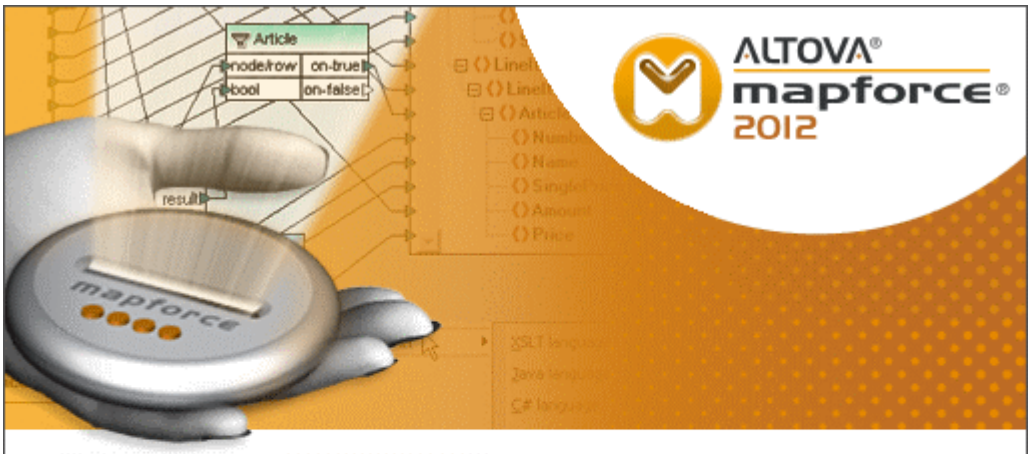
Benutzer- und Referenzhandbuch



Copyright © 1998–2011. Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSpy, MapForce, StyleVision, SemanticWorks, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, AltovaXML, MissionKit, and ALTOVA as well as their logos are trademarks and/or registered trademarks of Altova GmbH. Patent pending.

ALTOVA®

XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode Inc. This software contains 3rd party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html

Altova MapForce 2012 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: 2011

© 2011 Altova GmbH

Inhaltsverzeichnis

1 MapForce 2012 3

2 Was ist neu... 6

3 Übersicht über MapForce 10

3.1 Terminologie ... 12

4 MapForce Tutorial 18

4.1 Einrichten der Mapping-Umgebung ... 20
4.1.1 Hinzufügen von Komponenten zum Mapping-Fenster ... 21
4.2 Erstellen eines Mappings ... 24
4.2.1 Mappen von Schema-Elementen ... 25
4.2.2 Mappen von Daten mit Hilfe von Funktionen ... 28
4.2.3 Filtern von Daten ... 32
4.3 Generieren von XSLT 1.0- und 2.0-Code ... 36
4.4 Arbeiten mit mehreren Zielschemas / -dokumenten ... 37
4.4.1 Erstellen einer zweiten Zielschemakomponente ... 38
4.4.2 Anzeigen und Generieren mehrerer Zielschema-Ausgaben ... 41
4.5 Mappen mehrerer Quellelemente auf ein einziges Zielelement ... 43
4.5.1 Erzeugen der Mappings ... 45
4.5.2 Duplizieren von Input-Datenelementen ... 48
4.6 Input aus mehreren Dateien / Output in mehrere Dateien ... 53
4.6.1 Verarbeiten mehrerer Dateien pro Input-/Output-Komponente ... 54
4.7 Mappen einer Datenbank auf ein Schema ... 58
4.7.1 Herstellen einer Verbindung zu einer Access-Datenbank ... 60

5 MapForce Benutzeroberfläche 64

5.1 Das Register "Bibliotheken" ... 66
5.2 Mapping-Fenster ... 68
5.3 Die Fenster XSLT/XSLT2 ... 70

5.4 Fenster "Ausgabe" / BUILTIN-Ausführungsprozessor ... 71
5.5 Fenster im Zusammenhang mit StyleVision ... 73
5.6 Fenster "Übersicht" ... 74
5.7 Fenster "Meldungen" ... 75

6 Mapping zwischen Komponenten 78

6.1 Datenmappingmethoden (Standard / Mixed Content / Sub-Einträge kopieren) ... 80
6.1.1 Zielorientiert / Standard-Mapping ... 81
6.1.2 Quellorientiertes / Mixed Content Mapping ... 82
Mappen von Mixed Content ... 82
Mixed Content-Beispiel ... 87
Mappen von Mixed Content-Datenelementen mit Hilfe eines quellorientierten Mappings ... 88
6.1.3 "Alles kopieren" Konnektoren ... 90
6.2 Verbindungseinstellungen ... 93
6.3 Verschieben von Konnektoren ... 95
6.4 Verbindungen und Mapping-Ergebnisse ... 98
6.5 Fehlende Datenelemente ... 99
6.6 Verkettete Mappings / Weiterleitungskomponenten ... 103
6.6.1 Verkettete Mappings - Weiterleitung aktiv ... 105
6.6.2 Verkettete Mappings - Weiterleitung deaktiviert ... 110
6.6.3 Beispiel für ein verkettetes Mapping ... 115
6.7 Arbeiten mit Funktionen ... 117
6.8 Vorschau auf das Transformationsergebnis ... 120
6.9 Validieren von Mappings und Mapping-Ausgabe ... 122
6.10 Schleifen, Gruppen und Hierarchien ... 125
6.11 Mapping-Regeln und -Strategien ... 126

7 Reihenfolge der Verarbeitung von Mapping-Komponenten 132

8 Globale Ressourcen 136

8.1 Globale Ressoucen - Dateien ... 137
8.1.1 Definieren / Hinzufügen einer globalen Ressource ... 138
8.1.2 Zuweisen einer globalen Ressource ... 141
8.1.3 Verwendung / Aktivierung einer globalen Komponente ... 143
8.2 Globale Ressourcen - Ordner ... 145

8.3 Globale Ressourcen - Arbeitsablauf in Applikationen ... 149
8.3.1 Starten des Applikationsarbeitsablaufs ... 153
8.4 Globale Ressourcen - Eigenschaften ... 156

9 Dynamische Input/Output-Dateien pro Komponente 162

9.1 Dynamische Dateinamen - Input / Output ... 164
9.2 Dynamische Dateinamen als Input-Parameter ... 167
9.3 Mehrere XML-Dateien anhand einer einzigen XML-Quelldatei ... 169
9.4 Relative und absolute Dateipfade ... 172

10 Zwischenvariablen 178

10.1 Variablen - Anwendungsfälle ... 184

11 MapForce Tipps... 188

11.1 Filtern - Abrufen dynamischer Daten, Lookup-Tabelle ... 190
11.1.1 Filterkomponenten - Tipps ... 192
11.2 Wertezuordnung - Transformieren von Input-Daten ... 194
11.2.1 Übergeben von unveränderten Daten über eine Wertezuordnung ... 198
11.2.2 Eigenschaften der Wertezuordnungskomponente ... 201
11.3 Aggregatfunktionen: min, max, sum, count, avg ... 203
11.4 Mappings und das Root-Element von Zieldokumenten ... 205
11.5 Boolscher Vergleich von Input Nodes ... 206
11.6 Prioritätskontext Node/Datenelement ... 208
11.7 Zusammenführen mehrerer Dateien in einer Zieldatei ... 209
11.8 Befehlszeilenparameter ... 211
11.9 Befehlszeile - Definieren von Input/Output-Dateien ... 213
11.10 Input-Parameter - Standardeinstellungen und Einstellungen für die Vorschau ... 215
11.11 Node-Tests, Position und Gruppierung ... 218
11.11.1 Mappen fehlender Nodes - mit Hilfe von Not-exists ... 220
11.11.2 Position von Kontextelementen in einer Sequenz ... 222
11.11.3 Gruppieren von Nodes / Node-Inhalt ... 224
11.12 Verwendung von DTDs als "Schema-Komponenten" ... 231
11.13 Überprüfung der Typkonvertierung ... 232
11.14 Katalogdateien in MapForce ... 233
11.15 Derived XML-Schema-Typen - Mappen auf ... 237

11.16 Rekursives benutzerdefiniertes Mapping 239
11.16.1 Definieren eines rekursiven benutzerdefinierten Mappings 241

12 Nullwerte / Nillable Werte 250

13 Datenbanken und MapForce 254

13.1 Tutorial: Mappen von XML-Daten auf Datenbanken 255
13.1.1 Datenbankaktion: Einfügen 256
Einfügen von Daten in eine Datenbanktabelle 256
Einfügen von Tabellen und damit in Zusammenhang stehenden untergeordneten Tabellen 259
13.1.2 Datenbankaktion: Aktualisieren 264
Aktualisieren von Datenbankfeldern 264
Aktualisieren und Hinzufügen von neuen Datensätzen 267

14 Mappen von HL7 v3.x von/auf XML-Schemas 274

15 Bibliotheken und Funktionen 276

15.1 Definieren benutzerdefinierter Funktionen 277
15.1.1 Funktionsparameter 283
15.1.2 Inline-Funktionen und reguläre benutzerdefinierte Funktionen 287
15.1.3 Erstellen einer einfachen Lookup-Funktion 289
15.1.4 Komplexe benutzerdefinierte Funktion - XML-Node als Input 294
Definieren komplexer Input-Komponenten 295
15.1.5 Komplexe benutzerdefinierte Funktion - XML Node als Output 300
Definieren komplexer Output-Komponenten 300
15.1.6 Benutzerdefinierte Funktion - Beispiel 306
15.2 Hinzufügen benutzerdefinierter XSLT- und XQuery-Funktionen 311
15.2.1 Benutzerdefinierte XSLT 1.0-Funktionen 312
15.2.2 Benutzerdefinierte XSLT 2.0-Funktionen 317
15.2.3 Aggregatfunktionen - Aggregieren von Nodes in XSLT1 und 2 318
15.3 Hinzufügen benutzerdefinierter Java .class- und .NET DLL-Funktionen 321
15.4 Java- und .NET-Funktionen - Details 323
15.5 Referenz Funktionen 326
15.5.1 core 327
aggregates (Aggregatfunktionen) 327
conversion functions (Konvertierungsfunktionen) 330

file path functions (Dateipfadfunktionen) ... 339
logical functions (logische Funktionen) ... 341
math functions (mathematische Funktionen) ... 343
node functions (Node-Funktionen) ... 345
sequence functions (Sequenzfunktionen) ... 348
string functions (String-Funktionen) ... 349
Tokenize Beispiele ... 353
Regular Expressions ... 356
15.5.2 xpath2 ... 360
accessors (Accessor-Funktionen) ... 360
anyURI functions (Alle URI-Funktionen) ... 360
boolean functions (Boolesche Funktionen) ... 361
constructors (Konstruktoren) ... 361
context functions (Kontextfunktionen) ... 362
durations, date and time functions (Zeitdauer-, Datums- und Uhrzeitfunktionen) ... 363
node functions (Node-Funktionen) ... 365
numeric functions (numerische Funktionen) ... 366
qname-related functions (QName-Funktionen) ... 367
string functions (String-Funktionen) ... 367
15.5.3 xslt ... 370
xpath functions (XPath-Funktionen) ... 370
xslt functions (XSLT-Funktionen) ... 372

16 Unterstützung für QName 376

17 Benutzerreferenz 380

17.1 Datei ... 381
17.2 Bearbeiten ... 384
17.3 Einfügen ... 385
17.4 Komponente ... 386
17.5 Verbindung ... 390
17.6 Funktion ... 395
17.7 Ausgabe ... 398
17.8 Ansicht ... 400
17.9 Extras ... 402
17.10 Fenster ... 408
17.11 Hilfe ... 409
17.11.1 Inhaltsverzeichnis, Index, Suchen ... 410

17.11.2 Aktivierung, Bestellformular, Registrieren, Updates ... 411
17.11.3 Weitere Befehle ... 413

18 Anhänge 416

18.1 Informationen zu den Prozessoren ... 417
18.1.1 XSLT 1.0-Prozessor: Implementierungsinformationen ... 418
18.1.2 XSLT 2.0-Prozessor: Implementierungsinformationen ... 420
Allgemeine Informationen ... 420
XSLT 2.0-Elemente und -Funktionen ... 422
18.1.3 Erweiterungen ... 423
Java-Erweiterungsfunktionen ... 423
Benutzerdefinierte Klassendateien ... 424
Benutzerdefinierte Jar-Dateien ... 427
Java: Konstruktoren ... 428
Java: Statische Methoden und statische Felder ... 428
Java: Instanzmethoden und Instanzfelder ... 429
Datentypen: XPath/XQuery in Java ... 430
Datentypen: Java in XPath/XQuery ... 431
.NET-Erweiterungsfunktionen ... 431
.NET: Konstruktoren ... 433
.NET: Statische Methoden und statische Felder ... 434
.NET: Instanzmethoden und Instanzfelder ... 435
Datentypen: XPath/XQuery in .NET ... 436
Datentypen: .NET in XPath/XQuery ... 437
MSXSL Scripts für XSLT ... 437
Altova Erweiterungsfunktionen ... 439
Allgemeine Funktionen ... 440
18.2 Technische Daten ... 444
18.2.1 OS- und Arbeitsspeicheranforderungen ... 445
18.2.2 Altova XML Parser ... 446
18.2.3 Altova XSLT- und XQuery-Prozessor ... 447
18.2.4 Unicode-Unterstützung ... 448
Windows XP ... 448
Rechts nach Links-Schreibrichtung ... 449
18.2.5 Internet-Verwendung ... 450
18.3 Lizenzinformationen ... 451
18.3.1 Electronic Software Distribution ... 452
18.3.2 Software-Aktivierung und Lizenzüberwachung ... 453
18.3.3 Rechte am geistigen Eigentum ... 454
18.3.4 Altova Endbenutzer-Lizenzvereinbarung ... 455

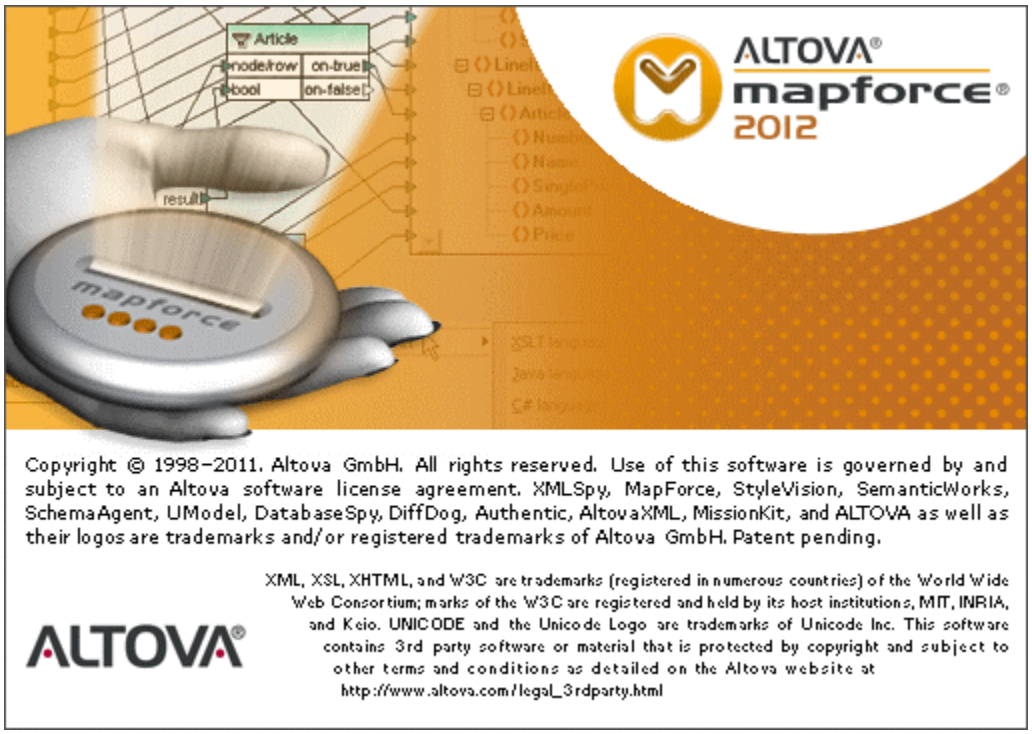
Index

Kapitel 1

MapForce 2012

1 MapForce 2012

MapForce® 2012 Basic Edition ist ein visuelles Mapping Tool für professionelle Datenintegrationsprojekte. MapForce® ist eine 32/64-Bit Windows Applikation, die auf Windows 7, Windows Vista, Windows Server 2003/2008 und Windows XP läuft. 64-Bit-Unterstützung steht für die Enterprise und die Professional Edition zur Verfügung.



Copyright © 1998–2011. Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSpy, MapForce, StyleVision, SemanticWorks, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, AltovaXML, MissionKit, and ALTOVA as well as their logos are trademarks and/or registered trademarks of Altova GmbH. Patent pending.

ALTOVA®

XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode Inc. This software contains 3rd party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html

Kapitel 2

Was ist neu...

2 Was ist neu...

MapForce Version **2012** enthält die folgenden neuen Funktionen:

- Automatisches Ausrichten von Komponenten im Mapping-Fenster
- Aufforderung eine Verbindung mit dem Parent-Ziel-Node herzustellen
- Spezielle Regeln, mit denen festgelegt wird, in welcher Reihenfolge Komponenten in einem Mapping verarbeitet werden

MapForce Version 2011R3 enthält die folgenden neuen Funktionen:

- Zwischenvariablen

MapForce Version 2011R2 enthält die folgenden neuen Funktionen:

- Suchen von Funktionen im Fenster "Bibliothek"
- Reverse mapping
- Erweiterbare IF-ELSE-Funktion
- Die Funktionen Node Name und Parser-Funktionen in der Core-Bibliothek

MapForce Version 2011 enthält die folgenden neuen Funktionen:

- Möglichkeit, mit Hilfe von StyleVision Power Stylesheets eine Vorschau von Zielkomponenten zu erstellen
- Funktion zur Anzeige einer Vorschau von Zwischenkomponenten in einer Mapping-Kette von zwei oder mehr Komponenten, die mit einer Zielkomponente verbunden sind (Weiterleitungsvorschau).
- Formatierungsfunktionen für dateTime- und numbers-Datentypen für alle unterstützten Programmiersprachen
- Verbesserungen an der auto-number Funktion
- Neue Zeitzonenfunktionen: remove-timezone und convert-to-utc

MapForce Version 2010 Release 3 enthält die folgenden neuen Funktionen:

- Unterstützung für Nillable Werte und das xsi:nil Attribut in XML-Instanzdateien
- Möglichkeit, die automatische Konvertierung in Zieltypen in XML-Dokumenten zu deaktivieren

MapForce Version 2010 Release 2 enthält die folgenden neuen Funktionen:

- Automatische Verbindung identischer Child-Konnektoren beim Verschieben eines Parent-Konnektors
- Möglichkeit, Input-Strings für die weitere Verarbeitung in Teile aufzuteilen

MapForce Version 2010 enthält die folgenden neuen Funktionen:

- Mehrere Input/Output-Dateien pro Komponente
- Verbesserte Unterstützung von relativen Pfaden
- Verwendung von derived types dank Unterstützung von xsi:type
- Neues internes Datentypsystem
- Verbesserte Navigation in benutzerdefinierten Funktionen
- Verbesserte Behandlung von Mixed Content in XML-Elementen

© 2011 Altova GmbH

MapForce Version 2009 SP1 enthält die folgenden neuen Funktionen:

- Die Reihenfolge der Parameter in benutzerdefinierten Funktionen kann durch den Benutzer festgelegt werden
- Möglichkeit XML-Dateien, die entsprechend dem XML-Schema nicht gültig sind, trotzdem zu validieren
- Reguläre benutzerdefinierte (Standard)funktionen unterstützen nun komplexe hierarchische Parameter
- Xerces 3.x Unterstützung beim Generieren von C++ Code

MapForce Version 2009 enthält die folgenden neuen Funktionen:

- EDI HL7 Versionen 3.x XML als Quell- und Zielkomponenten
- Gruppierung von Nodes oder Node-Inhalt
- Möglichkeit zum Filtern von Daten auf Basis einer Node-Position in einer Sequenz
- QName-Unterstützung
- Suchen von Datenelementen/Nodes in Komponenten

MapForce Version 2008 Release 2 enthält die folgenden neuen Funktionen:

- Funktion zum automatischen Generieren von XML-Schemas für XML-Dateien.
- Unterstützung für globale Altova-Ressourcen.
- Verbesserte Leistung

MapForce Version 2008 enthält die folgenden neuen Funktionen:

- Aggregatfunktionen
- Lookup-Komponente für Wertezuordnung
- Optionen zur verbesserten XML-Ausgabe: Pretty Print für XML-Ausgabe, XML-Schema Referenz auslassen und Kodierungseinstellungen für einzelne Komponenten.
- diverse interne Updates

MapForce Version 2007 Release 3 enthält die folgenden neuen Funktionen:

- Optimierter Code Generator und verbesserte Dokumentation

Kapitel 3

Übersicht über MapForce

3 Übersicht über MapForce

Altova Website: Einführung in MapForce

Was ist Mapping?
Im Prinzip wird der Inhalt einer Komponente auf eine andere Komponente gemappt, also abgebildet, bzw. dorthin transformiert. Ein XML- oder Textdokument kann auf ein anderes XML-Zieldokument gemappt werden. Die Transformation erfolgt über ein automatisch generiertes XSLT 1.0 oder 2.0 Stylesheet.

Außerdem kann MapForce in einer einzigen Komponente mehrere Input-Dateien eines Verzeichnisses verarbeiten und mehrere Dateien ebenfalls in einer einzigen Komponente ausgeben.

Bei der Erzeugung einer XSLT-Transformation wird ein **Quellschema** auf ein **Zielschema** gemappt. Dabei werden Elemente/Attribute im Quellschema mit anderen Elementen/Attributen im Zielschema "verbunden". Da ein XML-Dokument mit einer Schemadatei verknüpft ist und dadurch definiert wird, werden im Endeffekt zwei XML-Dokumente aufeinander gemappt.

MapForce® unterstützt:

- Grafisches Mapping auf und von jeder Kombination und Anzahl von:
  - XML-Schemas als Quell- oder Zieldatei

  Die **Professional** Edition unterstützt außerdem:
  - Flat Files: kommagetrennt (CSV) und Formate mit fester Länge als Quelle und Ziel
  - Relationale Datenbanken als Quelle und Ziel

  Die **Enterprise** Edition unterstützt außerdem:
  - EDI-Dateien: UN/EDIFACT, ANSI X12 einschließlich HIPAA, HL7 2.x, IATA PADIS und SAP IDocs als Quelle und Ziel
  - FlexText™-Dateien als Quelle und Ziel
  - Office Open XML Excel 2007 (und höher)-Dateien als Quelle und Ziel
  - XBRL-Instanzdateien und Taxonomien
- Automatische Codegenerierung
  - XSLT 1.0 und 2.0

  Professional Edition und **Enterprise** Edition unterstützen außerdem:
  - XQuery
  - Java, C# und C++
  - 64-Bit-Version
- On-the-fly-Transformation und Vorschau auf alle Mappings ohne Code-Generierung oder -Kompilierung
- die Anzeige einer Vorschau auf Zwischenkomponenten in einer Mapping-Kette bestehend aus zwei oder mehr mit einer Zielkomponente verbundenen Komponenten (Weiterleitungsvorschau)
- die Anzeige einer Vorschau auf Zielkomponenten mit Hilfe von StyleVision Power Stylesheets
- Leistungsstarker visueller Function Builder zur Erstellung benutzerdefinierter Funktionen
- Aufrufen der MapForce Benutzeroberfläche und -funktionen über die MapForce API (ActiveX Control)
- Definition von benutzerdefinierten XSLT 1.0- und 2.0-Bibliotheken
- Unterstützung für XPath 2.0-Funktionen in XSLT 2.0 und XQuery

 © 2011 Altova GmbH

- Definition von benutzerdefinierten Komponenten/Funktionen mit komplexen Inputs/Outputs
- Unterstützung für quellorientiertes Mapping / Mapping von Mixed Content und Alles-kopieren-Verbindungen
- Automatische Beibehaltung von Mapping-Konnektoren fehlender Nodes/Datenelemente
- Unterstützung für HL7 Version 3.x, da diese XML-Schema-basiert ist

Die **Professional** Edition unterstützt zusätzlich:

- XML-Datenmapping von/auf Datenbanken - IBM DB2 und andere
- Direkte Abfrage von Datenbanken
- SQL-WHERE-Filter und SQL-Anweisungsassistent
- SQL SELECT-Anweisungen als Datenquellen für das Mapping
- Integration benutzerdefinierter C++, Java- und C#-Funktionen
- Projektverwaltungsfunktionen zum Gruppieren von Mappings
- MapForce Plug-in für Eclipse 3.4 / 3.5 / 3.6
- MapForce für Microsoft Visual Studio
- Dokumentation des Mapping-Designs

Die **Enterprise** Edition unterstützt zusätzlich:die Anzeige

- Erstellung von SOAP 1.1 und SOAP 1.2 Webserviceprojekten und Mapping von Webservice-Operationen von WSDL 1.1 und WSDL 2.0-Dateien
- Direkter Aufruf von Webservice-Funktionen
- FlexText™: verbesserte Verarbeitung von Altdatenbeständen (Legacy-Dateien)

Alle Transformationen lassen sich in ein und derselben Arbeitsumgebung durchführen, in der mehrere Quellen und Ziele gemischt werden können und umfassende und erweiterbare Funktionsbibliotheken für alle Arten von Datenmanipulationen zur Verfügung stehen.

3.1 Terminologie

Im Folgenden sind die in dieser Dokumentation verwendeten Begriffe definiert.

Bibliothek
Eine Bibliothek ist eine Sammlung von Funktionen, die im Bibliotheksfenster zu sehen sind. Es gibt verschiedene Funktionstypen, Basisfunktionen (Core) und computersprachspezifische sowie benutzerdefinierte Funktionen. Nähere Informationen dazu finden Sie im Abschnitt "Funktionen".

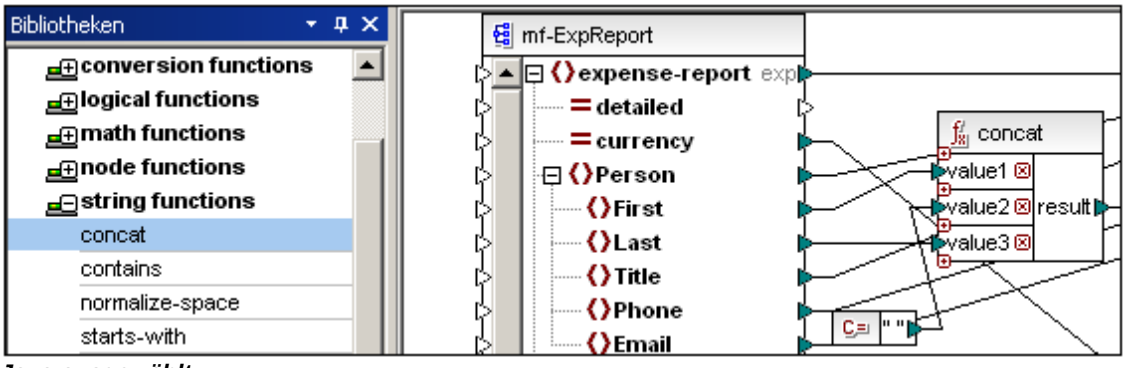
Komponente
In MapForce werden viele der grafischen Elemente, die Sie auf der Mapping-Registerkarte einfügen/importieren oder platzieren können, zu Komponenten. Komponenten haben kleine **Dreiecke**, über die Sie Daten zwischen Quell- und Zielkomponenten mappen können, indem Sie Verbindungen zwischen diesen ziehen.

Komponenten erkennt man an den kleinen **Dreiecken**, die sie besitzen. Mit Hilfe dieser Dreiecke (**Input-** und **Output-Symbole**) können Sie Daten aufeinander mappen, indem Sie eine Verbindung zwischen ihnen erstellen.

Die folgenden Dateien werden zu Komponenten, wenn Sie in den Mapping-Bereich platziert werden:

- Schemas und DTDs: Quell- und Zielschemas
-
- Funktionstypen: XSLT/XSLT2 sowie Konstanten, Filter und Bedingungen

Funktion
Bei einer Funktion handelt es sich um eine vordefinierte Komponente, die Datenoperationen durchführt. Datenoperationen wie z.B. **Concat**. Funktionen haben **Eingabe-** und/oder **Ausgabeparameter,** wobei es für jeden Parameter ein eigenes Input/Output-Symbol gibt. Die Funktionen stehen im Bibliotheksfenster zur Verfügung und sind nach logischen Kriterien gruppiert. Wenn Sie Strg+F drücken, können Sie nach einer Funktion suchen. Wenn Sie eine Funktion in das Mapping-Fenster ziehen, wird eine Funktionskomponente erstellt. Nähere Informationen dazu finden Sie im Abschnitt Funktionen und Bibliotheken.

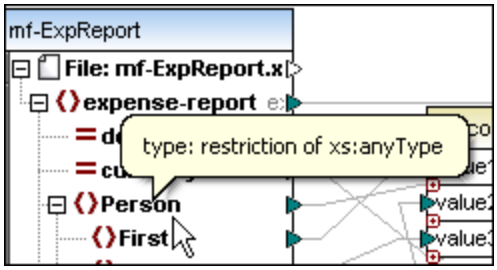


Java ausgewählt

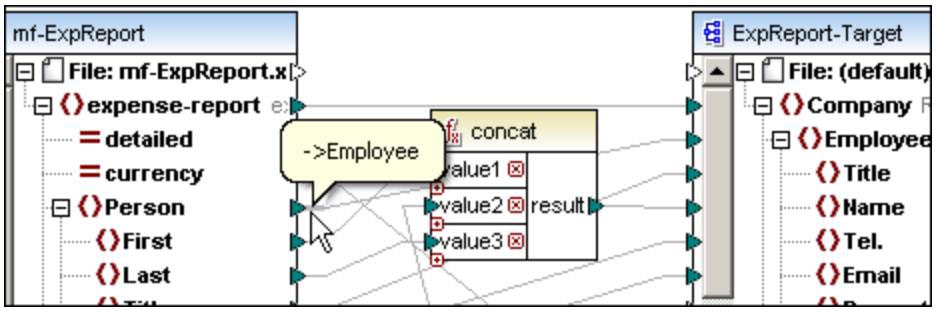
Datenelement
Ein Datenelement ist eine Dateneinheit, die von einer Komponente auf eine andere gemappt werden kann. Es kann sich dabei um ein **Element**, ein **Attribut** handeln.

Jedes Datenelement verfügt über ein **Input-** und ein **Output-Symbol**. Um ein Mapping zwischen zwei Datenelementen zu erzeugen, muss es sich **nicht** um denselben Typ (Element oder Attribut) handeln.

 © 2011 Altova GmbH



Eingabe-, Ausgabesymbol
Die kleinen Dreiecke neben den Komponenten sind Input- und Output-Symbole. Durch Anklicken und Ziehen eines Symbols erstellen Sie einen **Konnektor**, der eine Verbindung zu einem anderen Symbol herstellt, sobald Sie die Maustaste loslassen. Der Konnektor **kennzeichnet ein Mapping** zwischen den zwei durch die Symbole markierten Daten-Sets. Nähere Informationen dazu finden Sie im Abschnitt "Mapping zwischen Komponenten".



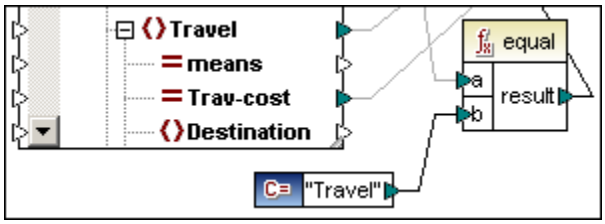
Konnektor
Der Konnektor ist die Verbindungslinie zwischen zwei Symbolen. Er kennzeichnet das Mapping zwischen den beiden durch die Symbole markierten Datensätzen. Nähere Informationen dazu finden Sie im Abschnitt "Mapping zwischen Komponenten".

Es können mehrere Arten von Konnektoren definiert werden:

- Zielorientierte (Standard) Konnektoren, siehe: Quellorientiertes/Mixed Content und Standard-Mapping
- "Alles kopieren" Konnektoren, siehe ""Alles kopieren" Konnektoren"
- Quellorientierte (Mixed Content) Konnektoren, siehe Quellorientiertes und Mixed Content Mapping

Konstante

Eine Konstante ist eine Komponente, die für ein Input-Symbol einer Funktion oder Komponente konstante Daten zur Verfügung stellt. So ist. z.B. der String "Travel" mit dem Parameter "b" der equal-Funktion verbunden. Beim Erstellen der Komponente oder wenn Sie darauf doppelklicken werden diese Daten in ein Dialogfeld eingegeben. Eine Konstantenfunktion hat nur ein Output-Symbol. Sie können zwischen den folgenden Datentypen wählen: String, Zahl und Alle anderen (String).

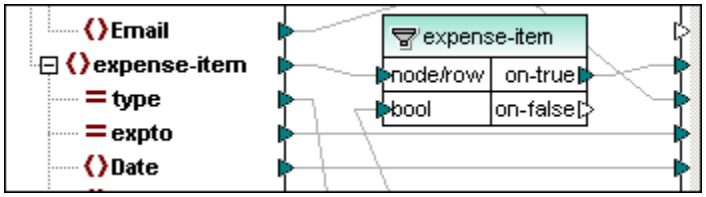


Variable

Fügt eine Zwischenvariable ein, die einer regulären (nicht-inline gesetzten) benutzerdefinierten Funktion entspricht. Variablen sind Strukturkomponenten ohne Instanzdateien und dienen zum Vereinfachen eines Mappings.

Filter: Node/Zeile

Ein Filter ist eine Komponente, die Daten über zwei Eingabe- und Ausgabeparameter filtert: **node/row** und **bool** und **on-true**, **on-false**. Ist der Boolesche Wert "true", wird der Wert/Inhalt des node/row-Parameters in den **on-true**-Parameter übernommen.

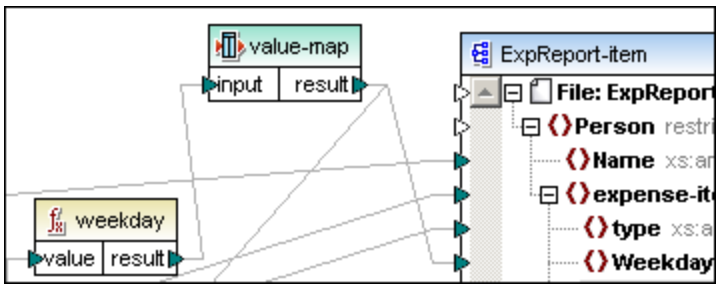


Der **on-false**-Ausgabeparameter gibt das vom Mapping definierte Komplement-Node-Set aus. Nähere Informationen dazu finden Sie unter Mehrere Zielschemas / -dokumente.

Wertezuordnung

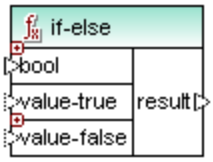
Mit Hilfe der Wertezuordnungskomponente können Sie eine Gruppe von Eingabedaten anhand einer Lookup-Tabelle in eine andere Gruppe von Ausgabedaten transformieren. Wenn Sie auf die Komponente doppelklicken, wird die Wertezuordnungstabelle geöffnet. In der linken Tabellenspalte wird der Input definiert, in der rechten Spalte sind die transformierten Daten, die ausgegeben werden sollen, definiert.

© 2011 Altova GmbH



IF-Else-Bedingung

Eine Bedingung ist eine Komponente, die gestattet, je nach Ergebnis einer vordefinierten Situation unterschiedliche Daten-Sets weiterzugeben. Im Titel der Komponente wird der Text **if-else** angezeigt. Ein Beispiel finden Sie im Referenzabschnitt unter "Bedingung".



- Der erste Eingabewert ist ein **Boolscher** Wert, der die Daten enthält, auf die hin der Datensatz überprüft wird.
- Der Eingabeparameter **value-true** enthält die Daten, die übernommen werden sollen, wenn die Bedingung zutrifft, also "true" ist.
- Der Parameter **value-false** enthält die Daten, die übernommen werden sollen, wenn die Bedingung nicht zutrifft, also "false" ist.
- Der Parameter **result** gibt die Daten der Eingabeparameter value-true/false aus.

Die IF-Else-Funktion ist nun **erweiterbar**. Das bedeutet, dass überprüft werden kann, ob mehrere IF-Werte vorhanden sind. Die Else-Bedingung/der Else-Wert kann über den Parameter **otherwise** ausgegeben werden. Nähere Informationen dazu finden Sie unter Einfügen | If-Else .

Kapitel 4

MapForce Tutorial

4 MapForce Tutorial

In diesem Tutorial erhalten Sie anhand einer Reihe von Aufgaben einen Überblick darüber, wie Sie die Funktionalitäten von MapForce 2012 am besten einsetzen können.

Ziel dieses Tutorials ist eine einfache Reisekostenabrechnung eines Mitarbeiters auf eine komplexere Firmenspesenabrechnung zu mappen. In unserem Tutorial-Beispiel füllt jeder Mitarbeiter die Felder seiner persönlichen Reisekostenabrechnung aus. Diese Abrechnung wird auf die Firmenspesenabrechnung gemappt und in die Verwaltungsabteilung geroutet. Nun müssen zusätzliche Daten im Zusammenhang mit dem Mitarbeiter eingegeben werden. Das Ergebnis ist eine standardisierte Firmenspesenabrechnung.

In diesem Tutorial lernen Sie Folgendes:

- Einrichten der Mapping-Umgebung
- Mappen der XML-Quelldatei (die persönliche Reisekostenabrechnung) auf die Zielkomponente (die Firmen-Spesenabrechnung)
- Anwendung von Filtern auf die Quelldaten
- Generierung einer XSLT-Transformationsdatei
- Transformation der Quelldaten in die Zieldaten mit Hilfe der generierten XSLT-Datei

Installation und Konfiguration
Es wird in diesem Tutorial davon ausgegangen, dass Sie MapForce erfolgreich auf Ihrem Rechner installiert haben und einen kostenlosen Evaluierungs-Keycode erhalten haben bzw. ein registrierter Benutzer des Produkts sind. Die Evaluierungsversion von MapForce hat den vollen Funktionsumfang, ist jedoch auf einen 30-Tage-Zeitraum befristet. Sie können eine reguläre Lizenz über unseren sicheren Webserver oder einen unserer Vertriebspartner erwerben.

Tutorial-Beispieldateien
In diesem Tutorial wird mit den folgenden Komponenten gearbeitet:

- Quell- und (mehrere) Zielschemas
- verschiedenen Funktionen u.a.: concat, filter, equal und Konstanten

Sie finden alle in diesem Tutorial verwendeten Dateien im Ordner **C:\Documents and Settings\All Users\Application Data\Altova**. Wenn die Applikation von einem Benutzer zum ersten Mal gestartet wird, werden die Beispieldateien für diesen Benutzer in den Ordner ...\MapForceExamples\Tutorial\ kopiert. Daher sollten die Beispieldateien im ursprünglichen Verzeichnis ...\All Users\.... nicht verschoben, bearbeitet oder gelöscht werden.

Mitinstalliert werden auch die XSLT-Dateien und die transformierten XML-Dateien. Im Tutorial werden die folgenden Dateien verwendet:

Persönliche Reisekostenabrechnung:

- Tut-ExpReport.mfd — Das Mapping für die Reisekostenbabrechnung (ein einziges Ziel)
- Tut-ExpReport-multi.mfd — Die Ziel-Spesenabrechnung, basierend auf mehreren Schemas
-
- mf-ExpReport.xml — Persönliche Reisekostenabrechnung als XML-Instanzdokument
- mf-ExpReport.xsd — Verknüpfte Schemadatei

Firmen-Spesenabrechnung:

© 2011 Altova GmbH

- ExpReport-Target.xml XML-Instanzdokument für die Firmen-Spesenabrechnung
- ExpReport-Target.xsd Verknüpfte Schemadatei

Dateipfade in Windows XP, Windows Vista und Windows 7

Die in dieser Dokumentation angegebenen Dateipfade sind nicht für alle Betriebssysteme gleich. Sie sollten die folgenden Übereinstimmungen beachten:

- *Verzeichnis (Eigene) Dateien:* Das Verzeichnis Eigene Dateien in Windows XP entspricht in Windows Vista und Windows 7 dem Verzeichnis Dateien. Es befindet sich standardmäßig an den nachstehend angeführten Orten. Beispieldateien befinden sich im Normalfall in einem Unterverzeichnis des Verzeichnisses (Eigene) Dateien.

Windows XP	`C:\Dokumente und Einstellungen\<Benutzername>\Eigene Dateien`
Windows Vista, Windows 7	`C:\Benutzer\<Benutzername>\Dateien`

- *Anwendungsverzeichnis:* Das Anwendungsverzeichnis ist jener Ordner, in dem sich Ihre Altova Anwendung befindet. Der Pfad zum Anwendungsverzeichnis ist standardmäßig folgender:

Windows XP	`C:\Programme\Altova`
Windows Vista, Windows 7	`C:\Programme\Altova`
32-Bit-Version auf 64-Bit OS	`C:\Programme (x86)\Altova`

4.1 Einrichten der Mapping-Umgebung

In diesem Abschnitt wird erklärt, wie man das Quell- und Zielschema für das Mapping definiert.

Ziel

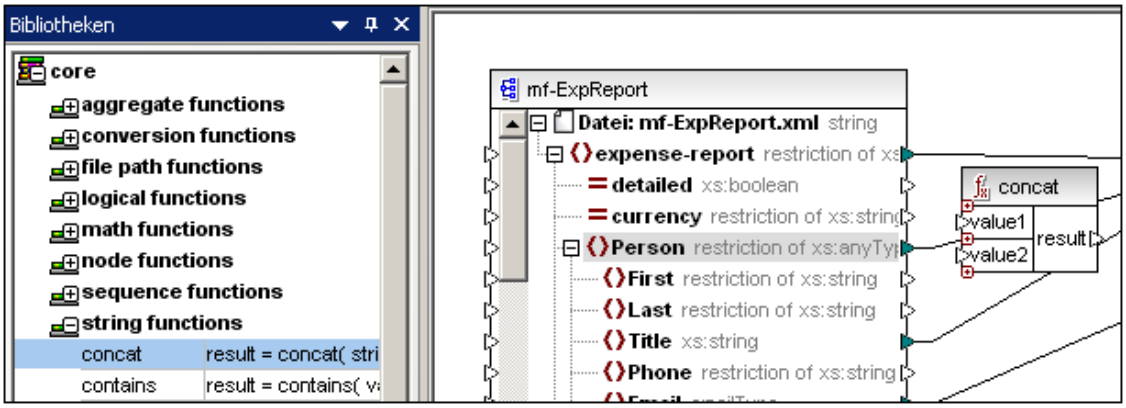
In diesem Abschnitt des Tutorials lernen Sie, wie man die Mapping-Umgebung in MapForce einrichtet. Insbesondere werden folgende Schritte beschrieben:

- Erstellen der Quell- und Zielschemakomponente
- Definieren der XML-Quelldatei
- Auswählen des Root-Elements des Zielschemas

In diesem Abschnitt verwendete Befehle

XML Schema/Datei einfügen: Klicken Sie auf diese Schaltfläche, um das Windows-Standarddialogfeld "Öffnen" aufzurufen und die Datei aus Ihrem Dateisystem auszuwählen.

Bitte beachten Sie:
Die Bildunterschriften unterhalb der Abbildungen zu den einzelnen Bibliotheksfenstern (hier z.B. ***XSLT ausgewählt)*** geben an, welche Ziel-/Ausgabesprache gerade verwendet wird, wenn Sie zur Vorschau auf das Mapping auf das Register "Ausgabe" klicken. Die ausgewählte Sprache bestimmt auch, welche Funktionen im Bibliotheksfenster angezeigt werden.



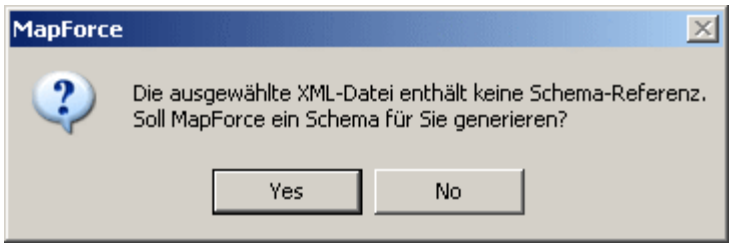
XSLT ausgewählt

 © 2011 Altova GmbH

4.1.1 Hinzufügen von Komponenten zum Mapping-Fenster

Nachdem Sie MapForce gestartet haben, müssen Sie die Quell- und Zieldateien in einem Mapping-Fenster hinzufügen; Sie können die Dateien auch hinzufügen, indem Sie sie mit der Maus aus dem Windows Explorer in das Mapping-Fenster ziehen.

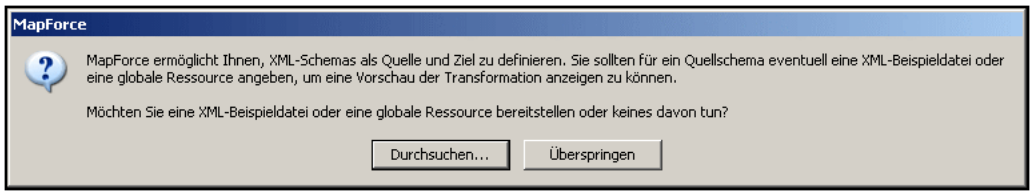
MapForce kann anhand einer bestehenden XML-Datei automatisch ein XML-Schema generieren, wenn kein XML-Schema vorhanden ist. Beim Einfügen einer XML-Datei über den Menübefehl "XML-Schema einfügen" aus dem Menü "Datei" erscheint automatisch ein Dialogfeld, in dem Sie gefragt werden, ob ein XML-Schema erstellt werden soll, wenn keine dazugehörige XML-Schema-Datei gefunden wird.



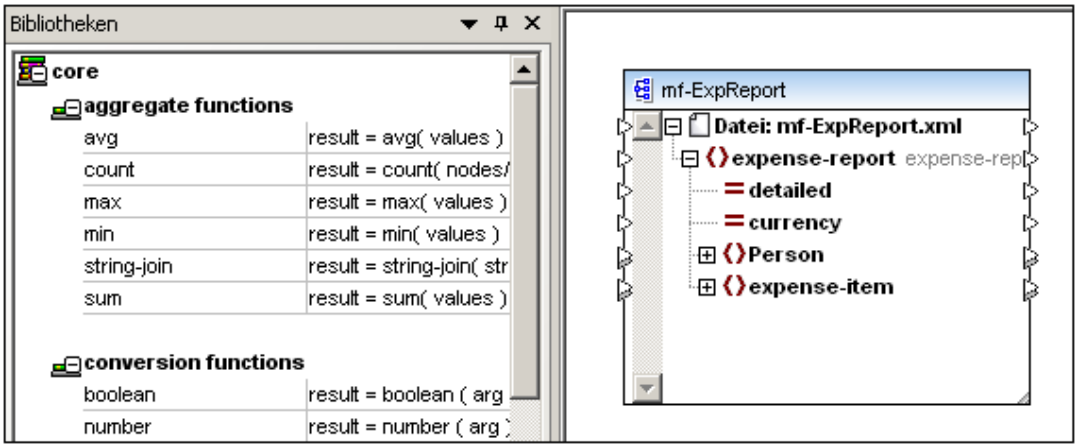
Bei Generierung eines Schemas anhand einer XML-Datei müssen die Datentypen für Elemente/Attribute anhand des XML-Instanzdokuments abgeleitet werden und entsprechen eventuell nicht genau Ihren Erwartungen. Überprüfen Sie bitte, ob das generierte Schema tatsächlich die Instanzdaten genau darstellt.

Erzeugen der Quellschema-Komponente:

1. Klicken Sie auf die Schaltfläche **XML-Schema/Datei einfügen** oder wählen Sie den Menübefehl **Einfügen | XML-Schema/Datei...**
2. Wählen Sie im Dialogfeld "Öffnen" unter ...\MapForce2012\MapForceExamples im Unterordner "Tutorial" die Datei mf-ExpReport.xsd.
   Sie werden nun aufgefordert, eine XML-Beispieldatei anzugeben, aus der die Daten für den Vorschaubereich verwendet werden sollen.

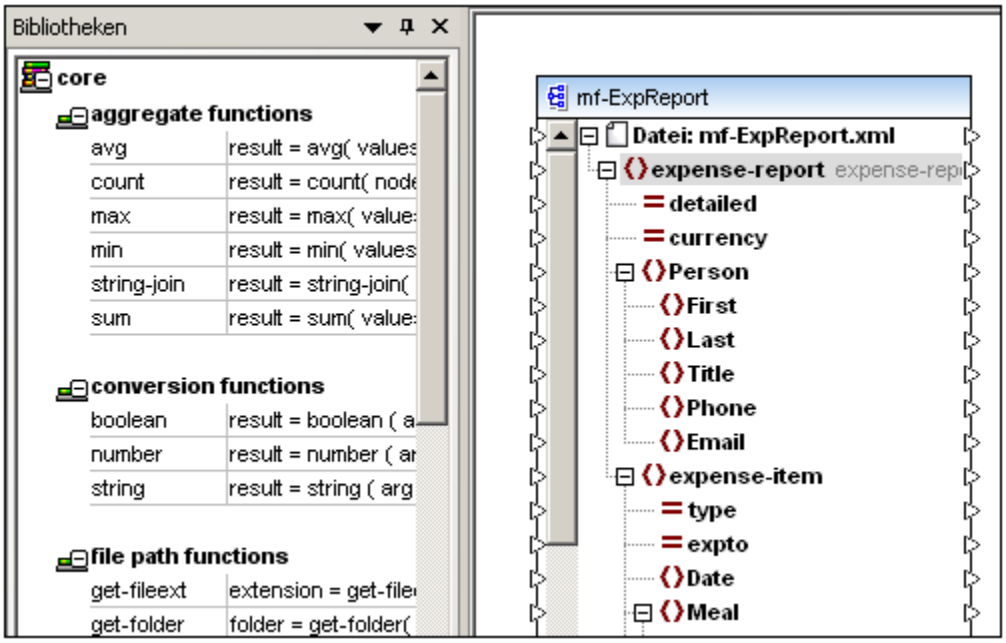


3. Klicken Sie auf die Schaltfläche **Durchsuchen...** und wählen Sie die Datei **mf-ExpReport.xml**.
   Die Quellschemakomponente wird nun im Mapping-Fenster angezeigt.



XSLT ausgewählt

4. Klicken Sie auf den Eintrag **expense-report** der Komponente und drücken Sie auf dem numerischen Tastenblock die ***-**Taste, um alle Datenelemente anzuzeigen.
5. Klicken Sie auf das Symbol zum Anpassen der Größe in der rechten unteren Ecke der Komponente und ziehen Sie es, um die Fenstergröße anzupassen.
   Anmerkung: Wenn Sie auf das Symbol doppelklicken, erhält die Komponente die "**optimale Größe**", so dass alle Elemente zu sehen sind.

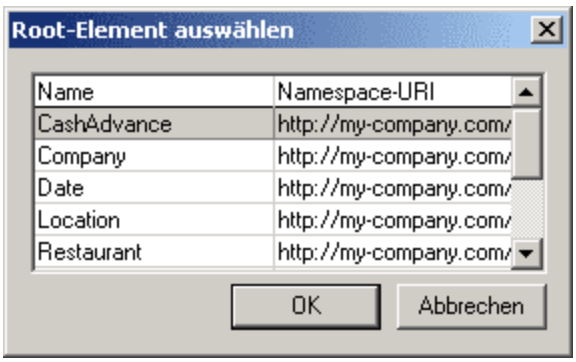


XSLT ausgewählt

Erzeugen der Zielschema-Komponente:

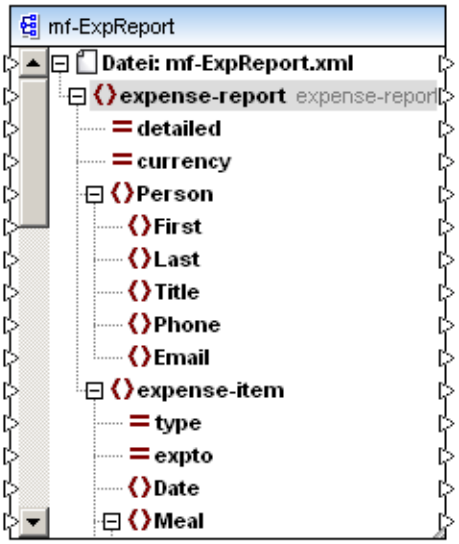
1. Klicken Sie auf das Symbol **XML-Schema/Datei einfügen** oder wählen Sie den Menübefehl **Einfügen | XML-Schema/Datei...**
2. Wählen Sie im Dialogfeld "Öffnen" die Datei **ExpReport-Target.xsd**.
   Sie werden nun aufgefordert, eine XML-Beispieldatei für das Schema anzugeben.
3. Klicken Sie auf die Schaltfläche **Überspringen** und wählen Sie **Company** als Root-Element des Zieldokuments aus.

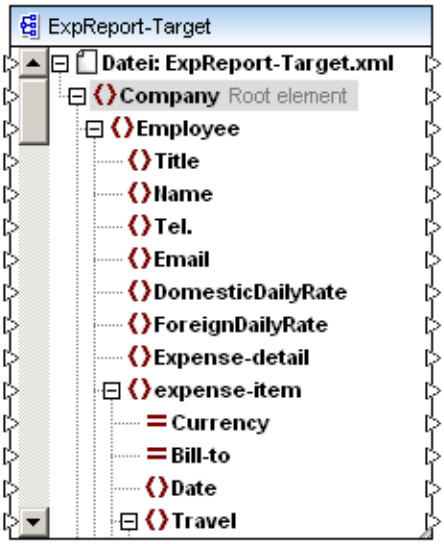
 © 2011 Altova GmbH



Die Zielschemakomponente wird nun auf dem Mapping-Register angezeigt.

4. Klicken Sie auf den Eintrag **Company** und drücken Sie die ***-**Taste auf dem numerischen Tastenblock, um alle Datenelemente anzuzeigen.
5. Doppelklicken Sie auf das Kicken Sie auf das Symbol zum Anpassen der Größe und passen Sie die Fenstergröße an.





Wir können nun mit dem Mappen der Schema-Elemente vom Quell- auf das Zielschema beginnen.

Anmerkung: Wenn Sie eine Komponente mit der Maus ziehen, werden Hilfslinien zur automatischen Ausrichtung der Komponente angezeigt, um die Komponente besser positionieren zu können.

4.2 Erstellen eines Mappings

Im vorhergehenden Abschnitt haben Sie die Quell- und Zielschemakomponente Ihres Mappings definiert. Wir beginnen nun mit dem Mappen der eigentlichen Daten.

Ziel

Sie lernen, wie man die Quell- und Zielkomponente aufeinander mappt und wie man das Mapping-Ergebnis mit Hilfe von Funktionen und Filtern modifizieren kann.

- Mappen von Schema-Datenelementen über Konnektoren
- Kombinieren von Elementen der Quelldaten mit Hilfe einer concat-Funktion
- Filtern von Quelldaten, um nur bestimmte Spesen in den Zielbericht zu übernehmen

In diesem Abschnitt verwendete Befehle

Idente Sub-Einträge automatisch verbinden: Klicken Sie auf diese Schaltfläche, um die automatische Verbindung identer Sub-Einträge ein- bzw. auszuschalten.

Konstante einfügen: Klicken Sie auf diese Schaltfläche, um eine Konstantenkomponente in das gerade aktive Mapping-Fenster einzufügen.

Filter: Nodes/Zeilen: Klicken Sie auf diese Schaltfläche, um eine Filterkomponente in das gerade aktive Mapping-Fenster einzufügen.

Altova Website: Mappen von Daten - Datenintegration und XML mapping

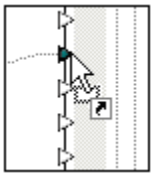
© 2011 Altova GmbH

4.2.1 Mappen von Schema-Elementen

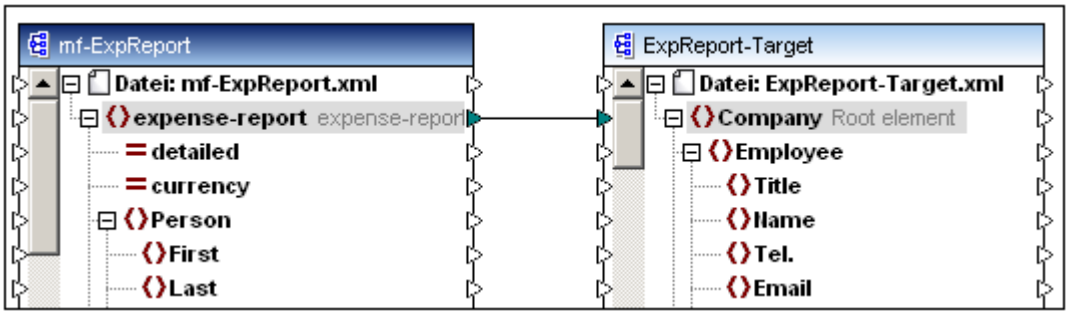
In diesem Abschnitt wird erklärt, wie das Mapping zwischen den Elementen des Quell- und des Zielschemas definiert wird.

So mappen Sie die Schemas mf-ExpReport und ExpReport-Target:

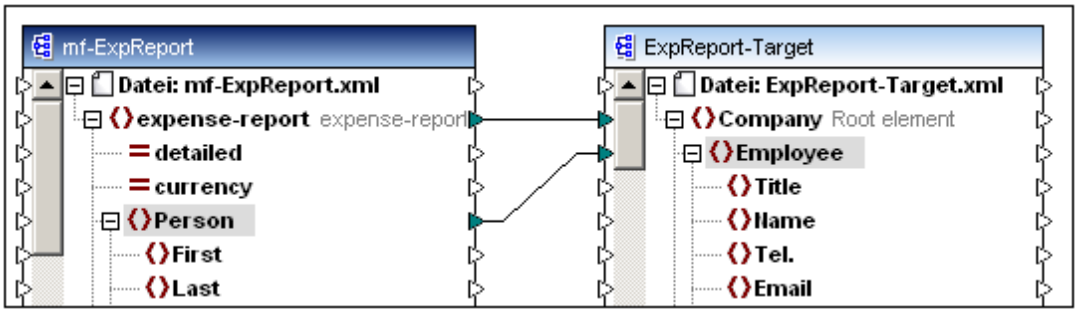
1. Klicken Sie im Quellschema mf-ExpReport auf das "Element" **expense-report** und ziehen Sie es mit der Maus.
   Vom Output-Symbol wird automatisch eine Verbindungslinie zum Mauszeiger, dessen Symbol sich nun geändert hat, erstellt.
2. Ziehen Sie den Mauszeiger in die Nähe des "Elements" **Company** im Schema ExpReport-Target und lassen Sie die Maustaste los, sobald als Mauszeiger wieder ein Pfeilsymbol angezeigt wird. Unterhalb des Mauszeigers wird ein kleines Verbindungssymbol angezeigt. Wenn diese Aktion zulässig ist, erscheint das Input-Symbol und der Name des Datenelements in der Komponente markiert.



Zwischen Quell- und Zielschema ist nun ein Konnektor gesetzt und ein Mapping erzeugt worden.

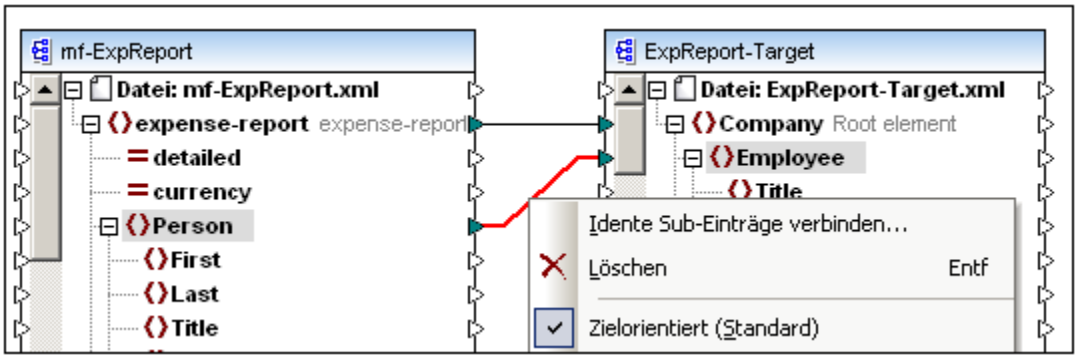


3. Mappen Sie nun die Elemente Person und Employee auf die gleiche Art und Weise.

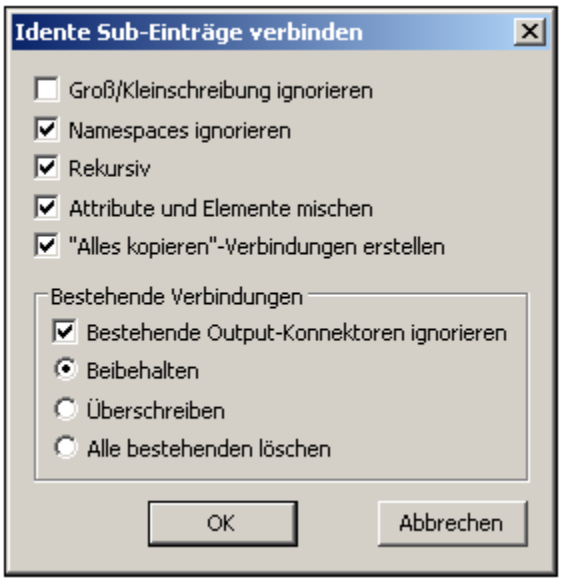


Wenn das Symbol "**Sub-Einträge automatisch verbinden**" aktiv ist, werden die Datenelemente "Title" und "Email" ebenfalls automatisch verbunden. Falls dies nicht der Fall ist:

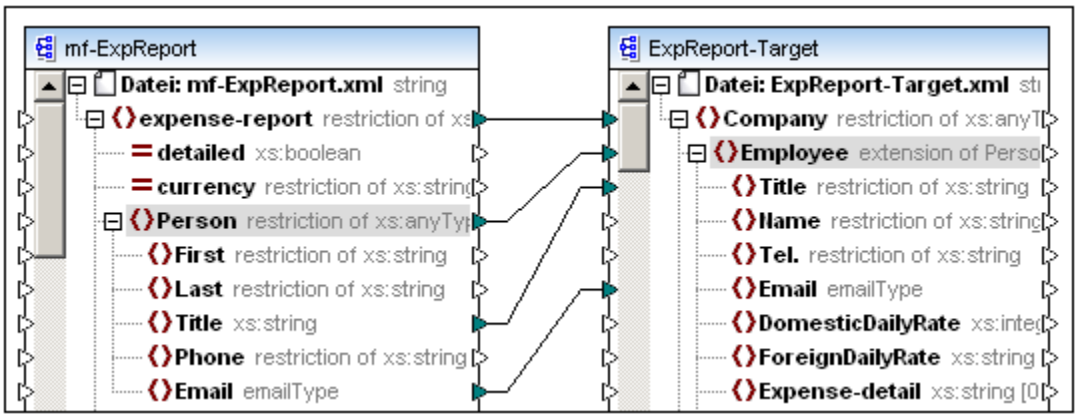
4. Rechtsklicken Sie auf den "Person"-Konnektor" und wählen Sie im Kontextmenü die Option "**Idente Sub-Einträge verbinden**".



Daraufhin wird das Dialogfeld **Idente Sub-Einträge verbinden geöffnet**.



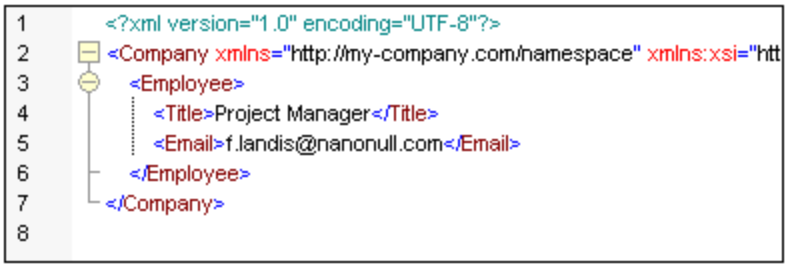
5. Aktivieren Sie die Kontrollkästchen wie in der Abbildung oben und klicken Sie auf OK. Nähere Informationen finden Sie im Abschnitt zu Konnektoreigenschaften.



Für die "Elemente" **Title** und **Email** in den beiden Schemas wurden automatisch Mappings erzeugt.

6. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis im Ausgabefenster zu sehen.

 © 2011 Altova GmbH



Sie werden feststellen, dass die Felder Title und Email Daten enthalten, die aus dem XML-Instanzdokument stammen.

7. Klicken Sie auf die Schaltfläche "Mapping", um wieder in das Mapping-Fenster zurückzuwechseln und mit dem Mapping fortzufahren.

Bitte beachten Sie: Die Einstellungen im Dialogfeld "Idente Sub-Einträge verbinden" werden so lange beibehalten, bis Sie sie ändern. Sie können diese Einstellungen auf eine Verbindung anwenden, indem Sie entweder das Kontextmenü verwenden, oder auf das Symbol "Idente Sub-Einträge automatisch verbinden" klicken, um diese Option zu aktivieren bzw. zu deaktivieren.

4.2.2 Mappen von Daten mit Hilfe von Funktionen

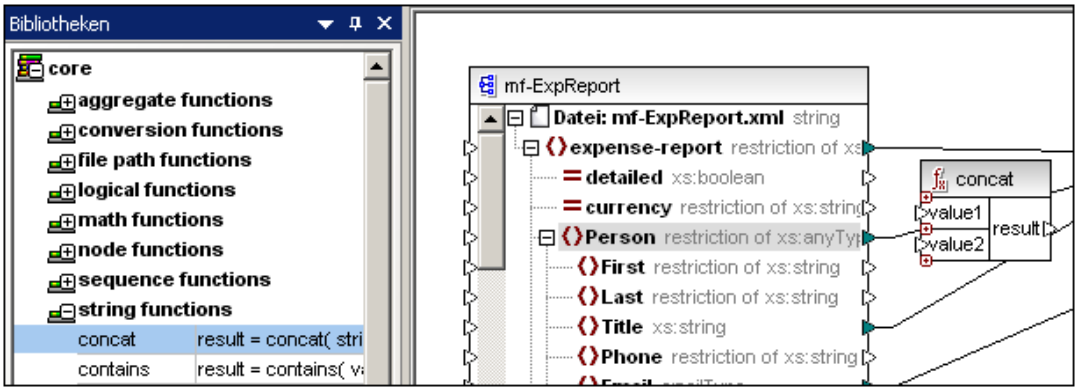
Ziel dieses Abschnitts ist es, zwei Daten-Sets aus dem Quellschema zu verbinden und das Ergebnis im Zieldokument in einem einzigen Datenelement zusammenzufassen. Dies wird in den folgenden Schritten durchgeführt:

- Einsatz der String-Funktion "**concat**", um die Quellschema-Elemente **First** und **Last** zu kombinieren
- Verwendung einer **Konstantenfunktion**, um zur Trennung der beiden Elemente ein Leerzeichen einzufügen
- Setzen des Ergebnisses dieses Vorgangs in das Zielschema-Element **Name.**

Beachten Sie bitte, dass einige der zuvor definierten Mappings aus Gründen der Übersichtlichkeit in den folgenden Abbildungen nicht zu sehen sind.

Kombinieren von Datenelementen mit Hilfe von Funktionen:

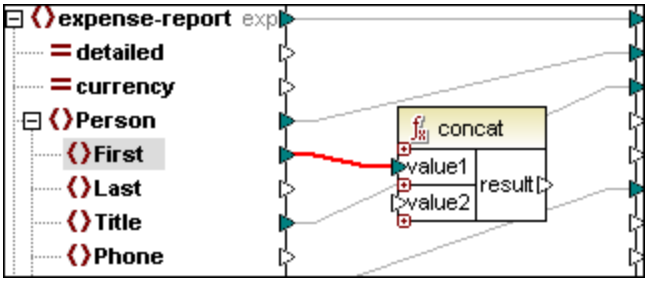
1. Erweitern Sie auf dem Register "Bibliotheken" unter **core** die Funktionsgruppe **string functions,** klicken Sie auf den Eintrag **concat** und ziehen Sie ihn in das Mapping-Fenster.



.

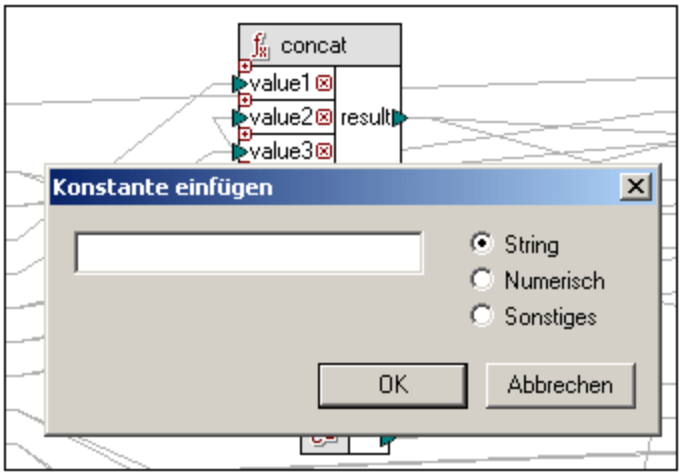
XSLT ausgewählt

2. Wählen Sie in der Komponente "mf-ExpReport" das Datenelement **First** aus, halten Sie die Maustaste gedrückt und ziehen Sie den Cursor auf den Input **value1** der concat-Komponente, um eine Verbindung herzustellen.

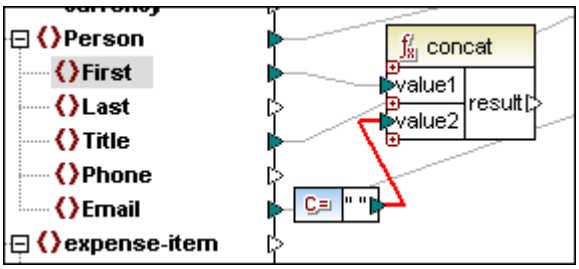


3. Klicken Sie in der Symbolleiste mit der rechten Maustaste auf den Hintergrund in der Nähe von **value2** und wählen Sie im Kontextmenü den Befehl **Konstante einfügen**, um eine Konstantenkomponente einzufügen.

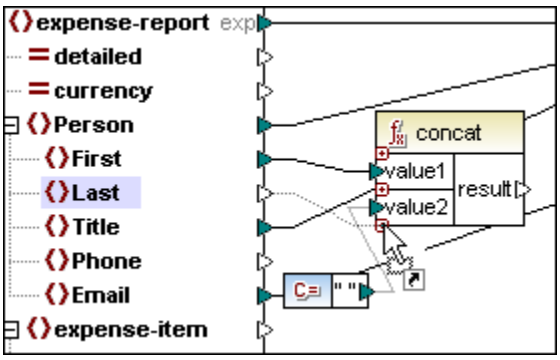
© 2011 Altova GmbH



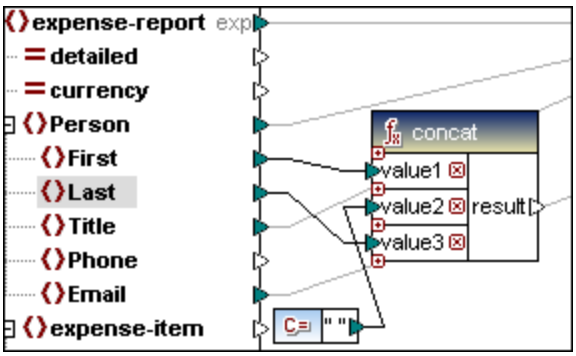
4. Geben Sie im Textfeld ein Leerzeichen ein und klicken Sie auf **OK**.
   Die Konstantenkomponente befindet sich nun im Arbeitsbereich und ihr Inhalt wird neben dem Output-Symbol angezeigt.
5. Erstellen Sie eine Verbindung zwischen der Komponente **constant** und **value2** der concat-Komponente.



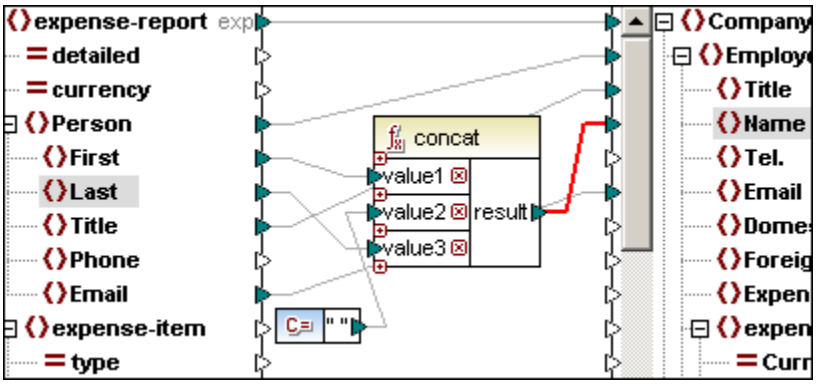
6. Klicken Sie in der Komponente "mf-ExpReport" auf das Datenelement **Last** und ziehen Sie den Konnektor auf das "**+**"-Symbol der concat-Funktion unterhalb von **value2**. Der Mauszeiger ändert sich und zeigt an, an welcher Stelle Sie den Konnektor platzieren können.



Dadurch wird die concat-Funktion automatisch um ein weiteres Datenelement (Wert) erweitert, mit dem das Datenelement "Last" verbunden wird.



7. Verbinden Sie das **result**-Symbol der concat-Komponente mit dem Datenelement **Name** im Zielschema.



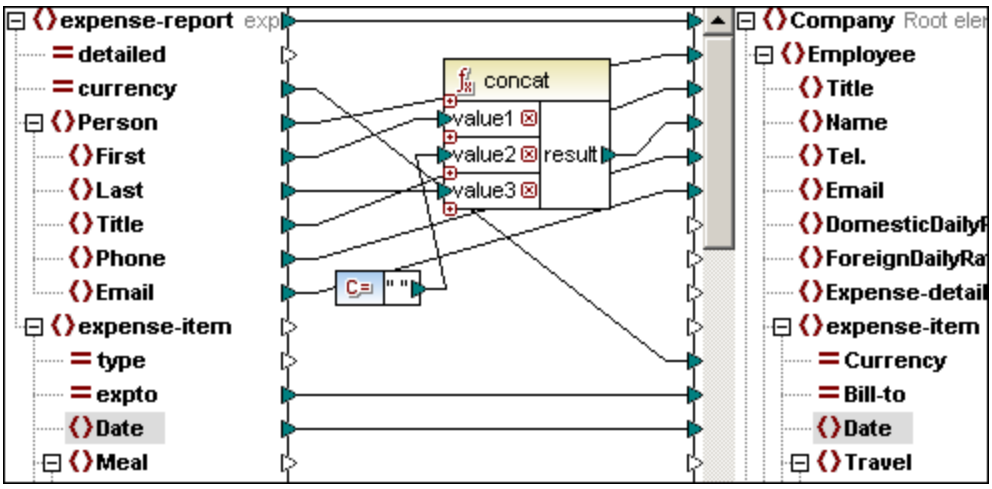
8. Klicken Sie auf die Schaltfläche **Ausgabe**, um das Ergebnis des aktuellen Mappings im Ausgabefenster zu sehen.

Sie werden sehen, dass sich zwischen den **Name-**Tags nun der Personennamen "Fred Landis" befindet. Vor- und Nachname sind durch ein Leerzeichen getrennt.

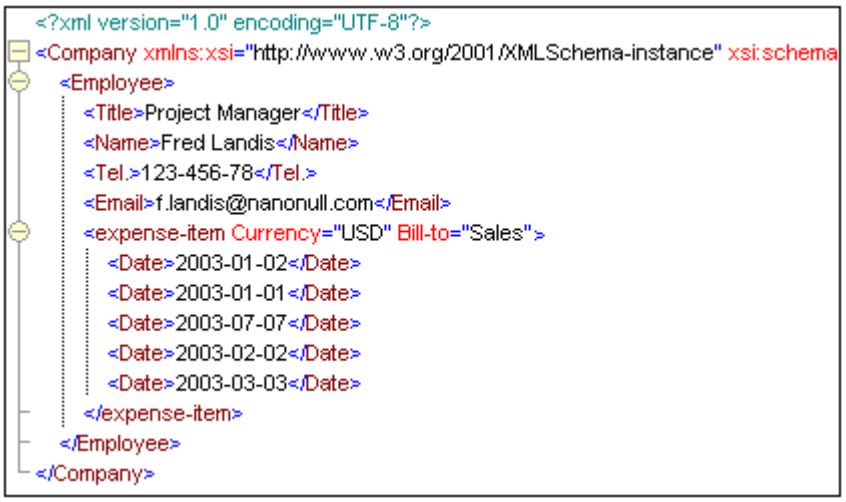
Mappen der übrigen Personendaten:

1. Erzeugen Sie zwischen den folgenden Datenelementen Mappings:
   - currency auf currency
   - Phone auf Tel.
   - expto auf Bill-to
   - Date auf Date

 © 2011 Altova GmbH



2. Kicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis zu sehen.



Im Moment stammen fünf Datenelemente aus der zugewiesenen XML-Instanzdatei.

Bitte beachten Sie:
Um Platz auf dem Bildschirm zu sparen können Funktionen in benutzerdefinierten Funktionen/Komponenten gruppiert werden. Ein Beispiel dafür, wie man die concat- und die constant-Funktion zu einer einzigen benutzerdefinierten Funktion/Komponente kombiniert finden Sie unter "Benutzerdefinierte Funktionen/Komponenten".

4.2.3 Filtern von Daten

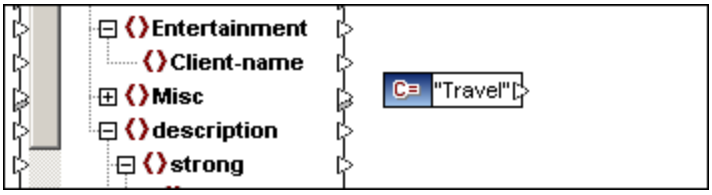
Im diesem Abschnitt sollen die Ausgaben für Unterkunft und Essen herausgefiltert werden, sodass nur die Reisekosten im Zielschema/-dokument übernommen werden.

Dies wird bewerkstelligt durch:

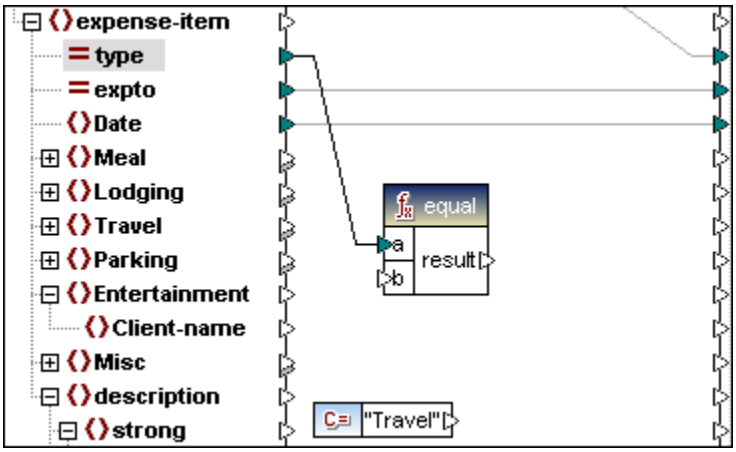
- Verwendung der **equal-**Funktion zum Testen des Werts eines Quelldatenelements
- Verwendung einer **Konstantenfunktion**, um den zu testenden Vergleichsstring anzugeben
- Verwendung der **Filter-**Komponente, die die Reisekostendaten weitergibt, bei denen der Boolesche Eingabewert True ergibt
- Setzen des on-true-Ergebnisses in das Zielschema/-dokument-Element **expense-item.**

Filtern von Daten:

1. Klicken Sie auf die Schaltfläche **Konstante einfügen** um eine Konstantenkomponente einzufügen und geben Sie in das Eingabefeld den String **Travel** ein.

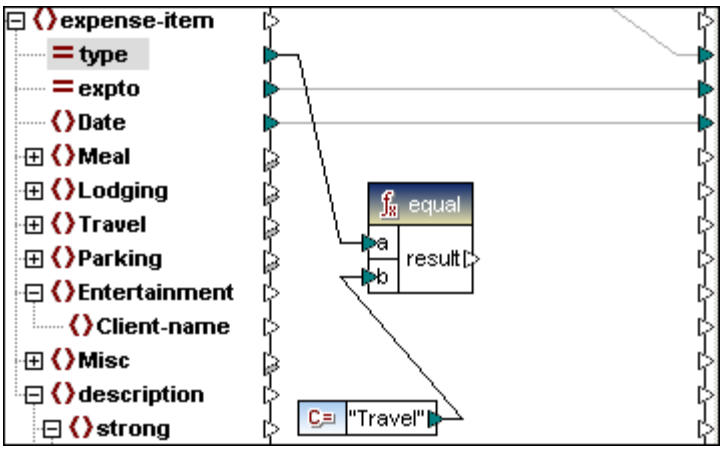


2. Erweitern Sie auf dem Register "Bibliotheken" in der **core**-Bibliothek die Gruppe **logical functions** und ziehen Sie die logische Funktion **equal** in das Mapping Fenster.
3. Verbinden Sie das expense-item-Attribut **type** im Quellschema mit dem Parameter **a** der equal-Funktion.

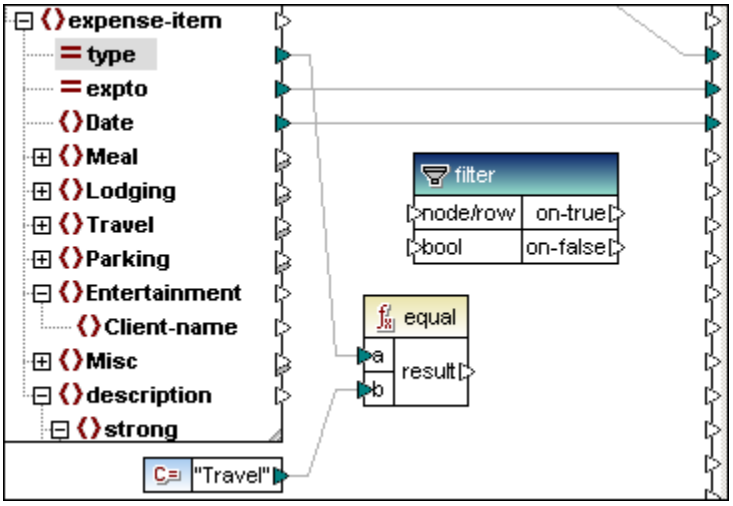


4. Verbinden Sie das **result-**Symbol der Travel-Konstantenkomponente mit dem Parameter **b** der equal-Funktion.

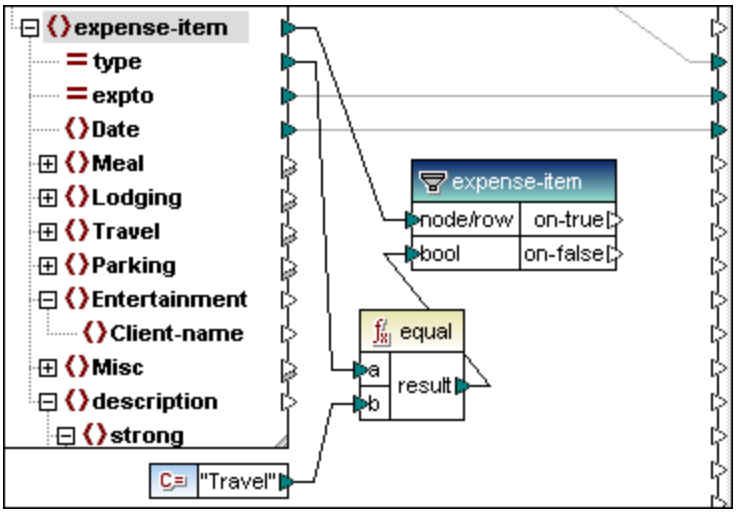
© 2011 Altova GmbH



5. Wählen Sie die Menüoption **Einfügen | Filter: Nodes/Zeilen**.

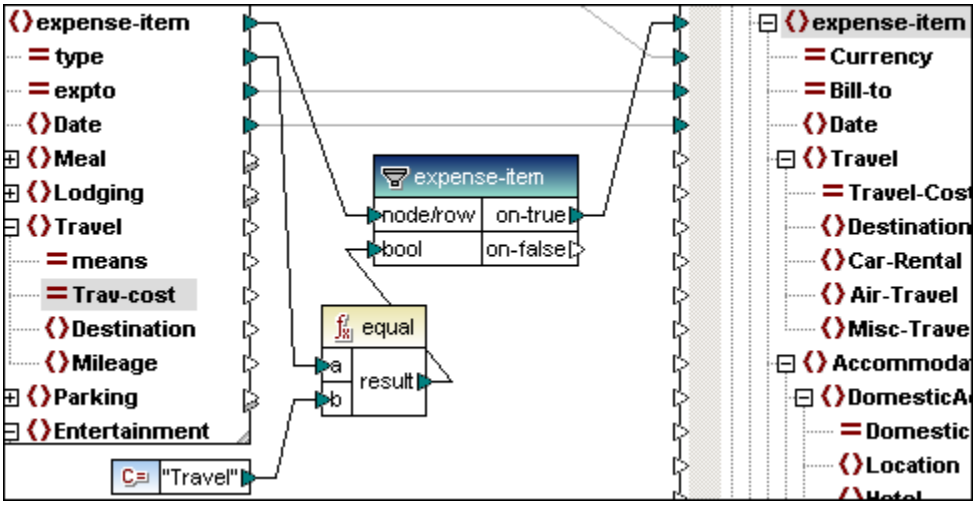


6. Verbinden Sie das **result**-Symbol der **equal**-Komponente mit dem **Bool-**Parameter der **Filter**-Komponente
7. Verbinden Sie das **expense-item**-Symbol des Quellschemas mit dem **node/row** -Parameter der Filterkomponente.

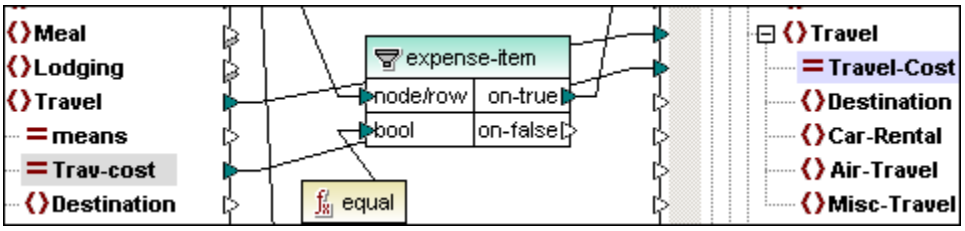


Beachten Sie dass sich der Name der Filterkomponente nun in "expense-item" ändert.

8. Verbinden Sie das **on-true**-Symbol der **Filter-**Komponente mit dem **expense-item-**Element des Zieldokuments.

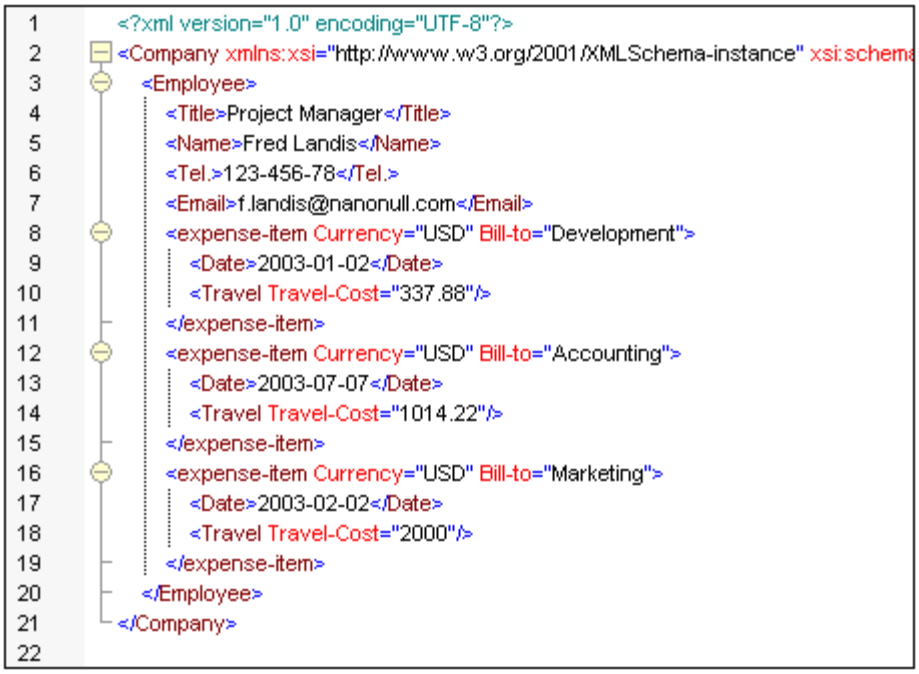


9. Verbinden Sie das Element **Travel** im Quellschema mit dem Element **Travel** im Zielschema/-dokument.
10. Verbinden Sie das Attribut **Trav-cost** mit dem Attribut **Travel-Cost** im Zielschema/-dokument.



11. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis im Ausgabefenster zu sehen.

© 2011 Altova GmbH



```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Company xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schema
3    <Employee>
4      <Title>Project Manager</Title>
5      <Name>Fred Landis</Name>
6      <Tel.>123-456-78</Tel.>
7      <Email>f.landis@nanonull.com</Email>
8      <expense-item Currency="USD" Bill-to="Development">
9        <Date>2003-01-02</Date>
10       <Travel Travel-Cost="337.88"/>
11     </expense-item>
12     <expense-item Currency="USD" Bill-to="Accounting">
13       <Date>2003-07-07</Date>
14       <Travel Travel-Cost="1014.22"/>
15     </expense-item>
16     <expense-item Currency="USD" Bill-to="Marketing">
17       <Date>2003-02-02</Date>
18       <Travel Travel-Cost="2000"/>
19     </expense-item>
20   </Employee>
21 </Company>
22
```

Bitte beachten Sie: Der **on-false**-Parameter der Filterkomponente gibt das **Komplement-Nodeset** aus, das vom on-true-Parameter gemappt wird. In diesem Beispiel wären das alle **nicht-travel**-Reisekostenposten.

Die Anzahl der Expense-item-Elemente hat sich nun auf drei reduziert. Wenn Sie dies mit der Datei **mf-ExpReport.xml** vergleichen, sehen Sie, dass nur die Reisekosten übrig bleiben, die Unterkunfts- und Essensausgaben wurden herausgefiltert.

4.3 Generieren von XSLT 1.0- und 2.0-Code

Nachdem Sie nun das Mapping erstellt haben, können Sie die eigentliche Transformation der Quelldaten durchführen. MapForce kann zwei Versionen von XSLT-Code generieren: XSLT 1.0 und XSLT 2.0.

Ziel

In diesem Abschnitt des Tutorials lernen Sie, wie man eine Vorschau des XSLT-Codes anzeigt, diesen generiert und speichert und wie man den generierten XSLT-Code ausführt. Insbesondere werden die folgenden Schritte beschrieben:

- Generieren und Speichern von XSLT-Code der gewünschten Art
- Downloaden des Altova XML-Prozessors und Ausführen der Batch-Transformationsdatei

In diesem Abschnitt verwendete Befehle

Datei | Code generieren in: Wählen Sie diese Option, um die Ausgabesprache (d.h. **XSLT 1.0** oder **XSLT 2.0**) auszuwählen. Mit dem Befehl wird das Dialogfeld **Verzeichnis auswählen** geöffnet, wo Sie den Ordner definieren, in dem der generierte XSLT-Code gespeichert werden soll.

So generieren Sie XSLT-Code:

1. Wählen Sie die Menüoption **Datei | Code generieren in | XSLT 1.0 (bzw. XSLT 2.0)**
2. Geben Sie den Ordner an, in dem die generierte XSLT-Datei abgelegt werden soll und bestätigen Sie dies mit OK.
   Es wird eine Meldung angezeigt, dass die Generierung erfolgreich war.
3. Navigieren Sie zum gewünschten Ordner, wo Sie die XSLT-Datei unter dem Namen **MappingMapToExpReport-Target.xslt** finden. (d.h. in der Form: MappingMapTo<TargetSchemaName>).

Transformieren der XML-Datei

Der Ordner, in dem die XSLT-Datei gespeichert wird, enthält auch eine Batch-Datei mit dem Namen **DoTransform.bat,** die die XML-Datei mittels AltovaXML transformiert. AltovaXML kann kostenlos von der Altova Website heruntergeladen werden.

So transformieren Sie die persönliche Reisekostenabrechnung in die Firmen-Spesenabrechnung:

1. Laden Sie den kostenlosen AltovaXML-Prozessor von der AltovaXML Download-Seite herunter und installieren Sie ihn. AltovaXML wird standardmäßig im Verzeichnis c:\Program Files\Altova\AltovaXML\2012\ installiert.
2. Starten Sie die Batch-Datei **DoTransform.bat,** die sich im zuvor angegebenen Ausgabeordner befindet.
   Daraufhin wird im aktuellen Ordner ...\Tutorial die Ausgabedatei **ExpReport-Target.xml** generiert.

Anmerkung: Eventuell müssen Sie den AltovaXML Installationsordner zur Pfadvariablen der Umgebungsvariablen hinzufügen.

© 2011 Altova GmbH

4.4 Arbeiten mit mehreren Zielschemas / -dokumenten

In diesem Abschnitt wird erklärt, wie man ein zweites Zielschema / -dokument erstellt, in das die Datensätze übernommen werden, die **nicht die Reisekosten selbst** betreffen. Wir arbeiten hierbei mit der Tutorial-Beispieldatei **Tut-ExpReport.mfd,** die wir bis jetzt benutzt haben.

Ziel

In diesem Abschnitt des Tutorials erfahren Sie, wie man eine zweite Zielkomponente hinzufügt und mehrere Ausgabeschemas generiert. Dabei werden folgende Schritte beschrieben:

- Erstellen einer zweiten Schemazielkomponente
- Herausfiltern aller nicht reisekostenbezogenen Ausgaben in Ihrem Beispielbericht
- Anzeigen einer bestimmten Ausgabe
- Generieren von XSLT-Code für mehrere Zielschemas

In diesem Abschnitt verwendete Befehle

XML Schema/Datei einfügen: Klicken Sie auf diese Schaltfläche, um das Windows-Standarddialogfeld "Öffnen" aufzurufen und die Datei aus Ihrem Dateisystem auszuwählen.

Vorschau: Wird in der Titelleiste von Komponenten angezeigt, wenn mehrere Zieldateien definiert wurden. Klicken Sie auf dieses Symbol, um eine bestimmte Komponente für die Ausgabevorschau auszuwählen.

Generierte Ausgabe speichern: Befindet sich im Menü/Fenster **Ausgabe**. Klicken Sie auf diese Schaltfläche, um das Windows-Standarddialogfeld **Speichern unter** aufzurufen und den Ordner auszuwählen, in dem die generierten Ausgabedaten gespeichert werden sollen.

Ausgabe validieren: Befindet sich im Menü/Fenster **Ausgabe**. Klicken Sie auf diese Schaltfläche, um zu überprüfen, ob die generierte Ausgabe gültig ist. Das Ergebnis der Validierung wird im Fenster "Meldungen" angezeigt.

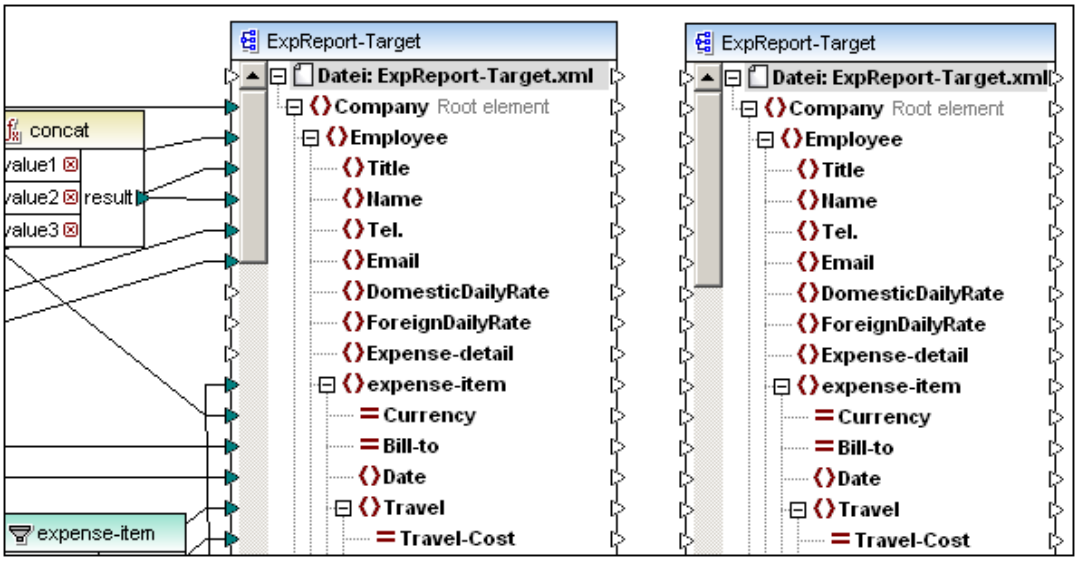
Datei | Code generieren in: Wählen Sie diese Option, um die Ausgabesprache (d.h. **XSLT 1.0** oder **XSLT 2.0**) auszuwählen. Mit dem Befehl wird das Dialogfeld **Verzeichnis auswählen** geöffnet, wo Sie den Ordner definieren, in dem der generierte XSLT-Code gespeichert werden soll.

4.4.1 Erstellen einer zweiten Zielschemakomponente

In diesem Abschnitt des Tutorials wird erklärt, wie man eine zweite Zielschemakomponente erstellt, in der alle nicht reisekostenbezogenen Daten herausgefiltert werden.

Erzeugen einer zweiten Zielschemakomponente:

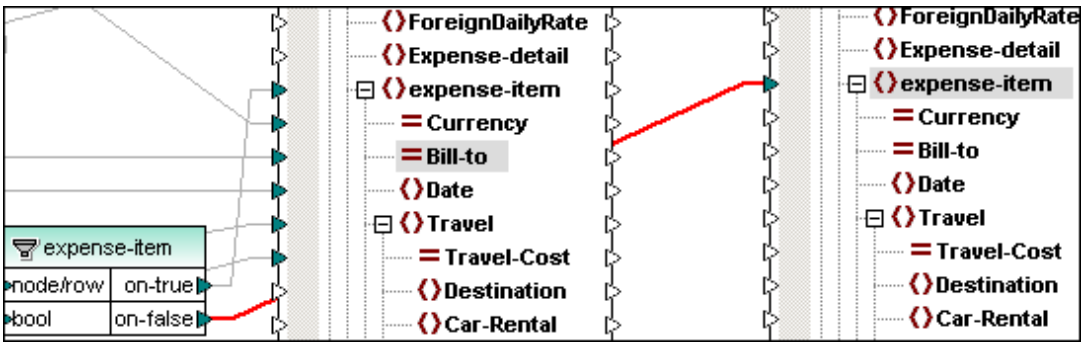
1. Klicken Sie auf das Symbol **XML-Schema / Datei einfügen**.
2. Wählen Sie im Dialogfeld "Öffnen" die Datei **ExpReport-Target.xsd**.
   Sie werden nun aufgefordert, eine XML-Beispieldatei für dieses Schema anzugeben.
3. Klicken Sie auf **Überspringen** und wählen Sie **Company** als Root-Element des Zieldokuments aus.
   Die Zielschemakomponente wird nun im Mapping-Fenster angezeigt.
4. Klicken Sie auf den Eintrag **Company** und drücken Sie auf der Zahlentastatur die ***-** Taste, um alle Datenelemente anzuzeigen.
5. Klicken Sie auf das Symbol zum Vergrößern des Fensters und passen Sie die Größe der Komponente an. Platzieren Sie die Schemakomponenten so, dass Sie gut zu sehen sind, damit sie optimal damit arbeiten können.
   Wir haben nun ein Quellschema **mf-expReport** und zwei Zielschemas, beide mit dem Namen **ExpReport-Target,** die im Mapping-Fenster zu sehen sind.



Herausfiltern von Daten, die nicht die Reiskosten selbst betreffen:

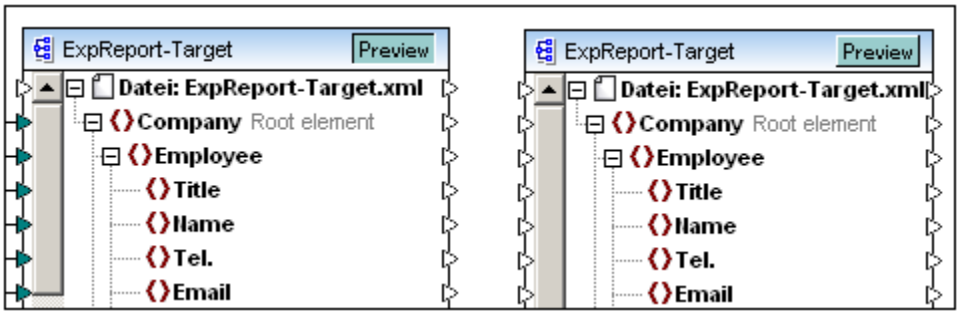
1. Verbinden Sie das **on-false**-Symbol der **filter**-Komponente mit dem Element **expense-item** des **zweiten** Zielschemas / -dokuments.

 © 2011 Altova GmbH



Daraufhin wird eine Meldung eingeblendet, dass Sie nun mit mehreren Zielschemas / -dokumenten arbeiten.

2. Bestätigen Sie dies mit **OK**.



In der Titelleiste der einzelnen Zielschemakomponenten wird nun ein **Vorschausymbol** angezeigt.

Durch Klicken auf dieses Symbol definieren Sie, welche der Zielschemadaten angezeigt werden sollen, wenn Sie anschließend auf die Schaltfläche XSLT, XSLT2, oder Ausgabe klicken.

Erzeugen von Mappings für die übrigen Daten der Reisekostenabrechnung:
Erstellen Sie die folgenden Mappings **zwischen der Schemaquell- und Zielkomponente**. Sie haben die gleichen Konnektoren für das erste Zielschema erstellt, gehen Sie daher vor wie vorhin:

- `Person auf Employee`
- `Tittle auf Title`
- `Phone auf Tel.`
- `Email auf Email`
- `currency auf Currency`
- `expto auf Bill-to`
- `Date auf Date`

Erstellen Sie das folgende Mapping zwischen der vorhandenen **concat-Funktion und dem zweiten Zielschema**:

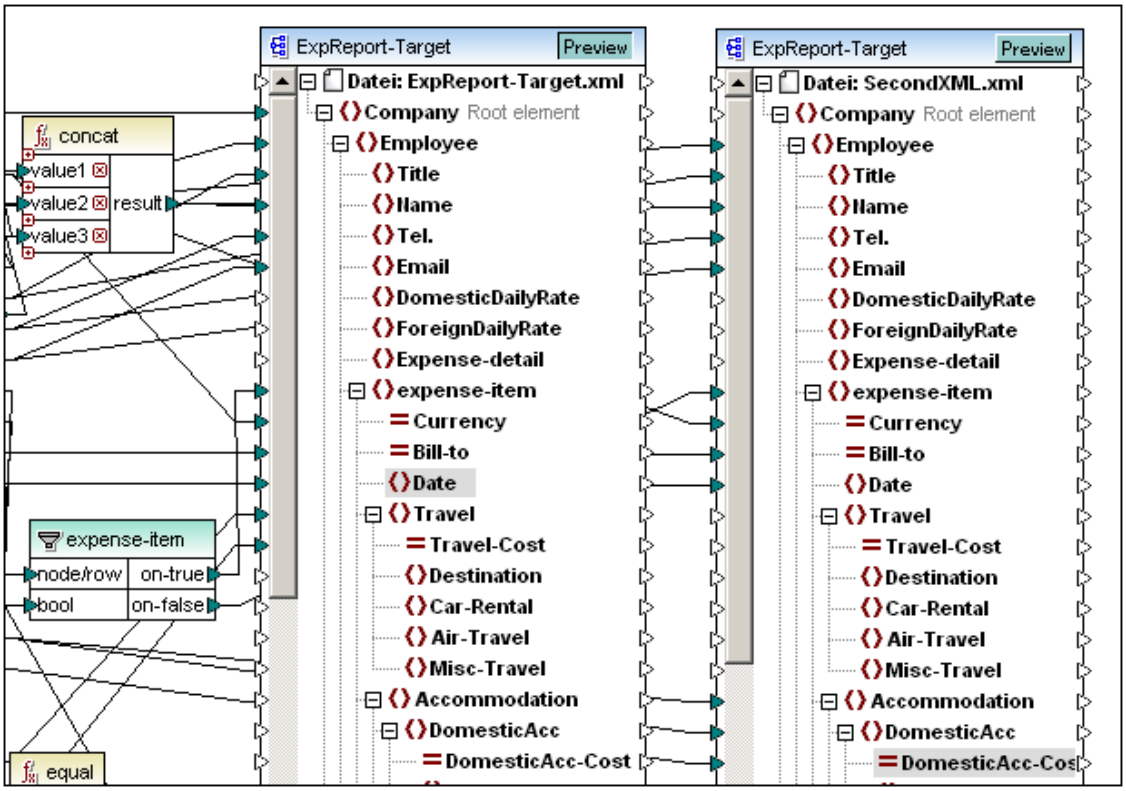
- `result auf Name`

So erstellen sie die restlichen nicht reisekostenbezogenen Mappings:
Stellen Sie sicher, dass die Option "Idente Sub-Einträge automatisch verbinden" deaktiviert ist und

1. Verbinden Sie das Datenelement **Lodging** im Quellschema mit **Accommodation** im

zweiten Zielschema.

2. Verbinden Sie das Element **Lodging** mit **DomesticAcc**.
3. Verbinden Sie das Datenelement **Lodge-Cost** mit **DomesticAcc-Cost**.



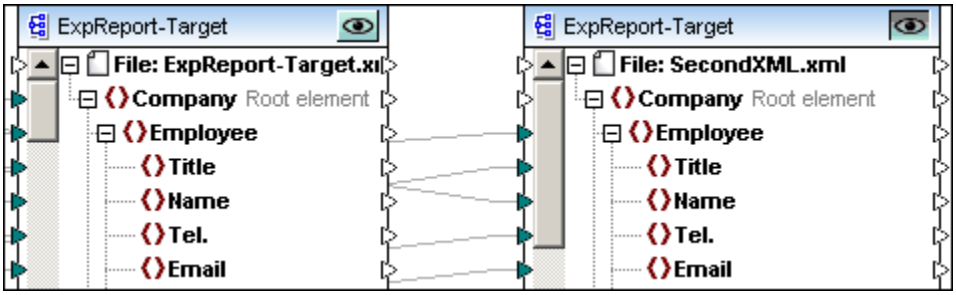
 © 2011 Altova GmbH

4.4.2 Anzeigen und Generieren mehrerer Zielschema-Ausgaben

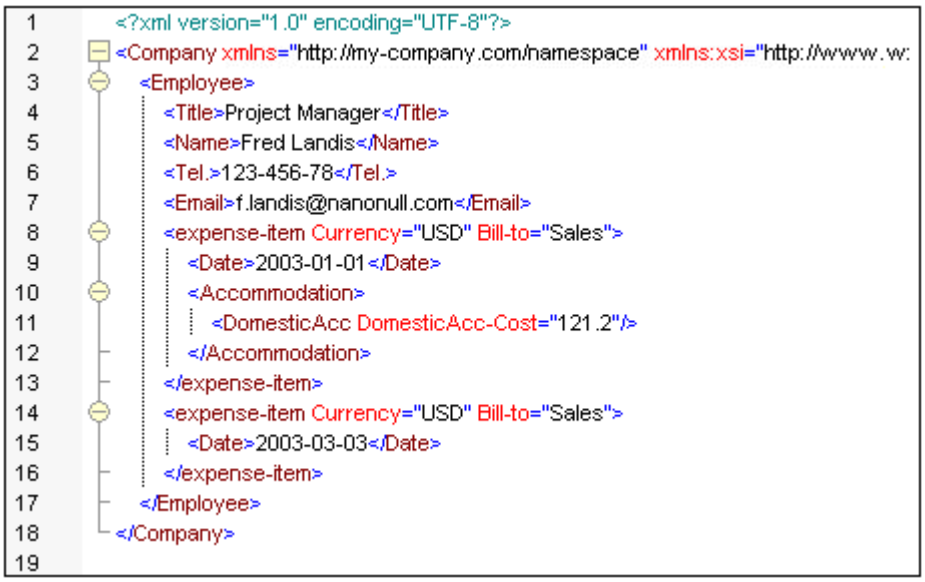
Wenn Sie auf das Preview-Symbol klicken, können Sie auswählen, welches der Ziele angezeigt werden soll.

So zeigen Sie eine bestimmte XSLT-Ausgabe an:

1. Klicken Sie in der Titelleiste der **zweiten** Schemakomponente auf das **Vorschausymbol** , um es aktiv zu setzen (falls es noch nicht aktiv ist).



2. Klicken Sie im Mapping-Fenster auf die Schaltfläche **Ausgabe**.



Die XML-Ausgabe enthält zwei Datensätze, die beide für den Bereich Sales verrechnet werden: Die Kosten für "Domestic Accommodation" in der Höhe von $121,2 und einen Expense-item-Datensatz, der nur ein Datum enthält. Dieser Datensatz stammt aus dem expense-item Meal. Gegenwärtig gibt es kein Mapping zwischen Meal costs und Domestic accommodation und selbst wenn eines vorhanden wäre, würden keine Kosten aufscheinen, da die XML-Instanz keine enthält.

Bitte beachten Sie: Sie können diese XML-Daten speichern. Klicken Sie dazu im Vorschau-Fenster für die Anzeige der XML-Ausgabe auf das Symbol **Ausgabedatei speichern** .

Sie können die erzeugte XML-Instanzdatei auch gegen das Zielschema validieren, wenn Sie auf die Schaltfläche "Validieren" klicken .

So generieren Sie XSLT 1.0/XSLT 2.0-Code für mehrere Zielschemas:

1. Wählen Sie die Menüoption **Datei | Code generieren in | XSLT oder XSLT2.**.

2. Wählen Sie den Ordner, in dem die erzeugten XSLT-Dateien gespeichert werden sollen, und klicken Sie auf OK.
   Es wird eine Meldung angezeigt, dass die Generierung erfolgreich war.
3. Navigieren Sie zum gewünschten Ordner, in dem sich nun zwei XSLT-Dateien mit den Namen **MappingExpReport-Target.xslt** und **MappingExpReport-Target2.xslt** befinden.

So transformieren Sie den persönlichen Ausgabebericht in den Firmen-Ausgabenbericht:

1. Laden Sie den kostenlosen AltovaXML Prozessor von der AltovaXML-Download-Seite herunter und installieren Sie ihn. AltovaXML wird standardmäßig im Ordner c:\Program Files\Altova\AltovaXML2012\ installiert.
2. Starten Sie die Batch-Datei **DoTransform.bat,** die sich im zuvor angegebenen Ausgabeordner befindet.
   Daraufhin wird im Ordner ...\Tutorial die Ausgabedatei **ExpReport-Target.xml** generiert.

   Hinweis: Eventuell müssen Sie die Installationspfad von AltovaXML zur Pfadvariablen der Umgebungsvariablen hinzufügen.

© 2011 Altova GmbH

4.5 Mappen mehrerer Quellelemente auf ein einziges Zielelement

In diesem Abschnitt werden wir zwei einfache Mitarbeiter-Reisekostenabrechnungen auf eine einzige Firmen-Spesenabrechnung mappen. Bei diesem Beispiel handelt es sich um eine vereinfachte Version des Mappings, das Sie im Tutorial bereits im Abschnitt über das Mappen mehrerer Zielschemas / - dokumente durchgeführt haben.

Bitte beachten Sie: Es gibt eine alternative Methode dazu, dies mit der Funktion für dynamischen Input/Output von Komponenten zu tun. Ein Beispiel dazu finden Sie unter " Dynamische Dateinamen - Input / Output.

Ziel
In diesem Abschnitt des Tutorials lernen Sie, wie man zwei **persönlichen Reisekostenabrechnungen** in einer Firmen-Spesenabrechnung zusammenführt. Dabei werden folgende Schritte beschrieben:

- Mappen von Schemakomponenten (Wiederholung)
- Duplizieren von Input-Datenelementen
- Entfernen von duplizierten Datenelementen

In diesem Abschnitt verwendete Befehle

Neu...: Dieser Befehl befindet sich im Menü **Datei** und in der Standard-Symbolleiste. Klicken Sie auf diese Schaltfläche, um das Dialogfeld **Neue Datei** aufzurufen, in dem Sie ein neues Mapping erstellen können.

XML Schema/Datei einfügen: Klicken Sie auf diese Schaltfläche, um das Windows-Standarddialogfeld "Öffnen" aufzurufen und die Datei aus Ihrem Dateisystem auszuwählen.

Idente Sub-Einträge automatisch verbinden: Klicken Sie auf diese Schaltfläche, um die automatische Verbindung identer Sub-Einträge ein- bzw. auszuschalten.

Duplik davor/danach einfügen: Befindet sich im Kontextmenü, das angezeigt wird, wenn Sie mit der rechten Maustaste auf ein Datenelement in einer Komponente klicken. Klicken Sie auf diesen Befehl, um das ausgewählte Datenelement zu duplizieren.

Duplikat löschen: Befindet sich im Kontextmenü, das angezeigt wird, wenn Sie mit der rechten Maustaste auf ein Datenelement in einer Komponente klicken. Klicken Sie auf diesen Befehl, um das ausgewählte Datenelement aus der Komponente zu löschenb.

In diesem Abschnitt verwendete Dateien

Die für dieses Beispiel verwendeten Dateien wurden speziell adaptiert, um zu demonstrieren, wie Daten aus zwei XML-Eingabedateien auf ein einziges Zieldatenelement im Zielschema gemappt werden können. Dieses Beispiel ist kein reales Szenario.

mf-ExpReport.xml	die im vorigen Abschnitt verwendete XML-Eingabedatei
mf-ExpReport2.xml	die zweite XML-Eingabedatei
mf-ExpReport-combined.xml	die nach erfolgtem Mapping erzeugte Datei
ExpReport-combined.xsd	die Zielschemadatei, in der die beiden XML-Quelldateien zusammengeführt werden sollen

Tut-ExpReport-msource.mfd	die Mapping-Datei für dieses Beispiel

Bitte beachten Sie: Die in diesem Abschnitt verwendeten Dateien liegen ebenfalls im Ordner ...\MapForceExamples\Tutorial\.

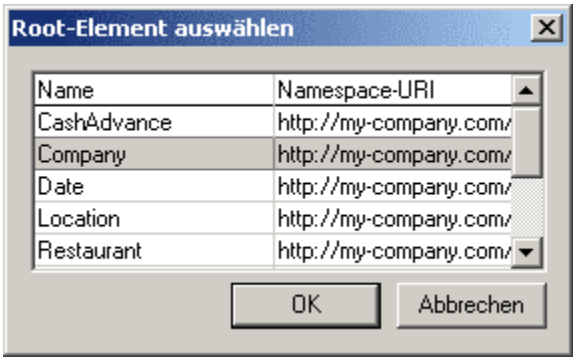
© 2011 Altova GmbH

4.5.1 Erzeugen der Mappings

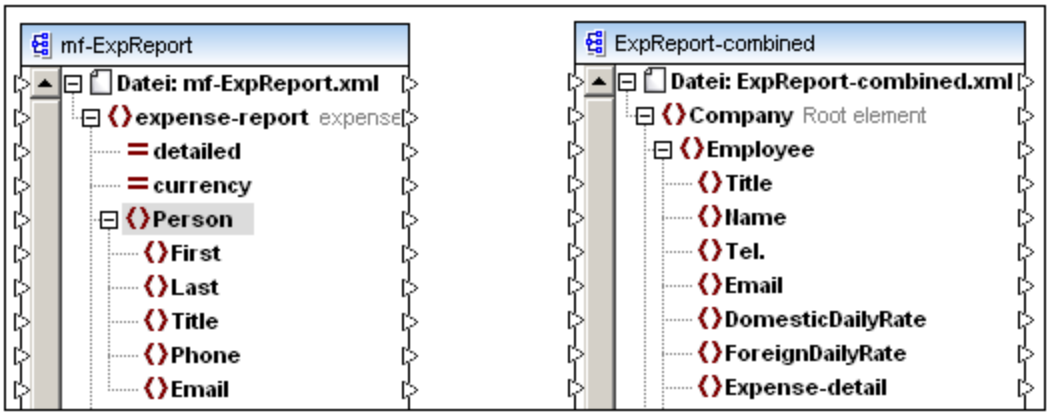
In der unten stehenden Anleitung wird rekapituliert, wie man eine Mapping-Umgebung einrichtet. Dieses Mapping finden Sie im Ordner ...\MapForceExamples\Tutorial\ unter dem Namen **Tut-ExpReport-msource.mfd.**

So erstellen Sie die Mapping-Umgebung:

1. Klicken Sie in der Standard-Symbolleiste auf die Schaltfläche **Neu** um das Dialogfeld **Neue Datei** zu öffnen.
2. Klicken Sie auf das Mapping-Symbol und anschließend auf **OK**, um ein neues Mapping-Register aufruzufen.
3. Klicken Sie auf die Schaltfläche **XML-Schema/Datei einfügen**.
4. Wählen Sie im Dialogfeld "Öffnen" die Datei **mf-ExpReport.xsd** aus dem Unterordner "Tutorial" im Verzeichnis MapForceExamples, klicken Sie auf **Durchsuchen...**und wählen Sie die Datei **mf-ExpReport.xml** als XML-Instanzdatei, wenn Sie danach gefragt werden.
5. Klicken Sie auf den Eintrag **expense-report**, drücken Sie auf der Zahlentastatur die ***-**Taste, um alle Datenelemente einzublenden; passen Sie ggf. die Größe der Komponente an.
6. Klicken Sie auf das Symbol **XML-Schema / Datei einfügen**.
7. Wählen Sie im Dialogfeld "Öffnen" die Datei **ExpReport-combined.xsd**.
   Sie werden nun aufgefordert, eine XML-Beispieldatei für dieses Schema anzugeben.
8. Klicken Sie auf "Überspringen" und wählen Sie **Company** als Root-Element des Zieldokuments.



   Die Zielschema-Komponente wird nun im Mapping-Fenster angezeigt.
9. Klicken Sie auf den Eintrag **Company**, drücken Sie die *-Taste, um alle Elemente einzublenden und passen Sie ggf. die Fenstergröße an.

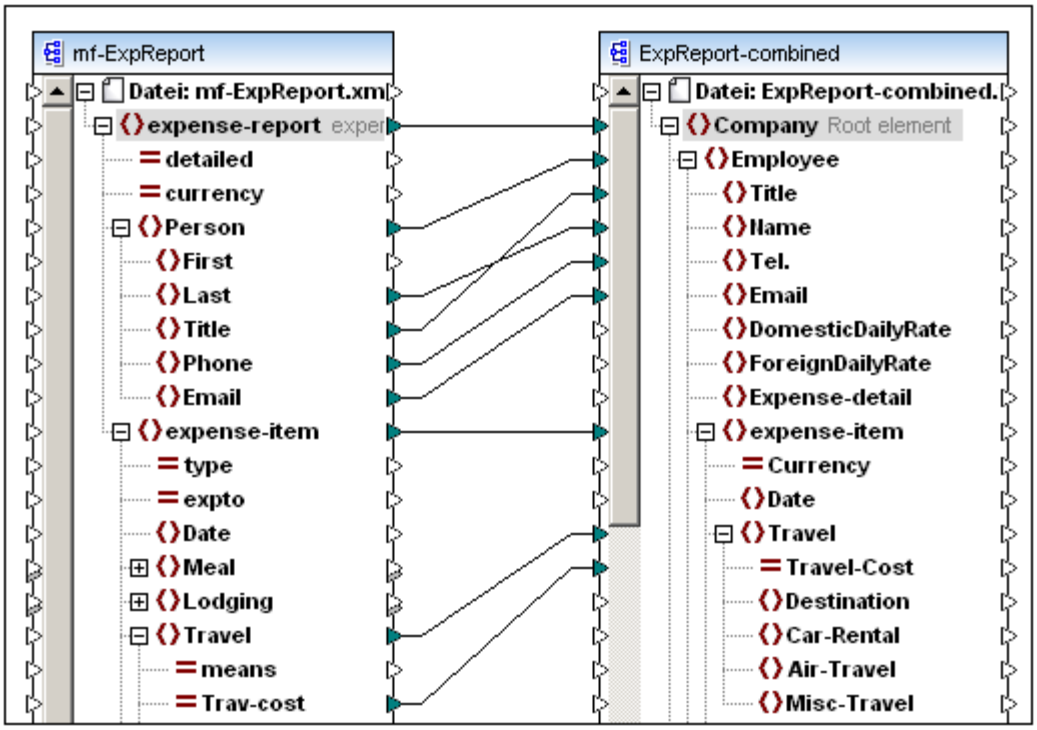


Mappen der Komponenten

Vergewissern Sie sich, dass das Symbol "Idente Sub-Einträge automatisch verbinden" **deaktiviert** ist, bevor Sie die folgenden Mappings erzeugen.

