestellt wurde, ändert sich der Name der Titelleiste in denjenigen des mit dem Datenelement `nodes/rows` verbundenen Datenelements.

So definieren Sie, nach welchem Datenelement sortiert werden soll:

- Verbinden Sie das Datenelement, nach dem sortiert werden soll, mit dem Parameter `key` der Sortierkomponente. Im unten gezeigten Mapping wurden die `Person`-Nodes/Zeilen z.B. nach dem Feld `Last` sortiert.

© 2016-2022 Altova GmbH

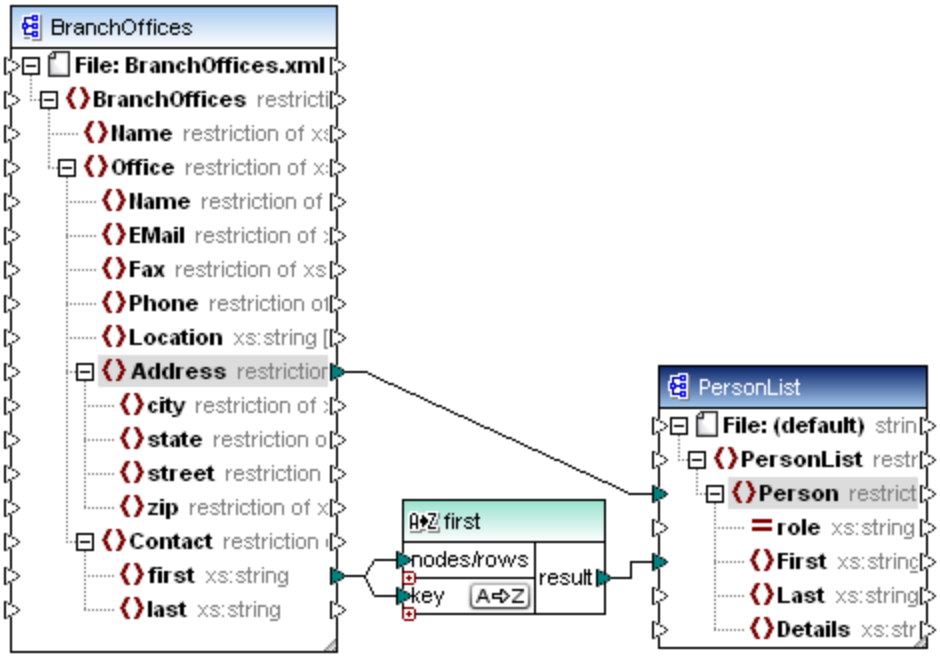


So ändern Sie die Sortierreihenfolge:

- Klicken Sie in der Sortierkomponente auf das Symbol A⇨Z. Es ändert sich daraufhin in Z➔A, um anzuzeigen, das die Sortierreihenfolge umgekehrt wurde.

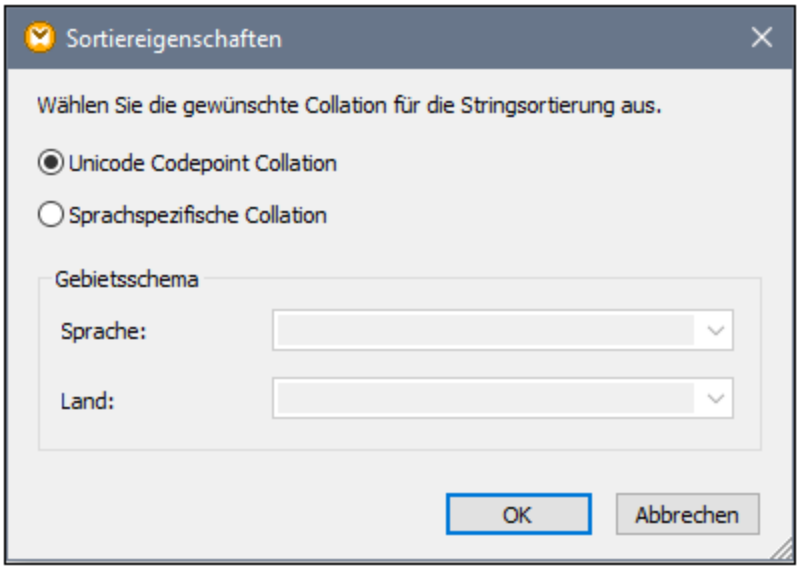
So sortieren Sie Input-Daten, die aus Einträgen vom Typ "simpleType" bestehen:

- Verbinden Sie das Datenelement mit den beiden Parametern `nodes/rows` und `key` der Sortierkomponente. Im unten gezeigten Mapping wird das Element vom simpleType `first` sortiert.



So sortieren Sie Strings mit Hilfe von sprachspezifischen Regeln:

- Doppelklicken Sie auf die Titelleiste der Sortierkomponente, um das Dialogfeld "Sortiereigenschaften" zu öffnen.



Unicode Codepoint Collation: Mit dieser Standardoption werden Strings auf Basis ihrer Codepoint-Werte verglichen/sortiert. Codepoint-Werte sind Ganzzahlen, die abstrakten Zeichen in dem vom Unicode Consortium abgesegneten universalen Zeichensatz zugewiesen sind. Mit Hilfe dieser Option kann eine Sortierung in den verschiedensten Sprachen und Scripts vorgenommen werden.

Sprachspezifische Collation: Mit dieser Option können Sie die Variante für eine bestimmte Sprache und ein bestimmtes Land festlegen, nach dem sortiert werden soll. Diese Option wird bei Verwendung des BUILT-IN-Ausführungsprozessors unterstützt. Im Fall von XSLT hängt die Unterstützung vom Prozessor ab, der für die Ausführung des Codes verwendet wird.

5.5.1 Sortieren nach mehreren Schlüsseln

Nachdem Sie eine Sortierkomponente zum Mapping hinzugefügt haben, wird standardmäßig ein einziger Sortierschlüssel namens `key` erstellt.



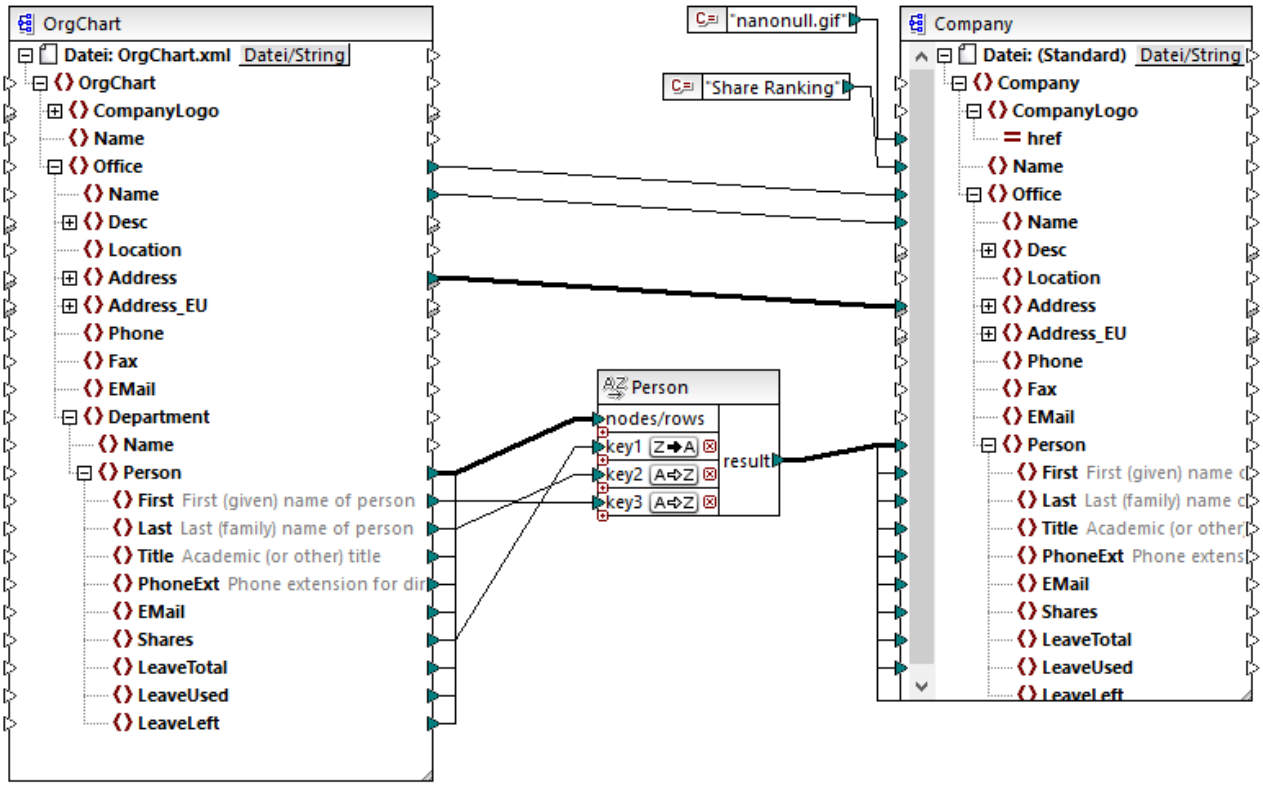
Standardsortierkomponente

Wenn Sie nach mehreren Schlüsseln sortieren möchten, passen Sie die Sortierkomponente folgendermaßen an:

- Klicken Sie auf das Symbol **Schlüssel hinzufügen** (⊡), um einen neuen Schlüssel (z.B. `key2` im Mapping unten) hinzuzufügen.
- Klicken Sie auf das Symbol **Schlüssel löschen** (⊠), um einen Schlüssel zu löschen.
- Ziehen Sie eine Verbindung auf das Symbol ⊡, um einen Schlüssel hinzuzufügen und gleichzeitig zu verbinden.

© 2016-2022 Altova GmbH

Unter dem folgenden Pfad finden Sie ein Mapping zum Veranschaulichung der Sortierung nach mehreren Schlüsseln: **<Dokumente>\Altova\MapForce2023\MapForceExamples\SortByMultipleKeys.mfd**.



SortByMultipleKeys.mfd

Im oben gezeigten Mapping wurden die `Person`-Datensätze nach drei Sortierschlüsseln sortiert:

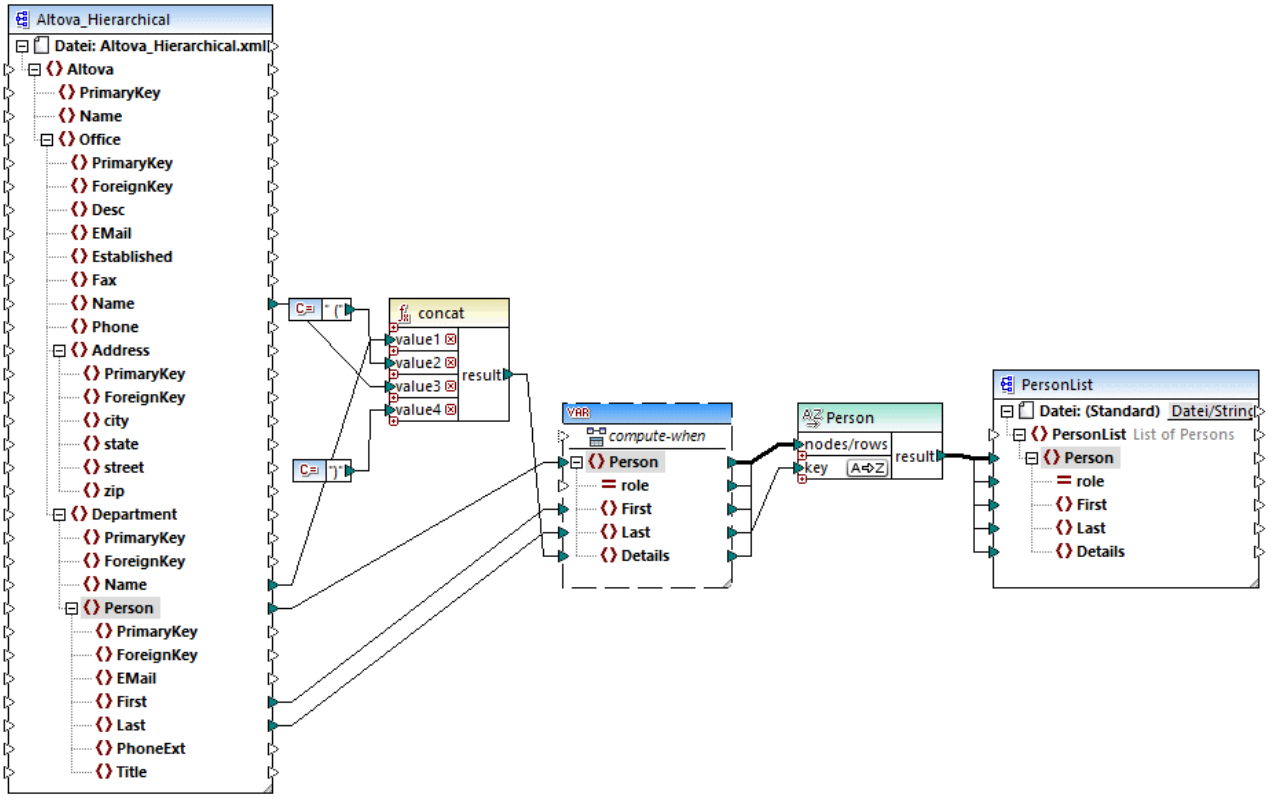
1. `Shares` (Anzahl der Aktien im Besitz einer Person)
2. `Last` (Nachname)
3. `First` (Vorname)

Beachten Sie, dass die Position des Sortierschlüssels in der Sortierkomponente die Sortierpriorität festlegt. So werden die Datensätze etwa im Beispiel oben zuerst nach der Anzahl der Aktien (shares) sortiert. Dies ist der Sortierschlüssel mit der höchsten Priorität. Wenn die Anzahl der Aktien dieselbe ist, werden die Personen nach ihrem Nachnamen sortiert. Wenn schließlich mehrere Personen dieselbe Anzahl von Aktien und denselben Nachnamen haben, wird der Vorname der Person berücksichtigt.

Die Sortierreihenfolge kann für jeden Schlüssel unterschiedlich sein. Im oben gezeigten Mapping hat der Schlüssel `Shares` eine absteigende Sortierreihenfolge (Z-A), während die anderen beiden Schlüssel eine aufsteigende Sortierreihenfolge haben (A-Z).

5.5.2 Sortieren mit Variablen

In einigen Fälle müssen eventuell Zwischenvariablen zum Mapping hinzugefügt werden, um das gewünschte Ergebnis zu erzielen. In diesem Beispiel wird gezeigt, wie Sie Datensätze aus einer XML-Dati extrahieren und mit Hilfe von Zwischenvariablen sortieren. Das Mapping-Beispiel dazu finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\Altova_Hierarchical_Sort.mfd**.



Altova_Hierarchical_Sort.mfd

In diesem Mapping werden Daten aus einer XML-Quelldatei namens **Altova_Hierarchical.xml** gelesen und in eine XML-Zieldatei geschrieben. Wie Sie oben sehen, enthält die XML-Quellkomponente Informationen über ein fiktives Unternehmen. Die Firma besteht aus Niederlassungen (Office). Die Niederlassungen sind in Abteilungen (Department) unterteilt, die wiederum aus Personen (Person) bestehen.

Die XML-Zielkomponente `PersonList` enthält eine Liste von `Person`-Datensätzen. Das Datenelement `Details` dient zur Aufnahme von Informationen über die Niederlassung und Abteilung, zu der die Person gehört.

Ziel ist es, alle Personen aus der XML-Quelldatei zu extrahieren und alphabetisch nach dem Nachnamen (last name) zu sortieren. Außerdem müssen der Name der Niederlassung und der Abteilung, zu der jede Person gehört, in das Datenelement `Details` geschrieben werden.

Um dies zu erreichen, werden in diesem Beispiel die folgenden Komponententypen verwendet:

© 2016-2022 Altova GmbH

1. die Funktion **concat**. Diese Funktion gibt in diesem Mapping einen String im Format `Office(Department)` zurück. Sie erhält als Input den Namen der Niederlassung und den Abteilungsnamen und zwei Konstanten, die die öffnende und schließende Klammer bereitstellen. Siehe auch Hinzufügen einer Funktion zum Mapping 943.
2. eine Zwischenvariable. Die Rolle der Variablen ist es, alle für eine Person relevanten Daten in denselben Mapping-Kontext zu bringen. Aufgrund der Variablen werden im Kontext jeder Person jeweils die Abteilung und die Niederlassung einer Person nachgesehen. Anders ausgedrückt, merkt sich die Variable den Namen der Niederlassung und Abteilung, zu der eine Person gehört. Ohne die Variable wäre der Kontext falsch, was im Mapping zu unerwünschten Ergebnissen führen würde (eine genauere Anleitung zum Ausführen eines Mappings finden Sie unter Mapping-Regeln und -Strategien 1330). Beachten Sie, dass die Variable die Struktur der XML-Zieldatei repliziert (sie verwendet dasselbe XML-Schema). Auf diese Art kann das Sortierergebnis mittels einer "Alles kopieren"-Verbindung mit der Zielkomponente verbunden werden. Siehe auch Verwendung von Variablen 882 und "Alles kopieren"-Verbindungen 97.
3. eine Sortierkomponente, die die eigentliche Sortierung durchführt. Beachten Sie, dass der Input "key" der `sort`-Komponente mit dem Datenelement `Last` der Variablen verbunden ist, wodurch alle Personendatensätze nach dem Nachnamen sortiert werden.

5.6 Filter und Bedingungen

Wenn Sie Daten filtern oder basierend auf einer Bedingung einen Wert ermitteln müssen, können Sie dazu die folgenden Komponententypen verwenden:

- Filter: Nodes/Zeilen ( )
- SQL WHERE/ORDER ( )
- If-Else-Bedingung ( )

Sie können diese Komponenten entweder über das Menü **Einfügen** oder über die Symbolleiste **Komponente einfügen** zum Mapping hinzufügen. Beachten Sie, dass jede der obigen Komponenten ein bestimmtes Verhalten aufweist und bestimmte Voraussetzungen hat. Die Unterschiede sind in den folgenden Abschnitten erklärt.

Filtern von Nodes oder Zeilen

Um Daten, darunter XML-Nodes oder CSV-Zeilen, zu filtern, verwenden Sie eine **Filter: Nodes/Zeilen**-Komponente. Mit Hilfe der **Filter: Nodes/Zeilen**-Komponente können Sie eine Untergruppe von Nodes anhand einer true- oder false-Bedingung aus einer größeren Datenmenge abrufen. Ihre Struktur im Mapping-Bereich sieht folgendermaßen aus:



In der oben gezeigten Struktur definiert die mit **bool** verbundene Bedingung, ob der verbundene Node/die verbundene Zeile in den **on-true** oder in den **on-false** Output geleitet wird. Wenn die Bedingung true lautet, wird **der Node/die Zeile** an den **on-true**-Output übergeben. Umgekehrt, wenn die Bedingung false ist, wird der **Node/die Zeile** an den **on-false**-Output übergeben.

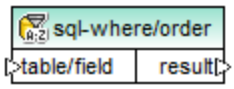
Wenn Sie in Ihrem Mapping nur Datenelemente, die die Filterbedingung erfüllen, verwenden möchten, können Sie den **on-false**-Output unverbunden lassen. Wenn die Datenelemente, die die Filterbedingung *nicht erfüllen*, verarbeitet werden müssen, verbinden Sie den **on-false**-Output mit einer Zielkomponente für solche Datenelemente. Wenn Sie eine Ausnahme hinzufügen möchten, falls die Filterbedingung nicht erfüllt wird, muss der **on-false**-Output verbunden werden (siehe Hinzufügen von Ausnahmeereignissen 938).

Eine schrittweise Anleitung anhand eines Mapping-Beispiels finden Sie unter Beispiel: Filtern von Nodes 916.

Filtern von Datenbankdaten

Filter: Nodes/Zeilen-Komponenten können Daten aus jeder anderen von MapForce unterstützten Komponentenstruktur filtern, darunter auch Datenbankdaten. Wenn Sie jedoch Daten aus einer Datenbank filtern möchten, wird empfohlen, stattdessen eine **SQL WHERE/ORDER**-Komponente zu verwenden. Die **SQL WHERE/ORDER**-Komponente ist für das Arbeiten mit Datenbanken optimiert und funktioniert hier besser als eine **Filter: Nodes/Zeilen**-Komponente.

© 2016-2022 Altova GmbH



Nähere Informationen zu solchen Komponenten finden Sie unter SQL WHERE/ORDER-Komponente[361].

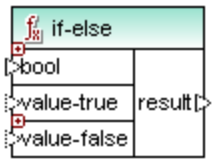
Rückgabe eines Werts auf Basis einer Bedingung

Wenn Sie einen einzelnen Wert (und nicht einen Node oder eine Zeile) auf Basis einer Bedingung abrufen möchten, verwenden Sie eine **If-Else-Bedingung**. Beachten Sie, dass If-Else-Bedingungen sich nicht zum Filtern von Nodes oder Zeilen eignen. Im Gegensatz zu **Filter: Nodes/Zeilen**-Komponenten gibt eine **If-Else-Bedingung** einen Wert vom Typ simpleType (z.B. einen String oder eine Ganzzahl) zurück. **If-Else-Bedingungen** eignen sich daher nur für Szenarien, in denen ein einzelner Wert anhand einer Bedingung verarbeitet werden soll. Angenommen, Sie haben eine Liste von Durchschnittstemperaturen pro Monat im folgenden Format:

```
<Temperatures>
   <data temp="19.2" month="2010-06" />
   <data temp="22.3" month="2010-07" />
   <data temp="19.5" month="2010-08" />
   <data temp="14.2" month="2010-09" />
   <data temp="7.8" month="2010-10" />
   <data temp="6.9" month="2010-11" />
   <data temp="-1.0" month="2010-12" />
</Temperatures>
```

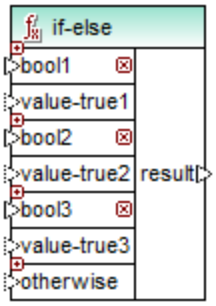
Mit Hilfe einer **If-Else-Bedingung** können Sie für jedes Datenelement in der Liste, dessen Temperaturwert höher als 20 Grad Celsius beträgt, den Wert "hoch" und für jedes Datenelement, dessen Temperaturwert niedriger als 5 Grad Celsius beträgt, "niedrig" zurückgeben.

Im Mapping sieht die Struktur der **If-Else-Bedingung** folgendermaßen aus:



Wenn die mit **bool** verbundene Bedingung true ist, so wird der mit **value-true** verbundene Wert als **result** ausgegeben. Wenn die Bedingung false ist, so wird der mit **value-false** verbundene Wert als **result** ausgegeben. Der Datentyp von **result** ist im Vorhinein nicht bekannt; er hängt vom Datentyp des mit **value-true** oder **value-false** verbundenen Werts ab. Wichtig ist, dass es sich immer um einen simpleType (String, Ganzzahl, usw.) handeln muss. Die Verbindung von Input-Werten von complexTypes (wie Nodes oder Zeilen) wird von **If-Else-Bedingungen** nicht unterstützt.

If-Else-Bedingungen sind erweiterbar, d.h. Sie können durch Klick auf die Schaltfläche **Hinzufügen** (⊡) mehrere Bedingungen zur Komponente hinzufügen. Um eine zuvor hinzugefügte Bedingung zu löschen, klicken Sie auf die Schaltfläche **Löschen** (⊠). Auf diese Art können Sie mehrere Bedingungen überprüfen und für jede Bedingung einen anderen Wert zurückgeben, falls die Bedingung true ist.



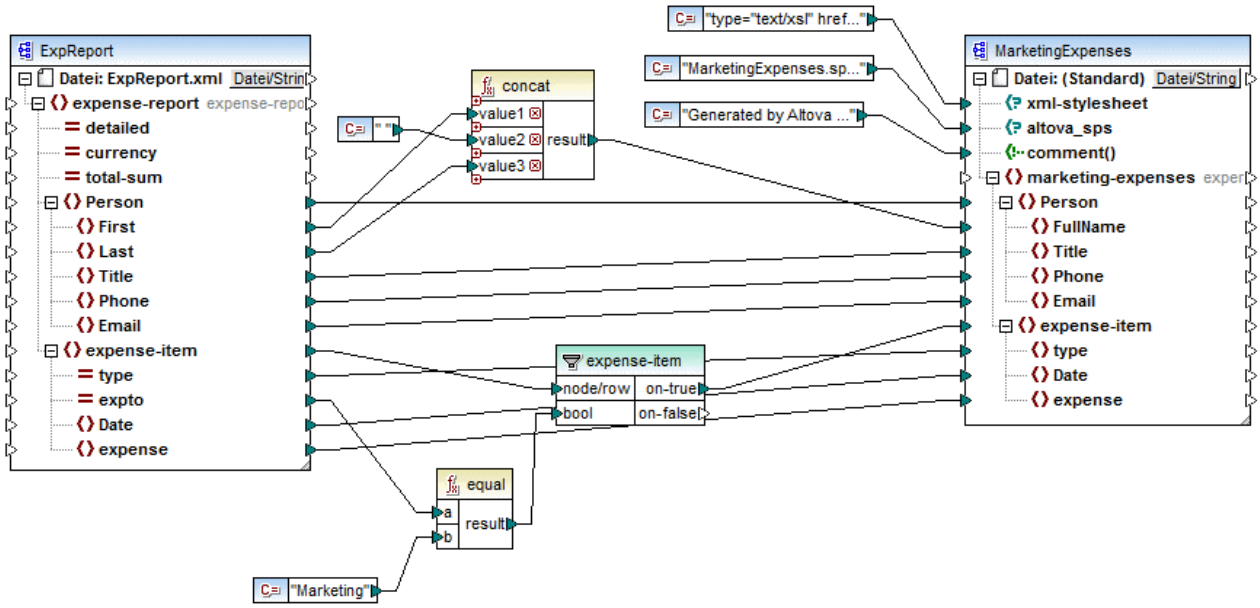
Erweiterte **If-Else-Bedingungen** werden von oben nach unten ausgewertet (die erste Bedingung wird als erste überprüft, anschließend die zweite, usw.). Wenn ein Wert zurückgegeben werden soll, wenn keine der Bedingungen true ist, verbinden Sie sie mit **otherwise**.

Eine schrittweise Anleitung anhand eines Mapping-Beispiels finden Sie unter Beispiel: Rückgabe eines Werts auf Basis einer Bedingung 918.

5.6.1 Beispiel: Filtern von Nodes

In diesem Beispiel wird gezeigt, wie Sie Nodes auf Basis einer true/false-Bedingung filtern können. Zu diesem Zweck wird eine **Filter: Nodes/Zeilen** ( )-Bedingung eingesetzt. Das in diesem Beispiel gezeigte Verfahren lässt sich nicht nur auf XML-Daten, sondern genauso auch auf andere Komponententypen wie CSV oder Text anwenden. Sie können für Datenbanken zwar einen Filter verwenden, doch empfiehlt sich in diesem Fall die Verwendung einer SQL WHERE/ORDER-Komponente, da dies bessere Ergebnisse liefert (siehe SQL WHERE/ORDER-Komponente 361).

Das in diesem Beispiel beschriebene Mapping befindet sich unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\MarketingExpenses.mfd**.



 © 2016-2022 Altova GmbH

Wie oben gezeigt, liest das Mapping Daten aus einer XML-Quelldatei, die einen Spesenbericht enthält aus ("ExpReport") und schreibt Daten in eine XML-Zieldatei ("MarketingExpenses"). Zwischen der Ziel- und der Quellkomponente befinden sich einige weitere Komponenten. Die wichtigste davon ist der **expense-item**-Filter ( ), der das Thema dieses Kapitels ist.

Ziel des Mappings ist es, nur die Ausgabenposten herauszufiltern, die der Marketing-Abteilung zugeschrieben werden. Zu diesem Zweck wurde eine Filter-Komponente zum Mapping hinzugefügt. (Um einen Filter hinzuzufügen, klicken Sie auf das Menü **Einfügen** und wählen Sie den Befehl **Filter: Nodes/Zeilen**.)

Um festzustellen, ob ein Ausgabenposten zu Marketing gehört, wird der Wert des Attributs "expto" in der Quellkomponente überprüft. Dieses Attribut hat den Wert "Marketing", wenn es sich um eine Marketing-Ausgabe handelt. Im Codefragment unten gehören das erste und dritte Datenelement zu Marketing, das zweite zu Development und das vierte zu Sales:

```
...
   <expense-item type="Meal" expto="Marketing">
      <Date>2003-01-01</Date>
      <expense>122.11</expense>
   </expense-item>
   <expense-item type="Lodging" expto="Development">
      <Date>2003-01-02</Date>
      <expense>122.12</expense>
   </expense-item>
   <expense-item type="Lodging" expto="Marketing">
      <Date>2003-01-02</Date>
      <expense>299.45</expense>
   </expense-item>
   <expense-item type="Entertainment" expto="Sales">
      <Date>2003-01-02</Date>
      <expense>13.22</expense>
   </expense-item>
...
```

XML-Input vor Ausführung des Mappings

Im Mapping-Bereich wird der **node/row**-Input des Filters mit dem **expense-item**-Node in der Quellkomponente verbunden. Damit wird gewährleistet, dass die Filterkomponente die Liste der zu verarbeitenden Nodes erhält.

Als Bedingung, anhand welcher die Filterung erfolgen soll, haben wir die **equal**-Funktion aus der MapForce core-Bibliothek hinzugefügt (nähere Informationen dazu finden Sie unter Hinzufügen einer Funktion zum Mapping 943). Die **equal**-Funktion vergleicht den Wert des Attributs `expto` mit einer Konstante, die den Wert "Marketing" hat. (Um eine Konstante hinzuzufügen, klicken Sie auf das Menü **Einfügen** und anschließend auf **Konstante**.)

Da wir nur diejenigen Datenelemente, die die Bedingung erfüllen, filtern müssen, haben wir nur den **on-true**-Input des Filters mit der Zielkomponente verbunden.

Wenn Sie durch Klicken auf das Register **Ausgabe** eine Vorschau auf das Mapping-Ergebnis anzeigen, wertet MapForce für jeden expense-item-Node die mit dem **bool**-Input des Filters verbundene Bedingung aus. Wenn die Bedingung true ist, wird der expense-item-Node an die Zielkomponente übergeben; andernfalls wird er ignoriert. In der Ausgabe werden folglich nur die Ausgabenposten, die den Kriterien entsprechen, angezeigt:

```
...
   <expense-item>
      <type>Meal</type>
      <Date>2003-01-01</Date>
      <expense>122.11</expense>
   </expense-item>
   <expense-item>
      <type>Lodging</type>
      <Date>2003-01-02</Date>
      <expense>299.45</expense>
   </expense-item>
...
```

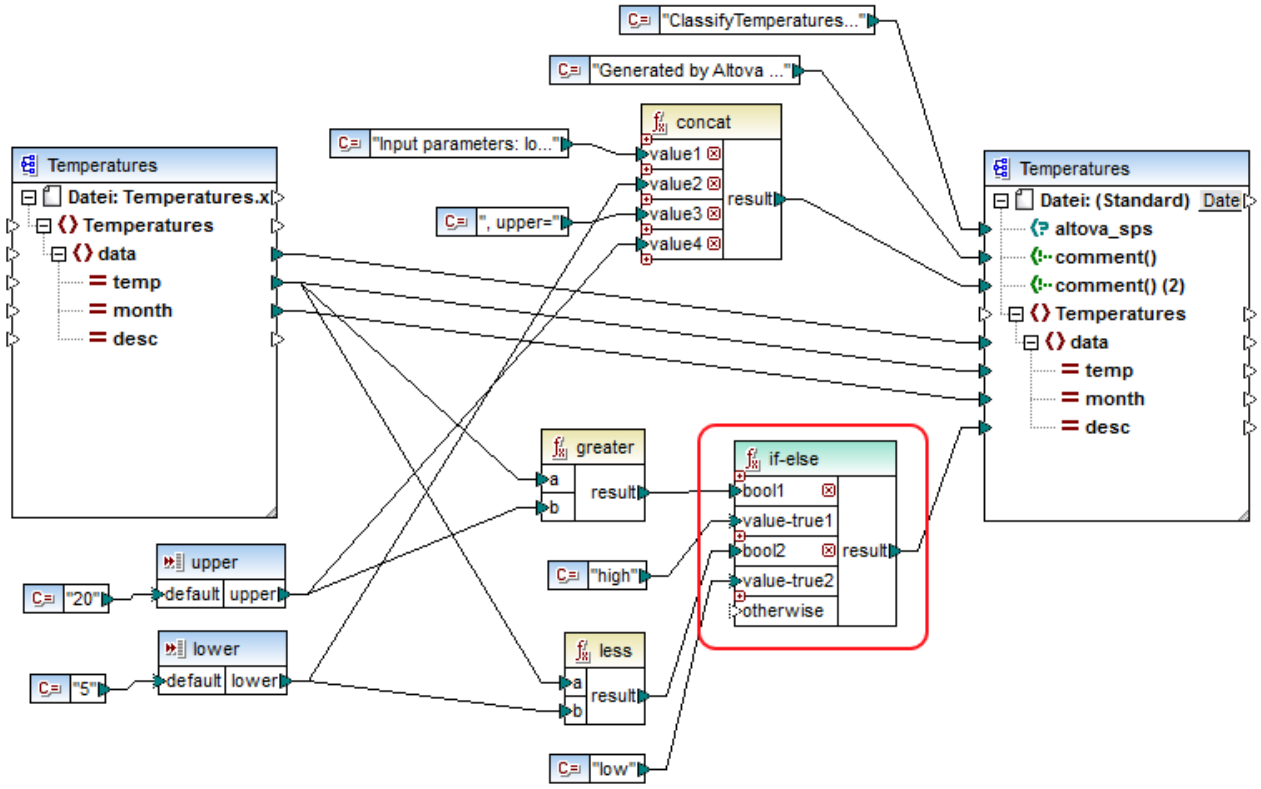
XML-Ausgabe nach Ausführung des Mappings

5.6.2 Beispiel: Rückgabe eines Werts auf Basis einer Bedingung

In diesem Beispiel wird gezeigt, wie Sie einen einfachen Wert auf Basis einer true/false-Bedingung aus einer Komponente abrufen. Zu diesem Zweck wird eine **If-Else-Bedingung** () verwendet. **If-Else-Bedingungen** dürfen nicht mit Filterkomponenten verwechselt werden. **If-Else-Bedingungen** eignen sich nur, um einfache Werte (Strings, Ganzzahlen, usw.) auf Basis einer Bedingung zu verarbeiten. Wenn Sie komplexe Werte wie Nodes filtern müssen, verwenden Sie stattdessen einen Filter (siehe Beispiel: Filtern von Nodes 916).

Das in diesem Beispiel beschriebene Mapping befindet sich unter dem folgenden Pfad: **%EXFOLDER%>ClassifyTemperatures.mfd**.

© 2016-2022 Altova GmbH



In diesem Mapping werden aus einer XML-Quelldatei, die Temperaturdaten ("Temperatures") enthält, Daten ausgelesen und in eine XML-Zieldatei geschrieben, die demselben Schema entspricht. Zwischen der Ziel- und der Quellkomponente befinden sich einige weitere Komponenten. Eine davon ist die hier beschriebene **if-else**-Bedingung (rot markiert).

Ziel des Mappings ist es, zu jedem Temperaturdatensatz in der Zielkomponente eine kurze Beschreibung hinzuzufügen. Wenn die Temperatur mehr als 20 Grad Celsius ist, so soll die Beschreibung "high" lauten. Wenn die Temperatur weniger als 5 Grad Celsius ist, soll die Beschreibung "low" lauten. In allen anderen Fällen soll keine Beschreibung hinzugefügt werden.

Zu diesem Zweck benötigen wir eine bedingte Verarbeitung. Um dies zu erreichen, wurde eine If-Else-Bedingung zum Mapping hinzugefügt. (Um eine If-Else-Bedingung hinzuzufügen, klicken Sie auf das Menü **Einfügen** und wählen Sie den Befehl **If-Else-Bedingung**.). Die If-Else-Bedingung in diesem Mapping wurde (mittels der Schaltfläche ⊡) erweitert, um zwei Bedingungen Platz zu bieten: **bool1** und **bool2**.

Die Bedingungen selbst werden von den aus der MapForce Core-Bibliothek hinzugefügten Funktionen **`greater`** und **`less`** bereitgestellt (nähere Informationen dazu finden Sie unter Hinzufügen einer Funktion zum Mapping 943). Diese Funktionen werten die von den beiden Input-Komponenten "upper" und "lower" bereitgestellten Werte aus. (Um eine Input-Komponente hinzuzufügen, klicken Sie auf das Menü **Einfügen** und anschließend auf **Input-Komponente einfügen**. Nähere Informationen zu Input-Komponenten finden Sie unter Bereitstellen von Parametern für das Mapping 867.)

Die Funktionen **`greater`** und **`less`** geben entweder true oder false zurück. Vom Ergebnis der Funktion ist es abhängig, was in die Zielinstanz geschrieben wird. Wenn also der Wert des Attributs "temp" in der Quellkomponente größer als 20 ist, so wird der Konstantenwert "high" an die **if-else**-Bedingung übergeben.

Wenn der Wert des Attributs "temp" in der Quellkomponente kleiner als 5 ist, so wird der Konstantenwert "low" an die **if-else**-Bedingung übergeben. Der **otherwise**-Input wird nicht verbunden. Wenn daher keine der obigen Bedingungen zutrifft, wird nichts an den **result**-Output-Konnektor übergeben.

Der **result** Output-Konnektor übergibt schließlich diesen Wert (einmal für jeden Temperaturdatensatz) an das Attribut "desc" in der Zielkomponente.

Wenn Sie fertig sind, klicken Sie auf das Register **Ausgabe**, um eine Vorschau auf das Mapping-Ergebnis zu sehen. Beachten Sie, dass die erzeugte XML-Ausgabe nun in allen Nodes, in denen "temperature" größer als 20 oder kleiner als 5 ist, das Attribut "desc" enthält.

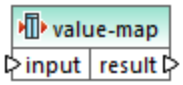
```
...
   <data temp="-3.6" month="2006-01" desc="low"/>
   <data temp="-0.7" month="2006-02" desc="low"/>
   <data temp="7.5" month="2006-03"/>
   <data temp="12.4" month="2006-04"/>
   <data temp="16.2" month="2006-05"/>
   <data temp="19" month="2006-06"/>
   <data temp="22.7" month="2006-07" desc="high"/>
   <data temp="23.2" month="2006-08" desc="high"/>
...
```

XML-Ausgabe nach Ausführung des Mappings

© 2016-2022 Altova GmbH

5.7 Wertezuordnungen

Mit Hilfe der Wertezuordnungskomponente können Sie einen Wert mit Hilfe einer Lookup-Tabelle durch einen anderen Wert ersetzen. Eine solche Komponente verarbeitet immer nur einen Wert auf einmal, daher hat sie im Mapping einen **Input** und ein Ergebnis (**result**).



Eine Wertezuordnung eignet sich, um einzelne Datenelemente in zwei Datensätzen zu mappen, um Datenelemente zu ersetzen. So könnten Sie z.B. in Form von Zahlen ausgedrückte Wochentage (1, 2, 3, 4, 5, 6 und 7) auf die Namen der einzelnen Wochentage ("Montag", "Dienstag", usw.) mappen. Oder Sie können die Namen der Monate ("Januar", "Februar", "März", usw.) auf die Zahlendarstellung der einzelnen Monate (1, 2, 3, usw.) mappen. Zur Mapping-Laufzeit werden die entsprechenden Werte entsprechend Ihrer benutzerdefinierten Lookup-Tabelle ersetzt. Die Werte in den beiden Datensätzen können einen unterschiedlichen Typ haben, doch müssen für jeden Datensatz Werte desselben Datentyps gespeichert sein.

Wertezuordnungskomponenten eignen sich für einfache Lookup-Verfahren, bei denen jeder Wert im ersten Datensatz einem einzigen Wert im zweiten Datensatz entspricht. Wenn ein Wert nicht in der Lookup-Tabelle gefunden wird, können Sie ihn entweder durch einen benutzerdefinierten Wert oder durch einen leeren Wert ersetzen oder ihn unverändert an die zweite Komponente übergeben. Wenn Sie Werte anhand komplexerer Kriterien nachschlagen oder filtern müssen, verwenden Sie stattdessen eine der Filterkomponenten 914.

Wenn Sie Code generieren oder eine MapForce Server-Ausführungsdatei anhand des Mappings generieren, wird die Lookup-Tabelle in den generierten Code bzw. die generierte Datei eingebettet. Daher empfiehlt es sich nur dann direkt im Mapping eine Lookup-Tabelle zu definieren, wenn sich Ihre Daten nicht ständig ändern und die Tabelle nicht sehr groß ist (nicht größer als einige hundert Einträge). Wenn sich die Lookup-Daten regelmäßig ändern, ist es oft schwierig sowohl das Mapping als auch den generierten Code regelmäßig auf aktuellem Stand zu halten. Es ist einfacher, die Lookup-Tabelle in Form einer Text-, XML-, Datenbank- oder eventuell Excel-Datei zu warten.

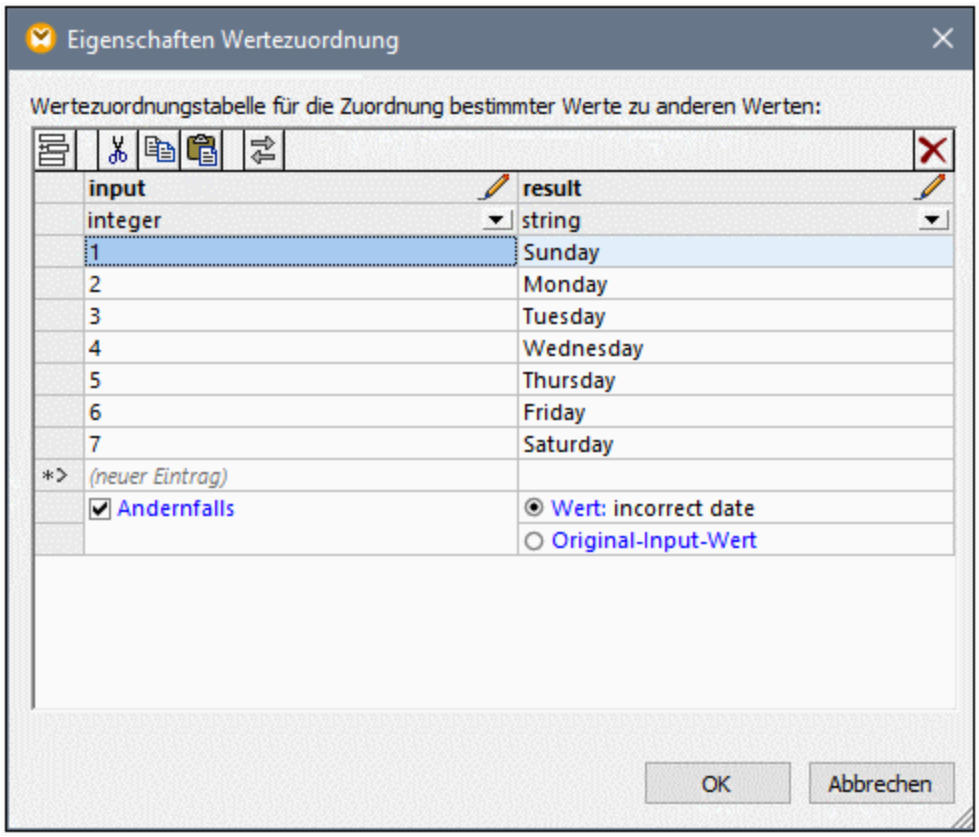
Bei sehr großen Lookup-Tabellen wird die Mapping-Ausführung durch die Lookup-Tabelle verlangsamt. In diesem Fall wird empfohlen, stattdessen eine Datenbankkomponente mit SQL-Where 361 zu verwenden. Besonders gut dafür eigenen sich aufgrund Ihrer Portabilität SQLite-Datenbanken. Serverseitig können Sie die Performance von Lookup-Tabellen durch Ausführung eines Mappings mit der MapForce Server oder MapForce Server Advanced Edition verbessern.

Erstellen von Wertezuordnungen

Um eine Wertezuordnungskomponente zum Mapping hinzuzufügen, wählen Sie eine der folgenden Methoden:

- Klicken Sie auf die Symbolleisten-Schaltfläche **Wertezuordnung einfügen** .
- Klicken Sie im Menü **Einfügen** auf **Wertezuordnung**.
- Klicken Sie mit der rechten Maustaste auf eine Verbindung und wählen Sie im Kontextmenü den Befehl **Wertezuordnung einfügen**.

Daraufhin wird eine neue Wertezuordnungskomponente zum Mapping hinzugefügt. Sie können nun Datenelementpaare zur Lookup-Tabelle hinzufügen. Doppelklicken Sie dazu auf die Titelleiste der Komponente oder rechtsklicken Sie auf die Komponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften**



MapForce ersetzt die einzelnen Werte, die zum **Input** der Wertezuordnung gelangen zur Mapping-Laufzeit. Wenn die *linke Spalte* der Lookup-Tabelle einen übereinstimmenden Wert enthält, wird der Original-Input-Wert durch den Wert aus der *rechten Spalte* ersetzt. Andernfalls haben Sie die Möglichkeit, folgende Alternativen zu konfigurieren:

- einen Ersetzungswert. Im obigen Beispiel ist der Ersetzungswert der Text "incorrect date" (falsches Datum). Sie können auch festlegen, dass der Ersetzungswert leer ist, indem Sie keinen Text eingeben.
- den Original-Input-Wert: In diesem Fall wird der Original-Input-Wert unverändert an das Mapping übergeben, wenn die Lookup-Tabelle keine Entsprechung enthält.

Wenn Sie keine "Andernfalls"-Bedingung konfigurieren, wird jedes mal, wenn keine Entsprechung gefunden wird, ein **leerer Node** zurückgegeben. In diesem Fall wird nichts an die Zielkomponente übergeben und die Ausgabe enthält fehlende Felder. Um dies zu vermeiden, sollten Sie entweder die Andernfalls-Bedingung konfigurieren oder die Funktion **substitute-missing** 1103 verwenden.

Es gibt einen Unterschied zwischen der Definition eines leeren Ersetzungswerts und einer überhaupt fehlenden Definition der Andernfalls-Bedingung. Im ersten Fall wird das Feld in der Ausgabe generiert, hat aber einen leeren Wert. Im zweiten Fall wird das Feld (oder XML-Element), das den Wert enthält, gar nicht erstellt. Nähere Informationen dazu finden Sie unter Beispiel: Ersetzen von Stellenbezeichnungen 927.

© 2016-2022 Altova GmbH

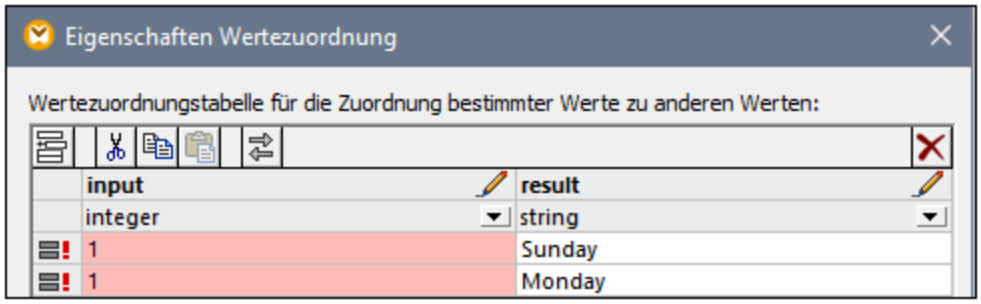
Befüllen einer Wertezuordnung

Sie können so viele Wertpaare wie nötig in einer Lookup-Tabelle definieren. Sie können die Werte manuell eingeben oder Tabellendaten aus Text-, CSV-, oder Excel-Tabellen hineinkopieren. In den meisten Fällen funktioniert es auch, wenn Sie Tabellen über einen gebräuchlichen Browser aus einer HTML-Seite kopieren und einfügen. Sie können Daten auch aus dem Datenbankraster im Fenster "DB-Abfrage" 388 einfügen. Wenn Sie Daten aus Textdateien kopieren, müssen die Felder durch Tabulatorzeichen voneinander getrennt sein. Außerdem erkennt MapForce in den meisten Fällen durch Kommas oder Semikola getrennten Text.

Beachten Sie bei der Erstellung von Lookup-Tabellen die folgenden Punkte:

1. Alle Datenelemente in der linken Spalte müssen eindeutig sein. Andernfalls ließe sich nicht ermitteln, für welches Datenelement genau eine Entsprechung benötigt wird.
2. Datenelemente, die zur selben Spalte gehören, müssen denselben Datentyp haben. Sie können den Datentyp aus der Dropdown-Liste am oberen Rand der jeweiligen Spalte der Lookup-Tabelle auswählen. Um Boolesche Typen zu konvertieren, geben Sie den Text "true" oder "false" wortwörtlich ein. Ein Beispiel dazu finden Sie unter Beispiel: Ersetzen von Wochentagen 925.

Wenn MapForce in der Lookup-Tabelle ungültige Daten findet, wird eine Fehlermeldung angezeigt und die ungültigen Zeilen werden rosa markiert, z.B:



So importieren Sie Daten aus einer externen Quelle in die Wertezuordnungskomponente:

1. Wählen Sie im Ausgangsprogramm (z.B. Excel) die entsprechenden Zellen aus. Dabei kann es sich entweder um eine einzelne Spalte oder um zwei benachbarte Spalten handeln.
2. Kopieren Sie die Daten mit dem Befehl **Kopieren** des externen Programms in die Zwischenablage.
3. Klicken Sie in der Wertezuordnungskomponente auf die Zeile, vor der die Daten eingefügt werden sollen.
4. Klicken Sie in der Wertezuordnungskomponente auf die Schaltfläche **Tabelle aus der Zwischenablage einfügen** oder drücken Sie alternativ dazu **Strg+V** oder **Umschalt+Einfg**.

Anmerkung: Die Schaltfläche **Tabelle aus der Zwischenablage einfügen** ist nur dann aktiv, wenn Sie vorher Daten aus einer Ausgangsdatei kopiert haben (d.h. wenn sich in der Zwischenablage Daten befinden).

Wenn Ihre Zwischenablage mehrere Spalten enthält, werden nur Daten aus den beiden ersten Spalten in die Lookup-Tabelle eingefügt, weitere Spalten werden ignoriert. Wenn Sie Daten aus einer einzigen Spalte über vorhandene Werte einfügen, erscheint ein Kontextmenü, in dem Sie gefragt werden, ob die Daten aus der Zwischenablage als neue Zeilen eingefügt werden sollen oder ob die vorhandenen Zeilen überschrieben werden

sollen. Stellen Sie daher sicher, dass die Zwischenablage nur eine Spalte und nicht mehrere enthält, wenn Sie vorhandene Werte in der Lookup-Tabelle überschreiben möchten, anstatt neue Zeilen einzufügen.

Um Zeilen manuell vor einer vorhandenen Zeile einzufügen, klicken Sie auf die betreffende Zeile und anschließend auf die Schaltfläche **Einfügen** .

Um eine vorhandene Zeile an eine andere Position zu verschieben, ziehen Sie die Zeile (noch oben oder nach unten) an die neue Position, während Sie die linke Maustaste gedrückt halten.

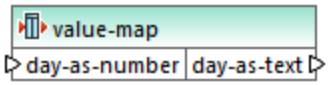
Um Zeilen zu kopieren oder auszuschneiden, um sie an einer anderen Position einzufügen, wählen Sie die Zeile zuerst aus und klicken Sie anschließend auf die Schaltfläche **Kopieren** (bzw. **Ausschneiden**). Sie können auch mehrere nicht unbedingt aufeinander folgende Zeilen kopieren oder ausschneiden. Halten Sie dazu beim Auswählen der Zeilen die **Strg**-Taste gedrückt. Beachten Sie, dass ausgeschnittener oder kopierter Text immer Werte aus beiden Spalten enthält. Sie können nicht Werte aus nur einer Spalte ausschneiden oder kopieren.

Um eine Zeile zu entfernen, klicken Sie darauf und anschließend auf die Schaltfläche **Löschen** .

Um die linke und die rechte Spalte zu vertauschen, klicken Sie auf die Schaltfläche **Spalten tauschen** .

Umbenennen von Wertezuordnungsparametern

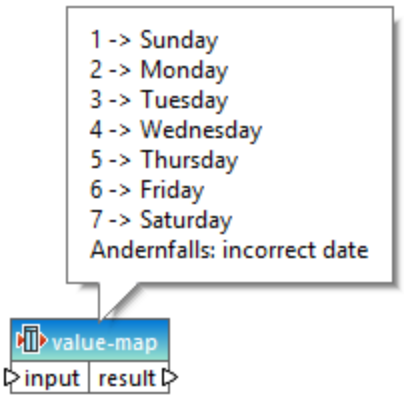
Standardmäßig hat der Input-Parameter einer Wertezuordnungskomponente den Namen "input" und der Output-Parameter den Namen "result". Um das Mapping übersichtlicher zu machen, haben Sie die Möglichkeit, jeden dieser Parameter durch Klicken auf die Schaltfläche **Bearbeiten** neben dem jeweiligen Namen umzubenennen. Im Folgenden sehen Sie ein Beispiel für eine Wertezuordnung mit benutzerdefinierten Parameternamen:



Vorschau auf eine Wertezuordnung

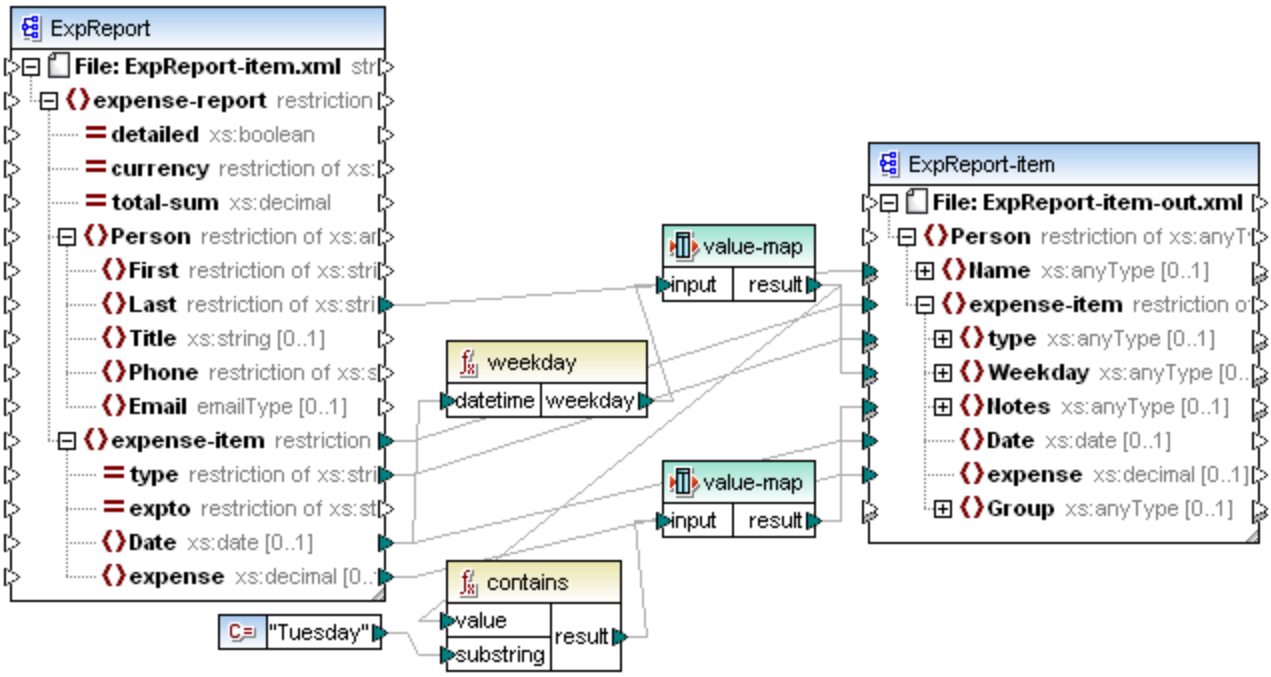
Nachdem Sie mit der Erstellung einer Wertezuordnung fertig sind, können Sie direkt über das Mapping schnell eine Vorschau darauf anzeigen, indem Sie die Maus über die Titelleiste der Komponente platzieren:

© 2016-2022 Altova GmbH



5.7.1 Beispiel: Ersetzen von Wochentagen

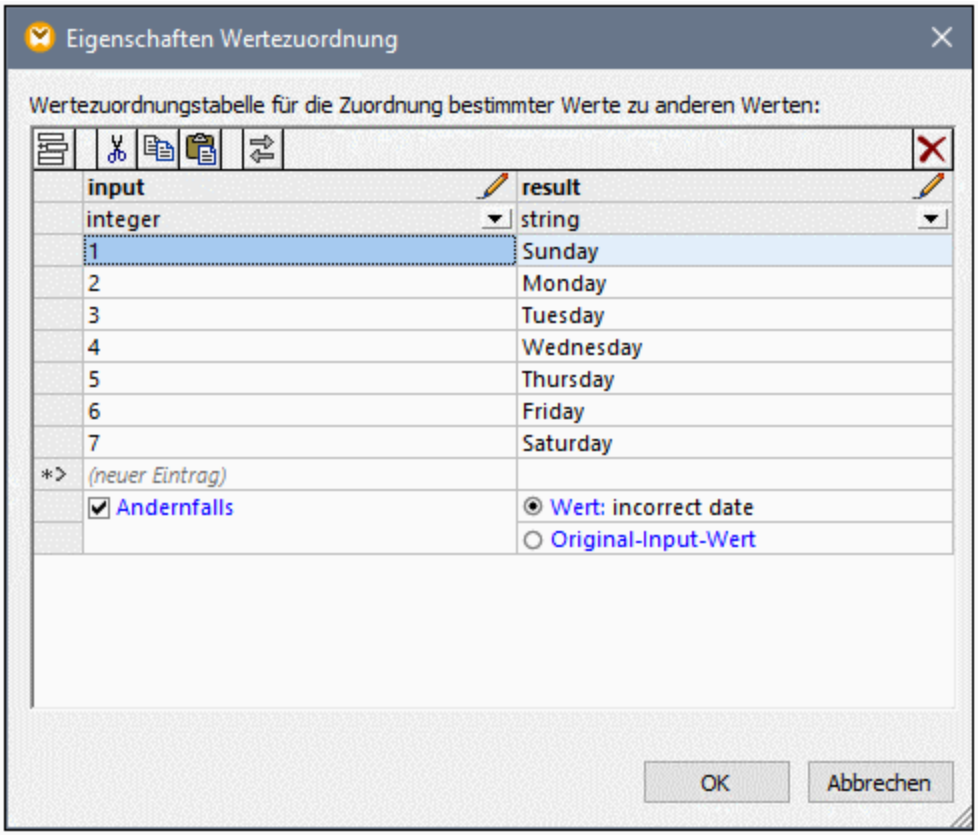
In diesem Beispiel sehen Sie eine Wertezuordnung, mit der Ganzzahlwerte durch Wochentagsnamen (1 = Sunday, 2 = Monday, usw.) ersetzt werden. Das Mapping zu diesem Beispiel befindet sich unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\Expense-valmap.mfd**.



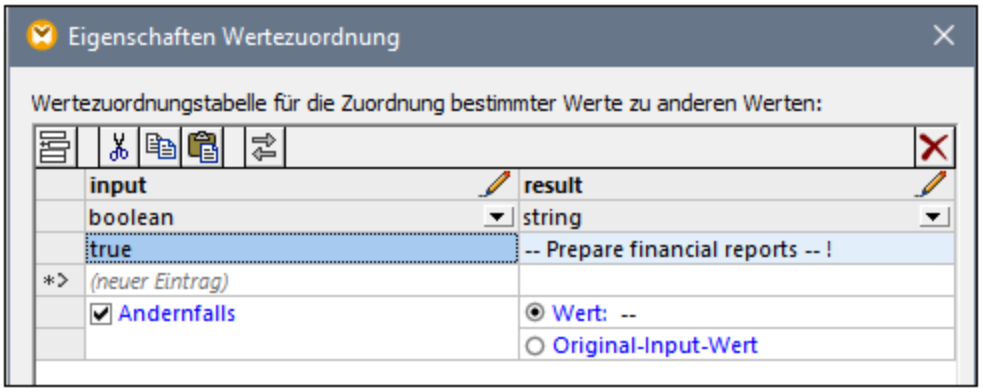
Expense-valmap.mfd

Der Wochentag wird aus dem Datenelement "**Date**" in der Datenquelle extrahiert, der numerische Wert wird in Text konvertiert und dieser Text wird in das Datenelement "**Weekday**" der Zielkomponente eingefügt. Dabei geschieht Folgendes:

- Die Funktion `weekday` extrahiert die Zahl für den Wochentag aus dem Datenelement **Date** in der Quelldatei. Das Ergebnis dieser Funktion sind Ganzzahlen von 1 bis 7.
- Die erste Wertezuordnungskomponente transformiert die Ganzzahlen in Wochentage (1 = Sunday, 2 = Monday, usw.). Wenn ein ungültiger Ganzzahlwert außerhalb des Bereichs von 1-7 gefunden wird, wird der Text "incorrect date" (falsches Datum) zurückgegeben.



- Wenn der Wochentag den Wert "Tuesday" enthält, so wird der Text "Prepare Financial Reports" in das Datenelement "**Notes**" in der Zielkomponente geschrieben. Dies erfolgt mit Hilfe der Funktion `contains`, die einen Booleschen Wert (**true** oder **false)** an eine zweite Wertezuordnungskomponente übergibt. Die zweite Wertezuordnung hat die folgende Konfiguration:



Die oben gezeigte Wertezuordnung ist folgendermaßen zu interpretieren:

© 2016-2022 Altova GmbH

- Immer, wenn der Boolesche Wert **true** ist, muss der Wert in den Text "-- Prepare financial reports -- ! ". konvertiert werden. In allen anderen Fällen wird der Text "--" zurückgegeben.

Beachten Sie, dass der Datentyp der ersten Spalte als "boolean" definiert ist. Damit wird sichergestellt, dass der Boolesche Input-Wert **true** als Boolescher Wert erkannt wird.

5.7.2 Beispiel: Ersetzen von Stellenbezeichnungen

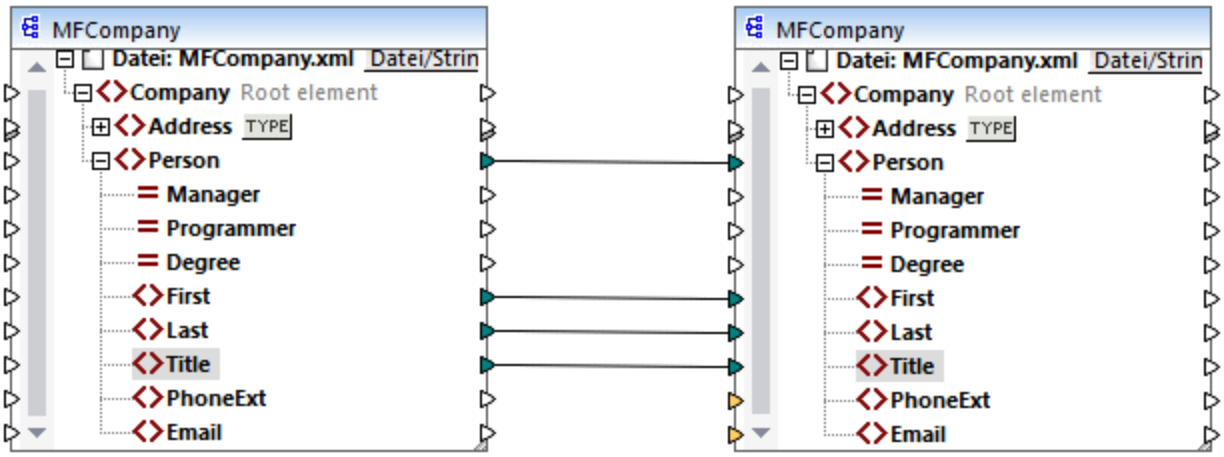
In diesem Beispiel wird gezeigt, wie Sie Werte von bestimmten Elementen in einer XML-Datei mit Hilfe von Wertezuordnungskomponenten (d.h. mit Hilfe einer vordefinierten Lookup-Tabelle) ersetzen.

Sie finden die XML-Datei zu diesem Beispiel unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\MFCompany.xml**. Unter anderem sind darin Informationen über die Angestellten einer Firma und deren Stellenbezeichnungen gespeichert, z.B.:

```
<Person>
   <First>Michelle</First>
   <Last>Butler</Last>
   <Title>Software Engineer</Title>
</Person>
<Person>
   <First>Lui</First>
   <Last>King</Last>
   <Title>Support Engineer</Title>
</Person>
<Person>
   <First>Steve</First>
   <Last>Meier</Last>
   <Title>Office Manager</Title>
</Person>
```

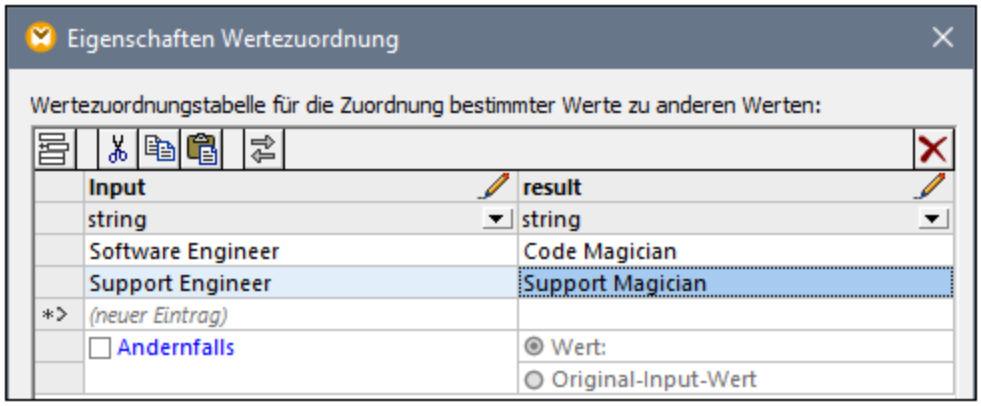
Angenommen, Sie müssen einige der Stellenbezeichnungen in der obigen XML-Datei ersetzen. So soll etwa der Titel "Software Engineer" durch "Code Magician" und der Titel "Support Engineer" durch "Support Magician" ersetzt werden. Alle anderen Stellenbezeichnungen sollen unverändert bleiben.

Fügen Sie für diese Aufgabe die XML-Datei durch Klicken auf die Symbolleisten-Schaltfläche **XML-Schema/Datei einfügen**  oder Auswahl des Menübefehls **Einfügen | XML-Schema/Datei** zum Mapping hinzu. Kopieren Sie als nächstes die XML-Komponente im Mapping, fügen Sie sie ein und erstellen Sie die Verbindungen wie unten gezeigt. Eventuell müssen Sie zuerst die Symbolleisten-Schaltfläche  **Automatische Verbindung von Sub-Einträge aktivieren/deaktivieren** deaktivieren, damit nicht benötigte Verbindungen nicht automatisch erstellt werden.



Im bisher erstellten Mapping werden die **Person**-Elemente einfach in die XML-Zieldatei kopiert, ohne dass an den Elementen **First**, **Last** und **Title** irgendwelche Änderungen vorgenommen werden.

Um die benötigten Stellenbezeichnungen zu ersetzen, fügen Sie eine Wertezuordnungskomponente hinzu. Klicken Sie mit der rechten Maustaste auf die Verbindung zwischen den beiden **Title**-Elementen und wählen Sie im Kontextmenü den Befehl **Wertezuordnung einfügen.** Konfigurieren Sie die Wertezuordnungseigenschaften, wie unten gezeigt:



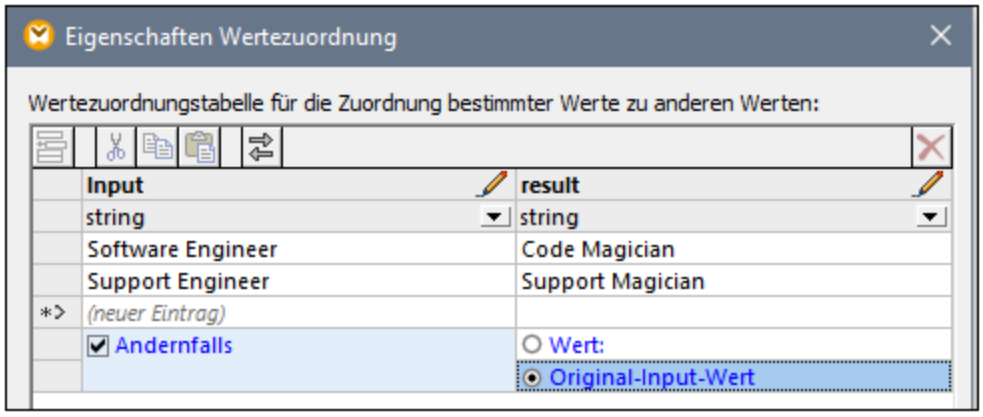
Gemäß der obigen Konfiguration wird jede Instanz von "Software Engineer" durch "Code Magician" und jede Instanz von "Support Engineer" durch "Support Magician" ersetzt. Beachten Sie, dass die **Andernfalls**-Bedingung noch nicht definiert wurde, daher gibt die Wertezuordnung überall dort, wo die Stellenbezeichnug eine andere als "Software Engineer" und "Support Engineer" ist, einen *leeren Node* zurück. Wenn Sie daher auf das Register Ausgabe klicken und eine Vorschau auf das Mapping anzeigen, fehlt bei einigen der Person-Elemente das Element **Title**, z.B.:

```
<Person>
   <First>Vernon</First>
   <Last>Callaby</Last>
</Person>
<Person>
```


© 2016-2022 Altova GmbH

```
   <First>Frank</First>
   <Last>Further</Last>
</Person>
<Person>
   <First>Michelle</First>
   <Last>Butler</Last>
   <Title>Code Magician</Title>
</Person>
```

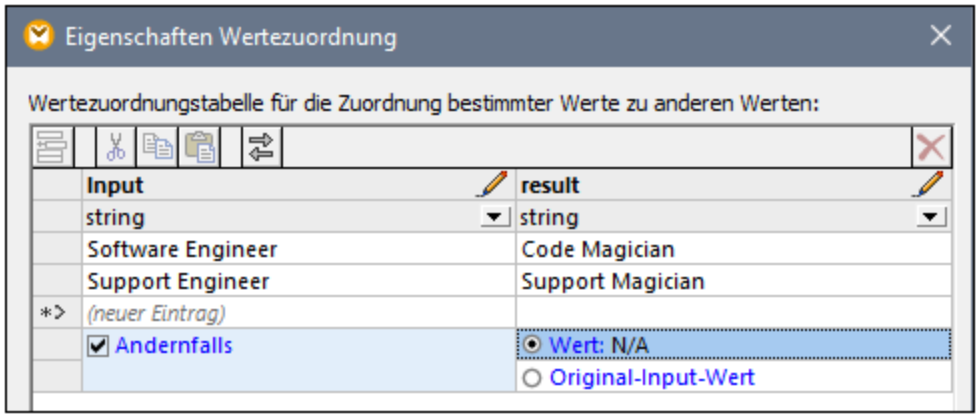
Wie zuvor erwähnt, verursachen fehlende Nodes fehlende Einträge in der generierten Ausgabe, daher wurde im obigen XML-Fragment nur bei Michelle Butler die Stellenbezeichnung (title) ersetzt, da ihre Stellenbezeichnung in der Lookup-Tabelle vorhanden war. Die bisher erstellte Konfiguration muss daher folgendermaßen ergänzt werden:



Mit der obigen Konfiguration geschieht zur Mapping-Laufzeit Folgendes:

- Jede einzelne Instanz von "Software Engineer" wird durch "Code Magician" ersetzt.
- Jede einzelne Instanz von "Support Engineer" wird durch "Support Magician" ersetzt.
- Wenn die ursprüngliche Stellenbezeichnung in der Lookup-Tabelle nicht gefunden wird, gibt die Wertezuordnung die Stellenbezeichnung unverändert zurück..

Zu Demonstrationszwecken können wir alle anderen Stellenbezeichnungen mit Ausnahme von "Software Engineer" und "Support Engineer", auch in einen benutzerdefinierten Wert, z.B. in "N/A" ändern. Definieren Sie dazu die Eigenschaften, wie unten gezeigt:



Wenn Sie jetzt eine Vorschau auf das Mapping anzeigen, sind alle Stellenbezeichnungen in der Ausgabe vorhanden, doch weisen diejenigen ohne Entsprechung in der Lookup-Tabelle den Wert "N/A" auf, z.B.:

```
<Person>
   <First>Vernon</First>
   <Last>Callaby</Last>
   <Title>N/A</Title>
</Person>
<Person>
   <First>Frank</First>
   <Last>Further</Last>
   <Title>N/A</Title>
</Person>
<Person>
   <First>Michelle</First>
   <Last>Butler</Last>
   <Title>Code Magician</Title>
</Person>
```

Damit ist das Wertezuordnungsbeispiel abgeschlossen. Durch Anwendung der oben gezeigten Logik können Sie das gewünschte Ergebnis nun auch in anderen Mappings erzielen.

© 2016-2022 Altova GmbH

5.8 Gruppierungsfunktionen

Wenn in Ihrem Mappings Nodes oder Zeilen gruppiert werden müssen, können Sie dies mit Hilfe der folgenden vordefinierten MapForce-Funktionen tun:

- `group-by`
- `group-adjacent`
- `group-into-blocks`
- `group-starting-with`
- `group-ending-with`

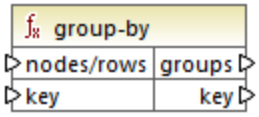
Um eine dieser Funktionen im Mapping zu verwenden, ziehen Sie sie aus dem Fenster "Bibliotheken" in das Mapping. Siehe auch Hinzufügen einer Funktion zu einem Mapping 943.

Anmerkung: Gruppierfunktionen stehen in den folgenden Sprachen nicht zur Verfügung: XSLT 2.0, XSLT 3.0, C++, C#, Java, Built-In.

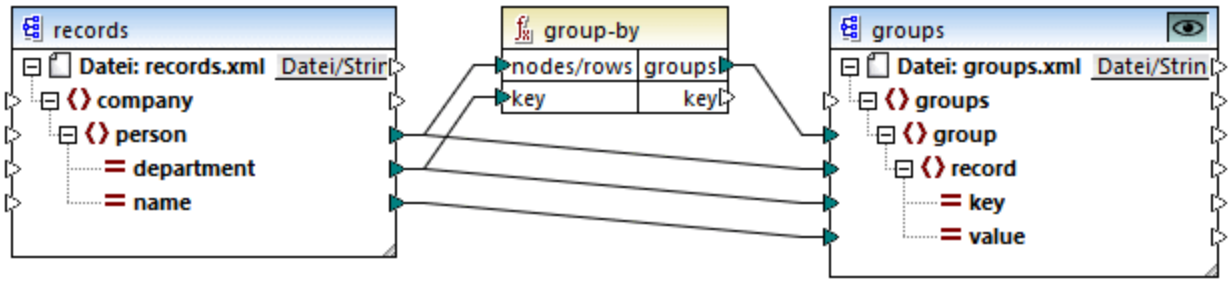
In den folgenden Abschnitten finden Sie typische Verwendungsbeispiele für die Gruppierung von Funktionen. Dazu gibt es das folgende Demo-Mapping: **<Dokumente>\Altova\MapForce2023\MapForceExamples\GroupingFunctions.mfd**. Beachten Sie, dass das Demo-Mapping mehrere Transformationen enthält, nämlich eine für jede Funktion. Da immer nur eine Ausgabevorschau auf einmal angezeigt werden kann, klicken Sie auf die **Vorschau** Schaltfläche für die gewünschte Transformation, bevor Sie auf das **Ausgaberegister** klicken.

group-by

Mit der Funktion `group-by` werden anhand eines von Ihnen definierten Gruppierungsschlüssels Datensatzstrukturen erstellt.



So ist etwa in der unten gezeigten abstrakten Transformation der Gruppierungsschlüssel "Department". Da es insgesamt drei eindeutige Abteilungen (Departments) gibt, würden bei Anwendung der Funktion group-by drei Gruppen erstellt:



Nähere Informationen dazu finden Sie unter der Funktion group-by 1085.

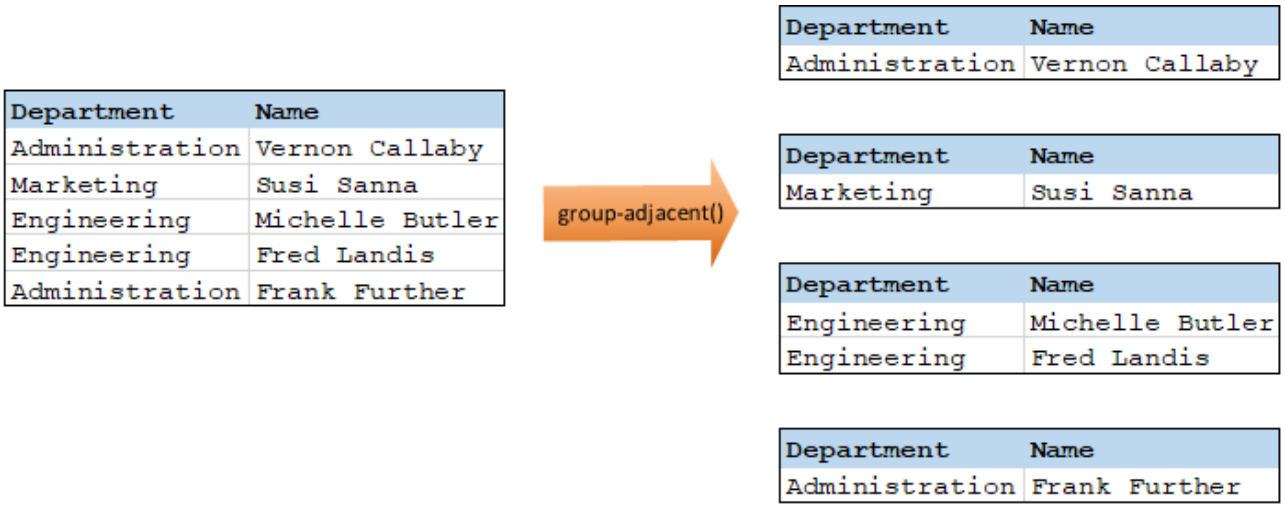
group-adjacent

Für die Funktion **`group-adjacent`** ist, ähnlich wie für die Funktion `group-by`, als Argument ein Gruppierungsschlüssel erforderlich. Im Gegensatz zur Funktion **`group-by`** wird mit dieser Funktion immer dann eine neue Gruppe erstellt, wenn der nächste Schlüssel ein anderer ist. Wenn zwei benachbarte Datensätze denselben Schlüssel haben, werden sie in dieselbe Gruppe eingeordnet.

So ist etwa in der unten gezeigten abstrakten Transformation der Gruppierungsschlüssel "Department". Auf der linken Seite des Diagramms sehen Sie die Input-Daten, während rechts die Ausgabedaten nach der Gruppierung angezeigt werden. Bei Ausführung der Transformation geschieht Folgendes:

- Anfangs wird anhand des ersten Schlüssels "Administration" eine neue Gruppe erstellt.
- Der nächste Schlüssel ist ein anderer, daher wird eine zweite Gruppe "Marketing" erstellt.
- Der dritte Schlüssel ist ebenfalls ein anderer, daher wird eine weitere Gruppe namens "Engineering" erstellt.
- Der vierte Schlüssel ist derselbe wie der dritte, daher wird dieser Datensatz in die bereits vorhandene Gruppe platziert.
- Der fünfte Schlüssel schließlich ist anders als der vierte, weshalb die letzte Gruppe erstellt wird.

Wie unten gezeigt, landen "Michelle Butler" und "Fred Landis" in derselben Gruppe, weil sie denselben Schlüssel aufweisen und nebeneinander liegen. "Vernon Callaby" und "Frank Further" hingegen befinden sich in separaten Gruppen, weil sie nicht benachbart sind, obwohl sie denselben Schlüssel aufweisen.

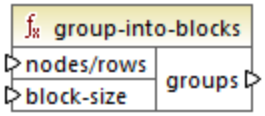


Nähere Informationen dazu finden Sie unter der Funktion group-adjacent[1083].

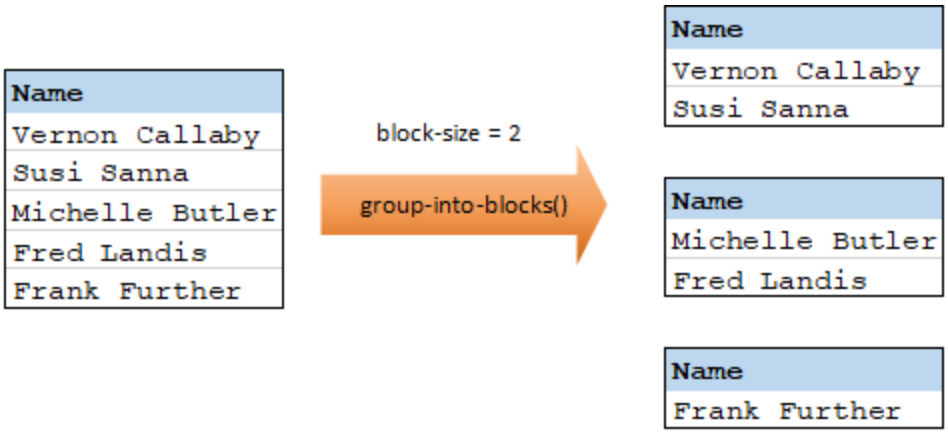
group-into-blocks

Mit der Funktion **`group-into-blocks`** werden gleiche Gruppen erstellt, die genau N Elemente enthalten, wobei N der an das Argument `block-size` (Blockgröße) gelieferte Wert ist. Beachten Sie, dass die letzte Gruppe, je nach Anzahl der Elemente in der Quellkomponente, N oder weniger Elemente enthalten kann.

© 2016-2022 Altova GmbH



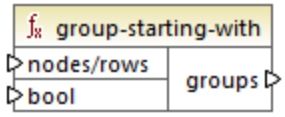
Im Beispiel unten ist `block-size` gleich 2. Da es insgesamt fünf Elemente gibt, enthält jede Gruppe mit Ausnahme der letzten genau zwei Elemente.



Nähere Informationen dazu finden Sie unter der Funktion group-into-blocks[1089].

group-starting-with

Die Funktion **`group-starting-with`** erhält eine Boolsche Bedingung als Argument. Wenn die Boolsche Bedingung "true" ergibt, wird ab dem Datensatz, auf den die Bedingung zutrifft, eine neue Gruppe erstellt.



Im Beispiel unten ist die Bedingung, dass "Key" (Schlüssel) gleich "heading" sein soll. Diese Bedingung trifft auf den ersten und vierten Datensatz zu, daher werden als Ergebnis zwei Gruppen erstellt:



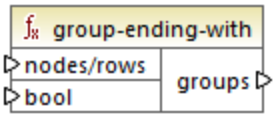
Anmerkung: Wenn vor dem ersten Datensatz, auf den die Bedingung zutrifft, weitere Datensätze vorhanden sind, wird eine zusätzliche Gruppe erstellt. Wenn vor dem ersten "heading"-Datensatz z.B. weitere

"line"-Datensätze vorhanden waren, würden diese in eine neue Gruppe platziert.

Nähere Informationen dazu finden Sie unter der Funktion group-starting-with[1091].

group-ending-with

Die Funktion **group-ending-with** erhält eine Boolsche Bedingung als Argument. Wenn die Boolsche Bedingung "true" ergibt, wird bis inklusive dem Datensatz, auf den die Bedingung zutrifft, eine neue Gruppe erstellt.



Im Beispiel unten ist die Bedingung, dass "Key" (Schlüssel) gleich "trailing" sein soll. Diese Bedingung trifft auf den dritten und fünften Datensatz zu, daher werden als Ergebnis zwei Gruppen erstellt:



Anmerkung: Wenn nach dem letzten Datensatz, auf den die Bedingung zutrifft, weitere Datensätze vorhanden sind, wird eine zusätzliche Gruppe erstellt. Wenn z.B. nach dem letzten "trailing"-Datensatz z.B. weitere "line"-Datensätze vorhanden wären, würden diese in eine neue Gruppe platziert.

Nähere Informationen dazu finden Sie unter der Funktion group-ending-with[1087].

5.8.1 Beispiel: Gruppieren von Datensätzen nach Schlüssel

In diesem Beispiel wird gezeigt, wie Sie Datensätze mit Hilfe der **group-by**-Funktion gruppieren. Außerdem wird darin gezeigt, wie Sie Daten aggregieren. Das Demo-Mapping zu diesem Beispiel finden Sie unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2023\MapForceExamples\GroupTemperaturesByYear.mfd. In diesem Mapping werden Daten aus einer XML-Datei ausgelesen, die ein Protokoll monatlicher Temperaturen enthält, siehe Codefragment unten:

```
<Temperatures>
   <data temp="-3.6" month="2006-01" />
   <data temp="-0.7" month="2006-02" />
   <data temp="7.5" month="2006-03" />
   <data temp="12.4" month="2006-04" />
```

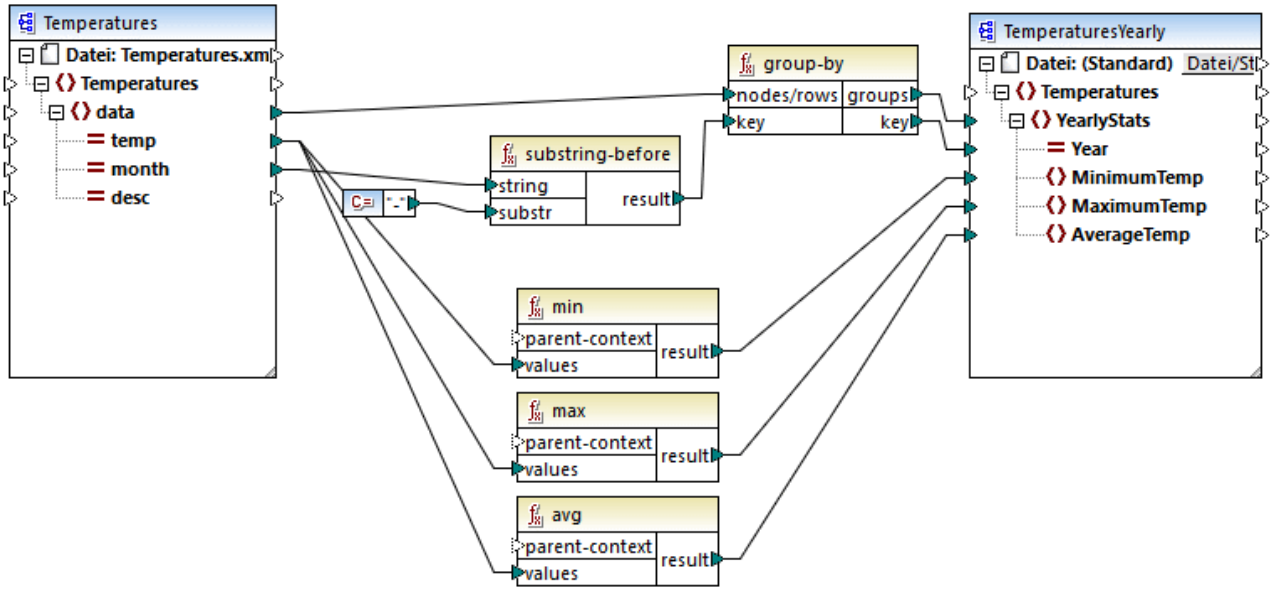

© 2016-2022 Altova GmbH

```
   <data temp="16.2" month="2006-05" />
   <data temp="19" month="2006-06" />
   <data temp="22.7" month="2006-07" />
   <data temp="23.2" month="2006-08" />
   <data temp="18.7" month="2006-09" />
   <data temp="11.2" month="2006-10" />
   <data temp="9.1" month="2006-11" />
   <data temp="0.8" month="2006-12" />
   <data temp="-3.2" month="2007-01" />
   <data temp="-0.3" month="2007-02" />
   <data temp="6.5" month="2007-03" />
   <data temp="10.6" month="2007-04" />
   <data temp="19" month="2007-05" />
   <data temp="20.3" month="2007-06" />
   <data temp="22.3" month="2007-07" />
   <data temp="20.7" month="2007-08" />
   <data temp="19.2" month="2007-09" />
   <data temp="12.9" month="2007-10" />
   <data temp="8.1" month="2007-11" />
   <data temp="1.9" month="2007-12" />
</Temperatures>
```

Dieses Mapping hat zwei Aufgaben:

1. Zusammengruppierung der Temperaturen jedes Jahres
2. Eruierung der jeweiligen Minima, Maxima und Durchschnittstemperaturen jedes Jahres

Für Aufgabe Nr. 1 wird die Funktion **group-by**[1085] im Mapping aufgerufen. Für Aufgabe Nr. 2 werden die Aggregierungsfunktionen **min**[1030], **max**[1029] und **avg**[1027] aufgerufen. Alle diese Funktionen sind vordefinierte MapForce-Funktionen, die Sie durch Ziehen aus dem Fenster "Bibliotheken" in jedes beliebige Mapping einfügen können, siehe Hinzufügen einer Funktion zum Mapping[943].



GroupTemperaturesByYear.mfd

Bei der Ausführung eines MapForce-Mappings (dies ist auch die empfohlene Methode, um ein Mapping zu lesen), wird mit dem obersten Datenelement der Zielkomponente begonnen. In diesem Beispiel wird für jede von der Funktion **group-by** zurückgegebene Gruppe ein **YearlySales**-Datenelement zurückgegeben. Die Funktion **group-by** nimmt als erstes Argument alle **data**-Datenelemente aus der Quellkomponente und gruppiert diese nach dem Input, der mit dem Input "**key**" verbunden ist. Da die Temperaturen nach Jahr gruppiert werden sollen, muss zuerst das Jahr ermittelt werden. Zu diesem Zweck extrahiert die Funktion **substring-before** das Jahr aus dem Attribut **month** der einzelnen **data**-Elemente. Dabei wird als Argument der Wert von **month** verwendet und es wird der Teil vor der ersten Instanz von **substr** zurückgegeben. Wie oben gezeigt, wird in diesem Beispiel für **substr** das Bindestrichzeichen definiert. Wenn die Funktion daher den Wert "2006-01" verarbeitet, ist das Ergebnis der Funktion "2006".

Die Werte von **MinimumTemp**, **MaximumTemp** und **AverageTemp** werden schließlich durch Verbinden dieser Datenelemente mit der jeweiligen Aggregationsfunktion **min**, **max** und **avg** ermittelt. Alle drei Funktionen erhalten als Input die aus der Quellkomponente ausgelesene Sequenz von Temperaturen. Für diese Funktionen wird kein **parent-context**-Argument benötigt, da sie bereits im Kontext der einzelnen Gruppen verwendet werden, d.h. es gibt eine übergeordnete Verbindung von **data** zu **YearlyStats,** die den Kontext für die einzelnen Aggrationsfunktionen liefert.

Um eine Vorschau auf die Mapping-Ausgabe zu sehen, klicken Sie auf das Register **Ausgabe**. Beachten Sie, dass die Anzahl der Gruppen der Anzahl der in der Quelldatei vorhandenen Jahre entspricht, z.B.:

```
<Temperatures>
   <YearlyStats Year="2006">
      <MinimumTemp>-3.6</MinimumTemp>
      <MaximumTemp>23.2</MaximumTemp>
      <AverageTemp>11.375</AverageTemp>
   </YearlyStats>
   <YearlyStats Year="2007">
      <MinimumTemp>-3.2</MinimumTemp>
```


© 2016-2022 Altova GmbH

```
      <MaximumTemp>22.3</MaximumTemp>
      <AverageTemp>11.5</AverageTemp>
   </YearlyStats>
</Temperatures>
```

Anmerkung: Die oben gezeigten Codefragmente enthalten aus Gründen der Einfachheit weniger Daten als die tatsächliche Input- und Output-Datei aus dem Demo-Mapping.

5.9 Ausnahmeereignisse

Ein Ausnahmeereignis ist eine spezielle Art von Komponente, über die Sie den Mapping-Vorgang abbrechen und eine Fehlermeldung zurückgeben lassen können, wenn eine von einem Filter zurückgegebene Bedingung erfüllt wird. Sie können ein Ausnahmeereignis hinzufügen, wenn Ihr Mapping einen Filter enthält, mit dem eine true/false-Bedingung überprüft wird (siehe Filter und Bedingungen 914). So können Sie z.B. festlegen, dass ein Ausnahmeereignis ausgelöst wird, wenn der Wert eines Datenelements aus dem Mapping einen bestimmten benutzerdefinierten Schwellenwert übersteigt.

Auch bei der Erstellung von WSDL-Webservices mit MapForce können Ausnahmeereignisse hinzugefügt werden (siehe Webservice Fehler 1270).

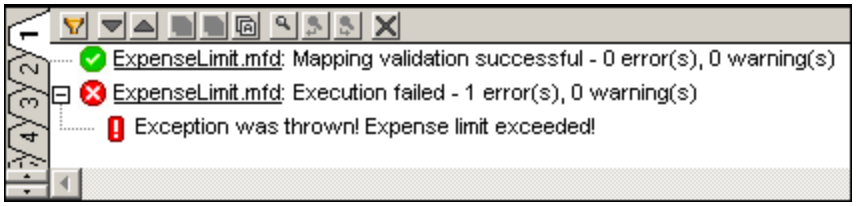
So fügen Sie ein Ausnahmeereignis zum Mapping hinzu:

1. Klicken Sie im Menü **Einfügen** auf **Ausnahme**.
2. Klicken Sie auf die Symbolleisten-Schaltfläche **Ausnahmeereignis einfügen** ( ).
3. Verbinden Sie den **throw**-Input der Ausnahme entweder mit einem **on-true** oder einem **on-false**-Output eines Filters.
4. Verbinden Sie optional den **error-text**-Input der Ausnahme mit einer anderen Komponente (normalerweise einer Konstante), die den Text für den Fehler liefert, wenn das Ausnahmeereignis ausgelöst wird.

Anmerkung: Es muss sowohl der **on-true** als auch der **on-false**-Output des Filters verbunden werden. Einer dieser Outputs muss direkt (ohne dazwischen geschaltete Funktionen oder Komponenten) mit der Ausnahme verbunden werden. Der andere Output muss entweder direkt oder über andere Zwischenkomponenten mit der Zielkomponente verbunden werden.

Wenn beim Mapping ein Ausnahmeereignis auftritt, werden Sie auf folgende Weise darüber informiert:

- Im Meldungsfenster von MapForce wird ein Fehler sowie der Text der Ausnahme angezeigt (in diesem Fall "Expense limit exceeded").



  Wenn die Mapping-Sprache XSLT 2.0 oder XQuery ist, wird im Fenster Meldungen ein Fehler "Ausführung fehlgeschlagen" angezeigt und das entsprechende XSLT2- bzw. XQuery-Register wird geöffnet. Die Zeile mit dem Fehler wird im Meldungsfenster markiert.

- Wenn Sie das Mapping mit MapForce Server ausführen, wird die Fehlermeldung "Es wurde eine Ausnahme ausgelöst" gefolgt von dem von Ihnen in MapForce definierten Text für die Ausnahme angezeigt.

```
c:\codegen\mfx\ExpenseLimit>"C:\Program Files (x86)\Altova\MapForceServer2016\bi
n\MapForceServer.exe" run ExpenseLimit.mfx
Exception was thrown! Expense limit exceeded!
```

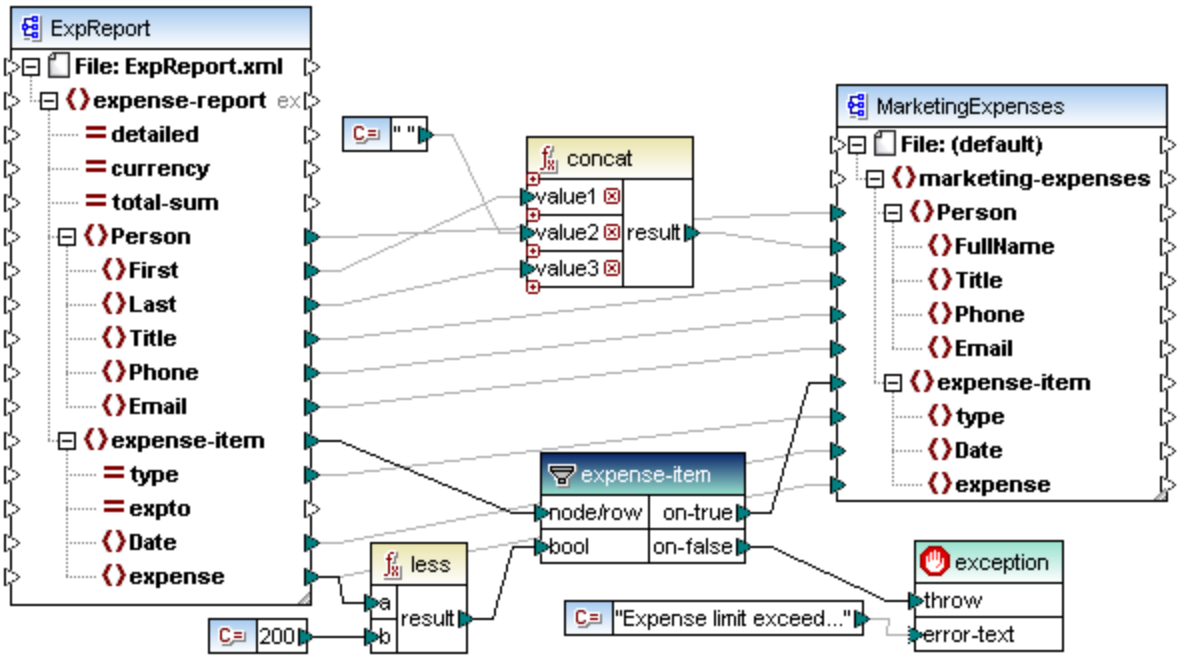
 © 2016-2022 Altova GmbH

- Wenn Sie das Mapping vom generierten C#-, C++- oder Java-Code ausführen, wird der Fehler "USER EXCEPTION" gefolgt von dem von Ihnen in MapForce definierten Text für die Ausnahme angezeigt.

```
C:\codegen\cs\ExpenseLimit\Mapping\bin\Debug>Mapping.exe
Mapping Application
USER EXCEPTION: Expense limit exceeded!
```

5.9.1 Beispiel: Ausnahme bei "größer als"-Bedingung

In diesem Beispiel wird ein Mapping beschrieben, das ein Ausnahmeereignis auslöst, wenn eine "größer als"-Bedingung erfüllt wird. Sie finden das Beispielmapping hierzu unter: **<Dokumente>\Altova\MapForce2023\MapForceExamples\ExpenseLimit.mfd**.



Dieses Mapping löst immer dann eine Ausnahme aus, wenn das Datenelement **expense** in der XML-Instanzquelldatei einen Wert hat, der größer als 200 ist. Der Wert "200" wird von einer Konstanten bereitgestellt. Anschließend werden die beiden Werte mit Hilfe der **less**-Funktion miteinander verglichen. Wenn der Wert von **expense** kleiner als 200 ist, so wird das übergeordnete Datenelement, nämlich **expense-item**, an den Filter übergeben und es wird keine Ausnahme ausgelöst. Andernfalls wird eine Ausnahme mit dem benutzerdefinierten Text "Expense limit exceed" ausgelöst.

Die mit dem Symbol gekennzeichnete Ausnahme besteht, wie oben gezeigt, aus zwei Teilen: **throw** und **error-text**. Der **throw**-Teil muss mit dem **on-false** oder **on-true**-Output eines Filters verbunden werden. Der **error-text**-Teil wird mit einer Konstante verbunden, die den benutzerdefinierten Text der Ausnahme bereitstellt.

Es ist wichtig, dass beide Outputs des Filters verbunden werden, da sonst keine Ausnahme ausgelöst wird. In diesem Beispiel ist der **on-false**-Output mit der Ausnahme verbunden, während der **on-true**-Output mit der Zielkomponente verbunden ist.

5.9.2 Beispiel: Ausnahme, wenn Node nicht vorhanden ist

In diesem Beispiel wird gezeigt, wie eine Ausnahme ausgelöst wird, wenn ein Node im XML-Quellschema nicht vorhanden ist. Aus Gründen der Einfachheit wird in diesem Beispiel dasselbe XML-Schema sowohl als Quell- als auch Zielkomponente verwendet.

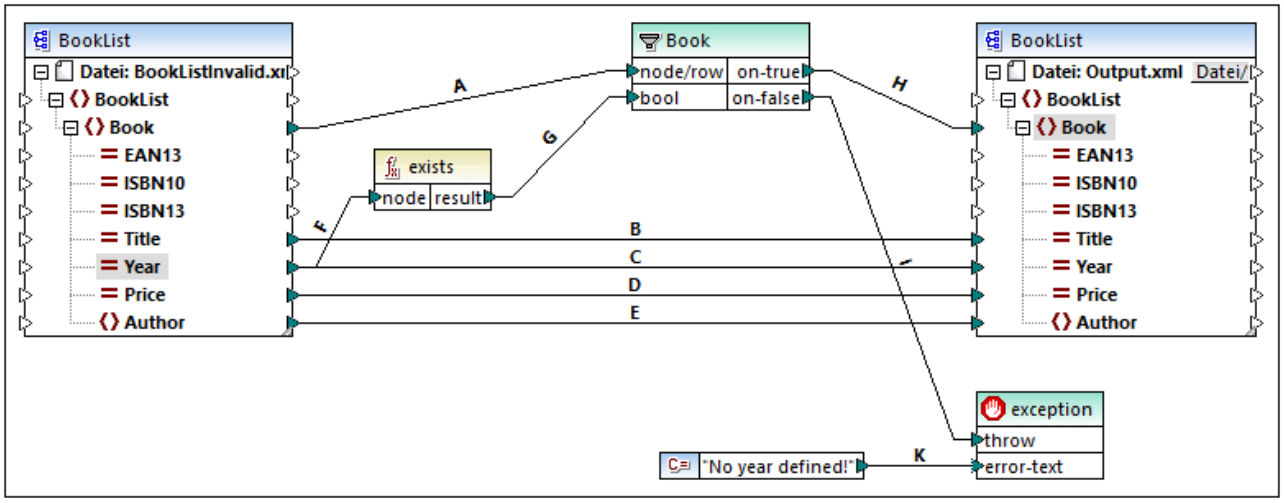
So fügen Sie das Quellschema zum Mapping hinzu:

1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zum Schema **<Dokumente>\Altova\MapForce2023\MapForceExamples\BookList.xsd**.
2. Wenn Sie aufgefordert werden, eine Instanzdatei anzugeben, klicken Sie auf **Überspringen**.
3. Wenn Sie aufgefordert werden, ein Schema-Root-Element auszuwählen, wählen Sie **BookList** als Root-Element aus.

Gehen Sie auf die gleiche Weise vor, um das Zielschema hinzuzufügen. Fügen Sie anschließend mit den entsprechenden Befehlen aus dem Menü **Einfügen** (bzw. über die entsprechenden Symbolleisten-Schaltflächen) die folgenden Komponenten hinzu:

- eine **Filter: Nodes/Zeilen**-Komponente (siehe auch Filter und Bedingungen 914)
- eine Konstante mit dem Text "No year defined!"
- eine Ausnahme

Ziehen Sie schließlich die Funktion **exists** aus dem Fenster "**Bibliotheken**" in den Mapping-Bereich und ziehen Sie die Verbindungen, wie unten gezeigt.



Gemäß dem XML-Schema sind alle Attribute des Elements **Book** mit Ausnahme des Buchtitels optional, d.h. das Attribut "Year" muss in einer gültigen XML-Instanz nicht unbedingt vorhanden sein. Ziel des Mappings ist, eine XML-Instanz, in der das Attribut "Year" für jedes Buch vorhanden ist, erfolgreich zu verarbeiten. In anderen Fällen soll das Mapping eine Ausnahme auslösen.

© 2016-2022 Altova GmbH

So testen Sie, ob das Mapping erfolgreich ausgeführt werden kann:

1. Doppelklicken Sie auf die Überschrift der Quellkomponente und navigieren Sie neben dem Eintrag **XML-Input-Datei** zur folgenden Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\BookList.xml**.
2. Klicken Sie auf die Schaltfläche **Ausgabe,** um das Mapping auszuführen.

So testen Sie die Ausnahme:

1. Erstellen Sie im selben Verzeichnis eine Kopie der Datei **BookList.xml** mit dem Namen **BookListInvalid.xml**.
2. Bearbeiten Sie die Datei, indem Sie das Year-Attribut aus einem Element entfernen.
3. Doppelklicken Sie auf die Überschrift der Quellkomponente und navigieren Sie neben dem Eintrag **XML-Input-Datei** zur Datei **BookListInvalid.xml**.
4. Klicken Sie auf die Schaltfläche **Ausgabe,** um das Mapping auszuführen.

Sehen wir uns nun die Funktionsweise des Mappings genauer an.

Mit der Verbindung **A** wird sichergestellt, dass für jedes Buch in der Quellinstanz ein Buch in der Zielinstanz erstellt wird. Mit den Verbindungen **B, C, D** und **E** wird sichergestellt, dass "Title", "Year", "Price" und "Author" für jedes Buch von der Quelldatei in die Zieldatei kopiert wird.

Durch die Verbindung **F** wird die `exists` Funktion ausgelöst, um zu überprüfen, ob das Attribut "Year" vorhanden ist. Mit der Verbindung **G** wird das Ergebnis der Funktion (`true` oder `false`) an den Filter übergeben. Wenn das Ergebnis `true` ist, ist das Attribut "Year" vorhanden und das Buch wird an den Filter und in der Folge über die Verbindung **H** an die Zielkomponente übergeben.

Beachten Sie, dass der Filter nicht direkt mit dem **Year-**Output der Quellkomponente verbunden wurde. Hätten wir die Verbindung direkt hergestellt, würde der Filter **Year** nach dem Vorhandensein von **Year** filtern, was nicht sinnvoll wäre. Die Ausnahme würde dabei nie ausgelöst.

Die Verbindung **I** wurde erstellt, weil die Ausnahme entweder mit einem **on-false** oder **on-true**-Output eines Filters verbunden werden muss. Die Verbindung **K** übergibt schließlich den benutzerdefinierten Fehlertext von der Konstanten an die Ausnahmekomponente.

6 Funktionen

Sie können Daten in MapForce mit Hilfe der folgenden Funktionskategorien gemäß Ihren Anforderungen transformieren:

- **Vordefinierte MapForce-Funktionen** - diese Funktionen wurden in MapForce vordefiniert und Sie können damit in Ihren Mappings die verschiedensten Verarbeitungsaufgaben im Zusammenhang mit Strings, Datumswerten und anderen Datentypen durchführen. Des Weiteren können damit Gruppierungen, Aggregationen, automatische Nummerierung und verschiedene andere Aufgaben durchgeführt werden. Informationen zu den einzelnen vordefinierten Funktionen finden Sie unter Referenz Funktionsbibliothek[1024].
- **Node-Funktionen und Standardwerte** - hierbei handelt es sich um speziellere Funktionen, mit denen Sie eine benutzerdefinierte Verarbeitungslogik erstellen und an einem oder mehreren untergeordneten Nodes einer Mapping-Komponente anwenden können. Sie können Daten damit entweder *vor* dem Erreichen eines Node einer Mapping-Struktur oder *unmittelbar nach* dem Verlassen des Node verarbeiten. Nähere Informationen dazu finden Sie unter Standardwerte und Node-Funktionen[951].
- **Benutzerdefinierte Funktionen (UDFs = user-defined functions)** - es handelt sich hierbei um MapForce-Funktionen, die Sie selbst anhand der nativen Komponentenarten und der bereits in MapForce verfügbaren vordefinierten Funktionen erstellen können, siehe Benutzerdefinierte Funktionen[967].
- **Benutzerdefinierte (angepasste) Funktionen** - es handelt sich hierbei um Funktionen, die Sie aus externen Quellen wie XSLT-Bibliotheken, XQuery-Bibliotheksmodulen, Java .class-Dateien, .NET .dll-Dateien importieren und an MapForce anpassen können. Damit diese Funktionen in MapForce wiederverwendet werden können, muss der Rückgabetyp dieser angepassten Funktionen ein simple type (wie String oder Ganzzahl) sein und auch die Parameter dieser Funktionen müssen den Typ simple type haben. Nähere Informationen dazu finden Sie unter Importieren benutzerdefinierter XSLT-Funktionen[984],Importieren benutzerdefinierter XQuery 1.0-Funktionen[992] und Importieren benutzerdefinierter Java- und .NET-Bibliotheken[997].

Anmerkung: Sie können externe benutzerdefinierte Funktionsbibliotheken entweder direkt (keine Konfiguration erforderlich) oder durch Konfiguration einer von MapForce erkannten MFF-Datei (MapForce Function File) importieren. Bei der zweitgenannten Methode können Sie zusätzlich zu Java-Klassen und .NET Assemblies auch C++-Bibliotheken importieren. Beachten Sie, dass mittels .mff-Dateien importierte Bibliotheken den unter Manuelles Referenzieren von Java, C# und C++-Bibliotheken[1005] erwähnten Vorgaben entsprechen müssen.

© 2016-2022 Altova GmbH

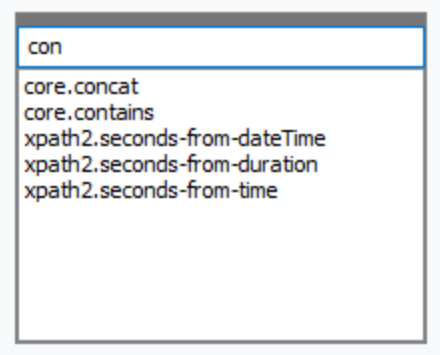
6.1 Grundlegendes zu Funktionen

Die folgenden Unterabschnitte enthalten eine Übersicht über grundlegende Aktionen im Zusammenhang mit Funktionen. Welche Funktionen im Fenster **Bibliotheken** angezeigt werden, hängt von der ausgewählten Transformationssprache ab. Nähere Informationen dazu finden Sie unter Transformationssprachen 28.

Hinzufügen einer Funktion

MapForce enthält eine große Zahl vordefinierter Funktionen, die Sie zum Mapping hinzufügen können. Nähere Informationen zu den einzelnen vordefinierten Funktionen finden Sie unter Referenz Funktionsbibliothek 1024. Wählen Sie eine der folgenden Methoden, um eine Funktion zu einem Mapping hinzuzufügen:

- Klicken Sie im Fenster **Bibliotheken** auf die benötigte Funktion und ziehen Sie sie in den Mapping-Bereich. Um Funktionen nach Namen zu filtern, beginnen Sie mit der Eingabe des Funktionsnamens in das Textfeld im unteren Bereich des Fensters.
- Doppelklicken Sie in den leeren Bereich des Mappings und beginnen Sie mit der Eingabe des Funktionsnamens (*siehe Abbildung unten*). Um einen Tooltip mit näheren Informationen zu einer Funktion zu sehen, wählen Sie die Funktion in der Liste aus. Um eine Funktion zu Ihrem Mapping hinzuzufügen, doppelklicken Sie auf die entsprechende Funktion in der Auswahlliste.



Hinzufügen einer benutzerdefinierten Funktion

Sie können benutzerdefinierte Funktionen (UDFs) auf dieselbe Art, wie oben beschrieben, zum Mapping hinzufügen, vorausgesetzt (i) die UDF wurde bereits im selben Mapping erstellt oder (ii) Sie haben ein Mapping importiert, das UDFs als lokale oder globale Bibliothek enthält.

Hinzufügen einer Konstante

Mit Hilfe von Konstanten können Sie benutzerdefinierten Text und Zahlen zu einem Mapping hinzufügen. Wählen Sie eine der folgenden Optionen, um eine Konstante zu einem Mapping hinzuzufügen:

- Klicken Sie mit der rechten Maustaste in den leeren Mapping-Bereich und wählen Sie im Kontextmenü **Konstante einfügen**. Geben Sie den Wert ein und wählen Sie einen Datentyp aus: *String*, *Zahl* oder *alle anderen*.
- Klicken Sie auf dem Menübefehl **Einfügen | Konstante**. Geben Sie den Wert ein und wählen Sie einen Datentyp aus: *String*, *Zahl* oder *alle anderen*.
- Klicken Sie auf die Symbolleisten-Schaltfläche **Konstante**. Geben Sie den Wert ein und wählen Sie einen Datentyp aus: *String*, *Zahl* oder *alle anderen*.
- Doppelklicken Sie in den leeren Bereich des Mappings. Geben Sie ein doppeltes Anführungszeichen, gefolgt vom Wert der Konstante ein. Das schließende Anführungszeichen ist optional. Um eine numerische Konstante einzufügen, geben Sie einfach die Zahl ein.

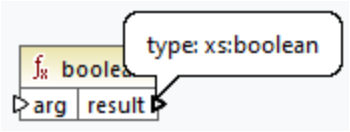
Suchen nach einer Funktion

Um im Fenster **Bibliotheken** nach einer Funktion zu suchen, geben Sie die ersten Zeichen des Funktionsnamens in das Textfeld im unteren Bereich des Fensters ein. Standardmäßig sucht MapForce nach Funktionsnamen und Beschreibungstext. Wenn Sie die Funktionsbeschreibung bei der Suche exkludieren möchten, klicken Sie auf den Nach unten-Pfeil und deaktivieren Sie die Option *In Funktionsbeschreibungen suchen*. Um die Suche abzubrechen, drücken Sie die **Esc**-Taste oder klicken Sie auf ✖.

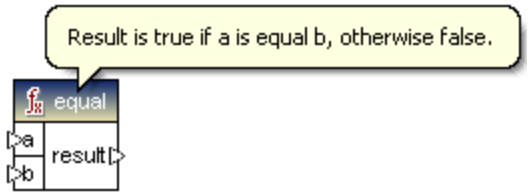
Um nach allen Instanzen einer Funktion im gerade aktiven Mapping zu suchen, klicken Sie im Fenster **Bibliotheken** mit der rechten Maustaste auf den Funktionsnamen und wählen Sie im Kontextmenü den Befehl **Alle Aufrufe suchen**. Die Suchergebnisse werden im Fenster **Meldungen** angezeigt.

Anzeigen von Typ und Beschreibung einer Funktion

Um den Datentyp eines Funktions-Input- oder -Output-Arguments zu sehen, platzieren Sie den Mauszeiger über den Argumentbereich einer Funktion (*siehe Abbildung unten*). Stellen Sie sicher, dass die Symbolleisten-Schaltfläche (**Tipps anzeigen**) aktiviert ist.



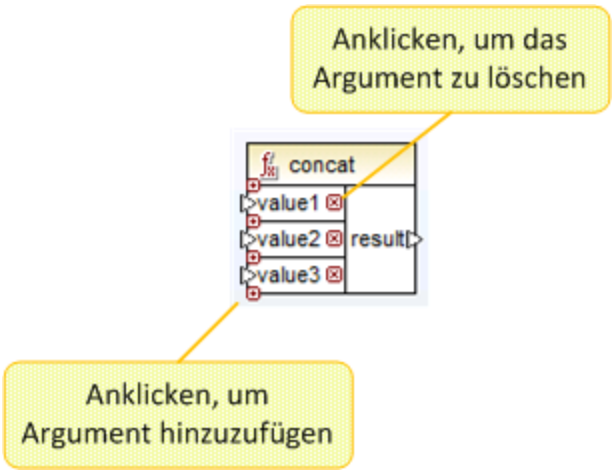
Um die Beschreibung einer Funktion anzuzeigen, platzieren Sie den Mauszeiger über die Titelleiste der Funktion (*siehe Abbildung unten*). Stellen Sie sicher, dass die Symbolleisten-Schaltfläche (**Tipps anzeigen**) aktiviert ist.



Hinzufügen/Löschen von Funktionsargumenten

Einige vordefinierte MapForce-Funktionen können erweitert werden, d.h. Sie können so viele Parameter, wie Sie für Ihre Mapping-Zwecke benötigen, hinzufügen. Ein gutes Beispiel dafür ist die Funktion **concat** 1105. Um (bei Funktionen, die dies unterstützen) Funktionsargumente hinzuzufügen oder zu löschen, klicken Sie neben dem entsprechenden Parameter auf **Parameter hinzufügen** (⊡) bzw. **Parameter löschen** (⊠) (*siehe unten*). Wenn Sie eine Verbindung auf das Symbol ⊡ ziehen, wird ein weiterer Parameter hinzugefügt und verbunden.

© 2016-2022 Altova GmbH


Anklicken, um das Argument zu löschen
concat
value1
value2
result
value3
Anklicken, um Argument hinzuzufügen

6.2 Verwalten von Funktionsbibliotheken

Sie können in MapForce die folgenden Arten von Bibliotheken in ein Mapping importieren und darin verwenden:

- Alle Mapping-Design-Dateien (*.mfd), die benutzerdefinierte Funktionen (UDFs) enthalten. Dies bezieht sich speziell auf Mapping-Dateien, die mit MapForce anhand von vordefinierten MapForce-Funktionen und Komponentenbausteinen erstellte benutzerdefinierte Funktionen (UDFs) enthalten. Nähere Informationen dazu finden Sie unter Benutzerdefinierte Funktionen 968.
- Benutzerdefinierte XSLT-Dateien, die Funktionen enthalten. Dies bezieht sich auf außerhalb von MapForce geschriebene XSLT-Funktionen, die sich wie unter Importieren benutzerdefinierter XSLT-Funktionen 984 beschrieben, für den Import in MapForce eignen.
- Benutzerdefinierte XQuery 1.0-Dateien, die Funktionen enthalten. Dies bezieht sich auf außerhalb von MapForce geschriebene XQuery-Funktionen, die sich wie unter Importieren benutzerdefinierter XQuery 1.0-Funktionen 992 beschrieben, für den Import in MapForce eignen.
- Java-Klassendateien und NET-Bibliotheken, die sich, wie unter Importieren benutzerdefinierter Java- und NET-Bibliotheken 997 beschrieben, für den Import in MapForce eignen.

Anmerkung: Sie können externe benutzerdefinierte Funktionsbibliotheken entweder direkt (keine Konfiguration erforderlich) oder durch Konfiguration einer von MapForce erkannten MFF-Datei (MapForce Function File) importieren. Bei der zweitgenannten Methode können Sie zusätzlich zu Java-Klassen und .NET Assemblies auch C++-Bibliotheken importieren. Beachten Sie, dass mittels .mff-Dateien importierte Bibliotheken den unter Manuelles Referenzieren von Java, C# und C++-Bibliotheken 1005 erwähnten Vorgaben entsprechen müssen.

Fenster "Bibliotheken verwalten"

Alle von einer Mapping-Datei verwendeten Bibliotheken können im Fenster "Bibliotheken verwalten" angezeigt und verwaltet werden. Dazu gehören auch benutzerdefinierte Funktionen (UDFs) und benutzerdefinierte Bibliotheken.

Standardmäßig wird das Fenster **Bibliotheken verwalten** nicht angezeigt. Um es anzuzeigen, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Ansicht** auf **Bibliotheken verwalten**.
- Klicken Sie im unteren Bereich des Fensters **Bibliotheken** auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**.

© 2016-2022 Altova GmbH



Sie können auswählen, ob benutzerdefinierte Funktionen (UDFs) und Bibliotheken nur für das gerade aktive Mapping-Dokument oder für alle geöffneten Mapping-Dokumente angezeigt werden sollen. Um die importierten Funktionen und Bibliotheken für alle gerade offenen Mapping-Dokumente anzuzeigen, klicken Sie mit der rechten Maustaste in das Fenster und wählen Sie im Kontextmenü den Befehl **Offene Dokumente anzeigen**.

Um anstelle des Namens den Pfad des geöffneten Mapping-Dokuments anzuzeigen, klicken Sie mit der rechten Maustaste in das Fenster und wählen Sie im Kontextmenü den Befehl **Dateipfade anzeigen**.

Die im Fenster "Bibliotheken verwalten" angezeigten Daten sind folgendermaßen hierarchisch gegliedert:

- Alle gerade offenen Mapping-Dokumente werden auf oberster Ebene angezeigt. Jeder Eintrag hat zwei Verzweigungen: **Benutzerdefinierte Funktionen** und **Eigene Bibliotheksimporte**.
  - Unter **Benutzerdefinierte Funktionen** werden alle in diesem Dokument enthaltenen UDFs angezeigt.
  - Unter **Eigene Bibliotheksdateien** werden *lokal* in das aktuelle Mapping-Dokument importierte Bibliotheken angezeigt. Mit dem Begriff "Bibliotheken" sind andere Mapping-Dokumente (.mfd-Dateien, die benutzerdefinierte Funktionen enthalten) oder externe in XSLT 1.0, XSLT 2.0, XQuery 1.0*, Java*, C#* geschriebene benutzerdefinierte Bibliotheken oder zuvor erwähnte .mff-Dateien gemeint. Beachten Sie, dass die Struktur **Eigene Bibliotheksimporte** mehrere Ebenen tief sein könnte, da in ein Mapping-Dokument wiederum weitere Mapping-Dokumente als Bibliothek importiert sein können.
- Der Eintrag **Globale Bibliotheksimporte** umfasst alle *global* auf Applikationsebene importierten benutzerdefinierten Bibliotheken. Auch im Fall von .mfd-Dateien könnte die Struktur aus den oben genannten Gründen mehrere Ebenen tief sein.

** Diese Sprachen werden nur in der MapForce Professional oder Enterprise Edition unterstützt.*

Anmerkung: Die XSLT-, XQuery-, C#- und Java-Bibliotheken können eigene Abhängigkeiten aufweisen. Solche Abhängigkeiten werden im Fenster "Bibliotheken" nicht angezeigt.

Kontextmenübefehle

Durch Rechtsklick auf ein Objekt und Auswahl einer der folgenden Kontextmenüoptionen können Sie verschiedene Operationen an Objekten im Fenster "Bibliotheken" ausführen.

Befehl	Beschreibung	Anwendbar auf
Öffnen	Öffnet das Mapping.	Mappings
Hinzufügen	Öffnet ein Dialogfeld, in dem Sie zu einer Bibliothek mit benutzerdefinierten Funktionen navigieren können.	Eigene Bibliotheksimporte
Sucht die Funktion im Fenster "Bibliotheken"	Ändert den Fokus in das Fenster "Bibliotheken" und wählt die Funktion aus.	Funktionen
Ausschneiden, Kopieren, Löschen	Diese Windows-Standardbefehle können nur auf benutzerdefinierte MapForce-Funktionen angewendet werden. Funktionen aus externen XSLT-Dateien oder anderen Bibliotheksarten können nicht kopiert und eingefügt werden.	Benutzerdefinierte Funktionen
Einfügen	Damit kann eine zuvor in die Zwischenablage kopierte benutzerdefinierte Funktion in die aktuelle Bibliothek eingefügt werden.	Bibliotheken (UDF)
Optionen	Öffnet ein Dialogfeld, in dem Sie Optionen für die aktuelle Bibliothek definieren oder ändern können.	Bibliotheken
Alle offenen Dokumente anzeigen	Wenn diese Option aktiviert ist, werden im Fenster "Bibliotheken verwalten" alle derzeit geöffneten Mappings angezeigt. Dies ist normalerweise dann nützlich, wenn Sie Funktionen zwischen Mappings kopieren und einfügen möchten. Andernfalls wird nur das Mapping angezeigt, auf dem sich gerade der Fokus befindet.	Immer
Dateipfade anzeigen	Wenn diese Option aktiviert ist, werden die Objekte im Fenster "Bibliotheken verwalten" mit ihrem vollständigen Dateipfad angezeigt. Andernfalls wird nur der Objektname angezeigt.	Immer

6.2.1 Lokale und globale Bibliotheken

Sie können Bibliotheken *lokal* oder *global* importieren. Globale Importe erfolgen auf Applikationsebene. Wenn eine Bibliothek global importiert wurde, können Sie ihre Funktionen von jedem Mapping aus verwenden.

Lokale Importe erfolgen auf Dateiebene. Angenommen, Sie entschließen sich bei der Arbeit am Mapping **A.mfd** dazu, alle benutzerdefinierten Funktionen aus dem Mapping **B.mfd** zu importieren. In diesem Fall gilt das Mapping **B.mfd** als lokale in das Mapping **A.mfd** importierte Bibliothek und Sie können Funktionen aus **B.mfd** auch in **A.mfd** verwenden. Auch wenn Sie Funktionen aus einer XSLT-Datei in **A.mfd** importieren, ist dies ein lokaler Import.

Alle lokalen und globalen Importe können im Fenster "Bibliotheken verwalten" angezeigt und verwaltet werden. Um eine Bibliothek zu importieren, wählen Sie eine der folgenden Methoden:

© 2016-2022 Altova GmbH

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken[34] auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).



2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

Miteinander in Konflikt stehende Funktionsnamen

Manchmal kommt es vor, dass derselbe Funktionsname auf jeder der folgenden Ebenen definiert ist.

- im Hauptmapping
- in einer lokal importierten Bibliothek
- in einer global importierten Bibliothek

Um Unklarheiten zu vermeiden, versucht MapForce in einem solchen Fall, die Funktion in genau der oben angeführten Reihenfolge aufzurufen. D.h. die direkt im Mapping definierte Funktion hat Vorrang, wenn derselbe Funktionsname in einer lokal importierten Bibliothek vorkommt. Außerdem hat die lokal importierte Funktion Vorrang vor der global importierten Funktion (vorausgesetzt beide Funktionen haben denselben Namen).

Wenn mehrere Funktionen desselben Namens vorhanden sind, wird nur die Funktion aufgerufen, die gemäß der obigen Regel Vorrang hat; andere nicht eindeutige Funktionsnamen werden blockiert. Diese blockierten Funktionen werden im Fenster "Bibliotheken" ausgegraut angezeigt und können im Mapping nicht verwendet werden.

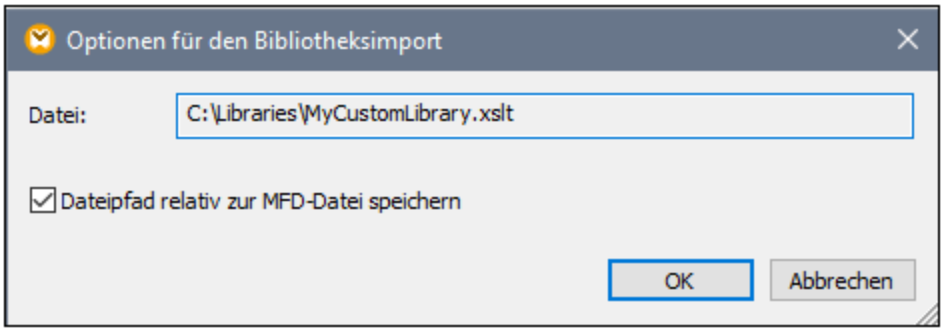
6.2.2 Relative Bibliothekspfade

Sie können den Pfad jeder importierten Bibliotheksdatei als relativ zur Mapping-Design-Datei (.mfd) definieren, vorausgesetzt, die Bibliothek wurde lokal (und nicht global) importiert, wie unter Lokale und globale Bibliotheken[948] beschrieben.

> Relative Bibliothekspfade können nur für *lokal* auf Dokumentebene importierte Bibliotheken definiert werden. Wenn ein Mapping *global* auf Programmebene importiert wurde, ist sein Pfad immer absolut.

So definieren Sie einen Bibliothekspfad als relativ zur Mapping-Design-Datei:

1. Klicken Sie im unteren Bereich des Fensters "Bibliotheken" auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster "Bibliotheken verwalten" 36 geöffnet.
2. Klicken Sie neben der gewünschten Bibliothek auf **Optionen**. (Klicken Sie alternativ dazu mit der rechten Maustaste auf die Bibliothek und wählen Sie im Kontextmenü den Befehl **Optionen** aus).



3. Aktivieren Sie das Kontrollkästchen **Alle Dateipfade relativ zur MFD-Datei speichern**.

Anmerkung: Wenn das Kontrollkästchen ausgegraut ist, vergewissern Sie sich, dass die Bibliothek tatsächlich lokal und nicht global importiert wurde.

Wenn das Kontrollkästchen aktiviert ist, aktualisiert MapForce den Pfad zu von der Komponente referenzierten Bibliotheksdateien, wenn Sie die Mapping-Datei mit dem Menübefehl **Speichern unter** in einem anderen Verzeichnis speichern. Wenn sich Bibliotheksdateien im selben Verzeichnis wie die Mapping-Datei befinden, funktioniert die Pfadreferenz weiterhin, wenn Sie das gesamte Verzeichnis in einen anderen Ordner auf dem Rechner verschieben, siehe auch Verwenden relativer Pfade in einer Komponente 85.

Beachten Sie, dass im Kontrollkästchen **Alle Dateipfade relativ zur MFD-Datei speichern** definiert ist, dass Pfade *relativ zur Mapping-Datei* sein sollen. Dies hat keine Auswirkung auf Pfade im generierten Code. Informationen zur Behandlung von Pfadreferenzen im generierten Code finden Sie unter Pfade in verschiedenen Ausführungsumgebungen 87.

© 2016-2022 Altova GmbH

6.3 Standardwerte und Node-Funktionen

Standardwerte und Node-Funktionen sind vor allem dann nützlich, wenn Sie dieselbe Verarbeitungslogik auf mehrere untergeordnete Datenelemente in einer Struktur anwenden möchten. Normalerweise müssten Sie dazu ein und dieselbe Funktion mehrmals in das Mapping kopieren. Dadurch würde das Mapping jedoch unübersichtlich werden. Standardwerte und Node-Funktionen können auf ein einziges Datenelement oder auf mehrere Datenelemente auf einmal angewendet werden. Standardwerte ersetzen leere Sequenzen. Wenn durch die Verbindung ein Wert übertragen wird, wird der Standardwert ignoriert.

Standardwerte und Node-Funktionen eignen sich für die meisten Komponenten, die eine Struktur bestehend aus Nodes haben (z.B. XML, EDI, Join-Komponenten, Variablen).

Standardwerte und Node-Funktionen sind nur mit der Built-In-Transformationssprache kompatibel. Die Ausführung solcher Mappings von generiertem C#-, C++-, Java-Programmcode aus oder mit Hilfe generierter XSLT/XQuery-Transformationen wird nicht unterstützt. Serverseitig können Sie solche Mappings mit der MapForce Server Advanced Edition ausführen.

Vorteile von Node-Funktionen und Standardwerten

Wenn Node-Funktionen und Standardwerte erstellt werden, bedeutet dies, dass eine *Regel* definiert wird. Regeln weisen die folgenden wichtigen Eigenschaften auf, wodurch sie einfach und flexibel eingesetzt werden können:

- *Vererbung.* Wenn Sie für ein Datenelement, das untergeordnete Datenelemente hat, eine Regel definieren, wird die Regel standardmäßig an diese untergeordneten Datenelemente vererbt, es sei denn Sie deaktivieren diese Option. Wenn das Datenelement, für das Sie die Funktion definieren, mehrere Ebenen oder verschachtelte untergeordnete Datenelemente hat, können Sie auswählen, ob die Regel nur auf die direkten Sub-Einträge oder auf alle untergeordneten Datenelemente angewendet werden soll.
- *Filtern.* MapForce wendet Regeln anhand von Bedingungen auf Basis des Datentyps der einzelnen Datenelemente an. Auf diese Art kann z.B. ein bestimmter Standardwert oder eine Funktion auf alle Datenelemente vom Typ `String` und ein anderer Standardwert oder eine Funktion auf alle Datenelement des Datentyps `decimal` angewendet werden. Nähere Informationen finden Sie und Anwendungsfallszenarien 957 im *Szenario 2*. Sie können auch komplexere Filteroptionen definieren: So können Sie etwa einen Datentyp definieren, mit dem Ihre Funktion übereinstimmen muss (z.B. eine Kategorie von Datentypen wie `numeric`) und die Nodes dieses Datentyps anschließend nach dem Node-Namen oder -Typ filtern (z.B. `integer`). Nähere Informationen dazu finden Sie unter *Szenario 5* 961.

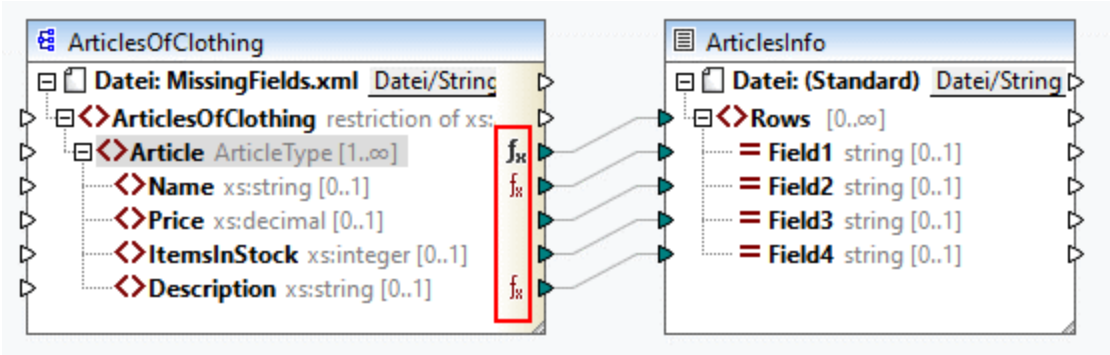
Sie können dadurch folgende Regeln in MapForce definieren:

- Ersetzen aller leeren oder Nullwerte durch einen anderen Wert, und zwar rekursiv für alle untergeordneten Datenelemente
- Ersetzen eines bestimmten Werts durch einen anderen Wert (oder einen leeren String), und zwar rekursiv für alle untergeordneten Datenelemente
- Ersetzen aller Datenbank-Nullwerte durch leere Strings oder benutzerdefinierten Text
- Anhängen eines benutzerdefinierten Präfix oder Suffix an alle Werte, die in eine Zieldatei oder -datenbank geschrieben werden

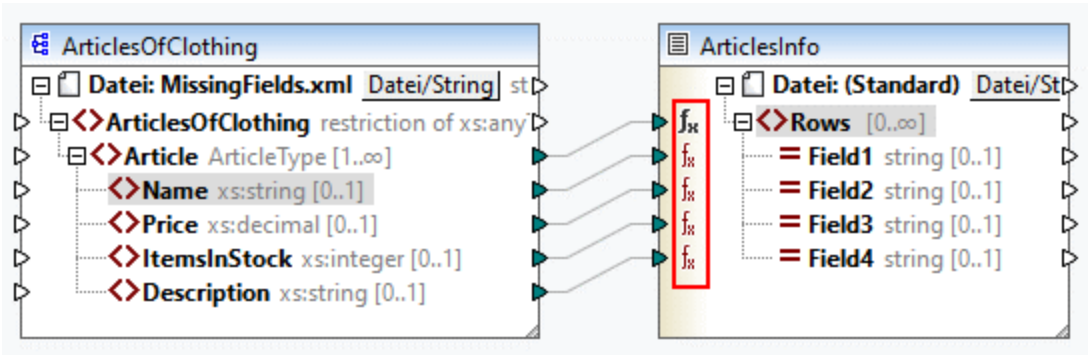
Output- oder Input-Seite

Je nach Bedarf können Sie auf der Input-, auf der Output-Seite oder auf beiden Seiten der Komponente Node-Funktionen und Standardwerte definieren. Ein Mapping funktioniert in MapForce folgendermaßen: (i) Zuerst werden Daten aus einer Quellkomponente (z.B. einer XML-Datei) ausgelesen, (ii) anschließend werden diese auf irgendeine Art verarbeitet (z.B. mit Hilfe einer Funktion) und (iii) schließlich werden die Daten in eine Zielkomponente (z.B. eine Datenbank) geschrieben. Sie können also in verschiedenen Phasen des Mappings Node-Funktionen oder Standardwerte definieren:

- Unmittelbar, *nachdem* Daten aus der Quellkomponente ausgelesen wurden, aber noch, bevor diese von Ihrem Mapping weiterverarbeitet werden, d.h. die Funktion/der Standardwert wurde auf der Output-Seite der Quellkomponente definiert (*siehe Beispiel unten*).

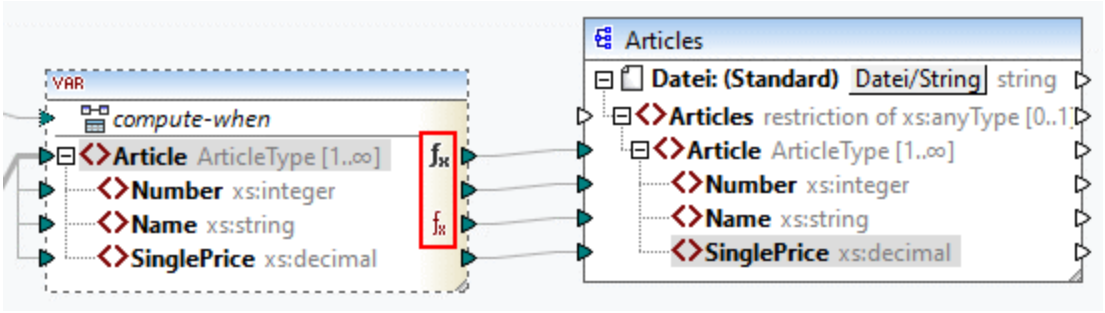


- Unmittelbar, *bevor* Daten in die Zielkomponente geschrieben werden (und nach Abschluss aller Zwischenverarbeitungsschritte), d.h. die Funktion/der Standardwert wurde auf der Input-Seite der Zielkomponente definiert (*siehe Beispiel unten*).

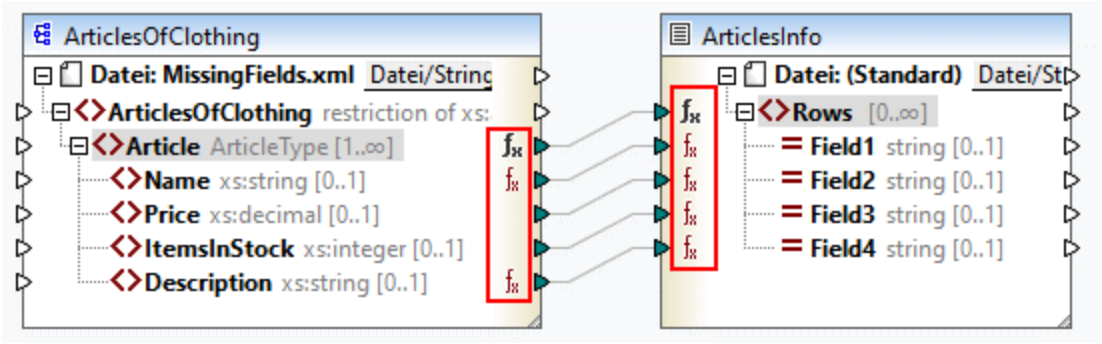


- in einem Zwischenschritt im Mapping-Prozess. Wenn das Mapping z.B. eine Zwischenvariable vom complexType enthält (z.B. eine XML-Struktur), könnten Sie alle Werte kürzen, bevor diese an die XML-Struktur übergeben werden oder unmittelbar nachdem diese von der XML-Struktur zurückgegeben werden (*siehe Beispiel unten*).

© 2016-2022 Altova GmbH



- auf der Output-Seite einer Quellkomponente und auf der Input-Seite einer Zielkomponente. Im Beispiel unten wurde in der `ArticlesOfClothing`-Komponente für alle Nodes vom Typ `string` ein Standardwert definiert. In der `ArticlesInfo`-Komponente wurde eine Node-Funktion definiert, die alle Werte von String-Nodes in Großbuchstaben transformiert.



In diesem Abschnitt

In diesem Abschnitt erfahren Sie, wie Sie eine Regel konfigurieren. Außerdem werden Anwendungsszenarien beschrieben, in denen sich Standardwerte und Node-Funktionen als nützlich erweisen und Sie erfahren, wie Sie Metadaten zu Node-Funktionen hinzufügen. Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

- Konfiguration von Regeln 953
- Anwendungsfallszenarien 957
- Node-Metadaten in Node-Funktionen 963

6.3.1 Konfiguration von Regeln

Sie können für beinahe jedes Datenelement im Mapping Node-Funktionen und Standardwerte erstellen. Wenn Node-Funktionen und Standardwerte erstellt werden, bedeutet dies, dass eine *Regel* definiert wird. Um eine Regel zu erstellen, wählen Sie das Datenelement (den Node oder das Feld) aus, für das Sie eine Regel definieren möchten. Es kann sich dabei einen einzelnen Node oder um einen Node mit untergeordneten Datenelementen handeln. Wenn Sie eine Regel für einen Node mit untergeordneten Datenelementen erstellen, wird die Regel auf alle untergeordneten Datenelemente angewendet, außer Sie deaktivieren diese Option explizit.

Wichtige Aspekte von Standardwerten und Node-Funktionen

Standardwerte und Node-Funktionen haben die folgenden wichtigen Aspekte:

- Sie können Standardwerte und Node-Funktionen auf der Input-Seite einer Zielkomponente oder auf der Output-Seite einer Quellkomponente erstellen. Um herauszufinden, welche Seite für Ihre Zwecke geeignet ist, lesen Sie nach unter Input- oder Output-Seite 952.

- Wenn für ein und dasselbe Datenelement mehrere Regeln vorhanden sind, wendet MapForce diejenige Regel an, die diesem Datenelement näher ist. Eine Anleitung dazu, wie Sie Regeln außer Kraft setzen, finden Sie in den Anwendungsfallszenarien 960 unter *Szenario 4*.

- Standardwerte und Node-Funktionen können erstellt werden, wenn der Verbindungstyp zwischen Quell- und Zielkomponente quellorientiert oder zielorientiert ist. "Alles kopieren"-Verbindungen werden jedoch nicht unterstützt. Node-Funktionen und Standardwerte werden nicht auf untergeordnete Datenelemente von "Alles kopieren"-Verbindungen angewendet. Der übergeordnete Node mit der "Alles kopieren"-Verbindung kann Node-Funktionen und Standardwerte haben, jedoch nur, wenn er einen einfachen Wert hat, z.B. ein XML-Element mit Inhalt und Attributen vom Typ simpleType. Nähere Informationen zu Verbindungsarten finden Sie in unter Verbindungsarten 93.

- Beachten Sie, dass die Erstellung von Standardwerten oder Node-Funktionen für den `Datei`-Node nicht unterstützt wird.

- In einer Node-Funktion werden nur bestimmte, in diesem Zusammenhang sinnvolle Komponenten wie z.B. vordefinierte Funktionen, Variablen, if-else-Bedingungen unterstützt. Komplexe Strukturen wie XML, JSON, EDI oder Datenbanken werden nicht unterstützt. Auch benutzerdefinierte Inline-Funktionen und Join-Komponenten werden in diesem Zusammenhang nicht unterstützt.

- Eine Node-Funktion kann einen Input-Parameter oder gar keinen Parameter haben. Der Input-Parameter hat immer den Namen `raw_value`. Löschen Sie den Input-Parameter auch dann nicht, wenn Sie für Ihre Funktion keinen Input benötigen, da sonst bei der Ausführung des Mappings Validierungsfehler entstehen. Dasselbe gilt auch für den Output der Funktion. Falls Sie eine versehentlich gelöschte Input-Komponente wiederherstellen müssen, wählen Sie den Menübefehl **Funktion | Input-Komponente einfügen.**

- Sie können in MapForce Node-Metadaten (z.B. einen Node-Namen, eine Annotation) zu einer Node-Funktion hinzufügen. Nähere Informationen dazu finden Sie unter Node-Metadaten in Node-Funktionen 963.

- Im Dialogfeld **Node-Funktionen und Standardwerte filtern** stehen verschiedene Datentypen zur Auswahl, von denen einige Typkategorien sind, d.h. sie stehen für eine größere Palette von Typen. So steht etwa der Typ `string` für verschiedene andere von `string` abgeleitete Datentypen wie z.B. `normalizedString`, `token`, `NCName`, `NMTOKEN`, `IDREF`, `ENTITY` und andere. Ebenso steht der Typ `decimal` für die abgeleiteten Typen `integer`, `long`, `short` und andere. Die Hierarchie von Typen ist in der XML Schema W3C Recommendation definiert.

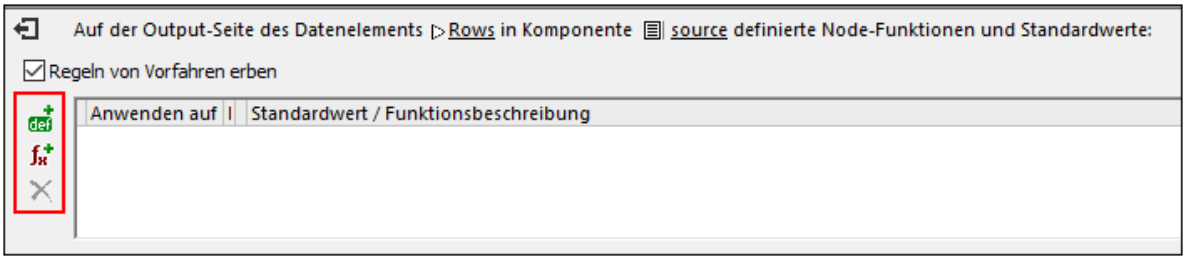
Erstellen einer Regel

Um eine Regel zu erstellen, gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf den gewünschten Node und wählen Sie im Kontextmenü die Option **Node-Funktionen und Standardwerte | Input/Output-Node-Funktionen und**

 © 2016-2022 Altova GmbH

Standardwerte. Ob es sich um *Input-* oder *Output-Node-Funktionen und Standardwerte* handelt, hängt davon ab, auf welcher Seite der Komponente Sie eine Node-Funktion oder einen Standardwert erstellen müssen. Klicken Sie alternativ dazu mit der rechten Maustaste auf einen Konnektor und wählen Sie den Node-Funktionsbefehl für die jeweilige Seite aus. Daraufhin wird im **Mapping**-Fenster das Node-Funktionsfenster angezeigt, in dem Sie Standardwerte und Node-Funktionen definieren können (*rotes Rechteck in der Abbildung unten*).



Wenn für ein übergeordnetes Datenelement Regeln definiert wurden und Sie diese vom Parent auf ein Child übertragen möchten, aktivieren Sie das Kontrollkästchen *Regeln von Vorfahren erben* (*Abbildung oben*). Nähere Informationen zur Vererbung finden Sie unter Anwendungsfallszenarien 957.

2. Im nächsten Schritt muss nun definiert werden, ob Sie einen Standardwert ( ) oder eine Funktion ( ) hinzufügen möchten. Nach Auswahl der gewünschten Option wird eine neue Regel erstellt (eine Zeile im Raster des Node-Funktionsfensters). Informationen zur Konfiguration von Regeln finden Sie weiter unten unter *Regelkonfiguration*. Wenn Sie eine Funktion definieren, werden die Input- und Output-Parameter der Funktion im Mapping-Bereich angezeigt.

Bearbeiten/Löschen von Regeln

Um eine Regel anzuzeigen, zu ändern oder zu löschen, klicken Sie auf das -Symbol (schwarz oder rot) neben dem entsprechenden Node. Im Node-Funktionsfenster werden alle für diesen Node definierten Regeln angezeigt. Wenn Sie eine Node-Funktion hinzugefügt haben, sehen Sie im Raster die Schaltfläche Bearbeiten , über die Sie die Implementierung der Funktion ändern können. Wenn die Schaltfläche Bearbeiten nicht vorhanden ist, wurde die Funktion höchstwahrscheinlich für ein übergeordnetes Datenelement definiert. Klicken Sie in diesem Fall neben dem Datenelement, für das die Regel definiert wurde, auf die Schaltfläche .

Um eine Regel zu löschen, wählen Sie sie im Raster im Node-Funktionsfenster aus und klicken Sie auf die Schaltfläche .

Konfigurieren einer Regel

Sobald Sie im Kontextmenü **Node-Funktionen und Standardwerte | Input/Output-Funktionen und Standardwerte** auswählen (*siehe Anleitung oben*), wird das Node-Funktionsfenster angezeigt, in dem Sie Node-Funktionen und Standardwerte konfigurieren können. Weiter unten finden Sie eine Beschreibung der verfügbaren Einstellungen.

- *Anwenden auf:* Eine Regel kann auf das aktuelle Datenelement, alle direkten Sub-Einträge oder auf alle untergeordneten Datenelement angewendet werden. Wenn das in ausgewählte Datenelement keine untergeordneten Datenelemente hat, steht nur die Option *Aktuelles Datenelement* zur Auswahl.

- *Datentyp:* Klicken Sie auf die Schaltfläche ... und wählen Sie einen Datentyp aus dem Dialogfeld aus. Die Regel wird nur auf Datenelemente dieses Datentyps (oder eines davon abgeleiteten

Datentyps) angewendet. Wenn das ausgewählte Datenelement keine untergeordneten Datenelemente hat, steht nur der Datentyp dieses Datenelements zur Auswahl.

- *Standardwert/Funktionsbeschreibung:* Wenn Sie einen Standardwert definieren, geben Sie den gewünschten Standardwert ein, der für das ausgewählte Datenelement (und ggf. auf alle untergeordneten Datenelemente) definiert werden soll. Um einen leeren String als Standardwert zu definieren, lassen Sie dieses Feld leer. Wenn Sie eine Funktion definieren, dient dieses Feld nur zu Informationszwecken: Sie sehen darin eine zusammenfassende Beschreibung der Funktion.

Um das Funktionsfenster zu schließen, klicken Sie in der linken oberen Ecke dieses Fensters auf die Schaltfläche oder drücken Sie die **Escape**-Taste.

Visuelle Anhaltspunkte

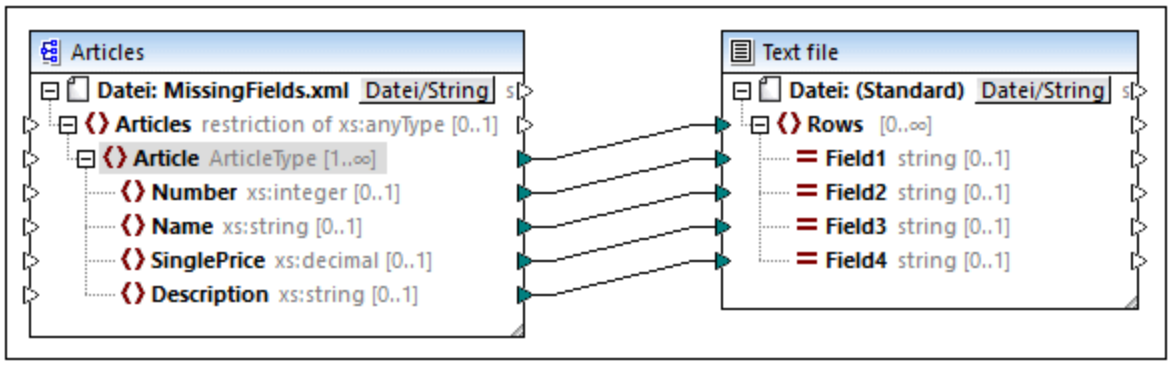
Damit Sie besser erkennen, welche Regeln definiert sind und ob sie auf einen bestimmten Node angewendet werden, werden in MapForce verschiedene visuelle Hilfsmittel angezeigt (*siehe Tabelle unten*).

Symbol	Beschreibung
	Dieses Symbol gibt an, dass für dieses Datenelement eine Regel definiert wurde, die sich eventuell auch auf alle untergeordneten Datenelemente auswirkt. Klicken Sie auf das Symbol, um die Regel zu ändern oder zu löschen.
	Dieses Symbol gibt an, dass das Datenelement die auf einer übergeordneten Ebene definierte Regel erbt.
	Dieses Symbol gibt an, dass für dieses Datenelement eine Regel definiert ist und darauf angewendet wird. Dieses Symbol wird normalerweise angezeigt, wenn für einen einzelnen Node eine Funktion oder ein Standardwert definiert ist.
	Dieses Symbol gibt an, dass zwar eine Regel auf dieses Datenelement angewendet würde, diese aber absichtlich blockiert wurde. Dieses Symbol wird nur dann angezeigt, wenn die Vererbung blockiert wurde und für diesen Node keine anderen Regeln definiert sind. Wenn eine Regel von einem übergeordneten Datenelement angewendet wird, hat das Symbol Vorrang.
	Dieses Symbol gibt an, dass eine für dieses Datenelement definierte Regel nicht aktiv ist. Dieses Symbol könnte z.B. bei Child-Datenelementen angezeigt werden, die noch nicht verbunden wurden.
	Dieses Symbol zeigt an, dass die Regel derzeit nicht aktiv ist, da keine Node-Entsprechungen und Verbindungen gefunden wurden. Mit *Node-Entsprechungen* ist Folgendes gemeint: Bevor eine Funktion oder ein Standardwert angewendet wird, sucht MapForce nach dem Datentyp, den Sie definiert haben; wenn Sie z.B. auch einen Filter definiert haben, dem ein bestimmter Node-Name entsprechen muss, so sucht MapForce auch nach dem entsprechenden Node-Namen.

© 2016-2022 Altova GmbH

6.3.2 Anwendungsfallszenarien

In diesem Kapitel werden verschiedene Szenarien beschrieben, in denen sich Standardwerte und Node-Funktionen als nützlich erweisen können. Um diese Szenarien auszuprobieren, benötigen Sie das folgende Beispielmapping: **`Tutorial\MissingFields.mfd`**. Unsere Ausgangsbasis bildet das unten gezeigte Mapping.



Einige der Nodes in der Quelldatei sind leer. Zu diesem Zeitpunkt sieht die Ausgabedatei folgendermaßen aus:

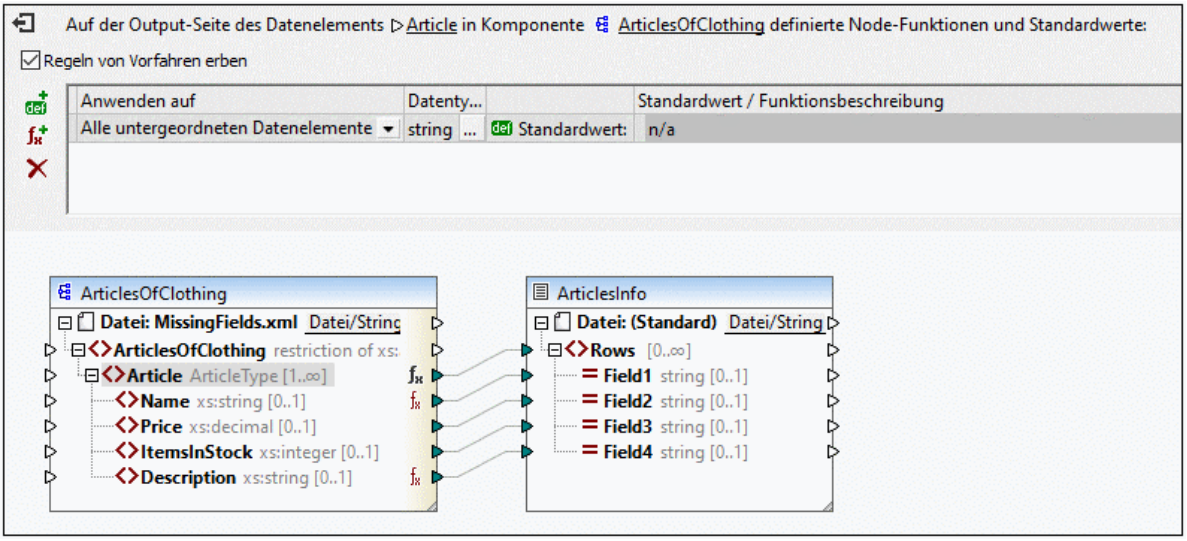
T-Shirt Shirt Pants Jacket	25.3 70.3 57.5	20 60 40	Available in black, blue, and red Pants Limited stock

In den unten beschriebenen Szenarien wird gezeigt, wie Sie für alle Nodes vom Typ `string` einen Standardwert definieren, Standardwerte für verschiedene Datentypen festlegen, Regeln für bestimmte Nodes blockieren, geerbte Regeln außer Kraft setzen, eine Node-Funktion definieren und diese Funktion auf Basis von Bedingungen auf die entsprechenden Nodes anwenden und Standardwerte auf nicht verbundene Nodes anwenden.

Szenario 1: Definieren eines Standardwerts für alle String-Nodes

Einige der Nodes vom Typ `string` sind in unserem Mapping leer. Wir möchten nun anstelle der leeren Strings den Standardwert `n/a` definieren. Gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf das Element `Article` und wählen Sie im Kontextmenü die Option **Node-Funktionen und Standardwerte | Output-Node-Funktionen und Standardwerte**.
2. Klicken Sie im Node-Funktionsfenster auf die Option  und geben Sie in das Feld *Standardwert* `n/a` ein (*siehe Abbildung unten*).



Die Elemente `Name` und `Description` haben in der Komponente `ArticlesOfClothing` den Typ `string`, weshalb der Standardwert für beide davon verwendet werden kann. Daher wird nun neben diesen Nodes das Symbol f_x angezeigt. Da Standardwerte nur auf leere Werte angewendet werden, haben nur zwei `Field4`-Nodes in der Ausgabe den Wert `n/a` erhalten (*unten gelb markiert*).

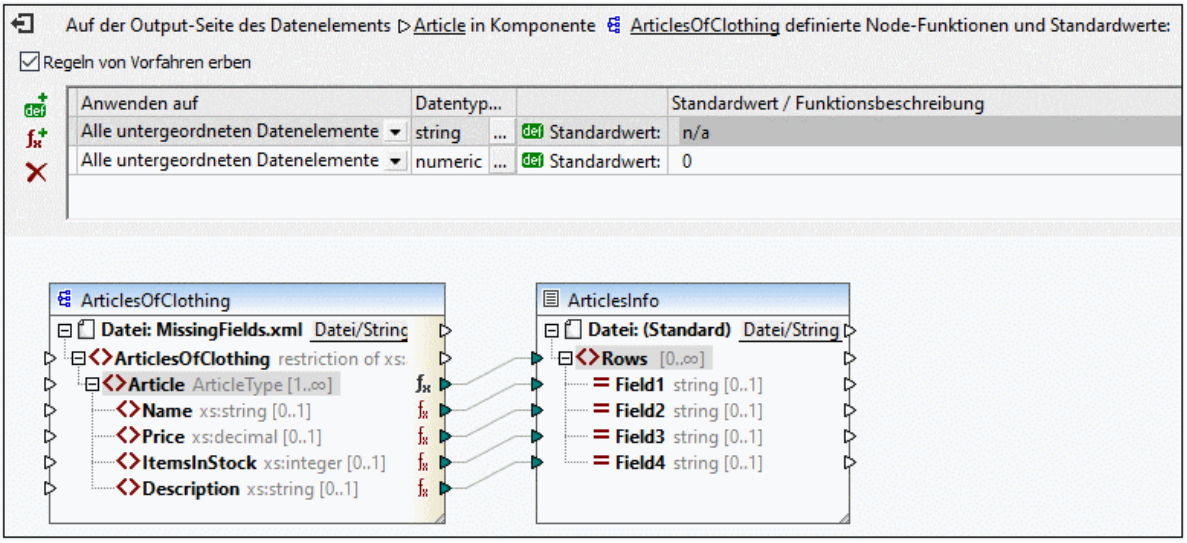
T-Shirt	25.3	20	Available in black, blue, and red
Shirt	70.3	60	n/a
Pants			Limited stock
Jacket	57.5	40	n/a

Szenario 2: Definieren der Standardwerte für unterschiedliche Datentypen

Sie können nicht nur für einen einzigen bestimmten Typ einen Standardwert definieren, sondern zusätzlich auch einen weiteren Standardwert für einige andere Datentypen definieren. So hat etwa einer der Artikel in `Price` und `ItemsInStock` leere Werte, wobei beide den Typ "Zahl" haben. Wir möchten nun diese leeren Werte durch den Wert `0` ersetzen. Gehen Sie folgendermaßen vor:

1. Wiederholen Sie die Schritte aus *Szenario 1*. Dadurch wird für alle Nodes vom Typ `string`, die einen leeren Wert haben, ein Standardwert hinzugefügt.
2. Fügen Sie einen weiteren Standardwert hinzu und wählen Sie als Datentyp `numeric` aus.
3. Geben Sie in das Feld *Standardwert* `0` ein (*siehe Abbildung unten*).

© 2016-2022 Altova GmbH



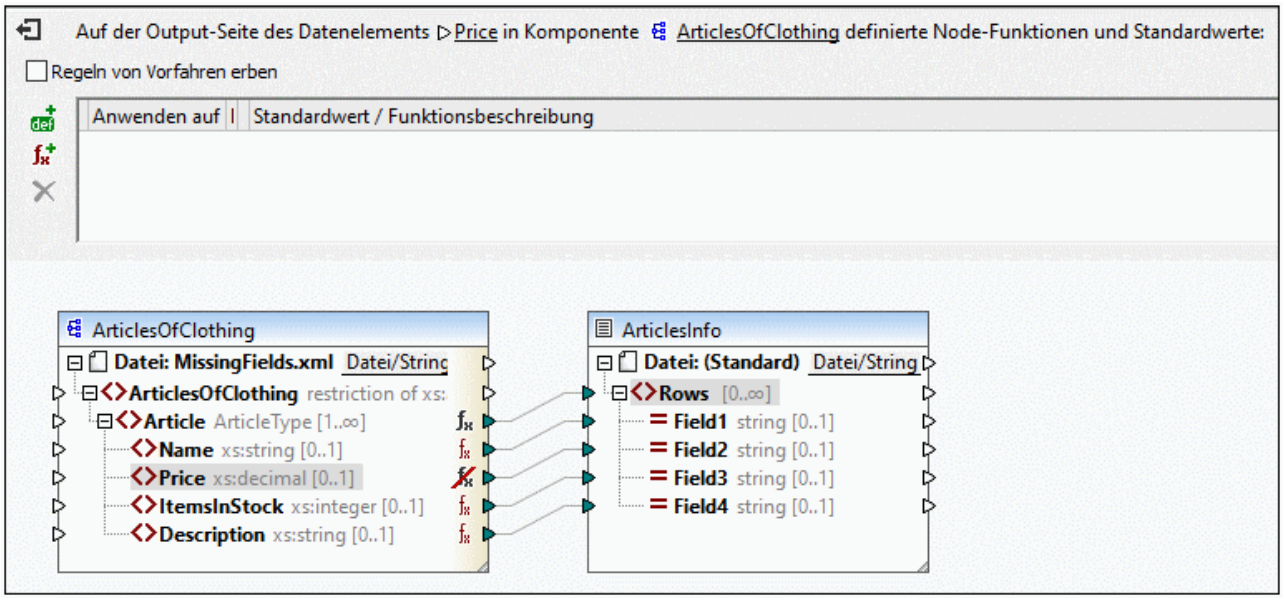
Die Ausgabe sieht nun folgendermaßen aus (*Beachten Sie die markierten Bereiche*):

T-Shirt	25.3	20	Available in black, blue, and red
Shirt	70.3	60	n/a
Pants	0	0	Limited stock
Jacket	57.5	40	n/a

Anmerkung: Gemäß der XML Schema-Spezifikation ist `integer` von `decimal` abgeleitet. Beide gehören zum numerischen Datentyp. In unserem Beispiel wird die Regel sowohl auf das Element `ItemsInStock` als auch auf das Element `Price` angewendet, wenn Sie im Node-Funktionsfenster als Datentyp `numeric` auswählen. Wenn Sie als Datentyp `decimal` auswählen, wird die Regel dennoch auf beide Elemente angewendet. Wenn Sie jedoch als Datentyp `integer` auswählen, wird die Regel nur auf das Element `ItemsInStock` angewendet.

Szenario 3: Blockieren von Regeln für bestimmte Nodes

In diesem Szenario sollen auf alle Nodes vom Typ "string" und "numeric" Standardwerte angewendet werden, aber für das Datenelement `Price` soll kein Standardwert definiert werden. Wiederholen Sie zu diesem Zweck die Schritte aus *Szenario 2*, klicken Sie anschließend auf das Element `Price` und deaktivieren Sie das Kontrollkästchen *Regeln von Vorfahren erben* Das Datenelement `Price` erbt im Mapping unten nun nicht mehr die Regeln von seinem übergeordneten Datenelement `Article`. Daher wird neben dem Element `Price` das Symbol  angezeigt (*siehe Abbildung unten*).

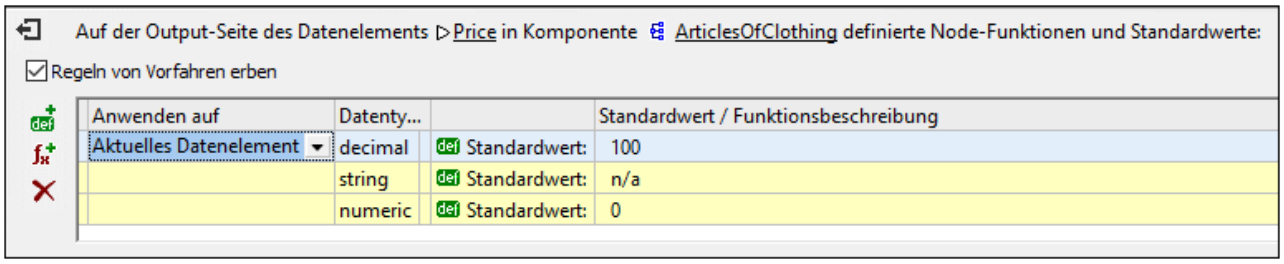


In der Ausgabe unten sehen Sie, dass der leere Wert des Elements `Price`, für den die Regel blockiert wurden, auf `Field2` gemappt wurde (*Beachten Sie die markierten Bereiche*).

T-Shirt	25.3	20	Available in black, blue, and red
Shirt	70.3	60	n/a
Pants		0	Limited stock
Jacket	57.5	40	n/a

Szenario 4: Außerkraftsetzen von geerbten Regeln

In diesem Szenario sollen für alle Nodes vom Typ "string" und "numeric" Standardwerte definiert werden. Nur für das Element `Price` soll jedoch ein Standardwert von `100` definiert werden. Wiederhole Sie dazu die Schritte aus *Szenario 2*, klicken Sie in der Komponente `ArticlesOfClothing` auf `Price`, fügen Sie einen neuen Standardwert hinzu und geben Sie `100` ein, wie in der Abbildung unten gezeigt. Die geerbten Regeln haben einen a gelben Hintergrund.



Die direkt für einen Node definierte Regel hat Vorrang vor Regeln, die auf einer übergeordneten Ebene erstellt wurden. Die Ausgabe sieht daher folgendermaßen aus (*Beachten Sie den markierten Bereich*):

T-Shirt	25.3	20	Available in black, blue, and red
Shirt	70.3	60	n/a

© 2016-2022 Altova GmbH

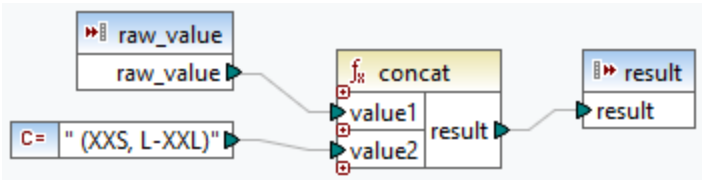
Pants Jacket	100 57.5	0 40	Limited stock n/a

Wenn Sie für ein und denselben Node mehr als eine Regel definiert haben, wird die oberste Regel des Rasters auf den aktuellen Node angewendet. Sie können die Reihenfolge dieser Regeln ändern, indem Sie die Regeln im Raster an eine andere Stelle ziehen.

Szenario 5: Erstellen einer Node-Funktion und Anwendung dieser Funktion auf Basis von Bedingungen

In diesem Szenario wird erläutert, wie Sie eine Node-Funktion erstellen und diese auf Basis von Bedingungen anwenden. Mit Hilfe der Funktion **concat** werden wir Text zu Nodes vom Typ `string` hinzufügen. Gehen Sie folgendermaßen vor:

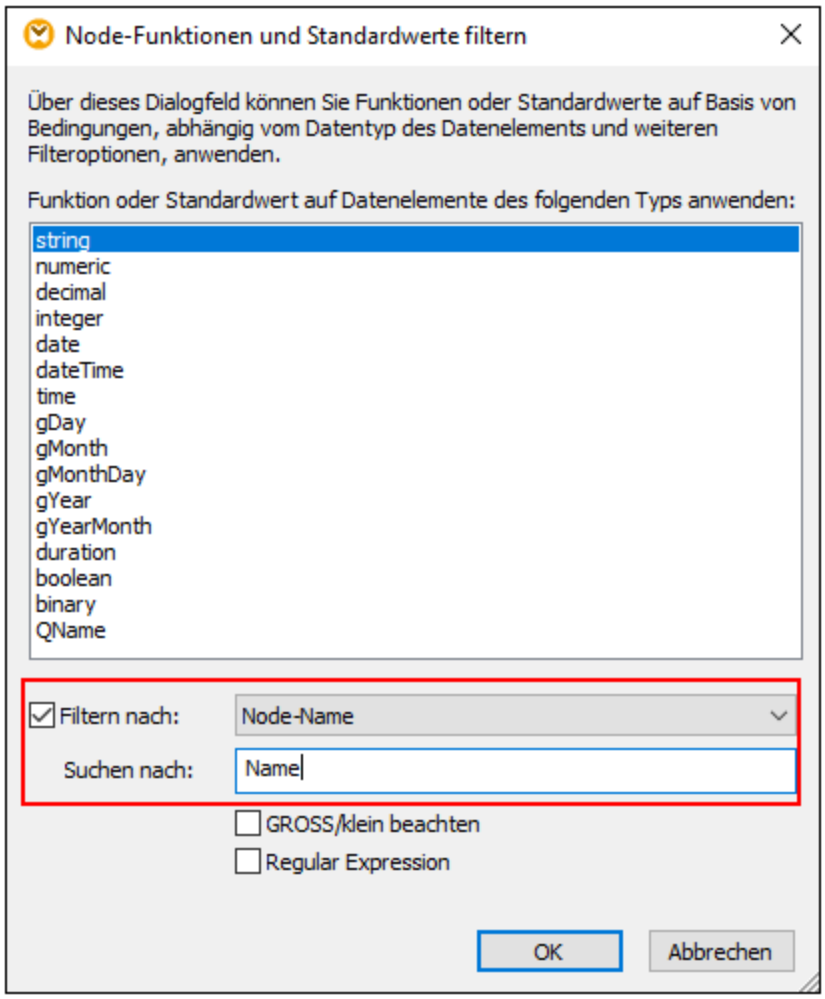
1. Klicken Sie mit der rechten Maustaste auf das Element `Article` und wählen Sie im Kontextmenü die Option **Node-Funktionen und Standardwerte | Output-Node-Funktionen und Standardwerte**.
2. Fügen Sie durch Klick auf das Symbol eine Node-Funktion hinzu und erstellen Sie die unten gezeigte Funktion.



Wir haben die Nodes zu diesem Zeitpunkt noch nicht gefiltert. Die Ausgabe sieht folgendermaßen aus (*Beachten Sie die markierten Bereiche*):

A Shirt (XXS, L-XXL) Shirt (XXS, L-XXL) Pants (XXS, L-XXL) Jacket (XXS, L-XXL)	25.3 70.3 57.5	20 60 40	Available in black, blue, and red (XXS, L-XXL) Limited stock (XXS, L-XXL)

In der Ausgabe sehen Sie, dass unsere Regel auf beide Elemente vom Typ `string` angewendet wurde: `Name` und `Description`. Die Regel soll nun nur auf das Element `Name` angewendet werden. Gehen Sie dazu zum Node-Funktionsfenster und klicken Sie in der Spalte *Datentyp und Filter* auf das Symbol ... Daraufhin wird das Dialogfeld **Node-Funktionen und Standardwerte filtern**, in dem Sie die Datentypen ändern und Filteroptionen definieren können, aufgerufen. Definieren Sie den Filter, wie in der Abbildung unten gezeigt.



Die Ausgabe sieht nun folgendermaßen aus:

A Shirt (XXS, L-XXL) Shirt (XXS, L-XXL) Pants (XXS, L-XXL) Jacket (XXS, L-XXL)	25.3 70.3 57.5	20 60 40	Available in black, blue, and red Limited stock

Filtern nach Node-Typ

Mit Hilfe dieser Option können Sie den im Dialogfeld **Node-Funktionen und Standardwerte filtern** ausgewählten Datentyp einschränken. Sie haben z.B. ausgewählt, dass Ihre Node-Funktion auf numerische Nodes angewendet wird. Anschließend können Sie einen Subtyp (z.B. *Filtern nach/Node-Name:* `xs:decimal`) definieren.

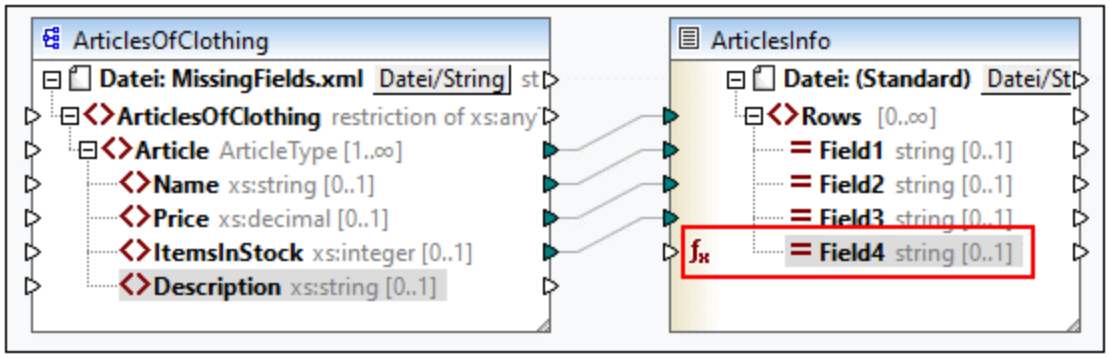
© 2016-2022 Altova GmbH

Filtern mittels Regular Expressions
Für komplexere Filteroptionen können Regular Expressions verwendet werden (z.B. für Entsprechungen mit mehreren Node-Namen oder Typnamen). Nähere Informationen dazu finden Sie unter Regular Expressions[1020]. Beachten Sie die folgenden Punkte:

- Die Anker `^` und `$` sind implizit und dürfen nicht in das Feld *Suchen nach* eingegeben werden.
- Die Berücksichtigung der Groß- und Kleinschreibung kann über das Kontrollkästchen *GROSS/klein beachten* definiert werden. Das `i`-Flag wird daher nicht unterstützt.
- Dass Übereinstimmungen über mehrere Zeilen erfolgen, ist bei der Filterung von Nodes nicht sinnvoll. Das `m`-Flag wird daher nicht unterstützt.

Szenario 6: Erstellen von Standardwerten für nicht verbundene Nodes

Sie können in MapForce Standardwerte auf nicht verbundene Nodes anwenden, jedoch nur, wenn der übergeordnete Node und/oder ein gleichrangiger Node verbunden ist. Betrachten Sie dazu das folgende Beispiel:



Wir haben im obigen Mapping in der Komponente `ArticlesInfo` einen Standardwert für den Node `Field4` definiert. `Field4` hat zu diesem Zeitpunkt keine Input-Verbindung. Deer übergeordnete Node und gleichrangige Nodes sind verbunden. Wenn `Field4` unverbunden bleibt, werden alle `Description`-Werte in der Ausgabe durch den Standardwert überschrieben (*siehe unten*). Wenn Sie `Description` und `Field4` verbinden, wird der Standardwert nur auf leere Sequenzen angewendet.

T-Shirt	25.3	20	n/a
Shirt	70.3	60	n/a
Pants			n/a
Jacket	57.5	40	n/a

Anmerkung: Standardwerte werden nicht auf unverbundene Nodes in JSON-Komponenten angewendet.

6.3.3 Node-Metadaten in Node-Funktionen

Manchmal gibt es Fälle, in denen eine Node-Funktion etwas auf Basis von *Node-Metadaten* tun soll. Unter "Node-Metadaten" verstehen wir verschiedene in Schema gespeicherte Informationen über den Node, auf den die Funktion angewendet wird, wie z.B. Node-Name, die Länge des Werts oder die Genauigkeit (precision). Die unten stehende Tabelle enthält eine Liste aller möglichen Metadatenparameter, die in einer Node-Funktion verwendet werden können. Beachten Sie, dass einige Metadatenparameter nur auf bestimmte Datentypen

angewendet werden. Es wird eine Warnung in MapForce angezeigt, wenn Sie versuchen, Metadaten zu verwenden, die mit dem aktuellen Node nicht kompatibel sind.

Metadatenparameter	Beschreibung
`node_name`	Liefert den Namen des aktuellen Node, so wie er in der Komponente angezeigt wird. Diese Metadaten werden von allen Nodes unterstützt. Bei XML-Daten bezieht sich `node_name` auf den Namen des aktuellen Elements oder Attributs. Bei CSV-Daten bezieht sich `node_name` auf den Namen eines CSV-Felds. In Datenbanken bezieht sich `node_name` auf den Namen einer Tabellenspalte.
`node_annotation`	Zeigt in der Ausgabe Annotationstext neben dem Node-Wert an. Diese Metadaten werden von allen Nodes unterstützt.
`node_minLength`	Liefert den Wert des `minLength` Facet des Datentyps des Node. Gilt für XML- und Text-Nodes des entsprechenden Typs. Nähere Informationen zu die Länge einschränkenden Facets finden Sie in der XML Schema Recommendation.
`node_maxLength`	Liefert den Wert des `maxLength` Facet des Datentyps des Node. Gilt für XML- und Text-Nodes des entsprechenden Typs. Nähere Informationen zu die Länge einschränkenden Facets finden Sie in der XML Schema Recommendation.
`node_totalDigits`	Liefert den Wert des `totalDigits` Facet des Datentyps des Node. Gilt für XML-Nodes des entsprechenden Typs. Nähere Informationen zum `totalDigits` Facet finden Sie in der XML Schema Recommendation.
`node_fractionDigits`	Liefert den Wert des `fractionDigits` Facet des Datentyps des Node. Gilt für XML-Nodes des entsprechenden Typs. Nähere Informationen zum `fractionDigits` Facet finden Sie in der XML Schema Recommendation.
`node_length`	Liefert die Länge einer Zahl (in Bytes). Gilt für Datenbankfelder des entsprechenden Typs. Nähere Informationen zur Länge finden Sie in der Microsoft-Dokumentation.
`node_precision`	Liefert die Anzahl aller Stellen einer Zahl. Gilt für Datenbankfelder des entsprechenden Typs. Nähere Informationen dazu finden Sie in der Microsoft-Dokumentation.
`node_scale`	Liefert die Anzahl der Stellen rechts vom Dezimalkomma einer Zahl. Gilt für Datenbankfelder des entsprechenden Typs. Nähere Informationen dazu finden Sie in der Microsoft-Dokumentation.

Anmerkung: Der Datentyp eines Metadateparameters muss mit dem Datentyp des entsprechenden Node übereinstimmen. Andernfalls schlägt die Ausführung der Funktion fehl.

Hinzufügen von Metadaten zu einer Node-Funktion

Um einen Metadatenparameter zu einer Node-Funktion hinzuzufügen, müssen Sie eine Node-Funktion erstellen 954 und anschließend unterhalb des Node-Funktionsfensters auf die Option **Node-Angaben hinzufügen** klicken. Daraufhin wird das Dialogfeld **Input mit Node-Informationen einfügen**, in dem Sie den

© 2016-2022 Altova GmbH

entsprechenden Metadatenparameter auswählen können, aufgerufen. Anstelle der Option **Node-Angaben hinzufügen** können Sie auch eine der folgenden Optionen auswählen:

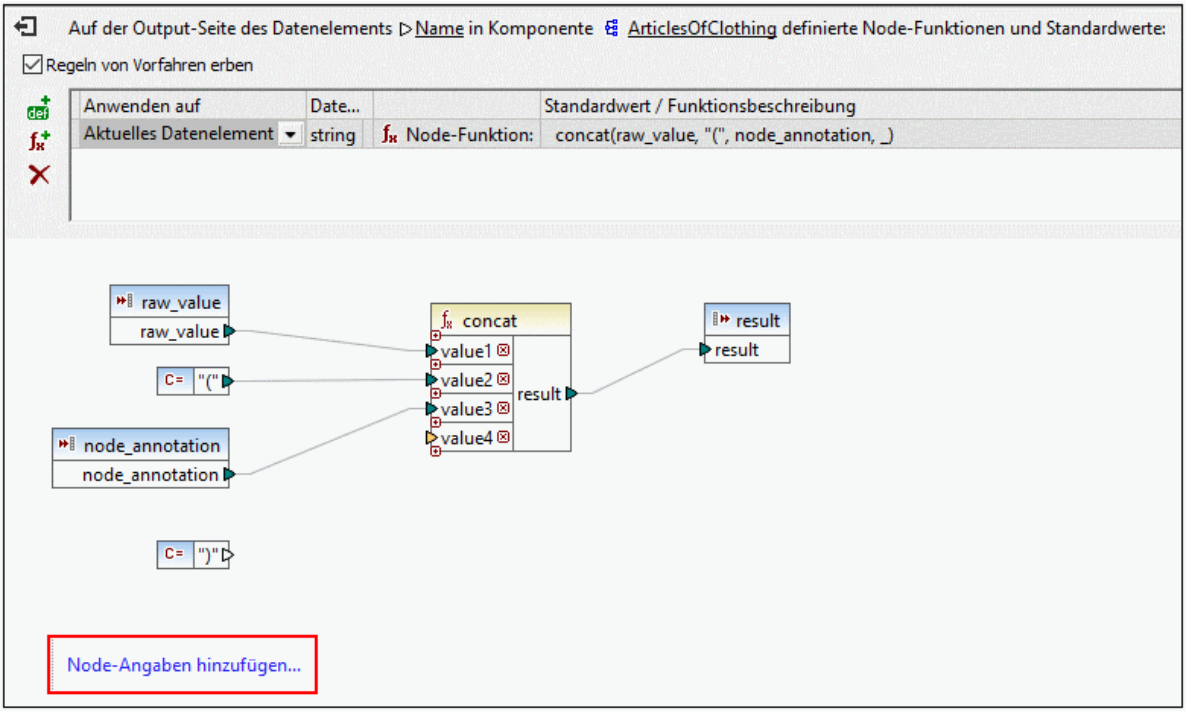
- Klicken Sie mit der rechten Maustaste auf einen leeren Bereich im Mapping und wählen Sie im Kontextmenü den Befehl **Input-Komponente einfügen**.
- Klicken Sie auf die Symbolleisten-Schaltfläche .
- Klicken Sie im Menü **Funktion** auf **Input-Komponente einfügen**.

Wenn Metadaten von einem Node nicht unterstützt werden
Wenn ein Metadatenparameter von einem Node nicht unterstützt wird, können Sie MapForce anweisen, eine leere Sequenz zurückzugeben oder die Node-Funktion nicht anzuwenden (beide Optionen stehen im Dialogfeld **Input mit Node-Informationen einfügen** zur Verfügung). Die leere Sequenz muss behandelt werden, da sonst möglicherweise gar kein Wert von der Funktion zurückgegeben wird. Normalerweise müssen sie diese mit Hilfe von Sequenzfunktionen wie `substitute-missing`[1103] oder `exists`[1079] oder anderen Komponententypen weiter verarbeiten.

Beispiel

Im unten gezeigten Beispiel sehen Sie, wie Sie Annotationstext zu einer Node-Funktion hinzufügen. Sie benötigen für dieses Beispiel das folgende Mapping: **`<Dokumente>\Altova\MapForce2023\MapForceExamples\MissingFields.mfd`**. Wir haben in **`MissingFields.xsd`** (der Datei, auf der **`MissingFields.xml`** basiert) die folgenden Annotation zum Element `Name`hinzugefügt: `Article name from catalog Spring 2022`. Diese Annotation soll nun an eine Node-Funktion übergeben und in der Ausgabe angezeigt werden. Gehen Sie folgendermaßen vor:

1. Erstellen Sie in der Komponente `ArticlesOfClothing` für das Element `Name` eine Node-Funktion[954].
2. Replizieren Sie die unten gezeigte Funktion. Um einen `node_annotation`-Parameter hinzuzufügen, klicken Sie auf **Node-Angaben hinzufügen** (*siehe rotes Rechteck unten*) und wählen Sie diesen Parameter aus der Liste aus. Eine Anleitung dazu, wie Sie Funktionen hinzufügen und bearbeiten, finden Sie unter Grundlegendes zu Funktionen[943].



Anmerkung: Tief verschachtelte Strukturen werden standardmäßig nicht zur Gänze untersucht, um Arbeitsspeicher zu sparen und die Leistung zu verbessern. Wenn die Komponente, auf die die Node-Funktion angewendet werden soll, so tief verschachtelte Strukturen hat, können Sie die entsprechenden Nodes im Mapping erweitern, damit MapForce diese untersucht. In diesem Fall berücksichtigt MapForce diese Nodes, wenn Sie einen neuen Metadatenparameter hinzufügen und die Warnung wird eventuell nicht mehr angezeigt. Die Node-Funktion muss verbunden sein, damit sie wirksam wird; ob nicht verbundene Datenelemente erweitert werden, spielt keine Rolle.

Ausgabe

Im Codefragment unten sehen Sie, dass die Annotation zu den Werten der `Name`-Elemente hinzugefügt wurde.

```
T-Shirt (Article name from catalog Spring 2022)   25.5  20    Available in black, blue,
and red
Shirt (Article name from catalog Spring 2022)     70.3  60

Pants (Article name from catalog Spring 2022)                 Limited
stock
Jacket (Article name from catalog Spring 2022)    57.5  40
```


© 2016-2022 Altova GmbH

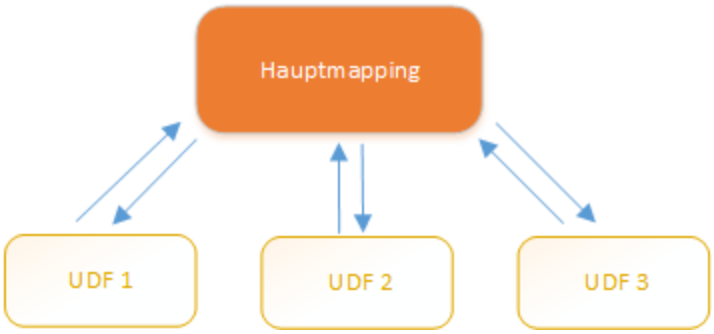
6.4 Benutzerdefinierte Funktionen

Benutzerdefinierte Funktionen (UDFs = User Defined Functions) sind eigene Funktionen, die der Benutzer einmal definiert und die anschließend im selben oder in mehreren Mappings wiederverwendet werden können. Benutzerdefinierte Funktionen bilden selbst eine Art von Mini-Mapping: Normalerweise bestehen sie aus einem oder mehreren Input-Parametern, einigen Datenverarbeitungszwischenkomponenten und einem Output zur Rückgabe der Daten an das aufrufende Mapping bzw. eine andere benutzerdefinierte Funktion, die diese aufruft.

Vorteile von benutzerdefinierten Funktionen

Benutzerdefinierte Funktionen haben folgende Vorteile:

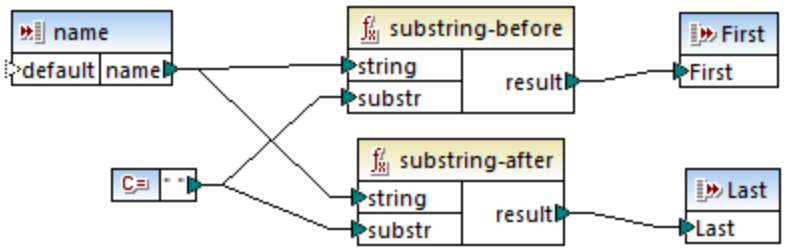
- Sie können mehrmals in einem Mapping oder auch in mehreren Mappings verwendet werden.
- Benutzerdefinierte Funktionen machen Ihr Mapping übersichtlicher. So können darin etwa Teile eines Mappings in kleinere Bausteine verpackt werden, sodass die Implementierungsdetails ausgeblendet werden. Ein Beispiel dafür sehen Sie im nachstehenden Diagramm.



- Benutzerdefinierte Funktionen sind flexibel und dienen dazu, Strings, Zahlen, Datumsangaben und andere Daten auf eine bestimmte über vordefinierte MapForce-Funktionen hinausgehende Weise zu verarbeiten. So können Sie etwa Text auf eine bestimmte Art verketten oder aufteilen, komplexe Berechnungen durchführen, Datums- und Uhrzeitangaben bearbeiten usw.
- Eine weitere häufige Anwendungsmöglichkeit von benutzerdefinierten Funktionen ist, den Inhalt eines bestimmten Felds in einer XML-Datei, einer Datenbank oder einem anderen von Ihrer MapForce Edition unterstützten Datenformat abzurufen und diese Daten auf geeignete Weise darzustellen. Nähere Informationen dazu finden Sie unter Look-up-Implementierung 980.
- Benutzerdefinierte Funktionen können rekursiv aufgerufen werden (d.h. die benutzerdefinierte Funktion ruft sich selbst auf). Die benutzerdefinierte Funktion muss dazu als reguläre Funktion (nicht inline gesetzte Funktion) 970 definiert sein. Mit Hilfe rekursiver benutzerdefinierter Funktionen 978 können Sie verschiedene komplexe Mapping-Aufgaben lösen, wie z.B. das Iterieren über Datenstrukturen mit einer Tiefe von *N* Children, wobei *N* im Vorhinein nicht bekannt ist.

Beispiel

Im Folgenden sehen Sie ein Beispiel für eine einfache benutzerdefinierte Funktion, die einen String in zwei separate Strings aufteilt. Diese benutzerdefinierte Funktion ist Teil eines größeren Mappings mit dem Namen **`MapForceExamples\ContactsFromPO.mfd`**. Sie erhält als Parameter einen Namen (z.B. `Helen Smith`), wendet die vordefinierten Funktionen **`substring-before`** und **`substring-after`** darauf an und gibt anschließend als Ergebnis zwei Werte zurück: `Helen` und `Smith`.



In diesem Abschnitt

In diesem Abschnitt wird erläutert, wie Sie mit benutzerdefinierten Funktionen arbeiten. Er ist in die folgenden Kapitel gegliedert:

- Benutzerdefinierte Funktionen: Grundlagen 968
- Parameter von benutzerdefinierten Funktionen 973
- Rekursive benutzerdefinierte Funktionen 978
- Look-up-Implementierung 980

6.4.1 Benutzerdefinierte Funktionen: Grundlagen

In diesem Kapitel wird beschrieben, wie Sie benutzerdefinierte Funktionen (kurz UDFs = User-Defined Functions) erstellen, importieren, bearbeiten, kopieren und einfügen und löschen.

Erstellen einer benutzerdefinierten Funktion

In diesem Unterabschnitt erfahren Sie, wie Sie eine benutzerdefinierte Funktion von Grund auf neu oder anhand bestehender Komponenten erstellen. Sie benötigen dazu mindestens eine Ausgabekomponente, mit der einige Daten verbunden sind. Als Input-Parameter kann eine Funktion null, einen oder mehrere Inputs haben. Die Input- und Output-Parameter können den Typ simpleType (z.B. String) oder complexType (eine Struktur) haben. Nähere Informationen zu einfachen und komplexen Parametern finden Sie unter Parameter von benutzerdefinierten Funktionen 973.

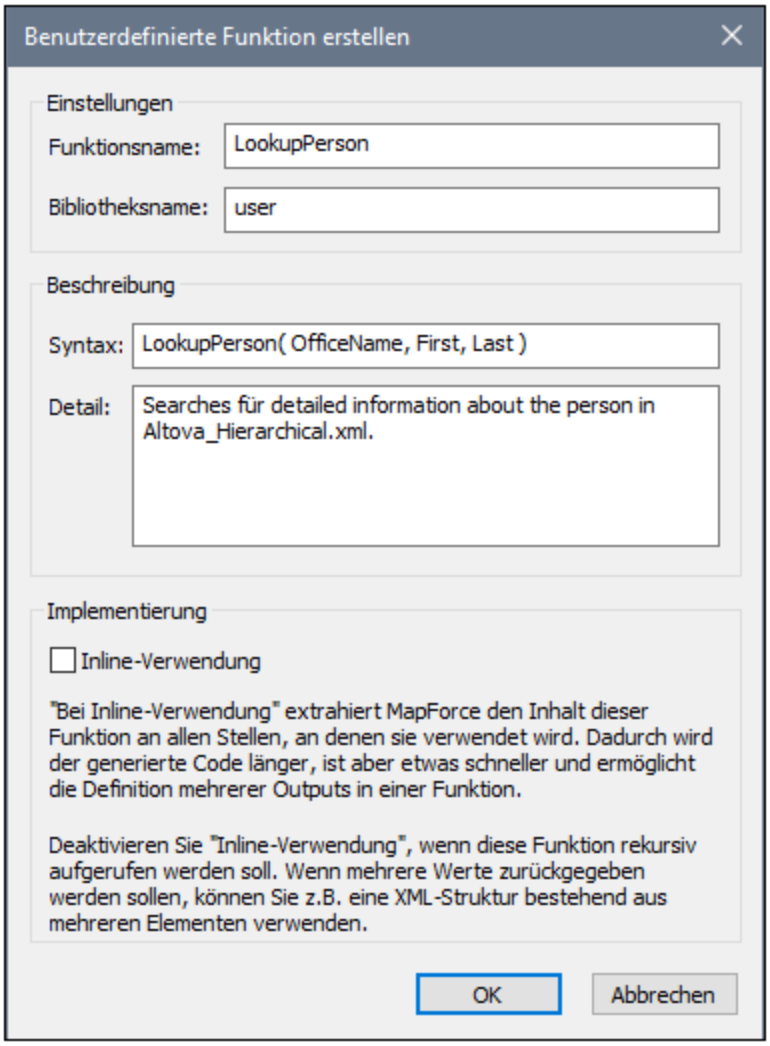
Neuerstellung einer benutzerdefinierten Funktion

Um eine benutzerdefinierte Funktion von Grund auf neu zu erstellen, gehen Sie folgendermaßen vor:

1. Wählen Sie **Funktion | Benutzerdefinierte Funktion erstellen**. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche .

2. Geben Sie die entsprechenden Informationen in das Dialogfeld **Benutzerdefinierte Funktion erstellen** ein (*siehe Abbildung unten*).

© 2016-2022 Altova GmbH



Es stehen die folgenden Optionen zur Verfügung:

- *Funktionsname:* Obligatorisches Feld. Im Namen der benutzerdefinierten Funktion können die folgenden Zeichen verwendet werden: Alphanumerische Zeichen (a-z, A-Z, 0-9), ein Unterstrich (_), ein Bindestrich (-) und ein Doppelpunkt (:).
- *Bibliotheksname:* Obligatorisches Feld. Dies ist der Name einer Funktionsbibliothek (im Fenster "Bibliotheken" 34), in dem Ihre Funktion gespeichert wird. Wenn Sie keine Bibliothek angeben, wird die Funktion in eine Standardbibliothek namens `user` platziert.
- *Syntax:* Optionales Feld. Geben Sie hier Text ein, um die Syntax der Funktion genau zu beschreiben (z.B. die erwarteten Parameter). Dieser Text wird im Fenster **Bibliotheken** neben der Funktion angezeigt. Er hat keine Auswirkung auf die Implementierung der Funktion.
- *Detail:* Optionales Feld. Diese Beschreibung wird angezeigt, wenn Sie im Fenster **Bibliotheken** oder an anderen Stellen den Cursor über die Funktion platzieren.
- *Inline-Verwendung:* Aktivieren Sie dieses Kontrollkästchen, wenn die Funktion als Inline-Funktion erstellt werden soll. Nähere Informationen dazu finden Sie weiter unten unter *Reguläre benutzerdefinierte Funktionen und Inline-Funktionen*.

3. Klicken Sie auf **OK.** Die Funktion wird daraufhin sofort im Fenster **Bibliotheken** unter dem oben angegebenen Bibliotheksnamen angezeigt. Das Mapping-Fenster wird nun neu gezeichnet, damit Sie eine neue Funktion erstellen können (es handelt sich hierbei um ein eigenständiges Mapping, das so genannte *Funktionsmapping*). Das Funktionsmapping enthält standardmäßig eine Ausgabekomponente.

4. Fügen Sie alle erforderlichen Komponenten zum Funktionsmapping hinzu. Gehen Sie dabei auf die gleiche Weise vor, wie in einem Standard-Mapping.

Um die benutzerdefinierte Funktion in einem Mapping zu verwenden, ziehen Sie diese aus dem Fenster **Bibliotheken** in den Hauptmapping-Bereich. Siehe auch *Aufrufen und Importieren von benutzerdefinerten Funktionen* wieder unten.

Benutzerdefinierte Funktionen anhand bestehender Komponenten
Um eine benutzerdefinierte Funktion anhand bestehender Komponenten zu erstellen, gehen Sie folgendermaßen vor:

1. Wählen Sie die gewünschten Komponenten im Mapping aus, indem Sie mit der Maus ein Rechteck aufziehen. Sie können die Komponenten auch durch Anklicken bei gedrückter **Strg**-Taste auswählen.
2. Wählen Sie den Menübefehl **Funktion | Benutzerdefinierte Funktion von Auswahl erstellen**. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche .
3. Gehen Sie vor, wie in Schritt 2-4 unter *Neuerstellung einer benutzerdefinierten Funktion* beschrieben.

Reguläre benutzerdefinierte Funktionen und Inline-Funktionen
Es gibt zwei Arten von benutzerdefinierten Funktionen: *Reguläre Funktionen* und *Inline-Funktionen*. Sie können beim Erstellen der Funktion den Typ Ihrer benutzerdefinierten Funktion definieren. Inline-Funktionen und reguläre Funktionen verhalten sich in Bezug auf die Codegenerierung, Rekursivität und die Möglichkeit, mehrere Output-Parameter zu haben, unterschiedlich. Die Tabelle unten enthält eine Übersicht über die wichtigsten Unterschiede zwischen regulären benutzerdefinierten Funktionen und inline gesetzten benutzerdefinierten Funktionen.

Inline-Funktionen (strichlierter Rand)	**Reguläre Funktionen (durchgezogener Rand)**
Bei Inline-Funktionen wird der Code der benutzerdefinierten Funktion an allen Stellen, an denen die Funktion aufgerufen wird, eingefügt. Wenn die benutzerdefinierte Funktion mehrmals aufgerufen wird, würde der generierte Programmcode beträchtlich länger.	Der Code für die benutzerdefinierte Funktion wird einmal aufgerufen und die Inputs dazu werden als Parameterwerte übergeben. Wenn die benutzerdefinierte Funktion mehrmals aufgerufen wird, wird sie jedes Mal mit den entsprechenden Parameterwerten ausgewertet.
Inline-Funktionen können mehrere Outputs haben und daher als Resultat mehrere Werte haben.	Reguläre Funktionen können nur einen Output haben. Um mehrere Werte zurückzugeben, können Sie den Output als complexType (z.B. als XML-Struktur) deklarieren. Auf diese Art könnten mehrere Werte an die aufrufende Komponente zurückgegeben werden.
Inline-Funktionen können nicht rekursiv aufgerufen werden.	Reguläre Funktionen können rekursiv aufgerufen werden.
Bei Inline-Funktionen kann kein Prioritätskontext[1344] für einen Parameter definiert werden.	Bei regulären Funktionen kann ein Prioritätskontext für einen Parameter definiert werden.

© 2016-2022 Altova GmbH

Anmerkung: Das Umschalten einer benutzerdefinierten Funktion zwischen "regulär" und "inline" kann sich auf den Mapping-Kontext[1333] auswirken, wodurch im Mapping ein anderes Ergebnis erzeugt werden kann.

Aufrufen und Importieren von benutzerdefinierten Funktionen

Nachdem Sie eine benutzerdefinierte Funktion erstellt haben, können Sie diese entweder vom selben Mapping, in dem Sie diese erstellt haben, oder von einem beliebigen anderen Mapping aus aufrufen.

Aufrufen von benutzerdefinierten Funktionen über dasselbe Mapping
Um eine benutzerdefinierte Funktion vom selben Mapping aus aufzurufen, gehen Sie folgendermaßen vor:

1. Suchen Sie die gewünschte Funktion im Fenster **Bibliotheken** unter der Bibliothek, die Sie bei der Erstellung der Funktion definiert haben. Geben Sie dazu die ersten Buchstaben des Namens in der Fenster **Bibliotheken** ein.
2. Ziehen Sie die Funktion aus dem Fenster **Bibliotheken** in das Mapping. Sie können nun alle erforderlichen Parameter damit verbinden.

Importieren einer benutzerdefinierten Funktion aus einem anderen Mapping
Um eine benutzerdefinierte Funktion aus einem anderen Mapping zu importieren, gehen Sie folgendermaßen vor:

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken[34] auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).



2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

3. Navigieren Sie zur **`.mfd`**-Datei, die die benutzerdefinierte Funktion enthält und klicken Sie auf **Öffnen**. Daraufhin erscheint ein Meldungsfeld, in dem Sie darüber informiert werden, dass eine neue Bibliothek hinzugefügt wurde und über das Fenster **Bibliotheken** aufgerufen werden kann.

Sie können nun jede der importierten Funktionen im aktuellen Mapping verwenden, indem Sie die Funktion aus dem **Bibliotheksfenster** in das Mapping ziehen. Nähere Informationen zum Anzeigen und Organisieren von Funktionsbibliotheken finden Sie unter Verwalten von Funktionsbibliotheken[946].

Mapping mit Anmeldeinformationen (Enterprise Edition)
Wenn die importierte **.mfd**-Datei Anmeldeinformationen enthält, werden diese im Anmeldeinformationen-Manager als importiert (gelb hinterlegt) angezeigt. Importierte Anmeldeinformationen werden standardmäßig nicht mit dem Mapping gespeichert. Sie können jedoch optional eine lokale Kopie anlegen und diese im Mapping speichern, siehe auch Anmeldeinformationen in UDFs 1358.

Bearbeiten von benutzerdefinierten Funktionen

Um eine benutzerdefinierte Funktion zu bearbeiten, gehen Sie folgendermaßen vor:

1. Öffnen Sie das Mapping, das die benutzerdefinierte Funktion enthält.
2. Doppelklicken Sie im Mapping auf die Titelleiste der benutzerdefinierten Funktion, um den Inhalt der Funktion zu sehen. Sie können Komponenten darin je nach Bedarf hinzufügen, bearbeiten oder entfernen.
3. Um die Eigenschaften der Funktion (wie Name oder Beschreibung) zu ändern, klicken Sie mit der rechten Maustaste in einen leeren Bereich des Mappings und wählen Sie im Kontextmenü den Befehl **Funktionseinstellungen**. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche .

Sie können eine Funktion auch durch Doppelklick auf ihren Namen im Fenster **Bibliotheken** bearbeiten, doch können nur Funktionen im derzeit aktiven Dokument auf diese Art geöffnet werden. Wenn Sie auf eine benutzerdefinierte Funktion doppelklicken, die in einem anderen Mapping erstellt wurde, so wird dieses Mapping in einem neuen Fenster geöffnet. Wenn Sie eine benutzerdefinierte Funktion, die in mehrere Mappings importiert wurde, bearbeiten oder löschen, so wirkt sich diese Änderung auf alle diese Mappings aus.

Um zum Hauptmapping zurückzugelangen, klicken Sie in der linken oberen Ecke des Mapping-Fensters auf die Schaltfläche .

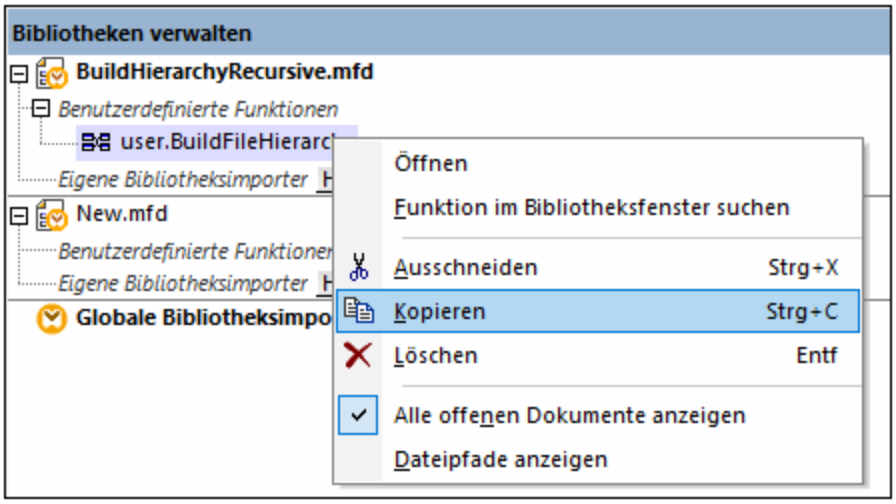
Mit Hilfe der Symbolleisten-Schaltflächen  und  können Sie in MapForce durch verschiedene Mappings und benutzerdefinierte Funktionen navigieren. Die Tastaturkürzel für diese Schaltflächen sind **Alt+nach links** bzw. **Alt+nach rechts**.

Kopieren und Einfügen von benutzerdefinierten Funktionen

Um eine benutzerdefinierte Funktion zu kopieren und in ein anderes Mapping einzufügen, gehen Sie folgendermaßen vor:

1. Öffnen Sie das Fenster "Bibliotheken verwalten" 946.
2. Klicken Sie mit der rechten Maustaste auf einen leeren Bereich im Fenster **Bibliotheken** und wählen Sie die Option **Alle offenen Dokumente anzeigen**.
3. Öffnen Sie das Quell- und das Zielmapping. Beide Mappings müssen unbedingt gespeichert worden sein, damit die Pfade korrekt aufgelöst werden. Siehe auch Kopieren-Einfügen und relative Pfade 85.
4. Klicken Sie mit der rechten Maustaste im Fenster **Bibliotheken verwalten** auf die gewünschte benutzerdefinierte Funktion aus dem Quellmapping und wählen Sie im Kontextmenü den Befehl **Kopieren** (*siehe Abbildung unten*) oder drücken Sie **Strg+C**. Lassen Sie des Fenster **Bibliotheken verwalten** geöffnet.

© 2016-2022 Altova GmbH



5. Wechseln Sie in das Zielmapping (das Fenster **Bibliotheken verwalten** ändert sich entsprechend), klicken Sie mit der rechten Maustaste auf *Benutzerdefinierte Funktionen* und wählen Sie im Kontextmenü den Befehl **Einfügen**.

Löschen von benutzerdefinierten Funktionen

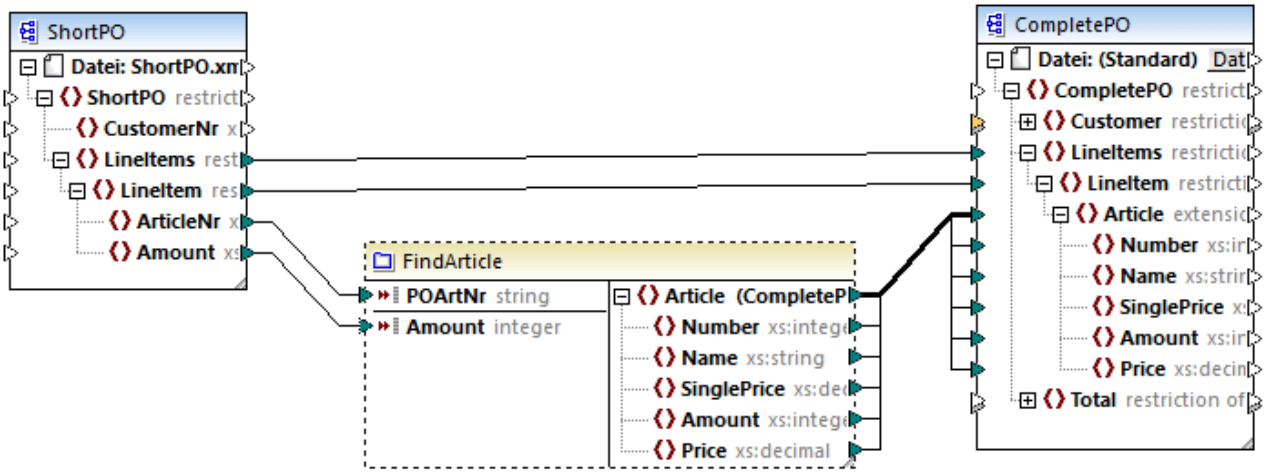
Um eine benutzerdefinierte Funktion zu löschen, gehen Sie folgendermaßen vor:

1. Doppelklicken Sie im Mapping auf die Titelleiste der Komponente.
2. Klicken Sie in der rechten oberen Ecke des Mappingfensters auf die Schaltfläche ERASE.
3. Falls die Funktion im aktuell geöffneten Mapping verwendet wird, werden Sie gefragt, ob alle Instanzen mit der internen Komponente gelöscht werden sollen. Klicken Sie auf **Ja**, wenn die Funktion gelöscht werden soll und alle Instanzen, in denen diese aufgerufen wird, durch die Komponenten der Funktion ersetzt werden sollen. Auf diese Art bleibt das Hauptmapping gültig, auch wenn die Funktion gelöscht wird. Wenn die gelöschte Funktion jedoch in anderen externen Mappings verwendet wird, werden diese ungültig. Klicken Sie auf **Nein**, wenn Sie die Funktion und alle ihre internen Komponenten permanent löschen möchten. In diesem Fall werden alle Mappings, in denen die Funktion verwendet wird, ungültig.

6.4.2 Parameter von benutzerdefinierten Funktionen

Wenn Sie eine benutzerdefinierte Funktion erstellen, müssen Sie angeben, welche Input-Parameter (falls überhaupt) diese erhalten soll und welchen Output die Funktion erzeugen soll. Während Input-Parameter manchmal nicht benötigt werden, ist ein Output-Parameter in jedem Fall erforderlich.. Funktionsparameter können einen simpleType (wie z.B. `String` oder `Ganzzahl`) oder eine komplexe Struktur[974] haben. So hat etwa die unten gezeigte benutzerdefinierte Funktion `FindArticle` zwei Input- und einen Output-Parameter:

- `POArtNr` ist ein Input-Parameter vom simpleType `string`.
- `Amount` ist ein Input-Parameter vom Typ `integer`.
- `CompletePO` ist ein Output-Parameter mit einer komplexen XML-Struktur.



Parameterreihenfolge

Wenn eine benutzerdefinierte Funktion mehrere Input- oder Output-Parameter hat, können Sie die Reihenfolge ändern, in der die Parameter für aufrufende Komponenten dieser Funktion erscheinen. Die Reihenfolge der Parameter im Funktionsmapping (von oben nach unten) bestimmt die Reihenfolge, in der diese für aufrufende Komponenten dieser Funktion angezeigt werden.

Achtung

- Input- und Output-Parameter werden nach ihrer Position von oben nach unten gereiht. Wenn Sie also den Parameter `input3` im Funktionsmapping an die oberste Stelle verschieben, wird dieser zum ersten Parameter dieser Funktion.
- Wenn zwei Parameter dieselbe vertikale Position haben, so wird zuerst der am weitesten links liegende Parameter verarbeitet.
- In den seltenen Fällen, in denen zwei Parameter genau die gleiche Position haben, wird automatisch die interne Komponenten-ID verwendet.

Strukturen vom Typ "complexType"

In der Liste unten sehen Sie, auf welchen Strukturen ein Parameter in einer benutzerdefinierten Funktion basieren kann.

MapForce Basic Edition
- XML-Schema-Struktur

MapForce Professional Edition
- XML-Schema-Struktur
- Datenbankstruktur

MapForce Enterprise Edition
- XML-Schema-Struktur
- Datenbankstruktur
- EDI-Struktur
- FlexText Structure
- JSON-Schema-Struktur

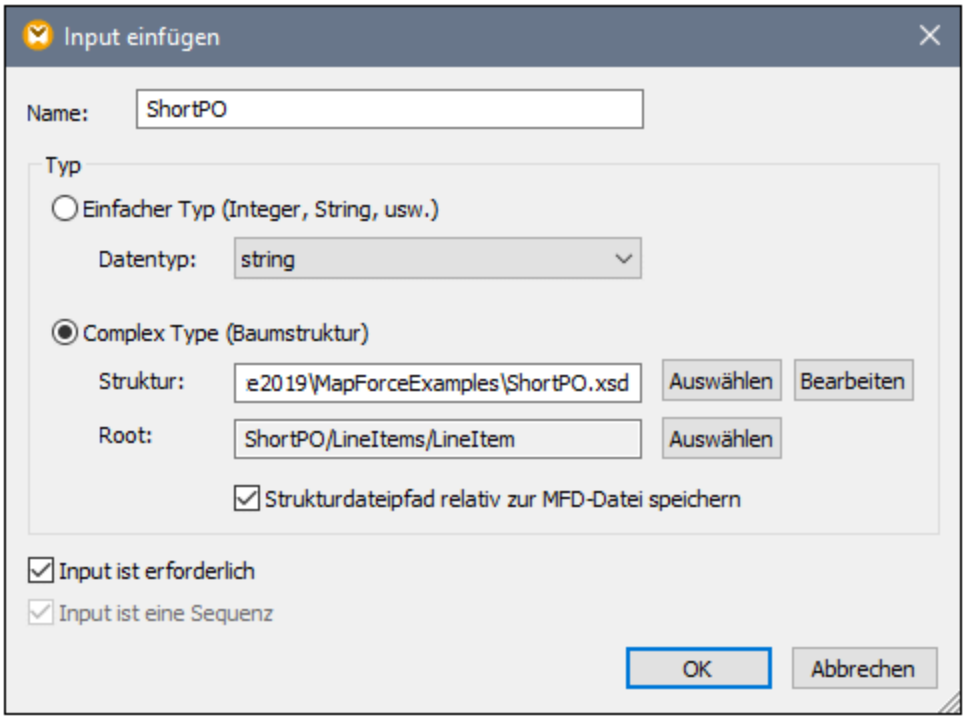
© 2016-2022 Altova GmbH

Auf Datenbankstrukturen basierende benutzerdefinierte Funktionen (Professional und Enterprise Edition)
Sie können in MapForce datenbankbasierte Parameter von benutzerdefinierten Funktionen mit einer Struktur damit in Zusammenhang stehender Tabellen erstellen und verwenden. Die Struktur dieser Tabellen bildet eine speicherresidente Struktur, die keine Verbindung zur Datenbank zur Laufzeit hat. Das heißt auch, dass es keine automatische Behandlung von Sekundärschlüsseln und keine Tabellenaktionen in Parametern oder Variablen gibt.

Hinzufügen von Parametern

Um einen Input- oder Output-Parameter hinzuzufügen, gehen Sie folgendermaßen vor:

1. Erstellen Sie eine benutzerdefinierte Funktion[968] oder öffnen Sie eine vorhandene[971].
2. Wählen Sie den Menübefehl **Funktion | Input-Komponente einfügen** bzw. **Funktion | Output-Komponente einfügen** (*siehe Abbildung unten*). Klicken Sie alternativ dazu in der Symbolleiste auf die Schaltfläche (**Input-Komponente einfügen**) oder (**Output-Komponente einfügen**).



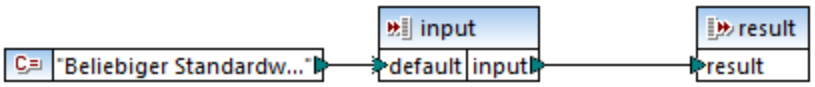
3. Wählen Sie beim Input- bzw. Output-Parameter um einen simpleType oder einen complexType handeln soll (*siehe Dialogfeld oben*). Siehe die Liste der verfügbaren komplexen Strukturen weiter oben. Um einen Parameter zu erstellen, der ein komplexer XML-Typ ist, klicken Sie neben *Struktur* auf **Auswählen** und navigieren Sie zum XML-Schema, das die erforderliche Struktur beschreibt.

Wenn das Funktionsmapping bereits XML-Schemas enthält, stehen sie als Strukturen zur Auswahl. Andernfalls können Sie ein komplett neues Schema für die Struktur des Parameters auswählen. Dasselbe gilt für Datenbanken und andere komplexe Strukturen, falls diese von Ihrer MapForce Edition unterstützt werden. Sie können bei XML-Strukturen ein Root-Element für Ihre Struktur auswählen, wenn es das XML-Schema erlaubt. Um ein Root-Element zu definieren, klicken Sie neben *Root* auf **Auswählen** und wählen Sie die Root im daraufhin angezeigten Dialogfeld aus.

Falls das Kontrollkästchen *Strukturdateipfad relativ zur MFD-Datei speichern* aktiviert ist, wird der absolute Pfad der Strukturdatei beim Speichern des Mappings in einen Pfad umgewandelt, der relativ zum aktuellen Mapping ist. Nähere Informationen dazu finden Sie unter Relative und absolute Pfade 85. Eine Erläuterung zu den Kontrollkästchen *Input ist erforderlich* und *Input ist eine Sequenz* finden Sie weiter unten.

Input ist erforderlich
Um einen Parameter in einer benutzerdefinierten Funktion zu einem obligatorischen Parameter zu machen, aktivieren Sie das Kontrollkästchen *Input ist erforderlich* (*siehe Dialogfeld oben*). Wenn Sie das Kontrollkästchen *Input ist erforderlich* deaktivieren, wird der Parameter optional und im Mapping mit einem strichlierten Rahmen angezeigt.

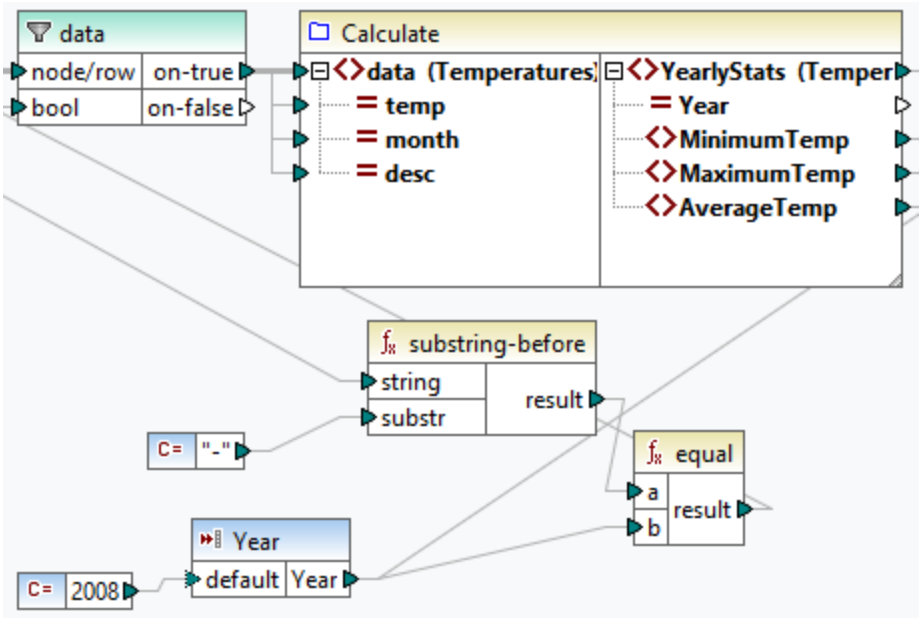
Außerdem können Sie einen Standardparameterwert definieren, indem Sie ihn mit dem Input "`default`" eines Parameters verbinden (*siehe Beispiel unten*). Der Standardwert wird nur wirksam, wenn kein anderer Wert vorhanden ist. Wenn der optionale Parameter bei Aufruf der Funktion einen Wert erhält, so hat dieser Wert Vorrang vor dem Standardwert.



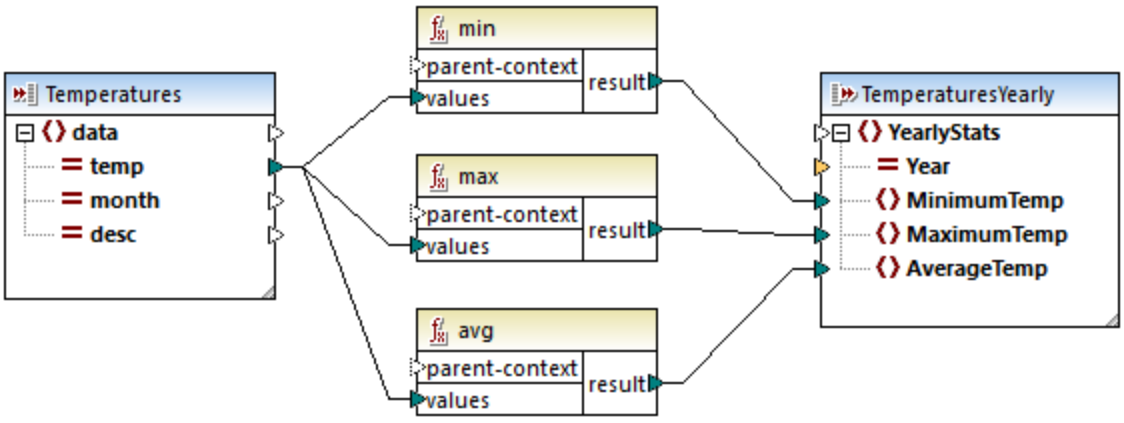
Input ist eine Sequenz
Sie können optional festlegen, ob ein Funktionsparameter als Einzelwert (Standardverhalten) oder als Sequenz behandelt werden soll. Eine Sequenz ist ein Bereich von null oder mehr Werten. Eine Sequenz kann sich als nützlich erweisen, wenn in Ihrer benutzerdefinierten Funktion Input-Daten in Form einer Sequenz erwartet werden, um in dieser Sequenz Werte zu berechnen (z.B. durch Aufruf von Funktionen wie **`avg`**, **`min`**, **`max`**. Damit der Input des Parameters als Sequenz behandelt wird, aktivieren Sie das Kontrollkästchen *Input ist eine Sequenz*. Beachten Sie, dass dieses Kontrollkästchen nur dann aktiv ist, wenn es sich um eine reguläre 970 benutzerdefinierte Funktion handelt.

Ein Beispiel für die Verwendung einer Sequenz sehen Sie im folgenden Mapping: **`MapForceExamples\InputIsSequence.mfd`**. In dem Ausschnitt aus diesem Mapping (*siehe Abbildung unten*) ist der Filter `data` mit der benutzerdefinierten Funktion `Calculate` verbunden. Das Ergebnis des Filters ist eine Sequenz von Datenelementen, sodass der Input-Parameter der Funktion als Sequenz definiert wurde.

© 2016-2022 Altova GmbH



Unten sehen Sie die Implementierung der Funktion `Calculate`, die alle Sequenzwerte aggregiert: Sie führt, die Funktionen **`avg`**, **`min`**, **`max`** an der Input-Sequenz aus. Um die interne Struktur der Funktion `Calculate` zu sehen, doppelklicken Sie im Mapping oben auf die Überschrift der `Calculate`-Komponente.



Im Allgemeinen gilt, dass es von den Input-Daten, bei denen es sich um eine Sequenz oder nicht um eine Sequenz handelt, abhängt, wie oft die Funktion aufgerufen wird:

- Wenn Input-Daten mit einem *Sequenzparameter* verbunden sind, wird die benutzerdefinierte Funktion nur *einmal* aufgerufen und die vollständige Sequenz wird an die benutzerdefinierte Funktion übergeben.
- Wenn Input-Daten mit einem *Nicht-Sequenzparameter* verbunden sind, wird die benutzerdefinierte Funktion *für jedes Datenelement in der Sequenz einmal* aufgerufen.
- Wenn Sie eine leere Sequenz mit einem Nicht-Sequenz-Parameter verbinden, wird die Funktion gar nicht aufgerufen. Dies kann vorkommen, wenn die Quellstruktur optionale Datenelemente hat oder wenn eine Filterbedingung keine übereinstimmenden Datenelemente zurückgibt. Um dies zu vermeiden, verwenden Sie vor dem Funktions-Input entweder die Funktion `substitute-missing`[1103],

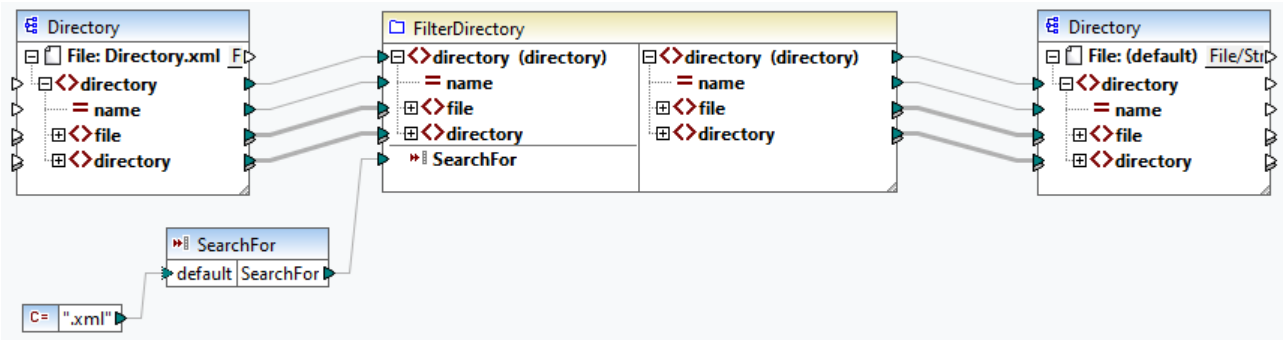
um sicherzustellen, dass die Sequenz nie leer ist oder setzen Sie den Parameter auf Sequenz und fügen Sie innerhalb der Funktion eine Behandlung für die leere Sequenz hinzu.

Das Kontrollkästchen *Output ist eine Sequenz* kann auch für Output-Parameter erforderlich sein. Wenn eine Funktion eine Sequenz aus mehreren Werten an ihre Output-Komponente übergibt und die Output-Komponente nicht auf Sequenz gesetzt ist, gibt die Funktion nur das erste Datenelement in der Sequenz zurück.

6.4.3 Rekursive benutzerdefinierte Funktionen

In diesem Beispiel wird beschrieben, wie Sie mit Hilfe einer rekursiven benutzerdefinierten Funktion nach Daten in einer XML-Quelldatei suchen. Um die rekursive benutzerdefinierte Funktion auszuprobieren, benötigen Sie das folgende Mapping: **`MapForceExamples\RecursiveDirectoryFilter.mfd`**. Im unten gezeigten Mapping erhält die benutzerdefinierte Funktion `FilterDirectory` Daten aus der Quelldatei `Directory.xml` und der einfachen Input-Komponente `SearchFor`, die die `.xml`-Erweiterung liefert. Nachdem die Daten von der benutzerdefinierten Funktion verarbeitet wurden, werden sie auf die Zieldatei gemappt.

Im Hauptmapping (*siehe Abbildung unten*) ist das allgemeine Mapping-Layout beschrieben. Wie die benutzerdefinierte Funktion die Daten verarbeitet, ist separat im Funktionsmapping definiert (*siehe UDF-Implementierung weiter unten*).



Aufgabenstellung
Ziel ist es, Dateien mit der Erweiterung **`.xml`** in der Ausgabe aufzulisten, wobei die gesamte Verzeichnisstruktur erhalten bleiben soll. In den folgenden Unterabschnitten wird das Mapping im Detail erläutert.

Quelldatei

Unten sehen Sie einen Ausschnitt aus der XML-Quelldatei (**`Directory.xml`**), die Informationen über Dateien und Verzeichnisse enthält. Beachten Sie, dass die Dateien in dieser Liste unterschiedliche Erweiterungen haben (z.B. **`.xml`**, **`.dtd`**, **`.sps`**). Gemäß dem Schema (**`Directory.xsd`**), kann das Element `directory` `file`- und `directory`-Children haben. Alle `directory`-Elemente sind rekursiv.

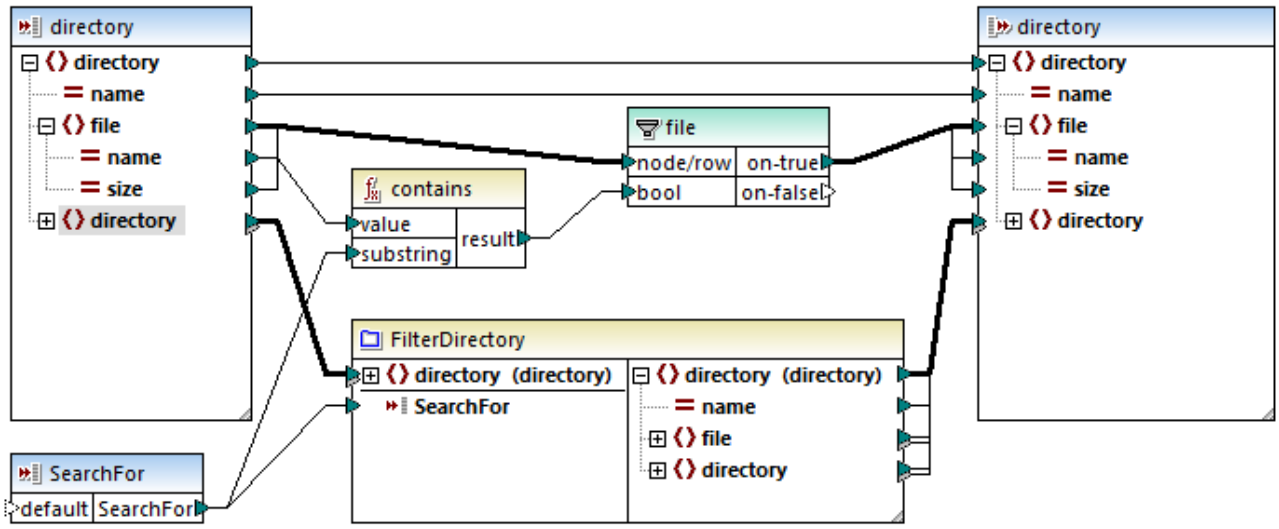
```
<directory name="ExampleSite">
	<file name="blocks.sps" size="7473"/>
	<file name="blocks.xml" size="670"/>
	<file name="block_file.xml" size="992"/>
	<file name="block_schema.xml" size="1170"/>
	<file name="contact.xml" size="453"/>
	<file name="dictionaries.xml" size="206"/>
```


© 2016-2022 Altova GmbH

```
        <file name="examplesite.dtd" size="230"/>
        <file name="examplesite.spp" size="1270"/>
        <file name="examplesite.sps" size="20968"/>
        ...
        <directory name="output">
               <file name="examplesite1.css" size="3174"/>
               <directory name="images">
                      <file name="blank.gif" size="88"/>
                      <file name="block_file.gif" size="13179"/>
                      <file name="block_schema.gif" size="9211"/>
                      <file name="nav_file.gif" size="60868"/>
                      <file name="nav_schema.gif" size="6002"/>
               </directory>
        </directory>
</directory>
```

Implementierung der benutzerdefinierten Funktion

Um die interne Implementierung der benutzerdefinierten Funktion zu sehen, doppelklicken Sie im Hauptmapping auf ihre Titelleiste. Die Funktion ist rekursiv, d.h. sie ruft sich selbst auf. Da die Funktion mit dem rekursiven Element `directory` verbunden ist, wird diese Funktion so oft aufgerufen, wie sich in der XML-Quellinstanz verschachtelte `directory`-Elemente befinden. Damit rekursive Aufrufe unterstützt werden, muss die Funktion regulär[970] sein.



Die Implementierung der benutzerdefinierten Funktion erfolgt in zwei Phasen: (i) der Definition der Dateien und (ii) der Definition der zu durchsuchenden Verzeichnisse.

Definition der Dateien

Die Dateien werden von der benutzerdefinierten Funktion folgendermaßen verarbeitet: Die Funktion **`contains`** überprüft, ob der erste String (der Dateiname) den (von der einfachen Input-Komponente `SearchFor`) gelieferten Substring `.xml` enthält. Wenn das Ergebnis der Funktion `true` ist, wird der Dateiname mit einer `.xml`-Erweiterung in die Ausgabe geschrieben.

Verarbeitung von untergeordneten Verzeichnissen
Die untergeordneten Verzeichnisse des aktuellen Verzeichnisses werden als Input an die aktuelle benutzerdefinierte Funktion gesendet. Die benutzerdefinierte Funktion iteriert somit durch alle `directory`-Elemente und überprüft, ob Dateien mit der Erweiterung **`.xml`** vorhanden sind.

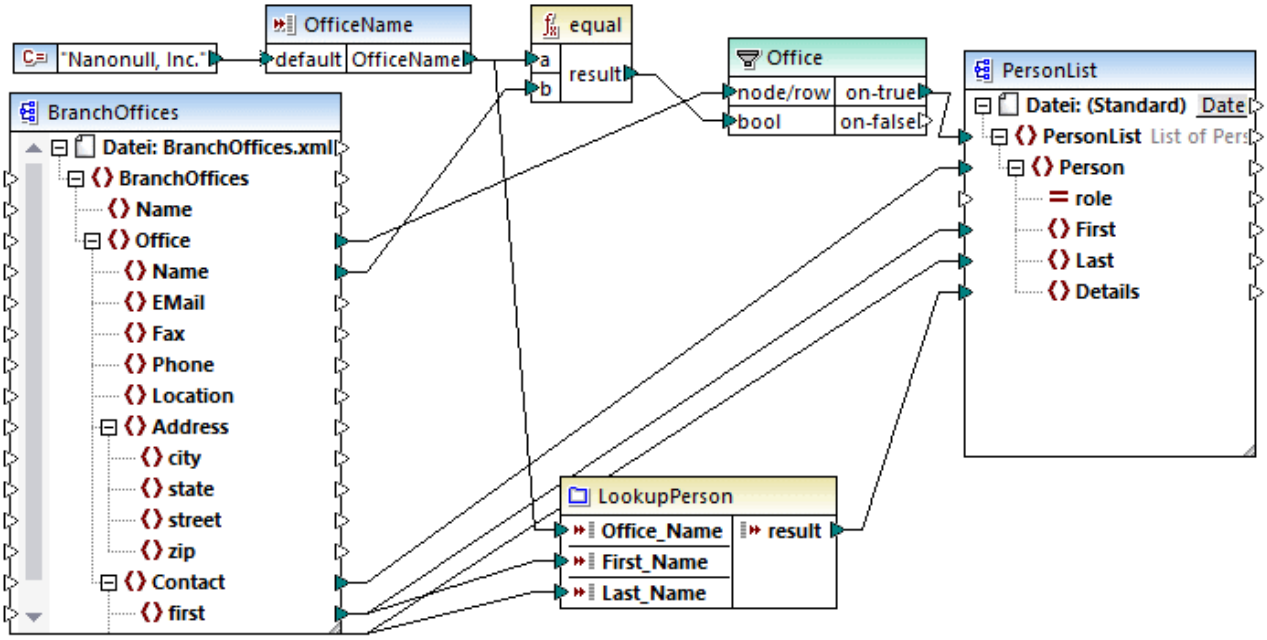
Ausgabe

Wenn Sie auf das Fenster **Ausgabe** klicken, werden nur Dateien mit der Erweiterung **`.xml`** angezeigt (*siehe Codefragment unten*).

```
<directory name="ExampleSite">
	<file name="blocks.xml" size="670"/>
	<file name="block_file.xml" size="992"/>
	<file name="block_schema.xml" size="1170"/>
	<file name="contact.xml" size="453"/>
	...
	<directory name="output">
		<directory name="images"/>
	</directory>
</directory>
```

6.4.4 Look-up-Implementierung

In diesem Kapitel wird erläutert, wie Sie Mitarbeiterdaten abrufen (Look-up) und auf geeignete Weise darstellen können. Um die Look-up-Implementierung auszuprobieren, benötigen Sie das folgende Mapping: **`MapForceExamples\PersonListByBranchOffice.mfd`**.



Aufgabenstellung
Wir möchten Folgendes erreichen:

 © 2016-2022 Altova GmbH

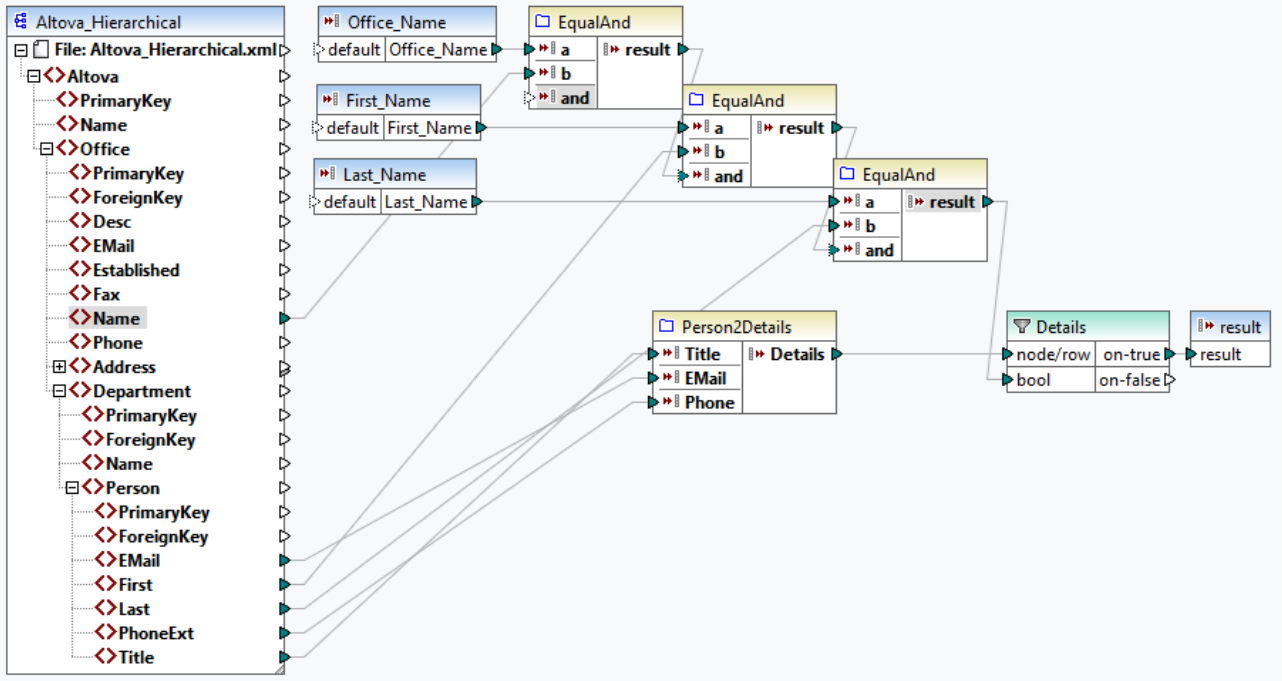
- Suchen bestimmter Daten über die einzelnen Mitarbeiter (Telnr., E-Mail-Adresse, Position) mittels Look-up in einer separaten XML-Datei.
- Darstellung dieser Daten in Form einer kommagetrennten Liste und Mappen dieser Liste auf das Element `Details` der XML-Zieldatei.
- Extraktion von Informationen über Mitarbeiter nur aus der Zweigniederlassung namens Nanonull, Inc.

Wir haben unser Mapping zu diesem Zweck folgendermaßen erstellt:

- Mit Hilfe des Filters `Office` werden nur die Mitarbeiter aus Nanonull, Inc. herausgefiltert.
- Um mittels Look-up Informationen aus einer anderen XML-Datei abzurufen, wird im Mapping die benutzerdefinierte Funktion `LookupPerson` aufgerufen. Die Implementierung dieser benutzerdefinierten Funktion wird im folgenden Unterabschnitt beschrieben.
- Zur Verarbeitung der Mitarbeiterdaten ruft die Funktion `LookupPerson` intern weitere Funktionen auf, die Informationen über die einzelnen Mitarbeiter abrufen und miteinander verketten. All diese Operationen befinden sich im Mapping dieser Funktion und sind im Hauptmapping nicht sichtbar. Mit Hilfe der Funktion `LookupPerson` werden anschließend die Mitarbeiterdaten auf das Datenelement `Details` in `PersonList` gemappt.

Implementierung von LookupPerson

Die Look-up-Funktionalität wird über die Funktion `LookupPerson`, deren Definition Sie unten sehen, bereitgestellt. Um die interne Implementierung der benutzerdefinierten Funktion zu sehen, doppelklicken Sie im Hauptmapping auf deren Titelleiste.



Die benutzerdefinierte Funktion ist folgendermaßen definiert:

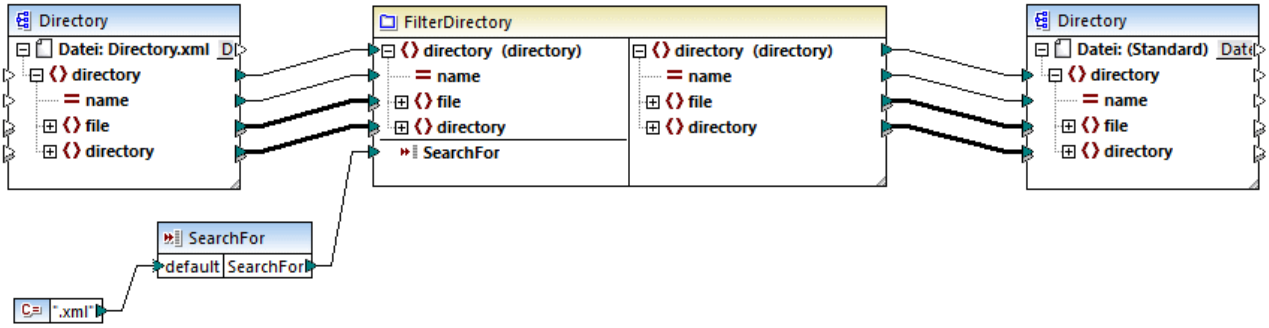
- Die Daten werden aus der XML-Datei **`Altova_Hierarchical.xml`** abgerufen: (i) der Name des Büros und die Vor- und Nachnamen der Mitarbeiter, anhand derer Mitarbeiter nur aus Nanonull, Inc. ausgewählt werden und (ii) die E-Mail, der Titel und die Durchwahl, die zu einem String verkettet

werden. Unten finden Sie eine Beschreibung der Definition der Funktionen `EqualAnd` und `Person2Detail`.

- Außerdem hat die benutzerdefinierte Funktion drei Input-Parameter, die die Look-up-Werte `Office_Name`, `First_Name` und `Last_Name` bereitstellen. Der Wert des Parameters `Office_Name` wird aus dem `OfficeName`-Input aus dem Hauptmapping abgerufen und die Werte von `First_Name` und `Last_Name` stammen aus der Komponente `BranchOffices` aus dem Hauptmapping.
- Der Wert der Funktion `EqaulAnd` (`true` oder `false`) wird jedes Mal, wenn die Daten eines neuen Mitarbeiters (title, email, phone) verarbeitet werden, an den `Details`-Filter übergeben. Wenn der `Details`-Filter den Wert `true` erhält, ist die Look-up-Operation erfolgreich und die Mitarbeiterdaten werden abgerufen und an das Hauptmapping übergeben. Andernfalls wird das nächste Datenelement im Kontext überprüft usw., bis die Schleife fertig verarbeitet ist.

EqualAnd-Implementierung

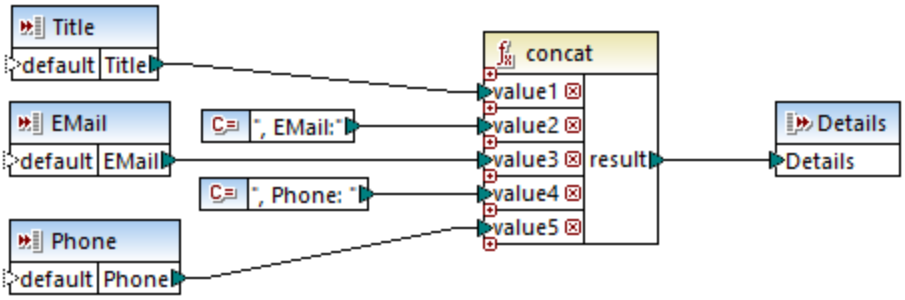
Die `EqualAnd`-Funktion (*siehe unten*) ist eine separate innerhalb der `LookupPerson`-Funktion definierte benutzerdefinierte Funktion. Die interne Struktur der benutzerdefinierten Funktion `EqualAnd` sehen Sie bei Doppelklick auf die Überschrift der Funktion.



Die benutzerdefinierte Funktion `EqualAnd` überprüft zuerst, ob `a` gleich `b` ist; wenn das Ergebnis `true` ist, wird es als erster Parameter der **`logical-and`**-Funktion übergeben. Wenn beide Werte in der **`logical-and`**-Funktion "true" sind, ist das Ergebnis ebenfalls "true" und wird an die nächste `EqualAnd`-Funktion übergeben. Das Ergebnis der dritten `EqaulAnd`-Funktion (*siehe LookupPerson Funktion oben*) wird an den `Details`-Filter übergeben.

Person2Detail-Implementierung

Eine weitere in die benutzerdefinierte Funktion `LookupPerson` verschachtelte Funktion ist die Funktion `Person2Details`. Die benutzerdefinierte Funktion `Person2Details` (*siehe unten*) verkettet drei Werte (aus **`Altova_Hierarchical.xml`**) und zwei Textkonstanten miteinander.



© 2016-2022 Altova GmbH

Ausgabe

Wenn Sie auf das Fenster **Ausgabe** klicken, werden in MapForce nur die Vor- und Nachnamen sowie die Daten der Mitarbeiter aus Nanonull, Inc angezeigt (*siehe Codefragment unten*).

```
<PersonList>
	<Person>
		<First>Vernon</First>
		<Last>Callaby</Last>
		<Details>Office Manager, EMail:v.callaby@nanonull.com, Phone:
582</Details>
	</Person>
	<Person>
		<First>Frank</First>
		<Last>Further</Last>
		<Details>Accounts Receivable, EMail:f.further@nanonull.com, Phone:
471</Details>
	</Person>
	...
</PersonList>
```

6.5 Spezielle benutzerdefinierte Funktionen

In diesem Abschnitt wird erläutert, wie Sie benutzerdefinierte XSLT-984, XQuery-992, Java- und .NET-997 Funktionen importieren. Außerdem erfahren Sie hier, wie Sie C#, C++ und Java-Bibliotheken manuell1005 referenzieren.

6.5.1 Importieren benutzerdefinierter XSLT-Funktionen

In MapForce haben Sie die Möglichkeit, die XSLT 1.0-, XSLT 2.0-, und XSLT 3.0-Funktionsbibliothek durch Ihre eigenen benutzerdefinierten Funktionen zu ergänzen, vorausgesetzt, das Ergebnis Ihrer benutzerdefinierten Funktionen sind simpleTypes.

> Es werden nur benutzerdefinierte Funktionen unterstützt, deren Ergebnis einfache Datentypen (z.B. Strings) sind.

So importieren Sie Funktionen aus einer XSLT-Datei:

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken34 auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).



2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

3. Navigieren Sie zur .xsl-Datei, die die Funktionen enthält und klicken Sie auf **Öffnen**. Daraufhin wird ein Meldungsfeld, in dem Sie informiert werden, dass eine neue Bibliothek hinzugefügt wurde, angezeigt.

Importierte XSLT-Dateien werden im Fenster "Bibliotheken" als Bibliotheken angezeigt. Unterhalb des Bibliotheksnamens werden alle Named Templates als Funktionen angezeigt. Wenn Sie die importierte Bibliothek nicht sehen, stellen Sie sicher, dass Sie XSLT als Transformationssprache28 ausgewählt haben. Siehe Verwalten von Funktionsbibliotheken946.

© 2016-2022 Altova GmbH

Beachten Sie dazu Folgendes:

- Funktionen müssen als Named Templates deklariert werden, die der XSLT-Spezifikation in der XSLT-Datei entsprechen, damit die Funktionen in MapForce importiert werden können. Sie können auch Funktionen, die in einem XSLT 2.0-Dokument in der Form **<xsl:function name="MyFunction">** vorkommen, importieren. Wenn die importierte XSLT-Datei andere XSLT-Dateien importiert oder beinhaltet, so werden auch diese XSLT-Dateien und Funktionen importiert.
- Die mapbaren Input-Konnektoren von importierten benutzerdefinierten Funktionen hängen von der Anzahl der Parameter ab, die im Template Call verwendet werden; auch optionale Parameter werden unterstützt.
- Namespaces werden unterstützt.
- Wenn Sie Änderungen an XSLT-Dateien, die bereits in MapForce importiert wurden, vornehmen, so werden die Änderungen automatisch erkannt und MapForce fragt Sie, ob die Dateien neu geladen werden sollen.
- Stellen Sie beim Erstellen von Named Templates sicher, dass die im Template verwendeten XPath-Anweisungen an den/die richtigen Namespace(s) gebunden sind. Um die Namespace Bindings des Mappings zu sehen, zeigen Sie eine Vorschau des generierten XSLT-Codes 109 an.

Datentypen in XPath 2.0

Wenn Ihr XML-Dokument ein XML-Schema referenziert und gemäß diesem Schema gültig ist, müssen Sie Datentypen, die nicht durch eine Operation implizit in den benötigten Datentyp konvertiert werden, explizit konstruieren oder konvertieren.

Im XPath 2.0 Datenmodell, das vom Altova XSLT 2.0-Prozessor verwendet wird, wird allen **atomized** Nodewerten aus dem XML-Dokument der Datentyp `xs:untypedAtomic` zugewiesen. Der Typ `xs:untypedAtomic` funktioniert gut mit impliziten Typkonvertierungen.

Beispiel:

- Der Ausdruck `xs:untypedAtomic("1") + 1` hat als Ergebnis den Wert 2, da der Wert `xdt:untypedAtomic` *implizit* vom Plus-Zeichen in `xs:double` geändert wird.
- Arithmetische Operatoren konvertieren Operanden implizit in `xs:double`.
- Operatoren zum Vergleich von Werten konvertieren Operanden vor dem Vergleich in `xs:string`.

Siehe auch:

Beispiel: Hinzufügen benutzerdefinierter XSLT-Funktionen 985
Beispiel: Summieren von Node-Werten 988
Implementierung des XSLT 1.0-Prozessors 1988
Implementierung des XSLT 2.0-Prozessors 1989

6.5.1.1 Beispiel: Hinzufügen benutzerdefinierter XSLT-Funktionen

In diesem Beispiel wird gezeigt, wie Sie benutzerdefinierte XSLT 1.0-Funktionen in MapForce importieren. Die für dieses Beispiel benötigten Dateien befinden sich im Verzeichnis **C:\Users\<username>\Documents\Altova\MapForce2023\MapForceExamples**.

- **Name-splitter.xslt**. In dieser XSLT-Datei ist ein Named Template namens **tokenize** mit einem einzigen Parameter "string" definiert. Das Template funktioniert über einen Input-String und trennt Großbuchstaben durch ein Leerzeichen für jede Instanz.

- **Name-splitter.xml** (die zu verarbeitende XML-Quellinstanzdatei)
- **Customers.xsd** (das XML-Quellschema)
- **CompletePO.xsd** (das XML-Zielschema)

So fügen Sie eine benutzerdefinierte XSLT-Funktion hinzu:

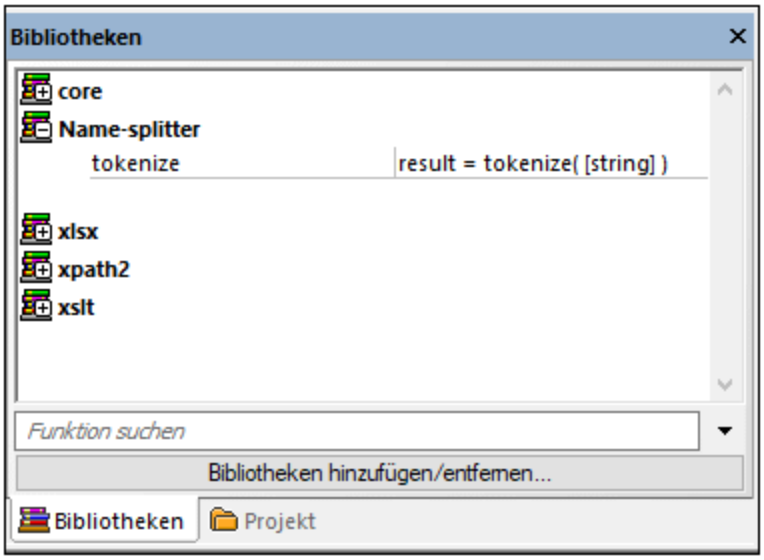
1. Klicken Sie im unteren Bereich des Fensters Bibliotheken[34] auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).



2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

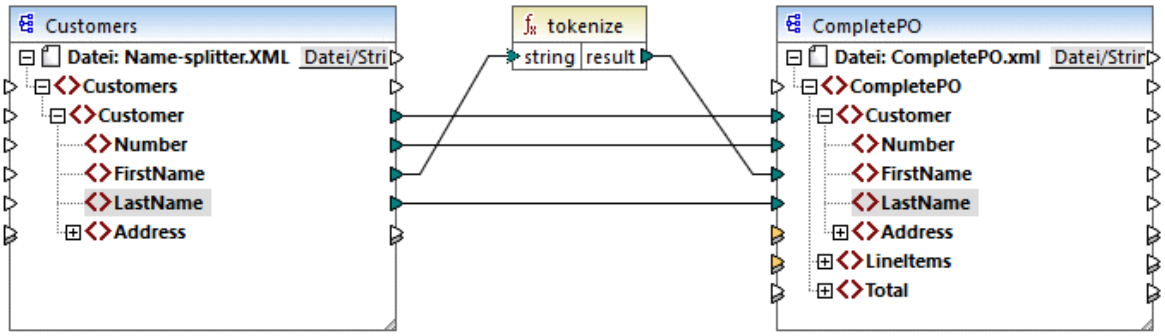
3. Navigieren Sie zur .xsl- oder .xslt-Datei, die das Named Template, das Sie als Funktion hinzufügen möchten, enthält - in diesem Fall zu **Name-splitter.xslt** und klicken Sie auf **Öffnen**. Daraufhin erscheint ein Meldungsfeld, in dem Sie darüber informiert werden, dass eine neue Bibliothek hinzugefügt wurde und der Name der XSLT-Datei sowie die als Named Templates definierten Funktionen (in diesem Beispiel **Name-splitter** mit der Funktion `tokenize`) werden im Fenster "Bibliotheken" angezeigt.

© 2016-2022 Altova GmbH



So verwenden Sie die XSLT-Funktion in Ihrem Mapping:

1. Ziehen Sie die **`tokenize`**-Funktion in das Mapping-Fenster und mappen Sie die Datenelemente wie folgt.



2. Klicken Sie auf das Register **XSLT**, um den erzeugten XSLT-Code zu sehen.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!--...-->
11 <xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xs="http
12     <xsl:include href="file:///C:/Users/altova/Documents/Altova/MapForce2020/MapForceExamples
13     <xsl:output method="xml" encoding="UTF-8" byte-order-mark="no" indent="yes"/>
14     <xsl:template match="/">
15         <CompletePO>
16             <xsl:attribute name="xsi:noNamespaceSchemaLocation" namespace="http://www.w3.org/
   CompletePO.xsd'"/>
17             <xsl:for-each select="Customers/Customer">
18                 <Customer>
19                     <Number>
20                         <xsl:sequence select="xs:string(xs:integer(fn:string(Number)))"/>
21                     </Number>
22                     <FirstName>
23                         <xsl:call-template name="tokenize">
24                             <xsl:with-param name="string" select="FirstName" as="item()"/>
25                         </xsl:call-template>
26                     </FirstName>
27                     <LastName>
28                         <xsl:sequence select="fn:string(LastName)"/>
29                     </LastName>
30                 </Customer>
31             </xsl:for-each>
32         </CompletePO>
33     </xsl:template>
34 </xsl:stylesheet>
35
```

Anmerkung: Sobald eine Named Template in einem Mapping verwendet wird, wird die XSLT-Datei, die den Namen des Named Template enthält, im generierten XSLT-Code **inkludiert** (**xsl:include href...**) und mit dem Befehl **xsl:call-template aufgerufen**.

3. Klicken Sie auf das Register **Ausgabe,** um das Ergebnis des Mappings zu sehen.

So entfernen Sie benutzerdefinierte XSLT-Funktionen aus MapForce:

1. Klicken Sie im unteren Bereich des Fensters "Bibliotheken" auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster "Bibliotheken verwalten" geöffnet.
2. Klicken Sie neben der zu löschenden XSLT-Bibliothek auf **Bibliothek löschen** .

6.5.1.2 Beispiel: Summieren von Node-Werten

In diesem Beispiel wird gezeigt, wie Sie mehrere Nodes eines XML-Dokuments verarbeiten und die Ergebnisse als einen einzigen Wert auf ein XML-Zieldokument mappen können. Ziel des Mappings ist es, den Preis aller Produkte in einer XML-Quelldatei zu berechnen und diesen als einen einzigen Wert in eine XML-Ausgabedatei zu schreiben. Die in diesem Beispiel verwendeten Dateien stehen im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** zur Verfügung.

- **Summing-nodes.mfd** -die Mapping-Datei
- **input.xml** - die XML-Quelldatei
- **input.xsd** - das XML-Quellschema
- **output.xsd** - die XML-Zieldatei
- **Summing-nodes.xslt** - ein benutzerdefiniertes XSLT-Stylesheet, das eine benannte Vorlage zum Summieren der einzelnen Nodes enthält.

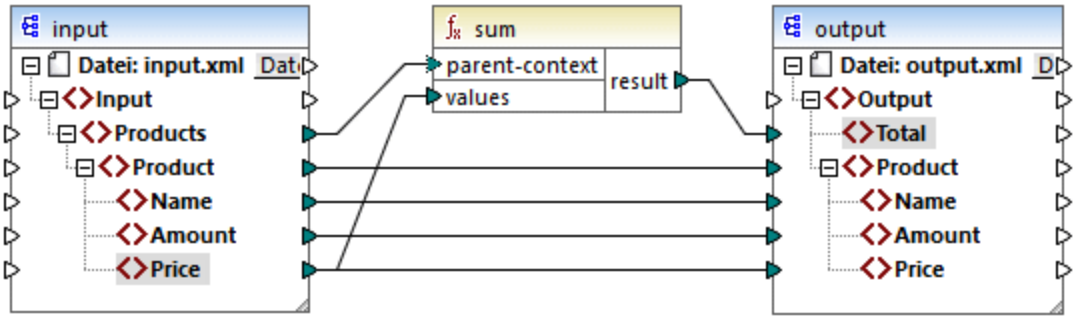
© 2016-2022 Altova GmbH

Es gibt zwei verschiedene Methoden, mit denen Sie das Ziel des Mappings erreichen können:

- Verwendung der sum[1033]-Funktion. Diese vordefinierte MapForce-Funktion steht im Fenster "Bibliotheken" zur Verfügung.
- durch Import eines benutzerdefinierten XSLT-Stylesheet in MapForce.

Lösungsvariante 1: Verwendung der Aggregatfunktion "sum"

Um die Funktion **sum** im Mapping zu verwenden, ziehen Sie sie aus dem Fenster Bibliotheken in das Mapping. Beachten Sie, dass es von der ausgewählten XSLT-Version abhängt (XSLT1 oder XSLT2), welche Funktionen im Fenster Bibliotheken zur Verfügung stehen. Erstellen Sie nun die Mapping-Verbindungen wie unten gezeigt.



Nähere Informationen zu Aggregatfunktionen der core-Bibliothek finden Sie unter core | aggregate functions (Aggregatfunktionen)[1026].

Lösungsvariante 2: Verwendung eines benutzerdefinierten XSLT-Stylesheet

Wie oben erwähnt, ist das Ziel des Beispiels, die `Price`-Felder von Produkten in der XML-Quelldatei, in diesem Fall der Produkte A und B, zu summieren.

```
<?xml version="1.0" encoding="UTF-8"?>
<Input xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="input.xsd">
   <Products>
      <Product>
         <Name>ProductA</Name>
         <Amount>10</Amount>
         <Price>5</Price>
      </Product>
      <Product>
         <Name>ProductB</Name>
         <Amount>5</Amount>
         <Price>20</Price>
      </Product>
   </Products>
</Input>
```

Im Codefragment unten sehen Sie ein benutzerdefiniertes XSLT-Stylesheet, in dem die benannte Vorlage (named template) "Total" und ein einziger Parameter `string` verwendet werden. Die Vorlage verarbeitet die XML-Input-Datei und summiert alle durch den XPath-Ausdruck `/Product/Price` bereitgestellten Werte.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>

   <xsl:template match="*">
      <xsl:for-each select=".">
      <xsl:call-template name="Total">
         <xsl:with-param name="string" select="."/>
      </xsl:call-template>
      </xsl:for-each>
   </xsl:template>

   <xsl:template name="Total">
   <xsl:param name="string"/>
      <xsl:value-of select="sum($string/Product/Price)"/>
   </xsl:template>
</xsl:stylesheet>
```

Anmerkung: Um die Nodes in XSLT 2.0 zu summieren, ändern Sie die Stylesheet-Deklaration in `version="2.0"`.

Bevor Sie das XSLT-Stylesheet in MapForce importieren, wählen Sie als Transformationssprache[28] XSLT 1.0. aus. Sie können die benutzerdefinierte Funktion nun folgendermaßen importieren:

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken[34] auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).

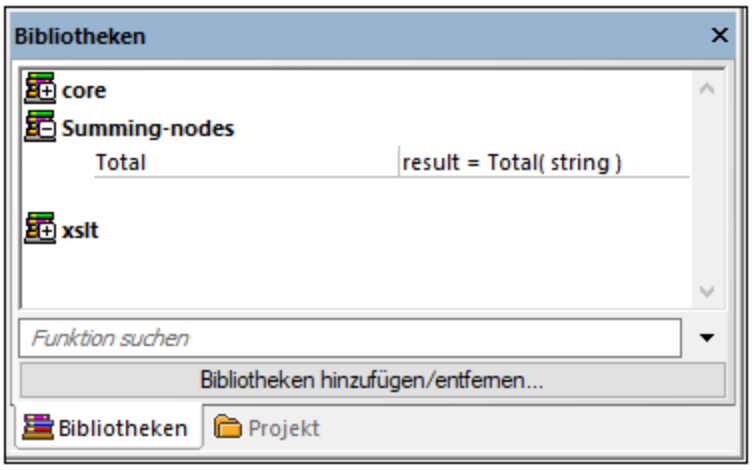


2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

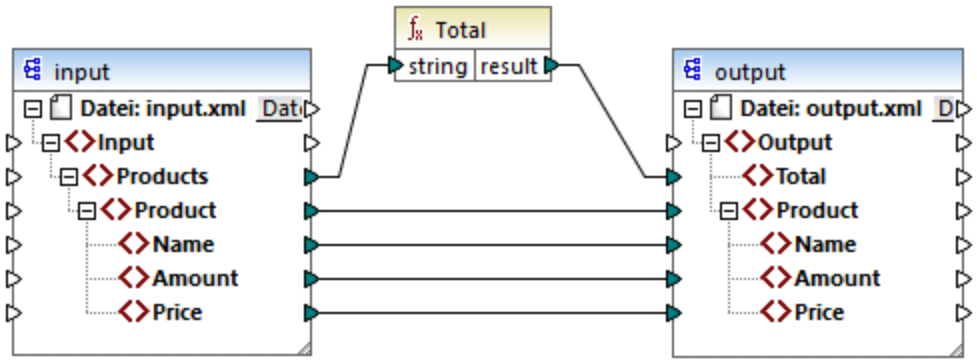
3. Navigieren Sie zur Datei **<Dokumente>\Altova\MapForce2023\MapForceExamples\Summing-nodes.xslt** und klicken Sie auf **Öffnen**. Daraufhin erscheint ein Meldungsfeld, in dem Sie darüber

 © 2016-2022 Altova GmbH

informiert werden, dass eine neue Bibliothek hinzugefügt wurde, und die neue Bibliothek wird im Fenster "Bibliotheken" angezeigt.



4. Ziehen Sie die Funktion **`Total`** aus der Bibliothek ins Mapping und erstellen Sie die Mapping-Verbindungen, wie unten gezeigt.



Um eine Vorschau auf das Mapping-Ergebnis zu sehen, klicken Sie auf das Register **Ausgabe**. Im Feld `Total` wird nun die Summe der zwei `Price`-Felder angezeigt.

```
<?xml version="1.0" encoding="UTF-8"?>
<Output xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="output.xsd">
   <Total>25</Total>
   <Product>
      <Name>ProductA</Name>
      <Amount>10</Amount>
      <Price>5</Price>
   </Product>
   <Product>
      <Name>ProductB</Name>
      <Amount>5</Amount>
      <Price>20</Price>
   </Product>
</Output>
```

6.5.2 Importieren benutzerdefinierter XQuery 1.0-Funktionen

Wenn als Mapping-Transformationssprache XQuery ausgewählt ist, werden die für XQuery verfügbaren vordefinierten Funktionsbibliotheken im Fenster "Bibliotheken" angezeigt. Falls nötig, können Sie diese Liste durch benutzerdefinierte XQuery-Funktionen erweitern, indem Sie benutzerdefinierte XQuery 1.0-Bibliotheksmodule in MapForce importieren.

Um diese Funktionen in MapForce importieren zu können, muss eine XQuery-Datei die folgenden Voraussetzungen erfüllen:

- Es muss sich gemäß der XQuery-Spezifikation um ein gültiges Bibliotheksmodul handeln, d.h. es muss am Anfang eine Moduldeklaration wie `module namespace <prefix>="<namespace name"` haben.
- Alle im importierten Bibliotheksmodul deklarierten Funktionen müssen atomare Datentypen (z.B. `Xs:string,`, `xs:boolean`, `xs:integer`, usw) zurückgeben. Auch Funktionsparameter müssen einen atomaren Typ haben.

So importieren Sie ein XQuery-Bibliotheksmodul:

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken[34] auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).



2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

3. Navigieren Sie zur .xq-oder .xquery-Bibliotheksdatei und klicken Sie auf **Öffnen**.

Die importierten Bibliotheksmoduls werden im Fenster "Bibliotheken" angezeigt. Anschließend können Sie einzelne Funktionen in den Mapping-Bereich ziehen und diese wie jede andere MapForce-Funktionskomponente verwenden, siehe auch Hinzufügen einer Funktion zum Mapping[943].

Wenn Sie das importierte XQuery-Bibliotheksmodul nicht sehen, stellen Sie sicher, dass Sie XQuery als Transformationssprache[28] ausgewählt haben.

© 2016-2022 Altova GmbH

Siehe auch:

Implementierung des XQuery-Prozessors 1990

6.5.2.1 Beispiel: Import einer benutzerdefinierten XQuery-Funktion

In diesem Beispiel wird gezeigt, wie Sie eine XQuery-Demobibliothek in MapForce importieren und deren Funktionen über ein Mapping aufrufen. Das Demomodul in diesem Beispiel besteht aus nur einer Funktion, die anhand von Dezimalbeträgen eine 20%ige Steuer berechnet. In einer Produktionsumgebung kann ein XQuery-Modul mehrere Funktionen enthalten.

> Alle im XQuery-Modul deklarierten Funktionen müssen atomare Typen zurückgeben und auch ihre Parameter müssen atomare Datentypen sein, da das Modul sonst nicht in MapForce importiert werden kann.

Das XQuery-Demomodul befindet sich auf dem Rechner, auf dem MapForce installiert ist, unter dem folgenden Pfad relativ zu Ihrem persönlichen Ordner "Dokumente" :
<Dokumente>\Altova\MapForce2023\MapForceExamples\module.xq.

```
xquery version "1.0";

module namespace demo="http://www.altova.com/mapforce/demo";

declare function demo:calculatetax($val as xs:decimal) as xs:decimal {
   $val*0.2
};
```

module.xq

Nachdem Sie die XQuery-Moduldatei in MapForce importiert haben, können Sie die Funktion **demo:calculatetax** von einem Mapping aus aufrufen. Beachten Sie, dass die Berechnung der Steuer mit Hilfe einer XQuery-Funktion nur zu Demozwecken erfolgt. Dasselbe Ergebnis können Sie auch mit Hilfe von vordefinierten MapForce-Funktionen erzielen.

Unter dem folgenden Pfad finden Sie ein Demo-Mapping, das die Funktion **demo:calculatetax** aufruft: **<Dokumente>\Altova\MapForce2023\MapForceExamples\CalculateTax_XQuery.mfd**. Wenn Sie dieses Mapping anfangs öffnen, zeigt MapForce eine Warnmeldung an, dass eine oder mehrere Komponenten darin enthalten sind, die in XQuery nicht zur Verfügung stehen. Diese Warnung ist normal, da sie auftritt, wenn im Mapping eine Funktion von einer benutzerdefinierten XQuery-Bibliothek aus referenziert wird, die noch nicht importiert wurde. Damit die Warnung nicht mehr angezeigt wird, werden wir das fehlende XQuery-Modul in MapForce, wie unten gezeigt, importieren.

So importieren Sie das XQuery-Modul in MapForce:

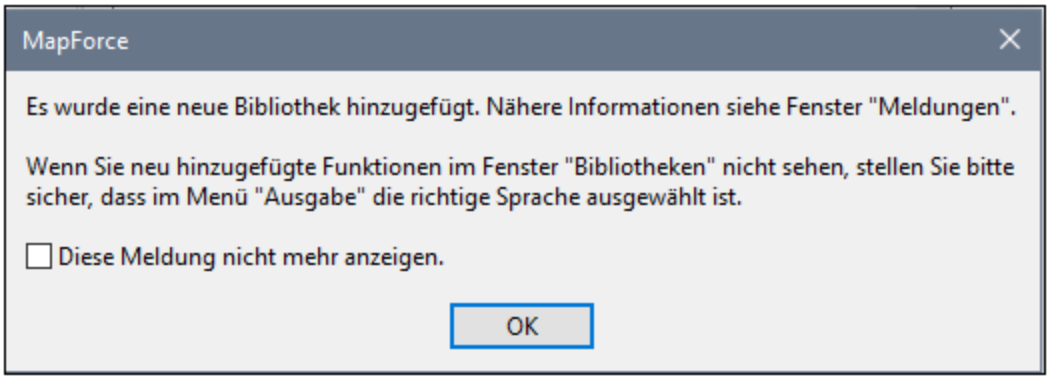
1.

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken[34] auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).



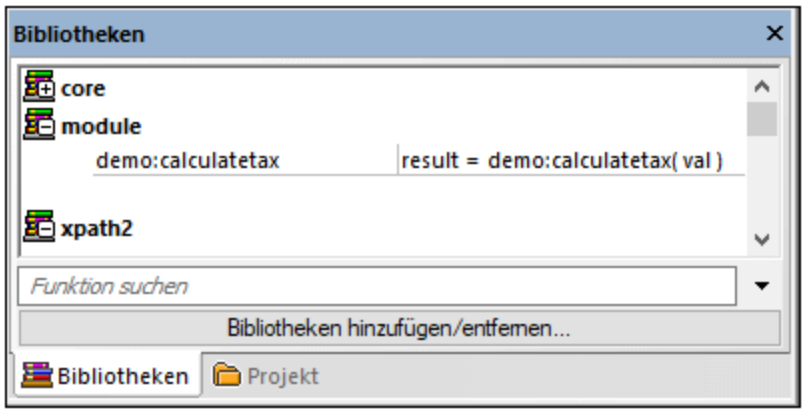
2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

3. Navigieren Sie zu **<Dokumente>\Altova\MapForce2023\MapForceExamples\module.xq** und klicken Sie auf **Öffnen**. Daraufhin erscheint ein Meldungsfeld, in dem Sie darüber informiert werden, dass eine neue Bibliothek hinzugefügt wurde.

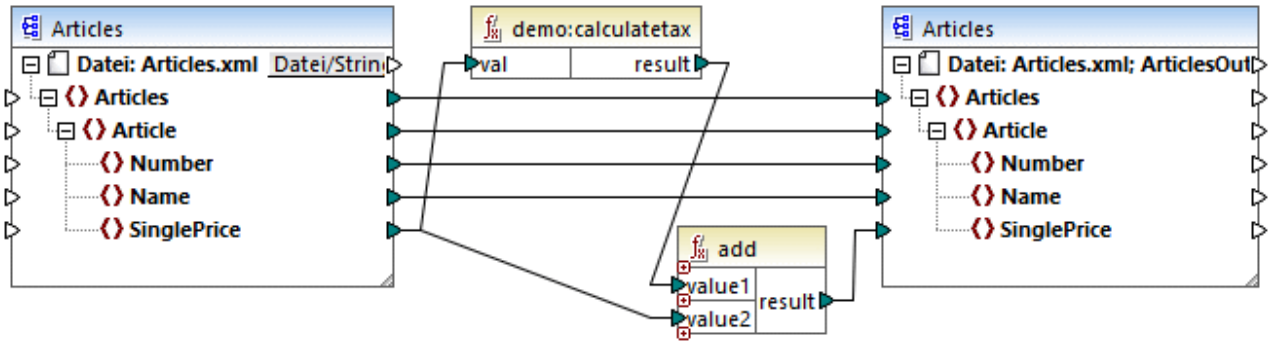


Die importierte Bibliothek und die Funktion `demo:calculatetax` werden nun im Fenster "Bibliotheken" angezeigt.

© 2016-2022 Altova GmbH



Das Mapping kann nun ohne Warnungen validiert und ausgeführt werden. Die Funktion **demo:calculatetax** in der Abbildung unten stammt aus dem importierten XQuery-Modul und kann wie jede andere vordefinierte Funktion zum Mapping hinzugefügt werden, siehe Hinzufügen einer Funktion zum Mapping 943.



CalculateTax_XQuery.mfd

Erläuterung des Mappings

Das oben gezeigte Mapping **CalculateTax_XQuery.mfd** erhält als Input eine XML-Datei, in der Artikel gespeichert sind. Jeder Artikel verfügt über einen einzigen Preis, der in Form eines Dezimalwerts angegeben ist, z.B.:

```
<Articles>
   <Article>
      <Number>1</Number>
      <Name>T-Shirt</Name>
      <SinglePrice>25</SinglePrice>
   </Article>
   <Article>
      <Number>2</Number>
      <Name>Socks</Name>
      <SinglePrice>2.30</SinglePrice>
   </Article>
```

```
   <Article>
      <Number>3</Number>
      <Name>Pants</Name>
      <SinglePrice>34</SinglePrice>
   </Article>
   <Article>
      <Number>4</Number>
      <Name>Jacket</Name>
      <SinglePrice>57.50</SinglePrice>
   </Article>
</Articles>
```

Articles.xml

Die mit dem Mapping erzeugte XML-Datei, die demselben Schema wie die XML-Quelldatei entspricht. Daher haben Quell- und Zielkomponente im Mapping dieselbe Struktur. Wie Sie an den Mappingverbindungen sehen, wurden fast alle Elemente direkt von der Quellkomponente auf die Zielkomponente gemappt - so gibt es z.B. für jedes Element **Article** in der Quellkomponente ein **Article**-Element in der Zielkomponente. Die Werte aller Datenelemente werden unverändert von der XML-Quelldatei kopiert. Die einzige Ausnahme ist **SinglePrice**. Der Wert von **SinglePrice** wird mit Hilfe zweier Funktionen berechnet:

- Die XQuery-Funktion **demo:calculatetax** berechnet den Steuerbetrag anhand des ursprünglichen **SinglePrice**-Elements als Input.
- Die vordefinierte MapForce Funktion **add** fügt den Steuerbetrag zum ursprünglichen **SinglePrice**-Betrag hinzu und gibt den Endbetrag zurück.

Beachten Sie, dass der Datentyp des Datenelements **SinglePrice** `xs:decimal` ist. Dies entspricht dem Input-Parametertyp und dem Rückgabetyp der XQuery-Funktion.

Unten sehen Sie das Ergebnis, das bei Klick auf das Register **Ausgabe** erzeugt wird. Beachten Sie, dass jeder Preis im Vergleich zur XML-Quelldatei um 20% erhöht wurde.

© 2016-2022 Altova GmbH

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Articles xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    <Article>
        <Number>1</Number>
        <Name>T-Shirt</Name>
        <SinglePrice>30</SinglePrice>
    </Article>
    <Article>
        <Number>2</Number>
        <Name>Socks</Name>
        <SinglePrice>2.76</SinglePrice>
    </Article>
    <Article>
        <Number>3</Number>
        <Name>Pants</Name>
        <SinglePrice>40.8</SinglePrice>
    </Article>
    <Article>
        <Number>4</Number>
        <Name>Jacket</Name>
        <SinglePrice>69</SinglePrice>
    </Article>
</Articles>
```

6.5.3 Importieren benutzerdefinierter Java- und .NET-Bibliotheken

In diesem Abschnitt wird erläutert, wie Sie kompilierte Java-Klassendateien sowie .NET DLL-Assembly-Dateien (einschließlich .NET 4.0 Assemblys) in MapForce importieren. Wenn die importierten Bibliotheken Funktionen enthalten, die Basisdatentypen als Parameter verwenden und simpleTypes zurückgeben, so werden solche Funktionen im Fenster **Bibliotheken** angezeigt und können wie jede andere Funktion in MapForce Mappings verwendet werden. Eine Vorschau der Mapping-Ausgabe der importierten Java- und .NET-Funktionen kann im **Ausgabefenster** angezeigt werden. Außerdem stehen die Funktionen im generierten Code zur Verfügung. Nähere Informationen zum Import benutzerdefinierter Bibliotheken finden Sie in den Beispielen in den Kapiteln Import einer benutzerdefinierten Java-Klasse[1001] und Import einer benutzerdefinierten .NET DLL-Assembly[1003].

Achtung:

- Um benutzerdefinierte Java- oder .NET-Funktionen zu importieren, benötigen Sie kompilierte Java-Klassen (`.class`) oder .NET.dll-Assembly-Dateien. Der Import von Java `.jar`-Dateien oder `.dll`-Dateien, bei denen es sich nicht um eine .NET-Assembly handelt, wird nicht unterstützt.

- .NET Assembly-Dateien werden unterstützt, wenn als Mapping-Sprache C# eingestellt ist. Die .NET Assemblies können in .NET-Sprachen mit Ausnahme von C# geschrieben sein (z.B. C++.NET oder VB.NET), wenn sie als Parameter und Rückgabetypen nur die Basisdatentypen aus der System Assembly verwenden. Nähere Informationen dazu finden Sie unter Unterstützung von .NET-Funktionen[999].

- Wenn Sie benutzerdefinierte NET-Funktionen in der integrierten Ausgabevorschau (im Fenster **Ausgabe**) verwenden möchten, müssen diese Funktionen für das .NET Framework 4.x oder :NET Standard 2.0 kompiliert werden.

- Kompilierte Java-Klassendateien (`.class`-Dateien) werden unterstützt, wenn als Mapping-Sprache Java eingestellt ist. Auf Ihrem Rechner muss Java Runtime Environment 7 oder höher installiert sein. Es werden nur bestimmte Typen und Member unterstützt (siehe Unterstützung von Java-Funktionen 998).

- Sie können die Mapping-Sprache nicht auf C++ setzen, wenn im Mapping importierte Java`.class`- oder .NET DLL-Assemblys verwendet werden.

- Sie können die Mapping-Sprache nicht auf XSLT setzen, wenn im Mapping importierte Java`.class`- oder .NET DLL-Assemblys verwendet werden (es müsste eine benutzerdefinierte XSLT-Funktion geschrieben werden, die als Adapter fungiert).

- Der Import von Funktionen aus nativen C++ DLLs ist eingeschränkt. Es ist dafür eine eigene Methode erforderlich. Nähere Informationen finden Sie unter Manuelles Referenzieren von Java, C# und C++-Bibliotheken 1005.

- Alle von einem MapForce Mapping aus aufgerufenen Funktionen sollten jedes Mal, wenn die Funktion mit denselben Input-Parametern aufgerufen wird, denselben Wert zurückgeben. Die genaue Reihenfolge und wie oft eine Funktion von MapForce aufgerufen wird, ist nicht definiert.

- Im Fall von Java müssen die importierten Klassendateien nicht zur CLASSPATH-Variablen hinzugefügt werden, da der Built-In-Ausführungsprozessor und der generierte Java-Code importierte Pakete automatisch zum Classpath bzw. zum ANT des Java-Prozessors hinzufügen. Abhängigkeiten der importierten Klassendateien und Pakete werden jedoch nicht automatisch behandelt. Wenn importierte Java-Klassendateien daher von anderen Klassendateien abhängig sind, stellen Sie sicher, dass auch die übergeordneten Verzeichnisse aller abhängigen Pakete zur CLASSPATH-Umgebungsvariable hinzugefügt werden.

Unterstützung von Java-Funktionen

Klassen der obersten Ebene, statische Member-Klassen und nicht statische Member-Klassen werden unterstützt:

- `new <classname>(<arg1>, <arg2>, ...)`
- `<object>.new <member-class>(<arg1>, <arg2>, ...)`

Member-Funktionen und statische Funktionen werden unterstützt:

- `<function>(<arg1>, <arg2>, ...)`
- `<object>.<method>(<arg1>, ...)`

Unterstützte Verbindungen zwischen XML-Schema- und Java-Typen:

Schema-Typ	Java-Typ
`xs:string`	`String`
`xs:byte`	`byte`
`xs:short`	`short`

© 2016-2022 Altova GmbH

Schema-Typ	Java-Typ
xs:int	int
xs:long	long
xs:boolean	boolean
xs:float	float
xs:double	double
xs:decimal	java.math.BigDecimal
xs:integer	java.math.BigInteger

Verbindungen können in beide Richtungen hergestellt werden. Andere Java-Typen (einschließlich Array-Typen) werden nicht unterstützt. Methoden, die solche Parameter oder Rückgabewerte verwenden, werden ignoriert. Object-Typen werden durch Aufruf ihres Konstruktors oder als Rückgabewert einer Methode unterstützt. Sie können auf andere Java-Methoden gemappt werden. Eine Bearbeitung des Objekts mit Hilfe von MapForce ist nicht möglich.

Unterstützung von .NET-Funktionen

Klassen der obersten Ebene werden unterstützt:

- `new <classname>(<arg1>, <arg2>, ...)`

Member-Funktionen und statische Funktionen werden unterstützt:

- `<function>(<arg1>, <arg2>, ...)`
- `<object>.<method>(<arg1>, ...)`

Unterstützte Verbindungen zwischen XML-Schema- und .NET/C#-Typen:

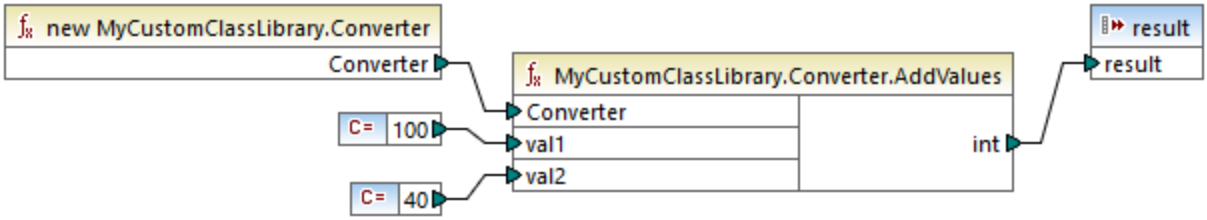
Schema-Typ	.NET-Typ	C#-Typ
xs:string	System.String	string
xs:byte	System.SByte	sbyte
xs:short	System.Int16	short
xs:int	System.Int32	int
xs:long	System.Int64	long
xs:unsignedByte	System.Byte	byte
xs:unsignedShort	System.UInt16	ushort
xs:unsignedInt	System.UInt32	uint
xs:unsignedLong	System.UInt64	ulong

Schema-Typ	.NET-Typ	C#-Typ
`xs:boolean`	`System.Boolean`	`bool`
`xs:float`	`System.Single`	`float`
`xs:double`	`System.Double`	`double`
`xs:decimal`	`System.Decimal`	`decimal`

Verbindungen können in beide Richtungen hergestellt werden. Andere .NET/C#-Typen (einschließlich Array-Typen) werden nicht unterstützt. Methoden, die solche Parameter oder Rückgabewerte verwenden, werden ignoriert. Object-Typen werden durch Aufruf ihres Konstruktors oder als Rückgabewert einer Methode unterstützt. Sie können auf andere .NET-Methoden gemappt werden. Eine Bearbeitung des Objekts mit Hilfe von MapForce ist nicht möglich.

Probleme und Umgehungslösungen im Zusammenhang mit Datentypen

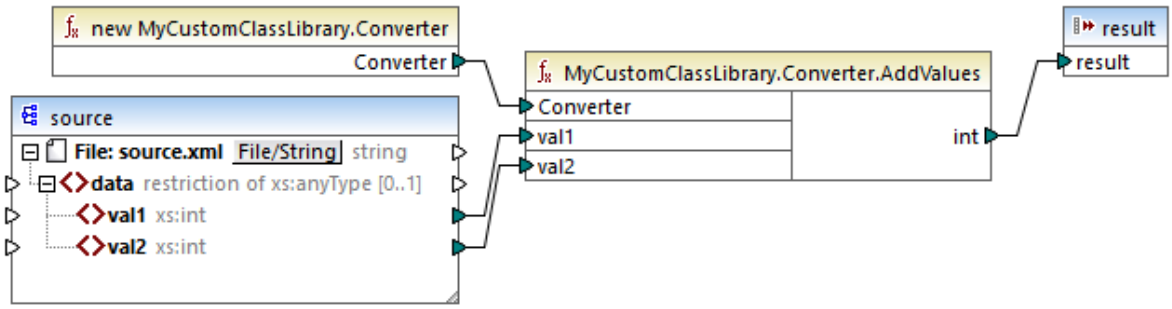
Wenn eine Funktion in Ihrer benutzerdefinierten Bibliothek Ganzzahltypen erwartet, kann es bei Verbindung von Konstanten vom Typ `Zahl` mit den Argumenten der Funktion zu einem Fehler wegen einem nicht übereinstimmendem Typ kommen ähnlich dem folgenden: `No match for MyCustomClassLibrary, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null.MyCustomClassLibrary.Converter.AddValues(MyCustomClassLibrary.Converter, xs:decimal, xs:decimal). Check argument types.` Dieses Problem tritt spezifisch bei Konstanten vom Typ `Zahl` auf. Unten sehen Sie ein Beispielmapping, bei dem dieser Fehler generiert werden könnte. In diesem Mapping sind zwei Konstanten vom Typ `Zahl` mit den Argumenten der Funktion vom Typ `Integer` verbunden.



Unten sind mögliche Umgehungslösungen beschrieben:

1. Ändern Sie den Konstantentyp von **Zahl** in **Alle anderen**. Doppelklicken Sie dazu auf die Titelleiste der Konstantenkomponente, um dies zu ändern.
2. Verwenden Sie anstatt einer Konstante eine Quellkomponente (z.B. eine XML-Datei), die Werte des von der Funktion erwarteten Datentyps bereitstellt.

© 2016-2022 Altova GmbH



3. Erstellen Sie in Ihrem externen Code eine Wrapper-Funktion, die einen Dezimalwert akzeptiert und einen Ganzzahlwert zurückgibt. Die Wrapper-Lösung kann als separate Bibliothek importiert werden. Daher müssen sie den Originalquellcode der Zielfunktion nicht ändern, um diese Methode verwenden zu können.

6.5.3.1 Beispiel: Import einer benutzerdefinierten Java-Klasse

In diesem Beispiel sehen Sie, wie Sie eine benutzerdefinierte Java **`.class`**-Datei in MapForce importieren.

Anmerkung: Für dieses Beispiel wird Java SE 8 Runtime Environment oder höher benötigt.

Java .class-Import
Um eine Java `.class`-Datei als MapForce-Bibliothek zu importieren, gehen Sie folgendermaßen vor:

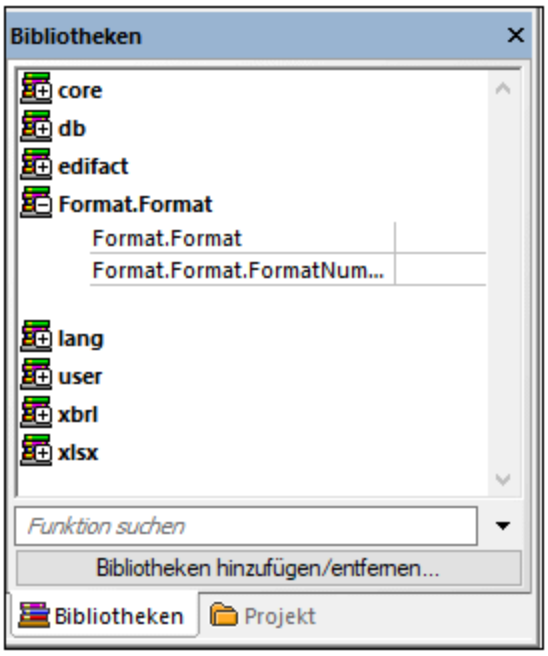
1. Klicken Sie im unteren Bereich des Fensters Bibliotheken[34] auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).



2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

3. Suchen Sie nach der folgenden Datei:
`<Dokumente>\Altova\MapForce2023\MapForceExamples\Java\Format\Format.class`. Daraufhin wird eine Meldung, in der Sie informiert werden, dass eine neue Bibliothek hinzugefügt wurde,

angezeigt. Die importierte Bibliothek wird nun im Fenster **Bibliotheken** angezeigt (*siehe Abbildung unten*).

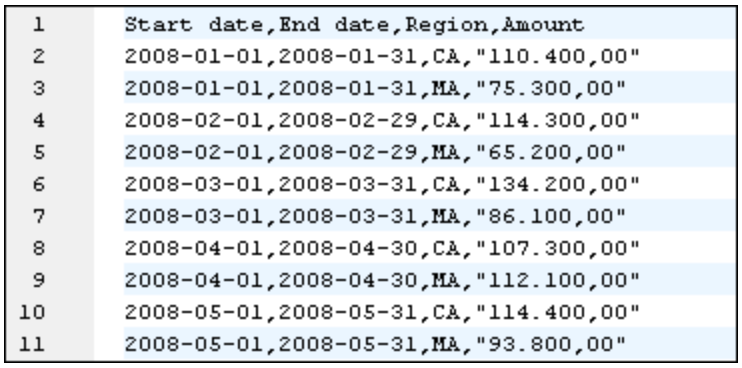


Wenn die neu importierte Bibliothek im Fenster **Bibliotheken** nicht angezeigt wird, vergewissern Sie sich, dass als Transformationssprache[28] Java eingestellt ist. Um die Funktionen zum Mappings hinzuzufügen, ziehen Sie diese aus dem Fenster **Bibliotheken** in den Mapping-Bereich. Nähere Informationen dazu finden Sie unter Hinzufügen einer Funktion zum Mapping[943].

Mapping-Ausgabe

Um eine Vorschau auf die Mapping-Ausgabe in MapForce anzuzeigen, gehen Sie folgendermaßen vor:

1. Öffnen Sie das folgende Mapping: **<Dokumente>\Altova\MapForce2023\MapForceExamples\Java\FormatNumber.mfd**. Es handelt sich hierbei um komplettes Mapping, in das die oben erwähnte Java `.class`-Bibliothek bereits importiert wurde.
2. Klicken Sie auf die Schaltfläche **Ausgabe**, um das Ergebnis des Mappings zu sehen (*siehe Abbildung unten*).



```
 1  Start date,End date,Region,Amount
 2  2008-01-01,2008-01-31,CA,"110.400,00"
 3  2008-01-01,2008-01-31,MA,"75.300,00"
 4  2008-02-01,2008-02-29,CA,"114.300,00"
 5  2008-02-01,2008-02-29,MA,"65.200,00"
 6  2008-03-01,2008-03-31,CA,"134.200,00"
 7  2008-03-01,2008-03-31,MA,"86.100,00"
 8  2008-04-01,2008-04-30,CA,"107.300,00"
 9  2008-04-01,2008-04-30,MA,"112.100,00"
10  2008-05-01,2008-05-31,CA,"114.400,00"
11  2008-05-01,2008-05-31,MA,"93.800,00"
```


© 2016-2022 Altova GmbH

Mapping in Java
Um das Mapping in Java auszuführen, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Datei** auf den Befehl **Code generieren in | Java**.
2. Wählen Sie ein Zielverzeichnis für den generierten Code aus und klicken Sie auf **OK**.
3. Importieren Sie die generierten Bibliotheken in Ihr Java-Projekt und erstellen Sie die Java-Applikation. Nähere Informationen dazu finden Sie unter Beispiel: Generieren und Ausführen von Java-Code. 1586)

6.5.3.2 Beispiel: Import einer benutzerdefinierten .NET DLL-Assembly

In diesem Beispiel sehen Sie, wie Sie eine benutzerdefinierte in C# erstellte .NET DLL-Assembly in MapForce importieren. Der Quellcode zu diesem Beispiel steht unter dem folgenden Pfad zur Verfügung: `<Dokumente>\Altova\MapForce2023\MapForceExamples\C#\Format`. Die `.dll` Assembly-Datei, die in MapForce importiert werden soll, befindet sich im Verzeichnis `..\bin\Debug`. Sie können auch die `.sln`-Projektmappendatei in Visual Studio öffnen und eine neue `.dll`-Datei kompilieren.

Anmerkung: Wenn Sie benutzerdefinierte NET-Funktionen in der integrierten Ausgabevorschau (im Fenster **Ausgabe**) verwenden möchten, müssen diese Funktionen für das .NET Framework 4.x oder :NET Standard 2.0 kompiliert werden.

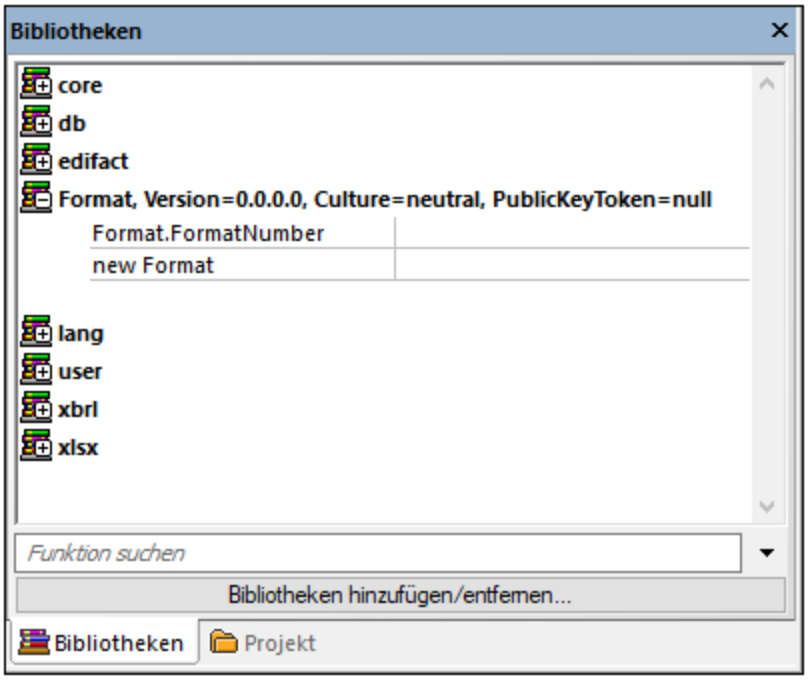
Import der .NET Assembly
Um eine NET-Assembly-Datei zu importieren, gehen Sie folgendermaßen vor:

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken 34 auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).



2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

3. Suchen Sie nach `Format.dll` im folgenden Verzeichnis: `...\MapForceExamples\C#\Format\bin\Debug\`. Daraufhin wird eine Meldung, in der Sie informiert werden, dass eine neue Bibliothek hinzugefügt wurde, angezeigt. Die importierte Bibliothek wird nun im Fenster **Bibliotheken** angezeigt.



Wenn die neu importierte Bibliothek im Fenster **Bibliotheken** nicht angezeigt wird, vergewissern Sie sich, dass als Transformationssprache[28] C# eingestellt ist. Um die Funktionen zum Mapping hinzuzufügen, ziehen Sie diese aus dem Fenster **Bibliotheken** in den Mapping-Bereich. Nähere Informationen dazu finden Sie unter Hinzufügen einer Funktion zum Mapping[943].

Mapping-Ausgabe
Um eine Vorschau auf die Mapping-Ausgabe anzuzeigen, gehen Sie folgendermaßen vor:

1. Öffnen Sie die Datei **FormatNumber.mfd** aus dem Ordner **...\MapForceExamples\C#**. Es handelt sich hierbei um ein Beispiel-Mapping, in das die oben erwähnte .dll-Bibliothek bereits importiert wurde.
2. Klicken Sie auf die Schaltfläche **Ausgabe**, um das Ergebnis des Mappings zu sehen (*siehe Abbildung unten*).

```
 1   Start date,End date,Region,Amount
 2   2008-01-01,2008-01-31,CA,"110.400,00"
 3   2008-01-01,2008-01-31,MA,"75.300,00"
 4   2008-02-01,2008-02-29,CA,"114.300,00"
 5   2008-02-01,2008-02-29,MA,"65.200,00"
 6   2008-03-01,2008-03-31,CA,"134.200,00"
 7   2008-03-01,2008-03-31,MA,"86.100,00"
 8   2008-04-01,2008-04-30,CA,"107.300,00"
 9   2008-04-01,2008-04-30,MA,"112.100,00"
10   2008-05-01,2008-05-31,CA,"114.400,00"
11   2008-05-01,2008-05-31,MA,"93.800,00"
```

Mapping in C#
Um das Mapping von einer benutzerdefinierten C#-Applikation aus auszuführen, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Datei** auf den Befehl **Code generieren in | C#**.

© 2016-2022 Altova GmbH

2. Wählen Sie ein Zielverzeichnis für den generierten Code aus und klicken Sie auf **OK**.
3. Erstellen Sie die Applikation mit Visual Studio und führen Sie die generierte Konsolenapplikation aus. Nähere Informationen dazu finden Sie unter Generieren von C#-Code[1572].

6.5.4 Manuelles Referenzieren von Java, C# und C++-Bibliotheken

In diesem Abschnitt wird erläutert, wie Sie benutzerdefinierte Bibliotheken in einer `.mff`-Datei (MapForce Function File) referenzieren. Die `.mff`-Datei, die die Referenz enthält, kann anschließend als MapForce-Bibliothek importiert werden. Eine `.mff`-Datei ist eine XML-Datei, in der Sie die Verknüpfungen zwischen Klassendefinitionen in Ihrem benutzerdefinierten Code und MapForce manuell definieren. Nach Erstellung einer benutzerdefinierten `.mff`-Datei können Sie diese ähnlich wie eine .NET DLL- oder Java class-Datei in MapForce importieren.

Achtung:

- Wenn Sie benutzerdefinierte NET-Funktionen in der integrierten Ausgabevorschau (im Fenster **Ausgabe**) verwenden möchten, müssen diese Funktionen für das .NET Framework 4.x oder :NET Standard 2.0 kompiliert werden.

- Sie können eine Funktion nur dann in MapForce importieren, wenn ihr Rückgabetyp und ihre Parameter den Typ simpleType haben. Eine Liste der verfügbaren Datentypen für die einzelnen Sprachen finden Sie unter Datentyp-Zuordnung[1011].

- Wenn Sie Funktionsbibliotheken aus benutzerdefinierten `.mff`-Dateien importieren, Vorschau auf das Mapping direkt in MapForce (im Fenster **Ausgabe**) eingeschränkt. Bei in C++ geschriebenen Bibliotheken wird die Vorschau auf das Mapping in MapForce nicht unterstützt. Bei Java und C# steht die Vorschau zur Verfügung, wenn in Ihrer Bibliothek native Sprachtypen verwendet werden, nicht aber, wenn darin die von Altova generierte Klassen importiert wurden. Sie können jedoch Code in der Zielsprache der jeweiligen Bibliothek generieren. Die benutzerdefinierten Funktionen stehen im generierten Code zur Verfügung, sodass Sie das Mapping über den generierten Code ausführen können.

- Die genaue Reihenfolge, in der Funktionen vom generierten Mapping-Code aufgerufen werden, ist nicht definiert. MapForce kann die berechneten Ergebnisse zur Wiederverwendung im Cache aufbewahren oder Ausdrücke in jeder beliebigen Reihenfolge auswerten. Es wird daher empfohlen, nur benutzerdefinierte Funktionen zu verwenden, die keine unerwünschten anderen Auswirkungen haben.

- Es muss zwischen benutzerdefinierten Funktionen und benutzerdefinierten Funktionsbibliotheken unterschieden werden. Benutzerdefinierte Funktionen werden grafisch in einem Mapping erstellt und können und müssen nicht in einer `.mff`-Datei gespeichert werden, da sie zusammen mit der `.mfd`-Datei, in der sie erstellt wurden, gespeichert werden. Nähere Informationen dazu finden Sie unter Aufrufen und Importieren von benutzerdefinierten Funktionen[971].

- Wenn Sie ein Upgrade von einer MapForce-Version vor 2010 installieren, müssen Sie die in Ihren benutzerdefinierten Funktionen verwendeten Datentypen eventuell aktualisieren (siehe Datentyp-Zuordnung[1011].)

Eine Anleitung zum Erstellen und Konfigurieren einer benutzerdefinierten `.mff`-Datei finden Sie unter Konfigurieren der MFF-Datei[1006]. Die Beispiele finden Sie in den folgenden Kapiteln:

- Referenzieren von C# in .mff 1014
- Referenzieren von C++ in .mff 1015
- Referenzieren von Java in .mff 1018

6.5.4.1 Konfigurieren der MFF-Datei

In diesem Kapitel wird erläutert, wie Sie eine MapForce-Funktionsdatei (`.mff`) konfigurieren. Eine `.mff`-Datei ist eine konfigurierbare Datei im XML-Format, mit deren Hilfe Sie Funktionen aus benutzerdefinierten Java-, C#- oder C++-Bibliotheken in MapForce importieren können, so dass sie im Fenster **Bibliotheken** angezeigt werden. Eine `.mff`-Datei dient als Verbindungsglied zwischen Ihren benutzerdefinierten Bibliotheken und MapForce. Die `.mff`-Datei muss so konfiguriert sein, dass sie i) die Schnittstellen zu den benutzerdefinierten Funktionen definiert und ii) angibt, wo die Implementierung im generierten Code zu finden ist.

Achtung:

- Die `*.mff`-Bibliotheksdateien müssen dem folgenden Schema gemäß gültig sein: **`C:\Programme\MapForceLibraries\mff.xsd`**. Das **`mff.xsd`**-Schema definiert die Konfiguration der benutzerdefinierten Bibliothek und dient nur zur internen Verwendung. Die Altova GmbH behält sich das Recht vor, dieses Dateiformat in neuen Releases zu ändern.
- Es kann nur eine C#-, C++- oder Java-Klasse pro `.mff`-Datei definiert werden.

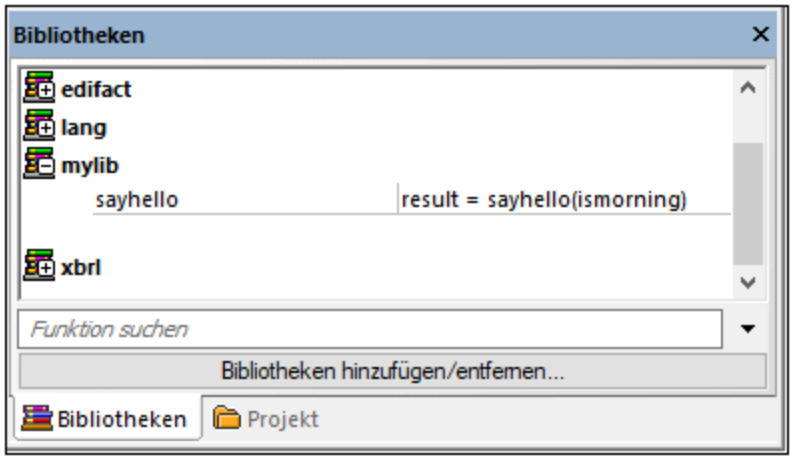
.mff-Beispiel für C#
Im folgenden Codefragment sehen Sie eine `.mff`-Datei für C++:

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping version="9" library="mylib" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
   <implementations>
      <implementation language="cpp">
         <setting name="namespace" value="mylib"/>
         <setting name="class" value="Greetings"/>
         <setting name="path" value="C:\Libraries\cpp"/>
         <setting name="include" value="Greetings.h"/>
         <setting name="source" value="Greetings.cpp"/>
      </implementation>
   </implementations>
   <group name="greetings">
      <component name="sayhello">
         <sources>
            <datapoint name="ismorning" type="xs:boolean"/>
         </sources>
         <targets>
            <datapoint name="result" type="xs:string"/>
         </targets>
         <implementations>
            <implementation language="cpp">
               <function name="SayHello"/>
            </implementation>
         </implementations>
         <description>
```

 © 2016-2022 Altova GmbH

```
            <short>result = sayhello(ismorning)</short>
            <long>Returns "Good morning" or "Good day", depending on the input
parameter.</long>
         </description>
      </component>
   </group>
</mapping>
```

Importierte benutzerdefinierte Bibliothek
In der Abbildung unten sehen Sie, wie eine benutzerdefinierte `.mff`-Datei nach dem Import in MapForce aussehen könnte. Die benutzerdefinierte Bibliothek **`mylib`** erscheint als Eintrag in der Bibliotheksliste (in alphabetischer Reihenfolge sortiert) und enthält die String-Funktion `sayhello`.



Konfigurationsschritte

Um die `.mff`-Datei zu konfigurieren, gehen Sie vor, wie unten beschrieben.

Schritt 1. Konfigurieren des Bibliotheksnamens
Der Bibliotheksname befindet sich in der `.mff`-Datei (*siehe unten*). Konventionsgemäß sind Bibliotheksnamen in MapForce in Kleinbuchstaben geschrieben, es können aber auch Großbuchstaben verwendet werden.

```
<mapping version="9" library="mylib" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
```

Im obigen Beispiel hat der im Fenster **Bibliotheken** angezeigte Eintrag den Namen **`mylib`**.

Schritt 2. Konfigurieren der Sprachimplementierungen
Das Element `<implementations>` ist ein obligatorisches Element. Es gibt an, welche Sprachen Ihre Bibliothek unterstützen soll und muss als Child-Element von `<mapping>` hinzugefügt werden (*siehe Beispiel unten*).

```
<!-- ... -->
<mapping version="9" library="mylib" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
   <implementations>
```

```
      <implementation language="cpp">
         <setting name="namespace" value="mylib"/>
         <setting name="class" value="Greetings"/>
         <setting name="path" value="C:\Libraries\cpp"/>
         <setting name="include" value="Greetings.h"/>
         <setting name="source" value="Greetings.cpp"/>
      </implementation>
   </implementations>
<!-- ... -->
```

Über die Einstellungen in den einzelnen `<implementation>`-Elementen kann der generierte Code die jeweiligen in Java, C++ oder C# definierten Funktionen aufrufen. Eine `.mff`-Datei kann so geschrieben werden, dass sie für mehrere Programmiersprachen verwendet werden kann. In diesem Fall muss jede Sprache ein weiteres `<implementation>`-Element enthalten. Im Folgenden werden die Einstellungen für die einzelnen Programmiersprachen erläutert.

Java-Bibliotheksreferenz

```
<!-- ... -->
<implementation language="java">
   <setting name="package" value="com.hello.functions"/>
   <setting name="class" value="Greetings"/>
</implementation>
<!-- ... -->
```

Damit der generierte Code Ihre **`Greetings.class`**-Datei findet, stellen Sie sicher, dass zum Java Classpath eine Referenz zu Ihrer Klasse hinzugefügt wurde.

C#-Bibliotheksreferenz

```
<!-- ... -->
      <implementation language="cs">
         <setting name="namespace" value="MyLibrary" />
         <setting name="class" value="Greetings" />
         <setting name="reference" value="C:
\Libraries\cs\MyLibrary\bin\debug\MyLibrary.dll" />
      </implementation>
<!-- ... -->
```

Es ist in C# wichtig, dass der Namespace im Code dem in der `.mff`-Datei definierten Namespace entspricht (im Codefragment oben ist der Namespace **`MyLibrary`**). Dasselbe gilt für den Klassennamen (im Codefragment oben ist der Klassenname **`Greetings`**). Die dritte Einstellung, **`reference`**, gibt den Pfad der mit dem generierten Code zu verknüpfenden `dll`-Datei an.

C++-Bibliotheksreferenz

```
<!-- ... -->
   <implementation language="cpp">
         <setting name="namespace" value="MyLibrary"/>
         <setting name="class" value="Greetings"/>
         <setting name="path" value="C:\Libraries\cpp"/>
         <setting name="include" value="Greetings.h"/>
         <setting name="source" value="Greetings.cpp"/>
   </implementation>
<!-- ... -->
```

Beachten Sie folgende Anmerkungen zu C++:

© 2016-2022 Altova GmbH

- **namespace** ist der Namespace, in dem Ihre `Greetings`-Klasse definiert wird. Er muss mit dem Attribut **library** im Element **mapping** übereinstimmen.
- **path** ist der Pfad, unter dem die include- und die Quelldatei zu finden sind.
- Wenn Code für ein Mapping generiert wird, werden die include- und die Quelldatei in das Verzeichnis **targetdir/libraryname** kopiert, das durch Auswahl der Menüoption **Datei | Code generieren in | C++** definiert wird, und in die Projektdatei inkludiert.

Alle bereitgestellten include-Dateien werden in den generierten Algorithmus inkludiert.

Schritt 3. Hinzufügen einer Komponente
Im **Bibliotheksfenster** wird jede Funktion unter einer Funktionsgruppe angezeigt, z.B. "string functions". In der `.mff`-Datei entspricht eine Funktion einem `<component>`-Element. Umgekehrt muss sich jedes `<component>`-Element unter einem `<group>`-Element befinden, z.B.:

```
<!-- ... -->
<group name="string functions">
   <component name="sayhello">
   <!-- ... -->
   </component>
</group>
<!-- ... -->
```

Im unten stehenden Code ist eine Beispielfunktion (component) namens **sayhello**definiert.

```
<!-- ... -->
<component name="sayhello">
   <sources>
      <datapoint name="ismorning" type="xs:boolean"/>
   </sources>
   <targets>
      <datapoint name="result" type="xs:string"/>
   </targets>
   <implementations>
   <!-- ... -->
   </implementations>
   <description>
     <short>result = sayhello(ismorning)</short>
     <long>Returns "Good morning" or "Good day", depending on the input
parameter.</long>
   </description>
</component>
<!-- ... -->
```

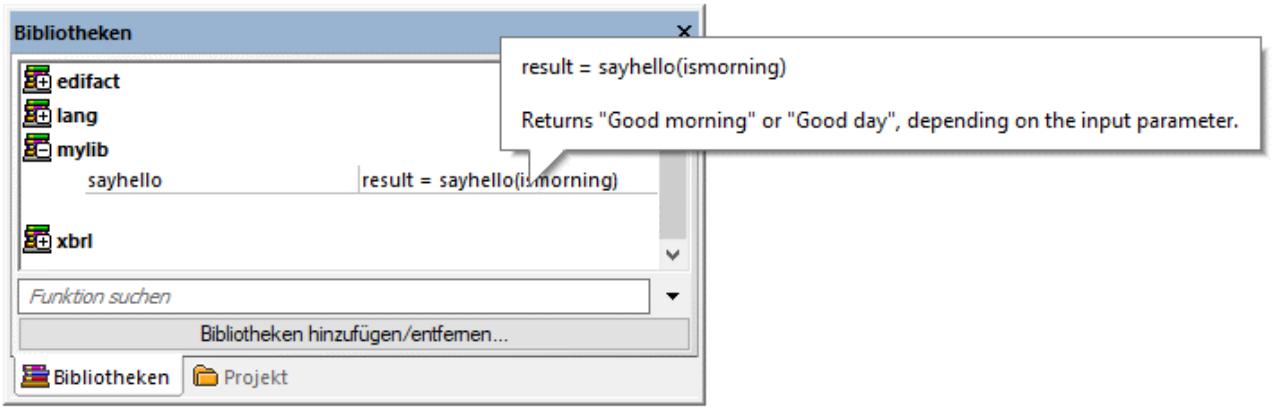
So würde die Komponente in MapForce angezeigt werden:



Ein `<datapoint>`-Element im obigen Codefragment ist sozusagen der Eingabe- oder Ausgabeparameter einer Funktion (auch als Input- oder Output-Konnektor bezeichnet). Das Argument `type` des `<datapoint>`-Elements definiert den Datentyp des Parameters oder den Datentyp des Rückgabewerts. Für jede Funktion kann nur ein Ziel-Datenpunkt definiert werden. Sie können jedoch beliebig viele Quell-Datenpunkte definieren.

Beim Datentyp für die einzelnen Datenpunkte muss es sich um einen der XML-Schematypen handeln (z.B. `xs:string`, `xs:integer`, usw.). Diese Datentypen müssen den Datentypen der Funktionsparameter, die Sie in Ihrer Java-, C++- oder C#-Bibliothek definiert haben, entsprechen. Informationen zum Mapping von XML-Schema-Datentypen auf Sprachtypen finden Sie unter Datentyp-Zuordnung 1011.

Funktionen sind im **Bibliotheksfenster** durch kurze und lange Beschreibungen ergänzt. Die Kurzbeschreibung wird immer rechts neben dem Funktionsnamen angezeigt, während die lange Beschreibung als Tooltip zu sehen ist, wenn Sie den Mauszeiger über die Kurzbeschreibung halten (*siehe Abbildung unten*).



Schritt 4. Definieren von Sprachimplementierungen

Wir können nun die Funktion im Fenster **Bibliotheken** mit der Funktion in der benutzerdefinierten Java-, C#- oder C++-Klasse verbinden. Dies wird über das Element `<implementation>` bewerkstelligt. Eine einzige Funktion kann mehrere `<implementation>`-Elemente haben - eines für jede unterstützte Programmiersprache. Eine Funktion kann in Java den Namen **`Hello`** oder in C++ den Namen **`SayHello`** haben. Aus diesem Grund müssen Sie für jede Programmiersprache einen eigenen Funktionsnamen definieren. Eine Funktion für jede einzelne der drei Programmiersprachen kann folgendermaßen aussehen.

```
<!-- ... -->
<component name="sayhello">
<!-- ... -->
   <implementations>
      <implementation language="cs">
         <function name="HelloFunction"/>
      </implementation>
      <implementation language="java">
         <function name="Hello"/>
      </implementation>
      <implementation language="cpp">
         <function name="SayHello"/>
      </implementation>
   </implementations>
<!-- ... -->
</component>
<!-- ... -->
```

Der als Funktionsname bereitgestellte Wert muss mit dem Namen der Methode in der Java-, C#- oder C++-Klasse übereinstimmen.

 © 2016-2022 Altova GmbH

6.5.4.2 Importieren der .mff-Datei in MapForce

Nachdem Sie eine benutzerdefinierte .mff-Datei erstellt[1006] haben, können Sie diese folgendermaßen in MapForce importieren:

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken[34] auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).



2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

3. Navigieren Sie zur benutzerdefinierten .mff-Datei und klicken Sie auf **Öffnen**.

Die importierte Bibliothek ist im Fenster "Bibliotheken" zu sehen, nachdem Sie als Mapping-Sprache die richtige Sprache für die benutzerdefinierte Bibliothek ausgewählt haben.

Wenn Sie die *.mff-Datei relativ zum Ordner **Programmdateien** (oder **Programmdateien (x86)** im Ordner **...\Altova\MapForce2023\MapForceLibraries** gespeichert haben, wird die Bibliothek automatisch beim Start von MapForce ins Fenster "Bibliotheken" geladen. Durch Löschen bzw. Hinzufügen der entsprechenden Bibliotheksdatei (*.mff) können Bibliotheken und ihre Funktionen ein- oder ausgeblendet werden.

6.5.4.3 Datentyp-Zuordnung

In der nachstehenden Tabelle sind die Datentypen aufgelistet, die als Funktionsrückgabetypen und Parametertypen unterstützt werden, wenn Sie benutzerdefinierte `.mff`-Dateien erstellen, die Ihre Java-, C#- und C++-Bibliotheken referenzieren. In der Tabelle sind sowohl native als auch nicht native Datentypen aufgelistet. Wenn Sie Unterstützung für nicht native Typen wie z.B. Altova-Datums-, Uhrzeit- und Zeitdauertypen benötigen, müssen Ihre benutzerdefinierten Java- und C#-Bibliotheken eine Referenz zu Altova-Bibliotheken enthalten. Im Fall von C++ müssen Altova-Bibliotheken immer importiert werden. Informationen zum Generieren der Altova-Bibliotheken finden Sie unter Code Generator[1563].

XML-Schema-Typ	Java-Typ	C#-Typ	C++-Typ
anyAtomicType	String	string	string_type
anySimpleType	String	string	string_type
anyURI	String	string	string_type
base64Binary	byte[]	byte[]	altova::mapforce::blob
boolean	boolean	bool	bool
byte	int	int	int
date	com.altova.types.DateTime 1684	Altova.Types.DateTime 1668	altova::DateTime 1652
dateTime	com.altova.types.DateTime 1684	Altova.Types.DateTime 1668	altova::DateTime 1652
dayTimeDuration	com.altova.types.Duration 1688	Altova.Types.Duration 1673	altova::Duration 1655
decimal	java.math.BigDecimal	decimal	double
double	double	double	double
duration	com.altova.types.Duration 1688	Altova.Types.Duration 1673	altova::Duration 1655
ENTITIES	String	string	string_type
ENTITY	String	string	string_type
float	double	double	double
gDay	com.altova.types.DateTime 1684	Altova.Types.DateTime 1668	altova::DateTime 1652
gMonth	com.altova.types.DateTime 1684	Altova.Types.DateTime 1668	altova::DateTime 1652
gMonthDay	com.altova.types.DateTime 1684	Altova.Types.DateTime 1668	altova::DateTime 1652
gYear	com.altova.types.DateTime 1684	Altova.Types.DateTime 1668	altova::DateTime 1652
gYearMonth	com.altova.types.DateTime 1684	Altova.Types.DateTime 1668	altova::DateTime 1652
hexBinary	byte[]	byte[]	altova::mapforce::blob
ID	String	string	string_type
IDREF	String	string	string_type

© 2016-2022 Altova GmbH

XML-Schema-Typ	Java-Typ	C#-Typ	C++-Typ
IDREFS	String	string	string_type
int	int	int	int
integer	java.math.BigInteger	decimal	__int64
language	String	string	string_type
long	long	long	__int64
Name	String	string	string_type
NCName	String	string	string_type
negativeInteger	java.math.BigInteger	decimal	__int64
NMTOKEN	String	string	string_type
NMTOKENS	String	string	string_type
nonNegativeInteger	java.math.BigInteger	decimal	unsigned __int64
nonPositiveInteger	java.math.BigInteger	decimal	__int64
normalizedString	String	string	string_type
NOTATION	String	string	string_type
positiveInteger	java.math.BigInteger	decimal	unsigned __int64
QName	javax.xml.namespace.QName	Altova.Types.QName	altova::QName
short	int	int	int
string	String	string	string_type
time	com.altova.types.DateTime 1684	Altova.Types.DateTime 1668	altova::DateTime 1652
token	String	string	string_type
unsignedByte	long	ulong	unsigned __int64
unsignedInt	long	ulong	unsigned __int64
unsignedLong	java.math.BigInteger	ulong	unsigned __int64
unsignedShort	long	ulong	unsigned __int64
untypedAtomic	String	string	string_type
yearMonthDuration	com.altova.types.Duration 1688	Altova.Types.Duration 1673	altova::Duration 1655

6.5.4.4 Referenzieren der C#-Bibliothek in .mff

In diesem Beispiel wird gezeigt, wie Sie eine C#-Beispielbibliothek erstellen und in einer MapForce Function File (`.mff`) referenzieren. Diese `.mff`-Datei kann anschließend als MapForce-Bibliothek importiert werden. Eine C#-Bibliothek in einer `.mff`-Datei zu referenzieren, ist nur eine der Methoden, um C#-Bibliotheken in MapForce zu importieren. Eine einfacher Alternative wäre der direkte Import von .NET Assemblys. Nähere Informationen dazu finden Sie unter Beispiel: Import einer benutzerdefinierten .NET DLL-Assembly 1003.

Konfigurationsschritte

Um eine C#-Bibliothek in einer `.mff`-Datei zu referenzieren, gehen Sie vor, wie unten beschrieben.

Anmerkung: Wenn Sie benutzerdefinierte NET-Funktionen in der integrierten Ausgabevorschau (im Fenster **Ausgabe**) verwenden möchten, müssen diese Funktionen für das .NET Framework 4.x oder :NET Standard 2.0 kompiliert werden.

Schritt 1. Erstellen einer neuen Klassenbibliothek in VS

Erstellen Sie ein neues Klassenbibliotheksprojekt in Visual Studio. Beachten Sie, dass die Funktion als `public static` definiert wurde.

```
namespace MyLibrary
{
    public class Greetings
    {
        public static string SayHello(bool isMorning)
        {
            if (isMorning)
                return "Good morning!";
            return "Good Day!";
        }
    }
}
```

Schritt 2. Hinzufügen einer Referenz zu Altova.dll

Wenn Sie spezielle XML-Schematypen (wie z.B. date und duration) benötigen, müssen Sie eine Referenz von Ihrem Visual Studio-Projekt zur `Altova.dll`-Bibliothek hinzufügen. Um diese Bibliothek zu erhalten, generieren Sie C#-Code anhand eines Mappings ohne benutzerdefinierte Funktionen. Sie finden die Datei `Altova.dll` relativ zum Verzeichnis, in dem der Code generiert wurde, im Verzeichnis **`..\Altova\bin\debug`**. Um in Visual Studio die Referenz zu `Altova.dll` hinzuzufügen, klicken Sie in Visual Studio im Menü **Projekt** auf **Verweis hinzufügen** und navigieren Sie zur Datei `Altova.dll`. Fügen Sie anschließend die folgenden Zeile zu Ihrem Code hinzu: **`using Altova.Types;`**. Informationen dazu, wie Sie XML-Schematypen C#-Typen zuordnen, finden Sie unter Datentyp-Zuordnung 1011.

Schritt 3. Erzeugen Ihres VS-Projekts mittels Build

Erzeugen Sie Ihr Visual Studio-Projekt mit Build. Die Datei `MyLibrary.dll` wird in Ihrem Projektausgabeverzeichnis generiert.

Schritt 4. Erstellen der .mff-Datei und Referenzieren Ihrer C#-Bibliothek

Erstellen Sie mit Hilfe eines XML-Editors eine neue `.mff`-Datei und validieren Sie diese anhand des Schemas **`C:\Programme\MapForceLibraries\mff.xsd`**. Stellen Sie sicher, dass alle Referenzen unter **`implementation language="cs"`** auf die korrekten C# Member und die zuvor erstellen Pfade verweisen. Die

 © 2016-2022 Altova GmbH

Zeile **`function name="SayHello"`** muss den Funktionsnamen außerdem genau, wie er in C# definiert war, referenzieren, siehe auch Konfigurieren der .mff-Datei. 1006

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping version="9" library="mylib" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
   <implementations>
      <implementation language="cs">
         <setting name="namespace" value="MyLibrary" />
         <setting name="class" value="Greetings" />
         <setting name="reference" value="C:
\Libraries\cs\MyLibrary\bin\debug\MyLibrary.dll" />
      </implementation>
   </implementations>
   <group name="string functions">
      <component name="sayhello">
         <sources>
            <datapoint name="ismorning" type="xs:boolean"/>
         </sources>
         <targets>
            <datapoint name="result" type="xs:string"/>
         </targets>
         <implementations>
            <implementation language="cs">
               <function name="SayHello"/>
            </implementation>
         </implementations>
         <description>
            <short>result = sayhello(ismorning)</short>
            <long>Returns "Good morning" or "Good day", depending on the input
parameter.</long>
         </description>
      </component>
   </group>
</mapping>
```

Schritt 5. Importieren der .mff-Datei als Bibliothek
Nachdem Sie Ihre benutzerdefinierte Bibliothek in der `.mff`-Datei referenziert haben, können Sie die `.mff`-Datei nun als Bibliothek in MapForce importieren. Nähere Informationen finden Sie unter Importieren von .mff-Dateien 1011.

6.5.4.5 Referenzieren von C++ in .mff

In diesem Beispiel wird gezeigt, wie Sie eine C++-Beispielbibliothek erstellen und in einer MapForce Function File (`.mff`) referenzieren. Diese `.mff`-Datei kann anschließend als MapForce-Bibliothek importiert werden.

Konfigurationsschritte

Um eine C++-Bibliothek in einer `.mff`-Datei zu referenzieren, gehen Sie vor, wie unten beschrieben.

Schritt 1. Erstellen einer Header-Datei
Erstellen Sie eine Header (.h)-Datei für Ihre Klassenbibliothek. Im folgenden Codefragment sehen Sie eine Beispiel-Header-Datei namens **Greetings.h**.

```
#ifndef MYLIBRARY_GREETINGS_H_INCLUDED
#define MYLIBRARY_GREETINGS_H_INCLUDED

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

using namespace altova;

namespace mylib {

class ALTOVA_DECLSPECIFIER Greetings
{
   public:
    static string_type SayHello(bool isMorning);
};

} // namespace mylib

#endif // MYLIBRARY_GREETINGS_H_INCLUDED
```

Beachten Sie, dass die Funktion als statisch deklariert wurde und dass der Namespace `altova` importiert wird. Vor den Klassennamen muss `ALTOVA_DECLSPECIFIER` gesetzt werden, damit Ihre Klassen - unabhängig davon, ob Sie im danach generierten Code dynamische oder statische Verknüpfungen verwenden, korrekt kompiliert werden können.

Schritt 2. Erstellen einer .cpp-Datei
Erstellen Sie eine .cpp-Datei mit demselben Namen wie die Header-Datei. Die .cpp-Datei muss sich im selben Verzeichnis wie die .h-Datei befinden. Im folgenden Codefragment sehen Sie eine .cpp-Beispieldatei namens **Greetings.cpp**, die die zuvor erstellte Datei **Greetings.h** inkludiert:

```
#include "StdAfx.h"
#include "../Altova/Altova.h"
#include "../Altova/AltovaException.h"
#include "../Altova/SchemaTypes.h"

#include "Greetings.h"

namespace mylib {

   string_type Greetings::SayHello(bool isMorning)
   {
      if( isMorning )
         return _T("Good morning!");
      return _T("Good day!");
   }
}
```

Beachten Sie die Zeilen, in denen **StdAfx.h** und eine Reihe weiterer Altova-Bibliotheken importiert werden. Diese Zeilen dürfen nicht geändert werden. Wenn die Pfade zu den Altova-Bibliotheken im generierten Code korrekt sind, verweisen sie auf die entsprechenden Dateien. Im Gegensatz zu Java oder C# müssen Sie Ihre

© 2016-2022 Altova GmbH

C++-Quelldateien nicht kompilieren. Sie werden in den generierten Code kopiert und mit dem restlichen generierten Mapping-Code kompiliert.

Schritt 3. Erstellen der .mff-Datei und Referenzieren Ihrer C++-Bibliothek
Erstellen Sie mit Hilfe eines XML-Editors eine neue `.mff`-Datei und validieren Sie diese anhand des Schemas **`C:\Programme\MapForceLibraries\mff.xsd`**. Der Namespace und die Funktionsnamen und Datentypen, die hier definiert sind, müssen, wie unter Konfigurieren der .mff-Datei[1006] beschrieben, mit denen im C++-Code übereinstimmen. Nähere Informationen dazu finden Sie unter Datentyp-Zuordnung[1011].

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping version="9" library="mylib" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
   <implementations>
      <implementation language="cpp">
         <setting name="namespace" value="mylib"/>
         <setting name="class" value="Greetings"/>
         <setting name="path" value="C:\Libraries\cpp"/>
         <setting name="include" value="Greetings.h"/>
         <setting name="source" value="Greetings.cpp"/>
      </implementation>
   </implementations>
   <group name="greetings">
      <component name="sayhello">
         <sources>
            <datapoint name="ismorning" type="xs:boolean"/>
         </sources>
         <targets>
            <datapoint name="result" type="xs:string"/>
         </targets>
         <implementations>
            <implementation language="cpp">
               <function name="SayHello"/>
            </implementation>
         </implementations>
         <description>
            <short>result = sayhello(ismorning)</short>
            <long>Returns "Good morning" or "Good day", depending on the input
parameter.</long>
         </description>
      </component>
   </group>
</mapping>
```

Schritt 4. Importieren der .mff-Datei als Bibliothek
Nachdem Sie Ihre benutzerdefinierte Bibliothek in der `.mff`-Datei referenziert haben, können Sie die `.mff`-Datei nun als Bibliothek in MapForce importieren. Nähere Informationen finden Sie unter Importieren der .mff-Datei[1011]
.

C++-Kompilierungsfehler
Um Mappings, in denen native C++-Bibliotheken verwendet werden, ausführen zu können, müssen Sie C++-Code generieren und das Mapping über Ihren C++-Code oder Ihre C++-Applikation ausführen, siehe Generieren von C++-Code[1567]. Wenn Sie in **`#import "msado15.dll" rename("EOF", "EndOfFile")`** einen

Kompilierungsfehler erhalten, ändern Sie die Projekteigenschaften, um eine Referenz auf **msado15.dll** in **C:\Programme\Common Files\System\ADO** zu inkludieren.

6.5.4.6 Referenzieren von Java in .mff

In diesem Beispiel wird gezeigt, wie Sie eine Java-Beispielbibliothek erstellen und in einer MapForce Function File (.mff) referenzieren. Diese .mff-Datei kann anschließend als MapForce-Bibliothek importiert werden. Eine Java-Bibliothek in einer .mff-Datei zu referenzieren, ist nur eine der Methoden, um Java-Bibliotheken in MapForce zu importieren. Eine einfachere Alternative wäre der direkte Import von Java .class-Dateien. Nähere Informationen dazu finden Sie unter Beispiel: Import einer benutzerdefinierten Java-Klasse 1001.

Konfigurationsschritte

Um eine C#-Bibliothek in einer .mff-Datei zu referenzieren, gehen Sie vor, wie unten beschrieben.

Schritt 1. Erstellen eines neuen Projekts

Erstellen Sie in der Entwicklungsumgebung Ihrer Wahl (z.B. in Eclipse) ein neues Java-Projekt.

Schritt 2. Hinzufügen des com.mylib-Pakets

Fügen Sie zum Projekt ein neues Paket namens **com.mylib** hinzu, welches aus einer Klasse namens **Greetings**besteht. Beachten Sie im Codefragment unten, dass die Funktion SayHello als public static definiert wurde.

```
package com.mylib;

public class Greetings {

   public static String SayHello ( boolean isMorning ) {
      if( isMorning )
         return "Good Morning!";
      return "Good Day!";
   }

}
```

Schritt 3. import com.altova.types

Wenn Sie in Ihrem Projekt Unterstützung für spezielle Schematypen wie date, time und duration benötigen, können Sie optional das Paket com.altova.types importieren. Um dieses Paket zu erhalten, generieren Sie Java-Code anhand eines Mappings ohne benutzerdefinierte Funktionen: **import com.altova.types.*;**.

Schritt 4. Kompilieren Ihrer benutzerdefinierten Bibliothek

Kompilieren Sie Ihre benutzerdefinierte Bibliothek zu einer Klassendatei und fügen Sie sie zum Java Classpath hinzu.