- Expense-report auf Company
- Person auf Employee
- Last auf Name
- Title auf Title
- Phone auf Tel.
- Email auf Email
- expense-item auf expense-item
- Travel auf Travel und
- Trav-cost auf Travel-Cost

Das Mapping wird unten stehend angezeigt.



Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis des aktuellen Mappings

 © 2011 Altova GmbH

anzuzeigen.

Bitte beachten Sie: Wenn ein **gemapptes Parent-Element** in der Quelldatei Child-Elemente enthält, die nicht auf das Zielschema gemappt wurden, werden leere <expense-item/> Elemente/Tags generiert. In diesem Fall wurden nur die Travel-Elemente des übergeordneten Elements expense-item gemappt. Die Liste enthält jedoch noch zwei weitere expense-Elemente: ein lodging-Element und ein meal-Element. Für jedes dieser Elemente wird ein leerer übergeordneter expense-item Tag erzeugt.

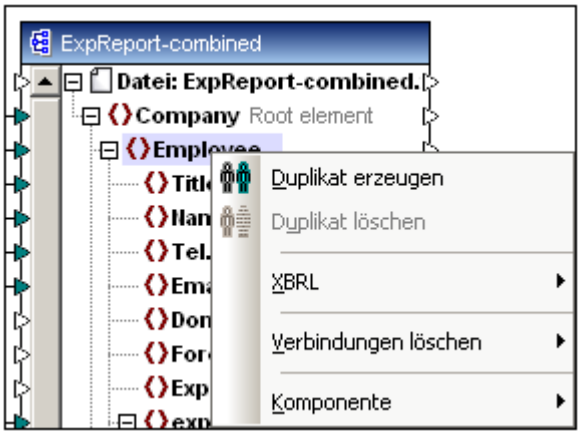
Um keine leeren Tags zu generieren, erstellen Sie einen Filter, wie zuvor im Tutorial unter "Filtern von Daten" beschrieben, oder verbinden Sie das Datenelement **Travel** mit dem Datenelement **expense-item**.

4.5.2 Duplizieren von Input-Datenelementen

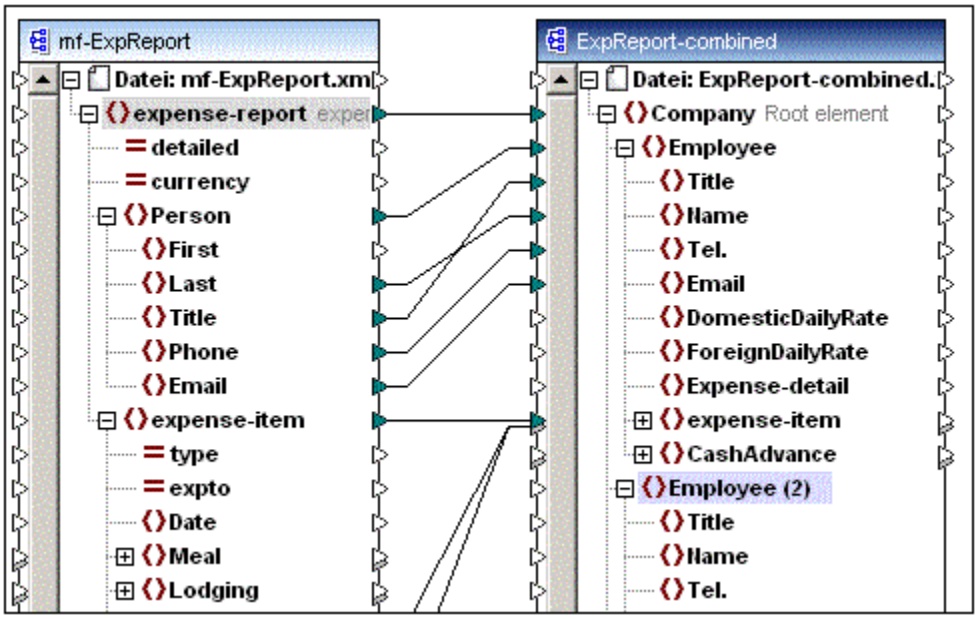
Um mehrere Quelldatenelemente auf ein und dasselbe Zieldatenelement mappen zu können, müssen wir nun die **Eingabeelemente** der Zielkomponente duplizieren, um Mappings aus einer anderen XML-Quelldatei zu erzeugen. Zu diesem Zweck fügen wir die zweite XML-Quelldatei hinzu und erstellen ein Mapping von dieser zu den gleichen Inputs des duplizierten Datenelements in der XML-Zieldatei.

Duplizieren von Datenelementen:

1. Rechtsklicken Sie in der XML-Zieldatei auf den Eintrag Employee.
2. Wählen Sie im Kontextmenü die Option **Duplikat danach einfügen.**

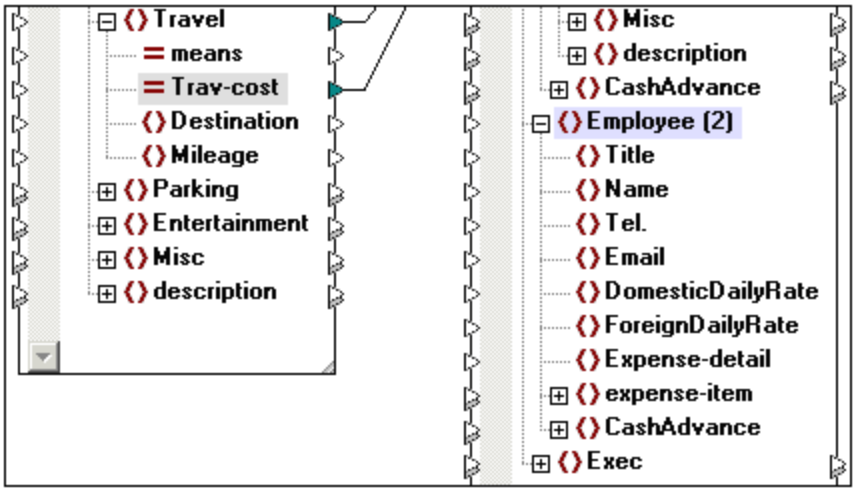


Der Komponente wurde nun ein zweites Element Employee hinzugefügt, nämlich **Employee(2)**.



3. Klicken Sie auf das Erweiterungssymbol, um die untergeordneten Elemente anzuzeigen.
   Die **Struktur** des neuen Employee-Elements ist eine exakte Kopie des Originals mit der einzigen Ausnahme, dass die duplizierten Elemente über **kein Output-Symbol** verfügen.

 © 2011 Altova GmbH



Sie können diese neuen duplizierten Elemente nun als **Zielelemente** für die zweite XML-Quelldatei verwenden.

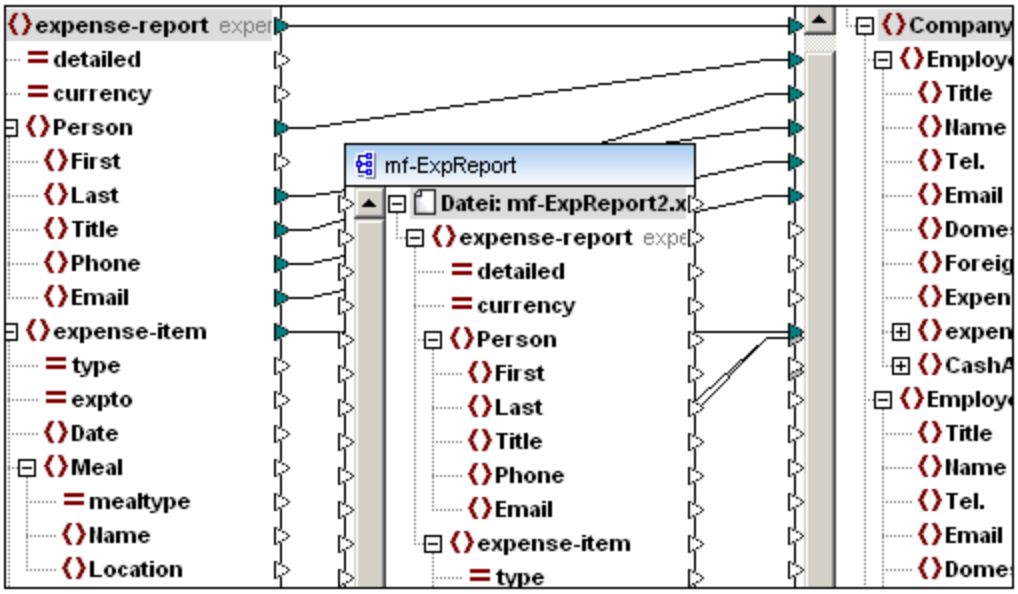
Einfügen der zweiten XML-Instanzdatei

Um die zweite XML-Instanzdatei einzufügen, gehen Sie vor wie oben beschrieben und verwenden Sie dazu dieselbe XML-Schemadatei wie zuvor.

So fügen Sie eine zweite Quellkomponente ein:

1. Klicken Sie auf das Symbol **XML-Schema/Datei einfügen** .
2. Wählen Sie im Dialogfeld "Öffnen" die Datei **mf-ExpReport.xsd,** klicken Sie auf "Durchsuchen..." und wählen Sie als XML-Instanzdatei die Datei **mf-ExpReport2.xml**.
3. Klicken Sie auf den Eintrag **expense-report**, drücken Sie auf der Zahlentastatur die *-Taste, um alle Elemente einzublenden und passen Sie die Größe der Komponente ggf. an.

   Aus Gründen der Übersichtlichkeit wurde die neue Komponente in der folgenden Abbildung zwischen den beiden bestehenden platziert.

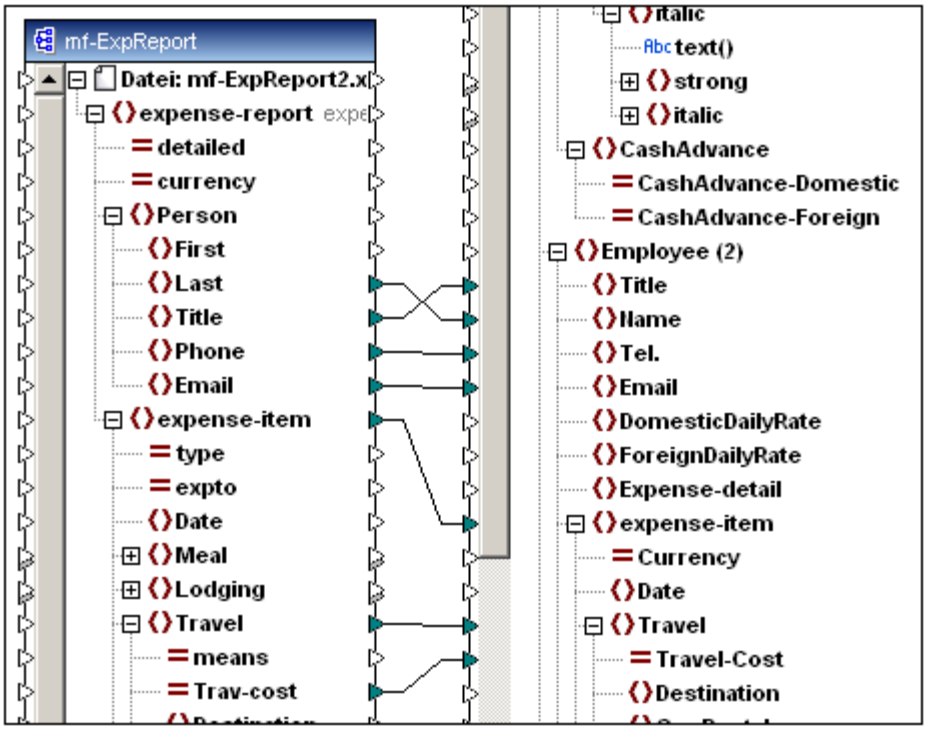


4. Erzeugen Sie dieselben Mappings wir für die erste XML-Quelldatei:

   - Person auf Employee(2)
   - Last auf Name
   - Title auf Title
   - Phone auf Tel.
   - Email auf Email
   - expense-item auf expense-item

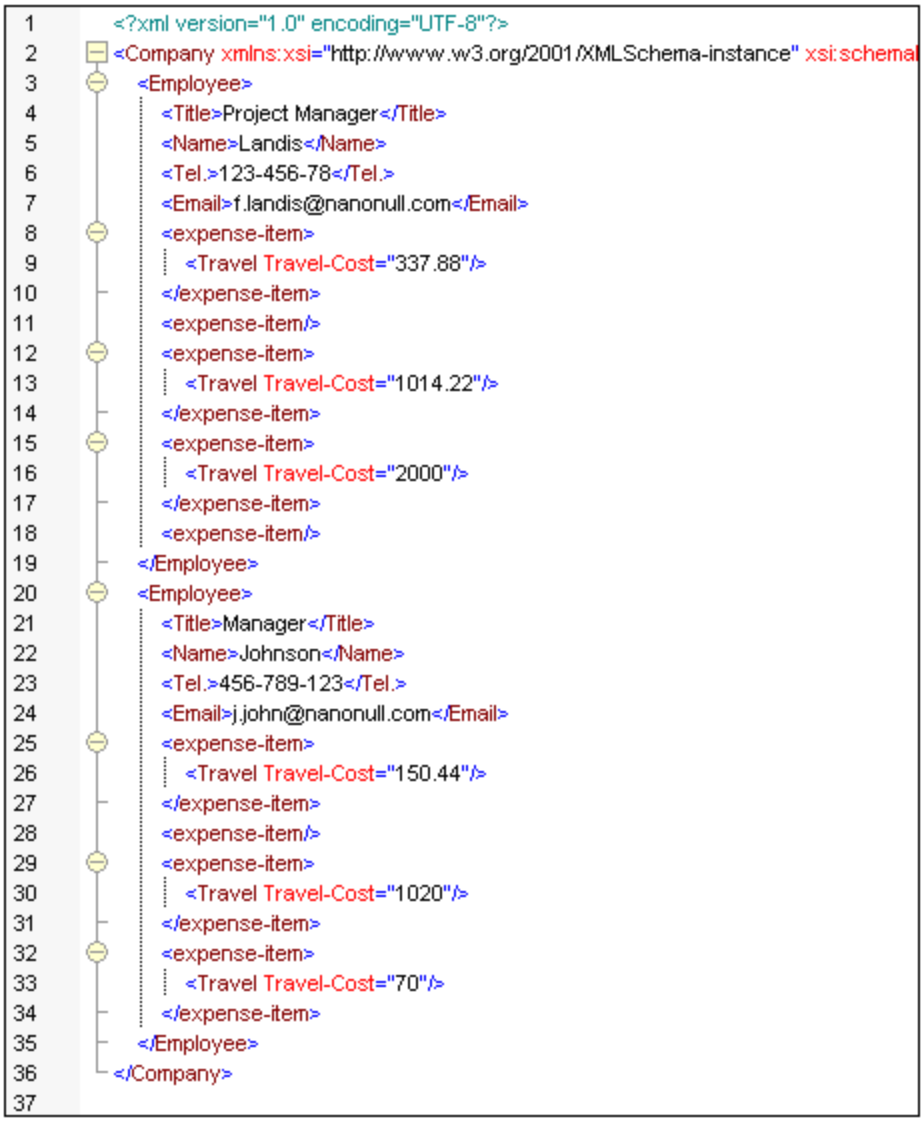
Führen Sie einen Bildlauf nach unten durch und mappen Sie

- Travel auf Travel und
- Trav-cost auf Travel-Cost



5. Klicken Sie auf die Schaltfläche "Ausgabe", um das Mapping-Ergebnis anzuzeigen.

 © 2011 Altova GmbH



```
<?xml version="1.0" encoding="UTF-8"?>
<Company xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemal
  <Employee>
    <Title>Project Manager</Title>
    <Name>Landis</Name>
    <Tel.>123-456-78</Tel.>
    <Email>f.landis@nanonull.com</Email>
    <expense-item>
      <Travel Travel-Cost="337.88"/>
    </expense-item>
    <expense-item/>
    <expense-item>
      <Travel Travel-Cost="1014.22"/>
    </expense-item>
    <expense-item>
      <Travel Travel-Cost="2000"/>
    </expense-item>
    <expense-item/>
  </Employee>
  <Employee>
    <Title>Manager</Title>
    <Name>Johnson</Name>
    <Tel.>456-789-123</Tel.>
    <Email>j.john@nanonull.com</Email>
    <expense-item>
      <Travel Travel-Cost="150.44"/>
    </expense-item>
    <expense-item/>
    <expense-item>
      <Travel Travel-Cost="1020"/>
    </expense-item>
    <expense-item>
      <Travel Travel-Cost="70"/>
    </expense-item>
  </Employee>
</Company>
```

Die Daten der zweiten Reisekostenabrechnung wurden der Ausgabedatei hinzugefügt. Johnson und seine Reisekosten wurden in der Firmen-Spesenabrechnung zu den Ausgaben-Elementen von Fred Landis hinzugefügt.

So speichern Sie die generierte Ausgabe in einer Datei:

Klicken Sie auf das Symbol **Generierte Ausgabe speichern** , das Sie in der Titelleiste sehen, wenn das Fenster "Ausgabe" aktiv ist.

Sie finden Sie Datei mf-ExpReport-combined.xml im Ordner ...\MapForceExamples\Tutorial\.

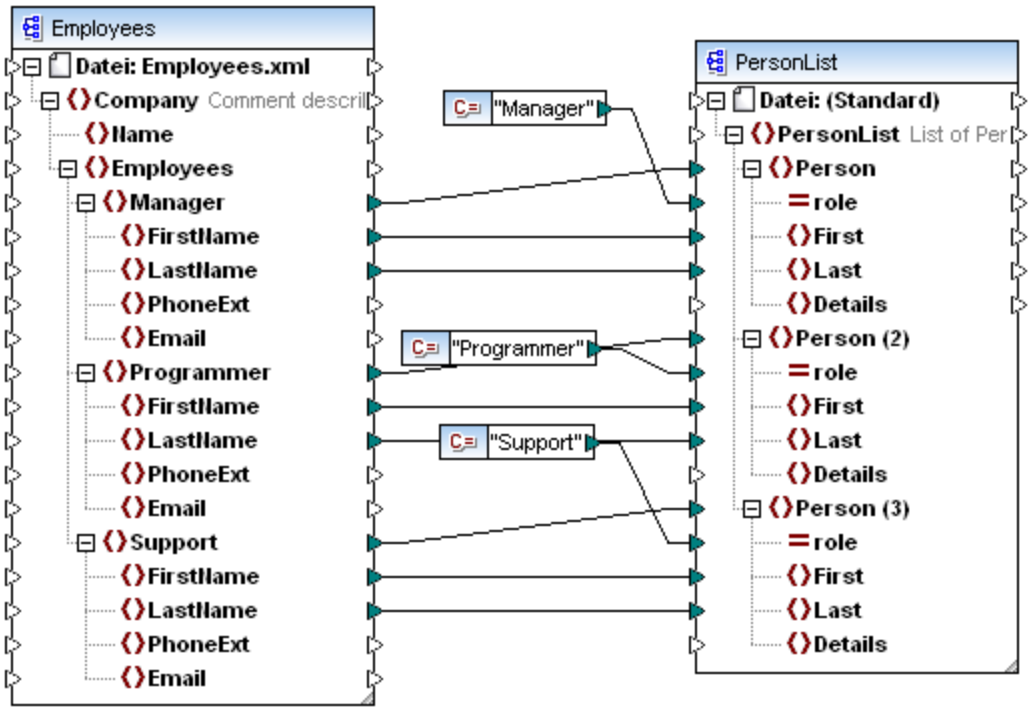
So löschen Sie duplizierte Datenelemente:

- Rechtsklicken Sie auf das duplizierte Datenelement und wählen Sie im Menü die Option **Duplikat löschen**.

Beispiel

Ein weiteres Beispiel, bei dem duplizierte Datenelemente verwendet werden, finden Sie in der Beispieldatei **PersonList.mfd** im Ordner **...\MapForceExamples**.

Im Beispiel **PersonList.mfd** werden unterschiedliche Datenelemente im Quelldokument auf "dasselbe" Element im Schema/XML-Zieldokument gemappt und spezifische Elemente (Manager usw.) mit Hilfe eines "role"-Attributs auf ein einziges generisches gemappt.




```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <PersonList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="
   C:/.../Altova/MapForce2011/MapForceExamples/PersonList.xsd">
3    <Person role="Manager">
4      <First>Vernon</First>
5      <Last>Callaby</Last>
6    </Person>
7    <Person role="Programmer">
8      <First>Frank</First>
9      <Last>Further</Last>
10   </Person>
11   <Person role="Support">
12     <First>Loby</First>
13     <Last>Matise</Last>
14   </Person>
15   <Person role="Support">
16     <First>Susi</First>
17     <Last>Sanna</Last>
18   </Person>
19 </PersonList>
```

 © 2011 Altova GmbH

4.6 Input aus mehreren Dateien / Output in mehrere Dateien

In diesem Abschnitt werden die neuen MapForce Funktionen für den Komponenten-Input aus mehreren Dateien / für den Komponenten-Output in mehrere Dateien beschrieben. Beachten Sie bitte, dass diese Funktionalität für XSLT 1.0 nicht zur Verfügung steht.

Eine einzige Input-Komponente verarbeitet zwei Quelldokumente, während eine einzige Output-Komponente zwei Ausgabedateien generiert. Das hier verwendete Beispiel wurde im Abschnitt Filtern von Daten eingerichtet und dient auch als Basis im Abschnitt Mappen mehrere Quellelemente auf ein einziges Zielelement.

Ziel
Ziel dieses Abschnitts ist die Erstellung eines Mappings, in dem die Quellkomponente zwei XML-Input-Dateien verarbeitet und die Zielkomponente zwei XML-Zieldateien ausgibt.

In diesem Abschnitt verwendete Befehle

Alle Ausgabedateien speichern...: Befindet sich im Menü **Ausgabe**. Wählen Sie diesen Befehl um alle gemappten Dateien aus dem Vorschaufenster zu speichern.

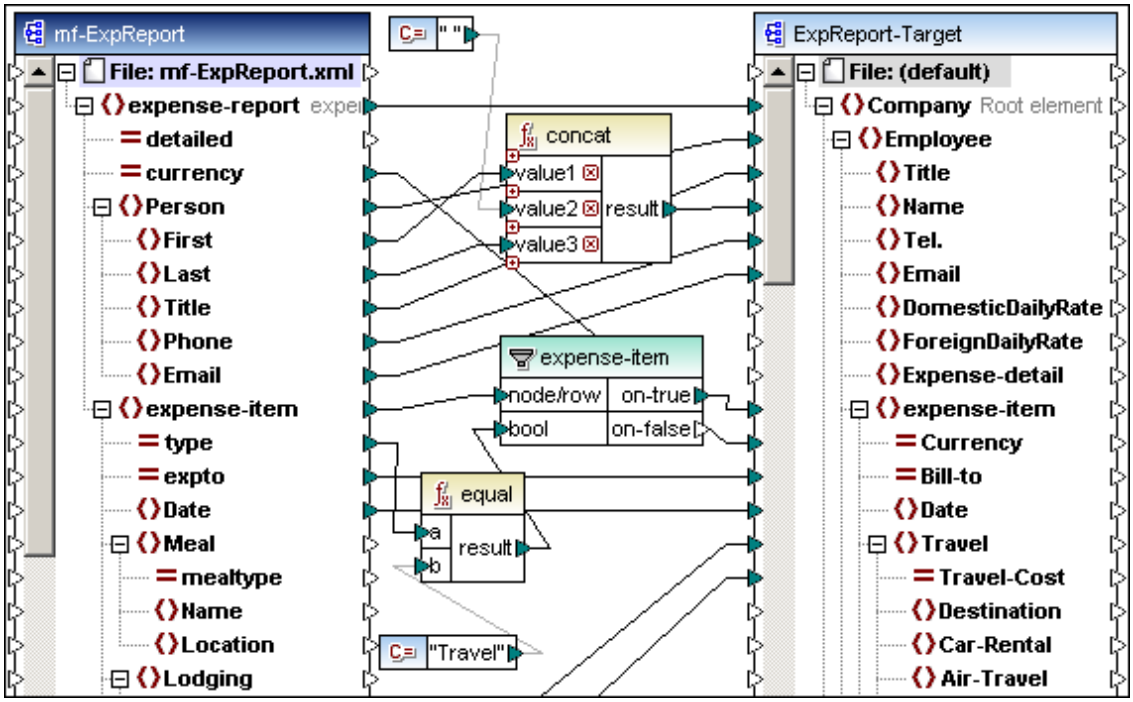
In diesem Abschnitt verwendete Beispieldateien:

mf-ExpReport.xml	Im vorigen Abschnitt verwendete XML-Input-Datei
mf-ExpReport2.xml	die zweite XML-Input-Datei
Tut-ExpReport-multi.mfd	Die Mapping-Datei für dieses Beispiel

Bitte beachten Sie: Die in diesem Abschnitt verwendeten Dateien stehen auch im Ordner ...\MapForceExamples\Tutorial\ zur Verfügung.

4.6.1 Verarbeiten mehrerer Dateien pro Input-/Output-Komponente

Die Datei **Tut-ExpReport.mfd** im Ordner MapForceExamples wird in diesem Beispiel geändert und unter einem anderen Namen gespeichert.



Beachten Sie bitte die folgenden Datenelemente am oberen Rand jeder Komponente:

- Das Datenelement **Datei:mf-ExpReport.xml** von mf-ExpReport enthält den Input/ Output-XML-Dateieintrag. Wenn die Input- und die Output-Datei identisch sind, wird ein Eintrag angezeigt; wenn dies nicht der Fall ist, wird **Input-Dateiname;Output-Dateiname** angezeigt.

  Dies wird automatisch ausgefüllt, wenn Sie einer XML-Schemadatei eine XML-Instanzdatei zuweisen.

- Das Datenelement **Datei: (Standard)** von ExpReport-Target zeigt an, dass der XML-Schemakomponente beim Einfügen keine Instanzdatei zugewiesen wurde., d.h. das Output-XML-Dateifeld ist leer. Daher wird bei der Ausführung des Mappings ein Standardwert verwendet.

Verarbeiten mehrerer Dateien

Um mehrere Dateien verarbeiten zu können, verwendet MapForce im Dateinamen der XML-Input-Datei den Platzhalter "?". Das Zeichen "?" kann durch kein Zeichen oder ein Zeichen ersetzt werden.

So werden mehrere Dateien verarbeitet:

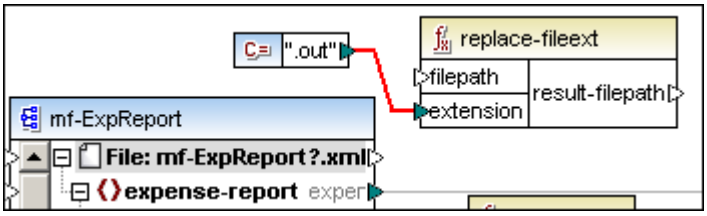
Geöffnet ist die Datei "Tut-ExpReport" aus dem Ordner ...\Tutorial, wobei Sie als Ausgabesprache in der Symbolleiste auf "XSLT 2.0" geklickt haben:

1. Doppelklicken Sie auf der linken Seite auf die Komponente **mf-ExpReport**.
2. Fügen Sie mf-expReport**?**.xml in das Feld "XML-Input-Datei" ein.

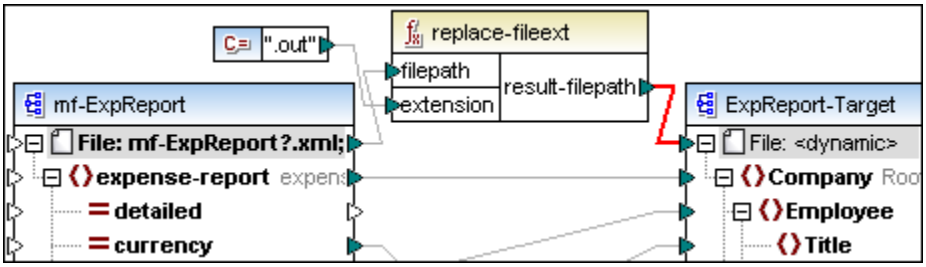
 © 2011 Altova GmbH

In Dateinamen werden die Platzhalterzeichen **?** und ***** unterstützt. Beachten Sie, dass hier ein relativer Pfad eingegeben wurde, da die Datei Tut-ExpReport.mfd sich im Ordner ...\Tutorial befindet (Auf Wunsch kann auch ein absoluter Pfad verwendet werden).

3. Fügen Sie die Funktion **replace-fileext** aus der Bibliothek **file path functions** ein und fügen Sie anschließend eine Konstantenkomponente ein.
4. Fügen Sie in die Konstantenkomponente ".out" ein und verbinden Sie sie mit dem **extension-**Parameter der Funktion.

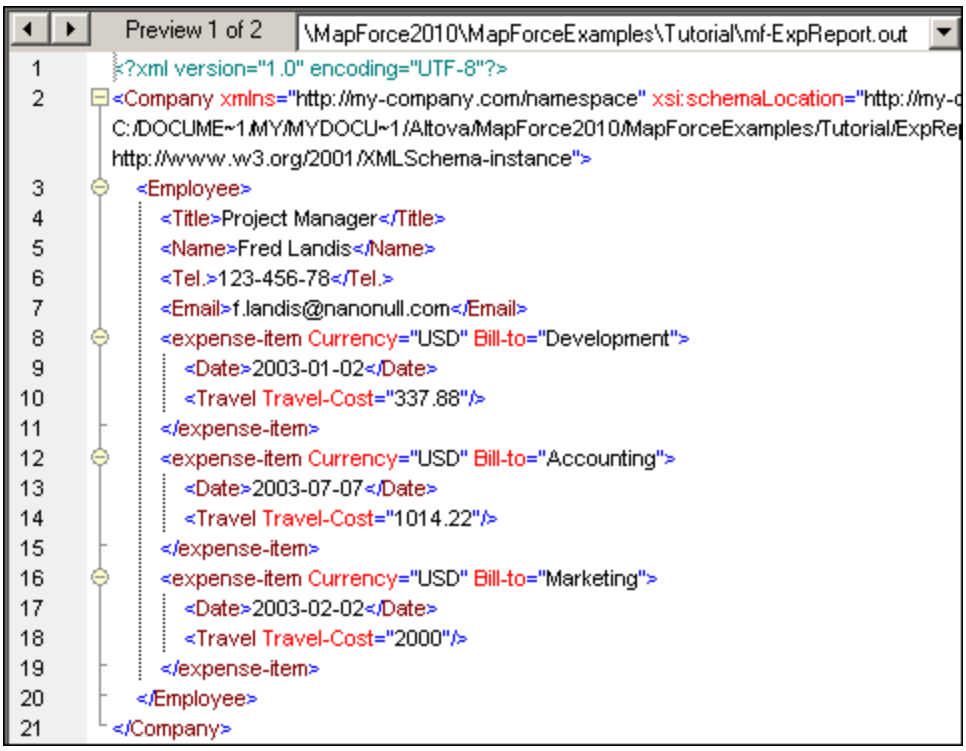


5. Verbinden Sie das Datenelement **Datei**:mf-ExpReport?.xml der Komponente mit dem **filepath**-Parameter der Funktion.
6. Verbinden Sie den **result-filepath**-Parameter der Funktion mit dem Datenelement **Datei <Standard>** der Zielkomponente.



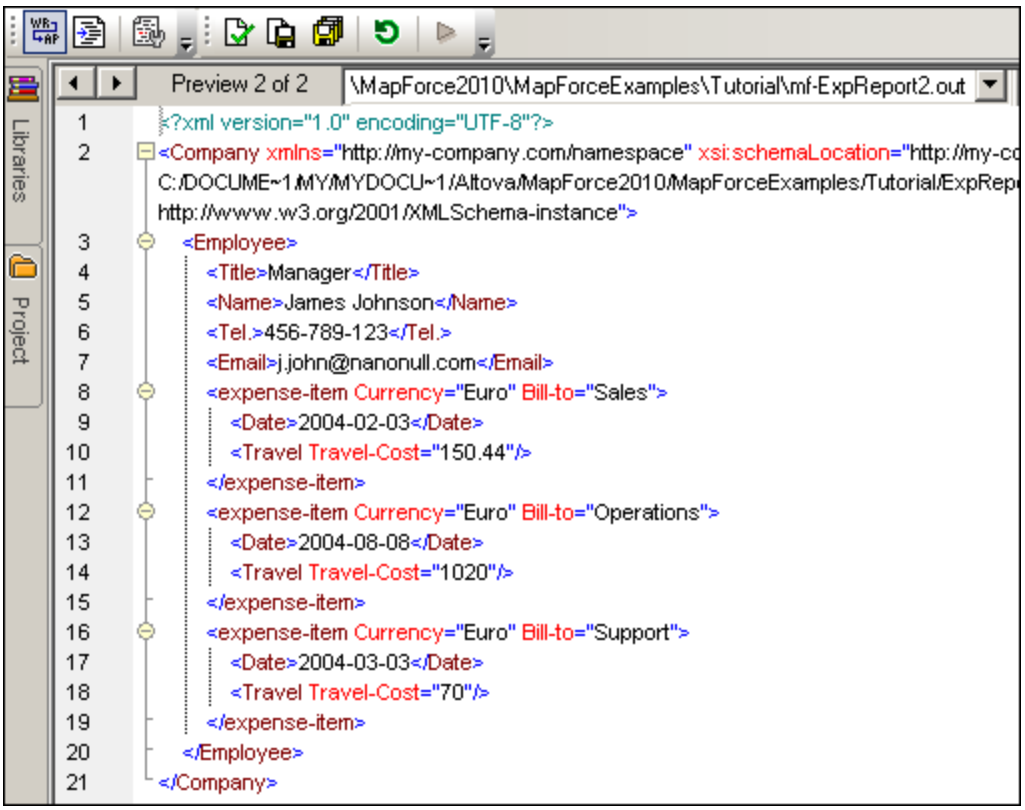
Das Datenelement File: der Zielkomponente hat sich nun ebenfalls in **Datei: <dynamisch>** geändert.

7. Klicken Sie auf die Schaltfläche "Ausgabe", um die Ergebnisse zu sehen.
   Im Ausgabefenster sehen Sie jetzt im Vorschaufenster die Ergebnisse für jede XML-Input-Datei, z.B. Vorschau 1 von 2, wie unten in der Abbildung gezeigt.



8. Klicken Sie auf den Bildlaufpfeil, um das Ergebnis der zweiten XML-Input-Datei zu sehen.
   Beachten Sie, dass in der Auswahlliste die Namen der einzelnen XML-Quelldateien angezeigt werden, wobei die Erweiterung ***.xml** durch die Erweiterung *.**out** ersetzt wurde.

© 2011 Altova GmbH



Wenn Sie auf die Schaltfläche **Alle speichern** klicken, werden alle gemappten Dateien aus dem Vorschaufenster gespeichert, ohne dass Code generiert werden muss. Es erscheint eine Meldung, wenn die Ausgabedateien im selben Ordner überschrieben werden.

9. Speichern Sie die Mapping-Datei unter einem neuen Namen.

Bitte beachten Sie: Nähere Informationen zu mehreren Input-/Output-Dateien finden Sie unter Dynamischer Input/Output.

4.7 Mappen einer Datenbank auf ein Schema

In diesem Abschnitt wird gezeigt, wie man eine einfache Microsoft Access-Datenbank als Datenquelle verwendet, um Datenbankdaten auf ein Schema zu mappen.

Ziel

In diesem Abschnitt lernen Sie, wie man eine Datenbank in ein MapForce-Mapping einfügt und die Tabellen dieser Datenbank auf die Schemazielkomponente mappt. Es werden die folgenden Schritte beschrieben:

- Herstellen einer Verbindung zu einer Microsoft Access-Datenbank
- Mappen von Datenbanktabellen auf XML-Schema-Nodes

In diesem Abschnitt verwendete Befehle

Neu...: Dieser Befehl befindet sich im Menü **Datei** und in der Standard-Symbolleiste. Klicken Sie auf diese Schaltfläche, um das Dialogfeld **Neue Datei** aufzurufen, in dem Sie ein neues Mapping erstellen können.

Datenbank einfügen: Klicken Sie auf diese Schaltfläche, um das Dialogfeld **Datenbank auswählen** aufzurufen, in dem Sie eine Verbindung zu einer Datenbank herstellen können.

XML Schema/Datei einfügen: Klicken Sie auf diese Schaltfläche, um das Windows-Standarddialogfeld "Öffnen" aufzurufen und die Datei aus Ihrem Dateisystem auszuwählen.

Idente Sub-Einträge automatisch verbinden: Klicken Sie auf diese Schaltfläche, um die automatische Verbindung identer Sub-Einträge ein- bzw. auszuschalten.

Code generieren in: Befindet sich im Menü **Datei.** Wenn Sie diesen Befehl auswählen, wird ein Untermenü geöffnet, in dem Sie die gewünschte Art von Code (d.h. **Java, C++** oder **C#**) auswählen können. Mit dem Befehl wird das Dialogfeld **Verzeichnis auswählen** geöffnet, in dem Sie definieren können, wo der generierte Code gespeichert werden soll.

Derzeit werden die folgenden Datenbanken (und Verbindungsarten) unterstützt:

- Microsoft Access; Version 2003 / 2007
- Microsoft SQL Server Versionen 2000, 2005 und 2008
- Oracle Versionen 9i, 10g und 11g
- MySQL 4.x und 5.x
- PostgreSQL 8.0, 8.1, 8.2, 8.3
- Sybase 12
- IBM DB2 Version 8.x und 9
- IBM DB2 für i 5.4
- IBM DB2 für i 6.1

Bitte beachten Sie:
MapForce bietet vollständige Unterstützung für die oben aufgelisteten Datenbanken. Altova ist bestrebt, auch Unterstützung für andere ODBC/ADO-Datenbanken zu gewährleisten, doch wurde die erfolgreiche Herstellung der Verbindung und Verarbeitung der Daten nur bei den aufgelisteten Datenbanken getestet.

 © 2011 Altova GmbH

Stellen Sie bei Installation der **64-Bit** Version von **MapForce** sicher, dass Sie Zugriff auf die 64-Bit-Datenbanktreiber haben, die für die jeweilige Datenbank, zu der Sie eine Verbindung hestellen möchten, benötigt werden.

Anmerkung: Für eine ADO-Verbindung zu MS SQL Server über den Treiber SQL Server Native Client 10.0 müssen die folgenden Eigenschaftswerte auf dem Register *All* des Dialogfelds "Data Link Properties" definiert sein: (i) Setzen Sie den Eigenschaftswert von Integrated Security auf ein Leerzeichen, (ii) Setzen Sie den Eigenschaftswert von Persist Security Info auf `true`.

In der unten stehenden Tabelle sehen Sie die erzeugten Datenbanktypen, die Einschränkungen und die Verbindungsmethoden für das Einbinden der Datenbanken.

	Datenbank einfügen - Verbindungsmethoden ("Schema" anhand einer Datenbank erzeugen)	
Unterstützte Datenbank	**ODBC-Einschränkungen** (Eindeutige Schlüssel werden von ODBC nicht unterstützt)	**ADO-Einschränkungen**
Microsoft Access (ADO)	OK Primär- und Sekundärschlüssel werden nicht unterstützt	OK *
MS SQL Server (ADO)	OK	OK *
Oracle (ODBC)	OK, Einschränkungen: Tabellen mit Spalten vom Typ CLOB, BLOB, BFILE; XML-Tabellen	OK, Einschränkungen: Tabellen mit Spalten vom Typ CLOB, BLOB, BFILE; XML-Tabellen; Owner-Informationen, Identity Constraints werden aus der Datenbank nicht ausgelesen
MySQL (ODBC)	OK *****	OK ⌖
Sybase (ODBC)	OK *****	OK
IBM DB2 (ODBC)	OK *****	OK

- ***** **Empfohlene Verbindungsmethode für die einzelnen Datenbanken.**
- ⌖ **MySQL: Beim Herstellen der ADO-Verbindung auf Basis von ODBC wird empfohlen, entweder die Benutzer- oder die System-DSN zu verwenden.**
- **-** **Nicht verfügbar**

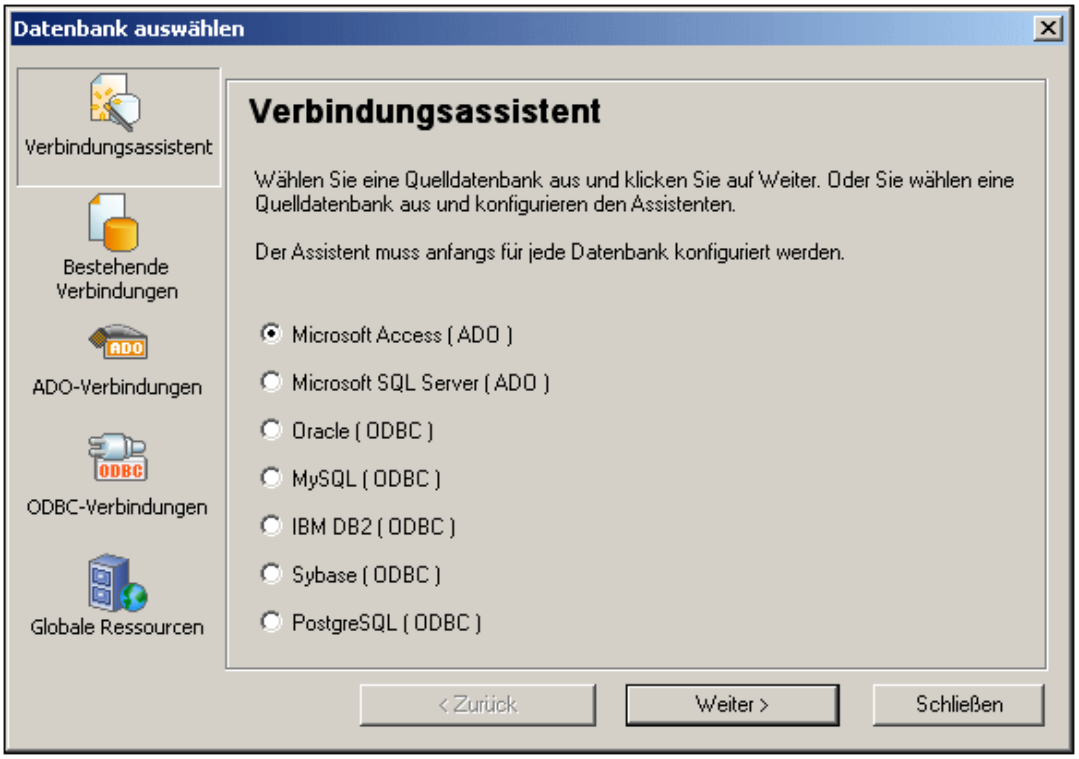
Bitte beachten Sie:

Datenbanken können auch als "globale Ressourcen" eingefügt werden. Nähere Informationen dazu finden Sie unter Globale Ressourcen.

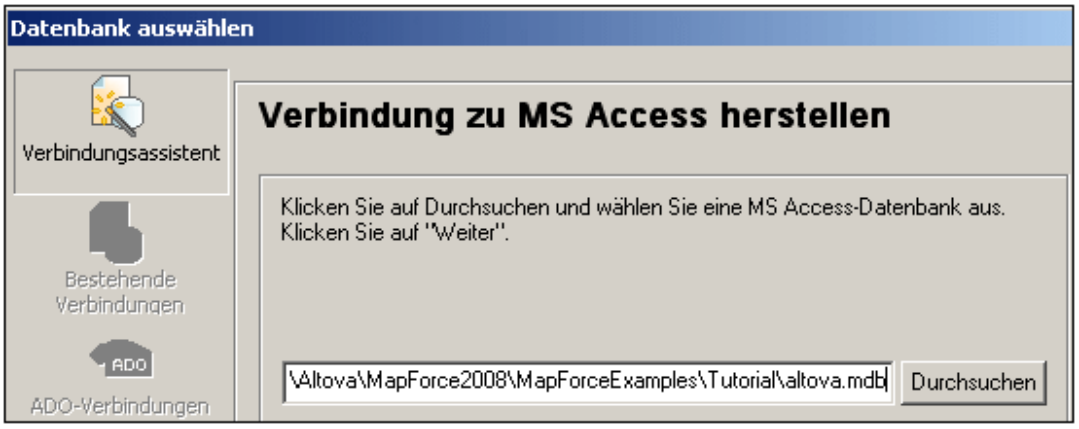
4.7.1 Herstellen einer Verbindung zu einer Access-Datenbank

Erzeugen der Datenbankkomponente in MapForce:

1. Wählen Sie in MapForce **Datei | Neu,** um ein neues Mapping zu erstellen.
2. Klicken Sie in der Titelleiste auf eine der Schaltflächen : Java, C#, C++ oder BUILTIN.
3. Klicken Sie in der Symbolleiste auf das Symbol **Datenbank einfügen**.
4. Klicken Sie im Dialogfeld **Datenbank auswählen** auf die Schaltfläche **Verbindungsassistent.**

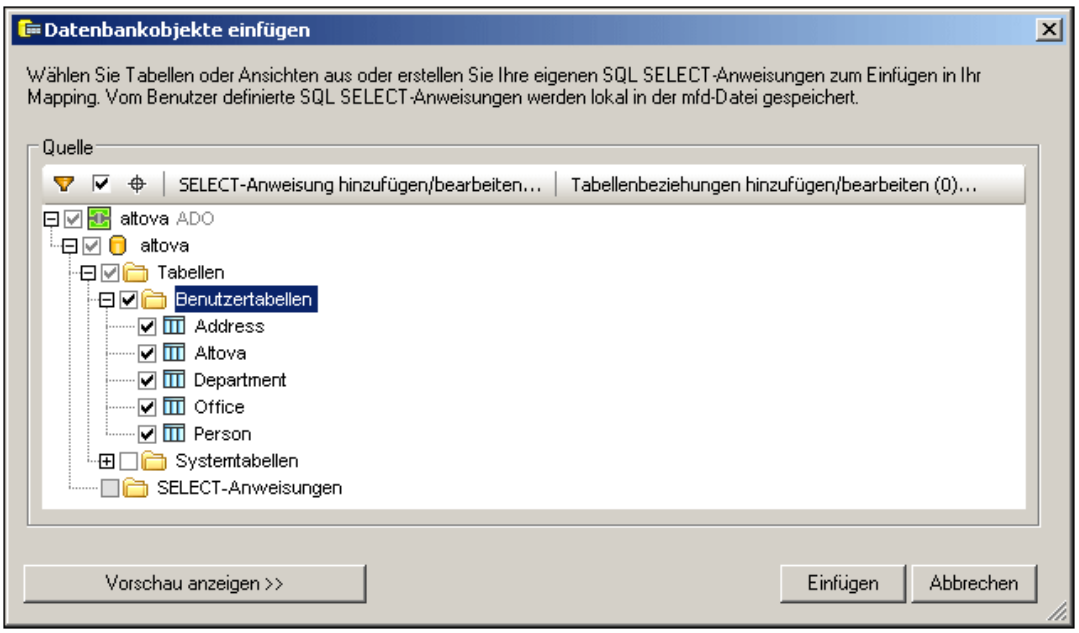


4. Klicken Sie auf das Optionsfeld **Microsoft Access**.
5. Klicken Sie zum Fortfahren auf "Weiter".
6. Klicken Sie auf die Schaltfläche "Durchsuchen", um die gewünschte Datenbank als Datenquelle auszuwählen: In diesem Fall ist die Quelle **altova.mdb** im Ordner ...\MapForceExamples\Tutorial\.
   Im Textfeld wird der Connection String angezeigt.



 © 2011 Altova GmbH

7. Klicken Sie auf **Verbinden**.



8. Klicken Sie im Dialogfeld **Datenbankobjekte einfügen** auf das Kontrollkästchen links von den Benutzertabellen und anschließend auf **Einfügen** um die Datenbank-(Schema-) Komponente einzufügen. Damit werden alle Tabellen des Ordners ausgewählt.

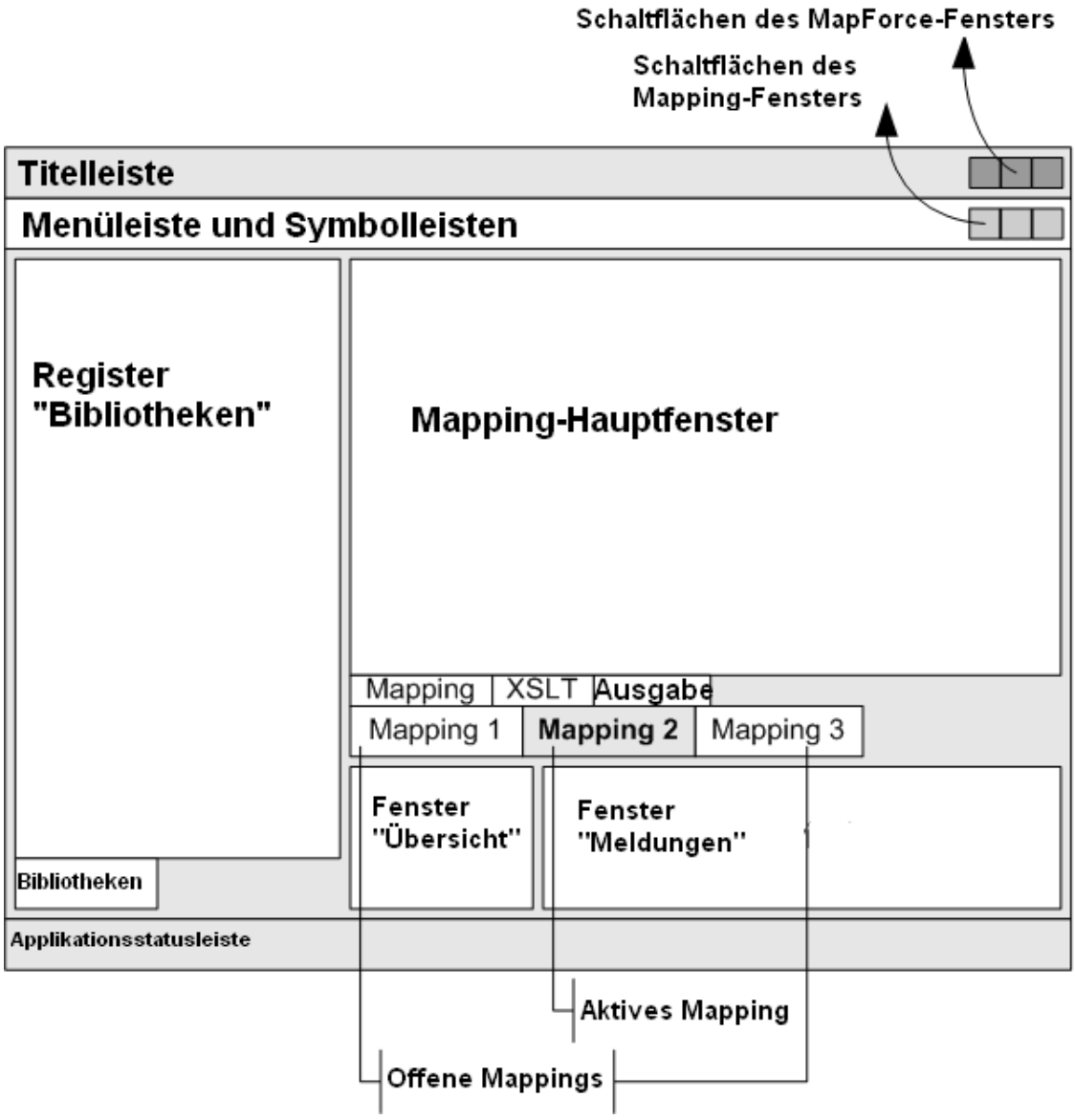
Die Datenbankkomponente wird im Mapping-Fenster angezeigt. Sie können nun Mappings auf ein Zielschema/XML-Dokument erzeugen.

Kapitel 5

MapForce Benutzeroberfläche

5 MapForce Benutzeroberfläche

MapForce besteht aus vier Hauptbereichen: Dem Bereich Bibliotheken auf der linken Seite, dem Bereich "Mapping" (mit den Fenstern Mapping, XSLT, XSLT2, und Ausgabe) auf der rechten Seite sowie den Fenstern Übersicht und Meldungen unterhalb davon.



Titelleiste

In der Titelleiste werden der Name der Applikation (d.h. MapForce) gefolgt vom Namen des aktiven Mapping Design-Fensters angezeigt. Auf der rechten Seite befinden sich Schaltflächen zum Steuern des MapForce-Applikationsfensters.

Menüleiste und Symbolleisten

In der Menüleiste werden die Menübefehle angezeigt. Jede Symbolleiste enthält eine Gruppe von Symbolen, die für MapForce-Befehle stehen. Sie können die Menüleiste und die Symbolleisten an ihren Ziehpunkten mit der Maus an die gewünschte Stelle ziehen.

© 2011 Altova GmbH

Register "Bibliotheken"

Das Register **Bibliotheken** enthält je nach der gewählten Ausgabesprache unterschiedliche Funktionen. Sie können eine Funktion direkt in das Mapping-Fenster ziehen.

Mapping-Fenster

Im Fenster **Mapping** werden die grafischen Elemente dargestellt, die zur Erstellung des Mappings (der Transformation) zwischen den verschiedenen Komponenten verwendet werden. Im Quellschema wird die Struktur des Quellschemas angezeigt und im Zielschema wird die Struktur des Zielschemas angezeigt. Die Input- und Output-**Symbole** der einzelnen Schema-Datenelemente werden durch **Konnektoren** verbunden. Bei Schema-**Datenelementen** kann es sich entweder um Elemente oder Attribute handeln.

Die folgenden Fenster können durch Klicken auf die entsprechenden Schaltflächen am unteren Rand des Mapping-Fensters angezeigt werden:

- In den Fenstern **XSLT** und **XSLT2** wird je nachdem, welche Sprache ausgewählt ist, eine Vorschau des Transformationscodes angezeigt.
- Im Fenster **Ausgabe** wird eine Vorschau der transformierten oder gemappten Daten in einer Textansicht angezeigt.
- In den Fenstern **HTML, RTF, PDF und Word 2007+** werden die Daten der Zielkomponente als HTML-, RTF-, PDF- oder Word 2007+-Dokumente angezeigt, wenn ein StyleVision Power Stylesheet (SPS) mit der Zielkomponente verknüpft ist.

Fenster "Übersicht" und "Meldungen"

Im Fenster **Übersicht** wird der Mappingbereich mit einem roten Rechteck markiert. Sie können das Rechteck verschieben, um in Ihrem Mapping zu navigieren.

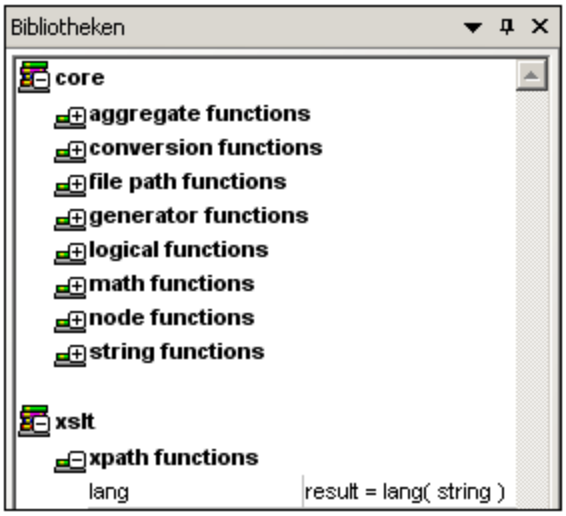
Im Fenster **Meldungen** werden alle Validierungswarnungen oder Fehlermeldungen angezeigt, die während eines Mappings auftreten können. Wenn Sie auf eine Meldung in diesem Fenster klicken, wird sie auf dem Mapping-Register markiert, damit Sie den Fehler beheben können.

Applikationsstatusleiste

Die Statusleiste der Applikation befindet sich am unteren Rand des Applikationsfensters. In ihr werden Informationen auf Applikationsebene angezeigt, darunter auch Tooltipps, die eingeblendet werden, wenn Sie den Mauszeiger über eine Symbolleisten-Schaltfläche platzieren. Wenn Sie die 64-Bit-Version von MapForce verwenden, wird in der Statusleiste das Suffix (x64) hinter dem Applikationsnamen angezeigt. Für die 32-Bit-Version gibt es kein Suffix.

5.1 Das Register "Bibliotheken"

Im **Bibliotheksfenster** werden die für die aktuell ausgewählte Programmiersprache verfügbaren Bibliotheken sowie deren einzelne **Funktionen** angezeigt. Sie können Funktionen mit der Maus direkt in das **Mapping-Fenster** ziehen, wo sie zu Funktionskomponenten werden.



XSLT ausgewählt

Die Standardbibliotheken **core** und **xslt** werden bei jedem Start von MapForce automatisch geladen und müssen nicht mehr vom Benutzer hinzugefügt werden. Die Bibliothek **Core** enthält eine Sammlung von Funktionen, mit Hilfe derer alle Arten von Ausgabecode wie XSLT, generiert werden können. In den anderen Bibliotheken (xslt, xslt2, xpath2, lang usw.) sind Funktionen zusammengefasst, die sich auf die einzelnen Ausgabearten beziehen.

Bei Auswahl von:
XSLT werden die Funktionen der Core- und der XSLT-Bibliothek (XPath 1.0 und XSLT 1.0-Funktionen) aktiviert.

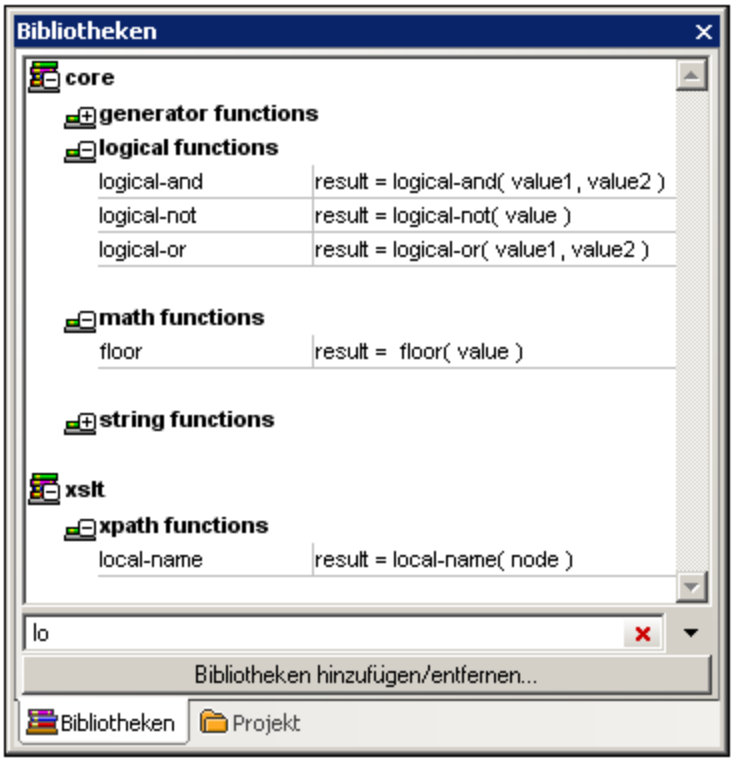
XSLT2 werden die Funktionen der Core-, XPath 2.0 und XSLT 2.0-Bibliothek aktiviert.

Einschränkungen bei XPath 2.0:
Einige XPath 2.0-Funktionen für die Verarbeitung von Sequenzen sind derzeit nicht verfügbar.

Suchen von Funktionen im Fenster "Bibliotheken"

Am unteren Rand des Registers "Bibliotheken" befindet sich ein Suchfeld, über das Sie nach Funktionsnamen suchen können.

© 2011 Altova GmbH



XSLT ausgewählt

Wenn Sie auf die **Esc**-Taste drücken, wird der Filter im Fenster gelöscht. Ein Klick auf **x** hat denselben Effekt.

So suchen Sie im Fenster "Bibliotheken" nach einer Funktion:

1. Klicken Sie in das Fenster "Bibliotheken" um es aktiv zu machen und geben Sie die gesuchten Zeichen ein, z.B. "lo".

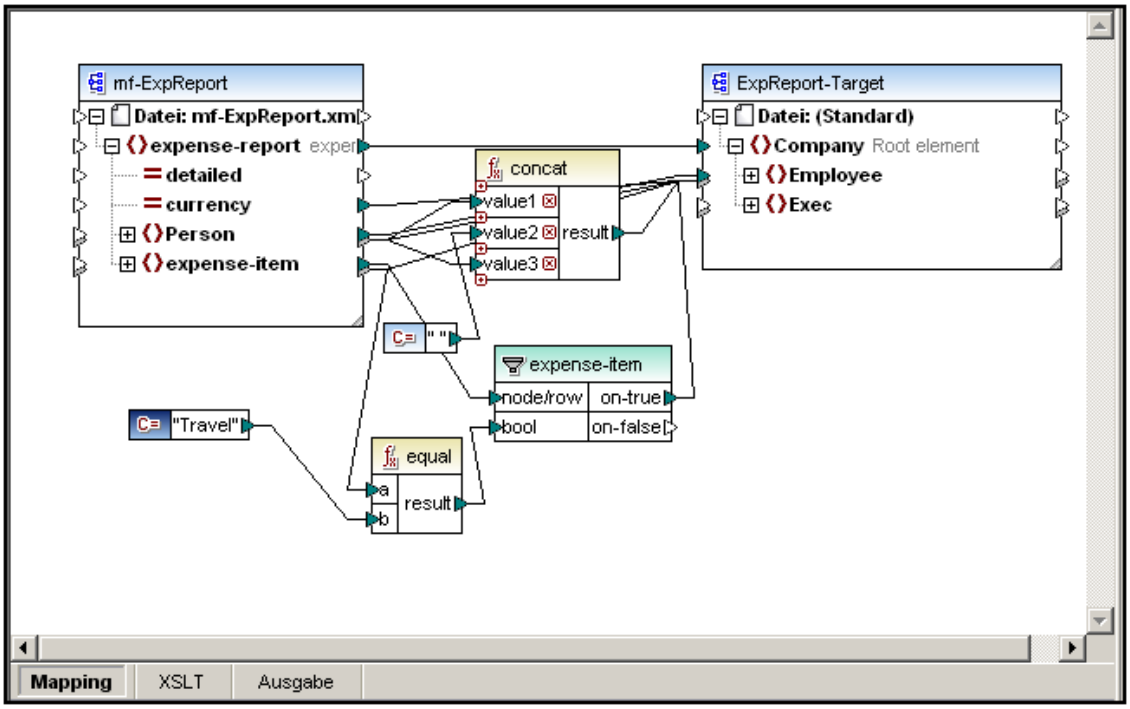
   Alle Funktionen, die diese Zeichen enthalten, werden nun im Fenster "Bibliotheken" in der jeweiligen Funktionsgruppe angezeigt.
3. Klicken Sie auf den Abwärtspfeil und wählen Sie "Funktionsbeschreibungen inkludieren", wenn der Text der Funktionsbeschreibungen in der Funktionssuche berücksichtigt werden soll.

Hinzufügen neuer Funktionsbibliotheken

In MapForce können Sie Ihre eigenen Funktionsbibliotheken erstellen und integrieren. Nähere Informationen dazu finden Sie unter "

5.2 Mapping-Fenster

Das Mapping-Fenster ist der Arbeitsbereich, in dem Sie Ihre Mappings in MapForce erstellen.

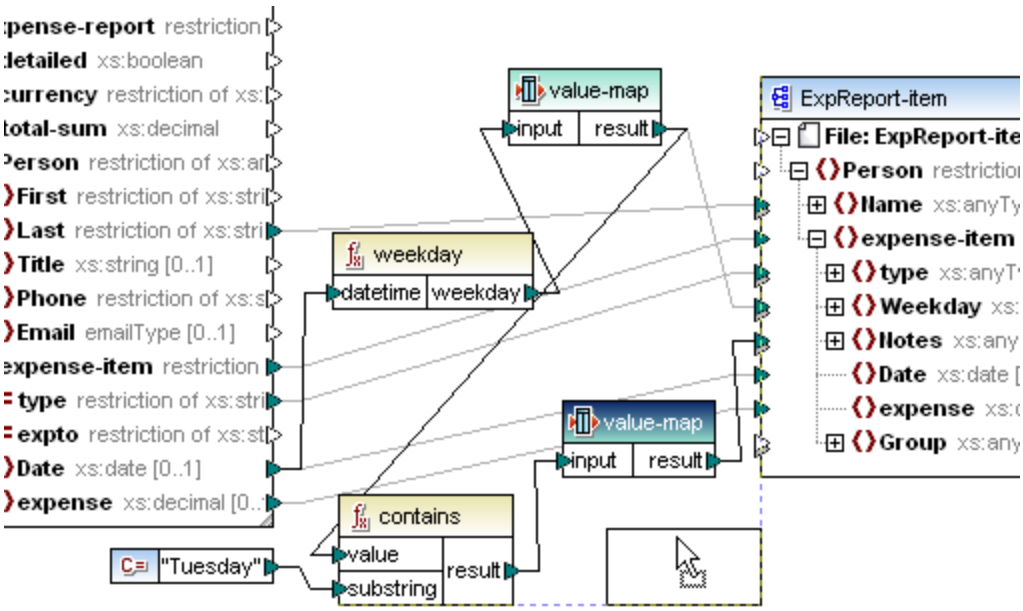


Im Mapping-Fenster werden die grafischen Elemente angezeigt, mit Hilfe derer das Mapping (die Transformation) zwischen den beiden Komponenten durchgeführt wird. Die Input- und Output-Symbole der einzelnen Schemadatenelemente werden durch Konnektoren miteinander verbunden. Bei Schema-Datenelementen kann es sich entweder um Elemente oder Attribute handeln.

Komponenten mit Hilfe von Hilfslinien ausrichten
Wenn Sie Komponenten im Mapping-Fenster verschieben, werden Hilfslinien zur automatischen Ausrichtung der Komponenten angezeigt, anhand derer Sie die Komponente an jeder beliebigen anderen Komponente im Mapping-Fenster ausrichten können. Diese Option kann über die Menüoption **Extras | Optionen | Allgemein** aktiviert/deaktiviert werden.

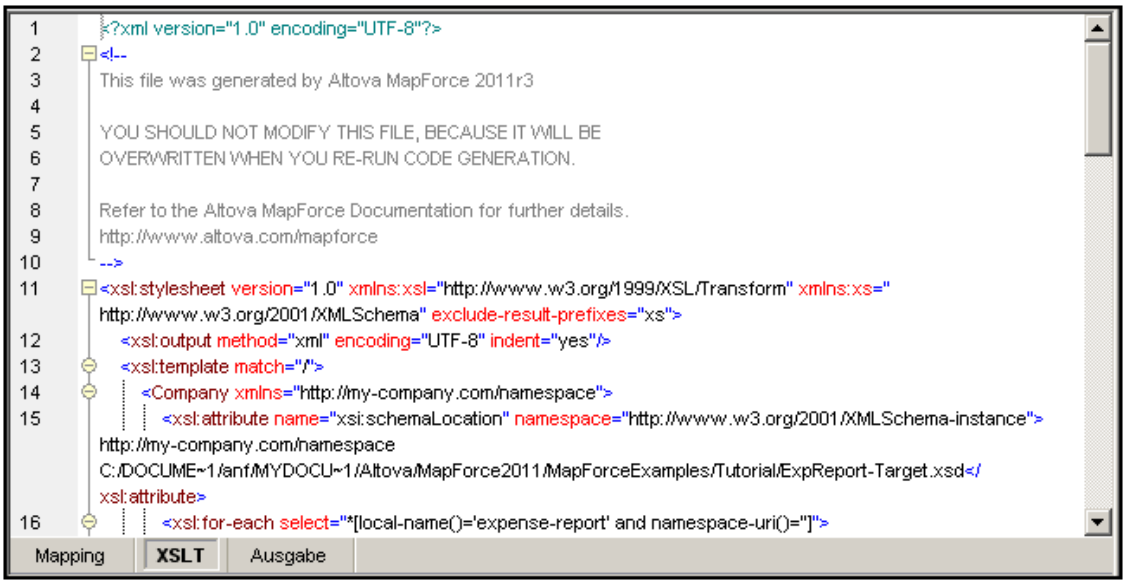
In der Abbildung unten wird die untere "value-map"-Komponente verschoben. Die Hilfslinien zeigen an, dass sie an der "contains"-Funktion und an der "ExpReport-item"-Komponente ausgerichtet werden kann.

© 2011 Altova GmbH


pense-report restriction
detailed xs:boolean
currency restriction of xs:
total-sum xs:decimal
Person restriction of xs:ar
First restriction of xs:stri
Last restriction of xs:stri
Title xs:string [0..1]
Phone restriction of xs:s
Email emailType [0..1]
expense-item restriction
type restriction of xs:stri
expto restriction of xs:st
Date xs:date [0..1]
expense xs:decimal [0..
value-map
input result
weekday
datetime weekday
value-map
input result
contains
value
substring
result
"Tuesday"
ExpReport-item
File: ExpReport-ite
Person restrictio
Name xs:anyTy
expense-item
type xs:anyT
Weekday xs:
Notes xs:any
Date xs:date [
expense xs:
Group xs:any

5.3 Die Fenster XSLT/XSLT2

In den Fenstern **XSLT** und **XSLT2** wird eine Vorschau der jeweils ausgewählten Transformation angezeigt. Die Vorschau hängt von der jeweils ausgewählten Sprache ab.



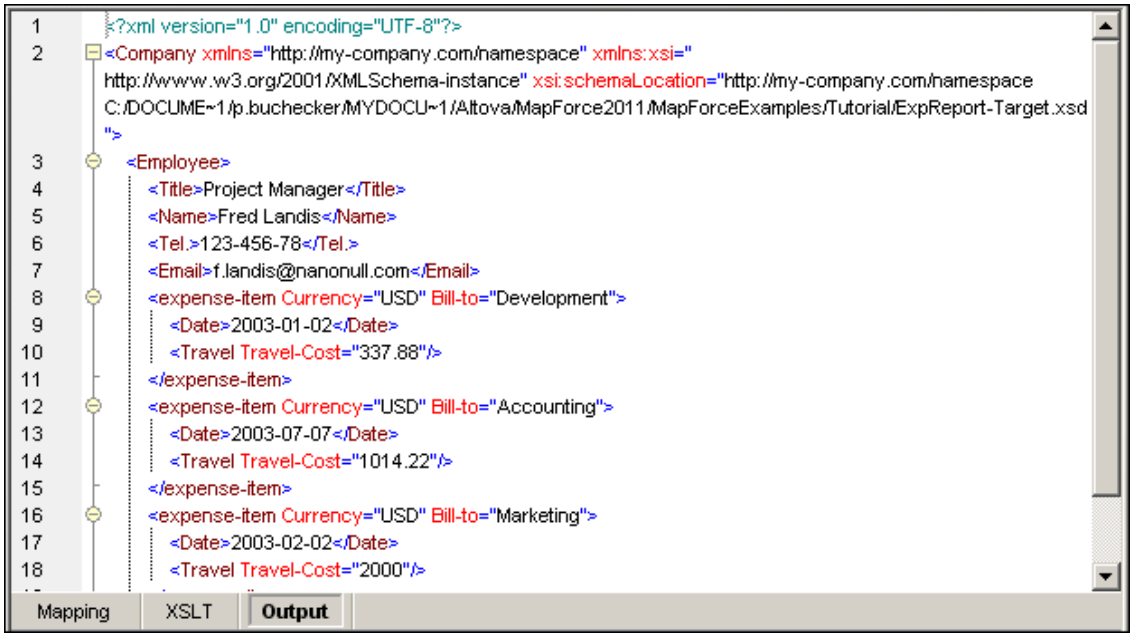
Anmerkung: Wenn Sie die Ausgabesprache ändern möchten, müssen Sie dazu zuerst in das Mapping-Fenster zurückwechseln. Wenn das Register für eine bestimmte Sprache aktiv ist, können Sie die Ausgabesprache im Menü **Ausgabe** bzw. der Symbolleiste **Auswahl der Sprache** nicht ändern.

© 2011 Altova GmbH

5.4 Fenster "Ausgabe" / BUILTIN-Ausführungsprozessor

Das Mapping-Ergebnis wird sofort, je nachdem welche Sprache ausgewählt ist, unter Verwendung des Altova XSLT oder XQuery-Prozessors im Ausgabe-Fenster dargestellt.

Wenn BUILTIN ausgewählt ist, generiert der Built-in-Ausführungsprozessor die Ausgabe anhand der Quellkomponenten: XML/Schema-Dateien. Das Ergebnis, das im Fenster "Ausgabe" angezeigt wird, ist dasselbe, das Sie bei Generierung, Kompilierung und Ausführung des Java-, C++- oder C#-Codes erhalten.

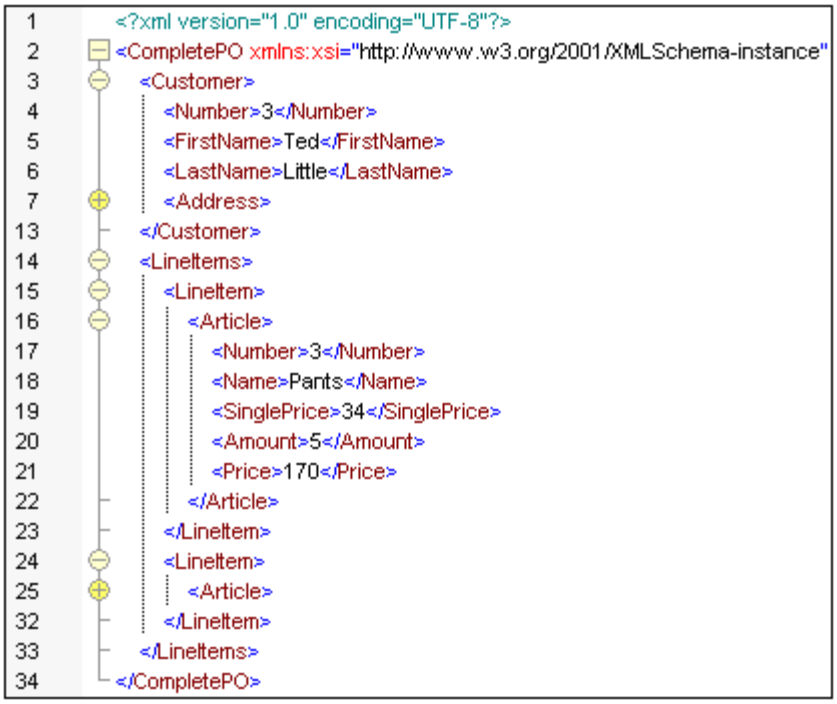


Bitte beachten Sie: Es ist nicht möglich ein Mapping zu öffnen, das Features enthält, die nur in einer höhere MapForce-Edition zur Verfügung stehen. Beispiel: Ein Mapping, das Webservice-Funktionen enthält, kann in der Professional Version nicht geöffnet werden, ebenso wenig wie Datenbankmappings in einer Basic Edition möglich sind.

Je nach **Zielkomponente** Ihres Mappings, werden im Ausgabe-Fenster unterschiedliche Dinge angezeigt:

- **XML Schema/Dokument als Ziel:**

  Im unten gezeigten Screenshot sehen Sie das Ergebnis der Mappings **DB_CompletePO.mfd** aus dem Ordner **...\MapForceExamples**. In diesem Mapping werden sowohl ein XML-Schema/Dokument als auch eine Datenbank als Quellkomponenten verwendet.



Die erzeugte XML-Datei lässt sich durch Klicken auf das Symbol **Generierte Ausgabe speichern** speichern, und durch einen Klick auf das Symbol **Ausgabe validieren** in der Symbolleiste anhand des referenzierten Schemas validieren.

Tastaturkürzel für das Fenster Ausgabe (Tastatur und Ziffernblock):

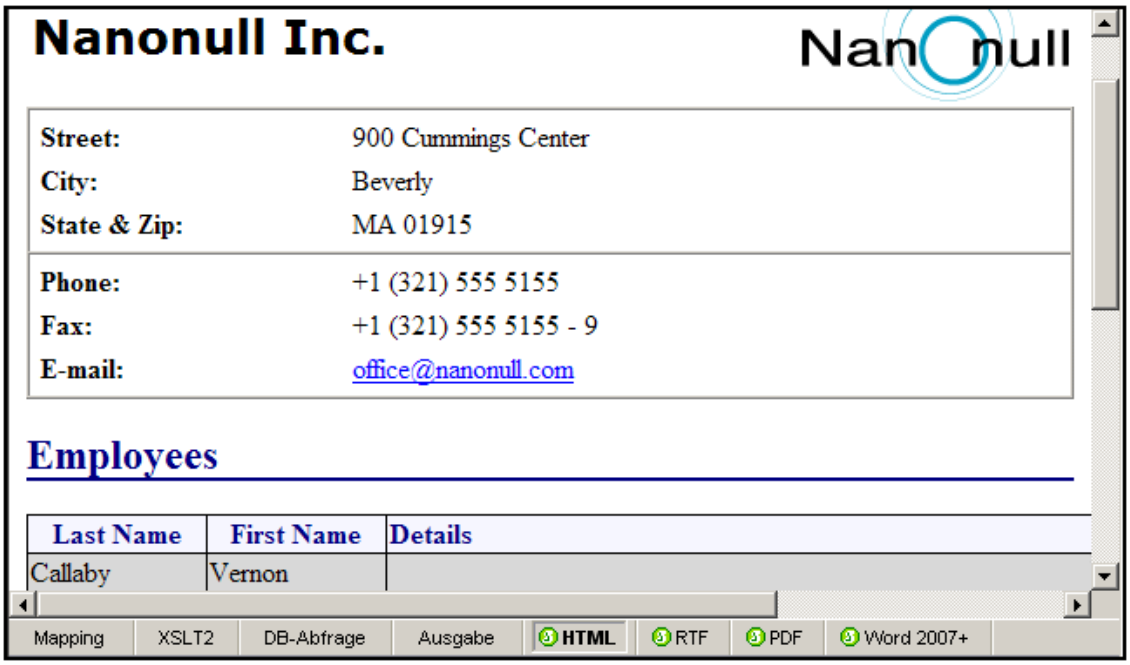
Strg und "+"	in den Text hineinzoomen
Strg und "-"	aus dem Text herauszoomen
Strg und "0"	setzt den Zoomfaktor auf die Standardeinstellung zurück

Strg und Mausrad nach vorne / nach hinten erzielt den gleichen Ein- und Auszoom-Effekt.

© 2011 Altova GmbH

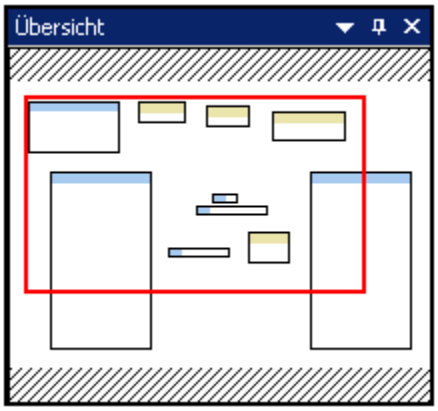
5.5 Fenster im Zusammenhang mit StyleVision

Auf den Registern **HTML**, **RTF**, **PDF** und **Word 2007+** werden die Daten der Zielkomponente als HTML-, RTF-, PDF- oder Word 2007+-Dokument dargestellt, wenn ein StyleVision Power Stylesheet (SPS) mit der Zielkomponente verknüpft ist.



5.6 Fenster "Übersicht"

Das Fenster "Übersicht" dient als Navigationsfenster für große Mappings. Der im Mapping-Fenster sichtbare Bereich wird im Fenster "Übersicht" durch ein rotes Rechteck markiert. Sie können das Rechteck im Fenster "Übersicht" mit der Maus ziehen, um den entsprechenden Bereich im Mapping-Fenster zu sehen.

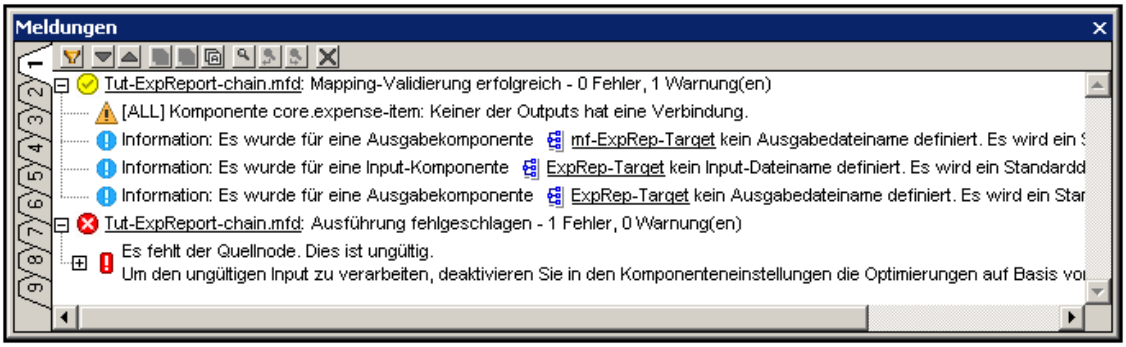


Durch Klicken in das Fenster "Übersicht" wird der Mittelpunkt der Anzeige im Mapping-Fenster definiert.

© 2011 Altova GmbH

5.7 Fenster "Meldungen"

Im Fenster "Meldungen" werden Meldungen, Fehler und Warnmeldungen angezeigt, wenn Sie auf die Schaltfläche "Ausgabe" klicken oder eine Mapping-Validierung ausführen.

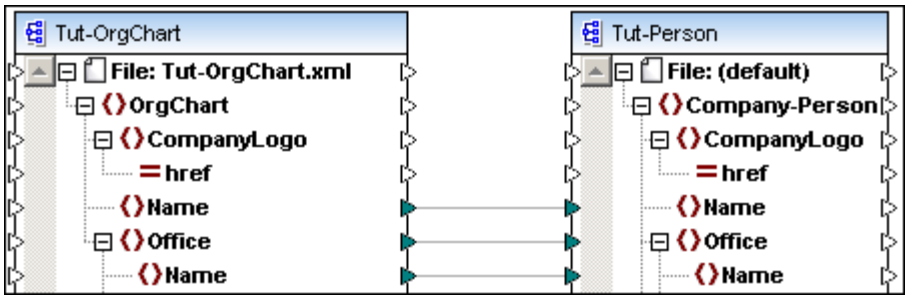


Kapitel 6

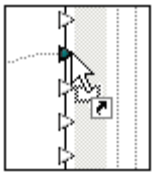
Mapping zwischen Komponenten

6 Mapping zwischen Komponenten

Ein **Konnektor** visualisiert das **Mapping** zwischen zwei Daten-Sets und dient dazu die **Quelldaten** (den Wert) in das Zielschema/-dokument zu transformieren, d.h. zu übernehmen.



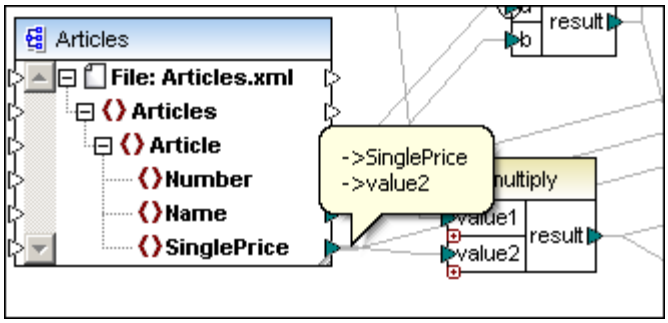
Komponenten und **Funktionen** weisen kleine "Verbindungsdreiecke" auf, die so genannten **Input-** oder **Output-**Symbole. Diese **Symbole** befinden sich links bzw. rechts von allen **Datenelementen**, die gemappt werden können. Wenn Sie auf ein solches Symbol klicken und es mit der Maus ziehen, wird ein **Mapping-Konnektor** erzeugt, den Sie nun auf ein anderes Symbol oder einen Datenelementnamen ziehen können. Neben dem Text-Cursor erscheint ein Verknüpfungssymbol, wenn dieses Mapping zulässig ist.



Wenn Sie auf einen **Datenelementnamen** (Element/Attribut) klicken, wird automatisch das dazugehörige **Quellsymbol** für die Verknüpfung ausgewählt. Wenn Sie den Konnektor auf den Namen des Zieldatenelements ziehen, wird das Zielsymbol automatisch ausgewählt.

Ein **Input-Symbol** kann nur einen Konnektor haben. Wenn Sie versuchen, einen zweiten Konnektor damit zu verbinden, werden Sie gefragt, ob Sie das Input-Symbol ersetzen oder **duplizieren** möchten. Ein **Output-Symbol** kann mehrere Konnektoren haben, die mit unterschiedlichen Input-Symbolen verbunden sind.

Wenn Sie den Mauszeiger über den geraden Abschnitt eines Konnektors platzieren (in der Nähe des Input/Output-Symbols), wird dieser markiert und es erscheint ein Popup-Fenster. Darin sehen Sie den/die Namen des/der Datenelement(e) am anderen Ende des Konnektors. Wenn mehrere Konnektoren vom selben Output-Symbol aus definiert wurden, werden maximal zehn Datenelementnamen angezeigt. In der Abbildung sehen Sie, dass die beiden Ziel-Datenelemente **SinglePrice** und **value2** der multiply-Funktion sind.



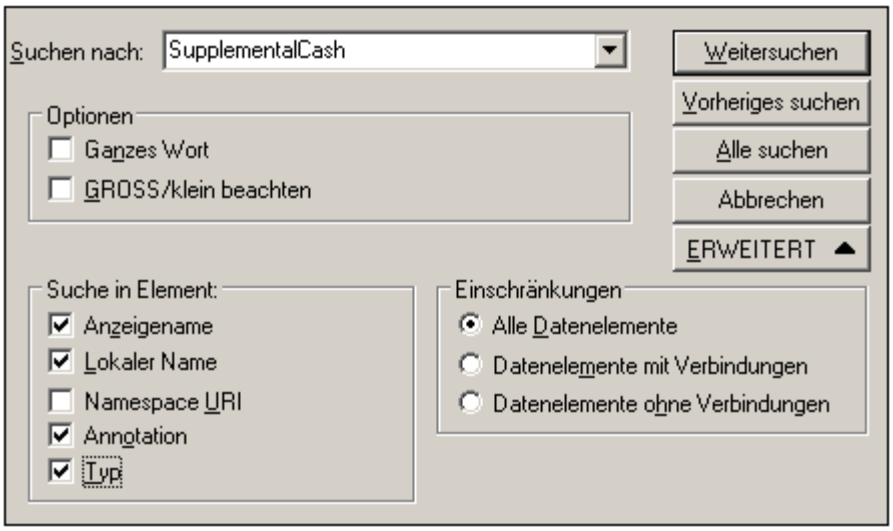
 © 2011 Altova GmbH

Wenn Sie auf den geraden Abschnitt des Konnektors klicken, wird er markiert, sodass Sie ihn mit der Maus an eine andere Stelle ziehen können.

Um einen **duplizierten Konnektor** von derselben Quelle zu einem anderen Ziel zu erstellen, positionieren Sie den Mauszeiger in der Nähe des ursprünglichen Ziels über den geraden Abschnitt des Konnektors und ziehen Sie den Konnektor auf ein anderes Ziel, während Sie die Strg-Taste gedrückt halten.

So suchen Sie in einer Komponente nach einem bestimmten Node/Datenelement:

1. Klicken Sie auf die gewünschte Komponente und drücken Sie die Tasten Strg+F.
2. Geben Sie den Suchbegriff ein und klicken Sie auf Weitersuchen.



Über die oben in der Abbildung angezeigten erweiterten Optionen können Sie festlegen, welche Datenelemente/Nodes durchsucht werden sollen. Außerdem können Sie die Suchoptionen auf Basis der Verbindungen einschränken.

Automatisches Verbindung von Datenelementen

Wenn Sie auf die Schaltfläche "Sub-Einträge automatisch verbinden" klicken und einen Konnektor zwischen den beiden Elementen erstellen, werden alle Sub-Einträge desselben Namens, die sich unter dem übergeordneten Datenelement befinden, automatisch verbunden.

Anzahl der Konnektoren

Input- und Output-Symbole werden an den meisten Komponenten angezeigt, Es besteht jedoch zwischen der Anzahl dieser Symbole keine 1:1-Beziehung.

- Jedes **Schemaelement** (Element/Attribut) hat ein Input- und ein Output-Symbol.
-
- Schema-Komponenten in benutzerdefinierten Funktionen haben nur Output-Symbole.
- Duplizierte Datenelemente haben nur Input-Symbole. Auf diese Art können mehrere Eingaben auf sie gemappt werden. Nähere Informationen dazu finden Sie unter "Duplizieren der Eingabeelemente".
- **Funktionen** können beliebig viele Input- und Output-Symbole haben, und zwar für jeden **Parameter eines**. So hat z.B. die Funktion "Add" zwei (oder mehr) Input-Symbole und ein Output-Symbol.
- **Spezielle** Komponenten können unterschiedlich viele Symbole haben. So hat z.B. die Komponente "Konstante" nur ein Output-Symbol.

6.1 Datenmappingmethoden (Standard / Mixed Content / Sub-Einträge kopieren)

MapForce unterstützt verschiedene Methoden, Daten zu mappen: Zielorientiert (Standard), quellorientiert (Mixed Content) und Alles kopieren (Sub-Einträge kopieren).

Konnektoren und ihre Eigenschaften:
Die folgenden Aktionen können - mit den beschriebenen Ergebnissen - an einem Konnektor ausgeführt werden:

- Durch Klicken auf einen Konnektor wird dieser rot markiert.
- Durch Drücken der **Entf**-Taste, wird der Konnektor sofort gelöscht, wenn er markiert ist.
- Durch Rechtsklicken auf einen Konnektor wird das Konnektor-Kontextmenü geöffnet.
- Durch Doppelklick auf einen Konnektor wird das Dialogfeld "Verbindungseinstellungen" geändert.

Anzeigen von Konnektoren

In MapForce können Sie die Konnektoren im Mapping-Fenster selektiv anzeigen.

Ausgewählte Komponenten-Konnektoren anzeigen
Wechselt zwischen der Anzeige:

- aller Mapping-Konnektoren in Schwarz und
- der Komponenten, die mit der aktuell ausgewählten Komponente verknüpft sind (in Schwarz). Andere Konnektoren erscheinen abgeblendet.

Konnektoren zwischen Quelle und Ziel anzeigen
Wechselt zwischen der Anzeige von:

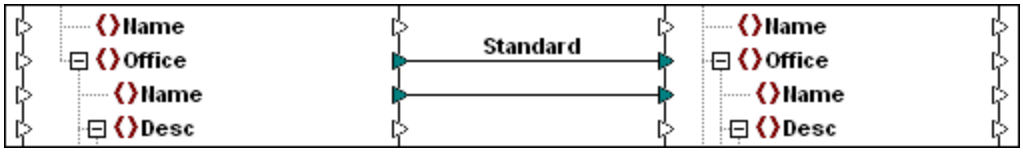
- Konnektoren, die mit der aktuell ausgewählten Komponente **direkt** verbunden sind und
- Konnektoren, die mit der aktuell ausgewählten Komponente verknüpft sind, von einer Quellkomponente ausgehen und diese mit den Zielkomponenten verbinden.

© 2011 Altova GmbH

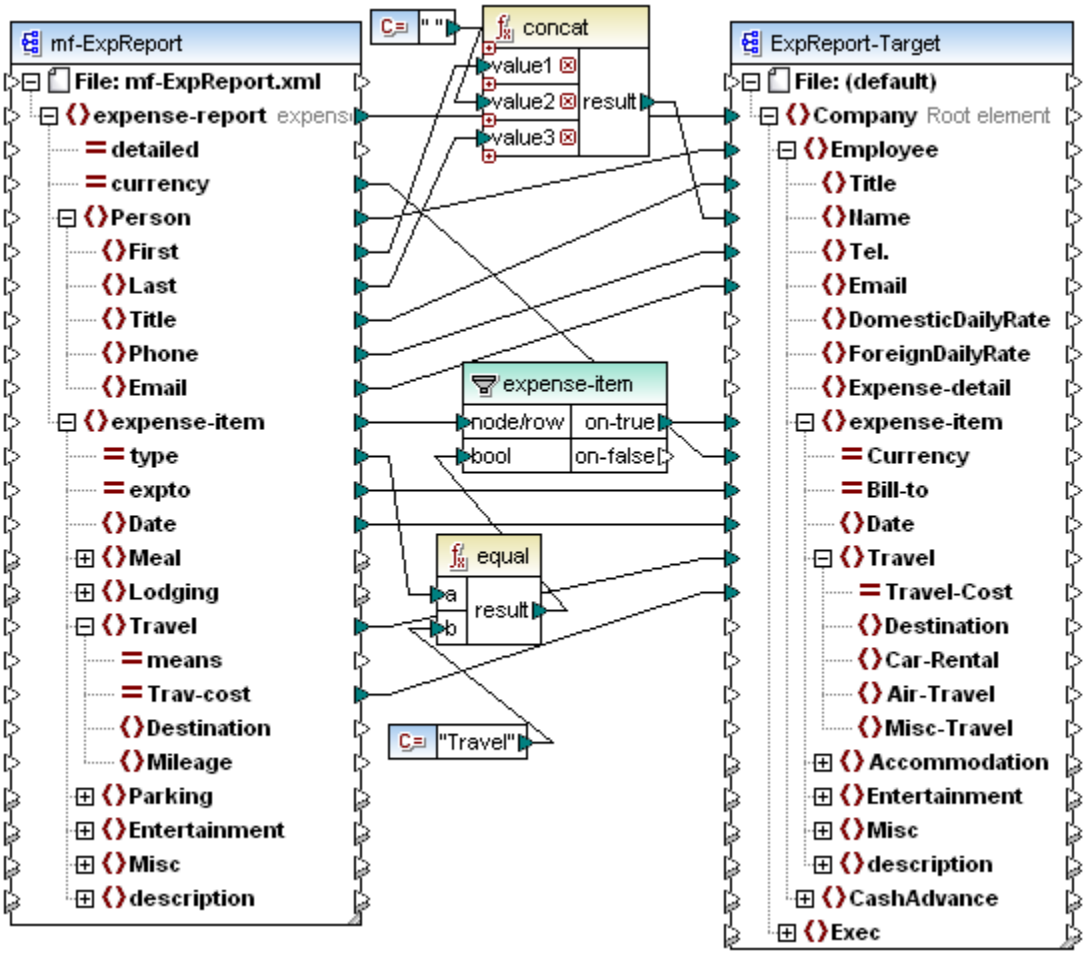
6.1.1 Zielorientiert / Standard-Mapping

Ein Standardmapping ist die normalerweise in MapForce verwendete Mappingmethode, d.h. die Ausgabe ist von der Reihenfolge der Zielnodes abhängig. Nähere Informationen finden Sie unter Quellorientiertes / Mixed Content Mapping im Gegensatz zum Standardmapping.

- Der Inhalt von Mixed Content Text Nodes wird **nicht** unterstützt/gemappt.
- Die Reihenfolge der Child Nodes ist von der **Ziel**-**Schema-**Datei abhängig.



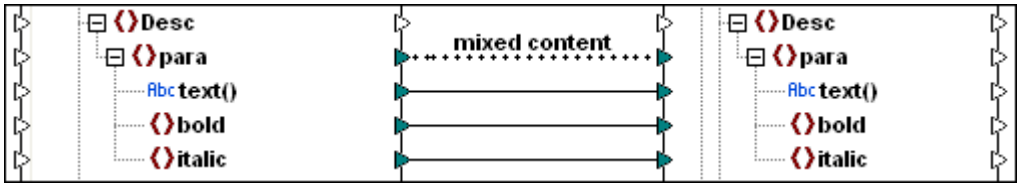
Standard-Mappings werden durch eine durchgezogene Linie gekennzeichnet.



6.1.2 Quellorientiertes / Mixed Content Mapping

In einem quellorientierten (Mixed Content) Mapping können Text-Nodes und Child Nodes automatisch in derselben Reihenfolge, in der sie in der XML-**Quelldatei** vorkommen, gemappt werden.

- Der Inhalt von Mixed Content Text-Nodes wird unterstützt/gemappt.
- Die Reihenfolge von Child Nodes ist abhängig von der **Quell**-XML-Instanzdatei.



Mixed Content-Mappings werden durch eine gepunktete Linie gekennzeichnet.

Quellorientiertes Mapping kann natürlich auf Wunsch auch auf XML-Schema-Datenelemente vom Typ **complexType** angewendet werden. Die Child Nodes werden dann entsprechend ihrer Reihenfolge in der XML-Quelldatei gemappt.

Quellorientiertes / Mixed Content Mapping unterstützt:

- als **Quellkomponenten:**
  - XML-Schema complexTypes (darunter auch Mixed Content, d.h. mixed=true)
- als **Zielkomponenten:**
  - XML-Schema complexTypes (einschließlich Mixed Content),

  Anmerkung: CDATA-Abschnitte werden als Text behandelt.

Mappen von Mixed Content

Sie finden die im folgenden Beispiel (**Tut-OrgChart.mfd**) verwendeten Dateien im Ordner **...\MapForceExamples\Tutorial**.

XML-Quellinstanz:
Im unten gezeigten Screenshot sehen Sie einen Ausschnitt aus der in diesem Abschnitt verwendeten XML-Datei Tut-OrgChart. Hier geht es um das Mixed Content-Element "para" mit seinen Child Nodes "bold" und "italic".

Beachten Sie, dass das Element "para" auch eine Processing Instruction (sort alpha-ascending, also sortiere in aufsteigend in alphanumerischer Reihenfolge) sowie einen Comment-Text (Company details...) enthält, die ebenfalls gemappt werden können.

 © 2011 Altova GmbH

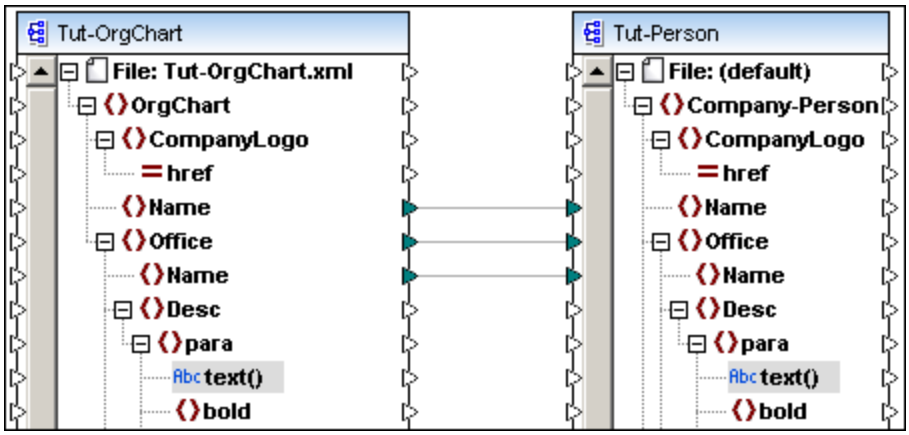
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 sp2 U (http://www.altova.com) by Mr. Nobody (Altova GmbH) -->
<OrgChart xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Tut-OrgChart.xsd">
  <CompanyLogo href="nanonull.gif"/>
  <Name>Organization Chart</Name>
  <Office>
    <Name>Nanonull, Inc.</Name>
    <Desc>
      <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.
        <?sort alpha-ascending?>
        <!--Company details: location and general company information.-->
      </para>
      <para>White papers and further information will be made available in the near future.
    </Desc>
```

Beachten Sie bitte die **Reihenfolge** der Text-Nodes und der bold/italic Nodes von Nanonull., Inc in the XML-Instanzdatei. Sie lautet wie folgt:

<para> The company...
<**bold**>Vereno<**/bold**>in 1995 ...
<**italic**>multi-core...<**/italic**>February 1999

<**bold**>Nano-grid.<**/bold**>The company ...
<**italic**>offshore...<**/italic**>to drive...
</para>

Anfangsmapping
Unterhalb sehen Sie das Mapping in seinem Anfangszustand, wie Sie es sehen, wenn Sie die Datei Tut-Orgchart.mfd öffnen.

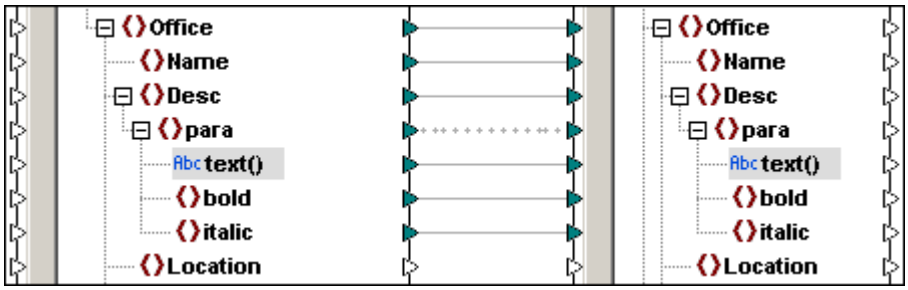


Ausgabe des obigen Mappings:
Unterhalb sehen Sie das Ergebnis dieses ersten Mappings: Es wurden Organization Chart sowie die Namen der einzelnen Büros ausgegeben.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Company-Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNames
3    <Name>Organization Chart</Name>
4    <Office>
5      <Name>Nanonull, Inc.</Name>
6    </Office>
7    <Office>
8      <Name>Nanonull Europe, AG</Name>
9    </Office>
10 </Company-Person>
11
```

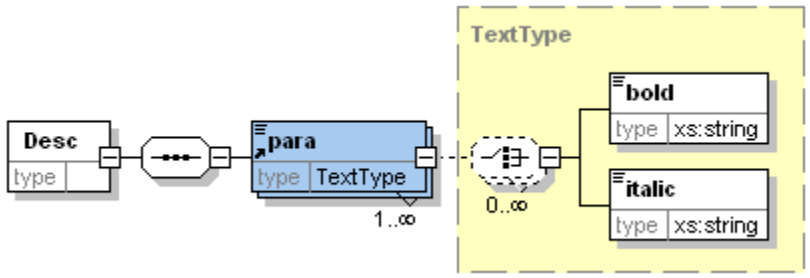
Mappen des para-Elements

In der nachfolgenden Abbildung sehen Sie ein Beispiel für ein Mixed Content Mapping. Das Element "para" hat Mixed Content, daher wird die Verbindungslinie **gepunktet** angezeigt. Der Node **text()** enthält Textdaten und muss gemappt werden, damit der Text in der Zielkomponente aufscheint.



Wenn Sie mit der rechten Maustaste auf einen Konnektor klicken und den Eintrag "Eigenschaften" wählen, können Sie einen Annotationstext zum Konnektor hinzufügen. Nähere Informationen dazu finden Sie im Abschnitt "Referenz" unter "Verbindung".

In der Abbildung unten sehen Sie das Content Model des Elements "Description" (Desc) der Schema-Datei **Tut-OrgChart.xsd**. Diese Definition ist in dem in diesem Beispiel verwendeten Quell- und Zielschema identisch.



Beachten Sie, dass das Element **para** im Content Model die folgenden Eigenschaften hat:

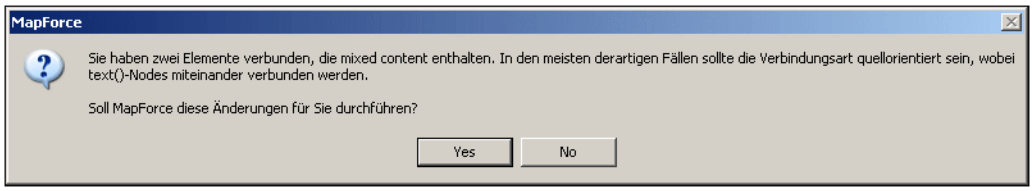
- para ist ein complexType mit **mixed** = true vom Typ TextType.
- Die Elemente "**bold**" (fett) und "**italic**" (kursiv) sind beide vom Typ **xsd:string**. Sie wurden in diesem Beispiel nicht als rekursiv definiert, d.h. sind weder "bold" noch "italic" vom Typ "TextType".
- Die Elemente "**bold**" und "**italic**" können beliebig oft in beliebiger Reihenfolge innerhalb von "**para**" vorkommen.
- Innerhalb des Elements "**para**" kann eine beliebige Anzahl von Text Nodes vorkommen, die eine beliebige Anzahl der Elemente "**bold**" und "**italic**" in beliebiger Reihenfolge enthalten können.

© 2011 Altova GmbH

Erstellen von Mixed Content-Verbindungen zwischen Datenelementen:

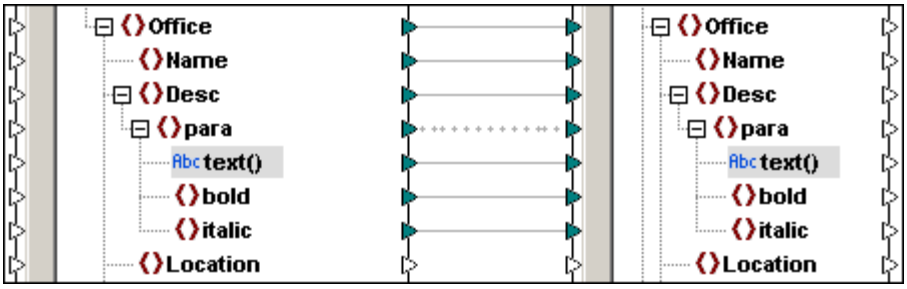
1. Wählen Sie die Menüoption **Verbindung | Idente Sub-Einträge automatisch verbinden,** um diese Option zu aktivieren, falls sie noch nicht aktiv ist.
2. Verbinden Sie das Datenelement **para** im Quellschema mit dem Datenelement **para** im Zielschema.

   Es erscheint eine Meldung, in der Sie gefragt werden, ob MapForce die Konnektores als "quellorientiert" definiert werden sollen.



3. Klicken Sie auf Ja, um eine Mixed Content-Verbindung zu erstellen.

Bitte beachten Sie:
"para" ist vom Typ Mixed Content. Aus diesem Grund wird die Meldung zu diesem Zeitpunkt angezeigt. Die Mixed Content-Meldung wird auch angezeigt, wenn Sie die Datenelemente "para" direkt mappen, ohne dass die Option zur automatischen Verbindung der Sub-Einträge aktiv ist.



Alle Child-Datenelemente von "para" wurden verbunden. Der Konnektor, der die Datenelemente "para" verbindet wird als gepunktete Linie anzeigt, da es sich hierbei um Mixed Content handelt.

4. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis des Mappings zu sehen.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Company-Person xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNames
3    <Name>Organization Chart</Name>
4    <Office>
5      <Name>Nanonull, Inc.</Name>
6      <Desc>
7        <para>The company was established in<bold> Vereno</bold>in 1995. Nanonull devel
8          </para>
9        <para>White papers and further information will be made available in the near future.
10       </para>
11     </Desc>
12   </Office>
13   <Office>
14     <Name>Nanonull Europe, AG</Name>
15     <Desc>
16       <para>In May 2000, Nanonull<italic>Europe</italic> was set up in Vienna. The team co
17     </Desc>
18   </Office>
19 </Company-Person>
20
```

5. Klicken Sie in der Symbolleiste des Ausgaberegisters auf das Symbol **Zeilenumbruch** , um den gesamten Text im Fenster "Ausgabe" sehen zu können.

```
3    <Name>Organization Chart</Name>
4    <Office>
5      <Name>Nanonull, Inc.</Name>
6      <Desc>
7        <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.
8          </para>
9        <para>White papers and further information will be made available in the near future.
10       </para>
11     </Desc>
12   </Office>
13   <Office>
14     <Name>Nanonull Europe, AG</Name>
15     <Desc>
16       <para>In May 2000, Nanonull<italic>Europe</italic> was set up in Vienna. The team
     consists of<bold> five research scientists </bold>and one administrative staff.</para>
17     </Desc>
18   </Office>
19 </Company-Person>
20
```

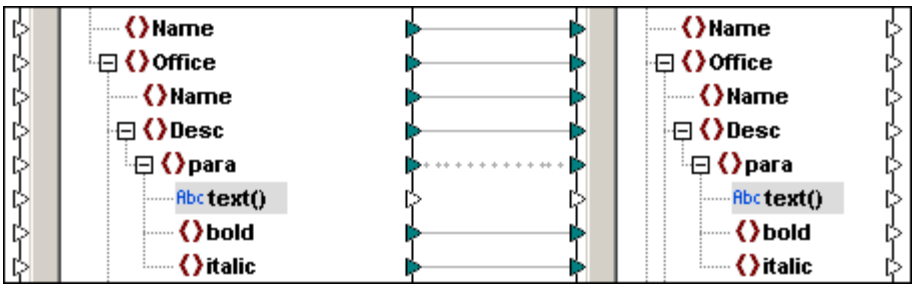
Der Mixed Content-Text jeder Beschreibung (Desc) wurde korrekt gemappt; sowohl der Text als auch der Inhalt der Bold- und Italic-Tags wurde so gemappt, wie er in der XML-Quelldatei aufscheint.

6. Wechseln Sie zurück zur Mapping-Ansicht.

Entfernen von Text Nodes aus Mixed Content-Elementen:

1. Klicken Sie auf den **text()** Node-Konnektor und drücken Sie die Entf-Taste um ihn zu löschen.

© 2011 Altova GmbH



2. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis des Mappings zu sehen.

```
5     <Name>Nanonull, Inc.</Name>
6     <Desc>
7       <para>
8         <bold> Vereno</bold>
9         <italic>multi-core processors.</italic>
10        <bold>Nano-grid.</bold>
11        <italic>offshore</italic>
12      </para>
13      <para/>
14    </Desc>
15  </Office>
16  <Office>
17    <Name>Nanonull Europe, AG</Name>
18    <Desc>
19      <para>
20        <italic>Europe</italic>
21        <bold> five research scientists </bold>
22      </para>
23    </Desc>
```

Ergebnis:

- Alle **Text** Nodes des Elements "para" wurden entfernt.
- Gemappter bold- und italic-Textinhalt wird beibehalten
- Die **Reihenfolge** der Datenelemente "bold" und "italic" entspricht weiterhin der in der XML-Quelldatei!

Mixed Content-Beispiel

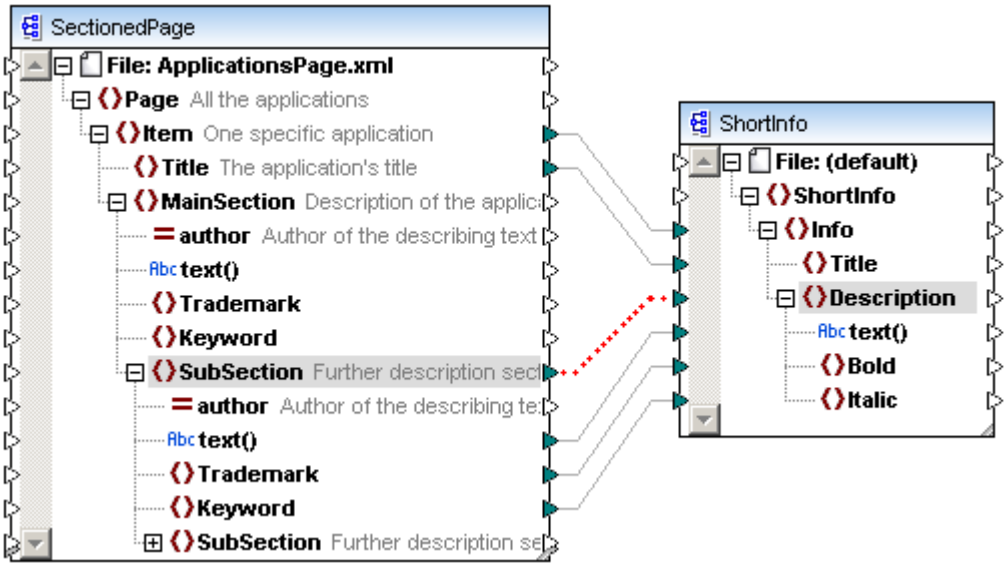
Das folgende Beispiel finden Sie im Ordner **...\MapForceExamples** unter "**ShortApplicationInfo.mfd**".

Unten sehen Sie ein Fragment der XML-Quelldatei für dieses Beispiel.

```
<Page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="SectionedPage.xsd">
    <Item>
        <Title>XMLSpy</Title>
        <MainSection author="altova">
            Altova <Trademark>XMLSpy</Trademark>
            <SubSection>Altova <Trademark>XMLSpy</Trademark> 2005 Enter
is the industry standard <Keyword>XML</Keyword> development environment
editing, debugging and transforming all <Keyword>XML</Keyword> technolo
automatically generating runtime code in multiple programming languages
        </MainSection>
    </Item>
```

In der nachfolgenden Abbildung sehen Sie das Mapping. Bitte beachten Sie:

- Der Konnektor für das Datenelement "SubSection" ist vom Typ "Mixed Content" und wird auf das Datenelement "Description" im XML/Schema-Zieldokument gemappt.
- Die text() Nodes werden in der Zielkomponente auf das Datenelement "Bold" gemappt.
- Der Trademark-Text wird auf das Datenelement "Bold" im Zieldokument gemappt.
- Der Keyword-Text wird auf das Datenelement "Italic" im Zieldokument gemappt.



Ergebnis des Mappings:

- Der Mixed Content-Text der einzelnen Beschreibungen wurde korrekt gemappt; sowohl der Text als auch der Inhalt der Tags "Bold" und "Italic" wurde so gemappt, wie er im XML-Quelldokument aufscheint.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ShortInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="
   C:/PROGRA~1/Altova/MapForce2005/MapForceExamples/ShortInfo.xsd">
3    <Info>
4      <Title>XMLSpy</Title>
5      <Description>Altova <Bold>XMLSpy</Bold> 2005 Enterprise Edition is the industry standard
   <Italic>XML</Italic> development environment for modeling, editing, debugging and transforming
   all <Italic>XML</Italic> technologies, then automatically generating runtime code in multiple
   programming languages.</Description>
6    </Info>
```

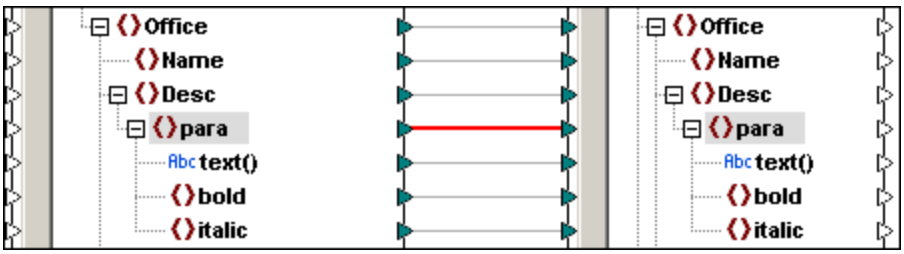
Mappen von Mixed Content-Datenelementen mit Hilfe eines quellorientierten Mappings

In diesem Abschnitt werden die Ergebnisse beschrieben, die Sie bei Definition eines Standard-Mappings (oder bei Verwendung von Standard-Konnektoren) für **Mixed Content** -Datenelemente erhalten. Sie finden die im folgenden Beispiel (**Tut-OrgChart.mfd**) verwendeten Dateien im Ordner ...\MapForceExamples\Tutorial\.

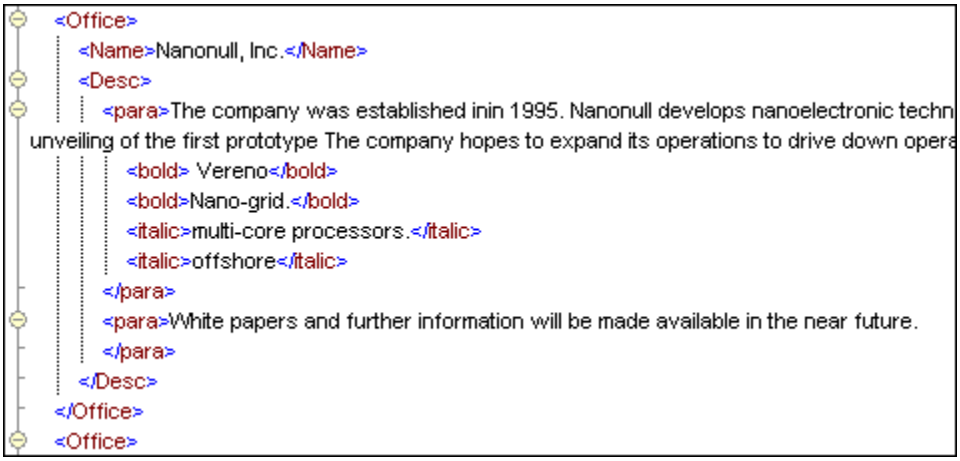
Erstellen von Standardverbindungen zwischen Mixed Content-Datenelementen:

1. Erstellen Sie einen Konnektor zwischen den zwei **para**-Elementen. Daraufhin wird eine Meldung angezeigt, in der Sie gefragt werden, ob MapForce die Konnektoren als quellorientiert erstellen soll.
2. Klicken Sie auf "Nein" um ein Standardmapping zu erstellen.

© 2011 Altova GmbH



3.. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis des Mappings zu sehen.



Ergebnis:
Ein Mapping von Mixed Content-Datenelementen mit einem Standard-Mapping hat das folgende Ergebnis:

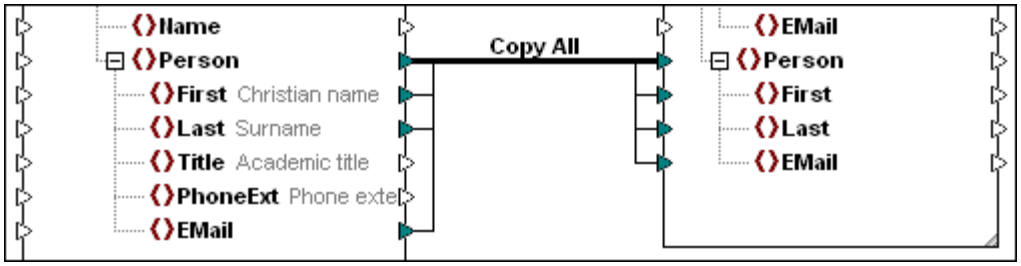
- Text() **Inhalt** wird unterstützt/gemappt
- Die Start-/End-Tags der Child-Nodes, bold und italic werden aus dem text-Node entfernt.
- Die Child Nodes erscheinen nach dem Mixed Content Node text.
- Die **Reihenfolge** der Child Nodes ist von der Reihenfolge der XML/Schema-**Zieldatei** abhängig

d.h.
Für jedes **para-**Element werden zuerst der text() Node, dann **alle bold**-Datenelemente und anschließend **alle italic**-Datenelemente gemappt. Daraus resultiert die oben gezeigte Reihenfolge: bold, bold - italic, italic. Der Inhalt jedes Datenelements wird gemappt, wenn ein Konnektor vorhanden ist.

6.1.3 "Alles kopieren" Konnektoren

Bei dieser Verbindungsart können Sie automatisch **alle** identischen Datenelemente in Quell- und Zielkomponente miteinander verbinden. Dies bedeutet, dass je nach Quell und Ziel**typ** alle Quelldatenelemente in die Zielkomponente **kopiert** werden, wenn der Quell- und Ziel**typ** entweder identisch sind oder der Zieltyp: xs:anyType ist

Wenn der Quell- und Zieltyp **nicht identisch** sind und der Zieltyp nicht vom Typ xs:anyType ist, werden die Quelldaten auf die entsprechenden Datenelemente desselben Namens und derselben Hierarchie gemappt. Wenn die Namen der Zieldatenelemente andere sind, wird kein Mapping zum Zieldatenelement erstellt.



Konnektoren vom Typ "Alles kopieren" werden durch eine einzige fett gedruckte Linie, die die verschiedenen identischen Datenelemente der Quell- und Zielkomponente miteinander verbindet, gekennzeichnet.

Beachten Sie, dass nur verglichen wird, ob die Namen der Sub-Einträge übereinstimmen, nicht aber ob deren jeweiliger Typ identisch ist.