Schritt 5. Erstellen der .mff-Datei und Referenzieren Ihrer Java-Bibliothek

Erstellen Sie mit Hilfe eines XML-Editors eine neue .mff-Datei und validieren Sie diese anhand des Schemas **C:\Programme\MapForceLibraries\mff.xsd.** Stellen Sie sicher, dass alle Referenzen unter **implementation language="java"** auf die korrekten Java-Member und die zuvor erstellen Pfade verweisen. Die Zeile **function name="SayHello"** muss den Funktionsnamen außerdem genau, wie er in Java definiert war, referenzieren, siehe auch Konfigurieren der .mff-Datei. 1006

 © 2016-2022 Altova GmbH

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping version="9" library="custom" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
   <implementations>
      <implementation language="java">
         <setting name="package" value="com.mylib"/>
         <setting name="class" value="Greetings"/>
      </implementation>
   </implementations>
   <group name="greetings">
      <component name="sayhello">
         <sources>
            <datapoint name="ismorning" type="xs:boolean"/>
         </sources>
         <targets>
            <datapoint name="result" type="xs:string"/>
         </targets>
         <implementations>
            <implementation language="java">
               <function name="SayHello"/>
            </implementation>
         </implementations>
         <description>
            <short>result = sayhello(ismorning)</short>
            <long>Returns "Good morning" or "Good day", depending on the input
parameter.</long>
         </description>
      </component>
   </group>
</mapping>
```

Schritt 6. Importieren der .mff-Datei als Bibliothek

Nachdem Sie Ihre benutzerdefinierte Bibliothek in der `.mff`-Datei referenziert haben, können Sie die `.mff`-Datei nun als Bibliothek in MapForce importieren. Nähere Informationen finden Sie unter Importieren der .mff-Dateien 1011.

6.6 Regular Expressions

Sie können Regular Expressions ("regex") bei der Erstellung eines MapForce Mappings in folgenden Situationen verwenden:

- im **pattern**-Parameter der Funktionen **match-pattern**[1173] und **tokenize-regexp**[1116]
- um die Nodes, auf die eine Node-Funktion angewendet werden sollen, zu filtern. Nähere Informationen dazu finden Sie unter Bedingte Anwendung von Node-Funktionen und Standardwerten[961].
- um Text bei der Erstellung von MapForce FlexText-Vorlagen auf Basis eines Musters zu trennen, siehe FlexText und Regular Expressions[730].

Die Syntax und Semantik von Regular Expressions für XSLT und XQuery entspricht den in Appendix F von "XML Schema Part 2: Datatypes Second Edition" definierten Regeln.

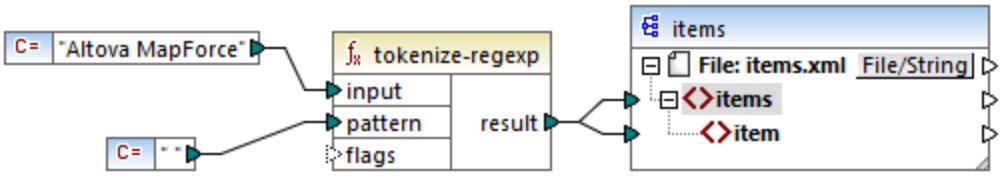
Anmerkung: Die komplexen Funktionalitäten der Regular Expression-Syntax können bei der Generierung von C++-, C#- oder Java-Code etwas unterschiedlich sein. Nähere Informationen dazu finden Sie in der regex-Dokumentation zu den einzelnen Sprachen.

Terminologie

Die grundlegende Terminologie von Regular Expressions wird hier anhand des Beispiels **tokenize-regexp** erläutert. Diese Funktion teilt Text mit Hilfe von Regular Expressions in eine Sequenz von Strings auf. Zu diesem Zweck erhält die Funktion die folgenden Input-Parameter:

input	Der Input-String, der von der Funktion verarbeitet werden soll. Die Regular Expression wird an diesem String angewendet.
pattern	Das eigentliche Muster der Regular Expression, die angewendet werden soll.
flags	Dies ist ein optionaler Parameter, der zusätzliche Optionen (Flags) definiert, die festlegen, wie die Regular Expression interpretiert werden soll, siehe "Flags" weiter unten.

Der Input-String im Mapping unten ist "Altova MapForce". Der Parameter **pattern** ist ein Leerzeichen. Es werden keine Regular Expression Flags verwendet.



Der Text wird dadurch überall dort, wo ein Leerzeichen steht, aufgeteilt, daher ist die Ausgabe des Mappings:

```
<items>
   <item>Altova</item>
   <item>MapForce</item>
</items>
```


© 2016-2022 Altova GmbH

Beachten Sie, dass die Funktion **tokenize-regexp** die übereinstimmenden Zeichen aus dem Ergebnis ausschließt. D.h. das Leerzeichen wird in diesem Beispiel aus der Ausgabe entfernt.

Das obige Beispiel ist sehr einfach gehalten. Dasselbe Ergebnis kann auch ohne Regular Expressions, mit Hilfe der Funktion tokenize 1111 erzielt werden. In einem realistischeren Szenario würde der Parameter **pattern** eine komplexere Regular Expression enthalten. Die Regular Expression kann aus beliebigen der folgenden Bestandteile bestehen:

- Literale
- Zeichenklassen
- Zeichenbereiche
- Negierte Klassen
- Metazeichen
- Quantifizierer

Literale

Literale dienen zum Finden von Zeichen in exakt dieser Schreibweise. Wenn der Input-String z.B. **abracadabra** lautet und **pattern** das Literal **br** ist, so ist die Ausgabe die folgende:

```
<items>
   <item>a</item>
   <item>acada</item>
   <item>a</item>
</items>
```

Der Grund dafür ist, dass das Literal **br** im Input-String **abracadabra** zwei Übereinstimmungen hat. Nachdem die übereinstimmenden Zeichen aus der Ausgabe entfernt wurden, wird die oben gezeigte Sequenz von drei Strings erzeugt.

Zeichenklassen

Wenn Sie eine Gruppe von Zeichen in eckige Klammern (**[** und **]**) setzen, wird dadurch eine Zeichenklasse erstellt. Es wird immer nur jeweils ein einziges der Zeichen innerhalb der Zeichenklasse gefunden, z.B:

- Mit dem Muster **[aeiou]** werden alle klein geschriebenen Vokale gefunden.
- Mit dem Muster **[mj]ust** werden "must" und "just" gefunden.

Anmerkung: Die Groß- und Kleinschreibung spielt beim Muster eine Rolle, d.h. ein kleingeschriebenes "a" stimmt nicht mit dem Großbuchstaben "A" überein. Damit Groß- und Kleinschreibung ignoriert werden, verwenden Sie das Flag **i**, siehe unten.

Zeichenbereiche

Mit **[a-z]** erstellen Sie einen Bereich zwischen den beiden Zeichen. Es wird immer nur jeweils eines der Zeichen gefunden. So wird etwa mit dem Muster **[a-z]** jeder Kleinbuchstabe zwischen "a" und "z" gefunden.

Negierte Klassen

Mit dem Zikumflexzeichen (`^`) als erstem Zeichen nach der öffnenden Klammer wird die Zeichenklasse negiert. Mit dem Muster `[^a-z]` wird z.B. jedes Zeichen außerhalb der Zeichenklasse, einschließlich Zeilenschaltungen (newline-Zeichen) gefunden.

Übereinstimmung mit jedem beliebigen Zeichen.

Mit Hilfe des Punkt-Metazeichens (`.`) wird jedes beliebige Einzelzeichen mit Ausnahme einer Zeilenschaltung (newline) gefunden. So wird z.B. mit dem Muster `.` jedes einzelne Zeichen gefunden.

Quantifizierer

Quantifizierer definieren in einer Regular Expression, wie oft das vorangestellte Zeichen oder der vorangestellte Unterausdruck vorkommen darf, damit eine Übereinstimmung gefunden wird.

?	Steht für null oder eine Instanz des direkt davor stehenden Elements. Das Muster `mo?` steht z.B. für "m" und "mo".
+	Steht für eine oder mehr Instanzen des direkt davor stehenden Elements. Das Muster `mo+` steht z.B. für "mo", "moo", "mooo", usw.
*****	Steht für null oder mehr Instanzen direkt davor stehenden Elements.
{min,max}	Steht für beliebig viele Instanzen zwischen *min* und *max*. So steht etwa das Muster `mo{1,3}` für "mo", "moo" und "mooo".

Klammern

Mit Hilfe von Klammern `(` und `)` werden Teile eines regex-Ausdrucks gruppiert. Sie können damit Quantifizierer auf einen Unterausdruck (anstatt auf nur ein Zeichen) anwenden. Oder Sie verwenden diese mit einer Alternative (siehe unten).

Alternative

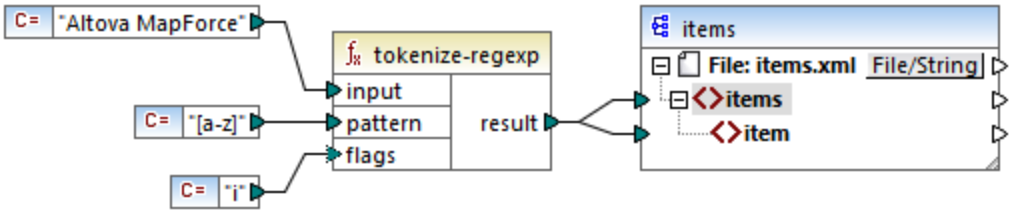
Der senkrechte Strich (Pipe-Zeichen) `|` bedeutet "oder". Er findet einen beliebigen von mehreren durch `|` getrennte Unterausdrücke. So findet etwa das Muster `(horse|make) sense` sowohl "horse sense" und "make sense".

Flags

Flags sind optionale Parameter, die definieren, wie die Regular Expression interpretiert werden soll. Jedes Flag entspricht einem Buchstaben. Die Buchstaben können in jeder Reihenfolge vorkommen und auch wiederholt werden.

s	Falls dieses Flag vorhanden ist, wird der Suchvorgang im "dot-all"-Modus durchgeführt. Wenn der Input-String "hello" und "world" in zwei *verschiedenen* Zeilen vorkommt, findet die Regular Expression `hello*world` nur dann eine Übereinstimmung, wenn das Flag **s** gesetzt wurde.

© 2016-2022 Altova GmbH

m	Wenn dieses Flag vorhanden ist, wird der Suchvorgang im mehrzeiligen Modus durchgeführt. Im mehrzeiligen Modus steht das Zirkumflexzeichen **^** für den Beginn jeder Zeile, d.h. den Beginn des gesamten Strings und das erste Zeichen nach einem "newline"-Zeichen. Das Dollarzeichen **$** steht für das Ende jeder Zeile, d.h. das Ende des gesamten Strings und das Zeichen unmittelbar vor einem "newline" Zeichen. Newline (Neue Zeile) ist das Zeichen **#x0A**.
i	Wenn dieses Flag vorhanden ist, wird die Groß- und Kleinschreibung ignoriert. So würden etwa mit der Regular Expression **[a-z]** plus dem **i** Flag alle Buchstaben von a-z und A-Z gefunden. 
x	Falls dieses Flag vorhanden ist, werden Leerzeichen und andere Whitespace-Zeichen vor dem Suchvorgang aus der Regular Expression entfernt. Whitespace-Zeichen sind **#x09**, **#x0A**, **#x0D** und **#x20**. **Anmerkung:** Whitespace-Zeichen innerhalb einer Zeichenklasse z.B. **[#x20]** werden nicht entfernt.

6.7 Referenz Funktionsbibliothek

In diesem Kapitel werden die vordefinierten MapForce-Funktionen, die im Fenster "Bibliotheken" 34 zur Verfügung stehen, nach Bibliothek geordnet, beschrieben. Welche Funktionsbibliotheken im Fenster **Bibliotheken** verfügbar sind, hängt von der ausgewählten Transformationssprache ab. Nähere Informationen über die Liste der verfügbaren Transformationssprachen finden Sie in diesem Kapitel 28.

Die folgenden Unterabschnitte enthalten Informationen über die Kompatibilität von Funktionen mit Transformationssprachen.

core functions (core-Funktionen)

Die nachfolgenden Listen enthalten eine Übersicht über die Kompatibilität von core-Funktionen mit Transformationssprachen.

core | aggregate functions (Aggregatfunktionen)

- **avg**, **max**, **max-string**, **min**, **min-string**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In.
- **count**, **sum**: alle Transformationssprachen.

core | conversion functions (Konvertierungsfunktionen)

- **boolean**, `string`, **number**: alle Transformationssprachen.
- **format-date**, **format-dateTime**, **format-time**: XSLT 2.0, XSLT 3.0, C#, C++, Java, Built-In.
- **format-number**: XSLT 1.0, XSLT 2.0, XSLT 3.0, C#, C++, Java, Built-In.
- **parse-date**, **parse-dateTime**, **parse-number**, **parse-time**: C#, C++, Java, Built-In.

core | file path functions (Dateipfadfunktionen)

Alle Dateipfadfunktionen sind mit allen Transformationssprachen kompatibel.

core | generator functions (Generierungsfunktionen)

Die Funktion **auto-number** steht für alle Transformationssprachen zur Verfügung.

core | logical functions (logische Funktionen)

Die logischen Funktionen sind mit allen Transformationssprachen kompatibel.

core | math functions (mathematische Funktionen)

- **add**, **ceiling**, **divide**, **floor**, **modulus**, **multiply**, **round**, **subtract**: alle Transformationssprachen.
- **round-precision**: C#, C++, Java, Built-In.

core | node functions (Node-Funktionen)

- **is-xsi-nil**, **local-name**, **static-node-annotation**, **static-node-name**: alle Transformationssprachen.
- **node-name**, **set-xsi-nil**, **substitute-missing-with-xsi-nil**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In.

core | QName functions (QName-Funktionen)

Die `QName`-Funktionen sind mit allen Transformationssprachen mit Ausnahme von XSLT 1.0 kompatibel.

© 2016-2022 Altova GmbH

core | sequence functions (Sequenzfunktionen)

- **exists**, **not-exists**, **position**, **substitute-missing**: alle Transformationssprachen.
- **distinct-values**, **first-items**, **generate-sequence**, **item-at**, **items-from-till**, **last-items**, **replicate-item**, **replicate-sequence**, **set-empty**, **skip-first-items**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In.
- **group-adjacent**, **group-by**, **group-ending-with**, **group-into-blocks**, **group-starting-with**: XSLT 2.0, XSLT 3.0, C#, C++, Java, Built-In.

core | string functions (String-Funktionen)

- **concat**, **contains**, **normalize-space**, **starts-with**, **string-length**, **substring**, **substring-after**, **substring-before**, **translate**: alle Transformationssprachen.
- **char-from-code**, **code-from-char**, **tokenize**, **tokenize-by-length**, **tokenize-regexp**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In.

bson functions (BSON-Funktionen) (nur MapForce Enterprise Edition)

Alle BSON-Funktionen sind nur mit Built-in kompatibel.

db functions (DB-Funktionen) (MapForce Professional und Enterprise Edition)

Die db-Funktionen sind mit C#, C++, Java, Built-In kompatibel.

edifact functions (EDIFACT-Funktionen) (nur MapForce Enterprise Edition)

Die edifact-Funktionen sind mit C#, C++, Java, Built-In kompatibel.

lang functions (MapForce Professional und Enterprise Edition)

Die nachfolgenden Listen enthalten eine Übersicht über die Kompatibilität von lang-Funktionen mit Transformationssprachen.

lang | datetime functions (Datums- und Uhrzeitfunktionen)

Die lang | datetime-Funktionen sind mit C#, C++, Java, Built-In kompatibel.

lang | file functions (Dateifunktionen)

Die Funktionen **read-binary-file** und **write-binary-file** sind nur mit Built-in kompatibel.

lang | generator functions (Generierungsfunktionen)

Die **create-guid**-Funktionen stehen für C#, C++, Java, Built-In zur Verfügung.

lang | logical functions (logische Funktionen)

Die lang | logical-Funktionen stehen für C#, C++, Java, Built-In zur Verfügung.

lang | math functions (mathematische Funktionen)

Die lang | math-Funktionen stehen für C#, C++, Java, Built-In zur Verfügung.

lang | QName functions (QName-Funktionen)

Die lang | QName-Funktionen stehen für C#, C++, Java, Built-In zur Verfügung.

lang | string functions (String-Funktionen)

- **charset-decode**, **charset-encode**: Built-In.
- **match-pattern**: C#, Java, Built-In.
- **capitalize**, **count-substring**, **empty**, **find-substring**, **format-guid-string**, **left**, **left-trim**, **lowercase**, **pad-string-left**, **pad-string-right**, **repeat-string**, **replace**, **reversefind-substring**, **right**, **right-trim**, **string-compare**, **string-compare-ignore-case**, **uppercase**: C#, C++, Java, Built-In.

mime functions (MIME-Funktionen) (nur MapForce Enterprise Edition)

Die `mime`-Funktionen stehen nur für Built-In zur Verfügung.

xbrl functions (XBRL-Funktionen) (nur MapForce Enterprise Edition)

Die `xbrl`-Funktionen sind mit C#, C++, Java, Built-In kompatibel.

xslx functions (XSLT-Funktionen) (nur MapForce Enterprise Edition)

Die `xlsx`-Funktionen sind mit XSLT 2.0, XSLT 3.0, C#, Java und Built-In kompatibel.

xpath2 functions (XPath2-Funktionen)

Alle `xpath2`-Funktionen sind mit XSLT 2.0, XSLT 3.0 und XQuery 1.0 kompatibel.

xpath3 functions (XPath3-Funktionen)

Alle `xpath3`-Funktionen sind nur mit XSLT 3.0 kompatibel.

xslt10 functions (XSLT10-Funktionen)

Die nachfolgenden Listen enthalten eine Übersicht über die Kompatibilität von `xslt10`-Funktionen mit Transformationssprachen.
xslt10 | xpath functions (XPath-Funktionen)

- **local-name**, **name**, **namespace-uri**: XSLT 1.0, XSLT 2.0 und XSLT 3.0.
- **lang**, **last**, **position**: XSLT 1.0.

xslt10 | xslt functions (XSLT-Funktionen)

- **generate-id**, **system-property**: XSLT 1.0, XSLT 2.0 und XSLT 3.0.
- **current**, **document**, **element-available**, **function-available**, **unparsed-entity-uri**: XSLT 1.0.

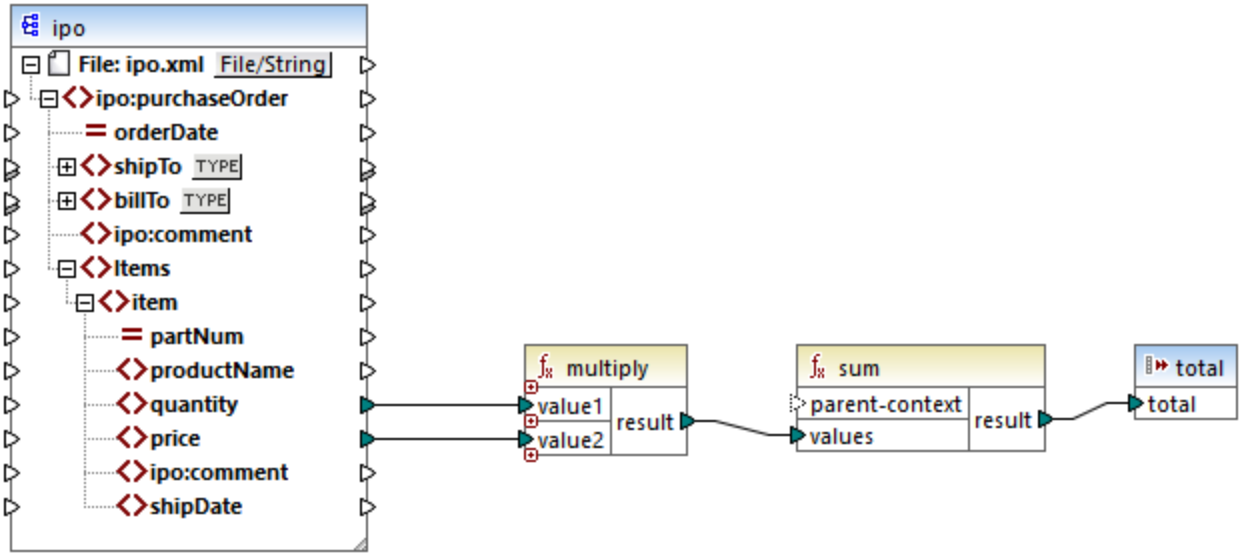
6.7.1 core | aggregate functions (Aggregatfunktionen)

"Aggregieren" bedeutet mehrere Werte desselben Typs zu verarbeiten, um ein einziges Resultat wie z.B. eine Summe, eine Anzahl oder einen Durchschnitt zu erhalten. Datenaggregationen können in MapForce mit Hilfe von Aggregatfunktionen wie `avg`, `count`, `max` und anderen durchgeführt werden.

Die folgenden beiden Argumente werden in allen Aggregatfunktionen verwendet:

© 2016-2022 Altova GmbH

1. **parent-context**. Dieses Argument ist optional. Sie können damit den Standard-Mapping-Kontext außer Kraft setzen (und dadurch den Geltungsbereich der Funktion oder die Werte, über die die Funktion iterieren muss, ändern). Ein Arbeitsbeispiel dazu finden Sie unter Beispiel: Ändern des Parent-Kontexts 1340.
2. **values**. Dieses Argument muss mit einem Quell-Datenelement, das die zu verarbeitenden Werte bereitstellt, verbunden sein. So erhält etwa die **sum**-Funktion im unten gezeigten Mapping als Input eine Sequenz numerischer Werte, die aus einer XML-Quelldatei stammen. Für jedes Datenelement in der XML-Quelldatei ermittelt die **multiply**-Funktion den Preis (price) des Datenelements mal Anzahl (quantity) und übergibt das Ergebnis an die **sum**-Funktion. Die **sum**-Funktion aggregiert alle Input-Werte und erzeugt eine Gesamtsumme, die auch den Output des Mappings bildet. Sie finden das Mapping im Ordner **MapForceExamples**.

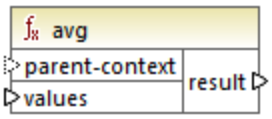


SimpleTotal.mfd

Einige Aggregatfunktionen wie `min`, `max`, `sum` und `avg` funktionieren ausschließlich mit numerischen Werten. Die Input-Daten dieser Funktionen werden für die Verarbeitung in den Datentyp **decimal** konvertiert.

6.7.1.1 avg

Gibt den Durchschnittswert aller Werte in der Input-Sequenz zurück. Der Durchschnittswert einer leeren Gruppe ist eine leere Gruppe.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parameter

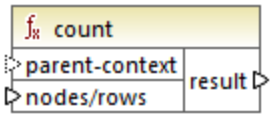
Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent context. Siehe auch Beispiel: Ändern des Parent-Kontexts 1340. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
values	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Beachten Sie, dass der bereitgestellte Argumentwert numerisch sein muss.

Beispiel

Siehe Beispiel: Gruppieren von Datensätzen nach Schlüssel 934

6.7.1.2 count

Gibt die Anzahl der einzelnen Datenelemente zurück, aus denen die Input-Sequenz besteht. Die Anzahl einer leeren Gruppe ist Null.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Beachten Sie, dass diese Funktion in XSLT 1.0 eingeschränkte Funktionen hat.

Parameter

Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 1340. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.

© 2016-2022 Altova GmbH

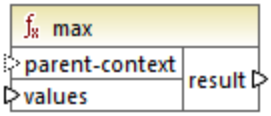
Argument	Beschreibung
nodes/rows	Dieses Argument muss mit einem Quelldatenelement verbunden sein, damit die Zählung erfolgen kann.

Beispiel

Siehe Beispiel: Ändern des Parent-Kontexts[1340], Beispiel: Zählen von Datenbanktabellenzeilen[891].

6.7.1.3 max

Gibt den Maximalwert aller numerischen Werte in der Input-Sequenz zurück. Der Maximalwert einer leeren Gruppe ist eine leere Gruppe.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parameter

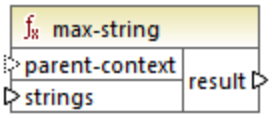
Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts[1340]. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
values	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Beachten Sie, dass der bereitgestellte Argumentwert numerisch sein muss. Um den Maximalwert einer Sequenz von Strings zu ermitteln, verwenden Sie die Funktion max-string[1030].

Beispiel

Siehe Beispiel: Gruppieren von Datensätzen nach Schlüssel[934]

6.7.1.4 max-string

Gibt den Maximalwert aller String-Werte in der Input-Sequenz zurück. So gibt `max-string("a", "b", "c")` z.B. `"c"` zurück. Die Funktion gibt eine leere Gruppe zurück, wenn das Argument **strings** eine leere Gruppe ist.



Sprachen

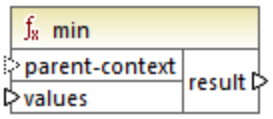
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 1340. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
strings	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Der bereitgestellte Argumentwert muss eine Sequenz (Null oder viele) von `xs:string` sein.

6.7.1.5 min

Gibt den Minimalwert aller numerischen Werte in der Input-Sequenz zurück. Der Minimalwert einer leeren Gruppe ist eine leere Gruppe.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

© 2016-2022 Altova GmbH

Parameter

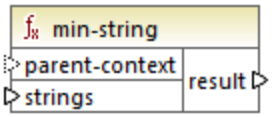
Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 1340. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
values	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Beachten Sie, dass der bereitgestellte Argumentwert numerisch sein muss. Um den Minimalwert einer Sequenz von Strings zu ermitteln, verwenden Sie die Funktion min-string 1031.

Beispiel

Siehe Beispiel: Gruppieren von Datensätzen nach Schlüssel 934

6.7.1.6 min-string

Gibt den Minimalwert aller String-Werte in der Input-Sequenz zurück. So gibt **`min-string("a", "b", "c")`** z.B. **`"a"`** zurück. Die Funktion gibt eine leere Gruppe zurück, wenn das Argument **strings** eine leere Gruppe ist.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

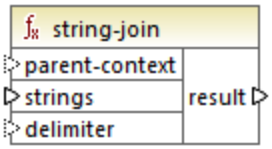
Parameter

Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 1340. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen,

Argument	Beschreibung
	an welcher Node-Gruppe die Funktion ausgeführt werden soll.
strings	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Der bereitgestellte Argumentwert muss eine Sequenz (Null oder viele) von `xs:string` sein.

6.7.1.7 string-join

Verkettet alle Werte der Input-Sequenz zu einem String, der durch ein beliebiges von Ihnen ausgewähltes Trennzeichen getrennt ist. Die Funktion gibt einen leeren String zurück, wenn das Argument **strings** eine leere Gruppe ist.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

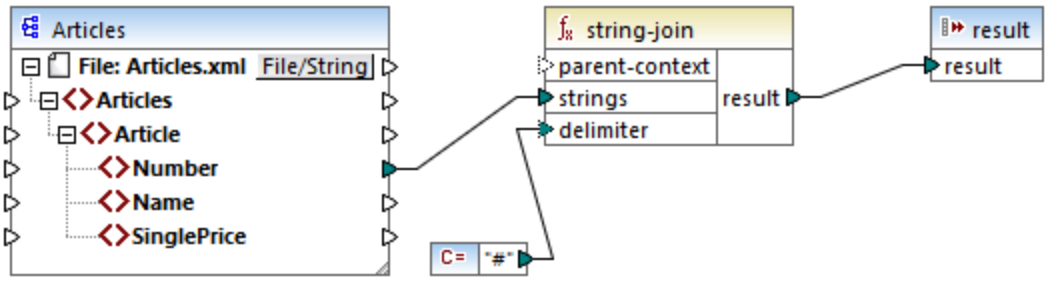
Parameter

Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 1340. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
strings	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Der bereitgestellte Argumentwert muss eine Sequenz (Null oder viele) von `xs:string` sein.
delimiter	Optionales Argument. Definiert, welches Trennzeichen zwischen zwei aufeinander folgende Strings eingefügt werden soll.

Beispiel

So enthält etwa die XML-Quelldatei vier **Article**-Datenelemente mit den folgenden Zahlen. 1, 2, 3 und 4.

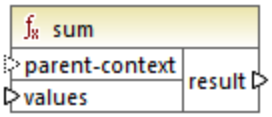
© 2016-2022 Altova GmbH



Die Konstante liefert als Trennzeichen eine Raute "#". Das Mapping-Ergebnis lautet daher **1#2#3#4**. Wenn Sie kein Trennzeichen angeben, wird das Ergebnis zu **1234**.

6.7.1.8 sum

Gibt die arithmetische Summe aller Werte in der Input-Sequenz zurück. Die Summe einer leeren Gruppe ist Null.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 1340. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
values	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Beachten Sie, dass der bereitgestellte Argumentwert numerisch sein muss.

Beispiel

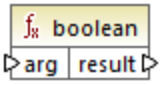
Siehe Beispiel: Summieren von Node-Werten 988.

6.7.2 core | conversion functions (Konvertierungsfunktionen)

Zur Unterstützung expliziter Datentypkonvertierungen steht in der Bibliothek **conversion** eine ganze Reihe von Konvertierungsfunktionen zur Verfügung. Beachten Sie, dass Konvertierungsfunktionen nicht immer notwendig sind, da MapForce die erforderlichen Konvertierungen in den meisten Fällen automatisch durchführt. Konvertierungsfunktionen dienen normalerweise zum Formatieren von Datums- und Uhrzeitwerten oder zum Vergleichen von Werten. Wenn einige Mapping-Datenelemente z.B. einen unterschiedlichen Typ haben (wie z.B. String und Integer), können Sie mit Hilfe der number[1043]-Konvertierungsfunktionen einen numerischen Vergleich erzwingen.

6.7.2.1 boolean

Konvertiert den Wert von **arg** in einen Booleschen Wert. Dies eignet sich für die Arbeit mit logischen Funktionen (wie z.B. `equal`, `greater`, usw.) sowie Filtern und if-else-Bedingungen[914]. Um den Booleschen Wert **false** zu erhalten, geben Sie als Argument einen leeren String oder den numerischen Wert 0 an. Um den Booleschen Wert **true** zu erhalten, geben Sie als Argument einen nicht leeren String oder den numerischen Wert 1 an.



Sprachen

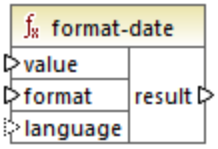
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
arg	Obligatorisches Argument. Liefert den konvertierende Wert.

6.7.2.2 format-date

Konvertiert einen Datumswert vom Typ `xs:date` in einen String und formatiert ihn gemäß den definierten Optionen.



© 2016-2022 Altova GmbH

Sprachen

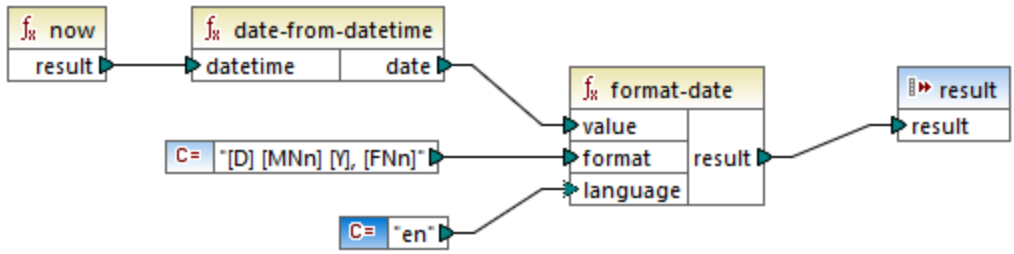
Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

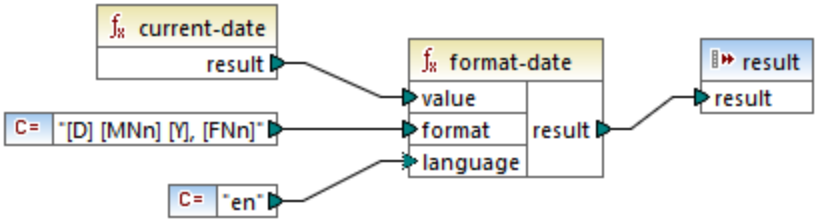
Argument	Beschreibung
value	Der zu formatierende `xs:date`-Wert
format	Ein Formatstring, der angibt, wie das Datum formatiert werden soll. Dieses Argument wird auf dieselbe Art, wie das Argument **format** in der Funktion format-dateTime[1036] verwendet.
language	Optionales Argument. Wird dieses Argument zur Verfügung gestellt, werden der Name des Monats und der Wochentag in einer bestimmten Sprache zurückgegeben. Gültige Werte sind: **de** Deutsch **en (Standardeinstellung)** Englisch **es** Spanisch **fr** Französisch **ja** Japanisch

Beispiel

Im folgenden Mapping wird das aktuelle Datum in einem Format wie "25 March 2020, Wednesday" ausgegeben. Um diesen Wert ins Spanische zu übersetzen, setzen Sie den Wert des Arguments **language** auf **es**.

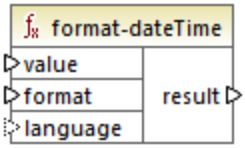


Beachten Sie, dass das obige Mapping für die Transformationssprachen Built-in, C++, C# oder Java erstellt wurde. In XSLT 2.0 kann dasselbe Ergebnis mit dem folgenden Mapping erzielt werden:



6.7.2.3 format-dateTime

Konvertiert einen Wert vom Typ `xs:dateTime` in einen String. Die String-Darstellung von Datum und Uhrzeit wird entsprechend dem Wert des Arguments **format** formatiert.



Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value	Der zu formatierenden `xs:dateTime`-Wert.
format	Ein Formatstring, der definiert, wie **value** formatiert werden soll. Siehe Anmerkungen weiter unten.
language	Optionales Argument. Wird dieses Argument zur Verfügung gestellt, werden der Name des Monats und der Wochentag in einer bestimmten Sprache zurückgegeben. Gültige Werte: **de** Deutsch **en (Standardeinstellung)** Englisch **es** Spanisch **fr** Französisch **ja** Japanisch

© 2016-2022 Altova GmbH

Anmerkung: Wenn die Ausgabe der Funktion (result) mit einem Node verbunden ist, dessen Typ nicht "string" ist, kann die Formatierung verloren gehen, da der Wert in den Zieltyp konvertiert wird. Diese automatische Konvertierung kann durch Deaktivieren des Kontrollkästchens **Zielwerte in Zieltypen konvertieren** in den Komponenteneinstellungen[82] der Zielkomponente deaktiviert werden.

Anmerkungen

Das Argument **format** besteht aus einem String, der so genannte Variablen-Marker enthält, die innerhalb von eckigen Klammern stehen, z.B. `[J]/[M]/[T]`. Zeichen außerhalb von eckigen Klammern sind Literalzeichen. Wenn das Ergebnis eckige Klammern als Literalzeichen enthalten soll, so sollten diese eckigen Klammern verdoppelt werden.

Jeder Variablen-Marker besteht aus einem Komponenten-Specifier, der angibt, welche Komponente des Datums oder der Uhrzeit angezeigt werden soll, einem optionalen Formatierungs-Modifier, einem weiteren optionalen Darstellungs-Modifier und einem optionalen Breiten-Modifier. Vor diesen Modifiern steht, falls vorhanden, ein Komma.

```
format := (literal | argument)*
argument := [component(format)?(presentation)?(width)?]
width := , min-width ("-" max-width)?
```

Die Komponenten sind:

Specifier	Beschreibung	Standarddarstellung
Y	Jahr (absoluter Wert)	vier Stellen (2010)
M	Monat des Jahres	1-12
D	Tag des Monats	1-31
d	Tag des Jahres	1-366
F	Wochentag	Name des Tages (sprachabhängig)
W	Woche des Jahres	1-53
w	Woche des Monats	1-5
H	Stunde (24 Stunden)	0-23
h	Stunde (12 Stunden)	1-12
P	A.M. oder P.M.	alphabetisch (sprachabhängig)
m	Minuten in Stunde	00-59
s	Sekunden in Minute	00-59
f	Sekundenbruchteile	numerisch, eine Dezimalstelle
Z	Zeitzone, als Zeitabstand von UTC	+08:00
z	Zeitzone als Zeitabstand von GMT	GMT+n

Als Formatierungs-Modifier kann eines der folgenden Zeichen verwendet werden:

Zeichen	Beschreibung	Beispiel
1	Numerisches Dezimalformat ohne vorangestellte Null:	1, 2, 3
01	Dezimalformat, zwei Stellen:	01, 02, 03
N	Name der Komponente, Großbuchstaben[1]	MONTAG, DIENSTAG
n	Name der Komponente, Kleinbuchstaben[1]	monday, tuesday
Nn	Name der Komponente, beginnend mit einem Großbuchstaben[1]	Montag, Dienstag

Fußnoten:

1. Die Modifier **N**, **n** und **Nn** werden nur von den folgenden Komponenten unterstützt: **M**, **d**, **D**.

Der Breiten-Modifier wird, falls erforderlich, durch ein Komma, gefolgt von einer Ziffer, die für die Mindestbreite steht, eingefügt. Optional können Sie einen Bindestrich, gefolgt von einer weiteren Ziffer, die die Maximalbreite angibt, hinzufügen. Zum Beispiel:

- **`[D,2]`** ist der Tag des Monats mit einer vorangestellten Null (zwei Stellen).
- **`[MNn,3-3]`** ist der Name des Monats in Form einer Abkürzung bestehend aus drei Buchstaben, z.B. *Jan*, *Feb*, *Mar* usw.

Beispiele

In der nachstehenden Tabelle sehen Sie einige Beispiele für die Formatierung von `xs:dateTime`-Werten mit Hilfe der **`format-dateTime`**-Funktion. In der Spalte "Wert" finden Sie den Wert, der für das Argument **value** bereitgestellt wird. In der Spalte "Format" ist der Wert des Arguments **format** angegeben. In der Spalte "Ergebnis" finden Sie das Ergebnis der Funktion.

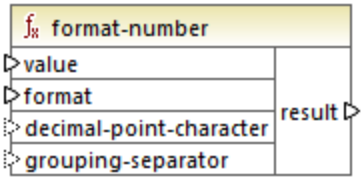
Wert	Format	Ergebnis
2003-11-03T00:00:00	`[D]/[M]/[Y]`	3/11/2003
2003-11-03T00:00:00	`[Y]-[M,2]-[D,2]`	2003-11-03
2003-11-03T00:00:00	`[Y]-[M,2]-[D,2] [H,2]:[m]:[s]`	2003-11-03 00:00:00
2010-06-02T08:02	`[Y] [MNn] [D01] [F,3-3] [d] [H]:[m]:[s].[f]`	2010 June 02 Wed 153 8:02:12.054
2010-06-02T08:02	`[Y] [MNn] [D01] [F,3-3] [d] [H]:[m]:[s].[f] [z]`	2010 June 02 Wed 153 8:02:12.054 GMT+02:00

© 2016-2022 Altova GmbH

Wert	Format	Ergebnis
2010-06-02T08:02	`[Y] [MNn] [D1] [F] [H]:[m]:[s].[f] [Z]`	2010 June 2 Wednesday 8:02:12.054 +02:00
2010-06-02T08:02	`[Y] [MNn] [D] [F,3-3] [H01]:[m]:[s]`	2010 June 2 Wed 08:02:12

6.7.2.4 format-number

Konvertiert eine Zahl in einen String und formatiert ihn gemäß den definierten Optionen.



Sprachen

Built-in, C++, C#, Java, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value	Obligatorisches Argument. Liefert die zu formatierende Zahl.
format	Obligatorisches Argument. Liefert einen Formatierungsstring, der festlegt, wie die Zahl formatiert werden soll. Siehe Anmerkungen weiter unten.
decimal-point-format	Optionales Argument. Liefert das Zeichen, das für das Dezimalzeichen, also im Deutschen das Komma, verwendet wird. Der Standardwert ist der Punkt (.).
grouping-seperator	Optionales Argument. Liefert das Trennzeichen, das zur Trennung von Tausenderstellen verwendet werden soll. Der Standardwert ist das Komma (,).

Anmerkung: Wenn die Ausgabe der Funktion (result) mit einem Node verbunden ist, dessen Typ nicht "string" ist, kann die Formatierung verloren gehen, da der Wert in den Zieltyp konvertiert wird. Diese automatische Konvertierung kann durch Deaktivieren des Kontrollkästchens **Zielwerte in Zieltypen konvertieren** in den Komponenteneinstellungen 82 der Zielkomponente deaktiviert werden.

Anmerkungen

Das Argument **format** hat die folgende Form:

```
format := subformat (;subformat)?
  subformat := (prefix)? integer (.fraction)? (suffix)?
  prefix := any characters except special characters
  suffix := any characters except special characters
  integer := (#)* (0)* ( allowing ',' to appear)
  fraction := (0)* (#)* (allowing ',' to appear)
```

Das erste *subformat* dient zum Formatieren positiver Zahlen, das zweite zum Formatieren negativer Zahlen. Wenn nur ein *subformat* definiert ist, so wird dasselbe Unterformat auch für negative Zahlen verwendet, wobei aber vor dem *prefix* ein Minuszeichen steht.

Sonderzeichen	Standardwert	Beschreibung
zero-digit	0	Eine Stelle, die immer an dieser Stelle im Ergebnis angezeigt wird
digit	#	An dieser Stelle erscheint im Ergebnisstring eine Ziffer, es sei denn, es handelt sich um eine nicht benötigte vorangestellte oder nachgestellte Null.
decimal-point	.	Trennt den Ganzzahlenbereich in der Zahl von den Kommastellen.
grouping-seperator	,	Trennt Zifferngruppen (Tausender-Trennzeichen).
percent-sign	%	Multipliziert die Zahl mit 100 und zeigt sie als Prozentwert an
per-mille	‰	Multipliziert die Zahl mit 1000 und zeigt sie als Promilwert an

In der Tabelle unten finden Sie Beispiele für Formatierungsstrings und deren Ergebnis.

Anmerkung: Die für die Funktion **format-number** verwendete Rundungsmethode ist, Aufrunden ab der Hälfte, d.h. wenn der Wert hinter dem Komma größer oder gleich, 0,5 ist, so wird aufgerundet. Wenn der Wert hinter dem Komma kleiner als 0,5 ist, wird abgerundet. Diese Rundungsmethode gilt nur für generierten Programmcode und für den Built-in-Ausführungsprozessor. In XSLT 1.0 ist die Rundungsmethode nicht definiert. In XSLT 2.0 ist die Rundungsmethod die mathematisch unverzerrte Rundung (half-to-even), also die Auf- oder Abrundung auf die nächste gerade Ziffer.

Zahl	Formatierungsstring	Ergebnis
1234.5	#,##0.00	1,234.50
123.456	#,##0.00	123.46
1000000	#,##0.00	1,000,000.00
-59	#,##0.00	-59.00
1234	###0.0###	1234.0

© 2016-2022 Altova GmbH

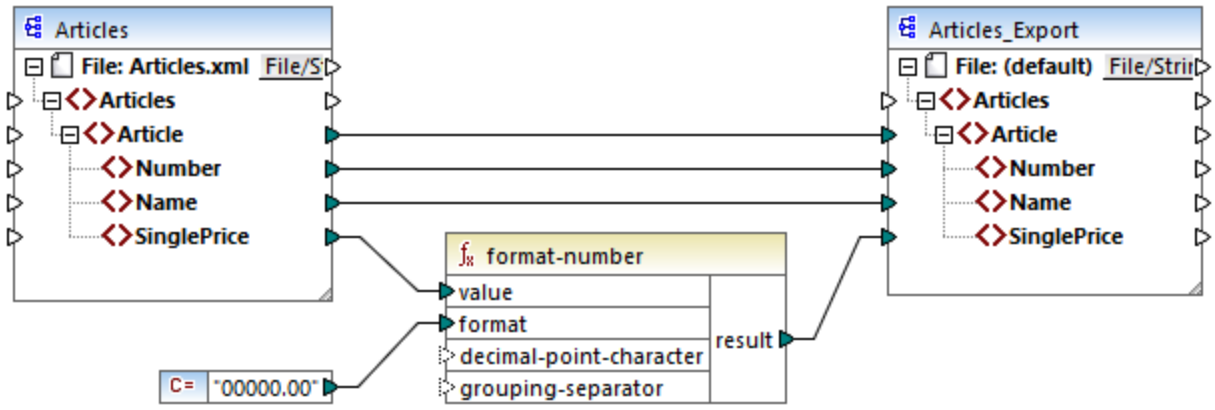
Zahl	Formatierungsstring	Ergebnis
1234.5	###0.0###	1234.5
.00025	###0.0###	0.0003
.00035	###0.0###	0.0004
0.25	#00%	25%
0.736	#00%	74%
1	#00%	100%
-42	#00%	-4200%
-3.12	#.00;(#.00)	(3.12)
-3,12	#.00;#.00CR	3.12CR

Beispiel

Im unten gezeigten Mapping werden Daten aus der XML-Quelldatei gelesen und in eine XML-Zieldatei geschrieben. Die Quellkomponente enthält mehrere **SinglePrice**-Elemente, die die folgenden Dezimalwerte enthalten: **25**, **2.30**, **34**, **57.50**. Das Mapping hat zwei Aufgaben:

1. Allen Werte links Nullen voranstellen, so dass der wichtige Teil genau 5 Stellen hat.
2. Allen Werte rechts Nullen hinzufügen, so dass der Kommastellenteil genau 2 Stellen hat.

Zu diesem Zweck wurde der Formatierungsstring `00000.00` als Argument für die Funktion `format-number` bereitgestellt.



PreserveFormatting.mfd

Die Werte in der Zielkomponente wurden folglich zu:

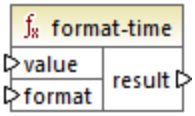
```
00025.00
00002.30
00034.00
```

```
00057.50
```

Sie finden die Mapping-Design-Datei unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2023\MapForceExamples\PreserveFormatting.mfd.

6.7.2.5 format-time

Konvertiert einen `xs:time`-Eingabewert in einen String.



Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

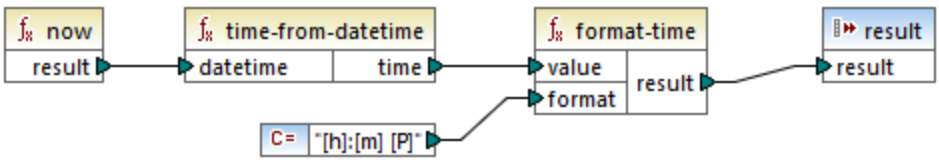
Parameter

Argument	Beschreibung
value	Obligatorisches Argument. Der zu formatierenden `xs:time`-Wert.
format	Obligatorisches Argument. Liefert einen Formatierungsstring. Dieses Argument wird auf dieselbe Art, wie das Argument **format** in der Funktion format-dateTime[1036] verwendet.

Beispiel

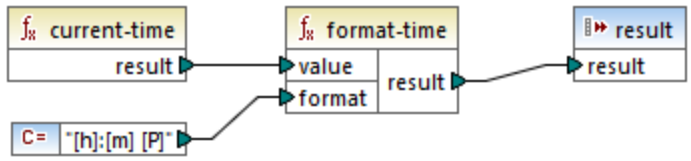
Im folgenden Mapping wird das aktuelle Datum in einem Format wie **`2:15 p.m.`** ausgegeben. Zu diesem Zweck wird der Formatierungsstring **`[h]:[m] [P]`** verwendet, wobei:

- **[h]** die aktuelle Stunde im 12-Stunden-Format ist.
- **[m]** die aktuelle Minute ist
- **[P]** den Teil "a.m." oder "p.m." darstellt



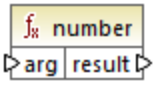
Beachten Sie, dass das obige Mapping für die Transformationssprachen Built-in, C++, C# oder Java erstellt wurde. In XSLT 2.0 kann dasselbe Ergebnis mit dem folgenden Mapping erzielt werden:

© 2016-2022 Altova GmbH



6.7.2.6 number

Konvertiert den Wert von **arg** in eine Zahl, wobei **arg** ein String oder ein Boolescher Wert ist. Wenn **arg** ein String ist, versucht MapForce ihn als Zahl zu parsen. So wird z.B. ein String wie `"12.56"` in den Dezimalwert `12.56` konvertiert. Wenn **arg** der Boolesche Wert **true** ist,. wird er in das numerische **1** konvertiert. Wenn **arg** der Boolesche Wert **false** ist, wird er in das numerische **0** konvertiert.



Sprachen

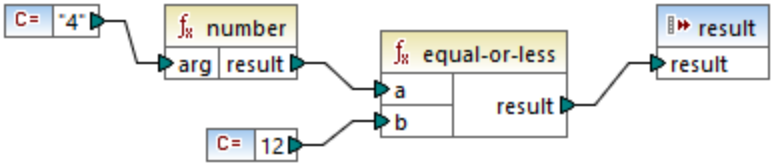
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
arg	Obligatorisches Argument. Liefert den zu konvertierenden Wert.

Beispiel

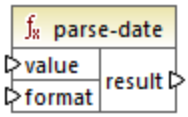
Im Beispiel unten hat die erste Konstante den Typ `string` und enthält den String "4". Die zweite Konstante enthält die numerische Konstante 12. Um die beiden Werte als Zahlen vergleichen zu können, muss der Typ übereinstimmen.



Durch Hinzufügen einer `number`-Funktion zur ersten Konstante wird der String "4" in den numerischen Wert 4 konvertiert. Das Ergebnis des Vergleich ist dann "true". Würde die Funktion `number` nicht verwendet (d.h. wenn "4" direkt mit **a** verbunden würde), würde es zu einem String-Vergleich kommen und das Ergebnis wäre "false".

6.7.2.7 parse-date

Konvertiert einem String in ein Datum. Für diese Funktion wird als Grundlage die Funktion parse-dateTime[1045] verwendet, wobei die Uhrzeitkomponente ignoriert wird. Das Ergebnis hat den Typ `xs:date`.



Sprachen

Built-in, C++, C#, Java.

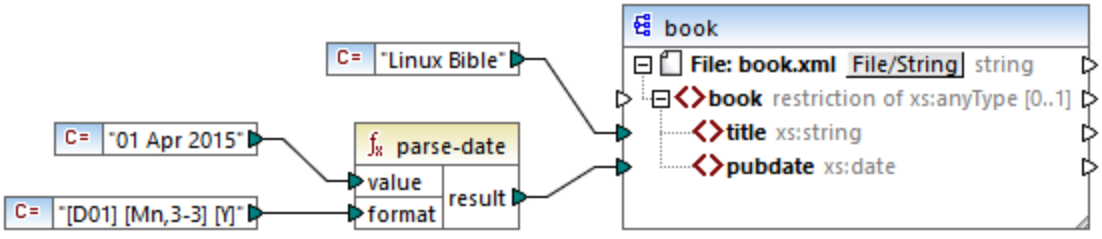
Parameter

Argument	Beschreibung
value	Obligatorisches Argument. Liefert den zu konvertierenden String-Wert.
format	Obligatorisches Argument. Liefert einen Formatierungsstring. Dieses Argument wird auf dieselbe Art, wie das Argument **format** in der Funktion parse-dateTime[1045] verwendet.

Beispiel

Im unten gezeigten Mapping wird der String **`"01 Apr 2015"`**, geparst, in ein Datum konvertiert und das Ergebnis wird in ein Ziel-Datenelement (**pubdate**) vom Typ `xs:date` geschrieben. Dazu wurde das Format **`[D01] [MNn,3-3] [Y]`** verwendet, wobei:

- **`[D,2]`** der Tag des Monats in Form einer zweistelligen Zahl ist
- **`[MNn,3-3]`** der Name des Monats mit einer Mindest- und Maximalbreite von 3 Zeichen ist
- **`[Y]`** das Jahr ist



Das Ergebnis ist das folgende (ausschließlich der XML- und Namespace-Deklaration):

```
<book>
   <title>Linux Bible</title>
```

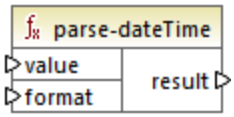

© 2016-2022 Altova GmbH

```
   <pubdate>2015-04-01</pubdate>
</book>
```

Siehe auch Beispiel: Mappen von Daten von einem RSS Feed 644.

6.7.2.8 parse-dateTime

Konvertiert einen als String ausgedrückten Datums- und Uhrzeitwert in einen Wert vom Typ `xs:dateTime`.



Sprachen

Built-in, C++, C#, Java.

Parameter

Argument	Beschreibung
value	Der zu konvertierende Stringwert.
format	Definiert die Formatmaske, die auf **value** angewendet werden soll.

Anmerkungen

Eine Formatmaske kann aus den folgenden Komponente bestehen:

Komponente	Beschreibung	Standarddarstellung
Y	Jahr (absoluter Wert)	vier Stellen (2010)
M	Monat des Jahres	1-12
D	Tag des Monats	1-31
d	Tag des Jahres	1-366
H	Stunde (24 Stunden)	0-23
h	Stunde (12 Stunden)	1-12
P	A.M. oder P.M.	alphabetisch (sprachabhängig)
m	Minuten in Stunde	00-59

Komponente	Beschreibung	Standarddarstellung
s	Sekunden in Minute	00-59
f	Sekundenbruchteile	numerisch, eine Dezimalstelle
Z	Zeitzone, als Zeitabstand von UTC	+08:00
z	Zeitzone als Zeitabstand von GMT	GMT+n

Einige der obigen Komponenten erhalten Modifikatoren (z.B. um ein Datum entweder als ein- oder zweistellige Zahl darzustellen):

Modifikator	Beschreibung	Beispiel
1	Numerisches Dezimalformat ohne vorangestellte Null: 1, 2, 3, ...	1, 2, 3
01	Dezimalformat, zwei Stellen: 01, 02, 03, ...	01, 02, 03
N	Name der Komponente, Großbuchstaben	FEBRUARY, MARCH
n	Name der Komponente, Kleinbuchstaben	february, march
Nn	Name der Komponente, beginnend mit einem Großbuchstaben	February, March

Anmerkung: Die Modifikatoren*N*, *n* und *Nn* unterstützen nur die Komponente *M* (Monat).

Der Breiten-Modifier wird, falls erforderlich, durch ein Komma, gefolgt von einer Ziffer, die für die Mindestbreite steht, eingefügt. Optional können Sie einen Bindestrich, gefolgt von einer weiteren Ziffer, die die Maximalbreite angibt, hinzufügen. Zum Beispiel:

- **`[D,2]`** ist der Tag des Monats mit einer vorangestellten Null (zwei Stellen).
- **`[MNn,3-3]`** ist der Name des Monats in Form einer Abkürzung bestehend aus drei Buchstaben, z.B. *Jan*, *Feb*, *Mar* usw.

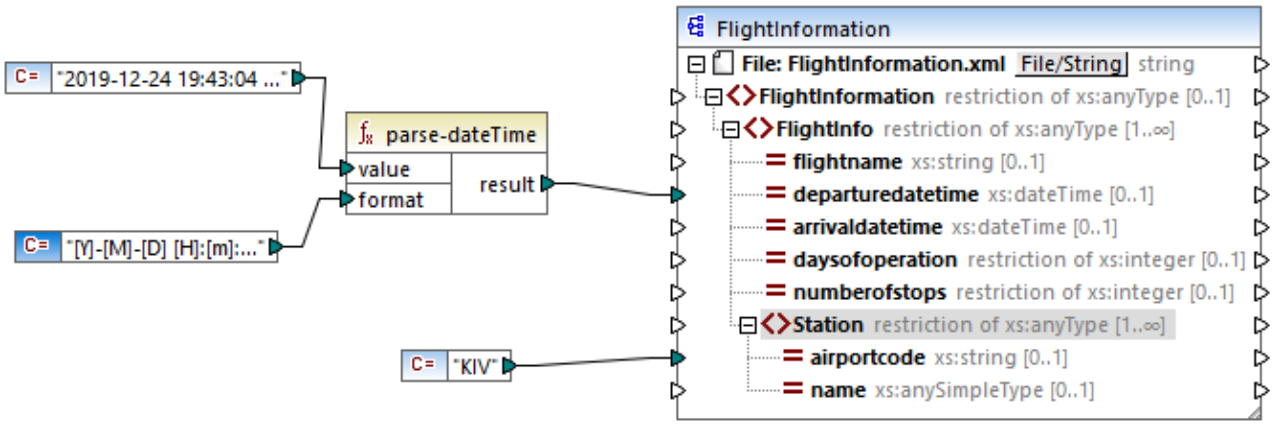
Die Tabelle unten enthält einige weitere Formatbeispiele:

Wert	Format	Ergebnis
21-03-2002 16:21:12.492 GMT+02:00	`[D]-[M]-[Y] [H]:[m]:[s].[f] [z]`	2002-03-21T16:21:12.492+02:00
315 2004 +01:00	`[d] [Y] [Z]`	2004-11-10T00:00:00+01:00
1.December.10 03:2:39 p.m. +01:00	`[D].[MNn].[Y,2-2] [h]:[m]:[s] [P] [Z]`	2010-12-01T15:02:39+01:00
20110620	`[Y,4-4][M,2-2][D,2-2]`	2011-06-20T00:00:00

© 2016-2022 Altova GmbH

Beispiel

In unten gezeigten Mapping wird der String-Wert `2019-12-24 19:43:04 +02:00` durch Anwendung der Formatmaske `[Y]-[M]-[D] [H]:[m]:[s] [Z]` in sein `dateTime`-Pendant konvertiert.



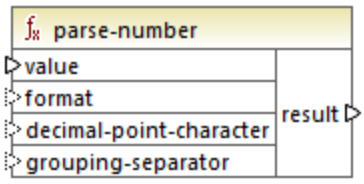
Das Ergebnis ist das folgende (ausschließlich der XML- und Namespace-Deklaration):

```
<FlightInformation>
   <FlightInfo departuredatetime="2019-12-24T19:43:04+02:00">
      <Station airportcode="KIV"/>
   </FlightInfo>
</FlightInformation>
```

Siehe auch Beispiel: Schreiben von Daten in Protocol Buffers 854.

6.7.2.9 parse-number

Die Funktion **parse-number** (*siehe unten*) konvertiert einen String gemäß dem definierten Muster in eine Dezimalzahl. Das Muster wird verwendet, um nur das Präfix, das Suffix und die Zahlengruppierung zu ermitteln. Die eigentliche Länge der Zahl wird nicht anhand des Musters überprüft. Wenn der Input-Wert länger oder kürzer als vom Muster vorgegeben ist, wird dies bei der Überprüfung ignoriert.



Sprachen

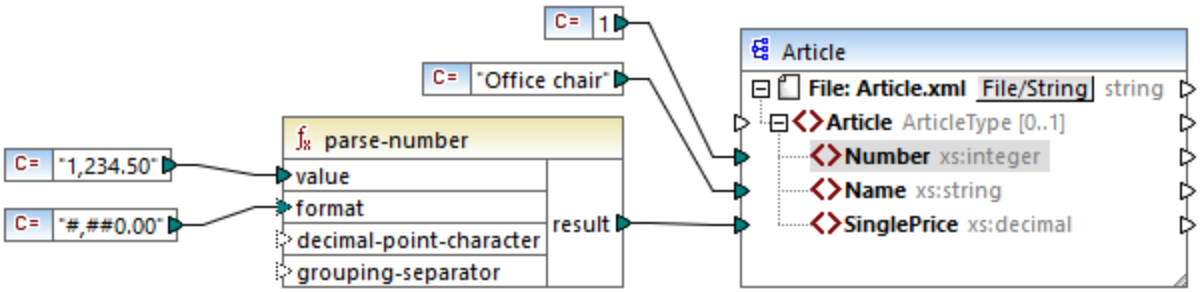
Built-in, C++, C#, Java.

Parameter

Argument	Beschreibung
`value`	Der in eine Zahl zu konvertierende String.
`format`	Optionales Argument. Ein Formatierungsstring, der angibt, wie die Zahl derzeit formatiert ist. Der Formatierungsstring ist derselbe, der auch in format-number[1039] verwendet wird. Die Standardeinstellung ist "#,##0.#"
`decimal-point-character`	Optionales Argument. Definiert das Zeichen, das für das Dezimalzeichen, also im Deutschen das Komma, verwendet wird. Standardeinstellung ist das Zeichen ".".
`grouping-seperator`	Optionales Argument. Das für separate Zahlengruppen zu verwendenden Trennzeichen. Standardeinstellung ist das Zeichen ",".

Beispiel

Im folgenden Mapping wird der String-Wert **`"1,234.50"`** mit Hilfe der Formatmaske **`#,##0.00`** zum entsprechenden Dezimalwert geparst. Die Argumente `decimal-point-character` und `grouping-separator` müssen in diesem Mapping nicht verbunden werden, da deren Standardwerte dem Format des Input-String entsprechen.



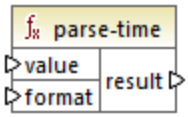
Die Ausgabe sieht folgendermaßen aus (ausschließlich der XML- und der Namespace-Deklaration):

```
<Article>
   <Number>1</Number>
   <Name>Office chair</Name>
   <SinglePrice>1234.5</SinglePrice>
</Article>
```


© 2016-2022 Altova GmbH

6.7.2.10 parse-time

Konvertiert einem String in einen `xs:time`-Wert. Für diese Funktion wird als Grundlage die Funktion parse-dateTime[1045] verwendet, wobei die Datumskomponente ignoriert wird.



Sprachen

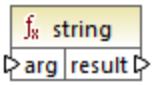
Built-in, C++, C#, Java.

Parameter

Argument	Beschreibung
value	Obligatorisches Argument. Liefert den zu konvertierenden String-Wert.
format	Obligatorisches Argument. Liefert einen Formatierungsstring. Dieses Argument wird auf dieselbe Art, wie das Argument **format** in der Funktion parse-dateTime[1045] verwendet.

6.7.2.11 string

Konvertiert einen Input-Wert in einen String. Sie können mit dieser Funktion auch den Textinhalt eines Node abrufen. Wenn es sich beim Input-Node um einen XML-complexType handelt, so werden alle untergeordneten Nodes ebenfalls als einzelner String ausgegeben.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

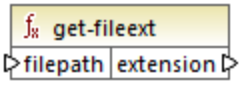
Argument	Beschreibung
arg	Obligatorisches Argument. Liefert den zu konvertierenden Wert.

6.7.3 core | file path functions (Dateipfadfunktionen)

Mit Hilfe von **file path**-Funktionen können Sie Dateipfaddaten wie Ordner, Dateinamen und Dateierweiterungen direkt aufrufen und für die weitere Verarbeitung in Ihren Mappings bearbeiten. Diese Funktionen können in allen Sprachen, die von MapForce unterstützt werden, verwendet werden.

6.7.3.1 get-fileext

Gibt die Erweiterung des Dateipfads einschließlich des Punktzeichens "." zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
filepath	Obligatorisches Argument. Gibt den zu verarbeitenden Dateipfad an.

Beispiel

Wenn Sie "c:\data\Sample.mfd" als Argument angeben, ist das Ergebnis `.mfd`.

6.7.3.2 get-folder

Gibt den Ordnernamen des Dateipfads einschließlich des Schrägstrichs oder umgekehrten Schrägstrichs zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

© 2016-2022 Altova GmbH

Parameter

Argument	Beschreibung
filepath	Obligatorisches Argument. Gibt den zu verarbeitenden Dateipfad an.

Beispiel

Wenn Sie "c:\data\Sample.mfd" als Argument angeben, ist das Ergebnis `c:\data\`.

6.7.3.3 main-mfd-filepath

Gibt den vollständigen Pfad der Mapping-Design-Datei (.mfd), die das Hauptmapping enthält, zurück. Wenn die mfd-Datei derzeit noch nicht gespeichert ist, wird ein leerer String zurückgegeben.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

6.7.3.4 mfd-filepath

Wenn die Funktion im Hauptmapping aufgerufen wird, gibt die Funktion dasselbe wie die Funktion main-mfd-filepath 1051 zurück, also den vollständigen Dateipfad der mfd-Datei, die das Haupt-Mapping enthält. Wenn die mfd-Datei derzeit noch nicht gespeichert ist, wird ein leerer String zurückgegeben. Wenn die Funktion von einer *importierten* benutzerdefinierten Funktion aus aufgerufen wird, gibt sie den vollständigen Pfad der *importierten* mfd-Datei, die die Definition der benutzerdefinierten Funktion enthält, zurück.

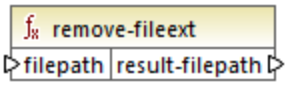


Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

6.7.3.5 remove-fileext

Entfernt die Erweiterung des Dateipfads einschließlich des Punkts.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

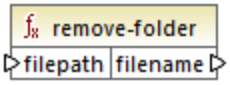
Argument	Beschreibung
filepath	Obligatorisches Argument. Gibt den zu verarbeitenden Dateipfad an.

Beispiel

Wenn Sie "c:\data\Sample.mfd" als Argument angeben, ist das Ergebnis `c:\data\Sample`.

6.7.3.6 remove-folder

Entfernt das Verzeichnis des Dateipfads einschließlich des nachgestellten oder umgekehrten Schrägstrichs.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
filepath	Obligatorisches Argument. Gibt den zu verarbeitenden Dateipfad an.

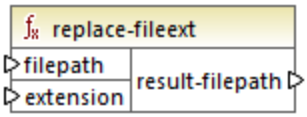
Beispiel

Wenn Sie "c:\data\Sample.mfd" als Argument angeben, ist das Ergebnis `Sample.mfd`.

© 2016-2022 Altova GmbH

6.7.3.7 replace-fileext

Ersetzt die durch den **filepath**-Parameter bereitgestellte Erweiterung des Dateipfads durch die Erweiterung, die durch die Verbindung zum **extension**-Parameter bereitgestellt wird.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

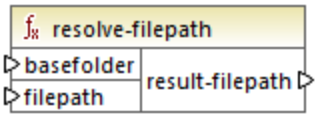
Argument	Beschreibung
filepath	Obligatorisches Argument. Gibt den zu verarbeitenden Dateipfad an.
extension	Obligatorisches Argument. Gibt die neue Erweiterung an, die verwendet werden soll.

Beispiel

Wenn Sie "c:\data\Sample.log" als **filepath** und ".txt" als **extension** angeben, ist das Ergebnis `c:\data\Sample.txt`.

6.7.3.8 resolve-filepath

Löst einen relativen Dateipfad anhand eines Basisordners auf. Die Funktion unterstützt '.' (aktuelles Verzeichnis) und '..' (übergeordnetes Verzeichnis).



Sprachen

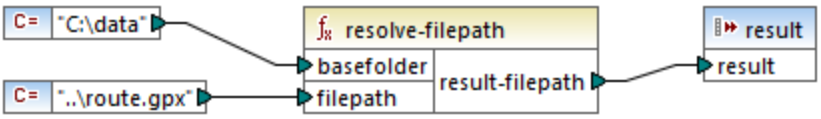
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
basefolder	Obligatorisches Argument. Gibt das Basisverzeichnis an, relativ zu dem der Pfad aufgelöst werden soll. Dabei kann es sich um einen absoluten oder einen relativen Pfad handeln.
filepath	Obligatorisches Argument. Gibt den aufzulösenden relativen Dateipfad an.

Beispiele

Im unten gezeigten Mapping wird der relative Dateipfad **..\route.gpx** anhand des Verzeichnisses **C:\data** aufgelöst.



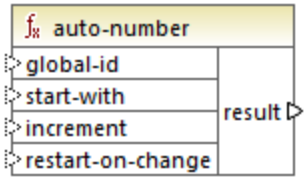
Das Ergebnis des Mappings ist `C:\route.gpx`.

6.7.4 core | generator functions (Generierungsfunktionen)

Die **core / generator** Funktionsbibliothek enthält Funktionen zum Generieren von Werten.

6.7.4.1 auto-number

Generiert Ganzzahlen in einer Sequenz (z.B. 1,2,3,4,...). Mit Hilfe von Parametern können Sie eine Anfangszahl (Ganzzahl), den Inkrementierungswert und andere Optionen angeben.



Die genaue Reihenfolge, in der Funktionen vom generierten Mapping-Code aufgerufen werden, ist nicht definiert. MapForce kann die berechneten Ergebnisse zur Wiederverwendung im Cache aufbewahren oder Ausdrücke in jeder beliebigen Reihenfolge auswerten. Im Gegensatz zu anderen Funktionen gibt die Funktion `auto-number` ein anderes Ergebnis zurück, wenn sie mehrmals mit denselben Input-Parametern aufgerufen wird. Es wird daher dringend empfohlen, die Funktion `auto-number` mit Vorsicht zu verwenden. In einigen Fällen erzielen Sie mit der Funktion position 1096 dasselbe Ergebnis.

© 2016-2022 Altova GmbH

Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

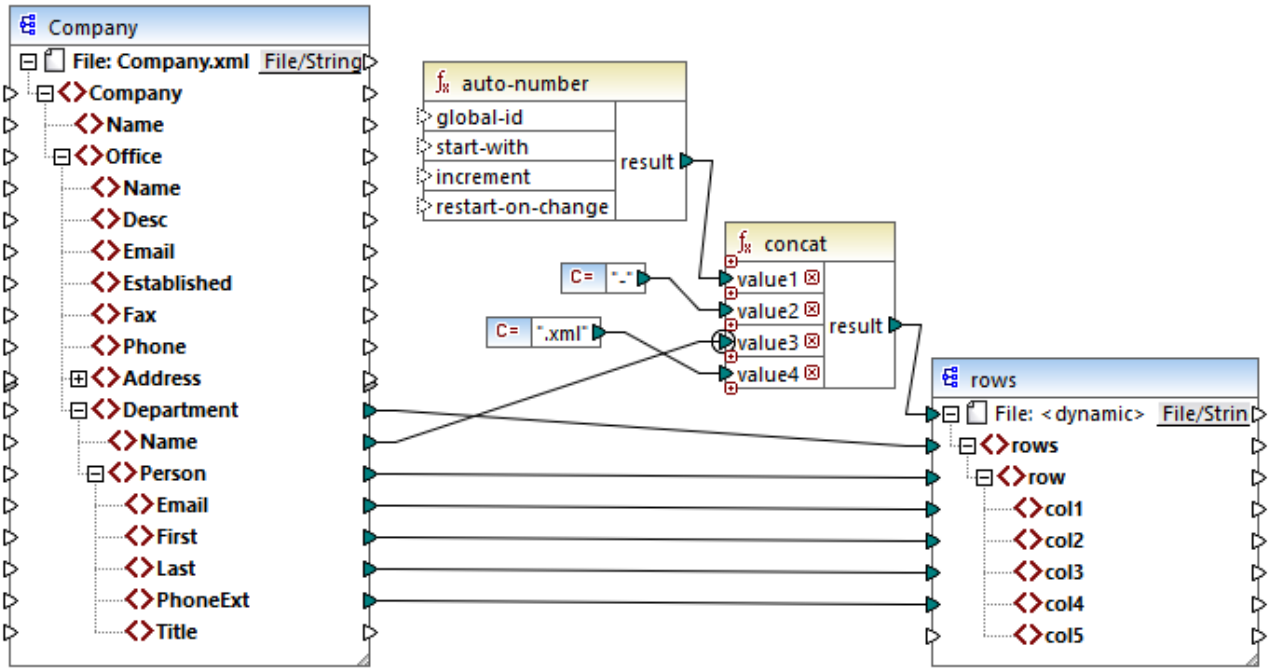
Parameter

Argument	Beschreibung
global-id	Optionaler Parameter. Wenn ein Mapping mehrere `auto-number`-Funktionen enthält, generieren diese Sequenzen mit doppelt vorhandenen Zahlen. Damit alle `auto-number`-Funktionen voneinander wissen und dadurch keine Sequenzen mit doppelten Werten generieren, verbinden Sie einen gemeinsamen String (z.B. eine Konstante) mit dem **global-id**-Input jeder `auto-number`-Funktion.
start-with	Optionaler Parameter. Definiert die Ganzzahl, mit der die generierte Sequenz beginnt. Der Standardwert ist **1**.
increment	Optionaler Parameter. Definiert den Inkrementierungswert. Der Standardwert ist **1**.
restart-on-change	Optionaler Parameter. Setzt den Zähler auf **start-with** zurück, wenn sich der Inhalt des damit verbundenen Datenelements ändert.

Beispiel

Das folgende Mapping ist eine Variante des im Beispiel: Ändern des Parent-Kontexts 1340 beschriebenen Mappings **ParentContext.mfd**.

Ziel des unten gezeigten Mappings ist es, mehrere XML-Dateien zu generieren, eine für jede Abteilung (department) in der XML-Quelldatei. Es gibt einige Abteilungen mit demselben Namen (da diese zu unterschiedlichen übergeordneten Büros gehören). Aus diesem Grund muss jeder generierte Dateiname mit einer sequenziellen Nummerierung beginnen, z.B. **1-Administration.xml**, **2-Marketing.xml**, usw.



Um das gewünschte Mapping-Resultat zu erhalten, wurde die Funktion **auto-number** verwendet. Das Ergebnis dieser Funktion wird mit Hilfe eines Bindestrichs, gefolgt vom Abteilungsnamen, gefolgt vom String ".xml" verkettet, um für die generierte Datei einen eindeutigen Namen zu erstellen. Beachten Sie, dass auf den dritten Parameter der **concat**-Funktion (den Abteilungsnamen) ein Prioritätskontext[1344] angewendet wurde. Dadurch wird die Funktion **auto-number** im Kontext jeder einzelnen Abteilung aufgerufen und erzeugt die erforderlichen Sequenzwerte. Würde kein Prioritätskontext verwendet, würde die Funktion **auto-number** (in Abwesenheit irgendeines Kontexts) immer wieder die Zahl 1 generieren, wodurch doppelt vorhandene Dateinamen generiert würden.

6.7.5 core | logical functions (logische Funktionen)

Logische Funktionen dienen (im Allgemeinen) dazu, Input-Daten miteinander zu vergleichen und als Ergebnis den Boolschen Wert "true" oder "false" zurückzugeben. Normalerweise werden Daten damit überprüft, bevor sie mittels eines Filters[914] an eine Untergruppe der Zielkomponente übergeben werden. Nahezu alle logischen Funktionen haben die folgende Struktur:

Input-Parameter = **a** | **b** oder **value1** | **value2**
Output-Parameter = result

Das ausgewertete Ergebnis hängt sowohl von den Eingabewerten als auch von den für den Vergleich verwendeten Datentypen ab. Der Vergleich "kleiner als" der Integerwerte **4** und **12** ergibt den Booleschen Wert "true", da 4 kleiner als 12 ist. Wenn die beiden Input-Parameter die String-Werte **4** und **12** enthalten, ist das Ergebnis der lexikalischen Analyse der Output-Wert "false", da '4' alphabetisch größer als das erste Zeichen '1' des zweiten Operanden (12) ist.

Wenn es sich bei allen Input-Werten um denselben Datentyp handelt, wird der Vergleich für den gemeinsamen Typ durchgeführt. Wenn die Input-Werte unterschiedliche Typen haben (z.B. `integer` und `string` oder `string`

© 2016-2022 Altova GmbH

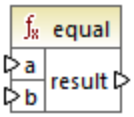
und `date`), wird als Datentyp für den Vergleich der allgemeingültigste (am wenigsten restriktive) der beiden verwendet.

Bevor zwei Werte unterschiedlichen Typs verglichen werden, werden alle Werte in einen gemeinsamen Datentyp konvertiert. Um beim vorhergehenden Beispiel zu bleiben: Der Datentyp `string` ist weniger restriktiv als `integer`. Wenn Sie den Integer-Wert **4** mit dem String **"12"**' vergleichen, wird der Integer-Wert **4** in den String **"4"** konvertiert, der anschließend mit dem String**"12"** verglichen wird.

Anmerkung: Logische Funktionen können nicht zum Prüfen des Vorhandenseins von Null-Werten verwendet werden. Wenn Sie einen Null-Wert als Argument für eine logische Funktion bereitstellen, gibt diese einen Null-Wert zurück. Nähere Informationen zur Behandlung von Nullwerten finden Sie unter Nullwerte / Nillable Werte 132.

6.7.5.1 equal

Die `equal`-Funktion (*siehe Abbildung unten*) gibt den Booleschen Wert `true` zurück, wenn **a** gleich **b**; false ist; gibt andernfalls `false` zurück. Die Groß- und Kleinschreibung wird beim Vergleich berücksichtigt.



Beispiel:

```
a = hi
b = hi
```

In diesem Beispiel sind beide Werte gleich. Daher ist das Ergebnis `true`. Wenn `b` z.B. gleich `Hi` wäre, wäre das Ergebnis der Funktion `false`.

Sprachen

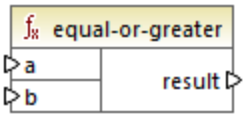
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

6.7.5.2 equal-or-greater

Gibt den Booleschen Wert **true** zurück, wenn *a* größer oder gleich *b* ist; gibt andernfalls **false** zurück.



Sprachen

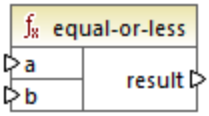
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

6.7.5.3 equal-or-less

Gibt den Booleschen Wert **true** zurück, wenn *a* kleiner oder gleich *b* ist; gibt andernfalls **false** zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

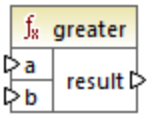
Parameter

Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

© 2016-2022 Altova GmbH

6.7.5.4 greater

Gibt den Booleschen Wert **true** zurück, wenn *a* größer als *b* ist; gibt andernfalls **false** zurück.



Sprachen

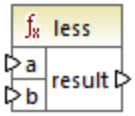
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

6.7.5.5 less

Gibt den Booleschen Wert **true** zurück, wenn *a* kleiner als *b* ist; gibt andernfalls **false** zurück.



Sprachen

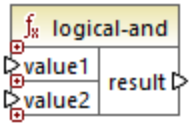
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

6.7.5.6 logical-and

Gibt den Booleschen Wert **true** nur dann zurück, wenn jeder Input-Wert true ist; gibt andernfalls **false** zurück. Sie können das Ergebnis mit einer weiteren **logical-and**-Funktion verbinden und dadurch eine beliebig gewählte Anzahl von Bedingungen mit einem logischen UND verbinden, um zu überprüfen, ob alle **true** zurückgeben. Diese Funktion außerdem erweitert werden und zusätzliche Argumente erhalten, siehe Hinzufügen oder Löschen von Funktionsargumenten 944.



Sprachen

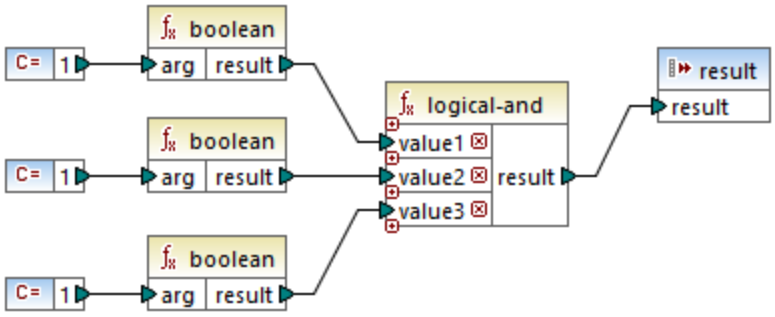
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
value2	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

Beispiel

Das Ergebnis im unten gezeigten Mapping ist **true**, da alle Input-Werte für die **logical-and**-Funktion ebenfalls **true** sind. Wäre einer der Input-Werte **false**, wäre auch das Ergebnis des Mappings **false**.

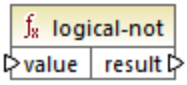


Siehe auch Beispiel: Look-up und Verkettung 980.

6.7.5.7 logical-not

Invertiert oder spiegelt das logische Ergebnis des Input-Werts. Wenn *value* z.B. **true** ist, ist das Ergebnis der "false". Wenn *value* **false** ist, ist das Ergebnis "true".

© 2016-2022 Altova GmbH



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

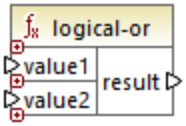
Parameter

Argument	Beschreibung
value	Obligatorischer Parameter. Liefert den Input-Wert.

6.7.5.8 logical-or

Bei dieser Funktion müssen beide Input-Werte Boolesche Werte sein. Wenn zumindest einer der Input-Werte **true** ist, ist das Ergebnis **true**. Andernfalls ist das Ergebnis **false**.

Diese Funktion kann außerdem erweitert werden und zusätzliche Argumente erhalten, siehe Hinzufügen oder Löschen von Funktionsargumenten 944.



Sprachen

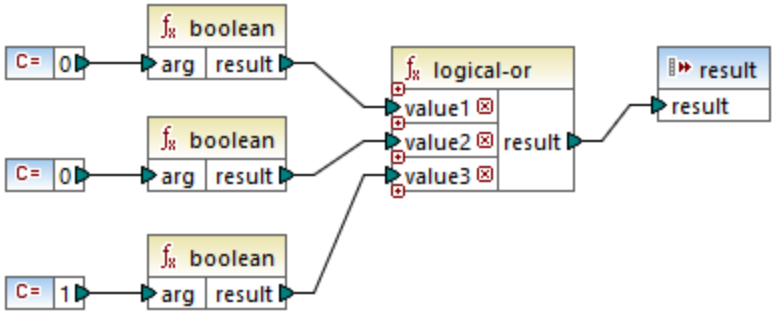
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
value2	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

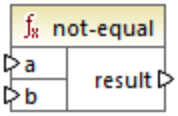
Beispiel

Das Ergebnis des unten gezeigten Mappings ist **true**, da mindestens eines der Argumente der Funktion **true** ist.



6.7.5.9 not-equal

Gibt den Booleschen Wert **true** zurück, wenn *a* nicht gleich *b* ist; gibt andernfalls **false** zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

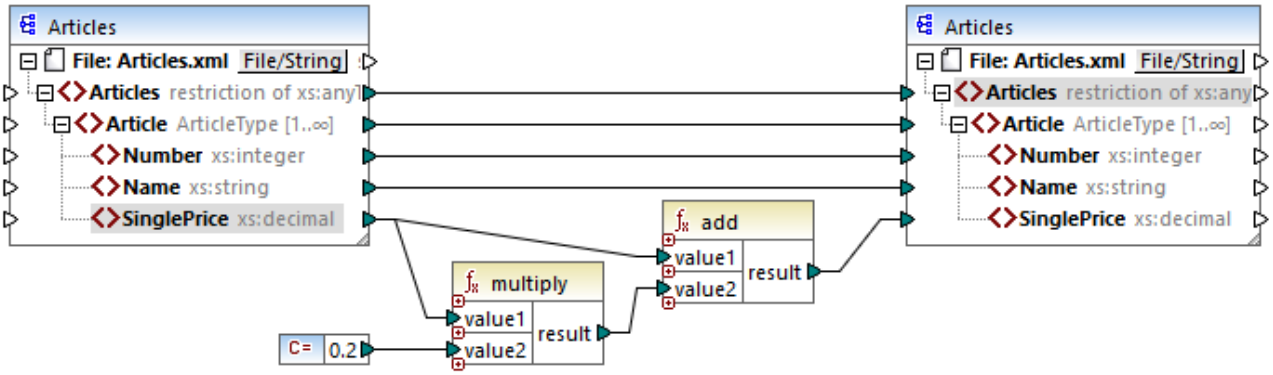
Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

6.7.6 core | math functions (mathematische Funktionen)

Mathematische Funktionen dienen zur Durchführung grundlegender mathematischer Operationen an Daten. Beachten Sie, dass diese Funktionen nicht zur Berechnung einer Zeitdauer oder zur Berechnung von `datetime`-Werten verwendet werden können.

Die meisten mathematischen Funktionen erhalten zwei Input-Parameter (**value1**, **value2**), die Operanden der mathematischen Operation sind. Die Input-Werte werden zur weiteren Verarbeitung automatisch in Werte vom Typ `decimal` (Dezimalwerte) konvertiert. Das Ergebnis von mathematischen Funktionen hat ebenfalls den Typ `decimal`.

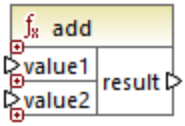
© 2016-2022 Altova GmbH



Im obigen Beispiel werden 20 % Umsatzsteuer zu den einzelnen auf die Zielkomponente gemappten Artikeln hinzugefügt.

6.7.6.1 add

Addiert **value1** zu **value2** und gibt das Ergebnis als Dezimalwert zurück. Diese Funktion kann außerdem erweitert werden und zusätzliche Argumente erhalten, siehe Hinzufügen oder Löschen von Funktionsargumenten 944.



Sprachen

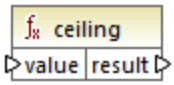
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten Operanden an.
value2	Obligatorischer Parameter. Gibt den zweiten Operanden an.

6.7.6.2 ceiling

Gibt die kleinste Ganzzahl zurück, die größer oder gleich **value** ist, zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

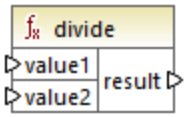
Argument	Beschreibung
value	Obligatorischer Parameter. Liefert den Input-Wert der Funktion.

Beispiel

Wenn der Input-Wert **11.2** ist, wird das Ergebnis durch Anwendung der `ceiling`-Funktion zu **12**, d.h. zur kleinsten Ganzzahl, die größer als **11.2** ist.

6.7.6.3 divide

Dividiert **value1** durch **value2** und gibt das Ergebnis als Dezimalwert zurück. Die Präzision des Ergebnisses hängt von der Zielsprache ab. Mit Hilfe der round-precision[1067] Funktion können Sie die Präzision des Ergebnisses definieren.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten Operanden an.
value2	Obligatorischer Parameter. Gibt den zweiten Operanden an.

© 2016-2022 Altova GmbH

6.7.6.4 floor

Gibt die größte Ganzzahl zurück, die kleiner oder gleich **value** ist, zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

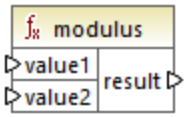
Argument	Beschreibung
value	Obligatorischer Parameter. Liefert den Input-Wert der Funktion.

Beispiel

Wenn der Input-Wert **11.7** ist, wird das Ergebnis durch Anwendung der `floor`-Funktion zu **11**, d.h. zur größten Ganzzahl, die kleiner als **11.7** ist.

6.7.6.5 modulus

Gibt den Rest, der sich bei der Division von **value1** durch **value2** ergibt, zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten Operanden an.
value2	Obligatorischer Parameter. Gibt den zweiten Operanden an.

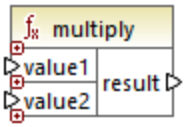
Beispiel

Wenn die Input Werte **1.5** und **1** sind, ist das Ergebnis der `modulus`-Funktion **0.5**. Die Erklärung dafür ist, dass bei `1.5 / 1` der Rest **0.5** bleibt.

Wenn die Input-Werte **9** und **3** sind, ist das Ergebnis **0**, da bei `9 / 3` kein Rest bleibt.

6.7.6.6 multiply

Multipliziert **value1** mit **value2** und gibt das Ergebnis als Dezimalwert zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten Operanden an.
value2	Obligatorischer Parameter. Gibt den zweiten Operanden an.

6.7.6.7 round

Gibt den auf die nächste Ganzzahl gerundeten Wert zurück. Wenn sich der Wert genau zwischen zwei Ganzzahlen befindet, wird der Algorithmus "Rundung Richtung positiv unendlich" verwendet. So wird z.B. der Wert "10,5" auf "11" und der Wert "-10,5" auf "-10" gerundet.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

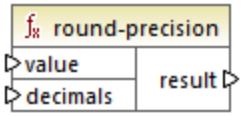
© 2016-2022 Altova GmbH

Parameter

Argument	Beschreibung
value	Obligatorischer Parameter. Liefert den Input-Wert der Funktion.

6.7.6.8 round-precision

Rundet den Input-Wert auf *N* Dezimalstellen, wobei *N* das Argument **decimals** ist.



Sprachen

Built-in, C++, C#, Java.

Parameter

Argument	Beschreibung
value	Obligatorischer Parameter. Liefert den Input-Wert der Funktion.
decimals	Obligatorischer Parameter. Definiert die Anzahl der Stellen, auf die gerundet werden soll.

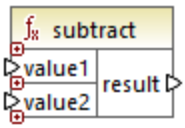
Beispiel

Die Rundung des Werts **2,777777** auf 2 Dezimalstellen ergibt **2,78**. Die Rundung des Werts **0,1234** auf 3 Dezimalstellen ergibt **0,123**.

Siehe auch Beispiel: Auslesen von Daten aus Protol Buffers 849.

6.7.6.9 subtract

Subtrahiert **value2** von **value1** und gibt das Ergebnis als Dezimalwert zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten Operanden an.
value2	Obligatorischer Parameter. Gibt den zweiten Operanden an.

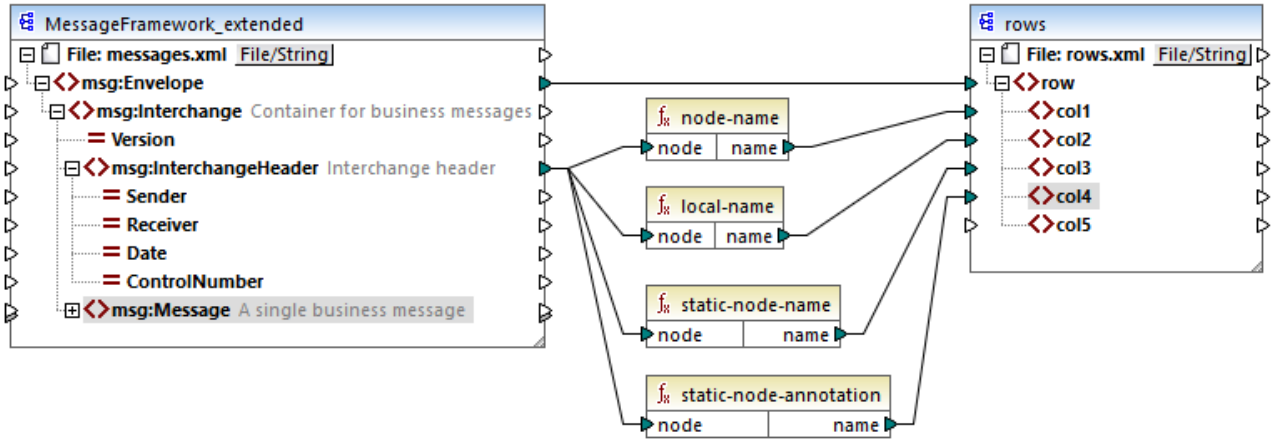
6.7.7 core | node functions (Node-Funktionen)

Mit Hilfe der Funktionen aus den **core | node-Funktionen** können Sie Informationen über Nodes in einer Mapping-Komponente (wie z.B. den Node-Namen oder die Annotation) aufrufen oder nillable Elemente verarbeiten, siehe auch Nullwerte / Nillable Werte 132.

Es gibt auch eine alternative Methode, um Node-Namen aufzurufen, für die gar keine Node-Funktionen benötigt werden, siehe Mappen von Node-Namen 1309.

Im unten gezeigten Mapping sehen Sie einige Node-Funktionen, die Informationen aus dem Node **msg:InterchangeHeader** der XML-Quelldatei abrufen. Dabei werden die folgenden Informationen extrahiert:

1. Die Funktion **`node-name`** gibt den qualifizierten Namen des Node einschließlich des Node-Präfix zurück.
2. Die Funktion **`local-name`** gibt nur den lokalen Teil zurück.
3. Die Funktion **`static-node-name`** ist ähnlich der Funktion **`node-name`**, steht aber auch in XSLT 1.0 zur Verfügung.
4. Die Funktion **`static-node-annotation`** ruft die gemäß dem XML-Schema definierte Annotation des Elements ab.



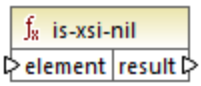
Das Mapping-Ergebnis ist das folgende (ausschließlich der XML- und Namespace-Deklaration):

© 2016-2022 Altova GmbH

```
<row>
   <col1>msg:InterchangeHeader</col1>
   <col2>InterchangeHeader</col2>
   <col3>msg:InterchangeHeader</col3>
   <col4>Interchange header</col4>
</row>
```

6.7.7.1 is-xsi-nil

Gibt **true** zurück, wenn das `xsi:nil`-Attribut des **element**-Node auf **true** gesetzt ist.



Sprachen

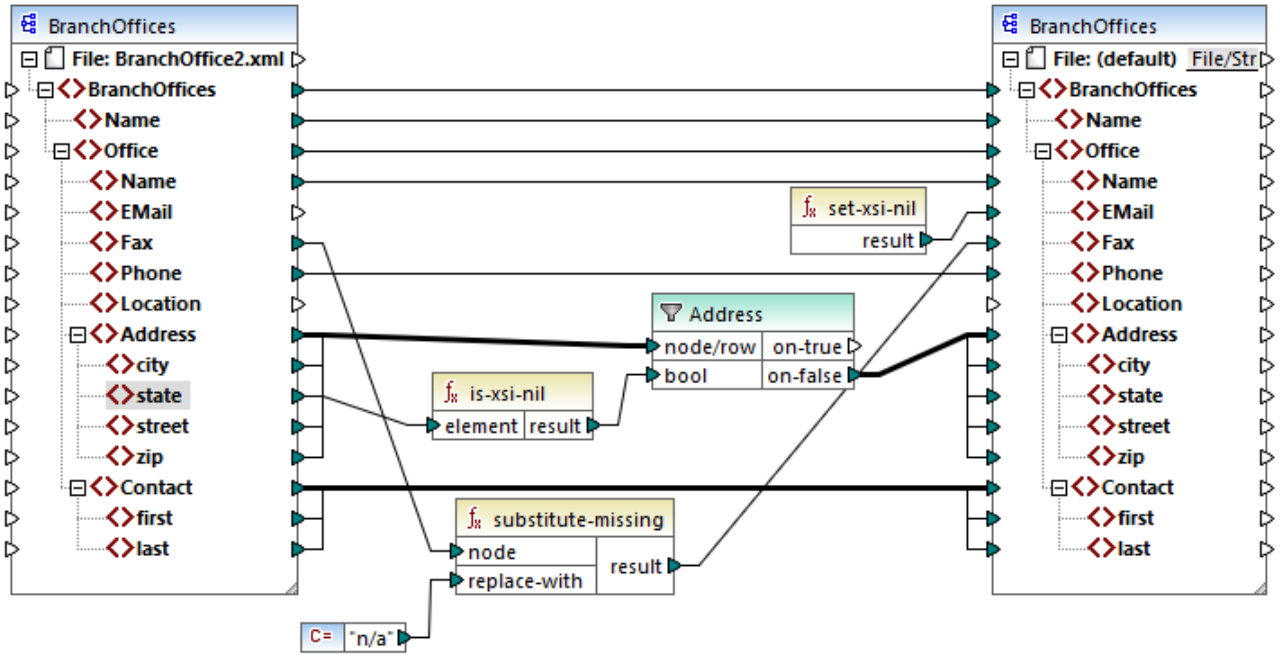
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
element	Obligatorischer Parameter. Muss mit dem zu überprüfenden Quell-Node verbunden sein.

Beispiel

Im unten gezeigten Mapping-Design werden Daten auf Basis von Bedingungen aus einer XML-Quelldatei in eine XML-Zieldatei kopiert. Außerdem wird hier die Verwendung einer Reihe von Funktionen, darunter der Funktion **is-xsi-nil** veranschaulicht. Das Mapping hat den Namen **HandlingXsiNil.mfd** und befindet sich im Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**.



Wie oben gezeigt, überprüft die Funktion **is-xsi-nil**, ob das Attribut `xsi:nil` für das Datenelement **state** der Quelldatei "true" ist. Wenn dieses Attribut "false" ist, kopiert der Filter das übergeordnete **Address**-Element in die Zielkomponente. Die XML-Quelldatei sieht folgendermaßen aus (ausschließlich der XML- und Namespace-Deklaration):

```
<BranchOffices>
   <Name>Nanonull</Name>
   <Office>
      <Name>Nanonull Research Outpost</Name>
      <EMail>sp@nanonull.com</EMail>
      <Fax xsi:nil="true"/>
      <Phone>+8817 3141 5926</Phone>
      <Address>
         <city>South Pole</city>
         <state xsi:nil="true"/>
         <street xsi:nil="true"/>
         <zip xsi:nil="true"/>
      </Address>
      <Contact>
         <first>Scott</first>
         <last>Amundsen</last>
      </Contact>
   </Office>
</BranchOffices>
```

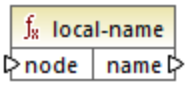
Das Ergebnis des Mappings ist, dass gar kein **Address**-Element in die Zielkomponente kopiert wird, da die Quelldatei nur ein **Address**-Element enthält und dessen Attribut `xsi:nil` für das Element **state** auf "true" gesetzt ist. Die Mapping-Ausgabe sieht daher folgendermaßen aus:

© 2016-2022 Altova GmbH

```
<BranchOffices>
   <Name>Nanonull</Name>
   <Office>
      <Name>Nanonull Research Outpost</Name>
      <EMail xsi:nil="true"/>
      <Fax>n/a</Fax>
      <Phone>+8817 3141 5926</Phone>
      <Contact>
         <first>Scott</first>
         <last>Amundsen</last>
      </Contact>
   </Office>
</BranchOffices>
```

6.7.7.2 local-name

Gibt den lokalen Namen des Node zurück. Im Gegensatz zur Funktion node-name[1071] gibt **local-name** das Präfix des Node nicht zurück. Wenn der Node kein Präfix hat, geben **local-name** und **node-name** denselben Wert zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

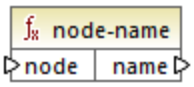
Parameter

Argument	Beschreibung
node	Obligatorischer Parameter. Verbinden Sie diesen Input mit den Node, dessen Namen Sie abrufen möchten.

6.7.7.3 node-name

Gibt den qualifizierten Namen (QName) des verbundenen Node zurück. Wenn es sich um einen XML **text()**-Node handelt, wird ein leerer QName zurückgegeben. Die Funktion funktioniert nur mit Nodes, die einen Namen haben. Wenn XSLT 2.0 die Zielsprache ist (die **fn:node-name** aufruft), wird eine leere Sequenz für Nodes, die keinen Namen haben, zurückgegeben.

Anmerkung: Der Abruf des Node-Namens wird für "Datei Input"-Nodes, Datenbanktabellen oder -felder, XBRL, Excel, JSON, oder Protocol Buffer-Felder nicht unterstützt.



Sprachen

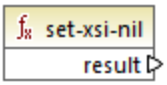
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
node	Obligatorischer Parameter. Verbinden Sie diesen Input mit den Node, dessen Namen Sie abrufen möchten.

6.7.7.4 set-xsi-nil

Setzt den Ziel-Node auf xsi:nil.



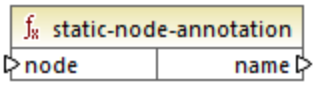
Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

6.7.7.5 static-node-annotation

Gibt den String mit der Annotation des verbundenen Node zurück. Beim Input muss es sich: (i) um eine Quellkomponente oder (ii) um eine benutzerdefinierte Funktion vom Typ inline[970] handeln, die direkt mit einerm Parameter[973] verbunden ist, der wiederum direkt mit einem Node im aufrufenden Mapping verbunden ist.

Die Verbindung muss direkt erfolgen und darf nicht über einen Filter oder eine reguläre benutzerdefinierte (Nicht-Inline gesetzte) Funktion laufen. Dies ist eine Pseudofunktion, die bei der Generierung durch den aus dem verbundenen Node abgerufenen Text ersetzt wird und daher für alle Programmiersprachen zur Verfügung steht.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

© 2016-2022 Altova GmbH

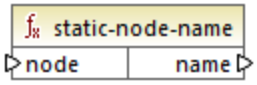
Parameter

Argument	Beschreibung
node	Obligatorischer Parameter. Verbinden Sie diesen Input mit den Node, dessen Annotation Sie abrufen möchten.

6.7.7.6 static-node-name

Gibt den String mit dem Names des damit verbundenen Node zurück. Beim Input muss es sich: (i) um eine Quellkomponente oder (ii) um eine benutzerdefinierte Funktion vom Typ inline[970] handeln, die direkt mit einerm Parameter[973] verbunden ist, der wiederum direkt mit einem Node im aufrufenden Mapping verbunden ist.

Die Verbindung muss direkt erfolgen und darf nicht über einen Filter oder eine benutzerdefinierte (nicht-Inline gesetzte) Funktion laufen. Dies ist eine Pseudofunktion, die bei der Generierung durch den aus dem verbundenen Node abgerufenen Text ersetzt wird und daher für alle Programmiersprachen zur Verfügung steht.



Sprachen

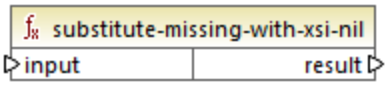
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
node	Obligatorischer Parameter. Verbinden Sie diesen Input mit den Node, dessen Namen Sie abrufen möchten.

6.7.7.7 substitute-missing-with-xsi-nil

Ersetzt bei Nodes mit Simple Content alle fehlenden (oder Null-Werte) der Quellkomponente im Ziel-Node durch das `xsi:nil` Attribut.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

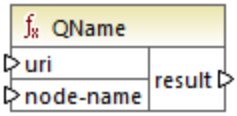
Argument	Beschreibung
input	Obligatorischer Parameter. Verbinden Sie diesen Input mit den Node, dessen Namen Sie abrufen möchten.

6.7.8 core | QName functions (QName-Funktionen)

QName--Funktionen bieten eine Methode, um die qualifizierten Namen (QName) in XML-Dokumenten zu bearbeiten.

6.7.8.1 QName

Konstruiert anhand einer Namespace URI und eines lokalen Teils einen QName. Mit Hilfe dieser Funktion können Sie einen QName in einer Zielkomponente erstellen. Die Parameter **uri** und **node-name** können mit Hilfe einer Konstantenfunktion bereitgestellt werden.



Sprachen

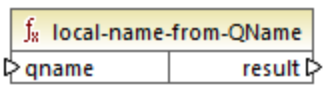
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
uri	Obligatorisch. Stellt die URI bereit.
node-name	Obligatorisch. Liefert den Namen des Node.

6.7.8.2 local-name-from-QName

Extrahiert den lokalen Namensteil aus einem Wert vom Typ `xs:QName`. Beachten Sie, dass diese Funktion im Gegensatz zur Funktion local-name[1071], die den lokalen Namen des *Node*zurückgibt, den *Inhalt* des mit dem Input **qname** verbundenen Datenelements zurückgibt.



© 2016-2022 Altova GmbH

Sprachen

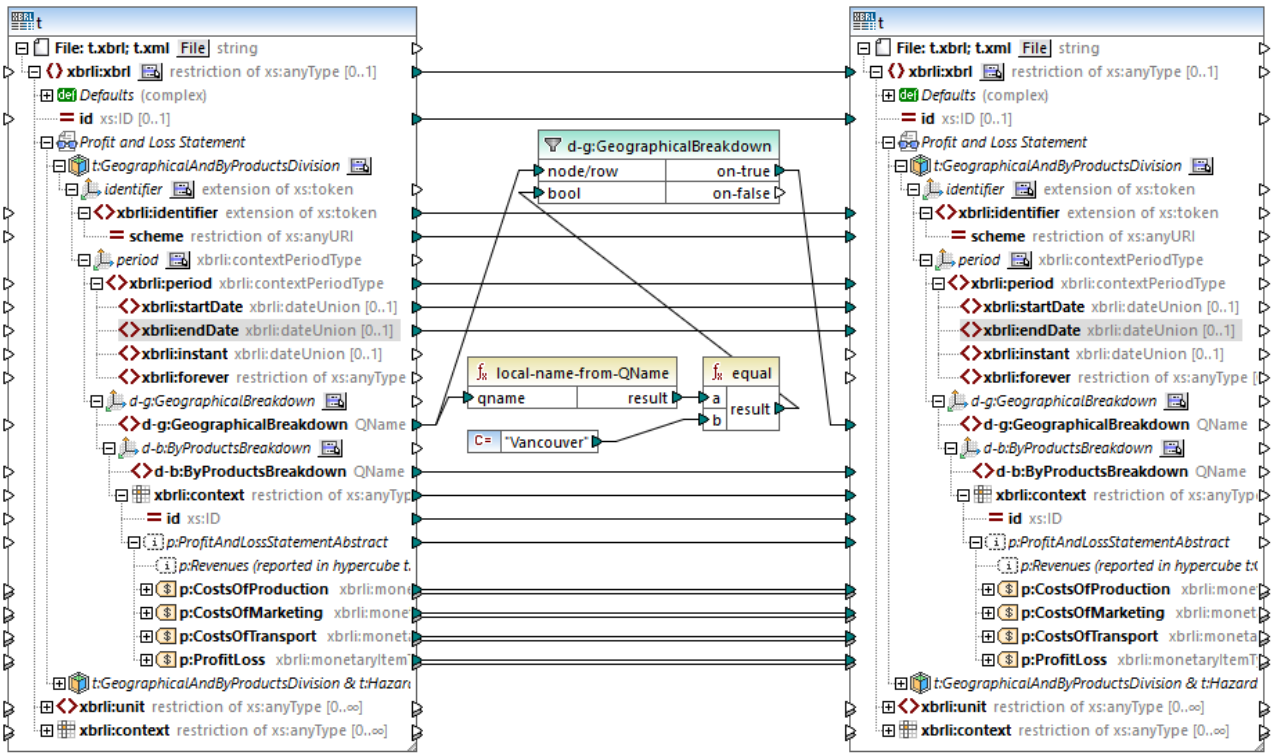
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
qname	Obligatorisch. Liefert den Input-Wert der Funktion, vom Typ `xs:QName`.

Beispiel

Diese Funktion ist nützlich, wenn Sie XBRL-Instanzdokumente mappen, die Hypercubes enthalten.



Das oben gezeigte Mapping filtert diejenigen Facts, bei denen der lokale Name des *Inhalts* des Explicit Member **d-g:Vancouver** gleich "Vancouver" ist. Beachten Sie, dass der Inhalt des Member selbst ein QName ist, siehe Codefragment unten.

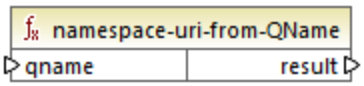
```
<xbrli:entity>
    <xbrli:identifier scheme="http://nasdaq.com/ticker">ACME2</xbrli:identifier>
    <xbrli:segment>
        <xbrldi:explicitMember dimension="d-b:ByProductsBreakdown">d-
b:Water</xbrldi:explicitMember>
        <xbrldi:explicitMember dimension="d-g:GeographicalBreakdown">d-
```

```
g:Vancouver</xbrldi:explicitMember>
    </xbrli:segment>
</xbrli:entity>
```

Nach Ausführung des Mappings, wurden alle zur Dimension **GeographicalBreakdown** gehörigen Facts gefiltert und an die Zielkomponente übergeben.

6.7.8.3 namespace-uri-from-QName

Gibt den Namespace URI-Teil des als Argument bereitgestellten QName-Werts zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
qname	Obligatorisch. Liefert den Input-Wert der Funktion.

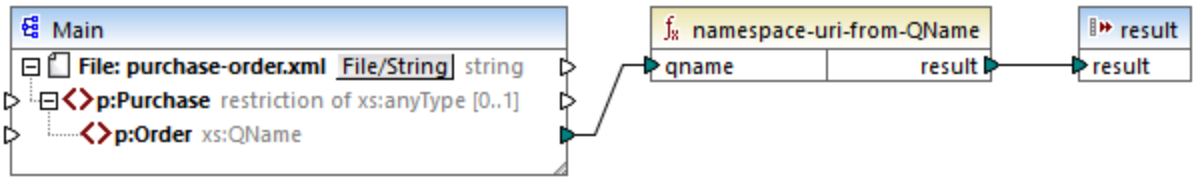
Beispiel

Die folgende XML-Datei enthält den QName-Wert **o:name**. Beachten Sie, dass das Präfix "o" auf den Namespace **`http://NamespaceTest.com/Order`** gemappt ist.

```
<?xml version="1.0" encoding="utf-8"?>
<p:Purchase xsi:schemaLocation="http://NamespaceTest.com/Purchase Main.xsd"
          xmlns:p="http://NamespaceTest.com/Purchase"
          xmlns:o="http://NamespaceTest.com/Order"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <p:Order>o:name</p:Order>
</p:Purchase>
```

Unten sehen Sie ein Mapping, in dem der QName-Wert verarbeitet wird und die Namespace URI abgerufen wird:

© 2016-2022 Altova GmbH



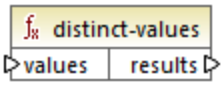
Die Ausgabe des Mappings ist `http://NamespaceTest.com/Order`.

6.7.9 core | sequence functions (Sequenzfunktionen)

Mit Hilfe von Sequenzfunktionen können Input-Sequenzen[1331] verarbeitet und ihr Inhalt gruppiert werden.

6.7.9.1 distinct-values

Verarbeitet die mit dem **values** Input verbundene Wertesequenz und gibt nur die eindeutigen Werte als Sequenz zurück. Dies dient zum Entfernen doppelt vorhandener Werte aus einer Sequenz und zum Mappen der eindeutigen Datenelemente auf die Zielkomponente.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
values	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz[1331] von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.

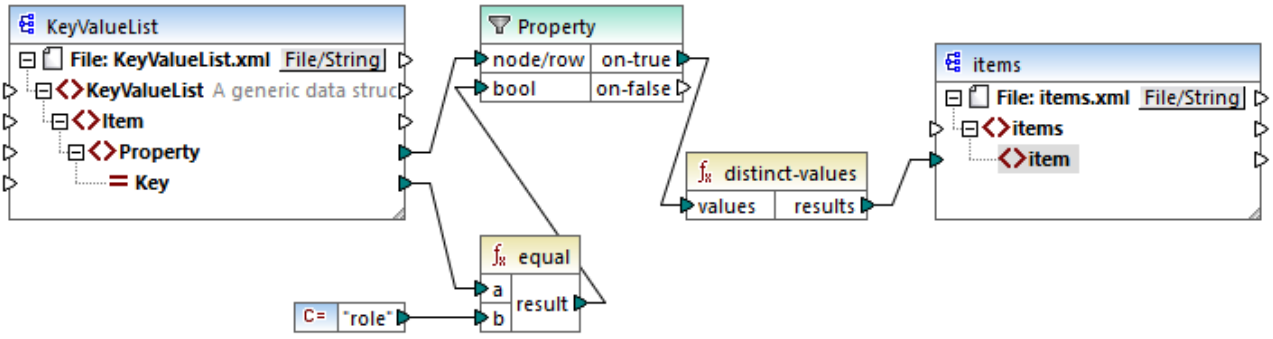
Beispiel

Die folgende XML-Datei enthält Informationen über Mitarbeiter einer Demo-Firma. Einige Mitarbeiter haben dieselbe Rolle (role), daher enthält das Attribut "role" doppelt vorhandene Werte. So haben etwa sowohl "Loby Matise" als auch "Susi Sanna" die Rolle "Support".

```
<?xml version="1.0" encoding="UTF-8"?>
<KeyValueList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="KeyValueList.xsd">
```

```
   <Item>
      <Property Key="role">Manager</Property>
      <Property Key="First">Vernon</Property>
      <Property Key="Last">Callaby</Property>
   </Item>
   <Item>
      <Property Key="role">Programmer</Property>
      <Property Key="First">Frank</Property>
      <Property Key="Last">Further</Property>
   </Item>
   <Item>
      <Property Key="role">Support</Property>
      <Property Key="First">Loby</Property>
      <Property Key="Last">Matise</Property>
   </Item>
   <Item>
      <Property Key="role">Support</Property>
      <Property Key="First">Susi</Property>
      <Property Key="Last">Sanna</Property>
   </Item>
</KeyValueList>
```

Angenommen, Sie möchten eine Liste aller *eindeutigen* Rollennamen, die in dieser XML-Datei vorkommen, extrahieren. Dies lässt sich mit einem Mapping wie dem unten gezeigten, ermitteln:



Im oben gezeigten Mapping geschieht Folgendes:

- Jedes **Property**-Element aus der XML-Quelldatei wird durch einen Filter verarbeitet.
- Mit Hilfe der Verbindung mit dem Input **bool** des Filters wird sichergestellt, dass an die Zielkomponente nur **Property**-Elemente, deren **Key**-Attribut gleich "role" ist, übergeben werden. Der String "role" wird von einer Konstanten bereitgestellt. Beachten Sie, dass die Ausgabe des Filters an diesem Punkt immer noch Duplikate enthält (da es zwei "Support"-Eigenschaften gibt, die den Filterbedingungen entsprechen).
- Die vom Filter erzeuge Sequenz wird von der **`distinct-values`**-Funktion verarbeitet, die etwaige doppelt vorhandenen Werte ausschließt.

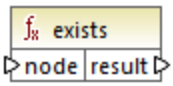
Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

© 2016-2022 Altova GmbH

```
<items>
   <item>Manager</item>
   <item>Programmer</item>
   <item>Support</item>
</items>
```

6.7.9.2 exists

Gibt **true** zurück, wenn der verbundene Node vorhanden ist und **false**, wenn dies nicht der Fall ist. Da ein Boolescher Wert zurückgegeben wird, wird diese Funktion normalerweise mit Filtern[914] verwendet, um nur Datensätze herauszufiltern, die ein Child-Element oder Attribut haben (oder eventuell nicht haben).



Sprachen

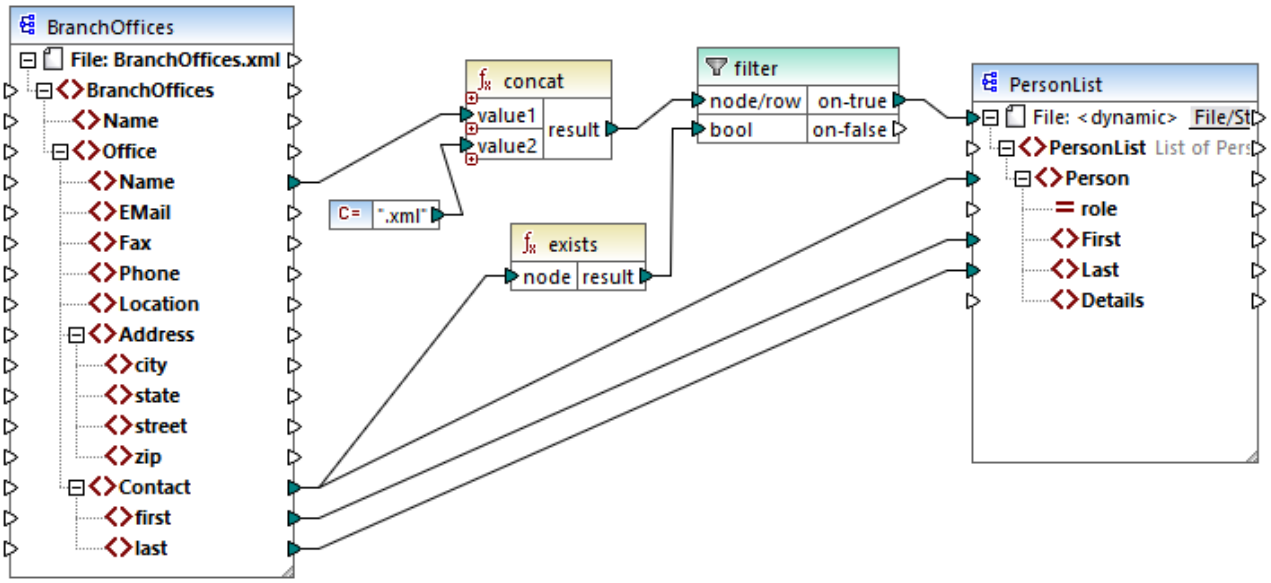
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
node	Der Node, dessen Vorhandensein überprüft werden soll.

Beispiele

Im folgenden Mapping wird gezeigt, wie Sie Daten mit Hilfe einer **exists**-Funktion filtern. Dieses Mapping hat den Namen **PersonListsForAllBranchOffices.mfd** und befindet sich im Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**.



PersonListsForAllBranchOffices.mfd

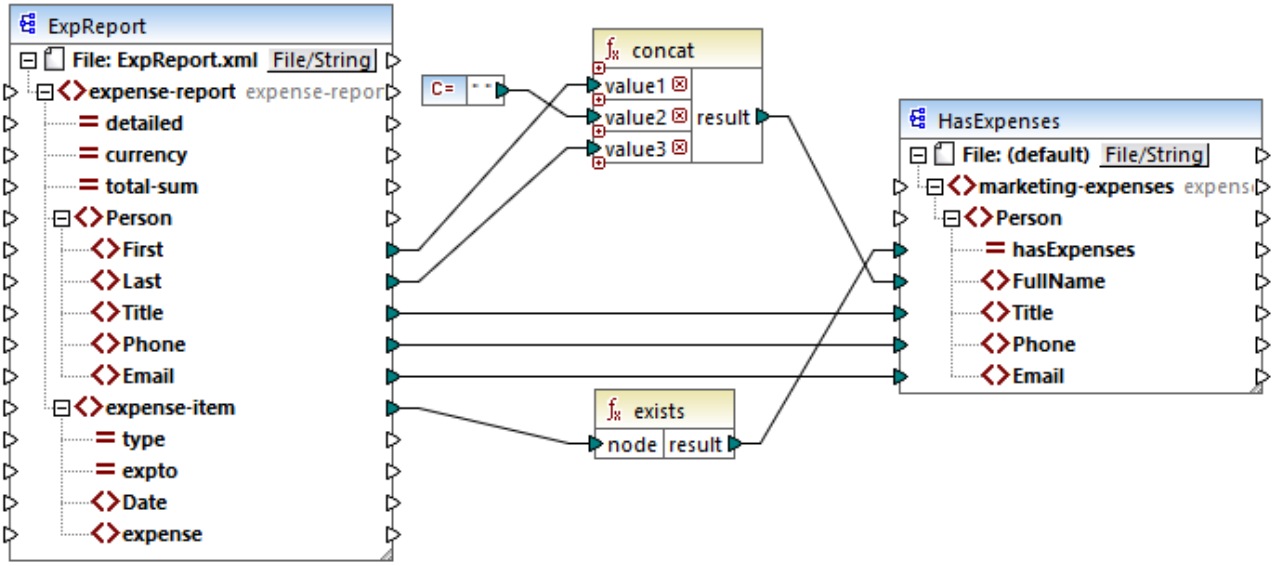
In der Quelldatei **BranchOffices.xml** gibt es drei **Office**-Elemente. Eines der Büros (office-Elemente) hat kein **Contact** Child-Element. Dieses Mapping hat mehrere Aufgaben:

a) Extraktion einer Liste der in jedem einzelnen Büro vorhandenen Kontakte.
b) Erstellung einer separaten XML-Datei für jedes Büro, wobei diese Datei denselben Namen wie das Büro hat.
c) Wenn das Büro keine Kontakte enthält, soll keine XML-Datei generiert werden.

Zu diesem Zweck wurde ein Filter zum Mapping hinzugefügt. Der Filter übergibt nur diejenigen **Office**-Datenelemente an die Zielkomponente, die mindestens ein **Contact**-Datenelement enthalten. Diese Boolesche Bedingung wird durch die `exists`-Funktion bereitgestellt. Wenn das Ergebnis der Funktion "true" ist, wird der Name des Büros (office) mit dem String `.xml` verkettet, um den Zieldateinamen zu erzeugen. Nähere Informationen zum Generieren von Dateinamen anhand des Mappings finden Sie unter Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 1371.

Ein weiteres Beispiel ist das folgende Mapping:
<Dokumente>\Altova\MapForce2023\MapForceExamples\HasMarketingExpenses.mfd. Wenn hier in der XML-Quelldatei ein Datenelement **expense-item** existiert, wird das Attribut **hasExpenses** in der XML-Zieldatei auf **true** gesetzt.

 © 2016-2022 Altova GmbH

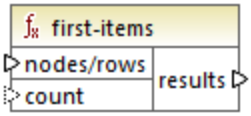


HasMarketingExpenses.mfd

Siehe auch Beispiel: Ausnahme, wenn Node nicht vorhanden ist 940.

6.7.9.3 first-items

Gibt die ersten *N* Datenelemente der Input-Sequenz zurück, wobei *N* die vom Parameter **count** bereitgestellte Anzahl ist.



Sprachen

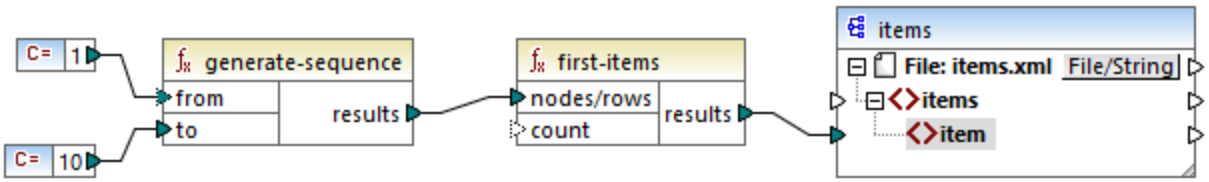
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 1331 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
count	Optionaler Parameter. Definiert, wie viele Datenelemente aus der Input-Sequenz abgerufen werden sollen. Der Standardwert ist **1**.

Beispiel

Im folgenden Modell-Mapping wird eine Sequenz von 10 Werten generiert. Die Sequenz wird von der Funktion **first-items** verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.



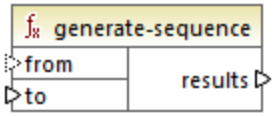
Da das Argument **count** keinen Wert hat, wird der Standardwert **1** angewendet. Infolgedessen wird nur der erste Wert aus der Sequenz in der Mapping-Ausgabe generiert:

```
<items>
   <item>1</item>
</items>
```

Ein realistisches Beispiel dazu finden Sie in dem in Bereitstellen von Parametern für das Mapping[867] beschriebenen Mapping **FindHighestTemperatures.mfd**.

6.7.9.4 generate-sequence

Erstellt eine Ganzzahlsequenz unter Verwendung der Parameter "from" und "to", um den Bereich einzugrenzen.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

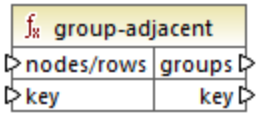
Parameter

Name	Beschreibung
from	Optionaler Parameter. Definiert die Ganzzahl, mit der die Sequenz beginnen soll (untere Grenze). Der Standardwert ist **1**.
to	Obligatorischer Parameter. Definiert die Ganzzahl, mit der die Sequenz enden soll (obere Grenze).

© 2016-2022 Altova GmbH

6.7.9.5 group-adjacent

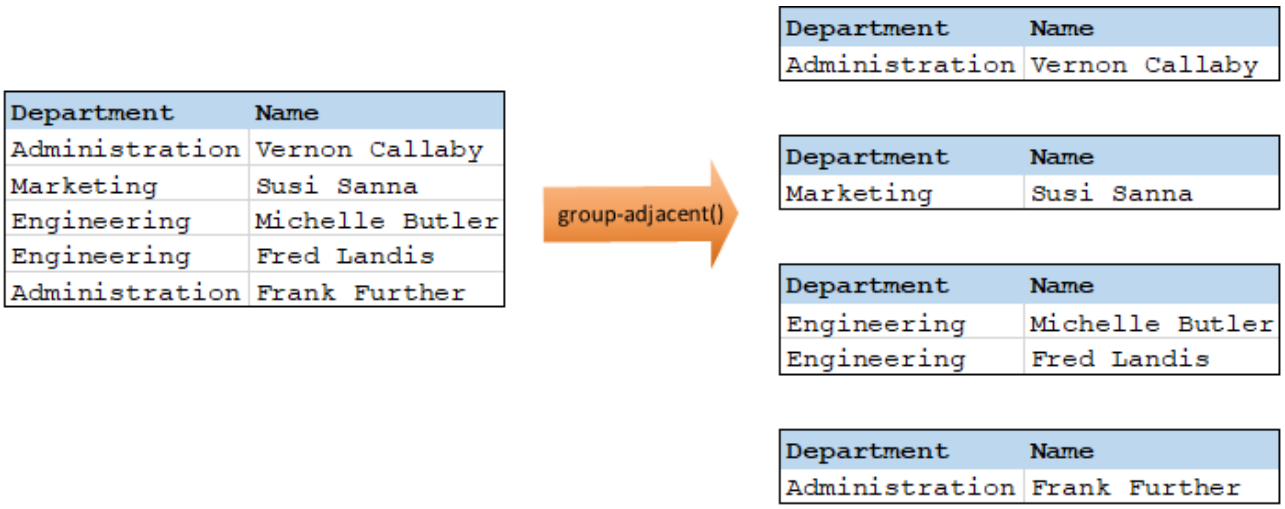
Die Funktion **`group-adjacent`** gruppiert die mit dem Input **nodes/rows** verbundenen Datenelemente nach dem mit dem **key**-Input verbundenen Schlüssel. Beachten Sie, dass Datenelemente mit demselben Schlüssel in separate Gruppen platziert werden, wenn sie nicht benachbart sind. Wenn mehrere aufeinander folgende (benachbarte) Datenelemente denselben Schlüssel haben, werden sie in dieselbe Gruppe platziert.



So ist etwa in der unten gezeigten abstrakten Transformation der Gruppierungsschlüssel "Department". Auf der linken Seite des Diagramms sehen Sie die Input-Daten, während rechts die Ausgabedaten nach der Gruppierung angezeigt werden. Bei Ausführung der Transformation geschieht Folgendes:

- Anfangs wird anhand des ersten Schlüssels "Administration" eine neue Gruppe erstellt.
- Der nächste Schlüssel ist ein anderer, daher wird eine zweite Gruppe "Marketing" erstellt.
- Der dritte Schlüssel ist ebenfalls ein anderer, daher wird eine weitere Gruppe namens "Engineering" erstellt.
- Der vierte Schlüssel ist derselbe wie der dritte, daher wird dieser Datensatz in die bereits vorhandene Gruppe platziert.
- Der fünfte Schlüssel schließlich ist anders als der vierte, weshalb die letzte Gruppe erstellt wird.

Wie unten gezeigt, landen "Michelle Butler" und "Fred Landis" in derselben Gruppe, weil sie denselben Schlüssel aufweisen und nebeneinander liegen. "Vernon Callaby" und "Frank Further" hingegen befinden sich in separaten Gruppen, weil sie nicht benachbart sind, obwohl sie denselben Schlüssel aufweisen.



Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

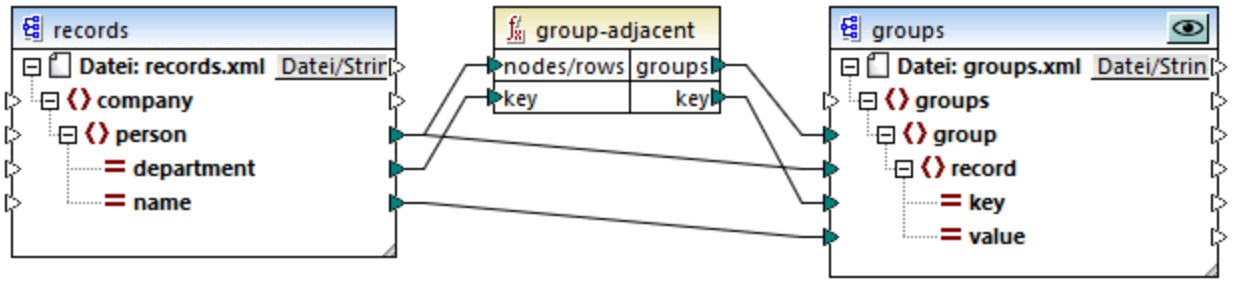
Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz[1331] von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
key	Der Schlüssel, nach dem Datenelemente gruppiert werden sollen.

Beispiel

Angenommen, bei Ihren Quelldaten handelt es sich um eine XML-Datei folgenden Inhalts (Beachten Sie, dass der Namespace und die XML-Deklarationen im Codefragment unten aus Gründen der Einfachheit entfernt wurden).

```
<company>
   <person department="Administration" name="Vernon Callaby"/>
   <person department="Marketing" name="Susi Sanna"/>
   <person department="Engineering" name="Michelle Butler"/>
   <person department="Engineering" name="Fred Landis"/>
   <person department="Administration" name="Frank Further"/>
</company>
```

Die Personendatensätze sollen nach Abteilung (department) gruppiert werden, vorausgesetzt die Datensätze sind benachbart. Zu diesem Zweck wird im folgenden Mapping die Funktion **`group-adjacent`** aufgerufen und **department** wird als **key** (Schlüssel) bereitgestellt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

```
<groups>
   <group>
      <record key="Administration" value="Vernon Callaby"/>
   </group>
   <group>
      <record key="Marketing" value="Susi Sanna"/>
   </group>
```

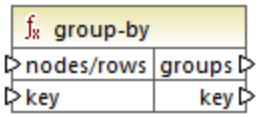

© 2016-2022 Altova GmbH

```
   <group>
      <record key="Engineering" value="Michelle Butler"/>
      <record key="Engineering" value="Fred Landis"/>
   </group>
   <group>
      <record key="Administration" value="Frank Further"/>
   </group>
</groups>
```

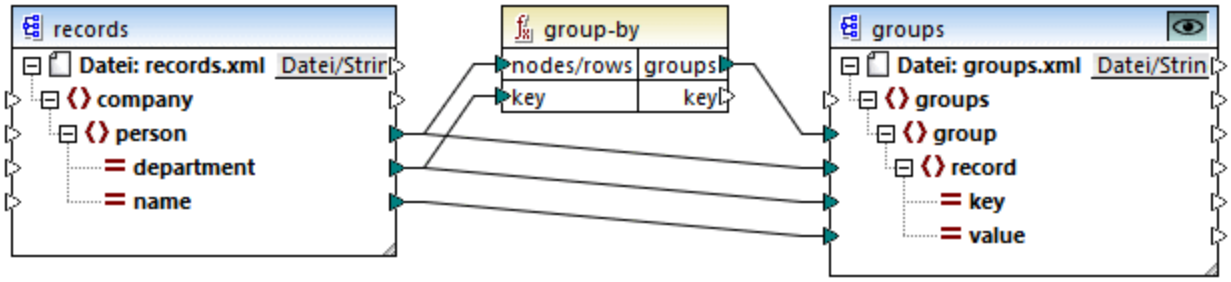
Sie finden dieses Beispiel sowie einige weitere Gruppierungsbeispiele in der folgenden Mapping-Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\GroupingFunctions.mfd**. Klicken Sie auf die Schaltfläche **Vorschau** für die jeweilige Funktion, bevor Sie auf das Register **Ausgabe** klicken.

6.7.9.6 group-by

Mit der Funktion **group-by** werden anhand eines von Ihnen definierten Gruppierungsschlüssels Datensatzstrukturen erstellt.



So ist etwa in der unten gezeigten abstrakten Transformation der Gruppierungsschlüssel "Department". Da es insgesamt drei eindeutige Abteilungen (Departments) gibt, würden bei Anwendung der Funktion group-by drei Gruppen erstellt:



Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz[1331] von null oder mehr Werten liefert. So kann damit etwa ein

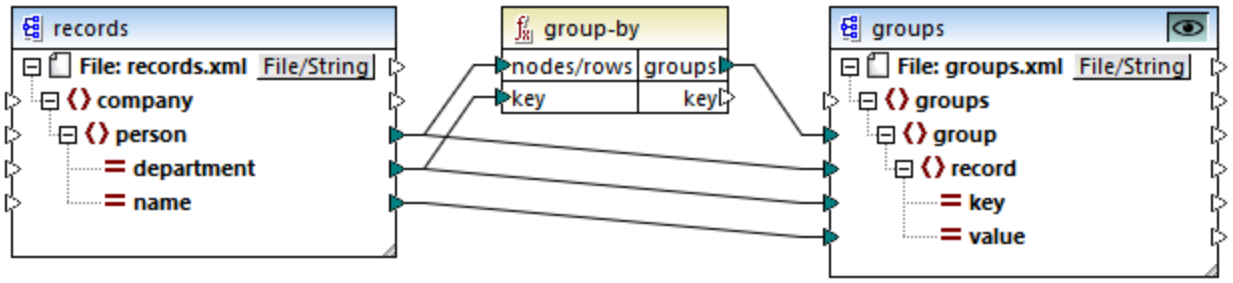
Name	Beschreibung
	Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
key	Der Schlüssel, nach dem Datenelemente gruppiert werden sollen.

Beispiel

Angenommen, bei Ihren Quelldaten handelt es sich um eine XML-Datei folgenden Inhalts (Beachten Sie, dass der Namespace und die XML-Deklarationen im Codefragment unten aus Gründen der Einfachheit entfernt wurden).

```
<company>
   <person department="Administration" name="Vernon Callaby"/>
   <person department="Marketing" name="Susi Sanna"/>
   <person department="Engineering" name="Michelle Butler"/>
   <person department="Engineering" name="Fred Landis"/>
   <person department="Administration" name="Frank Further"/>
</company>
```

Die Personendatensätze sollen nach Abteilung (department) gruppiert werden. Zu diesem Zweck wird im folgenden Mapping die Funktion **group-by** aufgerufen und **department** wird als key (Schlüssel) bereitgestellt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

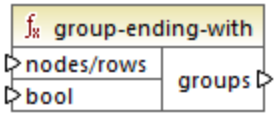
```
<groups>
   <group>
      <record key="Administration" value="Vernon Callaby"/>
      <record key="Administration" value="Frank Further"/>
   </group>
   <group>
      <record key="Marketing" value="Susi Sanna"/>
   </group>
   <group>
      <record key="Engineering" value="Michelle Butler"/>
      <record key="Engineering" value="Fred Landis"/>
   </group>
</groups>
```


© 2016-2022 Altova GmbH

Sie finden dieses Beispiel sowie einige weitere Gruppierungsbeispiele in der folgenden Mapping-Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\GroupingFunctions.mfd**. Klicken Sie auf die Schaltfläche **Vorschau** für die jeweilige Funktion, bevor Sie auf das Register **Ausgabe** klicken.

6.7.9.7 group-ending-with

Die Funktion **`group-ending-with`** erhält eine Boolsche Bedingung als Argument. Wenn die Boolsche Bedingung "true" ergibt, wird bis inklusive dem Datensatz, auf den die Bedingung zutrifft, eine neue Gruppe erstellt.



Im Beispiel unten ist die Bedingung, dass "Key" (Schlüssel) gleich "trailing" sein soll. Diese Bedingung trifft auf den dritten und fünften Datensatz zu, daher werden als Ergebnis zwei Gruppen erstellt:

Key	Value
line	A
line	B
trailing	Total 1
line	C
trailing	Total 2

group-ending-with()

Key	Value
line	A
line	B
trailing	Total 1

Key	Value
line	C
trailing	Total 2

Anmerkung: Wenn nach dem letzten Datensatz, auf den die Bedingung zutrifft, weitere Datensätze vorhanden sind, wird eine zusätzliche Gruppe erstellt. Wenn z.B. nach dem letzten "trailing"-Datensatz z.B. weitere "line"-Datensätze vorhanden wären, würden diese in eine neue Gruppe platziert.

Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

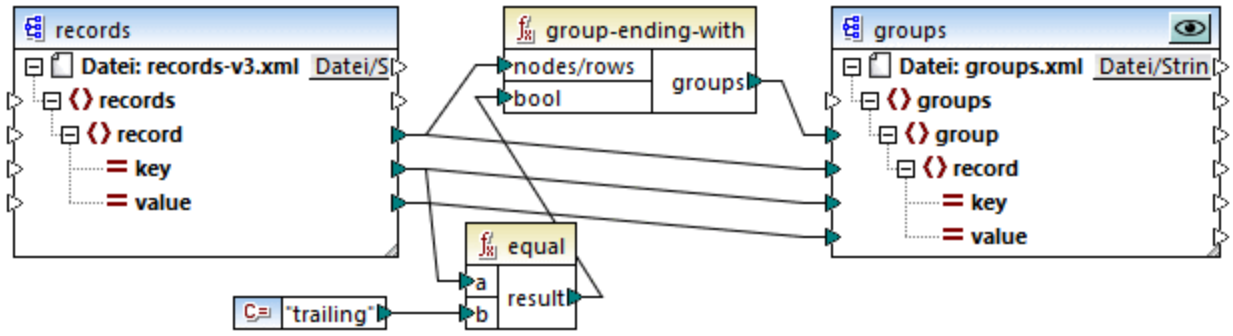
Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 1331 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
bool	Liefert die Boolsche Bedingung, durch die bei **true** eine neue Gruppe begonnen werden soll.

Beispiel

Angenommen, bei Ihren Quelldaten handelt es sich um eine XML-Datei folgenden Inhalts (Beachten Sie, dass der Namespace und die XML-Deklarationen im Codefragment unten aus Gründen der Einfachheit entfernt wurden).

```
<records>
   <record key="line" value="A"/>
   <record key="line" value="B"/>
   <record key="trailing" value="Total 1"/>
   <record key="line" value="C"/>
   <record key="trailing" value="Total 2"/>
</records>
```

Die Aufgabe ist es, für jeden nachfolgenden ("trailing") Datensatz Gruppen zu erstellen. Außerdem muss jede Gruppe auch alle "line" (Zeilen)-Datensätze enthalten, die vor dem nachfolgenden Datensatz stehen. Zu diesem Zweck wird im folgenden Mapping die Funktion **group-ending-with** aufgerufen. Immer, wenn der **key**-Name im nachstehenden Mapping gleich "trailing" ist, wird das an **bool** gelieferte Argument **true** und es wird eine neue Gruppe erstellt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

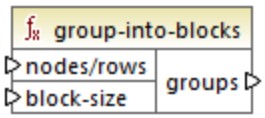
```
<groups>
   <group>
      <record key="line" value="A"/>
      <record key="line" value="B"/>
      <record key="trailing" value="Total 1"/>
   </group>
   <group>
      <record key="line" value="C"/>
      <record key="trailing" value="Total 2"/>
   </group>
</groups>
```

Sie finden dieses Beispiel sowie einige weitere Gruppierungsbeispiele in der folgenden Mapping-Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\GroupingFunctions.mfd**. Klicken Sie auf die Schaltfläche **Vorschau** für die jeweilige Funktion, bevor Sie auf das Register **Ausgabe** klicken.

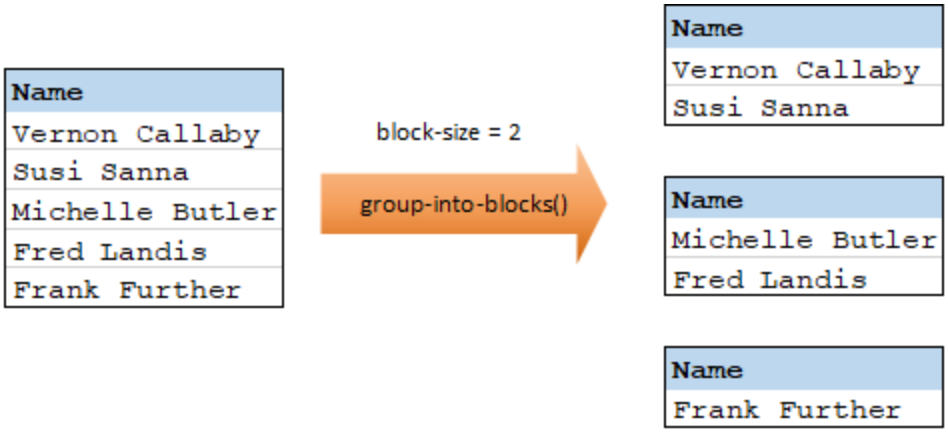
© 2016-2022 Altova GmbH

6.7.9.8 group-into-blocks

Mit der Funktion **`group-into-blocks`** werden gleiche Gruppen erstellt, die genau N Elemente enthalten, wobei N der an das Argument `block-size` (Blockgröße) gelieferte Wert ist. Beachten Sie, dass die letzte Gruppe, je nach Anzahl der Elemente in der Quellkomponente, N oder weniger Elemente enthalten kann.



Im Beispiel unten ist `block-size` gleich 2. Da es insgesamt fünf Elemente gibt, enthält jede Gruppe mit Ausnahme der letzten genau zwei Elemente.



Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

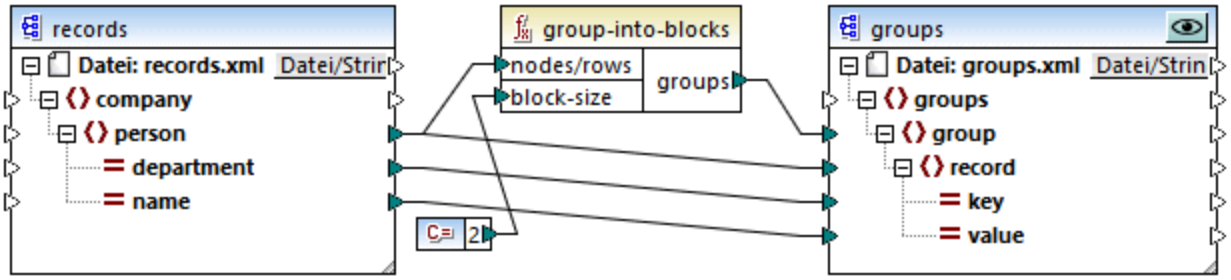
Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz[1331] von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
block-size	Definiert die Größe der einzelnen Gruppen.

Beispiel

Angenommen, bei Ihren Quelldaten handelt es sich um eine XML-Datei folgenden Inhalts (Beachten Sie, dass der Namespace und die XML-Deklarationen im Codefragment unten aus Gründen der Einfachheit entfernt wurden).

```
<company>
   <person department="Administration" name="Vernon Callaby"/>
   <person department="Marketing" name="Susi Sanna"/>
   <person department="Engineering" name="Michelle Butler"/>
   <person department="Engineering" name="Fred Landis"/>
   <person department="Administration" name="Frank Further"/>
</company>
```

Die Personendatensätze sollen in Blöcke zu jeweils zwei Datenelementen gruppiert werden. Zu diesem Zweck wird im folgenden Mapping die Funktion **group-into-blocks** aufgerufen und es wird der Ganzzahlwert "2" als **block-size** bereitgestellt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

```
<groups>
   <group>
      <record key="Administration" value="Vernon Callaby"/>
      <record key="Marketing" value="Susi Sanna"/>
   </group>
   <group>
      <record key="Engineering" value="Michelle Butler"/>
      <record key="Engineering" value="Fred Landis"/>
   </group>
   <group>
      <record key="Administration" value="Frank Further"/>
   </group>
</groups>
```

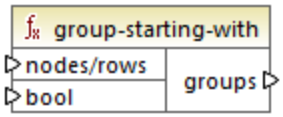
Beachten Sie, dass die letzte Gruppe nur ein Datenelement enthält, da die Summe aller Datenelemente (5) nicht gerade durch 2 dividiert werden kann.

Sie finden dieses Beispiel sowie einige weitere Gruppierungsbeispiele in der folgenden Mapping-Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\GroupingFunctions.mfd**. Klicken Sie auf die Schaltfläche **Vorschau** für die jeweilige Funktion, bevor Sie auf das Register **Ausgabe** klicken.

© 2016-2022 Altova GmbH

6.7.9.9 group-starting-with

Die Funktion **group-starting-with** erhält eine Boolsche Bedingung als Argument. Wenn die Boolsche Bedingung "true" ergibt, wird ab dem Datensatz, auf den die Bedingung zutrifft, eine neue Gruppe erstellt.



Im Beispiel unten ist die Bedingung, dass "Key" (Schlüssel) gleich "heading" sein soll. Diese Bedingung trifft auf den ersten und vierten Datensatz zu, daher werden als Ergebnis zwei Gruppen erstellt:



Anmerkung: Wenn vor dem ersten Datensatz, auf den die Bedingung zutrifft, weitere Datensätze vorhanden sind, wird eine zusätzliche Gruppe erstellt. Wenn vor dem ersten "heading"-Datensatz z.B. weitere "line"-Datensätze vorhanden waren, würden diese in eine neue Gruppe platziert.

Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

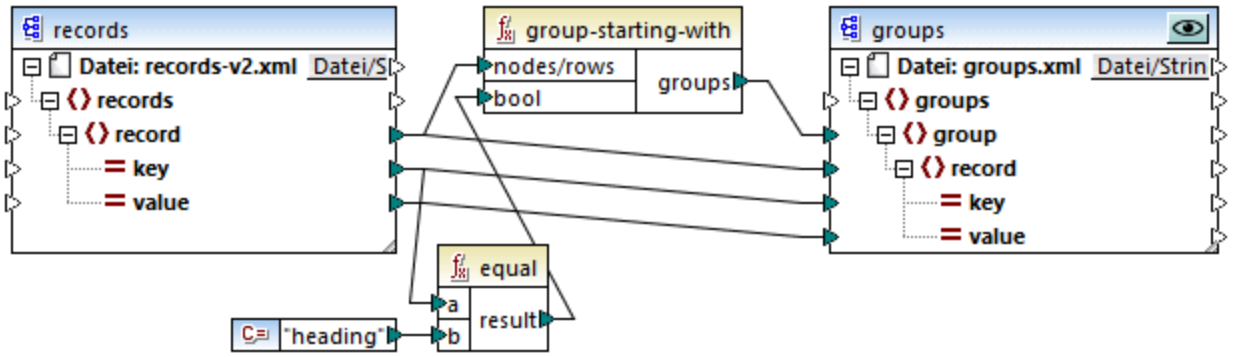
Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 1331 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
bool	Liefert die Boolsche Bedingung, durch die bei **true** eine neue Gruppe begonnen werden soll.

Beispiel

Angenommen, bei Ihren Quelldaten handelt es sich um eine XML-Datei folgenden Inhalts (Beachten Sie, dass der Namespace und die XML-Deklarationen im Codefragment unten aus Gründen der Einfachheit entfernt wurden).

```
<records>
   <record key="heading" value="Intro"/>
   <record key="line" value="A"/>
   <record key="line" value="B"/>
   <record key="heading" value="Body"/>
   <record key="line" value="C"/>
</records>
```

Die Aufgabe ist es, für jeden vorangestellten ("heading") Datensatz Gruppen zu erstellen. Außerdem muss jede Gruppe auch alle "line" (Zeilen)-Datensätze enthalten, die auf den "heading"-Datensatz folgen. Zu diesem Zweck wird im folgenden Mapping die Funktion **group-starting-with** aufgerufen. Immer, wenn der **key**-Name im nachstehenden Mapping gleich "heading" ist, wird das an **bool** gelieferte Argument **true** und es wird eine neue Gruppe erstellt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

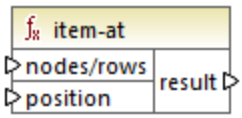
```
<groups>
   <group>
      <record key="heading" value="Intro"/>
      <record key="line" value="A"/>
      <record key="line" value="B"/>
   </group>
   <group>
      <record key="heading" value="Body"/>
      <record key="line" value="C"/>
   </group>
</groups>
```

Sie finden dieses Beispiel sowie einige weitere Gruppierungsbeispiele in der folgenden Mapping-Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\GroupingFunctions.mfd**. Klicken Sie auf die Schaltfläche **Vorschau** für die jeweilige Funktion, bevor Sie auf das Register **Ausgabe** klicken.

© 2016-2022 Altova GmbH

6.7.9.10 item-at

Gibt ein Datenelement aus einer als Argument bereitgestellten Sequenz von **nodes/rows** (Nodes/Zeilen) zurück, das sich an der durch das Argument **position** angegebenen Position befindet. Das erste Datenelement befindet sich an der Position **1**.



Sprachen

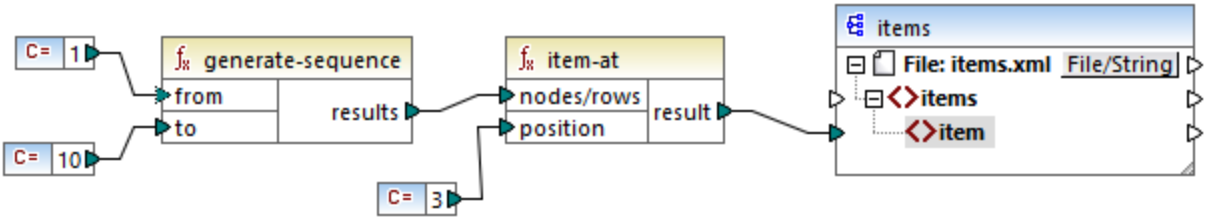
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 1331 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
position	Diese Ganzzahl gibt an, welches Datenelement aus der Sequenz von Datenelementen zurückgegeben werden soll.

Beispiel

Im folgenden Modell-Mapping wird eine Sequenz von 10 Werten generiert. Die Sequenz wird von der Funktion **item-at** verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.

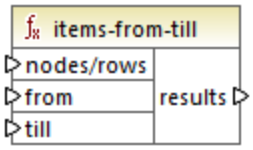


Da das **position**-Argument auf **3** gesetzt wurde, wird nur der dritte Wert aus der Sequenz an die Zielkomponente übergeben. Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

```
<items>
   <item>3</item>
</items>
```

6.7.9.11 items-from-till

Gibt eine Sequenz von **nodes/rows** (Nodes/Zeilen) unter Verwendung der Parameter "from" und "till" zur Eingrenzung des Bereichs zurück. Das erste Datenelement befindet sich an der Position **1**.



Sprachen

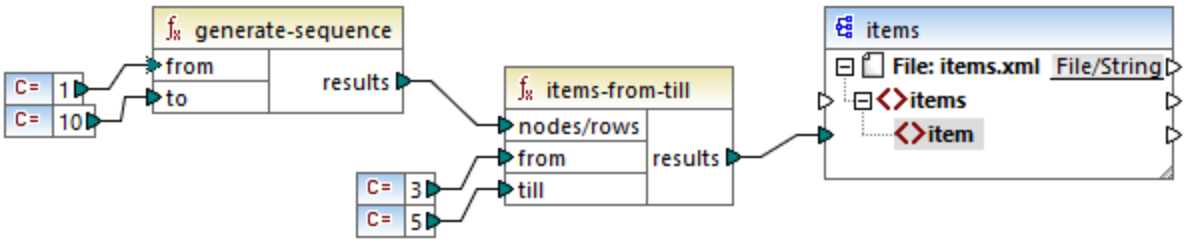
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 1331 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
from	Diese Ganzzahl gibt die Startposition, ab welcher Datenelemente abgerufen werden sollen, an.
till	Diese Ganzzahl gibt die Position, bis zu welcher Datenelemente abgerufen werden sollen, an.

Beispiel

Im folgenden Modell-Mapping wird eine Sequenz von 10 Werten generiert. Die Sequenz wird von der Funktion `items-from-till` verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.



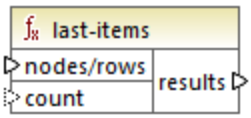
Da die Argumente **from** und **till** auf **3** bzw. **5** gesetzt wurden, wird nur die Untergruppe der Werte von **3** bis **5** an die Zielkomponente übergeben. Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

© 2016-2022 Altova GmbH

```
<items>
   <item>3</item>
   <item>4</item>
   <item>5</item>
</items>
```

6.7.9.12 last-items

Gibt die letzten *N* Datenelemente der Input-Sequenz zurück, wobei *N* die vom Parameter **count** bereitgestellte Anzahl ist. Das erste Datenelement befindet sich an der Position "1".



Sprachen

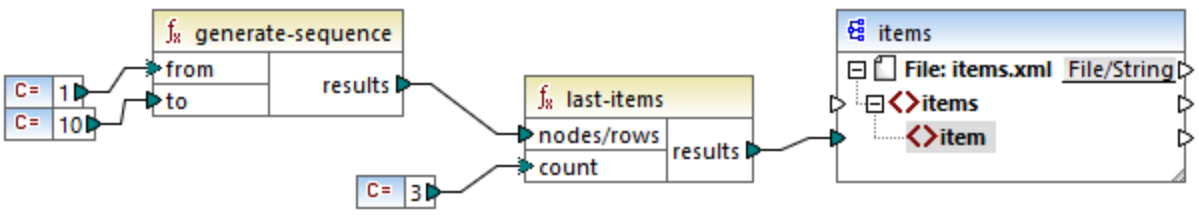
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz[1331] von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
count	Optionaler Parameter. Definiert, wie viele Datenelemente aus der Input-Sequenz abgerufen werden sollen. Der Standardwert ist **1**.

Beispiel

Im folgenden Modell-Mapping wird eine Sequenz von 10 Werten generiert. Die Sequenz wird von der Funktion `last-items` verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.

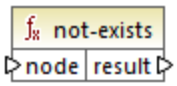


Da das count-Argument auf **3** gesetzt wurde, werden nur die drei letzten Werte aus der Sequenz an die Zielkomponente übergeben. Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

```
<items>
   <item>8</item>
   <item>9</item>
   <item>10</item>
</items>
```

6.7.9.13 not-exists

Gibt **false** zurück, wenn der verbundene Node vorhanden ist und **true**, wenn dies nicht der Fall ist. Diese Funktion ist das Gegenteil der Funktion exists[1079] und wird ansonsten auf dieselbe Art verwendet.



Sprachen

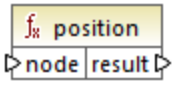
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
node	Der Node, dessen Fehlen überprüft werden soll.

6.7.9.14 position

Gibt die Position eines Datenelements innerhalb der gerade verarbeiteten Sequenz von Datenelementen zurück. Damit können Datenelemente z.B. automatisch sequenziell nummeriert werden.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

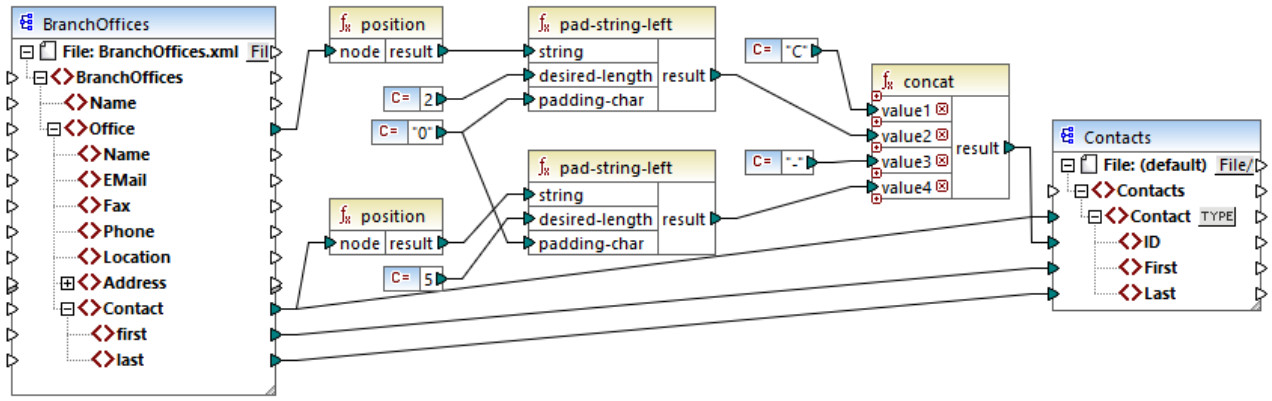
Name	Beschreibung
node	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz[1331] von null oder mehr Werten liefert. So kann damit etwa ein

© 2016-2022 Altova GmbH

Name	Beschreibung
	Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.

Beispiel

Im folgenden Mapping wird gezeigt, wie Sie mit Hilfe der **position**-Funktion in den vom Mapping generierten Daten eindeutige Identifikationswerte generieren. Die Mapping-Design-Datei zu diesem Beispiel finden Sie unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2023\MapForceExamples\ContactsFromBranchOffices.mfd.



ContactsFromBranchOffices.mfd

Die XML-Quelldatei im obigen Mapping enthält drei Niederlassungen (branch offices). Eine Niederlassung kann beliebig viele **Contact** Child-Elemente enthalten. Ziel des Mappings ist folgendes:

- Extrahierung aller **Contact**-Datenelemente aus der XML-Quelldatei und Schreiben der Daten in eine XML-Zieldatei.
- Jedem Kontakt muss eine eindeutige Idenifikationsnummer zugewiesen werden (das Datenelement **ID** in der XML-Zielkomponente).
- Die ID jedes Kontakts muss die Form **CXX-YYYYY** haben, wobei X die Nummer der Niederlassung (Office) und Y die Nummer des Kontakts (Contact) ist. Wenn die Niederlassungsnummer weniger als zwei Zeichen hat, müssen ihr links Nullen vorangestellt werden. Wenn die Kontaktnummer weniger als fünf Zeichen hat, müssen ihr links ebenfalls Nullen vorangestellt werden. Eine gültige Identifikationsnummer des ersten Kontakts aus der ersten Niederlassung wäre folglich **C01-00001**.

Dieses Ziel wurde mit Hilfe einer Reihe von MapForce-Funktionen, darunter der Funktion **position**, erreicht. Die obere **position**-Funktion ruft die Position der einzelnen office-Elemente ab. Die untere ruft die Position der einzelnen Kontakte im Kontext der einzelnen office-Elemente ab.

Bei Verwendung der **position**-Funktion muss der aktuelle Mapping-Kontext 1333 berücksichtigt werden. Das heißt, bei Ausführung des Mappings wird der Anfangs-Mapping-Kontext anhand des Root-Elements der Zielkomponente zu dem damit (auch indirekt über Funktionen) verbundenen Quelldatenelement ermittelt. In diesem Beispiel verarbeitet die obere **position**-Funktion die *Sequenz aller Niederlassungen (office)* und generiert anfangs für die erste Niederlassung in der Sequenz den Wert 1. Die untere **position**-Funktion generiert eine sequenzielle Nummerierung für die Position des Kontakts (Contact) *im Kontext dieses office-Elements* (1, 2, 3, usw.). Beachten Sie, dass diese "innere" Sequenz zurückgesetzt wird (und daher wieder mit

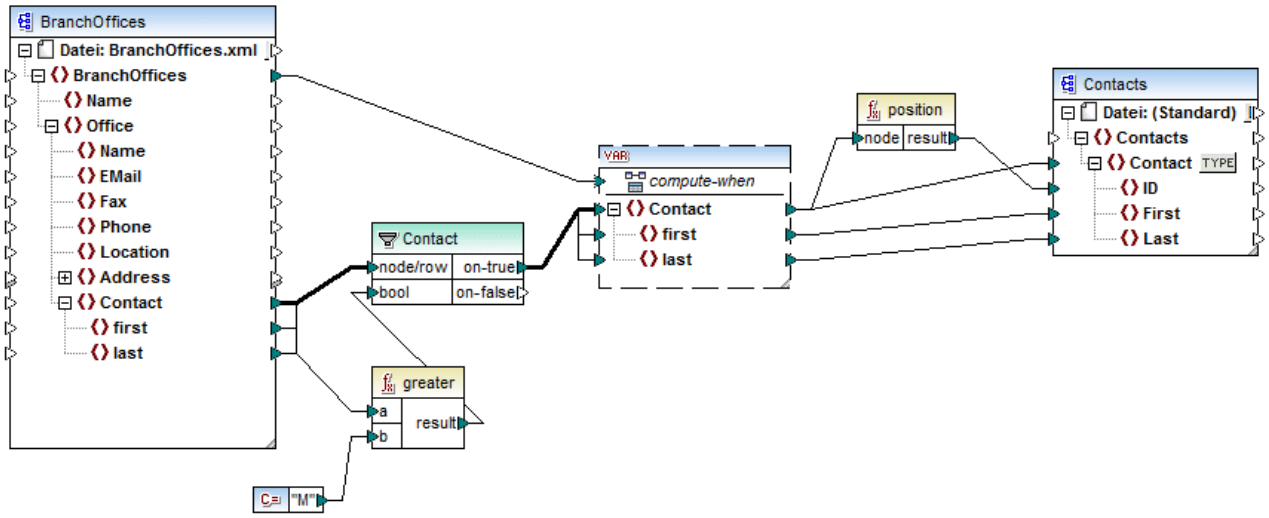
1 beginnt), wenn das nächste office-Element verarbeitet wird. Beide **pad-string-left**-Funktionen wenden gemäß den zuvor angeführten Anforderungen Füllzeichen auf die generierten Nummern an. Die concat-Funktion wird (aufgrund der übergeordneten Verbindung vom **Contact**-Element in der Quellkomponente zum **Contact**-Element in der Zielkomponente) im *Kontext der einzelnen Kontakte* angewendet. Sie verbindet alle berechneten Werte und gibt die eindeutige Identifikationsnummer der einzelnen Kontakte zurück.

Unten sehen Sie das Ergebnis, das vom oben gezeigten Mapping generiert wird (beachten Sie, dass einige der Datensätze aus Gründen der Übersichtlichkeit entfernt wurden):

```
<Contacts>
   <Contact>
      <ID>C01-00001</ID>
      <First>Vernon</First>
      <Last>Callaby</Last>
   </Contact>
   <Contact>
      <ID>C01-00002</ID>
      <First>Frank</First>
      <Last>Further</Last>
   </Contact>
   <!-- ... -->
   <Contact>
      <ID>C02-00001</ID>
      <First>Steve</First>
      <Last>Meier</Last>
   </Contact>
   <Contact>
      <ID>C02-00002</ID>
      <First>Theo</First>
      <Last>Bone</Last>
   </Contact>
   <!-- ... -->
</Contacts>
```

In einigen Fällen müssen Sie eventuell die Position von Datenelementen nach Anwendung eines Filters[914] ermitteln. Beachten Sie, dass es sich bei der Filterkomponente nicht um eine Sequenz-Funktion handelt. Sie kann nicht *direkt* zusammen mit der **position**-Funktion verwendet werden, um die Position gefilterter Datenelemente zu finden. Dies kann indirekt durch Hinzufügen einer Variablen[882]-Komponente zum Mapping ermittelt werden. Das unten gezeigte Mapping ist eine vereinfachte Version des vorherigen Mappings. Die verwendete Mapping-Design-Datei finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\PositionInFilteredSequence.mfd**.

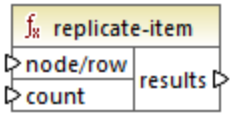
© 2016-2022 Altova GmbH



Das Ergebnis von Variablen in MapForce sind immer Sequenzen. Daher iteriert die **position**-Funktion im obigen Mapping durch die durch die Variable erstellte Sequenz und gibt die Position in den einzelnen Datenelementen in dieser Sequenz zurück. Dieses Mapping wird unter Beispiel: Filtern und Nummerieren von Nodes [892] ausführlicher erläutert.

6.7.9.15 replicate-item

Repliziert alle Datenelement in der Input-Sequenz so oft, wie im Argument **count** definiert. Wenn Sie ein einziges Datenelement mit dem **node/row** Input verbinden, gibt die Funktion *N* Datenelemente zurück, wobei *N* der Wert des Arguments **count** ist. Wenn Sie eine Sequenz von Datenelementen mit dem **node/row** Input verbinden, repliziert die Funktion jedes einzelne Datenelement in der Sequenz **count** Mal, wobei die Datenelemente der Reihe nach verarbeitet werden. Wenn z.B. count **2** ist, so erzeugt die Sequenz `1,2,3` das Ergebnis `1,1,2,2,3,3`. Es kann auch für jedes Datenelement in der Input-Sequenz ein anderer **count**-Wert angegeben werden, wie im Beispiel unten gezeigt.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
node/row	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz [1331] von null oder mehr Werten liefert. So kann damit etwa ein

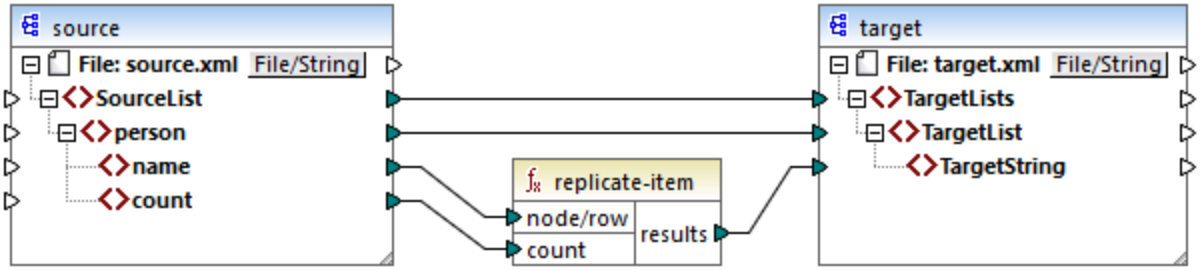
Name	Beschreibung
	Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
count	Definiert, wie oft jedes Datenelement oder jede mit **node/row** verbundene Sequenz repliziert werden soll.

Beispiel

Angenommen, Sie haben eine XML-Quelldatei mit der folgenden Struktur:

```
<SourceList>
   <person>
      <name>Michelle</name>
      <count>2</count>
   </person>
   <person>
      <name>Ted</name>
      <count>4</count>
   </person>
   <person>
      <name>Ann</name>
      <count>3</count>
   </person>
</SourceList>
```

Mit Hilfe der Funktion **replicate-item** können Sie jeden Personennamen unterschiedlich oft in der Zielkomponente replizieren. Verbinden Sie dazu den Node **count** der einzelnen person-Elemente mit dem **count**-Input der Funktion **replicate-item**:



Das Ergebnis ist das folgende:

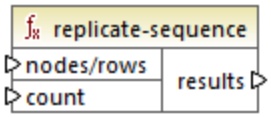
```
<TargetLists>
   <TargetList>
      <TargetString>Michelle</TargetString>
      <TargetString>Michelle</TargetString>
   </TargetList>
   <TargetList>
      <TargetString>Ted</TargetString>
```


© 2016-2022 Altova GmbH

```
      <TargetString>Ted</TargetString>
      <TargetString>Ted</TargetString>
      <TargetString>Ted</TargetString>
   </TargetList>
   <TargetList>
      <TargetString>Ann</TargetString>
      <TargetString>Ann</TargetString>
      <TargetString>Ann</TargetString>
   </TargetList>
</TargetLists>
```

6.7.9.16 replicate-sequence

Repliziert alle Datenelement in der Input-Sequenz so oft, wie im Argument **count** definiert. Wenn z.B. count **2** ist, so erzeugt die Sequenz `1,2,3` das Ergebnis `1,2,3,1,2,3`.



Sprachen

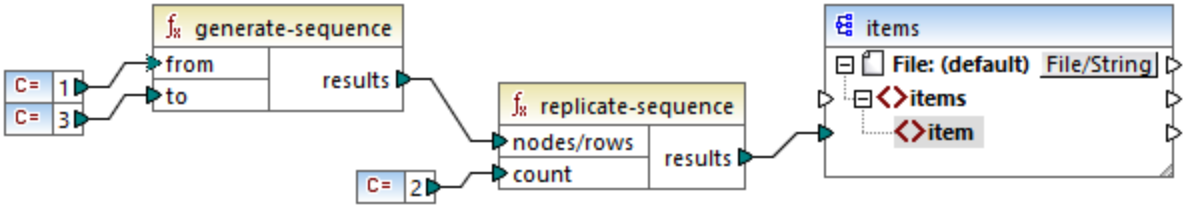
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
node-rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 1331 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
count	Definiert, wie oft die verbundene Sequenz repliziert werden soll.

Beispiel

Im folgenden Modell-Mapping wird die Sequenz `1,2,3` generiert. Die Sequenz wird von der Funktion `replicate-sequence` verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.

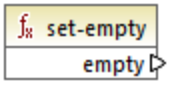


Da das **count**-Argument auf **2** gesetzt wurde, wird die Sequenz zwei Mal repliziert und an die Zielkomponente übergeben. Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

```
<items>
   <item>1</item>
   <item>2</item>
   <item>3</item>
   <item>1</item>
   <item>2</item>
   <item>3</item>
</items>
```

6.7.9.17 set-empty

Gibt eine leere Sequenz zurück. Damit können Sie z.B. Standardwerte[799] eines XBRL-Dokuments, die weiter oben in der XBRL-Komponente/Taxonomie definiert wurden, außer Kraft setzen.

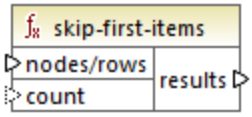


Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

6.7.9.18 skip-first-items

Überspringt die ersten N Datenelemente der Input-Sequenz, wobei N durch das **count** Argument angegeben wird, und gibt den Rest der Sequenz zurück.



© 2016-2022 Altova GmbH

Sprachen

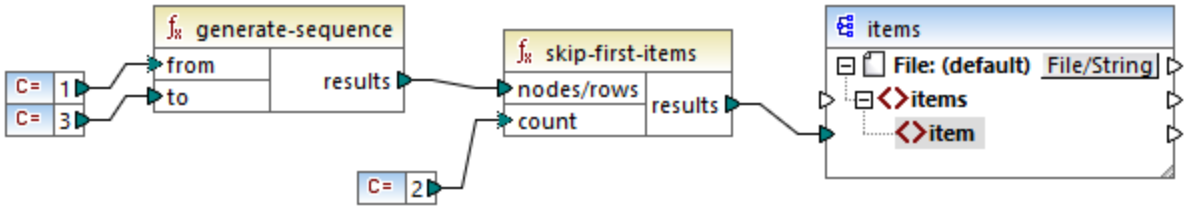
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
node-rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 1331 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
count	Optionales Argument. Definiert die Anzahl der Datenelemente, die übersprungen werden sollen. Der Standardwert ist **1**.

Beispiel

Im folgenden Modell-Mapping wird die Sequenz `1,2,3` generiert. Die Sequenz wird von der Funktion `skip-first-items` verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.

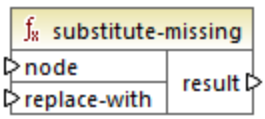


Da das **count**-Argument auf **2** gesetzt wurde, werden die ersten zwei Datenelemente übersprungen und die restlichen Datenelementen an die Zielkomponente übergeben. Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

```
<items>
   <item>3</item>
</items>
```

6.7.9.19 substitute-missing

Diese Funktion ist eine praktische Kombination aus der Funktion exists 1079 und einer "if-else"-Bedingung 918. Wenn das mit dem **node** Input verbundene Datenelement vorhanden ist, wird dessen Inhalt in die Zielkomponente kopiert. Andernfalls wird der Inhalt des mit dem **replace-with** Input verbundenen Datenelements in die Zielkomponente kopiert.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

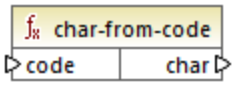
Name	Beschreibung
node	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 1331 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei, einem CSV-Feld, einem Datenbankdatensatz, usw. verbunden werden.
replace-with	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das den Ersetzungswert bereitstellt.

6.7.10 core | string functions (String-Funktionen)

Mit Hilfe von String-Funktionen können Sie String-Daten bearbeiten, um Abschnitte von Strings zu extrahieren, den String auf darin enthaltene Strings zu überprüfen oder Informationen über Strings abzurufen, Strings zu trennen usw.

6.7.10.1 char-from-code

Gibt die Zeichendarstellung des als Argument angegebenen Unicode-Dezimalwerts (Code) zurück. **Tipp:** Sie können den Unicode-Dezimalcode eines Zeichens mit Hilfe der Funktion **`code-from-char`** 1105 finden.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
code	Der Unicode-Wert als Dezimalzahl.

Beispiel

Laut den Tabellen auf der Unicode-Website (https://www.unicode.org/charts/) hat das Ausrufezeichen den Hexadezimalwert **`0021`**. Der entsprechende Wert im Dezimalformat ist **33**. Wenn Sie daher **`33`** als Argument für

© 2016-2022 Altova GmbH

die Funktion **char-from-code** bereitstellen, erhalten Sie das Zeichen **!** . Siehe auch Ersetzen von Sonderzeichen 283.

6.7.10.2 code-from-char

Gibt den Unicode-Dezimalwert (Code) des als Argument bereitgestellten Zeichens zurück. Wenn der als Argument bereitgestellte String mehrere Zeichen hat, wird der Code des ersten Zeichens zurückgegeben.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

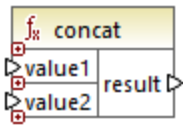
Name	Beschreibung
char	Der Input-String-Wert.

Beispiel

Wenn der Input **char** das Zeichen **$** (Dollarzeichen) ist, gibt die Funktion **36** zurück (Dies ist der Unicode-Dezimalwert dieses Zeichens).

6.7.10.3 concat

Verkettet zwei oder mehr Werte zu einem einzigen Ergebnisstring. Alle Input-Werte werden automatisch in den Typ "string" konvertiert. Standardmäßig hat diese Funktion nur zwei Parameter, Sie können jedoch weitere hinzufügen. Klicken Sie auf **Parameter hinzufügen** (⊡) oder **Parameter löschen** (⊠), um Parameter hinzuzufügen oder zu löschen.



Anmerkung: Alle Inputs für die `concat`-Funktion müssen einen Wert haben. Wenn einer der Inputs keinen Wert hat, wird die Funktion nicht aufgerufen und es tritt ein Fehler auf. Bedenken Sie, dass auch ein leerer String ein gültiger Input-Wert ist; eine leere Sequenz (wie z.B. das Ergebnis der Funktion **set-empty**) ist hingegen kein gültiger Wert, weswegen die Funktion als Ergebnis fehlschlägt. Um dies zu vermeiden, können Sie Werte zuerst mit der Funktion substitute-missing 1103 verarbeiten und das Ergebnis anschließend als Input für die **concat**-Funktion bereitstellen.

Sprachen

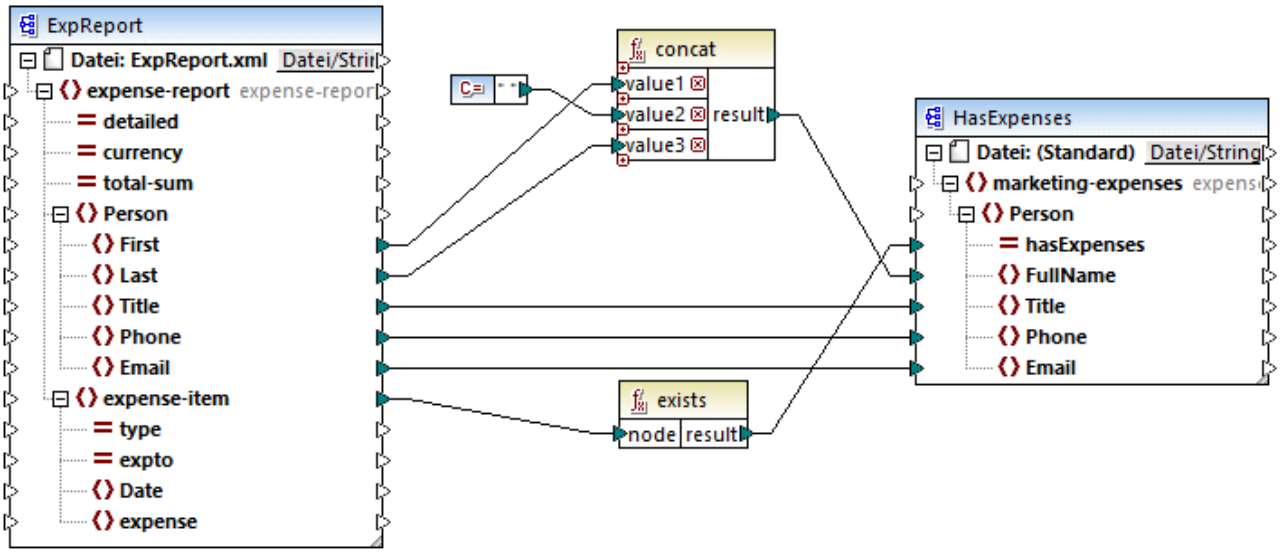
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
value1	Der erste Input-Wert.
value2	Der zweite Input-Wert.
valueN	Der Input-Wert *N*.

Beispiel

Die **concat**-Funktion im unten gezeigten Mapping verbindet den Vornamen, die Konstante " " und den Nachnamen miteinander. Der Ergebniswert wird anschließend in das Zieldatenelement **FullName** geschrieben. Sie finden das Mapping dieser Funktion unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\HasMarketingExpenses.mfd**.

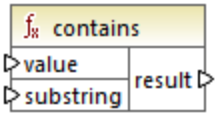


HasMarketingExpenses.mfd

6.7.10.4 contains

Gibt den Booleschen Wert **true** zurück, wenn das als String-Wert bereitgestellte Argument den als Argument bereitgestellten Substring enthält.

© 2016-2022 Altova GmbH



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

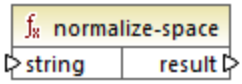
Name	Beschreibung
value	Der Input-Wert (d.h. der "Heuhaufen").
substring	Der Substring, nach dem gesucht wird (d.h. die "Nadel").

Beispiel

Wenn der Input **value** "category" und der Substring **substring** "cat" ist, gibt die Funktion **true** zurück.

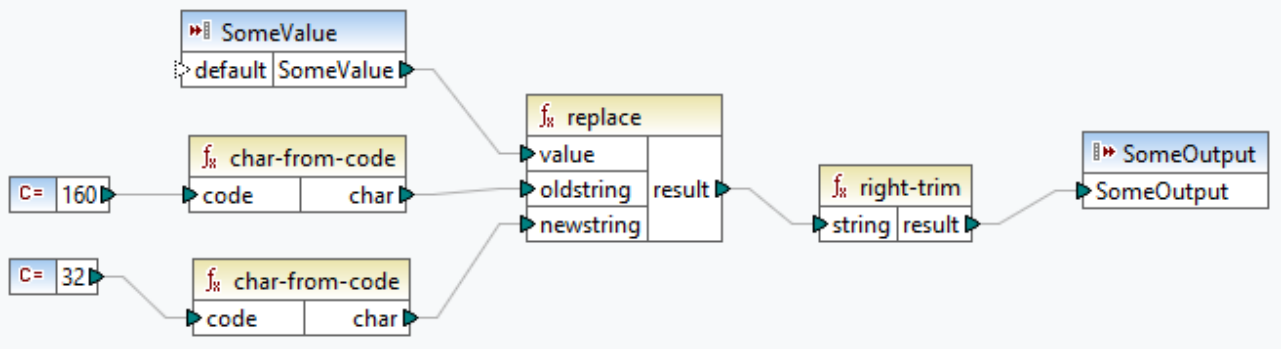
6.7.10.5 normalize-space

Die Funktion **`normalize-space`** function (*siehe Abbildung unten*) entfernt vorangestellte und nachgestellte Leerzeichen aus einem String und ersetzt interne Whitespace-Zeichen durch ein einziges Whitespace-Zeichen. Zu den Whitespace-Zeichen zählen das Leerzeichen (U+0020), der Tabulator (U+0009), der Wagenrücklauf (U+000D) und der Zeilenvorschub (U+000A). Nähere Informationen zu Whitespaces finden Sie in der XML Recommendation.



Informationen zu geschützten Leerzeichen

Die Funktionen **`left-trim`**, **`right-trim`** und **`normalize-space`** entfernen geschützte Leerzeichen nicht. Eine mögliche Lösung wäre, ein geschütztes Leerzeichen, dessen Dezimaldarstellung 160 ist, durch ein Leerzeichen mit der Dezimaldarstellung 32 zu ersetzen. Im nächsten Schritt wird der gekürzte Wert `Wert` auf das Zieldatenelement gemappt (*siehe Mapping unten*).



Wenn es sich bei Ihrer Quellkomponente um eine Excel-Datei handelt, können Sie überschüssige Leerzeichen in Excel mit Hilfe einer Kombination aus den Funktionen TRIM, CLEAN und SUBSTITUTE entfernen. Nähere Informationen dazu finden Sie unter *Entfernen von Leerzeichen und nicht druckbaren Zeichen aus dem Text*.

Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

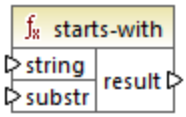
Name	Beschreibung
String	Der zu normalisierende Input-String.

Beispiel

Wenn der Input String `  The quick  brown fox  ` ist, gibt die Funktion `The quick brown fox` zurück.

6.7.10.6 starts-with

Gibt den Booleschen Wert **true** zurück, wenn das als String bereitgestellte Argument mit dem als Argument bereitgestellten Substring beginnt; gibt andernfalls **false** zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

© 2016-2022 Altova GmbH

Parameter

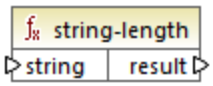
Name	Beschreibung
string	Der Input-String.
substr	Der Substring, auf den der Wert überprüft werden soll.

Beispiel

Wenn der Input **string** `category` und **substr** `cat` ist, gibt die Funktion **true** zurück.

6.7.10.7 string-length

Gibt die Anzahl der Zeichen in dem als Argument angegebenen String zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

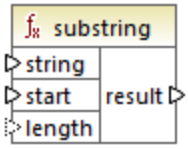
Name	Beschreibung
string	Der Input-String.

Beispiel

Wenn der Input-String `car` ist, gibt die Funktion **3** zurück. Wenn der Input-String ein leerer String ist, gibt die Funktion **0** zurück.

6.7.10.8 substring

Gibt den Anteil des durch die Parameter **start** und **length** definierten Strings zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

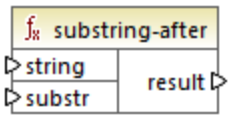
Name	Beschreibung
string	Der Input-String.
start	Definiert die Anfangsposition (Index), ab der der Substring abgerufen werden soll. Der erste Index ist **1**.
length	Optional. Definiert die Anzahl der abzurufenden Zeichen. Wenn der Parameter **length** nicht angegeben wird, ist das Ergebnis ein Fragment ab der Position **start** bis zum Ende des Strings.

Beispiel

Wenn der Input-String `MapForce` ist, "start" **1** und length **3** ist, gibt die Funktion `Map` zurück. Wenn der Input-String `MapForce`, "start" **4** ist und "length" nicht angegeben wird, gibt die Funktion `Force` zurück.

6.7.10.9 substring-after

Gibt den Teil des String zurück, der hinter der ersten Instanz von **substr** steht. Wenn **substr** in **string** nicht vorkommt, gibt die Funktion einen leeren String zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
string	Der Input-String.
substr	Der Substring. Das Ergebnis dieser Funktion sind alle Zeichen nach der ersten Instanz von **substr**.

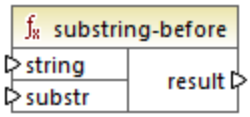
© 2016-2022 Altova GmbH

Beispiel

Wenn der Input-String **MapForce** und **substr** **Map** ist, gibt die Funktion **Force** zurück. Wenn der Input-String **2020/01/04** und **substr** **/** ist, gibt die Funktion **01/04** zurück. Siehe auch Beispiel: Mappen von Daten von einem RSS Feed 644.

6.7.10.10 substring-before

Gibt den Teil des String zurück, der vor der ersten Instanz von **substr** steht. Wenn **substr** in **string** nicht vorkommt, gibt die Funktion einen leeren String zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

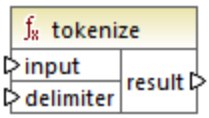
Name	Beschreibung
string	Der Input-String.
substr	Der Substring. Das Ergebnis dieser Funktion sind alle Zeichen vor der ersten Instanz von **substr**.

Beispiel

Wenn der Input-String **MapForce** und **substr** **Force** ist, gibt die Funktion **Map** zurück. Wenn der Input-String **2020/01/04** und **substr** **/** ist, gibt die Funktion **2020** zurück.

6.7.10.11 tokenize

Teilt den Input-String mit Hilfe des als Argument angegebenen Trennzeichens in eine Sequenz von Strings auf.



Sprachen

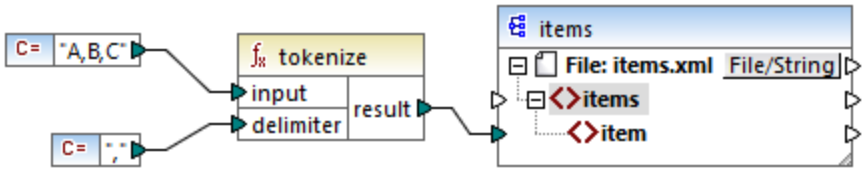
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
input	Der Input-String.
delimiter	Das zu verwendende Trennzeichen.

Beispiel

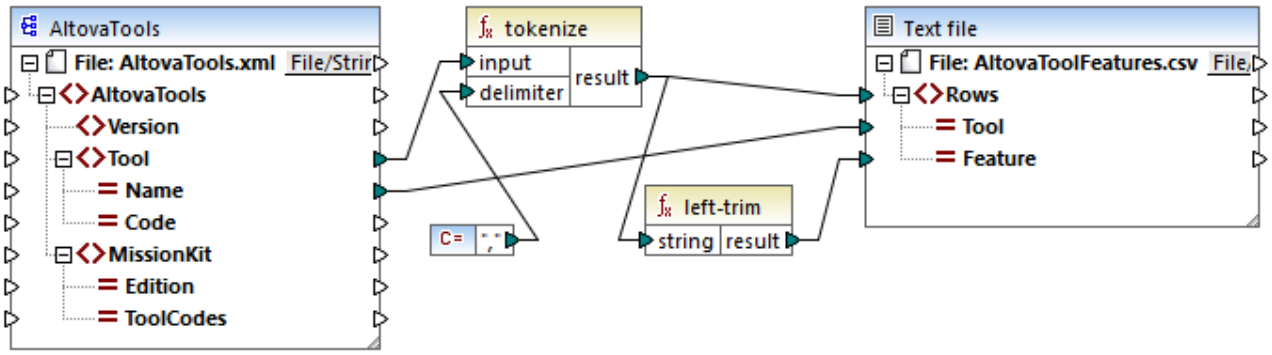
Wenn der Input-String `A,B,C` und "delimiter" `,` ist, gibt die Funktion eine Sequenz von drei Strings zurück: `A`, `B` und `C`.



Im oben gezeigten Modell-Mapping ist das Ergebnis der Funktion eine Sequenz von Strings. Entsprechend den allgemeinen Mappingregeln[1331] wird für jedes Datenelement in der Quellsequenz ein neues Datenelement **item** in der Zielkomponente erstellt. Die Mapping-Ausgabe sieht daher folgendermaßen aus:

```
<items>
   <item>A</item>
   <item>B</item>
   <item>C</item>
</items>
```

Ein komplexeres Beispiel dazu finden Sie im Mapping **tokenizeString1.mfd** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**.



tokenizeString1.mfd

Unten sehen Sie ein Fragment aus der XML-Quelldatei. Das Element **Tool** hat zwei Attribute: **Name** und **Code**. Das Element **Tool** besteht aus kommagetrenntem Text.

© 2016-2022 Altova GmbH

```
<?xml version="1.0" encoding="UTF-8"?>
<AltovaTools xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="AltovaTools.xsd">
   <Version>2010</Version>
   <Tool Name="XMLSpy" Code="XS">XML editor, XSLT editor, XSLT debugger, XQuery editor,
XQuery debugger, XML Schema / DTD editor, WSDL editor, SOAP debugger</Tool>
   <Tool Name="MapForce" Code="MF">Data integration, XML mapping, database mapping, text
conversion, EDI translator, Excel mapping, XBRL mapping, Web services</Tool>
   <Tool Name="StyleVision" Code="SV">Stylesheet designer, electronic forms, XSLT design,
XSL:FO design, database reporting, XBRL rendering</Tool>
   <Tool Name="UModel" Code="UM">UML modeling tool, code generation, reverse engineering,
UML, BPMN, SysML, project documentation, XMI interchange</Tool>
   <Tool Name="DatabaseSpy" Code="DS">Multi-database tool, SQL auto-completion, graphical
database design, table browser, content editor, database comparison tool</Tool>
   <!-- ... -->
</AltovaTools>
```

Im Mapping geschieht Folgendes:

- Die **tokenize**-Funktion erhält Daten aus dem Quelldatenelement **Tool** und unterteilt die Daten mit Hilfe des Komma-Trennzeichens , in separate Blöcke. Der erste Block ist "XML editor", der zweite "XSLT editor", usw.
- Für jeden mit der **tokenize**-Funktion erzeugten Block wird in der Zielkomponente eine neue Zeile generiert. Dies geschieht aufgrund der Verbindung zwischen dem Ergebnis der Funktion und dem Datenelement **Rows** in der Zielkomponente.
- Außerdem wird das Ergebnis der **tokenize**-Funktion auf die Funktion **left-trim** gemappt, die in jedem Block das vorangestellte Leerzeichen entfernt.
- Das Ergebnis der **left-trim**-Funktion (die einzelnen Blöcke) wird in das Datenelement **Feature** der Zielkomponente geschrieben.
- Die Ausgabedatei der Zielkomponente wurde als CSV-Datei (**AltovaToolFeatures.csv**) definiert, wobei als Feldtrennzeichen ein Semikolon definiert wurde (Doppelklicken Sie auf die Komponente, um die Einstellungen zu sehen).

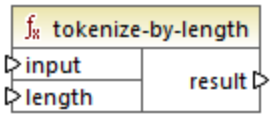
Das Ergebnis des Mappings ist, dass für jeden von der **tokenize**-Funktion erstellten Block in der CSV-Zieldatei eine neue Zeile erstellt wird. Ein Fragment der Mapping-Ausgabe sieht folgendermaßen aus:

```
Tool;Feature
XMLSpy;XML editor
XMLSpy;XSLT editor
XMLSpy;XSLT debugger
XMLSpy;XQuery editor
XMLSpy;XQuery debugger
XMLSpy;XML Schema / DTD editor
XMLSpy;WSDL editor
XMLSpy;SOAP debugger
MapForce;Data integration
MapForce;XML mapping
MapForce;database mapping
MapForce;text conversion
MapForce;EDI translator
```

```
MapForce;Excel mapping
```

6.7.10.12 tokenize-by-length

Teilt den Input-String in eine Sequenz von Strings auf. Die Größe der einzelnen erzeugten Strings wird durch den Parameter **length** definiert.



Sprachen

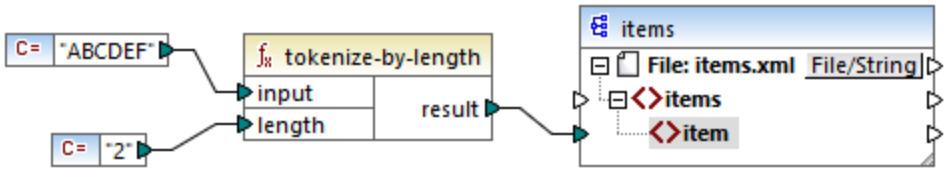
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
input	Der Input-String.
length	Definiert die Länge der einzelnen Strings in der generierten Stringsequenz.

Beispiel

Wenn der Input-String **ABCDEF** und "length " **2** ist, gibt die Funktion eine Sequenz von drei Strings zurück: **AB**, **CD** und **EF**.

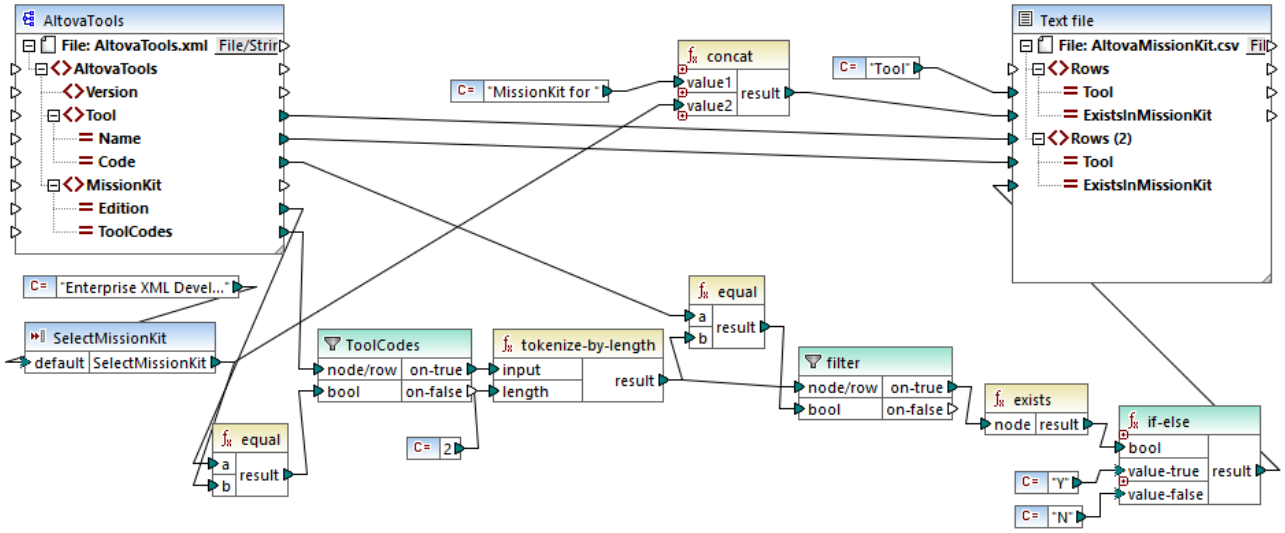


Im oben gezeigten Modell-Mapping ist das Ergebnis der Funktion eine Sequenz von Strings. Entsprechend den allgemeinen Mappingregeln[1331] wird für jedes Datenelement in der Quellsequenz ein neues Datenelement **item** in der Zielkomponente erstellt. Die Mapping-Ausgabe sieht daher folgendermaßen aus:

```
<items>
   <item>AB</item>
   <item>CD</item>
   <item>EF</item>
</items>
```


© 2016-2022 Altova GmbH

Ein komplexeres Beispiel dazu finden Sie im Mapping **tokenizeString2.mfd** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**.



tokenizeString2.mfd

Die unten gezeigte XML-Quelldatei ist dieselbe wie die im vorhergehenden Beispiel verwendete. Das Element **MissionKit** hat zwei Attribute: **Edition** und **ToolCodes**, aber keinen Inhalt für das Element "MissionKit". Beachten Sie, dass ein Teil des für dieses Beispiel nicht relevanten XML-Inhalts entfernt wurde.

```
<?xml version="1.0" encoding="UTF-8"?>
<AltovaTools xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="AltovaTools.xsd">
   <Version>2010</Version>
   <Tool Name="XMLSpy" Code="XS"><!--...--></Tool>
   <Tool Name="MapForce" Code="MF"><!--...--></Tool>
   <Tool Name="StyleVision" Code="SV"><!--...--></Tool>
   <Tool Name="UModel" Code="UM"><!--...--></Tool>
   <Tool Name="DatabaseSpy" Code="DS"><!--...--></Tool>
   <Tool Name="DiffDog" Code="DD"><!--...--></Tool>
   <Tool Name="SchemaAgent" Code="SA"><!--...--></Tool>
   <Tool Name="SemanticWorks" Code="SW"><!--...--></Tool>
   <Tool Name="Authentic" Code="AU"><!--...--></Tool>
   <MissionKit Edition="Enterprise Software Architects" ToolCodes="XSMFSVUMDSDDSASW"/>
   <MissionKit Edition="Professional Software Architects" ToolCodes="XSMFSVUMDS"/>
   <MissionKit Edition="Enterprise XML Developers" ToolCodes="XSMFSVDDSASW"/>
   <MissionKit Edition="Professional XML Developers" ToolCodes="XSMFSV"/>
</AltovaTools>
```

Ziel ist die Generierung einer Liste, in der Sie sehen, welche Altova-Tools jeweils Teil der entsprechenden MissionKit Editions sind.

Funktionsweise des Mappings:

- Die **SelectMissionKit** Input-Komponente fungiert als Parameter für das Mapping; sie erhält ihren Standardwert von einer Konstante, in diesem Fall von "Enterprise XML Developers".
- Die `equal`-Funktion vergleicht die als Parameter angegebene Edition mit dem Datenelement **Edition** aus der XML-Quelldatei und übergibt das Ergebnis an den Parameter **bool** des **ToolCodes**-Filters.
- Der **node/row**-Input des **ToolCodes**-Filters stammt aus dem Quelldatei-Datenelement **ToolCodes**. Der Wert für die Enterprise XML Developers Edition ist: `XSMFSVDDSASW`.
- Der Wert `XSMFSVDDSASW` wird an den **on-true**-Parameter übergeben und dann an den **input**-Parameter der `tokenize-by-length` Funktion.
- Die `tokenize-by-length`-Funktion teilt den Wert `XSMFSVDDSASW` in mehrere aus je zwei Zeichen bestehende Blöcke auf. Der Parameter **length** beträgt **2**; daher werden als Ergebnis 6 Blöcke erzeugt.
- Jeder Block wird mit dem aus 2 Zeichen bestehenden **Code**-Wert der Quelldatei verglichen (von der es insgesamt 9 Einträg/Datenelemente gibt). Das Ergebnis des Vergleichs (true/false) wird an den **bool**-Parameter des Filters übergeben. Beachten Sie, dass *alle* Blöcke der `tokenize-by-length` Funktion an den **node/row**-Parameter des Filters übergeben werden.
- Die `exists`-Funktion überprüft nun die Datei auf existing/non-existing (existierende/nicht existente) Nodes, die vom **on-true** Parameter der Filter-Komponente an die exists-Funktion übergeben werden. Existierende Nodes sind jene, für die *eine Übereinstimmung* zwischen dem **ToolCodes**-Block und dem **Code**-Wert besteht. Nicht existierende Nodes sind diejenigen, die keinen **ToolCodes**-Block enthalten haben, der mit einem **Code**-Wert übereingestimmt hat.
- Die einzelnen **bool**-Ergebniss der `exists`-Funktion werden an die **if-else** Funktion übergeben, die in der Zielkomponente ein "Y" generiert, wenn der Node vorhanden ist oder ein "N", wenn der Node nicht existiert.

Des Ergebnis des Mappings sieht folgendermaßen aus:

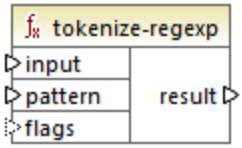
```
Tool;MissionKit for Enterprise XML Developers
XMLSpy;Y
MapForce;Y
StyleVision;Y
UModel;N
DatabaseSpy;N
DiffDog;Y
SchemaAgent;Y
SemanticWorks;Y
Authentic;N
```

6.7.10.13 tokenize-regexp

Teilt den Input-String in eine Sequenz von Strings auf. Als Trennzeichen wird ein beliebiger Substring definiert, der mit dem **pattern** (Muster) der als Argument bereitgestellten Regular Expression übereinstimmt. Die gefundenen Trennzeichen-Strings werden nicht in das durch die Funktion zurückgegebene Ergebnis inkludiert.

Anmerkung: Die komplexen Funktionalitäten der Regular Expression-Syntax können bei der Generierung von C++-, C#- oder Java-Code etwas unterschiedlich sein. Nähere Informationen dazu finden Sie in der regex-Dokumentation zu den einzelnen Sprachen.

© 2016-2022 Altova GmbH



Sprachen

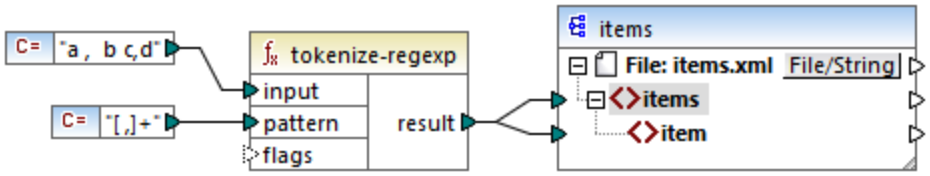
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
input	Der Input-String.
pattern	Gibt ein Muster für eine Regular Expression an. Jeder beliebige Substring, der mit dem Muster übereinstimmt, wird als Trennzeichen verwendet. Nähere Informationen dazu finden Sie unter Regular Expressions 1020.
flags	Optionaler Parameter. Stellt die Flags 1022 bereit, die für die Regular Expression verwendet werden sollen. So verarbeitet das Mapping etwa den Input aufgrund des Flags "i" ohne Berücksichtigung der Groß- und Kleinschreibung.

Beispiel

Ziel des unten gezeigten Mappings ist es, den String `a ,  b c,d` in eine Stringsequenz aufzuteilen, wobei jedes alphabetische Zeichen ein Element in der Sequenz bildet. Überzählige Whitespace-Zeichen oder Kommas müssen entfernt werden.



Zu diesem Zweck wurde als Parameter für die `tokenize-regexp` Funktion das Ausdrucksmuster `[ ,]+` bereitgestellt. Dieses Muster hat folgende Bedeutung:

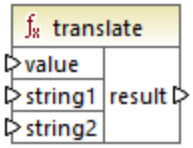
- Es steht für jedes der Zeichen innerhalb der Zeichenklasse `[ ,]`. Daher wird der Input-String überall dort, wo ein Komma oder Leerzeichen steht, aufgeteilt.
- Der Quantifizierer `+` gibt an, dass eine oder mehrere Instanzen des vorangestellten Zeichens übereinstimmen müssen. Ohne diesen Quantifizierer würde für jede Instanz eines Leerzeichens oder Kommas ein separates Datenelement in der Stringsequenz erzeugt, was nicht das Ziel der Aufgabe ist.

Das Ergebnis des Mappings sieht folgendermaßen aus:

```
<items>
   <item>a</item>
   <item>b</item>
   <item>c</item>
   <item>d</item>
</items>
```

6.7.10.14 translate

Führt eine Zeichen-für-Zeichen-Ersetzung durch. Sucht in **value** nach in **string1** enthaltenen Zeichen und ersetzt jedes Zeichen durch das Zeichen an derselben Position in **string2**. Wenn es in **string2** keine entsprechenden Zeichen gibt, wird das Zeichen entfernt.



Sprachen

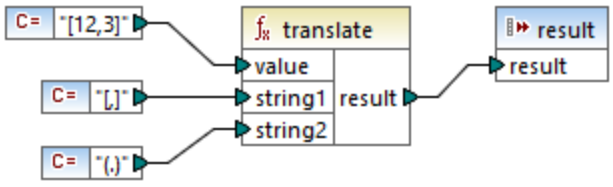
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
value	Der Input-String.
string1	Stellt eine Liste von zu suchenden Zeichen bereit. Die Position der einzelnen Zeichen im String ist von Bedeutung.
string2	Stellt eine Liste von Ersetzungszeichen bereit. Die Position der einzelnen Ersetzungszeichen muss derjenigen in **string1** entsprechen.

Beispiel

Angenommen, Sie möchten den String `[12,3]` in `(12.3)` konvertieren. D.h. die eckigen Klammern müssen durch runde Klammern, Kommas durch einen Punkt ersetzt werden. Sie können zu diesem Zweck die Funktion `translate` folgendermaßen verwenden:



© 2016-2022 Altova GmbH

Die erste Konstante im oben gezeigten Mapping liefert den zu verarbeitenden Input-String. Die zweite und dritte Konstante liefern als **string1** bzw. **string2**eine Liste von Zeichen.

string1 `[,]`

string2 `(.)`

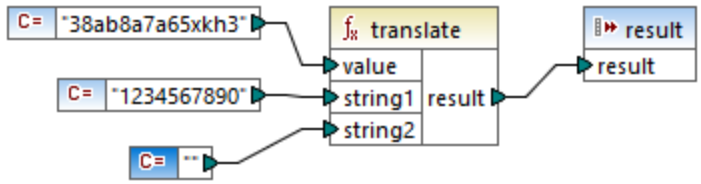
Beachten Sie dass **string1** und **string2** dieselbe Anzahl an Zeichen haben. Für jedes Zeichen in **string1** wird als Ersetzungszeichen das entsprechende Zeichen an derselben Position aus **string2** verwendet. Folglich werden die folgenden Ersetzungen vorgenommen:

- Jedes **[**-Zeichen wird durch ein **(**-Zeichen ersetzt.
- Jedes **,**-Zeichen wird durch ein **.**-Zeichen ersetzt.
- Jedes **]**-Zeichen wird durch ein **)**-Zeichen ersetzt.

Das Ergebnis des Mappings sieht folgendermaßen aus:

```
(12,3)
```

Diese Funktion kann auch dazu verwendet werden, um bestimmte Zeichen selektiv aus einem String zu entfernen. Setzen Sie dazu den Parameter **string1** auf das Zeichen, das entfernt werden soll und **string2** auf einen leeren String. Im unten gezeigten Mapping werden z.B. alle Ziffern aus dem String **38ab8a7a65xkh3** entfernt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

```
abaaxkh
```

6.7.11 bson

Dieser Abschnitt enthält Informationen über bson-Funktionen. Nähere Informationen über diese Funktionen finden Sie in den jeweiligen Kapiteln unten:

- bson | binary functions (Binärfunktionen) 1120
- bson | constructors (Konstruktoren) 1120
- bson | regex functions (Regex-Funktionen) 1121

Nähere Informationen zu BSON-Typen finden Sie in der BSON-Spezifikation.

Achtung: Die bson-Funktionen sind nur mit der Built-In 29-Transformationssprache kompatibel.

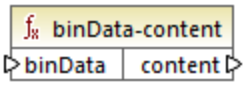
6.7.11.1 bson | binary functions

In diesem Unterabschnitt werden Binärfunktionen beschrieben. BSON-Binärwerte bestehen aus zwei Teilen: Sie haben einen in der BSON-Spezifikation beschriebenen Subtyp und Binärdateninhalt. Mit Hilfe der unten beschriebenen Funktionen können Sie den Inhalt aus einem BSON-Binärwert (**`binData-content`**) extrahieren, den Subtyp dieses Binärwerts (**`binData-subtype`**) ermitteln und auf Basis der Parameter `subtype` und `content` einen Binärwert (**`to-binData`**) erzeugen.

Achtung: Die `bson`-Funktionen sind nur mit der Built-In[29]-Transformationssprache kompatibel.

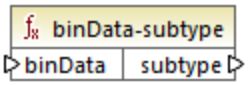
- binData-content

  Mit der Funktion **`binData-content`** (*siehe Abbildung unten*) können Sie Inhalt aus einem BSON-Binärwert extrahieren.



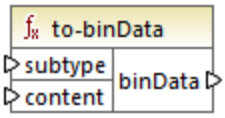
- binData-subtype

  BSON-Binärwerte haben einen Subtyp, der einen im Byte-Array gespeicherten Datentyp definiert. In der BSON-Spezifikation finden Sie eine Liste der BSON-Subtypen. Mit der Funktion **`binData-subtype`** (*siehe Abbildung unten*) können Sie den Subtyp aus einem BSON-Binärwert extrahieren.



- to-binData

  Mit der Funktion **`to-binData`** (*siehe Abbildung unten*) können Sie durch Angabe der Parameter `subtype` und `content` einen BSON-Binärwert erzeugen.



6.7.11.2 bson | constructors

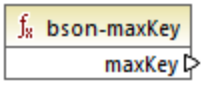
In diesem Unterabschnitt werden BON-Konstruktorfunktionen im Zusammenhang mit `maxKey`, `minKey`, `null` und `undefined` (nicht definiert)-Werten behandelt. Alle diese Funktionen sind in der BSON-Spezifikation definiert.

Achtung: Die `bson`-Funktionen sind nur mit der Built-In[29]-Transformationssprache kompatibel.

© 2016-2022 Altova GmbH

- bson-maxKey

  Mit Hilfe der Funktion **`bson-maxKey`** (*siehe Abbildung unten*) können Sie im ausgewählten Node der Datenbankkomponente einen `maxKey`-Wert konstruieren.



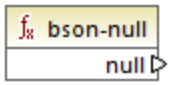
- bson-minKey

  Mit Hilfe der Funktion **`bson-minKey`** (*siehe Abbildung unten*) können Sie im ausgewählten Node der Datenbankkomponente einen `minKey`-Wert konstruieren.



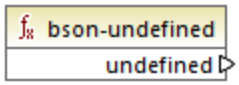
- bson-null

  Die Funktion **`bson-null`** (*siehe Abbildung unten*) konstruiert im ausgewählten Node Ihrer Datenbankkomponente einen BSON-Nullwert.



- bson-undifined

  Mit Hilfe der Funktion **`bson-undefined`** (*siehe Abbildung unten*) können Sie im ausgewählten Node der Datenbankkomponente einen `undefined` (nicht definiert)-Wert konstruieren.



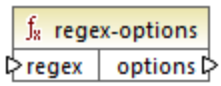
6.7.11.3 bson | regex functions

In diesem Unterabschnitt werden Regex-Funktionen zur Durchführung von Mustervergleichen beschrieben. Laut BSON-Spezifikation bestehen Regular Expression-Werte aus zwei Teilen: Sie haben ein Muster und Optionen. Mit Hilfe der unten aufgelisteten Funktionen können Sie diese zwei Teile voneinander trennen und jeweils extrahieren (die **`regex-options`**- und **`regex-pattern`**-Funktionen). Sie können auch einen Regex-Ausdruck auf Basis der Parameter `pattern` und `options` erzeugen (**`to-regex`**).

Achtung: Die `bson`-Funktionen sind nur mit der Built-In[29]-Transformationssprache kompatibel.

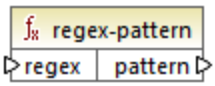
- regex-options

Mit der Funktion **regex-options** (*siehe Abbildung unten*) können Sie die Optionen aus einem BSON-Regex-Wert extrahieren.



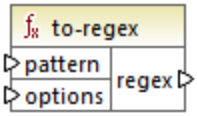
- regex-pattern

  Mit der Funktion **regex-pattern** (*siehe Abbildung unten*) können Sie das Muster aus einem BSON-Regex-Wert extrahieren.



- to-regex

  Mit der Funktion **to-regex** (*siehe Abbildung unten*) können Sie auf Basis der Parameter `pattern` und `options` einen BSON-Regex-Wert erzeugen.

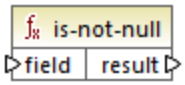


6.7.12 db

Die Bibliothek "db" enthält Funktionen, mit denen Mapping-Ergebnisse definiert werden können, wenn Datenbanken Felder mit dem Wert Null enthalten.

6.7.12.1 is-not-null

Gibt **false** zurück, wenn das Feld Null ist und **true**, wenn dies nicht der Fall ist.



Sprachen

Built-in, C++, C#, Java.

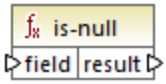
© 2016-2022 Altova GmbH

Parameter

Name	Beschreibung
field	Das Datenbankfeld.

6.7.12.2 is-null

Gibt **true** zurück, wenn das Feld Null ist und **false**, wenn dies nicht der Fall ist.



Sprachen

Built-in, C++, C#, Java.

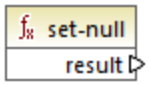
Parameter

Name	Beschreibung
field	Das Datenbankfeld.

6.7.12.3 set-null

Setzt ein Datenbankfeld auf Null. Mit dieser Funktion wird auch ein Standardwert durch Null überschrieben. Wenn die Funktion mit etwas verbunden ist, bei dem es sich nicht um ein Datenbankfeld handelt, verhält sie sich wie einer leere Sequenz. Beachten Sie dazu Folgendes:

- Wenn Sie **`set-null`** mit einer anderen Funktion verbinden, hat dies normalerweise zur Folge, dass die andere Funktion gar nicht aufgerufen wird. Wenn Sie **`set-null`** mit einer Sequenzfunktion wie z.B. **`count`** verbinden, so wird die Funktion mit einer leeren Sequenz aufgerufen.
- Wenn Sie **`set-null`** mit Filtern und IF-Else-Bedingungen verbinden, so werden die Felder, wie erwartet, auf Null gesetzt. Bei Filtern betrifft dies den "node/row"-Input.
- Bei Verwendung von **`set-null`** als Input für ein `simpleType`-Element wird dieses Element in der Zielkomponente nicht erstellt.

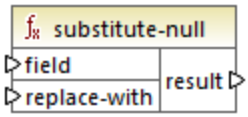


Sprachen

Built-in, C++, C#, Java.

6.7.12.4 substitute-null

Gibt das Feld selbst zurück, wenn es nicht Null ist. Andernfalls wird **replace-with** zurückgegeben.



Sprachen

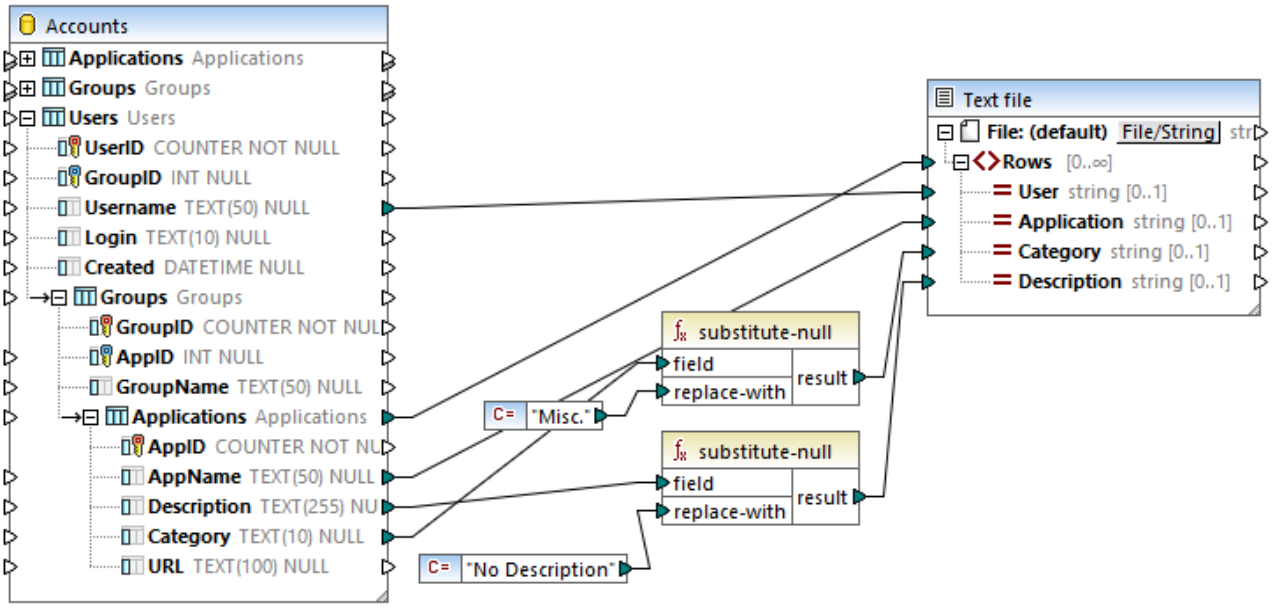
Built-in, C++, C#, Java.

Parameter

Name	Beschreibung
field	Das Datenbankfeld.
replace-with	Der Ersetzungswert.

Beispiel

Im Mapping unten sehen Sie ein Beispiel für die Verwendung der **`substitute-null`**-Funktion. Dieses Mapping hat den Namen **DB-ApplicationList.mfd** und befindet sich im Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**.



Im Mapping werden Daten aus einer Access-Datenbank, die eine Tabelle namens "Applications" enthält, ausgelesen.

© 2016-2022 Altova GmbH

	AppID	AppName	Description	Category	URL
1	1	Altova MapForce	The premier data mapping tool.	IDE	www.altova.com/xmlspy
2	2	Notepad	*[NULL]*	*[NULL]*	*[NULL]*

Die erste Funktion überprüft, ob das Feld **Category** in der Tabelle "Applications" Null ist. Da dieses Feld für die Applikation "Notepad" Null ist, wird der Ersetzungswert "Misc" auf das Datenelement **Category** der Zieltextdatei gemappt.

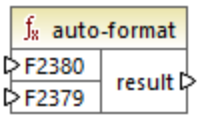
Die zweite Funktion überprüft, ob das Feld **Description** Null ist. Da auch dieses Feld für die Applikation "Notepad" Null ist, wird der Ersetzungswert "No description" auf das Datenelement **Description** der Zieldatei gemappt.

6.7.13 edifact

Die **edifact**-Bibliothek enthält Funktionen für Mappings im Zusammenhang mit UN/EDIFACT-Komponenten, siehe UN/EDIFACT[527]. Die Funktionen dienen zum Konvertieren von EDI-Datums-, Uhrzeit- und Zeitraumwerten in `xs:date`-Formate.

6.7.13.1 auto-format

Gibt den aus dem Datenelement **F2380** ("Datums- oder Uhrzeit- oder Zeitraumwert") extrahierten, mit Hilfe des Datenelements **F2379** ("Datums- oder Uhrzeit- oder Zeitraum-Formatcode") formatierten `date`-, `time`- oder `datetime`-Wert zurück.



Sprachen

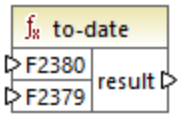
Built-in, C++, C#, Java.

Parameter

Name	Beschreibung
F2380	Dieser Input mit mit einem **F2380**-Datenelement einer UN/EDIFACT-Komponente verbunden werden.
F2379	Dieser Input mit mit einem **F2379**-Datenelement einer UN/EDIFACT-Komponente verbunden werden.

6.7.13.2 to-date

Gibt den aus dem Datenelement **F2380** ("Datums- oder Uhrzeit- oder Zeitraumwert") extrahierten, mit Hilfe des Datenelements **F2379** ("Datums- oder Uhrzeit- oder Zeitraum-Formatcode") formatierten `date`-Wert zurück.



Sprachen

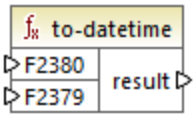
Built-in, C++, C#, Java.

Parameter

Name	Beschreibung
F2380	Dieser Input mit mit einem **F2380**-Datenelement einer UN/EDIFACT-Komponente verbunden werden.
F2379	Dieser Input mit mit einem **F2379**-Datenelement einer UN/EDIFACT-Komponente verbunden werden.

6.7.13.3 to-datetime

Gibt den aus dem Datenelement **F2380** ("Datums- oder Uhrzeit- oder Zeitraumwert") extrahierten, mit Hilfe des Datenelements **F2379** ("Datums- oder Uhrzeit- oder Zeitraum-Formatcode") formatierten `datetime`-Wert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Beschreibung
F2380	Dieser Input muss mit einem **F2380**-Datenelement einer UN/EDIFACT-Komponente verbunden werden.

© 2016-2022 Altova GmbH

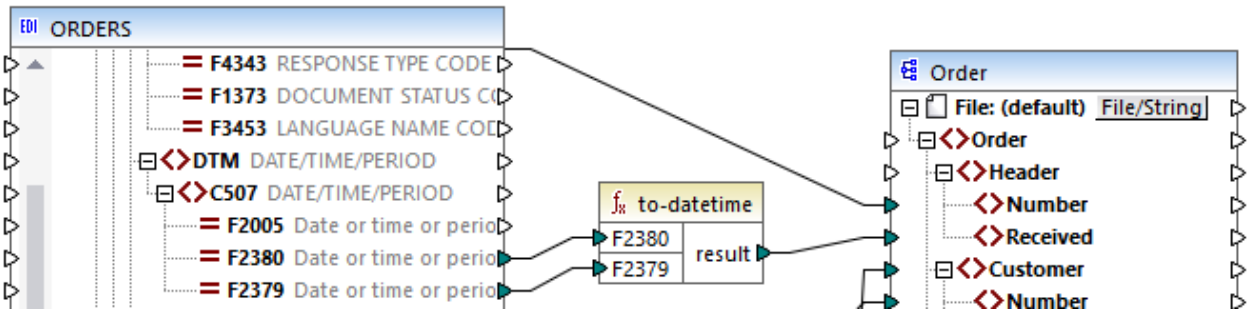
Name	Beschreibung
F2379	Dieser Input muss mit einem **F2379**-Datenelement einer UN/EDIFACT-Komponente verbunden werden.

Beispiel

Im Codefragment unten sehen Sie ein Fragment einer UN/EDIFACT-Datei, in der in der mit "DTM" beginnenden Zeile die Datenelemente **F2380** und **F2379** vorkommen. Der Wert von **F2380** ist hier `200404301742PDT` und der Wert von **F2379** ist `303`.

```
UNB+UNOB:1+003897733:01:MFGB+PARTNER ID:ZZ:ROUTING ADDR+970101:1230+00000000000001+
+ORDERS++++1'
UNH+0001+ORDERS:D:19B:UN'
BGM+221+ABC123456XYZ+9'
DTM+4:200404301742PDT:303'
FTX+PUR+3++Pizza purchase order'
RFF+CT:123-456'
RFF+CR:1122'
NAD+SE+999::92++24h Pizza+Long Way+San-Francisco+CA+34424+US'
CTA+SR+:Ted Little'
```

Im unten gezeigten Mapping wird der Wert von **F2380** extrahiert und mit Hilfe der `to-datetime`-Funktion mit dem Code von **F2379** formatiert. Dieses Mapping hat den Namen **EDI-Order.mfd** und befindet sich im Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**.

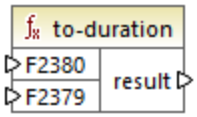


Unten sehen Sie den relevanten Abschnitt der Mapping-Ausgabe.

```
<?xml version="1.0" encoding="UTF-8"?>
<Order>
   <Header>
      <Number>ABC123456XYZ</Number>
      <Received>2004-04-30T17:42:00-09:00</Received>
   </Header>
   <!-- ... -->
</Order>
```

6.7.13.4 to-duration

Gibt den aus dem Datenelement **F2380** ("Datums- oder Uhrzeit- oder Zeitraumwert") extrahierten, mit Hilfe des Datenelements **F2379** ("Datums- oder Uhrzeit- oder Zeitraum-Formatcode") formatierten `duration`-Wert zurück.



Sprachen

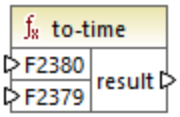
Built-in, C++, C#, Java.

Parameter

Name	Beschreibung
F2380	Dieser Input muss mit einem **F2380**-Datenelement einer UN/EDIFACT-Komponente verbunden werden.
F2379	Dieser Input muss mit einem **F2379**-Datenelement einer UN/EDIFACT-Komponente verbunden werden.

6.7.13.5 to-time

Gibt den aus dem Datenelement **F2380** ("Datums- oder Uhrzeit- oder Zeitraumwert") extrahierten, mit Hilfe des Datenelements **F2379** ("Datums- oder Uhrzeit- oder Zeitraum-Formatcode") formatierten `time`-Wert (Uhrzeitwert) zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Beschreibung
F2380	Dieser Input muss mit einem **F2380**-Datenelement einer UN/EDIFACT-Komponente verbunden werden.

© 2016-2022 Altova GmbH

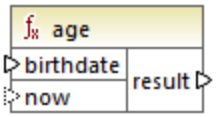
Name	Beschreibung
F2379	Dieser Input muss mit einem **F2379**-Datenelement einer UN/EDIFACT-Komponente verbunden werden.

6.7.14 lang | datetime functions (Datums- und Uhrzeitfunktionen)

Mit Hilfe der Datums- und Uhrzeitfunktionen aus der **lang**-Bibliothek können Sie Datums-, Uhrzeit- und Zeitraumwerte bearbeiten. Im Gegensatz zu den Datums- und Uhrzeitfunktionen aus der **core**-Bibliothek stehen diese Funktionen nur zur Verfügung, wenn die Sprachen Built-in, Java, C# oder C++ ausgewählt sind.

6.7.14.1 age

Gibt die Anzahl der vollständigen zwischen dem als Argument angegebenen Geburtsdatum und heute vergangenen Jahre zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
birthdate	`xs:date`	Obligatorisch. Gibt das Geburtsdatum als `xs:date`-Wert an.
now	`xs:date`	Optionaler Parameter. Der Standardwert ist das aktuelle Systemdatum. Wenn ein Wert auf das Argument now gemappt wird, ist das Ergebnis der Funktion der Unterschied zwischen dem Geburtsdatum und jetzt - und zwar in vollständigen Jahren.

6.7.14.2 convert-to-utc

Konvertiert den als Argument bereitgestellten Uhrzeitwert in UTC (Coordinated Universal Time). Die Funktion berücksichtigt dabei die Zeitzonenkomponente (z.B. "+5:00").



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
time	`xs:dateTime`	Liefert den zu konvertierenden `xs:dateTime`-Wert.

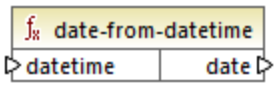
Beispiel

Wenn der Input-Wert `2001-12-17T09:30:02+05:00` ist, ist das Ergebnis der Funktion `2001-12-17T04:30:02`.

Wenn der Input-Wert `2001-12-17T09:30:02Z` ist, ist das Ergebnis der Funktion `2001-12-17T09:30:02`. In diesem Fall wurde keine Konvertierung durchgeführt, da das nachgestellte "Z" diese Uhrzeit bereits als "Null" (oder "Zulu"), was gleichbedeutend mit UTC ist, definiert.

6.7.14.3 date-from-datetime

Gibt den *Datums*-Teil des als Argument bereitgestellten `xs:dateTime`-Werts zurück. Der *Uhrzeit*-Teil wird auf Null gesetzt. Die Zeitzone wird nicht geändert.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den zu verarbeitenden `xs:dateTime`-Wert.

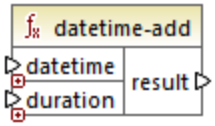
Beispiel

Wenn der Input-Wert `2001-12-17T09:30:02+05:00` ist, ist das Ergebnis der Funktion `2001-12-17+05:00`.

6.7.14.4 datetime-add

Das Ergebnis ist der `xs:dateTime`-Wert, der durch Hinzufügen einer Zeitdauer (duration) (zweites Argument) zu einem Datum und einer Uhrzeit (datetime) (erstes Argument) ermittelt wird.

© 2016-2022 Altova GmbH



Sprachen

Built-in, C++, C#, Java.

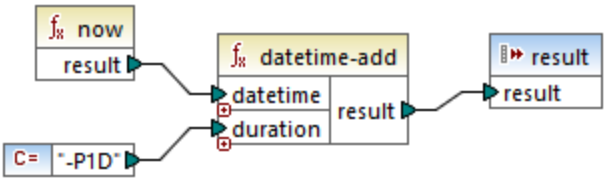
Parameter

Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den als Input zu verwendenden `xs:dateTime`-Wert.
duration	`xs:duration`	Liefert den `xs:duration`-Wert. Ein Beispiel für einen Zeitraum ist **`P1Y2M3DT04H05M59S`**, wobei: • "P" die Periodenangabe ist. Sie ist obligatorisch; • Die restlichen Zeichen stehen der Reihe nach für Folgendes: 1 Jahr, 2 Monate, 3 Tage, (T) Uhrzeit-Designator, 04 Stunden, 05 Minuten, 59 Sekunden. Wenn vor dem Designator "P" ein Minus steht, kennzeichnet dies eine negative Zeitdauer, z.B: **`-P1D`**.

Beispiel

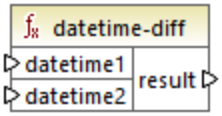
Angenommen, der **datetime**-Input-Wert ist **`2001-12-17T09:30:02+05:00`**. Wenn der Input-Wert **duration** **`P10D`** (10 Tage) ist, ist das Ergebnis der Funktion **`2001-12-27T09:30:02+05:00`**.

Um das gestrige Datum zu erhalten, verbinden Sie die **now**-Funktion mit dem **datetime** Input. Im Mapping unten bedeutet die Periode **`-P1D`** "minus 1 Tag". Daher ist das Ergebnis des Mappings das gestrige Datum.



6.7.14.5 datetime-diff

Das Ergebnis ist die Zeitdauer, die durch Subtraktion von **datetime2** (zweites Argument) von **datetime1** (erstes Argument) ermittelt wird. Das Ergebnis kann auf einen String oder einen Zeitdauer (duration)-Datentyp gemappt werden.



Sprachen

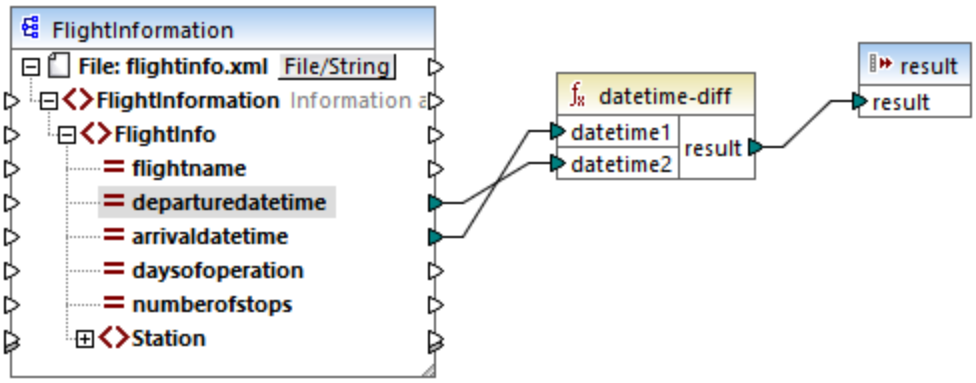
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
datetime1	`xs:dateTime`	Liefert den ersten `xs:dateTime`-Wert.
datetime2	`xs:duration`	Liefert den zweiten `xs:dateTime`-Wert.

Beispiel

Im unten gezeigten Mapping wird mit der Funktion **`datetime-diff`** die Abflugzeit (dateTime) **`2001-12-17T09:30:02+05:00`** von der Ankunftszeit **`2001-12-17T19:30:02+05:00`** subtrahiert. Beachten Sie das der Ankunftswert der größere Wert ist, daher wird er mit dem ersten Input der Funktion verbunden.



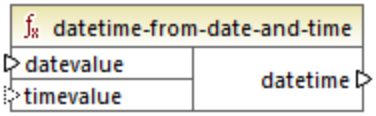
Die Ausgabe des Mappings ist der Unterschied zwischen den zwei Werten (eine Zeitdauer von 10 Stunden):

```
PT10H
```

6.7.14.6 datetime-from-date-and-time

Gibt einen anhand eines `xs:date`-Werts (erstes Argument) und eines `xs:time`-Werts (zweites Argument) erstellten `xs:dateTime` Wert zurück. Das Ergebnis kann auf einen String oder einen `xs.dateTime`-Datentyp gemappt werden.

© 2016-2022 Altova GmbH



Sprachen

Built-in, C++, C#, Java.

Parameter

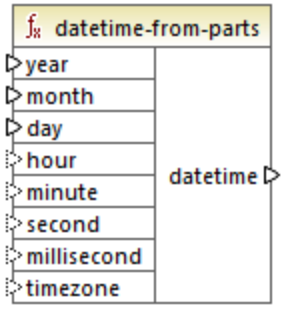
Name	Typ	Beschreibung
datevalue	`xs:date`	Liefert einen Wert vom Typ `xs:date`.
timevalue	`xs:time`	Liefert einen Wert vom Typ `xs:time`.

Beispiel

Wenn das erste Argument **`2012-06-29`** und das zweite Argument **`11:59:55`** ist, gibt die Funktion **`2012-06-29T11:59:55`** zurück.

6.7.14.7 datetime-from-parts

Das Ergebnis ist ein Wert vom Typ `xs:dateTime`, der aus jeder beliebigen Kombination der folgenden als Argumente bereitgestellten Teile zusammengesetzt wird: year, month, day, hour, minute, second, millisecond, timezone. Diese Funktion normalisiert die bereitgestellten Parameter automatisch. So wird etwa der 32. Jänner automatisch in den 1. Februar geändert.



Sprachen

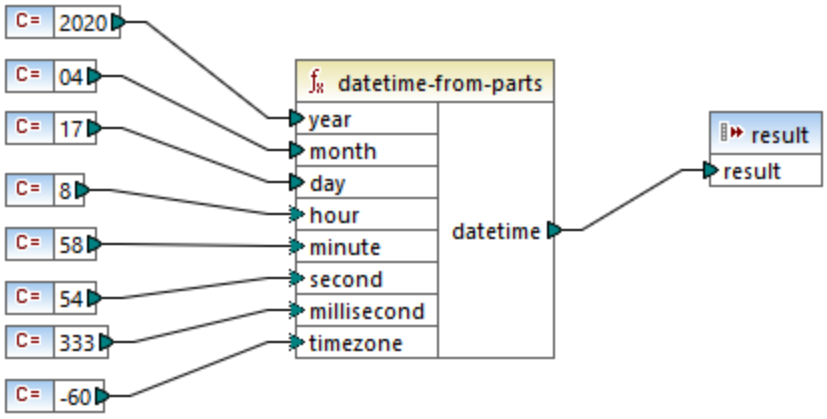
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
year	`xs:int`	Liefert das Jahr.
month	`xs:int`	Liefert den Monat.
day	`xs:int`	Liefert den Tag des Monats.
hour	`xs:int`	Optional. Liefert die Stunde.
minute	`xs:int`	Optional. Liefert die Minute.
second	`xs:int`	Optional. Liefert die Sekunde.
millisecond	`xs:decimal`	Optional. Liefert die Millisekunde.
timezone	`xs:int`	Optional. Liefert die Zeitzone in Minuten. Dieser Wert kann negativ sein.

Beispiel

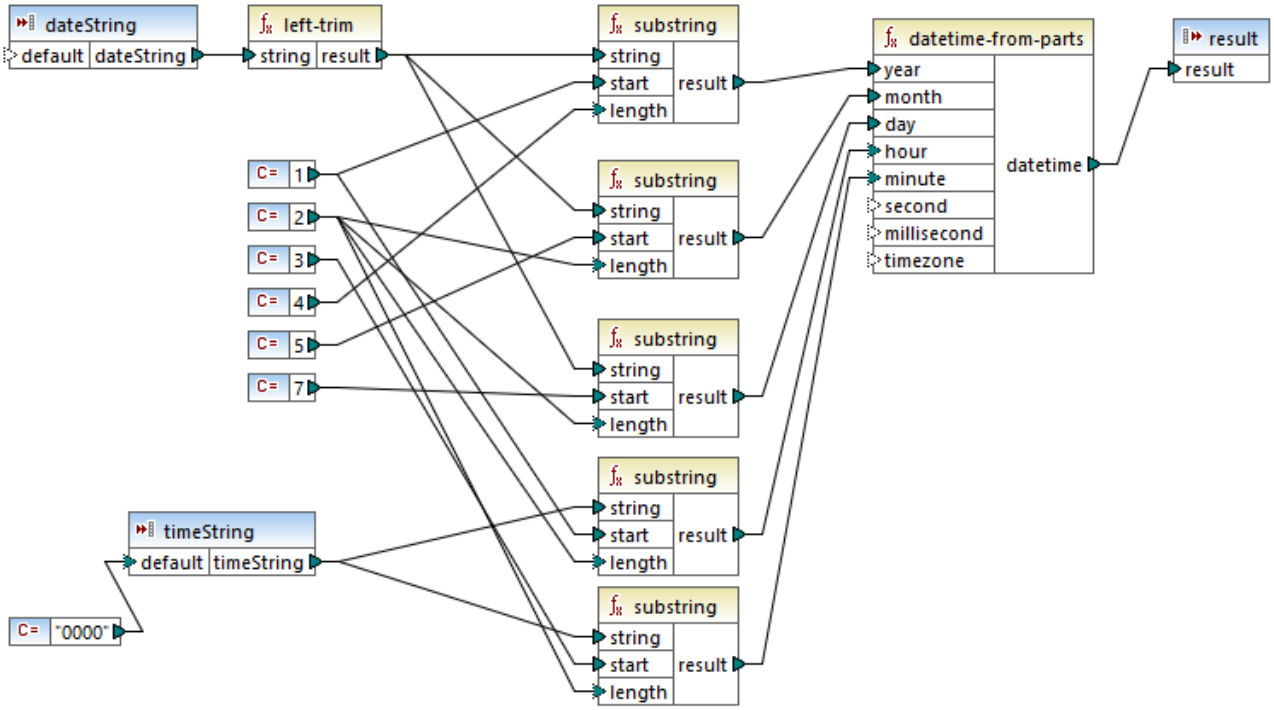
Im folgenden Mapping wird aus durch Konstanten bereitgestellten Teilen ein `xs:dateTime-Wert` zusammengesetzt.



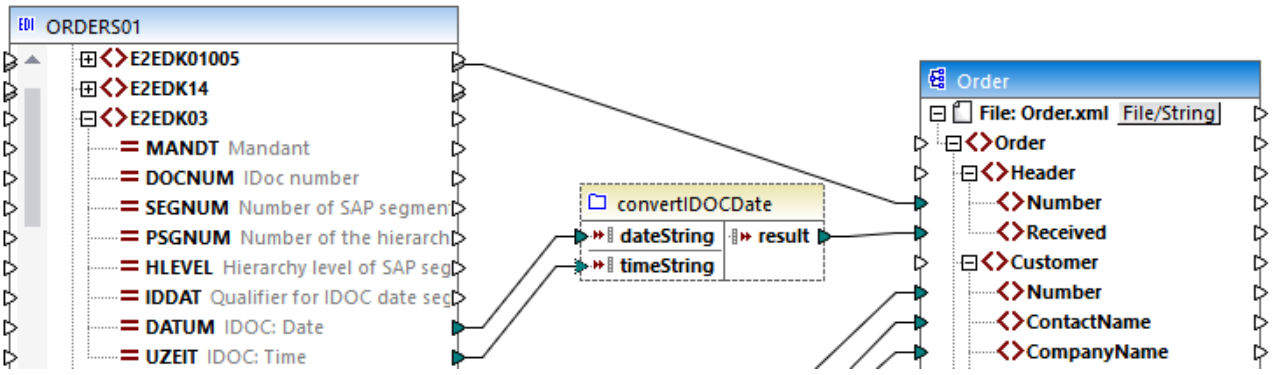
Die Mapping-Ausgabe ist **`2020-04-17T08:58:54.333-01:00`**.

Ein weiteres Beispiel dazu finden Sie im Mapping **IDoc_Order.mfd** im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. In diesem Mapping wird gezeigt, wie Daten aus SAP IDoc[501]-Instanzdateien ausgelesen werden. Es enthält eine benutzerdefinierte Funktion "convertIDOCDate", die den `xs:dateTime`-Wert unter Verwendung der **`left-trim`** und **`substring`**-Funktion aus zwei Input-Strings zusammensetzt.

© 2016-2022 Altova GmbH



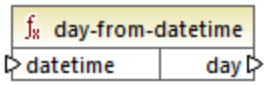
Die **dateString**- und die **timeString**-Komponente sind Parameter für die benutzerdefinierte Funktion. Sie werden dem Hauptmapping aus der SAP IDoc-Instanzdatei bereitgestellt:



In diesem Mapping ist der Datumswert als String `19990621`. Der Uhrzeitwert ist `0930`. Folglich gibt die Funktion `datetime-from-parts` den Wert `1999-06-21T09:30:00` zurück. Dieser Wert ist auch das Ergebnis der benutzerdefinierten Funktion und wird in das Datenelement **Received** der XML-Zielkomponente geschrieben.

6.7.14.8 day-from-datetime

Gibt aus dem als Argument bereitgestellten `xs:dateTime`-Wert den Tag als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

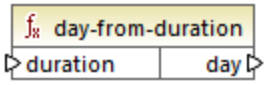
Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

Beispiel

Wenn **datetime** `2001-12-17T10:30:03+01:00` ist, gibt die Funktion `17` zurück.

6.7.14.9 day-from-duration

Gibt aus dem als Argument bereitgestellten `xs:duration`-Wert den Tag als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
duration	`xs:duration`	Liefert den Input-Wert vom Typ `xs:duration`.

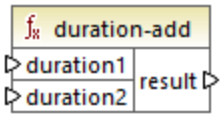
Beispiel

Wenn **duration** `P1Y2M3DT10H30M` ist, so gibt die Funktion `day-from-duration` `3` zurück.

© 2016-2022 Altova GmbH

6.7.14.10 duration-add

Das Ergebnis ist die durch Addition zweier Zeitdauerwerte ermittelte Zeitdauer.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
duration1	`xs:duration`	Liefert den ersten Input-Wert vom Typ `xs:duration`.
duration2	`xs:duration`	Liefert den zweiten Input-Wert vom Typ `xs:duration`.

Beispiel

Wenn die erste Zeitdauer **`P0Y0M3DT03H0M`** (3 Tage und 3 Stunden) ist und die zweite Zeitdauer **`P0Y0M3DT01H0M`** (3 Tage und 1 Stunde), so gibt die Funktion **`P6DT4H`** (6 Tage und 4 Stunden) zurück.

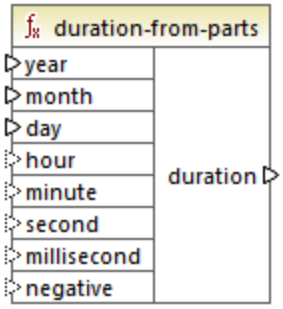
6.7.14.11 duration-from-parts

Das Ergebnis ist ein Wert vom Typ `xs:duration`, der durch Kombination der folgenden Bestandteile, die als Argumente bereitgestellt wurden, berechnet wird: year, month, day, hour, minute, second, millisecond, negative.

Ein Beispiel für einen Zeitraum ist **`P1Y2M3DT04H05M59S`**, wobei:

- "P" die Periodenangabe ist. Sie ist obligatorisch;
- Die restlichen Zeichen stehen der Reihe nach für Folgendes: 1 Jahr, 2 Monate, 3 Tage, (T) Uhrzeit-Designator, 04 Stunden, 05 Minuten, 59 Sekunden.

Wenn vor dem Designator "P" ein Minus steht, kennzeichnet dies eine negative Zeitdauer, z.B: **`-P1D`**.



Sprachen

Built-in, C++, C#, Java.

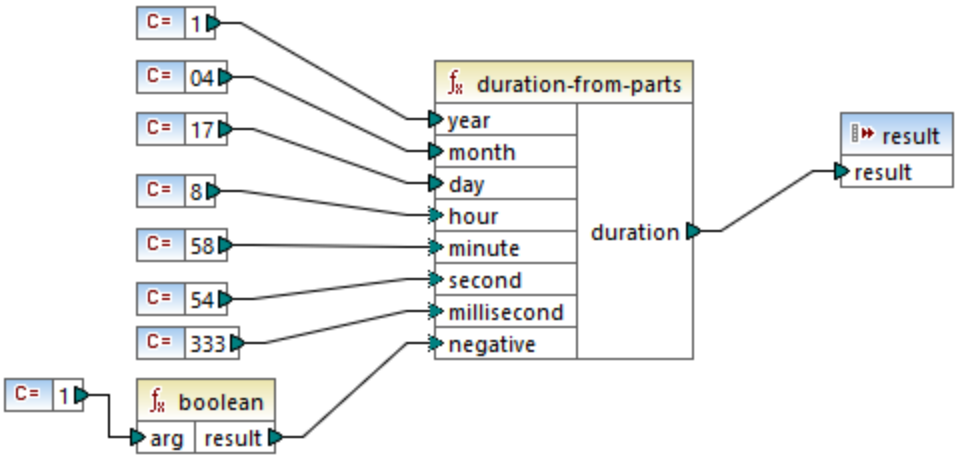
Parameter

Name	Typ	Beschreibung
year	`xs:int`	Liefert das Jahr.
month	`xs:int`	Liefert den Monat.
day	`xs:int`	Liefert den Tag des Monats.
hour	`xs:int`	Optional. Liefert die Stunde.
minute	`xs:int`	Optional. Liefert die Minute.
second	`xs:int`	Optional. Liefert die Sekunde.
millisecond	`xs:decimal`	Optional. Liefert die Millisekunde.
negative	`xs:boolean`	Optional. Muss bei einer negativen Zeitdauer **true** sein; muss andernfalls **false** sein.

Beispiel

Mit dem folgenden Mapping wird eine negative Zeitdauer von 1 Jahr, 4 Monaten, 17 Tagen, 8 Stunden, 58 Minuten und 54,333 Sekunden generiert.

© 2016-2022 Altova GmbH



Die Mapping-Ausgabe ist **P1Y4M17DT8H58M54.333S**.

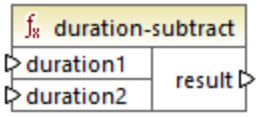
6.7.14.12 duration-subtract

Gibt den durch Subtraktion von **duration2** von **duration1** berechneten Wert `xs:duration` zurück.

Ein Beispiel für einen Zeitraum ist **P1Y2M3DT04H05M59S**, wobei:

- "P" die Periodenangabe ist. Sie ist obligatorisch;
- Die restlichen Zeichen stehen der Reihe nach für Folgendes: 1 Jahr, 2 Monate, 3 Tage, (T) Uhrzeit-Designator, 04 Stunden, 05 Minuten, 59 Sekunden.

Wenn vor dem Designator "P" ein Minus steht, kennzeichnet dies eine negative Zeitdauer, z.B: **-P1D**.



Sprachen

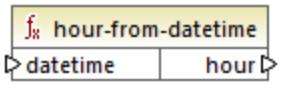
Built-in, C++, C#, Java.

Beispiel

Wenn **duration1** **P0Y0M0DT05H07M** (5 Stunden und 7 Minuten) und **duration2** **PT1H** (1 Stunde) ist, gibt die Funktion **PT4H7M** (4 Stunden und 7 Minuten) zurück.

6.7.14.13 hour-from-datetime

Gibt aus dem als Argument bereitgestellten `xs:dateTime`-Wert die Stunde als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

Beispiel

Wenn **datetime** `2001-12-17T09:30:02+05:00` ist, gibt die Funktion `9` zurück.

6.7.14.14 hour-from-duration

Gibt aus dem als Argument bereitgestellten `xs:duration`-Wert die Stunde als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
duration	`xs:duration`	Liefert den Input-Wert vom Typ `xs:duration`.

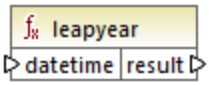
Beispiel

Wenn **duration** `P0Y0M0DT05H07M` ist, gibt die Funktion `5` zurück.

© 2016-2022 Altova GmbH

6.7.14.15 leapyear

Gibt den Booleschen Wert **true** zurück, wenn das Jahr des als Argument bereitgestellten `xs:dateTime`-Werts ein Schaltjahr ist; gibt andernfalls **false** zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

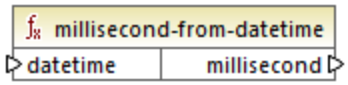
Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

Beispiel

Wenn **datetime** `2020-04-17T09:30:02+02:00` ist, gibt die Funktion **true** zurück, da das Jahr 2020 ein Schaltjahr ist.

6.7.14.16 millisecond-from-datetime

Gibt aus dem als Argument bereitgestellten `xs:dateTime`-Wert die Millisekunden als `xs:decimal`-Wert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

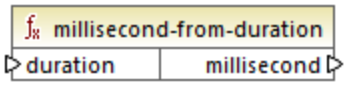
Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

Beispiel

Wenn **datetime** `2001-12-17T09:30:02,544+05:00` ist, gibt die Funktion `544` zurück.

6.7.14.17 millisecond-from-duration

Gibt aus dem als Argument bereitgestellten `xs:duration`-Wert die Millisekunden als `xs:decimal`-Wert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

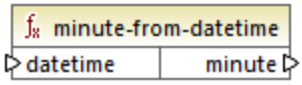
Name	Typ	Beschreibung
duration	`xs:duration`	Liefert den Input-Wert vom Typ `xs:duration`.

Beispiel

Wenn **duration** `P0Y0M0DT05H07M02.227S` ist, gibt die Funktion `227` zurück.

6.7.14.18 minute-from-datetime

Gibt aus dem als Argument bereitgestellten `xs:dateTime`-Wert die Minuten als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

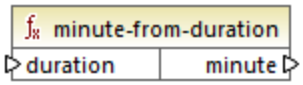
Beispiel

Wenn **datetime** `2001-12-17T09:30:02,544+05:00` ist, gibt die Funktion `30` zurück.

© 2016-2022 Altova GmbH

6.7.14.19 minute-from-duration

Gibt aus dem als Argument bereitgestellten `xs:duration`-Wert die Minuten als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

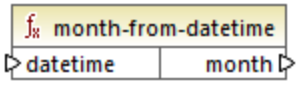
Name	Typ	Beschreibung
duration	`xs:duration`	Liefert den Input-Wert vom Typ `xs:duration`.

Beispiel

Wenn **duration** `P0Y0M0DT05H07M02.227S` ist, gibt die Funktion `7` zurück.

6.7.14.20 month-from-datetime

Gibt aus dem als Argument bereitgestellten `xs:dateTime`-Wert den Monat als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

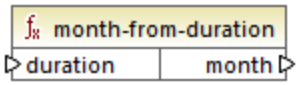
Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

Beispiel

Wenn **datetime** `2001-12-17T09:30:02,544+05:00` ist, gibt die Funktion `12` zurück.

6.7.14.21 month-from-duration

Gibt aus dem als Argument bereitgestellten `xs:duration`-Wert den Monat als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

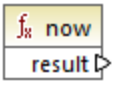
Name	Typ	Beschreibung
duration	`xs:duration`	Liefert den Input-Wert vom Typ `xs:duration`.

Beispiel

Wenn **duration** `P0Y04M0DT05H07M02.227S` ist, gibt die Funktion `4` zurück.

6.7.14.22 now

Gibt das aktuelle Datum und die Uhrzeit (einschließlich Zeitzone) als `xs:dateTime`-Wert zurück.

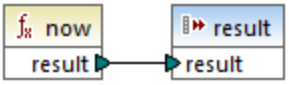


Sprachen

Built-in, C++, C#, Java.

Beispiel

Das folgende Mapping gibt das aktuelle Datum und die aktuelle Uhrzeit zurück. Die Ausgabe ändert sich jedes Mal, wenn das Mapping ausgeführt wird.



Ein Beispiel für eine Mapping-Ausgabve wäre `2020-04-17T11:42:34.684+02:00`.

Wie man das gestrige Datum extrahiert, wird unter der Funktion core | lang | datetime-add[1130] beschrieben.

© 2016-2022 Altova GmbH

6.7.14.23 remove-timezone

Entfernt die Zeitzonenkomponente aus dem Input-Parameter **time** (des Typs `xs:dateTime`).



Sprachen

Built-in, C++, C#, Java.

Parameter

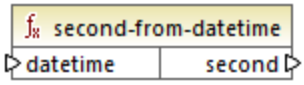
Name	Typ	Beschreibung
time	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

Beispiel

Wenn **time** `2001-12-17T09:30:02+05:00` ist, gibt die Funktion `2001-12-17T09:30:02` zurück.

6.7.14.24 second-from-datetime

Gibt aus dem als Argument bereitgestellten `xs:dateTime`-Wert die Sekunden als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

Beispiel

Wenn **datetime** `2001-12-17T09:30:02,544+05:00` ist, gibt die Funktion `2` zurück.

6.7.14.25 second-from-duration

Gibt aus dem als Argument bereitgestellten `xs:duration`-Wert die Sekunden als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

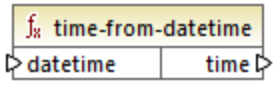
Name	Typ	Beschreibung
duration	`xs:duration`	Liefert den Input-Wert vom Typ `xs:duration`.

Beispiel

Wenn **duration** `P0Y04M0DT05H07M02.227S` ist, gibt die Funktion `2` zurück.

6.7.14.26 time-from-datetime

Gibt aus dem als Argument bereitgestellten `xs:dateTime`-Wert die Uhrzeitkomponente als `xs:time`-Wert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

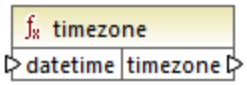
Beispiel

Wenn **datetime** `2001-12-17T09:30:02+05:00` ist, gibt die Funktion `09:30:02+05:00` zurück.

© 2016-2022 Altova GmbH

6.7.14.27 timezone

Gibt den Uhrzeitunterschied des als Argument bereitgestellten `xs:dateTime`-Werts in Minuten zurück. Gibt für UTC 0 zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

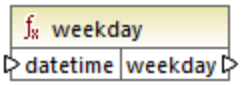
Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

Beispiel

Wenn **datetime** `2001-12-17T09:30:02,544+05:00` ist, gibt die Funktion `300` zurück.

6.7.14.28 weekday

Gibt aus dem als Argument bereitgestellten `xs:dateTime`-Wert den Wochentag zurück. Die Funktion gibt den Wert **1** für Monday, den Wert **2** für Dienstag, usw. zurück.



Sprachen

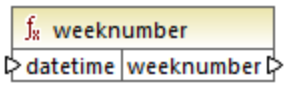
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

6.7.14.29 weeknumber

Gibt aus dem als Argument bereitgestellten `xs:dateTime`-Wert die Nummer der Woche im Jahr zurück. Die Funktion gibt für die erste Woche des Jahrens den Wert **1**, für die zweite Woche im Jahr den Wert **2**, usw. zurück.



Sprachen

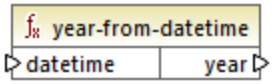
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

6.7.14.30 year-from-datetime

Gibt aus dem als Argument bereitgestellten `xs:dateTime`-Wert das Jahr als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
datetime	`xs:dateTime`	Liefert den Input-Wert vom Typ `xs:dateTime`.

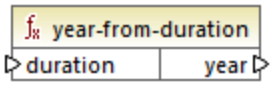
Beispiel

Wenn **datetime** `2001-12-17T09:30:02,544+05:00` ist, gibt die Funktion `2001` zurück.

© 2016-2022 Altova GmbH

6.7.14.31 year-from-duration

Gibt aus dem als Argument bereitgestellten `xs:duration`-Wert das Jahr als Ganzzahlwert zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
duration	`xs:duration`	Liefert den Input-Wert vom Typ `xs:duration`.

Beispiel

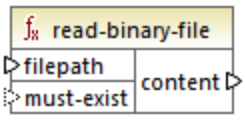
Wenn **duration** `P01Y04M0DT05H07M02.227S` ist, gibt die Funktion `1` zurück.

6.7.15 lang | file functions (Dateifunktionen)

Mit Hilfe der **file**-Funktionen aus der **lang**-Bibliothek können Sie binären Inhalt aus Dateien auslesen oder in Dateien schreiben.

6.7.15.1 read-binary-file

Diese Funktion gibt den Inhalt der angegebenen Datei als BLOB (Binary Large Object) vom Typ `xs:base64Binary` zurück. Zwar handelt es sich beim Datentyp um "base64Binary", doch wird er intern einfach als BLOB dargestellt. Erst wenn Sie das Ergebnis der Funktion auf einen XML-Node vom Typ `xs:base64Binary` mappen, wird er tatsächlich base64-kodiert. Sie könnten das Ergebnis der Funktion auch auf `xs:hexBinary`, einen Datenbank-Blob oder ein Binärdatenfeld in einer Protocol Buffer-Struktur mappen.



Um eine Binärdatei in ein Mapping einzulesen, geben Sie als Input den Pfad zum Argument **filepath** an. Wenn der Dateipfad (**filepath**) relativ ist, sucht MapForce im selben Verzeichnis wie dem Mapping-Verzeichnis nach der Datei. Das Argument **must-exist** ist optional; wenn die Datei nicht geöffnet werden kann und dieser Parameter **true** ist, gibt das Mapping einen Fehler aus. Wenn die Datei nicht geöffnet werden kann und dieser Parameter **false** ist, wird eine leere Binärdatei zurückgegeben.

Sprachen

Built-in.

Parameter

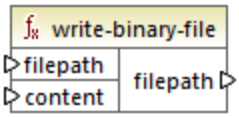
Name	Typ	Beschreibung
filepath	`xs:string`	Der Dateipfad.
must-exist	`xs:boolean`	Optionaler Parameter. Wenn die Datei nicht geöffnet werden kann und dieser Parameter **true** ist, gibt das Mapping einen Fehler aus. Wenn die Datei nicht geöffnet werden kann und dieser Parameter **false** ist, wird eine leere Binärdatei zurückgegeben. Der Standardwert ist **true**.

Beispiel

Siehe Beispiel: Lesen von Binärdateien 862.

6.7.15.2 write-binary-file

Diese Funktion schreibt den Inhalt einer Binärdatei in den angegebenen Dateipfad und gibt den Pfad der geschriebenen Datei zurück. Wenn eine Binärdatei das einzige gewünschte Ergebnis ist, verbinden Sie das Ergebnis der Funktion mit einer einfachen Output 879-Komponente. Da diese Funktion immer, wenn deren Ausgabe im Mapping verwendet wird, eine Datei schreibt, wird empfohlen, das Ergebnis der Funktion direkt mit einer Zielkomponente zu verbinden, ohne weiterer Verarbeitungen dazwischen durchzuführen.



Um Binärdateien in ein Mapping zu schreiben, geben Sie als Input für das Argument **filepath** ihren Pfad an. Wenn der Dateipfad (**filepath**) relativ ist, generiert MapForce die Datei im selben Verzeichnis wie dem Mapping-Verzeichnis. Das Argument **content** muss mit dem eigentlichen Binärdateninhalt (z.B. einem BLOB-Feld in einer Datenbank) verbunden werden.

Wenn Sie in MapForce eine Vorschau auf das Mapping anzeigen, generiert die Funktion standardmäßig temporäre Dateien, anstatt die Dateien direkt auf dem Datenträger zu speichern. Um die temporären Dateien zu speichern, klicken Sie zuerst auf das Register **Ausgabe** und anschließend je nach Bedarf auf eine der Symbolleisten-Schaltflächen **Generierte Ausgabe speichern** oder **Alle generierten Ausgaben speichern**.

© 2016-2022 Altova GmbH

Um MapForce so zu konfigurieren, dass die Ausgabe, anstatt in einer temporären Datei, direkt in einer endgültigen Datei gespeichert wird, wählen Sie den Menübefehl **Extras | Optionen**, klicken Sie auf **Allgemein** und aktivieren Sie die Option **Direkt in die endgültigen Output-Dateien schreiben**. Beachten Sie, dass mit dieser Option alle vorhandenen Dateien desselben Namens überschrieben werden.

Die Funktion gibt immer den endgültigen (und nicht den temporären) Dateinamen zurück, selbst wenn die endgültige Datei noch nicht auf dem Datenträger gespeichert wurde (Dies ist der Fall, wenn Sie eine Vorschau auf das Mapping anzeigen und die Option **Direkt in die endgültige Output-Datei schreiben** deaktiviert ist).

> Beachten Sie, dass dies bei einem Mapping, das seine eigene Ausgabedatei wieder ausliest, nicht unterstützt wird.

Sprachen

Built-in.

Parameter

Name	Typ	Beschreibung
filepath	`xs:string`	Der Pfad der Input-Datei.
content	`xs:base64Binary`	Der binäre Inhalt vom Typ `xs:base64Binary`.

Beispiel

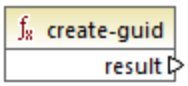
Siehe Beispiel: Schreiben von Binärdateien 864.

6.7.16 lang | generator functions (Generierungsfunktionen)

Die Generierungsfunktionen aus der **lang**-Bibliothek sind Funktionen, die Werte generieren (Derzeit ist `create-guid` die einzige dieser Funktionen).

6.7.16.1 create-guid

Das Ergebnis ist ein global eindeutiger Identifier (globally unique identifier = GUID) als hexadezimal kodierter String. Mit Hilfe dieser Funktion können direkt anhand des Mappings eindeutige Werte für Datenbankfelder oder andere Komponententypen generiert werden. Siehe auch die Funktion format-guid-string 1170.



Sprachen

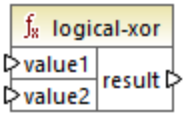
Built-in, C++, C#, Java.

6.7.17 lang | logical functions (logische Funktionen)

Zu den logischen Funktionen aus der **lang**-Bibliothek gehören Funktionen zur Auswertung verschiedener Wertetypen mit Hilfe Boolescher Logik.

6.7.17.1 logical-xor

Gibt **true** zurück, wenn **value1** sich von **value2** unterscheidet; gibt andernfalls **false** zurück.



Sprachen

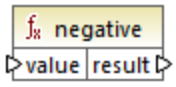
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value1	`xs:boolean`	Der erste Input-Wert.
value2	`xs:boolean`	Der zweite Input-Wert.

6.7.17.2 negative

Gibt **true** zurück, wenn der Input-Wert negativ (kleiner als Null) ist; gibt andernfalls **false** zurück.



Sprachen

Built-in, C++, C#, Java.

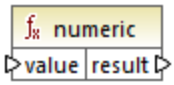
Parameter

Name	Typ	Beschreibung
value	`xs:decimal`	Der Input-Wert.

© 2016-2022 Altova GmbH

6.7.17.3 numeric

Gibt **true** zurück, wenn der Input-Wert eine Zahl oder ein als Zahl parsbarer String ist; gibt andernfalls **false** zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

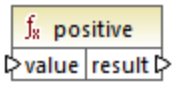
Name	Typ	Beschreibung
value	`xs:decimal`	Der Input-Wert.

Beispiel

Wenn der Input-Wert der String `"4.33"` ist, gibt die Funktion **true** zurück. Wenn der Input-Wert der String `"4.33 USD"` ist, gibt die Funktion **false** zurück.

6.7.17.4 positive

Gibt **true** zurück, wenn der Input-Wert positiv (größer oder gleich Null) ist; gibt andernfalls **false** zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

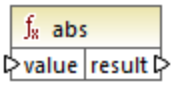
Name	Typ	Beschreibung
value	`xs:decimal`	Der Input-Wert.

6.7.18 lang | math functions (mathematische Funktionen)

Mit Hilfe der math-Funktionen aus der **lang**-Bibliothek können im Mapping verschiedene mathematische Operationen durchgeführt werden.

6.7.18.1 abs

Gibt den absoluten Wert des als Argument bereitgestellten numerischen Werts zurück. Wenn das Argument nicht negativ ist, wird das Argument zurückgegeben. Wenn das Argument negativ ist, wird die Negation des Arguments zurückgegeben.



Sprachen

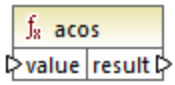
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:decimal`	Der Input-Wert.

6.7.18.2 acos

Gibt den ArkusCosinus von **value** im Bereich von `-pi/2` bis `pi/2` zurück.



Sprachen

Built-in, C++, C#, Java.

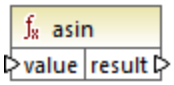
Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

© 2016-2022 Altova GmbH

6.7.18.3 asin

Gibt den ArkusSinus von **value** im Bereich von `-pi/2` bis `pi/2` zurück.



Sprachen

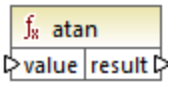
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.18.4 atan

Gibt den ArkusTangens von **value** im Bereich von `-pi/2` bis `pi/2` zurück.



Sprachen

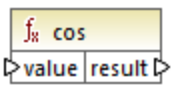
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.18.5 cos

Gibt den trigonometrischen Cosinus des durch **value** angegebenen Winkels zurück. Die Werteinheit ist radians.



Sprachen

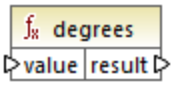
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.18.6 degrees

Konvertiert einen in Radiant gemessenen Winkel in den annähernd entsprechenden Winkel in Grad.



Sprachen

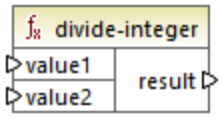
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.18.7 divide-integer

Gibt das ganzzahlige Ergebnis der Division von **value1** durch **value2** zurück.



Sprachen

Built-in, C++, C#, Java.

© 2016-2022 Altova GmbH

Parameter

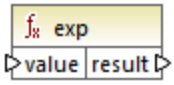
Name	Typ	Beschreibung
value1	`xs:decimal`	Der erste Input-Wert.
value2	`xs:decimal`	Der zweite Input-Wert.

Beispiel

Wenn der erste Wert **15** und der zweite Wert **2** ist, gibt die Funktion **7** zurück.

6.7.18.8 exp

Gibt die Eulersche Zahl **e** hoch **value** zurück.



Sprachen

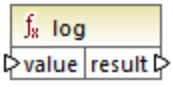
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.18.9 log

Gibt den natürlichen Logarithmus (Basis e) von **value** zurück.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.18.10 log10

Gibt den dekadischen Logarithmus (Basis 10) von **value** zurück.



Sprachen

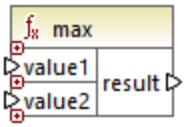
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.18.11 max

Gibt den numerischen Wert des größten als Argument bereitgestellten Werts zurück. Standardmäßig hat diese Funktion nur zwei Parameter, Sie können jedoch weitere hinzufügen. Klicken Sie auf **Parameter hinzufügen** (⊡) oder **Parameter löschen** (⊠), um Parameter hinzuzufügen oder zu löschen.



Sprachen

Built-in, C++, C#, Java.

Parameter

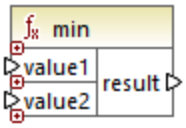
Name	Typ	Beschreibung
value1	`xs:decimal`	Der erste Input-Wert.

© 2016-2022 Altova GmbH

Name	Typ	Beschreibung
value2	`xs:decimal`	Der zweite Input-Wert.
valueN	`xs:decimal`	Der *n*-te Input-Wert.

6.7.18.12 min

Gibt den numerischen Wert des kleinsten als Argument bereitgestellten Werts zurück. Standardmäßig hat diese Funktion nur zwei Parameter, Sie können jedoch weitere hinzufügen. Klicken Sie auf **Parameter hinzufügen** (⊕) oder **Parameter löschen** (⊠), um Parameter hinzuzufügen oder zu löschen.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value1	`xs:decimal`	Der erste Input-Wert.
value2	`xs:decimal`	Der zweite Input-Wert.
valueN	`xs:decimal`	Der *n*-te Input-Wert.

6.7.18.13 pi

Gibt den Wert der mathematischen Konstante *Pi* zurück.

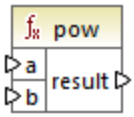


Sprachen

Built-in, C++, C#, Java.

6.7.18.14 pow

Gibt den Wert von **a** hoch **b** zurück.



Sprachen

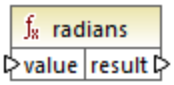
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
a	`xs:double`	Stellt den Wert **a** (die Basis) bereit.
b	`xs:double`	Stellt den Wert **b** (den Exponenten) bereit.

6.7.18.15 radians

Konvertiert einen in Grad gemessenen Winkel in den annähernd entsprechenden Winkel in Radiant.



Sprachen

Built-in, C++, C#, Java.

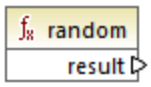
Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.18.16 random

Gibt einen Wert mit einem positiven Vorzeichen, der größer oder gleich 0,0 und kleiner als 1,0 ist, zurück. Die zurückgegebenen Werte werden pseudozufällig mit (annähernd) gleicher Verteilung in diesem Bereich gewählt.

© 2016-2022 Altova GmbH

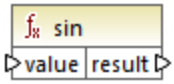


Sprachen

Built-in, C++, C#, Java.

6.7.18.17 sin

Das Ergebnis ist der Sinuswert von **value.**



Sprachen

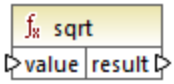
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.18.18 sqrt

Gibt die korrekt gerundete positive Quadratwurzel von **value** zurück.



Sprachen

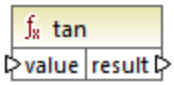
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.18.19 tan

Das Ergebnis ist der Tangens von **value.**



Sprachen

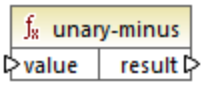
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.18.20 unary-minus

Das Ergebnis ist der Umkehrwert des Input-Werts.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:decimal`	Der Input-Wert.

Beispiel

Wenn der Input-Wert **3** ist, gibt die Funktion **-3** zurück. Wenn der Input-Wert **-3** ist, gibt die Funktion **3** zurück.

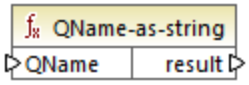
6.7.19 lang | QName functions (QName-Funktionen)

Mit den QName-Funktionen aus der **lang**-Bibliothek können Sie QName (qualifizierter Name)-Werte in Strings konvertieren und umgekehrt. Im Gegensatz zu den Funktionen aus der **core**-Bibliothek stehen diese Funktionen nur in den Sprachen Built-in, Java, C# oder C++ zur Verfügung.

© 2016-2022 Altova GmbH

6.7.19.1 QName-as-string

Gibt die String-Darstellung des als Argument bereitgestellten QName-Werts zurück.



Sprachen

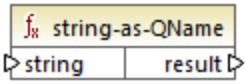
Built-in, C++, C#, Java.

Parameter

Name	Beschreibung
QName	Der Input-`xs:QName`-Wert.

6.7.19.2 string-as-QName

Konvertiert die Stringdarstellung eines QName zurück in einen QName.



Sprachen

Built-in, C++, C#, Java.

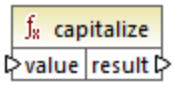
Name	Beschreibung
string	Der Input-`string`-Wert.

6.7.20 lang | string functions (String-Funktionen)

Mit Hilfe der String-Funktionen aus der **lang**-Bibliothek können Sie Strings verarbeiten (z.B. kürzen, mit Füllzeichen auffüllen, ersetzen, Strings in Groß- oder Kleinbuchstaben konvertieren, usw.).

6.7.20.1 capitalize

Das Ergebnis ist der Input-String **value,** wobei der erste Buchstabe jedes Worts groß geschrieben wird.



Sprachen

Built-in, C++, C#, Java.

Parameter

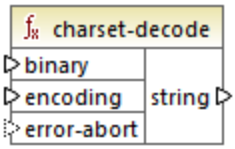
Name	Typ	Beschreibung
value	`xs:string`	Der Input-Wert.

Beispiel

Wenn der Input String `the quick brown fox` ist, gibt die Funktion `The Quick Brown Fox` zurück.

6.7.20.2 charset-decode

Die Funktion `charset-decode` erhält als Input als Base64-Text kodierte Binärdaten. Die Funktion dekodiert Daten gemäß dem definierten Zeichensatz (z.B. "utf-8") und gibt den erzeugten String-Wert zurück. Wenn Sie Binärdaten als Base64-Text kodieren müssen, verwenden Sie die Funktion charset-encode 1166.



Sprachen

Built-in.

Parameter

Name	Typ	Beschreibung
binary-data	`xs:base64Binary`	Die Binärdaten als Base64-Text.
encoding	`xs:string`	Der für die Kodierung verwendete Zeichensatz (z.B. "utf-8").

© 2016-2022 Altova GmbH

Name	Typ	Beschreibung
error-abort	`xs:boolean`	Optionales Argument, das angibt, wie die Verarbeitung fortgesetzt werden soll, wenn Fehler auftreten. Gültige Werte: • **true** - Verarbeitung mit einer Ausnahme am ungültigen Zeichen beenden. • **false** - Verarbeitung fortsetzen und ungültige Zeichen durch das Ersatzzeichen ⍰ ersetzen. Der Standardwert ist **true**.

Beispiel

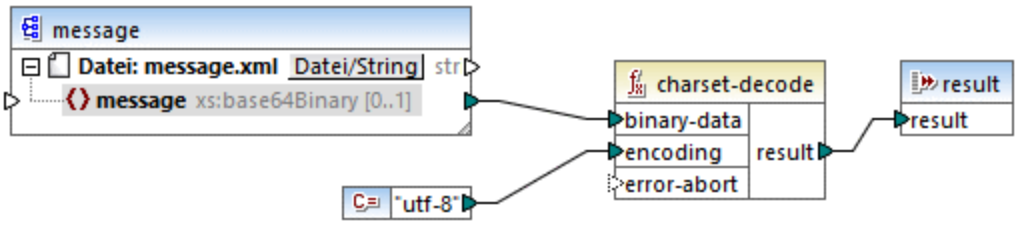
Angenommen, Sie möchten Binärdaten aus der folgenden XML-Quelldatei dekodieren. Beachten Sie, dass das Element **message** als Base64-Text kodierte Binärdaten enthält.

```
<?xml version="1.0" encoding="UTF-8"?>
<message xsi:noNamespaceSchemaLocation="message.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">TG9yZW0gaXBzdW0=</message>
```

Der Datentyp des Elements **message** ist `xs:base64Binary`, wie Sie im Schema sehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="message" type="xs:base64Binary"/>
</xs:schema>
```

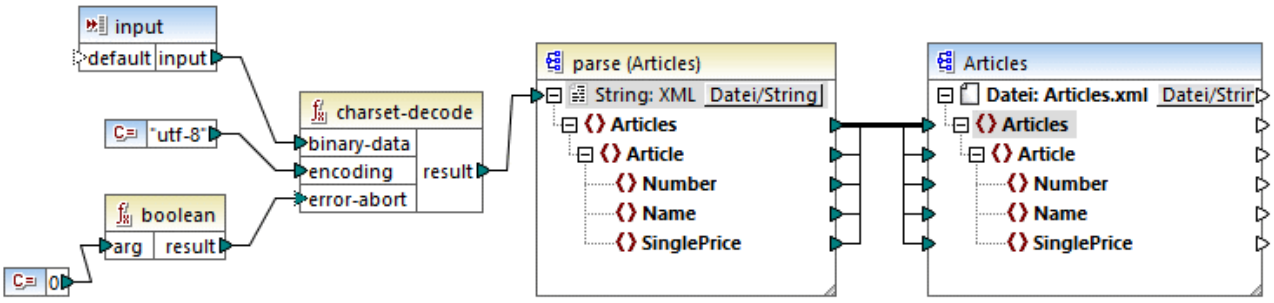
Im Folgenden sehen Sie ein Mapping, in dem die obige Meldung dekodiert wird:



Die Ausgabe des Mappings in diesem Beispiel ist der Text "Lorem ipsum".

Ein Mapping kann als Base64-Daten kodierten Text oder XML-Dateien auch mit Hilfe der MapForce-Serialisierungskomponente verarbeiten. So hat das unten gezeigte Mapping etwa einen **Input**-Parameter, der

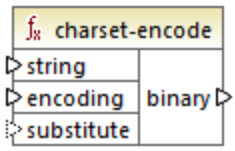
Base64-Textdaten erwartet. Wenn die Base64-Daten anhand einer im Beispiel charset-encode[1166] gezeigten XML-Datei erstellt wurden, können Sie mit dem unten gezeigten Mapping die XML-Originaldatei wieder erzeugen:



Das Argument **error-abort** erhält in diesem Mapping den Wert **false**. Dieser Wert wurde mit Hilfe der vordefinierten **boolean** Funktion erzeugt. Damit wird sichergestellt, dass die Verarbeitung auch bei ungültigen Zeichen fortgesetzt wird. Das String-Ergebnis der Funktion wird anschließend an eine XML-Parsing-Komponente übergeben, die es in eine XML-Datei konvertiert. Damit XML geparst werden kann, benötigen Sie eine XSD-Schema-Datei. Nähere Informationen dazu finden Sie unter Parsen und Serialisieren von Strings[1382]

6.7.20.3 charset-encode

Die Funktion **charset-encode** erhält als Input String-Daten und kodiert dieses als Base64-Text. Die Daten sind im definierten Zecihensatz (z.B. "utf-8") kodiert und werden als `xs:base64Binary`-Typ zurückgegeben. Wenn Sie zuvor als Base64-Text kodierte Binärdaten dekodieren müssen, verwenden Sie die Funktion charset-decode[1164].



Sprachen

Built-in.

Parameter

Name	Typ	Beschreibung
string-data	**xs:string**	Die zu kodierenden Stringdaten.
encoding	**xs:string**	Der für die Kodierung verwendete Zeichensatz (z.B. "utf-8").
substitute	**xs:string**	Optionales Argument, das ein Ersatzzeichen für ungültige Zeichen definiert. Dieses

© 2016-2022 Altova GmbH

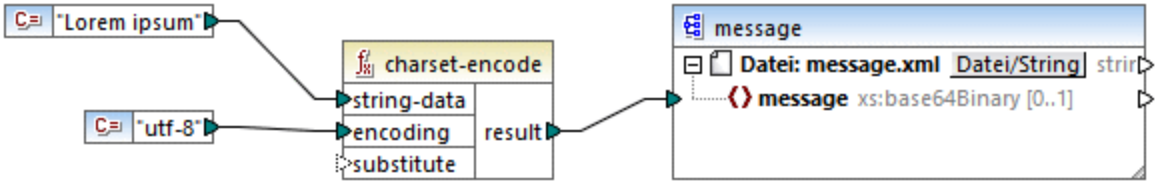
Name	Typ	Beschreibung
		Argument ist anwendbar, wenn Sie eine Nicht-Unicode-Kodierung verwenden. Das Ersatzzeichen für Unicode-Kodierungen ist .

Beispiel

Angenommen, Sie möchten den Text "Lorem ipsum" mit dem UTF-8-Zeichensatz als Base64-Daten kodieren und in eine XML-Zieldatei schreiben. Die XML-Zieldatei hat, wie Sie im Schema unten sehen, ein **message**-Element des Typs `xs:base64Binary`:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="message" type="xs:base64Binary"/>
</xs:schema>
```

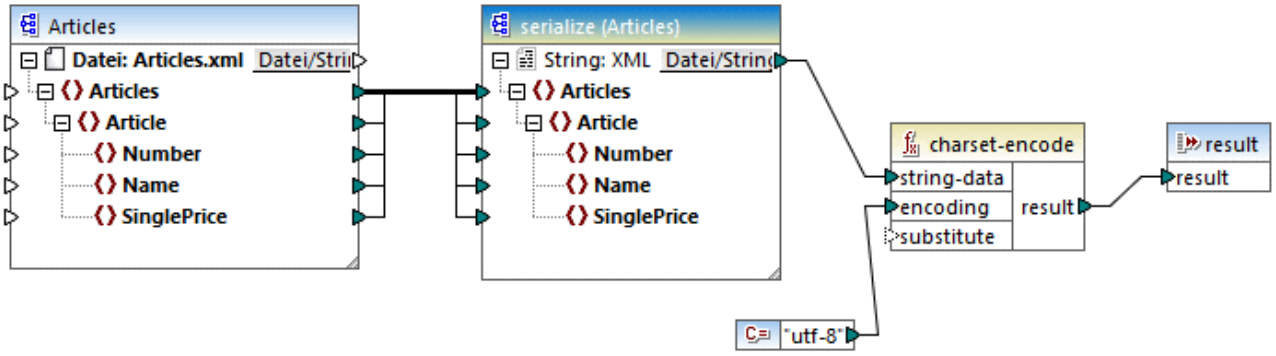
Ein Mapping, mit dem Sie die Base64-Kodierung durchführen, würde folgendermaßen aussehen:



Mit diesem Mapping wird eine XML-Ausgabe wie die unten gezeigte erzeugt (die Schemareferenzen und XML-Deklaration wurden übersprungen):

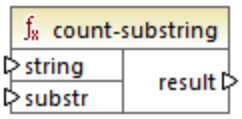
```
<message>TG9yZW0gaXBzdW0=</message>
```

Sie können Text oder XML-Dateien auch mit Hilfe der MapForce-Serialisierungskomponente im Base64-Format kodieren. So wird im unten gezeigten Mapping z.B. eine XML-Quelldatei in einen String serialisiert. Der Ergebnisstring wird anschließend als Argument für die Funktion **charset-encode** bereitgestellt. Das Funktionsergebnis wird schließlich mit Hilfe einer einfachen Ausgabekomponente als Mapping-Ausgabe zurückgegeben, siehe Rückgabe von String-Werten aus einem Mapping 878. Nähere Informationen zur Serialisierung finden Sie unter Parsen und Serialisieren von Strings 1382.



6.7.20.4 count-substring

Das Ergebnis ist ein Ganzzahlwert, der angibt, wie oft **substr** in **string** vorkommt.



Sprachen

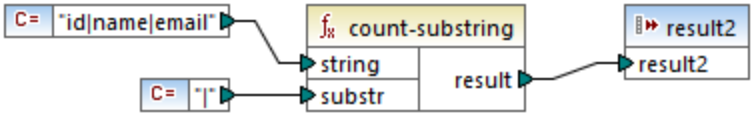
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String.
substr	`xs:string`	Der Substring, auf den der Wert überprüft werden soll.

Beispiel

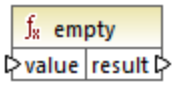
Das folgende Mapping gibt **2** zurück. Dies ist die Anzahl der Instanzen, die das Pipe-Trennzeichen im Input-String `id|name|email` vorkommt.



© 2016-2022 Altova GmbH

6.7.20.5 empty

Gibt **true** zurück, wenn der Input-String-Wert leer ist und **false**, wenn dies nicht der Fall ist.



Sprachen

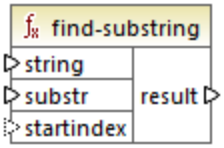
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
value	`xs:string`	Der Input-Wert.

6.7.20.6 find-substring

Gibt die Position der ersten Instanz von **substr** innerhalb von **string** an. Standardmäßig wird die Suche beim ersten Zeichen, das die Position (den Index) 1 hat, begonnen, doch haben Sie die Möglichkeit einen bestimmten Startindex zu definieren. Wenn **substr** nicht gefunden wird, gibt die Funktion **0** zurück.



Sprachen

Built-in, C++, C#, Java.

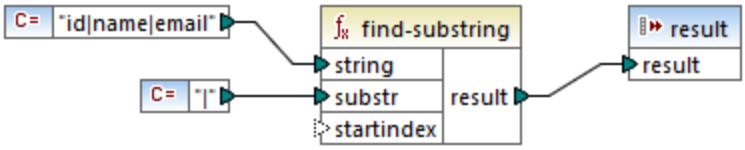
Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String.
substr	`xs:string`	Der Substring, nach dem gesucht werden soll.
startindex	`xs:int`	Optional. Definiert die Anfangsposition (den Index) für die Suche. Wenn dieser Parameter

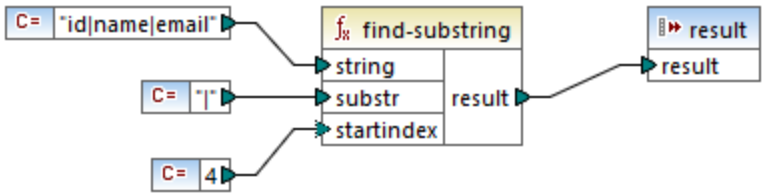
Name	Typ	Beschreibung
		nicht definiert wird, beginnt die Suche an der Position 1.

Beispiel

Das Ergebnis des folgenden Mappings ist **3**. Dies ist die Position der ersten Instanz des Pipe-Zeichens im Input-String `id|name|email`.

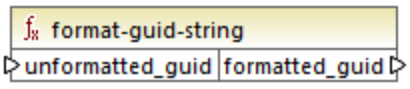


Wenn Sie als Startindex **4** definieren, beginnt die Funktion ab dem vierten Zeichen mit der Suche. Daher ist die Ausgabe des unten gezeigten Mappings **8**. Dies ist bei der Suche ab dem vierten Zeichen die erste Instanz des Pipe-Zeichens.



6.7.20.7 format-guid-string

Gibt einen korrekt formatierten GUID (Globally Unique Identifier)-String zurück, welcher normalerweise in Datenbankfeldern verwendet wird. Siehe auch die Funktion create-guid[1151].



Sprachen

Built-in, C++, C#, Java.

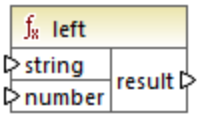
Parameter

Name	Typ	Beschreibung
unformatted_guid	`xs:string`	Der hexadezimal kodierte Input-String, der formatiert werden soll.

© 2016-2022 Altova GmbH

6.7.20.8 left

Das Ergebnis ist ein String, der die ersten n **number** Zeichen des Input-String enthält.



Sprachen

Built-in, C++, C#, Java.

Parameter

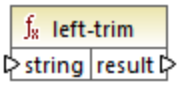
Name	Typ	Beschreibung
string	`xs:string`	Der Input-String.
number	`xs:int`	Definiert, wie viele Zeichen ab dem Beginn des String zurückgegeben werden sollen.

Beispiel

Wenn der Input-String `This is a sentence` lautet und "number" **4** ist, gibt die Funktion `This` zurück.

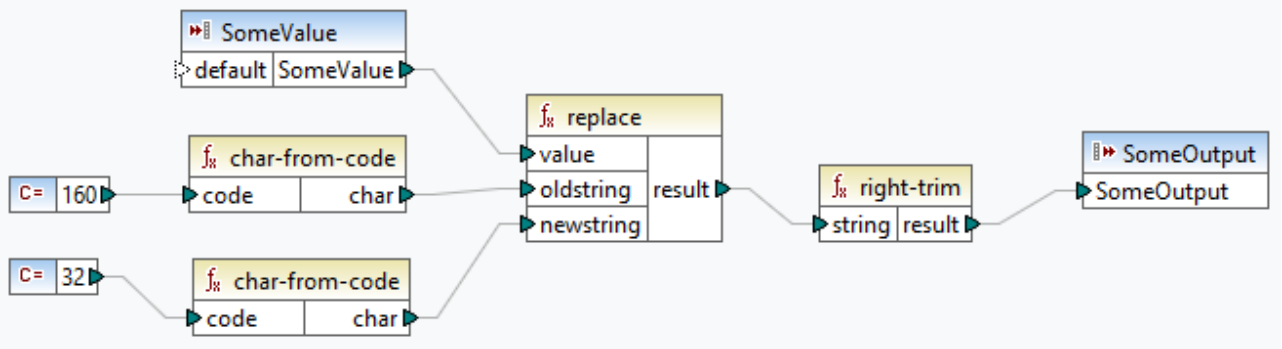
6.7.20.9 left-trim

Die Funktion `left-trim` (*siehe Abbildung unten*) entfernt vorangestellte Leerzeichen aus einem String. Zu den Whitespace-Zeichen zählen das Leerzeichen (U+0020), der Tabulator (U+0009), der Wagenrücklauf (U+000D) und der Zeilenvorschub (U+000A). Nähere Informationen zu Whitespaces finden Sie in der XML Recommendation.



Informationen zu geschützten Leerzeichen

Die Funktionen `left-trim`, `right-trim` und `normalize-space` entfernen geschützte Leerzeichen nicht. Eine mögliche Lösung wäre, ein geschütztes Leerzeichen, dessen Dezimaldarstellung 160 ist, durch ein Leerzeichen mit der Dezimaldarstellung 32 zu ersetzen. Im nächsten Schritt wird der gekürzte Wert `Wert` auf das Zieldatenelement gemappt (*siehe Mapping unten*).



Wenn es sich bei Ihrer Quellkomponente um eine Excel-Datei handelt, können Sie überschüssige Leerzeichen in Excel mit Hilfe einer Kombination aus den Funktionen TRIM, CLEAN und SUBSTITUTE entfernen. Nähere Informationen dazu finden Sie unter *Entfernen von Leerzeichen und nicht druckbaren Zeichen aus dem Text*.

Sprachen

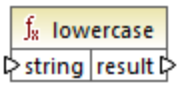
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
String	`xs:string`	Der Input-String.

6.7.20.10 lowercase

Konvertiert den Input **string** in Kleinbuchstaben. Bei Unicode-Zeichen werden die entsprechenden (vom Unicode-Konsortium definierten) Kleinbuchstaben verwendet.



Sprachen

Built-in, C++, C#, Java.

Parameter

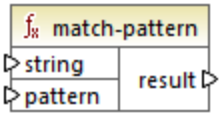
Name	Typ	Beschreibung
string	`xs:string`	Der Input-String.

© 2016-2022 Altova GmbH

6.7.20.11 match-pattern

Gibt den Booleschen Wert **true** zurück, wenn der Input-String mit der durch **pattern** definierten Regular Expression übereinstimmt; gibt andernfalls **false** zurück. Siehe auch Regular Expressions[1020].

Anmerkung: Die komplexen Funktionalitäten der Regular Expression-Syntax können bei der Generierung von C++-, C#- oder Java-Code etwas unterschiedlich sein. Nähere Informationen dazu finden Sie in der regex-Dokumentation zu den einzelnen Sprachen.



Sprachen

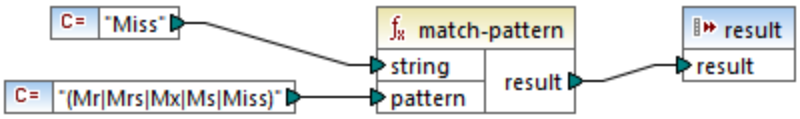
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String.
pattern	`xs:string`	Die Regular Expression, mit der der String übereinstimmen muss.

Beispiel

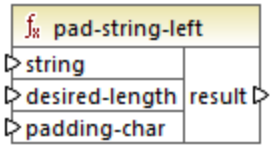
Mit dem folgenden Mapping werden verschiedene Anredeformen validiert. Dabei gibt das Mapping **true** für eine der folgenden Anreden zurück: Mr, Mrs, Mx, Ms, Miss.



Wenn der Input-String nicht mit den oben aufgelisteten Anredeformen übereinstimmt, gibt das Mapping **false** aus.

6.7.20.12 pad-string-left

Gibt einen String zurück, der links als Füllzeichen ein einziges Zeichen vorangestellt erhält, bis die erforderliche Länge erreicht ist. Die gewünschte String-Länge und das Füllzeichen werden als Argumente bereitgestellt.



Sprachen

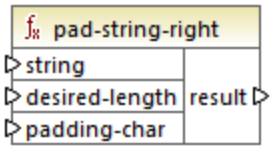
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Definiert den Input-String.
desired-length	`xs:int`	Definiert die gewünschte Länge des String nach dem Auffüllen mit dem Füllzeichen.
padding-char	`xs:string`	Definiert, welches Zeichen als Füllzeichen verwendet werden soll.

6.7.20.13 pad-string-right

Gibt einen String zurück, der rechts als Füllzeichen ein einziges Zeichen nachgestellt erhält, bis die erforderliche Länge erreicht ist. Die gewünschte String-Länge und das Füllzeichen werden als Argumente bereitgestellt.



Sprachen

Built-in, C++, C#, Java.

Parameter

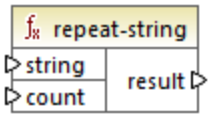
Name	Typ	Beschreibung
string	`xs:string`	Definiert den Input-String.
desired-length	`xs:int`	Definiert die gewünschte Länge des String nach dem Auffüllen mit

© 2016-2022 Altova GmbH

Name	Typ	Beschreibung
		dem Füllzeichen.
padding-char	`xs:string`	Definiert, welches Zeichen als Füllzeichen verwendet werden soll.

6.7.20.14 repeat-string

Wiederholt den als Argument bereitgestellten String *n* Mal. Das Argument **count** definiert, wie oft der String wiederholt werden soll.



Sprachen

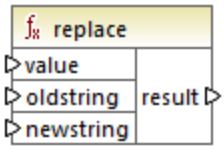
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String.
count	`xs:int`	Definiert, wie oft der String wiederholt werden soll.

6.7.20.15 replace

Das Ergebnis ist ein neuer String, wobei jede Instanz von **oldstring** im Input String **value** durch **newstring** ersetzt wird.



Sprachen

Built-in, C++, C#, Java.

Parameter

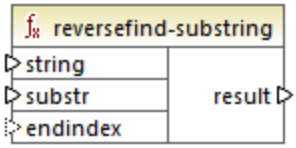
Name	Typ	Beschreibung
value	`xs:string`	Der Input-Wert.
oldstring	`xs:string`	Der alte, zu ersetzende String.
newstring	`xs:string`	Der neue String, der als Ersetzungsstring fungieren soll.

Beispiel

Siehe Ersetzen von Sonderzeichen 283.

6.7.20.16 reversefind-substring

Gibt die Position der letzten Instanz von **substr** innerhalb von **string** an. Standardmäßig wird die Suche beim ersten Zeichen, das die Position (den Index) 1 hat, begonnen und endet beim letzten Zeichen, doch haben Sie die Möglichkeit einen bestimmten Endindex zu definieren. Wenn **substr** nicht gefunden wird, gibt die Funktion **0** zurück.



Sprachen

Built-in, C++, C#, Java.

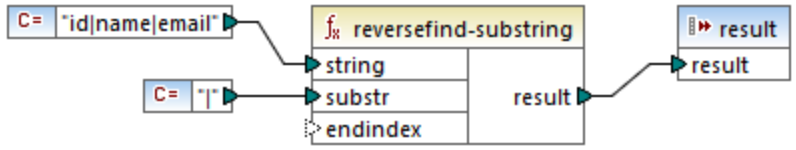
Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String.
substr	`xs:string`	Der Substring, nach dem gesucht werden soll.
endindex	`xs:int`	Optional. Definiert die Endposition (den Index) für die Suche. Wenn dieser Parameter nicht definiert wird, endet die Suche beim letzten Zeichen in **string**.

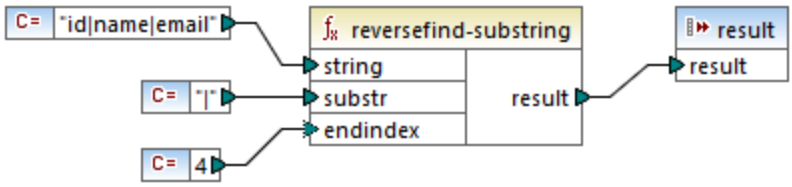
© 2016-2022 Altova GmbH

Beispiel

Das Ergebnis des folgenden Mappings ist **8**. Dies ist die Position der letzten Instanz des Pipe-Zeichens im Input-String `id|name|email`.

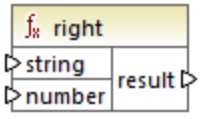


Wenn Sie als Endindex **4** definieren, endet die Suche beim vierten Zeichen. Das Ergebnis des unten gezeigten Mappings ist folglich **3**.



6.7.20.17 right

Das Ergebnis ist ein String, der die letzten n **number** Zeichen des Input-String enthält.



Sprachen

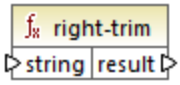
Built-in, C++, C#, Java.

Beispiel

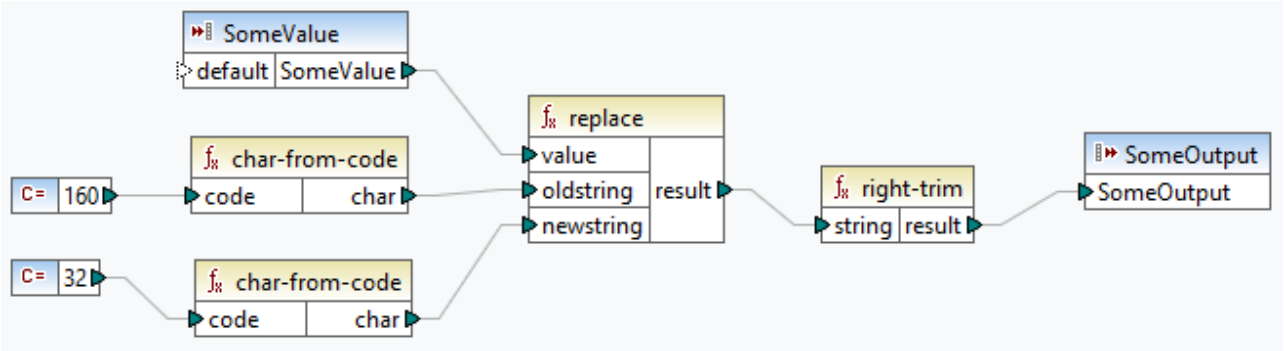
Wenn der Input-String `The brown red fox` lautet und "number" **3** ist, gibt die Funktion `fox` zurück.

6.7.20.18 right-trim

Die Funktion`right-trim` (*siehe Abbildung unten*) entfernt nachgestellte Leerzeichen aus einem String. Zu den Whitespace-Zeichen zählen das Leerzeichen (U+0020), der Tabulator (U+0009), der Wagenrücklauf (U+000D) und der Zeilenvorschub (U+000A). Nähere Informationen zu Whitespaces finden Sie in der XML Recommendation.



Informationen zu geschützten Leerzeichen
Die Funktionen **`left-trim`**, **`right-trim`** und **`normalize-space`** entfernen geschützte Leerzeichen nicht. Eine mögliche Lösung wäre, ein geschütztes Leerzeichen, dessen Dezimaldarstellung 160 ist, durch ein Leerzeichen mit der Dezimaldarstellung 32 zu ersetzen. Im nächsten Schritt wird der gekürzte Wert `Wert` auf das Zieldatenelement gemappt (*siehe Mapping unten*).



Wenn es sich bei Ihrer Quellkomponente um eine Excel-Datei handelt, können Sie überschüssige Leerzeichen in Excel mit Hilfe einer Kombination aus den Funktionen TRIM, CLEAN und SUBSTITUTE entfernen. Nähere Informationen dazu finden Sie unter *Entfernen von Leerzeichen und nicht druckbaren Zeichen aus dem Text*.

Sprachen

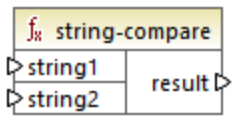
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
String	**`xs:string`**	Der Input-String.

6.7.20.19 string-compare

Die Funktion `string-compare` (*siehe Abbildung unten*) gibt das Ergebnis eines Zeichen-für-Zeichen-Vergleichs zweier Input-Strings `string1` und `string2` zurück. Der Vergleich basiert auf ASCII-Codes. Sowohl `string1` als auch `string2` haben den Typ `xs:string`. Die Groß- und Kleinschreibung wird von der Funktion berücksichtigt. Wenn die Strings gleich sind, ist das Ergebnis 0. Wenn `string1` kleiner als `string2` ist, ist das Ergebnis -1. Wenn `string1` größer als `string2` ist, ist das Ergebnis 1.



 © 2016-2022 Altova GmbH

Beispiel:

```
string1: hi
string2: Hit
```

Die Funktion `string-compare` vergleicht die Strings Zeichen für Zeichen. Der Vergleich wird beendet, nachdem die Funktion erkannt hat, dass sich das erste Zeichen von `string1` und das erste Zeichen von `string2` voneinander unterscheiden. Das Ergebnis basiert auf dem Vergleich der jeweils ersten Zeichen der einzelnen Strings. Da `h` eine höhere ACII-Codenummer (104 im Dezimalsystem) als `H` (72 im Dezimalsystem) hat, ist `string1` größer als `string2` und das Ergebnis des String-Vergleichs ist somit 1. Wären das erste Zeichen von `string1` und das erste Zeichen von `string2` gleich, würde die Funktion mit dem Vergleich fortfahren und das zweite Zeichen, usw. analysieren.

Informationen zum einfachen String-Vergleich mit einem Booleschen Ergebnis finden Sie unter `core | logical functions | equal`[1057].

Sprachen

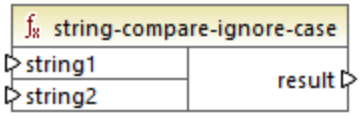
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
`string1`	`xs:string`	Der erste Input-String.
`string2`	`xs:string`	Der zweite Input-String.

6.7.20.20 string-compare-ignore-case

Die Funktion `string-compare-ignore-case` (*siehe Abbildung unten*) gibt das Ergebnis eines Zeichen-für-Zeichen-Vergleichs zweier Input-Strings `string1` und `string2` zurück. Sowohl `string1` als auch `string2` hat den Typ `xs:string`. Die Groß- und Kleinschreibung wird von der Funktion nicht berücksichtigt. Der Vergleich basiert auf ASCII-Codes. Wenn die Strings gleich sind, ist das Ergebnis 0. Wenn `string1` kleiner als `string2` ist, ist das Ergebnis -1. Wenn `string1` größer als `string2` ist, ist das Ergebnis 1.



Beispiel:

```
string1: hi
string2: Hit
```

Die Funktion `string-compare-ignore-case` vergleicht die Strings Zeichen für Zeichen. Obwohl `h` als in Form einer höheren ASCII-Codenummer als `H` dargestellt wird, werden diese beiden Zeichen in dieser Funktion als identisch behandelt. Das zweite Zeichen in beiden Strings ist identisch. `string2` hat jedoch ein drittes Zeichen, während `string1` keines hat. Das dritte Zeichen in `string1` hat einen leeren Wert. Der Wert `t` in

`string2` ist größer als der leere Wert in `string1`. Daher ist `string1` kleiner als `string2`, wodurch das Ergebnis -1 ist.

Sprachen

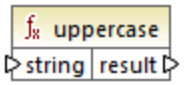
Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
`string1`	`xs:string`	Der erste Input-String.
`string2`	`xs:string`	Der zweite Input-String.

6.7.20.21 uppercase

Konvertiert den Input **string** in Großbuchstaben. Bei Unicode-Zeichen werden die entsprechenden (vom Unicode-Konsortium definierten) Großbuchstaben verwendet.



Sprachen

Built-in, C++, C#, Java.

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String.

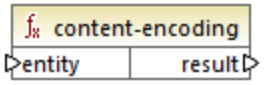
6.7.21 mime

Die **mime**-Bibliothek enthält Funktionen, mit denen MapForce mit MIME (Multipurpose Internet Mail Extensions)-Inhalt interagieren kann. Normalerweise benötigen Sie **mime**-Funktionen zum Erstellen von Webservice-Aufrufen, die benutzerdefinierten MIME-Inhalt (wie z.B. Binärdaten) senden. Außderdem können Sie mit **mime**-Funktionen die von einem Webservice-Aufruf retournierte MIME Response Entity behandeln.

6.7.21.1 content-encoding

Die **`content-encoding`**-Funktion gibt die Zeichenkodierung der als Argument bereitgestellten MIME Entity zurück. Im Fall der HTTP-Nachricht ist dies der Wert des Parameters **charset** aus dem **Content-Type** Header.

© 2016-2022 Altova GmbH



Sprachen

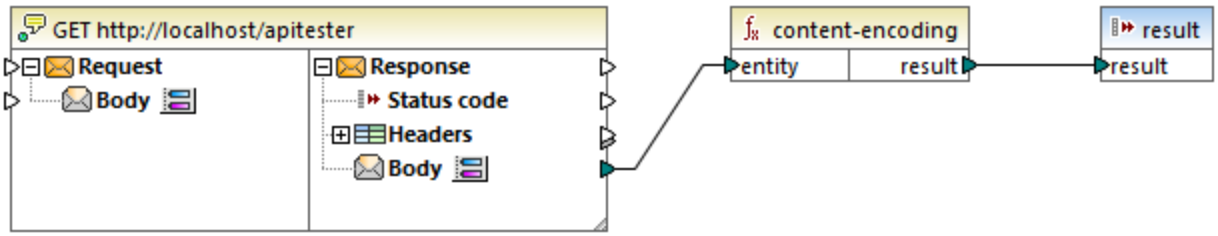
Built-in.

Parameter

Name	Typ	Beschreibung
entity	`mf:MimeEntity`	Der Inhalt der MIME Entity.

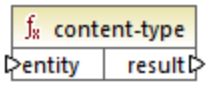
Beispiel

Wenn in einem Mapping ein Webservice aufgerufen wird, der einen **Content-Type** Header als `text/json; charset=utf-8` zurückgibt, so ist die Ausgabe der Funkion `utf-8`.



6.7.21.2 content-type

Die Funktion `content-type` gibt den Medientyp (**Content-Type**) der als Argument bereitgestellten MIME Entity zurück.



Sprachen

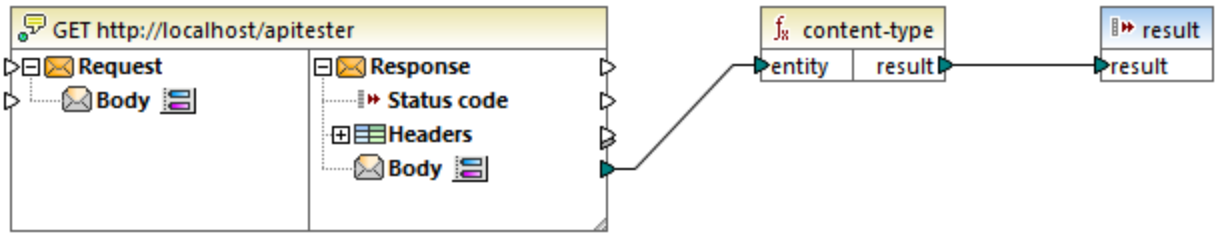
Built-in.

Parameter

Name	Typ	Beschreibung
entity	`mf:MimeEntity`	Der Inhalt der MIME Entity.

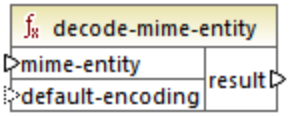
Beispiel

Wenn in einem Mapping ein Webservice aufgerufen wird, der einen **Content-Type** Header als `text/json; charset=utf-8` zurückgibt, so ist die Ausgabe der Funktion ebenfalls `text/json; charset=utf-8`.



6.7.21.3 decode-mime-entity

Die Funktion `decode-mime-entity` gibt einen String zurück, der die mit Hilfe des im **Content-Type**-Header der Entity enthaltenen Parameters **charset** dekodierte MIME Entity darstellt. Wenn kein **charset** definiert wurde, können Sie durch Verbinden des gewünschten String-Werts mit dem Parameter **default-encoding** eine Standardkodierung angeben.



Sprachen

Built-in.

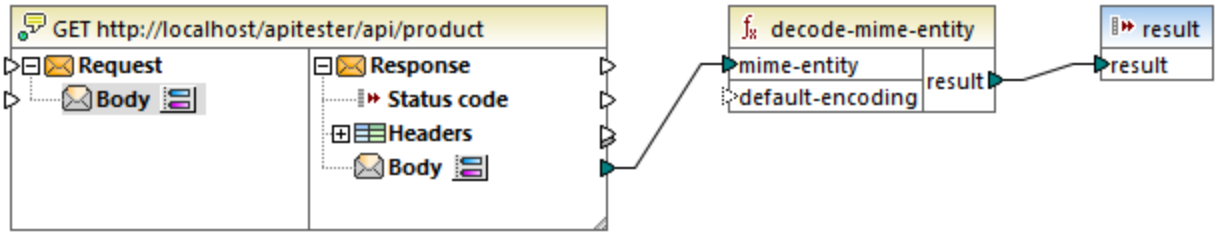
Argumente

Name	Typ	Beschreibung
mime-entity	`mf:MimeEntity`	Die zu dekodierende MIME Entity.
default-encoding	`xs:string`	Die Standardkodierung, die verwendet werden soll, wenn im Parameter **charset** keine Kodierung definiert wurde.

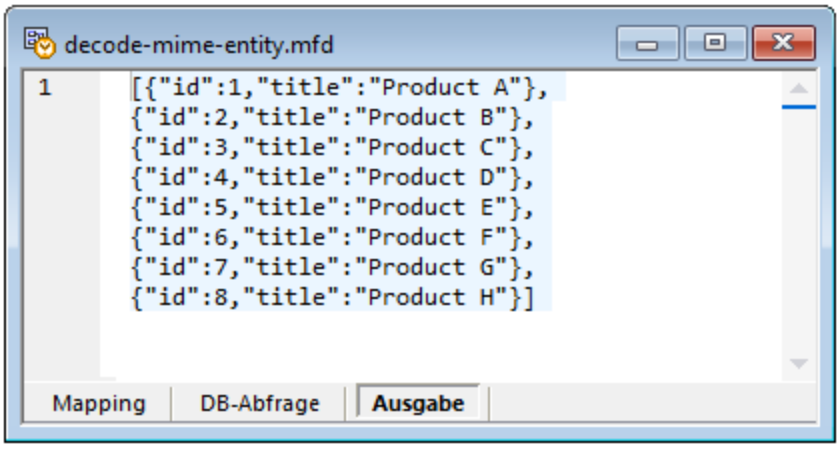
© 2016-2022 Altova GmbH

Beispiel

Im folgenden Mapping wird ein Webservice-Aufruf verwendet, der JSON-Inhalt zurückgibt.

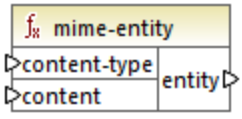


Bei Ausführung dieses Mappings wird das Ergebnis des Webservice-Aufrufs als Input für die Funktion **`decode-mime-entity`** bereitgestellt. Die Funktion dekodiert den vom Aufruf zurückgegebenen Raw (rohen) Body gemäß dem Parameter **charset** des **Content-Type**-Header. Sie erhalten das folgende Ergebnis:



6.7.21.4 mime-entity

Die Funktion **`mime-entity`** gibt eine MIME Entity zurück. Die MIME Entity wird anhand des für das **content**-Argument bereitgestellten Inhalts erstellt und hat den für das **type**-Argument bereitgestellten Medientyp.



Sprachen

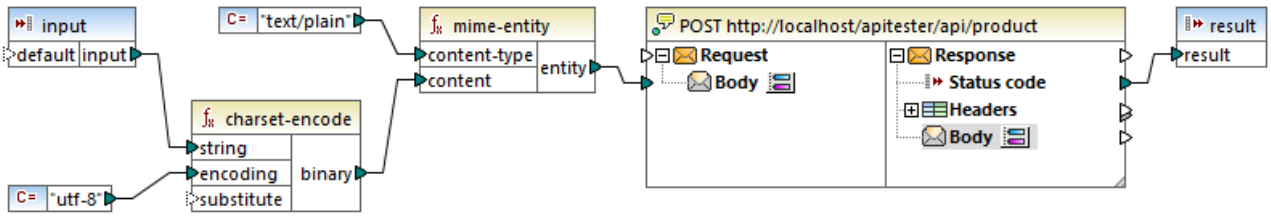
Built-in.

Argumente

Name	Typ	Beschreibung
content-type	`mf:MediaType`	Der Medientyp, der diese MIME Entity beschreibt, z.B. **`image/png`**.
content	`xs:base64Binary`	Die eigentlichen anhand des Base64-Kodierungsschemas kodierten Daten. Sie können solche Daten mit Hilfe der Funktion charset-encode[1166] kodieren.

Beispiel

Im folgenden Mapping wird ein Webservice aufgerufen, der per POST-Request übermittelte nicht strukturierte Daten erwartet.



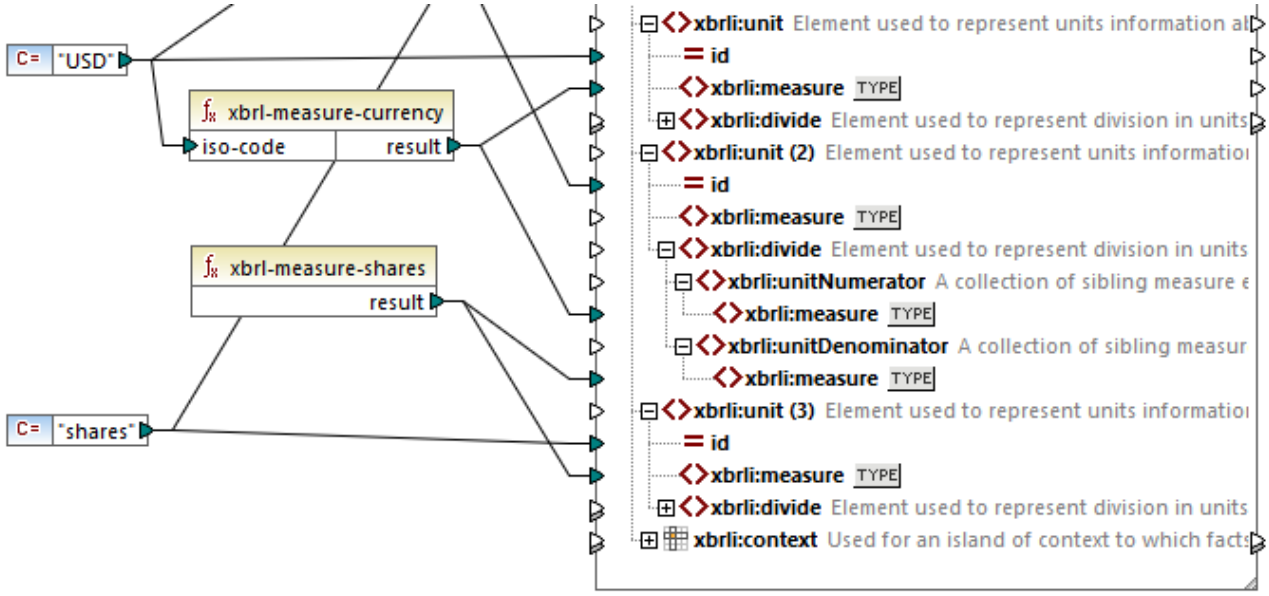
Der Input-Text im obigen Mapping wird mit Hilfe einer einfachen Input-Komponente als Parameter für das Mapping bereitgestellt, siehe Bereitstellen von Parametern für das Mapping[867]. Der als Input erhaltene String wird mit Hilfe der Funktion charset-encode[1166] und unter Verwendung der **utf-8** Kodierung in Binärdaten konvertiert. Die Funktion **`mime-entity`** erhält den binären Inhalt als Argument und gibt den Body des Request (die MIME Entity) zurück, welcher auf den **Body**-Input des Webservice-Aufrufs gemappt wurde.

6.7.22 xbrl

Die XBRL-Bibliothek enthält Funktionen, die Daten in das für XBRL-Instanz- oder Taxonomiedateien erforderliche QName-Format konvertieren.

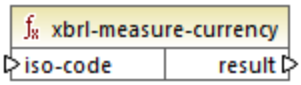
Die Child-Elemente **xbrl:ID** und **xbrl:measure** des Elements **xbrl:unit** sind in einer XBRL-Instanzdatei obligatorisch und müssen gemappt werden, damit das Mapping gültig ist.

© 2016-2022 Altova GmbH



6.7.22.1 xbrl-measure-currency

Das Ergebnis ist der QName für Datenelemente des Typs `monetaryItemType` aus einem ISO 4217-Währungscode.



Sprachen

Built-in, C++, C#, Java.

Parameter

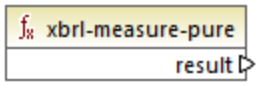
Name	Typ	Beschreibung
iso-code	`xs:string`	Der ISO 4217 Währungscode.

Beispiel

Siehe Beispiel Microsoft Excel auf XBRL [818].

6.7.22.2 xbrl-measure-pure

Das Ergebnis ist der QName für Tarif-, Prozentsatz- oder Verhältnisdatenelemente.

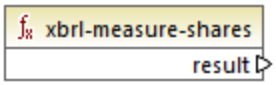


Sprachen

Built-in, C++, C#, Java.

6.7.22.3 xbrl-measure-shares

Das Ergebnis ist der QName für Datenelemente vom Typ `sharesItemType`.



Sprachen

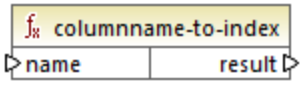
Built-in, C++, C#, Java.

6.7.23 xlsx

Die **xlsx**-Bibliothek enthält Funktionen, die Daten in erster Linie von/in Excel Datums- und Uhrzeitformate konvertieren. Die Funktionen **`date-to-xlsx`**, **`datetime-to-xlsx`** und **`time-to-xlsx`** dienen normalerweise dazu , Datums- und Uhrzeitwerte in ein Excel-Zieldatenblatt zu schreiben. Mit den Funktionen **`xlsx-to-date`**, **`xlsx-to-datetime`** und **`xslx-to-time`** können Sie Datums- und Uhrzeitwerte aus einem Excel-Quelldatenblatt in Werte vom Typ `xs:date`, `xs:time` und `xs:dateTime` konvertieren.

6.7.23.1 columnname-to-index

Das Ergebnis ist der Index der Spalte des angegebenen Namens. Der Index von Spalte "A" ist 1.



Sprachen

Built-in,C#, Java, XSLT 2.0, XSLT 3.0..

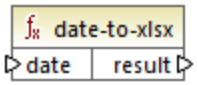
© 2016-2022 Altova GmbH

Parameter

Name	Typ	Beschreibung
name	`xs:string`	Der in Excel angezeigte Spaltenname, beginnend mit "A", "B", "C", usw.

6.7.23.2 date-to-xlsx

Das Ergebnis ist die Excel-Darstellung des aus der Quelle extrahierten *date*-Werts in Form einer laufenden Nummer, die für das Datum steht. Standardmäßig ist der 1. Jänner 1900 in Microsoft Excel für Windows die laufende Nummer **1**.



Sprachen

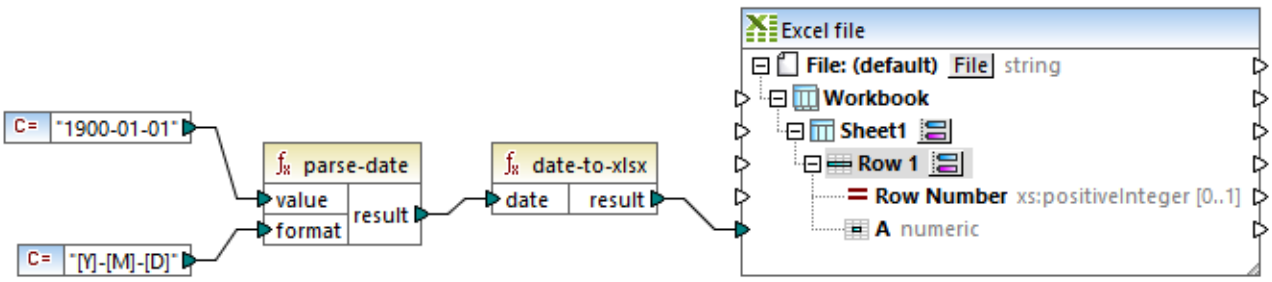
Built-in,C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	`xs:date`	Der Input-Datumswert.

Beispiel

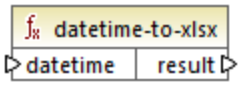
Im folgenden Mapping wird der String "1900-01-01" mit Hilfe der **`parse-date`**-Funktion in ein Datum konvertiert. Der damit erzeugte `xs:date`-Wert (1. Jänner 1900) wird anschließend als Argument für die Funktion **`date-to-xlsx`** bereitgestellt, welche seine numerische Darstellung zurückgibt und in eine Excel-Zelle schreibt. Der Ergebnis-Zellenwert ist, wie erwartet, 1.



Beachten Sie dass das Format der Excel-Zielzelle "Numerisch" ist. Wenn Sie mit der rechten Maustaste in Excel in die Zelle klicken und diese mit dem Befehl **Zellen formatieren** als Datum formatieren, wird der Zellenwert zu `1/1/1900`.

6.7.23.3 datetime-to-xslx

Das Ergebnis ist die Excel-Darstellung des aus der Quelle extrahierten *Datums- und Uhrzeitwerts*.



Sprachen

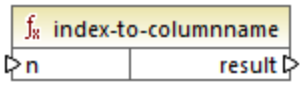
Built-in,C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
datetime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.

6.7.23.4 index-to-columnname

Das Ergebnis ist der Name der Excel-Spalte (z.B. "A", "B","C") mit dem Spaltenindex (z.B. 1, 2, 3). Der Name der Spalte 1 ist "A".



Sprachen

Built-in,C#, Java, XSLT 2.0, XSLT 3.0..

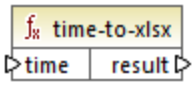
Parameter

Name	Typ	Beschreibung
n	`xs:int`	Definiert den 1-basierten Index der Spalte.

© 2016-2022 Altova GmbH

6.7.23.5 time-to-xlsx

Das Ergebnis ist die Excel-Darstellung des aus der Quelle extrahierten *Uhrzeitwerts*.



Sprachen

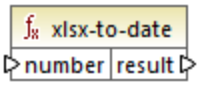
Built-in,C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time	`xs:time`	Der Input-Wert vom Typ `xs:time`.

6.7.23.6 xlsx-to-date

Das Ergebnis ist ein `xs:date`-Wert, der anhand der numerischen Darstellung des Datums in Excel ermittelt wird.



Sprachen

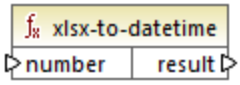
Built-in,C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
number	`xs:double`	Die fortlaufende Nummer des Datums in Excel.

6.7.23.7 xlsx-to-datetime

Das Ergebnis ist ein `xs: dateTime` -Wert, der anhand der numerischen Darstellung des Datums- und Uhrzeitwerts in Excel ermittelt wird.



Sprachen

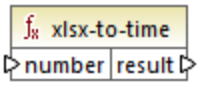
Built-in,C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
number	`xs:double`	Die Darstellung des Datums- und Uhrzeitwerts in Excel.

6.7.23.8 xlsx-to-time

Das Ergebnis ist ein `xs:time` -Wert, der anhand der numerischen Darstellung des Uhrzeitwerts in Excel ermittelt wird.



Sprachen

Built-in,C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
number	`xs:double`	Die Darstellung der Uhrzeit in Excel.

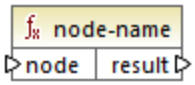
6.7.24 xpath2 | accessors (Accessor-Funktionen)

Die Funktionen aus der Unterbibliothek **xpath2 | accessors** rufen Informationen über XML-Nodes oder Datenelemente ab. Diese Funktionen stehen zur Verfügung, wenn entweder die Sprache XSLT2 oder XQuery ausgewählt ist.

© 2016-2022 Altova GmbH

6.7.24.1 base-uri

Die **base-uri** Funktion erhält ein Node-Argument als Input und gibt die URI der XML-Ressource, die den Node enthält, zurück. Die Ausgabe hat den Typ `xs:string`.



Sprachen

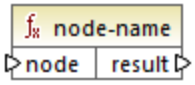
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
node	**mf:node**	Der Input-Node.

6.7.24.2 node-name

Die **node-name**-Funktion erhält einen Node-Namen als Input-Argument und gibt seinen QName zurück. Wenn der QName als String dargestellt wird, hat er die Form `präfix:lokalerName`, wenn der Node ein Präfix hat, oder `lokalerName`, wenn der Node kein Präfix hat. Um die Namespace URI eines Node zu eruieren, verwenden Sie die **namespace-URI-from-QName**[1076] Funktion.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

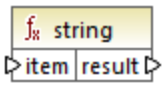
Parameter

Name	Typ	Beschreibung
node	**mf:node**	Der Input-Node.

6.7.24.3 string

Die **string** Funktion funktioniert wie der `xs:string` Konstruktor: Sie konvertiert ihr Argument in `xs:string`.

Wenn das Input-Argument ein Wert eines atomaren Typs ist (z.B. `xs:decimal`), wird dieser atomare Wert in einen Wert vom Typ `xs:string` konvertiert. Wenn das Input-Argument ein Node ist, wird der String-Wert des Node extrahiert. (Der String-Wert eines Node ist eine Verkettung der Werte der untergeordneten Nodes des Node.)



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
item	`mf:item`	Der Input-Wert.

6.7.25 xpath2 | anyURI functions (anyURI-Funktionen)

Die Unterbibliothek **xpath2 | anyURI** enthält die **`resolve-uri`**-Funktion. Diese Funktion steht zur Verfügung, wenn entweder die Sprache XSLT2 oder XQuery ausgewählt ist.

6.7.25.1 resolve-uri

Die **`resolve-uri`** Funktion erhält eine URI als erstes Argument und löst diese anhand der Basis-URI im zweiten Argument auf. Die Ausgabe hat den Datentyp `xs:string`. Beide Inputs werden von der Funktion als Strings behandelt; es wird nicht überprüft, ob die von diesen URIs angegebenen Ressourcen tatsächlich vorhanden sind.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

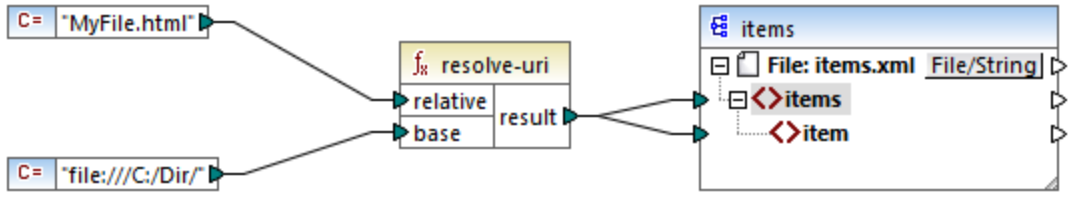
Parameter

Name	Typ	Beschreibung
relative	`xs:string`	Die relative URI, die anhand der Basis aufgelöst werden soll.
base	`xs:string`	Die Basis-URI.

© 2016-2022 Altova GmbH

Beispiel

Im Mapping unten liefert das erste Argument die relative URI `MyFile.html` und das zweite Argument die Basis-URI `file:///C:/Dir/`. Die aufgelöste URI ist eine Verkettung aus diesen beiden und lautet `file:///C:/Dir/MyFile.html`.

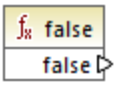


6.7.26 xpath2 | boolean functions (Boolesche Funktionen)

Die Booleschen Funktionen `true` und `false` erhalten kein Argument und geben die Booleschen Konstantenwerte **true** und **false** zurück. Sie können verwendet werden, wenn ein Boolescher Konstantenwert benötigt wird.

6.7.26.1 false

Gibt den Booleschen Wert **false** zurück.

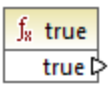


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

6.7.26.2 true

Gibt den Booleschen Wert **true** zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

6.7.27 xpath2 | constructors (Konstruktoren)

Mit den Funktionen aus der Unterbibliothek "constructors" der XPath 2.0-Bibliothek werden anhand des Input-Texts bestimmte Datentypen konstruiert. In der folgenden Tabelle sind die verfügbaren Konstruktorfunktionen aufgelistet.

xs:ENTITY	**xs:double**	**xs:nonPositiveInteger**
xs:ID	**xs:duration**	**xs:normalizedString**
xs:IDREF	**xs:float**	**xs:positiveInteger**
xs:NCName	**xs:gDay**	**xs:short**
xs:NMTOKEN	**xs:gMonth**	**xs:string**
xs:Name	**xs:gMonthDay**	**xs:time**
xs:QName	**xs:gYear**	**xs:token**
xs:anyURI	**xs:gYearMonth**	**xs:unsignedByte**
xs:base64Binary	**xs:hexBinary**	**xs:unsignedInt**
xs:boolean	**xs:int**	**xs:unsignedLong**
xs:byte	**xs:integer**	**xs:unsignedShort**
xs:date	**xs:language**	**xs:untypedAtomic**
xs:dateTime	**xs:long**	**xs:yearMonthDuration**
xs:dayTimeDuration	**xs:negativeInteger**	
xs:decimal	**xs:nonNegativeInteger**	

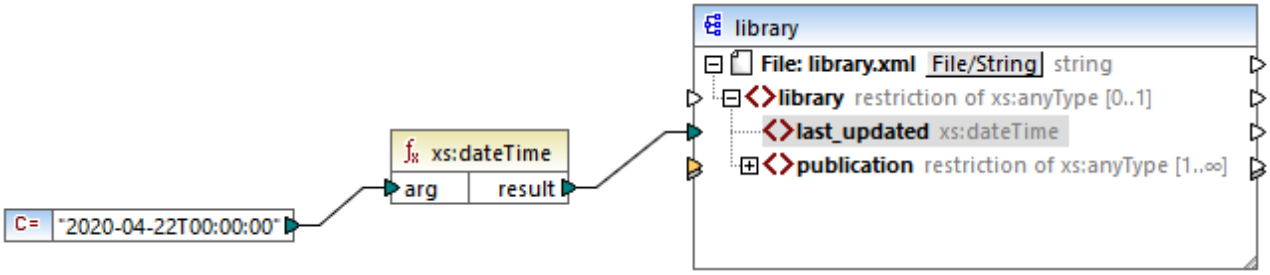
Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Beispiel

Normalerweise muss das lexikalische Format des Input-Texts dem vom zu konstruierenden Datentyp erwarteten entsprechen. Andernfalls ist die Transformation nicht erfolgreich. Um z.B. mit Hilfe der **xs:dateTime**-Konstruktorfunktion einen `xs:dateTime`-Wert zu konstruieren, muss der Input-Text das lexikalische Format des `xs:dateTime`-Datentyps haben. Dieses Format ist **YYYY-MM-DDTHH:mm:ss**.

© 2016-2022 Altova GmbH



Im oben gezeigten Mapping wurde als Input-Argument der Funktion eine String-Konstante (**"2020-04-28T00:00:00"**) verwendet. Dieser Input hätte auch aus einem Datenelement im Quelldokument abgerufen werden können. Die **xs:dateTime**-Funktion gibt den Wert **2020-04-28T00:00:00** vom Typ `xs:dateTime` zurück.

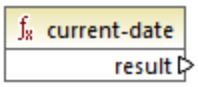
Um zu sehen, welcher Datentyp (einschließlich des Datentyps der Funktionsargumente) von einem Mapping-Datenelement erwartet wird, platzieren Sie den Mauszeiger über den entsprechenden Input- oder Output-Konnektor.

6.7.28 xpath2 | context functions (Kontextfunktionen)

Die Kontextfunktionen aus der **xpath2**-Bibliothek stellen verschiedene Informationen über das aktuelle Datum und die Uhrzeit, die vom Prozessor verwendete Standard-Collation, die Größe der aktuellen Sequenz und die Position des aktuellen Node bereit.

6.7.28.1 current-date

Gibt das aktuelle Datum (`xs:date`) anhand der Systemuhr zurück.

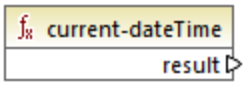


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

6.7.28.2 current-dateTime

Gibt das aktuelle Datum und die aktuelle Uhrzeit (`xs:dateTime`) anhand der Systemuhr zurück.

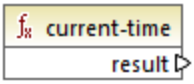


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

6.7.28.3 current-time

Gibt die aktuelle Uhrzeit (`xs:time`) anhand der Systemuhr zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

6.7.28.4 default-collation

Die Funktion **`default-collation`** erhält kein Argument und gibt die Standard-Collation zurück, d.h. die Collation, die verwendet wird, wenn keine Collation für eine Funktion, für die eine solche definiert werden kann, festgelegt ist.

Vergleiche wie für die Funktionen **`max-string`** und **`min-string`** basieren auf der Standard-Collation.

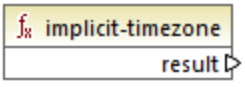


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

6.7.28.5 implicit-timezone

Gibt den Wert der Eigenschaft "implizite Zeitzone" aus dem Auswertungskontext zurück.



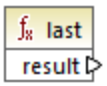
Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

© 2016-2022 Altova GmbH

6.7.28.6 last

Gibt die Anzahl der Datenelemente innerhalb der gerade verarbeiteten Sequenz von Datenelementen zurück. Beachten Sie, dass die Sequenz der Datenelemente durch den aktuellen Mapping-Kontext[1333] bestimmt wird, wie im Beispiel unten beschrieben.



Sprachen

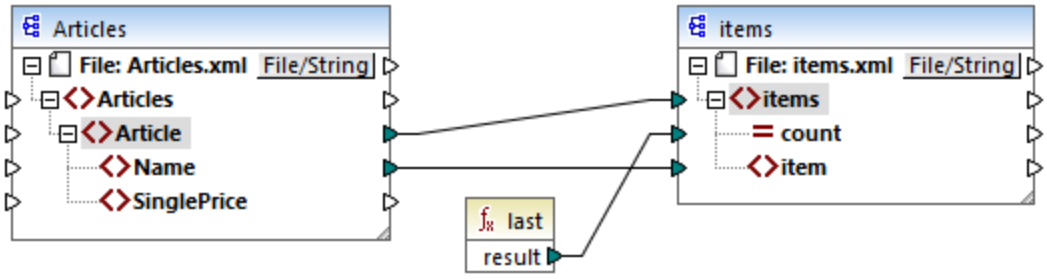
XQuery, XSLT 2.0, XSLT 3.0..

Beispiel

Angenommen, Sie haben die folgende XML-Quelldatei:

```
<Articles>
   <Article>
      <Name>T-Shirt</Name>
      <SinglePrice>25</SinglePrice>
   </Article>
   <Article>
      <Name>Socks</Name>
      <SinglePrice>2.30</SinglePrice>
   </Article>
   <Article>
      <Name>Jacket</Name>
      <SinglePrice>57.50</SinglePrice>
   </Article>
</Articles>
```

Es sollen Daten in eine XML-Datei mit einem anderen Schema kopiert werden. Außerdem muss die Anzahl aller Datenelemente in der XML-Zieldatei gespeichert werden. Dies lässt sich mit einem Mapping wie dem unten gezeigten, ermitteln:



Im Beispiel unten gibt die Funktion **last** die Position des letzten Node im aktuellen Parent-Kontext zurück und befüllt das Attribut **count** mit dem Wert **3**.

```
<items count="3">
   <item>T-Shirt</item>
   <item>Pants</item>
   <item>Jacket</item>
</items>
```

Beachten Sie, dass der Wert **3** die Position des letzen Datenelements (also die Anzahl aller Datenelemente) in dem durch die Verbindung zwischen **Article** und **items** erstellten Mapping-Kontext ist. Wäre diese Verbindung nicht vorhanden, würden dennoch Datenelemente in die Zielkomponente kopiert werden, doch würde die Funktion **last** fälschlicherweise den Wert **1** zurückgeben, da sie keinen Parent-Kontext[1340] hätte, über den sie iterieren kann. (Genauer gesagt, würde die Funktion den impliziten zwischen den Root-Elementen der beiden Komponenten erstellten Standardkontext verwenden, der eine Sequenz von 1- und nicht, wie erwartet 3 - Datenelementen erzeugt).

Im Allgemeinen empfiehlt es sich, anstelle der **last**-Funktion, die **count**[1028]-Funktion aus der **core**-Bibliothek zu verwenden, da die count-Funktion ein **parent-context**-Argument hat, mit Hilfe dessen Sie den Mapping-Kontext explizit ändern können.

6.7.29 xpath2 | durations, date and time functions (Zeitdauer-, Datums- und Uhrzeitfunktionen)

Mit Hilfe der Zeitdauer-, Datums- und Uhrzeitfunktionen aus der **xpath2**-Bibliothek können Sie die Zeitzone in Datums- und Uhrzeitwerten anpassen, bestimmte Komponenten aus Datums-, Uhrzeit- und Zeitdauerwerten extrahieren und Datums- und Uhrzeitwerte subtrahieren.

Anpassen der Zeitzone

Zum Anpassen der Zeitzone in Datums- und Uhrzeitwerten stehen die folgenden Funktionen zur Verfügung:

- **adjust-date-to-timezone**
- **adjust-date-to-timezone** (mit Zeitzonenargument)
- **adjust-dateTime-to-timezone**
- **adjust-dateTime-to-timezone** (mit Zeitzonenargument)
- **adjust-time-to-timezone**
- **adjust-time-to-timezone** (mit Zeitzonenargument)

Jede dieser Funktionen erhält einen `xs:date`, `xs:time` oder `xs:dateTime`-Wert als erstes Argument und passt die Input-Daten je nach dem Wert des zweiten Arguments (falls eines vorhanden ist) durch Hinzufügen, Entfernen oder Ändern der Zeitzonen-Komponente an.

Wenn das erste Argument keine Zeitzone enthält, (z.B. das Datum `2009-01` oder die Uhrzeit `14:00:00`), ergeben sich die folgenden Möglichkeiten:

- Wenn das **Zeitzonenargument** vorhanden ist, enthält das Ergebnis die im zweiten Argument angegebene Zeitzone. Die Zeitzone im zweiten Argument wird hinzugefügt.
- Wenn das **Zeitzonenargument** fehlt, enthält das Ergebnis die implizite Zeitzone, d.h. die Zeitzone des Systems. Die Zeitzone des Systems wird hinzugefügt.
- Wenn das **Zeitzonenargument** leer ist, enthält das Ergebnis keine Zeitzone.

© 2016-2022 Altova GmbH

Wenn das erste Argument eine Zeitzone enthält, (z.B. das Datum `01.01.2020+01:00` oder die Uhrzeit `14:00:00+01:00`), ergeben sich die folgenden Möglichkeiten:

- Wenn das **Zeitzonenargument** vorhanden ist, enthält das Ergebnis die im zweiten Argument angegebene Zeitzone. Die ursprüngliche Zeitzone wird durch die im zweiten Argument definierte Zeitzone ersetzt.
- Wenn das **Zeitzonenargument** fehlt, enthält das Ergebnis die implizite Zeitzone, d.h. die Zeitzone des Systems. Die ursprüngliche Zeitzone wird durch die Zeitzone des Systems ersetzt.
- Wenn das **Zeitzonenargument** leer ist, enthält das Ergebnis keine Zeitzone.

Extrahieren von Komponenten von Datums- und Uhrzeitwerten

Um aus Datums- und Uhrzeitwerten numerische Werte wie Stunden, Minuten, Monate, usw. zu extrahieren, stehen die folgenden Funktionen zur Verfügung:

- **day-from-date**
- **day-from-dateTime**
- **hours-from-dateTime**
- **hours-from-time**
- **minutes-from-dateTime**
- **minutes-from-time**
- **month-from-date**
- **month-from-dateTime**
- **seconds-from-dateTime**
- **seconds-from-time**
- **timezone-from-date**
- **timezone-from-dateTime**
- **timezone-from-time**
- **year-from-date**
- **year-from-dateTime**

Jede dieser Funktionen extrahiert eine bestimmte Komponente aus `xs:date`-, `xs:time`-, `xs:dateTime`- und `xs:duration`-Werten. Das Ergebnis ist entweder ein `xs:integer`- oder `xs:decimal`-Typ.

Extrahieren von Zeitdauerkomponenten

Um aus einer Zeitdauer Uhrzeitkomponenten zu extrahieren, stehen die folgenden Funktionen zur Verfügung:

- **days-from-duration**
- **hours-from-duration**
- **minutes-from-duration**
- **months-from-duration**
- **seconds-from-duration**
- **years-from-duration**

Die Zeitdauer muss entweder als `xs:yearMonthDuration` (zum Extrahieren von Jahren und Monaten) oder als `xs:dayTimeDuration` (zum Extrahieren von Tagen, Stunden, Minuten und Sekunden) definiert werden. Alle Funktionen geben ein Ergebnis vom Typ `xs:integer` zurück; ausgenommen davon ist die Funktion **seconds-from-duration**, die ein Ergebnis vom Typ `xs:decimal` zurückgibt.

Subtrahieren von Datums- und Uhrzeitwerten

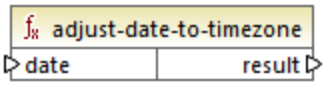
Zum Subtrahieren von Datums- und Uhrzeitwerten stehen die folgenden Funktionen zur Verfügung:

- **subtract-dateTimes**
- **subtract-dates**
- **subtract-times**

Mit Hilfe der Subtraktionsfunktionen können Sie einen Uhrzeitwert von einem anderen subtrahieren und einen Zeitdauerwert errechnen.

6.7.29.1 adjust-date-to-timezone

Passt einen `xs:date`-Wert an die implizite Zeitzone im Auswertungskontext (die Zeitzzone des Systems) an.



Sprachen

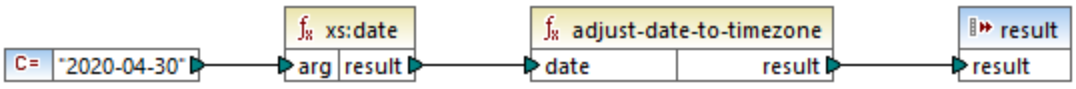
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	**xs:date**	Der Input-Wert vom Typ `xs:date`.

Beispiel

Im folgenden Mapping wird anhand eines String ein `xs:date`-Wert konstruiert und als Argument für die **adjust-date-to-timezone** bereitgestellt.



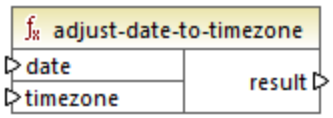
XSLT 2.0-Mapping

Wenn das Mapping auf einem Rechner läuft, dessen Systemzeitzone +02:00 ist, passt die Funktion den Datumswert an, um die Zeitzone des Systems zu inkludieren. Die Mapping-Ausgabe ist daher **2020-04-30+02:00**.

© 2016-2022 Altova GmbH

6.7.29.2 adjust-date-to-timezone

Passt einen `xs:date`-Wert an eine bestimmte Zeitzone oder gar keine Zeitzone an. Wenn das Argument **timezone** eine leere Sequenz ist, gibt die Funktion einen `xs:date`-Wert ohne eine Zeitzone zurück. Andernfalls gibt sie einen `xs:date`-Wert mit einer Zeitzone zurück.



Sprachen

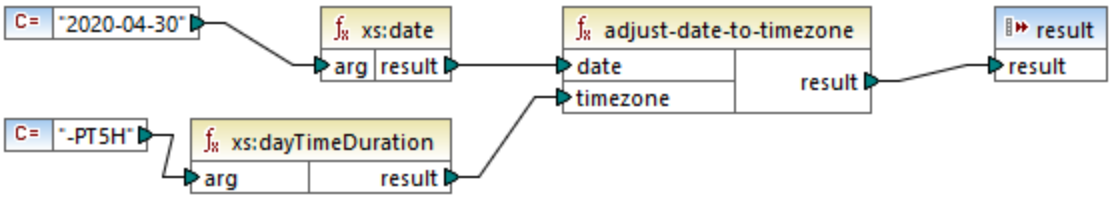
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	**`xs:date`**	Der Input-Wert vom Typ `xs:date`.
timezone	**`xs:dayTimeDuration`**	Die als `xs:dayTimeDuration`-Wert ausgedrückte Zeitzone. Der Wert kann negativ sein. So kann etwa ein Zeitzonenwert von -5 Stunden als **`-PT5H`** ausgedrückt werden.

Beispiel

Im folgenden Mapping werden beide Parameter für die Funktion **`adjust-date-to-timezone`** mit Hilfe der entsprechenden XPath 2 Konstruktorfunktionen[1194] anhand von Strings konstruiert. Ziel des Mappings ist eine Anpassung der Zeitzone an -5 Stunden. Diese Zeitzone kann als **`-PT5H`** ausgedrückt werden.

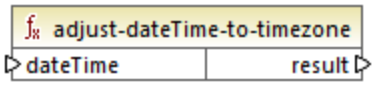


XSLT 2.0-Mapping

Die Funktion passt den Datumswert an die als Argument bereitgestellte Zeitzone an. Die Mapping-Ausgabe ist daher **`2020-04-30-05:00`**.

6.7.29.3 adjust-dateTime-to-timezone

Passt einen `xs:dateTime`-Wert an die implizite Zeitzone im Auswertungskontext (die Zeitzzone des Systems) an.



Sprachen

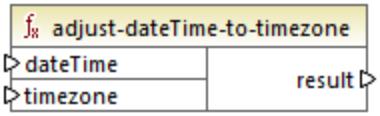
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.

6.7.29.4 adjust-dateTime-to-timezone

Passt einen `xs:dateTime`-Wert an eine bestimmte Zeitzone oder gar keine Zeitzone an. Wenn das Argument **timezone** eine leere Sequenz ist, gibt die Funktion einen `xs:dateTime`-Wert ohne eine Zeitzone zurück. Andernfalls gibt sie einen `xs:dateTime`-Wert mit einer Zeitzone zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

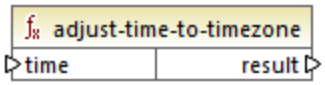
Name	Typ	Beschreibung
dateTime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.
timezone	`xs:dayTimeDuration`	Die als `xs:dayTimeDuration`-Wert ausgedrückte Zeitzone. Der Wert kann negativ sein. So kann etwa ein Zeitzonenwert von -5

© 2016-2022 Altova GmbH

Name	Typ	Beschreibung
		Stunden als **-PT5H** ausgedrückt werden.

6.7.29.5 adjust-time-to-timezone

Passt einen `xs:Time`-Wert an die implizite Zeitzone im Auswertungskontext (die Zeitzzone des Systems) an.



Sprachen

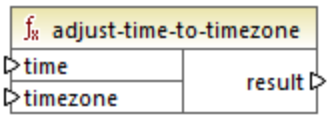
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time	**xs:time**	Der Input-Wert vom Typ `xs:time`.

6.7.29.6 adjust-time-to-timezone

Passt einen `xs:time`-Wert an eine bestimmte Zeitzone oder gar keine Zeitzone an. Wenn das Argument **timezone** eine leere Sequenz ist, gibt die Funktion einen `xs:time`-Wert ohne eine Zeitzone zurück. Andernfalls gibt sie einen `xs:time`-Wert mit einer Zeitzone zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

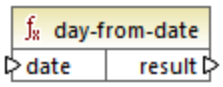
Parameter

Name	Typ	Beschreibung
time	**xs:time**	Der Input-Wert vom Typ `xs:time`.
timezone	**xs:dayTimeDuration**	Die als `xs:dayTimeDuration`-Wert ausgedrückte Zeitzone. Der

Name	Typ	Beschreibung
		Wert kann negativ sein. So kann etwa ein Zeitzonenwert von -5 Stunden als `-PT5H` ausgedrückt werden.

6.7.29.7 day-from-date

Gibt einen `xs:integer`-Wert für den Tagesteil des als Argument bereitgestellten `xs:date`-Werts zurück.



Sprachen

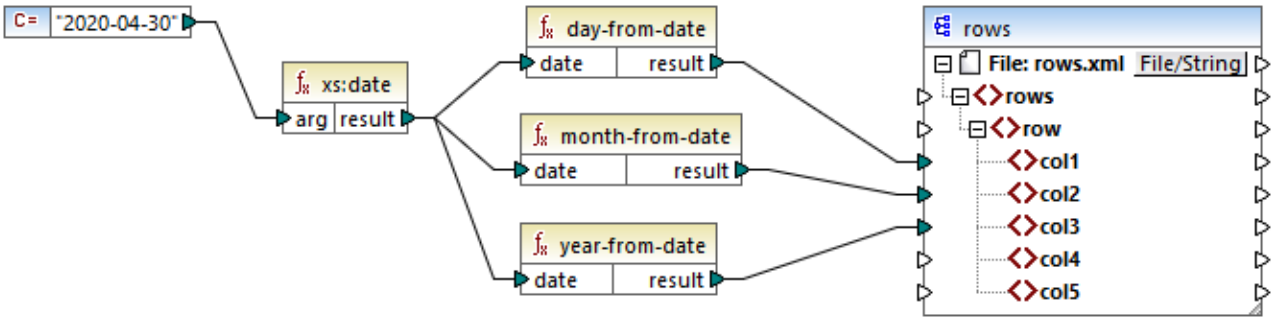
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	`xs:date`	Der Input-Wert vom Typ `xs:date`.

Beispiel

Im folgenden Mapping wird ein String mit Hilfe der **`xs:date`**-Konstruktorfunktion in einen `xs:date`-Wert konvertiert. Die Funktionen **`day-from-date`**, **`month-from-date`** und **`year-from-date`** extrahieren jeweils den entsprechenden Teil des Datums und schreiben ihn in ein separates Datenelement in der XML-Zieldatei.



XQuery 1.0-Mapping

Das Ergebnis des Mappings sieht folgendermaßen aus:

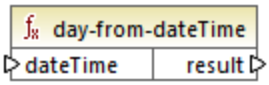
```
<rows>
   <row>
```


© 2016-2022 Altova GmbH

```
      <col1>30</col1>
      <col2>4</col2>
      <col3>2020</col3>
   </row>
</rows>
```

6.7.29.8 day-from-dateTime

Gibt einen `xs:integer`-Wert für den Tagesteil des als Argument bereitgestellten `xs:dateTime`-Werts zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.

6.7.29.9 days-from-duration

Gibt einen `xs:integer`-Wert für die "Tage"-Komponente der kanonischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

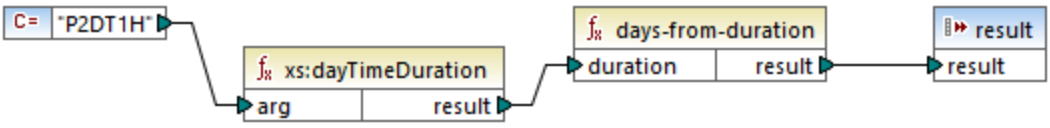
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
duration	`xs:duration`	Der Input-Wert vom Typ `xs:duration`.

Beispiel

Im unten gezeigten Mapping wird der `xs:dayTimeDuration`-Wert anhand von **`P2DT1H`** (2 Tage und 1 Stunde) konstruiert und als Input für die Funktion **`days-from-duration`** bereitgestellt. Das Ergebnis ist **2**.



XSLT 2.0-Mapping

Anmerkung: Wenn die Zeitdauer **P1DT24H** (1 Tag und 24 Stunden) ist, gibt die Funktion **2** und nicht **1** zurück. Der Grund dafür ist, dass die kanonische Darstellung von **P1DT24H** tatsächlich **P2D** (2 Tage) ist.

6.7.29.10 hours-from-dateTime

Gibt einen `xs:integer`-Wert für den Stundenteil des als Argument bereitgestellten `xs:dateTime`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	**xs:dateTime**	Der Input-Wert vom Typ `xs:dateTime`.

6.7.29.11 hours-from-duration

Gibt einen `xs:integer`-Wert für die "Stunden"-Komponente der kanonischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
duration	**xs:duration**	Der Input-Wert vom Typ `xs:duration`.

Beispiel

Wenn die Zeitdauer **PT1H60M** (1 Stunde und 60 Minuten) ist, gibt die Funktion **2** und nicht **1** zurück. Der Grund dafür ist, dass die kanonische Darstellung von **PT1H60M** tatsächlich **PT2H** (2 Stunden) ist.

© 2016-2022 Altova GmbH

6.7.29.12 hours-from-time

Gibt einen `xs:integer`-Wert für den Stundenteil des als Argument bereitgestellten `xs:time`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time	**`xs:time`**	Der Input-Wert vom Typ `xs:time`.

6.7.29.13 minutes-from-dateTime

Gibt einen `xs:integer`-Wert für den Minutenteil des als Argument bereitgestellten `xs:dateTime`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	**`xs:dateTime`**	Der Input-Wert vom Typ `xs:dateTime`.

6.7.29.14 minutes-from-duration

Gibt einen `xs:integer`-Wert für die "Minuten"-Komponente der kanonischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
duration	**`xs:duration`**	Der Input-Wert vom Typ `xs:duration`.

Beispiel

Wenn die Zeitdauer **PT1M60S** (1 Minute und 60 Sekunden) ist, gibt die Funktion **2** und nicht **1** zurück. Der Grund dafür ist, dass die kanonische Darstellung von **PT1M60S** tatsächlich **PT2M** (2 Minuten) ist.

6.7.29.15 minutes-from-time

Gibt einen `xs:integer`-Wert für den Minutenteil des als Argument bereitgestellten `xs:time`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time	**xs:time**	Der Input-Wert vom Typ `xs:time`.

6.7.29.16 month-from-date

Gibt einen `xs:integer`-Wert für den Monatsteil des als Argument bereitgestellten `xs:date`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	**xs:date**	Der Input-Wert vom Typ `xs:date`.

6.7.29.17 month-from-dateTime

Gibt einen `xs:integer`-Wert für den Monatsteil des als Argument bereitgestellten `xs:dateTime`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

© 2016-2022 Altova GmbH

Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.

6.7.29.18 months-from-duration

Gibt einen `xs:integer`-Wert für die Monatskomponente der kanonischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
duration	`xs:duration`	Der Input-Wert vom Typ `xs:duration`.

6.7.29.19 seconds-from-dateTime

Gibt einen xs:integer-Wert zurück, der für die Sekundenkomponente im lokalisierten Wert von dateTime steht.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	

6.7.29.20 seconds-from-duration

Gibt einen `xs:integer`-Wert für die Sekundenkomponente der kanonischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
duration	`xs:duration`	Der Input-Wert vom Typ `xs:duration`.

6.7.29.21 seconds-from-time

Gibt einen `xs:integer`-Wert für den Sekundenteil des als Argument bereitgestellten `xs:time`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time	`xs:time`	Der Input-Wert vom Typ `xs:time`.

6.7.29.22 subtract-dateTimes

Gibt den `xs:dayTimeDuration`-Wert zurück, der der Differenz zwischen dem normalisierten Wert von **dateTime1** und dem normalisierten Wert von **dateTime2** entspricht, zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime1	`xs:dateTime`	Der erste Input-Wert.
dateTime2	`xs:dateTime`	Der zweite Input-Wert.

6.7.29.23 subtract-dates

Gibt den `xs:dayTimeDuration`-Wert, der der Differenz zwischen dem normalisierten Wert von **date1** und dem normalisierten Wert von **date2** entspricht, zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

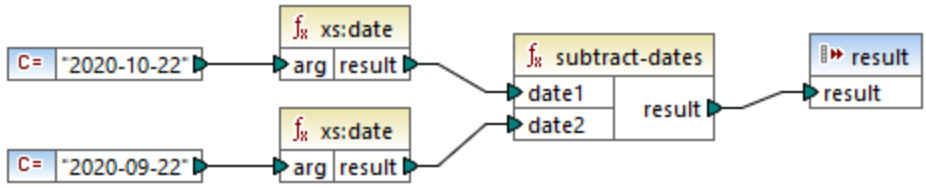
© 2016-2022 Altova GmbH

Parameter

Name	Typ	Beschreibung
date1	`xs:date`	Der erste Input-Wert.
date2	`xs:date`	Der zweite Input-Wert.

Beispiel

Im unten gezeigten Mapping werden zwei Datumswerte voneinander subtrahiert (2020-10-22 minus 2020-09-22). Das Ergebnis ist der Wert **`P30D`** vom Typ `xs:dayTimeDuration`, der eine Zeitdauer von 30 Tagen darstellt.



6.7.29.24 subtract-times

Gibt den `xs:dayTimeDuration`-Wert, der der Differenz zwischen dem normalisierten Wert von **time1** und dem normalisierten Wert von **time2** entspricht, zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time1	`xs:time`	Der erste Input-Wert.
time2	`xs:time`	Der zweite Input-Wert.

6.7.29.25 timezone-from-date

Gibt die Zeitzonenkomponente des als Argument bereitgestellten Datums zurück. Das Ergebnis ist ein `xs:dayTimeDuration`-Wert, der die Abweichung von UTC angibt; der Wert kann von +14:00 bis einschließlich -14:00 Stunden reichen.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	`xs:date`	Der Input-Wert vom Typ `xs:date`.

6.7.29.26 timezone-from-dateTime

Gibt die Zeitzonenkomponente des als Argument bereitgestellten `xs:dateTime`-Werts zurück. Das Ergebnis ist ein `xs:dayTimeDuration`-Wert, der die Abweichung von UTC angibt; der Wert kann von +14:00 bis einschließlich -14:00 Stunden reichen.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.

6.7.29.27 timezone-from-time

Gibt die Zeitzonenkomponente des als Argument bereitgestellten `xs:time`-Werts zurück. Das Ergebnis ist ein `xs:dayTimeDuration`-Wert, der die Abweichung von UTC angibt; der Wert kann von +14:00 bis einschließlich -14:00 Stunden reichen.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time	`xs:time`	Der Input-Wert vom Typ `xs:time`.

© 2016-2022 Altova GmbH

6.7.29.28 year-from-date

Gibt einen `xs:integer`-Wert für den Jahresteil des als Argument bereitgestellten `xs:date`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	**`xs:date`**	Der Input-Wert vom Typ `xs:date`.

6.7.29.29 year-from-dateTime

Gibt einen `xs:integer`-Wert für den Jahresteil des als Argument bereitgestellten `xs:dateTime`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	**`xs:dateTime`**	Der Input-Wert vom Typ `xs:dateTime`.

6.7.29.30 years-from-duration

Gibt einen `xs:integer`-Wert für die Jahreskomponente in der kanonisch-lexikalischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
duration	**`xs:duration`**	Der Input-Wert vom Typ `xs:duration`.

6.7.30 xpath2 | node functions (Node-Funktionen)

Die Node-Funktionen aus der **xpath2**-Bibliothek liefern Informationen über Nodes (Datenelemente) in einer Mapping-Komponente.

Die **lang** Funktion erhält ein String-Argument, das einen Sprachcode definiert (wie z.B. "en"). Die Funktion gibt je nachdem, ob der Kontext-Node ein **xml:lang** Attribut mit einem Wert hat, der mit dem Argument der Funktion übereinstimmt, entweder **true** oder `false` zurück.

Die Funktionen **local-name**, **name** und **namespace-uri** geben den lokalen Namen, den Namen bzw. die Namespace URI des Input-Node zurück. So ist z.B. beim Node **altova:Products** der lokale Name **Products**, der Name **altova:Products** und die Namespace URI die URI des Namespace, an den das Präfix altova: gebunden ist (siehe Beispiel zur Funktion **local-name**[1216]). Jede dieser drei Funktionen hat zwei Varianten:

- ohne Argument: In diesem Fall wird die Funktion auf den Kontext-Node (ein Beispiel für einen Kontext-Node finden Sie im Beispiel oben zur **lang**[1214] Funktion) angewendet.
- mit einem Argument, das ein Node sein muss: Die Funktion wird auf den verbundenen Node angewendet.

Die **number**-Funktion erhält einen Node als Input, zerlegt den Node in seine Bestandteile (d.h. extrahiert seinen Inhalt), konvertiert den Wert in eine Dezimalzahl und gibt den konvertierten Wert zurück. Die **number**-Funktion hat zwei Varianten:

- ohne Argument: In diesem Fall wird die Funktion auf den Kontext-Node (ein Beispiel für einen Kontext-Node finden Sie im Beispiel oben zur **lang**[1214] Funktion) angewendet.
- mit einem Argument, das ein Node sein muss: Die Funktion wird auf den verbundenen Node angewendet.

6.7.30.1 lang

Gibt **true** zurück, wenn der Kontext-Node ein `xml:lang`-Attribut mit einem Wert hat, der entweder genau mit dem Argument **testlang** übereinstimmt oder eine Untergruppe davon ist. Andernfalls gibt die Funktion **false** zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
testlang	`xs:string`	Der zu überprüfende Sprachencode, z.B. "en".

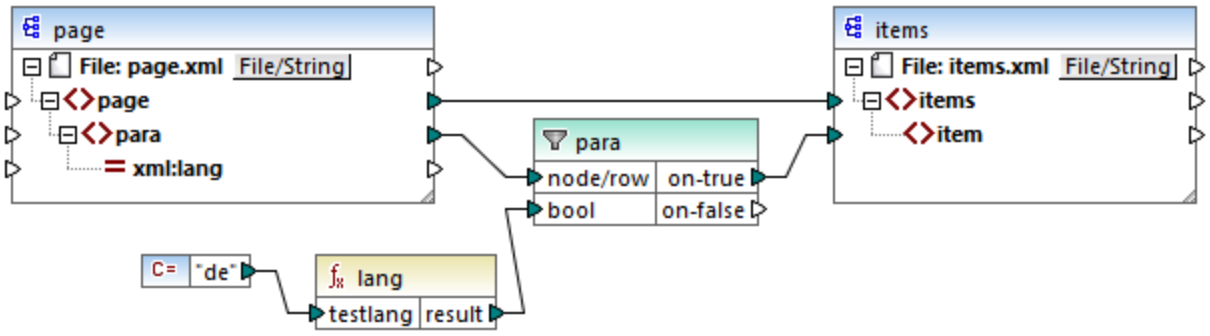
© 2016-2022 Altova GmbH

Beispiel

Die folgende XML-Datei enthält **para**-Elemente mit verschiedenen Werten für das Attribut `xml:lang`.

```
<page>
   <para xml:lang="en">Good day!</para>
   <para xml:lang="fr">Bonjour!</para>
   <para xml:lang="de-AT">Grüss Gott!</para>
   <para xml:lang="de-DE">Guten Tag!</para>
   <para xml:lang="de-CH">Grüezi!</para>
</page>
```

Im unten gezeigten Mapping werden mit Hilfe der Funktion **lang** unabhängig von der Landesvariante nur die deutschen Absätze gefiltert.



XSLT 2.0-Mapping

Im obigen Mapping wird für jedes **para**-Element in der Quellkomponente auf Basis von Bedingungen ein **item**Element in der Zielkomponente erstellt. Die Bedingung wird durch einen Filter bereitgestellt, der nur die Nodes an die Zielkomponente übergibt, für die die Funktion **lang** **true**, zurückgibt. D.h. nur die Nodes, deren `xml:lang`-Attribut auf "de" (oder eine Untergruppe von "de") gesetzt ist, erfüllen die Filterbedingung. Die Mapping-Ausgabe sieht daher folgendermaßen aus:

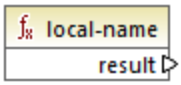
```
<items>
   <item>Grüss Gott!</item>
   <item>Guten Tag!</item>
   <item>Grüezi!</item>
</items>
```

Beachten Sie, dass die **lang**-Funktion aufgrund der Parent-Verbindung zwischen **para** und **item** im Kontext der einzelnen **para**-Elemente ausgeführt wird, siehe auch Der Mapping-Kontext 1333.

6.7.30.2 local-name

Gibt den lokalen Teil des Namens des Kontext-Node als `xs:string` zurück. Dies ist eine parameterlose Variante der **local-name**-Funktion, bei der der Kontext-Node durch die Verbindungen in Ihrem Mapping

festgelegt wird. Um einen Node explizit zu definieren, verwenden Sie die Funktion `local-name`[1216], die einen Input-Node als Parameter erhält.

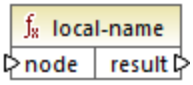


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

6.7.30.3 local-name

Gibt den lokalen Teil des Namens des **Node** als `xs:string` zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
node	`node()`	Der Input-Node.

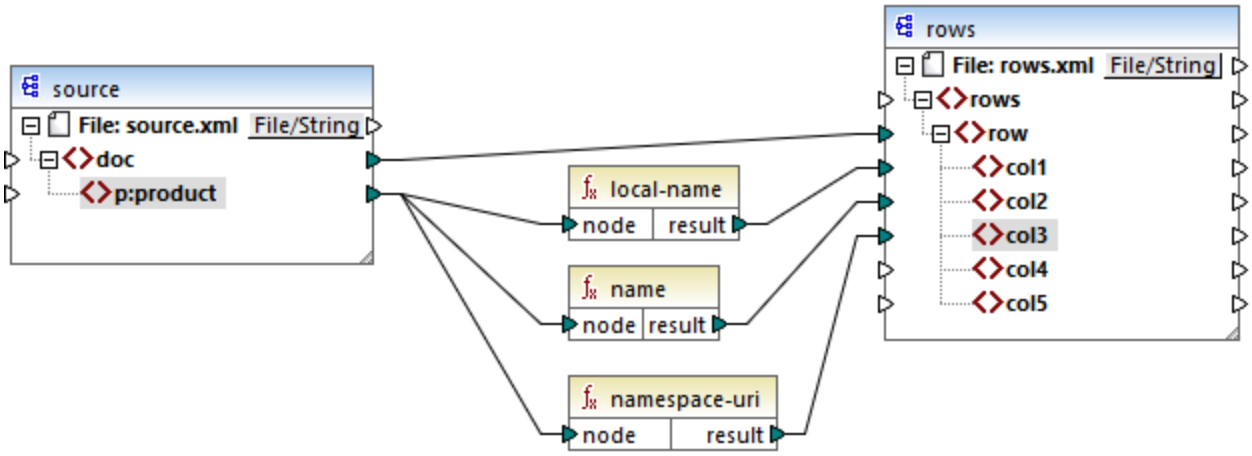
Beispiel

In der folgenden XML-Datei ist der Name des Elements `p:product` ein qualifizierter Name (QName) mit einem Präfix. Das Präfix "p" ist auf den Namespace "http://mycompany.com" gemappt.