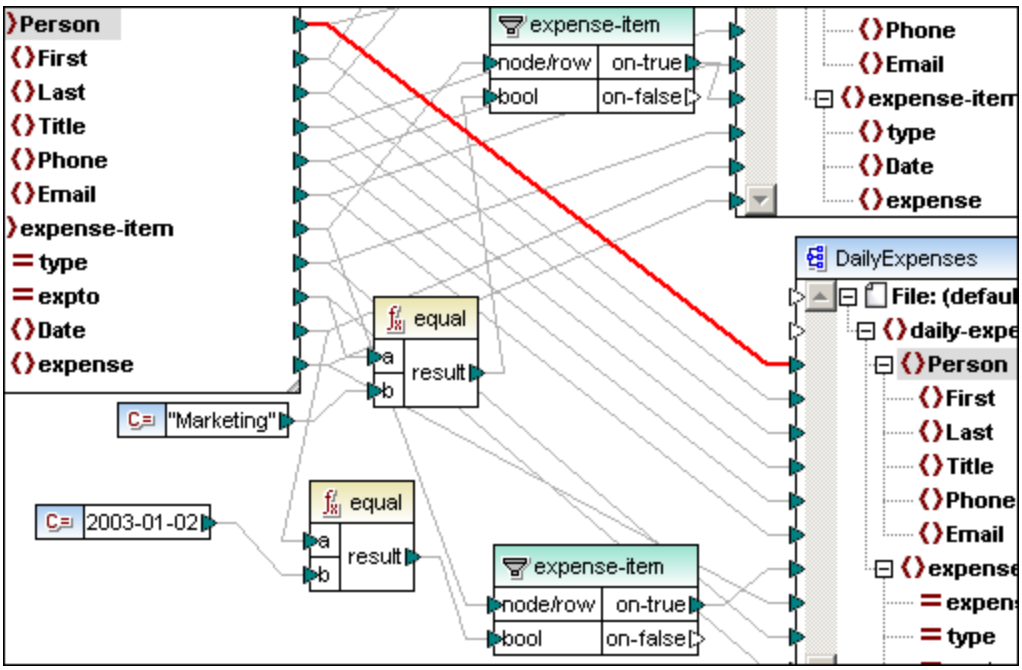
Derzeit werden die folgenden "Alles kopieren" Verbindungen unterstützt: (i) zwischen complexTypes eines XML Schemas und (ii) zwischen komplexen Komponenten (XML-Schema) und komplexen benutzerdefinierten Funktionen/Komponenten, die dieselben komplexen Parameter enthalten.

Im Beispiel unten sehen Sie diese Konnektoren in der Datei **MarketingAndDailyExpenses.mfd** aus dem Ordner **...\MapForceExamples**.

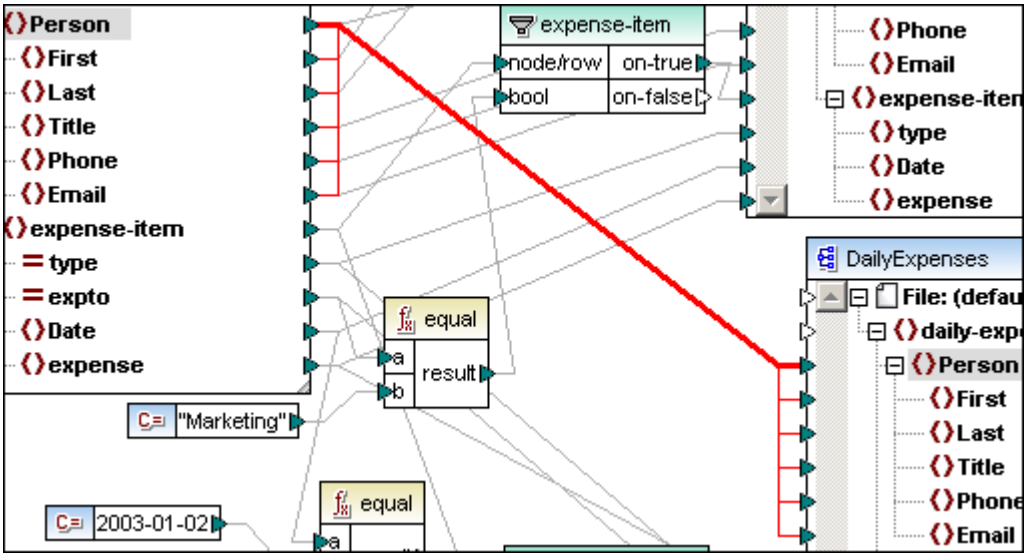
So definieren Sie eine "Alles kopieren"-Verbindung:

1. Rechtsklicken Sie auf einen vorhandenen Konnektor, z.B. auf den "Person" Konnektor und wählen Sie im Kontextmenü die Option "Alles kopieren".
   Es erscheint eine Meldung, die besagt, dass alle Verbindungen zu Sub-Einträgen der Zielkomponente durch die "Alles kopieren"-Verbindung ersetzt werden.

© 2011 Altova GmbH



2. Klicken Sie auf OK, um "Alles kopieren" Konnektoren zu erstellen.



Es werden nun alle Konnektoren zur Zielkomponente und alle Verbindungen zwischen Quell- und Ziel-Datenelementen mit demselben Namen erstellt.

Bitte beachten Sie:

- Wenn die bestehenden Ziel-Verbindungen gelöscht werden, werden Konnektoren von anderen Quellkomponenten oder anderen Funktionen ebenfalls gelöscht.
- Diese Art von Verbindung kann nicht zwischen einem Datenelement und dem Root-Element einer Schema-Komponente erzeugt werden.
- Nachdem Sie diese Methode verwendet haben, können einzelne Konnektoren aus der "Alles kopieren" Gruppe nicht mehr gelöscht oder neu verbunden werden.

Auflösen/Löschen von "Alles kopieren" Konnektoren:

1. Verbinden Sie ein beliebiges Datenelement mit einem Sub-Eintrag der "Alles

kopieren"-Verbindung in der Zielkomponente.
Sie werden darauf aufmerksam gemacht, dass im Zieldatenelement nur ein Konnektor vorhanden sein darf. Klicken Sie auf "Ersetzen", um den Konnektor zu ersetzen.

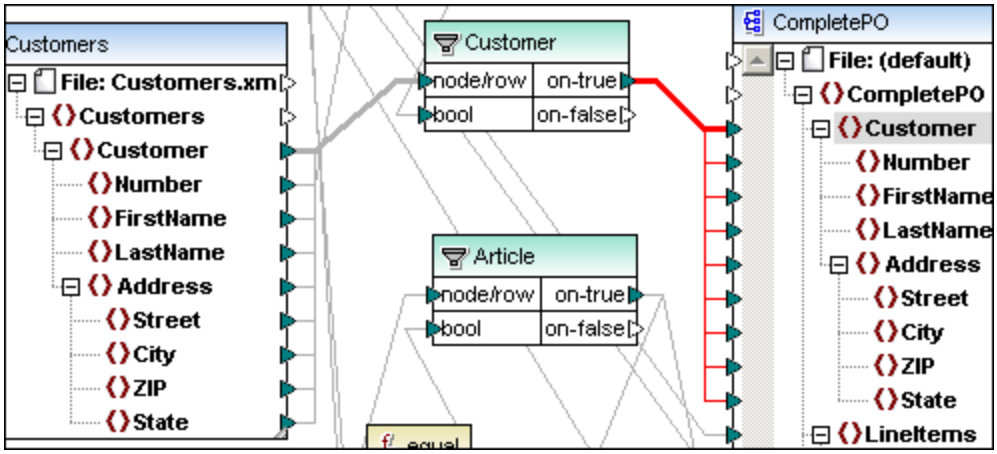
2. Klicken Sie im nächsten Meldungsfeld auf die Schaltfläche "Alles kopieren-Verbindung auflösen".
   Die "Alles kopieren"-Verbindung wird durch einzelne Konnektoren zu den Zielkomponenten ersetzt.

"Alles kopieren" Verbindungen und benutzerdefinierte Funktionen
Beim Erstellen von "Alles kopieren" Verbindungen zwischen einem Schema und einem benutzerdefinierten Funktionsparameter müssen die beiden Komponenten auf demselben Schema basieren! Sie müssen jedoch nicht notwendigerweise dasselbe Root-Element haben. Ein Beispiel dazu finden Sie unter "Definieren komplexer Output-Komponenten".

"Alles kopieren"-Verbindungen und Filter
"Alles kopieren"-Verbindungen können auch über Filterkomponenten erstellt werden, wenn die Quellkomponente:

- aus strukturierten Daten besteht, d.h. aus einer Schema-Komponente.
- Daten über einen komplexen Ausgabeparameter einer benutzerdefinierten Funktion oder eines Webservice erhält.
- Daten über eine weitere Filterkomponente erhält.

  Nur die gefilterten Daten werden an die Zielkomponente übergeben.



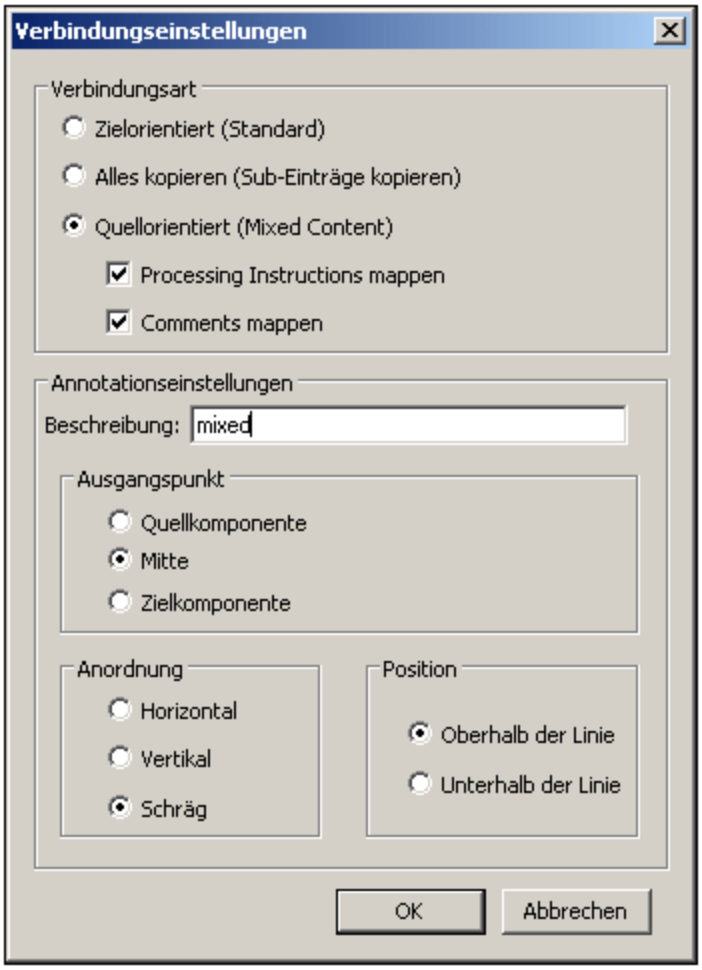
So definieren Sie eine "Alles kopieren"-Verbindung über eine Filterkomponente:

1. Erstellen Sie einen Konnektor vom **on-true**/on-false-Datenelement zum Zieldatenelement, z.B. Customer.
2. Klicken Sie mit der rechten Maustaste auf den Konnektor und wählen Sie im Kontextmenü den Eintrag "Alles kopieren (Sub-Einträge kopieren)".
   Der "Alles kopieren" Konnektor zwischen Datenelementen desselben Namens wird erstellt.

© 2011 Altova GmbH

6.2 Verbindungseinstellungen

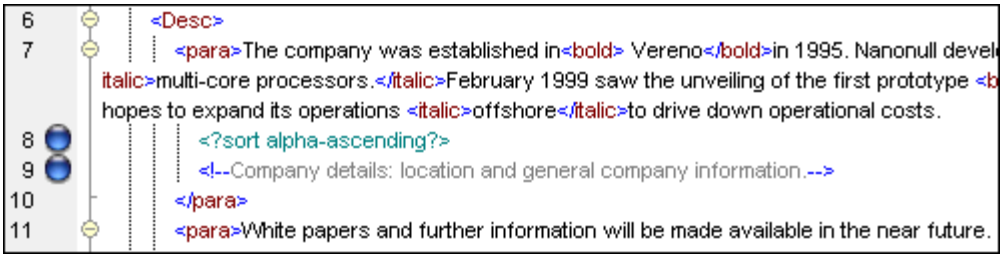
Durch Rechtsklick auf einen Konnektor und Auswahl des Befehls **Eigenschaften** aus dem Kontextmenü oder durch Doppelklick auf einen Konnektor wird das Dialogfeld **Verbindungseinstellungen** geöffnet, in welchem Sie die spezifischen Einstellungen (mixed content) für den aktuellen Konnektor festlegen können. Beachten Sie, dass nicht verfügbare Optionen ausgegraut sind.



Verbindungstyp
Bei Datenelementen vom Typ **complexType** können Sie für das Mapping eine der folgenden Verbindungsarten auswählen (Bitte beachten Sie, dass diese Einstellungen auch für **complexType-**Datenelemente gelten, die keine Text Nodes enthalten!):

- **Zielorientiert (Standard):** Ändert die Verbindungsart in Standard-Mapping. Nähere Informationen dazu siehe: "Zielorientiert / Standard-Mapping".
- **Alles kopieren (Sub-Einträge kopieren):** Ändert die Verbindungsart in "Alles kopieren" und verbindet automatisch alle identischen Datenelemente aus der Quell- und Zielkomponente. Nähere Informationen dazu finden Sie unter "Alles kopieren"-Verbindungen.
- **Quellorientiert (Mixed Content):** Ändert die Verbindungsart in quellorientiertes / Mixed Content Mapping, sodass eine Auswahl zusätzlicher Elemente gemappt werden kann. Die zusätzlichen Elemente müssen **Child-Elemente** des gemappten Datenelements in der XML-Quelldatei sein, damit sie gemappt werden können. Wenn Sie die

Kontrollkästchen **Processing Instructions mappen** und/oder **Comments mappen** aktivieren, werden diese Daten in die Ausgabedatei inkludiert.



Bitte beachten Sie: CDATA-Abschnitte werden als Text behandelt.

Annotationseinstellungen

Sie können die einzelnen Konnektoren mit einer Beschriftung versehen, um Ihr Mapping ausführlich zu kommentieren. Wenn Sie einen Buchstaben in das Feld "Beschreibung" eingeben, werden auch die Gruppenbereiche "Ausgangspunkt", "Anordnung" und "Position" aktiv und können bearbeitet werden. Diese Option steht für **alle Verbindungstypen** zur Verfügung.

So fügen Sie eine Annotation zu einem Konnektor hinzu:

1. Geben Sie den Namen des gerade ausgewählten Konnektors in das Feld **Beschreibung** ein.
   Damit werden alle Optionen in der Gruppe "Annotationseinstellungen" aktiviert.
2. In den restlichen Bereichen können Sie den **Ausgangspunkt, die Ausrichtung und Position** der Beschriftung definieren.
3. Aktivieren Sie in der Symbolleiste die Schaltfläche "Annotationen anzeigen" um den Annotationstext zu sehen.

Anmerkung: Wenn die Schaltfläche "Annotationen anzeigen" deaktiviert ist, können Sie den Annotationstext dennoch sehen, wenn Sie den Mauszeiger über den Konnektor platzieren. Daraufhin wird der Text in einem Popup-Fenster angezeigt, wenn die Schaltfläche **Tipps anzeigen** in der Symbolleiste "Ansichtsoptionen" aktiv ist.

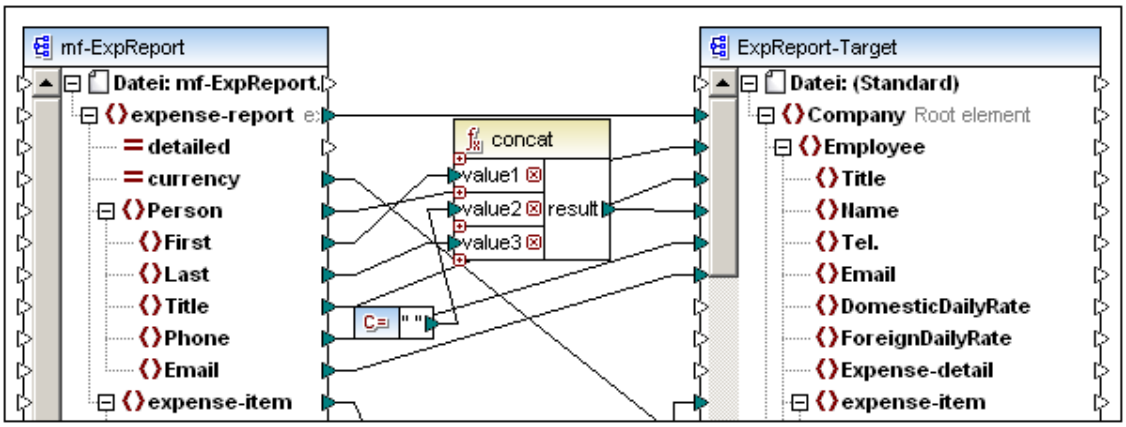
© 2011 Altova GmbH

6.3 Verschieben von Konnektoren

Verschieben von Konnektoren und die Auswirkung auf Child-Konnektoren
Wenn Sie einen Parent-Konnektor auf einen anderen Parent-Konnektor verschieben, findet MapForce automatisch die identen Child-Verbindungen unter dem neuen Konnektor. Diese Funktion ist nicht mit der Option "Idente Sub-Einträge automatisch verbinden" identisch, da das Ergebnis mit Hilfe anderer Regeln erzielt wird.

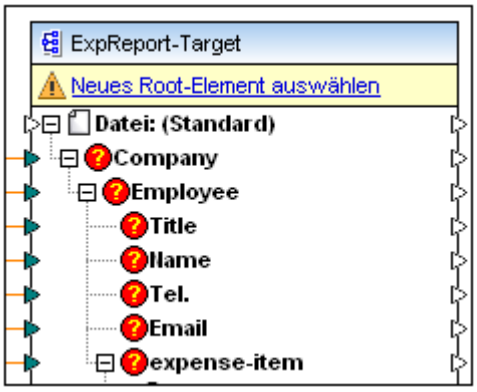
Häufig wird diese Funktion verwendet, wenn Sie das Root-Element des Zielschemas in einem bestehenden Mapping ändern. Normalerweise müssten Sie in diesem Fall alle untergeordneten Konnektoren manuell mappen

In diesem Beispiel wird die Datei **Tut-ExpReport.mfd** aus dem Ordner ...\MapForceExamples\Tutorial verwendet.

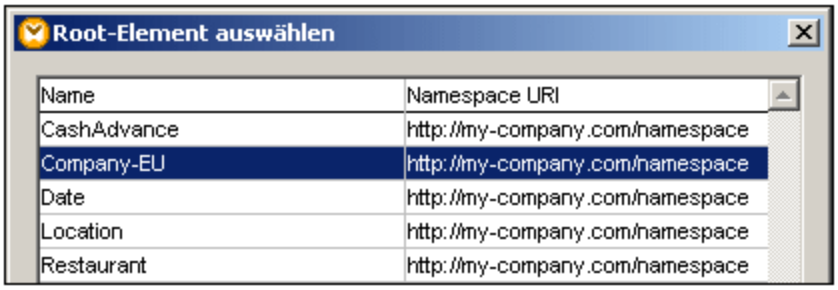


Wenn das Company Root-Element des Zielschemas in "Company-EU" geändert wird, wird in MapForce die Meldung "Geänderte Dateien" angezeigt.

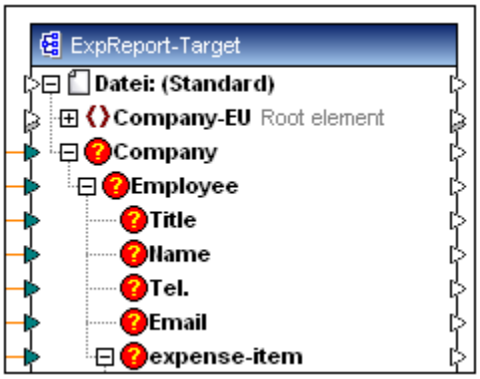
1. Klicken Sie auf die Schaltfläche **Neu laden**, um das aktualisierte Schema neu zu laden. Sie sehen nun, dass mehrere Nodes fehlen, da sich das Root-Element geändert hat.



2. Klicken Sie oben in der Komponente auf den Link "Neues Root-Element auswählen".

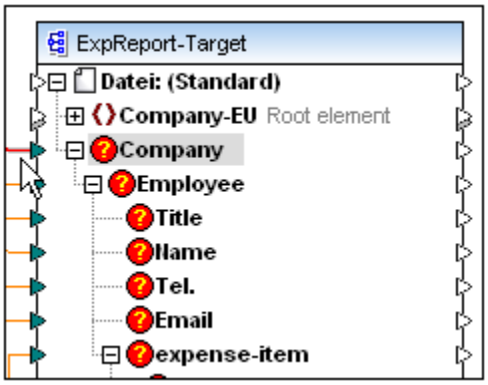


3. Wählen Sie das aktualisierte Root-Element Company-EU aus und klicken Sie zur Bestätigung auf OK.

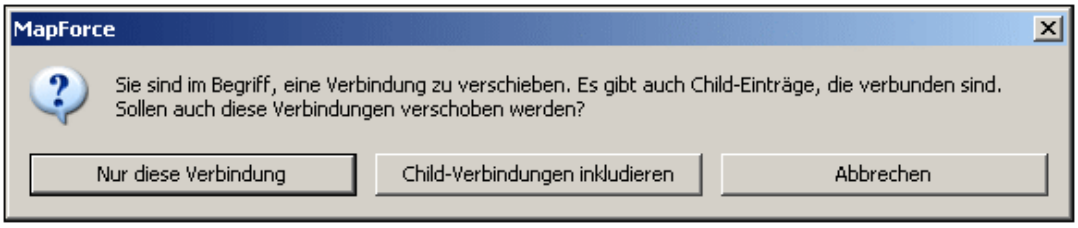


Das Root-Element "Company-EU" wird nun am oberen Rand der Komponente angezeigt.

4. Klicken Sie auf den Konnektor des Datenelements "Company" und ziehen Sie ihn mit der Maus auf das neue Root-Element "Company-EU".



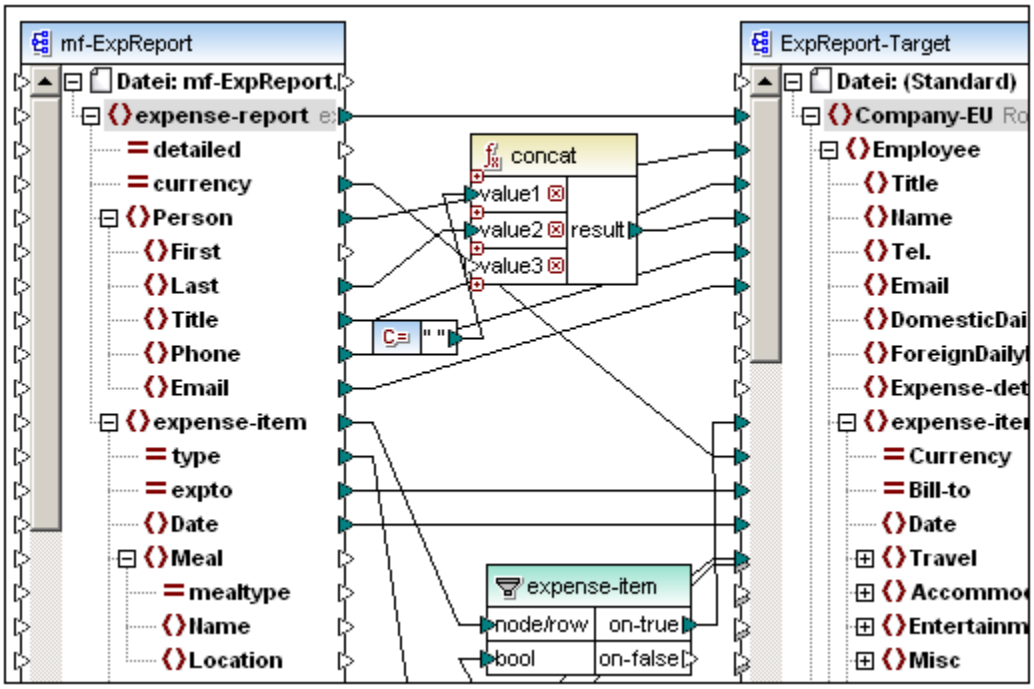
Daraufhin erscheint eine Meldung, in der Sie gefragt werden, welche Konnektoren verschoben werden sollen.



5. Klicken Sie auf die Schaltfläche "Child-Verbindungen inkludieren", wenn die

 © 2011 Altova GmbH

Child-Konnektoren gemappt werden sollen.
Die Nodes für die "fehlenden Datenelemente" wurden entfernt und alle Konnektoren wurden auf die richtigen Child-Elemente unter dem neuen Root-Element gemappt.



Bitte beachten Sie:

Wenn der Ziel-Node/das Zieldatenelement denselben Namen wie der Quell-Node hat, sich aber in einem anderen Namespace befindet, so enthält die Meldung eine zusätzliche Schaltfläche "Child-Verbindungen inkludieren und Namespace mappen".

Wenn Sie auf diese Schaltfläche klicken, werden die Child-Konnektoren desselben Namespace wie die des Parent-Node der Quellkomponente auf dieselben Child-Nodes unter dem Node des anderen Namespace verschoben. Wenn sich die Parent-Nodes also nur im Namespace unterscheiden, dann dürfen sich auch die Child-Nodes in dieser Hinsicht voneinander unterscheiden und können automatisch gemappt werden.

Sie können das Root-Element auch ändern, indem Sie auf die Komponentenüberschrift klicken und im Kontextmenü den Befehl "Root-Element ändern" auswählen.

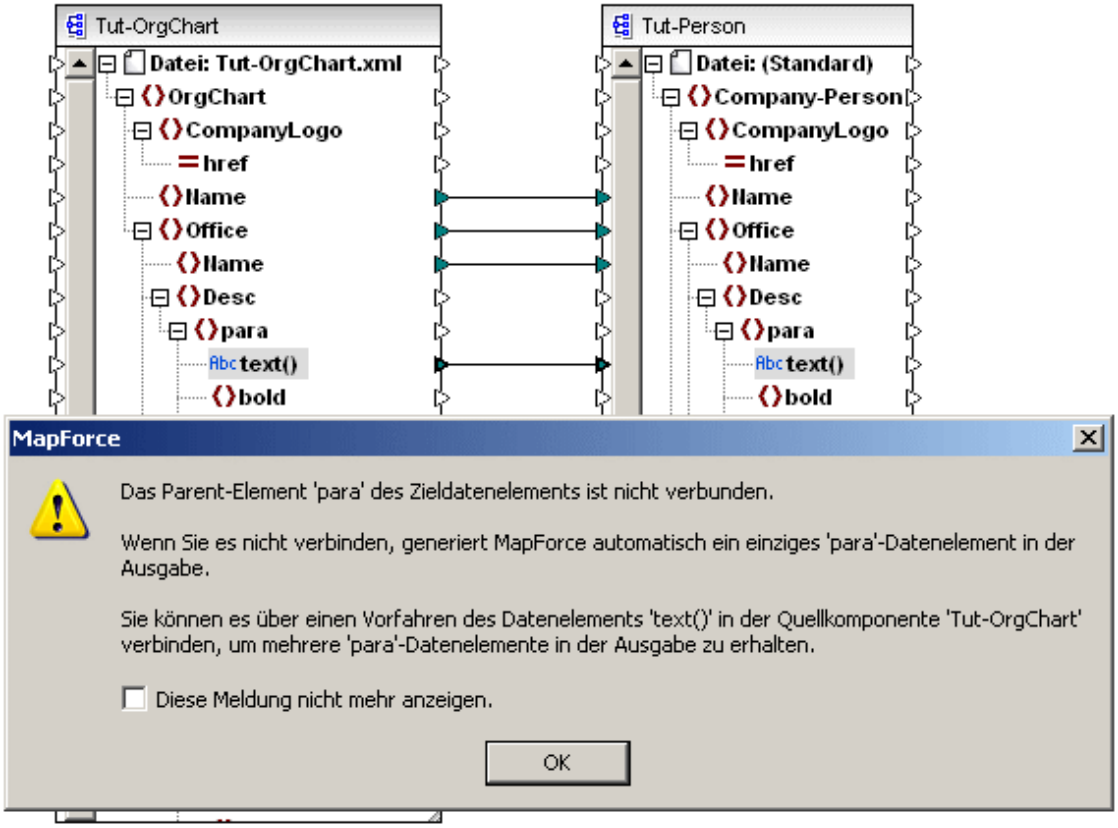
6.4 Verbindungen und Mapping-Ergebnisse

Wenn Sie manuell eine Verbindung zwischen einem Quell- und einem Zieldatenelement erstellen, analysiert MapForce automatisch die möglichen Ergebnisse. Wenn Sie zwei Subelemente mappen, wird eine Meldung angezeigt, in der vorgeschlagen wird, auch das übergeordnete Quelldatenelement mit dem übergeordnete Zieldatenelement zu verbinden.

Dadurch verhindern Sie, dass in der Mapping-Vorschau nur ein einziges Subelement im Ausgabefenster angezeigt wird. Im Allgemeinen ist dies der Fall, wenn der Quell-Node eine Sequenz anstatt eines einzelnen Werts liefert.

Das unten gezeigte Mapping Tut-OrgChart.mfd steht im Ordner ...\MapForceExamples\Tutorial zur Verfügung.

Wenn Sie das Quelldatenelement text() mit dem Zieldatenelement text() verbinden, erscheint ein Meldungsfeld, in dem Sie darauf aufmerksam gemacht werden, dass das übergeordnete Zieldatenelement "para" nicht verbunden ist und nur einmal in der Ausgabe generiert wird. Um mehrere para-Elemente in der Zielkomponente zu generieren, verbinden Sie die das para-Element in der Quellkomponente mit dem para-Element in der Zielkomponente.

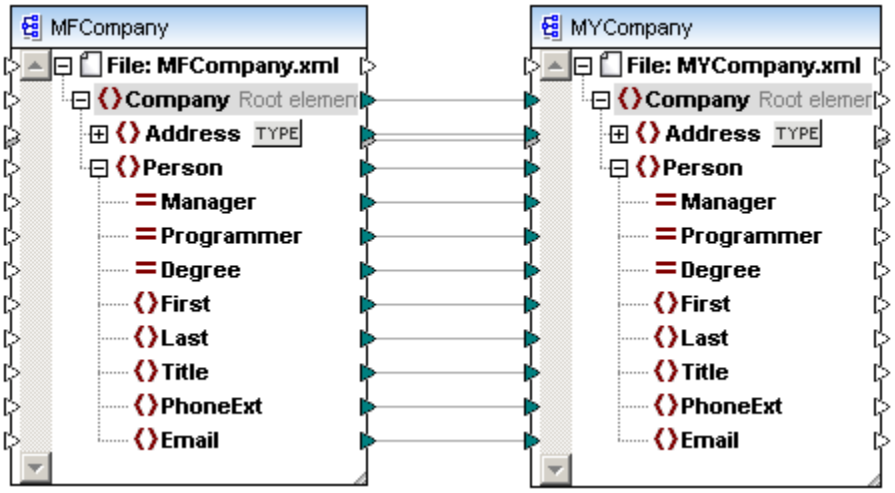


 © 2011 Altova GmbH

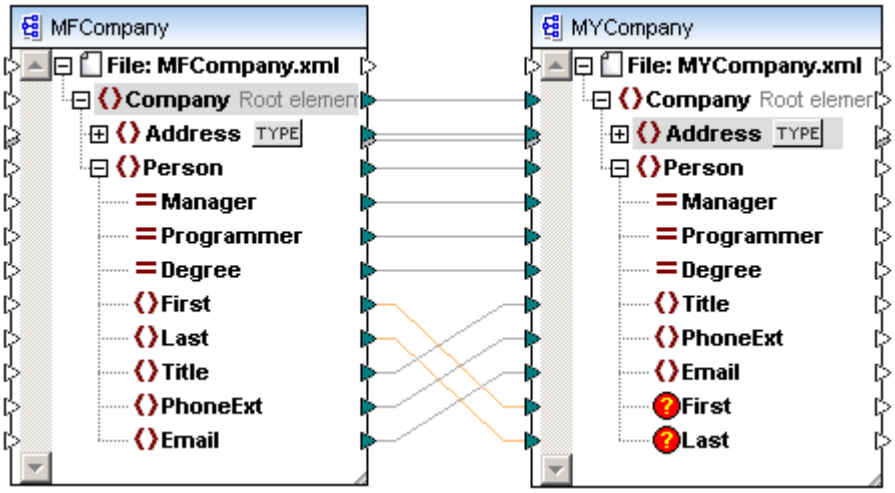
6.5 Fehlende Datenelemente

Oft kommt es vor, dass sich die Struktur einer der Komponenten in einem Mapping ändert, dass z.B. Attribute oder Elemente zu einem Schema hinzugefügt oder daraus gelöscht werden. In MapForce werden nun Platzhalter-Datenelemente verwendet, um alle Konnektoren sowie alle relevanten Verbindungsdaten zwischen Komponenten beizubehalten, wenn Datenelemente gelöscht wurden.

Beispiel:
In diesem Beispiel wird die Schemadatei **MFCompany.xsd** verwendet. Das Schema wird in MyCompany.xsd umbenannt und es wird ein Konnektor zwischen den beiden Company-Datenelementen in den beiden Schemas angelegt. Daraufhin werden Konnektoren für alle Child-Datenelemente zwischen den Komponenten erstellt, wenn der Befehl "Idente Sub-Einträge automatisch verbinden" aktiv ist.



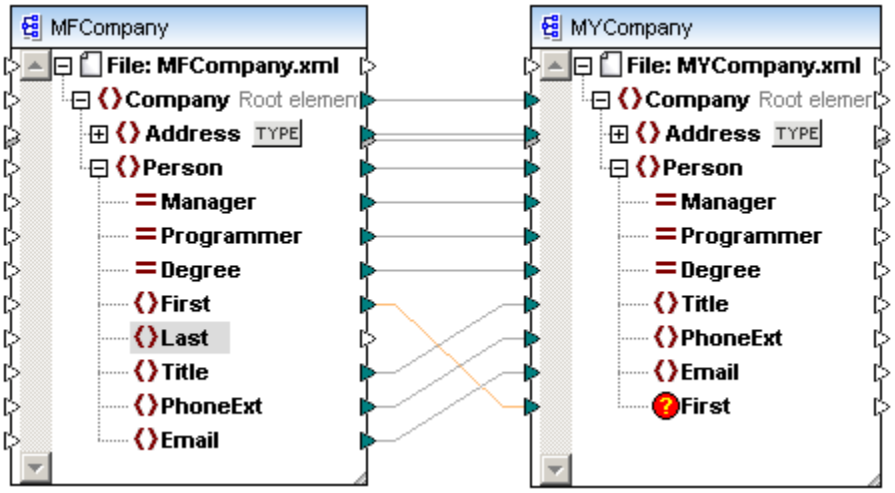
Während der Bearbeitung von MyCompany.xsd in XMLSpy werden die Elemente "First" und "Last" im Schema gelöscht. Wenn Sie wieder zu MapForcewechseln, wird ein Dialogfeld angezeigt, dass Dateien geändert wurden und Sie werden aufgefordert, das Schema neu zu laden. Wenn Sie auf "Neu laden" klicken, werden die Komponenten in MapForce aktualisiert.



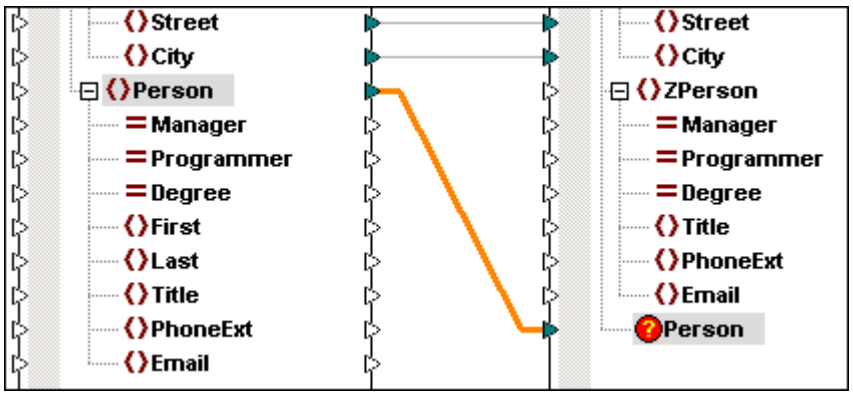
Die gelöschten **Datenelemente** und ihre **Konnektoren** erscheinen in der Komponente "MyCompany" nun markiert. Sie können die Konnektoren nun gegebenenfalls mit anderen Datenelementen verbinden oder die Konnektoren löschen.

Beachten Sie, dass Sie immer noch eine Vorschau des Mappings anzeigen (oder Code generieren) können, dass aber im Fenster "Meldungen" Warnmeldungen angezeigt werden, falls Sie dies zu diesem Zeitpunkt tun. Alle Verbindungen zu oder von fehlenden Datenelementen werden bei der Vorschau oder Codegenerierung ignoriert.

Wenn Sie auf einen der markierten Konnektoren klicken und ihn löschen, wird das "fehlende" Datenelement - z.B. "Last" in "MyCompany" aus der Komponente gelöscht.

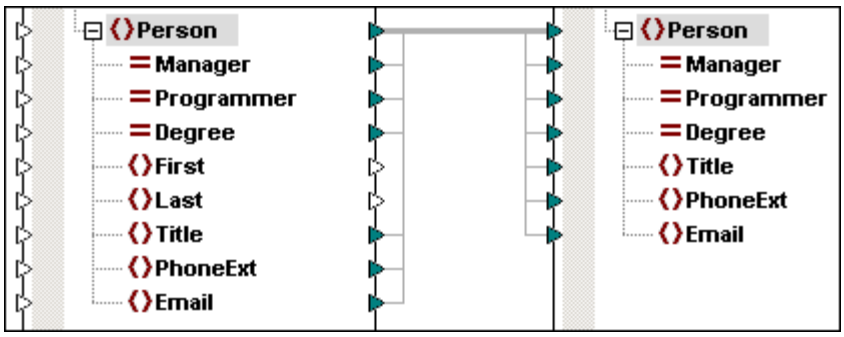


Umbenannte Datenelemente:
Wenn ein Parent-Datenelement umbenannt wird, z.B. Person in ZPerson, so wird der Original-Parent-Datenelement-Konnektor beibehalten und die Child-Elemente und deren Konnektoren werden gelöscht.

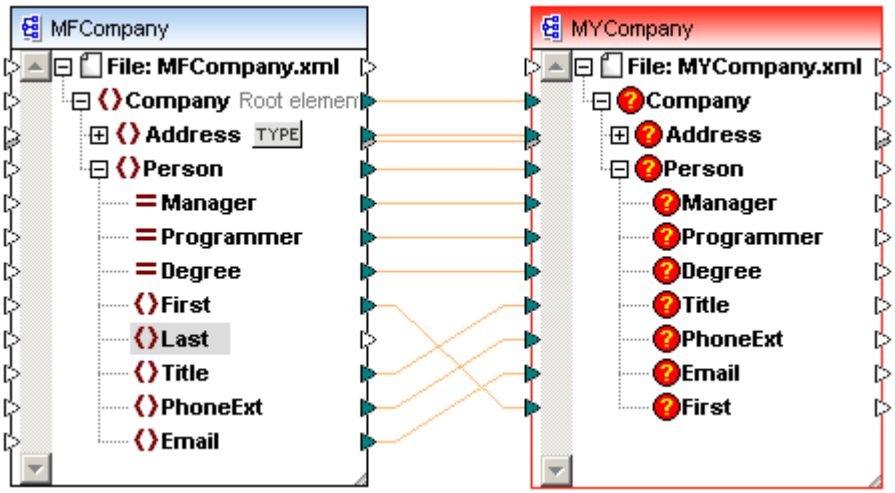


"Alles kopieren"-Konnektoren und fehlende Datenelemente:
"Alles kopieren"-Verbindungen werden genau wie normale Verbindungen behandelt, wobei der einzige Unterschied darin besteht, dass die Verbindungen zu den fehlenden Subeinträgen nicht beibehalten oder angezeigt werden.

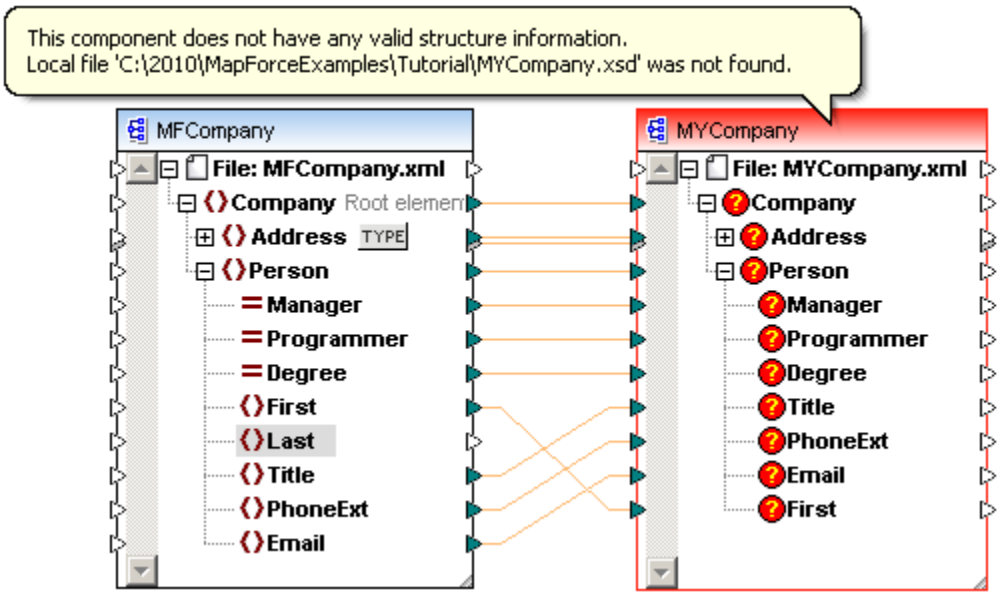
© 2011 Altova GmbH



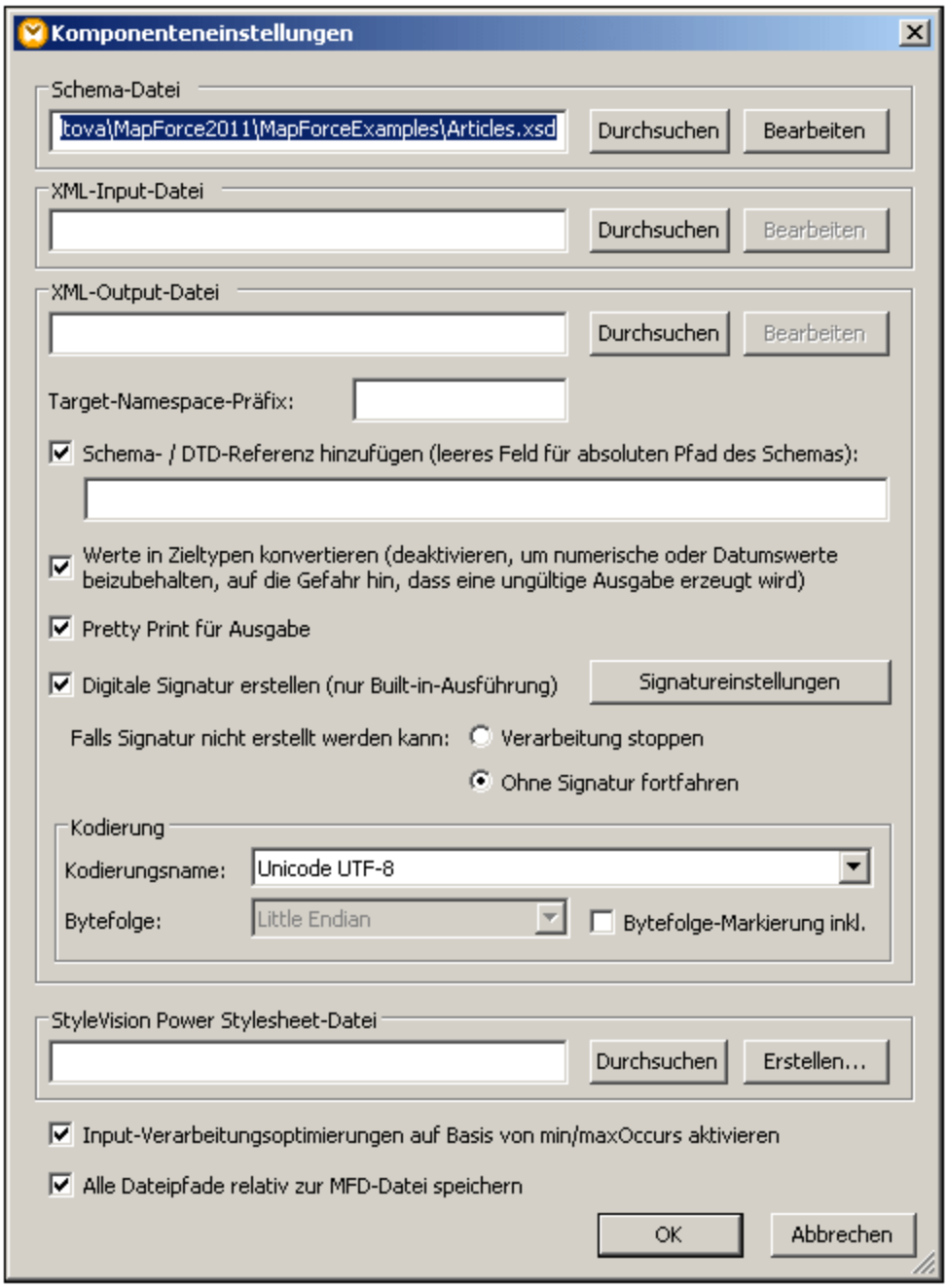
Umbenannte oder gelöschte Komponentenquellen:
Wenn die **Datenquelle** einer Komponente, d.h. ein Schema. umbenannt oder gelöscht wurde, werden alle darin enthaltenen Datenelemente markiert. Der rote Rahmen um die Komponente zeigt an, dass es keine gültige Verbindung zu einem Schema gibt. Es kann keine Vorschau und kein Code generiert werden.



Wenn Sie den Mauszeiger über die markierte Komponente platzieren, wird ein Popup-Fenster mit den dazugehörigen Informationen angezeigt.



Wenn Sie auf die Titelleiste der markierten Komponente doppelklicken, wird das Dialogfeld "Komponenteneinstellungen" geöffnet. Wenn Sie auf die Durchsuchen-Schaltfläche in der Gruppe **Schema-Datei** klicken, können Sie eine andere Version oder eine Sicherungskopie des Schemas öffnen. Nähere Informationen dazu finden Sie unter "Komponente" in Abschnitt "Referenz".



Alle gültigen/korrekten Verbindungen werden beibehalten, wenn Sie ein Schema derselben Struktur auswählen.

 © 2011 Altova GmbH

6.6 Verkettete Mappings / Weiterleitungskomponenten

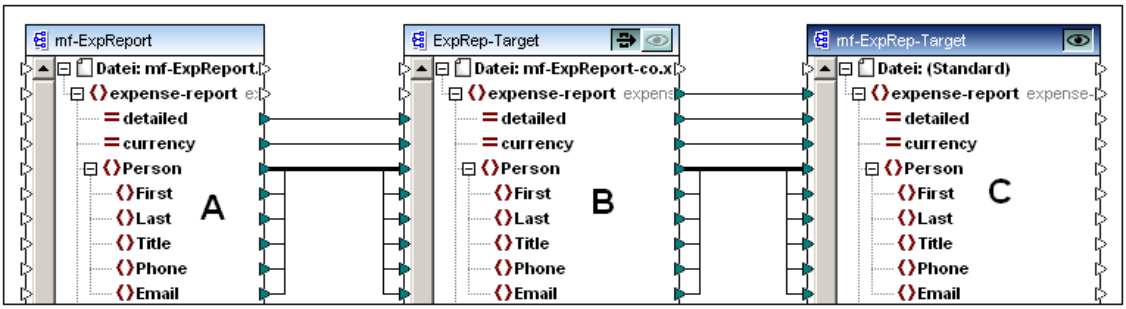
MapForce unterstützt Mappings, die aus mehreren Komponenten in einer Mapping-Kette bestehen. Verkettete Mappings sind Mappings, in denen mindestens eine Komponente gleichzeitig als Quell- und Zielkomponente fungiert. Mit Hilfe einer solchen Komponente wird eine Ausgabe erzeugt, die später als Input für den nächsten Mapping-Schritt in der Kette verwendet wird. Eine solche Komponente wird als "Zwischenkomponente" bezeichnet.

In verketteten Mappings wird eine Funktion, namens "Weiterleitung" verwendet. Mit Hilfe dieser Funktion können Zwischenresultate in Form von "Zwischenkomponenten" für die Vorschau, die Ausführung über die Befehlszeile und die Codegenerierung erzeugt werden. Die Weiterleitungsfunktion ist eine Vorschaufunktion, mit der Sie die verschiedenen Phasen eines verketteten Mappings im Fenster "Ausgabe" anzeigen können. Für die Anzeige der Vorschau wird der Built-in-Ausführungsprozessor verwendet. Die Ergebnisse werden in Form von temporären Dateien erzeugt.

Bei Ausführung des Mappings über die Befehlszeile bzw. bei der Generierung von Code, wird unabhängig davon, was in die Felder "XML-Input-Datei" und "XML-Output-Datei" der Zwischenkomponente eingegeben wurde, die gesamte Mapping-Kette ausgeführt und die Ausgabedaten eines vorherigen Mapping-Schritts werden als Eingabedaten für den nächsten Mapping-Schritt verwendet.

Anmerkung:
Die Funktion "Weiterleitung" steht nur für dateibasierte "Zwischenkomponenten", also XML-, CSV-, TXT-Dateien usw. 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.

In der Abbildung unten sehen Sie als Beispiel die drei Komponenten A, B und C, wobei C die Zielkomponente ist. Die Komponente B (ExpRep-Target) ist die "Zwischenkomponente", da sie sowohl Input- aus auch Output-Verbindungen aufweist.



Bei der Ausführung eines verketteten Mappings über die Befehlszeile oder bei Ausführung des generierten Codes führt das Mapping alle Schritte in der richtigen Reihenfolge aus und generiert die erforderlichen Ausgabedateien.

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 im Built-in-Ausführungsprozessor 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, da die Vorschau-Schaltfläche automatisch deaktiviert wird. Um die Ausgabe einer solchen Komponente zu sehen, klicken Sie auf die Schaltfläche "Weiterleitung" um sie zu deaktivieren und klicken Sie anschließend auf die Vorschau-Schaltfläche der Zwischenkomponente.

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 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-Schaltflächen (Komponente B oder C) aktiv sind:

- 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. Wenn 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.

  Beachten Sie Folgendes: Wenn das Mapping über die Befehlszeile oder anhand von generiertem Code ausgeführt wird, so wird, unabhängig von den Einstellungen der Schaltfläche "Weiterleitung" der Komponente B sowie von der gerade ausgewählten Vorschaukomponente, die Ausgabe aller Komponenten generiert.

  In unserem Beispiel werden zwei Ergebnisdateien generiert! Dies liegt daran, dass MapForce automatisch die Abhängigkeit aller Komponenten analysiert und alle Ausgabedateien der Zwischen- und der Endkomponente in der richtigen Reihenfolge generiert.

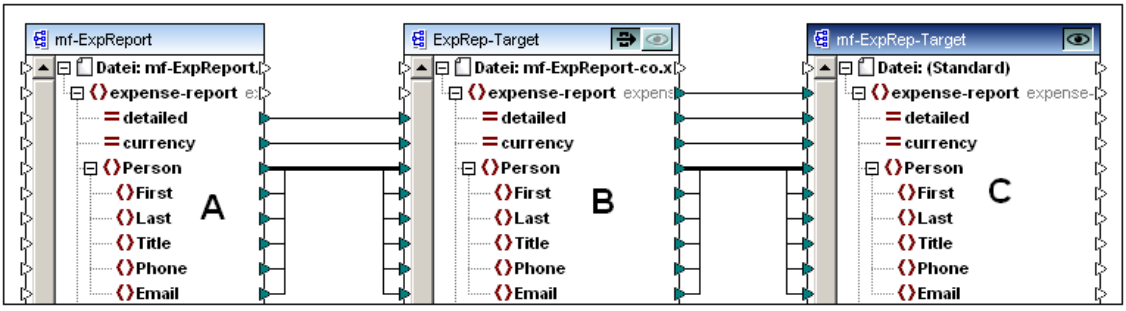
  Da die Einstellung "Weiterleitung" derzeit nicht aktiv ist, müssen die Dateinamen in den Feldern "XML-Input-Datei" und "XML-Output-Datei" der Zwischenkomponente unbedingt identisch sein.

Nähere Informationen zu diesem Beispiel und eine Beschreibung dazu, welche Unterschiede bei der Übertragung der Quelldaten bestehen, wenn die Schaltfläche "Weiterleitung" aktiv bzw. inaktiv ist, finden Sie in den folgenden Abschnitten. Ein Beispiel dazu finden Sie unter "Beispiel für ein verkettetes Mapping".

 © 2011 Altova GmbH

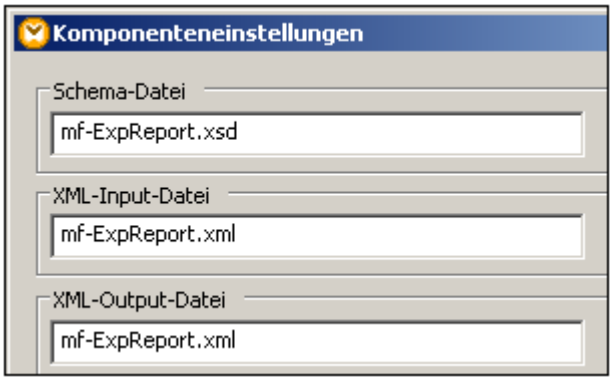
6.6.1 Verkettete Mappings - Weiterleitung aktiv

Die im folgenden Beispiel (**Tut-ExpReport-chain.mfd**) verwendeten Dateien stehen im Ordner **...\MapForceExamples\Tutorial** zur Verfügung.



Das Beispiel **Tut-ExpReport-chain.mfd** ist folgendermaßen eingerichtet:

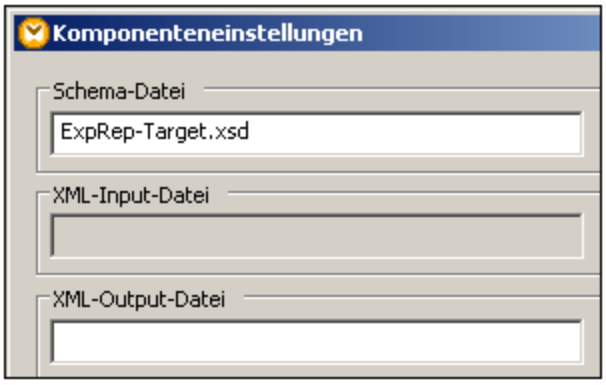
- **Die Komponente A** liefert anhand einer XML-Beispieldatei alle Mapping-Daten. Die XML-Datei (mf-ExpReport.xml) wird im Dialogfeld "**Komponenteneinstellungen**" im Feld "*XML-Input-Datei*" angezeigt. Die XML-Output-Datei desselben Namens wird automatisch eingefügt, wenn Sie eine XML-Input-Datei definieren.



- **Zwischenkomponente B** "Weiterleitung" ist aktiv:
  Wenn die Weiterleitung aktiviert ist, wird das Feld "*XML-Input-Datei*" der Zwischenkomponente automatisch deaktiviert. Es wird kein Dateiname benötigt, damit das Mapping ausgeführt werden kann, da die Zwischendaten in temporären Dateien gespeichert werden.

  Wenn keine XML-Output-Datei definiert wurde, wird ein Standarddateiname verwendet. Wenn im Feld "XML-Output-Datei" ein Dateiname definiert wurde, so wird dieser als Dateiname der Ausgabedatei der Zwischenkomponente verwendet.

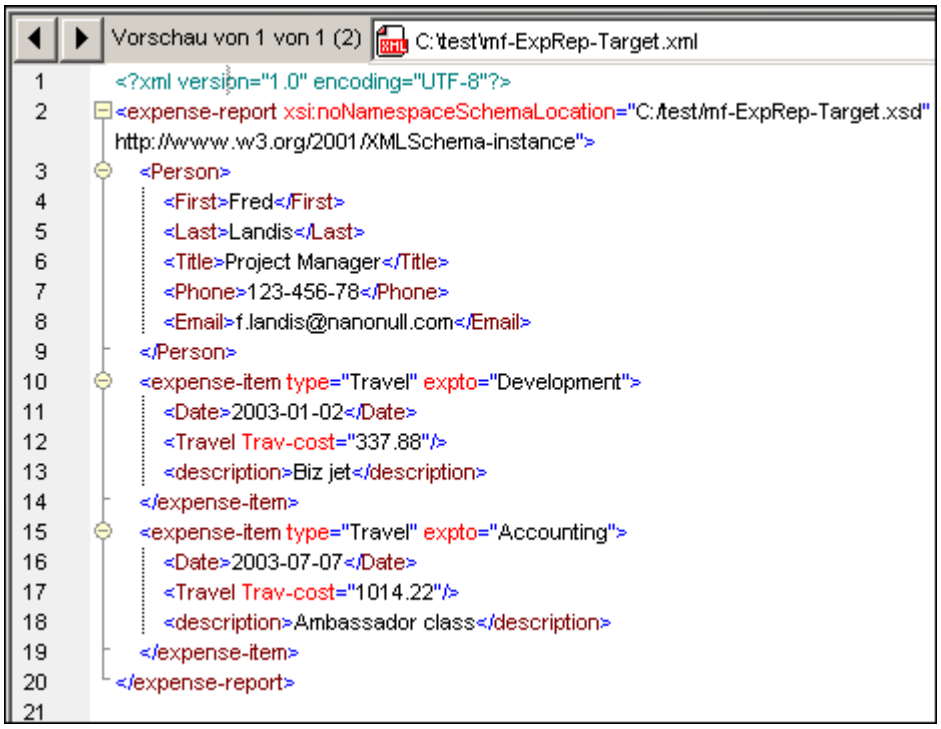
  Beachten Sie, dass Zwischenkomponenten auch dynamische Dateinamen, d.h. Konnektoren zum Eintrag "Datei:" einer Komponente (oder sogar Platzhalterzeichen für Dateinamen) haben können. Nähere Informationen dazu finden Sie unter Dynamische Input/Output-Dateien pro Komponente.



- Der Endkomponente C ist **keine** XML-Output-Datei zugewiesen. Die Vorschau-Schaltfläche der Komponente C ist aktiv.

Klicken Sie auf die Schaltfläche "Ausgabe", um eine Vorschau auf die Ergebnisse des Built-in-Ausführungsprozessors zu sehen.

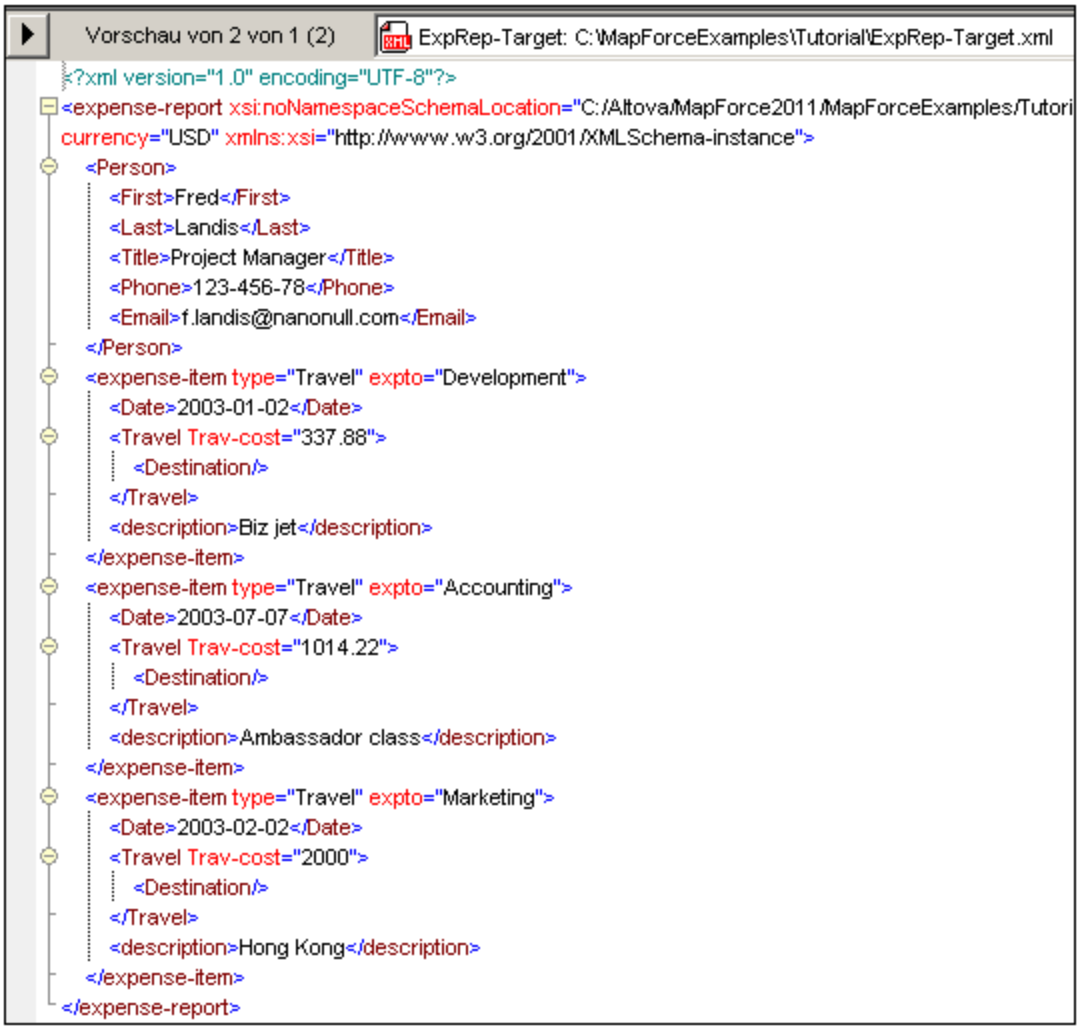
Vorschau 1:
Das Ergebnis des Mappings der Komponente A über die Zwischenkomponente B auf die Zielkomponente C. Dabei handelt es sich um alle Reisekosten unter 1500.



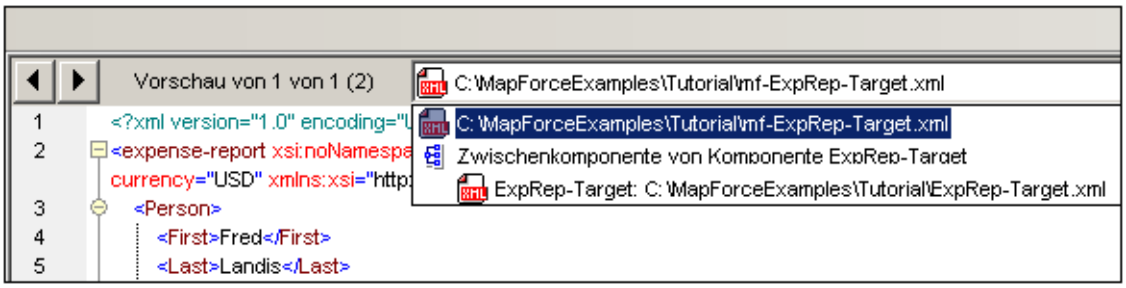
Vorschau 2:

Das Ergebnis des Mappings der Komponente A auf die Zwischenkomponente B, d.h. alle Reisekostendatenelemente. ExpRep-Target.xml ist ein Standarddateiname, der automatisch generiert wird, da in das Feld "XML-Output-Datei" kein Dateiname eingegeben wurde.

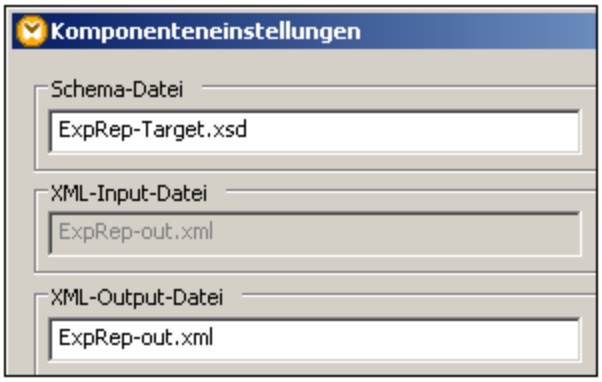
 © 2011 Altova GmbH



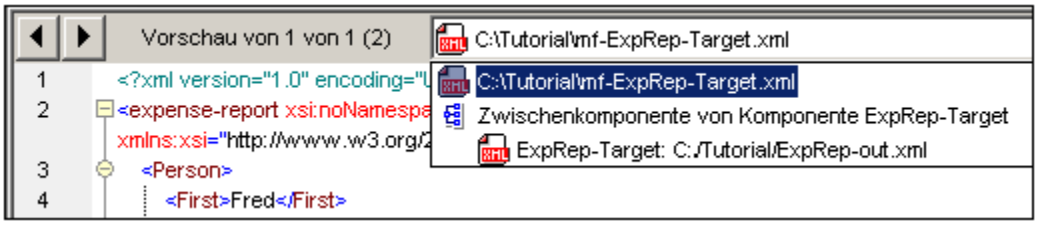
Bitte beachten Sie:
Jedes Mapping-Ergebnis wird in einem eigenen Vorschaufenster angezeigt. Klicken Sie auf die Pfeilschaltfläche(n) um die nächste/vorherige Vorschau zu sehen.



Wenn Sie auf die Dateiauswahlliste klicken, werden die Ergebnisdateien in einer Hierarchie angezeigt. Ganz oben wird das endgültige Ergebnis der Zielkomponente angezeigt, unterhalb davon die Zwischenergebnisdateien. Klicken Sie auf den Namen einer Datei, um diese auszuwählen oder geben Sie den Namen über die Tastatur ein, um durch die Dateiliste zu navigieren, und drücken Sie die Eingabetaste.



Wenn Sie in der Zwischenkomponente B eine XML-Output-Datei mit dem Namen "ExpRep-out.xml" einstellen, so werden die Daten der Zwischenkomponente B in einer Datei dieses Namens, also ExpRep-out.xml, gespeichert.



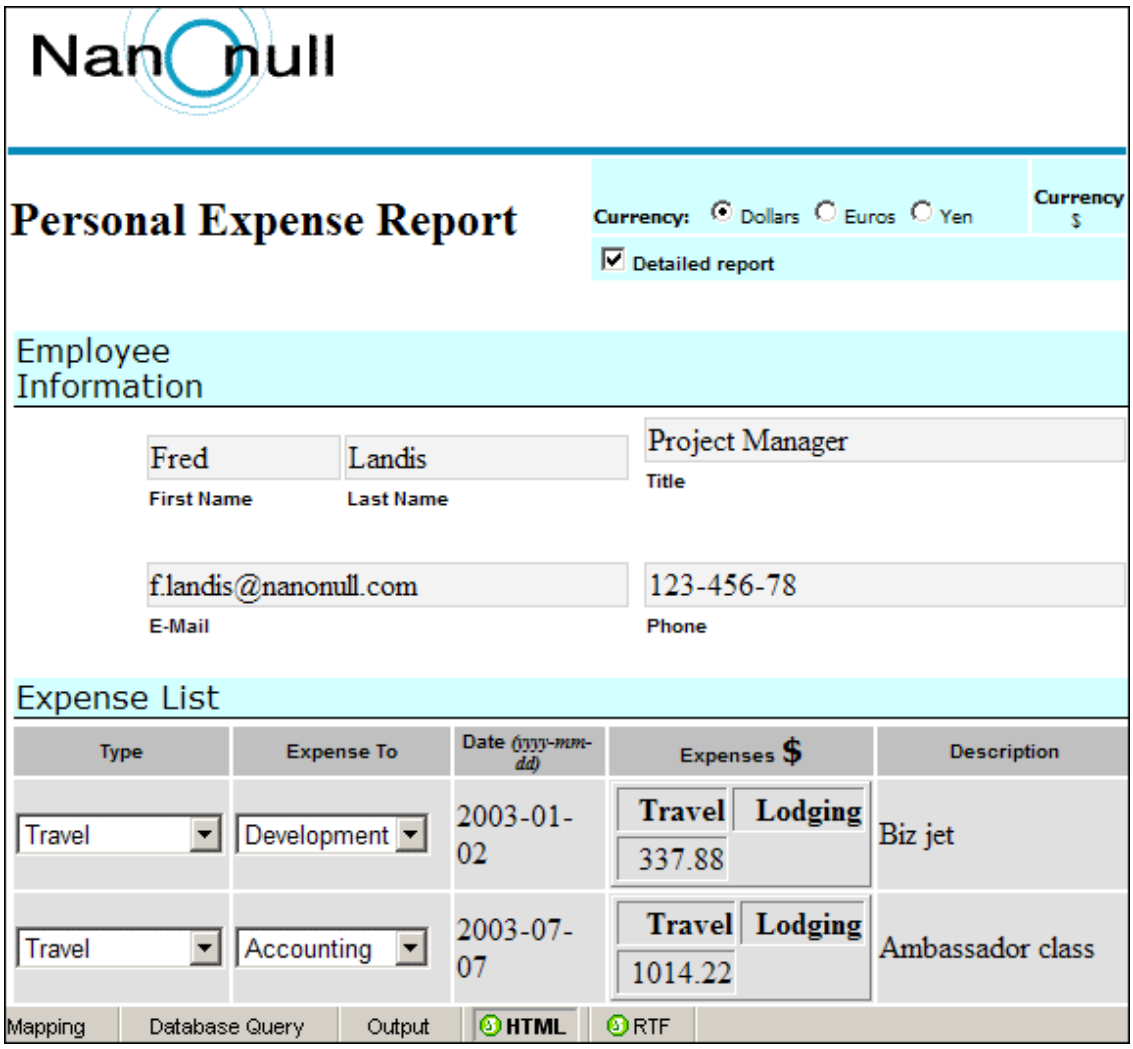
Wenn die Schaltfläche "Weiterleitung" aktiviert ist, werden Dateien, die mittels einer Zwischenkomponente erstellt werden, **automatisch** als Temp-Dateien gespeichert und für die weitere Verarbeitung der Ausgabe dieser Komponente verwendet.

Mit der Einstellung "Direkt in die endgültigen Output-Dateien schreiben" (Extras | Optionen | Allgemein) wird festgelegt, ob Zwischendateien als temporäre Dateien oder als physische Dateien gespeichert werden sollen. Für Zwischenkomponenten wird zum Speichern des Zwischenergebnisses ein Standarddateiname verwendet, außer es wird ein dynamischer Dateiname bereitgestellt bzw. durch ein Mapping geliefert.

Vorschau XX **von 1** gibt die Anzahl der endgültigen Zieldateien an, die von der ausgewählten Zielkomponente aus erzeugt werden können, in diesem Fall 1. Vorschau ... **(2**) bezieht sich auf die Gesamtanzahl der Ergebnisse einschließlich aller Zwischenkomponenten.

Anzeige des Ergebnisses mit Hilfe von StyleVision
Wenn einer Zielkomponente eine SPS-Datei zugewiesen wurde, so werden die Ergebnisdaten bei Klick auf das Register HTML, RTF auf dem entsprechenden StyleVision-Register in MapForce angezeigt.

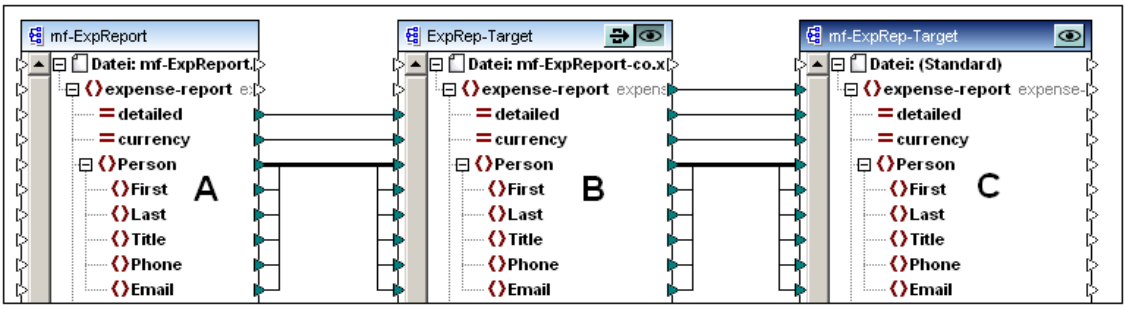
 © 2011 Altova GmbH



Beachten Sie, dass nur Ausgabedateien von Endkomponenten einer Mapping-Kette auf dem Register "StyleVision" von MapForce angezeigt werden. StyleVision-Ausgabedateien von Zwischenkomponenten können nicht angezeigt werden.

6.6.2 Verkettete Mappings - Weiterleitung deaktiviert

Das Beispiel **Tut-ExpReport-chain.mfd** funktioniert anders, wenn die Schaltfläche "Weiterleitung" für die Komponente B **deaktiviert ist.**



Die automatische Weiterleitung der Daten von Komponente A über Komponente B an Komponente C wurde durch Deaktivierung des Schaltfläche "Weiterleitung" unterbrochen. Die Vorschau-Schaltflächen der Komponenten B und C bestimmen, welcher Teil der Mapping-Kette generiert wird.

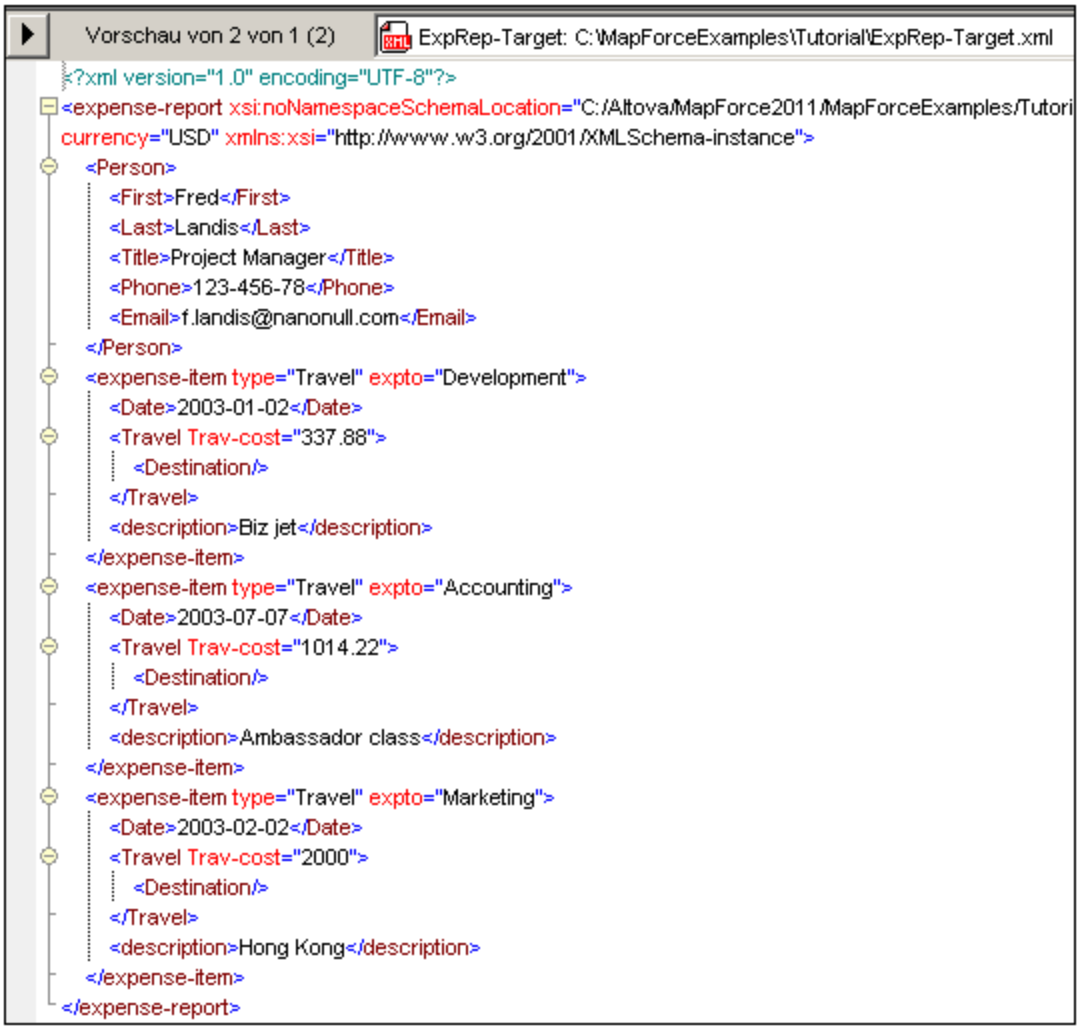
MapForce generiert die Ausgabe für die Komponente, in der die Vorschau-Schaltfläche aktiv ist.

- Wenn die Vorschau-Schaltfläche der Komponente **B** aktiv ist, so wird das Ergebnis des Mappings der Komponente A auf die Komponente B generiert. Komponente C wird ignoriert.

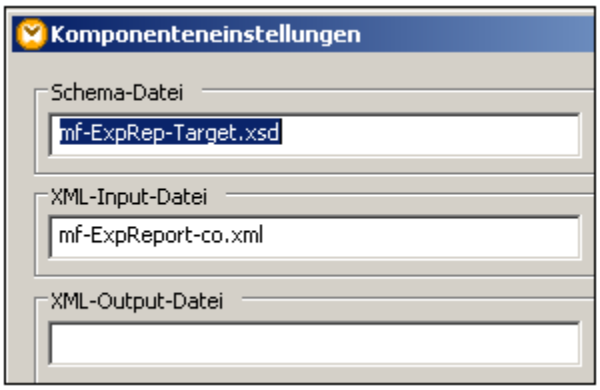
  Wenn Sie auf die Schaltfläche "Ausgabe" klicken, sehen Sie eine Vorschau auf die Ergebnisse im Built-in-Ausführungsprozessor.

Vorschau:
Das Ergebnis des Mappings von Komponente A auf die Zwischenkomponente B, d.h. alle Reisekostenposten.

 © 2011 Altova GmbH



- Wenn die Vorschau-Schaltfläche der Komponente C aktiv ist, werden die Daten von der Zwischenkomponente B auf Komponente C gemappt. Komponente A wird ignoriert. Der Komponente B wurde als XML-Input-Datei die Datei mf-ExpReport-co.xml zugewiesen. Informationen zum Speichern des Zwischenergebnisses eines Mappings finden Sie weiter unten.

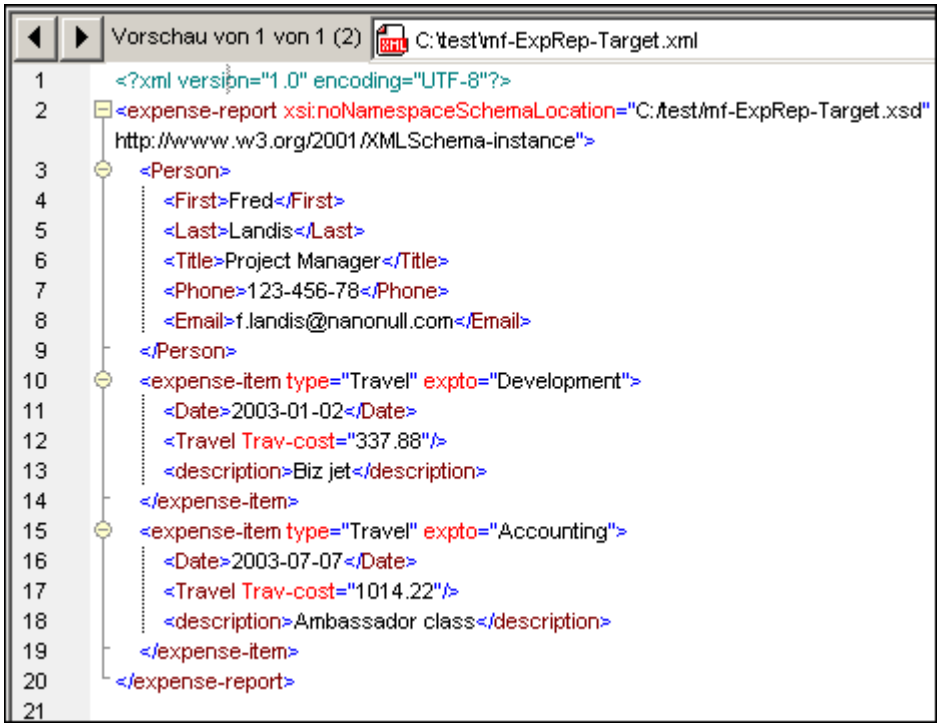


MapForce öffnet die Zwischendatei und mappt Ihre Daten auf Komponente C. Wenn die Input-Datei von Komponente B existiert, werden mit diesem Mapping-Ausgabedaten erzeugt. Es muss eine Datei im Feld "XML-Input-Datei" eingetragen sein, damit das Mapping ausgeführt werden kann. Wenn die Input-Datei fehlt, wird eine Fehlermeldung angezeigt.

Wenn die Schaltfläche "Weiterleitung" deaktiviert ist, ist das Feld *XML-Input-Datei* der Zwischenkomponente aktiviert, wie oben gezeigt.

Beachten Sie den Unterschied zum Fall, in dem die Schaltfläche "Weiterleitung" der Komponente B aktiv ist. In diesem Fall wird das Feld "XML-Input-Datei" automatisch deaktiviert.

Vorschau:
Das Ergebnis des Mappings der Zwischenkomponente B auf die Zielkomponente C, d.h. die Reisekosten unter 1500.



Anmerkung:
Wenn dieses Mapping über die Befehlszeile oder generierten Code ausgeführt wird, versucht MapForce, unabhängig davon, ob die Schaltfläche "Weiterleitung" in Komponente B aktiv ist oder nicht und unabhängig davon, welche Komponente für die Vorschau ausgewählt wurde, die Ausgabedaten von Komponente B und Komponente C zu generieren. Der Status der Vorschau-Schaltfläche hat keine Auswirkung.

Da die Einträge im Feld "XML-Input-Datei" und "XML-Output-Datei" (leer) nicht identisch sind, ist die Mapping-Kette unterbrochen und es kann keine Ausgabedatei für Komponente C generiert werden. Damit die Codegenerierung erfolgreich ist, müssen die Einträge in diesem beiden Feldern identisch sein.

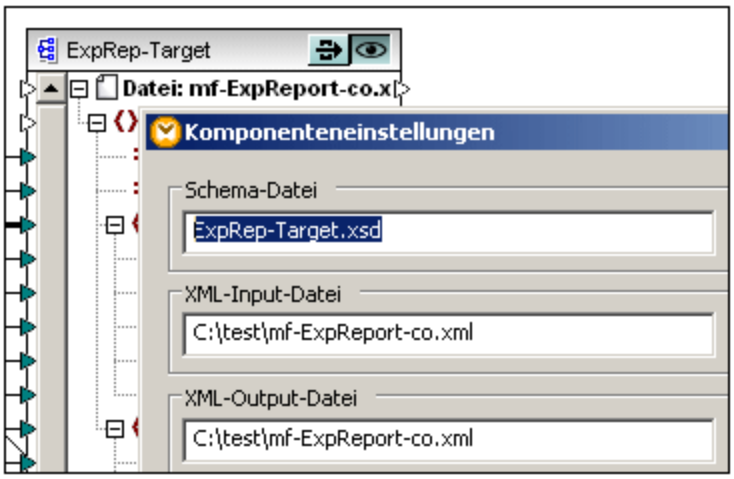
Speichern des Mapping-Zwischenergebnisses

 © 2011 Altova GmbH

Um Zugriff auf die Zwischenkomponente zu erhalten, wenn die Schaltfläche "Weiterleitung" deaktiviert ist, muss das Ergebnis des Mappings von Komponente A auf B gespeichert werden. Der Name dieser Datei muss in das Feld *XML-Input-Datei* der Komponente B eingesetzt werden. Erst dann können die Daten in der Endkomponente C angezeigt werden.

So speichern Sie das Mapping-Zwischenergebnis in einer Datei:

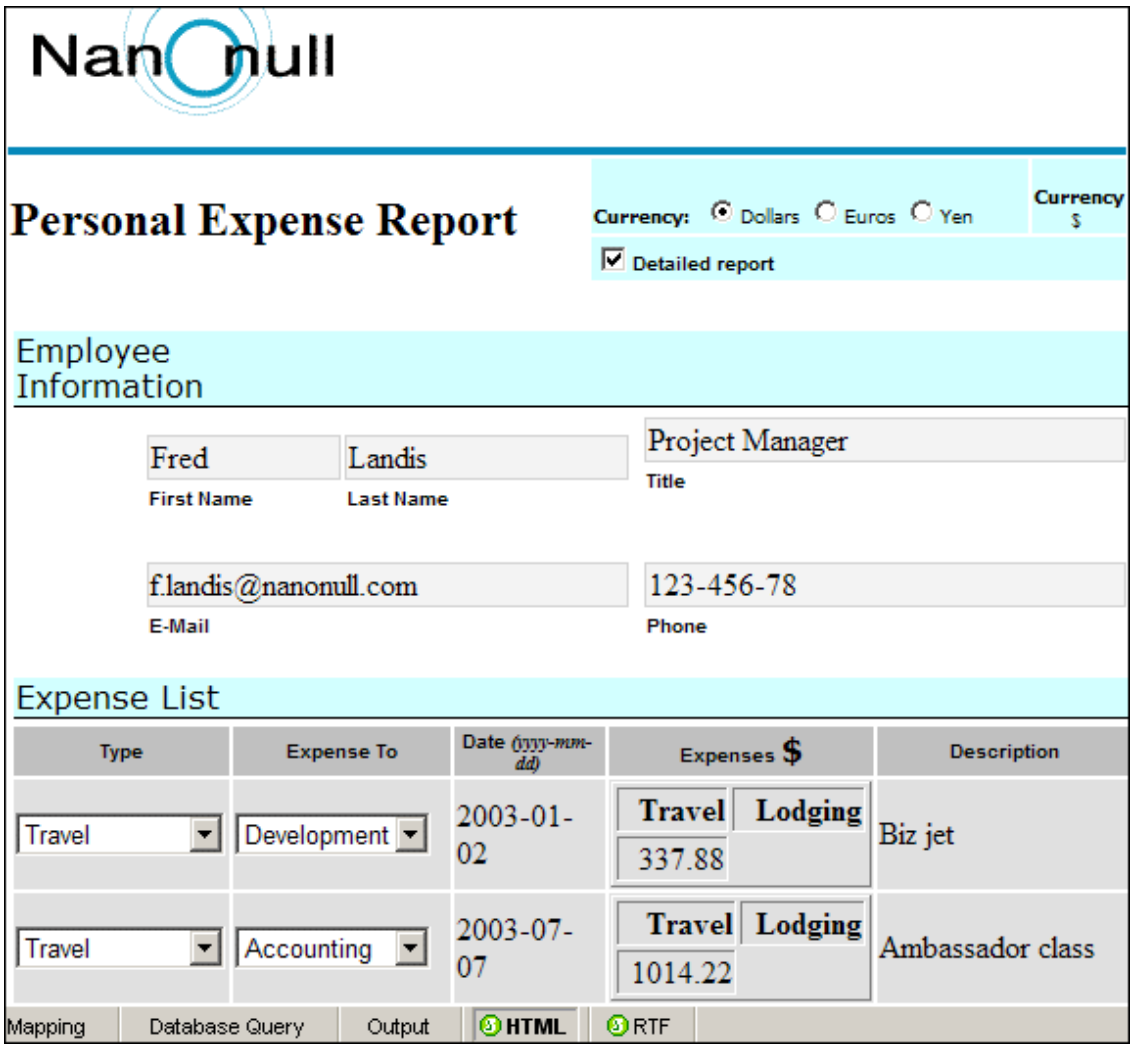
1. Klicken Sie auf die Vorschau-Schaltfläche der Komponente B, um diese zu aktivieren und klicken Sie anschließend auf die Schaltfläche "Ausgabe".
2. Klicken Sie in der Symbolleiste der Ausgabevorschau auf die Schaltfläche "**Generierte Ausgabe speichern**" und geben Sie der XML-Datei einen Namen, z.B. `mf-ExpReport-co.xml`.
3. Doppelklicken Sie auf die Titelleiste der Komponente B, um das Dialgofeld **Komponenteneinstellungen** zu öffnen, kopieren Sie den Dateinamen in das Feld *XML-Input-Datei* und klicken Sie auf **OK**.



Bitte beachten Sie: Damit **Code generiert** werden kann und damit das Mapping über die Befehlszeile ausgeführt werden kann, **muss** sowohl der Name der Input- als auch der der Output-Datei identisch (und vorhanden) sein.

Anzeigen des Ergebnisses mit Hilfe von StyleVision

Wenn einer Zielkomponente eine SPS-Datei zugewiesen wurde, so werden die Ergebnisdaten bei Klick auf eines der Register HTML, RTF auf dem entsprechenden StyleVision-Register von MapForce angezeigt.



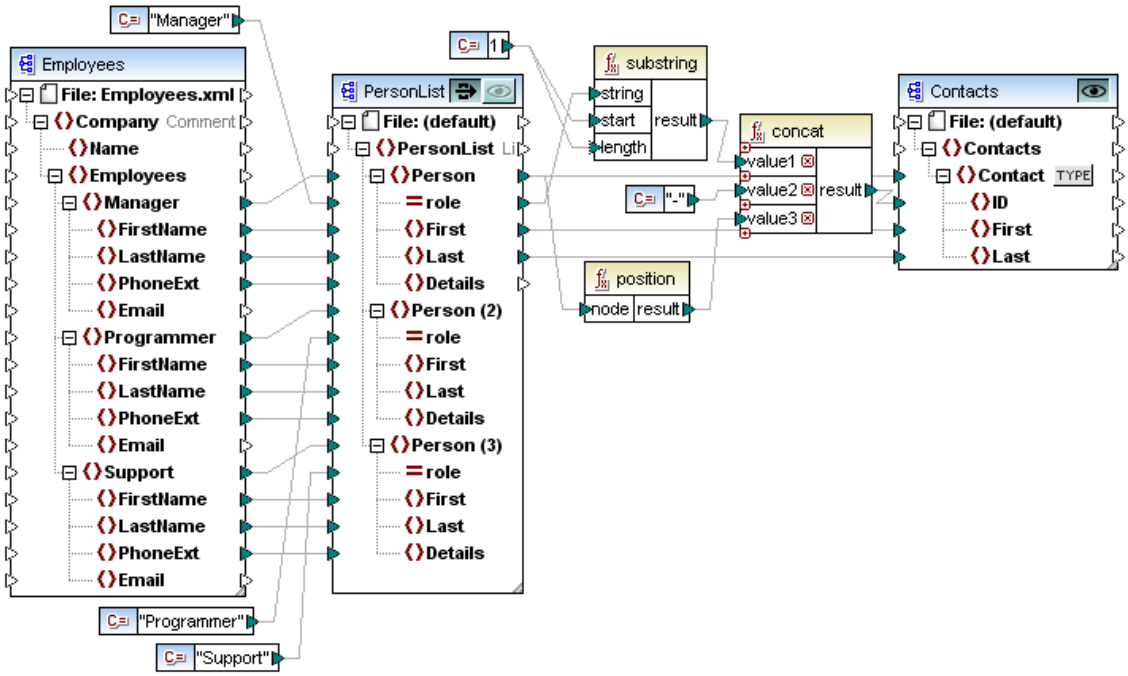
 © 2011 Altova GmbH

6.6.3 Beispiel für ein verkettetes Mapping

Sie finden das unten gezeigte Beispiel unter dem Namen **ChainedPersonList.mfd** im Ordner ...\MapForceExamples.

Ziel:
Die Erstellung zweier Gruppen von Employee-Dokumenten (Angestelltendokumenten), eine für die Personalabteilung und eine für die Buchhaltung.

- Im Dokument für die Buchhaltungsabteilung wird dem Angestellten eine eindeutige ID zugewiesen.
- Das Dokument für die Personalabteilung enthält die Personaldaten sowie die jeweilige Durchwahl (PhoneExt).



Komponenten:

Employees:
Die Instanzdatei employees.xml enthält vier Personen mit folgenden Berufstiteln (roles): manager, programmer und support.

PersonList: (die Ausgabe ist das Dokument für die Personalabteilung)

- Zu den Personendaten wird ein Attribut **role** hinzugefügt. Die Positionshierarchie aus der Komponente "Employees" wird entfernt.
- Die Schaltfläche "Weiterleitung" ist aktiv

Contacts: (ausgegeben wird das Dokument für die Buchhaltung)

- Zu den Contact-Daten wird ein Element "ID" hinzugefügt, um sicherzustellen, dass die Personendaten eindeutig sind.

Funktionsweise:
PersonList:

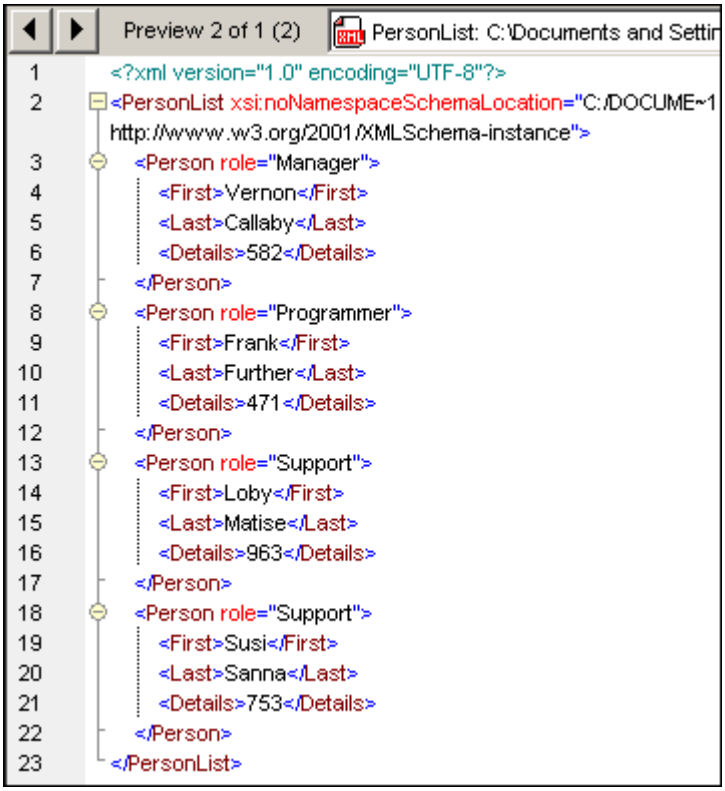
- Das Element "person" wird zweimal dupliziert, um die drei Arten von Berufsbezeichnungen, die es im Unternehmen gibt, zu ermöglichen.
- Die role -Namen werden mittels Konstantenkomponenten als Strings hinzugefügt, wobei die Konstantenkomponenten in derselben Reihenfolge verwendet werden wie in der Komponente "Employees".

Contacts:

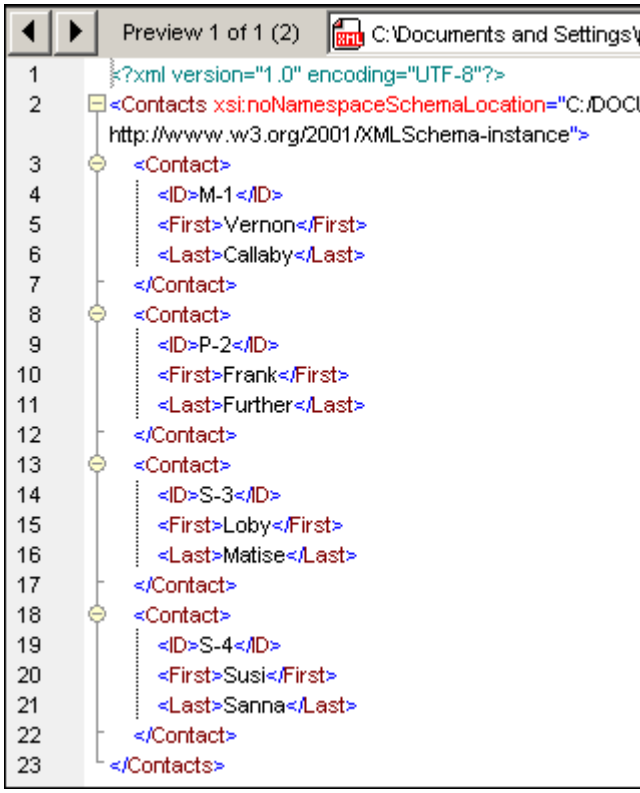
- Die Funktion **substring** trennt den ersten Buchstaben des Attributs "role" ab und leitet ihn an die concat-Funktion weiter.
- Die Funktion **position** iteriert über alle Person-Nodes, weist ihnen eine fortlaufende Nummer zu (beginnend mit 1) und leitet sie an die concat-Funktion weiter.
- Die **concat** Funktion kombiniert das substring-Zeichen, einen Bindestrich (aus der Konstantenkomponente) und die Positionsnummer und leitet das Ergebnis an das Element "ID" der Contacts-Komponente weiter.

Ergebnis:

Die Komponente **PersonList** (Ausgabe: Dokument für die Personalabteilung)



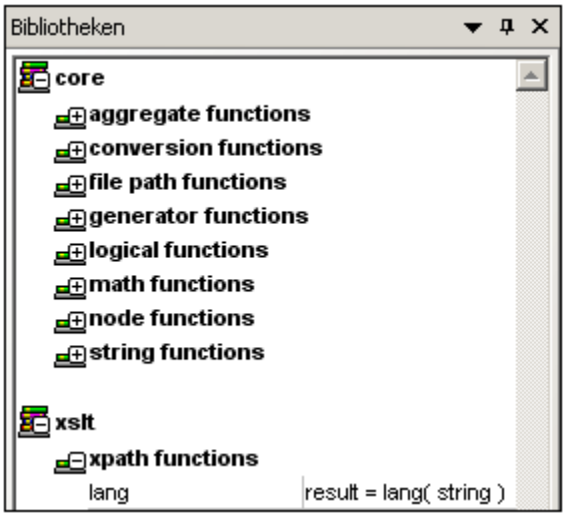
Die Komponente **Contacts** (Ausgabe: Dokument für d Buchhaltung)



 © 2011 Altova GmbH

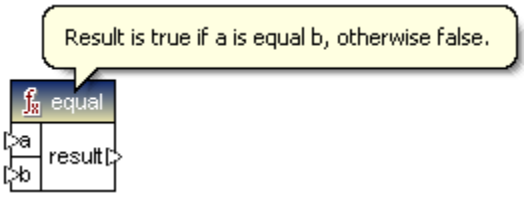
6.7 Arbeiten mit Funktionen

Die in der ausgewählten Sprache (XSLT/XSLT2, BUILTIN) verfügbaren Funktionen werden im Fenster "Bibliotheken" angezeigt. Mit Hilfe der Ein- und Ausklappsymbole können Sie die Funktionen der einzelnen Bibliotheken ein- und ausblenden.



XSLT ausgewählt

Funktions-Tooltips:
Erklärender Text zu den einzelnen Funktionen, der im Bibliotheksfenster angezeigt wird, kann nun durch Klicken auf das Symbol "Tipps anzeigen" in der Symbolleiste ein- oder ausgeblendet werden. Wenn Sie den Mauszeiger über eine Funktionsbezeichnung platzieren, werden Informationen zu dieser Funktion angezeigt.

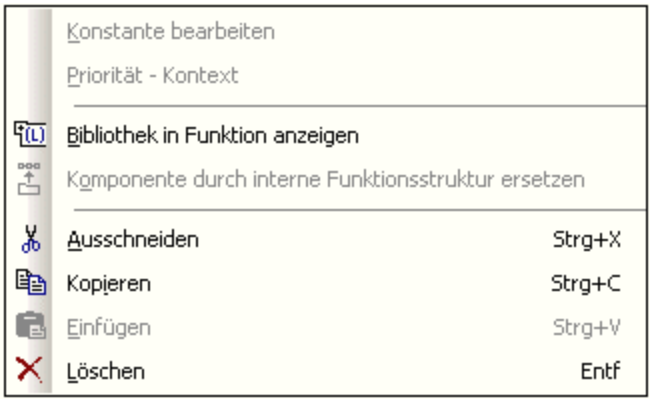


So verwenden Sie eine Funktion im Mapping-Fenster:

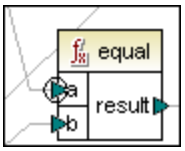
1. Wählen Sie zuerst die **Programmiersprache** aus, für die Code erzeugt werden soll, indem Sie in der Symbolleiste auf eines der Ausgabecode-Symbole klicken.
2. Klicken Sie auf den **Funktionsnamen** und ziehen Sie Ihn in das Mapping-Fenster.
3. Verbinden Sie die Eingabe- und Ausgabeparameter der verschiedenen Symbole durch Drag & Drop miteinander.

Hinweis: Wenn Sie den Mauszeiger im Bibliotheksfenster über dem Ausdruck "result = xxx" positionieren, wird ein Tooltip mit einer näheren Beschreibung der Funktion angezeigt.

Funktions-Kontextmenü:
Durch einen Rechtsklick auf eine Funktion im Mapping-Fenster wird das Kontextmenü geöffnet.



Prioritätskontext Bei Anwendung einer Funktion auf verschiedene Datenelemente eines Schemas muss MapForce den Kontext-Node kennen. Alle anderen Elemente werden dann relativ zu diesem verarbeitet. Zu diesem Zweck wird das Datenelement (oder der Node) als Prioritätskontext definiert. Das jeweilige Symbol wird durch einen Kreis markiert. Ein Beispiel dazu finden Sie im Abschnitt "Referenz" unter .



Bibliothek in Funktionstitelleiste anzeigen Zeigt den Namen der Bibliothek im Titel der Funktionskomponente an.

Komponente durch interne Funktionsstruktur ersetzen Ersetzt die benutzerdefinierte Komponente/Funktion durch ihre Bestandteile. Steht für Funktionen nicht zur Verfügung.

Ausschneiden/Kopieren/Einfügen/Löschen Mit Hilfe der MS Windows-Standardbearbeitungsbefehle können Sie alle im Mapping-Fenster angezeigten Komponenten oder Funktionen ausschneiden, kopieren, usw. Alle Konnektoren mit Ausnahme derer, die ersetzt werden müssten, bleiben erhalten.

Eigenschaften Steht für Funktionen nicht zur Verfügung.

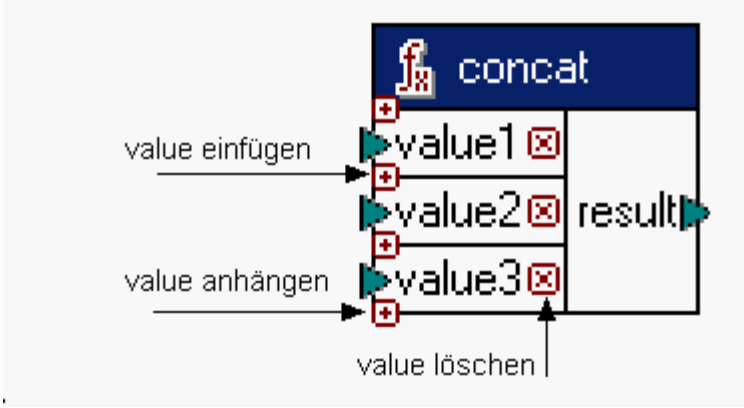
Erweiterbare Funktionen

Eine Reihe von Funktionen aus den Funktionsbibliotheken kann erweitert werden, z.B: die Funktionen concat, "logical-and", "logical-or" und IF-ELSE. Die Parameter dieser Arten von Funktionen können je nach Belieben eingefügt/angehängt und gelöscht werden.

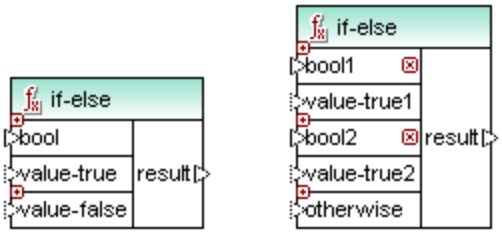
Wenn Sie auf das Plus-Symbol klicken, wird dieselbe Parameterart eingefügt oder angehängt. Wenn Sie auf das Häkchen klicken, wird der Parameter gelöscht.

Beachten Sie bitte. Wenn Sie einen Konnektor auf das Plus-Symbol ziehen, wird der Parameter automatisch eingefügt/angehängt und verbunden.

 © 2011 Altova GmbH



Die IF-Testparameter der IF-Else Funktion können auf dieselbe Art erweitert werden.



6.8 Vorschau auf das Transformationsergebnis

Mit Hilfe des Built-In Ausführungsprozessors können Sie das Ergebnis einer Transformation sofort in der Vorschau anzeigen und speichern, ohne dass vorher Programmcode generiert, kompiliert und anschließend die Ergebnisse angezeigt werden müssen.

Dazu müssen Sie einfach - unabhängig davon, welche Ausgabesprache Sie gewählt haben, auf die Schaltfläche **Ausgabe** klicken. Das Fenster **Ausgabe** unterstützt auch den Befehl "Suchen", um nach allen benötigten XML-Daten suchen zu können.

Wenn Sie auf die Schaltfläche **BUILTIN** klicken, wird der Built-In-Ausführungsprozessor explizit als Ausgabeziel ausgewählt. Nachdem Sie auf die Schaltfläche BUILTIN und anschließend auf **Ausgabe** geklickt haben, stehen alle Standardoptionen des Ausgabefenster zur Verfügung: also **Ausgabedatei validieren**, **Ausgabedatei speichern...**, **Alle Ausgabedateien speichern...**, usw.

MapForce lässt sich auch über die Befehlszeile starten und kann mit Hilfe des Parameters /BUILTIN auch ohne die Erzeugung von Zwischencode ein Ergebnis produzieren.

Streaming:
Der Built-In-Ausführungsprozessor ist das einzige Ziel, das XML, CSV und FLF **Streaming** unterstützt. Dies bedeutet, dass es für die **Ausgabedateien** keine Größenbeschränkung gibt. Bei sehr großen Ausgabedateien wird am unteren Rand des Ausgabefensters eine Schaltfläche "Mehr laden.." angezeigt. Wenn Sie darauf klicken, wird ein großer Abschnitt an die bereits sichtbaren Daten angehängt. Die Schaltfläche "Pretty-Print" wird aktiv, sobald die komplette Datei in das Ausgabefenster geladen wurde.

Digitale Signaturen:
MapForce unterstützt die Erstellung von digitalen XML-Signaturen für XML- und XBRL-Ausgabedateien. Digitale Signaturen können nur erstellt werden, wenn das BUILTIN das Ausgabeziel ist. Eine Signatur wird erstellt, wenn Sie auf die Schaltfläche "Ausgabe" klicken und die Ergebnisdatei gespeichert wird. Nähere Informationen dazu finden Sie unter Digitale Signaturen.

Vorschau auf die Transformationsausgabe
Im Menü **Ausgabe** oder bei Auswahl einer der Schaltflächen in der Symbolleiste "Auswahl der Sprache" (*Abbildung unten*) können Sie definieren, welche Sprach MapForce zum Transformieren der Daten der Input-Datei verwenden soll.

Die folgenden Optionen stehen zur Verfügung:

- XSLT
- XSLT2

Auf dem Register "XSLT/XSLT2" sehen Sie eine Vorschau auf den XSLT-Code. Er wird mit der Menüoption **Datei | Code generieren in | XSLT 1.0** bzw. **XSLT 2.0** generiert und gespeichert.

Anzeigen einer Vorschau auf den Programmcode

Bevor der Programmcode generiert wird, empfiehlt es sich, das Ergebnis von XSLT mit Hilfe des Built-In-Ausführungsprozessor anzuzeigen.

So zeigen Sie eine Vorschau eines XSLT- Ergebnisses an:

 © 2011 Altova GmbH

Öffnen Sie zuerst die Datei *.mfd in MapForce:

1. Wählen Sie eine der folgenden Methoden:
   - Klicken Sie auf die Schaltfläche **XSLT,** um eine Vorschau des generierten XSLT 1.0-Codes zu sehen.
   - Klicken Sie auf die Schaltfläche **XSLT2,** um eine Vorschau des generierten XSLT 2.0-Codes zu sehen.
2. Klicken Sie auf die Schaltfläche **Ausgabe**, um eine Vorschau des Mapping-Ergebnisses anzuzeigen.

Generieren von XSLT-Code

XSLT 1.0- und XSLT 2.0-Programmcode kann von MapForce mit Hilfe des AltovaXML-Prozessors, der im AltovaXML Paket enthalten ist, generiert werden. Der Prozessor kann auch direkt in XMLSpy aufgerufen und die Codegenerierung kann über den Menübefehl **XSL/XQuery | XSL-Transformation** durchgeführt werden.

So laden Sie das AltovaXML-Paket herunter:

1. Gehen Sie zu http://www.altova.com/download_components.html,
2. Klicken Sie auf den Link "Informieren Sie sich über AltovaXML und laden Sie das Tool jetzt herunter!"
3. Wählen Sie die gewünschte AltovaXML-Edition aus und installieren Sie sie.

So generieren Sie XSLT-Code:

1. Wählen Sie die Menüoption **Datei | Code generieren in | XSLT 1.0 (oder XSLT 2.0)**.
3. Wählen Sie den Ordner, in dem die generierte XSLT-Datei abgelegt werden soll. Es wird eine Meldung angezeigt, dass die Generierung erfolgreich war.
4. Navigieren Sie zum angegebenen Ordner, wo Sie die generierte XSLT-Datei finden.

Bitte beachten Sie: Eine Aktivierung des Kontrollkästchens "Bytefolge-Markierung" im Dialogfeld "Komponenteneinstellungen" hat keine Auswirkung auf die Ausgabe von XSLT 1.0/XSLT 2.0, da diese Sprachen BOMs (Byte Order Marks) nicht unterstützen.

Transformieren der XML-Quelldatei mit Hilfe des generierten XSLT-Codes.

Wenn Sie XSLT-Code generieren, wird eine Batch-Datei namens DoTransform.bat, die die XML-Datei mit Hilfe von AltovaXML transformiert, generiert. Diese Datei wird in dem Ordner gespeichert, in dem Sie auch die XSLT-Datei speichern.

So transformieren Sie die XML-Datei mittels der generierten XSLT-Datei:

1. Laden Sie den kostenlosen AltovaXML-Prozessor von der AltovaXML Download-Seite herunter und installieren Sie ihn. AltovaXML wird standardmäßig im Verzeichnis c:\Program Files\Altova\AltovaXML2012\ installiert.
2. Starten Sie die Batch-Datei **DoTransform.bat,** die sich im zuvor angegebenen Ausgabeordner befindet.
   Daraufhin wird im aktuellen Ordner die Ausgabedatei **ExpReport-Target.xml** generiert.

Anmerkung: Eventuell müssen Sie den AltovaXML Installationspfad zur Pfadvariablen der Umgebungsvariablen hinzufügen.

6.9 Validieren von Mappings und Mapping-Ausgabe

Es müssen nicht alle Funktionen oder Komponenten gemappt werden. Das Mapping-Register ist ein Arbeitsbereich, in den beliebige verfügbare Komponenten platziert werden können. XSLT 1.0, XSLT 2-, wird nur für die Komponenten generiert, für die gültige Verbindungen existieren.

Bei **frei stehenden** Komponenten werden keinerlei Fehlermeldungen oder Warnungen erzeugt.

Bei **teilweise verbundenen** Komponenten können zwei Arten von Warnungen generiert werden:

- Wenn das **Input-Symbol** einer Funktionskomponente nicht verbunden ist, wird eine Fehlermeldung generiert und die Transformation angehalten.
- Wenn das **Output-Symbol** der Funktion nicht verbunden ist, wird eine Warnung generiert und der Transformationsvorgang fortgesetzt. Die störende Komponente und ihre Daten werden ignoriert und nicht auf das Zieldokument gemappt.

Sie können mehrere Meldungsregister verwenden, wenn Ihr Projekt mehrere separate Mapping-Dateien enthält. Klicken Sie auf eines der nummerierten Register im Fenster "Meldungen" und klicken Sie auf das Register "Vorschau" um ein anderes Mapping in Ihrem Projekt zu sehen. Die Validierungsmeldung wird nun auf dem ausgewählten Register angezeigt. Die ursprüngliche Meldung bleibt jedoch auf Register 1 erhalten.

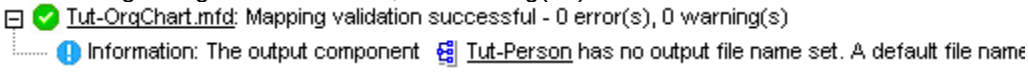
Verwenden Sie die verschiedenen Symbole auf dem Register "Meldungen" zu folgenden Zwecken:

- Filtern der Meldungsarten, Fehler oder Warnmeldungen
- Scrollen durch die Einträge
- Kopieren des Meldungstexts in die Zwischenablage
- Suchen eines bestimmten Strings in einer Meldung
- Löschen der Meldungen im Fenster " Meldungen"

So validieren Sie ein Mapping:

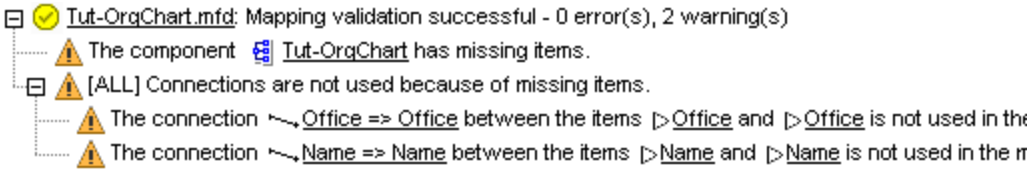
- Klicken Sie in der Symbolleiste der Applikation auf das Symbol "**Mapping validieren**" oder wählen Sie den Menübefehl **Datei | Mapping validieren.**
  Im Fenster "Meldungen" wird eine Validierungsmeldung angezeigt.

Validierungsmeldungen:
Validierungsmeldungen werden im Fenster "Validierungen" angezeigt und zeigen an, ob die Validierung erfolgreich war oder nicht. Zusätzlich dazu werden Fehlermeldungen, Warnungen und Informationen über das Mapping angezeigt. Es gibt zwei Arten von Validierungsmeldungen:

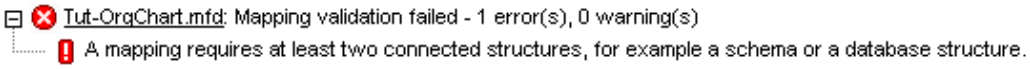
- Validierung erfolgreich - X Fehler, Y Warnung(en).
  

  Warnungen weisen auf ein Problem hin. Das Mapping und die Transformationsvorschau werden dadurch aber nicht abgebrochen. Ein Mapping kann daher 0 Fehler und n Warnungen aufweisen.

 © 2011 Altova GmbH

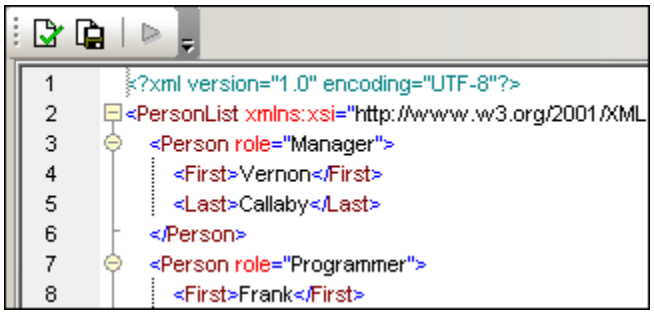


- Validierung fehlgeschlagen - X Fehler, Y Warnung(en).
  Fehler stoppen die Transformation und erzeugen eine Fehlermeldung. Bei Fehlern dieser Art ist eine XSLT, XQuery- oder Ausgabe nicht möglich. Wenn Sie im Fenster "Meldungen" auf eine Validierungsmeldung klicken, wird das Symbol der Komponente, bei der das Problem aufgetreten ist, im Mapping-Fenster markiert.

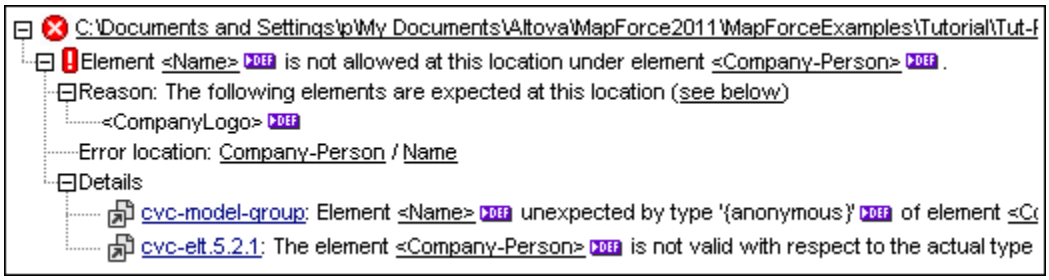


Validieren der Mapping-Ausgabe
Wenn Sie auf die Schaltfläche "Ausgabe" klicken, werden die Daten mit Hilfe des MapForce, XSLT 1.0/XSLT 2.0- oder XQuery-Prozessors transformiert und es wird ein Ergebnis in der Textansicht erzeugt.

Werden die Daten auf ein XML-Schema gemappt, kann das erzeugte XML-Dokument gegen das zugrunde liegende Schema validiert werden.



Das Ergebnis der Validierung wird im Fenster "Meldungen" angezeigt. Wenn die Validierung nicht erfolgreich war, so enthält dieses Fenster detaillierte Informationen über die aufgetretenen Fehler (*siehe Abbildung unten*).



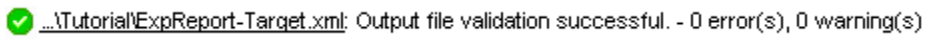
Die Validierungsmeldung enthält eine Reihe von Hyperlinks. Wenn Sie darauf klicken erhalten Sie weitere ausführliche Informationen::

- Wenn Sie auf den Dateipfad klicken, wird die Ausgabe der Transformation im Ausgabefenster von MapForce geöffnet.
- Wenn Sie auf den Link <ElementName> klicken, wird das Element im Ausgabefenster markiert.

- Wenn Sie auf das Symbol klicken, wird die Definition des Elements in XMLSpy geöffnet (falls das Programm installiert ist).
- Wenn Sie auf die Hyperlinks im Unterabschnitt "Details" klicken (z.B., cvc-model-group) wird eine Beschreibung der entsprechenden Validierungsregel auf der http://www.w3.org/ Website geöffnet.

So validieren Sie die Mapping-Ausgabe:

- Klicken Sie auf die Schaltfläche "Validieren" , um das Dokument gegen das Schema zu validieren. Daraufhin wird entweder die Meldung "Das ausgegebene XML-Dokument ist gültig" oder eine Meldung mit der Erklärung des Fehlers angezeigt.



Bitte beachten Sie:
Über das Feld **Schema /DTD-Referenz hinzufügen** im Dialogfeld "Komponenteneinstellungen" können Sie den Pfad der referenzierten XML-Schemadatei zum Root-Element der XML-Ausgabedatei hinzufügen.

 © 2011 Altova GmbH

6.10 Schleifen, Gruppen und Hierarchien

Es gibt verschiedene Möglichkeiten, Quell- und Zielhierarchien in Schleifen zu verarbeiten. Sie können definieren, wie bestimmte Gruppen von Daten in Form von Schleifen verarbeitet oder gruppiert werden sollen.

Unter den folgenden Links finden Sie nähere Informationen zu diesen Themen. Beachten Sie bitte, dass diese Beispiele nicht nacheinander aufgelistet sind, sondern in verschiedenen Kapiteln dieser Dokumentation behandelt werden.

Lookup-Tabelle für die Wertezuordnung

Prioritätskontext

6.11 Mapping-Regeln und -Strategien

MapForce mappt Daten im Allgemeinen auf intuitive Art, doch gibt es einige Szenarien, in denen die erzeugte Ausgabe scheinbar zu viele oder zu wenige Datenelemente hat. Diese Fälle werden in diesem Abschnitt behandelt.

Allgemeinregel
Im Allgemeinen bedeutet jede Verbindung zwischen einem Quell- und einem Zieldatenelement: Erstelle für jedes Quelldatenelement ein Zieldatenelement.