```
<?xml version="1.0" encoding="UTF-8"?>
<doc xmlns:p="http://mycompany.com" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="source.xsd">
   <p:product/>
</doc>
```

Im folgenden Mapping werden der lokale Name, der Name und die Namespace URI des Node extrahiert und diese Werte werden in eine Zieldatei geschrieben:

© 2016-2022 Altova GmbH



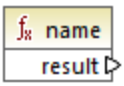
XSLT 2.0-Mapping

Weiter unten sehen Sie das Ergebnis des Mappings. In den einzelnen **col**-Datenelementen wird das Ergebnis der Funktion **local-name**, **name** bzw. **namespace-uri** aufgelistet.

```
<rows>
   <row>
      <col1>product</col1>
      <col2>p:product</col2>
      <col3>http://mycompany.com</col3>
   </row>
</rows>
```

6.7.30.4 name

Gibt den Namen des Kontext-Node zurück. Dies ist eine parameterlose Variante der **name**-Funktion, bei der der Kontext-Node durch die Verbindungen in Ihrem Mapping festgelegt wird. Um einen Node explizit zu definieren, verwenden Sie die Funktion **name**[1218], die einen Input-Node als Parameter erhält.

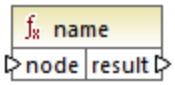


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

6.7.30.5 name

Gibt den Namen eines Node zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
node	`node()`	Der Input-Node.

Beispiel

Siehe das Beispiel zur Funktion `local-name`[1216].

6.7.30.6 namespace-uri

Gibt die Namespace URI des QName des Kontext-Node als `xs:string` zurück. Dies ist eine parameterlose Variante der `namespace-uri`-Funktion, bei der der Kontext-Node durch die Verbindungen in Ihrem Mapping festgelegt wird. Um einen Node explizit zu definieren, verwenden Sie die Funktion `namespace-uri`[1218], die einen Input-Node als Parameter erhält.

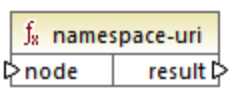


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

6.7.30.7 namespace-uri

Gibt die Namespace URI des QName von **node** als `xs:string` zurück.



© 2016-2022 Altova GmbH

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
node	`node()`	Der Input-Node.

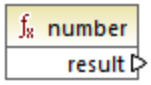
Beispiel

Siehe das Beispiel zur Funktion **`local-name`**[1216].

6.7.30.8 number

Gibt den in einen `xs:double`-Typ konvertierten Wert des Kontext-Node zurück. Dies ist eine parameterlose Variante der **`number`**-Funktion, bei der der Kontext-Node durch die Verbindungen in Ihrem Mapping festgelegt wird. Um einen Node explizit zu definieren, verwenden Sie die Funktion **`number`**[1219], die einen Input-Node als Parameter erhält.

Die einzigen Typen, die in Zahlen konvertiert werden können, sind Boolesche Werte, numerische Strings und andere numerische Typen. Nicht numerische Input-Werte (wie z.B. ein nicht numerischer String) führen zum Resultat NaN (Not a Number).

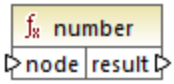


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

6.7.30.9 number

Gibt den in einen `xs:double`-Typ konvertierten Wert von **node** zurück. Die einzigen Typen, die in Zahlen konvertiert werden können, sind Boolesche Werte, numerische Strings und andere numerische Typen. Nicht numerische Input-Werte (wie z.B. ein nicht numerischer String) führen zum Resultat NaN (Not a Number).



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

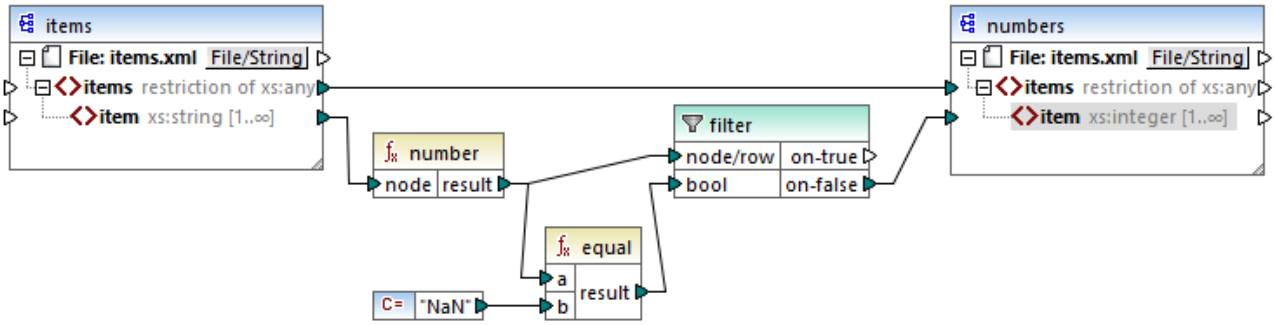
Name	Typ	Beschreibung
node	`mf:atomic`	Der Input-Node.

Beispiel

Die folgende XML-Datei enthält Datenelemente vom Typ `string`:

```
<items>
   <item>1</item>
   <item>2</item>
   <item>Jingle Bells</item>
</items>
```

Im unten gezeigten Mapping wird versucht, alle diese Strings in numerische Werte zu konvertieren und diese in eine XML-Zieldatei zu schreiben. Beachten Sie, dass der Datentyp von **item** in der XML-Zielkomponente `xs:integer` ist, während das Quelldatenelement **item** den Datentyp `xs:string` hat. Wenn die Konvertierung nicht erfolgreich war, muss das Datenelement übersprungen werden und darf nicht in die Zieldatei kopiert werden.



XSLT 2.0-Mapping

Um das gewünschte Mapping-Resultat zu erhalten, wurde ein Filter verwendet. Mit der Funktion **`equal`** wird überprüft, ob das Ergebnis der Konvertierung "NaN" ist. Wenn dies false ist, weist dies darauf hin, dass die Konvertierung erfolgreich war. Daher wird das Datenelement in die Zielkomponente kopiert. Des Ergebnis des Mappings sieht folgendermaßen aus:

```
<items>
   <item>1</item>
   <item>2</item>
</items>
```

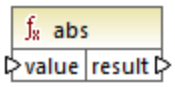

© 2016-2022 Altova GmbH

6.7.31 xpath2 | numeric functions

Zu den numerischen Funktionen der **xpath2**-Bibliothek gehören die Funktionen **abs** und **round-half-to-even**.

6.7.31.1 abs

Gibt den absoluten Wert des Arguments zurück. Wenn das Input-Argument z.B. **-2** oder **2** ist, gibt die Funktion **2** zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

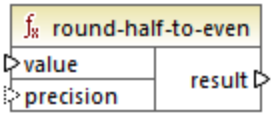
Parameter

Name	Typ	Beschreibung
value	**xs:decimal**	Der Input-Wert.

6.7.31.2 round-half-to-even

Die **round-half-to-even** Funktion rundet die bereitgestellte Zahl (das erste Argument) auf den Präzisionsgrad (Anzahl der Dezimalstellen) auf bzw. ab, der im optionalen zweiten Argument definiert ist. Wenn z.B. das erste Argument **2,141567** und das zweite Argument **3** ist, dann wird das erste Argument (die Zahl) auf drei Dezimalstellen gerundet, d.h. das Ergebnis ist **2,142**. Wenn kein Präzisionsgrad (zweites Argument) angegeben ist, wird die Zahl auf null Dezimalstellen, also eine Ganzzahl gerundet.

"even" im Funktionsnamen bezieht sich auf die Rundung auf eine gerade Zahl, wenn eine Ziffer in einer Zahl sich genau in der Mitte zwischen zwei Werten befindet. `round-half-to-even(3,475, 2)` ergäbe z.B. **3,48**.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

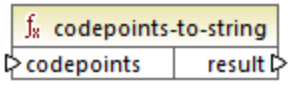
Name	Typ	Beschreibung
value	`xs:decimal`	Obligatorisches Argument, das den zu rundenden Input-Wert bereitstellt.
precision	`xs:integer`	Optionales Argument, das die Anzahl der Dezimalstellen angibt, auf die gerundet werden soll. Der Standardwert ist **0**.

6.7.32 xpath2 | string functions (String-Funktionen)

Mit Hilfe der String-Funktionen der **xpath2**-Bibliothek können Sie Strings verarbeiten (dazu gehören der Vergleich von Strings, die Konvertierung von Strings in Groß- und Kleinbuchstaben, die Extraktion von Substrings aus Strings und andere).

6.7.32.1 codepoints-to-string

Erstellt anhand einer Sequenz von Unicode-Codepoints einen String. Diese Funktoin ist das Gegenteil der Funktion `string-to-codepoints` 1231.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
codepoints	`ZeroOrMore xs:integer`	Dieser Input muss mit einer Sequenz von Datenelementen vom Typ Ganzzahl verbunden werden, wobei jede Ganzzahl einen Unicode-Codepoint definiert.

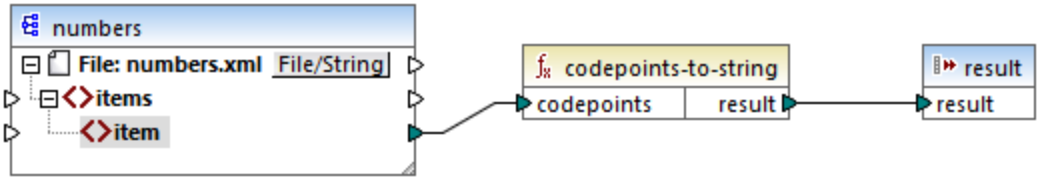
Beispiel

Die folgende XML-Datei enthält mehrere **item**-Elemente, in denen jeweils Unicode-Codepoint-Werte gespeichert sind.

© 2016-2022 Altova GmbH

```
<items>
   <item>77</item>
   <item>97</item>
   <item>112</item>
   <item>70</item>
   <item>111</item>
   <item>114</item>
   <item>99</item>
   <item>101</item>
</items>
```

Im unten gezeigten Mapping wird die Sequenz von Datenelementen für die Funktion **codepoint-to-string** bereitgestellt.



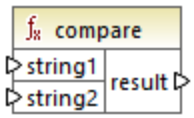
XSLT 2.0-Mapping

Die Mapping-Ausgabe ist **MapForce**.

6.7.32.2 compare

Die **compare**-Funktion erhält zwei Strings als Argumente und vergleicht diese alphabetisch und überprüft, ob diese identisch sind. Wenn **string1** im Alphabet vor **string2** (z.B. bei zwei Strings A und B) vorkommt, dann gibt die Funktion **-1** zurück. Wenn die beiden Strings gleich sind (z.B. A und A), gibt die Funktion **0** zurück. Wenn **string1** im Alphabet nach **string2** (z.B. bei zwei Strings B und A) vorkommt, dann gibt die Funktion **1** zurück.

In dieser Variante der Funktion wird die Standard-Collation, also Unicode verwendet. Es gibt eine weitere Variante[1224] dieser Funktion, bei der Sie die Collation als Argument angeben können.



Sprachen

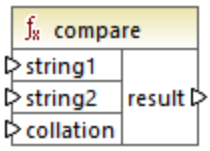
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
string1	`xs:string`	Der erste Input-String.
string2	`xs:string`	Der zweite Input-String.

6.7.32.3 compare

Die `compare`-Funktion erhält zwei Strings als Argumente und vergleicht diese unter Verwendung der als Argument bereitgestellten Collation alphabetisch und überprüft, ob diese identisch sind. Wenn **string1** im Alphabet vor **string2** (z.B. bei zwei Strings A und B) vorkommt, dann gibt die Funktion **-1** zurück. Wenn die beiden Strings gleich sind (z.B. A und A), gibt die Funktion **0** zurück. Wenn **string1** im Alphabet nach **string2** (z.B. bei zwei Strings B und A) vorkommt, dann gibt die Funktion **1** zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

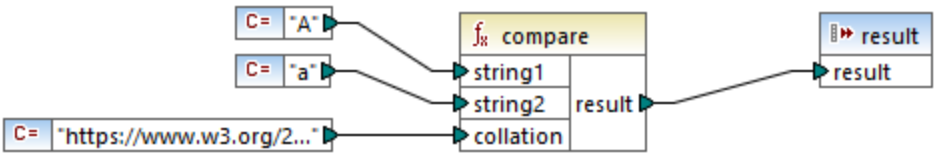
Parameter

Name	Typ	Beschreibung
string1	`xs:string`	Der erste Input-String.
string2	`xs:string`	Der zweite Input-String.
collation	`xs:string`	Definiert, welche Collation für den String-Vergleich verwendet werden soll. Dieser Input kann aus der Ausgabe der Funktion **`default-collation`**[1196] stammen oder kann eine Collation wie z.B. `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive` sein.

© 2016-2022 Altova GmbH

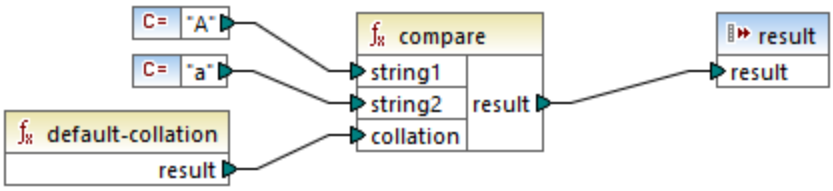
Beispiel

Im folgenden Mapping werden die Strings "A" und "a" mit Hilfe der durch eine Konstante bereitgestellten Collation `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`, in der die Groß- und Kleinschreibung keine Rolle spielt, verglichen.



XSLT 2.0 Mapping

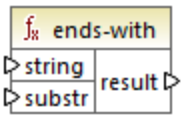
Das Ergebnis des obigen Mappings ist **0**, was bedeutet, dass beide Strings als identisch betrachtet werden. Wenn Sie die Collation jedoch durch die durch die Funktion **`default-collation`** bereitgestellte Collation ersetzen, wird stattdessen die Standard-Unicode Codepoint Collation verwendet und das Ergebnis des Mappings ist **-1** ("A" kommt im Alphabet vor "a").



6.7.32.4 ends-with

Gibt **true** zurück, wenn **string** mit **substr** endet; gibt andernfalls **false** zurück. Der Rückgabewert hat den Typ `xs:boolean`.

In dieser Variante der Funktion wird die Standard-Collation, also Unicode verwendet. Es gibt eine weitere Variante[1226] dieser Funktion, bei der Sie die Collation als Argument angeben können.



Sprachen

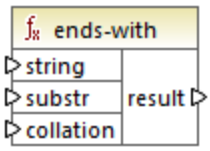
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String (d.h. der "Heuhaufen").
substr	`xs:string`	Der Substring (d.h. die "Nadel").

6.7.32.5 ends-with

Gibt **true** zurück, wenn **string** mit **substr** endet; gibt andernfalls **false** zurück. Der Rückgabewert hat den Typ `xs:boolean`.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

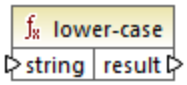
Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String (d.h. der "Heuhaufen").
substr	`xs:string`	Der Substring (d.h. die "Nadel").
collation	`xs:string`	Definiert, welche Collation für den String-Vergleich verwendet werden soll. Dieser Input kann aus der Ausgabe der Funktion **`default-collation`**[1196] stammen oder kann eine Collation wie z.B. `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive` sein.

© 2016-2022 Altova GmbH

6.7.32.6 lower-case

Gibt den Wert von **string** nach Konvertierung der einzelnen Zeichen in Kleinbuchstaben zurück.



Sprachen

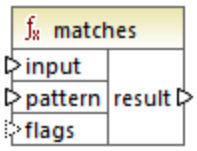
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-Wert.

6.7.32.7 matches

Die `matches` Funktion überprüft, ob ein bereitgestellter String (das erste Argument) einer Regular Expression (dem zweiten Argument) entspricht. Die Syntax der Regular Expression muss die Syntax sein, die für das `pattern` Facet von XML-Schema definiert wurde. Die Funktion gibt **true** zurück, wenn der String der Regular Expression entspricht, und andernfalls **false**.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

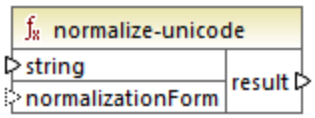
Parameter

Name	Typ	Beschreibung
input	`xs:string`	Der Input-String.
pattern	`xs:string`	Die Regular Expression, der der String entsprechen muss, siehe Regular Expressions 1020.
flags	`xs:string`	Optionales Argument, das die Übereinstimmung beeinflusst. In diesem Argument kann jede beliebige Kombination der folgenden Flags angegeben werden: `i`, `m`, `s`, `x`. Es können mehrere Flags

Name	Typ	Beschreibung
		verwendet werden, z.B. `imx`. Wenn kein Flag verwendet wird, werden die Standardwerte aller vier Flags verwendet. Es gibt die folgenden vier Flags: `i` Modus "Groß/Kleinschreibung wird nicht berücksichtigt" verwenden. Die Standardeinstellung ist "Groß-/Kleinschreibung berücksichtigen". `m` Mehrzeiligen Modus verwenden. In diesem Modus wird der Input-String als mehrzeilig betrachtet, wobei jede Zeile durch ein newline-Zeichen (`x0a`) getrennt wird. Die Metazeichen **^** und **$** kennzeichnen den Beginn und das Ende der einzelnen Zeilen. Die Standardeinstellung ist der String-Modus, in dem der String mit den Metazeichen **^** und **$** beginnt und endet. `s` dot-all Modus verwenden. Die Standardeinstellung ist der not-dot-all Modus, in dem das Metazeichen **.** für alle Zeichen mit Ausnahme des newline-Zeichens (`x0a`) steht. Im dot-all Modus steht der Punkt auch für das newline-Zeichen. `x` Whitespaces ignorieren. Standardmäßig werden Whitespace-Zeichen nicht ignoriert.

6.7.32.8 normalize-unicode

Gibt den Wert von **string** zurück, der entsprechend den Regeln des definierten Normalisierungsprotokolls (zweites Argument) normalisiert ist. Nähere Informationen zur Unicode-Normalisierung siehe §2.2 von https://www.w3.org/TR/charmod-norm/.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

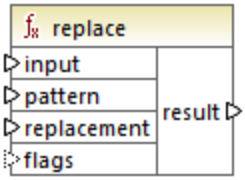
Name	Typ	Beschreibung
string	`xs:string`	Der zu normalisierende Stringwert.

© 2016-2022 Altova GmbH

Name	Typ	Beschreibung
normalizationForm	`xs:string`	Optionales Argument, das das Normalisierungsprotokoll angibt. Das Standardprotokoll ist Unicode Normalization Form C (NFC). Es werden die Normalisierungsprotokolle NFC, NFD, NFKC und NFKD unterstützt.

6.7.32.9 replace

Diese Funktion erhält einen Input-String, eine Regular Expression und einen Ersetzungsstring als Argumente. Sie ersetzt alle Übereinstimmungen mit der Regular Expression im Input-String durch den Ersetzungsstring. Wenn zwei einander überlappende Strings im Input-String mit der Regular Expression übereinstimmen, wird nur die erste Entsprechung ersetzt.



Sprachen

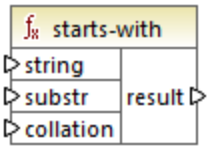
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
input	`xs:string`	Der Input-String.
pattern	`xs:string`	Die Regular Expression, der der String entsprechen muss, siehe Regular Expressions 1020.
replacement	`xs:string`	Der Ersetzungsstring.
flags	`xs:string`	Optionales Argument, das die Übereinstimmung beeinflusst. Dieses Argument wird auf dieselbe Art, wie das Argument **flags** in der Funktion `matches` 1227 verwendet.

6.7.32.10 starts-with

Gibt **true** zurück, wenn **string** mit **substr** beginnt; gibt andernfalls **false** zurück. Der Rückgabewert hat den Typ `xs:boolean`. Der String-Vergleich erfolgt gemäß der angegebenen Collation.



Sprachen

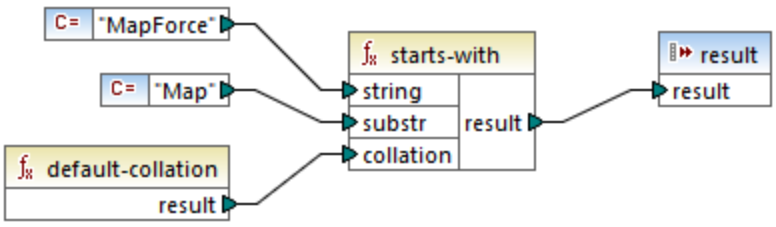
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String (d.h. der "Heuhaufen").
substr	`xs:string`	Der Substring (d.h. die "Nadel").
collation	`xs:string`	Definiert, welche Collation für den String-Vergleich verwendet werden soll. Dieser Input kann aus der Ausgabe der Funktion **`default-collation`**[1196] stammen oder kann eine Collation wie z.B. `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive` sein.

Beispiel

Im folgenden Mapping wird der Wert **`true`** zurückgegeben , da der Input-String "MapForce" mit dem Substring "Map" beginnt, vorausgesetzt, es wird die Standard-Unicode Collation verwendet.



© 2016-2022 Altova GmbH

6.7.32.11 string-to-codepoints

Gibt die Sequenz von Unicode Codepoints (Ganzzahlwerte), die den als Argument bereitgestellten String darstellen, zurück. Diese Funktoin ist das Gegenteil der Funktion **`codepoints-to-string`**[1222].



Sprachen

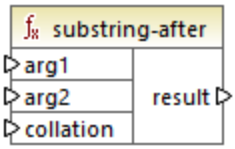
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
input	`xs:string`	Der Input-String

6.7.32.12 substring-after

Gibt den Teil des String **arg1** zurück, der nach dem String **arg2** kommt.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
arg1	`xs:string`	Der Input-String (d.h. der "Heuhaufen").
arg2	`xs:string`	Der Substring (d.h. die "Nadel").
collation	`xs:string`	Definiert, welche Collation für den String-Vergleich verwendet werden soll. Dieser Input kann aus der Ausgabe der Funktion **`default-`**

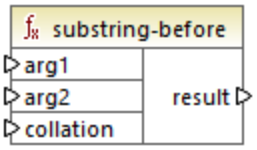
Name	Typ	Beschreibung
		collation[1196] stammen oder kann eine Collation wie z.B. `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive` sein.

Beispiel

Wenn **arg1** "MapForce" ist, **arg2** "Map" und **collation** die **default-collation**[1196] ist, gibt die Funktion "Force" zurück.

6.7.32.13 substring-before

Gibt den Teil des String **arg1** zurück, der vor dem String **arg2** kommt.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
arg1	`xs:string`	Der Input-String (d.h. der "Heuhaufen").
arg2	`xs:string`	Der Substring (d.h. die "Nadel").
collation	`xs:string`	Definiert, welche Collation für den String-Vergleich verwendet werden soll. Dieser Input kann aus der Ausgabe der Funktion **default-collation**[1196] stammen oder kann eine Collation wie z.B. `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive` sein.

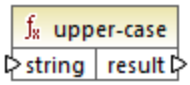
© 2016-2022 Altova GmbH

Beispiel

Wenn **arg1** "MapForce" ist, **arg2** "Force" und **collation** die `default-collation`[1196] ist, gibt die Funktion "Map" zurück.

6.7.32.14 upper-case

Gibt den Wert von **string** nach Konvertierung der einzelnen Zeichen in Großbuchstaben zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

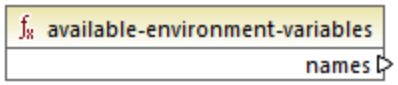
Name	Typ	Beschreibung
string	`xs:string`	Der Input-String.

6.7.33 xpath3 | external information functions

Mit Hilfe der Funktionen für externe Informationen der **xpath3**-Bibliothek können Sie Informationen über die XSLT-Ausführungsumgebung aufrufen oder Daten aus externen Ressourcen abrufen.

6.7.33.1 available-environment-variables

Gibt eine Liste von Umgebungsvariablennamen zurück, die sich für die Übergabe an die `environment-variable`-Funktion in Form einer (möglicherweise leeren) Stringsequenz eignen.



Sprachen

XSLT 3.0.

6.7.33.2 environment-variable

Gibt den Wert einer Systemumgebungsvariablen, falls vorhanden, zurück. Der Rückgabetyp ist `xs:string`.



Sprachen

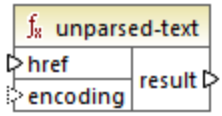
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
name	`xs:string`	Der Name der Umgebungsvariablen.

6.7.33.3 unparsed-text

Liest eine externe Ressource (z.B. eine Datei) und gibt eine String-Darstellung der Ressource zurück.



Sprachen

XSLT 3.0.

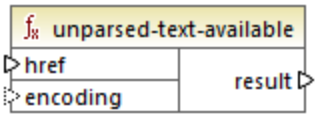
Parameter

Name	Typ	Beschreibung
href	`xs:string`	Ein String in der Form einer URI-Referenz.
encoding	`xs:string`	Optionales Argument. Definiert den Namen der Kodierung, z.B. "UTF-8", "UTF-16". Wenn die Kodierung nicht ermittelt werden kann, wird automatisch UTF-8 angenommen.

© 2016-2022 Altova GmbH

6.7.33.4 unparsed-text-available

Stellt fest, ob ein Aufruf von **`unparsed-text`** mit bestimmten Argumenten erfolgreich wäre. Der Rückgabetyp ist `xs: boolean`.



Sprachen

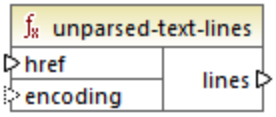
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
href	**`xs:string`**	Ein String in der Form einer URI-Referenz.
encoding	**`xs:string`**	Optionales Argument. Definiert den Namen der Kodierung, z.B. "UTF-8", "UTF-16". Wenn die Kodierung nicht ermittelt werden kann, wird automatisch UTF-8 angenommen.

6.7.33.5 unparsed-text-lines

Liest eine externe Ressource (z.B. eine Datei) und gibt ihren Inhalt als String-Sequenz zurück, und zwar eine für jede Textzeile in der String-Darstellung der Ressource.



Sprachen

XSLT 3.0.

Parameter

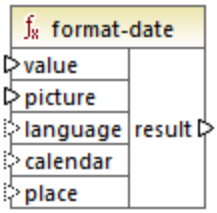
Name	Typ	Beschreibung
href	`xs:string`	Ein String in der Form einer URI-Referenz.
encoding	`xs:string`	Optionales Argument. Definiert den Namen der Kodierung, z.B. "UTF-8", "UTF-16". Wenn die Kodierung nicht ermittelt werden kann, wird automatisch UTF-8 angenommen.

6.7.34 xpath3 | formatting functions

Die Formatierungsfunktionen der **xpath3**-Bibliothek dienen zum Formatieren von Datums-, Uhrzeit und Ganzzahlwerten.

6.7.34.1 format-date

Gibt einen String zurück, der einen für die Anzeige formatierten `xs:date`-Wert enthält.



Sprachen

XSLT 3.0.

Parameter

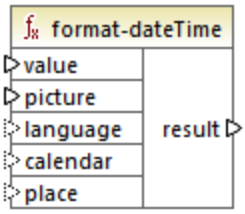
Name	Typ	Beschreibung
value	`xs:date`	Der zu formatierende `xs:date`-Input-Wert. Obligatorischer Parameter.
picture	`xs:string`	Obligatorischer Parameter.

© 2016-2022 Altova GmbH

Name	Typ	Beschreibung
		Siehe Abschnitt 9.8.4.1 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Optionaler Parameter. Siehe Abschnitt 9.8.4.8 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).
calendar	`xs:string`	Wie oben.
place	`xs:string`	Wie oben.

6.7.34.2 format-dateTime

Gibt einen String zurück, der einen für die Anzeige formatierten `xs:dateTime`-Wert enthält.



Sprachen

XSLT 3.0.

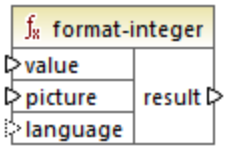
Parameter

Name	Typ	Beschreibung
value	`xs:dateTime`	Der zu formatierende `xs:dateTime`-Input-Wert.
picture	`xs:string`	Obligatorischer Parameter. Siehe Abschnitt 9.8.4.1 der "XPath and XQuery Functions and

Name	Typ	Beschreibung
		Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Optionaler Parameter. Siehe Abschnitt 9.8.4.8 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).
calendar	`xs:string`	Wie oben.
place	`xs:string`	Wie oben.

6.7.34.3 format-integer

Formatiert eine Ganzzahl anhand der Konventionen einer angegebenen natürlichen Sprache (falls angegeben) gemäß einem angegebenen picture-String.



Sprachen

XSLT 3.0.

Parameter

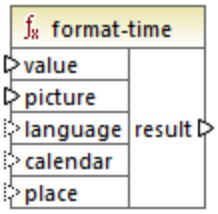
Name	Typ	Beschreibung
value	`xs:integer`	Der zu formatierende Integer-Input-Wert.
picture	`xs:string`	Obligatorischer Parameter. Siehe Abschnitt 4.6.1 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation

© 2016-2022 Altova GmbH

Name	Typ	Beschreibung
		(https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Optionaler Parameter. Definiert die natürliche Sprache, gemäß der der Wert formatiert werden soll. Falls angegeben, muss es sich bei diesem Wert entweder um eine leeren String oder einen beliebigen für das `xml:lang`-Attribut gemäß der "Extensible Markup Language (XML) 1.0 W3C Recommendation (https://www.w3.org/TR/xml) zulässigen Wert handeln.

6.7.34.4 format-time

Gibt einen String zurück, der einen für die Anzeige formatierten `xs:time`-Wert enthält.



Sprachen

XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:time`	Der zu formatierende `xs:time`-Input-Wert.
picture	`xs:string`	Obligatorischer Parameter. Siehe Abschnitt 9.8.4.1 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation

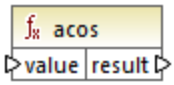
Name	Typ	Beschreibung
		(https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Optionaler Parameter. Siehe Abschnitt 9.8.4.8 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).
calendar	`xs:string`	Wie oben.
place	`xs:string`	Wie oben.

6.7.35 xpath3 | math functions

Mit Hilfe der math-Funktionen der **xpath3**-Bibliothek können Sie trigonometrische und andere mathematische Berechnungen durchführen.

6.7.35.1 acos

Gibt den Arkuscosinus eines Winkels im Bereich von **0** bis **pi** zurück.



Sprachen

XSLT 3.0.

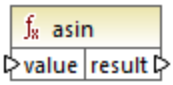
Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

© 2016-2022 Altova GmbH

6.7.35.2 asin

Gibt den ArkusSinus eines Winkels im Bereich von **-pi/2** bis **pi/2** zurück.



Sprachen

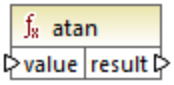
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.35.3 atan

Gibt den Arkustangens eines Winkels im Bereich von **-pi/2** bis **pi/2** zurück.



Sprachen

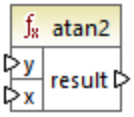
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.35.4 atan2

Gibt den Winkel in Bogenmaß zwischen einem Punkt auf einer Fläche mit den Koordinaten (x,y) und der positiven X-Achse zurück.



Sprachen

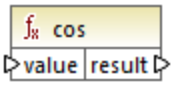
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
y	`xs:double`	Die x-Koordinate.
x	`xs:double`	The y-Koordinate.

6.7.35.5 cos

Gibt den trigonometrischen Cosinus des durch value angegebenen Winkels zurück. Die Werteinheit ist radians.



Sprachen

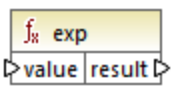
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.35.6 exp

Gibt die Eulersche Zahl *e* hoch value zurück.



© 2016-2022 Altova GmbH

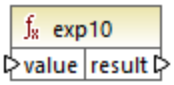
Sprachen
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.35.7 exp10

Gibt 10 hoch value zurück.



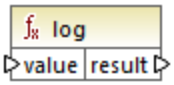
Sprachen
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.35.8 log

Gibt den natürlichen Logarithmus (Basis e) eines Werts zurück.



Sprachen
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.35.9 log10

Gibt den dekadischen Logarithmus (Basis 10) eines Werts zurück.



Sprachen

XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.35.10 pi

Gibt einen Näherungswert an die mathematische Konstante **Pi** zurück.

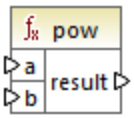


Sprachen

XSLT 3.0.

6.7.35.11 pow

Gibt den Wert von **a** hoch **b** zurück.



Sprachen

XSLT 3.0.

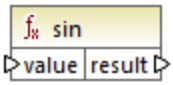
© 2016-2022 Altova GmbH

Parameter

Name	Typ	Beschreibung
a	`xs:double`	Der Input-Wert **a**.
b	`xs:double`	Der Input-Wert **b**.

6.7.35.12 sin

Gibt den trigonometrischen Sinus des durch value angegebenen Winkels zurück. Die Werteinheit ist Radians.



Sprachen

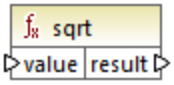
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.35.13 sqrt

Gibt die nicht negative Quadratwurzel des Arguments zurück.



Sprachen

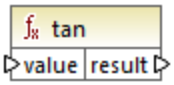
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.35.14 tan

Gibt den trigonometrischen Tangens des durch value angegebenen Winkels zurück. Die Werteinheit ist Radians.



Sprachen

XSLT 3.0.

Parameter

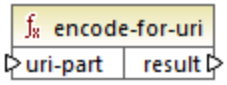
Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

6.7.36 xpath3 | URI functions

Die URI-Funktionen in der **xpath3**-Bibliothek dienen zum Kodieren, mit Escape-Zeichen Versehen und Konvertieren von Werte, die in URIs zur Verwendung kommen sollen.

6.7.36.1 encode-for-uri

Kodiert reservierte Zeichen in einem String, die im Pfadsegment einer URI verwendet werden sollen. Nähere Informationen zu dieser Funktion finden Sie im Abschnitt 6.2 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).



Sprachen

XSLT 3.0.

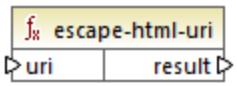
Parameter

Name	Typ	Beschreibung
uri-part	`xs:string`	Der zu kodierende URI-Input-Wert.

© 2016-2022 Altova GmbH

6.7.36.2 escape-html-uri

Versieht eine URI auf dieselbe Art, auf die auch HTML-Benutzer-Agenten Attributwerte behandeln, die wahrscheinlich URIs enthalten, mit Escape-Zeichen. Nähere Informationen zu dieser Funktion finden Sie im Abschnitt 6.4 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).



Sprachen

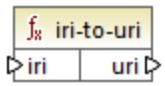
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
uri	`xs:string`	Der mit Escape-Zeichen zu versehende URI-Input-Wert.

6.7.36.3 iri-to-uri

Konvertiert einen String, der einen IRI (Internationalized Resource Identifier) enthält in eine URI (Uniform Resource Identifier). Nähere Informationen zu dieser Funktion finden Sie im Abschnitt 6.3 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).



Sprachen

XSLT 3.0.

Parameter

Name	Typ	Beschreibung
iri	`xs:string`	Der IRI-Input-Wert.

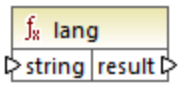
6.7.37 xslt | xpath functions

Die Funktionen in dieser Untergruppe sind XPath 1.0-Funktionen, die Informationen über Mapping-Datenelemente (oder Nodes) zurückgeben. Die meisten dieser Funktionen erhalten einen Node als Argument und geben Informationen über diesen Node zurück. Die Funktionen **last** und **position** werden im aktuellen Mapping-Kontext[1333], der durch die Verbindungen in Ihrem Mapping bestimmt wird, ausgeführt.

Anmerkung: Weitere XPath 1.0-Funktionen finden Sie in der **core**-Bibliothek.

6.7.37.1 lang

Gibt **true** zurück, wenn der Kontext-Node ein `xml:lang`-Attribut mit einem Wert hat, der entweder genau mit dem Argument **string** übereinstimmt oder eine Untergruppe davon ist. Andernfalls gibt die Funktion **false** zurück.



Sprachen

XSLT 1.0.

Parameter

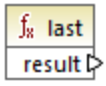
Name	Typ	Beschreibung
string	`xs:string`	Der zu überprüfende Sprachencode, z.B. "en".

Beispiel

Siehe das Beispiel unter der Funktion **lang**[1214] der **xpath2**-Bibliothek.

6.7.37.2 last

Gibt die Position des letzten Node in der Liste der verarbeiteten Nodes zurück.



Sprachen

XSLT 1.0.

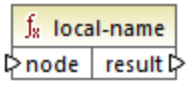
© 2016-2022 Altova GmbH

Beispiel

Siehe das Beispiel unter der Funktion **last** 1197 der **xpath2**-Bibliothek.

6.7.37.3 local-name

Gibt den lokalen Teil des Namens des als Argument bereitgestellten Node zurück.



Sprachen

XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

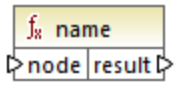
Name	Typ	Beschreibung
node	`node()`	Der Input-Node.

Beispiel

Siehe das Beispiel unter der Funktion **local-name** 1216 der **xpath2**-Bibliothek.

6.7.37.4 name

Gibt den Namen des als Argument bereitgestellten Node zurück.



Sprachen

XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

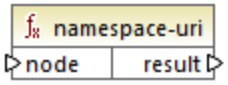
Name	Typ	Beschreibung
node	`node()`	Der Input-Node.

Beispiel

Siehe das Beispiel unter der Funktion **local-name** 1216 der **xpath2**-Bibliothek.

6.7.37.5 namespace-uri

Gibt die Namespace URI des als Argument bereitgestellten Node zurück.



Sprachen

XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

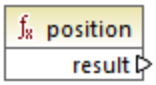
Name	Typ	Beschreibung
node	node()	Der Input-Node.

Beispiel

Siehe das Beispiel unter der Funktion **local-name** 1216 der **xpath2**-Bibliothek.

6.7.37.6 position

Gibt die Position des aktuellen Node im gerade verarbeiteten Nodeset zurück.



Sprachen

XSLT 1.0.

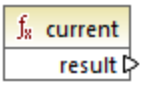
6.7.38 xslt | xslt function (XSLT-Funktionen)

Bei den Funktionen in dieser Gruppe handelt es sich um diverse XSLT 1.0-Funktionen.

© 2016-2022 Altova GmbH

6.7.38.1 current

Die **current**-Funktion erhält kein Argument und gibt den aktuellen Node zurück.

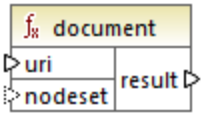


Sprachen

XSLT 1.0.

6.7.38.2 document

Ruft Nodes aus einem externen XML-Dokument auf. Das Ergebnis wird in einen Node im Ausgabedokument ausgegeben.



Sprachen

XSLT 1.0.

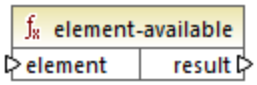
Parameter

Name	Typ	Beschreibung
uri	`xs:string`	Obligatorisch. Definiert den Pfad zum XML-Dokument. Das XML-Dokument muss gültig und parsebar sein.
nodeset	`node()`	Optional. Definiert einen Node, anhand dessen Basis-URI die URI aufgelöst wird, die als erstes Argument bereitgestellt wird, wenn diese URI relativ ist.

6.7.38.3 element-available

Die **element-available** Funktion überprüft, ob ein Element, das als das einzige String-Argument der Funktion bereitgestellt wird, vom XSLT-Prozessor unterstützt wird. Der Argument-String wird als QName ausgewertet. Daher müssen XSLT-Elemente ein `xsl:`-Präfix und XML-Schema-Elemente ein `xs:`-Präfix haben, da dies die

Präfixe sind, die im zugrunde liegenden XSLT-Dokument, das für das Mapping erstellt wird, deklariert sind. Die Funktion gibt einen Booleschen Wert zurück.



Sprachen

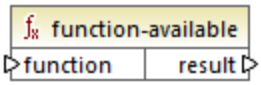
XSLT 1.0.

Parameter

Name	Typ	Beschreibung
element	`xs:string`	Der Elementname.

6.7.38.4 function-available

Die **`function-available`**-Funktion ähnelt der **`element-available`**-Funktion und überprüft, ob der als Argument der Funktion bereitgestellte Funktionsname vom XSLT-Prozessor unterstützt wird. Der Input-String wird als QName ausgewertet. Die Funktion gibt einen Booleschen Wert zurück.



Sprachen

XSLT 1.0.

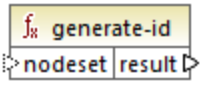
Parameter

Name	Typ	Beschreibung
function	`xs:string`	Der Funktionsname.

6.7.38.5 generate-id

Die **`generate-id`**-Funktion generiert einen eindeutigen String, der den ersten Node des Nodeset anhand des optionalen Input-Arguments identifiziert. Wenn kein Argument bereitgestellt wird, wird die ID am Kontext-Node generiert. Das Ergebnis kann an jeden Node im Ausgabedokument gerichtet werden.

© 2016-2022 Altova GmbH



Sprachen

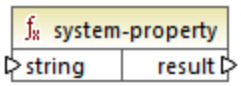
XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

Name	Typ	Beschreibung
nodeset	`node()`	Optionales Argument, das den Input-Node bereitstellt.

6.7.38.6 system-property

Die **`system-property`**-Funktion gibt die Eigenschaften des XSLT-Prozessors (des Systems) zurück. Drei Systemeigenschaften, alle im XSLT-Namespace, sind bei XSLT-Prozessoren obligatorisch, nämlich `xsl:version`, `xsl:vendor` und `xsl:vendor-url`. Der Input-String wird als QName ausgewertet und muss daher das Präfix `xsl:` haben, da dies das Präfix ist, das im zugrunde liegenden XSLT-Stylesheet mit dem XSLT-Namespace verknüpft ist.



Sprachen

XSLT 1.0, XSLT 2.0, XSLT 3.0.

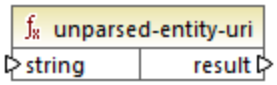
Parameter

Name	Typ	Beschreibung
string	`xs:string`	Definiert den Eigenschaftsnamen. Dieser kann einer der folgenden sein: `xsl:version`, `xsl:vendor`, `xsl:vendor-url`.

6.7.38.7 unparsed-entity-uri

Wenn Sie eine DTD verwenden, können Sie darin eine ungeparste Entity deklarieren. Diese ungeparste Entity, z.B. ein Bild, hat eine URI, die den Pfad zur Entity angibt. Der Input-String der Funktion muss dem Namen der

in der DTD deklarierten ungeparsten Entity entsprechen, dann gibt die Funktion die URI der ungeparsten Entity zurück. Diese URI kann dann an einen Node im Ausgabedokument, z.B. an einen **href** Node gerichtet werden.



Sprachen

XSLT 1.0.

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Name der ungeparsten Entity, deren URI abgerufen werden soll.

© 2016-2022 Altova GmbH

7 SOAP-Webservices

In der MapForce Enterprise Edition können Sie Java- oder C#-Programmcode zur Implementierung von SOAP-Webservices auf Basis vorhandener WSDL (Web Services Description Language)-Dateien generieren. Sie können Daten folgendermaßen auf WSDL-Operationen mappen:

- Vom Input der WSDL-Operation auf jede von MapForce unterstützte Datenquelle, wie Flat Files, XML-, XBRL-, EDI-, Microsoft Excel-Dateien und Datenbanken.
- Von den von MapForce unterstützten Datenquellen auf den Output der WSDL-Operation.

MapForce unterstützt WSDL 1.1 und WSDL 2.0 (weitere Informationen zur Unterstützung und zu Einschränkungen finden Sie unter Informationen zur SOAP/WSDL-Unterstützung 1258 und Java SOAP-Webservice-Spezifisches 1272).

Anmerkung: Anstatt SOAP-Webservices anhand von vorhandenen WSDL-Dateien zu entwickeln, können Sie auch MapForce-Mappings auf MapForce Server bereitstellen, wo Sie diese über eine API oder einen Aufruf über die Befehlszeile ausführen können. Zusätzlich dazu können Sie das Mapping mit Hilfe von FlowForce Server als generischen Webservice (nicht als SOAP-Webservice) auf dem Rechner, auf dem FlowForce Server ausgeführt wird, bereitstellen. Jedes Mapping, das für die Transformationssprache BUILT-IN 29 erstellt wurde, kann auf MapForce Server oder FlowForce Server bereitgestellt werden. FlowForce Server ist eine eigene Server-Lösung, mit der MapForce-Mappings nicht nur in generische Webservices konvertiert, sondern auch als geplante Aufträge oder Aufträge bei Bedarf ausgeführt werden können. Nähere Informationen dazu finden Sie unter Bereitstellen von Mappings auf FlowForce Server 1445.

Voraussetzungen

Um mit MapForce einen Web-Dienst erstellen zu können, benötigen Sie die WSDL-Datei des Web-Dienstes. Sie können WSDL-Dateien z.B. mit XMLSpy erstellen und damit auch SOAP Requests testen. Zusätzlich dazu benötigen Sie plattformspezifische Software zum Erstellen, Bereitstellen und Ausführen von SOAP-Webservices, z.B.:

Java

- Java SE JDK 8 oder 11 (einschließlich OpenJDK)
- Apache Tomcat: https://tomcat.apache.org
- Apache Axis2: https://ws.apache.org/axis2/, ein SOAP-Framework, das mit Tomcat ausgeführt wird
- Apache Ant: https://ant.apache.org/

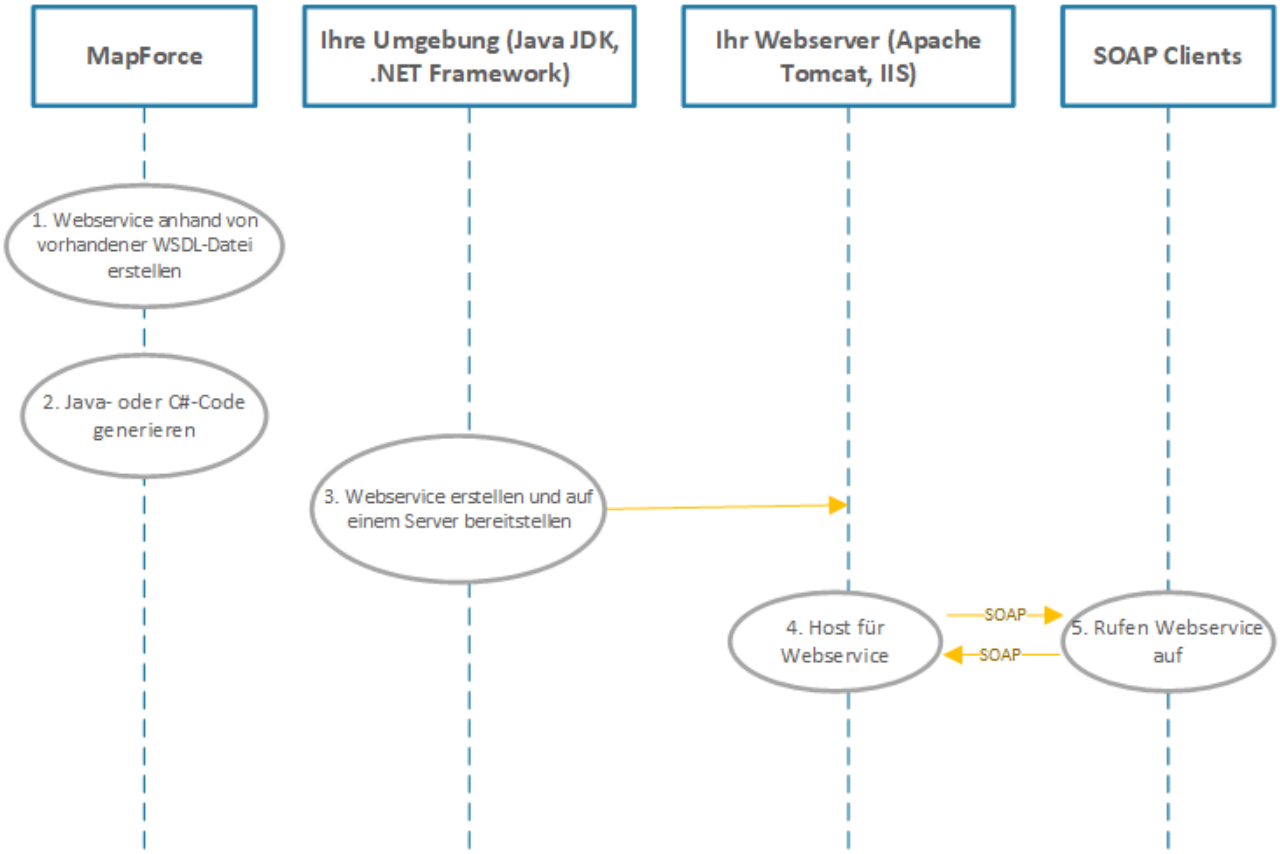
C#

- Microsoft Visual Studio 2013, 2015, 2017, 2019, 2022
- Microsoft Internet Information Services (IIS) Version 5.0 oder höher.

Funktionsweise

Im folgenden Diagramm sehen Sie den gesamten Prozess, angefangen vom Implementieren eines Webservices anhand einer vorhandenen WSDL-Datei bis zu dem Zeitpunkt, zu dem er von SOAP-Client-Applikationen aufgerufen werden kann. Beachten Sie, dass nur die ersten beiden Schritte in MapForce durchgeführt werden

können. Schritt 3 und 4 hängen von den Besonderheiten der Drittanbieter-Tools und Umgebungen ab, die Ihr Unternehmen verwendet.



Der oben dargestellte Vorgang läuft folgendermaßen ab:

Schritt #	Beschreibung
1. Erstellen des Webservice anhand einer vorhandenen WSDL-Datei	Sobald Sie die WSDL-Datei zur Verfügung haben, können Sie ein neues MapForce Webservice-Projekt beginnen. Ein MapForce Webservice-Projekt hat eine vordefinierte Struktur, über die Sie eine bestimmte WSDL-Operation oder einen Web-Dienst darin schnell aufrufen können. Wenn Sie ein MapForce Webservice-Projekt beginnen, können Sie MapForce optional für alle in der WSDL-Datei gefundenen WSDL-Operationen automatisch Mapping-Dateien generieren lassen. Alternativ dazu können Sie die WSDL-Dienste und -Endpoints, die in das MapForce Webservice-Projekt inkludiert werden sollen, einzeln auswählen. Auf der Benutzeroberfläche von MapForce entsprechen WSDL-Operationen Mappings und die Input- und Output-Parameter werden als Komponenten im Mapping angezeigt.

© 2016-2022 Altova GmbH

Schritt #	Beschreibung
	In Mappings, die Inputs oder Outputs von WSDL-Operationen enthalten, können Sie wie bei jedem anderen Mapping eine Vorschau des Ausführungsergebnisses anzeigen. Zu diesem Zweck benötigen Sie eine Beispiel-SOAP-Request-Datei (XML-Format), die der WSDL-Grammatik des Webservice entspricht (solche Instanzdateien können mit XMLSpy generiert werden).
2. Generieren von Java- oder C#-Code	Wenn Ihr Webservice-Projekt in MapForce fertig ist, können Sie (zu Testzwecken) entweder für einzelne Mappings oder für das gesamte Projekt C# oder Java-Code generieren. In letzterem Fall erstellt MapForce einen kompletten Webservice Server in der Sprache Ihrer Wahl (C# oder Java).
3. Erstellen und Bereitstellen des Webservice auf einem Server	In diesem Schritt kompilieren Sie den generierten C#- oder Java-Code (außerhalb von MapForce) und stellen ihn auf Ihrem benutzerdefinierten Webserver bereit. Ihr Webserver muss als Host für SOAP-Dienste fungieren und auf SOAP-Client-Calls antworten, siehe Abschnitt "Voraussetzungen" oben.
4. Hosten des Webservice	Der kompilierte Webservice kann auf dem Server Ihrer Wahl gehostet werden, z.B.: • Apache Tomcat • Internet Information Services (IIS)
5. Aufrufen des Web service	Als Client für den Webservice kann jede Applikation fungieren, die SOAP-Aufrufe generieren und die Antwort verarbeiten kann (z.B. XMLSpy oder MapForce selbst).

Nähere Informationen zu Schritt 1 finden Sie unter Erstellen von Webservice-Projekten anhand von WSDL-Dateien 1261. Informationen zu den anderen Schritten finden Sie unter Beispiel: Generieren von SOAP-Webservices (C#) 1275 und Beispiel: Generieren von SOAP Webservices (Java) 1290.

7.1 Anmerkungen zur SOAP/WSDL-Unterstützung

Die folgende Tabelle enhält eine Übersicht über Einzelheiten zur WSDL-Unterstützung in MapForce.

WSDL-Unterstützung	Version 1.1, W3C Note von http://www.w3.org/TR/wsdl Version 2.0, W3C Recommendation von http://www.w3.org/TR/wsdl20/
WSDL-Typsystem	XML Schema 2001
SOAP-Unterstützung	Version 1.1: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/ Version 1.2: http://www.w3.org/TR/soap12-part0/
Protokolle	SOAP über HTTP (HTTP POST, HTTP GET-Protokolle werden nicht unterstützt).
C#	Der SOAPAction Header muss für jede Operation in C# unterschiedlich sein.
Bindings	Derzeit werden mehrere Operationen mit demselben Namen nicht unterstützt (Abschnitt 2.5 der WSDL 1.1-Spezifikation).
Stil/Verwendung	• Document/literal: unterstützt. • RPC/literal: in C# unterstützt • RPC/encoded: eingeschränkt unterstützt • Derzeit wird ein Stile/eine Verwendung pro Web-Dienst (Java) oder Operation (C#) unterstützt.
SOAP Header	Abhängig von der zugrunde liegenden Plattform
SOAP encodingStyle	Wenn use="encoded", wird als Kodierungsstil für den kompletten `soap:Body` **"http://schemas.xmlsoap.org/soap/encoding/"** angenommen. Andere Kodierungsstile werden nicht unterstützt. Das Attribut `encodingStyle` wird in Messages ignoriert (Abschnitt 4.1.1 der SOAP 1.1-Spezifikation).
Referenzen	• Verweise auf externe Ressourcen werden derzeit nicht unterstützt (Abschnitt 5.4.1 der SOAP 1.1-Spezifikation). • Verweise auf unanhängige Elemente werden unterstützt
SOAP-ENC:Array	Linearer Zugriff unterstützt; partial arrays, sparse arrays werden derzeit nicht unterstützt.
Benutzerdefinierte SOAP-Verbesserungen:	Nicht unterstützt.
Standardwerte oder fixe Werte in Schemas	Nicht unterstützt.
Validierung von Nicht-SOAP Messages	Keine Validierung; Messages werden an das zugrunde liegende Framework übergeben.

© 2016-2022 Altova GmbH

Namespaces	Nicht-Namespace-Einträge sind ungültiger WSDL-Code und werden daher nicht unterstützt (WSDL und XML 1.0)

WSDL 1.1

portType
Ein <portType> Element definiert eine Webservice-Schnittstelle, d.h. es:

- definiert die **Operationen,** die ausgeführt werden können
- die **Messages,** die in den einzelnen Operationen als Inputs und Outputs verwendet werden.

types
Das <types> Element definiert die Datentypen, die vom Webservice verwendet werden. MapForce unterstützt XML-Schemas in WSDL-Dateien, da dies das gebräuchlichste Typsystem für WSDL-Dateien ist. In MapForce werden diese Elemente (Datentypen) als Datenelemente in einer (Message)-Komponente angezeigt, sodass Sie diese direkt auf andere Datenelemente/Konstrukte mappen können.

message
Das <message> Element definiert die **Parts** (Teile) jeder Message sowie die **Datenelemente** der Input- und Output-Parameter einer Operation. Dies sind die Messages, die zwischen Client und Server ausgetauscht werden. Es gibt drei Arten von Messages: Input, Output und Fault. In MapForce ist jede **message** eine **Komponente**, von der oder auf die Sie andere Datenelemente mappen können. **Messages** können aus einem oder mehreren Message Parts bestehen.

Bei Verwendung der document/literal-Kombination in MapForce muss sich das **message / part-Element** auf ein globales Element und nicht einen **type** beziehen. So referenziert z.B. das Attribut `element` im folgenden Code ein in einem Schema definiertes globales Element (`ns2:Vendor`):

```
<message name="processRequest">
      <part name="inputData" element="ns2:Vendor"/>
</message>
```

Der folgende Code hingegen referenziert einen "type" im Schema:

```
<message name="processRequest">
      <part name="inputData" type="ns2:VendorType"/>
</message>
```

operation
In Operationen werden Messages als Input- und Output-Parameter verwendet. Eine Operation kann die folgenden Messages enthalten:

- eine Input Message
- null oder mehr Output Messages
- null oder mehr Fault Messages

Input Messages können nur als Quellkomponenten verwendet werden. Output- und Fault Messages können nur als Zielkomponenten verwendet werden.

WSDL 2.0

WSDL 2.0 unterscheidet sich beträchtlich von WSDL 1.1. Die wichtigsten Unterschiede sind:

- PortTypes wurden in Interfaces umbenannt.
- Messages und Parts werden nun anhand des XML-Schematypsystems im Element "types" definiert
- Ports wurden in Endpoints umbenannt.
- WSDL 2.0 Operation Inputs und Outputs werden vom XML-Schema definiert.

In MapForce wird für eine WSDL-Komponente im Dialogfeld "Komponenteneinstellungen" sowohl für WSD 1.1 Ports als auch für WSDL 2.0 Endpoints "Endpoint" angezeigt.

© 2016-2022 Altova GmbH

7.2 Erstellen von SOAP-Webservice-Projekten anhand von WSDL-Dateien

In diesem Beispiel wird gezeigt, wie Sie einen SOAP-Webservice in MapForce erstellen. Sie finden das Mapping-Projekt zu diesem Beispiel unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\Query Person database.mfp**. Ziele in diesem Beispiel:

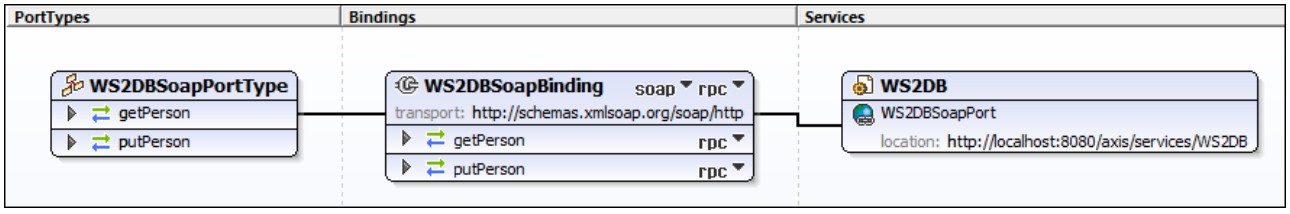
- Abrufen einer Gruppe von Personendatensätze anhand einiger Kriterien mittels SOAP-Request aus einer Microsoft Access-Datenbank auf einem Server. (Die Abfrage wird zur Laufzeit am Cient eingegeben und anschließend an den Server gesendet. Der Server sendet anschließend eine XML-Antwort mit den Suchergebnissen zurück).
- Einfügen neuer Datensätze in die Microsoft Access-Datenbank, ebenfalls mittels SOAP Requests. (Die Personendaten werden im SOAP-Request angegeben. Bei Erfolg sendet der Server eine Bestätigung zurück, dass die Personendaten erfolgreich zur Datenbank hinzugefügt wurden.)

Bitte beachten Sie die folgenden Punkte:

- Das Mapping, mit dem der Webservice in MapForce generiert wird, hängt nicht von der Ziel-Programmiersprache ab, d.h. es ist identisch, egal ob Java- oder C#-Webservices generiert wird. Der Unterschied entsteht erst beim Kompilieren und Bereitstellen des Webservice auf dem Webserver, wie später in dieser Dokumentation beschrieben.
- Um SOAP-Beispiel-Requests zu generieren, um eine Vorschau auf die Mapping-Ausgabe anzeigen zu können, wird in diesem Beispiel die XMLSpy Enterprise Edition verwendet. Die WSDL-Quelldatei in diesem MapForce-Projekt wurde ebenfalls mit XMLSpy erstellt (und kann damit auch validiert werden). Webservices können auch ohne XMLSpy in MapForce erstellt werden. In diesem Fall benötigen Sie jedoch einen SOAP-Client, der in der Lage ist, SOAP-Requests anhand von WSDL-Dateien zu generieren.

Erstellen eines Webservice-Projekts

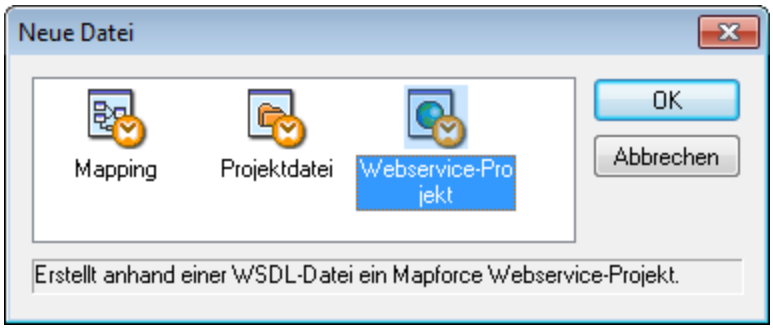
Wie bereits erwähnt, benötigen Sie zur Erstellungen eines SOAP-Webservice mit MapForce eine WSDL-Datei. Wir verwenden in diesem Beispiel einer WSDL-Datei, die mit der XMLSpy Enterprise Edition erstellt wurde. In der Abbildung unten sehen Sie die WSDL-Struktur, wie sie in der WSDL-Ansicht von XMLSpy angezeigt wird. Beachten Sie, dass darin zwei Operationen (`getPerson` und `putPerson`) für die oben angegebene Zwecke definiert sind.



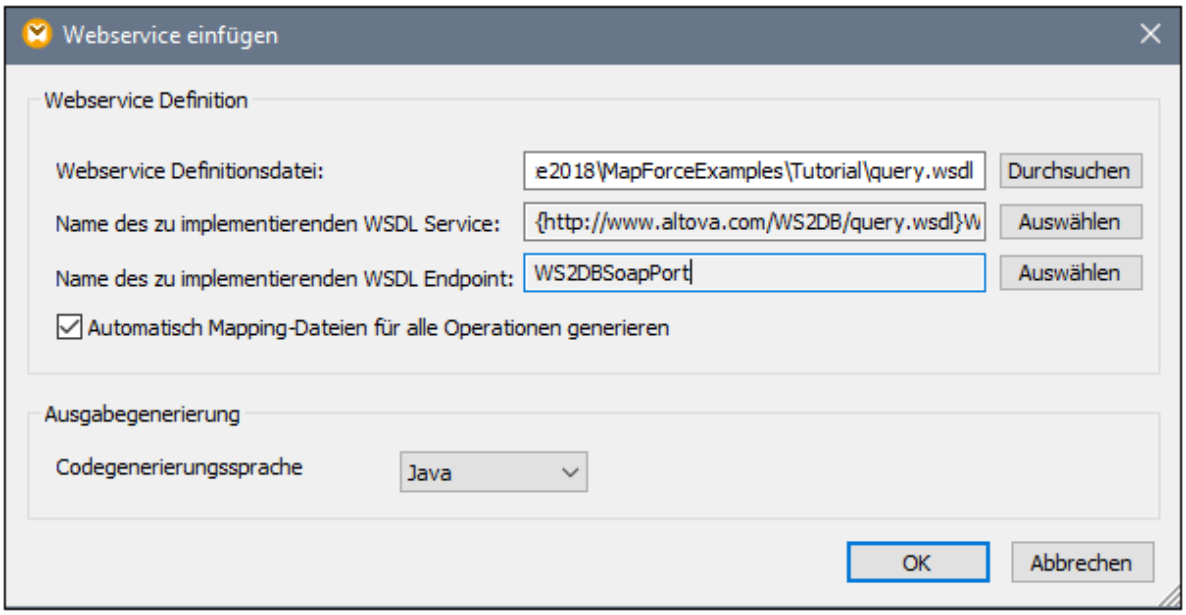
Anmerkung: Die WSDL-Datei muss gültig sein, damit Sie anhand dieser Datei ein MapForce Webservice-Projekt erstellen können.

So erstellen Sie anhand dieses Webservice ein Webservice-Projekt:

1. Wählen Sie im Menü **Datei** den Befehl **Neu** und wählen Sie "Webservice-Projekt" aus.



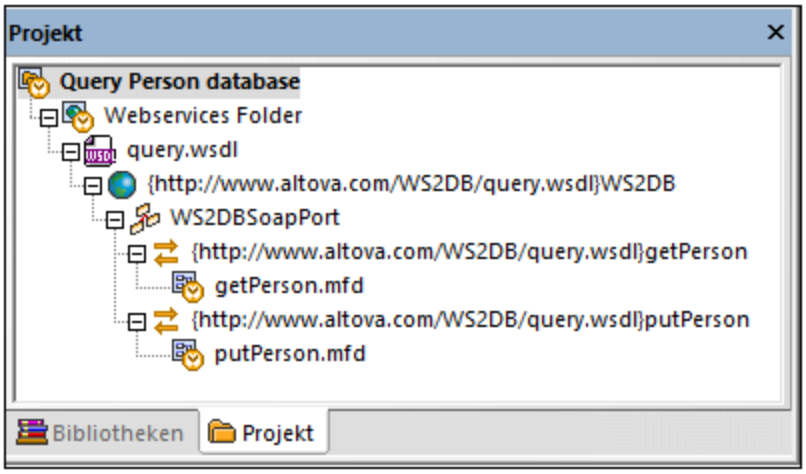
2. Navigieren Sie im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** zur Datei **query.wsdl**. Nach Auswahl der WSDL-Datei werden die restlichen Felder automatisch ausgefüllt.



3. Klicken Sie auf OK.
6. Geben Sie den Namen des WSDL-Projekts ein, wenn Sie dazu aufgefordert werden und klicken Sie auf **Speichern**.

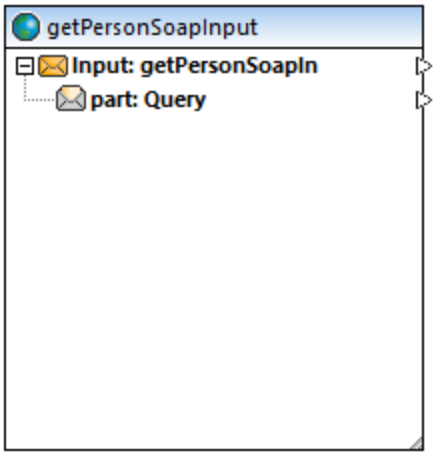
Im Projektfenster sehen Sie das Projekt und den WSDL-Namen sowie die einzelnen in der WSDL-Datei definierten Operationen. Die beiden Operationen sind **getPerson** und **putPerson**.

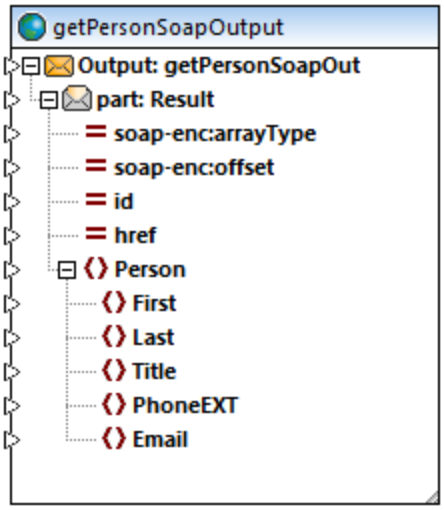
© 2016-2022 Altova GmbH



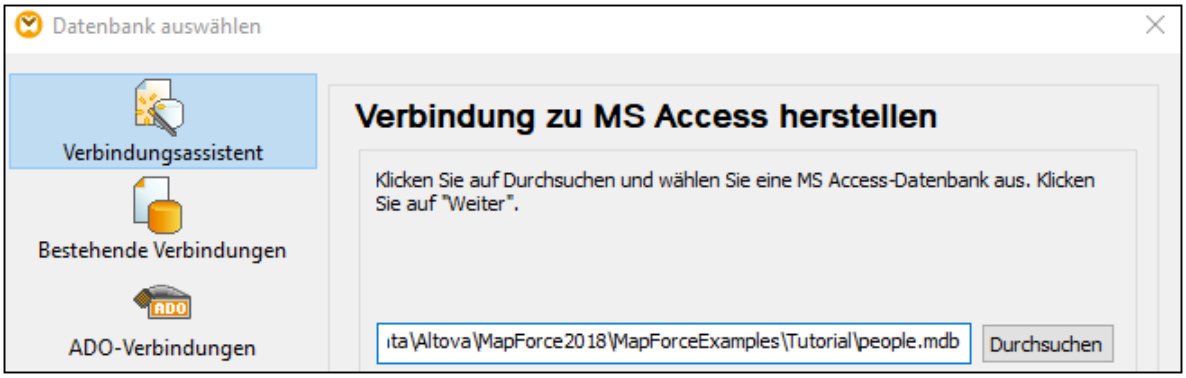
Erstellen der Operation "getPerson"

Doppelklicken Sie im Projektfenster auf die Datei **getPerson.mfd,** um deren Inhalt im Hauptbereich zu laden. Die **getPersonSoapIn**-Komponente enthält das Datenelement query, mit Hilfe dessen die Datenbank über den Webservice abgefragt werden soll. Die **getPersonSoapOut**-Komponente enthält die in der WSDL-Datei definierte `Person`-Struktur.

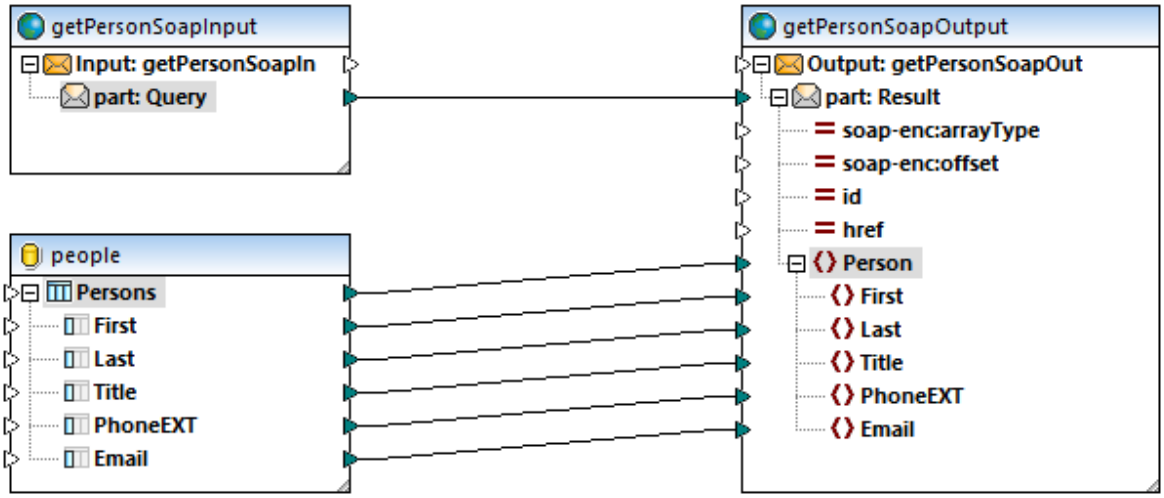




Wählen Sie **Einfügen | Datenbank** und fügen Sie die Datenbank **people.mdb** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** zum Mapping-Bereich hinzu. Die Datenbank enthält nur eine Tabelle, `Persons`, deren Felder mit denen in der Komponente `getPersonSoapOut` übereinstimmen. Nähere Informationen zum Hinzufügen von Datenbanken als Mapping-Komponenten finden Sie unter Datenbanken und MapForce 165.



Ziehen Sie Verbindungen zwischen den Komponenten, wie unten gezeigt:



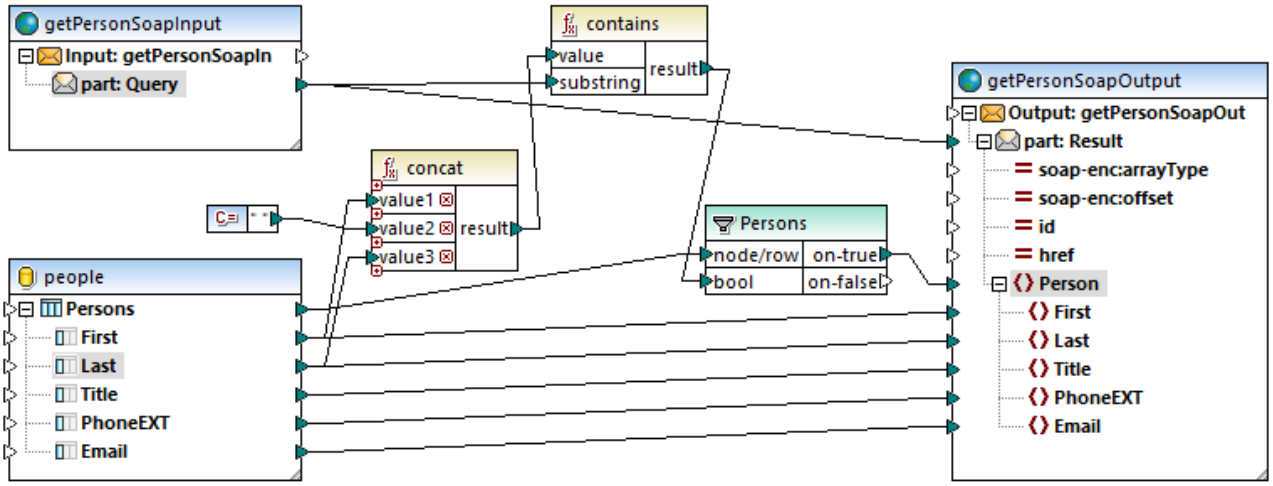
Das oben gezeigte Mapping-Design ist noch nicht fertig, Sie sehen aber schon, was zur Laufzeit geschieht. Die Verbindung zwischen **part:Query** und **part:Result** stellt den Webservice-Aufruf dar. Das Datenelement **part:Query** der Komponente `getPersonSoapIn` ist der Platzhalter für die Abfrage (hier wird der Abfragestring in den SOAP-Client eingegeben). Über die Verbindungen zwischen der Datenbanktabelle und dem Datenelement **Person** der **getPersonSoapOutput**-Komponente werden Werte aus den entsprechenden Datenbankfeldern extrahiert und an die entsprechenden Datenelemente in der Zielkomponente übergeben. Sie können bereits schnell eine Vorschau auf die Ausgabe anzeigen. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf die Komponente **getPersonSoapInput** und wählen Sie im Kontextmenü den Befehl **Eigenschaften.**
2. Klicken Sie auf **Durchsuchen** und wählen Sie den folgenden Beispiel-Request aus: **<Dokumente>\Altova\MapForce2023\MapForceExamples\getPersonRequest.xml**.
3. Klicken Sie auf **OK**.
4. Klicken Sie auf das Register **Ausgabe**.

Zu diesem Zeitpunkt werden alle Datensätze aus der Datenbanktabelle extrahiert, was nicht sehr hilfreich ist. Wir müssen daher einen Filter hinzufügen und nur die Datenelemente extrahieren, die dem Request-String entsprechen.

© 2016-2022 Altova GmbH

1. Fügen Sie die Funktion **concat** durch Ziehen aus dem Fenster "Bibliotheken" 34 zum Mapping-Bereich hinzu. Mit Hilfe dieser Funktion werden die Vor- (`First`) und Nachnamen (`Last`)der einzelnen Personen miteinander verkettet.
2. Klicken Sie im Menü **Einfügen** auf Konstante und fügen Sie eine Konstante bestehend aus einem Leerzeichen hinzu. Die Konstante liefert das Leerzeichen zwischen dem aus der Datenbank extrahierten Vor- und Nachnamen.
3. Fügen Sie die Funktion **contains** durch Ziehen aus dem Fenster "Bibliotheken" in den Mapping-Bereich hinzu.
4. Klicken Sie mit der rechten Maustaste auf die Verbindung zwischen der Quelltabelle **Persons** und dem Ziel-Node **Person** und wählen Sie im Kontextmenü den Befehl **Filter einfügen: Nodes/Zeilen**.
7. Ziehen Sie Verbindungslinien zwischen den Komponenten, wie unten gezeigt.

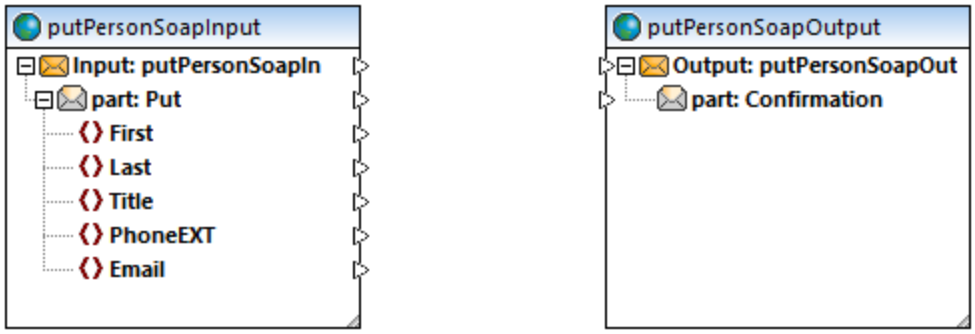


"getPerson" Mapping

Wie oben gezeigt, gibt die Funktion **contains** **true** zurück, wenn ein ganzer Name oder ein Teil eines Namens einer Person in der Datenbank mit dem Abfragestring übereinstimmt ("name" wird dabei von der **concat** Funktion bereitgestellt und besteht aus dem Vornamen (first), gefolgt von einem Leerzeichen und anschließend dem Nachnamen (last)). Wenn eine Übereinstimmung gefunden wird, werden die Details der gefundenen Person in die Response-Nachricht inkludiert. So würden etwa bei Eingabe von "Ro" in diesem Beispiel die folgenden Datensätze zurückgegeben: "Martin **Ro**pe", "**Ro**nald Superstring", "**Ro**bert Darkmatter" und "**Ro**ger Gravity".

Erstellen der Operation "putPerson"

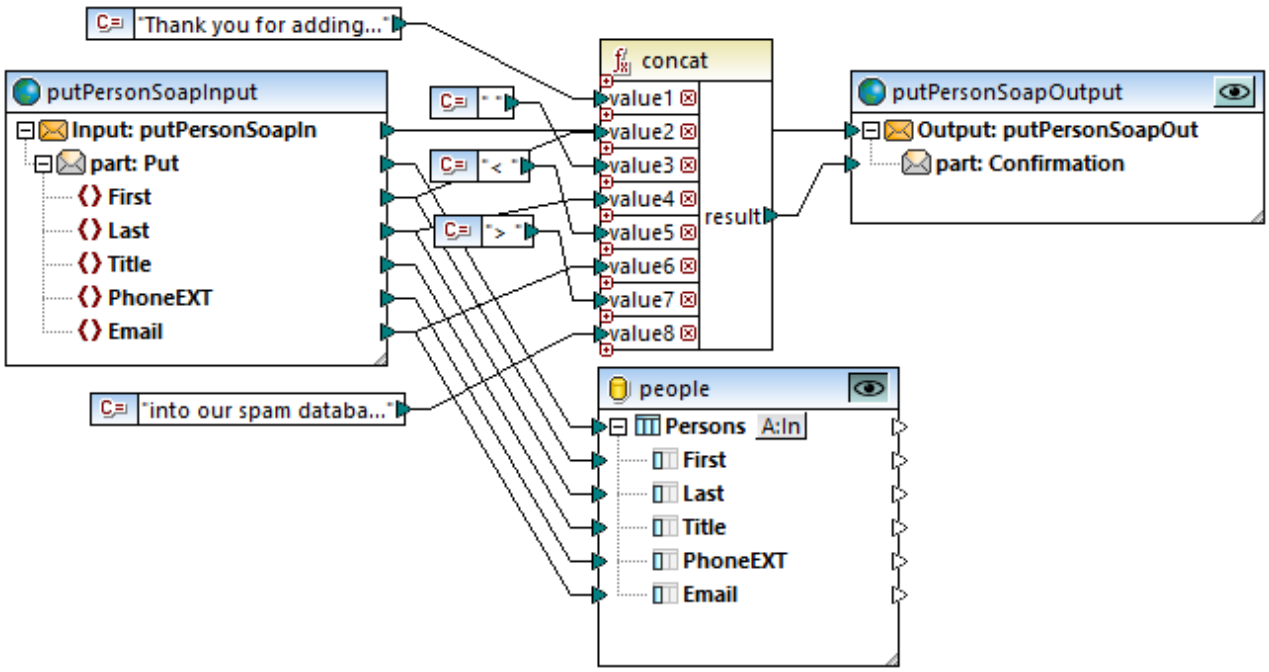
Doppelklicken Sie im Projektfenster auf die Datei **putPerson.mfd,** um ihren Inhalt in den Hauptbereich zu laden. Die Komponente **putPersonSoapInput** stellt die Request-Struktur (die Datenelemente, die über den Webservice in die Datenbank eingefügt werden) zur Verfügung. Die Komponente **putPersonSoapOutput** repräsentiert das Ergebnis des Webservice-Aufrufs. Sie enthält einen Bestätigungsteil (part: Confirmation), in dem das Ergebnis des Webservice-Aufrufs angezeigt wird.



Gehen Sie vor, wie oben beschrieben, um die folgenden Komponenten zum Mapping hinzuzufügen:

- die Access-Datenbank **people.mdb** (Hier werden die durch den Webservice-Aufruf abgerufenen Daten eingefügt)
- die Funktion **concat** und fünf Konstanten (Damit wird der Bestätigungsstring erstellt, der dem Aufrufer des Webservice angezeigt wird)

Ziehen Sie die Verbindungen wie unten gezeigt:



"putPerson"-Mapping

Im oben gezeigten Mapping wird die Datenbank aufgrund der Verbindung zwischen **part: Put** und der Datenbank zur Laufzeit aktualisiert. Auf Seite der Zielkomponente wird die **part: Confirmation** Response durch Verkettung verschiedener Felder aus dem Request sowie von durch die Konstanten bereitgestelltem Text erzeugt. Nähere Informationen zu Mappings, mit denen Datenbanken aktualisiert werden, finden Sie unter Mappen von Daten auf Datenbanken 290.

So zeigen Sie eine Vorschau auf das Mapping an:

 © 2016-2022 Altova GmbH

1. Klicken Sie mit der rechten Maustaste auf die Komponente **putPersonSoapInput**.
2. Klicken Sie auf **Durchsuchen** und wählen Sie den folgenden Beispiel-Request aus: **<Dokumente>\Altova\MapForce2023\MapForceExamples\putPersonRequest.xml**.
3. Klicken Sie auf das Register **Ausgabe**.

Wenn Sie eine Vorschau auf das Mapping anzeigen, generiert MapForce die INSERT-Abfrage, mit der die Datenbank aktualisiert wird und zeigt sie im Fenster **Ausgabe** an, führt aber die tatsächliche Aktualisierung nicht durch. Um die Abfrage an der Datenbank auszuführen, wählen Sie den Menübefehl **Ausgabe | SQL-Script ausführen.**

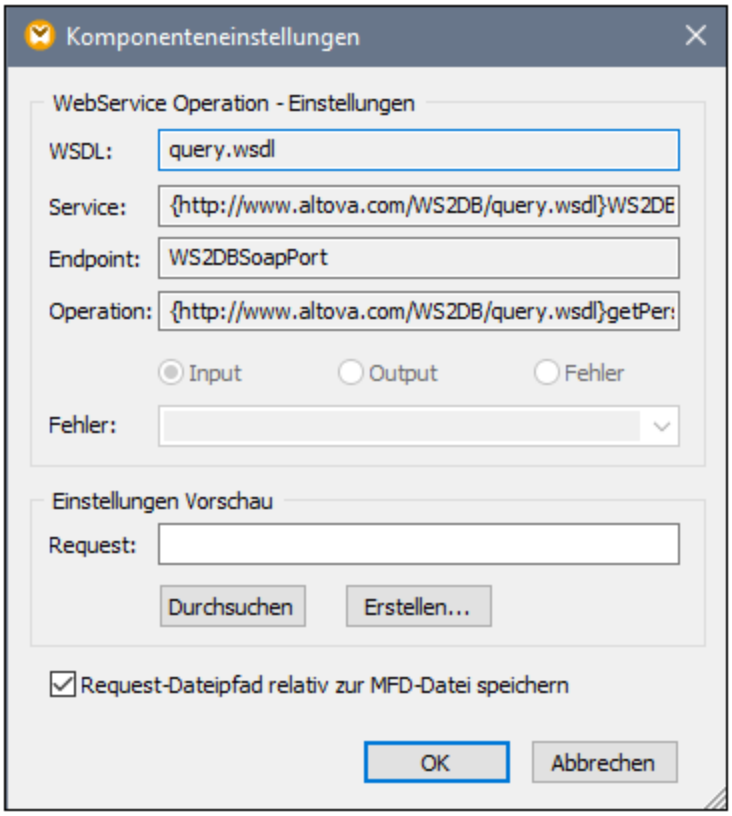
```
1  /*
2  The following SQL statements are only for preview and may not be executed in another SQL query tool!
3  To execute these statements use function "Run SQL-script" from menu "Output".
4  Connect to database using the following connection-string:
5  Data Source=C:\Users\altova\Documents\Altova\MapForce2018\MapForceExamples\Tutorial\people.mdb;Provider=Microsoft.Jet.OLEDB.4.0
6  */
7
8  INSERT INTO [Persons] ([First], [Last], [Title], [PhoneEXT], [Email]) VALUES ('John', 'Doe', 'Mr', '123', 'john.doe@example.org')
9
```

Wenn der kompilierte Webservice jedoch auf einem Server bereitgestellt wird und von einem Client aufgerufen wird, wird die Datenbank sofort aktualisiert. Siehe auch Ausführen von Mappings, die Datenbanken modifizieren 282.

Anzeige einer Vorschau auf die Mapping-Ausgabe

Sie können in MapForce eine Vorschau von Mappings, die anhand von WSDL-Dateien erstellt wurden, anzeigen, bevor Sie diese auf einem Webserver bereitstellen. Zur Anzeige einer Vorschau auf ein solches Mapping benötigen Sie einen SOAP-Beispiel-Request (XML-Datei), der der WSDL-Datei entspricht. Sie können entweder mit XMLSpy oder einem anderen SOAP-Client Ihrer Wahl einen SOAP-Beispiel-Request für jede einzelne Operation (Mapping) in Ihrem Webservice-Projekt erstellen. In XMLSpy können Sie den Request mit dem Menübefehl **SOAP | Neuen SOAP erzeugen** generieren. Wenn neben MapForce auch XMLSpy installiert ist, können Sie direkt von MapForce aus einen Beispiel-Request generieren, z.B.:

1. Klicken Sie mit der rechten Maustaste auf die **getPersonSoapInput**-Komponente des Mappings und wählen Sie im Kontextmenü den Befehl **Eigenschaften**.



2. Klicken Sie auf **Erstellen**.
3. Speichern Sie die XML-Datei in einem Ordner auf Ihrem Rechner, wenn Sie dazu aufgefordert werden.

Die Request-Datei wird mit Dummy-Input-Werten (z.B. "String") anstelle der Input-Parameter erstellt. Bevor Sie die Request-Datei verwenden, müssen Sie die Dummy-Werte durch die gewünschten Request-Werte ersetzen. Für dieses Projekt stehen zwei Beispiel-Requests zur Verfügung:

1. **<Dokumente>\Altova\MapForce2023\MapForceExamples\getPersonRequest.xml.** Verwenden Sie diese Datei, um eine Vorschau auf die Ausgabe des Mappings **getPerson** zu sehen. Standardmäßig stellt die Datei den Wert "Ro" als Input für den Webservice bereit. Damit werden aus der Datenbank alle Personendatensätze, deren Vor- oder Nachname den Text "Ro" enthält, abgerufen.
2. **<Dokumente>\Altova\MapForce2023\MapForceExamples\putPersonRequest.xml.** Verwenden Sie diese Datei, um eine Vorschau auf die Ausgabe des Mappings **putPerson** zu sehen. Standardmäßig stellt diese Request-Datei die Personendaten einer fiktiven Person als Input für den Webservice bereit.

Um bei der Mapping-Vorschau andere Ausgaberesultate zu erhalten, bearbeiten Sie die beiden obigen Dateien, sodass sie die gewünschten Input-Werte enthalten.

Nächste Schritte

Sie sind nun bereit, um Code zu generieren, der den Webservice implementiert (entweder Java oder C#), ihn kompiliert und auf einem Server bereitstellt, wie in den folgenden Beispielen beschrieben:

- Beispiel: Generieren von SOAP-Webservices (C#) 1275

© 2016-2022 Altova GmbH

- Beispiel: Generieren von SOAP-Webservices (Java)[1290]

7.3 Definieren von SOAP-Webservice-Fehlern

Eine-WSDL-Datei kann ein `fault`-Element für eine Operation und ein Message-Attribut, das die Fehlermeldung enthält, enthalten. Sie können daher, wenn Sie einen SOAP-Webservice in MapForce erstellen, eine "Fault"-Komponente zu Ihrem Mapping hinzufügen. Dadurch löst der Webservice bei Erfüllung einer bestimmten Bedingung eine Ausnahme aus.

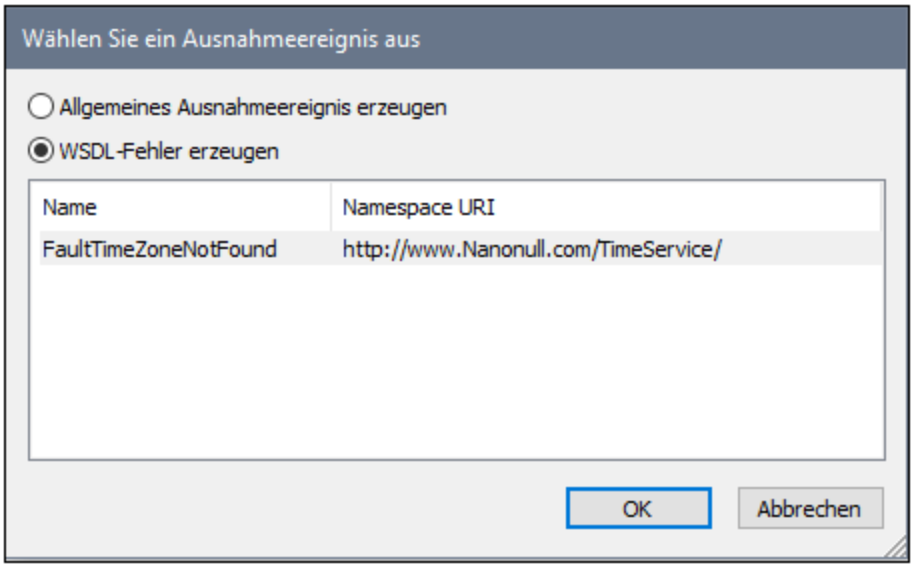
> Um ein Fault-Element in ein Mapping einzufügen, muss in der WSDL-Datei ein `fault`-Element vorhanden sein.

Sie können die Fehlerbedingung sowie den Text der Fehlermeldung im Mapping selbst definieren. Wenn die Bedingung zutrifft, kommt es zum folgenden Webservice-Fehler:

- In einer Live-Umgebung (wenn der Webservice auf einem Server bereitgestellt wurde und ausgeführt wurde) kommt es zu einem Webservice-Fehler und der Caller erhält die benutzerdefinierte Meldung.
- In MapForce (bei der Erstellung des Designs) wird die Mapping-Ausführung beendet und Sie sehen die Meldung im Fenster **Meldungen**.

So fügen Sie eine Fehlerkomponente zu einem MapForce Mapping hinzu:

1. Klicken Sie im Menü **Einfügen** auf **Ausnahme**. (Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche **Ausnahme** .)
2. Wählen Sie **WSDL-Fehler erzeugen** aus.

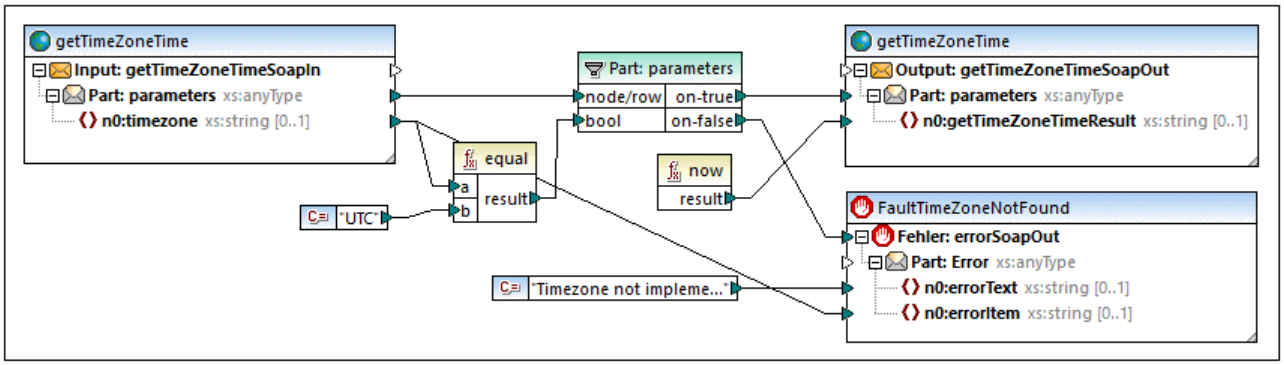


Anmerkung: Wenn die Option **WSDL-Fehler erzeugen** deaktiviert ist und Sie die Meldung "Keine Fehler für diese Operation" sehen, bedeutet dies, dass in der WSDL-Datei keine Fehler definiert sind und daher auch keine in MapForce hinzugefügt werden können.

Ein Beispiel für ein Mapping, das einen Webservice-Fehler enthält, finden Sie im folgenden MapForce-Beispielprojekt:

© 2016-2022 Altova GmbH

<Dokumente>\Altova\MapForce2023\MapForceExamples\TimeService\TimeService.mfp. Doppelklicken Sie auf die Mapping-Operation **getTimeZoneTime.mfd**.



Im obigen Beispiel sollte ein Ausnahmeereignis ausgelöst werden, wenn **n0:timezone** nicht gleich "UTC" ist.

- Die **equal**-Komponente überprüft, ob timezone gleich UTC ist, das bool-Ergebnis wird an die filter-Komponente weitergegeben.
- Wenn die Bedingung false ist, d.h. ein anderer Wert als UTC ist, aktiviert der **on-false**-Parameter der filter-Komponente das Ausnahmeereignis **Fehler:errorSoapOut** und der Mapping-Vorgang wird gestoppt. (Beachten Sie, dass Sie das Ausnahmeereignis bei Bedarf auch mit dem on-true-Parameter verbinden können.)
- Die SoapFault-Message liefert zwei Fehlertext-Gruppen.

Es ist äußerst wichtig, wo Sie in diesem Beispiel die filter-Komponente platzieren:

- Es müssen **beide Parameter** der filter-Komponente, sowohl on-true als auch on-false, gemappt werden! Einer muss auf die fault-Komponente gemappt werden und der andere auf die Zielkomponente, an die die gefilterten Daten der Quellkomponente übergeben werden. Andernfalls wird die fault-Komponente nie ausgelöst.
- Die **Ausnahmeereigniskomponente** und die **Zielkomponente** müssen direkt mit der **filter**-Komponente verbunden werden. Zwischen die filter-Komponente und die Ausnahmeereignis- bzw. die Zielkomponente dürfen keine Funktionen oder andere Komponenten platziert werden.

7.4 Java SOAP-Webservice-Spezifisches

MapForce generiert den gesamten Code und alle Scripts, die zur Erstellung eines SOAP-Webservice benötigt werden. Damit der Webservice für den Konsumenten jedoch zur Verfügung steht, muss der generierte Code erzeugt und auf dem Axis2 (Tomcat)-Server bereitgestellt werden. Zusätzlich dazu wird zum Erzeugen des Java-Codes Apache Ant benötigt.

Generieren und Erzeugen von Java-Code

1. Öffnen Sie das Webservice-Projek, anhand dessen Sie Code generieren möchten. Ein Beispiel, wie Sie ein solches Projekt erstellen, finden Sie im vorherigen Abschnitt unter Erstellen von SOAP-Webservice-Projekten anhand von WSDL-Dateien 1261.
2. Klicken Sie im Menü **Projekt** auf **Code generieren in | Java** und wählen Sie das Zielverzeichnis aus. Nach Abschluss der Codegenerierung werden im Zielverzeichnis mehrere Ordner und Dateien, darunter ein **com**-Verzeichnis erstellt, das allgemeine Altova-Klassen sowie die eigentliche Klasse des Webservice-Projekts enthält.
3. Erzeugen Sie den generierten Java-Code (indem Sie die von MapForce generierte **build.xml** für Apache Ant bereitstellen). Dadurch wird eine Axis-Archivdatei (*.aar) erstellt, die Sie auf Axis2 bereitstellen können.

Wenn Sie das generierte Projekt mit Eclipse öffnen, sehen Sie eventuell einen Fehler wie "The import org.apache cannot be resolved". Vergewissern Sie sich in diesem Fall, dass die Axis2-Bibliotheken zum Java Build-Pfad hinzugefügt wurden. Um die Axis2-Bibliotheken in Eclipse 4.4.2 zum Java Build-Pfad hinzuzufügen, gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste im Paket-Explorer auf das Projekt und wählen Sie **Eigenschaften**.
2. Klicken Sie auf **Java Build Path**.
3. Klicken Sie auf dem Register **Bibliotheken** auf **Externe JARs hinzufügen** und fügen Sie die Axis2-Bibliotheken aus dem Ordner `<AXIS2_HOME>\lib` hinzu.

Bereitstellen des Webservice

Um den Webservice bereitzustellen, wählen Sie eine der folgenden Methoden:

- Öffnen Sie die Web-Verwaltungsseite "Upload Services" von Axis2 und laden Sie die im vorherigen Schritt erstellte .aar-Datei hoch
- Laden Sie die Datei manuell hoch. Wenn Ihr Tomcat-Server z.B. im Ordner `<TOMCAT_HOME>`, installiert wurden, können Sie die .aar-Datei manuell in den Ordner `<TOMCAT_HOME>\webapps\axis2\WEB-INF\services` kopieren.

Ein Schritt-für-Schritt-Beispiel finden Sie unter Beispiel: Generieren von SOAP-Webservices (Java) 1290.

Webservice online entfernen

Löschen Sie die Datei *.aar aus dem Ordner `<TOMCAT_HOME>\webapps\axis2\WEB-INF\services`.

© 2016-2022 Altova GmbH

Axis2-Einschränkungen

Die Axis2-Unterstützung für **RPC/encoded** ist eingeschränkt. MapForce kann jedoch RPC/encoded WebServices (sowohl SOAP 1.1 als auch SOAP 1.2) generieren, mit der Einschränkung, dass die Original-WSDL-Datei nicht vom Webserver abgerufen wird.

D.h., dass z.B. http://127.0.0.1/axis2/services/WS2DB?wsdl keine verwendbare WSDL-Datei zurückgeben würde.

Bei **document/literal** Webservices stellt die obige URL eine brauchbare, korrekte WSDL-Datei bereit. Sie unterscheidet sich jedoch vom Original: Kommentare werden entfernt , Namespaces werden geändert. Der Webservice hat jedoch immer noch dieselbe Semantik wie die Original-WSDL-Datei, mit der der Service erstellt wurde.

Zwar unterstützt Axis2 **RPC/encoded** nicht, doch kann es WSDL von online gestelltem Java-Code (kompiliertem Code) generieren. Daher kann MapForce-generierter Code **RPC/encoded** Messages verarbeiten; Axis2 wird nur für die Übertragung verwendet.

Bekanntes Problem: Namespaces in der SOAP Response-Nachricht

Der von MapForce generierte Code instantiiert die **`javax.xml.transform.TransformerFactory`**-Klasse. Diese Klasse kann aus der Systemeigenschaft **`javax.xml.transform.TransformerFactory`** gelesen werden, wenn die Klassenimplementierung geladen ist (Nähere Informationen dazu finden Sie in der Java-Dokumentation zu dieser Klasse).

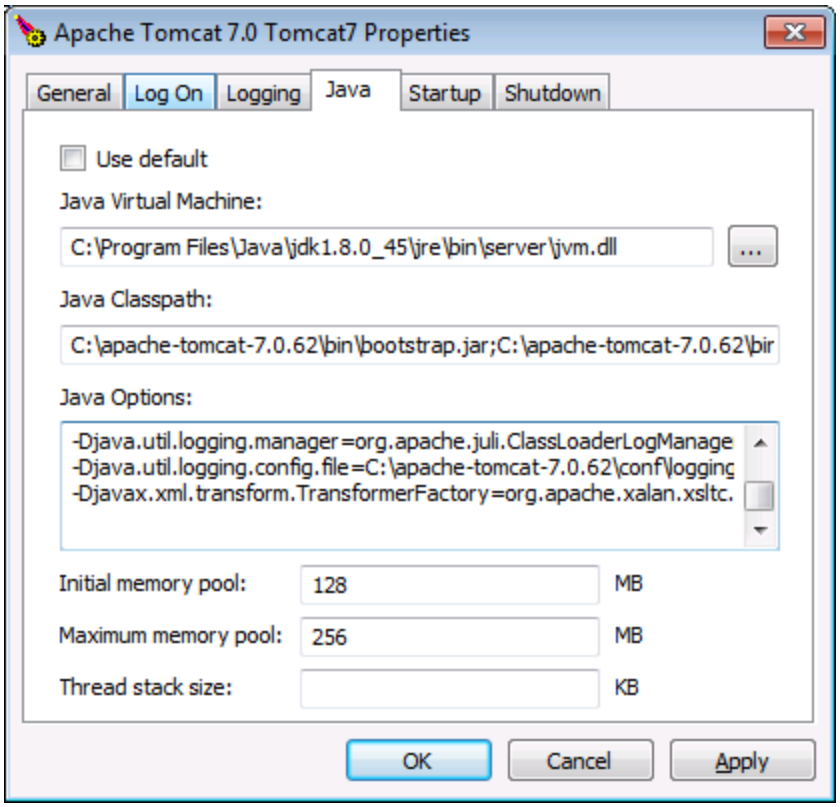
Wenn in Ihrer Java-Umgebung in der Systemeigenschaft **`javax.xml.transform.TransformerFactory`** die Implementierung **`org.apache.xalan.transformer.TransformerIdentityImpl`** definiert ist, wird die SOAP Response-Nachricht eventuell mit falschen Namespaces generiert.

Um dies zu verhindern, wird empfohlen, die Implementierung **`org.apache.xalan.xsltc.trax.TransformerFactoryImpl`** zu verwenden. Fügen Sie dazu die folgenden Parameter zur Java Virtual Machine (JVM) hinzu:

```
-
Djavax.xml.transform.TransformerFactory=org.apache.xalan.xsltc.trax.TransformerFactoryImp
l
```

Die Anleitung zum Hinzufügen der Parameter zur JVM ist von dem von Ihnen verwendeten Betriebssystem und Server abhängig. Die folgende Anleitung gilt für Apache Tomcat 7.0, das als Service-Applikation auf Windows 7 ausgeführt wird:

1. Führen Sie **Tomcat7w.exe** aus (Diese Datei befindet sich im Unterordner **\bin** des Tomcat Stammverzeichnisses `CATALINA_HOME`).
2. Klicken Sie auf das Register **Java**.
3. Geben Sie am Ende des Felds "Java Options" folgenden Text ein: `-Djavax.xml.transform.TransformerFactory=org.apache.xalan.xsltc.trax.TransformerFactoryImpl`



 © 2016-2022 Altova GmbH

7.5 Beispiel: Generieren von SOAP-Webservices (C#)

In diesem Beispiel wird gezeigt, wie Sie mit MapForce einen SOAP-Webservice generieren. Sie werden anhand des MapForce-Beispielprojekts unter **<Dokumente>\Altova\MapForce2023\MapForceExamples\Query Person database.mfp** C#-Programmcode generieren. In diesem Projekt ist ein SOAP-Webservice implementiert, der zwei Operationen bereitstellt: `getPerson` und `putPerson`. Dieser Webservice kommuniziert im Hintergrund mit einer Microsoft Access-Datenbank, um über die obigen Operationen Personendaten abzurufen bzw. hinzuzufügen. Ein Beispiel zum Erstellen solcher Projekte in MapForce finden Sie unter Erstellen von SOAP-Webservice-Projekten anhand von WSDL-Dateien 1261.

In diesem Beispiel wird C#-Code mit Hilfe von Visual Studio 2015 kompiliert. Die Lösung wird auf Internet Information Services (IIS) 10 bereitgestellt, wobei ASP.NET aktiviert ist.

In diesem Beispiel wird aus Gründen der Einfachheit eine einfache Webserver-Konfiguration verwendet. In einer Produktionsumgebung müssen Sie unter Umständen abhängig von Webserver, Sicherheitsvorgaben und anderen Faktoren eine andere Konfiguration verwenden.

In diesem Beispiel wird zum Aufrufen des SOAP-Webservice die XMLSpy Enterprise Edition verwendet. Wenn Sie XMLSpy nicht installiert haben, benötigen Sie einen SOAP-Client, um den Webservice zu testen oder müssen Programmcode schreiben, der den Webservice aufrufen kann.

Anmerkung: Wenn Sie IIS 7.x verwenden, müssen Sie eventuell zuerst die Windows-Funktionalität "Kompatibilität mit IIS-Metabasis und IIS 6-Konfiguration" installieren.

Schritt 1: Vorbereiten der query.wsdl-Datei

Dieser Beispiel-Webservice stammt aus der folgenden WSDL-Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\query.wsdl**. Standardmäßig sind die Service-Details in der Datei **query.wsdl** für C# und Java in zwei separaten Abschnitten definiert, von denen einer auskommentiert ist. Bevor Sie daher C#-Code generieren, gehen Sie zum folgenden Abschnitt in der Datei **query.wsdl** und stellen Sie sicher, dass dieser nicht auskommentiert ist:

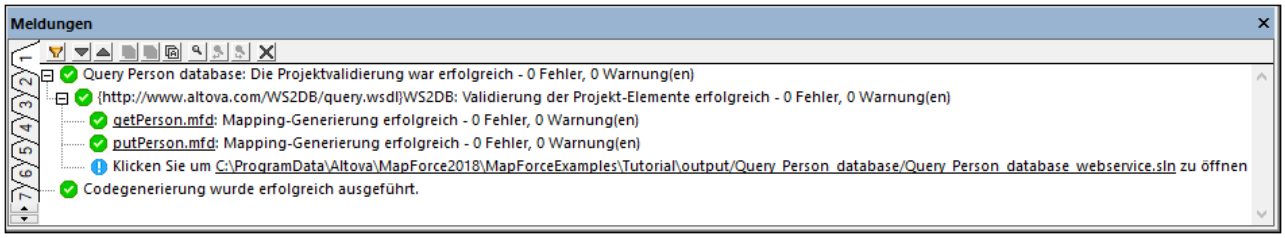
```
<service name="WS2DB">
   <port name="WS2DBSoapPort" binding="tns:WS2DBSoapBinding">
      <soap:address location="http://localhost/WS2DB/WS2DB.asmx"/>
   </port>
</service>
```

Stellen Sie außerdem sicher, dass der folgende Abschnitt (der nur für Java gilt) auskommentiert ist:

```
<!--<service name="WS2DB">
   <port name="WS2DBSoapPort" binding="tns:WS2DBSoapBinding">
      <soap:address location="http://localhost:8080/axis/services/WS2DB"/>
   </port>
</service>-->
```

Schritt 2: Generieren und Erzeugen von C#-Code

Starten Sie MapForce und öffnen Sie das Projekt "Query Person database.mfp" aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Klicken Sie mit der rechten Maustaste im Projektfenster auf das Projekt und wählen Sie den Befehl **Code generieren in | C#**. MapForce generiert daraufhin Programmcode und zeigt das Ergebnis im Fenster "Meldungen" an:



Code wird standardmäßig relativ zum Projektverzeichnis in einem Unterverzeichnis mit dem Namen "output" generiert. In diesem Tutorial behalten wir die Standardeinstellung unverändert bei. Wie Sie das Ausgabeverzeichnis ändern können, ist unter Definieren der Codegenerierungseinstellungen 122 erläutert.

Starten Sie Visual Studio und öffnen Sie die generierte Lösung.

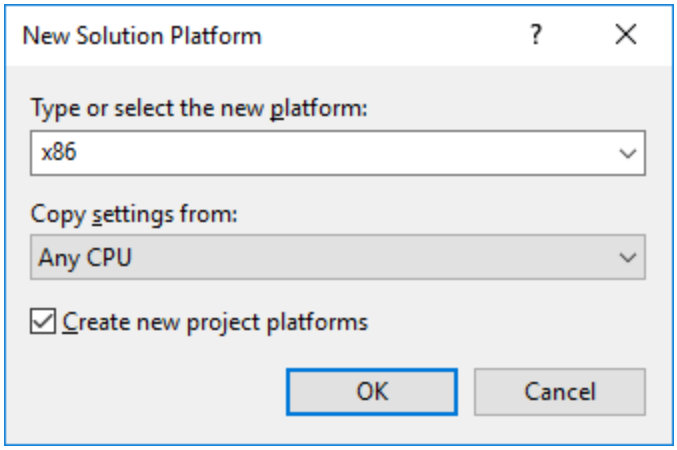
> Eventuell müssen Sie Visual Studio als Administrator ausführen. Diese Beispiellösung muss außerdem, wie unten gezeigt, als 32-Bit-Applikation kompiliert werden.

Sie können die Projektmappe folgendermaßen erstellen:

- Klicken Sie im Menü **Build** auf **Projektmappe erstellen (Strg+Umschalt+B)**.

Beachten Sie, dass in diesem Beispiel eine Microsoft Access-Datenbank mit Hilfe des ADO-Providers für Access abgefragt wird. Dieser Provider funktioniert nur mit 32-Bit-Applikationen. Die Projektmappe muss daher folgendermaßen als 32-Bit-Applikation kompiliert werden:

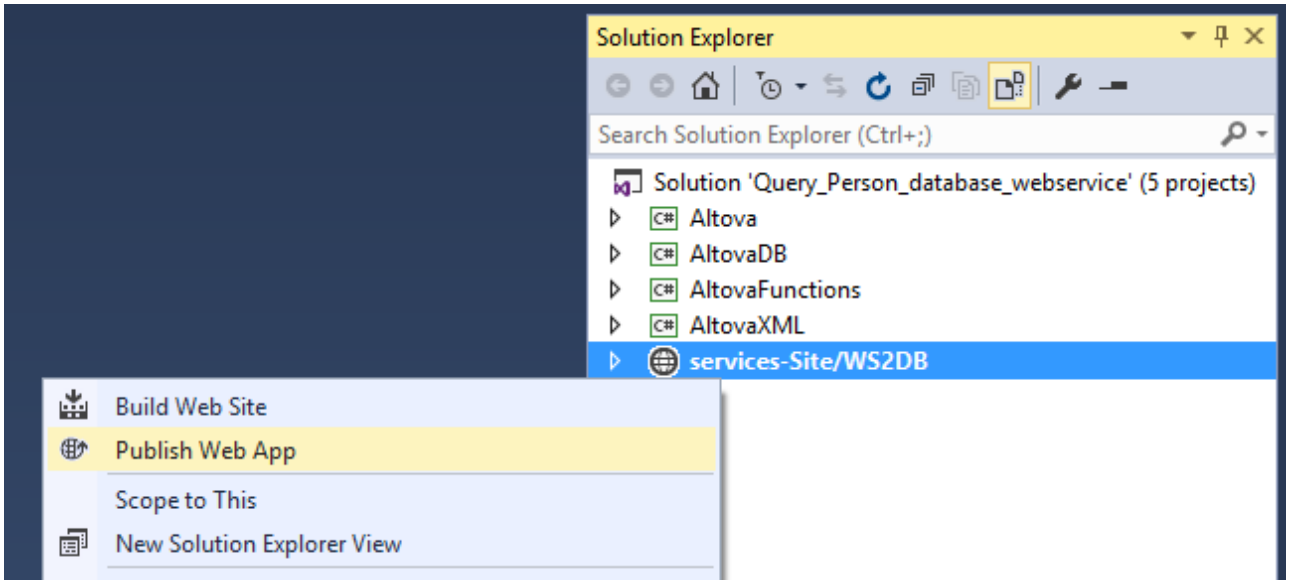
- Klicken Sie im Menü **Build** auf **Konfigurations-Manager**, erstellen Sie eine neue Projektmappenplattform für x86 und erstellen Sie den Build erneut.



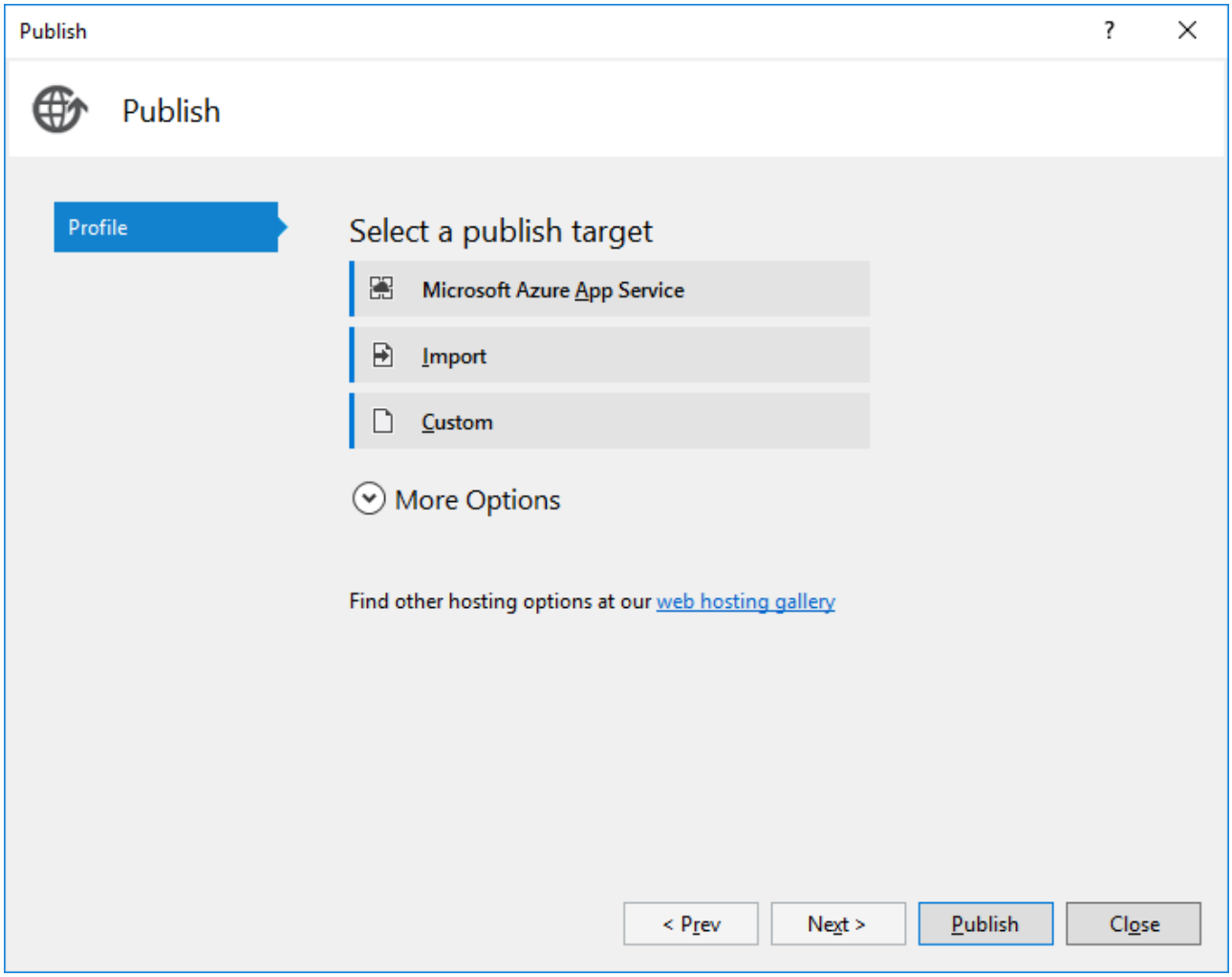
 © 2016-2022 Altova GmbH

Schritt 3: Bereitstellen des Webservice auf IIS

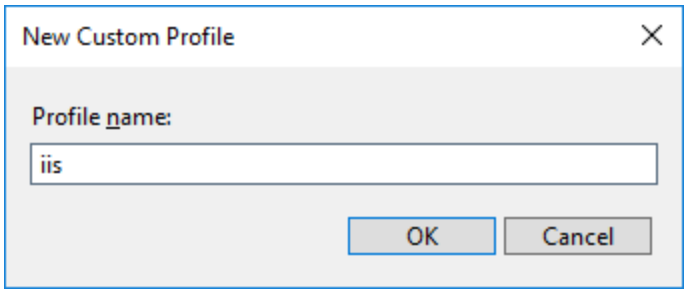
Klicken Sie mit der rechten Maustaste im Projektmappenfenster von Visual Studio auf das Projekt **services-Site/WS2DB** und wählen Sie im Kontextmenü den Befehl "Web-App veröffentlichen".



Wenn Sie aufgefordert werden, ein Ziel auszuwählen, klicken Sie auf "Benutzerdefiniert".

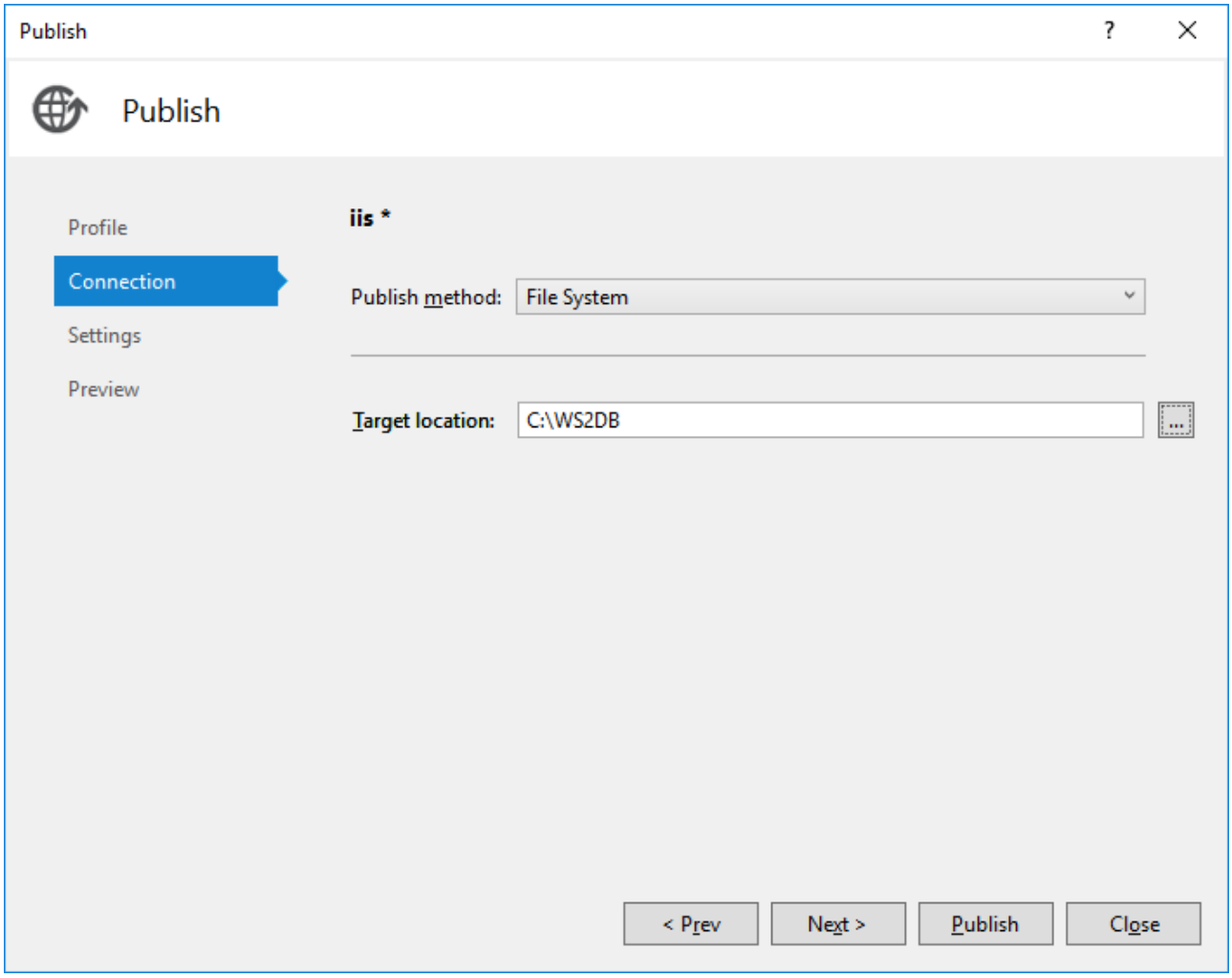


Wenn Sie aufgefordert werden, einen Profilnamen einzugeben, geben Sie einen Wert ein, den Sie später leicht erkennen, z.B. "iis".

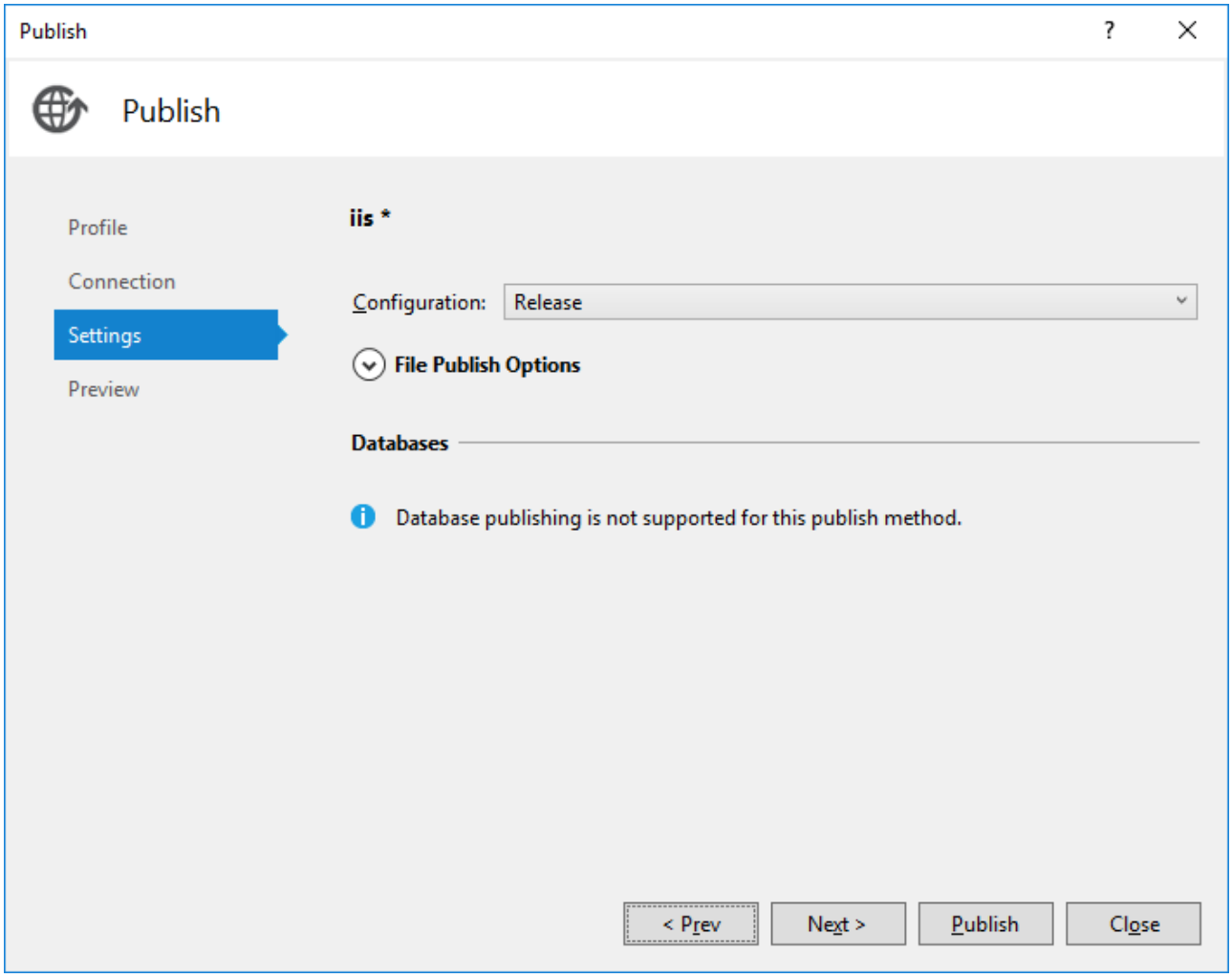


Wählen Sie als Veröffentlichungsmethode "Dateisystem" aus und wählen Sie einen Zielordner (in diesem Beispiel "C:\WS2DB") aus.

© 2016-2022 Altova GmbH

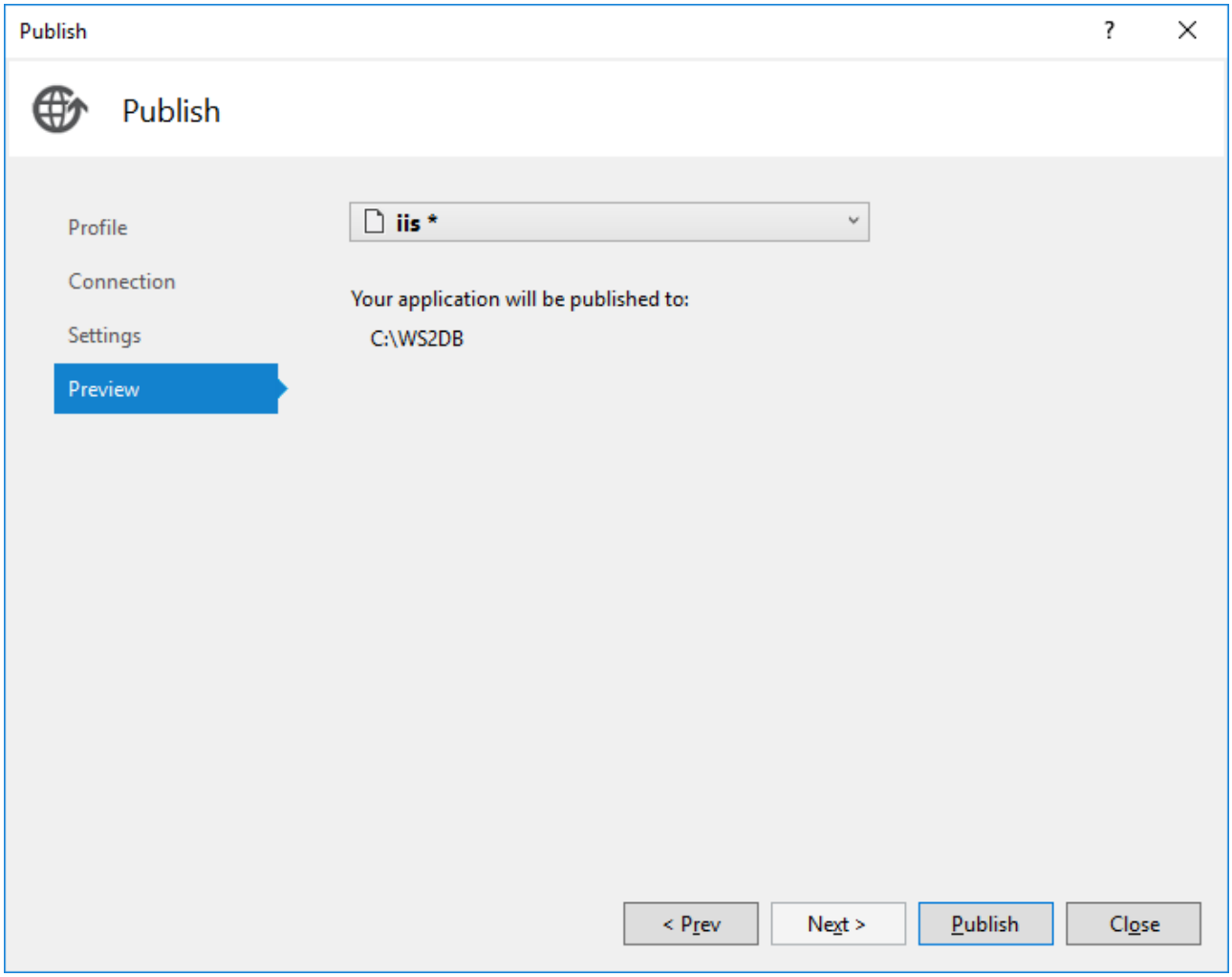


Klicken Sie auf **Weiter**. Behalten Sie die Standardeinstellungen unverändert bei.

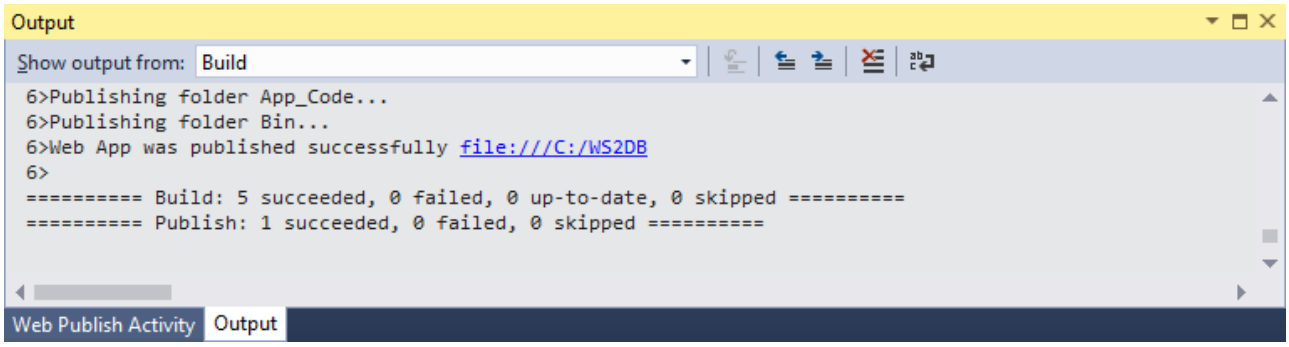


Klicken Sie auf **Weiter** und anschließend auf **Veröffentlichen**.

© 2016-2022 Altova GmbH



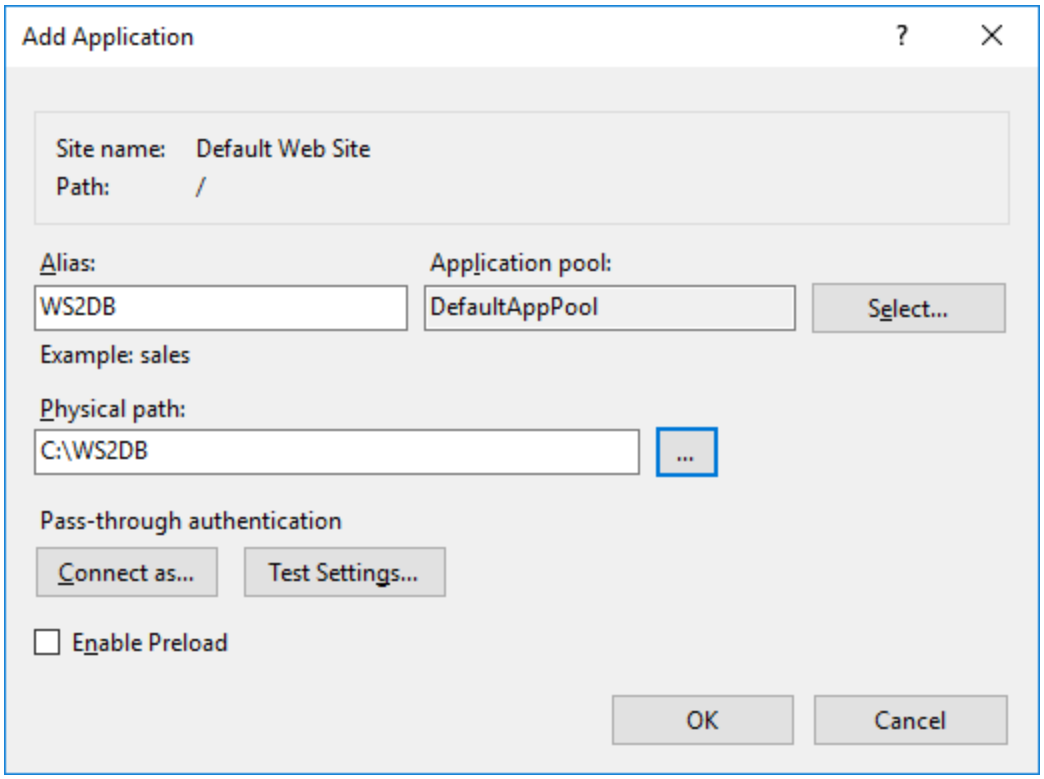
Im Ausgabefenster wird das Ergebnis angezeigt.



Schritt 4: Konfigurieren der Applikation im IIS Manager

Sie können die neue Applikation folgendermaßen zu IIS hinzufügen:

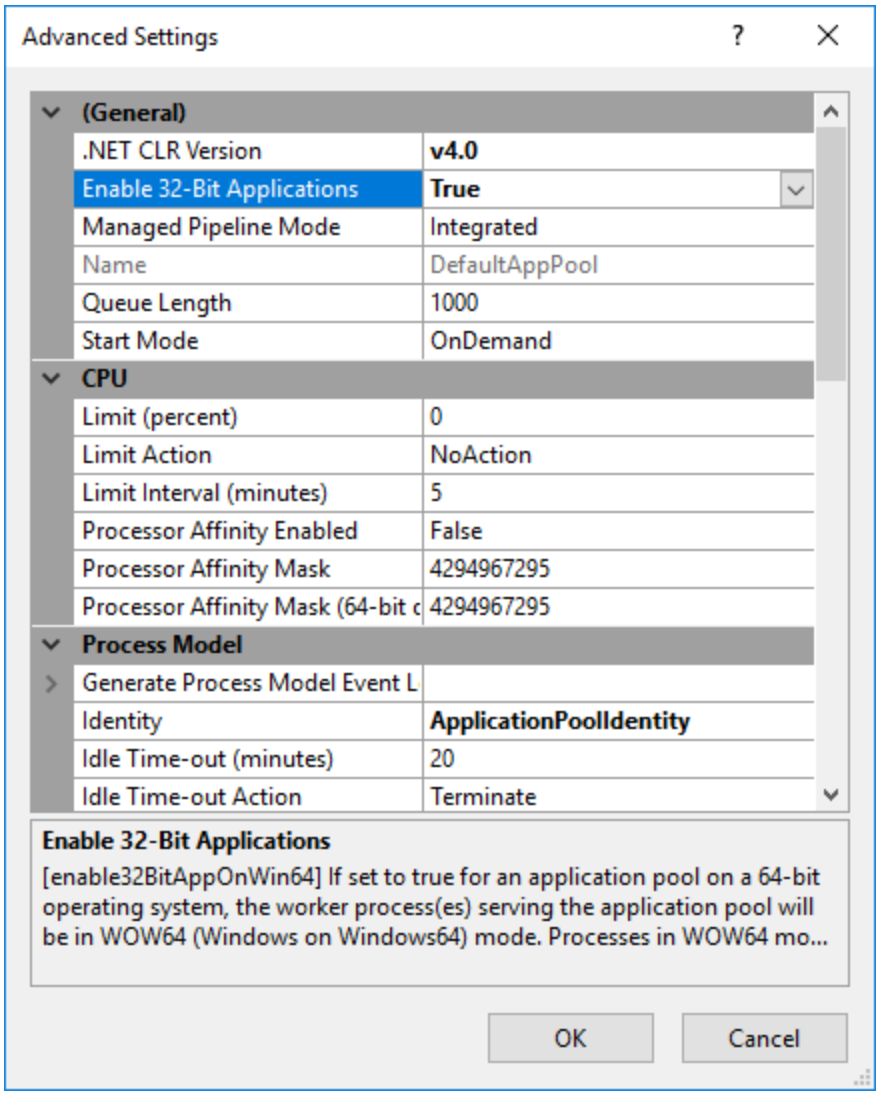
1. Klicken Sie mit der rechten Maustaste in IIS auf **Sites | Standardwebsite** und wählen Sie im Kontextmenü **Anwendung hinzufügen**.
2. Geben Sie unter "Alias-Name" einen Namen zum Identifizieren Ihrer Web-Applikation ein (in diesem Beispiel "services").
3. Geben Sie unter "Physischer Pfad" den Pfad, unter dem die Applikation zuvor veröffentlicht wurde, ein (in diesem Beispiel "C:\WS2DB").



Zusätzlich dazu müssen für dieses Beispiel im Pool, in dem die App ausgeführt wird, 32-Bit-Applikationen aktiviert werden.

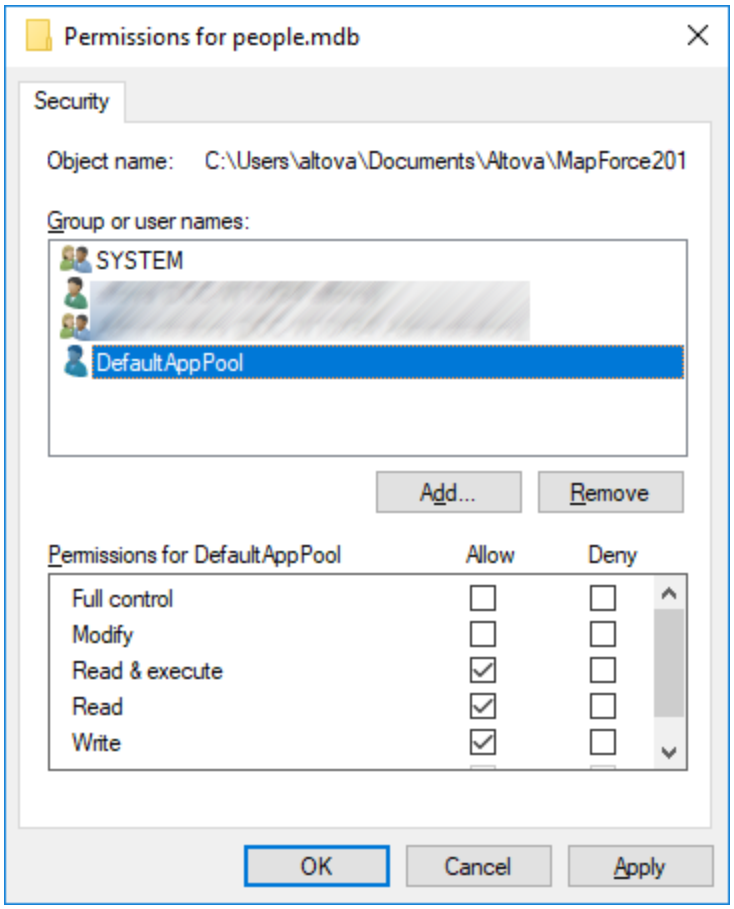
1. Klicken Sie in IIS mit der rechten Maustaste auf den Applikationspool, in dem die App bereitgestellt wurde, und wählen Sie im Kontextmenü **Erweiterte Einstellungen**.
2. Aktivieren Sie unter **32-Bit-Anwendungen aktivieren** die Option **Wahr**.

© 2016-2022 Altova GmbH

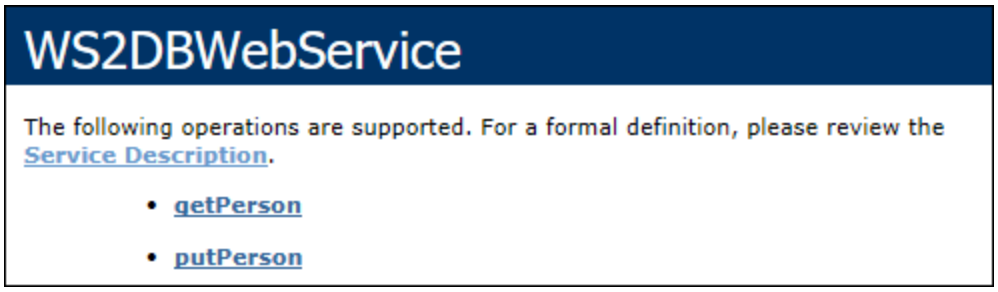


Da in diesem Beispiel Daten aus einer Microsoft Access-Datenbank gelesen werden, muss der IIS-Applikationspool außerdem Zugriffsrechte auf die Datenbankdatei haben. Wenn Sie das MapForce-Projekt nicht geändert haben, befindet sich die Datenbank unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\people.mdb**.

1. Klicken Sie in Windows Explorer mit der rechten Maustaste auf die Datei **people.mdb** und wählen Sie **Eigenschaften**.
2. Klicken Sie auf dem Register **Sicherheit** auf **Bearbeiten**.
3. Klicken Sie auf **Hinzufügen** und geben Sie den folgenden Objektnamen ein: **IIS AppPool\DefaultAppPool**. Ändern Sie "DefaultAppPool" in einen anderen Wert, wenn Sie die App in einem anderen Pool als dem Standardpool bereitgestellt haben.
4. Räumen Sie die benötigten Rechte ein und klicken Sie auf **Anwenden**. (In diesem Beispiel werden für die Datenbankdatei Lese- und Schreibrechte benötigt.)



Um zu testen, ob der Webservice hochgeladen wurde und läuft, rufen Sie die folgende URL auf: `http://localhost/WS2DB`.



Wenn die Webservice-Informationen im Browser geladen werden, können Sie fortfahren und den Webservice über eine Client-Applikation, wie unten gezeigt, aufrufen.

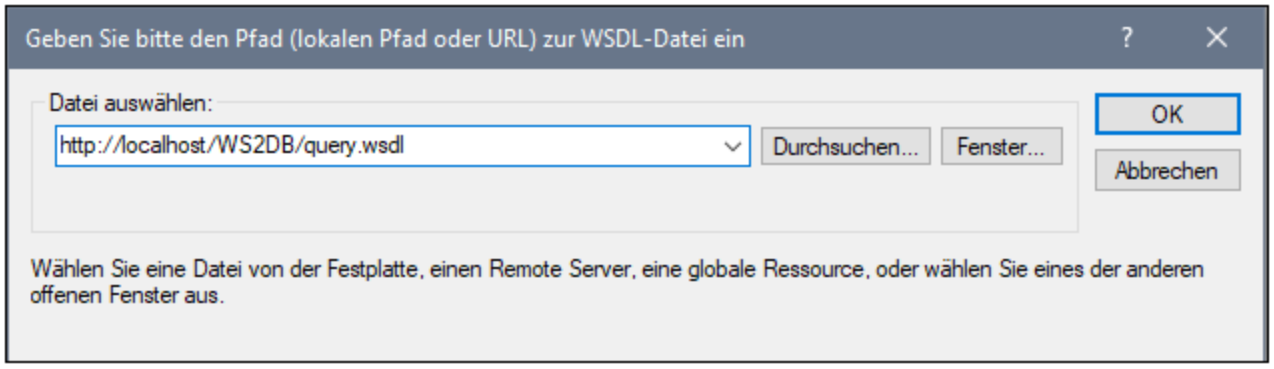
Aufrufen des Webservice

Wenn Sie die obige Anleitung genau befolgt haben, steht der Webservice unter `http://localhost/WS2DB/query.wsdl` zur Verfügung. Wir verwenden zum Aufrufen des soeben bereitgestellten Webservice XMLSpy. Sie können natürlich auch andere SOAP-Clients dafür verwenden.

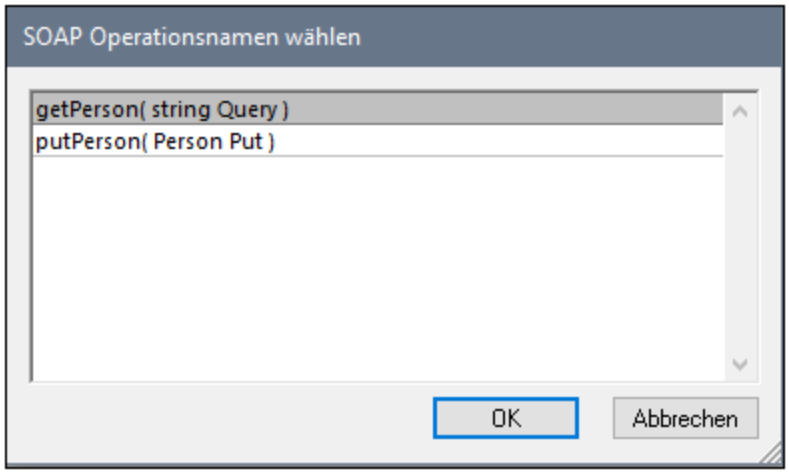
© 2016-2022 Altova GmbH

Anmerkung: Es wird empfohlen, die Original-WSDL-Datei unter `http://localhost/WS2DB/query.wsdl` aufzurufen, anstatt die "?wsdl"-Abfragemethode zu verwenden. Bei zweiterer Methode erhalten Sie unter Umständen eine WSDL-Datei, der einige Features aus der Originaldatei fehlen oder eine nicht funktionsfähige WSDL-Datei.

Starten Sie XMLSpy (Enterprise Edition). Klicken Sie im Menü **SOAP** auf **Neuen SOAP-Request erzeugen**. Geben Sie die WSDL-Datei des Webservice ein, wenn Sie dazu aufgefordert werden (in diesem Beispiel `http://localhost/WS2DB/query.wsdl`).



Klicken Sie auf OK. Wenn Sie aufgefordert werden, eine SOAP-Operation auszuwählen, wählen Sie die Operation **getPerson** aus.



XMLSpy generiert daraufhin einen Request wie den unten gezeigten.

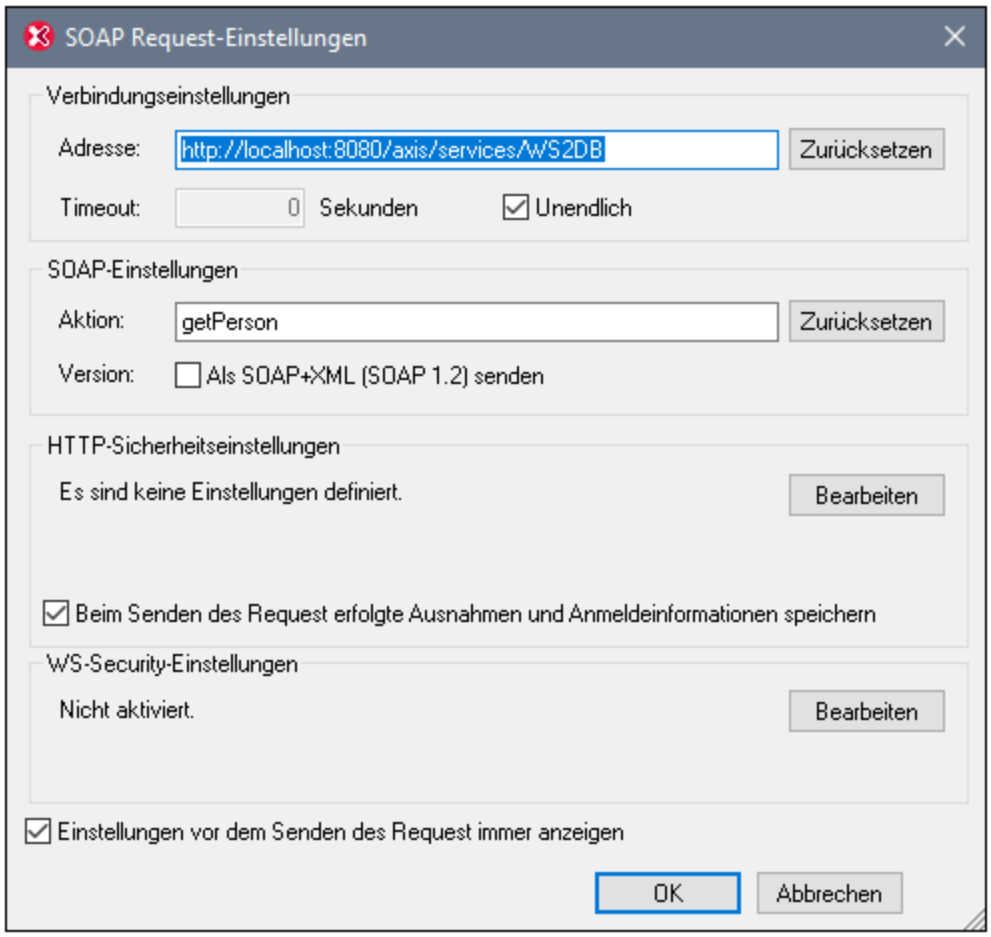
```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <SOAP-ENV:Body>
      <getPerson SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <Query xsi:type="xsd:string">String</Query>
      </getPerson>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Suchen Sie das Element <Query> des Request und ersetzen Sie "String" durch den Wert, der als Input für den Webservice fungieren soll. Wie bereits erwähnt ruft dieser Webservice Personendaten aus einer Access-Datenbank ab. Damit durch den Aufruf in diesem Beispiel Daten abgerufen werden, wird "String" durch "Ro" ersetzt (d.h. nur Personen, deren Vor- oder Nachname "Ro" enthält, würden abgerufen). Der neue Request sieht folgendermaßen aus:

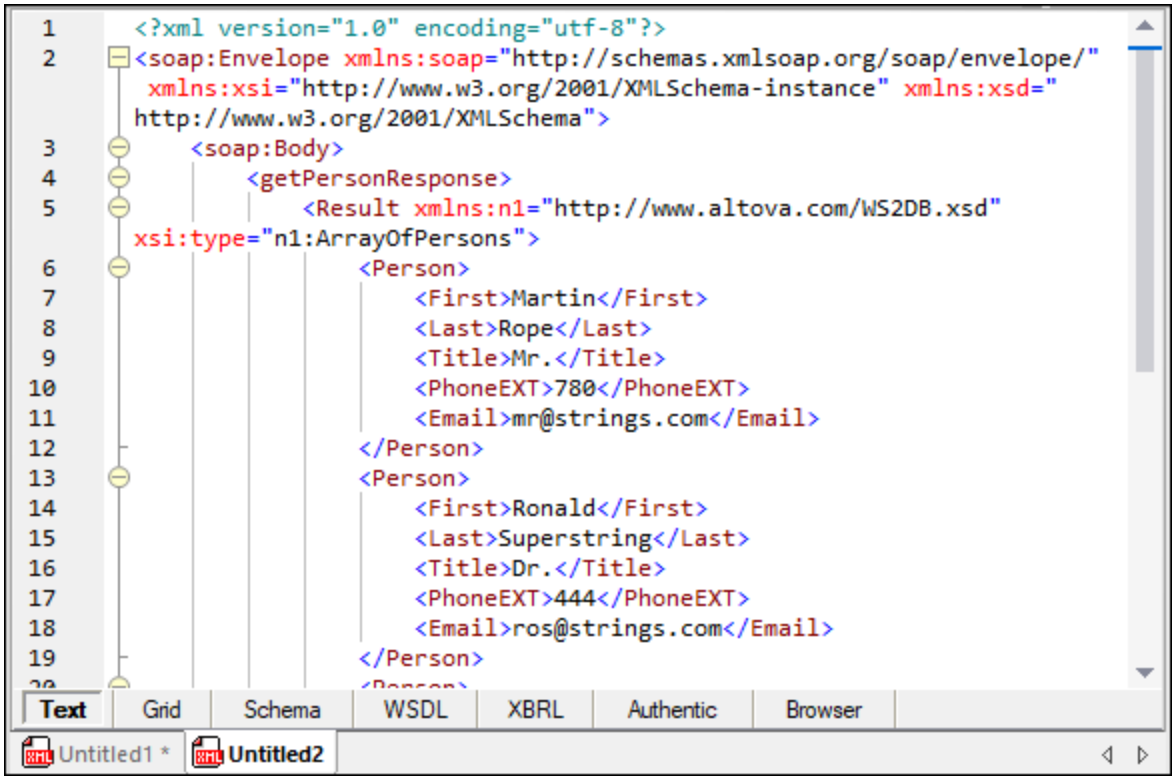
```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <SOAP-ENV:Body>
      <getPerson SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <Query xsi:type="xsd:string">Ro</Query>
      </getPerson>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sie können den obigen Request nun an den Server senden. Klicken Sie im Menü **SOAP** auf **Request an Server senden**. Daraufhin wird das Dialogfeld "SOAP Request-Einstellungen" angezeigt.

© 2016-2022 Altova GmbH



Klicken Sie auf **OK**. XMLSpy startet den Aufruf und zeigt im Editor die Antwort an. In der Abbildung unten sehen Sie eine erfolgreiche Antwort.



Auf dieselbe oben beschriebene Art können Sie auch die von diesem Webservice bereitgestellte **putPerson** aufrufen. Eine schrittweise Anleitung dazu finden Sie im Java-Tutorial im Abschnitt "Aufrufen des Webservice[1294]". Im Java-Tutorial ist nur die WSDL URL eine andere; alle anderen Anleitungen zum Aufrufen des Webservice sind gleich.

Fehlerbehebung

In der folgenden Tabelle sind häufige Probleme beschrieben, die beim Erzeugen oder Aufrufen von Webservices auftreten können. Daneben finden Sie eine Lösung dafür.

Problem	Lösung
Beim Aufruf des Webservice wird der folgende Fehler zurückgegeben: `Der 'Microsoft.Jet.OLEDB.4.0' Provider ist auf dem lokalen Rechner nicht registriert."`	1. Erstellen Sie die C#-Projektmappe in Visual Studio als 32-Bit-Applikation. 2. Stellen Sie sie auf IIS bereit. 3. Klicken Sie mit der rechten Maustaste auf den IIS-Applikationspool, in dem Sie die Applikation bereitgestellt haben und wählen Sie **Erweiterte Einstellungen**. 4. Aktivieren Sie unter **32-Bit-Anwendung aktivieren** die Einstellung **Wahr**.
Beim Aufruf des Webservice wird der folgende Fehler zurückgegeben:	• Räumen Sie dem IIS-Applikationspool Leserechte für die Access-Datenbankdatei

© 2016-2022 Altova GmbH

Problem	Lösung
`Der Microsoft Jet Datenbankprozessor kann die Datei '<path>\people.mdb' nicht öffnen. Sie wurde bereits von einem anderen Benutzer exklusiv geöffnet oder Sie benötigen eine Berechtigung, um ihre Daten anzuzeigen.`	ein, siehe "Schritt 4: Konfigurieren der Applikation im IIS Manager" oben.
Beim Aufruf des Webservice wird der folgende Fehler zurückgegeben: `System.Data.OleDb.OleDbException: Operation muss eine aktualisierbare Abfrage verwenden.`	• Räumen Sie dem IIS-Applikationspool Schreibrechte für die Access-Datenbankdatei ein, siehe "Schritt 4: Konfigurieren der Applikation im IIS Manager" oben.

7.6 Beispiel: Generieren von SOAP-Webservices (Java)

In diesem Beispiel wird gezeigt, wie Sie mit MapForce einen SOAP-Webservice generieren. Sie werden anhand des MapForce-Beispielprojekts unter **<Dokumente>\Altova\MapForce2023\MapForceExamples\\Query Person database.mfp** Java-Programmcode generieren. In diesem Projekt ist ein SOAP-Webservice implementiert, der zwei Operationen bereitstellt: `getPerson` und `putPerson`. Dieser Webservice kommuniziert im Hintergrund mit einer Microsoft Access-Datenbank, um über die obigen Operationen Personendaten abzurufen bzw. hinzuzufügen. Ein Beispiel zum Erstellen solcher Projekte in MapForce finden Sie unter Erstellen von SOAP-Webservice-Projekten anhand von WSDL-Dateien 1261.

In diesem Beispiel wird Java-Code mit Hilfe von Java 7 und Apache Ant kompiliert. Die .aar-Datei wird auf eine Apache Tomcat-Server bereitgestellt, auf dem Axis2 installiert ist.

In diesem Beispiel wird aus Gründen der Einfachheit eine einfache Webserver-Konfiguration verwendet. In einer Produktionsumgebung müssen Sie unter Umständen abhängig von Webserver, Sicherheitsvorgaben und anderen Faktoren eine andere Konfiguration verwenden.

In diesem Beispiel wird zum Aufrufen des SOAP-Webservice die XMLSpy Enterprise Edition verwendet. Wenn Sie XMLSpy nicht installiert haben, benötigen Sie einen SOAP-Client, um den Webservice zu testen oder müssen Programmcode schreiben, der den Webservice aufrufen kann.

Anmerkung: Wie bereits erwähnt, wird in diesem Webservice-Beispiel eine Microsoft Access-Datenbank aufgerufen. Bei Java 7 kann die Verbindung zur Access-Datenbank über die in Java inkludierte ODBC-JDBC Bridge hergestellt werden. Die ODBC-JDBC Bridge wurde jedoch in Java 8 entfernt und Oracle empfiehlt die Verwendung der vom Anbieter bereitgestellten JDBC-Treiber, siehe http://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/bridge.html. Um dieses Beispiel daher in einer Umgebung ab Java 8 ausführen zu können, müssen Sie JDBC-Treiber von Drittanbietern, die eine Verbindung zu einer Microsoft Access-Datenbank ermöglichen, finden und installieren.

Schritt 1: Vorbereiten der query.wsdl-Datei

Dieser Beispiel-Webservice stammt aus der folgenden WSDL-Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\query.wsdl**. Standardmäßig sind die Service-Details in der Datei **query.wsdl** für C# und Java in zwei separaten Abschnitten definiert, von denen einer auskommentiert ist. Bevor Sie daher Java-Code generieren, gehen Sie zum folgenden Abschnitt in der Datei **query.wsdl** und stellen Sie sicher, dass dieser auskommentiert ist:

```
<!--service name="WS2DB">
   <port name="WS2DBSoapPort" binding="tns:WS2DBSoapBinding">
      <soap:address location="http://localhost/WS2DB/WS2DB.asmx"/>
   </port>
</service-->
```

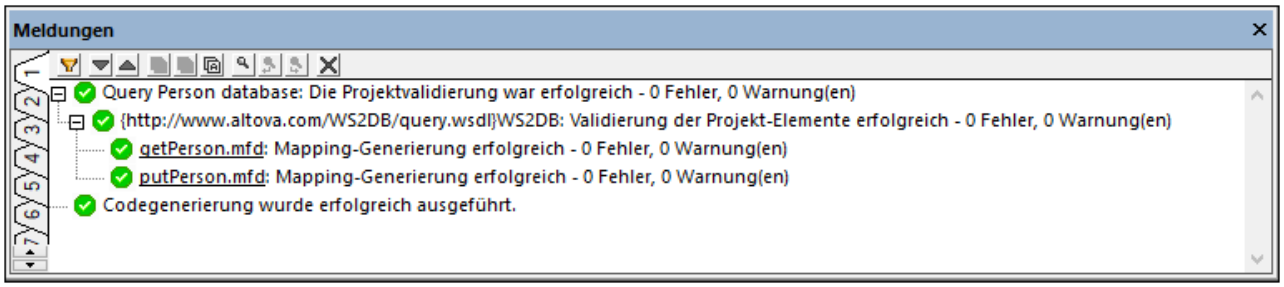
Stellen Sie außerdem sicher, dass der folgende Abschnitt (der nur für Java gilt) NICHT auskommentiert ist:

 © 2016-2022 Altova GmbH

```
<service name="WS2DB">
   <port name="WS2DBSoapPort" binding="tns:WS2DBSoapBinding">
      <soap:address location="http://localhost:8080/axis/services/WS2DB"/>
   </port>
</service>
```

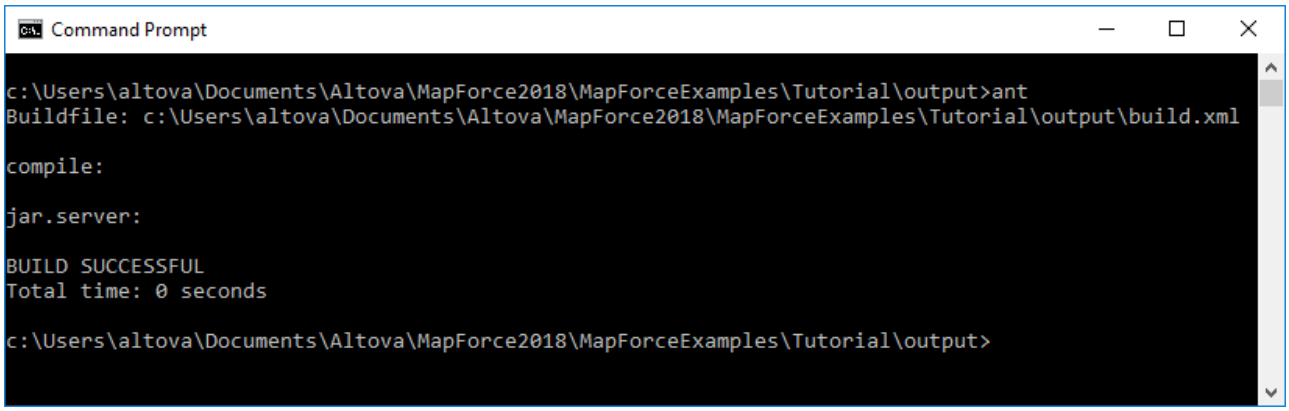
Schritt 2: Generieren und Erzeugen von Java-Code

Starten Sie MapForce und öffnen Sie das Projekt "Query Person database.mfp" aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Klicken Sie mit der rechten Maustaste im Projektfenster auf das Projekt und wählen Sie den Befehl **Code generieren in | Java**. MapForce generiert daraufhin Programmcode und zeigt das Ergebnis im Fenster "Meldungen" an:



Code wird standardmäßig relativ zum Projektverzeichnis in einem Unterverzeichnis mit dem Namen "output" generiert. In diesem Tutorial behalten wir die Standardeinstellung unverändert bei. Wie Sie das Ausgabeverzeichnis ändern können, ist unter Definieren der Codegenerierungseinstellungen 122 erläutert.

Um den Java-Code zu erstellen, führen Sie Ant im selben Verzeichnis wie die Datei **build.xml** über die Befehlszeile aus.



Als Ergebnis wird im Ausgabeverzeichnis eine Axis Archivdatei (.aar) namens **Query_Person_database.aar** generiert.

Bevor Sie Ant von irgendeinem Verzeichnis aus, wie oben gezeigt aufrufen, muss es korrekt installiert und konfiguriert werden. Hier eine kurze Zusammenfassung (Nähere Informationen dazu finden Sie im Ant-Handbuch unter http://ant.apache.org):

- Die `PATH`-Umgebungsvariable muss den Pfad zum Unterverzeichnis "bin" des Ant-Root-Verzeichnisses enthalten.
- Die `ANT_HOME`-Umgebungsvariable muss auf das Verzeichnis, in dem Ant installiert ist, gesetzt sein.
- Die `AXIS2_HOME`-Umgebungsvariable muss auf das Verzeichnis, in dem Axis2 installiert ist, gesetzt sein.
- Die `JAVA_HOME`-Umgebungsvariable muss auf das Verzeichnis, in dem Java JDK installiert ist, gesetzt sein.
- Überprüfen Sie, ob Ant ausgeführt wird, indem Sie in die Befehlszeile "ant" eingeben. Die Meldung "Buildfile: build.xml ist nicht vorhanden!" bedeutet, dass Ant konfiguriert ist und nach einer Build-Datei sucht.

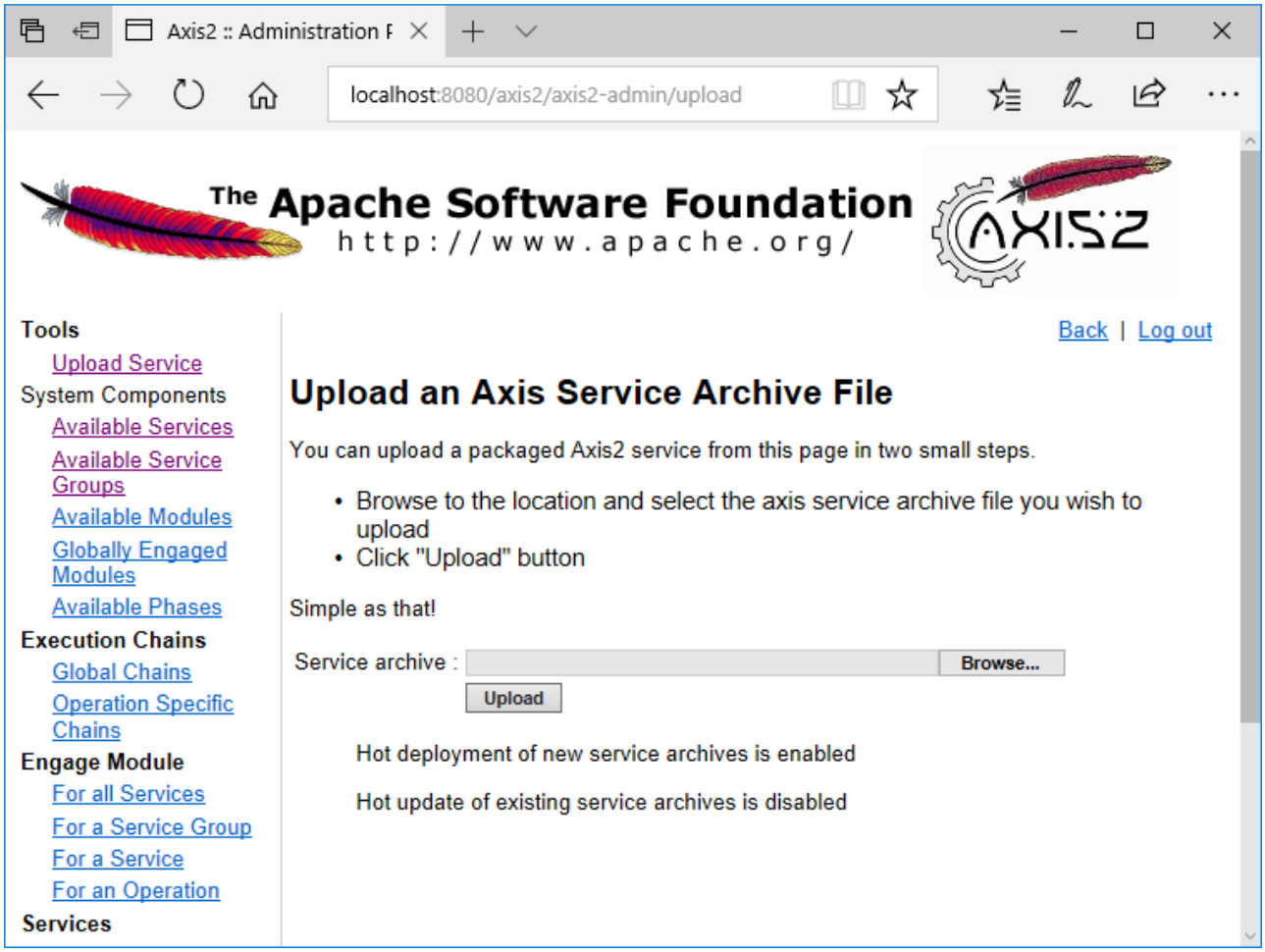
Schritt 3: Bereitstellen des Webservice auf Axis 2

Der Webservice wird in diesem Beispiel von der Axis2-Verwaltungsseite auf Tomcat hochgeladen. Die Seite kann folgendermaßen installiert und auf einem laufenden Tomcat-Server bereitgestellt werden (Nähere Informationen dazu finden Sie in der Dokumentation zu Axis2):

1. Laden Sie die Axis2 WAR (Web Archive) Distribution von http://axis.apache.org/axis2/java/core/download.html herunter.
2. Entpacken Sie das heruntergeladene Paket in ein lokales Verzeichnis.
3. Kopieren Sie die Datei **axis2.war** in das Tomcat-Verzeichnis **%CATALINA_HOME%\webapps**.
4. Stellen Sie sicher, dass Axis2 ausgeführt wird, indem Sie `http://localhost:8080/axis2` öffnen.

Öffnen Sie die Axis2-Webverwaltungsseite und laden Sie die im vorherigen Schritt generierte .aar-Datei hoch.

© 2016-2022 Altova GmbH



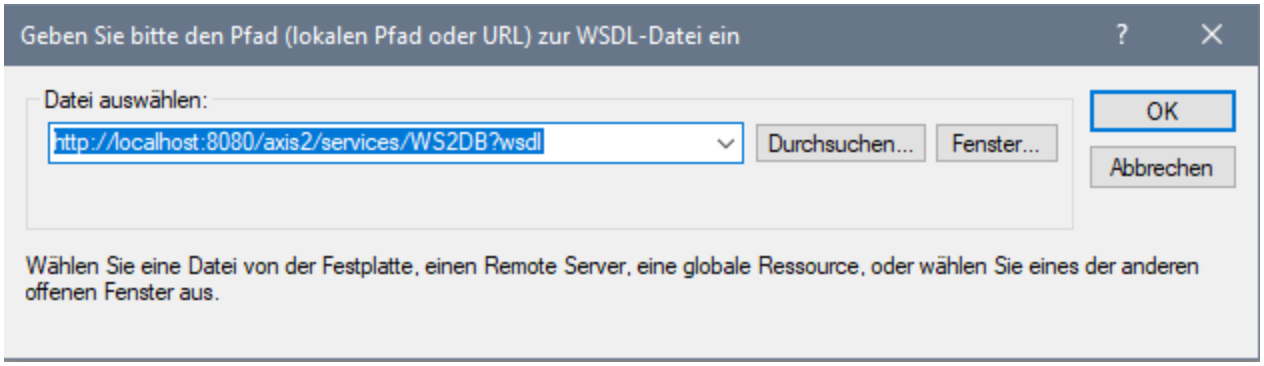
Der neue Webservice mit dem Namen "WS2DB" sollte jetzt auf der Axis2-Verwaltungsseite zur Verfügung stehen und aktiv sein.



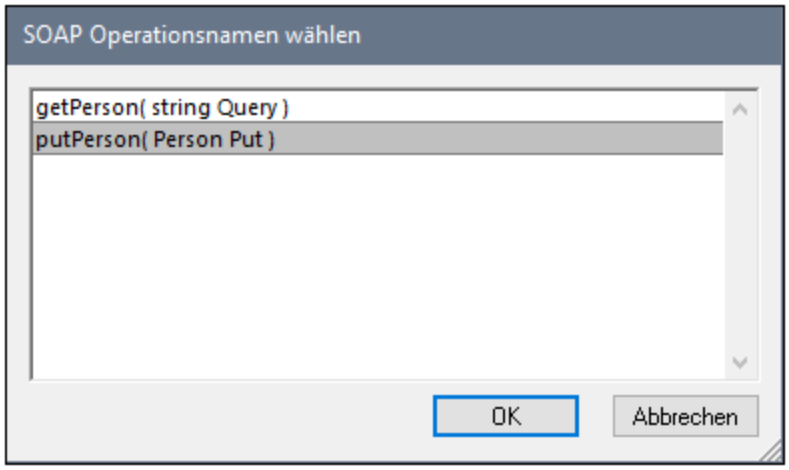
Aufrufen des Webservice

Wenn Sie die obige Anleitung genau befolgt haben, steht der Webservice unter `http://localhost:8080/axis2/services/WS2DB?wsdl` zur Verfügung. Wir verwenden zum Aufrufen des soeben bereitgestellten Webservice XMLSpy. Sie können natürlich auch andere SOAP-Clients dafür verwenden.

Starten Sie XMLSpy (Enterprise Edition). Klicken Sie im Menü **SOAP** auf **Neuen SOAP-Request erzeugen**. Geben Sie die WSDL-Datei des Webservice ein, wenn Sie dazu aufgefordert werden (in diesem Beispiel `http://localhost:8080/axis2/services/WS2DB?wsdl`).



Klicken Sie auf OK. Wenn Sie aufgefordert werden, eine SOAP-Operation auszuwählen, wählen Sie die Operation **putPerson** aus.



XMLSpy generiert daraufhin einen Request wie den unten gezeigten.

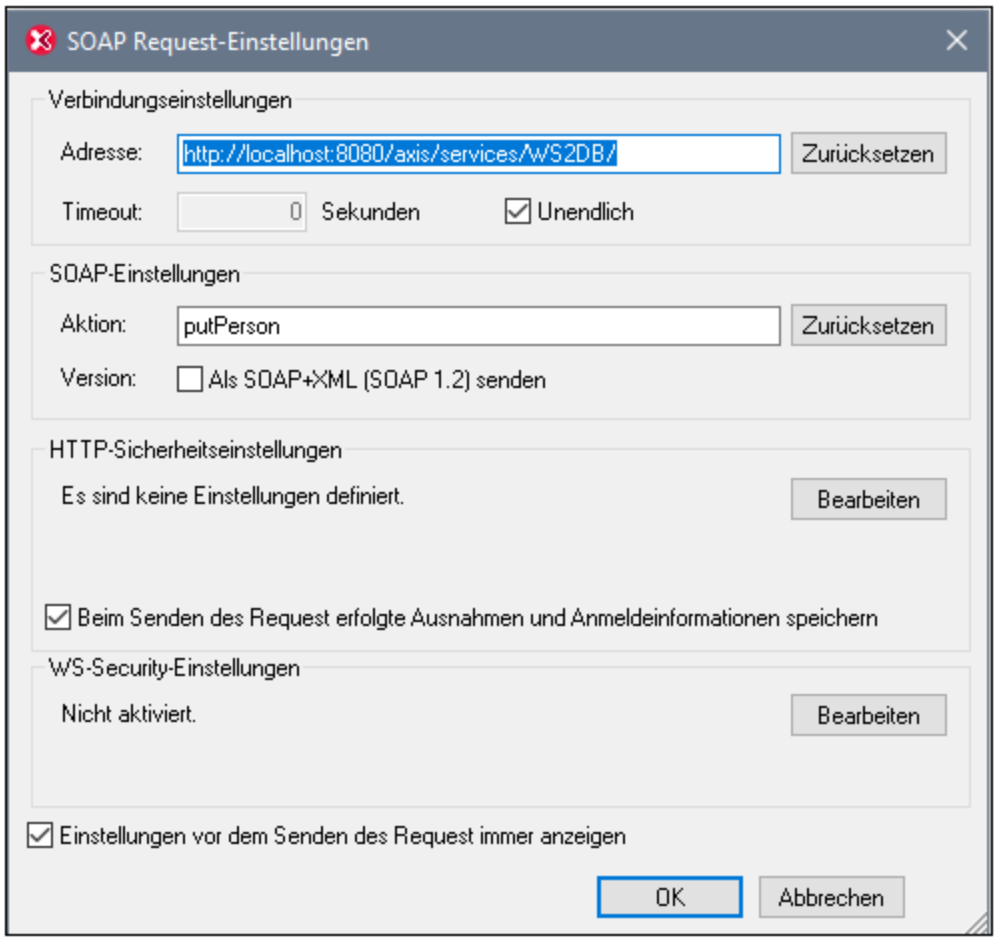
© 2016-2022 Altova GmbH

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:m0="http://www.altova.com/WS2DB.xsd">
   <SOAP-ENV:Body>
      <putPerson SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <Put xsi:type="m0:Person">
            <First xsi:type="xsd:string">String</First>
            <Last xsi:type="xsd:string">String</Last>
            <Title xsi:type="xsd:string">String</Title>
            <PhoneEXT xsi:type="xsd:string">String</PhoneEXT>
            <Email xsi:type="xsd:string">String</Email>
         </Put>
      </putPerson>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Ersetzen Sie "String" durch den Wert, der als Input für den Webservice fungieren soll. Für dieses Beispiel könnten Sie einen Request wie den unten stehenden verwenden:

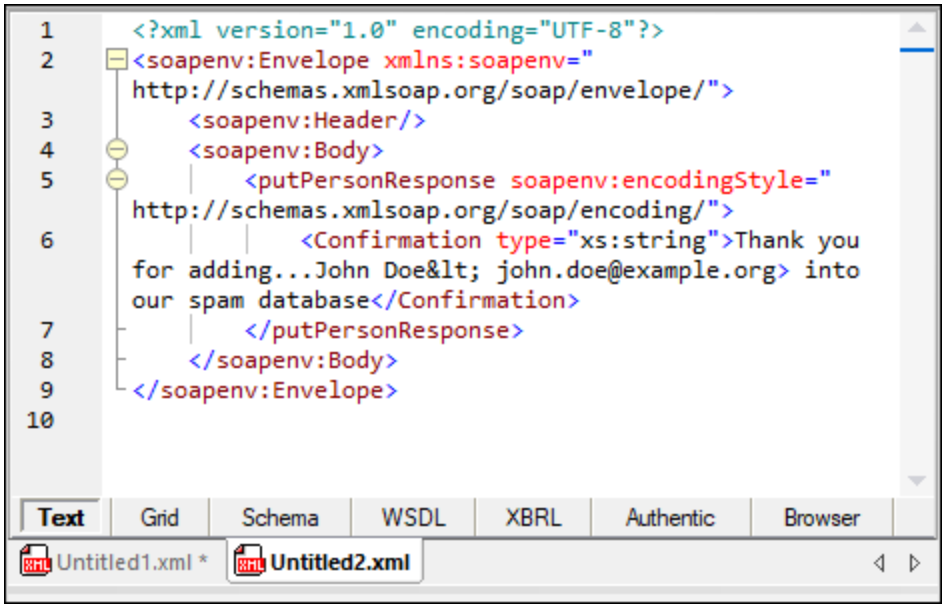
```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:m0="http://www.altova.com/WS2DB.xsd">
   <SOAP-ENV:Body>
      <putPerson SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <Put xsi:type="m0:Person">
            <First xsi:type="xsd:string">John</First>
            <Last xsi:type="xsd:string">Doe</Last>
            <Title xsi:type="xsd:string">Mr</Title>
            <PhoneEXT xsi:type="xsd:string">123</PhoneEXT>
            <Email xsi:type="xsd:string">john.doe@example.org</Email>
         </Put>
      </putPerson>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Sie können den obigen Request nun an den Server senden. Klicken Sie im Menü **SOAP** auf **Request an Server senden**. Daraufhin wird das Dialogfeld "SOAP Request-Einstellungen" angezeigt.



Klicken Sie auf **OK**. XMLSpy startet den Aufruf und zeigt im Editor die Antwort an. In der Abbildung unten sehen Sie eine erfolgreiche Antwort.

© 2016-2022 Altova GmbH



Auf dieselbe oben beschriebene Art können Sie auch die von diesem Webservice bereitgestellte Operation **`getPerson`** aufrufen. Eine schrittweise Anleitung dazu finden Sie im C#-Tutorial im Abschnitt "Aufrufen des Webservice[1284]". Im C#-Tutorial ist nur die WSDL URL eine andere; alle anderen Anleitungen zum Aufrufen des Webservice sind gleich.

8 Komplexe Mapping-Szenarien

Altova Website: Datenintegrationstool

In diesem Abschnitt sind komplexe Mapping-Szenarien beschrieben. Er enthält die folgenden Kapitel:

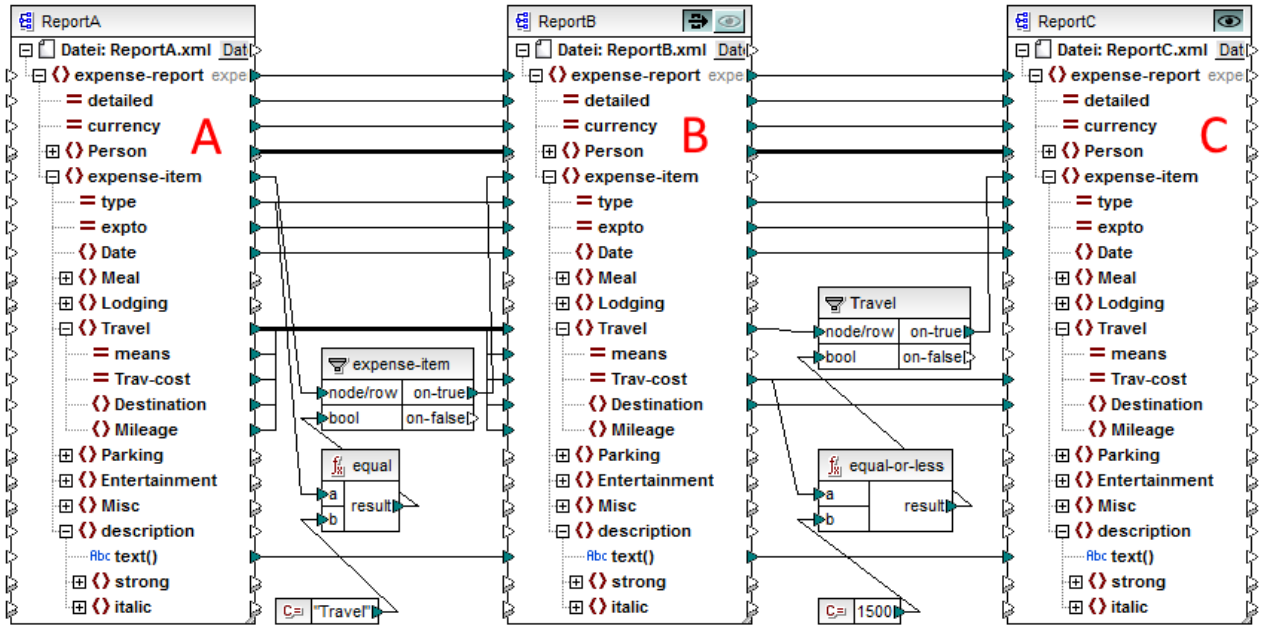
- Verkettete Mappings 1299
- Mappen von Node-Namen 1309
- Mapping-Regeln und -Strategien 1330
- Ausführen von Mappings mit Authentifizierung 1353
- Verarbeitung mehrerer Input- oder Output-Dateien 1371
- Parsen und Serialisieren von Strings 1382
- StyleVision-Ausgabefenster 1395
- Generieren von Mapping-Dokumentation 1399

© 2016-2022 Altova GmbH

8.1 Verkettete Mappings

MapForce unterstützt Mappings, die aus mehreren Komponenten in einer Mapping-Kette bestehen. Verkettete Mappings sind Mappings, in denen mindestens eine Komponente gleichzeitig als Quell- und Zielkomponente fungiert. Mit Hilfe einer solchen Komponente wird eine Ausgabe erzeugt, die später als Input für den nächsten Mapping-Schritt in der Kette verwendet wird. Eine solche Komponente wird als "Zwischenkomponente" bezeichnet.

So sehen Sie etwa im unten gezeigten Mapping einen Spesenbericht (im XML-Format), der in zwei Phasen verarbeitet wird. In dem Teil des Mappings von A auf B werden nur die Spesen herausgefiltert, die als "Travel" (Reisekosten) markiert sind. Im Mapping von B auf C werden nur die Reisekosten herausgefiltert, die weniger als 1500 betragen. Die Komponente B ist die "Zwischenkomponente", da sie sowohl Input- als auch Output-Verbindungen hat. Dieses Mapping steht unter folgendem Pfad zur Verfügung: **<Dokumente>\Altova\MapForce2023\MapForceExamples\ChainedReports.mfd**.



ChainedReports.mfd

In verketteten Mappings wird eine Funktion namens "Weiterleitung" verwendet. Die Weiterleitungsfunktion ist eine Vorschaufunktion, mit der Sie die in verschiedenen Phasen eines verketteten Mappings erzeugte Ausgabe im Fenster "Ausgabe" anzeigen können. So können Sie z.B. im oben gezeigten Mapping sowohl eine Vorschau auf das XML-Ergebnis des Mappings von A auf B anzeigen und speichern als auch dasselbe für das XML-Ergebnis des Mappings von B auf C tun.

Anmerkung:
Die Funktion "Weiterleitung" steht nur für dateibasierte "Zwischenkomponenten" (z.B. XML-, CSV-, TXT-Dateien) zur Verfügung. Datenbankkomponenten können als Zwischenkomponenten verwendet werden, doch wird die Schaltfläche "Weiterleitung" nicht angezeigt. Die Zwischenkomponente wird für die Ausgabevorschau oder die Codegenerierung immer von Grund auf neu generiert. Dies wäre mit einer Datenbank nicht möglich, da sie vor jeder Neugenerierung gelöscht werden müsste.

Wenn das Mapping von MapForce Server oder durch generierten Code ausgeführt wird, so wird die gesamte Mapping-Kette ausgeführt. Mit dem Mapping werden in den einzelnen Schritten der Kette die erforderlichen Ausgabedateien generiert. Die Ausgabe eines Mapping-Schritts wird als Input für den darauf folgenden Mapping-Schritt weitergeleitet.

Auch für Zwischenkomponenten können dynamische Dateinamen generiert werden, d.h. diese Zwischenkomponenten gestatten Verbindungen mit dem Datenelement "Datei" aus dem Mapping, wenn die Komponente entsprechend konfiguriert wurde. Nähere Informationen dazu finden Sie unter Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 1371.

Schaltfläche "Vorschau"

Sowohl die Komponente B als auch die Komponente C verfügt über eine Vorschau-Schaltfläche. Auf diese Art können Sie sowohl das Mapping-Zwischenergebnis von B als auch das endgültige Ergebnis des verketteten Mappings der Komponente C in MapForce anzeigen. Klicken Sie auf die Vorschau-Schaltfläche der jeweiligen Komponente und anschließend auf "Ausgabe", um das Mapping-Ergebnis zu sehen.

"Zwischenkomponenten", bei denen die Schaltfläche "Weiterleitung" aktiv ist, können nicht in der Vorschau angezeigt werden. Ihre Vorschau-Schaltfläche wird automatisch deaktiviert, da eine Vorschau bei gleichzeitiger Datenweiterleitung nicht sinnvoll ist. Um die Ausgabe einer solchen Komponente zu sehen, klicken Sie zuerst auf die Schaltfläche "Weiterleitung" um sie zu deaktivieren und klicken Sie anschließend auf die Vorschau-Schaltfläche.

Schaltfläche "Weiterleitung"

Die Zwischenkomponente B verfügt in der Titelleiste der Komponente über eine zusätzliche Schaltfläche mit dem Namen "Weiterleitung".

Wenn die Schaltfläche "Weiterleitung" **aktiv** ist , werden alle Daten auf einmal - von Komponente A auf Komponente B und weiter auf Komponente C - in das Vorschaufenster gemappt. Es werden zwei separate Datengruppen generiert:

- das Ergebnis des Mappings von Komponente A auf die Zwischenkomponente B
- das Ergebnis des Mappings der Zwischenkomponente B auf die Zielkomponente C.

Wenn die Schaltfläche "Weiterleitung" **deaktiviert** ist, wird nur ein Teil der kompletten Mapping-Kette ausgeführt. Welche Daten generiert werden, hängt davon ab, welche Vorschau-Schaltfläche aktiv ist:

- Wenn die Schaltfläche "Vorschau" der Komponente B aktiv ist, so wird das Ergebnis des Mappings von Komponente A auf Komponente B generiert. Die Mapping-Kette endet somit bei Komponente B. Die Komponente C wird in der Vorschau gänzlich ignoriert.
- Wenn die Schaltfläche "Vorschau" der Komponente C aktiv ist, so wird das Ergebnis des Mappings von Komponente B auf Komponente C generiert. Da die Schaltfläche "Weiterleitung" deaktiviert ist, wurde die automatische Verkettung zur Komponente B unterbrochen. Nur der rechte Teil des Mappings wird ausgeführt. Komponente A wird nicht verwendet.

Wenn die Schaltfläche "Weiterleitung" deaktiviert ist, ist es wichtig, dass die in den Feldern "XML-Input-

 © 2016-2022 Altova GmbH

Datei" und "XML-Output-Datei" definierte Zwischenkomponente denselben Namen hat. Damit stellen Sie sicher, dass die bei der Vorschau des Mappings zwischen A und B als Output-Datei generierte Datei als Input-Datei verwendet wird, wenn Sie eine Vorschau des Mappings zwischen B und C anzeigen. Auch im generierten Code oder bei der Ausführung mit MapForce Server gewährleisten Sie dadurch, dass die Mapping-Kette nicht unterbrochen wird.

Wenn das Mapping von MapForce Server oder generierten Code ausgeführt wird, so wird, wie bereits erwähnt, die Ausgabe aller Komponenten generiert. In diesem Fall werden die Einstellungen der Weiterleitungsschaltfläche von Komponente B sowie die der gerade ausgewählte Vorschaukomponente ignoriert. Wenn Sie das oben gezeigte Mapping als Beispiel nehmen, werden die folgenden zwei Ergebnisdateien generiert:

1. Die Ausgabedatei, die das Ergebnis des Mappings von Komponente A auf B bildet.
2. Die Ausgabedatei, die das Ergebnis des Mappings von Komponente B auf C bildet.

In den folgenden Abschnitten, Beispiel: Weiterleitung aktiv 1301 und Beispiel: Weiterleitung deaktiviert 1306, wird näher erläutert, welche Unterschiede bei der Übertragung der Quelldaten entstehen, wenn die Weiterleitungsschaltfläche aktiv bzw. deaktiviert ist.

8.1.1 Beispiel: Weiterleitung aktiv

Das in diesem Beispiel verwendete Mapping (**ChainedReports.mfd**) steht im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** zur Verfügung. In diesem Mapping wird eine XML-Datei namens **ReportA.xml,** die Reisekosten enthält und wie unten gezeigt aussieht, verarbeitet. Die Namespace-Deklaration und einige `expense-item`-Datenelemente wurden aus Gründen der Übersichtlichkeit weggelassen:

```
<?xml version="1.0" encoding="UTF-8"?>
<expense-report currency="USD" detailed="true">
   <Person>
      <First>Fred</First>
      <Last>Landis</Last>
      <Title>Project Manager</Title>
      <Phone>123-456-78</Phone>
      <Email>f.landis@nanonull.com</Email>
   </Person>
   <expense-item type="Travel" expto="Development">
      <Date>2003-01-02</Date>
      <Travel Trav-cost="337.88">
         <Destination/>
      </Travel>
      <description>Biz jet</description>
   </expense-item>
   <expense-item type="Lodging" expto="Sales">
      <Date>2003-01-01</Date>
      <Lodging Lodge-cost="121.2">
         <Location/>
      </Lodging>
```

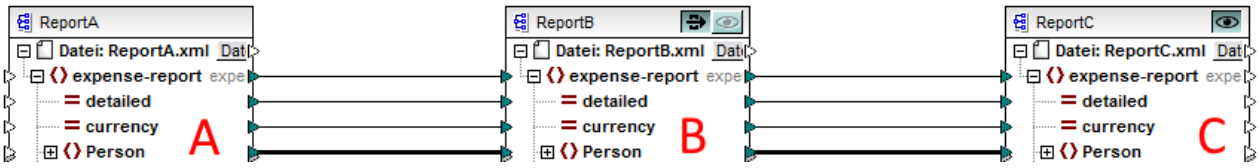
```
      <description>Motel mania</description>
   </expense-item>
   <expense-item type="Travel" expto="Marketing">
      <Date>2003-02-02</Date>
      <Travel Trav-cost="2000">
         <Destination/>
      </Travel>
      <description>Hong Kong</description>
   </expense-item>
</expense-report>
```

ReportA.xml

Ziel des Mappings ist die Erzeugung zweiter Berichte anhand der obigen Datei:

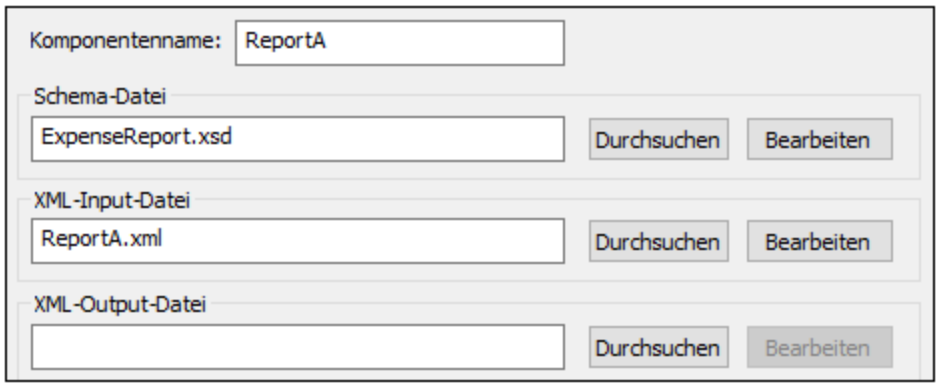
- **ReportB.xml** - dieser Bericht soll nur die Reisekosten vom Typ "Travel" enthalten.
- **ReportC.xml** - dieser Bericht soll nur die Reisekosten enthalten, die vom Typ "Travel" sind und 1500 nicht übersteigen.

Die Weiterleitungsschaltfläche der Zwischenkomponente des Mappings (Komponente B) wurde zu diesem Zweck aktiviert , wie unten gezeigt, sodass das Mapping in Phasen ausgeführt wird: von A auf B und anschließend von B auf C. Die durch die Zwischenkomponente erzeugte Ausgabe wird als Input für das Mapping zwischen B und C verwendet.



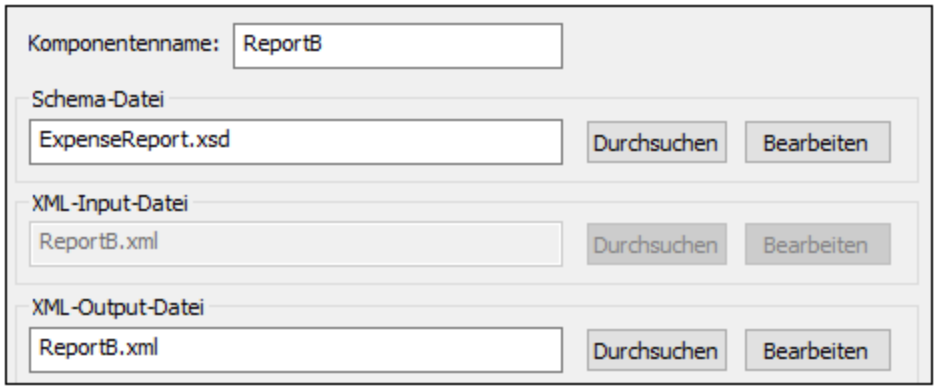
Die Namen der in den einzelnen Phasen der Mapping-Kette generierten Ausgabedateien sind in den Einstellungen der einzelnen Komponenten definiert. (Um die Komponenteneinstellungen zu öffnen, klicken Sie mit der rechten Maustaste auf die Komponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften**). Die erste Komponente wurde so konfiguriert, dass sie Daten aus einer XML-Datei namens **ReportA.xml** liest. Da es sich hierbei um eine Quellkomponente handelt, ist das Feld **XML-Output-Datei** hier nicht von Bedeutung und wurde leer gelassen.

© 2016-2022 Altova GmbH



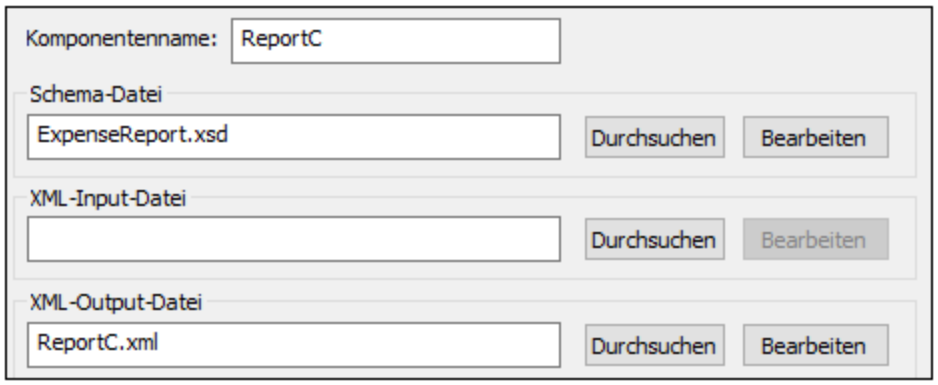
Einstellungen der Quellkomponente

Wie unten gezeigt, wurde für die zweite Komponente (**ReportB**) eine Output-Datei namens **ReportB.xml** konfiguriert. Beachten Sie, dass das Feld **XML-Input-Datei** ausgegraut ist. Wenn die Weiterleitung wie in diesem Beispiel aktiv ist, so wird das Feld **XML-Input-Datei** der Zwischenkomponente automatisch deaktiviert. Ein Input-Dateiname ist hier nicht nötig, damit das Mapping ausgeführt wird, da die in dieser Phase des Mappings erzeugte Ausgabe in einer temporären Datei gespeichert wird und im Mapping weiterverwendet wird. Wenn, wie unten gezeigt, eine XML-Output-Datei definiert wird, so wird dieser Name als Name der Zwischenausgabedatei verwendet. Wenn keine **XML-Output-Datei** definiert ist, so wird automatisch ein Standarddateiname verwendet.



Einstellungen der Zwischenkomponente

Schlussendlich wird die dritte Komponente konfiguriert. Sie soll eine Ausgabedatei namens **ReportC.xml** erzeugen. Das Feld **XML-Input-Datei** ist hier nicht relevant, da es sich hier um eine Zielkomponente handelt.

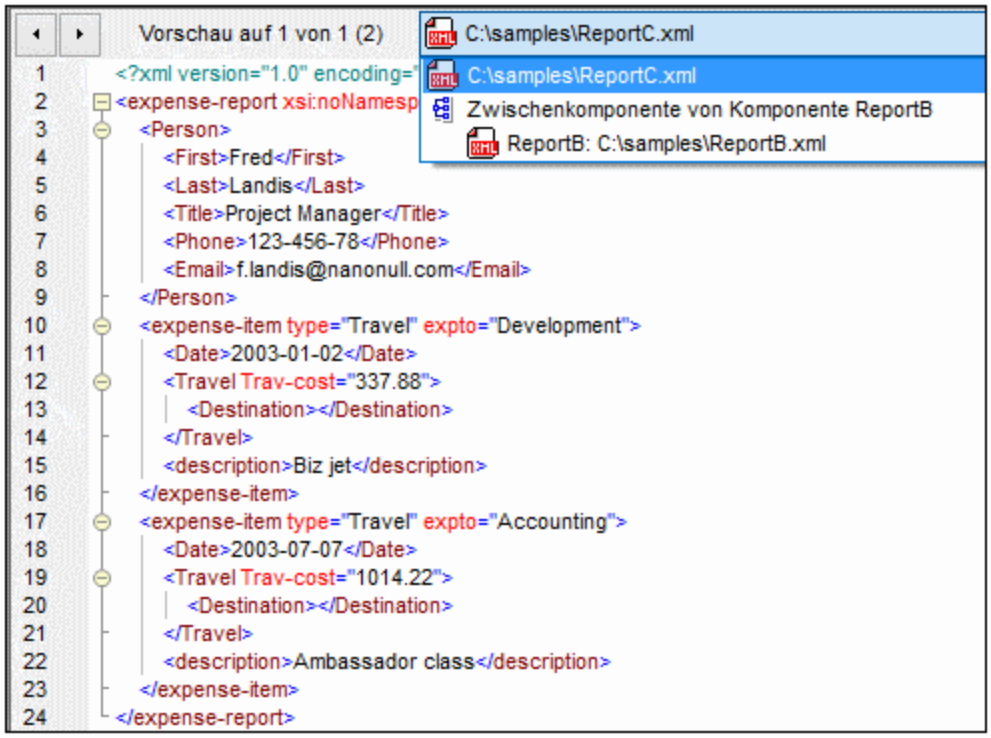


Einstellungen der Zielkomponente

Wenn Sie durch Klick auf das Register **Ausgabe** im Mapping-Fenster eine Vorschau auf das Mapping anzeigen, werden in der Ausgabe, wie erwartet, zwei Dateien angezeigt:

1. **ReportB.xml**, welcher das Ergebnis des Mappings A auf B darstellt
2. **ReportC.xml**, welcher das Ergebnis des Mappings B auf C darstellt

Um auszuwählen, welche der beiden generierten Ausgabedateien in Fenster "Ausgabe" angezeigt werden soll, klicken Sie entweder auf die Pfeilschaltflächen oder wählen Sie den gewünschten Eintrag aus der Dropdown-Liste aus.

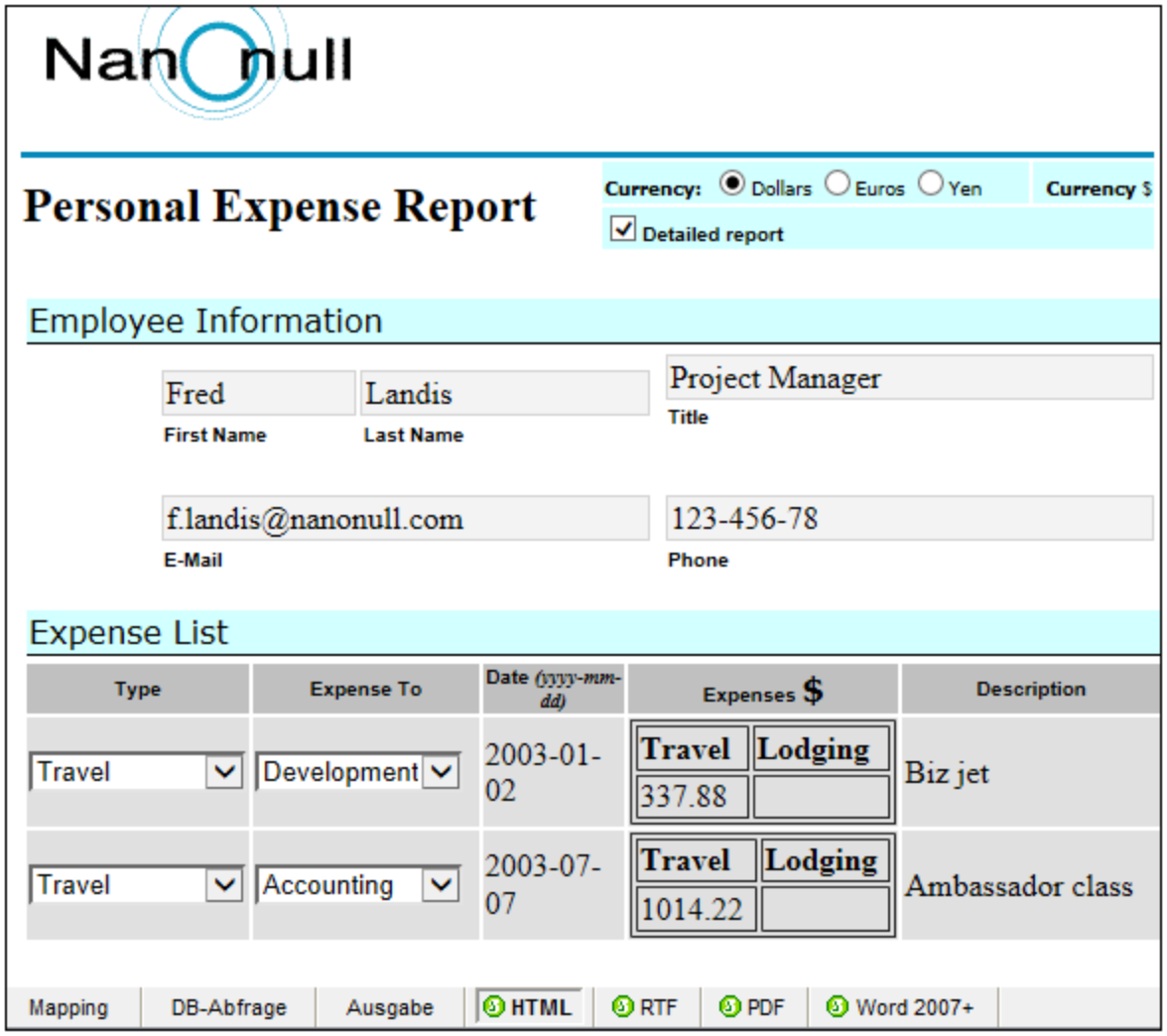


Generierte Ausgabedateien

 © 2016-2022 Altova GmbH

Wenn das Mapping mit MapForce ausgeführt wird, gibt die Einstellung "Direkt in die endgültigen Output-Dateien schreiben" (aus dem Menü **Extras | Optionen | Allgemein**) an, ob die Zwischendateien als temporäre oder als physischen Dateien gespeichert werden sollen. Beachten Sie, dass diese Einstellung nur dann gilt, wenn die Vorschau direkt in MapForce erfolgt. Würde dieses Mapping mit MapForce Server oder durch generierten Code ausgeführt, würden in jeder Phase der Mapping-Kette tatsächlich Dateien erzeugt.

Wenn StyleVision installiert ist und der Zielkomponente (wie in diesem Beispiel) eine StyleVision Power Stylesheet (SPS)-Datei zugewiesen wurde, so kann die endgültige Mapping-Ausgabe als HTML-, RTF-, PDF- oder Word-Datei angezeigt (und gespeichert) werden. Um diese Ausgabe in MapForce zu generieren und anzuzeigen, klicken Sie auf das Register mit dem entsprechenden Namen.

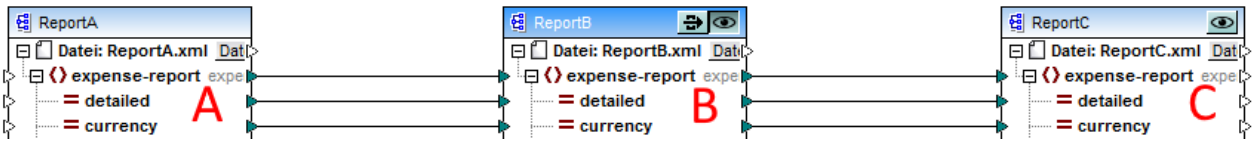


Generierte HTML-Ausgabe

Beachten Sie, dass nur die Ausgabe der endgültigen Zielkomponente in der Mapping-Kette angezeigt wird. Um die StyleVision-Ausgabe von Zwischenkomponenten anzuzeigen, müssten Sie die Weiterleitungsschaltfläche deaktivieren und die Vorschau der Zwischenkomponente anzeigen (wie im Beispiel: Weiterleitung deaktiviert 1306 gezeigt).

8.1.2 Beispiel: Weiterleitung deaktiviert

Das in diesem Beispiel verwendete Mapping (**ChainedReports.mfd**) steht im Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** zur Verfügung. In diesem Beispiel wird gezeigt, inwiefern sich die Generierung der Ausgabe unterscheidet, wenn die Schaltfläche "Weiterleitung" in der Zwischenkomponente deaktiviert ist.

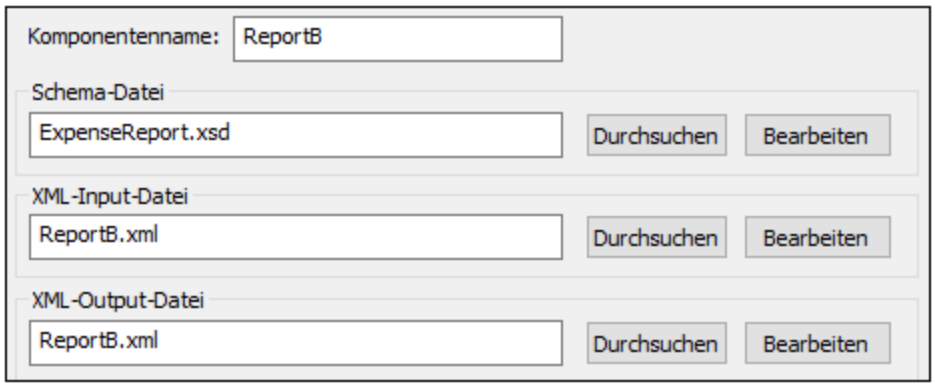


Wie im Beispiel: Weiterleitung aktiv[1301] erläutert, ist das Ziel des Mappings die Erzeugung zweier separater Berichte. Im vorherigen Beispiel war die Schaltfläche "Weiterleitung" aktiv und beide Berichte wurden wie erwartet generiert und konnten auf dem Register **Ausgabe** angezeigt werden. Wenn Sie aber nur einen der Berichte anzeigen möchten (entweder **ReportB.xml** oder **ReportC.xml**), so muss die Weiterleitungsschaltfläche deaktiviert werden ( ), d.h. es ist dann sinnvoll, die Weiterleitungsschaltfläche zu deaktivieren, wenn Sie folgendes Ziel erreichen wollen:

- Nur Vorschau auf die durch das Mappen von A auf B generierten Ausgabe, wobei der Mapping-Abschnitt B auf C ignoriert wird.
- Nur Vorschau auf die durch das Mappen von B auf C generierte Ausgabe, wobei der Mapping-Abschnitt A auf B ignoriert wird.

Wenn Sie die Weiterleitungsschaltfläche, wie oben gezeigt, deaktivieren, können Sie auswählen, ob Sie entweder eine Vorschau auf **ReportB** oder eine auf **ReportC** sehen möchten (Beachten Sie, dass beide Vorschauschaltflächen haben).

Wenn Sie die Weiterleitungsschaltfläche deaktivieren, können Sie auch auswählen, welche Input-Datei von der Zwischenkomponente gelesen werden soll. In den meisten Fällen sollte es sich hierbei um dieselbe Datei handeln, wie im Feld **XML-Output-Datei** definiert (wie in diesem Beispiel gezeigt).



Einstellungen der Zwischenkomponente

Dass in der Zwischenkomponente dieselbe Input- und Output-Datei definiert ist, ist vor allem dann wichtig, wenn Sie beabsichtigen, Code anhand des Mappings zu generieren oder das Mapping mit MapForce Server auszuführen. Wie bereits erwähnt, werden in diesen Umgebungen alle von den einzelnen Komponenten in der

© 2016-2022 Altova GmbH

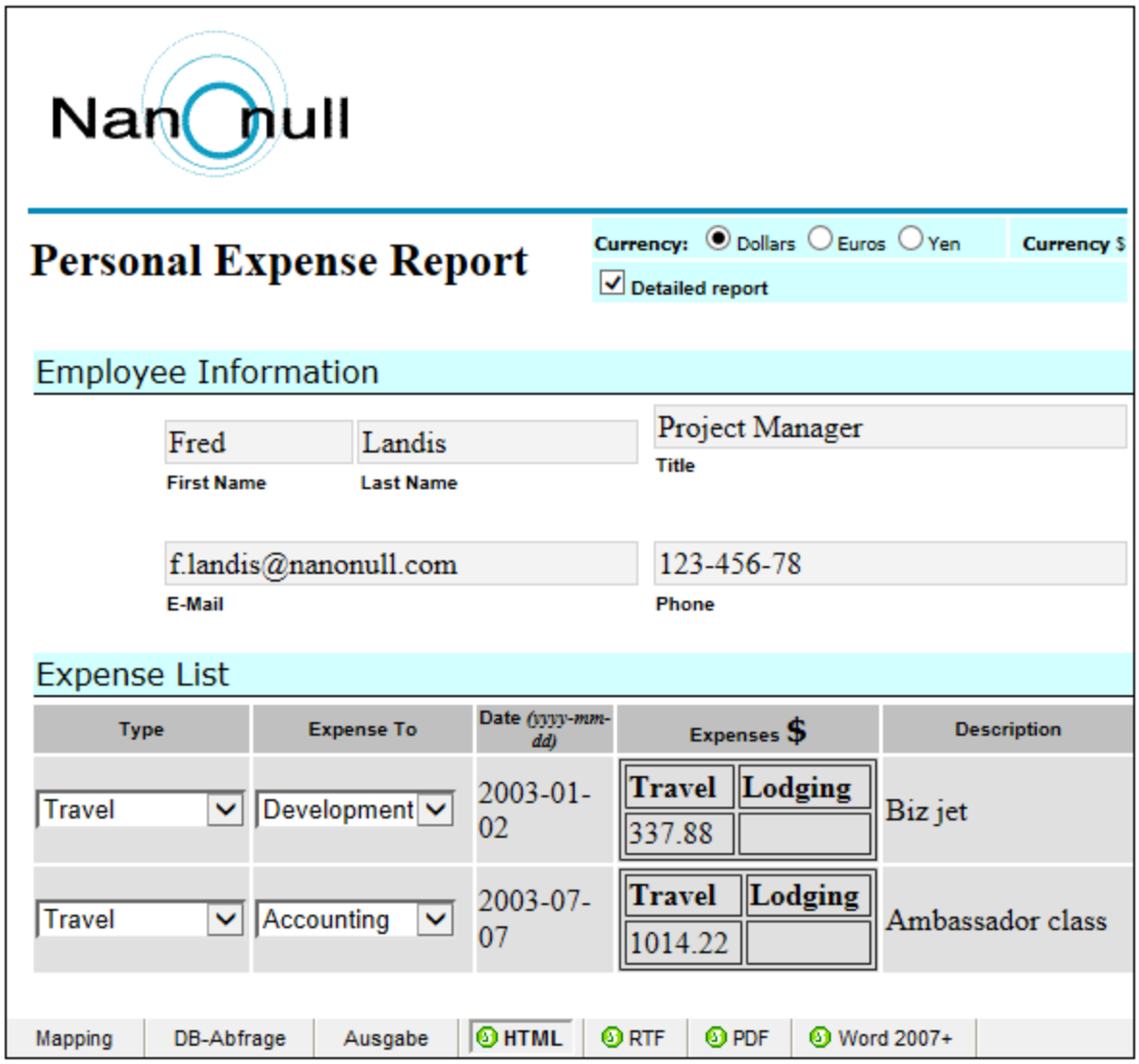
Mapping-Kette erzeugten Ausgaben generiert. Daher ist es normalerweise sinnvoll, dass die Zwischenkomponente eine Datei für die Verarbeitung erhält (in diesem Fall **ReportB.xml**) und dieselbe Datei an das darauf folgende Mapping weiterleitet, anstatt nach einem anderen Dateinamen zu suchen. Beachten Sie, dass die Verwendung unterschiedlicher Dateinamen für die Input- und Output-Datei in der Zwischenkomponente (bei deaktivierter Weiterleitungsschaltfläche) im generierten Code oder bei der MapForce Server-Ausführungen zu Fehlern wie "Das System kann die definierte Datei nicht finden" führen kann.

Wenn Sie in der dritten Komponente (**ReportC**) auf die Vorschau-Schaltfläche  klicken und versuchen, eine Vorschau auf das Mapping in MapForce anzuzeigen, werden Sie sehen, dass es zu einem Ausführungsfehler kommt. Dies ist ein erwartetes Verhalten, da gemäß den Einstellungen als Input eine Datei namens **ReportB.xml** erwartet wird. Eine solche Datei wurde vom Mapping aber noch nicht erzeugt (da die Weiterleitungsschaltfläche nicht aktiv ist und nur der Mapping-Abschnitt B auf C ausgeführt wurde). Sie können dieses Problem ganz einfach folgendermaßen beheben:

1. Klicken Sie auf die Vorschauschaltfläche der Zwischenkomponente.
2. Klicken Sie auf das Register **Ausgabe**, um eine Vorschau auf das Mapping anzuzeigen.
3. Speichern Sie die erzeugte Ausgabedatei im selben Ordner wie das Mapping (**<Dokumente>\Altova\MapForce2023\MapForceExamples**) als **ReportB.xml**.

Wenn Sie jetzt erneut auf die Vorschauschaltfläche der dritten Komponente klicken (**ReportC**), wird der Fehler nicht mehr angezeigt.

Wenn die Weiterleitungsschaltfläche deaktiviert ist, können Sie für die einzelnen Komponenten auch eine Vorschau auf die mit StyleVision generierte Ausgabe anzeigen, wenn diesen Komponenten eine StyleVision Power StyleSheet (SPS)-Datei zugewiesen wurde. So können Sie etwa auch die HTML-Version des Zwischenberichts (zusätzlich zum Endbericht) anzeigen:



HTML-Ausgabe der Zwischenkomponente

© 2016-2022 Altova GmbH

8.2 Mappen von Node-Namen

Wenn Sie ein Mapping mit MapForce erstellen, ist das Ziel in den meisten Fällen, *Werte* aus einer Quelldatei zu lesen und *Werte* in eine Zieldatei zu schreiben. In manchen Fällen benötigen Sie jedoch nicht nur Zugriff auf die Node-*Werte* aus der Quelldatei, sondern auch auf die Node-Namen. Sie könnten z.B. ein Mapping benötigen, in dem die Element- oder Attributnamen (nicht die Werte) aus einer XML-Quelldatei gelesen und in Element- oder Attributwerte (nicht Namen) in einer XML-Zieldatei konvertiert werden.

Betrachten Sie das folgende Beispiel: Sie haben eine XML-Datei, die eine Liste von Produkten enthält. Jedes Produkt hat das folgende Format:

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

Ihr Ziel ist es, Informationen über jedes Produkt in Namen-Wert-Paare zu konvertieren, z.B.:

```
<product>
   <attribute name="id" value="1" />
   <attribute name="color" value="red" />
   <attribute name="size" value="10" />
</product>
```

In diesem Szenario benötigen Sie im Mapping Zugriff auf den Node-Namen. Beim *dynamischen* Zugriff auf Node-Namen können Sie Datenkonvertierungen wie die oben beschriebene durchführen.

Anmerkung: Sie können die Transformation auch mit Hilfe der Funktionen **node-name**[1071] und **static-node-name**[1073] aus der core-Bibliothek durchführen. In diesem Fall müssen Sie jedoch genau wissen, welche Element-Namen in der Quelldatei vorkommen und Sie müssen jedes einzelne Element manuell mit der Zielkomponente verbinden. Diese Funktionen genügen oft außerdem nicht, wenn Sie z.B. Nodes nach Namen filtern oder gruppieren müssen oder den Datentyp des Node im Mapping bearbeiten müssen.

Ein dynamischer Zugriff auf Node-Namen ist nicht nur beim Lesen von Node-Namen, sondern auch beim Schreiben von Node-Namen möglich. In einem Standard-Mapping sind die Namen von Attributen oder Elementen in einer Zieldatei immer noch vor Ausführung des Mappings bekannt; sie stammen aus dem Schema, das der Komponente zugrunde liegt. Bei dynamischen Node-Namen können Sie hingegen neue Attribute oder Elemente erstellen, deren Namen vor Ausführung des Mappings nicht bekannt sind. Der Name des Attributs oder Elements kommt in diesem Fall aus dem Mapping selbst, nämlich aus jeder beliebigen von MapForce unterstützten Quelldatei.

Damit ein dynamischer Zugriff auf die Child-Elemente oder -Attribute eines Node möglich ist, muss der Node tatsächlich Child-Elemente oder -Attribute haben und darf nicht der XML-Node sein.

Dynamische Node-Namen werden unterstützt, wenn Sie von oder auf die folgenden Komponententypen mappen:

- XML
- CSV/FLF*

* MapForce Professional oder Enterprise Edition erforderlich.

Anmerkung: Im Fall von CSV/FLF bedeutet dynamischer Zugriff, Zugriff auf "Felder" anstelle von "Nodes", da CSV/FLF-Strukturen keine "Nodes" haben.

Wenn es sich bei der Zielkomponente des Mappings um eine CSV- oder FLF- (fixed-length field)-Datei handelt, müssen die Felder in den Komponenteneinstellungen definiert werden (Und es ist nicht möglich, den Namen, die Reihenfolge oder die Anzahl der Zielfelder zu ändern). Im Gegensatz zu XML-Dateien ist das Format von Textdateien festgelegt, daher kann nur der tatsächliche Feldwert, nicht aber der Feldname, die Anzahl der Felder oder die Reihenfolge des Felds bearbeitet werden.

Dynamische Node-Namen werden unterstützt in jeder der folgenden Mapping-Sprachen unterstützt: Built-In*, XSLT2, XSLT 3.0, XQuery*, C#*, C++*, Java*.

* MapForce Professional oder Enterprise Edition erforderlich.

Informationen zur Funktionsweise von dynamischen Node-Namen finden Sie unter Zugriff auf Node-Namen 1310. Ein Schritt-für-Schritt-Beispiel für ein solches Mapping finden Sie unter Beispiel: Mappen von Elementnamen auf Attributwerte 1322.

8.2.1 Zugriff auf Node-Namen

Wenn ein Node in einer XML-Komponente (oder ein Feld in einer CSV/FLF-Komponente) Child-Nodes hat, können sowohl der Name als auch der Wert jedes einzelnen Child-Node direkt im Mapping abgerufen werden. Diese Methode wird als "dynamische Node-Namen" bezeichnet. "Dynamisch" bezieht sich darauf, dass die Verarbeitung "on-the-fly" zur Mapping-Laufzeit und nicht auf Basis statischer noch vor Ausführung des Mappings bekannter Schemainformationen erfolgt. In diesem Kapitel wird näher erläutert, wie Sie dynamischen Zugriff auf Node-Namen erhalten und was Sie damit erreichen können.

Wenn Daten aus einer Quelldatei ausgelesen werden, bedeutet "dynamische Node-Namen", dass Folgendes möglich ist:

- Abrufen einer Liste aller Child-Nodes (oder Attribute) eines Node als Sequenz. Eine "Sequenz" in MapForce ist eine Liste von null oder mehr Datenelementen, die mit einer Zielkomponente verbunden werden können, sodass in der Zielkomponente dieselbe Anzahl an Datenelementen wie in der Quelldatei erstellt werden kann. Wenn z.B. ein Node fünf Attribute in der Quelldatei hat, so könnten in der Zieldatei fünf neue Elemente, eines für jedes Attribut, erstellt werden.
- Lesen nicht nur der Child-Node-Werte (wie bei einem Standard-Mapping), sondern auch der Namen dieser Nodes.

Wenn Daten in eine Zieldatei geschrieben werden sollen, bedeutet "dynamische Node-Namen", dass Folgendes möglich ist:

 © 2016-2022 Altova GmbH

- Erstellung neuer Nodes anhand von Namen aus dem Mapping (so genannter "dynamischer" Namen) anstelle von Namen, die von den Komponenteneinstellungen bereitgestellt werden (so genannte "statische" Namen).

Zur Veranschaulichung dynamischer Node-Namen wird in diesem Kapitel das folgende XML-Schema verwendet: **<Dokumente>\Altova\MapForce2023\MapForceExamples\Products.xsd**. Das Beispielinstanzdokument zu diesem Schema ist **Products.xml**. Um Schema und Instanzdatei zum Mapping-Bereich hinzuzufügen, wählen Sie den Befehl **Einfügen | XML-Schema/Datei** und navigieren Sie zum Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\Products.xml**. Wenn Sie aufgefordert werden, ein Root-Element auszuwählen, klicken Sie auf `products`.

Um dynamische Node-Namen für den Node `product` zu aktivieren, klicken Sie mit der rechten Maustaste darauf und wählen Sie einen der folgenden Kontextmenübefehle:

- **Attribute mit dynamischem Namen anzeigen,** wenn Sie Zugriff auf die Attribute des Node benötigen
- **Child-Elemente mit dynamischem Namen anzeigen**, wenn Sie Zugriff auf die Child-Elemente des Node benötigen

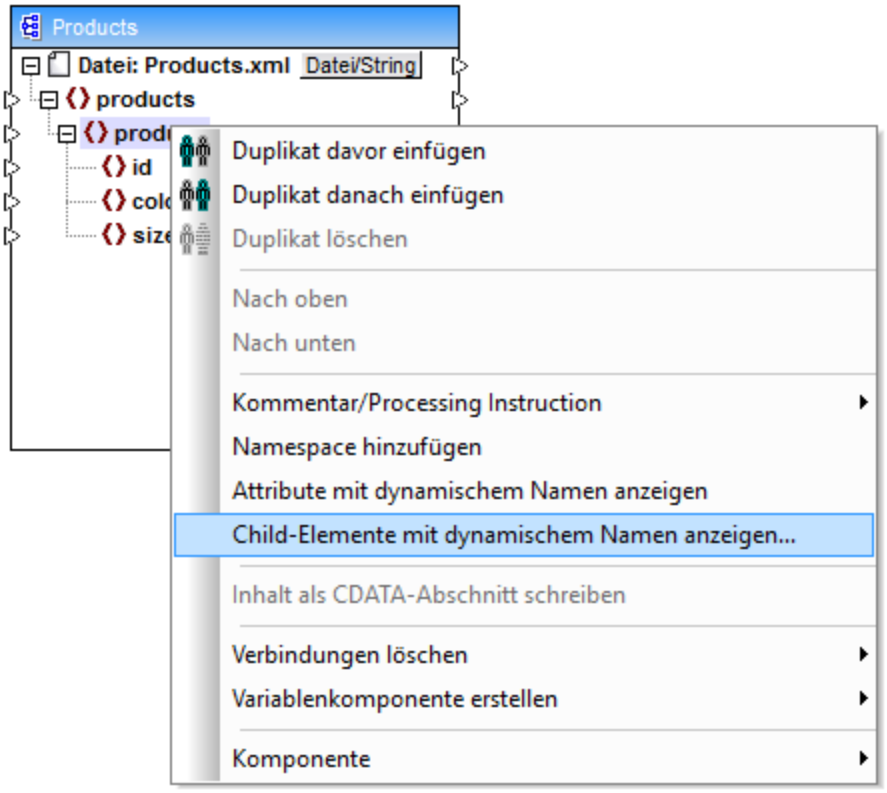


Abb. 1 Aktivieren dynamischer Node-Namen (für Child-Elemente)

Anmerkung: Die oben beschriebenen Befehle stehen nur für Nodes zur Verfügung, die Child-Nodes haben. Außerdem stehen die Befehle auch nicht für Root-Nodes zur Verfügung.

Wenn Sie bei einem Node in den dynamischen Modus wechseln, wird ein Dialogfeld wie das unten gezeigte, angezeigt. Wählen Sie für das Beispiel in diesem Kapitel die unten gezeigten Optionen aus; eine nähere Beschreibung zu diesen Optionen finden Sie unter Zugriff auf Nodes eines bestimmten Typs 1318.

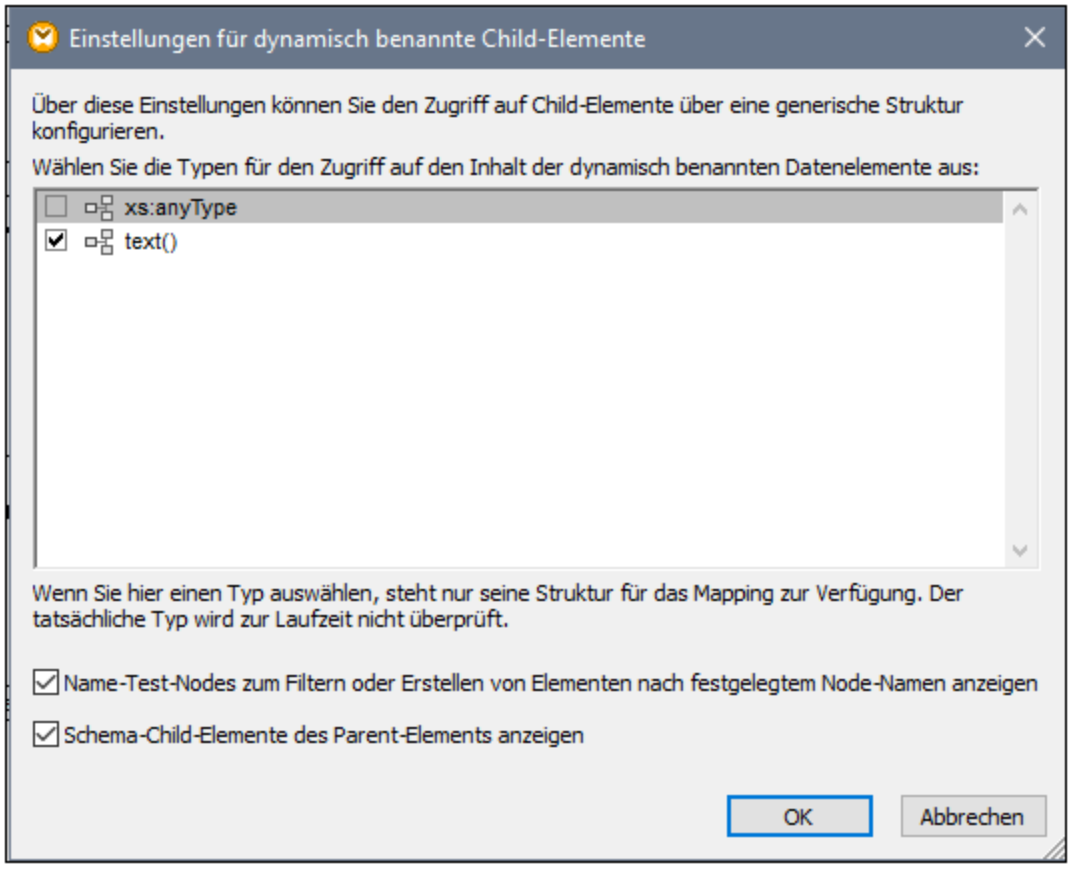


Abb. 2 Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente"

In Abb. 3 sehen Sie, wie die Komponente aussieht, wenn dynamische Node-Namen für den `product` Node aktiviert sind. Beachten Sie, wie sehr sich das Aussehen der Komponente jetzt geändert hat.

© 2016-2022 Altova GmbH

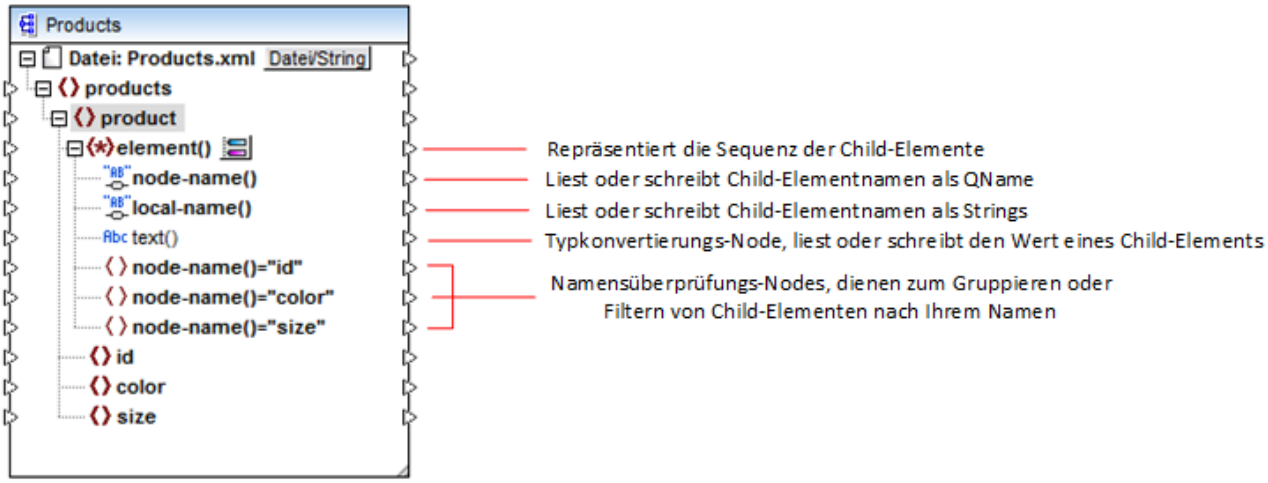


Abb.3 Aktivieren dynamischer Node-Namen (für Elemente)

Um die Komponente wieder zurück in den Standardmodus zu schalten, klicken Sie mit der rechten Maustaste auf den Node `product` und deaktivieren Sie die Option **Child-Elemente mit dynamischem Namen anzeigen** im Kontextmenü.

In der Abbildung unten sehen Sie, wie dieselbe Komponente aussieht, wenn der dynamische Zugriff auf Attribute eines Node aktiviert ist. Die Komponente wurde durch Rechtsklick auf das Element `product` und Auswahl des Kontextmenübefehls **Attribute mit dynamischem Namen anzeigen** definiert.

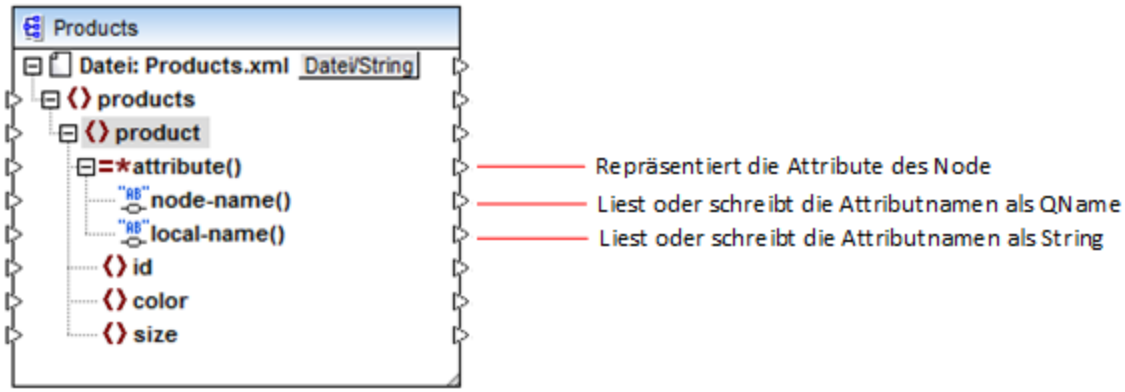


Abb. 4 Dynamische Node-Namen (für Attribute) sind aktiviert

Um die Komponente wieder zurück in den Standardmodus zu schalten, klicken Sie mit der rechten Maustaste auf den Node `product` und deaktivieren Sie die Option **Attribute mit dynamischem Namen anzeigen** im Kontextmenü.

Wie Sie in Abbildung 3 und 4 sehen, ändert sich das Aussehen der Komponente, wenn ein Node (in diesem Fall `product`) in den Modus "dynamischer Node-Name" wechselt. In diesem Modus ist nun Folgendes möglich:

- Lesen oder Schreiben einer Liste alle Child-Elemente oder -Attribute eines Node. Diese werden vom Datenelement `element()` bzw. `attribute()` bereitgestellt.

- Lesen oder Schreiben der Namen der einzelnen Child-Elemente oder -Attribute. Der Name wird von den Datenelementen `node-name()` und `local-name()` bereitgestellt.
- Lesen oder Schreiben des Werts von einzelnen Child-Elementen (bei Elementen) als spezifischer Datentyp. Dieser Wert wird vom Typkonvertierungs-Node (in diesem Fall dem Datenelement `text()`) bereitgestellt. Beachten Sie, dass nur Elemente Typkonvertierungs-Nodes haben. Attribute werden immer als "String"-Typ behandelt.
- Gruppieren oder Filtern generischer Child-Elemente nach Namen. Ein Beispiel finden Sie unter Beispiel: Gruppieren und Filtern von Nodes nach deren Namen 1325.

Im Folgenden finden Sie eine Beschreibung der Node-Typen, mit denen Sie im Modus "dynamischer Node-Name" arbeiten können.

element()

Dieser Node weist in einer Quellkomponente ein anderes Verhalten als in einer Zielkomponente auf. Er stellt in der Quellkomponente die Child-Elemente des Node als Sequenz bereit. In Abb.3 stellt `element()` eine Liste (Sequenz) aller Child-Elemente von `product` bereit. Die anhand des folgenden XML-Fragments erstellte Sequenz würde z.B. drei Datenelemente enthalten (da `product` drei Child-Elemente hat):

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

Beachten Sie, dass der tatsächliche Name und Typ der einzelnen Datenelemente in der Sequenz vom Node `node-name()` bzw. dem Typkonvertierungs-Node bereitgestellt wird (Beschreibung siehe unten). Um dies zu veranschaulichen, stellen Sie sich vor, Sie müssen Daten folgendermaßen aus einer XML-Quelldatei in eine XML-Zieldatei transformieren:



Abb. 6 Mappen von XML-Elementnamen auf Attributwerte (Aufgabe)

Das Mapping, mit dem Sie dies erreichen, sieht folgendermaßen aus:

 © 2016-2022 Altova GmbH

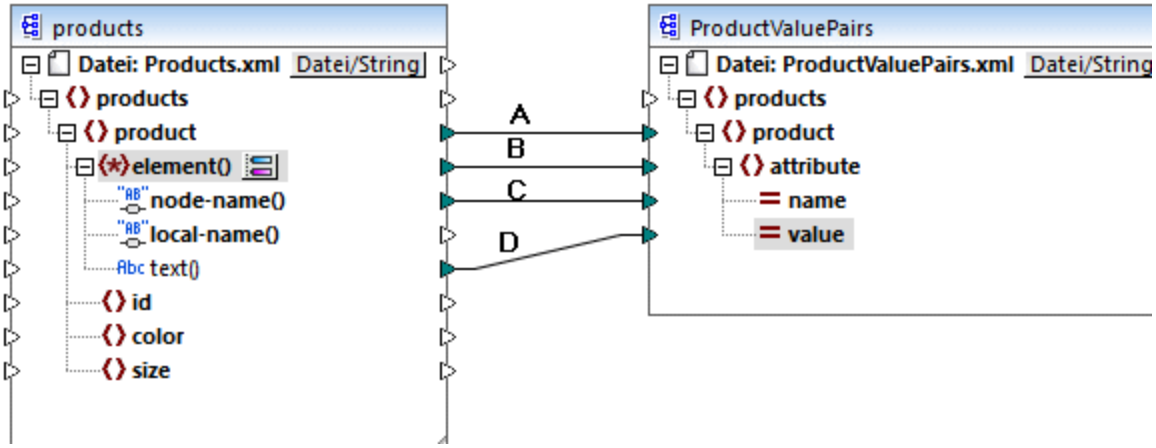


Abb. 7 Mappen von XML-Elementnamen auf Attributwerte (in MapForce)

Die Rolle von `element()` ist es hier, die Sequenz von Child-Elementen von `product`, bereitzustellen, während `node-name()` und `text()` den tatsächlichen Namen und Wert der einzelnen Datenelemente in der Sequenz bereitstellen. Zu diesem Mapping gibt es ein Tutorial-Beispiel, das unter Beispiel: Mappen von Elementnamen auf Attributwerte 1322 näher beschrieben ist.

`element()` selbst erstellt nichts in der Zielkomponente. Dies stellt eine Ausnahme der Grundregel "Erstelle für jedes Datenelement in der Quellkomponente ein Datenelement in der Zielkomponente" dar. Die eigentlichen Elemente werden (unter Verwendung des Werts von `node-name()`) durch die Typkonvertierungs- und (unter Verwendung der eigenen Namen) durch die Namensüberprüfungs-Nodes erstellt.

attribute()

Wie Sie in Abb. 4 sehen, ermöglicht dieses Datenelement zur Mapping-Laufzeit den Zugriff auf alle Attribute des Node. Es stellt in einer Quellkomponente die Attribute des damit verbundenen Quell-Node als Sequenz bereit. So würde die Sequenz im folgenden XML-Fragment etwas zwei Datenelemente enthalten (da `product` zwei Attribute hat):

```
<product id="1" color="red" />
```

Beachten Sie, dass der `attribute()`-Node nur den Wert der einzelnen Attribute in der Sequenz, und zwar immer als String-Typ, bereitstellt. Der Name der einzelnen Attribute wird vom Node `node-name()` bereitgestellt.

In einer Zielkomponente wird über diesen Node eine verbundene Sequenz verarbeitet und für jedes Datenelement in der Sequenz wird ein Attributwert erstellt. Der Attributname wird vom Node `node-name()` bereitgestellt. Angenommen, Sie möchten Daten aus einer XML-Quelldatei folgendermaßen in eine XML-Zieldatei transformieren:

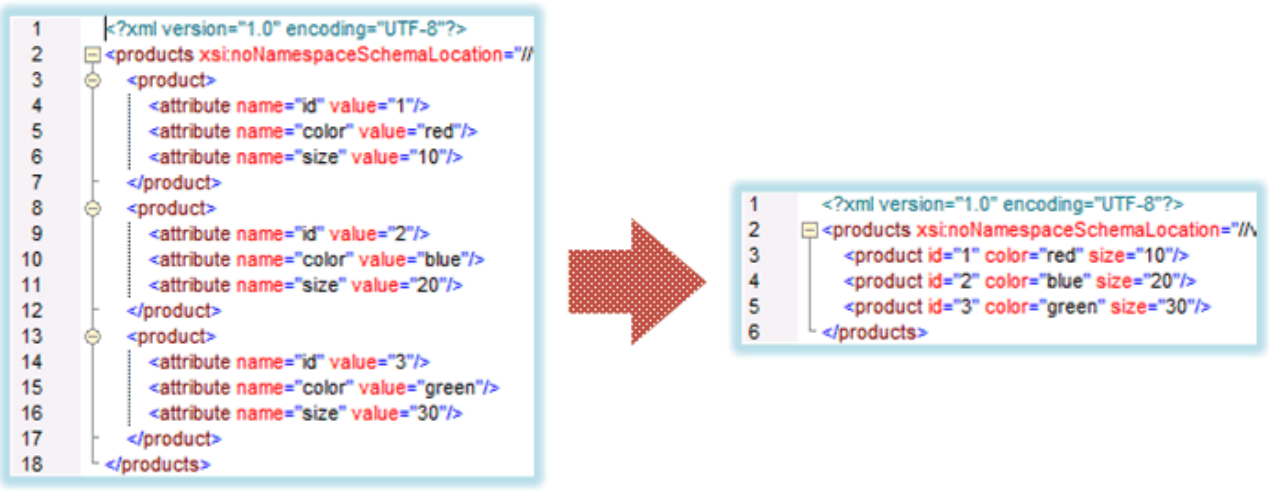


Abb. 8 Mappen von Attributwerten auf Attributnamen (Aufgabe)

Das Mapping, mit dem Sie dies erreichen, sieht folgendermaßen aus:

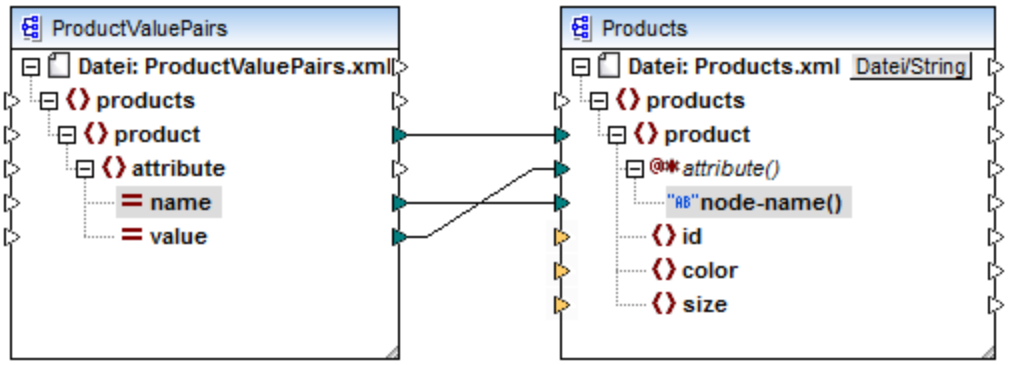


Abb. 9 Mappen von Attributwerten auf Attributnamen (in MapForce)

Anmerkung: Diese Transformation lässt sich auch ohne Aktivierung des dynamischen Zugriffs auf die Attribute eines Node bewerkstelligen. Hier wird nur gezeigt, wie `attribute()` in einer Zielkomponente funktioniert.

Wenn Sie dieses Mapping nachstellen möchten, verwenden Sie dazu dieselben XML-Komponenten wie im Mapping **ConvertProducts.mfd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Der einzige Unterschied besteht darin, dass die Zieldatei nun als Quelldatei und die Quelldatei als Zieldatei verwendet wird. Sie benötigen als Input-Daten für die Quellkomponente eine XML-Instanz, die tatsächlich Attribute-Werte enthält, wie z.B.:

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
   <product>
      <attribute name="id" value="1"/>
```

 © 2016-2022 Altova GmbH

```
      <attribute name="color" value="red"/>
      <attribute name="size" value="big"/>
   </product>
</products>
```

Beachten Sie, dass die Namespace- und die Schema-Deklaration im obigen Codefragment aus Gründen der Einfachheit weggelassen wurden.

node-name()

In einer Quellkomponente stellt `node-name()` die Namen der einzelnen Child-Elemente von `element()` bzw. die Namen der einzelnen Child-Attribute von `attribute()` bereit. Standardmäßig hat der bereitgestellte Name den Typ `xs:QName`. Um den Namen als String zu erhalten, verwenden Sie den Node `local-name()` (siehe Abb. 3) oder die Funktion QName-as-string[1163].

In einer Zielkomponente schreibt `node-name()` die Namen der einzelnen in `element()` oder `attribute()` enthaltenen Elemente bzw. Attribute.

local-name()

Dieser Node funktioniert auf dieselbe Art wie `node-name()` mit dem Unterschied, dass der Typ ist `xs:string` anstelle von `xs:QName`.

Typkonvertierungs-Node

Der Typkonvertierungs-Node in einer Quellkomponente stellt den Wert der einzelnen in `element()` enthaltenen Child-Elemente bereit. Der Name und die Struktur dieses Node hängen von dem im Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente" ausgewählten Typ ab (Abb. 2).

Um den Typ des Node zu ändern, klicken Sie auf die Schaltfläche **Auswahl ändern** ( ) und wählen Sie einen Typ aus der Liste der verfügbaren Typen (darunter auch eine Schema Wildcard (`xs:any`)) aus. Nähere Informationen dazu finden Sie unter Zugriff auf Nodes eines bestimmten Typs[1318].

In einer Zielkomponente wird über den Typkonvertierungs-Node der Wert der einzelnen in `element()` enthaltenen Child-Elemente als spezifischer Datentyp geschrieben. Der gewünschte Datentyp kann auch hier über die Schaltfläche **Auswahl ändern** ( ) ausgewählt werden.

Namensüberprüfungs-Nodes

Namensüberprüfiungs-Nodes bieten in einer Quellkomponente eine Möglichkeit, Child-Elemente aus einer Quellinstanz nach Namen zu filtern. Child-Elemente müssen eventuell nach Namen gefiltert werden, um sicherzustellen, dass das Mapping den korrekten Typ verwendet, um auf die Instanzdaten zuzugreifen (siehe Zugriff auf Nodes eines bestimmten Typs[1318]). Ein Beispiel dazu finden Sie unter Beispiel: Gruppieren und Filtern von Nodes nach Namen[1325].

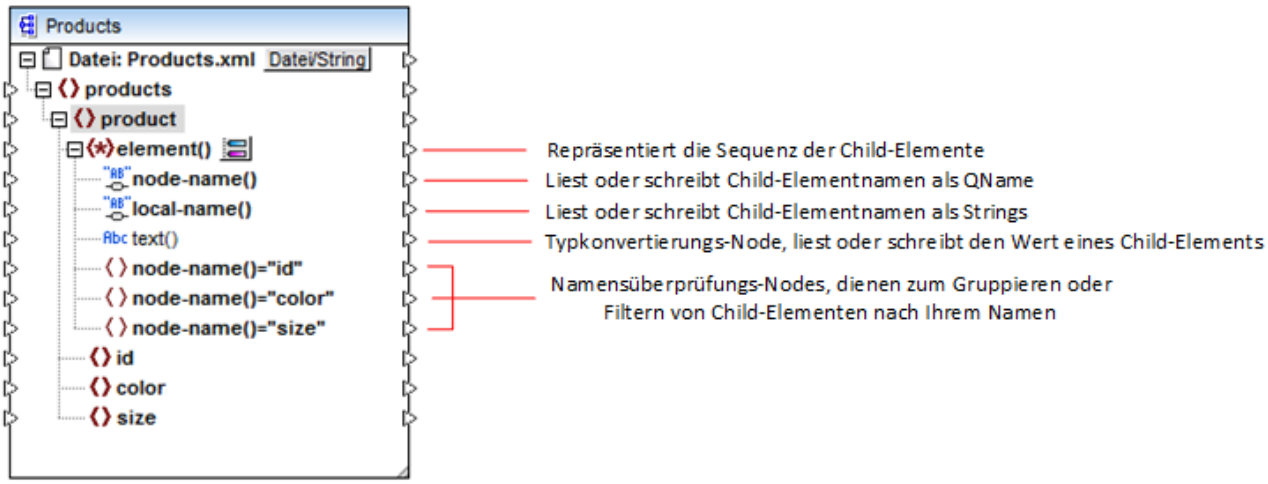
Im Allgemeinen funktionieren Namensüberprüfungs-Nodes beim Lesen und Schreiben von Werten und hierarchisch untergeordneten Strukturen fast wie normale Element-Nodes. Da sich die Mapping-Semantik aber unterscheidet, wenn der dynamische Zugriff aktiviert ist, gibt es einige Einschränkungen. So können Sie etwa die Werte von zwei Namensüberprüfungs-Nodes nicht miteinander verknüpfen.

In der Zielkomponente werden für Namensüberprüfungs-Nodes in der Ausgabe so viele Elemente erstellt, wie Datenelemente in der verbundenen Quellsequenz vorhanden sind. Ihr Name setzt den auf `node-name()` gemappten Wert außer Kraft.

Falls erforderlich, können Sie die Namensüberprüfungs-Nodes aus der Komponente ausblenden. Klicken Sie dazu auf die Schaltfläche **Auswahl ändern** ( ) neben dem `element()`-Node. Deaktivieren Sie anschließend im Dialogfeld "*Einstellungen für dynamisch benannte Child-Elemente*" (Abb. 2) das Kontrollkästchen **Name-Test-Nodes ...anzeigen"**.

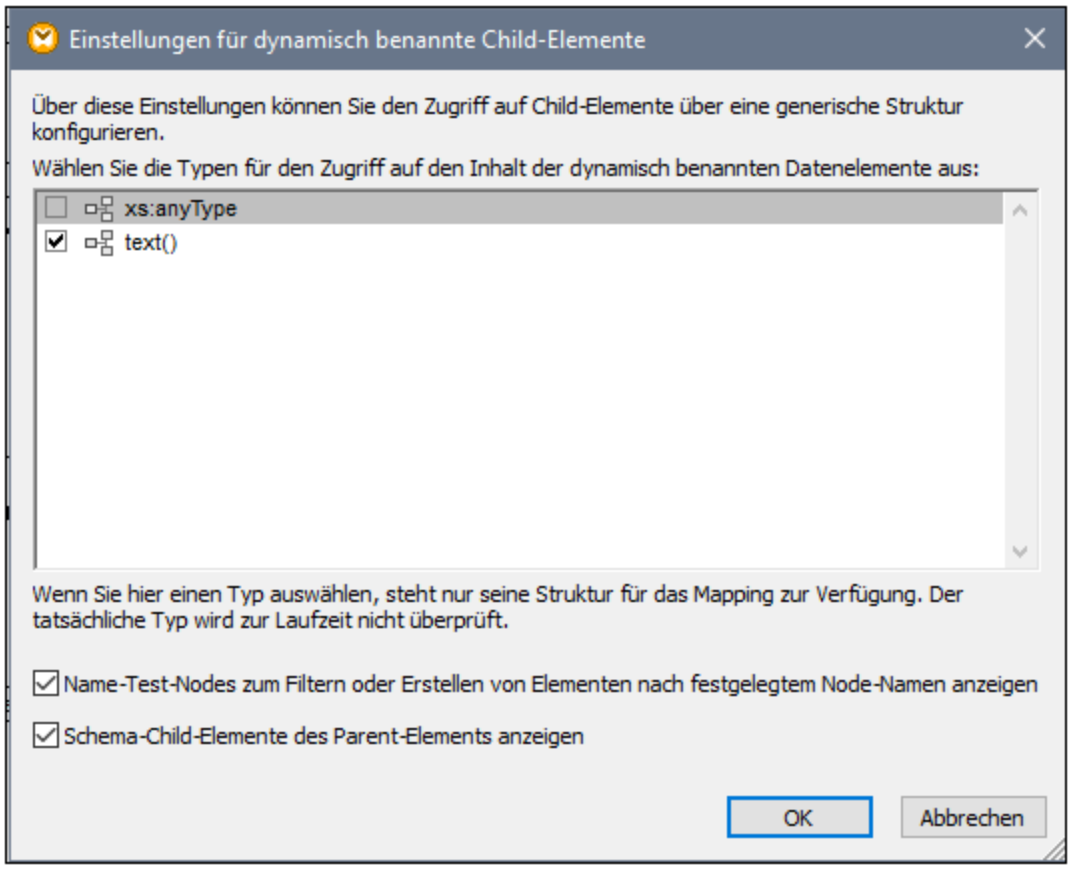
8.2.2 Zugriff auf Nodes eines bestimmten Typs

Wie in vorherigen Abschnitt, Zugriff auf Node-Namen 1310, erwähnt, erhalten Sie durch Rechtsklick auf den Node und Auswahl des Kontextmenübefehls **Child-Elemente mit generischem Namen anzeigen** Zugriff auf alle Chld-Elemente eines Node. Auf diese Art stehen die Namen der einzelnen Child-Elemente über den Node `node-name()` zur Mapping-Laufzeit Verfügung, während der Wert des Node über einen speziellen Typkonvertierungsnode (im Bild unten der Node `text()`) zur Verfügung steht.



Beachten Sie, dass der Datentyp der einzelnen Child-Elemente vor der Mapping-Laufzeit nicht bekannt ist. Jedes Child-Element kann außerdem einen anderen Datentyp haben. So kann z.B. ein `product`-Node in der XML-Instanz ein Child-Element `id` vom Typ `xs:integer` und ein Child-Element `size` vom Typ `xs:string` haben. Damit Sie Zugriff auf den Node-Inhalt eines bestimmten Typs haben, wird jedes Mal, wenn Sie den dynamischen Zugriff auf die Child-Elemente eines Node aktivieren, das unten gezeigte Dialogfeld geöffnet. Sie können dieses Dialogfeld auch später jederzeit durch Klick auf die Schaltfläche **Auswahl ändern** ( ) neben einem `element(`-Node aufrufen.

 © 2016-2022 Altova GmbH



Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente"

Es gibt verschiedene Möglichkeiten, um zur Laufzeit Zugriff auf den Inhalt der einzelnen Child-Elemente zu erhalten:

1. Aufruf des Inhalts als String. Aktivieren Sie dazu das Kontrollkästchen **text()** im oben gezeigten Dialogfeld. In diesem Fall wird beim Schließend des Dialogfelds in der Komponente ein `text()`-Node erstellt. Diese Option eignet sich, wenn der Inhalt einen simpleType hat (`xs:int`, `xs:string`, usw.). Eine Beschreibung dazu finden Sie unter Beispiel: Mappen von Elementnamen auf Attributwerte 1322. Beachten Sie, dass ein **text()**-Node nur angezeigt wird, wenn ein Child Node des aktuellen Node Text enthalten kann.
2. Aufruf des Inhalts als bestimmter complexType, der laut Schema zulässig ist. Wenn laut Schemadefinition global definierte benutzerdefinierte complexTypes für den ausgewählten Node zulässig sind, stehen diese ebenfalls im obigen Dialogfeld zur Verfügung und Sie können das Kontrollkästchen daneben aktivieren. In der Abbildung oben sind im Schema keine complexTypes definiert, daher stehen keine zur Auswahl zur Verfügung.
3. Aufruf des Inhalts als any-Typ. Dies kann in komplexen Mapping-Szenarien hilfreich sein (siehe Zugriff auf tiefer verschachtelte Strukturen") weiter unten. Aktivieren Sie dazu das Kontrollkästchen neben **xs:anyType**.

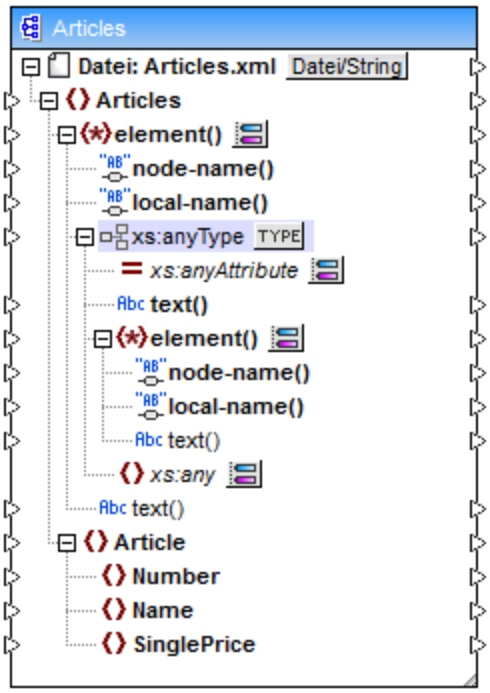
Beachten Sie, dass MapForce (über den Typkonvertierungs-Node) keinerlei Informationen darüber hat, was für einen Datentyp der Instanz-Node zur Mapping-Laufzeit tatsächlich haben wird. Ihr Mapping muss daher

den Node-Inhalt mit dem richtigen Typ aufrufen. Wenn Sie z.B. erwarten, dass der Node einer XML-Quellinstanz Child-Nodes verschiedener complexType-Arten haben kann, gehen Sie folgendermaßen vor:
a) Setzen Sie den Typkonvertierungs-Node auf den complexTyp, mit dem er übereinstimmen muss (siehe Punkt 2 in der Liste oben).
b) Fügen Sie einen Filter hinzu, damit nur der gewünschte complexType aus der Instanz ausgelesen wird. Ein Beispiel für diese Methode finden Sie unter Beispiel: Gruppieren und Filtern von Nodes nach Namen 1325.

Zugriff auf tiefer verschachtelte Strukturen

Es ist möglich, Zugriff auf Nodes, die sich auf einer Ebene noch unterhalb der unmittelbaren Child-Nodes eines Node befinden, zu erhalten. Dies eignet sich für komplexe Mapping-Szenarien. Bei einfachen Mappings wie z.B. im Beispiel: Mappen von Elementnamen auf Attributwerte 1322 ist diese Methode nicht notwendig, da im Mapping nur die unmittelbaren Child-Nodes eines XML-Node aufgerufen werden. Wenn Sie jedoch dynamischen Zugriff auf tiefer gelegene Strukturen wie z.B. "Enkel" und "Urenkel" usw. benötigen, so gehen Sie vor, wie unten gezeigt.

1. Erstellen Sie ein neues Mapping.
2. Klicken Sie im Menü "Einfügen" auf **XML-Schema/Datei einfügen** und navigieren Sie zur XML-Instanzdatei (in diesem Beispiel die Datei **Articles.xml** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**).
3. Klicken Sie mit der rechten Maustaste auf den Node `Articles` und wählen Sie den Kontextmenübefehl **Child-Elemente mit dynamischem Namen anzeigen**.
4. Wählen Sie im Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente" **xs:anyType** aus.
5. Klicken Sie mit der rechten Maustaste auf den Node **xs:anyType** und wählen Sie erneut den Kontextmenübefehl **Child-Elemente mit dynamischem Namen anzeigen**.
6. Wählen Sie im Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente" **text()** aus.



 © 2016-2022 Altova GmbH

Beachten Sie, dass die Komponente oben zwei `element()` Nodes enthält. Der zweite `element()`-Node ermöglicht dynamischen Zugriff auf die Enkel des Node `<Articles>` in der Instanz **Articles.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<Articles xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Articles.xsd">
   <Article>
      <Number>1</Number>
      <Name>T-Shirt</Name>
      <SinglePrice>25</SinglePrice>
   </Article>
   <Article>
      <Number>2</Number>
      <Name>Socks</Name>
      <SinglePrice>2.30</SinglePrice>
   </Article>
   <Article>
      <Number>3</Number>
      <Name>Pants</Name>
      <SinglePrice>34</SinglePrice>
   </Article>
   <Article>
      <Number>4</Number>
      <Name>Jacket</Name>
      <SinglePrice>57.50</SinglePrice>
   </Article>
</Articles>
```

Articles.xml

Um z.B. die Enkelelementnamen (`Number`, `Name`, `SinglePrice`) abzurufen, würden Sie eine Verbindung vom Node `local-name()` unterhalb des zweiten `element()` Node zu einem Zieldatenelement ziehen. Um die Enkelelementwerte (`1`, `T-Shirt`, `25`) abzurufen, würden Sie eine Verbindung vom Node `text()` ziehen.

Zwar gilt dies nicht für dieses Beispiel, doch können Sie in realen Einsatzszenarien auch für jeden nachfolgenden `xs:anyType` Node den dynamischen Zugriff auf Node-Namen aktivieren, um noch tiefer gelegene Ebenen zu erreichen.

Beachten Sie bitte Folgendes:

- Über die Schaltfläche TYPE können Sie jeden abgeleiteten Typ aus dem aktuellen Schema auswählen und in einem separaten Node anzeigen. Dies ist nur dann nützlich, wenn Sie von oder auf abgeleitete Schematypen mappen müssen (siehe Abgeleitete XML-Schema-Typen 130).
- Über die neben einem `element ()` Node gelegene Schaltfläche **Auswahl ändern** ( ) rufen Sie das in diesem Kapitel beschriebene Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente" auf.
- Über die Schaltfläche **Auswahl ändern** ( ) neben dem `xs:anyAttribute`-Attribut können Sie jedes beliebige global im Schema definierte Attribut auswählen. Auf die gleiche Art können Sie über die Schaltfläche **Auswahl ändern** ( ) neben dem `xs:any`-Element jedes beliebige global im Schema definierte Element auswählen. Dies funktioniert auf dieselbe Art und Weise wie das Mappen von oder auf Schema-Wildcards (siehe auch Wildcards - xs:any / xs:anyAttribute 138). Vergewissern Sie sich

bei Verwendung dieser Option, dass das ausgewählte Attribut oder Element gemäß dem Schema auf dieser Ebene auch wirklich zulässig ist.

8.2.3 Beispiel: Mappen von Elementnamen auf Attributwerte

In diesem Beispiel wird gezeigt, wie Sie Elementnamen aus einem XML-Dokument auf Attributwerte in einem XML-Zieldokument mappen. Das Mapping zu diesem Beispiel finden Sie im folgenden Ordner: **<Dokumente>\Altova\MapForce2023\MapForceExamples\ConvertProducts.mfd**.

Betrachten wir zur Erläuterung des Beispiels folgendes Szenario: Angenommen Sie haben eine XML-Datei, die eine Liste von Produkten enthält. Jedes Produkt hat das folgende Format:

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

Unser Ziel ist es, Informationen zu jedem Produkt in ein Namen-Wert-Paar zu konvertieren, z.B.:

```
<product>
   <attribute name="id" value="1" />
   <attribute name="color" value="red" />
   <attribute name="size" value="10" />
</product>
```

Um ein Mapping wie das oben beschriebene mit möglichst wenig Aufwand durchzuführen, wird in diesem Beispiel eine MapForce-Funktion namens "dynamischer Zugriff auf Node-Namen" verwendet. "Dynamisch" bedeutet, dass die Node-Namen (und nicht nur die Werte) während der Ausführung des Mappings gelesen und als Werte geschrieben werden. Das Mapping dazu wird in wenigen Schritten folgendermaßen erstellt:

Schritt 1: Hinzufügen der XML-Quellkomponente zum Mapping

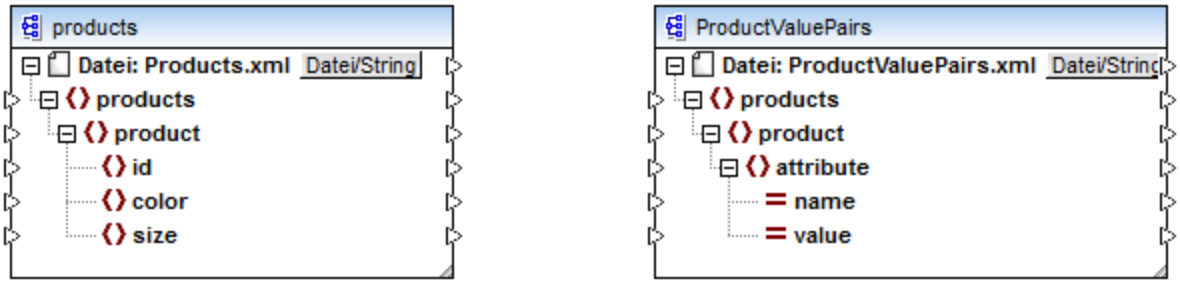
- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur folgenden Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\Products.xml**. Diese XML-Datei verweist auf das Schema **Products.xsd** im selben Ordner.

Schritt 2: Fügen Sie die XML-Zielkomponente zum Mapping hinzu

- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur folgenden Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\ProductValuePairs.xsd**. Wenn Sie aufgefordert werden, eine Instanzdatei anzugeben, klicken Sie auf **Überspringen**. Wenn Sie aufgefordert werden, ein Root-Element auszuwählen, wählen Sie `products` als Root-Element.

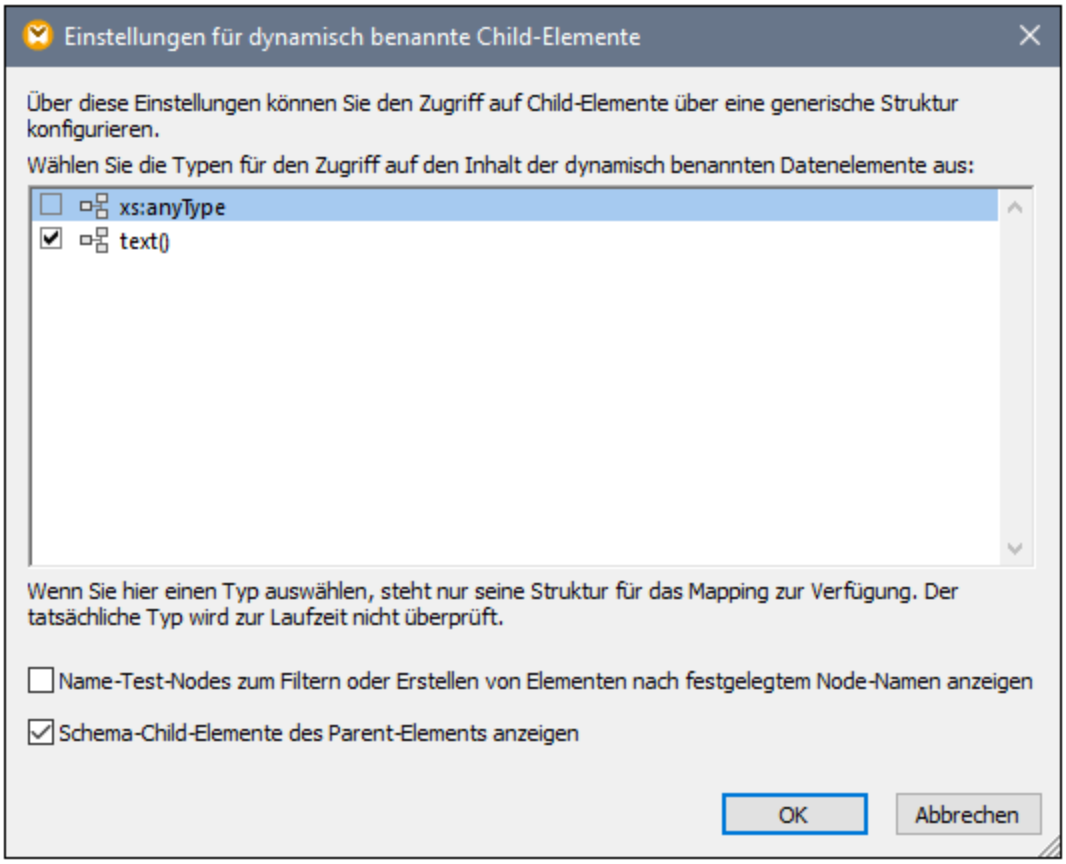
Das Mapping sollte zu diesem Zeitpunkt folgendermaßen aussehen:

© 2016-2022 Altova GmbH

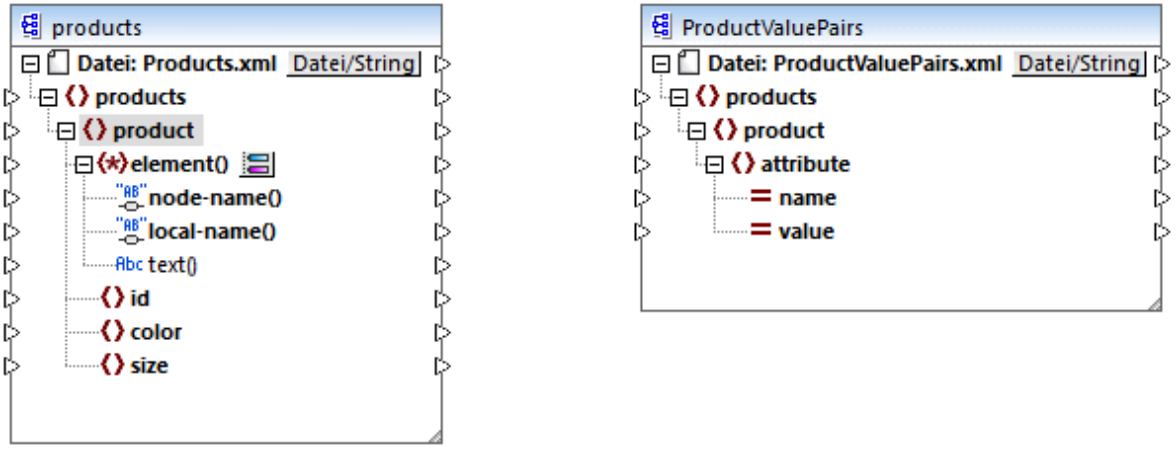


Schritt 3: Aktivieren Sie den dynamischen Zugriff auf Child-Nodes

1. Klicken Sie mit der rechten Maustaste in der Quellkomponente auf den Node `product` und wählen Sie im Kontextmenü den Befehl **Child-Elemente mit dynamischem Namen anzeigen**.
2. Wählen Sie im Dialogfeld, das daraufhin geöffnet wird, **text()** als Typ aus. Belassen Sie andere Optionen unverändert.

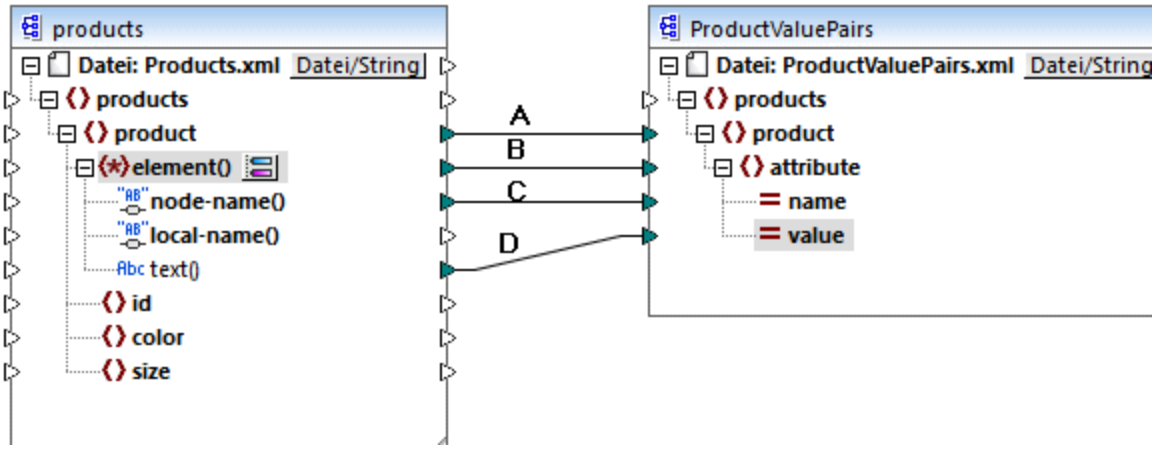


Beachten Sie, dass nun ein `text()`-Node zur Quellkomponente hinzugefügt wurde. Dieser Node stellt den Inhalt der einzelnen Child-Elemente für das Mapping bereit (in diesem Fall den Wert von "id", "color" und "size").



Schritt 4: Ziehen Sie die Mapping-Verbindungen

Ziehen Sie schließlich die Mapping-Verbindungen A, B, C, D, wie unten gezeigt. Doppelklicken Sie optional auf die einzelnen Verbindungen und geben Sie von oben nach unten die Buchstaben "A", "B", "C" und "D" in die einzelnen Beschreibungsfelder ein.



ConvertProducts.mfd

Im oben gezeigten Mapping wird mit Verbindung A für jedes Produkt in der Quellkomponente ein Produkt in der Zielkomponente erstellt. Bis jetzt handelt es sich um eine Standard-Mapping-Verbindung, die die Node-Namen nicht berücksichtigt. In Verbindung B wird jedoch für jedes Child-Element von `product` in der Zielkomponente ein neues Element namens `attribute` erstellt.

Die Verbindung B ist eine entscheidende Verbindung im Mapping, da damit eine Sequenz von Child-Elementen von `product` von der Quellkomponente in die Zielkomponente übertragen wird. Dabei werden nicht die tatsächlichen *Namen* oder *Werte* übertragen. Diese Verbindung ist folgendermaßen zu verstehen: Wenn **element()** in der Quellkomponente N Child-Elemente hat, so werden in der Zielkomponente N Instanzen dieses Datenelements erstellt. In diesem konkreten Beispiel hat `product` in der

 © 2016-2022 Altova GmbH

Quellkomponente drei Child-Elemente (`id`, `color` und `size`). Das bedeutet, dass jedes `product`-Element in der Zieldatei drei Child-Elemente mit dem Namen `attribute` hat.

In diesem Beispiel wird dies zwar nicht gezeigt, doch wird dieselbe Regel auch auf das Mappen von Child-Elemente von **attribute()** angewendet. Wenn das Datenelement **attribute()** in der Quellkomponente N Child-Attribute hat, so werden in der Zielkomponente N Instanzen dieses Datenelements erstellt.

Als Nächstes werden in Verbindung C die tatsächlichen Namen der einzelnen Child-Elemente von `product` in die Zielkomponente kopiert (nämlich "id", "color" und "size").

Schlussendlich werden in Verbindung D die Werte der einzelnen Child-Elemente von product als String-Typ in die Zielkomponente kopiert.

Um eine Vorschau auf die Mapping-Ausgabe zu sehen, klicken Sie auf das Register **Ausgabe** und werfen Sie einen Blick auf die generierte XML-Datei. Wie erwartet und beabsichtigt, enthält die Ausgabe mehrere Produkte, deren Daten als Namen-Wert-Paare gespeichert sind.

```
<?xml version="1.0" encoding="UTF-8"?>
<products xsi:noNamespaceSchemaLocation="ProductValuePairs.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <product>
      <attribute name="id" value="1"/>
      <attribute name="color" value="red"/>
      <attribute name="size" value="10"/>
   </product>
   <product>
      <attribute name="id" value="2"/>
      <attribute name="color" value="blue"/>
      <attribute name="size" value="20"/>
   </product>
   <product>
      <attribute name="id" value="3"/>
      <attribute name="color" value="green"/>
      <attribute name="size" value="30"/>
   </product>
</products>
```

Generierte Mapping-Ausgabe

8.2.4 Beispiel: Gruppieren und Filtern von Nodes nach Namen

In diesem Beispiel wird gezeigt, wie Sie ein Mapping erstellen, das Schlüssel-Wert-Paare aus einer Liste von XML-Eigenschaften (oder XML plist) ausliest und diese in eine CSV-Datei schreibt. (Die XML-Eigenschaftsliste stellte eine Methode zum Speichern von macOS- und iOS-Objektinformationen im XML-Format dar, siehe https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/PropertyLists/UnderstandXMLPlist/UnderstandXMLPlist.html.) Die Mapping-Beispieldatei hierzu finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\ReadPropertyList.mfd**.

Das unten gezeigte Codefragment bildet die XML-Quelldatei.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist SYSTEM "https://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
   <dict>
      <key>First Name</key>
      <string>William</string>
      <key>Last Name</key>
      <string>Shakespeare</string>
      <key>Birthdate</key>
      <integer>1564</integer>
      <key>Profession</key>
      <string>Playwright</string>
      <key>Lines</key>
      <array>
         <string>It is a tale told by an idiot,</string>
         <string>Full of sound and fury, signifying nothing.</string>
      </array>
   </dict>
</plist>
```

Ziel des Mappings ist es, anhand bestimmter Schlüssel-Wert-Paare aus dem Node `<dict>` (in der Eigenschaftslistendatei) eine neue Zeile in der CSV-Datei zu erstellen. Dabei soll das Mapping nur **<key>** - **<string>**-Paare filtern. Andere Schlüssel-Wert-Paare (wie z.B. **<key>** - **<integer>**) sollen ignoriert werden. In der CSV-Datei soll der Name der Eigenschaft, der durch ein Komma vom Wert der Eigenschaft getrennt ist, in einer Zeile gespeichert werden. Die Ausgabe sollte also folgendermaßen aussehen:

```
First Name,William
Last Name,Shakespeare
Profession,Playwright
```

Zu diesem Zweck wird ein dynamischer Zugriff auf alle Child-Nodes des Node `dict` im Mapping verwendet. Des Weiteren wird die Funktion group-starting-with[1091] verwendet, um die aus der XML-Datei abgerufenen Schlüssel-Wert-Paare zu gruppieren. Schließlich wird ein Filter verwendet, um nur die Nodes zu filtern, deren Node-Name "string" ist.

In den folgenden Schritten wird gezeigt, wie das Mapping erstellt wird.

Schritt 1: Hinzufügen der XML-Quellkomponente zum Mapping

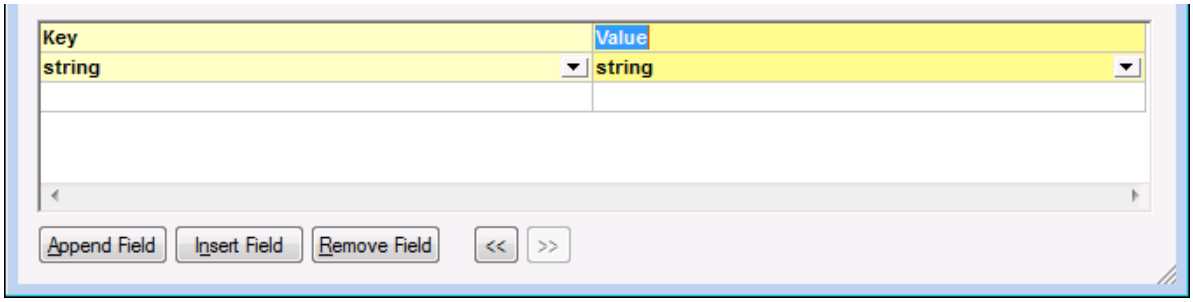
1. Setzen Sie die Mapping-Transformationssprache auf BUILT-IN[29].
2. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur folgenden Datei: **<Dokumente>\Altova\MapForce2023\MapForceExamples\plist.xml**. Diese XML-Datei verweist auf das Schema **plist.dtd** im selben Ordner.

Schritt 2: Hinzufügen der CSV-Zielkomponente zum Mapping

1. Klicken Sie im Menü **Einfügen** auf **Textdatei**. Wenn Sie dazu aufgefordert werden, wählen Sie die Option **Einfache Verarbeitung für Standard-CSV verwenden...** .
2. Fügen Sie ein CSV-Feld zur Komponente hinzu, indem Sie auf **Feld anhängen** klicken.

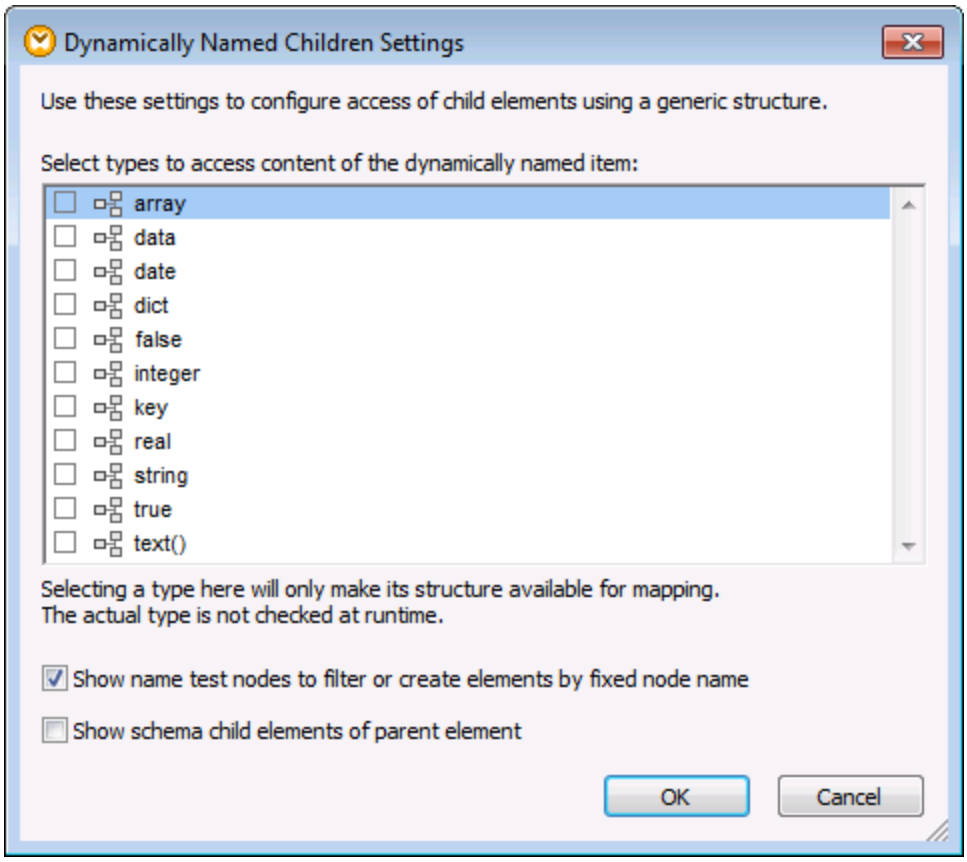
© 2016-2022 Altova GmbH

3. Doppelklicken Sie auf die Namen der einzelnen Felder und geben Sie als Namen des ersten Felds "Schüssel" und als Namen des zweiten Felds "Wert" ein. Im Feld "Schlüssel" wird der Name der Eigenschaft und im Feld "Wert" der Wert der Eigenschaft gespeichert. Nähere Informationen zu CSV-Komponenten finden Sie unter CSV- und Textdateien[585].



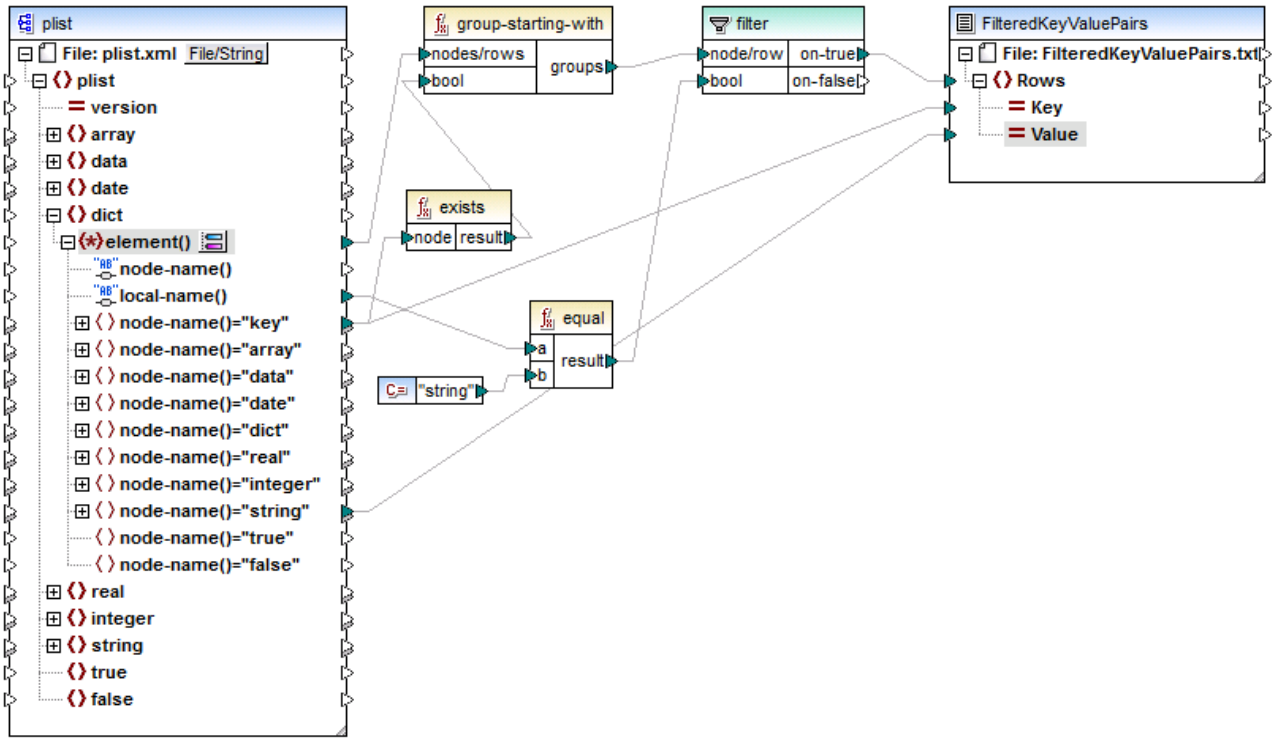
Schritt 3: Hinzufügen des Filters und der Funktionen

1. Ziehen Sie die Funktionen **equal**[1057], **exists**[1079] und **group-starting-with**[1091] aus dem Fenster "Bibliotheken" in das Mapping. Nähere Informationen zu Funktionen finden Sie unter Funktionen[942].
2. Um den Filter hinzuzufügen, klicken Sie auf das Menü **Einfügen** und anschließend auf **Filter: Nodes/Zeilen**. Allgemeine Informationen zu Filtern finden Sie unter Filter und Bedingungen[914].
3. Klicken Sie im Menü **Einfügen** auf **Konstante** und geben Sie anschließend den Text "string" ein.
4. Klicken sie mit der rechten Maustaste in der Quellkomponente auf den Node `dict` und wählen Sie im Kontextmenü den Befehl **Child-Elemente mit dynamischem Namen anzeigen.** Stellen Sie sicher, dass im Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente" das Kontrollkästchen **Name-Test-Nodes zum Filtern oder Erstellen von Elementen nach festgelegtem Node-Namen anzeigen** aktivert ist.



5. Ziehen Sie die Verbindungen wie unten gezeigt.

© 2016-2022 Altova GmbH



ReadPropertyList.mfd

Erläuterung zum Mapping

Das Datenelement `element()` der Quellkomponente liefert alle Children des Node `dict` als Sequenz an die Funktion **`group-starting-with`**. Die Funktion **`group-starting-with`** erstellt jedes Mal, wenn sie auf einen Node mit dem Namen `key` stößt, eine neue Gruppe. Die Funktion **`exists`** überprüft diese Bedingung und gibt das Ergebnis als Booleschen true/false-Wert an die Gruppierungsfunktion zurück.

Der Filter überprüft mit Hilfe der Funktion `equal` für jede Gruppe, ob der Name des aktuellen Node gleich "string" ist. Der Name selbst wird aus dem Node `local-name()`, der den Namen des Node als String bereitstellt, gelesen.

Die Verbindungen zur Zielkomponente haben die folgende Aufgabe:

- Nur wenn die Filterbedingung "true" ist, wird in der CSV-Zieldatei eine neue Zeile erstellt.
- `Key` (Eigenschaftsname) stammt aus dem Wert des `key`-Elements in der Quelldatei.
- `Value` (Eigenschaftswert) stammt aus dem Namensüberprüfungs-Node `string`.

8.3 Mapping-Regeln und Strategien

MapForce mappt Daten im Allgemeinen auf intuitive Art, doch gibt es einige Situationen, in denen die erzeugte Ausgabe zu viele oder zu wenige Datenelemente enthält. In diesem Kapitel wird beschrieben, wie Sie Situationen vermeiden, in denen infolge falscher Verbindungen oder des falschen Kontexts unerwünschte Ergebnisse erzeugt werden.

Mapping-Regeln

Damit ein Mapping gültig ist, muss es mindestens eine Quell- und Zielkomponente enthalten. Eine Quellkomponente ist eine Komponente, mit der Daten normalerweise aus einer Daten oder Datenbank ausgelesen werden. Eine Zielkomponente ist eine Komponente, mit der Daten normalerweise in eine Datei oder Datenbank geschrieben werden. Wenn Sie versuchen, ein Mapping zu speichern, in dem diese Voraussetzungen nicht erfüllt werden, wird im Fenster "Meldungen" ein Fehler angezeigt: "Für ein Mapping sind mindestens zwei miteinander verbundene Strukturen erforderlich, z.B. eine Schema- oder eine Datenbank-Struktur."

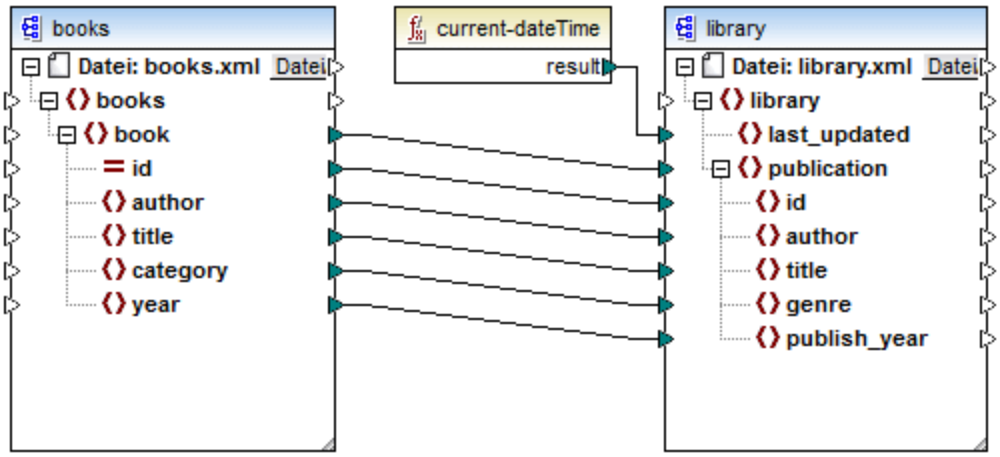
Um ein Datenmapping zu erstellen, ziehen Sie Mapping-Verbindungen zwischen Datenelementen in der Quellkomponente und Datenelementen in der Zielkomponente.

Alle erstellten Mapping-Verbindungen bilden gemeinsam einen Mapping-Algorithmus. Zur Laufzeit des Mappings wertet MapForce den Algorithmus aus und verarbeitet Daten anhand dessen. Ob und wie effizient der Mapping-Algorithmus funktioniert, hängt in erster Linie von den Verbindungen ab. Einige Einstellungen können außerdem auf Mapping- 116, Komponenten- 82 oder sogar Verbindungsebene 93 angepasst werden, doch im Prinzip wird mit den Mapping-*Verbindungen* festgelegt, wie Ihre Daten verarbeitet werden.

Beachten Sie beim Ziehen von Verbindung Folgendes:

1. Wenn Sie eine Verbindung *von* einem Quelldatenelement aus ziehen, liest das Mapping die mit diesem Datenelement verknüpften Daten aus der Quelldatei oder -datenbank aus. Die Daten können null, eine oder mehrere Instanzen haben (d.h. es kann sich dabei, wie weiter unten beschrieben, um eine Sequenz handeln). Wenn z.B. Daten aus einer XML-Datei, die Bücher enthält, ausgelesen werden, kann die XML-Quelldatei null, ein oder mehrere **book**-Elemente enthalten. Beachten Sie, dass das Datenelement **book** im unten gezeigten Mapping in der Mapping-Komponente nur einmal angezeigt wird, obwohl die Quelldatei (Instanzdatei) mehrere **book**-Elemente oder keines enthalten kann.

© 2016-2022 Altova GmbH



2. Wenn Sie eine Verbindung zu einem Zieldatenelement ziehen, generiert das Mapping Instanzdaten dieser Art. Wenn das Quelldatenelement einfachen Inhalt (z.B. Strings oder Ganzzahlen) enthält und im Zieldatenelement einfacher Inhalt zulässig ist, wird der Inhalt in das Zieldatenelement kopiert und der Datentyp, falls nötig, konvertiert. Es können je nach Quelldaten null, ein oder mehrere Werte generiert werden, siehe nächster Punkt.
3. Für jedes (Instanz)datenelement in der Quellkomponente wird ein (Instanz)datenelement in der Zielkomponente erstellt. **So lautet die allgemeine Mapping-Regel in MapForce.** Wenn also die XML-Quelldatei, wie in der Abbildung oben, drei **book**-Elemente enthält,so werden in der Zielkomponente drei **publication**-Elemente erstellt. Beachten Sie, dass es auch einige Sonderfälle gibt, siehe Sequenzen 1331.
4. Für jede Verbindung wird ein *aktueller Mapping-Kontext* erstellt. Der Kontext legt fest, welche Daten *aktuell gerade für den aktuellen Ziel-Node* zur Verfügung stehen, d.h. mit dem Kontext wird festgelegt, welche Quelldatenelemente tatsächlich aus der Quell- in die Zielkomponente kopiert werden. Indem Sie eine Verbindung ziehen oder nicht erstellen, ändern Sie eventuell unabsichtlich den aktuellen Kontext, was sich auf die Ausgabe des Mappings auswirkt. So könnte z.B. eine Datenbank oder ein Webservice unnötigerweise mehrmals während derselben Mapping-Ausführung aufgerufen werden. Eine ausführlichere Beschreibung dazu finden Sie weiter unten unter Der Mapping-Kontext 1333.

8.3.1 Sequenzen

Wie bereits erwähnt, lautet die allgemeine Mapping-Regel: "Erstelle für jedes Datenelement in der Quellkomponente eines in der Zielkomponente". Mit "Datenelement" ist hier eines der folgenden gemeint:

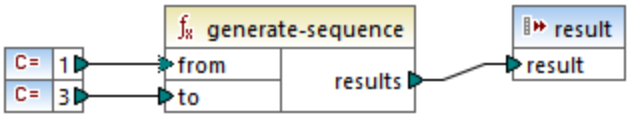
- ein einzelner Instanz-Node der Input-Datei oder Datenbank
- eine Sequenz von null bis zu mehreren Instanz-Nodes der Input-Datei oder Datenbank

Wenn bei der Mapping-Ausführung bei einer Sequenz ein Ziel-Datenelement erreicht wird, wird eine Schleife erstellt, mit der so viele Ziel-Nodes generiert werden, wie Quell-Nodes vorhanden sind. Für diese Regel gibt es jedoch einige Ausnahmen:

- Wenn es sich beim Zieldatenelement um ein XML-Root-Element handelt, wird dieses nur genau einmal erstellt. Wenn Sie damit eine Sequenz verbinden, ist das Ergebnis dem Schema gemäß möglicherweise nicht gültig. Wenn auch Attribute des Root-Elements verbunden werden, schlägt die

XML-Serialisierung zur Laufzeit fehl. Vermeiden Sie es daher, eine Sequenz mit dem XML-Root-Element zu verbinden.

- Wenn im Zieldatenelement nur ein einziger Wert zulässig ist, wird es nur einmal erstellt. Beispiele für Datenelemente, in denen nur ein einziger Wert zulässig ist: XML-Attribute, Datenbankfelder, einfache Output-Komponenten. So wird etwa mit dem unten gezeigten Mapping mit Hilfe der **`generate-sequence-`**Funktion eine Sequenz von drei Ganzzahlen (1, 2, 3) generiert. Trotzdem enthält die Ausgabe nur eine Ganzzahl, da es sich bei der Zielkomponente um eine einfache Output-Komponente handelt, in der nur ein einziger Wert zulässig ist. Die anderen beiden Werte werden ignoriert.



- Wenn im Quellschema festgelegt ist, dass ein bestimmtes Datenelement nur einmal vorkommen darf, die Instanzdatei aber viele davon enthält, extrahiert MapForce unter Umständen das erste Datenelement aus der Quelldatei (das laut Schema vorhanden sein muss) und erstellt in der Zielkomponente nur ein Datenelement. Um dieses Verhalten zu verhindern, deaktivieren Sie in den Komponenteneinstellungen das Kontrollkästchen **Input-Verarbeitungsoptimierungen auf Basis von min/maxOccurs aktivieren,** siehe auch XML-Komponenteneinstellungen 126.

Wenn die Sequenz leer ist, wird auf Seite der Zielkomponente nichts generiert. Wenn die Zielkomponente z.B. ein XML-Dokument ist und die Quellsequenz leer ist, werden in der Zielkomponente gar keine XML-Elemente erstellt.

Funktionen verhalten sich ähnlich: Wenn Sie als Input eine Sequenz erhalten, werden sie so oft aufgerufen (und erzeugen so viele Ergebnisse) wie Datenelemente in der Sequenz vorhanden sind.

> Wenn eine Funktion als Input eine leere Sequenz erhält, gibt sie auch ein leeres Ergebnis zurück und erzeugt somit gar keine Ausgabe.

Es gibt jedoch einige Funktionskategorien, die aufgrund ihres Designs auch dann einen Wert zurückgeben, wenn Sie als Input eine leere Sequenz erhalten:

- **`exists`**, **`not-exists`**, **`substitute-missing`**
- **`is-not-null`**, **`is-null`**, **`substitute-null`** (Diese drei Funktionen sind Aliasnamen für die erstgenannten drei Funktionen)
- aggregate-Funktionen (**`sum`**, **`count`**, usw.)
- benutzerdefinierte Funktionen, die Sequenzen unterstützen und regulär sind (also nicht-inline-Funktionen)

Wenn Sie einen leeren Wert ersetzen müssen, fügen Sie die Funktion **`substitute-missing`** zum Mapping hinzu und ersetzen sie den leeren Wert durch den Ersetzungswert Ihrer Wahl. Dasselbe Ergebnis erzielen Sie auch mit Hilfe von Standardwerten und Node-Funktionen 951.

Funktionen können mehrere Inputs haben. Wenn eine Sequenz mit jedem einzelnen Input verbunden wird, wird dadurch ein kartesisches Produkt aller Inputs erzeugt, was normalerweise nicht das gewünschte Ergebnis ist. Um dies zu vermeiden, verbinden Sie nur eine Sequenz mit einer Funktion mit mehreren Parametern; alle anderen Parameter müssen von übergeordneten Elementen oder anderen Komponenten aus mit Einzeldatenelementen verbunden werden.

 © 2016-2022 Altova GmbH

8.3.2 Der Mapping-Kontext

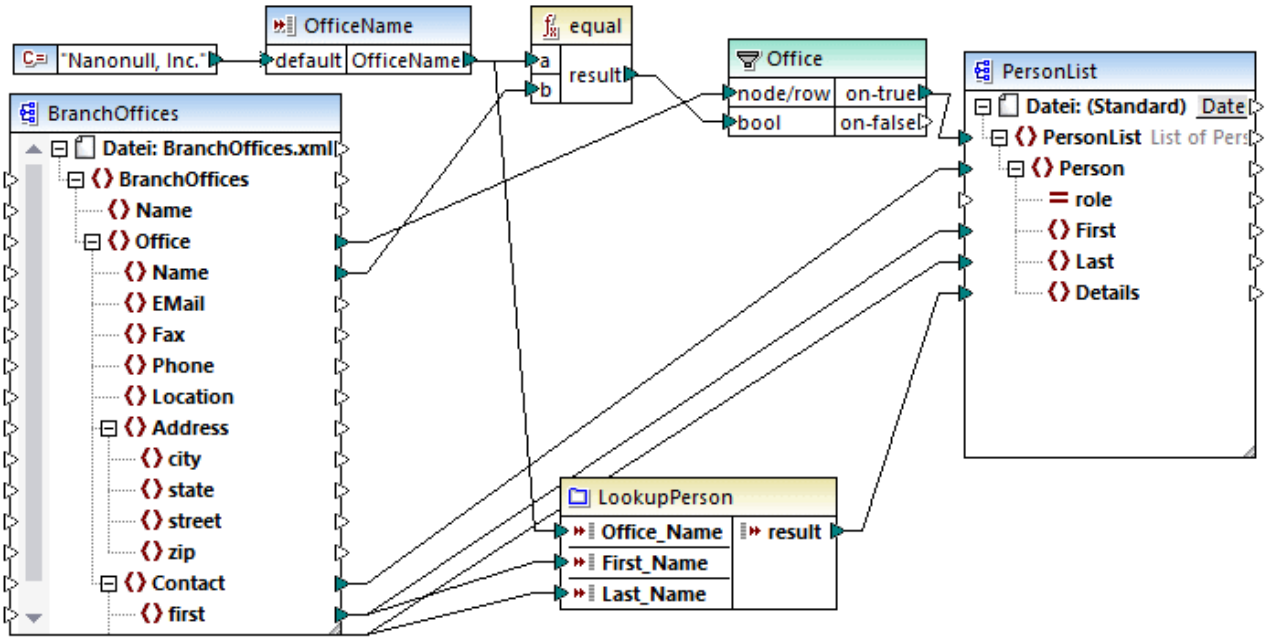
Mapping-Komponenten sind hierarchische Strukturen, die viele Ebenen tief verschachtelt sein können. Zudem kann ein Mapping mehrere Quell- und Zielkomponenten sowie Zwischenkomponente wie Funktionen, Filter, Wertezuordnungen, usw. enthalten. Dies verkompliziert den Mapping-Algorithmus, vor allem, wenn mehrere nicht miteinander in Zusammenhang stehende Komponenten miteinander verbunden sind. Damit das Mapping Schritt für Schritt abschnittsweise ausgeführt werden kann, muss für jede Verbindung ein aktueller Kontext festgelegt sein.

Man könnte auch sagen, dass für die Dauer der Mapping-Ausführung mehrere "aktuelle Kontexte" festgelegt werden, da sich der aktuelle Kontext bei jeder verarbeiteten Verbindung ändert.

MapForce ermittelt den aktuellen Kontext immer vom *Root-Zieldatenelement (Node)* aus. Die Mapping-Ausführung beginnt immer hier. Die Verbindung zum Root-Zieldatenelement wird zu allen direkt oder indirekt - auch über Funktionen oder andere Zwischenkomponenten - damit verbundenen Quelldatenelementen, zurückverfolgt. Alle Quelldatenelemente und durch Funktionen erzeugten Ergebnisse werden zum aktuellen Kontext hinzugefügt.

Nachdem der Ziel-Node verarbeitet wurde arbeitet sich MapForce von oben nach unten durch die Hierarchie durch. Dabei werden alle *gemappten Datenelemente* der Zielkomponente von oben nach unten verarbeitet. Für jedes neue Datenelement wird ein neuer Kontext ermittelt, der anfangs alle Datenelemente des Parent-Kontexts enthält. Folglich sind alle gemappten gleichrangigen Datenelemente einer Zielkomponente voneinander unabhängig, haben aber Zugriff auf alle Quelldatenelemente ihrer übergeordneten Datenelemente.

Ein praktisches Beispiel dazu sehen Sie im Beispielmapping **PersonListByBranchOffice.mfd** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**.



Im oben gezeigten Mapping handelt es sich sowohl bei der Quell- als auch der Zielkomponente um XML-Dateien. Die XML-Quelldatei enthält zwei **Office**-Elemente.

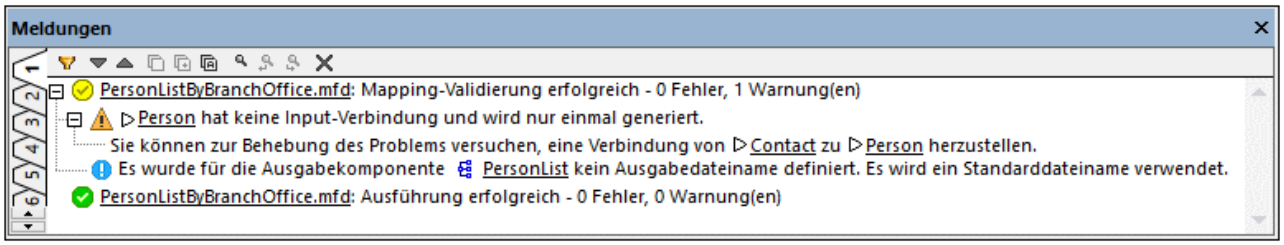
Wie bereits zuvor erwähnt, beginnt die Mapping-Ausführung immer am Ziel-Root-Node (in diesem Beispiel **PersonList)**. Wenn Sie die Verbindung (über den Filter und die Funktion) zu einem Quelldatenelement zurückverfolgen, sehen Sie, dass das Quelldatenelement **Office** ist. (Der andere Verbindungspfad führt zu einem Input-Parameter, dessen Zweck weiter unten erläutert wird).

Gäbe es eine einfache direkte Verbindung zwischen **Office** und **PersonList**, würden gemäß der allgemeinen Mapping-Regel genauso viele **PersonList**-Instanzdatenelemente erstellt, wie in der Quelldatei **Office**-Datenelemente vorhanden sind. Dies ist hier jedoch nicht der Fall, weil sich dazwischen ein Filter befindet. Der Filter liefert an die Zielkomponente nur einen Datensatz, der die mit dem Input **bool** des Filters verbundene Boolesche Bedingung erfüllt. Die Funktion `equal` gibt **true** zurück, wenn der Name (Name) des Büros "Nanonull, Inc." ist. Diese Bedingung trifft nur einmal zu, da die XML-Quelldatei nur einen solchen Office-Namen enthält.

Mit der Verbindung zwischen **Office** und **PersonList** wird folglich nur ein einziges Büro (Office) als Kontext für das gesamte Zieldokument definiert. Das bedeutet, alle Nachfahren des Datenelements **PersonList** haben Zugriff auf die Daten des Büros "Nanonull, Inc." und es gibt im aktuellen Kontext keine weiteren Büros.

Die nächste Verbindung ist die zwischen **Contact** und **Person**. Gemäß der allgemeinen Mapping-Regel wird dadurch für jedes **Contact**.-Quelldatenelement ein **Person**-Zieldatenelement erstellt. Bei jeder Iteration wird mit dieser Verbindung ein neuer aktueller Kontext festgelegt. Daher liefern die Child-Verbindungen (**first** zu **First**, **last** zu **Last**) im Kontext jeder einzelnen **Person** Daten aus der Quellkomponente an das Zieldatenelement.

Würden Sie die Verbindung zwischen **Contact** und **Person** weglassen, würde nur ein Person-Datenelement mit mehreren **First-**, **Last-** und **Details**-Nodes erstellt. In solchen Fällen gibt MapForce im Fenster "Meldungen" eine Warnmeldung sowie einen Lösungsvorschlag aus, z.B.:



Schließlich enthält das Mapping noch eine benutzerdefinierte Funktion `LookupPerson`. Aufgrund der übergeordneten Verbindung zwischen **Contact** und **Person** wird die benutzerdefinierte Funktion ebenfalls im Kontext der einzelnen **Person**-Datenelemente ausgeführt. So wird die Funktion jedes Mal, wenn in der Zielkomponente ein neues Person-Datenelement erstellt wird, aufgerufen, um das Element **Details** der jeweiligen Person zu befüllen. Diese Funktion hat drei Input-Parameter. Mit dem ersten (**OfficeName**) werden Daten aus dem Input-Parameter des Mappings ausgelesen. Die Quelldaten für diesen Parameter könnten genauso gut durch das Quelldatenelement **Name** bereitgestellt werden, ohne dass sich die Mapping-Ausgabe auf irgendeine Art ändern würde. Der Ausgangswert ist in beiden Fällen derselbe und stammt aus einem Parent-Kontext. Intern verkettet die Look-up-Funktion die als Argumente erhaltenen Werte und erzeugt einen einzigen Wert. Nähere Informationen zur `LookupPerson`-Funktion finden Sie unter Beispiel: Look-up und Verkettung 980.

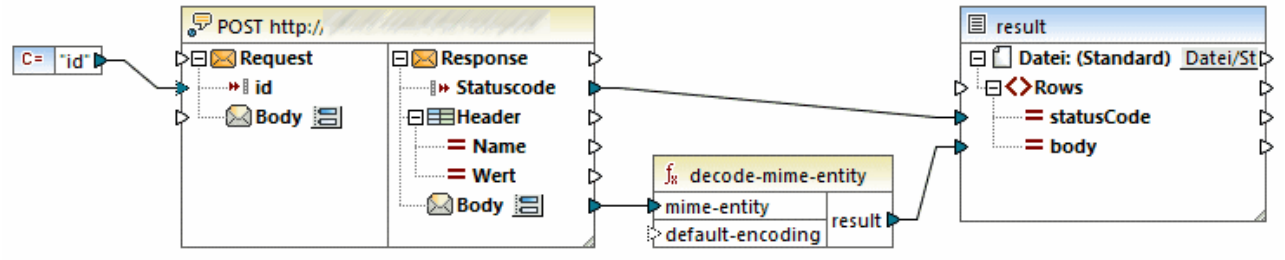
 © 2016-2022 Altova GmbH

8.3.2.1 Webservices

In Mappings, die Webservice-Aufrufe enthalten, kann derselbe Webservice, wenn es der Kontext zulässt, mehrmals aufgerufen werden, was normalerweise vermieden werden sollte. Betrachten Sie die folgenden Fälle:

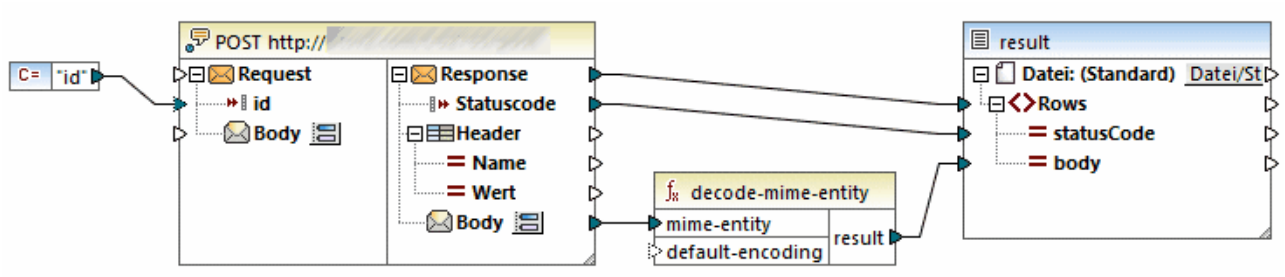
- Wenn die Verbindung zum **Request**-Datenelement der Webservice-Komponente eine Wertesequenz überträgt, wird der Webservice mehrmals aufgerufen, nämlich für jedes Datenelement in der Sequenz einmal.
- Wenn ein Webservice sich innerhalb einer benutzerdefinierten Funktion befindet und die benutzerdefinierte Funktion während der Mapping-Ausführung mehrmals aufgerufen wird, wird der Webservice mindestens genauso oft aufgerufen. Dies kann durch Konfiguration der Einstellungen für die benutzerdefinierte Funktion vermieden werden. Nähere Informationen dazu finden Sie unter Benutzerdefinierte Funktionen[1337].
- Normalerweise soll das Datenelement **Response** eines Webservice mit einem Zieldatenelement verbunden werden, welches:
  - erwartungsgemäß nur einmal erstellt wird (z.B. eine einfache Output-Komponente oder das oberste Datenelement einer komplexen Zielkomponente)
  - sich nicht in einem Parent-Kontext befindet, für den mehrere Iterationen über dasselbe Datenelement (und daher möglicherweise mehrere Webservice-Aufrufe) erforderlich sind.

Was den dritten oben beschriebenen Fall betrifft, könnte dies dazu führen, dass ein Webservice mehrmals aufgerufen wird, obwohl die Mapping-Ausgabe die erwartete wäre. So wird z.B. der unten gezeigte Webservice aufgrund der fehlenden Verbindung zwischen **Response** und **Rows** zwei Mal aufgerufen.



Wie gewöhnlich beginnt die Mapping-Ausführung beim Ziel-Root-Element (in diesem Fall **Rows**). **Rows** hat keine Verbindung, daher arbeitet sich MapForce durch die Hierarchie nach unten. Der erste Webservice-Aufruf wird gemacht, um das Zieldatenelement **statusCode** zu befüllen. Als nächstes muss das Datenelement **body** befüllt werden, daher wird der zweite Webservice-Aufruf durchgeführt. Gäbe es mehr Child-Elemente von **Rows,** die mit dem Webservice verbunden wären, würde für jedes davon ein neuer Aufruf durchgeführt.

Grund dafür ist, dass im oben gezeigten Mapping weder **statusCode** noch **body** einen expliziten Parent-Kontext haben (d.h. ihr übergeordnetes Datenelement ist nicht mit einem Quelldatenelement verbunden). Infolgedessen können keine Daten aus deren Parent-Kontext wiederverwendet werden, sondern diese müssen einzeln aus der Quellkomponente, welche in diesem Fall zufällig ein Webservice ist, abgerufen werden. Dieses Problem kann ganz einfach gelöst werden, indem Sie, wie unten gezeigt, eine Verbindung von **Response** zu **Rows** ziehen:



Dank der Verbindung zwischen **Response** und **Rows** wird der Webservice nun nur einmal aufgerufen. Nach diesem ersten Aufruf befinden sich alle Webservice-Daten im aktuellen Kontext und stehen den Child-Datenelementen von **Rows** zur Verfügung, daher entfallen weitere redundante Webservice-Aufrufe.

8.3.2.2 Datenbanken

Um die Effizienz zu verbessern und Hardware- und Netzwerk-Ressourcen weniger zu beanspruchen, sollten Sie vermeiden, dass dieselbe Datenbank im selben Mapping mehrmals unnötig aufgerufen wird. Natürlich kann es immer wieder vorkommen, dass eine Datenbank aufgrund der Natur des jeweiligen Mappings mehrmals aufgerufen werden muss, beachten Sie jedoch die folgenden allgemeinen Richtlinien:

- Wenn Sie nur einen Datenbankaufruf benötigen, sollten Sie die Datenbankkomponente nach Möglichkeit nicht in einen Parent-Kontext platzieren, der einen wiederholten Aufruf der Datenbank erforderlich machen würde. Dies könnte z.B. der Fall sein, wenn Sie eine Datenbankkomponente innerhalb einer benutzerdefinierten Funktion platzieren, die als Input eine Sequenz von Werten erhält und daher für jedes Datenelement in der Sequenz aufgerufen wird, siehe auch Benutzerdefinierte Funktionen 1337 weiter unten. Normalerweise sind Variablen hilfreich, um Daten im selben Kontext zu sammeln, bevor Sie diese an die Zielkomponente übergeben.
- Wenn Sie Werte aus einer Datenbank aggregieren müssen (z.B. um die Anzahl der Datensätze mit Hilfe der Funktion **count** zu zählen), wird empfohlen, die Ausgabe der Aggregate-Funktion mit einer Variablen zu verbinden, wobei **compute-when=once**. Dadurch werden wiederholte Aufrufe der Datenbank vermieden. Ein Beispiel dazu finden Sie unter Beispiel: Zählen von Datenbanktabellenzeilen 891.
- Versuchen Sie alle Datenbankdaten in einem einzigen Aufruf zu extrahieren (z.B. mit einer SQL-SELECT-Anweisung oder einer gespeicherten Prozedur), anstatt dieselbe Datenbankkomponente mehrmals zum Mapping hinzuzufügen.
- Wenn Sie Daten aus mehreren Tabellen oder Ansichten aus derselben Datenbank extrahieren müssen, ist es ratsam, entweder eine Join-Komponente (im SQL-Modus) oder eine SQL-SELECT-Anweisung zu verwenden. Zweitere eignet sich besser, wenn Sie es vorziehen, die SQL SELECT-Anweisung selbst zu schreiben. Wenn Sie Datenbankdaten mit Nicht-Datenbankdaten oder Daten aus verschiedenen Datenbanken miteinander verbinden müssen, verwenden Sie Nicht-SQL-Joins. Um die Ausführung von Nicht-SQL-Joins in datenintensiven Mappings zu optimieren, führen Sie die Mappings mit der MapForce Server Advanced Edition aus.
- Wenn Sie Daten aus einer Datenbank filtern müssen, ist es effizienter anstelle eines Standardfilters eine SQL-WHERE-Komponente zu verwenden, da eine SQL-WHERE-Komponente in der Grammatik der entsprechenden Datenbank speziell für die Arbeit mit Datenbanken optimiert ist.

 © 2016-2022 Altova GmbH

8.3.2.3 Benutzerdefinierte Funktionen

Benutzerdefinierte Funktionen (UDFs = User Defined Functions) sind eigens erstellte Funktionen, die in ein Mapping eingebettet werden, in denen Sie die Inputs, Outputs und Verarbeitungslogik definieren. Jede benutzerdefinierte Funktion kann dieselben Komponentenarten wie ein Hauptmapping, darunter auch Webservices und Datenbanken, enthalten.

> Wenn eine UDF eine Datenbank oder einen Webservice enthält und es sich bei den Input-Daten für die UDF um eine Sequenz aus mehreren Werten handelt, wird die UDF standardmäßig mit jedem Input-Wert aufgerufen, was einen Datenbank- oder Webservice-Aufruf zur Folge hat.

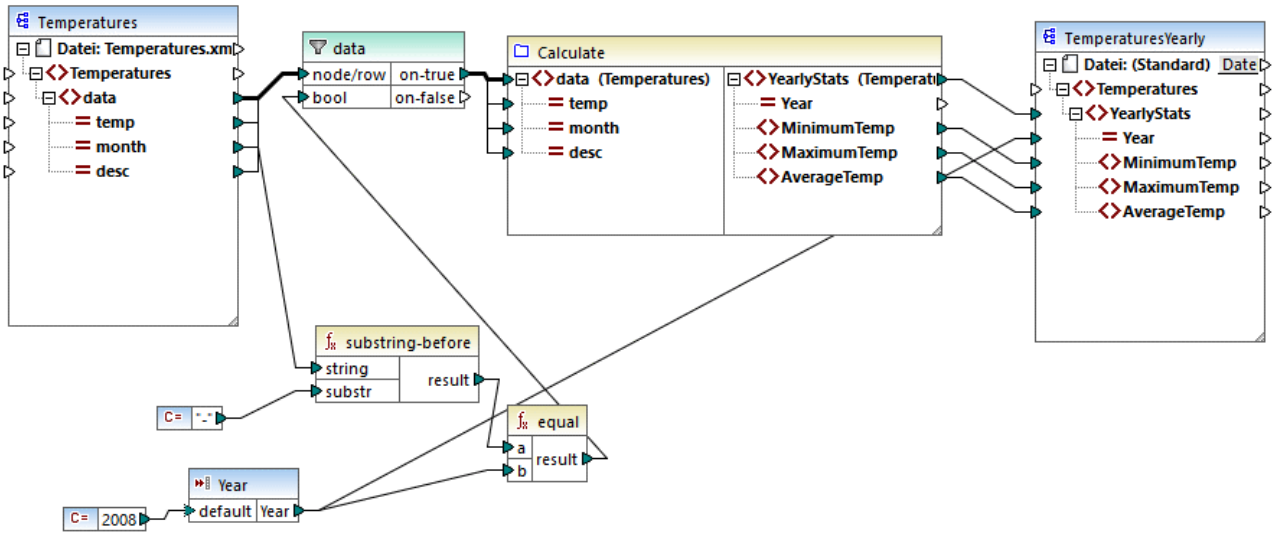
Das oben beschriebene Verhalten ist unter Umständen in Mappings, in denen die UDF wirklich so oft, wie Input-Werte vorhanden sind, aufgerufen werden muss, und in denen es keine andere Methode gibt, akzeptabel.

Wenn Sie das oben beschriebene Verhalten vermeiden möchten, können Sie die UDF so konfigurieren, dass sie, selbst wenn sie als Input eine Sequenz von Werten erhält, nur einmal aufgerufen wird. Normalerweise empfiehlt sich dies bei den UDFs, die an einer Gruppe von Werte ausgeführt werden, bevor sie ein Ergebnis zurückgeben (z.B. Funktionen, die Durchschnittswerte oder Summen berechnen).

Eine UDF so zu konfigurieren, dass sie im selben Funktionsaufruf mehrere Input-Werte akzeptiert, ist dann möglich, wenn es sich bei der UDF um eine reguläre UDF und nicht um eine "Inline"-UDF handelt (Nähere Informationen dazu finden Sie im Kapitel Benutzerdefinierte Funktionen 967.) Bei regulären Funktionen können Sie durch Aktivierung des Kontrollkästchens **Input ist eine Sequenz** definieren, dass der Input-Parameter eine Sequenz ist. Dieses Kontrollkästchen finden Sie in den Komponenteneinstellungen, die angezeigt werden, wenn Sie auf die Titelleiste eines Input-Parameters doppelklicken. Das Kontrollkästchen wirkt sich folgendermaßen darauf aus, wie oft die Funktion aufgerufen wird:

- Wenn Input-Daten mit einem **Sequenz**-Parameter verbunden werden, wird die benutzerdefinierte Funktion *nur einmal* aufgerufen und die vollständige Sequenz wird an die benutzerdefinierte Funktion übergeben.
- Wenn Input-Daten mit einem **Nicht-Sequenz**-Parameter verbunden werden, wird die benutzerdefinierte Funktion *für jedes einzelne Datenelement in der Sequenz nur einmal* aufgerufen.

Ein Beispiel dazu finden Sie im folgenden Demo-Mapping:
<Dokumente>\Altova\MapForce2023\MapForceExamples\InputIsSequence.mfd.



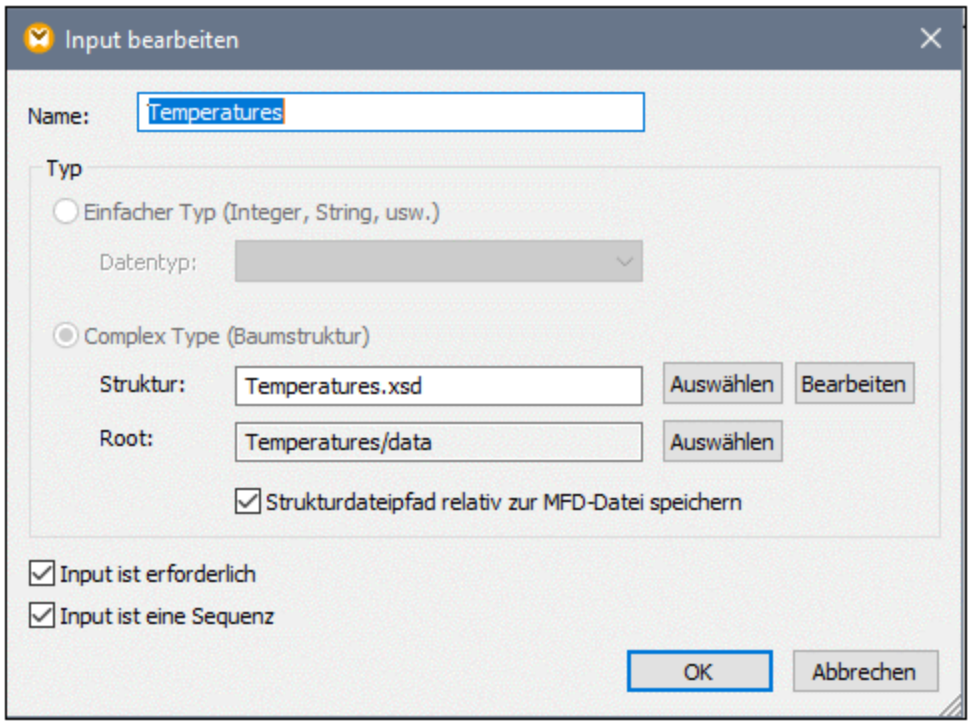
Im obigen Mapping sehen Sie ein Beispiel für einen typischen Fall einer UDF, die an einer Gruppe von Werten ausgeführt wird und für die daher alle Input-Werte in einem einzigen Aufruf abgerufen werden müssen. Die benutzerdefinierte Funktion **Calculate** gibt hier die Temperaturminima, -maxima und die Durchschnittstemperaturen zurück. Die Input-Daten dafür stammen aus einer XML-Quelldatei. Die erwartete Mapping-Ausgabe sieht folgendermaßen aus:

```
<Temperatures>
    <YearlyStats Year="2008">
        <MinimumTemp>-0.5</MinimumTemp>
        <MaximumTemp>24</MaximumTemp>
        <AverageTemp>11.6</AverageTemp>
    </YearlyStats>
</Temperatures>
```

Wie gewöhnlich beginnt die Mapping-Ausführung beim obersten Datenelement der Zielkomponente (in diesem Beispiel **YearlyStats)**. Um diesen Node zu befüllen, versucht das Mapping die Quelldaten aus der UDF zu holen, welche wiederum den Filter auslöst. Die Rolle des Filters ist es, nur Temperaturen aus dem Jahr 2008 an die UDF zu übergeben.

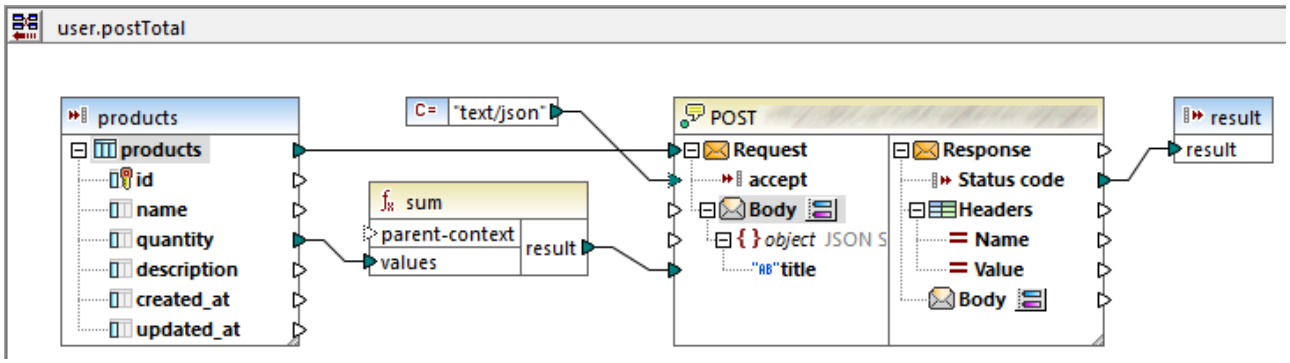
Für den Input-Parameter der UDF wurde das Kontrollkästchen **Input ist eine Sequenz** aktiviert (um dieses Kontrollkästchen zu sehen, doppelklicken Sie auf die Titelleiste der Funktion **Calculate,** um das Mapping der Funktion aufzurufen. Doppelklicken Sie anschließend auf die Titelleiste des Input-Parameters). Wie zuvor erwähnt, wird aufgrund der Option **Input ist eine Sequenz** die vollständige Sequenz der Werte als Input an die Funktion geliefert und die Funktion wird nur einmal aufgerufen.

© 2016-2022 Altova GmbH



Wäre das Kontrollkästchen **Input ist eine Sequenz** nicht aktiviert, wäre die UDF für jeden Wert in der Quellkomponente einzeln aufgerufen worden, sodass die Minima, Maxima und Durchschnittstemperaturen für jeden Wert einzeln berechnet würden, wodurch eine falsche Ausgabe erzeugt würde.

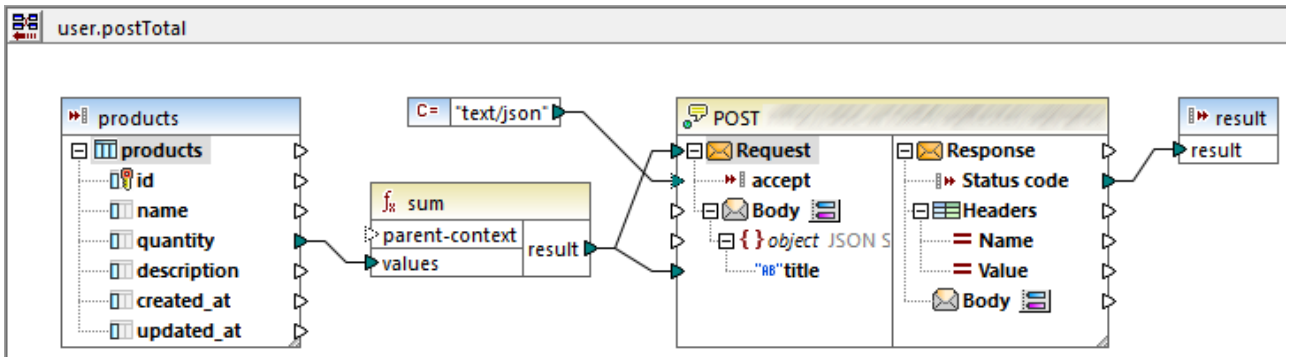
Durch Anwendung derselben Logik in komplexeren UDFs, die Datenbank- oder Webservice-Aufrufe enthalten, kann die Ausführung unter Umständen optimiert und unnötige Datenbank- oder Webservice-Aufrufe können vermieden werden. Bedenken Sie jedoch, dass das Kontrollkästchen **Input ist eine Sequenz** keinen Einfluss darauf hat, was mit der Wertesequenz *nach* ihrem Eintritt in die Funktion passiert, d.h. die eingehende Wertesequenz könnte auch mit dem Input eines Webservice verbunden werden, wodurch dieser mehrmals aufgerufen würde. Werfen Sie einen Blick auf das folgende Beispiel:



Die oben gezeigte UDF erhält aus dem externen Mapping eine Sequenz von Werten. Die an den Input-Parameter gelieferten Daten stammen in diesem Fall aus einer Datenbank. Für den Input-Parameter wurde die Option **Input ist eine Sequenz** aktiviert, daher wird die gesamte Sequenz in einem einzigen Aufruf an die Funktion geliefert. Mit der Funktion sollen mehrere **quantity**-Werte addiert und an einen Webservice gesendet

werden. Es wird genau ein Webservice-Aufruf erwartet. Bei Ausführung des Mappings wird der Webservice jedoch fehlerhafter Weise mehrmals aufgerufen. Der Grund dafür ist, dass der **Request**-Input des Webservice eine *Sequenz von Werten* anstatt eines einzigen Werts erhält

Um dieses Problem zu beheben, verbinden Sie den **Request**-Input des Webservice mit dem Ergebnis (result) der **sum**-Funktion. Das Ergebnis der Funktion ist ein einziger Wert, sodass auch der Webservice nur einmal aufgerufen wird:



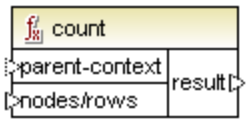
Normalerweise erzeugen Aggregatsfunktionen wie **sum**, **count**, einen einzigen Wert. Wenn es jedoch eine übergeordnete Verbindung gibt, die dies zulässt, können sie auch, wie im Beispiel: Ändern des Parent-Kontexts 1340 näher beschrieben, eine Sequenz von Werten erzeugen.

8.3.2.4 Beispiel: Ändern des Parent-Kontexts

Einige Mapping-Komponenten haben ein optionales **Parent-Kontext**-Datenelement.

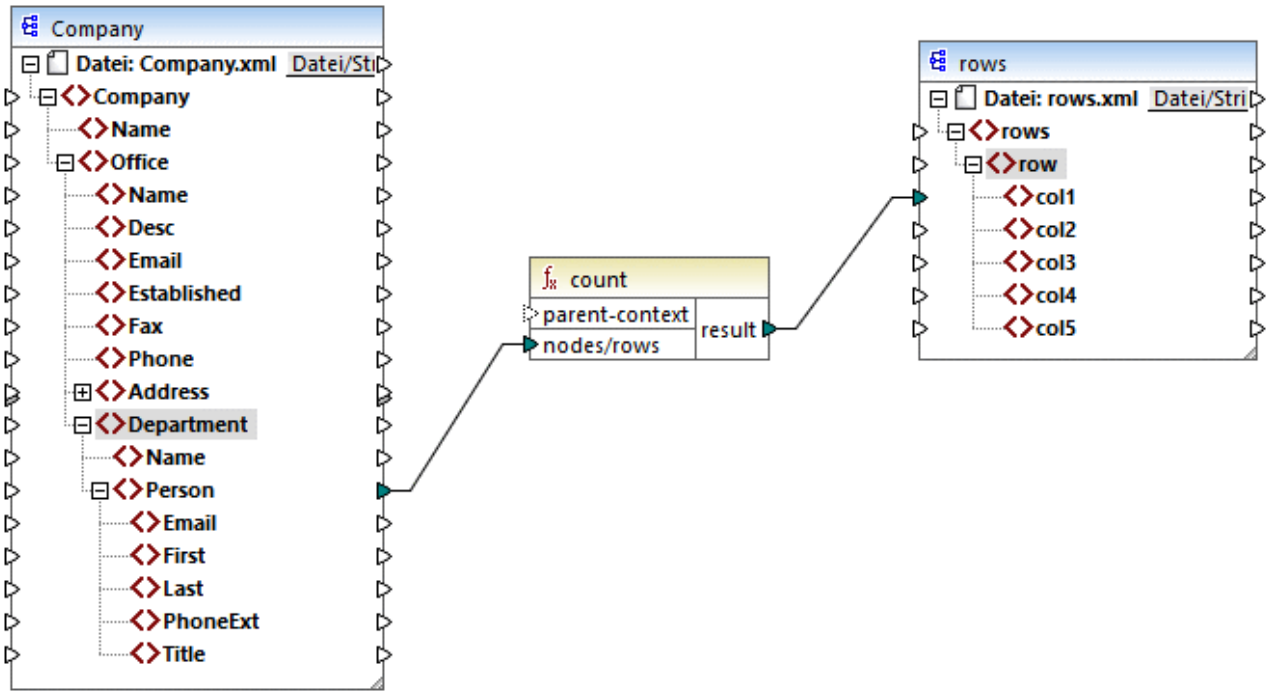
Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **min**, **max**, **avg**, **count**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.

Mit Hilfe dieses Datenelements können Sie den Mapping-Kontext, in dem diese Komponente verwendet werden soll, beeinflussen und somit die Mapping-Ausgabe ändern. Die Komponenten, die einen optionalen **Parent-Kontext** haben, sind: Aggregatfunktionen, Variablen und Join-Komponenten.



Um ein Beispiel dafür zu sehen, wie Sie mit der Änderung des Parent-Kontexts arbeiten können, öffnen Sie das folgende Mapping: **<Dokumente>\Altova\MapForce2023\MapForceExamples\ParentContext.mfd**.

 © 2016-2022 Altova GmbH



Die XML-Quelldatei im Mapping oben enthält einen einzigen **Company**-Node, der zwei **Office**-Nodes enthält. Jeder **Office**-Noden enthält mehrere **Department**-Nodes und jeder **Department**-Node enthält mehrere **Person**-Nodes. Wenn Sie die XML-Datei in einem XML-Editor öffnen, sehen Sie folgende Aufteilung von Personen nach Büro (Office) und Abteilung (Department):

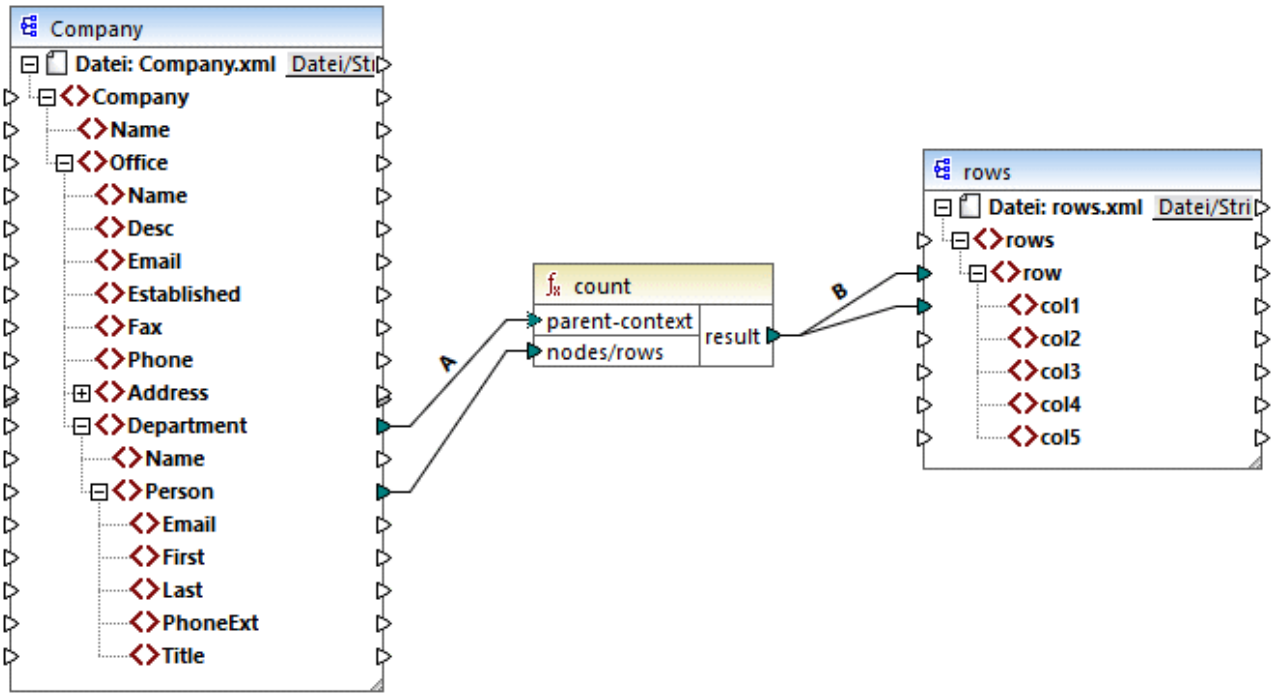
Office	Department	Personenzahl
Nanonull, Inc.	Administration	3
	Marketing	2
	Engineering	6
	IT & Technical Support	4
Nanonull Partners, Inc.	Administration	2
	Marketing	1
	IT & Technical Support	3

Im Mapping werden alle Personen in allen Abteilungen gezählt. Dazu wird die **count**-Funktion aus der `core`-Bibliothek verwendet. Wenn Sie auf das Register **Ausgabe** klicken, um eine Vorschau auf das Mapping anzuzeigen, sehen Sie, dass als Ergebnis ein einziger Wert, nämlich **21**, erzeugt wird, was der Gesamtanzahl aller Personen in der XML-Quelldatei entspricht.

Das Mapping funktioniert folgendermaßen:

- Das Mapping wird wie gewöhnlich ab dem obersten Node der Zielkomponente (in diesem Beispiel **rows**) ausgeführt. **rows** hat keine eingehende Verbindung. Infolgedessen wird zwischen **Company** (oberstes Datenelement der Quellkomponente) und **rows** (oberstes Datenelement der Zielkomponente) ein impliziter Mapping-Kontext erstellt.
- Das Resultat der Funktion ist ein einziger Wert, da die Quelldatei nur eine Firma (Company) enthält.
- Um das Zieldatenelement **col1** zu befüllen, führt MapForce die **count**-Funktion im oben erwähnten *impliziten Parent-Kontext* aus, d.h. es werden alle **Person**-Nodes aus allen Büros und allen Abteilungen gezählt.

Mit Hilfe des Arguments **parent-context** der Funktion können Sie den Mapping-Kontext ändern. Dadurch kann z.B. die Anzahl der Personen in jeder einzelnen Abteilung gezählt werden. Ziehen Sie dazu, wie unten gezeigt, zwei weitere Verbindungen:



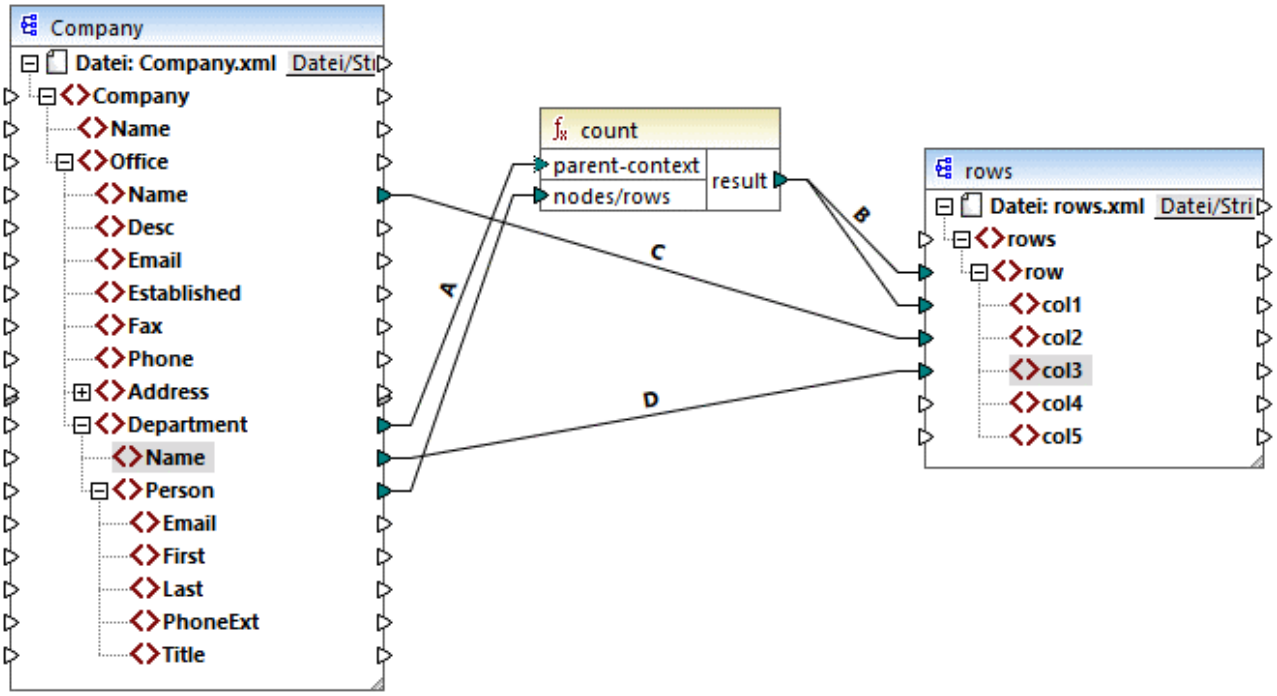
Im oben gezeigten Mapping wird durch die Verbindung A der Parent-Kontext der **count**-Funktion in **Department** geändert, wodurch die Funktion die Anzahl der Personen in jeder einzelnen Abteilung zählt. Beachten Sie, dass die Funktion nun eine Sequenz von Ergebnissen anstatt eines einzigen Ergebnisses zurückgibt, da in der Quelldatei mehrere Abteilungen (Department) vorhanden sind. Aus diesem Grund wurde die Verbindung B erstellt: Für jedes Datenelement in der erzeugten Sequenz wird in der Zieldatei eine neue Zeile (row) erstellt. Die Mapping-Ausgabe hat sich nun entsprechend geändert (Beachten Sie, dass die Zahlen genau der Anzahl der Personen in den einzelnen Abteilungen entspricht):

```
<rows>
    <row>
        <col1>3</col1>
    </row>
    <row>
        <col1>2</col1>
```


© 2016-2022 Altova GmbH

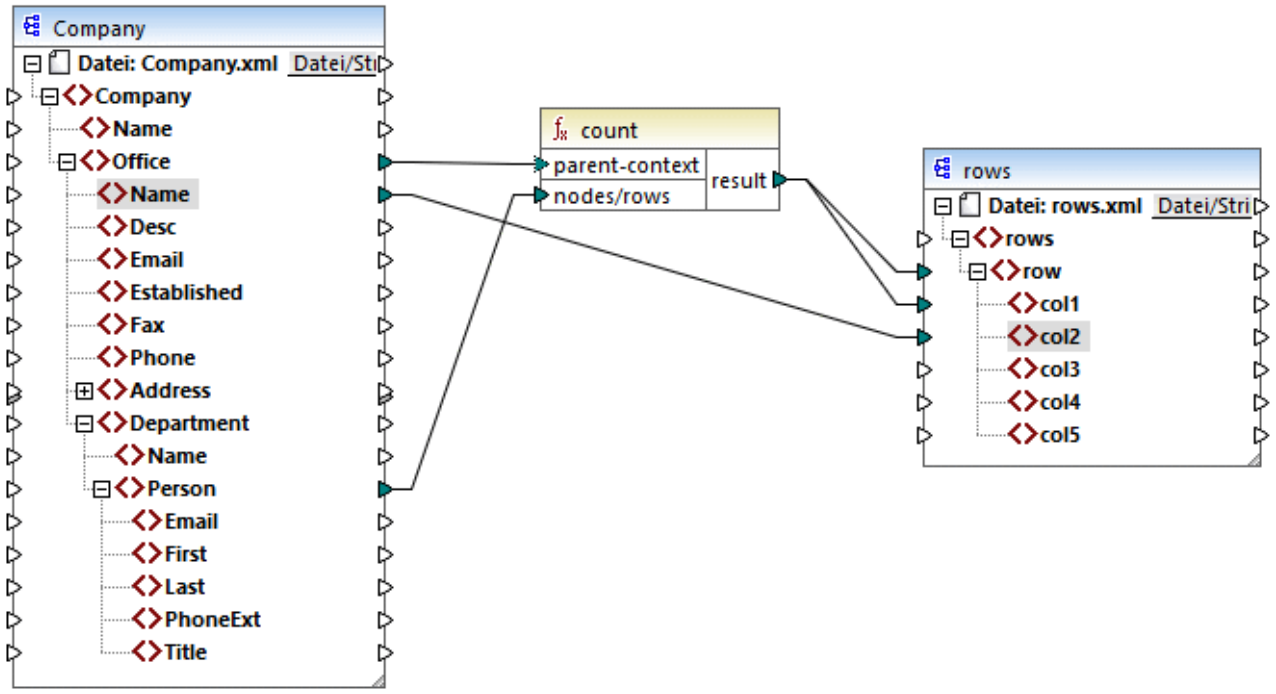
```
    </row>
    <row>
        <col1>6</col1>
    </row>
    <row>
        <col1>4</col1>
    </row>
    <row>
        <col1>2</col1>
    </row>
    <row>
        <col1>1</col1>
    </row>
    <row>
        <col1>3</col1>
    </row>
</rows>
```

Da im aktuellen Mapping für jede Abteilung (Department) eine Zeile (row) erstellt wird, können Sie durch Ziehen der Verbindungen C und D den Office-Namen und den Abteilungsnamen ebenfalls in die Zieldatei kopieren:



Dadurch wird in der Ausgabe nicht nur die Anzahl der Personen, sondern auch der entsprechende Büro- und Abteilungsname angezeigt.

Wenn Sie die Anzahl der Personen in den einzelnen Büros (Office) zählen möchten, verbinden Sie den Parent-Kontext der **count**-Funktion mit dem Datenelement **Office** in der Quellkomponente.



Mit den oben gezeigten Verbindungen gibt die **count**-Funktion ein Ergebnis für jedes Büro zurück. Die Quelldatei enthält zwei Büros, daher gibt die Funktion nun zwei Sequenzen zurück. Infolgedessen enthält die Ausgabe zwei Zeilen (row), von denen jede die Anzahl der Personen in diesem Büro enthält:

```
<rows>
    <row>
        <col1>15</col1>
        <col2>Nanonull, Inc.</col2>
    </row>
    <row>
        <col1>6</col1>
        <col2>Nanonull Partners, Inc.</col2>
    </row>
</rows>
```

8.3.3 Prioritätskontext

Der Prioritätskontext ist eine Methode, mit der Sie festlegen können, in welcher Reihenfolge die Input-Parameter einer Funktion ausgewertet werden. Unter Umständen muss ein Prioritätskontext definiert werden, wenn in Ihrem Mapping Daten aus zwei nicht miteinander in Zusammenhang stehenden Quelldateien verbunden werden müssen.

Um zu verstehen, wie der Prioritätskontext funktioniert, denken Sie daran, dass bei Ausführung eines Mappings die Verbindung zu einem Input-Datenelement eine *Sequenz* aus mehreren Werten übertragen kann. Bei Funktionen mit zwei Input-Parametern bedeutet dies, dass zwei Schleifen erstellt werden müssen, von denen eine zuerst verarbeitet werden muss. Die zuerst verarbeitete Schleife ist die äußere Schleife. So erhält z.B. die

© 2016-2022 Altova GmbH

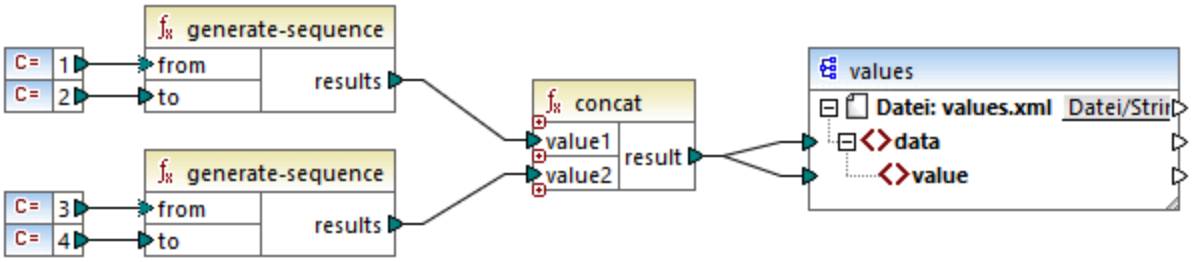
equal-Funktion zwei Parameter: *a* und *b*. Wenn sowohl *a* als auch *b* eine Wertesequenz erhält, verarbeitet MapForce diese folgendermaßen:

- Für jede Instanz von *a*
  - Für jede Instanz von *b*
    - Ist *a* gleich *b*?

Wie Sie oben sehen, wird jedes *b* im Kontext eines jeden *a* ausgewertet. Mit Hilfe des Prioritätskontexts können Sie die Verarbeitungslogik ändern, sodass jedes a im Kontext eines jeden *b* ausgewertet wird, d.h. Sie können die innere Schleife mit der äußeren vertauschen, z.B:

- Für jede Instanz von *b*
  - Für jede Instanz von *a*
    - Ist *a* gleich *b*?

Betrachten wir nun ein Mapping, in dem sich der Prioritätskontext auf die Mapping-Ausgabe auswirkt. Die **concat**-Funktion im unten gezeigten Mapping hat zwei Input-Parameter. Jeder Input-Parameter ist eine Sequenz, die mit Hilfe der **generate-sequence**-Funktion generiert wurde. Die erste Sequenz ist "1,2"und die zweite Sequenz ist "3,4".



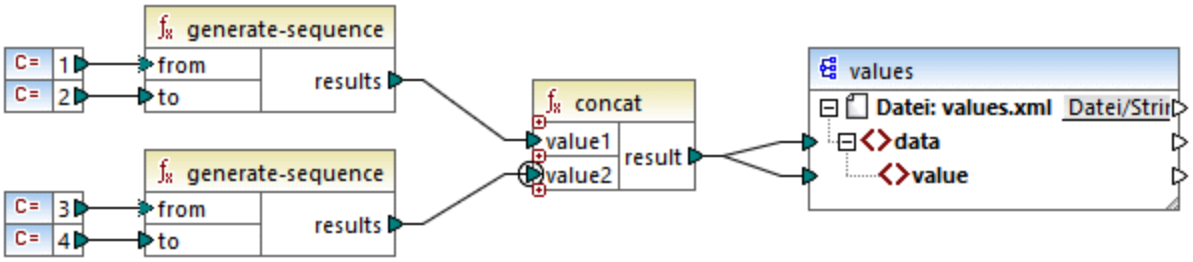
Führen wir das Mapping zuerst aus, ohne einen Prioritätskontext zu definieren. Die **concat**-Funktion beginnt zuerst mit der Auswertung der obersten Sequenz, daher werden die Werte in der folgenden Reihenfolge miteinander kombiniert:

- 1 mit 3
- 1 mit 4
- 2 mit 3
- 2 mit 4

Dies wirkt sich auf die Mapping-Ausgabe folgendermaßen aus:

```
<data>
    <value>13</value>
    <value>14</value>
    <value>23</value>
    <value>24</value>
</data>
```

Wenn Sie mit der rechten Maustaste auf den zweiten Input-Parameter klicken und im Kontextmenü den Eintrag **Prioritätskontext** auswählen, wird dieser zum Prioritätskontext. Wie Sie in der Abbildung unten sehen, ist der Prioritätskontext-Input mit einem Kreis umrandet.



Dieses Mal wird der zweite Input-Parameter zuerst ausgewertet. Die `concat`-Funktion verkettet nach wie vor dieselben Werte, diesmal wird jedoch die Sequenz '3,4' zuerst ausgewertet. Die Ausgabe wird folglich zu:

```
<data>
    <value>13</value>
    <value>23</value>
    <value>14</value>
    <value>24</value>
</data>
```

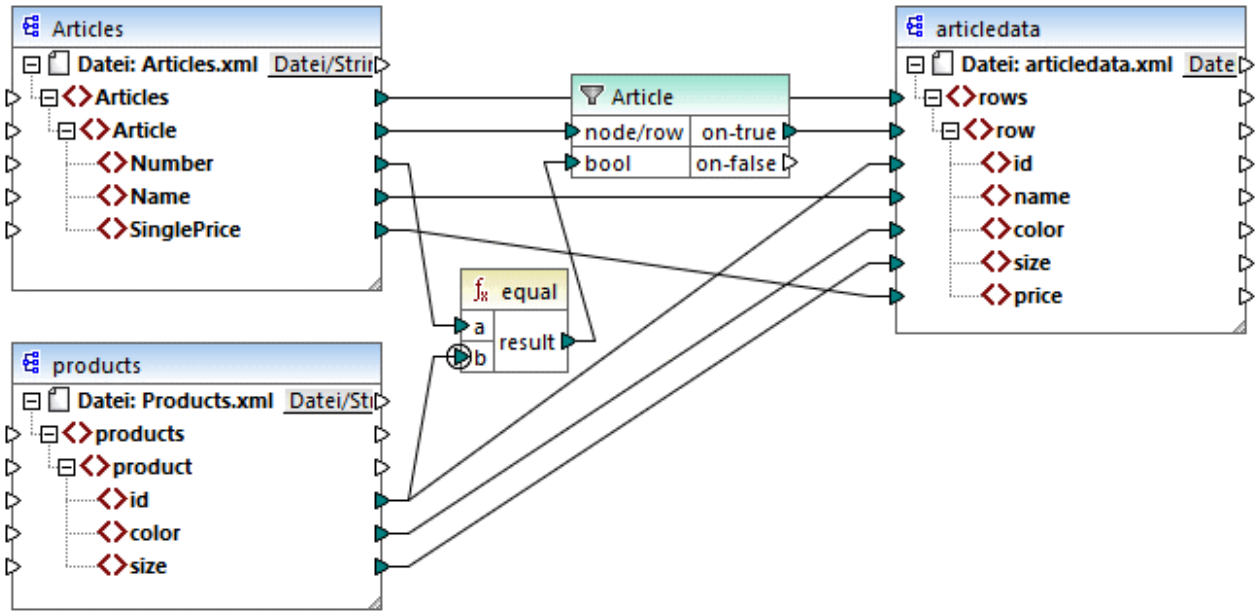
Bisher wurde nur der theoretische Hintergrund des Prioritätskontexts beschrieben. Ein praktisches Anwendungsszenario finden Sie im Beispiel: Filtern mit Prioritätskontext 1346.

8.3.3.1 Beispiel: Filtern mit Prioritätskontext

Wenn eine Funktion mit einem Filter verbunden wird, wirkt sich der Prioritätskontext nicht nur auf die Funktion selbst, sondern auch auf die Auswertung des Filters aus. Im Mapping unten sehen Sie ein Beispiel für einen typischen Fall, in dem ein Prioritätskontext definiert werden muss, um das korrekte Ergebnis zu erhalten. Sie finden dieses Mapping unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2023\MapForceExamples\FilterWithPriority.mfd.

Anmerkung: In diesem Mapping werden XML-Komponenten verwendet, doch gilt dieselbe unten beschriebene Logik auch für allen anderen in MapForce verwendeten Komponententypen, darunter auch für EDI, JSON usw. Im Fall von Datenbanken empfiehlt sich eine Filterung mittels SQL WHERE 361-Komponenten anstelle von Standardfiltern.

© 2016-2022 Altova GmbH



Ziel des oben gezeigten Mappings ist es, Daten aus **Articles.xml** in eine neue XML-Datei mit einem anderen Schema, nämlich **articledata.xml,** zu kopieren. Gleichzeitig sollen die Informationen zu jedem Artikel (Article) in der Datei **Products.xml** abgerufen und mit dem entsprechenden Artikeldatensatz verbunden werden. Beachten Sie, dass jeder Datensatz in **Articles.xml** eine Nummer (**Number)** und jeder Datensatz in **Products.xml** eine **id** hat. Wenn diese beiden Werte identisch sind, sollen alle andere Werte (**Name**, **SinglePrice**, **color**, **size**) in dieselbe Zeile (**row)** in der Zielkomponente kopiert werden.

Dieses Ziel wurde durch Hinzufügen eines Filters erreicht. Für jeden Filter wird als Input eine Boolesche Bedingung benötigt. Nur die Nodes/Zeilen, die die Bedingung erfüllen, werden in die Zielkomponente kopiert. Zu diesem Zweck gibt es im Mapping eine `equal`-Funktion. Mit der `equal`-Funktion wird überprüft, ob die Artikelnummer und die Produkt-ID in beiden Quelldateien identisch sind. Das Ergebnis wird anschließend als Input für den Filter bereitgestellt. Bei **true** wird das **Article**-Datenelement in die Zielkomponente kopiert.

Beachten Sie, dass für den zweiten Input-Parameter der zweiten `equal`-Funktion ein Prioritätskontext definiert wurde. Der Prioritätskontext macht in diesem Mapping einen großen Unterschied. Wenn er nicht definiert wird, führt dies zu einer falschen Mapping-Ausgabe.

Anfängliches Mapping: Kein Prioritätskontext

Hier die Mapping-Logik ohne Prioritätskontext:

- Gemäß der allgemeinen Mapping-Regel wird für jedes **Article**-Element, das die Filter-Bedingung erfüllt, in der Zielkomponente eine neuen Zeile (**row**) erstellt. Dafür sorgt die Verbindung zwischen **Article** und **row** (via die Funktion und den Filter).
- Mit dem Filter wird für jeden Artikel die Bedingung überprüft. Dazu iteriert der Filter durch alle Produkte und bringt mehrere Produkte in den aktuellen Kontext.
- Um in der Zielkomponente das Element **id** zu befüllen, geht MapForce nach der allgemeinen Regel vor (Erstelle für jedes Datenelement in der Quelldatei ein Datenelement in der Zielkomponente). Wie jedoch oben erläutert, befinden sich alle Produkte aus **Products.xml** im aktuellen Kontext. Es gibt keine Verbindung zwischen **product** und irgendeinem anderen Node in der Zielkomponente, damit nur

die **id** eines bestimmten Produkts ausgelesen wird. Folglich werden für jeden Artikel (**Article**) mehrere **id**-Elemente in der Zielkomponente erstellt. Dasselbe geschieht mit **color** und **size**.

Zusammenfassung: Die Datenelemente aus **Products.xml** haben den Kontext des Filters (der durch jedes Produkt iterieren muss). Daher werden die **id-**, **color-** und **size**-Werte so oft in die einzelnen **row**-Elemente in der Zielkomponente kopiert, wie sich in der Quelldatei Produkte befinden und es wird eine falsche Ausgabe wie die unten gezeigte generiert:

```
<rows>
    <row>
        <id>1</id>
        <id>2</id>
        <id>3</id>
        <name>T-Shirt</name>
        <color>red</color>
        <color>blue</color>
        <color>green</color>
        <size>10</size>
        <size>20</size>
        <size>30</size>
        <price>25</price>
    </row>
</rows>
```

Lösung A: Verwendung eines Prioritätskontexts

Das obige Problem wurde durch Hinzufügen eines Prioritätskontexts zur Funktion, die die Boolesche Bedingung des Filters berechnet, gelöst.

Wenn in diesem Fall der zweite Input-Parameter der `equal`-Funktion als Prioritätskontext definiert wird, erhält die Sequenz, die aus **Products.xml** eingeht, Priorität. Dies wird in die folgende Mapping-Logik übersetzt:

- Befülle für jedes Produkt den Input **b** der `equal`-Funktion (anders ausgedrückt: gib **b** Priorität). In dieser Phase befinden sich die Informationen des aktuellen Produkts im Kontext.
- Befülle für jeden Artikel den Input **a** der `equal`-Funktion und überprüfe, ob die Filterbedingung "true" ist. Falls ja, setze auch die Artikelinformationen in den aktuellen Kontext.
- Kopiere als nächstes die Artikel- und Produktinformationen aus dem aktuellen Kontext in die entsprechenden Datenelemente in der Zielkomponente.

Mit der obigen Mapping-Logik wird eine korrekte Ausgabe erzeugt, z.B:

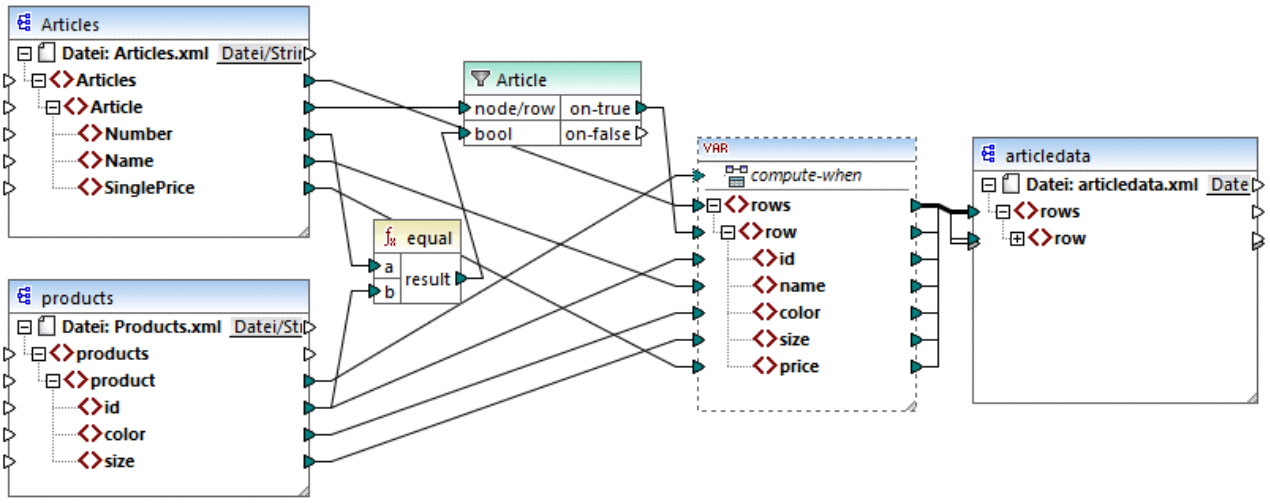
```
<rows>
    <row>
        <id>1</id>
        <name>T-Shirt</name>
        <color>red</color>
        <size>10</size>
        <price>25</price>
    </row>
</rows>
```


© 2016-2022 Altova GmbH

Lösung B: Verwendung einer Variablen

Als Alternativlösung dazu können Sie auch mit Hilfe einer Zwischenvariablen die einzelnen Artikel und Produkte, die die Filterbedingung erfüllen, in denselben Kontext bringen. Variablen eignen sich für derartige Szenarien, da Sie damit Daten temporär im Mapping speichern und dadurch bei Bedarf den Kontext ändern können.

In Szenarien wie diesem können Sie eine Variable zum Mapping hinzufügen, die dasselbe Schema wie die Zielkomponente hat. Klicken Sie im Menü **Einfügen** auf **Variablen** und geben Sie das Schema **articledata.xsd** als Struktur an, wenn Sie gefragt werden.



Im oben gezeigten Mapping geschieht Folgendes:

- Es wird kein Prioritätskontext mehr verwendet. Statt dessen gibt es eine Variable, die dieselbe Struktur wie die Zielkomponente hat.
- Das Mapping beginnt wie gewöhnlich am Ziel-Root-Node. Bevor die Zielkomponente befüllt wird, werden die Daten in der Variablen gesammelt.
- Die Variable wird im Kontext jedes einzelnen Produkts berechnet, da eine Verbindung von **product** zum **compute-when**-Input der Variablen besteht.
- Die Filterbedingung wird daher im Kontext jedes einzelnen Produkts überprüft. Nur, wenn die Bedingung "true" ist, wird die Variablenstruktur befüllt und an die Zielkomponente übergeben.

8.3.4 Mehrere Zielkomponenten

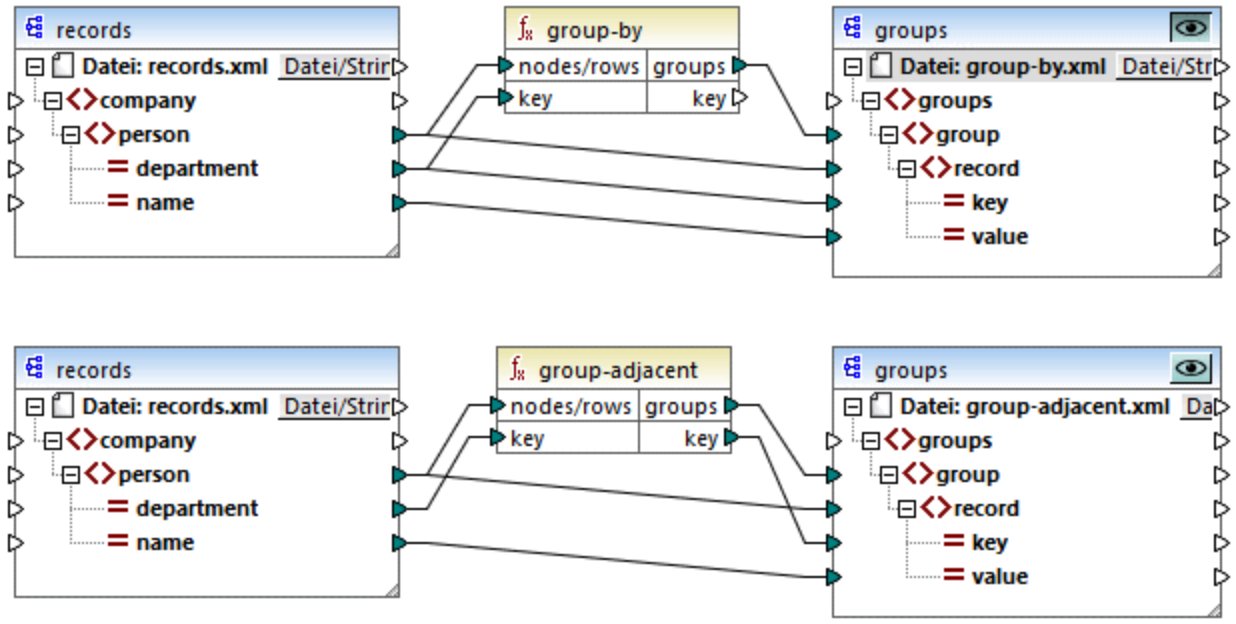
Ein Mapping kann mehrere Quell- und Zielkomponenten haben. Wenn mehrere Zielkomponenten vorhanden sind, können Sie in der MapForce-Vorschau immer nur eine Komponentenausgabe auf einmal anzeigen, nämlich diejenige, die Sie durch Klick auf die **Vorschau-**Schaltfläche markiert haben. In anderen Ausführungsumgebungen (MapForce Server oder generierter Code) werden alle Zielkomponente der Reihe nach ausgeführt und es wird die entsprechende Ausgabe für die einzelnen Komponente erzeugt.

Standardmäßig werden Zielkomponenten von oben nach unten und von links nach rechts verarbeitet. Bei Bedarf können Sie diese Reihenfolge durch Ändern der Position der Zielkomponenten im Mapping-Fenster

beeinflussen. Der Referenzpunkt ist jeweils die linke obere Ecke einer Komponente. Beachten Sie die folgenden Punkte:

- Wenn zwei Komponenten dieselbe vertikale Position haben, so wird zuerst die am weitesten links liegende Komponente verarbeitet.
- Wenn zwei Komponenten dieselbe horizontale Position haben, so wird die weiter oben liegende Komponente zuerst verarbeitet.
- In den seltenen Fällen, in denen zwei Komponenten sich an exakt derselben Stelle befinden, wird automatisch eine eindeutige interne Komponenten-ID verwendet. Damit ist eine genau definierte Reihenfolge definiert, die allerdings nicht geändert werden kann.

Um ein Beispiel dafür zu sehen, wie dies funktioniert, öffnen Sie das folgende Demo-Mapping: **<Dokumente>\Altova\MapForce2023\MapForceExamples\GroupingFunctions.mfd**. Dieses Mapping besteht aus mehreren Quell- und Zielkomponenten. Unten sehen Sie nur ein Fragment davon.



Gemäß der Regel ist die Standard-Verarbeitungsreihenfolge dieses Mappings in MapForce Server und im generierten Code von oben nach unten. Sie können dies überprüfen, indem Sie z.B. XSLT 2.0-Code generieren:

1. Klicken Sie im Menü **Datei** auf **Code generieren in | XSLT 2.0**.
2. Wenn Sie dazu aufgefordert werden, wählen Sie ein Zielverzeichnis für den generierten Code aus.

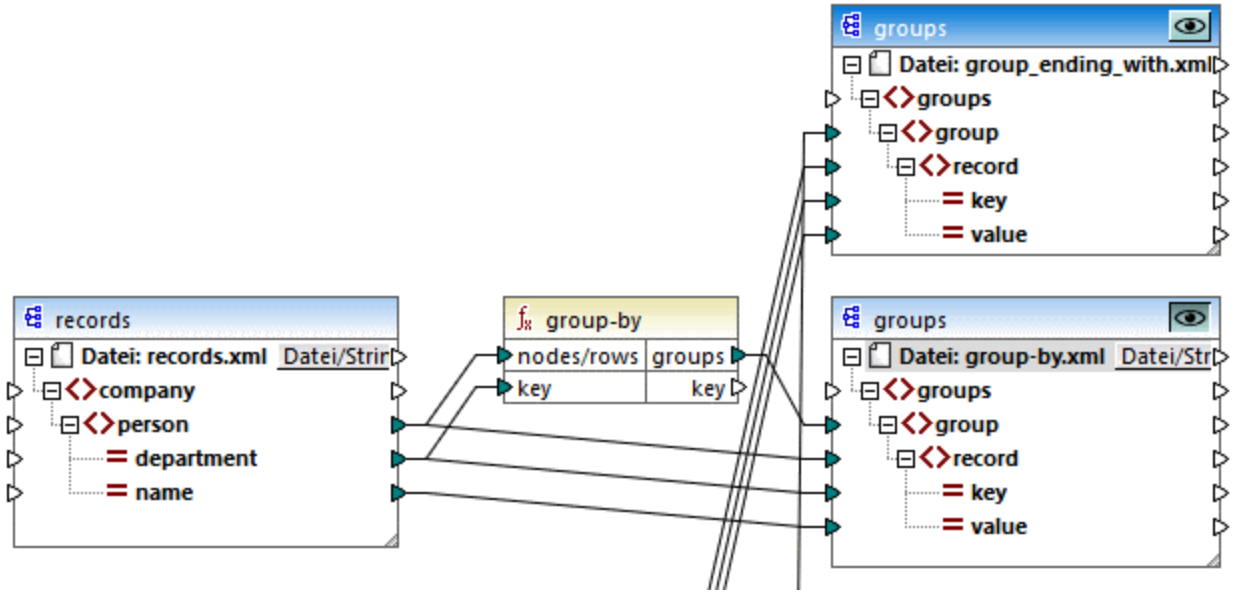
Das Zielverzeichnis enthält nach der Generierung mehrere XSLT-Dateien und eine **DoTransform.bat**-Datei. Letztere kann mit RaptorXML Server (eigene Lizenz erforderlich) ausgeführt werden. Die Datei **DoTransform.bat** verarbeitet Komponenten in derselben Reihenfolge, wie diese im Mapping definiert wurden, nämlich von oben nach unten. Überprüfen Sie dies, in dem Sie sich den `--output`-Parameter der einzelnen Transformationen ansehen.

```
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-by.xml" --xml-
validation-error-as-warning=true %* "MappingMapTogroups.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```


© 2016-2022 Altova GmbH

```
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-adjacent.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups2.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-into-blocks.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups3.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records-v2.xml" --output="group-starting-
with.xml" --xml-validation-error-as-warning=true %* "MappingMapTogroups4.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records-v3.xml" --output="group_ending_with.xml"
--xml-validation-error-as-warning=true %* "MappingMapTogroups5.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

Mit der letzten Transformation wird eine Ausgabedatei namens **group-ending-with.xml** erzeugt. Verschieben wir diese Zielkomponente nun im Mapping ganz nach oben:



Wenn Sie nun erneut XSLT 2.0-Code generieren, ändert sich die Verarbeitungsreihenfolge entsprechend:

```
RaptorXML xslt --xslt-version=2 --input="records-v3.xml" --output="group_ending_with.xml"
--xml-validation-error-as-warning=true %* "MappingMapTogroups.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-by.xml" --xml-
validation-error-as-warning=true %* "MappingMapTogroups2.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-adjacent.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups3.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records.xml" --output="group-into-blocks.xml" --
xml-validation-error-as-warning=true %* "MappingMapTogroups4.xslt"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
RaptorXML xslt --xslt-version=2 --input="records-v2.xml" --output="group-starting-
with.xml" --xml-validation-error-as-warning=true %* "MappingMapTogroups5.xslt"
```

```
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

Mit dem ersten Aufruf im oben gezeigten Codefragment wird nun **group-ending-with.xml** erzeugt.,

Sie können die Verarbeitungsreihenfolge auch in andere Codesprachen und in kompilierten MapForceServer-Ausführungsdateien (.mfx) auf ähnliche Weise ändern.

Verkettete Mappings

Dieselbe oben beschriebene Verarbeitungsreihenfolge gilt auch für verkettete Mappings. Die verkettete Mapping-Gruppe wird jedoch als eine einzige Einheit behandelt. Wenn Sie die Zwischen- oder Endkomponente eines einzigen verketteten Mappings verschieben, hat dies keine Auswirkung auf die Verarbeitungsreihenfolge. Die Position der Endkomponenten von einzelnen Gruppen hat nur dann einen Einfluss auf die Verarbeitungsreihenfolge, wenn mehrere Ketten bzw. mehrere Zielkomponenten in einem Mapping vorhanden sind.

- Wenn zwei Endkomponenten dieselbe vertikale Position einnehmen, so wird zuerst die weiter links gelegene verarbeitet.
- Wenn zwei Endkomponenten dieselbe horizontale Position einnehmen, so wird zuerst die weiter oben gelegene verarbeitet.
- In den seltenen Fällen, in denen zwei Komponenten sich an exakt derselben Stelle befinden, wird automatisch eine eindeutige interne Komponenten-ID verwendet. Damit ist eine genau definierte Reihenfolge definiert, die allerdings nicht geändert werden kann.

© 2016-2022 Altova GmbH

8.4 Ausführen von Mappings mit Authentifizierung

Manchmal müssen mit MapForce erstellte Mappings auf externe Ressourcen, für die eine Authentifizierung erforderlich ist (z.B. ein Webservice), zugreifen. Zu diesem Zweck bietet MapForce (und serverseitig MapForce Server und FlowForce Server) Unterstützung für die folgenden Authentifizierungsmöglichkeiten:

- Benutzername und Passwort. Diese werden normalerweise benötigt, um Webservices aufzurufen, für die eine einfache HTTP-Authentifizierung erforderlich ist.
- OAuth 2.0-Anmeldeinformationen. Diese werden benötigt, um Webservices aufzurufen, für die eine OAuth 2.0-.Authentifizierung erforderlich ist. Beachten Sie, dass OAuth 2.0-Anmeldeinformationen serverseitig in der MapForce Server Advanced Edition und der FlowForce Server Advanced Edition unterstützt werden.

Wenn Sie in MapForce ein Mapping erstellen, haben Sie die folgenden Optionen, um sich beim Webservice zu authentifizieren und das Mapping erfolgreich auszuführen:

- Definition von *Anmeldeinformationen*. Dies ist die empfohlene und flexibelste Methode. Ein Anmeldeinformationsobjekt kann die vollständigen Authentifizierungsinformationen enthalten oder nur eine benannte Referenz sein, die keine sensiblen Daten enthält. Im zweiten Fall müssen die eigentlichen Anmeldeinformationen (z.B. das Passwort) zur Mapping-Laufzeit bereitgestellt werden. Sie haben die Möglichkeit, die mit den Anmeldeinformationen verknüpften sensiblen Daten in verschlüsselter Form als Teil der Mapping-Design-Datei (*.mfd) zu speichern. Wenn das Mapping ohne manuelles Eingreifen mit MapForce Server in anderen Umgebungen ausgeführt werden soll, können Sie die Anmeldeinformationen übertragbar, also portabel, machen. Die portablen Anmeldeinformationen müssen den eigentlichen Benutzernamen und das Passwort nicht unbedingt enthalten (können dies aber wahlweise tun). Diese Informationen können zur Mapping-Laufzeit jederzeit definiert oder außer Kraft gesetzt werden. Nähere Informationen dazu finden Sie unter Anmeldeinformationen 1353.
- Verwendung einer *dynamischen Authentifizierung*, d.h. Bereitstellung von Benutzername und Passwort als Input-Parameter für das Mapping. Bei dieser Methode müssen Sie die Parameterwerte jedes Mal, wenn das Mapping ausgeführt wird, bereitstellen. Um in MapForce eine Vorschau auf das Mapping anzeigen zu können, können Sie den Benutzernamen und das Passwort optional als Ausführungswerte zur "Design-Zeit" speichern, doch werden diese im Klartext gespeichert. Beachten Sie, dass OAuth-Anmeldeinformationen nicht als Input-Parameter für ein Mapping bereitgestellt werden können. Dies ist nur für Benutzernamen und Passwort möglich. Nähere Informationen dazu finden Sie unter Dynamische Authentifizierung 1366.
- (Herkömmliche Methode) Definieren Sie den Benutzernamen und das Passwort in MapForce nicht als benanntes Anmeldeinformationsobjekt, sondern als Standalone-Benutzernamen und Passwort. Beim Speichern des Mappings werden Benutzername und Passwort in verschlüsselter Form als Teil der Mapping-Design-Datei (*.mfd) gespeichert. Diese Methode wird aus Gründen der Rückwärtskompatibilität von mit früheren Versionen von MapForce erstellten Mappings unterstützt. Nähere Informationen dazu finden Sie unter Speicherung von Benutzername und Passwort 1369.

In den folgenden Abschnitten wird beschrieben, wie Sie ein Mapping für die einzelnen oben erwähnten Szenarien konfigurieren.

8.4.1 Anmeldeinformationen

Mit Hilfe von Anmeldeinformationsobjekten können Authentifizierungsdaten (wie Benutzernamen, Passwörter und OAuth-Authentifzierungsinformationen) zwischen verschiedenen Mapping-Ausführungsumgebungen auf sichere Art portabel gemacht werden. Anmeldeinformationen eignen sich für die Verwendung in Mappings, für

die eine einfache HTTP-Authentifizierung oder eine OAuth 2.0-Authentifizierung erforderlich ist. Anmeldeinformationen können sowohl in MapForce als auch FlowForce Server definiert werden. Wenn die Anmeldeinformationen in MapForce definiert wurden, können diese, ähnlich wie Mappings, optional auch auf FlowForce Server bereitgestellt werden.

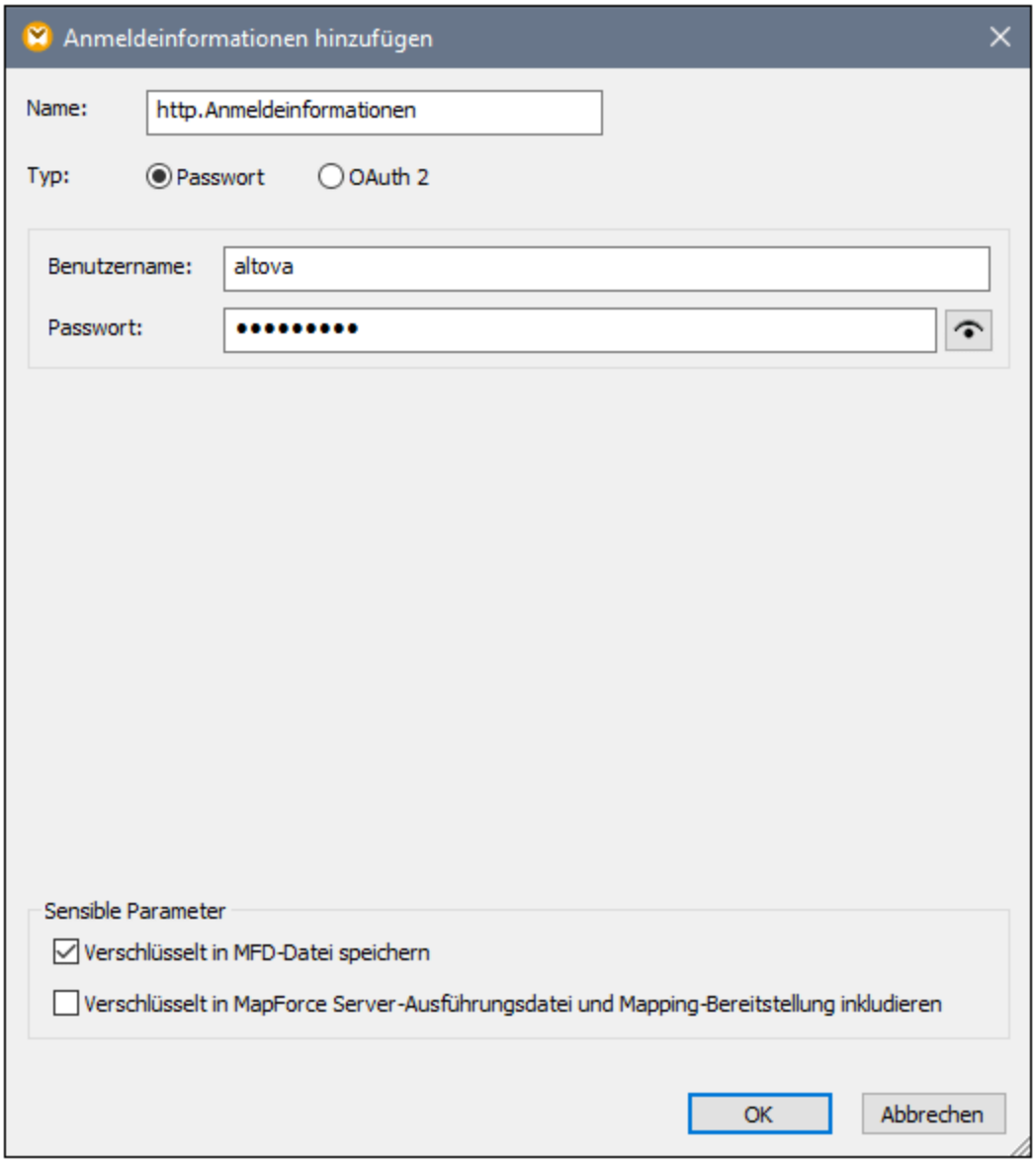
Ein Anmeldeinformationsobjekt muss einen Namen und einen Typ haben - alle anderen Informationen sind optional. Der Name ist nur eine Information für den Ausführungsprozessor (MapForce oder MapForce Server), dass für die Ausführung des Mappings Anmeldeinformationen erforderlich sind. Sie können wählen, ob die mit den Anmeldeinformationen verknüpften sensiblen Daten verschlüsselt innerhalb des Mappings gespeichert werden sollen oder ob sie erst zur Mapping-Laufzeit bereitgestellt werden sollen. Falls die sensiblen Daten innerhalb des Mappings gespeichert wurden, können Sie diese dennoch bei Ausführung des Mappings in einer Server-Umgebung außer Kraft setzen.

Es können entweder Anmeldeinformationen vom Typ **Passwort** oder vom Typ **OAuth 2.0** verwendet werden.

So erstellen Sie Anmeldeinformationen:

1. Klicken sie mit der rechten Maustaste auf einen leeren Bereich im Mapping und wählen Sie im Kontextmenü den Befehl **Anmeldeinformationen-Manager öffnen**. Daraufhin wird im oberen Bereich das Mapping-Fensters das Raster des Anmeldeinformationen-Managers[1357] geöffnet.
2. Klicken Sie auf **Hinzufügen.** Daraufhin wird ein Dialogfeld angezeigt, in das Sie die Anmeldeinformationen eingeben können. (Dassselbe Dialogfeld wird angezeigt, wenn Sie die HTTP-Sicherheitseinstellungen für eine Webservice-Komponente erstellen oder bearbeiten.)
3. Geben Sie einen Namen für die Anmeldeinformationen ein.
4. Wählen Sie einen Anmeldeinformationstyp aus (**Passwort** oder **OAuth 2.0**).

© 2016-2022 Altova GmbH



5. Um das Mapping zur Design-Zeit zu authentifizieren, füllen Sie die folgenden Felder aus:
   - Füllen Sie im Fall von "Passwort"-Anmeldeinformationen den für die Authentifizierung beim Webservice erforderlichen Benutzernamen und das Passwort aus. Bestätigen Sie das Passwort, wenn Sie dazu aufgefordert werden.
   - Nähere Informationen zu **OAuth 2.0**-Anmeldeinformationen finden Sie unter OAuth 2.0-Anmeldeinformationen[1358].

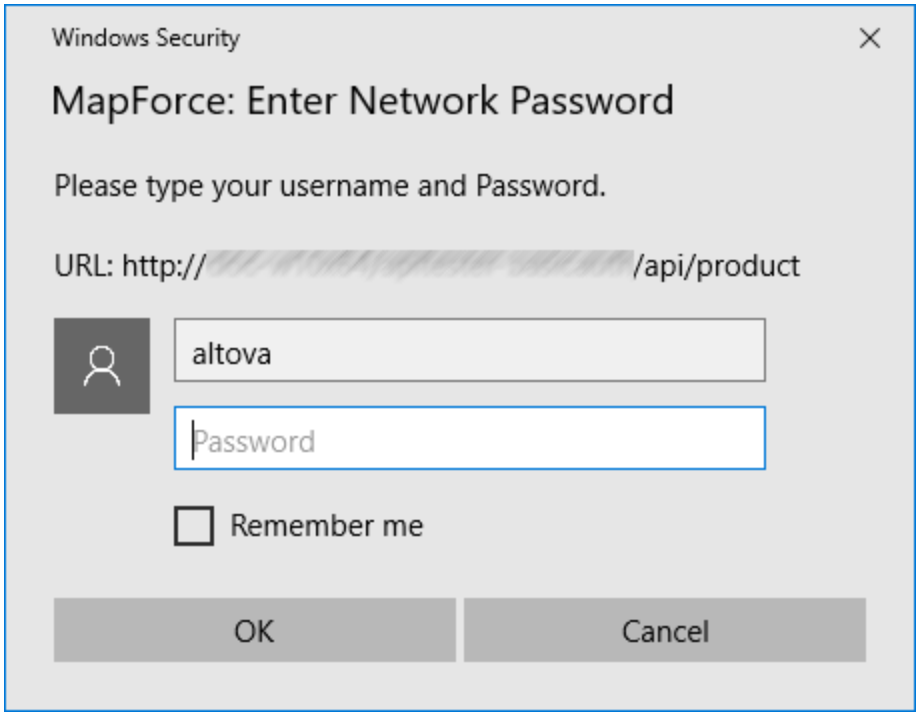
Sie können die sensiblen Informationen optional im Mapping-Design speichern und für andere Umgebungen in verschlüsselter Form bereitstellen oder diese gar nicht speichern. Wenn Sie sensible Daten gespeichert haben, können Sie diese nachdem Sie das Mapping geschlossen haben, dennoch anzeigen, in dem Sie auf die Schaltfläche **Anzeigen** klicken.

Im Fall von Passwort-Anmeldeinformationen handelt es sich beim Feld mit den sensiblen Daten um **Passwort**. Im Fall von OAuth 2-Anmeldeinformationen handelt es sich bei den Feldern mit sensiblen Daten um **Client Secret**, **Access Token** und **Refresh Token**.

Mit Hilfe der folgenden Kontrollkästchen können Sie festlegen, wie beim Speichern mit sensiblen Daten verfahren werden soll.

Verschlüsselt in MFD-Datei speichern	Aktivieren Sie dieses Kontrollkästchen, um die sensiblen Daten in verschlüsselter Form in der Mapping-Design-Datei (.mfd) zu speichern.
Verschlüsselt in MapForce Server-Ausführungsdatei und Mapping-Bereitstellung inkludieren	Aktivieren Sie dieses Kontrollkästchen, wenn die sensiblen Daten in die MapForce Server-Ausführungsdatei (.mfx) inkludiert werden sollen, wenn Sie eine solche Datei, wie unter Kompilieren von Mappings zu MapForce Server-Ausführungsdateien[1442] beschrieben, kompilieren. Wenn dieses Kontrollkästchen aktiviert wird, werden die sensiblen Daten auch in das auf FlowForce Server bereitgestellte Paket inkludiert, siehe Bereitstellen von Mappings auf FlowForce Server[1445].

Die Felder mit den sensiblen Daten werden in reversibel verschlüsselter Form gespeichert, da diese zum Zeitpunkt der Ausführung des Mappings entschlüsselt und verwendet werden müssen. Wenn Sie sensible Daten lieber nicht speichern möchten, deaktivieren Sie die oben genannten Kontrollkästchen. In diesem Fall werden Sie aufgefordert, das Passwort einzugeben, Wenn Sie versuchen, das Mapping auszuführen, z.B:

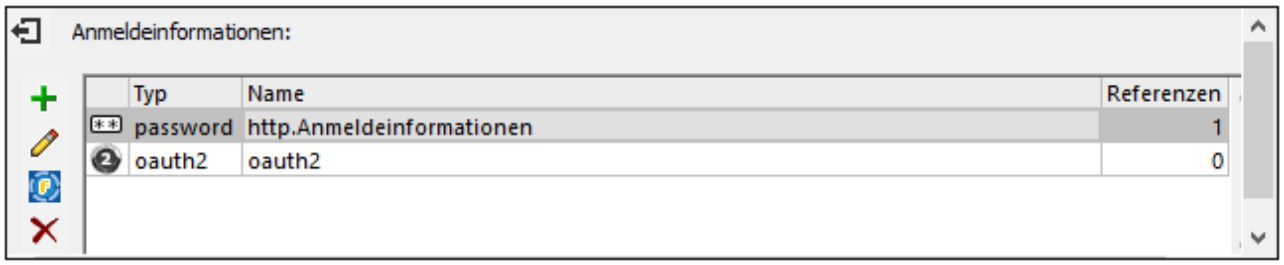


Serverseitig können Sie das Passwort zum Zeitpunkt der Ausführung des Mappings dynamisch bereitstellen. Nähere Informationen dazu finden Sie unter Anmeldeinformationen in MapForce Server[1361] und Anmeldeinformationen in FlowForce Server[1362].

© 2016-2022 Altova GmbH

8.4.1.1 Anmeldeinformationen-Manager

Der Anmeldeinformationen-Manager ist die grafische Benutzeroberfläche, über die Sie Anmeldeinformationen in jedem Mapping verwalten können. Wenn ein Mapping Anmeldeinformationen enthält, werden diese in einem Raster im oberen Bereich des Mapping-Fensters angezeigt. Im Raster werden für jedes Anmeldeinformationsobjekt der Typ (**OAuth** oder **Passwort**), der Name und die Anzahl der Referenzen zu diesen Anmeldeinformationen im aktuellen Mapping angezeigt.



Anmeldeinformationen-Manager

Um den **Anmeldeinformationen-Manager** auszublenden, klicken Sie auf **Schließen** oder drücken Sie die **Esc-Taste.**

Um den **Anmeldeinformationen-Manager** anzuzeigen, klicken Sie mit der rechten Maustaste auf ein Mapping und wählen Sie im Kontextmenü den Befehl **Anmeldeinformationen-Manager öffnen**.

Funktionalitäten des **Anmeldeinformationen-Managers**:

Symbol	Beschreibung
+	Neues Anmeldeinformationsobjekt erstellen, siehe Anmeldeinformationen 1353.
✎	Vorhandenes Anmeldeinformationsobjekt bearbeiten.
	Anmeldeinformationsobjekt auf FlowForce Server bereitstellen, siehe Bereitstellen von Anmeldeinformationen auf FlowForce Server 1364.
✕	Anmeldeinformationsobjekt aus dem Mapping löschen.

Sie können Anmeldeinformationseinträge auch durch Kopieren einfügen, indem Sie mit der rechten Maustaste auf einen Eintrag klicken und den entsprechenden Befehl aus dem Kontextmenü auswählen oder die Standard-Windows-Tastenkürzel (**Strg+X**, **Strg+C** bzw. **Strg+V** für **Ausschneiden**, **Kopieren** und **Einfügen**) verwenden. Um herauszufinden, ob Anmeldeinformationen von Komponenten im Mapping referenziert werden, klicken Sie mit der rechten Maustaste in den Anmeldeinformationen-Manager und wählen Sie im Kontextmenü den Befehl **Alle referenzierenden Komponenten suchen**. Das Ergebnis wird im Fenster "Meldungen" angezeigt.

8.4.1.2 Anmeldeinformationen in UDFs

Mappings, die benutzerdefinierte Funktionen (UDFs) enthalten, können in andere Mappings importiert werden, siehe Aufrufen und Importieren von benutzerdefinierten Funktionen 971. Wenn das importierte Mapping Anmeldeinformationen enthält, werden diese im Anmeldeinformationen-Manager als importierte Anmeldeinformationen angezeigt. Importierte Anmeldeinformationen werden mit gelbem Hintergrund angezeigt.

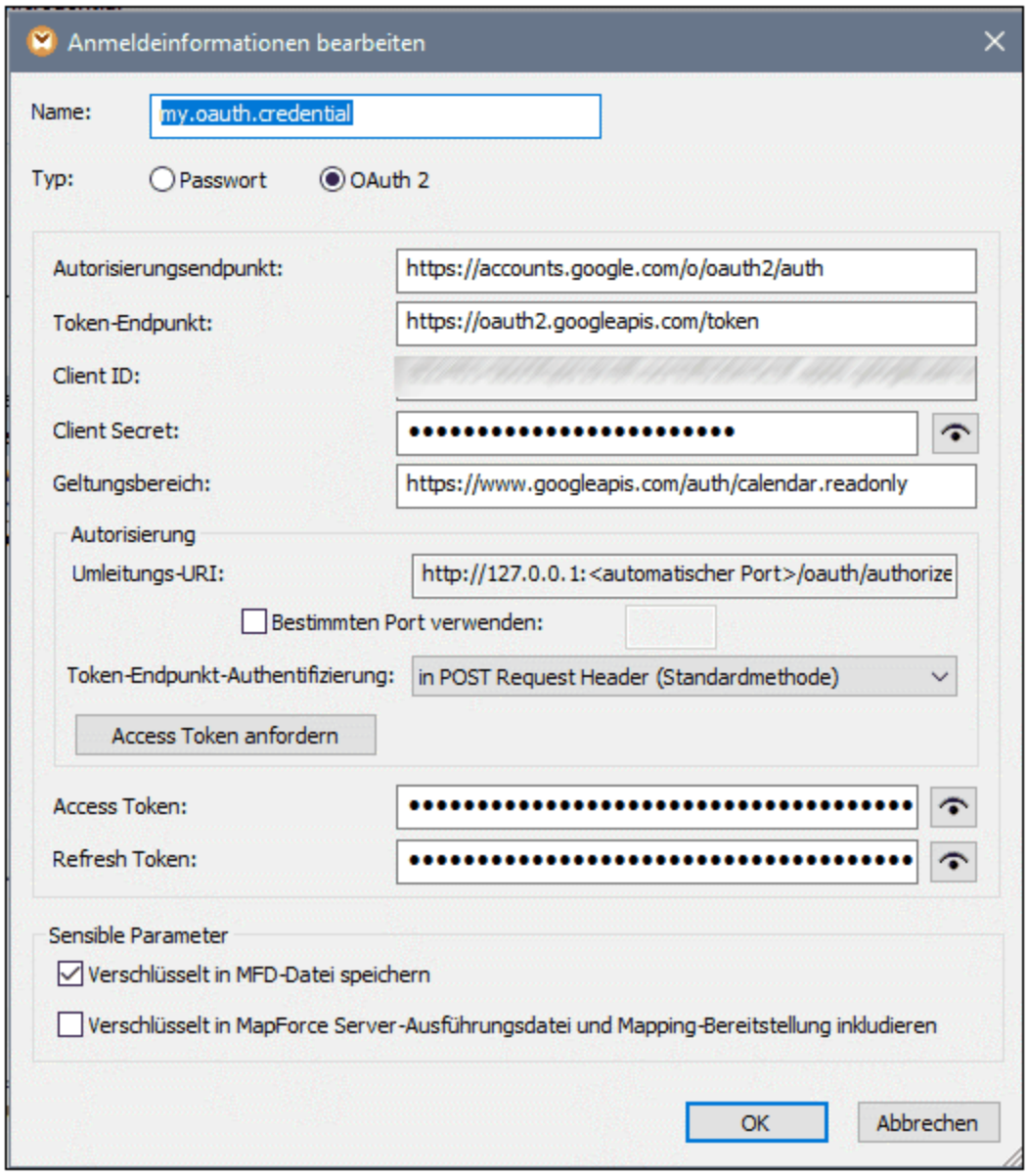
Standardmäßig werden importierte Anmeldeinformationen nicht mit dem Hauptmapping gespeichert. Sie können optional jedoch eine lokale Kopie erstellen und diese im Hauptmapping speichern. Wenn Sie versuchen, importierte Anmeldefeinformationen vom Hauptmapping aus zu bearbeiten, wird eine entsprechende Warnung angezeigt.

Anmerkung: Wenn das importierte Mapping und das Hauptmapping Anmeldeinformationen mit demselben Namen enthalten, kann es zu einem Konflikt kommen. In diesem Fall haben die Anmeldeinformationen aus dem Hauptmapping Vorrang. Um Konflikte zu vermeiden, stellen Sie sicher, dass die Namen von Anmeldeinformationen im Geltungsbereich eines Mappings und all seiner importierten UDFs eindeutig sind.

8.4.1.3 OAuth 2.0-Anmeldeinformationen

Um in MapForce ein Mapping ausführen zu können, für das eine OAuth 2.0-Authentifizierung erforderlich ist, müssen Sie über den Anmeldeinformationen-Manager 1357 Anmeldeinformationen vom Typ **OAuth 2.0** erstellen, die OAuth 2.0-Informationen ausfüllen und die unten beschriebene manuelle Autorisierung durchführen.

© 2016-2022 Altova GmbH



Unten sehen Sie eine Liste der mit einem OAuth 2.0-Anmeldeinformationsobjekt verknüpften Felder. Um diese Werte zu erhalten, müssen Sie sich zuerst beim Webservice-Anbieter (z.B. Google API Console, FaceBook API, BitBucket API, usw.) registrieren.

Autorisierungsendpunkt	Definiert die URI, von der aus MapForce Autorisierungsflüsse initiiert. Sie erhalten diesen Wert, nachdem Sie sich beim Webservice-Anbieter registriert haben.
Token-Endpunkt	Definiert die URI, von der aus MapForce Token-Flüsse initiiert. Sie erhalten diesen Wert, nachdem Sie sich beim Webservice-Anbieter registriert haben.
Client ID	Die ID der Client-Applikation (in diesem Fall MapForce). Sie erhalten diesen

	Wert, nachdem Sie sich beim Webservice-Anbieter registriert haben.
Client Secret	Das mit der Client-Applikation verknüpfte Secret. Sie erhalten diesen Wert, nachdem Sie sich beim Webservice-Anbieter registriert haben.
Geltungsbereich	Der Geltungsbereich der Client-Applikation, falls dies vom Anbieter verlangt wird. Sie erhalten diesen Wert, nachdem Sie sich beim Webservice-Anbieter registriert haben.
Umleitungs-URI	Der OAuth 2.0 Authorization Server benötigt eine Callback-URL, an die er Anworten an MapForce senden soll. MapForce designiert die URL und den Port für die Kommunikation mit dem Authorization Server automatisch. Wenn der Authorization Server verlangt, dass die Kommunikation über einen spezifischen Port erfolgt, wählen Sie die Option **Spezifischer Port** und geben Sie den gewünschten Port-Namen ein.
Token-Endpunkt-Authentifizierung	Für die meisten OAuth 2.0 Authorization Server müssen die Autorisierungsinformationen im **POST Request Header** übermittelt werden. Dies ist auch der Wert, der in der Dropdown-Liste standardmäßig ausgewählt ist. Einige OAuth 2.0 Authorization Server akzeptieren die Authentifizierungsinformationen nur im Body des POST Request. Aktivieren Sie für diese Authorization Server den Wert **im POST Request Body** aus der Dropdown-Liste.
Access Token	Dies ist das vom Authorization Server zurückgesendete Access Token. Das Mapping kann nur dann erfolgreich ausgeführt werden, wenn der Resource Server das Access Token als korrekt und gültig erkennt.
Refresh Token	Dies ist das vom Authorization Server zurückgesendete Refresh Token. Es wird benötigt, wenn das **Access Token** (siehe unten) abläuft. In seltenen Fällen, wenn das Access Token nie abläuft, ist dieses Token nicht erforderlich.

Nachdem Sie die obigen Felder ausgefüllt haben, klicken Sie auf **Access Token anfordern,** um das für die Ausführung des Mappings erforderliche Access Token abzurufen. Der genaue Ablauf des Autorisierungsverfahren hängt vom Webservice-Anbieter ab. Normalerweise müssen Sie in einem Browser-Fenster manuell bestätigen, dass Sie dem Webservice erlauben, Ihre Identität festzustellen (z.B. Ihrem Google-Konto, wenn im Mapping eine Google API aufgerufen wird). Nach Abschluss der Autorisierung wird in MapForce eine Bestätigungsmeldung angezeigt, dass das Access Token vom Server abgerufen wurde. Beachten Sie, dass das Access Token nur dann gespeichert wird, wenn Sie das Kontrollkästchen **Verschlüsselt in MFD-Datei speichern,** wie unter Anmeldeinformationen 1353 beschrieben, aktiviert haben. Falls Sie dieses Kontrollkästchen nicht aktiviert haben, müssen Sie die Autorisierung bei jeder Ausführung des Mappings manuell durchführen.

Das Access Token läuft unter Umständen nach einiger Zeit ab. Die Ablaufdauer für ein solches Token hängt normalerweise vom Webservice-Anbieter ab. Es kann auch vorkommen, dass das Access Token vom Webservice-Anbieter explizit widerrufen wurde. In diesem Fall werden im Fenster "Meldungen" Autorisierungsfehler angezeigt, wenn Sie versuchen, eine Vorschau auf das Mapping anzuzeigen. Klicken Sie daraufhin auf **Access Token anfordern** und führen Sie die Autorisierung erneut durch, bevor Sie das Mapping ausführen können.

 © 2016-2022 Altova GmbH

8.4.1.4 Anmeldeinformationen in MapForce Server

Nachdem Sie das Mapping zu einer MapForce Server-Ausführungsdatei (.mfx) kompiliert haben, führt MapForce Server die .mfx-Datei gemäß der von Ihnen zum Zeitpunkt des Designs des Mapping getroffenen Auswahl (siehe Erstellen von Anmeldeinformationen) 1353 aus.

Wenn Sie bei der Erstellung der Anmeldeinformationen in MapForce das Kontrollkästchen **Verschlüsselt in MapForce Server-Ausführungsdatei und Mapping-Bereitstellung inkludieren** aktiviert haben, verwendet MapForce Server zur Mapping-Laufzeit die in der .mfx-Datei gespeicherten Anmeldeinformationen. Das bedeutet, Sie können das Mapping mit einem Befehl wie dem folgenden ausführen:

```
<exec> run mapping.mfx
```

Hierbei steht **<exec>** für den Pfad zur ausführbaren MapForce Server-Datei. Dieser Pfad kann entweder ein absoluter sein oder Sie können nur den Namen der ausführbaren Datei eingeben, wenn das aktuelle Verzeichnis dasselbe wie das der ausführbaren Datei ist.

Wenn Sie in MapForce nur den Namen der Anmeldeinformationen (ohne Benutzernamen und Passwort) eingegeben haben, müssen Sie diese Informationen mit Hilfe der Befehlszeilenoption **`--credential`** des Befehls `run` zur Mapping-Laufzeit explizit bereitstellen. Auf diese Art können Sie z.B. in der Produktion andere Anmeldeinformationen als zum Zeitpunkt des Mapping-Designs verwenden. Die Option `--credential` hat die Form `--credential=KEY:VALUE`., wobei

- **`KEY`** der Name der in MapForce definierten Anmeldeinformationen ist.
- **`VALUE`** eine Anmeldeinformationseigenschaft oder eine durch das Zeichen (&) getrennte Liste von Eigenschaften ist. Bei Anmeldeinformationen vom Typ "Passwort" sind die möglichen Eigenschaften **`username`** und **`password`**. Bei Anmeldeinformationen vom Typ OAuth 2.0 ist die einzige unterstützte Eigenschaft die Eigenschaft **`oauth:token`**.
- Die eigentlichen Eigenschaftswerte werden genau wie Abfrageparameter in einer URL mit Hilfe des Zeichens "=" bereitgestellt.

Zum Beispiel:

```
<exec> run mapping.mfx --credential="mycredential:username=admin&password=4xJ38dnx7"
```

Der Wert der Option `--credential` im Codefragment oben wurde in Anführungszeichen gesetzt, damit der Wert als Literal behandelt wird, da der Benutzername und das Passwort durch ein Kaufmännisches Und-Zeichen voneinander getrennt sind.

Wenn mehrere benannte Anmeldeinformationsobjekte erforderlich sind, damit Ihr Mapping ausgeführt werden kann, können Sie die Option **`--credential`** mehrmals definieren.

> Anmeldeinformationen, die als Befehlszeilenoptionen bereitgestellt werden, haben Vorrang vor gespeicherten Anmeldeinformationen.

Wenn Sie die Option **Verschlüsselt in MapForce Server-Ausführungsdatei und Mapping-Bereitstellung inkludieren** nicht aktiviert haben, fehlen die Felder mit den sensiblen Daten, d.h. Sie müssen in der Befehlszeile das Passwort angeben, während Sie das Anmeldeinformationsobjekt anhand seines Namens

referenzieren, z.B:

```
<exec> run mapping.mfx --credential=mycredential:password=4xJ38dnx7
```

Die folgenden Felder gelten als Felder, die sensible Daten enthalten:

- **Passwort** (bei Anmeldeinformationen vom Typ "Passwort")
- **Client Secret**, **Access Token** und **Refresh Token** (bei Anmeldeinformationen vom Typ "OAuth 2.0")

Bei Mappings, für die eine OAuth 2.0-Autorisierung erforderlich ist, wird in der MapForce Server-Befehlszeile zur Mapping-Laufzeit ein OAuth 2.0 Access Token als Input akzeptiert. Beachten Sie, dass es in der MapForce Server-Befehlszeile keine interaktive Benutzeroberfläche gibt. Daher müssen Sie das OAuth 2.0 Access Token in diesem Fall auf externem Weg (z.B. durch Anforderung von MapForce aus) abrufen. Dies ist jedoch nicht notwendig, wenn MapForce Server unter FlowForce Server-Verwaltung ausgeführt wird, da FlowForce Server zur Laufzeit selbst ein neues OAuth 2.0 Access Token anfordern kann.

Ein Mapping kann solange mit gespeicherten Anmeldeinformationen über die Befehlszeile ausgeführt werden, als das gespeicherte OAuth 2.0 Token noch nicht abgelaufen ist oder vom Webservice-Anbieter widerrufen wurde. Andernfalls müssen Sie über die Option **`--credential`** ein neues (auf externem Weg abgerufenes) OAuth 2.0 Access Token angeben, z.B:

```
<exec> run mapping.mfx --
credential=my_oauth_credential:oauth:token=jdsaflkajlkewsaiurthczv904215-jhd
```

wobei:

- **`my_oauth_credential`** der Name der mit MapForce erstellten OAuth 2.0-Anmeldeinformationen ist.
- **`oauth:token`** die Methode ist, um MapForce Server mitzuteilen, dass zur Laufzeit ein neues OAuth 2.0 Access Token bereitgestellt wird.

MapForce Server API

Die MapForce Server API bietet Methoden, um Anmeldeinformationen zu erstellen, Eigenschaften zu Anmeldeinformationen hinzuzufügen und Anmeldeinformationen zu schließen, nachdem Sie diese fertig deklariert haben. Nähere Informationen dazu finden Sie in der Dokumentation zu MapForce Server (https://www.altova.com/de/documentation)

8.4.1.5 Anmeldeinformationen in FlowForce Server

Wenn Sie ein Mapping, das Anmeldeinformationen enthält, von MapForce aus auf FlowForce Server bereitstellen, werden auch die Anmeldeinformationen auf dem Server bereitgestellt. Die bereitgestellten Informationen enthalten nur die Felder, die Sie bei der Erstellung des Anmeldeinformationsdatensatzes ausgefüllt haben. Dabei kann es sich z.B. um ein leeres Anmeldeinformationsobjekt handeln (wenn Sie nur den Namen der Anmeldeinformationen speichern) oder ein Anmeldeinformationsobjekt, das sowohl den Benutzernamen als auch das Passwort enthält.

Sie können Anmeldeinformationsobjekte von MapForce aus auch separat vom Hauptmapping als Standalone-

© 2016-2022 Altova GmbH

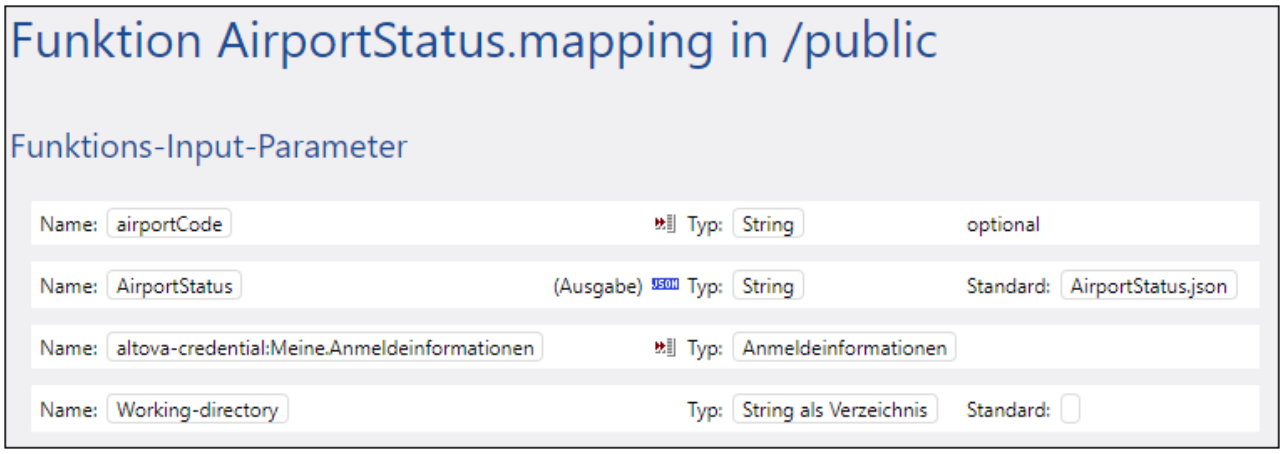
Objekte auf FlowForce Server bereitstellen, siehe Bereitstellen von Anmeldeinformationen auf FlowForce Server[1364].

Die folgenden Daten gelten als sensible Daten:

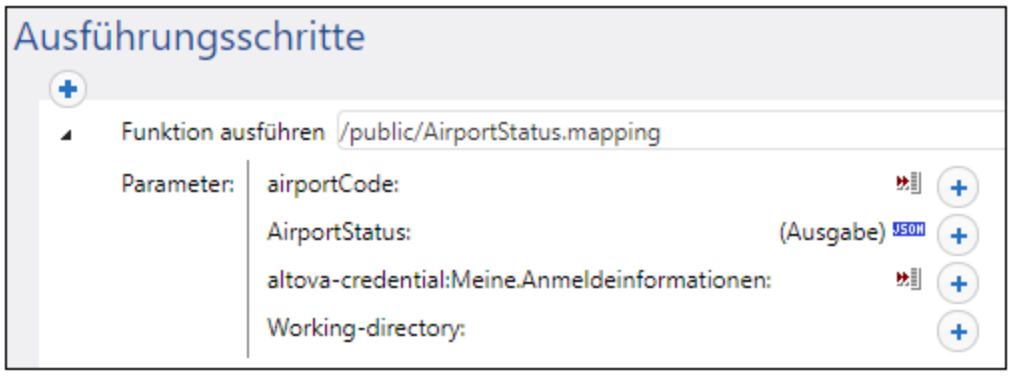
- **Passwort** (bei Anmeldeinformationen vom Typ "Passwort")
- **Client Secret**, **Access Token** und **Refresh Token** (bei Anmeldeinformationen vom Typ "OAuth 2.0")

Die sensiblen Daten werden nur dann bereitgestellt, wenn Sie, wie unter Erstellen von Anmeldeinformationen[1353] beschrieben, das Kontrollkästchen **Verschlüsselt in MapForce Server-Ausführungsdatei und Mapping-Bereitstellung inkludieren** aktiviert haben. Dies gilt sowohl für die Bereitstellung des Mappings als auch für die Bereitstellung der Standalone-Anmeldeinformationen.

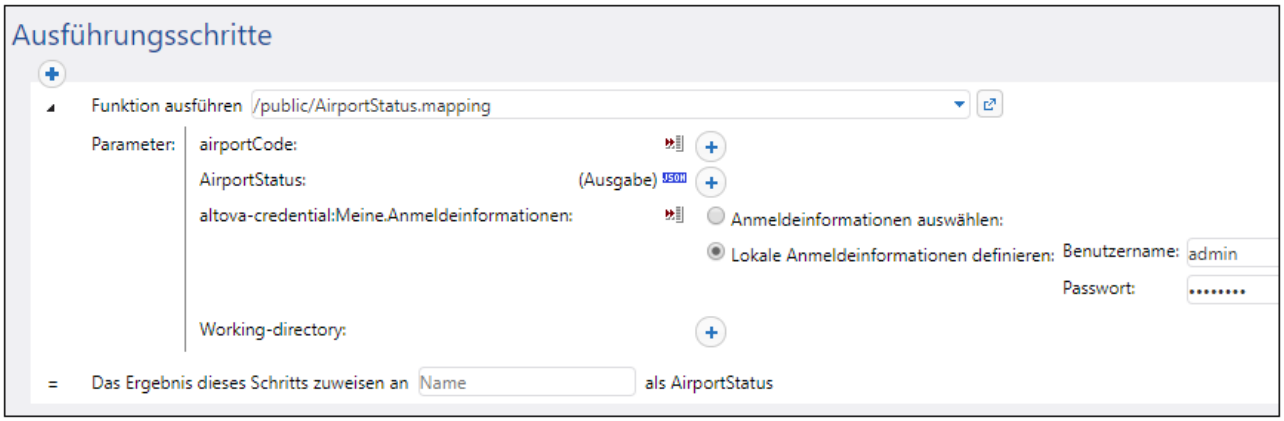
Sie sehen in FlowForce Server, ob für eine Mapping-Funktion Anmeldeinformationen benötigt werden, indem Sie die Seite der entsprechenden Mapping-Funktion öffnen, z.B:



Wenn Sie bei der Erstellung der Anmeldeinformationen die Option **Verschlüsselt in MapForce Server-Ausführungsdatei und Mapping-Bereitstellung inkludieren** aktiviert haben, werden die zusammen mit dem Mapping bereitgestellten Anmeldeinformationen im Auftrag verwendet. In diesem Fall müssen Sie diese nicht über die Auftragskonfigurationsseite definieren. So wird z.B. im folgenden Ausführungsschritt die Mapping-Funktion mit den gespeicherten Anmeldeinformationen, falls diese vorhanden sind, ausgeführt (beachten Sie bitte, dass der Parameter "Meine.Anmeldeinformationen" nicht erweitert wurde):



Sie können die gespeicherten Anmeldeinformationen jedoch durch ein anderes direkt in FlowForce Server definiertes Anmeldeinformationsobjekt oder andere lokale Anmeldeinformationen ersetzen und dadurch außer Kraft setzen. Klicken Sie dazu auf die Schaltfläche "+" und wählen Sie entweder ein bereits in FlowForce Server vorhandenes Anmeldeinformationsobjekt aus oder geben Sie den Benutzernamen und das Passwort direkt ein, z.B:



Anmeldeinformationen, die als Parameter für den Ausführungsschritt bereitgestellt werden, haben Vorrang vor innerhalb der Mapping-Funktion gespeicherten Anmeldeinformationen.

Wenn Sie die Option **Verschlüsselt in MapForce Server-Ausführungsdatei und Mapping-Bereitstellung inkludieren** beim Erstellen der Anmeldeinformationen in MapForce nicht aktiviert haben, müssen die Anmeldeinformationen unbedingt als Parameter für den Ausführungsschritt bereitgestellt werden, da die Ausführung des Auftrags sonst fehlschlägt.

8.4.1.6 Bereitstellen von Anmeldeinformationen auf FlowForce Server

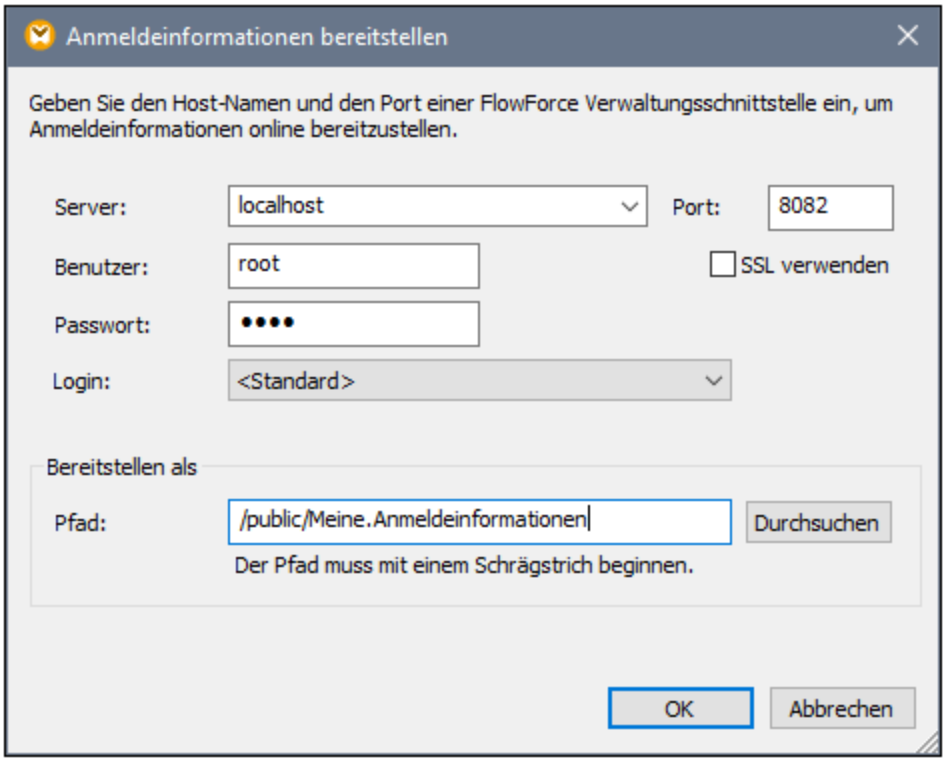
Anmeldeinformationen können sich häufig ändern. Dies bedeutet jedoch nicht, dass Sie das Mapping, jedes mal, wenn sich die Anmeldeinformationen ändern, ändern und erneut bereitstellen müssen. Statt dessen können Sie in diesem Fall jederzeit einfach ein Anmeldeinformationsobjekt auf FlowForce Server bereitstellen, ohne Änderungen am Mapping vorzunehmen. Auf diese Art können Sie ein Anmeldeinformationsobjekt in mehreren FlowForce Server-Aufträgen wiederverwenden.

© 2016-2022 Altova GmbH

Alternativ dazu können Sie ein Anmeldeinformationsobjekt direkt in FlowForce Server erstellen und es genau wie andere FlowForce-Anmeldeinformationen als Parameter für den Ausführungsschritt bereitstellen.

So stellen Sie MapForce-Anmeldeinformationen in FlowForce Server bereit:

1. Öffnen Sie das Mapping, das die Anmeldeinformationen enthält.
2. Klicken sie mit der rechten Maustaste auf einen leeren Bereich im Mapping und wählen Sie im Kontextmenü den Befehl **Anmeldeinformationen-Manager öffnen**.
3. Wählen Sie im Raster des Anmeldeinformationen-Managers einen Datensatz aus und klicken Sie anschließend auf die Schaltfläche **Anmeldeinformationen auf FlowForce Server bereitstellen**. Daraufhin wird ein Dialogfeld geöffnet, in das Sie die FlowForce Server-Verbindungsinformationen und auch den Zielpfad zum Server eintragen können:



4. Klicken Sie auf **OK** und sehen Sie nach, welche Meldungen im Fenster "Meldungen" angezeigt werden.

Die folgenden Optionen für die Bereitstellung stehen zur Verfügung.

Einstellung	Beschreibung
Server, Port SSL verwenden	Geben Sie den Server Host-Namen (oder die IP-Adresse) und den Port von FlowForce Server ein. Wenn FlowForce Server auf demselben Rechner und unter dem Standard-Port ausgeführt wird, dann wäre dies z.B. **localhost** und **8082.** Wenn Sie sich nicht sicher sind, loggen Sie sich auf

	der FlowForce Server Web-Verwaltungsoberfläche ein und überprüfen Sie, welche IP-Adresse und welcher Port in der Adressleiste des Web Browsers angezeigt werden. Stellen Sie bei Verbindungsproblemen sicher, dass der Rechner, auf dem FlowForce Server ausgeführt wird, so konfiguriert ist, dass er eingehende Verbindungen an der dafür vorgesehenen Adresse und am dafür vorgesehenen Port zulässt. Um das Mapping über eine SSL-verschlüsselte Verbindung bereitzustellen, aktivieren Sie das Kontrollkästchen **SSL verwenden**. Es wird dabei vorausgesetzt, dass FlowForce Server bereits so konfiguriert ist, dass SSL-Verbindungen zugelassen werden. Nähere Informationen dazu finden Sie in der Dokumentation zu FlowForce Server (https://www.altova.com/de/documentation)..
Benutzer und Passwort	Welcher Benutzername und welches Passwort eingegeben werden müssen, hängt vom Wert in der Login-Dropdown-Liste ab (siehe nächste Option). Wenn die Login-Dropdown-Liste auf **<Standard>** oder **Direkt** gesetzt ist, geben Sie Ihren FlowForce Server Benutzernamen und Ihr Passwort dafür ein. Geben Sie andernfalls Ihren Domain-Benutzernamen und Ihr Domain-Passwort ein und wählen Sie den Domain-Namen aus der Login-Dropdown-Liste aus.
Login	Wenn in FlowForce Server die Directory Service-Integration aktiviert ist, wählen Sie den Domain-Namen aus dieser Dropdown-Liste aus und geben Sie Ihre Domain-Anmeldeinformationen in die Felder "Benutzername" und "Passwort" ein (siehe vorherige Option).
Pfad	Klicken Sie auf **Durchsuchen** und wählen Sie den Pfad, unter dem die Anmeldeinformationen in der FlowForce Server Containerhierarchie gespeichert werden sollen. Standardmäßig ist als Pfad der Container **/public** von FlowForce Server eingestellt. Sie können über dieses Dialogfeld auch einen neuen Container erstellen oder vorhandene Container und Anmeldeinformationen löschen, vorausgesetzt Sie haben die erforderlichen FlowForce Server-Berechtigungen und Rechte dazu.

8.4.2 Dynamische Authentifizierung

Bei der dynamischen Authentifizierung werden Benutzername und Passwort zur Laufzeit als Input-Parameter an das Mapping übergeben. Die dynamische Authentifizierung wird für Mappings unterstützt, die Webservices aufrufen, für die eine einfache HTTP-Authentifizierung erforderlich ist.

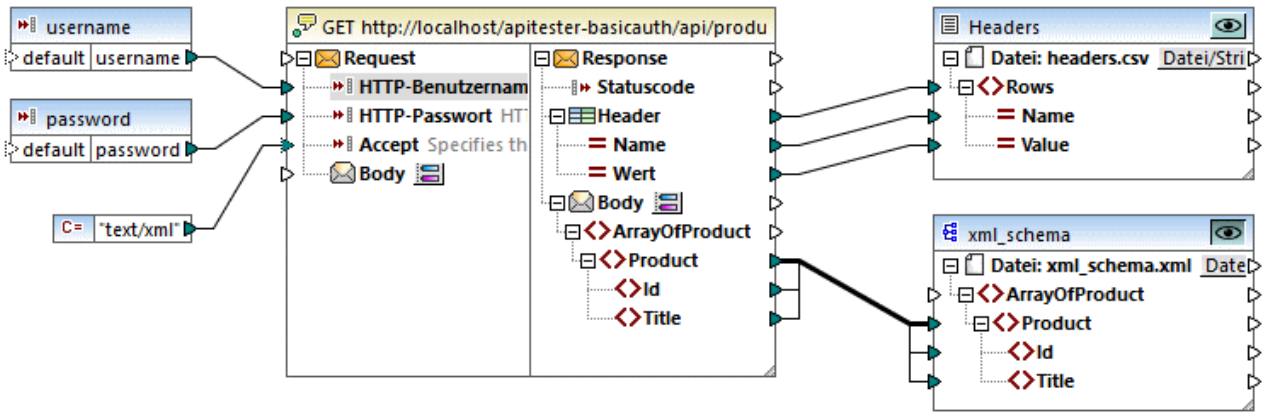
So richten Sie eine dynamische Authentifizierung ein:

1. Öffnen Sie ein Mapping, das einen Webservice-Aufruf enthält oder fügen Sie einen Webservice zu einem neuen Mapping hinzu. Nähere Informationen dazu finden Sie unter Hinzufügen eines Webservice-Aufrufs (REST) 612 und Hinzufügen eines (SOAP)-Webservice-Aufrufs 630.

 © 2016-2022 Altova GmbH

2. Doppelklicken Sie auf die Titelleiste der Webservice-Komponente, um das Einstellungsdialogfeld zu öffnen.
3. Klicken Sie neben "HTTP-Sicherheitseinstellungen" auf **Bearbeiten.**.
4. Wählen Sie die Option **Dynamische Authentifizierung** aus.
5. Klicken Sie auf **OK**.

Sobald die dynamische Authentifizierung aktiviert wurde, erhält die "Request"-Seite der Webservice-Komponente zwei neue Input-Konnektoren: **HTTP-Benutzername und HTTP-Passwort**. Um die tatsächlichen Werte bereitzustellen, fügen Sie mit dem Menübefehl **Einfügen | Input-Komponente einfügen** zwei neue Input-Komponenten zum Mapping hinzu und verbinden Sie diese mit dem **HTTP-Benutzernamen** bzw. dem **HTTP-Passwort**.



Um in MapForce eine Vorschau auf das Mapping anzuzeigen, können Sie den Benutzernamen und das Passwort optional als "Design-time-Ausführungswerte" bereitstellen. Doppelklicken Sie dazu auf die Titelleiste der jeweiligen Input-Komponente und geben Sie den Benutzernamen bzw. das Passwort in das Textfeld **Wert** ein. Nähere Informationen dazu finden Sie unter Bereitstellen von Parametern für das Mapping 867.

> Benutzername und Passwort werden genau wie jeder andere Design-Time-Ausführungswert behandelt. Daher werden sie in der Mapping-Design-Datei in Form von Klartext gespeichert. Um dies zu verhindern, geben Sie gar keine Design-Time-Ausführungswerte ein oder verwenden Sie die Methode "Anmeldeinformationen" 1353.

8.4.2.1 Dynamische Authentifizierung in MapForce Server

Wenn Sie das Mapping für eine dynamische Authentifizierung konfiguriert haben, müssen der entsprechende Benutzername und das entsprechende Passwort zur Mapping-Laufzeit genau wie bei der Bereitstellung anderer Parameterarten als Parameter bereitgestellt werden. Wenn Sie z.B. MapForce Server über die Befehlszeile aufrufen, lautet die Syntax für ein Mapping wie das oben beschriebene folgendermaßen:

```
<exec> run mapping.mfx --p=username:admin --p=password:dj9JaVax
```

Wobei:

- **<exec>** der Pfad zur ausführbaren MapForce Server-Datei ist. Dabei kann es sich entweder um einen

absoluten Pfad handeln oder Sie können einfach nur den Namen der ausführbaren Datei angeben, wenn das aktuelle Verzeichnis dasselbe wie das der ausführbaren Datei ist.

- **username** und **password** die Namen der entsprechenden Input-Parameter im MapForce Mapping sind.

Bei Aufruf der MapForce Server API können Sie das Mapping durch Aufruf der Methode **AddParameter** authentifizieren, bevor Sie die Methode **Run** aufrufen. Der Code dafür in C#, könnte folgendermaßen aussehen:

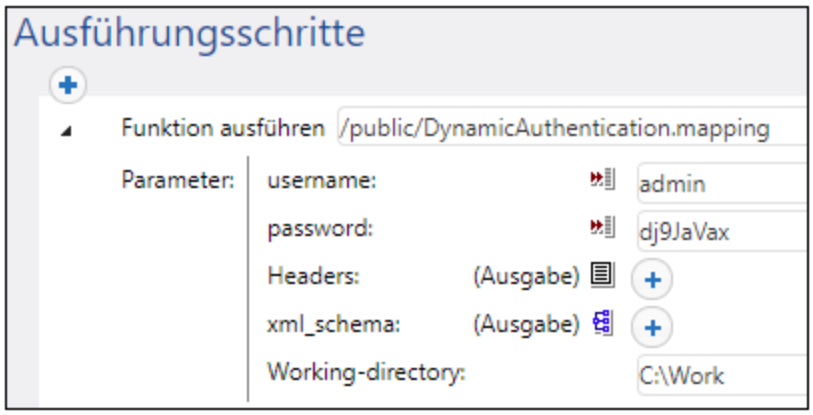
```
try
{
    Altova.MapForceServer.Server mfs = new Altova.MapForceServer.Server();
    mfs.AddParameter("username", "admin");
    mfs.AddParameter("password", "dj9JaVax");
    mfs.WorkingDirectory = "C:\\Work";
    if(mfs.Run("C:\\Work\\mapping.mfx"))
    {
      Console.WriteLine("Success");
    }
    else
    {
      Console.WriteLine(mfs.LastExecutionMessage);
    }
    Console.ReadLine();
}
catch(Exception ex)
{
  Console.WriteLine(ex);
}
```

Auch hier müssen **username** und **password** (das erste Argument für die Methode **AddParameter**) mit den Namen der entsprechenden Input-Parameter im MapForce-Mapping identisch sein.

8.4.2.2 Dynamische Authentifizierung in FlowForce Server

Wenn Sie ein Mapping, das eine dynamische Authentifizierung enthält, auf FlowForce Server bereitstellen, werden der Benutzername und das Passwort zu Input-Parametern für die Mapping-Funktion. In jedem FlowForce Server-Auftrag, in dem eine solche Mapping-Funktion aufgerufen wird, werden vor einer erfolgreichen Ausführung der Benutzername und das Passwort benötigt, z.B:

© 2016-2022 Altova GmbH



Im oben gezeigten Beispiel werden der Benutzername und das Passwort einfach in die entsprechenden Textfelder eingegeben. Sie können diese jedoch auch in Form von Input-Parametern für den Auftrag bereitstellen. Nähere Informationen dazu finden Sie in der FlowForce Server-Dokumentation (https://www.altova.com/de/documentation).

8.4.3 Speicherung von Benutzername und Passwort

Den Benutzernamen und das Passwort in verschlüsselter Form im Mapping zu speichern ist eine der Arten, Mappings mit Authentifizierung auszuführen. Bei dieser Methode kann das Mapping in jeder Ausführungsumgebung ganz einfach ausgeführt werden, da die sensiblen Daten in verschlüsselter Form als Teil des Mappings bzw. bei einer Server-Ausführung als Teil der Server-Ausführungsdatei gespeichert werden. Dies ist zwar weniger komplex, hat aber den Nachteil, dass das Mapping jedes mal, wenn sich der Benutzername und das Passwort ändern, aktualisiert und zu einer neuen Server-Ausführungsdatei (.mfx-Datei) kompiliert bzw. erneut auf FlowForce Server bereitgestellt werden muss. Außerdem können Benutzername und Passwort zur Mapping-Laufzeit nicht wie im Fall von Anmeldeinformationen[1353] geändert werden.

Diese Methode wird nur aus Gründen der Rückwärts-Kompatibilität mit früheren Versionen von MapForce unterstützt. Es wird empfohlen, stattdessen entweder Anmeldeinformationen[1353] oder eine Dynamische Authentifizierung[1366] zu verwenden.

So richten Sie eine Authentifizierung mit Hilfe eines gespeicherten Benutzernamens und Passworts ein:

1. Öffnen Sie ein Mapping, das einen Webservice-Aufruf enthält oder fügen Sie einen Webservice zu einem neuen Mapping hinzu.
2. Doppelklicken Sie auf die Titelleiste der Webservice-Komponente, um das Einstellungsdialogfeld zu öffnen.
3. Klicken Sie neben "HTTP-Sicherheitseinstellungen" auf **Bearbeiten.**.
4. Aktivieren Sie das Optionsfeld neben Benutzername und füllen Sie den zu speichernden Benutzernamen und das Passwort aus. Bestätigen Sie das Passwort, wenn Sie dazu aufgefordert werden.
5. Klicken Sie auf **OK**.

Sie können nun in MapForce eine Vorschau auf das Mapping anzeigen, ohne dass weitere Vorbedingungen erfüllt werden müssen. In MapForce Server kann das Mapping genauso wie jedes andere Mapping, für das keine Anmeldeinformationen erforderlich sind, ausgeführt werden. Bei Aufruf von MapForce Server über die Befehlszeile könnte die Syntax zur Ausführung des Mappings z.B. folgendermaßen lauten:

```
<exec> run mapping.mfx
```

Wenn Sie das Mapping auf FlowForce Server bereitstellen, um es als Mapping-Funktion auszuführen, gibt es in FlowForce Server keine spezielle Kennzeichnung, dass das Mappings gespeicherte Anmeldeinformationen enthält. Sie können einfach anhand der Mapping-Funktion einen Auftrag erstellen und diesen ohne spezielle Vorbedingungen ausführen.

© 2016-2022 Altova GmbH

8.5 Verarbeitung mehrerer Input- oder Output-Dateien

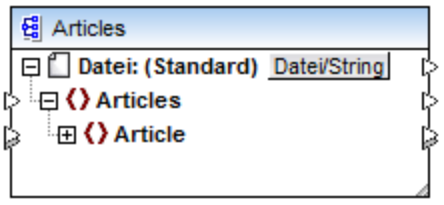
Sie können MapForce so konfigurieren, dass das Programm bei der Ausführung des Mappings mehrere Dateien (z.B. alle Dateien in einem Verzeichnis) verarbeitet. Mit Hilfe dieser Funktion können Sie die folgenden Aufgaben durchführen:

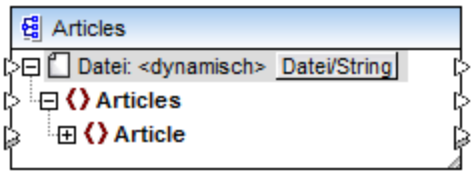
- Bereitstellen einer Liste von Input-Dateien, die vom Mapping verarbeitet werden sollen
- Generieren einer Liste von Dateien anstelle einer einzigen Ausgabedatei als Mapping-Ausgabe
- Generieren einer Mapping-Applikation, in der sowohl die Namen der Input- als auch die der Output-Dateien zur Laufzeit definiert werden
- Konvertieren einer Gruppe von Dateien in ein anderes Format
- Aufteilen einer großen Datei (oder Datenbank) in kleinere Teile
- Zusammenführen mehrerer Dateien in einer großen Datei (oder Laden dieser Dateien in einer Datenbank)

Sie können eine MapForce-Komponente so konfigurieren, dass mehrere Dateien auf eine der folgenden Arten verarbeitet werden:

- Bereitstellung des Pfads zur/zu den gewünschten Input- oder Output-Datei(en) mit Hilfe von Platzhalterzeichen anstelle eines festgelegten Dateinamens in den Komponenteneinstellungen (siehe Ändern der Komponenteneinstellungen 82). Sie können im Dialogfeld "Komponenteneinstellungen" die Platzhalterzeichen * und ? verwenden, sodass MapForce bei der Ausführung des Mappings den entsprechenden Pfad auflöst.
- Verbinden mit dem Root-Node einer Komponente einer Sequenz, die den Pfad dynamisch bereitstellt (z.B. das Ergebnis der Funktion `replace-fileext`). MapForce liest bei der Ausführung des Mappings alle Input-Dateien dynamisch bzw. generiert alle Output-Dateien dynamisch.

Je nachdem, welches Ergebnis Sie erzielen möchten, können Sie im selben Mapping entweder eine oder beide dieser Methoden verwenden. Es ist jedoch nicht sinnvoll, beide Methoden gleichzeitig für dieselbe Komponente zu verwenden. Um diese Methode für eine bestimmte Komponente auszuwählen, klicken Sie auf die Schaltfläche **Datei** (Datei) oder **Datei/String** (Datei/String) neben dem Root-Node einer Komponente. Über diese Schaltfläche können Sie die folgenden Einstellungen vornehmen:

Dateinamen aus Komponenteneinstellungen verwenden	Wenn die Komponente eine oder mehrere Instanzdateien verarbeiten soll, verarbeitet MapForce bei Auswahl dieser Option den/die im Dialogfeld "Komponenteneinstellungen" definierten Dateinamen. Wenn Sie diese Option auswählen, hat der Root-Node keinen Input-Konnektor, da dieser keine Bedeutung hat. 

	Wenn Sie im Dialogfeld "Komponenteneinstellungen" noch keine Input- oder Output-Datei definiert haben, lautet der Name des Root-Node **Datei: (Standard)**. Andernfalls wird im Root-Node der Name der Input-Datei, gefolgt von einem Semikolon (;), gefolgt vom Namen der Output-Datei angezeigt. Wenn der Name der Input-Datei mit dem der Output-Datei identisch ist, wird er als Name des Root-Node angezeigt.  Beachten Sie, dass Sie entweder diese Option oder die Option *Über das Mapping bereitgestellte dynamische Dateinamen verwenden* verwenden können.
Über das Mapping bereitgestellte dynamische Dateinamen verwenden	Bei Auswahl dieser Option verarbeitet MapForce den/die im Mapping-Bereich definierten Dateinamen, indem das Programm Werte mit dem Root-Node der Komponente verbindet. Bei Auswahl dieser Option erhält der Root-Node einen Input-Konnektor, mit dem Sie Werte verbinden können, die die zu verarbeitenden Dateinamen während der Mapping-Ausführung dynamisch bereitstellen. Wenn Sie auch im Dialogfeld "Komponenteneinstellungen" Dateinamen definiert haben, so werden diese Werte ignoriert. Wenn diese Option ausgewählt ist, wird als Name des Root-Node **Datei: <dynamisch>** angezeigt.  Diese Option kann nicht gleichzeitig mit der Option *Dateinamen aus Komponenteneinstellungen verwenden* verwendet werden.
Strings zu *XML parsen,* Strings zu *JSON parsen,* Strings zu *CSV parsen,* Strings zu *FLF*	Bei Auswahl dieser Option erhält die Komponente einen String-Wert als Input für den Root-Node und konvertiert

© 2016-2022 Altova GmbH

parsen, Strings zu *EDI parsen*	diesen in eine XML-, JSON-, CSV-, FLF- bzw. EDI-Struktur. Nähere Informationen dazu finden Sie unter Parsen und Serialisieren von Strings[1382].
XML in Strings serialisieren,JSON in Strings serialisieren, CSV in Strings serialisieren, FLF in Strings serialisieren, EDI in Strings serialisieren	Bei Auswahl dieser Option erhält die Komponente eine Struktur als Input und konvertiert diese in einen String. Bei der Input-Struktur kann es sich um eine XML-, JSON-, CSV-, FLF- bzw. EDI-Struktur handeln. Nähere Informationen dazu finden Sie unter Parsen und Serialisieren von Strings[1382]

Für die folgenden Komponenten können mehrere Input- oder Output-Dateien definiert werden:

- XML-Dateien
- Textdateien (CSV-*, FLF-* und FlexText**-Dateien)
- EDI-Dokumente**
- Excel-Arbeitsblätter**
- XBRL-Dokumente**
- JSON-Dateien**
- Protocol Buffer-Dateien**

* *MapForce Professional Edition erforderlich*
** *MapForce Enterprise Edition erforderlich*

In der folgenden Tabelle finden Sie Informationen über die Unterstützung für dynamische Input- und Output-Dateien und Platzhalter in MapForce-Sprachen.

Zielsprache	Dynamischer Input-Dateiname	Unterstützung von Platzhaltern für Input-Dateinamen	Dynamischer Output-Dateiname
XSLT 1.0	*	Nicht von XSLT 1.0 unterstützt	Nicht von XSLT 1.0 unterstützt
XSLT 2.0	*	*(1)	*
XSLT 3.0	*	*(1)	*
XQuery	*	*(1)	Nicht von XQuery unterstützt
C++	*	*	*
C#	*	*	*
Java	*	*	*
BUILT-IN	*	*	*

Legende:

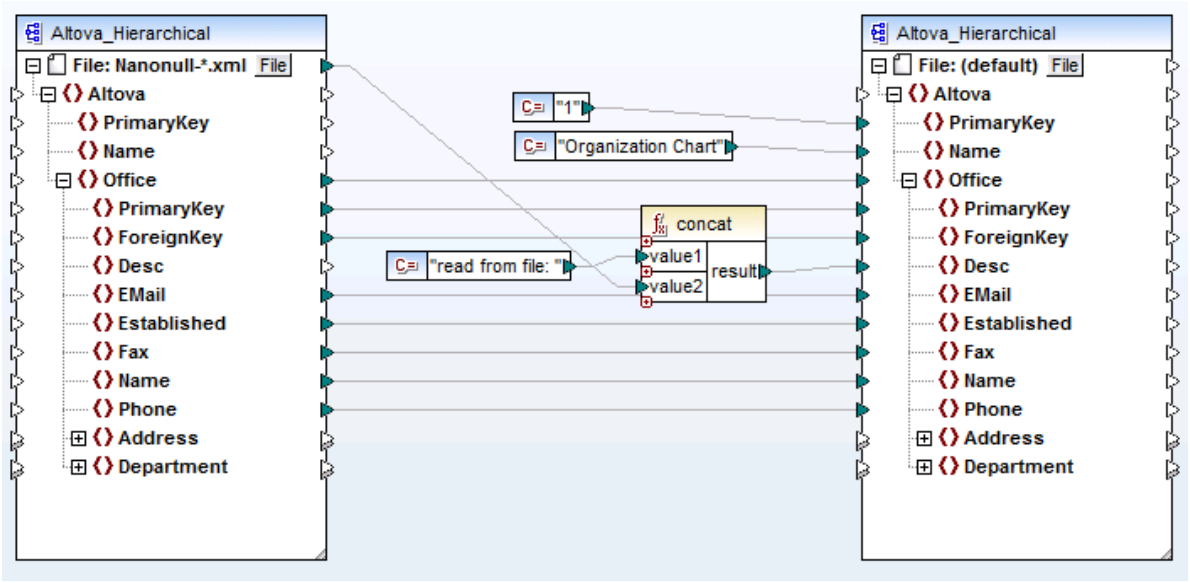
*	Unterstützt

(1)	Für XSLT 2.0, XSLT 3.0 und XQuery wird die Funktion **`fn:collection`** verwendet. In der Implementierung im Altova XSLT 2.0-, XSLT 3.0- und XQuery-Prozessor werden Platzhalter aufgelöst. Andere Prozessoren verhalten sich eventuell anders.

8.5.1 Mappen mehrerer Input-Dateien auf eine einzige Output-Datei

Um mehrere Input-Dateien zu verarbeiten, wählen Sie eine der folgenden Methoden:

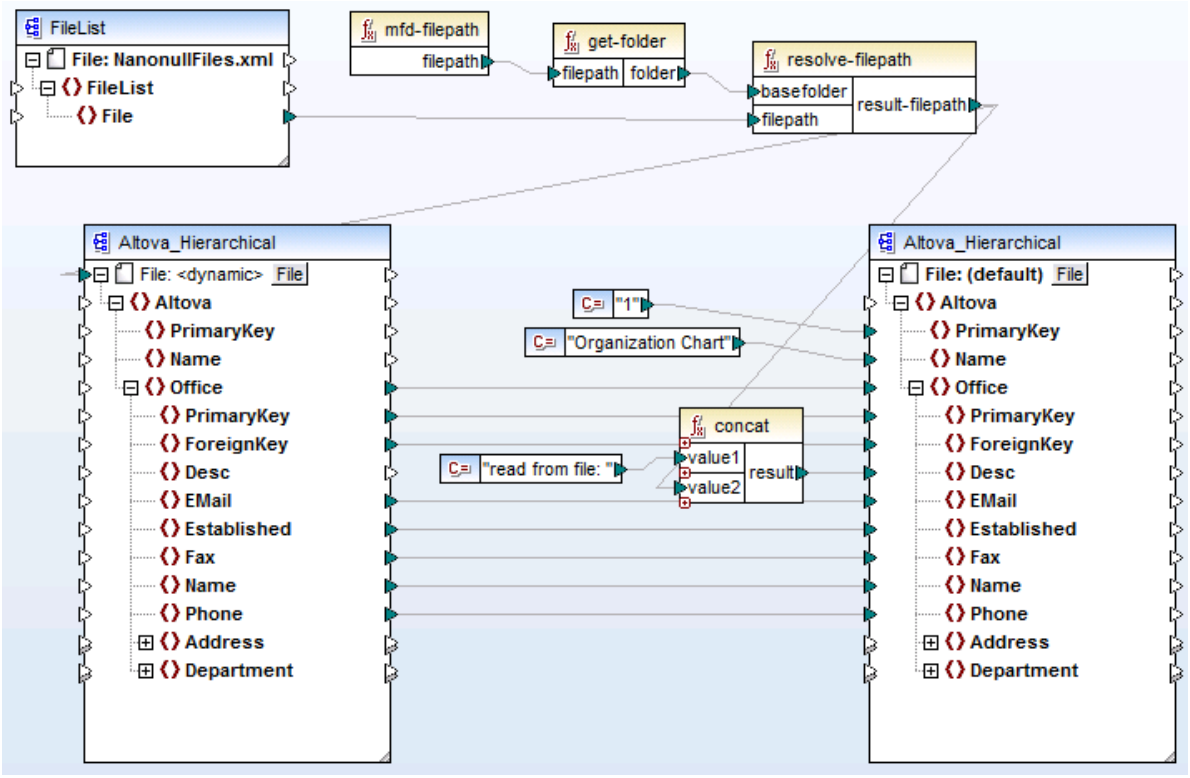
- Geben Sie im Dialogfeld "Komponenteneinstellungen" einen Dateipfad mit Platzhalterzeichen (* oder ?) als Input-Datei ein. Dadurch werden alle damit übereinstimmenden Dateien verarbeitet. Im Beispiel unten wird in der XML-Input-Datei das Platzhalterzeichen ***** verwendet, um alle Dateien, deren Name mit "Nanonull-" beginnt als Mapping-Input bereitzustellen. Mehrere Input-Dateien werden zu einer **einzigen** Output-Datei zusammengeführt, da es keinen dynamischen Konnektor zur Zielkomponente gibt, während die Quellkomponente über das Platzhalterzeichen * mehrere Dateien aufruft. Beachten Sie, dass der Name des Root-Node in der Zielkomponente **Datei: <Standard>** lautet, was darauf hinweist, dass im Dialogfeld "Komponenteneinstellungen" kein Output-Dateipfad definiert wurde. Die Quelldateien werden somit in der Zielkomponente angehängt.



MergeMultipleFiles.mfd (MapForce Basic Edition)

- Mappen Sie eine **Sequenz** von Strings auf den Node *Datei* der Quellkomponente. Jeder String in der Sequenz entspricht einem Dateinamen. Die Strings können auch Platzhalterzeichen enthalten, die automatisch aufgelöst werden. Eine Sequenz von Dateinamen kann von Komponenten wie einer XML-Datei , Datenbanktextfeldern, Textdateien (CSV oder FLF) oder einem Excel-Arbeitsblatt bereitgestellt werden..

© 2016-2022 Altova GmbH

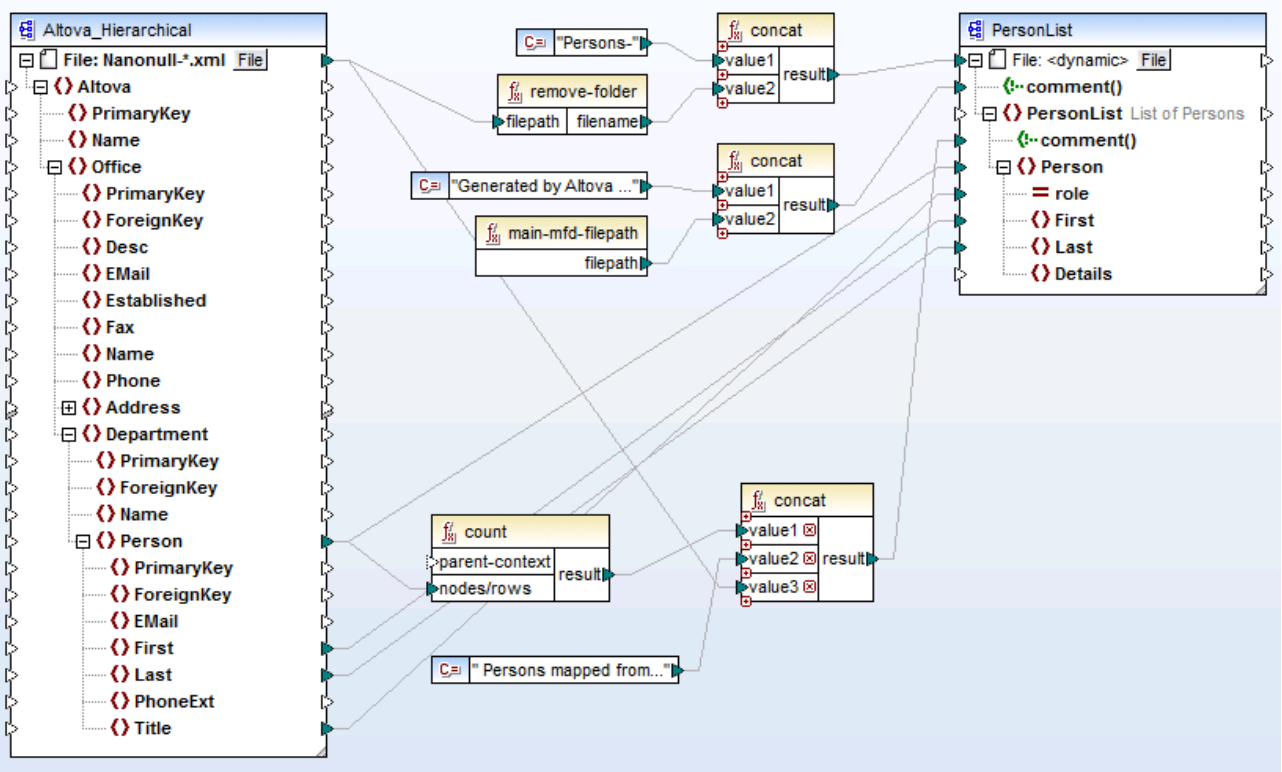


MergeMultipleFiles_List.mfd (MapForce Basic Edition)

8.5.2 Mappen mehrerer Input-Dateien auf mehrere Output-Dateien

Um mehrere Dateien auf mehrere Zieldateien zu mappen, müssen Sie eindeutige Output-Dateinamen generieren. In einigen Fällen können die Output-Dateinamen von Strings in den Input-Daten abgeleitet werden, in anderen Fällen ist es sinnvoll, den Output-Dateinamen vom Input-Dateinamen abzuleiten, z.B. durch Änderung der Dateierweiterung.

Im folgenden Mapping wird der Name der Output-Datei vom Namen der Input-Datei abgeleitet, indem mit Hilfe der `concat`-Funktion das Präfix "Persons-" hinzugefügt wird.



MultipleInputToMultipleOutputFiles.mfd (MapForce Basic Edition)

Anmerkung: Vermeiden Sie es, einfach die Input- und Output-Root-Nodes direkt und ohne Verwendung von Verarbeitungsfunktionen miteinander zu verbinden, da sonst Ihre Input-Dateien bei Ausführung des Mappings überschrieben werden. Sie können die Output-Dateinamen mit Hilfe einer Reihe von Funktionen wie z.B. der oben gezeigten `concat`-Funktion, ändern.

Mit Hilfe der Menüoption **Datei | Mapping-Einstellungen** können Sie die für das Mapping-Projekt verwendeten Dateipfadeinstellungen global definieren (siehe Ändern der Mapping-Einstellungen 116).

8.5.3 Bereitstellen von Dateinamen als Mapping-Parameter

Um benutzerdefinierte Dateinamen als Input-Parameter für das Mapping bereitzustellen, gehen Sie folgendermaßen vor:

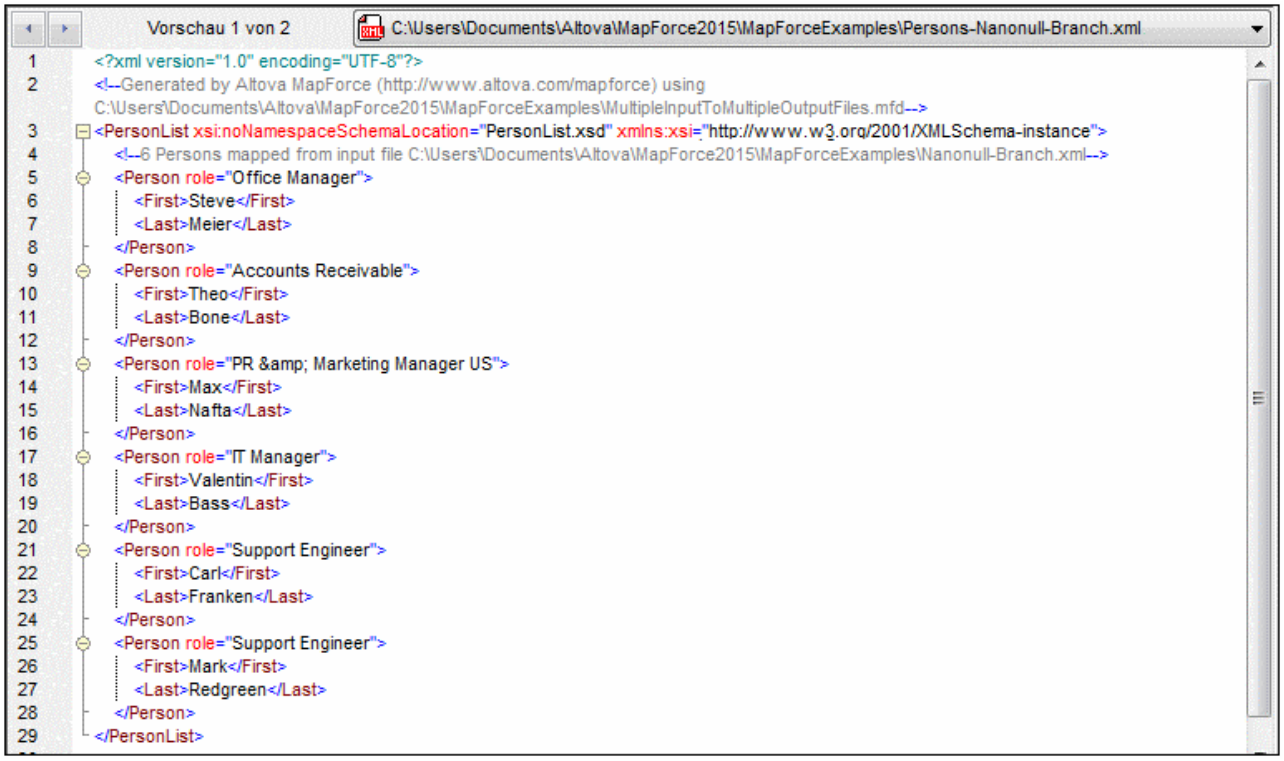
1. Fügen Sie eine einfache Input-Komponente zum Mapping hinzu (Klicken Sie im Menü **Funktion** auf **Input-Komponente einfügen**). Nähere Informationen zu solchen Komponenten finden Sie unter Bereitstellen von Parametern für das Mapping 867.)
2. Klicken Sie auf die Schaltfläche **Datei** (Datei) oder **Datei/String** (Datei/String) der Quellkomponente und wählen Sie **Über das Mapping bereitgestellte dynamische Dateinamen verwenden**.
3. Verbinden Sie die einfache Input-Komponente mit dem Root-Node der Komponente, die als Mapping-Quelle fungiert.

Ein Arbeitsbeispiel dazu finden Sie unter Beispiel: Verwenden von Dateinamen als Mapping-Parameter 872.

 © 2016-2022 Altova GmbH

8.5.4 Anzeigen einer Vorschau auf mehrere Output-Dateien

Klicken Sie auf das Register "Ausgabe", um das Mapping-Ergebnis in einem Vorschaufenster zu sehen. Wenn das Mapping mehrere Output-Dateien erzeugt, wird jede Datei auf einem eigenen nummerierten Register in der Vorschau angezeigt. Klicken Sie auf die Pfeilschaltflächen, um die einzelnen Output-Dateien zu sehen.



MultipleInputToMultipleOutputFiles.mfd

Um die generierten Ausgabedateien zu speichern, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Ausgabe** auf **Alle Ausgabedateien speichern** ( ).
- Klicken Sie auf die Symbolleistenschaltfläche **Alle generierten Ausgaben speichern** ( ).

8.5.5 Beispiel: Aufteilen einer XML-Datei in mehrere

In diesem Beispiel wird gezeigt, wie Sie anhand einer einzigen XML-Quelldatei dynamisch mehrere XML-Dateien generieren. Sie finden die Mapping-Beispieldatei dazu unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\Tut-ExpReport-dyn.mfd**.

Die XML-Quelldatei (die im selben Ordner wie das Mapping liegt) besteht aus der Spesenabrechnung für eine Person namens Fred Landis und enthält fünf Spesenposten (expense-item) unterschiedlichen Typs. Ziel dieses Beispiels ist es, für jeden der unten aufgelisteten Spesenposten eine separate XML-Datei zu generieren.

Person

First	Fred
Last	Landis
Title	Project Manager
Phone	123-456-78
Email	f.landis@nanonull.com

expense-item (5)

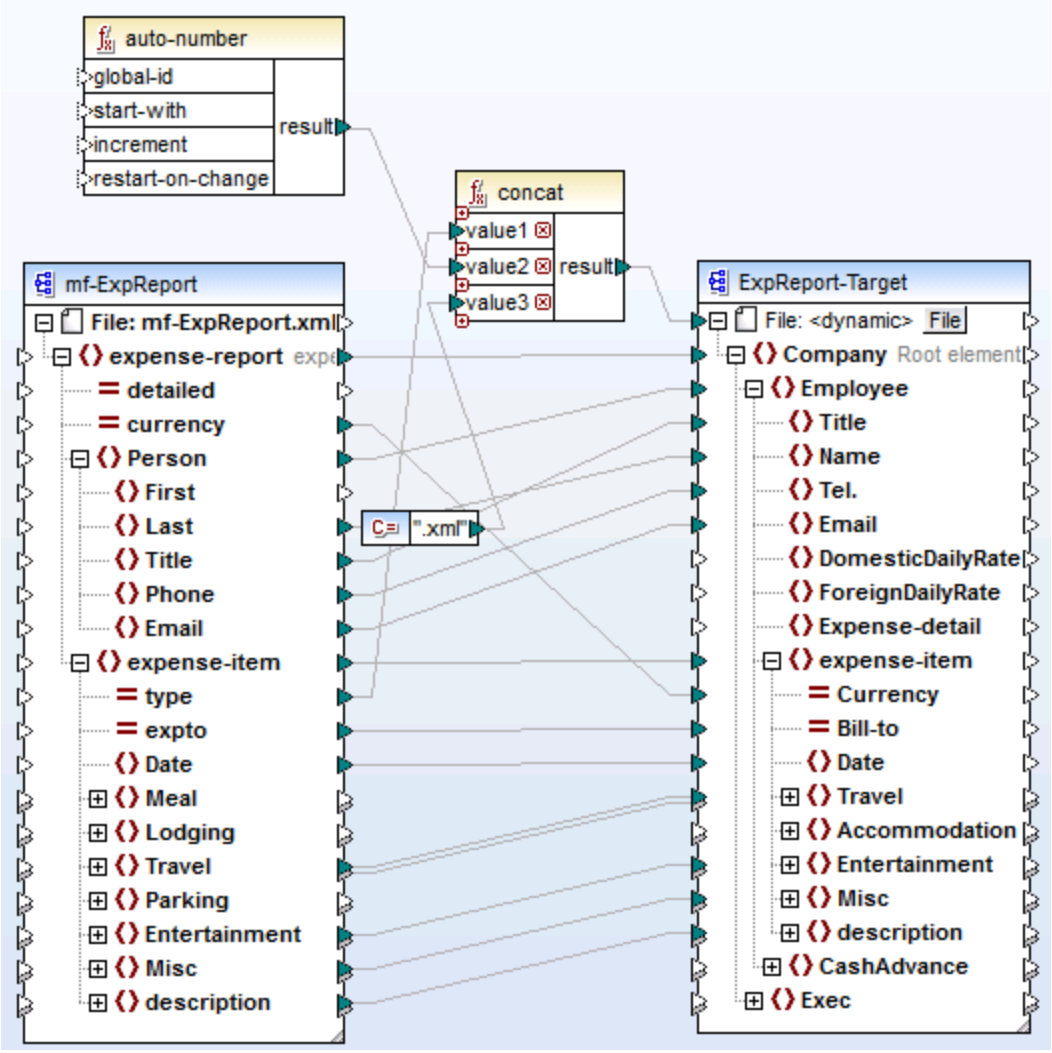
	type	expto	Date	Travel	Lodging
1	Travel	Development	2003-01-02	**Travel** Trav-cost=337.88	
2	Lodging	Sales	2003-01-01		**Lodging**
3	Travel	Accounting	2003-07-07	**Travel** Trav-cost=1014.22	
4	Travel	Marketing	2003-02-02	**Travel** Trav-cost=2000	
5	Meal	Sales	2003-03-03		

mf-ExpReport.xml (in der XMLSpy Grid-Ansicht)

Da das Attribut "type" den jeweiligen Spesentyp definiert, ist dies das Datenelement, das wir zum Aufteilen der Quelldatei verwenden werden. Gehen Sie dazu folgendermaßen vor:

1. Fügen Sie eine `concat`-Funktion ein (Sie können diese mit der Maus aus der Bibliothek **core | string functions** in den Mapping-Bereich ziehen).
2. Fügen Sie (über das Menü **Einfügen | Konstante**) eine Konstante ein und geben Sie als ihren Wert ".xml" ein.
3. Ziehen Sie die Funktion **`auto-number`** aus der Bibliothek **core | generator functions** mit der Maus in den Mapping-Bereich.
4. Klicken Sie auf die Schaltfläche **Datei** (Datei) oder **Datei/String** (Datei/String) der Zielkomponente und wählen Sie den Befehl **Über das Mapping bereitgestellte dynamische Dateinamen verwenden**.
5. Erstellen Sie die Verbindungen wie oben gezeigt und klicken Sie auf das Register "Ausgabe", um das Ergebnis des Mappings zu sehen.

© 2016-2022 Altova GmbH



Tut-ExpReport-dyn.mfd (MapForce Basic Edition)

Beachten Sie, dass die erzeugten Ausgabedateien folgendermaßen dynamisch benannt werden:

- Das Attribut `type` liefert den ersten Teil des Dateinamens, z.B. Travel.
- Die Funktion `auto-number` liefert die fortlaufend nummerierten Dateinummern (z.B. "Travel1", "Travel2", usw.)
- Die Konstante liefert die Dateierweiterung, d.h. .xml, also lautet der Dateiname der ersten Datei Travel1.xml.

8.5.6 Beispiel: Aufteilen einer Datenbanktabelle in mehrere XML-Dateien

In diesem Beispiel wird gezeigt, wie Sie mehrere XML-Dateien - für jeden Datensatz einer Datenbanktabelle eine - dynamisch generieren. Die Beispielmapping-Datei dazu finden Sie unter: **<Dokumente>\Altova\MapForce2023\MapForceExamples\PersonDB-dyn.mfd**.

Die Datenbank-Quelldatei (Sie liegt im selben Ordner wie das Mapping) enthält eine Tabelle namens "Person", die 21 Person-Datensätze enthält. Ziel ist die Generierung einer separaten XML-Datei für jeden einzelnen Datensatz in der Tabelle "Person".

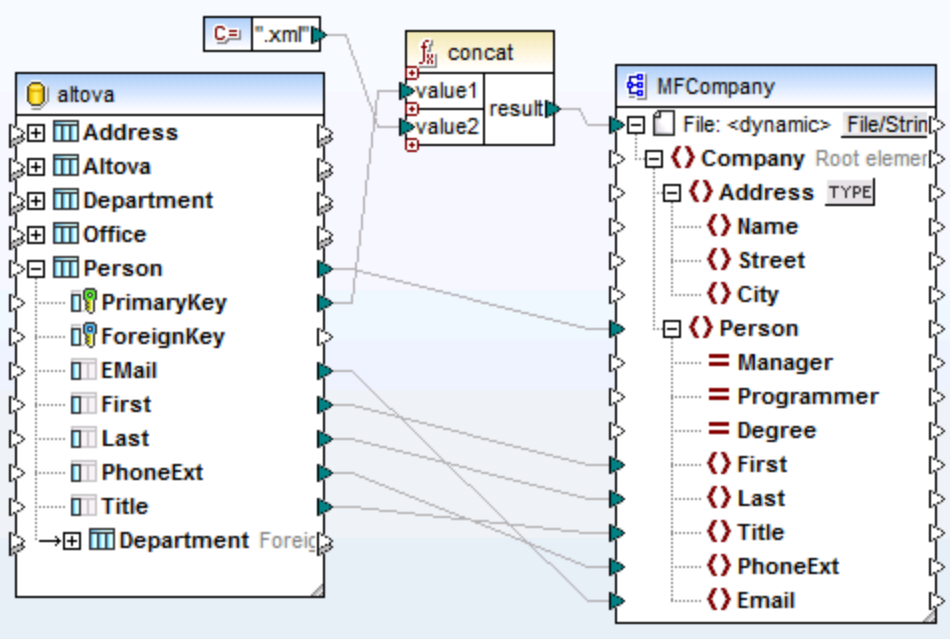
Person

PrimaryKey	ForeignKey	EMail	First	Last
1	1	v.callaby@nanonu	Vernon	Callaby
2	1	f.further@nanonu	Frank	Further
3	1	l.matise@nanonu	Loby	Matise
4	2	j.firstbread@nanc	Joe	Firstbread
5	2	s.sanna@nanonul	Susi	Sanna
6	3	f.landis@nanonul	Fred	Landis
7	3	m.landis@nanonu	Michelle	Butler
8	3	t.little@nanonull.	Ted	Little

Da das Feld "PrimaryKey" die einzelnen Personen in der Tabelle eindeutig identifiziert, ist dies das Datenelement, das wir zum Aufteilen der Quelldatenbank in separate Dateien verwenden werden. Gehen Sie dazu folgendermaßen vor:

1. Fügen Sie eine `concat`-Funktion ein (Sie können diese mit der Maus aus der Bibliothek **core | string functions** in den Mapping-Bereich ziehen).
2. Fügen Sie (über das Menü **Einfügen | Konstante**) eine Konstante ein und geben Sie als ihren Wert ".xml" ein.
3. Ziehen Sie die Funktion **`auto-number`** aus der Bibliothek **core | generator functions** mit der Maus in den Mapping-Bereich.
4. Klicken Sie auf die Schaltfläche **Datei** (Datei) oder **Datei/String** (Datei/String) der Zielkomponente und wählen Sie den Befehl **Über das Mapping bereitgestellte dynamische Dateinamen verwenden**.
5. Erstellen Sie die Verbindungen wie oben gezeigt und klicken Sie auf das Register "Ausgabe", um das Ergebnis des Mappings zu sehen.

© 2016-2022 Altova GmbH



PersonDB-dyn.mfd (MapForce Professional Edition)

Beachten Sie, dass die erzeugten Ausgabedateien folgendermaßen dynamisch benannt sind:

- Das Feld **PrimaryKey** liefert den ersten Teil des Dateinamens, z.B. 1.
- Die Konstante liefert die Dateierweiterung, d.h. .xml, also lautet der Dateiname der ersten Datei 1.xml.

8.6 Parsen und Serialisieren von Strings

Das Parsen und Serialisieren von Strings ist ein komplexe Mapping-Methode, mit der Sie die Komponente so konfigurieren können, dass sie entweder Daten aus einem String parst oder zu einem String serialisiert. Diese Methode kann alternativ zum Lesen von Daten aus (bzw. Schreiben von Daten in) Dateien verwendet werden. MapForce-Komponenten, die Strings parsen oder Daten zu Strings serialisieren, lassen sich in den verschiedensten Situationen einsetzen, z.B.:

- Sie müssen Strukturen wie z.B. XML-, Text- oder JSON-Dateien in Datenbankfelder oder Zellen eines Excel-Arbeitsblatts einfügen.
- Sie müssen in Datenbankfeldern gespeicherte XML-Fragmente in eigenständige XML-Dateien konvertieren.
- Sie haben in Form von Text gespeicherte alte Datenbestände (z.B. Inhalt mit fester Länge, der in einem einzigen Datenbankfeld gespeichert ist) und Sie möchten diese Daten (z.B. mit Hilfe von FlexText) in eine sortierbare, feldbasierte Struktur konvertieren.

Das Parsen und Serialisieren von Strings steht für die folgenden MapForce-Komponententypen zur Verfügung:

- Text (CSV, Textfelder mit fester Länge, EDI-Daten und MapForce FlexText-Vorlagen)
- JSON-Schemadateien
- XML-Schemadateien

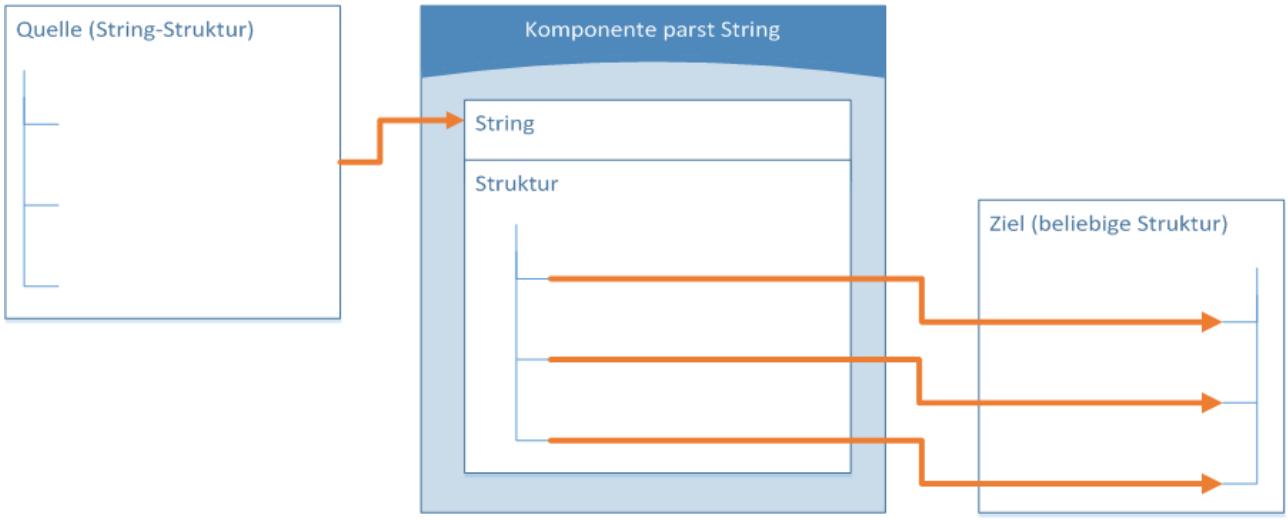
Das Parsen und die Serialisierung von Strings wird für alle Komponententypen in der Zielsprache BUILT-IN unterstützt. Zusätzlich dazu wird das Parsen von Strings ins JSON-Format und die Serialisierung von JSON anhand von Strings in BUILT-IN, C# und Java unterstützt.

8.6.1 Die Parsen/Serialisieren-Komponente

Eine Parsen/Serialisieren-Komponente in MapForce ist eine Hybrid-Komponente, die weder eine Quell- noch eine Zielkomponente ist. Aufgrund ihrer Rolle im Mapping-Design müssen solche Komponenten zwischen weitere Quell- und Zielkomponenten platziert werden.

Sie können eine "String parsen/serialisieren"-Komponente zum Parsen von Strings verwenden, wenn Sie einen String mit einer Struktur (z.B. in einer Datenbank als String gespeicherte XML-Daten) aus irgendeinem Grund in ein anderes Format konvertieren müssen. Wenn Daten aus dem Quellstring in die "Parsen/Serialisieren"-Komponente geparst werden, bedeutet dies, dass der Quellstring in eine MapForce-Struktur konvertiert wird. Dadurch erhalten Sie Zugriff auf jedes Element oder Attribut der als String gespeicherten XML-Quelle.

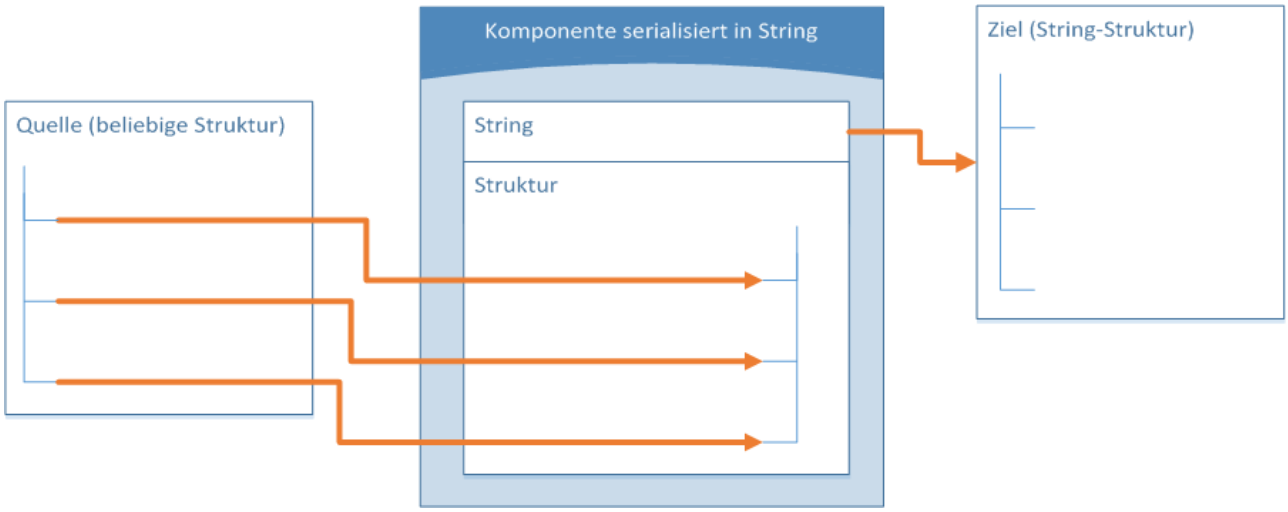
© 2016-2022 Altova GmbH



Allgemeine "String parsen"-Komponente

Im obigen Diagramm sehen Sie die typische Struktur einer MapForce-Komponente, die einen String parst. Beachten Sie, dass die "String parsen/serialisieren"-Komponente zwischen die Quell- und Zielkomponente des Mappings platziert wird. Diese Komponente erhält über einen einzigen MapForce-Konnektor, der mit ihrem obersten **String**-Node verbunden ist, eine String-Struktur als Input. Bei der Output-Struktur kann es sich um jedes der von MapForce unterstützten Datenzielformate handeln. Ein Beispiel dazu finden Sie unter Beispiel: Parsen eines String (Text mit fester Länge auf Excel) 1385.

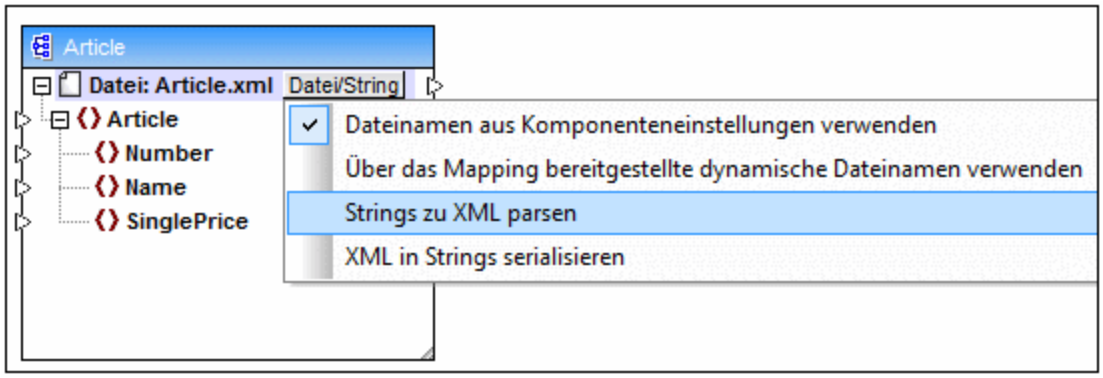
Beim Serialisieren von Daten von einer Komponente in einen String wird der Vorgang umgekehrt. In diesem Fall wird die gesamte Struktur der MapForce-Komponente zu einer String-Struktur, die Sie nach Bedarf weiter bearbeiten können. Auf diese Art können Sie z.B. eine XML-Datei (oder ein XML-Codefragment) in ein Datenbankfeld oder in eine einzelne Zelle eines Excel-Arbeitsblatts schreiben.



Allgemeine "In String serialisieren" Komponente

Im obigen Diagramm sehen Sie eine allgemeine MapForce-Komponente zum Serialisieren von Daten in einen String. Die Komponente erhält als Input eine beliebige von MapForce unterstützte Datenquelle (die mittels MapForce-Standardkonnektoren verbunden wird). Die Output-Struktur ist ein String, den Sie über einen einzigen vom obersten **String**-Node der Komponente zu einem Datenelement der Zielkomponente (z.B. der Zelle eines Excel-Arbeitsblatts) gezogenen MapForce-Konnektor weiterleiten können. Ein Beispiel dazu finden Sie unter Beispiel: Serialisieren eines String (XML auf Datenbank) 1390.

Sie können eine Komponente jederzeit über das Mapping-Fenster für das Parsen oder Serialisieren von Strings designieren. Klicken Sie dazu auf die Schaltfläche **Datei/String** (Datei/String) neben dem Root-Node und wählen Sie anschließend die gewünschte Option aus.



Wechseln des Komponentenmodus

Anmerkung: Eine "String parsen/serialisieren"-Komponente kann nicht gleichzeitig Daten aus einem String lesen und in einen String schreiben. Daher kann der Root-Node nur entweder einen eingehenden oder einen ausgehenden Konnektor (nicht aber beides) haben. Wenn Sie versuchen, dieselbe Komponente für beide Operationen zu verwenden, wird ein Fehler generiert.

Wenn Sie eine Komponente für das Parsen oder Serialisieren von Strings designieren, ändert sich die Darstellung der Komponente wie folgt:

- Die Komponente erhält in Ihrem Titel das **Parsen**- oder **Serialisieren**-Präfix.
- Die Titelleiste hat, ähnlich wie Funktionskomponenten, eine gelbe Hintergrundfarbe.
- Der oberste Node beginnt mit dem Präfix **String:** und wird durch das Symbol gekennzeichnet.
- Wenn die Komponente einen String parst, hat der Output-Konnektor des Root-Node keine Bedeutung und steht daher nicht zur Verfügung.
- Wenn die Komponente Daten in einen String serialisiert, hat der Input-Konnektor des Root-Node keine Bedeutung und steht daher nicht zur Verfügung.

Wenn sich eine Komponente im Modus "String parsen/serialisieren" befindet, können Sie ihre Einstellungen ähnlich wie im dateibasierten Modus ändern (siehe Ändern der Komponenteneinstellungen 82). Beachten Sie, dass im Modus "Parsen"- oder "Serialisieren" nicht alle Komponenteneinstellungen zur Verfügung stehen.

 © 2016-2022 Altova GmbH

8.6.2 Beispiel: Parsen eines String (Text mit fester Länge auf Excel)

In diesem Beispiel wird Schritt für Schritt beschrieben, wie Sie ein Mapping-Design zum Parsen von String-Daten erstellen. Sie finden die Beispieldatei dazu unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\ParseString.mfd**.

Angenommen, Sie haben einige alte Textdaten, die in einem einzigen Datenbankfeld gespeichert sind. Bei den Textdaten handelt es sich um eine Mitarbeiterliste, die in der Spalte RESOURCE gespeichert und als Felder mit fester Länge formatiert ist:

```
|----|-------------------------------------------------------------------------------------------|------------|
| ID | RESOURCE                                                                                  | TYPE       |
|----|-------------------------------------------------------------------------------------------|------------|
| 1  | P00001Callaby        Vernon         582Office Manager           v.callaby@nanonull.com    | Employees  |
|    | P00002Further        Frank          471Accounts Receivable      f.further@nanonull.com    |            |
|    | P00003Matise         Loby           963Accounting Manager       l.matise@nanonull.com     |            |
|    | P00004Firstbread     Joe            621Marketing Manager Europe j.firstbread@nanonull.com |            |
|    | P00005Sanna          Susi           753Art Director             s.sanna@nanonull.com       |            |
|    | P00006Landis         Fred           951Program Manager          f.landis@nanonull.com     |            |
|    | P00007Butler         Michelle       654Software Engineer        m.landis@nanonull.com     |            |
|    | P00008Little         Ted            852Software Engineer        t.little@nanonull.com     |            |
|    | P00009Way            Ann            951Technical Writer         a.way@nanonull.com        |            |
|    | P00010Gardner        Liz            753Software Engineer        l.gardner@nanonull.com    |            |
|    | P00011Smith          Paul           334Software Engineer        p.smith@nanonull.com      |            |
|----|-------------------------------------------------------------------------------------------|------------|
```

Da die Textdaten als ein einziges Datenbankfeld gespeichert sind, können Sie die einzelnen Mitarbeiterdatensätze nicht einfach aufrufen und bearbeiten. Auf diese Art wird es schwierig, neue Mitarbeiter hinzuzufügen, Mitarbeiter zu löschen oder Daten zu sortieren. Unser Ziel ist es in diesem Beispiel, die Textdaten aus dem Datenbankfeld RESOURCE zu extrahieren und in eine Struktur aufzuteilen, um die Datensätze einfach verarbeiten zu können.

Dies lässt sich mit Hilfe der Komponente "String parsen/serialisieren" bewerkstelligen. Als erstes erhält die Komponente "String parsen/serialisieren" als Input die Textdaten. Diese werden anschließend geparst und in eine Struktur konvertiert. Schließlich wird die Struktur in ein Zielformat - in diesem Beispiel in ein Excel-Arbeitsblatt - geschrieben. Sie können jedoch im Allgemeinen jedes andere von MapForce unterstützte Ausgabeformat verwenden.

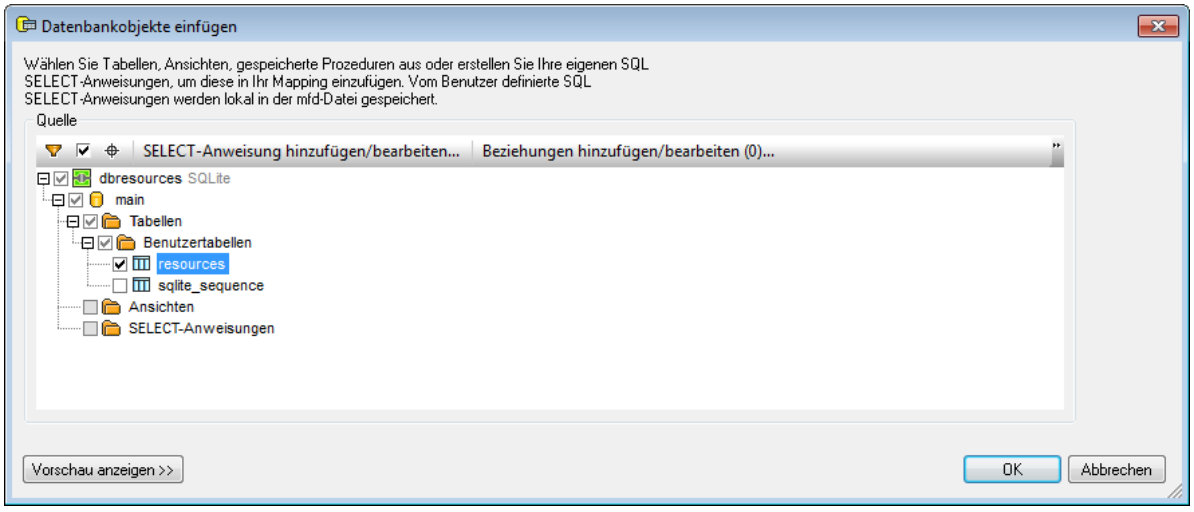
Kurz zusammengefasst, konvertiert das in diesem Beispiel beschriebene Mapping den Inhalt des Datenbankfelds RESOURCE in eine Tabelle. Nachdem das Mapping ausgeführt wurde, entspricht jede Tabellenzeile einem Mitarbeiter und jede Spalte einem der Felder mit fester Länge - und zwar in folgender Reihenfolge: ID, Nachname, Vorname, Durchwahl, Berufsbezeichnung, E-Mail-Adresse.

	A	B	C	D	E	F	G
1	P00001	Callaby	Vernon	582	Office Manager	v.callaby@nanonull.com	
2	P00002	Further	Frank	471	Accounts Receivable	f.further@nanonull.com	
3	P00003	Matise	Loby	963	Accounting Manager	l.matise@nanonull.com	
4	P00004	Firstbread	Joe	621	Marketing Manager Europe	j.firstbread@nanonull.com	
5	P00005	Sanna	Susi	753	Art Director	s.sanna@nanonull.com	
6	P00006	Landis	Fred	951	Program Manager	f.landis@nanonull.com	
7	P00007	Butler	Michelle	654	Software Engineer	m.landis@nanonull.com	
8	P00008	Little	Ted	852	Software Engineer	t.little@nanonull.com	
9	P00009	Way	Ann	951	Technical Writer	a.way@nanonull.com	
10	P00010	Gardner	Liz	753	Software Engineer	l.gardner@nanonull.com	
11	P00011	Smith	Paul	334	Software Engineer	p.smith@nanonull.com	
12							
13							

Erwarteter Output nach dem Parsen des String

Gehen Sie dazu folgendermaßen vor:

1. Fügen Sie die Quelldatenbank zum Mapping-Bereich hinzu. Die Datenbank steht unter dem folgenden Pfad als eigenständige SQLite-Datenbankdatei zur Verfügung: **<Dokumente>\Altova\MapForce2023\MapForceExamples\\dbresources.db**. (Verwenden Sie den Menübefehl **Einfügen | Datenbank**, um die Datenbankkomponente hinzuzufügen; siehe auch Herstellen einer Verbindung zu einer Datenbank 167).
2. Wenn Sie aufgefordert werden, die Datenbankobjekte einzufügen, wählen Sie die Tabelle **resources** aus.

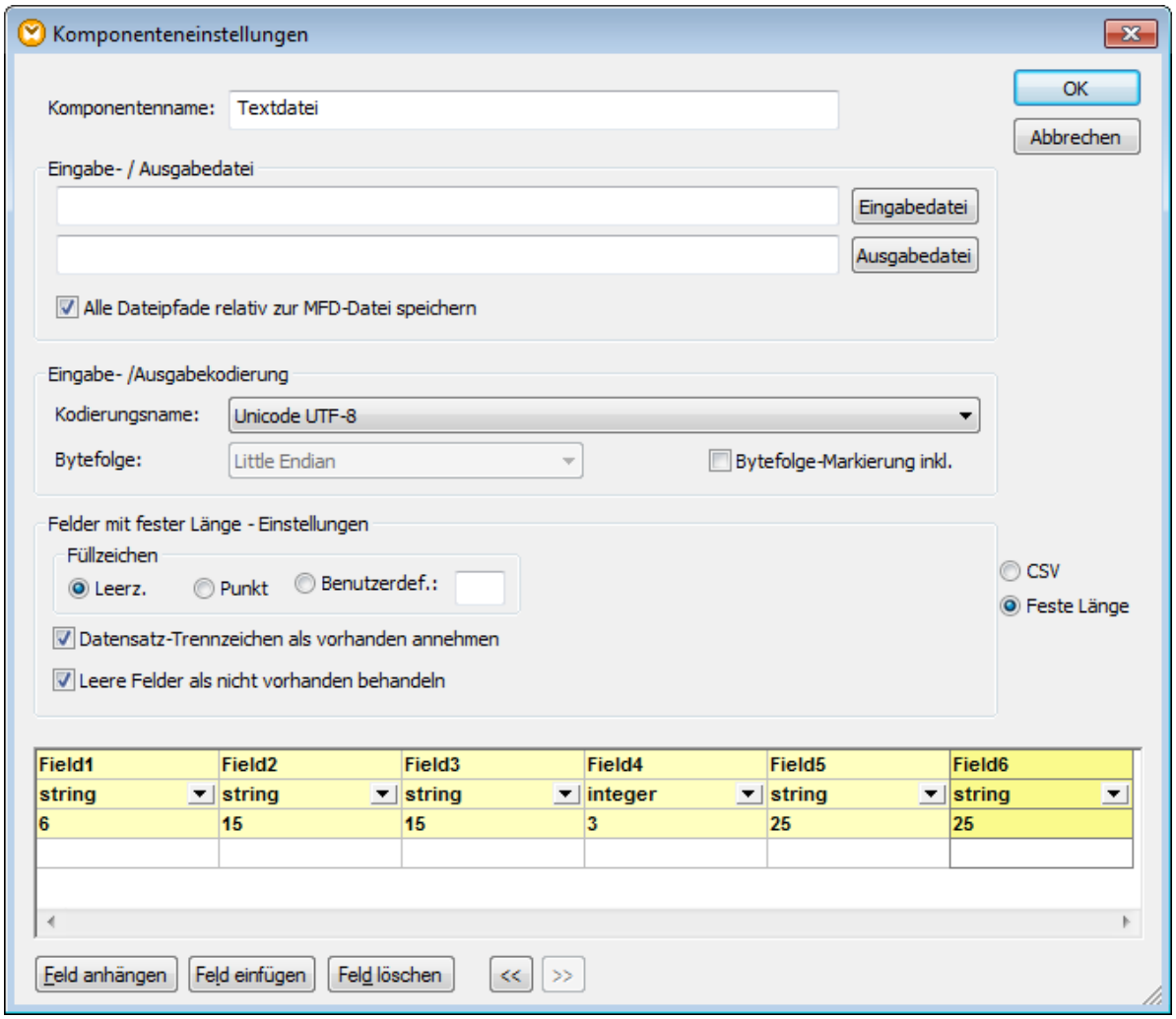


3. Fügen Sie eine Textkomponente zum Mapping-Bereich hinzu (Menübefehl **Einfügen | Textdatei**). Da es sich bei den Quelldaten um durch Felder getrennten Text handelt, wählen Sie die Option **Einfache Verarbeitung... verwenden,** wenn Sie von MapForce gefragt werden.
4. Konfigurieren sie die Struktur der Textkomponente entsprechend der Struktur der in der Datenbank gespeicherten Text-Quelldaten. Wie Sie möglicherweise bereits bemerkt haben, besteht der Quelltext aus den folgenden sechs Feldern mit fester Länge:

 © 2016-2022 Altova GmbH

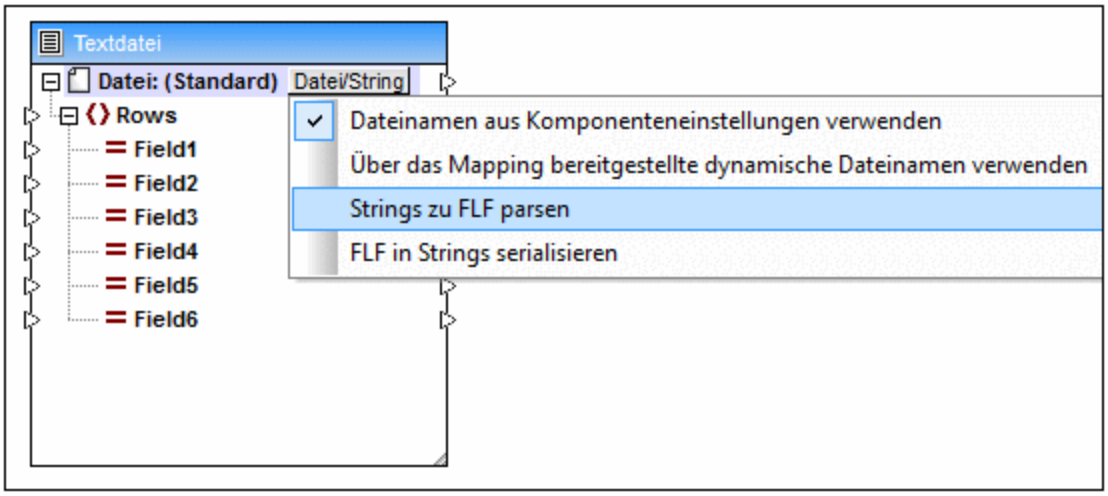
Größe in Zeichen	Feld
6	ID
15	Nachname
15	Vorname
3	Durchwahl
25	Berufsbezeichnung
25	E-Mail-Adresse

Deklarieren Sie die Textkomponente daher als **feste Länge** und fügen Sie sechs Felder hinzu, die den obigen Positionen entsprechen. (Um das Dialogfeld "Komponenteneinstellungen zu öffnen, klicken Sie mit der rechten Maustaste auf die Komponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften**.)



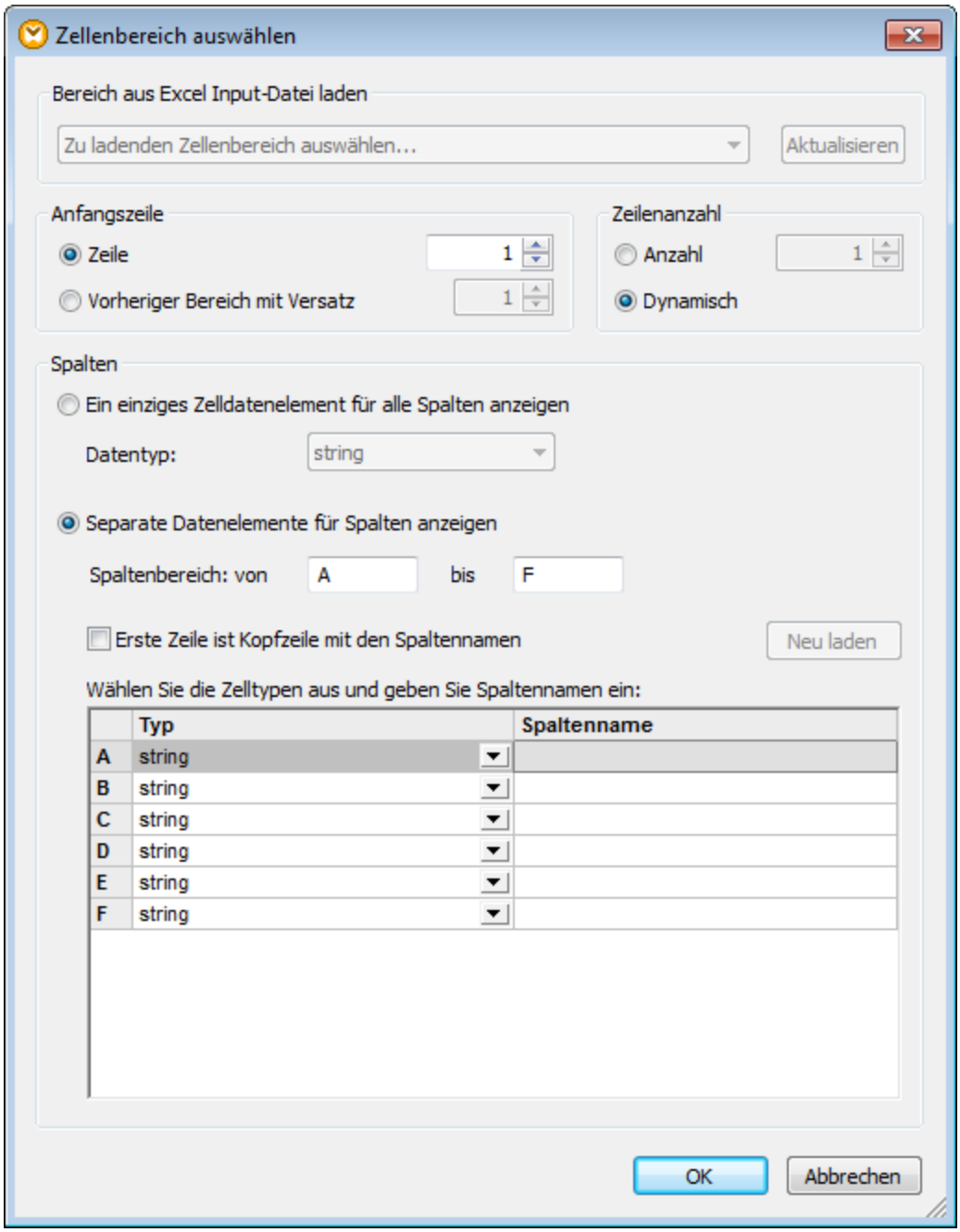
Beachten Sie das **Feld4**, in dem **integer** als Datentyp definiert ist. Die Deklaration von **Feld4** als numerischer Typ ist zwar für das aktuelle Beispiel nicht von Bedeutung, doch stellen Sie damit sicher, dass die aus dem Quelltext extrahierte Durchwahl (**Field4**) als numerischer Wert validiert wird.

5. Klicken Sie auf Datei/String und wählen Sie anschließend aus dem Kontextmenü den Befehl **Strings zu FLF parsen**. Dadurch wird MapForce angewiesen, Strings in dieser Komponente zum Format FLF (Felder mit fester Länge) zu parsen.

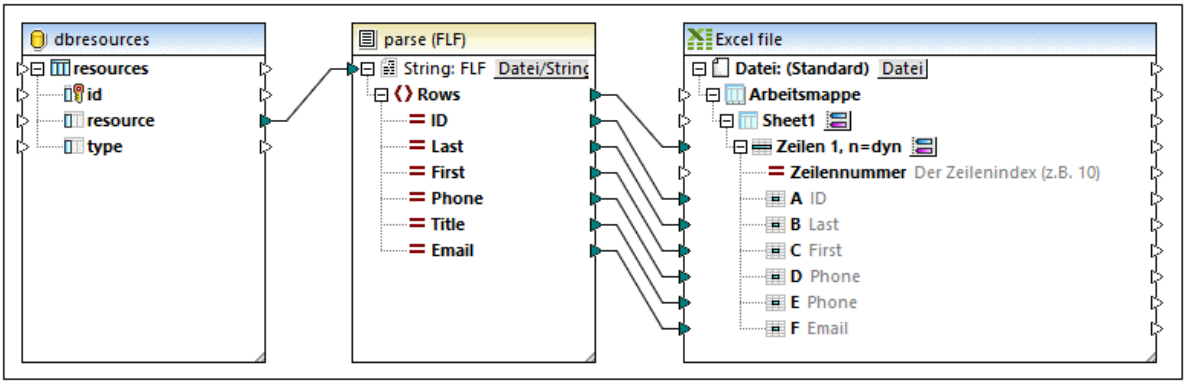


6. Fügen sie die Excel 2007+-Zielkomponente zum Mapping-Bereich hinzu. Klicken Sie auf **Überspringen**, wenn Sie aufgefordert werden, eine Beispieldatei anzugeben. (Sie können die Excel-Komponente mit dem Menübefehl **Einfügen | Excel 2007+** hinzufügen, siehe auch Hinzufügen von Microsoft Excel-Dateien als Mapping-Komponenten 740).
7. Klicken Sie auf die Schaltfläche neben dem Node **Zeile 1, n=dyn** und konfigurieren Sie die Excel-Komponente so, dass für jedes Textfeld, beginnend mit der ersten Zeile, eine Zeile geschrieben wird (siehe Abbildung unten) (Nähere Informationen zum Excel 2007+-Komponententypen und ihrer Konfiguration finden Sie unter Informationen zur Excel 2007+-Komponente 741).

 © 2016-2022 Altova GmbH



8. Ziehen Sie, wie unten gezeigt, die Verbindungslinien zwischen den Datenelementen der Komponenten.



Auf der linken Seite des Mappings wird der Inhalt der Datenbankspalte "resource" von einem Stringwert in eine MapForce-Struktur konvertiert (geparst). Auf der rechten Seite des Mappings werden die Datenelemente der Komponente *Parsen (Textdatei)* mit einzelnen Excel-Spalten verbunden, wodurch der Quellstring in einzelne sortierbare Zellen aufgeteilt wird.

Sie haben nun eine MapForce Design-Datei fertig erstellt, die String-Daten parst und anhand dieser Daten eine Struktur generiert. Wenn Sie auf das Register **Ausgabe** klicken, werden die alten Textdatenbestände nun, wie beabsichtigt, in einzelne Zeilen und Spalten eines Excel-Arbeitsblatts konvertiert.

8.6.3 Beispiel: Serialisieren in einen String (XML auf Datenbank)

In diesem Beispiel wird Schritt für Schritt beschrieben, wie Sie ein Mapping-Design zum Serialisieren von Daten in einen String erstellen. Sie finden die Beispieldatei dazu unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2023\MapForceExamples\SerializeToString.mfd**.

Angenommen, Sie haben eine XML-Datei (und das dazugehörige Schema), die aus mehreren `<Person>` Elementen besteht. In jedem `<Person>` Element sind der Vor- und Nachname, die Berufsbezeichnung, Telefondurchwahl und E-Mail-Adresse einer Person wie folgt beschrieben:

```
<Person>
      <First>Joe</First>
      <Last>Firstbread</Last>
      <Title>Marketing Manager Europe</Title>
      <PhoneExt>621</PhoneExt>
      <Email>j.firstbread@nanonull.com</Email>
</Person>
```

Ihr Ziel ist es, die einzelnen `<Person>` Elemente aus der XML-Datei zu extrahieren und diese (einschließlich der XML-Tags) als neue Datenbankdatensätze in die Tabelle `PEOPLE` einer SQLite-Datenbank einzufügen. Die Tabelle `PEOPLE` enthält nur zwei Spalten: `ID` und `PERSON`. Ihre vollständige Definition lautet:

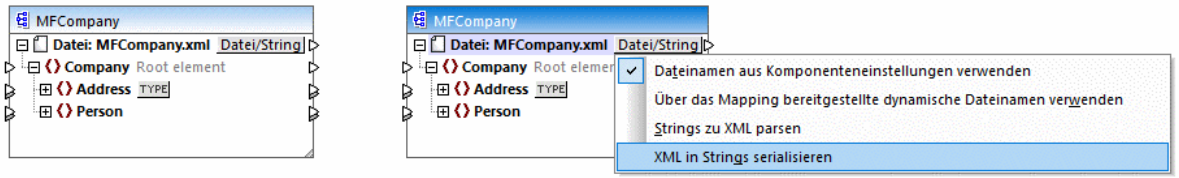
```
CREATE TABLE PEOPLE (ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, PERSON TEXT);
```

Nach Ausführung des Mappings sollte die Tabelle `PEOPLE` genauso viele Zeilen, wie die XML-Datei `<Person>`-Elemente enthält, haben.

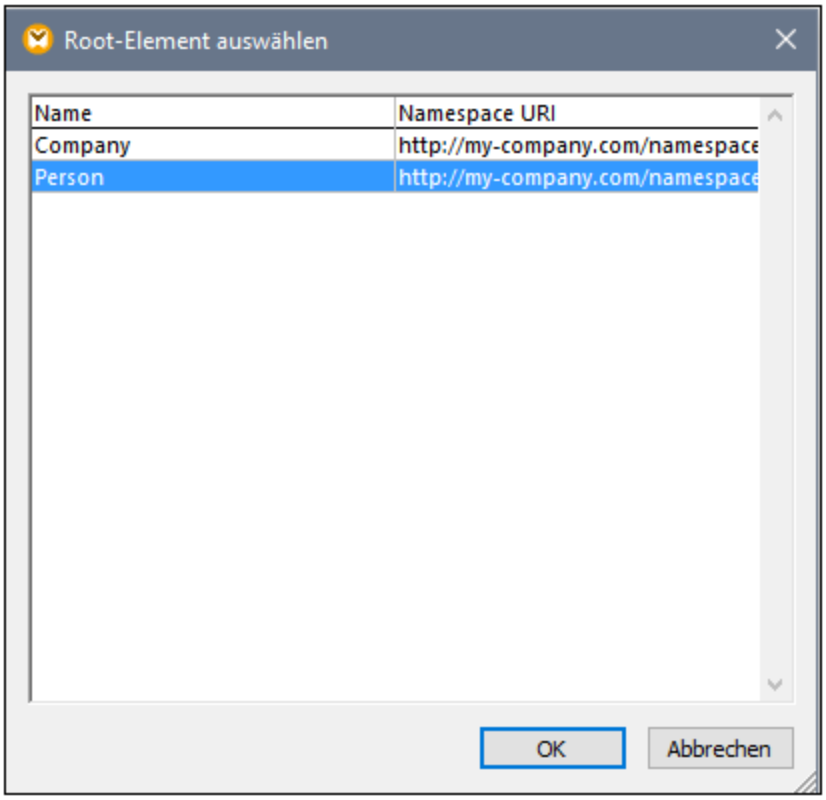
 © 2016-2022 Altova GmbH

Gehen Sie dazu folgendermaßen vor:

1. Fügen Sie die XML-Quellkomponente zum Mapping-Bereich hinzu (Menübefehl **Einfügen | XML-Schema/Datei**). Sie finden die Beispieldatei unter: **<Dokumente>\Altova\MapForce2023\MapForceExamples\MFCompany.xml**.
2. Duplizieren Sie die XML-Komponente (mittels Kopieren/Einfügen).
3. Klicken Sie in der duplizierten XML-Komponente auf Datei/String, und wählen Sie anschließend den Befehl **XML in Strings serialisieren**.

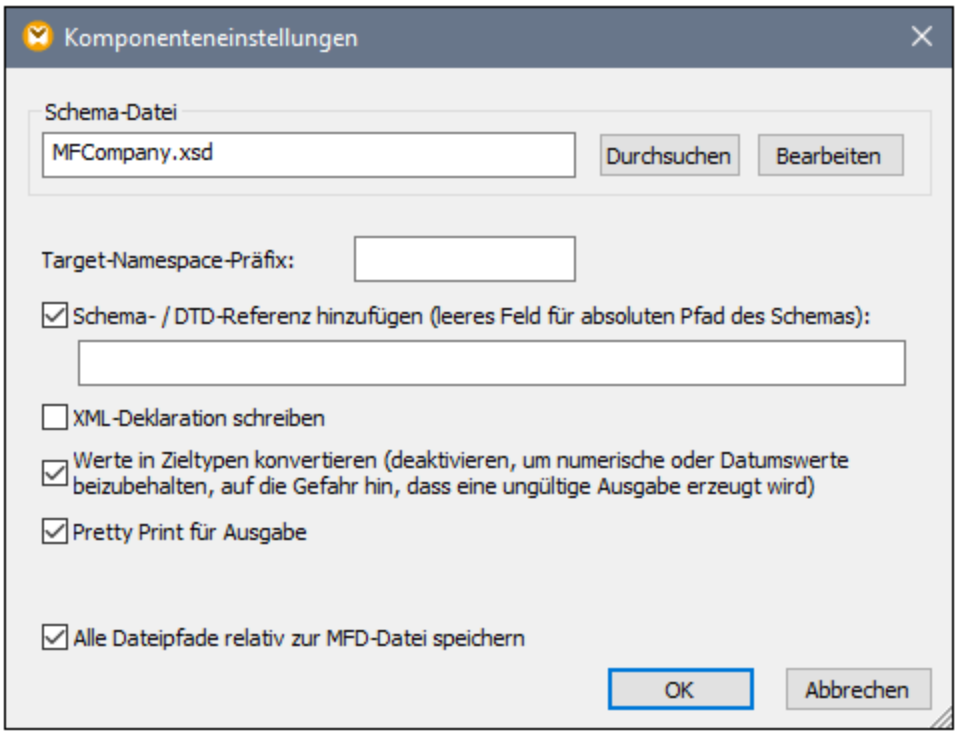


4. Klicken Sie mit der rechten Maustaste auf die duplizierte Komponente und wählen Sie im Kontextmenü den Befehl **Root-Element ändern**. Ändern Sie nun das Root-Element in `<Person>`.



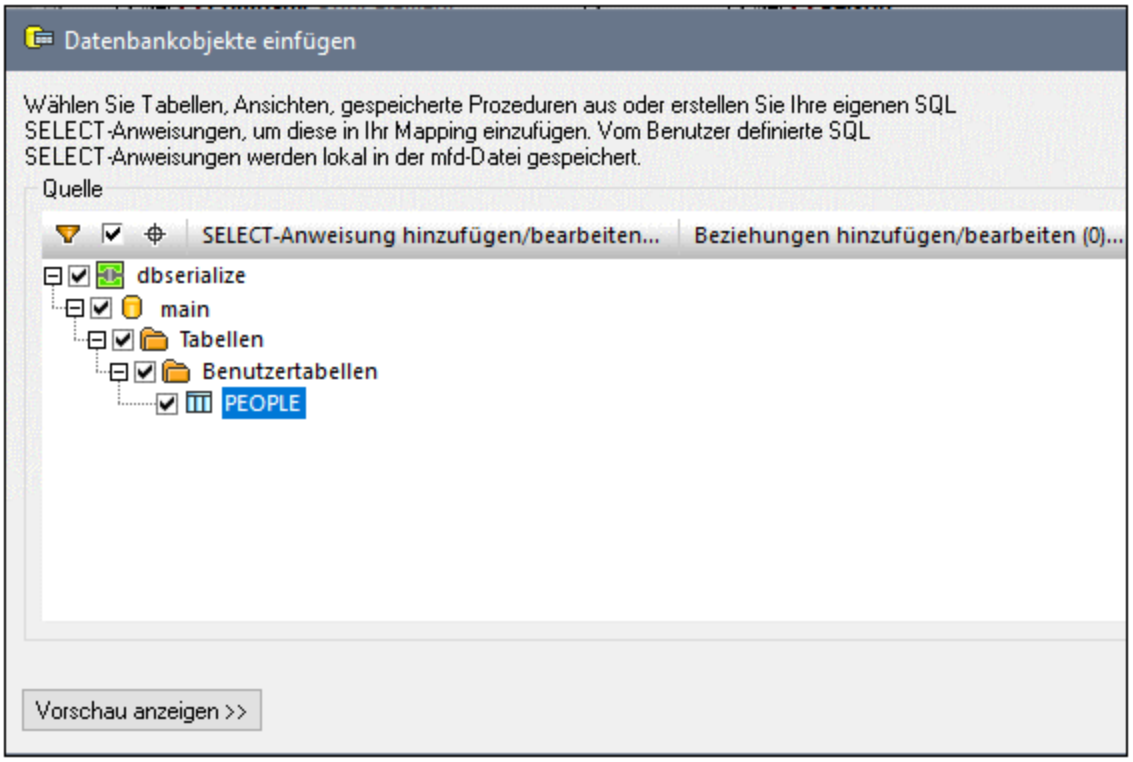
Im Allgemeinen können Sie das Root-Element in jedes Element ändern, das im XML-Schema eine globale (und nicht lokale) Deklaration hat. Elemente, die in Ihrem Schema nicht global definiert sind, werden im Dialogfeld "Root-Element auswählen" nicht aufgelistet.

5. Doppelklicken Sie auf die Komponente und deaktivieren Sie das Kontrollkästchen **XML-Deklaration schreiben**, um zu verhindern, dass die XML-Deklaration für jedes `<Person>`-Element geschrieben wird.

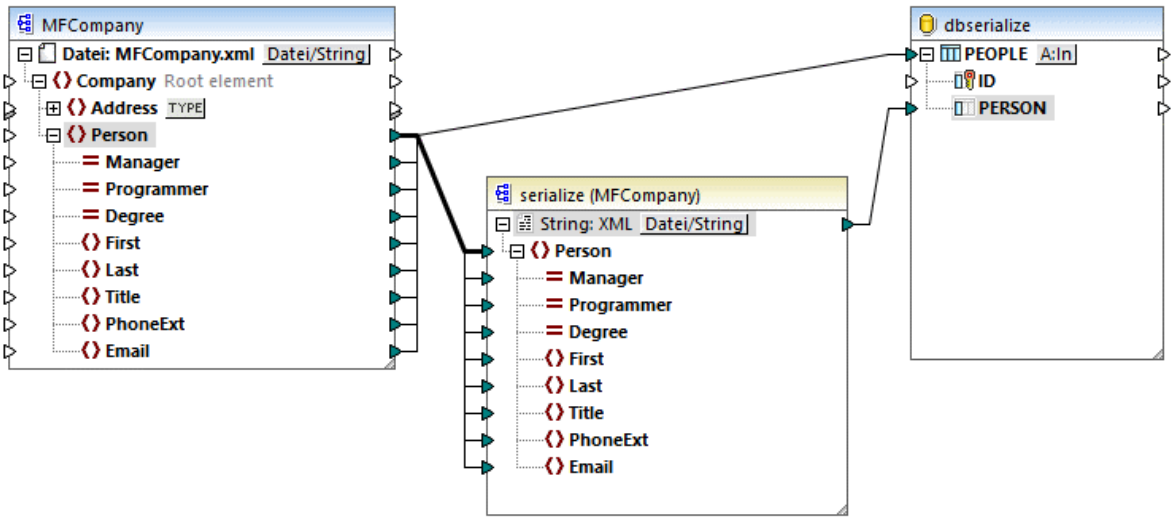


6. Fügen Sie die SQLite-Datenbankkomponente aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\\dbserialize.db** zum Mapping-Bereich hinzu. (Verwenden Sie zum Hinzufügen der Datenbankkomponente den Menübefehl **Einfügen | Datenbank**, siehe auch Herstellen einer Verbindung zu einer Datenbank 167). Wenn Sie aufgefordert werden, ein Datenbankobjekt einzufügen, wählen Sie die Tabelle `PEOPLE` aus.

 © 2016-2022 Altova GmbH

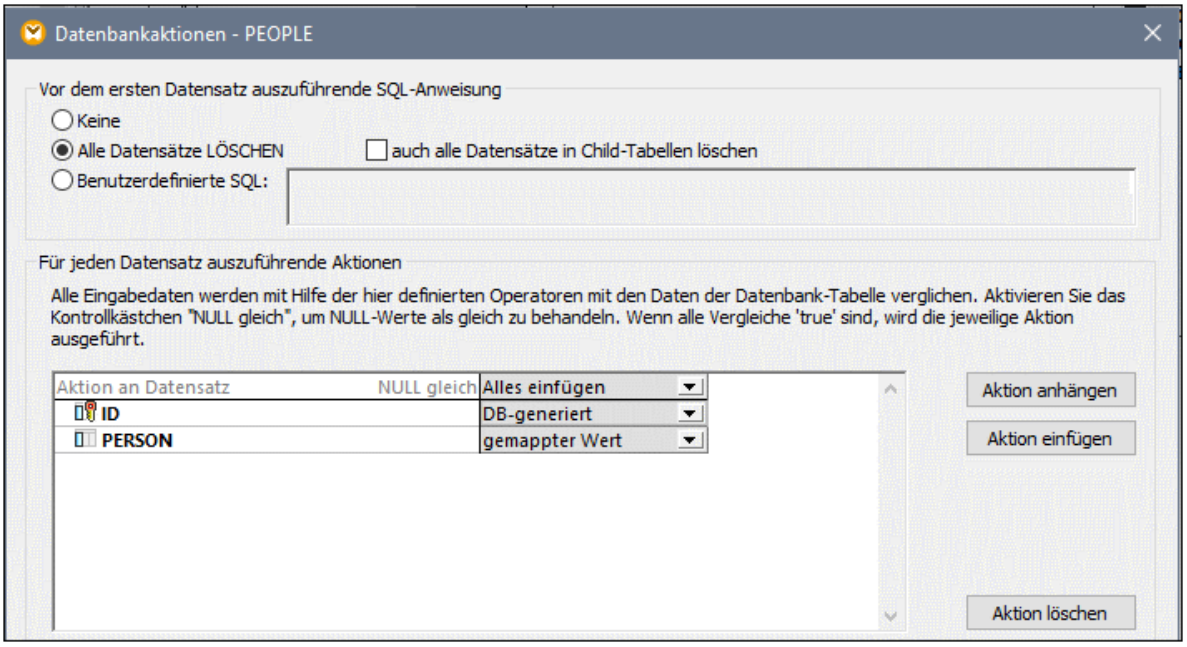


7. Verbinden Sie die Komponenten, wie unten gezeigt, miteinander. Auf der linken Seite des Mappings wird das Element `<Person>` auf die Serialisierungskomponente gemappt. Auf der rechten Seite des Mappings wird der serialisierte String-Wert in die Spalte `PERSON` der Datenbanktabelle `PEOPLE` eingefügt. Schließlich erstellt MapForce aufgrund der Verbindungslinie zwischen `<Person>` und der Tabelle `PEOPLE` für jedes `<Person>`-Element einen neuen Datensatz.



8. Klicken Sie auf die **A:In**-Schaltfläche der Datenbankkomponente und gehen Sie folgendermaßen vor:
   a. Aktivieren Sie die Option **Alle Datensätze löschen**. Dadurch werden alle vorhandenen Datensätze zur Mapping-Laufzeit aus der Datenbank gelöscht, bevor neue eingefügt werden.

b. Aktivieren Sie die Option **DB-generiert** neben der Spalte **ID**. Damit wird sichergestellt, dass die ID des Datensatzes von der Datenbank generiert wird. Beachten Sie, dass die Option **DB-generiert** nur angezeigt wird, wenn die Spalte diese Option unterstützt. Bei Spalten, bei denen es sich nicht um eine ID oder ein automatisch inkrementiertes Feld handelt, steht stattdessen die Option **max+1** zur Verfügung. Mit dieser Option wird überprüft, welcher Wert in dieser Spalte der bereits vorhandene Maximalwert ist und es wird die nächste verfügbare um 1 erhöhte Ganzzahl eingefügt.

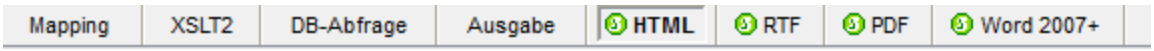


Sie haben nun ein Mapping-Design erstellt, das Daten in einen String serialisiert. Wenn Sie auf das Register **Ausgabe** klicken, zeigt die SQL-Abfrage für die Vorschau an, dass für jedes `<Person>`-Element in der XML-Datei, wie beabsichtigt, ein separater Datensatz in die Datenbank eingefügt wird.

© 2016-2022 Altova GmbH

8.7 StyleVision-Ausgabefenster

In Mappings, in denen die Zielkomponente eine XML- oder XBRL-Komponente ist, kann eine Vorschau auf die Mapping-Ausgabe in HTML-, RTF-, PDF-, Word 2007+- und Textdokumenten angezeigt und gespeichert werden, wenn Altova StyleVision auf Ihrem Rechner installiert ist. Wenn Sie die Enterprise Edition von StyleVision verwenden, werden in diesen Vorschaudokumenten auch Diagramme dargestellt. Wenn ein Mapping die Vorschau in einem dieser Formate unterstützt, stehen neben dem Fenster **Ausgabe** weitere Fenster zur Verfügung (*siehe Abbildung unten*).



Achtung:

- Wenn StyleVision Professional installiert ist, kann eine Vorschau auf die **HTML**- und **RTF**- und **Text**-Ausgabe angezeigt werden.
- Wenn StyleVision Enterprise installiert ist, kann eine Vorschau auf die **HTML**-, **RTF**-, **PDF**-, **Word 2007+**- und **Text**-Ausgabe angezeigt werden.
- Um eine Vorschau auf die Mapping-Ausgabe im PDF-Format zu sehen, müssen Java, Acrobat Reader und FOP (Formatting Objects Processor) Version 0.93 oder 1.0. installiert sein. FOP wird zusammen mit StyleVision installiert, es sei denn, Sie haben dies bei der Installation von StyleVision nicht mit installiert.
- In der 64-Bit Edition von MapForce werden die Word 2007+ und die RTF-Vorschau als nicht eingebettete Applikationen geöffnet.
- Wenn Ihr Mapping Komponenten enthält, die sowohl als Quell- als auch als Zielkomponente (Weiterleitungskomponente) fungieren, ist die StyleVision-Vorschau nur für die Komponenten möglich, für die die **Vorschauschaltfläche** der Komponente aktiviert wurde. Nähere Informationen zu derartigen Mappings finden Sie unter Verkettete Mappings 1299.

Um in den StyleVision-Ausgabefenstern eine Vorschau auf Daten aus einem Mapping zu sehen, werden folgende Voraussetzungen benötigt:

- Altova StyleVision muss auf Ihrem Rechner entweder als Standalone Edition oder als Teil des Altova MissionKit installiert sein.
- Die Zielkomponente muss mit einer StyleVision Power Stylesheet (SPS)-Datei verknüpft sein. Die Stylesheet-Datei kann mit StyleVision erstellt oder bearbeitet werden. Sie können das Stylesheet nicht direkt in MapForce bearbeiten oder ändern, doch können Sie es über MapForce in StyleVision öffnen. Sobald das Stylesheet fertig ist, können Sie es, wie unten gezeigt, einer MapForce-Zielkomponente zuweisen.

StyleVision-Ausgabefensterkonfiguration

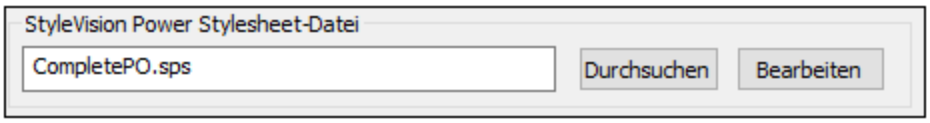
In der folgenden Anleitung erfahren Sie, wie Sie die StyleVision-Ausgabefenster einrichten.

Zuweisen eines StyleVision Power Stylesheet zu einer Zielkomponente

Um einer Zielkomponente eine SPS-Datei zuzuweisen, gehen Sie folgendermaßen vor:

1. Erstellen Sie die gewünschte Stylesheet-Datei in StyleVision. Stellen Sie sicher, dass Sie als Quelle dasselbe XML-Schema wie für die MapForce-Komponente verwenden.
2. Klicken Sie in MapForce mit der rechten Maustaste auf die XML-oder XBRL-Zielkomponente in MapForce und wählen Sie **Eigenschaften**.

3. Navigieren Sie im Dialogfeld **Komponenteneinstellungen** neben **StyleVision Power Stylesheet-Datei** zu der zuvor erstellten Stylesheet-Datei (*siehe Abbildung unten*).



Anmerkung: Der Pfad zur StyleVision Power Stylesheet-Datei kann absolut oder relativ sein, siehe Verwendung relativer und absoluter Pfade[84].

Speichern der mit StyleVision generierten Ausgabe
Genau wie das Ergebnis jedes anderen Mappings können Sie auch die mit StyleVision generierte Ausgabe in einer Datei speichern. Klicken Sie auf die Symbolleisten-Schaltfläche (**Generierte Ausgabe speichern**) oder gehen Sie zum Menü **Ausgabe** und klicken Sie auf **Ausgabedatei speichern**.

Automatisieren der Generierung unterschiedlicher Formate mit Altova-Produkten
Wenn Ihr Mapping (entweder auf demselben Rechner oder auf einem anderen Rechner oder sogar einer anderen Plattform) automatisch HTML-, PDF-, RTF-, Word 2007+- und Textdateien generieren soll, können Sie dies mit Hilfe von MapForce Server oder StyleVision Server tun. Es handelt sich hierbei um separat lizenzierte Produkte, die die Funktionalitäten von MapForce bzw. StyleVision erweitern. Die einzelnen Applikationen spielen in diesem Szenario jeweils ihre eigene spezifische Rolle:

- MapForce - dient zum Erstellen eines Mappings (`.mfd`-Datei), mit dem die Inputs und Outputs für die Datentransformation definiert werden (z.B. Datenbank in XML)
- MapForceServer - führt das ausführbare Mapping (`.mfx`-Datei) über die Befehlszeile oder eine API (entweder auf demselben oder einem anderen Betriebssystem) aus
- StyleVision - dient zum Erstellen des Stylesheet (`.sps`-Datei), anhand dessen die Mapping-Ausgabe in HTML, PDF, RTF, Word 2007+ und Textdateien transformiert wird.
- StyleVision Server - führt das `.sps`-Stylesheet, das die Mapping-Ausgabe in das gewünschte Format transformiert, aus. Dies erfolgt über die Befehlszeile oder eine API (auf demselben oder einem anderen Betriebssystem).
- Sowohl StyleVision Server als auch MapForce Server können optional unter Verwaltung durch (den separat lizenzierten) FlowForce Server ausgeführt werden. In diesem Szenario können MapForce Mappings und StyleVision-Transformationen nach Plan, d.h. vollständig automatisiert, entweder über einen Trigger, oder auf Abruf ausgeführt werden. D.h. diese MapForce Mappings und StyleVision-Transformationen können vollständig automatisiert werden.

Beispiele

Im Beispiel unten (**`MapForceExamples\CompletePO.mfd`**) sehen Sie die Ausgabe im StyleVision-Ausgabefenster **HTML**. Mit diesem Mapping wird eine Bestellung im XML-Format erzeugt. Klicken Sie mit der rechten Maustaste auf die Zielkomponente, wählen Sie **Eigenschaften** und Sie sehen, dass der Komponente eine `.sps`-Datei zugewiesen wurde.

Wenn Sie auf das Register **HTML** klicken, sehen Sie die folgende Ausgabe:

© 2016-2022 Altova GmbH



TO: Mrs./Mr. Ted Little

Long Way
Los-Angeles
CA 34424
Our Customer Identifier: ID-3

Order Date: _____

Shipping Date: _____

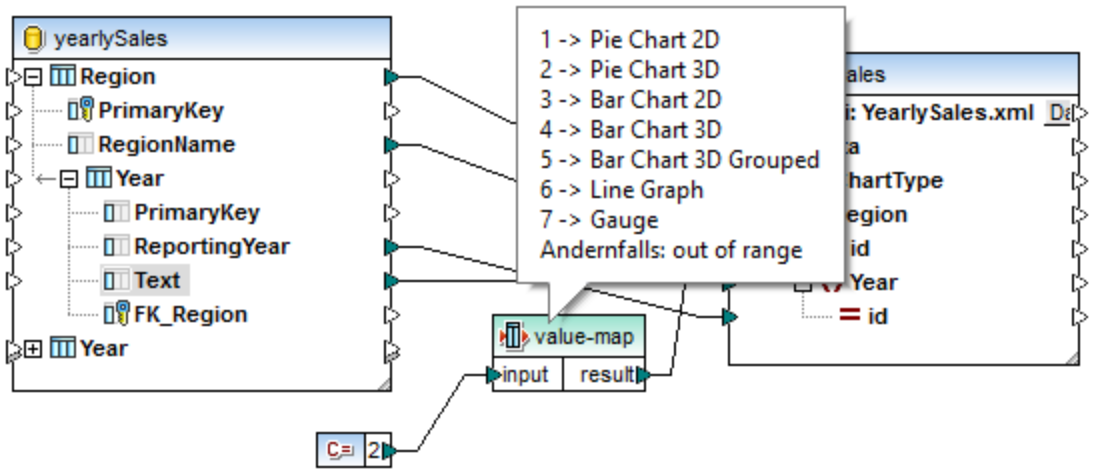
Item	Quantity	Name	Unit Price ($)	Total ($)
3	5	Pants	34	170
1	17	T-Shirt	25	425
				595

Other Comments: _____

_____ _____
Authorized Signature Date

Mapping | XSLT2 | DB-Abfrage | Ausgabe | HTML | RTF | PDF | Word 2007+

Ein weiteres Beispiel ist **`Tutorial\YearlySales.mfd`**. Das diesem Mapping zugewiesene Stylesheet wurde in StyleVision so entworfen, dass die Diagrammart durch Änderung des Werts des Elements `ChartType` gewählt werden kann. Auf diese Art kann der Diagrammtyp direkt über das Mapping geändert werden. So können Sie den Standardwert der Konstante in jeden Wert von 1 bis 7 ändern. Wenn Sie den Cursor über die Komponente **`value-map`** platzieren, werden die möglichen Werte angezeigt (*siehe Abbildung unten*).



Der Standardwert der Konstante ist "2", wodurch in der Ausgabe ein 3-D-Tortendiagramm generiert wird. Um andere Diagrammtypen anzuzeigen, ändern Sie den Wert in einen beliebigen anderen zulässigen Wert und klicken Sie auf das Register **Ausgabe**, um die Änderungen zu sehen.

© 2016-2022 Altova GmbH

8.8 Generieren von Mapping-Dokumentation

Sie können zu praktisch jedem Mapping eine ausführliche Dokumentation im HTML-, Microsoft Word- (.doc) oder RTF-Format generieren. Wenn StyleVision installiert ist, können Sie zusätzlich dazu auch Dokumentation im PDF-Format generieren.

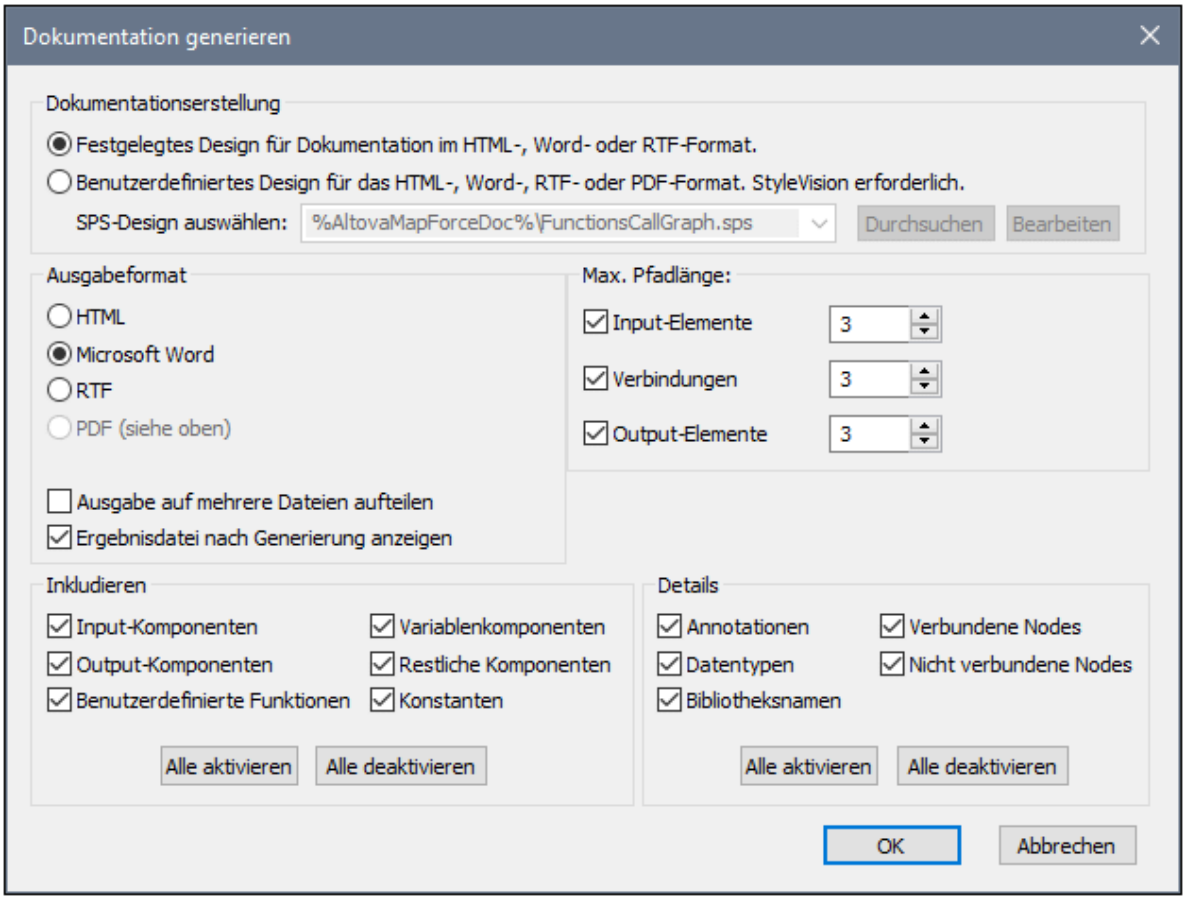
Voraussetzungen:

- Um Dokumentation im MS Word-Format zu generieren, muss Microsoft Word 2000 oder höher installiert sein.
- Um Dokumentation im PDF-Format zu generieren oder das Design der generierten Dokumentation anzupassen, müssen Sie StyleVision auf Ihrem Rechner installiert haben.

Standardmäßig wird die Dokumentation in einem festgelegten Design, in dem Sie grundlegende Optionen wie zu inkludierende Komponenten, die Tiefe des angezeigten Pfads und andere Einstellungen konfigurieren können, generiert. Wenn StyleVision installiert ist, können Sie zusätzlich dazu eine Reihe inkludierter StyleVision Power Stylesheets (SPS)-Dateien nutzen oder mit StyleVision sogar ein eigenes Design erstellen.

So generieren Sie Mapping-Dokumentation:

1. Klicken Sie im Menü **Datei** auf **Dokumentation generieren.** Daraufhin wird das Dialogfeld "Dokumentation generieren" geöffnet.



2. Wählen Sie die gewünschten Einstellungen aus und klicken Sie auf **OK**.

Dokumentationserstellung

- Wählen Sie die Option "Festgelegtes Design für..", um die vordefinierte Dokumentationsvorlage zu verwenden.
- Wählen Sie die Option "Für ein benutzerdefiniertes Design...", um ein mit StyleVision erstelltes StyleVision Power Stylesheet zu verwenden. Die SPS-Dateien stehen im Ordner **...\Dokumente\Altova\MapForce2023\Documentation\MapForce** zur Verfügung. Nähere Informationen dazu finden Sie unter Vordefinierte StyleVision Power Stylesheets 1401.
- Klicken Sie auf **Durchsuchen**, um zu einer vordefinierten SPS-Datei zu navigieren.
- Klicken Sie auf **Bearbeiten**, um StyleVision zu starten und das ausgewählte SPS in einem StyleVision-Fenster zu öffnen.

Ausgabeformat

- Wählen Sie eines der folgenden Ausgabeformate aus: HTML, Microsoft Word, RTF oder PDF. Bei Generierung der Dokumentation mit dem festgelegten Design werden Microsoft Word-Dokumente mit der Dateierweiterung **.doc** erzeugt. Bei Generierung unter Verwendung eines StyleVision SPS werden Dokumente mit der Dateierweiterung **.docx** erzeugt. Für das PDF-Ausgabeformat wird StyleVision benötigt und dieses Format steht nur zur Verfügung, wenn Sie Sie ein StyleVision SPS ausgewählt haben.

© 2016-2022 Altova GmbH

- Wählen Sie die Option **Ausgabe auf mehrere Dateien aufteilen**, wenn Sie mehrere Dopkumentationsdateien, und zwar eines für jede einzelne Komponente wie Input- oder Output-Komponenten generieren möchten. Wenn Sie ein festgelegtes Design verwenden, werden automatisch Links zwischen mehreren Dokumenten erstellt.
- Wenn die Option **Ergebnisdatei nach Generierung anzeigen** aktiv ist, werden die generierten Dateien im Standard-Browser bzw. der jeweiligen Standard-Applikation geöffnet.

Maximale Pfadlänge

Über diese Option können Sie die maximale Pfadlänge definieren, die für Input- oder Output-Datenelemente oder Verbindungen angezeigt werden soll. So würde etwa der Datenelementpfad bei einer Standardlänge 3 z.B. als **.../ShortPO/LineItems/LineItem** angezeigt.

Inkludieren

Hier können Sie definieren, welche Komponenten in der generierten Dokumentation aufscheinen sollen.

Details

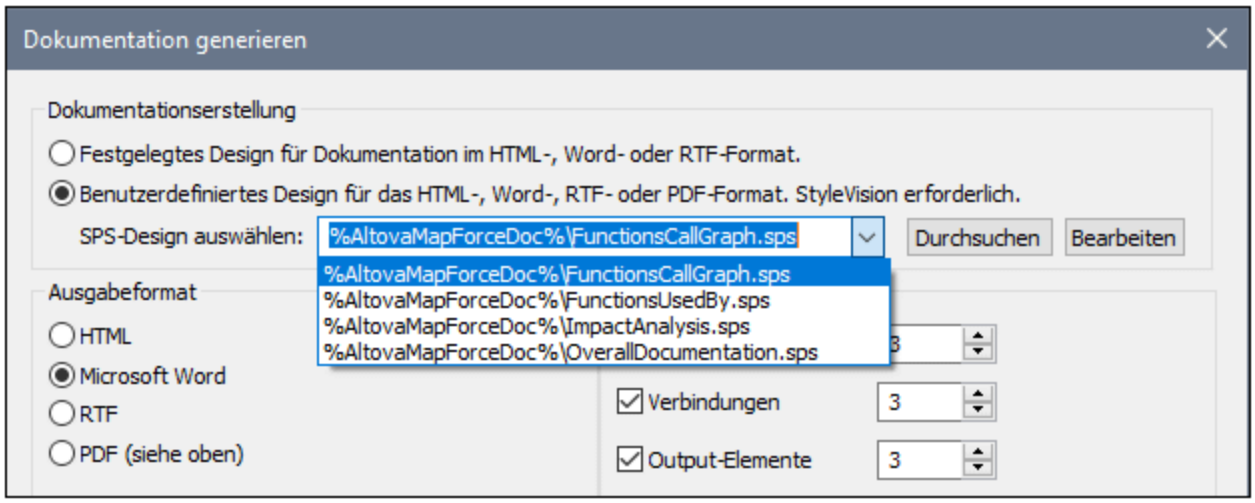
Mit Hilfe dieser Optionen können Sie den Detailliertheitsgrad der generierten Dokumentation festlegen. Mit der Option **Bibliotheksname** wird das "core"-Präfix für Funktionen inkludiert.

8.8.1 Vordefinierte StyleVision Power Stylesheets

Wenn StyleVision auf Ihrem Rechner installiert ist, können Sie Mapping-Dokumentation durch Auswahl einer der vordefinierten StyleVision Power Stylesheet (SPS)-Dateien als Vorlage anstelle des vordefinierten festgelegten Designs generieren. Es stehen die folgenden vordefinierten SPS Stylesheets zur Verfügung:

- **FunctionCallGraph.sps** - zeigt den Call-Graphen des Hauptmappings und aller benutzerdefinierter Funktionen an.
- **FunctionsUsedBy.sps** - zeigt an, welche Funktionen im Mapping direkt oder indirekt verwendet werden.
- **ImpactAnalysis.sps** - listet alle Quell- und Ziel-Nodes sowie den Weg über verschiedene Funktionen zum Ziel-Node auf.
- **OverallDocumentation.sps** - zeigt alle Nodes, Verbindungen, Funktionen und Ziel-Nodes an. Diese Vorlage bietet den größten Detailliertheitsgrad und ist mit dem vordefinierten festgelegten Design identisch.

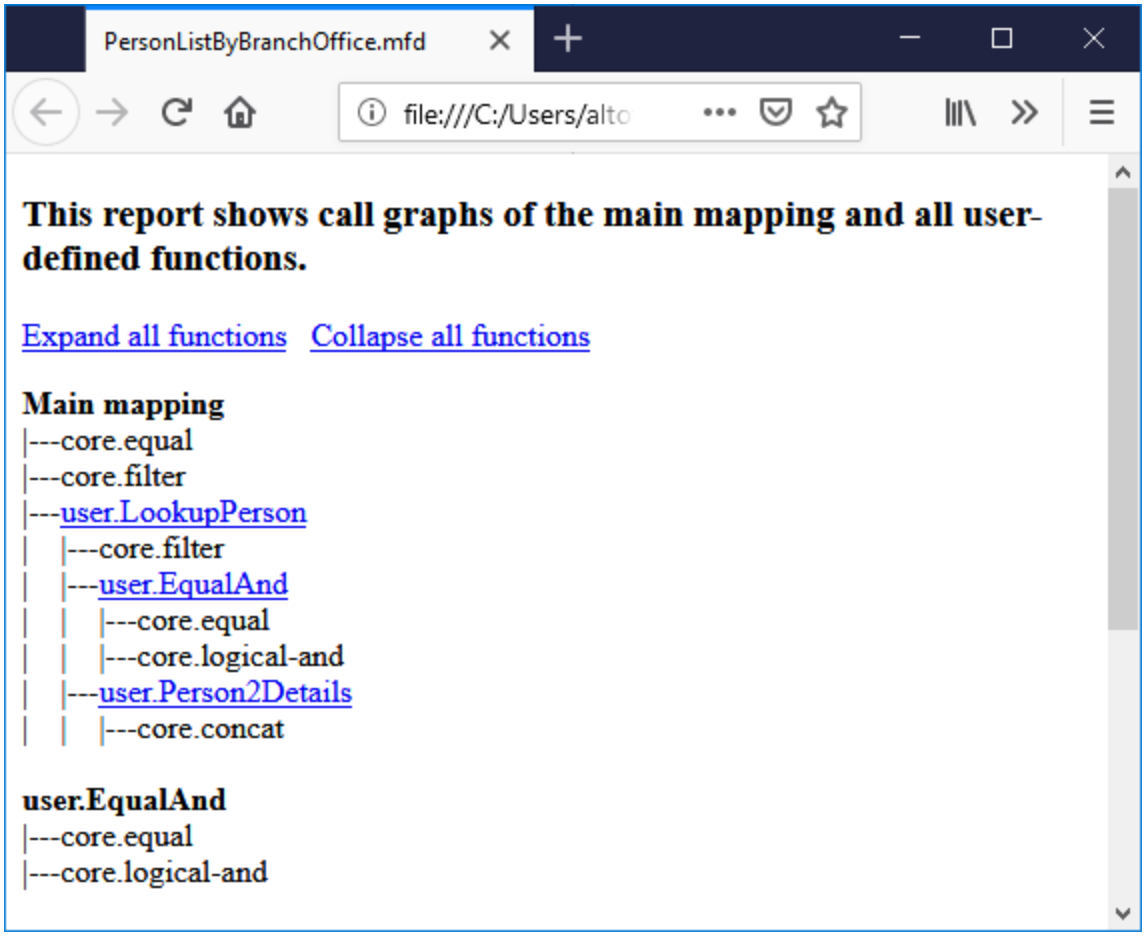
Sie können das gewünschte Stylesheet vor jeder Dokumentationsgenerierung, wie unten gezeigt, auswählen. Die Dateien befinden sich im Ordner **...\MapForce2023\Documentation\MapForce**.



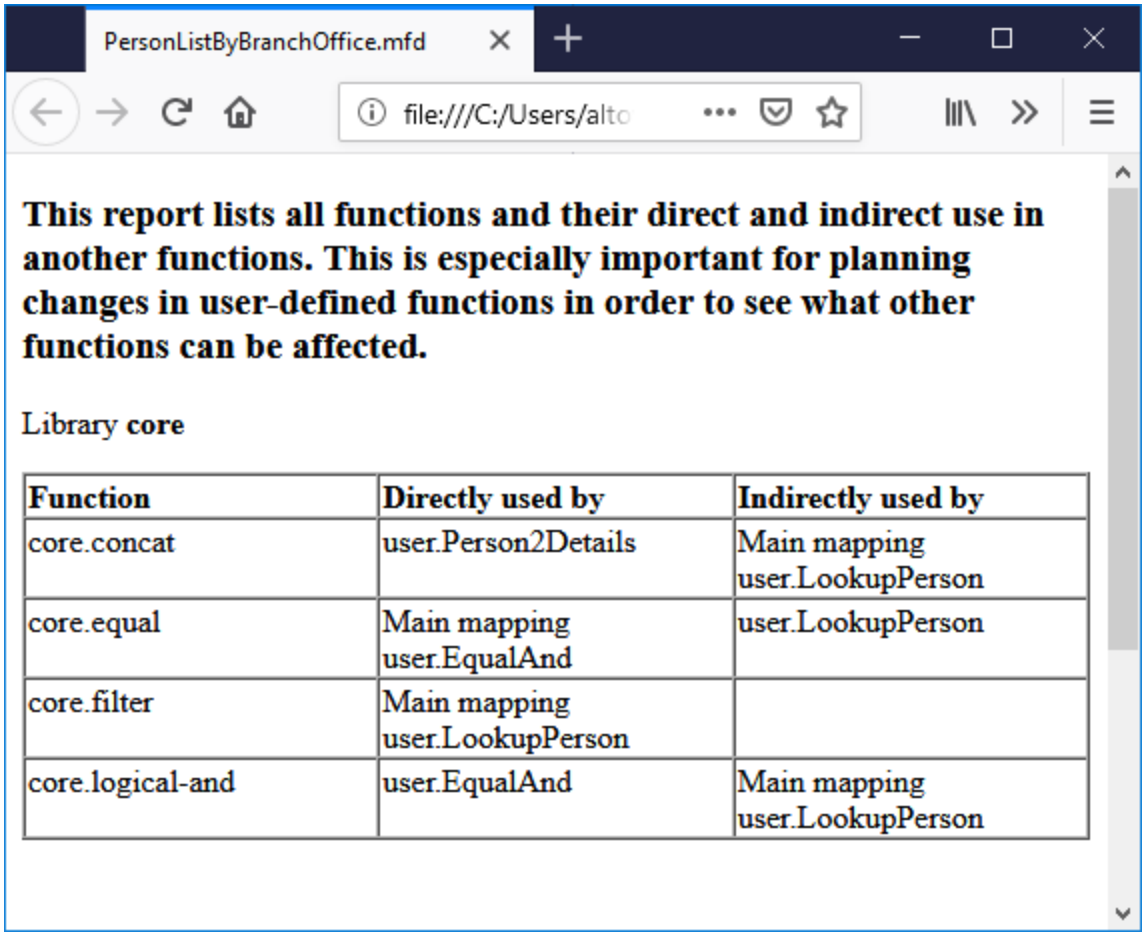
In den Beispielen weiter unten sehen Sie, welche Ausgabe von den einzelnen Stylesheets jeweils erzeugt wird. Die Beispiele wurden anhand eines der mit MapForce installierten Demo-Mappings, **PersonListByBranchOffice.mfd,** generiert. Zwar sehen Sie in diesem Beispielen die HTML-Ausgabe, doch ist das Layout auch bei anderen Formaten ähnlich. Nähere Informationen zur Erstellung oder zum Anpassen von SPS-Dateien finden Sie unter Benutzerdefinierte Stylesheets 1406.

 © 2016-2022 Altova GmbH

Stylesheet "FunctionCallGraph.sps"



Stylesheet "FunctionsUsedBy.sps"



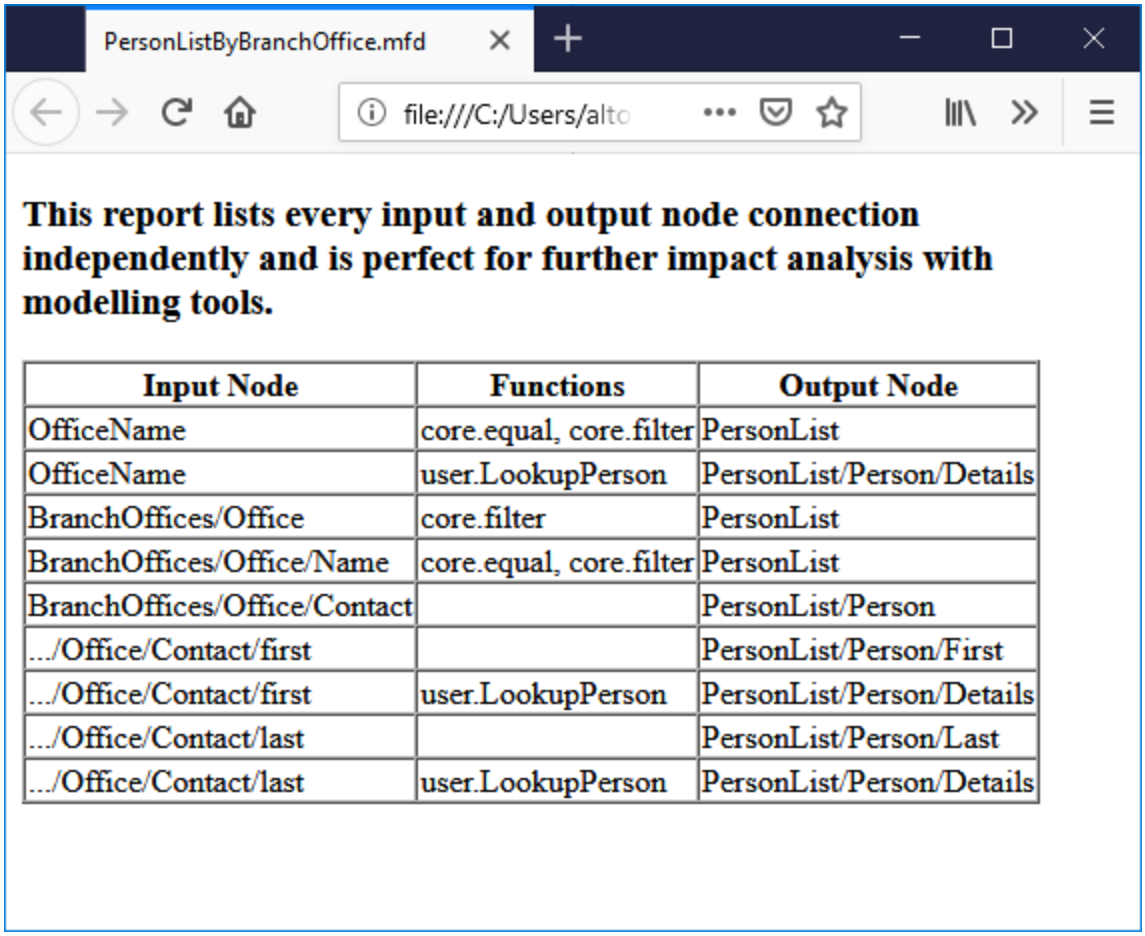
This report lists all functions and their direct and indirect use in another functions. This is especially important for planning changes in user-defined functions in order to see what other functions can be affected.

Library **core**

Function	Directly used by	Indirectly used by
core.concat	user.Person2Details	Main mapping user.LookupPerson
core.equal	Main mapping user.EqualAnd	user.LookupPerson
core.filter	Main mapping user.LookupPerson	
core.logical-and	user.EqualAnd	Main mapping user.LookupPerson

© 2016-2022 Altova GmbH

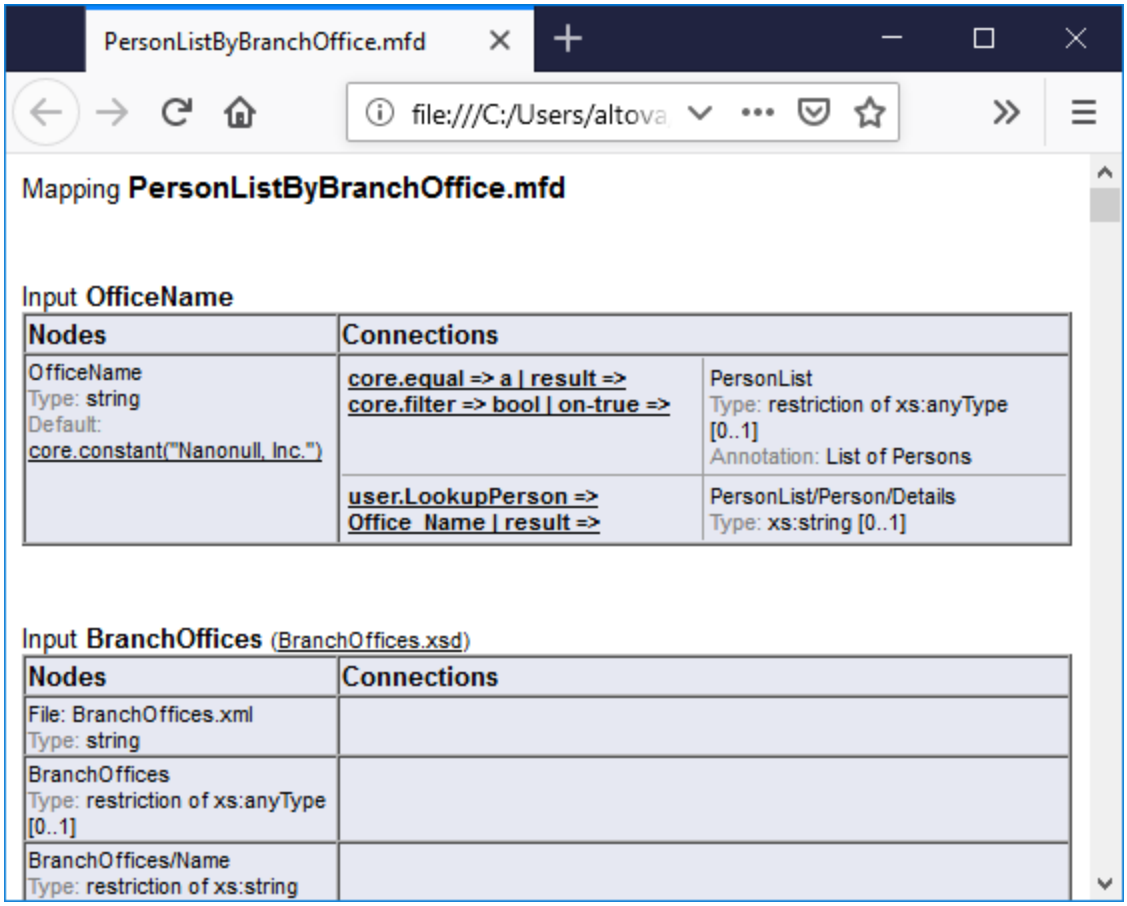
Stylesheet "ImpactAnalysis.sps"



This report lists every input and output node connection independently and is perfect for further impact analysis with modelling tools.

Input Node	Functions	Output Node
OfficeName	core.equal, core.filter	PersonList
OfficeName	user.LookupPerson	PersonList/Person/Details
BranchOffices/Office	core.filter	PersonList
BranchOffices/Office/Name	core.equal, core.filter	PersonList
BranchOffices/Office/Contact		PersonList/Person
.../Office/Contact/first		PersonList/Person/First
.../Office/Contact/first	user.LookupPerson	PersonList/Person/Details
.../Office/Contact/last		PersonList/Person/Last
.../Office/Contact/last	user.LookupPerson	PersonList/Person/Details

Stylesheet "OverallDocumentation.sps"



8.8.2 Benutzerdefinierte Stylesheets

Neben dem vordefinierten festgelegten Design können Sie mit StyleVision (https://www.altova.com/de/stylevision) auch benutzerdefinierte Stylesheets für die generierten Mapping-Dokumentation generieren. Sie können auch jedes der vordefinierten Stylesheets[1401] ändern, z.B. indem Sie die Schriftarten und andere Stile ändern.

Ein benutzerdefiniertes Design ist ein StyleVision Power StyleSheet (SPS). Der Vorteil der Verwendung eines SPS bei der Generierung von Dokumentation ist, dass Sie damit vollständige Kontrolle über das Design der Dokumentation haben.

Um eine benutzerdefinierte SPS-Datei zu erstellen, benötigen Sie die folgenden Dinge:

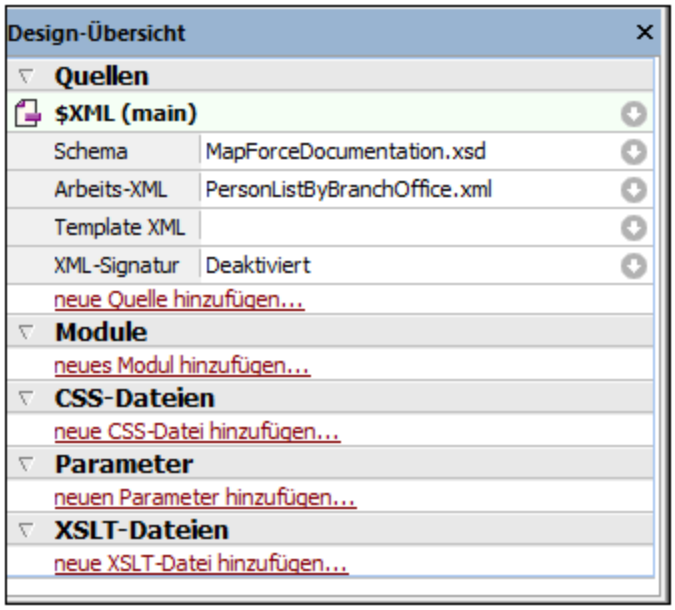
1. das XML-Schema, das die Struktur der generierten MapForce-Dokumentation definiert. Dieses Schema hat den Namen **MapForceDocumentation.xsd** und ist im Lieferumfang Ihrer MapForce-Installation enthalten. Es befindet sich im Ordner **...\Documents\Altova\MapForce2023\Documentation\MapForce**. Beachten Sie, dass das Schema

 © 2016-2022 Altova GmbH

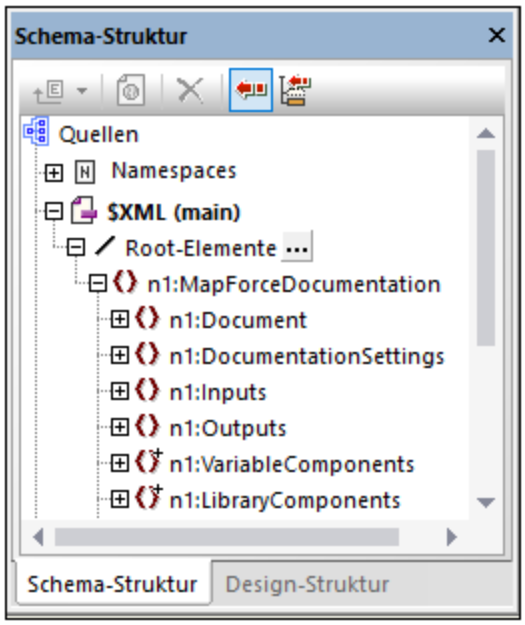
MapForceDocumentation.xsd die Datei **Documentation.xsd** aus dem übergeordneten Ordner inkludiert.

2. einige Beispieldaten, anhand welcher Sie das benutzerdefinierte Design testen und in einer Vorschau anzeigen können. Für die Beispieldaten können Sie die folgende XML-Datei verwenden: **...\Documents\Altova\MapForce2023\Documentation\MapForce\SampleData\PersonListByBranchOffice.xml**.

Die oben erwähnten Dateien müssen in StyleVision im Fenster Design-Übersicht referenziert werden, z.B.:



In StyleVision wird das Design durch Ziehen der Nodes aus dem Fenster "Schema-Struktur" in den Designbereich und Zuweisen von Stilen und Eigenschaften zu diesen Nodes erstellt.



Sie können auch zusätzliche Komponenten wie Links und Bilder zum SPS-Design hinzufügen. Um eine Vorschau auf das Design in einem bestimmten Format zu sehen, klicken Sie auf eines der folgenden Register: **HTML**, **RTF**, **PDF** oder **Word 2007+**. Nähere Informationen dazu finden Sie in der StyleVision-Dokumentation (https://www.altova.com/de/documentation).

© 2016-2022 Altova GmbH

9 Debugger

MapForce enthält für die eigene BUILT-IN-Transformationssprache einen Mapping Debugger. Mit dem Mapping Debugger können Sie Folgendes tun:

- Anzeigen und Analysieren der vom Mapping erzeugte Werte auf jeder einzelnen Konnektor-Ebene
- Markierung des Kontext (der Node-Gruppe) im Mapping, anhand desses ein bestimmter Wert erzeugt wird
- Schrittweise Ausführung eines Mappings, um zu sehen, wie MapForce die einzelnen Werte in Echtzeit verarbeitet oder berechnet und Anzeige einer Vorschau auf die Mapping-Ausgabe während der Generierung der Ausgabe
- Setzen von Meilensteinen (Breakpoints), an denen die Ausführung des Mappings angehalten und an denen der/die aktuell verarbeitete(n) Wert(e) angezeigt werden soll(en)
- Anzeige des Verlaufs für von einem Konnektor verarbeitete Werte ab dem Beginn der Mapping-Ausführung bis zur aktuellen Ausführungsposition

Der Mapping Debugger steht zur Verfügung, wenn als Transformationssprache für das Mapping BUILT-IN ausgewählt wurde. Wenn Sie den Debugger für ein Mapping, das für eine andere Sprache erstellt wurde, starten, werden Sie aufgefordert, zu BUILT-IN zu wechseln. Sie können ein Mapping auch über den Menübefehl **Ausgabe | Built-in-Ausführungsprozessor** in BUILT-IN konvertieren. In beiden Fällen ist die Konvertierung in BUILT-IN erfolgreich, wenn das Mapping keine Komponenten enthält, die in der Sprache BUILT-IN nicht zur Verfügung stehen (wie z.B. XSLT-Funktionen).

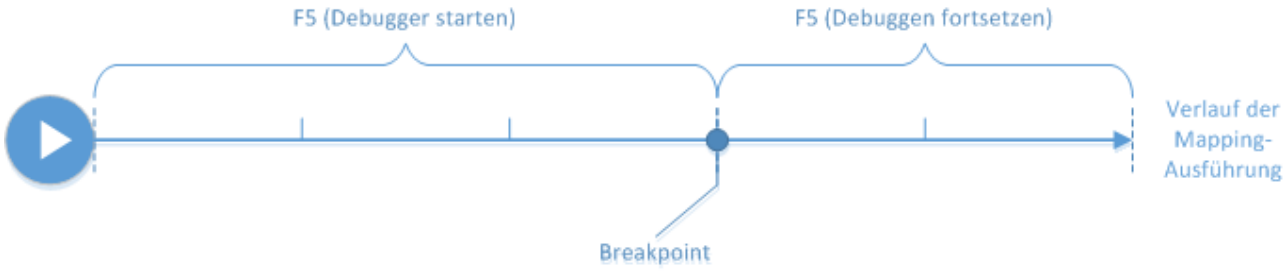
Der MapForce Debugger unterscheidet sich insofern von einem herkömmlichen Debugger, als er Ihren Programmcode nicht Zeile für Zeile durchläuft (da Sie mit MapForce ja keinen Code schreiben). Stattdessen untersucht der Debugger die anhand der Mappings in Ihrem Design erzeugten Ergebnisse von MapForce-generiertem Code. Genauer gesagt, protokolliert der Debugger alle Werte, die über ihre Input- und Output-Konnektoren von und an Mapping-Komponenten übergeben werden. Die protokollierten Werte stehen anschließend direkt im Mapping oder über eigene Fenster zur Analyse zur Verfügung.

In den folgenden Abschnitten werden verschiedene Debugging-Methoden gezeigt.

Die Debugger-Einstellungen finden Sie im Dialogfeld Optionen[1541]. Die Liste der Debugger-Befehle finden Sie unter Debuggen[1532].

Debuggen mit Breakpoints

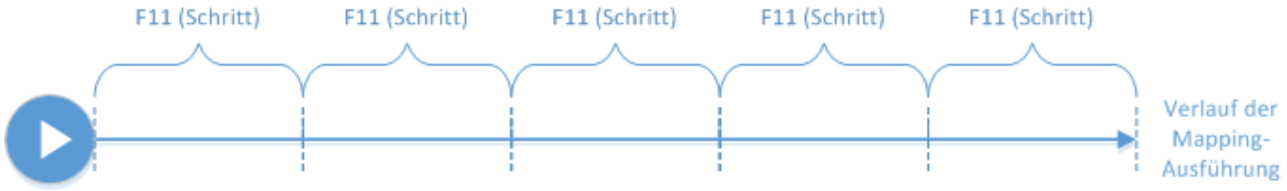
Wenn Sie die Debugger-Ausführung an einer bestimmten Stelle im Mapping anhalten müssen, können Sie ähnlich wie in einer herkömmlichen Entwicklungsumgebung Breakpoints setzen. Der Unterschied ist, dass die Breakpoints nicht zu einer Codezeile, sondern zu einem Input- oder Output-Konnektor einer Mapping-Komponente hinzugefügt werden. Sie können auch Bedingungen zu Breakpoints hinzufügen (Dies kann sich z.B. als nützlich erweisen, wenn Sie die Ausführung nur dann anhalten möchten, wenn die definierte Bedingung zutrifft).



Sie können an den gewünschten Konnektoren Breakpoints definieren und das Mapping bis zum ersten Breakpoint ausführen, anschließend zum nächsten gehen usw. Auf diese Art können Sie den Mapping-Kontext und die mit den ausgewählten Konnektoren in Zusammenhang stehenden Werte analysieren. In diesem Szenario können Sie die Ausführung mit Hilfe der Debugger-Befehle **Einsteigen, Aussteigen, Überspringen** und **Minimaler Schritt** beschleunigen oder verlangsamen. Mit Hilfe dieser Befehle können Sie Mapping-Abschnitte überspringen oder umgekehrt, Abschnitte eines Mappings bei Bedarf genauer analysieren.

Schrittweises Debuggen

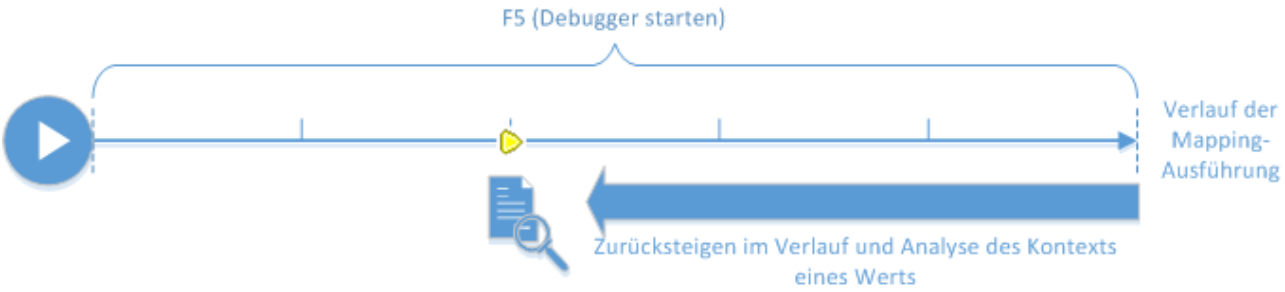
Sie können ein Mapping Schritt für Schritt debuggen und dabei den Mapping-Kontext und die mit den einzelnen Schritten in Zusammenhang stehenden Werte analysieren. Dieses Szenario ist dem zuvor beschriebenen insofern ähnlich, als Sie damit die Ausführung mit den Befehlen **Einsteigen, Aussteigen, Überspringen** und **Minimaler Schritt** beschleunigen oder verlangsamen können.



Analysieren der Log-Werte

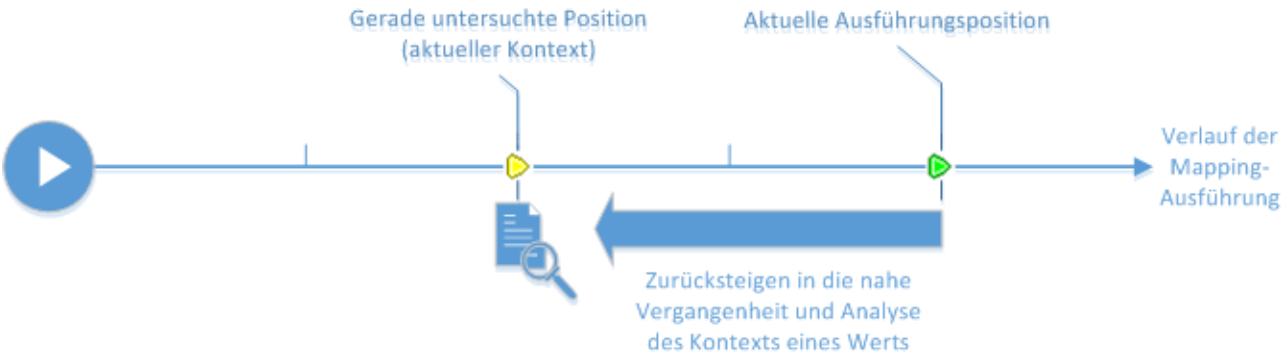
Sie können MapForce so konfigurieren, dass es das gesamte Log aller Werte, die von allen Konnektoren beim Debuggen eines Mappings verarbeitet wurden (das Verlaufsprotokoll der Ablaufverfolgung) aufbewahrt. Bei speicherintensiven Mappings ist es oft nicht sinnvoll, den gesamten Verlauf zu speichern, daher kann diese Option bei Bedarf auch deaktiviert werden. Wenn die Option aktiviert ist, können Sie das gesamte Protokoll der von jedem einzelnen Konnektor bis zur aktuellen Ausführungsposition verarbeiteten Werte analysieren. Außerdem können Sie den mit einem bestimmten Wert verknüpften Mapping-Kontext in MapForce wieder aufrufen, um zu verstehen, warum dieser Wert erzeugt wurde.

© 2016-2022 Altova GmbH



Setzen des Kontexts auf einen mit der aktuellen Ausführungsposition in Zusammenhang stehenden Wert

Es ist auch möglich, während sich der Debugger an einer bestimmten Ausführungsposition im Mapping befindet, den Mapping-Kontext eines vergangenen Werts im Verhältnis zur aktuellen Ausführungsposition zu analysieren (dies ist, als würden Sie einen Schritt in die Vergangenheit machen):



Ein Kontext erklärt, warum ein Wert berechnet wird, d.h. er beschreibt, wie ein bestimmter Wert im Mapping generiert wurde. Normalerweise ist der Kontext die aktuelle Ausführungsposition, doch können Sie als Kontext auch einen Kontext in der nahen Vergangenheit des Mappings definieren. Wenn ein bestimmter Wert als Kontext definiert wird, markiert MapForce die damit in Zusammenhang stehenden Nodes im Mapping, zeigt neben den Mapping-Konnektoren Tipps und in anderen Debugger-Fenstern (**Werte**, **Kontext** und **Breakpoints**) zusätzliche Informationen dazu an.

Nach Untersuchung des jeweiligen Mapping-Kontext können Sie den Kontext anschließend wieder zurück auf die aktuelle Ausführungsposition setzen.

Einschränkungen

- Wenn MapForce ein Mapping ausführt, optimiert das Programm Code eventuell intern (z.B. durch Zwischenspeichern von Daten im Cache oder durch Berechnung von Zwischenergebnissen an beliebig gewählten Stellen). Dadurch können bestimmte Konnektoren (und somit auch Breakpoints) möglicherweise beim Debuggen nicht erreicht werden. In diesem Fall wird eine entsprechende Meldung angezeigt. Beachten Sie, dass sich die MapForce Codeoptimierungen (und folglich auch das

Verhalten, das der Debugger untersucht) von Version zu Version ändern kann, auch wenn die Mapping-Ausgabe für ein bestimmtes Mapping dieselbe ist.

- Der Debugger kann die Ausgabegenerierung immer nur für eine Zielkomponente debuggen. Falls in einem Mapping mehrere Zielkomponenten vorhanden sind, müssen Sie auswählen, welche davon vom Debugger ausgeführt werden soll.
- Zur Zeit wird das Debuggen von Datenbanktabellenaktionen von Datenbankkomponenten (wie z.B. "Alle einfügen", "Aktualisieren wenn", usw.) nicht unterstützt.
- Für die folgenden Elemente können keine Breakpoints hinzugefügt werden: Konstanten, die Funktion **`core | position`**, Nachfahren-Datenelemente von "Alles kopieren"-Verbindungen, Parameter von inline gesetzten benutzerdefinierten Funktionen.

© 2016-2022 Altova GmbH

9.1 Debugger-Vorbereitung

Vor allem bei großen Datenmappings, für deren Ausführung viel Arbeitsspeicher benötigt wird, müssen Sie das Debuggen vorbereiten. Dies ist bei Mappings der Fall, in denen entweder sehr große Input- oder Output-Dateien verarbeitet werden oder in denen wiederholt durch große Datensammlungen iteriert wird.

Um das Debuggen zu beschleunigen und weniger speicherintensiv zu gestalten, wird empfohlen, vor dem Debuggen die folgenden Schritte zu setzen:

- Entfernen Sie bei komplexen Mappings die Teile des Mappings, die nicht debuggt werden müssen bzw. entfernen Sie die entsprechenden Mapping-Verbindungen.
- Wenn im Mapping große Input-Dateien verwendet werden, ersetzen Sie diese durch Dateien kleinerer Größe.
- Stellen Sie sicher, dass die Option **Vollständigen Verlauf der Ablaufverfolgung behalten** deaktiviert ist (siehe Debugger-Einstellungen 1541)

Um sicherzustellen, dass Sie die richtige Ausgabe debuggen, überprüfen Sie die folgenden Dinge, falls zutreffend:

- Wählen Sie durch Klick auf die Schaltfläche **Vorschau** () die zu debuggende Zielkomponente aus, wenn das Mapping mehrere Zielkomponenten hat.
- Deaktivieren Sie die Schaltfläche **Weiterleitung** () der Zwischenkomponente, wenn es sich um ein verkettetes Mapping (siehe Verkettete Mappings 1299) handelt. Das Debuggen von Weiterleitungskomponenten wird derzeit nicht unterstützt.

Wenn der Debugger an wichtigen Konnektoren, deren Wert Sie analysieren möchten, anhalten soll, so fügen Sie optional Breakpoints zu diesen Konnektoren hinzu (siehe Hinzufügen und Entfernen von Breakpoints 1418).

9.2 Informationen zum Debug-Modus

Wenn Sie den Debugger starten (durch Drücken von **F5**, **F11** oder **Strg + F11**), führt MapForce das Mapping im Debug-Modus aus.

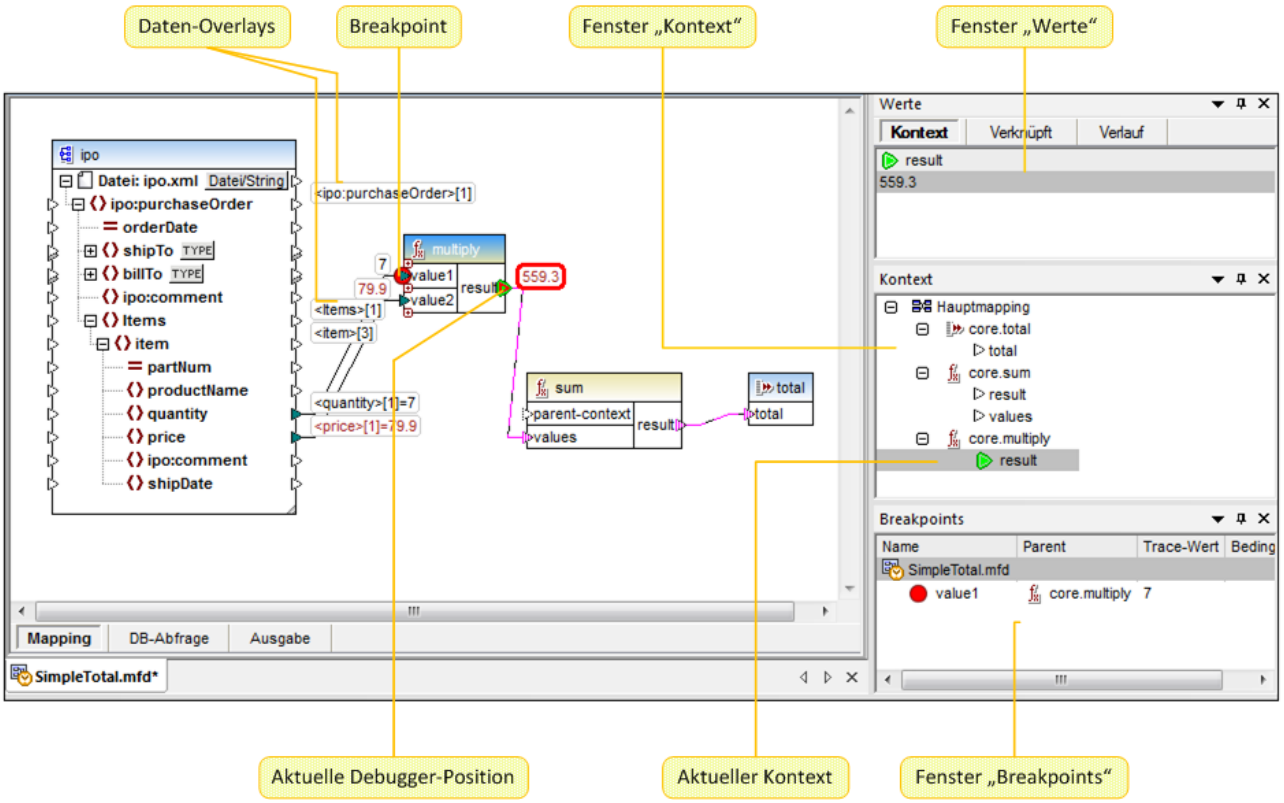
> Im Debug-Modus ist ein Mapping schreibgeschützt. Sie können Komponenten im Mapping-Bereich zwar verschieben, doch stehen die meisten Befehle nicht zur Verfügung. Dazu gehören Befehle wie die Mapping-Validierung, die Bereitstellung von Mappings, Codegenerierung, die Dokumentation von Mappings, das Hinzufügen neuer Komponenten zum Mapping-Bereich oder das Neuladen bestehender Mappings sowie andere Befehle.

Im Debug-Modus können Sie den Kontext, der für die Erzeugung eines bestimmten Werts verantwortlich ist, analysieren. Diese Informationen stehen direkt im Mapping sowie in den Fenstern "Werte", "Kontext" und "Breakpoints" zur Verfügung. Diese Fenster werden standardmäßig angezeigt, wenn Sie den Debugger starten, und ausgeblendet, wenn Sie den Debugger anhalten.

MapForce befindet sich solange im Debug-Modus (und das Mapping ist so lange schreibgeschützt), bis Sie das Debuggen durch Drücken von **Umschalt + F5** (oder durch Klicken auf die Symbolleisten-Schaltfläche **Debugger anhalten**) beenden.

Im folgenden Bild sehen Sie ein Beispielmapping (**SimpleTotal.mfd** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**), das schrittweise (mit **F11**) debuggt wird.

© 2016-2022 Altova GmbH



Die MapForce Entwicklungsumgebung im Debug-Modus

Im Folgenden sind die optischen Hinweise und andere Informationen beschrieben, die MapForce im Debug-Modus zur Verfügung stellt.

Der Mapping-Bereich

Im Debug-Modus werden im Mapping-Bereich die folgenden zusätzlichen Informationen angezeigt:

- Daten-Overlays (siehe unten) zeigen den aktuellen Wert sowie damit in Zusammenhang stehende Werte in der Nähe der jeweiligen Konnektoren an.
- Der aktuelle Kontext (der im Fenster "Kontext" als Struktur angezeigt wird), wird folgendermaßen gekennzeichnet:
  - Konnektoren im Kontext werden magentafarben gestreift angezeigt ( ).
  - Konnektoren in nicht eindeutigem Kontext werden magentafarben gepunktet angezeigt ( ).
  - Konnektoren im Kontext werden magentafarben gestreift angezeigt.
  - Konnektoren in nicht eindeutigem Kontext werden magentafarben gestreift, jedoch heller angezeigt.
- Die Stelle, die gerade ausgeführt wird, wird mit einem grünen Konnektorsymbol ( ) markiert.

Daten-Overlays

Die von den einzelnen Konnektoren verarbeiteten Werte werden in der Nähe des entsprechenden Konnektors als Daten-Overlays (kleine Rechtecke) angezeigt. Das gerade ausgewählte Daten-Overlay wird durch eine dicke rote Umrandung markiert. Werte, die sich seit dem letzten Schritt geändert haben, werden dunkelrot angezeigt. Bei Nodes mit einfachem Inhalt werden im Daten-Overlay zwei Werte kombiniert, nämlich Node-Name und Wert. Wenn vor der aktuellen Ausführungsposition mehrmals über den Node-Namen iteriert wurde, wird der Index der aktuellen Iteration durch die Zahl in eckigen Klammern angegeben.

Daten-Overlays weisen das folgende Verhalten auf:

- Wenn Sie den Mauszeiger über ein Daten-Overlay positionieren, so wird dieses Overlay vorübergehend im Vordergrund angezeigt, wenn Sie darauf klicken, so wird es permanent im Vordergrund angezeigt. Durch Anklicken wird auch der entsprechende Konnektor ausgewählt.
- Daten-Overlays können durch Ziehen mit der Maus verschoben werden.
- Daten-Overlays werden verschoben, wenn eine Komponente verschoben wird. Wenn die Daten-Overlays einander daher überlappen, weil sich die Komponenten zu nahe nebeneinander befinden, ziehen Sie die Komponenten an eine andere Stelle im Mapping-Bereich, um mehr Platz zu haben. Die Daten-Overlays werden zusammen mit der Komponente verschoben.
- Wenn Sie auf ein Daten-Overlay klicken, so wird sein Wert im Fenster "Werte" angezeigt.
- Wenn Sie auf einen Konnektor klicken, wird auch sein Daten-Overlay ausgewählt.

Breakpoints

Breakpoints sind gesetzte Meilensteine, an denen das Mapping während der Ausführung im Debug-Modus unterbrochen werden soll. Breakpoints sind bereits aus anderen integrierten Entwicklungsumgebungen bekannt. Im Gegensatz zu anderen Entwicklungsumgebungen, in denen Sie Breakpoints zu einer Codezeile hinzufügen, kann ein Breakpoint in MapForce zu einem Input- oder Output-Konnektor (kleines Dreieck links oder rechts von der Verbindung) hinzugefügt werden. Im Mapping-Bereich werden Breakpoints als rote Kreise angezeigt. Alle definierten Breakpoints werden außerdem im Fenster "Breakpoints" angezeigt. Siehe auch Hinzufügen und Entfernen von Breakpoints 1418.

Aktuelle Debugger-Position

Die Position des Debuggers wird durch ein grünes Dreieck ( ) gekennzeichnet. Dabei handelt es sich entweder um einen Input- oder einen Output-Konnektor einer Komponente.

Außerdem wird der gerade verarbeitete Wert im Fenster "Werte" auf dem Register "**Kontext**" angezeigt.

Der aktuellen Mapping-Kontext wird durch magentafarben gestreifte Verbindungen und/oder Konnektoren gekennzeichnet. Dieselben Informationen werden außerdem im Fenster "Kontext" in Form einer hierarchischen Struktur angezeigt (siehe Verwendung des Fensters "Kontext" 1423).

Wenn Sie den Kontext eines Werts manuell setzen, so befindet sich die aktuelle Debugger-Position auf einer Position, die sie in Bezug auf die aktuelle Ausführungsposition in der Vergangenheit befindet. Um die aktuelle Ausführungsposition von der in der Vergangenheit liegenden zu unterscheiden, kann der Konnektor der "aktuellen Position" im Debugger in den folgenden Farben angezeigt werden.

	Grün ist die "Gegenwart". Damit wird die aktuelle Ausführungsposition markiert (siehe Anzeigen des aktuellen Werts eines Konnektors 1428).

© 2016-2022 Altova GmbH

	Gelb ist die "Vergangenheit". Diese Farbe zeigt an, dass Sie sich einen Konnektor, der sich in Bezug auf die aktuelle Ausführungsposition in der Vergangenheit befindet, ansehen. Dies kann der Fall sein, wenn Sie einen Kontext manuell definiert haben (siehe Definieren des Kontexts für einen Wert[1431]).

Fenster "Werte"

Im Fenster "Werte" finden Sie Informationen über die von einem Mapping verarbeiteten Werte. In diesem Fenster sehen Sie, was an der aktuellen Ausführungsposition bzw. in einem von Ihnen selbst definierten bestimmten Kontext vom Mapping verarbeitet wird. Siehe auch Verwendung des Fensters "Werte"[1421].

Fenster "Kontext"

Im Fenster "Kontext" sehen Sie eine hierarchische Ansicht der Nodes und Funktionen, die für die aktuelle Debugger-Position relevant sind. Siehe auch Verwendung des Fensters "Kontext"[1423].

Fenster "Breakpoints"

Im Fenster "Breakpoints" wird die Liste der seit dem Start von MapForce erstellten Debugging Breakpoints angezeigt. Falls Sie in mehreren Mappings Breakpoints definiert haben, so werden alle davon im Fenster "Breakpoints" angezeigt. Siehe auch Verwendung des Fensters "Breakpoints"[1425].

9.3 Hinzufügen und Entfernen von Breakpoints

Breakpoints sind gesetzte Meilensteine, an denen das Mapping während der Ausführung im Debug-Modus unterbrochen wird. Alle von Ihnen erstellten Breakpoints werden global für alle Mappings gespeichert und im Fenster "Breakpoints" angezeigt. Breakpoints sind gültig, bis Sie diese explizit löschen oder MapForce schließen.

Anmerkung: Zu den folgenden Elementen können keine Breakpoints hinzugefügt werden: Konstanten, die Funktion `core | position`, Nachfahren-Datenelemente von "Alles kopieren"-Verbindungen, Parameter von "inline" gesetzten benutzerdefinierten Funktionen.

Breakpoints können einfache Breakpoints sein oder mit einer Bedingung versehen sein. Bei einfachen Breakpoints wird die Mapping-Ausführung in jedem Fall angehalten. Bei Breakpoints mit einer Bedingung wird die Mapping-Ausführung nur dann angehalten, wenn die Bedingung, die diesen zugewiesen wurde, erfüllt wird. Bedingungen haben die Form vordefinierter Funktionen aus der MapForce-Funktionsbibliothek, für die Sie benutzerdefinierte Werte bereitstellen, d.h. wenn die Bedingung erfüllt wird, also 'true' ist, wird die Mapping-Ausführung an diesem Breakpoint angehalten.

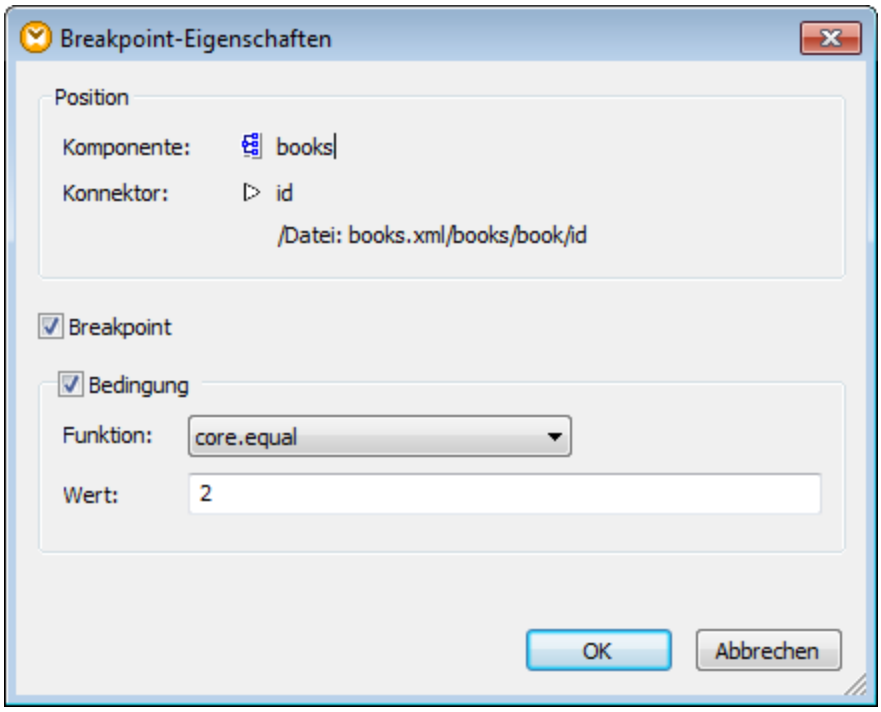
Um einen einfachen Breakpoint zu erstellen, wählen Sie eine der folgenden Methoden:

- Klicken Sie mit der rechten Maustaste auf einen Input- oder Output-Konnektor (die kleinen Dreiecke links oder rechts von einer Komponente) und wählen Sie **Debugger Breakpoint**.
- Klicken Sie auf einen Output-Konnektor und drücken Sie anschließend **F9**.

So erstellen Sie einen Breakpoint mit einer Bedingung:

1. Klicken Sie mit der rechten Maustaste auf einen Konnektor und wählen Sie **Breakpoint-Eigenschaften**.

© 2016-2022 Altova GmbH



2. Aktivieren Sie die beiden Kontrollkästchen **Breakpoint** und **Bedingung**.
3. Wählen Sie die gewünschte Funktion aus der Liste aus und geben Sie den Funktionswert (falls erforderlich) ein. Im Beispiel oben wird die Mapping-Ausführung am Breakpoint angehalten, wenn der Wert, der übergeben wird, größer als 2 ist.

> Wenn der Datentyp des Konnektors, zu dem Sie den Breakpoint mit der Bedingung hinzufügen, nicht mit dem/den von der Funktion erwarteten Typ(en) übereinstimmt, so versucht MapForce, den Datentyp automatisch zu konvertieren. Wenn eine automatische Konvertierung nicht möglich ist, schlägt die Ausführung des Mappings fehl. Wählen Sie daher nur kompatible Datentypen aus, um dies zu vermeiden. So wird z.B. von der Funktion `core.starts-with` ein String-Wert erwartet, daher muss der Konnektor des Breakpoint denselben Typ haben.

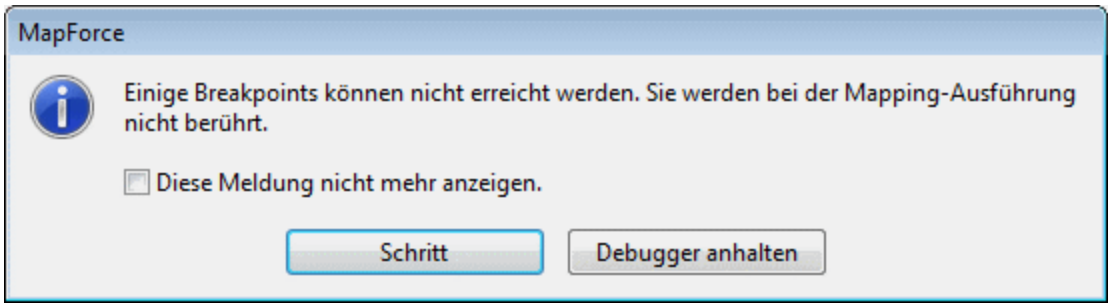
Entfernen von Breakpoints

Um einen Breakpoint zu entfernen, klicken Sie mit der rechten Maustaste auf den Konnektor, an dem der Breakpoint gesetzt wurde und wählen Sie den Befehl **Debugger-Breakpoint**. Klicken Sie alternativ dazu auf den Input- oder Output-Konnektor, an dem der Breakpoint gesetzt wurde, und drücken Sie **F9**.

Sie können Breakpoints aus über das Fenster "Breakpoints" entfernen (siehe Verwendung des Fensters "Breakpoints" 1425).

Nicht erreichbare Breakpoints

In manchen Fällen wird von MapForce eine Meldung angezeigt, dass Breakpoints nicht erreicht werden konnten:



Dies weist darauf hin, dass Breakpoints aus einem der folgenden Gründe vom Debugger nicht erreicht werden konnten:

- Es wurde ein Breakpoint für einen Konnektor definiert, der im Mapping nicht verwendet wird.
- Der Breakpoint konnte aufgrund interner Optimierungen von MapForce nicht erreicht werden (siehe Einschränkungen[1411]).

Klicken Sie auf **Fortsetzen,** um zum nächsten definierten Breakpoint oder zum Ende der Debugging-Ausführung weiterzugehen. Klicken Sie auf **Schritt**, um das Debuggen in Schritten zu starten.

Sie können Meldungen über vom Debugger nicht erreichte Breakpoints deaktivieren, indem Sie entweder auf **Diese Meldung nicht mehr anzeigen** klicken, oder folgendermaßen vorgehen:

1. Klicken Sie im Menü **Extras** auf **Optionen**.
2. Klicken Sie auf **Meldungen**.
3. Deaktivieren Sie das Kontrollkästchen **Über nicht erreichbare Breakpoints informieren**.

© 2016-2022 Altova GmbH

9.4 Verwendung des Fensters "Werte"

Im Debug-Modus werden im Fenster "Werte" Informationen über die vom Mapping verarbeiteten Werte angezeigt. Welche Informationen im Fenster "Werte" angezeigt werden, hängt von der aktuellen Debugger-Position und von den Elementen der Benutzeroberfläche, auf die Sie geklickt haben, ab. Das Fenster "Werte" enthält die folgenden Register:

Das Register "Kontext"

Auf dem Register **Kontext** wird der Wert angezeigt, der soeben verarbeitet wird (Dies ist derselbe Wert, dessen Kontext im Fenster **Kontext** angezeigt wird). Dabei handelt es sich entweder um den Wert an der aktuellen Ausführungsposition des Debuggers oder um den Wert eines in der Vergangenheit verarbeiteten Konnektors. Zur Unterscheidung kennzeichnet MapForce den Kontext in unterschiedlichen Farben:

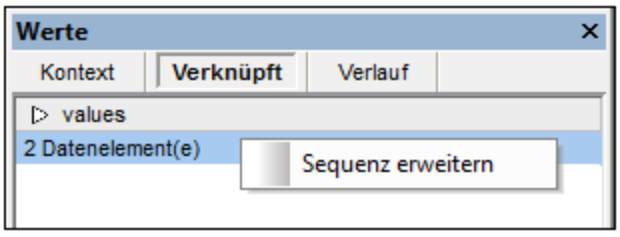
	Grün ist die "Gegenwart". Damit wird die aktuelle Ausführungsposition markiert (siehe Anzeigen des aktuellen Werts eines Konnektors 1428).
	Gelb ist die "Vergangenheit". Diese Farbe zeigt an, dass Sie sich einen Konnektor, der sich in Bezug auf die aktuelle Ausführungsposition in der Vergangenheit befindet, ansehen. Dies kann der Fall sein, wenn Sie einen Kontext manuell definiert haben (siehe Definieren des Kontexts für einen Wert 1431).

Das Register "Verknüpft"

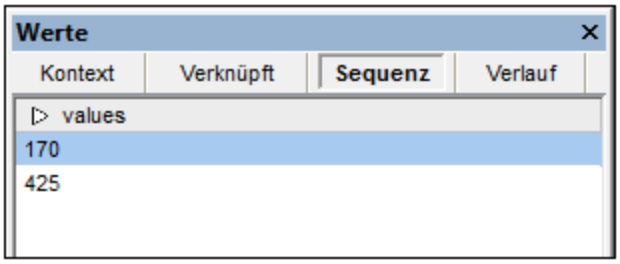
Auf dem Register **Verknüpft** werden Werte angezeigt, die mit dem aktuell verarbeiteten Wert in Zusammenhang stehen (oder dessen "nahe Vergangenheit" darstellen). Normalerweise müssen Sie explizit auf dieses Register klicken. Wenn Sie auf das Daten-Overlay eines Konnektors klicken, der mit der aktuellen Ausführungsposition des Debuggers in Zusammenhang steht, wechselt MapForce automatisch zu diesem Register. Siehe Zurücksteigen in die nahe Vergangenheit 1429.

Das Register "Sequenz"

Auf dem Register Sequenz erhalten Sie (wenn dieses angezeigt wird) Zugriff auf die Werte eines Konnektors, der eine Sequenz verarbeitet. Dieses Register ist nur dann sichtbar, wenn ein Konnektor eine Sequenz von Datenelementen verarbeitet hat (z.B. bei einer Aggregatfunktion wie z.B. `sum` oder `count`). Wenn Sie auf das Daten-Overlay eines Konnektors klicken, der eine Sequenz von Datenelementen verarbeitet hat, wird im Fenster "Werte" ein Eintrag im Format "**n Datenelemente**" angezeigt, wobei **n** die Anzahl der vom Konnektor verarbeiteten Datenelemente anzeigt. Um die einzelnen Werte aufzurufen, doppelklicken Sie auf diesen Eintrag (oder klicken Sie mit der rechten Maustaste darauf und wählen Sie im Kontextmenü den Befehl **Sequenz erweitern**).



Die Werte werden anschließend auf dem Register **Sequenz** angezeigt.



Das Register "Verlauf"

Auf dem Register **Verlauf** werden Werte angezeigt, die vom Beginn des Debuggens bis zur aktuellen Ausführungsposition von einem bestimmten Node verarbeitet wurden. Siehe Anzeigen des Verlaufs der von einem Konnektor verarbeiteten Werte 1430.

 © 2016-2022 Altova GmbH

9.5 Verwendung des Fensters "Kontext"

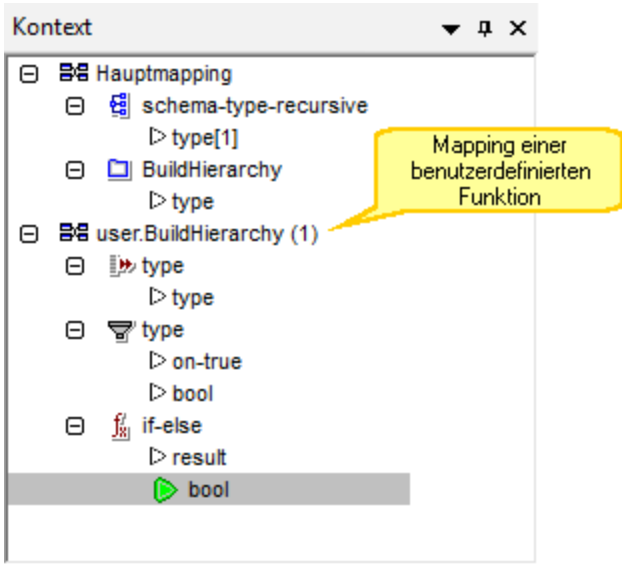
Während MapForce sich im Debug-Modus befindet, wird im Fenster "Kontext" eine Struktur von Konnektoren, die für die aktuelle Debugger-Position relevant sind, angezeigt, d.h. es wird der Mapping-Kontext angezeigt, in dem der aktuelle Mapping-Wert erzeugt wurde.

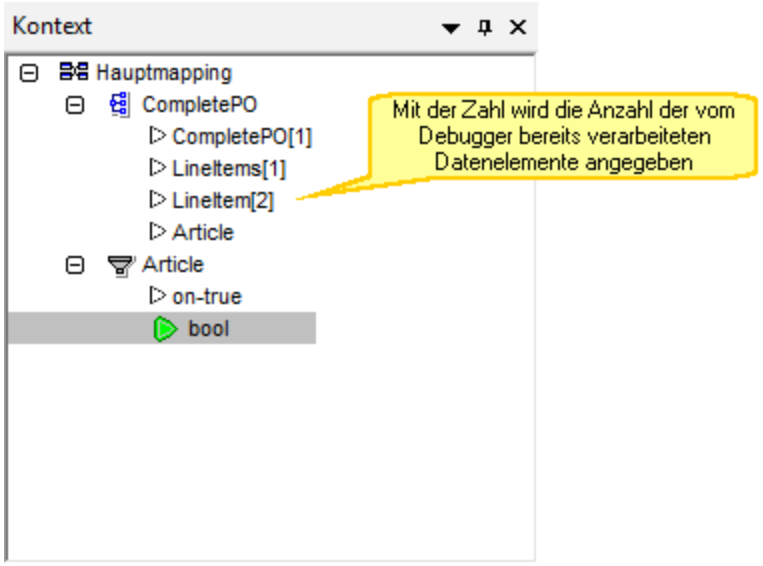
MapForce erstellen den aktuellen Kontext folgendermaßen:

1. Es startet mit dem Root-Node der Zielstruktur.
2. Es geht hinunter bis zu aktuellen Ziel-Node.
3. Es geht vom aktuellen Node aus im Mapping nach links durch alle Komponenten, die zur aktuellen Position führen. Dabei kann es sich um Filter- oder Sortierkomponenten, um vordefinierte oder benutzerdefinierte Funktionen, Variablen usw. handeln.

Das Fenster "Kontext" dient sowohl zu Informations- als auch zu Navigationszwecken. Um einen bestimmten Node im Mapping direkt vom aktuellen Kontext aus auszuwählen, klicken Sie im Fenster "Kontext" mit der rechten Maustaste auf den Node und wählen Sie den Befehl **Im Mapping auswählen**. Diese Methode eignet sich vor allem dann, wenn es sich um ein großes Mapping handelt, in dem Sie sonst weit nach unten scrollen müssten.

Im Fenster "Kontext" können die folgenden Sondersymbole und Vermerke angezeigt werden:

Symbol	Beschreibung
	Repräsentiert das Mapping, zu dem der Kontext gehört. Dabei kann es sich entweder um das Hauptmapping oder das Mapping einer benutzerdefinierten Funktion handeln. 
▷	Repräsentiert einen Konnektor. Die Position der bis zu diesem Punkt verarbeiteten Ziel-Nodes wird innerhalb von eckigen Klammern angezeigt.

Symbol	Beschreibung
	
	Repräsentiert den aktuellen Konnektor (die aktuelle Ausführungsposition). Dies ist die Quelle des aktuellen Werts im Fenster "Werte". In einigen seltenen Fällen ist es möglich, dass ein berechneter Wert für mehrere Konnektoren verwendet wird. In diesem Fall werden eventuell mehrere grüne Symbole angezeigt.
	Repräsentiert den aktuellen Konnektro, wenn sich der Debugger an einer Stelle befindet, die in Bezug auf die aktuelle Ausführungsposition in der Vergangenheit liegt, z.B. wenn Sie den Kontext auf einen Wert setzen (siehe Setzen des Kontexts auf einen Wert 1431).

Neben den oben gezeigten Symbolen enthält das Fenster "Kontext" die Standardsymbole aller im Mapping vorhandenen Komponententypen.

Fenster "Kontext" und benutzerdefinierte Funktionen

Falls der aktuelle Kontext benutzerdefinierte Funktionen enthält, werden diese ebenfalls im Fenster "Kontext" angezeigt. Beachten Sie Folgendes: Wenn der aktuelle Kontext zur Berechnung eines Input-Werts einer benutzerdefinierten Funktion verwendet wird, so wird er folgendermaßen ermittelt:

1. Von der Zielkomponente zum Output-Konnektor der benutzerdefinierten Funktion zum Input-Konnektor der benutzerdefinierten Funktion
2. Von dort weiter nach links

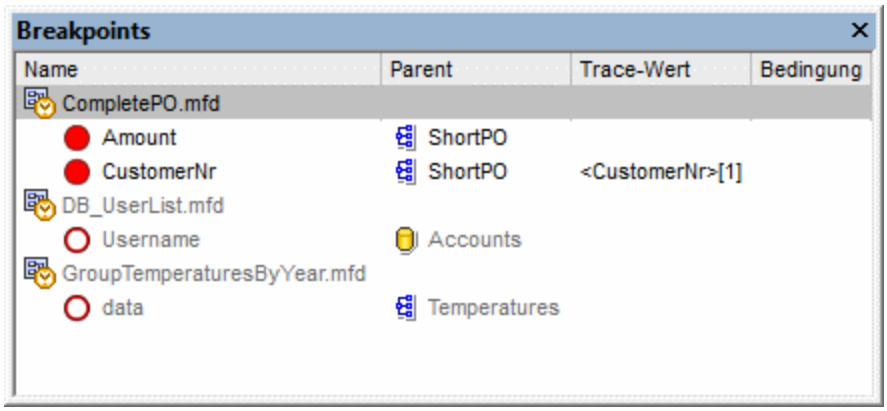
Anmerkung: Eine benutzerdefinierte Funktion kann mehrmals im Kontext vorkommen. Dies passiert entweder, weil mehrere Funktionsaufrufe miteinander verkettet sind oder weil die benutzerdefinierte Funktion als rekursiv definiert ist.

© 2016-2022 Altova GmbH

9.6 Verwendung des Fensters "Breakpoints"

Im Fenster "Breakpoints" können Sie Breakpoints global anzeigen und verwalten. Das Fenster "Breakpoints" wird standardmäßig angezeigt, wenn sich MapForce im Debug-Modus befindet. Um das Fenster "Breakpoints" immer einzublenden, wählen Sie den Menübefehl **Ansicht | Debug-Fenster | Breakpoints**.

Im Fenster "Breakpoints" werden alle seit dem Start von MapForce erstellten Breakpoints, gruppiert nach der Mapping-Datei, zu der sie gehören, angezeigt. Während MapForce geöffnet ist, notiert MapForce alle mit irgendeinem Mapping verknüpften Breakpoints und zeigt diese im Fenster "Breakpoints" an, selbst, wenn Sie die Mapping-Datei in der Zwischenzeit geschlossen haben. Während ein Mapping debuggt wird, wird es im Fenster "Breakpoints" in der Standardtextfarbe angezeigt, während andere Mappings (geschlossene oder nicht aktive Mappings) ausgegraut sind.



Durch Doppelklick auf ein Mapping (oder eines seiner Breakpoints) im Fenster "Breakpoints" können Sie ein Mapping schnell öffnen.

Anmerkung: Sobald Sie MapForce schließen oder neu starten, werden alle Breakpoints entfernt.

Informationen zu Breakpoints werden in Form einer Tabelle mit den folgenden Spalten angezeigt:

Spalte	Beschreibung
Name	Der Name des Node, zu dem der Breakpoint gehört.
Parent	Der Name der Mapping-Komponente, zu der der Breakpoint gehört.
Trace-Wert	Der Wert, der über den Konnektor, auf dem der Breakpoint gesetzt ist, übergeben wird. Der Trace-Wert wird während der Ausführung des Debuggers angezeigt.
Bedingung	Wenn es zum Breakpoint eine Bedingung gibt, so wird in dieser Spalte die Bedingung des Breakpoint angezeigt.

Breakpoints müssen mit einem der folgenden Symbole verknüpft sein.

Symbol	Beschreibung
	Aktiver Breakpoint. Kennzeichnet einen Breakpoint aus dem Mapping, das gerade debuggt wird.
	Nicht aktiver Breakpoint. Kennzeichnet einen Breakpoint aus einem Mapping, das offen ist, aber gerade nicht debuggt wird.
	Nicht erreichbarer Breakpoint. Kennzeichnet einen Breakpoint, der vom Debugger nicht erreicht werden kann.
	Breakpoint mit einer Bedingung. Kennzeichnet einen Breakpoint, der mit einer Bedingung versehen wurde.

So zeigen Sie die Eigenschaften eines Breakpoint an oder ändern diese:

- Klicken Sie mit der rechten Maustaste auf den Breakpoint und wählen Sie im Kontextmenü den Befehl **Breakpoint-Eigenschaften**.

So löschen Sie einen Breakpoint:

- Klicken Sie mit der rechten Maustaste auf den Breakpoint, den Sie löschen möchten und wählen Sie im Kontextmenü den Befehl **Breakpoint löschen**
- Klicken Sie auf einen Breakpoint und drücken Sie **Entf**.

Mit dem Kontextmenübefehl **Alle Breakpoints löschen** entfernen Sie alle im Fenster "Breakpoints" angezeigten Breakpoints, unabhängig davon, zu welchem Mapping diese gehören.

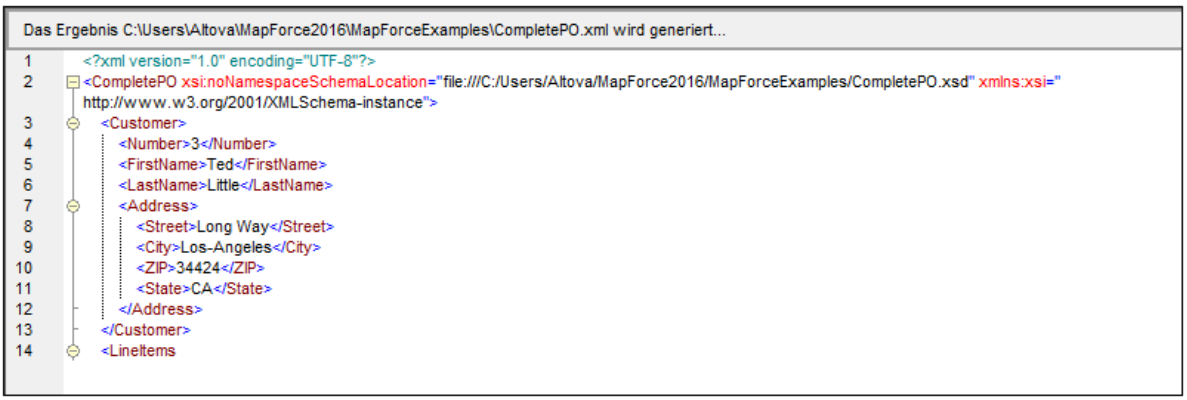
Siehe auch: Hinzufügen und Entfernen von Breakpoints 1418

 © 2016-2022 Altova GmbH

9.7 Vorschau auf teilweise generierte Ausgabe

Wenn Sie in Schritten oder mittels Breakpoints debuggen, können Sie das bis zur aktuellen Debugger-Position generierte Mapping anzeigen. Die Anzeige einer Vorschau auf die teilweise generierte Ausgabe wird von XML-, Flat Text und EDI-Zielkomponenten unterstützt.

Standardmäßig führt MapForce bei Drücken von **F5** (wenn keine Breakpoints definiert sind) das gesamte Mapping im Debug-Modus aus und wechselt anschließend zum Register **Ausgabe**, um das generierte Endresultat anzuzeigen. Wenn Sie allerdings Breakpoints definiert haben oder wenn Sie in Schritten debuggen (**F11** oder **Strg + F11**), so wird die Debugger-Ausführung noch während der Generierung der Mapping-Ausgabe angehalten. Sie können dennoch eine Vorschau auf diese nur zum Teil generierte Ausgabe ansehen, wenn Sie auf das Register **Ausgabe** klicken.



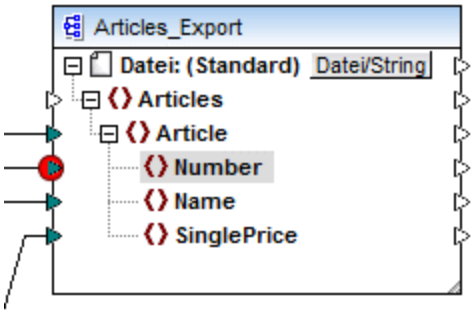
Einschränkungen

- Der aktuell berechnete Ziel-Node wird nicht immer auf dem Register "Ausgabe" angezeigt. So werden XML-Attribute z.B. intern gesammelt und auf einmal geschrieben.
- Wenn in der Ausgabe mehrere Dateien erzeugt werden, kann nur die aktuell geschriebene Datei angezeigt werden; das Wechseln zu einer anderen Ausgabedatei ist deaktiviert.

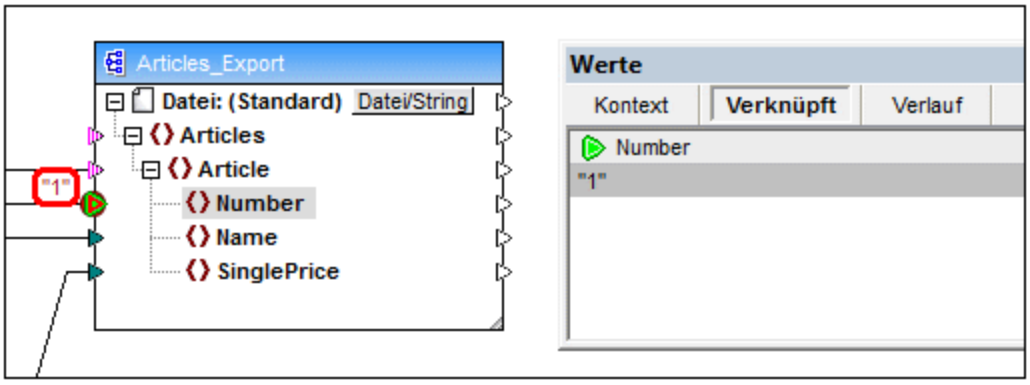
9.8 Anzeigen des aktuellen Werts eines Konnektors

Wenn sich die aktuelle Ausführungsposition des Debuggers ( ) auf einem bestimmten Konnektor befindet (entweder, weil Sie in Schritten debuggen oder weil an diesem Konnektor ein Breakpoint definiert wurde), so wird der vom Konnektor verarbeitete aktuelle Wert im Fenster "Werte" auf dem Register "**Kontext**" angezeigt. Dies ist ein Wert, der gerade in die Ausgabe geschrieben wird, d.h. dieser Wert ist die "Gegenwart". Es ist auch der Wert, dessen Kontext im Fenster "Kontext" angezeigt wird (siehe Verwendung des Fensters "Kontext" 1423).

Öffnen Sie zum besseren Verständnis dieses Falls die Beispieldatei **PreserveFormatting.mfd** aus dem Verzeichnis **<Dokumente>\Altova\MapForce2023\MapForceExamples**. Klicken Sie in der Zielkomponente auf den Input-Konnektor des Node `Number` und drücken Sie **F9,** um einen Breakpoint zum Node hinzuzufügen.



Drücken Sie anschließend **F5,** um das Debugging zu starten und die Ergebnisse zu beobachten.



Wie im Bild gezeigt, befinden sich der Debugger (und der Breakpoint) gerade auf dem Node `Number` der Zielkomponente. Im Fenster "Werte" wird angezeigt, dass in diesem Node der Wert "1" verarbeitet wird (dieser Wert wird im Mapping durch eine dicke rote Umrandung markiert).

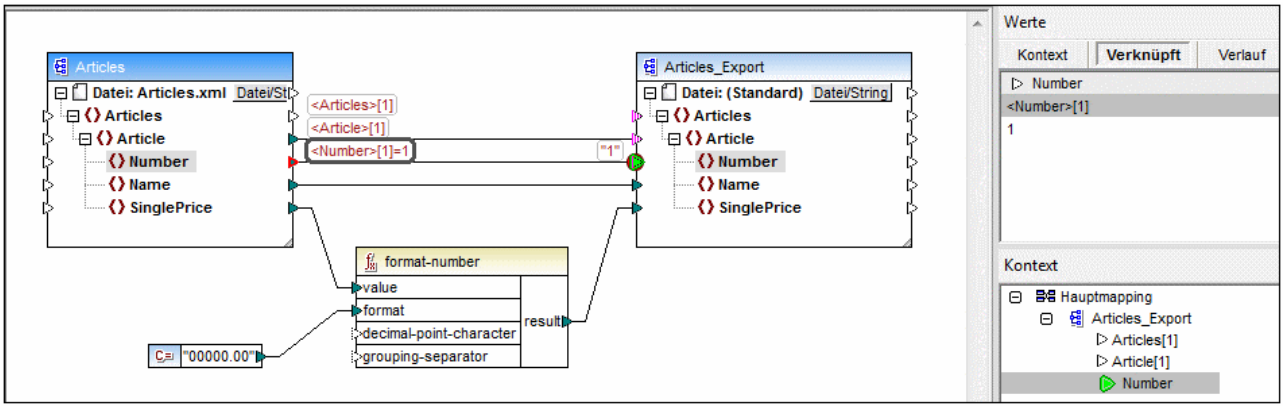
© 2016-2022 Altova GmbH

9.9 Zurücksteigen in die nahe Vergangenheit

Wenn Sie auf ein Daten-Overlay (ein kleines rechteckiges Feld) neben einem Mapping-Konnektor klicken, werden im Fenster **Werte** der Name und optional der mit dem ausgewählten Konnektor verknüpfte Wert angezeigt. Der Fokus ist nun nicht mehr auf der aktuellen Debugger-Position, sondern auf dem ausgewählten Daten-Overlay. Auf diese Art können Sie einen Blick in die unmittelbare Vergangenheit im Debugger-Verlauf werfen. Dies ist die "nahe" Vergangenheit, da nur Daten-Overlays für die letzten Konnektoren, die im Zusammenhang mit der aktuellen Debugger-Position standen, angezeigt werden. Wenn Sie daher auf ein solches "verknüpftes" Daten-Overlay klicken, wird im Fenster "Werte" automatisch das Register **Verknüpft** angezeigt.

Öffnen Sie zum besseren Verständnis das Mapping **PreserveFormatting.mfd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples**.

Nachdem Sie das Mapping geöffnet haben, klicken Sie auf den Konnektor neben dem Node `Number` der Zielkomponente und drücken Sie **F9,** um einen Breakpoint hinzuzufügen. Drücken Sie **F5,** um das Debuggen zu starten und klicken Sie anschließend auf das Daten-Overlay (das kleine rechteckige Feld) neben dem Node `Number` der Quellkomponente.



Da während eines Mappings normalerweise mehrmals über einen Konnektor iteriert wird, wird der aktuelle Index der Iteration innerhalb von eckigen Klammern angezeigt: **<Number>[1]**. Da der Konnektor außerdem einen Wert trägt, wird auch der Wert nach dem Ist gleich-Zeichen angezeigt: **<Number>[1]=1**. Derselbe Wert wird in einer neuen Zeile im Fenster "Werte" angezeigt (siehe Abbildung unten).

Wenn Sie zusätzliche Informationen zu einem bestimmten Wert benötigen, können Sie auch den Kontext, in dem dieser erzeugt wurde, wieder anzeigen (siehe Setzen des Kontexts auf einen Wert 1431).

9.10 Anzeigen des Verlaufs der von einem Konnektor verarbeiteten Werte

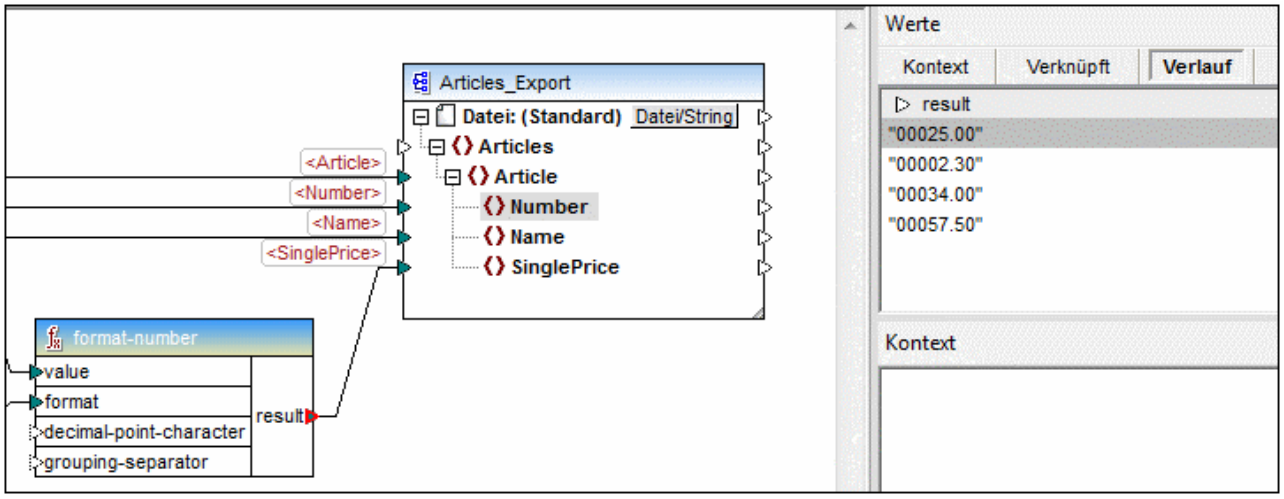
Wenn die Option **Vollständigen Verlauf der Ablaufverfolgung behalten** aktiviert ist (siehe Debugger-Einstellungen[1541]), können Sie den Verlauf aller von diesem Konnektor (bis zur aktuellen Ausführungsposition) verarbeiteten Werte anzeigen.

Der Verlauf wird angezeigt, wenn Sie auf einen Konnektor klicken und anschließend im Fenster "Werte" auf das Register **Verlauf** klicken. Beachten Sie, dass diese Operation nur bei Konnektoren sinnvoll ist, die seit Beginn der Mapping-Ausführung bis zur aktuellen Debugger-Position Werte verarbeitet haben.

Wir wollen nun zur Veranschaulichung dieses Falls ein Mapping von Anfang bis Ende ohne Verwendung von Breakpoints debuggen und anschließend den Verlauf aller von einem bestimmten Konnektor verarbeiteten Werte beobachten. Öffnen Sie zuerst das Mapping **PreserveFormatting.mfd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples\.** Wenn es bereits geöffnet ist, führen Sie zuerst die folgenden Schritte durch:

- Löschen Sie alle Breakpoints, falls vorhanden (siehe Hinzufügen und Entfernen von Breakpoints[1418])
- Halten Sie den Debugger durch Drücken von **Umschalt + F5** an, falls er gerade ausgeführt wird.

Wenn Sie bereit sind, drücken Sie **F5,** um einen neuen Debugging-Vorgang zu starten. Wenn Sie **F5** drücken, führt MapForce das Mapping im Debug-Modus aus und wechselt zum Register **Ausgabe**. Klicken Sie auf das Register **Mapping,** um wieder zurück ins Mapping-Hauptfenster zu wechseln und klicken Sie anschließend auf den Node `result` der Funktion `format-number` (im Bild unten rot markiert). Klicken Sie zum Schluss im Fenster "Werte" auf das Register **Verlauf** und beachten Sie die angezeigten Werte.

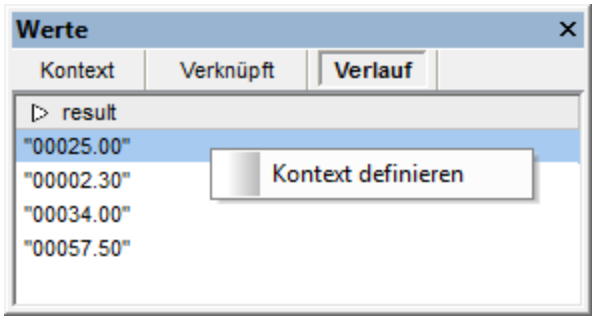


Wie in der Abbildung oben gezeigt, wurden von diesem bestimmten Node (`result`) insgesamt vier Werte verarbeitet. Wenn Sie zusätzliche Informationen zu einem bestimmten Wert benötigen, können Sie den Kontext, der für deren Erzeugung verantwortlich war, auch wiederherstellen (siehe Setzen des Kontexts auf einen Wert[1431]).

© 2016-2022 Altova GmbH

9.11 Setzen des Kontexts auf einen Wert

Wenn Sie den Kontext auf einen Wert setzen, so ist das, als würden Sie in die Vergangenheit zurückgehen, um nähere Einzelheiten zu dem Mapping-Kontext, von dem der Wert erzeugt wurde, anzuzeigen. Sie können als Kontext jeden im Fenster "Werte" (auf den Registern **Verknüpft, Sequenz** oder **Verlauf**) angezeigten Wert definieren. Wenn Sie die Option **Vollständigen Verlauf der Ablaufverfolgung behalten** aktiviert haben (siehe Debugger-Einstellungen[1541]), werden auf dem Register **Verlauf** alle vom aktuell ausgewählten Konnektor verarbeiteten Werte angezeigt, daher können Sie in diesem Fall den Kontext zusätzlich auf jeden Wert setzen, den dieser Konnektor in der Vergangenheit verarbeitet hat.

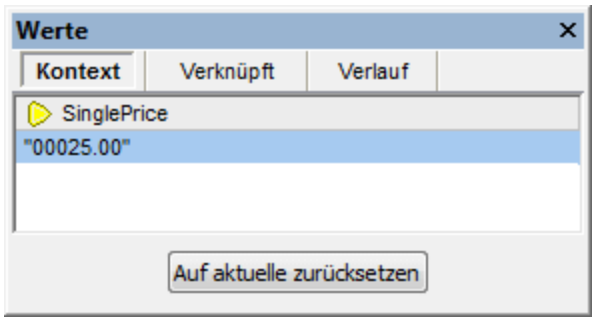


Um für den Kontext einen Wert zu definieren, wählen Sie eine der folgenden Methoden:

- Klicken Sie mit der rechten Maustaste auf den Wert und wählen Sie im Kontextmenü **Kontext definieren**.
- Doppelklicken Sie auf den Wert.

Wenn Sie den Kontext auf einen Wert setzen, markiert MapForce den Mapping-Bereich, um die Situation, in der der Wert erzeugt wurde, wiederherzustellen und befüllt die Fenster **Werte** und **Kontext** gemäß dem ausgewählten Kontext. Eine Legende zu den im Mapping-Bereich in einem Kontext angezeigten Symbolen finden Sie unter Informationen zum Debug-Modus[1414]. Informationen zum Kontext selbst finden Sie unter Verwendung des Fensters "Kontext"[1423].

Der Konnektor eines manuell definierten Kontexts ist gelb ( ). Damit wird angezeigt, dass Sie sich nicht mehr an der aktuellsten Ausführungsposition befinden. Um wieder zurück zur aktuellsten Ausführungsposition zu wechseln, klicken Sie im Fenster "Werte" auf dem Register **Kontext** auf **Auf aktuelle zurücksetzen**.



10 Automatisieren mit Altova-Produkten

Mit MapForce erstellte Mappings können von den folgenden (separate lizenzierten) Altova-Transformationsprozessoren in einer Server-Umgebung (u.a. auch mit Linux und macOS-Servern) und mit der Leistung eines Servers ausgeführt werden:

- *RaptorXML Server*. Dieser Prozessor eignet sich für die Ausführung des Mappings, wenn die Transformationssprache des Mappings XSLT 1.0, XSLT 2.0, XSLT 3.0 oder XQuery ist. Siehe Automatisierung mit RaptorXML Server[1433].
- *MapForce Server (oder MapForce Server Advanced Edition).* Dieser Prozessor eignet sich für Mappings, deren Transformationssprache BUILT-IN* ist. Die Sprache BUILT-IN unterstützt die meisten Mapping-Funktionen in MapForce, während MapForce Server (und v.a. die MapForce Server Advanced Edition) die höchste Leistung für die Verarbeitung eines Mappings bietet. Siehe Automatisierung mit MapForce Server[1434].

** Für die Transformationssprache BUILT-IN wird MapForce Professional oder Enterprise Edition benötigt.*

Zusätzlich dazu bietet MapForce die Möglichkeit, die Generierung von XSLT-, XQuery-, C#-, C++- und Java-Code über die Befehlszeile zu automatisieren. Dabei können auch Server-Ausführungsdateien (.mfx) für die MapForce Server-Ausführung kompiliert werden. Nähere Informationen dazu finden Sie unter MapForce-Befehlszeilenschnittstelle[1456].

© 2016-2022 Altova GmbH

10.1 Automatisierung mit RaptorXML Server

Altova RaptorXML Server (in der Folge als RaptorXML bezeichnet) ist Altovas ultraschneller XML- und XBRL-Prozessor der dritten Generation, der für die neuesten Standards und parallele Rechnerumgebungen optimiert wurde. RaptorXML lässt sich plattformübergreifend einsetzen und ermöglicht dank der Nutzung moderner Multi-Core Computer die ultraschnelle Verarbeitung von XML- und XBRL-Daten.

RaptorXML steht in mehreren Editionen zur Verfügung, die von der Altova Download-Seite heruntergeladen und anschließend installiert werden können (https://www.altova.com/de/download-trial-server.html):

- RaptorXML Server ist ein sehr schneller Prozessor zur Verarbeitung von XML, der XML, XML-Schema, XSLT, XPath, XQuery und mehr.
- RaptorXML+XBRL Server unterstützt alle Funktionen von RaptorXML Server und kann zusätzlich XBRL-Daten verarbeiten und validieren.

Beim Generieren von Code in XSLT oder XQuery generiert MapForce eine Batch-Datei namens **DoTransform.bat**, die in dem bei der Generierung gewählten Ausgabeordner gespeichert wird. Die Batch-Datei ruft RaptorXML Server auf und führt die XSLT/XQuery-Transformation auf dem Server aus.

Nähere Informationen zum Ausführen oder Automatisieren von MapForce Mappings auf einem Server für andere Ausgaben finden Sie unter Automatisierung mit MapForce Server 1434.

Anmerkung: Sie können mit dem Built-in-Ausführungsprozessor eine Vorschau des XSLT- 109 und XQuery -Codes anzeigen.

10.2 Automatisierung mit MapForce Server

MapForce Server ist ein Server-Produkt für Unternehmen für Windows-, Linux- und macOS-Betriebssysteme. MapForce Server dient dazu, Mappings in einer Server-Umgebung (auch auf Nicht-Windows-Plattformen) und mit der Leistung eines Servers auszuführen. Für die Server-Ausführung kann jedes MapForce-Mapping, dessen Zielsprache BUILT-IN ist, verwendet werden (siehe auch Auswählen einer Transformationssprache 28). MapForce Server kann als Standalone-Produkt (Aufruf über die Befehlszeile oder API) oder unter Verwaltung von FlowForce Server ausgeführt werden.

Wenn MapForce Server als eigenständiges Produkt verwendet wird, muss das MapForce Mapping kompiliert und auf den Rechner, auf dem MapForce Server ausgeführt wird, kopiert werden. Anschließend wird es über den MapForce Server-Befehlszeilenbefehl `run` ausgeführt. Sie können das Mapping auch durch Aufrufen der `run`-Methode der MapForce Server API ausführen. Nähere Informationen dazu finden Sie unter Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442.

Wenn MapForce Server unter Verwaltung von FlowForce Server läuft, kann das Mapping über eine HTTP- (oder SSL/HTTPS)-Verbindung direkt von MapForce aus für einen Zielrechner bereitgestellt werden. Am Server kann das Mapping anschließend in Form eines Auftrags über einen Trigger, als geplanter Auftrag oder über einen über die FlowForce Server-Verwaltungsschnittstelle definierten Webservice-Aufruf ausgeführt werden. Nähere Informationen dazu finden Sie unter Bereitstellen von Mappings auf FlowForce Server 1445.

Es gibt zwei Editionen von MapForce Server:

- MapForce Server
- MapForce Server Advanced Edition

Die MapForce Server Advanced Edition enthält dieselben Funktionalitäten wie MapForce Server sowie zusätzlich dazu Optimierungsfunktionalitäten für Mappings, die sich optimieren lassen. Dies ist bei Mappings der Fall, in denen große Datenmengen mittels Join verknüpft oder gefiltert werden und bei denen die Ausführungsgeschwindigkeit durch eine Join-Optimierung verbessert werden kann. Im Gegensatz zu MapForce Server, kann die MapForce Server Advanced Edition Mappings, in denen Node-Funktionen vorhanden sind, ausführen, siehe Standardwerte und Node-Funktionen 951.

Einschränkungen:

- Digitale XML-Signaturen werden nicht unterstützt
- ADO-, ADO.NET- und ODBC-Datenbankverbindungen werden nur von Windows unterstützt (Informationen zur Unterstützung auf anderen Betriebssystemen finden Sie unter Vorbereiten von Mappings für die Server-Ausführung 1435).

Nähere Informationen zu MapForce Server finden Sie in der Dokumentation zu MapForce Server (https://www.altova.com/de/documentation).

© 2016-2022 Altova GmbH

10.3 Vorbereiten von Mappings für die Server-Ausführung

Ein Mit MapForce erstelltes Mapping, das auch in der MapForce-Vorschau angezeigt wird, kann Ressourcen (z.B. Datenbanken) referenzieren, die sich nicht auf dem aktuellen Rechner und Betriebssystem befinden. Außerdem entsprechen in MapForce alle Mapping-Pfade standardmäßig den Windows-Konventionen. Zudem unterstützt der Rechner, auf dem MapForce Server läuft, nicht notwendigerweise dieselben Datenbankverbindungen wie der Rechner, auf dem das Mapping erstellt wurde. Aus diesem Grund, müssen Sie ein Mapping entsprechend vorbereiten, bevor Sie es in einer Server-Umgebung ausführen, vor allem dann, wenn der Zielrechner nicht derselbe wie der Quellrechner ist.

Anmerkung: Mit "Quellrechner" wird der Computer bezeichnet, auf dem MapForce installiert ist und mit "Zielrechner" wird der Computer bezeichnet, auf dem MapForce Server oder FlowForce Server installiert ist. Im einfachsten Szenario handelt es sich hierbei um denselben Computer. In einem komplexeren Szenario läuft MapForce auf einem Windows-Rechner, während MapForce Server auf einem Linux- oder macOS-Rechner ausgeführt wird.

Stellen Sie am besten immer sicher, dass sich das Mapping in MapForce erfolgreich validieren lässt, bevor Sie es auf FlowForce Server bereitstellen oder zu einer MapForce Server-Ausführungsdatei kompilieren (siehe Validieren von Mappings 107).

Wenn MapForce Server alleine (ohne FlowForce Server) ausgeführt wird, werden die folgenden Lizenzen benötigt:

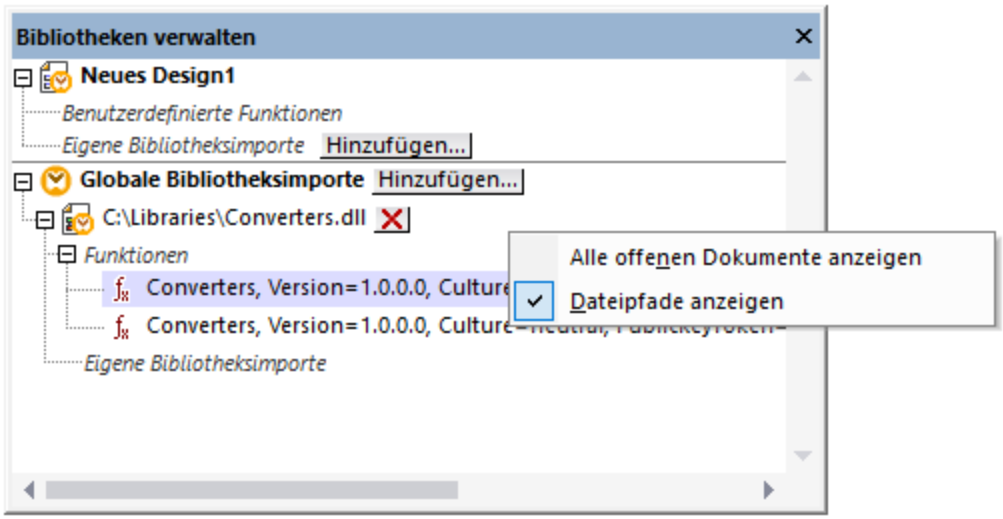
- Auf dem Quellrechner benötigen Sie zur Erstellung des Mappings und zum Kompilieren zu einer Server-Ausführungsdatei (.mfx) die MapForce Enterprise oder Professional Edition, siehe Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442.
- Auf dem Zielrechner benötigen Sie zur Ausführung des Mappings MapForce Server oder MapForce Server Advanced Edition.

Wenn MapForce Server unter Verwaltung von FlowForce Server ausgeführt wird, gelten die folgenden Voraussetzungen:

- Auf dem Quellrechner benötigen Sie zur Erstellung des Mappings und für die Bereitstellung des Mappings auf einem Zielrechner die MapForce Enterprise oder Professional Edition, siehe Bereitstellen von Mappings auf FlowForce Server 1481.
- Sowohl MapForce Server als auch FlowForce Server müssen auf dem Zielrechner lizenziert sein. Die Aufgabe von MapForce Server ist die Ausführung des Mappings; die Aufgabe von FlowForce ist es, das Mapping in Form eines Auftrags bereitzustellen, wodurch Funktionen wie Ausführung nach einem Zeitplan oder auf Wunsch, die Ausführung als Webservice, Fehlerbehandlung, Ausführung auf Basis von Bedingungen, E-Mail-Benachrichtigungen usw. zur Verfügung stehen.
- FlowForce Server muss an der konfigurierten Netzwerkadresse und am konfigurierten Port gestartet sein. Dabei muss der "FlowForce Web Server"-Dienst gestartet sein und so konfiguriert sein, dass er Verbindungen von HTTP-Clients (oder bei Bedarf von HTTPS-Clients) zulässt und er darf nicht durch die Firewall blockiert werden. Auch der "FlowForce Server"-Dienst muss unter der vorgesehenen Adresse und am angegebenen Port gestartet und verfügbar sein.
- Sie haben ein FlowForce Server-Benutzerkonto und Zugriff auf einen der Container (standardmäßig steht der Container **/public** jedem authentifizierten Benutzer zur Verfügung).

Allgemeine Überlegungen

- Wenn Sie beabsichtigen, das Mapping mit MapForce Server alleine auf einem Zielrechner auszuführen, müssen alle vom Mapping referenzierten Input-Dateien ebenfalls auf den Zielrechner kopiert werden. Wenn MapForce Server unter Verwaltung von FlowForce Server ausgeführt wird, müssen die Dateien nicht manuell kopiert werden. In diesem Fall sind die Instanz- und Schema-Dateien in dem auf dem Zielrechner bereitgestellten Paket inkludiert, siehe Bereitstellen von Mappings auf FlowForce Server 1481.
- Wenn das Mapping Datenbankkomponenten enthält, für die spezielle Datenbanktreiber benötigt werden, müssen diese Treiber auch auf dem Zielrechner installiert werden. Wenn in Ihrem Mapping z.B. Daten aus einer Microsoft Access-Datenbank ausgelesen werden, so muss Microsoft Access oder Microsoft Access Runtime (https://www.microsoft.com/en-us/download/details.aspx?id=50040) auch auf dem Zielrechner installiert sein.
- Wenn Sie ein Mapping auf Nicht-Windows-Plattformen bereitstellen, werden ADO-, ADO.NET- und ODBC-Datenbankverbindungen automatisch in JDBC konvertiert. Native SQLite- und PostgreSQL-Verbindungen werden als solche beibehalten und müssen nicht zusätzlich konfiguriert werden. Siehe auch "Datenbankverbindungen" weiter unten.
- Wenn das Mapping benutzerdefinierte Funktionsaufrufe enthält (z.B. Aufrufe von .dll- oder .class-Dateien), werden diese Abhängigkeiten nicht zusammen mit dem Mapping bereitgestellt, da sie vor der Laufzeit nicht bekannt sind. Kopieren Sie diese in diesem Fall manuell auf den Zielrechner. Der Pfad der .dll- oder .class-Datei auf dem Server muss derselbe sein, der in MapForce im Dialogfeld "Bibliotheken verwalten" definiert wurde, z.B.:



- In einigen Mappings werden mehrere Input-Dateien über einen Platzhalter-Pfad ausgelesen (siehe Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 1371). In diesem Fall sind die Namen der Input-Dateien vor der Laufzeit noch nicht bekannt und können daher nicht bereitgestellt werden. Damit das Mapping erfolgreich ausgeführt werden kann, muss die Input-Datei auf dem Zielrechner vorhanden sein.
- Wenn der Mapping-Ausgabepfad Verzeichnisse enthält, müssen diese auf dem Zielrechner vorhanden sein, da sonst bei Ausführung des Mappings ein Fehler generiert wird. Im Gegensatz dazu werden in MapForce nicht vorhandene Verzeichnisse automatisch generiert, wenn die Option Output-Datei als temporäre Datei generieren aktiviert ist (siehe Ändern der MapForce-Optionen 1539).
- Wenn im Mapping ein Webservice aufgerufen wird, für den eine HTTPS-Authentifizierung mit einem Client-Zertifikat erforderlich ist, muss das Zertifikat ebenfalls an den Zielrechner übertragen werden (siehe Verwaltung digitaler Zertifikate 664.
- Wenn im Mapping eine Verbindung zu dateibasierten Datenbanken wie Microsoft Access und SQLite hergestellt wird, muss die Datenbank manuell auf den Zielrechner transferiert werden oder in einem freigegebenen, sowohl Quell- als auch Zielrechner zugänglichen Verzeichnis gespeichert werden und

 © 2016-2022 Altova GmbH

von dort aus referenziert werden, siehe "Dateibasierte Datenbanken", weiter unten.

Übertragbarmachen von Pfaden

Wenn Sie beabsichtigen, das Mapping auf einem Server auszuführen, müssen Sie sicherstellen, dass es den jeweils geltenden Pfadkonventionen entspricht und dass eine unterstützte Datenbankverbindung verwendet wird.

Um Pfade auf Nicht-Windows-Betriebssysteme übertragen zu können, verwenden Sie beim Erstellen des Mappings in MapForce relative anstelle von absoluten Pfaden:

1. Öffnen Sie die gewünschte Mapping-Design-Datei (.mfd) mit MapForce unter Windows.
2. Wählen Sie im Menü **Datei** den Befehl **Mapping-Einstellungen** und deaktivieren Sie das Kontrollkästchen **Pfade im generierten Code absolut machen**, falls es aktiviert ist.
3. Öffnen Sie für jede Mapping-Komponente das Dialogfeld **Eigenschaften** (z.B. durch Doppelklick auf die Titelleiste) und ändern Sie alle Dateipfade von absoluten in relative um. Aktivieren Sie außerdem das Kontrollkästchen **Alle Dateipfade relativ zur MFD-Datei speichern**. Sie können alle Input-Dateien und Schemas aus praktischen Gründen in denselben Ordner wie das Mapping kopieren und diese einfach über den Dateinamen referenzieren.

Nähere Informationen zum Arbeiten mit relativen und absoluten Pfaden bei der Erstellung eines Mappings finden Sie unter Verwendung relativer und absoluter Pfade 84.

Beachten Sie, dass sowohl MapForce Server als auch FlowForce Server ein so genanntes Arbeitsverzeichnis ("working directory") unterstützen, anhand dessen alle relativen Pfade aufgelöst werden, siehe auch Pfade in verschiedenen Ausführungsumgebungen 87. Das Arbeitsverzeichnis wird zur Mapping-Laufzeit folgendermaßen definiert:

- in FlowForce Server durch Bearbeitung des Auftragsparameters "Working-directory".
- in der MapForce Server API über die Eigenschaft `WorkingDirectory` der COM und der .NET API oder über die Methode `setWorkingDirectory` der Java API.
- in der MapForce Server-Befehlszeile ist das Arbeitsverzeichnis das aktuelle Verzeichnis der Befehlszeile.

Datenbankverbindungen

Beachten Sie, dass ADO-, ADO.NET- und ODBC-Verbindungen auf Linux- und macOS-Rechnern nicht unterstützt werden. Wenn es sich beim Zielrechner daher um einen Linux- oder macOS-Rechner handelt, werden diese Verbindungen in JDBC konvertiert, wenn Sie das Mapping auf FlowForce Server bereitstellen oder das Mapping zu einer MapForce Server-Ausführungsdatei kompilieren. In diesem Fall haben Sie die folgenden Möglichkeiten, bevor Sie das Mapping bereitstellen oder es zu einer Server-Ausführungsdatei kompilieren:

- Erstellen Sie in MapForce eine JDBC-Verbindung zur Datenbank, bevor Sie das Mapping bereitstellen (siehe Einrichten einer JDBC-Verbindung 190)
- Füllen Sie in MapForce die JDBC-Datenbankverbindungsinformationen im Abschnitt "JDBC-spezifische Einstellungen" der Datenbankkomponente aus (siehe Datenbank-Komponenteneinstellungen 286).

Wenn im Mapping eine native Verbindung zu einer PostgreSQL- oder SQLite-Datenbank verwendet wird, wird die native Verbindung beibehalten und es wird keine JDBC-Konvertierung durchgeführt, siehe Datenbankmappings in verschiedenen Ausführungsumgebungen 166. Wenn im Mapping eine Verbindung zu einer dateibasierten Datenbank wie Microsoft Access und SQLite hergestellt wird, müssen zusätzliche Konfigurationsschritte durchgeführt werden, siehe "Dateibasierte Datenbanken" weiter unten.

Für die Ausführung von Mappings mit JDBC-Verbindungen muss Java Runtime Environment oder Java

Development Kit auf dem Server-Rechner installiert sein. Dabei kann es sich entweder um Oracle JDK oder einen Open Source Build wie z.B. Oracle OpenJDK handeln.

- Die JAVA_HOME-Umgebungsvariable muss auf das JDK-Installationsverzeichnis verweisen.
- Unter Windows hat ein Java Virtual Machine-Pfad aus der Windows Registry Vorrang vor der JAVA_HOME-Variablen.
- Die JDK-Plattform (64-Bit, 32-Bit) muss mit der von MapForce Server identisch sein, da Sie sonst eventuell einen Fehler erhalten, weil "kein Zugriff auf JVM besteht".

So richten Sie unter Linux oder macOS eine JDBC-Verbindung ein:

1. Laden Sie den vom Datenbankanbieter bereitgestellten JDBC-Treiber herunter und installieren Sie ihn auf dem Betriebssystem. Wählen Sie die 32-Bit-Version aus, wenn Ihr Betriebssystem mit 32 Bit ausgeführt wird, und die 64-Bit-Version, wenn Ihr Betriebssystem mit 64 Bit ausgeführt wird.
2. Definieren Sie für die Umgebungsvariablen den Pfad, unter dem der JDBC-Treiber installiert ist. Normalerweise müssen Sie die CLASSPATH-Variable und eventuell noch einige weitere Variablen definieren. Nähere Informationen dazu, welche Umgebungsvariablen konfiguriert werden müssen, finden Sie in der mit dem JDBC-Treiber mitgelieferten Dokumentation.

Anmerkung: Unter macOS werden die installierten JDBC-Bibliotheken im Verzeichnis **/Library/Java/Extensions** gesucht. Es wird daher empfohlen, den JDBC-Treiber unter diesem Pfad zu entpacken; andernfalls müssen Sie das System so konfigurieren, dass es die JDBC-Bibliothek unter dem Pfad sucht, unter dem Sie den JDBC-Treiber installiert haben.

Oracle Instant Client-Verbindungen auf macOS

Wenn Sie über den **Oracle Database Instant Client** eine Verbindung zu einer Oracle-Datenbank herstellen, gelten auf macOS die folgenden Voraussetzungen: Voraussetzungen:

- Es muss Java 8.0 oder höher installiert sein. Sie können auch über die **JDBC Thin for All Platforms**-Bibliothek eine Verbindung herstellen, wenn auf dem Mac-Rechner eine Java-Version älter als Version 8 läuft. In diesem Fall können Sie die unten stehende Anleitung ignorieren.
- Oracle Instant Client muss installiert sein. Sie können den Oracle Instant Client von der offiziellen Oracle Download-Seite herunterladen. Beachten Sie, dass auf dieser Seite mehrere Instant Client-Pakete zum Download zur Verfügung stehen. Stellen Sie sicher, dass Sie ein Paket mit Oracle Call Interface (OCI)-Unterstützung (z.B. Instant Client Basic) auswählen. Stellen Sie außerdem sicher, dass Sie die 32-Bit-Version auswählen, falls Ihr Betriebssystem auf 32 Bit läuft, und die 64-Bit-Version, falls Ihr Betriebssystem auf 64 Bit läuft.

Nachdem Sie den Oracle Instant Client heruntergeladen und entpackt haben, bearbeiten Sie die mit dem Installer mitgelieferte Eigenschaftsliste (plist-Datei), sodass die folgenden Umgebungsvariablen auf den entsprechenden Treiberpfad verweisen, z.B.:

Variable	Beispielwert
CLASSPATH	/opt/oracle/instantclient_11_2/ojdbc6.jar:/opt/oracle/instantclient_11_2/ojdbc5.jar
TNS_ADMIN	/opt/oracle/NETWORK_ADMIN

© 2016-2022 Altova GmbH

ORACLE_HOME	/opt/oracle/instantclient_11_2
DYLD_LIBRARY_PATH	/opt/oracle/instantclient_11_2
PATH	$PATH:/opt/oracle/instantclient_11_2

Anmerkung: Bearbeiten Sie die obigen Beispielwerte, sodass sie auf die Pfade verweisen, unter denen die Oracle Instant Client-Dateien auf Ihrem Betriebssystem installiert sind.

Dateibasierte Datenbanken

Dateibasierte Datenbanken wie Microsoft Access und SQLite sind in dem auf FlowForce Server bereitgestellten Paket oder in der kompilierten MapForce Server-Ausführungsdatei nicht enthalten. Wenn der Quellrechner daher nicht derselbe wie der Zielrechner ist, gehen Sie folgendermaßen vor:

1. Klicken Sie in MapForce mit der rechten Maustaste auf das Mapping und deaktivieren Sie das Kontrollkästchen **Pfade im generierten Code absolut machen** (siehe Ändern der Mapping-Einstellungen 116).
2. Klicken Sie mit der rechten Maustaste im Mapping auf die Datenbankkomponente und fügen Sie eine Verbindung zu einer Datenbankdatei über einen relativen Pfad hinzu, siehe Definieren des Pfads zu dateibasierten Datenbanken 84. Eine einfache Methode, um Pfadprobleme zu vermeiden, ist, das Mapping-Design (die .mfd-Datei) im selben Verzeichnis wie die Datenbankdatei zu speichern und letztere vom Mapping aus nur über den Dateinamen (d.h. über einen relativen Pfad) zu referenzieren.
3. Kopieren Sie die Datenbankdatei in ein Verzeichnis auf dem Zielrechner (nennen wir es das "Arbeitsverzeichnis"). Dieses Verzeichnis benötigen Sie später, um das Mapping auf dem Server auszuführen (siehe unten).

Um ein solches Mapping auf dem Server auszuführen, wählen Sie eine der folgenden Methoden:

- Wenn das Mapping unter Verwaltung von FlowForce Server mit MapForce Server ausgeführt wird, konfigurieren Sie den FlowForce Server-Auftrag so, dass das zuvor erstellte Arbeitsverzeichnis verwendet wird. Die Datenbankdatei muss sich in diesem Arbeitsverzeichnis befinden. Ein Beispiel dazu finden Sie in der FlowForce Server-Dokumentation im Kapitel "Bereitstellen eines Auftrags aus Web-Dienst" (https://www.altova.com/manual/de/FlowForceServer/FlowForceServerAdvanced/index.html?fs_example_web_service.htm)
- Wenn das Mapping von MapForce Server alleine über die Befehlszeile ausgeführt wird, ändern Sie das aktuelle Verzeichnis in das Arbeitsverzeichnis (z.B. `cd pfad\zum\arbeitsverzeichnis`), bevor Sie den MapForce Server-Befehl `run` aufrufen.
- Wenn das Mapping über die MapForce Server API ausgeführt wird, definieren Sie das Arbeitsverzeichnis programmatisch, bevor Sie das Mapping ausführen. Zu diesem Zweck steht für das MapForce Server-Objekt in der COM und .NET API die Eigenschaft `WorkingDirectory` zur Verfügung. In der Java API steht dafür die Methode `setWorkingDirectory` zur Verfügung.

Wenn es sich sowohl beim Quell- als auch beim Zielrechner um Windows-Rechner im lokalen Netzwerk handelt, wäre eine alternative Methode, das Mapping so zu konfigurieren, dass die Datenbankdatei folgendermaßen aus einem gemeinsamen, freigegebenen Verzeichnis gelesen wird:

1. Speichern Sie die Datenbankdatei in einem gemeinsamen freigegebenen Verzeichnis, auf das sowohl der Quell- als auch der Zielrechner Zugriff hat.
2. Klicken Sie mit der rechten Maustaste im Mapping auf die Datenbankkomponente und fügen Sie über

einen absoluten Pfad eine Verbindung zur Datenbankdatei hinzu (siehe Definieren des Pfads zu dateibasierten Datenbanken 84).

Globale Ressourcen

Wenn ein Mapping anstelle von direkten Pfaden oder Datenbankverbindungen Referenzen auf globale Ressourcen enthält, können Sie auch serverseitig globale Ressourcen verwenden. Wenn Sie ein Mapping zu einer MapForce Server-Ausführungsdatei (.mfx) kompilieren, bleiben die Referenzen auf globale Ressourcen intakt, so dass Sie diese zur Mapping-Laufzeit auf Seite des Servers bereitstellen können. Wenn Sie ein Mapping auf FlowForce Server bereitstellen, können Sie optional auswählen, ob darin die Ressourcen auf dem Server verwendet werden sollen.

Damit Mappings (oder, im Fall von FlowForce Server, Mapping-Funktionen) erfolgreich ausgeführt werden können, muss die eigentliche Datei, der eigentliche Ordner bzw. die Datenbankverbindungsinformationen, die Sie als globale Ressourcen bereitstellen mit der Server-Umgebung kompatibel sein. So muss z.B. bei Ausführung des Mappings auf einem Linux-Server bei Datei- und Ordnerpfaden die Linux-Pfadkonvention verwendet werden. Ebenso müssen als Datenbankverbindungen definierte globale Ressourcen auf dem Server-Rechner möglich sein.

Nähere Informationen dazu finden Sie unter Globale Ressourcen in MapForce Server 1480 und Globale Ressourcen in FlowForce Server 1480..

XBRL-Taxonomiepakete

Wenn Sie ein Mapping bereitstellen, in dem XBRL-Taxonomiepakete 1545 referenziert werden, sammelt MapForce alle externen Referenzen aus dem Mapping und löst diese dann unter Verwendung der aktuellen Konfiguration und der aktuell installierten Taxonomiepakete auf. Falls es aufgelöste externe Referenzen gibt, die auf ein Taxonomiepaket verweisen, so wird das Taxonomiepaket zusammen mit dem Mapping bereitgestellt. FlowForce Server verwendet dieses Paket in dem Zustand, in dem es sich bei der Bereitstellung befand, um das Mapping auszuführen. Um das von FlowForce Server verwendete Taxonomiepaket zu aktualisieren, müssen Sie es in MapForce ändern und das Mapping erneut bereitstellen.

Beachten Sie, dass der Root-Katalog von MapForce Server einen Einfluss darauf hat, wie Taxonomien auf dem Zielrechner aufgelöst werden. Der Root-Katalog befindet sich unter dem folgenden Pfad relativ zum MapForce Server-Installationsverzeichnis: **etc/RootCatalog.xml**.

Mit einem Mapping bereitgestellte Taxonomiepakete werden verwendet, wenn der Root-Katalog von MapForce Server nicht bereits ein solches Paket enthält oder kein Paket enthält, das für dasselbe URL-Präfix definiert ist. Der Root-Katalog von MapForce Server hat Vorrang vor einer bereitgestellten Taxonomie.

Wenn MapForce Server alleine (ohne FlowForce Server) ausgeführt wird, kann der für das Mapping zu verwendende Root-Katalog folgendermaßen definiert werden:

- in der Befehlszeile durch Hinzufügen der Option `-catalog` zum Befehl `run`.
- in der MapForce Server API durch Aufruf der Methode `SetOption` und Bereitstellen des String `"catalog"` als erstes Argument und des Pfads zum Root-Katalog als zweites Argument.

Wenn in einem Mapping XBRL-Komponenten mit Tabellen-Linkbases verwendet werden, muss das Taxonomiepaket oder die Taxonomiepaketkonfigurationsdatei dem Mapping folgendermaßen zur Laufzeit zur Verfügung gestellt werden:

- Fügen Sie in der MapForce Server-Befehlszeile die Option `--taxonomy-package` oder `--taxonomy-packages-config-file` zum Befehl `run` hinzu.

 © 2016-2022 Altova GmbH

- Rufen Sie über die MapForce Server API die Methode `SetOption` auf. Das erste Argument muss entweder `"taxonomy-package"` oder `"taxonomy-packages-config-file"` sein. Das zweite Argument muss der eigentliche Pfad zum Taxonomiepaket (oder zur Taxonomiepaketkonfigurationsdatei) sein.

10.4 Kompilieren von Mappings zu MapForce Server-Ausführungsdateien

Wenn als Zielsprache eines mit MapForce erstellten Mappings BUILT-IN ausgewählt ist, kann es nicht nur mit MapForce, sondern auch mit MapForce Server ausgeführt werden (siehe Informationen zu MapForce Server 1434). Es gibt zwei Methoden, wie ein Mapping mit MapForce Server ausgeführt werden kann:

- Wenn MapForce Server im Standalone-Modus (d.h. ohne FlowForce Server) ausgeführt wird, muss das Mapping, wie unten beschrieben, zu einer Server-Ausführungsdatei (.mfx) kompiliert werden. Sie können die .mfx.Datei über die Befehlszeile mit dem Befehl `run` ausführen. Sie können das Mapping auch durch Aufruf der `run`-Methode der MapForce Server API aufrufen. Nähere Informationen dazu finden Sie in der Dokumentation zu MapForce Server (https://www.altova.com/manual/de/MapForceServer/).
- Wenn MapForce Server unter Verwaltung von FlowForce Server ausgeführt wird, kann das Mapping auf einem Rechner bereitgestellt werden, auf dem sowohl MapForce Server als auch FlowForce Server läuft. Nähere Informationen zu diesem Szenario finden Sie unter Bereitstellen von Mappings auf FlowForce Server 1445.

Voraussetzungen

Siehe Vorbereiten von Mappings für die Server-Ausführung 1435.

So kompilieren Sie ein Mapping zu einer MapForce Server-Ausführungsdatei (.mfx):

1. Öffnen Sie ein Mapping in MapForce (z.B. **myMapping.mfd)**.
2. Wählen Sie die Menüoption **Datei | Zu MapForce Server-Ausführungsdatei kompilieren.**
3. Wählen Sie den Ordner aus, in dem die *.mfx-Datei gespeichert werden soll und ändern Sie gegebenenfalls den Dateinamen.
4. Klicken sie auf **Speichern**. Die Datei **myMapping.mfx** wird im angegebenen Ordner generiert.

So kompilieren Sie ein Mapping über die MapForce-Befehlszeile zu einer MapForce Server-Ausführungsdatei (.mfx):

- Führen Sie MapForce über die Befehlszeile aus und definieren Sie die Mapping-Datei und die Befehlszeilenoption `/COMPILE`.

Mit dem folgenden Befehl wird das Mapping **C:\Users\altova\Documents\Altova\MapForce2023\MapForceExamples\SimpleTotal.mfd** zu einer MapForce Server-Ausführungsdatei kompiliert, die im Ausgabezielverzeichnis **C:\Users\altova\Desktop** erstellt wird.

```
"C:\Program Files (x86)\Altova\MapForce2023\MapForce.exe" "C:
\Users\altova\Documents\Altova\MapForce2023\MapForceExamples" /COMPILE "C:
\Users\altova\Desktop"
```

Siehe auch MapForce-Befehlszeilenschnittstelle 1456.

© 2016-2022 Altova GmbH

Inhalt der .mfx-Datei

Die .mfx-Datei enthält die folgenden Daten:

- den Mapping-Algorithmus, der alle aus anderen Mappings importierten benutzerdefinierten Funktionen (UDFs) enthält.
- von den Komponenten referenzierte Input- und Output-Dateinamen. Die Pfade sind je nach Mapping-Einstellungen entweder absolut oder relativ, siehe Pfade in verschiedenen Ausführungsumgebungen 87 .
- Wenn das Mapping XML-Komponenten enthält, so enthält der Mapping-Algorithmus Informationen über das zum Ausführen des Mappings erforderliche XML-Schema.
- die Datenbankverbindungsinformationen, wenn das Mapping Datenbankverbindungen enthält. Die Passwörter sind verschlüsselt.
- je nach Mapping aus den folgenden Dateien[1] stammende Informationen:
  - EDI- und FlexText-Konfigurationen
  - JSON-Schemas
  - .proto-Dateien
- Wenn das Mapping Anmeldeinformationen enthält, werden die sensiblen Daten in verschlüsselter Form inkludiert, wenn Sie das Kontrollkästchen **Verschlüsselt in MapForce Server-Ausführungsdatei und Mapping-Bereitstellung inkludieren** aktiviert haben. Andernfalls werden die Felder mit den sensiblen Daten übersprungen. Nähere Informationen dazu finden Sie unter Anmeldeinformationen 1353.

Fußnoten:

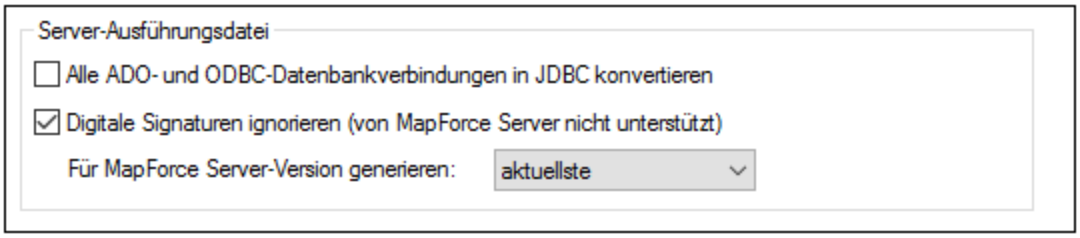
1. Aus EDI- oder FlexText-Konfigurationen, JSON-Schemas oder .proto-Dateien stammende Informationen werden in kompilierter und verarbeiteter Form in die .mfx-Datei eingebettet. Es ist nicht möglich, die Originalquelldateien aus einer kompilierten .mfx-Datei zu extrahieren. Die Daten werden in dem Zustand kompiliert, in dem sie sich zum Zeitpunkt der Kompilierung befinden. Wenn Sie diese später ändern, muss die .mfx-Datei neu kompiliert werden.

Die im Mapping verwendeten Input-Instanzdateien (XML, CSV, Text, EDI, Excel, JSON, XBRL) werden nicht in die kompilierte mfx-Datei inkludiert. Dies gilt auch für dateibasierte Datenbanken wie Access oder SQLite. Wenn im Mapping im Zusammenhang mit der Unterstützung von XBRL-Tabellen XBRL-Taxonomien referenziert werden, werden die Taxonomieinformationen nicht in der .mfx-Datei kompiliert. Nähere Informationen finden Sie unter Vorbereiten von Mappings für die Server-Ausführung 1435.

Kompilieren von Mappings zu einer spezifischen MapForce Server-Version

Wenn die Version Ihres MapForce Servers älter ist als die von MapForce, kann Ihr MapForce Server mit einer neueren Version von MapForce erstellte .mfx-Dateien eventuell nicht ausführen, da seither wahrscheinlich neue Funktionen hinzugekommen sind. In diesem Fall können Sie die .mfx-Datei für eine spezifische Version von MapForce Server kompilieren. Gehen Sie dazu folgendermaßen vor:

1. Klicken Sie im Menü **Extras** auf **Optionen** und anschließend auf **Code-Generierung**.
2. Wählen Sie unter **Server-Ausführungsdatei** neben **MapForce Server-Version generieren** die erforderliche MapForce Server-Version aus der Dropdown-Liste aus.



Denken Sie daran, die Option entsprechend zu ändern, sobald Sie eine neuere MapForce Server-Version haben. Wenn es keinen Grund gibt, die mfx-Datei für eine bestimmte Version von MapForce Server zu kompilieren, wählen Sie die Option "aktuellste" (dies ist die Standardoption). Wenn diese Option ausgewählt ist, wird die .mfx-Datei für die aktuellste Version von MapForce Server kompiliert. Dadurch können darin auch die neuesten Funktionen und Verbesserungen, die in früheren Versionen eventuell noch nicht zur Verfügung standen, genutzt werden.

Um in der Befehlszeile eine MapForce Server-Zielversion zu definieren, führen Sie den Befehl `/COMPILE` mit der Option `/MFXVERSION` aus, z.B.:

```
"C:\Program Files (x86)\Altova\MapForce2023\MapForce.exe" /COMPILE /MFXVERSION:2023
```

Siehe auch MapForce-Befehlszeilenschnittstelle 1456.

Andere Optionen

Außerdem wirken sich die folgenden Optionen auf die Kompilierung von MapForce Server-Ausführungsdateien aus:

Alle ADO- und ODBC-Datenbankverbindungen in JDBC konvertieren	Wenn die Option aktiv ist, werden ADO-, ADO.NET und ODBC-Datenbankverbindungen mit Hilfe des JDBC-Treibers und der im Dialogfeld "Komponenteneinstellungen" für die Datenbank definierten Datenbank-URL in JDBC transformiert (siehe Datenbank-Komponenteneinstellungen 286). Die JDBC-Verbindung wird implizit verwendet, wenn der Zielrechner ein Linux- oder macOS-Server ist.
Digitale Signaturen ignorieren (von MapForce Server nicht unterstützt)	Diese Option gilt nur für MapForce Enterprise. Sie ist standardmäßig deaktiviert. Fall im Mapping digitale XML-Signaturen verwendet werden, so werden diese mit dieser Option übersprungen, da MapForce Server digitale XML-Signaturen nicht unterstützt.

So zeigen Sie diese Optionen an oder ändern diese:

- Klicken Sie im Menü **Extras** auf **Optionen** und anschließend auf **Code-Generierung**.

Diese Optionen stehen auch über die Befehlszeilenschnittstelle zur Verfügung. Nähere Informationen dazu finden Sie unter MapForce-Befehlszeilenschnittstelle 1456.

© 2016-2022 Altova GmbH

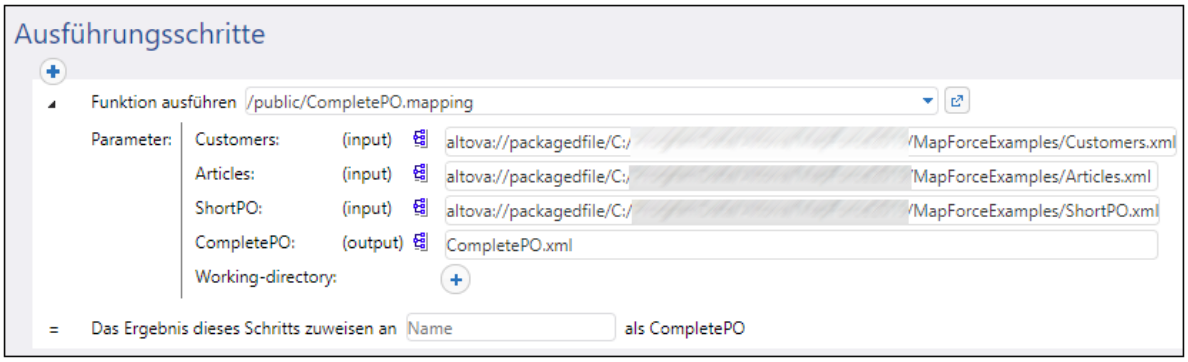
10.5 Bereitstellen von Mappings auf FlowForce Server

Bei der Bereitstellung eines Mappings auf FlowForce Server sammelt MapForce die vom jeweiligen Mapping verwendeten Ressourcen in einem Objekt und übergibt dieses über HTTP (oder HTTPS, falls konfiguriert) an den Rechner, auf dem FlowForce Server ausgeführt wird. MapForce Mappings werden normalerweise auf FlowForce Server bereitgestellt, um ihre Ausführung mit Hilfe von FlowForce Server-Aufträgen zu automatisieren. Nachdem ein Mapping bereitgestellt wurde, können Sie anhand dieses Mappings einen FlowForce Server-Auftrag erstellen und darin alle Funktionalitäten im Zusammenhang mit Aufträgen nutzen (z.B. können Sie benutzerdefinierte Bedingungen für den Start des Auftrags definieren, den Auftrag als Webservice bereitstellen, usw.).

Anmerkung: Mit "Quellrechner" wird der Computer bezeichnet, auf dem MapForce installiert ist und mit "Zielrechner" wird der Computer bezeichnet, auf dem FlowForce Server installiert ist. Im einfachsten Szenario handelt es sich hierbei um denselben Computer. In einem komplexeren Szenario läuft MapForce auf einem Windows-Rechner, während FlowForce Server auf einem Linux- oder macOS-Rechner ausgeführt wird.

Das auf FlowForce bereitgestellte Paket enthält die folgenden Komponenten:

- das Mapping selbst. Nachdem das Mapping bereitgestellt wurde, steht es als Mapping-Funktion (.mapping) unter dem von Ihnen definierten Pfad auf der FlowForce Server-Verwaltungsoberfläche zur Verfügung. Alle Quellkomponenten werden zu Input-Argumenten, Zielkomponenten werden zu Output-Argumenten dieser Funktion.



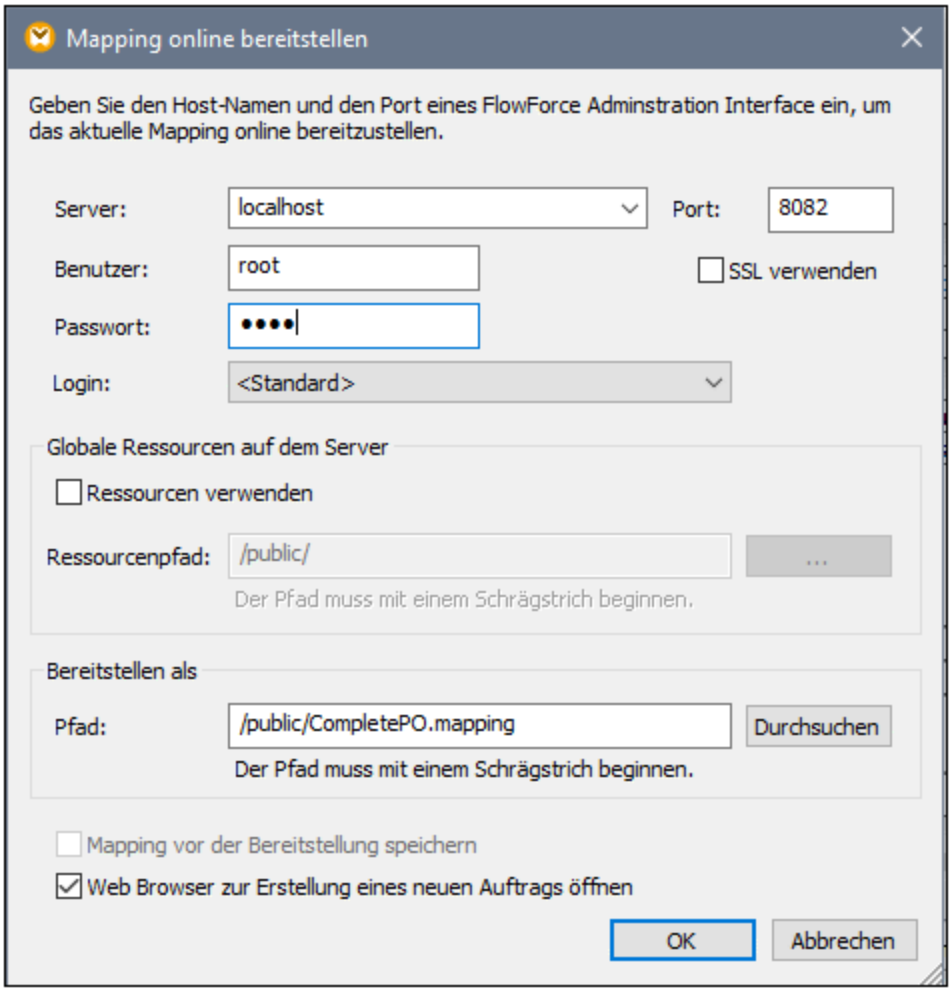
- alle Arten von Input-Instanzdateien (XML, CSV, Text, EDI, Excel, JSON, XBRL), die vom Mapping verwendet werden.
- Wenn im Mapping EDI- und FlexText-Konfigurationen verwendet werden, werden diese in dem Zustand bereitgestellt, in dem sie sich zum Zeitpunkt der Bereitstellung auf dem Quellrechner befinden. Wenn Sie die EDI- oder FlexText-Konfiguration nach der Bereitstellung ändern, muss das Mapping erneut auf FlowForce Server bereitgestellt werden.
- Wenn im Mapping XBRL-Taxonomien oder XBRL-Taxonomiepakete verwendet werden, werden diese in dem Zustand bereitgestellt, in dem Sie sich zum Zeitpunkt der Bereitstellung auf dem Quellrechner befinden.

Voraussetzungen

Siehe Vorbereiten von Mappings für die Server-Ausführung 1435

Bereitstellen des Mappings auf FlowForce Server

1. Starten Sie MapForce und stellen Sie sicher, dass als Transformationssprache BUILT-IN ausgewählt ist (Klicken Sie entweder auf die Symbolleisten-Schaltfläche **Built-in** oder wählen Sie den Menübefehl **Ausgabe | Built-In Ausführungsprozessor**).
2. Klicken Sie im Menü **Datei** auf **Auf FlowForce Server bereitstellen**. Daraufhin wird das Dialogfeld "Mapping online bereitstellen" geöffnet.



3. Geben Sie die Einstellungen für die Bereitstellung ein (Beschreibung siehe unten) und klicken Sie auf OK. Wenn Sie das Kontrollkästchen **Web Browser zur Erstellung eines neuen Auftrags öffnen** aktiviert haben, wird die FlowForce Server-Verwaltungsoberfläche im Browser geöffnet und Sie können sofort mit der Erstellung eines FlowForce Server-Auftrags beginnen.

In der folgenden Tabelle sind die Einstellungen für die Bereitstellung eines Mappings aus dem Dialogfeld "Mapping online bereitstellen" beschrieben.

Einstellung	Beschreibung
Server, Port, SSL verwenden	Geben Sie den Server Host-Namen (oder die IP-Adresse) und den Port von

© 2016-2022 Altova GmbH

Einstellung	Beschreibung
	FlowForce Server ein. Wenn FlowForce Server auf demselben Rechner und unter dem Standard-Port ausgeführt wird, dann wäre dies z.B. **localhost** und **8082**. Wenn Sie sich nicht sicher sind, loggen Sie sich auf der FlowForce Server Web-Verwaltungsoberfläche ein und überprüfen Sie, welche IP-Adresse und welcher Port in der Adressleiste des Web Browsers angezeigt werden. Stellen Sie bei Verbindungsproblemen sicher, dass der Rechner, auf dem FlowForce Server ausgeführt wird, so konfiguriert ist, dass er eingehende Verbindungen an der dafür vorgesehenen Adresse und am dafür vorgesehenen Port zulässt. Um das Mapping über eine SSL-verschlüsselte Verbindung bereitzustellen, aktivieren Sie das Kontrollkästchen **SSL verwenden**. Es wird dabei vorausgesetzt, dass FlowForce Server bereits so konfiguriert ist, dass SSL-Verbindungen zugelassen werden. Nähere Informationen dazu finden Sie in der FlowForce Server-Dokumentation (https://www.altova.com/de/documentation).
Benutzer und Passwort	Welcher Benutzername und welches Passwort eingegeben werden müssen, hängt vom Wert in der Login-Dropdown-Liste ab (siehe nächste Option). Wenn die Login-Dropdown-Liste auf **<Standard>** oder **Direkt** gesetzt ist, geben Sie Ihren FlowForce Server Benutzernamen und Ihr Passwort dafür ein. Geben Sie andernfalls Ihren Domain-Benutzernamen und Ihr Domain-Passwort ein und wählen Sie den Domain-Namen aus der Login-Dropdown-Liste aus.
Login	Wenn in FlowForce Server die Directory Service-Integration aktiviert ist, wählen Sie den Domain-Namen aus dieser Dropdown-Liste aus und geben Sie Ihre Domain-Anmeldeinformationen in die Felder "Benutzername" und "Passwort" ein (siehe vorherige Option).
Ressourcen verwenden, Ressourcenpfad	Aktivieren Sie das Kontrollkästchen **Ressourcen verwenden**, wenn die Mapping-Funktion nach ihrer Bereitstellung auf dem Server Ressourcen[1462] verwenden soll. Wenn Sie das Kontrollkästchen aktivieren, müssen Sie auch den Pfad zur entsprechenden Ressource auf dem Server in das Textfeld **Ressourcenpfad** eingeben. Um die Ressource auszuwählen, klicken Sie auf die **Auslassungszeichen**-Schaltfläche. Wenn auf dem Server noch keine Ressourcen zur Auswahl bereit stehen, klicken Sie auf **Globale Ressourcen bereitstellen** und stellen Sie die erforderlichen globalen Ressourcen auf dem Server bereit. Nähere Informationen dazu finden Sie unter Bereitstellen von globalen Ressourcen auf FlowForce Server[1481]. Wenn Sie das Kontrollkästchen **Ressourcen verwenden** nicht aktivieren, werden alle globalen Ressourcen auf Basis der aktuell ausgewählten Konfiguration aufgelöst. Für die Mapping-Funktion auf dem Server werden daraufhin keine globalen Ressourcen mehr benötigt, sondern es wird stattdessen der aufgelöste Wert verwendet.

Einstellung	Beschreibung
Pfad	Klicken Sie auf **Durchsuchen** und wählen Sie den Pfad, unter dem die Mapping-Funktion in der FlowForce Server Containerhierarchie gespeichert werden soll. Standardmäßig ist als Pfad der Container **/public** von FlowForce Server eingestellt. Sie können über das Dialogfeld auch einen neuen Container erstellen oder vorhandene Container und Mappings löschen, vorausgesetzt Sie haben die erforderlichen FlowForce Server-Berechtigungen und Rechte dazu.
Mapping vor der Bereitstellung speichern	Diese Option steht zur Verfügung, wenn Sie ein nicht gespeichertes Mapping bereitstellen. Aktivieren Sie dieses Kontrollkästchen, um das Mapping vor der Bereitstellung zu speichern.
Web Browser zur Erstellung eines neuen Auftrags öffnen	Wenn Sie dieses Kontrollkästchen aktiviert haben, wird nach der Bereitstellung die FlowForce Server-Verwaltungsoberfläche im Browser geöffnet und Sie können sofort mit der Erstellung eines FlowForce Server-Auftrags beginnen.

Fehlerbehebung

In der folgenden Tabelle sind häufige Probleme bei der Bereitstellung eines Mappings und deren Lösung aufgelistet.

Problem	Lösung

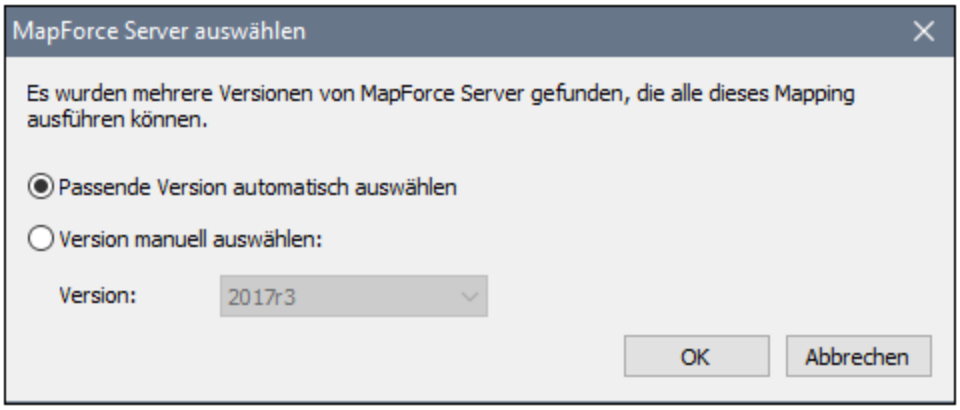
Bei der Bereitstellung des Mapping erhalte ich die folgende Fehlermeldung:
 	
`I/O-Operation an der Datei.... fehlgeschlagen.`
`I/O-Fehler 28: Es konnte keine Verbindung zu <server> port 8082 hergestellt werden. Zeit überschritten`
`Systemfehler 10060: Versuch einer Verbindung ist fehlgeschlagen, da der Verbindungsteilnehmer nach Ablauf einer bestimmten Zeit nicht ordnungsgemäß geantwortet hat, oder die hergestellte Verbindung ist fehlgeschlagen, da der verbundene Host nicht geantwortet hat.`
	Stellen Sie sicher, dass der *FlowForce Web Server*-Dienst auf dem Zielrechner läuft und so konfiguriert ist, dass er am angegebenen Port (standardmäßig **8082**) empfangsbereit ist. Stellen Sie außerdem sicher, dass über diesen Port eingehende Verbindungen nicht durch eine Firewall blockiert werden.
	
	Auch der *FlowForce Server*-Dienst muss gestartet sein, damit der Auftrag bereitgestellt werden kann.

Bei der Bereitstellung des Mapping erhalte ich die folgende Fehlermeldung:
 	
`I/O-Operation an der Datei.... fehlgeschlagen.`
`I/O-Fehler 413: Payload to groß`
	Dieser Fehler kann auftreten, wenn eine Input-Datei des bereitgestellten Mappings die maximale Größe der von FlowForce Server akzeptierten HTTP-Requests übersteigt (ca. 100 MB). Sie können das Limit über die Option `max_request_body_size` (in Bytes) in den Dateien **flowforceweb.ini** und **flowforce.ini** erhöhen. Nähere Informationen dazu finden Sie in der FlowForce Server-Dokumentation.

© 2016-2022 Altova GmbH

Auswahl der Server-Version (nur Windows)

Wenn auf dem Server, auf dem Sie das Mapping bereitstellen, mehrere Versionen von MapForce Server unter FlowForce Server-Verwaltung laufen (gilt nur für Windows Server), werden Sie zusätzlich dazu auch aufgefordert, die Version von MapForce Server, mit der dieses Mapping ausgeführt werden soll, zu definieren.



Anmerkung: Das Dialogfeld wird angezeigt, wenn das FlowForce Server-Installationsverzeichnis .tool-Dateien für jede unter FlowForce Server-Verwaltung laufende MapForce Server-Version enthält. Standardmäßig wird zu diesem Verzeichnis automatisch eine MapForce Server .tool-Datei hinzugefügt, wenn Sie MapForce Server als Teil der FlowForce Server Installation installieren. Die .tool-Dateien sind in FlowForce unter dem folgenden Pfad gespeichert: **C:\Programme\Altova\FlowForceServer2023\tools**. Wenn Sie zusätzliche Versionen von MapForce Server haben, die unter FlowForce Server-Verwaltung ausgeführt werden sollen, müssen deren .tool-Dateien eventuell manuell in das obige Verzeichnis kopiert werden. Die .tool-Datei von MapForce Server befindet sich unter: **C:\Programme\Altova\MapForceServer2023\etc**.

10.6 AS2-Integration

AS2 (Applicability Statement 2) ist eine Spezifikation für den sicheren Austausch von Dateien über das Internet. AS2 wird von Unternehmen dazu verwendet, um in erster Linie EDIINT- (EDI über Internet) und XML-Dateien entweder über HTTP oder über HTTPS auszutauschen.

Die FlowForce Server Advanced Edition bietet eine Funktionalität zum Senden von AS2-Nachrichten an Handelspartner bzw. zum Empfangen von AS2-Nachrichten von Handelspartnern. Zusätzlich dazu ist FlowForce Server in der Lage, AS2-Daten mit Hilfe vordefinierter, integrierter Funktionalitäten zu verarbeiten und lokal zu speichern. Für komplexere Aufgaben, wenn Sie AS2-Daten aus einer vorhandenen Datenquelle (z.B. einer Datenbank) aufbereiten oder in andere Formate konvertieren oder an einen Webservice senden müssen, können Sie auch MapForce und MapForce Server in den AS2-Prozess integrieren.

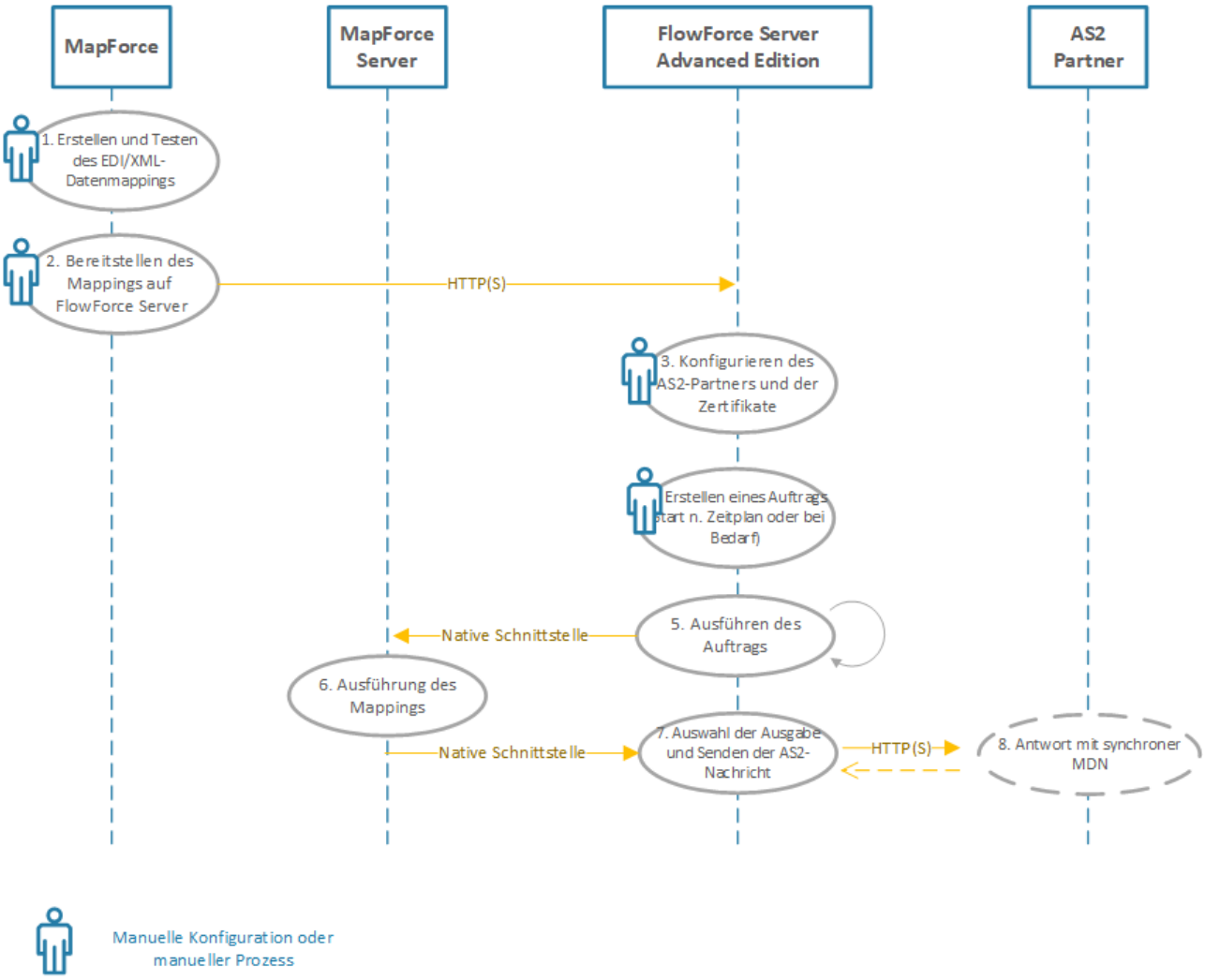
In zwei Szenarien sind MapForce und MapForce Server erforderlich:

1. Um Daten in einem beliebigen von MapForce unterstützten Format zu mappen oder zu generieren (z.B. XML, XBRL, Excel, Datenbanken, Webservices), bevor Sie diese an AS2-Partner senden.
2. Um von AS2-Partnern erhaltene Daten auf verschiedenste Weise zu transformieren (z.B. Konvertierung in Excel, Konvertierung in ein anderes XML-Schema, Speichern in einer Datenbank, Senden an einen Webservice, usw.).

Generieren und Senden von AS2-Daten

In einem Szenario, in dem Sie AS2-Daten mit MapForce vorbereiten oder generieren müssen, bevor Sie diese an Ihre Partner senden, sieht der grundlegende Ablauf folgendermaßen aus:

© 2016-2022 Altova GmbH



Generieren und Senden von AS2-Daten

Im obigen Diagramm müssen sowohl MapForce Server als auch FlowForce Server auf demselben Rechner (bei dem es sich um ein Windows-, Linux- oder macOS-Betriebssystem handeln kann) installiert sein). MapForce kann auf demselben Rechner wie MapForce Server und FlowForce Server (vorausgesetzt es handelt sich um einen Windows-Rechner) oder auf einem anderen Rechner, der sich über HTTP oder HTTPS mit FlowForce Server verbinden kann, ausgeführt werden. Der AS2-Partner ist ein Remote Server, mit dem FlowForce Server über HTTP(S) kommuniziert.