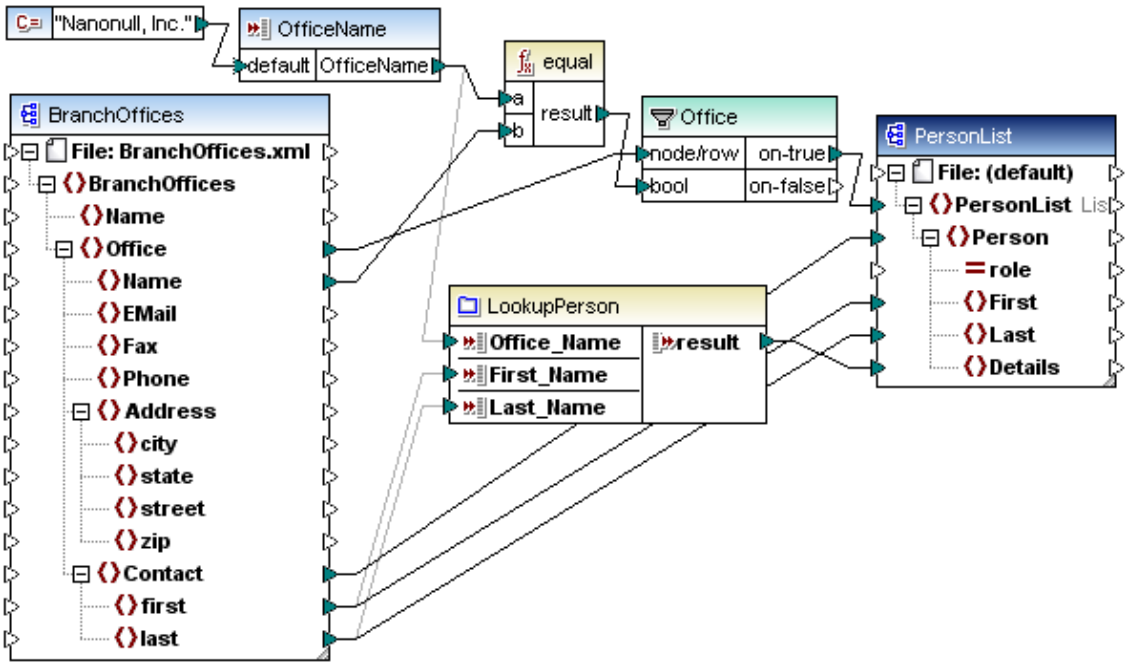
Wenn der Quell-Node simple Content (z.B. String, Ganzzahl usw.) enthält und im Ziel-Node simple Content gestattet ist, so wird der Inhalt kopiert und der Datentyp wird gegebenenfalls konvertiert.

Für diese Regel gibt es einige Ausnahmen, jedoch gilt sie im Allgemeinen für alle Verbindungen.

Die Kontext-Datenelemente und aktuelle Datenelemente
MapForce zeigt die Struktur eines Schemas in Form einer Hierarchie mapbarer Datenelemente in der Komponente an. Jeder dieser Nodes kann in der Instanzdatei bzw. der Datenbank viele (oder keine) Instanzen haben.

Beispiel: Wenn Sie sich in der Datei PersonListByBranchOffice.mfd die Quellkomponente ansehen, so gibt es dort nur einen einzigen Node namens **first** (unter **Contact**). In der Instanzdatei **BranchOffices.xml** gibt es unterhalb unterschiedlicher übergeordneter **Office** -Nodes mehrere **first**-Nodes und **Contact**-Nodes mit jeweils unterschiedlichem Inhalt.

Es hängt vom aktuellen **Kontext** (des **Ziel-**Node) ab, welche Quell-Nodes tatsächlich ausgewählt werden, damit ihre Daten über den Konnektor in die Zielkomponente/das Zieldatenelement kopiert werden.



Dieser Kontext definiert sich durch den **aktuellen Ziel-Node** und seine Verbindungen mit

 © 2011 Altova GmbH

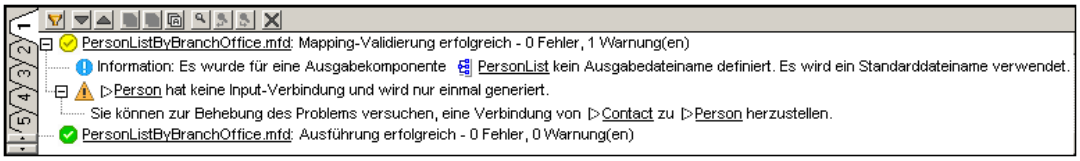
übergeordneten Nodes.

- Am Anfang enthält der Kontext nur die Quellkomponenten, aber keine spezifischen Nodes. MapForce verarbeitet beim Mapping zuerst den **Ziel-Node** (PersonList) und arbeitet sich anschließend durch die Hierarchie nach unten.
- Der Konnektor zum **Ziel-Node** wird zurück zu allen direkt oder indirekt (über Funktionen zwischen den zwei Komponenten) damit verbundenen Quell-Nodes verfolgt. Die Quell-Nodes und die Ergebnisse der Funktionen werden zum Kontext für diesen Node hinzugefügt.
- Für jeden neuen Ziel-Node wird ein neuer Kontext angelegt, der anfangs alle aktuellen Datenelemente des Kontexts des übergeordneten Nodes enthält. Gleichrangige Ziel-Nodes sind daher voneinander unabhängig, haben aber Zugriff auf alle Quelldaten der übergeordneten Datenelemente.

Angewendet auf das obige Beispielmapping (**PersonListByBranchOffice.mfd**):

- Die Verbindung von **Office** durch den Filter (Office) zu **PersonList** definiert ein **einziges** Büro (office) als den Kontext für das gesamte Zieldokument (weil PersonList des Root-Element der Zielkomponente ist). Der Büroname wird von der Input-Komponente geliefert, deren Standardinhalt "Nanonull, Inc." ist.
- Die Filterbedingung wirkt sich auf alle Verbindungen zu den **Nachfahren** bzw. Daten des Root-Elements "PersonList" aus, da sich das ausgewählte Büro im Kontext befindet.
- Durch die Verbindung von **Contact** zu **Person** wird eine Ziel-Person **pro** Contact-Element der XML-Quelldatei (allgemeine Regel) angelegt. Für jede Person wird ein bestimmter Contact zum Kontext, anhand dessen die untergeordneten Nodes von Person erstellt werden, hinzugefügt.
- Der Konnektor von **first** zu **First** wählt den Vornamen des aktuellen Kontakts (Contact) aus und schreibt ihn in das Zieldatenelement "First".

  Wenn wir den Konnektor von **Contact** zu **Person** weglassen, würde nur ein Person-Element mit mehreren First-, Last- und Detail-Nodes erstellt werden, was nicht das gewünschte Ergebnis erzielt. In solchen Fällen gibt MapForce eine Warnmeldung aus und schlägt eine Lösung für das Problem vor: "Sie können zur Behebung des Problems versuchen, eine Verbindung von Contact zu Person herzustellen".



Sequenzen

MapForce zeigt die Struktur eines Schemas in Form einer Hierarchie mapbarer Datenelemente in der Komponente an.

Je nach dem (Ziel)-Kontext kann ein **mapbares Datenelement** einer **Quellkomponente** Folgendes repräsentieren:

- einen **einzelnen Instanz-Node** der zugewiesenen Input-Datei
- eine **Sequenz** von 0 bis zu mehreren **Instanz-Nodes** der Input-Datei

Wenn eine Sequenz mit einem Ziel-Node verbunden wird, wird eine Schleife erstellt, mit der so viele Ziel-Nodes angelegt werden, wie Quell-Nodes vorhanden sind.

Wenn ein **Filter** zwischen die Sequenz und den Ziel-Node platziert wird, wird die Boolsche Bedingung für jeden Input-Node, also für jedes Datenelement in der Sequenz, überprüft. Genauer gesagt, wird überprüft, ob es mindestens eine Boolsche Bedingung in jeder Sequenz gibt, deren Ergebnis "true" ist. Die Prioritätskontext-Einstellung kann sich auf die Reihenfolge der Auswertung auswirken. Siehe unten.

Wie bereits oben erwähnt, werden Filterbedingungen automatisch auf alle untergeordneten Nodes angewendet.

Hinweis: Wenn im Quellschema angegeben ist, dass ein bestimmter Node nur genau einmal vorkommt, kann MapForce die Schleife entfernen und nur das erste Datenelement verwenden, weil ja bekannt ist, dass der Node vorhanden sein muss. Diese Optimierung kann im Dialogfeld "Komponenteneinstellungen" deaktiviert werden (Kontrollkästchen " Input-Verarbeitungsoptimierungen auf Basis von min/maxOccurs aktivieren").

Funktions-Inputs (von normalen Funktionen, die keine Sequenz-Funktionen sind) funktionieren ähnlich wie Ziel-Nodes: Wenn eine Sequenz mit einem solchen Input verbunden wird, wird eine Schleife rund um den Funktionsaufruf erstellt, damit **so viele Ergebnisse** erzeugt werden, wie es Datenelemente in der Sequenz gibt.

Wenn eine Sequenz mit mehr als einem solchen Funktions-Input verbunden wird, erzeugt MapForce verschachtelte Schleifen, die das **kartesische Produkt** aller Inputs verarbeiten. Gewöhnlich ist dies nicht erwünscht. Daher sollte nur eine einzige Sequenz mit mehreren Datenelementen mit einer Funktion (und allen anderen Parametern, die an einzelne aktuelle Datenelemente aus übergeordneten Nodes oder anderen Komponenten gebunden sind) verbunden werden .

Hinweis: Wenn eine leere Sequenz mit einer solchen Funktion (z.B. concat) verbunden ist, erhalten Sie als Ergebnis eine **leere Sequenz** und somit keinen einzigen Output Node. Wenn Ihre Ausgabedatei kein Ergebnis enthält, weil es keine Input-Daten gibt, können Sie mit Hilfe der “substitute-missing” Funktion einen Ersatzwert einfügen.

Funktionen mit Sequenz-Inputs sind die einzigen Funktionen, die ein Ergebnis erzeugen können, wenn die Input-Sequenz **leer** ist:

- "**exists**", "not-exists" und "substitute-missing" (sowie "is-not-null, "is-null" und "substitute-null", die Aliasnamen für die erstgenannten drei Funktionen sind)
- **aggregate** Funktionen ("sum", "count" usw.)
- **reguläre** benutzerdefinierte Funktionen, die Sequenzen unterstützen (also nicht-inline-Funktionen),

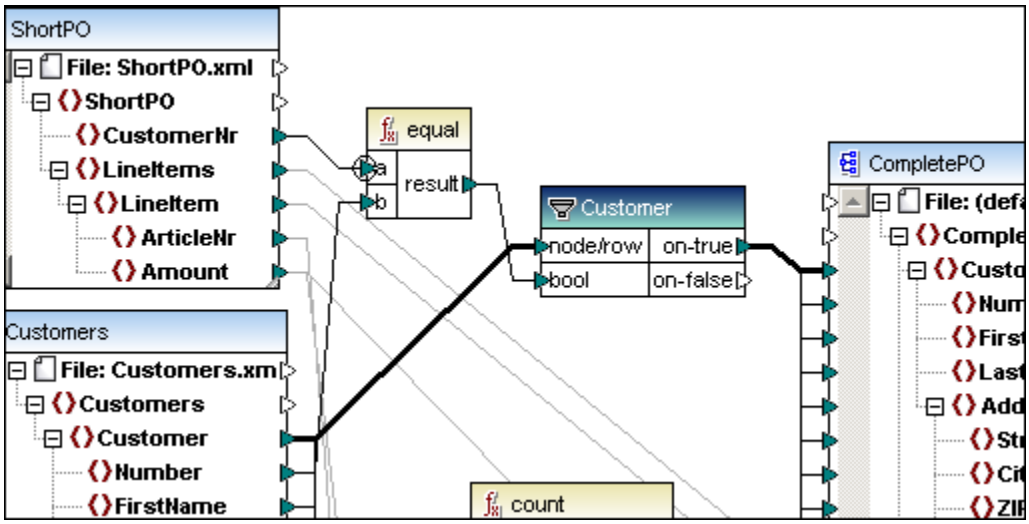
Der Sequenz-Input für derartige Funktionen wird immer unabhängig vom aktuellen Ziel-Node im Kontext seiner übergeordneten Nodes ausgewertet. Dies bedeutet auch, dass alle

 © 2011 Altova GmbH

Filter-Komponenten, die mit solchen Funktionen verbunden sind, sich nicht auf andere Verbindungen auswirken.

Prioritätskontext
Im Allgemeinen werden Funktionsparameter von oben nach unten ausgewertet, es kann aber mit Hilfe der Einstellung "**Prioritätskontext**" auch ein Parameter definiert werden, der vor allen anderen ausgewertet werden soll.

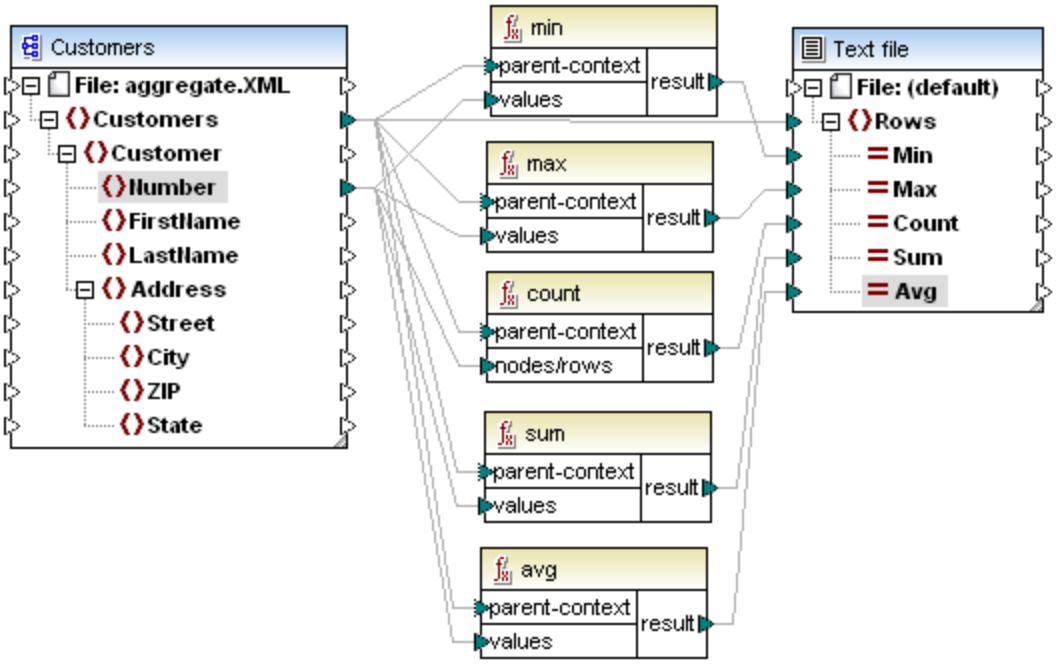
In Funktionen, die mit dem Boolschen Input von **Filterbedingungen** verbunden sind, wirkt sich der Prioritätskontext nicht nur auf die Vergleichsfunktion selbst, sondern auch auf die Auswertung des Filters aus, daher können auch zwei Quellsequenzen (siehe CompletePO.mfd, CustomerNo und Number) miteinander verbunden werden.



In diesem Beispiel wird ShortPO/CustomerNr aufgrund des Prioritätskontexts ausgewertet, bevor durch die Customer Nodes aus der Komponente "Customers" iteriert wird und diese gefiltert werden. Siehe Beispiel Prioritätskontext Node/Datenelement.

Außerkraftsetzung des Kontexts
Einige aggregate-Funktionen haben einen optionalen “parent-context” Input.

Wenn dieser Input nicht verbunden wird, hat dies keine Auswirkung und die Funktion wird innerhalb des normalen Kontexts für Sequenz-Inputs (also im Kontext des übergeordneten Node des Ziel-Node) ausgewertet.



Wenn der parent-context-Context-Input mit einem Quell-Node verbunden ist, wird die Funktion **für jeden** "parent-context" Node ausgewertet und für jede Instanz wird ein separates Ergebnis erzeugt.

Ausnahmen für die allgemeine Regel (Erstelle für jedes Quelldatenelement ein Zieldatenelement)

- Ein XML-Root-Zielelement wird immer nur ein einziges Mal erstellt. Wenn eine Sequenz damit verbunden wird, so wird nur der Inhalt des Elements wiederholt, nicht aber das Root-Element selbst. Das Ergebnis ist aber unter Umständen nicht immer dem Schema gemäß gültig. Wenn auch Attribute des Root-Elements verbunden werden, so schlägt die XML-Serialisierung zur Laufzeit fehl. Daher sollten Sie Sequenzen nicht mit einem Root-Element verbinden. Um mehrere Ausgabedateien zu erzeugen, verbinden Sie die Sequenz statt dessen über eine Funktion, die Dateinamen generiert, mit dem "Datei"-Node.

- Bei einigen anderen Nodes, wie z.B. XML-Attributen, oder der Ausgabekomponente innerhalb einer benutzerdefinierten Funktion, ist nur ein einziger Wert zulässig.

Einbeziehen mehrerer Nodes derselben Quellkomponente in den Kontext:
Dies ist in einigen speziellen Fällen nötig und kann mit Hilfe von Zwischenvariablen erzielt werden.

 © 2011 Altova GmbH

Kapitel 7

Reihenfolge der Verarbeitung von Mapping-Komponente

7 Reihenfolge der Verarbeitung von Mapping-Komponenten

MapForce unterstützt Mappings, die aus mehreren Zielkomponenten bestehen. Jede der Zielkomponenten hat eine Vorschau-Schaltfläche, mit der Sie eine Vorschau des Mapping-Ergebnisses für die jeweilige Komponente anzeigen können.

Wenn das Mapping über die Befehlszeile oder anhand von generiertem Code ausgeführt wird, so wird, unabhängig davon, welche Vorschau gerade aktiv ist, das gesamte Mapping ausgeführt und die Ausgabe für jede einzelne Zielkomponente generiert.

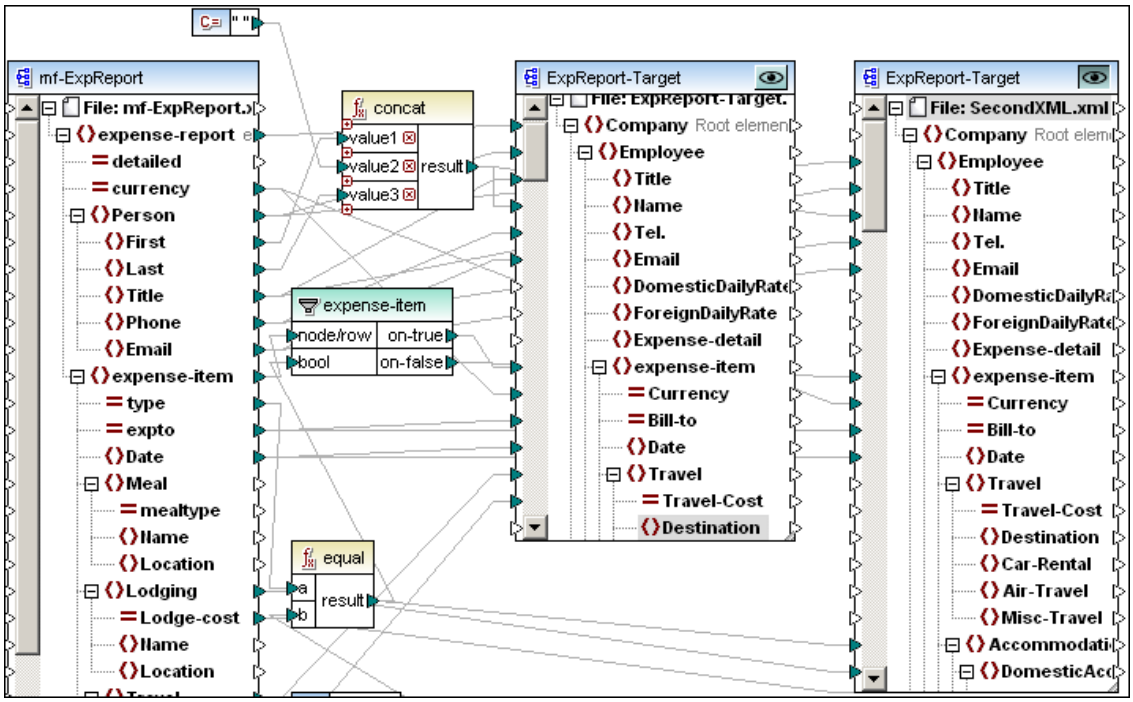
Die Reihenfolge, in der die Zielkomponenten verarbeitet werden, kann durch Ändern der Position der Zielkomponente im Mapping-Fenster beeinflusst werden. Die Position einer Komponente wird anhand ihrer linken oberen Ecke definiert.

Die Zielkomponenten werden anhand ihrer Y-X-Position am Bildschirm von oben nach unten und von links nach rechts verarbeitet.

- 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.

In der Abbildung unten sehen Sie das Tutorial-Beispiel Tut-ExpReport-multi.mfd aus dem Ordner ...\MapForceExamples\Tutorial.

Beide Zielkomponenten (ExpReport-Target) nehmen dieselbe **vertikale** Position ein. Die Vorschau-Schaltfläche ist bei der Zielkomponente auf der rechten Seite aktiv.



 © 2011 Altova GmbH

Nach Auswahl von XSLT2 und Generierung von Code:

- wird die am weitesten links gelegene Zielkomponente zuerst verarbeitet und die Datei ExpReport.xml generiert.
- wird als nächstes die Komponente rechts davon verarbeitet und die Datei SecondXML.xml generiert.

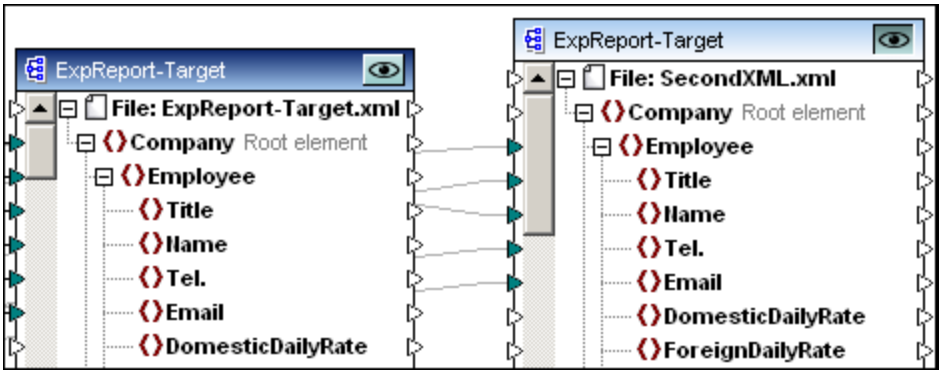
Um dies zu überprüfen, öffnen Sie die Datei **DoTransform.bat** (in dem von Ihnen definierten Ausgabeordner) und sehen Sie, in welcher Reihenfolge die Ausgabedateien generiert werden. ExpReport-Target.xml ist die erste Ausgabedatei, die von der Batch-Datei generiert wird, die zweite Datei ist SecondXML.xml.

```
@echo off

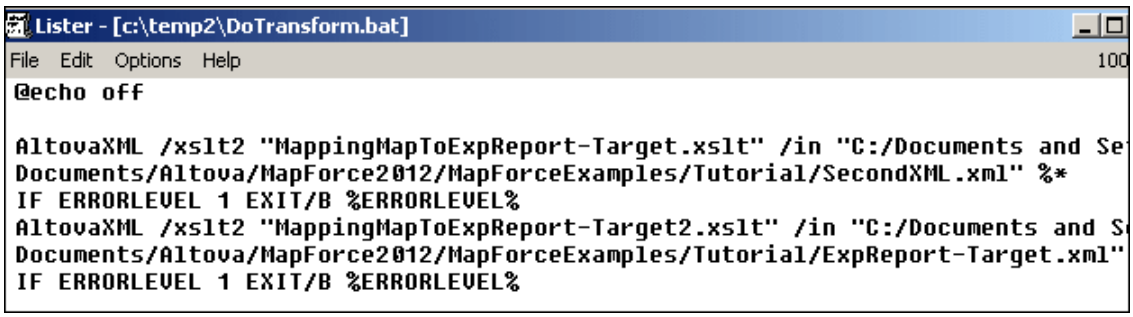
AltovaXML /xslt2 "MappingMapToExpReport-Target.xslt" /in "C:/Documents and Se
Documents/Altova/MapForce2012/MapForceExamples/Tutorial/ExpReport-Target.xml"
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
AltovaXML /xslt2 "MappingMapToExpReport-Target2.xslt" /in "C:/Documents and S
Documents/Altova/MapForce2012/MapForceExamples/Tutorial/SecondXML.xml" %*
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

Ändern der Reihenfolge der Mapping-Verarbeitung:

1. Klicken Sie auf die linke Zielkomponente und ziehen Sie sie tiefer als die rechte Komponente.



2. Generieren Sie Ihren Code erneut und werfen Sie einen Blick in die Datei DoTransform.bat.



Nun generiert die Batch-Datei als erstes die Datei SecondXML.xml und anschließend die Datei ExpReport-Target.xml.

Verkettete Mappings

Dieselbe oben beschriebene Verarbeitungsreihenfolge gilt auch für verkettete Mappings. Die verkettete Mapping-Gruppe wird jedoch als eine Einheit behandelt. Wenn Sie die Zwischen-, End- oder Zielkomponente 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.

© 2011 Altova GmbH

Kapitel 8

Globale Ressourcen

8 Globale Ressourcen

Globale Ressourcen stellen eine große Verbesserung hinsichtlich der Interoperabiltät zwischen den Produkten der Altova-Produktlinie dar und stehen derzeit in den folgenden Altova-Produkten zur Verfügung: XMLSpy, MapForce, StyleVision und DatabaseSpy. Benutzer der Altova MissionKit-Pakete haben in den entsprechenden Einzelprodukten Zugriff auf dieselben Funktionalitäten.

Allgemeine Verwendungsweise:

- Es können Arbeitsabläufe definiert werden, bei denen verschiedene Altova-Applikationen zur Verarbeitung der Daten verwendet werden.
- Eine Applikation kann eine Zielapplikation aufrufen, die Datenverarbeitung dort starten und die Daten wieder an die ursprüngliche Applikation zurückleiten.
- Definition von Input- und Output-Daten, Dateipfaden als globale Ressourcen.
- Wechseln zwischen globalen Ressourcen zur Laufzeit, um zwischen Ressourcen für Entwicklungszwecke und solchen für die reale Anwendung zu wechseln.
- Was wäre wenn-Szenarios für Zwecke der Qualitätskontrolle.

Der Standardpfad der Definitionsdatei für globale Ressourcen, **GlobalResources.xml**, lautet c:\Documents and Settings\UserName\My Documents\Altova\. Dies ist der Standardpfad für alle Altova-Applikationen, die globale Ressourcen verwenden können. Änderungen, die an globalen Ressourcen vorgenommen werden, stehen daher automatisch in allen Applikationen zur Verfügung. Der Pfad und Name der Datei kann geändert werden. Nähere Informationen dazu siehe Globale Ressourcen - Eigenschaften.

Allgemeine Funktionsweise:

- Globale Ressourcen werden in einer Applikation **definiert** und automatisch gespeichert.
- Globale Ressourcen werden Komponenten **zugewiesen**, deren Daten variabel sein sollen.
- Die globale Ressource wird in einer Applikation aufgerufen / **aktiviert**, sodass Sie zur Laufzeit Ressourcen wechseln können.

In diesem Abschnitt lesen Sie, wie globale Ressourcen mittels bestehender Mappings aus dem Ordner ...\MapForceExamples\Tutorial\ definiert und verwendet werden können.

So aktivieren Sie die Symbolleiste "Globale Ressourcen":

Wählen Sie den Menübefehl **Extras | Anpassen |** klicken Sie auf das Register **Symbolleiste** und aktivieren Sie das Kontrollkästchen "Globale Ressourcen". Daraufhin wird die Symbolleiste "Globale Ressourcen" angezeigt.

Über die Auswahlliste können Sie zwischen den verschiedenen Ressourcen wechseln, der Eintrag "Default" ist immer verfügbar.

Wenn Sie auf die Schaltfläche "Globale Ressourcen" klicken, wird das Dialogfeld "Globale Ressourcen" geöffnet (alternativ dazu Extras | Globale Ressourcen).

© 2011 Altova GmbH

8.1 Globale Ressoucen - Dateien

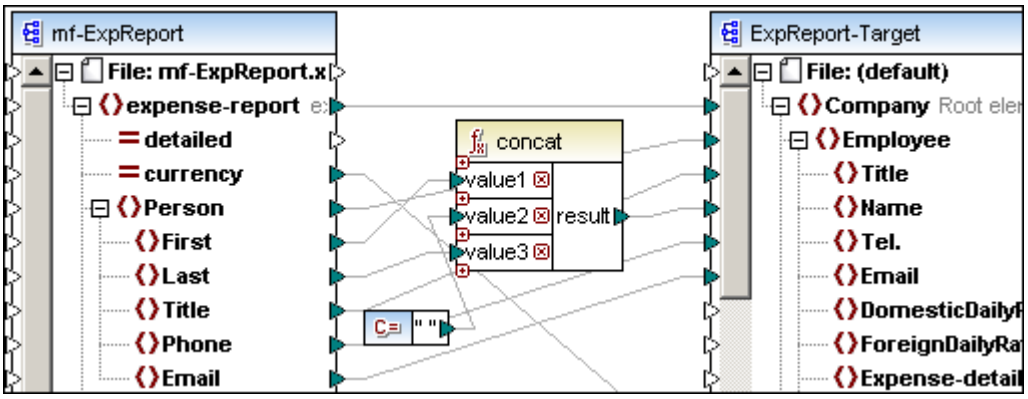
Globale Ressourcen in MapForce:

- Es kann jede Input/Output-Datei als globale Ressource definiert werden, z.B. XML, XML Schema, Text/CSV-Dateien.

Ziel dieses Abschnitts:

- die Input-Datei für die Quellkomponente, **mf-ExpReport**, zu einer globalen Ressource zu machen.
- zur Laufzeit zwischen den beiden XML-Dateien, die die Input-Daten der Quellkomponente bereitstellen, zu wechseln und die erzeugte XML-Ausgabe auf dem Register Ausgabevorschau anzuzeigen.

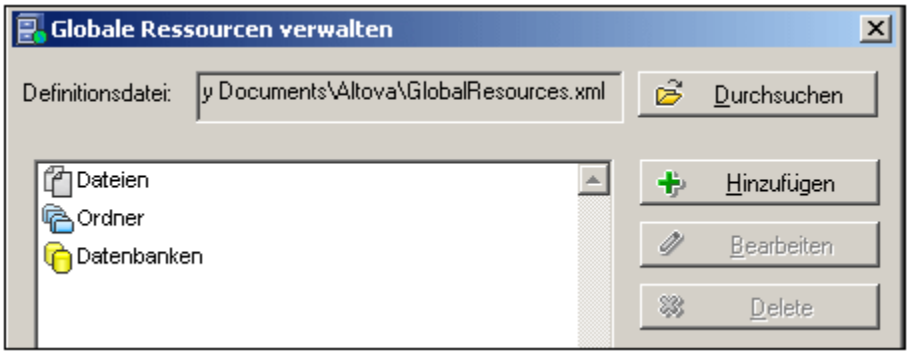
  In diesem Abschnitt wird die Datei **Tut-ExpReport.mfd** aus dem Ordner ... \MapForceExamples\Tutorial\ verwendet.



8.1.1 Definieren / Hinzufügen einer globalen Ressource

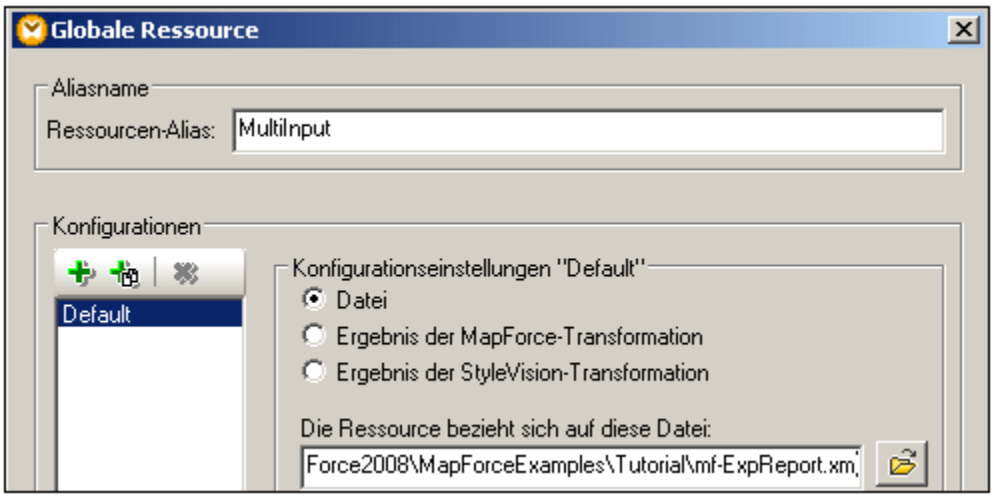
Definieren / Hinzufügen einer globalen Ressourcendatei

1. Klicken Sie auf die Schaltfläche "Globale Ressource" , um das Dialogfeld zu öffnen.



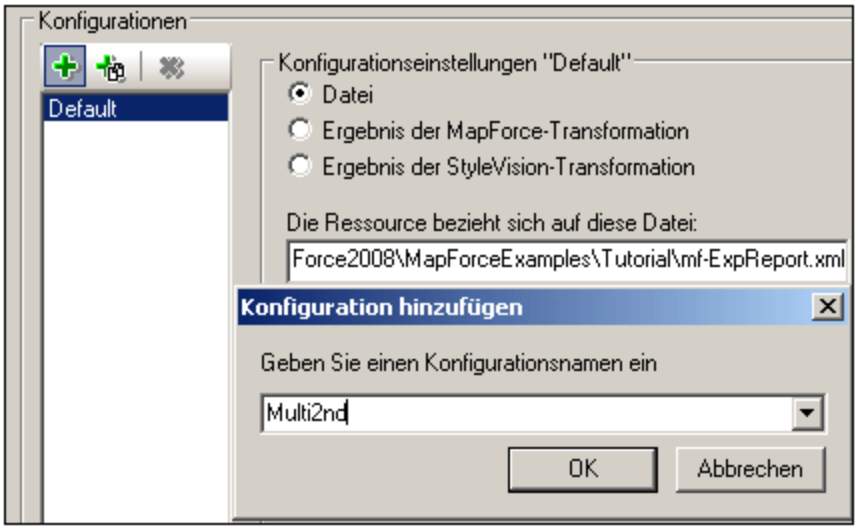
Hier sehen Sie eine leere globale Ressourcendatei.

2. Klicken Sie auf die Schaltfläche **Hinzufügen** und wählen Sie im Popup-Fenster den Eintrag **Datei.**
3. Geben Sie den Namen des Ressourcenalias ein, z.B. **MultiInput**.
4. Klicken Sie auf das Symbol zum Öffnen des Ordners und wählen Sie die XML-Datei aus, die als Standard (Default) Input-Datei dienen soll, z.B: **mf-ExpReport.xml**.

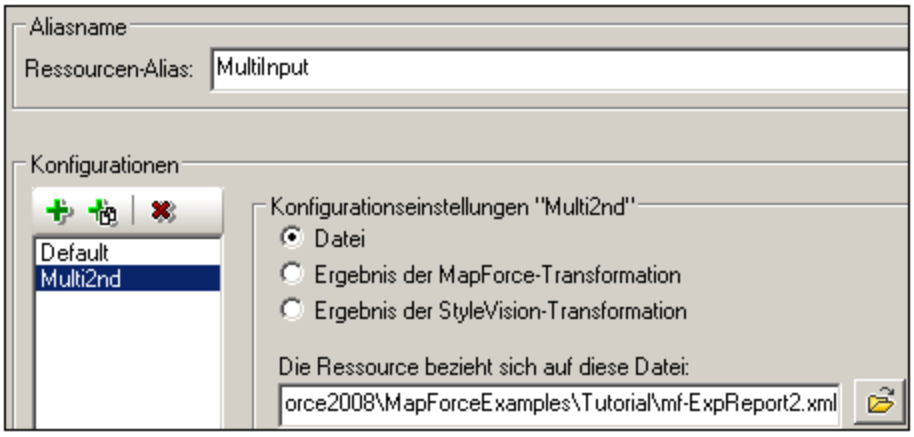


5. Klicken Sie in der Gruppe **Konfigurationen** auf die Schaltfläche "Hinzufügen" , um eine neue Konfiguration zum aktuellen Alias hinzuzufügen. Beachten Sie, dass Sie mit der Schaltfläche "Konfiguration kopieren" eine ausgewählte Konfiguration kopieren und unter einem neuen Namen speichern können.

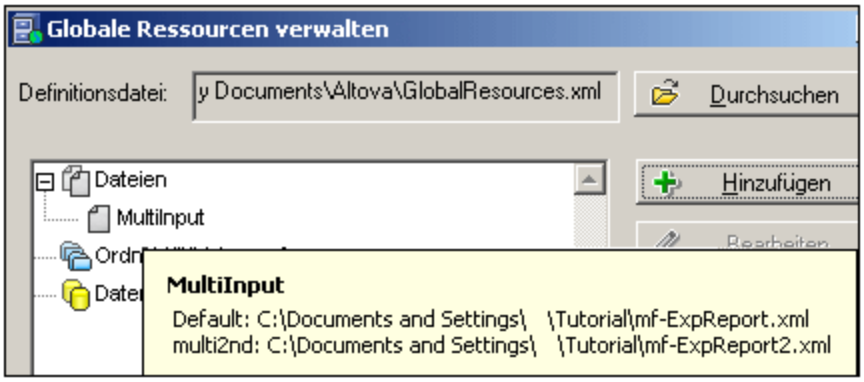
© 2011 Altova GmbH



6. Geben Sie einen Namen für die Konfiguration ein, **Multi2nd**, und klicken Sie zur Bestätigung auf OK.
   Multi2nd wurde nun zur Liste der Konfigurationen hinzugefügt.
7. Klicken Sie nochmals auf die Schaltfläche "Ordner öffnen" und wählen Sie die XML-Datei aus, die als Input-Datei für die Konfiguration multi2nd dienen soll, z.B. **mf-ExpReport2.xml**.



8. Klicken Sie auf OK, um die Definition der Ressource abzuschließen.
   Der Alias "MultiInput" wurde nun zum Abschnitt "Dateien" der globalen Ressourcen hinzugefügt.
   Wenn Sie den Mauszeiger über einen Aliaseintrag platzieren, wird ein Tooltip mit der Definition geöffnet.



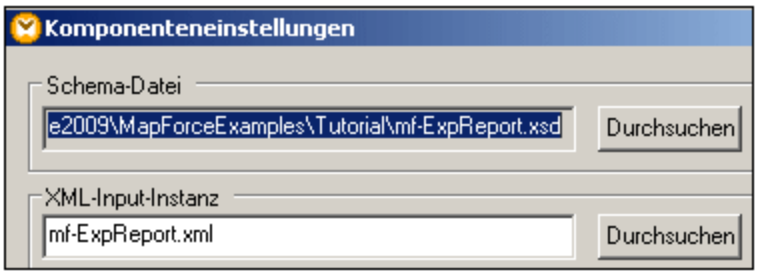
9. Klicken Sie zur Bestätigung auf OK.
   Damit ist die globale Ressource nun fertig definiert. Im nächsten Schritt wird eine globale Ressource nun einer Komponente zugewiesen.

 © 2011 Altova GmbH

8.1.2 Zuweisen einer globalen Ressource

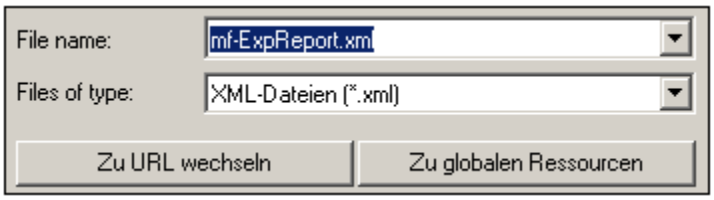
Zuweisen einer globalen Ressource zu einer Komponente
Wir müssen die globale Ressource nun der Komponente zuweisen, für die sie verwendet werden soll.

1. Doppelklicken Sie auf die Komponente **mf-ExpReport** und klicken Sie auf die Schaltfläche "Durchsuchen" neben dem Feld "XML-Input-Instanz".

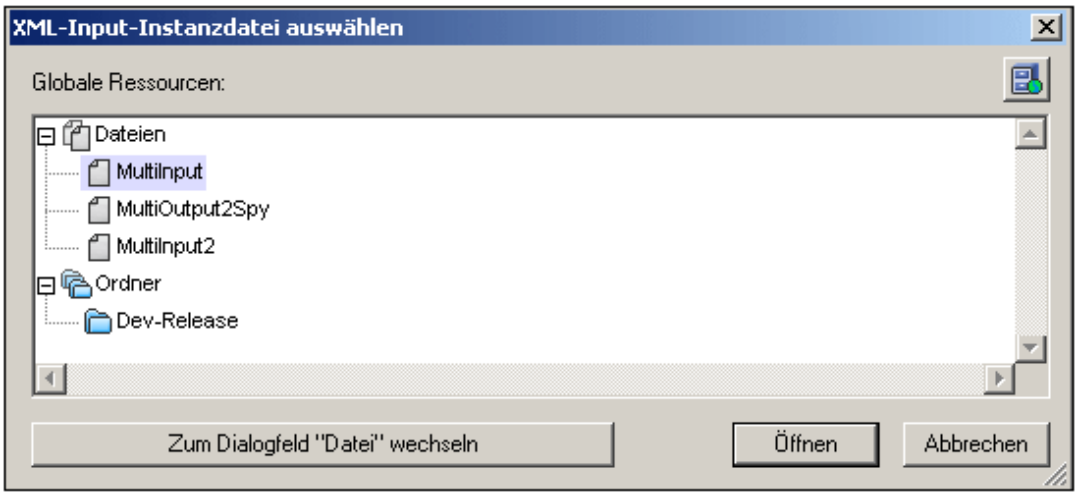


Daraufhin wird das Dialogfeld "XML-Input-Instanzdatei auswählen" geöffnet.

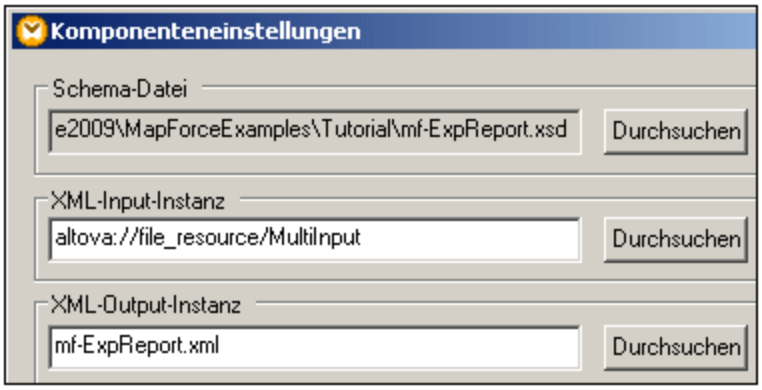
2. Klicken Sie auf die Schaltfläche **Zu globalen Ressourcen** am unteren Rand des Dialogfelds.



3. Klicken Sie auf die Ressource, die Sie zuweisen möchten, in diesem Fall auf **MultiInput,** und klicken Sie auf OK.



Die **XML-Input-Instanzdatei** enthält nun eine Referenz auf eine Ressource, also. **altova://file_resource/MultiInput**.



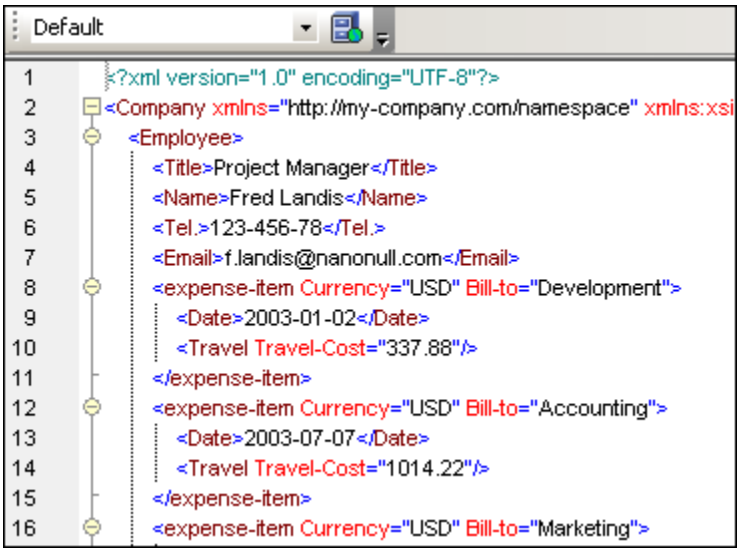
4. Klicken Sie auf OK, um die Zuweisung einer Ressource zu einer Komponente fertig zu stellen.
   Der nächste Schritt ist die Verwendung / Aktivierung einer globalen Ressource.

© 2011 Altova GmbH

8.1.3 Verwendung / Aktivierung einer globalen Komponente

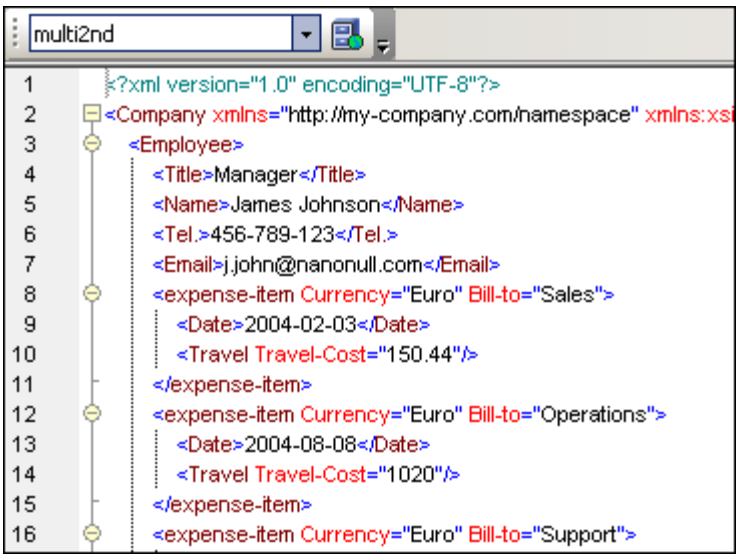
Verwenden / Aktivieren einer globalen Ressource
Zur Zeit ist die vorhin definierte Konfiguration **Default** für den Alias **MultiInput** aktiv. Dies sehen Sie daran, dass der Eintrag in der Symbolleiste "Globale Ressourcen" "Default" lautet.

1. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis des Mappings zu sehen.



2. Klicken Sie auf das Mapping-Register, um zur Mapping-Ansicht zurückzukehren.
3. Klicken Sie auf die Auswahlliste "Globale Ressourcen" und wählen Sie darin den Eintrag **multi2nd**.

4. Klicken Sie auf die Schaltfläche "Ausgabe", um das neue Ergebnis zu sehen. Die Datei mf-ExpReport2.xml wird nun als Quellkomponente für das Mapping verwendet und erzeugt nun eine andere Ausgabe.



Anmerkung:

Die **aktuell aktive** globale Ressource (**multi2nd** in der Symbolleiste "Globale Ressourcen") bestimmt das Ergebnis des Mappings. Dies ist auch der Fall, wenn Sie Code generieren.

© 2011 Altova GmbH

8.2 Globale Ressourcen - Ordner

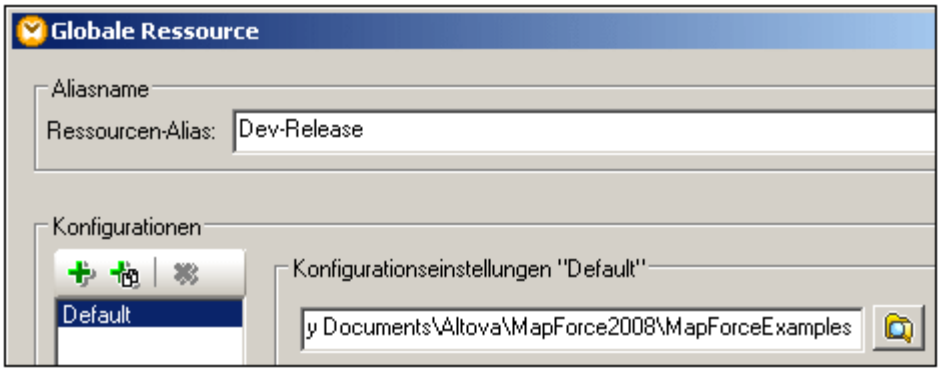
Auch Ordner können als globale Ressource definiert werden, d.h. in diesem Fall, dass Input-Komponenten Dateien enthalten können, die sich auf andere Ordner beziehen. Dies kann z.B. für Entwicklungs- und Release-Zyklen verwendet werden.

Ordner für Output-Komponenten in MapForce zu definieren, hat wenig Sinn, da Sie beim Generieren einer XSLT-Datei oder bei der Codegenerierung für andere Programmiersprachen immer nach den Zielordnern gefragt werden.

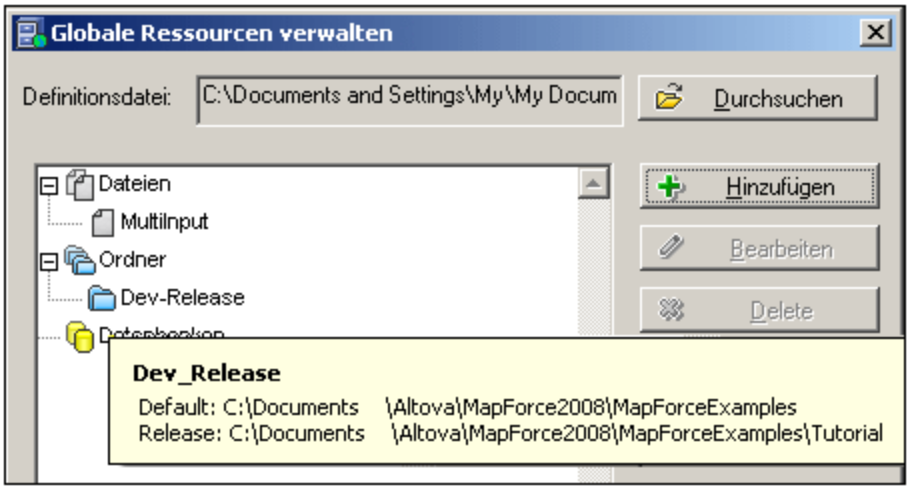
Die in diesem Abschnitt verwendete Mapping-Datei finden Sie unter dem Namen "**global-folder.mfd**" im Ordner ...\MapForceExamples\Tutorial\.

Definieren / Hinzufügen von Ordnern für globale Ressourcen

1. Klicken Sie auf die Schaltfläche "Globale Ressource" , um das Dialogfeld zu öffnen.
2. Klicken Sie auf die Schaltfläche **Hinzufügen** und wählen Sie im Popup-Fenster den Eintrag **Ordner**.
3. Geben Sie den Namen des Ressourcenalias ein, z.B. **Dev_Release**.
4. Klicken Sie auf die Schaltfläche "Ordner öffnen" und wählen Sie den Standard-Input-Ordner "Default", ...\MapForceExamples.



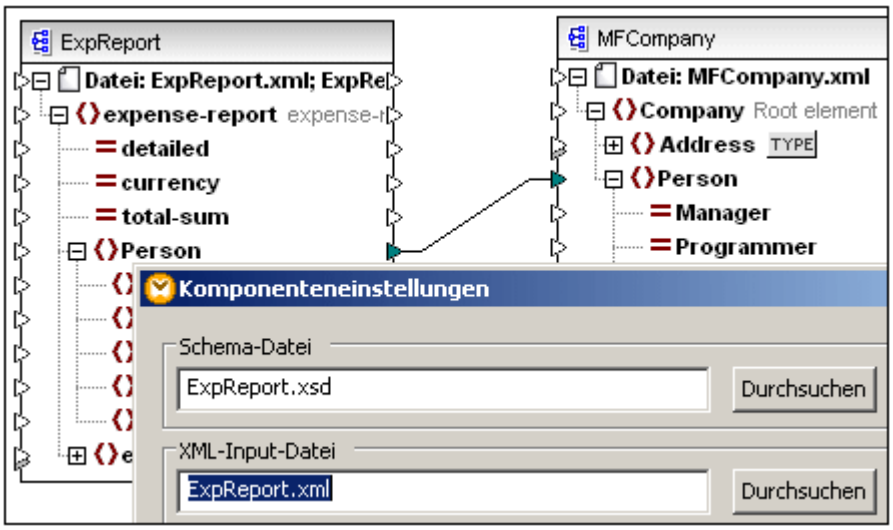
5. Klicken Sie in der Gruppe "Konfigurationen" auf die Schaltfläche "Hinzufügen" , um eine neue Konfiguration zum aktuellen Alias hinzuzufügen und geben Sie einen Namen dafür ein, z.B. Release. Beachten Sie, dass Sie mit der Schaltfläche "Konfiguration kopieren" eine ausgewählte Konfiguration kopieren und unter einem neuen Namen speichern können.
6. Klicken Sie auf die Schaltfläche "Ordner öffnen" und wählen Sie den Release Input-Ordner, ...\MapForceExamples\Tutorial.



7. Klicken Sie auf OK, um die Definition des globalen Ordners abzuschließen.

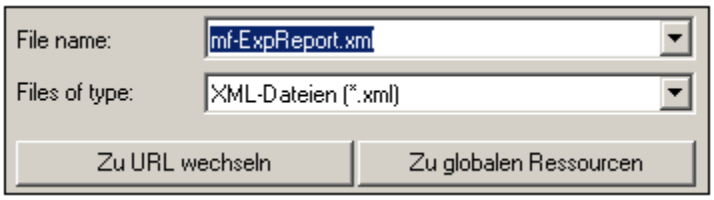
Zuweisen des Ordners für globale Ressourcen:

1. Doppelklicken Sie auf die Komponente **ExpReport** und klicken Sie auf die Schaltfläche "Durchsuchen" neben dem Feld **XML-Input-Datei**.



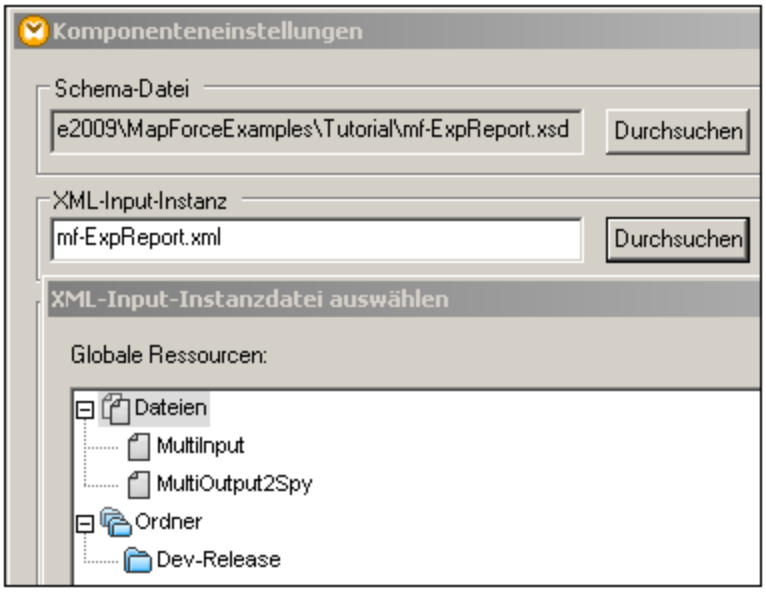
Daraufhin wird das Dialogfeld "XML-Input-Instanzdatei auswählen" geöffnet.

2. Klicken Sie auf die Schaltfläche **Zu globalen Ressourcen** am unteren Rand des Dialogfelds.



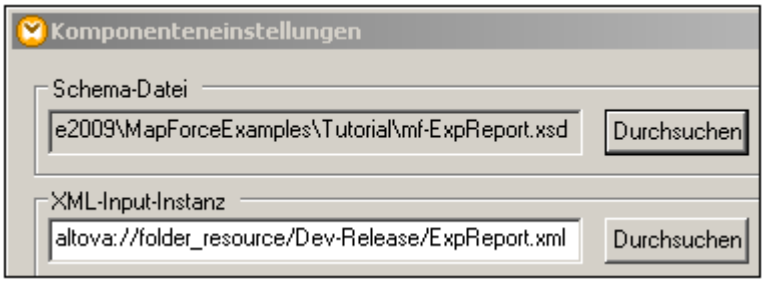
3. Klicken Sie auf die gewünschte Ressource, in diesem Fall auf **Dev_Release,** und bestätigen Sie mit OK.

 © 2011 Altova GmbH



Daraufhin erscheint das Dialogfeld "....öffnen".

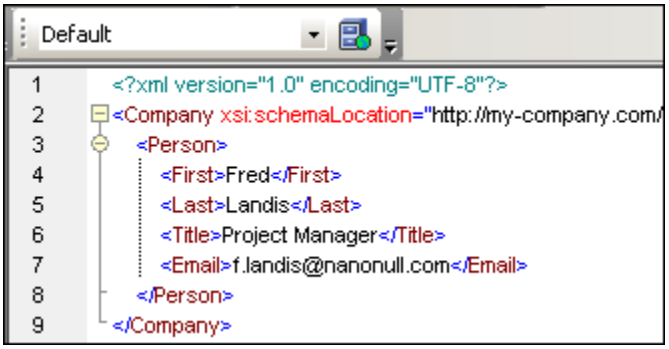
4. Wählen Sie in jedem der Ordner den Dateinamen, der sowohl als **Ressourcendatei** für die Entwicklung als auch für die Release verwendet werden soll, z.B. **ExpReport.xml** und klicken Sie auf OK, um die Zuweisung des Ressourceordners abzuschließen.



Beachten Sie, dass die Datei in beiden Verzeichnissen vorhanden ist, aber einen unterschiedlichen Inhalt hat.

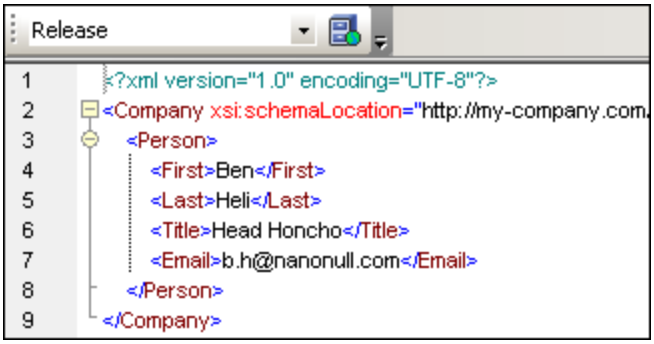
Wechseln des Ressourceordners zur Laufzeit:

1. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis der Transformation zu sehen.
   Beachten Sie, dass dies die Konfiguration / des Ordners **Default** in .../ MapforceExamples ist.



2. Klicken Sie auf das Mapping-Register, um zum Mapping-Fenster zurückzukehren.

3. Klicken Sie auf die Auswahlliste "Globale Ressource" und wählen Sie den Eintrag "**Release**".
3. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis bei Verwendung der Globalen Ressourcen "Release" zu sehen.



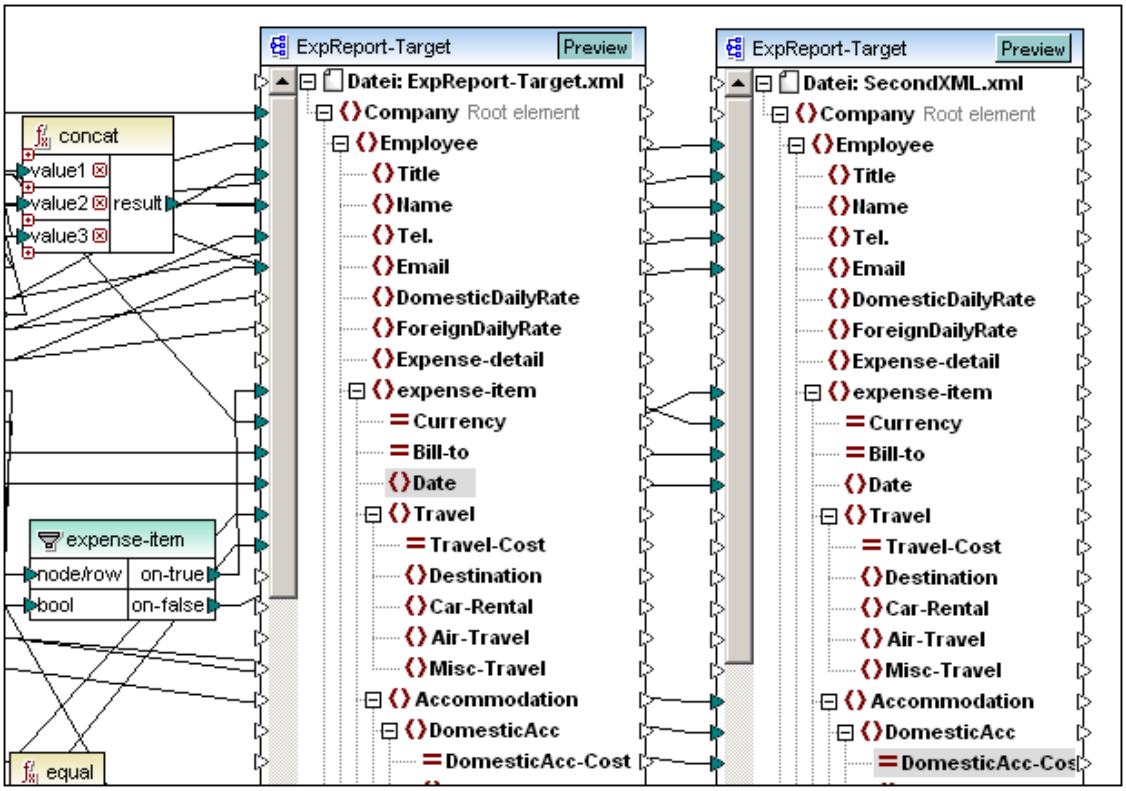
Es wird nun die Ausgabe für den Ordner "Release" aus .../MapforceExamples/Tutorial angezeigt.

 © 2011 Altova GmbH

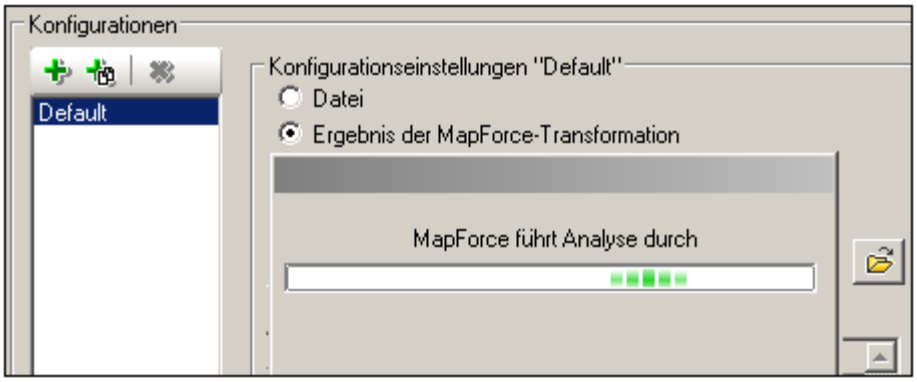
8.3 Globale Ressourcen - Arbeitsablauf in Applikationen

Ziel dieses Abschnittes ist ein Arbeitsablaufszenario zwischen zwei Altova-Applikationen zu erstellen. Der Arbeitsablauf beginnt in XMLSpy, das MapForce startet und von wo aus die generierte XML-Ausgabedatei zur weiteren Verarbeitung wieder an XMLSpy übergeben wird.

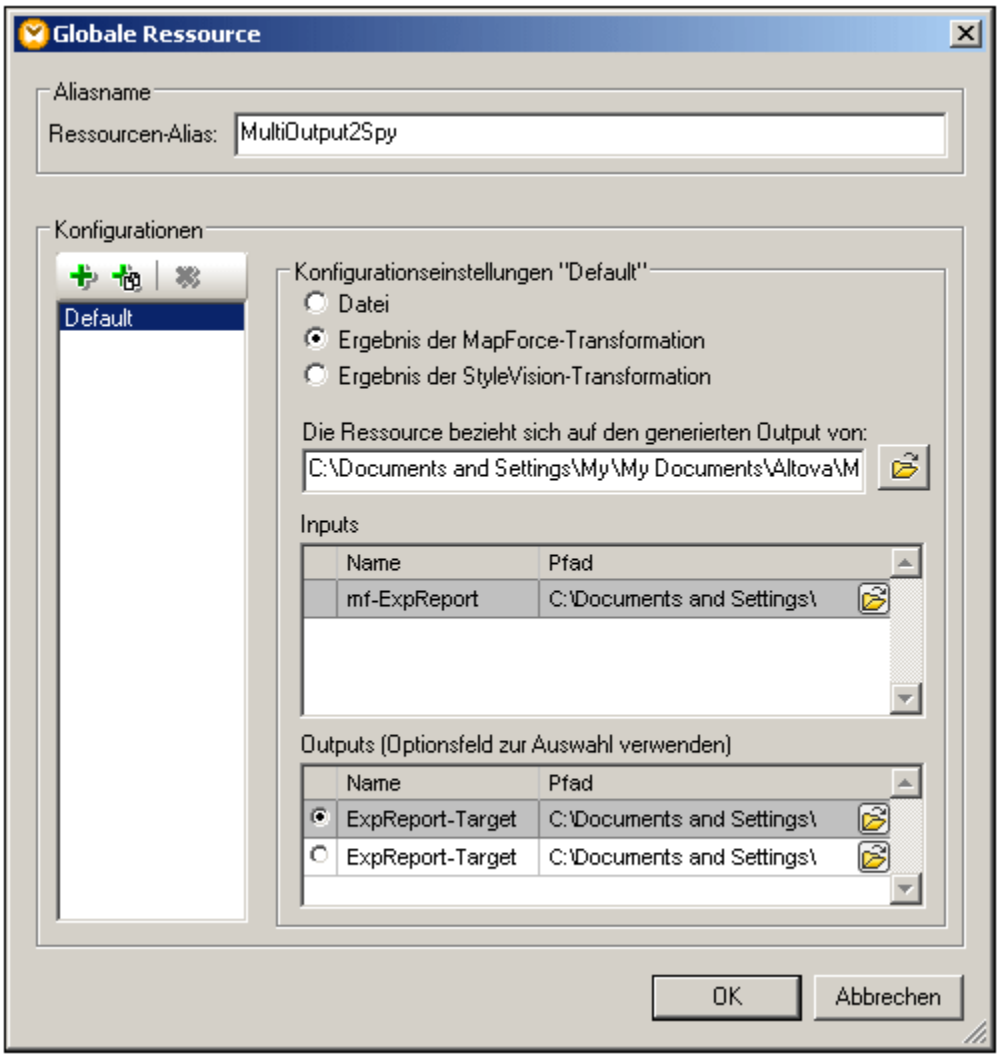
In diesem Mapping werden zwei Output-Komponenten verwendet, um zwei Arten einer gefilterten Ausgabe zu erzeugen; Reisekosten (Travel expenses) und andere Ausgaben (Non-travel expenses) der Reisekosten-Input-Datei. In diesem Abschnitt wird die Mapping-Datei **Tut-ExpReport-multi.mfd** aus dem Ordner ...\MapForceExamples\Tutorial\ verwendet.



1. Klicken Sie auf die Schaltfläche "Globale Ressource" , um das Dialogfeld zu öffnen.
2. Klicken Sie auf die Schaltfläche "Hinzufügen" und wählen Sie im Popup-Fenster den Eintrag "Datei".
3. Geben Sie den Namen des Ressourcealias ein, z.B. **MultiOutput2Spy**
4. Klicken Sie auf das Optionsfeld "**Ergebnis der MapForce-Transformation**" und klicken Sie auf die Schaltfläche "Datei öffnen".
5. Wählen Sie das Mapping **Tut-ExpReport-multi.mfd** aus.



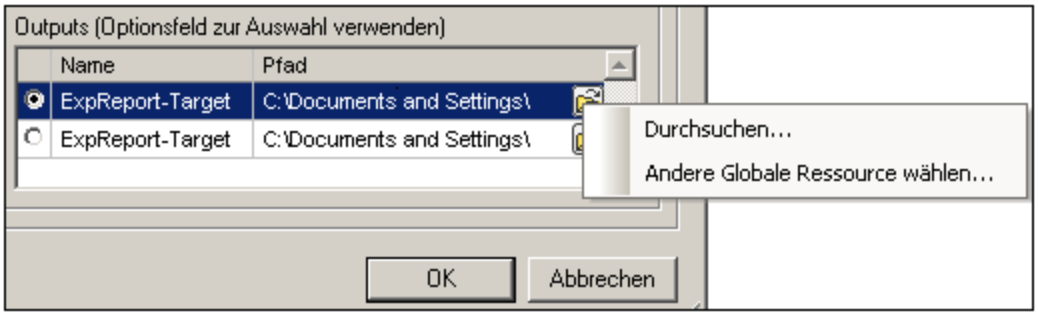
MapForce analysiert das Mapping und zeigt die Input- und Output-Dateien in separaten Listenfeldern an.



6. Klicken Sie auf den Optionsfeldeintrag im Abschnitt **Outputs,** falls er noch nicht ausgewählt ist.
   Beachten Sie, dass der Name der Ausgabedatei **ExpReport-Target.xml** ist, und dass wir zur Zeit die Standardkonfiguration (Default) definieren.
7. Klicken Sie auf die Schaltfläche , um einen **neuen** Pfad für die Ausgabedatei zu

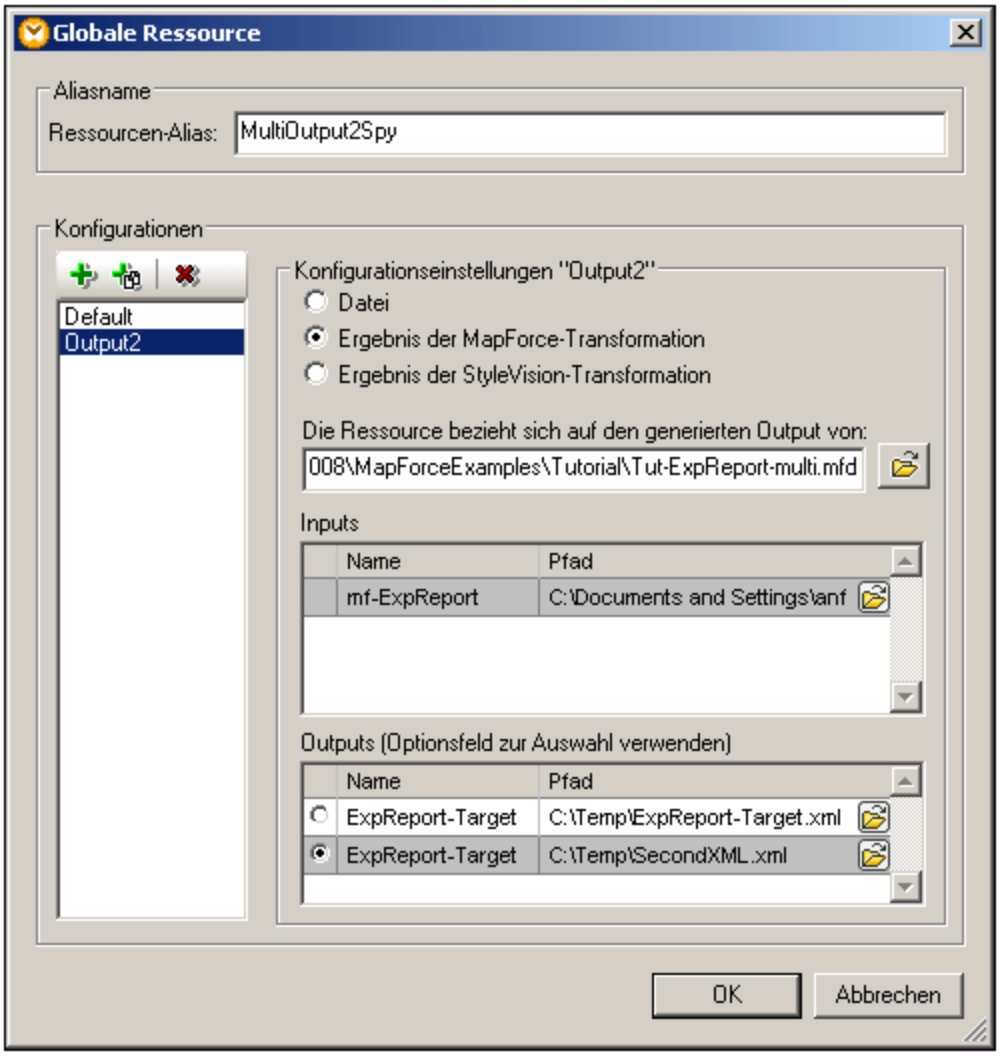
 © 2011 Altova GmbH

definieren und wählen Sie im Popup-Menü **Durchsuchen...**.

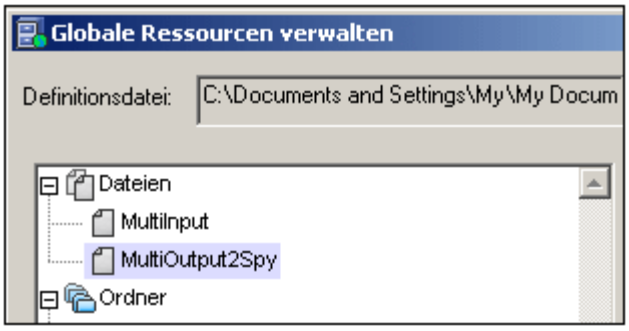


8. Geben Sie den neuen Ausgabepfad ein, z.B. C:\Temp und klicken Sie auf die Schaltfläche "Speichern". Dieser Pfad kann sich von dem in den Komponenteneinstellungen definierten unterscheiden.
9. Klicken Sie in der Gruppe "Konfigurationen" auf die Schaltfläche "Hinzufügen" (dieses Dialogfelds) um eine neue Konfiguration zum Ressourcenalias hinzuzufügen.
10. Geben Sie den Namen der Konfiguration ein, z.B. **Output2,** klicken Sie auf das Symbol "Datei öffnen" und aktivieren Sie das untere Optionsfeld im Listenfeld "Outputs". Beachten Sie, dass der Name der Ausgabedatei **SecondXML.xml** ist.

11. Klicken Sie auf die Schaltfläche "Speichern unter" **,** um den neuen Pfad der Ausgabedatei zu speichern, z.B. C:\Temp.
    Anmerkung: Wenn Sie im Popup-Fenster auf die Schaltfläche "**Andere Globale Ressource wählen...**" klicken, können Sie die MapForce-Ausgabe als globale Ressource definieren, d.h. die Ausgabe wird in einer Datei gespeichert, die die globale Ressource referenziert.



12. Klicken Sie auf OK, um die neuen globalen Ressourcen zu speichern.
Der neue Ressourcealias **MultiOutput2Spy** wurde nun zu Definitionsdatei für globale Ressourcen hinzugefügt.



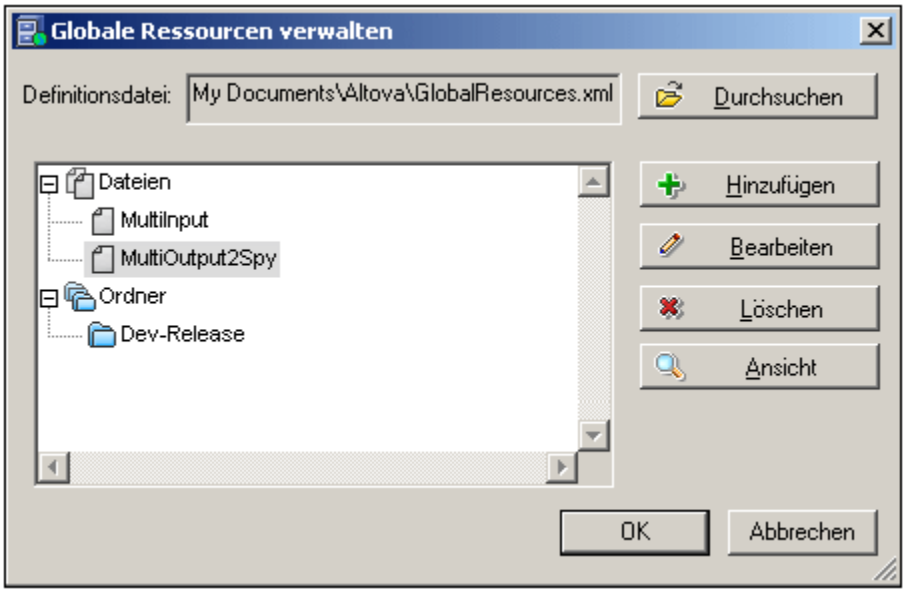
13. Klicken Sie auf OK., um die Definition abzuschließen.

 © 2011 Altova GmbH

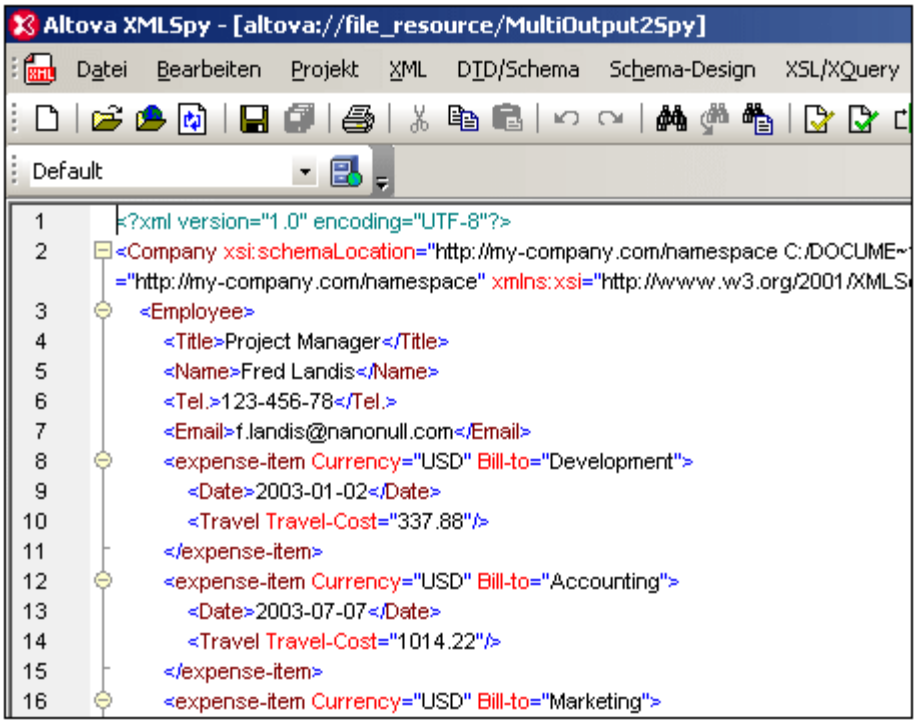
8.3.1 Starten des Applikationsarbeitsablaufs

In diesem Abschnitt wird beschrieben, wie die globale Ressource in XMLSpy aktiviert wird und wie die MapForce Transformation, die daraus resultiert, wieder an XMLSpy zurückgeleitet wird.

1. Starten Sie XMLSpy und schließen Sie MapForce, falls Sie das Programm geöffnet haben, um besser zu sehen, wie die beiden Applikationen interagieren.
2. Wählen Sie in XMLSpy die Menüoption **Extras | Globale Ressourcen**.
3. Wählen Sie den Eintrag **MultiOutput2Spy** aus und klicken Sie auf die Schaltfläche **Ansicht**.



Es erscheint ein Meldungsfeld, in dem Sie darüber informiert werden, dass MapForce eine Transformation durchführt. Das Ergebnis der Transformation wird im Fenster "Textansicht" angezeigt.

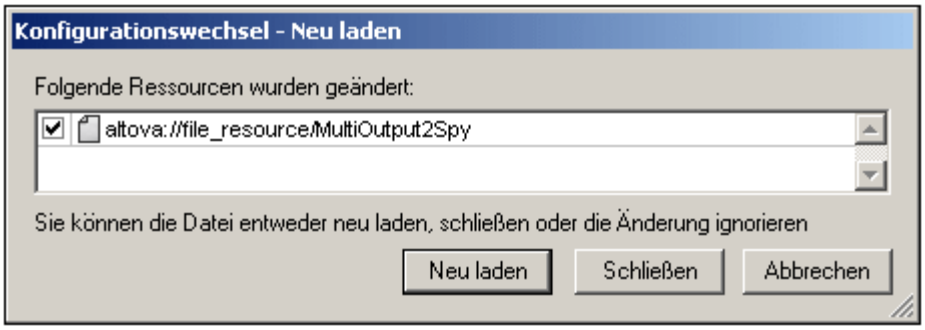


Anmerkung:

- Die derzeit ausgewählte Konfiguration ist "**Default**".
- Der Name des Ressourcealias steht in der Applikationstitelleiste **altova:// file_resource/MultiOutput2Spy**.
- Die Ausgabedatei wurde als "Untitled1.xml" zur weiteren Verarbeitung geöffnet.
- Die Datei **ExpReport-Target.xml** wurde in den Ordner C:\Temp kopiert.

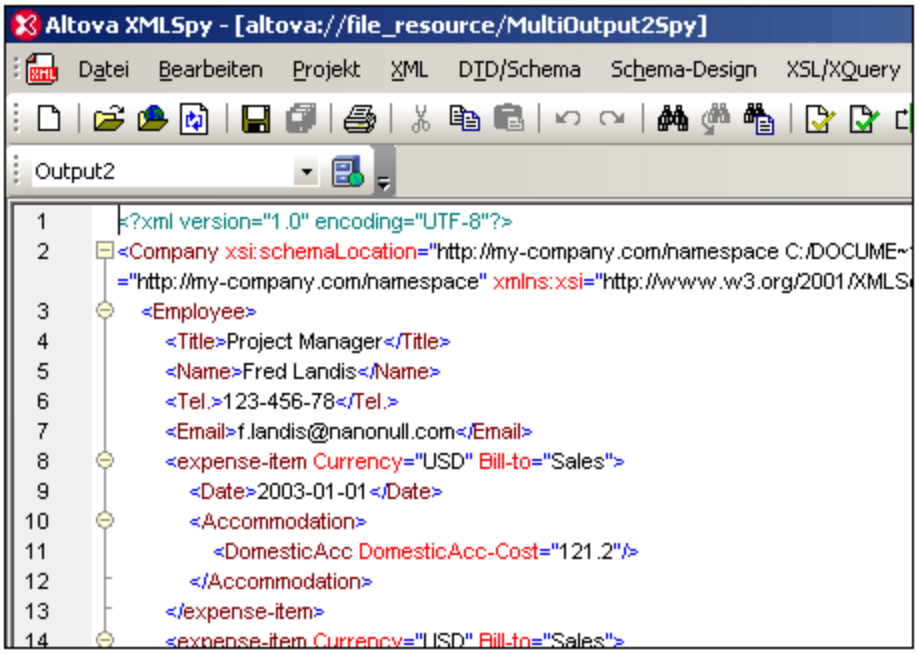
So rufen Sie die Ausgabe für die anderen Kosten (Nicht-Reisekosten) ab:

1. Klicken Sie auf die Auswahlliste "Globale Ressourcen" und wählen Sie "**Output2**". Daraufhin erscheint ein entsprechendes Meldungsfeld.



2. Klicken Sie auf **Neu laden**, um die zweite durch die Ressource definierte Ausgabedatei abzurufen.

 © 2011 Altova GmbH



Das Ergebnis der Transformation wird im Fenster "Textansicht" angezeigt und überschreibt die vorherige Datei MultiOutput2Spy.xml.

Anmerkung:

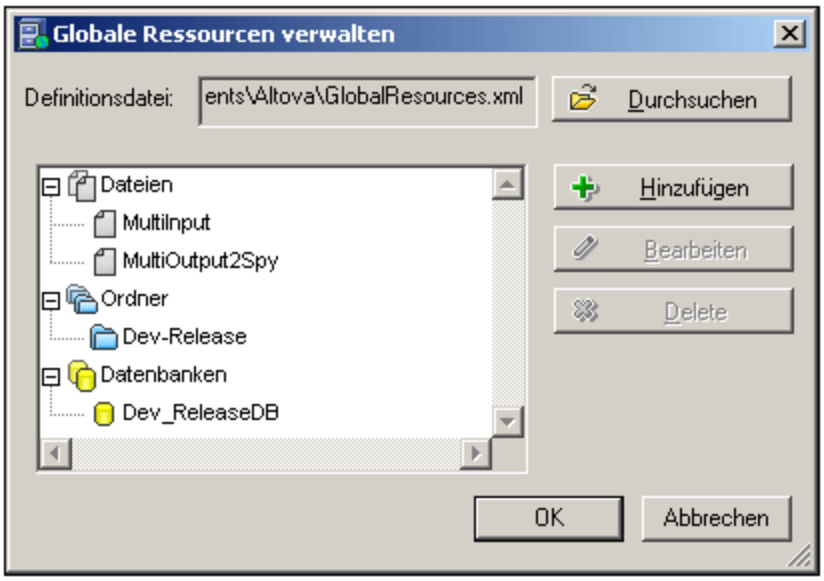
- Die aktuell ausgewählte Konfiguration ist "**Output2**"
- Die Ausgabedatei wurde als "Untitled1.xml" für die weitere Verarbeitung geöffnet.
- Die Datei **SecondXML.xml** wurde in den Ordner C:\Temp kopiert.

8.4 Globale Ressourcen - Eigenschaften

Die XML-Datei für globale Ressourcen
Die Definitionen der globalen Ressourcen sind in einer XML-Datei gespeichert. Standardmäßig hat diese XML-Datei den Namen `GlobalResources.xml` und ist im Ordner `C:\Documents and Settings\<username>\My Documents\Altova\` gespeichert. Diese Datei ist für alle Altova-Applikationen als XML-Standarddatei für globale Ressourcen definiert. Aus diesem Grund steht eine globale Ressource, die in jeder beliebigen Altova-Applikation definiert wurde, allen Altova-Applikationen zur Verfügung - vorausgesetzt diese Datei wird in diesen Applikationen verwendet.

Sie können die Datei umbenennen und in einem beliebigen Ordner speichern. Sie können daher mehrere XML-Dateien für globale Ressourcen haben; es kann jedoch immer nur eine dieser Dateien aktiv sein und nur die in dieser Datei enthaltenen Definitionen stehen der Applikation zur Verfügung.

Um eine XML-Datei für globale Ressourcen als aktive Datei auszuwählen, klicken Sie auf die **Durchsuchen**-Schaltfläche des Felds "Definitionsdatei" und wählen Sie die gewünschte im Dialogfeld "Öffnen" aus.



Verwalten globaler Ressourcen: Hinzufügen, Bearbeiten, Löschen
Im Dialogfeld "Globale Ressoucen" können Sie eine globale Ressource zur ausgewählten XML-Datei für globale Ressourcen hinzufügen oder eine ausgewählte globale Ressource löschen oder bearbeiten. In der XML-Datei für globale Ressourcen werden die von Ihnen hinzugefügten Aliasse in drei Abschnitten aufgelistet: Dateien, Ordner

So fügen Sie eine globale Ressource hinzu:
Klicken Sie auf die Schaltfläche **Hinzufügen** und definieren Sie die globale Ressource im Dialogfeld "Globale Ressource", das angezeigt wird. Nachdem Sie eine globale Ressource definiert und gespeichert haben, wird die globale Ressource (d.h. der Alias) zur Liste der globalen Definitionen in der ausgewählten XML-Datei für globale Ressourcen hinzugefügt.

So bearbeiten Sie eine globale Ressource:
Um eine globale Ressource zu bearbeiten, wählen Sie sie aus und klicken Sie auf **Bearbeiten**. Daraufhin wird das Dialogfeld "Globale Ressource" angezeigt, in dem Sie die nötigen

 © 2011 Altova GmbH

Änderungen vornehmen können.

So löschen Sie eine globale Ressource:
Um eine globale Ressource zu löschen, wählen Sie sie aus und klicken Sie auf **Löschen**.

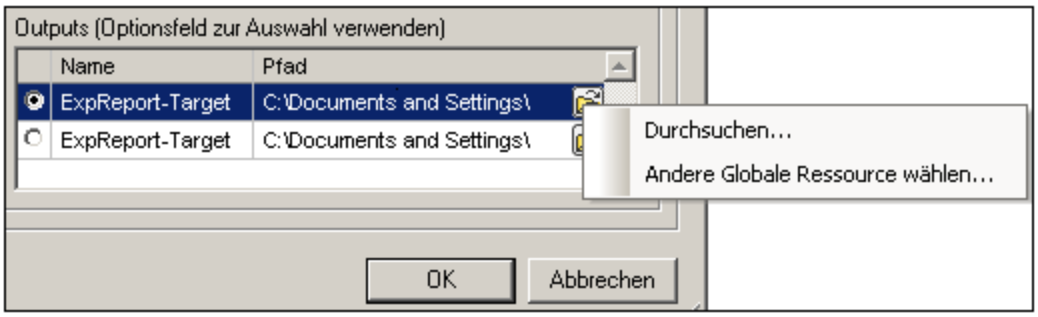
So zeigen Sie das Ergebnis eines Applikationsarbeitsablaufs an:
Wenn die aufrufende Applikation, z.B. XMLSpy, eine andere Applikation, z.B. MapForce aufruft, so steht im Dialogfeld "Globale Ressourcen verwalten" eine Schaltfläche "Ansicht" zur Verfügung.

Wenn Sie auf die Schaltfläche "Ansicht" klicken, wird die Auswirkung der gerade ausgewählten globalen Ressource in der aufrufenden Applikation angezeigt. Ein Beispiel dazu finden Sie unter Globale Ressourcen - Arbeitsablauf in Applikationen.

So speichern Sie Änderungen, die im Dialogfeld "Globale Ressourcen verwalten" vorgenommen wurden:
Nachdem Sie mit dem Hinzufügen, Bearbeiten oder Löschen globaler Ressourcen fertig sind, klicken Sie im Dialogfeld "Globale Ressourcen" auf **OK**, um Ihre Änderungen an der XML-Datei für globale Ressourcen zu speichern.

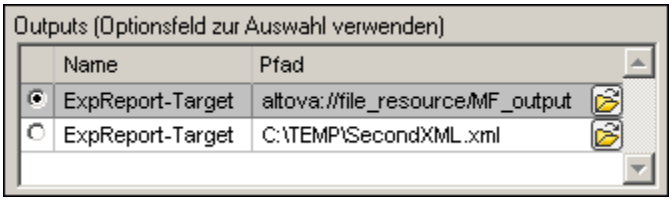
Anmerkung: Aliasressourcenamen müssen **innerhalb** von jedem der Abschnitte Dateien, Ordner eindeutig sein. Sie können jedoch in zwei unterschiedlichen Abschnitten einen identischen Aliasnamen definieren. So kann es z.B. sowohl im Abschnitt Dateien als auch im Abschnitt Ordner einen Alias "multiInput" geben.

Auswahl von Ergebnissen vom MapForce-Transformationen als globale Ressource
In einer MapForce Transformation, die mehrere Output-Komponenten hat, können Sie auswählen, welche der Output-Dateien für die globale Ressource verwendet werden soll, indem Sie das Optionsfeld für diese Datei aktivieren.



Die durch das Mapping generierte **Ausgabedatei** kann gespeichert werden:

- über den Eintrag zur Auswahl einer anderen globalen Ressource im Popup Menü, der angezeigt wird als **altova://file_resource/MF_output.** Die Ausgabe wird in einer Datei gespeichert, die von der globalen Ressource referenziert wird.



- als Datei (über die Schaltfläche), angezeigt als C:\TEMP\Second.xml

  Wenn keine dieser Optionen ausgewählt wird, wird eine **temporäre** XML-Datei erstellt, wenn die globale Ressource verwendet wird.

Festlegen, welche Ressource zur Laufzeit verwendet werden soll
Es gibt zwei applikationsweite Auswahloptionen, mit denen festgelegt wird, welche globalen Ressourcen verwendet werden und welche davon zu einem bestimmten Zeitpunkt tatsächlich verwendet wird:

- *die aktive XML-Datei für globale Ressourcen* wird im Dialogfeld "Globale Ressource" ausgewählt und kann jederzeit geändert werden. In diesem Fall werden die Definitionen der zuvor aktiven Datei sofort durch die der neuen aktiven Datei ersetzt.

  Die aktive XML-Datei für globale Ressourcen legt also Folgendes fest: (i) welche globale Ressource zugewiesen werden kann und (ii) welche globalen Ressourcen angezeigt werden können (Wenn z.B. eine globale Ressource in einer XML-Datei für globale Ressourcen zugewiesen ist, es aber in der aktiven XML-Datei für globale Ressourcen keine globale Ressource dieses Namens gibt, dann kann die zugewiesene globale Ressource (der Alias) nicht angezeigt werden).

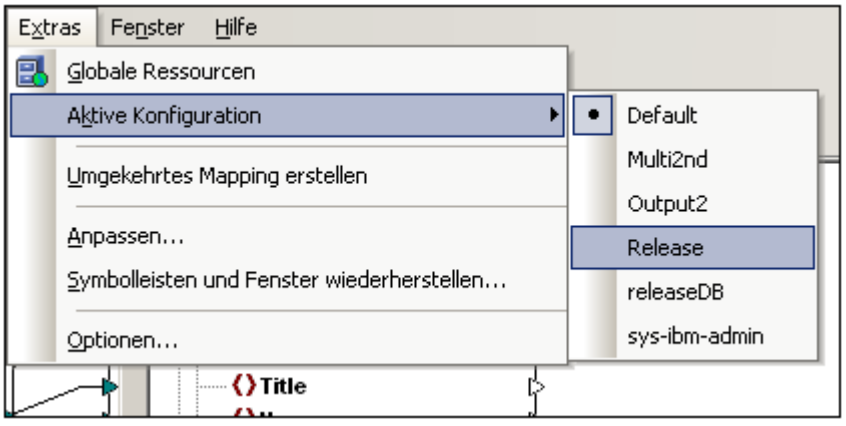
- *Die aktive Konfiguration* wird über den Menübefehl **Extras | Aktive Konfiguration** oder über die Symbolleiste "Globale Ressourcen" ausgewählt. Bei Auswahl dieses Befehls (bzw. der Dropdown-Liste in der Symbolleiste) wird eine Liste der Konfigurationen für alle Aliasse angezeigt.

  Bei Auswahl einer Konfiguration wird diese in der gesamten Applikation aktiv. D.h. überall, wo eine globale Ressource (oder ein Alias) verwendet wird, wird die Ressource, die der aktiven Konfiguration der einzelnen verwendeten Aliasse entspricht, geladen.

  Die aktive Konfiguration wird auf alle verwendeten Aliasse angewendet. Wenn ein Alias keine Konfiguration mit dem Namen der aktiven Konfiguration hat, wird die **Standardkonfiguration** dieses Aliasses verwendet. Die aktive Konfiguration spielt beim Zuweisen von Ressourcen keine Rolle; Sie ist nur dann von Bedeutung, wenn die Ressourcen tatsächlich verwendet werden

Wechseln von Ressourcen / Konfigurationen
Ressourcen können durch Auswahl eines anderen Konfigurationsnamen gewechselt werden. Dies kann auf zwei Arten erfolgen:

- Wenn Sie den Mauszeiger über den Menübefehl **Extras | Aktive Konfiguration** platzieren, wird ein Untermenü mit einer Liste aller Konfigurationen in der XML-Datei der globalen Ressourcen angezeigt. Wählen Sie die gewünschte Konfiguration aus dem Untermenü aus.



- Wählen Sie in der Auswahlliste der Symbolleiste "Globale Ressourcen" die gewünschte Konfiguration aus. Die Symbolleiste "Globale Ressourcen" kann über den Menübefehl

 © 2011 Altova GmbH

Extras | Anpassen, Klicken auf die Schaltfläche **Symbolleiste** und anschließendes Aktivieren/Deaktivieren des Kontrollkästchens "Globale Ressourcen" ein- bzw. ausgeblendet werden.

Kapitel 9

Dynamische Input/Output-Dateien pro Komponente

9 Dynamische Input/Output-Dateien pro Komponente

MapForce kann mehrere Input-/Output-Dateien pro Komponente verarbeiten und daher alle Dateien in einem Verzeichnis oder eine Teilmenge davon verarbeiten. Dies geschieht durch Verwendung von Platzhalterzeichen in der Input-Komponente.

Im Beispiel im Tutorial wird gezeigt, wie zwei XML-Input-Dateien von einer Quellkomponente verarbeitet werden und wie die Zielkomponente zwei XML-Dokumente ausgibt.

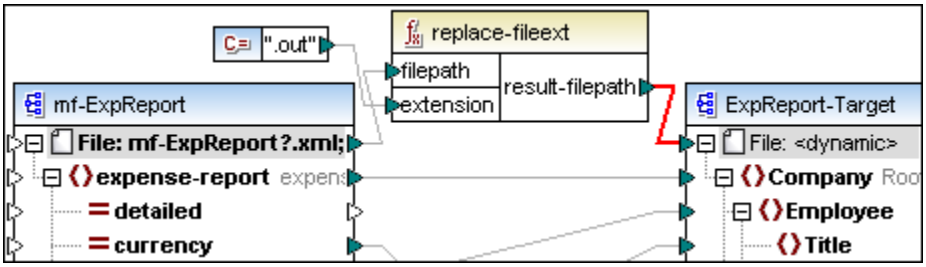
Für die folgenden Komponenten können mehrere Input- / Output-Dateien definiert werden:

- XML-Dateien

Beachten Sie bitte das Datenelement **Datei:** am oberen Rand der oben genannten Komponenten:

- Das Datenelement **Datei:mf-ExpReport.xml** von mf-ExpReport enthält als Eintrag die XML-Input-Datei. Dieser wird automatisch ausgefüllt, wenn Sie einer XML-Schema-Datei eine XML-Instanzdatei zuweisen. (Wenn eine Ausgabedatei definiert wurde, so wird auch der Name der Ausgabedatei angezeigt.)
- Das Datenelement **Datei: (Standard)** von ExpReport-Target zeigt an, dass der XML-Schema-Komponente keine Ausgabe-Instanzdatei zugewiesen wurde, als sie eingefügt wurde, d.h. die XML-Output-Datei ist leer. Daher wird bei der Ausführung des Mappings ein Standardwert verwendet.

  Die Unterstützung dynamischer Dateinamen wird aktiviert, indem ein String, der einen Dateinamen enthält, auf das Datenelement File gemappt wird. Wenn die Komponente als Input-Komponente verwendet wird, kann der Dateiname Platzhalterzeichen enthalten. Siehe auch: Behandlung relativer Pfade.
- Das Datenelement **Datei: <dynamic>** wird angezeigt, wenn eine Verbindung zum Datenelement "File" vorhanden ist, d.h. es werden nun mehrere Dateien unterstützt.
- Die Funktion **replace-fileext** konvertiert die Erweiterung .xml für die dynamischen Zieldateien in .out.



Dynamische Unterstützung/Unterstützung mehrerer Dateien und Unterstützung von Platzhalterzeichen in von MapForce unterstützten Sprachen:

 © 2011 Altova GmbH

Zielsprache	Dynamischer Input-Dateiname	Unterstützung von Platzhaltern für Input-Dateinamen	Dynamischer Output-Dateiname
XSLT 1.0	*	wird von XSLT 1.0 nicht unterstützt	wird von XSLT 1.0 nicht unterstützt

* unterstützt

(1) Verwendet die Funktion **fn:collection**. Die Implementierung im **Altova** XSLT 2.0- und XQuery-Prozessor löst Platzhalterzeichen auf. Andere Prozessoren verhalten sich unter Umständen anders.

Die Platzhalterzeichen * und ? werden bei Eingabe ins Dialogfeld "Komponenteneinstellungen" und beim Mappen eines Strings auf den Node "File: Name" aufgelöst.

Informationen zum Transformieren von XSLT 1.0/2.0- und XQuery-Code mit AltovaXML finden Sie unter Generieren von XSLT 1.0- und 2.0-Code

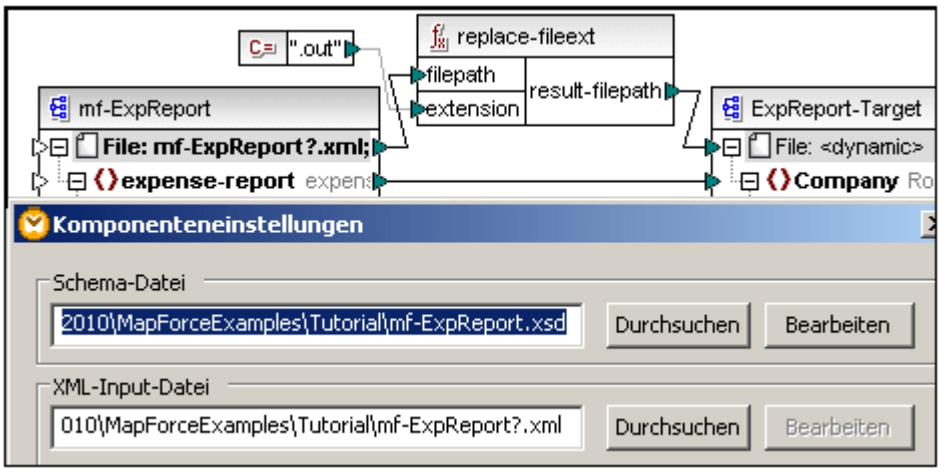
9.1 Dynamische Dateinamen - Input / Output

Durch dynamisches Mapping von Dateinamen innerhalb des Mappings können Sie:

- eine Mapping-Applikation generieren, in der die Input- und Output-Dateinamen während der Laufzeit generiert werden
- eine Gruppe von Dateien in ein anderes Format konvertieren (viele-in-viele)
- eine große Datei (oder Datenbank) in mehrere kleinere Abschnitte/Teile aufteilen
- mehrere Dateien in einer großen Datei zusammenführen (oder zu einer Datenbank zusammenführen)

Um mehrere Input-Dateien zu verarbeiten, stehen folgende Möglichkeiten zur Verfügung:

- Geben Sie im Dialogfeld **Komponenteneinstellungen** einen Dateipfad mit Platzhalterzeichen (* oder ?) als **Input-Datei** ein. Alle übereinstimmenden Dateien werden verarbeitet. Im Beispiel unten wird in der XML-Input-Datei das Platzhalterzeichen **?** verwendet, um alle Dateien, die mit mf-ExpReport beginnen und ein weiteres Zeichen danach aufweisen, zu mappen. Im Ordner ...\Tutorial gibt es davon zwei.

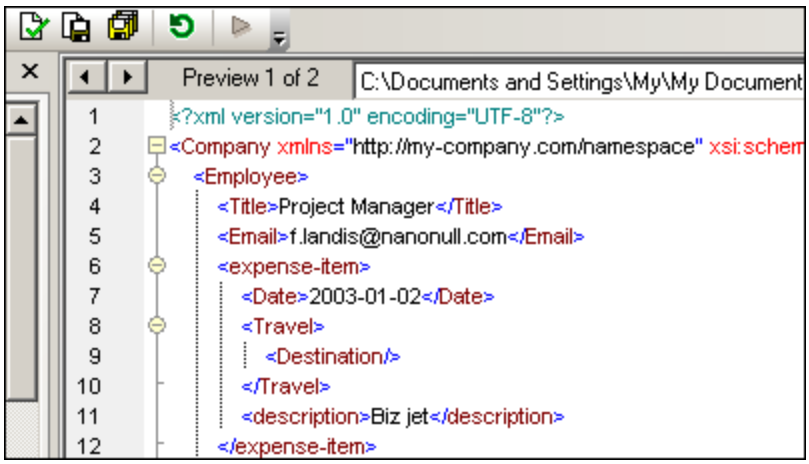


- Mappen Sie eine **Reihe** von Strings auf den *File*-Node der Quellkomponente. Jeder String in der Sequenz steht für einen Dateinamen. Die Strings können auch Platzhalterzeichen enthalten. Diese werden automatisch aufgelöst.

  Eine Reihe von Dateinamen kann aus folgenden Quellen stammen:
  - einer XML-Datei
  - .

Vorschau auf dynamische Input- / Output-Mappings
Wenn Sie auf die Schaltfläche "Ausgabe" klicken, werden die Mapping-Ergebnisse in einem Vorschaufenster angezeigt. Wenn beim Mapping mehrere Ausgabedateien erzeugt werden, wie in der Abbildung unten, Vorschau 1 von 2, gezeigt, hat jede Datei auf dem Ausgaberegister ihr eigenes nummeriertes Fenster. Klicken Sie auf die Pfeilschaltflächen, um die einzelnen Ausgabedateien zu sehen.

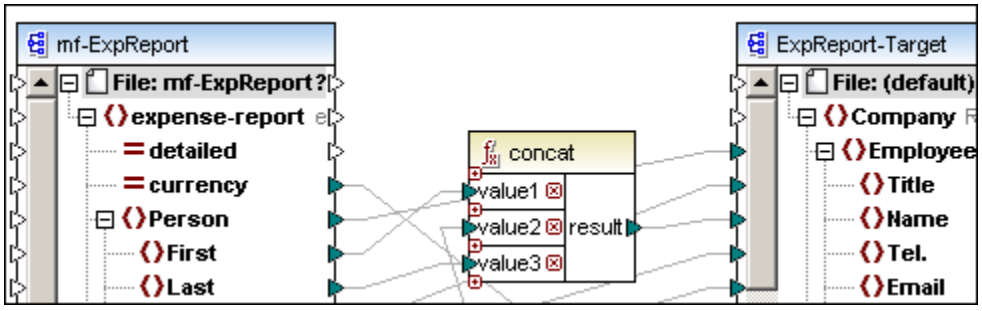
 © 2011 Altova GmbH



Klicken Sie auf die Schaltfläche "Alle generierten Ausgaben speichern" , um die hier gezeigten generierten Ausgabedateien zu speichern.

Mehrere Input-Dateien / eine Output-Datei - Zusammenführen von Input-Dateien

Mehrere Input-Dateien können in einer **einzigen** Output-Datei zusammengeführt werden, wenn der Konnektor **zwischen** den beiden **File:** Datenelementen entfernt wird, während in der Quellkomponente weiterhin z.B. über das Platzhalterzeichen "?" Zugriff auf mehrere Dateien besteht. Während also die Quellkomponente mehrere Dateien aufnehmen kann, kann dies die Output-Komponente nicht. Diese Dateien werden daher im Zieldokument aneinander angehängt.



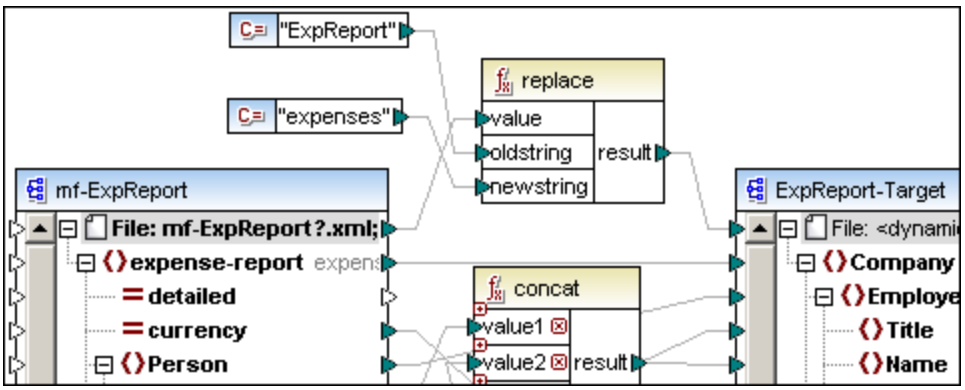
Mehrere Input-Dateien / mehrere Output-Dateien

Um mehrere Dateien n:n 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.

Der vollständige Pfadname der aktuell verarbeiteten Datei steht zur Verfügung, indem Sie das Output-Symbol des *File* Node z.B. mit "concat" verbinden, um eine neue Dateierweiterung hinzuzufügen.

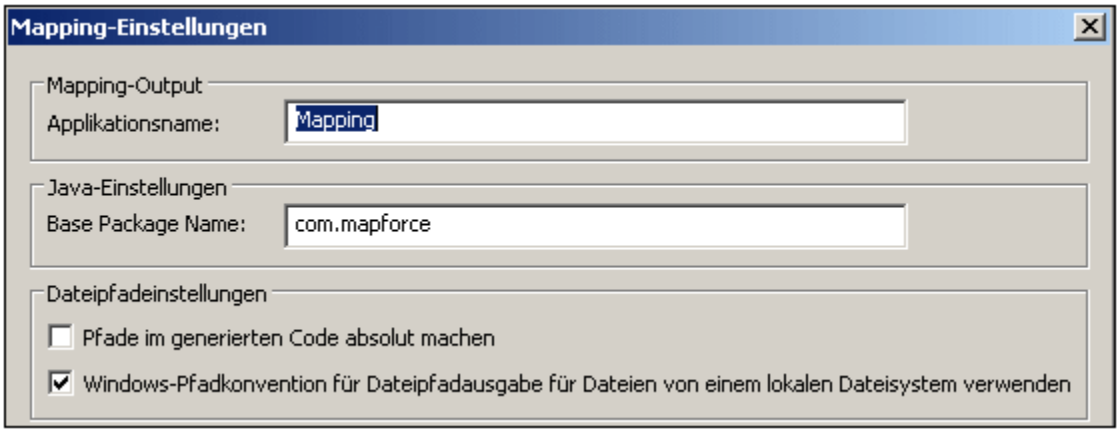
Bitte beachten Sie:

Vermeiden Sie es, einfach die File: 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 unten gezeigten replace-Funktion, ändern.



Im obigen Fall lauten die Output-Dateinamen **mf-expenses1.xml** und **mf-expenses2.xml**.

Mit Hilfe der Menüoption **Datei | Mapping-Einstellungen** können Sie die für das Mapping-Projekt verwendeten Dateipfadeinstellungen global definieren.



Über das Kontrollkästchen "Windows-Pfadkonvention...verwenden" können Sie sicherstellen, dass die Windows-Pfadkonventionen verwendet werden. Bei Ausgabe von XSLT2 (und XQuery) wird der Name der aktuell verarbeiteten Datei intern mit Hilfe der **document-uri** Funktion, die für lokale Dateien einen Pfad in der Form **file:// URI** zurückgibt, abgerufen.

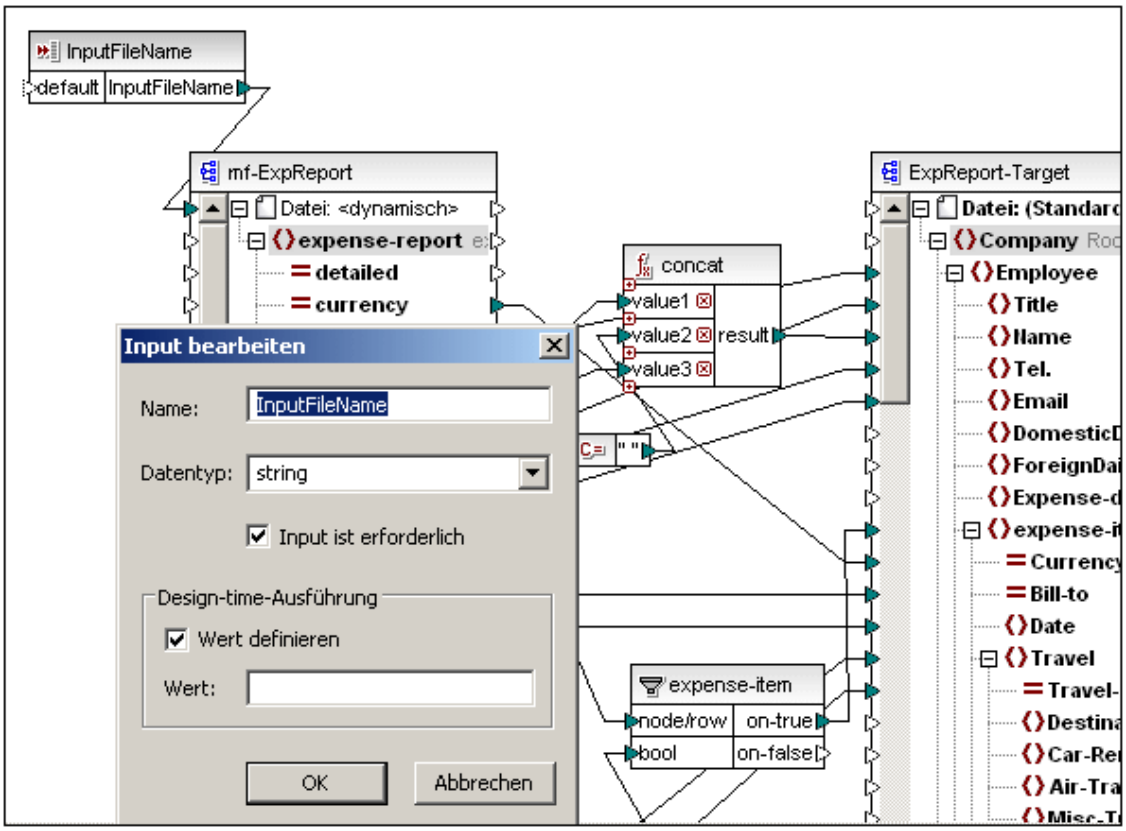
Wenn dieses Kontrollkästchen aktiv ist, wird eine file:// URI-Pfadspezifikation automatisch in einen vollständigen Windows-Dateipfad (z.B. "C:\...") konvertiert, um die Weiterverarbeitung zu vereinfachen.

 © 2011 Altova GmbH

9.2 Dynamische Dateinamen als Input-Parameter

Sie können in MapForce spezielle Input-Komponenten erstellen, die bei Ausführung des kompilierten Mappings über die Befehlszeile als **Parameter** verwendet werden können. Dieser spezifischen Input-Komponententyp kann nicht **innerhalb** einer benutzerdefinierten Funktion verwendet werden, sondern steht nur im Haupt-Mapping-Fenster zur Verfügung.

So verarbeiten Sie eine Datei mit Hilfe eines Input-Parameters während der Laufzeit:
Um den Pfad und den Dateinamen während der Laufzeit zu definieren, verbinden Sie eine Input-Parameter-Komponente mit dem Input-Symbol des *File* Node in der Quellkomponente. Je nachdem, welche Verbindungen zu anderen Datenelementen bestehen, wird dadurch der Input- und/oder Output-Dateiname definiert.



Der Eintrag mf-ExpReport.xml im Feld "'Wert" wird nur zu Vorschauzwecken im Ausgabefenster verwendet. Er hat keine Auswirkung auf die Parameterwerte, die bei Ausführung des Codes über die Befehlszeile verwendet werden. Nähere Informationen dazu finden Sie unter Befehlszeile - Definieren von Input/Output-Dateien.

Nach Generierung und Kompilierung des Codes können Sie den Dateinamen für das Mapping über die Befehlszeile bereitstellen:
mapping.exe /Input-Dateiname Dateiname.xml.

wobei:

- **/Input-Dateiname** der Name des ersten Parameters ist
- **Dateiname.xml** der zweite Parameter ist, d.h. der dynamische Dateiname, der verwendet werden soll, wenn die Applikation über die Befehlszeile ausgeführt wird.

© 2011 Altova GmbH

9.3 Mehrere XML-Dateien anhand einer einzigen XML-Quelldatei

Unten sehen Sie den Inhalt der XML-Quelldatei **mf-ExpReport.xml** aus dem Ordner ...\MapForceTutorial. Sie besteht aus der Spesenabrechnung für Fred Landis und enthält fünf Speseneinträge (expense-item) unterschiedlichen Typs. Dieses Beispiel steht unter dem Namen **Tut-ExpReport-dyn.mfd** im Ordner ...\Tutorial zur Verfügung.

Ziel:
Die Generierung einer separaten XML-Datei für jeden einzelnen unten aufgelisteten Spesenposten.

Person

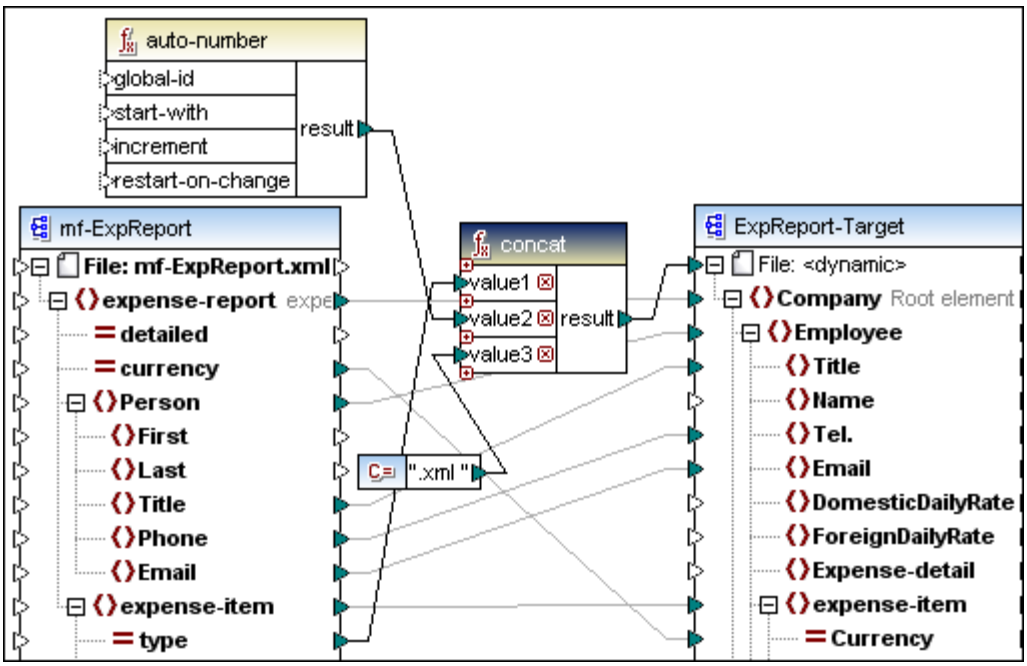
First	Fred
Last	Landis
Title	Project Manager
Phone	123-456-78
Email	f.landis@nanonull.com

expense-item (5)

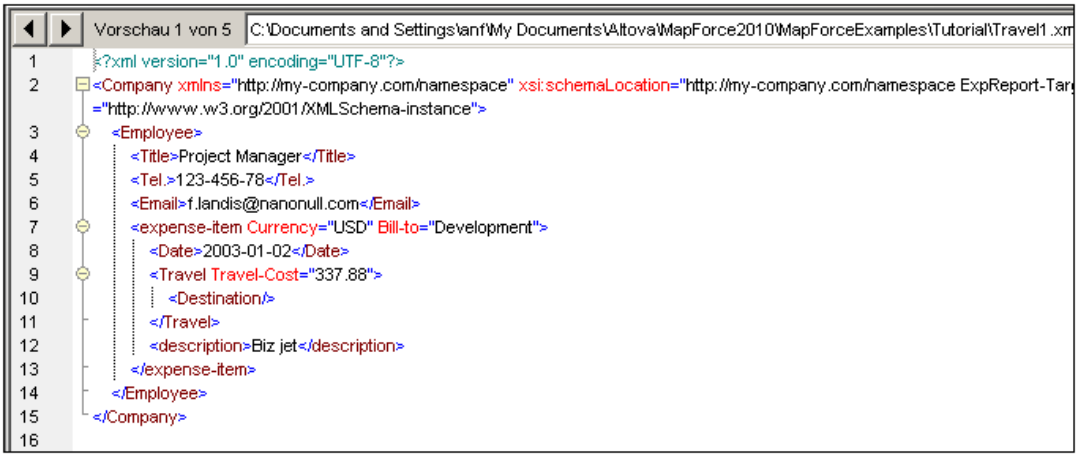
	type	expto	Date	Travel	Lodging
1	Travel	Development	2003-01-02	Travel Trav-cost=337.88	
2	Lodging	Sales	2003-01-01		Lodging
3	Travel	Accounting	2003-07-07	Travel Trav-cost=1014.22	
4	Travel	Marketing	2003-02-02	Travel Trav-cost=2000	
5	Meal	Sales	2003-03-03		

Da das Attribut "type" den jeweiligen Spesentyp definiert, ist dies das Datenelement, das wir zum Aufteilen der Quelldatei verwenden werden.

1. Fügen Sie eine concat-Funktion aus dem Bibliotheksfenster und über die Symbolleiste eine Konstantenkomponente ein.
2. Geben Sie als String-Wert der Konstantenkomponente *.xml ein, wenn das Dialogfeld angezeigt wird.
3. Fügen Sie die Funktion "auot-number" aus der Bibliothek **core | generator functions** des Bibliotheksfensters ein.

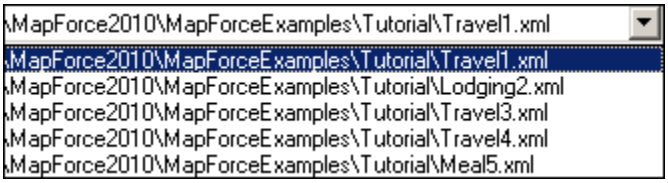


4. Erstellen Sie die Verbindungen wie oben gezeigt: type zu value1, auto-number zu value2 und die Konstante zu value3.
5. Verbinden Sie den **result**-Parameter/Output der concat-Funktion mit dem Datenelement **File:** der Zielkomponente. Beachten Sie, dass nun File: <dynamic> angezeigt wird.
6. Definieren Sie die restlichen benötigten Verbindungen.
7. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis des Mappings zu sehen.



Jeder Datensatz wird nun auf einem eigenen Vorschauregister angezeigt. Oben sehen Sie den ersten davon.

8. Klicken Sie auf den Pfeil der Dropdown-Liste, um alle generierten Dateien zu sehen.



 © 2011 Altova GmbH

Wenn Sie auf die Pfeile "Vorheriges" und "Nächstes" klicken, sehen Sie die einzelnen Dateien auf dem Ausgaberegister.

Anmerkung:

- Das Attribut **type** liefert den ersten Teil des Dateinamens, z.B. Travel.
- Die Funktion **auto-number** liefert die fortlaufend nummerierten Dateinummern (die Standardeinstellung ist Beginn bei=1 und erhöhen um=1) - daher Travel1.
- Die **Konstantenkomponente** liefert die Dateierweiterung, d.h. .xml, also lautet der Dateiname der ersten Datei Travel1.xml.
- Durch Klicken auf die Schaltfläche "Alle speichern" können Sie die einzelnen Dateien direkt über das Ausgaberegister speichern, ohne Code generieren zu müssen.

9.4 Relative und absolute Dateipfade

Ein relativer Pfad ist ein Pfad, der nicht mit einem Laufwerksbuchstaben beginnt, d.h. es kann sich um einen Dateinamen ohne einen Pfad handeln. Der spezifische Kontext, in dem der relative Dateiname verwendet wird, definiert den Basispfad. Die Behandlungsweise von relativen Dateinamen hat sich aufgrund der Unterstützung für das Mappen von Dateinamen als Daten in einem Mapping in MapForce Version 2010 geändert.

In früheren Versionen von MapForce (vor 2010) wurden Dateipfade für Dateien im selben oder einem untergeordneten Ordner relativ zur *.MFD-Datei gespeichert und beim Öffnen/Laden in absolute Pfade geändert.

Seit MapForce 2010 werden alle Referenzen auf externe Dateien wie z.B. Schemas oder XML-Instanzdateien so gespeichert, wie sie ins Dialogfeld eingegeben wurden. Auf diese Art können relative Pfade verwendet werden, wo dies erforderlich ist.

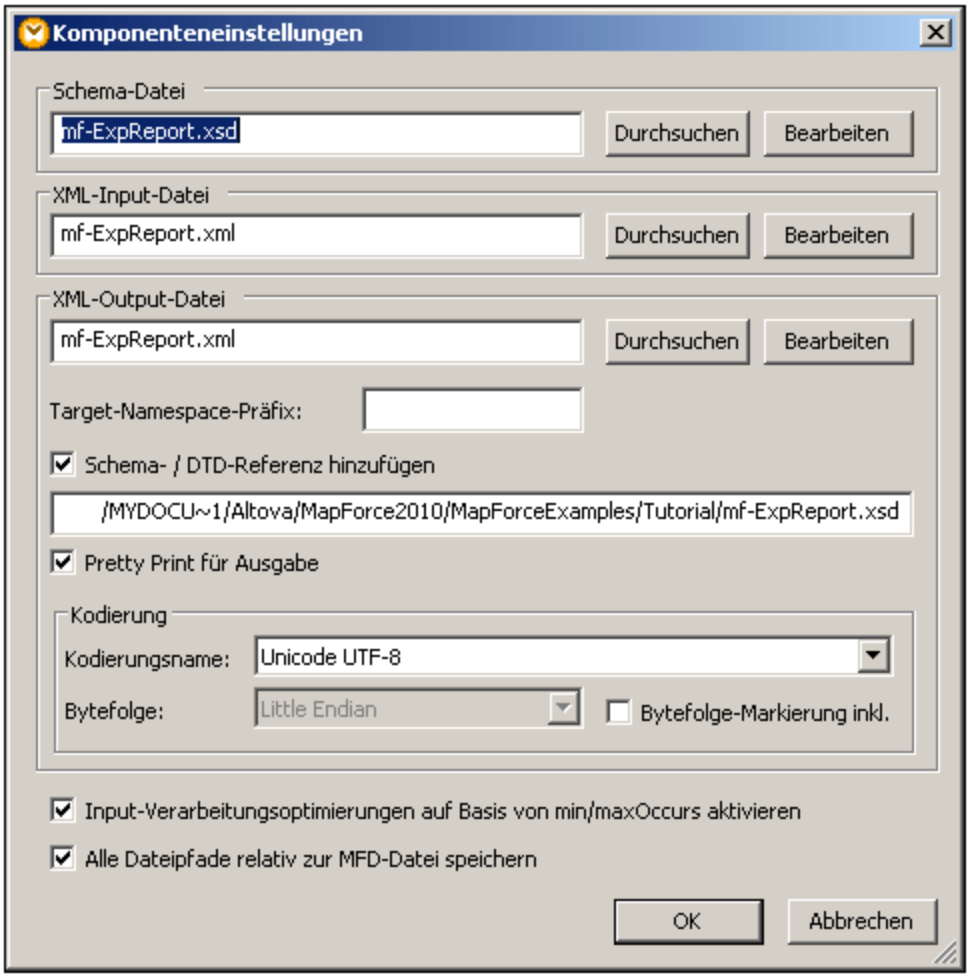
Alle Pfade relativ zur MFD-Datei speichern
Diese neue Option, die in allen Komponenteneinstellungsdialogfeldern zur Verfügung steht, speichert alle Dateipfade (der Komponente) relativ zum Ordner der aktuellen MFD-Datei. Auf diese Art können Sie ein Mapping zusammen mit allen damit in Zusammenhang stehenden Dateien in einen anderen Ordner verschieben, wobei alle Dateireferenzen weiterhin gültig bleiben.

Das bedeutet dass:

- absolute Dateipfade in relative Pfade geändert werden
- auch das übergeordnete Verzeichnis "**..**" eingefügt/geschrieben wird
- die Dateipfade bei Verwendung von "Speichern unter..." (relativ zur MFD-Datei) an den neuen Ordner, in dem die MFD-Datei gespeichert wird, angepasst werden.

Bitte beachten Sie:
Pfade, die ein nicht lokales Laufwerk referenzieren oder bei denen eine URL verwendet wird, werden nicht zu relativen Pfaden gemacht.

© 2011 Altova GmbH



Es gibt zwei separate Arten von Dateien, die von einer MFD-Datei aus referenziert werden:

- Dateien vom Typ Schema (XML-Schemas, WSDL-, FlexText-Konfigurationsdateien, …), die in das Feld **Schema-Datei** eingegeben werden.
- Instanzdateien, die in das Feld **xxx-Input-Datei** oder **xxx-Output-Datei** eingegeben werden.

Dateien vom Typ Schema

Dateien vom Typ Schema werden beim Erstellen eines Mappings verwendet. Sie definieren die **Struktur** der gemappten Input- und Output-Instanzdateien. Anhand dieser Informationen wird die Baumstruktur/Hierarchie der Datenelemente der verschiedenen Komponenten angezeigt. MapForce unterstützt das Eingeben und Speichern eines relativen Pfads zu Dateien vom Typ Schema.

- Relative Pfade zu Dateien **vom Typ Schema** werden immer **relativ zur MFD-Datei** aufgelöst.
- Wenn Sie ein Schema über das Dialogfeld "Öffnen", z.B. nachdem eine neue Komponente eingefügt wurde, oder durch Klicken auf die Schaltfläche "Durchsuchen" im Dialogfeld "Komponenteneinstellungen" auswählen, wird immer der **absolute** Pfad in das Feld eingefügt.
- Um einen relativen Pfad zu definieren, der ebenfalls in der MFD-Datei gespeichert wird,

löschen Sie den Pfad aus dem Textfeld oder geben Sie einen relativen Pfad oder Dateinamen ein. Dies geschieht automatisch beim Speichern der MFD-Datei, wenn das Kontrollkästchen "Alle Pfade relativ zur MFD-Datei speichern" aktiviert ist. Sie können den übergeordneten Ordner der MFD-Datei auch mit "**..**" definieren.

- Wenn Sie eine MFD-Datei, die Dateien aus demselben Verzeichnis referenziert, speichern und das ganze Verzeichnis anschließend in einen anderen Ordner verschieben, werden **absolute** in der mfd-Datei gespeicherte Pfade **nicht aktualisiert.** Daher sollten Benutzer, die Versionskontrollsysteme und unterschiedliche Arbeitsverzeichnisse verwenden, in diesem Fall relative Pfade verwenden.

Instanzdateien und die Ausführungsumgebung
Die Verarbeitung von Instanzdateien erfolgt in der generierten XSLT oder XQuery-Datei oder in der generierten Applikation sowie in der MapForce-Vorschau.

In den meisten Fällen ist es nicht sinnvoll, relative Pfade zu Instanzdateien als relativ zur MFD-Datei zu interpretieren, da dieser Pfad zum Zeitpunkt der Ausführung möglicherweise gar nicht existiert - der generierte Code könnte ja auf einem anderen Rechner ausgeführt werden. **Relative** Dateinamen für Instanzdateien werden daher relativ zur **Ausführungsumgebung** aufgelöst.

Zielsprache	Basispfad für relativen Instanzdateinamen
XSLT/XSLT2	Pfad der XSLT-Datei

Zum Dialogfeld **Datei | Mapping-Einstellungen** wurde ein neues Kontrollkästchen hinzugefügt, "**Pfade in generiertem Code absolut machen**", das die Kompatibilität von generiertem Code mit Mapping-Dateien (*.mfd) aus Versionen vor Version 2010 gewährleistet.

Der Status des Kontrollkästchens wird automatisch eingestellt und hängt davon ab, welche Dateien geöffnet sind. Das Kontrollkästchen ist:

- **deaktiviert,** wenn eine **neue** Mapping-Datei, d.h. Version 2010 erstellt oder geöffnet wird
  Relative Pfade für Input- und Output-Instanzdateien werden im aktuellen Zustand in den generierten Code geschrieben.

  Dies ermöglicht die Verwendung des generierten Codes in einem anderen Verzeichnis oder sogar auf einem anderen Rechner. Sie müssen sicherstellen, dass die mit relativen Pfaden angegebenen Dateien in der Laufzeitumgebung im richtigen Ordner zur Verfügung stehen.
- **aktiviert,** wenn eine ältere Mapping-Datei aus Version 2009, 2008 usw. geöffnet ist
  Relative Pfade für Input- und Output-Instanzdateien werden vor der Generierung von Code absolut gemacht (relativ zur *.MFD-Datei). Dies hat denselben Effekt, wie die Generierung von Code mit einer älteren Version von MapForce.

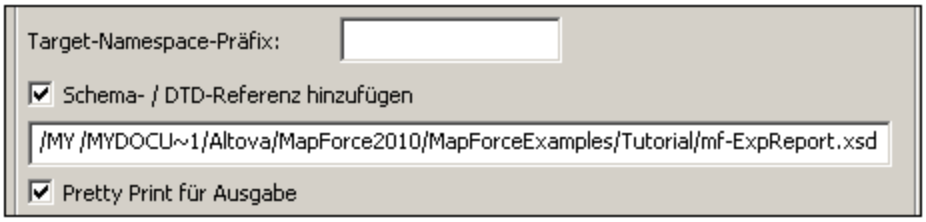
Beachten Sie, dass der **Quellinstanzdateiname** auch für folgende Zwecke verwendet wird:
- Ermittlung des XML-Root-Elements und des referenzierten Schemas
- Validierung anhand des ausgewählten Schemas
- Lesen von Excel Arbeitsblattnamen und - spalten
- zum Lesen von Spaltennamen und der Vorschau auf den Inhalt von Textdateien (CSV oder FLF)

Neues "schemaLocation" Feld für **XML-Zieldateien**

 © 2011 Altova GmbH

Da Schemareferenzen relativ zur MFD-Datei gespeichert werden können und sich die anhand einer Zielkomponente **generierte** XML-Datei oft in einem anderen Verzeichnis befindet, gibt es eine Methode, einen separaten **schemaLocation**-Pfad für die **XML-Zielinstanz** einzugeben, damit die XML-Datei dort validiert werden kann.

Dies ist das Feld unterhalb des Kontrollkästchens “Schema/DTD-Referenz hinzufügen” im Dialogfeld "Komponenteneinstellungen" (Doppelklicken Sie auf eine Komponente, um es zu öffnen). Ein ähnliches Feld gibt es für die Taxonomiereferenz in XBRL-Komponenten.



Der Pfad des referenzierten/verknüpften Schemas, der in dieses Feld eingegeben wird, wird in den generierten **Zielinstanzdateien** ins Attribut **xsi:schemaLocation** oder in die DOCTYPE-Deklaration geschrieben, wenn eine DTD verwendet wird.

Bei einer XBRL-Zieldatei wird die Taxonomiereferenz in das href-Attribut des Elements link:schemaRef geschrieben.

Anmerkung: Auch eine URL wie z.B. **http://mylocation.com/mf-expreport.xsd** kann hier eingegeben werden.

Kapitel 10

Zwischenvariablen

10 Zwischenvariablen

Bei Zwischenvariablen handelt es sich um eine spezielle Art von Komponente, die zur Lösung verschiedener komplexer Mapping-Probleme verwendet wird. Zwischenvariablen dienen zum Speichern von Mapping-Zwischenergebnissen für die weitere Verarbeitung.

- Variablen funktionieren in allen Sprachen mit Ausnahme von XSLT1.0.
- Variablenergebnisse sind immer Sequenzen, also eine Liste getrennter Werte, und können auch zum Erstellen von Sequenzen verwendet werden.
- Variablen sind Strukturkomponenten mit einem Root Node und haben keine Instanzen (XML-Dateien usw.) zugewiesen.
- Mit Hilfe von Variablen können Datenelemente einer Sequenz mit anderen Datenelementen derselben Sequenz verglichen werden.
- Mit Hilfe von Variablen können Zwischensequenzen erstellt werden. Datensätze können gefiltert werden, bevor sie an eine Zielkomponente übergeben werden oder sie können nach der Variablen z.B. mit Hilfe der position-Funktion gefiltert werden.

Unterschied zwischen Variablen und verketteten Mappings

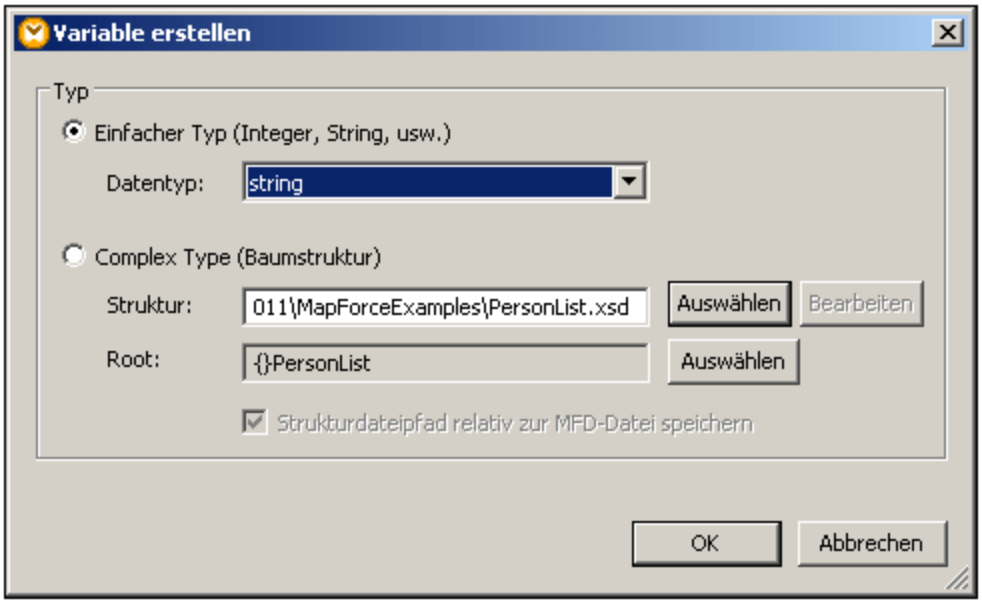
Verkettete Mappings	Variablen
Erfolgen in zwei voneinander unabhängigen Schritten.	Werden je nach Kontext / Geltungsbereich ausgewertet. Werden durch eine "compute-when"-Verbindung gesteuert
Die Zwischenergebnisse werden bei der Ausführung des Mappings extern in XML-Dateien gespeichert.	Die Zwischenergebnisse werden bei der Ausführung des Mappings intern gespeichert und nicht in physischen Dateien.
Eine Vorschau des Zwischenergebnisses kann mit Hilfe der Vorschau-Schaltfläche angezeigt werden.	Das Ergebnis der Variablen kann nicht in einer Vorschau angezeigt werden.

So fügen Sie eine Zwischenvariablenkomponente ein:
Es gibt verschiedene Methoden um Zwischenvariablen einzufügen: Über die Menüoption, durch Klicken auf die Schaltfläche "Var." oder durch Rechtsklick auf die Input/Output-Symbole und Erstellen von Variablen auf Basis der Input-/Output-Komponente.

1. Wählen Sie die Menüoption **Einfügen | Variable...** oder klicken Sie in der Symbolleiste auf die Schaltfläche "Variable"

. Sie können jetzt wählen, ob Sie eine Variable vom Typ "simple" oder "complex" einfügen möchten.

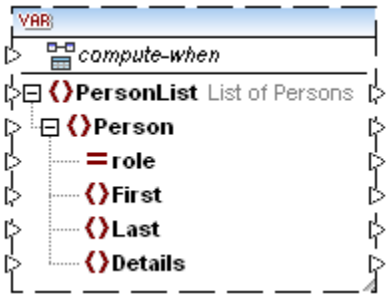
 © 2011 Altova GmbH



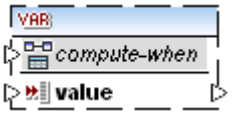
2. Aktivieren Sie das Optionsfeld für die Art der einzufügenden Variablen z.B. "Simple type" oder "Complex type".

   Wenn Sie das Optionsfeld "Complex type" aktiviert haben:
3. Klicken Sie auf die Schaltfläche "Auswählen" zum z.B. XML-Schema auszuwählen und wählen Sie im nächsten Dialogfeld das Root-Element aus.
4. Klicken Sie auf OK, um die Variable einzufügen.

Variable vom Typ "complex":



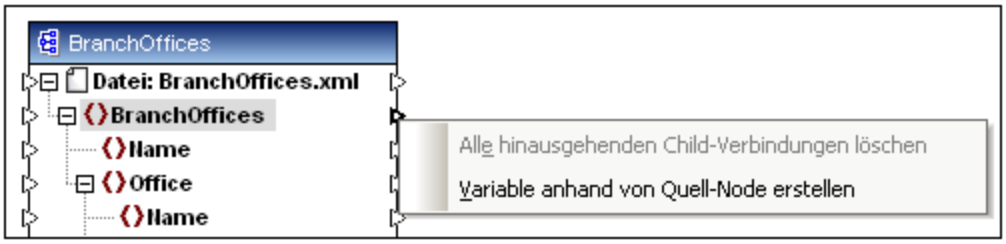
Variable vom Typ "simple":



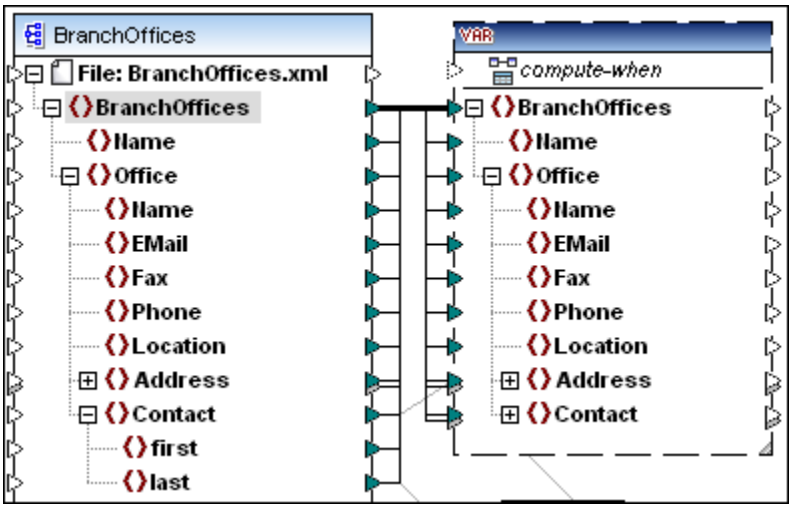
Eine solche Variable hat ein einziges mapbares Datenelement / einen einzigen Wert, z. B. string, integer. Beachten Sie, dass der "Wert" dupliziert werden kann.

Alternative Methoden zum Einfügen von Variablen:

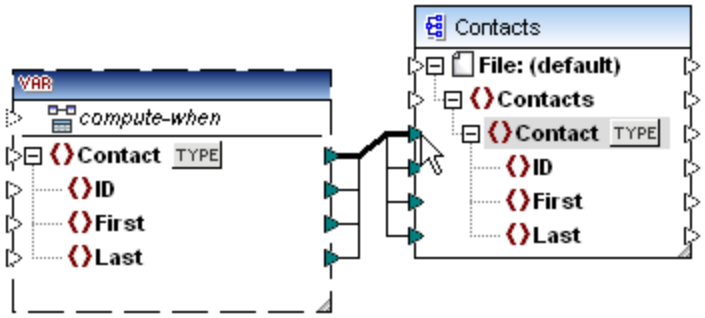
- Klicken Sie mit der rechten Maustaste auf ein **Output**-Symbol einer Komponente (z.B. BranchOffices) und wählen Sie "Variable anhand von Quell-Node erstellen".



Daraufhin wird anhand desselben Quellschemas eine Variable vom Typ "complex" erstellt und alle Datenelemente werden automatisch mit einer "Alles kopieren"-Verbindung verbunden.



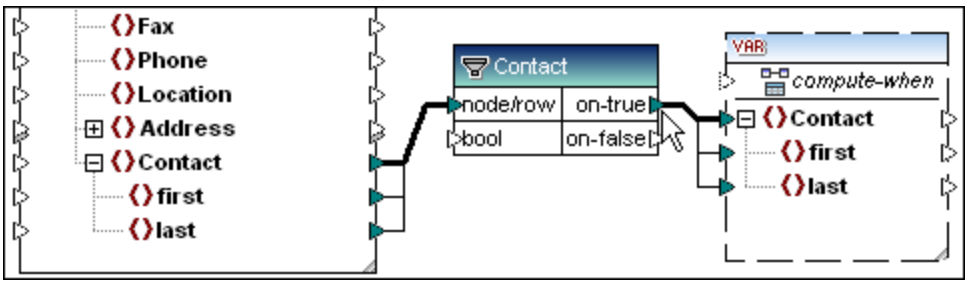
- Klicken Sie mit der rechten Maustaste auf ein **Input** -Symbol einer Zielkomponente (z. B. Contact) und wählen Sie den Befehl "Variable für Ziel-Node erstellen".



Daraufhin wird anhand desselben Quellschemas, das auch in von Zielkomponente verwendet wird, eine Variable vom Typ "complex" mit dem Datenelement "Contact" als Root-Node erstellt und alle Datenelemente werden automatisch mit einer "Alles kopieren"-Verbindung verbunden.

- Klicken Sie mit der rechten Maustaste auf das **Output**-Symbol einer Filter-Komponente (on-true/on-false) und wählen Sie den Befehl "Variable anhand von Quell-Node erstellen".

 © 2011 Altova GmbH



Daraufhin wird anhand desselben Quellschemas eine Komponente vom Typ "complex" erstellt und das mit dem Filter-Input-Node/ der Filter-Input-Zeile verbundene Datenelement, also Contact, wird automatisch als Root-Element der Zwischenkomponente verwendt.

Compute-when

Über das Input-Datenelement "compute-when" können Sie den **Geltungsbereich** der Variablen festlegen; anders ausgedrückt: Sie können festlegen, wann und wie oft der Wert der Variablen bei der Ausführung des Mappings berechnet wird. Sie müssen diesen Input nicht in vielen Fällen verbinden, es kann aber notwendig sein, um den Standard-Kontext außer Kraft zu setzen oder die Leistung des Mappings zu optimieren.

Im folgenden Text wird als **Substruktur** eine Gruppe von Datenelemente/Nodes in einer Zielkomponente mit allen ihren Nachfahren bezeichnet: z.B. ein Element <Person> mit seinen Child-Elementen <FirstName> und <LastName>.

Als **Variablenwert** werden die Daten bezeichnet, die auf der **Output**-Seite der Variablenkomponente zur Verfügung stehen.

- Bei "simple" Variablen handelt es sich hierbei um eine Sequenz atomarer Werte, die den in den Komponenteneigenschaften definierten Datentyp haben.
- Bei "complex" Variablen handelt es sich um eine Sequenz von Root Nodes (vom in den Komponenteneigenschaften definierten Typ), von denen jeder alle seine Nachfahren-Nodes enthält.

  Die Sequenz atomarer Werte (oder Nodes) darf natürlich nur ein oder sogar null Datenelemente enthalten. Dies hängt davon ab, was mit der Input-Seite der Variablenkomponente und allen Parent-Datenelementen in der Quell- und Zielkomponente verbunden ist.

Compute-when - 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** (entweder über einen Konnektor von der Variablenkomponente direkt 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 (siehe "Schleifen, Gruppen und Hierarchien - der Kontext").

Dieses Standardverhalten ist dasselbe wie das von "complex" Outputs von regulären benutzerdefinierten Funktionen und Webservice-Funktionsaufrufen.

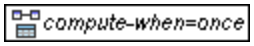
Wenn die Variable mit mehreren **nicht miteinander in Zusammenhang stehenden** Ziel-Nodes verbunden ist, wird der Wert der Variablen für jeden davon separat berechnet. Dabei können in jedem Fall unterschiedliche Ergebnisse erzeugt werden, da unterschiedliche Parent-Verbindungen sich auf den Kontext auswirken, in dem der Wert der Variablen ausgewertet wird.

Compute-when - verbunden
Wenn das compute-when-Datenelement mit einem **Output Node** einer Quellkomponente 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 dem 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". Bei compute-when bedeutet das "berechne für jedes Quelldatenelement den Variablenwert".

Compute-once 

Wenn Sie mit der rechten Maustaste auf das "compute-when"-Symbol klicken und im Kontextmenü den Befehl "Einmal berechnen" auswählen, ändert sich das Symbol in "compute-when=once" und das Input-Symbol wird entfernt.

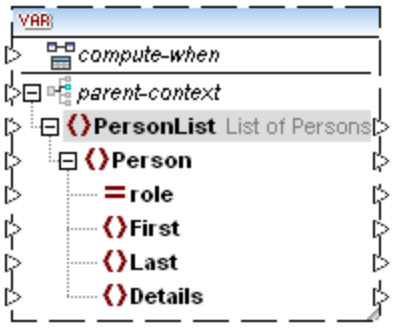
Bei dieser Einstellung wird der Variablenwert nur einmal berechnet, und zwar **vor** allen Werten der Zielkomponenten, sodass die Variable praktisch zu einer globalen Konstante für das restliche Mapping wird.

In einer benutzerdefinierten Funktion wird die Variable bei jedem Funktionsaufruf ausgewertet, bevor das tatsächliche Funktionsergebnis ausgewertet wird.

Parent-Kontext
Hauptsächlich wird ein Parent-Kontext hinzugefügt, wenn mehrere Filter verwendet werden, und Sie einen zusätzlichen Parent-Node benötigen, über den iteriert werden kann.

So fügen Sie einen Parent-Kontext zu einer Variablen hinzu:

- Klicken Sie mit der rechten Maustaste auf den Root Node, z.B. PersonList und wählen Sie im Kontextmenü den Befehl "Parent-Kontext hinzufügen".
  Daraufhin wird ein neuer Node hinzugefügt "parent-context" zur bestehenden Hierarchie hinzugefügt.

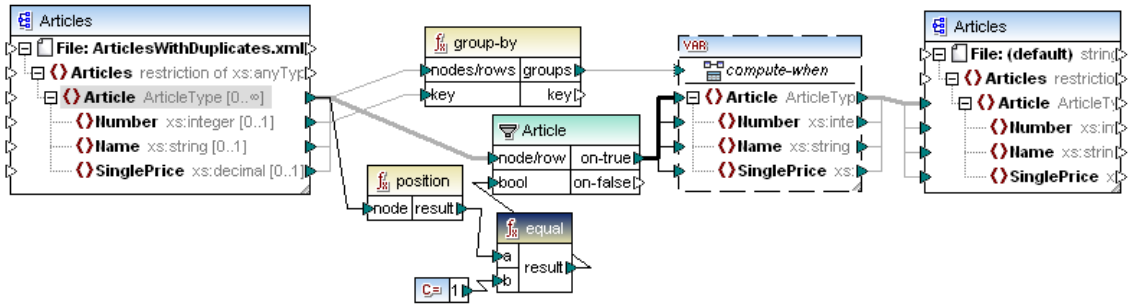


 © 2011 Altova GmbH

Durch den Parent-Kontext wird innerhalb der Komponente ein virtueller übergeordneter (Parent) Node zur Hierarchie hinzugefügt. Auf diese Art können Sie über einen zusätzlichen Node in der Quellkomponente iterieren.

10.1 Variablen - Anwendungsfälle

Herausfiltern mehrerer Instanzen desselben Datensatzes aus einer Quellinstanz
Quelldaten können oft mehrere Instanzen desselben Datensatzes enthalten. In den meisten Fällen soll nur einer dieser Datensätze auf die Zielkomponente gemappt werden. Das Beispiel unten finden Sie unter dem Namen **DistinctArticles.mfd** im Ordner ...\MapForceExamples.



Die Datei ArticlesWithDuplicates.xml enthält zwei Artikel mit derselben Artikelnummer (zwei mit der Artikelnummer 1 und zwei mit der Artikelnummer 3).

Articles

xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespac...	Articles.xsd

Article (6)

	Number	Name	SinglePrice
1	1	T-Shirt	25
2	1	T-Shirt Duplicate	25
3	2	Socks	2.30
4	3	Pants	34
5	4	Jacket	57.50
6	3	Pants Duplicate	34

Die Artikelnummer wird als Schlüssel in der group-by-Funktion verwendet, sodass damit eine Gruppe pro (eindeutiger) Artikelnummer erstellt wird. Daher enthält jede Gruppe einen Artikel und alle unerwünschten Duplikate dieses Artikels. Der nächste Schritt ist, das erste Datenelement jeder Gruppe zu extrahieren und den Rest zu löschen.

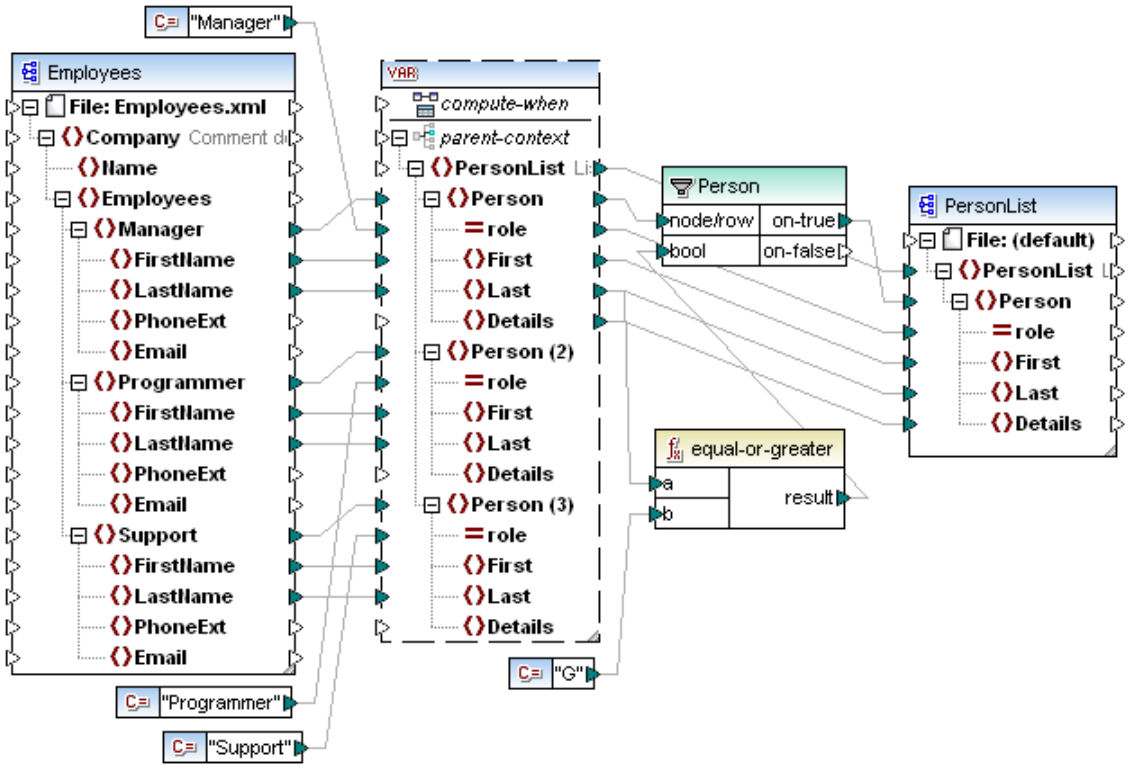
Durch die Verbindung der Gruppierungsausgabe mit compute-when wird die Variable einmal für jede Gruppe im Kontext dieser Gruppe ausgewertet. Auf diese Weise wird eine zusätzliche Kontextebene gebildet, so als hätten wir ein übergeordnetes Element des Article-Zielelements verbunden.

Um den ersten Artikel jeder Gruppe auszuwählen, verwenden wir die position-Funktion und eine Filter-Komponente, die mit der Input-Seite der Variablen verbunden wird.

Anwenden von Filtern auf Zwischensequenzen:
Nodes in Variablenkomponenten können wie bei jeder anderen Komponentenart dupliziert werden. Auf diese Art können Sie Sequenzen aus mehreren unterschiedlichen Quellen bilden und diese Sequenz dann weiterverarbeiten.

So sehen Sie z.B. in der unten stehenden Abbildung, wie PersonList.mfd mit Hilfe einer Zwischenvariablen geändert werden könnte und wie auch Konstantenkomponenten als Quelldatenelemente fungieren können.,

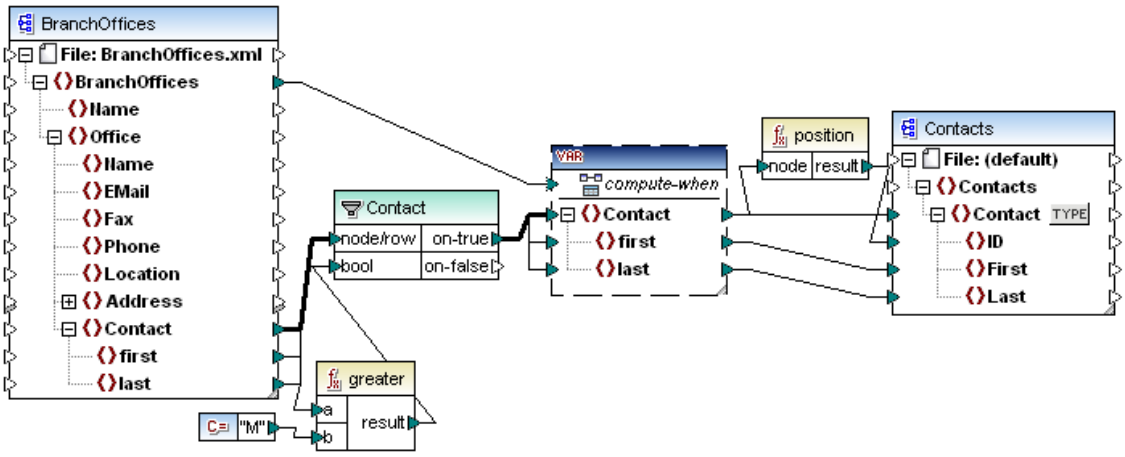
 © 2011 Altova GmbH



Der Node "Person" der Variable wurde zwei Mal kopiert und es wurde ein Filter hinzugefügt, um Personen, deren Nachname (LastName) mit einem Buchstaben nach "G" beginnt, zu filtern.

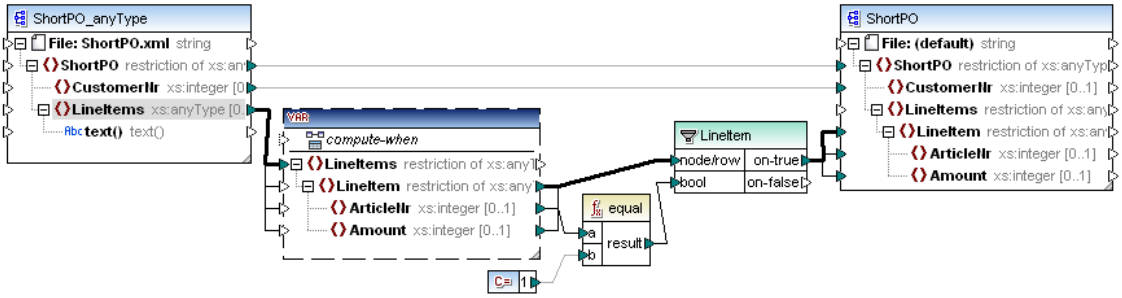
Nummerieren von Nodes einer gefilterten Sequenz
Das Beispiel unten finden Sie unter dem Namen **PositionInFilteredSequence.mfd** im Ordner ...\MapForceExamples. In diesem Beispiel werden mit Hilfe der Variablen die gefilterten Kontakte gesammelt, deren Nachname mit einem Buchstaben nach M beginnt.

Die Kontakte werden anschließend an die Zielkomponente übergeben, wobei diese Kontakte mit Hilfe der position-Funktion der Reihe nach nummeriert werden.



Extrahieren bestimmter Daten aus dem anyType-Node einer Quellkomponente
Dieses Beispiel besteht aus einer Quellkomponente, die anyType-Elemente enthält, aus denen

wir bestimmte Daten filtern möchten.



Die Zwischenvariable basiert auf einem Schema, das Nodes desjenigen Datentyps hat, den wir mappen möchten, d.h. ArticleNr und Amount sind beide vom Typ Integer. Diese spezifischen Daten werden von ArticleNr. gefiltert und an die Zielkomponente übergeben.

 © 2011 Altova GmbH

Kapitel 11

MapForce Tipps...

11 MapForce Tipps...

In diesem Abschnitt werden häufige Aufgabenstellungen beschrieben, wie sie beim Erstellen Ihrer eigenen Mapping-Projekte oft auftreten werden.

Beschrieben werden die folgenden Aufgaben:

Allgemeines:

Aufgabe:	**Informationen in diesem Abschnitt:**
Filtern von Daten anhand bestimmter Kriterien	Filtern - Abrufen dynamischer Daten
Verwenden von **dynamischen/mehreren Input- und Output-Dateien** beim Mapping	Dynamische Input/Output-Dateien pro Komponente
Mappen/Verwenden eines **derived complex type** (xsi:type)	Derived XML Schematypen - Mappen
Erstellen eines **rekursiven** benutzerdefinierten Mappings	Rekursives benutzerdefiniertes Mapping
Verwendung von min, max, sum, avg und count-Aggregatsfunktionen	Aggregatfunktionen: min, max, sum
Verwenden einer **Input-Komponente** als **Parameter** bei einer Ausführung über die Befehlszeile	Befehlszeile - Definieren von Input/Output-Dateien
Definieren eines **Alternativwerts** für eine Input-Komponente	Befehlszeile - Definieren von Input/Output-Dateien
Transformieren eines Input-Werts mittels **Lookup-Tabelle** in einen Output-Wert	Wertezuordnung - Transformieren von Input-Daten
Ausführen von MapForce über die **Befehlszeile**	Befehlszeileparameter
Informationen über die Typkonvertierung beim Mappen	Überprüfung der Typkonvertierung
Erstellen Ihrer eigenen **Katalogdateien**	Katalogdateien in MapForce
Zusammenführen mehrerer Quelldateien in einer einzigen Zieldatei	Zusammenführe mehrerer Dateien in einer Zieldatei

Nodes

Aufgabe	**Informationen in diesem Abschnitt:**
Testen von Nodes; vorhandene / nicht vorhandene Nodes	Node-Tests, exists / not-exists
Gruppieren von Nodes nach Ihrem Inhalt	Gruppieren von Nodes / Node-Inhalt
Mappen von Daten auf Basis der Position eines Node in einer Sequenz	Position von Kontextelementen in einer Sequenz

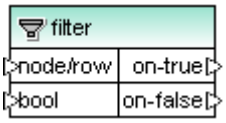
© 2011 Altova GmbH

Kommentieren eines Mappings / bestimmter Konnektoren oder Nodes	Annotationen / Kommentare
Definieren des Node, der als Kontextnode in einer Quelldatei verwendet werden soll.	Prioritätskontext Node/Datenelement

11.1 Filtern - Abrufen dynamischer Daten, Lookup-Tabelle

Wenn Sie Daten anhand bestimmter Kriterien abrufen/filtern möchten, verwenden Sie die **Filter** -Komponente.

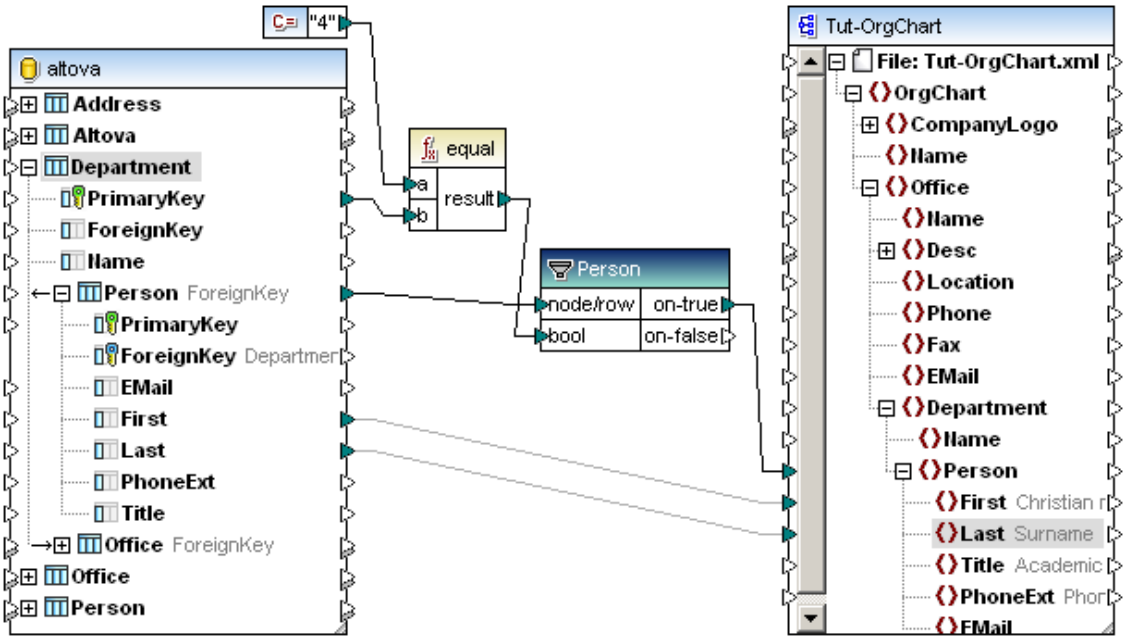
- Klicken Sie in der Symbolleiste auf das Filter-Symbol , um die Komponente einzufügen.
  Daraufhin wird die Filterkomponente ohne Verbindungslinien eingefügt. Dabei wird in der Titelleiste der Komponente der Titel "filter" angezeigt.



  Sobald eine Verbindung zur Quellkomponente hergestellt wurde, wird in der Titelleiste der Name des Node/Zeileneintrags, der mit der Filter-Komponente verbunden wurde, in diesem Fall "Person" angezeigt.

Ziel: Mappen aller Vor- und Nachnamen (First, Last) aus der Tabelle **Person,** wo der **Primärschlüssel** (PrimaryKey) 4 ist.

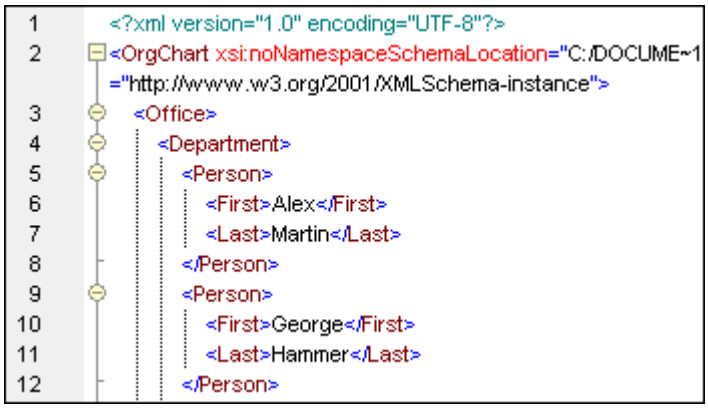
- Der Wert des Primärschlüssels wird mit dem Wert 4, der mit Hilfe der equal-Funktion von der Konstantenkomponente bereitgestellt wird, verglichen.
- Es wird der Eintrag **PrimaryKey** aus der "Root-Tabelle" "Department" gemappt.
- Aus der Tabelle "Person", die ein Child der Root-Tabelle "Department" ist, werden die Einträge **First** und **Last** gemappt.
- Die unter **derselben** "Root-Tabelle" (in diesem Fall "Department") definierten Mappings behalten die Beziehungen zwischen ihren Tabellen bei.



Ein Filter hat zwei Input- und Output-Parameter: **node/row** und **bool** und **on-true**, **on-false**. Wenn die Boolsche Bedingung **true** ergibt, wird der Inhalt der mit dem **node/row**-Parameter verbundenen **Child-**Nodes der Quellkomponente an die Zielkomponente übergeben. Wenn der Boolsche Parameter false ergibt, können die Komplementwerte verwendet werden, indem eine Verbindung zum on-false-Parameter hergestellt wird.

 © 2011 Altova GmbH

Auf dem Register "Ausgabe" sehen Sie die Vor- und Nachnamen der Personen in der IT-Abteilung mit dem Primärschlüssel 4.



Weitere Beispiele zum Filtern von Daten finden Sie unter:

- XML-Daten filtern: Filtern von Daten:
- Filtern von CSV-Dateien nach Kopfzeile und Details: Hierarchien für CSV-Dateien und Dateien mit fester Länge
- Tabellenbeziehungen und Filter: Datenbankbeziehungen und wie man sie beibehält oder ignoriert
- Definieren von SQL-Where-Filtern: SQL WHERE-Komponente / -Beziehung

- Tipps zu Filterkomponenten: Filterkomponenten - Tipps

11.1.1 Filterkomponenten - Tipps

Vermeiden Sie das Verketten von Filterkomponenten
Jede Filterkomponente führt dazu, dass die Quelldaten in einer Schleife verarbeitet werden, sodass **n** mal auf die Quelle zugegriffen wird. Wenn Sie zwei Filter miteinander verketten, wird die Schleife **n*n** mal verarbeitet.

Lösung:
Verwenden Sie "**logical-and**" Komponenten, um die Booleschen Ausdrücke zweier Filterkomponenten miteinander zu kombinieren. Das Ergebnis ist eine einzige Filterkomponente, die die Schleife nur **n** mal abarbeitet.

Verbinden Sie den "on-true/on-false"-Parameter der Filterkomponente mit den Parent-Datenelementen des Ziels
Am besten funktionieren Filterkomponenten, wenn Sie, anstatt mit einzelnen Datenelementen, mit den entsprechenden Parent-Elementen verbunden werden.

Der Boolesche Filterausdruck wird daher gegen den Parent evaluiert, **bevor** die einzelnen Child-Elemente verarbeitet werden. Bei Verwendung von Filtern, die von einer Datenbanktabelle gemappt werden, wird folgende SQL-Anweisung generiert:

- "SELECT * FROM table WHERE <expression>" wenn das **Parent-**Datenelement gemappt wird oder
- "SELECT * FROM table" und anschließend für jede Zeile überprüfen, wenn die Child-Datenelemente gemappt werden.

Bitte beachten Sie:
Bei Verbindung eines Filters von einem Parent-Datenelement der Quelle muss der on-true/on-false-Parameter auch mit dem Parent-Datenelement des Ziels verbunden werden. Ist dies nicht möglich, wenden Sie diese Regel nicht an.

Verbinden Sie den "on-false"-Parameter, um das Komplement-Nodeset zu mappen
Indem Sie diesen Parameter verbinden, haben Sie schnellen Zugriff auf das durch das aktuelle Mapping definierte Komplement-Nodeset. Wenn Sie diesen Parameter verwenden, eine Verbindung mit Parent-Datenelementen herstellen usw., gelten dieselben Tipps.

Verwenden Sie keine Filter zum Mappen von Child-Daten, wenn das Parent-Element gemappt wird
Bei Verwendung eines Filters zum Mappen von Daten von einem Parent der Quelle auf ein Parent des Ziels wird **derselbe Filter** automatisch auch auf **jedes Child**-Element dieses Parent-Elements angewendet.

Wenn das Parent-Datenelement gemappt werden kann, ist es nicht notwendig, Filter zu verwenden, um gefilterte Daten für die Child-Elemente zu liefern! Sie können daher Child-Daten direkt mappen.

Verwenden Sie den Prioritätskontext, um festzulegen, in welcher Reihenfolge nicht miteinander in Beziehung stehende Elemente gemappt werden sollen
Mappings werden immer von oben nach unten durchgeführt; wenn Sie zwei Tabellen durchsuchen/abarbeiten, wird zuerst eine Tabelle, dann die andere verarbeitet. Wenn Sie nicht miteinander in Beziehung stehende Elemente mappen, ohne den Prioritätskontext festzulegen, weiß MapForce nicht, welche Schleife zuerst verarbeitet werden soll, und wählt daher automatisch die erste Tabelle bzw. Datenquelle aus.

Lösung:
Legen Sie fest, welche Tabelle bzw. welche Quelldaten zuerst abgearbeitet/durchsucht werden

 © 2011 Altova GmbH

sollen und legen Sie am Konnektor zu dieser Tabelle den Prioritätskontext fest. Ein konkreteres Beispiel finden Sie unter "Prioritätskontext".

So definieren Sie einen Prioritätskontext:

- Rechtsklicken Sie auf ein Input-Symbol und wählen Sie im Popup-Menü "Priorität - Kontext".
  Steht die Option nicht zur Verfügung, müssen Sie zuerst die übrigen Input-Symbole mappen, damit die Option aktiv wird.

Filter und quellorientiertes (Mixed Content) Mapping

Quellorientierte Mappings funktionieren nur bei direkten Verbindungen zwischen Quell- und Zielkomponenten. Verbindungen unterhalb einer quellorientierten Verbindung werden nicht als quellorientiert betrachtet und die Datenelemente werden in der Reihenfolge Zielkomponentendatenelement/Node behandelt.

Ein einziger Filter, bei dem beide Output-Elemente mit demselben / separaten Zielelementen verbunden sind, verhält sich, als gäbe es zwei separate Filterkomponenten, wobei eine eine negierte Bedingung haben muss.

Wenn eine Ausnahmeereigniskomponente mit einer der Filterausgaben verbunden ist, so wird die Ausnahmebedingung überprüft, wenn die Mappings auf die andere Filterausgabe ausgeführt werden.

11.2 Wertezuordnung - Transformieren von Input-Daten

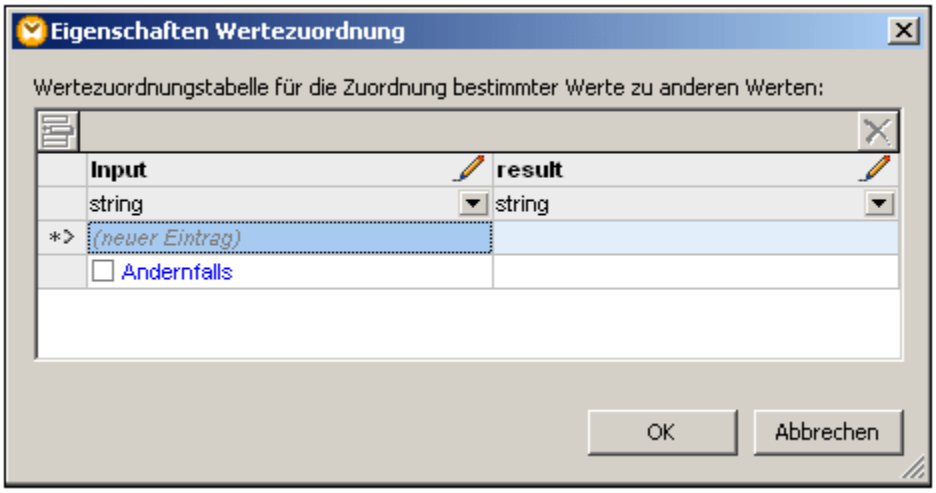
Mit Hilfe der Wertezuordnungskomponente können Sie einen Input-Wert mit Hilfe einer Lookup-Tabelle in einen anderen Output-Wert transformieren. Dies eignet sich zum Konvertieren unterschiedlicher Enumerationstypen. Die Komponente hat nur ein Input- und ein Output-Datenelement.

Anmerkung: Wenn Sie Daten anhand von bestimmten Kriterien abrufen/filtern möchten, verwenden Sie bitte die **Filter**-Komponente (siehe Filtern von XML-Daten: Filtern von Daten, .

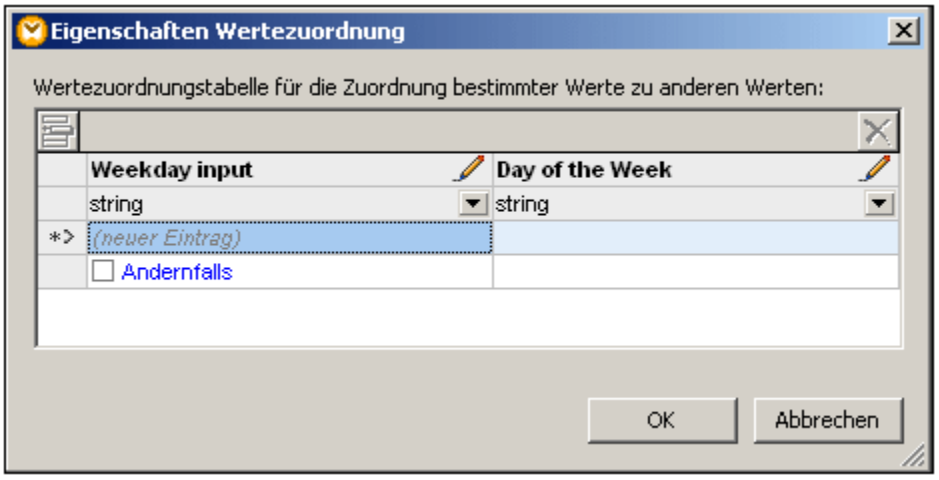
So verwenden Sie eine Wertezuordnungskomponente:

1. Wählen Sie die Menüoption **Einfügen | Wertezuordnung** oder klicken Sie in der Symbolleiste auf die Schaltfläche "Wertezuordnung" .

2. Doppelklicken Sie auf die Wertezuordnungskomponente, um die Wertezuordnungstabelle zu öffnen.

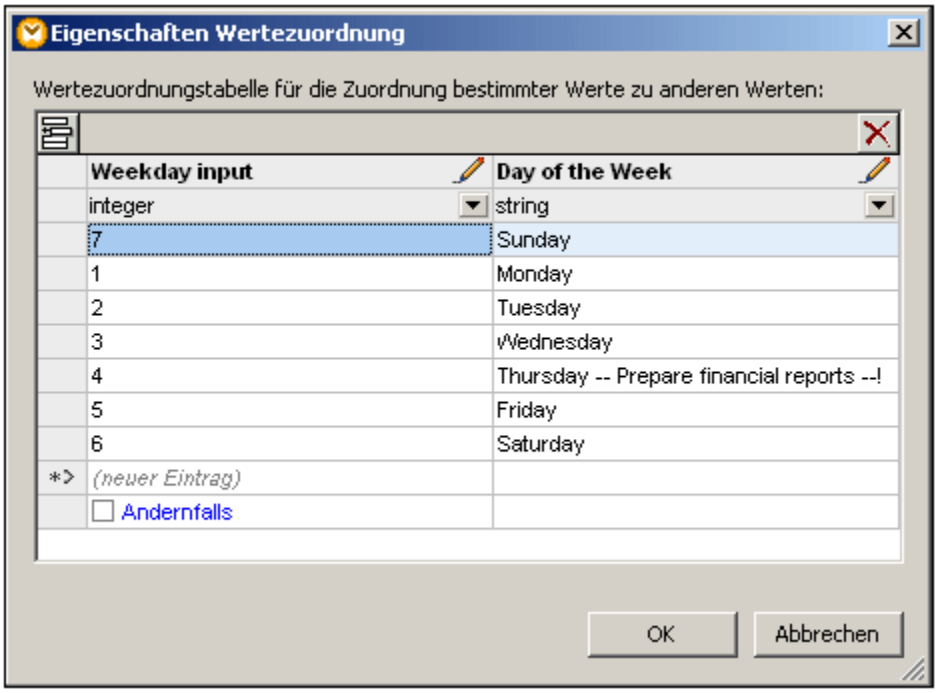


3. Klicken Sie in die Spaltenüberschriften und geben Sie in die erste Spalte **Weekday input** und in die zweite **Day of the Week** ein.



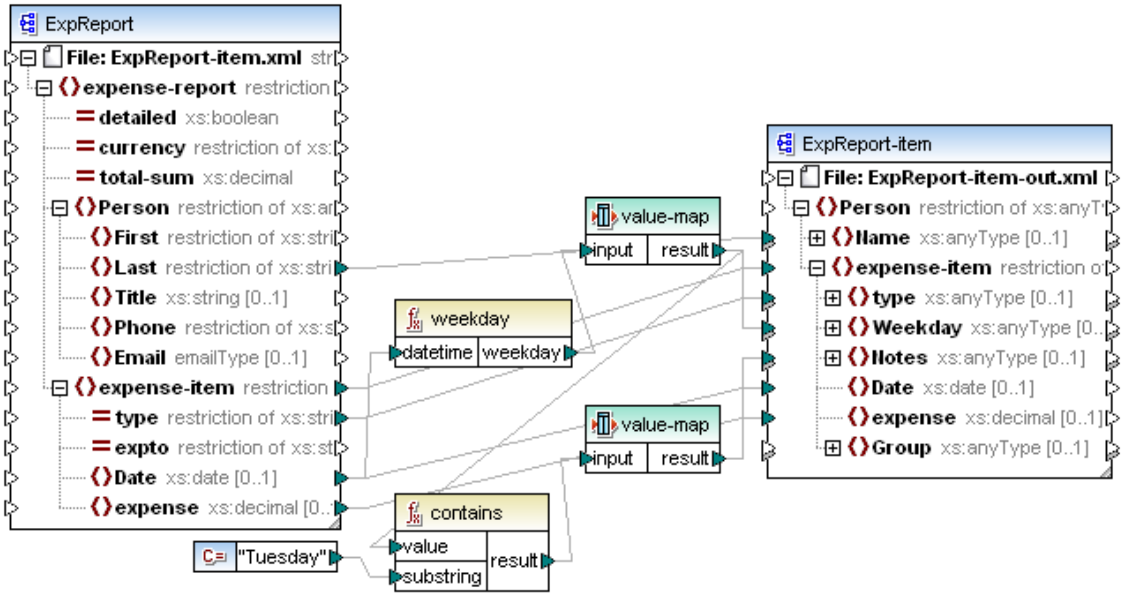
 © 2011 Altova GmbH

4. Geben Sie den Input-Wert, der transformiert werden soll, in die Spalte **Weekday input** ein.
5. Geben Sie den Ausgabewert, in den der Wert transformiert werden soll, in die Spalte **Day of the week** ein.
6. Geben Sie ein neues Wertepaar in das Input-Feld ***(neuer Eintrag)*** ein.
6. Klicken Sie auf die **Datentyp**-Auswahlliste unterhalb der Spaltenüberschrift, um den Input- und den Output-Datentyp auszuwählen, z.B:. Integer und String.



7. Anmerkung: Aktivieren Sie das Kontrollkästchen **Andernfalls** und geben Sie den Wert ein, um einen alternativen Ausgabewert zu definieren, wenn die bereitgestellten Werte in der Input-Komponente nicht vorhanden sind. Nähere Informationen dazu, wie Sie Quelldaten übergeben, ohne diese zu ändern finden Sie unter Übergeben von unveränderten Daten über eine Wertezuordnung.
8. Um die Spaltennamen, die auch im Mapping angezeigt werden, zu ändern, klicken Sie auf die Bearbeitungssymbole in den Kopfzeilen. Auf diese Art kann der Zweck der Komponente im Mapping besser verdeutlicht werden.

Die Datei **Expense-valmap.mfd** im Ordner **...\MapForceExamples\Tutorial** ist ein Beispiel-Mapping, in dem Sie sehen, wie die Wertezuordnung verwendet werden kann.



Funktionsweise dieses Mappings:
Der Wochentag wird aus dem Datenelement "Date" in der Datenquelle extrahiert, der numerische Wert wird in Text konvertiert und dieser Text (also Sunday, Monday, usw.) wird in das Datenelement "Weekday" der Zielkomponente eingesetzt.

- Die Funktion **weekday** extrahiert die Zahl für den Wochentag aus dem Datenelement **Date** in der Quelldatei "ExpReport". Das Ergebnis dieser Funktion sind Ganzzahlen von 1 bis 7.
- Die Wertezuordnungskomponente transformiert die Ganzzahlen in Wochentage, also i. e. Sunday, Monday, usw., wie in der Grafik oben in diesem Abschnitt gezeigt.
- Wenn die Ausgabe den Wert "Tuesday" enthält, so wird die entsprechende Ausgabe "Prepare Financial Reports" auf das Datenelement "Notes" in der Zielkomponente gemappt.
- Wenn Sie auf die Schaltfläche "Ausgabe" klicken, wird die XML-Zieldatei mit den transformierten Daten angezeigt.