Der oben dargestellte AS2-Prozess besteht aus den folgenden Schritten:

Schritt #	Beschreibung
1. Erstellen und Testen des EDI/XML-Datenmappings	Sie können mit MapForce eine Datenmapping-Transformation erstellen, die Input-Daten in verschiedenen Formaten (darunter reiner Text, CSV, JSON, XML, verschiedene EDI-Varianten, Datenbanken, Webservices) erhält und eine oder mehrere Dateien in einen

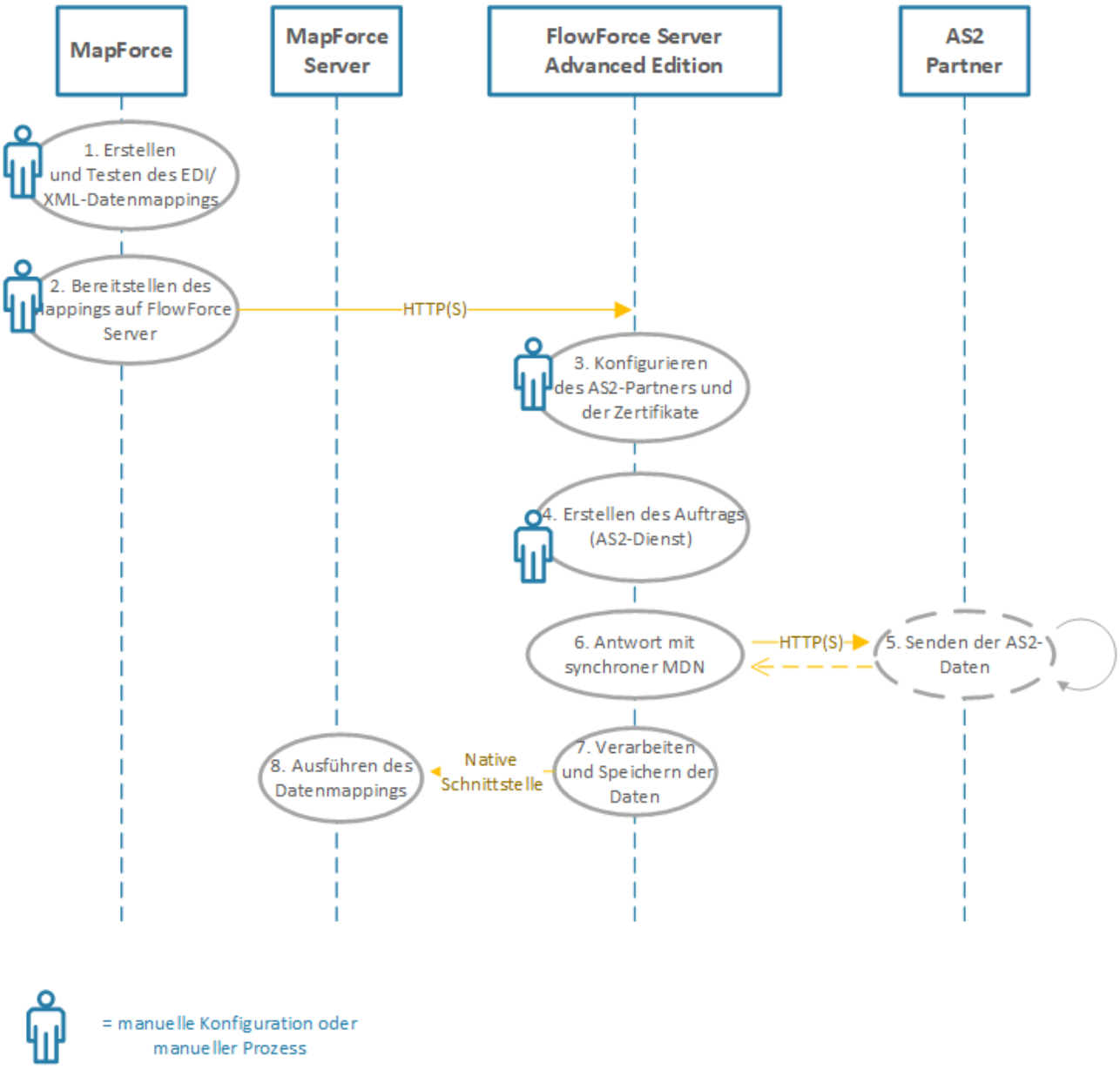
Schritt #	Beschreibung
	Zielformat (z.B. UN/EDIFACT) erzeugt. Die Erstellung eines Mappings für EDI-Zwecke unterscheidet sich nicht von anderen Mappings. Die MapForce-Dokumentation enthält verschiedene Beispiele dazu, siehe Kapitel EDI. Während Sie das Mapping erstellen, können Sie die Mapping-Ausgabe direkt in MapForce validieren und durch Klicken auf das Register "Ausgabe" eine Vorschau darauf anzeigen. Um sicherzustellen, dass das Mapping in einer Serverumgebung ausgeführt werden kann, müssen Sie es für die Transformationssprache BUILT-IN erstellen und testen, siehe auch Transformationssprachen 28.
2. Bereitstellen des Mappings auf FlowForce Server	FlowForce Server automatisiert verschiedene Aufgaben mit Hilfe von Aufträgen, die bei Bedarf oder nach einem bestimmten Zeitplan ausgeführt werden können. Diese Aufträge werden über eine Web-Oberfläche definiert werden. FlowForce Server kann auch die Ausführung eines mit MapForce erstellten Mappings automatisieren, vorausgesetzt MapForce Server wird unter FlowForce Server-Verwaltung ausgeführt. Sobald das gewünschte Resultat mit dem MapForce Mapping erzeugt wird, können Sie die Ausführung automatisieren, indem Sie das Mapping auf FlowForce Server bereitstellen. Eine Anleitung dazu finden Sie unter Bereitstellen von Mappings auf FlowForce Server 1445.
3. Konfigurieren von AS2-Partnern und Zertifikaten	Um die Kommunikation mit AS2-Partnern einzurichten, benötigen Sie deren AS2-Verbindungsinformationen (wie URI und AS2-Namen) und Exchange-Zertifikate. Die Zertifikate müssen importiert werden und die Partnerdaten müssen in FlowForce Server eingegeben werden.
4. Erstellung eines Auftrags	(a) Um das Mapping auszuführen und die gewünschte Ausgabe zu erzeugen und (b) die AS2-Nachricht zu senden (siehe auch Schritt 7), muss ein FlowForce Auftrag erstellt werden. Diese beiden Aktionen können entweder in Ausführungsschritten desselben Auftrags oder in zwei separaten Aufträgen ausgeführt werden.
5. Ausführen des Auftrags	Der im vorigen Schritt erstellte FlowForce Server-Auftrag kann je nach Geschäftsanforderung auf verschiedene Arten ausgeführt werden. So kann er z.B. als Webservice-Aufruf oder bei Änderung von Dateien im Dateisystem oder nach einem bestimmten Zeitplan (Uhrzeit und Datum) ausgeführt werden. Dieser Schritt ist vollständig automatisiert.
6. Ausführen des Datenmappings	Dieser Schritt wird ebenfalls automatisch ausgeführt und zwar von MapForce Server. Wenn in einem Auftrag die Ausführung eines Datenmappings (nach einem Zeitplan oder auf Anfrage) konfiguriert ist, wird intern MapForce Server aufgerufen. MapForce Server führt daraufhin das Mapping aus und gibt die Ausgabe an FlowForce Server zurück.
7. Auswählen der Ausgabe und senden der AS2-Nachricht	Um die AS2-Nachricht zu senden, muss die vordefinierte FlowForce Server-Funktion `/system/as2/send`. in Ihrem Auftrag (oder Auftragsschritt in einem Auftrag) aufgerufen werden. Für diese Funktion wird eine Reihe von Parametern zum Senden der AS2-

© 2016-2022 Altova GmbH

Schritt #	Beschreibung
	Nachricht benötigt, darunter das in Schritt 3 konfigurierte Partnerobjekt, die URI des Partners und der gewünschte Inhalt der AS2-Nachricht. Außerdem müssen in Ihrem Auftrag eventuell verschiedene FlowForce Server AS2-Ausdrucksfunktionen aufgerufen werden, um die Mapping-Ausgabe in die gewünschte Form zu konvertieren (z.B. von einer Datei in einen Stream).
8. Partner antwortet mit synchroner MDN	Wenn Sie das AS2-Partnerobjekt in Schritt 3 erstellen, können Sie optional auch verlangen, dass in Antwort auf die von FlowForce Server gesendete AS2-Nachricht eine Message Disposition Notification (MDN) vom Partner gesendet wird. Der Partner muss die MDN in derselben Sitzung senden, in der auch der ausgehende HTTP-Call von FlowForce Server gesendet wird (d.h. sie muss als "synchron" konfiguriert werden).

Empfangen und Verarbeiten von AS2-Daten

Wenn Ihr Unternehmen AS2-Daten von Handelspartnern erhält, können Sie zusätzlich dazu einen Workflow für den Empfang von Daten konfigurieren. In diesem Szenario kann Ihr Unternehmen AS2-Daten nicht nur empfangen und speichern, sondern auch in andere Formate transformieren, in einer Datenbank speichern oder an einen anderen Webservice senden. So könnten Sie z.B. Dateien im EDI- oder XML-Format von AS2-Handelspartnern empfangen und diese anschließend als Input für ein Mapping bereitstellen, das als wiederholter FlowForce Server-Auftrag ausgeführt wird. Ein Beispiel für einen solchen AS2-Ablauf würde folgendermaßen aussehen:



Empfangen und Verarbeiten von AS2-Daten

Der oben dargestellte AS2-Prozess besteht aus den folgenden Schritten:

Schritt #	Beschreibung
1, 2, 3	Dies sind dieselben Schritte wie in der vorhergehenden Tabelle mit dem einzigen Unterschied, dass das Mapping diesmal eine Datei als Input erhält, die Ihr Unternehmen von einem AS2-Handelspartner erhält (z.B. eine EDI- oder XML-Datei).

© 2016-2022 Altova GmbH

Schritt #	Beschreibung
4. Erstellen eines Auftrags (AS2-Dienst)	Dieser Schritt wird nur einmal durchgeführt. In diesem Schritt erstellen Sie einen FlowForce Server-Auftrag, in dem ein AS2-Dienst bereitgestellt wird. Der AS2-Dienst ist an einer vorkonfigurierten HTTP(S)-Adresse und einem vorkonfigurierten Port empfangsbereit für Requests von Ihrem AS2-Partner.
5. Senden der AS2-Daten	In diesem Schritt sendet ein Handelspartner AS2-Nachrichten an den AS2-Dienst. Damit die Kommunikation erfolgreich ist, müssen der AS2-Name des Partners und die Zertifikate bereits in FlowForce Server definiert worden sein.
6. Antwort mit synchroner MDN	FlowForce Server antwortet dem AS2-Partner mit einer synchronen MDN, die Auskunft über den Ausgang der Operation gibt (Erfolg oder Fehler).
7. Verarbeiten und Speichern der Daten	Sobald eine Nachricht eingegangen ist, konvertiert ein FlowForce Server-Auftrag die empfangenen Daten in einen String oder eine Datei und speichert diese anschließend in einem Verzeichnis oder übergibt sie als Argument an einen weiteren Auftrag. Die genaue Verarbeitungslogik kann mit Hilfe der vordefinierten FlowForce Server-Funktionen und mittels Ausdrucksfunktionen konfiguriert werden.
8. Ausführen des Datenmappings	Der FlowForce Server-Auftrag, der AS2-Daten empfängt, kann optional den im ersten Schritt erstellten Datenmappingauftrag aufrufen. Der Mapping-Auftrag erhält als Input die vom Partner erhaltenen AS2-Daten und verarbeitet diese auf eine der von MapForce unterstützten Arten: transformiert diese z.B. in ein anderes Format, speichert sie in einer Datenbank, sendet sie an einen anderen Webdienst, usw.

Die FlowForce Server-spezifischen AS2-Konfigurationsdetails entnehmen Sie bitte der Dokumentation zur FlowForce Server Advanced Edition (siehe https://www.altova.com/de/documentation).

10.7 MapForce-Befehlszeilenschnittstelle

Die allgemeine Syntax eines MapForce-Befehls in der Befehlszeile lautet:

```
MapForce.exe <dateiname> [/{ziel} [[<ausgabeverz>] [/optionen]]]
```

Legende

Die Befehlszeilensyntax wird folgendermaßen notiert:

Notation	Beschreibung
Text ohne runde oder geschweifte Klammern	Elemente, die Sie so, wie angezeigt, eingeben müssen
<Text innerhalb von spitzen Klammern>	Platzhalter, für die Sie einen Wert angeben müssen
[Text innerhalb von eckigen Klammern]	Optionale Elemente
{Text innerhalb von geschweiften Klammern}	Eine Gruppe zwingend erforderlicher Elemente; wählen Sie eines aus
Pipe-Zeichen (|)	Trennzeichen für einander gegenseitig ausschließende Elemente; wählen Sie eines aus
Auslassungszeichen (...)	Elemente, die wiederholt werden können

<dateiname>

Das Mapping-Design (.mfd) oder Mapping-Projekt (.mfp) , anhand dessen Code generiert werden soll. Um Code für das gesamte Projekt zu generieren, definieren Sie das Ziel für `/GENERATE` (siehe Beschreibung unten) und geben Sie den Projektpfad als `<dateiname>` ein, z.B. **MapForceExamples.mfp**.

/{ziel}

Definiert die Zielsprache oder Zielumgebung, für die Code generiert werden soll. Es werden die folgenden Codegenerierungsziele unterstützt.

Ziel	Beschreibung
`/COMPILE[:compileoptionen]`	Kompiliert ein Mapping zu einer MapForce Server-Ausführungdatei (.mfx). Optional können die folgenden Optionen, getrennt durch Kommas, definiert werden: `JDBC` Transformiert alle Datenbankverbindungen unter Verwendung des JDBC-Treibers und der im Dialogfeld "Datenbank-Komponenteneinstellungen" definierten

© 2016-2022 Altova GmbH

Ziel	Beschreibung
	Datenbank-URL in JDBC, siehe auch Datenbank-Komponenteneinstellungen[286]. `NOXMLSIGNATURES` Unterdrückt die Generierung digitaler Signaturen in der MapForce Server-Ausführungsdatei (Beachten Sie, dass digitale Signaturen von MapForce Server nicht unterstützt werden).
`/GENERATE`	Generiert für alle Mappings in der Projektdatei anhand der aktuellen Ordnereinstellungen Projektcode, siehe Verwalten von Projektordnern[122]. Wenn Sie dieses Ziel auswählen, muss unbedingt ein MapForce-Projekt (.mfp-Datei) als `<dateiname>` angegeben werden.
`/XSLT`	Generiert XSLT 1.0-Code.
`/XSLT2`	Generiert XSLT 2.0-Code.
`/XSLT3`	Generiert XSLT 3.0-Code.
`/XQuery`	Generiert XQuery-Code.
`/JAVA`	Generiert Java-Code.
`/CS`	Generiert C#-Code. Bei diesem Befehl können optional auch die folgenden Optionen für die Codegenerierung definiert werden: `/CS[:{VS2010|VS2013|VS2015|VS2017|VS2019}]` `VS2010` Visual Studio 2010 `VS2013` Visual Studio 2013 `VS2015` Visual Studio 2015 `VS2017` Visual Studio 2017 `VS2019` Visual Studio 2019 Wenn keine Visual Studio-Version definiert wird, wird der Code mit der in den Codegenerierungsoptionen (siehe Code Generator-Optionen[1701]) definierten Visual Studio-Version generiert.
`/CPP`	Generiert C++-Code. Bei diesem Befehl können optional auch die folgenden Optionen für die Codegenerierung definiert werden: `/CPP[:{VS2008|VS2010|VS2013|VS2015|VS2017},{MSXML|XERCES3},{LIB|DLL},{MFC|NoMFC}]`

Ziel	Beschreibung
	In der ersten Optionsgruppe wird die Visual Studio-Zielversion definiert. Gültige Werte: `VS2008` Visual Studio 2008 `VS2010` Visual Studio 2010 `VS2013` Visual Studio 2013 `VS2015` Visual Studio 2015 `VS2017` Visual Studio 2017 In der zweiten Optionsgruppe wird die vom generierten Code verwendete XML-Bibliothek definiert. Gültige Werte: `MSXML` Generiert Code für MSXML 6.0 `XERCES3` Generiert Code für Xerces 3 In der dritten Optionsgruppe wird definiert, ob statische oder dynamische Bibliotheken generiert werden sollen. Gültige Werte: `LIB` Generiert statische LIB-Bibliotheken `DLL` Generiert DLL-Bibliotheken In der vierten Optionsgruppe wird definiert, ob Code mit oder ohne MFC-Unterstützung generiert werden soll. Gültige Werte: `MFC` Aktiviert MFC-Unterstützung `NoMFC` Deaktiviert MFC-Unterstützung Wenn die Optionen oben nicht definiert sind, wird Code mit der in den Codegenerierungsoptionen (siehe Code Generator-Optionen 1701 definierten Visual Studio Version generiert.

<ausgabeverz>

Optionaler Parameter, mit dem das Ausgabeverzeichnis definiert wird. Wenn kein Ausgabepfad angegeben wird, wird das aktuelle Arbeitsverzeichnis verwendet. Beachten Sie, dass relative Dateipfade relativ zum aktuellen Arbeitsverzeichnis sind.

Wenn das Ziel `/GENERATE` ist und der Parameter `<ausgabeverz>` nicht definiert ist, werden die Codegenerierungssprache sowie der Ausgabepfad der einzelnen Mappings den für die einzelnen Ordner im Projekt definierten Einstellungen (siehe Verwalten von Projektordnern) 122 entnommen.

© 2016-2022 Altova GmbH

Wenn das Ziel `/GENERATE` ist und der Parameter `<ausgabeverz>` definiert ist, hat der in der Befehlszeile angegebene Wert `<ausgabeverz>` Vorrang vor dem auf Root-Ebene des Projekts definierten Ausgabeverzeichnis. Er hat jedoch keinen Vorrang vor den Codegenerierungseinstellungen, die für die einzelnen Ordner im Projekt definiert sind.

/optionen

Die `/optionen` schließen einander nicht gegenseitig aus. Sie können eine oder mehrere der folgenden Optionen definieren.

Option	Beschreibung
`[/MFXVERSION[:<version>]`	Diese Option kann verwendet werden, wenn das Ziel `/COMPILE` ist. Damit wird die MapForce Server-Ausführungsdatei (.mfx) für eine bestimmte Version von MapForce Server kompiliert. Sie können als Wert jede Version von MapForce Server ab Version **2013r2** bis zur aktuellen MapForce Version angeben. Siehe auch Kompilieren von Mappings zu einer spezifischen MapForce Server-Version 1443.
`/GLOBALRESOURCEFILE <dateiname>`	Diese Option kann verwendet werden, wenn im Mapping zum Auflösen von Input- oder Output-Datei- oder -Ordnerpfaden oder -Datenbanken globale Ressourcen verwendet werden. Nähere Informationen dazu finden Sie unter Globale Altova-Ressourcen 1462. Mit der Option `/GLOBALRESOURCEFILE` wird der Pfad zu einer globalen XML-Ressourcendatei definiert. Beachten Sie: Wenn `/GLOBALRESOURCEFILE` definiert ist, muss auch `/GLOBALRESOURCECONFIG` definiert sein.
`/GLOBALRESOURCECONFIG <konfig>`	Mit dieser Option wird der Name der globalen Ressourcen-Konfigurationsdatei definiert (siehe auch vorhergehende Option). Beachten Sie: Wenn `/GLOBALRESOURCEFILE` definiert ist, muss auch `/GLOBALRESOURCECONFIG` definiert sein.
`/LIBRARY <libname> (...)`	Wird zusammen mit einer Codegenerierungszielsprache verwendet, um zusätzliche Funktionsbibliotheken zu definieren. Diese Option kann mehrmals definiert werden, um damit mehrere Bibliotheken zu laden. Siehe auch Verwalten von Funktionsbibliotheken 946.
`/LOG <logdateiname>`	Generiert eine Log-Datei unter dem angegebenen Pfad. `<logdateiname>` kann ein vollständiger Pfadname sein. Er kann z.B. sowohl einen Verzeichnis- als auch einen Dateinamen enthalten. Wenn jedoch ein vollständiger Pfad angegeben wird, muss das Verzeichnis für die zu generierende Log-Datei vorhanden sein. Wenn Sie nur den Dateinamen angeben, wird die Datei in das aktuelle Verzeichnis der Windows-Befehlszeile geschrieben.

Anmerkungen

- Relative Pfade sind relativ zum Arbeitsverzeichnis, welches das aktuelle Verzeichnis der Applikation ist, die MapForce aufruft. Dies gilt für den Pfad der .mfd-Datei, der .mfp-Datei, das Ausgabeverzeichnis, den Log-Dateinamen und die globale Ressourcendatei.

- Verwenden Sie in der Befehlszeile am Ende nicht den umgekehrten Schrägstrich und das schließende Anführungszeichen (z.B., "C:\My directory\"). Diese beiden Zeichen werden vom Befehlszeilenparser als Literalzeichen, d.h. als doppeltes Anführungszeichen interpretiert. Verwenden Sie den doppelten umgekehrten Schrägstrich \\, wenn in der Befehlszeile Leerzeichen vorkommen und Sie die Anführungszeichen ("c:\Mein Verzeichnis\\") benötigen oder vermeiden Sie Leerzeichen nach Möglichkeit überhaupt.

Beispiele

1) Um MapForce zu starten, und das Mapping `<dateiname>.mfd` zu öffnen, verwenden sie:

```
MapForce.exe <dateiname>.mfd
```

2) Um XSLT 2.0-Code zu generieren und auch eine Log-Datei mit dem Namen `<logdateiname>` zu erstellen, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /XSLT2 <ausgabeverz> /LOG <logdateiname>
```

3) Um unter Verwendung der globalen Ressourcenkonfiguration `<grkonfigname>` aus der globalen Ressourcendatei `<grdateiname>` XSLT 2.0-Code zu generieren, verwenden Sie:

```
Mapforce.exe <dateiname>.mfd /XSLT2 <ausgabeverz> /GLOBALRESOURCEFILE
<grdateiname> /GLOBALRESOURCECONFIG <grkonfigname>
```

4) Um eine C#-Applikation für Visual Studio 2015 zu generieren und eine Log-Datei zu erstellen, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /CS:VS2015 <ausgabeverz> /LOG <logdateiname>
```

5) Um unter Verwendung der in **Extras | Optionen** definierten Codegenerierungseinstellungen eine C++-Applikation zu generieren und eine Log-Datei zu erstellen, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /CPP <ausgabeverz> /LOG <logdateiname>
```

6) Um eine C++-Applikation für Visual Studio 2015, MSXML, mit statischen Bibliotheken, MFC-Unterstützung und ohne Log-Datei zu generieren, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /CPP:VS2015,MSXML,LIB,MFC
```

7) Um eine C++-Applikation für Visual Studio 2015, Xerces, mit dynamischen Bibliotheken, ohne MFC-Unterstützung und einer Log-Datei zu generieren, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /CPP:VS2015,XERCES,DLL,NoMFC <ausgabeverz> /LOG
<logdateiname>
```

8) Um eine Java-Applikation zu generieren und eine Log-Datei zu erzeugen, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /JAVA <ausgabeverz> /LOG <logdateiname>
```


© 2016-2022 Altova GmbH

9) Um unter Verwendung der Sprache und des Ausgabeverzeichnisses, die/das in den Ordnereinstellungen (für die einzelnen Ordner im Projekt) definiert ist, für alle Mappings im Projekt Code zu generieren, verwenden Sie:

```
MapForce.exe <dateiname>.mfp /GENERATE /LOG <logdateiname>
```

10) Um für alle Mappings in der Projektdatei Java-Code zu generieren, verwenden Sie:

```
MapForce.exe <dateiname>.mfp /JAVA /LOG <logdateiname>
```

Beachten Sie, dass die in den Ordnereinstellungen definierte Codegenerierungssprache ignoriert wird und für alle Mappings Java verwendet wird.

11) Um in der Befehlszeile für ein zuvor kompiliertes Java-Mapping Input- und Output-Dateien anzugeben, verwenden Sie:

```
java -jar <mappingdatei>.jar /InputFileName <inputdateiname> /OutputFileName
<outputdateiname>
```

Die `/InputFileName` und `/OutputFileName` Parameter sind die Namen spezieller Input-Komponenten im MapForce Mapping, die Ihnen gestatten, Parameter in der Befehlszeilenausführung zu verwenden (siehe Bereitstellen von Parametern für das Mapping 867).

12) Um ein Mapping zu einer MapForce Server-Ausführungsdatei für die MapForce Server Version 2023 zu kompilieren und XML-Signaturen zu unterdrücken, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /COMPILE:NOXMLSIGNATURES <ausgabeverz> /MFXVERSION:2023 /LOG
<logdateiname>
```

11 Globale Altova-Ressourcen

Globale Altova-Ressourcen sind Aliasse für Datei-, Ordner und Datenbankressourcen. Jeder Alias kann mehrere Konfigurationen haben, wobei jede Konfiguration genau einer Ressource zugeordnet wird. Wenn Sie daher eine globale Ressource verwenden, können Sie zwischen ihren Konfigurationen wechseln. So könnten Sie etwa eine Datenbank-Ressource mit zwei Konfigurationen erstellen: `Entwicklung` und `Produktion`. Je nachdem, was Sie bezwecken möchten, können Sie zwischen diesen Konfiguration wechseln. In MapForce könnten Sie Daten entweder aus der `Entwicklungs-` oder der `Produktionsdatenbank` abrufen, indem Sie die gewünschte Konfiguration vor Anzeige der Vorschau auf das Mapping aus der Dropdown-Liste auswählen.

Globale Ressourcen können applikationsübergreifend in verschiedenen Altova Applikationen verwendet werden (*siehe Unterabschnitt weiter unten*).

Globale Ressourcen in anderen Altova-Produkten

Wenn Datei-, Ordner- und Datenbankverbindungsinformationen als globale Ressourcen gespeichert werden, lassen sich diese in mehreren Altova-Applikationen wiederverwenden. Wenn Sie ein und dieselbe Datei z.B. häufig in verschiedenen Altova Desktop-Applikationen öffnen müssen, können Sie diese als globale Ressource definieren. Wenn Sie den Dateipfad ändern müssen, muss er nur an einer einzigen Stelle geändert werden. Derzeit können globale Ressourcen in den folgenden Altova-Produkten definiert und verwendet werden:

- Altova Authentic
- DatabaseSpy
- MobileTogether Designer
- MapForce
- StyleVision
- XMLSpy
- FlowForce Server
- MapForce Server
- RaptorXML Server/RaptorXML+XBRL Server

In diesem Abschnitt

In diesem Abschnitt wird erläutert, wie Sie verschiedene Arten von globalen Ressourcen erstellen und konfigurieren. Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

- Einrichten globaler Ressourcen Teil 1 1463
- Einrichten globaler Ressourcen Teil 2 1465
- XML-Dateien als globale Ressourcen 1469
- Ordner als globale Ressourcen 1471
- Datenbanken als globale Ressourcen 1473
- Transformationsergebnisse als globale Ressourcen 1475
- Globale Ressourcen in Ausführungsumgebungen 1479

© 2016-2022 Altova GmbH

11.1 Einrichten globaler Ressourcen Teil 1

Die Konfiguration globaler Ressourcen besteht aus zwei Teilen: (i) dem Erstellen einer globalen Ressource im Dialogfeld **Globale Ressourcen verwalten** (*siehe unten*) und (ii) dem Definieren der Eigenschaften dieser globalen Ressource im Dialogfeld **Globale Ressource**. Der zweite Teil wird im nächsten Kapitel 1465 behandelt.

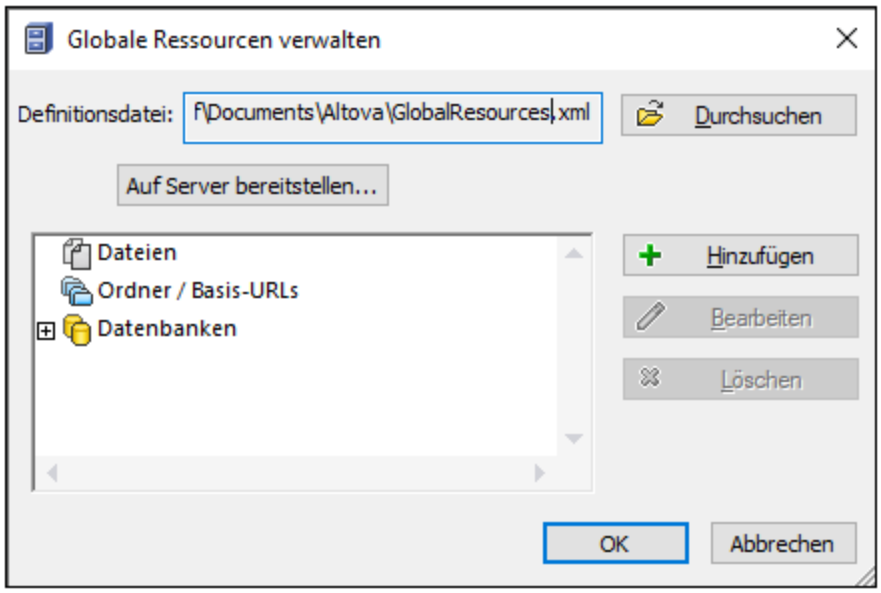
Globale Altova-Ressourcen werden im Dialogfeld **Globale Ressourcen verwalten** definiert. Dieses Dialogfeld kann auf zwei Arten aufgerufen werden:

- Wählen Sie den Menübefehl **Extras | Globale Ressourcen**.
- Klicken Sie in der Symbolleiste "Globale Ressourcen" auf das Symbol **Globale Ressourcen verwalten** (*Abbildung unten*).



Die Definitionsdatei für globale Ressourcen

Die Informationen über globale Ressourcen, werden in einer XML-Datei, der Definitionsdatei für globale Ressourcen, gespeichert. Diese Datei wird erstellt, sobald die erste globale Ressource im Dialogfeld **Globale Ressourcen verwalten** (*Abbildung unten*) definiert und gespeichert wird.



Wenn Sie das Dialogfeld **Globale Ressourcen verwalten** zum ersten Mal öffnen, wird der Standardpfad und -name der Definitionsdatei für globale Ressourcen im Textfeld *Definitionsdatei* (*siehe Abbildung oben*) definiert:

`C:\Benutzer\<Benutzername>\Dokumente\Altova\GlobalResources.xml.`

Diese Datei ist bei allen Altova-Applikationen als Standard-Definitionsdatei für globale Ressourcen definiert. Eine globale Ressource kann von einer beliebigen Altova-Applikation aus in dieser Datei gespeichert werden und steht dann allen anderen Altova-Applikationen sofort als globale Ressource zur Verfügung. Um eine globale Ressource zu definieren und in der Definitionsdatei für globale Ressourcen zu speichern, fügen Sie die globale

Ressource im Dialogfeld **Globale Ressourcen verwalten** hinzu und klicken Sie auf **OK**.

Um eine bereits vorhandene Definitionsdatei für globale Ressourcen als aktive Definitionsdatei einer bestimmten Altova-Applikation auszuwählen, navigieren Sie über die Schaltfläche **Durchsuchen** des Textfelds *Definitionen* zu dieser Datei (*siehe Abbildung oben*).

Über das Dialogfeld **Globale Ressourcen verwalten** können Sie vorhandene globale Ressourcen auch bearbeiten und löschen.

Anmerkungen:

- Sie können der Definitionsdatei für globale Ressourcen jeden beliebigen Namen geben und ihn in einem beliebigen Ordner, auf den Ihre Altova-Applikationen Zugriff haben, speichern. Sie müssen diese Datei in Ihrer Applikation nur (im Textfeld *Definitionen*) als die Definitionsdatei für globale Ressourcen für die jeweilige Applikation definieren. Die Ressourcen lassen sich in allen Altova-Produkten als globale Ressourcen verwenden, wenn Sie in allen Altova-Produkten eine einzige Definitionsdatei verwenden.

- Sie können auch mehrere Definitionsdateien für globale Ressourcen erstellen. Es kann aber immer nur eine davon in einer Altova-Applikation aktiv sein und nur die in dieser Datei enthaltenen Definitionen stehen in der Applikation zur Verfügung. Sie können dadurch je nach Bedarf festlegen, welche Ressourcen nur eingeschränkt und welche in mehreren Produkten zur Verfügung stehen sollen.

© 2016-2022 Altova GmbH

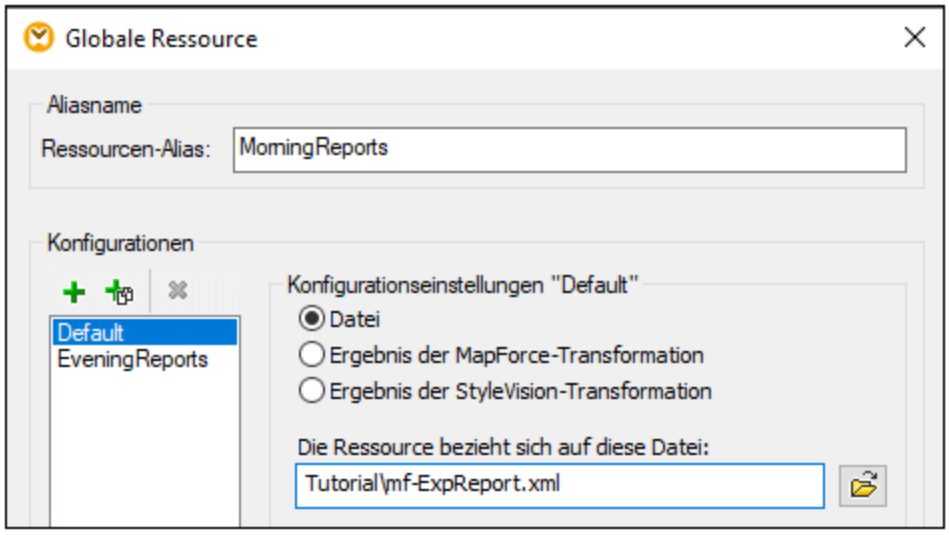
11.2 Einrichten globaler Ressourcen Teil 2

Im zweiten Teil der Konfiguration der globalen Ressourcen werden die Eigenschaften einer globalen Ressource im Dialogfeld **Globale Ressource** definiert. Die Eigenschaften sind von der Art der globalen Ressource abhängig (*siehe Unterabschnitte weiter unten*). Sie können das Dialogfeld **Globale Ressource** durch Klick auf die Schaltfläche **Hinzufügen** im Dialogfeld "Globale Ressourcen verwalten" 1463 aufrufen.

Nähere Informationen über das Einrichten verschiedener Arten von globalen Ressourcen finden Sie in den folgenden Beispielen: XML-Dateien als globale Ressourcen 1469, Ordner als globale Ressourcen 1471, Datenbanken als globale Ressourcen 1473.

Dateien

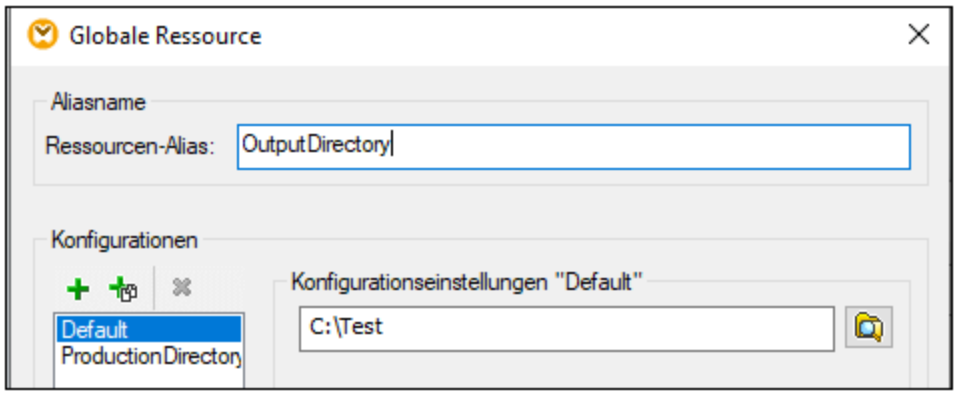
Die dateispezifischen Eigenschaften werden im Dialogfeld **Globale Ressource** unten angezeigt. Die Konfiguration besteht aus den folgenden drei Hauptteilen: (i) dem Namen der Datei, (ii) dem Pfad dieser Datei und (iii) der Liste der für diesen Dateialias definierten Konfigurationen.



Die Einstellungen *Ergebnis der MapForce-Transformation* und *Ergebnis der StyleVision Transformation* werden unter Transformationsergebnisse als globale Ressourcen 1475 behandelt.

Ordner

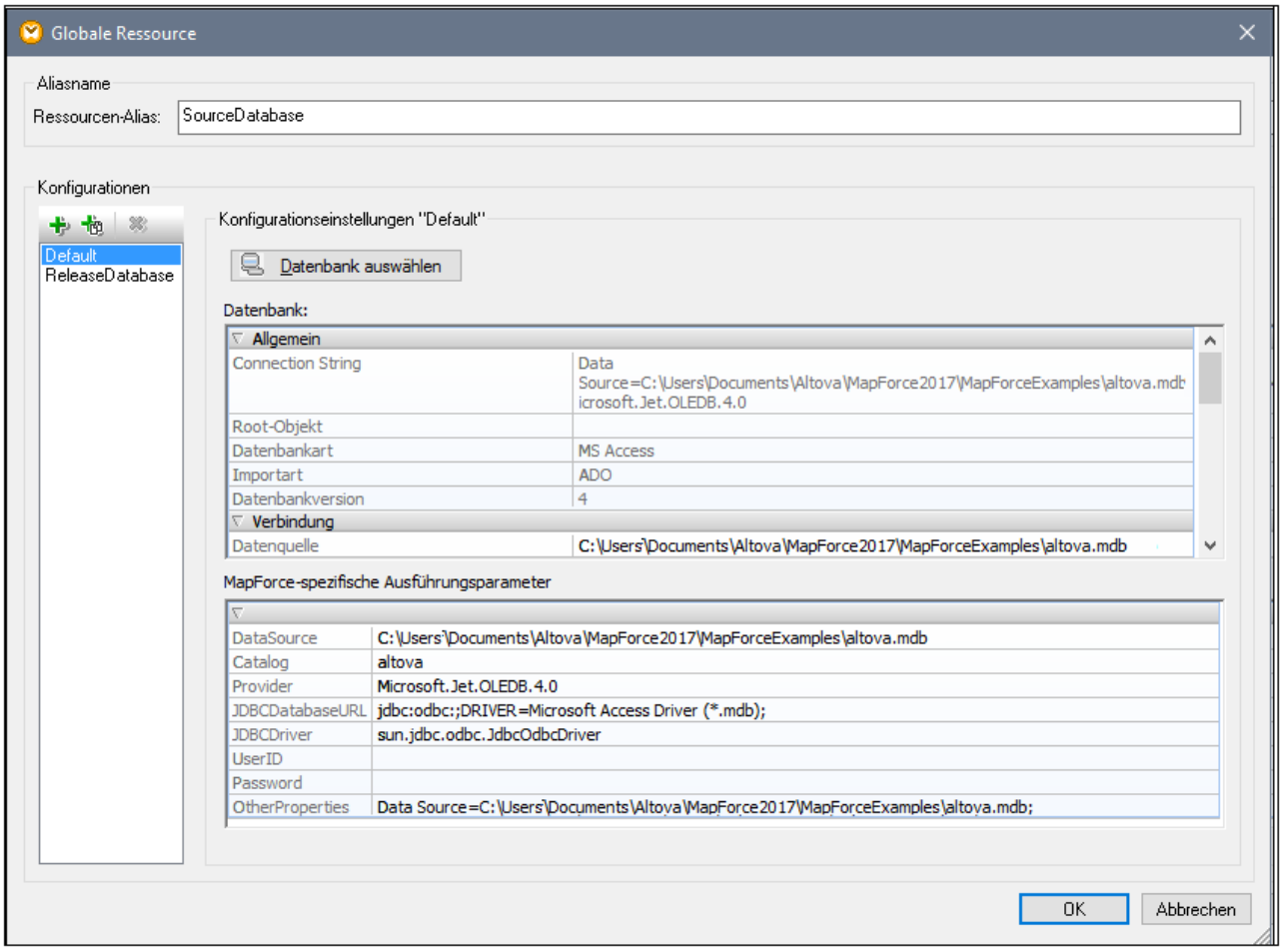
Die ordnerspezifischen Eigenschaften werden im Dialogfeld **Globale Ressource** unten angezeigt. Die Konfiguration besteht aus den folgenden drei Hauptbereichen: (i) dem Namen des Ordners, (ii) dem Pfad dieses Ordners und (iii) der Liste der für diesen Ordneralias definierten Konfigurationen.



Datenbanken

Wenn Sie eine Datenbankverbindung als globale Ressource hinzufügen, führt Sie ein Verbindungsassistent durch die einzelnen Schritte, um eine Verbindung zur Datenbank herzustellen, siehe Starten des Datenbankverbindungsassistenten 169. Nach Herstellung der Verbindung werden die Datenbankverbindungsparameter im **Dialogfeld "Globale Ressource"** (*siehe Abbildung unten*) angezeigt.

© 2016-2022 Altova GmbH



Sie können im Dialogfeld **Globale Ressource** einige der Datenbankverbindungsparameter bearbeiten. Die Parameter werden in zwei Kategorien gruppiert: Datenbankparameter und MapForce-spezifische Ausführungsparameter (*siehe unten*).

Datenbank

Diese Parameter werden von Altova-Applikationen gemeinsam verwendet. In MapForce werden sie zur Design-Zeit, d.h. wenn das Mapping geladen wird oder wenn Sie auf das Register **Ausgabe** klicken, um eine Vorschau auf das Mapping zu sehen, verwendet.

MapForce-spezifische Ausführungsparameter

Diese Parameter werden angewendet, wenn Sie Programmcode generieren oder ein Mapping zu einer MapForce Server-Ausführungsdatei (`.mfx`) kompilieren. Sie werden folgendermaßen zur Mapping-Laufzeit verwendet:

- in generiertem C++-, C#-, oder Java-Programmcode.
- Wenn Sie ein Mapping zu einer MapForce Server-Ausführungsdatei kompilieren und dabei eine automatische JDBC-Konvertierung durchgeführt wird. Nähere Informationen zur automatischen JDBC-Konvertierung finden Sie unter Datenbankmappings in verschiedenen Ausführungsumgebungen 166.

Wenn in einem Mapping eine globale Ressource verwendet wird, um eine Verbindung zu einer Datenbank herzustellen, so haben die im Dialogfeld **Globale Ressource** angezeigten Datenbankverbindungsinformationen Vorrang vor den im Mapping definierten. Im Dialogfeld "Komponenteneinstellungen" werden Sie darüber

informiert, dass die Verbindungsparameter als globale Ressource definiert werden. Um die Datenbankkomponente zu ändern, sodass direkt (und nicht über globale Ressourcen) eine Verbindung zur Datenbank hergestellt wird, klicken Sie auf **Ändern** und befolgen Sie die Anweisungen des Assistenten, um die Verbindung zur Datenbank herzustellen.

Schaltflächen im Dialogfeld "Globale Ressource"

	Konfiguration hinzufügen: Ruft das Dialogfeld "Konfiguration hinzufügen" auf, in das Sie den Namen der hinzuzufügenden Konfiguration eingeben.
	*Konfiguration als Kopie hinzufügen:* Ruft das Dialogfeld "Konfiguration hinzufügen" auf, in das Sie den Namen der Konfiguration, die als Kopie der ausgewählten Konfiguration erstellt werden soll, eingeben können.
	*Löschen:* Löscht die ausgewählte Konfiguration.
	*Öffnen:* Damit können Sie zur Datei navigieren, die als globale Ressource erstellt werden soll.
	*Öffnen:* Damit können Sie zum Ordner navigieren, der als globale Ressource erstellt werden soll.

Globale Ressourcenkonfiguration: Allgemeine Verfahren

Im Folgenden wird das Erstellen und Konfigurieren von globalen Ressourcen in groben Zügen beschrieben.

1. Klicken Sie auf die Symbolleisten-Schaltfläche  (**Globale Ressourcen verwalten**). Gehen Sie alternativ dazu zum Menü **Extras** und klicken Sie auf **Globale Ressourcen**.
2. Klicken Sie auf **Hinzufügen** und wählen Sie den gewünschten Ressourcentyp aus (Datei, Ordner, Datenbank). Daraufhin wird das Dialogfeld **Globale Ressource** angezeigt.
3. Geben Sie in das Textfeld **Ressourcen-Alias** einen beschreibenden Namen ein (z.B. `Input-Datei`).
4. Die Konfiguration der Standardkonfiguration ist von der Art der globalen Ressource abhängig: (i) Navigieren Sie im Fall einer Datei oder eines Ordners zur Datei bzw. zum Ordner, auf die die Ressource standardmäßig verweisen soll; (ii) klicken Sie im Fall einer Datenbankverbindung auf **Datenbank auswählen** und befolgen Sie die Anweisungen des Datenbankverbindungsassistenten, um eine Verbindung zur Datenbank herzustellen (siehe Herstellen einer Verbindung zu einer Datenbank. 167). Diese Datenbankverbindung wird standardmäßig verwendet, wenn Sie das Mapping ausführen.
5. Wenn Sie eine weitere Konfiguration benötigen (z.B. einen zusätzlichen Ausgabeordner), klicken Sie im Dialogfeld **Globale Ressource** auf die Schaltfläche , geben Sie den Namen dieser Konfiguration ein und definieren Sie den Pfad zu dieser Konfiguration.
6. Wiederholen Sie den vorherigen Schritt für jede weitere benötigte Konfiguration.

Anmerkung: Datenbankverbindungen als globale Ressourcen werden nur in der MapForce Professional und Enterprise Edition unterstützt.

© 2016-2022 Altova GmbH

11.3 XML-Dateien als globale Ressourcen

In diesem Kapitel wird beschrieben, wie Sie XML-Dateien als globale Ressourcen verwenden. In manchen Fällen muss eine XML-Input-Datei mehrmals pro Tag geändert werden. Zum Beispiel, wenn Sie jeden Morgen ein bestimmtes Mapping ausführen müssen, um anhand einer bestimmten XML-Datei als Mapping Input einen Bericht zu generieren, derselbe Bericht aber jeden Abend anhand einer anderen XML-Datei generiert werden muss. Anstatt das Mapping mehrmals pro Tag zu bearbeiten (oder mehrere Kopien des Mappings zu verwenden), könnten Sie das Mapping so konfigurieren, dass eine als globale Ressource definierte Datei (ein so genannter *Datei-Alias*) ausgelesen wird. Unser Datei-Alias hat in diesem Beispiel zwei Konfigurationen:

1. mit der `Default`-Konfiguration wird eine XML-Datei für die Morgenausführung als Mapping Input bereitgestellt.
2. mit der Konfiguration `EveningReports` wird eine XML-Datei für die Abendausführung als Mapping Input bereitgestellt.

Um den Datei-Alias zu erstellen und zu konfigurieren, gehen Sie vor, wie unten beschrieben.

Schritt 1: Erstellen einer globalen Ressource

Zuerst muss ein Datei-Alias erstellt werden. Gehen Sie folgendermaßen vor:

1. Klicken Sie auf die Symbolleisten-Schaltfläche  (**Globale Ressourcen verwalten**). Gehen Sie alternativ dazu zum Menü **Extras** und klicken Sie auf **Globale Ressourcen**.
2. Klicken Sie auf **Hinzufügen | Datei** und geben Sie in das Textfeld **Ressourcen-Alias** einen Namen ein. In diesem Beispiel nennen wir unsere Standardkonfiguration `MorningReports`.
3. Klicken Sie neben dem Textfeld **Die Ressource bezieht sich auf diese Datei** auf die **Durchsuchen**-Schaltfläche und wählen Sie die Datei **`Tutorial\mf-ExpReport.xml`** aus.
4. Klicken Sie im Abschnitt **Konfigurationen** auf  und geben Sie der zweiten Konfiguration den Namen `EveningReports`.
5. Klicken Sie auf **Durchsuchen** und wählen Sie die Datei **`Tutorial\mf-ExpReport2.xml`**.

Schritt 2: Verwenden der globalen Ressource im Mapping

Wir können die neu erstellte globale Ressource nun in unserem Mapping verwenden. Damit das Mapping die Daten aus der globalen Ressource liest, gehen Sie folgendermaßen vor:

1. Öffnen Sie das Mapping **`Tutorial\Tut-ExpReport.mfd`**.
2. Doppelklicken Sie auf die Titelleiste der Quellkomponente, um das Dialogfeld **Komponenteneinstellungen** zu öffnen.
3. Klicken Sie neben **XML-Input-Datei** auf **Durchsuchen**, anschließend auf **Globale Ressourcen** und wählen Sie den Datei-Alias `MorningReports` aus. Klicken Sie auf **Öffnen**.
4. Öffnen Sie das Dialogfeld **Komponenteneinsetllungen** erneut: Als XML-Input-Datei wird nun **`altova://file_resource/MorningReports`** verwendet, d.h. für den Pfad wird eine globale Ressource verwendet.

Schritt 3: Ausführen des Mappings mit der gewünschten Konfiguration

Sie können nun die XML-Input-Datei vor Ausführung des Mappings ganz einfach wechseln:

- Um **`mf-ExpReport.xml`** als Input zu verwenden, wählen Sie den Menübefehl **Extras | Aktive Konfiguration | Default**.

- Um **`mf-ExpReport2.xml`** als Input zu verwenden, wählen Sie den Menübefehl **Extras | Aktive Konfiguration | EveningReports**.

Alternativ dazu können Sie die gewünschte Konfiguration auch aus der Dropdown-Liste der **globalen Ressourcen** (*siehe Abbildung unten*) auswählen.



Um eine Vorschau auf das Mapping-Ergebnis mit einer der beiden Konfigurationen anzuzeigen, klicken Sie auf das Register **Ausgabe**.

© 2016-2022 Altova GmbH

11.4 Ordner als globale Ressourcen

In diesem Kapitel wird beschrieben, wie Sie Ordner als globale Ressourcen verwenden. Es kann vorkommen, dass dieselbe Ausgabe in verschiedenen Verzeichnissen generiert werden muss. Zu diesem Zweck erstellen wir einen Ordner-Alias mit zwei Konfigurationen:

1. Mit der Konfiguration `Default` wird die Ausgabe in **`C:\Test`** generiert.
2. Mit der Konfiguration `Production` wird die Ausgabe in **`C:\Production`** generiert.

Um den Ordner-Alias zu erstellen und zu konfigurieren, gehen Sie vor, wie unten beschrieben.

Schritt 1: Erstellen einer globalen Ressource

Zuerst muss ein Ordner-Alias erstellt werden. Gehen Sie folgendermaßen vor:

1. Klicken Sie auf die Symbolleisten-Schaltfläche  (**Globale Ressourcen verwalten**). Gehen Sie alternativ dazu zum Menü **Extras** und klicken Sie auf **Globale Ressourcen**.
2. Klicken Sie auf **Hinzufügen | Ordner** und geben Sie in das Textfeld **Ressourcen-Alias** einen Namen ein. In diesem Beispiel nennen wir unsere Standardkonfiguration `OutputDirectory`.
3. Klicken Sie neben dem Textfeld **Konfigurationseinstellungen "Default"** auf die Schaltfläche **Durchsuchen** und wählen Sie **`C:\Test`** aus. Stellen Sie sicher, dass dieser Ordner auf Ihrem Rechner vorhanden ist.
4. Klicken Sie auf  und geben Sie den Namen der zweiten Konfiguration ein. In diesem Beispiel nennen wir unsere zweite Konfiguration `ProductionDirectory`.
5. Klicken Sie auf **Durchsuchen** und wählen Sie den Ordner **`C:\Production`** aus. Stellen Sie sicher, dass dieser Ordner auf Ihrem Rechner vorhanden ist.

Schritt 2: Verwenden der globalen Ressource im Mapping

Im nächsten Schritt sorgen wir dafür, dass der soeben erstellte Ordner-Alias im Mapping verwendet wird. Gehen Sie folgendermaßen vor:

1. Öffnen Sie das Mapping **`Tutorial\Tut-ExpReport.mfd`**.
2. Doppelklicken Sie auf die Titelleiste der Zielkomponente, um das Dialogfeld **Komponenteneinstellungen** zu öffnen.
3. Klicken Sie auf **Globale Ressourcen** und anschließend auf **Speichern**.
4. Speichern Sie die XML-Ausgabedatei als **`Output.xml`**. Der Pfad der XML-Output-Datei lautet nun **`altova://folder_resource/OutputDirectory/Output.xml`**, was darauf hinweist, dass der Pfad als globale Ressource definiert wurde.

Schritt 3: Ausführen des Mappings mit der gewünschten Konfiguration

Sie können nun die Ausgabeordner vor Ausführung des Mappings ganz einfach wechseln:

- Um **`C:\Test`** als Ausgabeverzeichnis zu verwenden, wählen Sie den Menübefehl **Extras | Aktive Konfiguration | Default**.
- Um **`C:\Production`** als Ausgabeverzeichnis zu verwenden, wählen Sie den Menübefehl **Extras | Aktive Konfiguration | ProductionDirectory**.

Die Mapping-Ausgabe wird standardmäßig als temporäre Datei geschrieben, außer Sie haben in MapForce explizit konfiguriert, dass die Ausgabe in permanente Dateien geschrieben wird. Um MapForce so zu konfigurieren, dass permanente Dateien generiert werden, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Extras** auf **Optionen.**
2. Wählen Sie im Abschnitt **Allgemein** die Option **Direkt in endgültige Output-Dateien schreiben.**

© 2016-2022 Altova GmbH

11.5 Datenbanken als globale Ressourcen

In diesem Kapitel wird beschrieben, wie Sie Datenbanken als globale Ressourcen verwenden. Manchmal kann es vorkommen, dass Sie Daten aus Datenbanken mit derselben Struktur aber unterschiedlichen Daten aufeinander mappen müssen. Mit Hilfe einer Datenbankressource können Sie zwischen Datenbanken wechseln, ohne Ihr Mapping bearbeiten zu müssen. Zu diesem Zweck müssen wir einen Datenbank-Alias mit zwei Konfigurationen erstellen:

1. Die `Default`-Konfiguration verweist auf `DevelopmentDatabase`: **`MapForceExamples\altova.mdb`**.
2. Die Konfiguration `ReleaseDatabase` verweist auf **`Tutorial\altova.mdb`** verweisen.

Um den Datenbank-Alias zu erstellen und zu konfigurieren, gehen Sie vor, wie unten beschrieben.

Schritt 1: Erstellen einer globalen Ressource

Im ersten Schritt wird ein Datenbank-Alias erstellt. Gehen Sie folgendermaßen vor:

1. Klicken Sie auf die Symbolleisten-Schaltfläche  (**Globale Ressourcen verwalten**). Gehen Sie alternativ dazu zum Menü **Extras** und klicken Sie auf **Globale Ressourcen**.
2. Klicken Sie auf **Hinzufügen | Datenbank** und geben Sie in das Textfeld **Ressourcen-Alias** einen beschreibenden Namen ein. In diesem Beispiel nennen wir die Standardkonfiguration `DevelopmentDatabase`.
3. Klicken Sie auf **Datenbank auswählen**, wählen Sie **Microsoft Access (ADO)** aus und navigieren Sie zu **`MapForceExamples\altova.mdb`**.
4. Klicken Sie auf  und geben Sie der zweiten Konfiguration den Namen `ReleaseDatabase`.
5. Klicken Sie auf **Datenbank auswählen**, wählen Sie **Microsoft Access (ADO)** aus und navigieren Sie zu **`Tutorial\altova.mdb`**.

Schritt 2: Verwenden der globalen Ressource im Mapping

Im nächsten Schritt muss das Mapping nun so konfiguriert werden, dass es den Datenbank-Alias verwenden kann:

1. Öffnen Sie das Mapping **`Tutorial\PersonDB.mfd`**.
2. Doppelklicken Sie auf die Datenbankkomponente, um das Dialogfeld **Komponenteneinstellungen** zu öffnen. Klicken Sie auf **Ändern**.
3. Klicken Sie im Dialogfeld **Datenbank auswählen** auf **Globale Ressourcen** und wählen Sie den Alias `DevelopmentDatabase` aus. Klicken Sie auf **Verbinden**.
4. Wenn Sie aufgefordert werden, Datenbankobjekte auszuwählen, belassen Sie die Standardauswahl unverändert und klicken Sie auf **OK**.

Die Verbindungsparameter können über die Symbolleisten-Schaltfläche  geändert werden.

Schritt 3: Ausführen des Mappings mit der gewünschten Konfiguration

Sie können nun vor Ausführung des Mappings ganz einfach die Datenbank wechseln:

- Um die Konfiguration `DevelopmentDatabase` zu verwenden, wählen Sie den Menübefehl **Extras | Aktive Konfiguration | Default**.
- Um die Konfiguration `ReleaseDatabase` zu verwenden, wählen Sie den Menübefehl **Extras | Aktive Konfiguration | ReleaseDatabase**.

Alternativ dazu können Sie die gewünschte Konfiguration auch aus der Dropdown-Liste der **globalen Ressourcen** (*siehe Abbildung unten*) auswählen.



Wenn Sie die Konfiguration wechseln, werden Sie im **Konfigurationswechsel**-Dialogfeld darüber informiert, dass die Ressource geändert wurde. Klicken Sie auf **Neu laden**.

Anmerkung: Die in diesem Beispiel verwendeten Datenbanken enthalten dieselben Daten. Es gibt daher keine Unterschiede in der generierten Ausgabe.

© 2016-2022 Altova GmbH

11.6 Transformationsergebnisse als globale Ressource

Sie können das Ergebnis eines MapForce Mappings oder einer StyleVision-Transformation als globale Ressource verwenden. In diesem Kapitel erfahren Sie, wie Sie anhand des Transformationsergebnisses eine globale Ressource erstellen und diese in verschiedenen Altova-Applikationen verwenden.

Damit eine Mapping-Ausgabe als globale Ressource zur Verfügung steht, muss als Transformationssprache des Mappings entweder Built-in definiert sein oder das Mapping darf nur Komponenten enthalten, die von der Built-in-Sprache unterstützt werden.

Achtung:

- Die oben erwähnten Arbeitsabläufe sind nur zwischen auf demselben Rechner installierten Altova Desktop-Applikationen sinnvoll.
- Es ist **nicht** möglich, das Ergebnis von MapForce- und StyleVision-Transformationen in Form von globalen Ressourcen in Altova Server-Produkten und der MapForce Basic Edition zu verwenden.

Im Beispiel unten wird gezeigt, wie Sie das Ergebnis einer MapForce-Transformation als globale Ressource verwenden.

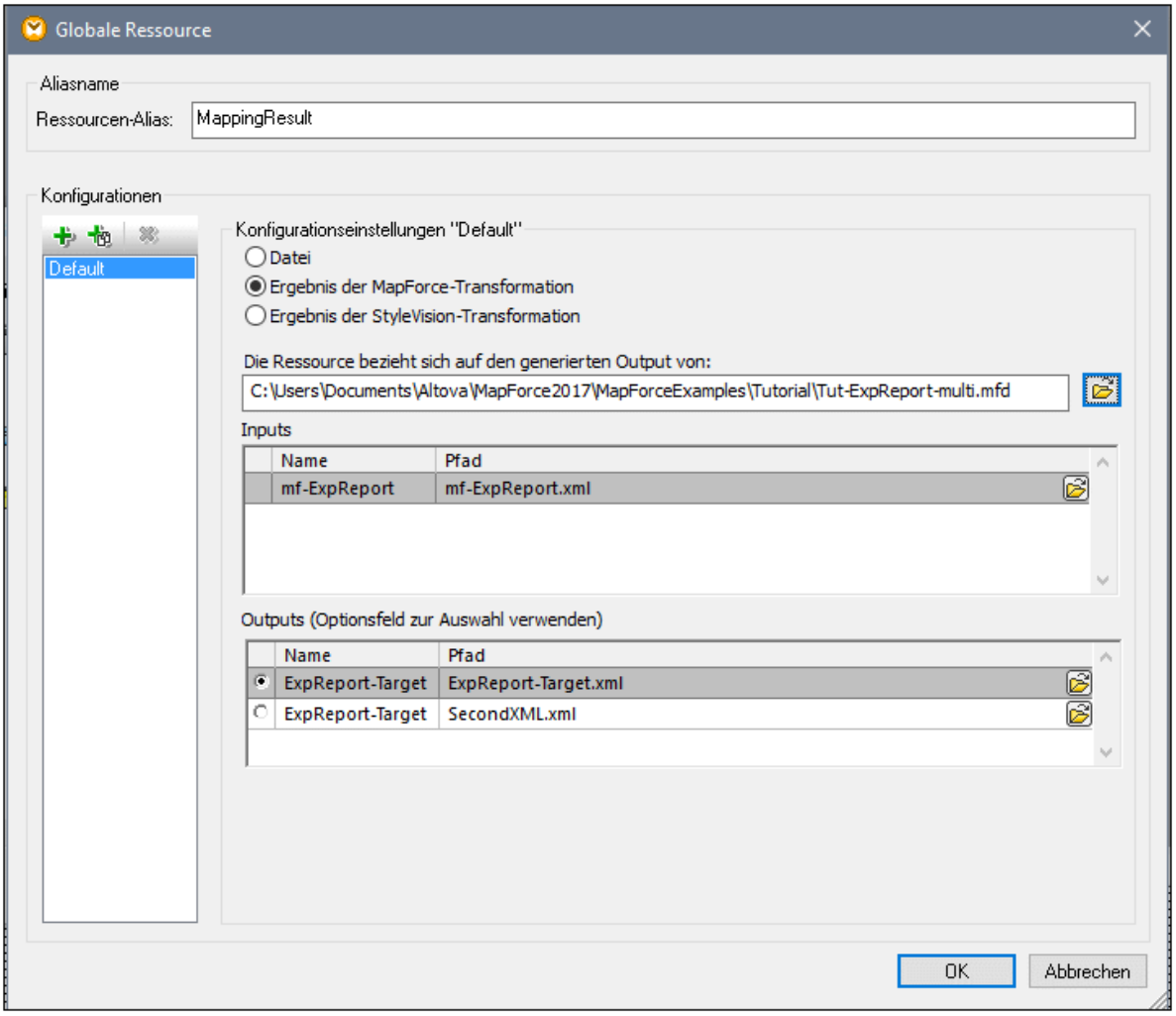
Beispiel: Ergebnis einer MapForce-Transformation

In diesem Beispiel wird gezeigt, wie Sie einen Workflow zwischen Altova MapForce und Altova XMLSpy erstellen. Dabei wird gezeigt, wie Sie eine globale Ressource anhand eines MapForce-Mappings erstellen, die Ausführung dieses Mappings direkt von XMLSpy aus starten und die von MapForce generierte Ausgabe in XMLSpy anzeigen.

Schritt 1: Erstellen einer globalen Ressource

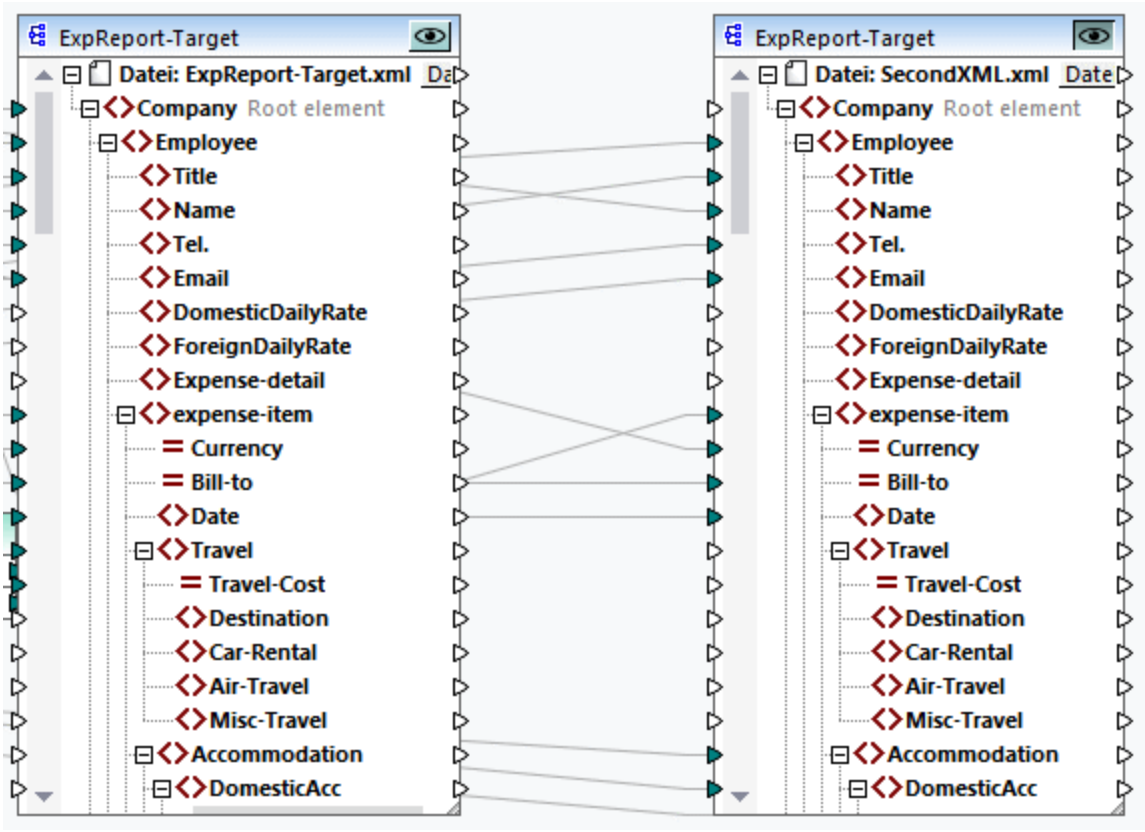
Sie können diesen Schritt in MapForce oder XMLSpy vornehmen.

1. Klicken Sie auf die Symbolleisten-Schaltfläche  (**Globale Ressourcen verwalten**). Gehen Sie alternativ dazu zum Menü **Extras** und klicken Sie auf **Globale Ressourcen**.
2. Klicken Sie auf **Hinzufügen | Datei** und geben Sie in das Textfeld **Ressourcen-Alias** einen beschreibenden Namen ein. In diesem Beispiel nennen wir unsere Standardkonfiguration `MappingResult`.
3. Aktivieren Sie die Option **Ergebnis der MapForce-Transformation**.
4. Klicken Sie auf **Durchsuchen** und wählen Sie das Mapping **`Tutorial\Tut-ExpReport-multi.mfd`** aus. Wie unten gezeigt, hat dieses Mapping einen Input und zwei Outputs.



In der Abbildung unten sehen Sie die beiden im Dialogfeld **Globale Ressource** aufgelisteten Ausgaben. Wir werden jede Ausgabedatei separat im Ordner **`C:\temp`** generieren (*siehe Schritt 2 unten*).

© 2016-2022 Altova GmbH



Schritt 2: Generieren von Ausgabedateien

Wir möchten nun jede der beiden Ausgabedateien (*siehe Abbildung oben*) im Ordner **C:\temp** generieren und die Dateinamen ändern. Zu diesem Zweck erstellen wir für jede Ausgabe eine Konfiguration, Gehen Sie folgendermaßen vor:

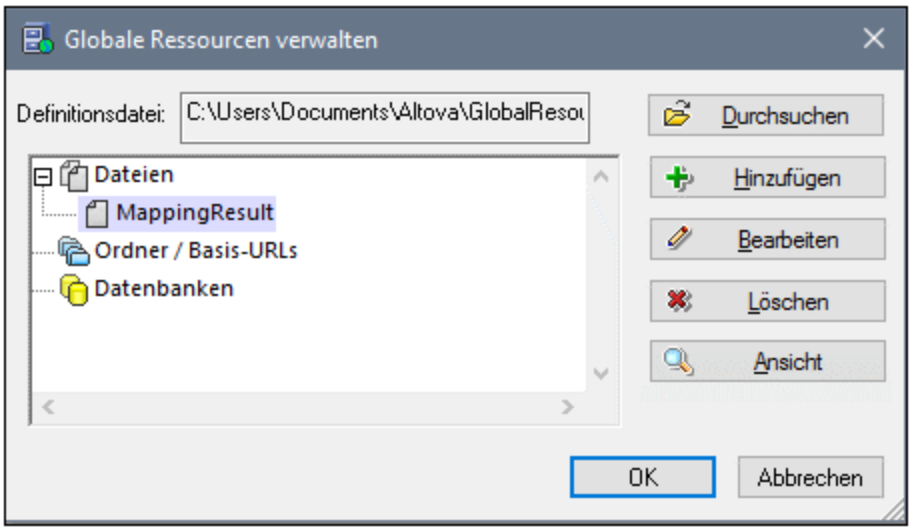
1. Klicken Sie im Bereich *Outputs* des Dialogfelds **Global Resource** neben der ersten Ausgabe auf **Durchsuchen** und geben Sie als Namen der Zieldatei **C:\temp\file1.xml** ein. Dies ist die Standardkonfiguration, mit der die erste Ausgabedatei erzeugt wird.
2. Klicken Sie unter *Konfigurationen* auf  und geben Sie einen Namen für die neue Konfiguration ein (in diesem Beispiel `Output2`). Klicken Sie im Bereich *Outputs* auf das Optionsfeld neben der zweiten Datei (**SecondXML.xml**).
3. Klicken Sie unter *Outputs* neben der zweiten Ausgabe auf **Durchsuchen** und geben Sie als Zieldateinamen **C:\temp\file2.xml** ein. Dies ist die zweite Konfiguration, mit der die zweite Ausgabedatei erzeugt wird.
4. Klicken Sie auf **OK.**

Schritt 3: Verwendung der globalen Ressource

In der Anleitung unten wird beschrieben, wie Sie die im vorherigen Schritt erstellte globale Ressource verwenden.

Standardkonfiguration
Um die Standardkonfiguration in XMLSpy zu verwenden, gehen Sie vor, wie unten beschrieben.

1. Starten Sie XMLSpy.
2. Gehen Sie zum Menü **Extras** und klicken Sie auf **Globale Ressourcen**.
3. Klicken Sie im Abschnitt *Dateien* auf die globale Ressource `MappingResult` und anschließend auf **Ansicht** (*siehe Abbildung unten*). Dadurch wird das Mapping ausgeführt, die Standardausgabe (**`file1.xml`**) wird erzeugt und im Hauptfenster von XMLSpy geladen. Die Datei wird als **`C:\temp\file1.xml`** gespeichert.



Zweite Konfiguration
Um die Mapping-Ausführung mit der zweiten Konfiguration zu starten, gehen Sie folgendermaßen vor:

1. Klicken Sie in XMLSpy im Menü **Extras** auf **Aktive Konfiguration | Output2**.
2. Klicken Sie auf **Neu laden**, wenn Sie dazu aufgefordert werden.

Daraufhin wird die zweite Ausgabedatei im Hauptfenster von XMLSpy geladen. Die Datei wird als **`C:\temp\file2.xml`** gespeichert.

© 2016-2022 Altova GmbH

11.7 Globale Ressourcen in Ausführungsumgebungen

In diesem Unterabschnitt erfahren Sie, wie Sie in verschiedenen Ausführungsumgebungen mit globalen Ressourcen arbeiten. Dieser Unterabschnitt ist in die folgenden Kapitel gegliedert:

- Globale Ressources im generierten Code 1479
- Globale Ressourcen in MapForce Server 1480
- Globale Ressourcen in FlowForce Server 1480

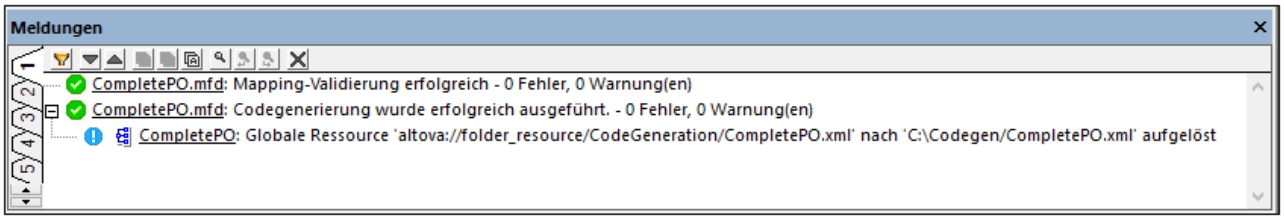
11.7.1 Globale Ressourcen im generierten Code

In diesem Kapitel wird erläutert, wie Sie globale Ressourcen im generierten Code verwenden. Nähere Informationen dazu finden Sie in den Unterabschnitten weiter unten.

Globale Ressourcen in XSLT, XSLT 2, XQuery
Wenn Sie XSLT- oder XSLT2-Code generieren und globale Ressourcen im Mapping verwendet werden, so hat dies keinerlei Auswirkung auf das generierte XSLT Stylesheet. Sie können die Input- und Output-Dateien bei Ausführung des XSLT Stylesheet in Ihrem XSLT-Prozessor mit oder ohne globale Ressourcen flexibel definieren. Dies gilt auch für generierten XQuery-Code.

Eine Ausnahme bildet die für die RaptorXML-Ausführung generierte **`DoTransform.bat`**-Datei. Im Mapping verwendete globale Ressourcen werden in **`DoTransform.bat`** zu tatsächlichen Pfaden aufgelöst. Dabei wird die derzeit in der Dropdown-Liste der globalen Ressourcen ausgewählte Konfiguration berücksichtigt. Informationen zur Bereitstellung globaler Ressourcen für RaptorXML finden Sie in der RaptorXML-Dokumentation).

Globale Ressourcen in C++, C#, Java
Wenn Sie C#-, C++- oder Java-Programmcode generieren, wird jede im Mapping verwendete globale Ressource aufgelöst. So wird z.B. ein als a globale Ressource definierter Datei- oder Ordner-Alias in einen Datei- bzw. Ordnerpfad konvertiert. Wenn in der Dropdown-Liste der globalen Ressourcen eine bestimmte globale Ressourcenkonfiguration ausgewählt ist, wird der Code für die ausgewählte Konfiguration generiert. Das Fenster **Meldungen** enthält Informationen darüber, wie genau die globale Ressource aufgelöst wurde (*siehe Abbildung unten*).



Um Code für eine bestimmte globale Ressourcenkonfiguration zu generieren, wählen Sie diese in der Dropdown-Liste für globale Ressourcen aus, bevor Sie Code generieren. Wenn Sie alternativ dazu Code über die Befehlszeile generieren, stellen Sie die Parameter **`GLOBALRESOURCEFILE`** und **`GLOBALRESOURCECONFIG`** in der Befehlszeile bereit. Nähere Informationen dazu finden Sie unter MapForce-Befehlszeilenschnittstelle 1456.

Von generiertem Code aus können Sie die globale Ressource nicht wechseln oder diese referenzieren. Sie können stattdessen den Code ändern, sodass der Pfad zur Input- oder Output-Datei geändert wird.

Anmerkung: Sie können in C# oder Java, nicht nur den Pfad, sondern auch den Datentyp des Input oder Output ändern[1600].

11.7.2 Globale Ressourcen in MapForce Server

Wenn Sie ein Mapping zu einer MapForce Server-Ausführungsdatei (**.mfx**) kompilieren, werden im Mapping verwendete globale Ressource-Referenzen beibehalten und nicht aufgelöst. Das bedeutet, Sie müssen diese Referenzen auf dem Server zur Verfügung stellen, damit das Mapping erfolgreich ausgeführt werden kann. In MapForce Server werden folgende Informationen benötigt, damit eine **.mfx**-Datei, in der globale Ressourcen verwendet werden, ausgeführt werden kann:

1. *Die Definitionsdatei für globale Ressourcen*. Die Datei hat auf dem Rechner, auf dem MapForce installiert ist, den Namen **GlobalResources.xml**. Sie befindet sich im Ordner **Dokumente\Altova**. Sie können diese Datei auf den Rechner, auf dem MapForce Server ausgeführt wird, kopieren und gegebenenfalls auch mehrere dieser Dateien erstellen. Siehe auch Einrichten globaler Ressourcen Teil 1[1463].
2. *Der Name der globalen Ressourcenkonfiguration*. Jede globale Ressource hat eine Standardkonfiguration. Sie können auch zusätzliche Konfigurationen erstellen. Nähere Informationen dazu finden Sie unter Einrichten globaler Ressourcen Teil 2[1465].

Die Definitionsdatei für globale Ressourcen und der Name der globalen Ressourcenkonfiguration werden in MapForce über die grafische Benutzeroberfläche definiert bzw. geändert. Diese werden in MapForce Server zur Mapping-Laufzeit definiert (*siehe unten*).

- Wenn Sie das Mapping über die Befehlszeilenschnittstelle ausführen, definieren Sie nach dem Befehl `run` die Optionen **`--globalresourceconfig`** and **`--globalresourcefile`**, z.B:

```
C:\Programme (x86)\Altova\MapForceServer2023\bin\MapForceServer.exe run
SomeMapping.mfx --globalresourcefile="C:
\Users\me\Documents\Altova\GlobalResources.xml" --globalresourceconfig="Default"
```

- Wenn Sie das Mapping über die MapForce Server API ausführen, rufen Sie die Methode **SetOptions** zwei Mal auf, bevor Sie die **Run**-Methode aufrufen. Mit dem ersten Aufruf wird der Dateipfad zur Definitionsdatei für globale Ressourcen als Option bereitgestellt, mit dem zweiten wird der Name der globalen Ressourcenkonfiguration bereitgestellt.

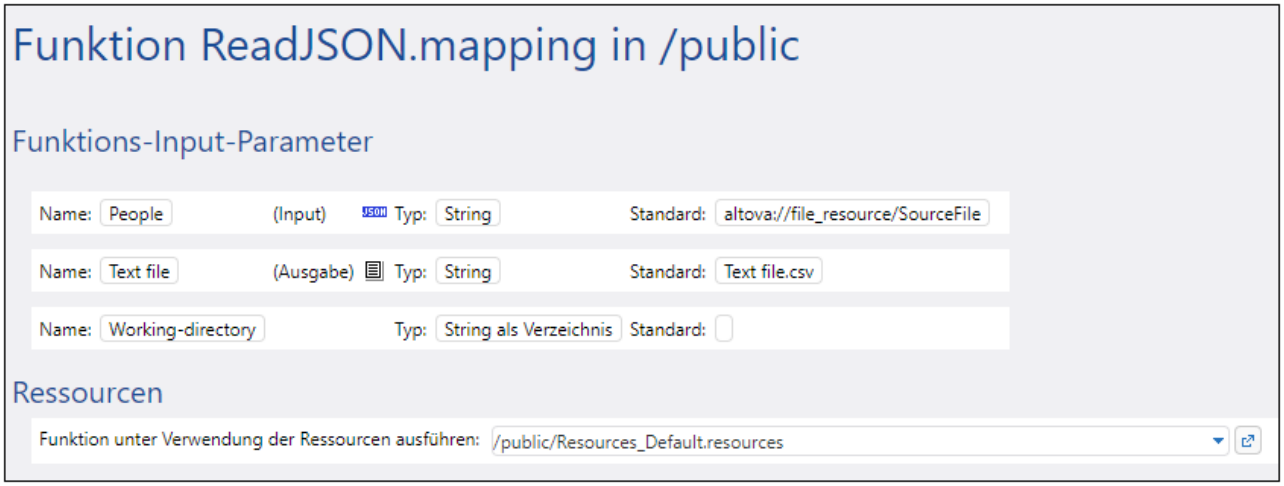
Nähere Informationen dazu finden Sie in der Dokumentation zu MapForce Server.

11.7.3 Globale Ressourcen auf FlowForce Server

Globale Ressourcen werden in FlowForce Server nicht wie in Desktop-Applikationen in einer einzigen XML-Datei gespeichert. In FlowForce ist jede Ressource ein wiederverwendbares Objekt, das Datei- oder Ordnerpfade oder Datenbankverbindungsinformationen enthalten kann. Ressourcen können kopiert, exportiert und importiert werden und unterliegen denselben Zugriffsmechanismen wie andere FlowForce-Objekte. D.h. jeder beliebige FlowForce-Benutzer kann jede beliebige Ressource in seinen Mapping-Funktionen verwenden, wenn er die entsprechenden Berechtigungen hat.

© 2016-2022 Altova GmbH

Nachdem Sie in MapForce ein Mapping mit globalen Ressourcen erstellt haben, können Sie es auf FlowForce Server bereitstellen. Sie können bei der Bereitstellung auswählen, ob im Mapping globale Ressourcen verwendet werden sollen, indem Sie im Bereitstellungsdialogfeld das Kontrollkästchen **Ressourcen verwenden** aktivieren. Wenn Sie dieses Kontrollkästchen nicht aktivieren, werden alle im Mapping verwendeten globalen Ressourcen auf Basis der aktuell ausgewählten Konfiguration aufgelöst. Wenn Sie das Kontrollkästchen aktiviert haben, werden für die Mapping-Funktion auch in FlowForce Server Ressourcen benötigt. In der Abbildung unten sehen Sie ein Beispiel für eine auf FlowForce Server bereitgestellte Mapping-Funktion, für die zur Ausführung globale Ressourcen erforderlich sind. Beachten Sie, dass der erste Parameter den Standarddateipfad aus einer Ressource abruft.



In FlowForce Server werden die globalen Ressourcen von der Mapping-Funktion und nicht vom Auftrag verwendet. Die Mapping-Funktion liest den Pfad der ersten Input-Datei aus der Ressource aus. Das bedeutet, dass alle Aufträge, in denen diese Funktion verwendet wird, denselben Pfad verwenden, es sei denn, Sie setzen den Pfad über die Auftragskonfigurationsseite außer Kraft.

Sie können globale Ressourcen auch als eigenständige Objekte auf FlowForce Server bereitstellen, d.h. es muss nicht zuerst ein Mapping bereitgestellt werden, damit eine globale Ressource bereitgestellt werden kann. Nähere Informationen dazu finden Sie unter *Bereitstellen von Ressourcen auf FlowForce Server*.

Informationen zur Verwendung von Ressourcen auf FlowForce Server finden Sie in der FlowForce Server-Dokumentation.

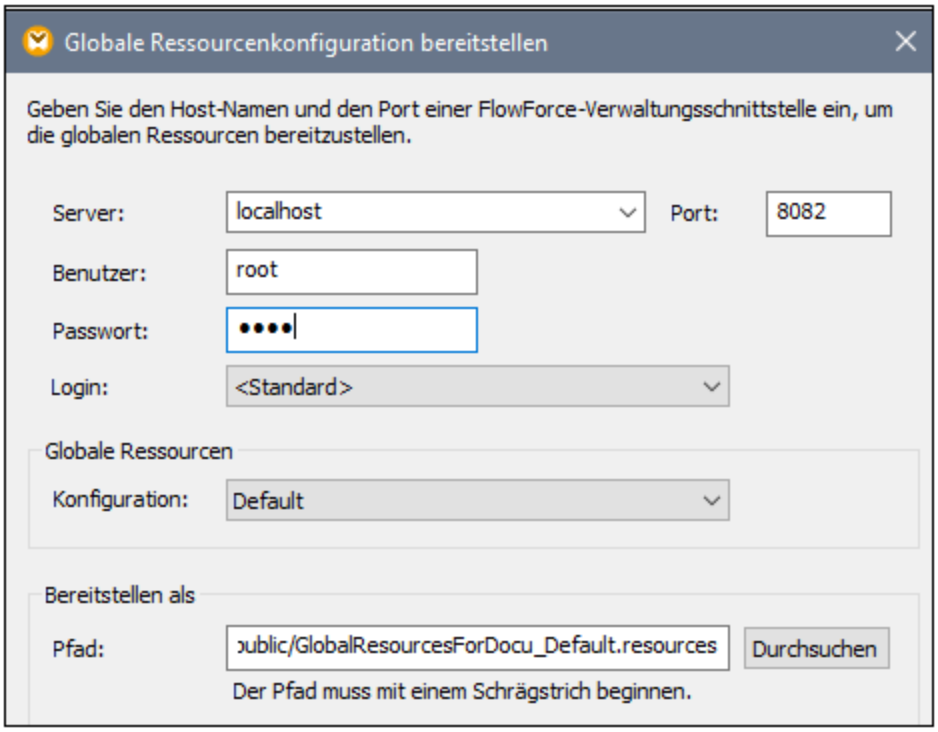
Bereitstellen von Ressourcen auf FlowForce Server

Sie können mit MapForce erstellte globale Ressourcen auf FlowForce Server bereitstellen. Bei der Bereitstellung müssen Sie die Konfigurationsdatei, mit der die Ressource auf dem Server bereitgestellt werden soll, auswählen. Wenn Sie auf dem Server alle Konfigurationen derselben globalen Ressource benötigen, können Sie diese mehrmals bereitstellen und bei jeder Bereitstellung die jeweils gewünschte Konfiguration auswählen. Sie können auch den Namen der einzelnen globalen Ressourcen auf dem Server ändern und die Zielcontainer auf dem Server auswählen.

Sie können globale Ressourcen gleichzeitig mit dem Mapping oder separat auf FlowForce Server bereitstellen. Um globale Ressourcen auf FlowForce Server bereitzustellen, gehen Sie folgendermaßen vor:

1. Starten Sie MapForce.

2. Klicken Sie auf die Symbolleisten-Schaltfläche **Globale Ressourcen verwalten**. Gehen Sie alternativ dazu zum Menü **Extras** und klicken Sie auf **Globale Ressourcen**.
3. Klicken Sie auf **Auf Server bereitstellen**. Daraufhin wird das Dialogfeld **Globale Ressourcenkonfiguration bereitstellen** (*siehe Abbildung unten*) geöffnet.



4. Geben Sie die Verbindungsinformationen in FlowForce Server ein (Server, Port, Benutzer, Passwort, Anmeldemethode). Diese Parameter sind dieselben wie bei der Bereitstellung eines Mappings auf FlowForce Server[1445].
5. Wählen Sie eine Konfiguration aus der **Konfigurationsliste** aus. Diese Liste enthält alle Konfigurationen aus der aktuellen Definitionsdatei für globale Ressourcen[1463]. Es kann immer nur eine globale Ressourcenkonfiguration auf einmal bereitgestellt werden. Wenn Sie alle Konfigurationen auf dem Server benötigen, können Sie dieselbe Ressource mehrmals unter einem anderen Namen bereitstellen.
6. Wählen Sie einen Zielpfad aus, unter dem die Ressource auf dem Server gespeichert werden soll. Klicken Sie auf **Durchsuchen**, um einen Zielcontainer auf FlowForce Server auswählen oder einen neuen erstellen können.
7. Klicken Sie auf **OK.**

Im Fenster **Meldungen** sehen Sie Informationen über die Bereitstellung der globalen Ressource auf FlowForce Server.

Anmerkung: Globale Ressourcen, die andere Altova-Applikationen ausführen, werden in einer Server-Umgebung *nicht* unterstützt. Nähere Informationen dazu finden Sie unter Transformationsergebnisse als globale Ressource[1475].

© 2016-2022 Altova GmbH

12 Kataloge in MapForce

MapForce unterstützt einen Teil der OASIS XML-Kataloge. Mit Hilfe dieser Methode kann MapForce häufig verwendete Schemas (und andere Dateien) aus lokalen Benutzerordnern abrufen. Dies erhöht die Verarbeitungsgeschwindigkeit, Benutzer können dadurch auch offline arbeiten (d.h. ohne mit einem Netzwerk verbunden zu sein) und Dokumente werden leichter übertragbar (da die URIs dann nur in den Katalogdateien geändert werden müssten).

Der Katalogmechanismus in MapForce funktioniert, wie in diesem Abschnitt beschrieben:

- Funktionsweise von Katalogen 1484
- Katalogstruktur in MapForce 1486
- Anpassen von Katalogen 1488
- Umgebungsvariablen 1490

Nähere Informationen zu Katalogen finden Sie in der XML-Katalogspezifikation.

12.1 Funktionsweise von Katalogen

Mit Hilfe von Katalogen können Umleitungen sowohl zu DTDs als auch XML-Schemas definiert werden. Das Prinzip des Mechanismus ist in beiden Fällen dasselbe, doch unterscheidet er sich in einigen weiter unten beschriebenen Details.

DTDs

Kataloge dienen normalerweise dazu, einen Aufruf von einer DTD an eine lokale URI umzuleiten. Dies geschieht in der Katalogdatei durch Mappen von Public und System Identifiern auf die gewünschte lokale URI. Wenn also die **DOCTYPE**-Deklaration in einer XML-Datei gelesen wird, findet ihr Public oder System Identifier über das Katalogdatei-Mapping die gewünschte lokale Ressource.

Für gebräuchliche Schemas ist der **PUBLIC** Identifier normalerweise vordefiniert, sodass für die URI in der Katalogdatei nur der **PUBLIC** Identifier auf den korrekten lokalen Identifier gemappt werden muss. Wenn das XML-Dokument geparst wird, wird der **PUBLIC** Identifier darin gelesen. Wenn dieser Identifier in einer Katalogdatei gefunden wird, wird die entsprechende URL in der Katalogdatei nachgeschlagen und das Schema wird von dort aus gelesen. Wenn also die folgende SVG-Datei in MapForce geöffnet wird:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="20" height="20" xml:space="preserve">
  <g style="fill:red; stroke:#000000">
     <rect x="0" y="0" width="15" height="15"/>
     <rect x="5" y="5" width="15" height="15"/>
  </g>
</svg>
```

Der Katalog wird nach dem **PUBLIC** Identifier dieser SVG-Datei durchsucht. Angenommen, die Katalogdatei enthält den folgenden Eintrag:

```
<catalog>
  ...
    <public publicId="-//W3C//DTD SVG 1.1//EN" uri="schemas/svg/svg11.dtd"/>
  ...
</catalog>
```

In diesem Fall gibt es einen Treffer für den **PUBLIC** Identifier, sodass der Lookup-Mechanismus für die SVG DTD an die URL **schemas/svg/svg11.dtd** umgeleitet wird; (dieser Pfad ist relativ zur Katalogdatei). Diese lokale Datei wird dann als DTD für die SVG-Datei verwendet. Wenn im Katalog kein passender Treffer für den **Public** Identifier gefunden wird, wird die URL im XML-Dokument verwendet (im SVG Beispiel oben ist dies die Internet URL **http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd**).

XML-Schemas

In MapForce können Sie eine **Umleitung zu einem XML-Schema** auch mit Hilfe von Katalogen durchführen. In der XML-Instanzdatei erfolgt die Referenz zum Schema im **xsi:schemaLocation** Attribut des Elements der obersten Ebene des XML-Dokuments. Beispiel:

© 2016-2022 Altova GmbH

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

Der Wert des **xsi:schemaLocation**-Attributs besteht aus zwei Teilen: einem Namespace-Teil (oben grün) und einem URI-Teil (markiert). Anhand des Namespace-Teils erfolgt im Katalog das Mapping auf die alternative Ressource. So erfolgt etwa im folgenden Katalogeintrag eine Umleitung der Schemareferenz oben auf ein Schema unter einem anderen Pfad.

```
<uri name="http://www.xmlspy.com/schemas/orgchart" uri="C:\MySchemas\OrgChart.xsd"/>
```

Normalerweise ist der URI-Teil des Attributwerts von **xsi:schemaLocation** ein Pfad zum aktuellen Schema. Wenn das Schema jedoch über einen Katalog referenziert wird, muss der URI-Teil nicht auf ein tatsächliches XML-Schema verweisen, muss aber vorhanden sein, damit das Attribut **xsi:schemaLocation** lexikalisch gültig ist. So würde z.B. der Wert `foo` als URI-Teil des Attributwerts genügen, um gültig zu sein.

12.2 Katalogstruktur in MapForce

MapForce lädt beim Start eine Datei namens **RootCatalog.xml**(*Struktur siehe unten*), die eine Liste von Katalogdateien enthält, die durchsucht werden. Sie können diese Datei bearbeiten und beliebig viele Katalogdateien definieren, die durchsucht werden sollen. Jede davon wird in einem **nextCatalog** referenziert. Diese Katalogdateien werden durchsucht und die URIs darin werden entsprechend ihren Mappings aufgelöst.

Codefragment von RootCatalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
   xmlns:spy="http://www.altova.com/catalog_ext"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd">
  <nextCatalog catalog="%PersonalFolder%/Altova/%AppAndVersionName%/CustomCatalog.xml"/>
  <!-- Include all catalogs under common schemas folder on the first directory level -->
  <nextCatalog spy:recurseFrom="%CommonSchemasFolder%" catalog="catalog.xml"
spy:depth="1"/>
  <nextCatalog spy:recurseFrom="%ApplicationWritableDataFolder%/pkgs/.cache"
catalog="remapping.xml" spy:depth="0"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
</catalog>
```

Im obigen Codefragment wird ein benutzerdefinierter Katalog namens **CustomCatalog.xml** und eine Gruppe von Katalogen, die gebräuchliche Schemas (wie z.B. W3C XML-Schemas udn das SVG-Schema) referenzieren, referenziert.

- **CustomCatalog.xml** befindet sich in Ihrem persönlichen Ordner (der über die Variable **%PersonalFolder%**) gefunden wird. erstellen. Diese Datei ist eine Skelettdatei, in der Sie Ihre eigenen Mappings erstellen können. Sie können für jedes gewünschte Schema, das nicht in den Katalogdateien im Altova Ordner "Common Schemas" enthalten ist, Mappings zu **CustomCatalog.xml** hinzufügen. Verwenden Sie dazu die unterstützten Elemente des OASIS-Katalogmechanismus (*siehe nächster Abschnitt*).
- Der durch die Variable **%CommonSchemasFolder%** definierte Ordner "Common Schemas" enthält eine Reihe gebräuchlicher Schemas. Innerhalb dieser einzelnen Schema-Ordner befindet sich eine **catalog.xml**-Datei, die Public und/oder System Identifier auf URIs mappt, die auf lokal gespeicherte Kopien des jeweiligen Schemas verweisen.
- **CoreCatalog.xml** befindet sich im MapForce-Applikationsordner und dient zum Auffinden von Schemas und Stylesheets, die von MapForce-spezifischen Prozessen wie z.B. StyleVision Power Stylesheets, anhand derer die Altova-Authentic-Ansicht von XML-Dokumenten generiert wird.

Pfadvariablen

Die in **RootCatalog.xml** verwendeten Variablen (*Codefragment oben*) haben die folgenden Werte:

%PersonalFolder%	Der persönliche Ordner des aktuellen Benutzers, z.B. C:\Benutzer\<Name>\Dokumente
%CommonSchemasFolder%	C:\ProgramData\Altova\Common2023\Schemas
%ApplicationWritableDataFolder%	C:\ProgramData\Altova

© 2016-2022 Altova GmbH

Speicherpfad von Katalogdateien und Schemas
Beachten Sie die Pfade der verschiedenen Katalogdateien.

- Die Dateien **RootCatalog.xml** und **CoreCatalog.xml** befinden sich im MapForce Applikationsordner.
- **CustomCatalog.xml** befindet sich in Ihrem Ordner **EigeneDokumente\Altova\MapForce**.
- Die **catalog.xml**-Dateien befinden sich jeweils in einem eigenen Schemaordner, wobei sich diese Schemaordner innerhalb des Ordners "Common Schemas" befinden:

12.3 Anpassen Ihrer Kataloge

Wenn Sie Einträge für **CustomCatalog.xml** (oder jede andere Katalogdatei, die von MapForce gelesen werden soll) erstellen, verwenden Sie nur die folgenden Elemente der OASIS-Katalogspezifikation. Jedes der unten angeführten Elemente wird mit einer Erläuterung der Attributwerte aufgelistet. Eine ausführlichere Beschreibung finden Sie in der XML Catalogs Specification. Beachten Sie, dass jedes Element das Attribut **xml:base**, mit dem die Basis-URI dieses Elements definiert wird, erhalten kann.

- <public publicId="PublicID of Resource" uri="URL of local file"/>
- <**system** **systemId**="SystemID of Resource" **uri**="URL of local file"/>
- <**uri** **name**="filename" **uri**="URL of file identified by filename"/>
- <**rewriteURI** **uriStartString**="StartString of URI to rewrite" **rewritePrefix**="String to replace StartString"/>
- <**rewriteSystem** **systemIdStartString**="StartString of SystemID" **rewritePrefix**="Replacement string to locate resource locally"/>

Beachten Sie die folgenden Punkte:

- In Fällen, in denen es keinen Public Identifier gibt, kann der System Identifier einer URL wie z.B. bei den meisten Stylesheets über das **system** Element direkt auf eine URL gemappt werden.
- Eine URI kann über das `uri` Element auf eine andere URI gemappt werden.
- Mit Hilfe der Elemente `rewriteURI` und `rewriteSystem` kann der Anfangsteil einer URI bzw. eines System Identifiers neu geschrieben werden. Dadurch kann der Anfang eines Dateipfads ersetzt werden, sodass ein anderes Verzeichnis als Ziel gewählt werden kann. Nähere Informationen zu diesen Elementen finden Sie in der XML Catalogs Specification.

Ab Release 2014 entspricht MapForce weitgehend der XML Catalogs Specification (OASIS Standard V1.1, 7 Oktober 2005). In dieser Spezifikation wird streng zwischen externen Identifier Look-ups (jenen mit einer öffentlichen ID oder einer System-ID) und URI Look-ups (URIs, die keine öffentlichen IDS oder System-IDs sind) getrennt. Namespace URIs müssen daher einfach als URIs - und nicht Public IDs oder System-IDs - behandelt werden und folglich als URI Look-ups anstelle von externen Identifier Look-ups verwendet werden. In MapForce Versionen vor Version 2014 wurden Schema Namespace URIs über **<public>** Mappings übersetzt. Ab Version 2014 müssen **<uri>** Mappings verwendet werden.

Vor v2014: <public **publicID**="http://www.MyMapping.com/ref"
uri="file:///C:/MyDocs/Catalog/test.xsd"/>
Ab V-2014: <**uri** **name**="http://www.MyMapping.com/ref"
uri="file:///C:/MyDocs/Catalog/test.xsd"/>

Wie findet MapForce ein referenziertes Schema

Ein Schema wird in einem XML-Dokument über das Attribut **xsi:scemaLocation** (*siehe unten*) referenziert. Der Wert des **xsi:schemaLocation**-Attributs besteht aus zwei Teilen: einem Namespace-Teil (grün) und einem URI-Teil (markiert).

xsi:schemaLocation="**http://www.xmlspy.com/schemas/orgchart** **OrgChart.xsd**"

Im Folgenden wird Schritt für Schritt beschrieben, wie MapForce vorgeht, um ein referenziertes Schema zu finden: Das Schema wird beim ersten erfolgreichen Schritt geladen.

1. Durchsuchen des Katalogs nach dem URI-Teil des **xsi:schemaLocation**-Werts. Wenn ein Mapping

© 2016-2022 Altova GmbH

gefunden wird - darunter auch in **rewriteURI** Mappings wird die erzeugte URI verwendet, um das Schema zu laden.

2. Durchsuchen des Katalogs nach dem Namespace-Teil des **xsi:schemaLocation**-Werts. Wenn ein Mapping gefunden wird - darunter auch in **rewriteURI** Mappings wird die erzeugte URI verwendet, um das Schema zu laden.
3. Der URI-Teil des **xsi:schemaLocation**-Werts wird zum Laden des Schemas verwendet.

XML-Schema-Spezifikationen

Die XML-Schemaspezifikationsinformationen sind in MapForce integriert und die Gültigkeit von XML-Schema-(`.xsd`)-Dokumenten wird anhand dieser internen Informationen überprüft. Daher sollte in einem XML-Schema-Dokument kein Verweis auf ein Schema, das die XML-Schema-Spezifikation definiert, vorgenommen werden.

Die Datei **catalog.xml** im Ordner `%AltovaCommonSchemasFolder%\Schemas\schema` enthält Referenzen auf DTDs, die ältere XML-Schema-Spezifikationen implementieren. Sie sollten Ihre XML-Schema-Dokumente nicht anhand dieser Schemas validieren. Zweck dieser beiden DTDs ist es einzig und allein, für die Eingabehilfen von MapForce zu Bearbeitungszwecken Informationen bereitzustellen, falls Sie Dateien gemäß diesen älteren Empfehlungen erstellen wollen.

12.4 Umgebungsvariablen

Shell-Umgebungsvariablen können im `nextCatalog` Element verwendet werden, um den Pfad zu Systemordnern zu definieren (*siehe RootCatalog.xml-Liste oben*). Es werden die folgenden Shell-Umgebungsvariablen unterstützt:

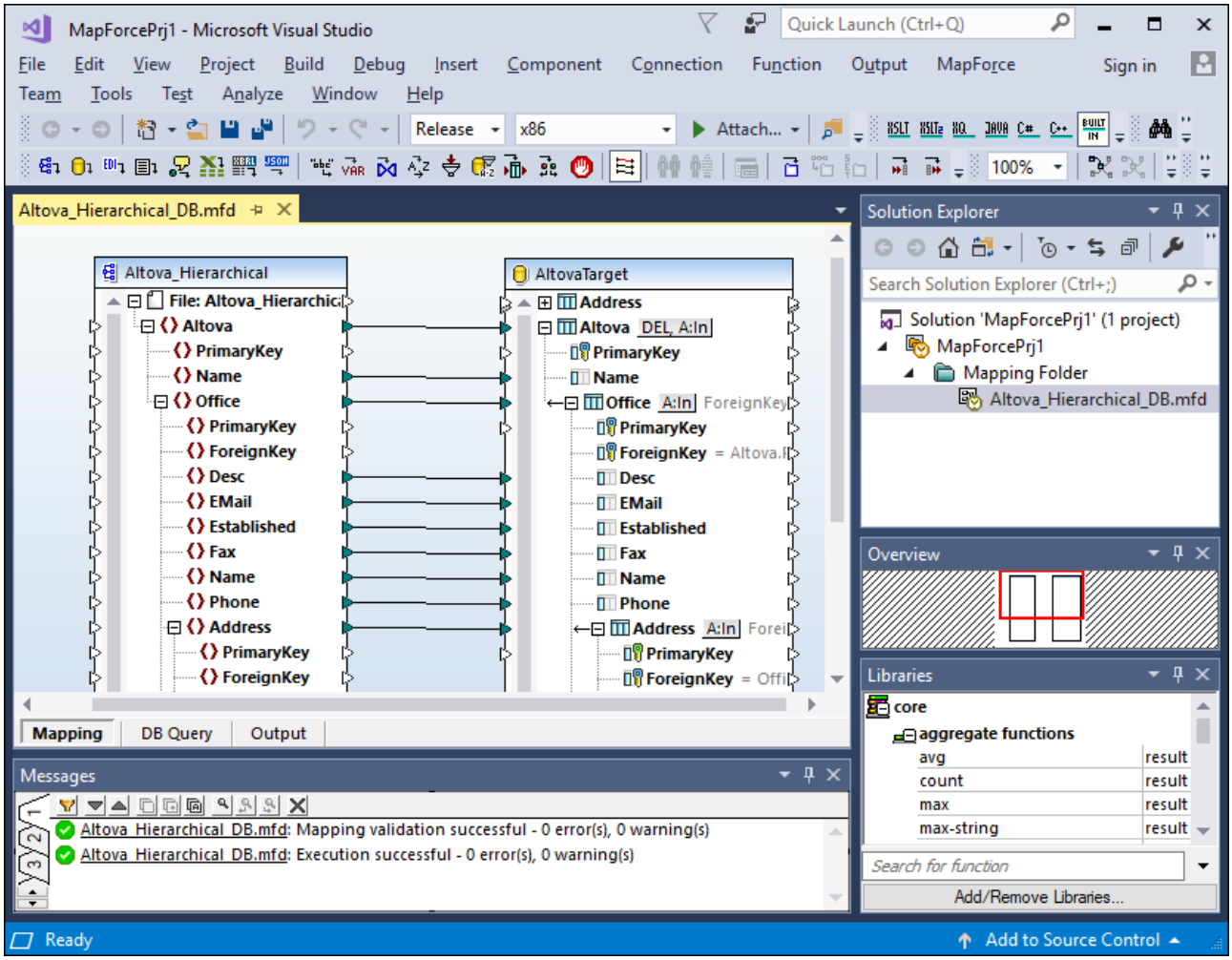
`%PersonalFolder%`	Vollständiger Pfad zum persönlichen Ordner des aktuellen Benutzers, z.B. `C:\Benutzer\<Name>\Dokumente`
`%CommonSchemasFolder%`	`C:\ProgramData\Altova\Common2023\Schemas`
`%ApplicationWritableDataFolder%`	`C:\ProgramData\Altova`
`%AltovaCommonFolder%`	C:\Programme\Altova\Common2023
`%DesktopFolder%`	Vollständiger Pfad des Desktop-Ordners des aktuellen Benutzers.
`%ProgramMenuFolder%`	Vollständiger Pfad zum Ordner 'Programme" des aktuellen Benutzers.
`%StartMenuFolder%`	Vollständiger Pfad zum Startmenüordner des aktuellen Benutzers.
`%StartUpFolder%`	Vollständiger Pfad zum Startordner des aktuellen Benutzers.
`%TemplateFolder%`	Vollständiger Pfad des Template-Ordners des aktuellen Benutzers.
`%AdminToolsFolder%`	Vollständiger Pfad zum Dateisystemverzeichnis, in dem Verwaltungstools des aktuellen Benutzers gespeichert sind.
`%AppDataFolder%`	Vollständiger Pfad zum Ordner "Anwendungsdaten" des aktuellen Benutzers.
`%CommonAppDataFolder%`	Vollständiger Pfad zum Dateisystem, das die Applikationsdaten aller Benutzer enthält.
`%FavoritesFolder%`	Vollständiger Pfad zum Ordner 'Favoriten' des aktuellen Benutzers.
`%PersonalFolder%`	Vollständiger Pfad zum Ordner "Personal" des aktuellen Benutzers.
`%SendToFolder%`	Vollständiger Pfad zum SendTo-Ordner des aktuellen Benutzers.
`%FontsFolder%`	Vollständiger Pfad zum Ordner "Systemschriftarten".
`%ProgramFilesFolder%`	Vollständiger Pfad zum Ordner 'Programme" des aktuellen Benutzers.
`%CommonFilesFolde`	Vollständiger Pfad zum Ordner "Gemeinsame Dateien" des aktuellen Benutzers.

© 2016-2022 Altova GmbH

r%	
%WindowsFolder%	Vollständiger Pfad zum Ordner "Windows" des aktuellen Benutzers.
%SystemFolder%	Vollständiger Pfad zum Ordner "System" des aktuellen Benutzers.
%LocalAppDataFolder%	Vollständiger Pfad zum Dateisystemverzeichnis, das als Datenspeicher für lokale (nicht-Roaming) Applikationen dient.
%MyPicturesFolder%	Vollständiger Pfad zum Ordner "Meine Bilder".

13 MapForce Plug-in für Visual Studio

Ihre Version von MapForce 2023 lässt sich in die Microsoft Visual Studio-Versionen 2012/2013/2015/2017/2019/2022 integrieren. Dadurch können Sie die Mapping-Funktionen von MapForce mit der Entwicklungsumgebung Visual Studio kombinieren. Wenn das MapForce Plug-in aktiviert ist, können Sie Mappings und Mapping-Projekte direkt in Visual Studio erstellen (*siehe Abbildung unten*).



Installation

Um das MapForce Plugin für Visual Studio zu installieren, gehen Sie folgendermaßen vor:

1. Installieren Sie Microsoft Visual Studio 2012/2013/2015/2017/2019/2022. Beachten Sie, dass Visual Studio ab Visual Studio 2022 nur als 64-Bit-Applikation zur Verfügung steht.
2. Installieren Sie MapForce (Enterprise oder Professional Edition). Wenn Sie Visual Studio 2022+ installiert haben, müssen Sie die 64-Bit-Version von MapForce installieren.
3. Downloaden Sie das MapForce Integrationspaket für Microsoft Visual Studio. Dieses Paket steht auf der MapForce (Enterprise und Professional Edition) Download-Seite unter www.altova.com zur Verfügung.

© 2016-2022 Altova GmbH

Nach Installation des Integrationspakets können Sie MapForce in der Visual Studio-Umgebung verwenden.

Achtung

Sie müssen das richtige Integrationspaket für Ihre jeweilige MapForce Version (die aktuelle Version ist 2023) verwenden. Das Integrationspaket ist nicht editionsspezifisch und daher sowohl für die Enterprise als auch die Professional Edition verwendet werden.

Informationen zu Menüs und Funktionen

Wenn das MapForce Plug-in für Visual Studio aktiviert ist, können Sie verschiedene MapForce-Menüs und Funktionen, wie unten gezeigt, aufrufen. Sie können die MapForce Menüs und Symbolleisten über das Visual Studio-Menü **Extras | Anpassen** anpassen.

Anmerkung: In Versionen ab Visual Studio 2019 können MapForce-Funktionalitäten über das Menü **Erweiterungen** von Visual Studio aufgerufen werden. In früheren Versionen von Visual Studio, stehen die MapForce-Funktionen in Visual Studio-Menüs der obersten Ebene zur Verfügung.

- Designs

  Im Menü **MapForce** von Visual Studio können Sie MapForce-Designs auswählen. Zur Wahl stehen: Klassisches, helles und dunkles Design.

- Erstellen und Öffnen von Dateien/Projekten

  Wenn das MapForce Plug-in für Visual Studio aktiviert ist, können Sie Mappings und Mapping-Projekte direkt in Visual Studio erstellen, öffnen und bearbeiten. Um ein neues Mapping-Design in Visual Studio zu erstellen, wählen Sie den Menübefehl **Datei | Neu**. Um ein neues Projekt zu erstellen, wählen Sie den Menübefehl **Datei | Neues Projekt**. Bestehende Mapping-Dateien oder -Projekte können Sie über die folgenden Visual Studio-Menübefehle öffnen. **Datei | Öffnen | Datei** oder **Datei | Öffnen | Projekt/Projektmappe**. Anschließend können Sie nach den zu MapForce gehörigen Dateitypen suchen.

- Globale Ressourcen

  Die globalen MapForce Ressourcen[1462] stehen über das Visual Studio Menü **MapForce | Globale Ressourcen** zur Verfügung. Ab Visual Studio 2019 ist das entsprechende Menü dazu **Erweiterungen | MapForce | Globale Ressourcen verwalten**.

- Debuggen

  Nachdem Sie eine Mapping-Datei geöffnet haben, stehen die Befehle zum Debuggen von Mappings im Menü Debuggen[1532] und in der Symbolleiste **Debuggen** zur Verfügung. In Versionen ab Visual Studio 2019 ist das entsprechende Menü dazu **Erweiterungen | MapForce | Debuggen**.

- MapForce-Optionen

  Die MapForce-Options finden Sie im Visual Studio-Menü **Extras | MapForce-Optionen**.

- Anpassen des Mapping-Fensters

  Wenn im Hauptfenster von Visual Studio ein Mapping geöffnet wird, steht das Menü **Ansicht | MapForce** zur Verfügung. Es enthält dieselben Optionen, wie das Menü **Ansicht** der Standalone-Version von

MapForce.

- Fenster "Bibliotheken"

  Wenn das MapForce-Fenster **Bibliotheken** in Visual Studio nicht angezeigt wird, können Sie es über das Menü **Ansicht | MapForce | Bibliotheken** aktivieren (Dieses Menü steht zur Verfügung, wenn eine Mapping-Datei in Visual Studio geöffnet ist). Sobald das Fenster **Bibliotheken** aktiviert ist, können Sie es wie jede andere andockbare Komponente von Visual Studio an einer bestimmten Stelle andocken.

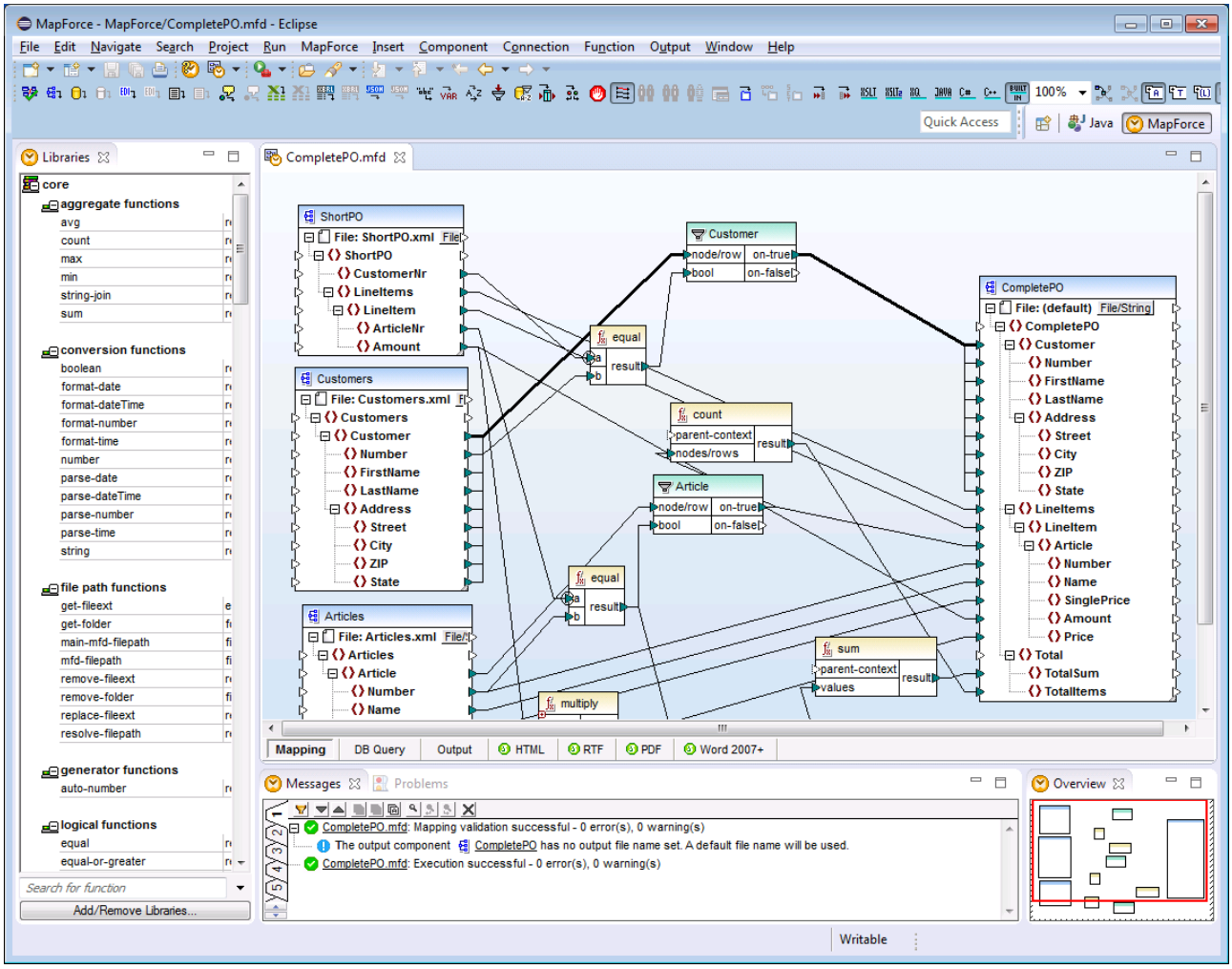
- Hilfe und Support

  Die MapForce Menüs **Hilfe**, **Support Center**, **Auf Updates überprüfen** und **Über** finden Sie im Visual Studio-Menü **Hilfe | MapForce-Hilfe**.

© 2016-2022 Altova GmbH

14 MapForce Plug-in für Eclipse

Eclipse ist ein Open Source Framework, in das unterschiedliche Arten von Applikationen in Form von Plug-ins integriert werden. Sie können MapForce Enterprise und Professional Edition in die Eclipse-Versionen 2022-09, 2022-06, 2022-03, 2021-12 integrieren und haben dann direkt von Eclipse aus Zugriff auf die Funktionalitäten von MapForce.



MapForce Enterprise Edition Plug-in für Eclipse

In den folgenden Kaptiteln finden Sie eine Hilfe zur Installation und Verwendung des MapForce Plug-in für Eclipse.

- Installieren des MapForce Plug-in für Eclipse 1496
- Die MapForce-Perspektive 1498
- Aufrufen häufig verwendeter Menüs und Funktionen 1501
- Arbeiten mit Mappings und Projekten 1504
- Erweitern des MapForce Plug-in für Eclipse 1514

14.1 Installieren des MapForce Plug-in für Eclipse

Voraussetzungen

- Eclipse 2022-09, 2022-06, 2022-03, 2021-12 (http://www.eclipse.org), 64-Bit.
- Java Runtime Environment (JRE) oder Java Development Kit (JDK) für die 64-Bit-Plattform.
- MapForce Enterprise oder Professional Edition 64-Bit.

Anmerkung: Alle oben aufgelisteten Programme müssen die 64-Bit-Plattform haben. Die Integration mit älteren Eclipse 32-Bit-Plattformen wird nicht mehr unterstützt, funktioniert aber eventuell noch.

Mit den obigen Voraussetzungen können Sie das MapForce Integrationspaket (64-Bit) installieren, um MapForce in Eclipse zu integrieren. Die Integration kann entweder während der Installation des Installationspakets oder nach Installation des Integrationspakets manuell von Eclipse aus durchgeführt werden. Das MapForce Integrationspaket kann von https://www.altova.com/de/components/download heruntergeladen werden.

Anmerkung: Eclipse muss während der Installation bzw. Deinstallation des MapForce Integrationspakets geschlossen sein.

Integration von MapForce während der Installation des Integrationspakets

Sie können MapForce während der Installation des MapForce-Integrationspakets in Eclipse integrieren. Gehen Sie dazu folgendermaßen vor:

1. Starten Sie das MapForceIntegrationspaket, um den Installationsassistenten aufzurufen.
2. Führen Sie mit dem Assistenten die ersten Schritte der Installation durch.
3. Aktivieren Sie beim Integrationsschritt die Option *Altova MapForce Plug-in mit diesem Assistenten in Eclipse integrieren* und suchen Sie nach dem Ordner, in dem sich die ausführbare Datei von Eclipse befindet (`eclipse.exe`).
4. Klicken Sie auf **Weiter** und schließen Sie die Installation ab.

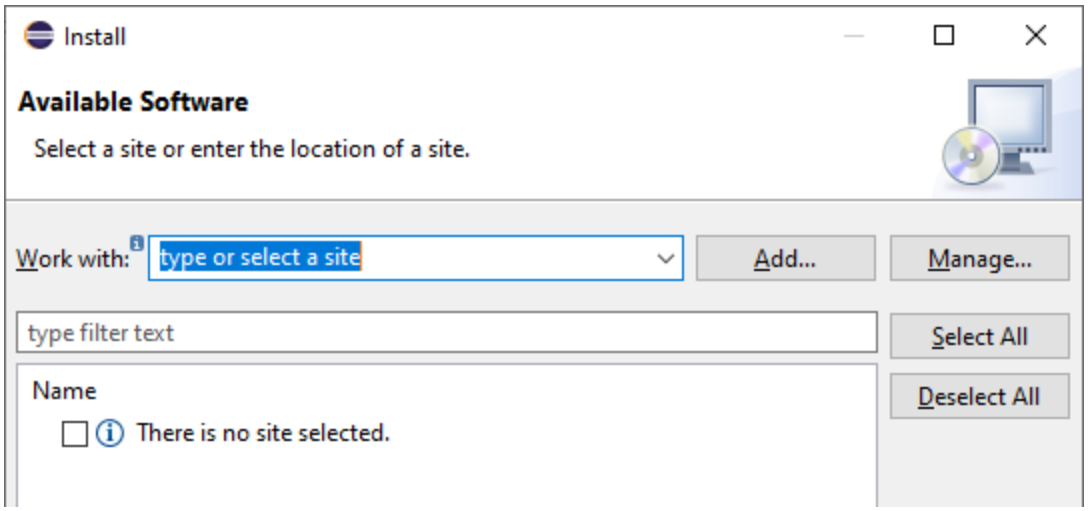
Die MapForce-Perspektive und die entsprechenden Menüs stehen daraufhin beim nächsten Start von Eclipse zur Verfügung.

Manuelle Integration von MapForce in Eclipse

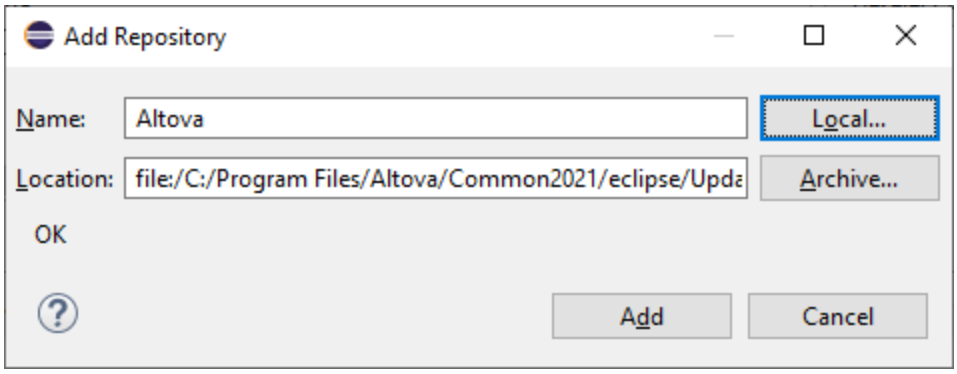
Nach Installation des MapForce-Integrationspakets können Sie MapForce folgendermaßen manuell in Eclipse integrieren:

1. Klicken Sie in Eclipse auf den Menübefehl **Help | Install new Software**.
2. Klicken Sie im Dialogfeld "Install", das daraufhin angezeigt wird, auf die Schaltfläche **Add**.

© 2016-2022 Altova GmbH



3. Klicken Sie im Dialogfeld "Add Repository" auf **Local**. Navigieren Sie zum Ordner `C:\Programme\Altova\Common2023\eclipse\UpdateSite`, wählen Sie ihn aus und klicken Sie auf OK. Geben Sie einen Namen dafür ein (z.B. "Altova").



4. Wiederholen Sie die Schritte 2 bis 3 und wählen Sie diesmal den Ordner `C:\Programme\Altova\<%APPNAMESHORT%>\eclipse\UpdateSite`aus. Geben Sie einen Namen wie z.B. "Altova MapForce" ein.
5. Wählen Sie im Installationsdialogfeld *Only Local Sites* aus. Wählen Sie als nächstes den "Altova category"-Ordner aus und klicken Sie auf **Next**.
6. Überprüfen Sie die zu installierenden Objekte und klicken Sie zum Fortfahren auf **Next**.
7. Aktivieren Sie das entsprechende Kontrollkästchen, um die Lizenzvereinbarung zu akzeptieren.
8. Klicken Sie anschließend auf **Finish**, um die Installation fertig zu stellen.

Anmerkung: Falls es Probleme mit dem Plug-in gibt (z.B. fehlende Symbole), starten Sie Eclipse über die Befehlszeile mit dem `-clean` Flag.

14.2 Die MapForce-Perspektive

Eine Perspektive in Eclipse ist eine Ansicht der Benutzeroberfläche, die mit den Funktionalitäten einer bestimmten Applikation konfiguriert ist. Nachdem MapForce in Eclipse integriert wurde, steht in Eclipse eine neue Perspektive namens MapForce, zur Verfügung. Diese Perspektive ähnelt der Benutzeroberfläche von MapForce und enthält eine Reihe ihrer Komponenten.

Wenn eine Datei geöffnet wird, deren Dateityp (z.B. `.mfd`) mit MapForce verknüpft ist, kann diese Datei in der MapForce Perspektive bearbeitet werden. Ebenso kann eine Datei eines anderen Dateityps in einer anderen Perspektive von Eclipse geöffnet werden. Zusätzlich dazu können Sie die Perspektive für jeden Dateityp wechseln (*siehe unten*) und so die Datei in einer anderen Umgebung bearbeiten bzw. verarbeiten.

Perspektiven haben daher zwei wichtige Vorteile:

1. Die Arbeitsumgebung für die aktive Datei kann schnell gewechselt werden und
2. Sie können zwischen Dateien wechseln, ohne eine neue Entwicklungsumgebung öffnen zu müssen (die mit der Datei verknüpfte Umgebung steht in einer Perspektive zur Verfügung).

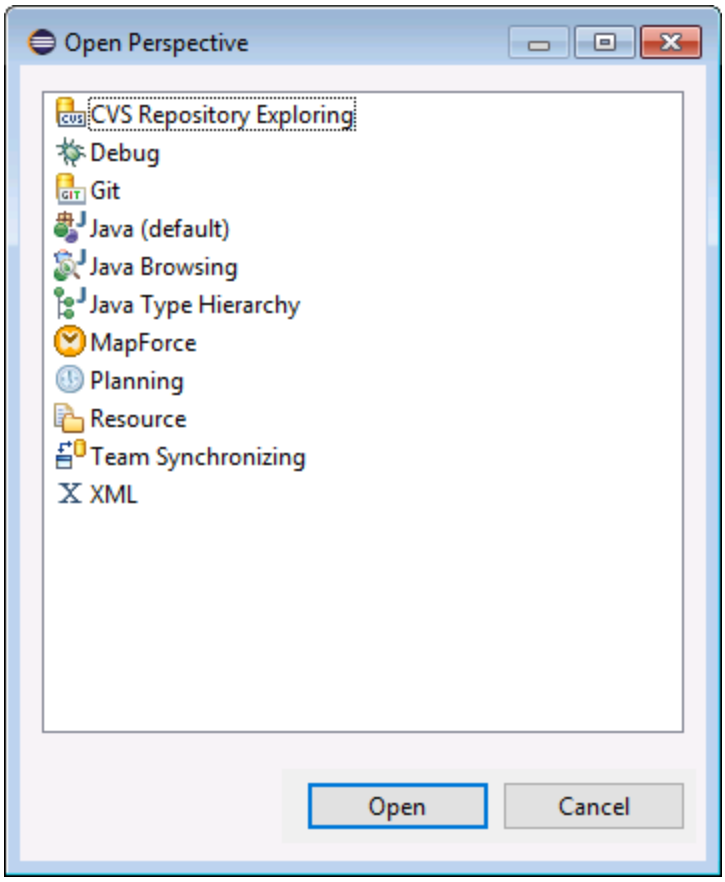
Beim Arbeiten mit der MapForce-Perspektive kommen die folgenden unten beschriebenen Verfahren zum Einsatz:

- Wechseln in die MapForce Perspektive.
- Festlegen der Einstellungen für die MapForce Perspektive.
- Anpassung der MapForce Perspektive.

Wechseln in die MapForce Perspektive

Wählen Sie in Eclipse den Befehl **Window | Perspective | Open Perspective | Other**. Wählen Sie im Dialogfeld, das jetzt angezeigt wird (*Abbildung unten*),**MapForce** aus und klicken Sie auf **Open**.

© 2016-2022 Altova GmbH



Das leere Fenster bzw. das aktive Dokument hat nun die MapForce Perspektive. Auf diese Art kann der Benutzer die Perspektive über das Menü wechseln. Um eine Perspektive von einer anderen Perspektive aus schneller aufrufen zu können, kann die gewünschte Perspektive im Untermenü **Open Perspective** oberhalb des Menübefehls **Other** aufgelistet werden; diese Einstellung kann im Anpassungsdialogfeld (*siehe weiter unten*) vorgenommen werden.

Perspektiven können auch gewechselt werden, wenn eine Datei geöffnet oder zur aktiven gemacht wird. Die Perspektive der Applikation, die mit dem Dateityp einer Datei verknüpft ist, wird automatisch geöffnet, wenn diese Datei das erste Mal geöffnet wird. Vor dem Wechseln der Perspektive wird ein Dialogfeld angezeigt, in dem Sie gefragt werden, ob die Standardperspektive automatisch mit diesem Dateityp verknüpft werden soll. Aktivieren Sie die Option *Do Not Ask Again,* wenn die Perspektive mit dem Dateityp verknüpft werden soll, ohne dass Sie jedes Mal gefragt werden, wenn eine Datei dieses Typs geöffnet wird, und klicken Sie anschließend auf **OK**.

Einstellungen für die MapForce Perspektive

Wählen Sie den Menübefehl **Window | Preferences**, um das Dialogfeld "Preferences" aufzurufen. Wählen Sie in der Liste der Perspektiven auf der linken Seite MapForce aus und wählen Sie dann die gewünschten Einstellungen aus. Klicken Sie anschließend auf **OK**.

Zu den Einstellungen einer Perspektive gehören:

- Der automatische Wechsel in die MapForce-Perspektive, wenn eine Datei eines verknüpften Dateityps (*siehe oben*) geöffnet wird

- Optionen zum Inkludieren oder Exkludieren einzelner MapForce-Symbolleisten
- Zugriff auf die MapForce-Optionen.

Anpassen der MapForce-Perspektive

Mit Hilfe der Anpassungsoptionen können Sie festlegen, welche Shortcuts und Befehle in der Perspektive enthalten sein sollen. Um das Dialogfeld "Customize Perspective" aufzurufen, machen Sie die Perspektive zur aktiven und wählen Sie den Befehl **Window | Perspective | Customize Perspective** aus.

- Auf den Registern *Toolbar Visibility* und *Menu Visibility* können Sie festlegen, welche Symbolleisten und Menüs angezeigt werden sollen.
- Auf dem Register *Action Set Availability* können Sie Aktionsgruppen zu den übergeordneten Menüs und zur Symbolleiste hinzufügen. Um eine Aktionsgruppe zu aktivieren, klicken Sie auf das entsprechende Kontrollkästchen.
- Auf dem Register *Shortcuts* des Dialogfelds "Customize Perspective" können Sie die Shortcuts für Untermenüs festlegen. Aktivieren Sie in der Untermenü-Auswahlliste das gewünschte Untermenü. Wählen Sie anschließend eine Shortcut-Kategorie aus und aktivieren Sie die Shortcuts, die in der Perspektive inkludiert werden sollen.

Klicken Sie auf **Apply and Close**, um die Anpassung abzuschließen und damit die Änderungen wirksam werden.

Übersicht über die MapForce-Perspektive

Standardmäßig ist die MapForce-Perspektive in Eclipse folgendermaßen angeordnet:

- Das Mapping-Design-Fenster steht als Eclipse-Editor zur Verfügung. Es hat dieselben Register und Funktionen wie die Standalone-Edition von MapForce.
- Das Fenster "Bibliotheken" steht links vom Mapping-Editor in Form einer Eclipse-Ansicht zur Verfügung. Wenn diese Ansicht nicht zu sehen ist, wechseln Sie in die MapForce-Perspektive und wählen Sie den Menübefehl **Fenster | Sicht anzeigen | Bibliotheken**. Über das Fenster "Bibliotheken" können Sie mit vordefinierten oder benutzerdefinierten Funktionen und Funktionsbibliotheken arbeiten.
- Das Fenster "Meldungen" steht unterhalb des Hauptmapping-Editors in Form einer Eclipse-Ansicht zur Verfügung. Wenn diese Ansicht nicht zu sehen ist, wechseln Sie in die MapForce-Perspektive und wählen Sie den Menübefehl **Fenster | Sicht anzeigen | Meldungen**. Im Fenster "Meldungen" werden Validierungsmeldungen, Fehler und Warnungen angezeigt.
- Das Fenster "Übersicht" steht in Form einer Eclipse-Ansicht zur Verfügung. Wenn diese Ansicht nicht zu sehen ist, wechseln Sie in die MapForce-Perspektive und wählen Sie den Menübefehl **Fenster | Sicht anzeigen | Übersicht**. In dieser Ansicht können Sie bei großen Mappings schnell zu einem bestimmten Bereich des Mapping-Designs navigieren.

© 2016-2022 Altova GmbH

14.3 Aufrufen häufig verwendeter Menüs und Funktionen

Sie können in Eclipse die meisten MapForce-Funktionen über dasselbe Menü wie in der Standalone-Version aufrufen. Es gibt allerdings einige Eclipse-spezifische Abweichungen, die unten aufgeführt sind. Dies ist die Standardkonfiguration; Sie können die Einstellungen der Benutzeroberfläche allerdings auf Wunsch über Eclipse weiter anpassen (siehe Die MapForce-Perspektive 1498).

Anmerkung: Einige MapForce-Menügruppen oder - befehle sind in Eclipse deaktiviert (oder nicht verfügbar), wenn der Kontext nicht relevant ist. So steht z.B. das Menü **Einfügen** nur zur Verfügung, wenn eine Mapping-Design-Datei (.mfd) in Eclipse aktiv ist.

Informationen zu den MapForce-Standardmenüs finden Sie in der Menüreferenz 1516.

Allgemeine MapForce-Befehle

In der Standalone-Edition von MapForce stehen die Befehle zu Mapping-Design-Dateien (wie z.B. **Validieren**, **Auf FlowForce Server bereitstellen**, **Code generieren**, usw.) im Menü **Datei** zur Verfügung. In Eclipse stehen diese Befehle im Menü **MapForce** bzw. in der MapForce-Symbolleiste zur Verfügung. Beachten Sie, dass die Befehle zum Öffnen oder Speichern von Dateien (einschließlich MapForce-Projektdateien) im Menü **Datei** von Eclipse zur Verfügung stehen.

Auch MapForce-Designs können im MapForce-Menü ausgewählt werden.



Die MapForce-Symbolleiste in Eclipse

Die Symbolleisten-Schaltfläche  öffnet die MapForce-Hilfedatei.

Über die Symbolleisten-Schaltfläche  werden MapForce-dateispezifische Befehle angezeigt. Wenn Sie diese Schaltfläche erweitern, hängt es von der im Eclipse-Editor derzeit aktiven Datei ab, welche Befehle zur Verfügung stehen. So stehen z.B. die Befehle für Mapping-Dateien (.mfd) zur Verfügung, wenn eine solche Datei im Eclipse-Editor aktiv (im Fokus) ist.

Debuggen von Mappings

Die Befehle zum Debuggen von Mappings befinden sich im Menü **MapForce | Debuggen** und in der Debugging-Symbolleiste. Beachten Sie, dass die Debugging-Tastenkürzel (wie F5, F10, usw) in Eclipse bereits vorreserviert sind und nicht zur Verfügung stehen.

Globale Ressourcen

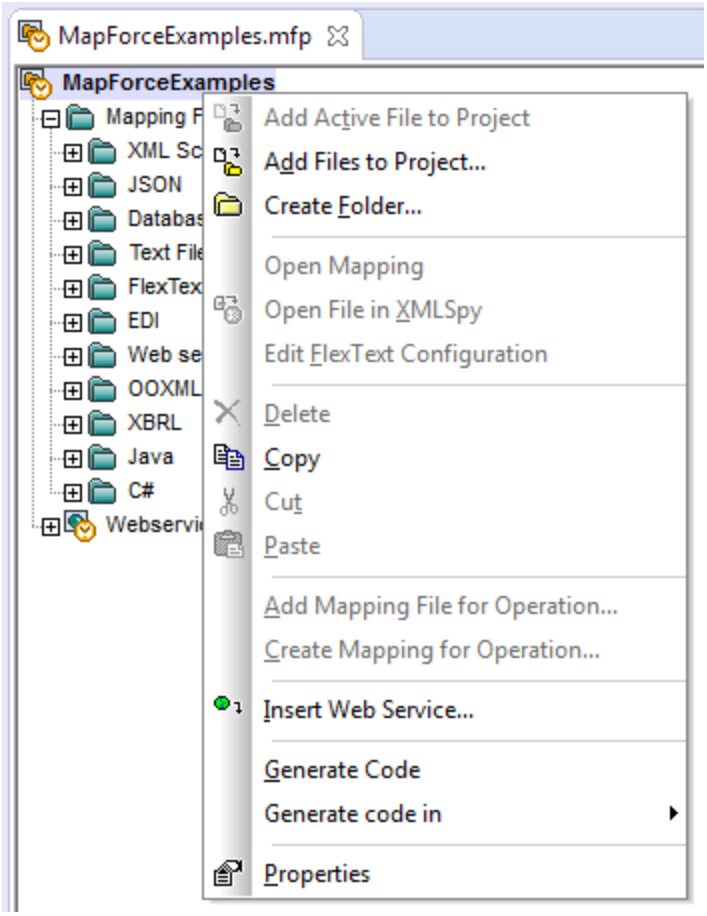
Um globale Ressourcen aufrufen oder verwalten zu können, wählen Sie eine der folgenden Methoden:

- Klicken Sie auf die MapForce-Symbolleisten-Schaltfläche, um sie zu erweitern und klicken Sie auf **Globale Ressourcen**.
- Wählen Sie im Menü **MapForce** den Befehl **Globale Ressourcen**.

MapForce-Projekte

In der Standard Edition von MapForce enthält das Menü **Projekt** eine Reihe von Befehlen zu Mapping-Projektdateien (.mfp). In Eclipse können Sie diese Befehle folgendermaßen aufrufen:

- Die Befehle zum Öffnen oder Speichern eines Projekts finden Sie im Eclipse-Menü **Datei**.
- Andere projektspezifische Befehle stehen in Form von Kontextmenübefehlen zur Verfügung. Um die Kontextbefehle aufzurufen, erstellen oder öffnen Sie ein MapForce-Projekt (.mfp) in Eclipse und klicken Sie mit der rechten Maustaste auf das Projekt.

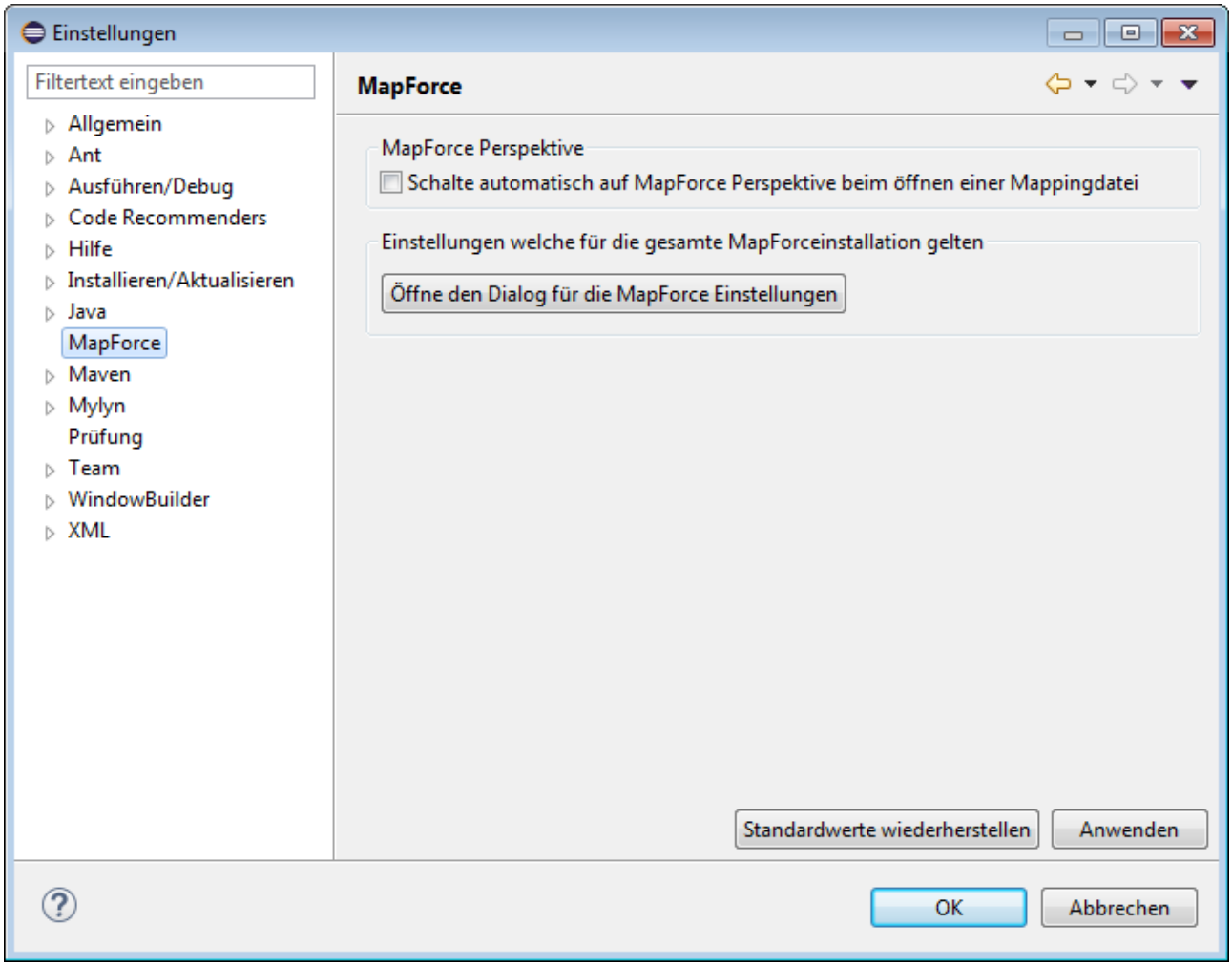


Beachten Sie, dass Sie in Eclipse neben MapForce-Standardprojekten (.mfp) auch Projekte vom Typ "MapForce/Eclipse" erstellen können. Solche Projekte haben eine zweifache Natur und können für die automatische Erstellung und Generierung von MapForce-Code konfiguriert werden. Siehe Arbeiten mit Mappings und Projekten 1504.

MapForce-Optionen

Die MapForce-Optionen stehen über das Menü **Fenster | Benutzervorgaben** zur Verfügung. Wählen Sie in diesem Menü den Befehl **MapForce** und klicken Sie anschließend auf **Dialog für die MapForce-Einstellungen öffnen**.

© 2016-2022 Altova GmbH



Dialogfeld "Einstellungen"

Fenster "Bibliotheken"

Das MapForce Fenster "Bibliotheken" steht in Eclipse Form einer Ansicht zur Verfügung. Standardmäßig befindet sich diese Ansicht links vom Editor-Hauptfenster (Alle MapForce-Ansichten werden in der Eclipse-Perspektive angezeigt, wenn die MapForce-Perspektive eingeschaltet ist, siehe auch Die MapForce-Perspektive 1498).

MapForce Plug-in-Version

Um die aktuell installierte Version des MapForce Plug-in für Eclipse zu sehen, wählen Sie die Eclipse-Menüoption **Hilfe | Info über Eclipse** und klicken Sie auf die MapForce-Schaltfläche.

Hilfe und Support

Die MapForce Menüs "Hilfe", "Support Center", "Auf Updates überprüfen" und "Über" finden Sie im Eclipse-Menü **Hilfe | MapForce-Hilfe**.

14.4 Arbeiten mit Mappings und Projekten

Wenn das MapForce Plug-in für Eclipse installiert ist, können Sie von Eclipse aus dieselben Mappings und Mapping-Projektarten wie in der Standalone Edition von MapForce erstellen. Um Mappings zu erstellen, kompilieren und bereitzustellen sowie Mapping Code zu generieren, können Sie entweder ein neues Eclipse-Projekt erstellen oder ein vorhandenes (z.B. ein Java-Projekt, zu dem Sie MapForce Mappings hinzufügen möchten) verwenden.

Außerdem können Sie mit allen Ihren Mappings in einem speziellen Projekttyp arbeiten, der in Eclipse nach Installation des MapForce Plug-in zur Verfügung steht, nämlich einem **MapForce/Eclipse-Projekt**. Wenn Sie keine weiteren Anpassungen vornehmen, ist einem MapForce/Eclipse-Projekt sowohl ein Java Builder als auch ein MaForce Code Generation Builder zugewiesen. Zusätzlich hat das Projekt zwei "Natures": die MapForce Nature und die JDT (Java Development Tools) Nature. Infolgedessen verhält sich ein MapForce/Eclipse-Projekt folgendermaßen, wenn Sie eine seiner Ressourcen (wie z.B. die Mapping-Datei) speichern oder ändern:

- Wenn die Option **Projekt > Automatisch erstellen** aktiv ist, wird der Mapping-Code automatisch generiert. Wenn das MapForce/Eclipse-Projekt eine oder mehrere MapForce-Projektdateien enthält, hängen die Codegenerierungssprachen und die Ausgabeordner von den Einstellungen in der jeweiligen Projektdatei ab. Andernfalls werden Sie von Eclipse nach einem Ordner gefragt.
- Fehler und Ausgabemeldungen werden in den Ansichten "Meldungen" und "Probleme" angezeigt.

Dieser Abschnitt enthält die folgenden Kapitel:

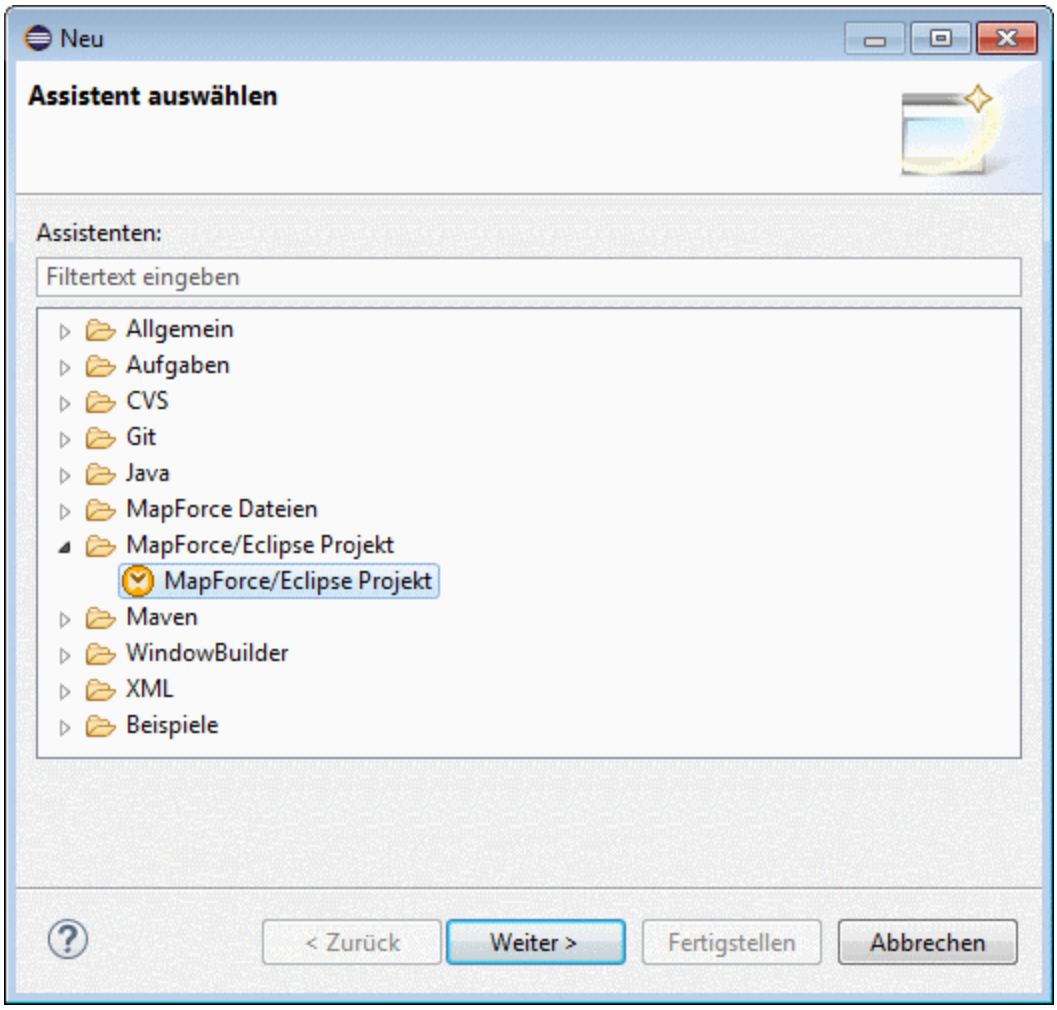
- Erstellen eines MapForce/Eclipse-Projekts 1504
- Erstellen neuer Mappings 1506
- Importieren bestehender Mappings in ein Eclipse-Projekt 1508
- Konfigurieren eines automatischen Build und Generierung von MapForce Code 1511

14.4.1 Erstellen eines MapForce/Eclipse-Projekts

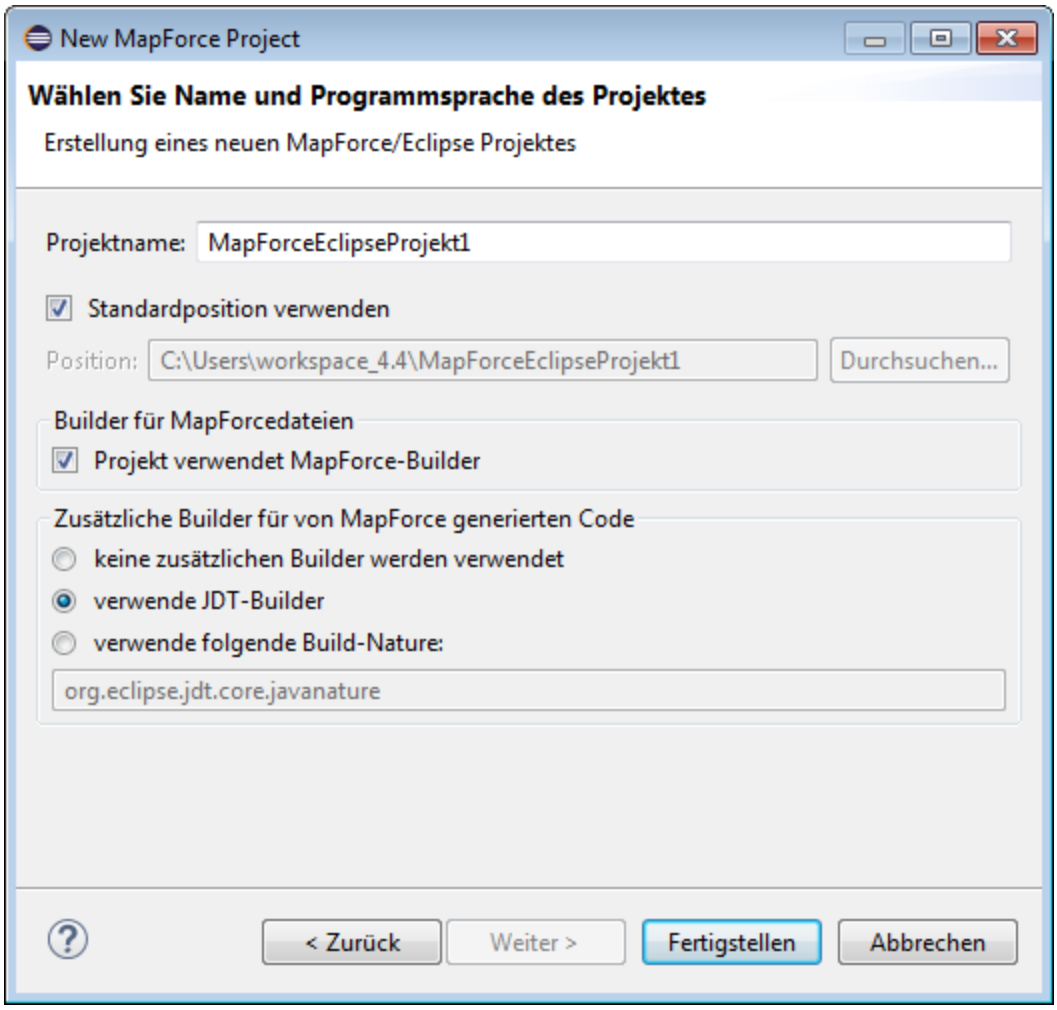
So erstellen Sie ein MapForce/Eclipse-Projekt:

1. Klicken Sie im Menü **Datei** auf **Neu | Andere**.
2. Wählen Sie die Kategorie **MapForce/Eclipse Projekt** aus.

© 2016-2022 Altova GmbH



3. Klicken Sie auf **Weiter**.



4. Geben Sie einen Projektnamen ein und wählen Sie einen Speicherpfad für das Projekt aus. Belassen Sie die Optionen **Projekt verwendet MapForce-Builder** und **verwende JDT-Builder** unverändert.
5. Klicken Sie auf **Fertigstellen**.

14.4.2 Erstellen neuer Mappings

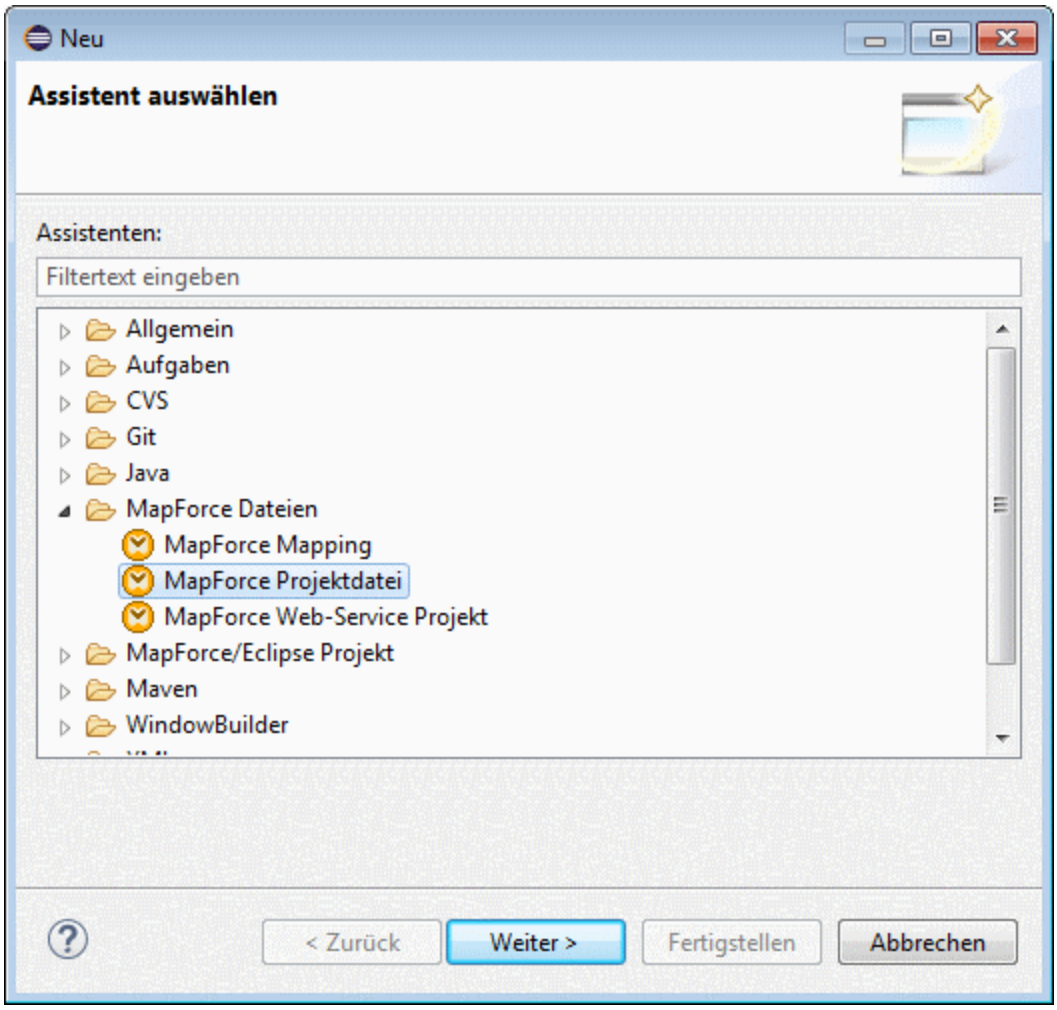
In einem Eclipse-Projekt können die folgenden MapForce-Dateitypen erstellt werden:

- MapForce Mappings
- MapForce Projektdateien
- MapForce Webservice-Projekte (verfügbar in der MapForce Enterprise Edition)

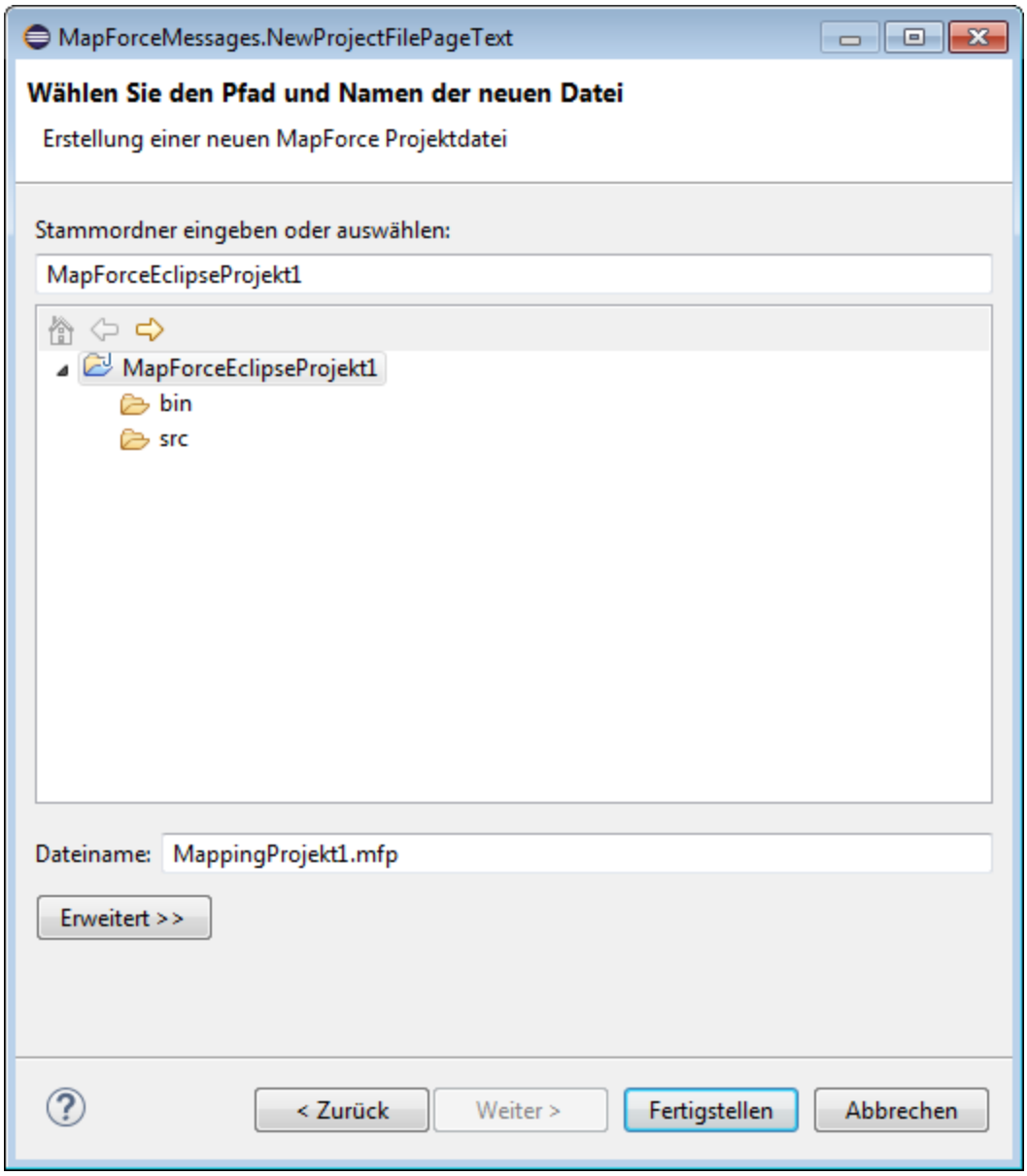
So erstellen Sie in einem Eclipse-Projekt eine dieser Dateiarten:

1. Erstellen Sie ein neues Eclipse-Projekt oder öffnen Sie ein bestehendes.
2. Klicken Sie im Menü **Datei** auf **Neu** und dann auf **Andere**.

© 2016-2022 Altova GmbH



3. Wählen Sie den gewünschten Dateityp im Assistentendialogfeld aus und klicken Sie auf **Weiter**.



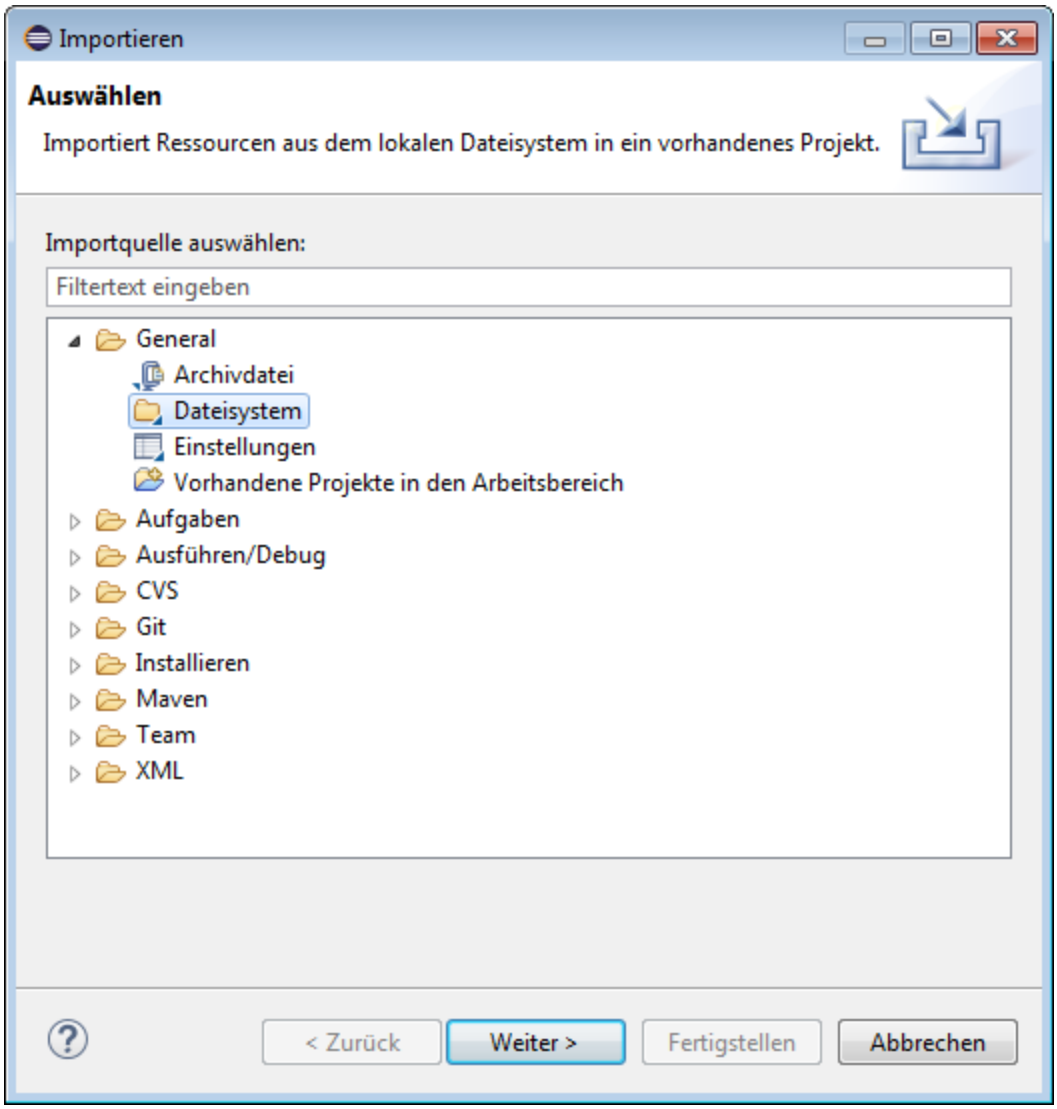
4. Wählen Sie eine übergeordneten Ordner in Ihrem bestehenden Projekt aus.

14.4.3 Importieren bestehender Mappings in ein Eclipse-Projekt

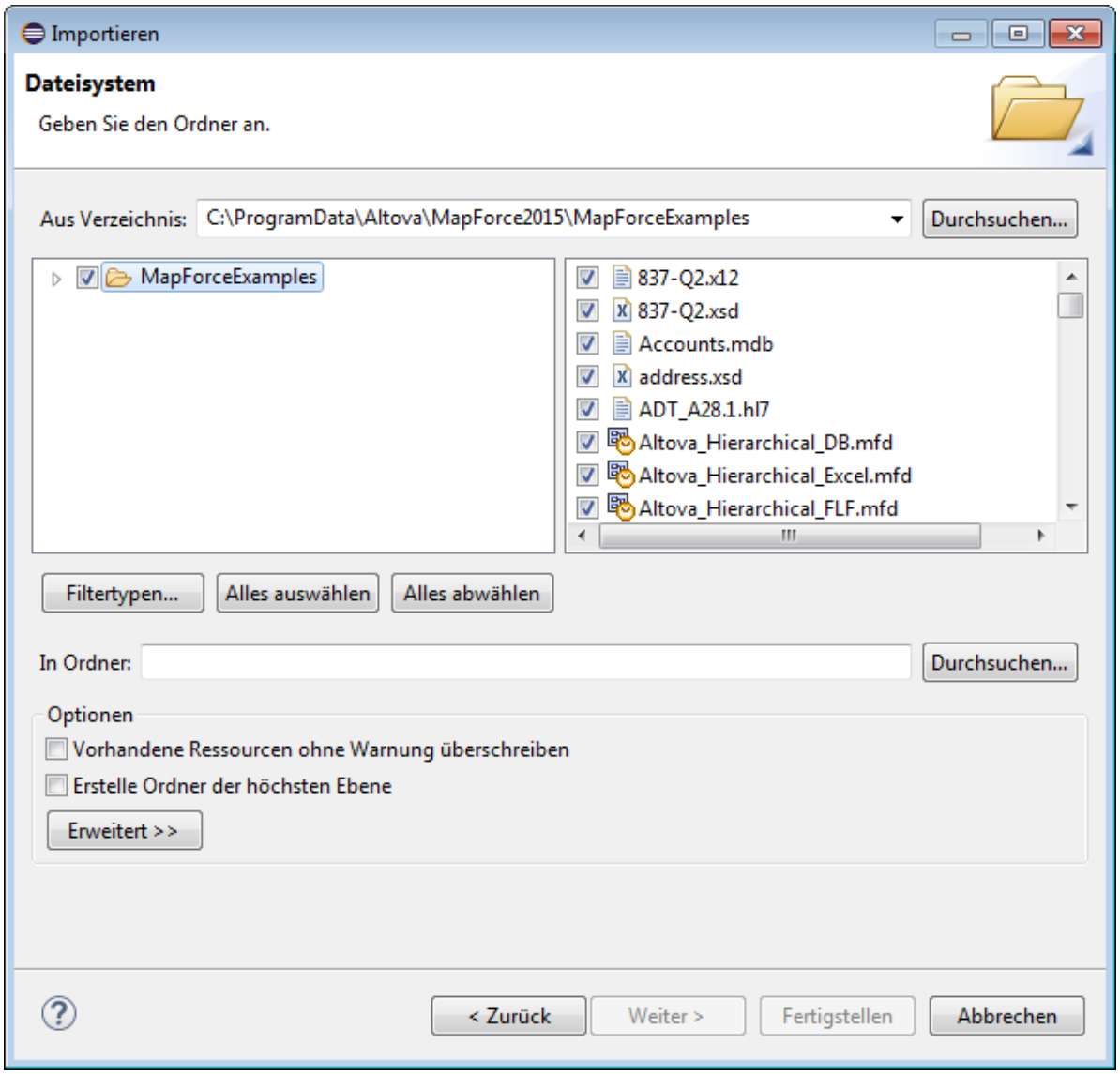
So importieren Sie MapForce-Mappings und die dazugehörigen Dateien in ein bestehendes Eclipse-Projekt:

1. Öffnen Sie das gewünschte Projekt, in das die Dateien importiert werden sollen.
2. Klicken Sie im Menü **Datei** auf **Importieren**.

© 2016-2022 Altova GmbH

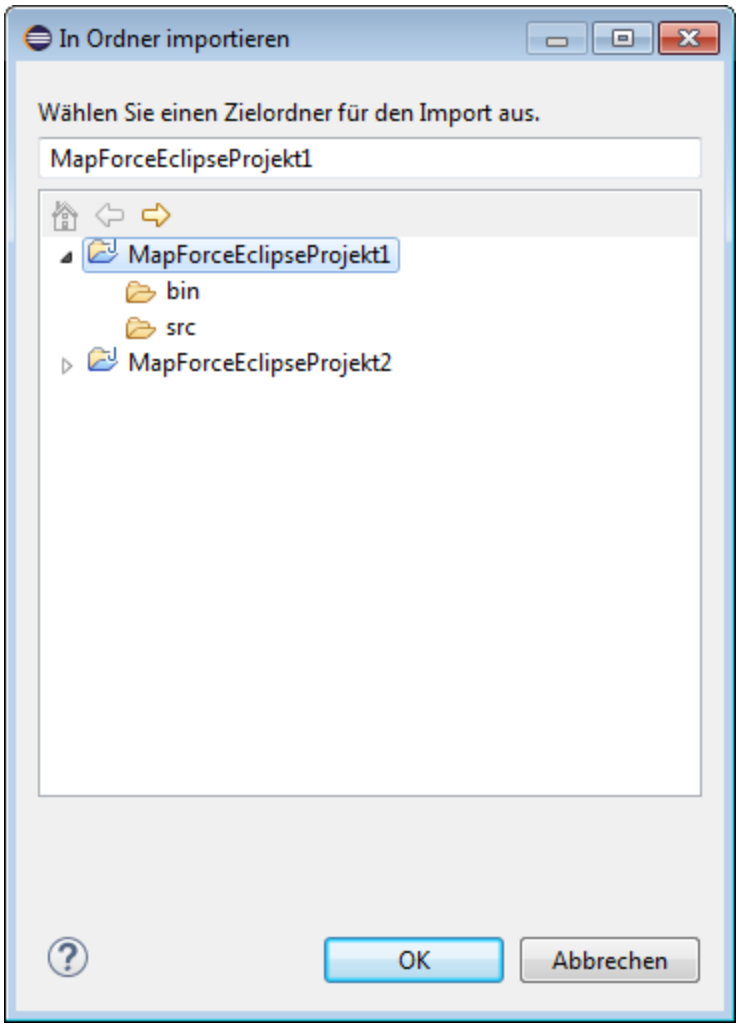


3. Wählen Sie **Dateisystem** und klicken Sie auf **Weiter**.



4. Navigieren Sie neben **Aus Verzeichnis** zu den zu importierenden Dateien und wählen Sie diese aus.
5. Klicken Sie neben **In Ordner** auf **Durchsuchen** und wählen Sie das Projekt aus, zu dem die Dateien hinzugefügt werden sollen *MapForceEclipseProject1*).

© 2016-2022 Altova GmbH



6. Klicken Sie auf **OK** und anschließend auf **Fertigstellen**.

14.4.4 Konfigurieren eines automatischen Build und Generierung von MapForce Code

Die automatische Erstellung und Generierung von MapForce-Code ist standardmäßig in jedem MapForce/Eclipse-Projekt aktiviert (siehe Erstellen eines MapForce/Eclipse-Projekts[1504]). Wenn Sie die automatische Erstellung und Generierung von MapForce-Code in einem bestehenden Projekt, das nicht vom Typ *MapForce/Eclipse* ist, aktivieren möchten, können Sie den *MapForce-Codegenerierungs*-Builder und die MapForce-Nature manuell zu diesem Projekt hinzufügen.

So fügen Sie den MapForce Codegenerierungs-Builder zu einem Projekt hinzu:

- Fügen Sie zur Eclipse **.project**-Datei die unten markierten Zeilen hinzu:

```
<buildSpec>
    <buildCommand>
```

```
            <name>org.eclipse.jdt.core.javabuilder</name>
            <arguments>
            </arguments>
        </buildCommand>
        <buildCommand>
            <name>com.altova.mapforceeclipseplugin.MapForceBuilder</name>
            <arguments>
            </arguments>
        </buildCommand>
    </buildSpec>
```

So fügen Sie die MapForce-Nature zu einem Projekt hinzu:

- Fügen Sie zur Eclipse **.project**-Datei die unten markierten Zeilen hinzu:

```
    <natures>
        <nature>org.eclipse.jdt.core.javanature</nature>
        <nature>com.altova.mapforceeclipseplugin.MapForceNature</nature>
    </natures>
```

Tipp: Sie können die **.project**-Datei schnell über die Navigatoransicht von Eclipse öffnen (Um diese Ansicht zu aktivieren, wählen Sie den Menübefehl **Fenster | Sicht anzeigen | Navigator**).

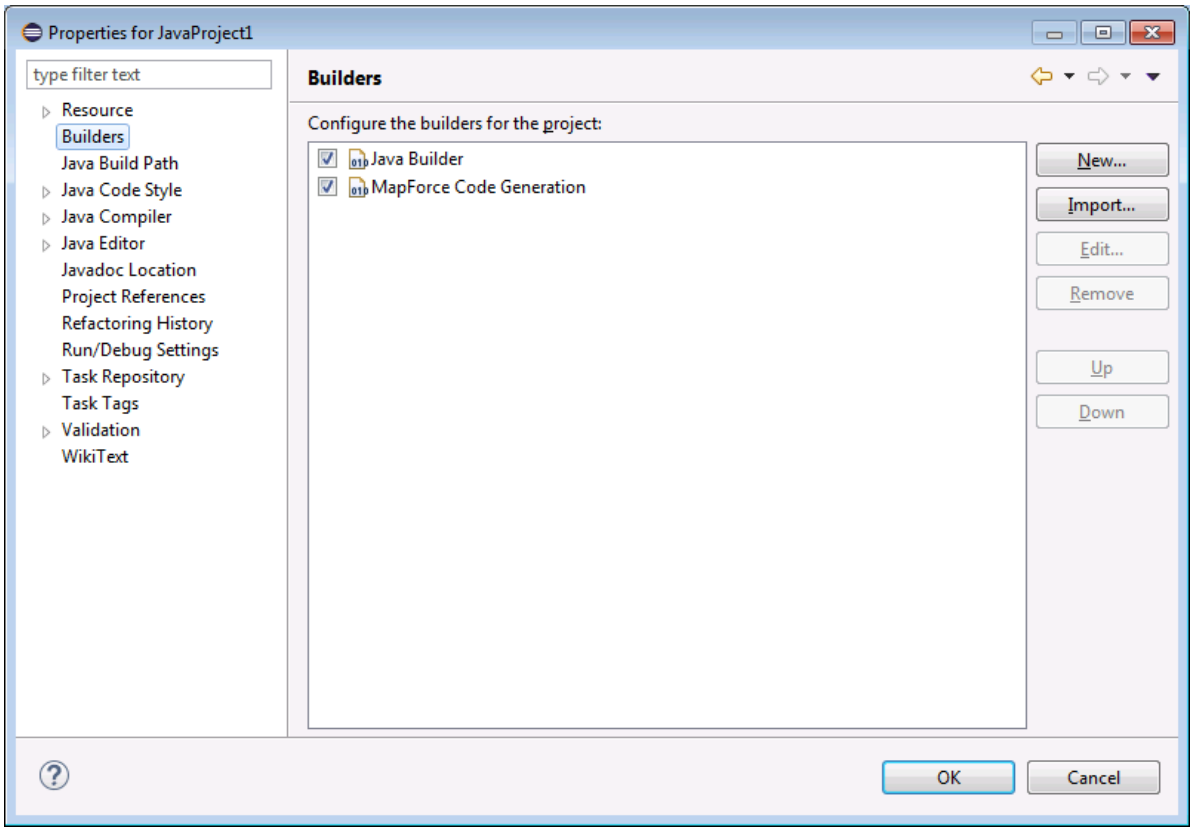
So schalten Sie die automatische MapForce-Codegenerierung ein/aus:

- Klicken Sie im Menü **Projekt** auf **Automatisch erstellen**.

So deaktivieren Sie den MapForce Codegenerierungs-Builder:

1. Klicken Sie im Menü **Projekt** auf **Eigenschaften**.
2. Klicken Sie auf **Builders**.

© 2016-2022 Altova GmbH



3. Klicken Sie auf das entsprechende Kontrollkästchen, um die **MapForce-Codegenerierung** zu deaktivieren.

14.5 Erweitern des MapForce Plug-in für Eclipse

Das MapForce Plug-in bietet einen Erweiterungspunkt für Eclipse mit der ID "**com.altova.mapforceeclipseplugin.MapForceAPI**". Über diesen Erweiterungspunkt (Extension Point) können Sie die Funktionalitäten des MapForce Plug-in anpassen oder erweitern. Über den Erweiterungspunkt haben Sie Zugriff auf die COM-Schnittstelle des MapForce Control und der MapForceAPI[1722].

Ihr MapForce Eclipse Installationspaket enthält ein einfaches Beispiel für ein Plug-in, das diesen Erweiterungspunkt verwendet. Es überprüft, ob ein neues MapForce Mapping ein File open-Ereignis enthält und setzt den Zoomfaktor der Mapping-Ansicht auf 70%.

Die JavaDoc-Dokumentation zum Erweiterungspunkt finden Sie im MapForce Plug-in-Installationsverzeichnis (**C:\Programme\Altova\MapForce2023\eclipse\docs**).

Bevor Sie das MapForce-Beispiel-Plug-in installieren und ausführen, vergewissern Sie sich, dass die folgenden Voraussetzungen erfüllt werden:

- Sie verwenden 64-Bit-Java, 64-Bit-Eclipse, 64-Bit-MapForce und ein 64-Bit-MapForce Integration Package.
- Das JDT (Java Development Tools) Plug-in ist installiert.
- Das Eclipse PDE (Plug-in Development Environment) ist installiert.

So importieren Sie das MapForce Beispielprojekt-Plug-in Ihrem Arbeitsbereich:

1. Starten Sie Eclipse.
2. Klicken Sie im Menü **Datei** auf **Importieren**.
3. Wählen Sie den Befehl **Allgemein | Vorhandene Projekte in den Arbeitsbereich** und klicken Sie auf **Weiter**.
4. Klicken Sie auf die Schaltfläche Durchsuchen... neben dem Feld "'Stammverzeichnis auswählen" und wählen Sie das Verzeichnis des Beispielprojekts z.B. **C:\Programme\Altova\MapForce2023\eclipse\workspace\MapForceExtension**.
5. Aktivieren Sie die Option **Projekte in den Arbeitsbereich kopieren** und klicken Sie auf "**Fertigstellen**". In Ihrem Arbeitsbereich wurde nun ein neues Projekt namens "MapForceExtension" erstellt.

So führen Sie das Beispiel-Erweiterungs-Plug-in aus:

1. Wechseln Sie zur Java-Perspektive.
2. Wählen Sie die Menüoption **Ausführen** | **Ausführungskonfigurationen**.
3. Klicken Sie mit der rechten Maustaste auf **Eclipse Application** und wählen Sie **Neu**. (Wenn "Eclipse Application" in der Liste nicht angezeigt wird, wurden die Eclipse Plug-In Development Tools in Ihrer Eclipse-Umgebung nicht installiert. Um die Eclipse Plug-in Development Tools zu installieren, klicken Sie im Menü **Hilfe** auf **Neue Software installieren** und installieren Sie von der Download-Webseite " Eclipse Project Updates" die "Eclipse Plugin Development Tools".
4. Geben Sie einen Namen für Ihre neue Konfiguration ein (in diesem Beispiel SampleMapForcePlugin) und klicken Sie auf **Anwenden**.
5. Überprüfen Sie, ob der MapForceClient-Arbeitsbereich auf dem Register 'Plug-ins' ausgewählt ist.
6. Klicken Sie auf **Ausführen**. Daraufhin wird eine neue Eclipse Workbench geöffnet.

© 2016-2022 Altova GmbH

7. Öffnen Sie ein beliebiges MapForce Mapping in der neuen Workbench. Es wird mit einem Zoomfaktor von 70 % geöffnet.

15 Menübefehle

Der folgende Abschnitt enthält eine Beschreibung der MapForce-Menübefehle. Es stehen die folgenden Menübefehle zur Verfügung:

- Datei 1517
- Bearbeiten 1520
- Einfügen 1521
- Projekt 1524
- Komponente 1526
- Verbindung 1528
- Funktion 1529
- Ausgabe 1530
- Debuggen 1532
- Ansicht 1533
- Extras 1535
- Fenster 1556
- Hilfe 1558

© 2016-2022 Altova GmbH

15.1 Datei

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Datei**.

- Neu

  Erstellt ein neues Mapping-Dokument. In der Professional und der Enterprise Edition können Sie auch ein Mapping-Projekt (`.mfp`) erstellen. Nähere Informationen dazu finden Sie unter Projekte[119].

- Öffnen

  Öffnet eine zuvor gespeicherte Mapping-Design-Datei (`.mfd`). In der Professional und der Enterprise Edition können Sie auch ein Mapping-Projekt (`.mfp`) öffnen. Nähere Informationen dazu finden Sie unter Projekte[119].

- Speichern/Speichern unter/Alles speichern

  Mit der Option **Speichern** wird das gerade aktive Mapping unter seinem aktuellen Namen gespeichert. Mit der Option **Speichern unter** können Sie das gerade aktive Mapping unter einem anderen Namen speichern. Mit dem Befehl **Alles speichern** werden alle aktuell geöffneten Mapping-Dateien gespeichert.

- Neu laden

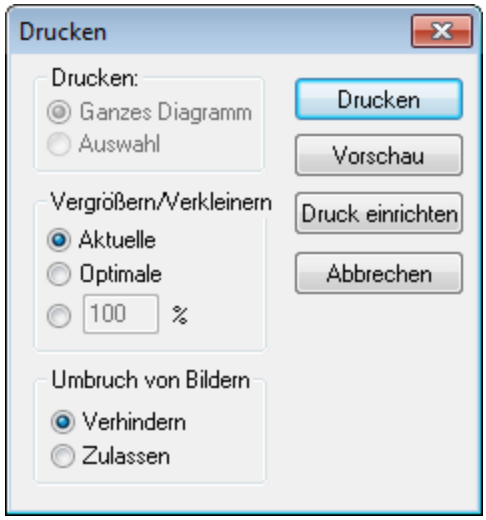
  Durch Neuladen des aktuell aktiven Mappings werden ihre letzten Änderungen rückgängig gemacht.

- Schließen/Alle schließen

  Mit dem Befehl **Schließen** wird das aktuell aktive Mapping geschlossen. Mit dem Befehl **Alle schließen** werden alle gerade offenen Mappings geschlossen. Sie werden gefragt, ob Sie eine der nicht gespeicherten Dateien speichern möchten.

- Drucken/Druckvorschau/Druckereinrichtung

  Mit dem Befehl **Drucken** wird das Dialogfeld **Drucken** (*siehe unten*) aufgerufen, über das Sie Ihr Mapping ausdrucken können. Mit **Aktuelle** wird der aktuell definierte Zoom-Faktor des Mappings beibehalten. Bei Auswahl von **Optimale** wird das Mapping auf Seitengröße vergrößert/verkleinert. Sie können auch einen numerischen Zoom-Faktor angeben. Die Bildlaufleisten der Komponente werden nicht gedruckt. Außerdem können Sie festlegen, ob Grafiken auf mehrere Seiten umbrochen werden sollen oder nicht.



Mit dem Befehl **Druckvorschau** wird dasselbe **Druckdialogfeld** mit denselben Einstellungen wie oben beschrieben, aufgerufen. Mit dem Befehl **Druckereinrichtung** wird das Dialogfeld **Druckereinrichtung** geöffnet, in dem Sie einen Drucker auswählen und die Papiereinstellungen konfigurieren können.

- Mapping validieren

  Mit dem Befehl **Mapping validieren** wird überprüft, ob alle Mappings gültig sind und entsprechende Warn-, Informations- oder Fehlermeldungen angezeigt. Nähere Informationen dazu finden Sie unter Validierung[107].

- Mapping-Einstellungen

  Öffnet das Dialogfeld Mapping-Einstellungen[116], in dem Sie die dokumentspezifischen Einstellungen definieren können.

- Anmeldeinformationen-Manager öffnen (*Enterprise Edition*)

  Öffnet den **Anmeldeinformationen-Manager**, in dem Sie Anmeldeinformationen, die in Mappings mit HTTP-Authentifizierung oder OAuth 2.0-Autorisierung erforderlich sind, verwalten können. Nähere Informationen dazu finden Sie unter Anmeldeinformationen-Manager[1357].

- Code in ausgewählter Sprache generieren/Code generieren in

  Generiert Code in der aktuell ausgewählten Sprache. Nähere Informationen über die verfügbaren Transformationssprachen finden Sie unter Transformationssprachen[28].

  Code generieren in | XSLT (XSLT2, XSLT3)
  Mit diesem Befehl wird/werden anhand der Quelldatei(en) die XSLT-Datei(en) generiert. Nach Auswahl dieser Option wird das Dialogfeld zum **Suchen des Ordners** geöffnet, in dem Sie den Speicherort der XSLT-Datei angeben müssen. Der Name der generierten XSLT-Datei(en) wird im Dialogfeld "Mapping-Einstellungen"[116] definiert.

© 2016-2022 Altova GmbH

Code generieren in | XQuery | Java | C# | C++ (Professional und Enterprise Edition)
Mit diesem Befehl wird Quellcode in einer der ausgewählten Transformationssprachen generiert. Nach Auswahl dieser Option wird das Dialogfeld zum **Suchen des Ordners** geöffnet, in dem Sie den Speicherort der generierten Dateien auswählen müssen. Die Namen der generierten Applikationsdateien (sowie die `*.csproj`-C#-Projektdatei, die `*.sln`-Datei sowie die `*.vcproj` Visual C++-Projektdatei) werden im Dialogfeld Mapping-Einstellungen[116] definiert.

Der durch den ausgeführten Code erstellte Dateiname wird im Feld **XML-Output-Datei** des Dialogfelds Komponenteneinstellungen[126] angezeigt, wenn es sich beim Ziel um ein XML/Schema-Dokument handelt.

- Zu MapForce Server-Ausführungsdatei kompilieren (*Professional und Enterprise Edition*)

  Generiert eine Datei, die auf MapForce Server ausgeführt werden kann, um die Mapping-Transformation auszuführen, siehe Kompilieren eines MapForce Mappings[1442].

- Auf FlowForce Server bereitstellen (*Professional und Enterprise Edition*)

  Stellt das gerade aktive Mapping auf FlowForce Server bereit. Nähere Informationen dazu finden Sie unter Bereitstellen von Mappings auf FlowForce Server[1445].

- Dokumentation generieren (*Professional und Enterprise Edition*)

  Generiert detaillierte Dokumentation zu Ihren Mapping-Projekten in verschiedenen Ausgabeformaten. Nähere Informationen dazu finden Sie unter Generieren von Mapping-Dokumentation[1399].

- Letzte Dateien

  Zeigt eine Liste der zuletzt geöffneten Dateien an.

- Beenden

  Beendet die Applikation. Sie werden gefragt, ob Sie nicht gespeicherte Dateien speichern möchten.

15.2 Bearbeiten

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Bearbeiten**. Die meisten der Befehle in diesem Menü werden erst aktiv, wenn Sie das Ergebnis eines Mappings im **Ausgabe**-Fenster bzw. Code z.B. im **XSLT**-Fenster ansehen.

- Rückgängig

  In MapForce steht eine unbegrenzte Anzahl an "Rückgängig"-Schritten zur Verfügung, mit denen Sie Ihr Mapping Schritt für Schritt wieder rückgängig machen können. Sie können Aktionen auch über die Symbolleisten-Schaltfläche  rückgängig machen.

- Wiederherstellen

  Mit Hilfe des Befehls "Wiederherstellen" können Sie zuvor rückgängig gemachte Aktionen wiederholen. Sie können sich innerhalb des Verlaufs der rückgängig gemachten Schritte vorwärts und rückwärts bewegen. Sie können Aktionen auch über die Symbolleisten-Schaltfläche  wiederherstellen.

- Suchen

  Dient zum Suchen von bestimmtem Text in den Fenstern **XQuery** (*Professional und Enterprise Edition*), **XSLT**, **XLST2**, **XSLT3** und **Ausgabe**. Sie können die Suche auch über die Symbolleisten-Schaltfläche  durchführen.

- Weitersuchen

  Sucht nach der nächsten Stelle, an der der Such-String vorkommt. Sie können dies auch über die Symbolleisten-Schaltfläche  tun.

- Vorheriges suchen

  Sucht nach der vorherigen Stelle, an der der Such-String vorkommt. Dies lässt sich auch über die Symbolleisten-Schaltfläche  bewerkstelligen.

- Ausschneiden/Kopieren/Einfügen/Löschen

  Mit den Windows-Standardbearbeitungsbefehlen können Sie beliebige im Mapping-Fenster angezeigte Komponenten oder Funktionen ausschneiden, kopieren, einfügen und löschen.

- Alle auswählen

  Markiert alle Komponenten im Fenster **Mapping** oder den Text/Code in den Fenstern **XQuery** (*Professional und Enterprise Edition*), **XSLT**, **XSLT2**, **XSLT3** und **Ausgabe**.

© 2016-2022 Altova GmbH

15.3 Einfügen

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Einfügen**.

- XML-Schema/Datei

  Fügt eine XML-Schema- oder Instanzdatei zum Mapping hinzu. Wenn Sie eine XML-Datei ohne Schemareferenz auswählen, kann MapForce ein passendes XML-Schema generieren[125]. Wenn Sie ein XML-Schema auswählen, werden Sie aufgefordert optional eine XML-Instanzdatei für die Vorschaudaten zu inkludieren. Sie können eine XML/XSD-Datei auch über die Symbolleisten-Schaltfläche  hinzufügen.

- Datenbank (*Professional und Enterprise Edition*)

  Fügt eine Datenbankkomponente hinzu. Siehe Datenbanken[165]. Sie können eine Datenbankkomponente auch über die Symbolleisten-Schaltfläche  hinzufügen. In der MapForce Enterprise Edition können auch NoSQL-Datenbanken als Komponenten hinzugefügt werden. Nähere Informationen dazu finden Sie unter NoSQL-Datenbanken[453].

- EDI (*Enterprise Edition*)

  Fügt eins EDI-Dokument hinzu. Siehe EDI[460]. Sie können eine EDI-Komponente auch über die Symbolleisten-Schaltfläche  hinzufügen.

- Textdatei (*Professional und Enterprise Edition*)

  Fügt ein Flat File-Dokument, wie z.B. eine CSV- oder Textdatei ("FLF" Fixed Length File) hinzu. Nähere Informationen dazu finden Sie unter CSV- und Textdateien[585]. Sie können eine Textdatei auch über die Symbolleisten-Schaltfläche  hinzufügen. In der MapForce Enterprise Edition können Textdateien mit Hilfe von FlexText verarbeitet werden. Siehe MapForce FlexText[684].

- Webservice-Funktion (*Enterprise Edition*)

  Fügt den Aufruf eines Webservice hinzu. Siehe Webservices[610]. Sie können einen Webservice auch über die Symbolleisten-Schaltfläche  hinzufügen.

- Excel 2007+-Datei (*Enterprise Edition*)

  Fügt eine Microsoft Excel 2007+ (**`.xlsx`**)-Datei hinzu. Siehe Microsoft OOXML Excel 2007+[738]. Wenn Sie Excel 2007+ nicht installiert haben, können Sie Dateien trotzdem von/auf Excel 2007+-Dateien mappen. In diesem Fall können Sie zwar keine Vorschau des Ergebnisses im Fenster **Ausgabe** anzeigen, das Ergebnis aber dennoch speichern. Sie können eine Excel-Datei auch über die Symbolleisten-Schaltfläche  hinzufügen.

- XBRL-Dokument (*Enterprise Edition*)

  Fügt eine XBRL-Instanz oder ein Taxonomiedokument hinzu. Nähere Informationen dazu finden Sie unter XBRL[769]. Sie können eine XBRL-Komponente auch über die Symbolleisten-Schaltfläche  hinzufügen.

- JSON-Schema/Datei (*Enterprise Edition*)

  Fügt ein JSON-Schema- oder eine JSON-Datei hinzu. Nähere Informationen dazu finden Sie unter JSON[823]. Sie können eine JSON-Komponente auch über die Symbolleisten-Schaltfläche  hinzufügen.

- Protocol Buffers-Datei (*Enterprise Edition*)

  Fügt eine im Protocol Buffers-Format kodierte Binärdatei hinzu. Nähere Informationen dazu finden Sie unter Protocol Buffer[843]. Sie können eine Datei im Protocol Buffers-Format auch über die Symbolleisten-Schaltfläche  hinzufügen.

- Input-Komponente einfügen

  Einfache Input-Komponenten können als für das gesamte Mapping relevante Input-Parameter oder nur im Kontext von benutzerdefinierten Funktionen verwendet werden. Nähere Informationen dazu finden Sie unter Einfache Input-Komponente[867] und Parameter in benutzerdefinierten Funktionen[973]. Sie können eine einfache Input-Komponente auch über die Symbolleisten-Schaltfläche  hinzufügen.

- Output-Komponente einfügen

  Einfache Output-Komponenten können als Output-Komponenten in Mappings und als Output-Parameter von benutzerdefinierten Funktionen verwendet werden. Nähere Informationen dazu finden Sie unter Einfache Output-Komponente[878] und Parameter in benutzerdefinierten Funktionen[973]. Sie können eine einfache Output-Komponente auch über die Symbolleisten-Schaltfläche  hinzufügen.

- Konstante

  Fügt eine Konstante ein, die Fixdaten für einen Input-Konnektor bereitstellt. Sie können die folgenden Datentypen wählen: `String`, `Zahl` und `alle anderen`. Sie können eine Konstante auch über die Symbolleisten-Schaltfläche  einfügen.

- Variable

  Fügt eine Variable[882] ein, die einer regulären (nicht-inline gesetzten) benutzerdefinierten Funktion entspricht. Bei Variablen handelt es sich um eine spezielle Art von Komponente, in der ein Mapping-Zwischenergebnis für die weitere Verarbeitung gespeichert wird. Sie können eine Variable auch über die Symbolleisten-Schaltfläche  einfügen.

- Join (*Professional und Enterprise Edition*)

  Mit Hilfe der Join-Komponente können Sie Daten im SQL- und Nicht-SQL-Modus verknüpfen. Sie können eine Join-Komponente auch über die Symbolleisten-Schaltfläche  hinzufügen. Nähere Informationen dazu finden Sie unter Verknüpfen von Daten mittels Join[896].

- Sortieren: Nodes/Zeilen

  Fügt eine Komponente ein, über die Sie Nodes sortieren können (siehe Sortieren von Daten[908]). Sie können eine Sortierkomponente auch über die Symbolleisten-Schaltfläche  hinzufügen.

© 2016-2022 Altova GmbH

- Filter: Nodes/Zeilen

  Fügt eine Filter-Komponente ein, mit der Daten aus jeder anderen von MapForce unterstützten Komponentenstruktur - auch Datenbanken - gefiltert werden können. Nähere Informationen finden Sie unter Filter und Bedingungen[914]. Sie können einen Filter auch über die Symbolleisten-Schaltfläche  hinzufügen.

- SQL/NoSQL-WHERE/ORDER (*Professional und Enterprise Edition*)

  Fügte eine Komponente ein, mit der Sie Datenbankdaten auf Basis von Bedingungen filtern können. Nähere Informationen dazu finden Sie unter SQL/NoSQL-WHERE/ORDER-Komponente[361]. Sie können die SQL/NoSQL-WHERE/ORDER-Komponente auch über die Symbolleisten-Schaltfläche  aufrufen.

- Wertezuordnung

  Fügt eine Komponente ein, die einen Input-Wert mittels einer Lookup-Tabelle in einen Output-Wert transformiert. Diese Funktion eignet sich dazu, eine Gruppe von Werten auf eine andere Gruppe von Werten zu mappen (z.B. Zahlen für einen Monat auf Monatsnamen). Nähere Informationen dazu finden Sie unter Wertezuordnungen[921]. Sie können eine Wertezuordnung auch über die Symbolleisten-Schaltfläche  einfügen.

- IF-Else-Bedingung

  Fügt eine If-Else-Bedingung ein. Dies eignet sich für Szenarien, in denen ein einfacher Wert anhand einer Bedingung verarbeitet werden soll. Nähere Informationen finden Sie unter Filter und Bedingungen[914]. Sie können eine If-Else-Bedingung auch über die Symbolleisten-Schaltfläche  hinzufügen.

- Ausnahme (*Professional und Enterprise Edition*)

  Mit Hilfe der Ausnahme-Komponente können Sie ein Mapping unterbrechen, wenn eine bestimmte Bedingung erfüllt wird. Sie können eine Ausnahmekomponente auch über die Symbolleisten-Schaltfläche  hinzufügen. In der MapForce Enterprise Edition können Sie mit Hilfe dieser Komponente auch Fehlermeldungen in WSDL-Mapping-Projekten definieren. Nähere Informationen finden Sie unter Definieren von SOAP-Webservice-Fehlern[1270]. Nähere Informationen zu Ausnahmekomponenten finden Sie unter Ausnahmeereignisse[938].

15.4 Projekt

MapForce gestattet Ihnen, Ihre Mappings in Mapping-Projekten[119] zu gruppieren. Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Projekt**.

- Projekt neu laden

  Lädt das aktuell aktive Projekt neu und wechselt zum Fenster **Projekt**.

- Projekt schließen

  Schließt das aktuell aktive Projekt.

- Projekt speichern

  Speichert das aktuell aktive Projekt.

- Dateien zu Projekt hinzufügen

  Damit können Sie Mappings zum aktuellen Projekt hinzufügen.

- Aktive Datei zu Projekt hinzufügen

  Fügt die aktuell aktive Datei zum aktuell geöffneten Projekt hinzu.

- Ordner erstellen

  Verwenden Sie diese Option, um einen neuen Ordner zum aktuellen Projekt hinzuzufügen. Siehe Projektordner[122].

- Mapping öffnen (*Enterprise Edition*)

  Öffnet das gerade im Fenster **Projekt** ausgewählte Mapping.

- Mapping für Operation erstellen (*Enterprise Edition*)

  Erstellt eine Mapping-Datei für die aktuell ausgewählte Operation des WSDL-Projekts. Nähere Informationen dazu finden Sie unter SOAP Webservices[1255].

- Mapping-Datei für Operation hinzufügen (*Enterprise Edition*)

  Mit dieser Option können Sie eine zuvor gespeicherte Mapping-Datei zur aktuell aktiven WSDL-Operation hinzufügen. Nähere Informationen dazu finden Sie unter SOAP Webservices[1255].

- WebService einfügen (*Enterprise Edition*)

  Dient zum Einfügen eines auf einer bestehenden WSDL-Datei basierenden Webservice. Nähere Informationen dazu finden Sie unter Webservices[610].

- in XMLSpy öffnen

  Öffnet die ausgewählte WSDL-Datei in Altova XMLSpy.

- Code für das gesamte Projekt generieren

© 2016-2022 Altova GmbH

Generiert Projektcode für das gesamte im **Projektfenster** angezeigte Projekt. Der Code wird für alle Mapping-Dateien (**.mfd**) in den einzelnen Ordnern in der aktuell ausgewählten Sprache generiert.

- Code generieren in

  Generiert Projektcode in der Sprache, die Sie aus dem Kontextmenü auswählen.

- Eigenschaften

  Ruft ein Dialogfeld auf, in dem Sie projektweite Einstellungen[122] definieren können.

- Letzte Projekte

  Zeigt eine Liste der zuletzt geöffneten Projekte an.

15.5 Komponente

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Komponente**.

- Root-Element ändern

  Dient zum Ändern des Root-Elements eines XML-Instanzdokuments.

- Schema-Definition in XMLSpy bearbeiten

  Um ein Schema in Altova XMLSpy bearbeiten zu können, müssen Sie auf eine XML-Komponente klicken und die Option **Schema-Definition in XMLSpy bearbeiten** auswählen.

- FlexText-Konfiguration bearbeiten (*Enterprise Edition*)

  Mit diesem Befehl können Sie eine FlexText-Datei bearbeiten. Nähere Informationen dazu finden Sie unter MapForce FlexText 684.

- Datenbankobjekte hinzufügen/entfernen/bearbeiten (*Professional und Enterprise Edition*)

  Dient zum Hinzufügen, Entfernen oder Ändern von Datenbankobjekten in einer Datenbankkomponente. Siehe Datenbanken 165.

- Mapping auf EDI X12 997 erstellen (*Enterprise Edition*)

  Das X12 997 Functional Acknowledgment gibt über den Status des EDI-Datenaustauschs Auskunft. Alle Fehler, die bei der Verarbeitung des Dokuments auftreten, werden hier ausgegeben. MapForce kann automatisch ein X12 997-Dokument generieren, das Sie an den Empfänger senden können. Nähere Informationen finden Sie unter Generieren eines X12 997 Functional Acknowledgment 481.

- Mapping auf EDI X12 999 erstellen (*Enterprise Edition*)

  Das X12 999 Implementation Acknowledgment Transaction Set meldet die Nichterfüllung von HIPAA-Implementierungsrichtlinien oder Applikationsfehler. MapForce kann automatisch eine X12 999-Komponente generieren und die erforderlichen Mapping-Verbindungen erstellen. Nähere Informationen dazu finden Sie unter Generieren eines X12 999 Implementation Acknowledgment 483.

- Aktualisieren (*Professional und Enterprise Edition*)

  Lädt die Struktur der gerade aktiven Datenbankkomponente neu.

- Duplikat davor/danach einfügen

  Fügt eine Kopie des ausgewählten Datenelements vor/nach dem aktuell ausgewählten ein. Ein duplizierter Input kann nicht als Datenquelle verwendet werden. Nähere Informationen dazu finden Sie unter Input duplizieren 84.

- Duplikat löschen

  Entfernt ein dupliziertes Datenelement.

- Kommentar/Processing Instruction

© 2016-2022 Altova GmbH

Mit dieser Option können Sie Kommentare und Processing Instructions[136] in XML-Komponenten einfügen.

- Inhalt als CDATA-Abschnitt schreiben

  Mit diesem Befehl wird ein CDATA-Abschnitt[137] erstellt. CDATA-Abschnitte dienen dazu, Abschnitte eines Dokuments, die normalerweise als Markup interpretiert würden, als Zeichendaten darzustellen.

- Datenbankaktionen (*Professional und Enterprise Edition*)

  Dient zum Konfigurieren von Datenbankeinfüge-, -aktualisierungs- und -löschaktionen und anderer Optionen für Datenbankdatensätze. Nähere Informationen dazu finden Sie unter Einstellungen für Datenbankaktionen[336].

- Datenbank abfragen (*Professional und Enterprise Edition*)

  Erstellt anhand der Tabelle bzw. des Felds, die bzw. das Sie in der Datenbankkomponente ausgewählt haben, eine SELECT -Anweisung. Wenn Sie auf eine Tabelle/ein Feld klicken, wird dieser Befehl aktiv und die SELECT -Anweisung wird automatisch in das **Select**-Fenster platziert.

- Links ausrichten

  Richtet die Struktur einer Komponenten linksbündig aus.

- Rechts ausrichten

  Richtet die Struktur einer Komponenten rechtsbündig aus.

- Eigenschaften

  Zeigt die Eigenschaften der aktuell ausgewählten Komponente an. Siehe auch Ändern der Komponenteneinstellungen[82].

15.6 Verbindung

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Verbindung**.

- Idente Sub-Einträge automatisch verbinden

  Aktiviert bzw. deaktiviert die Option **Idente Sub-Einträge automatisch verbinden**. Nähere Informationen zu Verbindungen und Verbindungsarten finden Sie unter Verbindungen [89].

- Einstellungen für 'Idente Sub-Einträge verbinden'

  Damit können Sie idente Sub-Einträge definieren. Nähere Informationen dazu finden Sie unter Verbindungen mit identen Sub-Einträgen [96].

- Idente Sub-Einträge verbinden

  Mit diesem Befehl können Sie sowohl in Quell- als auch im Zielkomponenten mehrere Verbindungen für Datenelemente **desselben Namens** erzeugen. Die Einstellungen in diesem Dialogfeld werden angewendet, wenn die Symbolleisten-Schaltfläche  (**Idente Sub-Einträge automatisch verbinden**) aktiv ist. Nähere Informationen dazu finden Sie unter Verbindungen mit identen Sub-Einträgen [96].

- Zielorientiert (Standard)

  Ändert den Konnektortyp in ein Standard-Mapping. Nähere Informationen dazu finden Sie unter Zielorientierte im Vergleich zu quellorientierten Verbindungen [93].

- Alles kopieren (Sub-Einträge kopieren)

  Erstellt Verbindungen für alle identen Subeinträge. Der Hauptvorteil von "Alles kopieren"-Verbindungen ist, dass der Mapping-Arbeitsbereich dadurch übersichtlicher wird: Anstelle mehrerer Verbindungen wird eine einzige durch eine dicke Linie dargestellte Verbindung erstellt. Nähere Informationen dazu finden Sie unter "Alles kopieren"-Verbindungen [97].

- Quellorientiert (Mixed Content)

  Ändert den Verbindungstyp in eine quellorientierte Verbindung. Dadurch kann gemischter Inhalt (Mixed Content) (Text- und Child-Nodes) automatisch in derselben Reihenfolge, wie er in der XML-*Quelldatei* vorkommt, gemappt werden. Nähere Informationen dazu finden Sie unter Quellorientierte Verbindungen [93] .

- Eigenschaften

  Öffnet das Dialogfeld **Verbindungseinstellungen**, in dem Sie Verbindungsarten und Annotationseinstellungen definieren können. Nähere Informationen dazu finden Sie unter Verbindungseinstellungen [99].

© 2016-2022 Altova GmbH

15.7 Funktion

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Funktion**.

- Benutzerdefinierte Funktion erstellen

  Erstellt eine neue benutzerdefinierte Funktion[967] (UDF). Sie können eine benutzerdefinierte Funktion auch über die Symbolleisten-Schaltfläche erstellen.

- Benutzerdefinierte Funktion von Auswahl erstellen

  Erstellt auf Basis der aktuell im Mapping-Fenster ausgewählten Elemente eine benutzerdefinierte Funktion. Nähere Informationen dazu finden Sie unter Erstellen benutzerdefinierter Funktionen[970]. Sie können eine benutzerdefinierte Funktion anhand der Auswahl auch über die Symbolleisten-Schaltfläche erstellen.

- Funktionseinstellungen

  Öffnet das Dialogfeld **Benutzerdefinierte Funktion bearbeiten**, wo Sie die aktuellen Einstellungen für die benutzerdefinierte Funktion ändern können. Nähere Informationen dazu finden Sie unter Bearbeiten benutzerdefinierter Funktionen[972].

- Funktion entfernen

  Löscht die aktuell aktive benutzerdefinierte Funktion, wenn Sie in einem Kontext arbeiten, der dies erlaubt.

- Input-Komponente einfügen

  Einfache Input-Komponenten können als für das gesamte Mapping relevante Input-Parameter oder nur im Kontext von benutzerdefinierten Funktionen verwendet werden. Nähere Informationen dazu finden Sie unter Einfache Input-Komponente[867] und Parameter in benutzerdefinierten Funktionen[973]. Sie können eine einfache Input-Komponente auch über die Symbolleisten-Schaltfläche hinzufügen.

- Output-Komponente einfügen

  Einfache Output-Komponenten können als Output-Komponenten in Mappings und als Output-Parameter von benutzerdefinierten Funktionen verwendet werden. Nähere Informationen dazu finden Sie unter Einfache Output-Komponente[878] und Parameter in benutzerdefinierten Funktionen[973]. Sie können eine einfache Output-Komponente auch über die Symbolleisten-Schaltfläche hinzufügen.

15.8 Ausgabe

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Ausgabe**.

- XSLT 1.0/XSLT 2.0/XSLT 3.0/XQuery/Java/C#/C++/Built-In

  Dient zum Definieren der Transformationssprache, in der das Mapping ausgeführt werden soll. Welche Transformationssprachen zur Auswahl stehen, hängt von Ihrer MapForce Edition ab. Nähere Informationen dazu finden Sie unter Transformationssprachen 28. Die Transformationssprache kann auch über die Symbolleiste ausgewählt werden.

- Ausgabedatei validieren

  Validiert die XML-Ausgabedatei anhand des referenzierten Schemas. Siehe Validierung 107.

- Ausgabedatei speichern

  Speichert die aktuell im Fenster **Ausgabe** angezeigten Daten in einer Datei.

- Alle Ausgabedateien speichern

  Speichert alle von dynamischen Mappings 1371 generierten Ausgabedateien. Siehe auch Tutorial 4 68.

- Ausgabedatei neu generieren

  Lädt die Daten im Fenster **Ausgabe** neu.

- SQL/NoSQL-Script ausführen (*Professional und Enterprise Edition*)

  Wenn im Fenster **Ausgabe** gerade ein SQL/NoSQL-Script angezeigt wird, wird das Mapping auf die Zieldatenbank unter Berücksichtigung der definierten Tabellenaktionen ausgeführt. Nähere Informationen über Datenbanken finden Sie unter Datenbanken 165.

- Lesezeichen einfügen/löschen

  Fügt im Fenster **Ausgabe** an der Cursorposition ein Lesezeichen ein bzw. löscht dieses.

- Nächstes/Vorhergehendes Lesezeichen

  Navigiert im Fenster **Ausgabe** zum nächsten/vorhergehenden Lesezeichen.

- Alle Lesezeichen löschen

  Entfernt im Fenster **Ausgabe** alle derzeit definierten Lesezeichen.

- Pretty-Print

  Formatiert Ihr XML-Dokument im Fenster **Ausgabe** neu, sodass Sie eine strukturierte Übersicht über das Dokument haben. Jedes Subelement ist einen Tabstopp vom übergeordneten Element eingerückt. Die im Dialogfeld Einstellungen für Textansicht 110 definierten Tabulatoreinstellungen (Gruppe "Tabulatoren") werden im Fenster **Ausgabe** angewendet.

- Einstellungen für Textansicht

© 2016-2022 Altova GmbH

Ruft das Dialogfeld **Einstellungen für Textansicht** auf. In diesem Dialogfeld können Sie die Textansichtseinstellungen in den Fenstern **XQuery** (*Professional und Enterprise Edition*), **Ausgabe** und **XSLT** anpassen. Außerdem sehen Sie darin die aktuell für das Fenster geltenden Tastaturkürzel. Nähere Informationen dazu finden Sie unter Funktionalitäten der Textansicht[110].

15.9 Debuggen

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Debuggen**.

- Debugger starten

  Startet das Debugging bzw. setzt das Debugging fort, bis ein Breakpoint oder das Ende des Mappings erreicht wird. Sie können das Debugging auch über die Symbolleisten-Schaltfläche starten. Nähere Informationen zum Debug-Modus finden Sie unter Debugger[1409].

- Debugger anhalten

  Stoppt das Debuggen. Mit diesem Befehl wird der Debug-Modus beendet und MapForce wird wieder im Standardmodus angezeigt. Sie können das Debugging auch über die Symbolleisten-Schaltfläche beenden. Nähere Informationen zum Debug-Modus finden Sie unter Debugger[1409].

- Einsteigen

  Führt das Mapping aus, bis ein einzelner Schritt an irgendeiner Stelle im Mapping fertig gestellt ist. Ein Schritt im Mapping Debugger ist eine logische Gruppe von voneinander abhängigen Berechnungen, deren Resultat normalerweise ein einzelnes Datenelement einer Sequenz ist. Je nach Mapping-Kontext bedeutet dieser Befehl in etwa: *gehe nach links/gehe zum Ziel-Child-Datenelement/gehe zum Parent in der Quellkomponente*. Sie können diesen Befehl auch über die Symbolleisten-Schaltfläche aufrufen. Nähere Informationen zum Debug-Modus finden Sie unter Debugger[1409].

- Überspringen

  Setzt die Ausführung fort, bis der aktuelle Schritt beendet ist (oder für ein anderes Datenelement der Sequenz beendet ist) oder bis ein nicht damit in Zusammenhang stehender Schritt fertig gestellt ist. Dieser Befehl überspringt Berechnungen, die Inputs des aktuellen Schritts sind. Sie können diesen Befehl auch über die Symbolleisten-Schaltfläche aufrufen. Nähere Informationen zum Debug-Modus finden Sie unter Debugger[1409].

- Aussteigen

  Setzt die Ausführung fort, bis das Ergebnis des aktuellen Schritts verwendet wird oder bis ein Schritt ausgeführt wird, der kein Input oder Child des verwendeten Werts ist. Dieser Befehl steigt aus der aktuellen Berechnung aus. Je nach Mapping-Kontext bedeutet dieser Befehl in etwa: *gehe nach rechts/gehe zum Parent-Zieldatenelement//gehe zum Child-Quelldatenelement*. Sie können diesen Befehl auch über die Symbolleisten-Schaltfläche aufrufen. Nähere Informationen zum Debug-Modus finden Sie unter Debugger[1409].

- Minimaler Schritt

  Fährt mit der Ausführung fort, bis ein Wert erzeugt oder verwendet wird. Normalerweise hält der Debugger bei jeder Verbindung zwei Mal an: (i) einmal, wenn die Quelle einen Wert erzeugt und (ii) einmal, wenn er vom Ziel verwendet wird. MapForce berechnet Werte nicht unbedingt in der Reihenfolge, die das Mapping vorgibt, daher folgen die Erzeugung und Verwendung nicht immer unmittelbar aufeinander. Sie können diesen Befehl auch über die Symbolleisten-Schaltfläche aufrufen. Nähere Informationen zum Debug-Modus finden Sie unter Debugger[1409].

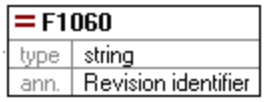
© 2016-2022 Altova GmbH

15.10 Ansicht

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Ansicht**.

- Annotationen anzeigen

  Zeigt Annotationen in der Komponente an. Sie können diese Option auch über die Symbolleisten-Schaltfläche  aktivieren. Wenn auch das Symbol **Datentypen anzeigen** aktiv ist, werden beide Informationen in Tabellenform angezeigt (*siehe Abbildung unten*). Sie können Verbindungen auch mit Hilfe von Annotationen beschriften. Nähere Informationen dazu finden Sie unter Verbindungseinstellungen 101.

- Datentypen anzeigen

  Zeigt die Datentypen in einer Komponente an. Sie können diese Option auch über die Symbolleisten-Schaltfläche  aktivieren. Wenn auch das Symbol **Datentypen anzeigen** aktiv ist, werden beide Informationen in Tabellenform angezeigt (*siehe "Annotationen anzeigen" oben*).

- Bibliothek in Funktionstitelleiste anzeigen

  Zeigt den Namen der Bibliothek in der Funktionsüberschrift an. Sie können diese Option auch über die Symbolleisten-Schaltfläche  aktivieren.

- Tipps anzeigen

  Zeigt einen Tooltip mit erklärendem Text an, wenn Sie den Mauszeiger über eine Funktionsüberschrift platzieren. Wenn die Option **Tipps anzeigen** aktiv ist, werden auch Informationen über Datentypen in einer Komponente angezeigt.

- XBRL-Anzeigeoptionen (*Enterprise Edition*)

  Sie können in MapForce die folgenden XBRL-Einstellungen konfigurieren:

  - die Label-Sprache für die XBRL Items und ihre Annotationen
  - die bevorzugten Label-Roles für XBRL Item-Namen
  - den spezifischen Typ von Label-Roles von Annotationen für XBRL Items
  - Benutzerdefinierte XBRL-Taxonomiepakete

  Nähere Informationen dazu finden Sie unter XBRL-Optionen 1543.

- Ausgewählte Komponentenkonnektoren anzeigen/Quell- und Zielkonnektoren anzeigen

  Mit Hilfe dieser Optionen können Verbindungen selektiv hervorgehoben werden. Eine Erläuterung dazu finden Sie unter Verbindungen 91.

- Vergrößern/Verkleinern

Öffnet das Dialogfeld **Vergrößern/Verkleinern**, Sie können entweder einen numerischen Zoom-Faktor eingeben oder den Zoom-Faktor mit Hilfe des Schiebereglers interaktiv einstellen.

- Zurück/Vorwärts

  Mit den Befehlen **Zurück** und **Vorwärts** können Sie zum vorherigen und nächsten Mapping, an dem Sie relativ zu aktuell geöffneten Mapping gearbeitet haben, wechseln.

- Statusleiste

  Blendet die unterhalb des Fensters **Meldungen** angezeigte **Statusleiste** ein oder aus.

- Bibliotheken/Bibliotheken verwalten

  Klicken Sie auf **Bibliotheken**, um das **Bibliotheksfenster** ein- bzw. auszublenden. Klicken Sie auf **Bibliotheken verwalten**, um das Fenster **Bibliotheken verwalten** ein- oder auszublenden.

- Meldungen

  Blendet das Fenster "Meldungen" 38 ein oder aus. Bei Generierung von Code wird das Fenster **Meldungen** automatisch aktiviert, um das Ergebnis der Validierung anzuzeigen.

- Übersicht

  Blendet das Übersichtsfenster 37 ein oder aus. Ziehen Sie das Rechteck mit der Maus, um in Ihrem Mapping zu navigieren.

- Projektfenster (*Professional und Enterprise Edition*)

  Blendet das **Projektfenster** ein oder aus. Nähere Informationen zu Projekten finden Sie unter Projekte 119 .

- Debug-Fenster (*Professional und Enterprise Edition*)

  Im Debug-Modus können Sie den Kontext, in dem ein bestimmter Wert erzeugt wird, analysieren. Diese Informationen stehen direkt im Mapping und in den Fenstern **Werte**, **Kontext** und **Breakpoints** zur Verfügung. Nähere Informationen dazu finden Sie unter Informationen zum Debug-Modus 1414 .

© 2016-2022 Altova GmbH

15.11 Extras

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Extras**.

- Globale Ressourcen

  Öffnet das Dialogfeld **Globale Ressourcen verwalten**, in dem Sie Einstellungen, die für mehrere Altova-Applikationen verwendet werden können, hinzufügen, bearbeiten und löschen können (siehe Globale Altova-Ressourcen[1462]).

- Aktive Konfiguration

  Dient zum Auswählen der aktuell aktiven globalen Ressourcenkonfiguration aus einer Liste von Konfigurationen. Informationen zum Erstellen und Konfigurieren verschiedener Arten von globalen Ressourcen finden Sie unter Globale Altova-Ressourcen[1462].

- Umgekehrtes Mapping erstellen

  Erstellt anhand des gerade in MapForce aktiven Mappings ein "umgekehrtes" Mapping, d.h. die Quellkomponente wird zur Zielkomponente und die Zielkomponente zur Quellkomponente. Beachten Sie, dass im umgekehrten Mapping nur direkte Verbindungen zwischen Komponenten beibehalten werden. Das neue Mapping wird wahrscheinlich nicht gültig sein und kann wahrscheinlich nicht im Fenster **Ausgabe** angezeigt werden. Das neue Mapping müsste vorher manuell bearbeitet werden.

  Die folgenden Daten bleiben erhalten:

  - direkte Verbindungen zwischen Komponenten
  - direkte Verbindungen zwischen Komponenten in einem verketteten Mapping
  - die Art der Verbindung[93]: Standard, gemischter Inhalt, Alles kopieren
  - Einstellungen für die Weiterleitung einer Komponente
  - Datenbankkomponenten (*Professional und Enterprise Edition*)

  Die folgenden Daten bleiben *nicht* erhalten:

  - Verbindungen über Funktionen, Filter, usw.
  - Benutzerdefinierte Funktionen
  - Webservice-Komponenten (*Enterprise Edition*)

- XBRL-Taxonomie-Manager (*Enterprise Edition*)

  Der XBRL-Taxonomie-Manager ist ein Tool, mit dem Sie XBRL-Taxonomien installieren und verwalten können. Nähere Informationen dazu finden Sie unter XBRL-Taxonomie-Manager[777].

- XML-Schema-Manager

  Der XML-Schema-Manager ist ein Altova-Tool, mit dem Sie XML-Schemas (DTDs für XML-Dateien und XML-Schemas) zentral installieren und verwalten können, um diese in allen XML-fähigen Applikationen von Altova verwenden zu können. Nähere Informationen dazu finden Sie unter Schema-Manager[149].

- Anpassen

  Über diese Option können Sie die grafische Benutzeroberfläche von MapForce anpassen. So können Sie hier etwa Symbolleisten ein- und ausblenden und die Menüs[1536] und Tastaturkürzel[1537] bearbeiten.

- Symbolleisten und Fenster wiederherstellen

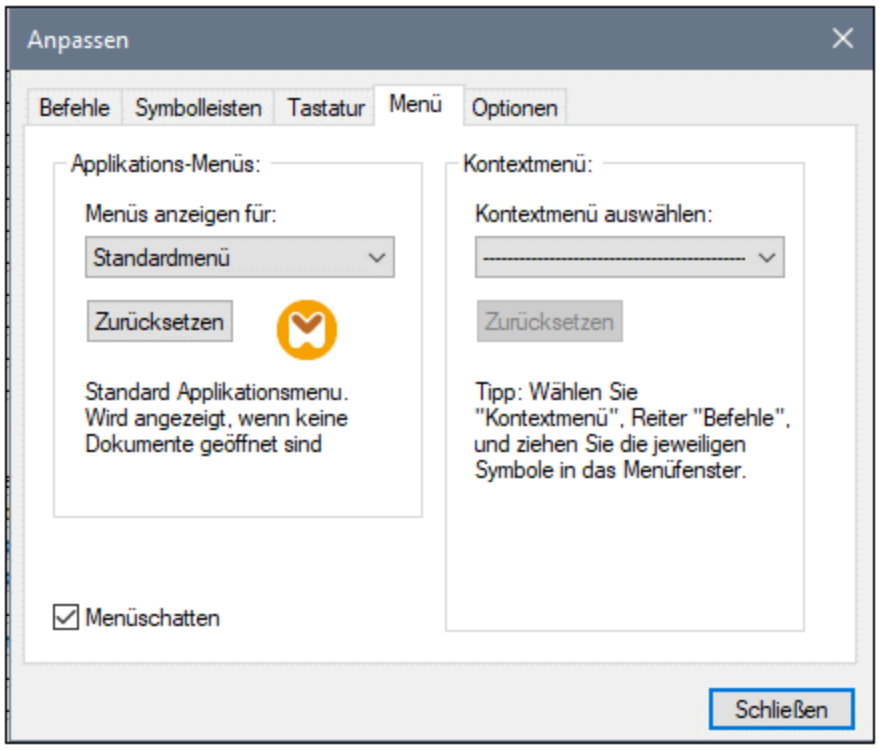
  Setzt die Symbolleisten, Eingabehilfefenster, angedockten Fenster usw. wieder auf ihre Standardeinstellung zurück. MapForce muss neu gestartet werden, damit die Änderungen wirksam werden.

- Optionen

  Ruft das Dialogfeld **Optionen** auf, in dem Sie die Standardeinstellungen von MapForce ändern können. Nähere Informationen dazu finden Sie unter Optionen[1539].

15.11.1 Anpassen von Menüs

Sie können MapForce-Standardmenüs und Kontextmenüs anpassen (um z.B. Befehle hinzufügen, ändern oder entfernen). Sie können Ihre Änderungen auch in den Standardzustand zurücksetzen (**Zurücksetzen**). Um Menüs anzupassen, klicken Sie auf **Extras | Anpassen** und anschließend auf das Register **Menü** (*siehe Abbildung unten*).



Standardmenü vs. MapForce Design

Das *Standardmenü* ist die Menüleiste, die angezeigt wird, wenn kein Dokument im Hauptfenster geöffnet ist. Das *MapForce Design*-Menü ist die Menüleiste, die angezeigt wird, wenn ein oder mehrere Mappings geöffnet sind. Jede der beiden Menüleisten kann separat angepasst werden. Änderungen, die an einer Menüleiste vorgenommen wurden, haben keine Auswirkung auf die andere Menüleiste.

© 2016-2022 Altova GmbH

Um eine Menüleiste anzupassen, wählen Sie diese in der Auswahlliste *Menüs anzeigen für* aus. Klicken Sie anschließend auf das Register **Befehle** und ziehen Sie die Befehle aus dem Listenfeld *Befehle* in die Menüleiste oder in eines der Menüs.

Löschen von Befehlen aus Menüs
So löschen Sie ein ganzes Menü oder einen Befehl in einem Menü:

1. Wählen Sie aus der Dropdown-Liste *Menüs anzeigen für Standardmenü* oder *MapForce* Design aus.
2. Während das Dialogfeld **Anpassen** geöffnet ist, wählen Sie einen Symbolleisten-Befehl aus, den Sie löschen möchten oder wählen Sie einen Befehl aus, den Sie aus einem der Menüs löschen möchten.
3. Ziehen Sie den Symbolleisten-Befehl mit der Maus aus der Symbolleiste bzw. aus dem Menü. Klicken Sie alternativ dazu mit der rechten Maustaste auf den Symbolleisten-Befehl oder den Menübefehl und wählen Sie **Löschen** aus.

Sie können jede Menüleiste in den Standardzustand zurücksetzen. Wählen Sie sie dazu aus der Dropdown-Liste *Menüs anzeigen für* aus und klicken Sie anschließend auf die Schaltfläche **Zurücksetzen**.

Anpassen von Kontextmenüs
Kontextmenüs sind die Menüs, die angezeigt werden, wenn Sie mit der rechten Maustaste auf bestimmte Objekte auf der Benutzeroberfläche der Applikation klicken. Jedes dieser Kontextmenüs kann folgendermaßen angepasst werden:

1. Wählen Sie das Kontextmenü in der Dropdown-Liste *Kontextmenü auswählen* aus. Daraufhin wird das entsprechende Kontextmenü geöffnet.
2. Klicken Sie auf das Register **Befehle** und ziehen Sie den gewünschten Befehl aus dem Listenfeld *Befehle* in das Kontextmenü.
3. Um einen Befehl aus dem Kontextmenü zu löschen, klicken Sie mit der rechten Maustaste darauf und wählen Sie den Befehl **Löschen**. Ziehen Sie den Befehl alternativ dazu mit der Maus aus dem Kontextmenü heraus.

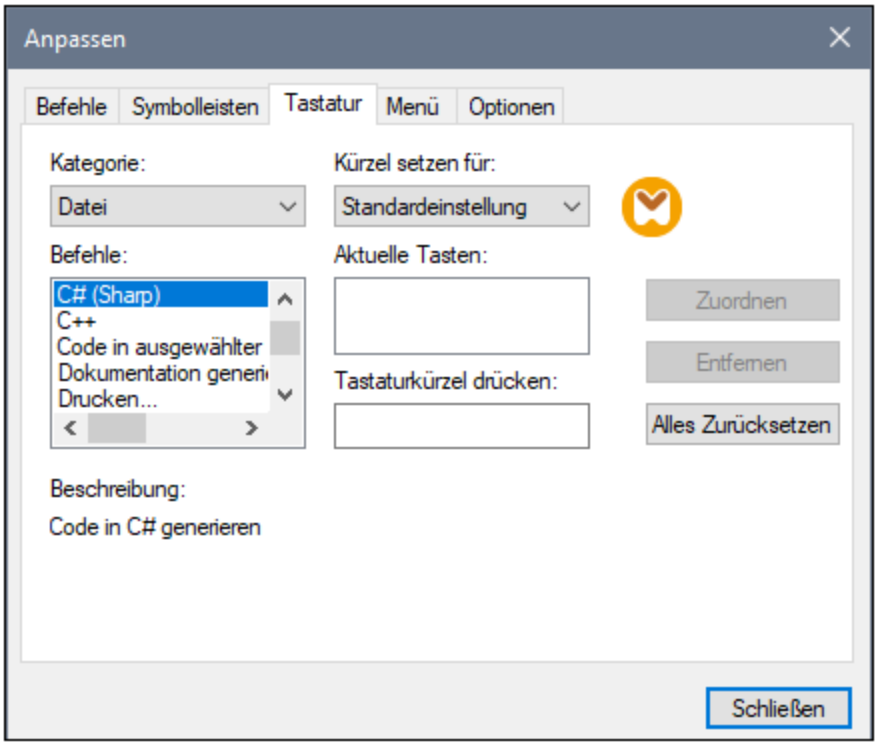
Sie können jedes Kontextmenü in den Standardzustand zurücksetzen. Wählen Sie es dazu aus der Dropdown-Liste *Kontextmenü auswählen* aus und klicken Sie anschließend auf die Schaltfläche **Zurücksetzen**.

Menüschatten
Aktivieren Sie das Kontrollkästchen *Menüschatten*, wenn Menüs mit Schatten dargestellt werden sollen.

15.11.2 Anpassen von Tastaturkürzeln

Sie können Tastaturkürzel in MapForce folgendermaßen definieren oder ändern: Wählen Sie den Befehl **Extras | Anpassen** und klicken Sie auf das Register **Tastatur**. Um einem Befehl ein neues Tastaturkürzel zuzuweisen, gehen Sie folgendermaßen vor:

1. Wählen Sie den Befehl **Extras | Anpassen** und klicken Sie auf das Register **Tastatur** (*siehe Abbildung unten*).
2. Klicken Sie auf die Auswahlliste *Kategorie*, um den Menünamen auszuwählen.
3. Wählen Sie in der Liste *Befehle* den Befehl aus, dem Sie ein neues Tastenkürzel zuweisen möchten.
4. Geben Sie in das Textfeld *Tastaturkürzel drücken* die neuen Tastaturkürzel ein und klicken Sie auf **Zuordnen**.



Um den Eintrag im Textfeld *Tastaturkürzel drücken* zu löschen, drücken Sie eine der Steuerungstasten **Strg**, **Alt** oder **Umschalt**. Um eine Tastenzuweisung zu löschen, klicken Sie unter *Aktuelle Tasten* auf das gewünschte Tastaturkürzel und anschließend auf **Entfernen**.

Anmerkung: Die Option *Kürzel setzen für* hat derzeit keine Funktion.

Tastaturkürzel

MapForce bietet standardmäßig die folgenden Tastaturkürzel:

F1	Hilfe
F2	Nächstes Lesezeichen (im Fenster "Ausgabe")
F3	Weitersuchen
F10	Menüleiste aktivieren
Num +	Aktuellen Node erweitern
Num -	Node reduzieren
Num *	Alle unterhalb des aktuellen Node erweitern
STRG + TAB	Wechselt zwischen offenen Mappings
STRG + F6	Wechselt zwischen offenen Fenstern
STRG + F4	Schließt das aktive Mapping-Dokument
Alt + F4	Schließt MapForce
Alt + F, F, 1	Öffnet die letzte Datei
Alt + F, T, 1	Öffnet das letzte Projekt

© 2016-2022 Altova GmbH

STRG + N	Datei neu
STRG + O	Datei öffnen
STRG + S	Datei speichern
STRG + P	Datei drucken
STRG + A	Alles markieren
STRG + X	Ausschneiden
Strg + C	Kopieren
STRG + V	Einfügen
STRG + Z	Rückgängig machen
STRG + Y	Wiederherstellen
Entf	Komponente löschen (mit Betätigungsmeldung)
Umschalt + Entf	Komponente löschen (ohne Bestätigungsmeldung)
STRG + F	Suchen
F3	Weitersuchen
Umschalt+F3	Vorheriges suchen
Pfeiltasten (nach oben / nach unten)	Nächstes Datenelement der Komponente auswählen
Esc	Bearbeitungen verwerfen/Dialogfeld schließen
Rückgabewert	Bestätigt eine Auswahl
Tastaturkürzel im Fenster "Ausgabe"	
STRG + F2	Lesezeichen einfügen/löschen
F2	Nächstes Lesezeichen
Umschalt + F2	Vorheriges Lesezeichen
Strg + Umschalt + F2	Alle Lesezeichen löschen
Zoom-Tasten	
Strg + Mausrad vorwärts	Vergrößern
Strg + Mausrad rückwärts	Verkleinern
Strg + 0 (Null)	Zoom zurücksetzen

15.11.3 Optionen

Sie können die allgemeinen Einstellungen und anderen Einstellungen in MapForce mit dem Befehl **Extras | Optionen** ändern. Weiter unten finden Sie eine Beschreibung der verfügbaren Optionen.

- Allgemein

  Im Abschnitt *Allgemein* können Sie die folgenden Optionen definieren:

  - *Logo anzeigen | Beim Start anzeigen:* Hier können Sie definieren, ob ein Bild (Willkommensbildschirm) beim Start von MapForce angezeigt werden soll oder nicht.

- *Mapping-Ansicht.* Sie können die Anzeige des schattierten Hintergrunds im Fenster **Mapping** aktivieren/deaktivieren (*Schattierten Hintergrund anzeigen*). Außerdem können Sie die Anzeige von Annotationen auf Wunsch auf N Zeilen begrenzen. Wenn der Annotationstext aus mehreren Zeilen besteht, so werden bei Aktivierung dieser Option in der Komponente nur die ersten N Zeilen angezeigt, wobei N der von Ihnen definierte Wert ist. Diese Einstellung gilt auch für SELECT-Anweisungen, die in einer Komponente angezeigt werden.

- *Standardkodierung für neue Komponenten.*

  Kodierungsname: Die Standardkodierung für neue XML-Dateien kann durch Auswahl einer Option aus der Dropdown-Liste festgelegt werden. Wird eine 2- oder 4-Byte-Kodierung als standardmäßige Kodierung ausgewählt (z.B. UTF-16, UCS-2 oder UCS-4) können Sie weiters wählen, ob die Byteordnung für XML-Dokumente Little Endian oder Big Endian sein soll. Diese Einstellung kann auch für jede Komponente einzeln geändert werden (siehe Ändern der Komponenteneinstellungen 82).

  Bytefolge: Wenn ein Dokument mit einer 2-Byte- oder 4-Byte-Zeichenkodierung gespeichert wird, kann es entweder mit der Byteordnung Little-Endian oder Big-Endian gespeichert werden. Außerdem kann festgelegt werden, ob eine Bytefolge-Markierung inkludiert werden soll.

- *Einstellungen Vorschau:* Mit der Option *Timeout benutzen* wird ein Ausführungs-Timeout für die Vorschau auf das Mapping-Ergebnis im Fenster **Ausgabe** definiert.

- *Bei Aktivieren des Ausgaberegisters:* Sie können die Ausgabe in temporären Dateien generieren oder sie direkt in eine Ausgabedatei schreiben (*siehe unten*).

  Output-Datei als temporäre Datei generieren: Dies ist die Standardoption. Wenn der Ausgabepfad Ordner enthält, die noch nicht vorhanden sind, erstellt MapForce diese. Für die Professional und Enterprise Edition: Falls Sie beabsichtigen, das Mapping für die Ausführung auf einem Server bereitzustellen, müssen die Verzeichnisse im Pfad auf dem Server vorhanden sein, da bei der Ausführung sonst ein Fehler auftritt. Siehe auch Vorbereiten von Mappings für die Server-Ausführung 1435.

  Direkt in die endgültigen Output-Dateien schreiben Wenn der Ausgabepfad Ordner enthält, die noch nicht vorhanden sind, kommt es zu einem Fehler. Mit dieser Option werden vorhandene Ausgabedateien ohne vorherige Bestätigung überschrieben.

- *Text in Schritten von N Millionen Zeichen anzeigen:* Definiert die maximale Größe von Text, der im Fenster **Ausgabe** angezeigt wird, wenn eine Vorschau auf Mappings, mit denen große XML- und Textdateien generiert werden, angezeigt wird. Wenn der Ausgabetext diesen Wert übersteigt, müssen Sie auf die Schaltfläche **Mehr laden** klicken, um den nächsten Block zu laden. Nähere Informationen dazu finden Sie unter Vorschau und Validieren der Ausgabe 107.

Bearbeiten

Im Abschnitt *Bearbeiten* können Sie Optionen für die Anzeige von Mappings definieren:

- *Komponenten beim Ziehen mit der Maus aneinander ausrichten:* Hier wird definiert, ob Komponenten oder Funktionen im Mapping-Fenster beim Ziehen mit der Maus an anderen Komponenten ausgerichtet werden sollen (siehe Ausrichten von Komponenten 83).

 © 2016-2022 Altova GmbH

- *Intelligente Komponentenlöschung:* Sie können Verbindungen in MapForce auch nach Löschung einiger Transformationskomponenten[78] beibehalten. Vor allem die Beibehaltung von Verbindungen mit mehreren Child-Verbindungen kann sich als nützlich erweisen, da Sie dadurch nach Löschung einer Transformationskomponente nicht jede einzelne Child-Verbindung manuell wiederherstellen müssen. Nähere Informationen dazu finden Sie unter Beibehalten von Konnektoren nach Löschen von Komponenten[105].

Meldungen

Im Abschnitt *Meldungen* können Sie die Anzeige von Meldungen wie z.B. den Vorschlag übergeordnete Datenelemente zu verbinden, eine Meldung über die Erstellungen mehrerer Zielkomponenten anzuzeigen, usw. aktivieren.

Code-Generierung (*Professional und Enterprise Edition*)

Im Abschnitt *Code-Generierung* können Sie die Einstellungen für die Generierung von Programmcode und MapForce Server-Ausführungsdateien definieren. Nähere Informationen dazu finden Sie unter Codegenerierungsoptionen[1701] und Kompilieren von Mappings zu MapForce Server-Ausführungsdateien[1442] .

Java

Wenn Sie eine Java Virtual Machine verwenden, die keinen Installer hat und keine Registry-Einträge erstellt (z.B. OpenJDK von Oracle), müssen Sie eventuell einen benutzerdefinierten Java VM-Pfad angeben. Auch wenn Sie automatisch von MapForce ermittelte Java VM-Pfade außer Kraft setzen müssen, müssen Sie diesen Pfad eventuell definieren. Nähere Informationen dazu finden Sie unter Java[1542].

XBRL (*Enterprise Edition*)

Sie können in MapForce die folgenden allgemeinen applikationsweiten XBRL-Einstellungen konfigurieren:

- die Label-Sprache für die XBRL Items und ihre Annotationen
- die bevorzugten Label-Roles für XBRL Item-Namen
- den spezifischen Typ von Label-Roles von Annotationen für XBRL Items
- Benutzerdefinierte XBRL-Taxonomiepakete (siehe XBRL-Taxonomiepakete[1545])

Nähere Informationen dazu finden Sie unter Allgemeine XBRL-Optionen[1543].

Debugger (*Professional und Enterprise Edition*)

Im Abschnitt *Debugger* können Sie die folgenden Debugeinstellungen definieren:

- *Maximale Länge gespeicherter Werte:* Definiert die String-Länge von Werten, die im Fenster **Werte** angezeigt werden (mindestens 15 Zeichen). Beachten Sie, dass es zu Arbeitsspeicherproblemen kommen kann, wenn Sie die Länge der gespeicherten Werte auf einen hohen Wert setzen.

- *Vollständigen Verlauf der Ablaufverfolgung behalten:* Wenn Sie diese Option aktivieren, speichert MapForce während des Debuggens den Verlauf aller Werte, die von allen Konnektoren aller Komponenten im Mapping verarbeitet wurden. In diesem Fall werden alle von Beginn des Debuggens an verarbeiteten Werte im Arbeitsspeicher gespeichert, so dass diese für Ihre Analyse im Fenster **Werte** zur Verfügung stehen, bis Sie den Debug-Vorgang beenden. Es ist

nicht empfehlenswert, diese Option beim Debuggen von Mappings mit vielen Daten zu aktivieren, da dies den Debug-Vorgang verlangsamen und zu viel Arbeitsspeicher beanspruchen könnte. Wenn diese Option deaktiviert ist, speichert MapForce nur den jüngsten Debug-Verlauf für Nodes, die mit der aktuellen Ausführungsposition in Zusammenhang stehen.

- Datenbank (*Professional und Enterprise Edition*)

  Im Abschnitt *Datenbank* können Sie Datenbankabfrageeinstellungen definieren. Nähere Informationen dazu finden Sie unter Datenbankabfrageeinstellungen[1547].

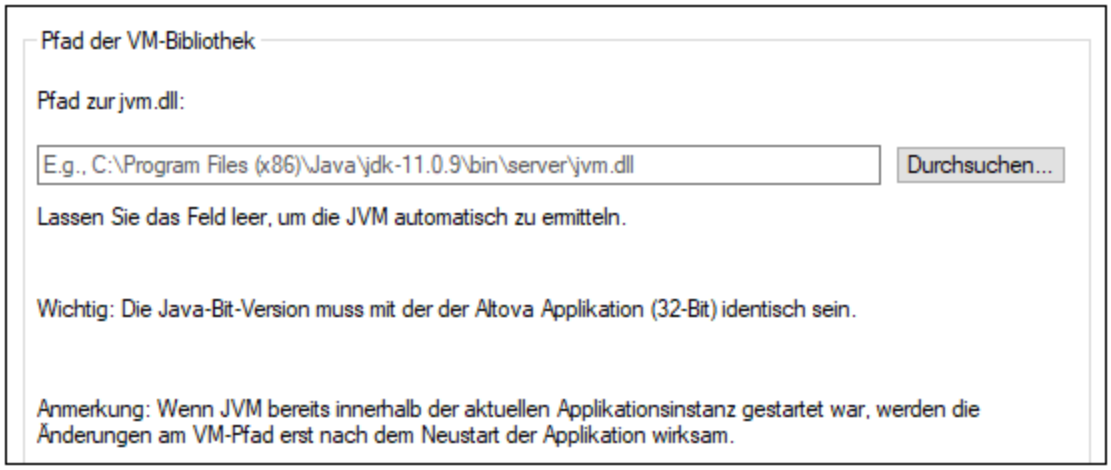
- Netzwerk-Proxy

  Im Abschnitt *Netzwerk-Proxy* können Sie die benutzerdefinierten Proxy-Einstellungen konfigurieren. Diese Einstellungen legen fest, wie sich die Applikation mit dem Internet verbindet. Standardmäßig werden die Proxy-Einstellungen des Systems verwendet, d.h. die Einstellungen funktionieren, ohne dass der Benutzer etwas daran ändern muss. Nähere Informationen dazu finden Sie unter Netzwerk-Proxy[1552].

15.11.3.1 Java

Im Abschnitt *Java* (*siehe Abbildung unten*) haben Sie die Möglichkeit, den Pfad zu einer Java VM (Virtual Machine) auf Ihrem Dateisystem einzugeben. Beachten Sie, dass dies nicht immer notwendig ist. MapForce versucht standardmäßig den Java VM-Pfad automatisch zu ermitteln. Dazu wird zuerst die Windows Registry und anschließend die JAVA_HOME-Umgebungsvariable gelesen. Ein in dieses Dialogfeld eingegebener benutzerdefinierter Pfad hat Vorrang vor allen automatisch ermittelten Java VM-Pfaden.

Wenn Sie eine Java Virtual Machine verwenden, die keinen Installer hat und keine Registry-Einträge erstellt (z.B. OpenJDK von Oracle), müssen Sie eventuell einen benutzerdefinierten Java VM-Pfad angeben. Auch wenn Sie automatisch von MapForce ermittelte Java VM-Pfade aus irgendeinem Grund außer Kraft setzen müssen, müssen Sie diesen Pfad eventuell definieren.



Beachten Sie dazu Folgendes:

- Der Java VM-Pfad wird gemeinsam von allen Altova Desktop-Applikationen (nicht aber den Server-

 © 2016-2022 Altova GmbH

Applikationen) verwendet. Wenn Sie den Pfad daher in einer Applikation ändern, gilt dies automatisch auch für alle anderen Altova-Applikationen.
- Der Pfad muss auf die Datei **jvm.dll** im Verzeichnis **\bin\server** oder **\bin\client** (relativ zum Verzeichnis, in dem JDK installiert ist) verweisen.
- Die MapForce-Plattform (32-Bit, 64-Bit) muss mit der des JDK identisch sein.
- Nachdem Sie den Java VM-Pfad geändert haben, müssen Sie MapForce eventuell neu starten, damit die neuen Einstellungen wirksam werden.

Eine Änderung des Java VM-Pfads wirkt sich auf die folgenden Bereiche aus:

- JDBC-Verbindungen
- Java-Erweiterungsfunktionen für XSLT/XPath

Diese Einstellung hat keine Auswirkung auf die Java-Codegenerierung.

15.11.3.2 XBRL

Sie können in MapForce die folgenden allgemeinen applikationsweiten XBRL-Einstellungen konfigurieren:

- die Label-Sprache für die XBRL Items und ihre Annotationen
- die bevorzugten Label-Roles für XBRL Item-Namen
- den spezifischen Typ von Label-Roles von Annotationen für XBRL Items
- Benutzerdefinierte XBRL-Taxonomiepakete

Im XBRL-Glossar finden Sie eine Definition von Labels. Nähere Informationen über Label Roles finden Sie unter *When to Use Multiple Label Roles and Types*.

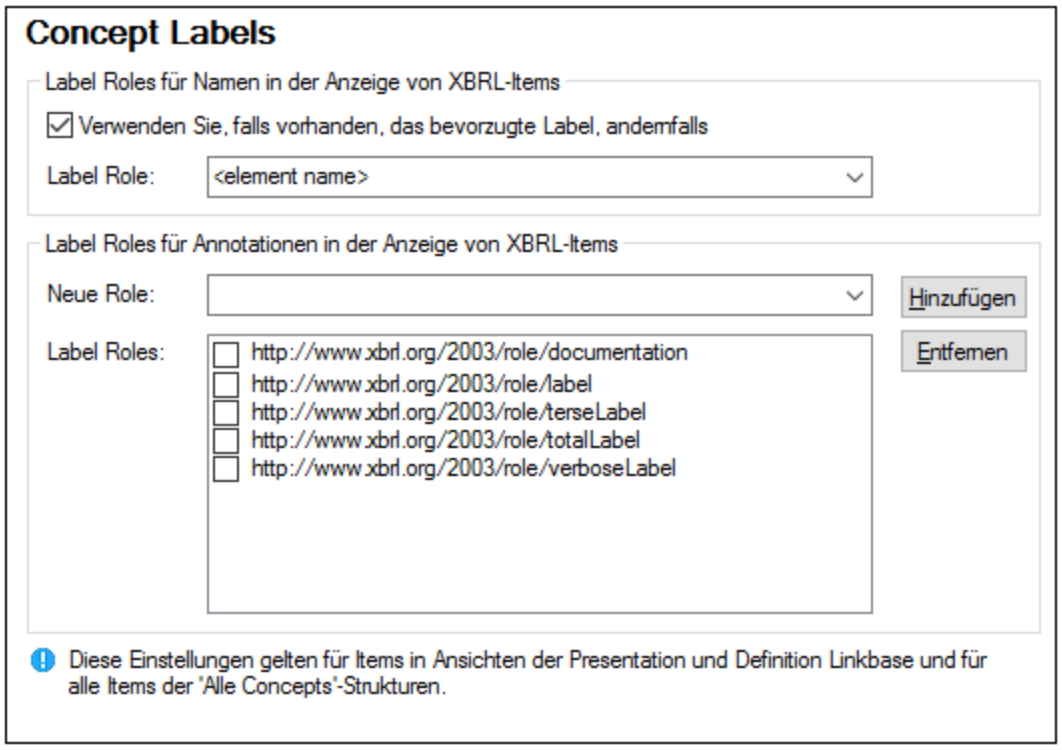
Um XBRL-Optionen zu konfigurieren, wählen Sie **Extras | Optionen** oder **Ansicht | XBRL-Anzeigeoptionen** und klicken Sie auf den *XBRL*-Abschnitt. Nähere Informationen dazu finden Sie in den Unterabschnitten weiter unten.

Allgemeine Einstellungen, Concept Labels, Generische Labels

In diesem Unterabschnitt werden verschiedene allgemeine Einstellungen, die Sie im Abschnitt *XBRL* definieren können, beschrieben. Dazu gehören die Label-Sprache von XBRL-Items und -Annotationen, Label Roles und Annotationen in verschiedenen Strukturansichten[770].

Allgemein
Im Unterabschnitt *Allgemein* können Sie die Label-Sprache von XBRL-Items und deren Annotationen wechseln.

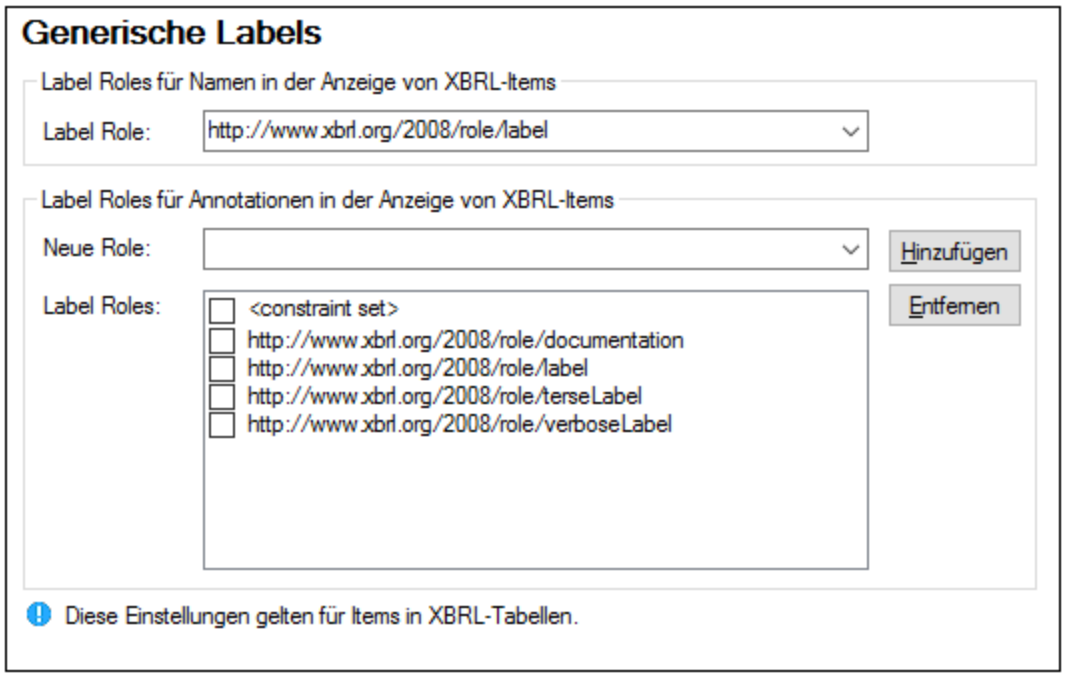
Concept Labels
Im Unterabschnitt *Concept Labels* (*siehe unten*) können Sie Label Roles und Annotationen in den folgenden Strukturansichten ändern: *Ansichten aus Presentation und Definition Linkbases*, *Alle Concepts (mit Kontext-Management)* und *Alle Concepts (nicht hierarchisch)*. Nähere Informationen zu Strukturansichten finden Sie unter Auswählen von Strukturansichten[770].



Generische Labels

Im Unterabschnitt *Generische Labels* können Sie Label Roles und Annotationen von Items in der Ansicht *Tabellen aus Tabellen-Linkbase* ändern: Nähere Informationen dazu finden Sie unter Auswählen von Strukturansichten[770].

© 2016-2022 Altova GmbH



Taxonomiepakete

Im Unterabschnitt *Taxonomiepakete* können Sie Standard-Taxonomiepakete und benutzerdefinierte Taxonomiepakete hinzufügen und ändern. Ein XBRL-Taxonomiepaket ist ein gezipptes Archiv, das eine Offline-Kopie einer Taxonomie enthält. Die Vorteile eines XBRL-Taxonomiepakets sind, dass XBRL-Ressourcen wie z.B. Schemas von lokalen (Offline) Taxonomiedateien anstatt über das Internet geladen werden. Eine Beschreibung der Struktur und des Formats von XBRL-Taxonomiepaketen finden Sie in der Taxonomy Packages 1.0 Recommendation.

Wenn Ihr XBRL-Taxonomieanbieter XBRL-Taxonomiepakete, die der obigen Recommendation entsprechen, zur Verfügung stellt, können Sie solche Pakete zu MapForce hinzufügen. Dies wirkt sich insofern auf die XBRL-Validierung und -Verarbeitung aus, als bestimmte URLs wie z.B. XBRL-Schemareferenzen auf lokale im Taxonomiepaket enthaltene Dateien umgeleitet werden. Mit Hilfe von Taxonomiepaketen können Sie alle Eintrittspunkte der XBRL-Taxonomie direkt in MapForce anzeigen und die XBRL-Validierung beschleunigen.

Viele XBRL-Taxonomien werden von Altova gehostet und können ganz einfach mit Hilfe des XBRL-Taxonomie-Managers 777. verwaltet werden. Falls die XBRL-Taxonomie, die Sie verwenden möchten, bereits vom XBRL-Taxonomie-Manager verwaltet wird, muss dafür kein benutzerdefiniertes Taxonomiepaket heruntergeladen werden.

Altova-Applikationen mit XBRL-Unterstützung
Die folgenden Altova Desktop-Applikationen unterstützen XBRL-Taxonomiepakete:

- Altova XBRL Add-ins für Excel (EBA, Solvency II, ESEF)
- MapForce Enterprise Edition
- MapForce Server
- RaptorXML+XBRL Server
- StyleVision Server

- StyleVision Enterprise Edition
- XMLSpy Enterprise Edition

Die Liste der Taxonomiepakete wird von allen diesen Applikationen gemeinsam verwendet. Änderungen in XBRL-Taxonomien, die den XBRL-Taxonomie-Manager[777] verwenden, werden für alle unten aufgelisteten Applikationen wirksam, wenn diese auf demselben Rechner installiert sind.

Funktionsweise von Taxonomiepaketen
Jedes XBRL-Taxonomiepaket enthält eine interne Katalogdatei (**`catalog.xml`**). In dieser Katalogdatei sind die URIs definiert, die auf lokale Ressourcen umgeleitet werden und somit neu zugewiesen werden sollen. Katalogdateien entsprechen einer eingeschränkten Untergruppe der XML Catalog-Spezifikation. Auch MapForce verwendet eine ähnliche Methode, um URIs neuen lokalen Ressourcen zuzuordnen (siehe Katalogdateien[1483]).

Nachdem Sie ein oder mehrere XBRL-Taxonomiepakete zu MapForce hinzugefügt haben, wirken sich diese Pakete ab diesem Moment darauf aus, wie MapForce Ressourcen für die XBRL-Validierung auflöst. Wenn Sie z.B. eine XBRL-Instanz zum Mapping hinzufügen, wird das dazugehörige Schema aus dem lokalen Taxonomiepaket anstatt aus dem Internet geladen (Voraussetzung dafür ist, dass das Taxonomiepaket aktiv ist und dass die dazugehörige Datei **`catalog.xml`** die Schema-URI auf eine lokale Ressource umleitet). Dies wirkt sich in der Folge auf alle XBRL-Validierungs- und Verarbeitungsvorgänge aus.

Anmerkung: Eine Ressource, die durch die Katalogdatei(en) eines aktiven Pakets referenziert wird, wird von allen MapForce-Operationen, die diese benötigen, verwendet. Ein Beispiel dafür wäre ein XML-Schema, das sowohl für die XML- als auch für die XBRL-Validierung verwendet wird.

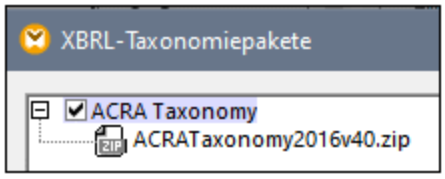
Wenn die von der Katalogdatei des Pakets referenzierte Offline-Ressource mit Ihrer vorhandenen Umgebung nicht kompatibel ist, kann dies zu Fehlern führen. Deaktivieren Sie in diesem Fall das Taxonomiepaket und wenden Sie sich an den Ersteller des Pakets.

Hinzufügen eines Taxonomiepakets
Um ein Standard-Taxonomiepaket zu installieren oder zu ändern, verwenden Sie den XBRL-Taxonomie-Manager[777]. Um ein benutzerdefiniertes Taxonomiepaket hinzuzufügen, gehen Sie folgendermaßen vor:

1. Laden Sie das XBRL-Taxonomiepaket des Anbieters herunter und speichern Sie es in einem lokalen Verzeichnis, das aller Voraussicht nach nicht geändert wird. Entpacken Sie das heruntergeladene Archiv *nicht*. Das heruntergeladene **`.zip`**-Archiv muss der Taxonomy Packages 1.0 Recommendation entsprechen.
2. Klicken Sie im Menü **Extras | Optionen | XBRL | Taxonomiepakete** auf **Bearbeiten**.
3. Klicken Sie auf **Taxonomie hinzufügen**, navigieren Sie zum gewünschten **`.zip`**-Taxonomiepaket. Um mehrere Pakete auszuwählen, halten Sie die **Umschalttaste** gedrückt.

Die Liste der hinzugefügten XBRL-Taxonomiepakete wird im Dialogfeld in Form einer aus zwei Ebenen bestehenden Struktur angezeigt. (i) Auf der ersten Ebene wird die Taxonomie angezeigt; (ii) auf der zweiten Ebene sehen Sie die Pakete der Taxonomie (*siehe Abbildung unten*). Wenn ein Taxonomiepaket im oberen Bereich des Dialogfelds **XBRL-Taxonomiepakete** ausgewählt ist, werden die Einzelheiten dazu im unteren Bereich des Dialogfelds angezeigt.

© 2016-2022 Altova GmbH



Das Kontrollkästchen links vom Taxonomieeintrag zeigt an, ob diese Taxonomie aktiv ist. Eine neu hinzugefügte Taxonomie ist standardmäßig aktiv. Wenn Sie weitere Pakete zu einer Taxonomie hinzufügen möchten, wählen Sie diese Taxonomie aus und klicken Sie auf die Schaltfläche **Pakete hinzufügen**. Um ein Paket zu entfernen, wählen Sie es aus und klicken Sie auf **Entfernen**. (Beachten Sie, dass die Taxonomiedateien damit nicht von der Festplatte gelöscht werden).

15.11.3.3 Datenbank

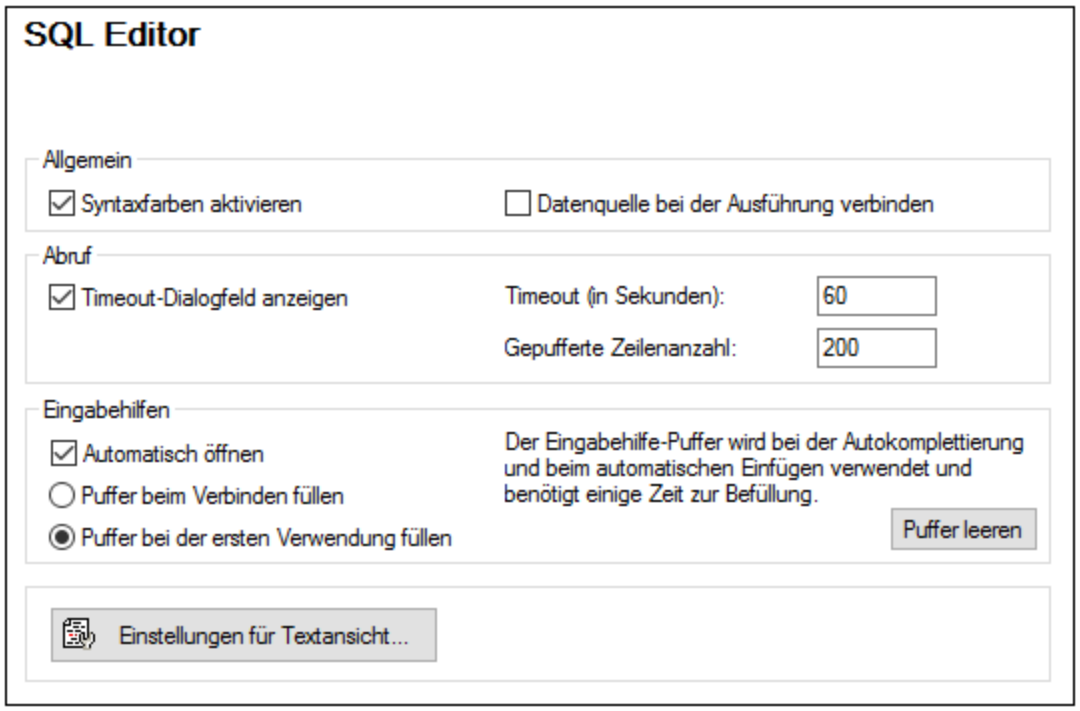
In diesem Abschnitt wird erläutert, wie Sie verschiedene SQL-Einstellungen konfigurieren. Sie können die Einstellungen auf eine der folgenden Arten aufrufen:

- Durch Aufruf des **DB-Abfrage**-Fensters und Klick auf  (**Optionen**).
- Durch Auswahl von **Extras | Optionen | Datenbank** und anschließend Öffnen des entsprechenden Abschnitts.

Im Abschnitt *Datenbank* können Sie allgemeine SQL-Bearbeitungseinstellungen, Optionen für die Kodierung und Ergebnisansicht, Parameter für die SQL-Generierung und Schriftarten definieren. Nähere Informationen dazu finden Sie in den Unterabschnitten weiter unten.

SQL Editor

Im Abschnitt *SQL Editor* können Sie die allgemeinen SQL-Bearbeitungseinstellungen ändern (*siehe Abschnitt unten*). Weiter unten finden Sie eine Beschreibung der verfügbaren Einstellungen.



- *Allgemein.* Um verschiedene Elemente der SQL-Syntax in unterschiedlichen Farben zu sehen, aktivieren Sie die Syntaxfärbung. Aktivieren Sie das Kontrollkästchen *Datenquelle bei der Ausführung verbinden*, um jedes Mal, wenn eine SQL-Anweisung ausgeführt wird, automatisch die Verbindung zur entsprechenden Datenquelle herzustellen.

- *Abruf.* Wenn Sie das Kontrollkästchen *Timeout-Dialogfeld anzeigen* aktivieren, können Sie die Timeout-Einstellungen ändern, wenn die zulässige Ausführungszeit überschritten wird. Sie können die maximal für die SQL-Ausführung zulässige Zeit in Sekunden angeben (*Ausführungs-Timeout*). Außerdem können Sie festlegen, wie viele Zeilen in einen Puffer geschrieben werden sollen.

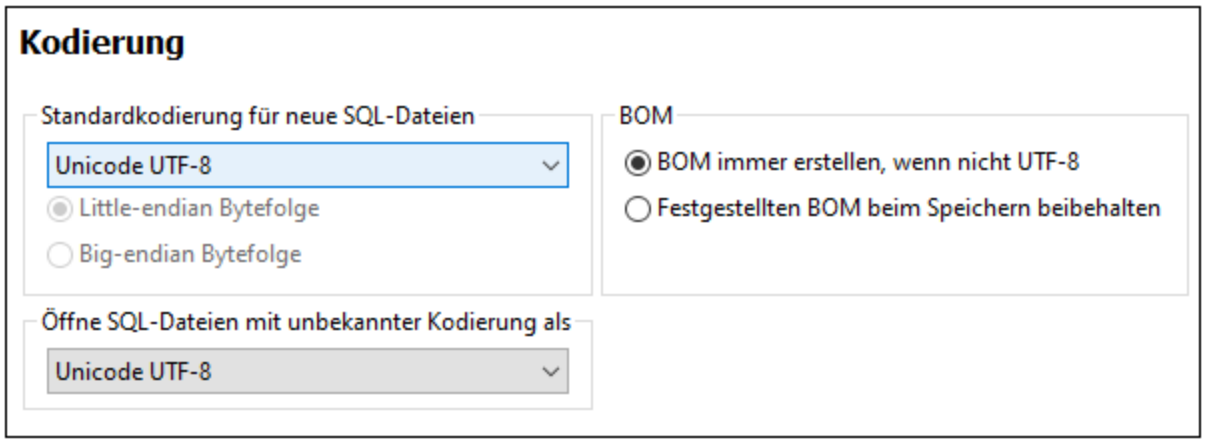
- *Eingabehilfen.* Um die Autokomplettierungsvorschläge bei Beginn der Eingabe von SQL-Anweisungen zu aktivieren, aktivieren Sie das Kontrollkästchen *Automatisch öffnen* (siehe auch SQL-Autokomplettierungsvorschläge 286). Sie können auswählen, wann der Eingabehilfe-Puffer befüllt werden soll: Wenn Sie eine Verbindung zu einer Datenquelle herstellen oder bei der ersten Verwendung des Puffers. Beachten Sie, dass dieser Vorgang einige Zeit dauern kann. Über die Schaltfläche **Puffer leeren** können Sie den Puffer zurücksetzen.

- *Einstellungen für Textansicht* Hier können Sie verschiedene Einstellungen für die Textansicht wie Leisten, Register, visuelle Hilfen, Optionen zur automatischen Markierung sowie Tastaturkürzel für die Navigation in der Textansicht definieren. Nähere Informationen dazu finden Sie unter Funktionalitäten der Textansicht 110.

Kodierung

Im Abschnitt *Kodierung* können Sie Kodierungsoptionen für mit dem SQL Editor erstellte oder geöffnete SQL-Dateien definieren (*siehe Abbildung unten*).

 © 2016-2022 Altova GmbH



- *Standardkodierung für neue SQL-Dateien:* Definieren Sie die Standardkodierung für neue Dateien, so dass jedes neue Dokument dieselbe Kodierung hat. Wenn als Standardkodierung eine Zwei- oder Vier-Byte Kodierung (d.h. UTF-16, UCS-2 oder UCS-4) festgelegt ist, können Sie auch zwischen der Little-Endian und der Big-Endian Bytefolge für SQL-Dateien wählen. Diese Einstellung hat keine Auswirkung auf die Kodierung vorhandener Dateien.

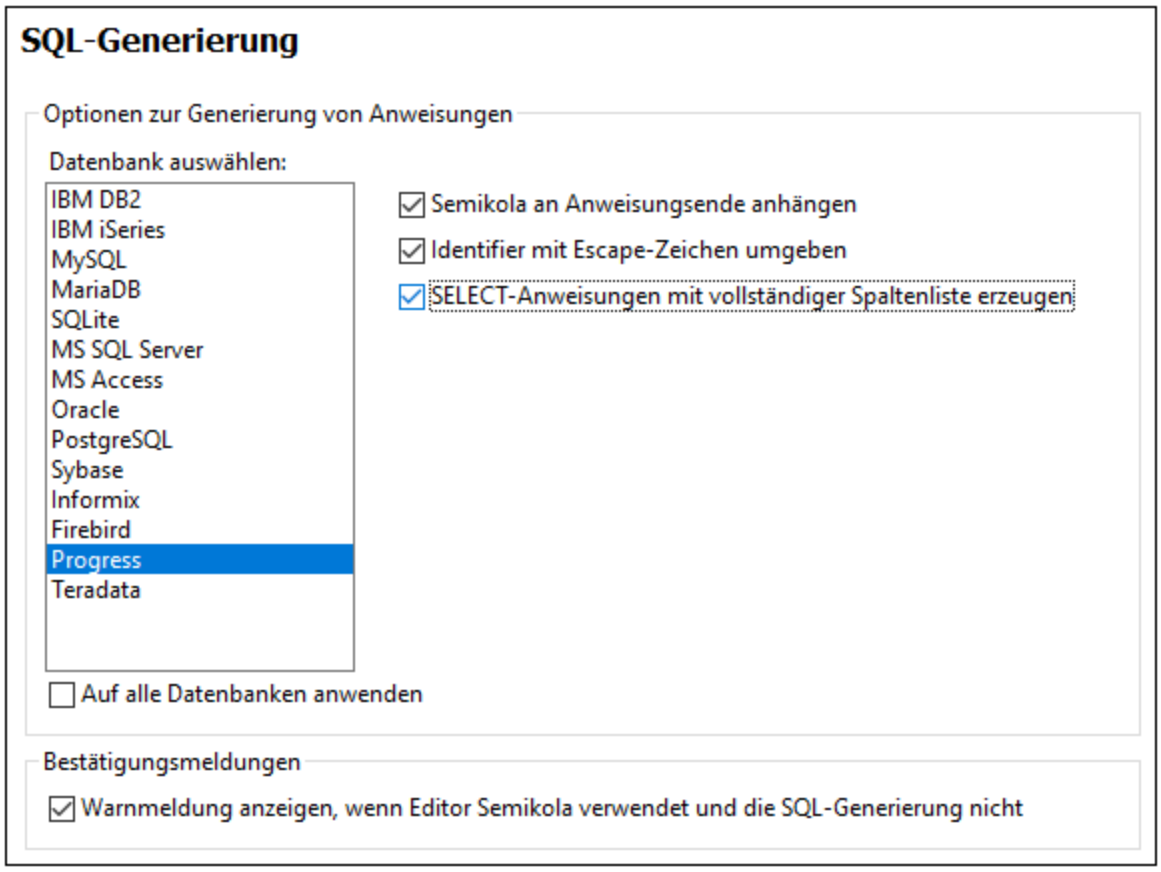
- *Öffne SQL-Dateien mit unbekannter Kodierung als:* Sie können die Kodierung, mit der eine SQL-Datei mit unbekannter Kodierung geöffnet werden soll, auswählen.

- *BOM:* Wählen Sie aus, ob das Bytefolgemarkierungszeichen (BOM) für andere Kodierungen als UTF-8 erstellt werden soll oder ob das erkannte BOM beibehalten werden soll.

Anmerkung: SQL-Dateien ohne Kodierungsspezifikation werden mit einer UTF-8-Kodierung gespeichert**.**

SQL-Generierung

Im Abschnitt *Generierung* können Sie die Syntax für die Generierung von SQL-Anweisungen für verschiedene Datenbankarten festlegen (*siehe Abbildung unten*).

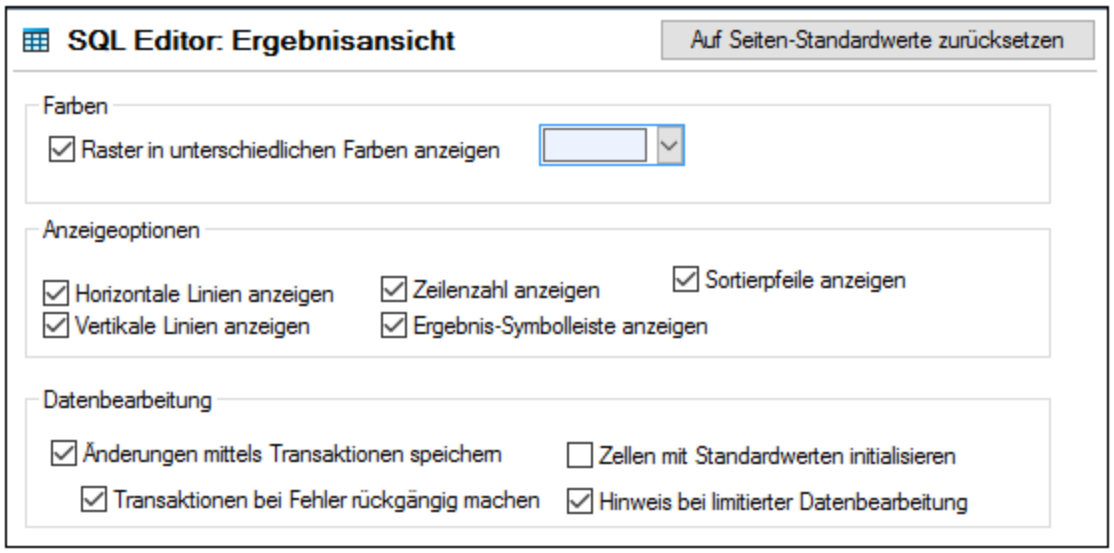


Um die Syntaxeinstellungen für eine bestimmte Datenbank zu definieren, wählen Sie diese aus der Liste aus und aktivieren oder deaktivieren Sie Optionen auf der rechten Seite. Sie können dieselben Einstellungen auf Wunsch auch auf alle Datenbanken anwenden (Kontrollkästchen *Auf alle Datenbanken anwenden*). Beachten Sie, dass bei Verwendung einer gemeinsamen Einstellung für alle Datenbanken, eventuell Daten in Oracle-, IBM DB2- und iSeries-Datenbanken über eine JDBC-Verbindung nicht mehr bearbeitet werden können.

Ergebnisansicht

Im Abschnitt *Ergebnisansicht* können Sie das Aussehen des Registers **Ergebnisse** des **DB-Abfragefensters** konfigurieren (*siehe Abbildung unten*).

© 2016-2022 Altova GmbH

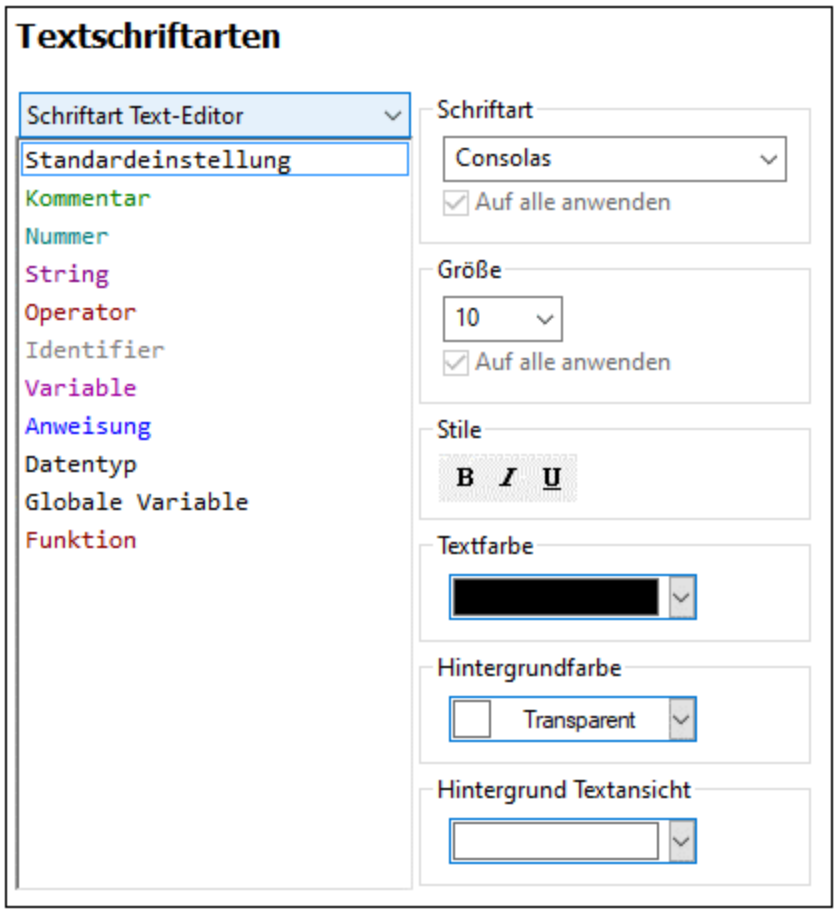


Aktivieren Sie das Kontrollkästchen *Raster in unterschiedlichen Farben anzeigen*, um die Zeilen auf den **Ergebnisregistern** als a einfaches Raster oder als Raster mit abwechselnd farbigen und weiße Zeilen anzuzeigen. Die alternierende Farbe ist konfigurierbar. In der Gruppe *Anzeigeoptionen* können Sie definieren, wie horizontale und vertikale Linien sowie Zeilennummern und die **Ergebnis**-Symbolleiste angezeigt werden.

In der Gruppe *Datenbearbeitung* können Sie die Transaktionseinstellungen definieren, ob die Zellen mit Standardwerten befüllt werden sollen und ob ein Hinweis angezeigt werden soll, wenn die Datenbearbeitung eingeschränkt ist.

Textschriftarten

Im Abschnitt *Textschriftarten* können Sie die Farbe und die Schriftarteinstellungen von SQL-Anweisungen konfigurieren (*siehe Abbildung unten*). Sie können die allgemeine Schriftart, den Stil, die Textgröße und die Syntaxfärbung von verschiedenen im SQL-Editor angezeigten Texttypen auswählen. Beachten Sie, dass für alle Texttypen dieselbe Schriftart und -größe verwendet wird. Nur der Stil kann für einzelne Textarten geändert werden. Wenn Sie das Dialogfeld **Textschriftarten** über das **DB-Abfrage**-Fenster aufrufen, können Sie durch Klick auf die Schaltfläche **Auf Seiten-Standardwerte zurücksetzen** die Originaleinstellungen wiederherstellen.

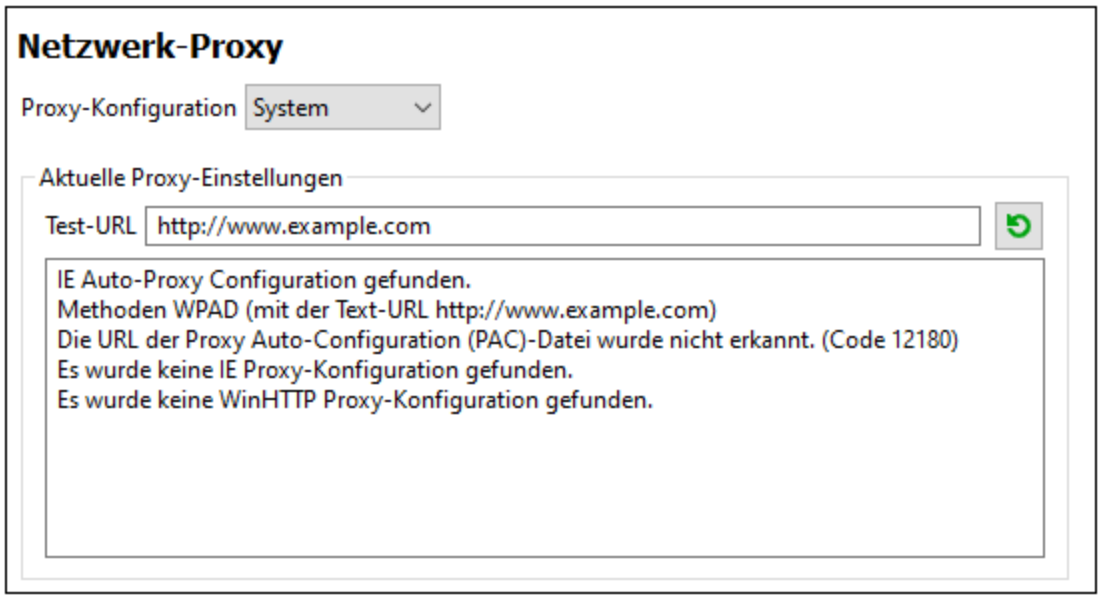


15.11.3.4 Netzwerk-Proxy

Im Abschnitt *Netzwerk-Proxy* können Sie die benutzerdefinierten Proxy-Einstellungen konfigurieren. Diese Einstellungen beeinflussen, wie die Applikation eine Verbindung mit dem Internet herstellt (z.B. zur XML-Validierung). Standardmäßig werden die Proxy-Einstellungen des Systems verwendet, d.h. die Einstellungen funktionieren, ohne dass der Benutzer etwas daran ändern muss. Falls nötig, können Sie jedoch einen anderen Netzwerk-Proxy-Server definieren. Wählen Sie dazu in der Auswahlliste *Proxy-Konfiguration* entweder die Option *Automatisch* oder *Manuell*, um die Einstellungen entsprechend zu konfigurieren.

Anmerkung: Die Nethwerk-Proxy-Einstellungen werden von allen Altova MissionKit-Applikationen gemeinsam verwendet. Wenn Sie daher die Einstellungen in einer Applikation ändern, wirkt sich dies automatisch auf alle anderen Applikationen aus.

© 2016-2022 Altova GmbH

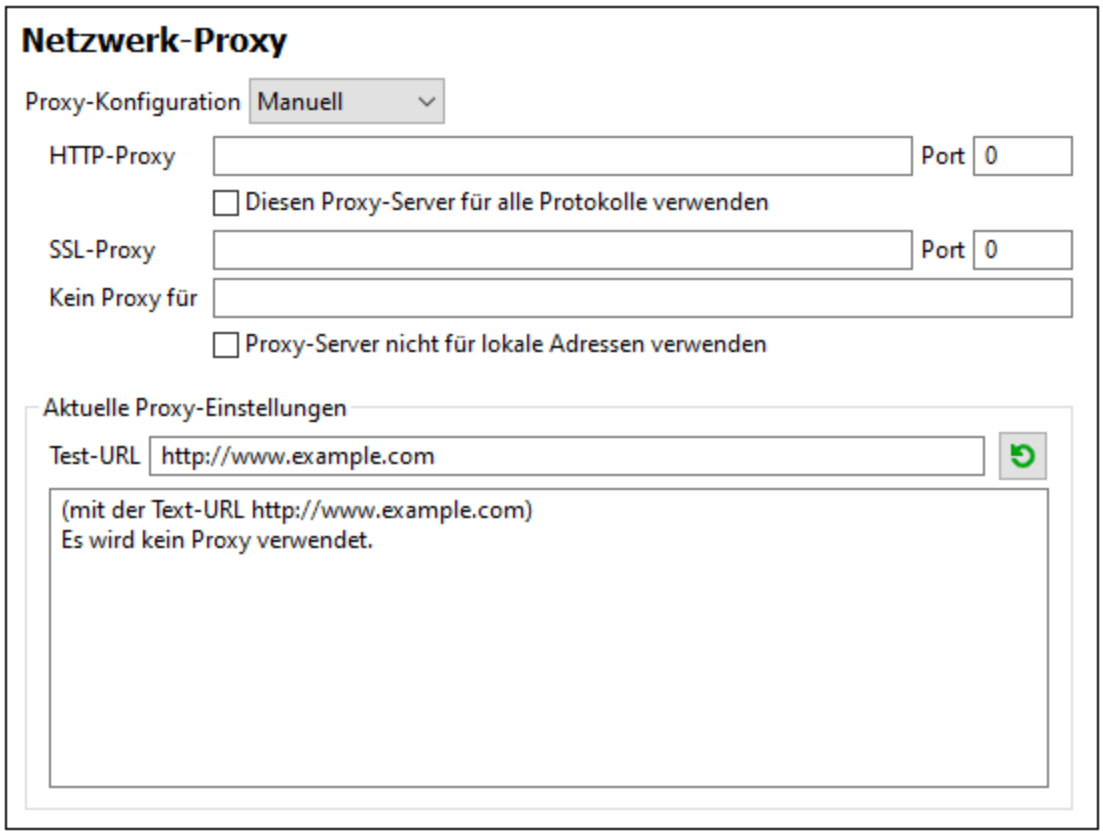


System-Proxy-Einstellungen verwenden
Dadurch werden die über die System-Proxy-Einstellungen konfigurierbaren Internet Explorer (IE)-Einstellungen verwendet. Führt außerdem eine Abfrage der mit **`netsh.exe winhttp`** konfigurierten Einstellungen durch.

Automatische Proxy-Konfiguration
Es stehen die folgenden Optionen zur Verfügung:

- *Einstellungen automatisch ermitteln:* **`Verwendet ein WPAD-Skript`** **(**http://wpad.LOCALDOMAIN/wpad.dat) über DHCP oder DNS, um die Einrichtung des Proxy-Servers zu konfigurieren.
- *Skript-URL:* Definieren Sie eine HTTP URL zu einem automatischen Proxy-Konfigurationsskript (**`.pac`**), mit dem der Proxy-Server eingerichtet wird.
- *Neu laden:* Setzt die aktuelle automatische Proxy-Konfiguration zurück und lädt sie neu. Dafür ist Windows 8 oder neuer erforderlich. Die Rücksetzung kann bis zu 30 Sekunden dauern.

Manuelle Proxy-Konfiguration
Definieren Sie den vollständig qualifizierten Host-Namen und Port für die Proxy-Server der jeweiligen Protokolle manuell. Im Host-Namen kann ein unterstütztes Schema inkludiert werden (z.B.: **`http://hostname`**). Das Schema muss nicht mit dem entsprechenden Protokoll übereinstimmen, wenn der Proxy-Server das Schema unterstützt.



Es stehen die folgenden Optionen zur Verfügung:

- *HTTP-Proxy:* Verwendet den angegebenen Host-Namen und Port für das HTTP-Protokoll. Wenn *Diesen Proxy-Server für alle Protokolle verwenden* aktiviert ist, wird der angegebene HTTP-Proxy-Server für alle Protokolle verwendet.
- *SSL-Proxy:* Verwendet den angegebenen Host-Namen und Port für das SSL-Protokoll.
- *Kein Proxy für:* eine durch Semikola (`;`) getrennte Liste von voll qualifizierten Host-Namen, Domain-Namen oder IP-Adressen für Hosts, die ohne einen Proxy-Server verwendet werden sollen. IP-Adressen dürfen nicht abgeschnitten werden und IPv6-Adressen müssen innerhalb von eckige Klammern gesetzt werden (z.B.: **`[2606:2800:220:1:248:1893:25c8:1946]`**). Domain-Namen muss ein Punkt vorangestellt werden (z.B: **`.example.com`**).
- *Proxy-Server nicht für lokale Adressen verwenden:* Falls dieses Kontrollkästchen aktiviert ist, wird **`<local>`** zur *Kein Proxy für*-Liste hinzugefügt. Falls diese Option ausgewählt ist, wird für die folgenden Adressen kein Proxy-Server verwendet: (i) **`127.0.0.1`**, (ii) [::1], (iii) alle Host-Namen, die kein Punktzeichen (`.`) enthalten.

Anmerkung: Wenn ein Proxy-Server definiert wurde und Sie ein Mapping auf Altova FlowForce Server bereitstellen möchten, müssen Sie die Option *Proxy-Server nicht für lokale Adressen verwenden* aktivieren

Aktuelle Proxy-Einstellungen
Stellt ein ausführliches Protokoll der Proxy-Ermittlung bereit. Es kann über die Schaltfläche **Aktualisieren** rechts vom Feld *Test-URL* aktualisiert werden (z.B. bei Wechsel zu einer anderen Test-URL oder wenn die Proxy-Einstellungen geändert wurden).

© 2016-2022 Altova GmbH

- *Test-URL:* Anhand einer Test-URL kann ermittelt werden, welcher Proxy-Server für diese bestimmte URL verwendet wird. Mit dieser URL erfolgt kein I/O. Dieses Feld darf nicht leer sein, wenn die automatische Proxy-Konfiguration verwendet wird (entweder über *System-Proxy-Einstellungen verwenden* oder *Automatische Proxy-Konfiguration*).

15.12 Fenster

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Fenster**.

- Überlappend

  Mit diesem Befehl werden alle offenen Dokumentenfenster so angeordnet, dass sie einander überlappend angezeigt werden.

- Horizontal anordnen

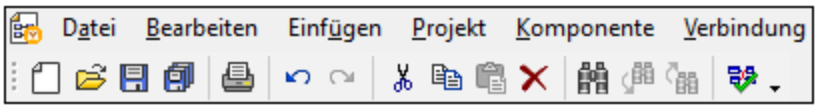
  Mit diesem Befehl ordnen Sie alle offenen Dokumentfenster horizontal nebeneinander an, sodass alle gleichzeitig sichtbar sind.

- Vertikal anordnen

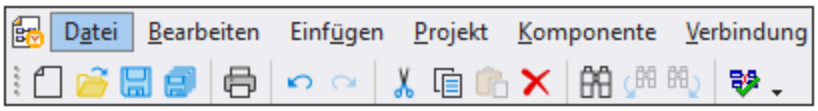
  Mit diesem Befehl werden alle offenen Dokumentenfenster vertikal übereinander angeordnet, sodass alle gleichzeitig sichtbar sind.

- Klassisches/Helles/Dunkles Design

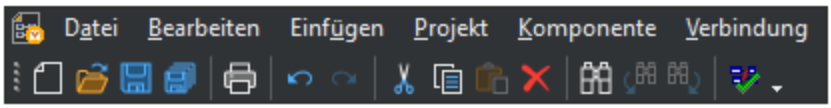
  In MapForce stehen die folgenden Designs zur Auswahl: *Klassisch*, *Hell* und *Dunkel*. Beispiele für diese Designs finden Sie in den Abbildungen weiter unten. Die Standardoption ist das klassische Design.

  Klassisches Design

  Helles Design

  Dunkles Design

- 1 <MappingName>

  Bezieht sich auf das erste offene Mapping-Design. Wenn mehrere Mappings gleichzeitig offen sind, werden diese ebenfalls im Kontextmenü aufgelistet.

- Fenster

  In der Liste sehen Sie alle offenen Fenster und Sie können jederzeit zwischen diesen Fenstern wechseln. Um zwischen den Fenstern zu wechseln, können Sie auch die Tastaturkürzel **Strg+Tab** oder **Strg+F6** verwenden.

© 2016-2022 Altova GmbH

15.13 Hilfe

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Hilfe**.

- Inhaltsverzeichnis

  Mit dem Befehl "Inhaltsverzeichnis" wird die Bildschirmhilfe für MapForce geöffnet, wobei das Inhaltsverzeichnis auf der linken Seite des Hilfefensters angezeigt wird. Das Inhaltsverzeichnis bietet eine Übersicht über das gesamte Hilfedokument. Wenn Sie auf einen Eintrag im Inhaltsverzeichnis klicken, gelangen Sie zu diesem Thema.

- Index

  Mit dem Befehl Index wird die Bildschirmhilfe für MapForce geöffnet, wobei auf der linken Seite des Hilfefensters der Stichwortindex angezeigt wird. Im Index werden Stichwörter aufgelistet und Sie können durch Doppelklick auf das Stichwort zum jeweiligen Thema navigieren. Wenn ein Stichwort mit mehr als einem Thema verknüpft ist, können Sie das Thema aus einer Liste von Themen auswählen.

- Suchen

  Mit dem Befehl Suchen wird die Bildschirmhilfe für MapForce geöffnet, wobei auf der linken Seite des Hilfefensters das Suchdialogfeld angezeigt wird. Um nach einem Begriff zu suchen, geben Sie ihn in das Eingabefeld ein und drücken Sie die **Eingabetaste** oder klicken Sie auf **Kapitel auflisten**. Das Hilfesystem führt eine Volltextsuche am gesamten Hilfedokument durch und gibt eine Trefferliste zurück. Doppelklicken Sie auf einen Eintrag, um das Thema anzuzeigen.

- Software-Aktivierung

  Lizenzieren Ihres Produkts
  Nachdem Sie Ihre Altova-Software heruntergeladen haben, können Sie sie entweder mit Hilfe eines kostenlosen Evaluierungs-Keycode oder eines käuflich erworbenen permanenten Lizenzkeycode lizenzieren oder aktivieren.

  - ***Kostenlose Evaluierungs-Lizenz.*** Wenn Sie die Software zum ersten Mal starten, wird das Dialogfeld **Software-Aktivierung** angezeigt. Es enthält eine Schaltfläche, über die Sie eine kostenlose Evaluierungs-Lizenz anfordern können. Geben Sie in das Dialogfeld Ihren Namen, den Namen Ihrer Firma und Ihre E-Mail-Adresse ein und klicken Sie auf **Anfordern**. Daraufhin wird an die von Ihnen angegebene E-Mail-Adresse eine Lizenz gesendet und sollte in wenigen Minuten eintreffen. Speichern Sie die Lizenzdatei in einem geeigneten Ordner.

    Wenn Sie auf **Anfordern** klicken, wird am unteren Rand des Dialogfelds ein Eingabefeld angezeigt. Über dieses Feld gelangen Sie zum Pfad der Lizenzdatei. Navigieren Sie zur Lizenzdatei oder geben Sie den Pfad dazu ein und klicken Sie auf **OK**. (Sie können im Dialogfeld **Software-Aktivierung** auch auf **Neue Lizenz hochladen** klicken, um das Dialogfeld, in das der Pfad zur Lizenzdatei eingegeben wird, aufzurufen). Die Software wird für einen Zeitraum von 30 Tagen aktiviert.

  - ***Permanenter Lizenz-Keycode.*** Über das Dialogfeld **Software-Aktivierung** können Sie einen permanenten Lizenz-Keycode erwerben. Wenn Sie auf diese Schaltfläche klicken, gelangen Sie zum Altova Online Shop, in dem Sie einen permanenten Lizenzschlüssel für Ihr Produkt erwerben können. Ihre Lizenz wird Ihnen in Form einer Lizenzdatei, die Ihre Lizenzdaten enthält, per E-Mail zugesendet.

© 2016-2022 Altova GmbH

Es gibt drei Arten von permanenten Lizenzen: *Einzelplatzlizenzen*, *Parallellizenzen* und *Named User-Lizenzen* (benutzerdefinierte Nutzung). Mit einer Einzelplatzlizenz wird die Software auf einem einzigen Rechner freigeschaltet. Wenn Sie eine Einzelplatzlizenz für *N* Rechner erwerben, gestattet Ihnen die Lizenz, die Software auf bis zu *N* Rechnern zu verwenden. Mit einer Parallellizenz für *N* Parallelbenutzer dürfen *N* Benutzer die Software gleichzeitig ausführen. (Die Software darf auf 10N Rechnern installiert sein.) Mit einer Named User-Lizenz darf ein bestimmter Benutzer die Software auf bis zu 5 verschiedenen Rechnern verwenden. Um Ihre Software zu aktivieren, klicken Sie auf **Neue Lizenz hochladen** und geben Sie im daraufhin angezeigten Dialogfeld den Pfad zur Lizenzdatei ein und klicken Sie auf **OK**.

Anmerkung: Bei Mehrplatzlizenzen wird jeder Benutzer aufgefordert, seinen eigenen Namen einzugeben.

Ihre Lizenz-E-Mail und die verschiedenen Methoden, Ihr Altova-Produkt zu lizenzieren
Die Lizenz-E-Mail, die Sie von Altova erhalten, enthält Ihre Lizenzdatei im Anhang. Die Lizenzdatei hat die Dateierweiterung **`.altova_licenses`**.

Sie haben folgende Möglichkeiten, Ihr Altova-Produkt zu aktivieren:

- Speichern Sie die Lizenzdatei (**`.altova_licenses`**) in einem geeigneten Ordner, doppelklicken Sie auf die Lizenzdatei, geben Sie etwaige erforderliche Informationen in das Dialogfeld ein, das daraufhin angezeigt wird und beenden Sie den Vorgang durch Klicken auf **Lizenzschlüssel anwenden**.
- Speichern Sie die Lizenzdatei (**`.altova_licenses`**) in einem geeigneten Ordner. Wählen Sie in Ihrem Altova-Produkt den Menübefehl **Hilfe | Software-Aktivierung** und klicken Sie anschließend auf **Neue Lizenz hochladen.** Navigieren Sie zur Lizenzdatei oder geben Sie den Pfad dazu ein und klicken Sie auf **OK**.
- Speichern Sie die Lizenzdatei (**`.altova_licenses`**) in einem geeigneten Ordner und laden Sie diese von dort aus in den Lizenz-Pool Ihres Altova LicenseServer hoch. Sie können die Lizenz anschließend (i) entweder von Ihrem Altova-Produkt über das Dialogfeld "Software-Aktivierung" abrufen (*siehe unten*) oder (ii) dem Produkt die Lizenz von Altova LicenseServer aus zuweisen. *Nähere Informationen zur Lizenzierung über LicenseServer finden Sie weiter unten in diesem Kapitel.*

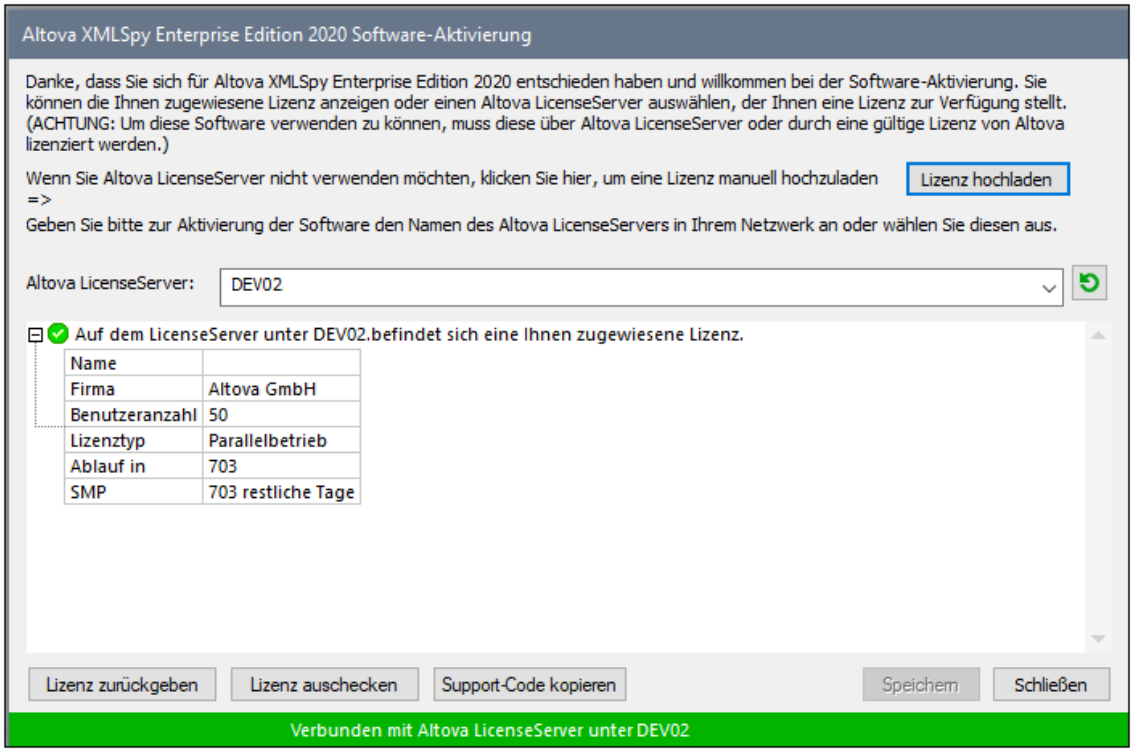
Das Dialogfeld **Software-Aktivierung** (*Abbildung unten*) kann über den Befehl **Hilfe | Software-Aktivierung** aufgerufen werden.

Aktivieren Ihrer Software
Sie können die Software durch Registrieren der Lizenz im Dialogfeld "Software-Aktivierung" oder durch Lizenzierung über Altova LicenseServer (*nähere Informationen siehe unten*) aktivieren.

- *Registrierung der Lizenz im Dialogfeld "Software-Aktivierung".* Klicken Sie im Dialogfeld auf **Neue Lizenz hochladen** und navigieren Sie zur Lizenzdatei. Klicken Sie auf **OK,** um den Pfad zur Lizenzdatei und alle eingegebenen Daten (im Fall einer Mehrplatzlizenz Ihren Namen) zu bestätigen und abschließend auf **Speichern**.

- *Lizenzierung über einen Altova LicenseServer in Ihrem Netzwerk:* Um eine Lizenz über einen Altova LicenseServer in Ihrem Netzwerk abzurufen, (klicken Sie am unteren Rand des Dialogfelds **Software-Aktivierung** auf **Altova LicenseServer verwenden**). Wählen Sie den Rechner aus,

auf dem der gewünschte LicenseServer installiert wurde. Beachten Sie, dass die automatische Ermittlung von License Servern durch die Aussendung eines Signals ins LAN erfolgt. Da diese Aussendung auf ein Subnetz beschränkt ist, muss sich der LicenseServer im selben Subnetz wie der Client-Rechner befinden, damit die Ermittlung von License Servern funktioniert. Falls die automatische Ermittlung nicht funktioniert, geben Sie den Namen des Servers ein. Der Altova LicenseServer muss in seinem Lizenzpool eine Lizenz für Ihre Altova-Produkt haben. Wenn im LicenseServer-Pool eine Lizenz verfügbar ist, wird dies im Dialogfeld **Software-Aktivierung** angezeigt (*siehe Abbildung unten, in der Sie das Dialogfeld in Altova XMLSpy sehen*) und Sie können auf **Speichern** klicken, um die Lizenz abzurufen.



Eine rechnerspezifische Lizenz (Einzelplatzlizenz) kann erst nach Ablauf von sieben Tagen wieder an LicenseServer zurückgegeben werden. Danach können Sie die rechnerspezifische Lizenz durch Klick auf **Lizenz zurückgeben** an den Server zurückgeben, sodass sie von einem anderen Client vom LicenseServer abgerufen werden kann. Ein LicenseServer-Administrator kann die Zuweisung einer abgerufenen Lizenz jedoch über die Web-Benutzeroberfläche von LicenseServer jederzeit aufheben. Beachten Sie, dass eine Rückgabe von Lizenzen nur bei rechnerspezifischen Lizenzen, nicht aber bei Parallellizenzen möglich ist.

Lizenz-Check-Out

Über den Lizenzpool können Sie eine Lizenz für einen Zeitraum von bis zu 30 Tagen auschecken, sodass die Lizenz auf dem lokalen Rechner gespeichert wird. Dadurch können Sie offline arbeiten, was nützlich ist, wenn Sie z.B. in einer Umgebung arbeiten möchten, in der Sie keinen Zugriff auf Ihren Altova LicenseServer haben (z.B. wenn Ihr Altova-Produkt auf einem Laptop installiert ist und Sie gerade unterwegs sind). Solange die Lizenz ausgecheckt ist, zeigt LicenseServer die Lizenz als in Verwendung an. Diese Lizenz kann dann von keinem anderen Rechner verwendet werden. Die Lizenz wird nach Ablauf des Check-Out-Zeitraums automatisch wieder eingecheckt. Alternativ dazu kann eine ausgecheckte Lizenz jederzeit über die Schaltfläche **Einchecken** des Dialogfelds **Software-Aktivierung** wieder eingecheckt werden.

© 2016-2022 Altova GmbH

Um eine Lizenz auszuchecken, gehen Sie folgendermaßen vor: (i) Klicken Sie im Dialogfeld **Software-Aktivierung** auf **Lizenz auschecken** (*siehe Abbildung oben*); (ii) Wählen Sie im daraufhin angezeigten Dialogfeld **Lizenz-Check-Out** den gewünschten Check-Out-Zeitraum aus und klicken Sie auf **Auschecken**. Daraufhin wird die Lizenz ausgecheckt. Nachdem Sie eine Lizenz ausgecheckt haben, geschehen zwei Dinge: (i) Die Check-Out-Informationen und das Ende des Check-Out-Zeitraums werden im Dialogfeld **Software-Aktivierung** angezeigt; (ii) Die Schaltfläche **Lizenz auschecken** im Dialogfeld ändert sich nun in **Einchecken**. Sie können die Lizenz jederzeit durch Klicken auf **Einchecken** einchecken. Da die Lizenz nach Ablauf des Check-Out-Zeitraums automatisch wieder in den Zustand "Eingecheckt" zurück wechselt, sollte der von Ihnen ausgewählte Zeitraum für das Check-Out den gewünschten Zeitraum, in dem Sie offline arbeiten möchten, entsprechend abdecken.

Wenn eine Lizenz wieder eingecheckt wird, muss diese Lizenz für dieselbe Hauptversion eines Altova-Produkts ausgestellt sein, wie die Lizenz, die ausgecheckt wurde. Stellen Sie daher sicher, dass die Lizenz eingecheckt ist, bevor Sie für Ihr Altova-Produkt ein Upgrade auf die nächste Hauptversion installieren.

Anmerkung: Damit Lizenzen ausgecheckt werden können, muss die Check-Out-Funktion auf dem LicenseServer aktiviert werden. Wenn diese Funktion nicht aktiviert wurde, erhalten Sie eine entsprechende Fehlermeldung, wenn Sie versuchen die Lizenz auszuchecken. Wenden Sie sich in diesem Fall an Ihren LicenseServer-Administrator.

Support-Code kopieren
Klicken Sie auf **Support-Code kopieren,** um Lizenzinformationen in die Zwischenablage zu kopieren. Dies sind die Daten, die Sie bei einer Support-Anfrage über das Online Support-Formular benötigen.

Altova LicenseServer bietet IT-Administratoren einen Echtzeitüberblick über alle Altova-Lizenzen in einem Netzwerk. Dazu werden die Einzelheiten zu jeder Lizenz sowie Client-Zuweisungen und die Verwendung von Lizenzen durch Clients angezeigt. Der Vorteil der Verwendung von LicenseServer liegt in seinen Funktionen zur Verwaltung großer Altova-Lizenzpools. Altova LicenseServer steht kostenlos auf der Altova Website zur Verfügung. Nähere Informationen zu Altova LicenseServer und der Lizenzierung mittels Altova LicenseServer finden Sie in der Dokumentation zu Altova LicenseServer.

- Bestellformular

  Sobald Sie eine lizenzierte Version des Software-Produkts bestellen möchten, klicken Sie im Dialogfeld **Software-Aktivierung** (*siehe oben*) auf die Schaltfläche **Permanenten Key-Code erwerben...** oder wählen Sie den Befehl **Bestellformular**, um zum sicheren Online-Shop von Altova weitergeleitet zu werden.

- Registrierung

  Bei Aufruf dieses Befehls wird die Altova-Produktregistrierungsseite auf einem Register Ihres Browsers geöffnet. Durch Registrierung Ihrer Altova-Software stellen Sie sicher, dass Sie immer die neuesten Produktinformationen erhalten.

- Auf Updates überprüfen

  Überprüft, ob am Altova Server eine neuere Version Ihres Produkts vorhanden ist und zeigt eine entsprechende Meldung an.

- Support Center

  Der Befehl "Support Center" ist ein Link zum Altova Support Center im Internet. Im Support Center finden Sie Antworten auf häufig gestellt Fragen, Diskussionsforen, in denen Sie Software-Probleme besprechen können und ein Formular, um unsere Mitarbeiter vom technischen Support zu kontaktieren.

- Fragen und Antworten im Web

  Der Befehl "Fragen und Antworten im Web" ist ein Link zur Altova-Fragen- und Antworten-Datenbank im Internet. Diese Datenbank wird ständig aktualisiert, wenn die Mitarbeiter von Altova von unseren Kunden mit neuen Fragen konfrontiert werden.

- Komponenten und Gratistools downloaden

  Dieser Befehl ist ein Link zum Komponenten Download Center von Altova im Internet. Von hier können Sie Software-Komponenten verschiedener anderer Anbieter herunterladen, die Sie mit Altova Produkten verwenden können. Dabei handelt es sich um XSLT- und XSL-FO-Prozessoren, Applikationsserverplattformen usw. Die im Komponenten Download Center verfügbare Software ist normalerweise kostenlos.

- MapForce im Internet

  Der Befehl MapForce im Internet ist ein Link zur Altova Website im Internet. Hier erfahren Sie mehr über MapForce und verwandte Technologien und Produkte auf der Altova Website.

- MapForce Training

  Ein Link zur Online Training-Seite der Altova Website. Hier können Sie zwischen den von Altova-Experten gehaltenen Online-Kursen wählen.

- Über MapForce

  Mit dem Befehl Über MapForce wird das Willkommensfenster und die Versionsnummer Ihres Produkts angezeigt. Wenn Sie die 64-Bit-Verson von MapForce verwenden, wird dies durch das Suffix (x64) nach dem Applikationsnamen angezeigt. Die 32-Bit-Version hat kein Suffix.

© 2016-2022 Altova GmbH

16 Code Generator

MapForce enthält einen integrierten Code Generator, der automatisch Java-, C++ oder C#-Klassendateien anhand von mit MapForce erstellten Mapping-Designs generieren kann. Dabei beschränkt MapForce das Mappen nicht auf 1:1-Beziehungen, sondern gestattet die gleichzeitige Verwendung mehrerer Quellen und Ziele. Bei der Code Generierung wird eine voll funktionsfähige und vollständige Applikation generiert, die das Mapping für Sie durchführt. Nachdem Sie den Code generiert haben, können Sie das Mapping ausführen, indem Sie die Applikation, so wie sie generiert wurde, direkt ausführen. Sie können den generierten Code auch in Ihre eigene Applikation importieren oder ihn durch Ihre eigenen Funktionen ergänzen.

Hauptaufgabe des generierten Codes ist die Ausführung eines MapForce Mappings. Wenn in Ihrem Mapping XML-Schemas oder DTDs verwendet werden, haben Sie zusätzlich dazu die Möglichkeit, Schema Wrapper-Bibliotheken für im Mapping verwendete XML-Schemas/DTDs zu generieren, wodurch Daten von/aus XML-Instanzen geschrieben oder gelesen werden können. Mit Hilfe von Schema Wrapper-Bibliotheken können Sie programmatisch mit XML-Daten arbeiten und dabei anhand des Schemas generierte Typen verwenden, ohne sich zu sehr um die zugrunde liegende XML API kümmern zu müssen. Nähere Informationen finden Sie unter Generieren von Code anhand von XML-Schemas oder DTDs 1607.

Der erzeugte Programmcode wird in den Programmiersprachen C++, Java oder C# ausgegeben.

Zielsprache	C++	C#	Java
Entwicklungsumgebungen	Microsoft Visual Studio 2013, 2015, 2017, 2019, 2022	Microsoft Visual Studio 2013, 2015, 2017, 2019, 2022 Ziel-Frameworks: • .NET Framework • .NET Core 3.1 • .NET 5.0 • .NET 6.0	Java SE JDK 8 oder 11 (einschließlich OpenJDK) Eclipse 4.4 oder höher Apache Ant (build.xml-Datei)
XML DOM Implementierungen	MSXML 6.0 Apache Xerces 3	System.Xml	JAXP
Datenbank API	ADO	ADO.NET	JDBC

Anmerkung: Der mit MapForce generierte Code gilt nur dann als threadsicher, wenn es die zugrunde liegenden Drittanbieter-XML DOM und Datenbank API-Bibliotheken sind. Zwar kann die Threadsicherheit des generierten Code nicht wirklich garantiert oder gewährleistet werden, doch lassen sich wahrscheinlich in den meisten Fällen mehrere gleichzeitige Instanzen des Mapping-Code erfolgreich ausführen.

C++

Sie können konfigurieren, ob der generierte C++-Ausgabecode MSXML 6.0 oder Apache Xerces 3 verwenden soll. Bei der C++-Codegenerierung werden in MapForce komplette Projekt- (.vcproj) und Solution- (.sln)-Dateien

für alle unterstützten Versionen von Visual Studio (siehe Tabelle oben) generiert. Der generierte Code unterstützt optional MFC.

Voraussetzungen:

1. Für die Kompilierung des generierten C++-Codes muss Windows SDK auf Ihrem Rechner installiert sein.
2. Um Xerces 3 für C++ verwenden zu können, müssen Sie es nach folgender Anleitung installieren und bauen: http://xerces.apache.org/xerces-c/. Dabei muss die Umgebungsvariable XERCES3, die auf das Verzeichnis verweist, in dem Xerces installiert wurde, hinzugefügt werden, z.B: `C:\xerces-c-3.2.2`. Außerdem muss die Umgebungsvariable PATH den Pfad inkludieren, unter dem sich die Xerces-Binärdateien befinden, z.B: `%XERCES3%\bin`.
3. Beim Erstellen von C++-Code für Visual Studio und bei Verwendung einer für Visual C++ vorkompilierten Xerces-Bibliothek muss eine Compiler-Einstellung in allen Projekten der Applikation geändert werden:

   a) Wählen Sie alle Projekte im Solution Explorer aus.
   b) Klicken Sie im Menü **Projekt** auf **Eigenschaften**.
   c) Klicken Sie auf **Konfigurationseigenschaften | C/C++ | Sprache**.
   d) Wählen Sie in der Liste der Konfigurationen *Alle Konfigurationen*.
   e) Ändern Sie **Treat wchar_t as Built-in Type** in **No (/Zc:wchar_t-)**.

C#

Der generierte C#-Code kann von jeder .NET-fähigen Programmiersprache verwendet werden, z.B. VB.NET, Managed C++ oder J#. Projektdateien können für alle unterstützten Versionen von Visual Studio generiert werden (siehe Tabelle oben).

Java

Der generierte Java-Code wird anhand der "Java API for XML Processing" (JAXP) geschrieben und enthält eine Ant build-Datei sowie Projektdateien für die unterstützten Versionen von Java und Eclipse (siehe Tabelle oben).

Generierte Ausgabe

Der Zielordner für den generierten Code enthält alle zum Ausführen des Mappings erforderlichen Bibliotheken und Dateien. Es sind dies die folgenden:

- Eine variable Anzahl an für das Mapping erforderlichen Altova-Bibliotheken (z.B. Altova-Funktionsbibliotheken, Datenbankbibliotheken, EDI-Bibliotheken)
- Eine komplette Mapping-Applikation. Wenn diese kompiliert und ausgeführt wird, führt sie die Mapping-Transformation durch.

Code Generator-Vorlagen

Der generierte Code unterstützt die Anpassung mittels einer Vorlagensprache[1703] namens SPL (Spy Programming Language). Dies ist unter Umständen hilfreich, wenn Sie Code z.B. gemäß den Codekonventionen Ihres Unternehmens anpassen möchten oder bestimmte Bibliotheken im generierten Code ersetzen möchten.

© 2016-2022 Altova GmbH

16.1 Was ist neu ...

Version 2022 R2

- Unterstützung für Visual Studio 2022.
- Unterstützung für NET 6.0.

Version 2021 R2

- Bei der Generierung von C#-Code kann in den Codegenerierungsoptionen[1701] als Ziel-Framework (zusätzlich zur bestehenden Unterstützung für .NET-Framework-Projekte) .NET Core 3.1, NET 5.0 oder .NET 6.0 ausgewählt werden.

Version 2020 R2

- Für XML-Schema-Wrapper-Bibliotheken generierter Code erlaubt nun mehr Kontrolle über Element-Namespaces und -Präfixe. Zum Deklarieren oder Außerkraftsetzen von Namespaces für ein Element oder zum Anhängen eines Elements mit einem vorangestellten Namespace stehen neue Methoden zur Verfügung. Siehe Beispiel: Bestellung[1644].

Version 2020

- Unterstützung für Visual Studio 2019
- Ende der Unterstützung für Visual Studio 2008

Version 2018

- Unterstützung für Microsoft Visual Studio 2013, 2015, 2017
- Ende der Unterstützung für Visual Studio 2005 und Xerces 2.x

Version 2014

- Kompatibilitätsmodus-Option für die Codegenerierung wurde entfernt

Version 2011

- Enthält Bug Fixes und Verbesserungen

Version 2010 R3

- Unterstützung für Microsoft Visual Studio 2010
- Unterstützung für MSXML 6.0 im generierten C++-Code
- Unterstützung für 64-Bit-Zielkomponenten für C++- und C#-Projekte

Version 2010

- Enumeration Facets aus XML-Schemas stehen nun als Symbolkonstanten in den generierten Klassen zur Verfügung (bei Verwendung von 2007r3 Vorlagen).

Version 2009 sp1

- Wurde um Apache Xerces Version 3.x Unterstützung erweitert. (auch die älteren Versionen ab Xerces 2.6.x werden weiterhin unterstützt.)

Version 2009

- Die generierte Mapping-Implementierung wurde überarbeitet und unterstützt nun Sequenzen und Gruppierungen. Die API hat sich nicht geändert..

Version 2008 R2

- Unterstützung für die Generierung von Visual Studio 2008 Projektdateien für C# und C++.
- Generierter MapForce Mapping Code in C# und Java kann Readers/Writers, Streams, Strings oder DOM-Dokumente als Quellen und Ziele verwenden.

Version 2008

- Die neuen 2007 SPL-Vorlagen im R3-Stil wurden weiter verbessert:
  - Es können nun einzelne Elemente entfernt werden.
  - Sie haben Zugriff auf Schema-Metadaten (z.B. Elementnamen, Facets, Enumerationen, Occurrence usw.).
  - ComplexType, die durch Erweiterungen abgeleitet werden, werden nun als abgeleitete Klassen generiert.

Version 2007 R3

Der Code Generator wurde in Version 2007 Release 3 zur Vereinfachung des generierten Codes, der Verringerung des Codevolumens und zur Verbesserung der Leistung überarbeitet.

- Behandlung von XML-Dokumenten und Nodes mit expliziter Ownership, um Probleme mit Speicherverwaltung und Multithreading zu vermeiden.
- Neue Syntax zur Vermeidung von Kollisionen
- Neue Datentypen zur einfacheren Verwendung und Leistungsoptimierung (wenn möglich "Native Types", neue Behandlung von Null, ...)
- Attribute werden nicht mehr als Sammlungen generiert.
- Der Inhalt von Simple Elements wird nun aus Konsistenzgründen ebenfalls wie ein spezielles Attribut behandelt.
- Neues internes Objektmodell (wichtig für benutzerdefinierte SPL-Vorlagen)
- Kompatibilitätsmodus zur Generierung von Code im Stile älterer Versionen
- Type Wrapper-Klassen werden nun nur mehr auf Anweisung für kleinere Codefragmente generiert.

© 2016-2022 Altova GmbH

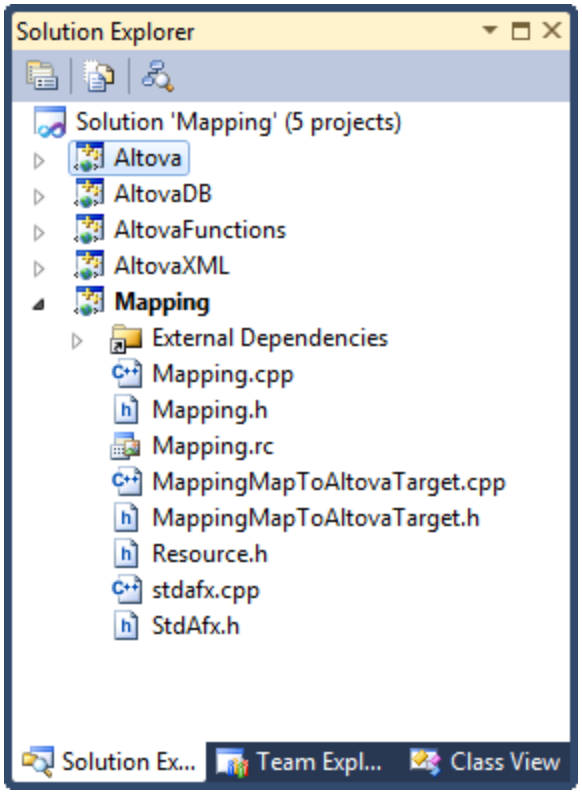
16.2 Generieren von C++-Code

Sie können C++-Code für Visual Studio 2013, 2015, 2017, 2019, 2022 generieren. Der generierte Code enthält **.sln**- und .**vcproj**-Dateien für Visual Studio. Beachten Sie bei der Codegenerierung die folgenden Punkte:

- Bestimmte MapForce-Funktionen werden im generierten Programmcode nicht unterstützt. Nähere Informationen dazu finden Sie unter Unterstützte Funktionalitäten im generierten Code[1984].
- Sie können Code entweder anhand eines einzigen Mapping-Designs (.mfd) oder anhand eines Mapping-Projekts (.mfp) generieren. Wenn Sie Code anhand eines einzigen Mappings generieren, führt die erzeugte Applikation die jeweilige Mapping-Transformation aus. Wenn Sie Code anhand eines MapForce-Projekts (.mfp), das mehrere Mappings enthält, generieren, führt die erzeugte Applikation alle im Projekt enthaltenen Mappings in einer Bulk-Operation aus.
- Sie können die allgemeinen Codegenerierungsoptionen über das Register **Code-Generierung** des Menüs **Extras | Optionen** ändern, siehe Code Generator-Optionen[1701].
- Sie können den Namen des generierten Mappings sowie andere Einstellungen über das Menü **Datei | Mapping-Einstellungen** ändern. Der Standardapplikationsname lautet **Mapping**. Siehe auch Ändern der Mapping-Einstellungen[116].
- Informationen zur Behandlung von Pfaden im generierten Code finden Sie unter Pfade in verschiedenen Ausführungsumgebungen[87].
- Wenn Ihr Mapping Datenbankkomponenten enthält, können Sie datenbankspezifische Einstellungen anzeigen, indem Sie auf eine Datenbankkomponente klicken und anschließend die Menüoption **Komponente | Eigenschaften** auswählen, siehe Datenbank-Komponenteneinstellungen[286].
- Die Unterstützung für Datenbankverbindungen variiert je nach Plattform. Es gibt Verbindungsarten, die nicht auf allen Plattformen unterstützt werden. Wenn Sie in Ihrem Mapping eine Verbindung zu einer Datenbank herstellen, wählen Sie eine Datenbankverbindung, die mit der Zielumgebung, für die Sie Code generieren, kompatibel ist. Nähere Informationen dazu finden Sie unter Datenbankmappings in verschiedenen Ausführungsumgebungen[166].

Eine typische mit MapForce generierte C++-Lösung enthält die folgenden Komponenten:

- eine Reihe von für das Mapping erforderlichen Altova-signierten Bibliotheken (alle mit dem Präfix **Altova**).
- Das Hauptmapping-Projekt (in diesem Beispiel **Mapping**), das die Mapping-Applikation und die davon abhängigen Dateien enthält.



Mit MapForce generierte C++-Beispiellösung

Dieser Abschnitt enthält die folgenden Kapitel:

- Generieren von Code anhand eines Mappings 1568
- Generieren von Code anhand eines Mapping-Projekts 1569
- Erstellen des Projekts 1570
- Ausführen der Applikation 1570

16.2.1 Generieren von Code anhand eines Mappings

So generieren Sie C++-Code anhand einer Mapping-Design-Datei (.mfd):

1. Überprüfen Sie die gewünschten Codegenerierungsoptionen 1701 bzw. wählen Sie die gewünschten Optionen aus.
2. Klicken Sie im Menü **Datei** auf den Befehl **Code generieren in | C++**.
3. Wählen Sie ein Zielverzeichnis für die generierten Dateien aus und klicken Sie anschließend zur Bestätigung auf OK. Das Ergebnis der Codegenerierung (Fehler- oder Erfolgsmeldung) wird im Fenster "Meldungen" angezeigt.

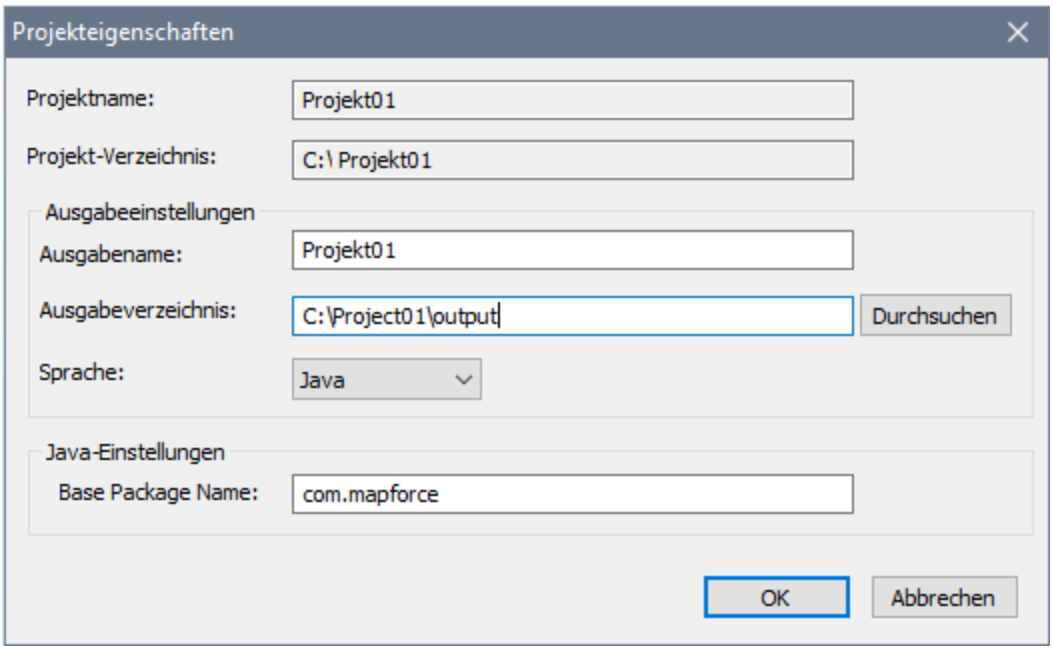
Der Standardname der generierten Applikation ist **Mapping**. Sie können den Namen sowie andere Einstellungen bei Bedarf im Dialogfeld Mapping-Einstellungen 116 ändern.

© 2016-2022 Altova GmbH

16.2.2 Generieren von Code anhand eines Mapping-Projekts

So generieren Sie Code anhand eines Mapping-Projekts (.mfp):

1. Öffnen Sie das Mapping-Projekt in MapForce, falls es noch nicht offen ist.
2. Klicken Sie mit der rechten Maustaste in das Fenster "Projekt" und wählen Sie anschließend den Befehl **Eigenschaften**.



3. Überprüfen Sie die Projekteinstellungen und ändern Sie diese bei Bedarf (stellen Sie v.a. sicher, dass die Zielsprache und das Ausgabeverzeichnis korrekt eingestellt wurden) und klicken Sie anschließend auf **OK**.
4. Wählen Sie im Menü **Projekt den Befehl Code für das gesamte Projekt generieren**.

Das Ergebnis der Codegenerierung (Fehler- oder Erfolgsmeldung) wird im Fenster "Meldungen" angezeigt.

Standardmäßig ist der Name der generierten Applikation mit dem Projektnamen identisch. Wenn der Projektname Leerzeichen enthält, werden diese im generierten Code in Unterstriche konvertiert. Standardmäßig wird Code im selben Verzeichnis wie das MapForce-Projekt generiert, nämlich im untergeordneten Ausgabeverzeichnis.

Um das Ausgabeverzeichnis und den Namen des Projekts zu ändern, klicken Sie im Projektfenster auf das Projekt und wählen Sie im Menü den Befehl **Projekt | Eigenschaften. Wenn Ihr MapForce-Projekt Ordner enthält, können Sie die Codegenerierungseinstellungen für die einzelnen Ordner (durch Rechtsklick auf den Ordner und Auswahl von Eigenschaften) ändern. Andernfalls erben alle Projektordner die Einstellungen vom MapForce-Projekt.**

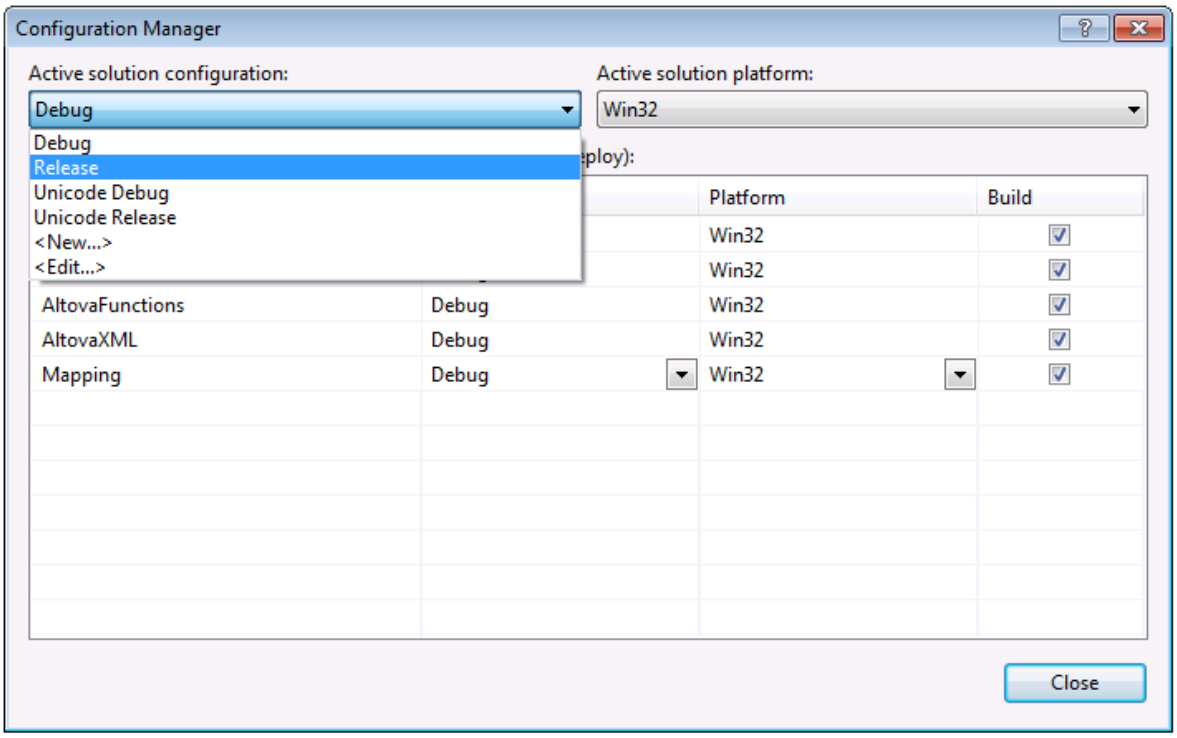
16.2.3 Erstellen des Projekts

Nachdem Sie den C++-Code generiert haben, müssen Sie als nächstes in Visual Studio den Build erstellen. Gehen Sie dazu folgendermaßen vor:

1. Öffnen Sie die generierte Lösungsdatei (.sln) in Visual Studio.

Standardmäßig lautet der Name der Lösungsdatei **Mapping.sln.** Die Datei befindet sich relativ zu dem Verzeichnis, in dem Sie den generierten Code gespeichert haben, im Unterverzeichnis **Mapping**. Wenn Sie den Applikationsnamen über die Mapping-Einstellungen geändert haben, so wird der Name der .sln-Datei entsprechend geändert. Wenn Sie z.B. den Applikationsnamen in **MyApplication** geändert haben, so erhält die Lösungsdatei den Namen **MyApplication.sln** und befindet sich im Unterverzeichnis **MyApplication**.

2. Klicken Sie im Menü **Build** auf **Configuration Manager**.



3. Wählen Sie die gewünschte Build-Konfiguration aus (Debug, Release, Unicode Debug, Unicode Release). Beachten Sie, dass nur Unicode Builds alle Unicode-Zeichen in XML und anderen Dateien unterstützen. In Nicht-Unicode-Builds wird die lokale Codepage Ihres Windows-Systems verwendet.
4. Klicken Sie im Menü **Build** auf **Build Solution**.

16.2.4 Ausführen der Applikation

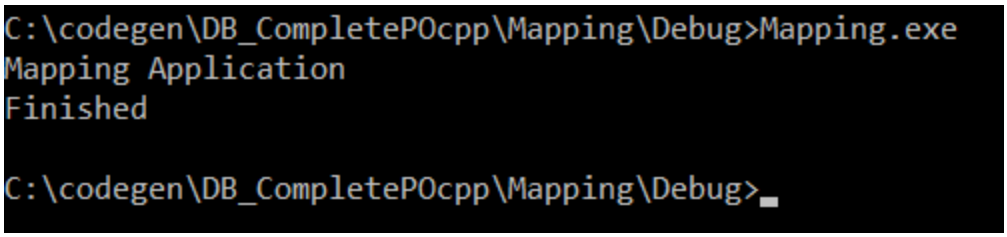
Sobald Sie das Visual Studio-Projekt kompiliert haben, wird eine Befehlszeilenapplikation namens **Mapping.exe**.erzeugt. (Beachten Sie, dass sich auch der Name der ausführbaren Datei ändert, wenn Sie den Applikationsnamen über die Mapping-Einstellungen geändert haben.)

© 2016-2022 Altova GmbH

Die erzeugte Mapping-Applikation befindet sich, je nach gewählter Build-Option, relativ zur .sln-Datei in einem der folgenden Unterverzeichnisse:

- Debug
- Release
- Unicode Debug
- Unicode Release

Um die Applikation auszuführen, öffnen Sie ein Eingabeaufforderungsfenster, ändern Sie das aktuelle Verzeichnis in den Pfad der ausführbaren Datei und führen Sie diese aus. z.B.:



16.3 Generieren von C#-Code

Sie können für jedes Mapping oder Mapping-Projekt, dessen Mapping-Komponenten in C# unterstützt werden, .NET C#-Code generieren. Eine Übersicht darüber, welche Komponenten in den einzelnen Sprachen einschließlich C# unterstützt werden, finden Sie unter Unterstützte Funktionalitäten im generierten Code[1984]. Beachten Sie, dass MapForce normalerweise Benachrichtigungen oder Warnmeldungen ausgibt, wenn versucht wird, Komponenten hinzuzufügen, die in einer bestimmten Sprache nicht unterstützt werden.

Sie können bei der Generierung von C#-Code nicht nur die .NET Framework-Plattform, sondern auch die .NET Core 3.1 NET 5.1- und die .NET 6.0-Plattform auswählen. In letzterem Fall ist das Ergebnis eine plattformübergreifende .NET (Core) Desktop-Applikation, die auf Windows, macOS und Linux ausgeführt werden kann.

Beachten Sie bei der Codegenerierung die folgenden Punkte:

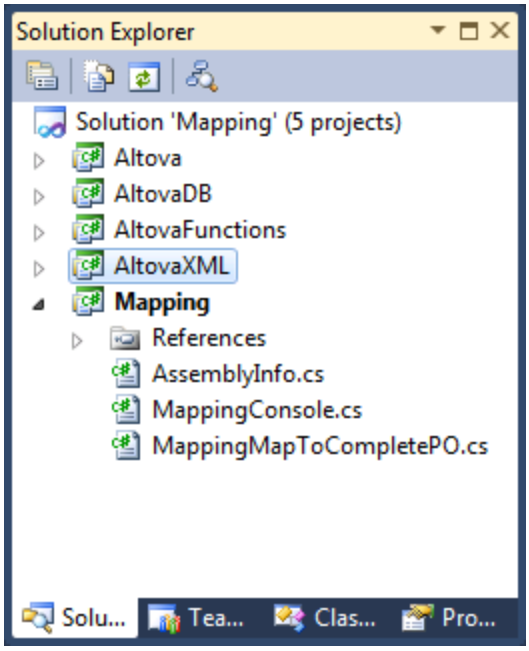
- Bestimmte MapForce-Funktionen werden im generierten Programmcode nicht unterstützt. Nähere Informationen dazu finden Sie unter Unterstützte Funktionalitäten im generierten Code[1984].
- Sie können Code entweder anhand eines einzigen Mapping-Designs (.mfd) oder anhand eines Mapping-Projekts (.mfp) generieren. Wenn Sie Code anhand eines einzigen Mappings generieren, führt die erzeugte Applikation die jeweilige Mapping-Transformation aus. Wenn Sie Code anhand eines MapForce-Projekts (.mfp), das mehrere Mappings enthält, generieren, führt die erzeugte Applikation alle im Projekt enthaltenen Mappings in einer Bulk-Operation aus.
- Sie können die allgemeinen Codegenerierungsoptionen über das Register **Code-Generierung** des Menüs **Extras | Optionen** ändern, siehe Code Generator-Optionen[1701].
- Sie können den Namen des generierten Mappings sowie andere Einstellungen über das Menü **Datei | Mapping-Einstellungen** ändern. Der Standardapplikationsname lautet **Mapping**. Siehe auch Ändern der Mapping-Einstellungen[116].
- Informationen zur Behandlung von Pfaden im generierten Code finden Sie unter Pfade in verschiedenen Ausführungsumgebungen[87].
- Wenn Ihr Mapping Datenbankkomponenten enthält, können Sie datenbankspezifische Einstellungen anzeigen, indem Sie auf eine Datenbankkomponente klicken und anschließend die Menüoption **Komponente | Eigenschaften** auswählen, siehe Datenbank-Komponenteneinstellungen[286].
- Die Unterstützung für Datenbankverbindungen variiert je nach Plattform. Es gibt Verbindungsarten, die nicht auf allen Plattformen unterstützt werden. Wenn Sie in Ihrem Mapping eine Verbindung zu einer Datenbank herstellen, wählen Sie eine Datenbankverbindung, die mit der Zielumgebung, für die Sie Code generieren, kompatibel ist. Nähere Informationen dazu finden Sie unter Datenbankmappings in verschiedenen Ausführungsumgebungen[166].

Wenn in Ihrem Mapping Datenbankkomponenten verwendet werden, stellen Sie sicher, dass Sie die Verbindung zur Datenbank mit Hilfe von Datenbanktreibern herstellen, die mit der Plattform, auf der das generierte Programm ausgeführt werden soll, kompatibel sind. So werden z.B. ADO-Datenbankverbindungen und einige ODBC-Treiber unter Linux nicht unterstützt.

Eine typische mit MapForce generierte C#-Lösung enthält die folgenden Komponenten:

- Projektmappen (.sln) und Projektdateien (.csproj), die in Visual Studio geöffnet werden können.
- eine Reihe von für das Mapping erforderlichen Altova-signierten Bibliotheken (alle mit dem Präfix **Altova**).
- Das Hauptmapping-Projekt (in diesem Beispiel **Mapping**), das die Mapping-Applikation und die davon abhängigen Dateien enthält.

© 2016-2022 Altova GmbH

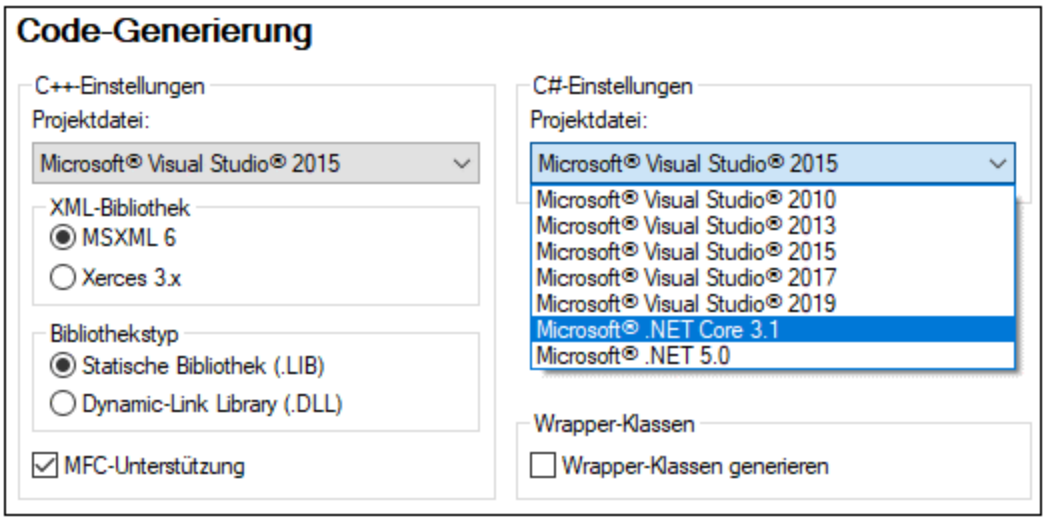


Mit MapForce generierte C#-Beispiellösung

16.3.1 Auswahl der Zielplattform

Bevor Sie C#-Code generieren, können Sie die Zielplattform auf Programmebene folgendermaßen definieren:

1. Klicken Sie im Menü **Extras** auf **Optionen.**
2. Klicken Sie auf das Register **Code-Generierung** .
3. Wählen Sie in der Gruppe "C#-Einstellungen" je nach Bedarf einen Wert aus der Dropdown-Liste aus.



Wählen Sie die Option **Microsoft .NET Core .NET 3.1**, **Microsoft .NET 5.0** oder **Microsoft .NET 6.0** aus, um eine Visual Studio-Projektmappe für die entsprechende Plattform zu generieren. Wenn Sie die **.NET**

Framework-Plattform für eine bestimmte Visual Studio-Version auswählen müssen, wählen Sie eine beliebige der **Microsoft Visual Studio 2010-2019**-Optionen aus - in diesem Fall ist das Ziel der generierten Projektmappe die .NET Framework-Version für die entsprechende Visual Studio-Version.

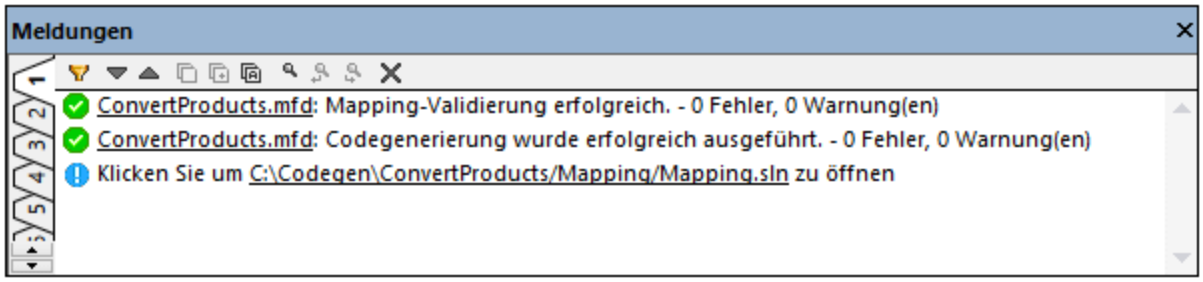
Wenn das Kontrollkästchen **Wrapper-Klassen generieren** aktiviert ist, wird im generierten Code für jedes XML-Schema bzw. für jede DTD-Komponente im Mapping ein zusätzliches Projekt erstellt. Mit Hilfe der generierten Wrapper-Projekte können Sie die entsprechenden XML-Instanzen außerhalb des Mappings programmatisch bearbeiten. Die Wrapper-Bibliotheken sind optional und unabhängig vom Mapping-Programm, können aber bei Bedarf vom Mapping-Programm aus verwendet werden. Nähere Informationen zu Wrapper-Klassen finden Sie unter Generieren von Code anhand von XML-Schemas oder DTDs or DTDs 1607.

16.3.2 Generieren von Code anhand eines Mappings

So generieren Sie C#-Code anhand einer Mapping-Design-Datei (.mfd):

1. Erstellen Sie ein neues Mapping oder öffnen Sie ein vorhandenes in MapForce, falls dies noch nicht geschehen ist.
2. Wählen Sie als Mapping-Transformationssprache C# aus. Klicken Sie dazu entweder auf die Symbolleistenschaltfläche C# oder wählen Sie den Menübefehl **Ausgabe | C# (Sharp)**.
3. Definieren oder ändern Sie die C#-Codegenerierungsoptionen, wie unter Auswahl der Zielplattform 1573 beschrieben.
4. Wählen Sie den Menübefehl **Datei | Code generieren in | C# (Sharp)**. Sie werden nun aufgefordert, ein Zielverzeichnis auszuwählen.
5. Gehen Sie zum gewünschten Verzeichnis (z.B. "C:\codegen\ConvertProducts") und klicken Sie auf **OK**.

Das Ergebnis der Codegenerierung wird im Fenster "Meldungen" angezeigt, z.B:

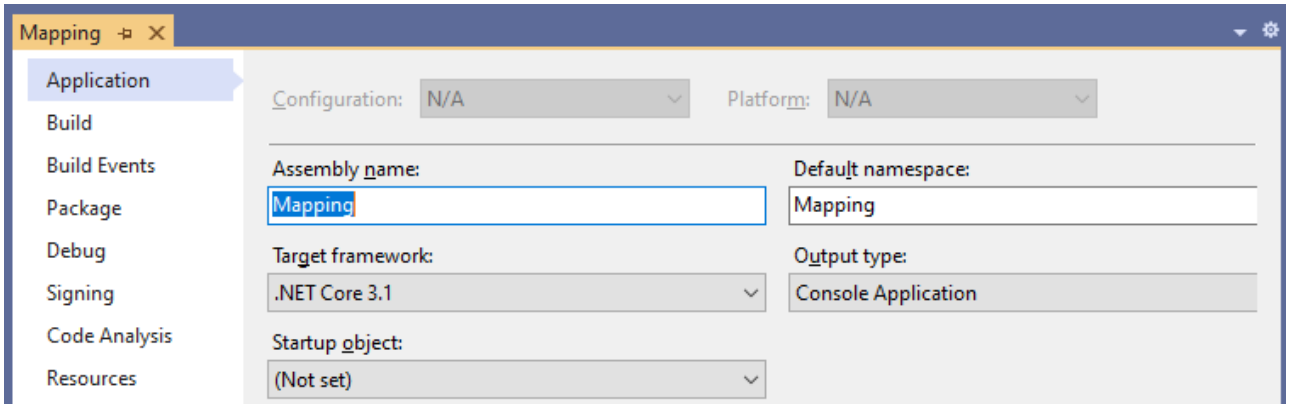


Klicken Sie optional auf den Link im Fenster "Meldungen", um die generierte Projektmappe in Visual Studio zu öffnen.

Beachten Sie, dass Sie die entsprechenden SDKs und eine kompatible Visual Studio-Version benötigen, wenn Sie Code für die Plattformen .NET 5.0, .NET 6,0 oder .NET Core generieren möchten. Informationen dazu, welches Download-Paket Sie für Ihr Betriebssystem und Ihre Plattform benötigen, finden Sie auf der Microsoft Website (https://dotnet.microsoft.com/download).

Um das Ziel-Framework von Visual Studio aus noch einmal zu überprüfen, klicken Sie im Solution Explorer mit der rechten Maustaste auf das "Mapping"-Projekt, wählen Sie **Properties** und kontrollieren Sie den Wert, der in der **Target Framework** Dropdown-Liste angezeigt wird, z.B:

© 2016-2022 Altova GmbH



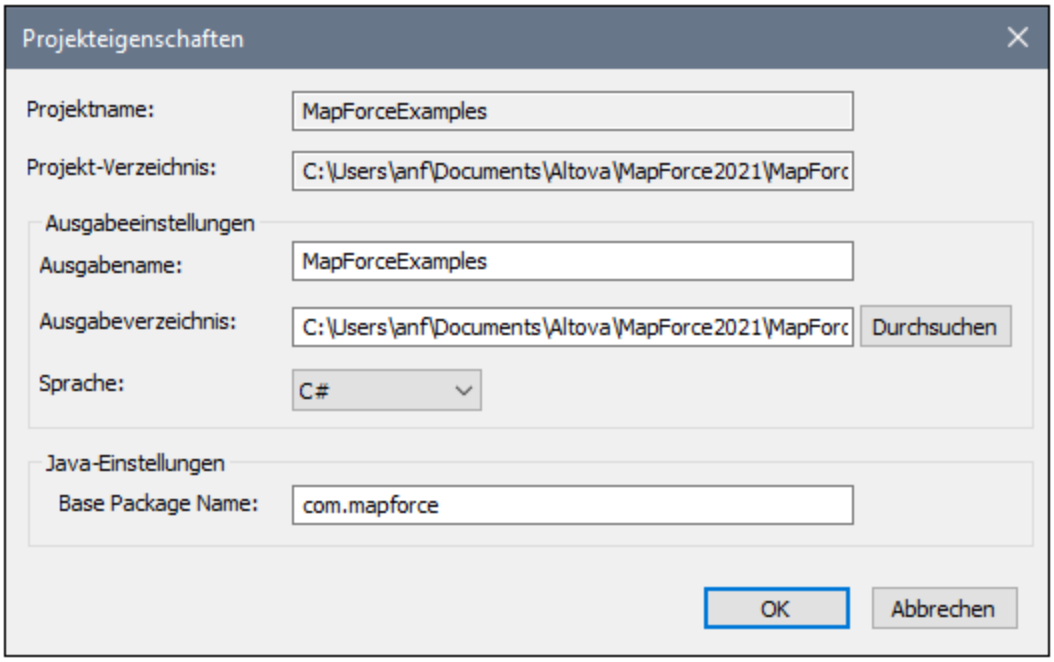
Der Standardname der generierten Applikation ist **Mapping**. Sie können den Namen sowie andere Einstellungen bei Bedarf im Dialogfeld Mapping-Einstellungen[116] ändern, bevor Sie Code generieren.

16.3.3 Generieren von Code anhand eines Mapping-Projekts

Sie können anhand eines Mapping-Projekts (.mfp), das aus mehreren Mapping-Design-Dateien (.mfd) besteht, C#-Code generieren. Beachten Sie, dass alle Mapping-Design-Dateien im Projekt für die C#-Codegenerierung geeignet sein müssen, d.h. alle ihre Komponenten müssen in C# unterstützt werden, wie unter Unterstützte Funktionalitäten im generierten Code[1984] beschrieben.

So generieren Sie Code anhand eines Mapping-Projekts (.mfp):

1. Öffnen Sie das Mapping-Projekt in MapForce, falls es noch nicht offen ist.
2. Klicken Sie mit der rechten Maustaste im Projektfenster auf das Projekt und klicken Sie auf **Eigenschaften**.



3. Überprüfen Sie die Projekteinstellungen und ändern Sie diese bei Bedarf (stellen Sie v.a. sicher, dass die Zielsprache und das Ausgabeverzeichnis korrekt eingestellt wurden) und klicken Sie anschließend auf **OK**.
4. Wählen Sie im Menü **Projekt** den Befehl **Code für das gesamte Projekt generieren**.

Der Fortschritt und das Ergebnis der Codegenerierung (Fehler- oder Erfolgsmeldung) werden im Fenster "Meldungen" angezeigt.

Standardmäßig ist der Name der generierten Applikation mit dem Projektnamen identisch. Wenn der Projektname Leerzeichen enthält, werden diese im generierten Code in Unterstriche konvertiert. Standardmäßig wird Code im selben Verzeichnis wie das MapForce-Projekt generiert, nämlich im untergeordneten **Ausgabeverzeichnis**.

Um das Ausgabeverzeichnis und den Namen des Projekts zu ändern, klicken Sie im Projektfenster auf das Projekt und wählen Sie im Menü den Befehl **Projekt | Eigenschaften**. Wenn Ihr MapForce-Projekt Ordner enthält, können Sie die Codegenerierungseinstellungen für die einzelnen Ordner (durch Rechtsklick auf den Ordner und Auswahl von **Eigenschaften**) ändern. Andernfalls erben alle Projektordner die Einstellungen vom MapForce-Projekt.

16.3.4 Erstellen von .NET Framework-Projekten

Wenn Ihr C#-Code für die .NET Framework-Plattform generiert wurde, können Sie das Projekt am einfachsten folgendermaßen erstellen:

1. Öffnen Sie die generierte Projektmappendatei (.sln) in Visual Studio.

Standardmäßig lautet der Name der Lösungsdatei **Mapping.sln.** Die Datei befindet sich relativ zu dem Verzeichnis, in dem Sie den generierten Code gespeichert haben, im Unterverzeichnis **Mapping**. Wenn Sie den Applikationsnamen über die Mapping-Einstellungen geändert haben, so wird der Name der .sln-Datei

© 2016-2022 Altova GmbH

entsprechend geändert. Wenn Sie z.B. den Applikationsnamen in **MyApplication** geändert haben, so erhält die Lösungsdatei den Namen **MyApplication.sln** und befindet sich im Unterverzeichnis **MyApplication**.

2. Klicken Sie im Menü **Build** auf **Configuration Manager**.
3. Wählen Sie die gewünschte Build-Konfiguration aus (Debug, Release).
4. Klicken Sie im Menü **Build** auf **Build Solution**.

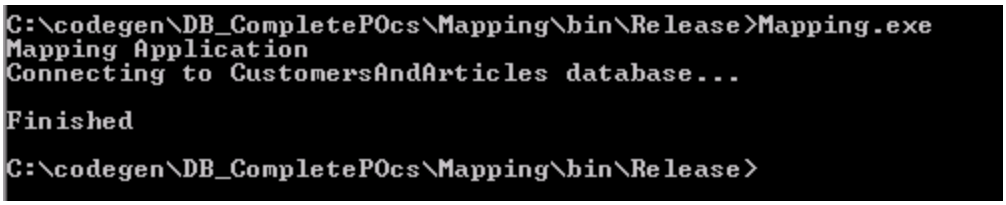
Ausführen der Applikation

Sobald Sie das Visual Studio-Projekt kompiliert haben, wird eine Befehlszeilenapplikation namens **Mapping.exe**erzeugt. Beachten Sie, dass sich auch der Name der ausführbaren Datei ändert, wenn Sie den Applikationsnamen über das Dialogfeld Mapping-Einstellungen[116] geändert haben.

Die erzeugte Mapping-Applikation befindet sich, je nach gewählter Build-Option, relativ zur .sln-Datei in einem der folgenden Unterverzeichnisse:

- bin\Debug
- bin\Release

Um die Applikation auszuführen, öffnen Sie ein Eingabeaufforderungsfenster, ändern Sie das aktuelle Verzeichnis in den Pfad der ausführbaren Datei und führen Sie diese aus. z.B.:



16.3.5 Erstellen von .NET- und .NET Core-Projekten

Um die generierte .NET Core 3.1, NET 5.0- oder .NET 6.0-Projektmappe erstellen und ausführen zu können, muss das entsprechende SDK installiert sein. Informationen dazu, welches Download-Paket Sie für Ihr Betriebssystem und Ihre Plattform benötigen, finden Sie auf der Microsoft Website (https://dotnet.microsoft.com/download). Nachdem Sie das entsprechende Paket heruntergeladen und installiert haben, können Sie die installierten SDKs einfach durch Eingabe des folgenden Befehls an der Eingabeaufforderung anzeigen.

```
dotnet --list-sdks
```

Wenn Sie das .NET Core .NET 3.1, .NET 5.0 oder .NET 6.0 SDK installiert haben, können Sie entweder über die Befehlszeile oder in Visual Studio mit dem Bauen des generierten Codes beginnen. Die .NET Core 3.1, .NET 5.0, .NET 6.0 SDKs sind mit Visual Studio 2019 (v16.7) kompatibel.

Erstellung über die Befehlszeile

So bauen Sie den generierten Code über die Befehlszeile:

1. Öffnen Sie ein Eingabeaufforderungsfenster und wechseln Sie in das Verzeichnis, in dem der Mapping-Code generiert wurde. Wenn Sie z.B. **C:\codegen\ConvertProducts** als Zielverzeichnis und die Standardmapping-Einstellungen ausgewählt haben, führen Sie folgenden Befehl aus:

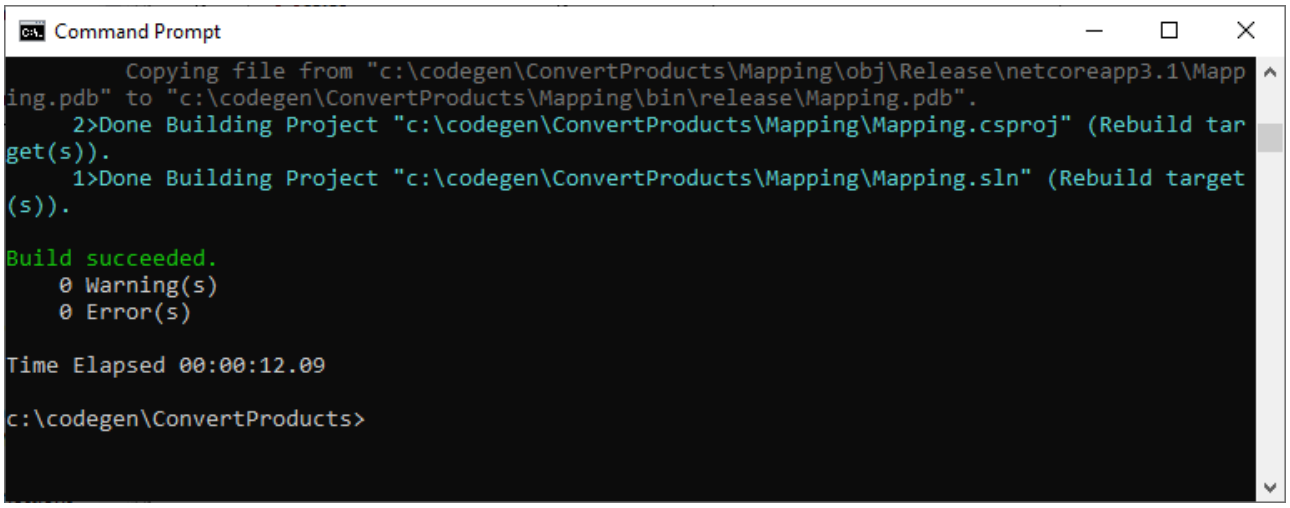
```
cd C:\codegen\ConvertProducts
```

2. Führen Sie den folgenden Befehl aus:

```
dotnet build Mapping\Mapping.sln --no-incremental --configuration Release --
verbosity normal --output Mapping\bin\release
```

Anmerkung: Eventuell lauten der Pfad zur Mapping-Lösung und ihr Name anders, wenn Sie im Dialogfeld Mapping-Einstellungen[116] die Eigenschaft **Applikationsname** geändert haben.

Das Resultat eines erfolgreichen Builds könnte ähnlich wie das unten gezeigte aussehen:



Erstellung mit Visual Studio

So bauen Sie den generierten Code in Visual Studio:

1. Doppelklicken Sie auf **Mapping.sln** (oder den entsprechenden Projektmappennamen), um die Mappe in Visual Studio zu öffnen.
2. Klicken Sie im Menü **Build** auf **Configuration Manager**.
3. Wählen Sie die gewünschte Build-Konfiguration (Debug, Release) und Projektmappenplattform (x64, x86, Any CPU) aus.
4. Klicken Sie im Menü **Build** auf **Build Solution (Strg+Umschalt+B)**.

Ausführen des generierten Programms

Nachdem Sie den generierten Code mit Build erstellt haben, wird im Ausgabeverzeichnis des Projekts eine **Mapping.exe**-Datei (oder ein Programm mit benutzerdefiniertem Namen, falls konfiguriert) erstellt. Der Pfad zum Ausgabeverzeichnis ist von der zuvor ausgewählten Build-Konfiguration (Debug, Release) abhängig.

© 2016-2022 Altova GmbH

Wenn Sie den generierten Code in Linux bauen, erhält die generierte ausführbare Datei einfach den Namen **Mapping** ohne Erweiterung.

Unter Windows können Sie das Programm durch Doppelklick auf die ausführbare Datei in Windows Explorer oder durch Aufruf der ausführbaren Datei über ein Eingabeaufforderungsfenster ausführen. Auf anderen Plattformen müssen Sie die ausführbare Datei mit einem Befehl wie dem folgenden über den Terminal aufrufen:

```
./Mapping
```

Beachten Sie Folgendes: Wenn das Mapping Referenzen zu Input- oder Output-Dateien auf der Festplatte aufweist, kann das Programm nur dann erfolgreich ausgeführt werden, wenn alle Pfade aufgelöst werden können. Wenn Sie das Mapping z.B. so konfiguriert haben, dass eine Komponente über einen relativen Pfad nach einer Input-Datei im selben Verzeichnis suchen soll, dann muss die entsprechende Datei im Ausführungsverzeichnis des Programms vorhanden sein. Informationen zur Behandlung von Pfaden zum Zeitpunkt des Mapping-Designs finden Sie unter Verwendung relativer und absoluter Pfade 84.

16.4 Generieren von Java-Code

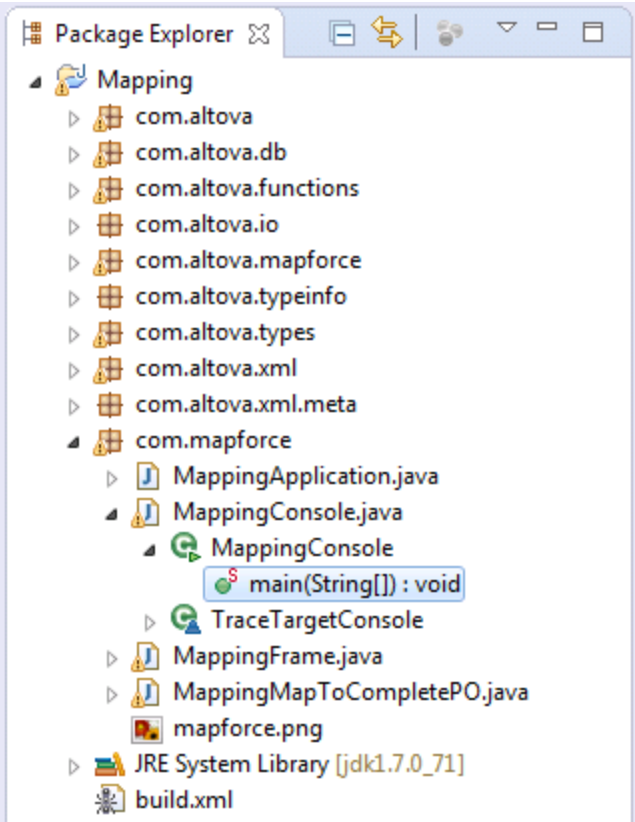
Sie können Programmcode für Java SE JDK 8 oder 11 (einschließlich OpenJDK) generieren. Beachten Sie dabei die folgenden Punkte:

- Bestimmte MapForce-Funktionen werden im generierten Programmcode nicht unterstützt. Nähere Informationen dazu finden Sie unter Unterstützte Funktionalitäten im generierten Code 1984.
- Sie können Code entweder anhand eines einzigen Mapping-Designs (.mfd) oder anhand eines Mapping-Projekts (.mfp) generieren. Wenn Sie Code anhand eines einzigen Mappings generieren, führt die erzeugte Applikation die jeweilige Mapping-Transformation aus. Wenn Sie Code anhand eines MapForce-Projekts (.mfp), das mehrere Mappings enthält, generieren, führt die erzeugte Applikation alle im Projekt enthaltenen Mappings in einer Bulk-Operation aus.
- Sie können die allgemeinen Codegenerierungsoptionen über das Register **Code-Generierung** des Menüs **Extras | Optionen** ändern, siehe Code Generator-Optionen 1701.
- Sie können den Namen des generierten Mappings sowie andere Einstellungen über das Menü **Datei | Mapping-Einstellungen** ändern. Der Standardapplikationsname lautet **Mapping**. Siehe auch Ändern der Mapping-Einstellungen 116.
- Informationen zur Behandlung von Pfaden im generierten Code finden Sie unter Pfade in verschiedenen Ausführungsumgebungen 87.
- Wenn Ihr Mapping Datenbankkomponenten enthält, können Sie datenbankspezifische Einstellungen anzeigen, indem Sie auf eine Datenbankkomponente klicken und anschließend die Menüoption **Komponente | Eigenschaften** auswählen, siehe Datenbank-Komponenteneinstellungen 286.
- Die Unterstützung für Datenbankverbindungen variiert je nach Plattform. Es gibt Verbindungsarten, die nicht auf allen Plattformen unterstützt werden. Wenn Sie in Ihrem Mapping eine Verbindung zu einer Datenbank herstellen, wählen Sie eine Datenbankverbindung, die mit der Zielumgebung, für die Sie Code generieren, kompatibel ist. Nähere Informationen dazu finden Sie unter Datenbankmappings in verschiedenen Ausführungsumgebungen 166.

Eine typische mit MapForce generierte Java-Lösung enthält die folgenden Komponenten:

- eine Reihe von für das Mapping erforderlichen Altova-signierten Java-Bibliotheken (alle mit dem Präfix **com.Altova**).
- das **com.mapforce**-Paket, das die Mapping-Applikation und davon abhängige Dateien enthält (wie unten gezeigt, können die Namen dieses Pakets geändert werden). Die beiden wichtigsten Dateien in diesem Paket sind die folgenden:
  - die Java Mapping-Applikation als Dialog-Applikation (**MappingApplication.java**).
  - die Java Mapping-Applikation als Konsolenapplikation (**MappingConsole.java**).
- eine **build.xml**-Datei, die Sie mit Apache Ant ausführen können, um das Projekt zu kompilieren und JAR-Dateien zu generieren.

© 2016-2022 Altova GmbH



Mit MapForce generierte Java-Beispielapplikation (Eclipse IDE)

Dieser Abschnitt enthält die folgenden Kapitel:

- Generieren von Code anhand eines Mappings 1581
- Generieren von Code anhand eines Mapping-Projekts 1582
- Behandlung von JDBC-Referenzen 1584
- Erstellen des Projekts mit Ant 1583
- Beispiel: Generieren und Ausführen von Java-Code 1586

16.4.1 Generieren von Code anhand eines Mappings

So generieren Sie Java-Code anhand einer Mapping-Design-Datei (.mfd):

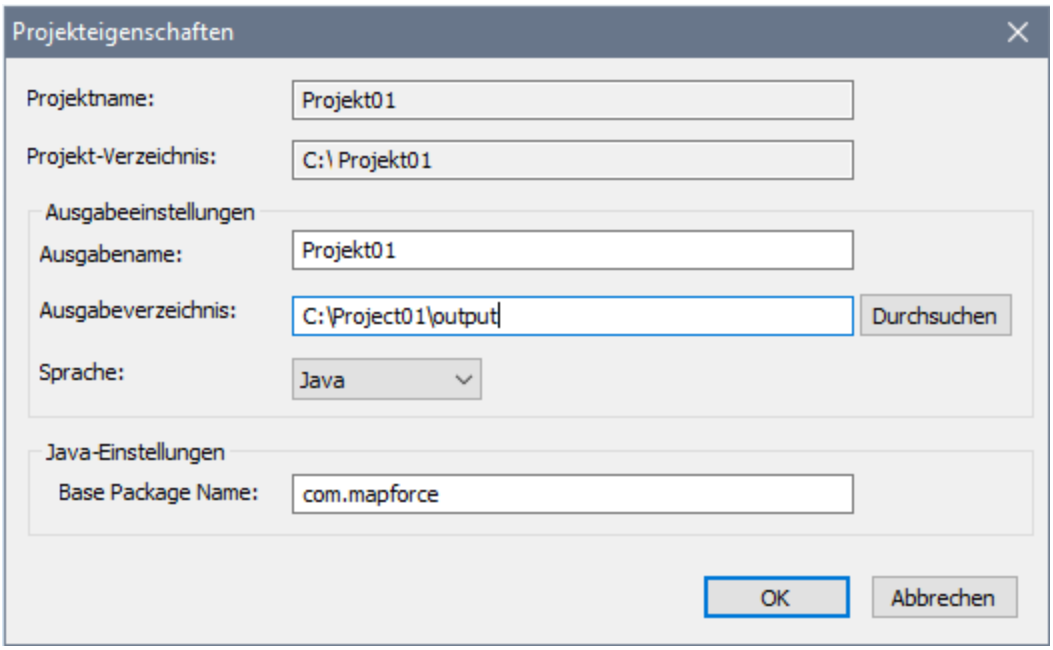
1. Überprüfen Sie die gewünschten Codegenerierungsoptionen 1701 bzw. wählen Sie die gewünschten Optionen aus.
2. Klicken Sie im Menü **Datei** auf den Befehl **Code generieren in | Java**.
3. Wählen Sie ein Zielverzeichnis für die generierten Dateien aus und klicken Sie anschließend zur Bestätigung auf **OK**. Das Ergebnis der Codegenerierung (Fehler- oder Erfolgsmeldung) wird im Fenster "Meldungen" angezeigt.

Der Standardname der generierten Applikation ist **Mapping** und der Standardname des Base Package ist **com.mapforce**. Sie können diese Namen bei Bedarf im Dialogfeld Mapping-Einstellungen 116 ändern.

16.4.2 Generieren von Code anhand eines Mapping-Projekts

So generieren Sie Code anhand eines Mapping-Projekts (.mfp):

1. Öffnen Sie das Mapping-Projekt in MapForce, falls es noch nicht offen ist.
2. Klicken Sie mit der rechten Maustaste in das Fenster "Projekt" und wählen Sie anschließend den Befehl **Eigenschaften**.



3. Überprüfen Sie die Projekteinstellungen und ändern Sie diese bei Bedarf (stellen Sie v.a. sicher, dass die Zielsprache und das Ausgabeverzeichnis korrekt eingestellt wurden) und klicken Sie anschließend auf **OK**.
4. Wählen Sie im Menü **Projekt den Befehl Code für das gesamte Projekt generieren**.

Das Ergebnis der Codegenerierung (Fehler- oder Erfolgsmeldung) wird im Fenster "Meldungen" angezeigt.

Standardmäßig ist der Name der generierten Applikation mit dem Projektnamen identisch. Wenn der Projektname Leerzeichen enthält, werden diese im generierten Code in Unterstriche konvertiert. Standardmäßig wird Code im selben Verzeichnis wie das MapForce-Projekt generiert, nämlich im untergeordneten Ausgabeverzeichnis.

Um das Ausgabeverzeichnis und den Namen des Projekts zu ändern, klicken Sie im Projektfenster auf das Projekt und wählen Sie im Menü den Befehl **Projekt | Eigenschaften. Wenn Ihr MapForce-Projekt Ordner enthält, können Sie die Codegenerierungseinstellungen für die einzelnen Ordner (durch Rechtsklick auf den Ordner und Auswahl von Eigenschaften) ändern. Andernfalls erben alle Projektordner die Einstellungen vom MapForce-Projekt.**

© 2016-2022 Altova GmbH

16.4.3 Erstellen des Projekts mit Ant

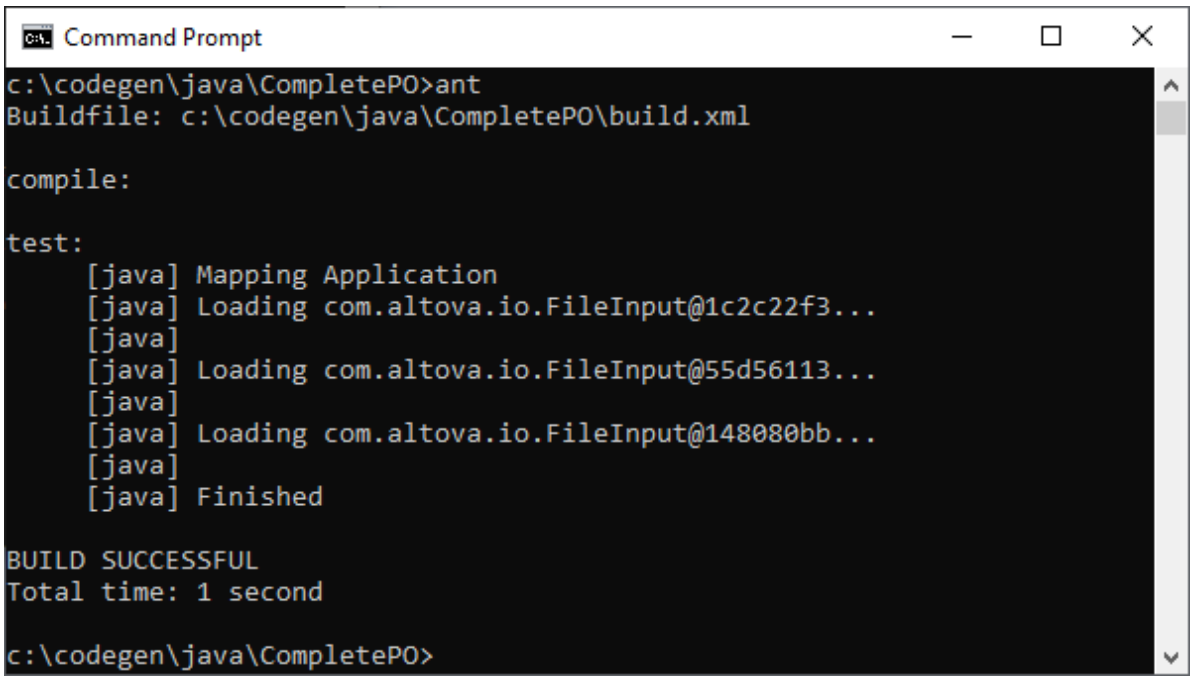
Apache Ant ist ein Open Source-Tool, das das Erstellen und Kompilieren von Java-Projekten automatisiert (siehe http://ant.apache.org/manual/). Ant funktioniert mit Build-Dateien (in diesen Dateien sind die Quellen und Ziele für die Codekompilierung sowie alle spezifischen Build-Optionen definiert). Da jedes MapForce-generierte Projekt eine von Ant erkannte **build.xml** Datei enthält, können Sie mit Ant ganz einfach MapForce-generierte Projekte erstellen.

Ant kann auf Ihrem System entweder als eigenständige Applikation oder mit Eclipse (oder anderen Java IDEs) gebündelt zur Verfügung stehen. Eine Anleitung zum Installieren von Ant auf Ihrem System finden Sie unter http://ant.apache.org/manual/. Eine Anleitung zur Verwendung von Ant in Eclipse finden Sie in der Dokumentation zu Eclipse.

Um schnell zu überprüfen, ob die Standalone-Version von Ant (und nicht die mit Eclipse gebündelte) auf Ihrem System zur Verfügung steht, öffnen Sie ein Eingabeaufforderungsfenster und geben Sie `ant` in die Befehlszeile ein. Wenn Ant bereits verfügbar ist, sollten Sie in etwa folgende Meldung erhalten: `Buildfile: buildxml does not exist!` Diese Meldung besagt, dass Ant installiert ist und versucht, eine **build.xml** zu erstellen, dass diese build.xml-Datei aber im aktuellen Verzeichnis nicht vorhanden ist. Wenn Sie Ant von einem Verzeichnis aus ausführen, das eine **build.xml** enthält, führt stattdessen die **build.xml** Datei mit den jeweils darin definierten Optionen aus.

So erstellen Sie ein MapForce-generiertes Java-Projekt mit Ant:

1. Öffnen Sie eine Befehlszeilenaufforderung und navigieren Sie zu dem Verzeichnis, in dem das Java-Projekt generiert wurde (Beachten Sie, dass das Verzeichnis die Datei **build.xml** enthalten muss).
2. Geben Sie `ant` in die Befehlszeile ein. Dadurch wird der Java-Code kompiliert und entsprechend den in der Datei **build.xml** definierten Optionen ausgeführt.



So generieren Sie eine JAR-Datei mit Ant:

- Geben Sie in die Befehlszeile `ant jar` ein.

Um die Hilfe zur Ant-Befehlszeilensyntax und den Optionen dazu anzuzeigen, geben Sie `ant -help` in die Befehlszeile ein. Eine Demo finden Sie unter Beispiel: Generieren und Ausführen von Java-Code 1586.

16.4.4 Behandlung von JDBC-Referenzen

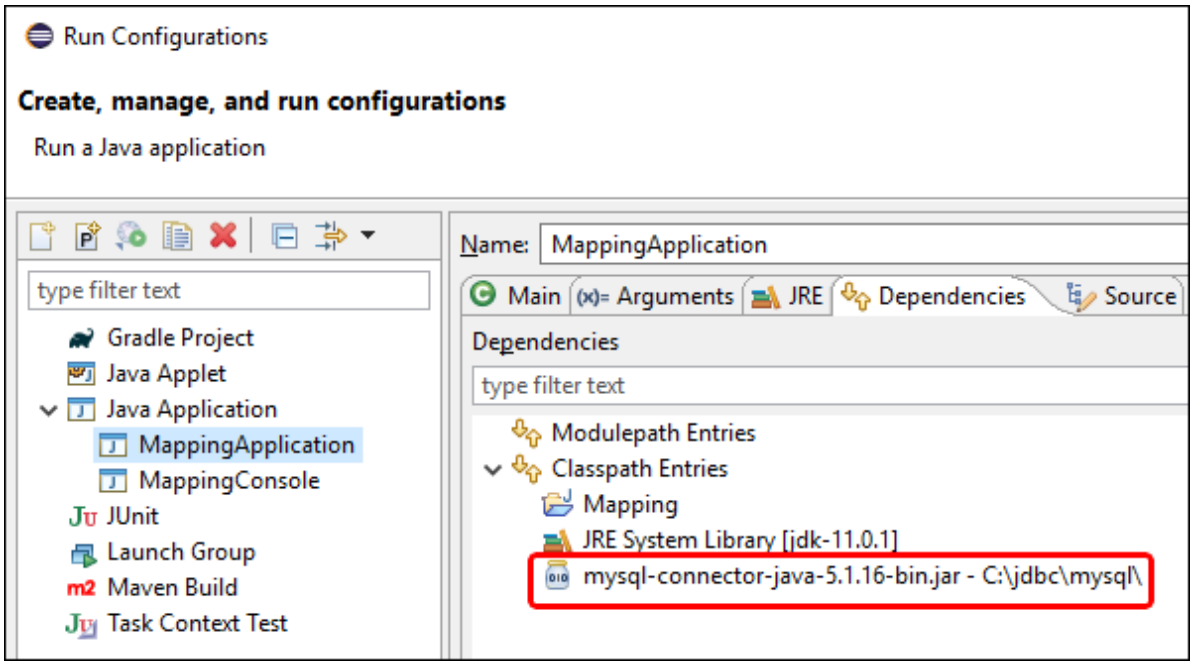
Wenn das Mapping über JDBC mit einer Datenbank verbunden wird, stellen Sie sicher, dass die vom Mapping verwendeten JDBC-Treiber auf Ihrem System installiert sind. Um die aktuellen JDBC-Einstellungen einer Datenbankkomponente in MapForce zu sehen, klicken Sie darauf und wählen Sie im Menü den Befehl **Komponente | Eigenschaften**. Nähere Informationen dazu finden Sie unter Einrichten einer JDBC-Verbindung 190.

> Wenn im Mapping eine Nicht-JDBC-Datenbankverbindung verwendet wired, wird die Verbindung bei der Generierung von Java-Code eventuell in eine JDBC-Verbindung umgewandelt, um in einer Java-Umgebung Kompatibilität zu gewährleisten. So werden etwa ADO-, ADO.NET- und ODBC-Verbindungen bei Generierung von Java-Code anhand des Mappings in JDBC-Verbindungen umgewandelt. Nähere Informationen dazu finden Sie unter Datenbankmappings in verschiedenen Ausführungsumgebungen 166.

Wenn Sie in Eclipse eine Testausführung der generierten Java-Applikation starten, muss eventuell der JDBC-Treiber als "Classpath"-Eintrag zur aktuellen run-Konfiguration hinzugefügt werden. Andernfalls könnte die Testausführung der Applikation in Eclipse mit einem Fehler wie dem folgenden fehlschlagen: `java.lang.ClassNotFoundException: com.mysql.jdbc.Driver`. Wenn Sie eine MySQL-Datenbank als Beispiel verwenden, können Sie die JDBC-Treiber in Eclipse folgendermaßen hinzufügen:

1. Generieren Sie den Java-Code über MapForce und importieren Sie das Projekt mit dem Menübefehl **File | Import | Existing Projects into Workspace** in **Eclipse.**
2. Klicken Sie im Menü **Run** auf **Run Configurations**. Auf der linken Seite des Dialogfelds sehen Sie unter "Java Application", die zwei Konfigurationen **MappingApplication** und **MappingConsole**.
3. Klicken Sie auf die gewünschte Konfiguration (z.B. **MappingApplication**).
4. Klicken Sie auf dem Register **Dependencies** auf **Classpath entries** und anschließend auf **Add External JARs**.
5. Navigieren Sie zur .jar-Datei des JDBC-Treibers, z.B:

© 2016-2022 Altova GmbH



6. Klicken Sie auf **Run,** um das Programm mit dem als Abhängigkeit hinzugefügten JDBC-Datenbanktreiber auszuführen.

Wenn Sie bei der Testausführung der Ant **build.xml**-Datei den obigen JDBC-Treiberfehler erhalten, fügen Sie die JDBC-Treiber zum Classpath der Aufgabe "test" hin. Im Folgenden sehen Sie ein Beispiel für eine Ant-Test-Aufgabe, die eine Referenz auf die .jar-Datei des JDBC-Treibers enthält.

```
<target name="test" depends="compile">
   <java classpath="C:\codegen\java\mysql_mapping"
classname="com.mapforce.MappingConsole" fork="true" failonerror="true">
      <classpath>
         <pathelement path="${classpath}"/>
         <pathelement location="C:\jdbc\mysql\mysql-connector-java-5.1.16-bin.jar"/>
      </classpath>
      <arg line="${cmdline}"/>
   </java>
</target>
```

Anmerkung: Im obigen Beispiel wird davon ausgegangen, dass die .jar-Datei unter dem folgenden Pfad vorhanden ist: **C:\jdbc\mysql\mysql-connector-java-5.1.16-bin.jar**. Stellen Sie sicher, dass Sie den .jar-Dateipfad entsprechend Ihrer Umgebung anpassen.

Inkludieren der JDBC-Treiber in den Manifest-Abschnitt der Applikation

Wenn Sie JAR-Dateien aus der generierten Java-Applikation erstellen, fügen Sie im "manifest"-Abschnitt der **build.xml** Datei eine Referenz zum Datenbanktreiber hinzu. Auf diese Weise stellen Sie sicher, dass die Referenz zum Datenbanktreiber nach Erstellung des Projekts in der manifest-Datei (**MANIFEST.MF**) vorhanden ist.

So fügen Sie die Datenbankreferenz zur manifest-Datei hinzu:

1. Suchen Sie in der **build.xml**-Datei das Element **manifest**.
2. Fügen Sie ein neues Attribut namens **attribute** hinzu, wobei das Attribute "name" "Class-Path" und der Wert des Attributs der Name der .jar-Datei ist. Für MySQL 5.1.16 sieht das neue Element z.B. folgendermaßen aus:

```
<attribute name="Class-Path" value="mysql-connector-java-5.1.16-bin.jar"/>
```

   Das **manifest**-Element der Datei **build.xml** sieht nun folgendermaßen aus:

```
<manifest file="C:\codegen\java\mysql_mapping/META-INF/MANIFEST.MF"
mode="replace">
   <attribute name="Created-By" value="MapForce 2023"/>
   <attribute name="Main-Class" value="com.mapforce.MappingConsole"/>
   <attribute name="Class-Path" value="mysql-connector-java-5.1.16-bin.jar"/>
</manifest>
```

3. Kopieren Sie die JAR-Datei des JDBC-Treibers in den Ordner, der die JAR-Datei der generierten Applikation enthält.

16.4.5 Beispiel: Generieren und Ausführen von Java-Code

In diesem Beispiel wird Schritt für Schritt erklärt, wie Sie mit MapForce eine Java-Applikation erstellen und diese außerhalb von MapForce über die Eclipse IDE (Integrated Development Environment) und mit Hilfe von Apache Ant kompilieren. Am Ende des Beispiels haben Sie eine fertige Java-Applikation kompiliert, mit der Sie eines der in MapForce verfügbaren Mapping-Beispiele ausführen können.

Wenn Sie bereits andere Java-Applikationen mit Eclipse und Ant kompilieren können, sind keine weiteren Voraussetzungen für dieses Beispiel erforderlich. Beachten Sie andernfalls die folgenden Punkte:

- Der Java Development Kit (JDK), Eclipse sowie Ant müssen auf Ihrem System installiert sein. Eclipse enthält normalerweise eine gebündelte Version von Ant. Sie können Ant jedoch auch separat installieren, siehe auch Erstellen des Projekts mit Ant 1583.
- Um Eclipse mit OpenJDK auszuführen, muss die PATH-Umgebungsvariable normalerweise den Pfad zum JDK-**bin**-Verzeichnis (z.B: **C:\Java\jdk-11.0.1\bin**) enthalten.
- Wenn Sie Apache wie in diesem Beispiel separat ausführen, sollte die PATH-Umgebungsvariable auch den Pfad zum Ant **bin**-Verzeichnis (z.B. **C:\apache-ant-1.10.5\bin**) enthalten, sodass Sie Ant jederzeit ausführen können, ohne den vollständigen Pfad zur ausführbaren Datei in der Befehlszeile eingeben zu müssen.

In diesem Beispiel wird die folgenden Konfiguration verwendet:

- MapForce 64-Bit
- OpenJDK 11.0.1 64-Bit
- Eclipse IDE für Java-Entwickler Version 2019-03 (4.11.0), 64-Bit

© 2016-2022 Altova GmbH

- Apache Ant 1.10.6 (als eigenständige Applikation installiert) und Apache Ant 1.10.5 (im Paket mit Eclipse)

Schritt 1: Generieren von Java-Code

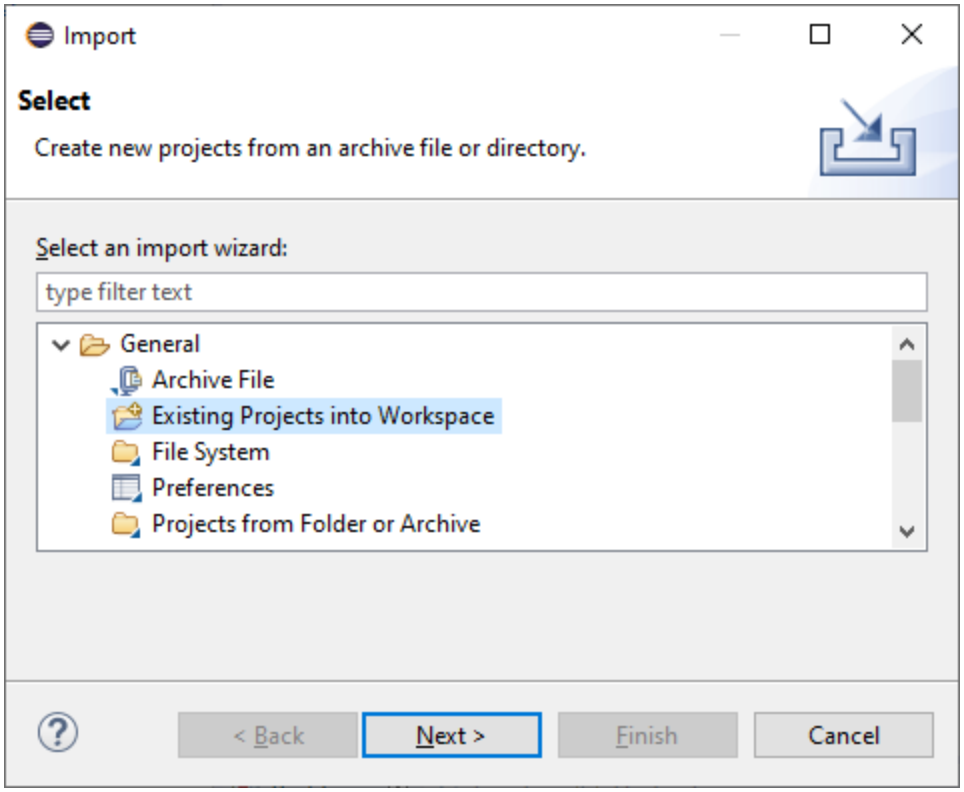
So generieren Sie den Java-Code in MapForce:

1. Klicken Sie im Menü **Datei** auf **Öffnen** und navigieren Sie zum Mapping **CompletePO.mfd** im Verzeichnis <Dokumente>\Altova\MapForce2023\MapForceExamples\.
2. Klicken Sie im Menü **Ausgabe** auf **Java**. Daraufhin wird die Transformationssprache in Java geändert.
3. Klicken Sie im Menü **Datei** auf **Code generieren in | Java**. Wenn Sie dazu aufgefordert werden, navigieren Sie zu dem Verzeichnis, in dem das Java-Projekt gespeichert werden soll. Aus Gründen der Einfachheit können Sie das Projekt in diesem Beispiel unter **C:\codegen\java\CompletePO** speichern. Sie müssen diesen Pfad in späteren Schritten referenzieren.

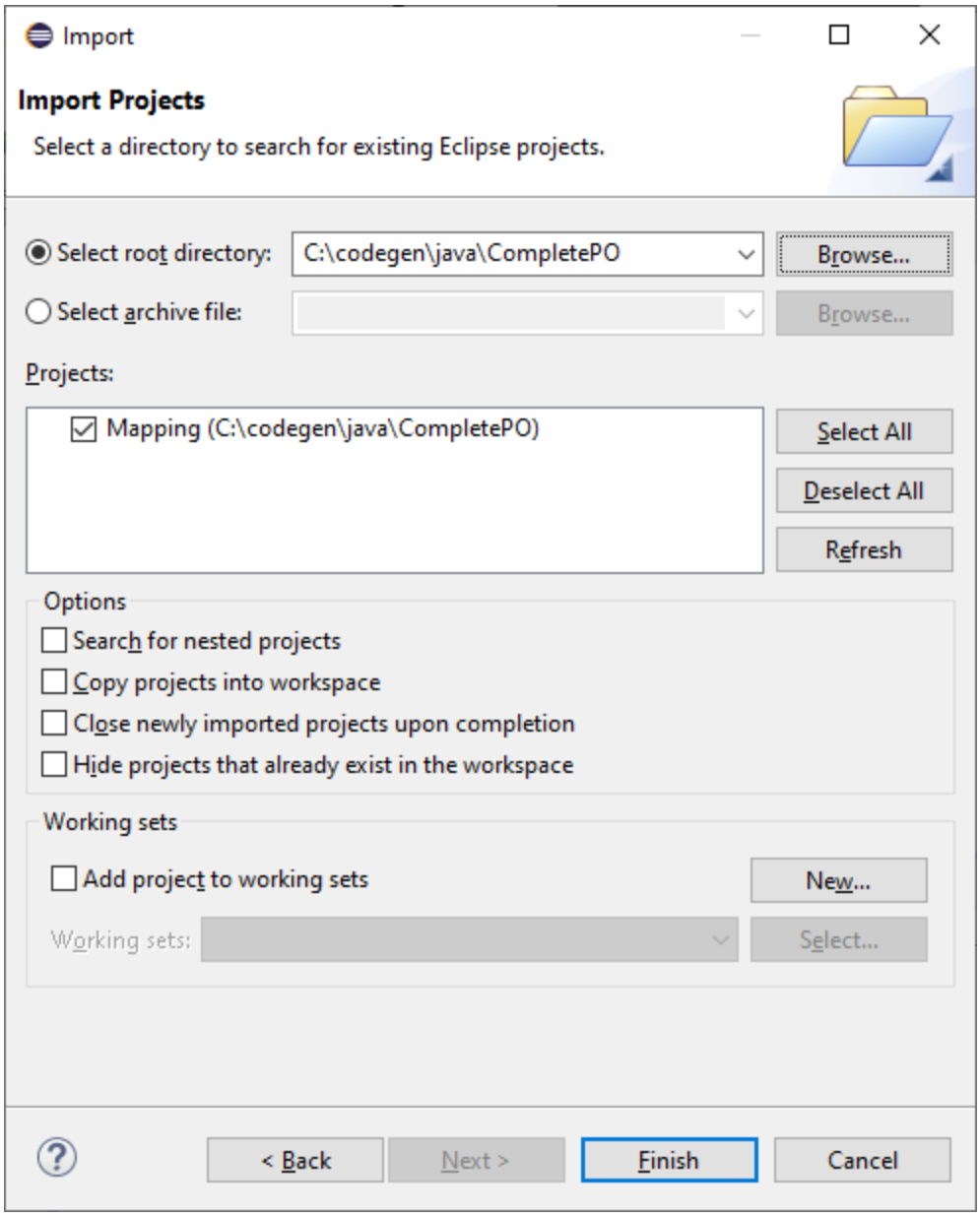
Schritt 2: Importieren des Projekts in Eclipse

So importieren Sie das Projekt in Eclipse:

1. Starten Sie Eclipse, falls Sie das noch nicht getan haben und wechseln Sie über den Menübefehl **Window | Open Perspective** in die Java-Standardperspektive.
2. Klicken Sie im Menü **File** auf **Import** und anschließend auf **General | Existing Projects into Workspace**.

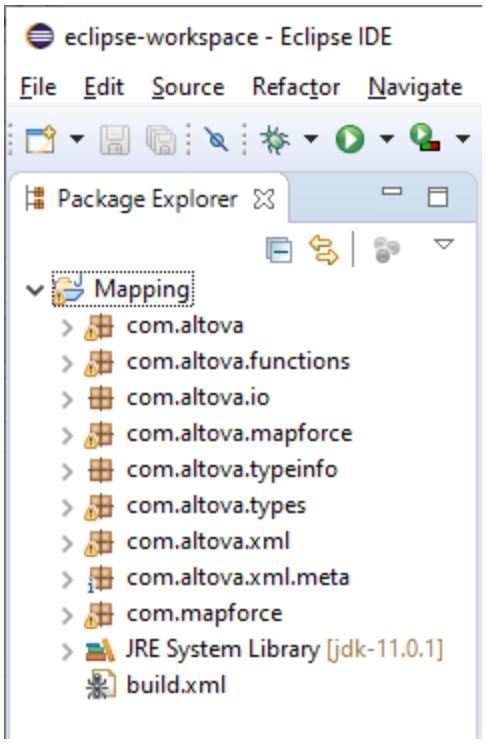


3. Klicken Sie auf **Next**.



4. Navigieren Sie zu dem Ordner, in dem Sie zuvor den generierten Code gespeichert haben und klicken Sie anschließend auf **Finish**. Das mit MapForce erstellte Java-Projekt steht nun in der Package Explorer-Ansicht zur Verfügung. Wenn Sie die Package Explorer-Ansicht nicht sehen, wählen Sie den Menübefehl **Window | Show View | Package Explorer,** um die Ansicht anzuzeigen.

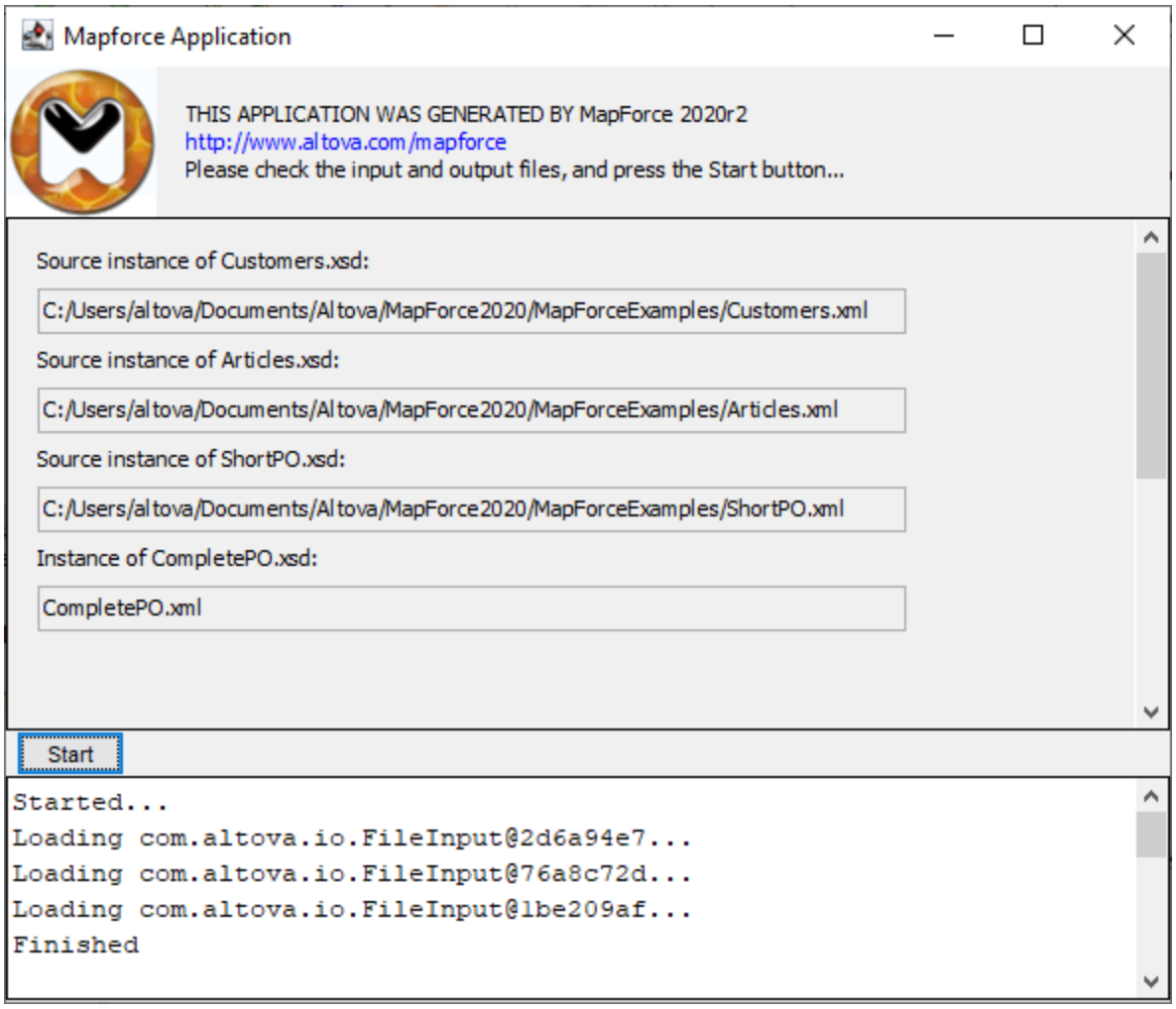
© 2016-2022 Altova GmbH



Schritt 3: Ausführen des Projekts als Dialog-Applikation

So führen Sie das Java-Projekt als GUI-Applikation aus:

1. Klicken Sie in der Package Explorer-Ansicht von Eclipse auf die Datei **MappingApplication.java** im Paket **com.mapforce**.
2. Wählen Sie im Menü **Run** den Befehl **Run As | Java application**.
3. Klicken Sie im MapForce-Applikationsfenster auf **Start,** um das Mapping auszuführen.



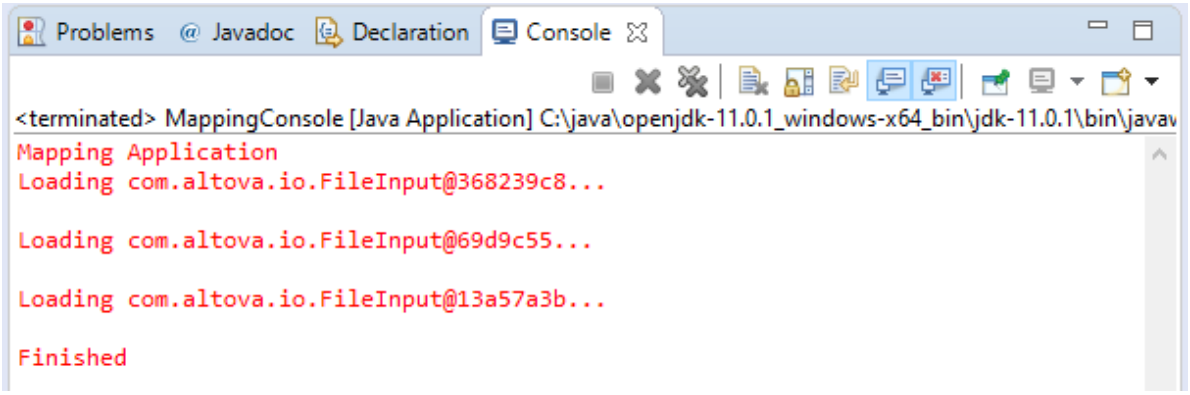
Wenn es in Eclipse zu Laufzeitfehlern oder zu Fehlern im Zusammenhang mit der Systemkonfiguration kommt, erscheint eine entsprechende Eingabeaufforderung, siehe auch Behandlung von JDBC-Referenzen[1584]. Andernfalls führt die Java-Applikation die Mapping-Transformation aus und generiert die Datei **CompletePO.xml** unter dem Ausgabepfad (in diesem Beispiel: **C:\codegen\Java\CompletePO**).

Schritt 4: Ausführen des Projekts als Konsolenapplikation

So führen Sie das Java-Projekt als Konsolenapplikation aus:

1. Klicken Sie in der Package Explorer-Ansicht von Eclipse auf die Datei **MappingConsole.java** im Paket **com.mapforce**.
2. Klicken Sie im Menü **Run** auf **Run As | Java application**.

© 2016-2022 Altova GmbH



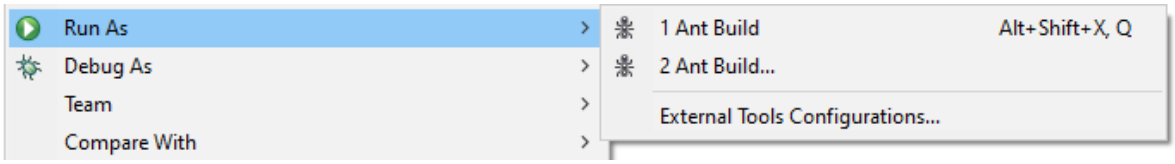
Wenn es in Eclipse zu Laufzeitfehlern oder zu Fehlern im Zusammenhang mit der Systemkonfiguration kommt, erscheint eine entsprechende Eingabeaufforderung, siehe auch Behandlung von JDBC-Referenzen[1584]. Andernfalls führt die Java-Applikation die Mapping-Transformation aus und generiert die Datei **CompletePO.xml** unter dem Ausgabepfad (in diesem Beispiel: **C:\codegen\Java\CompletePO**).

Schritt 5: Erstellen der JAR-Datei

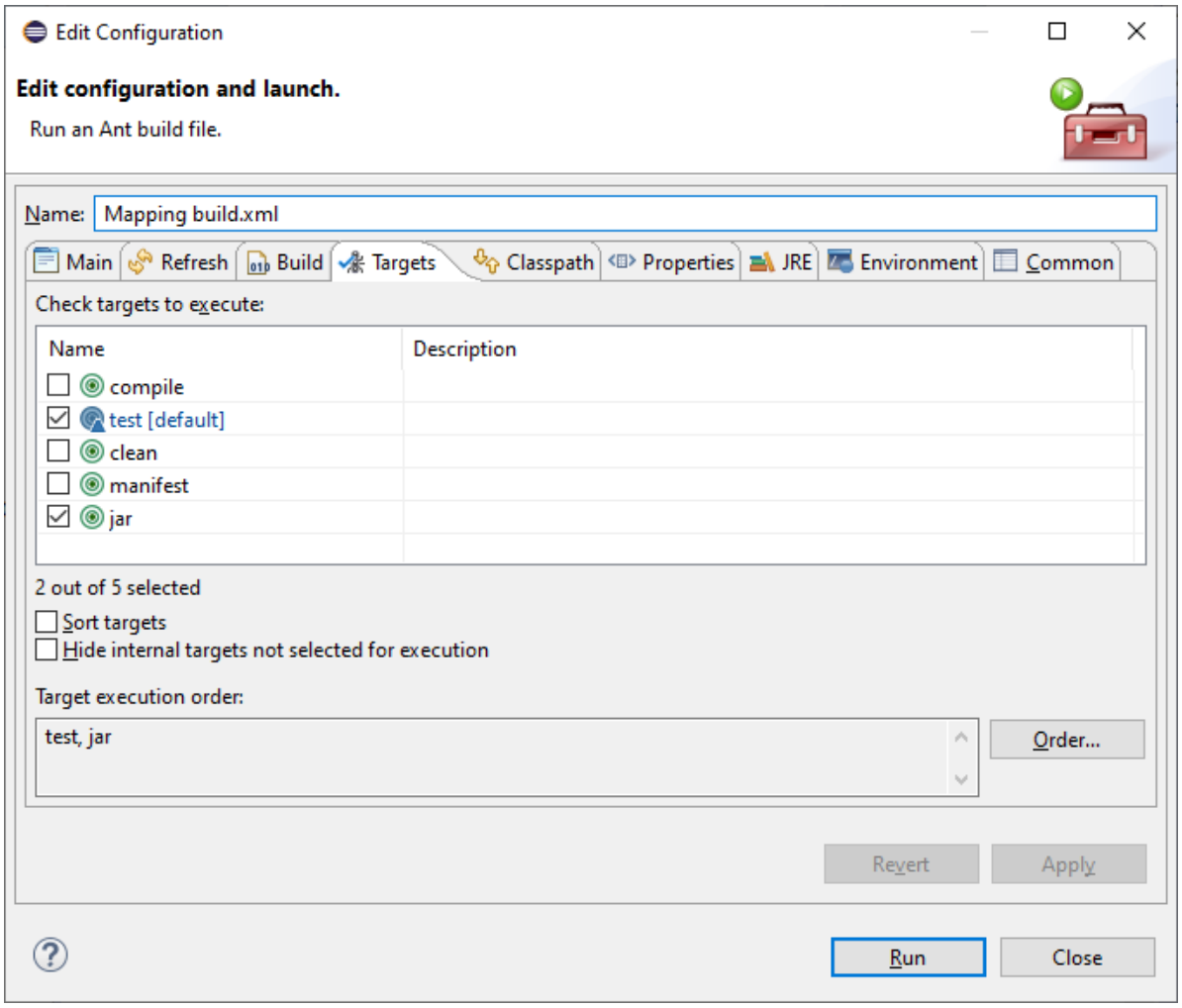
Wenn Apache Ant mit Eclipse gebündelt ist, können Sie die JAR-Datei direkt von der Eclipse IDE aus erstellen. Um die JAR-Datei außerhalb von Eclipse über die Befehlszeile zu erstellen, muss Apache Ant separat installiert sein.

So erstellen Sie die .jar-Datei in Eclipse mit Ant:

1. Klicken Sie in der Package Explorer-Ansicht von Eclipse mit der rechten Maustaste auf die Datei **build.xml** und verweisen Sie auf das Menü **Run as**. Im "Run As"-Menü werden zwei Optionen für die Ausführung der Ant Build-Datei angezeigt. Wenn Sie die erste Option auswählen, startet Eclipse den Ant Build mit den Standardeinstellungen. Bei Auswahl der zweiten Option können Sie die Einstellungen des Ant Build vor dem Start ändern. Wählen Sie die zweite Option aus.



2. Aktivieren Sie die Ziele, die im Ant Build inkludiert werden sollen, durch Anklicken. In diesem Beispiel sind die Ziele **test** und **jar** ausgewählt.



3. Klicken Sie auf **Run**. Eclipse führt die Ant Build-Datei aus und zeigt das Ergebnis in der Konsolenansicht an.

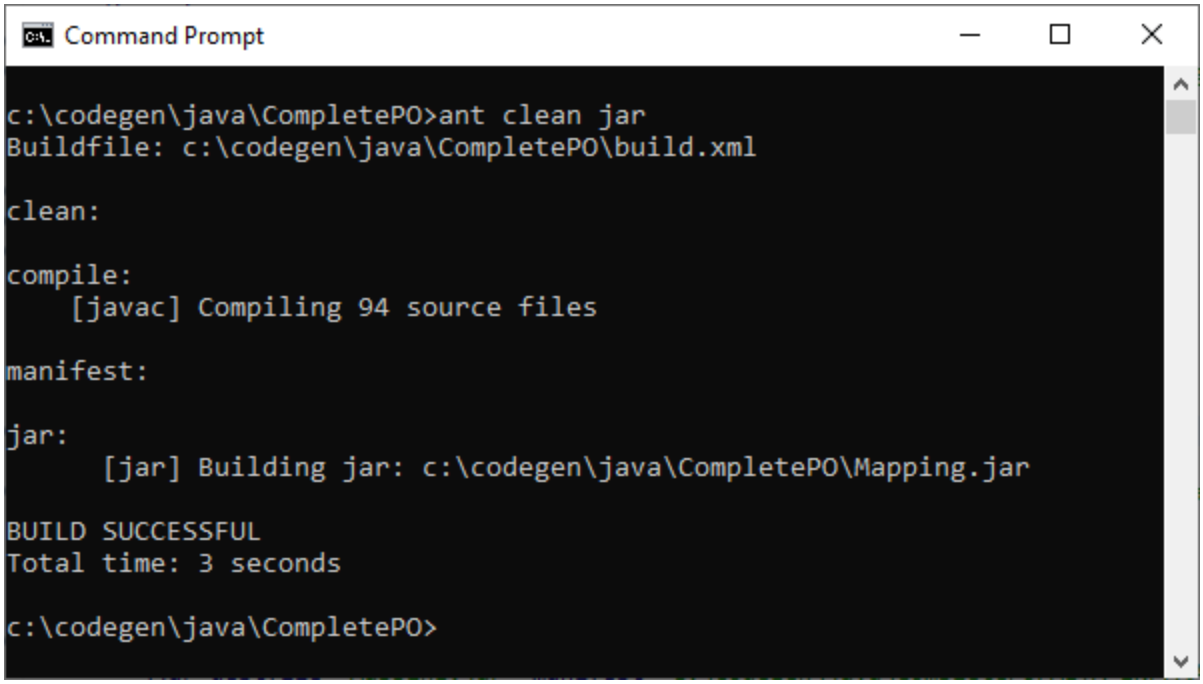
So erstellen Sie die JAR-Datei über die Befehlszeile:

1. Stellen Sie sicher, dass das Verzeichnis, in dem Apache Ant installiert ist (z.B. **C:\apache-ant-1.10.5\bin**), zur PATH-Umgebungsvariablen hinzugefügt wurde.
2. Öffnen Sie ein Befehlszeilenfenster und ändern Sie das aktuelle Verzeichnis in das, in dem Sie Java-Code generiert haben (in diesem Beispiel **C:\codegen\java\CompletePO**). Dies ist auch das Verzeichnis, in dem die **build.xml**-Datei generiert wurde.
3. Geben Sie Folgendes in die Befehlszeile ein:

```
ant clean jar
```

Daraufhin führt Ant die Ziele "clean" und "jar" aus der **build.xml**-Datei aus (einschließlich aller davon abhängigen Ziele) und zeigt das Ergebnis in der Befehlszeile an:

© 2016-2022 Altova GmbH

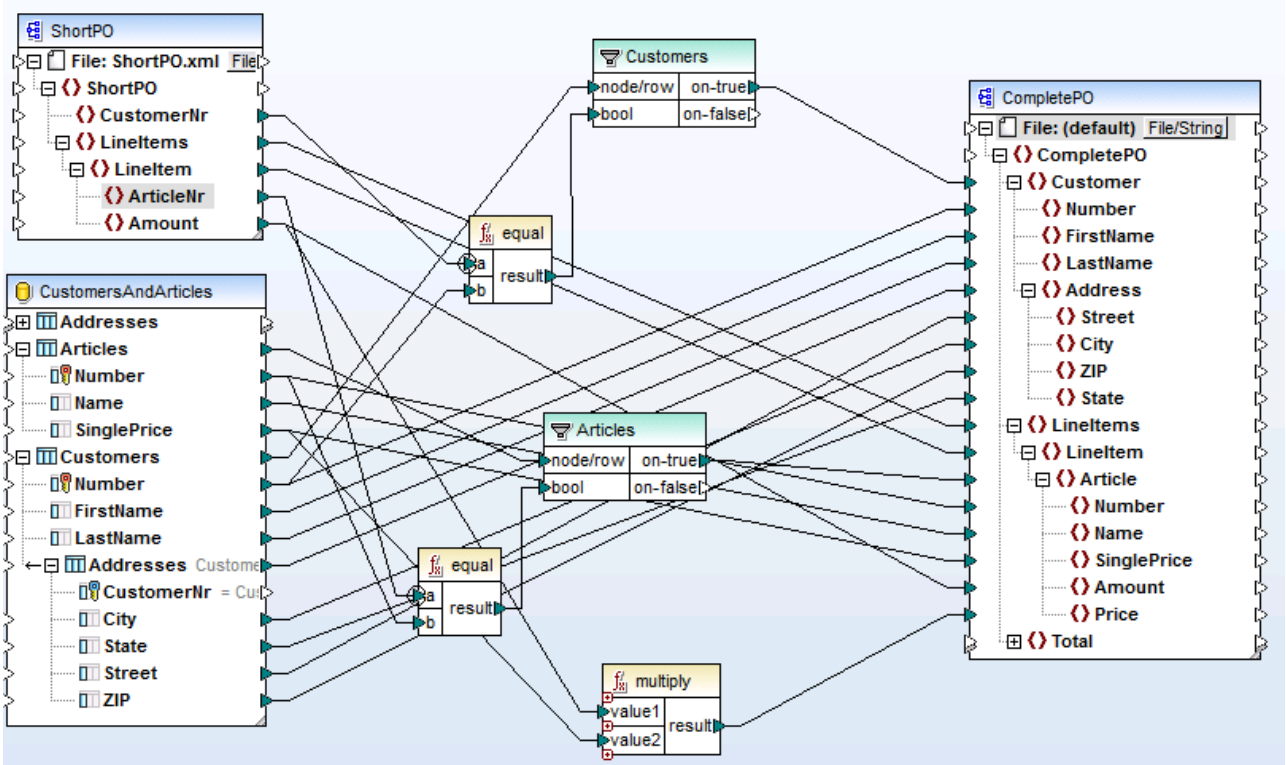

Command Prompt
c:\codegen\java\CompletePO>ant clean jar
Buildfile: c:\codegen\java\CompletePO\build.xml
clean:
compile:
[javac] Compiling 94 source files
manifest:
jar:
[jar] Building jar: c:\codegen\java\CompletePO\Mapping.jar
BUILD SUCCESSFUL
Total time: 3 seconds
c:\codegen\java\CompletePO>

16.5 Integrieren von mit MapForce generiertem Code

In MapForce generierter Code kann in Ihre jeweilige Applikation integriert oder dafür adaptiert werden. Bei der Codegenerierung wird eine komplette und voll funktionsfähige Applikation generiert. Einige typische Szenarien, in denen Sie den generierten Code eventuell ändern müssen, sind die folgenden:

- Definieren benutzdefinierter Quell- und Zieldateien für die Mapping-Applikation
- Hinzufügen von benutzerdefiniertem Fehlerbehandlungscode
- In mit C# oder Java generiertem Code können Sie auch den Datentyp des Mapping-Input im Programm ändern (z.B. von String in Stream).

In diesem Abschnitt wird anhand des Beispielmappings **DB_CompletePO.mfd** aus dem Ordner **<Dokumente>\Altova\MapForce2023\MapForceExamples** beschrieben, wie Sie dies bewerkstelligen.



DB_CompletePO.mfd mapping sample in MapForce

Wie unten beschrieben, besteht das Beispielmapping aus zwei Quellkomponenten und einer Zielkomponente:

- **ShortPO.xml** ist die XML-Quelldatei
- **CustomersAndArticles.mdb** ist eine Quelldatenbank
- **CompletePO.xml** ist die XML-Zieldatei.

Diese Quell- und Zielkomponenten werden im generierten Code in zwei Input- und einen Output-Parameter übersetzt, welche der run Methode, mit der das Mapping ausgeführt wird, zur Verfügung gestellt werden (Beschreibung siehe folgende Kapitel). Beachten Sie bei der Codegenerierung die folgenden grundlegenden Punkte:

© 2016-2022 Altova GmbH

- Die Anzahl der Quell- und Zielkomponenten im Mapping-Design entspricht im generierten Code der Anzahl der Mapping-Parameter für die **run** Methode.
- Wenn Sie die Anzahl der Quellen und Ziele des Mappings ändern, müssen Sie den Code entsprechend neu generieren.
- Wenn Sie Änderungen am generierten Code vornehmen und den Code anschließend im selben Ordner neu generieren, werden alle Änderungen überschrieben.

Wenn ein Mapping Datenbankkomponenten enthält, so enthält die generierte **run**-Methode das Datenbankverbindungsobjekt unter dem entsprechenden Pfad. Wenn im Mapping z.B. drei Quellkomponenten (Textinhalt, XML-Inhalt und eine Datenbank) verwendet werden, die auf eine einzige Output-Datei gemappt werden, so generiert MapForce die folgende **run**-Methode:

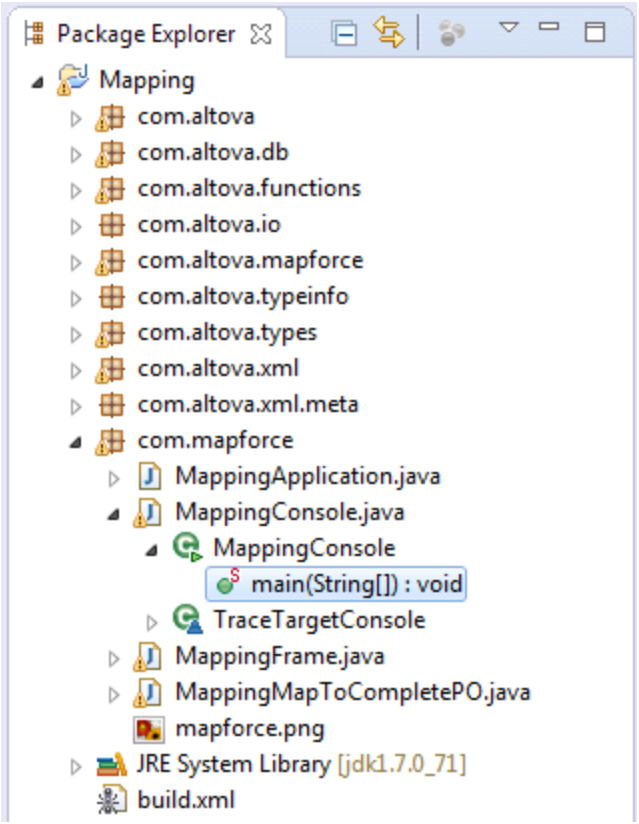
Java

```
void run(Input in1, Input in2, java.sql.Connection dbConn, Output out1);
```

Die Reihenfolge der Argumente ist von Bedeutung. Wie Sie in den nächsten Beispielen sehen werden, können Sie beim Integrieren Ihres Codes die `dbConn`-Parameter ändern oder die von MapForce generierten Standardparameter verwenden.

16.5.1 Java-Beispiel

In diesem Beispiel wird Eclipse als Java IDE verwendet. Generieren Sie als erstes anhand des Beispielmappings **DB_CompletePO** aus dem Verzeichnis <Dokumente>\Altova\MapForce2023\MapForceExamples\ Java-Code und importieren Sie das Projekt anschließend in Eclipse.



Mit MapForce generierte Java-Beispielapplikation (Eclipse IDE)

Um die generierte Java-Konsolenapplikation zu bearbeiten, gehen Sie zur `main(String[] args)` Methode Ihrer generierten Applikation (siehe Abbildung oben). Wenn Sie den Base Package-Standardnamen vor der Codegenerierung nicht geändert haben, befindet sich diese Methode in der Klasse `MappingConsole` des `com.mapforce`-Pakets. Andernfalls befindet sie sich in der Klasse `MappingConsole` Ihres benutzerdefinierten Pakets.

Um die generierte Java-Dialogapplikation zu bearbeiten, navigieren Sie zu der Stelle im Code, an der die `run`-Methode über Ihre generierte Applikation aufgerufen wird. Wenn Sie den Base Package-Standardnamen vor der Codegenerierung nicht geändert haben, wird die `run`-Methode über die Klasse `MappingFrame.java` des `com.mapforce`-Pakets aufgerufen.

Im folgenden Beispielcode sehen Sie einen Ausschnitt aus der `main`-Methode in der generierten Java-Konsolenapplikation. Mapping-Quelle und -Ziel sind gelb markiert und als Parameter für die run-Methode definiert. Da in diesem Mapping eine Datenbankverbindung verwendet wird, hat der entsprechende Parameter eine spezielle Struktur: Die Verbindung besteht aus dem Connection String (in diesem Fall `jdbc:odbc:;DRIVER=Microsoft Access Driver (*.mdb);DBQ=CustomersAndArticles.mdb`) sowie aus zwei leeren Argumenten (in Klartext) für den **Benutzernamen** und das **Passwort** für diese Datenbanken (falls diese benötigt werden).

Beachten Sie, dass die Dateipfade im unten gezeigten Code von absoluten in relative Pfade geändert wurden.

```
com.altova.io.Input ShortPO2Source =
```

 © 2016-2022 Altova GmbH

```
com.altova.io.StreamInput.createInput("ShortPO.xml");
com.altova.io.Output CompletePO2Target = new com.altova.io.FileOutput("CompletePO.xml");

MappingMapToCompletePOObject.run(
                  com.altova.db.Dbs.newConnection(
                     "jdbc:odbc:;DRIVER=Microsoft Access Driver
(*.mdb);DBQ=CustomersAndArticles.mdb",
                     "",
                     ""),
                  ShortPO2Source,
                  CompletePO2Target);
```

So definieren Sie benutzerdefinierte Quell- oder Zieldateien für das Mapping:

- Gehen Sie zu den an die `run`-Methode übergebenen Parametern und bearbeiten Sie diese entsprechend. Im Beispiel oben sind com.altova.db.Dbs.newConnection und ShortPO2Source die Input-Dateien für das Mapping und CompletePO2Target ist die Mapping-Output-Datei.

So fügen Sie zusätzlichen Fehlerbehandlungscode hinzu:

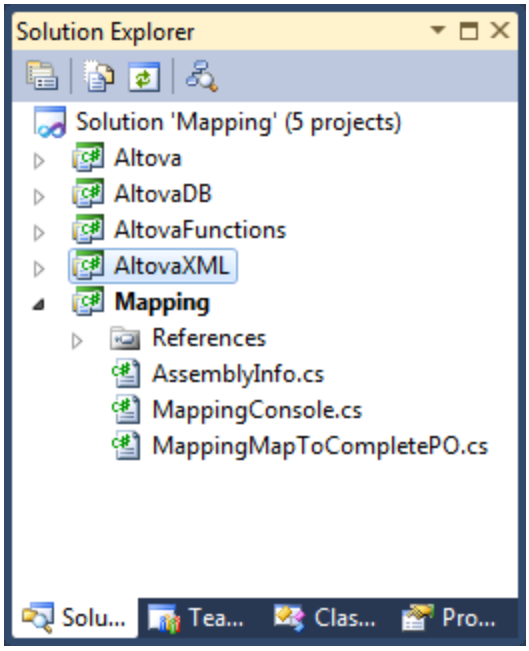
- Bearbeiten Sie den Code unterhalb des **catch** (Exception e) Codes (im Fall einer Java-Konsolenapplikation).
- Bearbeiten Sie den Code unterhalb des **catch** (Exception ex) Codes (im Fall einer Java-Dialogapplikation).

Eine Anleitung, wie Sie den Datentyp von Parametern ändern, die als Mapping-Input/Output bereitgestellt wurden, finden Sie unter Programmatisches Ändern des Input und Output 1600.

16.5.2 C#-Beispiel

In diesem Beispiel wird die Visual Studio 2010 IDE verwendet. Generieren Sie als erstes anhand des Beispielmappings **DB_CompletePO** aus dem Verzeichnis <Dokumente>\Altova\MapForce2023\MapForceExamples\ C#-Code und öffnen Sie die Lösung anschließend in Visual Studio.

Standardmäßig lautet der Name der Lösungsdatei **Mapping.sln.** Die Datei befindet sich relativ zu dem Verzeichnis, in dem Sie den generierten Code gespeichert haben, im Unterverzeichnis **Mapping**. Wenn Sie den Applikationsnamen über die Mapping-Einstellungen geändert haben, so wird der Name der .sln-Datei entsprechend geändert. Wenn Sie z.B. den Applikationsnamen in **MyApplication** geändert haben, so erhält die Lösungsdatei den Namen **MyApplication.sln** und befindet sich im Unterverzeichnis **MyApplication**.



Mit MapForce generierte C#-Beispiellösung

Öffnen Sie die Datei MappingConsole.cs und navigieren Sie zur `main(String[]args)`-Methode. Im folgenden Codebeispiel sehen Sie einen Ausschnitt aus der `main`-Methode. Mapping-Quelle und -Ziel sind als Parameter für die **Run**-Methode definiert. Da in diesem Mapping Daten aus einer Datenbank gelesen werden, gibt es auch einen Input-Parameter, bei dem es sich um einen Datenbankverbindungsstring handelt. Gegebenenfalls können Sie diesen String bearbeiten.

Beachten Sie, dass die Dateipfade im unten gezeigten Code von absoluten in relative Pfade geändert wurden.

```
Altova.IO.Input ShortPO2Source = Altova.IO.StreamInput.createInput("ShortPO.xml");
Altova.IO.Output CompletePO2Target = new Altova.IO.FileOutput("CompletePO.xml");

MappingMapToCompletePOObject.Run(
               "Provider=Microsoft.Jet.OLEDB.4.0; Data Source=CustomersAndArticles.mdb;
",
               ShortPO2Source,
               CompletePO2Target);
```

So definieren Sie benutzerdefinierte Quell- oder Zieldateien für das Mapping:

- Gehen Sie zu den an die Run-Methode übergebenen Parametern und bearbeiten Sie diese entsprechend. Im Beispiel oben ist der Mapping-Input ein Connection String zu **CustomersAndArticles.mdb** und `ShortPO2Source`. Die MappingAusgabe ist `CompletePO2Target`.

So fügen Sie zusätzlichen Fehlerbehandlungscode hinzu:

- Bearbeiten Sie den Code unterhalb des `catch (Exception e)` Codes.

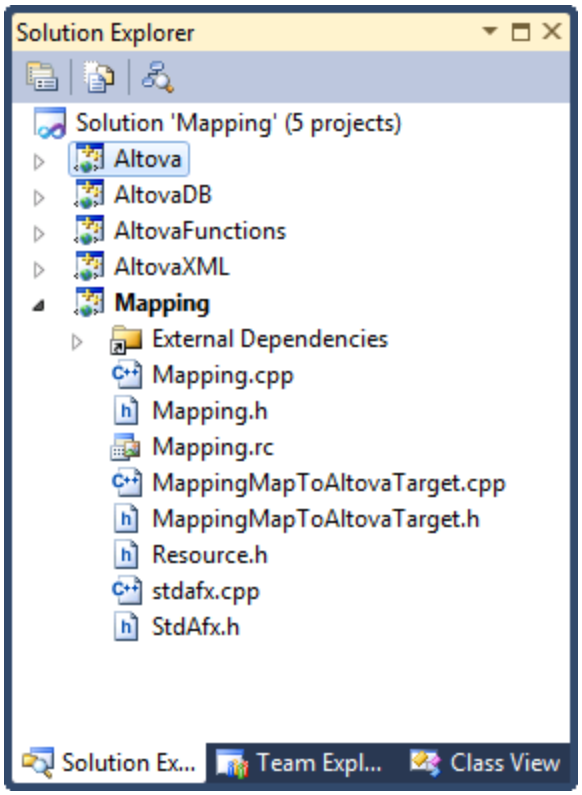
© 2016-2022 Altova GmbH

Eine Anleitung, wie Sie den Datentyp von Parametern ändern, die als Mapping-Input/Output bereitgestellt wurden, finden Sie unter Programmatisches Ändern von Input und Output[1600].

16.5.3 C++-Beispiel

In diesem Beispiel wird die Visual Studio 2010 IDE verwendet. Generieren Sie als erstes anhand des Beispielmappings **DB_CompletePO** aus dem Verzeichnis <Dokumente>\Altova\MapForce2023\MapForceExamples\ C++-Code und öffnen Sie die Lösung anschließend in Visual Studio.

Standardmäßig lautet der Name der Lösungsdatei **Mapping.sln.** Die Datei befindet sich relativ zu dem Verzeichnis, in dem Sie den generierten Code gespeichert haben, im Unterverzeichnis **Mapping**. Wenn Sie den Applikationsnamen über die Mapping-Einstellungen geändert haben, so wird der Name der .sln-Datei entsprechend geändert. Wenn Sie z.B. den Applikationsnamen in **MyApplication** geändert haben, so erhält die Lösungsdatei den Namen **MyApplication.sln** und befindet sich im Unterverzeichnis **MyApplication**.



Mit MapForce generierte C++-Beispiellösung

Öffnen Sie die Datei Mapping.cpp und navigieren Sie zur `_tmain`-Methode. Im folgenden Codebeispiel sehen Sie einen Ausschnitt aus dieser Methode. Mapping-Quelle und -Ziel sind als Parameter für die `Run`-Methode definiert. Da in diesem Mapping Daten aus einer Datenbank gelesen werden, gibt es auch einen Input-Parameter, bei dem es sich um einen Datenbankverbindungsstring handelt. Gegebenenfalls können Sie diesen String bearbeiten.

Beachten Sie, dass die Dateipfade im unten gezeigten Code von absoluten in relative Pfade geändert wurden.

```
MappingMapToCompletePO MappingMapToCompletePOObject;
         MappingMapToCompletePOObject.Run(
            _T("Provider=Microsoft.Jet.OLEDB.4.0; Data Source=CustomersAndArticles.mdb;
"),
            _T("ShortPO.xml"),
            _T("CompletePO.xml"));
```

So definieren Sie benutzerdefinierte Quell- und Zieldateien für ein Mapping:

- Gehen Sie zu den an die Run-Methode übergebenen Parametern und bearbeiten Sie diese entsprechend. Im Beispiel oben ist der Mapping-Input ein Connection String zu **CustomersAndArticles.mdb** und _T("ShortPO.xml"). _T("CompletePO.xml") ist die Mapping-Output-Datei.

So fügen Sie zusätzlichen Fehlerbehandlungscode hinzu:

- Bearbeiten Sie den Code unterhalb des **catch** (CAltovaException& e) Codes.

16.5.4 Programmatisches Ändern von Input und Output

Nachdem Sie mit MapForce Java- oder C#-Code generiert haben, können Sie optional den Datentyp des Mapping-Input oder -Output durch Bearbeitung des generierten Code ändern. Dabei können Sie andere Objekttypen als die standardmäßig generierten als Mapping-Parameter verwenden. So können Sie etwa einen String oder ein Stream-Objekt als Input bereitstellen, statt den Input aus einer Datei auf dem Datenträger auszulesen. Beachten Sie, dass diese Funktionalität nur für in C# oder Java generierten Code verfügbar ist.

In der ersten Spalte der nachstehenden Tabelle sind die als Input oder Output unterstützten Objekttypen aufgelistet. In den nachfolgenden Spalten sind die Datenformate, in denen der jeweilige Typ unterstützt wird, angegeben. Eine genauere Definition der einzelnen Typen finden Sie im Abschnitt "Typdefinitionen" weiter unten.

	XML	JSON*	Microsoft Excel*	EDI (einschließlich X12, HL7)*	FlexText*	CSV/Text
Dateien	Ja	Ja	Ja	Ja	Ja	Ja
Streams	Ja	Ja	Ja	Ja	Ja	Ja
Strings	Ja	Ja	–	Ja	Ja	Ja
Reader/Writer	Ja	Ja	–	Ja	Ja	Ja
DOM-Dokumente	Ja	–	–	–	–	–

© 2016-2022 Altova GmbH

** Nur in der MapForce Enterprise Edition unterstützte Formate*

So ändern Sie den Datentyps des Mapping-Input oder -Output:

1. Generieren Sie anhand eines Mappings C#- oder Java-Code.
2. So finden Sie im generierten Code den Aufruf der **run**-Methode (in Java) bzw. der **Run**-Methode (in C#):

   a. Wenn Sie C# verwenden, öffnen Sie die Datei **MappingConsole.cs**.
   b. Wenn Sie Java verwenden, öffnen Sie die Datei **MappingConsole.java** (das Konsolenprogramm) oder die Datei **MappingFrame.java** (das GUI-Programm).

Anmerkung: Wenn Sie den Applikationsnamen in den Mapping-Einstellungen[116] geändert haben, hat die Datei eventuell einen anderen Namen. Wenn Sie den Namen z.B. in "MyApp" geändert haben, so lautet der Name der generierten Datei **MyAppConsole.js** bzw. **MyAppConsole.java** bzw. **MyAppFrame.java**.

3. Erstellen Sie eine Instanz des gewünschten Typs (siehe Abschnitt "Typdefinitionen").
4. Stellen Sie die deklarierten Objekte, wie in den Beispielen unten gezeigt, als Parameter für die **run**-Methode (in Java) bzw. die **Run**-Methode (in C#) bereit.

Die **run**-Methode ist die wichtigste Methode generierter Mapping-Klassen. Sie hat einen Parameter für jede *statische* Quelle oder Input-Komponente im Mapping und einen Endparameter für die Output-Komponente. Wenn Ihr Mapping Komponenten enthält, die mehrere Dateien dynamisch verarbeiten, scheinen die entsprechenden Parameter im generierten Code nicht auf, da die Dateinamen in diesem Fall dynamisch[1371] innerhalb des Mappings verarbeitet werden.

Typdefinitionen

In C# handelt es sich bei den Typen, die Sie als Parameter für die **Run**-Methode bereitstellen können, um im `Altova.IO`-Namespace definierte Klassen. Die Basisklassen sind `Altova.IO.Input` bzw. `Altova.IO.Output`.

C#-Typen

Dateien	`Altova.IO.FileInput(string filename)` `Altova.IO.FileOutput(string filename)`
Streams	`Altova.IO.StreamInput(System.IO.Stream stream)` `Altova.IO.StreamOutput(System.IO.Stream stream)`
Strings	`Altova.IO.StringInput(string content)` `Altova.IO.StringOutput(System.Text.StringBuilder sbuilder)`
Reader/Writer	`Altova.IO.ReaderInput(System.IO.TextReader reader)` `Altova.IO.WriterOutput(System.IO.TextWriter writer)`
DOM-Dokumente	`Altova.IO.DocumentInput(System.Xml.XmlDocument document)` `Altova.IO.DocumentOutput(System.Xml.XmlDocument document)`

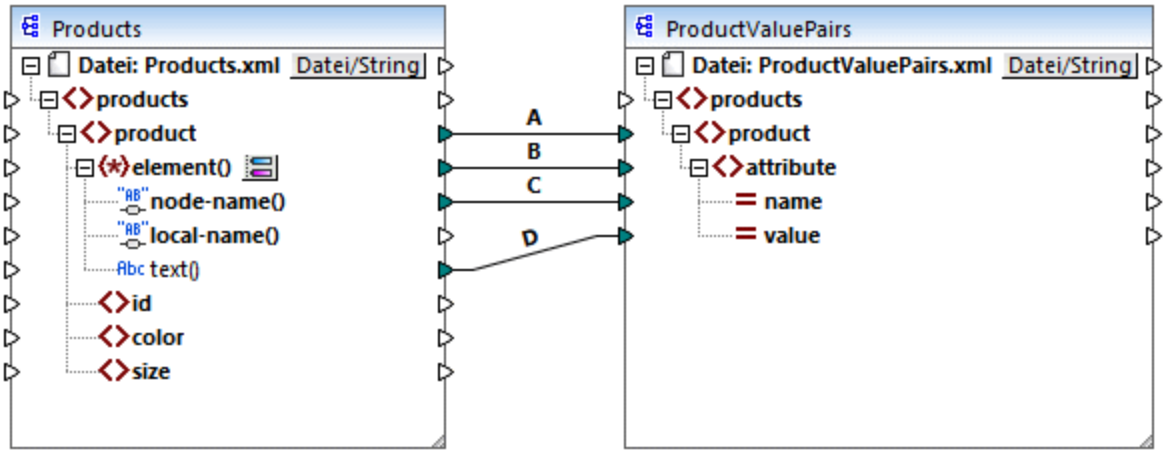
In Java handelt es sich bei den Typen, die Sie als Parameter für die **run**-Methode bereitstellen können, um im `com.altova.io`-Paket definierte Klassen. Die Basisklassen sind `com.altova.io.Input` bzw. `com.altova.io.Output`.

Java-Typen

Dateien	`com.altova.io.FileInput(String filename)` `com.altova.io.FileOutput(String filename)`
Streams	`com.altova.io.StreamInput(java.io.InputStream stream)` `com.altova.io.StreamOutput(String filename)`
Strings	`com.altova.io.StringInput(String content)` `com.altova.io.StringOutput()`
Reader/Writer	`com.altova.io.ReaderInput(java.io.Reader reader)` `com.altova.io.WriterOutput(java.io.Writer writer)`
DOM-Dokumente	`com.altova.io.DocumentInput(org.w3c.dom.Document document)` `com.altova.io.DocumentOutput(org.w3c.dom.Document document)`

Beispiel

Um zu zeigen, wie Sie den Input und Output programmatisch ändern, verwenden wir das Mapping **ConvertProducts.mfd** als Beispiel. Nachdem Sie MapForce installiert haben und mindestens einmal gestartet haben, finden Sie das Mapping im folgenden Verzeichnis: **C:\Benutzer\<Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForceExamples\Tutorials**.



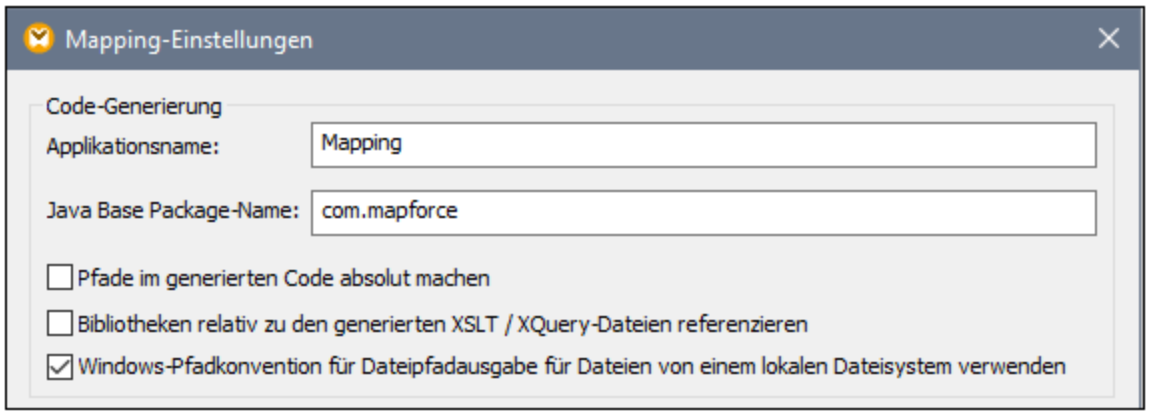
ConvertProducts.mfd

Wie oben gezeigt, konvertiert das Mapping Daten aus einem XML-Quelldokument in ein anderes XML-Dokument. Wir möchten Folgendes erreichen:

1. Generierung von Java- und C#-Programmcode anhand dieses Mappings.
2. Ändern des Datentyps der Quellkomponente in einen String-Typ.
3. Ändern des Datentyps der Zielkomponente in einen String-Writer-Typ.

Um den Programmcode zu generieren, öffnen Sie das Mapping **ConvertProducts.mfd** und wählen Sie den Befehl **Datei | Code generieren in | C#** (bzw. **Java**). Für dieses Beispiel nehmen wir an, dass als Mapping-Einstellungen von **ConvertProducs.mfd** die Standardeinstellungen verwendet werden.

© 2016-2022 Altova GmbH



In diesem Beispiel wird der Code in den folgenden Zielverzeichnissen generiert (Sie können den Pfad gegebenenfalls ändern):

- **C:\codegen\cs\ConvertProducts** für C#
- **C:\codegen\java\ConvertProducts** für Java

Nachdem Sie den Programmcode generiert haben, öffnen Sie die Datei **MappingConsole.cs** (in C#) bzw. **MappingConsole.java** (in Java) und gehen Sie zu den folgenden Zeilen:

C#

```
Altova.IO.Input Products2Source = Altova.IO.StreamInput.createInput("Products.xml");
Altova.IO.Output ProductValuePairs2Target = new
Altova.IO.FileOutput("ProductValuePairs.xml");
```

Java

```
com.altova.io.Input Products2Source =
com.altova.io.StreamInput.createInput("Products.xml");
com.altova.io.Output ProductValuePairs2Target = new
com.altova.io.FileOutput("ProductValuePairs.xml");
```

Kommentieren Sie die obigen Zeilen aus und ändern Sie den Code folgendermaßen:

C#

```
//Altova.IO.Input Products2Source = Altova.IO.StreamInput.createInput("Products.xml");
//Altova.IO.Output ProductValuePairs2Target = new
Altova.IO.FileOutput("ProductValuePairs.xml");

Altova.IO.Input Products2Source = new Altova.IO.StringInput("<?xml version=\"1.0\"
encoding=\"UTF-8\"?>\r\n" +
                                    "<products
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation=\"products.xsd\">\r\n" +
                                    "   <product>\r\n" +
                                    "      <id>100</id>\r\n" +
                                    "      <color>blue</color>\r\n" +
```

```
                                        "      <size>XXL</size>\r\n" +
                                        "   </product>\r\n" +
                                        "</products>\r\n");

System.IO.StringWriter writer = new System.IO.StringWriter(new
System.Text.StringBuilder());
Altova.IO.Output ProductValuePairs2Target = new Altova.IO.WriterOutput(writer);

try
{
    MappingMapToProductValuePairsObject.Run(Products2Source, ProductValuePairs2Target);

    // Print out the writer object
    Console.Write(writer.ToString());
}
finally
{
    Products2Source.Close();
    ProductValuePairs2Target.Close();
}
```

Java

```
//com.altova.io.Input Products2Source =
com.altova.io.StreamInput.createInput("Products.xml");
//com.altova.io.Output ProductValuePairs2Target = new
com.altova.io.FileOutput("ProductValuePairs.xml");

com.altova.io.Input Products2Source = new com.altova.io.StringInput("<?xml
version=\"1.0\" encoding=\"UTF-8\"?>\r\n" +
               "<products xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation=\"products.xsd\">\r\n" +
               "   <product>\r\n" +
               "      <id>100</id>\r\n" +
               "      <color>blue</color>\r\n" +
               "      <size>XXL</size>\r\n" +
               "   </product>\r\n" +
               "</products>\r\n");

java.io.StringWriter writer = new java.io.StringWriter();
com.altova.io.Output ProductValuePairs2Target = new com.altova.io.WriterOutput(writer);

try {
   MappingMapToProductValuePairsObject.run(Products2Source, ProductValuePairs2Target);

   // Print out the writer object
   System.out.print(writer.toString());

   } finally {
      (Products2Source).close();
      ProductValuePairs2Target.close();
}
```

Im obigen C#- und Java-Codefragment geschieht Folgendes:

© 2016-2022 Altova GmbH

- Die beiden Originalzeilen, in denen der Input und Output für die **run**-Methode angegeben wurde, wurden auskommentiert. Die Mapping-Applikation liest die Daten folglich nicht mehr aus **Products.xml**. Diese Datei musste auch nicht in das Arbeitsverzeichnis des Programms kopiert werden.
- Der Typ `Products2Source` wurde als `StringInput` deklariert, der den Inhalt der zu verarbeitenden XML-Datei bereitstellt.
- Der Typ `ProductValuePairs2Target` wurde als `WriterOutput`-Typ deklariert, der als Argument einen String Writer erhält.
- Nach Ausführung des Mappings wird der Inhalt des String Writer in die Konsole ausgedruckt.

Verwendungsrichtlinien für Streams und Reader/Writer-Objekte

Beachten Sie bei Verwendung von Binärdaten-Streams oder Reader/Writer-Objekten als Input oder Output des Mappings folgende Dinge:

- Binärdaten-Stream-Objekte und Reader/Writer müssen zuerst geöffnet werden und einsatzbereit sein, bevor die **run** Methode aufgerufen wird.
- Standardmäßig schließt die **run**-Methode den Stream nach dem Beenden. Um dies zu verhindern, fügen Sie vor Aufruf der **run**-Methode die folgende Zeile ein (bzw. heben Sie die Auskommentierung auf):

C#

```
MappingMapToSomething.CloseObjectsAfterRun = false;   // C#
```

Java

```
MappingMapToSomething.setCloseObjectsAfterRun(false);
```

Anmerkung: Stellen Sie sicher, dass Sie den Namen `MappingMapToSomething` in den Namen des entsprechenden Mapping-Objekts für Ihren generierten Code ändern.

Verwendungsrichtlinien für Strings

In Java erhält der Konstruktor von `StringOutput` kein Argument. Der vom Mapping erzeugte String-Inhalt kann mit der `getString()`-Methode aufgerufen werden, z.B:

Java

```
com.altova.io.Input Products2Source =
com.altova.io.StreamInput.createInput("Products.xml");
com.altova.io.StringOutput ProductValuePairs2Target = new com.altova.io.StringOutput();

try {
   // Run the mapping
   MappingMapToProductValuePairsObject.run(Products2Source, ProductValuePairs2Target);
   // Get the string object
   String str = ProductValuePairs2Target.getString().toString();
}
```

In C# erhält der Konstruktor von `StringOutput` einen Parameter vom Typ `StringBuilder`, welchen Sie vorher deklarieren müssen. Wenn das `StringBuilder`-Objekt bereits Daten enthält, wird die Mapping-Ausgabe daran angehängt.

C#

```
Altova.IO.Input Products2Source = Altova.IO.StreamInput.createInput("Products.xml");
System.Text.StringBuilder sb = new System.Text.StringBuilder();
Altova.IO.Output ProductValuePairs2Target = new Altova.IO.StringOutput(sb);

try
{
   // Run the mapping
   MappingMapToProductValuePairsObject.Run(Products2Source, ProductValuePairs2Target);
   // Get the string output
   String str = sb.ToString();
}
```

Um diese Codefragmente auszuführen, können Sie dasselbe generierte Projekt wie im vorherigen Beispiel verwenden. Stellen Sie jedoch sicher, dass Sie die Datei **Products.xml** aus **C:\Benutzer\<Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForceExamples\Tutorials** in das Arbeitsverzeichnis Ihres Programms kopieren, da im Mapping-Code Daten aus dieser Datei ausgelesen werden.

Verwendungsrichtlinien für DOM-Dokumente

Wenn Sie DOM-Dokumente als Mapping-Input oder -Output verwenden, beachten Sie Folgendes:

- Die als Parameter an den `DocumentOutput`-Konstruktor übergebene Dokumentinstanz muss leer sein.
- Nach Aufruf von **`run`** enthält das vom Konstruktor von `DocumentOutput` generierte DOM-Dokument bereits die Mapping-Ausgabe und Sie können das Dokument nach Bedarf bearbeiten.

© 2016-2022 Altova GmbH

16.6 Generieren von Code anhand von XML-Schemas oder DTDs

Bei der Codegenerierung wird in MapForce eine komplette Applikation generiert, die alle Mapping-Schritte automatisch ausführt. Sie können optional Bibliotheken für alle im Mapping verwendeten XML-Schemas generieren. Mit Hilfe dieser Bibliotheken kann Ihr Code jederzeit XML-Instanzen erstellen oder lesen, die vom Mapping-Code verwendet oder erstellt wurden.

Um für alle im Mapping verwendeten XML-Schemas Bibliotheken zu generieren, aktivieren Sie in den Codegenerierungsoptionen[1701] das Kontrollkästchen **Wrapper-Klassen generieren** . Wenn Sie das nächste Mal Code generieren, erstellt MapForce nicht nur die Mapping-Applikation für Sie, sondern zusätzlich dazu Wrapper-Klassen für alle im Mapping verwendeten Schemas.

C++ oder C#	Java	Verwendungszweck
Altova	`com.altova`	Basisbibliothek, enthält allgemeine Runtime-Unterstützung, ist für alle Schemas identisch.
AltovaXML	`com.altova.xml`	Basisbibliothek, enthält Runtime-Unterstützung für XML, ist für alle Schemas identisch.
[YourSchema]	`com.YourSchema`	Eine Bibliothek, die anhand des Input-Schemas generierte Deklarationen enthält, die denselben Namen wie die Schema-Datei oder DTD hat. Diese Bibliothek ist ein DOM (W3C Document Object Model) Wrapper, der es Ihnen gestattet, XML-Dokumente einfach und sicher zu lesen, zu bearbeiten und zu erstellen. Alle Daten befinden Sie im DOM und es gibt Methoden, um Daten aus dem DOM zu extrahieren und Daten im DOM zu aktualisieren und zu erstellen. Der generierte C++-Code unterstützt entweder Microsoft MSXML oder Apache Xerces 3. Die Syntax zur Verwendung des generierten Codes ist mit Ausnahme geringer Unterschiede (so unterstützt Xerces z.B. mehr überladene Funktionen) bei beiden DOM-Implementierungen fast die gleiche. Dem generierten C#-Code liegt die .NET Standard System.XML-Bibliothek als DOM-Implementierung zugrunde. Dem generierten Java-Code liegt JAXP (Java API for XML Processing) als DOM-Implementierung zugrunde.

Beim Erstellen eines Prototyps einer Applikation anhand eines häufig geänderten XML-Schemas müssen Sie eventuell immer wieder Code im selben Verzeichnis generieren, damit die Änderungen am Schema sofort im Code berücksichtigt werden. Beachten Sie, dass die generierte Testapplikation und die Altova-Bibliotheken jedes Mal, wenn Sie Code im selben Zielverzeichnis generieren, überschrieben werden. Fügen

Sie daher keinen Code zur generierten Testapplikation hinzu, sondern integrieren Sie stattdessen die Altova-Bibliotheken in Ihr Projekt (siehe Integrieren von Schema Wrapper-Bibliotheken[1617]).

Zusätzlich zu den oben aufgelisteten Basisbibliotheken werden einige weitere unterstützende Bibliotheken generiert, die zusammen mit den Altova-Basisbibliotheken verwendet werden und nicht für benutzerdefinierte Integrationen gedacht sind, da sie Änderungen unterworfen sind.

Namensgenerierung und Namespaces

MapForce generiert für alle deklarierten Elemente oder complexTypes, die einen complexTyp in Ihrem XML-Schema umdefinieren, Klassen, wobei die durch Erweiterungen von complexTyps in Ihrem XML-Schema definierte Klassenableitung beibehalten wird. Bei komplexen Schemas, in die Komponenten aus mehreren Namespaces importiert werden, behält MapForce diese Informationen durch Generierung der entsprechenden C#- oder C++-Namespaces oder Java-Pakete bei.

Im Allgemeinen versucht der Code Generator die Namen für generierte Namespaces, Klassen und Member aus dem Original-XML-Schema beizubehalten. Zeichen, die in Identifiern in der Zielsprache ungültig sind, werden durch ein "_" ersetzt. Namen, die mit anderen Namen oder reservierten Wörtern in Konflikt treten würden, werden durch Anhängen einer Nummer eindeutig identifizierbar gemacht. Die Generierung von Namen kann durch Änderung der Standardeinstellungen in der SPL[1703]-Vorlage beeinflusst werden.

Die Namespaces aus dem XML-Schema werden in Java in Pakete oder in C#- oder C++-Code in Namespaces konvertiert, wobei das Namespace-Präfix aus dem Schema als Code-Namespace verwendet wird. Die gesamte Bibliothek wird in ein Paket oder einen anhand des Schema-Dateinamens abgeleiteten Namespace eingeschlossen, sodass Sie in einem Programm mehrere generierte Bibliotheken verwenden können, ohne dass es zu Namenskonflikten kommt.

Datentypen

XML-Schema hat ein etwas komplexeres Datentypmodell als Java, C# oder C++. Der Code Generator konvertiert die vordefinierten XML-Schematypen in sprachspezifische primitive Typen oder in mit der Altova-Bibliothek bereitgestellte Klassen. ComplexTypes und abgeleitete Typen, die im Schema definiert sind, werden in der generierten Hierarchie in Klassen konvertiert. Enumeration Facets von simpleTypes werden in Symbolkonstanten konvertiert.

Das Mapping von simpleTypes kann in der SPL-Vorlage konfiguriert werden, siehe SPL-Referenz[1703].

Wenn in Ihren XML-Instanzdateien Schematypen im Zusammenhang mit Uhrzeit und Dauer verwendet werden, so werden diese im Code in native Altova-Klassen konvertiert. Informationen zu den Altova-Bibliotheksklassen finden Sie unter:

- Referenz zu generierten Klassen (C++)[1652]
- Referenz zu generierten Klassen (C#)[1668]
- Referenz zu generierten Klassen (Java)[1684]

Informationen zur Typkonvertierung und anderen Details zu den einzelnen Sprachen finden Sie unter:

- Informationen zu Schema Wrapper-Bibliotheken (C++)[1610]
- Informationen zu Schema Wrapper-Bibliotheken (C#)[1612]
- Informationen zu Schema Wrapper-Bibliotheken (Java)[1615]

© 2016-2022 Altova GmbH

Speicherverwaltung

Eine DOM-Struktur besteht aus Nodes, die immer einem bestimmten DOM-Dokument zugeordnet sind - selbst, wenn die Nodes derzeit nicht im Inhalt des Dokuments vorkommen. Bei allen generierten Klassen handelt es sich um Referenzen auf die DOM-Nodes, für die sie stehen, nicht um Werte. Dies bedeutet, dass bei Zuweisung einer Instanz einer generierten Klasse nicht der Wert kopiert wird, sondern nur eine zusätzliche Referenz auf dieselben Daten.

XML Schema-Unterstützung

Die folgenden XML-Schema-Konstrukte werden in Code übersetzt:

a) XML Namespaces

b) Simple Types

- Built-in XML Schema Typen
- Durch Extension abgeleitete Simple Types
- Durch Restriction abgeleitete Simple Types
- Facets
- Enumerations
- Patterns

c) Complex Types:

- Built-in anyType Node
- Benutzerdefinierte Complex Types
- Durch Extension abgeleitete: Auf abgeleitete Klassen gemappte
- Durch Restriction abgeleitete
- Complex Content
- Simple Content
- Mixed Content

Die folgenden komplexen XML-Schema-Funktionen werden in generierten Wrapper-Klassen nicht (oder nicht vollständig) unterstützt:

- Wildcards: `xs:any` und `xs:anyAttribute`
- Inhaltsmodelle (sequence, choice, all): Kompositor der obersten Ebene ist in SPL[1703] vorhanden, wird aber von den generierten Klassen nicht umgesetzt.
- Standardwerte und festgelegte Werte für Attribute: Diese sind in SPL[1703] verfügbar, werden aber von den generierten Klassen nicht gesetzt oder umgesetzt.
- Attribute `xsi:type`, abstract types: Verwenden Sie die `SetXsiType()`-Methode der generierten Klassen, wenn Sie das xsi:type-Attribut schreiben müssen.
- Union Types: Es werden nicht alle Kombinationen unterstützt.
- Substitution Groups werden zum Teil unterstützt (werden wie "choice" aufgelöst)
- Attribut `nillable="true"` und `xsi:nil`
- Uniqueness Constraints
- Identity Constraints (`key` und `keyref`)

16.6.1 Informationen zu Schema Wrapper-Bibliotheken (C++)

Zeichentypen

Der generierte C++-Code kann mit oder ohne Unicode-Unterstützung generiert werden. Je nachdem, welche Einstellung hier gewählt wurde, werden die Typen string_type und tstring beide als std::string und std::wstring und zwar bestehend aus schmalen oder breiten Zeichen, definiert. Um Unicode-Zeichen in Ihrer XML-Datei verwenden zu können, die nicht durch den aktuellen 8-Bit-Zeichensatz dargestellt werden können, muss die Unicode-Unterstützung aktiviert werden. Seien Sie bei den _T() Makros vorsichtig. Dieses Makro stellt sicher, dass String-Konstanten korrekt gespeichert werden, ob Sie nun für Unicode oder Nicht-Unicode-Programme kompilieren.

Datentypen

Für das Mappen von XML-Schematypen auf C++-Datentypen wird das folgende Standard-Mapping verwendet:

XML Schema	C++	Anmerkungen
`xs:string`	`string_type`	string_type wird definiert als std::string oder std:wstring
`xs:boolean`	`bool`	
`xs:decimal`	`double`	C++ hat keinen Dezimaltyp, daher wird "double" verwendet.
`xs:float, xs:double`	`double`	
`xs:integer`	`__int64`	xs:integer hat keine Bereichsbeschränkung, wird aus Gründen der Effizienz auf __int64 gemappt.
`xs:nonNegativeInteger`	`unsigned __int64`	siehe oben
`xs:int`	`int`	
`xs:unsignedInt`	`unsigned int`	
`xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth`	`altova::DateTime` 1652	
`xs:duration`	`altova::Duration` 1655	
`xs:hexBinary and xs:base64Binary`	`std::vector<unsigned char>`	Die Kodierung und Dekodierung von Binärdateien erfolgt automatisch.
`xs:anySimpleType`	`string_type`	

Alle XML-Schematypen, die nicht in der Liste enthalten sind, sind abgeleitete Typen und werden auf denselben C++-Typ wie der entsprechende Basistyp gemappt.

© 2016-2022 Altova GmbH

Generierte Klassen

Für jeden Typ im Schema wird eine Klasse generiert, die ein Member für jedes Attribut und Element des Typs enthält. Die Members erhalten den gleichen Namen wie die Attribute oder Elemente im Originalschema (bei möglichen Konflikten wird eine Zahl angehängt). Für Simple und Mixed Types werden Zuweisungs- und Konvertierungsoperatoren generiert. Für Simple Types mit Enumeration Facets können die Methoden `getEnumerationValue()` und `setEnumerationValue(int)` zusammen mit generierten Konstanten für die einzelnen Enumerationswerte verwendet werden. Des Weiteren können Sie mit Hilfe der Methode `StaticInfo()` Schemainformationen als einen der folgenden Typen abrufen:

`altova::meta::SimpleType` 1661
`altova::meta::ComplexType` 1659

Anhand von Complex Typs generierte Klassen enthalten die Methode `SetXsiType()`, mit Hilfe derer Sie das `xsi:type`-Attribut des Typs definieren können. Diese Methode ist nützlich, wenn Sie XML-Instanzelemente anhand eines Derived Type erstellen möchten.

Zusätzlich zu den Klassen für die im XML-Schema deklarierten Typen, wird eine (unten durch "CDoc" identifizierte) Dokument-Klasse generiert. Sie enthält alle möglichen Root-Elemente als Members sowie verschiedene andere Methoden. Nähere Informationen zur Klasse finden Sie unter `[YourSchema]::[CDoc]` 1662 .

Anmerkung: Der tatsächliche Klassenname ist vom Namen des .xsd-Schemas abhängig.

Für jedes Member-Attribut oder Elemente eines Schematyps wird eine neue Klasse generiert. Nähere Informationen zu solchen Klassen finden Sie unter:

`[YourSchema]::MemberAttribute` 1665
`[YourSchema]::MemberElement` 1666

Anmerkung: Der tatsächliche Klassenname ist vom Namen des Schema-Attributs oder -Elements abhängig.

Siehe auch Beispiel: Verwendung der Schema Wrapper-Bibliotheken 1620.

Behandlung von Fehlern

Fehler werden in Form von Ausnahmeereignissen ausgegeben. Im Namespace `altova` sind die folgenden Ausnahmeereignisklassen definiert:

Klasse	Basisklasse	Beschreibung
`Error`	`std::logic_error`	Interner Fehler in der Programmlogik (unabhängig von den Eingabedaten)
`Exception`	`std::runtime_error`	Basisklasse für Runtime-Fehler
`InvalidArgumentsException`	`Exception`	Es wurde eine Methode mit ungültigen Argumentwerten aufgerufen.
`ConversionException`	`Exception`	Ausnahmeereignis aufgrund eines Typkonvertierungsfehlers

StringParseException	ConversionException	Ein Wert im lexikalischen Bereich kann nicht in einen Wertebereich umgewandelt werden.
ValueNotRepresentableException	ConversionException	Ein Wert im Wertebereich kann nicht in einen lexikalischen Bereich umgewandelt werden.
OutOfRangeException	ConversionException	Ein Quellwert kann in der Ziel-Domain nicht dargestellt werden.
InvalidOperationException	Exception	Es wurde versucht, eine Operation auszuführen, die in diesem Kontext nicht gültig ist.
DataSourceUnavailableException	Exception	Beim Laden einer XML-Instanz ist ein Problem aufgetreten.
DataTargetUnavailableException	Exception	Beim Speichern einer XML-Instanz ist ein Problem aufgetreten.

Alle Ausnahmeereignisklassen enthalten einen Meldungstext und einen Verweis auf ein mögliches internes Ausnahmeereignis.

Methode	Verwendungszweck
string_type message()	Gibt eine Textbeschreibung des Ausnahmeereignisses zurück
std::exception inner()	Gibt, falls verfügbar, das Ausnahmeereignis zurück, das dieses Ausnahmeereignis verursacht hat, ansonsten wird der Wert NULL zurückgegeben.

Aufrufen von Schemainformationen

Über die generierte Bibliothek können statische Schemainformationen über die folgenden Klassen abgerufen werden. Alle Methoden werden als `const` deklariert. Die Methoden, die eine der Metadatenklassen zurückgeben, geben ein NULL-Objekt zurück, wenn die entsprechende Eigenschaft nicht existiert.

altova::meta::Attribute 1659
altova::meta::ComplexType 1659
altova::meta::Element 1660
altova::meta::SimpleType 1661

16.6.2 Informationen zu Schema Wrapper-Bibliotheken (C#)

Das Standardmapping von XML-Schematypen auf C#-Datentypen ist:

XML-Schema	C#	Anmerkungen
xs:string	string	
xs:boolean	bool	

© 2016-2022 Altova GmbH

`xs:decimal`	`decimal`	xs:decimal hat keine Bereichsbeschränkung und Genauigkeit, wird aus Gründen der Effizienz auf decimal gemappt.
`xs:float, xs:double`	`double`	
`xs:long`	`long`	
`xs:unsignedLong`	`ulong`	
`xs:int`	`int`	
`xs:unsignedInt`	`uint`	
`xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth`	`Altova.Types.DateTime` 1668	
`xs:duration`	`Altova.Types.Duration` 1673	
`xs:hexBinary and xs:base64Binary`	`byte[]`	Die Kodierung und Dekodierung von Binärdateien erfolgt automatisch.
`xs:anySimpleType`	`string`	

Alle XML-Schematypen, die nicht in der Liste enthalten sind, sind abgeleitete Typen und werden auf denselben C#-Typ wie der entsprechende Basistyp gemappt.

Generierte Klassen

Für jeden Typ im Schema wird eine Klasse generiert, die ein Member für jedes Attribut und Element des Typs enthält. Die Members erhalten den gleichen Namen wie die Attribute oder Elemente im Originalschema (bei möglichen Konflikten wird eine Zahl angehängt). Für Simple und Mixed Types werden Zuweisungs- und Konvertierungsoperatoren generiert. Für Simple Types mit Enumeration Facets können die Methoden `getEnumerationValue()` und `setEnumerationValue(int)` zusammen mit generierten Konstanten für die einzelnen Enumerationswerte verwendet werden. Des Weiteren können Sie mit Hilfe der Methode `StaticInfo()` Schemainformationen als einen der folgenden Typen abrufen:

`Altova.Xml.Meta.SimpleType` 1677
`Altova.Xml.Meta.ComplexType` 1676

Anhand von Complex Typs generierte Klassen enthalten die Methode `SetXsiType()`, mit Hilfe derer Sie das `xsi:type`-Attribut des Typs definieren können. Diese Methode ist nützlich, wenn Sie XML-Instanzelemente anhand eines Derived Type erstellen möchten.

Zusätzlich zu den Klassen für die im XML-Schema deklarierten Typen, wird eine (unten durch "CDoc" identifizierte) Dokument-Klasse generiert. Sie enthält alle möglichen Root-Elemente als Members sowie verschiedene andere Methoden. Nähere Informationen zur Klasse finden Sie unter `[YourSchema].[Doc]` 1678.

Anmerkung: Der tatsächliche Klassenname ist vom Namen des .xsd-Schemas abhängig.

Für jedes Member-Attribut oder Elemente eines Schematyps wird eine neue Klasse generiert. Nähere Informationen zu solchen Klassen finden Sie unter:

[YourSchemaType].MemberAttribute[1681]
[YourSchemaType].MemberElement[1682]

Anmerkung: Der tatsächliche Klassenname ist vom Namen des Schema-Attributs oder -Elements abhängig.

Behandlung von Fehlern

Fehler werden in Form von Ausnahmeereignissen ausgegeben. Im Namespace com.altova sind die folgenden Ausnahmeereignisklassen definiert:

Klasse	Basisklasse	Beschreibung
ConversionException	Exception	Ausnahmeereignis aufgrund eines Typkonvertierungsfehlers
StringParseException	ConversionException	Ein Wert im lexikalischen Bereich kann nicht in einen Wertebereich umgewandelt werden.
DataSourceUnavailableException	System.Exception	Beim Laden einer XML-Instanz ist ein Problem aufgetreten.
DataTargetUnavailableException	System.Exception	Beim Speichern einer XML-Instanz ist ein Problem aufgetreten.

Zusätzlich dazu werden die folgenden .NET-Ausnahmeereignisse verwendet:

Klasse	Beschreibung
System.Exception	Basisklasse für Runtime-Fehler
System.ArgumentException	Es wurde eine Methode mit ungültigen Argumentwerten aufgerufen, oder es gab einen Fehler bei einer Typkonvertierung.
System.FormatException	Ein Wert im lexikalischen Bereich kann nicht in einen Wertebereich umgewandelt werden.
System.InvalidCastException	Ein Wert kann nicht in einen anderen Typ umgewandelt werden.
System.OverflowException	Ein Quellwert kann in der Ziel-Domain nicht dargestellt werden.

Aufrufen von Schemainformationen

Über die generierte Bibliothek können statische Schemainformationen über die folgenden Klassen abgerufen werden.

Altova.Xml.Meta.Attribute[1675]
Altova.Xml.Meta.ComplexType[1676]
Altova.Xml.Meta.Element[1677]
Altova.Xml.Meta.SimpleType[1677]

© 2016-2022 Altova GmbH

Die Eigenschaften, die eine der Metadatenklassen zurückgeben, geben Null zurück, wenn die entsprechende Eigenschaft nicht existiert.

16.6.3 Informationen zu Schema Wrapper-Bibliotheken (Java)

Für das Mappen von XML-Schematypen auf Java-Datentypen wird das folgende Standard-Mapping verwendet:

XML Schema	Java	Remarks
`xs:string`	`String`	
`xs:boolean`	`boolean`	
`xs:decimal`	`java.math.BigDecimal`	
`xs:float, xs:double`	`double`	
`xs:integer`	`java.math.BigInteger`	
`xs:long`	`long`	
`xs:unsignedLong`	`java.math.BigInteger`	Java hat keine unsignierten Typen.
`xs:int`	`int`	
`xs:unsignedInt`	`long`	Java hat keine unsignierten Typen.
`xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth`	`com.altova.types.DateTime` 1684	
`xs:duration`	`com.altova.types.Duration` 1688	
`xs:hexBinary and xs:base64Binary`	`byte[]`	Die Kodierung und Dekoiderung binärer Daten erfolgt automatisch.
`xs:anySimpleType`	`string`	

Alle XML-Schematypen, die nicht in der Liste enthalten sind, sind abgeleitete Typen und werden auf denselben Java-Typ wie der entsprechende Basistyp gemappt.

Generierte Klassen

Für jeden Typ im Schema wird eine Klasse generiert, die ein Member für jedes Attribut und Element des Typs enthält. Die Members erhalten den gleichen Namen wie die Attribute oder Elemente im Originalschema (bei möglichen Konflikten wird eine Zahl angehängt). Für Simple Types werden Zuweisungs- und Konvertierungsoperatoren generiert. Für Simple Types mit Enumeration Facets können die Methoden `getEnumerationValue()` und `setEnumerationValue(int)` zusammen mit generierten Konstanten für die einzelnen Enumerationswerte verwendet werden. Des Weiteren können Sie mit Hilfe der Methode `StaticInfo()` Schemainformationen als einen der folgenden Typen abrufen:

com.altova.xml.meta.SimpleType 1694
com.altova.xml.meta.ComplexType 1693

Anhand von Complex Types generierte Klassen enthalten die Methode `SetXsiType()`, mit Hilfe derer Sie das `xsi:type`-Attribut des Typs definieren können. Diese Methode ist nützlich, wenn Sie XML-Instanzelemente anhand eines Derived Type erstellen möchten.

Zusätzlich zu den Klassen für die im XML-Schema deklarierten Typen, wird eine (unten durch "CDoc" identifizierte) Dokument-Klasse generiert. Sie enthält alle möglichen Root-Elemente als Members sowie verschiedene andere Methoden. Nähere Informationen zur Klasse finden Sie unter com.[YourSchema].[Doc] 1695.

Anmerkung: Der tatsächliche Klassenname ist vom Namen des .xsd-Schemas abhängig.

Für jedes Member-Attribut oder Elemente eines Schematyps wird eine neue Klasse generiert. Nähere Informationen zu solchen Klassen finden Sie unter:

com.[YourSchema].[YourSchemaType].MemberAttribute 1698
com.[YourSchema].[YourSchemaType] 1698

Anmerkung: Der tatsächliche Klassenname ist vom Namen des Schema-Attributs oder -Elements abhängig.

Behandlung von Fehlern

Fehler werden in Form von Ausnahmeereignissen ausgegeben. Im Namespace com.altova sind die folgenden Ausnahmeereignisklassen definiert:

Klasse	Basisklasse	Beschreibung
`SourceInstanceUnvailableException`	Exception	Beim Laden einer XML-Instanz ist ein Problem aufgetreten.
`TargetInstanceUnavailableException`	Exception	Beim Speichern einer XML-Instanz ist ein Problem aufgetreten.

Zusätzlich dazu werden die folgenden Java-Ausnahmeereignisse verwendet:

Klasse	Beschreibung
`java.lang.Error`	Interner Fehler in der Programmlogik (unabhängig von den Eingabedaten)
`java.lang.Exception`	Basisklasse für Runtime-Fehler
`java.lang.IllegalArgumentsException`	Es wurde eine Methode mit ungültigen Argumentwerten aufgerufen, oder es gab einen Fehler bei einer Typkonvertierung.
`java.lang.ArithmeticException`	Ausnahmeereignis aufgrund eines Fehlers bei der Konvertierung eines numerischen Typs.

© 2016-2022 Altova GmbH

Aufrufen von Schemainformationen

Über die generierte Bibliothek können statische Schemainformationen über die folgenden Klassen abgerufen werden.

com.altova.xml.meta.Attribute 1692
com.altova.xml.meta.ComplexType 1693
com.altova.xml.meta.Element 1693
com.altova.xml.meta.SimpleType 1694

Die Eigenschaften, die eine der Metadatenklassen zurückgeben, geben Null zurück, wenn die entsprechende Eigenschaft nicht existiert.

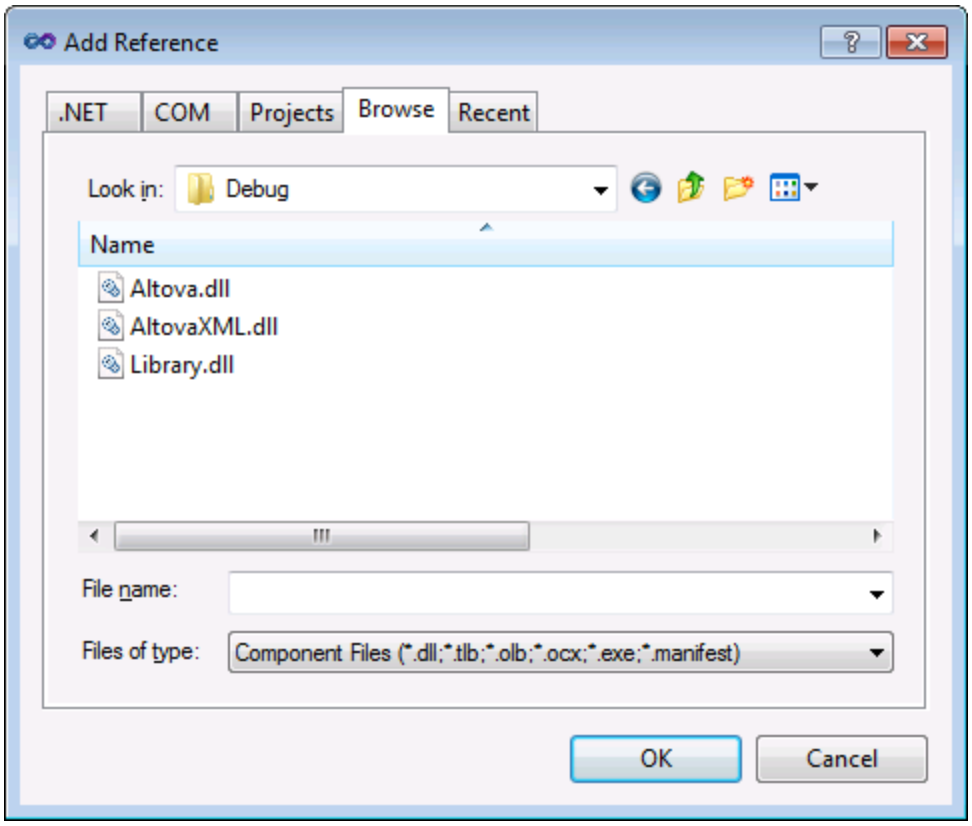
16.6.4 Integrieren von Altova-Bibliotheken in vorhandene Projekte

Um die Altova-Bibliotheken in Ihrem benutzerdefinierten Projekt zu verwenden, referenzieren Sie die Bibliotheken von Ihrem Projekt aus (oder inkludieren Sie diese in Ihr Projekt), wie unten für die einzelnen Sprachen gezeigt.

C#

So integrieren Sie die Altova-Bibliotheken in ein bestehendes C#-Projekt:

1. Nachdem MapForce Code anhand eines Schemas generiert hat (z.B. **YourSchema.xsd**), erstellen Sie die generierte Lösung **YourSchema.sln** in Visual Studio. Diese Lösung befindet sich in einem Projektordner, der denselben Name wie das Schema hat.
2. Klicken Sie in Visual Studio mit der rechten Maustaste auf Ihr bestehendes Projekt und wählen Sie den Befehl **Add Reference**.
3. Suchen Sie auf dem Register Browse nach den folgenden Bibliotheken: **Altova.dll**, **AltovaXML.dll** und **YourSchema.dll** im Ausgabeverzeichnis **der generierten Projekte (z.B. bin\Debug).**



C++

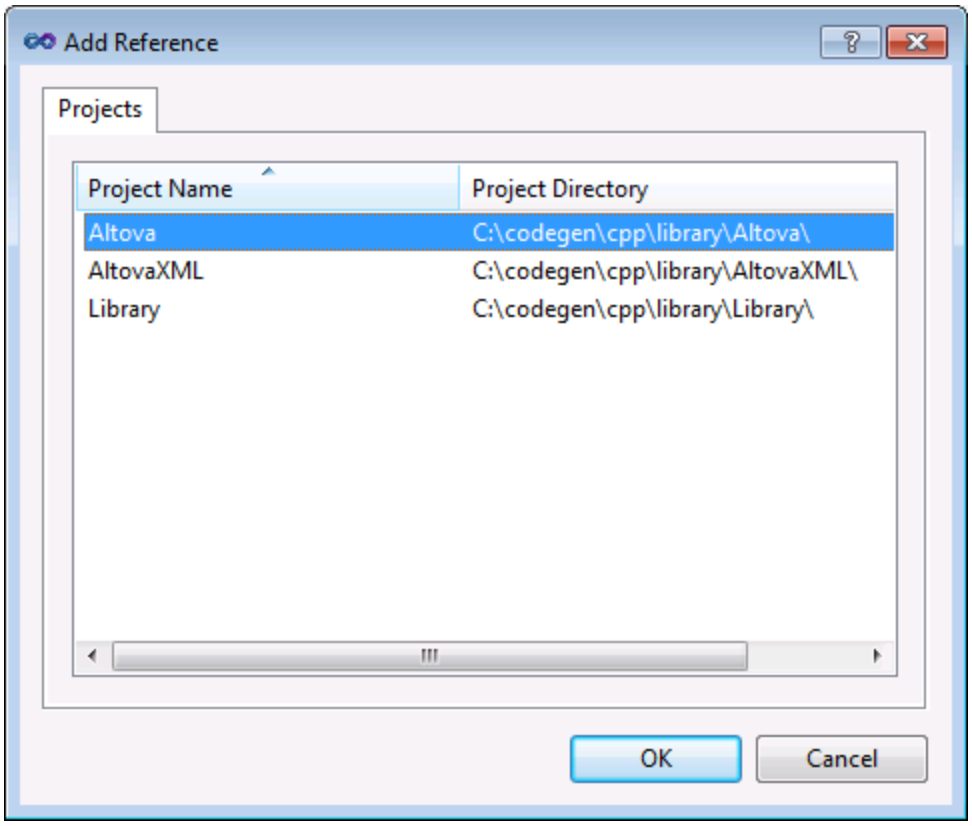
Die einfachste Methode, um die Bibliotheken in ein bestehendes C++-Projekt zu integrieren ist, die generierten Projektdateien zu Ihrer Lösung hinzuzufügen. Angenommen, Sie haben anhand eines Schemas namens **Library.xsd** Code generiert und **c:\codegen\cpp\library** als Zielverzeichnis ausgewählt. In diesem Fall stehen die generierten Bibliotheken unter folgendem Pfad zur Verfügung:

- c:\codegen\cpp\library\Altova.vcxproj
- c:\codegen\cpp\library\AltovaXML\AltovaXML.vcxproj
- c:\codegen\cpp\library\Library.vcxproj

Öffnen Sie zuerst die generierte Lösung **c:\codegen\cpp\library\Library.sln** und bauen Sie sie in Visual Studio.

Öffnen Sie anschließend Ihre bestehende Visual Studio-Lösung (in diesem Beispiel in Visual Studio 2010), klicken Sie mit der rechten Maustaste darauf und wählen Sie **Add | Existing Project** und fügen Sie die oben aufgelisteten Projektdateien einzeln nacheinander hinzu. Haben Sie einen Moment Geduld, während Visual Studio die Dateien parst. Klicken Sie anschließend mit der rechten Maustaste auf Ihr Projekt und wählen Sie **Properties**. Wählen Sie im Dialogfeld "Property Pages" die Option **Common Properties | Framework and References** und klicken Sie anschließend auf **Add New Reference**. Wählen Sie anschließend jedes einzelne der folgenden Projekte aus und fügen Sie es hinzu: *Altova*, *AltovaXML* und *Library*.

© 2016-2022 Altova GmbH



Informationen zur Verwendung von Funktionalitäten aus einer benutzerdefinierten Bibliothek finden Sie außerdem in der MSDN-Dokumentation zu Ihrer jeweiligen Visual Studio-Version, z.B.:

- Wenn Sie statische Bibliotheken generieren möchten, siehe https://msdn.microsoft.com/en-us/library/ms235627(v=vs.100).aspx
- Wenn Sie dynamische Bibliotheken generieren möchten: siehe https://msdn.microsoft.com/en-us/library/ms235636(v=vs.100).aspx

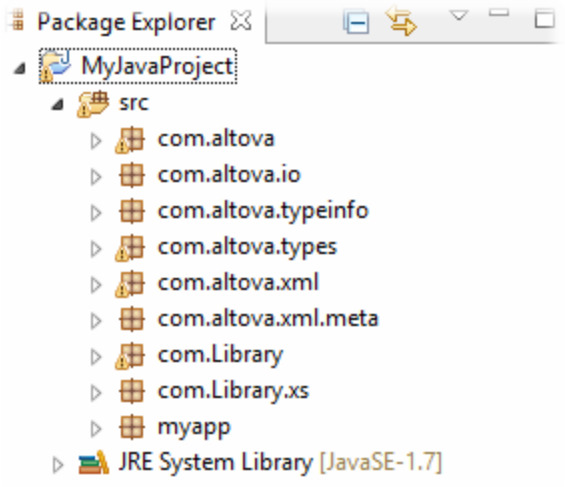
Die Option zum Generieren statischer oder dynamischer Bibliotheken finden Sie in den Code Generator-Optionen (siehe Code Generator-Optionen 1701).

Java

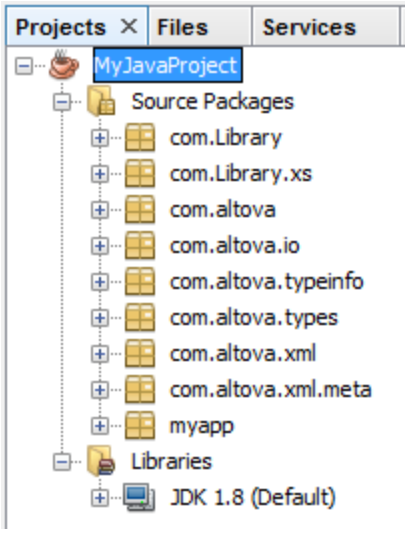
Eine der Möglichkeiten Altova-Pakete in Ihr Java-Projekt zu integrieren, ist, das **com**-Verzeichnis in das Verzeichnis zu kopieren, das die Quellpakete Ihres Java-Projekts enthält (z.B. **C:\Workspace\MyJavaProject\src**). Angenommen, Sie haben Code in **c:\codegen\java\library** generiert. In diesem Fall befinden sich die generierten Altova-Klassen unter **c:\codegen\java\library\com**.

Aktualisieren Sie das Projekt, nachdem Sie die Bibliotheken kopiert haben. Um das Projekt in Eclipse zu aktualisieren, wählen Sie es im Paket-Explorer aus und drücken Sie **F5. Um das Projekt in NetBeans IDE 8.0 zu aktualisieren, wählen Sie den Menübefehl Source | Scan for External Changes.**

Nach dem Kopiervorgang stehen die Altova-Pakete (bei Eclipse) im Paket-Explorer oder (bei NetBeans) im Projektbereich unter "Source Packages" zur Verfügung.



Altova-Pakete in Eclipse 4.4



Altova-Pakete in NetBeans IDE 8.0.2

16.6.5 Beispiel: Verwendung der Schema Wrapper-Bibliotheken

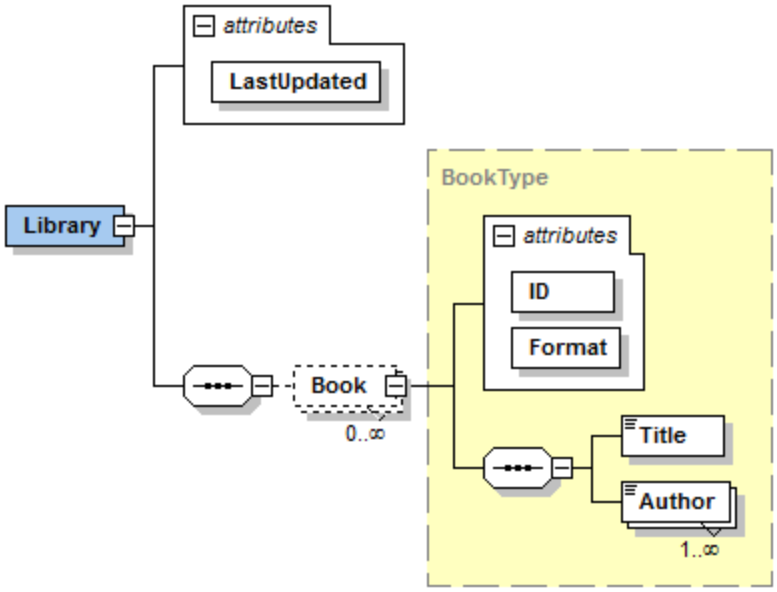
Anhand dieses Beispiels wird gezeigt, wie Sie mit Hilfe der generierten Schema Wrapper-Bibliotheken schemakonforme XML-Dokumente programmatisch schreiben oder lesen. Werfen Sie zuerst einen Blick auf die Struktur des inkludierten Beispielschemas, bevor Sie den Beispielcode verwenden.

In dem in diesem Beispiel verwendeten Schema ist eine Bibliothek mit Büchern beschrieben. Unten sehen Sie die vollständige Definition des Schemas. Speichern Sie dieses Codefragment als **`Library.xsd`**, wenn Sie dieselben Ergebnisse wie in diesem Beispiel erhalten möchten. Sie benötigen dieses Schema zur Generierung der in diesem Beispiel verwendeten Codebibliotheken.

© 2016-2022 Altova GmbH

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.nanonull.com/LibrarySample"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.nanonull.com/LibrarySample" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:element name="Library">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Book" type="BookType" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="LastUpdated" type="xs:dateTime"/>
      </xs:complexType>
   </xs:element>
   <xs:complexType name="BookType">
      <xs:sequence>
         <xs:element name="Title" type="xs:string"/>
         <xs:element name="Author" type="xs:string" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="ID" type="xs:integer" use="required"/>
      <xs:attribute name="Format" type="BookFormatType" use="required"/>
   </xs:complexType>
   <xs:complexType name="DictionaryType">
      <xs:complexContent>
         <xs:extension base="BookType">
            <xs:sequence>
               <xs:element name="FromLang" type="xs:string"/>
               <xs:element name="ToLang" type="xs:string"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:simpleType name="BookFormatType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="Hardcover"/>
         <xs:enumeration value="Paperback"/>
         <xs:enumeration value="Audiobook"/>
         <xs:enumeration value="E-book"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

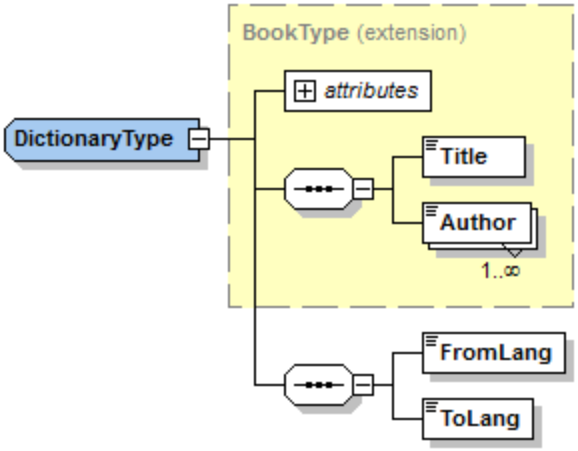
Library ist ein Root-Element vom Typ `complexType`. Grafisch kann es in der Schema-Ansicht von XMLSpy folgendermaßen dargestellt werden:



Wie oben gezeigt, hat die Bibliothek ein Attribut **LastUpdated** (definiert als `xs:dateTime`) und enthält eine Sequenz von Büchern. Jedes Buch hat den Typ `xs:complexType` und verfügt über zwei Attribute: eine **ID** (definiert als `xs:integer`) und ein **Format**. Das Format eines Buchs kann hardcover, paperback, audiobook oder e-book sein. Im Schema ist **Format** als `xs:simpleType` definiert, für den eine Enumeration der oben angeführten Werte verwendet wird..

Außerdem hat jedes Buch ein **Title**-Element (definiert als `xs:string`) sowie ein oder mehrere **Author**-Elemente (definiert als `xs:string`).

Die Bibliothek kann auch Bücher enthalten, die Wörterbücher sind. Wörterbücher haben den Typ `DictionaryType`, der durch Erweiterung vom Typ `BookType` abgeleitet ist. Ein Wörterbuch erbt also alle Attribute und Elemente eines Buchs plus zwei weitere Elemente: **FromLang** und **ToLang**, wie unten gezeigt.



In den Elementen **FromLang** und **ToLang** ist die Quell- und Zielsprache des Wörterbuchs gespeichert.

 © 2016-2022 Altova GmbH

Eine XML-Instanzdatei, die gemäß dem obigen Schema gültig ist, könnte also wie im Codefragment unten aussehen (Vorausgesetzt sie befindet sich im selben Verzeichnis wie die Schemadatei):

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-
02-03T17:10:08.4977404">
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

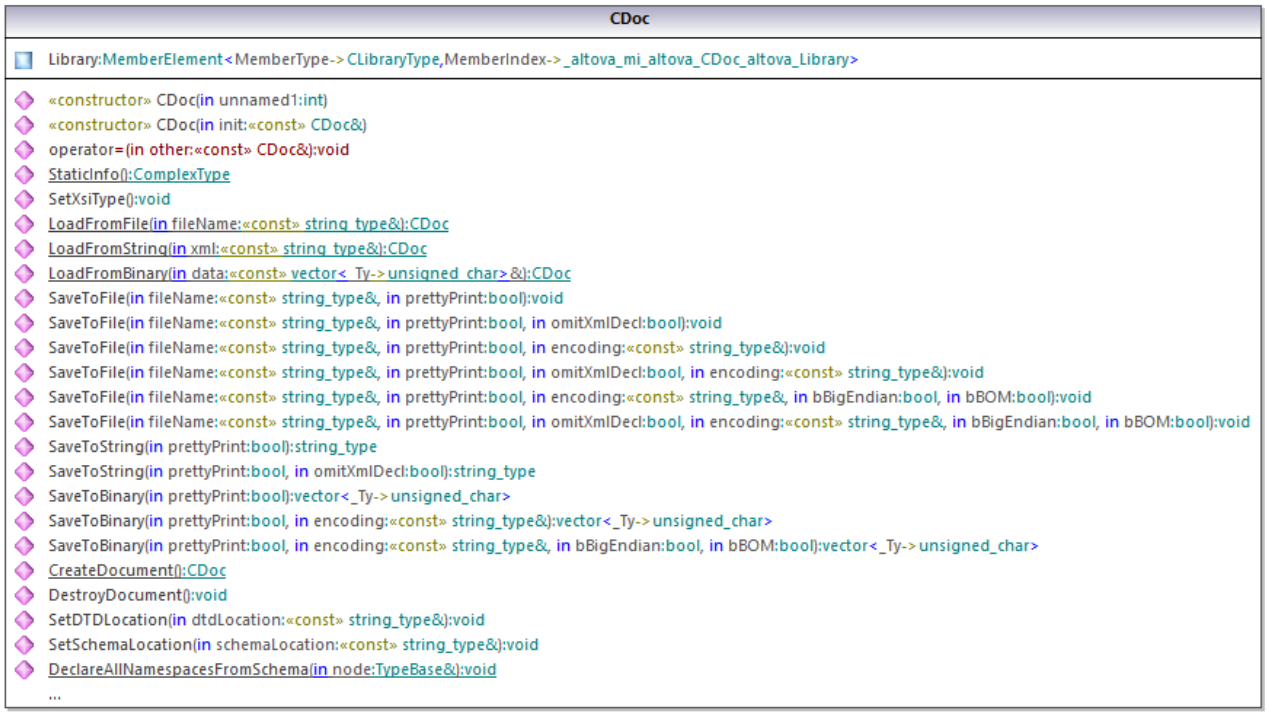
In den nächsten Kapiteln wird beschrieben, wie eine solche Datei programmatisch gelesen bzw. in diese Datei programmatisch geschrieben werden kann. Generieren Sie zuerst den Schema Wrapper-Code anhand des obigen Schemas. Gehen Sie dabei vor, wie unter Generieren von Code anhand von XML-Schemas oder DTDs 1607 beschrieben.

16.6.5.1 Lesen und Schreiben von XML-Dokumenten (C++)

Nachdem Sie anhand des Schemas "Library" (siehe Beispielschema 1620) Code generiert haben, wird eine C#-Testapplikation sowie eine Reihe von unterstützenden Altova-Bibliotheken generiert.

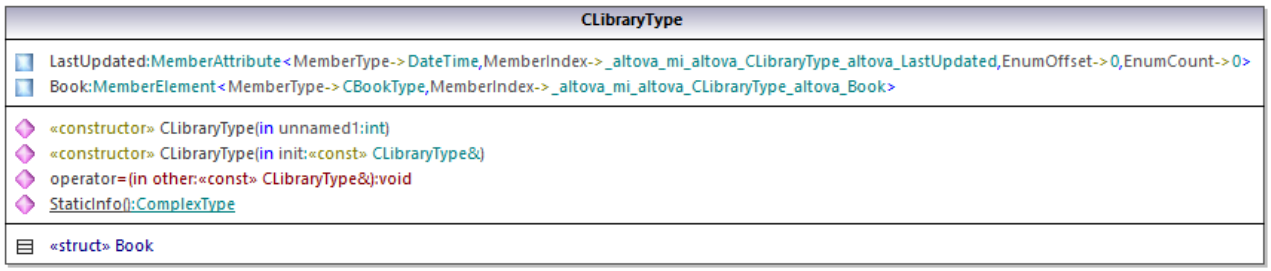
Informationen zu den generierten C++-Bibliotheken

Die zentrale Klasse des generierten Codes ist die Klasse `CDoc`. Sie repräsentiert das XML-Dokument. Eine solche Klasse wird für jedes Schema generiert. Ihr Name hängt vom Namen der Schemadatei ab. Wie im Diagramm gezeigt, bietet diese Klasse Methoden zum Laden von Dokumenten aus Dateien, Binär-Streams oder Strings (oder zum Speichern von Dokumenten in Dateien, Streams, Strings). In der Referenz zur Klasse (`[YourSchema]::[CDoc]` 1662) finden Sie eine Beschreibung aller von dieser Klasse bereitgestellten Members.



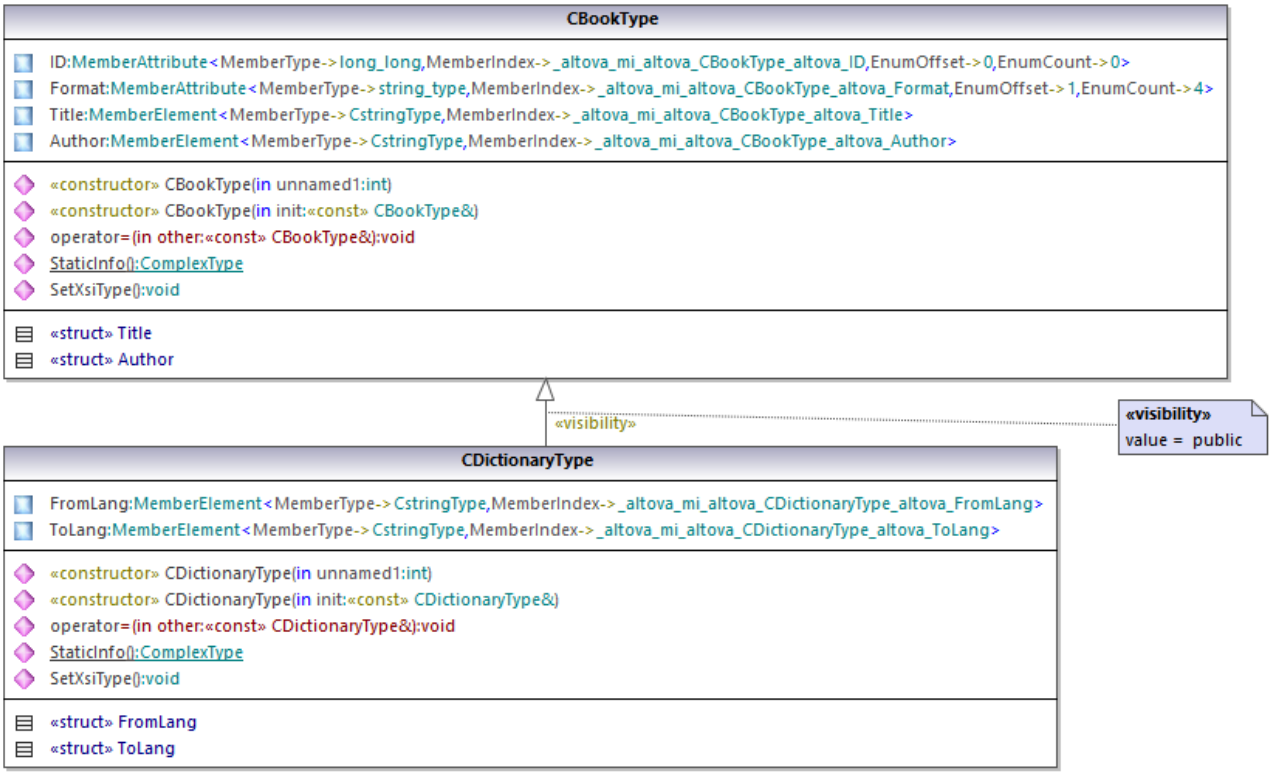
Das Feld `Library` der Klasse `CDoc` stellt die eigentliche Root des Dokuments dar. **Library** ist ein Element in der XML-Datei, daher hat es im C++-Code eine Vorlagenklasse als Typ (`MemberElement`). Die Vorlagenklasse stellt Methoden und Eigenschaften für die Interaktion mit dem Element **Library** bereit. Im Allgemeinen erhält jedes Attribut und jedes Element eines bestimmten Typs im Schema im generierten Code die Vorlagenklasse `MemberAttribute` bzw. `MemberElement`. Nähere Informationen dazu finden Sie in der Referenz zur Klasse unter `[YourSchema]::MemberAttribute`[1665] und `[YourSchema]::MemberElement`[1666].

Die Klasse `CLibraryType` wird anhand des complexType **LibraryType** im Schema generiert. Beachten Sie, dass die Klasse `CLibraryType` ein Feld `Book` und ein Feld `LastUpdated` enthält. Gemäß der oben erwähnten Logik entsprechen diese dem Element **Book** und dem Attribut **LastUpdated** im Schema. Sie ermöglichen die programmatische Bearbeitung (Anhängen, Entfernen, usw.) von Elementen und Attributen im XML-Instanzdokument.



Der Typ `DictionaryType` ist ein complexType, der im Schema von **BookType** abgeleitet wurde, daher wird diese Beziehung auch in den generierten Klassen übernommen. Im Diagramm sehen Sie, dass die Klasse `CDictionaryType` die Klasse `CBookType` erbt.

 © 2016-2022 Altova GmbH



Wenn in Ihrem XML-Schema simpleTypes als Enumerationen definiert sind, so stehen die enumerierten Werte im generierten Code als `Enum`-Werte zur Verfügung. In dem in diesem Beispiel verwendeten Schema gibt es die Buchformate hardcover, paperback, e-book, usw. Im generierten Code stehen diese Werte folglich in Form einer Enum, d.h. als Member der Klasse `CBookFormatType`, zur Verfügung.

Schreiben eines XML-Dokuments

1. Öffnen Sie die anhand des Schemas "Library" generierte Lösung **LibraryTest.sln** in Visual Studio.

Beim Erstellen eines Prototyps einer Applikation anhand eines häufig geänderten XML-Schemas müssen Sie eventuell immer wieder Code im selben Verzeichnis generieren, damit die Änderungen am Schema sofort im Code berücksichtigt werden. Beachten Sie, dass die generierte Testapplikation und die Altova-Bibliotheken jedes Mal, wenn Sie Code im selben Zielverzeichnis generieren, überschrieben werden. Fügen Sie daher keinen Code zur generierten Testapplikation hinzu, sondern integrieren Sie stattdessen die Altova-Bibliotheken in Ihr Projekt (siehe Integrieren von Schema Wrapper-Bibliotheken 1617).

2. Öffnen Sie die Datei **LibraryTest.cpp** im Solution Explorer und bearbeiten Sie die Methode `Example()` wie unten gezeigt.

```
#include <ctime> // required to get current time
using namespace Doc; // required to work with Altova libraries

void Example()
```

```
{
   // Create a new, empty XML document
   CDoc libDoc = CDoc::CreateDocument();

   // Create the root element <Library> and add it to the document
   CLibraryType lib = libDoc.Library.append();

   // Get current time and set the "LastUpdated" attribute using Altova classes
   time_t t = time(NULL);
   struct tm * now = localtime( & t );
   altova::DateTime dt = altova::DateTime(now->tm_year + 1900, now->tm_mon + 1, now-
>tm_mday, now->tm_hour, now->tm_min, now->tm_sec);
   lib.LastUpdated = dt;

   // Create a new <Book> and add it to the library
   CBookType book = lib.Book.append();

   // Set the "ID" attribute of the book
   book.ID = 1;

   // Set the "Format" attribute of the <Book> using an enumeration constant
   book.Format.SetEnumerationValue( CBookFormatType::k_Paperback );

   // Add the <Title> and <Author> elements, and set values
   book.Title.append() = _T("The XML Spy Handbook");
   book.Author.append() = _T("Altova");

   // Append a dictionary (book of derived type) and populate its attributes and elements
   CDictionaryType dictionary = CDictionaryType(lib.Book.append().GetNode());
   dictionary.ID = 2;
   dictionary.Format.SetEnumerationValue( CBookFormatType::k_E_book);
   dictionary.Title.append() = _T("English-German Dictionary");
   dictionary.Author.append() = _T("John Doe");
   dictionary.FromLang.append() = _T("English");
   dictionary.ToLang.append() = _T("German");

   // Since dictionary a derived type, set the xsi:type attribute of the book element
   dictionary.SetXsiType();

   // Optionally, set the schema location
   libDoc.SetSchemaLocation(_T("Library.xsd"));

   // Save the XML document to a file with default encoding (UTF-8),
   // "true" causes the file to be pretty-printed.
   libDoc.SaveToFile(_T("GeneratedLibrary.xml"), true);

   // Destroy the document
   libDoc.DestroyDocument();
}
```

3. Drücken Sie **F5,** um den Debugger zu starten. Wenn der Code erfolgreich ausgeführt wurde, wird im Lösungsausgabeverzeichnis die Datei **GeneratedLibrary.xml** erstellt.

© 2016-2022 Altova GmbH

Lesen eines XML-Dokuments

1. Öffnen Sie die Lösung **LibraryTest.sln** in Visual Studio.
2. Speichern Sie den unten gezeigten Code unter dem Namen **Library1.xml** in einem Verzeichnis, das vom Programmcode gelesen werden kann (z.B im selben Verzeichnis wie **LibraryTest.sln**).

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-
02-03T17:10:08.4977404">
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

3. Öffnen Sie die Datei **LibraryTest.cpp** im Solution Explorer und bearbeiten Sie die Methode `Example()` wie unten gezeigt.

```
using namespace Doc;
void Example()
{
   // Load XML document
   CDoc libDoc = CDoc::LoadFromFile(_T("Library1.xml"));

   // Get the first (and only) root element <Library>
   CLibraryType lib = libDoc.Library.first();

   // Check whether an element exists:
   if (!lib.Book.exists())
   {
      tcout << "This library is empty." << std::endl;
      return;
   }

   // iteration: for each <Book>...
   for (Iterator<CBookType> itBook = lib.Book.all(); itBook; ++itBook)
   {
      // output values of ISBN attribute and (first and only) title element
      tcout << "ID: " << itBook->ID << std::endl;
      tcout << "Title: " << tstring(itBook->Title.first()) << std::endl;

      // read and compare an enumeration value
```

```
      if (itBook->Format.GetEnumerationValue() == CBookFormatType::k_Paperback)
         tcout << "This is a paperback book." << std::endl;

      // for each <Author>...
      for (CBookType::Author::iterator itAuthor = itBook->Author.all(); itAuthor; +
+itAuthor)
         tcout << "Author: " << tstring(itAuthor) << std::endl;

      // alternative: use count and index
      for (unsigned int j = 0; j < itBook->Author.count(); ++j)
         tcout << "Author: " << tstring(itBook->Author[j]) << std::endl;
   }

   // Destroy the document
   libDoc.DestroyDocument();
}
```

4. Drücken Sie **F5,** um den Debugger zu starten.

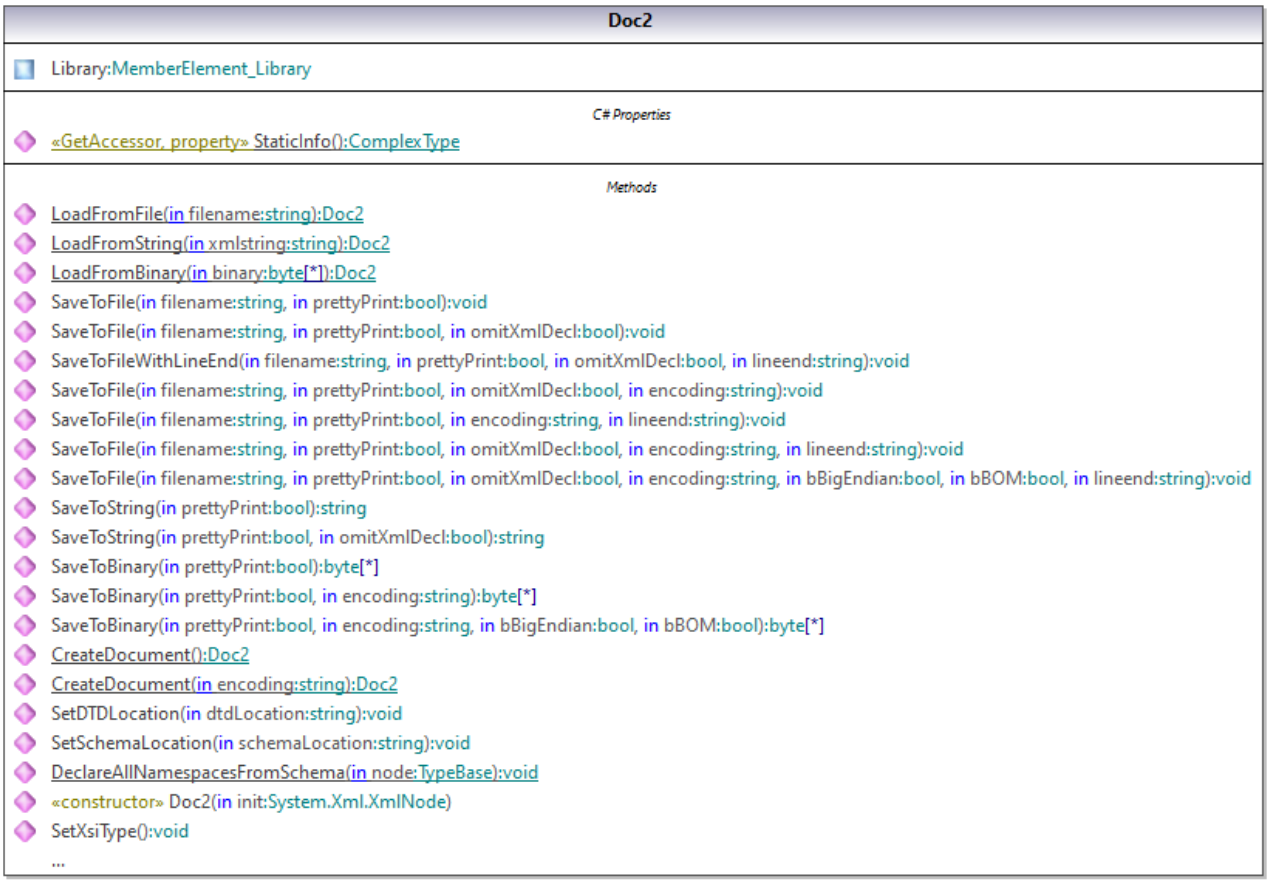
16.6.5.2 Lesen und Schreiben von XML-Dokumenten (C#)

Nachdem Sie anhand des Schemas "Library" (siehe Beispielschema[1620]) Code generiert haben, wird eine C#-Testapplikation sowie eine Reihe von unterstützenden Altova-Bibliotheken generiert.

Informationen zu den generierten C#-Bibliotheken

Die zentrale Klasse des generierten Codes ist die Klasse `Doc2`. Sie repräsentiert das XML-Dokument. Eine solche Klasse wird für jedes Schema generiert. Ihr Name hängt vom Namen der Schemadatei ab. Beachten Sie, dass diese Klasse den Namen `Doc2` erhält, um einen möglichen Konflikt mit dem Namespace-Namen zu vermeiden. Wie im Diagramm gezeigt, bietet diese Klasse Methoden zum Laden von Dokumenten aus Dateien, Binär-Streams oder Strings (oder zum Speichern von Dokumenten in Dateien, Streams, Strings). In der Referenz zur Klasse ([YourSchema].[Doc][1678]) finden Sie eine Beschreibung dieser Klasse.

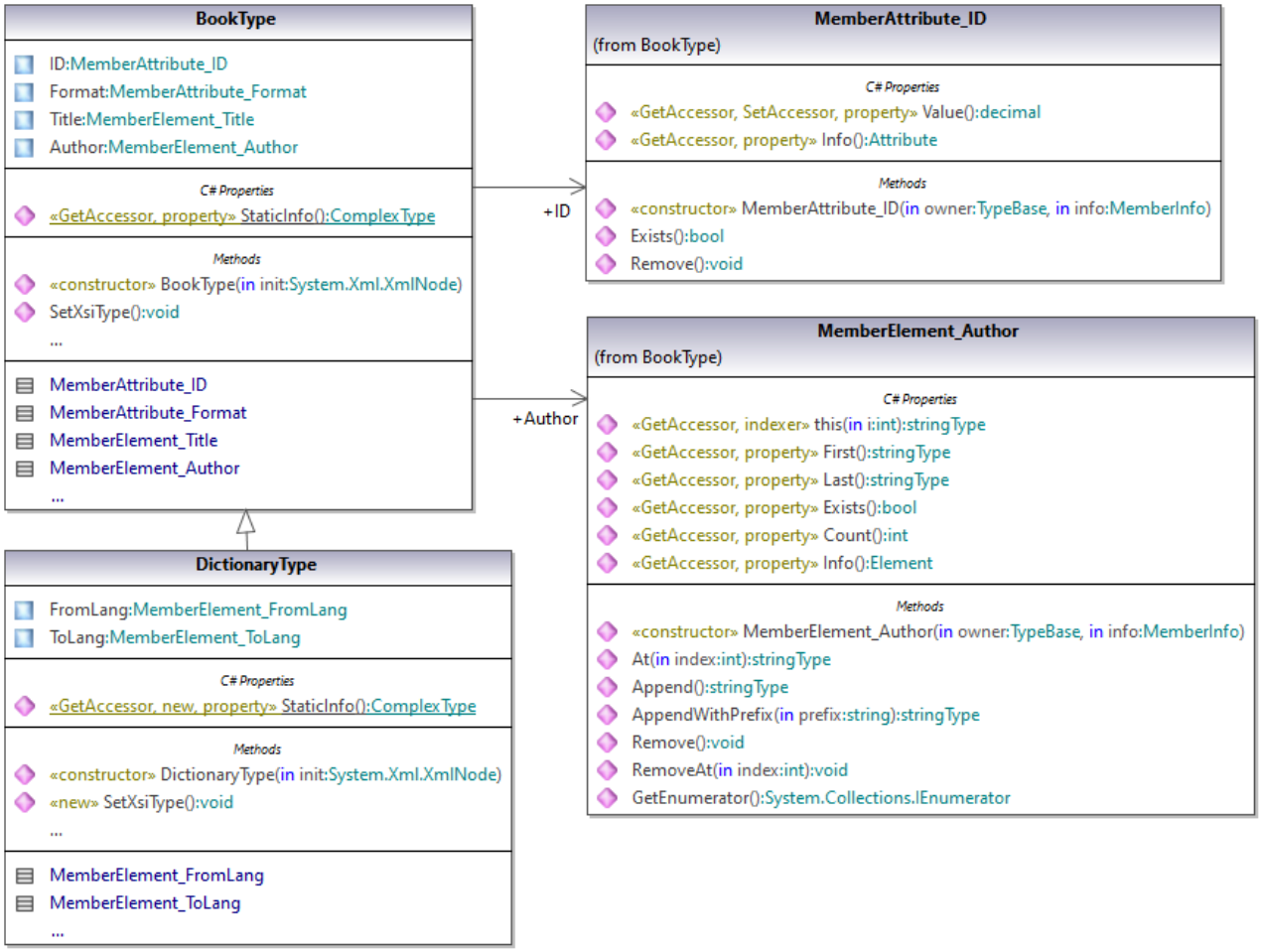
© 2016-2022 Altova GmbH



Der Member `Library` der Klasse `Doc2` stellt die eigentliche Root des Dokuments dar.

Gemäß den im Kapitel Informationen zu Schema Wrapper-Bibliotheken (C#) [1612] angeführten Codegenerierungsregeln werden für jedes Attribut und jedes Element eines Typs Member-Klassen generiert. Der Name solcher Member-Klassen erhält im generierten Code das Präfix `MemberAttribute_` bzw. `MemberElement_`. Beispiele für solche Klassen sind `MemberAttribute_ID` und `MemberElement_Author`, die anhand des Elements **Author** bzw. des Attributs **ID** eines Buchs generiert wurden (Im Diagramm unten sind dies unter `BookType` verschachtelte Klassen). Sie ermöglichen die programmatische Bearbeitung (z.B. Anhängen, Entfernen, Wert definieren, usw.) der entsprechenden Elemente und Attribute im XML-Instanzdokument. Nähere Informationen dazu finden Sie in der Referenz zur Klasse unter `[YourSchemaType].MemberAttribute` [1681] und `[YourSchemaType].MemberElement` [1682].

Der Typ **DictionaryType** ist ein complexType, der im Schema von **BookType** abgeleitet wurde, daher wird diese Beziehung auch in den generierten Klassen übernommen. Im Diagramm unten sehen Sie, dass die Klasse `DictionaryType` die Klasse `BookType` erbt.



Wenn in Ihrem XML-Schema simpleTypes als Enumerationen definiert sind, so stehen die enumerierten Werte im generierten Code als `Enum`-Werte zur Verfügung. In dem in diesem Beispiel verwendeten Schema gibt es die Buchformate hardcover, paperback, e-book, usw. Im generierten Code stehen diese Werte folglich in Form einer Enum, d.h. als Member der Klasse `CBookFormatType, zur Verfügung.`

Schreiben eines XML-Dokuments

1. Öffnen Sie die anhand des Schemas "Library" generierte Lösung **LibraryTest.sln** in Visual Studio.

Beim Erstellen eines Prototyps einer Applikation anhand eines häufig geänderten XML-Schemas müssen Sie eventuell immer wieder Code im selben Verzeichnis generieren, damit die Änderungen am Schema sofort im Code berücksichtigt werden. Beachten Sie, dass die generierte Testapplikation und die Altova-Bibliotheken jedes Mal, wenn Sie Code im selben Zielverzeichnis generieren, überschrieben werden. Fügen Sie daher keinen Code zur generierten Testapplikation hinzu, sondern integrieren Sie stattdessen die Altova-Bibliotheken in Ihr Projekt (siehe Integrieren von Schema Wrapper-Bibliotheken 1617).

2. Öffnen Sie die Datei **LibraryTest.cs** im Solution Explorer und bearbeiten Sie die Methode `Example()` wie unten gezeigt.

 © 2016-2022 Altova GmbH

```
protected static void Example()
{
    // Create a new XML document
    Doc2 doc = Doc2.CreateDocument();
    // Append the root element
    LibraryType root = doc.Library.Append();

    // Create the generation date using Altova DateTime class
    Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);
    // Append the date to the root
    root.LastUpdated.Value = dt;

    // Add a new book
    BookType book = root.Book.Append();
    // Set the value of the ID attribute
    book.ID.Value = 1;
    // Set the format of the book (enumeration)
    book.Format.EnumerationValue = BookFormatType.EnumValues.eHardcover;
    // Set the Title and Author elements
    book.Title.Append().Value = "The XMLSpy Handbook";
    book.Author.Append().Value = "Altova";

    // Append a dictionary (book of derived type) and populate its attributes and
elements
    DictionaryType dictionary = new DictionaryType(root.Book.Append().Node);
    dictionary.ID.Value = 2;
    dictionary.Title.Append().Value = "English-German Dictionary";
    dictionary.Format.EnumerationValue = BookFormatType.EnumValues.eE_book;
    dictionary.Author.Append().Value = "John Doe";
    dictionary.FromLang.Append().Value = "English";
    dictionary.ToLang.Append().Value = "German";
    // Since it's a derived type, make sure to set the xsi:type attribute of the
book element
    dictionary.SetXsiType();

    // Optionally, set the schema location (adjust the path if
    // your schema is not in the same folder as the generated instance file)
    doc.SetSchemaLocation("Library.xsd");

    // Save the XML document with the "pretty print" option enabled
    doc.SaveToFile("GeneratedLibrary.xml", true);
}
```

3. Drücken Sie **F5,** um den Debugger zu starten. Wenn der Code erfolgreich ausgeführt wurde, wird im Lösungsausgabeverzeichnis (normalerweise **bin/Debug**) die Datei **GeneratedLibrary.xml** erstellt.

Lesen eines XML-Dokuments

1. Öffnen Sie die Lösung **LibraryTest.sln** in Visual Studio.
2. Speichern Sie den unten gezeigten Code unter dem Namen **Library.xml** im Ausgabeverzeichnis (standardmäßig **bin/Debug**). Dies ist die Datei, die vom Programmcode gelesen wird.

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-
02-03T17:10:08.4977404">
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

3. Öffnen Sie die Datei **LibraryTest.cs** im Solution Explorer und bearbeiten Sie die Methode `Example()` wie unten gezeigt.

```
protected static void Example()
{
    // Load the XML file
    Doc2 doc = Doc2.LoadFromFile("Library.xml");
    // Get the root element
    LibraryType root = doc.Library.First;

    // Read the library generation date
    Altova.Types.DateTime dt = root.LastUpdated.Value;
    string dt_as_string = dt.ToString(DateTimeFormat.W3_dateTime);
    Console.WriteLine("The library generation date is: " + dt_as_string);

    // Iteration: for each <Book>...
    foreach (BookType book in root.Book)
    {
      // Output values of ID attribute and (first and only) title element
      Console.WriteLine("ID:    " + book.ID.Value);
      Console.WriteLine("Title: " + book.Title.First.Value);

      // Read and compare an enumeration value
      if (book.Format.EnumerationValue == BookFormatType.EnumValues.ePaperback)
         Console.WriteLine("This is a paperback book.");

      // Iteration: for each <Author>
      foreach (xs.stringType author in book.Author)
         Console.WriteLine("Author: " + author.Value);

      // Determine if this book is of derived type
      if (book.Node.Attributes.GetNamedItem("xsi:type") != null)
      {
         // Find the value of the xsi:type attribute
```


© 2016-2022 Altova GmbH

```
         string xsiTypeValue =
book.Node.Attributes.GetNamedItem("xsi:type").Value;
         // Get the namespace URI and the lookup prefix of this namespace
         string namespaceUri = book.Node.NamespaceURI;
         string prefix = book.Node.GetPrefixOfNamespace(namespaceUri);

         // if this book has DictionaryType
         if (namespaceUri == "http://www.nanonull.com/LibrarySample" &&
xsiTypeValue.Equals(prefix + ":DictionaryType"))
         {
            // output additional fields
            DictionaryType dictionary = new DictionaryType(book.Node);
            Console.WriteLine("Language from: " +
dictionary.FromLang.First.Value);
            Console.WriteLine("Language to: " + dictionary.ToLang.First.Value);
         }
         else
         {
            throw new Exception("Unexpected book type");
         }
      }
   }

   Console.ReadLine();
}
```

4. Drücken Sie **F5,** um den Debugger zu starten. Wenn der Code erfolgreich ausgeführt wurde, wird **Library.xml** vom Programmcode gelesen und ihr Inhalt wird als Konsolenausgabe angezeigt.

Lesen und Schreiben von Elemente und Attributen

Die Werte von Attributen und Elementen können über die Eigenschaft `Value` der generierten Member-Element- bzw. Attribut-Klasse gelesen werden, z.B.:

```
// output values of ID attribute and (first and only) title element
System.out.println("ID:    " + book.ID.Value);
Console.WriteLine("Title: " + book.Title.First.Value);
```

Um in diesem konkreten Beispiel den Wert des Elements **Title** abzurufen, haben wir auch die Methode `First()` verwendet, da es sich hier um des erste (und einzige) **Title**-Element eines Buchs handelt. In Fällen, in denen ein bestimmtes Element nach dem Index aus einer Liste gewählt werden soll, verwenden Sie die Methode `At()`.

Die für die einzelnen Member-Elemente eines Typs generierte Klasse implementiert die Standardschnittstelle `System.Collections.IEnumerable`. Auf diese Art können mehrere Elemente desselben Typs in einer Schleife verarbeitet werden. In diesem konkreten Beispiel können Sie alle Bücher eines Library-Objekts folgendermaßen in einer Schleife verarbeiten:

```
// Iteration: for each <Book>...
foreach (BookType book in root.Book)
{
```

```
    // your code here...
}
```

Mit Hilfe der Methode `Append()`können Sie ein neues Element hinzufügen. Im folgenden Code wird z.B. das Root-Element an das Dokument angehängt:

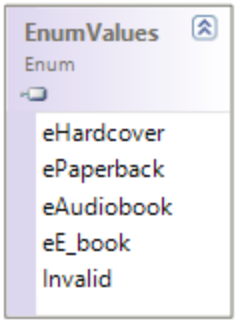
```
// Append the root element to the library
LibraryType root = doc.Library.Append();
```

Sie können den Wert eines Attributs (wie ID in diesem Beispiel) folgendermaßen definieren:

```
// Set the value of the ID attribute
book.ID.Value = 1;
```

Lesen und Schreiben von Enumerationswerten

Wenn in Ihrem XML-Schema simpleTypes als Enumerationen definiert sind, so stehen die enumerierten Werte im generierten Code als `Enum`-Werte zur Verfügung. In dem in diesem Beispiel verwendeten Schema gibt es die Buchformate hardcover, paperback, e-book, usw. Im generierten Code stehen diese Werte folglich in Form einer Enum zur Verfügung.



Mit Hilfe von Code wie dem unten gezeigten können Sie einem Objekt Enumerationswerte zuweisen:

```
// Set the format of the book (enumeration)
book.Format.EnumerationValue = BookFormatType.EnumValues.eHardcover;
```

Solche Enumerationswerte können folgendermaßen aus XML-Instanzdokumenten ausgelesen werden:

```
// Read and compare an enumeration value
if (book.Format.EnumerationValue == BookFormatType.EnumValues.ePaperback)
Console.WriteLine("This is a paperback book.");
```

Wenn eine "if"-Bedingung nicht genügt, erstellen Sie einen Switch, um jeden Enumerationswert zu ermitteln und wie erforderlich zu verarbeiten.

© 2016-2022 Altova GmbH

Arbeiten mit den Typen xs:dateTime und xs:duration

Wenn im Schema, anhand dessen Sie Code generiert haben, Uhrzeit- und Zeitdauer-Typen wie `xs:dateTime` oder `xs:duration` vorkommen, so werden diese im generierten Code in native Altova-Klassen konvertiert. Gehen Sie daher folgendermaßen vor, um einen Datums- oder Zeitdauerwert in das XML-Dokument zu schreiben:

1. Erstellen Sie ein `Altova.Types.DateTime`[1668]- oder `Altova.Types.Duration`[1673]-Objekt (entweder anhand von `System.DateTime` oder durch Verwendung von Teilen wie Stunden und Minuten; nähere Informationen siehe `Altova.Types.DateTime`[1668] und `Altova.Types.Duration`[1673]).
2. Definieren Sie das Objekt als Wert des benötigten Elements oder Attributs, z.B.:

```
// Create the library generation date using Altova DateTime class
Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);
// Append the date to the root
root.LastUpdated.Value = dt;
```

Um ein Datum oder eine Zeitdauer aus einem XML-Dokument zu lesen, gehen Sie folgendermaßen vor:

1. Deklarieren Sie den Elementwert (oder den Attributwert) als `Altova.Types.DateTime`[1668]- oder `Altova.Types.Duration`[1673]-Objekt.
2. Formatieren Sie das benötige Element oder Attribut, z.B.:

```
// Read the library generation date
Altova.Types.DateTime dt = root.LastUpdated.Value;
string dt_as_string = dt.ToString(DateTimeFormat.W3_dateTime);
Console.WriteLine("The library generation date is: " + dt_as_string);
```

Nähere Informationen dazu finde Sie in der Referenz zu den Klassen `Altova.Types.DateTime`[1668] und `Altova.Types.Duration`[1673].

Arbeiten mit abgeleiteten Typen

Wenn in Ihrem XML-Schema abgeleitete Typen (derived types) definiert sind, so können Sie die Typableitung in programmatisch erstellten oder geladenen XML-Dokumenten beibehalten. Im folgenden Codefragment wird gezeigt, wie Sie anhand des in diesem Beispiel verwendeten Schemas ein neues Buch mit dem abgeleiteten Typ `DictionaryType` erstellen.

```
// Append a dictionary (book of derived type) and populate its attributes and elements
DictionaryType dictionary = new DictionaryType(root.Book.Append().Node);
dictionary.ID.Value = 2;
dictionary.Title.Append().Value = "English-German Dictionary";
dictionary.Author.Append().Value = "John Doe";
dictionary.FromLanguage.Append().Value = "English";
dictionary.ToLanguage.Append().Value = "German";

// Since it's a derived type, make sure to set the xsi:type attribute of the book element
dictionary.SetXsiType();
```

Beachten Sie, dass es wichtig ist, dass Sie das `xsi:type` Attribut des neu erstellten Buchs definieren. Damit stellen Sie sicher, dass der Buchtyp korrekt vom Schema interpretiert wird, wenn das XML-Dokument validiert wird.

Im folgenden Codefragment gezeigt, wie ein Buch vom abgeleiteten Typ `DictionaryType` in der geladenen XML-Instanz identifiziert wird, wenn Sie Daten aus einem XML-Dokument laden. Zuerst wird im Code der Wert des `xsi:type`-Attributs des Buchs gefunden. Wenn die Namespace URI dieses Node `http://www.nanonull.com/LibrarySample` lautet und wenn das URI-Lookup-Präfix und der Typ mit dem Wert des `xsi:type`-Attributs übereinstimmen, so handelt es sich um ein Wörterbuch:

```
      // Determine if this book is of derived type
      if (book.Node.Attributes.GetNamedItem("xsi:type") != null)
      {
          // Find the value of the xsi:type attribute
          string xsiTypeValue = book.Node.Attributes.GetNamedItem("xsi:type").Value;
          // Get the namespace URI and the lookup prefix of this namespace
          string namespaceUri = book.Node.NamespaceURI;
          string prefix = book.Node.GetPrefixOfNamespace(namespaceUri);

          // if this book has DictionaryType
          if (namespaceUri == "http://www.nanonull.com/LibrarySample" &&
xsiTypeValue.Equals(prefix + ":DictionaryType"))
          {
             // output additional fields
             DictionaryType dictionary = new DictionaryType(book.Node);
             Console.WriteLine("Language from: " + dictionary.FromLang.First.Value);
             Console.WriteLine("Language to: " + dictionary.ToLang.First.Value);
          }
          else
          {
              throw new Exception("Unexpected book type");
          }
      }
```

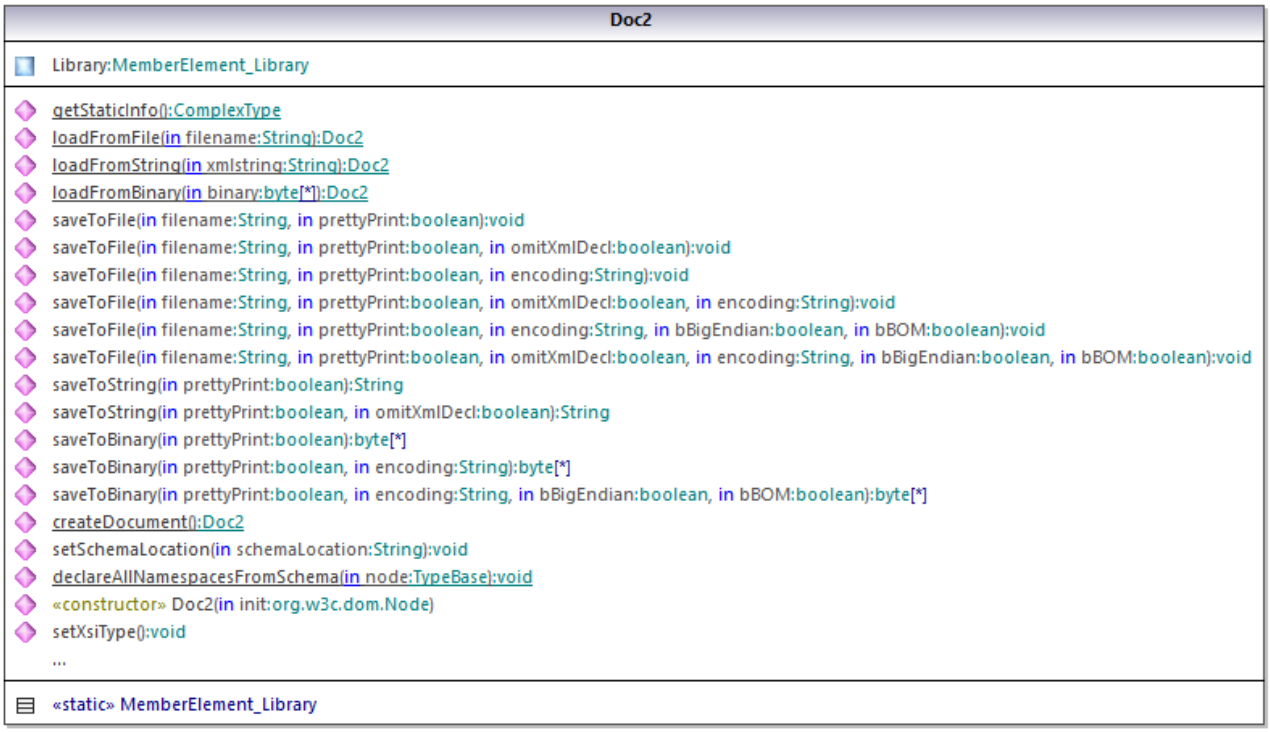
16.6.5.3 Lesen und Schreiben von XML-Dokumenten (Java)

Nachdem Sie anhand des Beispielschemas[1620]) Code generiert haben, wird ein Java-Testprojekt sowie eine Reihe von unterstützenden Altova-Bibliotheken generiert.

Informationen zu den generierten Java-Bibliotheken

Die zentrale Klasse des generierten Codes ist die Klasse `Doc2`. Sie repräsentiert das XML-Dokument. Eine solche Klasse wird für jedes Schema generiert. Ihr Name hängt vom Namen der Schemadatei ab. Beachten Sie, dass diese Klasse den Namen `Doc2` erhält, um einen möglichen Konflikt mit dem Namespace-Namen zu vermeiden. Wie im Diagramm gezeigt, bietet diese Klasse Methoden zum Laden von Dokumenten aus Dateien, Binär-Streams oder Strings (oder zum Speichern von Dokumenten in Dateien, Streams, Strings). In der Referenz zur Klasse `com.[YourSchema].[Doc]`[1695] finden Sie eine Beschreibung dieser Klasse.

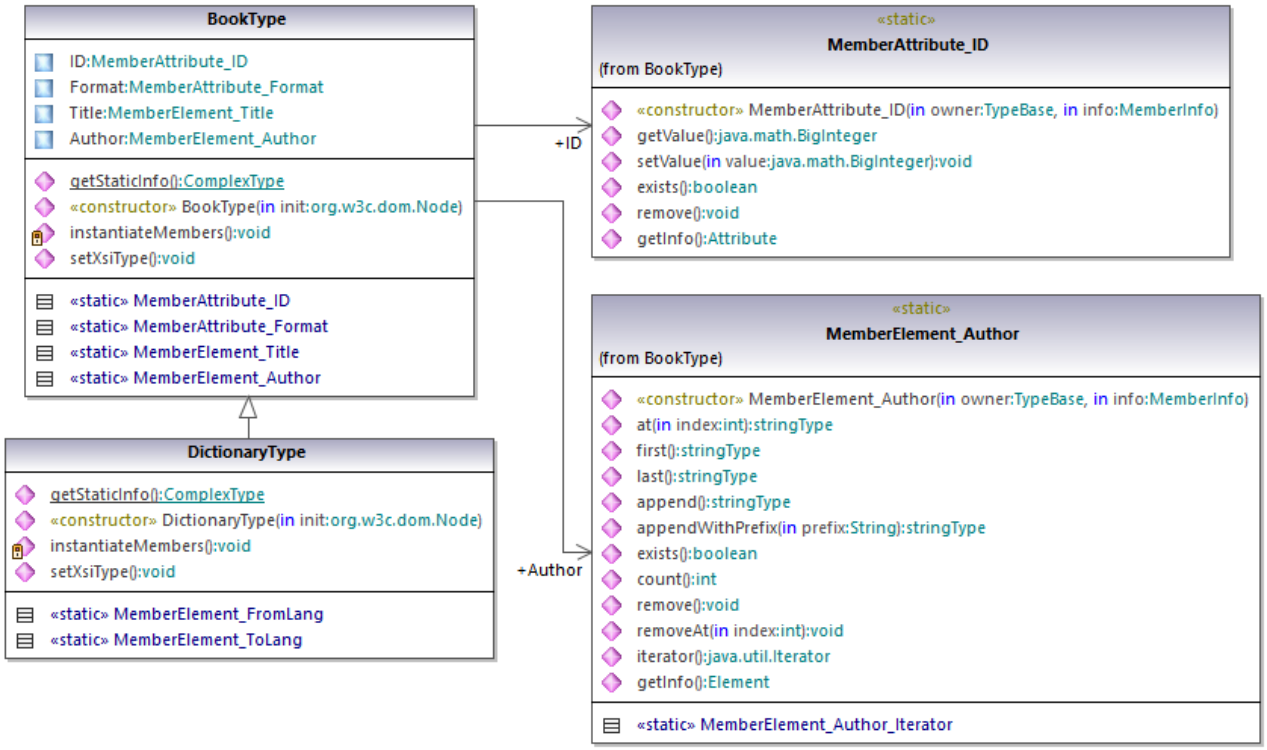
© 2016-2022 Altova GmbH



Der Member `Library` der Klasse `Doc2` stellt die eigentliche Root des Dokuments dar.

Gemäß den im Kapitel Informationen zu Schema Wrapper-Bibliotheken (Java)[1615] angeführten Codegenerierungsregeln werden für jedes Attribut und jedes Element eines Typs Member-Klassen generiert. Der Name solcher Member-Klassen erhält im generierten Code das Präfix `MemberAttribute_` bzw. `MemberElement_`. Im Diagramm unten sind Beispiele für solche Klassen `MemberAttribute_ID` und `MemberElement_Author`, die anhand des Elements **Author** bzw. des Attributs **ID** eines Buchs generiert wurden. Sie ermöglichen die programmatische Bearbeitung (z.B. Anhängen, Entfernen, Wert definieren, usw.) der entsprechenden Elemente und Attribute im XML-Instanzdokument. Nähere Informationen dazu finden Sie in der Referenz zur Klasse unter `com.[YourSchema].[YourSchemaType].MemberAttribute`[1698] und `com.[YourSchema].[YourSchemaType].MemberElement`[1693].

Der Typ **DictionaryType** ist ein complexType, der im Schema von **BookType** abgeleitet wurde, daher wird diese Beziehung auch in den generierten Klassen übernommen. Im Diagramm unten sehen Sie, dass die Klasse `DictionaryType` die Klasse `BookType` erbt.



Wenn in Ihrem XML-Schema simpleTypes als Enumerationen definiert sind, so stehen die enumerierten Werte im generierten Code als `Enum`-Werte zur Verfügung. In dem in diesem Beispiel verwendeten Schema gibt es die Buchformate hardcover, paperback, e-book, usw. Im generierten Code stehen diese Werte folglich in Form einer Enum, d.h. als Member der Klasse `CBookFormatType, zur Verfügung.`

Schreiben eines XML-Dokuments

1. Klicken Sie im Menü **File** von Eclipse auf **Import**, wählen Sie **Existing Projects into Workspace**, und klicken Sie auf **Next**.
2. Klicken Sie neben **Select root directory** auf **Browse**, wählen Sie das Verzeichnis, in dem der Java-Code generiert werden soll aus und klicken Sie auf **Finish**.
3. Erweitern Sie im Eclipse Package Explorer das Paket **com.LibraryTest** und öffnen Sie die Datei **LibraryTest.java**.

Beim Erstellen eines Prototyps einer Applikation anhand eines häufig geänderten XML-Schemas müssen Sie eventuell immer wieder Code im selben Verzeichnis generieren, damit die Änderungen am Schema sofort im Code berücksichtigt werden. Beachten Sie, dass die generierte Testapplikation und die Altova-Bibliotheken jedes Mal, wenn Sie Code im selben Zielverzeichnis generieren, überschrieben werden. Fügen Sie daher keinen Code zur generierten Testapplikation hinzu, sondern integrieren Sie stattdessen die Altova-Bibliotheken in Ihr Projekt (siehe Integrieren von Schema Wrapper-Bibliotheken 1617).

4. Bearbeiten Sie die Methode `Example()` wie unten gezeigt.

 © 2016-2022 Altova GmbH

```
protected static void example() throws Exception {
      // create a new, empty XML document
      Doc2 libDoc = Doc2.createDocument();

      // create the root element <Library> and add it to the document
      LibraryType lib = libDoc.Library3.append();

      // set the "LastUpdated" attribute
      com.altova.types.DateTime dt = new com.altova.types.DateTime(DateTime.now());
      lib.LastUpdated.setValue(dt);

      // create a new <Book> and populate its elements and attributes
      BookType book = lib.Book.append();
      book.ID.setValue(java.math.BigInteger.valueOf(1));
      book.Format.setEnumerationValue( BookFormatType.EPAPERBACK );
      book.Title.append().setValue("The XML Spy Handbook");
      book.Author.append().setValue("Altova");

      // create a dictionary (book of derived type)  and populate its elements and
attributes
      DictionaryType dict = new DictionaryType(lib.Book.append().getNode());
      dict.ID.setValue(java.math.BigInteger.valueOf(2));
      dict.Title.append().setValue("English-German Dictionary");
      dict.Format.setEnumerationValue(BookFormatType.EE_BOOK);
      dict.Author.append().setValue("John Doe");
      dict.FromLang.append().setValue("English");
      dict.ToLang.append().setValue("German");
      dict.setXsiType();

      // set the schema location (this is optional)
      libDoc.setSchemaLocation("Library.xsd");

      // save the XML document to a file with default encoding (UTF-8). "true" causes the
file to be pretty-printed.
      libDoc.saveToFile("Library1.xml", true);
   }
```

5. Bauen Sie das Java-Projekt und führen Sie es aus. Wenn der Code erfolgreich ausgeführt wurde, wird im Projektverzeichnis die Datei **Library1.xml** erstellt.

Lesen eines XML-Dokuments

1. Klicken Sie im Menü **File** von Eclipse auf **Import**, wählen Sie **Existing Projects into Workspace**, und klicken Sie auf **Next**.
2. Klicken Sie neben **Select root directory** auf **Browse**, wählen Sie das Verzeichnis, in dem der Java-Code generiert werden soll aus und klicken Sie auf **Finish**.
3. **Speichern Sie den unten gezeigten Code unter dem Namen Library1.xml** in einem lokalen Verzeichnis (Sie müssen den Pfad der Datei **Library1.xml** aus dem Beispielcode unten referenzieren).

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
```

```
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-
02-03T17:10:08.4977404">
   <Book ID="1" Format="E-book">
      <Title>The XMLSpy Handbook</Title>
      <Author>Altova</Author>
   </Book>
   <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
      <Title>English-German Dictionary</Title>
      <Author>John Doe</Author>
      <FromLang>English</FromLang>
      <ToLang>German</ToLang>
   </Book>
</Library>
```

4. Erweitern Sie im Eclipse Package Explorer das Paket **com.LibraryTest** und öffnen Sie die Datei **LibraryTest.java**.
5. Bearbeiten Sie die Methode `Example()` wie unten gezeigt.

```
protected static void example() throws Exception {
      // load XML document from a path, make sure to adjust the path as necessary
      Doc2 libDoc = Doc2.loadFromFile("Library1.xml");

      // get the first (and only) root element <Library>
      LibraryType lib = libDoc.Library3.first();

      // check whether an element exists:
     if (!lib.Book.exists()) {
        System.out.println("This library is empty.");
        return;
     }

     // read a DateTime schema type
     com.altova.types.DateTime dt = lib.LastUpdated.getValue();
     System.out.println("The library was last updated on: " + dt.toDateString());

     // iteration: for each <Book>...
     for (java.util.Iterator itBook = lib.Book.iterator(); itBook.hasNext();) {
        BookType book = (BookType) itBook.next();
        // output values of ID attribute and (first and only) title element
        System.out.println("ID: " + book.ID.getValue());
        System.out.println("Title: " + book.Title.first().getValue());

        // read and compare an enumeration value
        if (book.Format.getEnumerationValue() == BookFormatType.EPAPERBACK)
           System.out.println("This is a paperback book.");

        // for each <Author>...
        for (java.util.Iterator itAuthor = book.Author.iterator(); itAuthor
              .hasNext();)
           System.out.println("Author: " + ((com.Doc.xs.stringType)
itAuthor.next()).getValue());
```


© 2016-2022 Altova GmbH

```
        // find the derived type of this book
        // by looking at the value of the xsi:type attribute, using DOM
        org.w3c.dom.Node bookNode = book.getNode();
        if (bookNode.getAttributes().getNamedItem("xsi:type") != null) {
           // Get the value of the xsi:type attribute
           String xsiTypeValue =
bookNode.getAttributes().getNamedItem("xsi:type").getNodeValue();

           // Get the namespace URI and lookup prefix of this namespace
           String namespaceUri = bookNode.getNamespaceURI();
           String lookupPrefix = bookNode.lookupPrefix(namespaceUri);

           // If xsi:type matches the namespace URI and type of the book node
           if (namespaceUri == "http://www.nanonull.com/LibrarySample"
                 && (  xsiTypeValue.equals(lookupPrefix + ":DictionaryType" )))    {
              // ...then this is a book of derived type (dictionary)
              DictionaryType dictionary = new DictionaryType(   book.getNode());
              // output the value of the "FromLang" and "ToLang" elements
              System.out.println("From language: " +
dictionary.FromLang.first().getValue());
              System.out.println("To language: " + dictionary.ToLang.first().getValue());
           }
           else
           {
              // throw an error
              throw new java.lang.Error("This book has an unknown type.");
           }
        }
     }
   }
```

6. Bauen Sie das Java-Projekt und führen Sie es aus. Wenn der Code erfolgreich ausgeführt wurde, wird **Library1.xml** vom Programmcode gelesen und ihr Inhalt in der Konsolenansicht angezeigt.

Lesen und Schreiben von Elemente und Attributen

Die Werte von Attributen und Elementen können über die Methode `getValue()` der generierten Member-Element- bzw. Attribut-Klasse aufgerufen werden, z.B.:

```
// output values of ID attribute and (first and only) title element
System.out.println("ID: " + book.ID.getValue());
System.out.println("Title: " + book.Title.first().getValue());
```

Um in diesem konkreten Beispiel den Wert des Elements **Title** abzurufen, haben wir auch die Methode `First()` verwendet, da es sich hier um des erste (und einzige) **Title**-Element eines Buchs handelt. In Fällen, in denen ein bestimmtes Element nach dem Index aus einer Liste gewählt werden soll, verwenden Sie die Methode `At()`.

Um über mehrere Elemente zu iterieren, verwenden Sie entweder eine Index-basierte Iteration oder `java.util.Iterator`. So können Sie etwa folgendermaßen über die Bücher einer Bibliothek iterieren:

```
// index-based iteration
for (int j = 0; j < lib.Book.count(); ++j ) {
   // your code here
}

// alternative iteration using java.util.Iterator
for (java.util.Iterator itBook = lib.Book.iterator(); itBook.hasNext();) {
      // your code here
   }
```

Mit Hilfe der Methode `Append()`können Sie ein neues Element hinzufügen. Im folgenden Code wird z.B. ein leeres Root-Element **Library** an das Dokument angehängt:

```
// create the root element <Library> and add it to the document
LibraryType lib = libDoc.Library3.append();
```

Nachdem ein Element angehängt wurde, können Sie den Wert eines seiner Elemente oder Attribute mit Hilfe der setValue()-Methode definieren:

```
// set the value of the Title element
book.Title.append().setValue("The XML Spy Handbook");
// set the value of the ID attribute
book.ID.setValue(java.math.BigInteger.valueOf(1));
```

Lesen und Schreiben von Enumerationswerten

Wenn in Ihrem XML-Schema simpleTypes als Enumerationen definiert sind, so stehen die enumerierten Werte im generierten Code als `Enum`-Werte zur Verfügung. In dem in diesem Beispiel verwendeten Schema gibt es die Buchformate hardcover, paperback, e-book, usw. Im generierten Code stehen diese Werte folglich in Form einer `Enum zur Verfügung` (siehe die Klasse `BookFormatType` im Diagramm oben). Mit Hilfe von Code wie dem unten gezeigten können Sie einem Objekt Enumerationswerte zuweisen:

```
// set an enumeration value
book.Format.setEnumerationValue( BookFormatType.EPAPERBACK );
```

Solche Enumerationswerte können folgendermaßen aus XML-Instanzdokumenten ausgelesen werden:

```
// read an enumeration value
if (book.Format.getEnumerationValue() == BookFormatType.EPAPERBACK)
         System.out.println("This is a paperback book."
```

Wenn eine "if"-Bedingung nicht genügt, erstellen Sie einen Switch, um jeden Enumerationswert zu ermitteln und wie erforderlich zu verarbeiten.

Arbeiten mit den Typen xs:dateTime und xs:duration

Wenn im Schema, anhand dessen Sie Code generiert haben, Uhrzeit- und Zeitdauer-Typen wie `xs:dateTime` oder `xs:duration` vorkommen, so werden diese im generierten Code in native Altova-Klassen konvertiert.

© 2016-2022 Altova GmbH

Gehen Sie daher folgendermaßen vor, um einen Datums- oder Zeitdauerwert in das XML-Dokument zu schreiben:

1. Erstellen Sie ein `com.altova.types.DateTime`[1684]- oder `com.altova.types.Duration`[1688]-Objekt.
2. Definieren Sie das Objekt als Wert des benötigten Elements oder Attributs, z.B.:

```
// set the value of an attribute of DateTime type
com.altova.types.DateTime dt = new com.altova.types.DateTime(DateTime.now());
lib.LastUpdated.setValue(dt);
```

Um ein Datum oder eine Zeitdauer aus einem XML-Dokument zu lesen, gehen Sie folgendermaßen vor:

1. Deklarieren Sie den Elementwert (oder den Attributwert) als `com.altova.types.DateTime`[1684]- oder `com.altova.types.Duration`[1688]-Objekt.
2. Formatieren Sie das benötige Element oder Attribut, z.B.:

```
// read a DateTime type
com.altova.types.DateTime dt = lib.LastUpdated.getValue();
   System.out.println("The library was last updated on: " + dt.toDateString());
```

Nähere Informationen dazu finde Sie in der Referenz zu den Klassen `com.altova.types.DateTime`[1684] und `com.altova.types.Duration`[1688].

Arbeiten mit abgeleiteten Typen

Wenn in Ihrem XML-Schema abgeleitete Typen (derived types) definiert sind, so können Sie die Typableitung in programmatisch erstellten oder geladenen XML-Dokumenten beibehalten. Im folgenden Codefragment wird gezeigt, wie Sie anhand des in diesem Beispiel verwendeten Schemas ein neues Buch mit dem abgeleiteten Typ `DictionaryType` erstellen.

```
// create a dictionary (book of derived type)  and populate its elements and attributes
DictionaryType dict = new DictionaryType(lib.Book.append().getNode());
dict.ID.setValue(java.math.BigInteger.valueOf(2));
dict.Title.append().setValue("English-German Dictionary");
dict.Format.setEnumerationValue(BookFormatType.EE_BOOK);
dict.Author.append().setValue("John Doe");
dict.FromLang.append().setValue("English");
dict.ToLang.append().setValue("German");
dict.setXsiType();
```

Beachten Sie, dass es wichtig ist, dass Sie das `xsi:type` Attribut des neu erstellten Buchs definieren. Damit stellen Sie sicher, dass der Buchtyp korrekt vom Schema interpretiert wird, wenn das XML-Dokument validiert wird.

Im folgenden Codefragment gezeigt, wie ein Buch vom abgeleiteten Typ `DictionaryType` in der geladenen XML-Instanz identifiziert wird, wenn Sie Daten aus einem XML-Dokument laden. Zuerst wird im Code der Wert des `xsi:type`-Attributs des Buchs gefunden. Wenn die Namespace URI dieses Node `http://www.nanonull.com/LibrarySample` lautet und wenn das URI-Lookup-Präfix und der Typ mit dem Wert des `xsi:type`-Attributs übereinstimmen, so handelt es sich um ein Wörterbuch:

```
         // find the derived type of this book
         // by looking at the value of the xsi:type attribute, using DOM
         org.w3c.dom.Node bookNode = book.getNode();
         if (bookNode.getAttributes().getNamedItem("xsi:type") != null) {
            // Get the value of the xsi:type attribute
            String xsiTypeValue =
bookNode.getAttributes().getNamedItem("xsi:type").getNodeValue();

            // Get the namespace URI and lookup prefix of the book node
            String namespaceUri = bookNode.getNamespaceURI();
            String lookupPrefix = bookNode.lookupPrefix(namespaceUri);

            // If xsi:type matches the namespace URI and type of the book node
            if (namespaceUri == "http://www.nanonull.com/LibrarySample"
                  && (  xsiTypeValue.equals(lookupPrefix + ":DictionaryType" )))    {
               // ...then this is a book of derived type (dictionary)
               DictionaryType dictionary = new DictionaryType(   book.getNode());
               // output the value of the "FromLang" and "ToLang" elements
               System.out.println("From language: " +
dictionary.FromLang.first().getValue());
               System.out.println("To language: " +
dictionary.ToLang.first().getValue());
            }
            else
            {
               // throw an error
               throw new java.lang.Error("This book has an unknown type.");
            }
         }
```

16.6.6 Beispiel: Bestellung

In diesem Beispiel wird gezeigt, wie Sie mit Programmcode arbeiten, der anhand eines XML-Hauptschemas, durch das andere Schemas importiert wurden, generiert wurde. Jedes der importierten Schemas hat einen anderen Ziel-Namespace. Ziel ist es hier, ein XML-Dokument, in dem alle Elemente ein Präfix entsprechend ihrem Namespace erhalten, programmatisch zu erstellen. Das anhand Ihres C++-, C#- oder Java-Codes erstellte XML-Dokument sollte dabei wie das unten gezeigte aussehen:

```
<?xml version="1.0" encoding="utf-8"?>
<p:Purchase xsi:schemaLocation="http://NamespaceTest.com/Purchase Main.xsd"
            xmlns:p="http://NamespaceTest.com/Purchase"
            xmlns:o="http://NamespaceTest.com/OrderTypes"
            xmlns:c="http://NamespaceTest.com/CustomerTypes"
            xmlns:cmn="http://NamespaceTest.com/CommonTypes"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <p:OrderDetail>
        <o:Item>
            <o:ProductName>Lawnmower</o:ProductName>
            <o:Quantity>1</o:Quantity>
            <o:UnitPrice>148.42</o:UnitPrice>
        </o:Item>
```


© 2016-2022 Altova GmbH

```
    </p:OrderDetail>
    <p:PaymentMethod>VISA</p:PaymentMethod>
    <p:CustomerDetails>
        <c:Name>Alice Smith</c:Name>
        <c:DeliveryAddress>
            <cmn:Line1>123 Maple Street</cmn:Line1>
            <cmn:Line2>Mill Valley</cmn:Line2>
        </c:DeliveryAddress>
        <c:BillingAddress>
            <cmn:Line1>8 Oak Avenue</cmn:Line1>
            <cmn:Line2>Old Town</cmn:Line2>
        </c:BillingAddress>
    </p:CustomerDetails>
</p:Purchase>
```

Das in diesem Beispiel verwendete Hauptschema hat den Namen **Main.xsd**. Wie Sie im unten gezeigten Codefragment sehen, werden darin drei weitere Schemas importiert: **CommonTypes.xsd**, **CustomerTypes.xsd** und **OrderTypes.xsd**. Um dieselben Ergebnisse wie in diesem Beispiel zu erhalten, speichern Sie alle unten gezeigten Codefragmente in Dateien und verwenden Sie dieselben Namen wie oben. Beachten Sie, dass jedes der Präfixe **ord**, **pur**, **cmn** und **cust** im Schema einem Namespace zugeordnet ist (Order types, Purchase types, Common types bzw. Customer types). Im generierten Code stehen die entsprechenden Klassen für Orders, Purchases, Customers usw. dann unter ihrem jeweiligen Namespace zur Verfügung.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/Purchase"
      xmlns:ord="http://NamespaceTest.com/OrderTypes"
      xmlns:pur="http://NamespaceTest.com/Purchase"
      xmlns:cmn="http://NamespaceTest.com/CommonTypes"
      xmlns:cust="http://NamespaceTest.com/CustomerTypes"
      elementFormDefault="qualified">
    <xs:import schemaLocation="CommonTypes.xsd"
namespace="http://NamespaceTest.com/CommonTypes" />
    <xs:import schemaLocation="CustomerTypes.xsd"
namespace="http://NamespaceTest.com/CustomerTypes" />
    <xs:import schemaLocation="OrderTypes.xsd"
namespace="http://NamespaceTest.com/OrderTypes" />
    <xs:element name="Purchase">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="OrderDetail" type="ord:OrderType" />
                <xs:element name="PaymentMethod" type="cmn:PaymentMethodType" />
                <xs:element ref="pur:CustomerDetails" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="CustomerDetails" type="cust:CustomerType" />
</xs:schema>
```

Main.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/CommonTypes"
      elementFormDefault="qualified">
   <xs:complexType name="AddressType">
      <xs:sequence>
         <xs:element name="Line1" type="xs:string"/>
         <xs:element name="Line2" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
   <xs:simpleType name="PriceType">
      <xs:restriction base="xs:decimal">
         <xs:fractionDigits value="2"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="PaymentMethodType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="VISA"/>
         <xs:enumeration value="MasterCard"/>
         <xs:enumeration value="Cash"/>
         <xs:enumeration value="AMEX"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

CommonTypes.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/CustomerTypes"
      xmlns:cmn="http://NamespaceTest.com/CommonTypes"
      elementFormDefault="qualified">
    <xs:import schemaLocation="CommonTypes.xsd"
namespace="http://NamespaceTest.com/CommonTypes" />
    <xs:complexType name="CustomerType">
        <xs:sequence>
            <xs:element name="Name" type="xs:string" />
            <xs:element name="DeliveryAddress" type="cmn:AddressType" />
            <xs:element name="BillingAddress" type="cmn:AddressType" />
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

CustomerTypes.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/OrderTypes"
      xmlns:cmn="http://NamespaceTest.com/CommonTypes"
```


© 2016-2022 Altova GmbH

```
      elementFormDefault="qualified">
    <xs:import schemaLocation="CommonTypes.xsd"
namespace="http://NamespaceTest.com/CommonTypes" />
    <xs:complexType name="OrderType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" name="Item">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="ProductName" type="xs:string" />
                        <xs:element name="Quantity" type="xs:int" />
                        <xs:element name="UnitPrice" type="cmn:PriceType" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

OrderTypes.xsd

Um dieses Beispiel fertig zu stellen, gehen Sie folgendermaßen vor:

1. Speichern Sie alle Schemas aus den obigen Codefragmenten auf der Festplatte und stellen Sie dabei sicher, dass die angegebenen Dateinamen beibehalten werden.
2. Generieren Sie den Schema Wrapper-Code anhand des obigen Schemas **Main.xsd**. Gehen Sie dabei vor, wie unter Generieren von Code anhand von XML-Schemas oder DTDs[1607] beschrieben. Sie sollten dadurch ein kompilierbares Programm in der Sprache Ihrer Wahl (C++, C# oder Java) erhalten.
3. Fügen Sie je nach Bedarf aus einem der folgenden Beispiel-Codefragmente Code zu Ihrem C++-, C#- oder Java-Programm hinzu.

   - XML Namespaces und Präfixe (C++)[1647]
   - XML Namespaces und Präfixe (C#)[1649]
   - XML Namespaces und Präfixe (Java)[1650]

16.6.6.1 XML Namespaces und Präfixe (C++)

Nachdem Sie anhand des Schemas "Library" (siehe Beispielschema[1644]) Code generiert haben, wird eine C#-Testapplikation sowie eine Reihe von unterstützenden Altova-Bibliotheken generiert. Wie bereits erwähnt, hat das Beispielschema (**Main.xsd**) mehrere Namespace-Deklarationen. Der generierte Code enthält daher die folgenden Namespaces, die Namespace-Aliassen (Präfixen) aus dem Schema entsprechen: `Main::ord`, `Main::pur`, `Main::cmn` und `Main::cust`.

Wenn Sie XML Namespaces und -Präfixe mit Hilfe von Schema Wrapper-Bibliotheken festlegen möchten, stehen Ihnen die folgenden Methoden zur Verfügung:

- DeclareAllNamespacesFromSchema()[1662]. Mit Hilfe dieser Methode können Sie in Ihrer XML-Instanzdatei dieselben Namespaces wie im Schema deklarieren. Wenn Sie, wie in diesem Beispiel andere Namespaces benötigen, verwenden Sie `DeclareNamespace()`, Die Methode `DeclareAllNamespacesFromSchema()` wird in diesem Beispiel nicht verwendet, weil wir hier XML-Elemente erstellen möchten, deren Präfixe sich von den im Schema deklarierten geringfügig unterscheiden.

- DeclareNamespace() 1664. Mit Hilfe dieser Methode können Sie ein Namespace-Präfix-Attribut in einem Element erstellen oder das vorhandene überschreiben. Das Element muss bereits erstellt worden sein - entweder mit der Methode **`append()`** oder mit **`appendWithPrefix()`**, wie weiter unten gezeigt.
- appendWithPrefix(). 1666 Mit Hilfe dieser Methode können Sie ein bestimmtes Präfix an ein Instanzelement anhängen. Um das in diesem Beispiel gezeigte XML-Instanzdokument zu erstellen, genügte es, diese Methode nur für das Root-Element aufzurufen. Alle anderen Elemente wurden durch Aufruf von append() 1666 erstellt. Ihr Präfix wurde automatisch gemäß den obigen Regeln auf Basis des Namespace hinzugefügt.

Im Codefragment unten sehen Sie, wie Sie ein XML-Dokument mit mehreren Namespace-Deklarationen und als Präfix vorangestellten Elementnamen erstellen. Dabei wird eine Bestellungsinstanz, wie im Beispiel: Bestellung 1644 gezeigt, generiert. Beachten Sie bitte, dass einige Präfixe in der XML-Instanz zur Darstellungszwecken außer Kraft gesetzt werden (d.h. Sie sind nicht genau gleich wie die im Schema deklarierten).

```
void Example()
{
   // Create the XML document and append the root element
   Main::pur::CMain doc = Main::pur::CMain::CreateDocument();
   Main::pur::CPurchaseType purchase = doc.Purchase.appendWithPrefix(_T("p"));

   // Set schema location
   doc.SetSchemaLocation(_T("Main.xsd"));

   // Declare namespaces on root element
   purchase.DeclareNamespace(_T("o"), _T("http://NamespaceTest.com/OrderTypes"));
   purchase.DeclareNamespace(_T("c"), _T("http://NamespaceTest.com/CustomerTypes"));
   purchase.DeclareNamespace(_T("cmn"), _T("http://NamespaceTest.com/CommonTypes"));

   // Append the OrderDetail element
   Main::ord::COrderType order = purchase.OrderDetail.append();
   Main::ord::CItemType item = order.Item.append();
   item.ProductName.append() = _T("Lawnmower");
   item.Quantity.append() = 1;
   item.UnitPrice.append() = 148.42;

   // Append the PaymentMethod element
   Main::cmn::CPaymentMethodTypeType paymentMethod = purchase.PaymentMethod.append();
   paymentMethod.SetEnumerationValue(Main::cmn::CPaymentMethodTypeType::k_VISA);

   // Append the CustomerDetails element
   Main::cust::CCustomerType customer = purchase.CustomerDetails.append();
   customer.Name.append() = _T("Alice Smith");
   Main::cmn::CAddressType deliveryAddress = customer.DeliveryAddress.append();
   deliveryAddress.Line1.append() = _T("123 Maple Street");
   deliveryAddress.Line2.append() = _T("Mill Valley");
   Main::cmn::CAddressType billingAddress = customer.BillingAddress.append();
   billingAddress.Line1.append() = _T("8 Oak Avenue");
   billingAddress.Line2.append() = _T("Old Town");

   // Save to file and release object from memory
   doc.SaveToFile(_T("Main1.xml"), true);
   doc.DestroyDocument();
}
```


© 2016-2022 Altova GmbH

16.6.6.2 XML Namespaces und Präfixe (C#)

Nachdem Sie anhand des Schemas "Library" (siehe Beispielschema[1644]) Code generiert haben, wird eine C#-Testapplikation sowie eine Reihe von unterstützenden Altova-Bibliotheken generiert. Wie bereits erwähnt, hat das Beispielschema (**Main.xsd**) mehrere Namespace-Deklarationen. Der generierte Code enthält daher die folgenden Namespaces, die Namespace-Aliassen (Präfixen) aus dem Schema entsprechen: **`Main.ord`**, **`Main.pur`**, **`Main.cmn`** und **`Main.cust`**.

Wenn Sie XML Namespaces und -Präfixe mit Hilfe von Schema Wrapper-Bibliotheken festlegen möchten, stehen Ihnen die folgenden Methoden zur Verfügung:

- DeclareAllNamespacesFromSchema()[1678]. Mit Hilfe dieser Methode können Sie in Ihrer XML-Instanzdatei dieselben Namespaces wie im Schema deklarieren. Wenn Sie, wie in diesem Beispiel andere Namespaces benötigen, verwenden Sie **`DeclareNamespace()`**, Die Methode **`DeclareAllNamespacesFromSchema()`** wird in diesem Beispiel nicht verwendet, weil wir hier XML-Elemente erstellen möchten, deren Präfixe sich von den im Schema deklarierten geringfügig unterscheiden.
- DeclareNamespace()[1680]. Mit Hilfe dieser Methode können Sie ein Namespace-Präfix-Attribut in einem Element erstellen oder das vorhandene überschreiben. Das Element muss bereits erstellt worden sein - entweder mit der Methode **`Append()`** oder mit **`AppendWithPrefix()`**, wie weiter unten gezeigt.
- AppendWithPrefix().[1682] Mit Hilfe dieser Methode können Sie ein bestimmtes Präfix an ein Instanzelement anhängen. Um das in diesem Beispiel gezeigte XML-Instanzdokument zu erstellen, genügte es, diese Methode nur für das Root-Element aufzurufen. Alle anderen Elemente wurden durch Aufruf von Append()[1682] erstellt. Ihr Präfix wurde automatisch gemäß den obigen Regeln auf Basis des Namespace hinzugefügt.

Im Codefragment unten sehen Sie, wie Sie ein XML-Dokument mit mehreren Namespace-Deklarationen und als Präfix vorangestellten Elementnamen erstellen. Dabei wird eine Bestellungsinstanz, wie im Beispiel: Bestellung[1644] gezeigt, generiert. Beachten Sie bitte, dass einige Präfixe in der XML-Instanz zur Darstellungszwecken außer Kraft gesetzt werden (d.h. Sie sind nicht genau gleich wie die im Schema deklarierten).

```
protected static void Example()
{
   // Create the XML document and append the root element
   pur.Main2 doc = pur.Main2.CreateDocument();
   pur.PurchaseType purchase = doc.Purchase.AppendWithPrefix("p");

   // Set schema location
   doc.SetSchemaLocation(@"Main.xsd");

   // Declare namespaces on root element
   purchase.DeclareNamespace("o", "http://NamespaceTest.com/OrderTypes");
   purchase.DeclareNamespace("c", "http://NamespaceTest.com/CustomerTypes");
   purchase.DeclareNamespace("cmn", "http://NamespaceTest.com/CommonTypes");

   // Append the OrderDetail element
   ord.OrderType order = purchase.OrderDetail.Append();
   ord.ItemType item = order.Item.Append();
   item.ProductName.Append().Value = "Lawnmower";
   item.Quantity.Append().Value = 1;
```

```
    item.UnitPrice.Append().Value = 148.42M;

    // Append the PaymentMethod element
    cmn.PaymentMethodTypeType paymentMethod = purchase.PaymentMethod.Append();
    paymentMethod.EnumerationValue = cmn.PaymentMethodTypeType.EnumValues.eVISA;

    // Append the CustomerDetails element
    cust.CustomerType customer = purchase.CustomerDetails.Append();
    customer.Name.Append().Value = "Alice Smith";
    cmn.AddressType deliveryAddress = customer.DeliveryAddress.Append();
    deliveryAddress.Line1.Append().Value = "123 Maple Street";
    deliveryAddress.Line2.Append().Value = "Mill Valley";
    cmn.AddressType billingAddress = customer.BillingAddress.Append();
    billingAddress.Line1.Append().Value = "8 Oak Avenue";
    billingAddress.Line2.Append().Value = "Old Town";

    // Save to file
    doc.SaveToFile("PurchaseOrder.xml", true);
}
```

16.6.6.3 XML Namespaces und Präfixe (Java)

Nachdem Sie anhand des Beispielschemas[1644]) Code generiert haben, wird eine Java-Testapplikation sowie eine Reihe von unterstützenden Altova-Bibliotheken generiert. Wie bereits erwähnt, hat das Beispielschema (**Main.xsd**) mehrere Namespace-Deklarationen. Der generierte Code enthält daher die folgenden Namespaces, die Namespace-Aliassen (Präfixen) aus dem Schema entsprechen: **`com.Main.ord`**, **`com.Main.pur`**, **`com.Main.cmn`** und **`com.Main.cust`**.

Wenn Sie XML Namespaces und -Präfixe mit Hilfe von Schema Wrapper-Bibliotheken festlegen möchten, stehen Ihnen die folgenden Methoden zur Verfügung:

- declareAllNamespacesFromSchema()[1695]. Mit Hilfe dieser Methode können Sie in Ihrer XML-Instanzdatei dieselben Namespaces wie im Schema deklarieren. Wenn Sie, wie in diesem Beispiel andere Namespaces benötigen, verwenden Sie **`declareNamespace()`**, Die Methode **`declareAllNamespacesFromSchema()`** wird in diesem Beispiel nicht verwendet, weil wir hier XML-Elemente erstellen möchten, deren Präfixe sich von den im Schema deklarierten geringfügig unterscheiden.
- declareNamespace()[1696]. Mit Hilfe dieser Methode können Sie ein Namespace-Präfix-Attribut in einem Element erstellen oder das vorhandene überschreiben. Das Element muss bereits erstellt worden sein - entweder mit der Methode **`append()`** oder mit **`appendWithPrefix()`**, wie weiter unten gezeigt.
- appendWithPrefix().[1698] Mit Hilfe dieser Methode können Sie ein bestimmtes Präfix an ein Instanzelement anhängen. Um das in diesem Beispiel gezeigte XML-Instanzdokument zu erstellen, genügte es, diese Methode nur für das Root-Element aufzurufen. Alle anderen Elemente wurden durch Aufruf von append()[1698] erstellt. Ihr Präfix wurde automatisch gemäß den obigen Regeln auf Basis des Namespace hinzugefügt.

Im Codefragment unten sehen Sie, wie Sie ein XML-Dokument mit mehreren Namespace-Deklarationen und als Präfix vorangestellten Elementnamen erstellen. Dabei wird eine Bestellungsinstanz, wie im Beispiel: Bestellung[1644] gezeigt, generiert. Beachten Sie bitte, dass einige Präfixe in der XML-Instanz zur Darstellungszwecken außer Kraft gesetzt werden (d.h. Sie sind nicht genau gleich wie die im Schema deklarierten).

© 2016-2022 Altova GmbH

```
protected static void example() throws Exception {
   // Create the XML document and append the root element
   com.Main.pur.Main2 doc = com.Main.pur.Main2.createDocument();
   com.Main.pur.PurchaseType purchase = doc.Purchase.appendWithPrefix("p");

   // Set schema location
   doc.setSchemaLocation("Main.xsd");

   // Declare namespaces on root element
   purchase.declareNamespace("o", "http://NamespaceTest.com/OrderTypes");
   purchase.declareNamespace("c", "http://NamespaceTest.com/CustomerTypes");
   purchase.declareNamespace("cmn", "http://NamespaceTest.com/CommonTypes");

   // Append the OrderDetail element
   com.Main.ord.OrderType order = purchase.OrderDetail.append();
   com.Main.ord.ItemType item = order.Item.append();
   item.ProductName.append().setValue("Lawnmower");
   item.Quantity.append().setValue(1);
   java.math.BigDecimal price = new java.math.BigDecimal("148.42");
   item.UnitPrice.append().setValue(price);

   // Append the PaymentMethod element
   com.Main.cmn.PaymentMethodTypeType paymentMethod = purchase.PaymentMethod.append();
   paymentMethod.setEnumerationValue(com.Main.cmn.PaymentMethodTypeType.EVISA);

   // Append the CustomerDetails element
   com.Main.cust.CustomerType customer = purchase.CustomerDetails.append();
   customer.Name.append().setValue("Alice Smith");
   com.Main.cmn.AddressType deliveryAddress = customer.DeliveryAddress.append();
   deliveryAddress.Line1.append().setValue("123 Maple Street");
   deliveryAddress.Line2.append().setValue("Mill Valley");
   com.Main.cmn.AddressType billingAddress = customer.BillingAddress.append();
   billingAddress.Line1.append().setValue("8 Oak Avenue");
   billingAddress.Line2.append().setValue("Old Town");

   // Save to file
   doc.saveToFile("PurchaseOrder.xml", true);
}
```

16.7 Referenz zu generierten Klassen (C++)

Dieses Kapitel enthält eine Beschreibung der mit MapForce anhand einer DTD oder eines XML-Schemas (siehe Generieren von Code anhand von XML-Schemas oder DTDs 1607) generierten C++-Klassen. Sie können diese Klassen in Ihren Code integrieren, um XML-Dokumente zu lesen, zu bearbeiten und zu schreiben.

Anmerkung: Der generierte Code enthält andere unterstützende Klassen, die hier nicht aufgelistet sind und Änderungen unterworfen sind.

16.7.1 altova::DateTime

Mit Hilfe dieser Klasse können XML-Attribute oder -Elemente, die Datums- und Uhrzeittypen wie z.B. `xs:dateTime` haben, verarbeitet werden.

Konstruktoren

Name	Beschreibung
`DateTime()`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf 12:00:00 Mitternacht, 1. Jänner 0001.
`DateTime(__int64 value, short timezone)`	Initialisiert eine neue Instanz der `DateTime`-Klasse. Der Parameter `value` repräsentiert die Anzahl der Zeiteinheiten (100-Nanosekunden-Intervalle), die seit 12:00:00 Mitternacht, 1. Jänner 0001 verstrichen sind.
`DateTime(int year, unsigned char month, unsigned char day, unsigned char hour, unsigned char minute, double second)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf das als Argument bereitgestellte Jahr, den Monat, Tag, die Stunde, Minute und Sekunde.
`DateTime(int year, unsigned char month, unsigned char day, unsigned char hour, unsigned char minute, double second, short timezone)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf das als Argument bereitgestellte Jahr, den Monat, Tag, die Stunde, Minute, Sekunde und Zeitzone. Die Zeitzone wird in Minuten ausgedrückt und kann positiv oder negativ sein. So wird z.B. die Zeitzone "UTC-01:00" als "-60" ausgedrückt.

Methoden

Name	Beschreibung
`unsigned char Day() const`	Gibt den Tag des Monats des aktuellen `DateTime`-Objekts zurück. Die Rückgabewerte liegen im Bereich von 1 bis 31.
`int DayOfYear() const`	Gibt den Tag des Jahres des aktuellen `DateTime`-Objekts zurück. Die Rückgabewerte liegen im Bereich von 1 bis 366.

© 2016-2022 Altova GmbH

Name	Beschreibung
`bool HasTimezone() const`	Gibt den Booleschen Wert **true** zurück, wenn für das aktuelle `DateTime`-Objekt eine Zeitzone definiert ist; gibt andernfalls **false** zurück.
`unsigned char Hour() const`	Gibt die Stunde des aktuellen `DateTime`-Objekts zurück. Die Rückgabewerte liegen im Bereich von 0 bis 23.
`static bool IsLeapYear(int year)`	Gibt den Booleschen Wert **true** zurück, wenn das Jahr der `DateTime`-Klasse ein Schaltjahr ist; gibt andernfalls **false** zurück.
`unsigned char Minute() const`	Gibt die Minute des aktuellen `DateTime`-Objekts zurück. Die Rückgabewerte liegen im Bereich von 0 bis 59.
`unsigned char Month() const`	Gibt den Monat des aktuellen `DateTime`-Objekts zurück. Die Rückgabewerte liegen im Bereich von 1 bis 12.
`__int64 NormalizedValue() const`	Gibt den als Coordinated Universal Time (UTC) ausgedrückten Wert des aktuellen `DateTime`-Objekts zurück.
`double Second() const`	Gibt die Sekunde des aktuellen `DateTime`-Objekts zurück. Die Rückgabewerte liegen im Bereich von 0 bis 59.
`void SetTimezone(short tz)`	Setzt die Zeitzone des aktuellen `DateTime`-Objekts auf den als Argument bereigestellten Zeitzonenwert. Das Argument **tz** wird in Minuten ausgedrückt und kann positiv oder negativ sein.
`short Timezone() const`	Gibt die Zeitzone des aktuellen `DateTime`-Objekts in Minuten zurück. Stellen Sie vor Verwendung dieser Methode durch Aufruf der `HasTimezone()`-Methode sicher, dass das Objekt tatsächlich eine Zeitzone hat.
`__int64 Value() const`	Gibt den Wert des `DateTime` Objekts zurück, ausgedrückt in der Anzahl der Zeiteinheiten (100-Nanosekunden-Intervalle), die seit 12:00:00 Mitternacht, 1. Jänner 0001 verstrichen sind.
`int Weekday() const`	Gibt den Wochentag des aktuellen `DateTime`-Objekts als Ganzzahl zurück. Die Werte liegen im Bereich von 0 bis 6, wobei 0 Montag ist (ISO-8601).
`int Weeknumber() const`	Gibt den Nummer der Kalenderwoche des aktuellen `DateTime`-Objekts zurück. Die Werte entsprechen der ISO-8601-Norm.
`int WeekOfMonth() const`	Gibt den Nummer der Woche des Monats des aktuellen `DateTime`-Objekts zurück. Die Werte entsprechen der ISO-8601-Norm.
`int Year() const`	Gibt das Jahr des aktuellen `DateTime`-Objekts zurück.

Beispiel

```
void Example()
{
       // initialize a new DateTime instance to 12:00:00 midnight, January 1st, 0001
```

```
	altova::DateTime dt1 = altova::DateTime();

	// initialize a new DateTime instance using the year, month, day, hour, minute,
and second
	altova::DateTime dt2 = altova::DateTime(2015, 11, 10, 9, 8, 7);

	// initialize a new DateTime instance using the year, month, day, hour, minute,
second, and UTC +01:00 timezone
	 altova::DateTime dt = altova::DateTime(2015, 11, 22, 13, 53, 7, 60);

	// Get the value of this DateTime object
	std::cout << "The number of ticks of the DateTime object is: " << dt.Value() <<
std::endl;

	// Get the year
	cout << "The year is: " << dt.Year() << endl;
	// Get the month
	cout << "The month is: " << (int)dt.Month() << endl;
	// Get the day of the month
	cout << "The day of the month is: " << (int) dt.Day() << endl;
	// Get the day of the year
	cout << "The day of the year is: " << dt.DayOfYear() << endl;
	// Get the hour
	cout << "The hour is: " << (int) dt.Hour() << endl;
	// Get the minute
	cout << "The minute is: " << (int) dt.Minute() << endl;
	// Get the second
	cout << "The second is: " << dt.Second() << endl;
	// Get the weekday
	cout << "The weekday is: " << dt.Weekday() << endl;
	// Get the week number
	cout << "The week of year is: " << dt.Weeknumber() << endl;
	// Get the week in month
	cout << "The week of month is: " << dt.WeekOfMonth() << endl;

	// Check whether a DateTime instance has a timezone
	if (dt.HasTimezone() == TRUE)
	{
		// output the value of the Timezone
		cout <<  "The timezone is: " << dt.Timezone() << endl;
	}
	else
	{
		cout <<  "No timezone has been defined." << endl;
	}

	// Construct a DateTime object with a timezone UTC+01:00 (Vienna)
	altova::DateTime vienna_dt = DateTime(2015, 11, 23, 14, 30, 59, +60);
	// Output the result in readable format
	cout << "The Vienna time: "
		 << (int) vienna_dt.Month()
		 << "-" << (int) vienna_dt.Day()
		 << " " << (int) vienna_dt.Hour()
		 << ":" << (int) vienna_dt.Minute()
		 << ":" << (int) vienna_dt.Second()
```


© 2016-2022 Altova GmbH

```
               << endl;

       // Convert the value to UTC time
       DateTime utc_dt = DateTime(vienna_dt.NormalizedValue());
       // Output the result in readable format
       cout << "The UTC time:    "
              << (int) utc_dt.Month()
              << "-" << (int) utc_dt.Day()
              << " " << (int) utc_dt.Hour()
              << ":" << (int) utc_dt.Minute()
              << ":" << (int) utc_dt.Second()
              << endl;

       // Check if a year is a leap year
       int year = 2016;
       if( altova::DateTime::IsLeapYear(year) )
       { cout << year << " is a leap year" << endl; }
       else
       { cout << year << " is not a leap year" << endl; }
}
```

16.7.2 altova::Duration

This class enables you to process XML attributes or elements of type `xs:duration`.
Mit Hilfe dieser Klasse können XML-Attribute oder -Elemente vom Typ `xs:duration` verarbeitet werden.

Konstruktoren

Name	Beschreibung
`Duration()`	Initialisiert eine neue Instanz der `Duration`-Klasse auf einen leeren Wert.
`Duration(const DayTimeDuration& dt)`	Initialisiert eine neue Instanz der `Duration`-Klasse auf eine durch das Argument `dt` definierte Zeitdauer (siehe altova::DayTimeDuration 1657).
`Duration(const YearMonthDuration& ym)`	Initialisiert eine neue Instanz der `Duration`-Klasse auf eine durch das Argument `ym` definierte Zeitdauer (siehe altova::YearMonthDuration 1658).
`Duration(const YearMonthDuration& ym, const DayTimeDuration& dt)`	Initialisiert eine neue Instanz der `Duration`-Klasse auf eine durch die Argumente `dt` und `ym` definierte Zeitdauer (siehe altova::YearMonthDuration 1658 und altova::DayTimeDuration 1657).

Methoden

Name	Beschreibung
`int Days() const`	Gibt die Anzahl der Tage in der aktuellen `Duration`-Instanz zurück.

Name	Beschreibung
`DayTimeDuration DayTime() const`	Gibt die als `DayTimeDuration` Objekt ausgedrückte Zeitdauer in Tagen und Uhrzeitwerten zurück (siehe altova::DayTimeDuration 1657).
`int Hours() const`	Gibt die Anzahl der Stunden in der aktuellen `Duration`-Instanz zurück.
`bool IsNegative() const`	Gibt den Booleschen Wert **true** zurück, wenn die aktuelle `Duration`-Instanz negativ ist.
`bool IsPositive() const`	Gibt den Booleschen Wert **true** zurück, wenn die aktuelle `Duration`-Instanz positiv ist.
`int Minutes() const`	Gibt die Anzahl der Minuten in der aktuellen `Duration`-Instanz zurück.
`int Months() const`	Gibt die Anzahl der Monate in der aktuellen `Duration`-Instanz zurück.
`double Seconds() const`	Gibt die Anzahl der Sekunden in der aktuellen `Duration`-Instanz zurück.
`YearMonthDuration YearMonth() const`	Gibt die als `YearMonthDuration`-Objekt ausgedrückte Zeitdauer in Jahren und Monaten zurück (siehe altova::YearMonthDuration 1658).
`int Years() const`	Gibt die Anzahl der Jahre in der aktuellen `Duration`-Instanz zurück.

Beispiel

Im den folgenden Codefragment wird gezeigt, wie ein neues `Duration`-Objekten erstellt und Werte daraus ausgelesen werden.

```
void ExampleDuration()
{
	// Create an empty Duration object
	altova::Duration empty_duration = altova::Duration();

	// Create a Duration object using an existing duration value
	altova::Duration duration1 = altova::Duration(empty_duration);

	// Create a YearMonth duration of six years and five months
	altova::YearMonthDuration yrduration = altova::YearMonthDuration(6, 5);

	// Create a DayTime duration of four days, three hours, two minutes, and one
second
	altova::DayTimeDuration dtduration = altova::DayTimeDuration(4, 3, 2, 1);

	// Create a Duration object by combining the two previously created durations
	altova::Duration duration = altova::Duration(yrduration, dtduration);

	// Get the number of years in this Duration instance
	cout << "Years:  " << duration.Years() << endl;
```


© 2016-2022 Altova GmbH

```
	// Get the number of months in this Duration instance
	cout << "Months: " << duration.Months() << endl;

	// Get the number of days in this Duration instance
	cout << "Days:   " << duration.Days() << endl;

	// Get the number of hours in this Duration instance
	cout << "Hours:  " << duration.Hours() << endl;

	// Get the number of hours in this Duration instance
	cout << "Minutes: " << duration.Minutes() << endl;

	// Get the number of seconds in this Duration instance
	cout << "Seconds: " << duration.Seconds() << endl;
}
```

16.7.3 altova::DayTimeDuration

Mit Hilfe dieser Klasse können XML-Schema-Zeitdauertypen, die aus einem Tages- und Uhrzeitteil bestehen, verarbeitet werden.

Konstruktoren

Name	Beschreibung
`DayTimeDuration()`	Initialisiert eine neue Instanz der `DayTimeDuration`-Klasse auf einen leeren Wert.
`DayTimeDuration(int days, int hours, int minutes, double seconds)`	Initialisiert eine neue Instanz der `DayTimeDuration`-Klasse auf die als Argumente bereitgestellte Anzahl von Tagen, Stunden, Minuten und Sekunden.
`explicit DayTimeDuration(__int64 value)`	Initialisiert eine neue Instanz der `DayTimeDuration`-Klasse auf eine Zeitdauer, die aus der im Argument **value** bereitgestellte Anzahl von Zeiteinheiten (100-Nanosekunden-Intervalle) besteht.

Methoden

Name	Beschreibung
`int Days() const`	Gibt die Anzahl der Tage in der aktuellen `DayTimeDuration`-Instanz zurück.
`int Hours() const`	Gibt die Anzahl der Stunden in der aktuellen `DayTimeDuration`-Instanz zurück.
`bool IsNegative() const`	Gibt den Booleschen Wert **true** zurück, wenn die aktuelle `DayTimeDuration`-Instanz negativ ist.

`bool IsPositive() const`	Gibt den Booleschen Wert **true** zurück, wenn die aktuelle `DayTimeDuration`-Instanz positiv ist.
`int Minutes() const`	Gibt die Anzahl der Minuten in der aktuellen `DayTimeDuration`-Instanz zurück.
`double Seconds() const`	Gibt die Anzahl der Sekunden in der aktuellen `DayTimeDuration`-Instanz zurück.
`__int64 Value() const`	Gibt den Wert (in Zeiteinheiten) der aktuellen `DayTimeDuration`-Instanz zurück.

16.7.4 altova::YearMonthDuration

Mit Hilfe dieser Klasse können XML-Schema-Zeitdauertypen, die aus einem Jahres- und Monatsteil bestehen, verarbeitet werden.

Konstruktoren

Name	Beschreibung
`YearMonthDuration()`	Initialisiert eine neue Instanz der `YearMonthDuration`-Klasse auf einen leeren Wert.
`YearMonthDuration(int years, int months)`	Initialisiert eine neue Instanz der `YearMonthDuration`-Klasse auf die in den Argumenten **years** und **months** bereitgestellte Anzahl von Tagen und Monaten.
`explicit YearMonthDuration(int value)`	Initialisiert eine neue Instanz der `YearMonthDuration`-Klasse auf eine Zeitdauer, die aus der im Argument **value** bereitgestellte Anzahl von Zeiteinheiten (100-Nanosekunden-Intervalle) besteht.

Methoden

Name	Beschreibung
`bool IsNegative() const`	Gibt den Booleschen Wert **true** zurück, wenn die aktuelle `YearMonthDuration`-Instanz negativ ist.
`bool IsPositive() const`	Gibt den Booleschen Wert **true** zurück, wenn die aktuelle `YearMonthDuration`-Instanz positiv ist.
`int Months() const`	Gibt die Anzahl der Monate in der aktuellen `YearMonthDuration`-Instanz zurück.
`int Value() const`	Gibt den Wert (in Zeiteinheiten) der aktuellen `YearMonthDuration`-Instanz zurück.
`int Years()`	Gibt die Anzahl der Jahre in der aktuellen `YearMonthDuration`-Instanz zurück.

© 2016-2022 Altova GmbH

16.7.5 altova::meta::Attribute

Mit Hilfe dieser Klasse können Sie Schemainformationen über anhand von Attributen generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines Attributs in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über ein bestimmtes im XML-Schema definiertes Attribut programmatisch abzurufen.

Methoden

Name	Beschreibung
`SimpleType GetDataType()`	Gibt den Typ des Attributinhalts zurück.
`string_type GetLocalName()`	Gibt den lokalen Namen des Attributs zurück.
`string_type GetNamespaceURI()`	Gibt die Namespace URI des Attributs zurück.
`bool IsRequired()`	Gibt "true" zurück, wenn das Attribut obligatorisch ist.

Operatoren

Name	Beschreibung
`bool operator()`	Gibt "true" zurück, wenn dies nicht das NULL-Attribut ist.
`bool operator!()`	Gibt "true" zurück, wenn dies das NULL-Attribut ist.

16.7.6 altova::meta::ComplexType

Mit Hilfe dieser Klasse können Sie Schemainformationen über anhand von complexTypes generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines complexType in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über einen bestimmten im XML-Schema definierten complexType programmatisch abzurufen.

Methoden

Name	Beschreibung
`Attribute FindAttribute(const char_type* localName, const char_type* namespaceURI)`	Sucht das Attribut mit dem angegebenen lokalen Namen und der angegebenen Namespace URI.
`Element FindElement(const char_type* localName, const char_type* namespaceURI)`	Sucht das Element mit dem angegebenen lokalen Namen und der angegebenen Namespace URI.
`std::vector<Attribute> GetAttributes()`	Gibt eine Liste aller Attribute zurück.
`ComplexType GetBaseType()`	Gibt den Basistyp dieses Typs zurück.

Name	Beschreibung
`SimpleType GetContentType()`	Gibt den simpleType des Inhalts zurück.
`std::vector<Element> GetElements()`	Gibt eine Liste aller Elemente zurück.
`string_type GetLocalName()`	Gibt den lokalen Namen des Typs zurück.
`string_type GetNamespaceURI()`	Gibt die Namespace URI des Typs zurück.

Operatoren

Name	Beschreibung
`bool operator()`	Gibt "true" zurück, wenn dies nicht der NULL complexType ist.
`bool operator!()`	Gibt "true" zurück, wenn dies der NULL complexType ist.

16.7.7 altova::meta::Element

Mit Hilfe dieser Klasse können Sie Informationen über anhand von Schema-Elementen generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines Elements in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über ein bestimmtes im XML-Schema definiertes Element programmatisch abzurufen.

Methoden

Name	Beschreibung
`ComplexType GetDataType()`	Gibt den Typ des Elements zurück. Beachten Sie, dass dies immer ein complexType ist, selbst wenn er im Originalschema als simpleType deklariert wurde. Verwenden Sie getContentType() des zurückgegebenen Objekts zum den simple content type abzurufen.
`string_type GetLocalName()`	Gibt den lokalen Namen des Elements zurück.
`unsigned int GetMaxOccurs()`	Gibt den im Schema definierten maxOccurs-Wert zurück.
`unsigned int GetMinOccurs()`	Gibt den im Schema definierten minOccurs-Wert zurück.
`string_type GetNamespaceURI()`	Gibt die Namespace URI des Elements zurück.

Operatoren

Name	Beschreibung
`bool operator()`	Gibt "true" zurück, wenn dies nicht das NULL-Element ist.
`bool operator!()`	Gibt "true" zurück, wenn dies das NULL-Element ist.

© 2016-2022 Altova GmbH

16.7.8 altova::meta::SimpleType

Mit Hilfe dieser Klasse können Sie Schemainformationen über anhand von simpleTypes generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines simpleType in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über einen bestimmten im XML-Schema definierten simpleTypes programmatisch abzurufen.

Methoden

Name	Beschreibung
`SimpleType GetBaseType()`	Gibt den Basistyp dieses Typs zurück.
`std::vector<string_type> GetEnumerations()`	Gibt eine Liste aller Enumeration Facets zurück.
`unsigned int GetFractionDigits()`	Gibt den Wert dieses Facet zurück.
`unsigned int GetLength()`	Gibt den Wert dieses Facet zurück.
`string_type GetLocalName()`	Gibt den lokalen Namen des Tys zurück.
`string_type GetMaxExclusive()`	Gibt den Wert dieses Facet zurück.
`string_type GetMaxInclusive()`	Gibt den Wert dieses Facet zurück.
`unsigned int GetMaxLength()`	Gibt den Wert dieses Facet zurück.
`string_type GetMinExclusive()`	Gibt den Wert dieses Facet zurück.
`string_type GetMinInclusive()`	Gibt den Wert dieses Facet zurück.
`unsigned int GetMinLength()`	Gibt den Wert dieses Facet zurück.
`string_type GetNamespaceURI()`	Gibt die Namespace URI des Typs zurück.
`std::vector<string_type> GetPatterns()`	Gibt eine Liste aller Pattern Facets zurück.
`unsigned int GetTotalDigits()`	Gibt den Wert dieses Facet zurück.
`WhitespaceType GetWhitespace()`	Gibt den Wert des Whitespace Facet zurück. Er kann einer der folgenden Werte sein: • Whitespace_Unknown • Whitespace_Preserve • Whitespace_Replace • Whitespace_Collapse

Operatoren

Name	Beschreibung
`bool operator()`	Gibt "true" zurück, wenn dies nicht der NULL-simpleType ist.
`bool operator!()`	Gibt "true" zurück, wenn dies der NULL-simpleType ist.

16.7.9 [YourSchema]::[CDoc]

Wenn anhand eines XML-Schemas Code generiert wird, stellt der generierte Code eine Dokumentklasse mit demselben Namen wie dem des Schemas bereit. Diese Klasse enthält alle möglichen Root-Elemente als Members sowie die folgenden Methoden. Beachten Sie, dass "CDoc" in den unten aufgelisteten Methodennamen für den Namen der generierten Dokumentklasse selbst steht.

Methoden

Name	Beschreibung
`static CDoc CreateDocument()`	Erstellt ein neues leeres XML-Dokument. Muss mit DestroyDocument() freigegeben werden.
`static void DeclareAllNamespacesFromSchema(ElementType& node)`	Deklariert alle Namespaces aus dem XML-Schema an dem als Argument angegebenen Element (normalerweise dem XML-Root-Element). Diese Methode eignet sich dann, wenn Ihr Schema mehrere Namespace-Deklarationen, von denen jedes auf ein Präfix gemappt ist, hat, und Sie alle davon für das als Argument angegebene Element deklarieren möchten.
`void DestroyDocument()`	Löscht ein Dokument. Alle Referenzen auf das Dokument und seine Nodes werden ungültig gemacht. Dieses Methode muss aufgerufen werden, wenn Sie mit einem Dokument fertig sind.
`static CDoc LoadFromBinary(const std:vector<unsigned char>& xml)`	Lädt ein XML-Dokument aus einem Byte Array.
`static CDoc LoadFromFile(const string_type& fileName)`	Lädt ein XML-Dokument aus einer Datei.
`static CDoc LoadFromString(const string_type& xml)`	Lädt ein XML-Dokument aus einem String.
`std::vector<unsigned char> SaveToBinary(bool prettyPrint)`	Speichert ein XML-Dokument in einem Byte-Array. Wenn das Argument `prettyPrint` auf "'true" gesetzt ist, wird das XML-Dokument aus Gründen der besseren Lesbarkeit neu formatiert.
`std::vector<unsigned char> SaveToBinary(bool prettyPrint, const string_type & encoding)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einem Byte-Array.

© 2016-2022 Altova GmbH

Name	Beschreibung
`std::vector<unsigned char> SaveToBinary(bool prettyPrint, const string_type & encoding, bool bBigEndian, bool bBOM)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einem Byte-Array. Für Unicode-Kodierungen können die Bytefolge und Unicode-Bytefolge-Markierung definiert werden.
`void SaveToFile(const string_type & fileName, bool prettyPrint)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung in einer Datei.
`void SaveToFile(const string_type & fileName, bool omitXmlDecl)`	Speichert ein XML-Dokument in einer Datei. Wenn das Argument `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.
`void SaveToFile(const string_type & fileName, bool omitXmlDecl, const string_type & encoding)`	Speichert ein XML-Dokument mit der angegebenen Kodierung in einer Datei. Wenn das Argument `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.
`void SaveToFile(const string_type & fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, bool bBigEndian, bool bBOM)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einer Datei. Für Unicode-Kodierungen können die Bytefolge und Unicode-Bytefolge-Markierung definiert werden.
`void SaveToFile(const string_type & fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, bool bBigEndian, bool bBOM, const string_type & lineend)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung und dem angegebenen Zeilenende in einer Datei. Für Unicode-Kodierungen können die Bytefolge und Unicode-Bytefolge-Markierung definiert werden. Diese Methode steht nur zur Verfügung, wenn Sie den Code für die Xerces3 XML-Bibliothek generiert haben (siehe Code Generator-Optionen[1701]).
`void SaveToFile(const string_type& fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, const string_type & lineend)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung und dem angegebenen Zeilenende in einer Datei. Diese Methode steht nur zur Verfügung, wenn Sie den Code für die Xerces3 XML-Bibliothek generiert haben (siehe Code Generator-Optionen[1701]).
`void SaveToFile(const string_type & fileName, bool prettyPrint, const string_type & encoding)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einer Datei.
`void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, bool bBigEndian, bool bBOM)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einer Datei. Für Unicode-Kodierungen können die Bytefolge und Unicode-Bytefolge-Markierung definiert werden.
`void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, bool`	Speichert ein XML-Dokument mit der definierten Kodierung und dem definierten Zeilenende in einer Datei. Für Unicode-

Name	Beschreibung
`bBigEndian, bool bBOM, const string_type & lineend)`	Kodierungen können die Bytefolge und Unicode-Bytefolge-Markierung definiert werden. Diese Methode steht nur zur Verfügung, wenn Sie den Code für die Xerces3 XML-Bibliothek generiert haben (siehe Code Generator-Optionen 1701).
`void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, const string_type & lineend)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung und dem angegebenen Zeilenende in einer Datei. Diese Methode steht nur zur Verfügung, wenn Sie den Code für die Xerces3 XML-Bibliothek generiert haben (siehe Code Generator-Optionen 1701).
`string_type SaveToString(bool prettyPrint)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung in einem String.
`string_type SaveToString(bool prettyPrint, bool omitXmlDecl)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung in einem String. Wenn das Argument `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.
`void SetDTDLocation(const string_type & dtdLocation)`	Fügt eine DOCTYPE-Deklaration mit der angegebenen System-ID hinzu. Es muss bereits ein Root-Element vorhanden sein. Diese Methode wird für MSXML nicht unterstützt, weil es nicht möglich ist, eine DOCTYPE-Deklaration zu einem Dokument im Arbeitsspeicher hinzuzufügen.
`void SetSchemaLocation(const string_type & schemaLocation)`	Fügt ein xsi:schemaLocation- oder xsi:noNamespaceSchemaLocation-Attribut zum Root-Element hinzu. Es muss bereits ein Root-Element vorhanden sein.

16.7.10 [YourSchema]::[ElementType]

Diese Klasse enthält Methoden zur Bearbeitung von XML-Elementen aus Ihrem Schema. Die Methoden dieser Klasse können für Elemente, nicht aber für das XML-Dokument selbst, aufgerufen werden. Beachten Sie, dass Sie die Klasse nicht direkt instantiieren müssen, um Methoden dieser Klasse aufzurufen. Jedes mit den Methoden **`append()`** oder **`appendWithPrefix()`** erstelltes Element hat den Typ `[ElementType]`.

Methoden

Name	Beschreibung
`void DeclareNamespace(const string_type prefix, const string_type nsURI)`	Diese Methode erhält zwei Argumente, beide vom Typ "String": das gewünschte Präfix und die gewünschte Namespace URI. Das als Argument bereitgestellte Präfix wird auf den als Argument bereitgestellten Namespace URI-Wert gemappt. Wenn das als Argument angegebene Präfix leer ist, erstellt die

© 2016-2022 Altova GmbH

Name	Beschreibung

	Methode die Standard-Namespace-Deklaration im Element oder setzt diese außer Kraft.
	
	Angenommen, das XML-Dokument hat ein XML-Element namens "purchase". Bei Aufruf von
	
	`purchase.DeclareNamespace(_T("ord"), _T("http://OrderTypes"));`
	
	wird das XML-Dokument zu
	
	`<purchase xmlns:ord="http://OrderTypes" />`
	
	Ein weiteres Beispiel: Bei Aufruf von
	
	`purchase.DeclareNamespace(_T(""), _T("http://OrderTypes"));`
	
	wird das XML-Dokument zu
	
	`<purchase xmlns="http://OrderTypes" />`
	
	Anmerkung: Der deklarierte Namespace wird beim Anhängen danach folgender Child-Elemente oder Attribute nach den folgenden Regeln verwendet:
	
	1. Wenn der Child-Namespace der Standard-Namespace ist, wird ein leeres Präfix verwendet.
	2. Wenn der Child-Namespace der gleiche ist wie der Parent Namespace ist, wird das Parent-Präfix verwendet.
	3. Anderfalls wird unter Verwendung des in Abschnitt "B.2: Namespace Prefix Lookup" unter https://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20021022/namespaces-algorithms.html beschriebenen Lookup-Algorithmus nach dem nächstgelegenen Präfix vom Parent bis nach oben gesucht .
	4. Wenn kein Präfix für einen Element-Namespace gefunden wird, wird ein leeres Präfix verwendet.

16.7.11 [YourSchema]::MemberAttribute

Wenn anhand eines XML-Schemas Code generiert wird, wird für jedes Member-Attribut eines Typs eine Klasse wie diese erstellt.

Methoden

Name	Beschreibung
`bool exists()`	Gibt den Wert "true" zurück, wenn das Attribut vorhanden ist
`int GetEnumerationValue()`	Wird nur für Enumeration Types generiert. Gibt eine der für die möglichen Werte generierten Konstanten oder "Invalid" zurück, wenn der Wert mit keinem der Enumerationswerte im Schema übereinstimmt.
`altova::meta::Attribute info()`	Gibt ein Objekt zum Abfragen von Schemainformationen zurück (siehe `altova::meta::Attribute`[1659]).
`void remove()`	Entfernt das Attribut aus dem übergeordneten Element.
`void SetEnumerationValue(int)`	Wird nur für Enumeration Types generiert. Übergibt eine der für die möglichen Werte generierten Konstanten an diese Methode um den Wert zu definieren.

16.7.12 [YourSchema]::MemberElement

Wenn anhand eines XML-Schemas Code generiert wird, wird für jedes Member-Element eines Typs eine Klasse wie diese erstellt. In den Beschreibungen unten steht "MemberType" für den Namen des Member-Elements selbst.

Methoden

Name	Beschreibung
`Iterator<MemberType> all()`	Gibt ein Objekt für iterierende Instanzen des Member Elements zurück.
`MemberType append()`	Erstellt ein neues Element und hängt es an das übergeordnete Element an.
`MemberType appendWithPrefix(string_type prefix)`	Erstellt ein neues Element mit dem als Argument angegebenen Präfix und hängt es an seinen Parent an. Ein Beispiel dazu finden Sie unter Beispiel: Bestellung[1644].
`unsigned int count()`	Gibt die Anzahl der Elemente zurück.
`int GetEnumerationValue()`	Wird nur für Enumeration Types generiert. Gibt eine der für die möglichen Werte generierten Konstanten zurück oder **Invalid,** wenn der Wert mit keinem der Enumerationswerte im Schema übereinstimmt.
`bool exists()`	Gibt den Wert "true" zurück, wenn mindestens ein Element vorhanden ist.
`MemberType first()`	Gibt die erste Instanz des Member Elements zurück.

© 2016-2022 Altova GmbH

Name	Beschreibung
`MemberType operator[](unsigned int index)`	Gibt das vom Index definierte Member Element zurück.
`altova::meta::Element info()`	Gibt ein Objekt zum Abfragen von Schemainformationen zurück (siehe `altova::meta::Element`[1660]).
`MemberType last()`	Gibt die letzte Instanz des Member Elements zurück.
`void remove()`	Löscht alle Instanzen des Elements aus dem übergeordneten Element.
`void removeAt(unsigned int index)`	Löscht die Instanz des durch den Index definierten Elements.
`void SetEnumerationValue(int)`	Wird nur für Enumeration Types generiert. Übergeben Sie eine der für die möglichen Werte generierten Konstanten an diese Methode, um den Wert zu definieren.

16.8 Referenz zu generierten Klassen (C#)

Dieses Kapitel enthält eine Beschreibung der mit MapForce anhand einer DTD oder eines XML-Schemas (siehe Generieren von Code anhand von XML-Schemas oder DTDs 1607) generierten C#-Klassen. Sie können diese Klassen in Ihren Code integrieren, um XML-Dokumente zu lesen, zu bearbeiten und zu schreiben.

Anmerkung: Der generierte Code enthält andere unterstützende Klassen, die hier nicht aufgelistet sind und Änderungen unterworfen sind.

16.8.1 Altova.Types.DateTime

Mit Hilfe dieser Klasse können XML-Attribute oder -Elemente, die Datums- und Uhrzeittypen wie z.B. `xs:dateTime` haben, verarbeitet werden.

Konstruktoren

	Name	Beschreibung
	`DateTime(DateTime obj)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf das als Argument bereitgestellte `DateTime`-Objekt.
	`DateTime(System.DateTime newvalue)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf das als Argument bereitgestellte `System.DateTime`.
	`DateTime(int year, int month, int day, int hour, int minute, double second, int offsetTZ)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf die als Argumente bereitgestellten Werte für Jahr, Monat, Tag, Stunde, Minute, Sekunde und Zeitzone.
	`DateTime(int year, int month, int day, int hour, int minute, double second)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf die als Argumente bereitgestellten Werte für Jahr, Monat, Tag, Stunde, Minute und Sekunde.
	`DateTime(int year, int month, int day)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf die als Argumente bereitgestellten Werte für Jahr, Monat und Tag.

Eigenschaften

	Name	Beschreibung
	`bool HasTimezone`	Ruft einen Booleschen Wert ab, der angibt, ob `DateTime` eine Zeitzone hat.
	`static DateTime Now`	Ruft ein `DateTime` Objekt ab, das auf das aktuelle Datum und die aktuelle Uhrzeit dieses Computers gesetzt ist.
	`short TimezoneOffset`	Ruft den Zeitzonenunterschied des `DateTime`-Objekts in Minuten ab bzw. definiert ihn.

© 2016-2022 Altova GmbH

	Name	Beschreibung
	`System.DateTime Value`	Ruft den Wert des `DateTime` -Objekts als `System.DateTime`-Wert ab bzw. definiert ihn als solchen.

Methoden

	Name	Beschreibung
	`int CompareTo(object obj)`	Die `DateTime`-Klasse implementiert die `IComparable`-Schnittstelle. Diese Methode vergleicht die aktuelle Instanz von `DateTime` mit einem anderen Objekt und gibt eine Ganzzahl zurück, die angibt, ob die aktuelle Instanz in der Sortierreihenfolge vor, nach oder an derselben Position wie das andere Objekt vorkommt. Siehe auch https://msdn.microsoft.com/en-us/library/system.icomparable.compareto(v=vs.110).aspx
	`override bool Equals(object obj)`	Gibt **true** zurück, wenn das angegebene Objekt gleich dem aktuellen Objekt ist; gibt andernfalls **false** zurück.
	`System.DateTime GetDateTime(bool correctTZ)`	Gibt ein `System.DateTime`-Objekt anhand der aktuellen `Altova.Types.DateTime`-Instanz zurück. Das Boolesche Argument `correctTZ` gibt an, ob die Uhrzeit des zurückgegebenen Objekts entsprechend der Zeitzone der aktuellen `Altova.Types.DateTime`-Instanz angepasst werden muss.
	`override int GetHashCode()`	Gibt den Hash-Code der aktuellen Instanz zurück.
	`int GetWeekOfMonth()`	Gibt die Nummer der Woche im Monat als Ganzzahl zurück.

	`static DateTime Parse( string s )`	Erstellt anhand des als Argument bereitgestellten String ein `DateTime`-Objekt. So würden die folgenden Beispiel-Stringwerte etwa erfolgreich in ein `DateTime`-Objekt konvertiert:
		
		`2015-01-01T23:23:23`
		`2015-01-01`
		`2015-11`
		`23:23:23`
		
		Wenn der String nicht in ein `DateTime`-Objekt konvertiert werden kann, wird eine Ausnahme ausgegeben.
		
		Beachten Sie, dass es sich hierbei um eine statische Methode handelt, die nur in der `Altova.Types.DateTime`-Klasse selbst und nicht in einer Instanz der Klasse aufgerufen werden kann.

	`static DateTime Parse(string s, DateTimeFormat format)`	Erstellt anhand des als Argument bereitgestellten Formats ein `DateTime`-Objekt. Ein Liste der möglichen

	Name	Beschreibung
		Formate finden Sie unter Altova.Types.DateTimeFormat 1671. Wenn der String nicht in ein `DateTime`-Objekt konvertiert werden kann, wird eine Ausnahme ausgegeben. Beachten Sie, dass es sich hierbei um eine statische Methode handelt, die nur in der `Altova.Types.DateTime`-Klasse selbst und nicht in einer Instanz der Klasse aufgerufen werden kann.
	`override string ToString()`	Konvertiert das `DateTime`-Objekt in einen String.
	`string ToString(DateTimeFormat format)`	Konvertiert das `DateTime`-Objekt anhand des als Argument bereitgestellten Formats in einen String. Ein Liste der möglichen Formate finden Sie unter Altova.Types.DateTimeFormat 1671.

Operatoren

Name	Beschreibung
`!=`	Ermittelt, ob `DateTime` a ungleich `DateTime` b ist.
`<`	Ermittelt, ob `DateTime` a kleiner als `DateTime` b ist.
`<=`	Ermittelt, ob `DateTime` a kleiner oder gleich `DateTime` b ist.
`==`	Ermittelt, ob `DateTime` a gleich `DateTime` b ist.
`>`	Ermittelt, ob `DateTime` a größer als `DateTime` b ist.
`>=`	Ermittelt, ob `DateTime` a größer oder gleich `DateTime` b ist.

Beispiele

Stellen Sie sicher, dass die Altova-Typen importiert wurden, bevor Sie die folgenden Codefragmente in Ihrem Programm verwenden.

```
using Altova.Types;
```

In den folgenden Codefragmenten werden verschiedene Möglichkeiten zur Erstellung von `DateTime`-Objekten gezeigt:

```
protected static void DateTimeExample1()
{
	// Create a DateTime object from the current system time
	Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);
	Console.WriteLine("The current time is: " + dt.ToString());
```


© 2016-2022 Altova GmbH

```
      // Create an Altova DateTime object from parts (no timezone)
      Altova.Types.DateTime dt1 = new Altova.Types.DateTime(2015, 10, 12, 10, 50, 33);
      Console.WriteLine("My custom time is : " + dt1.ToString());

      // Create an Altova DateTime object from parts (with UTC+60 minutes timezone)
      Altova.Types.DateTime dt2 = new Altova.Types.DateTime(2015, 10, 12, 10, 50, 33,
60);
      Console.WriteLine("My custom time with timezone is : " + dt2.ToString());

      // Create an Altova DateTime object by parsing a string
      Altova.Types.DateTime dt3 = Altova.Types.DateTime.Parse("2015-01-01T23:23:23");
      Console.WriteLine("Time created from string: " + dt3.ToString());

      // Create an Altova DateTime object by parsing a string formatted as schema date
      Altova.Types.DateTime dt4 = Altova.Types.DateTime.Parse("2015-01-01",
DateTimeFormat.W3_date);
      Console.WriteLine("Time created from string formatted as schema date: " +
dt4.ToString());
}
```

In den folgenden Codefragmenten werden verschiedene Möglichkeiten zur Formatierung von `DateTime`-Objekten gezeigt:

```
protected static void DateTimeExample2()
{
      // Create a DateTime object from the current system time
      Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);

      // Output the unformatted DateTime
      Console.WriteLine("Unformatted time: " + dt.ToString());

      // Output this DateTime formatted using various formats
      Console.WriteLine("S_DateTime:       " + dt.ToString(DateTimeFormat.S_DateTime));
      Console.WriteLine("S_Days:           " + dt.ToString(DateTimeFormat.S_Days));
      Console.WriteLine("S_Seconds:        " + dt.ToString(DateTimeFormat.S_Seconds));
      Console.WriteLine("W3_date:          " + dt.ToString(DateTimeFormat.W3_date));
      Console.WriteLine("W3_dateTime:      " + dt.ToString(DateTimeFormat.W3_dateTime));
      Console.WriteLine("W3_gDay:          " + dt.ToString(DateTimeFormat.W3_gDay));
      Console.WriteLine("W3_gMonth:        " + dt.ToString(DateTimeFormat.W3_gMonth));
      Console.WriteLine("W3_gMonthDay:     " +
dt.ToString(DateTimeFormat.W3_gMonthDay));
      Console.WriteLine("W3_gYear:         " + dt.ToString(DateTimeFormat.W3_gYear));
      Console.WriteLine("W3_gYearMonth:    " +
dt.ToString(DateTimeFormat.W3_gYearMonth));
      Console.WriteLine("W3_time:          " + dt.ToString(DateTimeFormat.W3_time));
}
```

16.8.2 Altova.Types.DateTimeFormat

Der `DateTimeFormat` enum-Typ hat die folgenden Konstantenwerte:

Wert	Beschreibung	Beispiel
S_DateTime	Formatiert den Wert mit einer Genauigkeit einer zehnmillionstel Sekunde einschließlich Zeitzone als Standard-DateTime-Wert.	`2015-11-12 12:19:03.9019132+01:00`
S_Days	Formatiert den Wert als Anzahl der seit der UNIX-Epoche verstrichenen Tage.	`735913.631897345108796296 3`
S_Seconds	Formatiert den Wert mit einer Genauigkeit einer Zehnmillionstelsekunde als Anzahl der seit der UNIX-Epoche verstrichenen Sekunden.	`63582937678.0769062`
W3_date	Formatiert den Wert als Schema-Datum	`2015-11-12`
W3_dateTime	Formatiert den Wert als Schema-DateTime (Datum und Uhrzeit).	`2015-11-12T15:12:14.5194251`
W3_gDay	Formatiert den Wert als Schema-gDay.	`---12` (vorausgesetzt, das Datum ist der 12. des Monats)
W3_gMonth	Formatiert den Wert als Schema-gMonth.	`--11` (vorausgesetzt, der Monat ist November)
W3_gMonthDay	Formatiert den Wert als Schema-gMonthDay.	`--11-12` (vorausgesetzt, das Datum ist der 12. November)
W3_gYear	Formatiert den Wert als Schema-gYear.	`2015` (vorausgesetzt das Jahr ist 2015)
W3_gYearMonth	Formatiert den Wert als Schema-gYearMonth.	`2015-11` (vorausgesetzt das Jahr ist 2015 und der Monat ist November)
W3_time	Formatiert den Wert mit der Genauigkeit einer Zehnmillionstelsekunde als Schema-Uhrzeit.	`15:19:07.5582719`

© 2016-2022 Altova GmbH

16.8.3 Altova.Types.Duration

Die Klasse `Altova.Types.Duration` stellt die folgenden öffentlichen Mitglieder bereit.

Konstruktoren

	Name	Beschreibung
	`Duration(Duration obj)`	Initialisiert eine neue Instanz der `Duration`-Klasse auf das als Argument bereitgestellte `Duration`-Objekt.
	`Duration(System.TimeSpan newvalue)`	Initialisiert eine neue Instanz der `Duration`-Klasse auf das als Argument bereitgestellte `System.TimeSpan`-Objekt.
	`Duration(long ticks)`	Initialisiert eine neue Instanz der `Duration`-Klasse auf die Anzahl der als Argument bereitgestellten Zeiteinheiten.
	`Duration(int newyears, int newmonths, int days, int hours, int minutes, int seconds, double partseconds, bool bnegative)`	Initialisiert eine neue Instanz der `Duration`-Klasse auf eine Zeitdauer, die anhand von als Argumente bereitgestellten Bestandteilen erstellt wird.

Eigenschaften

	Name	Beschreibung
	`int Months`	Ruft die Anzahl der Monate der aktuellen Instanz von `Duration` ab oder definiert diese.
	`System.TimeSpan Value`	Ruft den Wert (als `System.TimeSpan`) der aktuellen Instanz von `Duration` ab oder definiert diesen.
	`int Years`	Ruft die Anzahl der Jahre der aktuellen Instanz von `Duration` ab oder definiert diese.

Methoden

	Name	Beschreibung
	`override bool Equals(object other)`	Gibt **true** zurück, wenn das angegebene Objekt gleich dem aktuellen Objekt ist; gibt andernfalls **false** zurück.
	`override int GetHashCode()`	Gibt den Hash-Code der aktuellen Instanz zurück.
	`bool IsNegative()`	Gibt **true** zurück, wenn die aktuelle Instanz von `Duration` eine negative Zeitdauer darstellt.

	Name	Beschreibung
	`static Duration Parse( string s, ParseType pt )`	Gibt anhand des als Argument bereitgestellten String ein `Altova.Types.Duration`-Objekt zurück, wobei der als Argument bereitgestellte Parse-Typ verwendet wird. Gültige Parse-Typwerte sind: **DURATION** Parse-Zeitdauer, wenn als Bestandteile der Zeitdauer Jahr, Monate, Tag sowie Uhrzeit vorhanden sind. **YEARMONTH** Parse-Zeitdauer, wenn nur die Teile Jahr und Monat vorhanden sind. **DAYTIME** Parse-Zeitdauer, wenn nur die Teile Tag und Uhrzeit vorhanden sind. Beachten Sie, dass es sich hierbei um eine statische Methode handelt, die nur in der Klasse selbst und nicht in einer Instanz der Klasse aufgerufen werden kann.
	`override string ToString()`	Konvertiert die aktuelle `Duration`-Instanz in einen String. So würde eine Zeitspanne von 3 Stunden, 4 Minuten und 5 Sekunden z.B. in "PT3H4M5S" konvertiert
	`string ToYearMonthString()`	Konvertiert die aktuelle `Duration`-Instanz anhand des Parse-Typs "Jahr und Monat" in einen String.

Operatoren

Name	Beschreibung
`!=`	Ermittelt, ob `Duration` a ungleich `Duration` b ist.
`==`	Ermittelt, ob `Duration` a gleich `Duration` b ist.

Beispiele

Stellen Sie sicher, dass die Altova-Typen importiert wurden, bevor Sie die folgenden Codefragmente in Ihrem Programm verwenden:

```
using Altova.Types;
```

In den folgenden Codefragmenten werden verschiedene Möglichkeiten zur Erstellung von `Duration`-Objekten gezeigt:

```
protected static void DurationExample1()
{
       // Create a new time span of 3 hours, 4 minutes, and 5 seconds
       System.TimeSpan ts = new TimeSpan(3, 4, 5);
       // Create a Duration from the time span
       Duration dr = new Duration(ts);
       // The output is: PT3H4M5S
```


© 2016-2022 Altova GmbH

```
       Console.WriteLine("Duration created from TimeSpan: " + dr.ToString());

       // Create a negative Altova.Types.Duration from 6 years, 5 months, 4 days, 3
hours,
       // 2 minutes, 1 second, and .33 of a second
       Duration dr1 = new Duration(6, 5, 4, 3, 2, 1, .33, true);
       // The output is: -P6Y5M4DT3H2M1.33S
       Console.WriteLine("Duration created from parts: " + dr1.ToString());

       // Create a Duration from a string using the DAYTIME parse type
       Duration dr2 = Altova.Types.Duration.Parse("-P4DT3H2M1S",
Duration.ParseType.DAYTIME);
       // The output is -P4DT3H2M1S
       Console.WriteLine("Duration created from string: " + dr2.ToString());

       // Create a duration from ticks
       Duration dr3 = new Duration(System.DateTime.UtcNow.Ticks);
       // Output the result
       Console.WriteLine("Duration created from ticks: " + dr3.ToString());
}
```

In den folgenden Codefragmenten wird gezeigt, wie Werte aus `Duration`-Objekten ermittelt werden können:

```
protected static void DurationExample2()
{
       // Create a negative Altova.Types.Duration from 6 years, 5 months, 4 days, 3
hours,
       // 2 minutes, 1 second, and .33 of a second
       Duration dr = new Duration(6, 5, 4, 3, 2, 1, .33, true);
       // The output is: -P6Y5M4DT3H2M1.33S
       Console.WriteLine("The complete duration is: " + dr.ToString());

       // Get only the year and month part as string
       string dr1 = dr.ToYearMonthString();
       Console.WriteLine("The YEARMONTH part is: " + dr1);

       // Get the number of years in duration
       Console.WriteLine("Years: " + dr.Years);

       // Get the number of months in duration
       Console.WriteLine("Months: " + dr.Months);
}
```

16.8.4 Altova.Xml.Meta.Attribute

Mit Hilfe dieser Klasse können Sie Schemainformationen über anhand von Attributen generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines Attributs in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über ein bestimmtes im XML-Schema definiertes Attribut programmatisch abzurufen.

Eigenschaften

	Name	Beschreibung
	`SimpleType DataType`	Gibt den Typ des Attributinhalts zurück.
	`string LocalName`	Gibt den lokalen Namen des Attributs zurück.
	`string NamespaceURI`	Gibt die Namespace URI des Attributs zurück.
	`XmlQualifiedName QualifiedName`	Gibt den qualifizierten Namen des Attributs zurück.
	`bool Required()`	Gibt "true" zurück, wenn das Attribut obligatorisch ist.

16.8.5 Altova.Xml.Meta.ComplexType

Mit Hilfe dieser Klasse können Sie Schemainformationen über anhand von complexTypes generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines complexType in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über einen bestimmten im XML-Schema definierten complexType programmatisch abzurufen.

Eigenschaften

	Name	Beschreibung
	`Attribute[] Attributes`	Gibt eine Liste aller Attribute zurück.
	`ComplexType BaseType`	Gibt den Basistyp dieses Typs zurück oder Null, wenn kein Basistyp vorhanden ist.
	`SimpleType ContentType`	Gibt dem simpleType des Inhalts zurück.
	`Element[] Elements`	Gibt eine Liste aller Elemente zurück.
	`string LocalName`	Gibt den lokalen Namen des Typs zurück.
	`string NamespaceURI`	Gibt die Namespace URI des Typs zurück.
	`XmlQualifiedName QualifiedName`	Gibt den qualifizierten Namen dieses Typs zurück.

Methoden

	Name	Beschreibung
	`ComplexType BaseType`	Gibt den Basistyp dieses Typs zurück.

© 2016-2022 Altova GmbH

	Name	Beschreibung
	`bool Equals(obj)`	Überprüft auf Basis eines Vergleichs des qualifizierten Namens, ob zwei Info-Objekte denselben Typ referenzieren. Gibt "true" zurück, wenn der Typ denselben qualifizierten Namen hat.
	`Attribute FindAttribute(string localName, string namespaceURI)`	Sucht das Attribut mit dem angegebenen lokalen Namen und der angegebenen Namespace URI.
	`Element FindElement(string localName, string namespaceURI)`	Sucht das Element mit dem angegebenen lokalen Namen und der angegebenen Namespace URI.

16.8.6 Altova.Xml.Meta.Element

Mit Hilfe dieser Klasse können Sie Informationen über anhand von Schema-Elementen generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines Elements in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über ein bestimmtes im XML-Schema definiertes Element programmatisch abzurufen.

Eigenschaften

	Name	Description
	`ComplexType DataType`	Gibt den Typ des Elements zurück. Beachten Sie, dass dies immer ein complexType ist, selbst wenn er im Originalschema als simpleType deklariert wurde. Verwenden Sie die Eigenschaft `ContentType` des zurückgegebenen Objekts zum den simple content type abzurufen.
	`string` `LocalName`	Gibt den lokalen Namen des Elements zurück.
	`int` `MaxOccurs`	Gibt den im Schema definierten `maxOccurs`-Wert zurück.
	`int` `MinOccurs`	Gibt den im Schema definierten `minOccurs`-Wert zurück.
	`string` `NamespaceURI`	Gibt die Namespace URI des Elements zurück.
	`XmlQualifiedName QualifiedName`	Gibt den qualifizierten Namen des Elements zurück.

16.8.7 Altova.Xml.Meta.SimpleType

Mit Hilfe dieser Klasse können Sie Schemainformationen über anhand von simpleTypes generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines simpleType in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über einen bestimmten im XML-Schema definierten simpleTypes programmatisch abzurufen.

Eigenschaften

	Name	Beschreibung
	`SimpleType BaseType`	Gibt den Basistyp dieses Typs zurück.
	`string[] Enumerations`	Gibt eine Liste aller Enumeration Facets zurück.
	`int FractionDigits`	Gibt den Wert dieses Facet zurück.
	`int Length`	Gibt den Wert dieses Facet zurück.
	`string LocalName`	Gibt den lokalen Namen des Tys zurück.
	`string MaxExclusive`	Gibt den Wert dieses Facet zurück.
	`string MaxInclusive`	Gibt den Wert dieses Facet zurück.
	`int MaxLength`	Gibt den Wert dieses Facet zurück.
	`string MinExclusive`	Gibt den Wert dieses Facet zurück.
	`string MinInclusive`	Gibt den Wert dieses Facet zurück.
	`int MinLength`	Gibt den Wert dieses Facet zurück.
	`string NamespaceURI`	Gibt die Namespace URI des Typs zurück.
	`string[] Patterns`	Gibt die Pattern Facets oder Null zurück, wenn keine Patterns definiert sind.
	`XmlQualifiedName QualifiedName`	Gibt den qualifizierten Namens dieses Typs zurück.
	`int TotalDigits`	Gibt den Wert dieses Facet zurück.
	`WhitespaceType Whitespace`	Gibt das Whitespace Normalization Facet zurück.

16.8.8 [YourSchema].[Doc]

Wenn anhand eines XML-Schemas Code generiert wird, stellt der generierte Code eine Dokumentklasse mit demselben Namen wie dem des Schemas bereit. Diese Klasse enthält alle möglichen Root-Elemente als Members sowie die folgenden Members. Beachten Sie, dass "Doc" in den unten aufgelisteten Methodennamen für den Namen der generierten Dokumentklasse selbst steht.

Methoden

	Name	Beschreibung
	`static Doc CreateDocument()`	Erstellt ein neues leeres XML-Dokument.

© 2016-2022 Altova GmbH

	Name	Beschreibung
	`static Doc CreateDocument(string encoding)`	Erstellt ein neues leeres XML-Dokument mit Kodierung des Typs "encoding".
	`static void DeclareAllNamespacesFromSchema(Altova.Xml.ElementType node)`	Deklariert alle Namespaces aus dem XML-Schema an dem als Argument angegebenen Element (normalerweise dem XML-Root-Element). Diese Methode eignet sich dann, wenn Ihr Schema mehrere Namespace-Deklarationen, von denen jedes auf ein Präfix gemappt ist, hat, und Sie alle davon für das als Argument angegebene Element deklarieren möchten.
	`static Doc LoadFromBinary(byte[] binary)`	Lädt ein XML-Dokument aus einem Byte Array.
	`static Doc LoadFromFile(string filename)`	Lädt ein XML-Dokument aus einer Datei.
	`static Doc LoadFromString(string xmlstring)`	Lädt ein XML-Dokument aus einem String.
	`byte[] SaveToBinary(bool prettyPrint)`	Speichert ein XML-Dokument mit optionaler Pretty-Print-Formatierung in einem Byte Array.
	`byte[] SaveToBinary(bool prettyPrint, string encoding)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einem Byte-Array.
	`byte[] SaveToBinary(bool prettyPrint, string encoding, bool bBigEndian, bool bBOM)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung, Bytefolge und BOM (Byte Order Mark) in einem Byte-Array.
	`void SaveToFile(string fileName, bool prettyPrint)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung in einer Datei.
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung in einer Datei. Wenn `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einer Datei. Wenn `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.
	`void SaveToFile(string fileName, bool prettyPrint, string encoding, string lineend)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung und dem angegebenen Zeilenende in einer Datei.
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding, string lineend)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung und dem angegebenen Zeilenende in einer Datei. Wenn `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.

	Name	Beschreibung
	`void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding, bool bBigEndian, bool bBOM, string lineend)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung, Bytefolge, BOM (Byte Order Mark) und dem definierten Zeilenende in einer Datei. Wenn `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.
	`void SaveToFileWithLineEnd(string fileName, bool prettyPrint, bool omitXmlDecl, string lineend)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung und dem definierten Zeilenende in einer Datei. Wenn `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.
	`string SaveToString(bool prettyPrint)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung in einer Datei.
	`string SaveToString(bool prettyPrint, bool omitXmlDecl)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung in einer Datei. Wenn `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.
	`void SetDTDLocation(string dtdLocation)`	Fügt eine DOCTYPE-Deklaration mit der angegebenen System-ID hinzu. Es muss bereits ein Root-Element vorhanden sein.
	`void SetSchemaLocation(string schemaLocation)`	Fügt ein xsi:schemaLocation- oder xsi:noNamespaceSchemaLocation-Attribut zum Root-Element hinzu. Es muss bereits ein Root-Element vorhanden sein.

16.8.9 [YourSchema].[ElementType]

Diese Klasse enthält Methoden zur Bearbeitung von XML-Elementen aus Ihrem Schema. Die Methoden dieser Klasse können für Elemente, nicht aber für das XML-Dokument selbst, aufgerufen werden. Beachten Sie, dass Sie die Klasse nicht direkt instantiieren müssen, um Methoden dieser Klasse aufzurufen. Jedes mit den Methoden **`Append()`** oder **`AppendWithPrefix()`** erstelltes Element hat den Typ `[ElementType]`.

Methoden

	Name	Beschreibung
	`void DeclareNamespace(string prefix, string nsURI)`	Diese Methode erhält zwei Argumente, beide vom Typ "String": das gewünschte Präfix und die gewünschte Namespace URI. Das als Argument bereitgestellte Präfix wird auf den als Argument bereitgestellten Namespace URI-Wert gemappt. Wenn das als Argument angegebene Präfix leer ist, erstellt die Methode die Standard-Namespace-Deklaration im Element oder setzt diese außer Kraft. Angenommen, das XML-Dokument hat ein XML-Element namens "purchase". Bei Aufruf von

© 2016-2022 Altova GmbH

	Name	Beschreibung

		`purchase.DeclareNamespace("ord", "http://OrderTypes");`
		
		wird das XML-Dokument zu
		
		`<purchase xmlns:ord="http://OrderTypes" />`
		
		Ein weiteres Beispiel: Bei Aufruf von
		
		`purchase.DeclareNamespace("", "http://OrderTypes");`
		
		wird das XML-Dokument zu
		
		`<purchase xmlns="http://OrderTypes" />`
		
		Anmerkung: Der deklarierte Namespace wird beim Anhängen danach folgender Child-Elemente oder Attribute nach den folgenden Regeln verwendet:
		
		1. Wenn der Child-Namespace der Standard-Namespace ist, wird ein leeres Präfix verwendet.
		2. Wenn der Child-Namespace der gleiche ist wie der Parent Namespace ist, wird das Parent-Präfix verwendet.
		3. Anderfalls wird unter Verwendung des in Abschnitt "B.2: Namespace Prefix Lookup" unter https://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20021022/namespaces-algorithms.html beschriebenen Lookup-Algorithmus nach dem nächstgelegenen Präfix vom Parent bis nach oben gesucht .
		4. Wenn kein Präfix für einen Element-Namespace gefunden wird, wird ein leeres Präfix verwendet.

16.8.10 [YourSchemaType].MemberAttribute

Wenn anhand eines XML-Schemas Code generiert wird, wird für jedes Member-Attribut eines Typs eine Klasse wie diese erstellt. In der Beschreibung unten steht "AttributeType" für den Typ des Member-Attributs selbst.

Methoden

	Name	Beschreibung
	`bool Exists()`	Gibt den Wert "true" zurück, wenn das Attribut vorhanden ist.

	Name	Beschreibung
	`void Remove()`	Entfernt das Attribut aus dem übergeordneten Element.

Properties

	Name	Beschreibung
	`int EnumerationValue`	Wird nur für Enumeration Types generiert. Definiert den Attributwert mittels einer der für die möglichen Werte generierten Konstanten bzw. definiert diesen. Gibt **Invalid** zurück, wenn der Wert mit keinem der enumerierten Werte im Schema übereinstimmt.
	`Altova.Xml.Meta.Attribute Info`	Gibt ein Objekt zum Abfragen von Schemainformationen zurück (siehe `Altova.Xml.Meta.Attribute` 1675).
	`AttributeType Value`	Definiert den Attributwert bzw. ruft ihn ab.

16.8.11 [YourSchemaType].MemberElement

Wenn anhand eines XML-Schemas Code generiert wird, wird für jedes Member-Element eines Typs eine Klasse mit den folgenden Menbers erstellt. Die Klasse implementiert die `System.Collections.IEnumerable`-Standardschnittstelle, daher kann sie mit der `foreach`-Anweisung verwendet werden.

In den Beschreibungen unten steht "MemberType" für den Typ des Member-Elements selbst.

Methoden

	Name	Beschreibung
	`MemberType Append()`	Erstellt ein neues Element und hängt es an das übergeordnete Element an.
	`MemberType AppendWithPrefix(string prefix)`	Erstellt ein neues Element mit dem als Argument angegebenen Präfix und hängt es an seinen Parent an. Ein Beispiel dazu finden Sie unter Beispiel: Bestellung 1644.
	`MemberType At(int index)`	Gibt das vom Index definierte Member Element zurück.
	`System.Collections.IEnumerator GetEnumerator()`	Gibt ein Objekt für iterierende Instanzen des Member Elements zurück.
	`void Remove()`	Löscht alle Instanzen des Elements aus dem übergeordneten Element.
	`void RemoveAt(int index)`	Löscht die Instanz des durch den Index definierten Elements.

© 2016-2022 Altova GmbH

Eigenschaften

	Name	Beschreibung
	`int Count`	Gibt die Anzahl der Elemente zurück.
	`int EnumerationValue`	Wird nur für Enumeration Types generiert. Definiert den Elementwert mittels einer der für die möglichen Werte generierten Konstanten bzw. definiert diesen. Gibt **Invalid** zurück, wenn der Wert mit keinem der enumerierten Werte im Schema übereinstimmt.
	`bool Exists`	Gibt den Wert "true" zurück, wenn mindestens ein Element vorhanden ist.
	`MemberType First`	Gibt die erste Instanz des Member Elements zurück.
	`Altova.Xml.Meta.Element Info`	Gibt ein Objekt zum Abfragen von Schemainformationen zurück (siehe `Altova.Xml.Meta.Element` 1677).
	`MemberType Last`	Gibt die letzte Instanz des Member Elements zurück.
	`MemberType this[int index]`	Gibt das vom Index definierte Member Element zurück.
	`MemberType Value`	Definiert den Elementinhalt bzw. ruft ihn ab (wird nur generiert, wenn das Element mixed oder simple content haben kann).

16.9 Referenz zu generierten Klassen (Java)

Dieses Kapitel enthält eine Beschreibung der mit MapForce anhand einer DTD oder eines XML-Schemas (siehe Generieren von Code anhand von XML-Schemas oder DTDs[1607]) generierten Java-Klassen. Sie können diese Klassen in Ihren Code integrieren, um XML-Dokumente zu lesen, zu bearbeiten und zu schreiben.

Anmerkung: Der generierte Code enthält andere unterstützende Klassen, die hier nicht aufgelistet sind und Änderungen unterworfen sind.

16.9.1 com.altova.types.DateTime

Mit Hilfe dieser Klasse können XML-Attribute oder -Elemente, die Datums- und Uhrzeittypen wie z.B. `xs:dateTime` haben, verarbeitet werden.

Konstruktoren

	Name	Beschreibung
C	`public DateTime()`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf einen leeren Wert.
C	`public DateTime(DateTime newvalue)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf den als Argument bereitgestellten `DateTime`-Wert.
C	`public DateTime(int newyear, int newmonth, int newday, int newhour, int newminute, int newsecond, double newpartsecond, int newoffsetTZ)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf das Jahr, den Monat, den Tag, die Stunde, Minute, Sekunde, den Sekundenbruchteil und die Zeitzone, die als Argumente bereitgestellt wurden. Der Sekundenbruchteil newpartsecond muss zwischen 0 und 1 sein. Die Zeitzonendifferenz newoffsetTZ kann entweder positiv oder negativ sein und wird in Minuten ausgedrückt.
C	`public DateTime(int newyear, int newmonth, int newday, int newhour, int newminute, int newsecond, double newpartsecond)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf das Jahr, den Monat, den Tag, die Stunde, Minute, Sekunde und den Sekundenbruchteil, die als Argumente bereitgestellt wurden.
C	`public DateTime(int newyear, int newmonth, int newday)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf das Jahr, den Monat, den Tag, die als Argumente bereitgestellt wurden.
C	`public DateTime(Calendar newvalue)`	Initialisiert eine neue Instanz der `DateTime`-Klasse auf den als Argument bereitgestellten `java.util.Calendar`-Wert.

Methoden

	Name	Beschreibung
S	`static DateTime now()`	Gibt die aktuelle Uhrzeit als `DateTime`-Objekt zurück.

© 2016-2022 Altova GmbH

	Name	Beschreibung
S	**static** DateTime parse(String s)	Gibt ein DateTime-Objekt zurück, das aus dem als Argument bereitgestellten String-Werts geparst wird. Der folgende Beispiel-String würde erfolgreich in ein DateTime-Objekt konvertiert werden. 2015-11-24T12:54:47.969+01:00 2015-11-24T12:54:47 2015-11-24
S	**int** getDay()	Gibt den Tag der aktuellen DateTime-Instanz zurück.
S	**int** getHour()	Gibt die Stunde der aktuellen DateTime-Instanz zurück.
S	**int** getMillisecond()	Gibt die Millisekunde der aktuellen DateTime-Instanz als Ganzzahlwert zurück.
S	**int** getMinute()	Gibt die Minute der aktuellen DateTime-Instanz zurück.
S	**int** getMonth()	Gibt den Monat der aktuellen DateTime-Instanz zurück.
S	**double** getPartSecond()	Gibt die Nachkommastellen der Sekunde der aktuellen DateTime-Instanz als **double** -Wert zurück. Der Rückgabewert ist größer als Null und kleiner als eins, z.B.: 0,313
S	**int** getSecond()	Gibt die Sekunde der aktuellen DateTime-Instanz zurück.
S	**int** getTimezoneOffset()	Gibt den Zeitzonenunterschied zur aktuellen DateTime-Instanz in Minuten zurück. Die Zeitzone "UTC-01:00" würde z.B. als folgender Wert zurückgegeben: -60
S	Calendar getValue()	Gibt die aktuelle DateTime Instanz als java.util.Calendar-Wert zurück.
S	**int** getWeekday()	Gibt den Wochentag der aktuellen DateTime-Instanz zurück. Die Werte liegen im Bereich von 0 bis 6, wobei 0 Montag ist (ISO-8601).
S	**int** getYear()	Gibt das Jahr der aktuellen DateTime-Instanz zurück.
S	**int** hasTimezone()	Gibt Informationen über die Zeitzone der aktuellen DateTime-Instanz zurück. Mögliche Rückgabewerte sind: CalendarBase.TZ_MISSING — Es ist kein Zeitzonenunterschied definiert. CalendarBase.TZ_UTC — Die Zeitzone ist UTC.

	Name	Beschreibung
		`CalendarBase.TZ_OFFSET` Es ist ein Zeitzonenunterschied definiert.
S	**void** setDay(**int** nDay)	Setzt den Tag der aktuellen `DateTime`-Instanz auf den als Argument bereitgestellten Wert.
S	**void** setHasTimezone(**int** nHasTZ)	Setzt die Zeitzoneninformation der aktuellen `DateTime`-Instanz auf den als Argument bereitgestellten Wert. Mit Hilfe dieser Methode können die Zeitzoneninformationen entfernt werden oder die Zeitzone kann auf UTC (Coordinated Universal Time) gesetzt werden. Gültige Werte für das nHasTZ-Argument: `CalendarBase.TZ_MISSING` Setzt die Zeitzone auf nicht definiert. `CalendarBase.TZ_UTC` Setzt die Zeitzone auf UTC. `CalendarBase.TZ_OFFSET` Wenn das aktuelle Objekt einen Zeitzonenunterschied hat, so bleibt dieser unverändert.
S	**void** setHour(**int** nHour)	Setzt die Stunde der aktuellen `DateTime`-Instanz auf den als Argument bereitgestellten Wert.
S	**void** setMinute(**int** nMinute)	Setzt die Minute der aktuellen `DateTime`-Instanz auf den als Argument bereitgestellten Wert.
S	**void** setMonth(**int** nMonth)	Setzt den Monat der aktuellen `DateTime`-Instanz auf den als Argument bereitgestellten Wert.
S	**void** setPartSecond(**double** nPartSecond)	Setzt die Sekundenbruchteile der aktuellen `DateTime`-Instanz auf den als Argument bereitgestellten Wert.
S	**void** setSecond(**int** nSecond)	Setzt die Sekunde der aktuellen `DateTime`-Instanz auf den als Argument bereitgestellten Wert.
S	**void** setTimezoneOffset(**int** nOffsetTZ)	Setzt den Zeitzonenunterschied der aktuellen `DateTime`-Instanz auf den als Argument bereitgestellten Wert. Der Wert nOffsetTZ muss eine (positive oder negative) Ganzzahl sein und in Minuten ausgedrückt werden.
S	**void** setYear(**int** nYear)	Setzt das Jahr der aktuellen `DateTime`-Instanz auf den als Argument bereitgestellten Wert.
S	String toString()	Gibt die String-Darstellung der aktuellen `DateTime`-Instanz zurück, z.B.: `2015-11-24T15:50:56.968+01:00`

© 2016-2022 Altova GmbH

Beispiele

Stellen Sie sicher, dass die Altova-Typen importiert wurden, bevor Sie die folgenden Codefragmente in Ihrem Programm verwenden.

```
import com.altova.types.*;
```

In den folgenden Codefragmenten werden verschiedene Möglichkeiten zur Erstellung von `DateTime`-Objekten gezeigt:

```
protected static void DateTimeExample1()
{
   // Initialize a new instance of the DateTime class to the current time
   DateTime dt = new DateTime(DateTime.now());
   System.out.println("DateTime created from current date and time: " + dt.toString());

   // Initialize a new instance of the DateTime class by supplying the parts
   DateTime dt1 = new DateTime(2015, 11, 23, 14, 30, 24, .459);
   System.out.println("DateTime from parts (no timezone): " + dt1.toString());

   // Initialize a new instance of the DateTime class by supplying the parts
   DateTime dt2 = new DateTime(2015, 11, 24, 14, 30, 24, .459, -60);
   System.out.println("DateTime from parts (with negative timezone): " + dt2.toString());

   // Initialize a new instance of the DateTime class by parsing a string value
   DateTime dt3 = DateTime.parse("2015-11-24T12:54:47.969+01:00");
   System.out.println("DateTime parsed from string: " + dt3.toString());
}
```

Im folgenden Codefragment wird gezeigt, wie Werte aus `DateTime`-Objekte ermittelt werden:

```
protected static void DateTimeExample2()
   {
      // Initialize a new instance of the DateTime class to the current time
      DateTime dt = new DateTime(DateTime.now());

      // Output the formatted year, month, and day of this DateTime instance
      String str1 = String.format("Year: %d; Month: %d; Day: %d;", dt.getYear(),
dt.getMonth(), dt.getDay());
      System.out.println(str1);

      // Output the formatted hour, minute, and second of this DateTime instance
      String str2 = String.format("Hour: %d; Minute: %d; Second: %d;", dt.getHour(),
dt.getMinute(), dt.getSecond());
      System.out.println(str2);

      // Return the timezone (in minutes) of this DateTime instance
      System.out.println("Timezone: " + dt.getTimezoneOffset());

      // Get the DateTime as a java.util.Calendar value
```

```
      java.util.Calendar dt_java = dt.getValue();
      System.out.println("" + dt_java.toString());

      // Return the day of week of this DateTime instance
      System.out.println("Weekday: " + dt.getWeekday());

      // Check whether the DateTime instance has a timezone defined
      switch(dt.hasTimezone())
      {
         case CalendarBase.TZ_MISSING:
            System.out.println("No timezone.");
            break;
         case CalendarBase.TZ_UTC:
            System.out.println("The timezone is UTC.");
            break;
         case CalendarBase.TZ_OFFSET:
            System.out.println("This object has a timezone.");
            break;
         default:
            System.out.println("Unable to determine whether a timezone is defined.");
            break;
      }
   }
```

Im folgenden Codefragment wird gezeigt, wie der Zeitzonenunterschied eines `DateTime`-Objekts geändert wird:

```
protected static void DateTimeExample3()
{
   // Create a new DateTime object with timezone -0100 UTC
   DateTime dt = new DateTime(2015, 11, 24, 14, 30, 24, .459, -60);
   // Output the value before the change
   System.out.println("Before: " + dt.toString());
   // Change the offset to +0100 UTC
   dt.setTimezoneOffset(60);
   // Output the value after the change
   System.out.println("After:  " + dt.toString());
}
```

16.9.2 com.altova.types.Duration

Mit dieser Klasse können Sie XML-Attribute oder -Elemente vom Typ `xs:duration` verarbeiten.

Konstruktoren

	Name	Beschreibung
	`Duration(Duration newvalue)`	Initialisiert eine neue Instanz der `Duration`-Klasse auf das als Argument bereitgestellte `Duration`-Objekt.

© 2016-2022 Altova GmbH

	Name	Beschreibung
C	`Duration(int newyear, int newmonth, int newday, int newhour, int newminute, int newsecond, double newpartsecond, boolean newisnegative)`	Initialisiert eine neue Instanz der `Duration`-Klasse auf eine Zeitdauer, die anhand der als Argumente bereitgestellten Bestandteile konstruiert wird.

Methoden

	Name	Beschreibung
S	`static Duration getFromDayTime( int newday, int newhour, int newminute, int newsecond, double newpartsecond )`	Gibt ein `Duration`-Objekt zurück, das anhand der als Argument bereitgestellten Anzahl von Tagen, Stunden, Minuten, Sekunden und Sekundenbruchteile erstellt wird.
S	`static Duration getFromYearMonth( int newyear, int newmonth )`	Gibt ein `Duration`-Objekt zurück, das anhand der als Argument bereitgestellten Anzahl von Jahren und Monaten erstellt wird.
S	`static Duration parse( String s )`	Gibt ein `Duration`-Objekt zurück, das anhand des als Argument bereitgestellten String erstellt wurde. So kann z.B. anhand des String -P1Y1M1DT1H1M1.333S eine negative Zeitdauer von einem Jahr, einem Monat, einem Tag, einer Stunde, einer Minute, einer Sekunde und 0,333 Zehntelsekunden erstellt werden. Um eine negative Zeitdauer zu erstellen, hängen Sie das Minuszeichen (-) an den String an.
S	`static Duration parse( String s, ParseType pt )`	Gibt ein `Duration`-Objekt zurück, das unter Verwendung eines bestimmten Parse-Formats anhand des als Argument bereitgestellten String erstellt wird. Als Parse-Format kann eines der folgenden verwendet werden: **ParseType.DAYTIME** – Kann verwendet werden, wenn der String **s** aus einem der folgenden Bestandteile besteht: Tage, Stunden, Minuten, Sekunden, Sekundenbruchteile, z.B. -P4DT4H4M4.774S. **ParseType.DURATION** – Kann verwendet werden, wenn der String **s** aus einem der folgenden Bestandteile besteht: Jahre, Monate, Tage, Stunden, Minuten, Sekunden, Sekundenbruchteile, z.B. P1Y1M1DT1H1M1.333S. **ParseType.YEARMONTH** – Kann verwendet werden, wenn der String **s** aus einem der folgenden

	Name	Beschreibung
		Bestandteile besteht: Zum Beispiel: P3Y2M.
	`int getDay()`	Gibt die Anzahl der Tage in der aktuellen `Duration`-Instanz zurück.
	`long getDayTimeValue()`	Gibt den Tag- und Uhrzeitwert (in Millisekunden) der aktuellen `Duration`-Instanz zurück. Jahre und Monate werden ignoriert.
	`int getHour()`	Gibt die Anzahl der Stunden in der aktuellen `Duration`-Instanz zurück.
	`int getMillisecond()`	Gibt die Anzahl der Millisekunden in der aktuellen `Duration`-Instanz zurück.
	`int getMinute()`	Gibt die Anzahl der Minuten in der aktuellen `Duration`-Instanz zurück.
	`int getMonth()`	Gibt die Anzahl der Monate in der aktuellen `Duration`-Instanz zurück.
	`double getPartSecond()`	Gibt die Anzahl der Sekundenbruchteile in der aktuellen `Duration`-Instanz zurück.
	`int getSecond()`	Gibt die Anzahl der Sekunden in der aktuellen `Duration`-Instanz zurück.
	`int getYear()`	Gibt die Anzahl der Jahre in der aktuellen `Duration`-Instanz zurück.
	`int getYearMonthValue()`	Gibt das Jahr und den Monatswert (in Monaten) der aktuellen `Duration`-Instanz zurück. Tage, Stunden, Sekunden und Millisekunden werden ignoriert.
	`boolean isNegative()`	Gibt den Booleschen Wert **true** zurück, wenn die aktuelle `Duration`-Instanz negativ ist.
	`void setDayTimeValue(long l)`	Setzt die Zeitdauer auf die als Argument angegebene Anzahl der Millisekunden, wobei sich dies nur auf den Tag- und Uhrzeitbestandteil der Zeitdauer auswirkt.
	`void setNegative( boolean isnegative )`	Konvertiert die aktuelle `Duration`-Instanz in eine negative Zeitdauer.
	`void setYearMonthValue(int l)`	Setzt die Zeitdauer auf die als Argument angegebene Anzahl der Monate. Nur der Jahr- und Monatsteil der Zeitdauer ist davon betroffen.
	`String toString()`	Gibt die String-Darstellung der aktuellen `Duration`-Instanz zurück, z.B.: `-P4DT4H4M4.774S`

© 2016-2022 Altova GmbH

	Name	Beschreibung
	`String toYearMonthString()`	Gibt die String-Darstellung des YearMonth-Teils der aktuellen `Duration`-Instanz zurück, z.B.: `P1Y2M`

Beispiele

Stellen Sie sicher, dass die Altova-Typen importiert wurden, bevor Sie die folgenden Codefragmente in Ihrem Programm verwenden:

```
import com.altova.types.*;
import com.altova.types.Duration.ParseType;
```

In den folgenden Codefragmenten werden verschiedene Möglichkeiten zur Erstellung von `Duration`-Objekten gezeigt:

```
protected static void ExampleDuration()
{
   // Create a negative duration of 1 year, 1 month, 1 day, 1 hour, 1 minute, 1 second,
   // and 0.333 fractional second parts
   Duration dr = new Duration(1, 1, 1, 1, 1, 1, .333, true);

   // Create a duration from an existing Duration object
   Duration dr1 = new Duration(dr);

   // Create a duration of 4 days, 4 hours, 4 minutes, 4 seconds, .774 fractional second
parts
   Duration dr2 = Duration.getFromDayTime(4, 4, 4, 4, .774);

   // Create a duration of 3 years and 2 months
   Duration dr3 = Duration.getFromYearMonth(3, 2);

   // Create a duration from a string
   Duration dr4 = Duration.parse("-P4DT4H4M4.774S");

   // Create a duration from a string, using specific parse formats
   Duration dr5 = Duration.parse("-P1Y1M1DT1H1M1.333S", ParseType.DURATION);
   Duration dr6 = Duration.parse("P3Y2M", ParseType.YEARMONTH);
   Duration dr7 = Duration.parse("-P4DT4H4M4.774S", ParseType.DAYTIME);
}
```

Im folgenden Codefragment wird gezeigt, wie Sie den Wert von `Duration`-Objekten abrufen und definieren:

```
protected static void DurationExample2()
{
   // Create a duration of 1 year, 2 month, 3 days, 4 hours, 5 minutes, 6 seconds,
       // and 333 milliseconds
   Duration dr = new Duration(1, 2, 3, 4, 5, 6, .333, false);
```

```
   // Output the number of days in this duration
   System.out.println(dr.getDay());

   // Create a positive duration of one year and 333 milliseconds
   Duration dr1 = new Duration(1, 0, 0, 0, 0, 0, .333, false);
   // Output the day and time value in milliseconds
   System.out.println(dr1.getDayTimeValue());

   // Create a positive duration of 1 year, 1 month, 1 day, 1 hour, 1 minute, 1 second,
       // and 333 milliseconds
   Duration dr2 = new Duration(1, 1, 1, 1, 1, 1, .333, false);
   // Output the year and month value in months
   System.out.println(dr2.getYearMonthValue());

   // Create a positive duration of 1 year and 1 month
   Duration dr3 = new Duration(1, 1, 0, 0, 0, 0, 0, false);
   // Output the value
   System.out.println("The duration is now: " + dr3.toString());
   // Set the DayTime part of duration to 1000 milliseconds
   dr3.setDayTimeValue(1000);
   // Output the value
   System.out.println("The duration is now: " + dr3.toString());
   // Set the YearMonth part of duration to 1 month
   dr3.setYearMonthValue(1);
   // Output the value
   System.out.println("The duration is now: " + dr3.toString());
   // Output the year and month part of the duration
   System.out.println("The YearMonth part of the duration is: " +
dr3.toYearMonthString());
}
```

16.9.3 com.altova.xml.meta.Attribute

Mit Hilfe dieser Klasse können Sie Schemainformationen über anhand von Attributen generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines Attributs in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über ein bestimmtes im XML-Schema definiertes Attribut programmatisch abzurufen.

Methoden

	Name	Beschreibung
	`SimpleType getDataType()`	Gibt den Typ des Attributinhalts zurück.
	`String getLocalName()`	Gibt den lokalen Namen des Attributs zurück.
	`String getNamespaceURI()`	Gibt die Namespace URI des Attributs zurück.
	`boolean` `isRequired()`	Gibt "true" zurück, wenn das Attribut obligatorisch ist.

© 2016-2022 Altova GmbH

16.9.4 com.altova.xml.meta.ComplexType

Mit Hilfe dieser Klasse können Sie Schemainformationen über anhand von complexTypes generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines complexType in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über einen bestimmten im XML-Schema definierten complexType programmatisch abzurufen.

Methoden

	Name	Beschreibung
	`Attribute findAttribute(String localName, String namespaceURI)`	Sucht das Attribut mit dem definierten lokalen Namen und der Namespace URI
	`Element findElement(String localName, String namespaceURI)`	Sucht das Element mit dem definierten lokalen Namen und der Namespace URI.
	`Attribute[] GetAttributes()`	Gibt eine Liste aller Attribute zurück.
	`ComplexType getBaseType()`	Gibt den Basistyp dieses Typs zurück.
	`SimpleType getContentType()`	Gibt dem simpleType des Inhalts zurück.
	`Element[] GetElements()`	Gibt eine Liste aller Elemente zurück.
	`String getLocalName()`	Gibt den lokalen Namen des Typs zurück.
	`String getNamespaceURI()`	Gibt die Namespace URI des Typs zurück.

16.9.5 com.altova.xml.meta.Element

Mit Hilfe dieser Klasse können Sie Informationen über anhand von Schema-Elementen generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines Elements in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über ein bestimmtes im XML-Schema definiertes Element programmatisch abzurufen.

Methoden

	Name	Beschreibung
	`ComplexType getDataType()`	Gibt den Typ des Elements zurück. Beachten Sie, dass dies immer ein complexType ist, selbst wenn er im Originalschema als simpleType deklariert wurde. Verwenden Sie `getContentType()` des zurückgegebenen Objekts zum den simple content type abzurufen.
	`String getLocalName()`	Gibt den lokalen Namen des Elements zurück.

	Name	Beschreibung
	`int getMaxOccurs()`	Gibt den im Schema definierten maxOccurs Wert zurück.
	`int getMinOccurs()`	Gibt den im Schema definierten minOccurs Wert zurück.
	`String getNamespaceURI()`	Gibt die Namespace URI des Elements zurück.

16.9.6 com.altova.xml.meta.SimpleType

Mit Hilfe dieser Klasse können Sie Schemainformationen über anhand von simpleTypes generierte Klassen aufrufen. Beachten Sie, dass diese Klasse nicht dazu gedacht ist, dynamische Informationen über bestimmte Instanzen eines simpleType in einem XML-Dokument bereitzustellen. Es dient stattdessen dazu, Informationen über einen bestimmten im XML-Schema definierten simpleTypes programmatisch abzurufen.

Methoden

	Name	Beschreibung
	`SimpleType getBaseType()`	Gibt den Basistyp dieses Typs zurück.
	`String[] getEnumerations()`	Gibt ein Array aller Enumeration Facets zurück.
	`int getFractionDigits()`	Gibt den Wert dieses Facet zurück.
	`int getLength()`	Gibt den Wert dieses Facet zurück.
	`String getLocalName()`	Gibt den lokalen Namen des Typs zurück.
	`String getMaxExclusive()`	Gibt den Wert dieses Facet zurück.
	`String getMaxInclusive()`	Gibt den Wert dieses Facet zurück.
	`int getMaxLength()`	Gibt den Wert dieses Facet zurück.
	`String getMinExclusive()`	Gibt den Wert dieses Facet zurück.
	`String getMinInclusive()`	Gibt den Wert dieses Facet zurück.
	`int getMinLength()`	Gibt den Wert dieses Facet zurück.
	`String getNamespaceURI()`	Gibt die Namespace URI des Typs zurück.
	`String[] getPatterns()`	Gibt ein Array aller Pattern Facets zurück.
	`int getTotalDigits()`	Gibt den Wert dieses Facet zurück.

	`int getWhitespace()`	Gibt den Wert des Whitespace Facet zurück, der einer der folgenden ist:
		`com.altova.typeinfo.WhitespaceType.Whitespace_Unknown`
		`com.altova.typeinfo.WhitespaceType.Whitespace_Preserve`
		`com.altova.typeinfo.WhitespaceType.Whitespace_Replace`

© 2016-2022 Altova GmbH

	Name	Beschreibung
		`com.altova.typeinfo.WhitespaceType.Whitespace_Collapse`

16.9.7 com.[YourSchema].[Doc]

Wenn anhand eines XML-Schemas Code generiert wird, stellt der generierte Code eine Dokumentklasse mit demselben Namen wie dem des Schemas bereit. Diese Klasse enthält alle möglichen Root-Elemente als Members sowie die folgenden Members. Beachten Sie, dass "Doc" in den unten aufgelisteten Methodennamen für den Namen der generierten Dokumentklasse selbst steht.

Methoden

	Name	Beschreibung
S	`static Doc createDocument()`	Erstellt ein neues leeres XML-Dokument.
S	`static void declareAllNamespacesFromSchema(com.altova.xml.ElementType node)`	Deklariert alle Namespaces aus dem XML-Schema an dem als Argument angegebenen Element (normalerweise dem XML-Root-Element). Diese Methode eignet sich dann, wenn Ihr Schema mehrere Namespace-Deklarationen, von denen jedes auf ein Präfix gemappt ist, hat, und Sie alle davon für das als Argument angegebene Element deklarieren möchten.
S	`static Doc loadFromBinary(byte[] xml)`	Lädt ein XML-Dokument aus einem Byte Array.
S	`static Doc loadFromFile(String fileName)`	Lädt ein XML-Dokument aus einer Datei.
S	`static Doc loadFromString(String xml)`	Lädt ein XML-Dokument aus einem String.
	`byte[] saveToBinary(boolean prettyPrint)`	Speichert ein XML-Dokument mit optionaler Pretty-Print-Formatierung in einem Byte Array.
	`byte[] saveToBinary(boolean prettyPrint, String encoding)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einem Byte-Array.
	`byte[] saveToBinary(boolean prettyPrint, String encoding, boolean bigEndian, boolean writeBOM)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einem Byte-Array. Für Unicode-Kodierungen können die Bytefolge und Unicode-Bytefolge-Markierung definiert werden.
	`void saveToFile(String fileName, boolean prettyPrint)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung in einer Datei.
	`void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit UTF-8-Kodierung in einer Datei. Wenn

	Name	Beschreibung
		`omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.
	`void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl, String encoding)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einer Datei. Wenn `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.
	`void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl, String encoding, boolean bBigEndian, boolean bBOM)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einer Datei. Wenn `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben. Für Unicode-Kodierungen können die Bytefolge und Unicode-Bytefolge-Markierung definiert werden.
	`void saveToFile(String fileName, boolean prettyPrint, String encoding)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einer Datei.
	`void saveToFile(String fileName, boolean prettyPrint, String encoding, boolean bBigEndian, boolean bBOM)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung mit der definierten Kodierung in einer Datei. Für Unicode-Kodierungen können die Bytefolge und Unicode-Bytefolge-Markierung definiert werden.
	`String saveToString(boolean prettyPrint)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung in einem String.
	`String saveToString(boolean prettyPrint, boolean omitXmlDecl)`	Speichert ein XML-Dokument mit optionaler "pretty-print"-Formatierung in einem String. Wenn `omitXmlDecl` auf "true" gesetzt ist, wird die XML-Deklaration nicht geschrieben.
	`void setSchemaLocation(String schemaLocation)`	Fügt ein `xsi:schemaLocation`- oder `xsi:noNamespaceSchemaLocation`-Attribut zum Root-Element hinzu. Es muss bereits ein Root-Element vorhanden sein.

16.9.8 com.[YourSchema].[ElementType]

Diese Klasse enthält Methoden zur Bearbeitung von XML-Elementen aus Ihrem Schema. Die Methoden dieser Klasse können für Elemente, nicht aber für das XML-Dokument selbst, aufgerufen werden. Beachten Sie, dass Sie die Klasse nicht direkt instantiieren müssen, um Methoden dieser Klasse aufzurufen. Jedes mit den Methoden **`append()`** oder **`appendWithPrefix()`** erstelltes Element hat den Typ [`ElementType`].

Methoden

	Name	Beschreibung
	`void declareNamespace(String prefix, String nsURI)`	Diese Methode erhält zwei Argumente, beide vom Typ "String": das gewünschte Präfix und die gewünschte Namespace URI. Das als Argument bereitgestellte Präfix

© 2016-2022 Altova GmbH

	Name	Beschreibung

		wird auf den als Argument bereitgestellten Namespace URI-Wert gemappt. Wenn das als Argument angegebene Präfix leer ist, erstellt die Methode die Standard-Namespace-Deklaration im Element oder setzt diese außer Kraft.
		
		Angenommen, das XML-Dokument hat ein XML-Element namens "purchase". Bei Aufruf von
		
		`purchase.declareNamespace("ord", "http://OrderTypes");`
		
		wird das XML-Dokument zu
		
		`<purchase xmlns:ord="http://OrderTypes" />`
		
		Ein weiteres Beispiel: Bei Aufruf von
		
		`purchase.declareNamespace("", "http://OrderTypes");`
		
		wird das XML-Dokument zu
		
		`<purchase xmlns="http://OrderTypes" />`
		
		Anmerkung: Der deklarierte Namespace wird beim Anhängen danach folgender Child-Elemente oder Attribute nach den folgenden Regeln verwendet:
		
		1. Wenn der Child-Namespace der Standard-Namespace ist, wird ein leeres Präfix verwendet.
		2. Wenn der Child-Namespace der gleiche ist wie der Parent Namespace ist, wird das Parent-Präfix verwendet.
		3. Anderfalls wird unter Verwendung des in Abschnitt "B.2: Namespace Prefix Lookup" unter https://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20021022/namespaces-algorithms.html beschriebenen Lookup-Algorithmus nach dem nächstgelegenen Präfix vom Parent bis nach oben gesucht .
		4. Wenn kein Präfix für einen Element-Namespace gefunden wird, wird ein leeres Präfix verwendet.

16.9.9 com.[YourSchema].[YourSchemaType].MemberAttribute

Wenn anhand eines XML-Schemas Code generiert wird, wird für jedes Member-Attribut eines Typs eine Klasse erstellt. In der Beschreibung unten steht "AttributeType" für den Typ des Member-Attributs selbst.

Methoden

	Name	Beschreibung
	`boolean exists()`	Gibt den Wert "true" zurück, wenn das Attribut vorhanden ist.
	`int getEnumerationValue()`	Wird nur für Enumeration Types generiert. Gibt eine der für die möglichen Werte generierten Konstanten zurück oder **Invalid,** wenn der Wert mit keinem der Enumerationswerte im Schema übereinstimmt.
	`com.altova.xml.meta.Attribute getInfo()`	Gibt ein Objekt zum Abfragen von Schemainformationen zurück (siehe `com.altova.xml.meta.Attribute` 1692).
	`AttributeType getValue()`	Ruft den Attributwert ab.
	`void remove()`	Entfernt das Attribut aus dem übergeordneten Element.
	`void setEnumerationValue(int)`	Wird nur für Enumeration Types generiert. Übergeben Sie eine der für die möglichen Werte generierten Konstanten an diese Methode, um den Wert zu definieren.
	`void setValue(AttributeType value)`	Definiert den Attributwert.

16.9.10 com.[YourSchema].[YourSchemaType].MemberElement

Wenn anhand eines XML-Schemas Code generiert wird, wird für jedes Member-Element eines Typs eine Klasse mit den folgenden Menbers erstellt. In den Beschreibungen unten steht "MemberType" für den Typ des Member-Elements selbst.

Methoden

	Name	Beschreibung
	`MemberType append()`	Erstellt ein neues Element und hängt es an das übergeordnete Element an.
	`MemberType appendWithPrefix(String prefix)`	Erstellt ein neues Element mit dem als Argument angegebenen Präfix und hängt es an seinen Parent an. Ein Beispiel dazu finden Sie unter Beispiel: Bestellung 1644.
	`MemberType at(int index)`	Gibt die Instanz des Member Elements am angegebenen Index zurück.

© 2016-2022 Altova GmbH

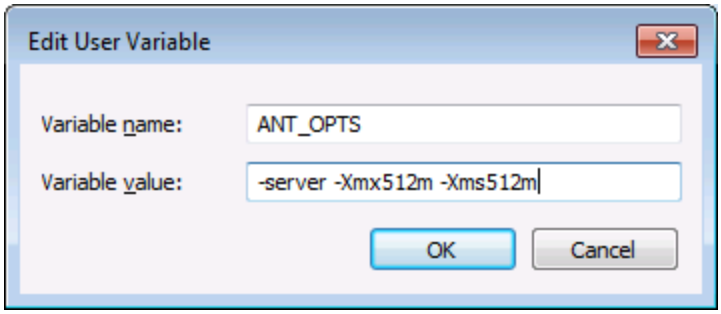
	Name	Beschreibung
	`int count()`	Gibt die Anzahl der Elemente zurück.
	`boolean exists()`	Gibt den Wert "true" zurück, wenn mindestens ein Element vorhanden ist.
	`MemberType first()`	Gibt die erste Instanz des Member Elements zurück.
	`int getEnumerationValue()`	Wird nur für Enumeration Types generiert. Gibt eine der für die möglichen Werte generierten Konstanten zurück oder **Invalid,** wenn der Wert mit keinem der Enumerationswerte im Schema übereinstimmt.
	`com.altova.xml.meta.Element getInfo()`	Gibt ein Objekt zum Abfragen von Schemainformationen zurück (siehe `com.altova.xml.meta.Element` 1693).
	`MemberType getValue()`	Ruft den Elementinhalt ab (wird nur generiert, wenn das Element simple oder mixed content haben kann).
	`java.util.Iterator iterator()`	Gibt ein Objekt für iterierende Instanzen des Member Elements zurück.
	`MemberType last()`	Gibt die letzte Instanz des Member Elements zurück.
	`void remove()`	Löscht alle Instanzen des Elements aus dem übergeordneten Element.
	`void removeAt(int index)`	Löscht die Instanz des durch den Index definierten Elements.
	`void setEnumerationValue(int index)`	Wird nur für Enumeration Types generiert. Übergeben Sie eine der für die möglichen Werte generierten Konstanten an diese Methode, um den Wert zu definieren.
	`void setValue(MemberType value)`	Definiert den Elementinhalt (wird nur generiert, wenn das Element simple oder mixed content haben kann).

16.10 Codegenerierung Tipps

Behebung von Ausnahmeereignissen im Zusammenhang mit zu wenig Arbeitsspeicher bei der Java-Kompilierung

Es kommt vor, dass bei komplexen Mappings mit großen Schemas eine so große Menge von Code generiert wird, dass es bei der Kompilierung mit Ant zu einem `java.lang.OutofMemory`-Ausnahmeereignis kommt. Um dieses Problem zu beheben:

- Fügen Sie die Umgebungsvariable `ANT_OPTS` hinzu. Damit werden bestimmte Ant-Optionen festgelegt, wie z.B. wie viel Arbeitsspeicher dem Compiler zur Verfügung steht. Sie können die Werte wie unten gezeigt definieren.



- Um sicherzustellen, dass der Compiler und der generierte Code im selben Prozess wie Ant ausgeführt wird, ändern Sie das Attribut `fork` in der Datei **build.xml** in `false` um.

Je nach verfügbarem Arbeitsspeicher auf Ihrem Computer und je nach Projektgröße müssen Sie diese Werte eventuell anpassen. Nähere Informationen finden Sie in der Java VM-Dokumentation.

Bei der Ausführung des `ant jar`-Befehls wird eventuell eine Fehlermeldung angezeigt, dass "`[...] das Archiv mehr als 65535 Einheiten enthält`". Damit diese Fehlermeldung nicht auftritt, wird empfohlen, Versionen ab Ant 1.9 zu verwenden und in der Datei **build.xml** zum Element `<jar> zip64mode="as-needed"` hinzuzufügen.

Reservieren von Methodennamen

Beim Anpassen der Codegenerierung mittels der vorgegebenen SPL-Dateien ist es unter Umständen nötig, Methodennamen zu reservieren. Gehen Sie dazu folgendermaßen vor:

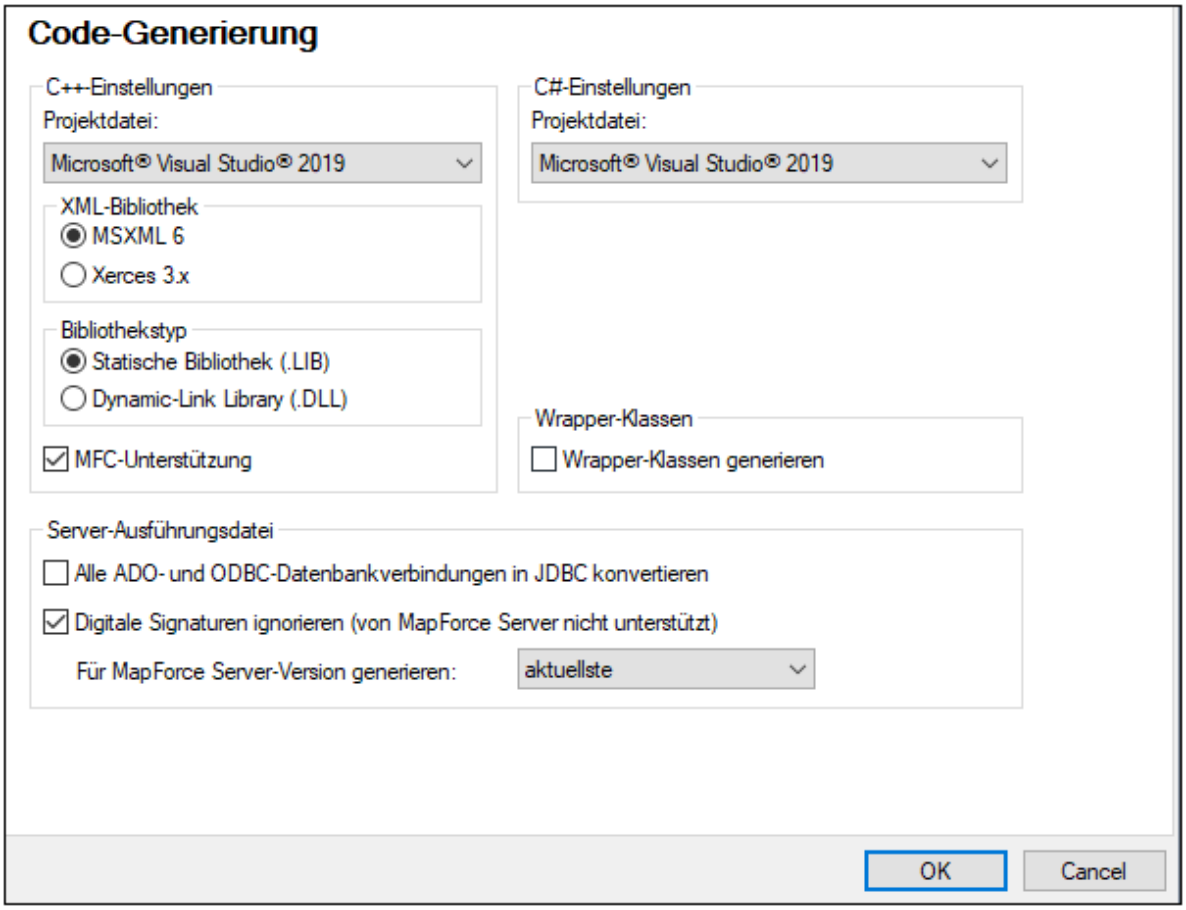
1. Navigieren Sie zum Programminstallationsverzeichnis z.B. `C:\Programme\Altova\MapForce2023`.
2. Navigieren Sie im Unterverzeichnis **spl** zum Verzeichnis für die entsprechende Programmiersprache z.B. **..\spl\java**.
3. Öffnen Sie entweder die Datei **settings.spl** und fügen Sie in den Abschnitt "reserve" eine neue Zeile ein, z.B. reserve `"myReservedWord"`.
4. Generieren Sie den Programmcode neu.

© 2016-2022 Altova GmbH

16.11 Code Generator-Optionen

Um die MapForce-Einstellungen für die Codegenerierung zu ändern:

- Klicken Sie im Menü **Extras** auf **Optionen** und anschließend auf **Code-Generierung**.



Es stehen die folgenden Einstellungen zur Verfügung.

C++-Einstellungen	Definiert die Compiler-Einstellungen für die C++-Umgebung: • die Visual Studio Version (2013, 2015, 2017, 2019, 2022) • die XML-Bibliothek (MSXML, Xerces 3.x) • ob statische oder dynamische Bibliotheken generiert werden müssen. • ob Code mit oder ohne MFC-Unterstützung generiert werden muss.
C#-Einstellungen	Wählen Sie die Option **Microsoft .NET Core .NET 3.1**, **Microsoft .NET 5.0** oder **Microsoft .NET 6.0** aus, um eine Visual Studio-Projektmappe für die entsprechende Plattform zu generieren.

	Wenn Sie die **.NET Framework**-Plattform für eine bestimmte Visual Studio-Version auswählen müssen, wählen Sie eine beliebige der **Microsoft Visual Studio 2010-2019**-Optionen aus - in diesem Fall ist das Ziel der generierten Projektmappe die .NET Framework-Version für die entsprechende Visual Studio-Version.
Wrapper-Klassen	Gestattet die Generierung von Wrapper-Klassen für XML-Schemas, siehe Generieren von Code anhand von XML-Schemas oder DTDs 1607. Diese Wrapper-Klassen können von benutzerdefiniertem Code wie z.B. von MapForce-generiertem Code verwendet werden.
Server-Ausführungsdatei	Diese Optionen werden angewendet, wenn Sie Mappings zu MapForce Server-Ausführungsdateien kompilieren. Sie wirken sich nicht auf die Generierung von C#, C++ oder Java-Code aus. Nähere Informationen dazu finden Sie unter Kompilieren von Mappings zu MapForce Server-Ausführungsdateien 1442.

© 2016-2022 Altova GmbH

16.12 SPL-Referenz

In diesem Abschnitt erhalten Sie eine Übersicht über Spy Programming Language (SPL), die Template-Sprache von Code Generator.

Es wird vorausgesetzt, dass Sie bereits über Programmiererfahrung verfügen und mit Operatoren, Funktionen, Variablen und Klassen sowie den Grundzügen von in SPl häufig verwendeter objektorientierter Programmierung vertraut sind.

Im Ordner \MapForce\spl finden Sie die von MapForce verwendeten Vorlagen**.** Anhand dieser Dateien können Sie Ihre eigenen Vorlagen entwickeln.

Funktionsweise von Code Generator

Die Basis von Code Generator bilden die Vorlagendateien (.spl) und das von MapForce bereitgestellte Objektmodell. Die Vorlagendateien enthalten SPL-Anweisungen zum Erstellen von Dateien, Lesen von Informationen aus dem Objektmodell und Ausführen von Berechnungen, die in der Zielprogrammiersprache auch Literalcodefragmente enthalten.

Die Vorlagendatei wird von Code Generator interpretiert und je nach Vorlage wird eine **.cpp, .java, .cs-**Quellcodedatei Projektdatei oder eine andere Dateiart ausgegeben.
Der Quellcode wird anschließend in eine exe-Datei kompiliert. Nach Starten der exe-Datei greift diese auf die durch die Schema-Datei beschriebenen XML-Daten zu.

SPL-Dateien haben Zugriff auf die verschiedensten Informationen, die aus den Quellschemas geholt werden. Beachten Sie bitte, dass eine SPL-Datei nicht an ein bestimmtes Schema gebunden ist, sondern Zugriff auf alle Schemas erlaubt! Stellen Sie sicher, dass Sie Ihre SPL-Dateien generisch anlegen und Strukturen, die nur auf bestimmte Schemas zutreffen, vermeiden.

Beispiel: Erstellen einer neuen Datei in SPL

Dies ist ein sehr einfacher SPL-Code. Damit wird eine Datei names **test.cpp** erstellt. Das Include Statement wird innerhalb davon positioniert. Der Befehl "close" schließt und beendet die Vorlage.

```
[create "test.cpp"]
#include "stdafx.h"
[close]
```

16.12.1 Grundlegende SPL-Struktur

Eine SPL-Datei enthält Literaltext, der ausgegeben werden soll, der zwischendurch Code Generator-Anweisungen enthält.

Code Generator-Anweisungen sind in eckige Klammern eingeschlossen '**[**' und '**]**'. Innerhalb einer Klammer können mehrere Anweisungen stehen. Zusätzliche Anweisungen müssen durch eine neue Zeile oder einen Doppelpunkt ':' getrennt werden.

Gültige Beispiele sind:

```
[$x = 42
$x = $x + 1]
```

oder

```
[$x = 42: $x = $x + 1]
```

Hinzufügen von Text zu Dateien

Text, der nicht innerhalb von **[** und **]** steht, wird direkt in die Ausgabedatei geschrieben. Falls es keine aktuelle Ausgabedatei gibt, wird der Text ignoriert (eine Anleitung zum Erstellen einer Ausgabedatei finden Sie unter Verwendung von Dateien[1709]).
Um eckige Klammern als Literale auszugeben, setzen Sie davor als Excape-Zeichen einen umgekehrten Schrägstrich: **\[** und **\]**. Um einen umgekehrten Schrägstrich auszugeben, verwenden Sie ****.

Kommentare

Kommentare innerhalb eines Anweisungsblocks beginnen immer mit einem **'** Zeichen und enden an der nächsten Zeile oder an einem Zeichen, zum Beenden eines Blocks **]**.

16.12.2 Deklarationen

Die folgenden Anweisungen werden beim Parsen der SPL-Vorlagendatei ausgewertet. Flusssteuerungsanweisungen wie z.B. Bedingungen, Schleifen oder Subroutinen haben **keinen** Einfluss darauf und werden immer nur einmal ausgewertet.

Die Groß- und Kleinschreibung spielt wie bei allen Schlüsselwörtern in SPL keine Rolle.

Alle diese Deklarationen müssen unbedingt innerhalb von eckigen Klammern stehen.

map ... to ...

```
map mapname key to value [, key to value ]...
```

Diese Anweisung fügt Informationen zu einer Zuordnung hinzu. Beispiele für konkrete Anwendungen siehe unten.

```
map schemanativetype schematype to typespec
```

Der angegebene built-in XML-Schematyp wird unter Verwendung der angegebenen Formatierung auf den angegebenen Native Type oder die Klasse gemappt. Diese Einstellung gilt nur für die Codegenerierung für Version 2007r3 und höher. Typespec ist ein Native Type oder ein Klassenname, gefolgt von einem Komma, gefolgt vom der Klasseninstanz des Formatters..

© 2016-2022 Altova GmbH

Beispiel:

```
map schemanativetype "double" to "double,Altova::DoubleFormatter"
```

map type ... to ...

```
map type schematype to classname
```

Der angegebene vordefinierte XML-Schematyp wird auf die angegebene Klasse gemappt. Diese Einstellung gilt nur für die Codegenerierung für Version 2007 oder niedriger.

Beispiel:

```
map type "float" to "CSchemaFloat"
```

default ... is ...

```
default setting is value
```

Mit dieser Anweisung können Sie festlegen, wie Klassen- und Member-Namen vom XML-Schema abgeleitet werden.
Beachten Sie, dass die die Groß- und Kleinschreibung bei Einstellungsnamen relevant ist.

Beispiel:

```
default "InvalidCharReplacement" is "_"
```

Einstellungsname	Erklärung
ValidFirstCharSet	Zulässige Zeichen zum Starten eines Identifier
ValidCharSet	Zulässige Zeichen für andere Zeichen in einem Identifier
InvalidCharReplacement	Das Zeichen, durch das alle Zeichen in Namen ersetzt werden, die nicht im ValidCharSet enthalten sind
AnonTypePrefix	Präfix für Namen von anonymen Typen*
AnonTypeSuffix	Suffix für Namen von anonymen Typen*
ClassNamePrefix	Präfix für generierte Klassennamen
ClassNameSuffix	Suffix für generierte Klassennamen
EnumerationPrefix	Präfix für Symbolkonstanten, die für Enumerationswerte deklariert wurden

EnumerationUpperCase	"ein" für das Konvertieren der Namen der Enumerationskonstanten in Großbuchstaben
FallbackName	Verwende diesen Namen, wenn ein Name nur aus Zeichen besteht, die sich nicht im ValidCharSet befinden

* Namen anonymer Typen werden aus AnonTypePrefix + Elementname + AnonTypeSuffix erzeugt.

reserve

```
reserve word
```

Fügt das angegebene Wort zur Liste der reservierte Wörter hinzu. Damit stellen Sie sicher, dass es nie als Klasse oder Member-Name generiert wird.

Beispiel:

```
reserve "while"
```

include

Inkludiert die angegebene Datei als SPL-Quelldatei. Auf diese Art können Sie Ihre Vorlage zur einfacheren Bearbeitung und Verwaltung in mehrere Dateien aufteilen.

```
include filename
```

Beispiel:

```
include "Module.cpp"
```

16.12.3 Variablen

Jede etwas komplexere SPL-Datei enthält Variablen. Einige Variablen sind vom Code Generator vordefiniert[1707]. Sie können jederzeit neue Variablen erstellen, indem Sie diesen Werte zuweisen.

Das **$**-Zeichen wird zur **Deklarierung** oder **Verwendung** einer Variable verwendet. Vor der Variable muss immer ein **$** stehen.

Bei Variablennamen muss die **Groß- und Kleinschreibung beachtet werden.**

Variablentypen:

- Integer - wird auch als Boolescher Wert verwendet, wobei 0 false ist und alle anderen Werte true.
- String
- Objekt - wird von MapForce bereitgestellt
- Iterator (siehe foreach[1712]-Anweisung)

© 2016-2022 Altova GmbH

Variablentypen werden durch die erste Zuweisung deklariert.

```
[$x = 0]
```

x ist nun ein Integer.

```
[$x = "teststring"]
```

x wird nun als String behandelt.

Strings

String-Konstanten müssen wie im oben gezeigten Beispiel immer innerhalb von doppelten Anführungszeichen stehen. **\n** und **\t** innerhalb von doppelten Anführungszeichen werden als neue Zeile und Tabulator interpretiert, \" ist ein literales doppeltes Anführungszeichen und \\ ist ein umgekehrter Schrägstrich. String-Konstanten können auch mehrere Zeilen einnehmen.

Bei der Verkettung von Strings wird das **&** Zeichen verwendet.

```
[$BasePath = $outputpath & "/" & $JavaPackageDir]
```

Objekte

Objekte stehen für die Informationen, die im XML-Schema/in den XML-Schemas, den Datenbankstrukturen, Textdateien und Mappings enthalten sind. Objekte haben **Eigenschaften**, die über den **.** Operator aufgerufen werden können. In SPL können keine neuen Objekte erstellt werden (sie werden durch den Code Generator vordefiniert, aus dem Eingabe-Mapping abgeleitet). Sie können Objekte jedoch Variablen zuweisen.

Beispiel:

```
class [=$class.Name]
```

In diesem Beispiel wird das Wort "class" ausgegeben, gefolgt von einem Leerzeichen und dem Wert der Eigenschaft **Name** des Objekts **$class**.

16.12.4 Vordefinierte Variablen

Nachdem eine Schema-Datei von Code Generator analysiert wurde, sind die in der Tabelle aufgelisteten Objekte im Template-Prozessor vorhanden.

Name	Typ	Beschreibung
$schematype	integer	1 für DTD, 2 für XML-Schema
$TheLibrary	Library 1716	Die anhand des XML-Schemas oder der DTD abgeleitete Bibliothek

$module	string	Name des Quellschemas ohne Erweiterung
$outputpath	string	Der vom Benutzer definierte Ausgabepfad oder der Standardausgabepfad

Nur bei **C++** Generierung:

Name	Typ	Beschreibung
$domtype	Integer	1 bei MSXML, 2 bei Xerces
$libtype	Integer	1 bei statischer LIB, 2 bei DLL
$mfc	Boolean	True, wenn MFC Unterstützung aktiviert ist
$VSVersion	Integer	Definiert die Visual Studio Version. Gültige Werte: **0** No Visual Studio project **2010** Visual Studio 2010 **2013** Visual Studio 2013 **2015** Visual Studio 2015 **2017** Visual Studio 2017 **2019** Visual Studio 2019

Nur bei **C#** Generierung:

Name	Typ	Beschreibung
$VSVersion	Integer	Definiert die Visual Studio Version. Gültige Werte: **0** No Visual Studio project **2010** Visual Studio 2010 **2013** Visual Studio 2013 **2015** Visual Studio 2015 **2017** Visual Studio 2017 **2019** Visual Studio 2019

© 2016-2022 Altova GmbH

16.12.5 Erstellen von Ausgabedateien

Diese Anweisung dienen zum Erstellen von Ausgabedateien mit Hilfe der Codegenerierung.
Beachten Sie, dass all diese Anweisungen sich innerhalb eines Blocks befinden müssen, der innerhalb von eckigen Klammern stehen muss.

create

create *filename*

Erstellt eine neue Datei. Die Datei muss mit der **close**-Anweisung geschlossen werden. Die gesamte darauf folgende Ausgabe wird in die angegebene Datei geschrieben.

Beispiel:

```
[create $outputpath & "/" & $JavaPackageDir & "/" & $application.Name & ".java"]
package [=$JavaPackageName];

public class [=$application.Name]Application {
...
}
[close]
```

close

Schließt die aktuelle Ausgabedatei.

=$*variable*

Schreibt den Wert der angegebenen Variable in die aktuelle Ausgabedatei.

Beispiel:

```
[$x = 20+3]
The result of your calculation is [=$x] - so have a nice day!
```

Die Dateiausgabe lautet:

```
The result of your calculation is 23 - so have a nice day!
```

write

write *string*

Schreibt den String in die aktuelle Ausgabedatei.

Beispiel:

```
[write "C" & $name]
```

Dies kann auch geschrieben werden als:

```
C[=$name]
```

filecopy ... to ...

filecopy *source* **to** *target*

Kopiert die Quelldatei ohne jede Interpretation in die Zieldatei.

Beispiel:

```
filecopy "java/mapforce/mapforce.png" to $outputpath & "/" & $JavaPackageDir &
"/mapforce.png"
```

16.12.6 Operatoren

Operatoren in SPL funktionieren wie in den meisten anderen Programmiersprachen.

Liste von SPL-Operatoren absteigend nach Vorrangigkeit gereiht:

.	Objekteigenschaft aufrufen
( )	Gruppierung eines Ausdrucks
true	Boolesche Konstante "true"
false	Boolesche Konstante "false"
&	String-Verkettung
-	Zeichen für negative Zahl
not	Logische Negation
*	Multiplizieren
/	Dividieren
%	Modulo
+	Addieren
-	Subtrahieren

© 2016-2022 Altova GmbH

<= Kleiner oder gleich
< Kleiner als
>= Größer oder gleich
> Größer als

= Gleich
<> Nicht gleich

and Logische Konjunktion (mit short circuit-Auswertung)
or Logische Disjunktion (mit short circuit-Auswertung)

= Zuweisung

16.12.7 Bedingungen

SPL erlaubt die Verwendung von Standard-"if"-Anweisungen. Die Syntax lautet wie folgt:

```
if condition
       statements
else
       statements
endif
```

oder ohne else:

```
if condition
       statements
endif
```

Beachten Sie bitte, dass für die Bedingung keine runden Klammern verwendet werden!

Wenn in allen anderen Programmiersprachen werden Bedingungen mit logischen und Vergleichsoperatoren[1710] konstruiert.

Beispiel:

```
[if $namespace.ContainsPublicClasses and $namespace.Prefix <> ""]
       whatever you want ['inserts whatever you want, in the resulting file]
[endif]
```

Switch

SPL enthält auch eine Multiple Choice-Anweisung.

Syntax:

```
switch $variable
```

```
       case X:
              statements
       case Y:
       case Z:
              statements
       default:
              statements
endswitch
```

Die Case-Bezeichnungen müssen Konstanten oder Variablen sein.

Die switch-Anweisung in SPL fällt nicht durch die Cases (wie in C), daher wird auch keine "break"-Anweisung benötigt.

16.12.8 Collections und foreach

Collections und Iteratoren

Eine Collection enthält mehrere Objekte - wie ein gewöhnliches Array. Iteratoren lösen das Problem des Speicherns und Inkrementierens von Array Indizes beim Aufrufen von Objekten.

Syntax:

```
foreach iterator in collection
       statements
next
```

Beispiel:

```
[foreach $class in $classes
       if not $class.IsInternal
              ]      class [=$class.Name];
[      endif
next]
```

Beispiel 2:

```
[foreach $i in 1 To 3
      Write "// Step " & $i & "\n"
      ‘ Do some work
next]
```

Die erste Zeile:
$classes ist das **globale Objekt**[1707] von allen Typen. $classes ist eine Collection (Sammlung) von einzelnen Klassenobjekten.

© 2016-2022 Altova GmbH

Foreach iteriert durch alle Datenelemente in $classes und führt für die Anweisung den Code der auf die Anweisung folgt, bis zur **nächsten** Anweisung aus.

In jeder Iteration wird **$class** dem nächsten Klassenobjekt zugewiesen. Sie arbeiten einfach mit dem Klassenobjekt, anstelle classes[i]->class->Name(), zu verwenden, wie das in C++ der Fall wäre.

Alle Collection-Iteratoren haben die folgenden zusätzlichen Eigenschaften:

Index	der aktuelle Index beginnend mit 0
IsFirst	true, wenn das aktuelle Objekt das erste der Collection ist (Index ist 0)
IsLast	true, wenn das aktuelle Objekt das letzte einer Collection ist
Current	das aktuelle Objekt (dies ist implizit, wenn es nicht definiert ist und kann weggelassen werden)

Beispiel:

```
[foreach $enum in $facet.Enumeration
       if not $enum.IsFirst
              ], [
       endif
       ]"[=$enum.Value]"[
next]
```

16.12.9 Subroutinen

Code Generator unterstützt nun Subroutinen in der Form von Prozeduren oder Funktionen.

Funktionen:

- Übergabe von Werten nach Wert und nach Referenz
- Lokale/globale Parameter (lokal innerhalb von Subroutinen)
- Lokale Variablen
- Rekursiver Aufruf (Subroutinen können sich selbst aufrufen)

16.12.9.1 Deklaration einer Subroutine

Subroutinen

Syntax-Beispiel:

```
Sub SimpleSub()
       ... lines of code
 EndSub
```

- **Sub** ist das Schlüsselwort, das die Prozedur notiert.
- **SimpleSub** ist der Name, der der Subroutine zugewiesen wurde.
- Runde **Klammern** können eine Parameterliste enthalten.
- Der Code-Block einer Subroutine beginnt direkt nach der geschlossenen Klammer nach dem Parameter.
- **EndSub** notiert das Ende des Code-Blocks.

Bitte beachten Sie:
Eine rekursive oder kaskadierende Subroutine-**Deklaration** ist nicht zulässig, d.h. eine Subroutine darf keine weiteren Subroutinen enthalten.

Parameter

Parameter können auch durch Prozeduren unter Verwendung der folgenden Syntax übergeben werden:

- Alle Parameter müssen Variablen sein
- Variablen muss das **$-**Zeichen vorangestellt werden
- Lokale Variablen werden in einer Subroutine definiert
- Globale Variablen werden explizit außerhalb von Subroutinen deklariert
- Mehrere Parameter werden innerhalb runder Klammern durch Kommas "**,**" voneinander getrennt
- Parameter können Werte übergeben

Parameter - Übergabe von Werten

Parameter können auf zwei Arten übergeben werden: nach Wert oder nach Referenz, wobei die Schlüsselwörter ByVal bzw. ByRef verwendet werden.

Syntax:

```
' define sub CompleteSub()
[Sub CompleteSub( $param, ByVal $paramByValue, ByRef $paramByRef )
] ...
```

- **ByVal** definiert, dass der Parameter nach Wert übergeben wird. Beachten Sie, dass die meisten Objekte nur nach Referenz übergeben werden können.
- **ByRef** definiert, dass der Parameter nach Referenz übergeben wird. Dies ist die Standardeinstellung, wenn weder ByVal noch ByRef definiert ist.

Funktionsrückgabewerte

Für die Rückgabe eines Werts von einer Subroutine verwenden Sie ein **Return-**Statement. Eine solche Funktion kann von innerhalb eines Ausdrucks aufgerufen werden.

Beispiel:

```
' define a function
[Sub MakeQualifiedName( ByVal $namespacePrefix, ByVal $localName )
if $namespacePrefix = ""
  return $localName
else
  return $namespacePrefix & ":" & $localName
```


© 2016-2022 Altova GmbH

```
endif
EndSub
]
```

16.12.9.2 Subroutinenaufruf

Verwenden Sie zum Aufrufen einer Subroutine **call**, gefolgt vom Namen der Prozedur und ggf. den Parametern.

```
Call SimpleSub()
```

oder

```
Call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
```

Funktionsaufruf

Um eine Funktion (jede Subroutine, die ein **Return**-Statement enthält) aufzurufen, verwenden Sie einfach deren Namen in einem Ausdruck. Verwenden Sie zum Aufrufen von Funktionen nicht das **Call**-Statement.
Beispiel:

```
$QName = MakeQualifiedName($namespace, "entry")
```

16.12.9.3 Beispiel für eine Subroutine

Das folgende Beispiel enthält die Deklaration und den Aufruf einer Subroutine.