```
3    <Name>Landis</Name>
4    <expense-item>
5      <type>Meal</type>
6      <Weekday>Tuesday</Weekday>
7      <Notes>-- Prepare financial reports -- !</Notes>
8      <Date>2003-01-01</Date>
9      <expense>122.11</expense>
10   </expense-item>
11   <expense-item>
12     <type>Lodging</type>
13     <Weekday>Monday</Weekday>
14     <Notes> --</Notes>
15     <Date>2003-01-14</Date>
16     <expense>122.12</expense>
17   </expense-item>
```

Anmerkung:
Wenn Sie den Mauszeiger über die Wertezuordnungskomponente platzieren, wird ein Popup-Fenster geöffnet, das die aktuell definierten Werte enthält.

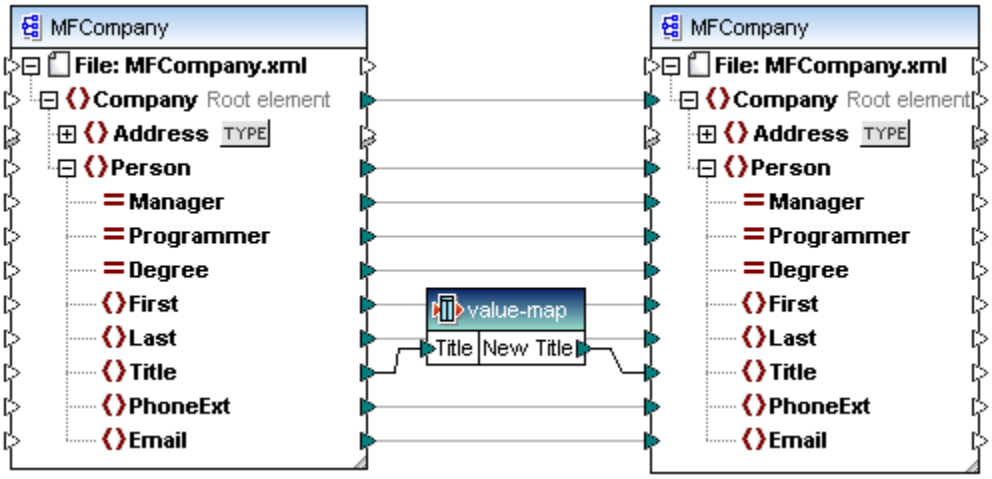
 © 2011 Altova GmbH

Die Ausgabe von verschiedenen Typen von logischen Funktionen oder String-Funktionen kann nur einen Bool'schen Wert "**true**" oder "**false**" haben. Der Wert, den Sie überprüfen möchten, muss daher in das **Input**-Feld der Wertezuordnungstabelle eingegeben werden, z.B. "true".

11.2.1 Übergeben von unveränderten Daten über eine Wertezuordnung

In diesem Abschnitt wird eine Mapping-Situation beschrieben, in der bestimmte Node-Daten transformiert werden müssen, während die restlichen Node-Daten unverändert an den Ziel-Node übergeben werden müssen.

Ein Beispiel dafür wäre eine Firma, die einige der Berufsbezeichnungen in einer ihrer Niederlassungen ändert. In diesem Fall sollen zwei Berufsbezeichnungen (Title) geändert werden, während der Rest unverändert beibehalten werden soll.

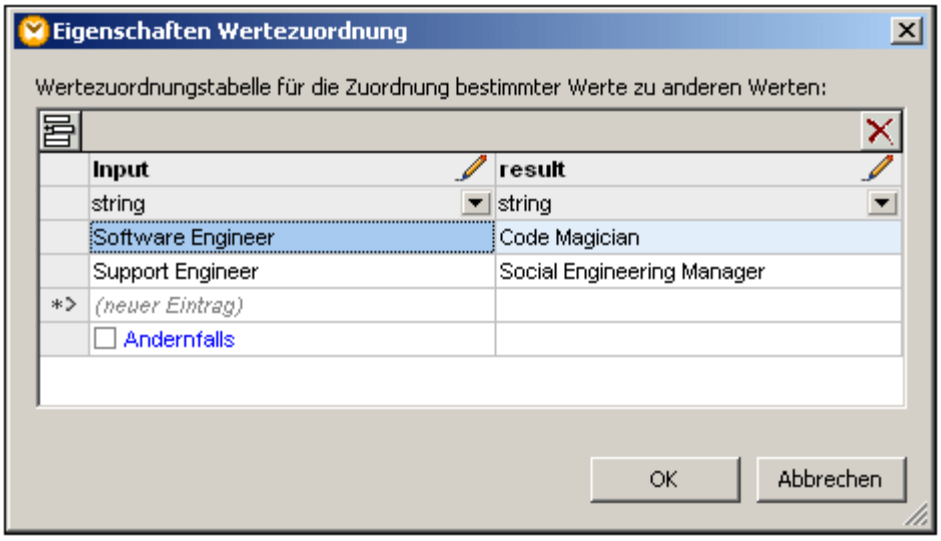


Oben sehen Sie ein Mapping dafür, in dem, wie erwartet, zum Transformieren der betreffenden Bezeichnungen eine Wertezuordnungskomponente verwendet wird.
Wenn Sie auf die Schaltfläche "Ausgabe" klicken, sehen Sie das Ergebnis des Mappings.

```
33  <Person>
34    <First>Fred</First>
35    <Last>Landis</Last>
36    <PhoneExt>951</PhoneExt>
37    <Email>f.landis@nanonull.com</Email>
38  </Person>
39  <Person>
40    <First>Michelle</First>
41    <Last>Butler</Last>
42    <Title>Code Magician</Title>
43    <PhoneExt>654</PhoneExt>
44    <Email>m.landis@nanonull.com</Email>
45  </Person>
```

Bei den Personen, deren Berufsbezeichnung (Title) mit keinem der beiden in der Wertezuordnungskomponente angezeigten Typen übereinstimmt, wird das Element "Title" in der Ausgabedatei gelöscht.

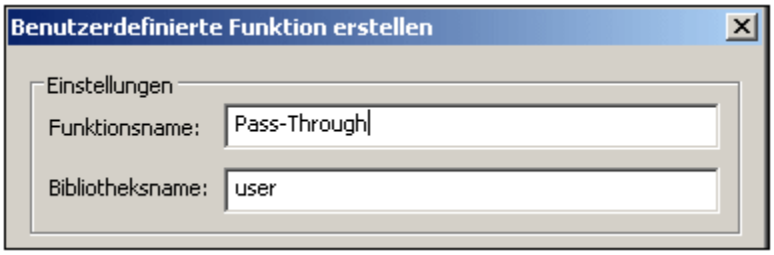
© 2011 Altova GmbH



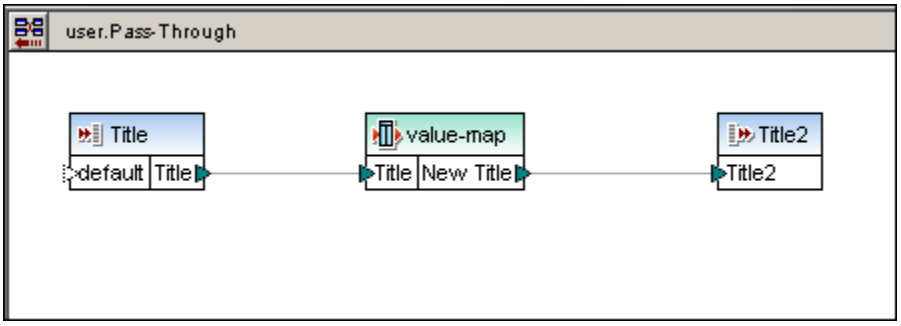
Mögliche Alternative:
Wenn Sie das Kontrollkästchen **Andernfalls** aktivieren und eine Ersatzbezeichnung eingeben, wird die Berufsbezeichnung in der Ausgabedatei zwar wieder angezeigt, enthält aber bei allen Personen in der Firma nun dieselbe **neue Berufsbezeichnung**.

Lösung:
Erstellen Sie eine benutzerdefinierte Funktion, die die Wertezuordnungskomponente enthält und verwenden Sie die **substitute-missing**-Funktion, um an leere Nodes die Originaldaten zu übergeben.

1. Klicken Sie auf die value-map-Komponente und wählen Sie den Befehl **Funktion | Benutzerdefinierte Funktion von Auswahl erstellen**.

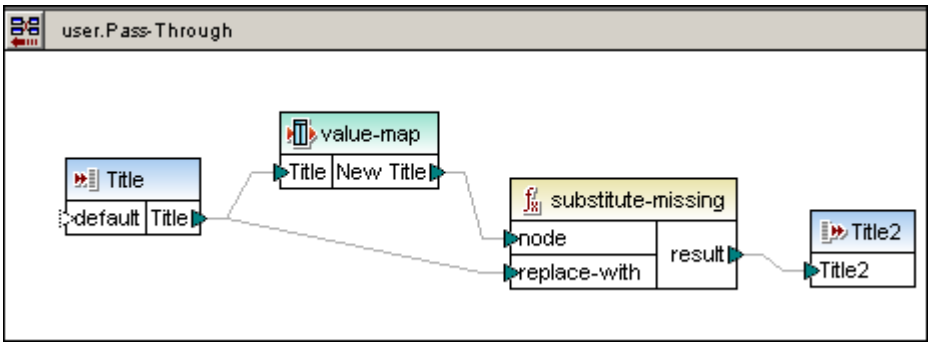


2. Geben Sie einen Namen für die Funktion ein, z.B. Pass-Through und klicken Sie auf OK.



3. Fügen Sie eine **substitute-missing**-Funktion aus dem Abschnitt **core | node function** des Bibliotheksfensters ein und stellen Sie, wie in der Abbildung unten gezeigt, die

Verbindungen her.



4. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis zu sehen:

Mapping-Ergebnis:

- Die beiden Berufsbezeichnungen in "Title" wurden in "New Title" transformiert.
- Alle anderen Title-Nodes der Quelldatei behalten ihre ursprünglichen Title-Daten in der Zieldatei bei.

```
38  <Person>
39    <First>Fred</First>
40    <Last>Landis</Last>
41    <Title>Program Manager</Title>
42    <PhoneExt>951</PhoneExt>
43    <Email>f.landis@nanonull.com</Email>
44  </Person>
45  <Person>
46    <First>Michelle</First>
47    <Last>Butler</Last>
48    <Title>Code Magician</Title>
49    <PhoneExt>654</PhoneExt>
50    <Email>m.landis@nanonull.com</Email>
51  </Person>
```

Warum geschieht das?
Die Wertezuordnungskomponente wertet die Input-Daten aus.

- Wenn die eingehenden Daten mit einem der Einträge in der ersten Spalte **übereinstimmen**, werden die Daten transformiert und an den Node-Parameter von substitute-missing und dann weiter an Title2 übergeben.

- Wenn die eingehenden Daten **mit keinem der Einträge in der linken Spalte übereinstimmen**, wird kein Wert von der Wertezuordnung an den Node-Parameter übergeben, d.h. dies ist ein **leerer Node**.

  Wenn das passiert, ruft die substitute-missing-Funktion den Original-Node und die Daten aus dem Title-Node ab und übergibt diese über den **replace-with** Parameter an Title2.

 © 2011 Altova GmbH

11.2.2 Eigenschaften der Wertezuordnungskomponente

Aktionen:

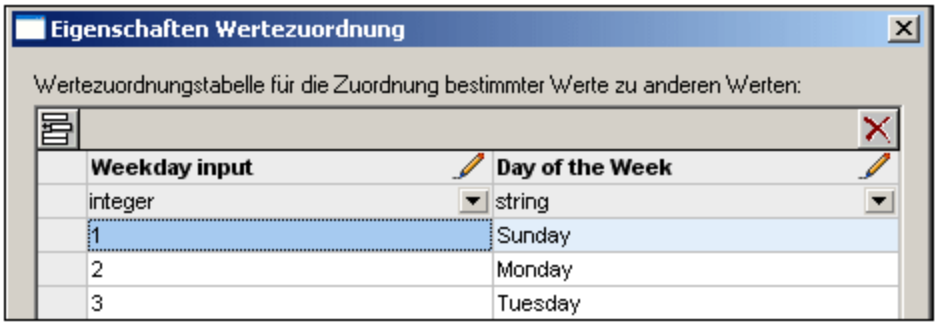
Klicken Sie auf die Schaltfläche "Einfügen", um vor der aktiven Zeile eine neue Zeile **einzufügen**.

Klicken Sie auf die Schaltfläche "Löschen", um die aktive Zeile zu **löschen**.

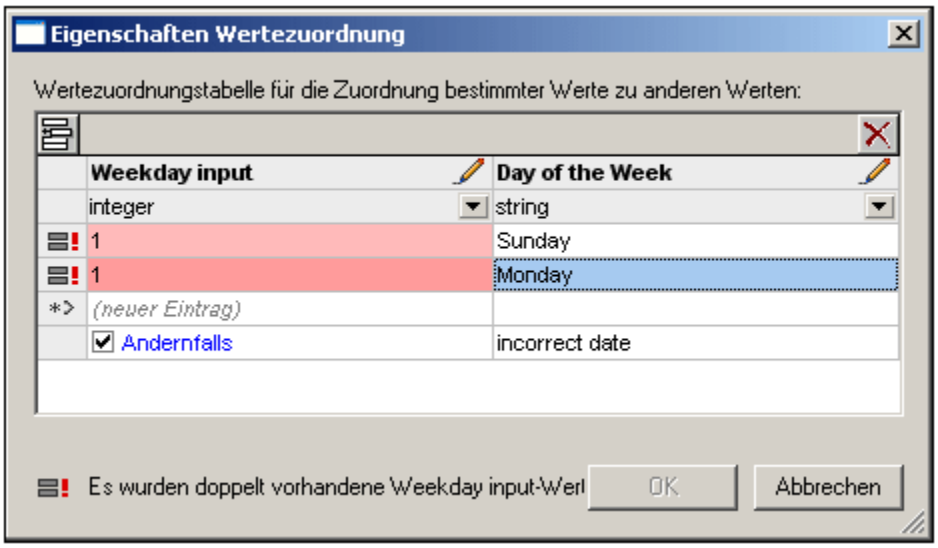
Klicken Sie auf die Schaltfläche "Bearbeiten", um die Spaltenüberschrift zu **bearbeiten.**

Sie können die Reihenfolge von Zeilen auch durch Ziehen mit der Maus ändern.

Ändern der Spaltenüberschrift:
Doppelklicken Sie auf die Spaltenüberschrift oder klicken Sie auf das Bleistiftsymbol, um den Spaltennamen zu bearbeiten und einen sinnvollen Namen einzugeben. Auf diese Art lässt sich der Zweck der Komponente besser erkennen, da die Spaltennamen auch im Mapping angezeigt werden.



Verwendung eindeutiger Input-Werte:
Die in die Spalte "Input" eingegebenen Werte müssen eindeutig sein. Wenn Sie zwei identische Werte eingeben, werden beide automatisch markiert, damit Sie einen davon korrigieren können.

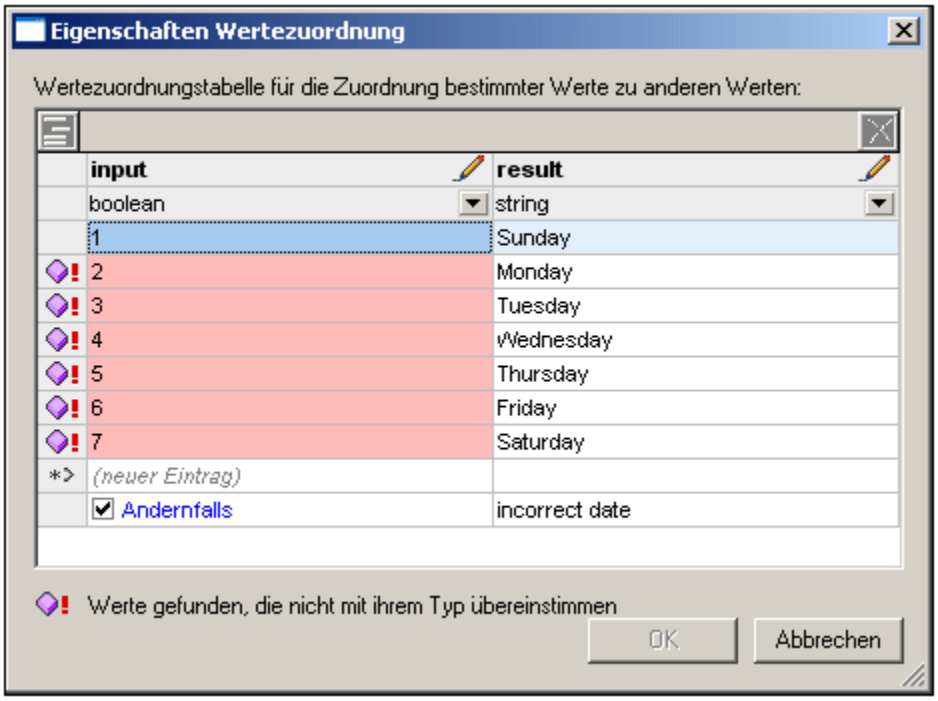


Sobald Sie einen der Werte korrigiert haben, ist die OK-Schaltfläche wieder aktiv.

Input- und Output-Datentypen

Die Input- und Ergebnis-Datentypen werden automatisch überprüft, wenn Sie eine Auswahl über die Auswahlliste treffen. Wenn ein Wert nicht übereinstimmt, werden die entsprechenden Felder markiert und die Schaltfläche "OK" wird deaktiviert. Ändern Sie den Datentyp in einen unterstützten Datentyp.

In der Abbildung unten wurden ein Boolescher Wert und ein String ausgewählt.

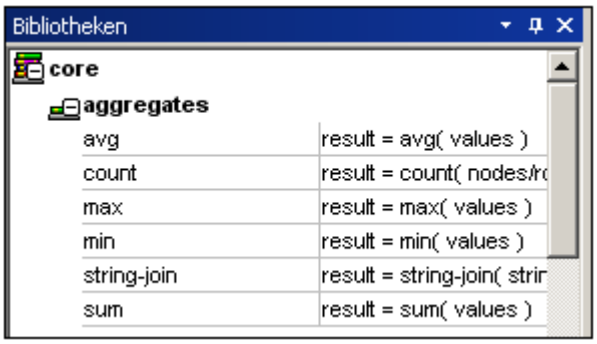


 © 2011 Altova GmbH

11.3 Aggregatfunktionen: min, max, sum, count, avg

Aggregatfunktionen dienen zum Ausführen von Operationen an einer Gruppe von Eingabewerten (oder an einer Folge von Werten) im Gegensatz zu einem einzigen Wert. Die Aggregatfunktionen befinden sich in der Core-Bibliothek. Das unten gezeigte Mapping finden Sie im Ordner ...\Tutorial unter dem Namen **Aggregates.mfd.**

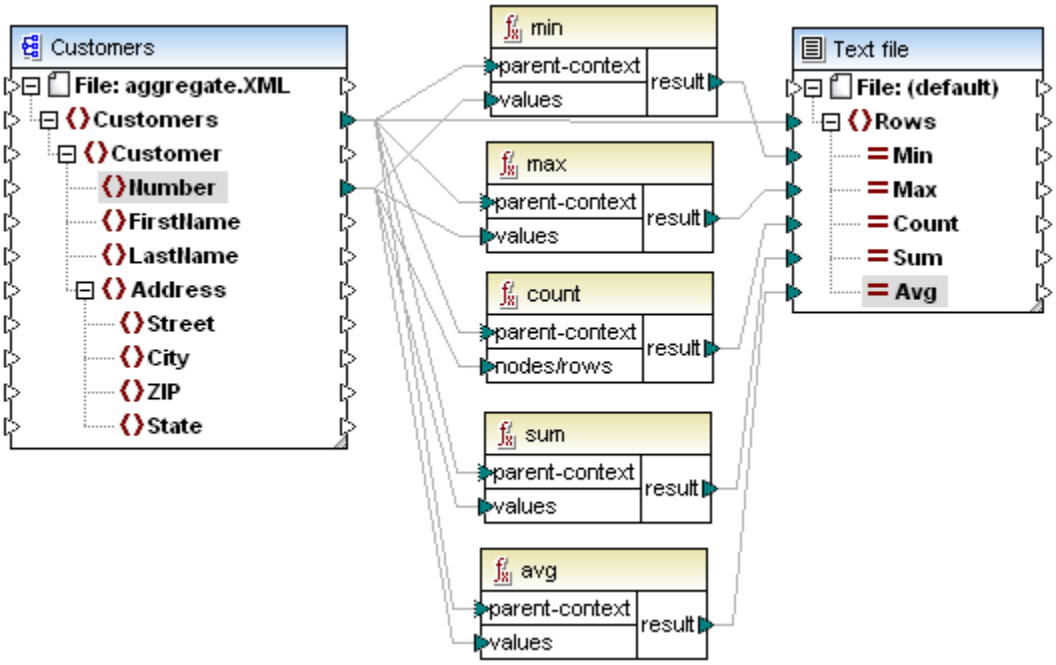
Informationen zum Erstellen von Aggregatfunktionen mit Hilfe von Named Templates finden Sie unter Aggregatfunktionen - Aggregieren von Nodes in XSLT1 und 2.



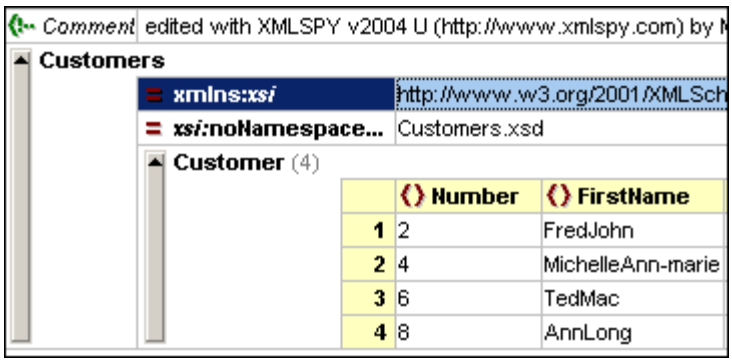
Java ausgewählt

Aggregatfunktionen haben zwei Input-Elemente.

- **values** (nodes/rows) ist mit dem Quellelement verbunden, aus dem die Daten stammen, in diesem Fall "Number".
- **parent-context** ist mit dem Element verbunden, an dem iteriert werden soll, in diesem Fall allen Kunden ("Customer"). Der Parameter ist allerdings optional.



Bei der Input-Instanz handelt es sich in diesem Fall um eine XML-Datei mit den folgenden Daten:



- Bei den Quelldaten, die an das Element "values" übergeben werden, handelt es sich um die Zahlenfolge 2,4,6,8.
- Bei der Output-Komponente handelt es sich in diesem Fall um eine einfache Textdatei. Wenn Sie für das obige Mapping auf das Register "Ausgabe" klicken, erhalten Sie das folgende Ergebnis:

```
1   2,8,4,20,5
2
```

min=2, max=8, count=4, sum=20 and avg=5.

 © 2011 Altova GmbH

11.4 Mappings und das Root-Element von Zieldokumenten

Root-Element von XML-Zieldateien
Stellen Sie beim Erzeugen eines Mappings auf das Root-Element des Zielschemas / der XML-Zieldatei bitte sicher, dass nur ein Element an die XML-Zieldatei übergeben wird, da ein XML-Dokument nur ein Root-Element haben darf.

Verwenden Sie die Filter-Komponente, um nur ein einziges Element bzw. nur einen Datensatz zu mappen.

Keine Einschränkung des Root-Elements:
Wenn Sie keine Einschränkung für das Root-Element des Zielschemas definieren, werden alle Elemente/Datensätze der Quelle zwischen dem ersten Root-Element eingefügt. Dabei werden zwar wohlgeformte, nicht aber gültige XML-Dateien erstellt.

11.5 Boolscher Vergleich von Input Nodes

Behandlung von Datentypen in Booleschen Funktionen (erstmals in MapForce 2006 SP2 eingeführt)
Während der Auswertung der Core-Funktionen: kleiner als, größer als, ist gleich, ist nicht gleich, kleiner gleich und größer gleich, hängt das ausgewertete Ergebnis zweier Input-Nodes sowohl von den Eingabewerten als auch von den für den Vergleich verwendeten Datentypen ab.

Beispiel:
Der Vergleich 'kleiner als' der Integerwerte 4 und 12 ergibt den Booleschen Wert "true", da 4 kleiner als 12 ist. Wenn die beiden Eingabestrings die 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"-Datentypen um denselben Typ handelt, wenn z.B. alle Input Node numerische Typen oder Strings sind, wird der Vergleich für den gemeinsamen Typ durchgeführt.

Unterschiedliche Input Node-Typen (nur Version SP2):
Wenn es sich bei den Input Nodes um unterschiedliche Typen handelt, z.B. um Integer-Werte und Strings oder Strings und Datumswerte, wird die folgende Regel angewandt:

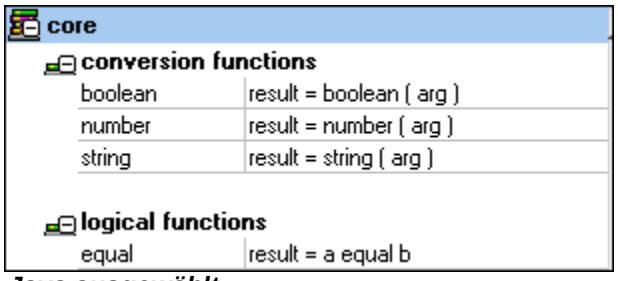
Der für den Vergleich verwendete Datentyp ist **immer der allgemeinste, d.h. der Input-Datentyp von den zwei Input-Typen mit den wenigsten Einschränkungen**.

Bevor die beiden Werte verglichen werden, werden alle Werte in einen Common Datatype konvertiert. Um beim vorhergehenden Beispiel zu bleiben: Der Datentyp "string" ist weniger restriktiv als der Wert "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.

Bei der Behandlung von Typen beim Vergleich von gemischten Typen wird nach den XSLT2-Richtlinien vorgegangen. Content-sensitive Typkonvertierungsstrategien werden verhindert. Der Vorteil ist, dass die Logik durch das Mapping fixiert wird und sich nicht dynamisch ändert.

Zusätzliche Überprüfungen:
Die Version 2006SP2 überprüft Mappings außerdem auf nicht kompatible Kombinationen und gibt gegebenenfalls Validierungsfehler und Warnungen aus. Beispiele sind der Vergleich von Daten mit Booleschen Werten oder "datetimes" mit numerischen Werten.

Zur Unterstützung expliziter Datentypkonvertierungen gibt es in der Core-Bibliothek die folgenden**Typkonvertierungsfunktionen**: "boolean", "number" und "string". Im oben genannten Zusammenhang eignen sich diese drei Funktionen zum Festlegen der Interpretation von Vergleichen.



Java ausgewählt

Wenn Sie diese Konvertierungsfunktionen zu Input Nodes damit in Zusammenhang stehender

 © 2011 Altova GmbH

Funktionen hinzufügen, können Sie dadurch den common datatype ändern und dadurch das gewünschte Ergebnis erzielen, z.B. können Sie, wenn String Nodes nur numerische Werte enthalten, einen numerischen Vergleich erzielen, indem Sie die Konvertierungsfunktion "number" (im Abschnitt "conversion" der **core**-Bibliothek) zu den einzelnen Input Nodes hinzufügen.

11.6 Prioritätskontext Node/Datenelement

Wenn Sie eine Funktion auf unterschiedliche Datenelemente in einem Schema anwenden, muss MapForce den Context Node kennen. Alle anderen Datenelemente werden dann relativ dazu verarbeitet. Dazu wird das gewünschte Datenelement (oder der Node) als Prioritätskontext definiert.

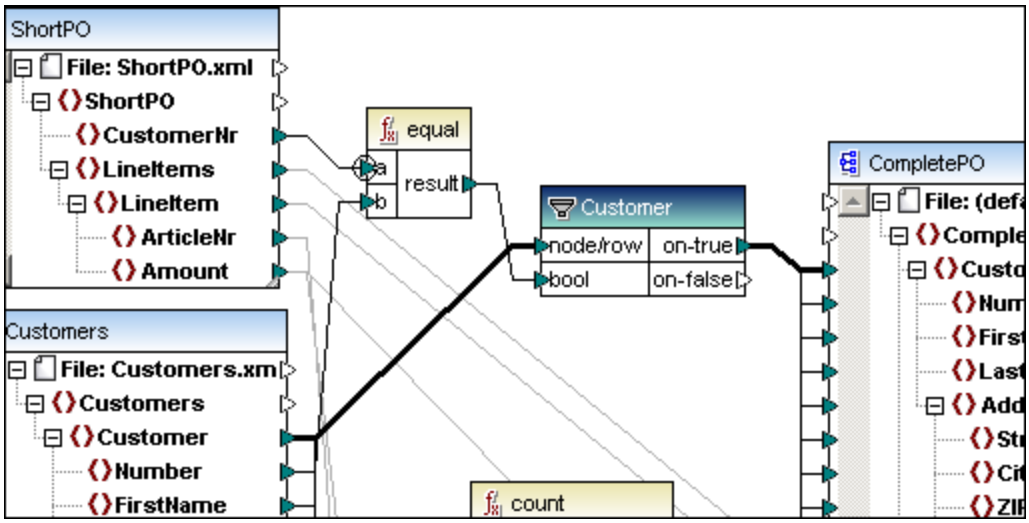
Unten sehen Sie die Datei **CompletePO.mfd** aus dem Ordner **...\MapForceExamples**.

Beachten Sie, dass es in diesem Beispiel mehrere Quellkomponenten gibt. **ShortPO, Customers und Articles** sind Schemas, die mit einer XML-Instanzdatei verknüpft sind. Die Daten aus diesen Komponenten werden nun auf das Schema / die XML-Datei "CompletePO" gemappt. Das Prioritätskontext-Symbol wird durch einen Kreis gekennzeichnet.

- Das Datenelement **CustomerNr** in ShortPO wird mit dem Datenelement **Number** in der Datei "Customers" verglichen.
- **CustomerNr** wurde als **Prioritätskontext** definiert und in den **a-**Parameter der equal-Funktion gesetzt.
- Anschließend wird die Datei **Customers** nach **derselben** Nummer durchsucht. Der **b-**Parameter enthält das Datenelement "Number" aus der Datei "Customers".
- Wird die Nummer gefunden, wird das Ergebnis an den **Boolschen** Parameter der **Filter**-Funktion übergeben.
- Der **node/row-**Parameter übergibt die **Customer**-Daten an "on-true", wenn der Boolesche Parameter "true" ist, d.h. wenn dieselbe Nummer gefunden wurde.
- Anschließend wird der Rest der Kundendaten übergeben: Die Datenelemente Number, FirstName, LastName werden alle mit den entsprechenden Datenelementen im Zielschema verbunden.

Wenn Sie den **b-**Parameter der equal-Funktion, d.h. die Datenelement-Nr. (Number), als **Prioritätskontext** definieren würden, hätte dies zur Folge dass:

- MapForce die erste Zahl in den **b-**Parameter lädt
- diese mit **CustomerNr** in **a** vergleicht; wenn sie nicht gleich ist
- die nächste Nummer in **b** lädt, diese mit **a** vergleicht und
- durch alle Number-Elemente in der Datei iteriert, um die entsprechende Nummer in ShortPO zu finden.



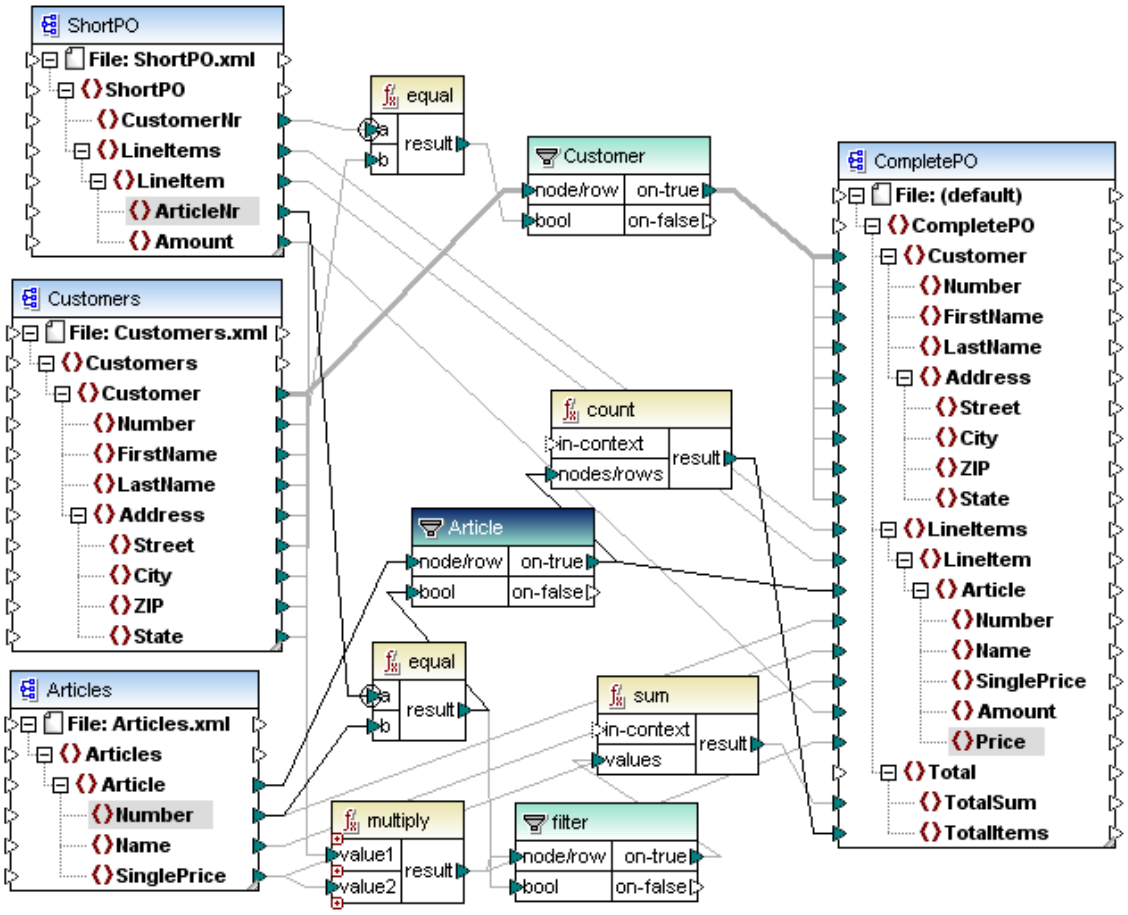
 © 2011 Altova GmbH

11.7 Zusammenführen mehrerer Dateien in einer Zieldatei

Sie können in MapForce mehrere Dateien in einer einzigen Zieldatei zusammenführen.

In diesem Beispiel werden mehrere Quellkomponenten mit unterschiedlichen Schemas zu einem Zielschema zusammengeführt. Informationen, wie Sie beliebig viele Dateien anhand desselben Schemas zusammenführen, finden Sie unter "Dynamische Dateinamen - Input / Output".

In der Datei **CompletePO.mfd** im Ordner **...\MapForceExamples** sehen Sie, wie drei XML-Dateien in einer XML-Bestelldatei zusammengeführt werden.



Beachten Sie, dass mehrere Quellkomponentendateien in einer XML-Zieldatei - CompletePO - kombiniert werden.

- **ShortPO** ist ein Schema mit einer dazugehörigen XML-Instanzdatei und enthält nur Kundennummern- und Artikeldaten, d.h. die Einträge LineItem, Number und Amount.
- **Customers** ist ein Schema mit einer dazugehörigen XML-Instanzdatei und enthält die Kundennummern sowie die Kundendaten, also Namen und Adresse (Diese Datei enthält nur einen Kunden mit der Kundennummer 3).
- **Articles** ist ein Schema mit einer dazugehörigen XML-Instanzdatei und enthält die Artikeldaten, d.h. den Artikelnamen, die Artikelnummer und den Preis.
- **CompletePO** ist eine Schemadatei ohne Instanzdatei, da alle Daten von den drei XML-Instanzdateien geliefert werden. Aufgrund der hierarchischen Struktur dieser Datei

können alle XML-Daten darin zusammengeführt und ausgegeben werden.

Diese Schemadatei muss in einem XML-Editor wie z.B. XMLSpy erstellt werden. Sie wird nicht von MapForce generiert (obwohl dies möglich wäre, wenn Sie eine Instanzdatei für CompletePO.xml hätten).

Die Struktur von CompletePO ist eine Kombination der Strukturen der XML-Quelldateien.

Mit Hilfe der **filter**-Komponente (Customer) werden die Daten, in denen die Kundennummern sowohl in der XML-Datei "ShortPO" als auch in der XML-Datei "Customers" identisch sind, gesucht/gefiltert und die verknüpften Daten werden an die Zielkomponente "CompletePO" übergeben.

- **CustomerNr** in ShortPO wird mit Hilfe der "equal"-Funktion mit **Number** in Customers verglichen.
- Da ShortPO nur einen Kunden (Nummer 3) enthält, können nur die Kunden und Artikeldaten für den Kunden Nr. 3 an die Filterkomponente übergeben werden.
- Der **node/row-**Parameter der Filterkomponente übergibt die **Customer**-Daten (Kundendaten) an "on-true", wenn der Boolsche Parameter "true" ist, d.h. wenn dieselbe Nummer gefunden wurden, in diesem Fall Kundennummer 3.
- Die restlichen Kunden- und Artikeldaten werden über die beiden anderen Filterkomponenten an das Zielschema übergeben.

© 2011 Altova GmbH

11.8 Befehlszeilenparameter

Die Befehlszeilenparameter-Syntax für MapForce wird unten gezeigt.

Allgemeine Syntax:
MapForce.exe *filename* **[/***target outputdir* **[/GLOBALRESOURCEFILE** *globalresourcefilename* **] [/GLOBALRESOURCECONFIG** *configurationname* **] [/LOG** *logfilename* **]]**

target **{ XSLT | XSLT2 }**

- Die eckigen Klammern **[...]** kennzeichnen optionale Parameter.
- Die geschwungenen **{...}** kennzeichnen eine Parametergruppe mit mehreren Wahlmöglichkeiten.
- Das **pipe** Symbol **|** steht für OR, z.B. /XSLT oder /JAVA.

Die MapForce.exe gibt einen Exit-Code 0 zurück, wenn die Ausführung über die Befehlszeile erfolgreich war. Jeder andere Wert gibt einen Fehler an. Sie können Batch-Dateien mit dem Befehl IF ERRORLEVEL überprüfen.

Beschreibung allgemeiner Parameter:

filename	Pfad und YourMAPFORCEfile.MFD **Wenn der Pfad oder der Dateiname ein Leerzeichen enthält, setzen Sie den Pfad bitte in Anführungszeichen, also "c:\Program Files\...\Filename"**
outputdir	Das Verzeichnis, in das bei der Codegenerierung der generierte Mapping-Code platziert werden soll. *outputdir* ist nun *optional*, wenn Sie den /target-Parameter verwenden. Wenn kein Ausgabepfad angegeben wird, wird das aktuelle Arbeitsverzeichnis verwendet. Relativ bedeutet hier, dass nur der Dateiname und nicht der gesamte Pfad beginnend mit einem Laufwerksbuchstaben angegeben wird.
/LOG *logfilename*	Generiert eine Log-Datei mit dem Namen *logfilename*. *Logfilename* kann ein vollständiger Pfadname sein, d.h. das Verzeichnis und der Dateiname der Log-Datei, doch muss das Verzeichnis für die zu generierende Log-Datei vorhanden sein, wenn ein vollständiger Pfadname angegeben wird. Wenn Sie nur den Dateinamen angeben, wird die Datei in das Verzeichnis */outputdir* platziert.

Beschreibung von Zielparametern:

/XSLT	generiert XSLT1-Code
/XSLT2	generiert XSLT2-Code

Beschreibung von Parametern für globale Ressourcen:

/GLOBALRESOURCEFILE *globalresourcefilename*	verwendet die in der angegebenen globalen Ressourcendatei definierten globalen Ressourcen
/GLOBALRESOURCECONFIG *configurationname*	verwendet die angegebene globale Ressourcenkonfiguration

Beispiele:

MapForce.exe *filename* startet MapForce und öffnet die durch *filename* definierte Datei.

I) Alle XSLT-Dateien und Log-Datei ausgeben.
MapForce.exe *filename* **/XSLT** *outputdir* **/LOG** *logfilename*

II) Alle XSLT-Dateien generieren und alle globalen Ressourcen der globalen Ressourcendatei für die definierte Konfiguration verwenden

Mapforce.exe *filename* **/XSLT** *outputdir* **/GLOBALRESOURCEFILE** *globalresourcefilename* **/GLOBALRESOURCECONFIG** *configurationname*

Bei Ausführung der Befehlszeile werden mehrere Dateien erzeugt:

- die generierte XSLT-Datei
- eine Batch-Datei namens **DoTransform.bat,** die die XML-Datei mit Hilfe von AltovaXML transformiert.
- eine Log-Datei, falls in der Befehlszeile eine definiert wurde.

So transformieren Sie die XML-Datei mit Hilfe der generierten XSLT-Datei:

1. Laden Sie den kostenlosen AltovaXML-Prozessor von der AltovaXML-Download-Seite herunter und installieren Sie ihn. AltovaXML wird standardmäßig im Verzeichnis c:\Program Files\Altova\AltovaXML2012\ installiert.
2. Starten Sie die Batch-Datei **DoTransform.bat,** die sich im zuvor angegebenen Ausgabeordner befindet.
   Dadurch wird die Ausgabedatei im aktuellen Ordner generiert.

Anmerkung:
Eventuell müssen Sie den Installationspfad von AltovaXML zur Pfadvariablen der Umgebungsvariablen hinzufügen.

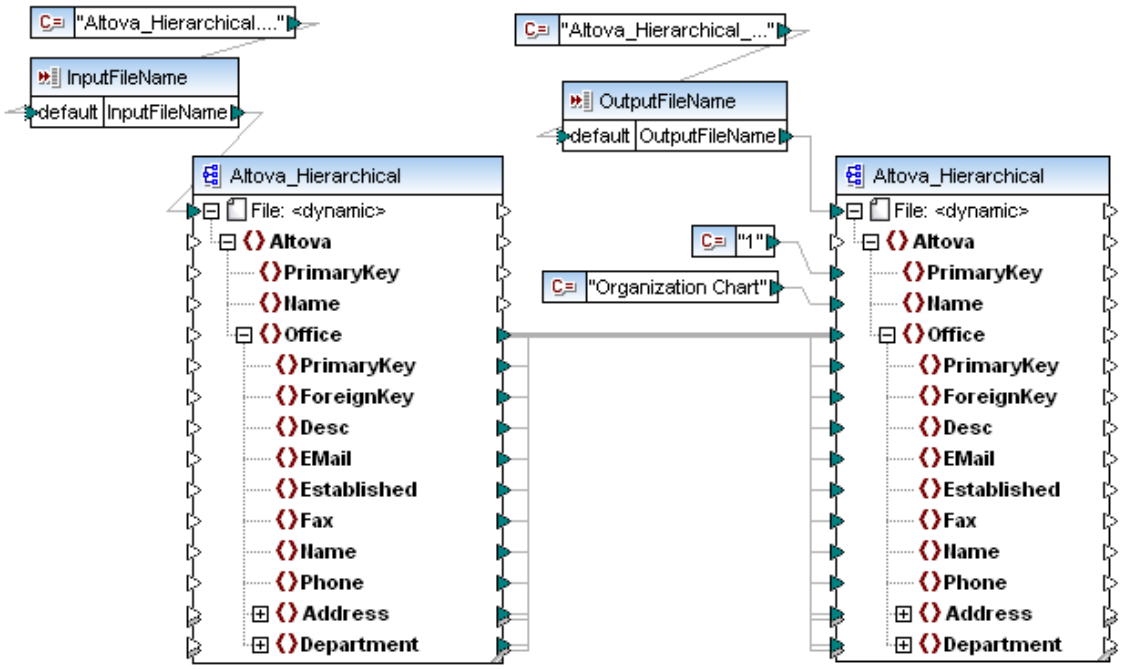
© 2011 Altova GmbH

11.9 Befehlszeile - Definieren von Input/Output-Dateien

MapForce gestattet das Erstellen spezieller "**Input"**-Komponenten, die als **Parameter** in der Befehlszeilenausführung des kompilierten Mappings fungieren können. Mit Hilfe dieser Input-Komponenten können Sie definieren, welchen Inhalt der Parameter bei der Ausführung der Befehlszeile haben soll und welche Werte in der Vorschau der Ausgabe in MapForce angezeigt werden sollen. Der Inhalt des Parameters kann ein Dateiname oder ein bestimmter Wert sein.

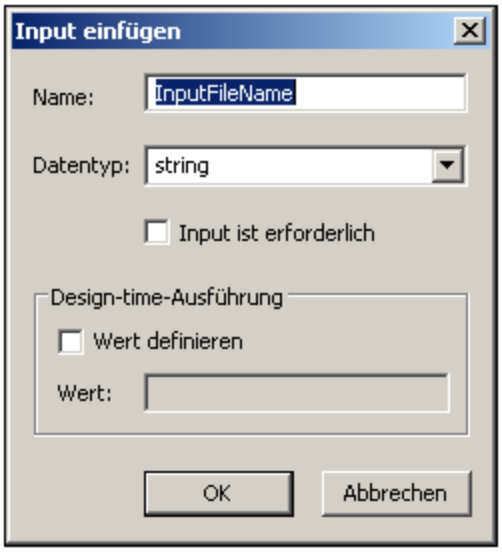
Diese spezifische Art von "Input"-Komponente kann nicht **innerhalb** einer benutzerdefinierten Funktion verwendet werden, sondern steht nur im Hauptmapping zur Verfügung.

Im unten gezeigten Mapping (**FileNamesAsParameters.mfd** in ...\MapForceExamples) werden zwei derartige Komponenten verwendet, eine liefert den Namen der Input-Datei und die andere definiert den Namen der Output-Datei für das Datenelement "Datei: <dynamisch>" der einzelnen Komponenten. Informationen zur Definition dynamischer Dateinamen finden Sie unter Dynamische Input/Output-Dateien pro Komponente.



So definieren Sie eine Input-Komponente / einen Befehlszeilenparameter:

1. Verwenden Sie die Menüoption **Funktion | Input-Komponente einfügen**, um die Komponente einzufügen.
   Daraufhin wird das Dialogfeld "Input einfügen" geöffnet.
2. Geben Sie einen Namen für die Funktion ein, z.B. **InputFileName** und wählen Sie den gewünschten Datentyp aus, z.B. string.



3. Klicken Sie auf OK, um die Definition fertigzustellen.
4. Definieren Sie auf dieselbe Art und Weise die Komponente / den Parameter **OutputFileName.**

   Die Komponenten /**InputFileName** und /**OutputFileName** sind die speziellen Input-Komponenten im Mapping, die Sie bei der Ausführung über die Befehlszeile als Dateinamenparameter verwenden können.

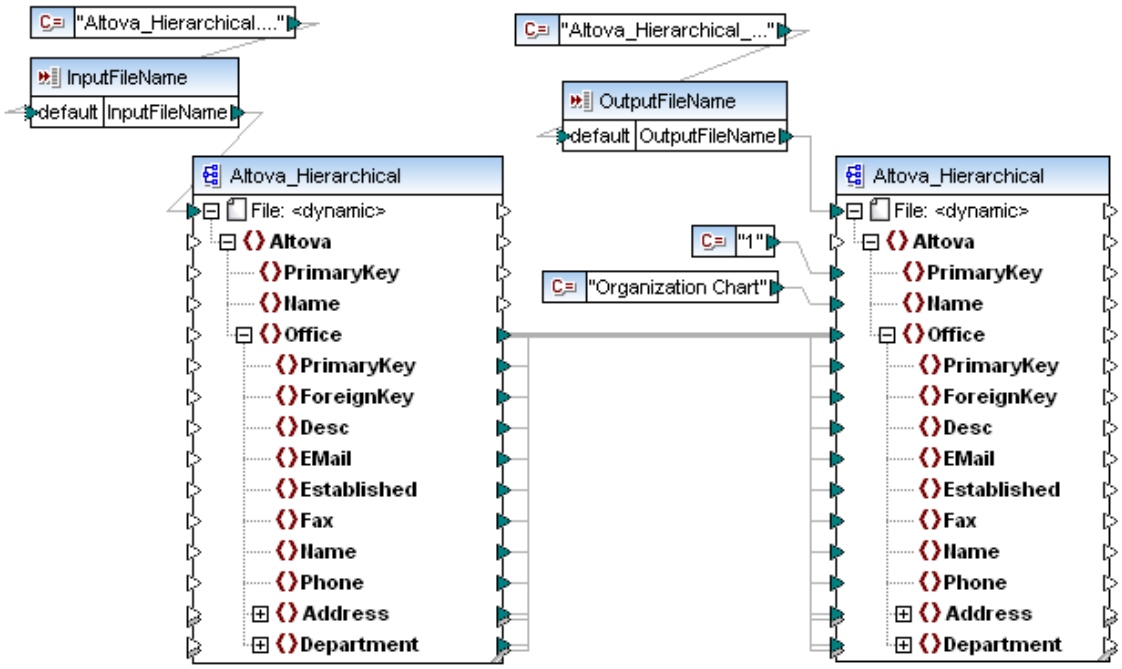
   Die anderen Felder in diesem Dialogfeld dienen dazu die Ausgabe der Vorschau (bzw. der Dateien) bei der Ausführung von MapForce zu definieren. Sie werden bei der Ausführung über die Befehlszeile nicht benötigt. Die beiden im obigen Mapping gezeigten Konstantenkomponenten liefern an die Input-Komponenten Standarddateinamen für die Ausgabe der Vorschau. Nähere Informationen dazu finden Sie unter Input-Parameter - Standardeinstellungen und Einstellungen für die Vorschau.

 © 2011 Altova GmbH

11.10 Input-Parameter - Standardeinstellungen und Einstellungen für die Vorschau

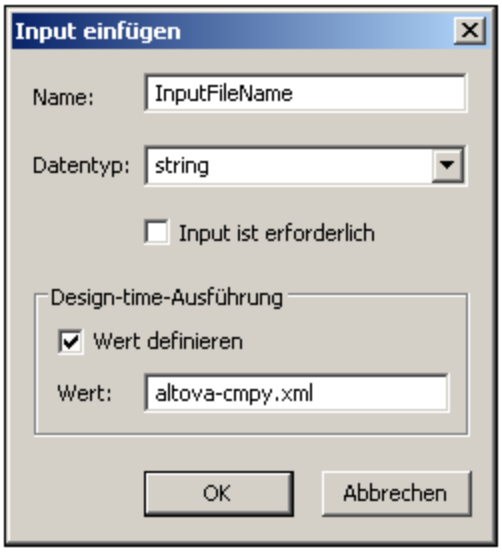
Im unten gezeigten Mapping (**FileNamesAsParameters.mfd** im Ordner ...\MapForceExamples) werden zwei Input-Komponenten verwendet, eine liefert den Namen der Input-Datei und die andere definiert den Namen der Output-Datei für das Datenelement "Datei: <dynamisch>" der einzelnen Komponenten.

Die Komponenten **InputFileName** und **OutputFileName** sind die speziellen Input-Komponenten im Mapping, mit Hilfe derer diese als Dateinamenparameter bei der Ausführung über die Befehlszeile verwendet werden können. Wenn Sie die Input-Daten über die Befehlszeile definieren möchten, lesen Sie bitte den Abschnitt Befehlszeile - Definieren von Input/Output-Dateien.



So definieren Sie eine Input-Komponente:

1. Verwenden Sie die Menüoption **Funktion | Input-Komponente einfügen**, um die Komponente einzufügen.
   Daraufhin wird das Dialogfeld "Input einfügen" geöffnet.
2. Geben Sie einen Namen für die Funktion ein, z.B. **InputFileName** und wählen Sie den gewünschten Datentyp aus, z.B. string.



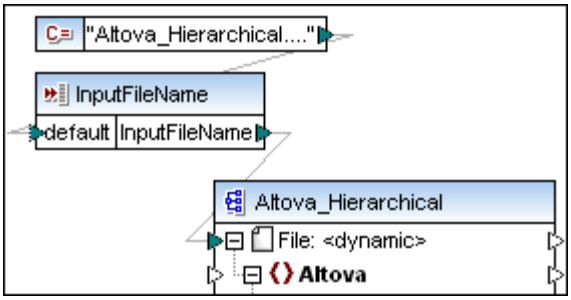
3. Aktivieren Sie das Kontrollkästchen "Wert definieren" und geben Sie den Wert/String, den Sie in der Ausgabevorschau verwenden möchten, in das Textfeld ein, z.B. altova-cmpy.xml.
4. Klicken Sie auf die Schaltfläche "OK" und anschließend auf die Schaltfläche "Ausgabe", um das Ergebnis zu sehen.
   Die Datei "altova-cmpy.xml" wird als Quelldatei für das Mapping verwendet und die gemappten Daten werden im Ausgabefenster angezeigt.

   Der Wert, der im Bereich "Design-time-Ausführung" in das Feld "**Wert**" eingegeben wurde, wird **nur** bei der **Vorschau** der Ergebnisse auf dem Register **Ausgabe** angewendet. Er wird nicht für die Codegenerierung oder die Mapping-Ausführung über die Befehlszeile verwendet.

So definieren Sie einen Standardwert für die Ausgabevorschau:

Nachdem Sie die Input-Komponente definiert und auf die Schaltfläche "OK" geklickt haben, wird die Komponente wie unten gezeigt im Mapping-Bereich angezeigt. Beachten Sie, dass auf der linken Seite der Eintrag "default" angezeigt wird, während der Name des anderen mapbaren Eintrags der Komponentenname ist.

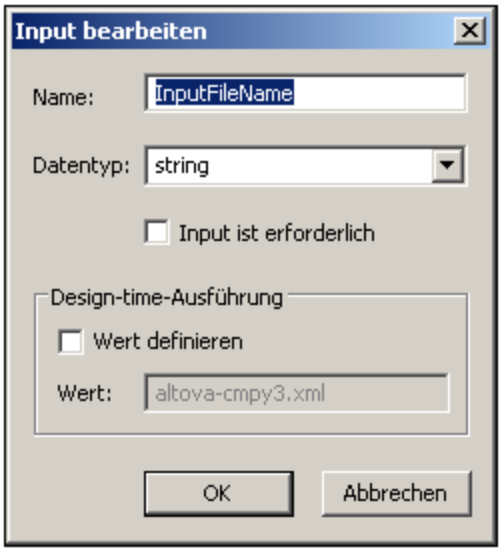
1. Sie können den Standardwert, d.h. den Wert von "default", wie oben gezeigt, mit Hilfe einer Konstantenkomponente angeben, z.B. Altova_Hierarchical.xml.
2. Erstellen Sie die Konstantenkomponente und verbinden Sie sie mit dem default-Element der Input-Komponente.



 © 2011 Altova GmbH

So verwenden Sie den Standardwert einer Input-Komponente:

1. Doppelklicken Sie auf die Input-Komponente und deaktivieren Sie die Kontrollkästchen "Input ist erforderlich" und "Wert definieren".



2. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis des Mappings zu sehen. Die Daten aus der Datei Altova_Hierarchical.xml werden nun für die Vorschau verwendet.

Generieren von XSLT 1.0- oder XSLT 2.0-Code:
Der Wert aus dem Textfeld "Wert" wird in die Batch-Datei **DoTransform.bat** geschrieben. Diese Batch-Datei wird automatisch für die Ausführung im AltovaXML-Prozessor generiert. Wenn Sie eine andere Input- (oder Output)-Datei verwenden möchten, können Sie den Namen in der Batch-Datei ändern.

Wenn kein Wert in das Textfeld "Wert" eingegeben wurde, so wird der Standardwert, der im generierten XSLT-Code vorhanden ist, verwendet.

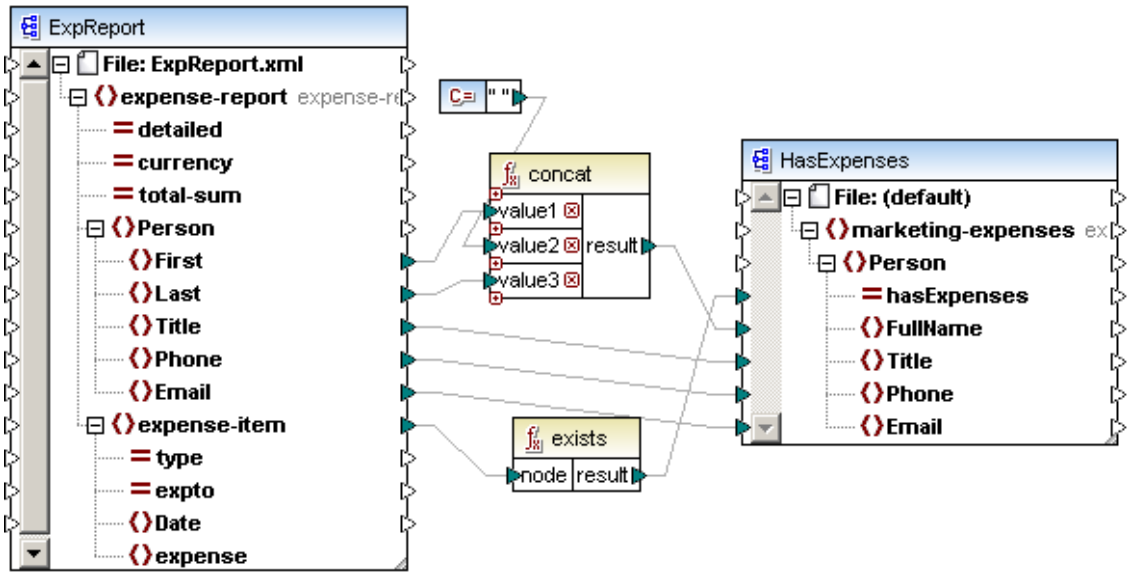
Verwendung von Standardparametern/Standardwerten in der Ausführung über die Befehlszeile:
Damit der Standardparameter aus der Befehlszeile verwendet wird, muss das Kontrollkästchen "Input ist erforderlich" (vor der Codegenerierung) **deaktiviert werden.** Geben Sie **mapping.exe** ein, damit der generierte Code den Standardparameter aus der Befehlszeile verwendet.

11.11 Node-Tests, Position und Gruppierung

Die Funktionen zum Testen von Nodes gestatten Ihnen, zu überprüfen, ob bestimmte Nodes in den **XML-Instanzdateien** existieren. Elemente oder Attribute, die im XML-Schema als optional definiert sind, können - müssen aber nicht - in der XML-Instanzdatei vorkommen. Mit Hilfe dieser Funktionen können Sie bestimmte Nodes überprüfen und dieses Resultat einer weiteren Verarbeitung zugrunde legen.

Exists
Gibt den Wert "true" zurück, wenn der Node vorhanden ist, andernfalls ist das Resultat "false". Die Datei "HasMarketingExpenses.mfd" im Ordner **...\MapForceExamples** enthält das unten gezeigte kurze Beispiel.

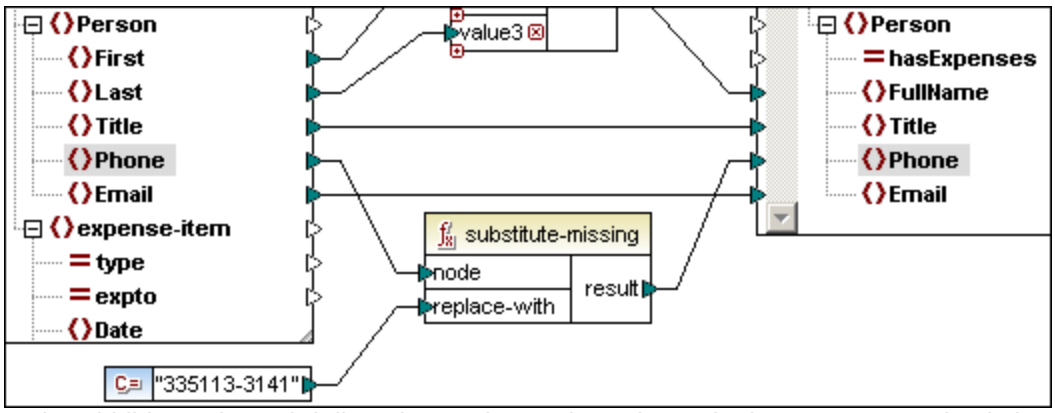
Wenn ein Datenelement "expense-item" in der XML-Quelldatei existiert, wird das Attribut "hasExpenses" in der XML/Schema-Zieldatei auf "true" gesetzt.



Not-exist
Gibt den Wert "false" zurück, wenn der Node existiert. Andernfalls ist das Resultat "true". Ein Beispiel zum Mappen von fehlenden Nodes finden Sie unter Mappen fehlender Nodes - mit Hilfe von "Not-exist".

substitute-missing
Diese Funktion ist eine praktische Kombination aus **exists** und einer passenden **if-else** -Bedingung. Wird verwendet, um den Inhalt des aktuellen Felds zu mappen, wenn der Node in der XML-Quelldatei existiert, andernfalls wird das Datenelement verwendet, das auf den Parameter "replace-with" gemappt ist.

© 2011 Altova GmbH



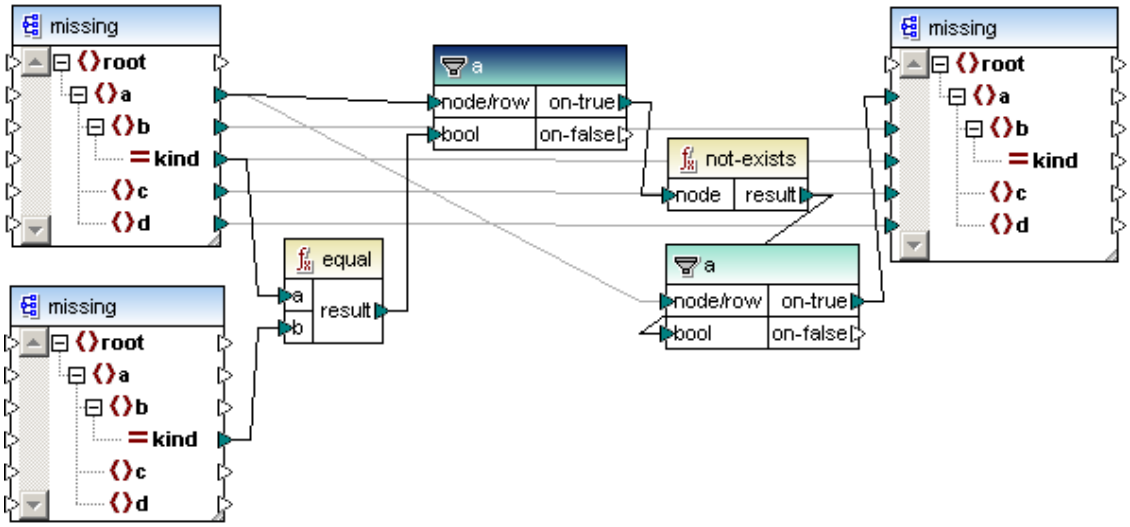
In der Abbildung oben wird die Existenz des Nodes "Phone" in der XML-Instanzdatei überprüft. Wenn dieser Node fehlt, wird der von der Konstante bereitgestellte Wert gemappt.

11.11.1 Mappen fehlender Nodes - mit Hilfe von Not-exists

Im Beispiel unten sehen Sie, wie Sie Nodes, die in einer von zwei Quelldateien fehlen, mit Hilfe der not-exists-Funktion mappen können.

Funktionsweise dieses Mappings:

- Vergleich der Nodes von zwei XML-Quelldateien
- Herausfilterung der Nodes der ersten XML-Quelldatei, die in der zweiten XML-Datei nicht vorhanden sind
- Mappen nur der fehlenden Nodes und ihres Inhalts auf die Zieldatei.

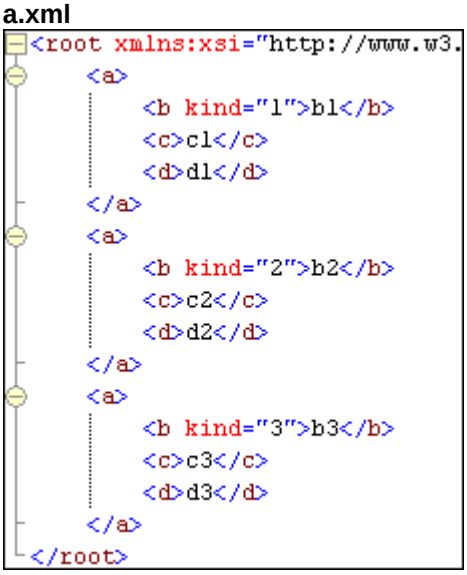


 © 2011 Altova GmbH

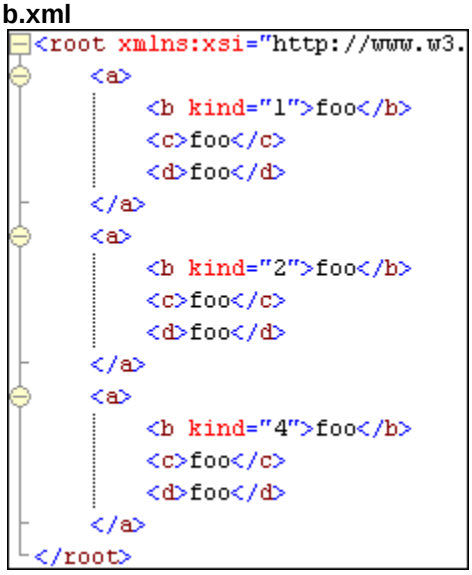
Die beiden XML-Instanzdateien werden unten angezeigt. Die Unterschiede zwischen ihnen sind die folgenden:

- **a.xml** auf der linken Seite enthält den Node <b **kind="3"**>, der in b.xml fehlt.
- **b.xml** auf der rechten Seite enthält den Node <b **kind="4">,** der in a.xml fehlt.

a.xml

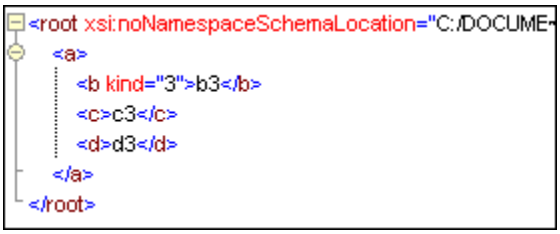


b.xml



- Die **equal-**Funktion vergleicht das Attribut **kind** der beiden XML-Dateien und übergibt das Ergebnis an den Filter.
- Die Funktion **not-exists** wird hinter dem Anfangsfilter platziert und wählt die fehlenden Nodes der beiden Quelldateien aus.
- Über den zweiten Filter werden der fehlende Node und andere Daten **ausschließlich** aus der Datei a.xml an die Zieldatei übergeben.

Das Ergebnis des Mappings ist, dass der in der Datei b.xml fehlende Node, <b kind="3">, an die Zielkomponente übergeben wird.

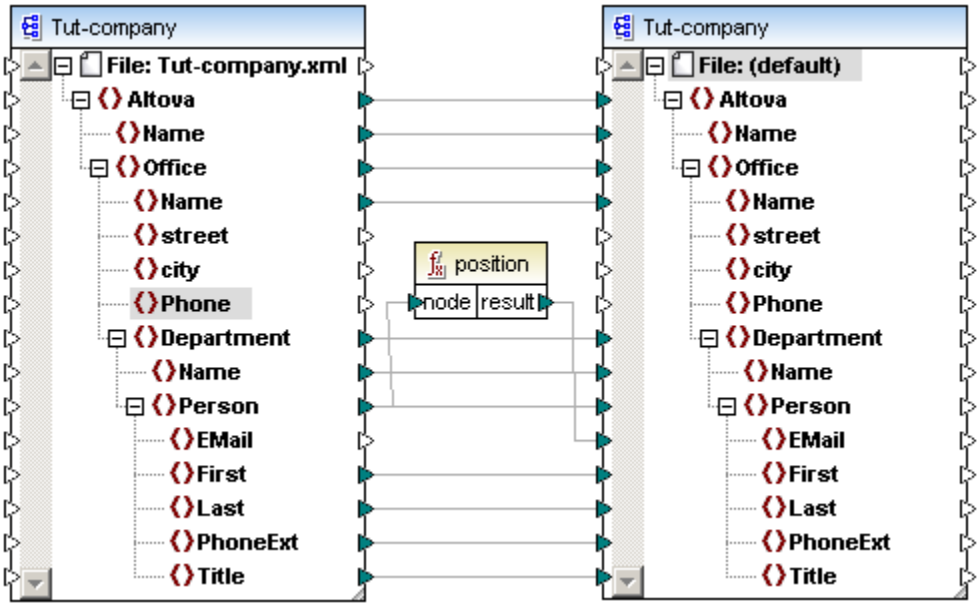


11.11.2 Position von Kontextelementen in einer Sequenz

Mit Hilfe der Positionsfunktion können Sie die Position bestimmter Nodes in einer Sequenz ermitteln oder anhand einer bestimmten Position Datenelemente herausfiltern.

Das Kontextelement wird durch das mit dem "node"-Parameter der Positionsfunktion verbundene Datenelement definiert, im Beispiel unten "Person".

Im unten gezeigten einfachen Mapping wird zu jedem Person-Element jeder Abteilung (Department) eine Positionsnummer hinzugefügt.



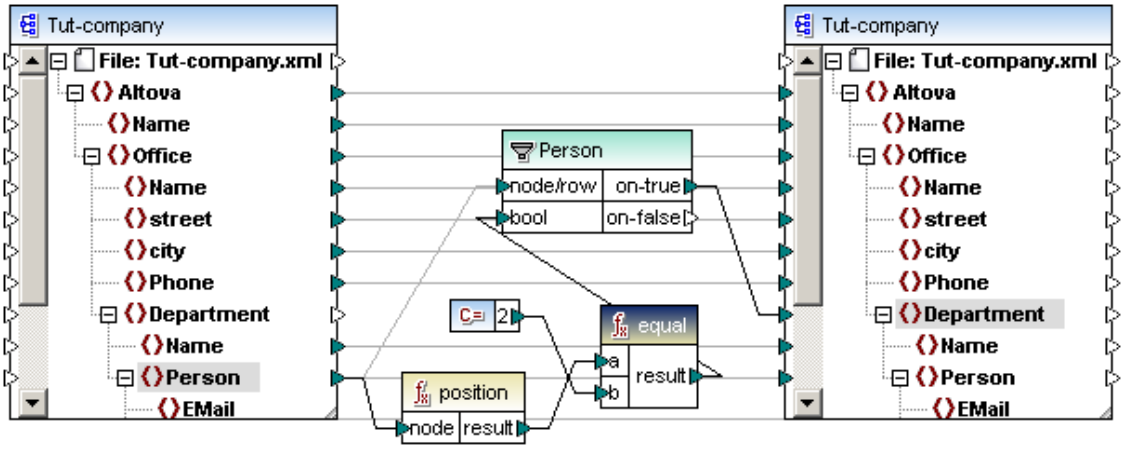
Die Positionsnummer wird für jede Abteilung im Büro zurückgesetzt.