```
[create $outputpath & $module & "output.txt"

' define sub SimpleSub()
Sub SimpleSub()
]SimpleSub() called
[endsub

' execute sub SimpleSub()
Call SimpleSub()

$ParamByValue       = "Original Value"
]ParamByValue       = [=$ParamByValue]
[$ParamByRef  = "Original Value"
]ParamByRef   = [=$ParamByRef]

' define sub CompleteSub()
[Sub CompleteSub( $param, ByVal $paramByValue, ByRef $paramByRef )
]CompleteSub called.
       param = [=$param]
       paramByValue = [=$paramByValue]
```

```
      paramByRef = [=$paramByRef]
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]      Set values inside Sub
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]CompleteSub finished.
[endsub

' run sub CompleteSub()
Call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
]
ParamByValue=[=$ParamByValue]
ParamByRef=[=$ParamByRef]
[
Close
]
```

16.12.10 Vordefinierte Typen

In diesem Abschnitt werden die in vordefinierten Variablen[1707] verwendeten Eigenschaften von integrierten Typen beschrieben, die das geparste Schema beschreiben.

16.12.10.1 Bibliothek

Dieses Objekt steht für die gesamte anhand des XML-Schemas oder der DTD generierte Bibliothek.

Eigenschaft	Typ	Beschreibung
SchemaNamespaces	Namespace[1716] collection	Namespaces in dieser Bibliothek
SchemaFilename	string	Name der XSD- oder DTD-Datei, von der diese Bibliothek abgeleitet wurde
SchemaType	integer	1 für DTD, 2 für XML-Schema
Guid	string	Eine globale eindeutige ID
CodeName	string	Generierter Bibliotheksname (vom Schemadateinamen abgeleitet)

16.12.10.2 Namespace

Pro XML-Schema Namespace wird ein Namespace-Objekt generiert. Schema-Komponenten, die sich in keinem Namespace befinden, werden in ein spezielles Namespace-Objekt mit einer leeren NamespaceURI gelegt.

© 2016-2022 Altova GmbH

Beachten Sie: Bei DTD werden Namespaces auch von Attributen abgeleitet, deren Namen mit "xmlns" beginnen.

Eigenschaft	Typ	Beschreibung
CodeName	String	Name für den generierten Code (abgeleitet vom Präfix)
LocalName	String	Namespace-Präfix
NamespaceURI	String	Namespace URI
Types	Type 1717 Collection	Alle in diesem Namespace enthaltenen Typen
Library	Library 1716	Bibliothek, die diesen Namespace enthält

16.12.10.3 Typ

Dieses Objekt stellt einen complex oder simpleType dar. Es dient zum Generieren einer Klasse in der Zielsprache.
Es gibt keinen zusätzlichen Typ pro Bibliothek, der für das Dokument steht, der alle möglichen Root-Elemente als Members hat.
Anonyme Typen haben einen leeren LocalName.

Eigenschaft	Typ	Beschreibung
CodeName	string	Name für generierten Code (wird vom lokalen Namen oder der Parent-Deklaration abgeleitet)
LocalName	string	Ursprünglicher Name im Schema
Namespace	Namespace 1716	Namespace, der diesen Typ enthält
Attributes	Member 1718 collection	Attribute, die in diesem Typ enthalten sind
Elements	Member 1718 collection	Child-Elemente, die in diesem Typ enthalten sind
IsSimpleType	boolean	True bei simpleTypes, False bei complexTypes
IsDerived	boolean	True, wenn dieser Typ von einem anderen Typ abgeleitet wurde, der ebenfalls durch ein Typobjekt dargestellt wird
IsDerivedByExtension	boolean	"true", wenn dieser Typ mittels extension abgeleitet wurde
IsDerivedByRestriction	boolean	"true", wenn dieser Typ mittels restriction abgeleitet wurde

IsDerivedByUnion	boolean	"true", wenn dieser Typ mittels union abgeleitet wurde
IsDerivedByList	boolean	"true", wenn dieser Typ mittels list abgeleitet wurde
BaseType	Type	Der Basistyp dieses Typs (wenn IsDerived "true" ist)
IsDocumentRootType	boolean	True, wenn dieser Typ für das Dokument selbst steht
Library	Library 1716	Bibliothek, die diesen Typ enthält
IsFinal	boolean	True, wenn als "final" im Schema deklariert
IsMixed	boolean	True, wenn dieser Typ Mixed Content haben kann
IsAbstract	boolean	True, wenn dieser Typ als abstrakt deklariert ist
IsGlobal	boolean	True, wenn dieser Typ im Schema global deklariert ist
IsAnonymous	boolean	True, wenn dieser Typ lokal in einem Element deklariert ist

Nur bei simpleTypes:

Eigenschaft	**Typ**	**Beschreibung**
IsNativeBound	boolean	True, wenn Native Type Binding vorhanden ist
NativeBinding	NativeBinding 1719	Native Binding für diesen Typ
Facets	Facets 1720	Facets dieses Typs
Whitespace	string	Shortcut zur Whitespace Facet

* ComplexTypes mit Textinhalt (dabei handelt es sich um Typen mit Mixed Content und complexType mit simpleContent) haben ein zusätzliches nicht benanntes Attribut-Member, das den Textinhalt repräsentiert.

16.12.10.4 Member

Dieses Objekt steht für ein Attribut oder Element im XML-Schema. Es dient zum Erstellen von Klassen-Members von Typen.

Eigenschaft	**Typ**	**Beschreibung**

© 2016-2022 Altova GmbH

CodeName	string	Name für generierten Code (vom lokalen Namen oder der Parent-Deklaration abgeleitet)
LocalName	string	Ursprünglicher Name im Schema. Leer bei dem speziellen Member, das den Textinhalt von complexTypes darstellt.
NamespaceURI	string	Die Namespace URI dieses Elements/Attributs in XML-Instanzdokumenten / -streams.
DeclaringType	Type 1717	Typ, der ursprünglich das Member deklariert (entspricht ContainingType für nicht vererbte Members)
ContainingType	Type 1717	Typ, dessen Member dies ist
DataType	Type 1717	Datentyp des Inhalts dieses Members
Library	Library 1716	Bibliothek, die den Datentyp dieses Members enthält
IsAttribute	boolean	True bei Attributen, False bei Elementen
IsOptional	boolean	True, wenn minOccurs = 0 oder optionales Attribut
IsRequired	boolean	True, wenn minOccurs > 0 oder required Attribut
IsFixed	boolean	True bei fixed Attributen, Wert ist in der Eigenschaft "Default"
IsDefault	boolean	True bei Attributen mit Standardwert, Wert ist in der Eigenschaft "Default"
IsNillable	boolean	True bei nillable Elementen
IsUseQualified	boolean	True, wenn NamespaceURI nicht leer ist
MinOccurs	integer	minOccurs wie in Schema. 1 bei required Attributen
MaxOccurs	integer	maxOccurs, wie in Schema. 0 bei prohibited Attributen, -1 bei unbounded
Default	string	Standardwert

16.12.10.5 NativeBinding

Dieses Objekt steht für das Binding eines simpleType an einen native Type in der Zielprogrammiersprache, wie in der "schemanativetype" Zuordnung angegeben.

Eigenschaft	**Typ**	**Beschreibung**

ValueType	string	Native Type
ValueHandler	string	Formatierungsklasseninstanz

16.12.10.6 Facets

Dieses Objekt steht für alle Facets eines simpleType.
Vererbte Facets werden mit den explizit deklarierten Facets zusammengeführt. Wenn ein Length Facet gilt, werden MinLength und MaxLength auf denselben Wert gesetzt.

Eigenschaft	Typ	Beschreibung
DeclaringType	Type	Type Facets werden darauf deklariert
Whitespace	string	"preserve", "collapse" oder "replace"
MinLength	integer	Facet-Wert
MaxLength	integer	Facet-Wert
MinInclusive	integer	Facet-Wert
MinExclusive	integer	Facet-Wert
MaxInclusive	integer	Facet-Wert
MaxExclusive	integer	Facet-Wert
TotalDigits	integer	Facet-Wert
FractionDigits	integer	Facet-Wert
List	Facet collection	Alle Facets als Liste

Facet

Dieses Objekt steht für eine einzelne Facet, deren berechneter Wert für einen bestimmten Typ wirksam wird.

Eigenschaft	Typ	Beschreibung
LocalName	string	Facet-Name
NamespaceURI	string	Facet Namespace
FacetType	string	Einer der folgenden: "normalization", "lexicalspace", "valuespace-length", "valuespace-enum" oder "valuespace-range"
DeclaringType	Type 1717	Typ, auf den dieses Facet deklariert ist

© 2016-2022 Altova GmbH

FacetCheckerName	string	Name des Facet Checker (aus der schemafacet-Zuordnung)
FacetValue	string or integer	Tatsächlicher Wert dieses Facet

17 Die MapForce API

Über die COM-basierte API von MapForce können Clients die Funktionalitäten von MapForce von benutzerdefiniertem Code oder einer benutzerdefinierten Applikation aus aufrufen. Auf diese Art kann nun ein breites Spektrum an Aufgaben automatisiert werden.

In der MapForce COM API finden die allgemeinen von Microsoft vorgegebenen Spezifikationen für Automation Server Anwendung. MapForce wird bei der Installation automatisch als COM-Server-Objekt registriert. Sobald das COM-Server-Objekt registriert wurde, können Sie es von Applikationen und Skriptsprachen mit Programmierunterstützung für COM-Aufrufe aufrufen. Dadurch haben Sie nicht nur von Entwicklungsumgebungen, in denen .NET, C++ und Visual Basic verwendet wird, sondern auch von Skriptsprachen wie JScript und VBScript Zugriff auf die MapForce API.

Beachten Sie die folgenden Punkte:

- Wenn Sie mit Hilfe der MapForce API eine Applikation erstellen, die mit anderen Clients verteilt werden soll, muss MapForce auf jedem Client-Rechner installiert sein. Auch Ihr benutzerdefinierter Integrationscode (bzw. Ihr Applikation) muss auf den einzelnen Client-Rechnern bereitgestellt bzw. installiert werden.
- Bei bestimmten API-Methoden wie `Document.GenerateOutput` muss das Hauptfenster von MapForce sichtbar sein oder MapForce muss (bei Ausführung als COM-Server) in eine grafische Benutzeroberflächer eingebettet sein. Wenn Mappings komplett ohne Benutzerinteraktion plattformunabhängig ausgeführt werden soll, empfiehlt sich die Verwendung von MapForce Server (https://www.altova.com/de/mapforce-server).

© 2016-2022 Altova GmbH

17.1 Aufruf der API

Um die MapForce COM API aufrufen zu können, muss in Ihrer Applikation (oder Ihrem Skript) eine neue Instanz des `Application`-Objekts erstellt werden. Anschließend haben Sie über dieses Objekt durch Aufruf der benötigten Methoden und Eigenschaften (z.B. Erstellen eines neuen Dokuments, Öffnen eines vorhandenen Dokuments, Generieren von Mapping-Code usw.) Zugriff auf MapForce.

Voraussetzungen

Um das MapForce COM-Objekt in Ihrem Visual Studio-Projekt verfügbar zu machen, fügen Sie eine Referenz zur MapForce Typbibliotheksdatei (.tlb) hinzu. Die folgende Anleitung gilt für 2013, ist aber auch bei anderen Visual Studio-Versionen ähnlich:

1. Klicken Sie im Menü **Projekt** auf **Verweis hinzufügen.**
2. Klicken Sie auf **Durchsuchen** und wählen Sie die Datei **MapForce.tlb** im MapForce-Installationsordner aus.

Unter **C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForceExamples\API\C#** steht ein MapForce API-Beispiel-Client in C# zur Verfügung.

In Java steht die MapForce API über Java-COM Bridge-Bibliotheken zur Verfügung. Sie finden diese Bibliotheken im MapForce Installationsordner: **C:\Programme (x86)\Altova\MapForce2022\JavaAPI** (Beachten Sie, dass dieser Pfad gilt, wenn ein 32-Bit-MapForce auf 64-Bit-Windows ausgeführt wird. Passen Sie den Pfad andernfalls entsprechend an).

- `AltovaAutomation.dll`: ein JNI Wrapper für Altova Automation Server
- `AltovaAutomation.jar`: Java-Klassen für den Zugriff auf Altova Automation Server
- `MapForceAPI.jar`: Java-Klassen, die den Wrap für die MapForce Automatisch-Schnittstelle bilden
- `MapForceAPI_JavaDoc.zip`: eine `Javadoc`-Datei mit der Hilfe zur Java API

Um direkt vom Java-Code aus Zugriff auf den MapForce Automation Server zu erhalten, müssen sich die Bibliotheken im Java `classpath` befinden.

Unter **C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForceExamples\API\Java** steht ein MapForce API-Beispiel-Client in Java zur Verfügung.

In Skriptsprachen wie JScript oder VBScript kann das MapForce COM-Objekt über den Microsoft Windows Script Host aufgerufen werden (siehe https://msdn.microsoft.com/en-us/library/9bbdkx3k.aspx). Solche Skripts können mit einem Text-Editor geschrieben werden und müssen nicht kompiliert werden, da sie von dem mit Windows verpackten Windows Script Host ausgeführt werden. (Um zu überprüfen, ob der Windows Script Host ausgeführt wird, geben Sie in der Befehlszeile `wscript.exe /?` ein). Unter **C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForceExamples\API\JScript** steht ein MapForce API-Beispiel-Client in JScript zur Verfügung.

Anmerkung: Für die 32-Bit-Version von MapForce ist der registrierte Name oder der programmatische Identifier (ProgId) des COM-Objekts `MapForce.Application`. Für die 64-Bit-Version von MapForce ist der Name `MapForce_x64.Application`. Beachten Sie jedoch, dass das aufrufende Programm die CLASSES Registry-Einträge in seiner eigenen Registry Hive oder -Gruppe (32-Bit oder 64-Bit) aufruft. Wenn Sie daher Skripts über die Standardbefehlszeileneingabe und mit Windows Explorer auf einem 64-Bit-Windows-System ausführen, werden die 64-Bit-Registry-Einträge, welche auf die 64-Bit-Version von MapForce verweisen, aufgerufen. Wenn daher sowohl MapForce 32-Bit als auch die 64-Bit-Version

installiert ist, ist eine spezielle Behandlung erforderlich, damit die 32-Bit-Version von MapForce aufgerufen wird. Angenommen, der Windows Skripting Host ist das aufrufende Programm, so gehen Sie folgendermaßen vor:

1. Wechseln Sie in das Verzeichnis **C:\Windows\SysWOW64**.
2. Geben Sie in der Befehlszeile **wscript.exe** gefolgt vom Pfad zum gewünschten Skript ein, z.B:

```
wscript.exe "C:\Users\...
\Documents\Altova\MapForce2023\MapForceExamples\API\JScript\start.js"
```

Richtlinien

Es sollten in Ihrem Client Code die folgenden Richtlinien beachtet werden:

- Behalten Sie Referenzen auf Objekte nicht länger im Arbeitsspeicher, als notwendig. Wenn ein Benutzer zwischen zwei Calls Ihres Client eine Eingabe macht, besteht keine Garantie, dass diese Referenzen noch gültig sind.
- Denken Sie daran, dass bei einem Absturz Ihres Client Code Instanzen von MapForce möglicherweise noch im System verbleiben.
- Nähere Informationen, wie man lästige Fehlermeldungen vermeidet, finden Sie unter Behandlung von Fehlern 1727.
- Geben Sie Referenzen explizit frei, wenn Sie Sprachen wie C++ verwenden.

Erstellen des Application-Objekts

Die Syntax zur Erstellung des `Application`-Objekts hängt von der Programmiersprache ab, wie in den Beispielen unten gezeigt:

C#

```
// Create a new instance of MapForce via its automation interface.
MapForceLib.Application objMapForce = new MapForceLib.Application();
```

Java

```
// Start MapForce as COM server.
com.altova.automation.MapForce.Application objMapForce = new Application();
// COM servers start up invisible so we make it visible
objMapForce.setVisible(true);
```


© 2016-2022 Altova GmbH

JScript

```
// Access a running instance, or create a new instance of MapForce.
try
{
	objMapForce = WScript.GetObject ("", "MapForce.Application");
	// unhide application if it is a new instance
	objMapForce.Visible = true;
}
catch(err) { WScript.Echo ("Can't access or create MapForce.Application"); }
```

VBA

```
' Create a new instance of MapForce.
Dim objMapForce As Application
Set objMapForce = CreateObject("MapForce.Application")
```

VBScript

```
' Access a running instance, or create a new instance of MapForce.
Set objMapForce = GetObject("MapForce.Application");
```

Visual Basic

```
Dim objMapForce As MapForceLib.Application = New MapForceLib.Application
```

17.2 Das Objektmodell

Der Ausgangspunkt für jede Applikation, die die MapForce API verwendet, ist das `Application`[1752] Objekt. Alle anderen Schnittstellen werden über andere Objekte aufgerufen, wobei das `Application` Objekt den Ausgangspunkt bildet.

Das Application-Objekt besteht aus den folgenden Teilen (jede Einrückung zeigt an, das es sich dabei um eine Child-Parent-Beziehung zur Ebene unmittelbar oberhalb davon handelt):

```
Application[1752]
   Options[1870]
   Project[1879]
      ProjectItem[1896]
   Documents[1829]
      Document[1809]
         MapForceView[1849]
         Mapping[1857]
            Component[1787]
               Datapoint[1806]
            Components[1801]
            Connection[1804]
         Mappings[1867]
   ErrorMarkers[1838]
      ErrorMarker[1833]
AppOutputLines[1781]
   AppOutputLine[1770]
      AppOutputLines
         ...
      AppOutputLineSymbol[1784]
```

Informationen zur Erstellung einer Instanz des `Application`-Objekts finden Sie unter Aufruf der API[1723]. Informationen zu den von der API bereitgestellten Objekten finden Sie in der Objektreferenz[1752].

© 2016-2022 Altova GmbH

17.3 Behandlung von Fehlern

Die MapForce API gibt Fehler auf zwei verschiedene Arten zurück. Jede API-Methode gibt ein `HRESULT` zurück. Dieser Rückgabewert informiert den Caller über etwaige Fehlfunktionen während der Ausführung dieser Methode. Wenn der Call erfolgreich ausgeführt wurde, ist der Rückgabewert gleich `S_OK`. C/C++-Programmierer verwenden im Allgemeinen `HRESULT` zum Ausfindingmachen von Fehlern.

Visual Basic, Script-Sprachen und andere komplexe Entwicklungsumgebungen geben dem Programmierer keinen Zugriff auf das zurückgegebene `HRESULT` eines COM Call. Sie verwenden den zweiten Mechanismus zur Auslösung von Fehlern, der von der MapForce API unterstützt wird, die `IErrorInfo` Schnittstelle. Wenn ein Fehler auftritt, erstellt die API ein neues Objekt, das die `IErrorInfo` Schnittstelle implementiert. Die Entwicklungsumgebung nimmt diese Schnittstelle und befüllt ihren Fehlerbehandlungsmechanismus mit den erhaltenen Informationen.

Im folgenden Text wird beschrieben, wie man von der MapForce API ausgelöste Fehler in verschiedenen Entwicklungsumgebungen behandelt.

VisualBasic

Eine häufige Methode zur Fehlerbehandlung in Visual Basic ist, einen Fehler-Handler zu definieren. Dieser Fehler-Handler kann mit der `On Error` Anweisung definiert werden. Normalerweise zeigt der Fehler-Handler eine Fehlermeldung an und bereinigt den Code um überflüssige Referenzen und alle Arten von Resource Leaks zu vermeiden. Visual Basic befüllt sein eigenes `Err` Objekt mit den Informationen aus der `IErrorInfo` Schnittstelle.

```
Sub Validate()
  'place variable declarations here

  'set error handler
  On Error GoTo ErrorHandler

  'if generation fails, program execution continues at ErrorHandler:
  objMapForce.ActiveDocument.GenerateXSLT()

  'additional code comes here

  'exit
  Exit Sub

  ErrorHandler:
  MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) &
     "Description: " & Err.Description)
End Sub
```

JavaScript

Die Microsoft Implementierung von JavaScript (JScript) bietet einen try-catch Mechanismus zur Behandlung von Fehlern, die von COM Calls ausgegeben werden. Er ähnelt der Methode von VisualBasic insofern, als ein Fehlerobjekt deklariert wird, das die nötigen Informationen enthält.

```
function Generate() {
    // please insert variable declarations here

    try {
        objMapForce.ActiveDocument.GenerateXSLT();
    }
    catch (Error) {
        sError = Error.description;
        nErrorCode = Error.number & 0xffff;
        return false;
    }

    return true;
}
```

C/C++

C/C++ gibt Ihnen einfachen Zugriff auf das `HRESULT` des COM Call und das `IErrorInterface`.

```
   HRESULT hr;

   // Call GenerateXSLT() from the MapForce API
   if(FAILED(hr = ipDocument->GenerateXSLT()))
   {
      IErrorInfo *ipErrorInfo = Null;

      if(SUCCEEDED(::GetErrorInfo(0, &ipErrorInfo)))
      {
         BSTR   bstrDescr;
         ipErrorInfo->GetDescription(&bstrDescr);

         // handle Error information
         wprintf(L"Error message:\t%s\n",bstrDescr);
         ::SysFreeString(bstrDescr);

         // release Error info
         ipErrorInfo->Release();
      }
   }
```


© 2016-2022 Altova GmbH

17.4 C#-Beispielprojekt

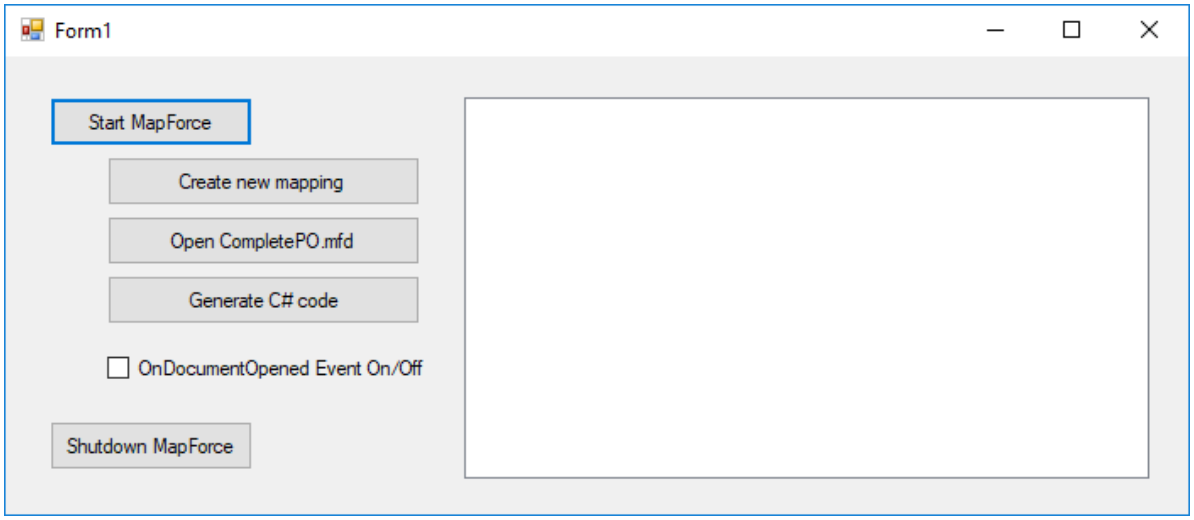
Nach Installation von MapForce steht ein MapForce API Client-Projekt für C# im folgenden Verzeichnis zur Verfügung: **C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForceExamples\API**.

Um das Beispiel zu kompilieren und auszuführen, öffnen Sie die Projektmappendatei (.sln) in Visual Studio und führen Sie den Befehl **Debugging | Debugging starten** oder **Debugging | Starten ohne Debugging** aus.

Anmerkung: Wenn Sie ein 64-Bit-Betriebssystem haben und eine 32-Bit-Version von MapForce verwenden, fügen Sie die **x86**-Plattform im Konfigurations-Manager der Projektmappe hinzu und erstellen Sie das Beispiel mittels "Build" mit dieser Konfiguration. Im Dialogfeld "Neue Projektmappenplattform" (**Build | Konfigurations-Manager | Aktive Projektmappenplattform | <Neu...>**) kann eine neue x86-Plattform (für die aktive Projektmappe in Visual Studio) erstellt werden.

Wenn Sie das Beispiel ausführen, wird ein Windows-Formular mit Schaltflächen zum Aufrufen der grundlegenden MapForce-Operationen angezeigt:

- Starten von MapForce
- Erstellen eines neuen Mapping-Designs
- Öffnen der Datei CompletePO.mfd aus dem Ordner **...\MapForceExamples** (möglicherweise müssen Sie den Pfad für den Ordner **\MapForceExamples** auf Ihrem Rechner anpassen)
- Generieren von C#-Code in einem Temp-Verzeichnis
- Beenden von MapForce



Codefragment

Das Codefragment enthält zum besseren Verständnis Kommentare. Vom Aufbau her besteht der Code aus einer Reihe von Handlern für die Schaltflächen auf der oben gezeigten Benutzeroberfläche.

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        // An instance of MapForce accessed via its automation interface.
        MapForceLib.Application MapForce;

        // Location of examples installed with MapForce
        String strExamplesFolder;

        private void Form1_Load(object sender, EventArgs e)
        {
        }

        // handler for the "Start MapForce" button
        private void StartMapForce_Click(object sender, EventArgs e)
        {
            if (MapForce == null)
            {
                Cursor.Current = Cursors.WaitCursor;

                // if we have no MapForce instance, we create one a nd make it visible.
                MapForce = new MapForceLib.Application();
                MapForce.Visible = true;

                // locate examples installed with MapForce.
                int majorVersionYear = MapForce.MajorVersion + 1998;
                strExamplesFolder = Environment.GetEnvironmentVariable("USERPROFILE") +
"\\My Documents\\Altova\\MapForce" + Convert.ToString(majorVersionYear) + "\
\MapForceExamples\\";

                Cursor.Current = Cursors.Default;
            }
            else
            {
                // if we have already an MapForce instance running we toggle its
visibility flag.
                MapForce.Visible = !MapForce.Visible;
            }
        }

        // handler for the "Open CompletePO.mfd" button
```


© 2016-2022 Altova GmbH

```
        private void openCompletePO_Click(object sender, EventArgs e)
        {
            if (MapForce == null)
                StartMapForce_Click(null, null);

            // Open one of the sample files installed with the product.
            MapForce.OpenDocument(strExamplesFolder + "CompletePO.mfd");
        }

        // handler for the "Create new mapping" button
        private void newMapping_Click(object sender, EventArgs e)
        {
            if (MapForce == null)
                StartMapForce_Click(null, null);

            // Create a new mapping
            MapForce.NewMapping();
        }

        // handler for the "Shutdown MapForce" button
        // shut-down application instance by explicitly releasing the COM object.
        private void shutdownMapForce_Click(object sender, EventArgs e)
        {
            if (MapForce != null)
            {
                // allow shut-down of MapForce by releasing UI
                MapForce.Visible = false;

                // explicitly release COM object
                try
                {
                    while
(System.Runtime.InteropServices.Marshal.ReleaseComObject(MapForce) > 0) ;
                }
                finally
                {
                    // avoid later access to this object.
                    MapForce = null;
                }
            }
        }

        // handler for button "Generate C# Code"
        private void generateCppCode_Click(object sender, EventArgs e)
        {
            if (MapForce == null)
                listBoxMessages.Items.Add("start MapForce first.");
            // COM errors get returned to C# as exceptions. We use a try/catch block to
handle them.
            try
            {
                MapForceLib.Document doc = MapForce.ActiveDocument;
```

```
                listBoxMessages.Items.Add("Active document " + doc.Name);
                doc.GenerateCHashCode();

            }
            catch (Exception ex)
            {
                // The COM call was not successful.
                // Probably no application instance has been started or no document is
open.
                MessageBox.Show("COM error: " + ex.Message);
            }
        }

        delegate void addListBoxItem_delegate(string sText);
        // called from the UI thread
        private void addListBoxItem(string sText)
        {
            listBoxMessages.Items.Add(sText);
        }
        // wrapper method to allow to call UI controls methods from a worker thread
        void syncWithUIthread(Control ctrl, addListBoxItem_delegate methodToInvoke,
String sText)
        {
            // Control.Invoke: Executes on the UI thread, but calling thread waits for
completion before continuing.
            // Control.BeginInvoke: Executes on the UI thread, and calling thread doesn't
wait for completion.
            if (ctrl.InvokeRequired)
                ctrl.BeginInvoke(methodToInvoke, new Object[] { sText });
        }

        // event handler for OnDocumentOpened event
        private void handleOnDocumentOpened(MapForceLib.Document i_ipDocument)
        {
            String sText = "";

            if (i_ipDocument.Name.Length > 0)
                sText = "Document " + i_ipDocument.Name + " was opened!";
            else
                sText = "A new mapping was created.";

            // we need to synchronize the calling thread with the UI thread because
            // the COM events are triggered from a working thread
            addListBoxItem_delegate methodToInvoke = new
addListBoxItem_delegate(addListBoxItem);
            // call syncWithUIthread with the following arguments:
            // 1 - listBoxMessages - list box control to display messages from COM events
            // 2 - methodToInvoke  - a C# delegate which points to the method which will
be called from the UI thread
            // 3 - sText           - the text to be displayed in the list box
            syncWithUIthread(listBoxMessages, methodToInvoke, sText);
        }
```


© 2016-2022 Altova GmbH

```
        private void checkBoxEventOnOff_CheckedChanged(object sender, EventArgs e)
        {
            if (MapForce != null)
            {
                if (checkBoxEventOnOff.Checked)
                    MapForce.OnDocumentOpened += new
MapForceLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
                else
                    MapForce.OnDocumentOpened -= new
MapForceLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
            }
        }
    }
}
```

17.5 Java-Beispielprojekt

Nach Installation von MapForce steht ein MapForce API Client-Projekt für Java im folgenden Verzeichnis zur Verfügung: **C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForceExamples\API**.

Sie können das Java-Beispielprojekt mit Hilfe der Batch-Datei `BuildAndRun.bat` direkt über die Befehlszeile testen oder Sie können es in Eclipse kompilieren und ausführen. Anleitungen dafür finden Sie weiter unten.

Dateiliste

Der Ordner für die Java-Beispiele enthält alle zum Ausführen des Beispielprojekts erforderlichen Dateien. Diese Dateien sind unten aufgelistet:

`AltovaAutomation.dll`	Java-COM Bridge: DLL-Teil
`AltovaAutomation.jar`	Java-COM Bridge: Java-Bibliotheksteil
`XMLSpyAPI.jar`	Java-Klassen der MapForce API
`RunXMLSpy.java`	Java-Beispielquellcode
`BuildAndRun.bat`	Batch-Datei zum Kompilieren und Ausführen des Beispielcodes über die Befehlszeile. Es wird ein Ordner benötigt, in dem sich die Java Virtual Machine als Parameter befindet.
`.classpath`	Hilfedatei Eclipse-Projekt
`.project`	Eclipse-Projektdatei
`MapForceAPI_JavaDoc.zip`	`Javadoc` Datei, die die Hilfedokumentation für die Java API enthält

Funktionen in diesem Beispiel

In diesem Beispielprojekt wird MapForce gestartet und einige Operationen wie das Öffnen und Schließen von Dokumenten werden ausgeführt. MapForce bleibt danach geöffnet. Sie müssen die Applikation manuell schließen.

Ausführen des Beispiels über die Befehlszeile

Um das Beispiel von der Befehlszeile aus auszuführen, öffnen Sie ein Eingabeaufforderungsfester, gehen Sie zum Ordner Java des Ordners API Examples (*Pfad siehe oben*) und geben Sie folgende Zeile ein:

```
buildAndRun.bat "<Path-to-the-Java-bin-folder>"
```

Der Java Binary-Ordner muss von einer JDK 1.5 oder höheren Version auf Ihrem Rechner sein.

Drücken Sie die **Eingabetaste.** Der Java-Quellcode in `RunMapForce.java` wird kompiliert und anschließend ausgeführt.

Laden des Beispiels in Eclipse

Öffnen Sie Eclipse und wählen Sie den Befehl **Import | Existing Projects into Workspace** um die Eclipse-Projektdatei (`.project`) im Ordner Java des Ordners API Examples (*Pfad siehe oben*) zu Eclipse hinzuzufügen. Daraufhin wird das Projekt `RunMapForce` in Ihrem Package Explorer oder Navigator angezeigt.

© 2016-2022 Altova GmbH

Wählen Sie das Projekt aus und klicken Sie anschließend auf **Run as | Java Application** um das Beispiel auszuführen.

Anmerkung: Sie können einen Klassennamen oder eine Methode der Java API auswählen und F1 drücken, um Hilfe zu dieser Klasse oder Methode zu erhalten.

Java-Quellcode

Im Folgenden finden Sie den mit Kommentaren versehenen Java-Quellcode aus der Beispieldatei `RunMapForce.java`.

```
// access general JAVA-COM bridge classes
import java.util.Iterator;

import com.altova.automation.libs.*;

// access XMLSpy Java-COM bridge
import com.altova.automation.MapForce.*;
import com.altova.automation.MapForce.Enums.ENUMProgrammingLanguage;

/**
 * A simple example that starts the COM server and performs a few operations on it.
 * Feel free to extend.
 */
public class RunMapForce
{
   public static void main(String[] args)
   {
      // an instance of the application.
      Application mapforce = null;

      // instead of COM error handling use Java exception mechanism.
      try
      {
         // Start MapForce as COM server.
         mapforce = new Application();
         // COM servers start up invisible so we make it visible
         mapforce.setVisible(true);

         // The following lines attach to the application events using a default
implementation
         // for the events and override one of its methods.
         // If you want to override all document events it is better to derive your
listener class
         // from DocumentEvents and implement all methods of this interface.
         mapforce.addListener(new ApplicationEventsDefaultHandler()
         {
            @Override
            public void onDocumentOpened(Document i_ipDoc) throws AutomationException
            {
               String name = i_ipDoc.getName();

               if (name.length() > 0)
```

```
                System.out.println("Document " + name + " was opened.");
             else
                System.out.println("A new mapping was created.");
          }
       });

       // Locate samples installed with the product.
       int majorVersionYear = mapforce.getMajorVersion() + 1998;
       String strExamplesFolder = System.getenv("USERPROFILE") + "\\Documents\\Altova\
\MapForce" + Integer.toString(majorVersionYear) + "\\MapForceExamples\\";
       // create a new MapForce mapping and generate c++ code
       Document newDoc = mapforce.newMapping();
       ErrorMarkers err1 = newDoc.generateCodeEx(ENUMProgrammingLanguage.eCpp);
       display(err1);
       // open CompletePO.mfd and generate c++ code
       Document doc = mapforce.openDocument(strExamplesFolder + "CompletePO.mfd");
       ErrorMarkers err2 = doc.generateCodeEx(ENUMProgrammingLanguage.eCpp);
       display(err2);

       doc.close();
       doc = null;

       System.out.println("Watch MapForce!");
    }
    catch (AutomationException e)
    {
       // e.printStackTrace();
    }
    finally
    {
       // Make sure that MapForce can shut down properly.
       if (mapforce != null)
          mapforce.dispose();

       // Since the COM server was made visible and still is visible, it will keep
running
       // and needs to be closed manually.
       System.out.println("Now close MapForce!");
    }
 }

 public static void display(ErrorMarkers err) throws AutomationException
 {
    Iterator<ErrorMarker> itr = err.iterator();

    if (err.getCount() == 0)
        System.out.print("Code generation completed successfully.\n");

    while (itr.hasNext())
    {
       String sError = "";
        Object element = itr.next();
        if (element instanceof ErrorMarker)
           sError = ((ErrorMarker)element).getText();
```


© 2016-2022 Altova GmbH

```
            System.out.print("Error text: " + sError + "\n");
        }
    }
}
```

17.6 JScript-Beispiele

Nach Installation von MapForce steht eine Reihe von JScript-Beispieldateien im Verzeichnis **C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForceExamples\API** zur Verfügung.

Die Beispieldateien können auf zwei Arten ausgeführt werden:

- *Über die Befehlszeile:* Öffnen Sie ein Eingabeaufforderungsfenster und geben Sie den Namen eines der Beispiel-Skripts ein (z.B. `Start.js`). Der mit Windows mitgelieferte Windows Scripting Host führt das Skript aus.
- *Über den Windows Explorer:* Navigieren Sie im Windows Explorer zur JScript-Datei und doppelklicken Sie darauf. Der mit Windows mitgelieferte Windows Scripting Host führt das Skript aus. Die Befehlskonsole wird nach Ausführung des Skript automatisch geschlossen.

Die folgenden Beispieldateien sind inkludiert:

`Start.js`	Startet das als Automation Server registrierte MapForce oder stellt eine Verbindung zu einer laufenden Instanz her (siehe Applikation starten[1738]).
`DocumentAccess.js`	Hier wird gezeigt, wie man Dokumente öffnet, durch diese iteriert und sie schließt (siehe Einfacher Dokumentaufruf[1739])..
`GenerateCode.js`	Hier wird gezeigt, wie man die Codegenerierung über JScript aufruft (siehe Code generieren[1740]).
`Readme.txt`	Enthält eine Hilfe zur Ausführung der Skripts.

Außerdem enthält diese Dokumentation einige zusätzliche JScript-Codebeispiele:

- Beispiel: Codegenerierung[1742]
- Beispiel: Mapping-Ausführung[1744]
- Beispiel: Projektunterstützung[1747]

17.6.1 Applikation starten

Mit dem unten aufgeführten JScript-Code wird die Applikation gestartet und beendet. Wenn bereits eine Instanz der Applikation ausgeführt wird, so wird diese Instanz aufgerufen. Um das Skript auszuführen, starten Sie es von der Befehlszeileneingabe oder von Windows Explorer aus, siehe auch Aufruf der API[1723].

```
// Initialize application's COM object. This will start a new instance of the application
and
// return its main COM object. Depending on COM settings, a the main COM object of an
already
// running application might be returned.

try {   objMapForce = WScript.GetObject("", "MapForce.Application");   }
```


© 2016-2022 Altova GmbH

```
catch(err) {}

if( typeof( objMapForce ) == "undefined" )
{
   try   {   objMapForce = WScript.GetObject("", "MapForce_x64.Application")   }
   catch(err)
   {
      WScript.Echo( "Can't access or create MapForce.Application" );
      WScript.Quit();
   }
}

// if newly started, the application will start without its UI visible. Set it to
visible.
objMapForce.Visible = true;

WScript.Echo(objMapForce.Edition + " has successfully started. ");

objMapForce.Visible = false; // will shutdown application if it has no more COM
connections
//objMapForce.Visible = true;   // will keep application running with UI visible
```

17.6.2 Einfacher Dokumentaufruf

Im unten aufgelisteten JScript-Code wird gezeigt, wie man Dokumente öffnet, ein Dokument zum aktiven Dokument macht, durch die offenen Dokumente iteriert und Dokumente schließt.

```
// Initialize application's COM object. This will start a new instance of the application
and
// return its main COM object. Depending on COM settings, a the main COM object of an
already
// running application might be returned.
try {   objMapForce = WScript.GetObject("", "MapForce.Application");   }
catch(err) {}

if( typeof( objMapForce ) == "undefined" )
{
   try   {   objMapForce = WScript.GetObject("", "MapForce_x64.Application")   }
   catch(err)
   {
      WScript.Echo( "Can't access or create MapForce.Application" );
      WScript.Quit();
   }
}

// if newly started, the application will start without its UI visible. Set it to
visible.
objMapForce.Visible = true;

// ************************** code snippet for "Simple Document Access"
```

```
**********************

// Locate examples via USERPROFILE shell variable. The path needs to be adapted to major
release versions.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objMapForce.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\Documents\
\Altova\\MapForce" + majorVersionYear + "\\MapForceExamples\\";

objMapForce.Documents.OpenDocument(strExampleFolder + "CompletePO.mfd");
objMapForce.Documents.OpenDocument(strExampleFolder + "Altova_Hierarchical_DB.mfd");

// *************************** code snippet for "Simple Document Access"
**********************

// *************************** code snippet for "Iteration"
***********************************

// go through all open documents using a JScript Enumerator
for (var iterDocs = new Enumerator(objMapForce.Documents); !iterDocs.atEnd();
iterDocs.moveNext())
{
   objName = iterDocs.item().Name;
   WScript.Echo("Document name: " + objName);
}

// go through all open documents using index-based access to the document collection
for (i = objMapForce.Documents.Count; i > 0; i--)
   objMapForce.Documents.Item(i).Close();


// *************************** code snippet for "Iteration"
***********************************

//objMapForce.Visible = false;      // will shutdown application if it has no more COM
connections
objMapForce.Visible = true;   // will keep application running with UI visible
```

Der oben angeführte Code steht in Form einer Beispieldatei zur Verfügung (siehe JScript-Beispiele[1738]). Um das Skript auszuführen, starten Sie es über ein Eingabeaufforderungsfenster oder über den Windows Explorer.

17.6.3 Code generieren

Im unten stehenden JScript-Code wird gezeigt, wie man Dokumente öffnet, ein Dokument zum aktiven macht, durch die offenen Dokumente iteriert und C++-Code generiert.

```
// Initialize application's COM object. This will start a new instance of the application
and
// return its main COM object. Depending on COM settings, a the main COM object of an
already
// running application might be returned.
```


© 2016-2022 Altova GmbH

```
try {   objMapForce = WScript.GetObject("", "MapForce.Application");   }
catch(err) {}

if( typeof( objMapForce ) == "undefined" )
{
   try   {   objMapForce = WScript.GetObject("", "MapForce_x64.Application")   }
   catch(err)
   {
      WScript.Echo( "Can't access or create MapForce.Application" );
      WScript.Quit();
   }
}

// if newly started, the application will start without its UI visible. Set it to
visible.
objMapForce.Visible = true;

// ************************** code snippet for "Simple Document Access"
***********************

// Locate examples via USERPROFILE shell variable. The path needs to be adapted to major
release versions.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objMapForce.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\Documents\
\Altova\\MapForce" + majorVersionYear + "\\MapForceExamples\\";

objMapForce.Documents.OpenDocument(strExampleFolder + "CompletePO.mfd");
//objMapForce.Documents.OpenDocument(strExampleFolder + "Altova_Hierarchical_DB.mfd");
objMapForce.Documents.NewDocument();

// ************************** code snippet for "Simple Document Access"
***********************

// ************************** code snippet for "Iteration"
***********************************

objText = "";
// go through all open documents using a JScript Enumerator and generate c++ code
for (var iterDocs = new Enumerator(objMapForce.Documents); !iterDocs.atEnd();
iterDocs.moveNext())
{
   objText += "Generated c++ code result for document " + iterDocs.item().Name + " :\n";
   objErrorMarkers = iterDocs.item().generateCodeEx(1); // ENUMProgrammingLanguage.eCpp =
1

   bSuccess = true;
   for (var iterErrorMarkers = new
Enumerator(objErrorMarkers); !iterErrorMarkers.atEnd(); iterErrorMarkers.moveNext())
   {
      bSuccess = false;
      objText += "\t" + iterErrorMarkers.item().Text + "\n";
   }
```

```
   if (bSuccess)
       objText += "\tCode generation completed successfully.\n";

   objText += "\n";
}

WScript.Echo(objText);

// go through all open documents using index-based access to the document collection
for (i = objMapForce.Documents.Count; i > 0; i--)
   objMapForce.Documents.Item(i).Close();


// *************************** code snippet for "Iteration"
**********************************

//objMapForce.Visible = false;      // will shutdown application if it has no more COM
connections
objMapForce.Visible = true;   // will keep application running with UI visible
```

Der oben angeführte Code steht in Form einer Beispieldatei zur Verfügung (siehe JScript-Beispiele 1738). Um das Skript auszuführen, starten Sie es über ein Eingabeaufforderungsfenster oder über den Windows Explorer.

17.6.4 Codegenerierung (alternative Methode)

Im folgenden JScript-Beispiel wird gezeigt, wie man ein vorhandenes Dokument lädt und verschiedene Arten von Mapping-Code dafür generiert.

```
// ------------------- begin JScript example ---------------------
// Generate Code for existing mapping.
// works with Windows scripting host.

// ----------------- helper function ------------------
function Exit(strErrorText)
{
   WScript.Echo(strErrorText);
   WScript.Quit(-1);
}

function ERROR(strText, objErr)
{
   if (objErr != null)
      Exit ("ERROR: (" + (objErr.number & 0xffff) + ")" + objErr.description + " - " +
strText);
   else
      Exit ("ERROR: " + strText);
}
// -----------------------------------------------------

// ----------------- MAIN ------------------

// ----- create the Shell and FileSystemObject of the windows scripting
```


© 2016-2022 Altova GmbH

```
try
{
   objWshShell = WScript.CreateObject("WScript.Shell");
   objFSO = WScript.CreateObject("Scripting.FileSystemObject");
}
catch(err)
   { Exit("Can't create WScript.Shell object"); }

// ----- open MapForce or access running instance and make it visible
try
{
   objMapForce = WScript.GetObject ("", "MapForce.Application");
   objMapForce.Visible = true;      // remove this line to perform background processing
}
catch(err) { WScript.Echo ("Can't access or create MapForce.Application"); }

// ----- open an existing mapping. adapt this to your needs!
objMapForce.OpenDocument(objFSO.GetAbsolutePathName ("Test.mfd"));

// ----- access the mapping to have access to the code generation methods
var objDoc = objMapForce.ActiveDocument;

// ----- set the code generation output properties and call the code generation methods.
// ----- adapt the output directories to your needs
try
{
   // ----- code generation uses some of these options
   var objOptions = objMapForce.Options;

   // ----- generate XSLT -----
   objOptions.XSLTDefaultOutputDirectory = "C:\\test\\TestCOMServer\\XSLT";
   objDoc.GenerateXSLT();

   // ----- generate Java Code -----
   objOptions.CodeDefaultOutputDirectory = "C:\\test\\TestCOMServer\\Java";
   objDoc.GenerateJavaCode();

   // ----- generate CPP Code, use same cpp code options as the last time -----
   objOptions.CodeDefaultOutputDirectory = "C:\\test\\TestCOMServer\\CPP";
   objDoc.GenerateCppCode();

   // ----- generate C# Code, use options C# code options as the last time -----
   objOptions.CodeDefaultOutputDirectory = "C:\\test\\TestCOMServer\\CHash";
   objDoc.GenerateCHashCode();
}
catch (err)
   { ERROR ("while generating XSL or program code", err); }


// hide MapForce to allow it to shut down
objMapForce.Visible = false;

// -------------------- end example ---------------------
```

17.6.5 Ausführen eines Mappings

Im folgenden JScript-Beispiel wird gezeigt, wie Sie ein bestehendes Dokument mit einem einfachen Mapping laden, seine Komponenten aufrufen, Input- und Output-Instanzdateinamen definieren und das Mapping ausführen.

```
/*
   This sample file performs the following operations:

   Load existing MapForce mapping document.
   Find source and target component.
   Set input and output instance filenames.
   Execute the transformation.

   Works with Windows scripting host.
*/

// ---- general helpers ------------------------------

function Exit( message )
{
   WScript.Echo( message );
   WScript.Quit(-1);
}

function ERROR( message, err )
{
   if( err != null )
      Exit( "ERROR: (" + (err.number & 0xffff) + ") " + err.description + " - " + message
);
   else
      Exit( "ERROR: " + message );
}



// ---- MapForce constants -----------------------

var eComponentUsageKind_Unknown     = 0;
var eComponentUsageKind_Instance    = 1;
var eComponentUsageKind_Input       = 2;
var eComponentUsageKind_Output      = 3;



// ---- MapForce helpers -----------------------

// Searches in the specified mapping for a component by name and returns it.
// If not found, throws an error.
function FindComponent( mapping, component_name )
{
   var components = mapping.Components;
   for( var i = 0 ; i < components.Count ; ++i )
```


© 2016-2022 Altova GmbH

```
   {
      var component = components.Item( i + 1 );
      if( component.Name == component_name )
         return component;
   }
   throw new Error( "Cannot find component with name " + component_name );
}



// Browses components in a mapping and returns the first one found acting as
// source component (i.e. having connections on its right side).
function GetFirstSourceComponent( mapping )
{
   var components = mapping.Components;
   for( var i = 0 ; i < components.Count ; ++i )
   {
      var component = components.Item( i + 1 );
      if( component.UsageKind == eComponentUsageKind_Instance &&
         component.HasOutgoingConnections )
      {
         return component;
      }
   }
   throw new Error( "Cannot find a source component" );
}

// Browses components in a mapping and returns the first one found acting as
// target component (i.e. having connections on its left side).
function GetFirstTargetComponent( mapping )
{
   var components = mapping.Components;
   for( var i = 0 ; i < components.Count ; ++i )
   {
      var component = components.Item( i + 1 );
      if( component.UsageKind == eComponentUsageKind_Instance &&
         component.HasIncomingConnections )
      {
         return component;
      }
   }
   throw new Error( "Cannot find a target component" );
}



function IndentTextLines( s )
{
   return "\t" + s.replace( /\n/g, "\n\t" );
}

function GetAppoutputLineFullText( oAppoutputLine )
{
   var s = oAppoutputLine.GetLineText();
```

```
   var oAppoutputChildLines = oAppoutputLine.ChildLines;
   var i;

   for( i = 0 ; i < oAppoutputChildLines.Count ; ++i )
   {
      oAppoutputChildLine = oAppoutputChildLines.Item( i + 1 );
      sChilds = GetAppoutputLineFullText( oAppoutputChildLine );
      s += "\n" + IndentTextLines( sChilds );
   }

   return s;
}

// Create a nicely formatted string from AppOutputLines
function GetResultMessagesString( oAppoutputLines )
{
   var s1 = "Transformation result messages:\n";
   var oAppoutputLine;
   var i;

   for( i = 0 ; i < oAppoutputLines.Count ; ++i )
   {
      oAppoutputLine = oAppoutputLines.Item( i + 1 );
      s1 += GetAppoutputLineFullText( oAppoutputLine );
      s1 += "\n";
   }

   return s1;
}



// ---- MAIN -----------------------------------

var wshShell;
var fso;
var mapforce;

// create the Shell and FileSystemObject of the windows scripting system
try
{
   wshShell = WScript.CreateObject( "WScript.Shell" );
   fso = WScript.CreateObject( "Scripting.FileSystemObject" );
}
catch( err )
   { ERROR( "Can't create windows scripting objects", err ); }

// open MapForce or access currently running instance
try
{
   mapforce = WScript.GetObject( "", "MapForce.Application" );
}
catch( err )
   { ERROR( "Can't access or create MapForce.Application", err ); }
```


© 2016-2022 Altova GmbH

```
try
{
   // Make MapForce UI visible. This is an API requirement for output generation.
  mapforce.Visible = true;

   // open an existing mapping.
   // **** adjust the examples path to your needs ! **************
   var sMapForceExamplesPath = fso.BuildPath(
            wshShell.SpecialFolders( "MyDocuments" ),
            "Altova\\MapForce2023\\MapForceExamples" );
   var sDocFilename = fso.BuildPath( sMapForceExamplesPath, "PersonList.mfd" );
   var doc = mapforce.OpenDocument( sDocFilename );

   // Find existing components by name in the main mapping.
   // Note, the names of components may not be unique as a schema component's name
   // is derived from its schema file name.
   var source_component = FindComponent( doc.MainMapping, "Employees" );
   var target_component = FindComponent( doc.MainMapping, "PersonList" );
   // If you do not know the names of the components for some reason, you could
   // use the following functions instead of FindComponent.
   //var source_component = GetFirstSourceComponent( doc.MainMapping );
   //var target_component = GetFirstTargetComponent( doc.MainMapping );

   // specify the desired input and output files.
   source_component.InputInstanceFile = fso.BuildPath( sMapForceExamplesPath,
"Employees.xml" );
   target_component.OutputInstanceFile = fso.BuildPath( sMapForceExamplesPath,
"test_transformation_results.xml" );

   // Perform the transformation.
   // You can use doc.GenerateOutput() if you do not need result messages.
   // If you have a mapping with more than one target component and you want
   // to execute the transformation only for one specific target component,
   // call target_component.GenerateOutput() instead.
   var result_messages = doc.GenerateOutputEx();

   var summary_info =
         "Transformation performed from " + source_component.InputInstanceFile + "\n" +
         "to " + target_component.OutputInstanceFile + "\n\n" +
         GetResultMessagesString( result_messages );
   WScript.Echo( summary_info );
}
catch( err )
{
   ERROR( "Failure", err );
}
```

17.6.6 Projektaufgaben

Im folgenden JScript-Beispiel wird gezeigt, wie Sie mit Hilfe der MapForce API komplexe Aufgaben im Zusammenhang mit MapForce-Projekten automatisieren. Stellen Sie vor Ausführung des Mappings sicher,

dass der Inhalt der Variablen `strSamplePath` auf den folgenden Ordner Ihrer MapForce-Installation verweist: **`C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2023\MapForceExamples`**.

Um alle Operationen im nachfolgenden Beispiel erfolgreich ausführen zu können, benötigen Sie die Enterprise Edition von MapForce. Wenn Sie mit der Professional Edition arbeiten, sollten Sie die Zeilen, in denen das Webservice-Projekt eingefügt wird, auskommentieren.

```
// ////////////// global variables /////////////////
var objMapForce = null;
var objWshShell = null;
var objFSO = null;

// !!! adapt the following path to your needs. !!!
var strSamplePath = "C:\\Users\\<username>\\Documents\\Altova\\MapForce2023\
\MapForceExamples\\";

// ////////////////////// Helpers ///////////////////////////////

function Exit(strErrorText)
{
   WScript.Echo(strErrorText);
   WScript.Quit(-1);
}

function ERROR(strText, objErr)
{
   if (objErr != null)
      Exit ("ERROR: (" + (objErr.number & 0xffff) + ")" + objErr.description + " - " +
strText);
   else
      Exit ("ERROR: " + strText);
}

function CreateGlobalObjects ()
{
   // the Shell and FileSystemObject of the windows scripting host often useful
   try
   {
      objWshShell = WScript.CreateObject("WScript.Shell");
      objFSO = WScript.CreateObject("Scripting.FileSystemObject");
   }
   catch(err)
      { Exit("Can't create WScript.Shell object"); }

   // create the MapForce connection
   // if there is a running instance of MapForce (that never had a connection) - use it
   // otherwise, we automatically create a new instance
   try
   {
      objMapForce = WScript.GetObject("", "MapForce.Application");
   }
   catch(err)
   {
      { Exit("Can't access or create MapForce.Application"); }
```


© 2016-2022 Altova GmbH

```
   }
}

// ------------------------------------------------------
// print project tree items and their properties recursively.
// ------------------------------------------------------
function PrintProjectTree( objProjectItemIter, strTab )
{
   while ( ! objProjectItemIter.atEnd() )
   {
      // get current project item
      objItem = objProjectItemIter.item();

      try
      {
         // ----- print common properties
         strGlobalText += strTab + "[" + objItem.Kind + "]" + objItem.Name + "\n";

         // ----- print code generation properties, if available
         try
         {
            if ( objItem.CodeGenSettings_UseDefault )
               strGlobalText += strTab + "  Use default code generation settings\n";
            else
               strGlobalText += strTab + "  code generation language is " +
                                                    objItem.CodeGenSettings_Language +
                                  " output folder is " +
objItem.CodeGenSettings_OutputFolder + "\n";
         }
         catch( err ) {}

         // ----- print WSDL settings, if available
         try
         {
            strGlobalText += strTab + "  WSDL File is " + objItem.WSDLFile +
                             " Qualified Name is " + objItem.QualifiedName + "\n";
         }
         catch( err ) {}
      }
      catch( ex )
         { strGlobalText += strTab + "[" + objItem.Kind + "]\n" }

      // ---- recurse
      PrintProjectTree( new Enumerator( objItem ), strTab + '  ' );

      objProjectItemIter.moveNext();
   }
}

// ------------------------------------------------------------
// Load example project installed with MapForce.
// ------------------------------------------------------------
function LoadSampleProject()
{
```

```
   // close open project
   objProject = objMapForce.ActiveProject;
   if ( objProject != null )
      objProject.Close();

   // open sample project and iterate through it.
   objProject = objMapForce.OpenProject(strSamplePath + "MapForceExamples.mfp");
   // dump properties of all project items
   strGlobalText = '';
   PrintProjectTree( new Enumerator (objProject), ' ' )
   WScript.Echo( strGlobalText );

   objProject.Close();
}

// ------------------------------------------------------------
// Create a new project with some folders, mappings and a
// Web service project.
// ------------------------------------------------------------
function CreateNewProject()
{
   try
   {
      // create new project and specify file to store it.
      objProject = objMapForce.NewProject(strSamplePath + "Sample.mfp");

      // create a simple folder structure
      objProject.CreateFolder( "New Folder 1");
      objFolder1 = objProject.Item(1);
      objFolder1.CreateFolder( "New Folder 2");
      objFolder2 = ( new Enumerator( objFolder1 ) ).item();   // an alternative to
Item(0)

      // add two different mappings to folder structure
      objFolder1.AddFile( strSamplePath + "DB_Altova_SQLXML.mfd");
      objMapForce.Documents.OpenDocument(strSamplePath + "InspectionReport.mfd");
      objFolder2.AddActiveFile();

      // override code generation settings for this folder
      objFolder2.CodeGenSettings_UseDefault = false;
      objFolder2.CodeGenSettings_OutputFolder = strSamplePath + "SampleOutput"
      objFolder2.CodeGenSettings_Language = 1;      //C++

      // insert Web service project based on a wsdl file from the installed examples
      objProject.InsertWebService( strSamplePath + "TimeService/TimeService.wsdl",
                                   "{http://www.Nanonull.com/TimeService/}TimeService",
                           "TimeServiceSoap",
                           true );
      objProject.Save();
      if ( ! objProject.Saved )
         WScript.Echo("problem occurred when saving project");

      // dump project tree
      strGlobalText = '';
      PrintProjectTree( new Enumerator (objProject), ' ' )
```


© 2016-2022 Altova GmbH

```
      WScript.Echo( strGlobalText );
   }
   catch (err)
   { ERROR("while creating new project", err ); }
}

// ---------------------------------------------------------
// Generate code for a project's sub-tree. Mix default code
// generation parameters and overloaded parameters.
// ---------------------------------------------------------
function GenerateCodeForNewProject()
{
   // since the Web service project contains only initial mappings,
   // we generate code only for our custom folder.
   // code generation parameters from project are used for Folder1,
   // whereas Folder2 provides overwritten values.
   objFolder = objProject.Item(1);
   objFolder1.GenerateCode();
}

// ////////////////////// MAIN //////////////////////////////

CreateGlobalObjects();
objMapForce.Visible = true;

LoadSampleProject();
CreateNewProject();
GenerateCodeForNewProject();

// uncomment to shut down application when script ends
// objMapForce.Visible = false;
```

17.7 Objektreferenz

Dieser Abschnitt enthält Informationen zu den Objekten der MapForce COM API. Die Objekte sind allgemein beschrieben, da die API mit praktisch jeder Sprache, die den Aufruf eines COM-Objekts unterstützt, verwendet werden kann. Sprachspezifische Beispiele finden Sie unter:

- C#-Beispielprojekt 1729
- Java-Beispielprojekt 1734
- JScript-Beispiele 1738

Die API-Referenz besteht aus zwei Hauptabschnitten, in denen die in der jeweiligen API verwendeten Schnittstellen und Enumerationstypen beschrieben sind. Die Enumerationswerte enthalten sowohl den String-Namen als auch einen numerischen Wert. Wenn Ihre Skripting-Umgebung Enumerationen nicht unterstützt, verwenden Sie stattdessen die numerischen Werte.

In .NET gibt es für jede Schnittstelle der MapForce COM Automation Interface eine .NET-Klasse mit demselben Namen. Auch COM-Typen werden in den entsprechenden .NET-Typ konvertiert. So wird etwa ein Typ wie `Long` aus der COM API in .NET als `System.Int32` angezeigt.

Beachten Sie in Java die folgenden Syntaxvarianten:

- **Klassen und Klassennamen** Für jede Schnittstelle des MapForce Automation Interface gibt es eine Java-Klasse mit dem Namen der Schnittstelle.
- **Methodennamen** Die Methodennamen im Java Interface sind dieselben wie die in den COM Interfaces, beginnen aber aufgrund der Java-Namenskonventionen mit einem Kleinbuchstaben. Zum Aufrufen von COM-Eigenschaften können Java-Methoden verwendet werden, deren Eigenschaftsname das Präfix get und set erhalten. Wenn eine Eigenschaft keinen Schreibzugriff ermöglicht, steht keine Setter-Methode zur Verfügung. So stehen z.B. für die Eigenschaft `Name` des `Document` Interface stehen die Java-Methoden `getName` und `setName` zur Verfügung.
- **Enumerationen** Für jede im Automation Interface definierte Enumeration ist eine Java-Enumeration desselben Namens und mit denselben Werten definiert.
- **Events und Event Handler** Für jedes Interface im Automation Interface, das Events unterstützt, steht ein Java-Interface desselben Namens plus 'Event' zur Verfügung. Um das Überladen von Einzel-Events zu vereinfachen, gibt es eine Java-Klasse mit Standardimplementierungen für alle Events. Der Name dieser Java-Klasse ist der Name des Event Interface plus 'DefaultHandler'. Beispiel:

```
Application // Java class to access the application
ApplicationEvents // Events interface for the application
ApplicationEventsDefaultHandler // Default handler for "ApplicationEvents"
```

17.7.1 Schnittstellen

17.7.1.1 Application

Die `Application`-Schnittstelle ist die Schnittstelle zu einem MapForce application-Objekt. Sie bildet den Hauptzugriffspunkt für die MapForce-Applikation selbst. Diese Schnittstelle ist der Ausgangspunkt für alle weiteren Operationen mit MapForce oder für das Abrufen oder Erstellen anderer Automation-Objekte im

© 2016-2022 Altova GmbH

Zusammenhang mit MapForce. Informationen zur Erstellung einer Instanz des Application-Objekts finden Sie unter Aufruf der API[1723].

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`
- `Options`
- `Project`
- `Documents`

Applikationsstatus:

- `Visible`
- `Name`
- `Quit`
- `Status`
- `WindowHandle`

MapForce-Designs:

- `NewDocument`
- `OpenDocument`
- `OpenURL`
- `ActiveDocument`

MapForce-Projekte:

- `NewProject`
- `OpenProject`
- `ActiveProject`

MapForce-Codegenerierung:

- `HighlightSerializedMarker`

Globale Ressourcen:

- `GlobalResourceConfig`
- `GlobalResourceFile`

Versionsinformationen:

- `Edition`
- `IsAPISupported`
- `MajorVersion`
- `MinorVersion`

Eigenschaften

Name	Beschreibung
`ActiveDocument`[1757]	Schreibgeschützt.

Name	Beschreibung
	Gibt das Automation-Objekt des gerade aktiven Dokuments zurück. Diese Eigenschaft gibt denselben Wert zurück wie `Documents.ActiveDocument`.
`ActiveProject` 1757	Schreibgeschützt. Gibt das Automation Objekt des gerade aktiven Projekts zurück.
`Application` 1758	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`Documents` 1758	Schreibgeschützt. Gibt eine Sammlung aller gerade offenen Dokumente zurück.
`Edition` 1758	Schreibgeschützt. Gibt die Version (Edition) der Applikation, z.B. "Altova MapForce Enterprise Edition" für die Enterprise Edition, zurück.
`GlobalResourceConfig` 1759	Ruft den Namen der aktiven Konfigurationsdatei für die globalen Ressourcen ab oder setzt ihn. Standardmäßig trägt die Datei den Namen **GlobalResources.xml**. Die Konfigurationsdatei kann umbenannt und unter einem beliebigen Pfad gespeichert werden. Sie können daher mehrere XML-Dateien für globale Ressourcen haben. Allerdings kann immer nur eine dieser Dateien pro Applikation aktiv sein und nur die in dieser Datei enthaltenen Definitionen stehen der Applikation zur Verfügung.
`GlobalResourceFile` 1759	Ruft die Definitionsdatei für globale Ressourcen ab oder definiert sie. Standardmäßig trägt die Datei den Namen GlobalResources.xml.
`IsAPISupported` 1760	Schreibgeschützt. Gibt "true" zurück, wenn die API in dieser Version von MapForce unterstützt wird.
`LibraryImports` 1760	Schreibgeschützt. Ruft eine Sammlung importierter Bibliotheken ab. Diese entsprechen auf der grafischen MapForce-Benutzeroberfläche den auf Applikationsebene hinzugefügten Einträgen im Fenster **Bibliotheken verwalten**.
`MajorVersion` 1761	Schreibgeschützt. Ruft die Hauptversionsnummer von MapForce ab. Die Version wird ab 1998 berechnet und wird jedes Jahr um 1 erhöht. So ist die Hauptversion für die Release 2016 z.B. "18".
`MinorVersion` 1761	Schreibgeschützt. Die Zusatznummer zur Hauptversion des Produkts z.B. 2 für 2006 R2 SP1.
`Name` 1762	Schreibgeschützt.

© 2016-2022 Altova GmbH

Name	Beschreibung
	Der Name der Applikation.
Options 1762	Schreibgeschützt. Mit dieser Eigenschaft haben Sie Zugriff auf Optionen zum Konfigurieren der Codegenerierung.
Parent 1763	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.
ServicePackVersion 1763	Schreibgeschützt. Die Service Pack-Versionsnummer des Produkts, z.B. 1 für 2016 R2 SP1.
Status 1764	Schreibgeschützt. Der Status der Applikation. Es ist einer der Werte der `ENUMApplicationStatus` Enumeration.
Visible 1764	**True**, wenn MapForce auf dem Bildschirm angezeigt wird (unter Umständen wird es von anderen Applikationen verdeckt oder als Symbol angezeigt). **False**, wenn MapForce ausgeblendet ist. Der Standardwert für MapForce, wenn es automatisch aufgrund eines Request der Automation Server `Application` gestartet wird, ist false. In allen anderen Fällen wird die Eigenschaft als **true** initialisiert. Eine Applikationsinstanz, die sichtbar ist, gilt als vom Benutzer (und möglicherweise von Clients, die über die Automation-Schnittstelle verbunden sind) gesteuert. Sie wird nur auf einen expliziten Benutzer-Request hin beendet. Um eine Applikationsinstanz zu beenden, setzen Sie ihre Sichtbarkeit auf false und entfernen Sie alle Referenzen auf diese Instanz aus Ihrem Programm. Die Applikationsinstanz wird automatisch beendet, wenn keine weiteren COM-Clients Referenzen darauf enthalten.
WindowHandle 1765	Schreibgeschützt. Ruft den Fenster-Handle der Applikation ab.

Methoden

Name	Beschreibung
HighlightSerializedMarker 1765	Verwenden Sie diese Methode, um einen Pfad in einer Mapping-Datei zu markieren, die zuvor serialisiert wurde. Wenn das entsprechende Dokument noch nicht geladen ist, wird es zuerst geladen. Eine Methode zum Aufrufen eines serialisierten Markers finden Sie unter `Document.GenerateCodeEx`.

Name	Beschreibung
NewDocument 1766	Erstellt ein neues leeres Dokument. Das neu geöffnete Dokument wird das ActiveDocument. Diese Methode ist die Kurzform von Documents.NewDocument.
NewProject 1766	Erstellt ein neues leeres Projekt. Das aktuelle Projekt wird geschlossen. Das neue Projekt kann unter ActiveProject aufgerufen werden.
NewWebServiceProject 1767	Erstellt ein neues leeres Webservice-Projekt. Das neue Projekt kann unter ActiveProject aufgerufen werden. Diese Methode steht nur in der MapForce Enterprise Edition zur Verfügung.
OpenDocument 1767	Lädt eine zuvor gespeicherte Dokumentdatei und setzt die Bearbeitung fort. Das neu geöffnete Dokument wird das ActiveDocument. Diese Methode ist eine Kurzform von Documents.OpenDocument.
OpenProject 1768	Öffnet ein vorhandenes MapForce Projekt (*.mfp). Das aktuelle Projekt wird geschlossen. Das neu geöffnete Projekt kann unter ActiveProject aufgerufen werden.
OpenURL 1768	Lädt eine zuvor gespeicherte Dokumentdatei von einem URL-Pfad. Ermöglicht die Eingabe von Benutzername und Passwort.
Quit 1769	Trennt die Verbindung zu MapForce, um die Applikation beenden zu können. Der Aufruf dieser Methode ist optional, da MapForce alle externen COM-Verbindungen aufzeichnet und getrennte Verbindungen automatisch erkennt. Nähere Informationen zum automatischen Beenden finden Sie unter der Eigenschaft Visible.

Events

Name	Beschreibung
OnDocumentOpened 1770	Das Event wird ausgelöst, wenn ein vorhandenes oder neues Dokument geöffnet wird. Das entsprechende Event zum Schließen ist Document.OnDocumentClosed.
OnProjectOpened 1770	Das Event wird ausgelöst, wenn ein vorhandenes oder neues Projekt in die Applikation geladen wird. Das entsprechende Event zum Schließen ist Project.OnProjectClosed.
OnShutdown 1770	Dieses Event wird ausgelöst, wenn die Applikation beendet wird.

© 2016-2022 Altova GmbH

17.7.1.1.1 Eigenschaften

17.7.1.1.1.1 ActiveDocument

Gibt das Automation-Objekt des gerade aktiven Dokuments zurück. Diese Eigenschaft gibt denselben Wert zurück wie `Documents.ActiveDocument`.

Signatur

```
ActiveDocument : Document
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.2 ActiveProject

Gibt das Automation Objekt des gerade aktiven Projekts zurück.

Signatur

```
ActiveProject : Project
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.3 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.4 Documents

Gibt eine Sammlung aller gerade offenen Dokumente zurück.

Signatur

```
Documents : Documents
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.5 Edition

Gibt die Version (Edition) der Applikation, z.B. "Altova MapForce Enterprise Edition" für die Enterprise Edition, zurück.

Signatur

```
Edition : String
```


© 2016-2022 Altova GmbH

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.6 GlobalResourceConfig

Ruft den Namen der aktiven Konfigurationsdatei für die globalen Ressourcen ab oder setzt ihn. Standardmäßig trägt die Datei den Namen **GlobalResources.xml**.

Die Konfigurationsdatei kann umbenannt und unter einem beliebigen Pfad gespeichert werden. Sie können daher mehrere XML-Dateien für globale Ressourcen haben. Allerdings kann immer nur eine dieser Dateien pro Applikation aktiv sein und nur die in dieser Datei enthaltenen Definitionen stehen der Applikation zur Verfügung.

Signatur

```
GlobalResourceConfig : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.7 GlobalResourceFile

Ruft die Definitionsdatei für globale Ressourcen ab oder definiert sie. Standardmäßig trägt die Datei den Namen GlobalResources.xml.

Signatur

```
GlobalResourceFile : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.8 IsAPISupported

Gibt "true" zurück, wenn die API in dieser Version von MapForce unterstützt wird.

Signatur

```
IsAPISupported : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.9 LibraryImports

Ruft eine Sammlung importierter Bibliotheken ab. Diese entsprechen auf der grafischen MapForce-Benutzeroberfläche den auf Applikationsebene hinzugefügten Einträgen im Fenster **Bibliotheken verwalten**.

Signatur

```
LibraryImports : LibraryImports
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.1.1.10 MajorVersion

Ruft die Hauptversionsnummer von MapForce ab. Die Version wird ab 1998 berechnet und wird jedes Jahr um 1 erhöht. So ist die Hauptversion für die Release 2016 z.B. "18".

Signatur

```
MajorVersion : Long
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.11 MinorVersion

Die Zusatznummer zur Hauptversion des Produkts z.B. 2 für 2006 R2 SP1.

Signatur

```
MinorVersion : Long
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.12 Name

Der Name der Applikation.

Signatur

```
Name : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.13 Options

Mit dieser Eigenschaft haben Sie Zugriff auf Optionen zum Konfigurieren der Codegenerierung.

Signatur

```
Options : Options
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.1.1.14 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.15 ServicePackVersion

Die Service Pack-Versionsnummer des Produkts, z.B. 1 für 2016 R2 SP1.

Signatur

```
ServicePackVersion : Long
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.16 Status

Der Status der Applikation. Es ist einer der Werte der `ENUMApplicationStatus` Enumeration.

Signatur

```
Status : ENUMApplicationStatus
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.1.17 Visible

True, wenn MapForce auf dem Bildschirm angezeigt wird (unter Umständen wird es von anderen Applikationen verdeckt oder als Symbol angezeigt).

False, wenn MapForce ausgeblendet ist. Der Standardwert für MapForce, wenn es automatisch aufgrund eines Request der Automation Server `Application` gestartet wird, ist false. In allen anderen Fällen wird die Eigenschaft als **true** initialisiert.

Eine Applikationsinstanz, die sichtbar ist, gilt als vom Benutzer (und möglicherweise von Clients, die über die Automation-Schnittstelle verbunden sind) gesteuert. Sie wird nur auf einen expliziten Benutzer-Request hin beendet. Um eine Applikationsinstanz zu beenden, setzen Sie ihre Sichtbarkeit auf false und entfernen Sie alle Referenzen auf diese Instanz aus Ihrem Programm. Die Applikationsinstanz wird automatisch beendet, wenn keine weiteren COM-Clients Referenzen darauf enthalten.

Signatur

```
Visible : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.1.1.18 WindowHandle

Ruft den Fenster-Handle der Applikation ab.

Signatur

```
WindowHandle : Long
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.2 Methoden

17.7.1.1.2.1 HighlightSerializedMarker

Verwenden Sie diese Methode, um einen Pfad in einer Mapping-Datei zu markieren, die zuvor serialisiert wurde. Wenn das entsprechende Dokument noch nicht geladen ist, wird es zuerst geladen. Eine Methode zum Aufrufen eines serialisierten Markers finden Sie unter `Document.GenerateCodeEx`.

Signatur

```
HighlightSerializedMarker(in i_strSerializedMarker:String) -> Void
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strSerializedMarker	`String`	Das zu markierende `ErrorMarker`-Objekt. Diesen Wert erhalten Sie mit Hilfe von `ErrorMaker.Serialized`.

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1007	Der in **i_strSerializedMarker** übergebene String wird nicht als serialisierter MapForce Marker erkannt.
1008	Der Marker verweist auf einen nicht mehr gültigen Pfad.

17.7.1.1.2.2 NewDocument

Erstellt ein neues leeres Dokument. Das neu geöffnete Dokument wird das `ActiveDocument`. Diese Methode ist die Kurzform von `Documents.NewDocument`.

Signatur

```
NewDocument() -> Document
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.2.3 NewProject

Erstellt ein neues leeres Projekt. Das aktuelle Projekt wird geschlossen. Das neue Projekt kann unter `ActiveProject` aufgerufen werden.

Signatur

```
NewProject() -> Project
```

Allgemeine Signatur

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.2.4 NewWebServiceProject

Erstellt ein neues leeres Webservice-Projekt. Das neue Projekt kann unter `ActiveProject` aufgerufen werden. Diese Methode steht nur in der MapForce Enterprise Edition zur Verfügung.

Signatur

```
NewWebServiceProject() -> Project
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1004	Fehler bei der Erstellung des neuen Projekts.
1005	Falsche MapForce-Edition.

17.7.1.1.2.5 OpenDocument

Lädt eine zuvor gespeicherte Dokumentdatei und setzt die Bearbeitung fort. Das neu geöffnete Dokument wird das `ActiveDocument`. Diese Methode ist eine Kurzform von `Documents.OpenDocument`.

Signatur

```
OpenDocument(in i_strFileName:String) -> Document
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFileName	`String`	Der Pfad zu dem zu öffnenden Dokument.

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.2.6 OpenProject

Öffnet ein vorhandenes MapForce Projekt (*.mfp). Das aktuelle Projekt wird geschlossen. Das neu geöffnete Projekt kann unter `ActiveProject` aufgerufen werden.

Signatur

```
OpenProject(in i_strFileName:String) -> Project
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFileName	`String`	Der Pfad zu dem zu öffnenden Projekt.

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1002	Der bereitgestellte Dateiname ist nicht gültig.

17.7.1.1.2.7 OpenURL

Lädt eine zuvor gespeicherte Dokumentdatei von einem URL-Pfad. Ermöglicht die Eingabe von Benutzername und Passwort.

Signatur

```
OpenURL(in strURL:String, in strUser:String, in strPassword:String) -> Void
```


© 2016-2022 Altova GmbH

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
strURL	`String`	Die URL, von der das Dokument geladen werden soll.
strUser	`String`	Der Benutzername, der für den Zugriff auf die URL benötigt wird.
strPassword	`String`	Das Passwort, das für den Zugriff auf die URL benötigt wird.

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1002	Die bereitgestellte URL ist nicht gültig.
1006	Fehler beim Öffnen der URL-Datei.

17.7.1.1.2.8 Quit

Trennt die Verbindung zu MapForce, um die Applikation beenden zu können. Der Aufruf dieser Methode ist optional, da MapForce alle externen COM-Verbindungen aufzeichnet und getrennte Verbindungen automatisch erkennt. Nähere Informationen zum automatischen Beenden finden Sie unter der Eigenschaft `Visible`.

Signatur

```
Quit() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1000	Das application-Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.1.3 Events

17.7.1.1.3.1 OnDocumentOpened

Das Event wird ausgelöst, wenn ein vorhandenes oder neues Dokument geöffnet wird. Das entsprechende Event zum Schließen ist `Document.OnDocumentClosed`.

Signatur

```
OnDocumentOpened(in i_ipDocument:Document) : Void
```

17.7.1.1.3.2 OnProjectOpened

Das Event wird ausgelöst, wenn ein vorhandenes oder neues Projekt in die Applikation geladen wird. Das entsprechende Event zum Schließen ist `Project.OnProjectClosed`.

Signatur

```
OnProjectOpened(in i_ipProject:Project) : Void
```

17.7.1.1.3.3 OnShutdown

Dieses Event wird ausgelöst, wenn die Applikation beendet wird.

Signatur

```
OnShutdown : Void
```

17.7.1.2 AppOutputLine

Repräsentiert eine Meldungszeile. Seine Struktur ist im Gegensatz zu `ErrorMarker` detaillierter und kann eine Sammlung von untergeordneten Zeilen, die eine hierarchische Struktur von Meldungszeilen bilden, enthalten.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Zeilenzugriff:

- `GetLineSeverity`
- `GetLineSymbol`
- `GetLineText`

© 2016-2022 Altova GmbH

- `GetLineTextEx`
- `GetLineTextWithChildren`
- `GetLineTextWithChildrenEx`

Eine einzelne `AppOutputLine` besteht aus einer oder mehreren untergeordneten Zeilen. Zugriff auf untergeordnete Zeilen:

- `GetLineCount`

Eine untergeordnete Zeile besteht aus einer oder mehreren Zellen. Zellenzugriff:

- `GetCellCountInLine`
- `GetCellIcon`
- `GetCellSymbol`
- `GetCellText`
- `GetCellTextDecoration`
- `GetIsCellText`

Unterhalb einer `AppOutputLine` können null, eine oder mehrere untergeordnete Zeilen vorhanden sein, die selbst den Typ `AppOutputLine` haben und somit eine hierarchische Struktur bilden.

Zugriff auf untergeordnete Zeilen:

- `ChildLines`

Eigenschaften

Name	Beschreibung
`Application` 1772	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`ChildLines` 1773	Schreibgeschützt. Gibt eine Sammlung der Zeilen zurück, die der aktuellen Zeile direkt untergeordnet sind.
`Parent` 1773	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.

Methoden

Name	Beschreibung
`GetCellCountInLine` 1774	Ruft die Anzahl der Zellen in der Subzeile ab, die durch `nLine` in der aktuellen `AppOutputLine` angegeben ist.
`GetCellIcon` 1774	Ruft das Symbol der Zelle ab, die in der durch `nLine` bezeichneten Subzeile der aktuellen `AppOutputLine` durch `nCell` angegeben ist.
`GetCellSymbol` 1775	Ruft das Symbol der Zelle ab, das in der durch `nLine` bezeichneten Subzeile der aktuellen `AppOutputLine` durch `nCell` angegeben ist.

Name	Beschreibung
GetCellText 1775	Ruft den Text der Zelle ab, die in der durch nLine bezeichneten Subzeile der aktuellen AppOutputLine durch nCell bezeichnet ist.
GetCellTextDecoration 1776	Ruft die Verzierung der Textzelle ab, die in der durch nLine bezeichneten Subzeile der aktuellen AppOutputLine durch nCell bezeichnet ist. Der Wert kann einer der ENUMAppOutputLine_TextDecoration Werte sein.
GetIsCellText 1777	Gibt true zurück, wenn die Zelle, die in der durch nLine bezeichneten Subzeile der aktuellen AppOutputLine durch nCell bezeichnet ist, eine Textzelle ist.
GetLineCount 1778	Ruft die Anzahl der Subzeilen ab, aus denen die aktuelle Zeile besteht.
GetLineSeverity 1778	Ruft den Schweregrad der Zeile ab. Der Wert kann einer der ENUMAppOutputLine_Severity Werte sein.
GetLineSymbol 1779	Ruft das Symbol ab, das der gesamten Zeile zugewiesen ist.
GetLineText 1779	Ruft den Inhalt der Zeile als Text ab.
GetLineTextEx 1780	Ruft unter Verwendung der angegebenen Part- und Zeilentrennzeichen den Inhalt der Zeile als Text ab.
GetLineTextWithChildren 1780	Ruft den Inhalt der Zeile inklusive aller untergeordneten Zeilen als Text ab.
GetLineTextWithChildrenEx 1781	Ruft den Inhalt der Zeile inklusive aller untergeordneten Zeilen als Text ab, wobei die angegebenen Trennzeichen für Teile, Zeilen, Tabulatoren und Elemente verwendet werden.

17.7.1.2.1 Eigenschaften

17.7.1.2.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.1.2 ChildLines

Gibt eine Sammlung der Zeilen zurück, die der aktuellen Zeile direkt untergeordnet sind.

Signatur

```
ChildLines : AppOutputLines
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.1.3 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : AppOutputLines
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.2 Methoden

17.7.1.2.2.1 GetCellCountInLine

Ruft die Anzahl der Zellen in der Subzeile ab, die durch `nLine` in der aktuellen `AppOutputLine` angegeben ist.

Signatur

```
GetCellCountInLine(in nLine:Long) -> Long
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
nLine	**Long**	Definiert den nullbasierten Index der Zeile.

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.2.2 GetCellIcon (obsolete)

Ruft das Symbol der Zelle ab, die in der durch `nLine` bezeichneten Subzeile der aktuellen `AppOutputLine` durch `nCell` angegeben ist.

Signatur

```
GetCellIcon(in nLine:Long, in nCell:Long) -> Long
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
nLine	**Long**	

© 2016-2022 Altova GmbH

Name	Typ	Beschreibung
nCell	**Long**	

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.2.3 GetCellSymbol

Ruft das Symbol der Zelle ab, das in der durch `nLine` bezeichneten Subzeile der aktuellen `AppOutputLine` durch `nCell` angegeben ist.

Signatur

```
GetCellSymbol(in nLine:Long, in nCell:Long) -> AppOutputLineSymbol
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
nLine	**Long**	Definiert den nullbasierten Index der Zeile.
nCell	**Long**	Definiert den nullbasierten Index der Zelle.

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.2.4 GetCellText

Ruft den Text der Zelle ab, die in der durch `nLine` bezeichneten Subzeile der aktuellen `AppOutputLine` durch `nCell` bezeichnet ist.

Signatur

```
GetCellText(in nLine:Long, in nCell:Long) -> String
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
nLine	**Long**	Definiert den nullbasierten Index der Zeile.
nCell	**Long**	Definiert den nullbasierten Index der Zelle.

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.2.5 GetCellTextDecoration

Ruft die Verzierung der Textzelle ab, die in der durch `nLine` bezeichneten Subzeile der aktuellen `AppOutputLine` durch `nCell` bezeichnet ist. Der Wert kann einer der `ENUMAppOutputLine_TextDecoration` Werte sein.

Signatur

```
GetCellTextDecoration(in nLine:Long, in nCell:Long) -> Long
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
nLine	**Long**	Definiert den nullbasierten Index der Zeile.
nCell	**Long**	Definiert den nullbasierten Index der Zelle.

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.2.6 GetIsCellText

Gibt true zurück, wenn die Zelle, die in der durch `nLine` bezeichneten Subzeile der aktuellen `AppOutputLine` durch `nCell` bezeichnet ist, eine Textzelle ist.

Signatur

```
GetIsCellText(in nLine:Long, in nCell:Long) -> Boolean
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
nLine	**Long**	Definiert den nullbasierten Index der Zeile.
nCell	**Long**	Definiert den nullbasierten Index der Zelle.

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.2.7 GetLineCount

Ruft die Anzahl der Subzeilen ab, aus denen die aktuelle Zeile besteht.

Signatur

```
GetLineCount() -> Long
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.2.8 GetLineSeverity

Ruft den Schweregrad der Zeile ab. Der Wert kann einer der `ENUMAppOutputLine_Severity` Werte sein.

Signatur

```
GetLineSeverity() -> Long
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.2.2.9 GetLineSymbol

Ruft das Symbol ab, das der gesamten Zeile zugewiesen ist.

Signatur

```
GetLineSymbol() -> AppOutputLineSymbol
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.2.10 GetLineText

Ruft den Inhalt der Zeile als Text ab.

Signatur

```
GetLineText() -> String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.2.11 GetLineTextEx

Ruft unter Verwendung der angegebenen Part- und Zeilentrennzeichen den Inhalt der Zeile als Text ab.

Signatur

```
GetLineTextEx(in psTextPartSeperator:String, in psLineSeperator:String) -> String
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
psTextPartSeperator	`String`	
psLineSeperator	`String`	

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.2.2.12 GetLineTextWithChildren

Ruft den Inhalt der Zeile inklusive aller untergeordneten Zeilen als Text ab.

Signatur

```
GetLineTextWithChildren() -> String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.2.2.13 GetLineTextWithChildrenEx

Ruft den Inhalt der Zeile inklusive aller untergeordneten Zeilen als Text ab, wobei die angegebenen Trennzeichen für Teile, Zeilen, Tabulatoren und Elemente verwendet werden.

Signatur

```
GetLineTextWithChildrenEx(in psPartSep:String, in psLineSep:String, in psTabSep:String,
in psItemSep:String) -> String
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
psPartSep	`String`	
psLineSep	`String`	
psTabSep	`String`	
psItemSep	`String`	

Fehler

Fehlercode	Beschreibung
4100	Das Objekt ist nicht mehr gültig.
4101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.3 AppOutputLines

Repräsentiert eine Sammlung von `AppOutputLine`-Meldungszeilen.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Iterierend durch die Sammlung:

- `Count`
- `Item`

Eigenschaften

Name	Beschreibung
Application 1782	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
Count 1783	Schreibgeschützt. Ruft die Anzahl der Zeilen in der Sammlung ab.
Item 1783	Schreibgeschützt. Ruft die Zeile am Index n aus der Sammlung ab. Indizes beginnen mit 1.
Parent 1784	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.

17.7.1.3.1 Eigenschaften

17.7.1.3.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4000	Das Objekt ist nicht mehr gültig.
4001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.3.1.2 Count

Ruft die Anzahl der Zeilen in der Sammlung ab.

Signatur

```
Count : Integer
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4000	Das Objekt ist nicht mehr gültig.
4001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.3.1.3 Item

Ruft die Zeile am Index `n` aus der Sammlung ab. Indizes beginnen mit 1.

Signatur

```
Item(in n:Integer) : AppOutputLine
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4000	Das Objekt ist nicht mehr gültig.
4001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.3.1.4 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : AppOutputLine
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4000	Das Objekt ist nicht mehr gültig.
4001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.4 AppOutputLineSymbol

Ein `AppOutputLineSymbol` stellt einen Link in einer `AppOutputLine`-Meldungszeile dar, der im MapForce Meldungsfenster angeklickt werden kann. Wird auf eine Zelle einer `AppOutputLine` oder auf die gesamte Zeile angewendet.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Zugriff auf die AppOutputLineSymbol-Methoden:

- `GetSymbolHREF`
- `GetSymbolID`
- `IsSymbolHREF`

Eigenschaften

Name	Beschreibung
`Application` 1785	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`Parent` 1785	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.

© 2016-2022 Altova GmbH

Methoden

Name	Beschreibung
GetSymbolHREF 1786	Wenn das Symbol vom Typ URL ist, wird die URL als String zurückgegeben.
GetSymbolID 1786	Ruft die ID des Symbols ab.
IsSymbolHREF 1787	Gibt an, ob das Symbol vom Typ URL ist.

17.7.1.4.1 Eigenschaften

17.7.1.4.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4200	Das Objekt ist nicht mehr gültig.
4201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.4.1.2 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4200	Das Objekt ist nicht mehr gültig.
4201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.4.2 Methoden

17.7.1.4.2.1 GetSymbolHREF

Wenn das Symbol vom Typ URL ist, wird die URL als String zurückgegeben.

Signatur

```
GetSymbolHREF() -> String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4200	Das Objekt ist nicht mehr gültig.
4201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.4.2.2 GetSymbolID

Ruft die ID des Symbols ab.

Signatur

```
GetSymbolID() -> Long
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4200	Das Objekt ist nicht mehr gültig.

© 2016-2022 Altova GmbH

Fehlercode	Beschreibung
4201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.4.2.3 IsSymbolHREF

Gibt an, ob das Symbol vom Typ URL ist.

Signatur

```
IsSymbolHREF() -> Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
4200	Das Objekt ist nicht mehr gültig.
4201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5 Component

Eine `Komponente` stellt eine MapForce-Komponente dar.

Mit Hilfe der Eigenschaften `Application` und `Parent` können Sie durch das Control navigieren.

Komponenteneigenschaften:

- `HasIncomingConnections`
- `HasOutgoingConnections`
- `CanChangeInputInstanceFile`
- `CanChangeOutputInstanceFile`
- `ComponentName`.
- `ID`
- `IsParameterInputRequired`
- `IsParameterSequence`
- `Name`
- `Preview`
- `Schema`
- `SubType`
- `Type`

Eigenschaften im Zusammenhang mit Instanzen:

- `InputInstanceFile`
- `OutputInstanceFile`

Datapoints:

- `GetRootDatapoint`

Ausführung:

- `GenerateOutput`

Eigenschaften

Name	Beschreibung
`Application` 1790	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`CanChangeInputInstanceFile` 1790	Schreibgeschützt. Gibt an, ob der Name der Input-Instanzdatei geändert werden kann. Gibt "false" zurück, wenn die Komponente einen Dateinamen-Node hat und dieser Node eine Verbindung auf seiner linken (Input-) Seite hat. Andernfalls wird "true" zurückgegeben. Wenn die Komponente keinen Dateinamen-Node hat, wird "false" zurückgegeben.
`CanChangeOutputInstanceFile` 1791	Schreibgeschützt. Gibt an, ob der Name der Output-Instanzdatei geändert werden kann. Gibt "false" zurück, wenn die Komponente einen Dateinamen-Node hat und dieser Node eine Verbindung auf seiner linken (Input-) Seite hat. Andernfalls wird "true" zurückgegeben. Wenn die Komponente keinen Dateinamen-Node hat, wird "false" zurückgegeben.
`ComponentName` 1792	Ruft den Namen der Komponente ab oder definiert ihn.
`HasIncomingConnections` 1792	Schreibgeschützt. Gibt an, ob die Komponente (auf der linken Seite) mit Ausnahme des Dateinamen-Node eingehende Verbindungen hat. Eine eingehende Verbindung am Dateinamen-Node hat keine Auswirkung auf den zurückgegebenen Wert.
`HasOutgoingConnections` 1793	Schreibgeschützt. Gibt an, ob die Komponente (auf der rechten Seite) ausgehende Verbindungen hat.
`ID` 1793	Schreibgeschützt. Ruft die ID der Komponente ab.
`InputInstanceFile` 1794	Ruft die Input-Instanzdatei der Komponente ab oder definiert sie.

© 2016-2022 Altova GmbH

Name	Beschreibung
IsParameterInputRequired 1794	Ruft ab oder definiert, ob für die Input-Parameterkomponente in der Funktionsaufruf-Komponente der benutzerdefinierten Funktion, in der sich diese Input-Parameterkomponente befindet, eine eingehende Verbindung erforderlich ist. Diese Eigenschaft funktioniert nur bei Komponenten, die Input-Parameterkomponenten sind.
IsParameterSequence 1795	Ruft ab oder definiert, ob die Input- oder Output-Parameterkomponente Sequenzen unterstützt. Diese Eigenschaft funktioniert nur bei Komponenten, die Input- oder Output-Parameterkomponenten sind.
Name 1795	Schreibgeschützt. Ruft den Namen der Komponente ab.
OutputInstanceFile 1795	Ruft die Output-Instanzdatei der Komponente ab oder definiert sie. Wenn der "Datei"-Konnektor einer Komponente mit einem anderen Datenelement im Mapping verbunden wurde, erhalten Sie beim Versuch die Ausgabeinstanzdatei einer Komponente mittels `OutputInstanceFile` über die API aufzurufen, keine Daten.
Parent 1796	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.
Preview 1796	Ruft ab oder definiert, ob die Komponente die aktuelle Vorschaukomponente ist. Diese Eigenschaft funktioniert nur bei Komponenten, bei denen es sich um Zielkomponenten im Hauptmapping des Dokuments handelt. Es kann im Hauptmapping immer nur eine Zielkomponente gleichzeitig als Vorschaukomponente verwendet werden. Wenn Sie diese Eigenschaft definieren, kann sie nur auf "true" gesetzt werden. Damit wird dann die Eigenschaft `Preview` bei allen anderen Komponenten implizit auf "false" gesetzt. Wenn es im Hauptmapping nur eine einzige Zielkomponente gibt, so ist diese auch die Vorschaukomponente.
Schema 1797	Schreibgeschützt. Ruft den Schemadateinamen der Komponente ab.
SubType 1798	Schreibgeschützt. Ruft den Subtyp der Komponente ab.
Type 1798	Schreibgeschützt. Ruft den Typ der Komponente ab.

Name	Beschreibung
UsageKind 1798	Schreibgeschützt. Ruft die Verwendungsart der Komponente ab.

Methoden

Name	Beschreibung
GenerateOutput 1799	Generiert mit Hilfe einer MapForce-internen Mapping-Sprache die Ausgabedatei(en), die im Mapping nur für die aktuelle Komponente definiert sind. Der Name/Die Namen der Ausgabedatei(en) sind als Eigenschaft der aktuellen Komponente definiert, welche im Mapping für diese Generierung das Ausgabeelement ist.
GetRootDatapoint 1800	Ruft einen Root-Datapoint auf der linken (Input) oder rechten (Output)-Seite einer Komponente ab. Um Subelemente und untergeordnete Elemente davon aufzurufen, stellt das `Datapoint`-Objekt weitere Methoden bereit.

17.7.1.5.1 Eigenschaften

17.7.1.5.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.1.2 CanChangeInputInstanceFile

Gibt an, ob der Name der Input-Instanzdatei geändert werden kann.

© 2016-2022 Altova GmbH

Gibt "false" zurück, wenn die Komponente einen Dateinamen-Node hat und dieser Node eine Verbindung auf seiner linken (Input-) Seite hat. Andernfalls wird "true" zurückgegeben. Wenn die Komponente keinen Dateinamen-Node hat, wird "false" zurückgegeben.

Signatur

```
CanChangeInputInstanceFile : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.1.3 CanChangeOutputInstanceFile

Gibt an, ob der Name der Output-Instanzdatei geändert werden kann.

Gibt "false" zurück, wenn die Komponente einen Dateinamen-Node hat und dieser Node eine Verbindung auf seiner linken (Input-) Seite hat. Andernfalls wird "true" zurückgegeben.
Wenn die Komponente keinen Dateinamen-Node hat, wird "false" zurückgegeben.

Signatur

```
CanChangeOutputInstanceFile : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.1.4 ComponentName

Ruft den Namen der Komponente ab oder definiert ihn.

Signatur

```
ComponentName : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1246	Die Komponente unterstützt die Definition ihres Namens nicht.
1247	Ungültiger Komponentenname.

17.7.1.5.1.5 HasIncomingConnections

Gibt an, ob die Komponente (auf der linken Seite) mit Ausnahme des Dateinamen-Node eingehende Verbindungen hat. Eine eingehende Verbindung am Dateinamen-Node hat keine Auswirkung auf den zurückgegebenen Wert.

Signatur

```
HasIncomingConnections : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.5.1.6 HasOutgoingConnections

Gibt an, ob die Komponente (auf der rechten Seite) ausgehende Verbindungen hat.

Signatur

```
HasOutgoingConnections : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.1.7 ID

Ruft die ID der Komponente ab.

Signatur

```
ID : Long
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.1.8 InputInstanceFile

Ruft die Input-Instanzdatei der Komponente ab oder definiert sie.

Signatur

```
InputInstanceFile : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.1.9 IsParameterInputRequired

Ruft ab oder definiert, ob für die Input-Parameterkomponente in der Funktionsaufruf-Komponente der benutzerdefinierten Funktion, in der sich diese Input-Parameterkomponente befindet, eine eingehende Verbindung erforderlich ist. Diese Eigenschaft funktioniert nur bei Komponenten, die Input-Parameterkomponenten sind.

Signatur

```
IsParameterInputRequired : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1232	Diese Operation funktioniert nur bei einer Input-Parameterkomponente
1240	Eine Änderung des Dokuments ist nicht zulässig. Es ist schreibgeschützt.

© 2016-2022 Altova GmbH

17.7.1.5.1.10 IsParameterSequence

Ruft ab oder definiert, ob die Input- oder Output-Parameterkomponente Sequenzen unterstützt. Diese Eigenschaft funktioniert nur bei Komponenten, die Input- oder Output-Parameterkomponenten sind.

Signatur

```
IsParameterSequence : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1233	Diese Operation funktioniert nur bei einer Input- oder Output-Parameterkomponente
1240	Eine Änderung des Dokuments ist nicht zulässig. Es ist schreibgeschützt.

17.7.1.5.1.11 Name

Ruft den Namen der Komponente ab.

Signatur

```
Name : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.1.12 OutputInstanceFile

Ruft die Output-Instanzdatei der Komponente ab oder definiert sie.

Wenn der "Datei"-Konnektor einer Komponente mit einem anderen Datenelement im Mapping verbunden wurde, erhalten Sie beim Versuch die Ausgabeinstanzdatei einer Komponente mittels `OutputInstanceFile` über die API aufzurufen, keine Daten.

Signatur

```
OutputInstanceFile : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.1.13 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Mapping
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.1.14 Preview

Ruft ab oder definiert, ob die Komponente die aktuelle Vorschaukomponente ist.

Diese Eigenschaft funktioniert nur bei Komponenten, bei denen es sich um Zielkomponenten im Hauptmapping des Dokuments handelt. Es kann im Hauptmapping immer nur eine Zielkomponente gleichzeitig als Vorschaukomponente verwendet werden.

Wenn Sie diese Eigenschaft definieren, kann sie nur auf "true" gesetzt werden. Damit wird dann die Eigenschaft `Preview` bei allen anderen Komponenten implizit auf "false" gesetzt.

© 2016-2022 Altova GmbH

Wenn es im Hauptmapping nur eine einzige Zielkomponente gibt, so ist diese auch die Vorschaukomponente.

Signatur

```
Preview : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1234	Nur eine Zielkomponente im Hauptmapping kann als Vorschaukomponente definiert werden.
1235	Eine Komponente kann nicht als Nicht-Vorschaukomponente definiert werden. Definieren Sie statt dessen eine andere Komponente als Vorschaukomponente.

17.7.1.5.1.15 Schema

Ruft den Schemadateinamen der Komponente ab.

Signatur

```
Schema : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.1.16 SubType

Ruft den Subtyp der Komponente ab.

Signatur

```
SubType : ENUMComponentSubType
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.1.17 Type

Ruft den Typ der Komponente ab.

Signatur

```
Type : ENUMComponentType
```

Allgemeine Signatur

17.7.1.5.1.18 UsageKind

Ruft die Verwendungsart der Komponente ab.

Signatur

```
UsageKind : ENUMComponentUsageKind
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.

© 2016-2022 Altova GmbH

Fehlercode	Beschreibung
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.5.2 Methoden

17.7.1.5.2.1 GenerateOutput

Generiert mit Hilfe einer MapForce-internen Mapping-Sprache die Ausgabedatei(en), die im Mapping nur für die aktuelle Komponente definiert sind. Der Name/Die Namen der Ausgabedatei(en) sind als Eigenschaft der aktuellen Komponente definiert, welche im Mapping für diese Generierung das Ausgabeelement ist.

Signatur

```
GenerateOutput(out pbError:Boolean) -> AppOutputLines
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
pbError	`Boolean`	Die ist ein Parameter, der nur für die Ausgabe verwendet wird. Sie erhalten nur dann einen Wert, wenn die aufrufende Sprache Ausgabeparameter unterstützt. Falls nicht, bleibt der hier übergebene Wert unverändert, wenn die Funktion fertig ist.

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1248	Die Ausgabegenerierung wird nur unterstützt, wenn die grafische Benutzeroberfläche angezeigt wird.

17.7.1.5.2.2 GetRootDatapoint

Ruft einen Root-Datapoint auf der linken (Input) oder rechten (Output)-Seite einer Komponente ab. Um Subelemente und untergeordnete Elemente davon aufzurufen, stellt das `Datapoint`-Objekt weitere Methoden bereit.

Signatur

```
GetRootDatapoint(in side:ENUMComponentDatapointSide, in strNamespace:String, in
strLocalName:String, in strParameterName:String) -> Datapoint
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
side	ENUMComponentDatapointSide	Der Parameter "side" gibt an, ob ein Input oder Output Datapoint einer Komponente abgerufen werden soll.
strNamespace	`String`	Der angegebene Namespace und der lokale Name geben den spezifischen Namen des Node an, dessen Datapoint abgerufen werden soll. Bei Komponenten mit Strukturinformationen wie z.B. Schemakomponenten muss der Namespace zusammen mit dem lokalen Namen angegeben werden, oder Sie übergeben einfach einen leeren String für den Namespace. Dateibasierte Komponenten wie die Schemakomponente enthalten einen speziellen Node an ihrer Root, den Dateinamen-Node. Hier findet `GetRootDatapoint` nur den Dateinamen-Node. Sie müssen den Namespace `"http://www.altova.com/mapforce"` und lokalen Namen `"FileInstance"` übergeben, um einen Datapoint dieses Node abzurufen.
strLocalName	`String`	Siehe oben.

© 2016-2022 Altova GmbH

Name	Typ	Beschreibung
strParameterName	`String`	Der angegebene Parametername sollte ein leerer String sein, es sei denn, es handelt sich bei der betreffenden Komponente um eine Funktionsaufrufskomponente. Da eine benutzerdefinierte Funktion Input- oder Output-Parameter mit derselben Struktur enthalten kann, kann die Funktionsaufrufskomponente, die diese benutzerdefinierte Funktion aufruft, mehr als einen Root-Node mit einem identischen Namespace und lokalen Namen haben. Diese unterscheiden sich dann nur anhand ihrer Parameternamen, die eigentlich die Namen der jeweiligen Parameterkomponenten im Mapping der benutzerdefinierten Funktion selbst sind. Der Parametername muss jedoch nicht definiert werden. In diesem Fall gibt die Methode den ersten Root-Datapoint, der mit dem angegebenen Namespace und lokalen Namen übereinstimmt, zurück.

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1248	Datapoint wurde nicht gefunden.

17.7.1.6 Components

Repräsentiert eine Sammlung von `Component`-Objekten.

Verwenden Sie für die Navigation im Objektmodell die folgenden Eigenschaften:

- `Application`

- Parent

Zum Iterieren durch die Sammlung:

- Count
- Item

Eigenschaften

Name	Beschreibung
Application 1802	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
Count 1803	Schreibgeschützt. Ruft die Anzahl der Komponenten in der Sammlung ab.
Item 1803	Schreibgeschützt. Ruft die Komponente am Index n aus der Sammlung ab. Indizes beginnen mit 1.
Parent 1804	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.

17.7.1.6.1 Eigenschaften

17.7.1.6.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.6.1.2 Count

Ruft die Anzahl der Komponenten in der Sammlung ab.

Signatur

```
Count : Integer
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.6.1.3 Item

Ruft die Komponente am Index `n` aus der Sammlung ab. Indizes beginnen mit 1.

Signatur

```
Item(in n:Integer) : Component
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.6.1.4 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Mapping
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.7 Connection

Ein `Connection`-Objekt steht für einen Konnektor zwischen zwei Komponenten.

Verwenden Sie für die Navigation im Objektmodell die folgenden Eigenschaften:

- `Application`
- `Parent`

Verwenden Sie zum Abrufen oder Definieren des Verbindungstyps `ConnectionType`.

Eigenschaften

Name	Beschreibung
`Application` 1805	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`ConnectionType` 1805	Ruft den Verbindungstyp ab oder definiert ihn.
`Parent` 1806	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.

© 2016-2022 Altova GmbH

17.7.1.7.1 Eigenschaften

17.7.1.7.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2100	Das Objekt ist nicht mehr gültig.
2101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.7.1.2 ConnectionType

Ruft den Verbindungstyp ab oder definiert ihn.

Signatur

```
ConnectionType : ENUMConnectionType 1916
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2100	Das Objekt ist nicht mehr gültig.
2101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
2102	Eine Änderung des Dokuments ist nicht zulässig. Es ist schreibgeschützt.
2103	Der Verbindungstyp konnte nicht geändert werden.

17.7.1.7.1.3 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Mapping
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2100	Das Objekt ist nicht mehr gültig.
2101	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.8 Datapoint

Ein `Datapoint`-Objekt repräsentiert ein Input- oder Output-Symbol einer Komponente.

Eigenschaften

Name	Beschreibung
`Application` 1807	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`Parent` 1807	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.

Methoden

Name	Beschreibung
`GetChild` 1808	Sucht nach einem direkten Child Datapoint des aktuellen Datapoint. Sucht nach Namespace und lokalem Namen. Wenn eine Schemakomponente Elemente, die mixed Content enthalten, hat, wird für jedes ein zusätzlicher Child Node, der so genannte **text()** Node, angezeigt. Um einen Datapoint eines **text()** Node abzurufen, müssen Sie in `strNamespace` einen leeren String sowie in `strLocalName` und `eSearchDatapointElement` in `searchFlags` `#text` übergeben.

© 2016-2022 Altova GmbH

17.7.1.8.1 Eigenschaften

17.7.1.8.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2000	Das Objekt ist nicht mehr gültig.
2001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.8.1.2 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Component
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2000	Das Objekt ist nicht mehr gültig.
2001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.8.2 Methoden

17.7.1.8.2.1 GetChild

Sucht nach einem direkten Child Datapoint des aktuellen Datapoint. Sucht nach Namespace und lokalem Namen.

Wenn eine Schemakomponente Elemente, die mixed Content enthalten, hat, wird für jedes ein zusätzlicher Child Node, der so genannte **text()** Node, angezeigt. Um einen Datapoint eines **text()** Node abzurufen, müssen Sie in `strNamespace` einen leeren String sowie in `strLocalName` und `eSearchDatapointElement` in `searchFlags` `#text` übergeben.

Signatur

```
GetChild(in strNamespace:String, in strLocalName:String, in
searchFlags:ENUMSearchDatapointFlags[1918]) -> Datapoint
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
strNamespace	`String`	Der Namespace des direkten Child-Datapoint.
strLocalName	`String`	Der Name direkten Child-Datapoint.
searchFlags	`ENUMSearchDatapointFlags`[1918]	Such-Flags können als Kombination von Werten der `ENUMSearchDatapointFlags` Enumeration übergeben werden (kombiniert mittels binärem OR).

Fehler

Fehlercode	Beschreibung
2000	Das Objekt ist nicht mehr gültig.
2001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
2002	Datapoint wurde nicht gefunden.

© 2016-2022 Altova GmbH

17.7.1.9 Document

Ein `Document`-Objekt repräsentiert ein MapForce-Dokument (eine geladene MFD-Datei). Ein Dokument enthält ein Hauptmapping und null oder mehr lokale benutzerdefinierte Funktionsmappings.

Verwenden Sie für die Navigation im Objektmodell die folgenden Eigenschaften:

- `Application`
- `Parent`

Verwenden Sie zur Behandlung von Dateien:

- `Activate`
- `Close`
- `FullName`
- `Name`
- `Path`
- `Saved`
- `Save`
- `SaveAs`

Verwenden Sie zur Behandlung des Mappings:

- `MainMapping`
- `Mappings`
- `CreateUserDefinedFunction`

Verwenden Sie zur Behandlung von Komponenten:

- `FindComponentByID`

Verwenden Sie zur Codegenerierung:

- `OutputSettings_ApplicationName`
- `JavaSettings_BasePackageName`
- `GenerateCHashCode`
- `GenerateCodeEx`
- `GenerateCppCode`
- `GenerateJavaCode`
- `GenerateXQuery`
- `GenerateXSLT`
- `GenerateXSLT2`
- `GenerateXSLT3`
- `HighlightSerializedMarker`

Verwenden Sie zur Ausführung von Mappings:

- `GenerateOutput`
- `GenerateOutputEx`

Aufruf der Ansicht:

- `MapForceView`

Veraltet:

- `OutputSettings_Encoding`

Eigenschaften

Name	Beschreibung
`Application` 1813	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`FullName` 1813	Pfad und Name der Dokumentdatei.
`JavaSettings_BasePackageName` 1814	Setzt den beim Generieren von Java-Code verwendeten Base Package-Namen oder ruft ihn ab. Auf der grafischen MapForce-Benutzeroberfläche steht diese Eigenschaft im Dialogfeld **Mapping-Einstellungen** (Aufruf durch Rechtsklick in das Mapping und Auswahl des Kontextmenüeintrags **Mapping-Einstellungen** zur Verfügung).
`LibraryImports` 1814	Schreibgeschützt. Ruft eine Sammlung importierter Bibliotheken ab. Diese entsprechen auf der grafischen MapForce-Benutzeroberfläche den auf Dokumentebene hinzugefügten Einträgen im Fenster **Bibliotheken verwalten**.
`MainMapping` 1815	Schreibgeschützt. Ruft das Hauptmapping des Dokuments auf.
`MapForceView` 1815	Schreibgeschützt. Mit dieser Eigenschaft erhalten Sie Zugriff auf die Funktionalitäten der MapForce-Ansicht.
`Mappings` 1816	Schreibgeschützt. Gibt eine Sammlung der im Dokument enthaltenen Mappings zurück.
`Name` 1816	Schreibgeschützt. Name der Dokumentdatei ohne Dateipfad.
`OutputSettings_ApplicationName` 1816	Setzt den im Dialogfeld **Mapping-Einstellungen** verfügbaren Applikationsnamen bzw. ruft diesen ab. (Um dieses Dialogfeld in MapForce aufzurufen, klicken Sie mit der rechten Maustaste in das Mapping und wählen Sie im Kontextmenü den Befehl **Mapping-Einstellungen**).
`OutputSettings_Encoding` 1817	Diese Eigenschaft wird nicht mehr unterstützt. Es gibt keine Kodierungseinstellungen mehr für die Mapping-Ausgabe. Die Kodierungseinstellungen werden jeweils für die einzelnen Komponenten festgelegt.
`Parent` 1817	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.

© 2016-2022 Altova GmbH

Name	Beschreibung
Path 1818	Schreibgeschützt. Pfad der Dokumentdatei ohne Namen.
Saved 1818	Schreibgeschützt. **True,**wenn das Dokument seit dem letzten Speichern nicht geändert wurde, andernfalls **false**.

Methoden

Name	Beschreibung
Activate 1819	Macht dieses Dokument zum aktiven Dokument.
Close 1819	Schließt das Dokument, ohne es zu speichern.
CreateUserDefinedFunction 1820	Erstellt eine benutzerdefinierte Funktion im aktuellen Dokument.
FindComponentByID 1821	Durchsucht das gesamte Dokument und alle seine Mappings nach der Komponente mit der angegebenen ID.
GenerateCHashCode 1821	Generiert C#-Code, der das Mapping ausführt. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.
GenerateCodeEx 1822	Generiert Code, der das Mapping ausführt. Der Parameter **i_nLanguage** definiert die Zielsprache. Die Methode gibt ein Objekt zurück, das verwendet werden kann, um alle vom Codegenerator erstellten Meldungen aufzuzählen. Dabei handelt es sich um dieselben Meldungen, die im Meldungsfenster von MapForce angezeigt werden.
GenerateCppCode 1822	Generiert C++-Code, der das Mapping ausführt. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.
GenerateJavaCode 1823	Generiert Java-Code, der das Mapping ausführt. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.
GenerateOutput 1823	Generiert alle im Mapping definierten Ausgabedateien mittels einer MapForce-internen Mapping-Sprache. Die Namen der Ausgabedateien werden als Eigenschaften der Ausgabeelemente im Mapping definiert. **Anmerkung:** Diese Methode kann nur verwendet werden, wenn das (als COM-Server ausgeführte) MapForce-Hauptfenster zu sehen ist oder in eine grafische Benutzeroberfläche eingebettet ist. Wenn die Methode aufgerufen wird, während MapForce nicht angezeigt wird, wird ein Fehler ausgegeben.

Name	Beschreibung
GenerateOutputEx 1824	Generiert alle im Mapping definierten Ausgabedateien mittels einer MapForce-internen Mapping-Sprache. Die Namen der Ausgabedateien werden als Eigenschaften der Ausgabeelemente im Mapping definiert. Diese Methode ist mit Ausnahme des Rückgabewerts, der die erzeugten Meldungen, Warnmeldungen und Fehler in Form einer Baumstruktur von `AppOutputLines` enthält, identisch mit `GenerateOutput`. **Anmerkung:** Diese Methode kann nur verwendet werden, wenn das (als COM-Server ausgeführte) MapForce-Hauptfenster zu sehen ist oder in eine grafische Benutzeroberfläche eingebettet ist. Wenn die Methode aufgerufen wird, während MapForce nicht angezeigt wird, wird ein Fehler ausgegeben.
GenerateXQuery 1824	Generiert Mappingcode als XQuery. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.
GenerateXSLT 1825	Generiert Mappingcode als XSLT. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.
GenerateXSLT2 1825	Generiert Mappingcode als XSLT2. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.
GenerateXSLT3 1826	Generiert XSLT 3.0 Mapping-Code. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.
HighlightSerializedMarker 1826	Verwenden Sie diese Methode, um einen Pfad in einer Mapping-Datei zu markieren, die zuvor serialisiert wurde. Wenn das entsprechende Dokument noch nicht geladen ist, wird es zuerst geladen. Eine Methode zum Aufrufen eines serialisierten Markers finden Sie unter `GenerateCodeEx`.
Save 1827	Speichert das Dokument in der durch `Document.FullName` definierten Datei.
SaveAs 1827	Speichert das Dokument unter dem angegebenen Dateinamen und setzt `Document.FullName` auf diesen Wert, wenn die Operation erfolgreich war.

Events

Name	Beschreibung
OnDocumentClosed 1828	Dieses Event wird ausgelöst, wenn ein Dokument geschlossen wird. Das an den Event Handler übergebene Dokumentobjekt sollte nicht aufgerufen werden. Das entsprechenden Event zum Öffnen ist `Application.OnDocumentOpened`.

© 2016-2022 Altova GmbH

Name	Beschreibung
OnModifiedFlagChanged 1828	Dieses Event wird ausgelöst, wenn sich der Änderungsstatus eines Dokuments ändert.

17.7.1.9.1 Eigenschaften

17.7.1.9.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.1.2 FullName

Pfad und Name der Dokumentdatei.

Signatur

```
FullName : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.1.3 JavaSettings_BasePackageName

Setzt den beim Generieren von Java-Code verwendeten Base Package-Namen oder ruft ihn ab. Auf der grafischen MapForce-Benutzeroberfläche steht diese Eigenschaft im Dialogfeld **Mapping-Einstellungen** (Aufruf durch Rechtsklick in das Mapping und Auswahl des Kontextmenüeintrags **Mapping-Einstellungen** zur Verfügung).

Signatur

```
JavaSettings_BasePackageName : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.1.4 LibraryImports

Ruft eine Sammlung importierter Bibliotheken ab. Diese entsprechen auf der grafischen MapForce-Benutzeroberfläche den auf Dokumentebene hinzugefügten Einträgen im Fenster **Bibliotheken verwalten**.

Signatur

```
LibraryImports : LibraryImports
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.9.1.5 MainMapping

Ruft das Hauptmapping des Dokuments auf.

Signatur

```
MainMapping : Mapping
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.1.6 MapForceView

Mit dieser Eigenschaft erhalten Sie Zugriff auf die Funktionalitäten der MapForce-Ansicht.

Signatur

```
MapForceView : MapForceView
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.1.7 Mappings

Gibt eine Sammlung der im Dokument enthaltenen Mappings zurück.

Signatur

```
Mappings : Mappings
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.1.8 Name

Name der Dokumentdatei ohne Dateipfad.

Signatur

```
Name : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.1.9 OutputSettings_ApplicationName

Setzt den im Dialogfeld **Mapping-Einstellungen** verfügbaren Applikationsnamen bzw. ruft diesen ab. (Um dieses Dialogfeld in MapForce aufzurufen, klicken Sie mit der rechten Maustaste in das Mapping und wählen Sie im Kontextmenü den Befehl **Mapping-Einstellungen**).

© 2016-2022 Altova GmbH

Signatur

```
OutputSettings_ApplicationName : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.1.10 OutputSettings_Encoding (obsolete)

Diese Eigenschaft wird nicht mehr unterstützt. Es gibt keine Kodierungseinstellungen mehr für die Mapping-Ausgabe. Die Kodierungseinstellungen werden jeweils für die einzelnen Komponenten festgelegt.

Signatur

```
OutputSettings_Encoding : String
```

Allgemeine Signatur

17.7.1.9.1.11 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Documents
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.1.12 Path

Pfad der Dokumentdatei ohne Namen.

Signatur

```
Path : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.1.13 Saved

True,wenn das Dokument seit dem letzten Speichern nicht geändert wurde, andernfalls **false**.

Signatur

```
Saved : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.9.2 Methoden

17.7.1.9.2.1 Activate

Macht dieses Dokument zum aktiven Dokument.

Signatur

```
Activate() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.

17.7.1.9.2.2 Close

Schließt das Dokument, ohne es zu speichern.

Signatur

```
Close() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.2.3 CreateUserDefinedFunction

Erstellt eine benutzerdefinierte Funktion im aktuellen Dokument.

Signatur

```
CreateUserDefinedFunction(in strFunctionName:String, in strLibraryName:String, in
strSyntax:String, in strDetails:String, in bInlinedUse:Boolean) -> Mapping
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
strFunctionName	`String`	Der Name der Funktion.
strLibraryName	`String`	Der Name der Bibliothek, zu der diese Funktion gehört.
strSyntax	`String`	Ein String, der die Syntax dieser Funktion beschreibt (dies dient nur zu Informationszwecken).
strDetails	`String`	Eine Beschreibung dieser Funktion.
bInlinedUse	`Boolean`	Boolesches Flag, das angibt, ob die Funktion eine inline-Verwendung aufweist.

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1208	Benutzerdefinierte Funktion konnte nicht erstellt werden.
1209	Eine Änderung des Dokuments ist nicht zulässig. Es ist schreibgeschützt.

© 2016-2022 Altova GmbH

17.7.1.9.2.4 FindComponentByID

Durchsucht das gesamte Dokument und alle seine Mappings nach der Komponente mit der angegebenen ID.

Signatur

```
FindComponentByID(in nID:Unsigned Long) -> Component
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
nID	`Unsigned Long`	Die ID der zu suchenden Komponente.

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.2.5 GenerateCHashCode

Generiert C#-Code, der das Mapping ausführt. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.

Signatur

```
GenerateCHashCode() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1205	Fehler bei der Codegenerierung.

17.7.1.9.2.6 GenerateCodeEx

Generiert Code, der das Mapping ausführt. Der Parameter **i_nLanguage** definiert die Zielsprache. Die Methode gibt ein Objekt zurück, das verwendet werden kann, um alle vom Codegenerator erstellten Meldungen aufzuzählen. Dabei handelt es sich um dieselben Meldungen, die im Meldungsfenster von MapForce angezeigt werden.

Signatur

```
GenerateCodeEx(in i_nLanguage:ENUMProgrammingLanguage[1916]) -> ErrorMarkers
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_nLanguage	ENUMProgrammingLanguage[1916]	Definiert die Zielsprache für den zu generierenden Code.

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1205	Fehler bei der Codegenerierung.

17.7.1.9.2.7 GenerateCppCode

Generiert C++-Code, der das Mapping ausführt. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.

Signatur

```
GenerateCppCode() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.

© 2016-2022 Altova GmbH

Fehlercode	Beschreibung
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1205	Fehler bei der Codegenerierung.

17.7.1.9.2.8 GenerateJavaCode

Generiert Java-Code, der das Mapping ausführt. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.

Signatur

```
GenerateJavaCode() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1205	Fehler bei der Codegenerierung.

17.7.1.9.2.9 GenerateOutput

Generiert alle im Mapping definierten Ausgabedateien mittels einer MapForce-internen Mapping-Sprache. Die Namen der Ausgabedateien werden als Eigenschaften der Ausgabeelemente im Mapping definiert.

Anmerkung: Diese Methode kann nur verwendet werden, wenn das (als COM-Server ausgeführte) MapForce-Hauptfenster zu sehen ist oder in eine grafische Benutzeroberfläche eingebettet ist. Wenn die Methode aufgerufen wird, während MapForce nicht angezeigt wird, wird ein Fehler ausgegeben.

Signatur

```
GenerateOutput() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1206	Fehler bei der Ausführung eines Mapping-Algorithmus.
1210	Die Ausgabegenerierung wird nur unterstützt, wenn die grafische Benutzeroberfläche angezeigt wird.

17.7.1.9.2.10 GenerateOutputEx

Generiert alle im Mapping definierten Ausgabedateien mittels einer MapForce-internen Mapping-Sprache. Die Namen der Ausgabedateien werden als Eigenschaften der Ausgabeelemente im Mapping definiert. Diese Methode ist mit Ausnahme des Rückgabewerts, der die erzeugten Meldungen, Warnmeldungen und Fehler in Form einer Baumstruktur von `AppOutputLines` enthält, identisch mit `GenerateOutput`.

Anmerkung: Diese Methode kann nur verwendet werden, wenn das (als COM-Server ausgeführte) MapForce-Hauptfenster zu sehen ist oder in eine grafische Benutzeroberfläche eingebettet ist. Wenn die Methode aufgerufen wird, während MapForce nicht angezeigt wird, wird ein Fehler ausgegeben.

Signatur

```
GenerateOutputEx() -> AppOutputLines
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1206	Fehler bei der Ausführung eines Mapping-Algorithmus.
1210	Die Ausgabegenerierung wird nur unterstützt, wenn die grafische Benutzeroberfläche angezeigt wird.

17.7.1.9.2.11 GenerateXQuery

Generiert Mappingcode als XQuery. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.

© 2016-2022 Altova GmbH

Signatur

```
GenerateXQuery() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1204	Fehler bei der XSLT/XSLT2/XSLT3/XQuery-Codegenerierung.

17.7.1.9.2.12 GenerateXSLT

Generiert Mappingcode als XSLT. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.

Signatur

```
GenerateXSLT() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1204	Fehler bei der XSLT/XSLT2/XSLT3/XQuery-Codegenerierung.

17.7.1.9.2.13 GenerateXSLT2

Generiert Mappingcode als XSLT2. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.

Signatur

```
GenerateXSLT2() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1204	Fehler bei der XSLT/XSLT2/XSLT3/XQuery-Codegenerierung.

17.7.1.9.2.14 GenerateXSLT3

Generiert XSLT 3.0 Mapping-Code. Verwendet zur Konfigurierung der Codegenerierung die in `Application.Options` definierten Eigenschaften.

Signatur

```
GenerateXSLT3() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1204	Fehler bei der XSLT/XSLT2/XSLT3/XQuery-Codegenerierung.

17.7.1.9.2.15 HighlightSerializedMarker

Verwenden Sie diese Methode, um einen Pfad in einer Mapping-Datei zu markieren, die zuvor serialisiert wurde. Wenn das entsprechende Dokument noch nicht geladen ist, wird es zuerst geladen. Eine Methode zum Aufrufen eines serialisierten Markers finden Sie unter `GenerateCodeEx`.

Signatur

```
HighlightSerializedMarker(in i_strSerializedMarker:String) -> Void
```

Allgemeine Signatur

© 2016-2022 Altova GmbH

Parameter

Name	Typ	Beschreibung
i_strSerializedMarker	`String`	Das zu markierende `ErrorMarker`-Objekt. Diesen Wert erhalten Sie mit Hilfe von `ErrorMaker.Serialized`.

Fehler

Fehlercode	Beschreibung
1000	Das Objekt ist nicht mehr gültig.
1001	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1007	Der in `i_strSerializedMarker` übergebene String wird nicht als serialisierter MapForce Marker erkannt.
1008	Der Marker verweist auf einen nicht mehr gültigen Pfad.

17.7.1.9.2.16 Save

Speichert das Dokument in der durch `Document.FullName` definierten Datei.

Signatur

```
Save() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.2.17 SaveAs

Speichert das Dokument unter dem angegebenen Dateinamen und setzt `Document.FullName` auf diesen Wert, wenn die Operation erfolgreich war.

Signatur

```
SaveAs(in i_strFileName:String) -> Void
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFileName	`String`	Definiert den Pfad, unter dem das Dokument gespeichert werden soll.

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.9.3 Events

17.7.1.9.3.1 OnDocumentClosed

Dieses Event wird ausgelöst, wenn ein Dokument geschlossen wird. Das an den Event Handler übergebene Dokumentobjekt sollte nicht aufgerufen werden. Das entsprechenden Event zum Öffnen ist `Application.OnDocumentOpened`.

Signatur

```
OnDocumentClosed(in i_ipDocument:Document) : Void
```

17.7.1.9.3.2 OnModifiedFlagChanged

Dieses Event wird ausgelöst, wenn sich der Änderungsstatus eines Dokuments ändert.

Signatur

```
OnModifiedFlagChanged(in i_bIsModified:Boolean) : Void
```


© 2016-2022 Altova GmbH

17.7.1.10 Documents

Repräsentiert eine Sammlung von `Document`-Objekten.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Öffnen und Erstellen von Mappings:

- `OpenDocument`
- `NewDocument`

Iterieren durch die Sammlung:

- `Count`
- `Item`
- `ActiveDocument`

Eigenschaften

Name	Beschreibung
`ActiveDocument` 1830	Schreibgeschützt. Ruft das aktive Dokument ab. Wenn kein Dokument offen ist, wird `null` zurückgegeben.
`Application` 1830	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`Count` 1831	Schreibgeschützt. Ruft die Anzahl der Dokumente in der Sammlung ab.
`Item` 1831	Schreibgeschützt. Ruft das Dokument am Index `n` aus der Sammlung ab. Indizes beginnen mit 1.
`Parent` 1832	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.

Methoden

Name	Beschreibung
`NewDocument` 1832	Erstellt ein neues Dokument, fügt es zum Ende der Sammlung hinzu und machte es zum aktiven Dokument.
`OpenDocument` 1832	Öffnet ein vorhandenes Mapping-Dokument (*.mfd). Fügt das neu geöffnete Dokument zum Ende der Sammlung hinzu und macht es zum aktiven Dokument.

17.7.1.10.1 Eigenschaften

17.7.1.10.1.1 ActiveDocument

Ruft das aktive Dokument ab. Wenn kein Dokument offen ist, wird `null` zurückgegeben.

Signatur

```
ActiveDocument : Document
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1600	Das Objekt ist nicht mehr gültig.
1601	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.10.1.2 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1600	Das Objekt ist nicht mehr gültig.
1601	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.10.1.3 Count

Ruft die Anzahl der Dokumente in der Sammlung ab.

Signatur

```
Count : Integer
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1600	Das Objekt ist nicht mehr gültig.
1601	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.10.1.4 Item

Ruft das Dokument am Index `n` aus der Sammlung ab. Indizes beginnen mit 1.

Signatur

```
Item(in n:Integer) : Document
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1600	Das Objekt ist nicht mehr gültig.
1601	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.10.1.5 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1600	Das Objekt ist nicht mehr gültig.
1601	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.10.2 Methoden

17.7.1.10.2.1 NewDocument

Erstellt ein neues Dokument, fügt es zum Ende der Sammlung hinzu und machte es zum aktiven Dokument.

Signatur

```
NewDocument() -> Document
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1600	Das Objekt ist nicht mehr gültig.
1601	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.10.2.2 OpenDocument

Öffnet ein vorhandenes Mapping-Dokument (*.mfd). Fügt das neu geöffnete Dokument zum Ende der Sammlung hinzu und macht es zum aktiven Dokument.

© 2016-2022 Altova GmbH

Signatur

```
OpenDocument(in strPath:String) -> Document
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
strPath	`String`	Der Pfad zur Mapping-Datei.

Fehler

Fehlercode	Beschreibung
1600	Das Objekt ist nicht mehr gültig.
1601	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.11 ErrorMarker

Repräsentiert eine einfache Meldungszeile. Im Unterschied zu `AppOutputLine` haben errorMarkers keine hierarchische Struktur.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Zugriff auf die Meldungsinformationen:

- `DocumentFileName`
- `ErrorLevel`
- `Highlight`
- `Serialization`
- `Text`

Eigenschaften

Name	Beschreibung
`Application` 1834	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`DocumentFileName` 1835	Schreibgeschützt. Ruft den Namen der Mapping-Datei ab, mit der der Fehler-Marker verknüpft ist.

Name	Beschreibung
ErrorLevel 1835	Schreibgeschützt. Ruft den Schweregrad des Fehlers ab.
Parent 1836	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.
Serialization 1836	Schreibgeschützt. Serialisiert den Fehlermarker in einen String. Verwenden Sie diesen String in Aufrufen von `Application.HighlightSerializedMarker` oder `Document.HighlightSerializedMarker` zum Hervorheben des markierten Elements im Mapping. Der String kann in anderen Instantiierungen von MapForce oder seinem Control weiterverwendet werden.
Text 1837	Schreibgeschützt. Ruft den Meldungstext ab.

Methoden

Name	Beschreibung
Highlight 1837	Markiert das Element, mit dem der Fehler-Marker verknüpft ist. Wenn das entsprechende Dokument nicht offen ist, wird es geöffnet.

17.7.1.11.1 Eigenschaften

17.7.1.11.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1900	Das Objekt ist nicht mehr gültig.
1901	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.11.1.2 DocumentFileName

Ruft den Namen der Mapping-Datei ab, mit der der Fehler-Marker verknüpft ist.

Signatur

```
DocumentFileName : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1900	Das Objekt ist nicht mehr gültig.
1901	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.11.1.3 ErrorLevel

Ruft den Schweregrad des Fehlers ab.

Signatur

```
ErrorLevel : ENUMCodeGenErrorLevel 1914
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1900	Das Objekt ist nicht mehr gültig.
1901	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.11.1.4 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : ErrorMarkers
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1900	Das Objekt ist nicht mehr gültig.
1901	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.11.1.5 Serialization

Serialisiert den Fehlermarker in einen String. Verwenden Sie diesen String in Aufrufen von `Application.HighlightSerializedMarker` oder `Document.HighlightSerializedMarker` zum Hervorheben des markierten Elements im Mapping. Der String kann in anderen Instantiierungen von MapForce oder seinem Control weiterverwendet werden.

Signatur

```
Serialization : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1900	Das Objekt ist nicht mehr gültig.
1901	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.11.1.6 Text

Ruft den Meldungstext ab.

Signatur

```
Text : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1900	Das Objekt ist nicht mehr gültig.
1901	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.11.2 Methoden

17.7.1.11.2.1 Highlight

Markiert das Element, mit dem der Fehler-Marker verknüpft ist. Wenn das entsprechende Dokument nicht offen ist, wird es geöffnet.

Signatur

```
Highlight() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1900	Das Objekt ist nicht mehr gültig.
1901	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1008	Der Marker verweist auf einen nicht mehr gültigen Pfad.

17.7.1.12 ErrorMarkers

Repräsentiert eine Sammlung von `ErrorMarker`-Objekten.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Iterierend durch die Sammlung:

- `Count`
- `Item`

Eigenschaften

Name	Beschreibung
`Application` 1838	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`Count` 1839	Schreibgeschützt. Ruft die Anzahl der Fehler-Marker in der Sammlung auf.
`Item` 1839	Schreibgeschützt. Ruft den Fehler-Marker am Index `n` aus der Sammlung ab. Indizes beginnen mit 1.
`Parent` 1840	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.

17.7.1.12.1 Eigenschaften

17.7.1.12.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
1800	Das Objekt ist nicht mehr gültig.
1801	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.12.1.2 Count

Ruft die Anzahl der Fehler-Marker in der Sammlung auf.

Signatur

```
Count : Integer
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1800	Das Objekt ist nicht mehr gültig.
1801	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.12.1.3 Item

Ruft den Fehler-Marker am Index `n` aus der Sammlung ab. Indizes beginnen mit 1.

Signatur

```
Item(in n:Integer) : ErrorMarker
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1800	Das Objekt ist nicht mehr gültig.
1801	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.12.1.4 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1800	Das Objekt ist nicht mehr gültig.
1801	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.13 LibraryImport

Ein `LibraryImport`-Objekt repräsentierte eine importierte Bibliotheksdatei (einen Eintrag im Fenster **Bibliotheken verwalten**).

Eigenschaften

Name	Beschreibung
`Application` 1841	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`Parent` 1841	Schreibgeschützt. Ruft das Parent-Objekt laut Objektmodell auf.
`Path` 1842	Schreibgeschützt. Ruft den Pfad der importierten Bibliothek auf.
`SaveRelativePath` 1842	Wenn Sie das Dokument speichern, gibt diese Eigenschaft an, ob der Bibliothekspfad als absoluter oder relativer Pfad gespeichert werden soll. Bei **true** ist der Pfad der Bibliothek relativ zum Dokument. Bei **false** ist der Bibliothekspfad absolut. Verwenden Sie diese Eigenschaft nicht, um zu ermittlen, ob der Pfad absolut oder relativ ist, da der Pfad möglicherweise geändert wurde, seitdem das Dokument (entweder über die Benutzeroberfläche oder über die API) aus der .mfd-Datei geladen wurde.

© 2016-2022 Altova GmbH

Name	Beschreibung
	Wenn Sie diese Eigenschaft (entweder über die API oder die Benutzeroberfläche) definieren, wird auf der Benutzeroberfläche im Fenster "Bibliotheken verwalten" sofort der korrekte Status des Pfads angezeigt. Intern ändert sich der `Path` des Objekts `ImportedLibrary` jedoch erst, wenn das Dokument gespeichert wird. Global importierte Bibliotheken können nicht mit einem relativen Pfad gespeichert werden. Dies ist nur bei auf Dokumentebene importierten Bibliotheken möglich.

17.7.1.13.1 Eigenschaften

17.7.1.13.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2500	Das Objekt ist nicht mehr gültig.
2501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.13.1.2 Parent

Ruft das Parent-Objekt laut Objektmodell auf.

Signatur

```
Parent : LibraryImports
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2500	Das Objekt ist nicht mehr gültig.
2501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.13.1.3 Path

Ruft den Pfad der importierten Bibliothek auf.

Signatur

```
Path : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2500	Das Objekt ist nicht mehr gültig.
2501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.13.1.4 SaveRelativePath

Wenn Sie das Dokument speichern, gibt diese Eigenschaft an, ob der Bibliothekspfad als absoluter oder relativer Pfad gespeichert werden soll. Bei **true** ist der Pfad der Bibliothek relativ zum Dokument. Bei **false** ist der Bibliothekspfad absolut.

Verwenden Sie diese Eigenschaft nicht, um zu ermittlen, ob der Pfad absolut oder relativ ist, da der Pfad möglicherweise geändert wurde, seitdem das Dokument (entweder über die Benutzeroberfläche oder über die API) aus der .mfd-Datei geladen wurde.

Wenn Sie diese Eigenschaft (entweder über die API oder die Benutzeroberfläche) definieren, wird auf der Benutzeroberfläche im Fenster "Bibliotheken verwalten" sofort der korrekte Status des Pfads angezeigt. Intern ändert sich der `Path` des Objekts `ImportedLibrary` jedoch erst, wenn das Dokument gespeichert wird.

> Global importierte Bibliotheken können nicht mit einem relativen Pfad gespeichert werden. Dies ist nur bei auf Dokumentebene importierten Bibliotheken möglich.

© 2016-2022 Altova GmbH

Signatur

```
SaveRelativePath : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2500	Das Objekt ist nicht mehr gültig.
2501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
2502	Global importierte Bibliotheken können nicht mit einem relativen Pfad gespeichert werden.

17.7.1.14 LibraryImports

Repräsentiert eine Sammlung importierter Bibliotheken (`LibraryImport`-Objekte). Mit Hilfe der Eigenschaften `Application` und `Parent` können Sie im Objektmodell navigieren. Mit Hilfe der Eigenschaften `Count` und `Item` können Sie durch die Sammlung iterieren. Sie können diese Sammlung folgendermaßen abrufen:

- lokal (auf Dokumenebene) über die Eigenschaft `Document.LibraryImports`
- global (auf Applikationsebene) über die Eigenschaft `Application.LibraryImports`.

> Wenn Sie die Sammlung `LibraryImports` über das Applikationsobjekt abrufen, ist die Eigenschaft `Parent` der Sammlung Null.

Eigenschaften

Name	Beschreibung
`Application` 1844	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`Count` 1845	Schreibgeschützt. Ruft die Anzahl der `LibraryImport`-Objekte in dieser Sammlung ab.
`Item` 1845	Schreibgeschützt. Ruft einen Bibliothekseintrag am Index *n* dieser Sammlung ab. Der Index ist 1-basiert.
`Parent` 1845	Schreibgeschützt. Ruft das Parent-Dokument für lokale Bibliotheksimporte ab. Wenn Sie die Sammlung `LibraryImports` über das

Name	Beschreibung
	Applikationsobjekt abrufen, ist die Eigenschaft `Parent` der Sammlung Null.

Methoden

Name	Beschreibung
`Add` 1846	Fügt eine neue Bibliothek zu diesem `LibraryImports`-Objekt hinzu. Die neue Bibliothek erhält den durch den Parameter **i_strFileName** angegebenen Pfad.
`Find` 1847	Gibt eine Bibliotheksreferenz mit dem Pfad der Bibliotheksdatei zurück.
`Remove` 1848	Entfernt eine Bibliotheksreferenz aus dem Fenster **Bibliotheken verwalten**.

17.7.1.14.1 Eigenschaften

17.7.1.14.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2400	Das Objekt ist nicht mehr gültig.
2401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.14.1.2 Count

Ruft die Anzahl der `LibraryImport`-Objekte in dieser Sammlung ab.

Signatur

```
Count : Integer
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2400	Das Objekt ist nicht mehr gültig.
2401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.14.1.3 Item

Ruft einen Bibliothekseintrag am Index *n* dieser Sammlung ab. Der Index ist 1-basiert.

Signatur

```
Item(in n:Integer) : LibraryImport
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2400	Das Objekt ist nicht mehr gültig.
2401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.14.1.4 Parent

Ruft das Parent-Dokument für lokale Bibliotheksimporte ab. Wenn Sie die Sammlung `LibraryImports` über das Applikationsobjekt abrufen, ist die Eigenschaft `Parent` der Sammlung Null.

Signatur

```
Parent : Document
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
2400	Das Objekt ist nicht mehr gültig.
2401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.14.2 Methoden

17.7.1.14.2.1 Add

Fügt eine neue Bibliothek zu diesem `LibraryImports`-Objekt hinzu. Die neue Bibliothek erhält den durch den Parameter **i_strFileName** angegebenen Pfad.

Signatur

```
Add(in i_strFileName:String) -> LibraryImport
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFileName	`String`	Definiert den Pfad der Bibliotheksdatei. Dieser Pfad kann, je nach Status, in dem er an das Objekt übergeben wurde, entweder absolut oder relativ zum Mapping sein. Wenn ein Dokument gespeichert wird, wird der Pfad relativ gemacht, wenn das Flag `LibraryImport.SaveRelativePath` **true** ist; andernfalls wird er absolut gemacht.

Fehler

Fehlercode	Beschreibung
2400	Das Objekt ist nicht mehr gültig.

© 2016-2022 Altova GmbH

Fehlercode	Beschreibung
2401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
2402	Die Bibliothek konnte nicht hinzugefügt werden.

17.7.1.14.2.2 Find

Gibt eine Bibliotheksreferenz mit dem Pfad der Bibliotheksdatei zurück.

Signatur

```
Find(in i_strFileName:String) -> LibraryImport
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFileName	`String`	Der Pfad der zu suchenden Bibliotheksdatei. Bei lokal importierten Bibliotheken können Sie entweder den absoluten oder den relativen Pfad zur Bibliotheksdatei definieren (Im Gegensatz dazu muss bei der `Remove`-Methode der exakte Pfad angegeben werden). Bei global importierten Bibliotheken muss der Pfad immer absolut sein (da global importierte Bibliotheken keinen relativen Pfad haben können).

Fehler

Fehlercode	Beschreibung
2400	Das Objekt ist nicht mehr gültig.
2401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.14.2.3 Remove

Entfernt eine Bibliotheksreferenz aus dem Fenster **Bibliotheken verwalten**.

Signatur

```
Remove(in i_strFileName:String) -> Void
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFileName	`String`	Der Pfad der zu entfernenden Bibliotheksdatei. Beachten Sie, dass der Pfad genau dem aktuellen (neuesten) Status des `LibraryImport`-Objekts entsprechen muss. Denken Sie daran, dass der Pfad entweder relativ oder absolut sein kann und sich eventuell je nach `LibraryImport.SaveRelativePath`-Flag geändert hat, wenn Sie das Dokument gespeichert haben. Wenn das Objekt `LibraryImport` daher derzeit einen relativen Pfad enthält, sollten Sie einen relativen Pfad als Wert dieses Parameters angeben. Andernfalls wird die Bibliothek nicht gefunden und die Methode `Remove` schlägt fehl. Obige Anmerkung gilt nur für lokal importierte Bibliotheken. Bei global importierten Bibliotheken muss der Pfad immer absolut sein (da global importierte Bibliotheken keinen relativen Pfad haben können).

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
2400	Das Objekt ist nicht mehr gültig.

17.7.1.15 MapForceView

Repräsentiert die aktuelle Ansicht eines Dokument auf dem MapForce-Register "Mapping". Ein Dokument hat genau eine MapForce-Ansicht (`MapForceView`), in der das aktuell aktive Mapping angezeigt wird.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Aktivierung der Ansicht und Ansichtseigenschaften:

- `Active`
- `ShowItemTypes`
- `ShowLibraryInFunctionHeader`
- `HighlightMyConnections`
- `HighlightMyConnectionsRecursivly`

Eigenschaften im Zusammenhang mit dem Mapping:

- `ActiveMapping`
- `ActiveMappingName`

Hinzufügen von Datenelementen:

- `InsertWSDLCall`
- `InsertXMLFile`
- `InsertXMLSchema`
- `InsertXMLSchemaWithSample`

Eigenschaften

Name	Beschreibung
`Active` 1850	Mit Hilfe dieser Eigenschaft können Sie abfragen, ob die Mapping-Ansicht die aktive Ansicht ist oder Sie können diese Ansicht zur aktiven machen.
`ActiveMapping` 1851	Ruft das aktuell aktive Mapping im Dokument auf, zu dem diese MapForce-Ansicht (`MapForceView`) gehört, bzw. definiert es.
`ActiveMappingName` 1851	Ruft das aktuell aktive Mapping nach seinem Namen im Dokument, zu dem diese MapForce-Ansicht (MapForceView) gehört, ab bzw. definiert es.

Name	Beschreibung
Application 1852	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
HighlightMyConnections 1852	Mit dieser Eigenschaft wird definiert, ob nur Verbindungen vom ausgewählten Element aus markiert werden sollen.
HighlightMyConnectionsRecursively 1853	Mit dieser Eigenschaft wird definiert, ob nur die Verbindungen, die direkt von und zu dem ausgewählten Datenelement führen, markiert werden sollen.
Parent 1853	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.
ShowItemTypes 1854	Mit dieser Eigenschaft wird definiert, ob Elementtypen im Mapping-Diagramm angezeigt werden sollen.
ShowLibraryInFunctionHeader 1854	Mit dieser Eigenschaft wird definiert, ob der Name der Funktionsbibliothek Teil von Funktionsnamen sein soll.

Methoden

Name	Beschreibung
InsertWSDLCall 1855	Fügt eine neue WSDL Call-Komponente zum Mapping hinzu.
InsertXMLFile 1855	`MapForceView.InsertXMLFile` wird nicht mehr verwendet. Verwenden Sie stattdessen `Mapping.InsertXMLFile`.
InsertXMLSchema 1856	`MapForceView.InsertXMLSchema` wird nicht mehr verwendet. Verwenden Sie stattdessen `Mapping.InsertXMLSchema`.
InsertXMLSchemaWithSample 1856	`MapForceView.InsertXMLSchemaWithSample` wird nicht mehr verwendet. Verwenden Sie stattdessen `Mapping.InsertXMLFile`. Beachten Sie, dass zur Übergabe des Root-Elements kein Parameter für `Mapping.InsertXMLFile` erforderlich ist. Das Root-Element wird automatisch als Root-Elementname der XML-Datei definiert.

17.7.1.15.1 Eigenschaften

17.7.1.15.1.1 Active

Mit Hilfe dieser Eigenschaft können Sie abfragen, ob die Mapping-Ansicht die aktive Ansicht ist oder Sie können diese Ansicht zur aktiven machen.

Signatur

```
Active : Boolean
```


© 2016-2022 Altova GmbH

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1300	Das Objekt ist nicht mehr gültig.
1301	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.15.1.2 ActiveMapping

Ruft das aktuell aktive Mapping im Dokument auf, zu dem diese MapForce-Ansicht (`MapForceView`) gehört, bzw. definiert es.

Signatur

```
ActiveMapping : Mapping
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1300	Das Objekt ist nicht mehr gültig.
1301	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.15.1.3 ActiveMappingName

Ruft das aktuell aktive Mapping nach seinem Namen im Dokument, zu dem diese MapForce-Ansicht (MapForceView) gehört, ab bzw. definiert es.

Signatur

```
ActiveMappingName : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1300	Das Objekt ist nicht mehr gültig.
1301	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.15.1.4 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1300	Das Objekt ist nicht mehr gültig.
1301	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.15.1.5 HighlightMyConnections

Mit dieser Eigenschaft wird definiert, ob nur Verbindungen vom ausgewählten Element aus markiert werden sollen.

Signatur

```
HighlightMyConnections : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1300	Das Objekt ist nicht mehr gültig.
1301	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.15.1.6 HighlightMyConnectionsRecursively

Mit dieser Eigenschaft wird definiert, ob nur die Verbindungen, die direkt von und zu dem ausgewählten Datenelement führen, markiert werden sollen.

Signatur

```
HighlightMyConnectionsRecursively : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1300	Das Objekt ist nicht mehr gültig.
1301	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.15.1.7 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Document
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1300	Das Objekt ist nicht mehr gültig.
1301	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.15.1.8 ShowItemTypes

Mit dieser Eigenschaft wird definiert, ob Elementtypen im Mapping-Diagramm angezeigt werden sollen.

Signatur

```
ShowItemTypes : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1300	Das Objekt ist nicht mehr gültig.
1301	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.15.1.9 ShowLibraryInFunctionHeader

Mit dieser Eigenschaft wird definiert, ob der Name der Funktionsbibliothek Teil von Funktionsnamen sein soll.

Signatur

```
ShowLibraryInFunctionHeader : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1300	Das Objekt ist nicht mehr gültig.
1301	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.15.2 Methoden

17.7.1.15.2.1 InsertWSDLCall

Fügt eine neue WSDL Call-Komponente zum Mapping hinzu.

Signatur

```
InsertWSDLCall(in i_strWSDLFileName:String) -> Void
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strWSDLFileName	`String`	Definiert den Pfad der WSDL-Datei, die zum Mapping hinzugefügt werden soll.

Fehler

Fehlercode	Beschreibung
1300	Das Objekt ist nicht mehr gültig.
1301	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.15.2.2 InsertXMLFile (obsolete)

`MapForceView.InsertXMLFile` wird nicht mehr verwendet. Verwenden Sie stattdessen `Mapping.InsertXMLFile`.

Signatur

```
InsertXMLFile(in i_strFileName:String, in i_strXMLRootName:String) -> Void
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFileName	`String`	

Name	Typ	Beschreibung
i_strXMLRootName	`String`	

17.7.1.15.2.3 InsertXMLSchema (obsolete)

`MapForceView.InsertXMLSchema` wird nicht mehr verwendet. Verwenden Sie stattdessen `Mapping.InsertXMLSchema`.

Signatur

```
InsertXMLSchema(in i_strSchemaFileName:String, in i_strXMLRootName:String) -> Void
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strSchemaFileName	`String`	
i_strXMLRootName	`String`	

17.7.1.15.2.4 InsertXMLSchemaWithSample (obsolete)

`MapForceView.InsertXMLSchemaWithSample` wird nicht mehr verwendet. Verwenden Sie stattdessen `Mapping.InsertXMLFile`. Beachten Sie, dass zur Übergabe des Root-Elements kein Parameter für `Mapping.InsertXMLFile` erforderlich ist. Das Root-Element wird automatisch als Root-Elementname der XML-Datei definiert.

Signatur

```
InsertXMLSchemaWithSample(in i_strSchemaFileName:String, in i_strXMLExampleFile:String,
in i_strXMLRootName:String) -> Void
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strSchemaFileName	`String`	
i_strXMLExampleFile	`String`	
i_strXMLRootName	`String`	

© 2016-2022 Altova GmbH

17.7.1.16 Mapping

Ein `Mapping`-Objekt repräsentiert ein Mapping in einem Dokument, also das Hauptmapping oder ein benutzerdefiniertes Funktionsmapping.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Mapping-Eigenschaften:

- `IsMainMapping`
- `Name`

Komponenten im Mapping:

- `Components`

Hinzufügen von Datenelementen:

- `CreateConnection`
- `InsertFunctionCall`
- `InsertXMLFile`
- `InsertXMLSchema`
- `InsertXMLSchemaInputParameter`
- `InsertXMLSchemaOutputParameter`

Eigenschaften

Name	Beschreibung
`Application` 1859	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`Components` 1860	Schreibgeschützt. Gibt eine Sammlung aller Komponenten im aktuellen Mapping zurück.
`IsMainMapping` 1860	Schreibgeschützt. Gibt an, ob das aktuelle Mapping das Hauptmapping des Dokuments ist, in dem sich das Mapping befindet. **True** bedeutet, dass es sich um das Hauptmapping handelt. **False** bedeutet, es ist eine benutzerdefinierte Funktion (UDF).
`Name` 1861	Schreibgeschützt. Der Name des Mappings oder der benutzerdefinierten Funktion (UDF).
`Parent` 1861	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.

Methoden

Name	Beschreibung
CreateConnection 1861	Erstellt zwischen den beiden bereitgestellten Datapoints (DatapointFrom & DatapointTo) eine Verbindung. Es kann keine Verbindung erstellt werden, wenn DatapointFrom kein Datapoint auf der Output-Seite ist, DatapointTo kein Datapoint auf der Input-Seite ist oder bereits eine Verbindung zwischen diesen beiden Datapoints besteht.
InsertFunctionCall 1862	Fügt eine Funktionsaufruf-Komponente in das aktuelle Mapping ein. Die angegebenen Bibliotheks- und Funktionsnamen geben die aufzurufende Funktion bzw. benutzerdefinierte Funktion an.
InsertXMLFile 1863	Fügt eine neue XML-Schemakomponente zum Mapping hinzu. Die interne Struktur der Komponente wird vom Schema bestimmt, das in der angegebenen XML-Datei (i_strFileName) referenziert wird oder, wenn die XML-Datei keine Schemadatei referenziert, von einer separat definierten Schemadatei (i_strSchemaFileName). Wenn die XML-Datei eine Schemadatei referenziert, wird der Parameter i_strSchemaFileName ignoriert. Das Root-Element der XML-Datei wird in der Komponente verwendet. Die angegebene XML-Datei wird als Input-Beispieldatei zur Überprüfung des Mappings verwendet.
InsertXMLSchema 1864	Fügt eine neue XML-Schemakomponente zum Mapping hinzu. Die interne Struktur der Komponente hängt von der Schemadatei ab, die im ersten Parameter definiert ist. Der zweite Parameter definiert das Root-Element dieses Schemas, wenn mehr als eines zur Auswahl steht. Wenn das übergebene Root-Element ein leerer String ist und mehr Auswahlmöglichkeiten zur Verfügung stehen, wird das Dialogfeld **Root-Element auswählen** angezeigt, wenn MapForce sichtbar ist. Wenn MapForce im Hintergrund läuft, wird kein Dialogfeld angezeigt und es wird eine Fehlermeldung zurückgegeben.

© 2016-2022 Altova GmbH

Name	Beschreibung
	Dieser Komponente ist keine Input-XML-Beispieldatei zugewiesen.
InsertXMLSchemaInputParameter 1865	Fügt eine XML-Schema-Input-Parameterkomponente in das aktuelle Mapping ein. Beim aktuellen Mapping muss es sich um eine benutzerdefinierte Funktion handeln. Ein Versuch ihn (den Schema-Input-Parameter) in das Hauptmapping einzufügen, wird fehlschlagen.
InsertXMLSchemaOutputParameter 1866	Fügt einen XML-Schema-Output-Parameter in das aktuelle Mapping ein. Beim aktuellen Mapping muss es sich um eine benutzerdefinierte Funktion handeln. Ein Versuch ihn (den Schema-Output-Parameter) in das Hauptmapping einzufügen, wird fehlschlagen.

17.7.1.16.1 Eigenschaften

17.7.1.16.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.16.1.2 Components

Gibt eine Sammlung aller Komponenten im aktuellen Mapping zurück.

Signatur

```
Components : Components
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.16.1.3 IsMainMapping

Gibt an, ob das aktuelle Mapping das Hauptmapping des Dokuments ist, in dem sich das Mapping befindet.

True bedeutet, dass es sich um das Hauptmapping handelt.
False bedeutet, es ist eine benutzerdefinierte Funktion (UDF).

Signatur

```
IsMainMapping : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.16.1.4 Name

Der Name des Mappings oder der benutzerdefinierten Funktion (UDF).

Signatur

```
Name : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.16.1.5 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Document
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.16.2 Methoden

17.7.1.16.2.1 CreateConnection

Erstellt zwischen den beiden bereitgestellten Datapoints (`DatapointFrom` & `DatapointTo`) eine Verbindung.

Es kann keine Verbindung erstellt werden, wenn `DatapointFrom` kein Datapoint auf der Output-Seite ist, `DatapointTo` kein Datapoint auf der Input-Seite ist oder bereits eine Verbindung zwischen diesen beiden Datapoints besteht.

Signatur

```
CreateConnection(in DatapointFrom:Datapoint, in DatapointTo:Datapoint) -> Connection
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
DatapointFrom	`Datapoint`	Der Datapoint, von dem aus die Verbindung erstellt wird.
DatapointTo	`Datapoint`	Der Ziel-Datapoint.

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1240	Eine Änderung des Dokuments ist nicht zulässig. Es ist schreibgeschützt.
1241	Die Verbindung konnte nicht erstellt werden.

17.7.1.16.2.2 InsertFunctionCall

Fügt eine Funktionsaufruf-Komponente in das aktuelle Mapping ein.

Die angegebenen Bibliotheks- und Funktionsnamen geben die aufzurufende Funktion bzw. benutzerdefinierte Funktion an.

Signatur

```
InsertFunctionCall(in strFunctionName:String, in strLibraryName:String) -> Component
```

Allgemeine Signatur

© 2016-2022 Altova GmbH

Parameter

Name	Typ	Beschreibung
strFunctionName	`String`	Der Name der einzufügenden Funktion.
strLibraryName	`String`	Der Bibliotheksname der einzufügenden Funktion.

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1240	Eine Änderung des Dokuments ist nicht zulässig. Es ist schreibgeschützt.
1242	Die Funktionsaufruf-Komponente konnte nicht erstellt werden.

17.7.1.16.2.3 InsertXMLFile

Fügt eine neue XML-Schemakomponente zum Mapping hinzu.

Die interne Struktur der Komponente wird vom Schema bestimmt, das in der angegebenen XML-Datei (`i_strFileName`) referenziert wird oder, wenn die XML-Datei keine Schemadatei referenziert, von einer separat definierten Schemadatei (`i_strSchemaFileName`).

Wenn die XML-Datei eine Schemadatei referenziert, wird der Parameter `i_strSchemaFileName` ignoriert.

Das Root-Element der XML-Datei wird in der Komponente verwendet.

Die angegebene XML-Datei wird als Input-Beispieldatei zur Überprüfung des Mappings verwendet.

Signatur

```
InsertXMLFile(in i_strFileName:String, in i_strSchemaFileName:String) -> Component
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFileName	`String`	Der Pfad der XML-Instanzdatei, die hinzugefügt werden soll.

Name	Typ	Beschreibung
i_strSchemaFileName	`String`	Der Pfad der XML-Schemadefinitionsdatei, die hinzugefügt werden soll.

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1240	Eine Änderung des Dokuments ist nicht zulässig. Es ist schreibgeschützt.
1244	Die Komponente konnte nicht erstellt werden.

17.7.1.16.2.4 InsertXMLSchema

Fügt eine neue XML-Schemakomponente zum Mapping hinzu.

Die interne Struktur der Komponente hängt von der Schemadatei ab, die im ersten Parameter definiert ist.

Der zweite Parameter definiert das Root-Element dieses Schemas, wenn mehr als eines zur Auswahl steht.

Wenn das übergebene Root-Element ein leerer String ist und mehr Auswahlmöglichkeiten zur Verfügung stehen, wird das Dialogfeld **Root-Element auswählen** angezeigt, wenn MapForce sichtbar ist. Wenn MapForce im Hintergrund läuft, wird kein Dialogfeld angezeigt und es wird eine Fehlermeldung zurückgegeben.

Dieser Komponente ist keine Input-XML-Beispieldatei zugewiesen.

Signatur

```
InsertXMLSchema(in i_strSchemaFileName:String, in i_strXMLRootName:String) -> Component
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strSchemaFileName	`String`	Der Pfad der XML-Schemadefinitionsdatei, die hinzugefügt werden soll.
i_strXMLRootName	`String`	Das Root-Element des Schemas (anwendbar, wenn das Schema mehr als ein Root-Element hat).

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1240	Eine Änderung des Dokuments ist nicht zulässig. Es ist schreibgeschützt.
1244	Die Komponente konnte nicht erstellt werden.

17.7.1.16.2.5 InsertXMLSchemaInputParameter

Fügt eine XML-Schema-Input-Parameterkomponente in das aktuelle Mapping ein.

Beim aktuellen Mapping muss es sich um eine benutzerdefinierte Funktion handeln. Ein Versuch ihn (den Schema-Input-Parameter) in das Hauptmapping einzufügen, wird fehlschlagen.

Signatur

```
InsertXMLSchemaInputParameter(in strParamName:String, in strSchemaFileName:String, in
strXMLRootElementName:String) -> Component
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
strParamName	`String`	Der Name der Input-Parameter-Komponente, die erstellt werden soll.
strSchemaFileName	`String`	Der Pfad der XML-Schemadefinitionsdatei, die hinzugefügt werden soll.
strXMLRootElementName	`String`	Das Root-Element des Schemas (anwendbar, wenn das Schema mehr als ein Root-Element hat). Wenn das übergebene Root-Element ein leerer String ist und mehr Auswahlmöglichkeiten zur Verfügung stehen, wird das Dialogfeld **Root-Element auswählen** angezeigt, wenn MapForce sichtbar ist. Wenn

Name	Typ	Beschreibung
		MapForce im Hintergrund läuft, wird kein Dialogfeld angezeigt und es wird eine Fehlermeldung zurückgegeben.

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1240	Eine Änderung des Dokuments ist nicht zulässig. Es ist schreibgeschützt.
1243	Die Parameterkomponente konnte nicht erstellt werden.
1245	Diese Operation wird für das Hauptmapping nicht unterstützt.

17.7.1.16.2.6 InsertXMLSchemaOutputParameter

Fügt einen XML-Schema-Output-Parameter in das aktuelle Mapping ein.

Beim aktuellen Mapping muss es sich um eine benutzerdefinierte Funktion handeln. Ein Versuch ihn (den Schema-Output-Parameter) in das Hauptmapping einzufügen, wird fehlschlagen.

Signatur

```
InsertXMLSchemaOutputParameter(in strParamName:String, in strSchemaFileName:String, in
strXMLRootElementName:String) -> Component
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
strParamName	`String`	Der Name der Output-Parameter-Komponente, die erstellt werden soll.
strSchemaFileName	`String`	Der Pfad der XML-Schemadefinitionsdatei, die hinzugefügt werden soll.
strXMLRootElementName	`String`	Das Root-Element des Schemas (anwendbar, wenn das Schema

© 2016-2022 Altova GmbH

Name	Typ	Beschreibung
		mehr als ein Root-Element hat). Wenn das übergebene Root-Element ein leerer String ist und mehr Auswahlmöglichkeiten zur Verfügung stehen, wird das Dialogfeld **Root-Element auswählen** angezeigt, wenn MapForce sichtbar ist. Wenn MapForce im Hintergrund läuft, wird kein Dialogfeld angezeigt und es wird eine Fehlermeldung zurückgegeben.

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1240	Eine Änderung des Dokuments ist nicht zulässig. Es ist schreibgeschützt.
1243	Die Parameterkomponente konnte nicht erstellt werden.
1245	Diese Operation wird für das Hauptmapping nicht unterstützt.

17.7.1.17 Mappings

Repräsentiert eine Sammlung von `Mapping`-Objekten.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Iterierend durch die Sammlung:

- `Count`
- `Item`

Eigenschaften

Name	Beschreibung
`Application` 1868	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.

Name	Beschreibung
Count 1868	Schreibgeschützt. Ruft die Anzahl der Mappings in der Sammlung ab.
Item 1869	Schreibgeschützt. Ruft das Mapping am Index n aus der Sammlung ab. Indizes beginnen mit 1.
Parent 1869	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.

17.7.1.17.1 Eigenschaften

17.7.1.17.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.17.1.2 Count

Ruft die Anzahl der Mappings in der Sammlung ab.

Signatur

```
Count : Integer
```

Allgemeine Signatur

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.17.1.3 Item

Ruft das Mapping am Index n aus der Sammlung ab. Indizes beginnen mit 1.

Signatur

```
Item(in n:Integer) : Mapping
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.17.1.4 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Document
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1200	Das Objekt ist nicht mehr gültig.
1201	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.18 Options

Über dieses Objekt haben Sie Zugriff auf alle im Dialogfeld **Extras | Optionen** verfügbaren MapForce-Optionen.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Allgemeine Optionen:

- `ShowLogoOnPrint`
- `ShowLogoOnStartup`
- `UseGradientBackground`

Optionen für die Codegenerierung:

- `DefaultOutputEncoding`
- `DefaultOutputByteOrder`
- `DefaultOutputByteOrderMark`
- `XSLTDefaultOutputDirectory`
- `CodeDefaultOutputDirectory`
- `CPPSettings_DOMType`
- `CPPSettings_GenerateVC6ProjectFile`
- `CppSettings_GenerateVSProjectFile`
- `CPPSettings_LibraryType`
- `CPPSettings_UseMFC`
- `CSharpSettings_ProjectType`

Eigenschaften

Name	Beschreibung
`Application` 1871	Schreibgeschützt. Ruft das Objekt auf oberster Ebene einer Applikation ab.
`CodeDefaultOutputDirectory` 1872	Definiert das Zielverzeichnis, in das von `Document.GenerateCppCode`, `Document.GenerateJavaCode` und `Document.GenerateCHashCode` generierte Dateien platziert werden sollen.
`CPPSettings_DOMType` 1872	Definiert den von `Document.GenerateCppCode` verwendeten DOM-Typ.
`CPPSettings_GenerateVC6ProjectFile` 1873	Definiert, ob von `Document.GenerateCppCode` VisualC++ 6.0-Projektdateien generiert werden sollen.
`CppSettings_GenerateVSProjectFile` 1873	Definiert, welche Version von VisualStudio-Projektdateien von `Document.GenerateCppCode` generiert werden soll.
`CPPSettings_LibraryType` 1874	Definiert den von `Document.GenerateCppCode` verwendeten Bibliothekstyp.

© 2016-2022 Altova GmbH

Name	Beschreibung
CPPSettings_UseMFC 1874	Definiert, ob von C++-Code, der von `Document.GenerateCppCode` generiert wurde, MFC-Unterstützung verwendet werden soll.
CSharpSettings_ProjectType 1875	Definiert die Art des von `Document.GenerateCHashCode` verwendeten C#-Projekts.
DefaultOutputByteOrder 1875	Bytefolge für die für Ausgabedateien verwendete Dateikodierung.
DefaultOutputByteOrderMark 1875	Gibt an, ob eine Bytefolgemarkierung (Byte Order Mark = BOM) in die Dateikodierung der Ausgabedateien inkludiert werden soll.
DefaultOutputEncoding 1876	Für Ausgabedateien verwendete Dateikodierung.
GenerateWrapperClasses 1876	Gibt an, ob bei der Codegenerierung auch Wrapper-Klassen generiert werden sollen.
JavaSettings_ApacheAxisVersion 1877	Diese Eigenschaft wird nicht mehr verwendet.
Parent 1877	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.
ShowLogoOnPrint 1878	MapForce-Logo in der Druckausgabe ein- oder ausblenden.
ShowLogoOnStartup 1878	MapForce-Logo beim Start der Applikation ein- oder ausblenden.
UseGradientBackground 1879	Hintergrundfarbmodus für ein Mapping-Fenster definieren oder abrufen.
XSLTDefaultOutputDirectory 1879	Definiert das Zielverzeichnis, in das von `Document.GenerateXSLT` generierte Dateien platziert werden sollen.

17.7.1.18.1 Eigenschaften

17.7.1.18.1.1 Application

Ruft das Objekt auf oberster Ebene einer Applikation ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.18.1.2 CodeDefaultOutputDirectory

Definiert das Zielverzeichnis, in das von `Document.GenerateCppCode`, `Document.GenerateJavaCode` und `Document.GenerateCHashCode` generierte Dateien platziert werden sollen.

Signatur

```
CodeDefaultOutputDirectory : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.18.1.3 CPPSettings_DOMType

Definiert den von `Document.GenerateCppCode` verwendeten DOM-Typ.

Signatur

```
CPPSettings_DOMType : ENUMDOMType 1916
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

Fehlercode	Beschreibung
1402	Der Parameter befindet sich außerhalb des Geltungsbereichs
1403	Der Parameterwert steht nicht mehr zur Verfügung

17.7.1.18.1.4 CPPSettings_GenerateVC6ProjectFile (obsolete)

Definiert, ob von `Document.GenerateCppCode` VisualC++ 6.0-Projektdateien generiert werden sollen.

Signatur

```
CPPSettings_GenerateVC6ProjectFile : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1402	Der Parameter befindet sich außerhalb des Geltungsbereichs
1403	Der Parameterwert steht nicht mehr zur Verfügung

17.7.1.18.1.5 CppSettings_GenerateVSProjectFile

Definiert, welche Version von VisualStudio-Projektdateien von `Document.GenerateCppCode` generiert werden soll.

Signatur

```
CppSettings_GenerateVSProjectFile : ENUMProjectType 1917
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

Fehlercode	Beschreibung
1402	Der Parameter befindet sich außerhalb des Geltungsbereichs
1403	Der Parameterwert steht nicht mehr zur Verfügung

17.7.1.18.1.6 CPPSettings_LibraryType

Definiert den von `Document.GenerateCppCode` verwendeten Bibliothekstyp.

Signatur

```
CPPSettings_LibraryType : ENUMLibType
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.18.1.7 CPPSettings_UseMFC

Definiert, ob von C++-Code, der von `Document.GenerateCppCode` generiert wurde, MFC-Unterstützung verwendet werden soll.

Signatur

```
CPPSettings_UseMFC : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.18.1.8 CSharpSettings_ProjectType

Definiert die Art des von `Document.GenerateCHashCode` verwendeten C#-Projekts.

Signatur

```
CSharpSettings_ProjectType : ENUMProjectType
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1402	Der Parameter befindet sich außerhalb des Geltungsbereichs
1403	Der Parameterwert steht nicht mehr zur Verfügung

17.7.1.18.1.9 DefaultOutputByteOrder

Bytefolge für die für Ausgabedateien verwendete Dateikodierung.

Signatur

```
DefaultOutputByteOrder : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.18.1.10 DefaultOutputByteOrderMark

Gibt an, ob eine Bytefolgemarkierung (Byte Order Mark = BOM) in die Dateikodierung der Ausgabedateien inkludiert werden soll.

Signatur

```
DefaultOutputByteOrderMark : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.18.1.11 DefaultOutputEncoding

Für Ausgabedateien verwendete Dateikodierung.

Signatur

```
DefaultOutputEncoding : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.18.1.12 GenerateWrapperClasses

Gibt an, ob bei der Codegenerierung auch Wrapper-Klassen generiert werden sollen.

Signatur

```
GenerateWrapperClasses : Boolean
```

Allgemeine Signatur

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.18.1.13 JavaSettings_ApacheAxisVersion (obsolete)

Diese Eigenschaft wird nicht mehr verwendet.

Signatur

```
JavaSettings_ApacheAxisVersion : ENUMApacheAxisVersion
```

Allgemeine Signatur

17.7.1.18.1.14 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.18.1.15 ShowLogoOnPrint

MapForce-Logo in der Druckausgabe ein- oder ausblenden.

Signatur

```
ShowLogoOnPrint : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.18.1.16 ShowLogoOnStartup

MapForce-Logo beim Start der Applikation ein- oder ausblenden.

Signatur

```
ShowLogoOnStartup : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.18.1.17 UseGradientBackground

Hintergrundfarbmodus für ein Mapping-Fenster definieren oder abrufen.

Signatur

```
UseGradientBackground : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.18.1.18 XSLTDefaultOutputDirectory

Definiert das Zielverzeichnis, in das von `Document.GenerateXSLT` generierte Dateien platziert werden sollen.

Signatur

```
XSLTDefaultOutputDirectory : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1400	Das Objekt ist nicht mehr gültig.
1401	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.19 Project

Ein `Project`-Objekt repräsentiert ein Projekt und seine Projektelementstruktur in MapForce.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Behandlung von Dateien:

- FullName
- Name
- Path
- Saved
- Save
- Close

Navigation in der Projektstruktur:

- Count
- Item
- _NewEnum

Bearbeitung der Projektstruktur

- AddActiveFile
- AddFile
- InsertWebService (nur Enterprise Edition)
- CreateFolder

Codegenerierung:

- Output_Folder
- Output_Language
- Output_TextEncoding
- Java_BasePackageName
- GenerateCode
- GenerateCodeEx
- GenerateCodeIn
- GenerateCodeInEx

Beispiele zur Verwendung der oben aufgelisteten Eigenschaften und Methoden finden Sie unter Beispiel[1747]: Projektaufgaben[1747]. Zur Durchführung von Operationen, an denen Webservices beteiligt sind, wird die MapForce Enterprise Edition benötigt.

Eigenschaften

Name	Beschreibung
_NewEnum[1883]	Schreibgeschützt. Diese Eigenschaft unterstützt die sprachspezifische Standardenumeration.
Application[1884]	Schreibgeschützt. Ruft das oberste Applikationsobjekt ab.
Count[1884]	Schreibgeschützt. Ruft die Anzahl der Children des Root-Elements des Projekts ab. Beispiele dazu finden Sie unter Item oder _NewEnum
FullName[1885]	Pfad und Name der Projektdatei.

© 2016-2022 Altova GmbH

Name	Beschreibung
Item 1885	Schreibgeschützt. Gibt das Child an der Position *n* der Projekt-Root zurück. Der Index ist 1-basiert (der erste Index ist 1). Der größte gültige Index ist `Count`. Alternativen dazu finden Sie unter `_NewEnum`.
Java_BasePackageName 1886	Definiert den Basispaketnamen der Java-Pakete, die generiert werden, bzw. ruft diesen ab. Diese Eigenschaft wird nur beim Generieren von Java-Code verwendet.
Name 1886	Schreibgeschützt. Name der Projektdatei ohne Dateipfad.
Output_Folder 1887	Definiert den Standardausgabeordner, der mit `GenerateCode` und `GenerateCodeIn` verwendet wird, bzw. ruft diesen ab. Projektelemente können diesen Wert in ihrer Eigenschaft `CodeGenSettings_OutputFolder` überschreiben, wenn `CodeGenSettings_UseDefault` auf false gesetzt wurde.
Output_Language 1887	Definiert die Standardsprache für die Codegenerierung bei Verwendung von `GenerateCode` bzw. ruft diese ab. Projektelemente können diesen Wert in ihrer Eigenschaft `CodeGenSettings_OutputLanguage` überschreiben, wenn `CodeGenSettings_UseDefault` auf false gesetzt wurde.
Output_TextEncoding 1888	Definiert die beim Generieren von XML-basiertem Code verwendete Textkodierung bzw. ruft diese ab.
Parent 1888	Schreibgeschützt. Das Parent-Objekt laut Objektmodell.
Path 1889	Schreibgeschützt. Pfad der Projektdatei ohne Namen.
Saved 1889	Schreibgeschützt. **True,** wenn das Projekt seit der letzten Speicherung mit `Save` nicht geändert wurde, andernfalls **false**.

Methoden

Name	Beschreibung
AddActiveFile 1890	Fügt das gerade offene Dokument zum Mapping-Ordner der Root des Projekts hinzu.
AddFile 1890	Fügt das angegebene Dokument zum Mapping-Ordner oder zur Root des Projekts hinzu.
Close 1891	Schließt das Projekt ohne es zu speichern.
CreateFolder 1891	Erstellt einen neuen Ordner als Child des Root-Elements des Projekts.

Name	Beschreibung
GenerateCode 1892	Generiert Code für alle Projektelemente des Projekts. Die Codesprache und der Ausgabepfad werden von den Eigenschaften des Projekts und der Projektelemente bestimmt.
GenerateCodeEx 1892	Generiert Code für alle Projektelemente des Projekts. Die Codesprache und der Ausgabepfad werden von den Eigenschaften des Projekts und der Projektelemente bestimmt. Bei der Codegenerierung wird ein Objekt zurückgegeben, mit dem Sie durch alle ausgegebenen Meldungen iterieren können. Diese Meldungen sind dieselben wie die im Fenster "Meldungen" von MapForce angezeigten.
GenerateCodeIn 1893	Generiert Code für alle Projektelemente des Projekts in der angegebenen Sprache. Der Ausgabepfad wird von den Eigenschaften des Projekts und der Projektelemente bestimmt.
GenerateCodeInEx 1893	Generiert Code für alle Projektelemente des Projekts in der angegebenen Sprache. Der Ausgabepfad wird von den Eigenschaften des Projekts und der Projektelemente bestimmt. Bei der Codegenerierung wird ein Objekt zurückgegeben, mit dem Sie durch alle ausgegebenen Meldungen iterieren können. Diese Meldungen sind dieselben wie die im Fenster **Meldungen** von MapForce angezeigten.
InsertWebService 1894	Fügt ein neues Webservice Projekt in den Webservice-Ordner des Projekts ein. Wenn **i_bGenerateMappings** true ist, werden die Mapping-Anfangsdokumente für alle Ports automatisch generiert.
Save 1895	Speichert das Projekt in der durch `FullName` definierten Datei.

Events

Name	Beschreibung
OnProjectClosed 1895	Dieses Event wird beim Schließen des Projekts ausgelöst. Das an den Event Handler übergebene Projektobjekt sollte nicht aufgerufen werden. Das entsprechende Event zum Öffnen ist `Application.OnProjectOpened`.

© 2016-2022 Altova GmbH

17.7.1.19.1 Eigenschaften

17.7.1.19.1.1 _NewEnum

Diese Eigenschaft unterstützt die sprachspezifische Standardenumeration.

Signatur

```
_NewEnum : IUnknown
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.

Beispiele

```
// ------------------------------------------------------------
// JScript sample - enumeration of a project's project items.
function AllChildrenOfProjectRoot()
{
    objProject = objMapForce.ActiveProject;
    if ( objProject != null )
    {

for ( objProjectIter = new Enumerator(objProject); ! objProjectIter.atEnd(); objProjectIt
er.moveNext() )
        {
            objProjectItem = objProjectIter.item();

            // do something with project item here
        }
    }
}
```

```
// ------------------------------------------------------------
// JScript sample - iterate all project items, depth first.
function IterateProjectItemsRec(objProjectItemIter)
{
    while ( ! objProjectItemIter.atEnd() )
    {
        objProjectItem = objProjectItemIter.item();
        // do something with project item here

        IterateProjectItemsRec( new Enumerator(objProjectItem) );
```

```
        objProjectItemIter.moveNext();
    }
}
function IterateAllProjectItems()
{
    objProject = objMapForce.ActiveProject;
    if ( objProject != null )
    {
        IterateProjectItemsRec( new Enumerator(objProject) );
    }
}
```

17.7.1.19.1.2 Application

Ruft das oberste Applikationsobjekt ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.19.1.3 Count

Ruft die Anzahl der Children des Root-Elements des Projekts ab. Beispiele dazu finden Sie unter `Item` oder `_NewEnum`

Signatur

```
Count : Integer
```

Allgemeine Signatur

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.

17.7.1.19.1.4 FullName

Pfad und Name der Projektdatei.

Signatur

```
FullName : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.19.1.5 Item

Gibt das Child an der Position *n* der Projekt-Root zurück. Der Index ist 1-basiert (der erste Index ist 1). Der größte gültige Index ist `Count`. Alternativen dazu finden Sie unter `_NewEnum`.

Signatur

```
Item(in n:Integer) : ProjectItem
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.

Beispiele

```
// ------------------------------------------------------------
```

```
// JScript code snippet - enumerate children using Count and Item.
for( nItemIndex = 1; nItemIndex <= objProject.Count; nItemIndex++ )
{
    objProjectItem = objProject.Item(nItemIndex);
    // do something with project item here
}
```

17.7.1.19.1.6 Java_BasePackageName

Definiert den Basispaketnamen der Java-Pakete, die generiert werden, bzw. ruft diesen ab. Diese Eigenschaft wird nur beim Generieren von Java-Code verwendet.

Signatur

```
Java_BasePackageName : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Es wurde ein ungültiger Paketname definiert. Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.19.1.7 Name

Name der Projektdatei ohne Dateipfad.

Signatur

```
Name : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.19.1.8 Output_Folder

Definiert den Standardausgabeordner, der mit `GenerateCode` und `GenerateCodeIn` verwendet wird, bzw. ruft diesen ab. Projektelemente können diesen Wert in ihrer Eigenschaft `CodeGenSettings_OutputFolder` überschreiben, wenn `CodeGenSettings_UseDefault` auf false gesetzt wurde.

Signatur

```
Output_Folder : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Es wurde ein ungültiger Ordnername definiert. Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.19.1.9 Output_Language

Definiert die Standardsprache für die Codegenerierung bei Verwendung von `GenerateCode` bzw. ruft diese ab. Projektelemente können diesen Wert in ihrer Eigenschaft `CodeGenSettings_OutputLanguage` überschreiben, wenn `CodeGenSettings_UseDefault` auf false gesetzt wurde.

Signatur

```
Output_Language : ENUMProgrammingLanguage 1916
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Es wurde eine ungültige Sprache definiert. Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.19.1.10 Output_TextEncoding

Definiert die beim Generieren von XML-basiertem Code verwendete Textkodierung bzw. ruft diese ab.

Signatur

```
Output_TextEncoding : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Es wurde eine ungültige Textkodierung definiert. Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.19.1.11 Parent

Das Parent-Objekt laut Objektmodell.

Signatur

```
Parent : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

© 2016-2022 Altova GmbH

17.7.1.19.1.12 Path

Pfad der Projektdatei ohne Namen.

Signatur

```
Path : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.19.1.13 Saved

True, wenn das Projekt seit der letzten Speicherung mit `Save` nicht geändert wurde, andernfalls **false**.

Signatur

```
Saved : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.19.2 Methoden

17.7.1.19.2.1 AddActiveFile

Fügt das gerade offene Dokument zum Mapping-Ordner der Root des Projekts hinzu.

Signatur

```
AddActiveFile() -> ProjectItem
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1503	Es ist kein aktives Dokument verfügbar.
1504	Das aktive Dokument muss einen Pfadnamen erhalten, bevor es zum Projekt hinzugefügt werden kann.
1705	Das Mapping konnte dem Projekt nicht zugewiesen werden. Möglicherweise ist es bereits im Zielordner vorhanden.

17.7.1.19.2.2 AddFile

Fügt das angegebene Dokument zum Mapping-Ordner oder zur Root des Projekts hinzu.

Signatur

```
AddFile(in i_strFileName:String) -> ProjectItem
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFileName	`String`	Definiert den Pfad zu dem hinzuzufügenden Dokument.

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1705	Das Mapping konnte dem Projekt nicht zugewiesen werden. Entweder die Datei existiert nicht oder ist kein MapForce-Mapping. Möglicherweise wurde die Datei dem Zielordner bereits zugewiesen.

17.7.1.19.2.3 Close

Schließt das Projekt ohne es zu speichern.

Signatur

```
Close() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.

17.7.1.19.2.4 CreateFolder

Erstellt einen neuen Ordner als Child des Root-Elements des Projekts.

Signatur

```
CreateFolder(in i_strFolderName:String) -> ProjectItem
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFolderName	`String`	Der Name des zu erstellenden Ordners.

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Für den Rückgabeparameter wurde ein ungültiger Ordnername oder eine ungültige Adresse angegeben.

17.7.1.19.2.5 GenerateCode

Generiert Code für alle Projektelemente des Projekts. Die Codesprache und der Ausgabepfad werden von den Eigenschaften des Projekts und der Projektelemente bestimmt.

Signatur

```
GenerateCode() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1706	Fehler bei der Codegenerierung.

17.7.1.19.2.6 GenerateCodeEx

Generiert Code für alle Projektelemente des Projekts. Die Codesprache und der Ausgabepfad werden von den Eigenschaften des Projekts und der Projektelemente bestimmt. Bei der Codegenerierung wird ein Objekt zurückgegeben, mit dem Sie durch alle ausgegebenen Meldungen iterieren können. Diese Meldungen sind dieselben wie die im Fenster "Meldungen" von MapForce angezeigten.

Signatur

```
GenerateCodeEx() -> ErrorMarkers
```

Allgemeine Signatur

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1706	Fehler bei der Codegenerierung.

17.7.1.19.2.7 GenerateCodeIn

Generiert Code für alle Projektelemente des Projekts in der angegebenen Sprache. Der Ausgabepfad wird von den Eigenschaften des Projekts und der Projektelemente bestimmt.

Signatur

```
GenerateCodeIn(in i_nLanguage:ENUMProgrammingLanguage) -> Void
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_nLanguage	ENUMProgrammingLanguage	Definiert die Programmiersprache, in der der Code generiert werden soll.

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1706	Fehler bei der Codegenerierung.

17.7.1.19.2.8 GenerateCodeInEx

Generiert Code für alle Projektelemente des Projekts in der angegebenen Sprache. Der Ausgabepfad wird von den Eigenschaften des Projekts und der Projektelemente bestimmt. Bei der Codegenerierung wird ein Objekt zurückgegeben, mit dem Sie durch alle ausgegebenen Meldungen iterieren können. Diese Meldungen sind dieselben wie die im Fenster **Meldungen** von MapForce angezeigten.

Signatur

```
GenerateCodeInEx(in i_nLanguage:ENUMProgrammingLanguage) -> ErrorMarkers
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_nLanguage	ENUMProgrammingLanguage	Definiert die Programmiersprache, in der der Code generiert werden soll.

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1706	Fehler bei der Codegenerierung.

17.7.1.19.2.9 InsertWebService

Fügt ein neues Webservice Projekt in den Webservice-Ordner des Projekts ein. Wenn **i_bGenerateMappings** true ist, werden die Mapping-Anfangsdokumente für alle Ports automatisch generiert.

Signatur

```
InsertWebService(in i_strWSDLFile:String, in i_strService:String, in i_strPort:String, in
i_bGenerateMappings:Boolean) -> ProjectItem
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strWSDLFile	String	Definiert den Pfad zu der hinzuzufügenden WSDL-Datei.
i_strService	String	Definiert den Namen des hinzuzufügenden Webservice.

© 2016-2022 Altova GmbH

Name	Typ	Beschreibung
i_strPort	`String`	Definiert den Port des hinzuzufügenden Webservice.
i_bGenerateMappings	`Boolean`	Wenn dieser Parameter true ist, werden die Mapping-Anfangsdokumente für alle Ports automatisch generiert.

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1501	WSDL-Datei wurde nicht gefunden oder ist ungültig. Service- oder Port-Namen sind ungültig. Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1503	Die Operation wird von der aktuellen Edition nicht unterstützt.

17.7.1.19.2.10 Save

Speichert das Projekt in der durch `FullName` definierten Datei.

Signatur

```
Save() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1500	Das Objekt ist nicht mehr gültig.
1502	Kann nicht in Datei gespeichert werden.

17.7.1.19.3 Events

17.7.1.19.3.1 OnProjectClosed

Dieses Event wird beim Schließen des Projekts ausgelöst. Das an den Event Handler übergebene Projektobjekt sollte nicht aufgerufen werden. Das entsprechende Event zum Öffnen ist `Application.OnProjectOpened`.

Signatur

```
OnProjectClosed(in i_ipProject:Project) : Void
```

17.7.1.20 ProjectItem

Ein `ProjectItem`-Objekt repräsentiert einen Eintrag in der Projektstruktur.

Eigenschaften zum Navigieren im Objektmodell:

- `Application`
- `Parent`

Navigation in der Projektstruktur:

- `Count`
- `Item`
- `_NewEnum`

Eigenschaften des Projektelements:

- `Kind`
- `Name`
- `WSDLFile` (steht nur bei Webservice-Projektelementen zur Verfügung)
- `QualifiedName` (steht nur bei Webservice-Projektelementen zur Verfügung)

Bearbeitung der Projektstruktur:

- `AddActiveFile` (steht nur für Ordnerelemente zur Verfügung)
- `AddFile` (steht nur für Ordnerelemente zur Verfügung)
- `CreateFolder` (steht nur für Ordnerelemente zur Verfügung)
- `CreateMappingForProject` (steht nur für Webservice-Operationen zur Verfügung)
- `Remove`

Dokumentzugriff:

- `Open` (steht nur für Mapping-Elemente und Webservice-Operationen zur Verfügung)

Codegenerierung:

- `CodeGenSettings_UseDefault`
- `CodeGenSettings_OutputFolder`
- `CodeGenSettings_Language`
- `GenerateCode`
- `GenerateCodeEx`
- `GenerateCodeIn`
- `GenerateCodeInEx`

Beispiele zur Verwendung der oben aufgelisteten Eigenschaften und Methoden finden Sie unter Beispiel: Projektaufgaben 1747. Für Operationen mit Webservices benötigen Sie die MapForce Enterprise Edition.

 © 2016-2022 Altova GmbH

Eigenschaften

Name	Beschreibung
_NewEnum 1899	Schreibgeschützt. Diese Eigenschaft unterstützt die sprachspezifische Standardenumeration. Beispiele dazu finden Sie unter `Project.Item` oder `Project._NewEnum`.
Application 1900	Schreibgeschützt. Ruft das oberste Applikationsobjekt ab.
CodeGenSettings_Language 1900	Definiert die mit `GenerateCode` oder `Project.GenerateCode` zu verwendende Sprache oder ruft diese ab. Diese Eigenschaft wird nur benötigt, wenn `CodeGenSettings_UseDefault` auf false gesetzt ist.
CodeGenSettings_OutputFolder 1901	Definiert das mit `GenerateCode`, `GenerateCodeIn`, `Project.GenerateCode` oder `Project.GenerateCodeIn` zu verwendende Ausgabeverzeichnis oder ruft dieses ab. Diese Eigenschaft wird nur benötigt, wenn `CodeGenSettings_UseDefault` auf false gesetzt ist.
CodeGenSettings_UseDefault 1901	Definiert bzw. ruft ab, ob das Ausgabeverzeichnis und die Codesprache, wie von (a) den Parent-Ordnern oder (b) der Projekt-Root definiert, verwendet werden sollen. Diese Eigenschaft wird bei Aufrufen von `GenerateCode`, `GenerateCodeIn`, `Project.GenerateCode` und `Project.GenerateCodeIn` verwendet. Wenn diese Eigenschaft auf false gesetzt ist, werden die Werte von `CodeGenSettings_OutputFolder` und `CodeGenSettings_Language` verwendet, um Code für dieses Projektelement zu generieren.
Count 1902	Schreibgeschützt. Ruft die Anzahl der Children dieses Projektelements ab. Siehe auch `Item`. Beispiele dazu finden Sie unter `Project.Item` oder `Project._NewEnum`.
Item 1902	Schreibgeschützt. Gibt das Child an der Position `n` dieses Projektelements zurück. Der Index ist 1-basiert (der erste Index ist 1). Der größte gültige Index ist `ProjectItem.Count`. Alternativen dazu finden Sie unter `ProjectItem._NewEnum`. Beispiele dazu finden Sie unter `Project.Item` oder `Project._NewEnum`.
Kind 1903	Schreibgeschützt. Ruft die Art des Projektelements ab. Die Verfügbarkeit einiger Eigenschaften und die Anwendbarkeit bestimmter Methoden ist auf bestimmte Arten von Projektelementen beschränkt. Die Beschreibung aller Methoden und Eigenschaften enthält Informationen über diese Einschränkungen.

Name	Beschreibung
Name 1903	Ruft den Namen eines Projektelements ab oder definiert diesen. Der Name der meisten Elemente ist schreibgeschützt. Ausnahmen sind vom Benutzer erstellte Ordner. Die Namen solcher Ordner können nach der Erstellung geändert werden.
Parent 1904	Schreibgeschützt. Ruft das Projekt ab, von dem dieses Element ein Child-Element ist. Hat dieselbe Wirkung wie `Application.ActiveProject`.
QualifiedName 1904	Schreibgeschützt. Ruft den qualifizierten Namen eines Webservice-Elements ab.
WSDLFile 1904	Schreibgeschützt. Ruft den Dateinamen der WSDL-Datei ab, die den Webservice definiert, der das aktuelle Projektelement enthält.

Methoden

Name	Beschreibung
AddActiveFile 1905	Fügt das gerade aktive Dokument zu diesem Projektelement hinzu, wenn es ein gültiges Child ist. Andernfalls wird das Dokument zum Mapping-Ordner der Projekt-Root hinzugefügt.
AddFile 1906	Fügt des angegebene Dokument zu diesem Projektelement hinzu, wenn es ein gültiges Child-Dokument ist. Andernfalls wird das Dokument zum Mapping-Ordner der Projekt-Root hinzugefügt.
CreateFolder 1906	Erstellt einen neuen Ordner als Child dieses Projektelements.
CreateMappingForProject 1907	Erstellt ein Mapping-Anfangsdokument für eine Webservice-Operation und speichert es unter **i_strFileName**. Bei Verwendung von `Project.InsertWebService` können Sie das Flag **i_bGenerateMappings** verwenden, damit MapForce automatisch Anfangsmappings für alle Ports erstellt.
GenerateCode 1908	Generiert Code für dieses Projektelement und seine Children. Die Codesprache und das Ausgabeverzeichnis werden durch `CodeGenSettings_UseDefault`, `CodeGenSettings_Language` und `CodeGenSettings_OutputFolder` definiert. Child-Elemente dieses Projektelements können ihre eigenen Eigenschaftseinstellungen für die Codegenerierung haben.
GenerateCodeEx 1908	Generiert Code für dieses Projektelement und seine Children. Die Codesprache und das Ausgabeverzeichnis werden durch `CodeGenSettings_UseDefault`, `CodeGenSettings_Language` und `CodeGenSettings_OutputFolder` definiert. Child-Elemente dieses Projektelements können ihre eigenen Eigenschaftseinstellungen für die Codegenerierung haben.

© 2016-2022 Altova GmbH

Name	Beschreibung
GenerateCodeIn 1909	Generiert in der angegebenen Sprache Code für dieses Projektelement und seine Child-Elemente. Das Ausgabeverzeichnis wird durch `CodeGenSettings_UseDefault` und `CodeGenSettings_OutputFolder` definiert. Child-Elemente dieses Projektelements können ihre eigenen Eigenschaftseinstellungen für die Codegenerierung haben.
GenerateCodeInEx 1909	Generiert in der angegebenen Sprache Code für dieses Projektelement und seine Child-Elemente. Das Ausgabeverzeichnis wird durch `CodeGenSettings_UseDefault` und `CodeGenSettings_OutputFolder` definiert. Child-Elemente dieses Projektelements können ihre eigenen Eigenschaftseinstellungen für die Codegenerierung haben. Bei der Codegenerierung wird ein Objekt zurückgegeben, mit dem Sie durch alle ausgegebenen Meldungen iterieren können. Diese Meldungen sind dieselben wie die im Fenster "Meldungen" von MapForce angezeigten.
Open 1910	Öffnet das Projektelement als Dokument oder macht das entsprechende Dokument zum aktiven, wenn es bereits geöffnet ist. Das Projektelement muss ein MapForce-Mapping sein oder - nur bei der Enterprise Edition - eine Webservice-Operation.
Remove 1911	Entfernt dieses Projektelement und alle seine Child-Elemente aus der Projektstruktur.

Events

Name	Beschreibung
OnModifiedFlagChanged 1911	Kommt vor, wenn sich der Änderungsstatus von `ProjectItem` ändert.
OnProjectClosed 1911	Dieses Event wird beim Schließen des Projekts ausgelöst. Das an den Event Handler übergebene Projektobjekt sollte nicht aufgerufen werden. Das entsprechende Event zum Öffnen ist `Application.OnProjectOpened`.

17.7.1.20.1 Eigenschaften

17.7.1.20.1.1 _NewEnum

Diese Eigenschaft unterstützt die sprachspezifische Standardenumeration. Beispiele dazu finden Sie unter `Project.Item` oder `Project._NewEnum`.

Signatur

```
_NewEnum : IUnknown
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.

17.7.1.20.1.2 Application

Ruft das oberste Applikationsobjekt ab.

Signatur

```
Application : Application
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.20.1.3 CodeGenSettings_Language

Definiert die mit `GenerateCode` oder `Project.GenerateCode` zu verwendende Sprache oder ruft diese ab. Diese Eigenschaft wird nur benötigt, wenn `CodeGenSettings_UseDefault` auf false gesetzt ist.

Signatur

```
CodeGenSettings_Language : ENUMProgrammingLanguage 1916
```

Allgemeine Signatur

© 2016-2022 Altova GmbH

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Es wurde für den Rückgabeparameter eine ungültige Sprache oder eine ungültige Adresse angegeben.

17.7.1.20.1.4 CodeGenSettings_OutputFolder

Definiert das mit `GenerateCode`, `GenerateCodeIn`, `Project.GenerateCode` oder `Project.GenerateCodeIn` zu verwendende Ausgabeverzeichnis oder ruft dieses ab. Diese Eigenschaft wird nur benötigt, wenn `CodeGenSettings_UseDefault` auf false gesetzt ist.

Signatur

```
CodeGenSettings_OutputFolder : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Es wurde für den Rückgabeparameter ein ungültiger Ausgabeordner oder eine ungültige Adresse angegeben.

17.7.1.20.1.5 CodeGenSettings_UseDefault

Definiert bzw. ruft ab, ob das Ausgabeverzeichnis und die Codesprache, wie von (a) den Parent-Ordnern oder (b) der Projekt-Root definiert, verwendet werden sollen. Diese Eigenschaft wird bei Aufrufen von `GenerateCode`, `GenerateCodeIn`, `Project.GenerateCode` und `Project.GenerateCodeIn` verwendet. Wenn diese Eigenschaft auf false gesetzt ist, werden die Werte von `CodeGenSettings_OutputFolder` und `CodeGenSettings_Language` verwendet, um Code für dieses Projektelement zu generieren.

Signatur

```
CodeGenSettings_UseDefault : Boolean
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.20.1.6 Count

Ruft die Anzahl der Children dieses Projektelements ab. Siehe auch `Item`. Beispiele dazu finden Sie unter `Project.Item` oder `Project._NewEnum`.

Signatur

```
Count : Integer
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.

17.7.1.20.1.7 Item

Gibt das Child an der Position `n` dieses Projektelements zurück. Der Index ist 1-basiert (der erste Index ist 1). Der größte gültige Index ist `ProjectItem.Count`. Alternativen dazu finden Sie unter `ProjectItem._NewEnum`. Beispiele dazu finden Sie unter `Project.Item` oder `Project._NewEnum`.

Signatur

```
Item(in n:Integer) : ProjectItem
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.

© 2016-2022 Altova GmbH

17.7.1.20.1.8 Kind

Ruft die Art des Projektelements ab. Die Verfügbarkeit einiger Eigenschaften und die Anwendbarkeit bestimmter Methoden ist auf bestimmte Arten von Projektelementen beschränkt. Die Beschreibung aller Methoden und Eigenschaften enthält Informationen über diese Einschränkungen.

Signatur

```
Kind : ENUMProjectItemType 1917
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.20.1.9 Name

Ruft den Namen eines Projektelements ab oder definiert diesen. Der Name der meisten Elemente ist schreibgeschützt. Ausnahmen sind vom Benutzer erstellte Ordner. Die Namen solcher Ordner können nach der Erstellung geändert werden.

Signatur

```
Name : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1702	Der Name des Projektelements kann nicht geändert werden.

17.7.1.20.1.10 Parent

Ruft das Projekt ab, von dem dieses Element ein Child-Element ist. Hat dieselbe Wirkung wie `Application.ActiveProject`.

Signatur

```
Parent : Project
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.

17.7.1.20.1.11 QualifiedName

Ruft den qualifizierten Namen eines Webservice-Elements ab.

Signatur

```
QualifiedName : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1702	Das Projektelement ist nicht Teil eines Webservice.

17.7.1.20.1.12 WSDLFile

Ruft den Dateinamen der WSDL-Datei ab, die den Webservice definiert, der das aktuelle Projektelement enthält.

© 2016-2022 Altova GmbH

Signatur

```
WSDLFile : String
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1702	Das Projektelement ist nicht Teil eines Webservice.

17.7.1.20.2 Methoden

17.7.1.20.2.1 AddActiveFile

Fügt das gerade aktive Dokument zu diesem Projektelement hinzu, wenn es ein gültiges Child ist. Andernfalls wird das Dokument zum Mapping-Ordner der Projekt-Root hinzugefügt.

Signatur

```
AddActiveFile() -> ProjectItem
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Der Dateiname ist leer. Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1703	Es ist kein aktives Dokument verfügbar.
1704	Das aktive Dokument muss einen Pfadnamen erhalten, bevor es zum Projekt hinzugefügt werden kann.
1705	Das Mapping konnte dem Projekt nicht zugewiesen werden. Entweder die Datei existiert nicht oder ist kein MapForce-Mapping. Möglicherweise wurde die Datei dem Zielordner bereits zugewiesen.

17.7.1.20.2.2 AddFile

Fügt des angegebene Dokument zu diesem Projektelement hinzu, wenn es ein gültiges Child-Dokument ist. Andernfalls wird das Dokument zum Mapping-Ordner der Projekt-Root hinzugefügt.

Signatur

```
AddFile(in i_strFilePath:String) -> ProjectItem
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFilePath	`String`	Der Pfad zu dem hinzuzufügenden Dokument.

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Der Dateiname ist leer. Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1705	Das Mapping konnte dem Projekt nicht zugewiesen werden. Entweder die Datei existiert nicht oder ist kein MapForce-Mapping. Möglicherweise wurde die Datei dem Zielordner bereits zugewiesen.

17.7.1.20.2.3 CreateFolder

Erstellt einen neuen Ordner als Child dieses Projektelements.

Signatur

```
CreateFolder(in i_strFolderName:String) -> ProjectItem
```

Allgemeine Signatur

© 2016-2022 Altova GmbH

Parameter

Name	Typ	Beschreibung
i_strFolderName	`String`	Der Name des zu erstellenden Ordners.

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Für den Rückgabeparameter wurde ein ungültiger Ordnername oder eine ungültige Adresse angegeben.
1702	Das Projektelement unterstützt keine Child-Elemente.

17.7.1.20.2.4 CreateMappingForProject

Erstellt ein Mapping-Anfangsdokument für eine Webservice-Operation und speichert es unter **i_strFileName**. Bei Verwendung von `Project.InsertWebService` können Sie das Flag **i_bGenerateMappings** verwenden, damit MapForce automatisch Anfangsmappings für alle Ports erstellt.

Signatur

```
CreateMappingForProject(in i_strFileName:String) -> ProjectItem
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_strFileName	`String`	Definiert den Pfad, unter dem das Mapping gespeichert werden soll.

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1707	Neues Mapping kann nicht erstellt werden. Das Projektelement unterstützt entweder die automatische Erstellung von Anfangsmappings nicht oder es ist bereits ein

Fehlercode	Beschreibung
	Mapping vorhanden.
1708	Die Operation wird in der aktuellen Edition nicht unterstützt.

17.7.1.20.2.5 GenerateCode

Generiert Code für dieses Projektelement und seine Children. Die Codesprache und das Ausgabeverzeichnis werden durch `CodeGenSettings_UseDefault`, `CodeGenSettings_Language` und `CodeGenSettings_OutputFolder` definiert. Child-Elemente dieses Projektelements können ihre eigenen Eigenschaftseinstellungen für die Codegenerierung haben.

Signatur

```
GenerateCode() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1706	Fehler bei der Codegenerierung.

17.7.1.20.2.6 GenerateCodeEx

Generiert Code für dieses Projektelement und seine Children. Die Codesprache und das Ausgabeverzeichnis werden durch `CodeGenSettings_UseDefault`, `CodeGenSettings_Language` und `CodeGenSettings_OutputFolder` definiert. Child-Elemente dieses Projektelements können ihre eigenen Eigenschaftseinstellungen für die Codegenerierung haben.

Signatur

```
GenerateCodeEx() -> ErrorMarkers
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.

© 2016-2022 Altova GmbH

Fehlercode	Beschreibung
1701	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1706	Fehler bei der Codegenerierung.

17.7.1.20.2.7 GenerateCodeIn

Generiert in der angegebenen Sprache Code für dieses Projektelement und seine Child-Elemente. Das Ausgabeverzeichnis wird durch `CodeGenSettings_UseDefault` und `CodeGenSettings_OutputFolder` definiert. Child-Elemente dieses Projektelements können ihre eigenen Eigenschaftseinstellungen für die Codegenerierung haben.

Signatur

```
GenerateCodeIn(in i_nLanguage:ENUMProgrammingLanguage) -> Void
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_nLanguage	ENUMProgrammingLanguage	Definiert die Programmiersprache für die Codegenerierung.

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Es wurde eine ungültige Sprache definiert.
1706	Fehler bei der Codegenerierung.

17.7.1.20.2.8 GenerateCodeInEx

Generiert in der angegebenen Sprache Code für dieses Projektelement und seine Child-Elemente. Das Ausgabeverzeichnis wird durch `CodeGenSettings_UseDefault` und `CodeGenSettings_OutputFolder` definiert. Child-Elemente dieses Projektelements können ihre eigenen Eigenschaftseinstellungen für die Codegenerierung haben.

Bei der Codegenerierung wird ein Objekt zurückgegeben, mit dem Sie durch alle ausgegebenen Meldungen iterieren können. Diese Meldungen sind dieselben wie die im Fenster "Meldungen" von MapForce angezeigten.

Signatur

```
GenerateCodeInEx(in i_nLanguage:ENUMProgrammingLanguage1916) -> ErrorMarkers
```

Allgemeine Signatur

Parameter

Name	Typ	Beschreibung
i_nLanguage	ENUMProgrammingLanguage1916	Definiert die Programmiersprache für die Codegenerierung.

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Es wurde für den Rückgabeparameter eine ungültige Sprache oder eine ungültige Adresse angegeben.
1706	Fehler bei der Codegenerierung.

17.7.1.20.2.9 Open

Öffnet das Projektelement als Dokument oder macht das entsprechende Dokument zum aktiven, wenn es bereits geöffnet ist. Das Projektelement muss ein MapForce-Mapping sein oder - nur bei der Enterprise Edition - eine Webservice-Operation.

Signatur

```
Open() -> Document
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.
1701	Für den Rückgabeparameter wurde eine ungültige Adresse angegeben.
1702	Das Projektelement bezieht sich auf keine MapForce Mapping-Datei.

© 2016-2022 Altova GmbH

Fehlercode	Beschreibung
1708	Die Operation wird in der aktuellen Edition nicht unterstützt.

17.7.1.20.2.10 Remove

Entfernt dieses Projektelement und alle seine Child-Elemente aus der Projektstruktur.

Signatur

```
Remove() -> Void
```

Allgemeine Signatur

Fehler

Fehlercode	Beschreibung
1700	Das Objekt ist nicht mehr gültig.

17.7.1.20.3 Events

17.7.1.20.3.1 OnModifiedFlagChanged

Kommt vor, wenn sich der Änderungsstatus von `ProjectItem` ändert.

Signatur

```
OnModifiedFlagChanged(in i_bIsModified:Boolean) : Void
```

17.7.1.20.3.2 OnProjectClosed

Dieses Event wird beim Schließen des Projekts ausgelöst. Das an den Event Handler übergebene Projektobjekt sollte nicht aufgerufen werden. Das entsprechende Event zum Öffnen ist `Application.OnProjectOpened`.

Signatur

```
OnProjectClosed(in i_ipProject:Project) : Void
```

17.7.2 Enumerationen

17.7.2.1 ENUMApacheAxisVersion (obsolete)

Dieser Enumerationstyp wird nicht mehr verwendet.

Members

eApacheAxisVersion_Axis = 1

eApacheAxisVersion_Axis2 = 2

17.7.2.2 ENUMApplicationStatus

Enumerationswerte zur Angabe des Status der Applikation.

Members

eApplicationRunning = 0

eApplicationAfterLicenseCheck = 1

eApplicationBeforeLicenseCheck = 2

eApplicationConcurrentLicenseCheckFailed = 3

eApplicationProcessingCommandLine = 4

17.7.2.3 ENUMAppOutputLine_Severity

Enumerationswerte zur Angabe des Schweregrads einer `AppOutputLine`.

Members

eSeverity_Undefined = -1

eSeverity_Info = 0

eSeverity_Warning = 1

eSeverity_Error = 2

eSeverity_CriticalError = 3

eSeverity_Success = 4

eSeverity_Summary = 5

© 2016-2022 Altova GmbH

eSeverity_Progress = 6

eSeverity_DataEdit = 7

eSeverity_ParserInfo = 8

eSeverity_PossibleInconsistencyWarning = 9

eSeverity_Message = 10

eSeverity_Document = 11

eSeverity_Rest = 12

eSeverity_NoSelect = 13

eSeverity_Select = 14

eSeverity_Autoinsertion = 15

eSeverity_GlobalResources_DefaultWarning = 16

eSeverity_XPath_Styles_Changed = 17

eSeverity_XPath_Styles_Unchanged = 18

eSeverity_XPath_Styles_Skipped = 19

eSeverity_XPath_ComboBox_Values_Changed = 20

eSeverity_XPath_ComboBox_Values_Unchanged = 21

eSeverity_XPath_ComboBox_Values_Skipped = 22

eSeverity_XPath_Assertions_Changed = 23

eSeverity_XPath_Assertions_Unchanged = 24

eSeverity_XPath_Assertions_Skipped = 25

17.7.2.4 ENUMAppOutputLine_TextDecoration

Enumerationswerte für die verschiedenen Arten der Textdekoration eines `AppOutputLine`.

Members

eTextDecorationDefault = 0

eTextDecorationBold = 1

eTextDecorationDebugValues = 2

eTextDecorationDB_ObjectName = 3

eTextDecorationDB_ObjectLink = 4

eTextDecorationDB_ObjectKind = 5

eTextDecorationDB_TimeoutValue = 6

eTextDecorationFind_MatchingString = 7

eTextDecorationValidation_Speclink = 8

eTextDecorationValidation_ErrorPosition = 9

eTextDecorationValidation_UnkownParam = 10

17.7.2.5 ENUMCodeGenErrorLevel

Enumerationswerte zur Angabe des Schweregrads von Codegenerierungsmeldungen.

Members

eCodeGenErrorLevel_Information = 0

eCodeGenErrorLevel_Warning = 1

eCodeGenErrorLevel_Error = 2

eCodeGenErrorLevel_Undefined = 3

17.7.2.6 ENUMComponentDatapointSide

Enumerationswerte zur Angabe der Seite eines Datapoint in seiner Komponente. Siehe auch `Component.GetRootDatapoint`.

Members

eDatapointSideInput = 0

eDatapointSideOutput = 1

17.7.2.7 ENUMComponentSubType

Enumerationswerte zur Angabe der Komponentensubtypen.

Members

eComponentSubType_None = 0

eComponentSubType_Text_EDI = 1

eComponentSubType_Text_Flex = 2

© 2016-2022 Altova GmbH

eComponentSubType_Text_CSVFLF = 3

17.7.2.8 ENUMComponentType

Enumerationswerte zur Angabe der Komponententypen.

Members

eComponentType_Unknown = 0

eComponentType_XML = 1

eComponentType_DB = 2

eComponentType_Text = 3

eComponentType_Excel = 4

eComponentType_WSDL = 5

eComponentType_XBRL = 6

eComponentType_Input = 7

eComponentType_JSON = 8

17.7.2.9 ENUMComponentUsageKind

Enumerationswerte zur Angabe der Komponentenverwendungsart.

Members

eComponentUsageKind_Unknown = 0

eComponentUsageKind_Instance = 1

eComponentUsageKind_Input = 2

eComponentUsageKind_Output = 3

eComponentUsageKind_Variable = 4

eComponentUsageKind_String = 5

17.7.2.10 ENUMConnectionType

Enumerationswerte zur Angabe des Verbindungstyps. Siehe auch `Connection.ConnectionType`.

Members

eConnectionTypeTargetDriven = 0

eConnectionTypeSourceDriven = 1

eConnectionTypeCopyAll = 2

17.7.2.11 ENUMDOMType

Enumerationswerte zur Definition des vom generierten C++ Mapping-Code verwendeten DOM-Typs.

ANMERKUNG: Der Wert `eDOMType_xerces` wird nicht mehr verwendet. `eDOMType_xerces3` zeigt an, dass Xerces 3.x verwendet wird. "Nicht mehr verwendet" bedeutet in diesem Zusammenhang, dass dieser Wert nicht unterstützt wird und nicht verwendet werden sollte.

Members

eDOMType_xerces = 1 (obsolete)

eDOMType_xerces3 = 2

eDOMType_msxml6 = 3

17.7.2.12 ENUMLibType

Enumerationswerte zur Definition des vom generierten C++ Mapping-Code verwendeten Bibliothekstyps.

Members

eLibType_static = 0

eLibType_dll = 1

17.7.2.13 ENUMProgrammingLanguage

Enumerationswerte zur Auswahl einer Programmiersprache.

Members

eUndefinedLanguage = -1

© 2016-2022 Altova GmbH

eJava = 0

eCpp = 1

eCSharp = 2

eXSLT = 3

eXSLT2 = 4

eXQuery = 5

eXSLT3 = 6

17.7.2.14 ENUMProjectItemType

Enumerationswerte zur Angabe der verschiedenen Arten von Projektelementen, die Children von `Project` oder ordnerähnlichen `ProjectItems` sein können. Siehe auch `ProjectItem.Kind`.

Members

eProjectItemType_MappingFolder = 0

eProjectItemType_Mapping = 1

eProjectItemType_WebServiceFolder = 2

eProjectItemType_WebServiceRoot = 3

eProjectItemType_WebServiceService = 4

eProjectItemType_WebServicePort = 5

eProjectItemType_WebServiceOperation = 6

eProjectItemType_ExternalFolder = 7

eProjectItemType_LibraryFolder = 8

eProjectItemType_ResourceFolder = 9

eProjectItemType_VirtualFolder = 10

eProjectItemType_Count = 11

eProjectItemType_Invalid = -1

17.7.2.15 ENUMProjectType

Enumerationswerte zur Auswahl eines Projekttyps für generierten C#- und C++-Mapping-Code.

Members

eVisualStudio2010Project = 6

eVisualStudio2013Project = 7

eVisualStudio2015Project = 8

eVisualStudio2017Project = 9

eVisualStudio2019Project = 10

eDotNetCore3_1 = 11

eDotNet5_0 = 12

17.7.2.16 ENUMSearchDatapointFlags

Enumerationswerte, die als Bit-Flags verwendet werden; zur Verwendung als Kombination von Flags beim Suchen nach einem Datapoint. Siehe auch `GetChild`.

Members

eSearchDatapointElement = 1

eSearchDatapointAttribute = 2

17.7.2.17 ENUMViewMode

Enumerationswerte zur Auswahl einer MapForce-Ansicht.

Members

eMapForceView = 0

eXSLView = 1

eOutputView = 2

© 2016-2022 Altova GmbH

18 ActiveX Integration

Die in diesem Abschnitt beschriebene MapForce-Benutzeroberfläche und deren Funktionalitäten können in benutzerdefinierte Applikationen integriert werden, die ActiveX Controls verwenden können. Mit Hilfe der ActiveX-Technologie können die verschiedensten Programmiersprachen wie z.B. C++, C#, VB.NET oder HTML für die Integration verwendet werden. (Beachten Sie, dass in HTML integrierte ActiveX-Komponenten mit Microsoft Internet Explorer-Versionen und Plattformen, die ActiveX unterstützen, ausgeführt werden müssen). Alle Komponenten sind vollständige OLE Controls, Die Integration in Java wird durch Wrapper-Klassen möglich gemacht.

Um ActiveX Controls in Ihren benutzerdefinierten Code zu integrieren, müssen Sie das MapForce-Integrationspaket installieren (siehe https://www.altova.com/de/components/download). Stellen Sie sicher, dass Sie zuerst MapForce installieren und dann erst das MapForce-Integrationspaket. Je nach Sprache und Plattform gelten andere Voraussetzungen (siehe Voraussetzungen 1920).

Sie haben die Wahl zwischen zwei verschiedenen Ebenen der Integration: auf Applikations- und auf Dokumentebene.

Bei einer Integration auf Applikationsebene wird die komplette Benutzeroberfläche von MapForce (einschließlich aller Menüs, Symbolleisten, Fenster usw.) als ActiveX Control in Ihre benutzerdefinierte Applikation eingebettet. So könnte Ihre benutzerdefinierte Applikation im einfachsten Szenario z.B. aus nur einem Formular bestehen, in das die grafische Benutzeroberfläche von MapForce eingebettet ist. Diese Methode ist einfacher zu implementieren als die Integration auf Dokumentebene, ist aber möglicherweise nicht geeignet, wenn Sie die grafische Benutzeroberfläche von MapForce Ihren Anforderungen gemäß flexibel konfigurieren möchten.

Bei der Integration auf Dokumentebene wird MapForce Stück für Stück in Ihre eigene Applikation eingebettet. Dabei werden nicht nur das MapForce Haupt-Control, sondern auch das Dokument-Editor-Hauptfenster und optional zusätzliche Fenster implementiert. Bei dieser Methode haben Sie größere Flexibilität beim Konfigurieren der grafischen Benutzeroberfläche, es ist aber mehr Interaktion mit den ActiveX Controls der Sprache Ihrer Wahl erforderlich.

In den Abschnitten Integration auf Applikationsebene 1924 und Integration auf Dokumentebene 1927 werden die grundlegenden Schritte auf diesen Ebenen beschrieben. Im Abschnitt Beispiele zur ActiveX-Integration 1931 finden Sie Beispiele in C#, HTML und Java. Diese sollen Ihnen dabei helfen, rasch die richtige Entscheidung zu treffen. Der Abschnitt Objektreferenz 1959 enthält eine Beschreibung aller für die Integration verwendbaren COM-Objekte mit ihren Eigenschaften und Methoden.

Informationen zur Verwendung von MapForce als Visual Studio Plug-in finden Sie unter MapForce in Visual Studio 1492.

18.1 Voraussetzungen

Um das MapForce ActiveX Control in eine benutzerdefinierte Applikation zu integrieren, müssen die folgenden Programme auf Ihre Computer installiert sein:

- MapForce
- Das MapForce Integrationspaket, das Sie von https://www.altova.com/de/components/download herunterladen können.

Um das 64-Bit ActiveX Control zu integrieren, installieren Sie die 64-Bit-Version von MapForce und dem MapForce Integrationspaket. Für Applikationen, die mit Hilfe von Visual Studio unter der Microsoft .NET-Plattform entwickelt wurden, müssen, wie unten erläutert, die 32- und die 64-Bit-Version von MapForce und dem MapForce Integrationspaket installiert sein.

Microsoft .NET (C#, VB.NET) mit Visual Studio

Um das MapForce ActiveX Control in eine unter Microsoft .NET entwickelte 32-Bit-Applikation zu integrieren, müssen die folgenden Programme auf Ihrem Computer installiert sein:

- Microsoft .NET Framework 4.0 oder höher
- Visual Studio 2012/2013/2015/2017/2019/2022
- MapForce 32-Bit und MapForce Integrationspaket 32-Bit
- Die ActiveX Controls müssen zur Visual Studio Toolbox hinzugefügt werden (siehe Hinzufügen der ActiveX Controls zur Toolbox 1922).

Falls Sie das 64-Bit ActiveX Control integrieren möchten, sind zusätzlich zu den oben erwähnten Voraussetzungen die folgenden erforderlich:

- MapForce 32-Bit und das MapForce Integrationspaket 32-Bit müssen weiterhin installiert sein (dies ist erforderlich, damit das 32-Bit ActiveX Control dem Visual Studio Designer zur Verfügung steht, da Visual Studio unter 32-Bit läuft)
- MapForce 64-Bit und das MapForce Integrationspaket 64-Bit müssen installiert sein (stellt Ihrer benutzerdefinierten Applikation zur Laufzeit das eigentliche 64-Bit ActiveX Control zur Verfügung)
- Erstellen Sie in Visual Studio eine 64-Bit Build-Konfiguration und bauen Sie Ihre Applikation mit Hilfe dieser Konfiguration. Ein Beispiel dazu finden Sie unter Ausführen der C#-Beispiellösung 1931.

Java

Um das MapForce ActiveX Control über die Eclipse-Entwicklungsumgebung in die Java-Applikation zu integrieren, müssen die folgenden Programme auf Ihrem Computer installiert sein:

- Java Runtime Environment (JRE) oder Java Development Kit (JDK) 7 oder höher
- Eclipse
- MapForce und das MapForce Integrationspaket

Anmerkung: Verwenden Sie zur Ausführung der 64-Bit-Version des MapForce ActiveX Control eine 64-Bit-Version von Eclipse sowie eine 64-Bit-Version von MapForce und dem MapForce Integrationspaket.

© 2016-2022 Altova GmbH

MapForce Integration und Bereitstellung auf Client-Rechnern

Wenn Sie eine .NET-Applikation erstellen und beabsichtigen, diese auf anderen Client-Rechnern zur Verfügung zu stellen, müssen auf dem/den Client-Rechner(n) die folgenden Programme installiert sein:

- MapForce
- Das MapForce Integrationspaket
- Der benutzerdefinierte Integrationscode oder die benutzerdefinierte Applikation.

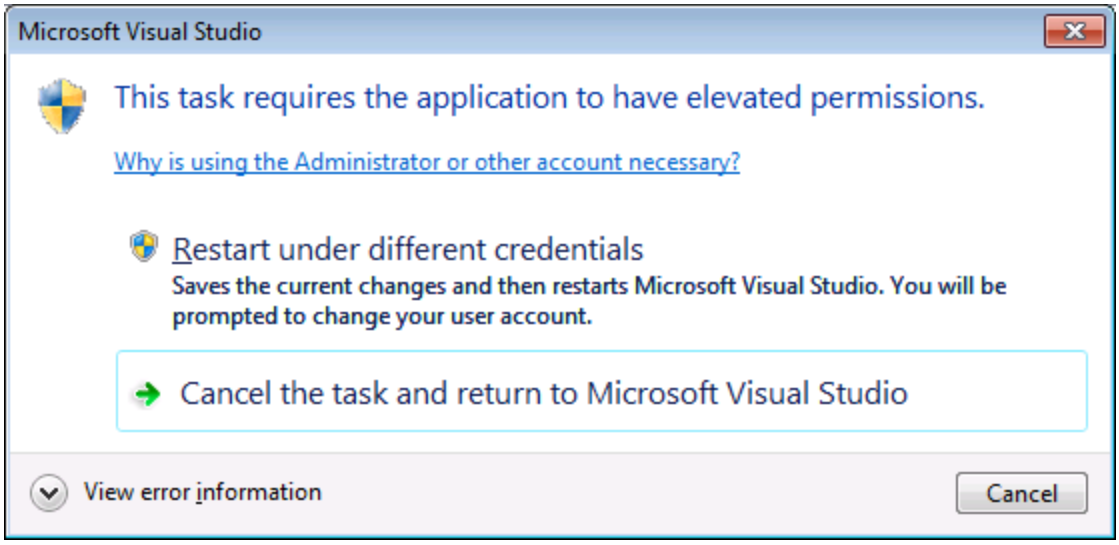
18.2 Hinzufügen der ActiveX Controls zur Toolbox

Um die MapForce ActiveX Controls in einer mit Visual Studio entwickelten Applikation verwenden zu können, fügen Sie diese folgendermaßen zu Ihrer Visual Studio Toolbox hinzu:

1. Klicken Sie im Menü **Tools** von Visual Studio auf **Choose Toolbox Items**.
2. Aktivieren Sie auf dem Register **COM Components** die Kontrollkästchen neben dem MapForceControl, dem MapForceControl-Dokument und dem MapForceControl-Platzhalter.

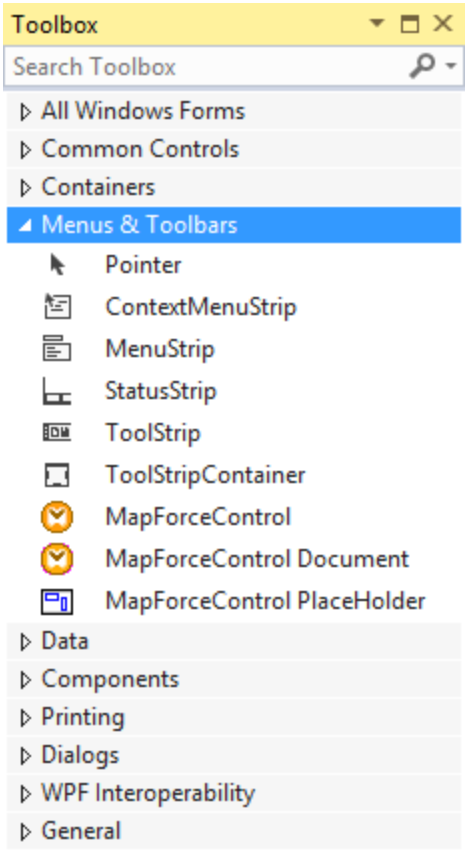
Falls die oben erwähnten Controls nicht zur Verfügung stehen, gehen Sie folgendermaßen vor:

1. Klicken Sie auf dem Register **COM Components** auf **Browse** und wählen Sie die Datei **MapForceControl.ocx** aus dem MapForce-Installationsordner aus. Beachten Sie, dass das MapForce-Integrationspaket installiert sein muss, da die Datei sonst nicht zur Verfügung steht (siehe Voraussetzungen 1920).
3. Sobald Sie aufgefordert werden, Visual Studio mit erweiterten Berechtigungen zu starten, klicken Sie auf **Restart under different credentials**.



Wenn die obigen Schritte erfolgreich ausgeführt wurden, stehen die MapForce ActiveX Controls in der Visual Studio Toolbox zur Verfügung.

© 2016-2022 Altova GmbH



Anmerkung: Bei einer Integration auf Applikationsebene wird nur das **MapForceControl** ActiveX Control verwendet (siehe Integration auf Applikationsebene[1924]). Die Controls **MapForceControl Document** und **MapForceControl Placeholder** werden für die Integration auf Dokumentebene verwendet (siehe Integration auf Dokumentebene[1927]).

18.3 Integration auf Applikationsebene

Bei der Integration auf Applikationsebene können Sie die gesamte Benutzeroberfläche von MapForce in ein Fenster Ihrer Applikation einbetten. Bei dieser Art von Integration steht Ihnen die gesamte Benutzeroberfläche von MapForce einschließlich aller Menüs, Symbolleisten, der Statusleiste, der Dokumentfenster und Eingabehilfen zur Verfügung. Die Anpassung der Benutzeroberfläche der Applikation ist auf die Optionen eingeschränkt, die MapForce bietet. Dazu gehören die Neuanordnung und Anpassung der Größe der Eingabehilfen und die Anpassung von Menüs und Symbolleisten.

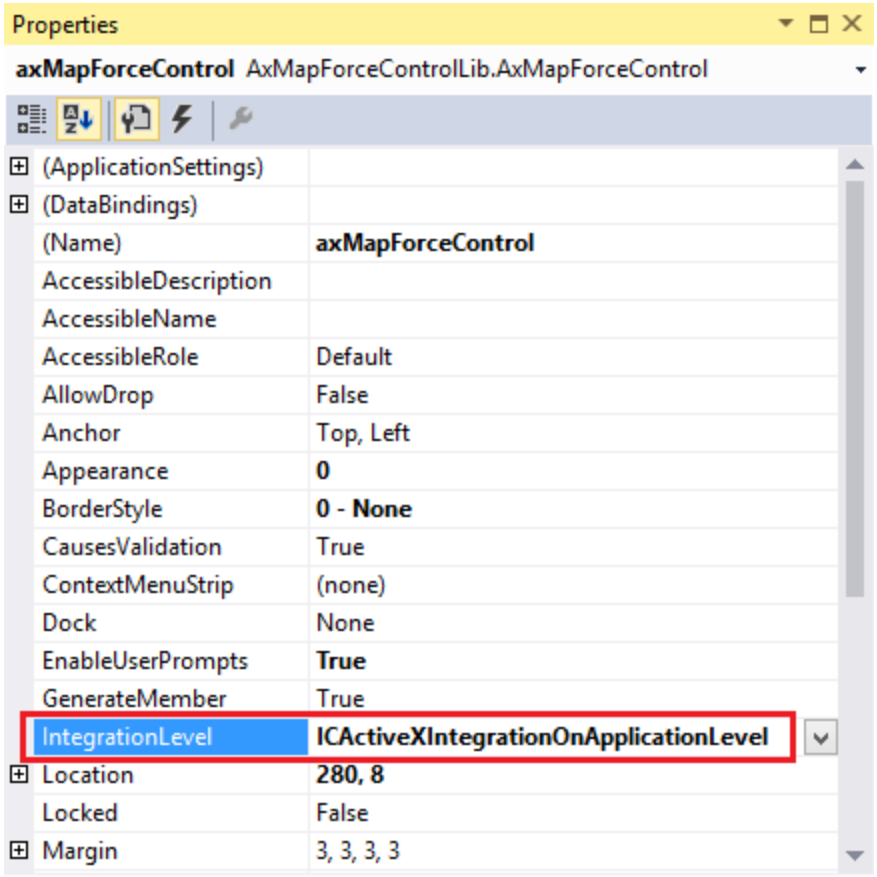
Das einzige ActiveX Control, das Sie integrieren müssen, ist `MapForceControl`[1962]. Bei Integration auf Applikationsebene dürfen `MapForceControlDocument`[1970] oder `MapForceControlPlaceHolder`[1977] ActiveX Controls nicht instantiiert oder aufgerufen werden.

Wenn Sie Initialisierungen vornehmen müssen oder ein bestimmtes Verhalten von MapForce automatisieren wollen, verwenden Sie die für `MapForceControl`[1962] beschriebenen Eigenschaften, Methoden und Events. Um komplexere MapForce Funktionsaufrufe auszuführen, sollten Sie eventuell `MapForceControl.Application`[1963] verwenden.

Gehen Sie in C# oder VB.NET bei Verwendung von Visual Studio folgendermaßen vor, um eine aus einem Formular bestehende Applikation zu erstellen, in der die MapForce ActiveX Controls auf Applikationsebene integriert sind:

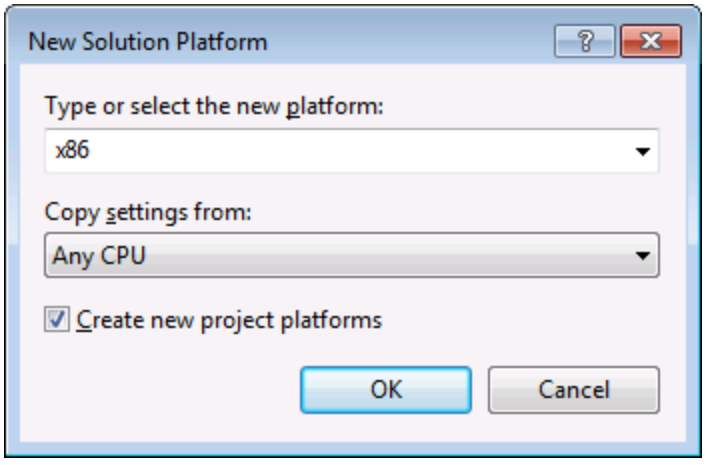
1. Überprüfen Sie, ob alle Voraussetzungen erfüllt werden (siehe Voraussetzungen[1920]).
2. Erstellen Sie ein Visual Studio Windows Forms-Projekt mit einem neuen leeren Formular.
3. Fügen Sie die ActiveX Controls zur Toolbox hinzu, falls Sie das noch nicht getan haben (siehe Hinzufügen der ActiveX Controls zur Toolbox[1922]).
4. Ziehen Sie das **MapForceControl** aus der Toolbox in Ihr neues Formular.
5. Wählen Sie das **MapForceControl** im Formular aus und definieren Sie im Fenster "Properties" für die Eigenschaft **IntegrationLevel** den Wert **ICActiveXIntegrationOnApplicationLevel**.

© 2016-2022 Altova GmbH



6. Erstellen Sie eine Build-Plattform-Konfiguration für die Plattform, unter der Sie die Projektmappe erstellen möchten (x86, x64). So erstellen Sie die Build-Konfiguration:

   a. Klicken Sie mit der rechten Maustaste in Visual Studio auf die Projektmappen und wählen Sie **Configuration Manager**.
   b. Wählen Sie unter **Active solution platform** den Befehl **New...** und wählen Sie anschließend die x86- oder x64-Konfiguration aus (in diesem Beispiel **x86**).



Sie sind nun fertig und können die Projektmappe in Visual Studio erstellen und ausführen. Denken Sie daran, die Projektmappe mit der richtigen Konfiguration für Ihre Zielplattform (x86, x64) zu erstellen.

© 2016-2022 Altova GmbH

18.4 Integration auf Dokumentebene

Im Vergleich zur Integration auf Applikationsebene ist die Integration auf Dokumentebene komplexer, bietet aber dafür mehr Flexibilität beim Einbetten von MapForce-Funktionalitäten in Ihre Applikation mit Hilfe von ActiveX Controls. Auf diese Art haben Sie selektiven Zugriff auf die folgenden Teile der MapForce-Benutzeroberfläche:

- Dokument-Bearbeitungsfenster
- Projektfenster
- Bibliotheksfenster
- Übersichtsfenster
- Fenster "Meldungen"

Wie bereits im Abschnitt Integration auf Applikationsebene[1924] erwähnt, ist für eine ActiveX-Integration auf Applikationsebene nur ein Control, nämlich das **MapForceControl**, erforderlich. Für eine ActiveX-Integration auf Dokumentebene werden die MapForce-Funktionalitäten jedoch durch die folgenden ActiveX Controls zur Verfügung gestellt:

- MapForceControl[1962]
- MapForceControl Document[1970]
- MapForceControl Placeholder[1977]

Diese Controls werden durch die Datei **MapForceControl.ocx** aus dem Applikationsinstallationsordner von MapForce zur Verfügung gestellt. Wenn Sie die ActiveX-Integration mit Hilfe von Visual Studio erstellen, benötigen Sie über die Visual Studio Toolbox Zugriff auf diese Controls (siehe Hinzufügen der ActiveX Controls zur Toolbox[1922]).

Die grundlegenden Schritte, um die ActiveX Controls auf Dokumentebene in Ihre Applikation zu integrieren sind die folgenden:

1. Instantiieren Sie in Ihrer Applikation zuerst **MapForceControl**. Die Instantiierung dieses Control ist obligatorisch; Es ermöglicht die Unterstützung für die oben erwähnten Controls **MapForceControl Document** und **MapForceControl Placeholder**. Die Eigenschaft `IntegrationLevel`[1964] muss auf **ICActiveXIntegrationOnDocumentLevel** (oder "1") gesetzt werden. Um das Control für den Benutzer auszublenden, setzen Sie seine Eigenschaft **Visible** auf **False**.

Anmerkung: Verwenden Sie für die Integration auf Dokumentebene nicht die **Open**-Methode des **MapForceControl**, da dies zu unerwünschten Ergebnissen führen könnte. Verwenden Sie stattdessen die entsprechenden open-Methoden von **MapForceControl Document** und **MapForceControl PlaceHolder**.

2. Erstellen Sie mindestens eine Instanz von **MapForceControl Document** in Ihrer Applikation. Dieses Control stellt Ihrer Applikation das Dokumentbearbeitungsfenster von MapForce zur Verfügung und kann bei Bedarf mehrmals instantiiert werden.

   Verwenden Sie die Methode **Open,** um eine vorhandene Datei zu laden. Um Funktionen im Zusammenhang mit einem Dokument aufzurufen, verwenden Sie die Methoden **Path** und **Save** oder Methoden und Eigenschaften, die über die Eigenschaft **Document** zur Verfügung stehen.

Anmerkung: Das Control unterstützt keinen schreibgeschützten Modus. Der Wert der Eigenschaft **ReadOnly** wird ignoriert.

3. Fügen Sie optional für jedes weitere Fenster (bei dem es sich nicht um das Dokumentfenster handelt) und das Ihrer Applikation zur Verfügung stehen soll, den **MapForceControl Placeholder** zur Ihrer Applikation hinzu.

   Mit Hilfe von Instanzen von **MapForceControl PlaceHolder** können Sie selektiv zusätzliche Fenster von MapForce in Ihre Applikation einbetten. Die Art des Fensters (z.B. Projektfenster) wird durch die Eigenschaft **PlaceholderWindowID** definiert. Um daher die Art des Fensters zu definieren, definieren Sie die Eigenschaft **PlaceholderWindowID**. Gültige Fenster-IDs finden Sie unter MapForceControlPlaceholderWindow[1980].

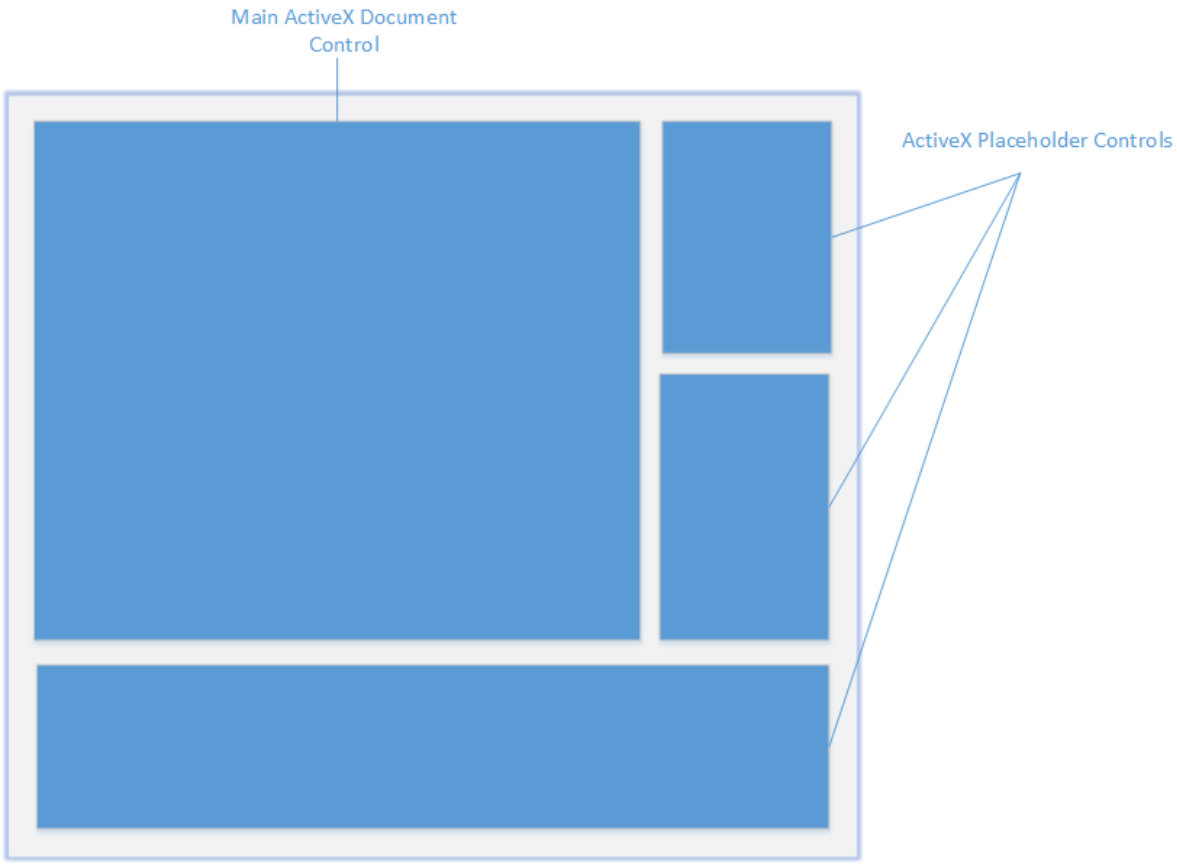
Anmerkung: Verwenden Sie für jeden Windows Identifier nur einen **MapForceControl PlaceHolder**.

Für Placeholder Controls, die das MapForce-Projektfenster auswählen, stehen zusätzliche Methoden zur Verfügung. Mit Hilfe von **OpenProject** können Sie ein MapForce-Projekt laden. Mit Hilfe der Eigenschaft "Project" und der Methoden und Eigenschaften aus der MapForce Automation-Schnittstelle können Sie andere Operationen im Zusammenhang mit einem Projekt durchführen.

Ein Beispiel, wie Sie in C# oder VB.NET mit Visual Studio eine einfache, aus einem Formular bestehende Applikation, in der die MapForce ActiveX Controls auf Dokumentebene integriert sind, erstellen, sehen Sie in der Beschreibung unten. Beachten Sie, dass Ihre Applikation bei Bedarf auch komplexer sein kann. In der Anleitung unten sehen Sie jedoch, welche Voraussetzungen mindestens gegeben sein müssen, damit eine ActiveX-Integration auf Dokumentebene erfolgen kann.

1. Erstellen Sie ein neues Visual Studio Windows Forms-Projekt mit einem neuen leeren Formular.
2. Fügen Sie die ActiveX Controls zur Toolbox hinzu, falls dies noch nicht geschehen ist (siehe Hinzufügen der ActiveX Controls zur Toolbox[1922]).
3. Ziehen Sie das **MapForceControl**[1962] aus der Toolbox in Ihr neues Formular.
4. Setzen Sie die Eigenschaft **IntegrationLevel** des **MapForceControl** auf **ICActiveXIntegrationOnDocumentLevel** und die Eigenschaft **Visible** auf **False**. Sie können dies entweder über den Code oder über das Fenster **Properties** tun.
5. Ziehen Sie das **MapForceControl Document**[1970] aus der Toolbox in das Formular. Dieses Control stellt das Dokument-Hauptfenster von MapForce für Ihre Applikation zur Verfügung. Eventuell müssen Sie die Größe des Fensters an die Ihres Dokuments anpassen.
6. Fügen Sie optional ein oder mehrere **MapForceControl Placeholder**[1977] Controls zum Formular hinzu (eines für jeden zusätzlichen Fenstertyp, der für Ihre Applikation benötigt wird, z.B. das Projektfenster). Solche zusätzlichen Placeholder Controls werden normalerweise entweder unterhalb oder rechts oder links vom Dokument-Haupt-Control platziert z.B.:

© 2016-2022 Altova GmbH



7. Setzen Sie die Eigenschaft **PlaceholderWindowID** jedes **MapForceControl Placeholder** auf einen gültigen Fenster-Identifier. Eine Liste der gültigen Werte finden Sie unter MapForceControlPlaceholderWindow[1980].
8. Fügen Sie, wie unten gezeigt, Befehle zu Ihrer Applikation hinzu (Sie benötigen zumindest Befehle zum Öffnen, Speichern und Schließen von Dokumenten).

Abfragen von MapForce-Befehlen

Bei einer Integration auf Dokumentebene stehen in Ihrer Applikation kein MapForce-Menü- und keine MapForce-Symbolleiste zur Verfügung. Sie können statt dessen die benötigten Befehle aufrufen, deren Status anzeigen und diese programmatisch ausführen, wie folgt:

- Um alle verfügbaren Befehle abzurufen, verwenden Sie die Eigenschaft `CommandsList`[1964] des **MapForceControl**.
- Um die Befehle, geordnet nach ihrer Menüstruktur abzurufen, verwenden Sie die Eigenschaft `MainMenu`[1965].
- Um die Befehle, geordnet nach der Symbolleiste, in der sie vorkommen, abzurufen, verwenden Sie die Eigenschaft `Toolbars`[1965].
- Um Befehle an MapForce zu senden, verwenden Sie die Methode `Exec`[1966].
- Um abzufragen, ob ein Befehl gerade aktiviert oder deaktiviert ist, verwenden Sie die Methode `QueryStatus`[1967].

Auf diese Art können Sie MapForce-Befehle flexibel in die Menüs und Symbolleisten Ihrer Applikation integrieren.

Über Ihre Installation von MapForce stehen Ihnen auch Symbole für in MapForce verwendete Befehle zur Verfügung. Im Ordner **<Applicationsordner>\Examples\ActiveX\Images** Ihrer MapForce-Installation finden Sie Symbole im GIF-Format. Die Dateinamen entsprechen den im Abschnitt Befehlsreferenz[1950] aufgelisteten Befehlsnamen.

Allgemeines

Um das Verhalten von MapForce zu automatisieren, verwenden Sie die für **MapForceControl**[1962], **MapForceControl Document**[1970] und **MapForceControl Placeholder**[1977] beschriebenen Eigenschaften, Methoden und Ereignisse.

Für einen komplexeren Zugriff auf MapForce-Funktionalitäten sollten Sie die folgenden Eigenschaften verwenden:

- `MapForceControl.Application`[1963]
- `MapForceControlDocument.Document`[1972]
- `MapForceControlPlaceHolder.Project`[1978]

Über diese Eigenschaften erhalten Sie Zugriff auf die MapForce Automation-Schnittstelle (<%APPAPI%)

Anmerkung: Verwenden zum Öffnen eines Dokuments immer `MapForceControlDocument.Open`[1974] oder `MapForceControlDocument.New`[1973] für das entsprechenden Dokument-Control. Verwenden Sie zum Öffnen eines Projekts immer `MapForceControlPlaceHolder.OpenProject`[1979] in einem Placeholder Control, das ein MapForce-Projektfenster einbettet.

Beispiele, wie Sie die erforderlichen Controls in unterschiedlichen Programmierumgebungen instantiieren und aufrufen finden Sie unter Beispiele zur ActiveX-Integration[1931].

© 2016-2022 Altova GmbH

18.5 Beispiele zur ActiveX-Integration

Dieser Abschnitt enthält Beispiele für die Integration von MapForce auf Dokumentebene über unterschiedliche Container-Umgebungen und Programmiersprachen. Der Quellcode für alle Beispiele steht im Ordner `<ApplicationFolder>\Examples\ActiveX` Ihrer MapForce Installation zur Verfügung.

18.5.1 C#

Im Ordner `<ApplicationFolder>\Examples\ActiveX\C#` finden Sie ein einfaches Beispiel für eine ActiveX-Integration mittels C# und Visual Studio. Bevor Sie den Quellcode kompilieren und das Beispiel ausführen, stellen Sie sicher, dass alle Voraussetzungen erfüllt werden (siehe Ausführen der C#-Beispiellösung 1931).

18.5.1.1 Ausführen der C#-Beispiellösung

In der Visual Studio-Beispielprojektmappe im Ordner **<ApplicationFolder>\Examples\ActiveX\C#** wird gezeigt, wie die MapForce ActiveX Controls verwendet werden. Beachten Sie die folgenden Schritte, bevor Sie versuchen, diese Projektmappen zu erstellen und auszuführen:

Schritt 1: Überprüfen Sie die Voraussetzungen

Um die Beispielprojektmappen öffnen zu können, benötigen Sie Visual Studio 2010 oder höher. Eine Liste der Voraussetzungen finden Sie unter Voraussetzungen 1920.

Schritt 2: Kopieren Sie das Beispiel in ein Verzeichnis, auf das Sie Schreibzugriff haben

Um Visual Studio nicht als Administrator ausführen zu müssen, kopieren Sie den Quellcode in ein Verzeichnis, auf das Sie Schreibzugriff haben, anstatt ihn vom Standardverzeichnis aus auszuführen.

Schritt 3: Überprüfen und definieren Sie alle erforderlichen Control-Eigenschaften

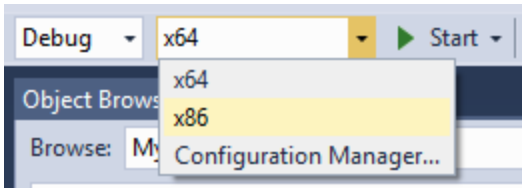
Die Beispielapplikation enthält eine Instanz von MapForceControlDocument 1970 und mehrere Instanzen von MapForceControlPlaceHolder 1977 Controls. Überprüfen Sie noch einmal, ob die folgenden Eigenschaften dieser Controls, wie in der Tabelle unten angegeben, konfiguriert sind:

Control-Name	Eigenschaft	Eigenschaftswert
axMapForceControl	IntegrationLevel	ICActiveXIntegrationOnDocumentLevel
axMapForceControlLibrary	PlaceholderWindowID	0
axMapForceControlOutput	PlaceholderWindowID	2
axMapForceControlPreview	PlaceholderWindowID	1

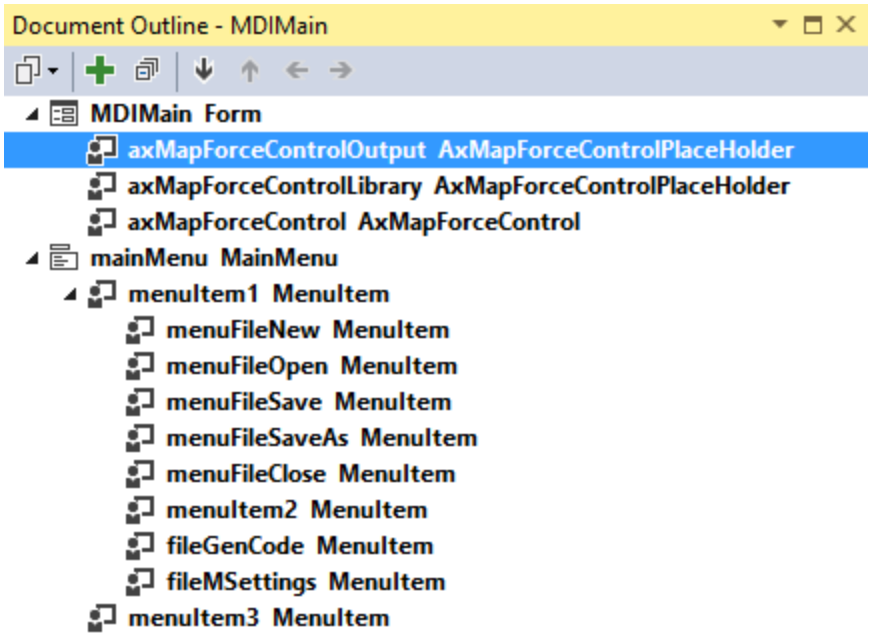
So können Sie die Eigenschaften eines ActiveX Control anzeigen oder definieren:

1. Öffnen Sie das Formular **MDIMain.cs** im Designer-Fenster.

Anmerkung: Unter 64-Bit Windows müssen Sie die Build-Konfiguration der Visual Studio-Projektmappen eventuell in "x86" ändern, **bevor** Sie das Designer-Fenster öffnen. Wenn Sie das Beispiel als 64-Bit-Applikation erstellen müssen, schlagen Sie nach unter Voraussetzungen[1920].

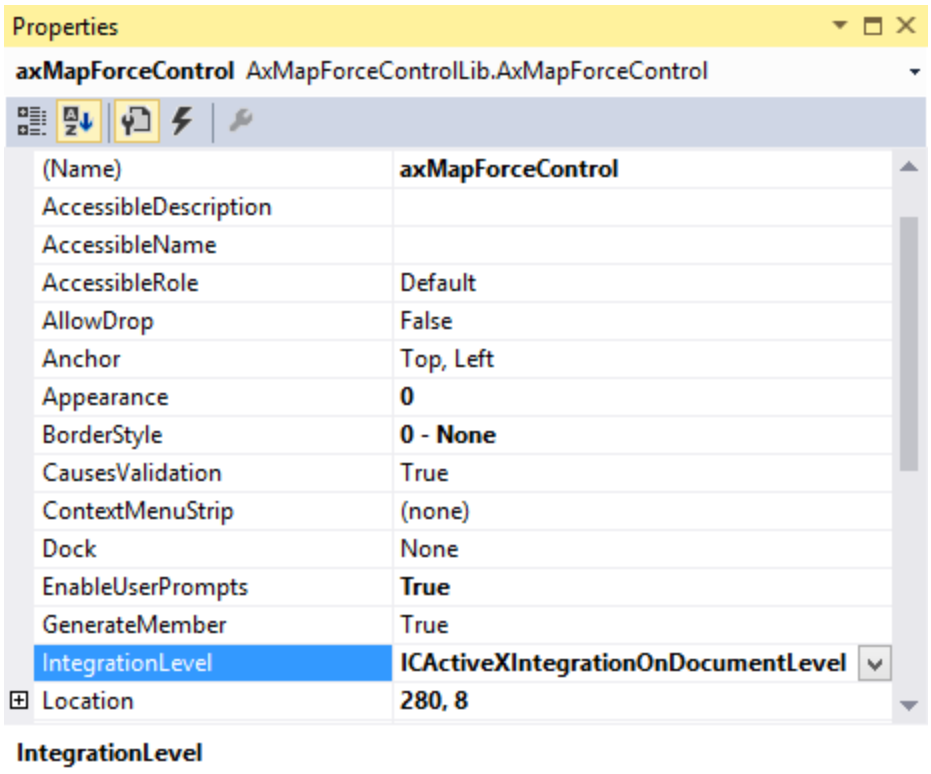


2. Öffnen Sie das **Document Outline**-Fenster von Visual Studio (Klicken Sie im Menü **View** auf **Other Windows | Document Outline**).



3. Klicken Sie im Fenster **Document Outline** auf ein ActiveX Control und bearbeiten Sie die benötigte Eigenschaft im Fenster **Properties,** z.B.:

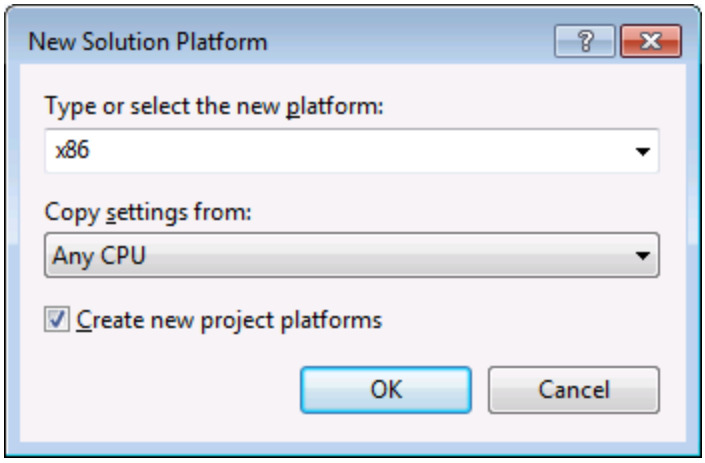
© 2016-2022 Altova GmbH



Schritt 4: Definieren Sie die Build-Plattform

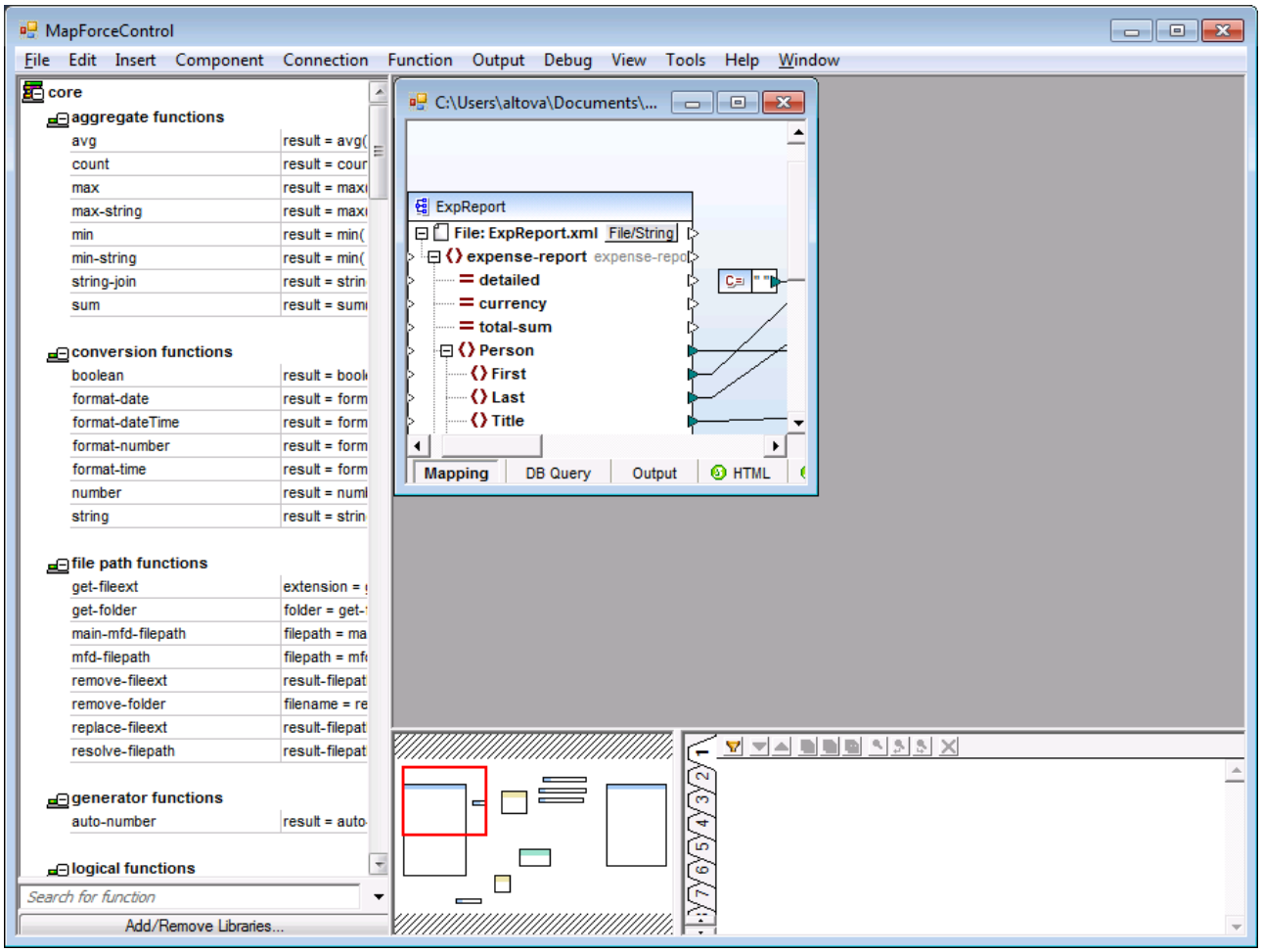
- Erstellen Sie eine Build-Plattform-Konfiguration für die Plattform, unter der Sie die Projektmappe erstellen möchten (x86, x64). So erstellen Sie die Build-Konfiguration:

  a. Klicken Sie mit der rechten Maustaste auf die Projektmappe in Visual Studio und wählen Sie **Configuration Manager**.
  b. Wählen Sie unter **Active solution platform** den Befehl **New...** und wählen Sie anschließend die x86- oder x64-Konfiguration (in diesem Beispiel **x86**).



Sie sind nun fertig und können die Projektmappe in Visual Studio erstellen und ausführen. Denken Sie daran, die Projektmappe mit der richtigen Konfiguration für Ihre Zielplattform (x86, x64) zu erstellen, da es sonst zu Laufzeitfehlern kommen kann.

Bei der Ausführung des Beispiels wird das MDI-Haupt-Frame-Fenster angezeigt. Öffnen Sie mit dem Befehl **Datei | Öffnen** eine Mapping-Datei (z.B. `MarketingExpenses.mfd` aus dem MapForce-Beispielordner). Die Datei wird in einem neuen, untergeordneten Dokumentfenster geladen und angezeigt:



Nachdem Sie das Dokument geladen haben, können Sie über das Menü Befehle am aktiven Dokument ausführen. Es stehen auch Kontextmenüs zur Verfügung. Sie können zusätzliche Dokumente laden. Speichern Sie Ihre Änderungen mit dem Befehl **Datei | Speichern**.

18.5.1.2 Abrufen von Befehlsinformationen

Über das MapForceControl haben Sie über seine Eigenschaften `CommandsList` , `MainMenu` und `Toolbars` Zugriff auf alle Befehle von MapForce. Im Beispielprojekt im Ordner **<ApplicationFolder>\Examples\ActiveX\C#** wird die MapForce-Menüstruktur anhand der Eigenschaft `MainMenu` dynamisch erstellt.

© 2016-2022 Altova GmbH

Sie finden den Code zum Abrufen der Menübefehle in der Methode `MDIMain` in der Datei `MDIMain.cs` :

```
public MDIMain()
{
    // ...

    // Get the MainMenu property of the control and create the menu structure from it.
    MFLib.MapForceCommand objCommand = this.axMapForceControl.MainMenu;
    InsertMenuStructure(mainMenu, objCommand);
}
```

`mainMenu` im obigen Codefragment ist das vorhandene statische Menü des MDI-Frame-Hauptfensters. Wenn Sie das Formular **MDIMain.cs** in Visual Studio Designer öffnen, sehen Sie, dass dieses Menü zwei Menübefehle enthält: **File** und **Window**.



MDIMain.cs

Die Methode `InsertMenuStructure` erhält als Parameter die Objekte `mainMenu` und `objCommand` (ersteres ist das bestehende statische Menü, während zweiteres die gesamte aus dem MapForce ActiveX Control abgerufene Menüstruktur enthält). Anschließend wird die abgerufene MapForce-Menüstruktur mit dem vorhandenen statischen Menü zusammengeführt. Beachten Sie, dass die Menüs **File**, **Project** und **Window** absichtlich nicht dynamisch hinzugefügt werden, da diese Menüs mit aktiv offenen Dokumenten zu tun haben und dafür Code benötigt würde, der den Rahmen dieses Beispiels sprengen würde. Die grundlegenden Dateiverwaltungsbefehle (create, open, save, bring into focus) werden von den vorhandenen statischen Menüs **File** und **Window** übernommen. Alle anderen Menübefehle werden auf Basis der Informationen aus der Eigenschaft `MainMenu` des ActiveX Control dynamisch eingefügt. Die neuen Menüs werden hinter "File", jedoch vor "Window" eingefügt, d.h. beginnend mit dem Menüindex 1.

Die Methode `InsertMenuStructure` iteriert durch alle Menüs auf oberster Ebene im Objekt `MapForceCommand` und fügt für jedes einen neuen Menübefehl hinzu. Da jedes Menü auf oberster Ebene eigene untergeordnete Menüs hat, wird für jeden gefundenen Untermenübefehl die Methode `InsertMenuCommand` aufgerufen. Da jeder untergeordnete Menübefehl wiederum eigene untergeordnete Menübefehle haben kann, usw. durchläuft die Methode `InsertMenuCommand` sich selbst rekursiv, bis keine weiteren untergeordneten Menübefehle vorhanden sind.

Die dynamisch hinzugefügten Befehle sind Instanzen der Klasse `CustomMenuItem`, welche in `CustomMenuItem.cs` definiert ist. Diese Klasse ist von der Klasse `System.Windows.Forms.MenuItem` abgeleitet und hat ein zusätzliches Mitglied zum Speichern der MapForce-Befehls-ID.

```
public class CustomMenuItem : System.Windows.Forms.MenuItem
{
```

```
        public int m_MapForceCmdID;
}
```

Alle dynamisch hinzugefügten Befehle (mit Ausnahme derer, die Container für andere Befehle sind) rufen denselben Event Handler `AltovaMenuItem_Click` ab, der den folgenden Befehl verarbeitet:

```
private void AltovaMenuItem_Click(object sender, EventArgs e)
{
  if(sender.GetType() == System.Type.GetType("MapForceApplication.CustomMenuItem"))
  {
    CustomMenuItem   customItem = (CustomMenuItem)sender;
    ProcessCommand(customItem.m_MapForceCmdID);
  }
}
```

Wenn es sich beim Befehl um einen Container für andere Befehle handelt (d.h. wenn er untergeordnete Befehle enthält), ruft er den Event Handler `AltovaSubMenu_Popup` auf. Dieser Handler fragt den Status der einzelnen untergeordneten Menübefehle ab und aktiviert bzw. deaktiviert den Befehl je nach Bedarf. Damit wird sichergestellt, dass jeder Befehl nur dann aktiviert ist, wenn dies sinnvoll ist (So sollte der Menübefehl **File | Save** etwa deaktiviert sein, wenn kein aktives Dokument geöffnet ist).

Die Methode `ProcessCommand` delegiert die Ausführung entweder an das MapForceControl selbst oder an ein beliebiges aktives, in einem `MapForceControlDocument`-Control geladenes MapForce-Dokument. Der Grund hierfür ist, dass das MapForceControl keine Möglichkeit hat, zu eruieren, welches Dokument in der Host-Applikation gerade aktiv ist.

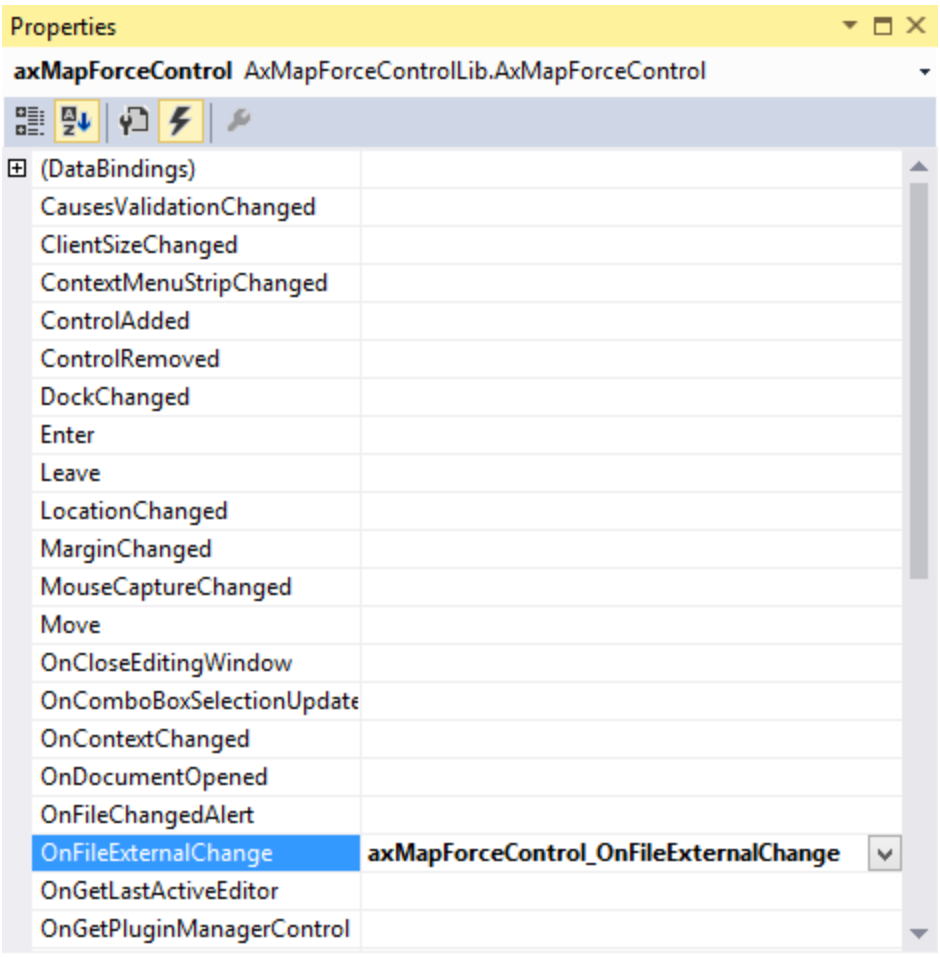
```
private void ProcessCommand(int nID)
{
   MapForceDoc docMapForce = GetCurrentMapForceDoc();

   if(docMapForce != null)
      docMapForce.axMapForceControlDoc.Exec(nID);
   else
      axMapForceControl.Exec(nID);
}
```

18.5.1.3 Behandlung von Events

Da alle Events in der MapForce Bibliothek auf Connection Points basieren, können Sie mit Hilfe der C#-Delegatsfunktionalität benutzerdefinierte Event Handler bereitstellen. Auf der Seite "Eigenschaften" der einzelnen Controls der MapForce Bibliothek finden Sie jeweils eine vollständige Liste von Events. In der Abbildung unten sehen Sie die Events des Haupt-MapForceControl:

© 2016-2022 Altova GmbH



Wie Sie sehen, wird in dem Beispiel nur das `OnFileExternalChange` Event überschrieben. Die Erstellung des C#-Delegaten wird vom C# Framework erledigt. Sie müssen nur den leeren Event Handler ausfüllen.

Im unten stehenden Beispiel deaktiviert die Handler-Implementierung das Neuladen von Dateien und zeigt ein Meldungsfeld an, um den Benutzer zu informieren, dass eine vom MapForceControl geladene Datei von außerhalb geändert wurde:

```
private void axMapForceControl_OnFileExternalChange(object sender,
AxMapForceControlLib._DMapForceControlEvents_OnFileExternalChangeEvent e)
{
    MessageBox.Show("Attention: The file " + e.strPath + " has been changed from
outside\nbut reloading is turned off in the sample application!");

    // This turns off any file reloading:
    e.varRet = false;
}
```

18.5.2 Java

MapForce ActiveX-Komponenten können von Java-Code aus aufgerufen werden. Die Java-Integration wird von den unten aufgelisteten Bibliotheken zur Verfügung gestellt. Diese Bibliotheken stehen im Ordner `<ApplicationFolder>\Examples\JavaAPI` Ihrer MapForce Installation zur Verfügung, nachdem Sie MapForce und das MapForce Integrationspaket installiert haben (siehe auch Voraussetzungen 1920).

- `AltovaAutomation.dll`: ein JNI Wrapper für Altova Automation Server (bei 32-Bit-Installationen von MapForce)
- `AltovaAutomation_x64.dll`: ein JNI-Wrapper für Altova Automation Server (bei 64-Bit-Installationen von MapForce)
- `AltovaAutomation.jar`: Java-Klassen zum Aufrufen von Altova Automation Servern
- `MapForceActiveX.jar`: Java-Klassen, die als Wrapper für die MapForce ActiveX-Schnittstelle dienen
- `MapForceActiveX_JavaDoc.zip`: eine Javadoc Datei, die die Hilfedokumentation zur Java API enthält

Anmerkung: Um die Java ActiveX Integration verwenden zu können, müssen sich die .dll- und .jar-Dateien im Java Class-Suchpfad befinden.

Java-Beispielprojekt

Im Lieferumfang Ihres Produkts ist ein Java-Beispielprojekt enthalten. Sie können das Java-Projekt nach Belieben testen und verwenden. Nähere Informationen finden Sie im Abschnitt Java-Beispielprojekt 1939.

Regeln für das Mappen der ActiveX Control-Namen auf Java

Eine Dokumentation zu den ActiveX Controls finden Sie in der Objektreferenz 1959. Beachten Sie, dass sich die Objektbenennungskonventionen in Java etwas von anderen Sprachen unterscheiden. Für das Mapping zwischenden ActiveX Controls und dem Java Wrapper gelten die folgenden Regeln:

- ***Klassen und Klassennamen***
  Für jede Komponente des MapForce ActiveX Interface gibt es eine Java-Klasse mit dem Namen der Komponente.

- ***Methodennamen***
  Die Methodennamen im Java Interface sind dieselben wie die in den COM Interfaces, beginnen aber aufgrund der Java-Namenskonventionen mit einem Kleinbuchstaben. Zum Aufrufen von COM-Eigenschaften können dem Eigenschaftsnamen vorangestellte Java-Methoden mit `get` und `set` verwendet werden. Wenn eine Eigenschaft keinen Schreibzugriff ermöglicht, steht keine Setter-Methode zur Verfügung. Beispiel: Für die Eigenschaft `IntegrationLevel` des MapForceControl stehen die Java-Methoden `getIntegrationLevel` und `setIntegrationLevel` zur Verfügung.

- ***Enumerationen***
  Für jede im ActiveX Interface definierte Enumeration ist eine Java-Enumeration desselben Namens und mit denselben Werten definiert.

- ***Events und Event Handler***
  Für jedes Interface im Automation Interface, das Events unterstützt, steht ein Java-Interface desselben Namens plus '`Event`' zur Verfügung. Um das Überladen von Einzel-Events zu vereinfachen, gibt es eine Java-Klasse mit Standardimplementierungen für alle Events. Der Name dieser Java-Klasse ist der Name des Event Interface plus '`DefaultHandler`'. Beispiel:

© 2016-2022 Altova GmbH

`MapForceControl`: Java-Klasse zum Aufrufen der Applikation
`MapForceControlEvents`: Events Interface für das MapForceControl Control
`MapForceControlEventsDefaultHandler`: Standard-Handler für `MapForceControlEvents`

Ausnahmen für Mapping-Regeln

Zu den oben aufgelisteten Regeln gibt es die folgenden Ausnahmen:

Interface	Änderungen in der Java-Klasseclass
`MapForceControlDocument`, Methode `New`	Umbenannt in `newDocument`
`MapForceControlDocument`, Methode `OpenDocument`	Entfernt. Verwenden Sie die Methode `Open`
`MapForceControlDocument`, Methode `NewDocument`	Entfernt. Verwenden Sie die Methode `newDocument`
`MapForceControlDocument`, Methode `SaveDocument`	Entfernt. Verwenden Sie die Methode `Save`

Dieser Abschnitt

In diesem Abschnitt wird erklärt, wie einige grundlegende MapForce ActiveX-Funktionen über Java-Code aufgerufen werden können. Der Abschnitt ist in die folgenden Unterabschnitte gegliedert:

- Java-Beispielprojekt 1939
- Erstellen der ActiveX Controls 1941
- Laden der Daten in die Controls 1942
- Behandlung von Events 1942
- Menüs 1943
- Behandlung von UI Update Events 1945
- Erstellen einer MapForce Mapping-Tabelle 1945

18.5.2.1 Java-Beispielprojekt

Das MapForce-Installationspaket enthält ein Java-Beispielprojekt, das Sie Ordner "ActiveX Examples" des Applikationsordners `<ApplicationFolder>\Examples\ActiveX\Java\` finden.

Im Java-Beispiel wird gezeigt, wie Sie das MapForceControl in eine mit Java erstellte Desktop Applikation integrieren können. Sie können das Beispielprojekt mit Hilfe der Batch-Datei `BuildAndRun.bat`, direkt über die Befehlszeile testen oder Sie können es in Eclipse kompilieren und ausführen. Anleitungen dafür finden Sie weiter unten.

Dateiliste

Der Ordner für die Java-Beispiele enthält alle zum Ausführen des Beispielprojekts erforderlichen Dateien. Diese Dateien sind unten aufgelistet:

`.classpath`	Hilfedatei Eclipse-Projekt
`.project`	Eclipse-Projektdatei
`AltovaAutomation.dll`	Java-COM Bridge: DLL-Teil (für die 32-Bit-Installation)
`AltovaAutomation_x64.dll`	Java-COM Bridge: DLL-Teil (für die 64-Bit-Installation)
`AltovaAutomation.jar`	Java-COM Bridge: Java-Bibliotheksteil
`BuildAndRun.bat`	Batch-Datei zum Kompilieren und Ausführen des Beispielcodes über die Befehlszeile. Es wird ein Ordner benötigt, in dem sich die Java Virtual Machine als Parameter befindet.
`MapForceActiveX.jar`	Java-Klassen des MapForce ActiveX Control
`MapForceActiveX_JavaDoc.zip`	Javadoc Datei, die die Hilfedokumentation für die Java API enthält
`MapForceContainer.java`	Java-Beispielquellcode
`MapForceContainerEventHandler.java`	Java-Beispielquellcode
`MapForceTable.java`	Java-Beispielquellcode

Funktionen in diesem Beispiel

Im Beispiel werden ein MapForce Dokument-Editor-Fenster, das MapForce-Projektfenster, das MapForce-Bibliotheksfenster und das MapForce-Validierungsfenster in ein AWT-Rahmenfenster platziert. Das für MapForce definierte Hauptmenü wird ausgelesen und ein AWT-Menü mit derselben Struktur wird erstellt. Über dieses Menü oder über das Projektfenster können Sie Dateien im Dokument-Editor öffnen und damit arbeiten.

Sie können das Beispiel nach Ihren Wünschen modifizieren.

In den Codefragmenten werden die folgenden spezifischen Schritte beschrieben:

- Erstellen der ActiveX Controls 1941: Startet MapForce, das als Automation Server registriert ist, bzw. aktiviert das Programm, wenn MapForce bereits ausgeführt wird.
- Laden der Daten in die Controls 1942: Navigiert zu einem der mit MapForce installierten Beispieldokumente und öffnet es.
- Grundlegendes zur Event-Behandlung 1942: Wechselt von der Ansicht aller offenen Dokumente in die Textansicht. Im Code wird auch gezeigt, wie man durch offene Dokumente iteriert.
- Menüs 1943: Validiert das aktive Dokument und zeigt das Ergebnis in einem Meldungsfeld an. Im Code wird gezeigt, wie Ausgabeparameter verwendet werden.
- Behandlung von UI Update Events 1945: Zeigt, wie MapForce Events behandelt werden.
-
- Erstellen einer MapForce Mapping-Tabelle 1945: Zeigt wie, Sie eine MapForce Mapping-Tabelle erstellen und sie für die modale Aktivierung vorbereiten.

Aktualisieren des Pfads zum Ordner "Examples"

Bevor Sie das zur Verfügung gestellte Beispiel ausführen, müssen Sie die **MapForceContainer.java**-Datei möglicherweise bearbeiten und überprüfen, ob der folgende Pfad auf den tatsächlichen Ordner verweist, in dem die MapForce Beispieldateien auf Ihrem Betriebssystem gespeichert sind.

© 2016-2022 Altova GmbH

```
// Locate samples installed with the product.
final String strExamplesFolder = System.getenv( "USERPROFILE" ) + "\\Documents\\Altova\
\MapForce2023\\MapForceExamples\\";
```

Ausführen des Beispiels über die Befehlszeile

So führen Sie das Beispiel über die Befehlszeile aus:

1. Überprüfen Sie, ob alle Voraussetzungen erfüllt werden (siehe Voraussetzungen 1920).
2. Öffnen Sie ein Eingabeaufforderungsfester, wechseln Sie in den Java-Beispielprojektordner und geben Sie folgende Zeile ein:

   ```
   buildAndRun.bat "<Pfad-zum-Java-bin-Ordner>"
   ```

1. Drücken Sie die **Eingabetaste**.

Der Java-Quellcode in `MapForceContainer.java` wird kompiliert und anschließend ausgeführt.

Kompilieren und Ausführen des Beispiels in Eclipse

So importieren Sie das Java-Beispielprojekt in Eclipse:

3. Überprüfen Sie, ob alle Voraussetzungen erfüllt werden (siehe Voraussetzungen 1920).
1. Klicken Sie im Menü **File** auf **Import**.
2. Wählen Sie **Existing Projects into Workspace** und navigieren Sie zur Eclipse-Projektdatei unter `<ApplicationFolder>\Examples\ActiveX\Java\`. Da Sie möglicherweise keinen Schreibzugriff auf diesen Ordner haben, wird empfohlen, im Importdialogfeld das Kontrollkästchen **Copy projects into workspace** zu aktivieren.

Um die Beispielapplikation zu starten, klicken Sie mit der rechten Maustaste im Package Explorer auf das Paket und wählen Sie den Befehl **Run as | Java Application**.

Hilfe zu Java API-Klassen steht in Form von Kommentaren im Code sowie als Javadoc-Ansicht von Eclipse zur Verfügung. Um die Javadoc-Ansicht in Eclipse zu aktivieren, wählen Sie den Menübefehl **Window | Show View | JavaDoc**.

18.5.2.2 Erstellen der ActiveX Controls

Im unten gezeigten Code sehen Sie wie ActiveX Controls erstellt werden. Die Konstruktoren erstellen die Java Wrapper-Objekte. Durch Hinzufügen dieser aus Canvas stammenden Objekte zu einem Bereich oder Rahmen wird die Erstellung des in Wrapper verpackten ActiveX-Objekts ausgelöst.

```
01   /**
02    * MapForce manager control - always needed
03    */
04   public static MapForceControl         mapForceControl = null;
05
```

```
06   /**
07    * MapForceDocument editing control
08    */
09   public static MapForceControlDocument     mapForceDocument = null;
10
11   /**
12    * Tool windows - MapForce place-holder controls
13    */
14   private static MapForceControlPlaceHolder   mapForceProjectToolWindow = null;
15   private static MapForceControlPlaceHolder   mapForceValidationToolWindow = null;
16   private static MapForceControlPlaceHolder   mapForceLibraryToolWindow = null;
17
18   // Create the MapForce ActiveX control; the parameter determines that we want
     // to place document controls and place-holder controls individually.
19   // It gives us full control over the menu, as well.
20      mapForceControl = new MapForceControl(
        ICActiveXIntegrationLevel.ICActiveXIntegrationOnDocumentLevel.getValue(), false );
21
22      mapForceDocument = new MapForceControlDocument();
23      frame.add( mapForceDocument, BorderLayout.CENTER );
24
25
26   // Create a project window and open the sample project in it
27      mapForceProjectToolWindow = new MapForceControlPlaceHolder(
        MapForceControlPlaceholderWindow.MapForceXProjectWindow.getValue(),
        strExamplesFolder + "MapForceExamples.mfp" ) ;
28      mapForceProjectToolWindow.setPreferredSize( new Dimension( 200, 200 ) );
).
```

18.5.2.3 Laden der Daten in die Controls

Im unten gezeigten Code sehen Sie wie Daten in ActiveX Controls geladen werden können.

```
1    // Locate samples installed with the product.
2       final String strExamplesFolder = System.getenv( "USERPROFILE" ) +
        "\\Documents\\Altova\\MapForce2022\\MapForceExamples\\";
3       mapForceProjectToolWindow = new
MapForceControlPlaceHolder( MapForceControlPlaceholderWindow.MapForceXProjectWindow.getValu
e(), strExamplesFolder + "MapForceExamples.mfp" ) ;
```

18.5.2.4 Grundlegendes zur Event-Behandlung

Im unten gezeigten Code sehen Sie wie grundlegende Events behandelt werden. Bei Aufruf der MapForceControl-Methode `open` oder beim Öffnen einer Datei über das Menü oder die Projektstruktur wird das `onOpenedOrFocused` Event an den dazugehörigen Event Handler gesendet. Im Prinzip wird dieses Event behandelt, indem die Datei durch Aufruf der `open`-Methode des MapForceDocumentControl geöffnet wird.

```
01       // Open the Marketing file when button is pressed
02       btnMarkExp.addActionListener( new ActionListener() {
03         public void actionPerformed(ActionEvent e) {
04           try {
```


© 2016-2022 Altova GmbH

```
05             // Instruct the Document control to open the file - avoid calling the open
method of MapForceControl (see help)
06             mapForceDocument.open( strExamplesFolder + "MarketingExpenses.mfd" );
07             mapForceDocument.requestFocusInWindow();
08           } catch (AutomationException e1) {
09             e1.printStackTrace();
10           }
11         }
12       } );
13       public void onOpenedOrFocused( String i_strFileName, boolean
i_bOpenWithThisControl, boolean i_bFileAlreadyOpened ) throws AutomationException
14   {
15     // Handle the New/Open events coming from the Project tree or from the menus
16     if ( !i_bFileAlreadyOpened )
17     {
18       // This is basically an SDI interface, so open the file in the already existing
document control
19       try {
20         MapForceContainer.mapForceDocument.open( i_strFileName );
21         MapForceContainer.mapForceDocument.requestFocusInWindow();
22       } catch (Exception e) {
23         e.printStackTrace();
24       }
25     }
26   }
```

18.5.2.5 Menüs

Im unten gezeigten Code sehen Sie, wie Menüeinträge erstellt werden können. Jedes `MapForceCommand` Objekt holt ein entsprechendes `MenuItem` Objekt, wobei der `ActionCommand` auf die ID des Befehls gesetzt wird. Die von allen Menüeinträgen generierten Aktionen werden von derselben Funktion gehandelt. Diese kann bestimmte Handlings (wie z.B Neuinterpretieren der Schließfunktion) durchführen oder die Ausführung durch Aufruf seiner exec-Methode an das `MapForceControl` Objekt delegieren. Das `menuMap` Objekt, das bei der Menüerstellung befüllt wird, wird später verwendet (siehe Abschnitt Behandlung von UI Update Events 1945).

```
01
02       // Load the file menu when the button is pressed
03       btnMenu.addActionListener( new ActionListener() {
04         public void actionPerformed(ActionEvent e) {
05           try {
06             // Create the menubar that will be attached to the frame
07             MenuBar mb = new MenuBar();
08             // Load the main menu's first item - the File menu
09             MapForceCommand xmlSpyMenu =
mapForceControl.getMainMenu().getSubCommands().getItem( 0 );
10             // Create Java menu items from the Commands objects
11             Menu fileMenu = new Menu();
12             handlerObject.fillMenu( fileMenu, xmlSpyMenu.getSubCommands() );
13             fileMenu.setLabel( xmlSpyMenu.getLabel().replace( "&", "" ) );
14             mb.add( fileMenu );
15             frame.setMenuBar( mb );
16             frame.validate();
17           } catch (AutomationException e1) {
18             e1.printStackTrace();
```

```
19            }
20            // Disable the button when the action has been performed
21            ((AbstractButton) e.getSource()).setEnabled( false );
22          }
23        } ) ;
24    /**
25     * Populates a menu with the commands and submenus contained in an MapForceCommands
object
26     */
27    public void fillMenu(Menu newMenu, MapForceCommands mapForceMenu) throws
AutomationException
28    {
29      // For each command/submenu in the mapForceMenu
30      for ( int i = 0 ; i < mapForceMenu.getCount() ; ++i  )
31      {
32        MapForceCommand mapForceCommand = mapForceMenu.getItem( i );
33        if ( mapForceCommand.getIsSeparator() )
34          newMenu.addSeparator();
35        else
36        {
37          MapForceCommands subCommands = mapForceCommand.getSubCommands();
38          // Is it a command (leaf), or a submenu?
39          if ( subCommands.isNull() || subCommands.getCount() == 0 )
40          {
41            // Command -> add it to the menu, set its ActionCommand to its ID and store in
in the menuMap
42            MenuItem mi = new MenuItem( mapForceCommand.getLabel().replace( "&", "" ) );
43            mi.setActionCommand( "" + mapForceCommand.getID() );
44            mi.addActionListener( this );
45            newMenu.add( mi );
46            menuMap.put( mapForceCommand.getID(), mi );
47          }
48          else
49          {
50            // Submenu -> create submenu and repeat recursively
51            Menu newSubMenu = new Menu();
52            fillMenu( newSubMenu, subCommands );
53            newSubMenu.setLabel( mapForceCommand.getLabel().replace( "&", "" ) );
54            newMenu.add( newSubMenu );
55          }
56        }
57      }
58    }
59    /**
60     * Action handler for the menu items
61     * Called when the user selects a menu item; the item's action command corresponds to
the command table for MapForce
62     */
63    public void actionPerformed( ActionEvent e )
64    {
65      try
66      {
67        int iCmd = Integer.parseInt( e.getActionCommand() );
68        // Handle explicitly the Close commands
69        switch ( iCmd )
70        {
71          case 57602:       // Close
72          case 34050:       // Close All
73            MapForceContainer.initMapForceDocument();
```


© 2016-2022 Altova GmbH

```
74            break;
75          default:
76            MapForceContainer.mapForceControl.exec( iCmd );
77            break;
78        }
79      }
80      catch ( Exception ex )
81      {
82        ex.printStackTrace();
83      }
84
85    }
```

18.5.2.6 Behandlung von UI Update Events

Im unten gezeigten Code, sehen Sie wie ein UI-Update Event Handler erstellt werden kann.

```
01 /**
02     * Call-back from the MapForceControl.
03     * Called to enable/disable commands
04     */
05    @Override
06    public void onUpdateCmdUI() throws AutomationException
07    {
08      // A command should be enabled if the result of queryStatus contains the Supported
(1) and Enabled (2) flags
09      for ( java.util.Map.Entry<Integer, MenuItem> pair : menuMap.entrySet() )
10
       pair.getValue().setEnabled( MapForceContainer.mapForceControl.queryStatus( pair.getKe
y() ) > 2 );
11    }
12   /**
13     * Call-back from the MapForceControl.
14     * Usually called while enabling/disabling commands due to UI updates
15     */
16    @Override
17    public boolean onIsActiveEditor( String i_strFilePath ) throws AutomationException
18    {
19      try {
20        return
MapForceContainer.mapForceDocument.getDocument().getFullName().equalsIgnoreCase( i_strFileP
ath );
21      } catch ( Exception e ) {
22        return false;
23      }
24    }
```

18.5.2.7 Auflisten der Eigenschaften eines MapForce Mappings

Im unten gezeigten Code sehen Sie, wie ein Mapping-Objekt in MapForce als Tabelle geladen und für die modale Aktivierung vorbereitet werden kann.

```
01 //access MapForce Java-COM bridge
02 import com.altova.automation.MapForce.*;
03 import com.altova.automation.MapForce.Component;
04 import com.altova.automation.MapForce.Enums.ENUMComponentUsageKind;
05
06 //access AWT and Swing components
07 import java.awt.*;
08 import javax.swing.*;
09 import javax.swing.table.*;
10
11
12 /**
13  * A simple example of a table control loading the structure from a Mapping object.
14  * The class receives an Mapping object, loads its components in a JTable, and prepares
15  * for modal activation.
16  *
17  * Feel free to modify and extend this sample.
18  *
19  * @author Altova GmbH
20  */
21 class MapForceTable extends JDialog
22 {
23   /**
24    * The table control
25    */
26   private JTable myTable;
27
28   /**
29    * Constructor that prepares the modal dialog containing the filled table control
30    * @param mapping The data to be displayed in the table
31    * @param parent  Parent frame
32    */
33   public MapForceTable( Mapping mapping, Frame parent )
34   {
35     // Construct the modal dialog
36     super( parent, "MapForce component table", true );
37     // Build up the tree
38     fillTable( mapping );
39     // Arrange controls in the dialog
40     setContentPane( new JScrollPane( myTable ) );
41   }
42
43   /**
44    * Loads the components of a Mapping object in the table
45    * @param mapping Source data
46    */
47   private void fillTable( Mapping mapping)
48   {
49     try
50     {
51       // count how many Instance components do we have
52       int size = 0;
53       for (Component comp : mapping.getComponents())
54         if ( comp.getUsageKind() ==
ENUMComponentUsageKind.eComponentUsageKind_Instance )
55           ++size;
56
```


© 2016-2022 Altova GmbH

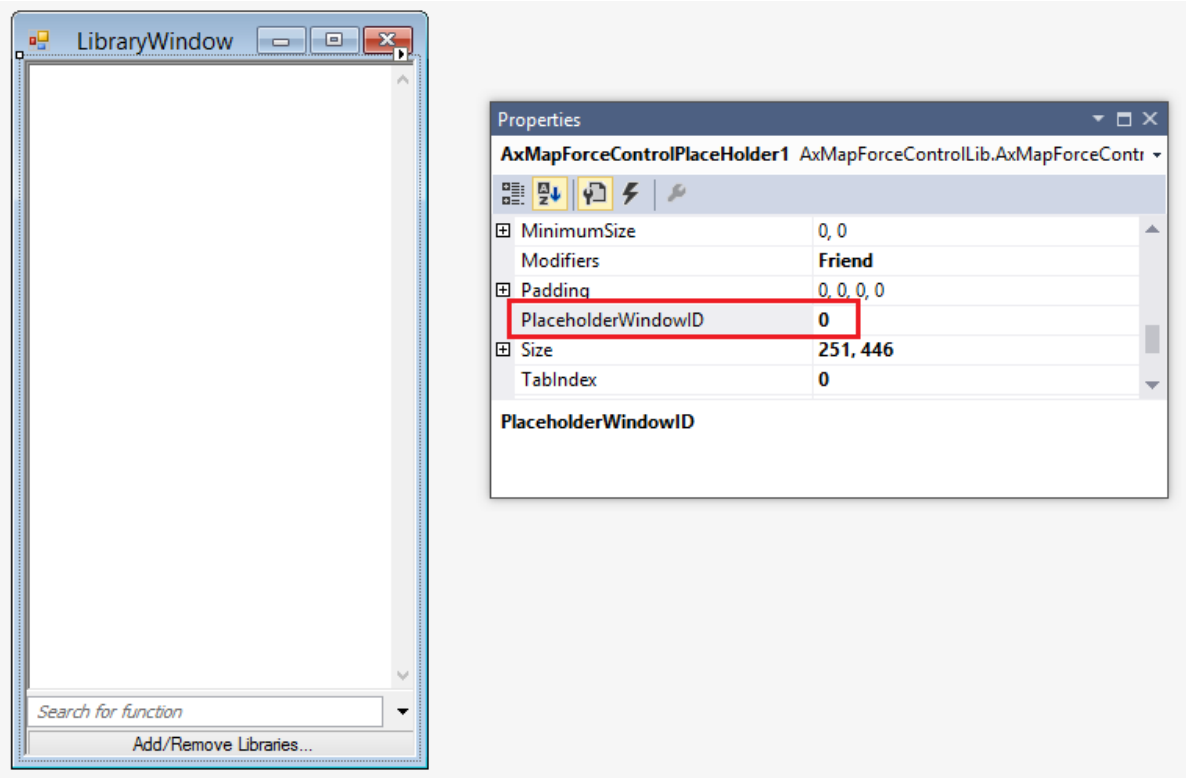
```
57        // Prepare data
58        final String[] columnNames = { "Component", "Has inputs", "Has outputs", "Input
file", "Output file", "Schema" };
59        final Object[][] data = new Object[size ][ 7 ] ;
60        int index = 0 ;
61        for (Component comp : mapping.getComponents())
62          if ( comp.getUsageKind() ==
ENUMComponentUsageKind.eComponentUsageKind_Instance )
63          {
64            int i = 0;
65            data[ index ][ i++ ] = comp.getName() ;
66            data[ index ][ i++ ] = new Boolean( comp.getHasIncomingConnections() );
67            data[ index ][ i++ ] = new Boolean( comp.getHasOutgoingConnections() );
68            data[ index ][ i++ ] = comp.getInputInstanceFile();
69            data[ index ][ i++ ] = comp.getOutputInstanceFile();
70            data[ index++ ][ i ] = comp.getSchema() ;
71          }
72
73        // Set up table
74        myTable = new JTable( new AbstractTableModel() {
75            public String getColumnName(int col) { return columnNames[col]; }
76            public int getRowCount() { return data.length; }
77            public int getColumnCount() { return columnNames.length; }
78            public Object getValueAt(int row, int col) { return data[row][col]; }
79            public boolean isCellEditable(int row, int col) { return false; }
80            public Class getColumnClass(int c) { return getValueAt(0, c).getClass(); }

81        } );
82
83        // Set width
84        for( index = 0 ; index < columnNames.length ; ++index )
85          myTable.getColumnModel().getColumn( index ).setMinWidth( 80 );
86        myTable.getColumnModel().getColumn( 5 ).setMinWidth( 400 );
87      }
88      catch (Exception e)
89      {
90        e.printStackTrace();
91      }
92    }
93
94 }
```

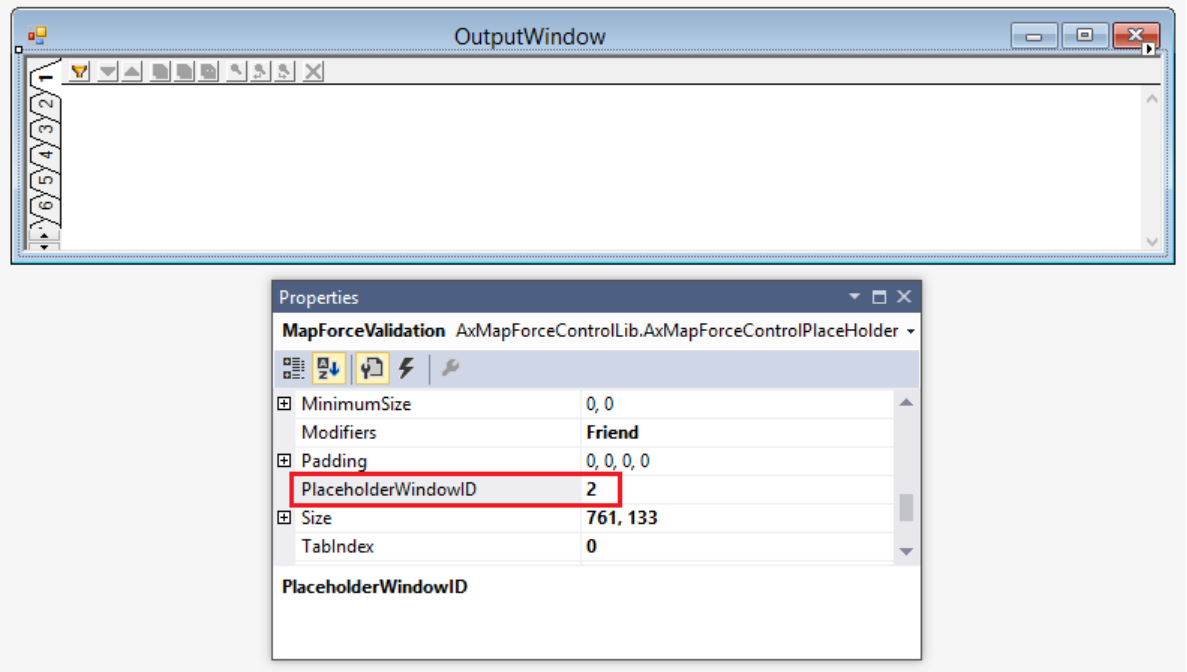
18.5.3 VB.NET

Der Quellcode, der das Beispiel für eine Integration von MapForceControl in eine VB.NET-Applikation bildet, befindet sich im Ordner `<ApplicationFolder>\Examples\ActiveX\VB.NET` Ihrer MapForce Installation. Die Lösung besteht aus den folgenden drei Fenstern:

1. **MainWindow.vb** - das Dokument-Hauptfenster, das auch ein einfaches Applikationsmenü enthält.
2. **LibraryWindow.vb** - das Bibliotheksfenster. Dieses Fenster wird durch ein Placeholder Control, dessen **PlaceholderWindowID**-Eigenschaft auf 0 gesetzt ist, befüllt (aufgrund dieses Werts wird vom Control speziell das Fenster "Bibliotheken" angezeigt).



3. **OutputWindow.vb** - Das Fenster "Meldungen" (Ausgabefenster). Dieses Fenster wird durch ein Placeholder Control dessen **PlaceholderWindowID-**Eigenschaft auf 2 gesetzt ist, befüllt (aufgrund dieses Werts wird im Control das Ausgabefenster angezeigt).



 © 2016-2022 Altova GmbH

Bevor Sie versuchen, diese Projektmappe zu erstellen und auszuführen, beachten Sie die folgenden Schritte:

Schritt 1: Überprüfen Sie die Voraussetzungen

Eine Liste der Voraussetzungen finden Sie unter Voraussetzungen[1920].

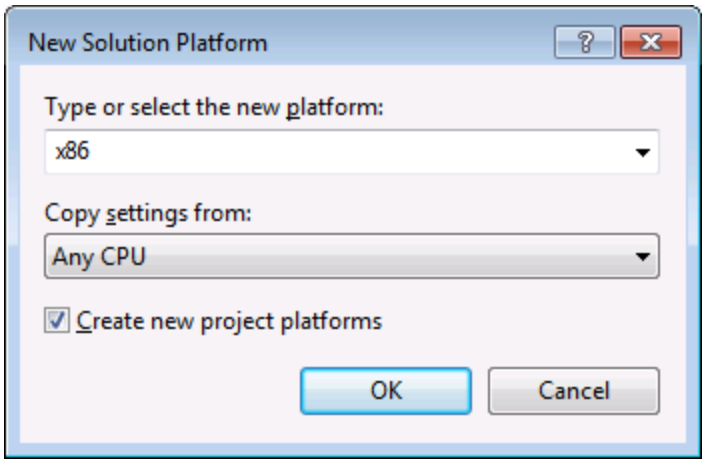
Schritt 2: Kopieren Sie das Beispiel in ein Verzeichnis, auf das Sie Schreibzugriff haben

Um Visual Studio nicht als Administrator ausführen zu müssen, kopieren Sie den Quellcode in ein Verzeichnis, auf das Sie Schreibzugriff haben, anstatt ihn vom Standardverzeichnis aus auszuführen.

Schritt 3: Definieren Sie die Build-Plattform

- Erstellen Sie eine Build-Plattform-Konfiguration für die Plattform, unter der Sie die Projektmappe erstellen möchten (x86, x64). So erstellen Sie die Build-Konfiguration:

  a. Klicken Sie mit der rechten Maustaste auf die Projektmappe in Visual Studio und wählen Sie **Configuration Manager**.
  b. Wählen Sie unter **Active solution platform** den Befehl **New...** und wählen Sie anschließend die x86- oder x64-Konfiguration (in diesem Beispiel **x86**).



Sie sind nun fertig und können die Projektmappe in Visual Studio erstellen und ausführen. Denken Sie daran, die Projektmappe mit der richtigen Konfiguration für Ihre Zielplattform (x86, x64) zu erstellen.

18.6 Befehlsreferenz

In diesem Abschnitt werden die Namen und Identifier aller Menübefehle aufgelistet, die in MapForce zur Verfügung stehen. In den einzelnen Unterabschnitten werden jeweils die Befehle aus dem entsprechenden Menü von MapForce aufgelistet. Die Befehlstabellen sind folgendermaßen gegliedert:

- In der Spalte "Menübefehl" sehen Sie den Menütext des Befehls, wie er in MapForce, angezeigt wird, damit Sie die dem Befehl zugrunde liegende Funktionalität leichter erkennen.
- In der Spalte "Befehlsname" ist der String angegeben, anhand dessen ein Symbol desselben Namens aus dem Ordner **ActiveX\Images** des MapForce Installationsverzeichnis aufgerufen werden kann.
- In den "ID"-Spalten sehen Sie die numerischen Identifier der Spalte, die als Argument an die Methoden geliefert wird, die diesen Befehl ausführen oder abfragen.

Um einen Befehl auszuführen, verwenden Sie die Methoden `MapForceControl.Exec` 1966 oder `MapForceControlDocument.Exec` 1973. Um den Status eines Befehls abzufragen, verwenden Sie die Methoden `MapForceControl.QueryStatus` 1967 oder `MapForceControlDocument.QueryStatus` 1974.

Einige dieser Befehle werden je nach installierter MapForce Edition eventuell nicht unterstützt.

18.6.1 Menü "Datei"

Das Menü "Datei" enthält die folgenden Befehle:

Menübefehl	Befehlsname	ID
Neu...	ID_FILE_NEW	57600
Öffnen...	ID_FILE_OPEN	57601
Speichern	ID_FILE_SAVE	57603
Speichern unter...	ID_FILE_SAVE_AS	57604
Alles speichern	ID_FILE_SAVEALL	32377
Neu laden	IDC_FILE_RELOAD	32467
Schließen	ID_WINDOW_CLOSE	32453
Alles schließen	ID_WINDOW_CLOSEALL	32454
Drucken...	ID_FILE_PRINT	57607
Druckvorschau	ID_FILE_PRINT_PREVIEW	57609
Druckereinrichtung...	ID_FILE_PRINT_SETUP	57606
Mapping validieren	ID_MAPPING_VALIDATE	32347
Mapping-Einstellungen	ID_MAPPING_SETTINGS	32396
Code in ausgewählter Sprache generieren	ID_FILE_GENERATE_SELECTED_CODE	32362

© 2016-2022 Altova GmbH

Menübefehl	Befehlsname	ID
XSLT 1.0	ID_FILE_GENERATEXSLT	32360
XSLT 2.0	ID_FILE_GENERATEXSLT2	32361
XQuery	ID_FILE_GENERATEXQUERY	32359
Java	ID_FILE_GENERATEJAVACODE	32358
C# (Sharp)	ID_FILE_GENERATECSCODE	32357
C++	ID_FILE_GENERATECPPCODE	32356
Zu MapForce Server-Ausführungsdatei kompilieren...	ID_FILE_CREATE_SERVER_EXECUTION_FILE	32517
Auf FlowForce Server bereitstellen...	ID_FILE_DEPLOY_MAPPING	32506
Dokumentation generieren...	ID_FILE_GENERATE_DOCUMENTATION	32468
Letzte Datei	ID_FILE_MRU_FILE1	57616
Beenden	ID_APP_EXIT	57665

18.6.2 Menü "Bearbeiten"

Das Menü "Bearbeiten" enthält die folgenden Befehle:

Menübefehl	Befehlsname	ID
Rückgängig	ID_EDIT_UNDO	57643
Wiederherstellen	ID_EDIT_REDO	57644
Suchen...	ID_EDIT_FIND	57636
Weitersuchen	ID_EDIT_FINDNEXT	32349
Vorheriges suchen	ID_EDIT_FINDPREV	32350
Ausschneiden	ID_EDIT_CUT	57635
Kopieren	ID_EDIT_COPY	57634
Einfügen	ID_EDIT_PASTE	57637
Löschen	ID_EDIT_CLEAR	57632
Alle auswählen	ID_EDIT_SELECT_ALL	57642

18.6.3 Menü "Einfügen"

Das Menü "Einfügen" enthält die folgenden Befehle:

Menübefehl	Befehlsname	ID
XML-Schema/Datei...	ID_INSERT_XSD	32393
Datenbank...	ID_INSERT_DATABASE	32389
EDI...	ID_INSERT_EDI	32390
Textdatei...	ID_INSERT_TXT	32392
Webservice-Funktion...	ID_INSERT_WEBSERVICE_FUNCTION	32319
Excel 2007+-Datei...	ID_INSERT_EXCEL	32376
XBRL-Dokument...	ID_INSERT_XBRL	32469
JSON-Schema/Datei...	ID_INSERT_JSON	32531
Input-Komponente einfügen...	ID_FUNCTION_INSERT_INPUT	32383
Output-Komponente einfügen...	ID_FUNCTION_INSERT_OUTPUT	32402
Konstante...	ID_INSERT_CONSTANT	32388
Variable...	ID_INSERT_VARIABLE	32500
Join	ID_INSERT_JOIN	32581
Sortieren: Nodes/Zeilen	ID_INSERT_SORT	32444
Filter: Nodes/Zeilen	ID_INSERT_FILTER	32391
SQL-WHERE/ORDER	ID_INSERT_SQLWHERE_CONDITION	32351
Wertezuordnung	ID_INSERT_VALUEMAP	32354
IF-Else-Bedingung	ID_INSRT_CONDITION	32394
Ausnahme	ID_INSERT_EXCEPTION	32311

18.6.4 Menü "Projekt"

Das Menü "Projekt" enthält die folgenden Befehle:

Menübefehl	Befehlsname	ID
Projekt neu laden	ID_PROJECT_RELOAD	32476

© 2016-2022 Altova GmbH

Menübefehl	Befehlsname	ID
Projekt schließen	ID_FILE_CLOSEPROJECT	32355
Projekt speichern	ID_FILE_SAVEPROJECT	32378
Dateien zu Projekt hinzufügen...	ID_PROJECT_ADDFILESTOPROJECT	32420
Aktive Datei zu Projekt hinzufügen	ID_PROJECT_ADDACTIVEFILETOPROJECT	32419
Ordner erstellen ...	ID_PROJECT_CREATE_FOLDER	32310
Mapping öffnen	ID_PROJECT_OPEN_MAPPING	32307
Mapping für Operation erstellen...	ID_PROJECT_CREATE_MAPPING_FOR_OPERATION	32399
Mapping-Datei für Operation hinzufügen...	ID_PROJECT_ADD_MAPPING	32309
WebService einfügen...	ID_PROJECT_INSERT_WEBSERVICE	32306
Datei in XMLSpy öffnen	ID_PROJECT_OPEN_IN_XMLSPY	32305
Code für das gesamte Projekt generieren	ID_PROJECT_GENERATE_ALL	32303
XSLT 1.0	ID_PROJECT_GENERATEXSLTCODE_ENTIRE	32408
XSLT 2.0	ID_PROJECT_GENERATEXSLT2CODE_ENTIRE	32409
XQuery	ID_PROJECT_GENERATEXQUERYCODE_ENTIRE	32410
Java	ID_PROJECT_GENERATEJAVACODE_ENTIRE	32411
C# (Sharp)	ID_PROJECT_GENERATECSCODE_ENTIRE	32412
C++	ID_PROJECT_GENERATECPPCODE_ENTIRE	32413
Eigenschaften	ID_PROJECT_PROPERTIES	32404
Letztes Projekt	ID_FILE_MRU_PROJECT1	32364

18.6.5 Menü "Komponente"

Das Menü "Komponente" enthält die folgenden Befehle:

Menübefehl	Befehlsname	ID
Root-Element ändern...	ID_COMPONENT_CHANGEROOTELEMENT	32334
Schema-Definition in XMLSpy bearbeiten	ID_COMPONENT_EDIT_SCHEMA	32337

Menübefehl	Befehlsname	ID
FlexText-Konfiguration bearbeiten	ID_COMPONENT_EDIT_MFT	32301
Datenbankobjekte hinzufügen/entfernen/bearbeiten...	ID_COMPONENT_SELECTTABLES	32346
Mapping auf EDI X12 997 erstellen	ID_COMPONENT_CREATE_MAPPING_TO_997	32483
Mapping auf EDI X12 999 erstellen	ID_COMPONENT_CREATE_MAPPING_TO_999	32484
Aktualisieren	IDC_COMMAND_REFRESH_COMPONENT	32373
Duplikat davor einfügen	ID_COMPONENT_CREATE_DUPLICATE_ICON_BEFORE	32503
Duplikat danach einfügen	ID_COMPONENT_CREATE_DUPLICATE_ICON	32335
Duplikat löschen	ID_COMPONENT_REMOVE_DUPLICATE_ICON	32339
Kommentar davor hinzufügen	ID_COMPONENT_ADD_COMMENT_BEFORE	32518
Kommentar danach hinzufügen	ID_COMPONENT_ADD_COMMENT_AFTER	32519
Processing Instruction davor hinzufügen...	ID_COMPONENT_ADD_PI_BEFORE	32520
Processing Instruction danach hinzufügen...	ID_COMPONENT_ADD_PI_AFTER	32521
Processing Instruction-Namen bearbeiten...	ID_COMPONENT_EDIT_PI	32524
Kommentar/Processing Instruction löschen	ID_COMPONENT_REMOVE_COMMENT_PI	32522
Inhalt als CDATA-Abschnitt schreiben	ID_COMPONENT_TOGGLE_CDATA	32525
Datenbankaktionen	ID_POPUP_DATABASETABLEACTIONS	32400
Datenbank abfragen...	ID_QUERY_DATABASE	32341
Links ausrichten	ID_COMPONENT_LEFTALIGNTREE	32338
Rechts ausrichten	ID_COMPONENT_RIGHTALIGNTREE	32340
Eigenschaften	ID_COMPONENT_PROPERTIES	32336

18.6.6 Menü "Verbindung"

Das Menü "Verbindung" enthält die folgenden Befehle:

© 2016-2022 Altova GmbH

Menübefehl	Befehlsname	ID
Idente Sub-Einträge automatisch verbinden	ID_CONNECTION_AUTOCONNECTCHILDREN	32342
Einstellungen für 'Idente Sub-Einträge verbinden'	ID_CONNECTION_SETTINGS	32344
Idente Sub-Einträge verbinden...	ID_CONNECTION_MAPCHILDELEMENTS	32343
Zielorientiert (Standard)	ID_POPUP_NORMALCONNECTION	32401
Alles kopieren (Sub-Einträge kopieren)	ID_POPUP_NORMALWITHCHILDREN_CONNECTION	32460
Quellorientiert (Mixed Content)	ID_POPUP_ORDERBYSOURCECONNECTION	32403
Eigenschaften	ID_POPUP_CONNECTION_SETTINGS	32398

18.6.7 Menü "Funktion"

Das Menü "Funktion" enthält die folgenden Befehle:

Menübefehl	Befehlsname	ID
Benutzerdefinierte Funktion erstellen...	ID_FUNCTION_CREATE_EMPTY	32380
Benutzerdefinierte Funktion von Auswahl erstellen...	ID_FUNCTION_CREATE_FROM_SELECTION	32381
Funktionseinstellungen	ID_FUNCTION_SETTINGS	32387
Funktion entfernen	ID_FUNCTION_REMOVE	32385
Input-Komponente einfügen...	ID_FUNCTION_INSERT_INPUT	32383
Output-Komponente einfügen...	ID_FUNCTION_INSERT_OUTPUT	32402

18.6.8 Menü "Ausgabe"

Das Menü "Ausgabe" enthält die folgenden Befehle:

Menübefehl	Befehlsname	ID
XSLT 1.0	ID_SELECT_LANGUAGE_XSLT	32433
XSLT 2.0	ID_SELECT_LANGUAGE_XSLT2	32434
XQuery	ID_SELECT_LANGUAGE_XQUERY	32432
Java	ID_SELECT_LANGUAGE_JAVA	32431

Menübefehl	Befehlsname	ID
C# (Sharp)	ID_SELECT_LANGUAGE_CSHARP	32430
C++	ID_SELECT_LANGUAGE_CPP	32429
Built-In Ausführungsprozessor	ID_SELECT_LANGUAGE_BUILTIN	32490
Ausgabedatei validieren	ID_XML_VALIDATE	32458
Ausgabedatei speichern...	IDC_FILE_SAVEGENERATEDOUTPUT	32321
Alle Ausgabedateien speichern...	IDC_FILE_SAVEALLGENERATEDOUTPUT	32374
Ausgabedatei neu generieren	ID_REGENERATE_PREVIEW_OUTPUT	32480
SQL-Script ausführen	ID_TRANSFORM_RUN_SQL	32442
Lesezeichen einfügen/löschen	ID_TOGGLE_BOOKMARK	32317
Nächstes Lesezeichen	ID_GOTONEXTBOOKMARK	32315
Vorhergehendes Lesezeichen	ID_GOTOPREVBOOKMARK	32314
Alle Lesezeichen löschen	ID_REMOVEALLBOOKMARKS	32313
Pretty-Print	ID_PRETTY_PRINT_OUTPUT	32363
Einstellungen für Textansicht	ID_TEXTVIEWSETTINGSDIALOG	32472

18.6.9 Menü "Debuggen"

Das Menü "Debuggen" enthält die folgenden Befehle:

Menübefehl	Befehlsname	ID
Debugger starten	ID_DEBUG_START	32540
Debugger anhalten	ID_DEBUG_STOP	32541
Einsteigen	ID_DEBUG_STEP_INTO	32545
Überspringen	ID_DEBUG_STEP_OVER	32551
Aussteigen	ID_DEBUG_STEP_OUT	32552
Minimaler Schritt	ID_DEBUG_STEP_NEXT_TRACE	32554

18.6.10 Menü "Ansicht"

Das Menü "Ansicht" enthält die folgenden Befehle:

© 2016-2022 Altova GmbH

Menübefehl	Befehlsname	ID
Annotationen anzeigen	ID_SHOW_ANNOTATION	32435
Datentypen anzeigen	ID_SHOW_TYPES	32437
Bibliothek in Funktionstitelleiste anzeigen	ID_VIEW_SHOWLIBRARYINFUNCTIONHEADER	32448
Tipps anzeigen	ID_SHOW_TIPS	32436
XBRL-Anzeigeoptionen	ID_VIEW_XBRL_DISPLAY_OPTIONS	32473
Ausgewählte Komponentenkonnektoren anzeigen	ID_VIEW_AUTOHIGHLIGHTCOMPONENTCONNECTIONS	32443
Quell- und Zielkonnektoren anzeigen	ID_VIEW_RECURSIVEAUTOHIGHLIGHT	32447
Vergrößern/Verkleinern...	ID_VIEW_ZOOM	32451
Zurück	ID_CMD_BACK	32479
Vorwärts	ID_CMD_FORWARD	32478
Statusleiste	ID_VIEW_STATUS_BAR	59393
Bibliotheksfenster	ID_VIEW_LIBRARY_WINDOW	32445
Meldungen	ID_VIEW_VALIDATION_OUTPUT	32450
Übersicht	ID_VIEW_OVERVIEW_WINDOW	32446
Projektfenster	ID_VIEW_PROJECT_WINDOW	32302
Werte	ID_DEBUG_VIEW_VALUES_WINDOW	32544
Kontext	ID_DEBUG_VIEW_CONTEXT_WINDOW	32546
Breakpoints	ID_DEBUG_VIEW_DEBUGPOINTS_WINDOW	32547

18.6.11 Menü "Extras"

Das Menü "Extras" enthält die folgenden Befehle:

Menübefehl	Befehlsname	ID
Globale Ressourcen	IDC_GLOBALRESOURCES	37401
<Plugin nicht geladen>	IDC_GLOBALRESOURCES_SUBMENUENTRY1	37408
Umgekehrtes Mapping erstellen	ID_CREATE_REVERSED_MAPPING	32489
Anpassen...	IDC_APP_TOOLS_CUSTOMIZE	32959

Menübefehl	Befehlsname	ID
Optionen...	ID_TOOLS_OPTIONS	32441

18.6.12 Menü "Fenster"

Das Menü "Fenster" enthält die folgenden Befehle:

Menübefehl	Befehlsname	ID
Überlappend	ID_WINDOW_CASCADE	57650
Horizontal anordnen	ID_WINDOW_TILE_HORZ	57651
Vertikal anordnen	ID_WINDOW_TILE_VERT	57652

18.6.13 Menü "Hilfe"

Das Menü "Hilfe" enthält die folgenden Befehle:

Menübefehl	Befehlsname	ID
Inhaltsverzeichnis...	IDC_HELP_CONTENTS	32966
Index...	IDC_HELP_INDEX	32967
Suchen...	IDC_HELP_SEARCH	32969
Software-Aktivierung...	IDC_ACTIVATION	32970
Bestellformular...	IDC_OPEN_ORDER_PAGE	32971
Registrieren...	IDC_REGISTRATION	32972
Auf Updates überprüfen...	IDC_CHECK_FOR_UPDATES	32973
MapForce-Produktvergleich...	IDC_PRODUCT_COMPARISON	32955
Support Center...	IDC_OPEN_SUPPORT_PAGE	32961
Fragen und Antworten im Web...	IDC_SHOW_FAQ	32962
Komponenten und Gratistools downloaden...	IDC_OPEN_COMPONENTS_PAGE	32963
MapForce im Internet..	IDC_OPEN_HOME_PAGE	32964
MapForce Training...	IDC_OPEN_TRAINING_PAGE	32965
Über MapForce...	ID_APP_ABOUT	57664

© 2016-2022 Altova GmbH

18.7 Objektreferenz

Objekte:
MapForceCommand 1959
MapForceCommands 1961
MapForceControl 1962
MapForceControlDocument 1970
MapForceControlPlaceHolder 1977

Um Zugriff auf die MapForce Standardfunktionalitäten zu erhalten, können auch Objekte des **MapForce Automation Interface** aufgerufen werden. Nähere Informationen dazu finden Sie unter MapForceControl.Application 1963, MapForceControlDocument.Document 1972 und MapForceControlPlaceHolder.Project 1978.

18.7.1 MapForceCommand

Eigenschaften:
`ID` 1960
`Label` 1960
`Name` 1961
`IsSeparator` 1960
`ToolTip` 1961
`StatusText` 1961
`Accelerator` 1960
`SubCommands` 1961

Beschreibung:
Jedes Command Objekt kann einer von drei möglichen Typen sein: ein ausführbarer Befehl, ein Befehlscontainer (z.B. ein Menü, ein Untermenü oder eine Symbolleiste) oder ein Menütrennzeichen. Um herauszufinden, welche Art von Informationen im aktuellen `Command`-Objekt gespeichert sind, fragen Sie seine Eigenschaften `ID`, `IsSeparator` und `SubCommands` folgendermaßen ab.

Das Befehlsobjekt ist...	**wenn...**
ein ausführbarer Befehl	• `ID` größer als Null ist • `IsSeparator` "false" ist • `SubCommands` leer ist
ein Befehlscontainer	• `ID` Null ist • `IsSeparator` "false" ist • `SubCommands` eine Sammlung von `Command` Objekten enthält
ein Trennzeichen	• `ID` Null ist • `IsSeparator` "true" ist

18.7.1.1 Accelerator

Eigenschaft: `Accelerator` als string

Beschreibung:
Gibt die für den Befehl definierte Zugriffstaste zurück. Wenn dem Befehl keine Zugriffstaste zugewiesen wurde, gibt diese Eigenschaft den leeren String zurück. Die String-Darstellung der Zugriffstaste hat das folgenden Format:

```
[ALT+][CTRL+][SHIFT+]key
```

`key` wird mittels der Windows Plattform SDK-Funktion `GetKeyNameText` konvertiert.

18.7.1.2 ID

Eigenschaft: `ID` als long

Beschreibung:
Mit dieser Eigenschaft wird der eindeutige Identifier des Befehls abgerufen. Die ID eines Befehls wird benötigt, um den Befehl (mittels `Exec`[1966]) auszführen oder seinen Status (mittels `QueryStatus`[1967]) abzurufen. Wenn der Befehl ein Container für andere Befehle (z.B. für ein Menü der obersten Ebene) oder ein Trennzeichen ist, ist die ID 0.

18.7.1.3 IsSeparator

Eigenschaft: `IsSeparator` als boolean

Beschreibung:
Die Eigenschaft gibt `true` zurück, wenn das Befehlsobjekt ein Menütrennzeichen ist; andernfalls wird `false` zurückgegebven. Siehe auch Command[1959].

18.7.1.4 Label

Eigenschaft: `Label` als string

Beschreibung:
Diese Eigenschaft ruft den Text des Befehls, wie er auf der grafischen Benutzeroberfläche von MapForce angezeigt wird, ab. Wenn der Befehl ein Trennzeichen ist, ist "Label" ein leerer String. Diese Eigenschaft kann für einige Symbolleistenbefehle, zu denen es keinen GUI-Text gibt, auch einen leeren String zurückgeben.

© 2016-2022 Altova GmbH

18.7.1.5 Name

Eigenschaft: `Name` als string

Beschreibung:
Diese Eigenschaft ruft den eindeutigen Namen des Befehls ab. Anhand dieses Werts kann die Symboldatei das Befehls, falls verfügbar, abgerufen werden. Die verfügbaren Symboldateien befinden sich im Ordner **<ApplicationFolder>\Examples\ActiveX\Images** Ihrer MapForce-Installation.

18.7.1.6 StatusText

Eigenschaft: `Label` als string

Beschreibung:
Der Statustext ist der Text, der in der Statusleiste von MapForce angezeigt wird, wenn der Befehl ausgewählt wird. Dies gilt nur für Befehlsobjekte, die keine Trennzeichen oder Container von anderen Befehlen sind. Andernfalls ist die Eigenschaft ein leerer String.

18.7.1.7 SubCommands

Eigenschaft: `SubCommands` als `Commands` 1961

Beschreibung:
Die Eigenschaft `SubCommands` ruft die Sammlung von `Command` 1959-Objekten, die dem aktuellen Befehl untergeordnet sind, auf. Die Eigenschaft gilt nur für Befehle, die Container für andere Befehle sind (Menüs, Untermenüs oder Symbolleisten). Bei solchen Container-Befehlen ist die `ID` auf `0` gesetzt und die Eigenschaft `IsSeparator` auf `false`.

18.7.1.8 ToolTip

Eigenschaft: `ToolTip` als string

Beschreibung:
Diese Eigenschaft ruft den Text, der als Tooltipp zu den einzelnen Befehlen angezeigt wird, ab. Wenn der Befehl keinen Tooltipp-Text hat, so gibt die Eigenschaft einen leeren String zurück.

18.7.2 MapForceCommands

Eigenschaften:
`Count` 1962
`Item` 1962

Beschreibung:

Sammlung von Command[1959]-Objekten für den Zugriff auf Befehlsbezeichnungen und IDs des MapForceControl. Diese Befehle können mit der Exec[1966] Methode ausgeführt werden und ihr Status kann mittels QueryStatus[1967] abgefragt werden.

18.7.2.1 Count

Eigenschaft: `Count` als long

Beschreibung:
Anzahl der Command[1959] Objekte auf dieser Ebene der Collection

18.7.2.2 Item

Eigenschaft: `Item` (n als long) als Command[1959]

Beschreibung:
Ruft den Befehl mit den Index n in dieser Sammlung auf. Der Index basiert auf 1.

18.7.3 MapForceControl

Eigenschaften:
IntegrationLevel[1964]
Appearance[1963]
Application[1963]
BorderStyle[1963]
CommandsList[1964]
EnableUserPrompts[1964]
MainMenu[1965]
Toolbars[1965]

Methoden:
Open[1967]
Exec[1966]
QueryStatus[1967]

Events:
OnUpdateCmdUI[1970]
OnOpenedOrFocused[1969]
OnCloseEditingWindow[1968]
OnFileChangedAlert[1969]
OnContextChanged[1968]
OnDocumentOpened[1968]
OnValidationWindowUpdated[1970]

Dieses Objekt ist ein vollständiges ActiveX Control und sollte nur sichtbar sein, wenn die MapForce Bibliothek im Applikationsebenenmodus verwendet wird.

`CLSID: A38637E9-5759-4456-A167-F01160CC22C1`

© 2016-2022 Altova GmbH

`ProgID: Altova.MapForceControl`

18.7.3.1 Eigenschaften

Es sind die folgenden Eigenschaften definiert:

`IntegrationLevel` 1964
`EnableUserPrompts` 1964
`Appearance` 1963
`BorderStyle` 1963

Befehlsbezogene Eigenschaften:
`CommandsList` 1964
`MainMenu` 1965
`Toolbars` 1965

Zugriff auf die MapForceAPI:
`Application` 1963

18.7.3.1.1 Appearance

Eigenschaft: `Appearance` als short

Dispatch Id: *-520*

Beschreibung:
Bei einem Wert, der nicht gleich `0` ist, wird ein Client-Rand rund um das Control angezeigt. Der Standardwert ist `0`.

18.7.3.1.2 Application

Eigenschaft: `Application` als Application

Dispatch Id: *1*

Beschreibung:
Mit der Eigenschaft `Application` erhalten Sie Zugriff auf das `Application` Objekt der vollständigen MapForce Automation Server API. Die Eigenschaft ist schreibgeschützt.

18.7.3.1.3 BorderStyle

Eigenschaft: `BorderStyle` als short

Dispatch Id: *-504*

Beschreibung:
Bei einem Wert `1` wird das Control mit einer dünnen Umrandung angezeigt. Der Standardwert ist `0`.

18.7.3.1.4 CommandsList

Eigenschaft: `CommandList` als `Commands` 1961 (schreibgeschützt)

Dispatch Id: *1004*

Beschreibung:
Diese Eigenschaft gibt eine flache Liste aller Befehle zurück, die mit MapForceControl verfügbar sind. Um Befehle, geordnet nach Menüstruktur, abzurufen, verwenden Sie `MainMenu` 1965. Um Symbolleistenbefehle abzurufen, verwenden Sie `Toolbars` 1965.

```
public void GetAllMapForceCommands()
{
    // Get all commands from the MapForce ActiveX control assigned to the current form
    MapForceControlLib.MapForceCommands commands = this.axMapForceControl1.CommandList;
    // Iterate through all commands
    for (int i = 0; i < commands.Count; i++)
    {
      // Get each command by index and output it to the console
      MapForceControlLib.MapForceCommand cmd = axMapForceControl1.CommandList[i];
      Console.WriteLine("{0} {1} {2}", cmd.ID, cmd.Name, cmd.Label.Replace("&", ""));
    }
}
```

C#-Beispiel

18.7.3.1.5 EnableUserPrompts

Eigenschaft: `EnableUserPrompts` als boolean

Dispatch Id: *1006*

Beschreibung:
Wenn Sie diese Eigenschaft auf *false* setzen, wird die Eingabeaufforderung im Control deaktiviert. Der Standardwert ist *true*.

18.7.3.1.6 IntegrationLevel

Eigenschaft: `IntegrationLevel` als `ICActiveXIntegrationLevel` 1980

© 2016-2022 Altova GmbH

Dispatch Id: *1000*

Beschreibung:
Die Eigenschaft `IntegrationLevel` bestimmt den Operationsmodus des Control. Nähere Informationen dazu siehe auch Integration auf Applikationsebene[1924] und Integration auf Dokumentebene[1927].

Anmerkung: Diese Eigenschaft muss unbedingt sofort nach Erstellung des `MapForceControl` Objekts definiert werden.

18.7.3.1.7 MainMenu

Eigenschaft: `MainMenu als` Command[1959] (schreibgeschützt)

Dispatch Id: *1003*

Beschreibung:
Diese Eigenschaft enthält Informationen über die Struktur und die Befehle im MapForceControl-Hauptmenü als `Command`-Objekt. Das `Command`-Objekt enthält alle verfügbaren Untermenüs von MapForce (z.B. Datei, Bearbeiten, Ansicht, usw.). Verwenden Sie die Eigenschaft `SubCommands` der Eigenschaft `MainMenu`, um die Untermenüobjekte abzurufen. Jedes Untermenü ist ebenfalls ein `Command`-Objekt. Sie können bei jedem Untermenü weiter durch dessen `SubCommands`-Eigenschaft iterieren, um die jeweiligen Child-Befehle und Trennzeichen dieser Untermenüs abzurufen (Auf diese Art können Sie z.B. das Applikationsmenü programmatisch erstellen). Beachten Sie, dass einige Menübefehle als Container ("Parents") für andere Menübefehle dienen. In diesen Fällen haben diese ebenfalls eine Eigenschaft `SubCommands`. Um die Struktur aller Menübefehle programmatisch abzurufen, müssen Sie wahrscheinlich eine rekursive Funktion erstellen, wie unter Abrufen von Befehlsinformationen[1934] für C# gezeigt.

```
public void GetMapForceMenus()
{
    // Get the main menu from the MapForce ActiveX control assigned to the current form
    MapForceControlLib.MapForceCommand mainMenu = this.axMapForceControl1.MainMenu;

    // Loop through entries of the main menu (e.g. File, Edit, etc.)
    for (int i = 0; i < mainMenu.SubCommands.Count; i++)
    {
      MapForceControlLib.MapForceCommand menu = mainMenu.SubCommands[i];
      Console.WriteLine("{0} menu has {1} children items (including separators)",
menu.Label.Replace("&", ""), menu.SubCommands.Count);
    }
}
```

C# example

18.7.3.1.8 Toolbars

Eigenschaft: `Toolbars` als Commands[1961] (schreibgeschützt)

Dispatch Id: *1005*

Beschreibung:
Diese Eigenschaft enthält Informationen über die Struktur von MapForceControl-Symbolleisten, als `Command`-Objekt. Das `Command`-Objekt enthält alle verfügbaren Symbolleisten von MapForce. Verwenden Sie die Eigenschaft `SubCommands` der Eigenschaft `Toolbars`, um die Symbolleisten abzurufen. Jede Symbolleiste ist ebenfalls ein `Command`-Objekt. Sie können bei jeder Symbolleiste weiter durch deren `SubCommands`-Eigenschaft iterieren, um deren Befehle abzurufen (Auf diese Art können Sie z.B. die Symbolleisten der Applikation programmatisch erstellen).

```
public void GetMapForceToolbars()
{
    // Get the application toolbars from the MapForce ActiveX control assigned to the
current form
    MapForceControlLib.MapForceCommands toolbars = this.axMapForceControl1.Toolbars;

    // Iterate through all toolbars
    for (int i = 0; i < toolbars.Count; i++)
    {
      MapForceControlLib.MapForceCommand toolbar = toolbars[i];
      Console.WriteLine();
      Console.WriteLine("The toolbar \"{0}\" has the following commands:",
toolbar.Label);

      // Iterate through all commands of this toolbar
      for (int j = 0; j < toolbar.SubCommands.Count; j++)
      {
         MapForceControlLib.MapForceCommand cmd = toolbar.SubCommands[j];
         // Output only command objects that are not separators
         if (!cmd.IsSeparator)
         {
            Console.WriteLine("{0}, {1}, {2}", cmd.ID, cmd.Name, cmd.Label.Replace("&",
""));
         }
      }
    }
}
```

C# example

18.7.3.2 Methoden

Es sind die folgenden Methoden definiert:

Open 1967
Exec 1966
QueryStatus 1967

18.7.3.2.1 Exec

Methode: `Exec` (`nCmdID` als long) als boolean

© 2016-2022 Altova GmbH

Dispatch Id: *6*

Beschreibung:
`Exec` ruft den MapForce Befehl mit der ID `nCmdID` auf. Wenn der Befehl ausgeführt werden kann, gibt die Methode `true` zurück. Eine Liste aller verfügbaren Befehle finden Sie unter `CommandsList`[1964]. Um den Status eines Befehls abzurufen, verwenden Sie `QueryStatus`[1967].

18.7.3.2.2 Open

Methode: `Open` (`strFilePath` als string) als boolean

Dispatch Id: *5*

Beschreibung:
Das Ergebnis der Methode ist von der Erweiterung abhängig, die im Argument `strFilePath` übergeben wird. Ist die Dateierweiterung `.sps`, wird ein neues Dokument geöffnet. Ist die Dateierweiterung `.svp`, wird das entsprechende Projekt geöffnet. Wird eine andere Dateierweiterung in die Methode übergeben, versucht das Control, die Datei als neue Komponente in das aktive Dokument zu laden.

Verwenden Sie diese Methode nicht, um Dokumente oder Projekte bei Verwendung des Control auf Dokumentebene zu laden. Verwenden Sie statt dessen `MapForceControlDocument.Open`[1974] und `MapForceControlPlaceHolder.OpenProject`[1979].

18.7.3.2.3 QueryStatus

Methode: `QueryStatus` (`nCmdID` als long) als long

Dispatch Id: *7*

Beschreibung:
`QueryStatus` gibt den Status "enabled/disabled" und "checked/unchecked" des von `nCmdID`. definierten Befehls zurück. Der Status wird als Bitmaske zurückgegeben.

Bit	Wert	Name	Bedeutung
0	1	Supported	Setzen, wenn der Befehl unterstützt wird.
1	2	Enabled	Setzen, wenn der Befehl aktiviert ist (ausgeführt werden kann).
2	4	Checked	Setzen, wenn der Befehl angehakt ist.

Das bedeutet, dass die Befehls-ID bei Rückgabe von 0 durch `QueryStatus` nicht als gültiger MapForce Befehl erkannt wird. Wenn `QueryStatus` einen Wert `1` oder `5` zurückgibt, wird der Befehl deaktiviert.

18.7.3.3 Events

Das MapForceControl ActiveX Control stellt die folgenden Verbindungspunkt-Events bereit:

OnUpdateCmdUI 1970
OnOpenedOrFocused 1969
OnCloseEditingWindow 1968
OnFileChangedAlert 1969
OnContextChanged 1968

OnDocumentOpened 1968
OnValidationWindowUpdated 1970

18.7.3.3.1 OnCloseEditingWindow

Event: `OnCloseEditingWindow` (i_strFilePath als String) als boolean

Dispatch Id: *1002*

Beschreibung:
Dieses Event wird ausgelöst, wenn MapForce ein bereits geöffnetes Dokument schließen muss. Als Antwort auf dieses Event müssen Clients das mit *i_strFilePath* verknüpfte Bearbeitungsfenster schließen. Bei Rückgabe von *true* von diesem Event, wird angezeigt, dass der Client das Dokument geschlossen hat. Clients können *false* zurückgeben, wenn keine bestimmte Behandlung erforderlich ist und MapForceControl versuchen soll, das Bearbeitungsfenster zu schließen und das damit verknüpfte Dokument-Control zu zerstören.

18.7.3.3.2 OnContextChanged

Event: `OnContextChanged` (`i_strContextName` als String, `i_bActive` als bool) als bool

Dispatch Id: *1004*

Beschreibung:
Dieses Event wird in MapForce nicht verwendet

18.7.3.3.3 OnDocumentOpened

Event: `OnDocumentOpened` (`objDocument` als Document)

Dispatch Id: *1*

Beschreibung:
Dieses Event wird immer, wenn ein Dokument geöffnet wird, ausgelöst. Das Argument `objDocument` ist ein `Document` Objekt aus dem MapForce Automation Interface und kann dazu verwendet werden, weitere Details zum Dokument abzurufen oder weitere Operationen durchzuführen. Bei Integration auf Dokumentebene ist es oft besser, stattdessen das Event `MapForceControlDocument.OnDocumentOpened` 1976 zu verwenden.

© 2016-2022 Altova GmbH

18.7.3.3.4 OnFileChangedAlert

Event: `OnFileChangedAlert` (`i_strFilePath` als String) als bool

Dispatch Id: *1001*

Beschreibung:
Dieses Event wird ausgelöst, wenn eine mit MapForceControl geladene Datei auf der Festplatte von einer anderen Applikation geändert wurde. Clients sollten true zurückgeben, wenn Sie das Event behandelt haben oder false, wenn MapForce es auf die übliche Art behandeln soll, also fragen soll, ob das Dokument neu geladen werden soll.

18.7.3.3.5 OnLicenseProblem

Event: `OnLicenseProblem` (`i_strLicenseProblemText` als String)

Dispatch Id: *1005*

Beschreibung:
Dieses Event wird ausgelöst, wenn MapForceControl feststellt, dass für dieses Control keine gültige Lizenz vorhanden ist. Wenn die Lizenz auf eine bestimmte Anzahl an Benutzern beschränkt ist, kann dies auch einige Zeit nach Initialisierung des Control geschehen. Dieses Event sollte dazu verwendet werden, um den Zugriff auf die Funktionalität dieses Control zu deaktivieren. Nach diesem Event blockiert dieses Control den Zugriff auf seine Funktionalitäten (und zeigt z.B. leere Fenster in seinen Controls an und gibt bei Anfragen Fehlermeldungen zurück).

18.7.3.3.6 OnOpenedOrFocused

Event: `OnOpenedOrFocused` (`i_strFilePath` als String, `i_bOpenWithThisControl` als bool)

Dispatch Id: *1000*

Beschreibung:
Bei Integration auf Applikationsebene informiert dieses Event Clients, dass ein Dokument von MapForce geöffnet oder aktiv gemacht wurde.

Bei Integration auf Dokumentebene gibt dieses Event dem Client die Anweisung, die Datei `i_strFilePath` in einem Dokumentfenster zu öffnen. Wenn die Datei bereits offen ist, sollte das entsprechende Dokumentfenster zum aktiven Fenster gemacht werden.

Wenn `i_bOpenWithThisControl` `true` ist, muss das Dokument mit MapForceControl geöffnet werden, da ein interner Zugriff erforderlich ist. Andernfalls kann die Datei mit anderen Editoren geöffnet werden.

18.7.3.3.7 OnToolWindowUpdated

Event: `OnToolWindowUpdated(pToolWnd` als long `)`

Dispatch Id: *1006*

Beschreibung:
Dieses Event wird ausgelöst, wenn das Fenster "Extras" aktualisiert wird.

18.7.3.3.8 OnUpdateCmdUI

Event: `OnUpdateCmdUI ()`

Dispatch Id: *1003*

Beschreibung:
Wird häufig aufgerufen, um dem Programmierer die Möglichkeit zu geben, den Status von MapForce Befehls mittels `MapForceControl.QueryStatus`[1967] zu überprüfen. Führen Sie an diesem Callback keine langen Operationen durch.

18.7.3.3.9 OnValidationWindowUpdated

Event: `OnValidationWindowUpdated` ()

Dispatch Id: *3*

Beschreibung:
Dieses Event wird ausgelöst, wenn das Fenster "Validierungsausgabe" mit neuen Informationen aktualisiert wird.

18.7.4 MapForceControlDocument

Eigenschaften:
`Appearance`[1971]
`BorderStyle`[1972]
`Document`[1972]
`IsModified`[1972]
`Path`[1972]
`ReadOnly`[1973]

Methoden:
`Exec`[1973]
`New`[1973]
`Open`[1974]
`QueryStatus`[1974]

© 2016-2022 Altova GmbH

Reload 1974
Save 1975
SaveAs 1975

Events:
OnDocumentOpened 1976
OnDocumentClosed 1976
OnModifiedFlagChanged 1977
OnContextChanged 1975
OnFileChangedAlert 1976
OnActivate 1975

Wenn das MapForceControl im Modus auf Dokumentebene integriert ist, wird jedes Dokument in einem eigenen Objekt vom Typ `MapForceControlDocument` angezeigt. Das `MapForceControlDocument` enthält immer nur ein Dokument, kann aber wiederverwendet werden, um hintereinander mehrere Dateien anzuzeigen.

Dieses Objekt ist ein komplettes ActiveX Control.

```
CLSID: DFBB0871-DAFE-4502-BB66-08CEB7DF5255
ProgID: Altova.MapForceControlDocument
```

18.7.4.1 Eigenschaften

Es sind die folgenden Eigenschaften definiert:

ReadOnly 1973
IsModified 1972
Path 1972
Appearance 1971
BorderStyle 1972

Zugriff auf die MapForceAPI:
Document 1972

18.7.4.1.1 Appearance

Eigenschaft: `Appearance` als short

Dispatch Id: *-520*

Beschreibung:
Bei einem Wert, der nicht gleich `0` ist, wird ein Client-Rand rund um das Control angezeigt. Der Standardwert ist `0`.

18.7.4.1.2 BorderStyle

Eigenschaft: `BorderStyle` als short

Dispatch Id: *-504*

Beschreibung:
Bei einem Wert `1` wird das Control mit einer dünnen Umrandung angezeigt. Der Standardwert ist `0`.

18.7.4.1.3 Document

Eigenschaft: `Document` als Document

Dispatch Id: *3*

Beschreibung:
Mit der Eigenschaft `Document` erhalten Sie Zugriff auf das `Document` Objekt der MapForce Automation Server API. Diese Schnittstelle bietet zusätzliche Funktionalitäten, die mit dem im Control geladenen Dokument verwendet werden können. Die Eigenschaft ist schreibgeschützt.

18.7.4.1.4 IsModified

Eigenschaft: `IsModified` als boolean (schreibgeschützt)

Dispatch Id: *1006*

Beschreibung:
`IsModified` ist *true,* wenn der Dokumentinhalt seit dem letzten Öffnen, Neuladen oder Speichern geändert wurde. Andernfalls ist es *false*.

18.7.4.1.5 Path

Eigenschaft: `Path` als String

Dispatch Id: *1005*

Beschreibung:
Definiert den vollständigen Pfadnamen des im Control geladenen Dokuments bzw. ruft diesen ab.

© 2016-2022 Altova GmbH

18.7.4.1.6 ReadOnly

Eigenschaft: `ReadOnly` als boolean

Dispatch Id: *1007*

Beschreibung:
Mit Hilfe dieser Eigenschaft können Sie den schreibgeschützten Status des Dokuments aktivieren und deaktivieren. Wenn `ReadOnly` `true` ist, können keine Änderungen vorgenommen werden.

18.7.4.2 Methoden

Es sind die folgenden Methoden definiert:

Behandlung von Dokumenten:
`New` 1973
`Open` 1974
`Reload` 1974
`Save` 1975
`SaveAs` 1975

Behandlung von Befehlen:
`Exec` 1973
`QueryStatus` 1974

18.7.4.2.1 Exec

Methode: `Exec` (`nCmdID` as long) als boolean

Dispatch Id: *8*

Beschreibung:
`Exec` ruft den MapForce Befehl mit der ID `nCmdID` auf. Wenn der Befehl ausgeführt werden kann, gibt die Methode `true` zurück. Diese Methode sollte nur dann aufgerufen werden, wenn in der Applikation gerade ein aktives Dokument verfügbar ist.

Um Befehle, geordnet nach Menüstruktur, abzurufen, verwenden Sie die Eigenschaft `MainMenu` 1965 von MapForceControl. Um Symbolleistenbefehle abzurufen, verwenden Sie die Eigenschaft `Toolbars` 1965 von MapForceControl.

18.7.4.2.2 New

Methode: `New` () als boolean

Dispatch Id: *1000*

Beschreibung:
Diese Methode initialisiert ein neues Mapping innerhalb des Control.

18.7.4.2.3 Open

Methode: `Open` (`strFilePath` als string) als boolean

Dispatch Id: *1001*

Beschreibung:
`Open` lädt die Datei `strFileName` als das neue Dokument in das Control.

18.7.4.2.4 QueryStatus

Methode: `QueryStatus` (`nCmdID` als long) als long

Dispatch Id: *9*

Beschreibung:
`QueryStatus` gibt den Status "enabled/disabled" und "checked/unchecked" des von `nCmdID`. definierten Befehls zurück. Der Status wird als Bitmaske zurückgegeben.

Bit	Wert	Name	Bedeutung
0	1	Supported	Setzen, wenn der Befehl unterstützt wird.
1	2	Enabled	Setzen, wenn der Befehl aktiviert ist (ausgeführt werden kann).
2	4	Checked	Setzen, wenn der Befehl angehakt ist.

Das bedeutet, dass die Befehls-ID bei Rückgabe von 0 durch `QueryStatus` nicht als gültiger MapForce Befehl erkannt wird. Wenn `QueryStatus` einen Wert `1` oder `5` zurückgibt, ist der Befehl deaktiviert. Der Client sollte die `QueryStatus` Methode des Dokument-Control aufrufen, wenn in der Applikation gerade ein aktives Dokument vorhanden ist.

18.7.4.2.5 Reload

Methode: `Reload` () als boolean

Dispatch Id: *1002*

Beschreibung:
`Reload` aktualisiert den Dokumentinhalt aus dem Dateisystem.

© 2016-2022 Altova GmbH

18.7.4.2.6 Save

Methode: `Save` () als boolean

Dispatch Id: *1003*

Beschreibung:
`Save` speichert das aktuelle Dokument unter dem Pfad `Path`[1972].

18.7.4.2.7 SaveAs

Methode: `SaveAs` (`strFileName` als string) als boolean

Dispatch Id: *1004*

Beschreibung:
`SaveAs` setzt `Path`[1972] auf *strFileName* und speichert das Dokument anschließend unter diesem Pfad.

18.7.4.3 Events

Das MapForceControlDocument ActiveX Control stellt die folgenden Verbindungspunkt-Events zur Verfügung:

`OnDocumentOpened`[1976]
`OnDocumentClosed`[1976]
`OnModifiedFlagChanged`[1977]
`OnContextChanged`[1975]
`OnFileChangedAlert`[1976]
`OnActivate`[1975]
`OnSetEditorTitle`[1977]

18.7.4.3.1 OnActivate

Event: `OnActivate` ()

Dispatch Id: *1005*

Beschreibung:
Dieses Event wird ausgelöst, wenn das Dokument-Control aktiviert ist, den Fokus hat und bereit für die Benutzereingabe ist.

18.7.4.3.2 OnContextChanged

Event: `OnContextChanged` (`i_strContextName` als String, `i_bActive` als bool) als bool

Dispatch Id: *1004*

Beschreibung: Keine

18.7.4.3.3 OnDocumentClosed

Event: `OnClosed` ()

Dispatch Id: *1001*

Beschreibung:
Dieses Event wird immer dann ausgelöst, wenn das in dieses Control geladene Event geschlossen wird. Das Argument `objDocument` ist ein `Document` Objekt aus dem MapForce Automation Interface und sollte mit Vorsicht verwendet werden.

18.7.4.3.4 OnDocumentOpened

Event: `OnDocumentOpened` (`objDocument` als Document)

Dispatch Id: *1000*

Beschreibung:
Dieses Event wird immer dann ausgelöst, wenn ein Dokument in diesem Control geöffnet wird. Das Argument `objDocument` ist ein `Document` Objekt aus dem MapForce Automation Interface und dient dazu, nähere Informationen über das Dokument abzurufen oder weitere Operationen auszuführen.

18.7.4.3.5 OnDocumentSaveAs

Event: `OnContextDocumentSaveAs` (`i_strFileName` als String)

Dispatch Id: *1007*

Beschreibung:
Dieses Event wird ausgelöst, wenn dieses Dokument intern unter einem neuen Namen gespeichert wird.

18.7.4.3.6 OnFileChangedAlert

Event: `OnFileChangedAlert` () als bool

Dispatch Id: *1003*

Beschreibung:
Dieses Event wird ausgelöst, wenn eine in dieses Control geladene Datei auf der Festplatte von einer anderen Applikation geändert wurde. Clients sollten true zurückgeben, wenn Sie das Event behandelt haben oder false, wenn MapForce es auf die übliche Art behandeln soll, also fragen soll, ob das Dokument neu geladen werden soll.

© 2016-2022 Altova GmbH

18.7.4.3.7 OnModifiedFlagChanged

Event: `OnModifiedFlagChanged` (`i_bIsModified` als boolean)

Dispatch Id: *1002*

Beschreibung:
Dieses Event wird immer dann ausgelöst, wenn das Dokument zwischen dem Status "geändert" und "nicht geändert" wechselt. Der Parameter *i_bIsModifed* ist *true,* wenn sich der Inhalt des Dokuments vom ursprünglichen Inhalt unterscheidet und ist andernfalls *false*.

18.7.4.3.8 OnSetEditorTitle

Event: OnSetEditorTitle ()

Dispatch Id: *1006*

Beschreibung:
Dieses Event wird ausgelöst, wenn das enthaltene Dokument intern umbenannt wird.

18.7.5 MapForceControlPlaceHolder

Eigenschaften für alle Arten von Platzhalterfenstern:
`PlaceholderWindowID` 1978

Eigenschaften für das Projekt-Platzhalterfenster:
`Project` 1978

Methoden für das Projekt-Platzhalterfenster:
`OpenProject` 1979
`CloseProject` 1979

Das `MapForceControlPlaceHolder` Control dient dazu, die zusätzlichen MapForce Fenster, wie Übersicht, Bibliothek oder das Projektfenster anzuzeigen. Es wird wie jedes andere ActiveX Control verwendet und kann überall in die Client-Applikation platziert werden.