```
<Office>
  <Name>Microtech, Inc.</Name>
  <Department>
    <Name>Admin</Name>
    <Person>
      <EMail>1</EMail>
      <First>Albert</First>
      <Last>Aldrich</Last>
      <PhoneExt>582</PhoneExt>
      <Title>Manager</Title>
    </Person>
    <Person>
      <EMail>2</EMail>
      <First>Bert</First>
      <Last>Bander</Last>
      <PhoneExt>471</PhoneExt>
      <Title>Accounts Receivable</Title>
    </Person>
```

Herausfiltern bestimmter Nodes mit Hilfe der Positionsfunktion
Mit Hilfe der Positionsfunktion in Verbindung mit einem Filter können Sie nur die Nodes, die eine bestimmte Position in der Quellkomponente haben, mappen.

 © 2011 Altova GmbH

Es scheint, dass der Filterparameter "node/row" und die position "node"-Funktion mit demselben Datenelement der Quellkomponente verbunden werden müssen, um eine bestimmte Position dieser Sequenz herauszufiltern.



Das Ergebnis des Mappings ist Folgendes:

- Ausgabe der **zweiten** Person in jeder Abteilung (Department)
- von jedem Büro bei Altova.

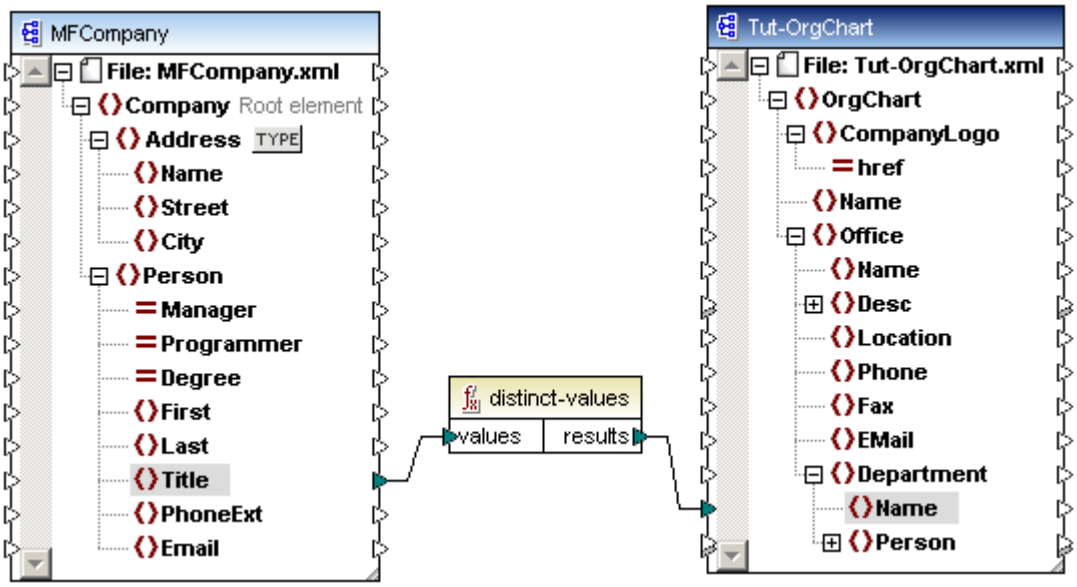
```
<Office>
  <Name>Microtech, Inc.</Name>
  <Department>
    <Name>Admin</Name>
    <Person>
      <EMail>b.bander@microtech.com</EMail>
      <First>Bert</First>
      <Last>Bander</Last>
      <Title>Accounts Receivable</Title>
    </Person>
  </Department>
  <Department>
    <Name>Sales and Marketing</Name>
    <Person>
      <EMail>e.ellas@microtech.com</EMail>
      <First>Eve</First>
      <Last>Ellas</Last>
      <Title>Art Director</Title>
    </Person>
  </Department>
  <Department>
    <Name>Manufacturing</Name>
    <Person>
      <EMail>g.gundall@microtech.com</EMail>
```

11.11.3 Gruppieren von Nodes / Node-Inhalt

MapForce unterstützt nun die Gruppierung von Nodes und ihrem Inhalt. Sie finden diese Funktionen im Fenster "Bibliotheken" im Abschnitt "Sequence functions".

distinct-values
Mit Hilfe dieser Funktion können Sie doppelt vorhandene Werte aus einem Ergebnis entfernen und die eindeutigen Datenelemente auf die Zielkomponente mappen.

Im unten gezeigten Beispiel wird der Inhalt der Quellkomponente "Title" durchsucht und jedes eindeutige Title-Element wird auf den Department / Name-Node der Zielkomponente gemappt.



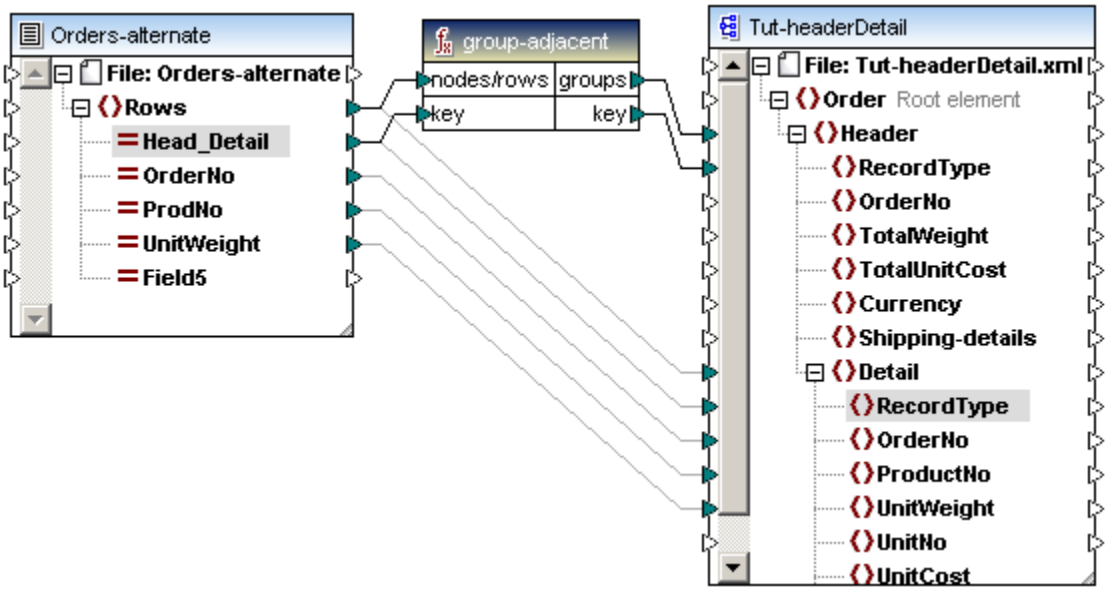
Beachten Sie, dass die Reihenfolge der einzelnen Title-Elemente in der Quellkomponente beim Mappen auf die Zielkomponente beibehalten wird.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <OrgChart xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart C:/DOCU
3    <Office>
4      <Department>
5        <Name>Office Manager</Name>
6        <Name>Accounts Receivable</Name>
7        <Name>Accounting Manager</Name>
8        <Name>Marketing Manager Europe</Name>
9        <Name>Art Director</Name>
10       <Name>Program Manager</Name>
11       <Name>Software Engineer</Name>
12       <Name>Technical Writer</Name>
13       <Name>IT Manager</Name>
14       <Name>Web Developer</Name>
15       <Name>Support Engineer</Name>
16       <Name>PR &amp;amp; Marketing Manager US</Name>
17     </Department>
18   </Office>
19 </OrgChart>
```

group-adjacent
Gruppiert die Input-Sequenz in einer Reihe von Gruppen, wobei jede Gruppe identisch

 © 2011 Altova GmbH

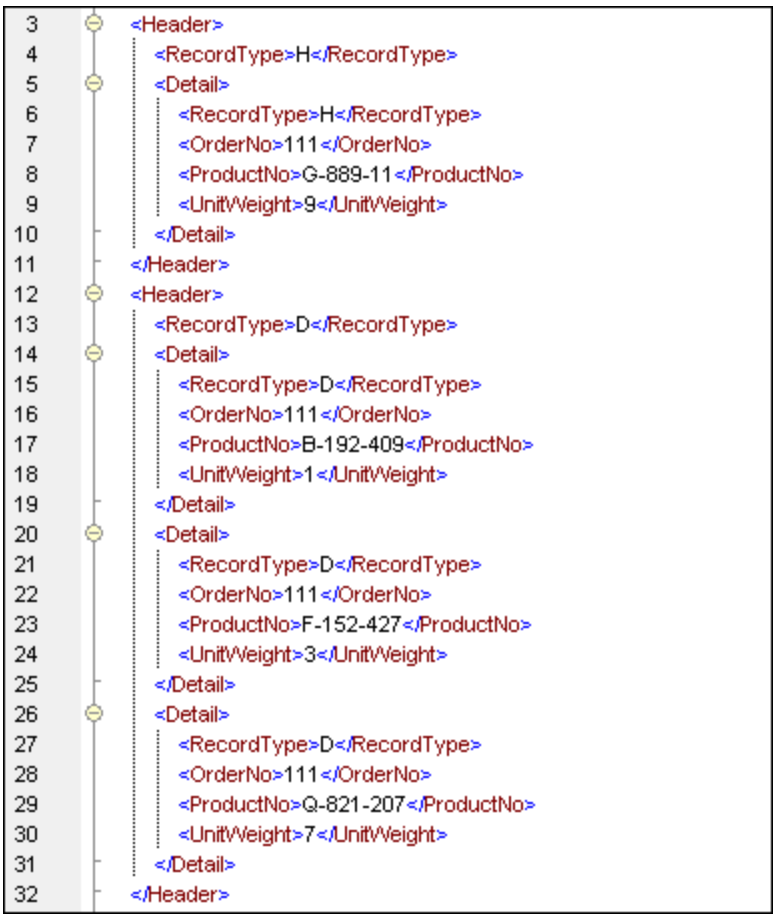
nebeneinanderliegender Datenelemente/Nodes in eine neue separate Gruppe platziert wird.



Wenn wir uns die unten gezeigte CSV-Datei ansehen, möchten wir als Ergebnis alle Header und Detail-Datensätze in einer eigenen Gruppe haben.

Head/Detail	OrderNo	ProdNo	Unitweight	Field5	Field6	Field7	Field8	Fi
string	string	string	string	string	string	string	string	st
H	111	G-889-11	9		Container ship			
D	111	B-192-409	1	2	232	Barley		
D	111	F-152-427	3	1	456	Corn		
D	111	Q-821-207	7	5	52	Coconut		
H	222	A-978-4	22		Air freight			
D	222	M-623-111	8	8	78	Oil		
D	222	L-524-201	2	3	669	Miscellaneous		

- Eine neue Gruppe mit dem ersten Element, in diesem Fall H, wird begonnen.
- Da das nächste Element (oder der Schlüssel) in der Sequenz unterschiedlich ist, d.h. D, wird eine zweite Gruppe namens D begonnen.
- Die nächsten beiden D-Elemente werden nun zur selben Gruppe D hinzugefügt, da sie vom selben Typ sind.
- Eine neue Gruppe H mit einem einzigen H-Element wird begonnen.
- Diese wird gefolgt von einer neuen D-Gruppe mit zwei D-Elementen.



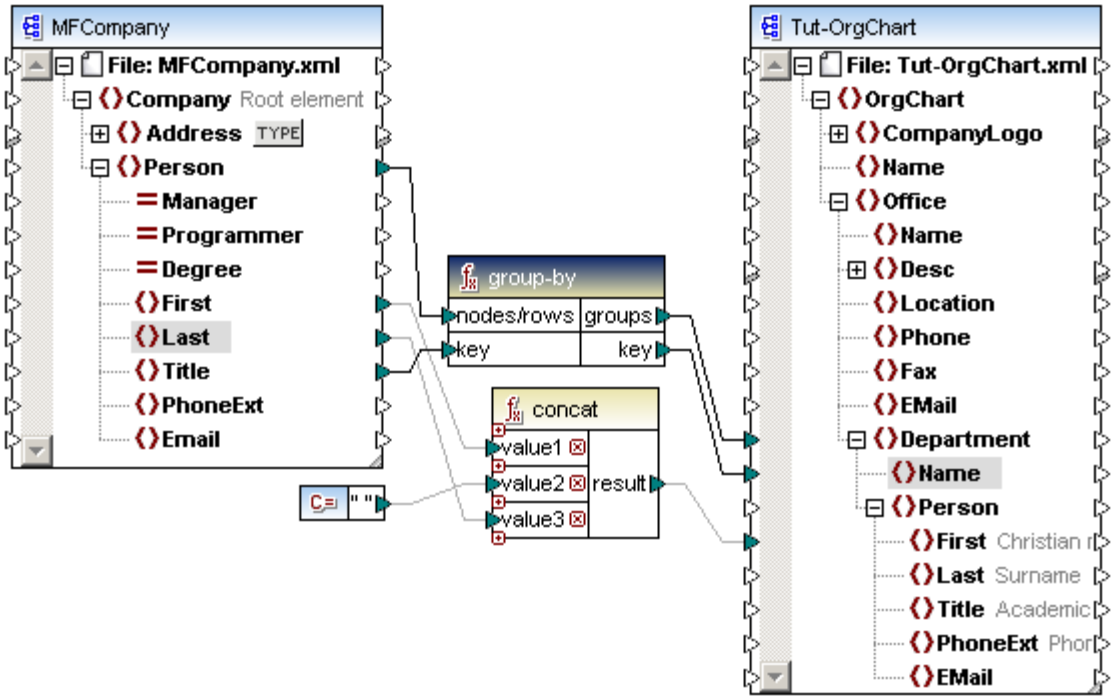
```
<Header>
  <RecordType>H</RecordType>
  <Detail>
    <RecordType>H</RecordType>
    <OrderNo>111</OrderNo>
    <ProductNo>G-889-11</ProductNo>
    <UnitWeight>9</UnitWeight>
  </Detail>
</Header>
<Header>
  <RecordType>D</RecordType>
  <Detail>
    <RecordType>D</RecordType>
    <OrderNo>111</OrderNo>
    <ProductNo>B-192-409</ProductNo>
    <UnitWeight>1</UnitWeight>
  </Detail>
  <Detail>
    <RecordType>D</RecordType>
    <OrderNo>111</OrderNo>
    <ProductNo>F-152-427</ProductNo>
    <UnitWeight>3</UnitWeight>
  </Detail>
  <Detail>
    <RecordType>D</RecordType>
    <OrderNo>111</OrderNo>
    <ProductNo>Q-821-207</ProductNo>
    <UnitWeight>7</UnitWeight>
  </Detail>
</Header>
```

Beachten Sie, dass die Funktion "group-adjacent" als Gruppierungsschlüssel den **Inhalt** des Node/Datenelements verwendet! Die Datensätze werden in der Zieldatei anhand des Inhalts des Felds "Head-Detail" nach Datentyp gruppiert.

group-by
Gruppiert die Input-Sequenz nach eindeutigen Schlüsseln und gibt die verschiedenen Gruppen zusammen mit ihren Schlüsseln aus. Ein Beispiel dafür sehen Sie unten:

- Der Schlüssel, der die einzelnen Gruppen der Quellkomponente definiert, ist das Datenelement **Title**. Anhand dieses Elements werden die Personen des Unternehmens gruppiert.
- Der Gruppenname wird in das Datenelement "Department/Name" der Zielkomponente platziert, wobei der miteinander verkettete Vor- und Nachname (First, Last) in den Sub-Eintrag (Person/First) platziert wird.

 © 2011 Altova GmbH



Beachten Sie, dass die Funktion "group-by" den **Inhalt** des Node/Datenelements als Gruppierungsschlüssel verwendet! Die Personen werden anhand des Inhalts des Felds "Title" gruppiert und anhand dessen auf das Datenelement "Department/Name" in der Zielkomponente gemappt.

Beachten Sie außerdem: Im inkludierten Beispiel sehen Sie einen **impliziten Filter** der Zeilen vom Quelldokument zum Zieldokument. Im Zieldokument hat jedes Department-Element nur die Person-Elemente, die mit dem Gruppierschlüssel (key) **übereinstimmen,** wenn die entsprechende Hierarchie on-the-fly von der "group-by"-Komponente erstellt wird.

Wenn Sie eine flache Hierarchie mit einem dynamischen Output-Dateinamen haben, die zum Teil anhand des key-Werts erstellt wird, so bleibt der implizite Filter weiterhin vorhanden. Dadurch müssen sie die 'groups'-Ausgabe möglicherweise nicht mit einem Element in der Zielkomponente verbinden.

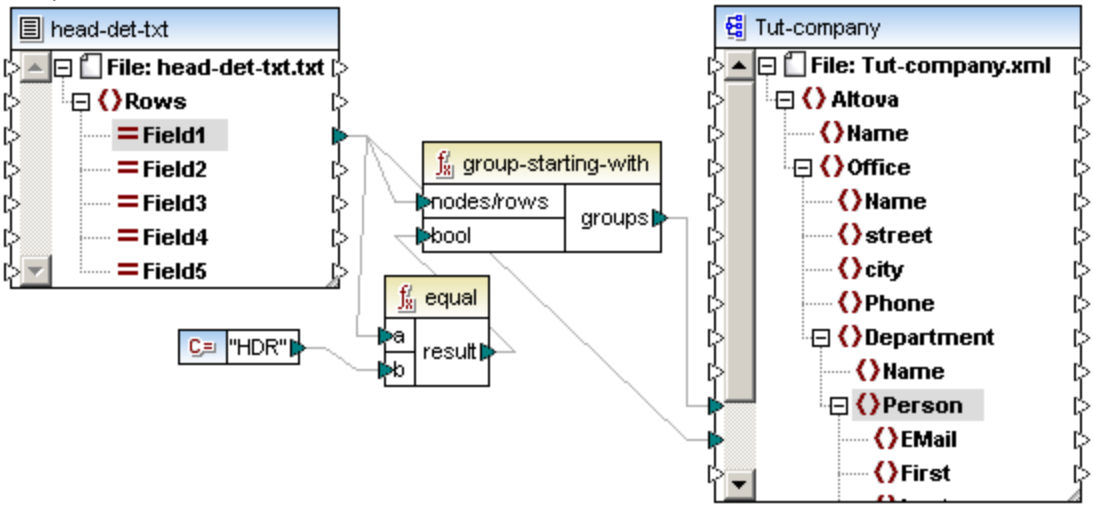
Wenn Sie auf die Schaltfläche "Ausgabe" klicken, sehen Sie das Ergebnis des Mappings.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <OrgChart xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart C:/DOCU
3    <Office>
4      <Department>
5        <Name>Office Manager</Name>
6        <Person>
7          <First>Vernon Callaby</First>
8          <First>Steve Meier</First>
9        </Person>
10     </Department>
11     <Department>
12       <Name>Accounts Receivable</Name>
13       <Person>
14         <First>Frank Further</First>
15         <First>Theo Bone</First>
16       </Person>
17     </Department>
```

group-starting-with
Diese Funktion gruppiert die Input-Sequenz nach dem bereitgestellten Datenelement, wenn es in den Quelldaten vorhanden ist. Zum Testen der Input-Daten wird eine Boolsche Funktion verwendet.

Field1	Field2	Field3	Field4	Field5
string	string	string	string	string
HDR	custid0001	ordid	0001	EUR
DTL	itemABC	qty0100	price	0001.100
TXT	please	deliver	ASAP	
DTL	itemxx2	qty0001	price	0010.000
DTL	itemDDD	qty0010	price	0010.500
HDR	custid0002	ordid	0001	EUR
DTL	itemABC	qty0100	price0001.100	
HDR	custid0003	ordid	0002	USD
DTL	itemDEF	qty0003	price	200.000

Die Funktion erstellt Gruppen auf Basis des **ersten** Datenelements einer Gruppe, in diesem Beispiel HDR.



Der **Wert** der Datenelemente/Nodes muss nicht identisch oder überhaupt vorhanden sein. Das

 © 2011 Altova GmbH

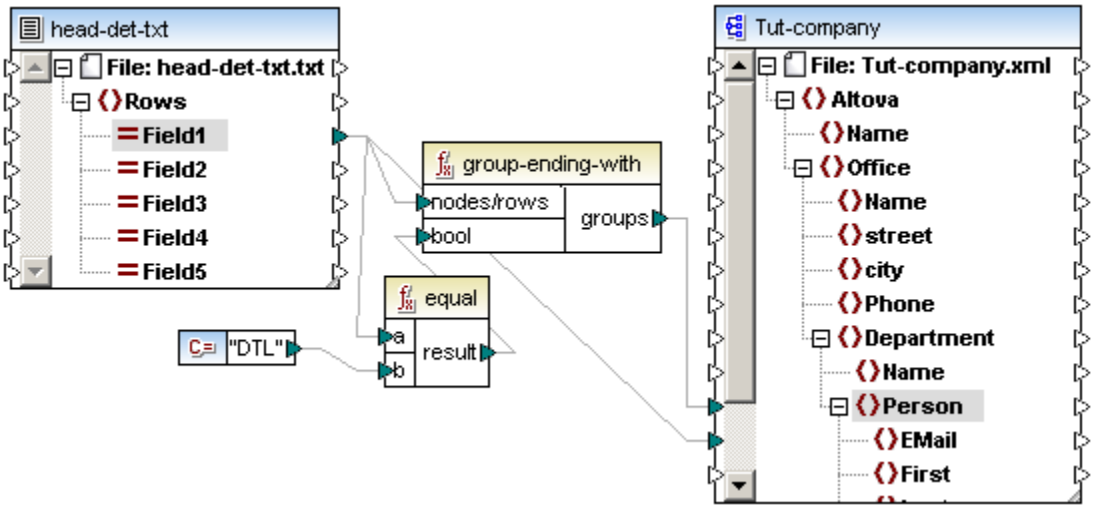
Node "**Pattern**" (Muster), d.h. die Namen der Nodes/Datenelemente müssen identisch sein, damit die Gruppierung erfolgen kann.

```
<Office>
  <Department>
    <Person>
      <EMail>HDR</EMail>
      <EMail>DTL</EMail>
      <EMail>TXT</EMail>
      <EMail>DTL</EMail>
      <EMail>DTL</EMail>
    </Person>
    <Person>
      <EMail>HDR</EMail>
      <EMail>DTL</EMail>
    </Person>
    <Person>
      <EMail>HDR</EMail>
      <EMail>DTL</EMail>
    </Person>
```

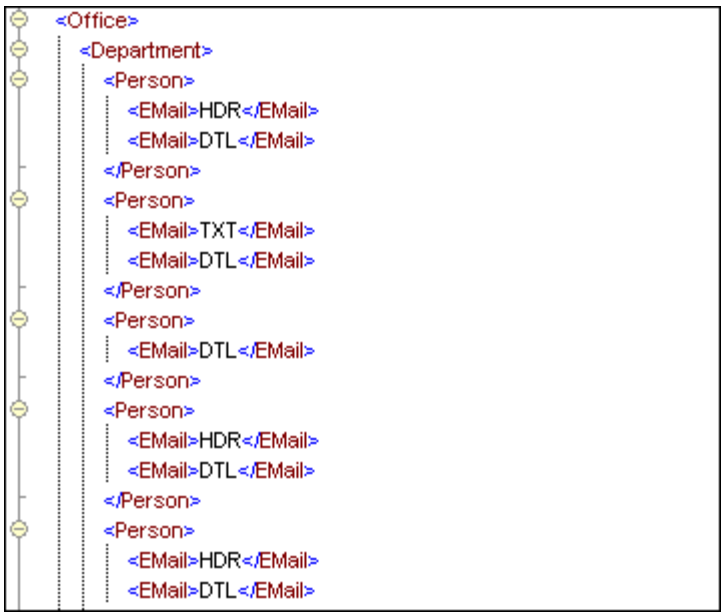
Im Ergebnis oben sehen Sie, dass für jedes HDR-Element eine neue Gruppe begonnen wurde.

group-ending-with
Diese Funktion ist die Ergänzung zur "group-starting-with" Funktion und beendet jede Gruppe der Input-Sequenz anhand des gelieferten Datenelements, falls dieses in der Quelldatei vorhanden ist. Zum Testen der Input-Daten wird eine Boolsche Funktion verwendet.

In diesem Beispiel wird dieselbe Quellkomponente wie im "group-starting-with"-Beispiel oben verwendet. Sie sehen das Ergebnis, wenn DTL als das "group-ending-with"-Datenelement verwendet wird.



In diesem Fall muss der **Wert** des Datenelements/Node nicht identisch oder überhaupt vorhanden sein. Das Node "**Pattern**" (Muster), d.h. die Namen der Nodes/Datenelemente müssen identisch sein, damit die Gruppierung erfolgen kann.



Im Ergebnis oben sehen Sie, dass, immer wenn DTL das letzte Element sein kann, eine neue Gruppe begonnen wurde.

set-empty
Dient zum Generieren einer leeren Sequenz für einen bestimmten Node. Wenn diese Funktion mit einem übergeordneten Node verbunden ist, werden auch alle untergeordneten Nodes auf leer gesetzt. Dadurch wird der Node praktisch aus der Zielkomponente entfernt.
Damit können Sie eines XBRL-Dokuments, die weiter oben in der XBRL-Komponente/ Taxonomie definiert wurden.

 © 2011 Altova GmbH

11.12 Verwendung von DTDs als "Schema-Komponenten"

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.

Hinzufügen von DTD Namespace URIs
Es gibt allerdings einige DTDs, z.B. DTDs, die von StyleVision verwendet werden, die xmlns*-Attributdeklarationen ohne Namespace URIs verwenden. Diese DTDs müssen erweitert werden, um in MapForce verwendet werden zu können:

- Die DTD muss durch Definition des xmlns-Attributs mit der Namespace URI geändert werden, siehe unten:

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

11.13 Überprüfung der Typkonvertierung

Ab der Version MapForce 2012 **SP2** führen die generierten Applikationen und die Vorschau (im Built-In-Ausführungsprozessor) entsprechend den XSLT2- und XQUERY-Vorgaben eine genauere Überprüfung des Ergebnisses auf Typkonvertierungsfehler durch.

Die Konvertierung der Werte von einem Typ in einen anderen kann nun zu einem Runtime-Fehler führen, wohingegen in früheren Versionen von MapForce ein Ergebnis erzeugt wurde.

Beispiel: Konvertierung von xs:string 'Hello', in xs:decimal

MapForce 2012 Versionen bis einschließlich **SP1**:

XSLT:	'Hello' (oder 'NaN' bei Übergabe an eine Funktion, die mit Zahlen arbeitet)
XSLT2:	Fehler: "ungültiger lexikalischer Wert"
Xquery:	Fehler: "ungültiger lexikalischer Wert"
Vorschau mit Built-In-Ausführungsprozessor:	0
C++ app:	0
C# app:	Fehler: "Werte können nicht konvertiert werden"
Java app:	Fehler: "Werte können nicht konvertiert werden"

MapForce 2012 **SP2**:

XSLT:	'Hello' (oder 'NaN' bei Übergabe an eine Funktion, die mit Zahlen arbeitet)
XSLT2:	Fehler: "ungültiger lexikalischer Wert"
Xquery:	Fehler: "ungültiger lexikalischer Wert"
Vorschau mit dem Built-In-Ausführungsprozessor:	**Fehler: "String-Wert 'Hello' konnte nicht in Dezimalwert konvertiert werden"**
C++ app:	**Fehler: "Werte können nicht konvertiert werden"**
C# app:	Fehler: "Werte können nicht konvertiert werden"
Java app:	Fehler: "Werte können nicht konvertiert werden"

Bei Auftreten von Typkonvertierungsfehlern überprüfen Sie, ob die Typen korrekt behandelt wurden. Verwenden Sie z.B. die Funktion lang:numeric(), um zu überprüfen, ob der Quellwert in eine Zahl konvertiert werden kann, und verwenden Sie anschließend eine if-else-Komponente, um einen anderen Wert zu übergeben, falls dies fehlschlägt (z.B. eine Konstante, die im Parameter "value-false" den Wert -1 enthält).

© 2011 Altova GmbH

11.14 Katalogdateien in MapForce

MapForce unterstützt einen Teil der OASIS XML-Kataloge. Mit Hilfe von Katalogen kann MapForce gebräuchliche Schemas (sowie Stylesheets und andere Dateien) aus lokalen Benutzerordnern abrufen, wodurch die Verarbeitungsgeschwindigkeit erhöht wird, Benutzer auch offline arbeiten können (d.h. ohne mit einem Netzwerk verbunden zu sein) und Dokumente leichter übertragbar werden (da die URIs nur in den Katalogdateien geändert werden müssen).

Der Mechanismus in MapForce funktioniert, wie unten beschrieben.

RootCatalog.xml
MapForce lädt eine Datei namens `RootCatalog.xml`(*Struktur siehe unten*), die eine Liste von Katalogdateien enthält, die durchsucht werden. Sie können jeweils im Element `nextCatalog` in `RootCatalog.xml` beliebig viele Katalogdateien definieren, die durchsucht werden sollen.

```
<?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"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
  <!-- Include all catalogs under common schemas folder on the first directory
level -->
  <nextCatalog spy:recurseFrom="%AltovaCommonFolder%/Schemas" catalog="
catalog.xml" spy:depth="1"/>
  <!-- Include all catalogs under common XBRL folder on the first directory
level -->
  <nextCatalog spy:recurseFrom="%AltovaCommonFolder%/XBRL" catalog="
catalog.xml" spy:depth="1"/>
</catalog>
```

Die in `RootCatalog.xml` inkludierten Katalogdateien werden durchsucht und die URIs werden entsprechend den Zuordnungen in den Katalogdateien aufgelöst. Beachten Sie, dass die Katalogdateien `catalog.xml` in den Ordnern `Schemas` und `XBRL` des durch die Variable `%AltovaCommonFolder%` definierten Ordners durchsucht werden. (Der Wert der `%AltovaCommonFolder%` Variablen wird in der Tabelle unten aufgelistet.)

Die Katalogdateien im Altova-Ordner "Common" mappen die vordefinierten Public und System Identifiers häufig verwendeter Schemas (wie z.B. SVG und WSDL) und XBRL-Taxonomien auf URIs, die auf lokal gespeicherte Kopien der entsprechenden Schemas verweisen. Diese Schemas werden bei der Installation von MapForce im Altova Ordner "Common" installiert. Achten Sie darauf, keine Zuordnungen doppelt zu vergeben, da dies Fehler verursachen könnte.

CoreCatalog.xml, CustomCatalog.xml und Catalog.xml
Beachten Sie in der `RootCatalog.xml` Liste oben, dass `CoreCatalog.xml` und `CustomCatalog.xml` als Lookup aufgelistet sind:

- `CoreCatalog.xml` enthält bestimmte Altova-spezifische Zuweisungen zum Auffinden von Schemas im Altova Ordner "Common".
- `CustomCatalog.xml` ist eine Skelettdatei, in der Sie ihre eigenen Zuweisungen erstellen können. Sie können für jedes gewünschte Schema Zuweisungen zu `CustomCatalog.xml` hinzufügen, doch dieses wird nicht von den Katalogdateien im Altova Common-Ordner angegeben. Sie können dies mit Hilfe der unterstützten

Elemente des OASIS-Katalogs (*siehe unten*) tun (*siehe unten*).

- Der Altova-Ordner "Common" enthält eine Reihe von `Catalog.xml` Dateien. Jede befindet sich in einem Ordner eines bestimmten Schemas oder einer XBRL-Taxonomie im Altova-Ordner "common" und jede weist URIs Public und/oder System Identifier zu, die auf lokal gespeicherte Kopien der entsprechenden Schemas verweisen.

Speicherpfad von Katalogdateien und Schemas
Die Dateien `RootCatalog.xml` und `CoreCatalog.xml` sind im MapForce Applikationsordner installiert. Die Datei `CustomCatalog.xml` befindet sich im Ordner `MyDocuments/Altova/MapForce`. Die Catalog.xml Dateien befinden sich in den jeweiligen Schemaordnern. Diese Schemaordner befinden sich im Ordner: `%AltovaCommonFolder%\Schemas` and `%AltovaCommonFolder%\XBRL`.

Shell-Umgebungsvariablen und Altova-Variablen
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:

`%AltovaCommonFolder%`	C:\Program Files\Altova\CommonMapForce
`%DesktopFolder%`	Vollständiger Pfad zum Desktop-Ordner für den aktuellen Benutzer.
`%ProgramMenuFolder%`	Vollständiger Pfad zum Programmmenü-Ordner für den aktuellen Benutzer.
`%StartMenuFolder%`	Vollständiger Pfad zum Startmenü-Ordner für den aktuellen Benutzer.
`%StartUpFolder%`	Vollständiger Pfad zum Startordner für den aktuellen Benutzer.
`%TemplateFolder%`	Vollständiger Pfad zum Vorlagen-Ordner für den aktuellen Benutzer.
`%AdminToolsFolder%`	Vollständiger Pfad zum Dateisystemverzeichnis, in dem die Verwaltungstools für den aktuellen Benutzer gespeichert sind.
`%AppDataFolder%`	Vollständiger Pfad zum Applikationsdatenordner für den aktuellen Benutzer.
`%CommonAppDataFolder%`	Vollständiger Pfad zum Dateiverzeichnis mit den Applikationsdaten für alle Benutzer.
`%FavoritesFolder%`	Vollständiger Pfad zum Ordner "Favoriten" für den aktuellen Benutzer.
`%PersonalFolder%`	Vollständiger Pfad zum persönlichen Ordner für den aktuellen Benutzer.
`%SendToFolder%`	Vollständiger Pfad zum Ordner "SendTo" für den aktuellen Benutzer.
`%FontsFolder%`	Vollständiger Pfad zum Systemschriftarten-Ordner.
`%ProgramFilesFolder%`	Vollständiger Pfad zum Programmdateiordner für den aktuellen Benutzer.
`%CommonFilesFolder%`	Vollständiger Pfad zum Ordner "Gemeinsame Dateien" für den aktuellen Benutzer.
`%WindowsFolder%`	Vollständiger Pfad zum Windows-Ordner für den aktuellen Benutzer.
`%SystemFolder%`	Vollständiger Pfad zum Systemordner für den aktuellen Benutzer.

 © 2011 Altova GmbH

%CommonAppDataFolder%	Vollständiger Pfad zum Dateiverzeichnis mit den Applikationsdaten für alle Benutzer.
%LocalAppDataFolder%	Vollständiger Pfad zum Dateisystemverzeichnis, das als Datenspeicher für lokale (nicht-Roaming) Applikationen dient.
%MyPicturesFolder%	Vollständiger Pfad zum Ordner "Meine Bilder".

Funktionsweise von Katalogen

Kataloge dienen normalerweise dazu, einen Aufruf von einer DTD auf eine lokale URI umzuleiten. Dies geschieht in der Katalogdatei durch Mappen von Public und System Identifiers auf die gewünschte lokale URI. Wenn also die DOCTYPE-Deklaration in einer XML-Datei gelesen wird, findet der Public oder System Identifier über das Katalogdatei-Mapping die gewünschte lokale Ressource.

Für gebräuchliche Schemas ist der PUBLIC Identifier normalerweise vordefiniert, sodass nur die URI in der Katalogdatei in die Adresse der richtigen lokalen Kopie geändert 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>
```

Dieses Dokument wird gelesen und der Katalog wird nach dem `PUBLIC` Identifier 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 auf die URI `schemas/svg/svg11.dtd` umgeleitet wird; (dieser Pfad ist relativ zur Katalogdatei). Diese lokale Datei wird dann als DTD verwendet. Wenn im Katalog kein passender Treffer gefunden wird, wird die URL im XML-Dokument verwendet (im Beispiel oben: `http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd`).

Die von MapForce unterstützte Katalogteilmenge

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. Bitte beachten Sie: Jedes Element kann das Attribut `xml:base` erhalten, welches dazu dient, die Basis-URI dieses Elements zu definieren.

- `<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"/>`

In Fällen, in denen es wie z.B. bei den meisten Stylesheets keinen Public Identifier gibt, kann der System Identifier einer URL direkt über das `system` Element zugewiesen werden. Außerdem kann eine URI auch über das `uri` Element einer anderen URI zugewiesen werden. Mit Hilfe der Elemente `rewriteURI` und `rewritsSystem` kann der Anfangsteil einer URI bzw. eines System Identifier neu geschrieben werden. Auf diese Art kann der Beginn eines Dateipfads ersetzt werden, wodurch ein anderes Verzeichnis als Ziel definiert werden kann. Nähere Informationen zu diesen Elementen finden Sie in der XML Catalogs Specification.

Dateierweiterungen und intelligente schemagemäße Bearbeitung

Mittels Katalogdateien können Sie auch festlegen, dass auf Dokumente mit einer bestimmten Dateierweiterung die intelligenten Bearbeitungsfunktionen von MapForce gemäß den Regeln des von Ihnen angegebenen Schemas angewendet werden. Wenn Sie z.B. eine benutzerdefinierte Dateierweiterung `.myhtml` für (HTML)-Dateien erstellen, die gemäß der HTML-DTD gültig sein sollen, können Sie die intelligente Editierung für Dateien mit dieser Erweiterung aktivieren, indem Sie das folgende Textelement als Child-Element des `<catalog>` Elements zu `CustomCatalog.xml` hinzufügen.

```
<spy:fileExtHelper ext="myhtml" uri="schemas/xhtml/xhtml1-transitional.dtd"/>
```

Damit werden die intelligenten Bearbeitungsfunktionen (Autokomplettierung, Eingabehilfen usw.) von `.myhtml`-Dateien in MapForce gemäß der XHTML 1.0 Transitional DTD aktiviert.

XML-Schema und Kataloge

XML-Schemainformationen sind in MapForce integriert und die Gültigkeit von XML-Schema-Dokumenten wird anhand dieser internen Informationen überprüft. Daher sollte in einem XML-Schema-Dokument kein Verweis auf ein Schema für XML-Schema vorgenommen werden.

Die Datei `Catalog.xml` im Ordner `%AltovaCommonFolder%\Schemas\schema` enthält Referenzen auf DTDs, die ältere XML-Schema-Spezifikationen implementieren. 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.

Weitere Informationen

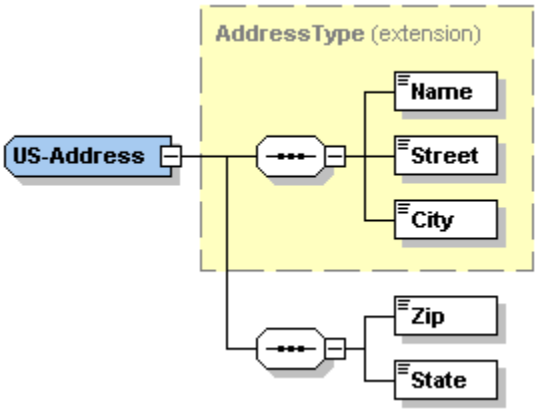
Nähere Informationen zu Katalogen finden Sie in der `XML-Katalogspezifikation`.

 © 2011 Altova GmbH

11.15 Derived XML-Schema-Typen - Mappen auf

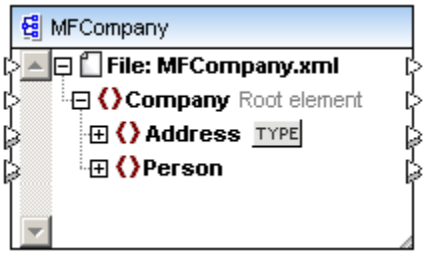
MapForce unterstützt das Mappen von/auf Derived Types eines complexType. Derived Types sind complexTypes eines XML-Schemas, die das Attribut xsi:type verwenden, um die spezifischen Derived Types zu identifizieren.

In der Abbildung unten sehen Sie die Definition des Derived Type "US-Address" in XMLSpy. Der Basistyp (oder der ursprüngliche complexType) ist in diesem Fall **AddressType**. Es wurden zwei zusätzliche Elemente hinzugefügt, um den Derived Type US-Address zu erstellen.

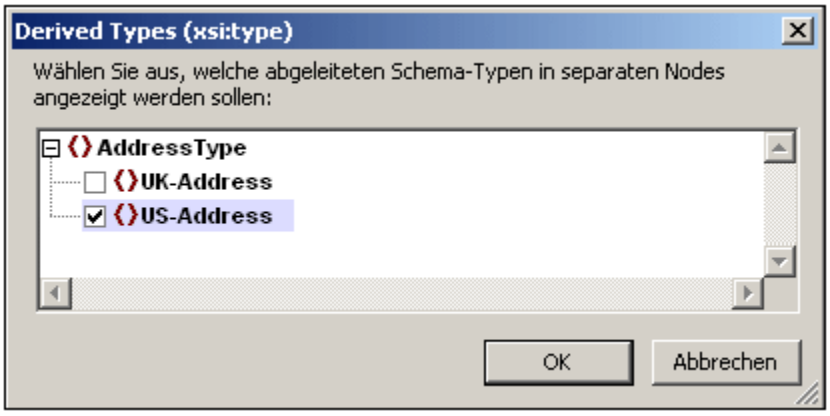


Mappen auf Derived Types in MapForce:

1. Fügen Sie das XML-Schema MFCompany.xsd ein, das im Ordner ...\Tutorial zur Verfügung steht, klicken Sie auf "Überspringen" und wählen Sie anschließend als Root-Element "Company" aus.



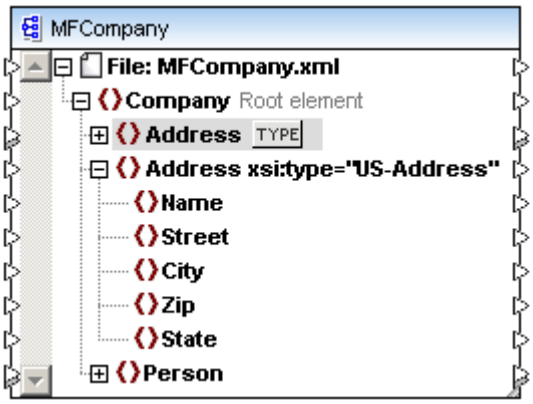
2. Klicken Sie auf die Schaltfläche TYPE rechts vom Element "Address", welche anzeigt, dass in der Schemakomponente Derived Types vorhanden sind.



3. Aktivieren Sie das Kontrollkästchen neben dem Derived Typen, den Sie verwenden

möchten, z.B. US-Address und bestätigen Sie mit OK.
Ein neues Element **Address xsi:type="US-Address"** wurde zur Komponente hinzugefügt.

4. Klicken Sie auf die Erweiterungsschaltfläche, um die mapbaren Datenelemente des Elements zu sehen.



5. Sie können nun direkt von/auf diese Datenelemente mappen.

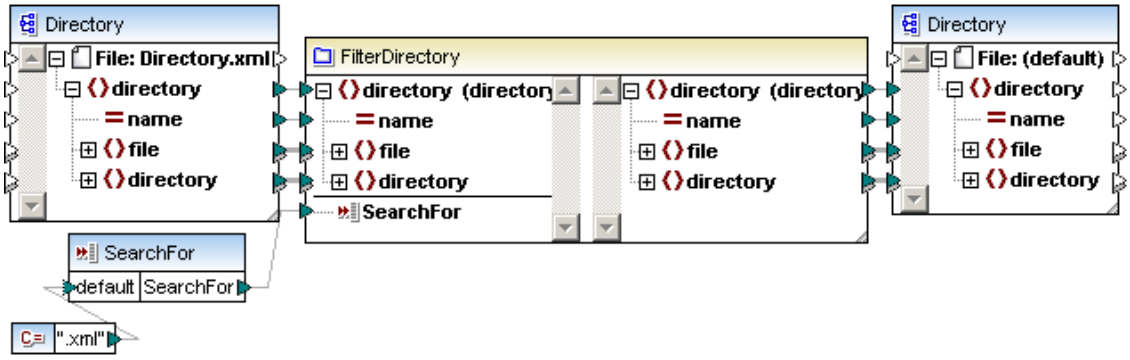
Bitte beachten Sie:
Sie können mehrere Derived Types inkludieren/einfügen. Wählen Sie diese dazu im Dialogfeld "Derived Types" aus. Jedes davon hat in der Komponente ein eigenes xsi:type-Element.

 © 2011 Altova GmbH

11.16 Rekursives benutzerdefiniertes Mapping

In diesem Abschnitt wird beschrieben, wie das Mapping **RecursiveDirectoryFilter.mfd** aus dem Ordner ...**\MapForceExamples** erstellt wurde und wie rekursive Mappings erstellt werden. Der Projektordner MapForceExamples enthält weitere Beispiele für rekursive Mappings.

In der Abbildung unten sehen Sie das fertige Mapping, das die rekursive benutzerdefinierte Funktion FilterDirectory enthält, deren Ziel es ist, in der Quelldatei eine Liste der .xml-Dateien zu filtern.

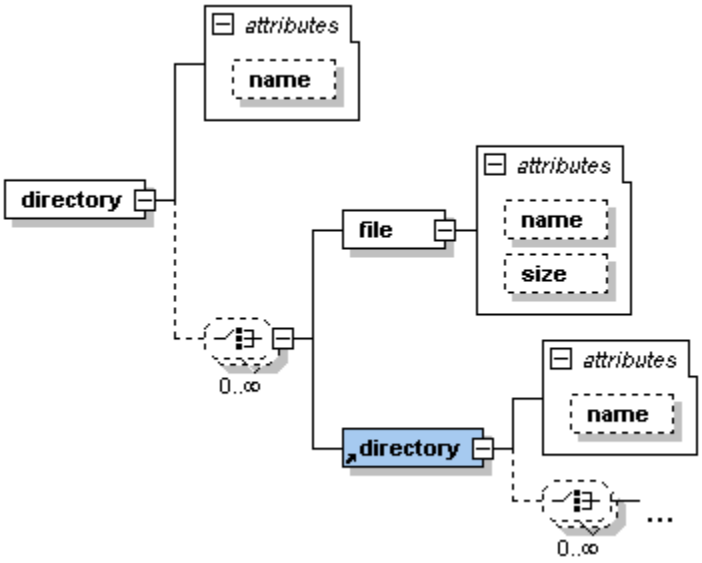


Die **Quelldatei,** die die Datei- und Verzeichnisdaten für dieses Mapping enthält, ist Directory.xml. Diese XML-Datei liefert, wie unten gezeigt, das Verzeichnis und die Dateidaten in hierarchischer Form.

```
24    <directory name="output">
25      <file name="examplesite1.css" size="3174"/>
26      <directory name="images">
27        <file name="blank.gif" size="88"/>
28        <file name="block_file.gif" size="13179"/>
29        <file name="block_schema.gif" size="9211"/>
30        <file name="nav_file.gif" size="60868"/>
31        <file name="nav_schema.gif" size="6002"/>
32      </directory>
33    </directory>
34  </directory>
35  <directory name="Import">
36    <file name="altova.mdb" size="266240"/>
37    <file name="Data_shape.mdb" size="225280"/>
38  </directory>
39  <directory name="IndustryStandards">
40    <directory name="News">
41      <file name="high-tide.jpg" size="10793"/>
42      <file name="Newsml-example.xml" size="5004"/>
43      <file name="nitf-example.xml" size="9327"/>
44    </directory>
```

Die von Directory.xml referenzierte XML-Schemadatei hat ein **rekursives** Element namens "directory", das beliebig viele Unterverzeichnisse und Dateien unterhalb des directory-Elements

zulässt.

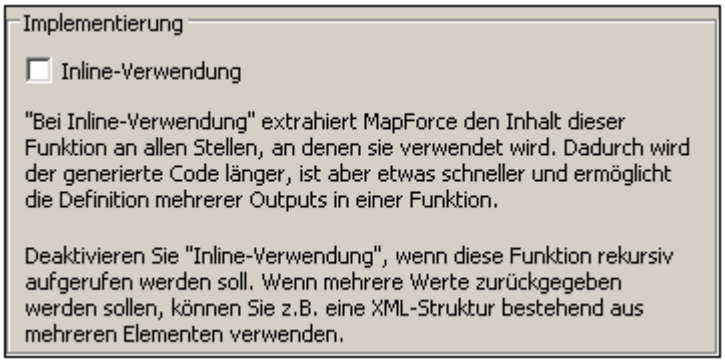


 © 2011 Altova GmbH

11.16.1 Definieren eines rekursiven benutzerdefinierten Mappings

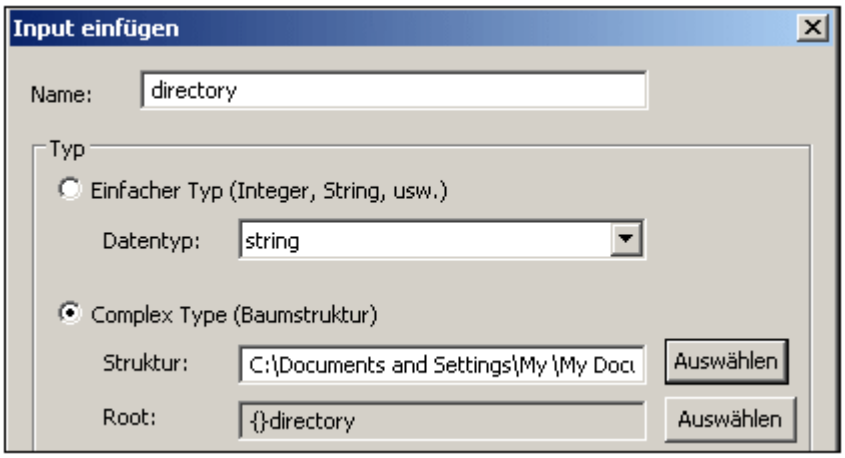
Aktionen im Hauptmappingfenster:

1. Wählen Sie den Befehl **Funktion | Benutzerdefinierte Funktion erstellen,** um mit der Erstellung der Funktion zu beginnen und geben Sie einen Namen ein, z.B. FilterDirectory.
2. Stellen Sie sicher, dass Sie das Kontrollkästchen **Inline-Verwendung** in der Gruppe **Implementierung** deaktiviert haben, um die Funktion rekursiv zu machen und klicken Sie anschließend auf OK.

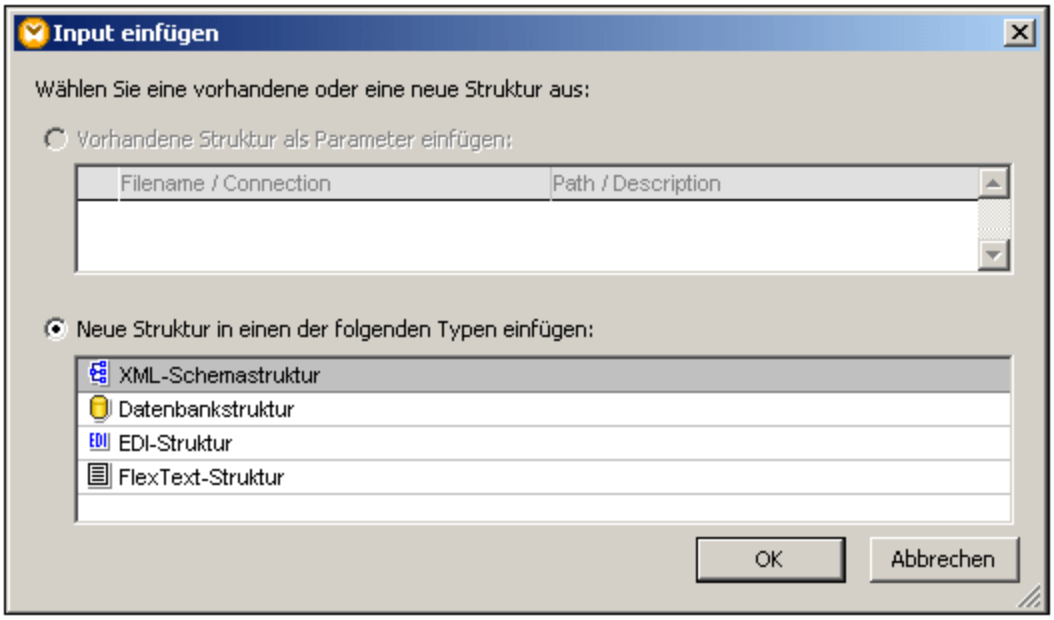


Sie befinden sich jetzt im Fenster **FilterDirectory,** wo Sie die benutzerdefinierte Funktion erstellen können.

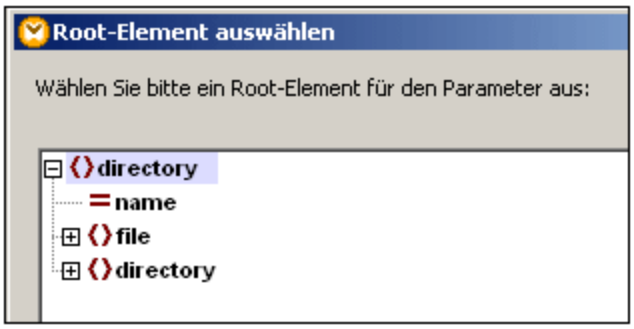
3. Wählen Sie **Funktion | Input-Komponente einfügen** um eine Input-Komponente einzufügen.
4. Geben Sie der Komponente einen Namen, z.B. directory und klicken Sie auf das Optionsfeld **Complex Type** (Baumstruktur).



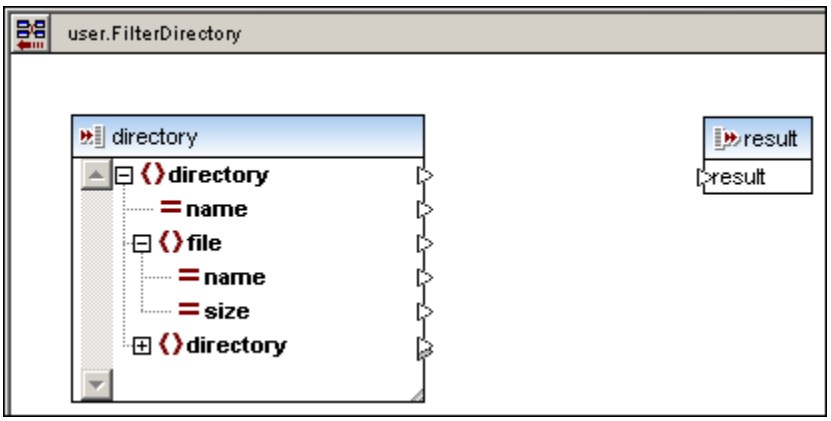
5. Klicken Sie auf die Schaltfläche **Auswählen** und wählen Sie die Datei directory.xsd im Verzeichnis ...\MapForceExamples.



6. Wählen Sie im die Datei **Directory.xsd** im Ordner ...\MapForceExamples aus und klicken Sie auf die Schaltfläche "Öffnen".
7. Klicken Sie nochmals auf OK, wenn Sie gebeten werden, das Root-Element auszuwählen, welches "directory" sein sollte.



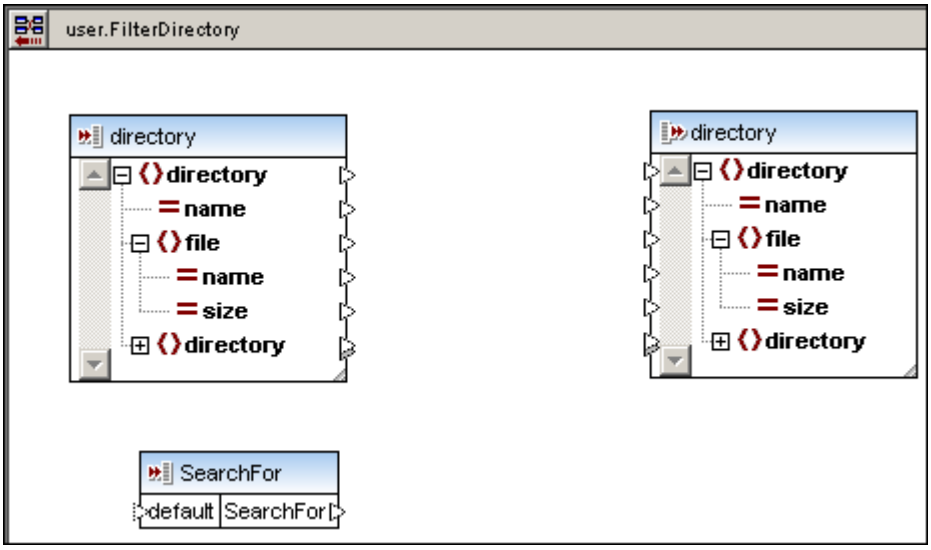
8. Klicken Sie wieder auf Ok, um den complex Input-Parameter einzufügen. Unten sehen Sie die benutzerdefinierte Funktion.



9. Löschen Sie die simple-Output-Komponente "result", da hier eine complex Output-Komponente eingefügt werden muss.

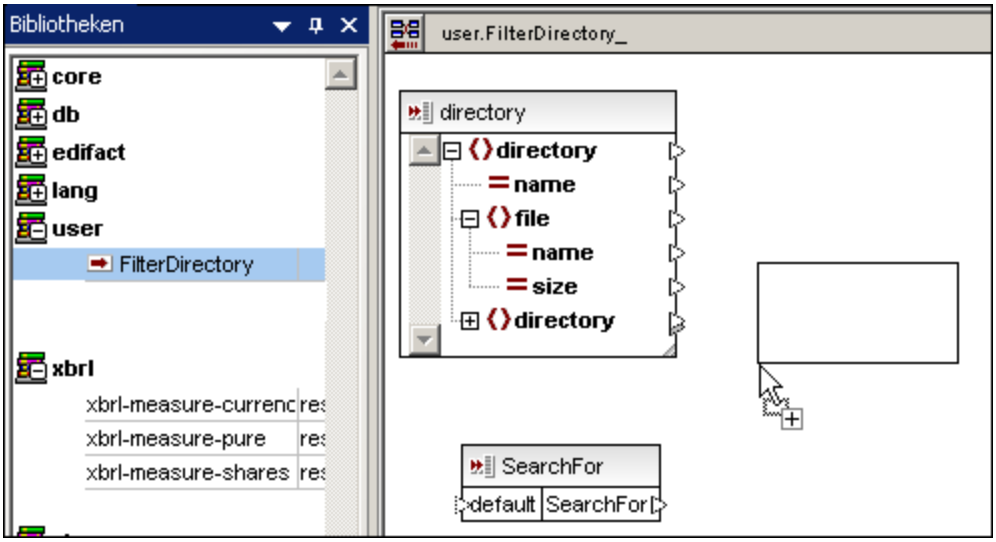
 © 2011 Altova GmbH

10. Wählen Sie **Funktion | Output-Komponente einfügen...,** um eine Output-Komponente einzufügen und verwenden Sie dieselbe Methode wie oben, um die Output-Komponente "directory" zu einem complexType zu machen.
    Sie haben jetzt zwei complex-Komponenten, eine Input- und eine Output-Komponente.
11. Wählen Sie **Funktion | Input-Komponente einfügen...** und fügen Sie eine Komponente vom Typ Simple type ein und geben Sie einen Namen ein, z.B. **SearchFor**. Deaktivieren Sie das Kontrollkästchen "Input ist erforderlich".



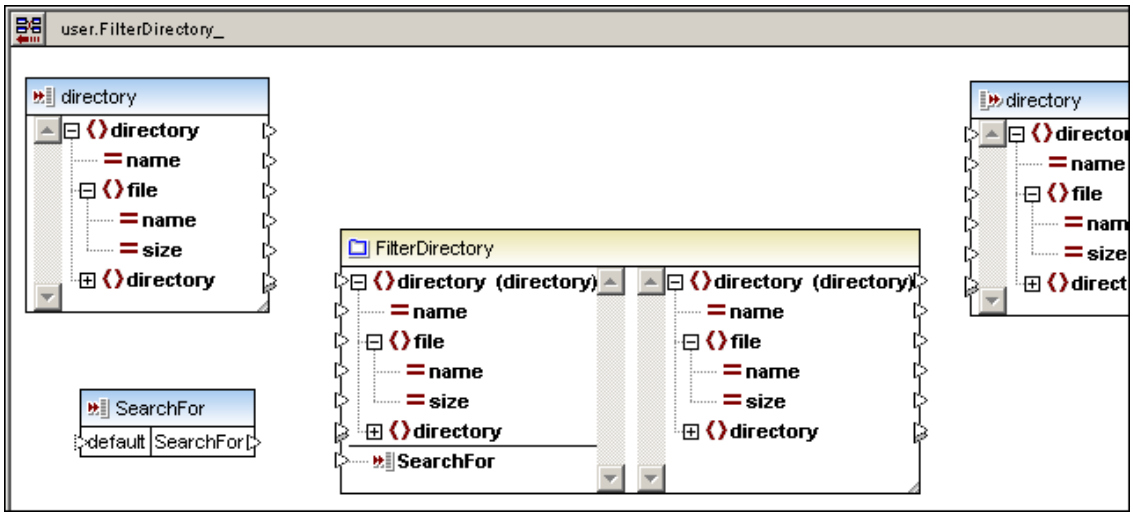
Einfügen der rekursiven benutzerdefinierten Funktion
Wir haben jetzt alle erforderlichen Input- und Output-Komponenten für die benutzerdefinierte Funktion definiert. Nun muss die "unfertige" Funktion in das aktuelle Fenster für die benutzerdefinierte Funktion eingefügt werden. (Sie können dies jedoch zu jedem beliebigen Zeitpunkt tun.)

1. Suchen Sie die Funktion FilterDirectory in Fenster **Bibliotheken** im Abschnitt **user**.
2. Klicken Sie auf FilterDirectory und ziehen Sie sie in das Fenster FilterDirectory, in dem Sie gerade gearbeitet haben.

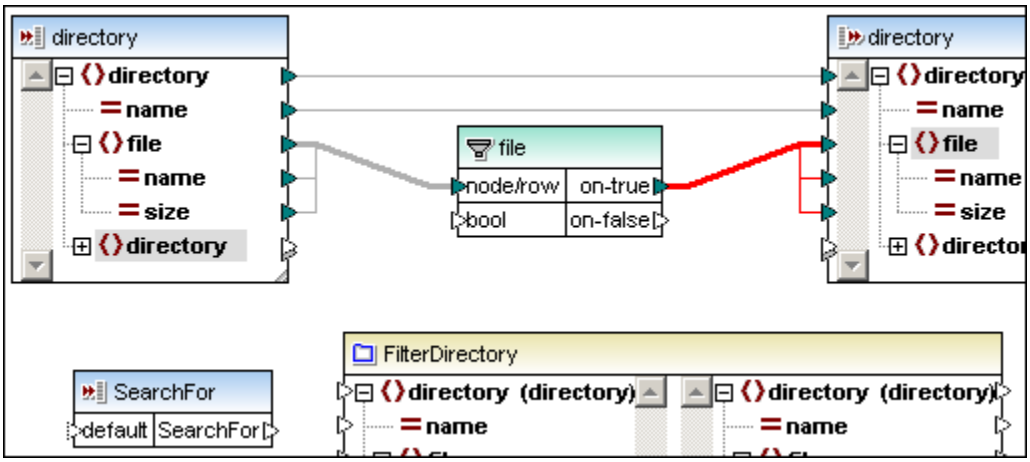


Java ausgewählt

Die benutzerdefinierte Funktion wird nun als rekursive Komponente im Fenster für die benutzerdefinierte Funktion angezeigt.

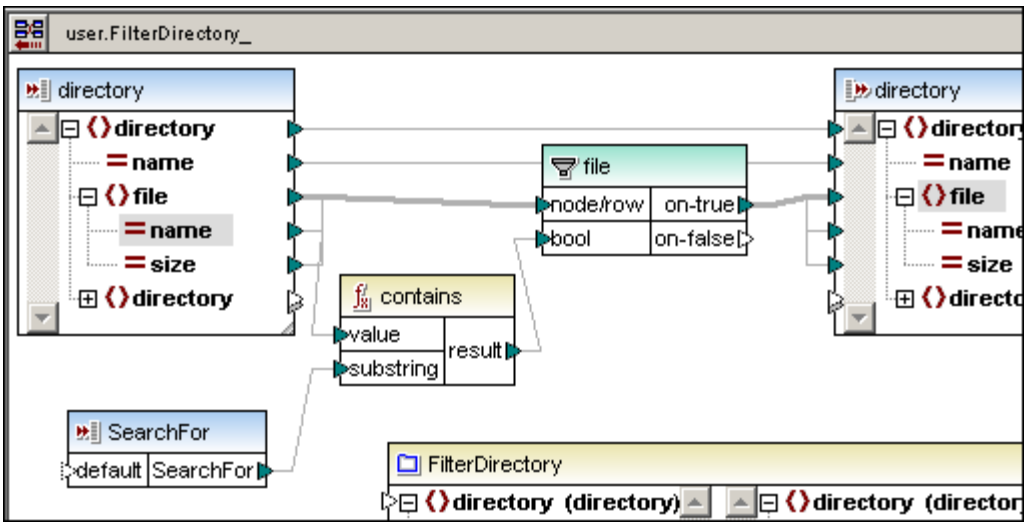


3. Verbinden Sie die Datenelemente **directory, name** und **file** der Input-Komponente mit den entsprechenden Datenelementen in der Output-Komponente.
4. Klicken Sie mit der rechten Maustaste auf den Konnektor zwischen den Datenelementen **file** und wählen Sie den Befehl "Filter einfügen", um eine Filterkomponente einzufügen.
5. Klicken Sie mit der rechten Maustaste auf den on-true-Konnektor und wählen Sie im Kontextmenü den Befehl **Alles kopieren**.
   Die Konnektoren ändern ihr Aussehen und werden als "Alles kopieren"-Konnektoren angezeigt.



6. Fügen Sie aus der Bibliothek **Core | String functions** eine Contains-Funktion ein.
7. Verbinden Sie **name** mit **value** und den Parameter **SearchFor** mit **substring**, anschließend **result** mit dem Bool-Element des Filters.

 © 2011 Altova GmbH



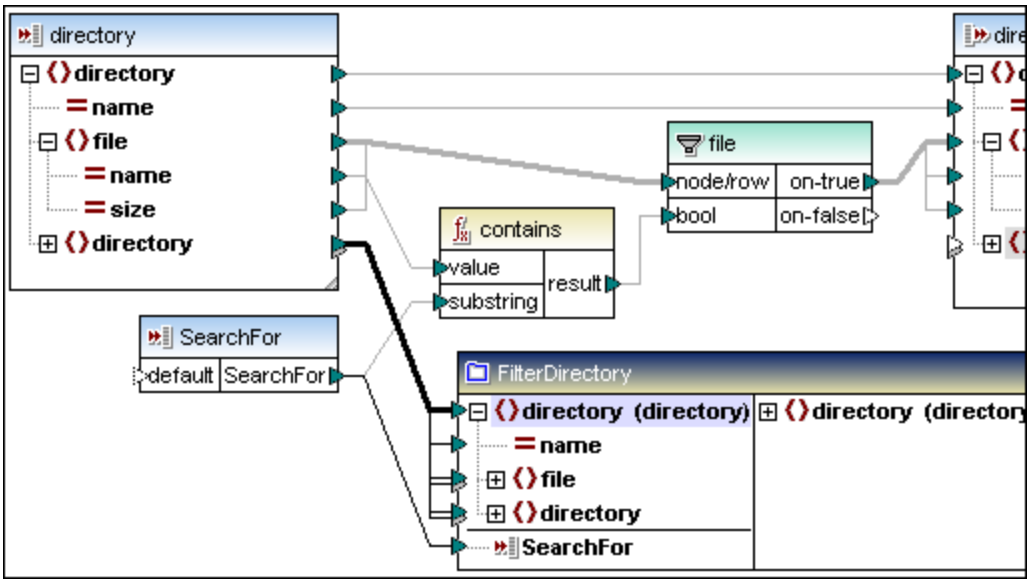
8. Verbinden Sie das Datenelement SearchFor der Input-Komponente mit dem Datenelement SearchFor der benutzerdefinierten Funktion.

Definieren der Rekursion

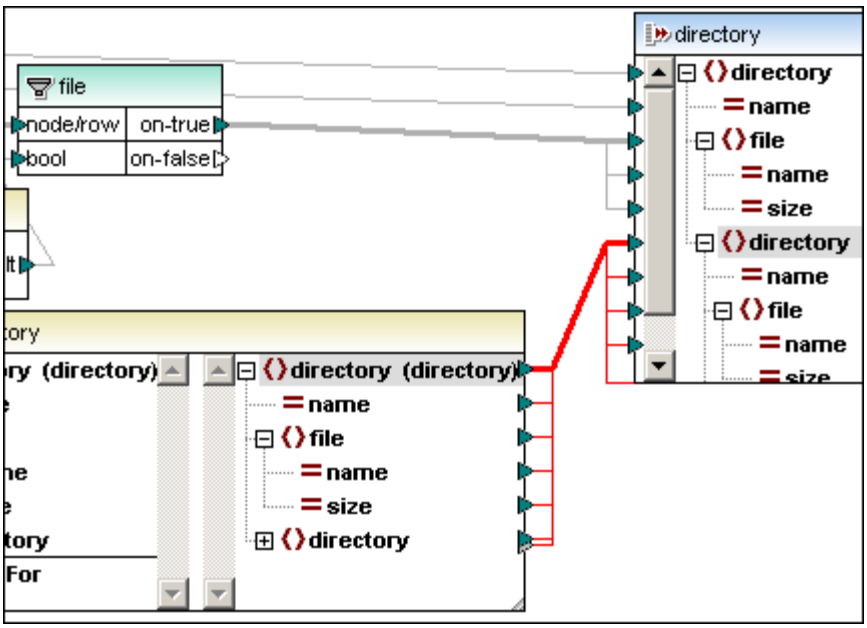
Zu diesem Zeitpunkt ist das Mapping einer einzigen Verzeichnisrekursionsebene fertig. Nun müssen wir nur noch definieren, wie ein Unterverzeichnis verarbeitet werden soll.

Stellen Sie sicher, dass in der Symbolleiste die Schaltfläche "Automatisch verbinden" aktiv ist:

1. Verbinden Sie das untere **directory** Element der Input-Komponente mit dem oberen **directory** Element der rekursiven benutzerdefinierten Funktion.



2. Verbinden Sie das obere Output directory-Element der benutzerdefinierten Funktion mit dem unteren directory-Element der Output-Komponente.
3. Klicken Sie mit der rechten Maustaste auf den Konnektor, wählen Sie im Kontextmenü den Befehl "Alles kopieren" und klicken Sie auf Ok, wenn Sie gefragt werden, ob Sie eine "Alles-kopieren"-Verbindung erstellen möchten.

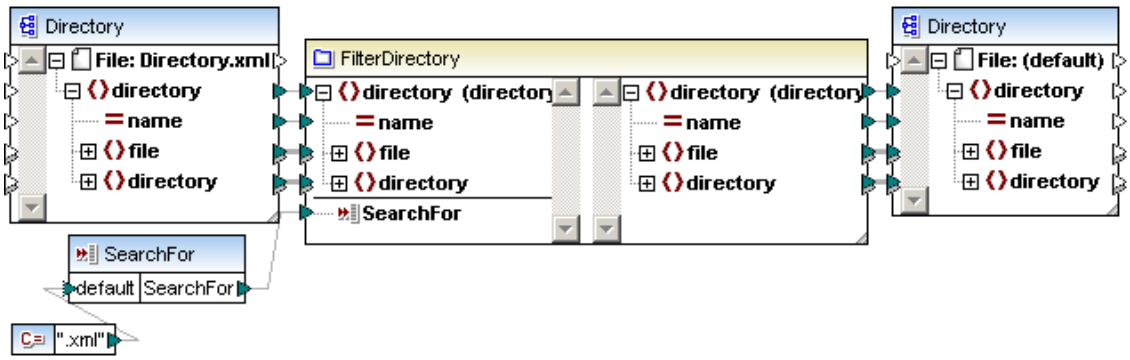


Damit ist die Definition der benutzerdefinierten Funktion in diesem Fenster abgeschlossen.

Klicken Sie auf die Schaltfläche "Zurück zum Hauptmapping-Fenster" , um das Mapping dort fertig definieren zu können.

Hauptmapping-Fenster

1. Ziehen Sie die Funktion FilterDirectory aus dem **user** Abschnitt des Fensters "Bibliotheken" in das **Hauptmapping.**
2. Wählen Sie den Befehl **Einfügen | XML-Schemadatei** um Directory.xsd einzufügen und wählen Sie Directory.xml als Instanzdatei.
3. Fügen Sie auf dieselbe Art Directory.xsd ein und wählen Sie **Überspringen** um die Output-Komponente zu erstellen.
4. Fügen Sie eine Konstantenkomponente ein, anschließend eine Input-Komponente, z.B. SearchFor.
5. Erstellen Sie die Verbindungen wie in der Abbildung unten gezeigt.
6. Wenn Sie auf beiden Seiten der benutzerdefinierten Komponente die Verbindungen auf der obersten Ebene erstellen, directory zu directory, klicken Sie mit der rechten Maustaste auf den Konnektor und wählen Sie im Kontextmenü den Befehl **Alles kopieren**.



7. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis des Mappings zu sehen.

 © 2011 Altova GmbH

Anmerkungen:
Wenn Sie in der Komponente "Directory" auf das unterste "directory" Datenelement doppelklicken, wird eine neue Rekursionsebene geöffnet, d.h. Sie sehen eine neue **directory | file | directory** Unterebene. Mit Hilfe des "Alles kopieren"-Konnektors werden automatisch alle vorhandenen Rekursionsebenen in der XML-Instanz verwendet und Sie müssen die Rekursionsebenen nicht mehr manuell erweitern.

Kapitel 12

Nullwerte / Nillable Werte

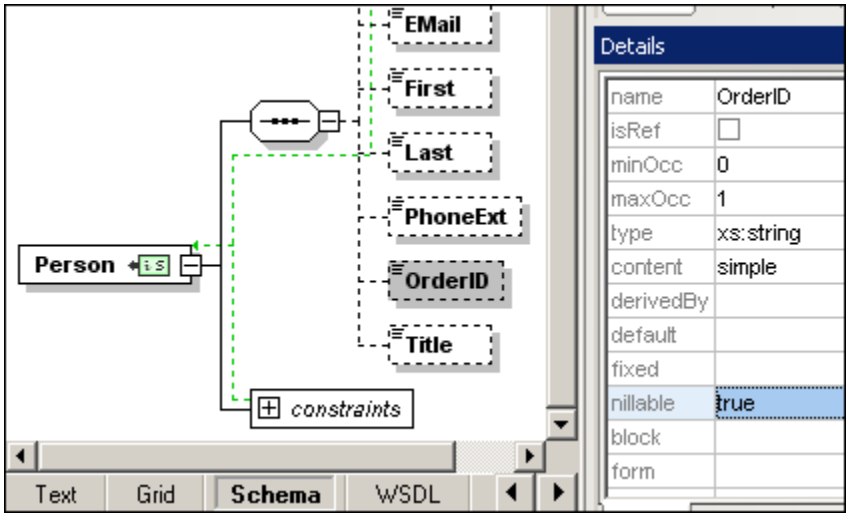
12 Nullwerte / Nillable Werte

Nach der XML-Schema-Spezifikation dürfen gültige Elemente auch leer sein, d.h. keinen Inhalt enthalten, (auch wenn ein Content Type leeren Inhalt nicht zulässt), wenn für das entsprechende Element im Schema das Attribut nillable="true" definiert ist.

Das bedeutet, dass im XML-**Instanzdokument** mit Hilfe eines **Attributs** xsi:nil="true" angegeben wird, dass das Element zwar vorhanden ist, dass der **Inhalt** dieses Elements aber Null ist.

Beachten Sie, dass dies nur für **Elementwerte**, nicht aber für Attributwerte gilt. Das Element mit xsi:nil="true" hat zwar vielleicht keinen Inhalt oder enthält keinen Text, kann aber weitere Attribute haben.

Das Attribut **nillable="true"** ist im XML-**Schema** definiert und kann sowohl in der Quell- als auch in der Zielkomponente vorkommen.



Das Attribut **xsi:nil="true"** ist in der XML-**Instanzdatei** definiert und kommt nur in der Instanzdatei vor.

```
14   <Person>
15     <PrimaryKey>2</PrimaryKey>
16     <ForeignKey>1</ForeignKey>
17     <EMail>biff@amail.com</EMail>
18     <First>biff</First>
19     <Last>bander</Last>
20     <PhoneExt>22</PhoneExt>
21     <OrderID xsi:nil="true"/>
22     <Title>IT services</Title>
23   </Person>
```

Das Attribut xsi:nil wird im grafischen MapForce-Mapping nicht explizit angezeigt, da es in den meisten Fällen automatisch verarbeitet wird. Ein auf Null gesetzter ("nilled") Node, also ein Node, für den das Attribut xsi:nil="true" gesetzt wurde, ist zwar vorhanden, hat aber keinen Inhalt.

© 2011 Altova GmbH

Nillable Elemente als Quellkomponente
Immer, wenn in einem Mapping Daten **aus** einem XML-Element ausgelesen werden, das auf Null gesetzt wurde, wird das Attribut xsi:nil automatisch überprüft. Wenn der Wert von xsi:nil "true" ist, so wird der Inhalt als nicht vorhanden behandelt.

Bei Erstellung eines **zielorientierten** Mappings von einem nillable Quellelement auf ein nillable Zielelement mit simpleType Inhalt (ein einziger Wert mit optionalen Attributen aber ohne Child-Elemente), bei dem xsi:nil für ein Quellelement definiert ist, wird das Attribut xsi:nil in das Zielelement eingefügt z.B. <OrderID xsi:nil="true"/>.

Bei Erstellung eines **Alles kopieren**-Mappings von einem nillable Quellelement auf ein nillable Zielelement, in dem xsi:nil für ein Quellelement definiert ist, wird das Attribut xsi:nil in das Zielelement z.B. eingefügt <OrderID xsi:nil="true"/>.

Verwenden von Funktionen
Wenn Sie die "exists"-Funktion mit einem auf Null gesetzten Quellelement verbinden, wird für alle Elemente "true" zurückgegeben, da der **Element**-Node ja tatsächlich existiert, auch wenn er keinen Inhalt enthält.

Um explizit zu überprüfen, ob das Attribut xsi:nil bei einem Quellelement auf "true" gesetzt ist, verwenden Sie die Funktion **is-xsi-nil**. Es gibt für auf Null gesetzte Elemente "true" zurück und für andere Nodes "false".

Die Verwendung von Funktionen, die einfache Werte erwarten (z.B. multiply, concat), für Elemente, für die xsi:nil definiert wurde, hat kein Ergebnis zur Folge, da kein Elementinhalt vorhanden ist und kein Wert extrahiert werden kann. Diese Funktionen verhalten sich, als wäre der Quell-Node nicht vorhanden.

Um einen auf Null gesetzten (nicht existierenden) Quellelementwert durch einen anderen Wert zu ersetzen, verwenden Sie die Funktion **substitute-missing**.

Auf Null gesetzte Elemente als Zielkomponente
Bei Erstellung eines **zielorientierten** Mappings von einem nillable Quellelement auf ein nillable Zielelement mit **simpleType** Inhalt (ein einziger Wert mit optionalen Attributen aber ohne Child-Elemente), bei dem xsi:nil für ein Quellelement definiert ist, wird das Attribut xsi:nil in das Zielelement eingefügt z.B. <OrderID xsi:nil="true"/>.

Wenn das Attribut xsi:nil="true" im XML-Quellelement **nicht** definiert wurde, so wird der Elementinhalt auf die übliche Weise auf das Zielelement gemappt.

Beim Mappen eines nillable Zielelements vom Typ **complex type** (mit Child-Elementen) wird das Attribut xsi:nil **nicht** automatisch geschrieben, da MapForce zum Zeitpunkt, zu dem es die Attribute des Elements schreibt, nicht weiß, ob Child-Elemente folgen. Definieren Sie eine "**Alles kopieren"-Verbindung**, um das Attribut xsi:nil aus dem Quellelement zu kopieren.

Beim Mappen einer **leeren Sequenz** auf ein Zielelement wird das Element gar nicht erstellt - unabhängig davon, ob es nillable ist oder nicht.

Verwendung von Funktionen
Um die Erstellung eines leeren Zieldatenelements mit xsi:nil auf "true" gesetzt zu erzwingen, verbinden Sie die Funktion **set-xsi-nil** direkt mit dem Zieldatenelement. Dies funktioniert bei Zieldatenelementen vom Typ "simpleType" und "complexType".

Verwenden Sie die Funktion **substitute-missing-with-xsi-nil** um xsi:nil in die Zielkomponente einzufügen, wenn in Ihrer Mapping-Quellkomponente kein Wert zur Verfügung steht. Dies kann

vorkommen, wenn der Quell-Node nicht vorhanden ist oder wenn bei einer Berechnung (z.B. einer Multiplikation) ein auf Null gesetzter Quell-Node beteiligt war, aufgrund dessen kein Ergebnis erzeugt wurde.

Diese Funktion funktioniert nur bei Nodes mit Inhalt vom Typ simpleType.

Um xsi:nil bei einem Element vom Typ **complexType** zu erstellen, gehen Sie folgendermaßen vor:

1. Erstellen Sie das Mapping wie gewohnt für den Fall, in dem der Ziel-Node Inhalt enthält.
2. Fügen Sie eine Filterkomponente in die Verbindung zum Ziel-Node ein und definieren Sie die Bedingung, unter der Inhalt vorhanden ist.
3. Verwenden Sie die Funktion "Duplikat erzeugen" um Ihren Ziel-Node zu kopieren.
4. Verbinden Sie die Funktion "set-xsi-nil" über eine weitere Filterkomponente mit dem kopierten Ziel-Node.
5. Fügen Sie eine "logical-not"-Funktion ein, um die Bedingung zu negieren und verbinden Sie sie mit dem Filter zwischen "set-xsi-nil" und dem kopierten Ziel-Node.

© 2011 Altova GmbH

Kapitel 13

Datenbanken und MapForce

13 Datenbanken und MapForce

Altova Website: Datenbank-Mapping

MapForce bietet umfangreiche Unterstützung für Datenbank-Mappings, u.a. auch Mappen von Datenbankdaten und XML-Dokumenten Flat Files, und sogar anderen Datenbankformaten.

MapForce berücksichtigt Einschränkungen durch Primär- und Sekundärschlüssel und erzeugt auch Transaktionen, um die Datenintegrität sicherzustellen.

Bitte beachten Sie:
Für einen Zugriff auf eine Datenbank müssen Sie eine der folgenden Optionen verwenden: Java, C#, C++ oder BUILTIN (den Built-In-Ausführungsprozessor). **XQuery** -Code kann nur für XML-Datenquellen und -ziele generiert werden.

Derzeit werden die folgenden Datenbanken (und Verbindungsarten) unterstützt:

- Microsoft Access; Version 2003 / 2007
- Microsoft SQL Server Versionen 2000, 2005 und 2008
- Oracle Versionen 9i, 10g und 11g
- MySQL 4.x und 5.x
- PostgreSQL 8.0, 8.1, 8.2, 8.3
- Sybase 12
- IBM DB2 Version 8.x und 9
- IBM DB2 für i 5.4
- IBM DB2 für i 6.1

Bitte beachten Sie:
MapForce bietet vollständige Unterstützung für die oben aufgelisteten Datenbanken. Altova ist bestrebt, auch Unterstützung für andere ODBC/ADO-Datenbanken zu gewährleisten, doch wurde die erfolgreiche Herstellung der Verbindung und Verarbeitung der Daten nur bei den aufgelisteten Datenbanken getestet.

Stellen Sie bei Installation der **64-Bit** Version von **MapForce** sicher, dass Sie Zugriff auf die 64-Bit-Datenbanktreiber haben, die für die jeweilige Datenbank, zu der Sie eine Verbindung hestellen möchten, benötigt werden.

Anmerkung: Für eine ADO-Verbindung zu MS SQL Server über den Treiber SQL Server Native Client 10.0 müssen die folgenden Eigenschaftswerte auf dem Register *All* des Dialogfelds "Data Link Properties" definiert sein: (i) Setzen Sie den Eigenschaftswert von Integrated Security auf ein Leerzeichen, (ii) Setzen Sie den Eigenschaftswert von Persist Security Info auf `true`.

© 2011 Altova GmbH

13.1 Tutorial: Mappen von XML-Daten auf Datenbanken

In diesem Tutorial lernen Sie Folgendes:

- Einrichten eines Mappings von XML auf eine Datenbank
- Einfügen von Daten in Datenbanktabellen
- Aktualisieren
- Löschen
- Ignorieren
- Generieren von Datenbank Ausgabewerten

Beispieldateien für das Tutorial
In diesem Abschnitt verwendete Dateien:

Altova_Hierarchical.xsd	die Datei mit dem hierarchischen Schema mit Identity Constraints
Altova-cmpy.xml	die Datei mit den Altova-XML-Firmendaten
Altova.mdb	die Altova MS-Access-Datenbankdatei, die als Zieldatenbank verwendet wird

Alle diese Beispieldateien liegen im Ordner **...\MapForceExamples**. Bitte beachten Sie: In diesem Abschnitt werden wir viel mit der Datenbank **Altova.mdb** arbeiten, um die Funktionen von MapForce im Zusammenhang mit der Verwendung von Datenbanken als Ziel zu erläutern. Legen Sie bitte eine Sicherungskopie der Datei an, bevor Sie eines der hier beschriebenen Beispiele ausprobieren. Um dieselben Ergebnisse wie in den Tutorial-Beispielen zu erhalten, sollten Sie jeden Abschnitt von Grund auf neu beginnen, ohne, dass Sie an der Datenbank vorher Änderungen vorgenommen haben.

MapForce kann Daten auch von passwortgeschützten Access-Dateien mappen, wenn Sie über die Option **Beliebige ODBC** aus dem Dialogfeld "Quelldatenbank auswählen" hinzugefügt werden. Die Benutzer- und Passworteinstellungen können im Assistenten eingegeben werden.

13.1.1 Datenbankaktion: Einfügen

In diesem Abschnitt des Tutorials wird erläutert, wie man Daten in eine Datenbank einfügt. Anhand der Quelldaten aus einer XML-Datei und des entsprechenden XML-Schemas werden SQL-Scripts generiert, die die entsprechenden Datenbankaktionen ausführen.

Ziel

In diesem Abschnitt des Tutorials lernen Sie, wie man mit MapForce Daten in eine Datenbank einfügt. Insbesondere lernen Sie:

- wie man ein neues Büroorganigramm (orgchart) zur Tabelle Altova in Ihrer Beispieldatenbank hinzufügt
- wie man damit in Zusammenhang stehende office-Tabellen zum neuen orgchart-Datensatz hinzufügt.

In diesem Abschnitt verwendete Befehle

XML Schema/Datei einfügen: Klicken Sie auf diese Schaltfläche, um das Windows-Standarddialogfeld "Öffnen" aufzurufen und die Datei aus Ihrem Dateisystem auszuwählen.

Datenbank einfügen: Dieser Befehl befindet sich in der Symbolleiste "Komponente einfügen" und im Menü **Einfügen.** Klicken Sie auf diesen Befehl, um ein Dialogfeld **Datenbank auswählen** zu öffnen, in dem Sie eine Datenbankverbindung definieren und auswählen können, welche Datenbanktabellen in das MapForce Mapping inkludiert werden sollen.

Datenbankaktionen: Dieser Befehl befindet sich im Menü **Komponente** einer Datenbankkomponente und wird in der Datenbankkomponente in Form einer Schaltfläche rechts vom Tabellennamen angezeigt. Klicken Sie auf diesen Befehl um das Dialogfeld **Datenbankaktionen** aufzurufen, in dem Sie definieren können, welche Aktionen für die einzelnen Datensätze ausgeführt werden sollen.

SQL-Script ausführen: Dieser Befehl befindet sich in der Symbolleiste "Ausgabevorschau" und im Menü "Ausgabe". Klicken Sie auf diesen Befehl, um das im Ausgabefenster generierte SQL-Script auszuführen.

Ausgabedatei neu generieren: Dieser Befehl befindet sich in der Symbolleiste "Ausgabevorschau" und im Menü "Ausgabe". Klicken Sie auf diesen Befehl, um das SQL-Script zurückzusetzen, damit es ein zweites Mal ausgeführt werden kann.

Einfügen von Daten in eine Datenbanktabelle

Im ersten Beispiel dieses Abschnitts wird die einfache Aufgabe erklärt, wie man zu der Tabelle Altova ein neues Organigramm **hinzufügt**. Für jeden Datensatz wird ein neuer Primärschlüssel generiert.

Primärschlüsseleinstellungen
Die Primärschlüsseleinstellungen für die Einfügeaktion werden über die Auswahllisten rechts vom jeweiligen Feld definiert.

 © 2011 Altova GmbH

Es stehen die folgenden Optionen zur Auswahl:

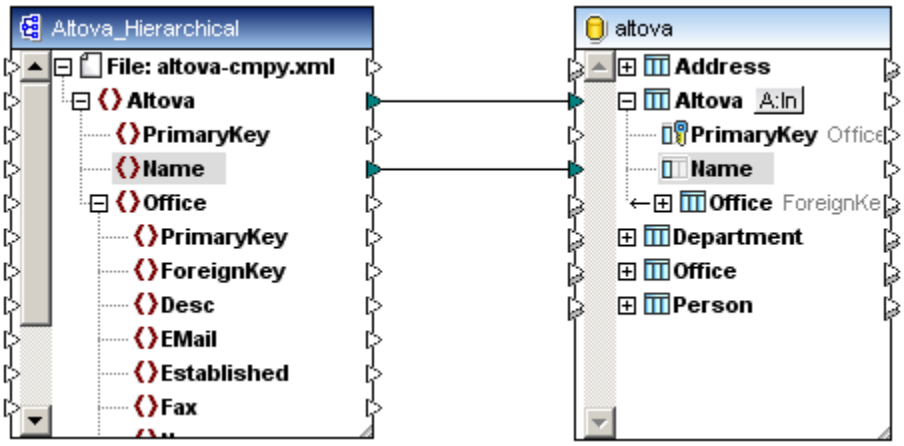
- Mit der Option **gemappter Wert** können Quelldaten direkt auf das Datenbankfeld gemappt werden. Dies ist die Standardeinstellung für alle Datenbankfelder.
- Um die Schlüsselwerte auf Basis der bestehenden Schlüssel in der Datenbank zu generieren, wählen Sie "**Max() + 1**", damit neue Datensätze automatisch an bestehende angehängt werden.
- Verwenden Sie die Einstellung **DB-generiert,** wenn die Datenbank die Schlüsselwerte generiert bzw. mit Hilfe der **Identity-Funktion** generiert.

So fügen Sie Werte aus einer XML-Datei in eine Datenbanktabelle ein:

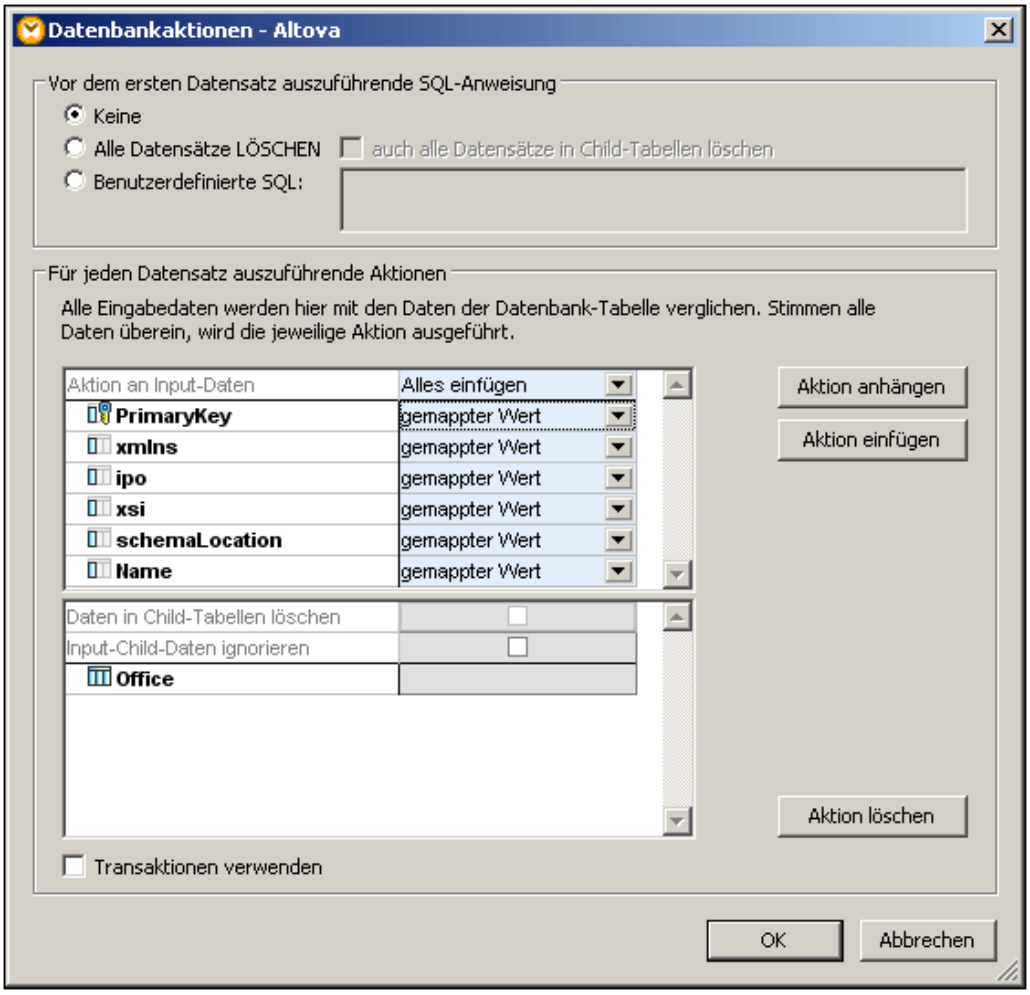
1. Öffnen Sie ein neues Mapping-Fenster und fügen Sie das Schema **Altova_Hierarchical.xsd** ein (und weisen Sie ihm **altova-cmpy.xml** als XML-Eingabeinstanz zu). Fügen Sie die Datenbank altova.mdb, wie im Abschnitt Einrichten von XML-auf-Datenbank Mappings beschrieben, in das Mapping-Fenster ein.
2. Erzeugen Sie die folgenden Mappings:

   Altova auf Altova, und
   Name auf Name

   Bitte beachten Sie: Wenn alle Altova-, Office- usw. Datenelemente automatisch gemappt werden, ist die Option "Idente Sub-Einträge automatisch verbinden" aktiv. Wählen Sie den Befehl **Bearbeiten | Rückgängig** und anschließend die Menüoption **Verbindung | Idente Sub-Einträge automatisch verbinden,** um die Option zu deaktivieren.

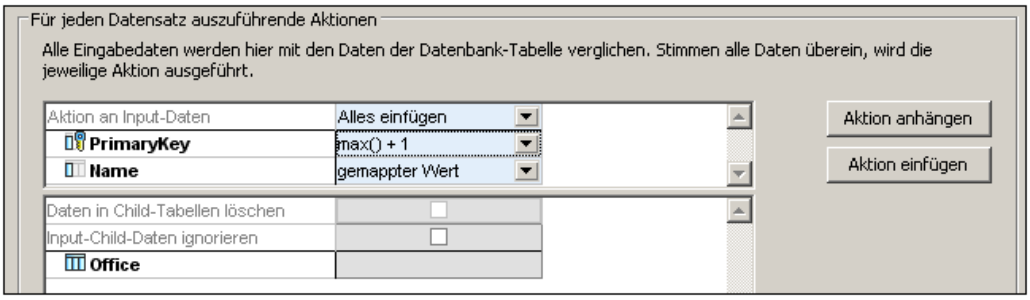


4. Das Symbol rechts von der Tabelle "Altova" (In der Abbildung oben **A:In**) zeigt die derzeit für diese Tabelle definierte Tabellenaktion an, d.h. Aktion:Einfügen (In = Insert). Wenn Sie auf das Symbol klicken, wird das Dialogfeld **Datenbankaktionen** geöffnet, in dem verschiedenen Tabellenaktionen definiert werden können.
5. Klicken Sie auf das Tabellenaktionssymbol A:In neben dem Tabelleneintrag **Altova**. In diesem Dialogfeld ist im Moment nur eine Tabellen-Aktionsspalte definiert, nämlich **Alles einfügen.**



Die Tabellenaktion "Alle einfügen", fügt Datensätze mit allen **gemappten Feldern** der aktuellen Tabelle in die Datenbank ein. Wir müssen nun definieren, wie der neue Primärschlüssel (PrimaryKey) für diese Aktion generiert wird.

5. Klicken Sie in die Auswahlliste rechts vom Feld **PrimaryKey** und wählen Sie **max() + 1.**



6. Bestätigen Sie die aktuellen Einstellungen mit OK.
7. Klicken Sie auf das Ausgaberegister am unteren Rand des Mapping-Fensters um das SQL Script zu sehen, das von diesem Mapping erzeugt wird.

 © 2011 Altova GmbH

```
/*
The following SQL statements are only for preview and may not be executed in another SQL query tool!
To execute these statements use function "Run SQL-script" from menu "Output".
Connect to database using the following connection-string:
Provider=Microsoft.Jet.OLEDB.4.0; Data Source=C:\Documents and Settings\...\My Documents\
Altova\MapForce2012\MapForceExamples\altova.mdb;
*/

SELECT IIF(MAX([Altova].[PrimaryKey]) IS NULL,0,MAX([Altova].[PrimaryKey]))+1 AS [PrimaryKey]
FROM [Altova]
-->>> %PrimaryKey1%

INSERT INTO [Altova] ([Name], [PrimaryKey]) VALUES ('Microtech OrgChart', '%PrimaryKey1%')
```

Das Script führt das Mapping auf die Zieldatenbank aus und berücksichtigt dabei die definierten Tabellenaktionen. Sie können SQL Scripts über das Ausgabefenster erneut ausführen, indem Sie auf die Schaltfläche **Ausgabedatei neu generieren** klicken.

7. Klicken Sie in der Funktionsleiste auf das Symbol **SQL-Script ausführen**, um das Script auszuführen und die Tabellendaten in die Datenbank einzufügen. Nach erfolgreichem Abschluss des Vorgangs wird im Fenster "Meldungen" eine Bestätigungsmeldung angezeigt. Bestätigen Sie mit OK.
8. Öffnen Sie die Altova-Datenbank in DatabaseSpy oder Access, um zu sehen, wie sich diese Aktion ausgewirkt hat.

Altova : Table

	PrimaryKey	Name
⊞	1	Organization Chart
⊞	2	Microtech OrgChart
▶		

Der Altova-Tabelle wurde ein neuer Microtech OrgChart-Datensatz mit dem neuen PrimaryKey 2 hinzugefügt. Die Daten für diesen Datensatz stammen aus der XML-Eingabeinstanz.

9. Wechseln Sie wieder zurück zu MapForce.

Im Fenster "Meldungen" sehen Sie nun eine Aufzeichnung darüber, was bei der Verarbeitung des SQL-Scripts passiert ist.

```
SELECT IIF(MAX([Altova].[PrimaryKey]) IS NULL,0,MAX([Altova].[PrimaryKey]))+1 AS [PrimaryKey]
FROM [Altova]
-->>> OK. One or more rows.

INSERT INTO [Altova] ([Name], [PrimaryKey]) VALUES ('Microtech OrgChart', 2)
-->>> OK. 1 row(s).
```

Einfügen von Tabellen und damit in Zusammenhang stehenden untergeordneten Tabellen

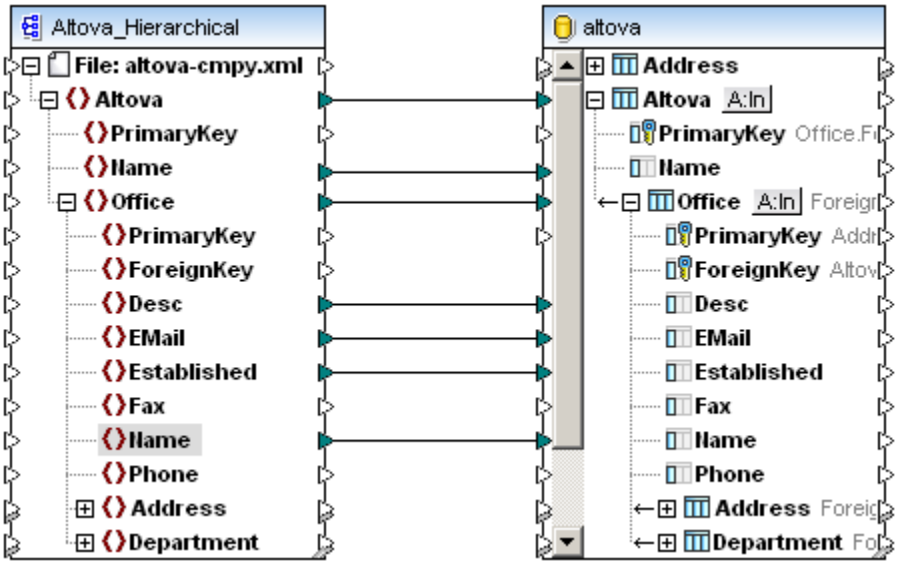
Als Ausgangsbasis für dieses Beispiel wird das vorhergehende Beispiel verwendet. Es wird durch Einfügen von untergeordneten Office-Tabellen in die übergeordnete Altova-Tabelle erweitert.

Tabellenbeziehungen werden nur dann automatisch generiert, wenn Mappings zwischen untergeordneten Tabellen einer "Root"-Tabelle erstellt werden. In diesem Fall werden Mappings zwischen den Office-Feldern, die sich direkt unterhalb der Altova-Parent (oder "Root")-Tabelle

befinden, erzeugt.

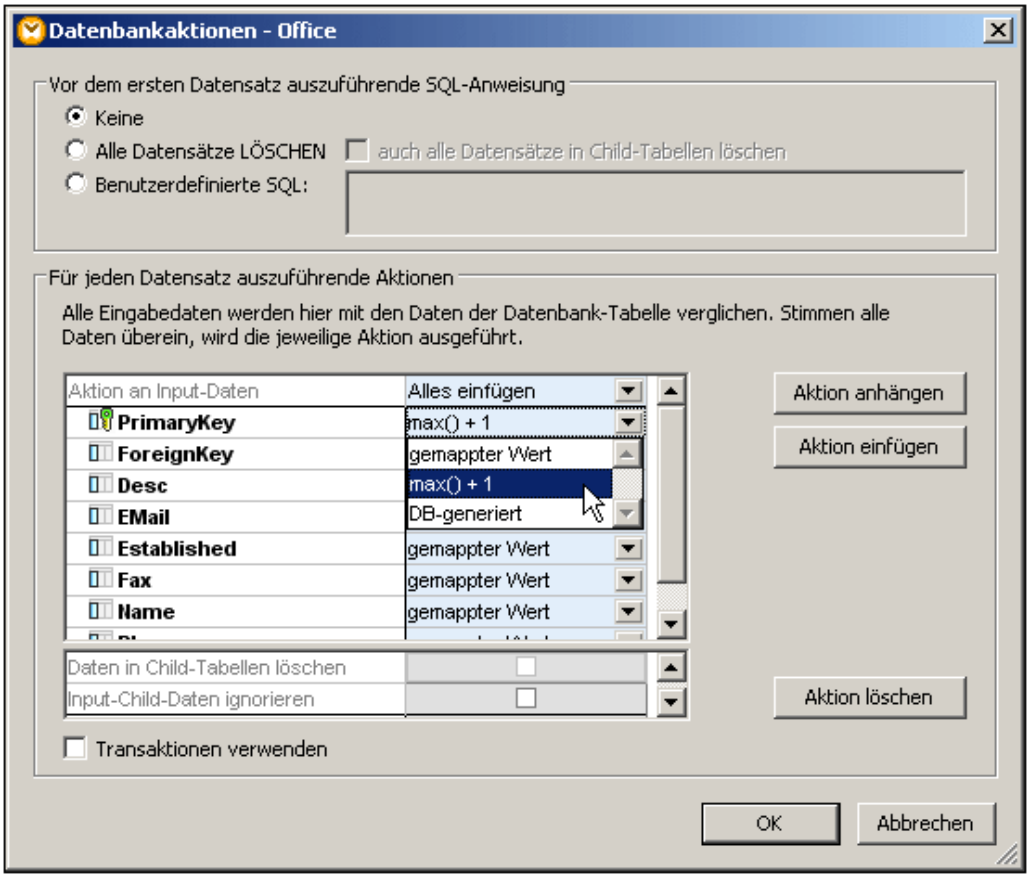
Einfügen von Tabellen und verwandten Child-Tabellen:

1. Öffnen Sie ein neues Mapping-Fenster und fügen Sie das Schema **Altova_Hierarchical.xsd** ein (und weisen Sie ihm **altova-cmpy.xml** als XML-Eingabeinstanz zu). Fügen Sie die Datenbank **altova.mdb**, wie im Abschnitt Einrichten von XML-auf-Datenbank Mappings beschrieben, in das Mapping-Fenster ein.
2. Erzeugen Sie Mappings zwischen den Datenelementen Altova und Name, wie im vorherigen Abschnitt beschrieben.
3. Klicken Sie in beiden Komponenten auf das +-Erweiterungssymbol des Datenelements **Office**, um die Office-Tabellenfelder anzuzeigen.
4. Erzeugen Sie zwischen den beiden Komponenten die folgenden Mappings:
   Office auf Office
   Desc auf Desc
   EMail auf EMail
   Established auf Established, und
   Name auf Name.



5. Klicken Sie auf das A:In Tabellenaktionssymbol neben dem Tabelleneintrag **Office.** Die Tabellenaktion **Alles einfügen..** ist standardmäßig ausgewählt. Sie müssen hier keine Änderungen vornehmen.
6. Klicken Sie in die **PrimaryKey**-Auswahlliste und wählen Sie den Eintrag **max()+1** und klicken Sie zur Bestätigung auf OK.

© 2011 Altova GmbH



7. Klicken Sie auf die Schaltfläche Ausgabe, um das SQL-Skript anzuzeigen.

```
/*
The following SQL statements are only for preview and may not be executed in another SQL query tool!
To execute these statements use function "Run SQL-script" from menu "Output".
Connect to database using the following connection-string:
Provider=Microsoft.Jet.OLEDB.4.0; Data Source=C:\Documents and Settings\...\My Documents\Altova\
MapForce2012\MapForceExamples\altova.mdb;
*/

SELECT IIF(MAX([Altova].[PrimaryKey]) IS NULL,0,MAX([Altova].[PrimaryKey]))+1 AS [PrimaryKey] FROM
 [Altova]
-->>> %PrimaryKey1%

INSERT INTO [Altova] ([Name], [PrimaryKey]) VALUES ('Microtech OrgChart', '%PrimaryKey1%')

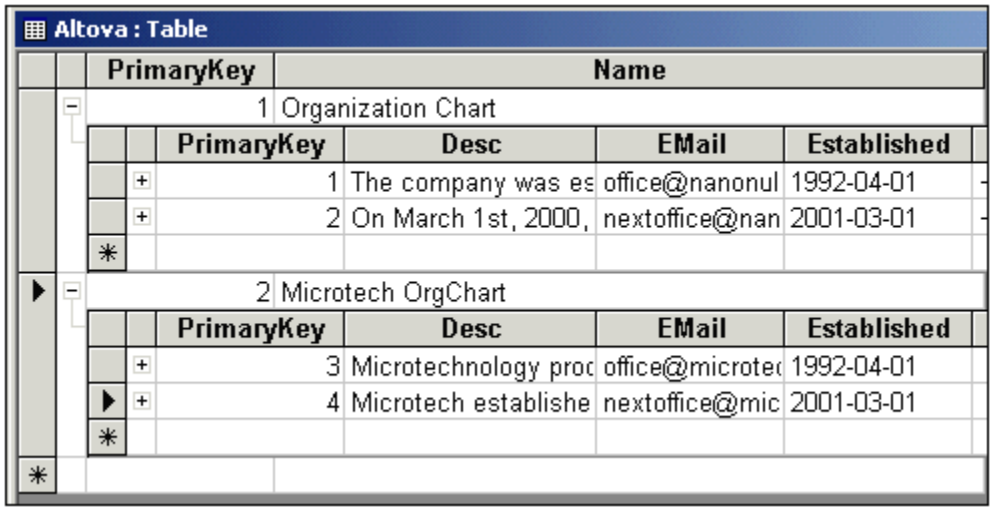
SELECT IIF(MAX([Office].[PrimaryKey]) IS NULL,0,MAX([Office].[PrimaryKey]))+1 AS [PrimaryKey] FROM
[Office]
-->>> %PrimaryKey2%

INSERT INTO [Office] ([ForeignKey], [Desc], [EMail], [Established], [Name], [PrimaryKey]) VALUES (
'%PrimaryKey1%', 'Microtechnology products are currently the bleeding edge of computer technology.',
'office@microtech.com', '1992-04-01', 'Microtech, Inc.', '%PrimaryKey2%')

SELECT IIF(MAX([Office].[PrimaryKey]) IS NULL,0,MAX([Office].[PrimaryKey]))+1 AS [PrimaryKey] FROM
[Office]
-->>> %PrimaryKey3%

INSERT INTO [Office] ([ForeignKey], [Desc], [EMail], [Established], [Name], [PrimaryKey]) VALUES (
'%PrimaryKey1%', 'Microtech established its new office on Feb 30', 'nextoffice@microtech.com',
'2001-03-01', 'Microtech Partners, Inc.', '%PrimaryKey3%')
```

8. Klicken Sie auf das Symbol "SQL-Script ausführen" , um das Skript zu starten und die neuen Tabellen einzufügen.
9. Öffnen Sie die Datenbank in MS Access und doppelklicken Sie auf die Tabelle **Altova**, um das Ergebnis in MS-Access zu sehen.



Altova : Table

PrimaryKey	Name
1	Organization Chart

PrimaryKey	Desc	EMail	Established
1	The company was es	office@nanonul	1992-04-01
2	On March 1st, 2000,	nextoffice@nan	2001-03-01

PrimaryKey	Name
2	Microtech OrgChart

PrimaryKey	Desc	EMail	Established
3	Microtechnology proc	office@microte	1992-04-01
4	Microtech establishe	nextoffice@mic	2001-03-01

Dem Microtech OrgChart wurden zwei neue Büros (Office) hinzugefügt.

10. Doppelklicken Sie auf die Tabelle **Office,** um die Einzelheiten des erzeugten Ergebnisses zu sehen.

© 2011 Altova GmbH

Office : Table

	PrimaryKey	ForeignKey	Desc	EMail
▶	1	1	The company was establis	office@nanonul
	2	1	On March 1st, 2000, Nanoi	nextoffice@nan
	3	2	Microtechnology products	office@microte
	4	2	Microtech established its r	nextoffice@mic
*				

Die neuen Büros wurden mit den Primärschlüsseln 3 bzw. 4 hinzugefügt. Beide Büros sind über den Sekundärschlüssel 2 mit der Tabelle Altova verknüpft, der auf den Datensatz Microtech OrgChart referenziert.

```
SELECT IIF(MAX([Altova].[PrimaryKey]) IS NULL,0,MAX([Altova].[PrimaryKey]))+1 AS [PrimaryKey] FROM [Altova]
-->>> OK. One or more rows.

INSERT INTO [Altova] ([Name], [PrimaryKey]) VALUES ('Microtech OrgChart', 2)
-->>> OK. 1 row(s).

SELECT IIF(MAX([Office].[PrimaryKey]) IS NULL,0,MAX([Office].[PrimaryKey]))+1 AS [PrimaryKey] FROM [Office]
-->>> OK. One or more rows.

INSERT INTO [Office] ([ForeignKey], [Desc], [EMail], [Established], [Name], [PrimaryKey]) VALUES (2,
'Microtechnology products are currently the bleeding edge of computer technology.', 'office@microtech.com',
'1992-04-01', 'Microtech, Inc.', 3)
-->>> OK. 1 row(s).

SELECT IIF(MAX([Office].[PrimaryKey]) IS NULL,0,MAX([Office].[PrimaryKey]))+1 AS [PrimaryKey] FROM [Office]
-->>> OK. One or more rows.

INSERT INTO [Office] ([ForeignKey], [Desc], [EMail], [Established], [Name], [PrimaryKey]) VALUES (2, 'Microtech
established its new office on Feb 30', 'nextoffice@microtech.com', '2001-03-01', 'Microtech Partners, Inc.', 4)
-->>> OK. 1 row(s).
```

13.1.2 Datenbankaktion: Aktualisieren

In diesem Abschnitt wird erklärt, wie man Datenbanken aktualisiert. Die verschiedenen Datenbankaktionen, die MapForce zu diesem Zweck zur Verfügung stellt, werden beschrieben.

Ziel
In diesem Abschnitt des Tutorials lernen Sie, wie Sie mit Hilfe von MapForce Daten in einer Datenbanktabelle aktualisieren können. Insbesondere werden die folgenden Schritte beschrieben:

- Aktualisieren der Fleder einer bestimmten Tabelle
- Hinzufügen neuer Datensätze zu einer Tabelle
- Verwendung der Aktualisieren, wenn... Bedingung

In diesem Abschnitt verwendete Befehle

XML Schema/Datei einfügen: Klicken Sie auf diese Schaltfläche, um das Windows-Standarddialogfeld "Öffnen" aufzurufen und die Datei aus Ihrem Dateisystem auszuwählen.

Datenbank einfügen: Dieser Befehl befindet sich in der Symbolleiste "Komponente einfügen" und im Menü **Einfügen.** Klicken Sie auf diesen Befehl, um ein Dialogfeld **Datenbank auswählen** zu öffnen, in dem Sie eine Datenbankverbindung definieren und auswählen können, welche Datenbanktabellen in das MapForce Mapping inkludiert werden sollen.

Idente Sub-Einträge automatisch verbinden: Klicken Sie auf diese Schaltfläche, um die automatische Verbindung identer Sub-Einträge ein- bzw. auszuschalten.

Datenbankaktionen: Dieser Befehl befindet sich im Menü **Komponente** einer Datenbankkomponente und wird in der Datenbankkomponente in Form einer Schaltfläche rechts vom Tabellennamen angezeigt. Klicken Sie auf diesen Befehl um das Dialogfeld **Datenbankaktionen** aufzurufen, in dem Sie definieren können, welche Aktionen für die einzelnen Datensätze ausgeführt werden sollen.

SQL-Script ausführen: Dieser Befehl befindet sich in der Symbolleiste "Ausgabevorschau" und im Menü "Ausgabe". Klicken Sie auf diesen Befehl, um das im Ausgabefenster generierte SQL-Script auszuführen.

In diesem Beispiel verwendete Dateien

- Altova_Hierarchical.xsd
- altova-cmpy.xml
- altova-cmpy-extra.xml
- altova.mdb

Aktualisieren von Datenbankfeldern

Im ersten Beispiel dieses Abschnitts wird die einfache Aufgabe erklärt, wie man bestehende Personendatensätze aktualisiert. Es werden Mappings von der XML-Datenquelle zur "Root"-Tabelle Person erzeugt.

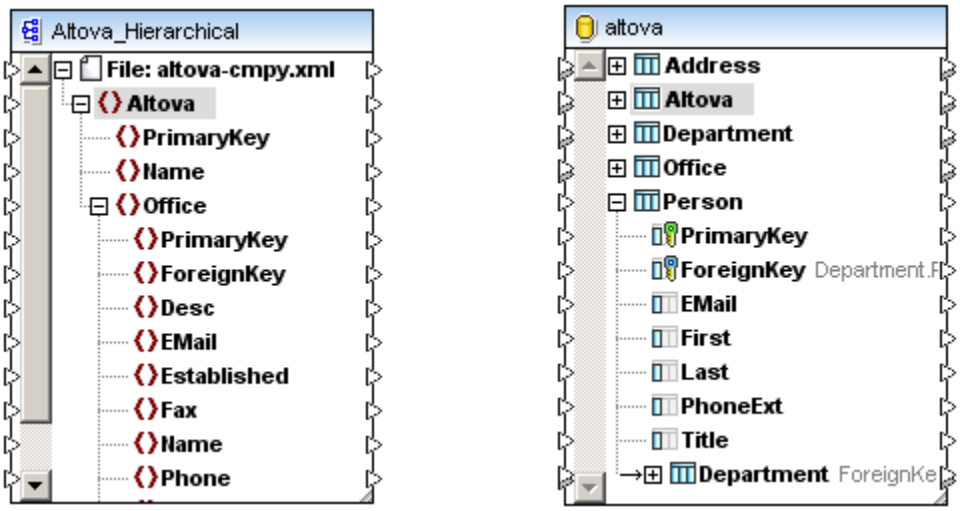
So aktualisieren Sie die Tabelle "Person":

1. Fügen Sie das Schema Altova_Hierarchical.xsd ein (und weisen Sie ihm

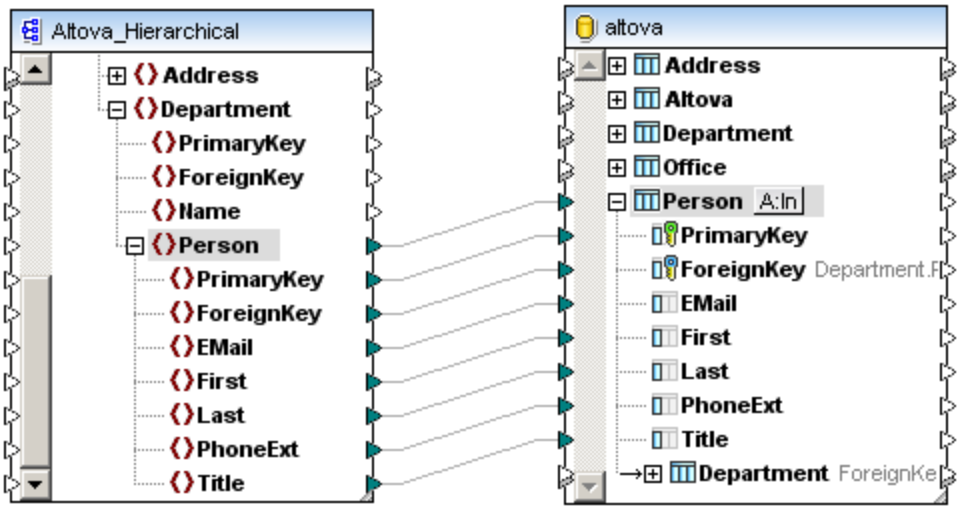
 © 2011 Altova GmbH

altova-cmpy.xml als XML-Eingabeinstanz zu), wie im vorigen Abschnitt beschrieben.

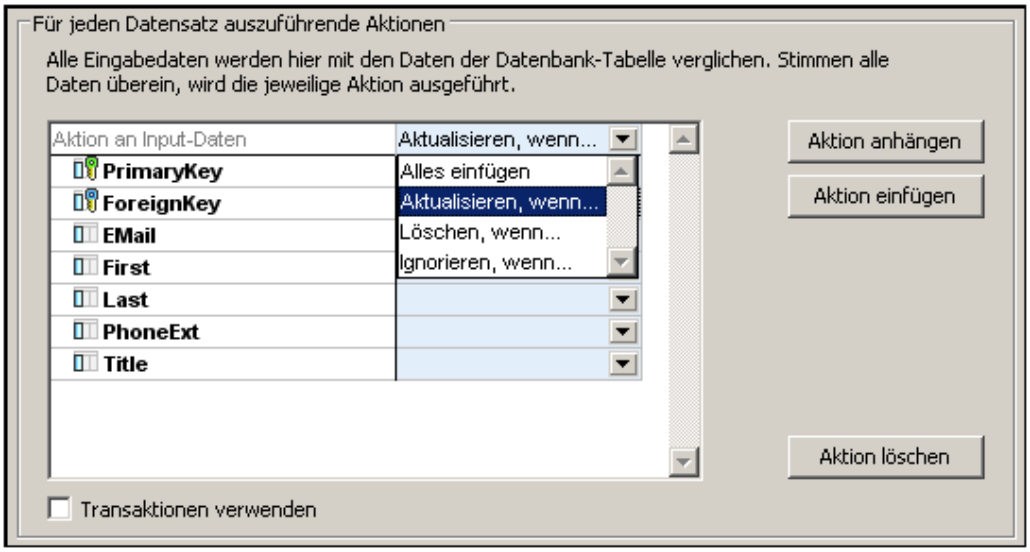
2. Fügen Sie die MS Access-Datenbank **altova.mdb** in das Mapping ein, wie im vorigen Abschnitt beschrieben.



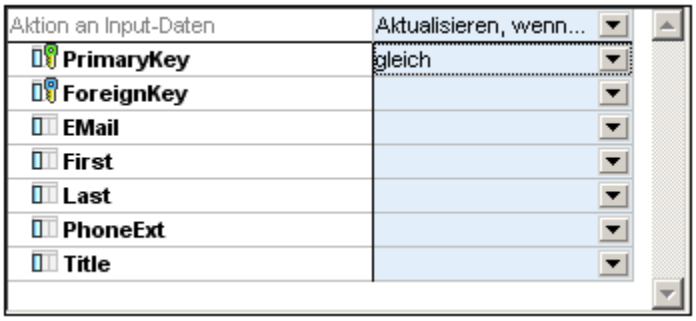
3. Aktivieren Sie die Schaltfläche **Idente Sub-Einträge automatisch verbinden** .
4. Erweitern Sie in der XML-Schema-Komponente die Datenelemente Office und Department, und klicken Sie auf das Element **Person** und ziehen Sie den Konnektor zum Datenelement "Person" der Datenbank. Stellen Sie sicher, dass Sie es mit der "**Root**"-Tabelle "Person" verbinden.
   Alle übereinstimmenden Child-Elemente werden automatisch gemappt.



5. Klicken Sie auf das **Person**-Tabellenaktionssymbol A:In, um das Dialogfeld zu öffnen.
6. Klicken Sie in der Auswahlliste "Aktion an Input-Daten" auf den obersten Eintrag und wählen Sie **Aktualisieren, wenn...**

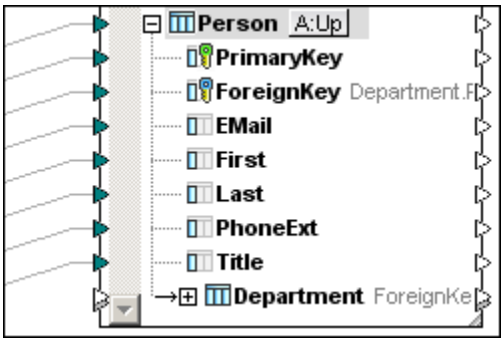


7. Klicken Sie auf das Dropdown-Listenfeld in der Zeile des Eintrags PrimaryKey, wählen Sie den Eintrag **gleich** und bestätigen Sie mit **OK**.



Die Primärschlüssel von Quelle und Ziel werden mit Hilfe des Operators "gleich" verglichen. Wenn die beiden Schlüssel identisch sind, werden alle gemappten Felder der Personentabellen aktualisiert.

Das Tabellenaktionssymbol hat sich jetzt in **A:Up** geändert, um anzuzeigen, dass die Tabellenaktion "Aktualisieren" (Update) ausgewählt wurde.



8. Klicken Sie auf die Schaltfläche Ausgabe am unteren Rand des Mapping-Fensters, um das von diesem Mapping erzeugte SQL-Script anzuzeigen. Das Script führt das Mapping auf die Zieldatenbank unter Berücksichtigung der definierten Tabellenaktionen aus.
9. Klicken Sie in der Funktionsleiste auf die Schaltfläche **SQL-Script ausführen** , um

 © 2011 Altova GmbH

das Script auszuführen und die Datensätze der Datenbank zu aktualisieren. Bei erfolgreicher Ausführung erscheint eine Bestätigungsmeldung im Fenster "Meldungen".

10. Öffnen Sie die **Altova-**Datenbank in MS Access und doppelklicken Sie auf die Tabelle **Person,** um das Ergebnis zu sehen.
    Alle Personendatensätze der Datenbank wurden aktualisiert.

Person : Table

PrimaryKey	ForeignKey	EMail	First	Last	PhoneExt
1	1	A.Aldrich@micı	Albert	Aldrich	582
2	1	b.bander@micr	Bert	Bander	471
3	1	c.clovis@micro	Clive	Clovis	963
4	2	d.Durnell@micr	Dave	Durnelll	621
5	2	e.ellas@microt	Eve	Ellas	753
6	3	f.fortunas@micı	Fred	Fortunas	951
7	3	g.gundall@micr	Gerry	Gundall	654
8	3	h.hardy@micro	Harry	Hardy	852
9	3	i.idilko@microte	Ingrid	Idilko	951
10	3	j.judy@microte	June	Judy	753
11	3	k.krove@microt	Karl	Krove	334

Aktualisieren und Hinzufügen von neuen Datensätzen

In diesem etwas komplexeren Beispiel wird versucht, Datensätze sowohl in der Tabelle "Department" als auch in der Tabelle "Person" zu aktualisieren und außerdem alle neuen Personendatensätze, die die XML-Eingabedatei enthält, hinzuzufügen. Als "**Root**"-Tabelle wird in diesem Beispiel daher die Tabelle **Department** verwendet.

In diesem Beispiel verwendete Dateien:

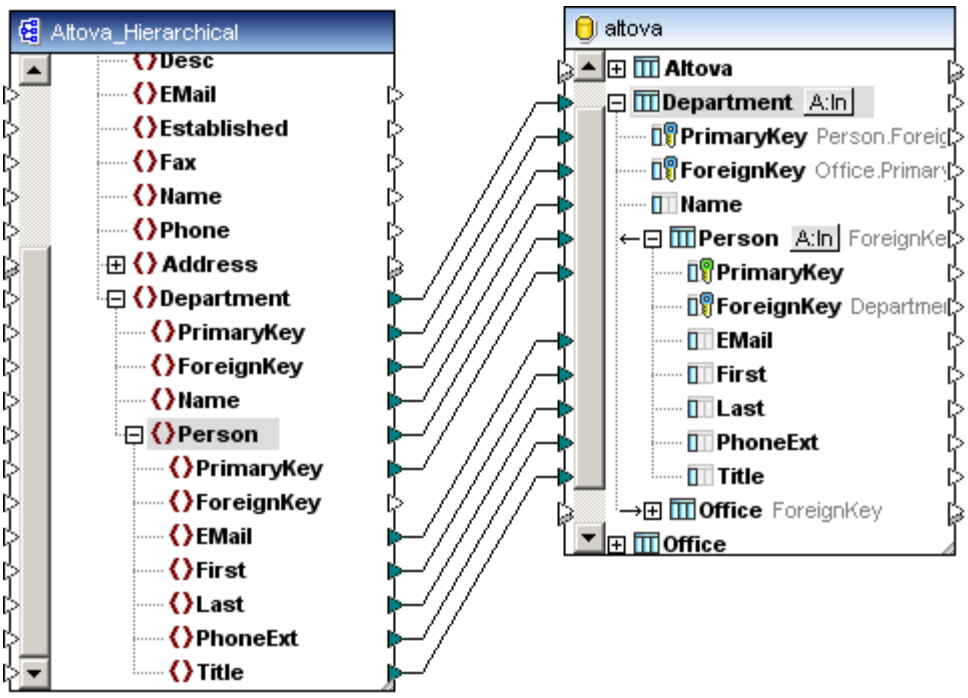
- Altova_Hierarchical.xsd
- altova-cmpy-**extra**.xml (die XMl-Instanzdatei für Altova_hierarchical.xsd)
- altova.mdb

Ziel:

- **Aktualisierung** der Datensätze "Name" der Tabelle "Department"
- **Aktualisierung** vorhandener Personendatensätze
- **Einfügen** aller neuen Personendatensätze

So mappen Sie die Tabellen Department und Person aufeinander:

1. Fügen Sie das Schema Altova_Hierarchical.xsd ein (und weisen Sie ihm **altova-cmpy.xml** als XML-Eingabeinstanz zu), wie im vorigen Abschnitt beschrieben.
2. Fügen Sie die MS Access-Datenbank **altova.mdb** in das Mapping ein, wie im vorigen Abschnitt beschrieben.
3. Aktivieren Sie die Schaltfläche **Idente Sub-Einträge automatisch verbinden** .
4. Erweitern Sie in der Datenbankkomponente die Datenelemente "Department" und "Person".
5. Erweitern Sie in der XML-Schema-Komponente die Datenelemente "Office" und "Department". Klicken Sie auf das Datenelemente **Person** und ziehen Sie den Konnektor auf das Datenelement "Person" der Datenbank. Stellen Sie sicher, dass Sie es mit der Tabelle "Person" innerhalb der Tabelle "Department" verbinden.
   Alle übereinstimmenden untergeordneten Datenelemente werden automatisch verbunden.



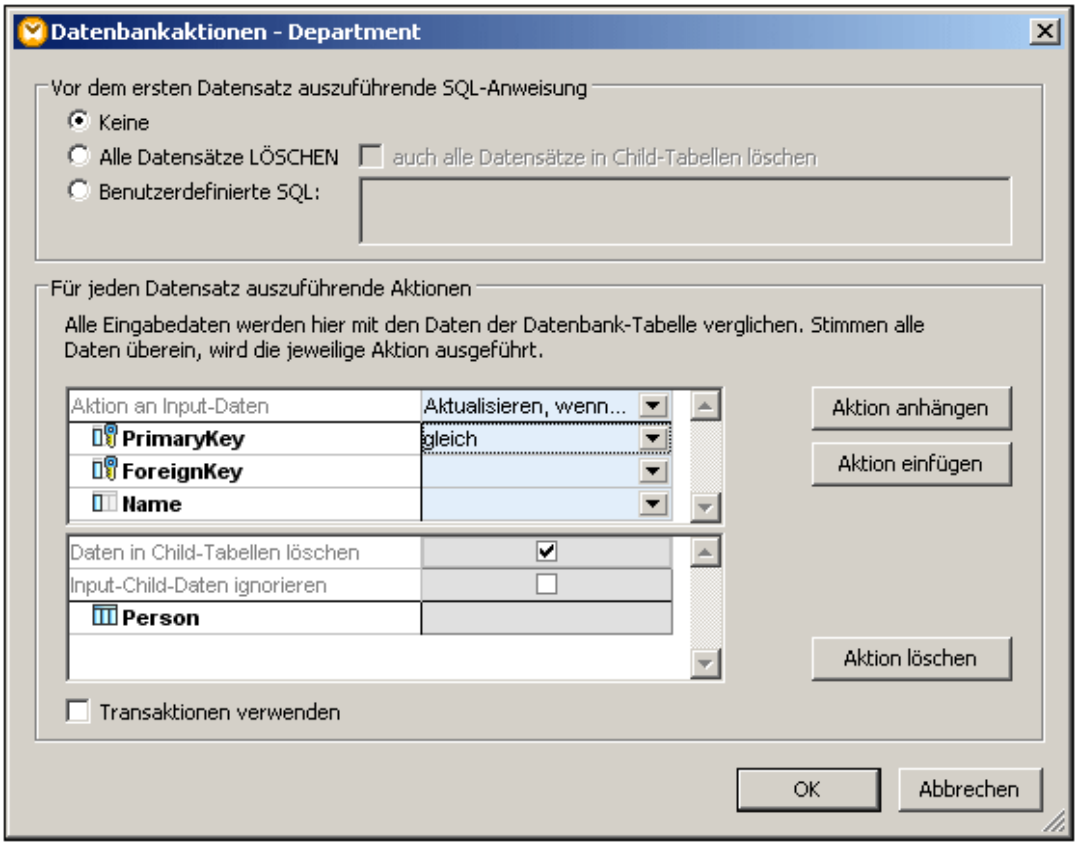
Definieren der Tabellenaktionen

Die Quell- und Zielprimärschlüssel der beiden Tabellen werden mit Hilfe des Operators "gleich" verglichen. Sind die beiden Schlüssel identisch, werden die **gemappten** Felder der Tabellen "Department" und "Person" aktualisiert. Wenn der Vergleich (in der Tabelle Person) fehlschlägt, wird die nächste Tabellenaktion durchgeführt, d.h. "Rest einfügen".

So definieren Sie die Tabellenaktion für die Tabelle "Department":

1. Klicken Sie auf das A:In Tabellenaktionssymbol von **Person,** um das Dialogfeld zu öffnen.
2. Klicken Sie in der Auswahlliste "Aktion an Input-Daten" auf den obersten Eintrag und wählen Sie **Aktualisieren, wenn...**
3. Klicken Sie in die Auswahlliste in der Zeile mit dem Eintrag PrimaryKey, wählen Sie den Eintrag **gleich** und klicken Sie zur Bestätigung auf **OK**.

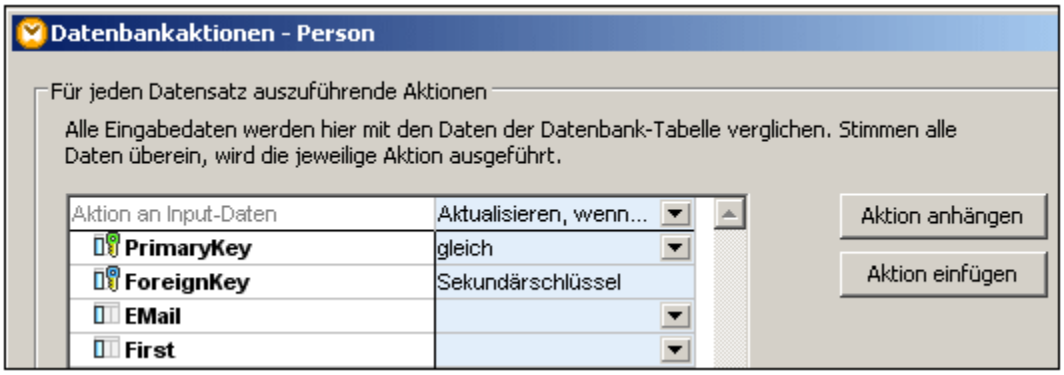
 © 2011 Altova GmbH



Die Primärschlüssel der Quell- und Zielkomponente werden mit Hilfe des "gleich" Operators verglichen. Wenn die beiden Schlüssel identisch sind, werden alle gemappten Felder der Department-Tabellen aktualisiert.

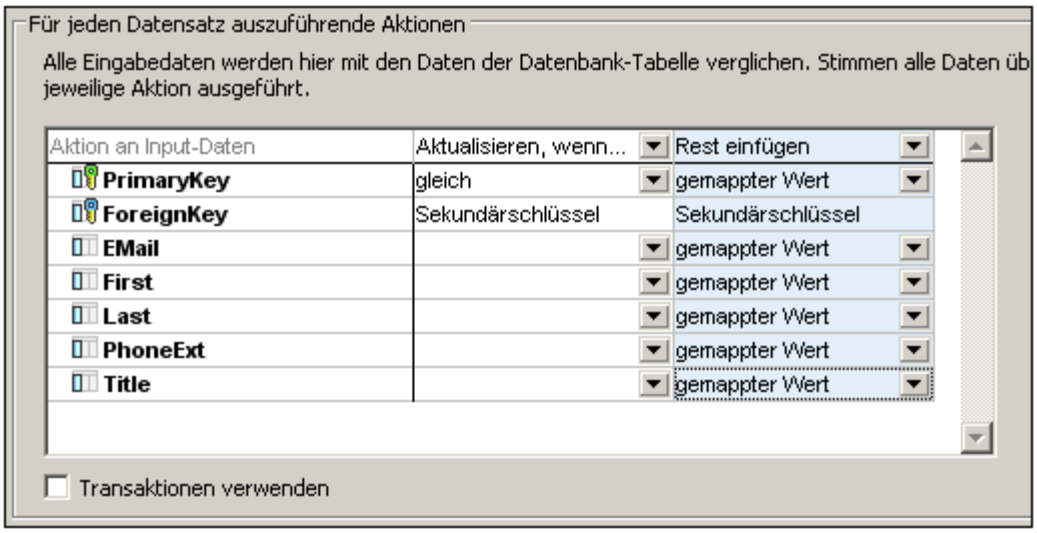
So definieren Sie die Tabellenaktion für die Tabelle "Person":

1. Klicken Sie auf das A:In Tabellenaktionssymbol von **Department,** um das Dialogfeld zu öffnen.
2. Klicken Sie in der Auswahlliste "Aktion an Input-Daten" auf den obersten Eintrag und wählen Sie **Aktualisieren, wenn...**
3. Klicken Sie in die Auswahlliste in der Zeile mit dem Eintrag PrimaryKey und wählen Sie den Eintrag **gleich**.



- Klicken Sie auf die Schaltfläche **Aktion anhängen**, um eine neue

Tabellenaktionsspalte anzuhängen.
Die Tabellenaktion **Rest einfügen** wird automatisch als zweite Tabellenaktion eingefügt. Wenn die Aktion "Aktualisieren, wenn.." fehlschlägt, dann wird diese Tabellenaktion verarbeitet.



2. Klicken Sie auf OK, um die Definition der Tabellenaktionen fertig zu stellen. Beachten Sie, dass das Tabellenaktionssymbol nun zu **A:Up,In** aktualisiert wurde.

4. Klicken Sie auf die Schaltfläche "Ausgabe", um das SQL Script zu sehen.

```
UPDATE [Department] SET [ForeignKey] = 1, [Name] = 'Admin' WHERE ([Department].[PrimaryKey]=1)

SELECT [ForeignKey], [PrimaryKey] FROM [Department] WHERE ([PrimaryKey]=1)
-->>> %ForeignKey1%

-->>> %PrimaryKey1%

DELETE FROM [Person] WHERE EXISTS(SELECT * FROM [Department] WHERE [Department].[PrimaryKey] =
[Person].[ForeignKey] AND ([Department].[PrimaryKey]='%PrimaryKey1%'))

UPDATE [Person] SET [EMail] = 'A.Aldrich@microtech.com', [First] = 'Albert', [Last] = 'Aldrich', [PhoneExt] = 582
, [Title] = 'Manager' WHERE ([Person].[ForeignKey] = '%PrimaryKey1%') AND ([Person].[PrimaryKey]=1)
```

Bitte beachten Sie:
Das Script führt das Mapping auf die Zieldatenbank unter Berücksichtigung der definierten Tabellenaktionen aus.

Verarbeitungssreihenfolge

Die Bedingung **Aktualisieren, wenn...** überprüft, ob die Primärschlüssel des Quell- und Zieldatenelements identisch sind.

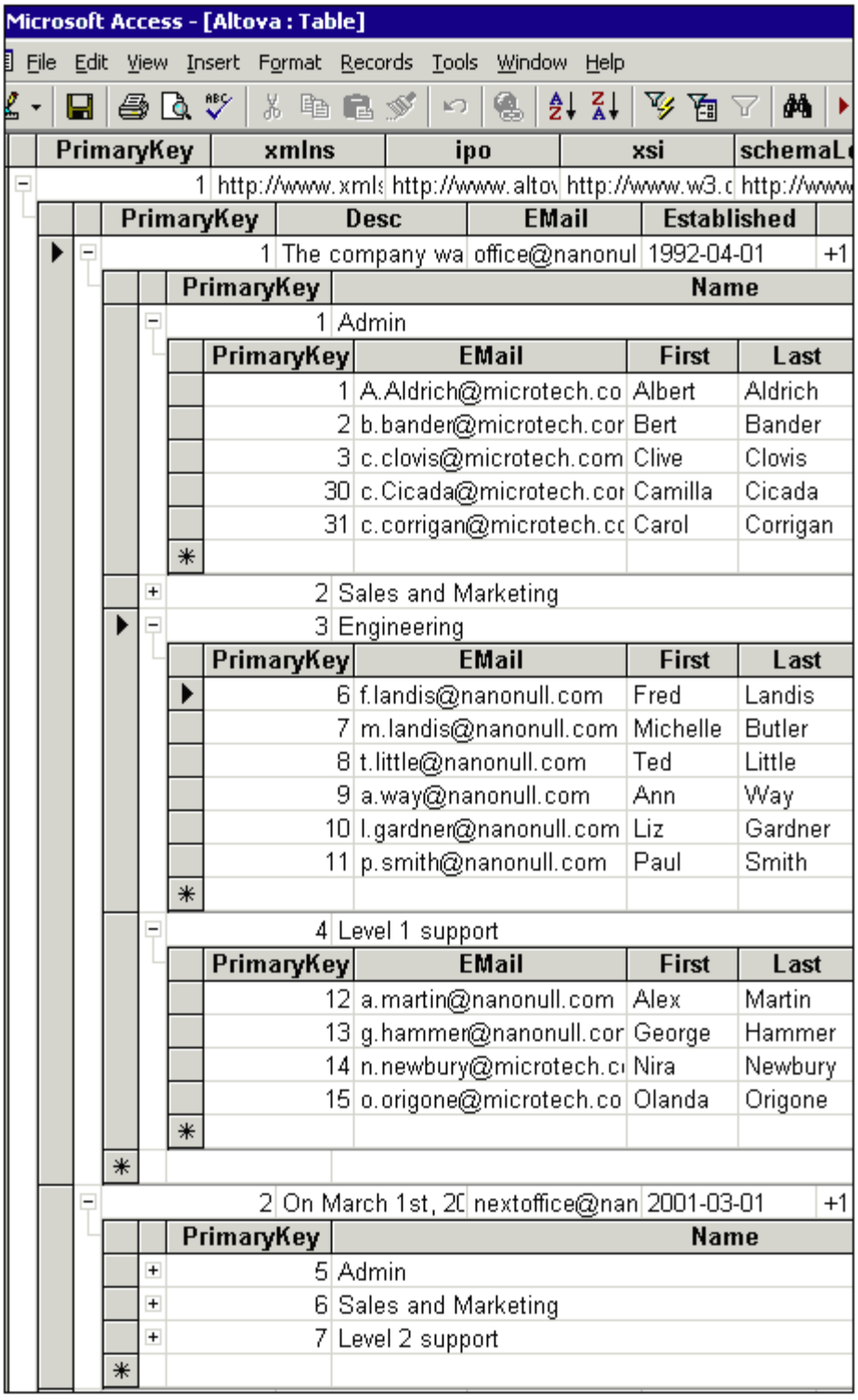
 © 2011 Altova GmbH

Tabelle "Department"

- Wenn die **Bedingung** "**true"** ist, wird jeder Department-Datensatz, in dem die Schlüssel identisch sind, aktualisiert. Wenn es in der Datenbank Datensätze gibt, die in der Quelldatei keine Entsprechung haben, werden diese Datensätze beibehalten und bleiben unverändert (in diesem Beispiel die Tabelle Engineering).
- Wenn die **Bedingung "false" ist,** d.h. wenn Schlüssel in der Quellkomponente vorhanden sind, die in der Zieldatenbank keine Entsprechung haben, wird keiner der Department-Datensätze aktualisiert.

Tabelle "Person":

- Wenn die **Bedingung** "**true"** ist, wird jeder Person-Datensatz, in dem die Schlüssel identisch sind, aktualisiert. Wenn es in der Datenbank Datensätze gibt, die in der Quelldatei keine Entsprechung haben, werden diese Datensätze beibehalten und bleiben unverändert.
- Wenn die **Bedingung "false" ist,** d.h. wenn Schlüssel in der Quellkomponente vorhanden sind, die in der Zieldatenbank keine Entsprechung haben, wird zur nächsten Tabellenaktionsspalte weitergegangen: **Rest einfügen**...Mit dieser Aktion werden neue Person-Datensätze, falls vorhanden, in die Tabelle "Person" eingefügt. In diesem Beispiel werden der Abteilung Admin zwei neue Personendatensätze mit den Primärschlüsseln 30 bzw. 31 hinzugefügt.



Microsoft Access - [Altova : Table]

File Edit View Insert Format Records Tools Window Help

PrimaryKey	xmlns	ipo	xsi	schemaL
1	http://www.xmls	http://www.altov	http://www.w3.c	http://www

PrimaryKey	Desc	EMail	Established	
1	The company wa	office@nanonul	1992-04-01	+1

PrimaryKey	Name
1	Admin

PrimaryKey	EMail	First	Last
1	A.Aldrich@microtech.co	Albert	Aldrich
2	b.bander@microtech.cor	Bert	Bander
3	c.clovis@microtech.com	Clive	Clovis
30	c.Cicada@microtech.cor	Camilla	Cicada
31	c.corrigan@microtech.cc	Carol	Corrigan

PrimaryKey	Name
2	Sales and Marketing
3	Engineering

PrimaryKey	EMail	First	Last
6	f.landis@nanonull.com	Fred	Landis
7	m.landis@nanonull.com	Michelle	Butler
8	t.little@nanonull.com	Ted	Little
9	a.way@nanonull.com	Ann	Way
10	l.gardner@nanonull.com	Liz	Gardner
11	p.smith@nanonull.com	Paul	Smith

PrimaryKey	Name
4	Level 1 support

PrimaryKey	EMail	First	Last
12	a.martin@nanonull.com	Alex	Martin
13	g.hammer@nanonull.cor	George	Hammer
14	n.newbury@microtech.c	Nira	Newbury
15	o.origone@microtech.co	Olanda	Origone

PrimaryKey	Desc	EMail	Established	
2	On March 1st, 2(	nextoffice@nan	2001-03-01	+1

PrimaryKey	Name
5	Admin
6	Sales and Marketing
7	Level 2 support

© 2011 Altova GmbH

Kapitel 14

Mappen von HL7 v3.x von/auf XML-Schemas

14 Mappen von HL7 v3.x von/auf XML-Schemas

MapForce 2012 unterstützt automatisch HL7 Version **3.x**., da diese auf XML basiert.

Auf der Seite MapForce-Bibliotheken der Altova-Website finden Sie einen separaten Installer für die HL7 V2.2 - V2.5.1 XML-Schemas und Konfigurationsdateien. Wählen Sie im Installer die benutzerdefinierte Installation aus, um nur die HL7 V3-Komponenten und XML-Schemas zu installieren.

Pfad der HL7 XML-Schemas nach der Installation:

Windows XP-System:	“C:\Program Files\Altova\Common2011\Schemas\ hl7v3“
Windows Vista-System:	“C:\Program Files\Altova\Common2011\Schemas\ hl7v3“
Windows7-System:	“C:\Program Files\Altova\Common2011\Schemas\ hl7v3“

Wenn eine 32-Bit-MapForce Applikation auf einem 64-Bit-Betriebssystem verwendet wird, so lautet der Pfad
“C:\Program Files(x86)\Altova\Common2011\Schemas\ hl7v3“.

HL7-Dokumente können in MapForce als Quell- und Zielkomponenten verwendet werden. Diese Daten können auf beliebig viele XML-Schema- Komponenten gemappt werden.

© 2011 Altova GmbH

Kapitel 15

Bibliotheken und Funktionen

15 Bibliotheken und Funktionen

In den folgenden Abschnitten wird beschrieben, wie Sie Ihre eigenen benutzerdefinierten Funktionen sowie Bibliotheken für die verschiedenen Programmiersprachen definieren.

Definieren benutzerdefinierter Funktionen

Hinzufügen benutzerdefinierter XSTL- und XQuery-Funktionen

Hinzufügen benutzerdefinierter Java, C# und C++-Funktionsbibliotheken

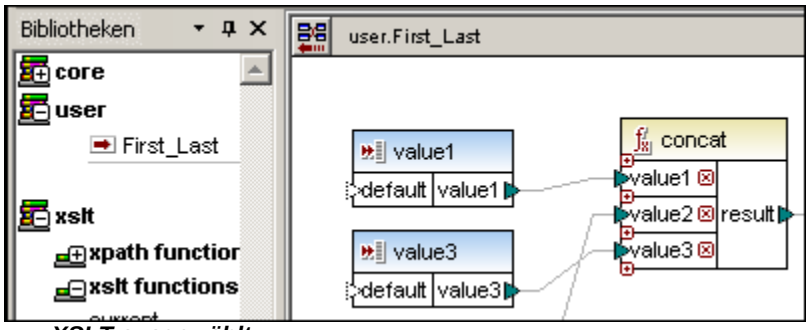
Referenz Funktionsbibliotheken

© 2011 Altova GmbH

15.1 Definieren benutzerdefinierter Funktionen

In MapForce können Sie benutzerdefinierte Funktionen visuell auf dieselbe Art wie im Hauptmappingfenster definieren.

Diese Funktionen stehen dann im Fenster "Bibliotheken" als Einträge zur Verfügung (z.B. First_Last) und können wie die bereits vorhandenen Funktionen verwendet werden. Auf diese Art können Sie Ihre Mappings in kleinere Bausteine aufteilen und diese mehrmals und auch in anderen Mappings verwenden.



XSLT ausgewählt

Benutzerdefinierte Funktionen werden zusammen mit dem Hauptmapping in der ***.mfd**-Datei gespeichert.

In einer benutzerdefinierter Funktion werden **Input-** und **Output-Komponenten** verwendet, um Informationen aus dem Hauptmapping (oder einer anderen benutzerdefinierten Funktion) an die benutzerdefinierte Funktion zu übergeben und anschließend wieder zurückzugeben.

Benutzerdefinierte Funktionen können "lokale" Quellkomponenten (also Komponenten, die sich selbst innerhalb der benutzerdefinierten Funktion befinden) wie z.B. XML-Schemas enthalten, die für die Implementierung von lookup-Funktionen nützlich sind.

Benutzerdefinierte Funktionen können aus beliebig vielen Inputs und Outputs bestehen, wobei jeder dieser Inputs und Outputs aus einfachen Werten, oder XML Nodes Datenbanken oder bestehen kann.

Benutzerdefinierte Funktionen eigenen sich für folgende Zwecke:

- um mehrere Verarbeitungsfunktionen zu einer einzigen Komponente zusammenzufassen, z.B. um ein bestimmtes Feld zu formatieren oder einen Wert nachzuschlagen
- um diese Komponenten beliebig oft wiederzuverwenden
- um benutzerdefinierte Funktionen durch Laden der Mappingdatei als Bibliothek in andere Mappings zu importieren.
- um ein komplexes Mapping mittels inline Funktionen in kleinere Teile aufzuteilen, die separat bearbeitet werden können
- um **rekursive Schemas** durch Erstellung von rekursiven benutzerdefinierten Funktionen zu mappen

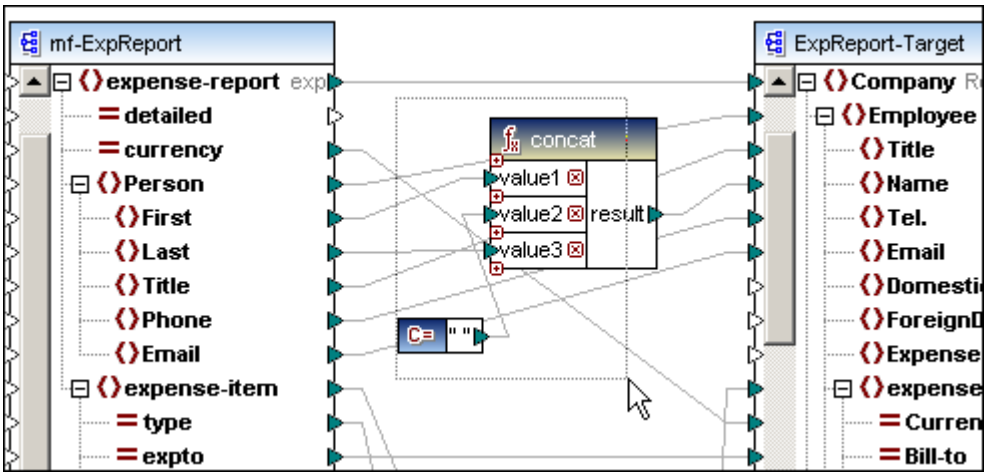
Benutzerdefinierte Funktionen können entweder von Grund auf neu erstellt werden oder aus Funktionen, die bereits auf dem Register "Mapping" verfügbar sind, zusammengesetzt werden.

In diesem Beispiel wird die Datei **Tut-ExpReport.mfd** aus dem Ordner **...\MapForceExamples\Tutorial** verwendet.

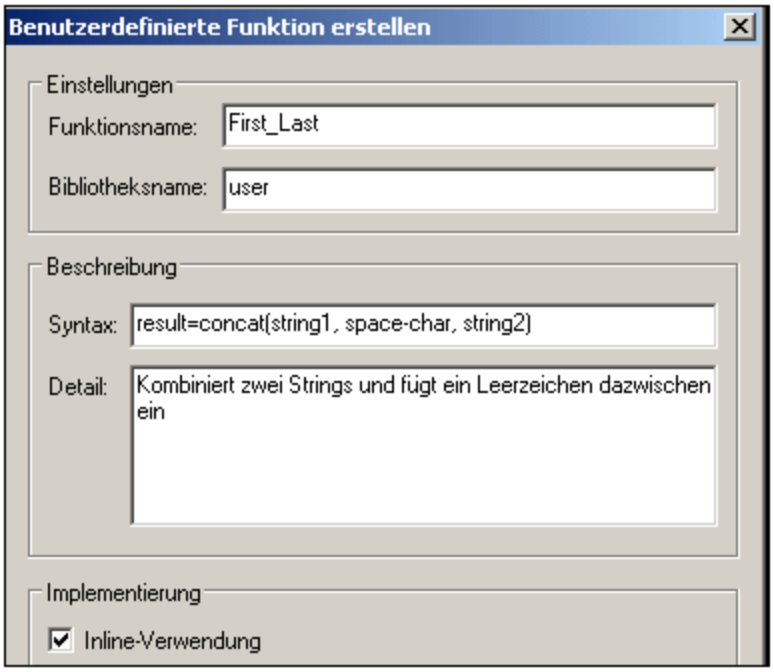
So erstellen Sie eine benutzerdefinierte Funktion aus vorhandenen Komponenten:

1. Ziehen Sie mit der Maus ein Rechteck über die concat-Funktion und die

constant-Funktion, um sie zu markieren (Sie können auch die Strg-Taste gedrückt halten und auf die Funktionen klicken, um diese zu markieren).

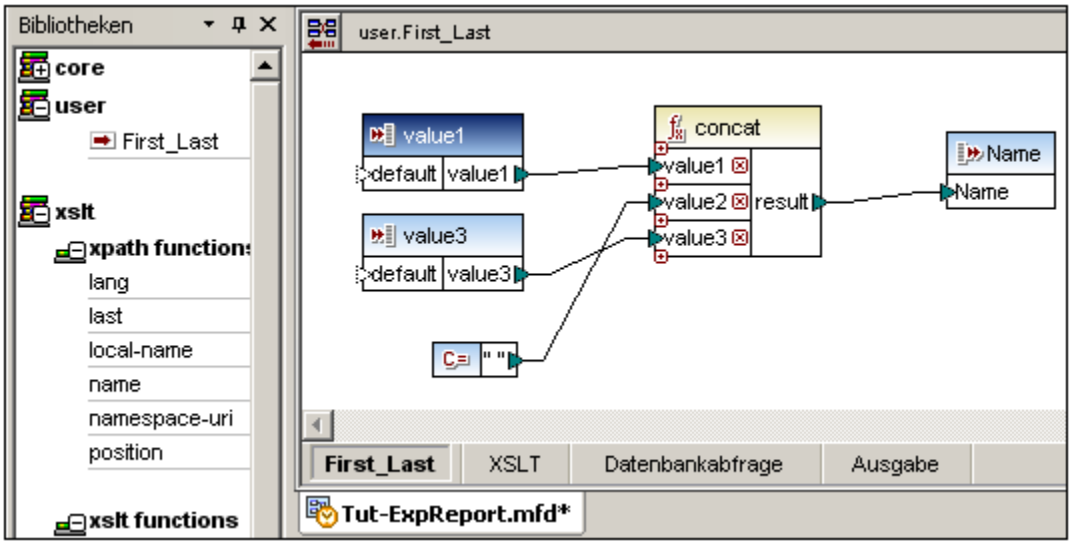


2. Wählen Sie die Menüoption **Funktion | Benutzerdefinierte Funktion von Auswahl erstellen**.
3. Geben Sie einen Namen für die neue benutzerdefinierte Funktion ein (First_Last). Hinweis: Gültige Zeichen sind: alphanumerische, a-z, A-Z, 0-9 sowie Unterstrich "_", Bindestrich/Schrägstrich "-" und Doppelpunkt ":".
4. Über die Felder "Syntax" und "Detail" können Sie zusätzliche Informationen zur neuen Funktion eingeben. Bestätigen Sie die Eingabe anschließend mit OK. Der von Ihnen eingegebene Text wird anschließend als Tooltip angezeigt, wenn Sie den Cursor über die Funktion platzieren.
   Standardmäßig wird als Bibliotheksname "user" vorgeschlagen. Sie können in diesem Feld aber natürlich Ihren eigenen Bibliotheksnamen definieren.



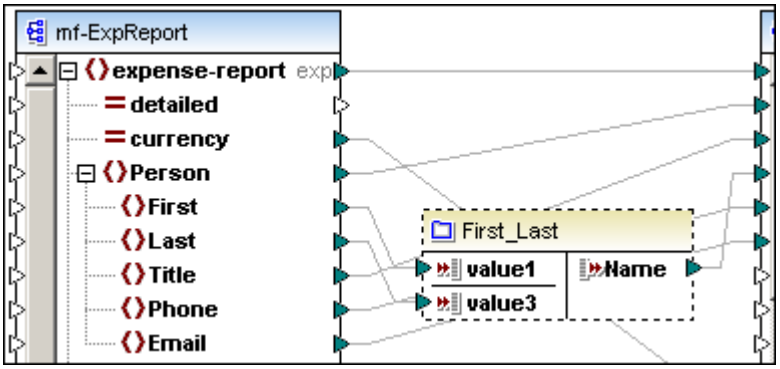
Die einzelnen Elemente, aus denen die Funktion besteht, werden auf einem Register mit dem Namen der Funktion angezeigt. Die neue Bibliothek "user" erscheint im Bibliotheksfenster mit dem Funktionsnamen "First_Last" unterhalb davon.

 © 2011 Altova GmbH



XSLT ausgewählt

Klicken Sie auf die Schaltfläche "Startseite" , um zum Hauptmapping-Fenster zurückzukehren. Die zwei Komponenten wurden nun zu einer einzigen Funktionskomponente mit dem Namen First_Last zusammengefasst. Die Input- und Output-Parameter wurden automatisch verbunden.



Benutzerdefinierte inline-Funktionen werden mit einer gestrichelten Umrandung angezeigt. Im Allgemeinen verhalten sich diese Funktionen genauso, als wären ihre einzelnen Bestandteile anstatt der Funktionskomponente eingefügt worden. Nähere Informationen dazu siehe "benutzerdefinierte inline-Funktionen".

Durch Ziehen des Funktionsnamens aus dem Bibliotheksfenster in das Mapping-Fenster können Sie die Funktion an jeder Stelle des aktuellen Mappings zu verwenden. Informationen dazu, wie Sie die Funktion in einem anderen Mapping verwenden finden Sie unter Wiederverwenden von benutzerdefinierten Funktionen.

So öffnen Sie eine benutzerdefinierte Funktion:

Wählen Sie eine der folgenden Methoden:

- Doppelklicken Sie auf die Titelleiste der benutzerdefinierten Funktionskomponente oder
- Doppelklicken Sie im Bibliotheksfenster auf die jeweilige benutzerdefinierte Funktion.

Daraufhin werden die Einzelkomponenten innerhalb der Funktion auf einem Register mit diesem Namen angezeigt. Klicken Sie auf die Schaltfläche "Startseite" um zum

Mapping zurückzukehren.

Navigieren in benutzerdefinierten Funktionen

Wenn Sie zwischen den verschiedenen Registern (oder den Registern der benutzerdefinierten Funktionen) in MapForce navigieren, wird automatisch ein Verlauf erstellt, sodass Sie durch Klicken auf die Schaltflächen "Vorwärts" und "Zurück" zwischen den Registern vorwärts- und zurücknavigieren können. Der Verlauf gilt für die gesamte Sitzung, d.h. Sie können auch zwischen mehreren MFD-Dateien wechseln.

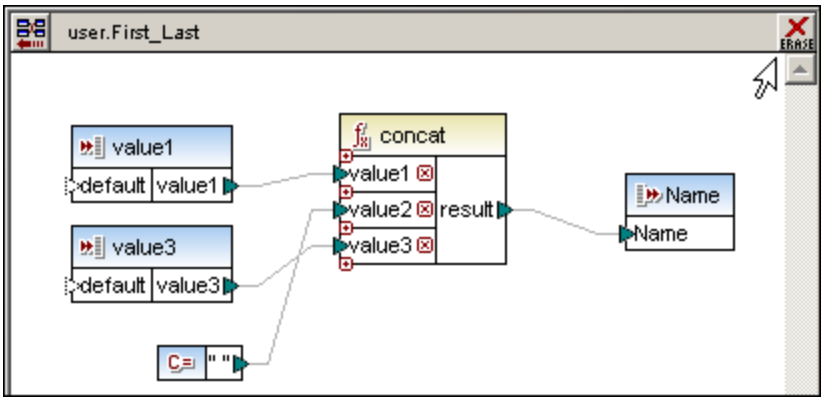
Über die Schaltfläche "Startseite" kehren Sie von der benutzerdefinierten Funktion aus zum Register des Hauptmappings zurück.

Über die Schaltfläche "Zurück" können Sie im Verlauf zurück navigieren

Über die Schaltfläche "Vorwärts" können Sie im Verlauf vorwärts navigieren

So löschen Sie eine benutzerdefinierte Funktion aus der Bibliothek:

1. Doppelklicken Sie auf eine bestimmte benutzerdefinierte Funktion im Bibliotheksfenster. Die benutzerdefinierte Funktion erscheint nun auf dem dazugehörigen Register.
2. Klicken Sie rechts oben in der Titelleiste auf die Schaltfläche **Erase,** um die Funktion zu löschen.



Wiederverwenden - Exportieren und Importieren von benutzerdefinierten Funktionen:

Benutzerdefinierte Funktionen, die in einem Mapping erstellt wurden, können in ein anderes Mapping importiert werden:

1. Klicken Sie am unteren Rand des Fensters "Bibliotheken" auf die Schaltfläche **Bibliotheken hinzufügen/entfernen,** klicken Sie auf die Schaltfläche "Hinzufügen" und wählen Sie eine ***.mfd-**Datei aus, die die gewünschte(n) benutzerdefinierte(n) Funktion(en) enthält.

   Die benutzerdefinierte Funktion wird nun im Bibliotheksfenster angezeigt (unter "user", wenn das die Standardbibliothek ist, die Sie ausgewählt haben). Sie können beim Definieren der benutzerdefinierten Funktion natürlich einen beliebigen Namen in das Feld "Bibliotheksname" eingeben.
2. Ziehen Sie die importierte Funktion in das Mapping, um sie zu verwenden.

   Bibliotheksnamen
   Anmerkung: Sie können dieselbe Bibliothek für benutzerdefinierte Funktionen in mehreren *.mfd Dateien und/oder benutzerdefinierten Bibliotheken verwenden (siehe

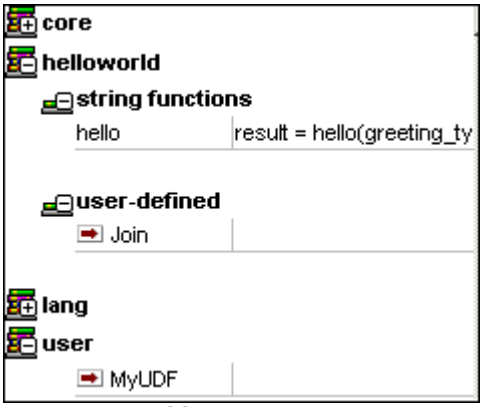
 © 2011 Altova GmbH

Abschnitt).

Funktionen aus allen verfügbaren Quellen scheinen im Fenster "Bibliotheken" unter demselben Bibliotheksnamen auf. Doch nur die Funktionen im derzeit aktiven Dokument können durch Doppelklicken bearbeitet werden.

Im folgenden Beispiel:

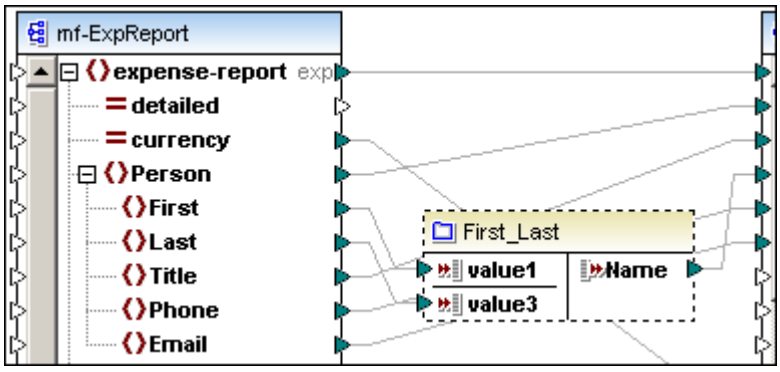
- wird die Funktion "hello" in der Bibliothek "helloworld" aus einer benutzerdefinierten *.mff-Datei importiert.
- ist die Funktion "Join" in der Bibliothek "helloworld" eine benutzerdefinierte Funktion, die in der aktuellen *.mfd-Datei definiert ist
- und die Funktion "MyUDF" in der Bibliothek "User" ist ebenfalls eine benutzerdefinierte Funktion, die in der aktuellen *.mfd-Datei definiert ist.



Java ausgewählt

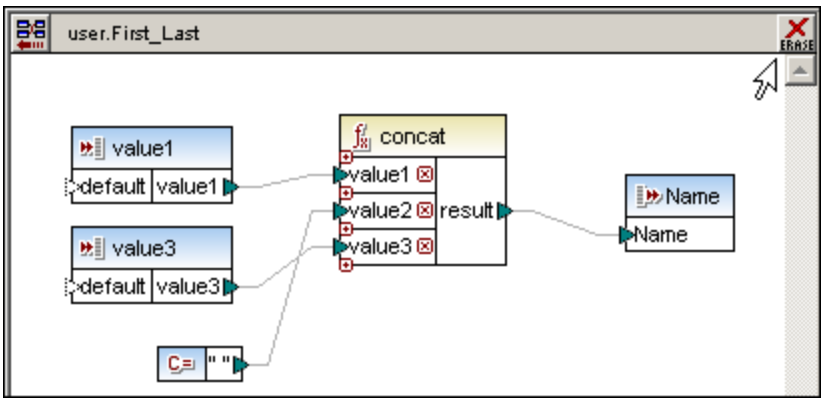
Beachten Sie, dass eventuelle Änderungen an importierten Funktionen auf importierte Mappings angewendet werden, wenn Sie die MFD-Bibliothekdatei speichern.

Parameterreihenfolge in benutzerdefinierte Funktionen



Sie können nun auch die Parameterreihenfolge innerhalb von benutzerdefinierten Funktionen direkt beeinflussen:

- Input- und Output-Parameter werden von oben nach unten (von der linken oberen Ecke der Parameterkomponente) nach ihrer Position sortiert .
- Wenn zwei Parameter dieselbe vertikale Position haben, hat der weiter links gelegene Vorrang.
- Fall zwei Parameter ungewöhnlicherweise genau die gleiche Position haben, wird automatisch die interne Komponenten-ID verwendet.



Anmerkungen:

- Die Positionierung der Komponente und die Anpassung der Größe der Komponente können nun rückgängig gemacht werden.
- Neu hinzugefügte Input- oder Output-Komponenten werden unterhalb der letzten Input- oder Output-Komponente erstellt.
- Komplexe und einfache Parameter können gemischt verwendet werden. Die Reihenfolge der Parameter wird anhand der Position der Komponente ermittelt.

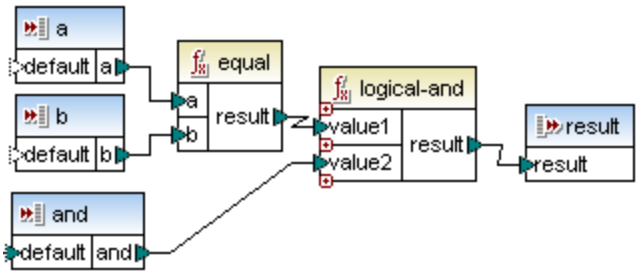
© 2011 Altova GmbH

15.1.1 Funktionsparameter

Funktionsparameter werden innerhalb einer benutzerdefinierten Funktion durch **Input-** und **Output-Komponenten** dargestellt.

Input-Komponente/Parameter: **a, b and**

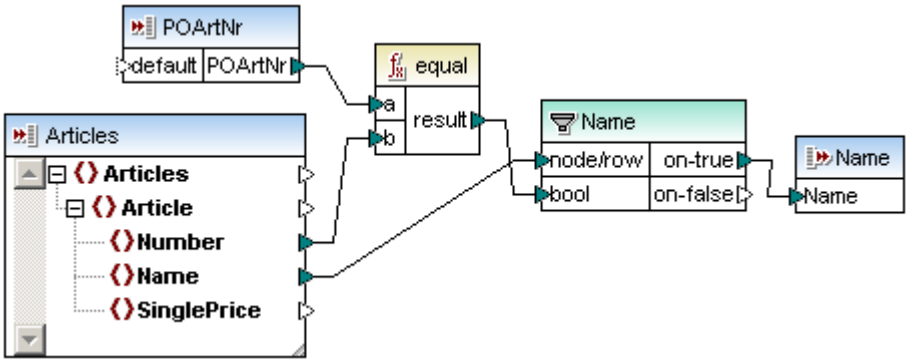
Output-Komponenten/Parameter: **result**



Input-Parameter dienen dazu, Informationen vom Hauptmapping an die benutzerdefinierte Funktion zu übergeben, während Output-Parameter dazu dienen, Informationen an das Hauptmapping zurückzugeben.

Einfache und komplexe Parameter
Die Input- und Output-Parameter von benutzerdefinierten Funktionen können eine beliebige Kombination aus folgenden Komponenten sein:

- einfachen Werten, z.B. Strings oder Ganzzahlen
- komplexen Node-Strukturen, z.B. einem XML-Element mit seinen Attributen und Child Nodes



Der Input-Parameter **POArtNr** ist ein einfacher Wert des Datentyps "string"

Der Input-Parameter **Articles** ist eine komplexe XML-Dokument Node-Struktur

Der Output-Parameter **Name** ist ein einfacher Wert vom Typ String

Anmerkung:
Die oben gezeigten benutzerdefinierten Funktionen stehen alle in der Datei **PersonListByBranchOffice.mfd** im Ordner ...\MapForceExamples zur Verfügung.

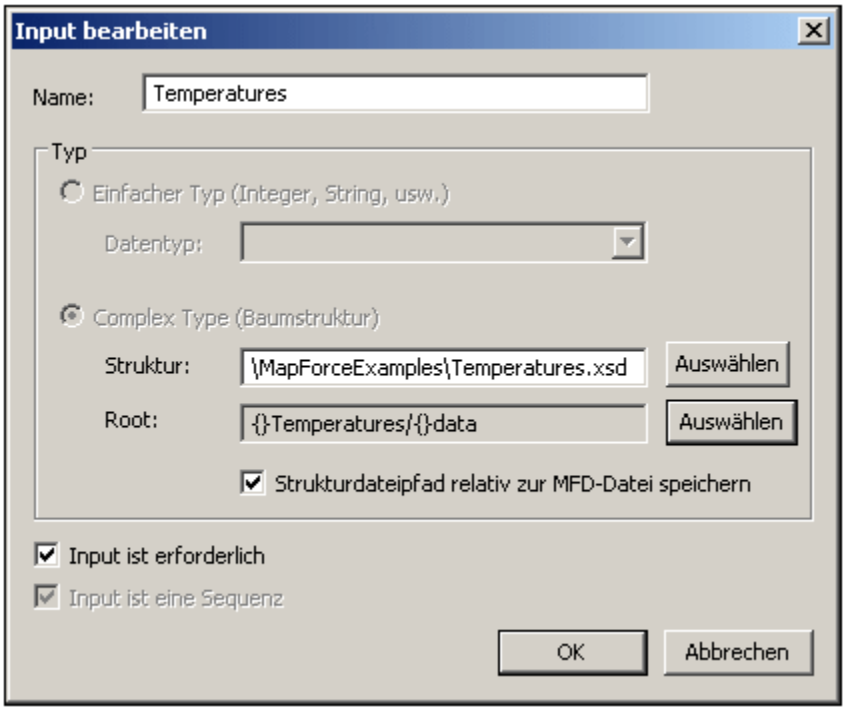
Sequenzen

Sequenzen sind Quelldaten, die aus einem Bereich oder einer Sequenz von Werten bestehen. Im Dialogfeld "Komponenteneigenschaften" können einfache und komplexe benutzerdefinierte

Parameter (Input/Output) als Sequenzen definiert werden.

Bei Aggregatfunktionen wie z.B. min, max, avg, usw. wird diese Art von Input verwendet, um einen einzigen bestimmten Wert aus der Input-Sequenz bereitzustellen. Nähere Informationen zu Aggregatfunktionen finden Sie unter Aggregatfunktionen: min, max, sum, count, avg.

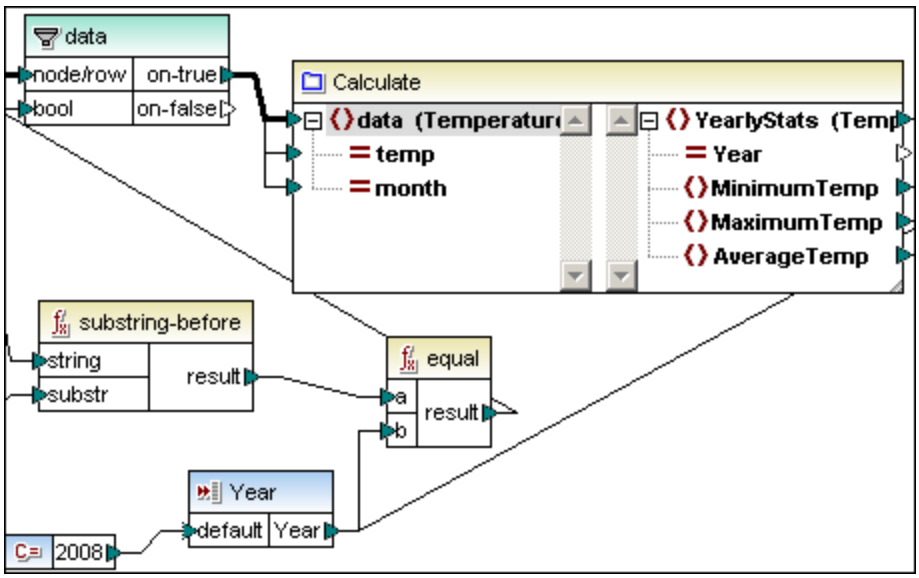
Wenn das Kontrollkästchen "**Input ist eine Sequenz**" aktiv ist, so behandelt die Komponente den Input als Sequenz. Wenn das Kontrollkästchen deaktiviert ist, wird der Input als Einzelwert behandelt.



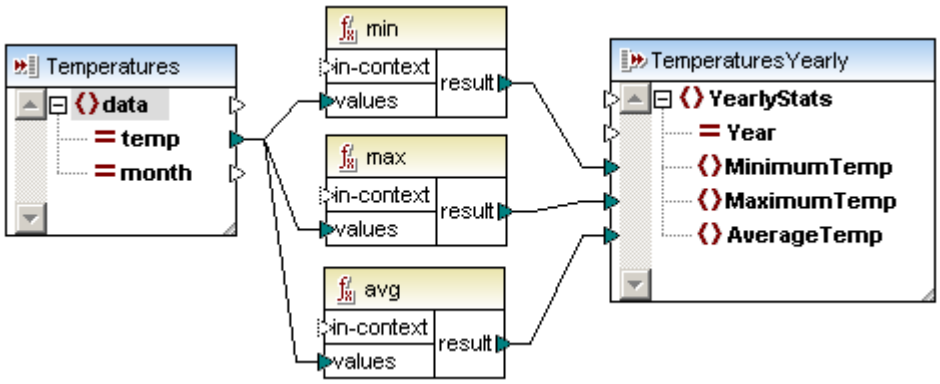
Mit dieser Unterscheidung zwischen Sequenz oder nicht Sequenz wird festgelegt, **wie oft** die Funktion aufgerufen wird.

- Bei Verbindung mit einem **Sequenz**-Parameter wird die benutzerdefinierte Funktion **nur einmal** aufgerufen und die komplette Sequenz wird an die benutzerdefinierte Funktion übergeben.

 © 2011 Altova GmbH



In der Abbildung sehen Sie die benutzerdefinierte Funktion "Calculate" des Mappings **InputIsSequence.mfd**" aus dem Ordner ...\MapForceExamples. Die Input-Komponente "Temperatures" ist, wie im obigen Dialogfeld gezeigt, als Sequenz definiert.



- Wenn Input-Daten mit einem Parameter verbunden sind, bei dem es sich **nicht** um eine Sequenz handelt, wird die Funktion für jedes einzelne Datenelement in der Sequenz **einmal** aufgerufen.

Bitte beachten Sie:
Die Sequenzeinstellung von Input/Output-Parametern wird ignoriert, wenn die Funktion vom Typ inline ist.

Wenn Sie **eine leere Sequenz** mit einem **Parameter**, verbinden, bei dem es sich um **keine Sequenz** handelt, hat dies zur Folge, dass die Funktion **gar nicht aufgerufen wird**!

Dies kann passieren, wenn die Quellstruktur **optionale** Datenelemente enthält oder wenn eine Filterbedingung keine passenden Datenelemente zurückgibt. Um dies zu vermeiden und sicherzustellen, dass die Sequenz nie leer ist, verwenden Sie entweder vor dem Input der Funktion die substitute-missing Funktion oder setzen Sie den Parameter auf sequence und fügen Sie innerhalb der Funktion Behandlungsroutinen für die leere Sequenz hinzu.

Wenn eine Funktion eine Sequenz mehrerer Werte an ihre Output-Komponente übergibt und die Output-Komponente nicht auf Sequenz gesetzt wurde, wird bei Aufruf der Funktion nur das

erste Ergebnis aufgerufen.

© 2011 Altova GmbH

15.1.2 Inline-Funktionen und reguläre benutzerdefinierte Funktionen

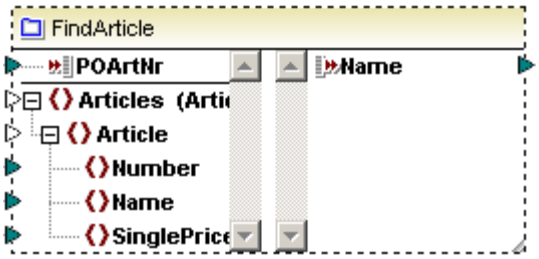
Inline-Funktionen unterscheiden sich in der Art, wie sie bei der Generierung von Code implementiert werden, grundlegend von nicht-inline-Funktionen.

- Der Code für Funktionen vom Typ **inline** wird überall dort eingefügt, wo die benutzerdefinierten Funktionen aufgerufen/verwendet werden
- Der Code einer **regulären** Funktion wird als Funktionsaufruf implementiert.

Daher verhalten sich inline-Funktionen auf dieselbe Art, als wären sie durch ihre Implementierung ersetzt worden. Dadurch eignen sie sich ideal dazu, ein komplexes Mapping in mehrere separate Teile aufzuteilen.

Bitte beachten Sie:
Durch Verwendung von **Inline**-Funktionen kann der generierte Programmcode erheblich länger werden! Der Code der benutzerdefinierten Funktion wird an allen Stellen, an denen die Funktion aufgerufen/verwendet wird, eingefügt, wodurch der Code - im Gegensatz zur Verwendung einer normalen Funktion - erheblich länger wird.

Benutzerdefinierte **INLINE**-Funktionen werden durch eine **strichlierte** Umrandung dargestellt.



Benutzerdefinierte **Inline-**Funktionen unterstützen:

- **mehrere Output-Komponenten** innerhalb einer Funktion
-

Benutzerdefinierte **Inline**-Funktionen unterstützen Folgendes **nicht**:

- die Festlegung eines Prioritätskontexts bei einem Parameter.
- Rekursive Aufrufe benutzerdefinierter inline-Funktionen

REGULÄRE benutzerdefinierte Funktionen, d.h. Funktionen, die nicht inline gesetzt wurden, werden durch eine **durchgezogene** Umrandung dargestellt

Benutzerdefinierte **reguläre** Funktionen (nicht inline gesetzte) **unterstützen**:

- lediglich eine **einzige Output-Komponente** innerhalb einer Funktion
- rekursive Aufrufe (wobei die exit-Bedingung bereitgestellt werden muss. Verwenden Sie z.B. eine if-Else-Bedingung, wenn ein Branch oder Wert die Rekursion beendet)
- das Definieren eines Prioritätskontexts bei einem Parameter

Bitte beachten Sie:
Zwar unterstützen reguläre Funktionen **nicht** mehrere Output-Komponenten, doch können sie

in dieser Funktionsart **erstellt werden**. Es erscheint allerdings eine Fehlermeldung, wenn Sie versuchen, Code zu generieren oder eine Vorschau des Ergebnisses dieses Mappings anzuzeigen.

Wenn Sie nicht mit Rekursionen in Ihren Funktionen arbeiten, können Sie die Art der Funktion in "inlined" ändern.

Benutzerdefinierte reguläre Funktionen (nicht inline gesetzte Funktionen) **unterstützen Folgendes n**icht:

- die direkte Verbindung von Filtern mit einfachen **Input**-Komponenten, bei denen es sich nicht um eine Sequenz handelt
- Sequenz- oder Aggregatfunktionen zu einfachen Input-Komponenten (wie like exists, substitute-missing, sum, group-by, ...)

Codegenerierung:

Die Implementierung einer regulären benutzerdefinierten Funktion wird nur einmal als aufrufbare XSLT-Vorlage oder Funktion generiert. Jede benutzerdefinierte Funktion generiert Code für einen Funktionsaufruf, bei dem Inputs als Parameter übergeben werden und der Output der Rückgabewert der Funktionskomponente ist.

Die Input-Parameter werden zuerst in Laufzeit ausgewertet, anschließend wird die Funktion jedesmal aufgerufen, wenn die Input-Daten vorkommen. Nähere Informationen dazu finden Sie unter Funktionsparameter.

So ändern Sie die benutzerdefinierte Funktion "type":

1. Doppelklicken Sie auf die benutzerdefinierte Funktion, um deren Komponenten zu sehen.
2. Wählen Sie die Menüoption **Funktion | Funktionseinstellungen** und aktivieren Sie das Kontrollkästchen "Inline Verwendung".

Benutzerdefinierte Funktionen und "Alles kopieren" Verbindungen

Beim Erstellen von "Alles kopieren" Verbindungen zwischen einem Schema und einem komplexen benutzerdefinierten Funktionsparameter müssen die beiden Komponenten auf demselben Schema basieren! Sie müssen jedoch nicht notwendigerweise dasselbe Root-Element haben. Ein Beispiel dazu finden Sie unter "Definieren komplexer Output-Komponenten".

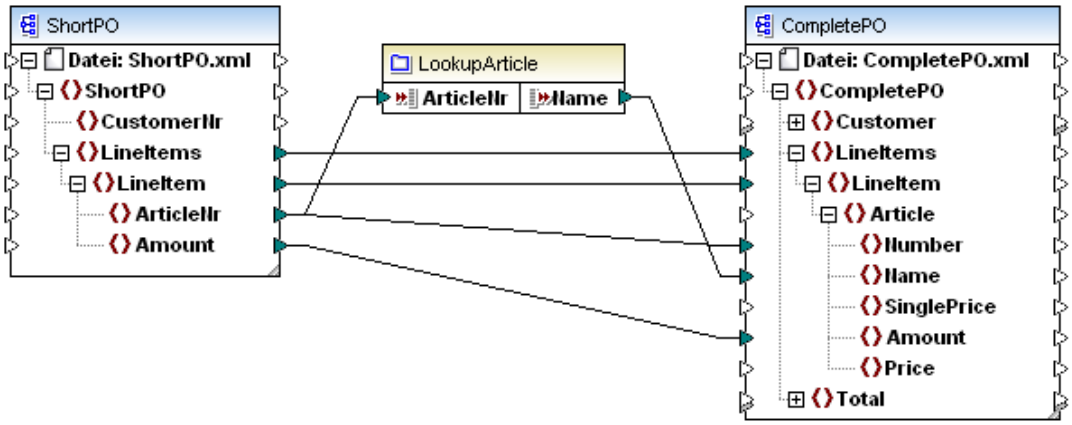
© 2011 Altova GmbH

15.1.3 Erstellen einer einfachen Lookup-Funktion

In diesem Beispiel werden die Komponenten aus der Datei **lookup-standard.mfd** verwendet, die Sie im Ordner **...\MapForceExamples** finden.

Ziel:
Die Erstellung einer generischen Abfragefunktion, die:

- die Artikel-/Nummerndaten aus der XML-Artikeldatei liefert, die mit den Artikelnummern einer anderen XML-Datei - in diesem Fall ShortPO - verglichen werden.

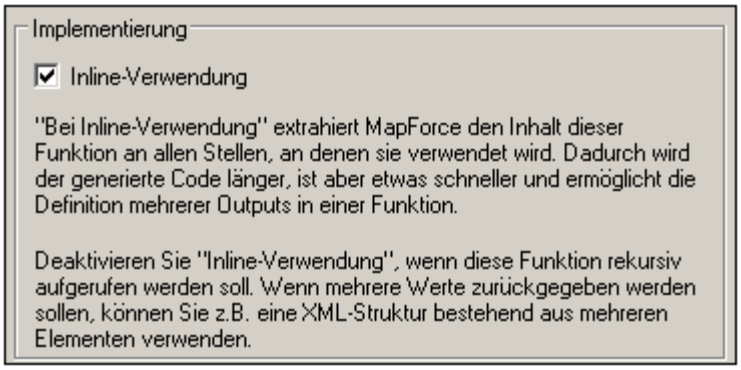


- Fügen Sie die Datei ShortPO.xml ein und weisen Sie dieser ShortPO.xml als XML-Quelldatei zu.
- Fügen Sie die Schema-Datei CompletePO.xsd ein und wählen Sie als Root-Element CompletePO.
- Fügen Sie auf die unten beschriebene Weise eine neue benutzerdefinierte Funktion ein.

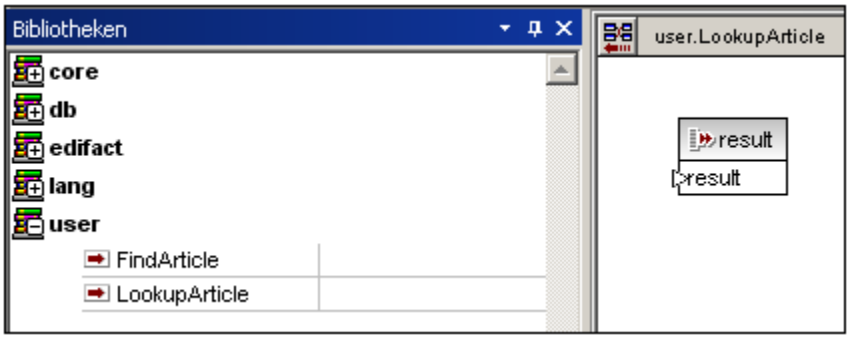
So erstellen Sie eine benutzerdefinierte Funktion von Grund auf neu:

1. Wählen Sie die Menüoption **Funktion | Benutzerdefinierte Funktion erstellen**.
2. Geben Sie einen Namen für die Funktion ein, z.B. LookupArticle.

3. Deaktivieren Sie das Optionsfeld "Inline Verwendung" und bestätigen Sie mit OK.



Daraufhin wird eine Registerkarte angezeigt, die nur ein Element, nämlich eine Ausgabefunktion mit dem Namen "result", enthält.



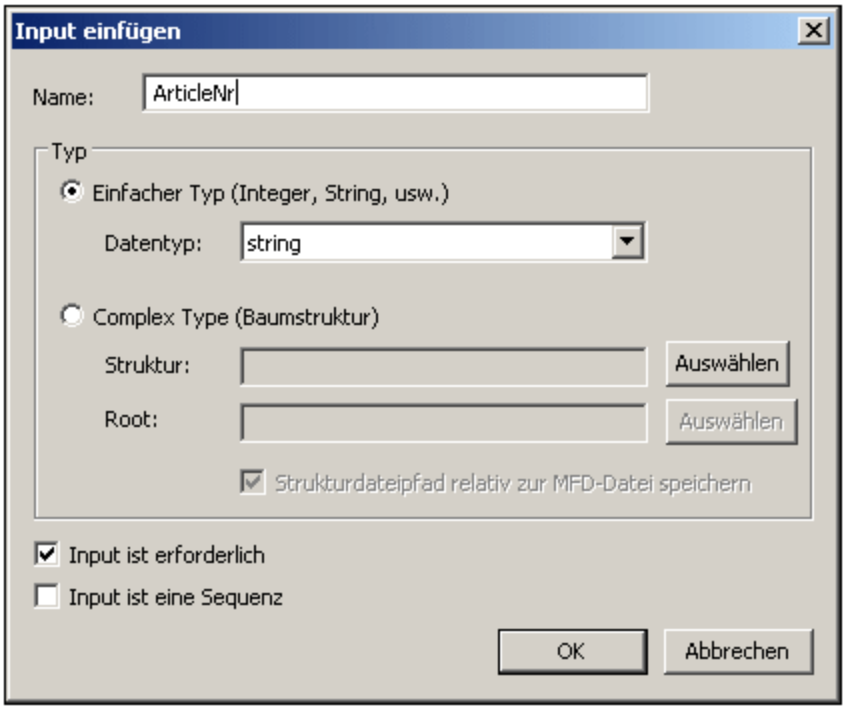
Java ausgewählt

Dies ist der Arbeitsbereich, in dem die benutzerdefinierte Funktion definiert wird.

Im Bibliotheksfenster wurde eine neue Bibliothek mit dem Namen "user" angelegt, die die Funktion mit den Titel "LookupArticle" enthält.

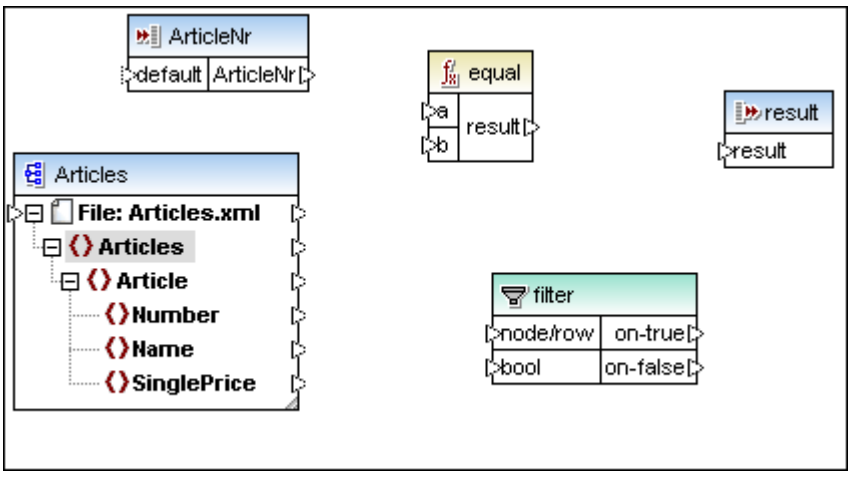
4. Klicken Sie auf das Symbol **XML-Schema/Datei einfügen** , um das Schema **Articles** einzufügen und wählen Sie die XML-Datei desselben Namens als Datenquelle aus.
5. Klicken Sie auf das Symbol **Input-Komponente einfügen** , um eine Input-Komponente einzufügen.
6. Geben Sie den Namen des Eingabeparameters ein, in diesem Fall ArticleNr und klicken Sie auf OK.

 © 2011 Altova GmbH



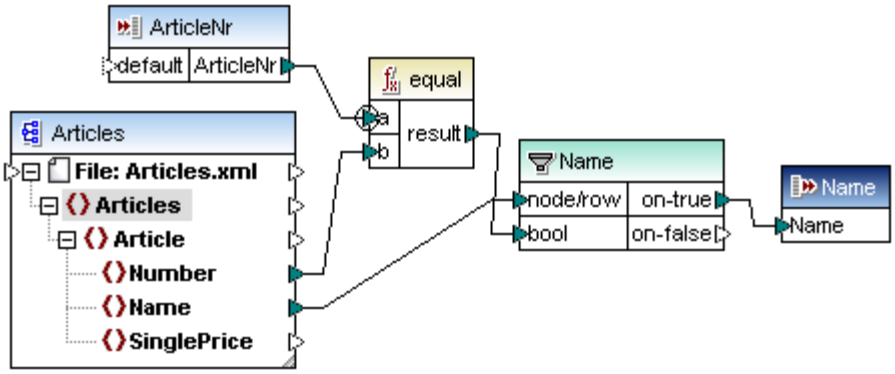
Diese Komponente dient als Dateninput für die benutzerdefinierte Funktion und stellt das Input-Symbol der benutzerdefinierten Funktion bereit.
Sie können den Datentyp des Eingabeparameters ändern und definieren, ob die Funktion ein Input-Symbol in diesem Dialogfeld haben soll (Verbindung erforderlich).

7. Fügen Sie eine "**equal**"-Komponente ein, indem Sie sie aus der Gruppe der logical functions der core-Bibliothek auf das Register ziehen.
8. Fügen Sie eine "**filter**"-Komponente ein. Klicken Sie dazu auf das Filter-Symbol in der Symbolleiste.



Nehmen Sie beim Erstellen der Mappings in der benutzerdefinierten Funktion das unten stehende Diagramm zu Hilfe und beachten Sie bitte Folgendes:

9. Klicken Sie mit der rechten Maustaste auf den Parameter **a** und wählen Sie im Popup-Menü den Eintrag **Priorität - Kontext** wählen.
10. **Doppelklicken** Sie auf die Ausgabefunktion und geben Sie den Namen des Ausgabeparameters ein, in diesem Fall "**Name**".



Damit ist die Definition der benutzerdefinierten Funktion abgeschlossen.

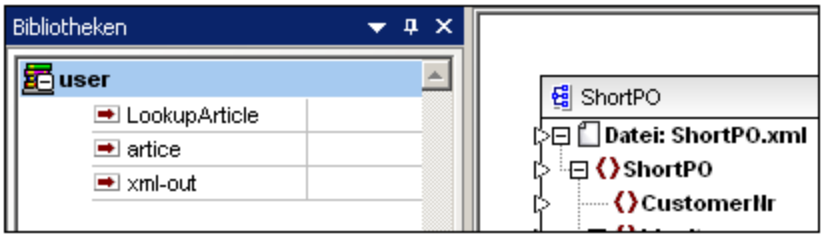
Bitte beachten Sie:

Wenn Sie auf die Eingabe- und die Ausgabefunktion doppelklicken, wird ein Dialogfeld geöffnet, in dem Sie den Namen und Datentyp des Input-Parameters ändern und definieren können, ob die Funktion ein Input-Symbol (Input erforderlich) haben soll und ob sie außerdem als Sequenz definiert werden soll.

Diese benutzerdefinierte Funktion:

- hat eine **Eingabefunktion,** ArticleNr, die Daten von der XML-Datei ShortPO erhält.
- **vergleicht** die Artikelnummer (ArticleNr) aus der Datei ShortPO mit dem Artikel/der Nummer aus der XML-Instanzdatei **Articles**, die zu diesem Zweck in die benutzerdefinierte Funktion eingefügt wurde.
- verwendet eine **filter**-Komponente, um die Datensätze Article/Name an die Ausgabekomponente zu übergeben, wenn das Ergebnis des Vergleichs "true" ist.
- hat eine Ausgabefunktion, Name, die die Datensätze mit den Artikelnamen an die XML-Datei CompletePO übergibt.

8. Klicken Sie auf das Symbol "Startseite" , um zum Haupt-Mapping zurückzukehren. Die benutzerdefinierte Funktion "LookupArticle" steht nun in der user-Bibliothek zur Verfügung.



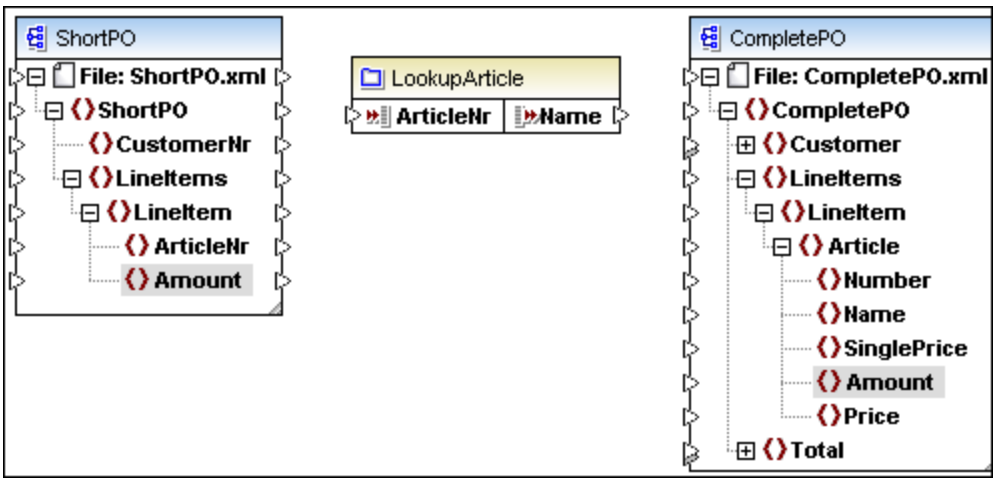
Java ausgewählt

9. Ziehen Sie die Funktion LookupArticle in das Mapping-Fenster.

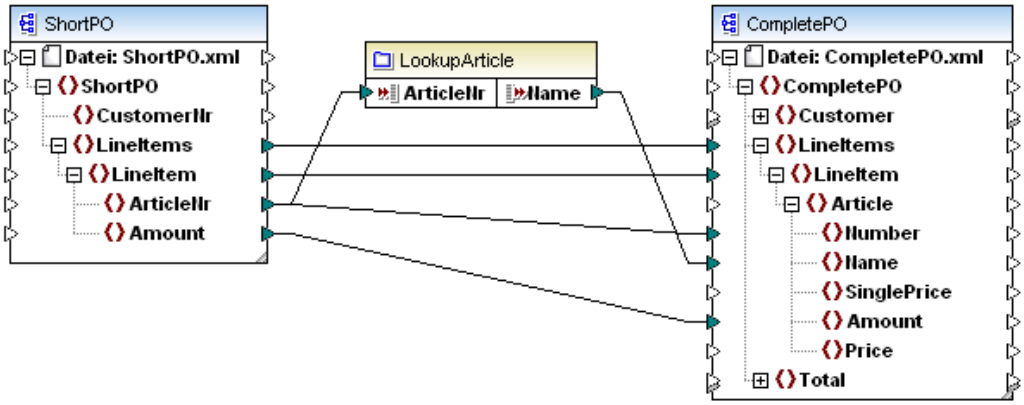
Die benutzerdefinierte Funktion wird angezeigt:

- Ihr Name "LookupArticle" erscheint in der Funktionstitelleiste,
- das Input- und das Output-Symbol sind mit einem Namen versehen.

 © 2011 Altova GmbH



10. Erstellen Sie die in der Abbildung unten gezeigten Verbindungen und klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis des Mappings zu sehen.



15.1.4 Komplexe benutzerdefinierte Funktion - XML-Node als Input

Für dieses Beispiel wird die Datei **lookup-udf-in.mfd** aus dem Ordner **...\MapForceExamples** verwendet. In diesem Abschnitt wird beschrieben, wie man benutzerdefinierte Inline-Funktionen definiert, die komplexe Input- und Output-Komponenten enthalten.

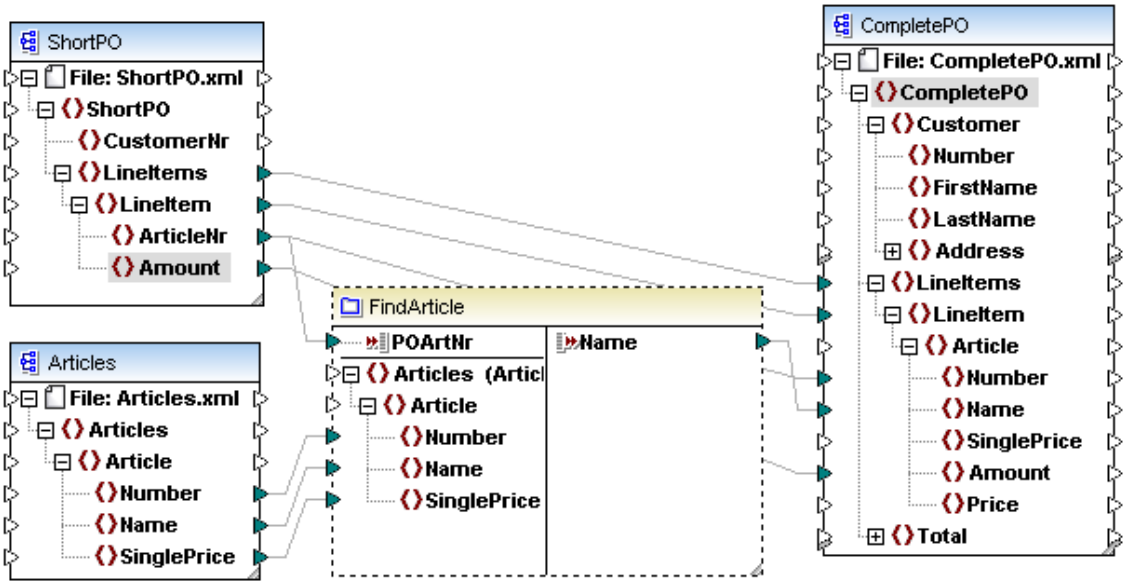
Beachten Sie, dass die benutzerdefinierte Funktion FindArticle aus zwei Hälften besteht:

einer linken Hälfte, die die Input-Parameter enthält:

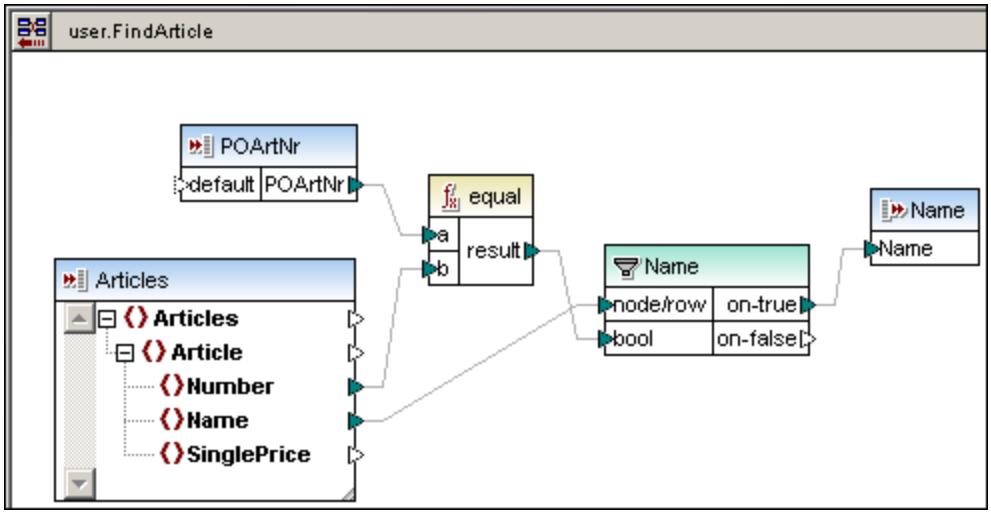
- einen einfachen Input-Parameter **POArtNr**
- eine komplexe Input-Komponente **Articles** mit Mappings direkt zu ihren XML Child Nodes

eine rechte Hälfe mit folgendem Inhalt:

- einen einfachen Output-Parameter namens "**Name**".



Im Screenshot unten sehen Sie die Komponenten, aus denen sich die benutzerdefinierte Funktion zusammensetzt: den beiden Input-Komponenten auf der linken Seite und der Output-Komponente auf der rechten Seite.

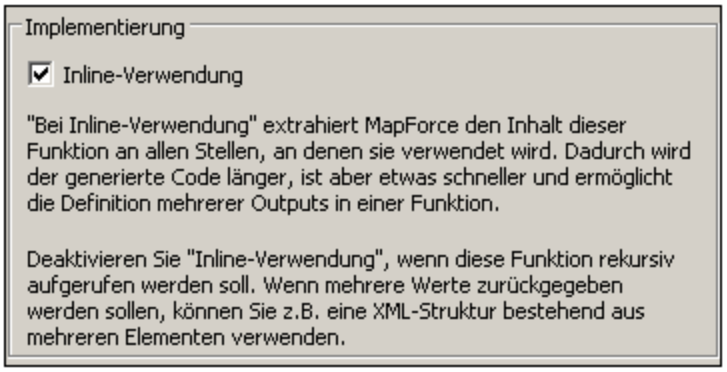


 © 2011 Altova GmbH

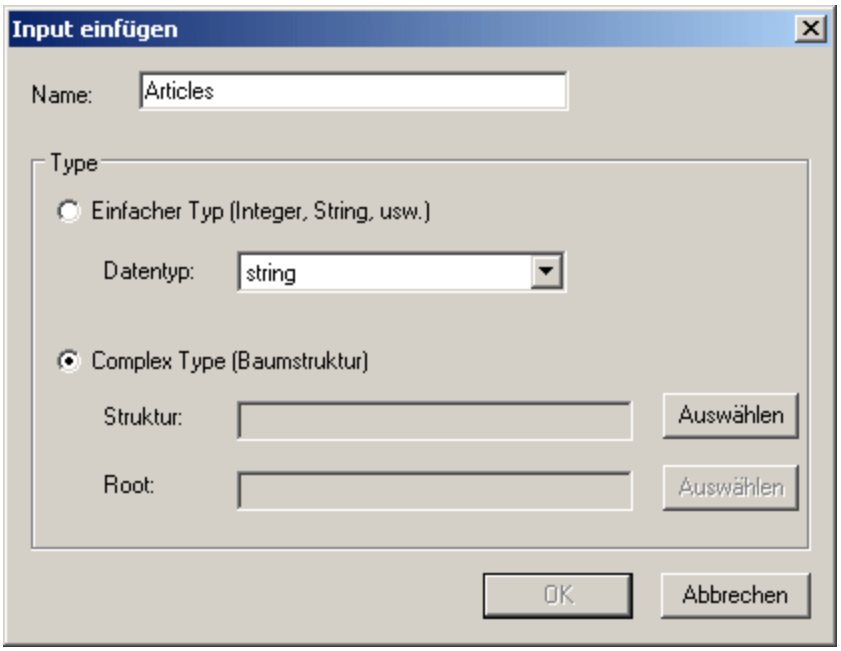
Definieren komplexer Input-Komponenten

Definieren komplexer Input-Komponenten:

1. Erstellen Sie wie gewohnt, also durch Auswahl von **Funktion | Benutzerdefinierte Funktion erstellen** eine benutzerdefinierte Funktion und klicken Sie auf OK. Beachten Sie, dass automatisch das Kontrollkästchen **Inline Verwendung** ausgewählt wird.



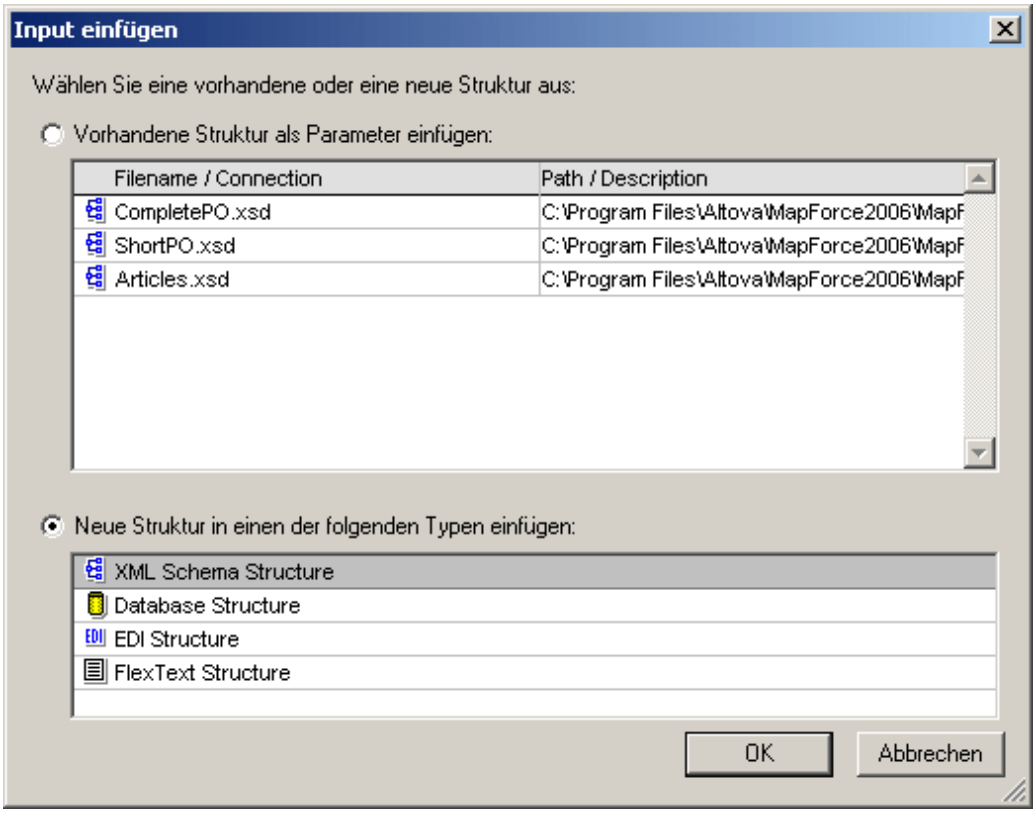
2. Klicken Sie in der Symbolleiste auf das Symbol **Input-Komponente einfügen** .
3. Geben Sie den Namen der Input-Komponente in das Feld "Name" ein.



4. Klicken Sie auf das Optionsfeld **Complex Type (Baumstruktur)** und anschließend auf die Schaltfläche "Auswählen" neben dem Feld "Struktur". Daraufhin wird ein weiteres Dialogfeld geöffnet.

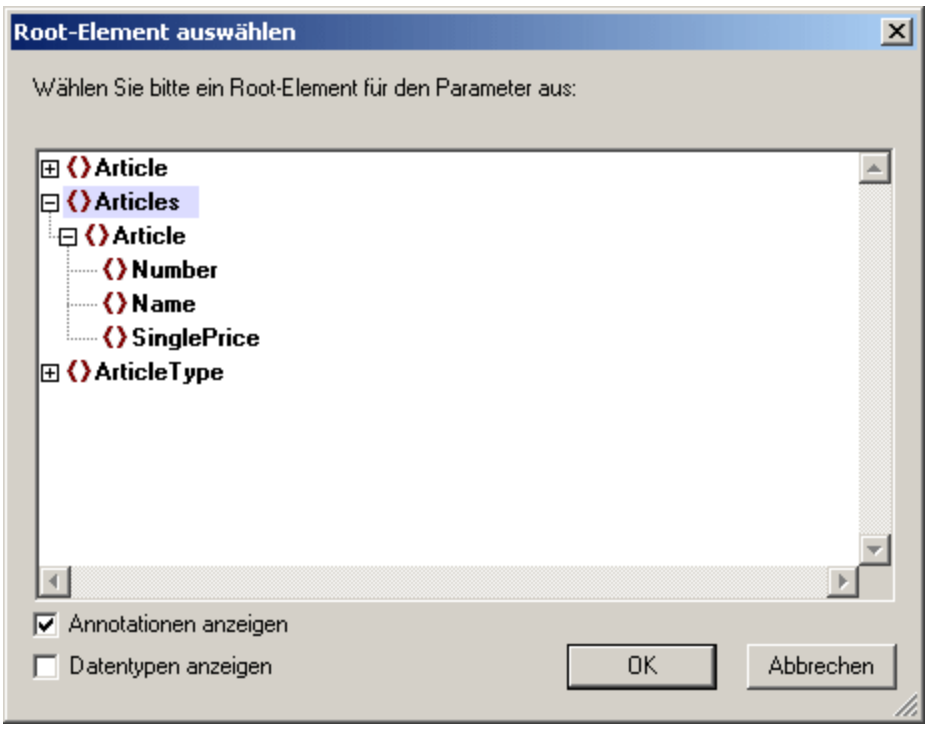
Im oberen Listenfeld sehen Sie die **vorhandenen** Komponenten im Mapping (drei Schemas, wenn Sie das Beispiel-Mapping geöffnet haben). Beachten Sie, dass diese Liste alle Komponenten enthält, die in das aktive Mapping eingefügt wurden: z.B. XML-Schemas.

Im unteren Listenfeld können Sie eine neue komplexe Datenstruktur auswählen, d.h. XML-Schema.

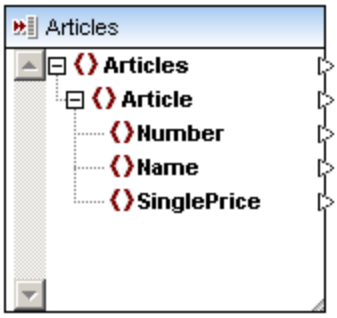


5. Klicken Sie auf das Optionsfeld "Neue Struktur einfügen...", wählen Sie den Eintrag "XML-Schema-Struktur" aus und klicken Sie zum Fortfahren auf OK.
6. Wählen Sie im Dialogfeld "Öffnen" die Datei **Articles.xsd** aus.
7. Klicken Sie auf das Element, das als Root-Element in der Komponente verwendet werden soll, z.B. Articles und bestätigen Sie die Auswahl durch Klicken auf OK. Klicken Sie anschließend nochmals auf OK, um beide Dialogfelder zu schließen.

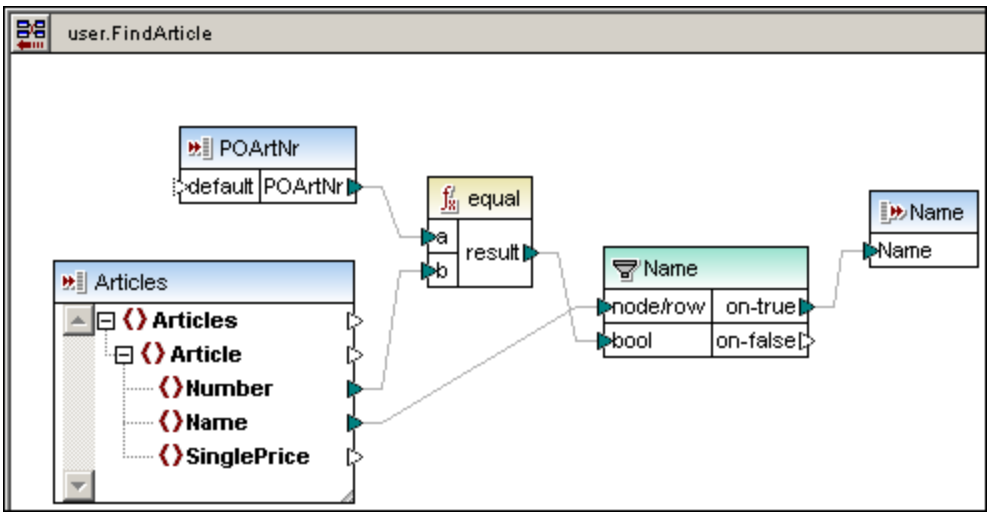
© 2011 Altova GmbH



Die Komponente "Articles" wird in die benutzerdefinierte Funktion eingefügt. Beachten Sie bitte das Input-Symbol links vom Komponentennamen. Dieses Symbol zeigt an, dass die Komponente als komplexe Input-Komponente verwendet wird.



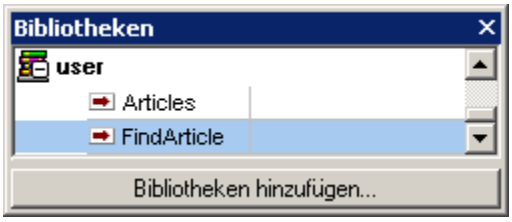
8. Fügen Sie die restlichen Komponenten, nämlich eine zweite "einfache" Input-Komponente (namens POArtNr), Filter-, equal- und eine Output-Komponente (namens "Name") ein, wie im Screenshot unten gezeigt, und verbinden Sie diese wie gezeigt.



Bitte beachten Sie:

- Die Input-Komponente "Articles" erhält ihre Daten von **außerhalb** der benutzerdefinierten Funktion. Hier stehen Input-Symbole zur Verfügung, die ein Mapping auf diese Komponente gestatten.
- Sie können einer komplexen Input-Komponente keine **XML-Instanzdatei** zuweisen, um Daten aus der benutzerdefinierten Funktion bereitzustellen.
- Die andere Input-Komponente (POArtNr) liefert die Daten für die ShortPO-Artikelnummer, mit der die Elemente **Article | Number** verglichen werden.
- Die Filterkomponente filtert jene Datensätze, in denen beide Nummern identisch sind, und übergibt diese an die Output-Komponente.

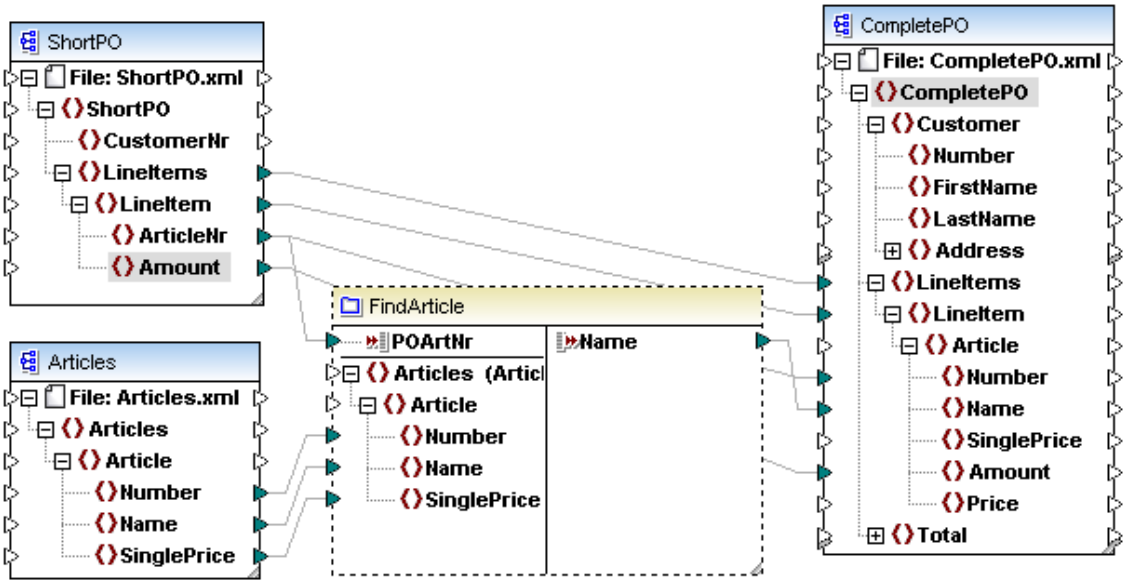
10. Klicken Sie auf das Startseitensymbol um zum Mapping zurückzukehren.
11. Ziehen Sie die neu erstellte benutzerdefinierte Komponente aus dem Bibliotheksfenster in das Mapping.



Java ausgewählt

12. Erstellen Sie die Verbindungen wie im Screenshot unten gezeigt.

 © 2011 Altova GmbH



Die linke Hälfte enthält die Input-Parameter, auf die Datenelemente aus zwei Schema-/XML-Dateien gemappt werden:

- ShortPO liefert die Daten für die Input-Komponente **POArtNr**
- Articles stellt die Daten für die komplexe Input-Komponente bereit. Die Instanzdatei Articles.xml wurde der Schema-Datei Articles beim Einfügen dieser Datei zugewiesen.
- Die komplexe Input-Komponente **Articles** mit ihren XML Child Nodes, auf die Daten aus der Komponente "Articles" gemappt wurden.