```
CLSID: FDEC3B04-05F2-427d-988C-F03A85DE53C2
ProgID: Altova.MapForceControlPlaceHolder
```

18.7.5.1 Eigenschaften

Es sind die folgenden Eigenschaften definiert:

`PlaceholderWindowID` 1978

Zugriff auf die MapForceAPI:
Project[1978]

18.7.5.1.1 Label

Eigenschaft: `Label` als String (schreibgeschützt)

Dispatch Id: *1001*

Beschreibung:
Mit dieser Eigenschaft erhalten Sie Zugriff auf den Titel des Platzhalters. Diese Eigenschaft ist schreibgeschützt.

18.7.5.1.2 PlaceholderWindowID

Eigenschaft: `PlaceholderWindowID` als `MapForceControlPlaceholderWindow`[1980]

Dispatch Id: *1*

Beschreibung:
Mit Hilfe dieser Eigenschaft weiß das Objekt, welches MapForce Fenster im Client-Bereich des Control angezeigt werden soll. Die `PlaceholderWindowID` kann jederzeit auf jeden gültigen Wert der `MapForceControlPlaceholderWindow`[1980] Enumeration gesetzt werden. Das Control ändert seinen Status sofort und zeigt das neue MapForce Fenster an.

18.7.5.1.3 Project

Eigenschaft: `Project` als Project (schreibgeschützt)

Dispatch Id: *2*

Beschreibung:
Mit der Eigenschaft `Project` erhalten Sie Zugriff auf das `Project` Objekt der MapForce Automation Server API. Diese Schnittstelle bietet zusätzliche Funktionalitäten, die mit dem in das Control geladenen Projekt verwendet werden können. Die Eigenschaft gibt nur dann eine gültige Projektschnittstelle zurück, wenn das Platzhalterfenster `PlaceholderWindowID`[1978] mit einem Wert von `MapForceXProjectWindow (=3)` hat. Die Eigenschaft ist schreibgeschützt.

18.7.5.2 Methoden

Es sind die folgenden Methoden definiert:

`OpenProject`[1979]

© 2016-2022 Altova GmbH

CloseProject[1979]

18.7.5.2.1 OpenProject

Methode: `OpenProject` (`strFileName` als string) als boolean

Dispatch Id: *3*

Beschreibung:
`OpenProject` lädt die Datei `strFileName` als das neue Projekt in das Control. Wenn das Platzhalterfenster eine `PlaceholderWindowID`[1978] hat, die nicht mit `XMLSpyXProjectWindow (=3)`übereinstimmt, schlägt die Methode fehl.

18.7.5.2.2 CloseProject

Methode: `CloseProject` ()

Dispatch Id: *4*

Beschreibung:
`CloseProject` schließt das in das Control geladene Projekt. Wenn das Platzhalterfenster eine `PlaceholderWindowID`[1978] hat, die nicht mit `MapForceXProjectWindow (=3)`übereinstimmt, schlägt die Methode fehl.

18.7.5.3 Events

Das MapForceControlPlaceholder ActiveX Control stellt die folgenden Verbindungspunkt-Events zur Verfügung:

`OnModifiedFlagChanged`[1979]

18.7.5.3.1 OnModifiedFlagChanged

Event: `OnModifiedFlagChanged` (`i_bIsModified` als boolean)

Dispatch Id: *1*

Beschreibung:
Dieses Event wird nur bei Platzhalter-Controls mit einer `PlaceholderWindowID`[1978] von `MapForceXProjectWindow (=3)` ausgelöst. Das Event wird immer dann ausgelöst, wenn der Projektinhalt sich zwischen dem Status "geändert" und "nicht geändert" ändert. Der Parameter *i_bIsModifed* ist *true,* wenn sich der Projektinhalt vom ursprünglichen Inhalt unterscheidet und *false*, wenn dies nicht der Fall ist.

18.7.5.3.2 OnSetLabel

Event: `OnSetLabel` (i_strLabel als String)

Dispatch Id: *1000*

Beschreibung:
Dieses Event wird immer dann ausgelöst, wenn die Bezeichnung eines Platzhalter-Control-Fensters geändert wird.

18.7.6 Enumerationen

Es sind die folgenden Enumerationen definiert:

`ICActiveXIntegrationLevel`[1980]
`MapForceControlPlaceholderWindow`[1980]

18.7.6.1 ICActiveXIntegrationLevel

Mögliche Werte für die Eigenschaft IntegrationLevel[1964] für das MapForceControl.

```
ICActiveXIntegrationOnApplicationLevel  = 0
ICActiveXIntegrationOnDocumentLevel     = 1
```

18.7.6.2 MapForceControlPlaceholderWindow

Diese Enumeration enthält die Liste der unterstützten zusätzlichen MapForce Fenster.

```
MapForceXNoWindow                = -1
MapForceXLibraryWindow           = 0
MapForceXOverviewWindow          = 1
MapForceXValidationWindow        = 2
MapForceXProjectWindow           = 3
MapForceXDebuggerValuesWindow    = 4
MapForceXDebuggerContextWindow   = 5
MapForceXDebuggerPointsWindow    = 6
```


© 2016-2022 Altova GmbH

19 Anhänge

Diese Anhänge enthalten technische Informationen über MapForce, seine Aspekte und die Lizenzierung. Des Weiteren finden Sie darin eine Liste von Schlüsselbegriffen in MapForce und mit MapForce in Zusammenhang stehenden Produkten. Dieser Abschnitt ist in die folgenden Unterabschnitte gegliedert:

- Anmerkungen zur Unterstützung 1982
- Prozessoren 1988
- Technische Daten 2106
- Lizenzinformationen 2109

19.1 Anmerkungen zur Unterstützung

MapForce® ist eine 32/64-Bit-Windows-Applikation, die auf den folgenden Betriebssystemen läuft:

- Windows 7 SP1 mit Plattformupdate, Windows 8, Windows 10, Windows 11
- Windows Server 2008 R2 SP1 mit Plattformupdate oder höher

64-Bit-Unterstützung steht für die Enterprise und die Professional Edition zur Verfügung.

MapForce ist optional als Plug-in für die folgenden integrierten Entwicklungsumgebungen erhältlich:

- Visual Studio 2012/2013/2015/2017/2019/2022, siehe MapForce Plug-in für Visual Studio 1492
- Eclipse 2022-09, 2022-06, 2022-03, 2021-12, siehe MapForce Plug-in für Eclipse 1495.

MapForce lässt sich mit den folgenden Microsoft Office-Produkten integrieren:

- MapForce kann Daten von oder auf Access-Datenbanken mappen. Welche Versionen unterstützt werden, finden Sie unter Datenbanken und MapForce 165
- MapForce kann Mapping-Dokumentation in Word-Versionen ab Version 2000 generieren, siehe Generieren und Anpassen von Mapping-Dokumentation 1399.
- MapForce kann Daten von oder auf Excel-Arbeitsblätter ab Excel 2007 mappen, siehe Microsoft OOXML Excel 2007+ 738

19.1.1 Unterstützte Quellen und Ziele

Wenn Sie die Transformationssprache eines MapForce-Mappings ändern, kann es vorkommen, dass bestimmte Funktionalitäten für diese spezifische Sprache nicht zur Verfügung stehen. Die folgende Tabelle enthält eine Übersicht über die Kompatibilität von Mapping-Formaten und Transformationssprachen in **MapForce Enterprise Edition**.

Anmerkungen:

- *Built-in* bedeutet, dass Sie das Mapping durch Klick auf das Register **Ausgabe** in MapForce oder mit MapForceServer ausführen können.

Mapping-Format	XSLT 1.0	XSLT 2.0	XSLT 3.0	XQuery	C++	C#	Java	BUILT-IN
XML[1]	●	●	●	●	●	●	●	●
CSV und Text					●	●	●	●
Text über MapForce FlexText 684					●	●	●	●
JSON						●	●	●
XML[2]					●	●	●	●

© 2016-2022 Altova GmbH

Mapping-Format		XSLT 1.0	XSLT 2.0	XSLT 3.0	XQuery	C++	C#	Java	BUILT-IN
Excel 2007 und höher[3]	Als Datenquelle		●	●			●	●	●
	Als Datenziel (Generierung einer neuen Datei)						●	●	●
	Als Datenziel (Aktualisierung einer vorhandenen Datei)								●
EDI	ASC X12					●	●	●	●
	HIPAA X12					●	●	●	●
	HL7 Version 2					●	●	●	●
	HL7 Version 3	●	●	●	●	●	●	●	●
	IATA PADIS					●	●	●	●
	NCPDP SCRIPT EDIFACT-Format					●	●	●	●
	NCPDP SCRIPT XML-Format	●	●	●	●	●	●	●	●
	SAP IDoc					●	●	●	●
	SWIFT MT								●
	SWIFT MX / ISO 20022	●	●	●	●	●	●	●	●
	TRADACOMS					●	●	●	●

Mapping-Format		XSLT 1.0	XSLT 2.0	XSLT 3.0	XQuery	C++	C#	Java	BUILT-IN
	UN/EDIFACT					●	●	●	●
Datenbanken[4]	ADO					●	●		●
	ADO.NET						●		●
	JDBC							●	●
	Natives SQLite								●
	Natives PostgreSQL								●
	ODBC					●	●	●	●
Webservices[5]	REST								●
	SOAP						●	●	●
Protocol Buffer									●
Rohe Binärdateien									●

Fußnoten:

1. XML wird mit Verarbeitung digitaler Signaturen nur in BUILT-IN unterstützt.
2. XBRL wird mit Verarbeitung digitaler Signaturen nur in BUILT-IN unterstützt.
3. Dies bezieht sich auf Dateiformate ab Excel 2007 im *Office Open XML*-Format (.xlsx). Beachten Sie, dass auch für das Format *Strict Open XML Spreadsheet* die Dateierweiterung .xlsx verwendet wird, dass dieses Format jedoch nur als Datenquelle (und nicht als Datenziel) unterstützt wird. Nähere Informationen dazu siehe Microsoft OOXML Excel 2007+ 738.
4. Je nach Datenbanktyp und Zielumgebung gelten Einschränkungen. Nähere Informationen dazu finden Sie unter Datenbankmappings in verschiedenen Ausführungsumgebungen 166.
5. Dies bezieht sich auf den Aufruf von Webservices und nicht die Generierung von Webservices anhand von Mappings. Beachten Sie, dass Client-Zertifikate, Authentifzierung im Vorhinein und WS-Security für SOAP Webservices im generierten C#- und Java-Code nicht unterstützt werden. Nähere Informationen dazu finden Sie unter Aufrufen von Webservices 610.

19.1.2 Unterstützte Funktionalitäten im generierten Code

In der folgenden Tabelle finden Sie eine Liste der Funktionalitäten, die für die Codegenerierung relevant sind und Informationen dazu, inwieweit diese in der **MapForce Enterprise Edition** in der jeweiligen Sprache unterstützt werden.

© 2016-2022 Altova GmbH

Funktion	XSLT 1.0	XSLT 2.0	XSLT 3.0	XQuery	C++	C#	Java	BUILT-IN
Bereitstellung von Parametern für das Mapping 867	●	●	●	●	●	●	●	●
Dynamische Bereitstellung der Input-Dateinamen über das Mapping 1371	●	●	●	●	●	●	●	●
Dynamische Bereitstellung von Platzhalterdateinamen als Mapping-Input 1371 1		●	●	●	●	●	●	●
Dynamische Generierung der Ausgabedateinamen anhand des Mappings 1371		●	●		●	●	●	●
Rückgabe von String-Werten aus einem Mapping 878	●	●	●		●	●	●	●
Variablen 882		●	●	●	●	●	●	●
Sortierkomponenten 908		●	●	●				●
Gruppierungsfunktionen 931		●	●		●	●	●	●
Filter 914	●	●	●	●	●	●	●	●
Join-Komponenten 896								●
Wertezuordnungskomponenten 921	●	●	●	●	●	●	●	●
Standardwerte und Node-Funktionen 951								●
Mapping-Ausnahmeereignisse 938		●	●	●	●	●	●	●

Funktion	XSLT 1.0	XSLT 2.0	XSLT 3.0	XQuery	C++	C#	Java	BUILT-IN
Parsen und Serialisieren von Strings 1382 2								●
Dynamische Node-Namen 1309		●	●	●	●	●	●	●
Digitale XML- und XBRL-Signaturen 143								●
Datenbank-Bulk-Einfügungen 329								●
SQL SELECT für Datenbanken ohne Input-Parameter 368					●	●	●	●
SQL SELECT für Datenbanken mit Input-Parametern 368								●
Gespeicherte Prozeduren für Datenbanken 402								●
Behandlung von Datenbankausnahmen 429 3					●	●	●	●
Datenbankablaufverfolgung und Fehlerprotokollierung 443								●
Schnelle (Nicht-Standard) Konvertierung von EDI in XML 475								●
Generierung von SOAP-Webservices 1255						●	●	
Generierung von MapForce Server-Ausführungsdateien 1442								●

© 2016-2022 Altova GmbH

Funktion	XSLT 1.0	XSLT 2.0	XSLT 3.0	XQuery	C++	C#	Java	BUILT-IN
Bereitstellung von Mappings auf FlowForce Server[1445]								●
Lesen von Daten aus Binärdateien[860]								●
Schreiben von Daten in Binärdateien[860]								●

Fußnoten:

1. Für XSLT 2.0, XSLT 3.0 und XQuery wird die Funktion `fn:collection` verwendet. In der Implementierung im Altova XSLT 2.0-, XSLT 3.0- und XQuery-Prozessor werden Platzhalter aufgelöst. Andere Prozessoren verhalten sich eventuell anders.
2. Das Parsen und die Serialisierung wird für JSON zusätzlich in Java und C# unterstützt.
3. Die Behandlung von Datenbankausnahmen ist möglich, wenn die Mapping-Sprache vom aktuell verwendeten Datenbanktreiber (siehe vorherige Tabelle) unterstützt wird.

19.2 Informationen zu den Prozessoren

Dieser Abschnitt enthält Informationen über implementierungsspezifische Funktionen des Altova XML Validators, des Altova XSLT 1.0-Prozessors, des Altova XSLT 2.0-Prozessors und des Altova XQuery-Prozessors.

19.2.1 Informationen zum XSLT- und XQuery-Prozessor

Der XSLT- und der XQuery-Prozessor von MapForce hält sich genau an die W3C-Spezifikationen und ist daher strenger als die früheren Altova-Prozessoren, wie z.B. die in frühere Versionen von XMLSpy integrierten . Infolgedessen werden auch leichte Fehler, die von früheren Prozessoren ignoriert wurden, von MapForce als Fehler gekennzeichnet.

Zum Beispiel:

- Wenn das Ergebnis eines Pfad-Operators sowohl Nodes als auch Nicht-Nodes enthält, wird ein Typfehler (`err:XPTY0018`) ausgegeben.
- Wenn `E1` in einem Pfadausdruck `E1/E2` nicht zu einer Node-Sequenz ausgewertet wird, wird ein Typfehler (`err:XPTY0019`) ausgegeben.

Ändern Sie bei Auftreten eines solchen Fehlers je nach Bedarf, entweder das XSLT/XQuery-Dokument oder das Instanzdokument.

In diesem Abschnitt sind implementierungsspezifische Funktionalitäten der Prozessoren geordnet nach Spezifikation beschrieben:

- XSLT 1.0 1988
- XSLT 2.0 1989
- XQuery 1.0 1990

19.2.1.1 XSLT 1.0

Der XSLT 1.0-Prozessor von MapForce entspricht der XSLT 1.0 Recommendation vom 16. November 1999 und der XPath 1.0 Recommendation vom 16. November 1999 des World Wide Web Consortium (W3C). Beachten Sie die folgenden Informationen zur Implementierung.

Anmerkungen zur Implementierung

Wenn das `method`-Attribut von `xsl:output` auf HTML gesetzt ist oder wenn standardmäßig die HTML-Ausgabe ausgewählt ist, werden Sonderzeichen in der XML- oder XSLT-Datei als HTML-Zeichenreferenzen in das HTML-Ausgabedokument eingefügt. So wird z.B. das Zeichen U+00A0 (die hexadezimale Zeichenreferenz für ein geschütztes Leerzeichen) entweder als Zeichenreferenz (` ` or ` `) oder als Entity-Referenz ` ` in den HTML-Code eingefügt.

© 2016-2022 Altova GmbH

19.2.1.2 XSLT 2.0

In diesem Abschnitt:

- Prozessorkonformität 1989
- Rückwärtskompatibilität 1989
- Namespaces 1989
- Schema-Fähigkeit 1990
- Implementierungsspezifisches Verhalten 1990

Standardkonformität

Der XSLT 2.0-Prozessor von MapForce entspricht der XSLT 2.0 Recommendation vom 23. Jänner 2007 und der XPath 2.0 Recommendation vom 14. Dezember 2010 des World Wide Web Consortium (W3C).

Rückwärtskompatibilität

Der XSLT 2.0-Prozessor ist rückwärtskompatibel. Die Rückwärtskompatibilität des XSLT 2.0.-Prozessors kommt nur dann zum Einsatz, wenn Sie den XSLT 2.0-Prozessor zur Verarbeitung eines XSLT 1.0 Stylesheets verwenden. Beachten Sie, dass sich das Ergebnis des XSLT 1.0-Prozessors und des rückwärtskompatiblen XSLT 2.0.Prozessors unter Umständen unterscheiden kann.

Namespaces

In Ihrem XSLT 2.0 Stylesheet sollten die folgenden Namespaces deklariert sein, damit Sie die in XSLT 2.0 verfügbaren Typ-Konstruktoren und Funktionen verwenden können. Normalerweise werden die unten aufgelisteten Präfixe verwendet; bei Bedarf können Sie auch andere Präfixe verwenden.

Namespace Name	Präfix	Namespace URI
XML Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
XPath 2.0-Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`

Normalerweise werden diese Namespaces im Element `xsl:stylesheet` oder `xsl:transform` deklariert, wie unten gezeigt:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   ...
</xsl:stylesheet>
```

Beachten Sie die folgenden Punkte:

- Der XSLT 2.0-Prozessor verwendet als **Standard-Funktions-Namespace** den Namespace für XPath 2.0- und XQuery 1.0-Funktionen (siehe Tabelle oben). Sie können daher XPath 2.0- und XSLT 2.0-Funktionen in Ihrem Stylesheet ohne Präfix verwenden. Wenn Sie den Namespace für XPath 2.0-Funktionen in Ihrem Stylesheet mit einem Präfix deklarieren, können Sie zusätzlich dazu das in der Deklaration zugewiesene Präfix verwenden.

- Bei Verwendung von Typ-Konstruktoren und Typen aus dem XML Schema-Namespace, muss bei Aufruf des Typ-Konstruktors (z.B. `xs:date`) das in der jeweiligen Namespace-Deklaration verwendeten Präfix verwendet werden.
- Einige XPath 2.0-Funktionen haben denselben Namen wie XML Schema-Datentypen. So gibt es z.B. für die XPath-Funktionen `fn:string` und `fn:boolean` XML-Schema-Datentypen mit demselben lokalen Namen: `xs:string` und `xs:boolean`. Wenn Sie daher den XPath-Ausdruck `string('Hello')` verwenden, wird der Ausdruck als `fn:string('Hello')` ausgewertet und nicht als `xs:string('Hello')`.

Schemafähigkeit

Der XSLT 2.0-Prozessor ist schemafähig. Sie können daher benutzerdefinierte Schematypen und die `xsl:validate`-Anweisung verwenden.

Implementierungsspezifisches Verhalten

Im Folgenden finden Sie eine Beschreibung, wie der XSLT 2.0-Prozessor implementierungsspezifische Aspekte des Verhaltens bestimmter XSLT 2.0-Funktionen behandelt.

`xsl:result-document`
Zusätzlich werden die folgenden Kodierungen unterstützt: `x-base16tobinary` und `x-base64tobinary`.

`function-available`
Die Funktion überprüft, ob in-scope-Funktionen (XSLT, XPath und Erweiterungsfunktionen) verfügbar sind.

`unparsed-text`
Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll. Zusätzlich werden die folgenden (Altova-spezifischen) Kodierungen unterstützt: `binarytobase16` und `binarytobase64`.

`unparsed-text-available`
Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll. Zusätzlich werden die folgenden (Altova-spezifischen) Kodierungen unterstützt: `binarytobase16` und `binarytobase64`.

Anmerkung: Die folgenden Kodierungswerte, die in früheren Versionen von AltovaXML, dem Vorgängerprodukt von RaptorXML, verwendet wurden, werden nun nicht mehr verwendet:`base16tobinary`, `base64tobinary`, `binarytobase16` und `binarytobase64`.

19.2.1.3 XQuery 1.0

In diesem Abschnitt:

- Standardkonformität 1991
- Schema-Fähigkeit 1991
- Kodierung 1991
- Namespaces 1989
- XML-Quelle und Validierung 1992
- Statische und dynamische Typüberprüfung 1992
- Bibliotheksmodule 1992

 © 2016-2022 Altova GmbH

- Externe Funktionen 1993
- Collations 1993
- Präzision von numerischen Daten 1993
- Unterstützung für XQuery-Anweisungen 1993

Standardkonformität

Der XQuery 1.0-Prozessor von MapForce entspricht der XQuery 1.0 Recommendation vom 14. Dezember 2010 des W3C. Der Query-Standard stellt bei vielen Funktionen frei, wie viele diese zu implementieren sind. Im Folgenden finden Sie eine Liste, wie der Altova XQuery 1.0-Prozessor diese Funktionen implementiert.

Schemafähigkeit

Der Altova XQuery 1.0-Prozessor ist **schemafähig.**

Kodierung

Die UTF-8 und die UTF-16 Zeichen-Kodierungen werden unterstützt.

Namespaces

Die folgenden Namespace-URIs und die damit verknüpften Bindings sind vordefiniert.

Namespace-Name	Präfix	Namespace URI
XML Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
Schema-Instanz	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
Vordefinierte Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`
Lokale Funktionen	`local:`	`http://www.w3.org/2005/xquery-local-functions`

Beachten Sie bitte die folgenden Punkte:

- Der Altova XQuery 1.0-Prozessor ist so konfiguriert, dass die oben aufgelisteten Präfixe an die entsprechenden Namespaces gebunden sind.
- Da der oben angeführte Namespace für vordefinierte Funktionen der Standard-Funktions-Namespace in XQuery ist, muss beim Aufruf von vordefinierten Funktionen das Präfix `fn:` nicht verwendet werden (`string("Hello")` ruft z.B. die Funktion `fn:string` auf). Das Präfix `fn:` kann jedoch verwendet werden, um eine vordefinierte Funktion aufzurufen, ohne die Namespace im Abfrage-Prolog deklarieren zu müssen (z.B.: `fn:string("Hello")`).
- Sie können den Standard-Funktions-Namespace durch Deklarierung des `default function namespace`-Ausdrucks im Abfrageprolog ändern.
- Bei Verwendung von Typen aus dem XML Schema-Namespace kann das Präfix `xs:` verwendet werden, ohne dass Sie den Namespace explizit deklarieren müssen und dieses Präfix im Abfrageprolog daran binden müssen. (Beispiele: `xs:date` und `xs:yearMonthDuration`.) Wenn Sie ein anderes Präfix verwenden möchten, muss dieses im Abfrageprolog für die Namespaces explizit deklariert werden. (Beispiel `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)

- Beachten Sie, dass die Datentypen `untypedAtomic`, `dayTimeDuration` und `yearMonthDuration` mit den Candidate Recommendations vom 23 January 2007 aus dem XPath Datentypen-Namespace in den XML-Schema Namespace verschoben wurden, d.h. `xs:yearMonthDuration`.

Wenn Namespaces für Funktionen, Typ-Konstruktoren, Node Tests usw. falsch zugewiesen wurden, wird ein Fehler ausgegeben. Beachten Sie jedoch, dass einige Funktionen denselben Namen wie Schema-Datentypen haben, z.B. `fn:string` und `fn:boolean`. (Sowohl `xs:string` als auch `xs:boolean` ist definiert.) Das Namespace-Präfix legt fest, ob die Funktion oder der Typ-Konstruktor verwendet wird.

XML-Quelldokument und Validierung

XML-Dokumente, die bei der Ausführung eines XQuery-Dokuments mit dem Altova XQuery 1.0-Prozessor verwendet werden, müssen wohlgeformt sein. Sie müssen jedoch nicht gemäß einem XML-Schema gültig sein. Wenn die Datei nicht gültig ist, wird die ungültige Datei ohne Schemainformationen geladen. Wenn die XML-Datei mit einem externen Schema verknüpft ist und gemäß diesem Schema gültig ist, werden für die XML-Daten nachträglich Validierungsinformationen generiert und für die Auswertung der Abfrage verwendet.

Statische und dynamische Typ-Überprüfung

In der statischen Analysephase werden Aspekte der Abfrage überprüft wie z.B. die Syntax, ob externe Referenzen (z.B. für Module) vorhanden sind, ob aufgerufene Funktionen und Variablen definiert sind, usw. Wenn in dieser Phase ein Fehler gefunden wird, wird eine Meldung ausgegeben und die Ausführung wird gestoppt.

Die dynamische Typ-Überprüfung wird in Laufzeit durchgeführt, während die Abfrage ausgeführt wird. Wenn ein Typ mit den Anforderungen einer Operation nicht kompatibel ist, wird ein Fehler ausgegeben. So gibt z.B. der Ausdruck `xs:string("1") + 1` einen Fehler zurück, weil die Operation "Addition" nicht an einem Operanden vom Typ `xs:string` ausgeführt werden kann.

Bibliotheksmodule

Bibliotheksmodule dienen zum Speichern von Funktionen und Variablen, damit diese wiederverwendet werden können. Der Altova XQuery 1.0-Prozessor unterstützt Module, die in einer **einzigen externen XQuery-Datei** gespeichert sind. Eine solche Moduldatei muss im Prolog eine `module`-Deklaration enthalten, in der ein Target Namespace zugewiesen wird. Hier ein Beispielmodul:

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns:webaddress() { "http://www.altova.com" };
```

Alle im Modul deklarierten Funktionen und Variablen gehören zu dem mit dem Modul verknüpften Namespace. Das Modul wird durch Import in eine XQuery-Datei mittels der `import module`-Anweisung im Abfrageprolog verwendet. Die `import module`-Anweisung importiert nur Funktionen und Variablen, die direkt in der Bibliotheksmodul-Datei deklariert sind:

```
import module namespace modlib = "urn:module-library" at "modulefilename.xq";
if    ($modlib:company = "Altova")
then   modlib:webaddress()
else   error("No match found.")
```


© 2016-2022 Altova GmbH

Externe Funktionen

Externe Funktionen, d.h. diejenigen Funktionen, die das Schlüsselwort `external` verwenden, werden nicht unterstützt:

```
declare function hoo($param as xs:integer) as xs:string external;
```

Collations

Die Standard-Collation ist die Unicode Codepoint Collation, die Strings auf Basis ihrer Unicode-Codepunkte vergleicht. Andere unterstützte Collations sind die hier[1993] aufgelisteten ICU-Collations. Um eine bestimmte Collation zu verwenden, geben Sie ihre in der Liste der unterstützten Collations[1993] angeführte URI an. String-Vergleiche, wie die Funktionen `fn:max` und `fn:min` werden anhand der angegebenen Collation durchgeführt. Wenn die Collation-Option nicht definiert ist, wird die Standard-Unicode Codepoint Collation verwendet.

Präzision von numerischen Typen

- Der Datentyp `xs:integer` hat eine beliebige Präzision, d.h. er kann beliebig viele Stellen haben.
- Der Datentyp `xs:decimal` kann nach dem Dezimalpunkt maximal 20 Stellen haben.
- Die Datentypen `xs:float` und `xs:double` sind auf 15 Stellen beschränkt.

Unterstützung für XQuery-Anweisungen

Die `Pragma` Anweisung wird nicht unterstützt. Gegebenenfalls wird sie ignoriert und der Fallback-Ausdruck wird evaluiert.

19.2.2 XSLT- und XPath/XQuery-Funktionen

Dieser Abschnitt enthält eine Liste von Altova-Erweiterungsfunktionen und anderen Erweiterungsfunktionen, die in XPath und/oder XQuery-Ausdrücken verwendet werden können. Altova-Erweiterungsfunktionen können mit dem XSLT- und XQuery-Prozessor von Altova verwendet werden und bieten zusätzliche Funktionalitäten zu den in den W3C-Standards definierten Funktionsbibliotheken.

Allgemeine Punkte

Beachten Sie bitte die folgenden allgemeinen Punkte:

- Funktionen aus den in den W3C-Spezifikationen definierten core-Funktionsbibliotheken können ohne Präfix aufgerufen werden, da der XSLT- und XQuery-Prozessor Funktionen, die kein Präfix haben, als Funktionen des Standard-Funktions-Namespace liest. Der Standard-Funktions-Namespace ist derjenige, der in den XPath/XQuery-Funktionsspezifikationen definiert ist `http://www.w3.org/2005/xpath-functions`. Wenn dieser Namespace in einem XSLT- oder XQuery-Dokument explizit deklariert ist, kann das in der Namespace-Deklaration definierte Präfix optional auch in Funktionsnamen verwendet werden.
- Grundsätzlich gilt: Wenn bei einer Funktion eine Sequenz von einem Datenelement als Argument erwartet wird und eine Sequenz von mehr als einem Datenelement gesendet wird, wird ein Fehler zurückgegeben.

- Alle String-Vergleiche werden unter Verwendung der Unicode Codepoint Collation ausgeführt.
- Ergebnisse, bei denen es sich um QNames handelt, werden in der Form `[prefix:]localname` serialisiert.

Präzision von xs:decimal
Die Präzision bezieht sich auf die Anzahl der Stellen in einer Zahl. Laut Spezifikation sind mindestens 18 Stellen erforderlich. Bei Divisionen, bei denen ein Ergebnis vom Typ `xs:decimal` erzeugt wird, beträgt die Präzision 19 Kommastellen ohne Runden.

Implizite Zeitzone
Beim Vergleich zweier `date`, `time`, oder `dateTime` Werte müssen die Zeitzonen der verglichenen Werte bekannt sein. Wenn die Zeitzone in einem solchen Wert nicht explizit angegeben ist, wird die implizite Zeitzone verwendet. Als implizite Zeitzone wird die der Systemuhr verwendet. Der Wert kann mit Hilfe der Funktion `implicit-timezone()` überprüft werden.

Collations
Die Standard-Collation ist die Unicode Codepoint Collation, die Strings auf Basis ihrer Unicode-Codepunkte vergleicht. Es wird der Unicode Collation Algorithmus verwendet. Andere unterstützte Collations sind die unten aufgelisteten ICU-Collations. Um eine bestimmte dieser Collations zu verwenden, geben Sie ihre in der nachstehenden Tabelle angegebene URI an. String-Vergleiche, wie die Funktionen `max` und `min` werden anhand der angegebenen Collation durchgeführt. Wenn die Collation-Option nicht definiert ist, wird die Standard-Unicode Codepoint Collation verwendet.

Sprache	URIs
`da`: Dänisch	`da_DK`
`de`: Deutsch	`de_AT, de_BE, de_CH, de_DE, de_LI, de_LU`
`en`: Englisch	`en_AS, en_AU, en_BB, en_BE, en_BM, en_BW, en_BZ, en_CA, en_GB, en_GU, en_HK, en_IE, en_IN, en_JM, en_MH, en_MP, en_MT, en_MU, en_NA, en_NZ, en_PH, en_PK, en_SG, en_TT, en_UM, en_US, en_VI, en_ZA, en_ZW`
`es`: Spanisch	`es_419, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GQ, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE`
`fr`: Französisch	`fr_BE, fr_BF, fr_BI, fr_BJ, fr_BL, fr_CA, fr_CD, fr_CF, fr_CG, fr_CH, fr_CI, fr_CM, fr_DJ, fr_FR, fr_GA, fr_GN, fr_GP, fr_GQ, fr_KM, fr_LU, fr_MC, fr_MF, fr_MG, fr_ML, fr_MQ, fr_NE, fr_RE, fr_RW, fr_SN, fr_TD, fr_TG`
`it`: Italienisch	`it_CH, it_IT`
`ja`: Japanisch	`ja_JP`
`nb`: Norwegisch (Bokmal)	`nb_NO`
`nl`: Holländisch	`nl_AW, nl_BE, nl_NL`

© 2016-2022 Altova GmbH

`nn`: Norwegisch (Nynorsk)	`nn_NO`
`pt`: Portugiesisch	`pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST`
`ru`: Russisch	`ru_MD, ru_RU, ru_UA`
`sv`: Schwedisch	`sv_FI, sv_SE`

Namespace-Achse
Die Namespace-Achse wird in XPath 2.0 nicht mehr verwendet, wird aber weiterhin unterstützt. Um Namespace-Informationen mit XPath 2.0-Mechanismen aufzurufen, verwenden Sie die Funktionen `in-scope-prefixes()`, `namespace-uri()` und `namespace-uri-for-prefix()`.

19.2.2.1 Altova-Erweiterungsfunktionen

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

Die in der "XPath/XQuery Functions"-Spezifikation des W3C definierten Funktionen können (i) in einem XSLT-Kontext in XPath-Ausdrücken und (ii) in einem XQuery-Dokument in XQuery-Ausdrücken verwendet werden. In dieser Dokumentation sind die Funktionen, die im Zusammenhang mit XPath in XSLT verwendet werden können, mit einem **XP**-Symbol und Funktionen, die im Zusammenhang mit XQuery verwendet werden können, mit einem **XQ**-Symbol markiert; sie fungieren als XQuery-Funktionen. In den XSLT-Spezifikationen des W3C (nicht in den "XPath/XQuery Functions"-Spezifikationen) sind außerdem Funktionen definiert, die in XSLT-Dokumenten in XPath-Ausdrücken verwendet werden können. Diese Funktionen sind mit dem Symbol **XSLT** gekennzeichnet und werden als XSLT-Funktionen bezeichnet. In welcher XPath/XQuery- und XSLT-Version eine Funktion verwendet werden kann, wird in der Beschreibung der Funktion (*siehe Symbole unten)* angegeben. Funktionen aus der XPath/XQuery- und XSLT-Funktionsbibliothek werden ohne Präfix aufgelistet. Erweiterungsfunktionen aus anderen Bibliotheken wie z.B. Altova-Erweiterungsfunktionen werden mit einem Präfix angegeben.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XP1** **XP2** **XP3.1**
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XSLT1** **XSLT2** **XSLT3**
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	**XQ1** **XQ3.1**

XSLT-Funktionen[1996]
XSLT-Funktionen können in XPath-Ausdrücken nur im XSLT-Kontext verwendet werden (ähnlich wie die XSLT 2.0-Funktionen `current-group()` oder `key()`). Diese Funktionen sind nicht für Nicht-XSLT-Kontext gedacht

und funktionieren in einem solchen Kontext (z.B. in einem XQuery-Kontext) nicht. Beachten Sie, dass XSLT-Funktionen für XBRL nur mit Altova Produkteditionen verwendet werden können, die XBRL unterstützen.

XPath/XQuery-Funktionen

XPath/XQuery-Funktionen können sowohl in XPath-Ausdrücken im XSLT-Kontext als auch in XQuery-Ausdrücken verwendet werden.

- Datum/Uhrzeit 1999
- Standort 2017
- Bildbezogene 2029
- Numerisch 2034
- Sequenz 2056
- String 2065
- Verschiedenes 2071

Diagrammfunktionen (nur Enterprise- und Server-Editionen)

Altova-Erweiterungsfunktionen für Diagramme 2074 werden nur in der Enterprise und Server Edition von Altova-Produkten unterstützt und ermöglichen die Generierung von Diagrammen anhand von XML-Daten.

19.2.2.1.1 XSLT-Funktionen

XSLT-Erweiterungsfunktionen können in XPath-Ausdrücken in einem XSLT-Kontext verwendet werden. In einem Nicht-XSLT-Kontext (z.B. in einem XQuery-Kontext) funktionieren sie nicht.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

Allgemeine Funktionen

▼ distinct-nodes [altova:]

`altova:distinct-nodes(`*`node()*`***`)`** **`as node()*`** XSLT1 XSLT2 XSLT3

Erhält eine Gruppe von einem oder mehreren Nodes als Input und gibt dieselbe Gruppe ohne Nodes mit doppelt vorhandenen Werten zurück. Der Vergleich wird mittels der XPath/XQuery-Funktion `fn:deep-`

© 2016-2022 Altova GmbH

`equal` durchgeführt.

Beispiele

- **`altova:distinct-nodes`**`(country)` gibt alle Child `country` Nodes ohne diejenigen mit doppelt vorhandenen Werten zurück.

evaluate [altova:]

`altova:evaluate(XPathExpression` *`as xs:string`***`[, ValueOf$p1, ... ValueOf$pN])`** XSLT1 XSLT2 XSLT3

Erhält einen XPath-Ausdruck als obligatorisches Argument, der als String übergeben wird, und gibt das Resultat des ausgewerteten Ausdrucks zurück. Beispiel: **`altova:evaluate('//Name[1]')`** gibt den Inhalt des ersten `Name` Elements im Dokument zurück. Beachten Sie, dass der Ausdruck `//Name[1]` durch Einschließen in einfache Anführungszeichen als String übergeben wird.

Die Funktion `altova:evaluate` kann zusätzliche (optionale) Argumente erhalten. Diese Argumente sind die Werte der einzelnen im Geltungsbereich befindlichen Variablen und haben die Namen `p1`, `p2`, `p3`... `pN`. Beachten Sie zur Verwendung die folgenden Punkte: (i) Die Variablennamen müssen die Form `pX` haben, wobei `X` eine Ganzzahl ist; (ii) die Argumente der Funktion `altova:evaluate` (*siehe Signatur oben*) liefern vom zweiten Argument an die Werte der Variablen, wobei die Reihenfolge der Argumente der numerisch geordneten Variablensequenz entspricht: `p1` bis `pN`. Das zweite Argument wird der Wert der Variablen `p1`, das dritte Argument der der Variablen `p2`, usw.; (iii) Die Werte der Variablen müssen vom Typ `item*` sein

Beispiel

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 'hi')" />
```

gibt aus `"hi 20 10"`

Beachten Sie im obigen Beispiel folgende Punkte:

- Das zweite Argument des Ausdrucks `altova:evaluate` ist der der Variablen `$p1` zugewiesene Wert, das dritte Argument ist das der Variablen `$p2` zugewiesene usw.
- Beachten Sie, dass das vierte Argument der Funktion ein String-Wert ist. Als String-Wert wird dieser innerhalb von Anführungszeichen gesetzt.
- Das `select` Attribut des Elements `xs:variable` liefert den XPath-Ausdruck. Da dieser Ausdruck den Typ `xs:string`, haben muss, wird er in einfache Anführungszeichen gesetzt.

Weitere Beispiele für die Verwendung der Variablen

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
  ```
  Gibt den Wert des ersten `Name` *Elements zurück.*

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, '//Name[1]')" />
  ```
  Gibt "`//Name[1]`*" aus*

Die `altova:evaluate()` Erweiterungsfunktion ist in Situationen nützlich, in denen ein XPath-Ausdruck im XSLT-Stylesheet einen oder mehrere Teile enthält, die dynamisch ausgewertet werden müssen.

Angenommen ein Benutzer gibt seinen Request für das Sortierkriterium ein und das Sortierkriterium ist im Attribut `UserReq/@sortkey` gespeichert. Im Stylesheet könnten Sie den folgenden Ausdruck haben: `<xsl:sort select="altova:evaluate(../UserReq/@sortkey)" order="ascending"/>` Die `altova:evaluate()` Funktion liest das `sortkey` Attribut des `UserReq` Child-Elements des Parent des Kontext-Node. Angenommen der Wert des `sortkey` Attributs ist `Price`, dann wird von der `altova:evaluate()` Funktion `Price` zurückgegeben und wird zum Wert des `select` Attributs: `<xsl:sort select="Price" order="ascending"/>`. Wenn diese `sort` Anweisung im Kontext eines Elements namens `Order` vorkommt, dann werden die `Order` Elemente nach den Werten Ihrer `Price` Children sortiert. Alternativ dazu, wenn der Wert von `@sortkey` z.B. `Date` ist, werden die `Order` Elemente nach den Werten ihrer `Date` Children sortiert. Das Sortierkriterium für `Order` wird also zur Laufzeit aus dem `sortkey` Attribut ausgewählt. Diese hätte man mit einem Ausdruck wie dem folgenden nicht bewerkstelligen können: `<xsl:sort select="../UserReq/@sortkey" order="ascending"/>`. Im oben gezeigten Beispiel wäre das Sortierkriterium das `sortkey` Attribut selbst, nicht `Price` oder `Date` (oder jeder beliebige andere Inhalt von `sortkey`)

Hinweis:
Der statische Kontext enthält Namespaces, Typen und Funktionen - aber keine Variablen - aus der aufrufenden Umgebung. Die Basis-URI und der Standard-Namespace werden vererbt.

Weitere Beispiele

- Statische Variablen:
```
<xsl:value-of select="$i3, $i2, $i1" />
```
Gibt die Werte von drei Variablen aus.

- Dynamischer XPath-Ausdruck mit dynamischen Variablen:
```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />
```
Gibt "30 20 10" aus

- Dynamischer XPath-Ausdruck ohne dynamische Variable:
```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath)" />
```
Gibt einen Fehler aus.: Es wurde keine Variable für `$p3` *definiert.*

encode-for-rtf [altova:]

altova:encode-for-rtf(input *als xs:string*, **preserveallwhitespace** *als xs:boolean*, **preservenewlines** *als xs:boolean*) **als xs:string** XSLT2 XSLT3
Konvertiert den Input-String in Code für RTF. Whitespaces und neue Zeilen werden gemäß dem für die entsprechenden Parameter definierten Booleschen Wert beibehalten.

[Nach oben 1996]

XBRL-Funktionen

Altova XBRL-Funktionen können nur mit Editionen von Altova-Produkten verwendet werden, die XBRL unterstützen.

© 2016-2022 Altova GmbH

▼ xbrl-footnotes [altova:]

altova:xbrl-footnotes(*node()***)** als **node()*** XSLT2 XSLT3
Erhält einen Node als Input-Argument und gibt die durch den Input-Node referenzierte Gruppe der XBRL-Fußnoten-Nodes zurück.

▼ xbrl-labels [altova:]

altova:xbrl-labels(*xs:QName, xs:string***)** als **node()*** XSLT2 XSLT3
Erhält zwei Input-Argumente: einen Node-Namen und den Pfad der Taxonomiedatei, die den Node enthält. Die Funktion gibt die XBRL Labels zurück, die mit dem Input-Node verknüpft sind.

[Nach oben 1996 **]**

19.2.2.1.2 XPath/XQuery-Funktionen: Datum und Uhrzeit

Die Datums- und Uhrzeit-Erweiterungsfunktionen von Altova können im Zusammenhang mit XPath- und XQuery-Ausdrücken verwendet werden und bieten zusätzliche Funktionalitäten für die Verarbeitung von Daten, die in Form von XML-Schema-Datums- und Uhrzeit-Datentypen zur Verfügung stehen. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0** - und dem **XQuery 3.0** -Prozessor von Altova verwendet werden. Sie stehen in verschiedenen XPath/XQuery-Kontexten zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

▼ Nach Funktionalität gruppiert

- Hinzufügen einer Zeitdauer zu xs:dateTime und Rückgabe von xs:dateTime 2001
- Hinzufügen einer Zeitdauer zu xs:date und Rückgabe von xs:date 2002
- Hinzufügen einer Zeitdauer zu xs:time und Rückgabe von xs:time 2004
- Formatieren und Abrufen einer Zeitdauer 2003
- Entfernen der Zeitzone aus Funktionen, die das aktuelle Datum/die aktuelle Uhrzeit generieren 2005
- Rückgabe von Tagen, Stunden, Minuten und Sekunden anhand einer Zeitdauer 2006
- Rückgabe des Wochentags anhand des Datums als Ganzzahl 2008

- Rückgabe eines Wochentags als Ganzzahl anhand eines Datums 2008
- Erstellen des Datums, der Uhrzeit oder des Zeitdauertyps anhand der lexikalischen Komponenten der einzelnen Typen 2011
- Konstruieren des Typs "Datum", "Datum und Uhrzeit" oder "Uhrzeit" anhand eines String Input 2012
- Funktionen zur Berechnung des Alters 2014
- Epochen-Zeit (Unix-Zeit)-Funktionen 2015

▼ in alphabetischer Reihenfolge

altova:add-days-to-date 2002
altova:add-days-to-dateTime 2001
altova:add-hours-to-dateTime 2001
altova:add-hours-to-time 2004
altova:add-minutes-to-dateTime 2001
altova:add-minutes-to-time 2004
altova:add-months-to-date 2002
altova:add-months-to-dateTime 2001
altova:add-seconds-to-dateTime 2001
altova:add-seconds-to-time 2004
altova:add-years-to-date 2002
altova:add-years-to-dateTime 2001
altova:age 2014
altova:age-details 2014
altova:build-date 2011
altova:build-duration 2011
altova:build-time 2011
altova:current-dateTime-no-TZ 2005
altova:current-date-no-TZ 2005
altova:current-time-no-TZ 2005
altova:date-no-TZ 2005
altova:dateTime-from-epoch 2015
altova:dateTime-from-epoch-no-TZ 2015
altova:dateTime-no-TZ 2005
altova:days-in-month 2006
altova:epoch-from-dateTime 2015
altova:hours-from-dateTimeDuration-accumulated 2006
altova:minutes-from-dateTimeDuration-accumulated 2006
altova:seconds-from-dateTimeDuration-accumulated 2006
altova:format-duration 2003
altova:parse-date 2012
altova:parse-dateTime 2012
altova:parse-duration 2003
altova:parse-time 2012
altova:time-no-TZ 2005
altova:weekday-from-date 2008
altova:weekday-from-dateTime 2008
altova:weeknumber-from-date 2009
altova:weeknumber-from-dateTime 2009

[**Nach oben** 1999]

© 2016-2022 Altova GmbH

Hinzufügen einer Zeitdauer zu xs:dateTime XP3.1 XQ3.1

Mit diesen Funktionen werden Zeitdauerwerte zu **xs:dateTime** hinzugefügt, bevor **xs:dateTime** zurückgegeben wird. Der Typ `xs:dateTime` hat das Format `JJJJ-MM-TTZhh:mm:ss.sss`. Es handelt sich hierbei um eine Verkettung des `xs:date` und `xs:time` Formats, getrennt durch den Buchstaben `Z`. Ein Zeitzonensuffix (wie z.B. **`+01:00`**) ist optional.

add-years-to-dateTime [altova:]

altova:add-years-to-dateTime(DateTime *als xs:dateTime*, **Years** *als xs:integer*) **als xs:dateTime** XP3.1 XQ3.1

Fügt eine Zeitdauer in Jahren zu einem `xs:dateTime` Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Jahre, die zu dem im ersten Parameter angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **`altova:add-years-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), 10)` gibt `2024-01-15T14:00:00` zurück
- **`altova:add-years-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), -4)` gibt `2010-01-15T14:00:00` zurück

add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(DateTime *als xs:dateTime*, **Months** *als xs:integer*) als **xs:dateTime** XP3.1 XQ3.1

Fügt eine Zeitdauer in Monaten zu einem `xs:dateTime` Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Monate, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **`altova:add-months-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), 10)` gibt `2014-11-15T14:00:00` zurück
- **`altova:add-months-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), -2)` gibt `2013-11-15T14:00:00` zurück

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(DateTime *als xs:dateTime*, **Days** *als xs:integer*) als **xs:dateTime** XP3.1 XQ3.1

Fügt eine Zeitdauer in Tagen zu einem `xs:dateTime` Wert (*siehe Beispiel unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Tage, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **`altova:add-days-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), 10)` gibt `2014-01-25T14:00:00` zurück
- **`altova:add-days-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), -8)` gibt `2014-01-07T14:00:00` zurück

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(DateTime *als xs:dateTime*, **Hours** *als xs:integer*) als

xs:dateTime XP3.1 XQ3.1
Fügt eine Zeitdauer in Stunden zu einem `xs:dateTime` Wert (*siehe Beispiel unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Stunden, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) gibt 2014-01-15T23:00:00 zurück
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) gibt 2014-01-15T05:00:00 zurück

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(DateTime *als xs:dateTime*, **Minutes** *als xs:integer*) als **xs:dateTime** XP3.1 XQ3.1
Fügt eine Zeitdauer in Minuten zu einem `xs:dateTime` Wert (*siehe Beispiele unten)* hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Minuten, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) gibt 2014-01-15T14:55:00 zurück
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) gibt 2014-01-15T14:05:00 zurück

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(DateTime *als xs:dateTime*, **Seconds** *als xs:integer*) als **xs:dateTime** XP3.1 XQ3.1
Fügt eine Zeitdauer in Sekunden zu einem `xs:dateTime` Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Sekunden, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) gibt 2014-01-15T14:00:30 zurück
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) gibt 2014-01-15T14:00:05 zurück

[Nach oben 1999]

Hinzufügen einer Zeitdauer zu xs:date XP3.1 XQ3.1

Mit diesen Funktionen werden Zeitdauerwerte zu **xs:date** hinzugefügt, bevor **xs:date** zurückgegeben wird. Der Typ `xs:date` hat das Format `JJJJ-MM-TT`.

add-years-to-date [altova:]

© 2016-2022 Altova GmbH

altova:add-years-to-date(Date *als xs:date*, **Years** *als xs:integer*) als **xs:date** XP3.1 XQ3.1
Fügt eine Zeitdauer in Jahren zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Jahre, die zu dem im ersten Argument angegebenen `xs:date` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:date`.

Beispiele

- **altova:add-years-to-date**(xs:date("2014-01-15"), 10) gibt 2024-01-15 zurück
- **altova:add-years-to-date**(xs:date("2014-01-15"), -4) gibt 2010-01-15 zurück

add-months-to-date [altova:]

altova:add-months-to-date(Date *als xs:date*, **Months** *als xs:integer*) als **xs:date** XP3.1 XQ3.1
Fügt eine Zeitdauer in Monaten zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Monate, die zu dem im ersten Argument angegebenen `xs:date` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:date`.

Beispiele

- **altova:add-months-to-date**(xs:date("2014-01-15"), 10) gibt 2014-11-15 zurück
- **altova:add-months-to-date**(xs:date("2014-01-15"), -2) gibt 2013-11-15 zurück

add-days-to-date [altova:]

altova:add-days-to-date(Date *als xs:date*, **Days** *als xs:integer*) als **xs:date** XP3.1 XQ3.1
Fügt eine Zeitdauer in Tagen zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Tage, die zu dem im ersten Argument angegebenen `xs:date` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:date`.

Beispiele

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10) gibt 2014-01-25 zurück
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) gibt 2014-01-07 zurück

[Nach oben 1999]

Formatieren und Abrufen einer Zeitdauer XP3.1 XQ3.1

Mit diesen Funktionen wird ein Input **xs:duration**- oder **xs:string**-Wert geparst und ein **xs:string**- bzw. **xs:duration**-Wert zurückgegeben.

format-duration [altova:]

altova:format-duration(Duration *als xs:duration*, **Picture** *als xs:string*) als **xs:string** XP3.1 XQ3.1
Formatiert eine Zeitdauer, die als erstes Argument bereitgestellt wird, gemäß einem Muster-String, der als zweites Argument bereitgestellt wird. Die Ausgabe ist ein Textstring, der dem Muster-String entsprechend formatiert ist.

Beispiele

- **altova:format-duration**(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") gibt "Days:02 Hours:02 Minutes:53

Seconds:11 Fractions:7" gibt
- **altova:format-duration**(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") gibt "Months:03 Days:02 Hours:02 Minutes:53" gibt

▼ parse-duration [altova:]

altova:parse-duration(InputString *als xs:string*, **Picture** *als xs:string*) als **xs:duration** XP3.1 XQ3.1
Erhält einen Pattern-String als erstes Argument und eine Muster-String als zweites Argument. Der Input-Strings wird auf Basis des Muster-Strings geparst und ein xs:duration wird zurückgegeben.

Beispiele

- **altova:parse-duration**("Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") gibt "P2DT2H53M11.7S" zurück
- **altova:parse-duration**("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") gibt "P3M2DT2H53M" zurück

[Nach oben 1999]

Hinzufügen einer Zeitdauer zu xs:time XP3.1 XQ3.1

Diese Funktionen fügen einen Zeitdauerwert zu **xs:time** hinzu und geben **xs:time** zurück. Der Typ xs:time entspricht in seiner lexikalischen Form hh:mm:ss.sss. Eine optionale Zeitzone kann angehängt werden. Der Buchstabe Z steht für Coordinated Universal Time (UTC). Alle anderen Zeitzonen werden in Form des Zeitunterschieds zur UTC im Format +hh:mm, oder -hh:mm dargestellt. Wenn kein Wert für die Zeitzone vorhanden ist, wird sie als unbekannt und nicht als UTC angenommen.

▼ add-hours-to-time [altova:]

altova:add-hours-to-time(Time *als xs:time*, **Hours** *als xs:integer*) als **xs:time** XP3.1 XQ3.1
Fügt eine Zeitdauer in Stunden zu einem Uhrzeitwert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Stunden, die zu dem im ersten Argument angegebenen xs:time Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ xs:time.

Beispiele

- **altova:add-hours-to-time**(xs:time("11:00:00"), 10) gibt 21:00:00 zurück
- **altova:add-hours-to-time**(xs:time("11:00:00"), -7) gibt 04:00:00 zurück

▼ add-minutes-to-time [altova:]

altova:add-minutes-to-time(Time *als xs:time*, **Minutes** *als xs:integer*) als **xs:time** XP3.1 XQ3.1
Fügt eine Zeitdauer in Minuten zu einem xs:time Wert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Minuten, die zu dem im ersten Argument angegebenen xs:time Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ xs:time.

Beispiele

- **altova:add-minutes-to-time**(xs:time("14:10:00"), 45) gibt 14:55:00 zurück

© 2016-2022 Altova GmbH

- **altova:add-minutes-to-time**(xs:time("14:10:00"), -5) gibt 14:05:00 zurück

add-seconds-to-time [altova:]

altova:add-seconds-to-time(Time *as xs:time*, **Minutes** *als xs:integer***)** als **xs:time** XP3.1 XQ3.1
Fügt eine Zeitdauer in Sekunden zu einem Uhrzeitwert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Sekunden, die zu dem im ersten Argument angegebenen xs:time Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ xs:time. Die Seconds Komponenten kann sich im Bereich von 0 bis 59.999 befinden.

Beispiele

- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20) gibt 14:00:20 zurück
- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20.895) gibt 14:00:20.895 zurück

[Nach oben 1999]

Entfernen der Zeitzone aus date/time-Datentypen XP3.1 XQ3.1

Diese Funktionen entfernen die Zeitzone aus den aktuellen **xs:dateTime**, **xs:date** bzw. **xs:time** Werten. Beachten Sie, dass im Unterschied zu xs:dateTime bei xs:dateTimeStamp die Zeitzone erforderlich ist (während sie im ersteren Fall optional ist). Das Format eines xs:dateTimeStamp Werts lautet daher: JJJJ-MM-TTZhh:mm:ss.sss±hh:mm. oder JJJJ-MM-TTZhh:mm:ss.sssZ. Wenn das Datum und die Uhrzeit von der Systemuhr als xs:dateTimeStamp ausgelesen wird, können Sie die Zeitzone, falls erforderlich, mit der Funktion current-dateTime-no-TZ() entfernen.

current-date-no-TZ [altova:]

altova:current-date-no-TZ() als **xs:date** XP3.1 XQ3.1
Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem current-date() Wert (welcher das aktuelle Datum laut Systemuhr ist) und gibt einen xs:date Wert zurück.

Beispiele

Wenn das aktuelle Datum **2014-01-15+01:00** lautet:

- **altova:current-date-no-TZ**() gibt 2014-01-15 zurück

current-dateTime-no-TZ [altova:]

altova:current-dateTime-no-TZ() als **xs:dateTime** XP3.1 XQ3.1
Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem current-dateTime() Wert (welcher das aktuelle Datum und die aktuelle Uhrzeit laut Systemuhr ist) und gibt einen xs:dateTime Wert zurück.

Beispiele

Wenn der aktuelle Datums- und Uhrzeitwert **2014-01-15T14:00:00+01:00** lautet:

- **altova:current-dateTime-no-TZ**() gibt 2014-01-15T14:00:00 zurück

current-time-no-TZ [altova:]

altova:current-time-no-TZ() als **xs:time** XP3.1 XQ3.1
Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem `current-time()` Wert (welcher die aktuelle Uhrzeit laut Systemuhr ist) und gibt einen `xs:time` Wert zurück.

Beispiele

Wenn der aktuelle Uhrzeitwert **14:00:00+01:00** lautet:

- **altova:current-time-no-TZ**() gibt 14:00:00 zurück

date-no-TZ [altova:]

altova:date-no-TZ(InputDate *as xs:date*) als **xs:date** XP3.1 XQ3.1
Diese Funktion verwendet ein `xs:date` Argument, entfernt den Zeitzonenteil daraus und gibt einen `xs:date` Wert zurück. Beachten Sie, dass das Datum nicht geändert wird..

Beispiele

- **altova:date-no-TZ**(xs:date("2014-01-15+01:00")) gibt 2014-01-15 zurück

dateTime-no-TZ [altova:]

altova:dateTime-no-TZ(InputDateTime *als xs:dateTime*) als **xs:dateTime** XP3.1 XQ3.1
Diese Funktion verwendet ein `xs:dateTime` Argument, entfernt den Zeitzonenteil daraus und gibt einen `xs:dateTime` Wert zurück. Beachten Sie, dass weder Datum noch Uhrzeit geändert werden.

Beispiele

- **altova:dateTime-no-TZ**(xs:date("2014-01-15T14:00:00+01:00")) gibt 2014-01-15T14:00:00 zurück

time-no-TZ [altova:]

altova:time-no-TZ(InputTime *als xs:time*) als **xs:time** XP3.1 XQ3.1
Diese Funktion verwendet ein `xs:time` Argument, entfernt den Zeitzonenteil daraus und gibt einen `xs:time` Wert zurück. Beachten Sie, dass die Uhrzeit nicht geändert wird.

Beispiele

- **altova:time-no-TZ**(xs:time("14:00:00+01:00")) gibt 14:00:00 zurück

[Nach oben 1999]

Rückgabe der Anzahl von Tagen, Stunden, Minuten, Sekunden anhand einer Zeitdauer XP3.1 XQ3.1

Diese Funktionen geben die Anzahl der Tage in einem Monat bzw. die Anzahl der Stunden, Minuten und Sekunden anhand einer Zeitdauer zurück.

days-in-month [altova:]

altova:days-in-month(Year *als xs:integer*, **Month** *als xs:integer*) als **xs:integer** XP3.1 XQ3.1
Gibt die Anzahl der Tage im angegebenen Monat zurück. Der Monat wird mit Hilfe der Argumente `Year`

© 2016-2022 Altova GmbH

und `Month` angegeben.

Beispiele

- **`altova:days-in-month`**`(2018, 10)` gibt `31` zurück
- **`altova:days-in-month`**`(2018, 2)` gibt `28` zurück
- **`altova:days-in-month`**`(2020, 2)` gibt `29` zurück

hours-from-dayTimeDuration-accumulated

`altova:hours-from-dayTimeDuration-accumulated(DayAndTime` *als xs:duration*`)` als **`xs:integer`**
XP3.1 **XQ3.1**

Gibt die Gesamtzahl der Stunden in der durch das Argument `DayAndTime` bereitgestellten Zeitdauer (vom Typ `xs:duration`) zurück. Die Stunden in den Komponenten `Day` und `Time` werden addiert, um ein Ergebnis in Form einer Ganzzahl zu erhalten. Nur für volle 60 Minuten wird eine neue Stunde berechnet. Negative Zeitdauerergebnisse ergeben einen negativen Stundenwert.

Beispiele

- **`altova:hours-from-dayTimeDuration-accumulated`**`(xs:duration("P5D"))` gibt `120` zurück, d.h. die Gesamtanzahl der Stunden in 5 Tagen.
- **`altova:hours-from-dayTimeDuration-accumulated`**`(xs:duration("P5DT2H"))` gibt `122` zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden.
- **`altova:hours-from-dayTimeDuration-accumulated`**`(xs:duration("P5DT2H60M"))` gibt `123` zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden und 60 Minuten.
- **`altova:hours-from-dayTimeDuration-accumulated`**`(xs:duration("P5DT2H119M"))` gibt `123` zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden und 119 Minuten.
- **`altova:hours-from-dayTimeDuration-accumulated`**`(xs:duration("P5DT2H120M"))` gibt `124` zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden und 120 Minuten.
- **`altova:hours-from-dayTimeDuration-accumulated`**`(xs:duration("-P5DT2H"))` gibt `-122` zurück.

minutes-from-dayTimeDuration-accumulated

`altova:minutes-from-dayTimeDuration-accumulated(DayAndTime` *als xs:duration*`)` als **`xs:integer`** **XP3.1** **XQ3.1**

Gibt die Gesamtzahl der Minuten in der durch das Argument `DayAndTime` bereitgestellten Zeitdauer (vom Typ `xs:duration`) zurück. Die Minuten in den Komponenten `Day` und `Time` werden addiert, um ein Ergebnis in Form einer Ganzzahl zu erhalten. Negative Zeitdauerergebnisse ergeben einen negativen Minutenwert.

Beispiele

- **`altova:minutes-from-dayTimeDuration-accumulated`**`(xs:duration("PT60M"))` gibt `60` zurück.
- **`altova:minutes-from-dayTimeDuration-accumulated`**`(xs:duration("PT1H"))` gibt `60` zurück, d.h. die Gesamtzahl der Minuten in 1 Stunde.
- **`altova:minutes-from-dayTimeDuration-accumulated`**`(xs:duration("PT1H40M")` gibt `100` zurück.
- **`altova:minutes-from-dayTimeDuration-accumulated`**`(xs:duration("P1D"))` gibt `1440` zurück, d.h. die Gesamtzahl der Minuten an einem Tag.
- **`altova:minutes-from-dayTimeDuration-accumulated`**`(xs:duration("-P1DT60M"))` gibt `-1500` zurück.

▼ seconds-from-dayTimeDuration-accumulated

altova:seconds-from-dayTimeDuration-accumulated(DayAndTime *als xs:duration*) als **xs:integer** XP3.1 XQ3.1
Gibt die Gesamtzahl der Sekunden in der durch das Argument `DayAndTime` bereitgestellten Zeitdauer (vom Typ `xs:duration`) zurück. Die Sekunden in den Komponenten `Day` und `Time` werden addiert, um ein Ergebnis in Form einer Ganzzahl zu erhalten. Negative Zeitdauerergebnisse ergeben einen negativen Sekundenwert.

Examples

- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1M")) gibt 60 zurück, d.h. die Gesamtzahl der Sekunden in 1 Minute.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H")) gibt 3600 zurück, d.h. die Gesamtzahl der Sekunden in 1 Stunde.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H2M")) gibt 3720 zurück.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("P1D"))gibt 86400 zurück, d.h. die Gesamtzahl der Sekunden an 1 Tag.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("-P1DT1M")) gibt -86460 zurück.

Rückgabe des Wochentages anhand von xs:dateTime oder xs:date XP3.1 XQ3.1

Diese Funktionen geben anhand des `xs:dateTime` oder `xs:date` Werts den Wochentag in Form einer Ganzzahl zurück. Die Tage der Woche sind (im amerikanischen Format) von `1` bis `7` nummeriert, wobei `Sonntag=1`. Im europäischen Format beginnt die Woche am Montag (`=1`). Das amerikanische Format, in dem `Sonntag=1`, kann mittels der Ganzzahl `0` definiert werden, wenn das Format mittels einer Ganzzahl angegeben werden kann.

▼ weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *als xs:dateTime*) als **xs:integer** XP3.1 XQ3.1
Erhält ein Datum mit einer Uhrzeit als einziges Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit `Sonntag=1` nummeriert. Wenn das europäische Format benötigt wird (wo `Montag=1`), verwenden Sie die andere Signatur dieser Funktion (*siehe nächste Signatur unten)*.

Beispiele

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00")) gibt 2 zurück, wobei 2 für Montag steht.

altova:weekday-from-dateTime(DateTime *als xs:dateTime*, **Format** *als xs:integer*) als **xs:integer** XP3.1 XQ3.1
Erhält ein Datum mit einer Uhrzeit als erstes Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit `Montag=1` nummeriert. Wenn das zweite (Integer)-Argument `0` ist, werden die Wochentage beginnend mit `Sonntag=1` von `1` bis `7` nummeriert. Wenn das zweite Argument eine andere Ganzzahl als `0` ist, so ist `Montag=1`. Wenn es kein zweites Argument gibt, wird die Funktion gelesen, als ob sie die andere Signatur dieser Funktion hätte

© 2016-2022 Altova GmbH

(*siehe vorherige Signatur).*

Beispiele

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 1) gibt 1, zurück, wobei 1 für Montag steht
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 4) gibt 1, zurück, wobei 1 für Montag steht
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 0) gibt 2, zurück, wobei 2 für Montag steht

weekday-from-date [altova:]

altova:weekday-from-date(Date *als xs:date***)** als **xs:integer** XP3.1 XQ3.1
Erhält ein Datum als einziges Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit Sonntag=1 nummeriert. Wenn das europäische Format benötigt wird (wo Montag=1), verwenden Sie die andere Signatur dieser Funktion (*siehe nächste Signatur unten).*

Beispiele

- **altova:weekday-from-date**(xs:date("2014-02-03+01:00")) gibt 2 zurück, wobei 2 für Montag steht.

altova:weekday-from-date(Date *als xs:date*, **Format** *als xs:integer***)** als **xs:integer** XP3.1 XQ3.1
Erhält ein Datum als erstes Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit Montag=1 nummeriert. Wenn das zweite Argument (Format) 0 ist, werden die Wochentage beginnend mit Sonntag=1 von 1 bis 7 nummeriert. Wenn das zweite Argument eine andere Ganzzahl als 0 ist, so ist Montag=1. Wenn es kein zweites Argument gibt, wird die Funktion gelesen, als ob sie die andere Signatur dieser Funktion hätte (*siehe vorherige Signatur).*

Beispiele

- **altova:weekday-from-date**(xs:date("2014-02-03"), 1) gibt 1, zurück, wobei 1 für Montag steht
- **altova:weekday-from-date**(xs:date("2014-02-03"), 4) gibt 1, zurück, wobei 1 für Montag steht
- **altova:weekday-from-date**(xs:date("2014-02-03"), 0) gibt 2, zurück, wobei 2 für Montag steht

[Nach oben 1999]

Rückgabe der Wochennummer anhand von xs:dateTime oder xs:date XP2 XQ1 XP3.1 XQ3.1

Diese Funktionen geben anhand von xs:dateTime oder xs:date die Wochennummer als Ganzzahl zurück. Die Wochennummer steht in den Kalenderformaten US, ISO/European und Islamic zur Verfügung. Die Wochennummerierung unterscheidet sich in diesen Kalenderformaten, da die Woche in diesen Formaten an unterschiedlichen Tagen beginnt (Im Format US am Sonntag, im Format ISO/European am Montag und im Format Islamic am Samstag).

weeknumber-from-date [altova:]

altova:weeknumber-from-date(Date *als xs:date*, **Calendar** *als xs:integer*) als **xs:integer** XP2 XQ1 XP3.1 XQ3.1
Gibt die Wochennummer des bereitgestellten **Date** Arguments als Ganzzahl zurück. Das zweite Argument (**Calendar**) definiert das zu verwendende Kalendersystem.
Unterstützte **Calendar** Werte sind:

- **0 = US-Kalender** (Woche beginnt am Sonntag)
- **1 = ISO-Standard, Europäischer Kalender** (Woche beginnt am Montag)
- **2 = Islamischer Kalender** (Woche beginnt am Samstag)

Der Standardwert ist **0**.

Beispiele

- `altova:weeknumber-from-date(xs:date("2014-03-23"), 0)` gibt `13` zurück
- `altova:weeknumber-from-date(xs:date("2014-03-23"), 1)` gibt `12` zurück
- `altova:weeknumber-from-date(xs:date("2014-03-23"), 2)` gibt `13` zurück
- `altova:weeknumber-from-date(xs:date("2014-03-23") )` gibt `13` zurück

Der Tag des Datums in den obigen Beispielen (`2014-03-23`) ist ein Sonntag. Daher ist der US- und der islamische Kalender dem europäischen Kalender an diesem Tag eine Woche voraus.

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(DateTime *als xs:dateTime*, **Calendar** *als xs:integer*) **als xs:integer** XP2 XQ1 XP3.1 XQ3.1
Gibt die Wochennummer des bereitgestellten **DateTime** Arguments als Ganzzahl zurück. Das zweite Argument (**Calendar**) definiert das zu verwendende Kalendersystem.
Unterstützte **Calendar** Werte sind:

- **0 = US-Kalender** (*Woche beginnt am Sonntag*)
- **1 = ISO-Standard, Europäischer Kalender** (*Woche beginnt am Montag*)
- **2 = Islamischer Kalender** (*Woche beginnt am Samstag*)

Der Standardwert ist **0**.

Beispiele

- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 0)` gibt `13` zurück
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 1)` gibt `12` zurück
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 2)` gibt `13` zurück
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00") )` gibt `13` zurück

Der Tag des Datums- und Uhrzeitwerts in den obigen Beispielen (`2014-03-23T00:00:00`) ist ein Sonntag. Daher ist der US- und der islamische Kalender dem europäischen Kalender an diesem Tag eine Woche voraus.

© 2016-2022 Altova GmbH

[Nach oben 1999]

Erstellen des Datums-, Uhrzeit- oder Zeitdauer-Datentyps anhand der lexikalischen Komponenten der einzelnen Typen XP3.1 XQ3.1

Die Funktionen erhalten die lexikalischen Komponenten des `xs:date`, `xs:time` oder `xs:duration`-Datentyps als Input-Argumente und kombinieren diese zum entsprechenden Datentyp.

▼ build-date [altova:]

altova:build-date(Year *als xs:integer*, **Month** *als xs:integer*, **Date** *als xs:integer*) als **xs:date** XP3.1 XQ3.1

Das erste, zweite und dritte Argument steht für das Jahr, bzw. den Monat bzw. das Datum. Sie werden zu einem Wert vom Typ `xs:date` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Datumsteils befinden. So sollte z.B. das zweite Argument nicht größer als `12` sein.

⊟ *Beispiele*

- **altova:build-date**(2014, 2, 03) gibt 2014-02-03 zurück

▼ build-time [altova:]

altova:build-time(Hours *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*) als **xs:time** XP3.1 XQ3.1

Das erste, zweite und dritte Argument steht für die Stunde (`0` bis `23`), bzw. die Minuten (`0` bis `59`) bzw. die Sekunden (`0` bis `59`). Sie werden zu einem Wert von Typ `xs:time` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Uhrzeitteils befinden. So sollte z.B. der zweite Parameter nicht größer als `59` sein. Um eine Zeitzone zum Wert hinzuzufügen, verwenden Sie die andere Signatur der Funktion (*siehe nächste Signatur)*.

⊟ *Beispiele*

- **altova:build-time**(23, 4, 57) gibt 23:04:57 zurück

altova:build-time(Hours *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*, **TimeZone** *als xs:string*) als **xs:time** XP3.1 XQ3.1

Das erste, zweite und dritte Argument steht für die Stunde (`0` bis `23`), bzw. die Minuten (`0` bis `59`) bzw. die Sekunden (`0` bis `59`). Das vierte Argument ist ein String, der den Zeitzonenteil des Werts liefert. Die vier Argumente werden zu einem Wert vom Typ `xs:time` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Uhrzeitteils befinden. So sollte z.B. das zweite Argument (Minuten) nicht größer als `59` sein.

⊟ *Beispiele*

- **altova:build-time**(23, 4, 57, '+1') gibt 23:04:57+01:00 zurück

▼ build-duration [altova:]

altova:build-duration(Years *als xs:integer*, **Months** *als xs:integer*) als **xs:yearMonthDuration** XP3.1 XQ3.1

Setzt aus zwei Argumenten einen Wert vom Typ `xs:yearMonthDuration` zusammen. Das erste Argument liefert den Jahr-Teil des Zeitdauerwerts, während das zweite Argument den Monat-Teil liefert. Wenn der zweite Parameter (Monate) größer oder gleich `12` ist, so wird die Ganzzahl durch 12 dividiert. Der Quotient wird zum ersten Argument hinzugefügt, um den Jahr-Teil des Zeitdauerwerts zu liefern, während der Rest (der Division) den Monat-Teil liefert. Eine Beschreibung zur Erstellung einer Zeitdauer vom Typ `xs:dayTimeDuration` finden Sie in der nächsten Signatur.

Beispiele

- **altova:build-duration**(2, 10) gibt P2Y10M zurück
- **altova:build-duration**(14, 27) gibt P16Y3M zurück
- **altova:build-duration**(2, 24) gibt P4Y zurück

altova:build-duration(Days *als xs:integer*, **Hours** *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*) als **xs:dayTimeDuration** XP3.1 XQ3.1

Kombiniert vier Argumente zu einem Wert vom Typ `xs:dayTimeDuration`. Das erste Argument liefert den Tage-Teil, das zweite die Stunden, das dritte die Minuten und das vierte die Sekunden des Zeitdauerwerts. Die einzelnen Uhrzeitparameter werden in den entsprechenden Wert für die nächsthöhere Einheit konvertiert und das Ergebnis wird zur Berechnung der Gesamtdauer weitergegeben. So werden z.B. 72 Sekunden in `1M(inute)12S(ekunden)` konvertiert. Dieser Wert wird zur Berechnung der Gesamtdauer weitergegeben. Um eine Zeitdauer vom Typ `xs:yearMonthDuration` zu berechnen, verwenden Sie die vorherige Signatur.

Beispiele

- **altova:build-duration**(2, 10, 3, 56) gibt P2DT10H3M56S zurück
- **altova:build-duration**(1, 0, 100, 0) gibt P1DT1H40M zurück
- **altova:build-duration**(1, 0, 0, 3600) gibt P1DT1H zurück

[Nach oben 1999]

Konstruieren von Datum, Datum und Uhrzeit und Zeit-Datentypen anhand des String-Input XP2 XQ1 XP3.1 XQ3.1

Diese Funktionen erhalten Strings als Argumente und konstruieren anhand dieser die Datentypen `xs:date`, `xs:dateTime` oder `xs:time`. Der String wird anhand eines bereitgestellten Pattern-Arguments nach Komponenten des Datentyps analysiert.

parse-date [altova:]

altova:parse-date(Date *als xs:string*, **DatePattern** *als xs:string*) als **xs:date** XP2 XQ1 XP3.1 XQ3.1

Gibt den Input-String **Date** als **xs:date** Wert zurück. Das zweite Argument **DatePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **DatePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

D	Datum
M	Monat
Y	Jahr

Das Pattern in **DatePattern** muss mit dem Pattern in **Date** übereinstimmen. Da die Ausgabe vom Typ **xs:date** ist, hat sie immer das lexikalische Format **YYYY-MM-DD**.

© 2016-2022 Altova GmbH

Beispiele

- **altova:parse-date**(xs:string("09-12-2014"), "[D]-[M]-[Y]") gibt 2014-12-09 zurück
- **altova:parse-date**(xs:string("09-12-2014"), "[M]-[D]-[Y]") gibt 2014-09-12 zurück
- **altova:parse-date**("06/03/2014", "[M]/[D]/[Y]") gibt 2014-06-03 zurück
- **altova:parse-date**("06 03 2014", "[M] [D] [Y]") gibt 2014-06-03 zurück
- **altova:parse-date**("6 3 2014", "[M] [D] [Y]") gibt 2014-06-03 zurück

parse-dateTime [altova:]

altova:parse-dateTime(DateTime *als xs:string*, **DateTimePattern** *als xs:string*) als **xs:dateTime** XP2 XQ1 XP3.1 XQ3.1

Gibt den Input-String **DateTime** als **xs:dateTime** Wert zurück. Das zweite Argument **DateTimePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **DateTimePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

D	Datum
M	Monat
Y	Jahr
H	Stunde
m	Minuten
s	Sekunden

Das Pattern in **DateTimePattern** muss mit dem Pattern in **DateTime** übereinstimmen. Da die Ausgabe vom Typ **xs:dateTime** ist, hat sie immer das lexikalische Format **YYYY-MM-DDTHH:mm:ss**.

Beispiele

- **altova:parse-dateTime**(xs:string("09-12-2014 13:56:24"), "[M]-[D]-[Y] [H]:[m]:[s]") gibt 2014-09-12T13:56:24 zurück
- **altova:parse-dateTime**("time=13:56:24; date=09-12-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") gibt 2014-12-09T13:56:24 zurück

parse-time [altova:]

altova:parse-time(Time *als xs:string*, **TimePattern** *als xs:string*) als **xs:time** XP2 XQ1 XP3.1 XQ3.1

Gibt den Input-String **Time** als **xs:time** Wert zurück. Das zweite Argument **TimePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **TimePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

H	Stunde
m	Minuten
s	Sekunden

Das Pattern in **TimePattern** muss mit dem Pattern in **Time** übereinstimmen. Da die Ausgabe vom Typ **xs:Time** ist, hat sie immer das lexikalische Format **HH:mm:ss**.

Beispiele

- **altova:parse-time**(xs:string("13:56:24"), "[H]:[m]:[s]") gibt 13:56:24 zurück
- **altova:parse-time**("13-56-24", "[H]-[m]") gibt 13:56:00 zurück
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") gibt 13:56:24 zurück
- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") gibt 13:56:24 zurück

[Nach oben 1999]

Funktionen zur Berechnung des Alters XP3.1 XQ3.1

Diese Funktionen geben das Alter berechnet (i) anhand von einem Input-Argument und dem aktuellen Datum oder (ii) anhand zweier Input-Argumentdaten zurück. Die Funktion **altova:age** gibt das Alter in Jahren zurück, die Funktion **altova:age-details** gibt das Alter als Sequenz von drei Ganzzahlen zurück, die die Jahre, Monate und Tage des Alters angeben.

age [altova:]

altova:age(StartDate *als xs:date***)** als **xs:integer** XP3.1 XQ3.1
Gibt eine Ganzzahl zurück, die das Alter eines Objekts in *Jahren* angibt. Berechnet wird das Alter anhand des durch das Argument gelieferten Startdatums endend mit dem aktuellen Datum (laut Systemuhr). Wenn das Input-Argument eines Datums größer oder gleich einem Jahr in der Zukunft ist, ist der Rückgabewert negativ.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- **altova:age**(xs:date("2013-01-15")) gibt 1 zurück
- **altova:age**(xs:date("2013-01-16")) gibt 0 zurück
- **altova:age**(xs:date("2015-01-15")) gibt -1 zurück
- **altova:age**(xs:date("2015-01-14")) gibt 0 zurück

altova:age(StartDate *als xs:date*, **EndDate** *als xs:date***)** als **xs:integer** XP3.1 XQ3.1
Gibt eine Ganzzahl zurück, die das Alter eines Objekts in *Jahren* angibt. Berechnet wird das Alter anhand des durch das erste Argument gelieferten Startdatums endend mit dem als zweites Datum gelieferten Enddatum. Wenn das erste Argument ein Jahr oder mehr nach dem zweiten Argument liegt, ist der Rückgabewert negativ.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- **altova:age**(xs:date("2000-01-15"), xs:date("2010-01-15")) gibt 10 zurück
- **altova:age**(xs:date("2000-01-15"), current-date()) gibt 14 zurück, wenn das aktuelle Datum 2014-01-15 ist
- **altova:age**(xs:date("2014-01-15"), xs:date("2010-01-15")) gibt -4 zurück

age-details [altova:]

© 2016-2022 Altova GmbH

altova:age-details(InputDate *als xs:date*) als (**xs:integer**)* XP3.1 XQ3.1
Gibt drei Ganzzahlen zurück. Dabei handelt es sich um die Jahre, Monate bzw. Tage zwischen dem als Argument angegebenen Datum und dem aktuellen Datum (laut Systemuhr). Die Summe der zurückgegebenen `years+months+days` gibt zusammen die Gesamtzeitdifferenz zwischen den beiden Datumswerten (dem Input-Datum und dem aktuellen Datum) an. Das Input-Datum hat eventuell einen Wert, der vor oder nach dem aktuellen Datum liegt, doch wird dies nicht aus dem Vorzeichen der Rückgabewerte ersichtlich; die Rückgabewerte sind immer positiv.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- `altova:age-details(xs:date("2014-01-16"))` gibt `(0 0 1)` zurück
- `altova:age-details(xs:date("2014-01-14"))` gibt `(0 0 1)` zurück
- `altova:age-details(xs:date("2013-01-16"))` gibt `(1 0 1)` zurück
- `altova:age-details(current-date())` gibt `(0 0 0)` zurück

altova:age-details(Date-1 *als xs:date*, **Date-2** *als xs:date*) als (**xs:integer**)* XP3.1 XQ3.1
Gibt drei Ganzzahlen zurück. Dabei handelt es sich um die Jahre, Monate bzw. Tage zwischen den beiden Argumentdaten. Die Summe der zurückgegebenen `years+months+days` gibt zusammen die Gesamtzeitdifferenz zwischen den beiden Input-Datumswerten an. Es ist unerheblich, ob das frühere oder spätere Datum als erstes Argument angegeben wird. Die Rückgabewerte geben nicht an, ob das Input-Datum vor oder nach dem aktuellen Datum liegt. Die Rückgabewerte sind immer positiv.

Beispiele

- `altova:age-details(xs:date("2014-01-16"), xs:date("2014-01-15"))` gibt `(0 0 1)` zurück
- `altova:age-details(xs:date("2014-01-15"), xs:date("2014-01-16"))` gibt `(0 0 1)` zurück

[Nach oben 1999]

Epochen-Zeit (Unix-Zeit)-Funktionen XP3.1 XQ3.1

Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Diese Epochenzeitfunktionen konvertieren **xs:dateTime**-Werte in Epochenzeitwerte und umgekehrt.

dateTime-from-epoch [altova:]

altova:dateTime-from-epoch(Epoch *als xs:decimal* **als xs:dateTime** XP3.1 XQ3.1
Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Die Funktion **dateTime-from-epoch** gibt das **xs:dateTime**-Äquivalent einer Epochenzeit zurück, passt die lokale Zeitzone an und inkludiert die Zeitzoneninformation im Ergebnis.

Die Funktion erhält ein `xs:decimal`-Argument und gibt einen `xs:dateTime`-Wert, der einen **ZZ**-Teil (Zeitzone) enthält, zurück. Das Ergebnis wird durch Berechnung des UTC **dateTime**-Äquivalents der Epochenzeit und Hinzufügen der (anhand der Systemuhr ermittelten) lokalen Zeitzone ermittelt. Wenn die Funktion z.B. auf einem Rechner, der in der Zeitzone +01:00 (relativ zur UTC) konfiguriert wurde, ausgeführt wird, so wird nach Berechnung des UTC-**dateTime**-Äquivalents eine Stunde zum Ergebnis addiert. Auch die Zeitzoneninformation, die einen optionalen lexikalischen Bestandteil des **xs:dateTime**-Ergebnisses bildet, wird im **dateTime**-Ergebnis ausgegeben. Vergleichen Sie dieses Ergebnis mit dem von **dateTime-from-epoch-no-TZ** und auch der Funktion **epoch-from-dateTime**.

Beispiele

In den Beispielen unten wird eine lokale Zeitzone von UTC +01:00 angenommen. Das UTC **dateTime**-Äquivalent der angegebenen Epochenzeit wird folglich um eine Stunde erhöht. Die Zeitzone wird im Ergebnis ausgegeben.

- **altova:dateTime-from-epoch**(34) gibt 1970-01-01T01:00:34+01:00 zurück.
- **altova:dateTime-from-epoch**(62) gibt 1970-01-01T01:01:02+01:00 zurück.

dateTime-from-epoch-no-TZ [altova:]

altova:dateTime-from-epoch-no-TZ(Epoch *als xs:decimal* **als xs:dateTime** XP3.1 XQ3.1
Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Die Funktion **dateTime-from-epoch-no-TZ** gibt das **xs:dateTime**-Äquivalent einer Epochenzeit zurück, passt es an die lokale Zeitzone an, inkludiert die Zeitzoneninformation jedoch nicht im Ergebnis.

Die Funktion erhält ein xs:decimal-Argument und gibt einen xs:dateTime-Wert, der keinen **ZZ**-Teil (Zeitzone) enthält, zurück. Das Ergebnis wird durch Berechnung des UTC **dateTime**-Äquivalents der Epochenzeit und Hinzufügen der (anhand der Systemuhr ermittelten) lokalen Zeitzone ermittelt. Wenn die Funktion z.B. auf einem Rechner, der in der Zeitzone +01:00 (relativ zur UTC) konfiguriert wurde, ausgeführt wird, so wird nach Berechnung des UTC-**dateTime**-Äquivalents eine Stunde zum Ergebnis addiert. Die Zeitzoneninformation, die einen optionalen lexikalischen Bestandteil des **xs:dateTime**-Ergebnisses bildet, wird nicht im **dateTime**-Ergebnis ausgegeben. Vergleichen Sie dieses Ergebnis mit dem von **dateTime-from-epoch** und auch der Funktion **epoch-from-dateTime**.

Beispiele

In den Beispielen unten wird eine lokale Zeitzone von UTC +01:00 angenommen. Das UTC **dateTime**-Äquivalent der angegebenen Epochenzeit wird folglich um eine Stunde erhöht. Die Zeitzone wird nicht im Ergebnis ausgegeben.

- **altova:dateTime-from-epoch**(34) gibt 1970-01-01T01:00:34 zurück.
- **altova:dateTime-from-epoch**(62) gibt 1970-01-01T01:01:02 zurück.

epoch-from-dateTime [altova:]

altova:epoch-from-dateTime(dateTimeValue *als xs:dateTime*) **als xs:decimal** XP3.1 XQ3.1
Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Die Funktion **epoch-from-dateTime** gibt das Epochenzeitäquivalent von **xs:dateTime** zurück, welches als Argument der Funktion bereitgestellt wird. Beachten Sie, dass Sie den **xs:dateTime**-Wert eventuell explizit konstruieren müssen. Der angegebene **xs:dateTime**-Wert kann den optionalen **ZZ** (Zeitzonen)-Wert enthalten, muss ihn aber nicht enthalten.

Unabhängig davon, ob der Zeitzonenteil als Bestandteil des Arguments angegeben wird oder nicht, wird der (anhand der Systemuhr ermittelte) lokale Zeitzonenunterschied vom angegebenen **dateTimeValue**-Argument subtrahiert. Dadurch wird das UTC-Zeit-Äquivalent erzeugt, anhand dessen die entsprechende Epochenzeit berechnet wird. Wenn die Funktion z.B. auf einem Rechner, der für die Zeitzone +01:00

© 2016-2022 Altova GmbH

(relativ zur UTC) konfiguriert wurde, ausgeführt wird, so wird vor Berechnung des Epochenzeitwerts eine Stunde vom angegebenen **dateTimeValue** subtrahiert. Siehe dazu auch die Funktion **dateTime-from-epoch**.

Beispiele

In den Beispielen unten wird eine lokale Zeitzone von UTC +01:00 angenommen. Daher wird vor Berechnung der Epochenzeit eine Stunde vom angegebenen **dateTime**-Wert subtrahiert.

- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34+01:00")) gibt 34 zurück.
- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34")) gibt 34 zurück.
- **altova:epoch-from-dateTime**(xs:dateTime("2021-04-01T11:22:33")) gibt 1617272553 zurück.

[Nach oben 1999]

19.2.2.1.3 XPath/XQuery-Funktionen: Standort

Die folgenden XPath/XQuery-Erweiterungsfunktionen zu Standortdaten werden in der aktuellen Version von MapForce unterstützt und können in (i) in einem XSLT-Kontext in XPath-Ausdrücken oder (ii) in einem XQuery-Dokument in einem XQuery-Ausdruck verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, http://www.altova.com/xslt-extensions** und sind in diesem Abschnitt mit dem Präfix **altova:**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

format-geolocation [altova:]

altova:format-geolocation(Latitude *als xs:decimal*, **Longitude** *als xs:decimal*, **GeolocationOutputStringFormat** *als xs:integer*) als **xs:string** XP3.1 XQ3.1
Erhält als die ersten beiden Argumente die geografische Breite und Länge und gibt den Standort als String zurück. Das dritte Argument, **GeolocationOutputStringFormat**, ist das Format des Ausgabestring für den Standort; darin werden zum Identifizieren des Ausgabestringformats Ganzzahlwerte von 1 bis 4 verwendet (*siehe 'Format des Ausgabestrings für die geografische Position' weiter unten*). Die Werte für die Breite liegen im Bereich von +90 bis -90 (N nach S). Die Werte für die Länge liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Zur Bereitstellung der Input-Strings können die Funktion `image-exif-data` 2029 und die Attribute der Exif-Metadaten verwendet werden.

- *Beispiele*

  - **`altova:format-geolocation`**`(33.33, -22.22, 4)` gibt `xs:string "33.33 -22.22"` zurück
  - **`altova:format-geolocation`**`(33.33, -22.22, 2)` gibt `xs:string "33.33N 22.22W"` zurück
  - **`altova:format-geolocation`**`(-33.33, 22.22, 2)` gibt `xs:string "33.33S 22.22E"` zurück
  - **`altova:format-geolocation`**`(33.33, -22.22, 1)` gibt `xs:string "33°19'48.00"S 22° 13'12.00"E"` zurück

- *Ausgabestringformate für die geografische Position:*

  Die bereitgestellte Breite und Länge ist in einem der unten aufgelisteten Ausgabeformate formatiert. Das gewünschte Format wird anhand seiner Ganzzahl-ID (`1` bis `4`) identifiziert. Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

1
Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, E/W`) **`D°M'S.SS"N/S D°M'S.SS"E/W`** *Beispiel:* **`33°55'11.11"N 22°44'66.66"W`**

2
Dezimalgrad, mit nachgestellter Orientierung (`N/S`, `E/W`) **`D.DDN/S D.DDE/W`** *Beispiel:* **`33.33N 22.22W`**

3
Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); Plus-Zeichen für (**`N/E`**) ist optional **`+/-D°M'S.SS" +/-D°M'S.SS"`** *Beispiel:* **`33°55'11.11" -22°44'66.66"`**

4
Dezimalgrad, mit Vorzeichen (`+/-`); Plus-Zeichen für (**`N/E`**) ist optional **`+/-D.DD +/-D.DD`** *Beispiel:* **`33.33 -22.22`**

- *Altova Exif-Attribut: Geolocation*

  Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **`Geolocation`**. **`Geolocation`** ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe Tabelle unten*).

`GPSLatitude`	**`GPSLatitudeRe`**	**`GPSLongitude`**	**`GPSLongitudeRe`**	**`Geolocation`**

© 2016-2022 Altova GmbH

	f		f	
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

parse-geolocation [altova:]

altova:parse-geolocation(GeolocationInputString *als xs:string***)** als **xs:decimal+** XP3.1 XQ3.1
Parst das bereitgestellte `GeolocationInputString`-Argument und gibt die geografische Breite und Länge (in dieser Reihenfolge) als Sequenz aus zwei `xs:decimal` Elementen zurück. Die Formate, in denen der Input-String für die geografische Position bereitgestellt werden kann, sind unten aufgelistet.

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion `image-exif-data`2029 und das `@Geolocation`2029-Attribut der Exif-Metadaten verwendet werden (*siehe Beispiel unten*).

Beispiele

- **altova:parse-geolocation**("33.33 -22.22") gibt die Sequenz bestehend aus zwei `xs:decimals` (33.33, 22.22) Elementen zurück
- **altova:parse-geolocation**("48°51'29.6""N 24°17'40.2""") gibt die Sequenz bestehend aus zwei `xs:decimals` (48.8582222222222, 24.2945) Elementen zurück
- **altova:parse-geolocation**('48°51''29.6"N 24°17''40.2"') gibt die Sequenz bestehend aus zwei `xs:decimals` (48.8582222222222, 24.2945) Elementen zurück
- **altova:parse-geolocation**(**image-exif-data**(//MyImages/Image20141130.01)/**@Geolocation**) gibt die Sequenz bestehend aus zwei `xs:decimals` Elementen zurück

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert ("), während Maßeinheitsangaben blau ("") markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Beispiel: **33°55'11.11"N 22°44'55.25"W**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N/O`**) ist optional
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Beispiel: **`33°55'11.11" -22°44'55.25"`**

- Grad, Dezimalminuten mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D°M.MM'N/S D°M.MM'W/E`
  Beispiel: **`33°55.55'N 22°44.44'W`**

- Grad, Dezimalminuten mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N/O`**) ist optional
  `+/-D°M.MM' +/-D°M.MM'`
  Beispiel: **`+33°55.55' -22°44.44'`**

- Dezimalgrade, mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D.DDN/S D.DDW/E`
  Beispiel: **`33.33N 22.22W`**

- Dezimalgrade mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N/S O/W`**) ist optional
  `+/-D.DD +/-D.DD`
  Beispiel: **`33.33 -22.22`**

Beispiele für Formatkombinationen:
`33.33N -22°44'55.25"`
`33.33 22°44'55.25"W`
`33.33 22.45`

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **`Geolocation`**. **`Geolocation`** ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-km [altova:]

`altova:geolocation-distance-km(GeolocationInputString-1` *`als xs:string`*`,` **`GeolocationInputString-2`** *`als xs:string`***`)`** als **`xs:decimal`** **`XP3.1`** **`XQ3.1`**
Berechnet die Entfernung zwischen zwei geografischen Positionen in Kilometern. Die Formate, in denen der Input-String für die geografischen Position angegeben werden kann, sind unten aufgelistet. Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion `image-exif-data`[2029] und das `@Geolocation`[2029]-Attribut der Exif-Metadaten verwendet werden.

© 2016-2022 Altova GmbH

- *Beispiele*
  - **altova:geolocation-distance-km**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""") gibt xs:decimal 4183.08132372392 zurück

- *Input-String-Formate der Standortdaten:*

  Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

  Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert ("), während Maßeinheitsangaben blau ("") markiert sind.

  - Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (N/S, O/W)
    D°M'S.SS"N/S D°M'S.SS"W/E
    Beispiel: **33°55'11.11"N 22°44'55.25"W**

  - Grad, Minuten, Dezimalsekunden mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
    +/-D°M'S.SS" +/-D°M'S.SS"
    Beispiel: **33°55'11.11" -22°44'55.25"**

  - Grad, Dezimalminuten mit nachgestellter Orientierung (N/S, O/W)
    D°M.MM'N/S D°M.MM'W/E
    Beispiel: **33°55.55'N 22°44.44'W**

  - Grad, Dezimalminuten mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
    +/-D°M.MM' +/-D°M.MM'
    Beispiel: **+33°55.55' -22°44.44'**

  - Dezimalgrade, mit nachgestellter Orientierung (N/S, O/W)
    D.DDN/S D.DDW/E
    Beispiel: **33.33N 22.22W**

  - Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (**N/S O/W**) ist optional
    +/-D.DD +/-D.DD
    Beispiel: **33.33 -22.22**

  Beispiele für Formatkombinationen:
  33.33N -22°44'55.25"
  33.33 22°44'55.25"W
  33.33 22.45

- *Altova Exif-Attribut: Geolocation*

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

▼ geolocation-distance-mi [altova:]

altova:geolocation-distance-mi(GeolocationInputString-1 *als xs:string*, **GeolocationInputString-2** *als xs:string*) als **xs:decimal** **XP3.1** **XQ3.1**
Berechnet die Entfernung zwischen zwei geografischen Positionen in Meilen. Die Formate, in denen der Input-String für die geografischen Position angegeben werden kann, sind unten aufgelistet. Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion `image-exif-data`[2029] und das `@Geolocation`[2029]-Attribut der Exif-Metadaten verwendet werden.

⊟ *Beispiele*

- **altova:geolocation-distance-mi**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""") gibt `xs:decimal 2599.40652340653` zurück

⊟ *Input-String-Formate der Standortdaten:*

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert ("), während Maßeinheitsangaben blau ("") markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Beispiel: **33°55'11.11"N 22°44'55.25"W**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); das Plus-Zeichen für (**N/O**) ist

© 2016-2022 Altova GmbH

optional
+/-D°M'S.SS" +/-D°M'S.SS"
Beispiel: **33°55'11.11" -22°44'55.25"**

- Grad, Dezimalminuten mit nachgestellter Orientierung (N/S, O/W)
  D°M.MM'N/S D°M.MM'W/E
  Beispiel: **33°55.55'N 22°44.44'W**

- Grad, Dezimalminuten mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M.MM' +/-D°M.MM'
  Beispiel: **+33°55.55' -22°44.44'**

- Dezimalgrade, mit nachgestellter Orientierung (N/S, O/W)
  D.DDN/S D.DDW/E
  Beispiel: **33.33N 22.22W**

- Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (**N/S O/W**) ist optional
  +/-D.DD +/-D.DD
  Beispiel: **33.33 -22.22**

Beispiele für Formatkombinationen:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocations-bounding-rectangle [altova:]

altova:geolocations-bounding-rectangle(Geolocations *als xs:sequence*, **GeolocationOutputStringFormat** *als xs:integer***)** als **xs:string** XP3.1 XQ3.1
Erhält als erstes Argument eine Sequenz von Strings, wobei es sich bei jedem String in der Sequenz um eine geografische Position handelt. Die Funktion gibt eine Sequenz von zwei Strings zurück, die die geografischen Positionskoordinaten der linken oberen bzw. rechten unteren Ecke eines Rechtecks bilden, dessen Größe so angepasst ist, dass es alle im ersten Argument angegebenen Positionskoordinaten enthält. Die Formate, in denen der Input-String für die geografischen Position angegeben werden kann, sind unten aufgelistet (*siehe 'Input-String-Formate der Standortdaten'*). Die Breitenwerte liegen im Bereich von +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Im zweiten Argument der Funktion ist das Format der beiden Geolocation-Strings in der Ausgabesequenz angegeben. Das Argument erhält einen Ganzzahlwert von `1` bis `4`, wobei die einzelnen Werte ein jeweils unterschiedliches String-Format definieren (*siehe 'Ausgabestringsformate für die geografische Position' weiter unten).*

Anmerkung: Zur Bereitstellung der Input-Strings können die Funktion `image-exif-data` 2029 und die Attribute der Exif-Metadaten verwendet werden.

Beispiele

- **`altova:geolocations-bounding-rectangle`**`(("48.2143531 16.3707266", "51.50939 -0.11832"), 1)` gibt die Sequenz `("51°30'33.804"N 0°7'5.952"W", "48°12'51.67116"N 16°22'14.61576"E")`zurück.
- **`altova:geolocations-bounding-rectangle`**`(("48.2143531 16.3707266", "51.50939 -0.11832", "42.5584577 -70.8893334"), 4)` gibt die Sequenz `("51.50939 -70.8893334", "42.5584577 16.3707266")` zurück.

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (**`"`**), während Maßeinheitsangaben blau (**`""`**) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)
  `D°M'S.SS"N/S D°M'S.SS"W/E`
  Beispiel: **`33°55'11.11"N 22°44'55.25"W`**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N/O`**) ist optional
  `+/-D°M'S.SS" +/-D°M'S.SS"`
  Beispiel: **`33°55'11.11" -22°44'55.25"`**

- Grad, Dezimalminuten mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D°M.MM'N/S D°M.MM'W/E`
  Beispiel: **`33°55.55'N 22°44.44'W`**

- Grad, Dezimalminuten mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N/O`**) ist optional
  `+/-D°M.MM' +/-D°M.MM'`
  Beispiel: **`+33°55.55' -22°44.44'`**

- Dezimalgrade, mit nachgestellter Orientierung (`N/S`, `O/W`)

© 2016-2022 Altova GmbH

D.DDN/S D.DDW/E
Beispiel: **33.33N 22.22W**

- Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (**N/S O/W**) ist optional
  +/-D.DD +/-D.DD
  Beispiel: **33.33 -22.22**

Beispiele für Formatkombinationen:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

- *Ausgabestringformate für die geografische Position:*

Die bereitgestellte Breite und Länge ist in einem der unten aufgelisteten Ausgabeformate formatiert. Das gewünschte Format wird anhand seiner Ganzzahl-ID (1 bis 4) identifiziert. Die Breitenwerte liegen im Bereich von +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

1
Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (N/S, E/W) **D°M'S.SS"N/S D°M'S.SS"E/W** *Beispiel*: **33°55'11.11"N 22°44'66.66"W**

2
Dezimalgrad, mit nachgestellter Orientierung (N/S, E/W) **D.DDN/S D.DDE/W** *Beispiel*: **33.33N 22.22W**

3
Grad, Minuten, Dezimalsekunden mit Vorzeichen (+/-); Plus-Zeichen für (**N/E**) ist optional **+/-D°M'S.SS" +/-D°M'S.SS"** *Beispiel*: **33°55'11.11" -22°44'66.66"**

4
Dezimalgrad, mit Vorzeichen (+/-); Plus-Zeichen für (**N/E**) ist optional **+/-D.DD +/-D.DD** *Beispiel*: **33.33 -22.22**

- *Altova Exif-Attribut: Geolocation*

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

▼ geolocation-within-polygon [altova:]

altova:geolocation-within-polygon(Geolocation *als xs:string*, **((PolygonPoint** *als xs:string*)+))** als **xs:boolean** XP3.1 XQ3.1
Ermittelt ob sich **Geolocation** (das erste Argument) innerhalb des durch die **PolygonPoint**-Argumente beschriebenen Polygonbereichs befindet. Wenn die `PolygonPoint`-Argumente keine geschlossene Form (wenn der erste und der letzte Punkt identisch sind) bilden, so wird der erste Punkt implizit zum letzten Punkt hinzugefügt, um die Form zu schließen. Alle Argumente (`Geolocation` und `PolygonPoint+`) werden durch Input-Strings für die geografische Position (*Formatliste siehe unten*) angegeben. Wenn sich das `Geolocation` Argument innerhalb des Polygons befindet, gibt die Funktion `true()`zurück; andernfalls gibt sie `false()` zurück. Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion `image-exif-data`[2029] und das `@Geolocation`[2029]-Attribut der Exif-Metadaten verwendet werden.

▣ *Beispiele*

- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48 24", "58 -32")**)** gibt `true()` zurück
- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48 24")**)** gibt `true()` zurück
- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48°51'29.6""N 24°17'40.2""")**)** gibt `true()` zurück

▣ *Input-String-Formate der Standortdaten:*

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (**"**), während Maßeinheitsangaben blau (**""**) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)

© 2016-2022 Altova GmbH

D°M'S.SS"N/S D°M'S.SS"W/E
Beispiel: **33°55'11.11"N 22°44'55.25"W**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M'S.SS" +/-D°M'S.SS"
  Beispiel: **33°55'11.11" -22°44'55.25"**

- Grad, Dezimalminuten mit nachgestellter Orientierung (N/S, O/W)
  D°M.MM'N/S D°M.MM'W/E
  Beispiel: **33°55.55'N 22°44.44'W**

- Grad, Dezimalminuten mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M.MM' +/-D°M.MM'
  Beispiel: **+33°55.55' -22°44.44'**

- Dezimalgrade, mit nachgestellter Orientierung (N/S, O/W)
  D.DDN/S D.DDW/E
  Beispiel: **33.33N 22.22W**

- Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (**N/S O/W**) ist optional
  +/-D.DD +/-D.DD
  Beispiel: **33.33 -22.22**

Beispiele für Formatkombinationen:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-within-rectangle [altova:]

altova:geolocation-within-rectangle(Geolocation *als xs:string*, **RectCorner-1** *als xs:string*, **RectCorner-2** *als xs:string*) als **xs:boolean** XP3.1 XQ3.1
Ermittelt, ob sich **Geolocation** (das erste Argument) innerhalb des durch das zweite und dritte Argument, **RectCorner-1** und **RectCorner-2**, definierten Rechtecks befindet. **RectCorner-1** und **RectCorner-2** definieren gegenüberliegende Eckpunkte des Rechtecks. Alle Argumente (Geolocation, **RectCorner-1**

und **RectCorner-2**) werden durch Input-Strings für die geografische Position (*Formatliste siehe unten*) angegeben. Wenn sich das Geolocation-Argument innerhalb des Rechtecks befindet, gibt die Funktion true() zurück; andernfalls gibt sie false() zurück. Die Breitenwerte liegen im Bereich von +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion image-exif-data[2029] und das @Geolocation[2029]-Attribut der Exif-Metadaten verwendet werden.

Beispiele

- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "-48 24"**)** gibt true() zurück
- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "48 24"**)** gibt false() zurück
- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "48°51'29.6""S 24° 17'40.2"""**)** gibt true() zurück

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert ("), während Maßeinheitsangaben blau ("") markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (N/S, O/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Beispiel: **33°55'11.11"N 22°44'55.25"W**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M'S.SS" +/-D°M'S.SS"
  Beispiel: **33°55'11.11" -22°44'55.25"**

- Grad, Dezimalminuten mit nachgestellter Orientierung (N/S, O/W)
  D°M.MM'N/S D°M.MM'W/E
  Beispiel: **33°55.55'N 22°44.44'W**

- Grad, Dezimalminuten mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M.MM' +/-D°M.MM'
  Beispiel: **+33°55.55' -22°44.44'**

- Dezimalgrade, mit nachgestellter Orientierung (N/S, O/W)
  D.DDN/S D.DDW/E
  Beispiel: **33.33N 22.22W**

© 2016-2022 Altova GmbH

- Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (**N/S O/W**) ist optional
  +/-D.DD +/-D.DD
  Beispiel: **33.33 -22.22**

Beispiele für Formatkombinationen:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

[Nach oben 2017]

19.2.2.1.4 XPath/XQuery-Funktionen: Bildbezogene

Die folgenden XPath/XQuery-Erweiterungsfunktionen im Zusammenhang mit Bildern werden in der aktuellen Version von MapForce unterstützt und können in (i) in einem XSLT-Kontext in XPath-Ausdrücken oder (ii) in einem XQuery-Dokument in einem XQuery-Ausdruck verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3

XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

▼ suggested-image-file-extension [altova:]

altova:suggested-image-file-extension(Base64String *als string***)** als **string?** XP3.1 XQ3.1
Erhält die Base64-Kodierung einer Bilddatei als Argument und gibt die darin enthaltene Dateierweiterung des Bilds zurück. Der Rückgabewert ist ein Vorschlag, basierend auf den in der Kodierung enthaltenen Bilddateitypinformationen. Wenn diese Informationen nicht verfügbar sind, wird ein leerer String zurückgegeben. Diese Funktion ist nützlich, wenn Sie ein Base64-Bild als Datei speichern und die entsprechende Dateierweiterung dynamisch abrufen möchten.

⊟ *Beispiele*

- `altova:suggested-image-file-extension(/MyImages/MobilePhone/Image20141130.01)` gibt `'jpg'` zurück
- `altova:suggested-image-file-extension($XML1/Staff/Person/@photo)` gibt `''` zurück

In den Beispielen oben wird von den als Argument der Funktion bereitgestellten Nodes angenommen, dass sie ein Base64-kodiertes Bild enthalten. Im ersten Beispiel wird `jpg` als Dateityp bzw. Dateierweiterung abgerufen. Im zweiten Beispiel enthält die angegebene Base54-Kodierung keine brauchbaren Dateierweiterungsinformationen.

▼ image-exif-data [altova:]

altova:image-exif-data(Base64BinaryString *als string***)** als **element?** XP3.1 XQ3.1
Erhält ein Base64-kodiertes JPEG-Bild als Argument und gibt ein Element namens **Exif** zurück, das die Exif-Metadaten des Bilds enthält. Die Exif-Metadaten werden als Attribut-Wert-Paare des `Exif`-Elements erstellt. Bei den Attributnamen handelt es sich um die Exif-Daten-Tags aus der Base64-Kodierung. Weiter unten sehen Sie eine Liste der Exif-Tags. Wenn die Exif-Daten einen anbieterspezifischen Tag enthalten, so wird auch dieser Tag und sein Wert als Attribut-Wert-Paar zurückgegeben. Zusätzlich zu den Standard-Exif-Metadatentags (*siehe Liste unten*) werden auch Altova-spezifische Attribut-Wert-Paare generiert. Diese Altova Exif-Attribute sind unten aufgelistet.

⊟ *Beispiele*

- Um ein einziges Attribut abzurufen, verwenden Sie die Funktion folgendermaßen:
  `image-exif-data(//MyImages/Image20141130.01)/@GPSLatitude`
  `image-exif-data(//MyImages/Image20141130.01)/@Geolocation`
- Um alle Attribute abzurufen, verwenden Sie die Funktion folgendermaßen:
  `image-exif-data(//MyImages/Image20141130.01)/@*`
- Um die Namen aller Attribute abzurufen, verwenden Sie den folgenden Ausdruck:
  `for $i in image-exif-data(//MyImages/Image20141130.01)/@* return name($i)`
  Auf diese Art können Sie die Namen der von der Funktion zurückgegebenen Attribute eruieren.

⊟ *Altova Exif-Attribut: Geolocation*

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe*

© 2016-2022 Altova GmbH

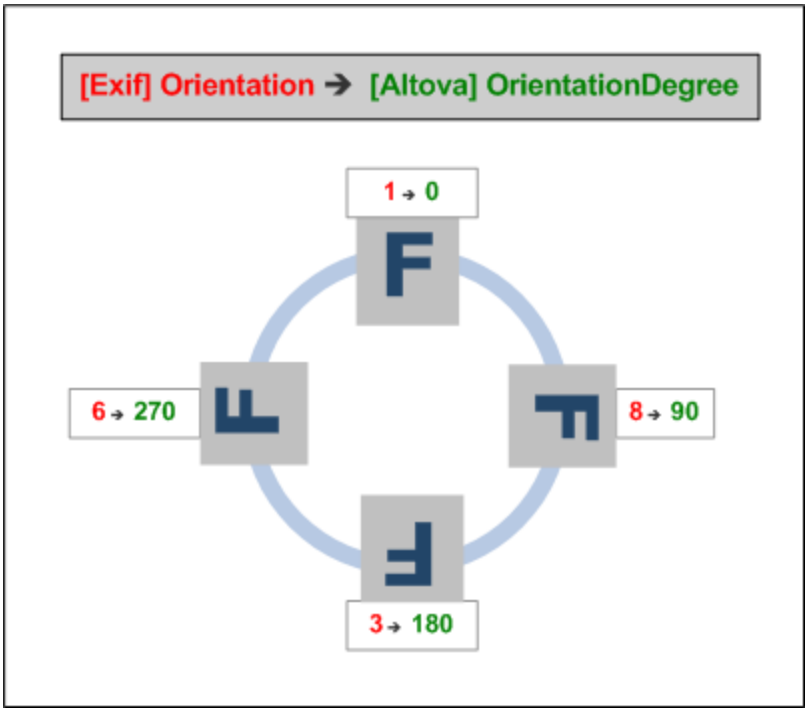
Tabelle unten).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

- *Altova Exif-Attribut: OrientationDegree*

  Der Altova XPath/XQuery-Prozessor generiert anhand des Exif-Metadaten-Tags **Orientation** das benutzerdefinierte Attribut **OrientationDegree**.

  OrientationDegree übersetzt den Standard-Exif-Tag `Orientation` von einem Ganzzahlwert (`1`, `8`, `3` oder `6`) in die entsprechenden Gradwerte dafür (`0`, `90`, `180`, `270`) (*siehe Abbildung unten*). Beachten Sie dass es keine Übersetzung der `Orientation`-Werte `2`, `4`, `5`, `7` gibt. (Diese Ausrichtungen werden durch Spiegelung des Bilds `1` an seiner senkrechten Mittelachse zur Erzeugung des Bilds mit dem Wert `2` und anschließende Drehung dieses Bilds um jeweils 90 Grad zur Erzeugung der Werte `7` bzw. `4` bzw. `5` erzielt).



- *Liste der Standard-Exif-Metatags*

  - `ImageWidth`
  - `ImageLength`
  - `BitsPerSample`
  - `Compression`
  - `PhotometricInterpretation`

- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription
- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode

© 2016-2022 Altova GmbH

- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio
- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing

- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

[Nach oben 2029]

19.2.2.1.5 XPath/XQuery-Funktionen: Numerische

Die numerischen Erweiterungsfunktionen von Altova können in XPath- und XQuery-Ausdrücken verwendet werden und stellen zusätzliche Funktionen für die Verarbeitung von Daten zur Verfügung. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0- und XQuery 3.0**-Prozessor von Altova verwendet werden. Sie stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

Funktionen zur automatischen Nummerierung

▼ generate-auto-number [altova:]

`altova:generate-auto-number(ID` *`als xs:string`*`, `**`StartsWith`** *`als xs:double`*`, `**`Increment`** *`als xs:double`*`, `**`ResetOnChange`** *`als xs:string`*`)` als `xs:integer` XP1 XP2 XQ1 XP3.1 XQ3.1
Generiert jedes Mal, wenn die Funktion aufgerufen wird, eine Zahl. Die erste Zahl, die beim ersten Aufruf der Funktion generiert wird, wird durch das Argument `StartsWith` definiert. Bei jedem erneuten Aufruf der Funktion wird eine neue Zahl generiert. Diese Zahl wird durch den im Argument `Increment` definierten Wert anhand der zuvor generierten Zahl inkrementiert. Auf diese Art erstellt die Funktion `altova:generate-auto-number` einen Zähler, dessen Name durch das Argument `ID` definiert wird und der jedes Mal, wenn die Funktion aufgerufen wird, inkrementiert wird. Wenn sich der Wert des Arguments `ResetOnChange` seit dem vorherigen Funktionsaufruf geändert hat, so wird der Wert der zu generierenden Zahl auf den Wert `StartsWith` zurückgesetzt. Die Automatische Nummerierung kann auch mit der

 © 2016-2022 Altova GmbH

Funktion `altova:reset-auto-number` zurückgesetzt werden.

Beispiele

- **`altova:generate-auto-number`**`("ChapterNumber", 1, 1, "SomeString")` gibt bei jedem Aufruf der Funktion eine einzige Zahl beginnend mit `1` zurück, die bei jedem Aufruf der Funktion um `1` inkrementiert wird. Solange das vierte Argument in jedem anschließenden Aufruf `"SomeString"` bleibt, wird die Inkrementierung fortgesetzt. Wenn sich der Wert des vierten Arguments ändert, wird der Zähler (namens `ChapterNumber`) auf 1 zurückgesetzt. Der Wert von `ChapterNumber` kann auch folgendermaßen durch Aufruf der Funktion `altova:reset-auto-number` zurückgesetzt werden: `altova:reset-auto-number("ChapterNumber")`.

reset-auto-number [altova:]

`altova:reset-auto-number(ID` *`als xs:string`***`)`** XP1 XP2 XQ1 XP3.1 XQ3.1
Diese Funktion setzt die Zahl des im `ID`-Argument angegebenen Zählers zur automatischen Nummerierung zurück. Die Zahl wird auf die Zahl zurückgesetzt, die durch das Argument `StartsWith` der Funktion `altova:generate-auto-number`, die den im `ID`-Argument genannten Zähler erstellt hat, definiert ist

Beispiele

- **`altova:reset-auto-number`**`("ChapterNumber")` setzt die Zahl des Zählers zur automatischen Nummerierung (`ChapterNumber`), der durch die Funktion `altova:generate-auto-number` erstellt wurde, zurück. Die Zahl wird auf den Wert des Arguments `StartsWith` der Funktion `altova:generate-auto-number`, die `ChapterNumber` erstellt hat, zurückgesetzt.

[Nach oben 2034]

Numerische Funktionen

hex-string-to-integer [altova:]

`altova:hex-string-to-integer(HexString` *`als xs:string`***`)`** als **`xs:integer`** XP3.1 XQ3.1
Verwendet ein String-Argument, das das Base-16-Äquivalent einer Ganzzahl im Dezimalsystem (Base-10) ist, und gibt die dezimale Ganzzahl zurück.

Beispiele

- **`altova:hex-string-to-integer`**`('1')` gibt `1` zurück
- **`altova:hex-string-to-integer`**`('9')` gibt `9` zurück
- **`altova:hex-string-to-integer`**`('A')` gibt `10` zurück
- **`altova:hex-string-to-integer`**`('B')` gibt `11` zurück
- **`altova:hex-string-to-integer`**`('F')` gibt `15` zurück
- **`altova:hex-string-to-integer`**`('G')` gibt einen Fehler zurück
- **`altova:hex-string-to-integer`**`('10')` gibt `16` zurück
- **`altova:hex-string-to-integer`**`('01')` gibt `1` zurück
- **`altova:hex-string-to-integer`**`('20')` gibt `32` zurück
- **`altova:hex-string-to-integer`**`('21')` gibt `33` zurück
- **`altova:hex-string-to-integer`**`('5A')` gibt `90` zurück
- **`altova:hex-string-to-integer`**`('USA')` gibt einen Fehler zurück

integer-to-hex-string [altova:]

altova:integer-to-hex-string(Integer *as xs:integer*) **as xs:string** XP3.1 XQ3.1
Verwendet ein Ganzzahlargument und gibt das Base-16-Äquivalent als String zurück.

Beispiele

- **altova:integer-to-hex-string**(1) gibt 1 zurück
- **altova:integer-to-hex-string**(9) gibt '9' zurück
- **altova:integer-to-hex-string**(10) gibt 'A' zurück
- **altova:integer-to-hex-string**(11) gibt 'B' zurück
- **altova:integer-to-hex-string**(15) gibt 'F' zurück
- **altova:integer-to-hex-string**(16) gibt '10' zurück
- **altova:integer-to-hex-string**(32) gibt '20' zurück
- **altova:integer-to-hex-string**(33) gibt '21' zurück
- **altova:integer-to-hex-string**(90) gibt '5A' zurück

[Nach oben 2034 **]**

Funktionen zur Formatierung von Zahlen

[nach oben]

19.2.2.1.6 XPath/XQuery-Funktionen: Schema

Die unten aufgelisteten Altova-Erweiterungsfunktionen geben Schemainformationen zurück. Weiter unten finden Sie Beschreibungen der Funktionen zusammen mit (i) Beispielen und (ii) einer Liste von Schemakomponenten und den dazugehörigen Eigenschaften. Diese Funktionen können mit dem **XPath 3.0-** und dem **XQuery 3.0-**Prozessor von Altova verwendet werden und stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Schemainformationen aus Schema-Dokumenten
Die Funktion **altova:schema** hat zwei Argumente: eines mit null Argumenten und das andere mit zwei Argumenten. Die Funktion mit null Argumenten gibt das gesamte Schema zurück. Sie können anschließend von diesem Ausgangspunkt aus durch das Schema navigieren und zu den gewünschten Schemakomponenten gehen. Die Funktion mit zwei Argumenten gibt eine bestimmte, durch Ihren QName identifizierte Komponentenart zurück. Der Rückgabewert ist in beiden Fällen eine Funktion. Um durch die zurückgegebene Komponente zu navigieren, müssen Sie eine Eigenschaft dieser spezifischen Komponente auswählen. Wenn es sich bei der Eigenschaft um ein nicht atomares Element handelt (d.h. wenn es sich um eine Komponente handelt), können Sie weiter navigieren, indem Sie eine Eigenschaft dieser Komponente auswählen. Wenn es sich bei der ausgewählten Eigenschaft um ein atomares Element handelt, wird der Wert des Elements zurückgegeben und Sie können nicht weiter navigieren.

Anmerkung: In XPath-Ausdrücken muss das Schema in die Verarbeitungsumgebung, z.B. in XSLT mit der Anweisung **xslt:import-schema** importiert werden. In XQuery-Ausdrücken muss das Schema explizit mit Hilfe eines Schemaimports importiert werden.

Schemainformationen aus XML-Nodes
Die Funktion **altova:type** übermittelt den Node eines XML-Dokuments und gibt die Typinformationen des Node aus dem PSVI zurück.

© 2016-2022 Altova GmbH

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XP1** **XP2** **XP3.1**
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XSLT1** **XSLT2** **XSLT3**
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	**XQ1** **XQ3.1**

▼ Schema (null Argumente)

`altova:schema()` **`als (function(xs:string) als item()*)?`** **XP3.1** **XQ3.1**
Gibt die **`schema`**-Komponente als ganzes zurück. Durch Auswahl einer der Eigenschaften der **`schema`**-Komponente können Sie weiter in die **`schema`**-Komponente navigieren.

- Wenn es sich bei dieser Eigenschaft um eine Komponente handelt, können Sie durch Auswahl einer dieser Komponenteneigenschaften einen Schritt tiefer navigieren. Dieser Schritt kann wiederholt werden, um weiter in das Schema zu navigieren.
- Wenn es sich bei der Komponente um einen atomaren Wert handelt, wird der atomare Wert zurückgegeben und Sie können nicht tiefer navigieren.

Die Eigenschaften der **`schema`**-Komponente sind:

```
"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"
```

Die Eigenschaften aller anderen Komponentenarten (neben **`schema`**) sind unten aufgelistet.

Anmerkung: In XQuery-Ausdrücken muss das Schema explizit importiert werden. In XPath-Ausdrücken muss das Schema in die Verarbeitungsumgebung, z.B. in XSLT mit der Anweisung **`xslt:import`** importiert worden sein.

▪ *Beispiele*

- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema()("type definitions")return $typedef ("name")` gibt die Namen aller simpleTypes oder complexTypes im Schema zurück.

- `import schema "" at "C:\Test\ExpReport.xsd";`
  `altova:schema() ("type definitions")[1]("name")` gibt den Namen des ersten aller simpleTypes oder complexTyps im Schema zurück.

Komponenten und ihre Eigenschaften

Assertion

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Assertion"
test	XPath-Eigenschaftsdatensatz	

Attribute Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Declaration"
name	String	Lokaler Name des Attributs
target namespace	String	Namespace URI des Attributs
type definition	SimpleType oder ComplexType	
scope	Eine Funktion mit Eigenschaften ("class":"Scope", "variety": "global" oder "local", "parent": der enthaltende ComplexTyp bzw. die enthaltende Attributgruppe)	
value constraint	Falls vorhanden, eine Funktion mit Eigenschaften ("class": "Value Constraint", "variety": "fixed" oder "default", "value": atomarer Wert, "lexical form": String. Beachten Sie, dass die Eigenschaft "value" für Namespace-sensitive Typen nicht zur Verfügung steht.	
inheritable	Boolean	

Attribute Group Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Group Definition"
name	String	Lokaler Name der Attributgruppe
target namespace	String	Namespace URI der Attributgruppe
attribute uses	Sequenz von (Attribute Use)	

© 2016-2022 Altova GmbH

attribute wildcard	Optionale Attribut-Wildcard	

Attribute Use

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Use"
required	Boolean	true, wenn das Attribut obligatorisch ist, false, wenn es optional ist.
value constraint	Siehe Attribute Declaration	
inheritable	Boolean	

Attribute Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	string	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": Sequenz von xs:anyURI, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

Complex Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
base type definition	Complex Type Definition	
final	String-Sequenz ("restriction"|"extension")	
context	Leere Sequenz (nicht implementiert)	
derivation method	String ("restriction"|"extension")	
abstract	Boolean	
attribute uses	Attribute Use-Sequenz	
attribute wildcard	Optionale Attribute Wildcard	
content type	Funktion mit Eigenschaften:	

	("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optionales Partikel, "open content": Funktion mit Eigenschaften ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	String-Sequenz ("restriction"|"extension")	
assertions	Assertion-Sequenz	

Element Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
type definition	Simple Type oder Complex Type	
type table	Funktion mit Eigenschaften ("class":"Type Table", "alternatives": Type Alternative-Sequenz, "default type definition": Simple Type oder Complex Type)	
scope	Funktion mit Eigenschaften ("class":"Scope", "variety": ("global"|"local"), "parent": optionaler Complex Type)	
value constraint	siehe Attribute Declaration	
nillable	Boolean	
identity-constraint definitions	Identity Constraint-Sequenz	
substitution group affiliations	Element Declaration-Sequenz	
substitution group exclusions	String-Sequenz ("restriction"|"extension")	
disallowed substitutions	String-Sequenz ("restriction"|"extension"|"substitution")	
abstract	Boolean	

Element Wildcard

© 2016-2022 Altova GmbH

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": xs:anyURI-Sequenz, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

- Facet

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	Der Name des Facet, z.B. "minLength" oder "enumeration"
value	abhängig vom Facet	Der Wert des Facet
fixed	Boolean	
typed-value	Nur für das Enumeration Facet, Array(xs:anyAtomicType*)	Ein Array, das Enumeration-Werte, von denen jeder im Allgemeinen eine Sequenz atomarer Werte sein kann, enthält. (Anmerkung: Die Eigenschaft "value" ist unabhängig vom tatsächlichen Typ für das Enumeration Facet eine String-Sequenz)

- Identity Constraint

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Identity-Constraint Definition"
name	String	Lokaler Name des Constraint
target namespace	String	Namespace URI des Constraint
identity-constraint category	String ("key"|"unique"|"keyRef")	
selector	XPath-Eigenschaftsdatensatz	
fields	Sequenz von XPath-Eigenschaftsdatensätzen	
referenced key	(nur für keyRef): Identity Constraint	Der entsprechende Key Constraint

- Model Group

Property name	Eigenschaftstyp	Eigenschaftswert

kind	String	"Model Group"
compositor	String ("sequence"|"choice"|"all")	
particles	Partikel-Sequenz	

Model Group Definition

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group Definition"
name	String	Lokaler Name der Model Group
target namespace	String	Namespace URI der Model Group
model group	Model Group	

Notation

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Notation Declaration"
name	String	Lokaler Name der Notation
target namespace	String	Namespace URI der Notation
system identifier	anyURI	
public identifier	String	

Particle

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Particle"
min occurs	Integer	
max occurs	Integer oder String("unbounded")	
term	Element Declaration, Element Wildcard oder ModelGroup	

Simple Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Simple Type Definition"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
final	String-Sequenz("restriction"|"extension"|"list"|"union")	

© 2016-2022 Altova GmbH

context	enthaltende Komponente	
base type definition	Simple Type	
facets	Facet-Sequenz	
fundamental facets	Leere Sequenz (nicht implementiert)	
variety	String ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(nur für Listentypen) Simple Type	
member type definitions	(nur für Union-Typen) Simple Type-Sequenz	

Type Alternative

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Type Alternative"
test	XPath-Eigenschaftsdatensatz	
type definition	Simple Type oder Complex Type	

XPath Property Record

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
namespace bindings	Sequenz von Funktionen mit Eigenschaften ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	Die statische Basis-UI des XPath-Ausdrucks
expression	String	Der XPath-Ausdruck als String

Schema (zwei Argumente)

altova:schema(ComponentKind *als xs:string*, **Name** *als xs:QName*) **als (function(xs:string) als item()*)? XP3.1 XQ3.1**

Gibt die im ersten Argument angegebene Komponentenart zurück, welche einen Namen hat, der mit dem im zweiten Argument angegebenen Namen übereinstimmt. Durch Auswahl einer der Eigenschaften der Komponente können Sie weiter navigieren.

- Wenn es sich bei dieser Eigenschaft um eine Komponente handelt, können Sie durch Auswahl einer dieser Komponenteneigenschaften einen Schritt tiefer navigieren. Dieser Schritt kann wiederholt werden, um weiter in das Schema zu navigieren.
- Wenn es sich bei der Komponente um einen atomaren Wert handelt, wird der atomare Wert zurückgegeben und Sie können nicht tiefer navigieren.

Anmerkung: In XQuery-Ausdrücken muss das Schema explizit importiert werden. In XPath-Ausdrücken muss das Schema in die Verarbeitungsumgebung, z.B. in XSLT mit der Anweisung **xslt:import** importiert worden sein.

Beispiele

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  altova:schema("element declaration", xs:QName("OrgChart"))("type definition")
  ("content type")("particles")[3]!.("term")("kind")
  ```
  gibt die `kind`-Eigenschaft des Terms der dritten `particles`-Komponente zurück. Diese particles-Komponente ist ein Nachfahr der Element-Deklaration mit dem QName `QName OrgChart`.

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  let $typedef := altova:schema("type definition", xs:QName("emailType"))
  for $facet in $typedef ("facets")
  return [$facet ("kind"), $facet("value")]
  ```
  gibt für jedes **facet** jeder **emailType**-Komponente ein Array mit der Art und dem Wert des Facet zurück.

Komponenten und ihre Eigenschaften

Assertion

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Assertion"
test	XPath-Eigenschaftsdatensatz	

Attribute Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Declaration"
name	String	Lokaler Name des Attributs
target namespace	String	Namespace URI des Attributs
type definition	SimpleType oder ComplexType	
scope	Eine Funktion mit Eigenschaften ("class":"Scope", "variety": "global" oder "local", "parent": der enthaltende ComplexTyp bzw. die enthaltende Attributgruppe)	
value constraint	Falls vorhanden, eine Funktion mit Eigenschaften ("class": "Value Constraint", "variety": "fixed" oder "default", "value": atomarer Wert, "lexical form": String. Beachten Sie, dass die Eigenschaft "value" für Namespace-sensitive Typen nicht zur Verfügung steht.	

© 2016-2022 Altova GmbH

inheritable	Boolean	

Attribute Group Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Group Definition"
name	String	Lokaler Name der Attributgruppe
target namespace	String	Namespace URI der Attributgruppe
attribute uses	Sequenz von (Attribute Use)	
attribute wildcard	Optionale Attribut-Wildcard	

Attribute Use

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Use"
required	Boolean	true, wenn das Attribut obligatorisch ist, false, wenn es optional ist.
value constraint	Siehe Attribute Declaration	
inheritable	Boolean	

Attribute Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	string	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": Sequenz von xs:anyURI, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

Complex Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)

base type definition	Complex Type Definition	
final	String-Sequenz ("restriction"|"extension")	
context	Leere Sequenz (nicht implementiert)	
derivation method	String ("restriction"|"extension")	
abstract	Boolean	
attribute uses	Attribute Use-Sequenz	
attribute wildcard	Optionale Attribute Wildcard	
content type	Funktion mit Eigenschaften: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optionales Partikel, "open content": Funktion mit Eigenschaften ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	String-Sequenz ("restriction"|"extension")	
assertions	Assertion-Sequenz	

Element Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
type definition	Simple Type oder Complex Type	
type table	Funktion mit Eigenschaften ("class":"Type Table", "alternatives": Type Alternative-Sequenz, "default type definition": Simple Type oder Complex Type)	
scope	Funktion mit Eigenschaften ("class":"Scope", "variety": ("global"|"local"), "parent": optionaler Complex Type)	
value constraint	siehe Attribute Declaration	
nillable	Boolean	
identity-constraint	Identity Constraint-Sequenz	

© 2016-2022 Altova GmbH

definitions		
substitution group affiliations	Element Declaration-Sequenz	
substitution group exclusions	String-Sequenz ("restriction"|"extension")	
disallowed substitutions	String-Sequenz ("restriction"|"extension"|"substitution")	
abstract	Boolean	

- Element Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": xs:anyURI-Sequenz, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

- Facet

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	Der Name des Facet, z.B. "minLength" oder "enumeration"
value	abhängig vom Facet	Der Wert des Facet
fixed	Boolean	
typed-value	Nur für das Enumeration Facet, Array(xs:anyAtomicType*)	Ein Array, das Enumeration-Werte, von denen jeder im Allgemeinen eine Sequenz atomarer Werte sein kann, enthält. (Anmerkung: Die Eigenschaft "value" ist unabhängig vom tatsächlichen Typ für das Enumeration Facet eine String-Sequenz)

- Identity Constraint

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Identity-Constraint Definition"
name	String	Lokaler Name des Constraint
target namespace	String	Namespace URI des Constraint

identity-constraint category	String ("key"|"unique"|"keyRef")	
selector	XPath-Eigenschaftsdatensatz	
fields	Sequenz von XPath-Eigenschaftsdatensätzen	
referenced key	(nur für keyRef): Identity Constraint	Der entsprechende Key Constraint

Model Group

Property name	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group"
compositor	String ("sequence"|"choice"|"all")	
particles	Partikel-Sequenz	

Model Group Definition

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group Definition"
name	String	Lokaler Name der Model Group
target namespace	String	Namespace URI der Model Group
model group	Model Group	

Notation

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Notation Declaration"
name	String	Lokaler Name der Notation
target namespace	String	Namespace URI der Notation
system identifier	anyURI	
public identifier	String	

Particle

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Particle"
min occurs	Integer	
max occurs	Integer oder String("unbounded")	
term	Element Declaration, Element Wildcard oder ModelGroup	

Simple Type

© 2016-2022 Altova GmbH

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Simple Type Definition"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
final	String-Sequenz("restriction"|"extension"|"list"|"union")	
context	enthaltende Komponente	
base type definition	Simple Type	
facets	Facet-Sequenz	
fundamental facets	Leere Sequenz (nicht implementiert)	
variety	String ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(nur für Listentypen) Simple Type	
member type definitions	(nur für Union-Typen) Simple Type-Sequenz	

Type Alternative

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Type Alternative"
test	XPath-Eigenschaftsdatensatz	
type definition	Simple Type oder Complex Type	

XPath Property Record

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
namespace bindings	Sequenz von Funktionen mit Eigenschaften ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	Die statische Basis-UI des XPath-Ausdrucks
expression	String	Der XPath-Ausdruck als String

Typ

altova:type(Node *als item?***) als (function(xs:string) als item()*)?** XP3.1 XQ3.1

Die Funktion **altova:type** übermittelt einen Element- oder Attribut-Node eines XML-Dokuments und gibt die Typinformationen des Node aus dem PSVI zurück.

Anmerkung: Das XML-Dokument muss eine Schema-Deklaration haben, damit das Schema referenziert werden kann.

- *Beispiele*

  - ```
    for $element in //Email
    let $type := altova:type($element)
    return $type
    ```
    gibt eine Funktion zurück, die die Typinformationen des Node `Email` enthält.

  - ```
    for $element in //Email
    let $type := altova:type($element)
    return $type ("kind")
    ```
    ermittelt anhand der Typ-Komponente des Node (Simple Type oder Complex Type) den Wert der Eigenschaft **kind** der Komponente.

  Der Parameter **"_props"** gibt die Eigenschaften der ausgewählten Komponente zurück, z.B:

  - ```
    for $element in //Email
    let $type := altova:type($element)
    return ($type ("kind"), $type ("_props"))
    ```
    nimmt die Typkomponente des Node `Email` (Simple Type oder Complex Type) und gibt (i) den Wert der Eigenschaft **kind** der Komponente zurück und anschließend (ii) die Eigenschaften dieser Komponente.

Komponenten und ihre Eigenschaften

- Assertion

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Assertion"
test	XPath-Eigenschaftsdatensatz	

- Attribute Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Declaration"
name	String	Lokaler Name des Attributs
target namespace	String	Namespace URI des Attributs
type definition	SimpleType oder ComplexType	
scope	Eine Funktion mit Eigenschaften ("class":"Scope", "variety": "global" oder "local", "parent": der enthaltende ComplexTyp bzw. die enthaltende Attributgruppe)	

© 2016-2022 Altova GmbH
    ```

value constraint	Falls vorhanden, eine Funktion mit Eigenschaften ("class": "Value Constraint", "variety": "fixed" oder "default", "value": atomarer Wert, "lexical form": String. Beachten Sie, dass die Eigenschaft "value" für Namespace-sensitive Typen nicht zur Verfügung steht.	
inheritable	Boolean	

- Attribute Group Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Group Definition"
name	String	Lokaler Name der Attributgruppe
target namespace	String	Namespace URI der Attributgruppe
attribute uses	Sequenz von (Attribute Use)	
attribute wildcard	Optionale Attribut-Wildcard	

- Attribute Use

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Use"
required	Boolean	true, wenn das Attribut obligatorisch ist, false, wenn es optional ist.
value constraint	Siehe Attribute Declaration	
inheritable	Boolean	

- Attribute Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	string	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": Sequenz von xs:anyURI, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

- Complex Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
base type definition	Complex Type Definition	
final	String-Sequenz ("restriction"|"extension")	
context	Leere Sequenz (nicht implementiert)	
derivation method	String ("restriction"|"extension")	
abstract	Boolean	
attribute uses	Attribute Use-Sequenz	
attribute wildcard	Optionale Attribute Wildcard	
content type	Funktion mit Eigenschaften: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optionales Partikel, "open content": Funktion mit Eigenschaften ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	String-Sequenz ("restriction"|"extension")	
assertions	Assertion-Sequenz	

Element Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
type definition	Simple Type oder Complex Type	
type table	Funktion mit Eigenschaften ("class":"Type Table", "alternatives": Type Alternative-Sequenz, "default type definition": Simple Type oder Complex Type)	

© 2016-2022 Altova GmbH

scope	Funktion mit Eigenschaften ("class":"Scope", "variety": ("global"|"local"), "parent": optionaler Complex Type)	
value constraint	siehe Attribute Declaration	
nillable	Boolean	
identity-constraint definitions	Identity Constraint-Sequenz	
substitution group affiliations	Element Declaration-Sequenz	
substitution group exclusions	String-Sequenz ("restriction"|"extension")	
disallowed substitutions	String-Sequenz ("restriction"|"extension"|"substitution")	
abstract	Boolean	

- Element Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": xs:anyURI-Sequenz, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

- Facet

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	Der Name des Facet, z.B. "minLength" oder "enumeration"
value	abhängig vom Facet	Der Wert des Facet
fixed	Boolean	
typed-value	Nur für das Enumeration Facet, Array(xs:anyAtomicType*)	Ein Array, das Enumeration-Werte, von denen jeder im Allgemeinen eine Sequenz atomarer Werte sein kann, enthält. (Anmerkung: Die Eigenschaft "value" ist unabhängig vom tatsächlichen Typ für das Enumeration Facet eine String-Sequenz)

Identity Constraint

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Identity-Constraint Definition"
name	String	Lokaler Name des Constraint
target namespace	String	Namespace URI des Constraint
identity-constraint category	String ("key"|"unique"|"keyRef")	
selector	XPath-Eigenschaftsdatensatz	
fields	Sequenz von XPath-Eigenschaftsdatensätzen	
referenced key	(nur für keyRef): Identity Constraint	Der entsprechende Key Constraint

Model Group

Property name	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group"
compositor	String ("sequence"|"choice"|"all")	
particles	Partikel-Sequenz	

Model Group Definition

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group Definition"
name	String	Lokaler Name der Model Group
target namespace	String	Namespace URI der Model Group
model group	Model Group	

Notation

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Notation Declaration"
name	String	Lokaler Name der Notation
target namespace	String	Namespace URI der Notation
system identifier	anyURI	
public identifier	String	

Particle

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert

© 2016-2022 Altova GmbH

kind	String	"Particle"
min occurs	Integer	
max occurs	Integer oder String("unbounded")	
term	Element Declaration, Element Wildcard oder ModelGroup	

Simple Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Simple Type Definition"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
final	String-Sequenz("restriction"|"extension"|"list"|"union")	
context	enthaltende Komponente	
base type definition	Simple Type	
facets	Facet-Sequenz	
fundamental facets	Leere Sequenz (nicht implementiert)	
variety	String ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(nur für Listentypen) Simple Type	
member type definitions	(nur für Union-Typen) Simple Type-Sequenz	

Type Alternative

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Type Alternative"
test	XPath-Eigenschaftsdatensatz	
type definition	Simple Type oder Complex Type	

XPath Property Record

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
namespace bindings	Sequenz von Funktionen mit Eigenschaften ("prefix": string, "namespace": anyURI)	

default namespace	anyURI	
base URI	anyURI	Die statische Basis-UI des XPath-Ausdrucks
expression	String	Der XPath-Ausdruck als String

19.2.2.1.7 XPath/XQuery-Funktionen: Sequenz

Die Sequenz-Erweiterungsfunktionen von Altova können in XPath- und XQuery-Ausdrücken verwendet werden und stellen zusätzliche Funktionen für die Verarbeitung von Daten zur Verfügung. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0-** und **XQuery 3.0**-Prozessor von Altova verwendet werden. Sie stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

▼ attributes [altova:]

`altova:attributes(AttributeName` *`als xs:string`***`)`** als **`attribute()*`** XP3.1 XQ3.1
Gibt alle Attribute zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `AttributeName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `attribute::` Achse durchgeführt wird, beachtet. Das bedeutet, dass der Kontext-Node der Parent-Element-Node sein muss.

▣ *Beispiele*

- **`altova:attributes`**`("MyAttribute")` gibt `MyAttribute()*` zurück

`altova:attributes(AttributeName` *`als xs:string`***`, SearchOptions`** *`als xs:string`***`)`** als **`attribute()*`** XP3.1 XQ3.1
Gibt alle Attribute zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `AttributeName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `attribute::` Achse durchgeführt wird, beachtet. Der Kontext-Node muss der Parent-Element-

© 2016-2022 Altova GmbH

Node sein. Das zweite Argument ist ein String, der Options-Flags enthält. Zur Verfügung stehen die folgenden Flags:
r = wechselt zu einer Suche mittels Regular Expression; bei `AttributeName` muss es sich in diesem Fall um einen Regular Expression-Suchstring handeln;
f = Wenn diese Option definiert ist, liefert `AttributeName` eine vollständige Übereinstimmung; andernfalls muss `AttributeName` nur teilweise mit einem Attributnamen übereinstimmen, damit dieses Attribut zurückgegeben wird. Wenn **f** z.B. nicht definiert ist, gibt `MyAtt MyAttribute` zurück;
i = wechselt zu einer Suche ohne Berücksichtigung der Groß- und Kleinschreibung;
p = inkludiert das Namespace-Präfix in die Suche; `AttributeName` sollte in diesem Fall das Namespace-Präfix enthalten, z.B.: `altova:MyAttribute`.
Die Flags können in jeder Reihenfolge angegeben werden. Ungültige Flags erzeugen eine Fehlermeldung. Sie können ein oder mehrere Flags weglassen. Es ist auch der leere String zulässig. Das Resultat ist dasselbe wie bei Verwendung der Funktion mit nur einem Argument (*siehe vorherige Signatur*). Unzulässig ist jedoch die Verwendung einer leeren Sequenz als zweites Argument.

Beispiele

- **altova:attributes**("MyAttribute", "rfip") gibt `MyAttribute()*` zurück
- **altova:attributes**("MyAttribute", "pri") gibt `MyAttribute()*` zurück
- **altova:attributes**("MyAtt", "rip") gibt `MyAttribute()*` zurück
- **altova:attributes**("MyAttributes", "rfip") gibt keine Übereinstimmung zurück
- **altova:attributes**("MyAttribute", "") gibt `MyAttribute()*` zurück
- **altova:attributes**("MyAttribute", "Rip") gibt einen Fehler zurück, dass das Flag unbekannt ist.
- **altova:attributes**("MyAttribute",) gibt den Fehler zurück, dass das zweite Argument fehlt.

elements [altova:]

altova:elements(ElementName *als xs:string***)** als **element()*** XP3.1 XQ3.1
Gibt alle Elemente zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `ElementName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `child::` Achse durchgeführt wird, beachtet. Der Kontext-Node muss der Parent-Node des gesuchten Elements sein.

Beispiele

- **altova:elements**("MyElement") gibt `MyElement()*` zurück

altova:elements(ElementName *als xs:string*, **SearchOptions** *als xs:string***)** als **element()*** XP3.1 XQ3.1
Gibt alle Elemente zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `ElementName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `child::` Achse durchgeführt wird, beachtet. Der Kontext-Node muss der Parent-Node des gesuchten Elements sein. Das zweite Argument ist ein String, der Options-Flags enthält. Zur Verfügung stehen die folgenden Flags:
r = wechselt zu einer Suche mittels Regular Expression; bei `ElementName` muss es sich in diesem Fall um einen Regular Expression-Suchstring handeln;
f = Wenn diese Option definiert ist, liefert `ElementName` eine vollständige Übereinstimmung; andernfalls muss `ElementName` nur teilweise mit einem Elementnamen übereinstimmen, damit dieses Element zurückgegeben wird. Wenn **f** z.B. nicht definiert ist, gibt `MyElem MyElement` zurück;
i = wechselt zu einer Suche ohne Berücksichtigung der Groß- und Kleinschreibung;
p = inkludiert das Namespace-Präfix in die Suche; `ElementName` sollte in diesem Fall das Namespace-Präfix enthalten, z.B.: `altova:MyElement`.

Die Flags können in jeder Reihenfolge angegeben werden. Ungültige Flags erzeugen eine Fehlermeldung. Sie können ein oder mehrere Flags weglassen. Es ist auch der leere String zulässig. Das Resultat ist dasselbe wie bei Verwendung der Funktion mit nur einem Argument (siehe vorherige Signatur). Unzulässig ist jedoch die Verwendung einer leeren Sequenz.

Beispiele

- **altova:elements**("MyElement", "rip") gibt MyElement()* zurück
- **altova:elements**("MyElement", "pri") gibt MyElement()* zurück
- **altova:elements**("MyElement", "") gibt MyElement()* zurück
- **altova:elements**("MyElem", "rip") gibt MyElement()* zurück
- **altova:elements**("MyElements", "rfip") gibt keine Übereinstimmung zurück
- **altova:elements**("MyElement", "Rip") gibt einen Fehler zurück, dass das Flag unbekannt ist.
- **altova:elements**("MyElement",) gibt den Fehler zurück, dass das zweite Argument fehlt.

find-first [altova:]

altova:find-first((*item()**)**, (CheckFunction(item()** *als xs:boolean***))** als **item()?** XP3.1 XQ3.1

Diese Funktion verwendet zwei Argumente. Das erste Argument ist eine Sequenz von einem oder mehreren Elementen eines beliebigen Datentyps. Das zweite Argument, Condition, ist eine Referenz zu einer XPath-Funktion, die ein Argument erhält. (hat einen Stellenwert 1) und einen Booleschen Wert zurückgibt. Jedes Element von **Sequence** wird der Reihe nach der in Condition referenzierten Funktion bereitgestellt. (*Beachten Sie:* Die Funktion hat ein einziges Argument.) Das erste **Sequence** Element, bei dem das Resultat von **Condition true()** ist, wird als das Ergebnis von altova:find-first zurückgegeben. Anschließend wird die Iteration gestoppt.

Beispiele

- **altova:find-first**(5 to 10, function($a) {$a mod 2 = 0}) gibt xs:integer 6 zurück

Das Argument **Condition** referenziert die XPath 3.0 Inline-Funktion, **function()**, welche eine Inline-Funktion **$a** deklariert und diese anschließend definiert. Die einelnen Elemente im Argument **Sequence** von **altova:find-first** werden der Reihe nach an **$a** als sein Input-Wert übergeben. Der Input-Wert wird an der Bedingung in der Funktionsdefinition ($a mod 2 = 0) überprüft. Der erste Input-Wert, der diese Bedingung erfüllt, wird als das Ergebnis von **altova:find-first** (in diese Fall **6**) zurückgegeben.

- **altova:find-first**((1 to 10), (function($a) {$a+3=7})) gibt xs:integer 4 zurück

Weitere Beispiele

Wenn die Datei **C:\Temp\Customers.xml** vorhanden ist:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt C:\Temp\Customers.xml zurück

Wenn die Datei **C:\Temp\Customers.xml** nicht vorhanden ist und **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt http://www.altova.com/index.html zurück

Wenn weder die Datei **C:\Temp\Customers.xml** noch **http://www.altova.com/index.html** vorhanden ist:

© 2016-2022 Altova GmbH

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt kein Ergebnis zurück

Anmerkungen zu den obigen Beispielen

- Die XPath 3.0-Funktion, doc-available, erhält ein einziges Argument, das als URI verwendet wird. Sie gibt nur dann true zurück, wenn unter der angegebenen URI ein Dokument-Node gefunden wird. Das Dokument unter der angegebenen URI muss daher ein XML-Dokument sein.
- Die Funktion doc-available kann für **Condition,** das zweite Argument von altova:find-first verwendet werden, da sie nur ein Argument erhält (Stelligkeit=1), da sie ein Element item() als Input erhält (ein String, der als URI verwendet wird) und einen Booleschen Wert zurückgibt.
- Beachten Sie, dass doc-available nur referenziert und nicht direkt aufgerufen wird. Das angehängte Suffix #1 gibt eine Funktion mit einer Stelligkeit 1 an. Als Ganzes bedeutet doc-available#1: *Verwende die Funktion doc-availabe(), welche die Stelligkeit=1 hat und übergib die einzelnen Elemente in der ersten Sequenz der Reihe nach als einziges Argument an die Funktion.* Als Ergebnis wird jeder der beiden Strings an **doc-available()** übergeben. Die Funktion verwendet den String als URI und überprüft, ob unter der URI ein Dokument-Node vorhanden ist. Wenn dies der Fall ist, wird **doc-available()** zu true() ausgewertet und der String wird als Ergebnis der Funktion **altova:find-first** zurückgegeben. *Beachten Sie zur Funktion doc-available(), dass relative Pfade relativ zu aktuellen Basis-URI aufgelöst werden. Die Basis-URI ist standardmäßig die URI des XML-Dokuments, von dem aus die Funktion geladen wird.*

▼ find-first-combination [altova:]

altova:find-first-combination((Seq-01 *as item()**)**,* **(Seq-02** *as item()**)**,*
(Condition(Seq-01-Item, Seq-02-Item *as xs:boolean***)) as item()*** XP3.1 XQ3.1
Diese Funktion verwendet drei Argumente:

- Die ersten beiden Argumente, **Seq-01** und **Seq-02**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.
- Das dritte Argument, **Condition**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit **2** hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **Seq-01** und **Seq-02** werden als die Argumente der Funktion **Condition** in geordneten Paaren übergeben (je ein Element aus jeder Sequenz bildet ein Paar). Die Paare sind folgendermaßen geordnet.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

Das erste geordnete Paar, bei dem die Funktion **Condition** zu **true()** ausgewertet wird, wird als Ergebnis von **altova:find-first-combination** zurückgegeben. Beachten Sie: (i) Wenn die Funktion **Condition** durch die bereitgestellten Argumentpaare iteriert und nicht ein einziges Mal zu **true()** ausgewertet wird, so gibt **altova:find-first-combination** *Keine Ergebnisse* zurück; (ii) Das Ergebnis von **altova:find-first-combination** ist immer ein Elementpaar (eines beliebigen Datentyps) oder gar kein Element.

Beispiele

- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) gibt die Sequenz xs:integers (11, 21) zurück
- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) gibt die Sequenz xs:integers (11, 22) zurück
- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) gibt die Sequenz xs:integers (11, 23) zurück

find-first-pair [altova:]

altova:find-first-pair((Seq-01 *als item()**, **(Seq-02** *als item()**, **(Condition(Seq-01-Item, Seq-02-Item** *als xs:boolean***))** als **item()*** XP3.1 XQ3.1
Diese Funktion erhält drei Argumente:

- Die ersten beiden Argumente, **Seq-01** und **Seq-02**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.
- Das dritte Argument, **Condition**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit **2** hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **Seq-01** und **Seq-02** werden als die Argumente der Funktion **Condition** in geordneten Paaren übergeben. Die Paare sind folgendermaßen geordnet.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Das erste geordnete Paar, bei dem die Funktion **Condition** zu **true()** ausgewertet wird, wird als Ergebnis von **altova:find-first-pair** zurückgegeben. Beachten Sie: (i) Wenn die Funktion **Condition** durch die bereitgestellten Argumentpaare iteriert und nicht ein einziges Mal zu **true()** ausgewertet wird, so gibt **altova:find-first-pair** *Keine Ergebnisse* zurück; (ii) Das Ergebnis von **altova:find-first-pair** ist immer ein Elementpaar (eines beliebigen Datentyps) oder gar kein Element.

Beispiele

- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) gibt die Sequenz xs:integers (11, 21) zurück
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) gibt *Keine Ergebnisse* zurück

Beachten Sie anhand der zwei Beispiel oben, dass die Paare folgendermaßen geordnet sind: (11, 21) (12, 22) (13, 23)...(20, 30). Aus diesem Grund gibt das zweite Beispiel *Keine Ergebnisse* zurück (da keine geordnetes Paar die Summe 33 ergibt).

find-first-pair-pos [altova:]

altova:find-first-pair-pos((Seq-01 *as item()**, **(Seq-02** *as item()**, **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean***))** **as xs:integer** XP3.1 XQ3.1
Diese Funktion erhält drei Argumente:

© 2016-2022 Altova GmbH

- Die ersten beiden Argumente, **Seq-01** und **Seq-02**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.
- Das dritte Argument, **Condition**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit **2** hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **Seq-01** und **Seq-02** werden als die Argumente der Funktion **Condition** in geordneten Paaren übergeben. Die Paare sind folgendermaßen geordnet.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Als Ergebnis von **altova:find-first-pair-pos** wird die Indexposition des ersten geordneten Paars, bei dem die Funktion **Condition** zu **true()** ausgewertet wird, zurückgegeben. Beachten Sie: Wenn die Funktion **Condition** durch die bereitgestellten Argumentpaare iteriert und kein einziges Mal zu **true()** ausgewertet wird, so gibt **altova:find-first-pair-pos** *Keine Ergebnisse* zurück.

Beispiele

- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) gibt 1 zurück
- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) gibt *Keine Ergebnisse* zurück

Beachten Sie anhand der zwei Beispiel oben, dass die Paare folgendermaßen geordnet sind: (11, 21) (12, 22) (13, 23)...(20, 30). Im ersten Beispiel gibt die Funktion **Condition** bei Auswertung des ersten Paars **true()** zurück, daher wird dessen Indexposition in der Sequenz, 1, zurückgegeben. Das zweite Beispiel gibt *Keine Ergebnisse* zurück (da keine geordnetes Paar die Summe 33 ergibt).

find-first-pos [altova:]

altova:find-first-pos((*item()**), **(CheckFunction(item()** *als xs:boolean*) **)** als **xs:integer?** XP3.1 XQ3.1

Diese Funktion verwendet zwei Argumente. Das erste Argument ist eine Sequenz von einem oder mehreren Elementen eines beliebigen Datentyps. Das zweite Argument, Condition, ist eine Referenz zu einer XPath-Funktion, die ein Argument erhält. (hat einen Stellenwert 1) und einen Booleschen Wert zurückgibt. Jedes Element von **Sequence** wird der Reihe nach der in Condition referenzierten Funktion bereitgestellt. (*Beachten Sie:* Die Funktion hat ein einziges Argument.) Das erste **Sequence** Element, bei dem das Resultat von **Condition true()** ist, wird als das Ergebnis von **altova:find-first-pos** zurückgegeben. Anschließend wird die Iteration gestoppt.

Beispiele

- **altova:find-first-pos**(5 to 10, function($a) {$a mod 2 = 0}) gibt xs:integer 2 zurück

Das Argument **Condition** referenziert die XPath 3.0 Inline-Funktion, **function()**, welche eine Inline-Funktion **$a** deklariert und diese anschließend definiert. Die einelnen Elemente im Argument **Sequence** von **altova:find-first-pos** werden der Reihe nach an **$a** als sein Input-Wert übergeben. Der Input-Wert wird an der Bedingung in der Funktionsdefinition ($a mod 2 = 0) überprüft. Die Indexposition in der Sequenz des ersten Input-Werts, die diese Bedingung erfüllt, wird als das Ergebnis von **altova:find-first-pos** zurückgegeben (in diesem Fall 2, da **6,** der erste

Wert in der Sequenz, der die Bedingung erfüllt, sich in der Sequenz an der Indexposition 2 befindet).

Weitere Beispiele
Wenn die Datei **C:\Temp\Customers.xml** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt 1 zurück

Wenn die Datei **C:\Temp\Customers.xml** nicht vorhanden ist und **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt 2 zurück

Wenn weder die Datei **C:\Temp\Customers.xml** noch **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt kein Ergebnis zurück

Anmerkungen zu den obigen Beispielen

- Die XPath 3.0-Funktion, doc-available, erhält ein einziges Argument, das als URI verwendet wird. Sie gibt nur dann true zurück, wenn unter der angegebenen URI ein Dokument-Node gefunden wird. (Das Dokument unter der angegebenen URI muss daher ein XML-Dokument sein.)
- Die Funktion doc-available kann für **Condition,** das zweite Argument von altova:find-first-pos verwendet werden, da sie nur ein Argument erhält (Stelligkeit=1), da sie ein Element item() als Input erhält (ein String, der als URI verwendet wird) und einen Booleschen Wert zurückgibt.
- Beachten Sie, dass doc-available nur referenziert und nicht direkt aufgerufen wird. Das angehängte Suffix #1 gibt eine Funktion mit einer Stelligkeit 1 an. Als Ganzes bedeutet doc-available#1: *Verwende die Funktion doc-availabe(), welche die Stelligkeit=1 hat und übergib die einzelnen Elemente in der ersten Sequenz der Reihe nach als einziges Argument an die Funktion.* Als Ergebnis wird jeder der beiden Strings an **doc-available()** übergeben. Die Funktion verwendet den String als URI und überprüft, ob unter der URI ein Dokument-Node vorhanden ist. Wenn dies der Fall ist, wird **doc-available()** zu true() ausgewertet und der String wird als Ergebnis der Funktion **altova:find-first** zurückgegeben. *Beachten Sie zur Funktion doc-available(), dass relative Pfade relativ zu aktuellen Basis-URI aufgelöst werden. Die Basis-URI ist standardmäßig die URI des XML-Dokuments, von dem aus die Funktion geladen wird.*

▼ for-each-attribute-pair [altova:]

altova:for-each-attribute-pair(Seq1 *als element()?*, **Seq2** *als element()?*, **Function** *als function()*) als **item()*** XP3.1 XQ3.1
Die beiden ersten Argumente identifizieren zwei Elemente, anhand deren Attribute Attributpaare gebildet werden, wobei das eine Attribut eines Paars aus dem ersten Element und das andere aus dem zweiten Element stammt. Die Attributpaare werden auf Basis ihres übereinstimmenden Namens ausgewählt und alphabetisch (nach ihren Namen) zu einer Gruppe geordnet. Falls es zu einem Attribut im anderen

© 2016-2022 Altova GmbH

Element keine Entsprechung gibt, ist das Paar "nicht verbunden", d.h. es besteht nur aus einem Mitglied. Das Funktionselement (das dritte Argument `Function`) wird auf die einzelnen Paare (verbundene und nicht verbundene) in der Sequenz der Paare separat angewendet, wodurch als Ausgabe eine Sequenz von Einträgen erzeugt wird.

Beispiele

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, function($a, $b) {$a+b}) gibt zurück ...

  (2, 4, 6) wenn
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (2, 4, 6) wenn
  <Test-A att2="2" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  (2, 6) wenn
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  Anmerkung: *Das Ergebnis* **(2, 6)** *wird mit Hilfe der folgenden Aktion ermittelt:* **(1+1, ()+2, 3+3, 4+())**. *Wenn einer der Operanden eine leere Sequenz ist, wie dies bei Eintrag 2 und 4 der Fall ist, so ist das Ergebnis der Addition eine leere Sequenz.*

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, concat#2) gibt zurück ...

  (11, 22, 33) wenn
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (11, 2, 33, 4) wenn
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

for-each-combination [altova:]

altova:for-each-combination(FirstSequence *als item()**, **SecondSequence** *als item()**, **Function($i,$j){$i || $j})** als **item()*** XP3.1 XQ3.1

Die Elemente der zwei Sequenzen in den ersten beiden Argumenten werden miteinander kombiniert, so dass jedes Element in der ersten Sequenz der Reihe nach einmal mit jedem Element in der zweiten Sequenz kombiniert wird. Die als drittes Argument angegebene Funktion wird auf die einzelnen Kombinationen in der erzeugten Sequenz angewendet, wodurch als Ausgabe eine Sequenz von Elementen erzeugt wird (*siehe Beispiel*).

Beispiele

- **altova:for-each-combination**(('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j}) gibt ('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3') zurück

u

▼ for-each-matching-attribute-pair [altova:]

altova:for-each-matching-attribute-pair(Seq1 *als element()?*, **Seq2** *als element()?*, **Function** *als function()*) als **item()*** XP3.1 XQ3.1

Die beiden ersten Argumente identifizieren zwei Elemente, anhand deren Attribute Attributpaare gebildet werden, wobei das eine Attribut eines Paars aus dem ersten Element und das andere aus dem zweiten Element stammt. Die Attributpaare werden auf Basis ihres übereinstimmenden Namens ausgewählt und alphabetisch (nach ihren Namen) zu einer Gruppe geordnet. Falls es zu einem Attribut im anderen Element keine Entsprechung gibt, wird kein Paar gebildet. Das Funktionselement (das dritte Argument `Function`) wird auf die einzelnen Paare in der Sequenz der Paare separat angewendet, wodurch als Ausgabe eine Sequenz von Einträgen erzeugt wird.

⊟ *Beispiele*

- **altova:for-each-matching-attribute-pair**(/Example/Test-A, /Example/Test-B, function($a, $b){$a+b}) gibt zurück ...

  (2, 4, 6) wenn
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (2, 4, 6) wenn
  <Test-A att2="2" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  (2, 6) wenn
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att3="1" />

- **altova:for-each-matching-attribute-pair**(/Example/Test-A, /Example/Test-B, concat#2) gibt zurück ....

  (11, 22, 33) wenn
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (11, 33) wenn
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

▼ substitute-empty [altova:]

altova:substitute-empty(FirstSequence *als item()**, **SecondSequence** *als item()*) als **item()*** XP3.1 XQ3.1

Wenn `FirstSequence` leer ist, wird `SecondSequence` zurückgegeben. Wenn `FirstSequence` nicht leer ist, wird `FirstSequence` zurückgegeben.

⊟ *Beispiele*

- **altova:substitute-empty**((1,2,3), (4,5,6)) gibt (1,2,3) zurück
- **altova:substitute-empty**((), (4,5,6)) gibt (4,5,6)zurück

© 2016-2022 Altova GmbH

19.2.2.1.8 XPath/XQuery-Funktionen: String

Die folgenden XPath/XQuey-Erweiterungsfunktionen für Strings werden in der aktuellen Version Ihres Altova-Produkts unterstützt und bieten Zusatzfunktionalitäten für die Verarbeitung von Daten. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0-** und **XQuery 3.0**-Prozessor von Altova verwendet werden. Sie stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

▼ camel-case [altova:]

`altova:camel-case(InputString` *als xs:string*`)` als **`xs:string`** XP3.1 XQ3.1
Gibt den Input-String **`InputString`** in CamelCase zurück. Der String wird mit Hilfe der Regular Expression **`'\s'`** (welches ein Kürzel für das Leerzeichen ist) analysiert. Das erste Zeichen nach einem Leerzeichen oder einer Sequenz aufeinander folgender Leerzeichen, das kein Leerzeichen ist, wird mit einem Großbuchstaben geschrieben. Das erste Zeichen im Ausgabestring wird mit einem Großbuchstaben geschrieben.

Beispiele

- **`altova:camel-case`**`("max")` gibt `Max` zurück
- **`altova:camel-case`**`("max max")` gibt `Max Max` zurück
- **`altova:camel-case`**`("file01.xml")` gibt `File01.xml` zurück
- **`altova:camel-case`**`("file01.xml file02.xml")` gibt `File01.xml File02.xml` zurück
- **`altova:camel-case`**`("file01.xml  file02.xml")` gibt `File01.xml  File02.xml` zurück
- **`altova:camel-case`**`("file01.xml -file02.xml")` gibt `File01.xml -file02.xml` zurück

`altova:camel-case(InputString` *als xs:string*`,` **`SplitChars`** *als xs:string*`,` **`IsRegex`** *als xs:boolean*`)` als **`xs:string`** XP3.1 XQ3.1
Konvertiert den Input-String **`InputString`** in CamelCase, indem anhand von **`SplitChars`** festgelegt wird, welche(s) Zeichen die nächste Konvertierung in Großbuchstaben auslöst. **`SplitChars`** wird als Regular Expression verwendet, wenn **`IsRegex = true()`** oder als einfache Zeichen, wenn **`IsRegex = false()`**. Das erste Zeichen im Ausgabestring wird mit einem Großbuchstaben geschrieben.

Beispiele

- **`altova:camel-case`**`("setname getname", "set|get", true())` gibt `setName getName` zurück

- **altova:camel-case**("altova\documents\testcases", "\", false()) gibt Altova\Documents\Testcases zurück

▼ char [altova:]

altova:char(Position *as xs:integer***)** als **xs:string** XP3.1 XQ3.1
Gibt einen String zurück, der das Zeichen an der durch das Argument Position definierten Position enthält. Dieses Zeichen wird durch Konvertierung des Werts des Kontextelements in xs:string ermittelt. Der Ergebnisstring ist leer, wenn an dem durch das Position Argument gelieferten Index kein Zeichen vorhanden ist.

Beispiele

Wenn das Kontextelement **1234ABCD** lautet:

- **altova:char**(2) gibt 2 zurück
- **altova:char**(5) gibt A zurück
- **altova:char**(9) gibt den leeren String zurück.
- **altova:char**(-2) gibt den leeren String zurück.

altova:char(InputString *als xs:string*, **Position** *als xs:integer***)** als **xs:string** XP3.1 XQ3.1
Gibt einen String zurück, der das Zeichen enthält, das sich in dem als InputString Argument gelieferten String an der durch das Argument Position definierten Position befindet. Der Ergebnisstring ist leer, wenn an dem durch das Position Argument gelieferten Index kein Zeichen vorhanden ist.

Beispiele

- **altova:char**("2014-01-15", 5) gibt - zurück
- **altova:char**("USA", 1) gibt U zurück
- **altova:char**("USA", 1) gibt den leeren String zurück.
- **altova:char**("USA", -2) gibt den leeren String zurück.

▼ create-hash-from-string[altova:]

altova:create-hash-from-string(InputString *als xs:string***)** als **xs:string** XP2 XQ1 XP3.1 XQ3.1
altova:create-hash-from-string(InputString *als xs:string*, **HashAlgo** *als xs:string***)** als **xs:string** XP2 XQ1 XP3.1 XQ3.1
Generiert anhand von InputString mit Hilfe des durch das Argument HashAlgo definierten Hash-Algorithmus einen Hash-String. Es können die folgenden Hash-Algorithmen definiert werden (in Groß- oder Kleinbuchstaben): **MD5**, **SHA-1**, **SHA-224**, **SHA-256**, **SHA-384**, **SHA-512**. Wenn das zweite Argument nicht definiert ist (*siehe erste Signatur oben*), wird der Hash-Algorithmus **SHA-256** verwendet.

Beispiele

- **altova:create-hash-from-string**('abc') gibt einen Hash-String zurück, der mit Hilfe des Hash-Algorithmus **SHA-256** generiert wurde.
- **altova:create-hash-from-string**('abc', 'md5') gibt einen Hash-String zurück, der mit Hilfe des Hash-Algorithmus **MD5** generiert wurde.
- **altova:create-hash-from-string**('abc', 'MD5') gibt einen Hash-String zurück, der mit Hilfe des Hash-Algorithmus **MD5** generiert wurde.

© 2016-2022 Altova GmbH

▼ first-chars [altova:]

altova:first-chars(X-Number *as xs:integer*) als **xs:string** XP3.1 XQ3.1
Gibt einen String zurück, der die ersten x Zeichen (bezeichnet durch `X-Number`) des String enthält, der durch Konvertierung des Werts des Kontextelements in `xs:string` erzeugt wird.

Beispiele

Wenn das Kontextelement **1234ABCD** lautet:

- `altova:first-chars(2)` gibt `12` zurück
- `altova:first-chars(5)` gibt `1234A` zurück
- `altova:first-chars(9)` gibt `1234ABCD` zurück

altova:first-chars(InputString *als xs:string*, **X-Number** *als xs:integer*) als **xs:string** XP3.1 XQ3.1
Gibt einen String zurück, der die ersten x Zeichen (bezeichnet durch `X-Number`) des String enthält, das als das Argument `InputString` angegeben ist.

Beispiele

- `altova:first-chars("2014-01-15", 5)` gibt `2014-` zurück
- `altova:first-chars("USA", 1)` gibt `U` zurück

▼ format-string [altova:]

altova:format-string(InputString *als xs:string*, **FormatSequence** *als item()**) als **xs:string** XP3.1 XQ3.1
Der Input String (erstes Argument) enthält Positionsparameter (`%1`, `%2`, usw.). Jeder Parameter wird durch das String-Element ersetzt, das sich in der (als zweites Argument bereitgestellten) Formatsequenz an der entsprechenden Position befindet. Daher ersetzt des erste Element in der Formatsequenz den Positionsparameter `%1`, das zweite den Positionsparameter `%2`, usw. Die Funktion gibt diesen formatierten String zurück, der die Ersetzungen enthält. Wenn für einen Positionsparameter kein String existiert, wird der Positionsparameter selbst zurückgegeben. Dies kommt vor, wenn der Index eines Positionsparameters größer als die Anzahl der Elemente in der Formatsequenz ist.

Beispiele

- `altova:format-string('Hello %1, %2, %3', ('Jane','John','Joe'))` gibt `"Hello Jane, John, Joe"` zurück.
- `altova:format-string('Hello %1, %2, %3', ('Jane','John','Joe', 'Tom'))` gibt `"Hello Jane, John, Joe"` zurück.
- `altova:format-string('Hello %1, %2, %4', ('Jane','John','Joe', 'Tom'))` gibt `"Hello Jane, John, Tom"` zurück.
- `altova:format-string('Hello %1, %2, %4', ('Jane','John','Joe'))` gibt `"Hello Jane, John, %4"` zurück.

▼ last-chars [altova:]

altova:last-chars(X-Number *als xs:integer*) als **xs:string** XP3.1 XQ3.1
Gibt einen String zurück, der die letzten x Zeichen (bezeichnet durch `X-Number`) des String enthält, der durch Konvertierung des Werts des Kontextelements in `xs:string` erzeugt wird.

Beispiele

Wenn das Kontextelement **1234ABCD** lautet:

- **altova:last-chars**(2) gibt CD zurück
- **altova:last-chars**(5) gibt 4ABCD zurück
- **altova:last-chars**(9) gibt 1234ABCD zurück

altova:last-chars(InputString *als xs:string*, **X-Number** *als xs:integer*) als **xs:string** XP3.1 XQ3.1

Gibt einen String zurück, der die letzten x Zeichen (bezeichnet durch X-Number) des String enthält, das als das Argument InputString angegeben ist.

Beispiele

- **altova:last-chars**("2014-01-15", 5) gibt 01-15- zurück
- **altova:last-chars**("USA", 10) gibt USA zurück

pad-string-left [altova:]

altova:pad-string-left(StringToPad *als xs:string*, **Repeats** *als xs:integer*, **PadCharacter** *als xs:string*) als **xs:string** XP3.1 XQ3.1

Das Argument PadCharacter ist ein einzelnes Zeichen. Es wird links vom String als Auffüllzeichen eingefügt, um die Anzahl der Zeichen in StringToPad zu erhöhen, damit diese Anzahl dem Ganzzahlwert des Arguments StringLength entspricht. Das Argument StringLength kann jeden beliebigen (positiven oder negativen) Ganzzahlwert haben, Auffüllzeichen werden aber nur verwendet, wenn der Wert von StringLength größer als die Anzahl der Zeichen in StringToPad ist. Wenn StringToPad mehr Zeichen als der Wert von StringLength hat, bleibt StringToPad unverändert.

Beispiele

- **altova:pad-string-left**('AP', 1, 'Z') gibt 'AP' zurück
- **altova:pad-string-left**('AP', 2, 'Z') gibt 'AP' zurück
- **altova:pad-string-left**('AP', 3, 'Z') gibt 'ZAP' zurück
- **altova:pad-string-left**('AP', 4, 'Z') gibt 'ZZAP' zurück
- **altova:pad-string-left**('AP', -3, 'Z') gibt 'AP' zurück
- **altova:pad-string-left**('AP', 3, 'YZ')gibt einen Fehler zurück, dass das Auffüllzeichen zu lang ist

pad-string-right [altova:]

altova:pad-string-right(StringToPad *als xs:string*, **Repeats** *als xs:integer*, **PadCharacter** *als xs:string*) als **xs:string** XP3.1 XQ3.1

Das Argument PadCharacter ist ein einzelnes Zeichen. Es wird rechts vom String als Auffüllzeichen eingefügt, um die Anzahl der Zeichen in StringToPad zu erhöhen, damit diese Anzahl dem Ganzzahlwert des Arguments StringLength entspricht. Das Argument StringLength kann jeden beliebigen (positiven oder negativen) Ganzzahlwert haben, Auffüllzeichen werden aber nur verwendet, wenn der Wert von StringLength größer als die Anzahl der Zeichen in StringToPad ist. Wenn StringToPad mehr Zeichen als der Wert von StringLength hat, bleibt StringToPad unverändert.

Beispiele

- **altova:pad-string-right**('AP', 1, 'Z') gibt 'AP' zurück
- **altova:pad-string-right**('AP', 2, 'Z') gibt 'AP' zurück

© 2016-2022 Altova GmbH

- **altova:pad-string-right**('AP', 3, 'Z') gibt 'ZAP' zurück
- **altova:pad-string-right**('AP', 4, 'Z') gibt 'ZZAP' zurück
- **altova:pad-string-right**('AP', -3, 'Z') gibt 'AP' zurück
- **altova:pad-string-right**('AP', 3, 'YZ')gibt einen Fehler zurück, dass das Auffüllzeichen zu lang ist

▼ repeat-string [altova:]

altova:repeat-string(InputString *als xs:string*, **Repeats** *als xs:integer***)** als **xs:string** XP2 XQ1 XP3.1 XQ3.1

Generiert einen String, der sich zusammensetzt aus dem ersten `InputString`-Argument, das die Anzahl der `Repeats` wiederholt wird.

Beispiele

- **altova:repeat-string**("Altova #", 3) gibt "Altova #Altova #Altova #" zurück

▼ substring-after-last [altova:]

altova:substring-after-last(MainString *als xs:string*, **CheckString** *als xs:string***)** als **xs:string** XP3.1 XQ3.1

Falls in `MainString` `CheckString` gefunden wird, so wird der Substring zurückgegeben, der in `MainString` nach `CheckString` steht. Falls `CheckString` in `MainString` nicht gefunden wird, so wird der leere String zurückgegeben. Wenn `CheckString` ein leerer String ist, so wird der gesamte `MainString` zurückgegeben. Falls `CheckString` mehrmals in `MainString`, vorkommt, so wird der Substring nach der letzten Instanz von `CheckString` zurückgegeben.

Beispiele

- **altova:substring-after-last**('ABCDEFGH', 'B') gibt 'CDEFGH' zurück
- **altova:substring-after-last**('ABCDEFGH', 'BC') gibt 'DEFGH'zurück
- **altova:substring-after-last**('ABCDEFGH', 'BD') gibt '' zurück
- **altova:substring-after-last**('ABCDEFGH', 'Z') gibt '' zurück
- **altova:substring-after-last**('ABCDEFGH', '') gibt 'ABCDEFGH' zurück
- **altova:substring-after-last**('ABCD-ABCD', 'B') gibt 'CD' zurück
- **altova:substring-after-last**('ABCD-ABCD-ABCD', 'BCD') gibt '' zurück

▼ substring-before-last [altova:]

altova:substring-before-last(MainString *as xs:string*, **CheckString** *as xs:string***)** **as** **xs:string** XP3.1 XQ3.1

Falls in `MainString` `CheckString` gefunden wird, so wird der Substring zurückgegeben, der in `MainString` vor `CheckString` steht. Falls `CheckString` in `MainString` nicht gefunden wird, so wird der leere String zurückgegeben. Wenn `CheckString` ein leerer String ist, so wird der gesamte `MainString` zurückgegeben. Falls `CheckString` mehrmals in `MainString`, vorkommt, so wird der Substring vor der letzten Instanz von `CheckString` zurückgegeben.

Beispiele

- **altova:substring-before-last**('ABCDEFGH', 'B') gibt 'A' zurück
- **altova:substring-before-last**('ABCDEFGH', 'BC') gibt 'A' zurück

- **altova:substring-before-last**('ABCDEFGH', 'BD') gibt '' zurück
- **altova:substring-before-last**('ABCDEFGH', 'Z') gibt '' zurück
- **altova:substring-before-last**('ABCDEFGH', '') gibt '' zurück
- **altova:substring-before-last**('ABCD-ABCD', 'B') gibt 'ABCD-A' zurück
- **altova:substring-before-last**('ABCD-ABCD-ABCD', 'ABCD') gibt 'ABCD-ABCD-' zurück

substring-pos [altova:]

altova:substring-pos(StringToCheck *als xs:string*, **StringToFind** *als xs:string*) als **xs:integer** XP3.1 XQ3.1

Gibt die Zeichenposition der ersten Instanz von StringToFind im String StringToCheck zurück. Die Zeichenposition wird in Form einer Ganzzahl angegeben. Das erste Zeichen von StringToCheck hat die Position 1. Wenn StringToFind in StringToCheck nicht vorkommt, wird die Ganzzahl 0 zurückgegeben. Um den String auf eine zweite oder eine weiter hinten folgende Instanz von StringToCheck zu überprüfen, verwenden Sie die nächste Signatur dieser Funktion.

Beispiele

- **altova:substring-pos**('Altova', 'to') gibt 3 zurück
- **altova:substring-pos**('Altova', 'tov') gibt 3 zurück
- **altova:substring-pos**('Altova', 'tv') gibt 0 zurück
- **altova:substring-pos**('AltovaAltova', 'to') gibt 3 zurück

altova:substring-pos(StringToCheck *als xs:string*, **StringToFind** *als xs:string*, **Integer** *als xs:integer*) als **xs:integer** XP3.1 XQ3.1

Gibt die Zeichenposition von StringToFind im String StringToCheck zurück. Die Suche nach StringToFind beginnt an der durch das Argument Integer angegebenen Zeichenposition; der Zeichen-Substring vor dieser Position wird nicht durchsucht. Die zurückgegebene Ganzzahl gibt jedoch die Position des gefundenen String innerhalb des *gesamten* String StringToCheck an. Diese Signatur dient dazu, die zweite oder eine weiter hinten folgende Position eines String zu finden, der mehrmals in StringToCheck vorkommt. Wenn StringToFind in StringToCheck nicht vorkommt, wird die Ganzzahl 0 zurückgegeben.

Beispiele

- **altova:substring-pos**('Altova', 'to', 1) gibt 3 zurück
- **altova:substring-pos**('Altova', 'to', 3) gibt 3 zurück
- **altova:substring-pos**('Altova', 'to', 4) gibt 0 zurück
- **altova:substring-pos**('Altova-Altova', 'to', 0) gibt 3 zurück
- **altova:substring-pos**('Altova-Altova', 'to', 4) gibt 10 zurück

trim-string [altova:]

altova:trim-string(InputString *als xs:string*) als **xs:string** XP3.1 XQ3.1

Diese Funktion verwendet ein xs:string Argument, entfernt alle voran- und nachgestellten Leerzeichen und gibt einen "getrimmten" xs:string zurück.

Beispiele

- **altova:trim-string**(" Hello World ") gibt "Hello World" zurück
- **altova:trim-string**("Hello World ") gibt "Hello World" zurück
- **altova:trim-string**(" Hello World") gibt "Hello World" zurück

© 2016-2022 Altova GmbH

- **altova:trim-string**("Hello World") gibt "Hello World" zurück
- **altova:trim-string**("Hello World") gibt "Hello World" zurück

▼ trim-string-left [altova:]

altova:trim-string-left(InputString *als xs:string***)** als **xs:string** XP3.1 XQ3.1
Diese Funktion verwendet ein xs:string Argument, entfernt alle vorangestellten Leerzeichen und gibt einen "links getrimmten" xs:string zurück.

Beispiele

- **altova:trim-string-left**(" Hello World ") gibt "Hello World " zurück
- **altova:trim-string-left**("Hello World ") gibt "Hello World " zurück
- **altova:trim-string-left**(" Hello World") gibt "Hello World" zurück
- **altova:trim-string-left**("Hello World") gibt "Hello World" zurück
- **altova:trim-string-left**("Hello World") gibt "Hello World" zurück

▼ trim-string-right [altova:]

altova:trim-string-right(InputString *als xs:string***)** als **xs:string** XP3.1 XQ3.1
Diese Funktion verwendet ein xs:string Argument, entfernt alle nachgestellten Leerzeichen und gibt einen "rechts getrimmten" xs:string zurück.

Beispiele

- **altova:trim-string-right**(" Hello World ") gibt " Hello World" zurück
- **altova:trim-string-right**("Hello World ") gibt "Hello World" zurück
- **altova:trim-string-right**(" Hello World") gibt " Hello World" zurück
- **altova:trim-string-right**("Hello World") gibt "Hello World" zurück
- **altova:trim-string-right**("Hello World") gibt "Hello World" zurück

19.2.2.1.9 XPath/XQuery-Funktionen: Diverse Funktionen

Die folgenden XPath/XQuery-Funktionen für allgemeine Zwecke werden in der aktuellen Version von MapForce unterstützt und können in (i) in einem XSLT-Kontext in XPath-Ausdrücken oder (ii) in einem XQuery-Dokument in XQuery-Ausdrücken verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, http://www.altova.com/xslt-extensions** und sind in diesem Abschnitt mit dem Präfix **altova:**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

decode-string [altova:]

altova:decode-string(Input *als xs:base64Binary*) **als xs:string** XP3.1 XQ3.1
altova:decode-string(Input *als xs:base64Binary*, **Encoding** *als xs:string*) **als xs:string** XP3.1 XQ3.1
Dekodiert den angegebenen base64Binary-Input anhand der definierten Kodierung zu einem String. Wenn keine Kodierung definiert ist, wird die UTF-8-Kodierung verwendet. Die folgenden Kodierungen werden unterstützt: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

Beispiele

- **altova:decode-string**(`$XML1/MailData/Meta/b64B`) gibt den base64Binary-Input als UTF-8-kodierten String zurück.
- **altova:decode-string**(`$XML1/MailData/Meta/b64B, "UTF-8"`) gibt den base64Binary-Input als UTF-8-kodierten String zurück.
- **altova:decode-string**(`$XML1/MailData/Meta/b64B, "ISO-8859-1"`) gibt den base64Binary-Input als ISO-8859-1-kodierten String zurück.

encode-string [altova:]

altova:encode-string(InputString *als xs:string*) **als xs:base64Binaryinteger** XP3.1 XQ3.1
altova:encode-string(InputString *als xs:string*, **Encoding** *als xs:string*) **als xs:base64Binaryinteger** XP3.1 XQ3.1
Kodiert den angegebenen String gemäß der definierten Kodierung, falls eine angegeben wird. Wenn keine Kodierung definiert ist, wird die UTF-8-Kodierung verwendet. Der kodierte String wird in base64Binary-Zeichen konvertiert und es wird der konvertierte base64Binary-Wert zurückgegeben. Anfangs wird die UTF-8-Kodierung unterstützt. Die Unterstützung wird auf die folgenden Kodierungen ausgeweitet werden: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

Beispiele

- **altova:encode-string**(`"Altova"`) gibt das base64Binary-Äquivalent des UTF-8-kodierten String `"Altova"` zurück.
- **altova:encode-string**(`"Altova", "UTF-8"`) gibt das base64Binary-Äquivalent des UTF-8-kodierten String `"Altova"` zurück.

get-temp-folder [altova:]

altova:get-temp-folder() als `xs:string` XP2 XQ1 XP3.1 XQ3.1
Diese Funktion hat kein Argument. Sie gibt den Pfad zum temporären Ordner des aktuellen Benutzers zurück.

© 2016-2022 Altova GmbH

Beispiele

- **altova:get-temp-folder**() würde auf einem Windows-Rechner z.B. den folgenden Pfad als xs:string zurückgeben: C:\Users\<UserName>\AppData\Local\Temp\.

generate-guid [altova:]

altova:generate-guid() als **xs:string** XP2 XQ1 XP3.1 XQ3.1
Generiert einen eindeutigen String GUID-String.

Beispiele

- **altova:generate-guid**() gibt (z.B.) 85F971DA-17F3-4E4E-994E-99137873ACCD zurück

high-res-timer [altova:]

altova:high-res-timer() als **xs:double** XP3.1 XQ3.1
Gibt einen hochauflösenden System-Timer-Wert in Sekunden zurück. Wenn in einem System ein hochauflösender Timer zur Verfügung steht, können bei Bedarf (z.B. bei Animationen und zur Ermittlung des exakten Codeausführungszeitpunkts) hochauflösende Zeitmessungen vorgenommen werden. Diese Funktion stellt die Auflösung des Hochauflösungs-Timers des Systems zur Verfügung.

Beispiele

- **altova:high-res-timer**() gibt eine Wert wie '1.16766146154566E6' zurück.

parse-html [altova:]

altova:parse-html(HTMLText *als xs:string***)** als **node()** XP3.1 XQ3.1
Das Argument HTMLText ist ein String, der den Text eines HTML-Dokuments enthält. Die Funktion erstellt anhand des Strings eine HTML-Struktur. Der bereitgestellte String kann das HTML-Element enthalten, muss dies aber nicht tun. In beiden Fällen ist das Root-Element der Struktur ein Element namens **HTML**. Sie sollten sicher stellen, dass der HTML-Code im bereitgestellten String gültiger HTML-Code ist.

Beispiel

- **altova:parse-html**("<html><head/><body><h1>Header</h1></body></html>") erstellt anhand des bereitgestellten Strings eine HTML-Struktur.

sleep[altova:]

altova:sleep(Millisecs *als xs:integer***)** als **empty-sequence()** XP2 XQ1 XP3.1 XQ3.1
Unterbricht die Ausführung der aktuellen Operation für die Anzahl der durch das Argument Millisecs angegebenen Millisekunden.

Beispiel

- **altova:sleep**(1000) unterbricht die Ausführung der aktuellen Operation für 1000 Millisekunden.

[Nach oben 2071]

19.2.2.1.10 Diagrammfunktionen

Mit Hilfe der unten aufgelisteten Diagrammfunktionen können Sie Diagramme als Bilder erstellen, generieren und speichern. Sie werden in der aktuellen Version Ihres Altova-Produkts auf die unten beschriebene Art unterstützt. Beachten Sie jedoch, dass eine oder mehrere dieser Funktionen in zukünftigen Produktversionen eventuell nicht mehr unterstützt werden, bzw. dass sich das Verhalten einzelner Funktionen ändern kann. Um Informationen über die Unterstützung für Altova Erweiterungsfunktionen in der jeweiligen Release zu erhalten, schlagen Sie bitte in der Dokumentation der jeweils aktuellen Release nach.

Anmerkung: Diagrammfunktionen werden nur in **Altova Server-Produkten** und den **Enterprise-Editionen von Altova-Produkten** unterstützt.

Anmerkung: Unterstützte Bildformate für Diagramme in Server-Editionen sind `jpg`, `png` und `bmp`. Die beste Option ist png, da dabei keine Daten verloren gehen und es sich um ein komprimiertes Format handelt. In Enterprise-Editionen werden die folgenden Formate unterstützt: `jpg`. `png`, `bmp` und `gif`.

Funktionen zum Generieren und Speichern von Diagrammen

Diese Funktionen generieren anhand des (mit Hilfe der Diagrammerstellungsfunktionen) erzeugten Diagrammobjekts entweder ein Bild oder speichern ein Bild in einer Datei

`altova:generate-chart-image` `($chart, $width, $height, $encoding) als atomic`

wobei

- `$chart` das Diagramm-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart` erzeugt wurde
- `$width` und `$height` mit einer Längeneinheit definiert werden müssen
- `$encoding` kann `x-binarytobase64` oder `x-binarytobase16` sein

Die Funktion gibt das Diagrammbild in der definierten Kodierung zurück.

`altova:generate-chart-image` `($chart, $width, $height, $encoding, $imagetype) als atomic`

wobei

- `$chart` das Diagramm-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart` erzeugt wurde
- `$width` und `$height` mit einer Längeneinheit definiert werden müssen
- `$encoding` kann `x-binarytobase64` oder `x-binarytobase16` sein
- `$imagetype` eines der folgenden Bildformate sein kann: `png`, `gif`, `bmp`, `jpg`, `jpeg`. Beachten Sie, dass `gif` in Server-Produkten nicht unterstützt wird. *Siehe oben auf dieser Seite*.

Die Funktion gibt das Diagrammbild in der definierten Kodierung und im definierten Bildformat zurück.

`altova:save-chart-image` `($chart, $filename, $width, $height) als empty()`***(nur Windows)***

wobei

© 2016-2022 Altova GmbH

- `$chart` das Diagramm-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart` erzeugt wurde
- `$filename` der Pfad und Name der Datei ist, unter dem das Diagrammbild gespeichert werden soll
- `$width` und `$height` mit einer Längeneinheit definiert werden müssen

Die Funktion speichert das Diagrammbild unter dem in `$filename` definierten Dateinamen. Alternativ zu dieser Funktion könnten Sie auch die Funktion `xsl:result-document` mit `encoding="x-base64tobinary"` verwenden, wobei der image-data-Inhalt entweder über die Funktion `generate-chart-image()` oder die Funktion `chart()` erhalten wird.

`altova:save-chart-image` `($chart, $filename, $width, $height, $imagetype) als empty()`***(nur Windows)***

wobei

- `$chart` das Diagramm-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart` erzeugt wurde
- `$filename` der Pfad und Name der Datei ist, unter dem das Diagrammbild gespeichert werden soll
- `$width` und `$height` mit einer Längeneinheit definiert werden müssen
- `$imagetype` eines der folgenden Bildformate sein kann: `png`, `gif`, `bmp`, `jpg`, `jpeg`. Beachten Sie, dass `gif` in Server-Produkten nicht unterstützt wird. *Siehe oben auf dieser Seite*.

Die Funktion speichert das Diagrammbild im definierten Bildformat unter dem in `$filename` definierten Dateinamen. Alternativ zu dieser Funktion könnten Sie auch die Funktion `xsl:result-document` mit `encoding="x-base64tobinary"` verwenden, wobei der image-data-Inhalt entweder über die Funktion `generate-chart-image()` oder die Funktion `chart()` erhalten wird.

Funktionen zur Erstellung von Diagrammen

Die folgenden Funktionen dienen zur Erstellung von Diagrammen.

`altova:create-chart``($chart-config, $chart-data-series*) als chart extension item`

wobei

- `$chart-config` das Diagrammkonfigurations-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-config` oder über die Funktion `altova:create-chart-config-from-xml` erzeugt wurde
- `$chart-data-series` das chart-data-series-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-data-series` oder der Funktion `altova:create-chart-data-series-from-rows` abgerufen wird

Die Funktion gibt ein Diagrammerweiterungsobjekt zurück, das anhand der über die Argumente gelieferten Daten erzeugt wird.

`altova:chart``($chart-config, $chart-data-series*) als chart extension item`

wobei

- `$chart-config` das Diagrammkonfigurations-Erweiterungsobjekt ist. Es handelt sich hierbei um eine nicht geordnete Reihe von vier Schlüssel:Wert-Paaren, wobei die vier Schlüssel `"width"`, `"height"`, `"title"` und `"kind"` sind. Die Werte von `width` und `height` sind Ganzzahlen und definieren die Höhe und Breite des Diagramms in Pixel. Der Wert von `kind` ist einer der folgenden: `Pie, Pie3d, BarChart, BarChart3d, BarChart3dGrouped, LineChart, ValueLineChart, RoundGauge, BarGauge`.
- `$chart-data-series` ist jeweils ein Array der Größe 3, wobei jedes Array eine Diagrammdatenreihe definiert. Jedes Array besteht aus: (i) dem Namen der Datenreihe, (ii) den Werten der x-Achse, (iii) den Werten der y-Achse. Es können mehrere Datenreihen angegeben werden. Im Beispiel unten enthalten die zwei Arrays Daten für die Temperatur-Monatsminima bzw. -maxima.

Die Funktion gibt ein Element vom Typ `xs:base64Binary` zurück, welches das Diagrammbild enthält. Dieses Bild wird anhand der über die Argumente der Funktion bereitgestellten Daten erstellt. Beachten Sie, dass diese Funktion, da sie Arrays und Zuordnungen verwendet, nur in XPath 3.1, XQuery 3.1 oder XSLT 3.0 verwendet werden kann.

Beispiel:
```
altova:chart( map{'width':800, 'height':600, "kind":"LineChart", "title":"Monthly Temperatures"}, (['Min', $temps/Month, $temps/Month/@min], ['Max', $temps/Month, $temps/Month/@max]) )
```

`altova:create-chart-config``($type-name, $title) als chart-config Erweiterungsobjekt`

wobei

- `$type-name` den Typ des zu erstellenden Diagramms definiert: `Pie, Pie3d, BarChart, BarChart3d, BarChart3dGrouped, LineChart, ValueLineChart, RoundGauge, BarGauge`
- `$title` der Name des Diagramms ist

Die Funktion gibt ein Diagrammkonfigurations-Erweiterungsobjekt zurück, das die Konfigurationsinformationen zum Diagramm enthält.

`altova:create-chart-config-from-xml``($xml-struct) als chart-config Erweiterungsobjekt`

wobei

- `$xml-struct` die XML-Struktur ist, die die Konfigurationsinformationen des Diagramms enthält

Die Funktion gibt ein Diagrammkonfigurations-Erweiterungsobjekt zurück, das die Konfigurationsinformationen zum Diagramm enthält. Diese Informationen werden in einem XML-Datenfragment[2078] geliefert.

`altova:create-chart-data-series``($series-name?, $x-values*, $y-values*) als chart-data-series Erweiterungsobjekt`

wobei

- `$series-name` der Name der Datenreihe ist
- `$x-values` die Liste der Werte für die X-Achse liefert
- `$y-values` die Liste der Werte für die Y-Achse liefert

© 2016-2022 Altova GmbH

Die Funktion gibt ein Diagrammdatenreihen-Erweiterungsobjekt zurück, das die Daten zur Erstellung des Diagramms, also die Namen der Datenreihen, und die Achsendaten enthält.

altova:create-chart-data-row(x, y1, y2, y3, ...) als chart-data-x-*N*y-row Erweiterungsobjekt

wobei

- x der Wert der X-Achsen-Spalte der Diagrammdatenzeile ist
- yN die Werte der Spalten für die Y-Achse sind

Die Funktion gibt ein chart-data-x-Ny-row Erweiterungsobjekt zurück, das die Daten für die X-Achsen-Spalte und die Y-Achsen-Spalten einer einzigen Datenreihe enthält.

altova:create-chart-data-series-from-rows($series-names as xs:string*, $row*) als chart-data-series-Erweiterungsobjekt

wobei

- $series-name der Name der zu erstellenden Datenreihen ist
- $row das chart-data-x-Ny-row Erweiterungsobjekt ist, das als Datenreihe erstellt werden soll

Die Funktion gibt ein chart-data-series Erweiterungsobjekt zurück, das die Daten für die X- und die Y-Achse der Datenreihe enthält.

altova:create-chart-layer($chart-config, $chart-data-series*) als chart-layer Erweiterungsobjekt

wobei

- $chart-config das Diagrammkonfigurations-Erweiterungsobjekt ist, das mit der Funktion altova:create-chart-config oder über die Funktion altova:create-chart-config-from-xml erzeugt wurde
- $chart-data-series das chart-data-series-Erweiterungsobjekt ist, das mit der Funktion altova:create-chart-data-series oder der Funktion altova:create-chart-data-series-from-rows abgerufen wird

Die Funktion gibt ein chart-layer Erweiterungsobjekt zurück, das chart-layer-Daten enthält.

altova:create-multi-layer-chart($chart-config, $chart-data-series*, $chart-layer*)

wobei

- $chart-config das Diagrammkonfigurations-Erweiterungsobjekt ist, das mit der Funktion altova:create-chart-config oder über die Funktion altova:create-chart-config-from-xml erzeugt wurde
- $chart-data-series das chart-data-series-Erweiterungsobjekt ist, das mit der Funktion altova:create-chart-data-series oder der Funktion altova:create-chart-data-series-from-rows abgerufen wird

- `$chart-layer` das chart-layer-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-layer` abgerufen wird

Die Funktion gibt ein multi-layer-chart-Objekt zurück.

`altova:create-multi-layer-chart``($chart-config, $chart-data-series*, $chart-layer*, xs:boolean $mergecategoryvalues)`

wobei

- `$chart-config` das Diagrammkonfigurations-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-config` oder über die Funktion `altova:create-chart-config-from-xml` erzeugt wurde
- `$chart-data-series` das chart-data-series-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-data-series` oder der Funktion `altova:create-chart-data-series-from-rows` abgerufen wird
- `$chart-layer` das chart-layer-Erweiterungsobjekt ist, das mit der Funktion `altova:create-chart-layer` abgerufen wird
- `$mergecategoryvalues` bei `true` die Werte mehrerer Datenreihen zusammenführt und dies bei `false` nicht tut.

Die Funktion gibt ein multi-layer-chart-Objekt zurück.

19.2.2.1.10.1 XML-Struktur von Diagrammdaten

Unten sehen Sie die XML-Struktur von Diagrammdaten, wie sie für Altova-Erweiterungsfunktionen für Diagramme 2074 angezeigt werden könnte. Diese Funktionen beeinflussen das Aussehen der einzelnen Diagramme. Nicht alle Elemente werden für alle Diagrammarten verwendet, so wird z.B. das Element <Pie> bei Balkendiagrammen ignoriert.

Anmerkung: Diagrammfunktionen werden nur in **Enterprise und Server-Editionen** von Altova Produkten unterstützt.

```
<chart-config>
        <General
        SettingsVersion="1"   must be provided
        ChartKind="BarChart"   Pie, Pie3d, BarChart, StackedBarChart, BarChart3d, BarChart3dGrouped,
LineChart, ValueLineChart, AreaChart, StackedAreaChart, RoundGauge, BarGauge, CandleStick
        BKColor="#ffffff"   Color
        BKColorGradientEnd="#ffffff"   Color. In case of a gradient, BKColor and BKColorGradientEnd
define the gradient's colors
        BKMode="#ffffff"   Solid, HorzGradient, VertGradient
        BKFile="Path+Filename"   String. If file exists, its content is drawn over the background.
        BKFileMode="Stretch"   Stretch, ZoomToFit, Center, Tile
        ShowBorder="1"   Bool
        PlotBorderColor="#000000"   Color
        PlotBKColor="#ffffff"   Color
        Title=""   String
        ShowLegend="1"   Bool
```


© 2016-2022 Altova GmbH

```
OutsideMargin="3.%"   PercentOrPixel
TitleToPlotMargin="3.%"   PercentOrPixel
LegendToPlotMargin="3.%"   PercentOrPixel
Orientation="vert"   Enumeration: possible values are: vert, horz
>
<TitleFont
   Color="#000000"   Color
   Name="Tahoma"   String
   Bold="1"   Bool
   Italic="0"   Bool
   Underline="0"   Bool
   MinFontHeight="10.pt"   FontSize (only pt values)
   Size="8.%"   FontSize />
<LegendFont
   Color="#000000"
   Name="Tahoma"
   Bold="0"
   Italic="0"
   Underline="0"
   MinFontHeight="10.pt"
   Size="3.5%" />
<AxisLabelFont
   Color="#000000"
   Name="Tahoma"
   Bold="1"
   Italic="0"
   Underline="0"
   MinFontHeight="10.pt"
   Size="5.%" />
</General>

<Line
      ConnectionShapeSize="1.%"   PercentOrPixel
      DrawFilledConnectionShapes="1"   Bool
      DrawOutlineConnectionShapes="0"   Bool
      DrawSlashConnectionShapes="0"   Bool
      DrawBackslashConnectionShapes="0"   Bool
/>

<Bar
      ShowShadow="1"   Bool
      ShadowColor="#a0a0a0"   Color
      OutlineColor="#000000"   Color
      ShowOutline="1"   Bool
/>

<Area
      Transparency="0"   UINT ( 0-255 ) 255 is fully transparent, 0 is opaque
      OutlineColor="#000000"   Color
      ShowOutline="1"   Bool
/>

<CandleStick
```

```
            FillHighClose="0"   Bool. If 0, the body is left empty. If 1, FillColorHighClose is used for
the candle body
            FillColorHighClose="#ffffff"   Color. For the candle body when close > open
            FillHighOpenWithSeriesColor="1"   Bool. If true, the series color is used to fill the
candlebody when open > close
            FillColorHighOpen="#000000"   Color. For the candle body when open > close and
FillHighOpenWithSeriesColor is false
      />

      <Colors   User-defined color scheme: By default this element is empty except for the style and has no
Color attributes
            UseSubsequentColors ="1"  Boolean. If 0, then color in overlay is used. If 1, then
subsequent colors from previous chart layer is used
            Style="User"   Possible values are: "Default", "Grayscale", "Colorful", "Pastel", "User"
            Colors="#52aca0"   Color: only added for user defined color set
            Colors1="#d3c15d"   Color: only added for user defined color set
            Colors2="#8971d8"   Color: only added for user defined color set
            ...
            ColorsN=""   Up to ten colors are allowed in a set: from Colors to Colors9
      </Colors>

      <Pie
            ShowLabels="1"   Bool
            OutlineColor="#404040"   Color
            ShowOutline="1"   Bool
            StartAngle="0."   Double
            Clockwise="1"   Bool
            Draw2dHighlights="1"   Bool
            Transparency="0"   Int (0 to 255: 0 is opaque, 255 is fully transparent)
            DropShadowColor="#c0c0c0"   Color
            DropShadowSize="5.%"   PercentOrPixel
            PieHeight="10.%"   PercentOrPixel. Pixel values might be different in the result because of
3d tilting
            Tilt="40.0"   Double (10 to 90: The 3d tilt in degrees of a 3d pie)
            ShowDropShadow="1"   Bool
            ChartToLabelMargin="10.%"   PercentOrPixel
            AddValueToLabel="0"   Bool
            AddPercentToLabel="0"   Bool
            AddPercentToLabels_DecimalDigits="0"  UINT ( 0 – 2 )
            >
            <LabelFont
                  Color="#000000"
                  Name="Arial"
                  Bold="0"
                  Italic="0"
                  Underline="0"
                  MinFontHeight="10.pt"
                  Size="4.%"    />
      </Pie>

      <XY>
            <XAxis   Axis
                  AutoRange="1"   Bool
```


© 2016-2022 Altova GmbH

```
                AutoRangeIncludesZero="1"  Bool
                RangeFrom="0."  Double: manual range
                RangeTill="1."  Double : manual range
                LabelToAxisMargin="3.%"  PercentOrPixel
                AxisLabel=""  String
                AxisColor="#000000"  Color
                AxisGridColor="#e6e6e6"  Color
                ShowGrid="1"  Bool
                UseAutoTick="1"  Bool
                ManualTickInterval="1."  Double
                AxisToChartMargin="0.px"  PercentOrPixel
                TickSize="3.px"  PercentOrPixel
                ShowTicks="1"  Bool
                ShowValues="1"  Bool
                AxisPosition="LeftOrBottom"  Enums: "LeftOrBottom", "RightOrTop", "AtValue"
                AxisPositionAtValue = "0"  Double
                >
                <ValueFont
                        Color="#000000"
                        Name="Tahoma"
                        Bold="0"
                        Italic="0"
                        Underline="0"
                        MinFontHeight="10.pt"
                        Size="3.%"    />
        </XAxis>
        <YAxis  Axis (same as for XAxis)
                AutoRange="1"
                AutoRangeIncludesZero="1"
                RangeFrom="0."
                RangeTill="1."
                LabelToAxisMargin="3.%"
                AxisLabel=""
                AxisColor="#000000"
                AxisGridColor="#e6e6e6"
                ShowGrid="1"
                UseAutoTick="1"
                ManualTickInterval="1."
                AxisToChartMargin="0.px"
                TickSize="3.px"
                ShowTicks="1"  Bool
                ShowValues="1"  Bool
                AxisPosition="LeftOrBottom"  Enums: "LeftOrBottom", "RightOrTop", "AtValue"
                AxisPositionAtValue = "0"  Double
                >
                <ValueFont
                        Color="#000000"
                        Name="Tahoma"
                        Bold="0"
                        Italic="0"
                        Underline="0"
                        MinFontHeight="10.pt"
                        Size="3.%"/>
        </YAxis>
</XY>
```

<XY3d

AxisAutoSize="1" *Bool: If false, XSize and YSize define the aspect ration of x and y axis. If true, aspect ratio is equal to chart window*

XSize="100.%" *PercentOrPixel. Pixel values might be different in the result because of 3d tilting and zooming to fit chart*

YSize="100.%" *PercentOrPixel. Pixel values might be different in the result because of 3d tilting and zooming to fit chart*

SeriesMargin="30.%" *PercentOrPixel. Pixel values might be different in the result because of 3d tilting and zooming to fit chart*

Tilt="20." *Double. -90 to +90 degrees*

Rot="20." *Double. -359 to +359 degrees*

FoV="50."> *Double. Field of view: 1-120 degree*

```
>
<ZAxis
       AutoRange="1"
       AutoRangeIncludesZero="1"
       RangeFrom="0."
       RangeTill="1."
       LabelToAxisMargin="3.%"
       AxisLabel=""
       AxisColor="#000000"
       AxisGridColor="#e6e6e6"
       ShowGrid="1"
       UseAutoTick="1"
       ManualTickInterval="1."
       AxisToChartMargin="0.px"
       TickSize="3.px" >
       <ValueFont
              Color="#000000"
              Name="Tahoma"
              Bold="0"
              Italic="0"
              Underline="0"
              MinFontHeight="10.pt"
              Size="3.%"/>
</ZAxis>
```

</XY3d>

<Gauge

MinVal="0." *Double*

MaxVal="100." *Double*

MinAngle="225" *UINT: -359-359*

SweepAngle="270" *UINT: 1-359*

BorderToTick="1.%" *PercentOrPixel*

MajorTickWidth="3.px" *PercentOrPixel*

MajorTickLength="4.%" *PercentOrPixel*

MinorTickWidth="1.px" *PercentOrPixel*

MinorTickLength="3.%" *PercentOrPixel*

BorderColor="#a0a0a0" *Color*

FillColor="#303535" *Color*

MajorTickColor="#a0c0b0" *Color*

MinorTickColor="#a0c0b0" *Color*

BorderWidth="2.%" *PercentOrPixel*

© 2016-2022 Altova GmbH

```
                NeedleBaseWidth="1.5%"   PercentOrPixel
                NeedleBaseRadius="5.%"   PercentOrPixel
                NeedleColor="#f00000"   Color
                NeedleBaseColor="#141414"   Color
                TickToTickValueMargin="5.%"   PercentOrPixel
                MajorTickStep="10."   Double
                MinorTickStep="5."   Double
                RoundGaugeBorderToColorRange="0.%"   PercentOrPixel
                RoundGaugeColorRangeWidth ="6.%"   PercentOrPixel
                BarGaugeRadius="5.%"   PercentOrPixel
                BarGaugeMaxHeight="20.%"   PercentOrPixel
                RoundGaugeNeedleLength="45.%"   PercentOrPixel
                BarGaugeNeedleLength="3.%"   PercentOrPixel
                >
                <TicksFont
                       Color="#a0c0b0"
                       Name="Tahoma"
                       Bold="0"
                       Italic="0"
                       Underline="0"
                       MinFontHeight="10.pt"
                       Size="4.%"
                />
                <ColorRanges>   User-defined color ranges. By default empty with no child element entries
                       <Entry
                              From="50. "   Double
                              FillWithColor="1"   Bool
                              Color="#00ff00"   Color
                       />
                       <Entry
                              From="50.0"
                              FillWithColor="1"
                              Color="#ff0000"
                       />
                       ...
                </ColorRanges>
        </Gauge>
</chart-config>
```

19.2.2.1.10.2 Beispiel: Diagrammfunktionen

Anhand des XSLT-Beispieldokuments weiter unten sehen Sie, wie Altova-Erweiterungsfunktionen für Diagramme[2074] eingesetzt werden können. Weiter unten sehen Sie ein XML-Dokument und eine Abbildung des Ausgabebilds, das generiert wird, wenn ein XML-Dokument mit dem XSLT 2.0-Prozessor anhand des XSLT-Dokuments verarbeitet wird.

Anmerkung: Diagrammfunktionen werden nur in **Enterprise und Server-Editionen** von Altova Produkten unterstützt.

Anmerkung: Weitere Informationen zur Erstellung von Diagrammdatentabellen finden Sie in der Dokumentation zu den Altova-Produkten XMLSpy und StyleVision.

XSLT-Dokument

In diesem (*unten aufgelisteten*) XSLT-Dokument wird mit Hilfe der Altova Diagramm-Erweiterungsfunktionen ein Kreisdiagramm generiert. Das XSLT-Dokument kann zur Verarbeitung des weiter unten angeführten XML-Dokuments verwendet werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:altovaext="http://www.altova.com/xslt-extensions"
    exclude-result-prefixes="#all">
    <xsl:output version="4.0" method="html" indent="yes" encoding="UTF-8"/>
    <xsl:template match="/">
        <html>
            <head>
                <title>
                    <xsl:text>HTML Page with Embedded Chart</xsl:text>
                </title>
            </head>
            <body>
                <xsl:for-each select="/Data/Region[1]">
                    <xsl:variable name="extChartConfig" as="item()*">
                        <xsl:variable name="ext-chart-settings" as="item()*">
                            <chart-config>
                                <General
                                    SettingsVersion="1"
                                    ChartKind="Pie3d"
                                    BKColor="#ffffff"
                                    ShowBorder="1"
                                    PlotBorderColor="#000000"
                                    PlotBKColor="#ffffff"
                                    Title="{@id}"
                                    ShowLegend="1"
                                    OutsideMargin="3.2%"
                                    TitleToPlotMargin="3.%"
                                    LegendToPlotMargin="6.%"
                                    >
                                    <TitleFont
                                        Color="#023d7d"
                                        Name="Tahoma"
                                        Bold="1"
                                        Italic="0"
                                        Underline="0"
                                        MinFontHeight="10.pt"
                                        Size="8.%" />
                                </General>
                            </chart-config>
                        </xsl:variable>
                        <xsl:sequence select="altovaext:create-chart-config-from-xml( $ext-
chart-settings )"/>
```


© 2016-2022 Altova GmbH

```
                </xsl:variable>
                <xsl:variable name="chartDataSeries" as="item()*">
                    <xsl:variable name="chartDataRows" as="item()*">
                        <xsl:for-each select="(Year)">
                            <xsl:sequence select="altovaext:create-chart-data-row( (@id),
( .) )"/>
                        </xsl:for-each>
                    </xsl:variable>
                    <xsl:variable name="chartDataSeriesNames" as="xs:string*"
select=" ( (&quot;Series 1&quot;), &apos;&apos; )[1]"/>
                    <xsl:sequence
                        select="altovaext:create-chart-data-series-from-
rows( $chartDataSeriesNames, $chartDataRows)"/>
                </xsl:variable>
                <xsl:variable name="ChartObj" select="altovaext:create-
chart( $extChartConfig, ( $chartDataSeries), false() )"/>
                <xsl:variable name="sChartFileName" select="'mychart1.png'"/>
                <img src="{$sChartFileName, altovaext:save-chart-image( $ChartObj,
$sChartFileName, 400, 400 ) }"/>
            </xsl:for-each>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

XML-Dokument

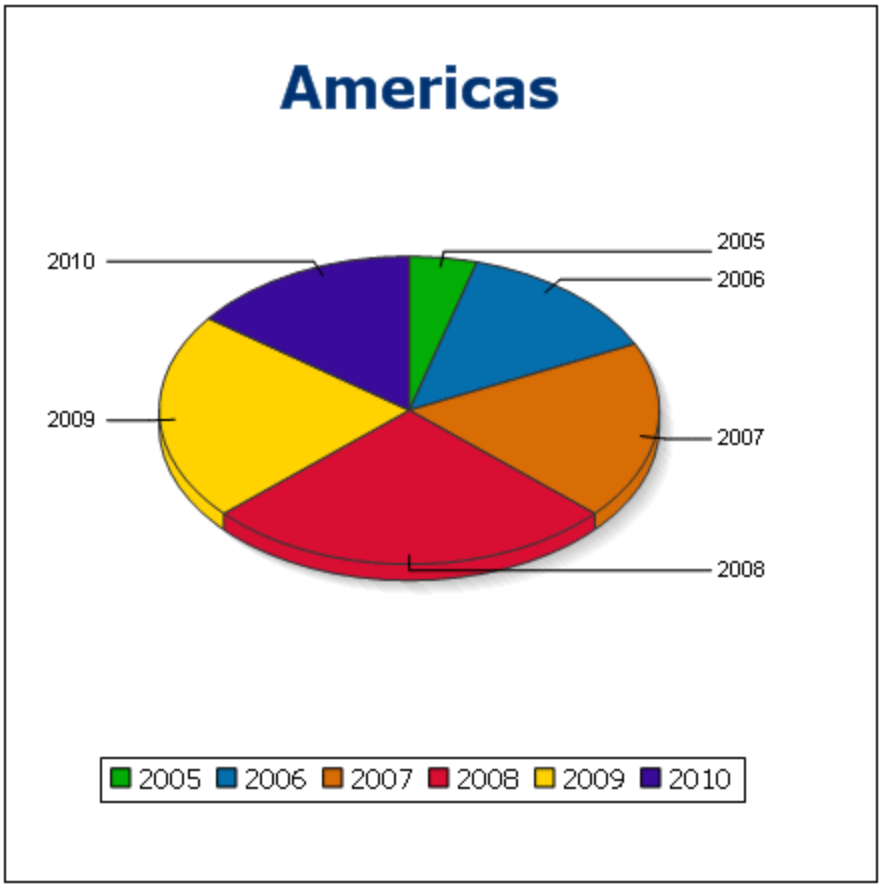
Dieses XML-Dokument kann mit dem oben stehenden XSLT-Dokument verarbeitet werden. Anhand der Daten im XML-Dokument wird das in der unten stehenden Abbildung gezeigte Kreisdiagramm generiert.

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
   <ChartType>Pie Chart 2D</ChartType>
   <Region id="Americas">
      <Year id="2005">30000</Year>
      <Year id="2006">90000</Year>
      <Year id="2007">120000</Year>
      <Year id="2008">180000</Year>
      <Year id="2009">140000</Year>
      <Year id="2010">100000</Year>
   </Region>
   <Region id="Europe">
      <Year id="2005">50000</Year>
      <Year id="2006">60000</Year>
      <Year id="2007">80000</Year>
      <Year id="2008">100000</Year>
      <Year id="2009">95000</Year>
      <Year id="2010">80000</Year>
   </Region>
   <Region id="Asia">
      <Year id="2005">10000</Year>
```

```
      <Year id="2006">25000</Year>
      <Year id="2007">70000</Year>
      <Year id="2008">110000</Year>
      <Year id="2009">125000</Year>
      <Year id="2010">150000</Year>
   </Region>
</Data>
```

Ausgabebild

Das unten gezeigt Kreisdiagramm wird generiert, wenn das oben aufgelistete XML-Dokument mit Hilfe des XSLT-Dokuments verarbeitet wird.



19.2.2.2 Diverse Erweiterungsfunktionen

Es gibt in Programmiersprachen wie Java und C# eine Reihe von fertigen Funktionen, die nicht als XQuery / XPath 2.0- oder XSLT-Funktionen zur Verfügung stehen. Ein gutes Beispiel dafür sind die mathematischen in Java verfügbaren Funktionen wie z.B. `sin()` und `cos()`. Stünden diese Funktionen für die Erstellung von XSLT Stylesheets und XQuery-Abfragen zur Verfügung, würde sich der Einsatzbereich von Stylesheets und Abfragen

© 2016-2022 Altova GmbH

erweitern und die Erstellung von Stylesheets wäre viel einfacher. Der in einer Reihe von Altova-Produkten verwendete XSLT- und XQuery-Prozessor von Altova unterstützt die Verwendung von Erweiterungsfunktionen in Java[2087] und .NET[2096] sowie MSXSL Skripts für XSLT[2103]. MSXSL-Skripts für XSLT[2103] und die Altova-Erweiterungsfunktionen[1995]. In diesem Abschnitt wird beschrieben, wie Sie Erweiterungsfunktionen und MSXSL-Skripts in Ihren XSLT Stylesheets und XQuery-Dokumenten verwenden können. Diese Beschreibungen finden Sie in den folgenden Abschnitten:

- Java-Erweiterungsfunktionen[2087]
- .NET-Erweiterungsfunktionen[2096]
- MSXSL-Skripts für XSLT[2103]

Hauptsächlich werden dabei die folgenden beiden Punkte behandelt: (i) Wie Funktionen in den entsprechenden Bibliotheken aufgerufen werden; und (ii) welche Regeln beim Konvertieren von Argumenten in einem Funktionsaufruf in das erforderliche Format der Funktion befolgt werden und welche Regeln bei der Rückwärtskonvertierung (Funktionsresultat in XSLT/XQuery Datenobjekt) befolgt werden.

Voraussetzungen

Damit die Erweiterungsfunktionen unterstützt werden, muss auf dem Rechner, auf dem die XSLT-Transformation oder die XQuery-Ausführung stattfindet, eine Java Runtime-Umgebung (zum Aufrufen der Java-Funktionen) und ein .NET Framework 2.0 (Mindestvoraussetzung für Zugriff auf .NET-Funktionen) installiert sein oder es muss Zugriff auf eine solche bestehen.

19.2.2.2.1 Java-Erweiterungsfunktionen

Eine Java-Erweiterungsfunktion kann in einem XPath- oder XQuery-Ausdruck verwendet werden, um einen Java-Konstruktor oder eine Java-Methode (statisch oder Instanz) aufzurufen.

Ein Feld in einer Java-Klasse wird als Methode ohne Argument betrachtet. Bei einem Feld kann es sich um ein statisches Feld oder eine Instanz handeln. Wie man Felder aufruft, wird in den entsprechenden Unterabschnitten zu statischen Feldern und Instanzen beschrieben.

Dieser Abschnitt enthält die folgenden Unterabschnitte:

- Java: Konstruktoren[2093]
- Java: Statische Methoden und statische Felder[2093]
- Java: Instanzmethoden und Instanzfelder[2094]
- Datentypen: XPath/XQuery in Java[2095]
- Datentypen: Java in XPath/XQuery[2096]

Beachten Sie die folgenden Punkte

- Wenn Sie ein Altova Desktop-Produkt verwenden, versucht die Altova-Applikation, den Pfad zur Java Virtual Machine automatisch zu ermitteln. Dazu wird zuerst (i) die Windows Registry und dann (ii) die **JAVA_HOME**-Umgebungsvariable gelesen. Sie können im Dialogfeld "Optionen" der Applikation auch einen benutzerdefinierten Pfad hinzufügen. Dieser Eintrag hat Vorrang vor allen anderen automatisch ermittelten Java VM-Pfaden.
- Wenn Sie ein Altova Server-Produkt auf einem Windows-Rechner ausführen, wird der Pfad zur Java Virtual Machine zuerst aus der Windows Registry ausgelesen. Falls dies nicht gelingt, wird die **JAVA_HOME**-Umgebungsvariable verwendet.

- Wenn Sie ein Altova Server-Produkt auf einem Linux- oder macOS-Rechner ausführen, sollten Sie sicherstellen, dass die **JAVA_HOME**-Umgebungsvariable ordnungsgemäß definiert ist und dass sich die Java Virtual Machine-Bibliothek (unter Windows die Datei **jvm.dll**) entweder im Verzeichnis **\bin\server** oder **\bin\client** befindet.

Form der Erweiterungsfunktion

Die Erweiterungsfunktion im XPath/XQuery-Ausdruck muss die folgenden Form haben `präfix:fname()`.

- Der Teil `präfix:` kennzeichnet die Erweiterungsfunktion als Java-Funktion, indem er die Erweiterungsfunktion mit einer in-scope Namespace-Deklaration verknüpft, deren URI mit `java:` beginnen muss (*Beispiele siehe unten*). Die Namespace-Deklaration sollte eine Java-Klasse bezeichnen, z.B:
  `xmlns:myns="java:java.lang.Math"`. Sie könnte aber auch einfach lauten:
  `xmlns:myns="java"` (ohne Doppelpunkt), wobei die Identifizierung der Java-Klasse dem `fname()` Teil der Erweiterungsfunktion überlassen bleibt.
- Der Teil `fname()` identifiziert die aufgerufene Java-Methode und liefert die Argumente für die Methode (*Beispiele siehe unten*). Wenn die durch das `prefix:` Teil identifizierte Namespace URI jedoch keine Java-Klasse bezeichnet (*siehe vorheriger Punkt),* dann sollte die Java-Klasse im `fname()` Teil vor der Klasse identifiziert werden und von der Klasse durch einen Punkt getrennt sein (*siehe zweites XSLT-Beispiel unten*).

Anmerkung: Die aufgerufene Klasse muss sich unter dem Klassenpfad des Rechners befinden.

XSLT-Beispiel

Hier sehen Sie zwei Beispiele dafür, wie eine statische Methode aufgerufen werden kann. Im ersten Beispiel ist der Klassenname (`java.lang.Math`) in der Namespace URI enthalten und darf daher nicht im `fname()` Teil enthalten sein. Im zweiten Beispiel liefert der `prefix:` Teil das Präfix `java:`, während der `fname()` Teil die Klasse sowie die Methode identifiziert.

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jmath="java"
              select="jmath:java.lang.Math.cos(3.14)" />
```

Die in der Erweiterungsfunktion (im Beispiel oben `cos()`) angegebene Methode muss mit dem Namen einer öffentlichen statischen Methode in der angegebenen Java-Klasse (im Beispiel oben `java.lang.Math`) übereinstimmen.

XQuery-Beispiel

Hier sehen Sie ein XQuery-Beispiel, das dem XSLT-Beispiel oben ähnlich ist:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```


© 2016-2022 Altova GmbH

Benutzerdefinierte Java-Klassen

Wenn Sie Ihre eigenen Java-Klassen erstellt haben, werden die Methoden in diesen Klassen unterschiedlich aufgerufen, je nachdem: (i) ob die Klassen über eine JAR-Datei oder eine Klassendatei aufgerufen werden, und (ii) ob sich diese Dateien (JAR oder Klasse) im aktuellen Verzeichnis befinden (im selben Verzeichnis wie das XSLT- oder XQuery-Dokument) oder nicht. Wie Sie diese Dateien finden, wird in den Abschnitten Benutzerdefinierte Klassendateien[2089] und Benutzerdefinierte Jar-Dateien[2092] beschrieben. Pfade zu Klassendateien, die sich nicht im aktuellen Verzeichnis befinden, und Pfade zu allen JAR-Dateien müssen jedoch angegeben werden.

19.2.2.2.1.1 Benutzerdefinierte Klassendateien

Wenn der Zugriff über eine Klassendatei erfolgt, gibt es vier Möglichkeiten:

- Die Klassendatei befindet sich in einem Paket. Die XSLT-oder XQuery-Datei befindet sich im selben Ordner wie das Java-Paket. (*Siehe Beispiel unten*[2089].)
- Die Klassendatei befindet sich nicht in einem Paket. Die XSLT-oder XQuery-Datei befindet sich im selben Ordner wie die Klassendatei. (*Siehe Beispiel unten*[2090].)
- Die Klassendatei befindet sich in einem Paket. Die XSLT-oder XQuery-Datei befindet sich in irgendeinem beliebig gewählten Ordner. (*Siehe Beispiel unten*[2090].)
- Die Klassendatei befindet sich nicht in einem Paket. Die XSLT-oder XQuery-Datei befindet sich in irgendeinem beliebig gewählten Ordner. (*Siehe Beispiel unten*[2091].)

Gesetzt der Fall, die Klassendatei befindet sich nicht in einem Paket, sondern im selben Ordner wie das XSLT- oder XQuery-Dokument, so muss der Dateipfad nicht angegeben werden, da alle Klassen im Ordner gefunden werden. Die Syntax zum Identifizieren einer Klasse lautet:

```
java:classname
```

wobei

`java:` angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird; (Java-Klassen im aktuellen Verzeichnis werden standardmäßig geladen)
`classname` der Name der Klasse der erforderlichen Methode ist

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem Methodenaufruf als Präfix vorangestellt wird.

Klassendatei in einem Paket, XSLT/XQuery-Datei befindet sich im selben Ordner wie das Java-Paket

Im Beispiel unten wird die Methode `getVehicleType()`der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Das Paket `com.altova.extfunc` befindet sich im Ordner `JavaProject`. Die XSLT-Datei befindet sich ebenfalls im Ordner `JavaProject`.

```
<xsl:stylesheet version="2.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:fn="http://www.w3.org/2005/xpath-functions"
        xmlns:car="java:com.altova.extfunc.Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

Die Klassendatei wird referenziert, die XSLT/XQuery-Datei befindet sich im selben Ordner wie die Klassendatei

Im Beispiel unten wird die Methode `getVehicleType()` der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Angenommen, (i) die Klassendatei `Car` class befindet sich im folgenden Ordner: `JavaProject/com/altova/extfunc` und (ii) dieser Ordner ist der aktuelle Ordner im Beispiel unten. Die XSLT-Datei befindet sich ebenfalls im Ordner `JavaProject/com/altova/extfunc`.

```
<xsl:stylesheet version="2.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:fn="http://www.w3.org/2005/xpath-functions"
        xmlns:car="java:Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

Die Klassendatei befindet sich in einem Paket, die XSLT/XQuery-Datei befindet sich in einem beliebigen Ordner

Im Beispiel unten wird die Methode `getCarColor()`der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Das Paket `com.altova.extfunc` befindet sich im Ordner `JavaProject`. Die XSLT-Datei befindet sich in einem beliebigen Ordner. In diesem Fall muss der Pfad des Pakets mit der URI als Abfragestring definiert werden. Die Syntax lautet:

```
java:classname[?path=uri-of-classfile]
```

wobei

`java:` angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird
`uri-of-classfile` die URI der Klassendatei ist

© 2016-2022 Altova GmbH

`classname` der Name der Klasse der benötigten Methode ist

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem Methodenaufruf als Präfix vorangestellt wird. Im Beispiel unten sehen Sie, wie eine Klassendatei aufgerufen wird, die sich in einem anderen als dem aktuellen Verzeichnis befindet.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:com.altova.extfunc.Car?path=file:///C:/JavaProject/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

Die Klassendatei wird referenziert, die XSLT/XQuery-Datei befindet sich in einem beliebigen Ordner

Im Beispiel unten wird die Methode `getCarColor()`der Klasse `Car` aufgerufen. Angenommen, die Klassendatei `Car` befindet sich im Ordner `C:/JavaProject/com/altova/extfunc`. Die XSLT-Datei befindet sich in einem beliebigen Ordner. Der Pfad der Klassendatei muss dann in der Namespace-URI als Abfragestring definiert werden. Die Syntax lautet:

```
java:classname[?path=<uri-of-classfile>]
```

wobei

`java:` angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird
`uri-of-classfile` die URI der Klassendatei ist
`classname` der Name der Klasse der benötigten Methode ist

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem Methodenaufruf als Präfix vorangestellt wird. Im Beispiel unten sehen Sie, wie eine Klassendatei aufgerufen wird, die sich in einem anderen als dem aktuellen Verzeichnis befindet.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:Car?path=file:///C:/JavaProject/com/altova/extfunc/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
```

```
    <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
  </xsl:template>

  </xsl:stylesheet>
```

Anmerkung: Wenn ein Pfad über eine Erweiterungsfunktion angegeben wird, wird er zum ClassLoader hinzugefügt.

19.2.2.2.1.2 Benutzerdefinierte Jar-Dateien

JAR-Dateien

Wenn der Zugriff über eine JAR-Datei erfolgt, muss die URI der JAR-Datei mit Hilfe der folgenden Syntax definiert werden:

```
xmlns:classNS="java:classname?path=jar:uri-of-jarfile!/"
```

Die Methode wird anschließend durch Verwendung des Präfix der Namespace URI aufgerufen, der die Klasse bezeichnet: `classNS:method()`

wobei im obigen Beispiel:

`java:` angibt, dass eine Java-Funktion aufgerufen wird
`classname` der Name der Klasse der benutzerdefinierten Klasse ist
`?` das Trennzeichen zwischen dem Klassennamen und dem Pfad ist
`path=jar:` angibt, dass es sich um einen Pfad zu einer JAR-Datei handelt
`uri-of-jarfile` die URI der jar-Datei angibt
`!/` das Trennzeichen am Ende des Pfades ist
`classNS:method()` der Aufruf der Methode ist

Alternativ dazu kann der Klassenname mit dem Methodenaufruf angegeben werden. Hier sehen Sie zwei Beispiele für die Syntax:

```
xmlns:ns1="java:docx.layout.pages?
path=jar:file:///c:/projects/docs/docx.jar!/"
ns1:main()

xmlns:ns2="java?path=jar:file:///c:/projects/docs/docx.jar!/"
ns2:docx.layout.pages.main()
```

Hier sehen Sie ein komplettes XSLT-Beispiel, in dem eine JAR-Datei verwendet wird, um eine Java-Erweiterungsfunktion aufzurufen.:

```
<xsl:stylesheet version="2.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:fn="http://www.w3.org/2005/xpath-functions"
    xmlns:car="java?path=jar:file:///C:/test/Car1.jar!/" >
<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
    <xsl:variable name="myCar" select="car:Car1.new('red')" />
    <a><xsl:value-of select="car:Car1.getCarColor($myCar)"/></a>
```


© 2016-2022 Altova GmbH

```
</xsl:template>

<xsl:template match="car"/>

</xsl:stylesheet>
```

Anmerkung: Wenn ein Pfad über eine Erweiterungsfunktion angegeben wird, wird er zum ClassLoader hinzugefügt.

19.2.2.2.1.3 Java: Konstruktoren

Eine Erweiterungsfunktion kann dazu verwendet werden, um einen Java-Konstruktor aufzurufen. Alle Konstruktoren werden mit der Pseudofunktion `new()` aufgerufen.

Wenn das Ergebnis eines Java-Konstruktors implizit in XPath/XQuery-Datentypen konvertiert werden kann[2096], dann gibt die Java-Erweiterungsfunktion eine Sequenz zurück, bei der es sich um einem XPath/XQuery-Datentyp handelt. Wenn das Ergebnis eines Konstruktoraufrufs nicht in einen passenden XPath/XQuery-Datentyp konvertiert werden kann, dann erstellt der Konstruktor ein wrapped Java-Objekt mit einem Typ, der den Namen der Klasse hat, die dieses Java-Objekt zurückgibt. Wenn z.B. ein Konstruktor für die Klasse `java.util.Date` aufgerufen wird (`java.util.Date.new()`), so wird ein Objekt vom Typ `java.util.Date` zurückgegeben. Das lexikalische Format des zurückgegebenen Objekts stimmt unter Umständen nicht mit dem lexikalischen Format des XPath-Datentyps überein und der Wert müsste daher in das lexikalische Format des erforderlichen XPath-Datentyps und anschließend in den erforderlichen XPath-Datentyp konvertiert werden.

Ein von einem Konstruktor erstelltes Java-Objekt kann für zwei Zwecke verwendet werden:

- Es kann einer Variable zugewiesen werden:
  ```
  <xsl:variable name="currentdate" select="date:new()"
  xmlns:date="java:java.util.Date" />
  ```
- Es kann an eine Erweiterungsfunktion übergeben werden (*siehe Instanzmethode und Instanzfelder*[2094]):
  ```
  <xsl:value-of select="date:toString(date:new())" xmlns:date="java:java.util.Date" />
  ```

19.2.2.2.1.4 Java: Statische Methoden und statische Felder

Eine statische Methode wird direkt über ihren Java-Namen und durch Angabe der Argumente für die Methode aufgerufen. Statische Felder (Methoden, die keine Argumente haben), wie z.B. die Konstantenwertfelder `E` und `PI` werden ohne Angabe eines Arguments aufgerufen.

XSLT-Beispiele

Hier sehen Sie einige Beispiele dafür, wie statische Methoden und Felder aufgerufen werden können:

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />
```

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos( jMath:PI() )" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:E() * jMath:cos(3.14)" />
```

Beachten Sie, dass die Erweiterungsfunktionen die Form `prefix:fname()` haben. Das Präfix ist in allen drei Fällen `jMath:`. Es ist mit der Namespace URI `java:java.lang.Math` verknüpft. (Die Namespace URI muss mit `java:`. beginnen. In den obigen Beispielen wurde es um den Klassennamen erweitert (`java.lang.Math`).) Der Teil `fname()` der Erweiterungsfunktionen muss mit dem Namen der öffentlichen Klasse (z.B. `java.lang.Math`) gefolgt vom Namen einer öffentlichen statischen Methode mit ihrem/ihren Argument(en) (wie z.B.`(3.14)`) oder einem öffentlichen statischen Feld (z.B. `PI()`) übereinstimmen.

In den obigen Beispielen wurde der Klassenname in die Namespace URI inkludiert. Wäre sie nicht in der Namespace URI enthalten, müsste sie in den `fname()` Teil der Erweiterungsfunktion inkludiert werden. Z.B:

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

XQuery-Beispiel

Ein ähnliches Beispiel in XQuery wäre:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

19.2.2.2.1.5 Java: Instanzmethoden und Instanzfelder

Bei einer Instanzmethode wird als erstes Argument eines Methodenaufrufs ein Java-Objekt an die Methode übergeben. Ein solches Java-Objekt würde normalerweise mit Hilfe einer Erweiterungsfunktion (z.B. eines Konstruktoraufrufs) oder eines Stylesheet-Parameters/einer Stylesheet-Variablen erstellt. Ein XSLT-Beispiel dafür wäre:

```
<xsl:stylesheet version="1.0" exclude-result-prefixes="date"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:date="java:java.util.Date"
  xmlns:jlang="java:java.lang">
  <xsl:param name="CurrentDate" select="date:new()"/>
  <xsl:template match="/">
      <enrollment institution-id="Altova School"
                date="{date:toString($CurrentDate)}"
                type="{jlang:Object.toString(jlang:Object.getClass( date:new() ))}">
      </enrollment>
  </xsl:template>
</xsl:stylesheet>
```

Im Beispiel oben wird der Wert des Node `enrollment/@type` folgendermaßen erstellt:

© 2016-2022 Altova GmbH

1. Es wird ein Objekt mit einem Konstruktor für die Klasse `java.util.Date` (mit dem Konstruktor `date:new()`) erstellt.
2. Dieses Java-Objekt wird als das Argument der Methode `jlang.Object.getClass` übergeben.
3. Das mit der Methode `getClass` abgerufene Objekt wird als das Argument an die Methode `jlang.Object.toString` übergeben.

Das Ergebnis (der Wert von `@type`) ist ein String, der den Wert `java.util.Date` hat.

Ein Instanzfeld unterscheidet sich theoretisch insofern von einer Instanzmethode, als es sich nicht um ein Java-Objekt per se handelt, das als Argument an das Instanzfeld übergeben wird. Stattdessen wird ein Parameter oder eine Variable als Argument übergeben. Der Parameter/die Variable kann allerdings selbst den Wert enthalten, der von einem Java-Objekt zurückgegeben wird. So erhält z.B. der Parameter `CurrentDate` den Wert, der von einem Konstruktor für die Klasse `java.util.Date` zurückgegeben wird. Dieser Wert wird anschließend als Argument an die Instanzmethode `date:toString` übergeben, um den Wert von `/enrollment/@date` bereitzustellen.

19.2.2.2.1.6 Datentypen: XPath/XQuery in Java

Wenn von einem XPath/XQuery-Ausdruck aus eine Java-Funktion aufgerufen wird, spielt der Datentyp der Argumente der Funktion eine wichtige Rolle, welche von mehreren Java-Klassen desselben Namens aufgerufen wird.

In Java gelten die folgenden Regeln:

- Wenn es mehr als eine Java-Methode mit demselben Namen gibt, jede aber eine andere Anzahl von Argumenten als die andere(n) hat, so wird die Java-Methode ausgewählt, die der Anzahl der Argumente im Funktionsaufruf am ehesten entspricht.
- Die XPath/XQuery-Datentypen "string", "number" und "boolean" (*siehe Liste unten*) werden implizit in einen entsprechenden Java-Datentyp konvertiert. Wenn der bereitgestellte XPath/XQuery-Datentyp in mehr als einen Java-Typ konvertiert werden kann (z.B: `xs:integer`), so wird jener Java-Typ ausgewählt, der für die ausgewählte Methode deklariert wurde. Wenn die aufgerufene Java-Methode z.B. `fx(decimal)` und der bereitgestellte XPath/XQuery-Datentyp `xs:integer` ist, so wird `xs:integer` in den Java-Datentyp `decimal` konvertiert.

In der Tabelle unten sehen Sie eine Liste der impliziten Konvertierungen der XPath/XQuery-Datentypen "string", "number" und "boolean"in Java-Datentypen.

`xs:string`	`java.lang.String`
`xs:boolean`	`boolean` (primitive), `java.lang.Boolean`
`xs:integer`	int, long, short, byte, float, double und die Wrapper-Klassen davon wie z.B. `java.lang.Integer`
`xs:float`	`float` (primitive), `java.lang.Float, double` (primitive)
`xs:double`	`double` (primitive), `java.lang.Double`

`xs:decimal`	`float` (primitive), `java.lang.Float`, `double`(primitive), `java.lang.Double`

Die oben aufgelisteten Subtypen von XML-Schema-Datentypen (die in XPath und XQuery verwendet werden) werden ebenfalls in den/die Java-Typ(en), der/die dem übergeordneten Subtyp entsprechen, konvertiert.

In einigen Fällen ist es nicht möglich, auf Basis der verfügbaren Informationen die richtige Java-Methode auszuwählen. Nehmen Sie als Beispiel den folgenden Fall.

- Das bereitgestellte Argument ist ein `xs:untypedAtomic` Wert 10 und ist für die Methode `mymethod(float)` bestimmt.
- Es gibt jedoch eine weitere Methode in der Klasse, die ein Argument eines anderen Datentyps erhält: `mymethod(double)`.
- Da die Methodennamen dieselben sind und der bereitgestellte Typ (`xs:untypedAtomic`) sowohl in `float` als auch `double` korrekt konvertiert werden könnte, kann es geschehen, dass `xs:untypedAtomic` in `double` anstelle von `float` konvertiert wird.
- Infolgedessen handelt es sich dann bei der ausgewählten Methode nicht um die benötigte Methode, sodass nicht das erwartete Ergebnis erzielt wird. Als Umgehungslösung können Sie eine benutzerdefinierte Methode mit einem anderen Namen erstellen und diese Methode verwenden.

Typen, die in der Liste oben nicht enthalten sind (z.B. `xs:date`), werden nicht konvertiert und generieren einen Fehler. Beachten Sie jedoch, dass es in einigen Fällen unter Umständen möglich ist, den benötigten Java-Typ mittels eines Java-Konstruktors zu erstellen.

19.2.2.2.1.7 Datentypen: Java in XPath/XQuery

Wenn eine Java-Methode einen Wert zurückgibt und der Datentyp des Werts "string", "numeric" oder "boolean" ist, wird anschließend in den entsprechenden XPath/XQuery-Typ konvertiert. So werden z.B. die Java-Datentypen `javaa.lang.Boolean` und `boolean` in `xsd:boolean` konvertiert.

Von Funktionen zurückgegebene eindimensionale Arrays werden zu einer Sequenz erweitert. Mehrdimensionale Arrays werden nicht konvertiert und sollten daher in einen Wrapper gesetzt werden.

Wenn ein wrapped Java-Objekt oder ein Datentyp zurückgegeben wird, bei dem es sich nicht um den Typ "string", "numeric" oder "boolean" handelt, können Sie sicherstellen, dass die Konvertierung in den benötigten XPath/XQuery-Typ erfolgt, indem Sie zuerst eine Java-Methode (e.g `toString`) verwenden, um das Java-Objekt in einen String zu konvertieren. In XPath/XQuery kann der String geändert werden, damit er der lexikalischen Darstellung des benötigten Typs entspricht, und anschließend z.B. mit Hilfe des Ausdrucks `cast as` in den benötigten Typ konvertiert werden.

19.2.2.2.2 .NET-Erweiterungsfunktionen

Wenn Sie auf einem Windows-Rechner mit der .NET-Plattform arbeiten, können Sie Erweiterungsfunktionen verwenden, die in jeder beliebigen der .NET-Sprachen geschrieben wurden (z.B. C#). Eine .NET Erweiterungsfunktion kann in einem XPath- oder XQuery-Ausdruck verwendet werden, um einen Konstruktor, eine Eigenschaft oder Methode (statische oder Instanz) in einer .NET-Klasse aufzurufen.

© 2016-2022 Altova GmbH

Eine Eigenschaft einer .NET-Klasse wird mit der Syntax `get_PropertyName()` aufgerufen.

Dieser Abschnitt ist in die folgenden Unterabschnitte gegliedert:

- .NET: Konstruktoren 2099
- .NET: Statische Methoden und statische Felder 2100
- .NET: Instanzmethoden und Instanzfelder 2100
- Datentypen: XPath/XQuery in .NET 2101
- Datentypen: .NET in XPath/XQuery 2102

Form der Erweiterungsfunktion

Die Erweiterungsfunktion im XPath/XQuery-Ausdruck muss die folgende Form haben `präfix:fname()`.

- Der Teil `präfix:` ist mit einer URI verknüpft, die die benötigte .NET-Klasse definiert.
- Der Teil `fname()` identifiziert den Konstruktor, die Eigenschaft oder die Methode (statisch oder Instanz) innerhalb der .NET-Klasse und liefert alle gegebenenfalls benötigten Argumente.

- Die URI muss mit `clitype:` beginnen (welches die Funktion als .NET-Erweiterungsfunktion kennzeichnet).
- Die Form `prefix:fname()` der Erweiterungsfunktion kann mit Systemklassen und mit Klassen in einer geladenen Assembly verwendet werden. Wenn eine Klasse allerdings geladen werden muss, müssen zusätzliche Parameter mit den benötigten Informationen bereitgestellt werden.

Parameter

Zum Laden einer Assembly werden die folgenden Parameter verwendet:

`asm`	Der Name der zu ladenden Assembly
`ver`	Die Versionsnummer: eine Maximalzahl von vier Ganzzahlen, die durch Punkte getrennt sind
`sn`	Das Key Token des Strong Name der Assembly (16 Hex-Stellen).
`from`	Eine URI gibt den Pfad der zu ladenden Assembly (DLL) an. Wenn die URI relativ ist, ist sie relativ zum XSLT- oder XQuery-Dokument. Wenn dieser Parameter vorhanden ist, werden alle anderen Parameter ignoriert.
`partialname`	Der partielle Name der Assembly. Er wird für `Assembly.LoadWith.PartialName()` bereitgestellt, welches versuchen wird, die Assembly zu laden. Wenn `partialname` vorhanden ist, werden alle anderen Parameter ignoriert.
`loc`	Die Locale, z.B. `en-US`. Die Standardeinstellung ist `neutral`

Wenn die Assembly aus einer DLL geladen werden soll, verwenden Sie den `from` Parameter und lassen Sie den `sn` Parameter weg. Wenn die Assembly aus dem Global Assembly Cache (GAC) geladen werden soll, verwenden Sie den `sn` Parameter und lassen Sie den `from` Parameter weg.

Vor dem ersten Parameter muss ein Fragezeichen eingefügt werden. Parameter müssen durch ein Semikolon getrennt werden. Der Wert des Parameternamens wird durch ein Ist-Gleich-Zeichen angegeben (*siehe Beispiele unten*).

Beispiele für Namespace-Deklarationen

Ein Beispiel für eine Namespace Deklaration in XSLT, die die Systemklasse `System.Environment`: identifiziert.

```
xmlns:myns="clitype:System.Environment"
```

Ein Beispiel für eine Namespace Deklaration in XSLT, die die zu ladende Klasse als `Trade.Forward.Scrip`: identifiziert.

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

Ein Beispiel für eine Namespace-Deklaration in XQuery, die die Systemklasse `MyManagedDLL.testClass` identifiziert. Es werden zwei Klassen unterschieden:

1. Wenn die Assembly aus dem GAC geladen wird:
   ```
   declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
         ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
   ```

2. Wenn die Assembly aus der DLL geladen wird (vollständige und partielle Referenzen unten):
   ```
   declare namespace cs="clitype:MyManagedDLL.testClass?from=file:///C:/Altova
   Projects/extFunctions/MyManagedDLL.dll;

   declare namespace cs="clitype:MyManagedDLL.testClass?from=MyManagedDLL.dll;
   ```

XSLT-Beispiel

Hier sehen Sie ein vollständiges XSLT-Beispiel, in dem Funktionen in der Systemklasse `System.Math`: aufgerufen werden:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes" />
   <xsl:template match="/">
      <math xmlns:math="clitype:System.Math">
         <sqrt><xsl:value-of select="math:Sqrt(9)"/></sqrt>
         <pi><xsl:value-of select="math:PI()"/></pi>
         <e><xsl:value-of select="math:E()"/></e>
         <pow><xsl:value-of select="math:Pow(math:PI(), math:E())"/></pow>
      </math>
   </xsl:template>
</xsl:stylesheet>
```

Die Namespace-Deklaration für das Element `math` verknüpft das Präfix `math:` mit der URI `clitype:System.Math`. Der Beginn der URI `clitype:` gibt an, dass danach entweder eine Systemklasse oder eine geladene Klasse definiert wird. Das Präfix `math:` im XPath-Ausdruck verknüpft die Erweiterungsfunktionen mit der URI (und durch Erweiterung der Klasse) `System.Math`. Die Erweiterungsfunktionen identifizieren Methoden in der Klasse `System.Math` und stellen Argumente bereit, wo dies erforderlich ist.

© 2016-2022 Altova GmbH

XQuery-Beispiel

Hier sehen Sie ein XQuery-Beispielfragment ähnlich dem XSLT-Beispiel oben:

```
<math xmlns:math="clitype:System.Math">
   {math:Sqrt(9)}
</math>
```

Wie beim XSLT-Beispiel weiter oben identifiziert die Namespace-Deklaration die .NET-Klasse, in diesem Fall eine Systemklasse. Der XQuery-Ausdruck identifiziert die aufzurufenden Methode und liefert das Argument.

19.2.2.2.2.1 .NET: Konstruktoren

Eine Erweiterungsfunktion kann verwendet werden, um einen .NET-Konstruktor aufzurufen. Alle Konstruktoren werden mit der Pseudofunktion `new()` aufgerufen. Wenn es mehrere Konstruktoren für eine Klasse gibt, wird der Konstruktor ausgewählt, der der Anzahl der bereitgestellten Argumente am ehesten entspricht. Wenn kein passender Konstruktor gefunden wird, der den bereitgestellten Argumenten entspricht, wird die Fehlermeldung `'No constructor found'` zurückgegeben.

Konstruktoren, die XPath/XQuery-Datentypen zurückgeben

Wenn das Ergebnis eines .NET-Konstruktors implizit in XPath/XQuery-Datentypen konvertiert werden kann 2096, gibt die .NET-Erweiterungsfunktion eine Sequenz zurück, bei der es sich um einen XPath/XQuery-Datentyp handelt.

Konstruktoren, die .NET-Objekte zurückgeben

Wenn das Ergebnis eines .NET-Konstruktoraufrufs nicht in einen passenden XPath/XQuery-Datentyp konvertiert werden kann, erstellt der Konstruktor ein wrapped .NET-Objekt mit einem Typ, der der Name der Klasse ist, die dieses Objekt zurückgibt. Wenn z.B. ein Konstruktor für die Klasse `System.DateTime` aufgerufen wird (mit `System.DateTime.new()`), so wird ein Objekt mit dem Typ `System.DateTime` zurückgegeben.

Das lexikalische Format des zurückgegebenen Objekts stimmt unter Umständen nicht mit dem lexikalischen Format eines erforderlichen XPath-Datentyps überein. In solchen Fällen müsste der zurückgegebene Wert: (i) in das lexikalische Format des benötigten XPath-Datentyps konvertiert werden; und (ii) in den erforderlichen XPath-Datentyp konvertiert werden.

Ein von einem Konstruktor erstelltes .NET-Objekt kann für drei Zwecke verwendet werden:

- Es kann innerhalb einer Variable verwendet werden:
  ```
  <xsl:variable name="currentdate" select="date:new(2008, 4, 29)"
  xmlns:date="clitype:System.DateTime" />
  ```
- Es kann an eine Erweiterungsfunktion übergeben werden (*siehe* *Instanzmethode und Instanzfelder* 2094):
  ```
  <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
  xmlns:date="clitype:System.DateTime" />
  ```
- Es kann in einen String, eine Zahl oder einen Booleschen Ausdruck konvertiert werden:

- `<xsl:value-of select="xs:integer(date:get_Month(date:new(2008, 4, 29)))" xmlns:date="clitype:System.DateTime" />`

19.2.2.2.2.2 .NET: Statische Methoden und statische Felder

Eine statische Methode wird direkt über ihren Namen und durch Angabe der Argumente für die Methode aufgerufen. Der im Aufruf verwendete Name muss exakt mit einer öffentlichen statischen Methode in der angegebenen Klasse übereinstimmen. Wenn der Methodenname und die Anzahl der in der Funktion angegebenen Argumente mit mehr als einer Methode in einer Klasse übereinstimmen, werden die Typen der bereitgestellten Argumente nach der besten Übereinstimmung überprüft. Wenn keine eindeutig passende Methode gefunden werden kann, wird ein Fehler ausgegeben.

Anmerkung: Ein Feld in einer .NET-Klasse wird als Methode ohne Argument betrachtet. Eine Eigenschaft wird mit der Syntax `get_PropertyName()` aufgerufen.

Beispiele

Ein XSLT-Beispiel, in dem Sie einen Methodenaufruf mit einem Argument (`System.Math.Sin(arg)`) sehen:

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

Ein XSLT-Beispiel, in dem Sie einen Aufruf eines Felds (wird als Methode ohne Argument betrachtet) sehen (`System.Double.MaxValue()`):

```
<xsl:value-of select="double:MaxValue()" xmlns:double="clitype:System.Double"/>
```

Ein XSLT-Beispiel, in dem Sie einen Aufruf einer Eigenschaft (Syntax ist `get_PropertyName()`) (`System.String()`) sehen:

```
<xsl:value-of select="string:get_Length('my string')"
xmlns:string="clitype:System.String"/>
```

Ein XQuery-Beispiel, in dem Sie einen Aufruf einer Methode mit einem Argument (`System.Math.Sin(arg)`) sehen:

```
<sin xmlns:math="clitype:System.Math">
   { math:Sin(30) }
</sin>
```

19.2.2.2.2.3 .NET: Instanzmethoden und Instanzfelder

Bei einer Instanzmethode wird als erstes Argument des Methodenaufrufs ein .NET-Objekt an die Methode übergeben. Dieses .NET-Objekt wird normalerweise mit Hilfe einer Erweiterungsfunktion (z.B. durch einen

© 2016-2022 Altova GmbH

Konstruktoraufruf) oder einen Stylesheet-Parameter/eine Stylesheet-Variable erstellt. Ein XSLT-Beispiel dieser Art wäre:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes"/>
   <xsl:template match="/">
      <xsl:variable name="releasedate"
         select="date:new(2008, 4, 29)"
         xmlns:date="clitype:System.DateTime"/>
      <doc>
         <date>
            <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
               xmlns:date="clitype:System.DateTime"/>
         </date>
         <date>
            <xsl:value-of select="date:ToString($releasedate)"
               xmlns:date="clitype:System.DateTime"/>
         </date>
      </doc>
   </xsl:template>
</xsl:stylesheet>
```

Im Beispiel oben wird ein `System.DateTime` Konstruktor (`new(2008, 4, 29)`) verwendet, um ein .NET-Objekt vom Typ `System.DateTime` zu erstellen. Diese Objekt wird zweimal erstellt, einmal als Wert der Variablen `releasedate`, ein zweites Mal als das erste und einzige Argument der Methode `System.DateTime.ToString()`. Die Instanzmethode `System.DateTime.ToString()` wird zwei Mal aufgerufen, beide Male mit dem `System.DateTime` Konstruktor (`new(2008, 4, 29)`) als erstem und einzigem Argument. In einer dieser Instanzen wird die Variable `releasedate` verwendet, um das .NET-Objekt abzurufen.

Instanzmethoden und Instanzfelder

Der Unterschied zwischen einer Instanzmethode und einem Instanzfeld ist ein theoretischer. In einer Instanzmethode wird ein .NET-Objekt direkt als Argument übergeben; in einem Instanzfeld wird stattdessen ein Parameter oder eine Variable übergeben - auch wenn der Parameter bzw. die Variable selbst ein .NET-Objekt enthalten kann. So enthält z.B. die Variable `releasedate` im Beispiel oben ein .NET-Objekt und es ist diese Variable, die als das Argument von `ToString()` an den zweiten `date` Elementkonstruktor übergeben wird. Die `ToString()` Instanz im ersten `date` Element ist daher eine Instanzmethode, während die zweite als Instanzfeld betrachtet wird. Das in beiden Instanzen erzeugte Ergebnis ist jedoch dasselbe.

19.2.2.2.2.4 Datentypen: XPath/XQuery in .NET

Wenn in einem XPath/XQuery-Ausdruck eine .NET-Erweiterungsfunktion verwendet wird, spielen die Datentypen der Argumente der Funktion eine wichtige Rolle bei der Entscheidung, welche der vielen .NET-Methoden mit demselben Namen aufgerufen werden soll.

In .NET gelten die folgenden Regeln:

- Wenn es mehr als eine Methode mit demselben Namen in einer Klasse gibt, so stehen nur die Methoden zur Auswahl, die dieselbe Anzahl von Argumenten wie der Funktionsaufruf haben.
- Die XPath/XQuery-Datentypen "string", "number" und "boolean" (*siehe Liste unten*) werden implizit in einen entsprechenden .NET-Datentyp konvertiert. Wenn der bereitgestellte XPath/XQuery-Datentyp in mehr als einen .NET-Typ konvertiert werden kann (z.B: `xs:integer`), so wird jener .NET-Typ ausgewählt, der für die ausgewählte Methode deklariert wurde. Wenn die aufgerufene .NET-Methode z.B. `fx(double)` und der bereitgestellte XPath/XQuery-Datentyp `xs:integer` ist, so wird `xs:integer` in den .NET-Datentyp `double`

In der Tabelle unten sehen Sie eine Liste der impliziten Konvertierungen der XPath/XQuery-Datentypen "string", "number" und "boolean"in .NET-Datentypen.

`xs:string`	`StringValue, string`
`xs:boolean`	`BooleanValue, bool`
`xs:integer`	`IntegerValue, decimal, long, integer, short, byte, double, float`
`xs:float`	`FloatValue, float, double`
`xs:double`	`DoubleValue, double`
`xs:decimal`	`DecimalValue, decimal, double, float`

Die oben aufgelisteten Subtypen von XML-Schema-Datentypen (die in XPath und XQuery verwendet werden) werden ebenfalls in den/die .NET-Typ(en), der/die dem übergeordneten Subtyp entsprechen, konvertiert.

In einigen Fällen ist es nicht möglich, auf Basis der verfügbaren Informationen die richtige .NET-Methode auszuwählen. Nehmen Sie als Beispiel den folgenden Fall.

- Das bereitgestellte Argument ist ein `xs:untypedAtomic` Wert 10 und ist für die Methode `mymethod(float)` bestimmt.
- Es gibt jedoch eine weitere Methode in der Klasse, die ein Argument eines anderen Datentyps erhält: `mymethod(double)`.
- Da die Methodennamen dieselben sind und der bereitgestellte Typ (`xs:untypedAtomic`) sowohl in `float` als auch `double` korrekt konvertiert werden könnte, kann es geschehen, dass `xs:untypedAtomic` in `double` anstelle von `float` konvertiert wird.
- Infolgedessen handelt es sich dann bei der ausgewählten Methode nicht um die benötigte Methode, sodass nicht das erwartete Ergebnis erzielt wird. Als Umgehungslösung können Sie eine benutzerdefinierte Methode mit einem anderen Namen erstellen und diese Methode verwenden.

Typen, die in der Liste oben nicht enthalten sind (z.B. `xs:date`), werden nicht konvertiert und generieren einen Fehler.

19.2.2.2.2.5 Datentypen: .NET in XPath/XQuery

Wenn eine .NET-Methode einen Wert zurückgibt und der Datentyp des Werts "string", "numeric" oder "boolean" ist, wird er anschließend in den entsprechenden XPath/XQuery-Typ konvertiert. So wird z.B. der .NET-Datentyp `decimal` in `xsd:decimal` konvertiert.

© 2016-2022 Altova GmbH

Wenn ein .NET-Objekt oder ein Datentyp zurückgegeben wird, bei dem es sich nicht um den Typ "string", "numeric" oder "boolean" handelt, können Sie sicherstellen, dass die Konvertierung in den benötigten XPath/XQuery-Typ erfolgt, indem Sie zuerst eine .NET-Methode (z.B. `System.DateTime.ToString()`) verwenden, um das .NET-Objekt in einen String zu konvertieren. In XPath/XQuery kann der String geändert werden, damit er der lexikalischen Darstellung des benötigten Typs entspricht, und anschließend z.B. mit Hilfe des Ausdrucks `cast as` in den benötigten Typ konvertiert werden.

19.2.2.2.3 MSXSL-Skripts für XSLT

Das Element `<msxsl:script>` enthält benutzerdefinierte Funktionen und Variablen, die von XPath-Ausdrücken im XSLT-Stylesheet aufgerufen werden können. Das Element `<msxsl:script>` ist ein Element der obersten Ebene, d.h. es muss ein Child-Element von `<xsl:stylesheet>` oder `<xsl:transform>` sein.

Das Element `<msxsl:script>` muss sich im Namespace `urn:schemas-microsoft-com:xslt` (*siehe Beispiel unten*) befinden.

Scripting-Sprache und Namespace

Die im Block verwendete Scripting-Sprache wird im Attribut `language` des Elements `<msxsl:script>` definiert und der für Funktionsaufrufe von XPath-Ausdrücken aus zu verwendende Namespace wird durch das Attribut `implements-prefix` (*siehe unten*) identifiziert.

```
<msxsl:script language="scripting-language implements-prefix="user-namespace-prefix">

   function-1 or variable-1
   ...
   function-n or variable-n

</msxsl:script>
```

Das Element `<msxsl:script>` interagiert mit der Windows Scripting Runtime. Daher können nur Sprachen, die auf Ihrem Rechner installiert sind, im Element `<msxsl:script>` verwendet werden. **Um MXSL Scripts verwenden zu können muss die Plattform .NET Framework 2.0 oder höher installiert sein.** Folglich können die .NET Scripting Sprachen innerhalb des Elements `<msxsl:script>` verwendet werden.

Das Attribut `language` akzeptiert dieselben Werte wie das Attribut `language` des HTML `<script>` Elements. Wenn das Attribut `language` nicht definiert ist, wird als Standardsprache Microsoft JScript verwendet.

Das Attribut `implements-prefix` erhält einen Wert, der ein Präfix eines deklarierten in-scope Namespace ist. Bei diesem Namespace handelt es sich normalerweise um einen Benutzer-Namespace, der für eine Funktionsbibliothek reserviert ist. Alle Funktionen und Variablen, die im Element `<msxsl:script>` definiert sind, werden sich im Namespace befinden, der durch das im Attribut `implements-prefix` definierte Präfixe identifiziert wird. Wenn eine Funktion von einem XPath-Ausdruck aus aufgerufen wird, muss sich der vollständig qualifizierte Funktionsname im selben Namespace wie die Funktionsdefinition befinden.

Beispiel

Hier sehen Sie ein Beispiel für ein vollständiges XSLT Stylesheet, in dem eine Funktion verwendet wird, die in einem <msxsl:script> Element definiert ist.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:msxsl="urn:schemas-microsoft-com:xslt"
   xmlns:user="http://mycompany.com/mynamespace">

  <msxsl:script language="VBScript" implements-prefix="user">
    <![CDATA[
    ' Input: A currency value: the wholesale price
    ' Returns: The retail price: the input value plus 20% margin,
    ' rounded to the nearest cent
    dim a as integer  = 13
    Function AddMargin(WholesalePrice) as integer
      AddMargin = WholesalePrice * 1.2 + a
    End Function
  ]]>
  </msxsl:script>

  <xsl:template match="/">
    <html>
      <body>
        <p>
          <b>Total Retail Price =
            $<xsl:value-of select="user:AddMargin(50)"/>
          </b>
          <br/>
          <b>Total Wholesale Price =
            $<xsl:value-of select="50"/>
          </b>
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Datentypen

Die Werte von Parametern, die an und aus dem Script-Block heraus übergeben werden, sind auf XPath-Datentypen beschränkt. Diese Einschränkung gilt nicht für Daten, die zwischen Funktionen und Variablen innerhalb des Script-Blocks übergeben werden.

© 2016-2022 Altova GmbH

Assemblies

Eine Assembly kann über das Element **msxsl:assembly** in das Script importiert werden. Die Assembly wird über einen Namen oder eine URL identifiziert. Die Assembly wird beim Kompilieren des Stylesheet importiert. Hier sehen Sie ein einfaches Beispiel, wie das Element `msxsl:assembly` zu verwenden ist.

```
<msxsl:script>
       <msxsl:assembly name="myAssembly.assemblyName" />
       <msxsl:assembly href="pathToAssembly" />

       ...

</msxsl:script>
```

Der Assembly-Name kann ein vollständiger Name sein, wie z.B.:

```
"system.Math, Version=3.1.4500.1 Culture=neutral PublicKeyToken=a46b3f648229c514"
```

oder ein Kurzname wie z.B. `"myAssembly.Draw"`.

Namespaces

Namespaces können mit dem Element **msxsl:using** deklariert werden. Auf diese Art können Assembly-Klassen ohne ihre Namespaces in das Script geschrieben werden, wodurch Sie sich das mühsame Eintippen ersparen. Hier sehen Sie, wie das Element `msxsl:using` verwendet wird, um Namespaces zu deklarieren.

```
<msxsl:script>
       <msxsl:using namespace="myAssemblyNS.NamespaceName" />

       ...

</msxsl:script>
```

Der Wert des `namespace` Attributs ist der Name des Namespace.

19.3 Technische Daten

Dieses Kapitel enthält Informationen zu einigen technischen Aspekten Ihrer Software. Es ist in die folgenden Abschnitte gegliedert:

- OS- und Arbeitsspeicheranforderungen 2106
- Altova-Prozessoren 2106
- Unicode-Unterstützung 2107
- Internet-Verwendung 2107

19.3.1 OS- und Arbeitsspeicheranforderungen

Betriebssystem

Die Altova-Software-Applikationen stehen für die folgenden Plattformen zur Verfügung:

- Windows 7 SP1 mit Plattformupdate, Windows 8, Windows 10, Windows 11
- Windows Server 2008 R2 SP1 mit Plattformupdate oder höher

Arbeitsspeicher

Da die Software in C++ geschrieben wurde, wird dafür nicht so viel Platz wie in einer Java Runtime Umgebung benötigt und normalerweise wird dafür weniger Arbeitsspeicher als bei einer vergleichbaren Java-basierten Applikation benötigt. Es ist notwendig, dass die einzelnen Dokumente in den Hauptarbeitsspeicher geladen werden, damit jedes Dokument zur Gänze geparst und analysiert werden kann und die Anzeige- und Bearbeitungsgeschwindigkeit während der normalen Arbeit verbessert wird. Daher steigt mit der Größe des Dokuments auch der Arbeitsspeicherbedarf.

Auch die unbegrenzte Rückgängig-Funktion kann einiges an Arbeitsspeicher in Anspruch nehmen. Wenn Sie große Abschnitte in großen Dokumenten immer wieder ausschneiden und einfügen, kann dies enorm viel Speicherplatz verbrauchen.

19.3.2 Altova-Prozessoren

XML Validator

Wenn Sie ein XML-Dokument öffnen, verwendet die Applikation den integrierten XML Validator, um das Dokument auf Wohlgeformtheit zu prüfen, es anhand eines Schemas zu validieren (falls eines angegeben wurde) und Baumstrukturen und Infosets zu erstellen. Der Altova XML Validator dient auch dazu, beim Editieren von Dokumenten intelligente Eingabehilfen zur Verfügung zu stellen und etwaige Validierungsfehler dynamisch anzuzeigen.

Im integrierten XML Validator ist die Final Recommendation der W3C XML Schema Spezifikationen 1.0 und 1.1 implementiert. Neue Entwicklungen, die von der XML Schema-Arbeitsgruppe empfohlen werden, werden ständig in den XML Validator integriert, sodass Ihnen mit Altova-Produkten immer eine Entwicklungsumgebung auf dem neuesten Stand der Technik zur Verfügung steht.

© 2016-2022 Altova GmbH

XSLT- und XQuery-Prozessor

Die Altova Produkte arbeiten mit dem Altova XSLT 1.0, 2.0 und 3.0-Prozessor und dem Altova XQuery 1.0 und 3.1-Prozessor. Falls einer dieser Prozessoren im Produkt enthalten ist, finden Sie die Dokumentation zurm implementierungsspezifischen Verhalten der einzelnen Prozessoren in den Anhängen zu dieser Dokumentation.

Anmerkung: Altova MapForce verwendet den XSLT 1.0-, 2.0- und XQuery 1.0-Prozessor zur Codegenerierung.

19.3.3 Unicode-Unterstützung

Die XML-Produkte von Altova bieten vollständige Unicode-Unterstützung. Um ein XML-Dokument benötigen Sie eine Schriftart, die die von diesem Dokument verwendeten Unicode-Zeichen unterstützt.

Beachten Sie bitte, dass die meisten Schriftarten nur eine bestimmte Untergruppe des gesamten Unicode-Bereichs enthalten und normalerweise für das entsprechende Schriftsystem ausgelegt sind. Wenn Zeichen falsch dargestellt werden, könnte der Grund darin liegen, dass die gewählte Schriftart die erforderlichen Glyphen nicht enthält. Es ist daher nützlich, eine Schriftart zu verwenden, die den gesamten Unicode-Bereich abdeckt - v.a. wenn Sie XML-Dokumente in unterschiedlichen Sprachen oder Schriftsystemen editieren. Ein typische auf Windows PCs installierte Unicode-Schriftart ist Arial Unicode MS.

Im Ordner`/Examples` Ihres Applikationsordners finden Sie eine XHTML-Datei mit dem Namen `UnicodeUTF-8.html`, die den folgenden Satz in verschiedenen Sprachen und Schriftsystemen enthält:

- *When the world wants to talk, it speaks Unicode*
- *Wenn die Welt miteinander spricht, spricht sie Unicode*
- 世界的に話すなら、Unicode です。)

Wenn Sie diese XHTML-Datei öffnen, erhalten Sie einen kurzen Eindruck davon, was mit Unicode möglich ist und welche Schriftsysteme von den auf Ihrem PC verfügbaren Schriftarten unterstützt werden.

19.3.4 Internet-Verwendung

Altova-Applikationen können für Sie auch eine Verbindung mit dem Internet herstellen. Dies geschieht in den folgenden Fällen:

- Wenn Sie im Registrierungsdialogfeld (**Hilfe | Software-Aktivierung**) auf "Kostenlosen Evaluierungs-Key anfordern" klicken, werden die drei Felder im Registrierungsdialogfeld über eine normale HTTP-Verbindung (Port 80) an unseren Webserver übertragen. Sie erhalten dann per E-Mail (normales SMTP) den kostenlosen Evaluierungs-Keycode zugesandt.
- In einige Altova-Produkten können Sie eine Datei über das Internet öffnen (**Datei | Öffnen | Zu URL wechseln**). In diesem Fall wird das Dokument mittels einer der folgenden Protokollmethoden und Verbindungen aufgerufen: HTTP (normalerweise Port 80), FTP (nomalerweise Port 20/21), HTTPS (normalerweise Port 443). Sie können auch einen HTTP-Server auf Port 8080 verwendden. (Definieren Sie den Port im Dialogfeld "URL", indem Sie ihn durch ein Komma getrennt nach dem Servernamen angeben.*)*

- Wenn Sie ein XML-Dokument öffnen, das sich auf ein XML-Schema oder eine DTD bezieht und das Dokument durch eine URL definiert wird, wird das referenzierte Schema-Dokument ebenfalls über eine HTTP-Verbindung (Port 80) oder ein anderes in der URL definiertes Protokoll (siehe Punkt 2 oben) aufgerufen. Ebenso wird ein Schema-Dokument aufgerufen, wenn eine XML-Datei validiert wird. Beachten Sie, dass die Validierung automatisch beim Öffnen des Dokuments erfolgen kann, wenn Sie dies in der Applikation so konfiguriert haben (Register "Datei" des Dialogfelds "Optionen" **Extras | Optionen**).
- Webservice-Verbindungen werden in Altova Applikationen, die WSDL und SOAP verwenden, mittels WSDL-Dokumenten definiert.
- Wenn Sie den Befehl "**Als Mail senden**..." (**Datei | Als Mail senden**) in XMLSpy verwenden, wird die aktuelle Auswahl bzw. Datei über ein MAPI-kompatibles Mail-Programm, das auf dem PC des Benutzers installiert ist, versendet
- Im Rahmen der Altova-Software-Lizenzvereinbarung beschriebenen Software-Aktivierung und beim Live-Update.

© 2016-2022 Altova GmbH

19.4 Lizenzinformationen

Dieser Anhang enthält die folgenden Informationen:

- Informationen über den Vertrieb dieses Software-Produkts
- Informationen zur Software-Aktivierung und Lizenzüberwachung
- die Lizenzvereinbarung zu diesem Software-Produkt

Lesen Sie die Informationen bitte sorgfältig - sie sind rechtlich bindend, da Sie sich bei der Installation dieses Software-Produkts damit einverstanden erklärt haben.

Den Inhalt aller Altova-Lizenzenvereinbarungen finden Sie auf der Altova Website unter Rechtliches.

19.4.1 Electronic Software Distribution

Dieses Produkt ist über EDS (Electronic Software Distribution), also auf elektronischem Weg erhältlich, eine Methode, die die folgenden einzigartigen Vorteile bietet:

- Sie können die Software kostenlos 30 Tage lang testen, bevor Sie sich zu einem Kauf entscheiden. (*Anmerkung: Die Lizenz für MobileTogether Designer ist kostenlos*.)
- Wenn Sie sich entschieden haben, die Software zu kaufen, können Sie Ihre Bestellung online auf der Altova Website tätigen. Sie erhalten dann innerhalb weniger Minuten ein vollständig lizenziertes Produkt.
- Sie erhalten immer die neueste Version unserer Software
- Die Software enthält ein umfassendes Hilfesystem, das Sie von der Benutzeroberfläche der Appikation aus aufrufen können. Die neueste Version des Benutzerhandbuchs steht auf unserer Website www.altova.com (i) im HTML-Format zum Aufrufen online und (ii) im PDF-Format zum Download und Ausdrucken zur Verfügung.

30-Tage-Evaluierungszeitraum

Nachdem Sie dieses Software-Produkt heruntergeladen haben, können Sie es 30 Tage lang kostenlos testen. Während dieses Zeitraums werden Sie nach etwa 20 Tagen in regelmäßigen Abständen daran erinnert, dass die Software noch nicht lizenziert wurde. Diese Erinnerungsmeldung wird allerdings nur einmal, nämlich bei jedem Start des Programms, angezeigt. Wenn Sie das Programm nach Ablauf des 30-tägigen Evaluierungszeitraums weiterhin verwenden möchten, müssen Sie eine Produktlizenz erwerben, die Sie in Form einer Lizenzdatei mit einem Keycode erhalten. Laden Sie die Lizenzdatei über das Dialogfeld "Software-Aktivierung" Ihres Produkts hoch, um das Produkt freizuschalten.

Sie können Ihre Produktlizenz über https://shop.altova.com/ erwerben.

Weitergabe der Software an andere Mitarbeiter in Ihrem Unternehmen zu Testzwecken

Wenn Sie die Evaluierungsversion der Software auch anderen Personen in Ihrem Unternehmen über das Netzwerk zur Verfügung stellen möchten oder wenn Sie sie auf einem PC installieren möchten, der nicht mit dem Internet verbunden ist, dürfen Sie nur das Installationsprogramm weitergeben, vorausgesetzt es wurde nicht modifiziert. Jeder, der das von Ihnen zur Verfügung gestellte Installationsprogramm aufruft, muss einen eigenen Evaluierungs-Keycode für 30 Tage anfordern. Nach Ablauf des Testzeitraums, muss eine Lizenz erworben werden, damit das Produkt weiter verwendet werden kann.

19.4.2 Software-Aktivierung und Lizenzüberwachung

Im Rahmen der Aktivierung der Software durch Altova, verwendet die Software unter Umständen Ihr internes Netzwerk und Ihre Internetverbindung, um die Lizenzdaten während der Installation, Registrierung, der Verwendung oder der Aktualisierung an einen von Altova betriebenen Lizenzserver zu übertragen und die Authentizität der Lizenzdaten zu überprüfen, damit Altova-Software nicht ohne Lizenz oder auf unzulässige Art und Weise verwendet werden kann und um den Kundenservice gleichzeitig zu verbessern. Bei der Aktivierung werden zwischen Ihrem Computer und dem Altova-Lizenzserver für die Lizenzierung erforderliche Daten wie Informationen über Betriebssystem, IP-Adresse, Datum/Uhrzeit, Software-Version und Computername sowie andere Informationen ausgetauscht.

Ihr Altova-Produkt verfügt über ein integriertes Lizenzüberwachungsmodul, das ebenfalls dazu beiträgt, unbeabsichtigte Verletzungen der Lizenzvereinbarung zu vermeiden. Ihr Produkt kann entweder mit einer Einzelplatzlizenz oder einer Mehrfachlizenz erworben werden. Je nach Lizenz stellt das Lizenzüberwachungsmodul sicher, dass nicht mehr als die lizenzierte Anzahl an Benutzern die Applikation gleichzeitig verwendet.

Bei dieser Lizenzüberwachungsmethode wird Ihr LAN-Netzwerk verwendet, um die Kommunikation zwischen Instanzen der Applikation, die auf verschiedenen Computern laufen, zu überwachen.

Einzelplatzlizenz

Beim Start der Applikation wird im Rahmen der Lizenzüberprüfung ein kurzes Broadcast-Datagramm abgesendet, um andere Instanzen des Produkts, die auf anderen Computern im selben Netzwerk laufen, zu finden. Wenn keine Antwort einlangt, wird ein Port geöffnet, der Informationen von anderen Instanzen der Applikation empfangen kann.

Mehrplatzlizenz

Wenn Sie im selben LAN mehrere Instanzen der Applikation verwenden, kommunizieren diese beim Start kurz miteinander, um Keycode-Informationen auszutauschen, damit Sie sicher sein können, dass nicht mehr als die lizenzierte Anzahl an Lizenzen gleichzeitig in Verwendung ist. Dieselbe Lizenzüberwachungstechnologie wird auch bei Unix und vielen anderen Datenbankentwicklungstools verwendet. Sie gestattet Benutzern den Erwerb von Parallellizenzen für mehrere Benutzer zu vernünftigen Preisen.

Wir sind außerdem bestrebt, nur wenige, kleine Netzwerkpakete zu versenden, um Ihr Netzwerk nicht zu überlasten. Die von Ihrem Altova Produkt verwendeten TCP/IP Ports (2799) sind offiziell bei IANA registriert, (*nähere Informationen siehe IANA Service Name Registry*) und unser Lizenzüberwachungsmodul basiert auf einer bewährten und erprobten Technologie.

Wenn Sie eine Firewall verwenden, werden Sie unter Umständen feststellen, dass die Computer, auf denen Altova-Produkte laufen, über Port 2799 miteinander kommunizieren. Sie können diesen Netzwerkverkehr

© 2016-2022 Altova GmbH

zwischen verschiedenen Gruppen in Ihrem Unternehmen natürlich blockieren, solange Sie mit anderen Mitteln sicherstellen können, dass Ihre Lizenzvereinbarung eingehalten wird.

Anmerkung zu Zertifikaten

Ihre Altova Applikation kontaktiert den Altova Lizenzierungsserver über HTTPS (`link.altova.com`). Für diese Kommunikation verwendet Altova ein registriertes SSL-Zertifikat. Wenn dieses Zertifikat ersetzt wird (z.B. von Ihrer IT-Abteilung oder einer externen Agentur), werden Sie von Ihrer Altova Applikation gewarnt, dass die Verbindung nicht sicher ist. Sie könnten Ihre Altova Applikation mit dem Ersetzungszertifikat starten. Dies würde jedoch auf Ihr eigenes Risiko geschehen. Wenn Sie eine Warnung sehen, dass die *Verbindung nicht sicher* ist, überprüfen Sie den Ursprung des Zertifikats und wenden Sie sich an Ihr IT-Team (die in der Lage sein sollten, zu entscheiden, ob das Abfangen und die Ersetzung des Altova-Zertifkats fortgesetzt werden soll).

Wenn Ihr Unternehmen sein eigenes Zertifikat verwenden muss (z.B. um die Kommunikation zu und von Client-Rechnern zu überwachen), empfehlen wir Ihnen, Altova LicenseServer, die kostenlose Lizenzverwaltungssoftware von Altova in Ihrem Netzwerk zu installieren. Client-Rechner verwenden mit dieser Konfiguration weiterhin die Zertifikate Ihres Unternehmens, während der Altova LicenseServer für die Kommunikation mit Altova das Altova-Zertifikat das Altova-Zertifikat verwenden kann.

19.4.3 Altova Endbenutzer-Lizenzvereinbarung

- Die Altova-Endbenutzer-Lizenzvereinbarung kann unter https://www.altova.com/de/legal/eula eingesehen werden.
- Die Altova-Datenschutzbestimmungen finden Sie unter https://www.altova.com/de/privacy.

Index

.

.NET Erweiterungsfunktionen,
Datentypkonvertierungen, .NET in XPath/XQuery, 2102
Datentypkonvertierungen, XPath/XQuery in .NET, 2101
für XSLT und XQuery, 2096
Instanzmethoden, Instanzfelder, 2100
Konstruktoren, 2099
statische Methoden, statische Felder, 2100
Übersicht, 2096

A

A bis Z,
Sortierkomponente, 908
Abgeleitete Typen,
mappen von/auf, 130
xsi:type, 130
abs,
als MapForce-Funktion (in lang | math functions), 1154
als MapForce-Funktion (in xpath2 | numeric functions), 1221
Access-Datenbank,
auf Basis von IF-Bedingung aktualisieren, 599
acos,
als MapForce-Funktion (in lang | math functions), 1154
ActiveX,
Integration auf Applikationsebene, 1924
Integration auf Dokumentebene, 1927
Voraussetzungen für die Integration, 1920
ActiveX Controls,
zur Visual Studio Toolbox hinzufügen, 1922
add,
als MapForce-Funktion (in core | math functions), 1063
ADO,
als Datenverbindungsschnittstelle, 167
Verbindung einrichten, 174
ADO.NET,
Verbindung einrichten, 180
age,
als MapForce-Funktion (in lang | datetime functions), 1129
Aktionen,
im Zusammenhang mit Verbindungen, 89
Als CSV speichern (getrennt),
als FlexText-Funktion, 721
Als FLF speichern (feste Länge),
als FlexText-Funktion, 727
Als Wert speichern,
als FlexText-Funktion, 729
Altova XML Parser,
Info, 2106
Altova-Erweiterungen,
Diagrammfunktionen(siehe Diagrammfunktionen), 1995
Anführungszeichen,
in CSV-Dateien, 595
Anmeldeinformationen,
erstellen, 1353
in FlowForce Server, 1362
in FlowForce Server bereitstellen, 1364
in MapForce Server, 1361
Annehmen,
Trennzeichen vorhanden, 603
Anpassen,
Befehle, 1536
Befehle löschen, 1536
Kontextmenüs, 1536
Kürzel, 1537
Menüleisten zurücksetzen, 1536
Menüs, 1536
Menüschatten, 1536
Standardmenü vs. MapForce Design, 1536
Tastatur, 1537
ANSI X12,
Nachrichtenstruktur anpassen, 566
Ansicht,
Annotationen anzeigen, 1533
Ausgewählte Komponentenkonnektoren anzeigen, 1533
Bibliothek in Funktionstitelleiste anzeigen, 1533
Bibliotheken, 1533
Bibliotheken verwalten, 1533
Datentypen anzeigen, 1533
Debug-Fenster, 1533
Meldungen, 1533
Menübefehl, 1533
Projekt-Fenster, 1533
Quell- und Zielkonnektoren anzeigen, 1533
Statusleiste, 1533
Tipps anzeigen, 1533
Übersicht, 1533

© 2016-2022 Altova GmbH

Ansicht,
Vergrößern/Verkleinern, 1533
Vorwärts, 1533
XBRL-Anzeigeoptionen, 1533
Zurück, 1533
Ant,
Java-Code erstellen mit, 1586
Java-Projekt erstellen mit, 1583
Umgebungsvariablen definieren für, 1700
Anweisung,
SQL ausführen, 392
API,
Dokumentation, 1722
Applikationsobjekt, 1726
Arbeitsblatt,
in XML-Ausgabedateien, 757
Arbeitsspeicher-Anforderungen, 2106
AS2,
Integration mit Altova-Produkten, 1450
asin,
als MapForce-Funktion (in lang | math functions), 1155
atan,
als MapForce-Funktion (in lang | math functions), 1155
ATTLIST,
DTD Namespace URIs, 125
Aufteilen,
als FlexText-Funktion, 715
Ausführen,
einzelne SQL-Anweisungen, 392
SQL, 392
SQL-Datei, 392
Ausgabe,
Alle Ausgabedateien speichern, 1530
Alle Lesezeichen löschen, 1530
Ausgabedatei neu generieren, 1530
Ausgabedatei speichern, 1530
Ausgabedatei validieren, 1530
Built-In Ausführungsprozessor, 1530
C#, 1530
C++, 1530
Einstellungen für die Textansicht, 1530
Java, 1530
Lesezeichen einfügen/löschen, 1530
Nächstes Lesezeichen, 1530
Pretty-Print, 1530
speichern, 107
SQL/NoSQL-Script ausführen, 1530
validieren, 107
Vorheriges Lesezeichen, 1530
Vorschau anzeigen, 107
XQuery, 1530
XSLT 1.0, 1530
XSLT 2.0, 1530
XSLT 3.0, 1530
Ausnahmen,
hinzufügen, 938
Ausnahmen wegen zu wenig Arbeitsspeicher,
auflösen, 1700
Auswahl des Designs für MapForce in Eclipse, 1501
Auswählen,
Tabellendaten - Datenbankabfrage, 400
Authentifizierung,
einfache HTTP, 1353
OAuth 2.0, 1353
auto-format,
als MapForce-Funktion (in edifact functions), 1125
Automatische,
Verarbeitung, 1434
Automatisches,
Laden von Bibliotheken, 1006
auto-number,
als MapForce-Funktion (in core | generator functions), 1054
avg,
als MapForce-Funktion (in core | aggregate-Funktionen), 1027
Axis2,
für SOAP Webservices konfigurieren, 1290
für SOAP-Webservices konfigurieren, 1272

B

base-uri,
als MapForce-Funktion (in xpath2 | accessors-Bibliothek), 1191
Batch,
automatische Batch-Verarbeitung, 1434
Bearbeiten,
Alle auswählen, 1520
Ausschneiden/Kopieren/Einfügen/Löschen, 1520
Rückgängig, 1520
Suchen, 1520
Vorheriges suchen, 1520
Weitersuchen, 1520
Wiederherstellen, 1520

© 2016-2022 Altova GmbH

Benutzerdefiniert,
Füllzeichen (feste Länge), 603
Benutzerdefinierte Bibliotheken,
C#, 1005
C++, 1005
hinzufügen, 1005
Java, 1005
referenzieren, 1005
Benutzerdefinierte Funktionen,
aufrufen, 968
bearbeiten, 968
Beispiel, 967
Beispiele, 978, 980
erstellen, 968
importieren, 968
inline, 968
Input-Parameter, 968
kopieren und einfügen, 968
Look-up, 980
löschen, 968
mit Anmeldeinformationen, 1358
Navigation, 968
Output-Parameter, 968
Parameter, 973
Parameter hinzufügen, 973
Parameterreihenfolge, 973
reguläre, 968
rekursiv, 978
rekursiv aufrufen, 978
rekursive Suche, 978
Strukturen von komplexen Typen (complexTypes), 973
Übersicht, 967
vom Typ complexType, 973
vom Typ simpleType, 973
Vorteile, 967
Benutzer-DSN,
einrichten, 187
Benutzeroberfläche, 33
Bereiche,
Ausgabe, 39
DB-Abfrage, 39
Mapping, 39
StyleVision-Ausgabe, 39
XQuery, 39
XSLT, 39
Betriebssysteme für Altova-Produkte, 2106
Bibliothek, 1717
automatisch laden, 1006
Binärdateien,
Daten auslesen aus, 862
Daten schreiben in, 864
Übersicht, 860
zum Mapping hinzufügen, 860
binData-content (BSON-Binärfunktion), 1120
binData-subtype (BSON-Binärfunktion), 1120
Blockkommentar, 393
boolean,
als MapForce-Funktion (in core | conversion functions), 1034
Breakpoints,
entfernen, 1418
hinzufügenadding, 1418
Info, 1414
Breakpoints (Fenster),
Info, 1414, 1425
Browser,
Datenbankabfrage, 395
Filter anwenden, 397
filtern, 397
SQL generieren, 391
Browser-Ansicht,
Kontextmenüoptionen, 398
SQL-Anweisungen generieren, 398
bson | binary functions, 1120
bson | constructor functions (Konstruktorfunktionen), 1120
bson | regex funstions (Regex-Funktionen), 1121
BSON-Funktionen, 1119
bson-maxKey (BSON-Konstruktorfunktionen), 1120
bson-minKey (BSON-Konstruktorfunktionen), 1120
bson-null (BSON-Konstruktorfunktionen), 1120
bson-undefined (BSON-Konstruktorfunktionen), 1120
build.xml,
zip64mode aktivieren in, 1700

C

C#,
Code, 1563
Codegenerierungseinstellungen, 1701
Fehlerbehandlung, 1727
generierten Code integrieren, 1597
Integration von MapForce, 1931
Programmcode generieren, 1572

© 2016-2022 Altova GmbH

C#,
Referenz zu generierten Klassen, 1668
Webservices generieren in, 1275
C++,
Code, 1563
Codegenerierungseinstellungen, 1701
Fehlerbehandlung, 1727
generierten Code integrieren, 1599
Programmcode generieren, 1567
Referenz zu generierten Klassen, 1652
capitalize,
als MapForce-Funktion (in lang | string functions), 1164
CDATA,
Abschnitt, 137
ceiling,
als MapForce-Funktion (in core | math functions), 1064
char-from-code,
als MapForce-Funktion (in core | string functions), 1104
charset-decode,
als MapForce-Funktion (in lang | string functions), 1164
charset-encode,
als MapForce-Funktion (in lang | string functions), 1166
Code,
integrierte Typen, 1716
MapForce Code integrieren, 1594
SPL, 1703
Code Generator, 1563
Code Point,
Collation, 908
code-from-char,
als MapForce-Funktion (in core | string functions), 1105
Codegenerierung, 109, 116
Beispiel, 1742
Codegenerierungseinstellungen,
global für das gesamte Projekt definieren, 122
Collation,
Locale Collation, 908
Sortierkomponente, 908
Unicode Code Point, 908
columnname-to-index,
als MapForce-Funktion, 1186
COM-API,
Dokumentation, 1722
ComplexType,
sortieren, 908
concat,
(als Funktion) Beispiel für die Verwendung, 599
als MapForce-Funktion (in core | string functions), 1105
contains,
als MapForce-Funktion (in core | string functions), 1106
convert-to-utc,
als MapForce-Funktion (in lang | datetime functions), 1129
Copyright-Informationen, 2109
cos,
als MapForce-Funktion (in lang | math functions), 1155
CouchDB-Verbindung, 198
count,
als MapForce-Funktion (in core | aggregate-Funktionen), 1028
count-substring,
als MapForce-Funktion (in lang | string functions), 1168
create-guid,
als MapForce-Funktion (in lang | generator functions), 1151
Credentials,
managing, 1357
CSV,
als Mapping-Quelle, 585
Hierarchien erstellen - Schlüssel, 591
mehrere Zeilen erstellen, 588
CSV-Dateien,
als Quellkomponente, 595
als Zielkomonente, 595
Anzeigen einer Vorschau auf die Daten aus, 595
Definieren der Kodierung, 595
Felder hinzufügen oder entfernen, 595
current,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 1251
current-date,
als MapForce-Funktion (in xpath2 | context functions), 1195
current-dateTime,
als MapForce-Funktion (in xpath2 | context functions), 1195
current-time,
als MapForce-Funktion (in xpath2 | context functions), 1196

D

Database exceptions,
handling, 435
logging, 447
Database transactions,
rolling back, 435

© 2016-2022 Altova GmbH

date-from-datetime,
als MapForce-Funktion (in lang | datetime functions), 1130
Datei,
Alle schließen, 1517
Alles speichern, 1517
als Schaltfläche in einer Komponente, 82
als Schaltfläche in Komponenten, 1371
Anmeldeinformationen-Manager öffnen, 1517
Auf FlowForce Server bereitstellen, 1517
Beenden, 1517
Code generieren, 1517
Dokumentation generieren, 1517
Drucken, 1517
Druckereinrichtung, 1517
Druckvorschau, 1517
Encoding, 603
Letzte Dateien, 1517
Mapping validieren, 1517
Mapping-Einstellungen, 1517
Neu, 1517
Neu laden, 1517
Öffnen, 1517
Schließen, 1517
Speichern, 1517
Speichern unter, 1517
Zu MapForce Server-Ausführungsdatei kompilieren, 1517
Datei mit fester Länge,
leere erstellen, 603
Datei/String,
als Schaltfläche in einer Komponente, 82
als Schaltfläche in Komponenten, 1371
Datei: (Standard),
als Name des Root-Node, 1371
Datei: <dynamisch>,
als Name des Root-Node, 1371
Datei-DSN,
einrichten, 187
Dateien,
mehrere Dateien anhand einer Datenbank, 1377
Dateinamen,
als Mapping-Input-Parameter bereitstellen, 1376
Dateipfade,
absolute, 84, 85
falsche, 85
falsche Referenzen korrigieren, 85
im generierten Code, 87
in Ausführungsumgebungen, 87
relative, 84, 85
relative im Gegensatz zu absoluten, 87
von dateibasierten Datenbanken, 85
Daten sortieren,
Sortierkomponente, 908
Datenbank,
Abfrage, 388
Ergebnisansicht, 1547
Kodierung, 1547
mehrere XML-Dateien generieren, 1379
ORDER BY, 361
sequenzielle und eindeutige Werte für Datenbanken generieren, 285
SQL Editor, 1547
SQL Editor-Einstellungen, 1547
SQL-Generierung, 1547
Textschriftarten, 1547
XML-Dateien schreiben in, 380
XML-Schema einem Feld zuweisen, 377
zum Mapping hinzufügen, 256
Datenbankabfrage,
Lesezeichen, 394
Regionen, 395
SQL ausführen, 392
SQL generieren, 391, 395, 400
SQL-Fenster, 390
Tabellen filtern, 397
Text auskommentieren, 393
Datenbankausnahmen,
behandeln, 429
potokollieren, 443
Datenbankbeziehungen,
in Mappings definieren, 276
in Mappings erhalten, 266
Datenbanken,
Aktualisieren von Tabellendaten, 300
als globale Ressourcen, 1473
Daten in Datenbanken zusammenführen, 303
Daten in eine Tabelle einfügen, 292
Daten in mehrere verknüpfte Tabellen einfügen, 295
Mappings an Datenbanken ausführen, 282
Tabellendaten aktualisieren und einfügen, 303
Tabellendaten löschen, 320
und Mapping-Kontext, 1336
Verwendung von Datenbank-Bulk-Einfügungen, 329
Datenbankobjekte,
aus dem Mapping entfernen, 260
filtern, 260
zum Mapping hinzufügen, 260

© 2016-2022 Altova GmbH

Datenbanktransaktionen,
für gespeicherte Prozeduren aktivieren, 415
im Mappings verwenden, 429
Datenbanktreiber,
Übersicht, 171
Datenbankverbindung,
Assistenten starten, 169
aus globalen Ressourcen wiederverwenden, 199
Beispiele einrichten, 200
einrichten, 167
Datenelement,
fehlendes, 103
Daten-Overlays,
Info, 1414
Datenstreaming,
Definition, 80
datetime-add,
als MapForce-Funktion (in lang | datetime functions), 1130
datetime-diff,
als MapForce-Funktion (in lang | datetime functions), 1131
datetime-from-date-and-time,
als MapForce-Funktion (in lang | datetime functions), 1132
datetime-from-parts,
als MapForce-Funktion (in lang | datetime functions), 1133
datetime-to-xlsx,
als MapForce-Funktion, 1188
date-to-xlsx,
als MapForce-Funktion, 1187
day-from-datetime,
als MapForce-Funktion (in lang | datetime functions), 1136
day-from-duration,
als MapForce-Funktion (in lang | datetime functions), 1136
DB-Abfrage,
SQL speichern / öffnen, 392
Debuggen,
abbrechen, 1532
aussteigen, 1532
einsteigen, 1532
Minimaler Schritt, 1532
starten, 1532
überspringen, 1532
Vorbereitung, 1413
Debugger-Position,
Anzeige des aktuellen Werts, 1421
Debugging,
Einschränkungen, 1409
Info, 1414
mit Breakpoints, 1409
Schritt für Schritt, 1409
default-collation,
als MapForce-Funktion (in xpath2 | context functions), 1196
degrees,
als MapForce-Funktion (in lang | math functions), 1156
Diagrammfunktionen,
Beispiel, 2083
Diagrammdatenstruktur, 2078
Liste, 2074
Digitale Signaturen,
aktivieren, 143
Canonical XML, 143
detached, 143
Einstellungen, 143
enveloped, 143
Generieren, 143
Passwort, 143
Zertifikat, 143
Digitale Zertifikate,
auf Linux übertragen, 680
auf Mac übertragen, 681
aus Windows exportieren, 674
in MapForce Mappings, 664
unter Windows verwalten, 671
vertrauenswürdige auf Mac-Systemen einrichten, 669
vertrauenswürdige unter Linux einrichten, 667
vertrauenswürdige unter Windows einrichten, 670
distinct-values,
als MapForce-Funktion (in core | sequence functions), 1077
divide,
als MapForce-Funktion (in core | math functions), 1064
divide-integer,
als MapForce-Funktion (in lang | math functions), 1156
document,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 1251
Dokumentebene,
Beispiele für die Integration von XMLSpy, 1931
DoTransform.bat,
mit RaptorXML Server ausführen, 1433
Drag and Drop,
SQL-Anweisung generieren, 391
DTD,
Quelle und Ziel, 125
Duplikat erzeugen, 54
duration-add,
als MapForce-Funktion (in lang | datetime functions), 1137

© 2016-2022 Altova GmbH

duration-from-parts,
als MapForce-Funktion (in lang | datetime functions), 1137
duration-subtract,
als MapForce-Funktion (in lang | datetime functions), 1139
Dynamische Authentifizierung,
in FlowForce Server, 1368
in MapForce Server, 1367
konfigurieren, 1366

E

Eclipse,
Java-Code kompilieren mit, 1586
Programmcode generieren, 1580
EDI,
Alle Dateipfade relativ zur MFD-Datei speichern, 466
anpassen, 539, 540
ASC X12, 460
EDIFACT, 460
EDI-Input-Datei, 466
EDI-Output-Datei, 466
Eingabe- / Ausgabekodierung, 466
Einstellungen, 466
Erweitert, 466
HIPAA X12, 460, 484
HL7, 460
IATA PADIS, 460, 494
Inhalt im XML-Format schreiben, 466
Input-Verarbeitungsoptimierungen auf Basis von min/maxOccurs aktivieren, 466
Komponenteneinstellungen, 466
Komponentenname, 466
Konfiguration, 466
mapbare Trennzeichen, 466
Nach jedem Segment neue Zeile beginnen, 466
NCPDP SCRIPT, 498
ODETTE EDI, 460
SAP IDocs, 460
schnelle Konvertierung in XML, 475, 476, 478
Standard, 460
SWIFT MT, 460
TRADACOMS, 460
Trennzeichen, 463
Trennzeichen mappen, 466
UN/EDIFACT, 460
unterstützte Standards, 460
Validierung, 466
Validierung anpassen, 573
XML-Dateien generieren von, 475, 476, 478
EDIFACT,
auf XML mappen, 531
Terminologie, 463
zusammengeführte Einträge, 551
Einfügen,
Blockkommentar, 393
Kommentare, 393
Lesezeichen, 394
Regionen, 395
SQL WHERE-Komponente, 361
Zeilenkommentar, 393
Einmal trennen,
im Modus "feste Länge", 710
im Modus "getrennt (beliebig)", 711
im Modus "getrennt (zeilenbasiert)", 712
mit dem Modus "getrennt (Zeile beginnt mit)", 713
Standardeinstellung, 709
Einstellungen, 603
Füllzeichen, 603
für die Ausgabedatei, 116
für Mappings, 116
für Webservice-Operation, 116
Textdatei mit fester Länge, 603
element-available,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 1251
empty,
als MapForce-Funktion (in lang | string functions), 1169
Encoding,
Datei, 603
Endbenutzer-Lizenzvereinbarung, 2109, 2111
Entfernen,
Blockkommentar, 393
Kommentare, 393
Lesezeichen, 394
Regionen, 395
Zeilenkommentar, 393
Enumerationen,
in MapForceControl, 1980
equal,
als MapForce-Funktion (in core | logical functions), 1057
equal-or-greater,
als MapForce-Funktion (in core | logical functions), 1058
equal-or-less,
als MapForce-Funktion (in core | logical functions), 1058

© 2016-2022 Altova GmbH

Ergebnisse,
Fenster - Datenbankabfrage, 400
Symbole in Datenbankabfrage, 400
Erstellen,
Regionen, 395
Erweitern,
Regionen, 395
Erweiterungsfunktionen für XSLT und XQuery, 2086
Erweiterungsfunktionen in .NET für XSLT und XQuery,
siehe .NET Erweiterungsfunktionen, 2096
Erweiterungsfunktionen in Java für XSLT und XQuery,
siehe Java-Erweiterungsfunktionen, 2087
Erweiterungsfunktionen in MSXSL Scripts, 2103
Evaluierungszeitraum,
für Altova-Software-Produkte, 2109
von Altova Software-Produkten, 2109
Excel 2007+,
als Mapping-Komponente, 741
als Mapping-Komponente hinzufügen, 740
als Mapping-Quelle oder - Ziel, 738
Auslesen von Mapping-Daten aus Zellenbereichen, 745
Auswählen von Zellenbereichen aus der Mapping-Komponente, 746
bestehende Arbeitsblätter aktualisieren, 764
Daten mappen auf, 1385
Schreiben von Mapping-Daten in Zellenbereiche, 745
von RSS Feed mappen, 644
XML-Dateien generieren von, 757
Excel 2007+-Arbeitsblätter,
Daten lesen aus, 744
Daten schreiben in, 744
Excel 2007+-Dateien,
auf XML mappen, 755
Mappen von Datenbanken auf, 761
Excel 2007+-Komponenteneinstellungen,
ändern, 752
exists,
als MapForce-Funktion (in core | sequence functions), 1079
exp,
als MapForce-Funktion (in lang | math functions), 1157
Extras,
Aktive Konfiguration, 1535
Anpassen, 1535
Globale Ressourcen, 1535
Menübefehl, 1535
Optionen, 1535
Symbolleisten und Fenster wiederherstellen, 1535
Umgekehrtes Mapping erstellen, 1535
XBRL-Taxonomiepakete, 1535
Extras | Optionen,
Allgemein, 1539
Bearbeiten, 1539
Code-Generierung, 1539
Datenbank, 1539
Debugger, 1539
Java, 1539
Meldungen, 1539
Netzwerk-Proxy, 1539
XBRL, 1539

F

false,
als MapForce-Funktion (in xpath2 | boolean functions), 1193
Fehlend,
leere Felder, 603
Fehlende Datenelemente, 103
Fehler,
Behebung, 80
zu wenig Speicher, 80
Fehlerbehandlung,
allgemeine Beschreibung, 1727
Fehlerhafte Verbindungen,
in Datenbanken, 103
in XML-Dateien, 103
nach Ändern des Schemas, 103
Feld,
benutzerdefiniert - CSV, 603
Schlüssel in Textdateien, 591
Felder mit fester Länge,
Feld "Benutzerdefiniert", 603
Fenster,
Bibliotheken, 34
Bibliotheken verwalten, 34
Design, 1556
Dialogfeld "Fenster", 1556
Dunkles Design, 1556
Helles Design, 1556
Horizontal/Vertikal anordnen, 1556
Klassisches Design, 1556
Mapping, 34
Mehrere Mapping-Fenster, 34
Meldungen, 38

© 2016-2022 Altova GmbH

Fenster,
Projekt, 34
Überlappend, 1556
Übersicht, 34
Feste Länge, 603
Dateien mappen, 585
Input-Datei, 603
Output-Datei, 603
Textdateieinstellungen, 603
Filter,
Datenbankobjekte, 397
den Online Browser, 397
zum Mapping hinzufügen, 914
Filtern,
Daten aus Komponenten, 914
Datenbankobjekte im Mapping filtern, 260
Datenbanktabellen, 914
in Datenbanken, 365
find-substring,
als MapForce-Funktion (in lang | string functions), 1169
Firebird,
über ODBC verbinden, 202
Verbindung über JDBC, 200
first-items,
als MapForce-Funktion (in core | sequence functions), 1081
Flat File,
mappen, 585
FlexText,
als Zielkomponente, 700
Benutzeroberfläche, 684
Container, 684
Komponenteneinstellungen ändern, 698
Tutorial, 687
Übersicht, 684
FlexText-Funktionen,
als CSV speichern (getrennt), 721
als FLF speichern (feste Länge), 727
Als Wert speichern, 729
Aufteilen, 715
Einmal trennen, 709
Ignorieren, 720
Mehrmals trennen, 702
Node, 719
floor,
als MapForce-Funktion (in core | math functions), 1065
FlowForce Server,
Anmeldeinformationen, 1362
Anmeldeinformationen bereitstellen, 1364
globale Ressourcen bereitstellen auf, 1480
Globale Ressources in, 1480
Mappings bereitstellen auf, 1445
OAuth 2.0-Autorisierung, 1362
format-date,
als MapForce-Funktion (in core | conversion functions), 1034
format-dateTime,
als MapForce-Funktion (in core | conversion functions), 1036
format-guid-string,
als MapForce-Funktion (in lang | string functions), 1170
format-number,
als MapForce-Funktion (in core | conversion functions), 1039
format-time,
als MapForce-Funktion (in core | conversion functions), 1042
Fragezeichen,
fehlende Datenelemente, 103
Füllzeichen, 603
Datei mit feste Länge, 603
entfernen, 603
function-available,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 1252
Funktionen, 942
Argumentdatentyp, 943
Benutzerdefinierte Funktion erstellen, 1529
Benutzerdefinierte Funktion von Auswahl erstellen, 1529
Beschreibung, 943
durchsuchen, 943
Funktion entfernen, 1529
Funktionseinstellungen, 1529
Grundlagen, 943
hinzufügen, 943
im Fenster "Bibliotheken" suchen, 943
Input-Komponente einfügen, 1529
Instanzen im aktiven Mapping suchen, 943
Konstanten, 943
Node, 951, 953
Output-Komponente einfügen, 1529
Parameter, 943
Parameter hinzufügen, 943
Parameter löschen, 943

© 2016-2022 Altova GmbH

G

generate-id,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 1252
generate-sequence,
als MapForce-Funktion (in core | sequence functions), 1082
Generieren,
Code von Schema, 1563
SQL, 391
Generierter Code,
Ausnahmen auslösen von, 938
Gespeicherte Prozeduren,
Anmerkungen zur Unterstützung, 402
über ein Mapping aufrufen, 420, 424
von einem Mapping aus aufrufen, 402, 408, 411
vopn einem Mapping aus aufrufen, 415
zu einem Mapping hinzufügen, 405
get-fileext,
als MapForce-Funktion (in core | file path functions), 1050
get-folder,
als MapForce-Funktion (in core | file path functions), 1050
Global, 2107
Globale Objekte,
in SPL, 1707
Globale Ressourcen,
auf FlowForce Server, 1480
auf FlowForce Server bereitstellen, 1480
Datenbanken als, 1473
Definitionsdatei, 1463
Einführung, 1462
einrichten, 1463
erstellen, 1463
in Ausführungsumgebungen, 1479
in FlowForce Server, 1480
Ordner, 1471
XML-Dateien als, 1469
Grafische Benutzeroberfläche, 33
greater,
als MapForce-Funktion (in core | logical functions), 1059
group-adjacent,
als MapForce-Funktion (in core | sequence functions), 1083
group-by,
als MapForce-Funktion (in core | sequence functions), 1085
group-ending-with,
als MapForce-Funktion (in core | sequence functions), 1087
group-into-blocks,
als MapForce-Funktion (in core | sequence functions), 1089
group-starting-with,
als MapForce-Funktion (in core | sequence functions), 1091

H

Health Level 7,
Beispiel, 492
Hierarchie,
von Textdateien, 591
Hilfe,
Auf Updates überprüfen, 1558
Bestellformular, 1558
Fragen und Antworten im Web, 1558
Index, 1558
Inhaltsverzeichnis, 1558
Komponenten und Gratistools downloaden, 1558
MapForce im Internet, 1558
MapForce Training, 1558
Registrierung, 1558
Software-Aktivierung, 1558
Suchen, 1558
Support Center, 1558
Über MapForce, 1558
Hintergrundinformationen, 2106
HIPAA,
X12 999 Implementation Acknowledgement generieren, 483
HIPAA X12, 484
HL7 2.6 auf 3.x,
Beispiel, 492
hour-from-datetime,
als MapForce-Funktion (in lang | datetime functions), 1140
hour-from-duration,
als MapForce-Funktion (in lang | datetime functions), 1140
HRESULT,
und Fehlerbehandlung, 1727
HTML,
Ausgabevorschau als, 1395
Mapping-Dokumentation generieren, 1399
HTTPS,
Webservices aufrufen über, 664

© 2016-2022 Altova GmbH

I

IATA PADIS, 494
IBM DB2,
Lesen aus Feldern vom Typ XML, 377
Schreiben in Felder vom Typ XML, 377
über JDBC verbinden, 204
über ODBC verbinden, 206
IBM DB2 für i,
über ODBC verbinden, 213
Verbinden über JDBC, 212
IBM Informix,
über JDBC verbinden, 216
If-Else-Bedingungen,
zum Mapping hinzufügen, 914
Ignorieren,
als FlexText-Funktion, 720
IIS,
für SOAP-Webservices konfigurieren, 1275
implicit-timezone,
als MapForce-Funktion (in xpath2 | context functions), 1196
index-to-columnname,
als MapForce-Funktion, 1188
INNERER JOIN,
in Join-Komponenten, 345
Input,
Datei - Text, 603
duplizieren, 82
Input mappen,
mehrere Dateien bereitstellen als Input, 1374
Integrating,
MapForce in Applikationen, 1919
Integration,
mit Altova-Produkten, 31
Integrieren,
in C#, 1597
in C++, 1599
in Java, 1595
Integrieren von MapForce Code, 1594
Intelligente Komponentenlöschung, 105
Internet-Verwendung,
in Altova-Produkten, 2107
is-not-null,
als MapForce-Funktion (in db functions), 1122
is-null,
als MapForce-Funktion (in db functions), 1123
ISO 20022,
Anmerkungen zur Unterstützung, 512
Daten bidirektional mappen, 512
ISO/IEC 10646, 2107
is-xsi-nil,
als MapForce-Funktion (in core | node functions), 1069
item-at,
als MapForce-Funktion (in core | sequence functions), 1093
items-from-till,
als MapForce-Funktion (in core | sequence functions), 1094

J

Java, 1938
Ausnahmen im generierten Code vermeiden, 1700
Code, 1563
generierten Code integrieren, 1595
Mapping-Code generieren für, 1580
Programmcode generieren, 1586
Referenz zu generierten Klassen, 1684
VM-Bibliothekspfad, 1542
Webservices generieren in, 1290
Java Erweiterungsfunktionen,
Datentypkonvertierungen, Java in Xpath/XQuery, 2096
Instanzmethoden, Instanzfelder, 2094
Konstruktoren, 2093
statische Methoden, statische Felder, 2093
Java-Erweiterungsfunktionen,
benutzerdefinierte JAR-Dateien, 2092
benutzerdefinierte Klassendateien, 2089
Datentyp-Konvertierungen, XPath/XQuery in Java, 2095
für XSLT und XQuery, 2087
Übersicht, 2087
JavaScript,
Fehlerbehandlung, 1727
JDBC,
als Datenverbindungsschnittstelle, 167
mit Teradata verbinden, 249
Referenzen im generierten Code behandeln, 1584
Verbindung einrichten (Windows), 190
JScript,
Beispiel für Codegenerierung, 1742
JSON,
Beispielmapping, 832

© 2016-2022 Altova GmbH

JSON,
Daten von oder auf JSON mappen, 823
Mappen von Webservices, 640
Mappen von zusätzlichen Eigenschaften, 832
JSON5,
Mappen von Daten von auf JSON5, 827
JSON-Dateien,
Daten von und auf JSON mappen, 821

K

Kataloge, 1483
anpassen, 1488
in DTD, 1484
in XML-Schema, 1484
Struktur, 1486
Umgebungsvariablen, 1490
Kodierung,
in CSV-Dateien ändern, 595
Kommentare,
zu Zieldateien hinzufügen, 136
Komponente,
Daten sortieren, 908
Komponenten,
Aktualisieren, 1526
Ausnahme, 1521
ausrichten, 82
Datenbank, 1521
Datenbank abfragen, 1526
Datenbankaktionen, 1526
Datenbanken, 165
Datenbankobjekte hinzufügen/entfernen/bearbeiten, 1526
Duplikat danach einfügen, 1526
Duplikat davor einfügen, 1526
Duplikat löschen, 1526
durchsuchen, 82
EDI, 460, 1521
Eigenschaften, 1526
Einfache Input-Komponente, 1521
Einfache Output-Komponente, 1521
Einstellungen, 82
Einstellungen ändern, 82
Excel 2007+-Datei, 1521
Filter: Nodes/Zeilen, 1521
FlexText-Konfiguration bearbeiten, 1526
gelöschte Datenelemente, 103
Grundlagen, 82
IF-Else-Bedingung, 1521
Inhalt als CDATA-Abschnitt schreiben, 1526
Input-Komponente einfügen, 1521
Join, 1521
JSON-Schema/Datei, 1521
Konstante, 1521
Links ausrichten, 1526
löschen, 105
Mapping auf EDI X12 997 erstellen, 1526
Mapping auf EDI X12 999 erstellen, 1526
Menübefehle, 1526
Output-Komponente einfügen, 1521
Protocol Buffers-Datei, 1521
Rechts ausrichten, 1526
Root-Element ändern, 1526
Schema-Definition in XMLSpy bearbeiten, 1526
Sortieren: Nodes/Zeilen, 1521
SQL/NoSQL-WHERE/ORDER, 1521
Strukturkomponenten, 78, 124, 125, 165, 460, 768, 769, 770, 772, 775, 792, 797, 801, 810, 816
Textdatei, 1521
Transformation, 78
Übersicht, 78
Variable, 1521
Webservice-Funktion, 1521
Wertezuordnung, 1521
XBRL-Dokument, 1521
XML, 126
XML und XML-Schema, 126, 130, 132, 136, 137, 138, 141, 143, 144, 147
XML-Schema, 126
XML-Schema/Datei, 1521
zum Mapping hinzufügen, 80
Konfigurieren,
mff-Datei, 1006
Konnektor,
Anzeige des Verlaufs von verarbeiteten Werten, 1421
Konstanten,
hinzufügen, 943
Kontext (Fenster),
Info, 1414, 1423
Konventionen, 26

© 2016-2022 Altova GmbH

L

last,
als MapForce-Funktion (in xpath2 | context functions), 1197
last-items,
als MapForce-Funktion (in core | sequence functions), 1095
Layout,
Browser, 395
leapyear,
als MapForce-Funktion (in lang | datetime functions), 1141
Leere,
neue Textdatei erstellen, 603
Leere Felder,
als nicht vorhanden behandeln, 603
in CSV-Dateien, 595
left,
als MapForce-Funktion (in lang | string functions), 1171
left-trim,
als MapForce-Funktion (in lang | string functions), 1171
Leisten,
Applikationsstatus, 34
Menü, 34
Symbolleisten, 34
Lesezeichen,
einfügen, 394
entfernen, 394
Lesezeichenleiste, 394
navigieren, 394
less,
als MapForce-Funktion (in core | logical functions), 1059
LINKER ÄUSSERER JOIN,
in Join-Komponenten, 345
Linux,
Client-Zertifikate übertragen auf, 680
Mappings mit Webservice-Aufrufen über HTTPS ausführen, 680
vertrauenswürdige Server-Zertifikate einrichten, 667
Lizenz, 2111
Informationen, 2109
Lizenzierung, 1558
Lizenzüberwachung,
in Altova-Produkten, 2110
Locale Collation, 908
local-name-from-QName,
als MapForce-Funktion (in lang | QName functions), 1074
log,
als MapForce-Funktion (in lang | math functions), 1157
log10,
als MapForce-Funktion (in lang | math functions), 1158
Logging,
database exceptions, 447
logical-and,
als MapForce-Funktion (in core | logical functions), 1060
logical-not,
als MapForce-Funktion (in core | logical functions), 1060
logical-or,
als MapForce-Funktion (in core | logical functions), 1061
logical-xor,
als MapForce-Funktion (in lang | logical functions), 1152
Lokale Beziehungen,
und gespeicherte Prozeduren, 418, 420, 424
Lookup-Tabellen,
im Mapping verwenden, 921
Löschen,
fehlende Datenelemente, 103
lowercase,
als MapForce-Funktion (in lang | string functions), 1172

M

Mac,
Client-Zertifikate übertragen auf, 681
Mappings mit Webservice-Aufrufen über HTTPS ausführen, 681
vertrauenswürdige Server-Zertifikate einrichten, 669
main-mfd-filepath,
als MapForce-Funktion (in core | file path functions), 1051
MapForce,
API, 1722
Integration, 1919
Übersicht, 26
MapForce API, 1722
MapForce plug-in for Eclipse,
über, 1495
MapForce Plug-in für Eclipse,
Arbeiten mit Mappings und Projekten, 1504
Erstellen eines MapForce/Eclipse-Projekts, 1504
Erweiterungspunkt, 1514
Funktionalitäten erweitern, 1514
für die automatische Codegenerierung konfigurieren, 1511

© 2016-2022 Altova GmbH

MapForce Plug-in für Eclipse,
häufig verwendete Menüs und Funktionen aufrufen, 1501
Info, 1498, 1504
installieren, 1496
Mappings in ein Eclipse-Projekt importieren, 1508
neue Mappings erstellen, 1506
Wechseln in die MapForce-Perspektive, 1498
MapForce Plug-in für Visual Studio,
aktivieren, 1492
Arbeiten mit Mappings und Projekten, 1492
Info, 1492
Informationen zu Menüs und Funktionen, 1492
MapForce Server,
Anmeldeinformationen, 1361
Ausnahmen auslösen von, 938
Automatisieren von Mappings, 1434
globale Ressources in, 1480
Mappings dafür kompilieren, 1442
OAuth 2.0-Autorisierung, 1361
MapForceCommand,
in MapForceControl, 1959
MapForceCommands,
in MapForceControl, 1961
MapForceControl, 1962
Beispiele für die Integration auf Dokumentebene, 1931
Dokumentation, 1919
Integration mittels C#, 1931
Integration mittels Visual Basic, 1947
Objektreferenz, 1959
MapForceControlDocument, 1970
MapForceControlPlaceHolder, 1977
Mapping,
Begriffe, 76
Bestandteile, 76
debuggen, 1409
Einstellungen, 116
erstellen, 76
Flat File-Format, 585
Grundlagen, 76
Komponenten, 76
Konnektoren, 76
quellorientiert - Mixed Content, 93
Terminologie, 76
validieren, 107
Verbindungen, 76
Mapping Input,
benutzerdefinierte Dateinamen bereitstellen als, 1376
mehrere Dateien bereitstellen als, 1375
Mapping Output,
mehrere Dateien generieren als, 1375
Mapping-Dokumentation,
generieren, 1399
Mapping-Input,
Mehrere Dateien bereitstellen als, 1371
Mapping-Kontext, 1333
Mapping-Output,
Mehrere Dateien generieren als, 1371
Mappings,
automatische Verarbeitung, 1434
Mapping-Szenarien, 29
MariaDB,
über ODBC verbinden, 218
match-pattern,
als MapForce-Funktion (in lang | string functions), 1173
max,
als MapForce-Funktion (in core | aggregate-Funktionen), 1029
als MapForce-Funktion (in lang | math functions), 1158
max-string,
als MapForce-Funktion (in core | aggregate-Funktionen), 1030
Mehrere aufeinander folgende Elemente,
Edifact - X12, 551
Mehrere XML-Dateien,
anhand einer einzigen XML-Quelle, 1377
Mehrmals trennen,
im Modus "feste Länge", 703
im Modus "getrennt (beliebig)", 704
im Modus "getrennt (Zeile beginnt mit)", 707
im Modus "getrennt (zeilenbasiert)", 706
Standard, 702
Meldungen,
Fenster - Datenbankabfrage, 400
Symbole in Datenbankabfrage, 400
Menübefehle, 1516
Anpassen, 1536, 1537
Ansicht, 1533
Ausgabe, 1530
Bearbeiten, 1520
Datei, 1517
Debuggen, 1532
Einfügen, 1521
Extras, 1535
Extras | Anpassen, 1536
Extras | Optionen, 1539
Extras | Optionen | Datenbank, 1547

© 2016-2022 Altova GmbH

Menübefehle, 1516
Extras | Optionen | Java, 1542
Extras | Optionen | Netzwerk-Proxy, 1552
Extras | Optionen | XBRL, 1543
Extras | Tastatur, 1537
Fenster, 1556
Funktion, 1529
Hilfe, 1558
Komponente, 1526
Projekt, 1524
Verbindung, 1528
Menüreferenz, 1516
MERGE,
als Anweisung in MapForce-generiertem SQL-Code, 303
in MapForce-generiertem SQL-Code, 310
Methodennamen im generierten Code,
reservieren, 1700
mfd-filepath,
als MapForce-Funktion (in core | file path functions), 1051
mff,
Bibliotheksdatei, 1005
mff.xsd-Datei, 1005
mff-Datei,
konfigurieren, 1006
Microsoft Access,
über ADO verbinden, 174, 220
Microsoft Excel,
siehe "Excel 2007+", 738
Microsoft SharePoint Server,
Dateien als Komponenten hinzufügen über, 80
Microsoft SQL Server,
über ADO verbinden, 222
über ODBC verbinden, 225
millisecond-from-datetime,
als MapForce-Funktion (in lang | datetime functions), 1141
millisecond-from-duration,
als MapForce-Funktion (in lang | datetime functions), 1142
min,
als MapForce-Funktion (in core | aggregate-Funktionen), 1030
als MapForce-Funktion (in lang | math functions), 1159
min-string,
als MapForce-Funktion (in core | aggregate-Funktionen), 1031
minute-from-datetime,
als MapForce-Funktion (in lang | datetime functions), 1142
minute-from-duration,
als MapForce-Funktion (in lang | datetime functions), 1143
Mixed,
Content mappen, 93
quellorientiertes Mapping, 93
Mixed Content,
mappen, 93
mit Standardverbindungen, 93
mit zielorientierten Verbindungen, 93
modulus,
als MapForce-Funktion (in core | math functions), 1065
MongoDB-Verbindung, 197
month-from-datetime,
als MapForce-Funktion (in lang | datetime functions), 1143
month-from-duration,
als MapForce-Funktion (in lang | datetime functions), 1144
MSXML,
Code generieren für, 1701
msxsl:Script, 2103
multiply,
als MapForce-Funktion (in core | math functions), 1066
MySQL,
über ODBC verbinden, 231

N

Namespace URI,
DTD, 125
Namespaces,
benutzerdefinierte, 141
manuell deklarieren, 141
namespace-uri-form-QName,
als MapForce-Funktion (in lang | QName functions), 1076
Navigieren,
Lesezeichenbookmarks, 394
NCPDP SCRIPT,
als Mapping-Komponente, 498
negative,
als MapForce-Funktion (in lang | logical functions), 1152
Netzwerk-Proxy,
automatisch, 1552
Einstellungen, 1552
Konfiguration, 1552
manuell, 1552
System, 1552
Neue Funktionen, 20
Version 2019, 24
Version 2020, 22

© 2016-2022 Altova GmbH

Neue Funktionen, 20
Version 2021, 21
Version 2022, 20
Version 2023, 20
Node,
als FlexText-Funktion, 719
Node-Funktionen,
erstellen, 953
node-name,
als MapForce-Funktion (in core | node functions), 1071
als MapForce-Funktion (in xpath2 | accessors-Bibliothek), 1191
node-name (Funktion),
Alternativen, 1309
Node-Namen,
Daten mappen von/auf, 1309
normalize-space,
als MapForce-Funktion (in core | string functions), 1107
not-equal,
als MapForce-Funktion (in core | logical functions), 1062
not-exists,
als MapForce-Funktion (in core | sequence functions), 1096
now,
als MapForce-Funktion (in lang | datetime functions), 1144
NULL,
Attribut, 132
in Datenbankmappings behandeln, 333
Werte, 132
Werte in Datenbanken, 132
Nullwerte,
in Datenbankkomponenten behandeln, 285
number,
als MapForce-Funktion (in core | conversion functions), 1043
numeric,
als MapForce-Funktion (in lang | logical functions), 1153

O

OAuth 2.0,
Anmeldeinformationen, 1353
credentials, 1358
in FlowForce Server, 1362
in MapForce Server, 1361
obtaining the access token, 1358
Objektmodell,
Übersicht, 1726
Objektnamen relativ zum Standardschema verwenden,
als Option, 263
ODBC,
als Datenverbindungsschnittstelle, 167
mit MariaDB verbinden, 218
mit Teradate verbinden, 250
Verbindung einrichten, 187
ODBC-Treiber,
Verfügbarkeit überprüfen, 187
Öffnen,
SQ- Skript, 392
OLE DB,
als Datenverbindungsschnittstelle, 167
OOXML,
als Excel 2007-Standardformat, 738
OpenJDK,
als Java Virtual Machine, 190
Oracle,
Lesen aus Feldern vom Typ XML, 377
Schreiben in Felder vom Typ XML, 377
Oracle-Datenbank,
über JDBC verbinden, 233
über ODBC verbinden, 235
Oracle-Paket,
gespeicherte Prozeduren und Funktionen, 405
ORDER BY,
SQL where-Komponente, 361
Ordner,
als globale Ressourcen, 1471
Layout - Datenbankabfrage, 395
Output,
Datei - Text, 603

P

PADIS,
IATA, 494
pad-string-left,
als MapForce-Funktion (in lang | string functions), 1173
pad-string-right,
als MapForce-Funktion (in lang | string functions), 1174
Parameter,
für das Mapping bereitstellen, 867, 872
Parent-Kontext,
Beispiel, 1340

© 2016-2022 Altova GmbH

parse-date,
als MapForce-Funktion (in core | conversion functions), 1044
parse-dateTime,
als MapForce-Funktion (in core | conversion functions), 1045
parse-number,
als MapForce-Funktion (in core | conversion functions), 1047
Parser,
in Altova-Produkte integrierter, 2106
parse-time,
als MapForce-Funktion (in core | conversion functions), 1049
PDF,
Ausgabevorschau als, 1395
Mapping-Dokumentation generieren, 1399
Pfade,
absolut machen, 116
im generierten Code, 116
pi,
als MapForce-Funktion (in lang | math functions), 1159
Plattformen für Altova-Produkte, 2106
position,
als MapForce-Funktion (in core | sequence functions), 1096
positive,
als MapForce-Funktion (in lang | logical functions), 1153
PostgreSQL,
nativ verbinden, 195
über ODBC verbinden, 240
pow,
als MapForce-Funktion (in lang | math functions), 1160
Primärschlüssel,
für eine Datenbankkomponente generieren, 285
in Datenbankmappings, 266, 276
während der Datenbankeinfügeaktion generieren, 292
Prioritätskontext, 1344
Beispiel, 1346
Processing Instructions,
zu Zieldateien hinzufügen, 136
Processing Instructions und Kommentare,
mappen, 93
Progress OpenEdge-Datenbank,
über ODBC verbinden, 244
verbinden über JDBC, 242
Projekt,
Bildvorschau, 119
Codegenerierung, 119
Codegenerierungseinstellungen, 122
durchsuchen, 119
Eigenschaften, 122
Einstellungen, 122
entfernen, 119
erstellen, 119
Grundlagen, 119
löschen, 119
Mappings hinzufügen, 119
neu, 119
öffnen, 119
Ordner, 122
schließen, 119
strukturieren, 119
Projektdateien (.mfp), 119
Projekte, 119
Aktive Datei zu Projekt hinzufügen, 1524
Code für das gesamte Projekt generieren, 1524
Code generieren in, 1524
Datei in XMLSpy öffnen, 1524
Dateien zu Projekt hinzufügen, 1524
Eigenschaften, 1524
Mapping für Operation erstellen, 1524
Mapping öffnen, 1524
Mapping-Datei für Operation hinzufügen, 1524
Neu laden, 1524
Ordner erstellen, 1524
Schließen, 1524
Speichern, 1524
WebService einfügen, 1524
Protocol Buffer-Dateien,
als Mapping-Komponenten, 843
Protocol Buffers,
als Mapping-Komponenten, 843
Daten auslesen aus, 849
Daten schreiben in, 854
Komponenteneinstellungen, 847
zum Mapping hinzufügen, 844
Protokollieren,
Datenbankausnahmen, 443

Q

QName,
als MapForce-Funktion (in lang | QName functions), 1074
QName-as-string,
als MapForce-Funktion (in lang | QName functions), 1163

© 2016-2022 Altova GmbH

Quelldatei,
in mehrere Zieldateien aufteilen, 1377
Quelle, 27
Quellorientiertes Mapping,
Mixed Content Mapping, 93

R

radians,
als MapForce-Funktion (in lang | math functions), 1160
random,
als MapForce-Funktion (in lang | math functions), 1160
RaptorXML Server,
Transformation ausführen, 1433
read-binary-file,
als MapForce-Funktion (in lang | file functions), 1149
Rechtliches, 2109
Reduzieren,
Regionen, 395
Referenz der Komponentensymbole, 78
regex-options (BSON-Regex-Funktionen), 1121
regex-pattern (BSON-Regex-Funktionen), 1121
Regionen,
einfügen, 395
entfernen, 395
erstellen, 395
erweitern, 395
reduzieren, 395
Regular Expressions,
FlexText-Komponente trennen mit, 731
in FlexText-Aufteilen-Bedingungen verwenden, 733
in MapForce FlexText, 730
in Mappings verwenden, 1020
remove-fileext,
als MapForce-Funktion (in core | file path functions), 1052
remove-folder,
als MapForce-Funktion (in core | file path functions), 1052
remove-timezone,
als MapForce-Funktion (in lang | datetime functions), 1145
repeat-string,
als MapForce-Funktion (in lang | string functions), 1175
replace,
als MapForce-Funktion (in lang | string functions), 1175
replace-fileext,
als MapForce-Funktion (in core | file path functions), 1053
replicate-item,
als MapForce-Funktion (in core | sequence functions), 1099
replicate-sequence,
als MapForce-Funktion (in core | sequence functions), 1101
resolve-filepath,
als MapForce-Funktion (in core | file path functions), 1053
resolve-uri,
als MapForce-Funktion (in xpath2 | anyURI functions), 1192
reversefind-substring,
als MapForce-Funktion (in lang | string functions), 1176
right,
als MapForce-Funktion (in lang | string functions), 1177
right-trim,
als MapForce-Funktion (in lang | string functions), 1177
round,
als MapForce-Funktion (in core | math functions), 1066
round-half-to-even,
als MapForce-Funktion (in xpath2 | numeric functions), 1221
round-precision,
als MapForce-Funktion (in core | math functions), 1067
RSS Feed,
mappen von, 644
RTF,
Ausgabevorschau als, 1395
Mapping-Dokumentation generieren, 1399

S

SAP IDoc,
als Mapping-Komponenten hinzufügen, 501
auf XML mappen, 501
Daten von/auf mappen, 501
Schema,
Branchenstandard, 125
Code Generator, 1563
generieren, 125
vorverpackt, 125
Schema-Manager,
CLI-Befehl Help, 158
CLI-Befehl Info, 158
CLI-Befehl Initialize, 159
CLI-Befehl Install, 159
CLI-Befehl List, 160
CLI-Befehl Reset, 161
CLI-Befehl Uninstall, 162

© 2016-2022 Altova GmbH

Schema-Manager,
CLI-Befehl Update, 163
CLI-Befehl Upgrade, 163
CLI-Übersicht, 157
Patch für Schema installieren, 155
Schema deinstallieren, 156
Schema installieren, 155
Schemas nach Status auflisten in, 154
starten, 152
Status von Schemas, 154
Übersicht, 149
Upgrade für Schema installieren, 155
zurücksetzen, 156
Schemanamen aus Tabellennamen entfernen,
als Codegenerierungsoption, 263
schemanativetype, 1704
Schlüssel,
Felder in Textdateien, 591
Sortierschlüssel, 908
Scripts in XSLT/XQuery,
siehe Erweiterungsfunktionen, 2086
second-from-datetime,
als MapForce-Funktion (in lang | datetime functions), 1145
second-from-duration,
als MapForce-Funktion (in lang | datetime functions), 1146
Sekundärschlüssel,
in Datenbankmappings, 266, 276
Sensible Daten,
speichern, 1353
Sequenz, 1331
set-empty,
als MapForce-Funktion (in core | sequence functions), 1102
set-null,
als MapForce-Funktion (in db functions), 1123
set-xsi-nil,
als MapForce-Funktion (in core | node functions), 1072
SimpleType,
sortieren, 908
sin,
als MapForce-Funktion (in lang | math functions), 1161
skip-first-items,
als MapForce-Funktion (in core | sequence functions), 1102
SOAP,
Anmerkungen zur Unterstützung, 1258
Fehler definieren, 1270
Java-Spezifisches, 1272
SOAP-Meldung,
Einstellen des TTL (time-to-live), 638
UsernameToken hinzufügen zu, 638
Software-Produktlizenz, 2111
Sortieren,
Daten im Ergebnisfenster, 400
Sortierkomponente, 908
Spaltensymbol im Ergebnisfenster, 400
Tabellen in Datenbankabfrage, 395
Sortierreihenfolge,
ändern, 908
Sortierschlüssel,
Sortierkomponente, 908
Sortierung,
in Datenbanken, 365
Speichern,
SQL-Skripts, 392
SPL, 1703
Codeblöcke, 1703
Dateien verwenden, 1709
foreach, 1712
globale Objekte, 1707
Variablen, 1706
SQL, 392
Anweisungen ausführen, 392
Anweisungen generieren, 391
Daten mittels Join verknüpfen, 345
Fenster in Datenbankabfrage, 390
Skript öffnen, 392
SQL-Skripts speichern, 392
SQL Editor,
Kommentare einfügen, 393
Kommentare entfernen, 393
Lesezeichen einfügen, 394
Lesezeichen entfernen, 394
Lesezeichen verwenden, 394
Lesezeichenleiste, 394
Regionen einfügen, 395
Regionen entfernen, 395
Regionen erstellen, 395
Regionen verwenden, 395
SQL ausführen, 392
Text auskommentieren, 393
SQL Server,
aus XML-Typ-Feldern lesen, 377
in XML-Typ-Felder schreiben, 377
Lesen aus Feldern vom Typ XML, 377
Schreiben in Felder vom Typ XML, 377
über ADO verbinden, 174
über ADO.NET verbinden, 180

© 2016-2022 Altova GmbH

SQL Server,
über JDBC verbinden, 190
SQL WHERE,
Komponente - einfügen, 361
ORDER BY, 361
SQL WHERE/ORDER,
als MapForce-Komponente, 365
SQLite,
Daten mappen auf, 339, 1390
Daten mappen von, 1385
Datenbankpfad im generierten Code in einen absoluten ändern, 87
XML-Dateien schreiben in, 380
sqrt,
als MapForce-Funktion (in lang | math functions), 1161
Standardwerte, 951
erstellen, 953
Standardwerte und Node-Funktionen,
Annotation, 963
auf Basis von Bedingungen anwenden, 957
erstellen, 953
filtern, 957
fractionDigits, 963
Funktionssymbole, 951
geerbte Regeln außer Kraft setzen, 957
Input-Seite, 951
konfigurieren, 953
length, 963
maxLength, 963
Metadaten, 963
minLength, 963
Output-Seite, 951
precision, 963
Regelkonfiguration, 953
Regeln blockieren, 957
scale, 963
Standardwerte für nicht verbundene Nodes, 957
Szenarien, 957
totalDigits, 963
visuelle Anhaltspunkte, 953
starts-with,
als MapForce-Funktion (in core | string functions), 1108
static-node-annotation,
als MapForce-Funktion (in core | node functions), 1072
static-node-name,
als MapForce-Funktion (in core | node functions), 1073
String,
als MapForce-Funktion (in core | conversion functions), 1049
als MapForce-Funktion (in xpath2 | accessors-Bibliothek), 1191
Daten parsen aus, 1382
Daten parsen von, 1382, 1385
Daten serialisieren in, 1382, 1390
string-as-QName,
als MapForce-Funktion (in lang | QName functions), 1163
string-compare,
als MapForce-Funktion (in lang | string functions), 1178
string-compare-ignore-case,
als MapForce-Funktion (in lang | string functions), 1179
string-join,
als MapForce-Funktion (in core | aggregate-Funktionen), 1032
string-length,
als MapForce-Funktion (in core | string functions), 1109
Strukturkomponenten,
Datenbanken, 165
EDI, 460
Webservices, 610
XBRL, 768, 769, 770, 772, 775, 792, 797, 801, 810, 816
XML, 125
XML und XML-Schema, 125
XML-Schema, 125
Stylevision,
Mapping-Dokumentation erstellen mit, 1401
Stylesheets erstellen mit, 1406
substitute-missing,
als MapForce-Funktion (in core | sequence functions), 1103
substitute-missing-with-xsi-nil,
als MapForce-Funktion (in core | node functions), 1073
substitute-null,
als MapForce-Funktion (in db functions), 1124
substring,
als MapForce-Funktion (in core | string functions), 1109
substring-after,
als MapForce-Funktion (in core | string functions), 1110
substring-before,
als MapForce-Funktion (in core | string functions), 1111
subtract,
als MapForce-Funktion (in core | math functions), 1067
Suchen,
Dateien im Projektfenster, 119
Datenelemente in Mapping-Komponenten, 82
Projekte, 119
sum,

© 2016-2022 Altova GmbH

sum,
als MapForce-Funktion (in core | aggregate-Funktionen), 1033
SWIFT MT,
Anmerkungen zur Unterstützung, 504
Beispielmapping, 507
Daten bidirektional mappen, 504
SWIFT MX,
Anmerkungen zur Unterstützung, 512
Beispielmapping, 507
Daten bidirektional mappen, 512
Sybase,
über JDBC verbinden, 247
Symbole,
in Ergebnisfenster in Datenbankabfrage, 400
in Meldungsfenster von Datenbankabfrage, 400
System-DSN,
einrichten, 187
system-property,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 1253

T

Tabellendaten,
sortieren, 908
tan,
als MapForce-Funktion (in lang | math functions), 1162
Taxonomie-Manager,
CLI-Befehl Help, 786
CLI-Befehl Info, 786
CLI-Befehl Initialize, 787
CLI-Befehl Install, 787
CLI-Befehl List, 788
CLI-Befehl Reset, 789
CLI-Befehl Uninstall, 790
CLI-Befehl Update, 791
CLI-Befehl Upgrade, 791
CLI-Übersicht, 785
Installieren einer Taxonomie, 783
Patch auf Taxonomie anwenden, 783
starten, 780
Status von Taxonomien, 782
Taxonomie deinstallieren, 784
Taxonomien nach Status auflisten, 782
Übersicht, 777
Upgrade für eine Taxonomie installieren, 783
zurücksetzen, 784
Technische Informationen, 2106
Teradata,
über JDBC verbinden, 249
über ODBC verbinden, 250
Terminologie,
EDIFACT, 463
Text,
Dateien - Schlüsselfelder definieren, 591
neue Textdatei erstellen, 603
Textdateien mappen, 585
Textansicht,
durchsuchen, 113
Einrücklinien, 110
Klappleiste, 110
Klappleisten, 110
Lesezeichen, 110
Pretty-Print-Anzeige, 110
Syntaxfärbung, 110
Textmarkierung, 110
Vergrößern/Verkleinern, 110
Whitespace-Markierungen, 110
Zeilenendemarkierungen, 110
Zeilennummerierung, 110
Zeilenumbruch, 110
Textbegrenzung, 603
Textdateien,
Daten mappen von, 599
Füllzeichen definieren, 599
Größe der Felder mit fester Länge definieren, 599
time-from-datetime,
als MapForce-Funktion (in lang | datetime functions), 1146
time-to-xlsx,
als MapForce-Funktion, 1189
timezone,
als MapForce-Funktion (in lang | datetime functions), 1147
to-binData (BSON-Binärfunktion), 1120
to-date,
als MapForce-Funktion (in edifact functions), 1126
to-datetime,
als MapForce-Funktion (in edifact functions), 1126
to-duration,
als MapForce-Funktion (in edifact functions), 1128
tokenize,
als MapForce-Funktion (in core | string functions), 1111
tokenize-by-length,
als MapForce-Funktion (in core | string functions), 1114

© 2016-2022 Altova GmbH

tokenize-regexp,
als MapForce-Funktion (in core | string functions), 1116
Tomcat,
für SOAP Webservices konfigurieren, 1290
für SOAP-Webservices konfigurieren, 1272
to-regex (BSON-Regex-Funktionen), 1121
to-time,
als MapForce-Funktion (in edifact functions), 1128
TRADACOMS,
als Mapping-Komponente, 518
as mapping component, 512
in XML konvertieren, 523
Konfigurationsdateien, 522
Validierungsregeln, 519
Transformationen,
RaptorXML Server, 1433
Transformationssprachen,
BUILT-IN, 28
C#, 28
C++, 28
Java, 28
XQuery, 28
XSLT 1.0, 28
XSLT 2.0, 28
XSLT 3.0, 28
translate (in core | string functions),
als MapForce-Funktion, 1118
Trennzeichen,
als vorhanden annehmen, 603
EDI, 463
in CSV-Dateien ändern, 595
true,
als MapForce-Funktion (in xpath2 | boolean functions), 1193
Typen,
integrierte, 1716

U

UCS-2, 2107
UCS-4, 2107
UDFs,
and mapping context, 1337
Umgebungsvariablen,
ANT_OPS, 1700
UN/EDIFACT,
auf XML mappen, 531
Nachrichtenstruktur anpassen, 559
Terminologie, 463
unary-minus,
als MapForce-Funktion (in lang | math functions), 1162
Unicode,
Code Point Collation, 908
Ersetzen von Sonderzeichen, 283
Konsortium, 2107
Standard, 2107
unparsed-entity-uri,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 1253
uppercase,
als MapForce-Funktion (in lang | string functions), 1180
URI,
in DTDs, 125
URL,
Dateien als Komponenten hinzufügen von, 80
UTF-8, 2107

V

Validator,
in Altova-Produkten, 2106
Validieren,
Mapping-Ausgabe, 107
Mapping-Design, 107
Variablen,
Beispiel für die Verwendung, 891, 893
Beispiele für die Verwendung, 892
DB-basiert, 882
einfache, 882
Geltungsbereich ändern, 888
in SPL, 1706
komplexe, 882
zum Mapping hinzufügen, 884
Verarbeitung,
Mappings automatisieren, 1434
Verbindungen,
alles kopieren, 93, 97
Alles kopieren (Sub-Einträge kopieren), 1528
ändern, 89
Annotation, 99
Arten, 93
beibehalten nach Löschen von Komponenten, 105

© 2016-2022 Altova GmbH

Verbindungen,
Eigenschaften, 1528
Einstellungen, 99
Einstellungen für 'Idente Sub-Einträge verbinden', 1528
erstellen, 89
fehlende übergeordnete Verbindungen, 89
Fehler nach Bearbeitung des Schemas beheben, 103
Idente Sub-Einträge automatisch verbinden, 1528
Idente Sub-Einträge verbinden, 1528
Kontextmenü, 101
kopieren, 89
löschen, 89
mixed, 93
obligatorische Inputs, 89
quellorientiert, 93
Quellorientiert (Mixed Content), 1528
reparieren, 103
selektiv markieren, 89
Standard, 93
Sub-Einträge kopieren, 93, 96
Tooltips zu Verbindungen anzeigen, 89
Typen, 99
verschieben, 89, 103
zielorientiert, 93
Zielorientiert (Standard), 1528
Verbindungsart,
alles kopieren, 97
mixed, 93
quellorientiert, 93
Standard, 93
Standard mit Mixed Content, 93
Sub-Einträge kopieren, 96
zielorientiert, 93
zielorientiert mit Mixed Content, 93
zielorientierte im Vergleich zu quellorientierten Verbindungen, 93
Vertrieb,
von Altova Software-Produkten, 2109
von Altova-Software-Produkten, 2109
Visual Basic,
Fehlerbehandlung, 1727
Integration von MapForce, 1947
Visual Studio,
Code generieren für, 1701
die MapForce ActiveX Controls zur Toolbox hinzufügen, 1922
Mapping-Code generieren für, 1567, 1572
Visual Studio Plug-in,
MapForce ausführen als, 1492
Vorangestellte Nullen,
bei der Konvertierung beibehalten, 521

W

WADL, 631
WebDAV Server,
Dateien als Komponenten hinzufügen von, 80
Webservice-Projekt,
erstellen, 1261
Webservices,
auf JSON mappen, 640
Authentifizierung im Vorhinein aktivieren, 635
Beispiel (REST), 640
Beispiel (WSDL), 648
einfache HTTPS-Authentifizierungsanmeldeinformationen bereitstellen, 635
hinzufügen, 612, 630
Konfiguieren des Timeout-Werts, 631
mit digitalen HTTPS-Zertifikaten signieren, 635
REST, 610, 612
REST oder SOAP, 610
REST-Aufrufe hinzufügen, 640
REST-Webservice-Parameter, 617
SOAP, 630
über HTTPS aufrufen, 664
Übersicht, 610
und Mapping-Kontext, 1335
UsernameToken hinzufügen, 638
von RSS Feed mappen, 644
WSDL, 610, 630
WSDL-basiert, 610
WS-Sicherheit aktivieren, 638
Webservices (SOAP),
implementieren, 1255
mit MapForce generieren, 1255, 1275, 1290
Voraussetzungen, 1255
weekday,
als MapForce-Funktion (in lang | datetime functions), 1147
weeknumber,
als MapForce-Funktion (in lang | datetime functions), 1148
Werte (Fenster),
Info, 1414, 1421
Kontext (Register), 1421
Verknüpft (Register), 1421

© 2016-2022 Altova GmbH

Werte (Fenster),
Verlauf (Register), 1421
Wertezuordnung,
als Mapping-Komponente, 921
Beispiele, 925, 927
Wert-Schlüssel-Paare,
im Mapping verwenden, 921
WHERE,
SQL WHERE-Komponente, 361
Wildcards,
Auswahl, 138
Schema importieren, 138
Wrapper-Schema, 138
xs:any/xs:anyAtrribute, 138
Windows,
Mappings mit Webservice-Aufrufen über HTTPS ausführen, 682
Unterstützung für Altova-Produkte, 2106
vertrauenswürdige Server-Zertifikate einrichten, 670
Word,
Mapping-Dokumentation generieren, 1399
Word 2007+,
Ausgabevorschau als, 1395
Wrapper-Klassen,
im generierten Code, 1701
write-binary-file,
als MapForce-Funktion (in lang | file functions), 1150
WSDL,
Anmerkungen zur Unterstützung, 1258
Aufrufen eines Webservice von einem Mapping aus, 648

X

X12,
X12 997 Functional Acknowledgement generieren, 481
X12 999 Implementation Acknowledgement generieren, 483
zusammengeführte Einträge, 551
XBRL,
allgemeine Einstellungen, 1543
allgemeine Optionen, 1543
applikationsweite Einstellungen, 1543
Beispiele, 816
benutzerdefinierte Taxonomiepakete, 1543
Concept Labels, 1543
Filing Indicators (EBA), 810
Generische Labels, 1543
hinzufügen, 769
Hypercubes, 801
Inline, 775
Komponenteneinstellungen, 792
Label Roles, 1543
Labels, 1543
Standard-Taxonomiepakete, 1543
Standardwerte, 797
Strukturansichten, 810
Tabellen, 810
Taxonomiepakete, 1543
unterstützte Standards, 768
xbrl-measure-currency,
als MapForce-Funktion, 1185
xbrl-measure-pure,
als MapForce-Funktion, 1186
xbrl-measure-shares,
als MapForce-Funktion, 1186
Xerces,
Code generieren für, 1701
xlsx-to-date,
als MapForce-Funktion, 1189
xlsx-to-datetime,
als MapForce-Funktion, 1189
xlsx-to-time,
als MapForce-Funktion, 1190
XML, 2107
"Detached" oder "Enveloped"Signatur, 147
als Mapping-Ziel, 585
aus TRADACOMS konvertieren, 523
Daten von CSV mappen auf, 585
Deklaration, 126
digitale Signatur, 126
in Datenbankfeld schreiben, 380
Kodierungseinstellungen, 126
Komponenteneinstellungen, 126
Schema-Version, 116
Signatureinstellungen, 144, 147
von Excel 2007+ auf XML mappen, 755
von UN/EDIFACT mappen, 531
XML data,
Lesen aus Datenbankfeldern, 377
Schreiben in Datenbankfelder, 377
XML Parser,
Info, 2106
XML-Dateien,
als globale Ressourcen, 1469
anhand einer einzigen XML-Quelle, 1377

© 2016-2022 Altova GmbH

XML-Dateien,
anhand von Excel generieren, 757
von Datenbankdatensätzen generieren, 1379
XML-Schema-Manager, 125
XML-Signatur,
Authentifizierungsmethode, 144
detached, 147
Einstellungen, 144
enveloped, 147
Platzierung der Signatur, 144
Schlüsselinformationen anhängen, 144
Transformationen, 144
XQuery,
benutzerdefinierte Funktionen hinzufügen, 992, 993
Erweiterungsfunktionen, 2086
Module importieren, 993
xs: any, 138
xs:anyAttribute, 138
xsi:nil,
als Attribut in einer XML-Instanz, 132
XSLT,
benutzerdefinierte Funktionen entfernen, 985
benutzerdefinierte Funktionen hinzufügen, 985
Erweiterungsfunktionen, 2086
Template Namespace, 985

Y

year-from-datetime,
als MapForce-Funktion (in lang | datetime functions), 1148
year-from-duration,
als MapForce-Funktion (in lang | datetime functions), 1149

Z

Z bis A,
Sortierkomponente, 908
Zeilen,
mappen von - Textdateien, 591
Zeilenkommentar, 393
Ziel, 27
Zieldatei,
mehrere Zieldateien anhand einer einzigen Quelldatei, 1377
Zielkomponente,
FlexText Komponente, 700
Verarbeitungsreihenfolge ändern, 1349
zip64mode,
in der build.xml-Datei aktivieren, 1700
Zusammengeführte Einträge, 551

© 2016-2022 Altova GmbH