Die rechte Hälfte enthält:

- einen einfachen Output-Parameter mit der Bezeichnung "**Name"**, der die gefilterten "Line Items", die dieselbe Artikelnummer haben, an das Datenelement "Name" in der Komponenten "CompletePO" übergibt.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <CompletePO xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3    <LineItems>
4      <LineItem>
5        <Article>
6          <Number>3</Number>
7          <Name>Pants</Name>
8          <Amount>5</Amount>
9        </Article>
10     </LineItem>
11     <LineItem>
12       <Article>
13         <Number>1</Number>
14         <Name>T-Shirt</Name>
15         <Amount>17</Amount>
16       </Article>
17     </LineItem>
18   </LineItems>
19 </CompletePO>
```

Bitte beachten Sie:
Beim Erstellen von "Alles kopieren" Verbindungen zwischen einem Schema und einem benutzerdefinierten Funktionsparameter müssen die beiden Komponenten auf demselben Schema basieren! Sie müssen jedoch nicht unbedingt dasselbe Root-Element haben.

15.1.5 Komplexe benutzerdefinierte Funktion - XML Node als Output

Für dieses Beispiel wird die Datei **lookup-udf-out.mfd** aus dem Ordner **...\MapForceExamples** verwendet. In diesem Abschnitt wird beschrieben, wie man benutzerdefinierte Inline-Funktionen definiert, die komplexe Input- und Output-Komponenten möglich machen.

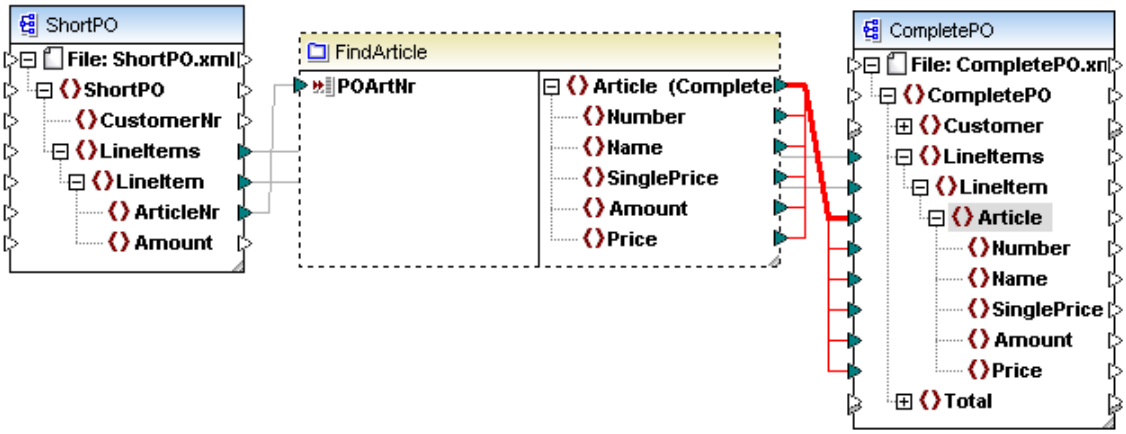
Beachten Sie, dass die benutzerdefinierte Funktion FindArticle aus zwei Hälften besteht:

einer linken Hälfte, die den Input-Parameter enthält:

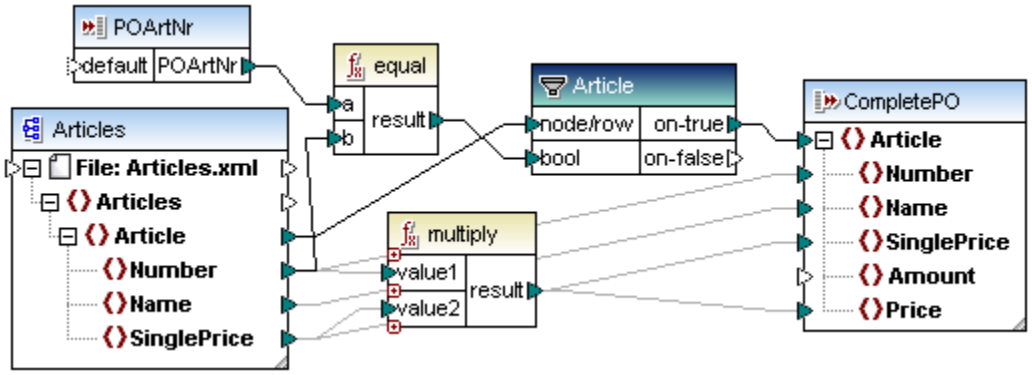
- einen einfachen Input-Parameter **POArtNr**

eine rechte Hälfe mit folgendem Inhalt:

- eine komplexe Output-Komponente namens **Article (CompletePO),** deren XML Child Nodes auf CompletePO gemappt sind.



Im Screenshot unten sehen Sie die Komponenten, aus denen sich die benutzerdefinierte Funktion zusammensetzt: den beiden Input-Komponenten auf der linken Seite und der Output-Komponente auf der rechten Seite.

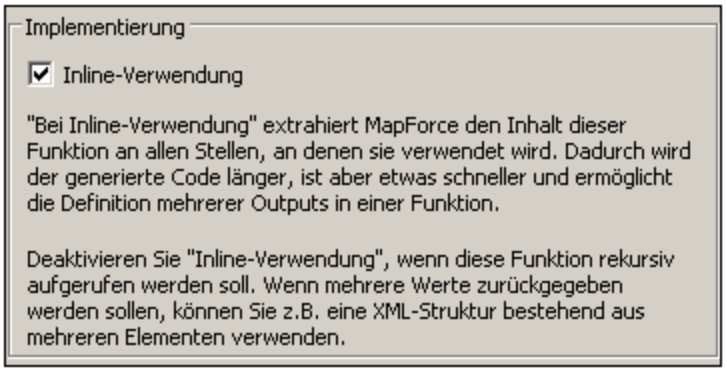


Definieren komplexer Output-Komponenten

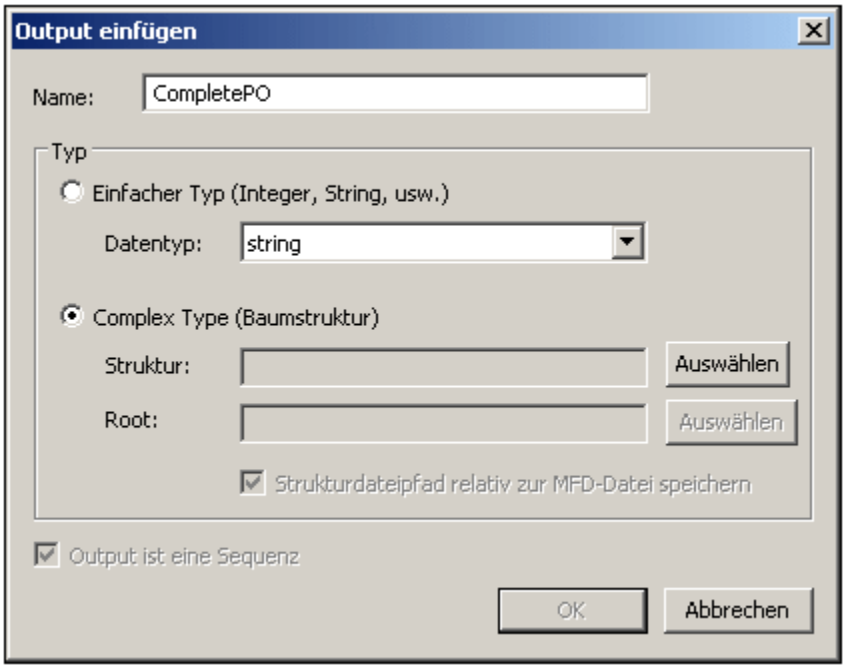
Definieren komplexer Output-Komponenten:

1. Erstellen Sie wie gewohnt, also durch Auswahl von **Funktion | Benutzerdefinierte Funktion erstellen** eine benutzerdefinierte Funktion, geben Sie ihr den Namen **FindArticle** und bestätigen Sie den Vorgang durch Klicken auf OK. Beachten Sie, dass automatisch die Option **Inline...** ausgewählt wird.

 © 2011 Altova GmbH



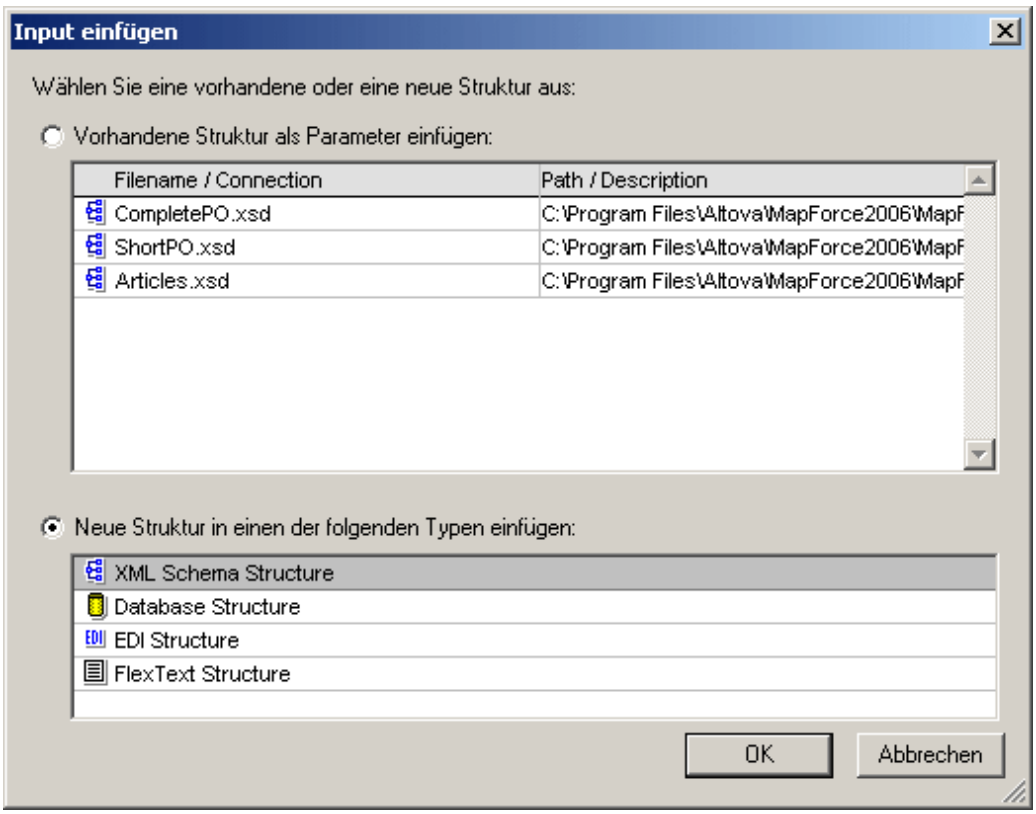
2. Klicken Sie auf das Symbol "Output-Komponente einfügen" und geben Sie einen Namen ein, z.B. CompletePO.



3. Klicken Sie auf das Optionsfeld **Complex type...** und anschließend auf die Schaltfläche "Auswählen".
   Daraufhin wird ein weiteres Dialogfeld geöffnet.

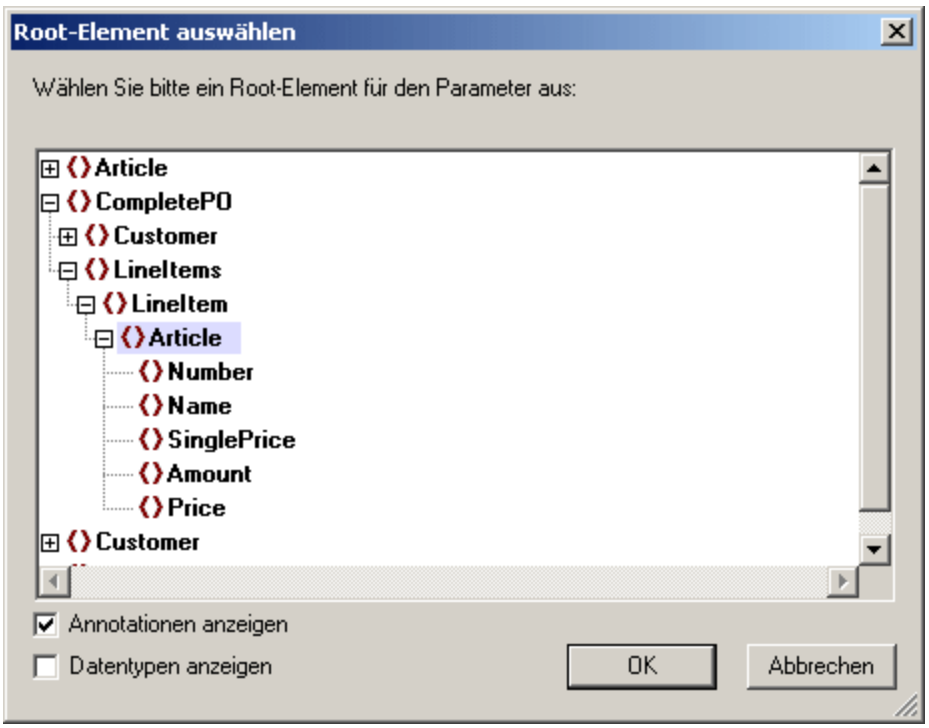
   Im oberen Listenfeld sehen Sie die **vorhandenen** Komponenten im Mapping, (drei Schemas, wenn Sie die Beispieldatei geöffnet haben). Beachten Sie, dass diese Liste alle Komponenten enthält, die in das aktive Mapping eingefügt wurden: z.B. XML-Schemas.

   Im unteren Listenfeld können Sie eine neue komplexe Datenstruktur auswählen, d.h. XML-Schema.



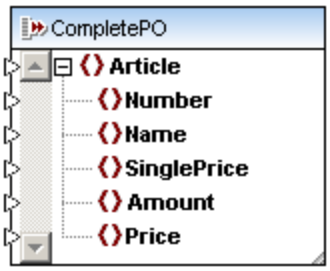
4. Klicken Sie auf das Optionsfeld "Neue Struktur einfügen...", wählen Sie den Eintrag "XML-Schema-Struktur" aus und klicken Sie zum Fortfahren auf OK.
5. Wählen Sie im Dialogfeld "Öffnen" die Datei CompletePO.xsd aus.
6. Klicken Sie auf das Element, das als Root-Element in der Komponente verwendet werden soll, z.B. **Article** und bestätigen Sie die Auswahl durch Klicken auf OK. Klicken Sie anschließend nochmals auf OK, um die Dialogfelder zu schließen.

© 2011 Altova GmbH

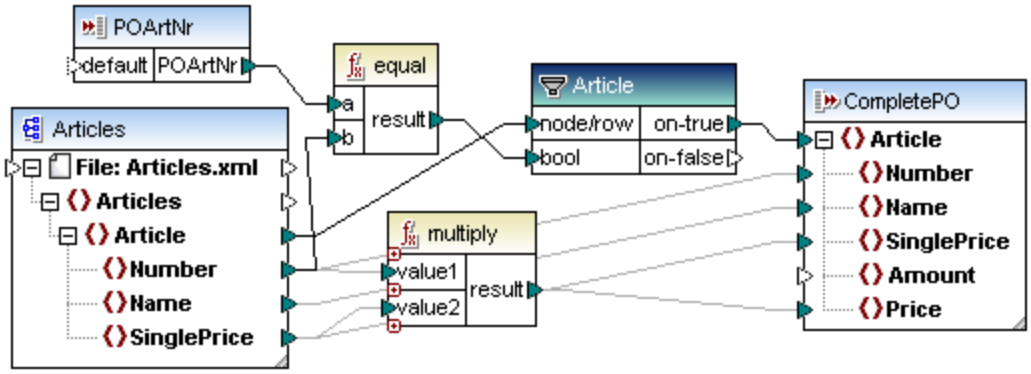


Die Komponente "CompletePO" wird in die benutzerdefinierte Funktion eingefügt. Beachten Sie bitte das Output-Symbol links vom Komponentennamen. Dieses Symbol zeigt an, dass die Komponente als komplexe Output-Komponente verwendet wird.

.



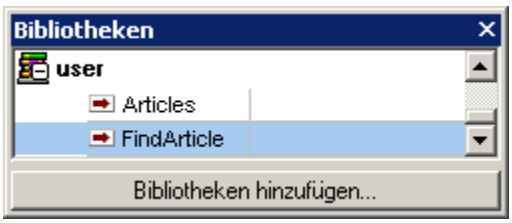
7. Fügen Sie das Schema-/die XML-Datei Articles in die benutzerdefinierte Funktion ein und weisen Sie die Datei Articles.xml als XML-Input-Instanz zu.
8. Fügen Sie die restlichen Komponenten wie im Screenshot unten gezeigt hinzu, nämlich: zwei "einfache" Input-Komponenten (POArtNr) sowie Filter-, equal- und multiply-Komponenten und verbinden Sie diese wie gezeigt.



Bitte beachten Sie:

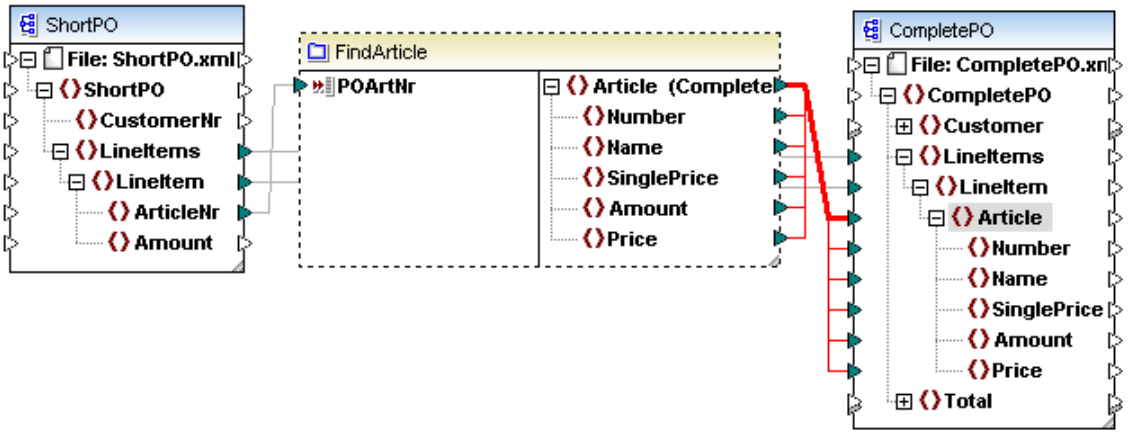
- Die Komponente "Articles" erhält ihre Daten aus der Beispieldatei Articles.xml in der benutzerdefinierten Funktion.
- Die Input-Komponenten stellen die Artikelnummer (POArtNr) und die Amount-Daten bereit, mit denen Articles | Number & Price verglichen werden.
- Die Filterkomponente filtert jene Datensätze, in denen die beiden Nummern identisch sind und übergibt sie an die Output-Komponente CompletePO.

9. Klicken Sie auf das Startseite-Symbol um zum Mapping zurückzukehren.
10. Ziehen Sie die neu erstellte benutzerdefinierte Funktion aus dem Bibliotheksfenster in das Mapping.



Java ausgewählt

11. Erstellen Sie die Verbindungen, wie im Screenshot unten gezeigt.
Nachdem Sie den Article (CompletePO) Konnektor zur Zielkomponente erstellt haben, klicken Sie mit der rechten Maustaste darauf und wählen Sie im Kontextmenü die Option"Alles kopieren" aus. Die restlichen Konnektoren werden automatisch generiert und sind im Screenshot unten markiert.



 © 2011 Altova GmbH

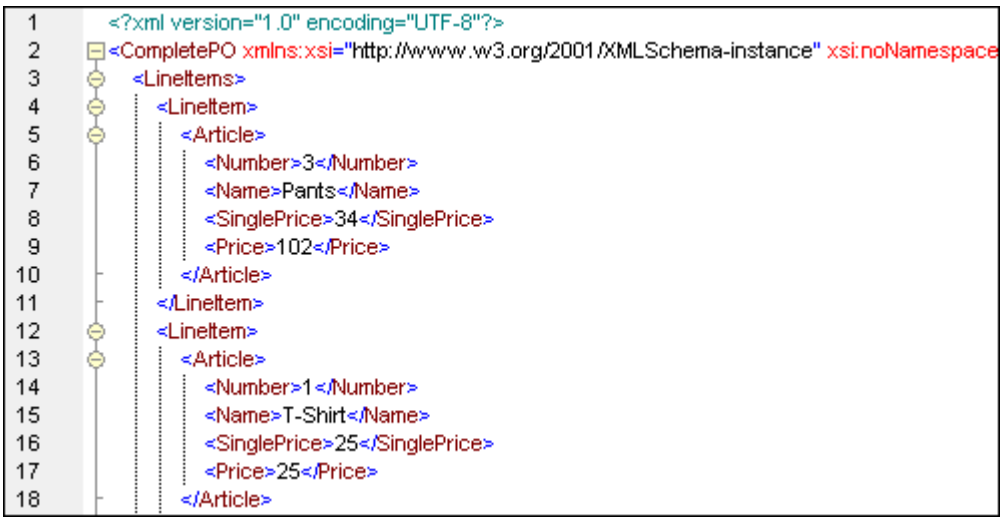
Bitte beachten Sie:
Beim Erstellen von "Alles kopieren" Verbindungen zwischen einem Schema und einer benutzerdefinierten Funktion vom Typ "Inline" müssen die beiden Komponenten auf demselben Schema basieren! Sie müssen jedoch nicht unbedingt dasselbe Root-Element haben.

Die linke Hälfte enthält die Input-Parameter, auf die ein einziges Datenelement gemappt wurde:

- ShortPO liefert die Artikelnummer für die Input-Komponente **POArtNr**

Die rechte Hälfte enthält:

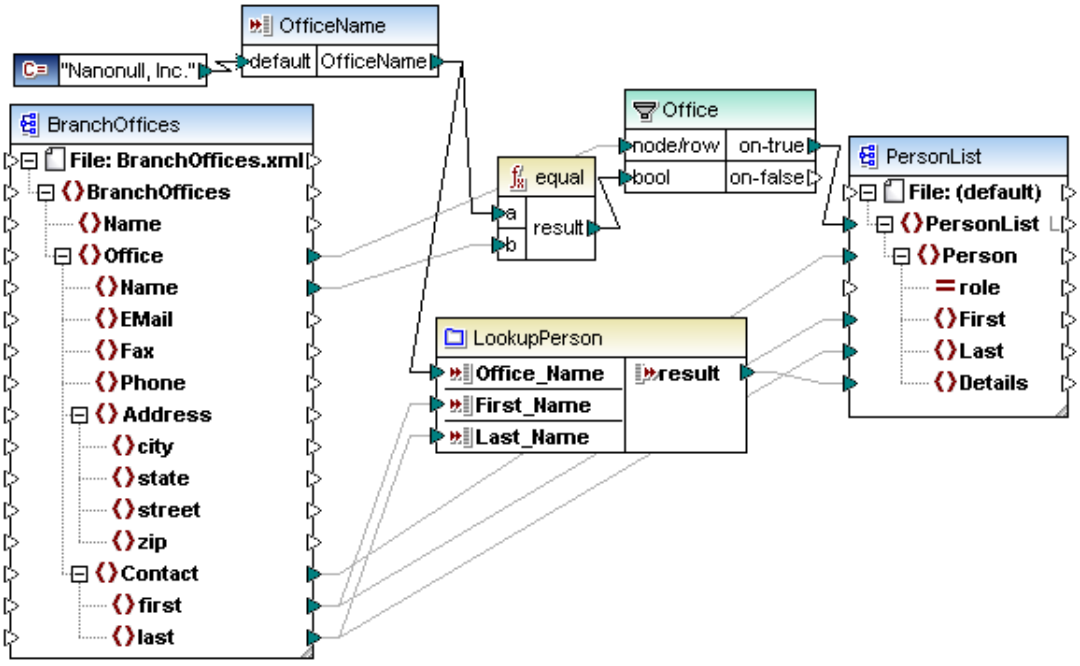
- eine komplexe Output-Komponente mit der Bezeichnung "Article (CompletePO)**"** mit ihren XML Child Nodes, die die gefilterten Datenelemente derselben Artikelnummer auf CompletePO mappt.



15.1.6 Benutzerdefinierte Funktion - Beispiel

Anhand der Beispieldatei **PersonListByBranchOffice.mfd** aus dem Ordner **...\MapForceExamples** werden die folgenden Features näher erläutert:

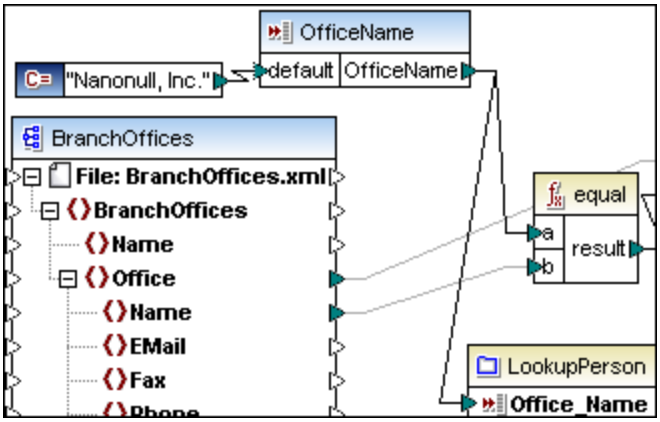
- Geschachtelte benutzerdefinierte Funktionen, z.B. **LookupPerson**
- Abfragefunktionen, die eine Ausgabe in Form eines Strings erzeugen, z.B. **LookupPerson**
- **Optionale** Eingabeparameter, die auch einen **default-**Wert liefern können, z.B. die **EqualAnd-**Komponente (enthalten in der LookupPerson-Komponente)
- **Konfigurierbare** Eingabeparameter, die auch als Befehlszeilenparameter dupliziert werden können, wenn der generierte Mapping-Code ausgeführt wird!



 © 2011 Altova GmbH

Konfigurierbare Eingabeparameter
Die Eingabekomponente (OfficeName) erhält Daten, wenn ein Mapping ausgeführt wird. Dies kann auf zwei Arten erfolgen:

- als ein Befehlzeilenparameter, wenn der generierte Code, z.B. Mapping.exe/OfficeName "Nanonull Partners, Inc." ausgeführt wird.
- als Vorschauwert, wenn Sie den Built-In-Ausführungsprozessor verwenden, um eine Vorschau der Daten im Fenster "Ausgabe" anzuzeigen.

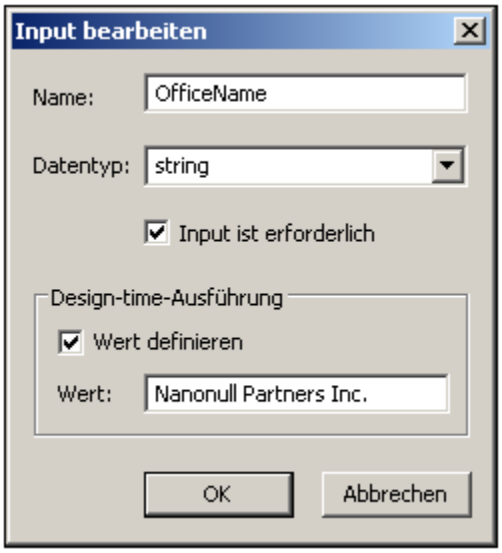


So definieren Sie den Eingabewert:

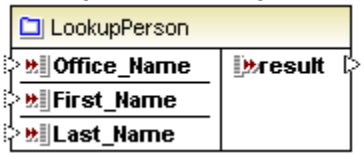
1. Doppelklicken Sie auf die Input-Komponente und geben Sie in der Ausgabe in das Textfeld "Wert" einen anderen Wert ein, z.B. "Nanonull Partners, Inc." und bestätigen Sie mit OK.
2. Klicken Sie auf die Schaltfläche "Ausgabe", um die Auswirkungen zu sehen. Es wird nun eine andere Gruppe von Personen angezeigt.

   Beachten Sie bitte, dass die in dieses Dialogfeld eingegebenen Daten nur im Vorschaufenster bzw. bei der Codegenerierung verwendet werden. Wurde kein Wert eingegeben oder ist das Kontrollkästchen deaktiviert, werden die auf das Input-Symbol "default" gemappten Daten verwendet.

   Nähere Informationen dazu finden Sie unter "Befehlszeile - Definieren von Input/Output-Dateien".



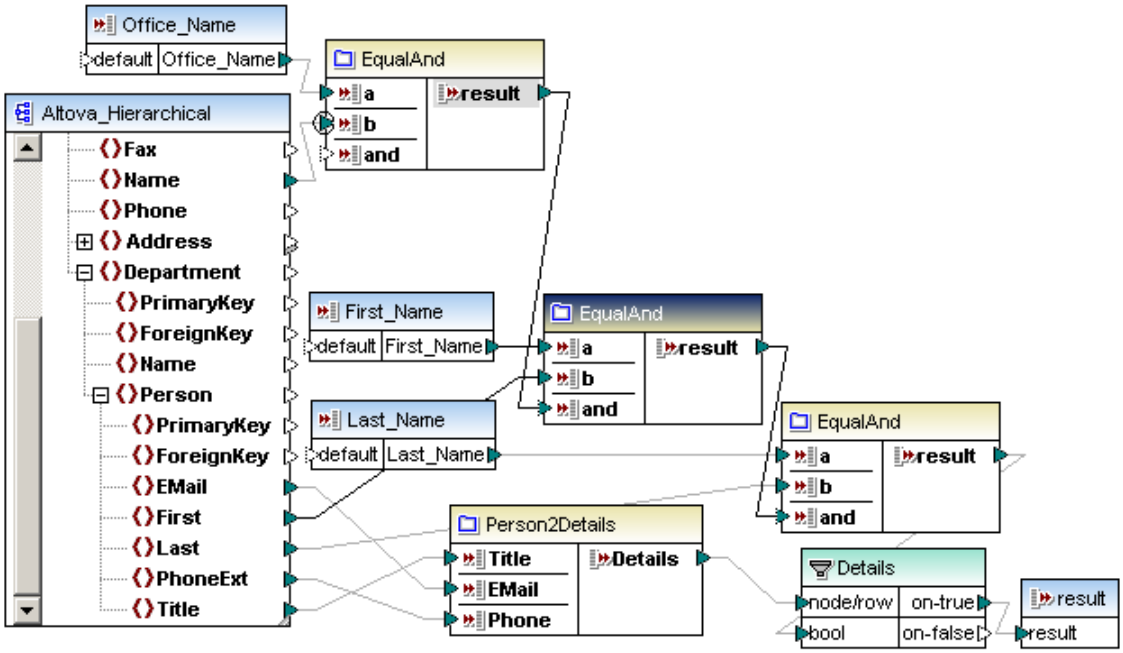
Die LookupPerson-Komponente



Wenn Sie auf diese benutzerdefinierte Komponente doppelklicken, sehen Sie unterhalb davon die Komponenten, aus denen sie sich zusammensetzt. Diese Komponente hat folgende Funktionen:

- Sie vergleicht mit Hilfe der **input-**Komponente und der benutzerdefinierten Komponente **EqualAnd** die Namen in Office_Name, First_Name und Last_Name der Datei BranchOffices.xml mit den gleichnamigen Feldern der Datei Altova_Hierarchical.xml.
- Sie kombiniert die Datenelemente Email, PhoneExt und Title mit Hilfe der benutzerdefinierten Funktion **Person2Details**.
- Sie übergibt die kombinierten Personendaten an die **output-**Komponente, wenn die Ergebnisse der vorherigen EqualAnd-Vergleiche alle "true" sind (d.h. es wird "true" an die filter-Komponente übergeben).

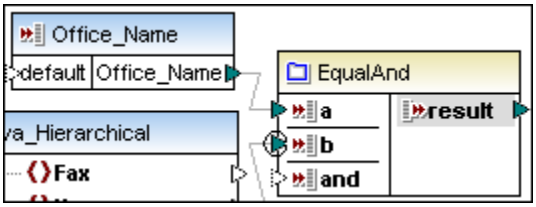
Eine benutzerdefinierte Funktion gibt immer einen Wert aus. Es kann sich dabei auch um einen leeren String handeln! Dies wäre der Fall, wenn der Boolesche Wert der filter-Komponente "false" ist. In diesem Fall würde nur ein leerer String anstelle der aus der Komponente Person2Details stammenden Daten ausgegeben.



- Die drei **input**-Komponenten Office_Name, First_Name, Last_Name erhalten ihre Daten aus der Datei BranchOffices.xml.
- Die **EqualAnd**-Komponente vergleicht zwei Werte und liefert einen **optionalen** Vergleichswert sowie einen Standardwert.
- Person2Details kombiniert drei Personenfelder und übergibt das Ergebnis an die Filterkomponente.

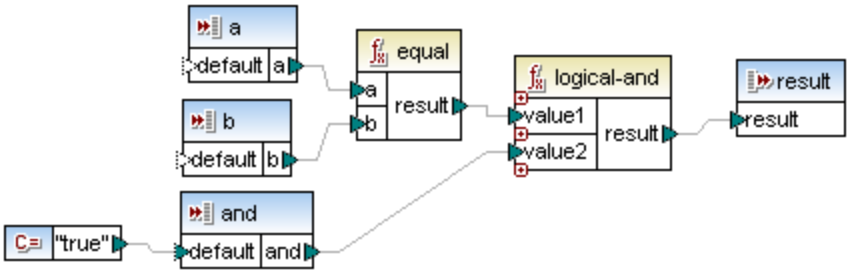
© 2011 Altova GmbH

Die EqualAnd-Komponente

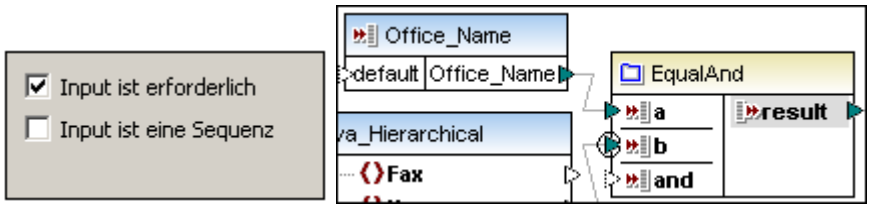


Wenn Sie auf diese benutzerdefinierte Komponente doppelklicken, werden die unten gezeigten Komponenten als ihre Bestandteile angezeigt. Diese Komponente:

- vergleicht zwei Eingabeparameter **a** und **b** und übergibt das Ergebnis an die logical-and-Komponente. Beachten Sie, dass der Parameter **b** als **Prioritätskontext** definiert wurde (rechtsklicken Sie auf das Symbol, um dies festzulegen). Damit stellen Sie sicher, dass die Personendaten des vom Eingabeparameter a bereitgestellten Büros zuerst verarbeitet werden.
- Die **Logical-and**-Komponente übergibt das Ergebnis des ersten Vergleichs mit einem **optionalen** Eingabeparameter "and".
- gibt den Booleschen Wert dieses Vergleichs an den Ausgabeparameter weiter.



Optionale Parameter
Wenn Sie auf den "and" Parameter der oben gezeigten benutzerdefinierten Funktion "EqualAnd" doppelklicken, können Sie Parameter als optional definieren, indem Sie das Kontrollkästchen "Input ist erforderlich" deaktivieren.



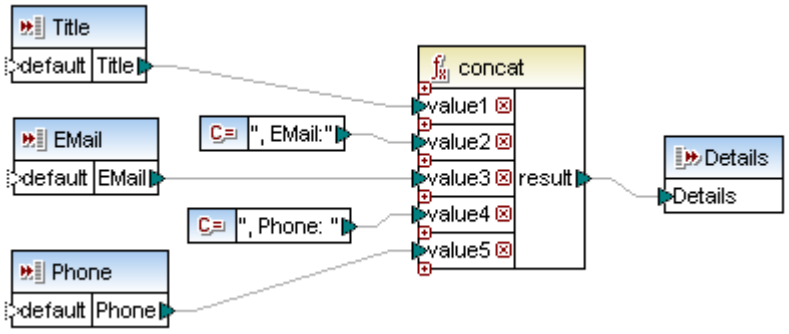
Wenn "Input ist erforderlich" **deaktiviert** ist, dann:

- wird für das Input-Symbol dieser benutzerdefinierten Funktion kein Mapping-Konnektor benötigt. So hat z.B. der **and**-Parameter der ersten EqualAnd-Funktion keinen Input-Konnektor. Das Input-Symbol erscheint strichliert, um dies visuell darzustellen.
- kann ein **default**-Wert angegeben werden, indem Sie eine Komponente innerhalb der benutzerdefinierten Funktion verbinden, z.B. mit Hilfe einer Konstanten-Komponente, die den Wert "true" enthält.
- hat ein Mapping von einem anderen Datenelement, das auf die optionale Eingabekomponente gemappt wird, Vorrang vor dem default-Wert. So erhält z.B. der "and"-Parameter der zweiten EqualAnd-Funktion Inputdaten vom "result"-Parameter der ersten benutzerdefinierten EqualAnd-Funktion.

Die **Person2Details-**Komponente

Wenn Sie auf diese benutzerdefinierte Komponente doppelklicken, werden die unten gezeigten Bestandteilkomponenten angezeigt. Diese Komponente:

- verkettet die drei Inputs und übergibt den Ergebnis-String an den Ausgabeparameter.
- Wenn Sie auf einen Ausgabeparameter doppelklicken, können Sie den Parameternamen (Details) ändern und den Datentyp auswählen (String).



 © 2011 Altova GmbH

15.2 Hinzufügen benutzerdefinierter XSLT- und XQuery-Funktionen

In MapForce haben Sie die Möglichkeit, die installierten XSLT-Funktionsbibliotheken durch Ihre eigenen benutzerdefinierten Funktionen zu ergänzen. Wählen Sie dazu XSLT als Ausgabeformat, indem Sie auf das Symbol XSLT klicken oder wählen Sie **Ausgabe | XSLT 1.0** .

XSLT-Dateien werden als Bibliotheken angezeigt. Unterhalb des Bibliotheksnamens werden alle Named Templates als Funktionen angezeigt.

- Funktionen müssen als Named Templates deklariert werden, die der XSLT 1.0 Spezifikation in der XSLT-Datei entsprechen.
- Wenn die importierte XSLT-Datei andere XSLT-Dateien importiert oder beinhaltet, so werden auch diese XSLT-Dateien und Funktionen importiert.
- Jedes Named Template wird als Funktion unterhalb der einzelnen Bibliotheksnamen angezeigt.
- Die Anzahl der Input-Symbole, die gemappt werden können, hängt von der Anzahl der Parameter ab, die im Template Call verwendet werden; auch optionale Parameter werden unterstützt.
- Aktualisierungen der importierten XSLT-Dateien werden beim Programmstart oder bei Dateiänderungen durchgeführt.
- Namespaces werden unterstützt.

Bitte beachten Sie:

Stellen Sie beim Erstellen von Named Templates bitte sicher, dass die im Template verwendeten XPath-Anweisungen an den/die richtigen Namespace(s) gebunden sind. Sie können die Namespace Bindings des Mappings durch Klicken auf das Register XSLT anzeigen. Nähere Informationen dazu finden Sie auch unter der Dokumentation zur Implementierung von XSLT 1.0 .

15.2.1 Benutzerdefinierte XSLT 1.0-Funktionen

Die Dateien, die Sie für das unten gezeigte Beispiel benötigen, liegen im Verzeichnis **...\MapForceExamples:**

- Name-splitter.xslt
- Name-splitter.xml (die XML-Instanzdatei für Customer.xsd)
- Customers.xsd
- CompletePO.xsd

Ein zusätzliches Beispiel für die Verwendung von Names Templates zum Summieren von Nodes finden Sie unter Aggregationsfunktionen.

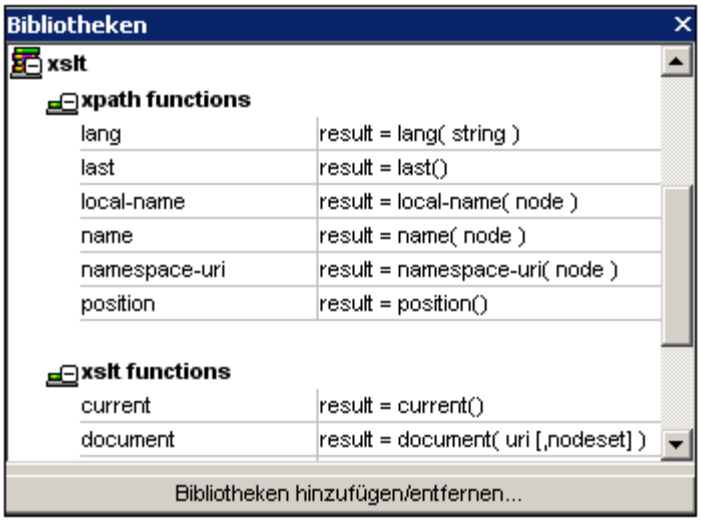
So fügen Sie eine benutzerdefinierte XSLT-Funktion hinzu:

1. Erstellen Sie eine XSLT-Datei, die die gewünschte Transformation/das gewünschte Ergebnis erzeugt.
   Im unten gezeigten Beispiel, **Name-splitter.xslt**, sehen Sie eine Named Template mit dem Namen "**tokenize**" mit einem einzigen Parameter "string". Die Vorlage verarbeitet einen Eingabe-String und trennt alle Großbuchstaben durch ein Leerzeichen voneinander.

```
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
4
5    <xsl:template match="*">
6      <xsl:for-each select=".">
7      <xsl:call-template name="tokenize">
8      <xsl:with-param name="string" select="."/>
9      </xsl:call-template>
10     </xsl:for-each>
11   </xsl:template>
12
13   <xsl:template name="tokenize">
14     <xsl:param name="string" select="."/>
15     <xsl:variable name="caps" select="translate($string, '-abcdefghijklmnopqrstuv
16     <xsl:variable name="capscount" select="string-length($caps)"/>
17     <xsl:variable name="token">
```

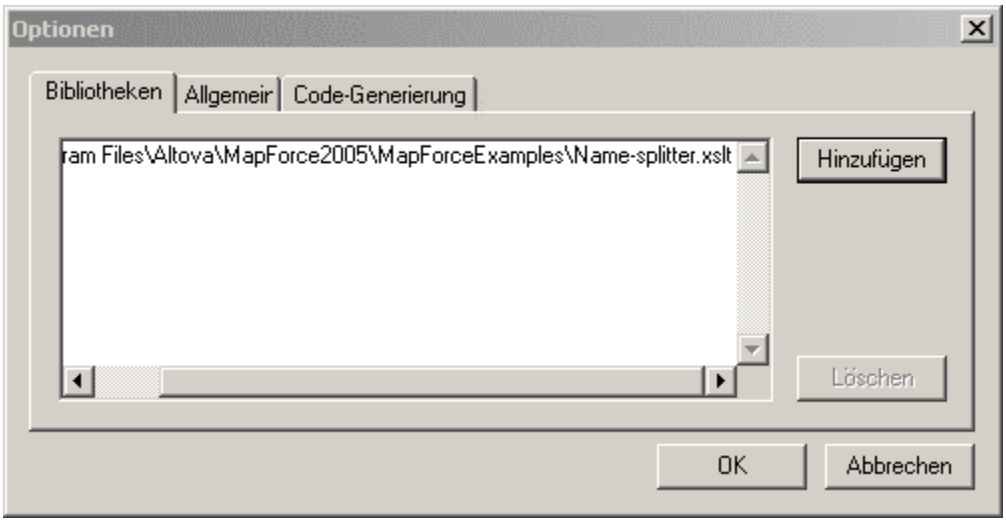
2. Klicken Sie auf die Schaltfläche **Bibliotheken hinzufügen/entfernen...** und im daraufhin angezeigten Dialogfeld auf "Hinzufügen".

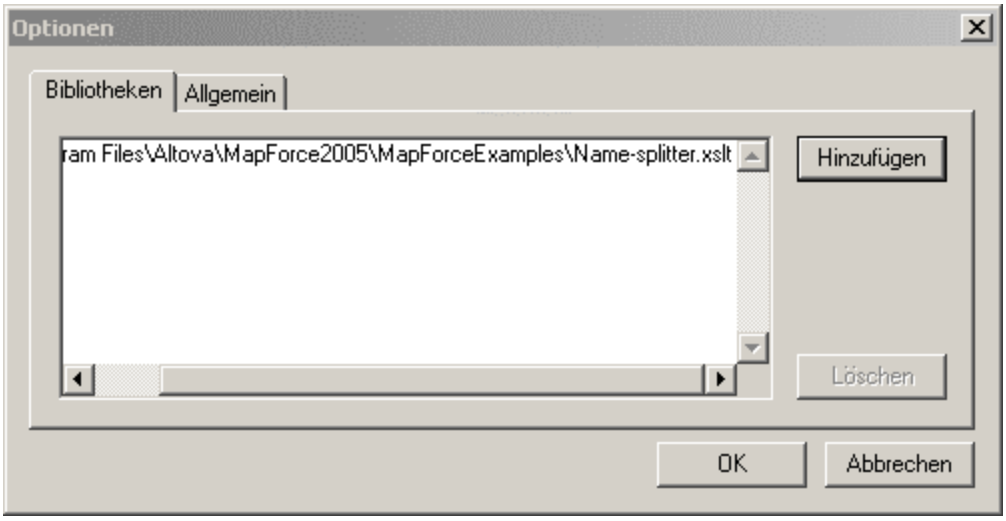
© 2011 Altova GmbH



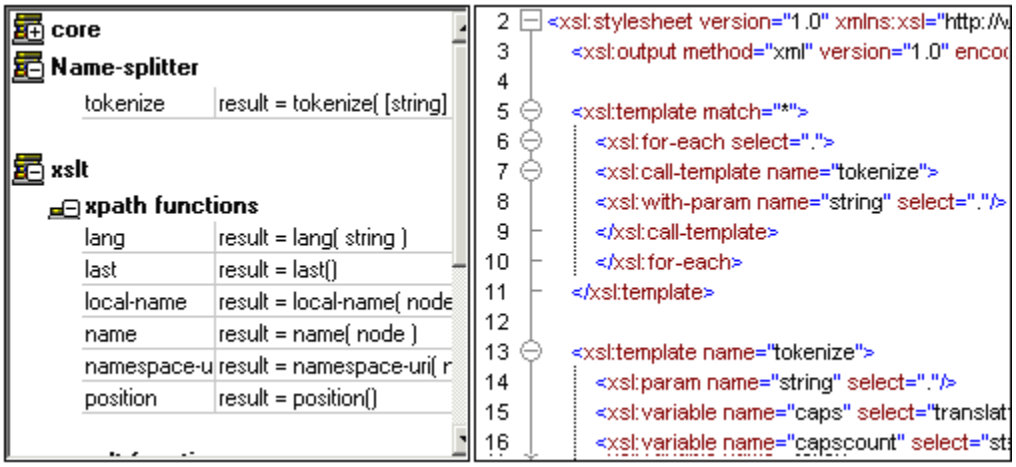
XSLT ausgewählt

3. Wählen Sie die XSL- oder XSLT-Datei aus, die das named Template enthält, das als Funktion verwendet werden soll, in diesem Fall die Datei **Name-splitter.xslt**. Die XSLT-Datei wird im Bibliotheksfenster angezeigt.





4. Klicken Sie auf **OK**, um die neue Funktion einzufügen.

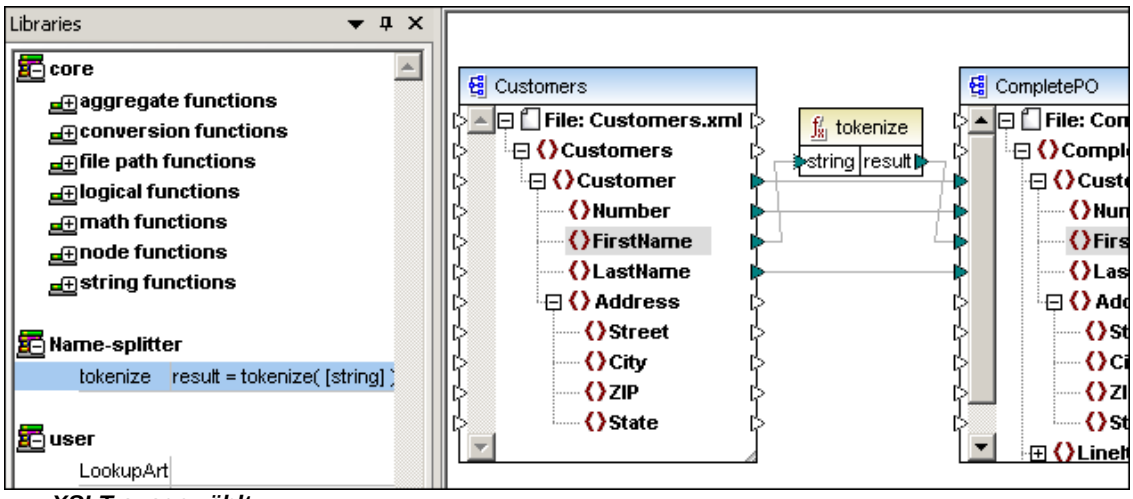


XSLT ausgewählt

Die XSLT-Datei wird zusammen mit der/den als Named Template definierte(n) Funktion(en) im Bibliotheksfenster angezeigt. In diesem Fall sehen wir die Datei **Name-splitter** mit der unterhalb davon angezeigten Funktion **tokenize**.

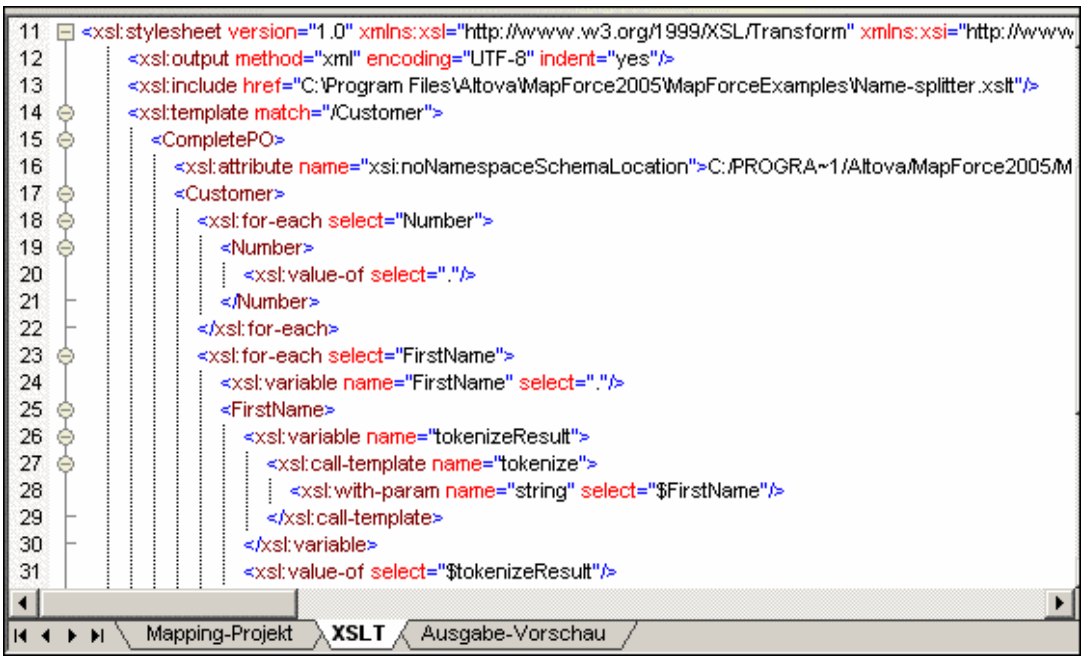
5. Ziehen Sie die Funktion in das Mapping-Fenster, um sie in Ihrem aktuellen Mapping zu verwenden und mappen Sie die nötigen Datenelemente, wie in der unten stehenden Abbildung gezeigt.

 © 2011 Altova GmbH



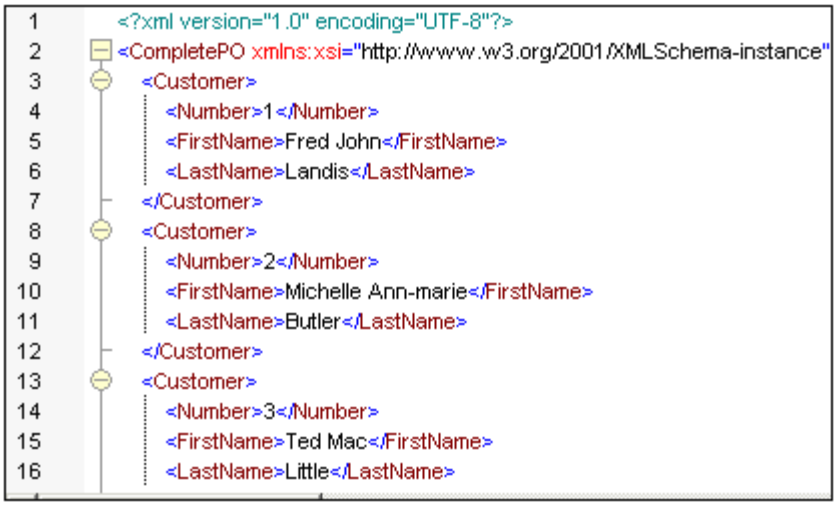
XSLT ausgewählt

6. Klicken Sie auf das Register XSLT, um den erzeugten XSLT-Code zu sehen.



Bitte beachten Sie:
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.**

7. Klicken Sie auf die Schaltfläche Ausgabe, um das Ergebnis des Mappings anzuzeigen.



So löschen Sie benutzerdefinierte XSLT-Funktionen:

1. Klicken Sie auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**.
2. Klicken Sie auf dem Bibliotheksregister auf den **Bibliotheksnamen** der gewünschten XSLT-Datei.
3. Klicken Sie auf die Schaltfläche **Löschen** und bestätigen Sie den Vorgang mit OK.

© 2011 Altova GmbH

15.2.2 Benutzerdefinierte XSLT 2.0-Funktionen

In MapForce können Sie auch XSLT 2.0-Funktionen importieren, die in einem XSLT 2.0-Dokument in folgender Form vorkommen:

<xsl:function name="MyFunction">

Nähere Informationen dazu finden Sie in der Dokumentation zur Implementierung von XSLT 2.0. Ein weiteres Beispiel zur Verwendung von Named Templates zum Summieren von Nodes finden Sie unter Aggregationsfunktionen.

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 "casten".

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`.

15.2.3 Aggregatfunktionen - Aggregieren von Nodes in XSLT1 und 2

In diesem Abschnitt wird beschrieben, wie Sie mehrere Nodes eines XML-Instanzdokuments verarbeiten und die Ergebnisse als einen einzigen Wert auf das Ziel-Datenelement mappen können. Die in diesem Beispiel verwendeten Dateien liegen im Ordner **...\MapForceExamples\Tutorial** und bestehen aus:

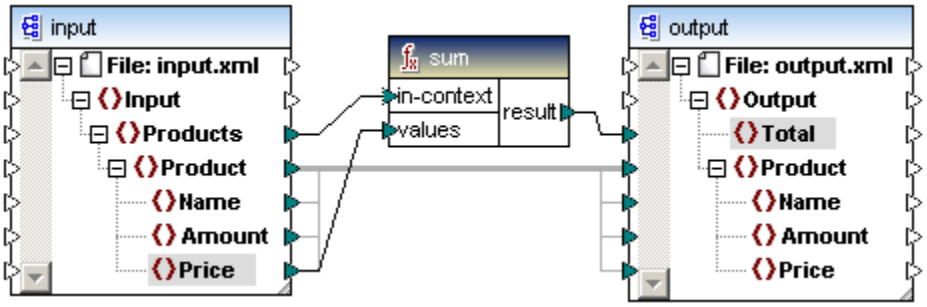
Summing-nodes.mfd	Mapping-Datei
input.xml	XML-Input-Datei
input.xsd and output.xsd	Quell- und Ziel-Schemadateien
Summing-nodes.xslt	xslt file, die ein Named Template zum Summieren einzelner Nodes enthält

Es gibt zwei separate Methoden zum Erstellen und Verwenden von Aggregatfunktionen:

- Verwendung der Aggregatfunktionen in der **core-Bibliothek** im Bibliotheksfenster
- Verwenden eines **Named Template**.

Aggregatfunktionen - Bibliothek
Je nachdem, welche XSLT-Bibliothek Sie auswählen, XSLT 1 oder XSLT 2, stehen unterschiedliche Aggregatfunktionen in der Core-Bibliothek zur Verfügung. XSLT 1 unterstützt count und sum, während XSLT 2 avg, count, max, min, string-join und sum unterstützt.

Ziehen Sie die zu verwendende Aggregatfunktion aus der Bibliothek in das Mapping und verbinden Sie die Quell- und Zielkomponenten wie in der Abbildung unten gezeigt.



Nähere Informationen zu dieser Art von Aggregatfunktion finden Sie unter Aggregatfunktionen.

Aggregatfunktionen - Named Templates
Im unten gezeigten Screenshot sehen Sie die **XML-Input-Datei.** Ziel dieses Beispiels ist, die Preisfelder beliebig vieler Produkte, in diesem Fall von Produkt A und B zu summieren.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Input xmlns:xsi="http://www.w3.org/2001/XMLSchema
3      <Products>
4          <Product>
5              <Name>ProductA</Name>
6              <Amount>10</Amount>
7              <Price>5</Price>
8          </Product>
9          <Product>
10             <Name>ProductB</Name>
11             <Amount>5</Amount>
12             <Price>20</Price>
13         </Product>
14     </Products>
15 </Input>
```

 © 2011 Altova GmbH

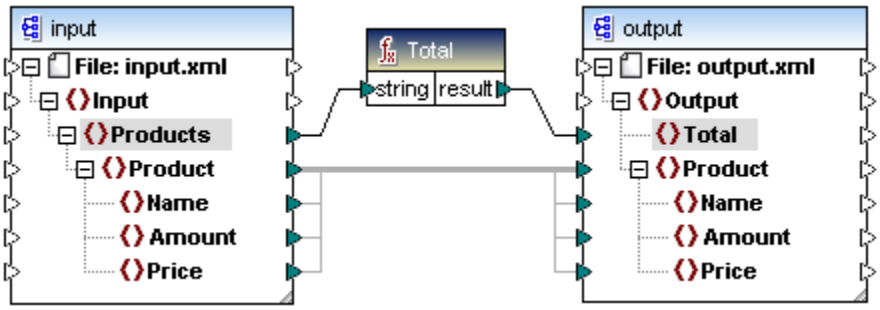
Im unten gezeigten Screenshot sehen Sie das XSLT Stylesheet, welches das Named Template "**Total**" und einen einzelnen Parameter "string" verwendet. Das Template arbeitet die XML-Input-Datei ab und summiert alle Werte, die mit dem XPath-Ausdruck **/Product/Price** im Dokument gefunden werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/19
    <xsl:output method="xml" version="1.0" encoding="UTF-8" i

    <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>
```

1. Klicken Sie auf die Schaltfläche **Bibliotheken hinzufügen/entfernen** und wählen Sie im Dialogfeld "Optionen" das Register "Bibliotheken".
2. Klicken Sie auf die Schaltfläche "Hinzufügen" und wählen Sie die Datei **Summing-nodes.xslt** aus dem Ordner **...\MapForceExamples\Tutorial**.
3. Ziehen Sie die Funktion "Total" aus der neu erstellten Bibliothek "Summing-nodes" ins Fenster und erstellen Sie Mappings wie unten gezeigt.



4. Klicken Sie auf die Schaltfläche "Ausgabe", um das Mapping-Ergebnis zu sehen.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Output xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNa
3    <Total>25</Total>
4    <Product>
5      <Name>ProductA</Name>
6      <Amount>10</Amount>
7      <Price>5</Price>
8    </Product>
9    <Product>
10     <Name>ProductB</Name>
11     <Amount>5</Amount>
12     <Price>20</Price>
13   </Product>
14 </Output>
15
```

Die beiden "Price"-Felder der beiden Produkten wurden addiert und in das Feld "Total" gesetzt.

So summieren Sie die Nodes in XSLT 2.0:

- Ändern Sie die Stylesheet-Deklaration in der Vorlage in ... version="2.0".

```
1    <?xml version="1.0" encoding="UTF-8"?>
2    <xsl:stylesheet version="2.0" xmlns:xs
3        <xsl:output method="xml" version="
```


© 2011 Altova GmbH

15.3 Hinzufügen benutzerdefinierter Java .class- und .NET DLL-Funktionen

Sie können im Bibliotheksfenster kompilierte Java-Klassendateien sowie .NET Assembly-Dateien (einschließlich .NET 4.0 Assemblies) hinzufügen, die dann wie jede andere Funktion in MapForce verwendet werden können. Eine Vorschau der Mapping-Ausgabe dieser Java- und .NET-Funktionen kann im Ausgabefenster angezeigt werden. Außerdem stehen die Funktionen im generierten Code zur Verfügung.

Unterstützt:

- Kompilierte Java-Klassendateien (.class-Dateien) werden unterstützt, wenn als Ausgabesprache Java eingestellt ist.
- .NET Assembly-Dateien werden unterstützt, wenn als Ausgabesprache C# eingestellt ist. .NET Assemblies werden im Allgemeinen unterstützt unabhängig davon, was die Ausgangssprache war (C++ oder VB.NET), vorausgesetzt sie verwenden als Parameter und Rückgabetypen nur die Basisdatentypen aus der System Assembly.

Nicht unterstützt:

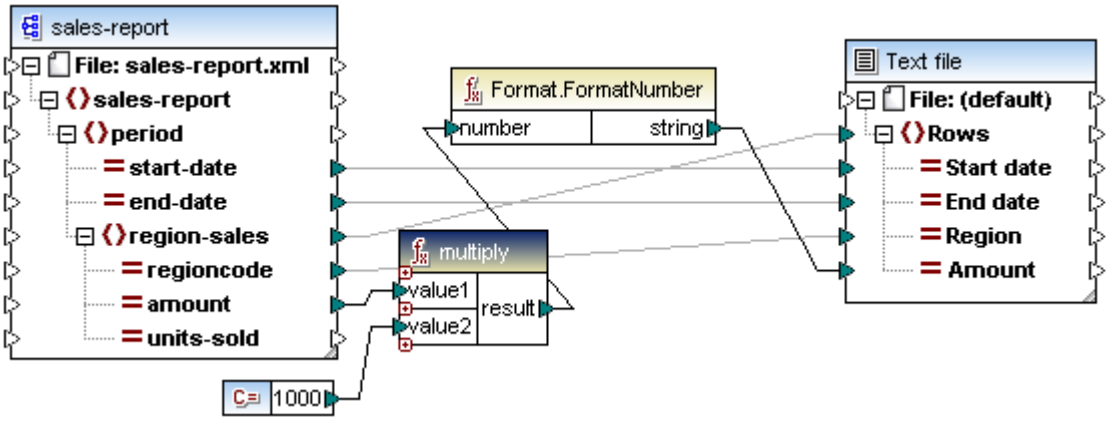
- Native C++ DLLs (oder andere DLLs, bei denen es sich nicht um eine .NET Assembly handelt)
- Verwendung von .class oder DLL Dateien/Funktionen, während als Ausgabesprache C++ eingestellt ist
- Verwendung von Java oder .NET Funktionen direkt in XSLT (es müsste eine benutzerdefinierte XSLT-Funktion geschrieben werden, die als Adapter fungiert)

Von C++-Funktionen kann keine Vorschau im Ausgabefenster angezeigt werden, doch steht die Ausgabe im generierten Code zur Verfügung.

Um benutzerdefinierte Java- oder .NET-Funktionen hinzuzufügen, benötigen Sie:

- die kompilierten Java-Klassen (.class) oder
- die .NET Assembly-Dateien (.dll)

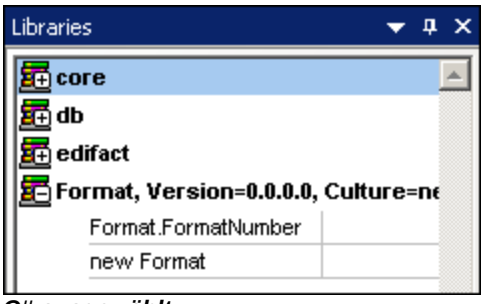
In den Ordnern ...\MapForceExamples**Java** und ...\MapForceExamples**C#**MapForce finden Sie Beispieldateien, in denen gezeigt wird, wie diese Funktionen verwendet werden. Beide Beispieldateien haben den Namen **FormatNumber.mfd**.



So fügen Sie die .NET Assembly-Datei hinzu:

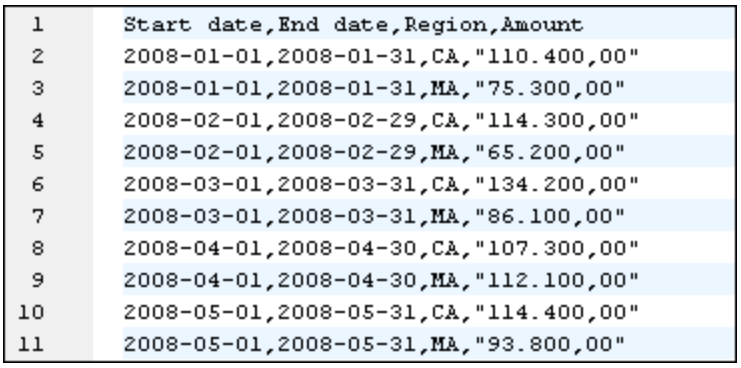
1. Klicken Sie im Bibliotheksfenster auf die Schaltfläche "Bibliotheken

hinzufügen/entfernen..." und im Ordner ...\MapForceExamples**C#\Format\bin\Debug** die Datei **Format.dll** aus.
Daraufhin wird eine Meldung angezeigt, dass eine neue Funktion hinzugefügt wurde. Die Funktion steht nun im Fenster "Bibliotheken" unter "Format" zur Verfügung.



C# ausgewählt

2. Öffnen Sie die Datei **FormatNumber.mfd** im Ordner ...\MapForceExamples**C#** (siehe Abbildung oben).
3. Klicken Sie auf die Schaltfläche "Ausgabe", um das Ergebnis des Mappings zu sehen.



```
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"
```

Nähere Informationen zur Implementierung finden Sie unter: Java- und .NET-Funktionen - Details.

© 2011 Altova GmbH

15.4 Java- und .NET-Funktionen - Details

Implementierungsdetails:

Hinzufügen von Bibliotheken:

- Java: `.class`-Dateien können zu MapForce hinzugefügt werden (.jar-Dateien werden nicht unterstützt)
- .NET: `.dll` Assembly-Dateien können zu MapForce hinzugefügt werden

Warnung: Alle von einem MapForce-Mapping aufgerufenen Funktionen sollten **"idempotent"** sein (Das bedeutet, sie 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 und kann sich jederzeit ändern.

Java-Funktionen - Konfiguration:
Wenn importierte Java-Klassendateien von anderen Klassendateien abhängig sind, stellen Sie sicher, dass Sie vor dem Start von MapForce die CLASSPATH-Umgebungsvariable anpassen. Zur CLASSPATH-Variablen müssen die übergeordneten Verzeichnisse aller abhängigen Pakete hinzugefügt werden.

Die importierten Klassendateien müssen nicht zur CLASSPATH-Variablen hinzugefügt werden, da der Built-In-Ausführungsprozessor - genau wie der generierte Java-Code - importierte Pakete automatisch zum Classpath bzw. zum ANT des Java-Prozessors hinzufügt.

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>, ...)`

Typen
Unterstützte Verbindungen zwischen XML-Schema- und Java-Typen:

Schema-Typ	Java-Typ
xs:string	String
xs:byte	byte
xs:short	short
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 werden nicht unterstützt.

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>, ...)`

Typen:
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
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 werden nicht unterstützt.

Array-Typen werden nicht unterstützt. Methoden, die solche Parameter oder Rückgabewerte verwenden, werden ignoriert.

 © 2011 Altova GmbH

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.

15.5 Referenz Funktionen

In diesem Abschnitt werden alle Funktionen beschrieben, die im Fenster "Bibliotheken" für die einzelnen unterstützten Sprachen XSLT1.

Derzeit stehen die folgenden Bibliotheken zur Verfügung:

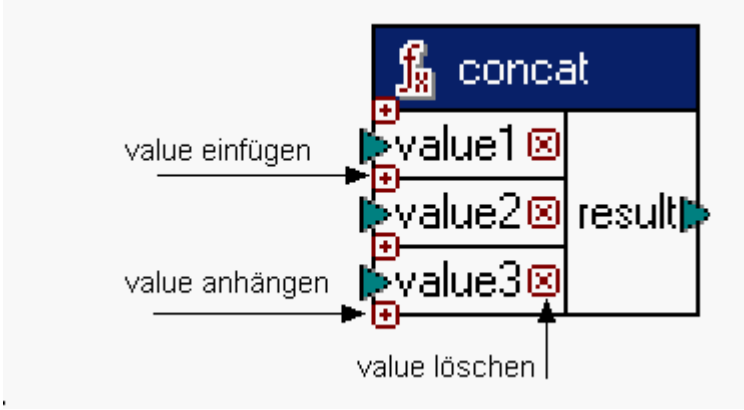
- **core**
- **xpath2**
- **xslt**

Erweiterbare Funktionen

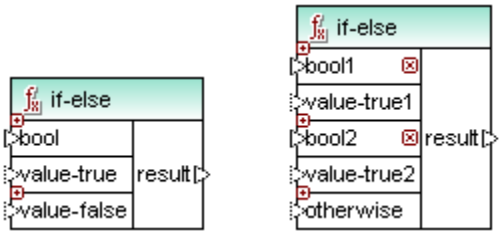
Eine Reihe von Funktionen aus den Funktionsbibliotheken kann erweitert werden, z.B: die Funktionen concat, "logical-and", "logical-or" und IF-ELSE. Die Parameter dieser Arten von Funktionen können je nach Belieben eingefügt/angehängt und gelöscht werden.

Wenn Sie auf das Plus-Symbol klicken, wird dieselbe Parameterart eingefügt oder angehängt. Wenn Sie auf das Häkchen klicken, wird der Parameter gelöscht.

Beachten Sie bitte. Wenn Sie einen Konnektor auf das Plus-Symbol ziehen, wird der Parameter automatisch eingefügt/angehängt und verbunden.



Die IF-Testparameter der IF-Else Funktion können auf dieselbe Art erweitert werden.



Wenn Sie den Mauszeiger über die Titelleiste der Funktion platzieren, erscheint ein Tooltipp mit einer Beschreibung der Funktion.

Wenn Sie ihn über einen Parameter (jeden beliebigen Input- oder Ergebnisparameter) platzieren, wird der Datentyp des Arguments in einem Tooltipp angezeigt.

© 2011 Altova GmbH

15.5.1 core

Die core-Bibliothek enthält die wichtigsten Funktionen für alle Sprachen. Die sequence-Funktionen stehen nicht zur Verfügung, wenn XSLT (XSLT 1.0)ausgewählt ist.

Core-Bibliothek

- aggregates
- conversion functions
- file path functions
- generator functions
- logical functions
- math functions
- node functions
- sequence functions
- string functions

aggregates (Aggregatfunktionen)

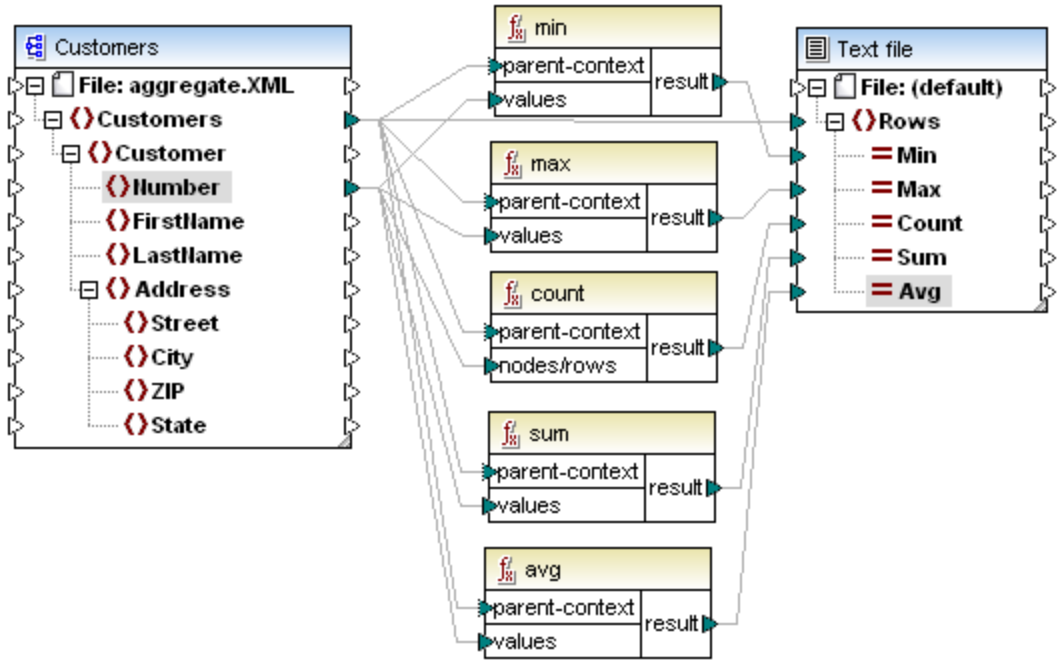
Mit Hilfe von Aggregatfunktionen werden Operationen an einer Gruppe oder Reihe von Input-Werten ausgeführt. Die Input-Daten für min, max, sum und avg werden für die Verarbeitung in den Datentyp **decimal** konvertiert.

- Die Input-Werte müssen mit dem **values** Parameter der Funktion verbunden werden.
- Der Kontextnode (Datenelement) muss mit dem **parent-context** Parameter verbunden werden, um den Standardkontext, aus dem die Input-Sequenz geholt wird, außer Kraft zu setzen. Das bedeutet auch, dass der parent-context-Parameter optional ist.
- Das Ergebnis der Funktion (**result)** wird mit dem entsprechenden Zieldatenelement verbunden.

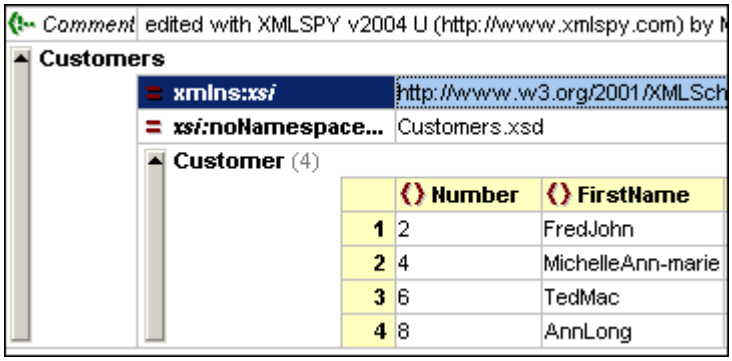
Das nachfolgende Mapping finden Sie unter dem Namen Aggregates.mfd im Ordner .... \Tutorial. Es zeigt, wie diese Funktionen verwendet werden.

Aggregatfunktionen haben zwei Input-Elemente.

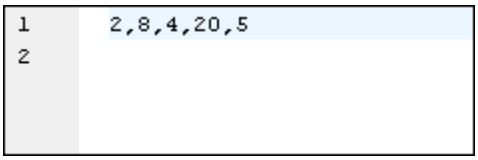
- **values** (nodes/rows) ist mit dem Quellelement verbunden, aus dem die Daten stammen, in diesem Fall "Number".
- **parent-context** ist mit dem Element verbunden, an dem iteriert werden soll, in diesem Fall allen Kunden ("Customer"). Der Parameter ist allerdings optional.



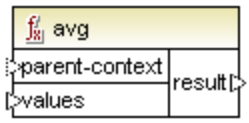
Bei der Input-Instanz handelt es sich in diesem Fall um eine XML-Datei mit den folgenden Daten:



- Bei den Quelldaten, die an das Element "values" übergeben werden, handelt es sich um die Zahlenfolge 2,4,6,8.
- Bei der Output-Komponente handelt es sich in diesem Fall um eine einfache Textdatei. Wenn Sie für das obige Mapping auf das Register "Ausgabe" klicken, erhalten Sie das folgende Ergebnis:



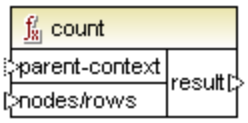
min=2, max=8, count=4, sum=20 and avg=5.



avg

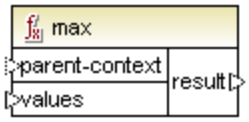
 © 2011 Altova GmbH

Gibt den Durchschnittswert aller Werte in der Input-Sequenz zurück. Der Durchschnittswert einer leeren Gruppe ist eine leere Gruppe. Steht in XSLT1 nicht zur Verfügung.



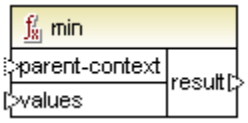
count

Gibt die Anzahl der einzelnen Datenelemente zurück, aus denen die Input-Sequenz besteht. Die Anzahl einer leeren Gruppe ist Null. Eingeschränkte Funktionalität in XSLT1.



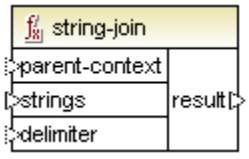
max

Gibt den Maximalwert aller Werte in der Input-Sequenz zurück. Der Maximalwert einer leeren Gruppe ist eine leere Gruppe. Steht in XSLT1 nicht zur Verfügung.



min

Gibt den Minimalwert alle Werte in der Input-Sequenz zurück. Der Minimalwert einer leeren Gruppe ist eine leere Gruppe. Steht in XSLT1 nicht zur Verfügung.

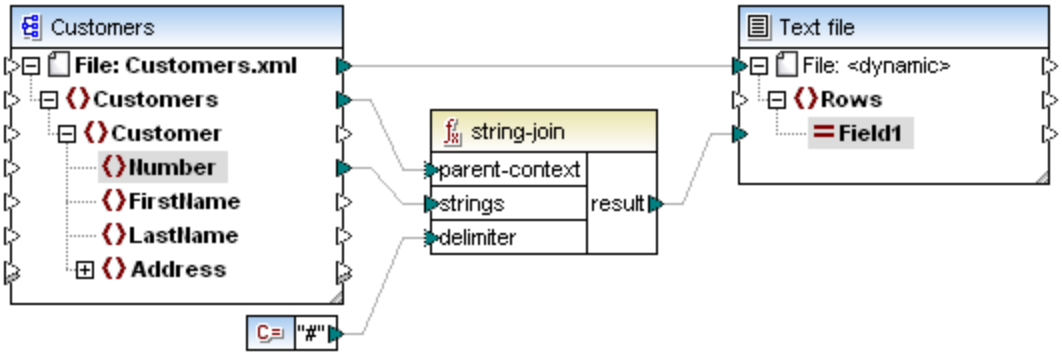


string-join

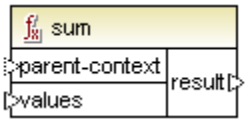
Verkettet alle Werte der Input-Sequenz zu einem String, der durch ein beliebiges von Ihnen ausgewähltes Trennzeichen getrennt ist. Der **string-join** Wert einer leeren Gruppe ist ein leere String. Steht in XSLT1 nicht zur Verfügung.

Das Beispiel unten enthält vier separate Kundennummern 2 4 6 und 8. Das Konstantenzeichen liefert als Trennzeichen eine Raute "#".

Ergebnis = 2#4#6#8



Wenn Sie kein Trennzeichen definieren, ist die Standardeinstellung ein leerer String, d.h. keinerlei Trennzeichen. Ergebnis = 2468.



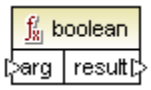
sum

Gibt die arithmetische Summe aller Werte in der Input-Sequenz zurück. Die Summe einer leeren Gruppe ist Null. Steht in XSLT1 nicht zur Verfügung.

conversion functions (Konvertierungsfunktionen)

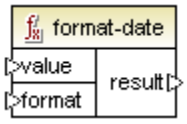
Zur Unterstützung expliziter Datentypkonvertierungen stehen in der Bibliothek "conversion functions" die Konvertierungsfunktionen "boolean", "number" und "string" zur Verfügung. Beachten Sie, dass MapForce notwendige Konvertierungen in den meisten Fällen automatisch durchführt und dass diese Funktionen nur in speziellen Fällen verwendet werden müssen.

Input-Parameter = **arg/value**
Output-Parameter = **result**



boolean

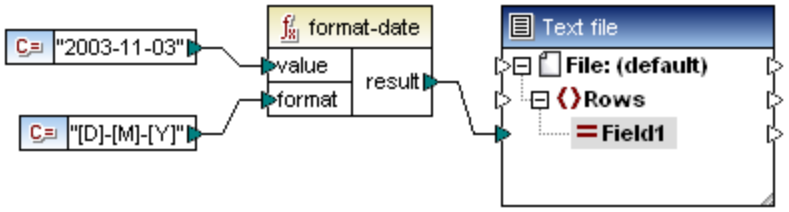
Konvertiert einen numerischen Input-Wert in einen Booleschen (und konvertiert einen String in einen numerischen Wert - true in 1). Wert, z.B. 0 in "false" oder 1 in "true". Dieser Wert kann anschließend in logischen Funktionen (gleich, größer als usw.), Filtern oder if-else-Funktionen verwendet werden.



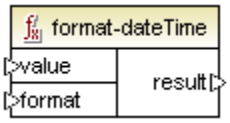
format-date

Konvertiert einen xs:date Eingabewert in einen String.

 © 2011 Altova GmbH



Ergebnis: is 3-11-2003



format-dateTime

Konvertiert einen dateTime-Datentyp in einen String.

Argument	Beschreibung
value	Der zu formatierenden dateTime-Wert
format	Ein Formatstring, der definiert, wie dateTime formatiert werden soll

Anmerkung:
Wenn die Ausgabe der Funktion (also 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 der Zielkomponente abgeschaltet werden.

Formatierungsstring

Das Argument "format" besteht aus einem String, der so genannte Variablen-Marker enthält, die innerhalb von eckigen Klammern stehen. Die Zeichen außerhalb der eckigen Klammern sind Literalzeichen, die in das Ergebnis kopiert werden. 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. Falls 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 der Stunde	00-59
s	Sekunden in der Minute	00-59
f	Sekundenbruchteile	numerisch, eine Dezimalstelle
Z	Zeitzone gerechnet als Zeitabstand zur UTC	+08:00 oder PST mit alphabetischem Modifier
z	Zeitzone gerechnet als Zeitabstand zur GMT (Westeuropäische Zeit)	GMT+n

Der Darstellungs-Modifier:

Specifier	Beschreibung	Beispiel
o	Bezeichnet die Nummerierung in Ordnungszahlen	[1o] bezeichnet 1., 2., 3., 4....

Der Formatierungs-Modifier:

Zeichen	**Beschreibung**	**Beispiel**
1	Numerisches Dezimalformat, wobei keine Null vorangestellt ist: 1, 2, 3, ...	1, 2, 3
01	Dezimalformat, zwei Stellen: 01, 02, 03, ...	01, 02, 03
i	Römische Ziffern	i, ii, iii, iv
I	Römische Zahlen, Großbuchstaben	I, II, III, IV
N	Name der Komponente, Großbuchstaben	MONTAG, DIENSTAG
n	Name der Komponente, Kleinbuchstaben	monday, tuesday
Nn	Name der Komponente, erster Buchstabe ist ein Großbuchstabe	Montag, Dienstag
W	Ausgeschriebene Zahl, Großbuchstaben (nicht für das Parsen)	EINS, ZWEI, DREI
w	Ausgeschriebene Zahl, in Kleinbuchstaben (nicht für das Parsen)	eins, zwei, drei
Ww	Ausgeschriebene Zahl, erster Buchstabe ist ein Großbuchstabe (nicht für das Parsen)	Eins, Zwei, Drei

1) Die Modifier N, n und Nn unterstützen nur die folgenden Komponenten: M, d, D.

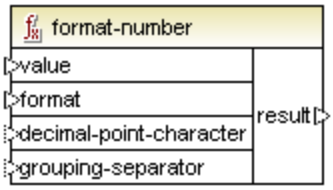
© 2011 Altova GmbH

Vor dem Modifier "width", steht, falls vorhanden, ein Komma. Der Modifier hat die folgende Form:

```
, min-width ("-" max-width)?
```

Unterstützte Beispiele

DateTime	Formatierungsstring	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]	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
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



format-number

Verfügbar für XSLT 1.0, XSLT 2.0, Java, C#, C++ und den Built-In-Ausführungsprozessor.

Argument	**Beschreibung**
value	Die zu formatierende Zahl
format	Ein Formatierungsstring, der festlegt, wie die Zahl formatiert werden soll
decimal-point-format	Das Zeichen, das für das Dezimalzeichen, also im Deutschen das Komma, verwendet wird. Standardeinstellung ist der Punkt '.' (optional)
grouping-separator	Das Trennzeichen, das zur Trennung von Tausenderstellen verwendet werden soll. Standardmäßig ist das Zeichen "," eingestellt (optional).

Anmerkung:

Wenn die Ausgabe der Funktion (also 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 der Zielkomponente abgeschaltet werden.

format
Ein Formatierungsstring legt fest, wie eine Zahl formatiert werden soll.

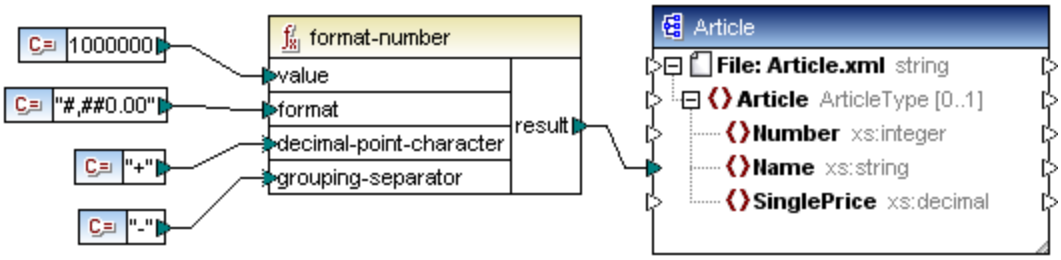
Format:

```
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	Standard	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 Promille an

Die Zeichen, die als Dezimalpunktzeichen und als Tausender-Trennzeichen verwendet werden, sind immer die Zeichen "." bzw. ",". Durch Mappen von Konstanten auf diese Nodes kann dies allerdings in der formatierten Ausgabe geändert werden.

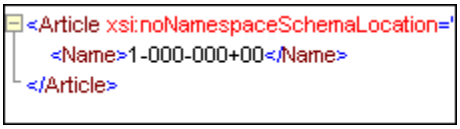


Das Ergebnis oben gezeigten Zahlenformatierungsfunktion.

- Der Dezimalpunkt wird in ein "+" Zeichen geändert.

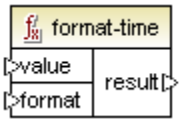
 © 2011 Altova GmbH

- Das Tausender-Trennzeichen wurde in ein "-" Zeichen geändert.



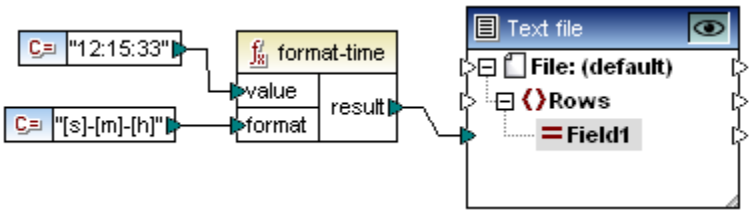
Rounding
Die für diese Funktion verwendete Rundungsmethode ist, Aufrunden ab der Hälfte, d.h. wenn der Wert hinter dem Komme > oder gleich, 0,5 ist, so wird aufgerundet. Wenn der Wert kleiner als 0,5 ist, wird abgerundet.

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
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

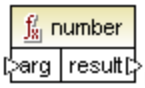


format-time
Konvertiert einen xs:time Eingabewert in einen String.

z.B.

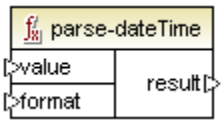


Ergebnis: 33-15-12



number

Konvertiert einen Input String in eine Zahl. Konvertiert einen Booleschen Input auch in eine Zahl.



parse-dateTime

Verfügbar für Java, C#, C++ und den Built-In-Ausführungsprozessor.

Konvertiert einen String in einen dateTime-Wert.

Argument	Beschreibung
value	Der String, der Datum und Uhrzeit enthält.
format	Der Bildstring, der definiert, wie der Wert derzeit formatiert wird

Die Komponenten sind:

Parameter	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
s	Sekunden in Minute	00-59
f	Sekundenbruchteile	numerisch, eine Dezimalstelle
Z	Zeitzone, als Zeitabstand von UTC	+08:00 oder PST mit alphabetischem Modifier
z	Zeitzone als Zeitabstand von GMT	GMT+n

Der Formatierungsparameter:

Zeichen	Beschreibung	Beispiel
1	Numerisches Dezimalformat ohne vorangestellte Null: 1, 2, 3, ...	1, 2, 3
01	Dezimalformat, zwei Ziffern: 01, 02, 03, ...	01, 02, 03

 © 2011 Altova GmbH

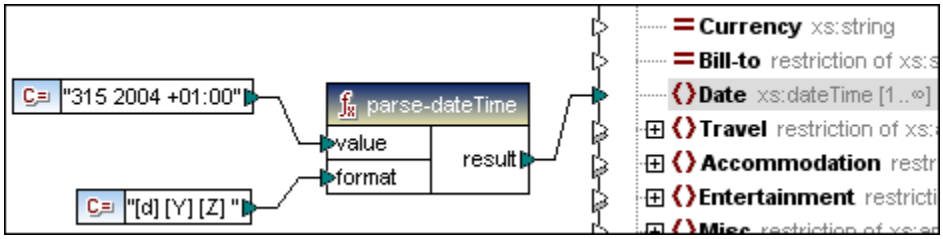
N	Name der Komponente, Großbuchstaben	MONDAY, TUESDAY
n	Name der Komponente, Kleinbuchstaben	monday, tuesday
Nn	Name der Komponente, beginnend mit einem Großbuchstaben	Monday, Tuesday

1) Die Parameter **N, n und Nn** unterstützen nur die Komponente M (Monat).

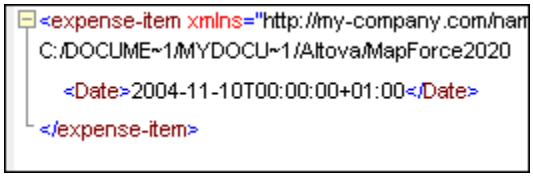
Beispiele:

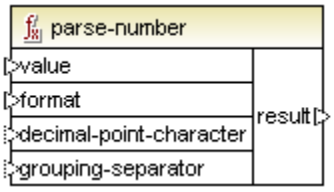
Datumsstring	Bildstring	Ergebnis
21-03-2002 16:21:12.492 GMT+02:00	[D]-[M]-[Y] [H]:[m]:[s].[f] [z]	2002-03-21T16:21:12.492+02:00
315 2004 +01:00	[d] [Y] [Z]	2004-11-10T00:00:00+01:00
1.December.10 03:2:39 p.m. +01:00	[D].[MNn].[Y,2-2] [h]:[m]:[s] [P] [Z]	2010-12-01T15:02:39+01:00
20110620	[Y,4-4][M,2-2][D,2-2]	2011-06-20T00:00:00

Beispiel in MapForce:



Ergebnis:





parse-number

Verfügbar für Java, C#, C++ und den Built-In-Ausführungsprozessor.

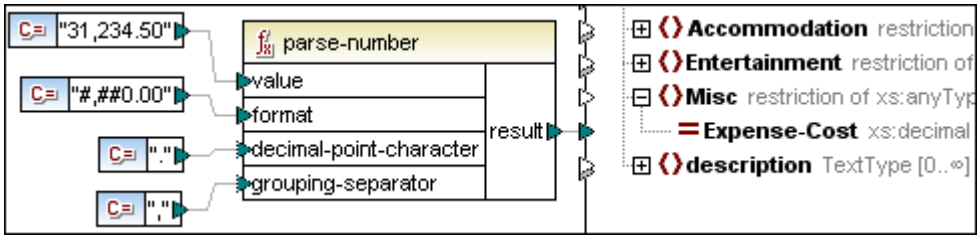
Konvertiert einen Input-String in eine Dezimalzahl.

Argument	Beschreibung
value	Der zu parsende/in eine Zahl zu konvertierende String

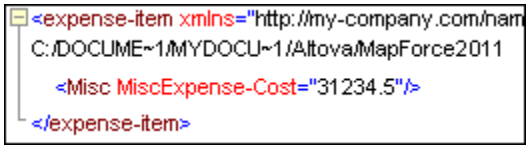
format	Ein Formatierungsstring, der angibt, wie die Zahl derzeit formatiert ist (optional). Die Standardeinstellung ist "#,##0.#"
decimal-point-character	Das als Dezimaltrennzeichen zu verwendende Zeichen. Standardeinstellung ist das Zeichen '.' (optional)
grouping-separator	Das für separate Zahlengruppen zu verwendenden Trennzeichen. Standardeinstellung ist das Zeichen "," (optional)

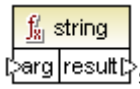
Der in parse-number verwendete ***Formatierungsstring*** ist derselbe, der auch in format-number verwendet wird.

Beispiel in MapForce:



Ergebnis:



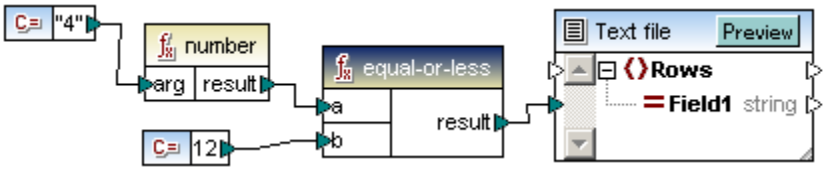


string

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 complexType handelt, so werden alle untergeordneten Nodes ebenfalls als einzelner String ausgegeben.

Vergleich unterschiedlicher Input Node Typen

Wenn es sich bei den Input Nodes um unterschiedliche Typen handelt, z.B. Integer und String, können Sie mit Hilfe der Konvertierungsfunktionen einen String-Vergleich oder einen numerischen Vergleich erzwingen.



Im Beispiel oben hat die erste Konstante den Typ string und enthält den String "4".

 © 2011 Altova GmbH

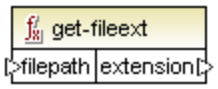
Die zweite Konstante enthält die numerische Konstante 12. Damit die beiden Werte explizit verglichen werden können, müssen sie denselben Typ haben.

Durch Hinzufügen einer **number** Funktion zur ersten Konstante wird die string-Konstante in den numerischen Wert 4 konvertiert. Das Ergebnis der Vergleiche ist dann "true".

Beachten Sie: Wenn die number-Funktion nicht verwendet würde und 4 direkt mit einem Parameter verbunden würde, würde es zu einem String-Vergleich kommen, wobei das Ergebnis "false" wäre.

file path functions (Dateipfadfunktionen)

Mit Hilfe von Dateipfadfunktionen können Sie Dateipfaddaten, also Ordner, Dateinamen und Erweiterungen direkt aufrufen und für die weitere Verarbeitung in Ihren Mappings bearbeiten. Diese Funktionen können in allen Sprachen, die von MapForce unterstützt werden, verwendet werden.



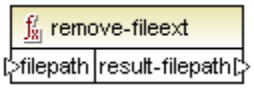
get-fileext
Gibt die Erweiterung des Dateipfads einschließlich des Punktzeichens "." zurück.
z.B. gibt 'c:\data\Sample.mfd' die Erweiterung '.mfd' zurück.

get-folder
Gibt den Ordnernamen des Dateipfads einschließlich des Schrägstrichs oder umgekehrten Schrägstrichs zurück.
z.B. gibt 'c:/data/Sample.mfd' 'c:/data/' zurück.

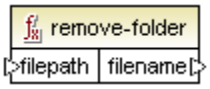
main-mfd-filepath
Gibt den vollständigen Pfad der mfd-Datei, die das Hauptmapping enthält, zurück. Wenn die mfd-Datei derzeit noch nicht gespeichert ist, wird ein leerer String zurückgegeben.

mfd-filepath
Wenn die Funktion im Hauptmapping aufgerufen wird, gibt die Funktion dasselbe wie die Funktion main-mfd-filepath 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 **benutzerdefinierten Funktion** aus aufgerufen wird, die von einer mfd-Datei **importiert** wird, gibt sie den vollständigen Pfad der importierten mfd-Datei zurück, die die **Definition** der benutzerdefinierten Funktion enthält.

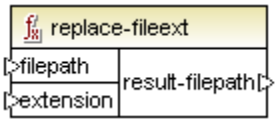


remove-fileext
Entfernt die Erweiterung des Dateipfads einschließlich des Punkts.
z.B. gibt 'c:/data/Sample.mfd' 'c:/data/Sample' zurück.



remove-folder

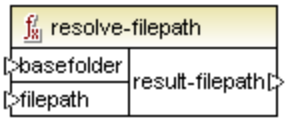
Entfernt das Verzeichnis des Dateipfads einschließlich des nachgestellten Schrägstrichs oder umgekehrten Schrägstrichs.
z.B. gibt 'c:/data/Sample.mfd' 'Sample.mfd' zurück.



replace-fileext

Ersetzt die durch den filepath-Parameter bereitgestellte Erweiterung des Dateipfads durch die Erweiterung, die durch die Verbindung zum extension-Parameter bereitgestellt wird.

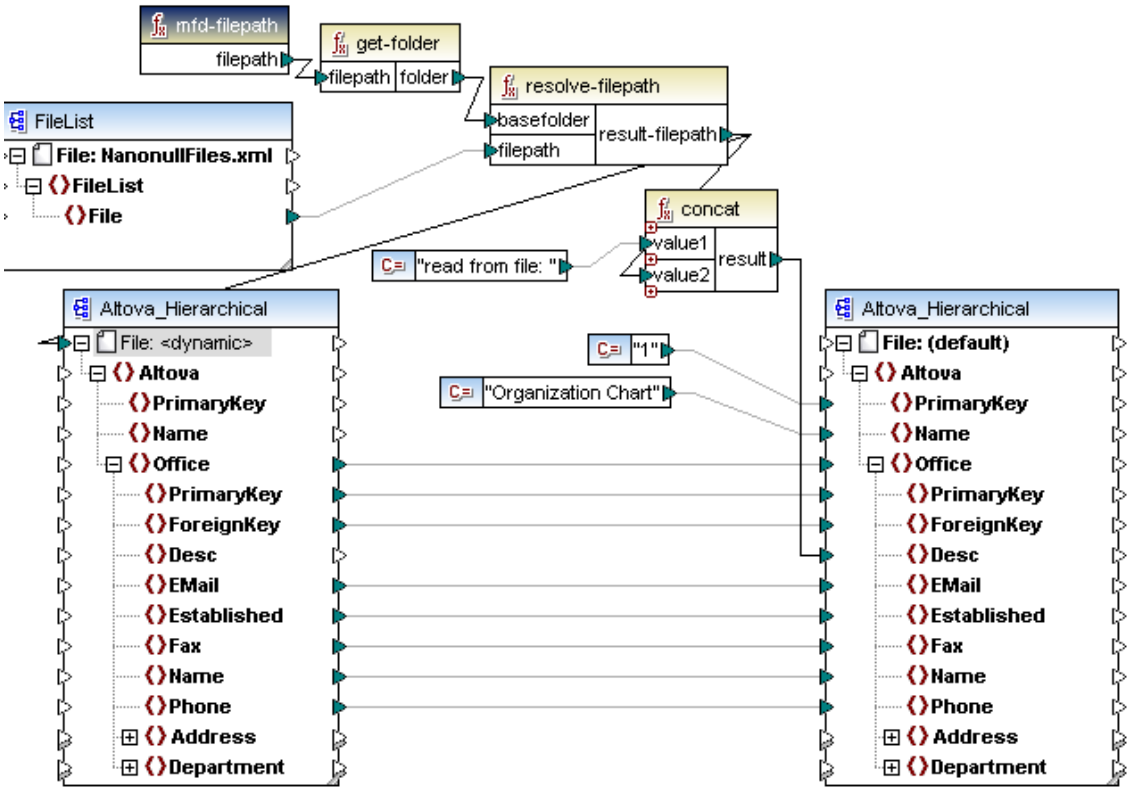
z.B. gibt c:/data/Sample.**mfd'** als Input-Dateipfad und '.**mfp'** als die Erweiterung 'c:/data/Sample.mfp' zurück.



resolve-filepath

Löst einen relativen Dateipfad zu einem relativen oder absoluten Basisordner auf. Die Funktion unterstützt '.' (aktuelles Verzeichnis) und '..' (übergeordnetes Verzeichnis).

Ein Beispiel dazu finden Sie im Mapping **MergeMultipleFiles_List.mfd** im Ordner ... \MapForceExamples.



 © 2011 Altova GmbH

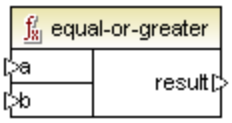
logical functions (logische Funktionen)

Logische Funktionen dienen (im Allgemeinen) dazu, zwei Gruppen von Input-Daten mit dem Ergebnis ("true" oder "false") zu vergleichen. Normalerweise werden Daten damit überprüft, bevor sie mittels eines Filters an eine Untergruppe der Zielkomponente übergeben werden.

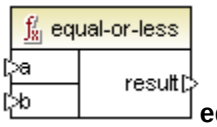
Beim Vergleich von Input-Parametern wählt MapForce den am engsten definierten gemeinsamen Typ für jeden Parameter aus und vergleicht diese dann. Wenn es sich beim gemeinsamen Typ um anySimpleType handelt, so werden beide Input-Parameter als Strings verglichen.

Input-Parameter = **a** | **b** oder **value1** | **value2**
Output-Parameter = result

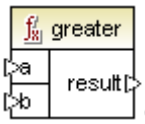
equal
Das Ergebnis ist "true", wenn a=b, andernfalls "false".



equal-or-greater
Das Ergebnis ist "true", wenn a gleich/größer als b ist, andernfalls ist es "false".

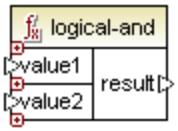


equal-or-less
Das Ergebnis ist "true", wenn a gleich/kleiner als b ist, andernfalls ist es "false".



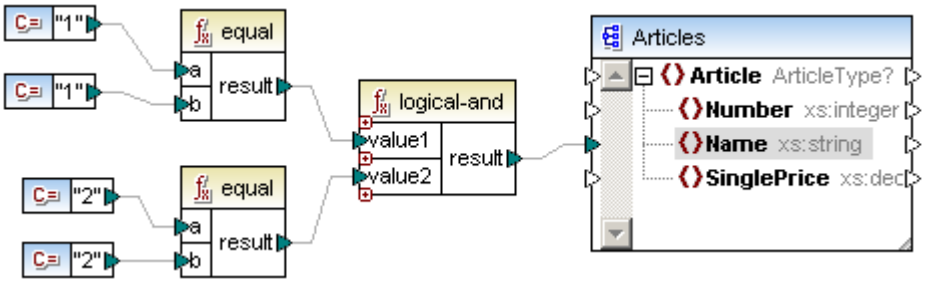
greater
Das Ergebnis ist "true", wenn a größer als b ist, andernfalls ist es "false".

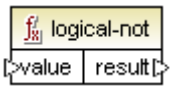
less
Das Ergebnis ist "true", wenn a kleiner als b ist, andernfalls ist es "false".



logical-and

Wenn **sowohl** value1 **als auch** value2 der logical-and Funktion **true** ist, dann ist das Ergebnis true. Andernfalls ist es false.

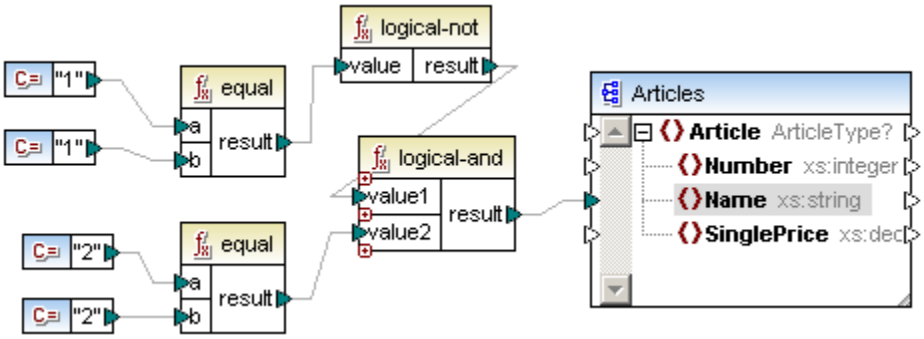


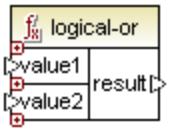


logical-not

Invertiert oder spiegelt den logischen Zustand/das Ergebnis; wenn der Input-Wert "true" ist, ist das Ergebnis der logical-not-Funktion "false". Wenn der Input "false" ist, ist das Ergebnis "true".

In der unten gezeigten logical-not Funktion wird das Ergebnis der equal-Funktion umgekehrt. Die logical-and Funktion gibt nun nur dann "true" zurück, wenn die Booleschen Werte von value1 und value2 unterschiedlich sind, d.h. true-false oder false-true.

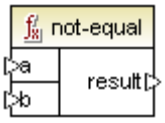




logical-or

Hierfür müssen beide Input-Werte Boolesche Werte sein. Wenn **entweder** value1 **oder** value2 der logical-or Funktion **true** ist, so ist das Ergebnis "true". Wenn beide Werte "false" sind, so ist das Ergebnis "false".

 © 2011 Altova GmbH



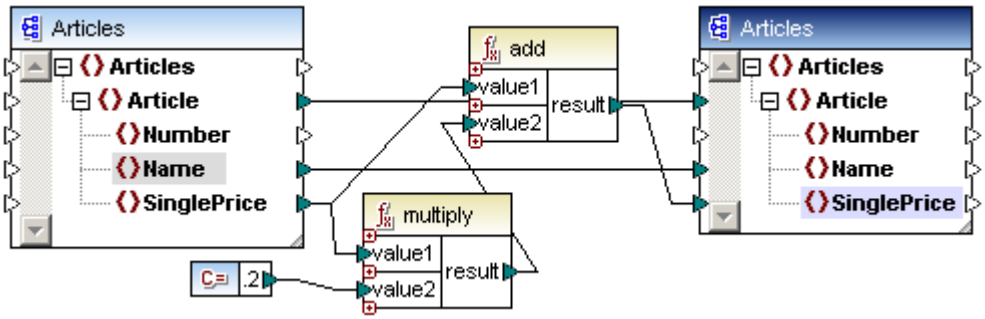
not equal
Das Ergebnis ist "true", wenn a nicht gleich b ist.

math functions (mathematische Funktionen)

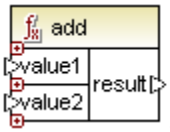
Mathematische Funktionen dienen zur Durchführung grundlegender mathematischer Funktionen an Daten. Beachten Sie, dass diese Funktionen nicht zur Berechnung einer Zeitdauer oder zur Berechnung von Datum und Uhrzeit verwendet werden können.

Input-Parameters = **value1** | **value2**
Output-Parameter = **result**

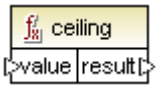
Input-Werte werden zur weiteren Verarbeitung automatisch in Dezimalwerte (decimal) konvertiert.



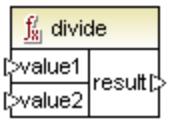
Im obigen Beispiel werden 20 % Ust zu den einzelnen auf die Zielkomponente gemappten Artikeln hinzugefügt.



add
Das Ergebnis ist der Dezimalwert der Addition von value1 zu value2.



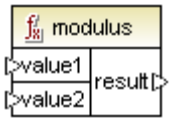
ceiling
Das Ergebnis ist die kleinste Ganzzahl, die größer oder gleich dem nächsthöheren Ganzzahlwert des Input-Dezimalwert ist. Wenn z.B. das Ergebnis einer Divisionsfunktion 11,2 ist, ist das Ergebnis durch Anwendung der ceiling-Funktion 12, also die nächsthöhere Ganzzahl.



divide
Das Ergebnis ist der Dezimalwert der Division von value1 durch value2. Die Präzision des Ergebnisses hängt von der Zielsprache ab. Mit Hilfe der round-precision Funktion können Sie die Präzision des Ergebnisses definieren.

floor

Das Ergebnis ist der nächstniedrigere Ganzzahlwert, der kleiner oder gleich **value** ist, d.h. der nächstniedrigere Ganzzahlwert des Input-Dezimalwerts. Wenn also das Ergebnis der Divisionsfunktion 11,2 ist, so wird das Ergebnis durch Anwendung der floor-Funktion zu 11, d.h. der nächstniedrigeren Ganzzahl.

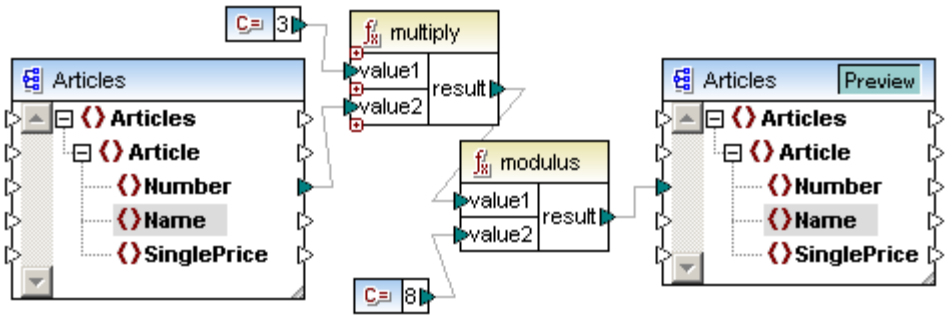


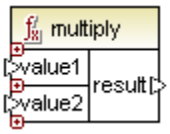
modulus

Das Ergebnis ist der Rest, der sich bei der Division von **value1** durch **value2** ergibt.

Im unten gezeigten Mapping wurden die Zahlen (Number) mit 3 multipliziert und an value1 der modulus-Funktion übergeben. Die Input-Werte sind nun 3, 6, 9 und 12.

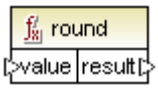
Bei Anwendung/Verwendung von modulus 8 als value2, sind die Restwerte 3, 6, 1 und 4.





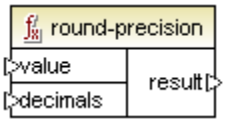
multiply

Das Ergebnis ist der Dezimalwert der Multiplikation von value1 mit value2.



round

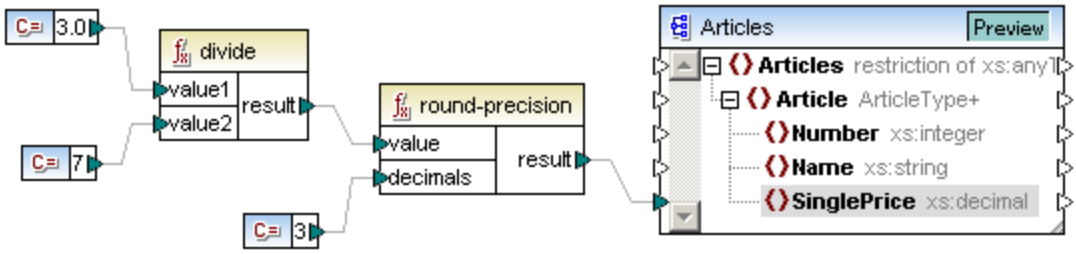
Das Ergebnis ist der Rundungswert auf die nächste Ganzzahl.



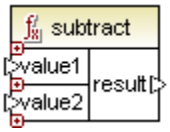
round-precision

Das Ergebnis ist der Dezimalwert der Zahl, gerundet auf die durch "decimals" definierten Dezimalstellen.

 © 2011 Altova GmbH



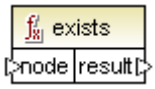
Im oben gezeigten Mapping ist das Ergebnis = 0,429.
Wenn dieses Ergebnis in einer XML-Datei korrekt angezeigt werden soll, muss es auf ein Element vom Typ xs:decimal gemappt werden.



subtract
Das Ergebnis ist der Dezimalwert der Subtraktion von value2 von value1.

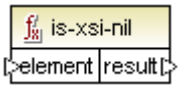
node functions (Node-Funktionen)

Mit Hilfe der Node Test-Funktionen können Sie überprüfen, ob bestimmte Nodes in vielen verschiedenen Arten von Input-Dateien, XML-Schema, Textdateien, Datenbanken, EDI-Dateien und sogar Funktionsergebnissen vorhanden sind. exists überprüft auf nicht leere Sequenzen, d.h. ob ein Node vorhanden ist.

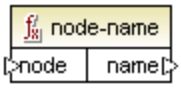


exists
Gibt "true" zurück, wenn der Node vorhanden ist und andernfalls "false".

Ein Beispiel dazu finden Sie unter exists.



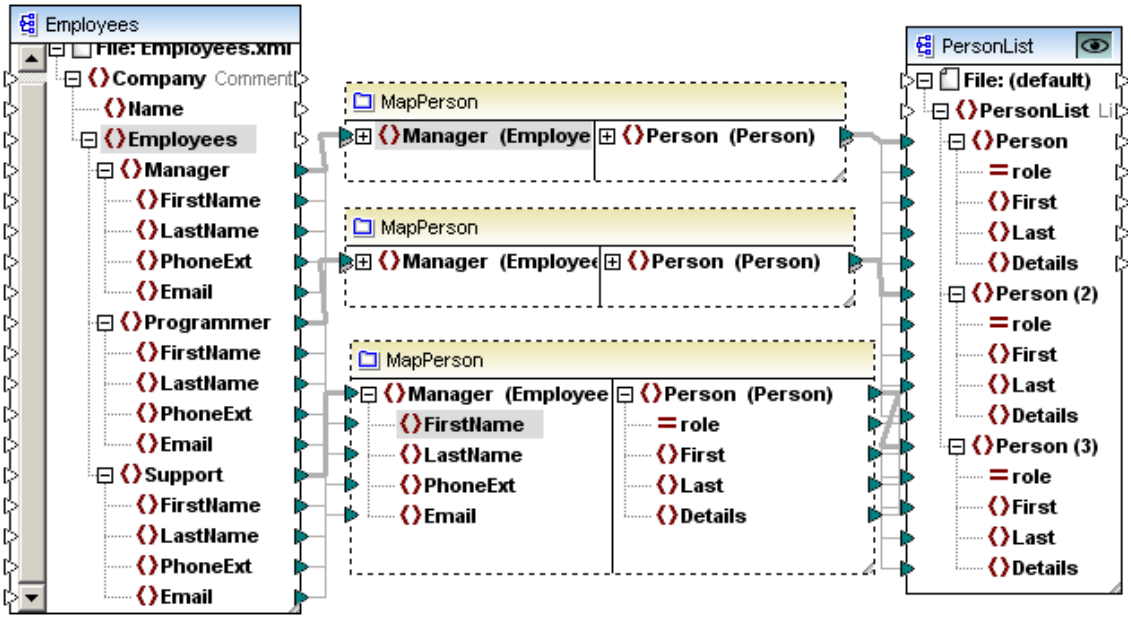
is-xsi-nil
Gibt true (<OrderID>true</OrderID>) zurück, wenn das xsi:nil-Attribut für den Element-Node der Quellkomponente auf "true" gesetzt ist.



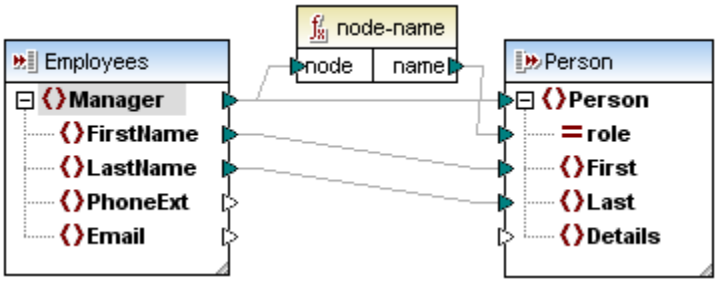
node-name
Gibt den QName des verbundenen Node zurück, außer es handelt sich um einen XML text()-Node. Wenn dies der Fall ist, wird ein leerer QName zurückgegeben. Die Funktion funktioniert nur mit Nodes, die einen Namen haben. Wenn XSLT die Zielsprache ist (die fn:node-name aufruft), wird eine leere Sequenz für Nodes, die keinen Namen haben, zurückgegeben.

- Das Abrufen eines Namens aus Datenbanktabellen bzw. - feldern wird nicht unterstützt.
- XBRL und Excel werden nicht unterstützt
- Das Abrufen eines Names des Datei-Input-Node wird nicht unterstützt
- Webservice-Nodes werden wie XML-Nodes behandelt mit folgenden Ausnahmen:

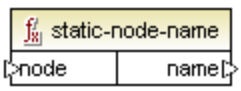
- node-Name anhand eines Teils wird nicht unterstützt.
- node-Name anhand des Root Node ("Output" oder "Input") wird nicht unterstützt



In der benutzerdefinierten Funktion "MapPerson" wird **node-name** verwendet, um den Namen des Input-Node zurückzugeben und in das Role-Attribut einzusetzen. Als Root-Node der Datei Employees.xsd wurde in der benutzerdefinierten Input-Funktion "Manager" definiert.



Manager erhält seine Daten wie oben gezeigt von **außerhalb** der benutzerdefinierten Funktion, wo es entweder Manager, Programmer oder Support sein kann. Dies sind die Daten, die anschließend an das role-Attribut in PersonList übergeben werden.



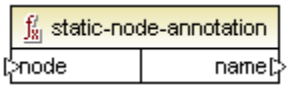
static-node-name

Gibt den String mit dem Names des damit verbundenen Node zurück. Beim Input muss es sich: (i) um eine Quellkomponente oder (ii) um eine Inline-Funktion handeln, die direkt mit einerm Parameter 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-Funktion laufen. Dies ist eine Pseudofunktion, die bei der Generierung durch den aus dem verbundenen Node abgerufenen Text ersetzt wird und daher

 © 2011 Altova GmbH

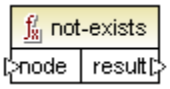
für alle Programmiersprachen zur Verfügung steht.



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 Inline-Funktion handeln, die direkt mit einerm Parameter verbunden ist, der wiederum direkt mit einem Node im aufrufenden Mapping verbunden ist.

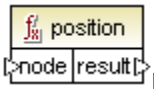
Die Verbindung muss direkt erfolgen und darf nicht über einen Filter oder eine benutzerdefinierte Nicht-Inline-Funktion laufen. Dies ist eine Pseudofunktion, die bei der Generierung durch den aus dem verbundenen Node abgerufenen Text ersetzt wird und daher für alle Programmiersprachen zur Verfügung steht.



not-exists

Gibt "false" zurück, wenn der Node vorhanden ist und andernfalls "true".

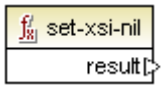
Ein Beispiel dazu finden Sie unter not-exists.



position

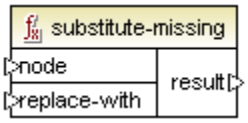
Gibt die Position eines bestimmten Node in einer Sequenz zurück.

Ein Beispiel dazu finden Sie unter position.



set-xsi-nil

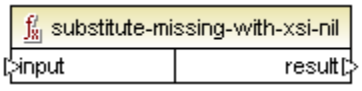
Setzt den Ziel-Node auf xsi:nil.



subsitute-missing

Diese Funktion ist eine praktische Kombination aus "exists" und einer passenden "if-else" Bedingung. Dient zum Mappen des aktuellen Feldinhalts, wenn der Node in der XML-Quelldatei existiert. Andernfalls wird das Datenelement auf den Parameter "replace-with" gemappt.

Ein Beispiel dazu finden Sie unter substitute-missing.



substitute-missing-with-xsi-nil

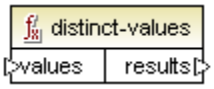
Ersetzt alle fehlenden (oder Null-Werte) der Quellkomponente im Ziel-Node durch das xsi:nil Attribut.

sequence functions (Sequenzfunktionen)

Sequenzfunktionen stehen nicht zur Verfügung, wenn XSLT (XSLT 1.0) ausgewählt ist.

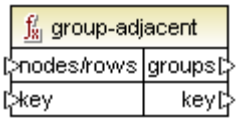
MapForce unterstützt nun Sequenzfunktionen. Mit Hilfe von Sequenzfunktionen können Input-Sequenzen verarbeitet und ihr Inhalt gruppiert werden. Um die Funktion zu gruppieren, wird Wert/Inhalt des Input-Parameters **key** auf Nodes/Zeilen gemappt.

- der Input-Parameter **key** hat bei **group-adjacent** und **group-by** einen beliebigen Datentyp, der in "string" konvertiert werden kann.
- der Input-Parameter **bool** hat bei **group-starting-with** und **group-ending-with** den Typ Boolean.
- Der Output **key** ist der Schlüssel der aktuellen Gruppe.



distinct-values
Dient zum Entfernen doppelt vorhandener Werte aus einer Sequenz und zum Mappen der eindeutigen Datenelemente auf die Zielkomponente.

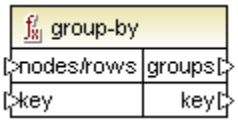
Ein Beispiel dazu finden Sie unter distinct-values.



group-adjacent
Gruppiert die Input-Sequenz **nodes/rows** in Gruppen benachbarter Datenelemente, die denselben **key (**Schlüssel) haben.

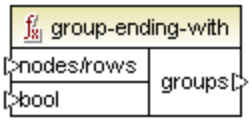
Beachten Sie, dass group-adjacent den **Inhalt** des Node/Datenelements als Gruppierungsschlüssel verwendet!

Ein Beispiel dazu finden Sie unter group-adjacent.



group-by
Gruppiert die Input-Sequenz **nodes/rows** in Gruppen nicht unbedingt benachbarter Datenelemente, die denselben **key (**Schlüssel) haben. Die Gruppen werden in der Reihenfolge ausgegeben, in der der Schlüssel (key) in der Input-Sequenz vorkommt.

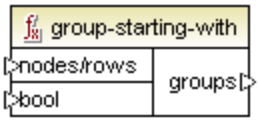
Ein Beispiel dazu finden Sie unter group-by.



group-ending-with
Diese Funktion Gruppiert die Input-Sequenz **nodes/rows** in Gruppen und beendet jede neue Gruppe, wenn **bool** true ist.

Ein Beispiel dazu finden Sie unter group-ending-with.

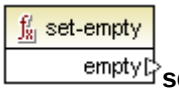
 © 2011 Altova GmbH



group-starting-with

Diese Funktion Gruppiert die Input-Sequenz **nodes/rows** in Gruppen und beginnt jede neue Gruppe, wenn **bool** true ist.

Ein Beispiel dazu finden Sie unter group-starting-with.

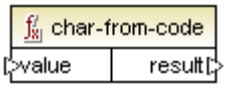


set-empty

Dient zum Löschen von Standardwerten eines XBRL-Dokuments, die weiter oben in der XBRL-Komponente/Taxonomie definiert wurden.

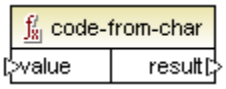
string functions (String-Funktionen)

Mit diesen Funktionen können Sie viele Quelldatenarten mit Hilfe der am häufigsten verwendeten String-Funktionen bearbeiten um Abschnitte zu extrahieren, den String auf darin enthaltene Strings zu überprüfen oder Informationen über Strings abzurufen.



char-from-code

Das Ergebnis die Zeichendarstellung des Unicode-Dezimalwerts von **value**.

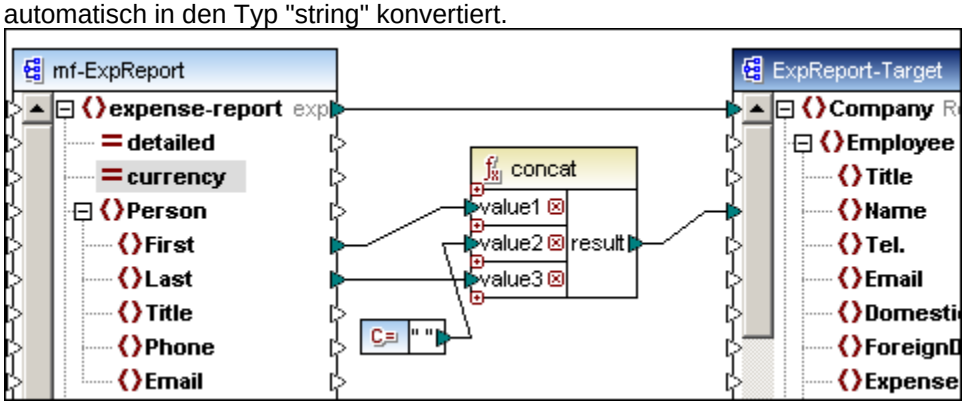


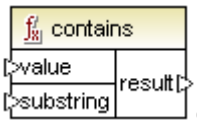
code-from-char

Das Ergebnis ist der Unicode-Dezimalwert des ersten Zeichens von **value**.

concat

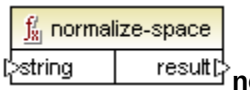
Verkettet zwei oder mehr Werte zu einem einzigen Ergebnisstring. Alle Input-Werte werden automatisch in den Typ "string" konvertiert.





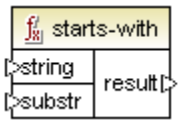
contains

Das Ergebnis ist "true", wenn die an den Parameter "value" gelieferten Daten den im Substring-Parameter definierten String enthalten.



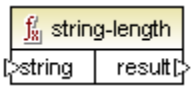
normalize-space

Das Ergebnis ist der normalisierte Input-String, d.h. vorangestellte und nachgestellte Leerzeichen werden entfernt, mehrere Whitespace-Zeichen werden zu einem einzigen Whitespace-Zeichen zusammengefasst.



starts-with

Das Ergebnis ist "true", wenn der Input-String "string" mit substr beginnt, andernfalls ist es "false".



string-length

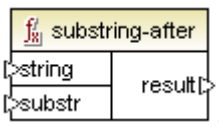
Das Ergebnis ist die Anzahl der vom String-Parameter gelieferten Zeichen.

substring

Das Ergebnis ist der Substring (das String-Fragment) des Parameters "string", wobei "start" die Position des Anfangszeichens und "length" die Länge des Substring definiert.

Wenn der Parameter "length" nicht definiert ist, ist das Ergebnis ein Fragment, das an der Anfangsposition des String beginnt und an der Endposition des String endet.

z.B. substring("56789",2,3) hat das Ergebnis 678.

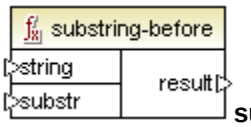


substring-after

Das Ergebnis ist der Rest des Parameters "**string**", wobei die Stelle, an der der Parameter "**substr**" das erste Mal vorkommt, das Anfangszeichen definiert; der Rest des String ist das Ergebnis der Funktion. Das Ergebnis ist ein leerer String, wenn **substr** in **string** nicht vorkommt.

z.B. hat substring-after("2009/01/04","/") als Ergebnis den Substring 01/04. substr ist in diesem Fall das erste "/" Zeichen.

 © 2011 Altova GmbH



substring-before

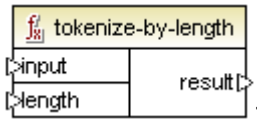
Das Ergebnis ist das String-Fragment des Parameters "string" bis zu dem ersten Zeichen, an dem die Zeichen des Substring zum ersten Mal vorkommen. Das Ergebnis ist ein leerer String, wenn **substr** in **string** nicht vorkommt.
z.B. substring-before ("2009/01/04","/") hat als Ergebnis den Substring 2009. substr ist in diesem Fall das erste "**/"** Zeichen.

tokenize

Als Ergebnis wird der Input-String in eine Sequenz von Blöcken/Abschnitten aufgeteilt, die durch den Parameter "delimiter" definiert werden. Anschließend kann das Ergebnis zur Weiterverarbeitung weitergegeben werden.

Beispiel: Der Input-String ist A,B,C und der "delimiter" (das Trennzeichen) ist "," - das Ergebnis ist A B C.

Ein Beispiel dazu finden Sie unter Tokenize Beispiele.

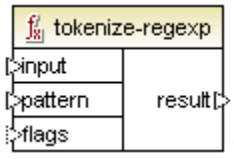


tokenize-by-length

Als Ergebnis wird der Input-String in eine Anzahl von Blöcken/Abschnitten aufgeteilt, die durch den Parameter "length" definiert werden. Anschließend kann das Ergebnis zur Weiterverarbeitung weitergegeben werden.

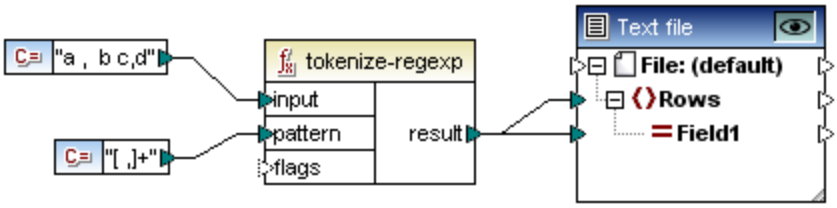
Beispiel: der Input-String ist ABCDEF und length ist "2" - dann ist das Ergebnis AB CD EF.

Ein Beispiel dazu finden Sie unter Tokenize Beispiele.



tokenize-regexp

Als Ergebnis der Funktion wird der Input-String in eine Reihe von Strings aufgeteilt, wobei das Ergebnis des definierten Regular Expression Pattern (Muster) als Trennzeichen fungiert. Die Trennzeichenstrings werden nicht vom result-Parameter ausgegeben. Es können auch optionale Flags verwendet werden.



Im oben gezeigten Beispiel:
ist der **input** String ist eine Folge von Zeichen, die durch Leerzeichen und/oder Kommas getrennt sind, also a , b c,d

definiert das regex **pattern** definiert eine Zeichenklasse **[**"space""comma"**],** von der nur ein einziges Zeichen in einer Zeichenklasse verwendet wird, also entweder space (Leerzeichen) oder comma (Komma).

definiert der **+** Quantifizierer "eine oder mehrere" Instanzen der Zeichenklasse/des Strings.

Der Ergebnisstring ist:

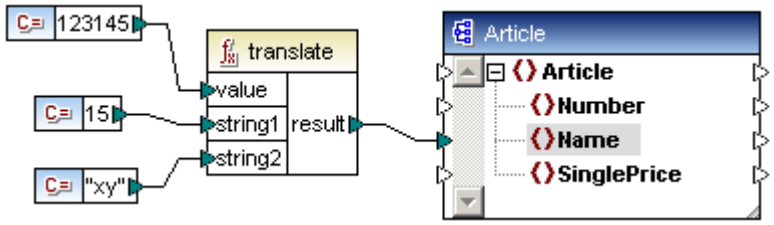
1	a
2	b
3	c
4	d
5	

Bitte beachten Sie: Die Syntax von Regular Expressions unterscheidet sich in den verschiedenen Sprachen geringfügig voneinander. Die Funktion Tokenize-regexp steht in C++ nur in Visual Studio 2008 und höheren Versionen zur Verfügung.

Nähere Informationen zu Regular Expressions finden Sie unter: Regular Expressions.

translate
Die Zeichen von **string1** (Suchstring) werden im Input-String **value** durch die an derselben Position stehenden Zeichen in **string2(Ersetzungsstring**) ersetzt.

Wenn es in String2 keine entsprechenden Zeichen gibt, wird das Zeichen entfernt.



Beispiel
der Input-String ist 123145
der (Such)string1 ist 15
(der Ersetzungs)string2 ist xy

Folglich:
wird jede 1 im Input-String-Wert durch **x** ersetzt
und jede 5 wird im Input-String durch **y** ersetzt

 © 2011 Altova GmbH

Der Ergebnisstring ist **x23x4y**

Wenn string2 leer ist (weniger Zeichen als string1), so wird das Zeichen entfernt.

Beispiel B. 2
Der Input-String aabaacbca
string1 ist "**a**"
string2 ist **""** (leerer String)

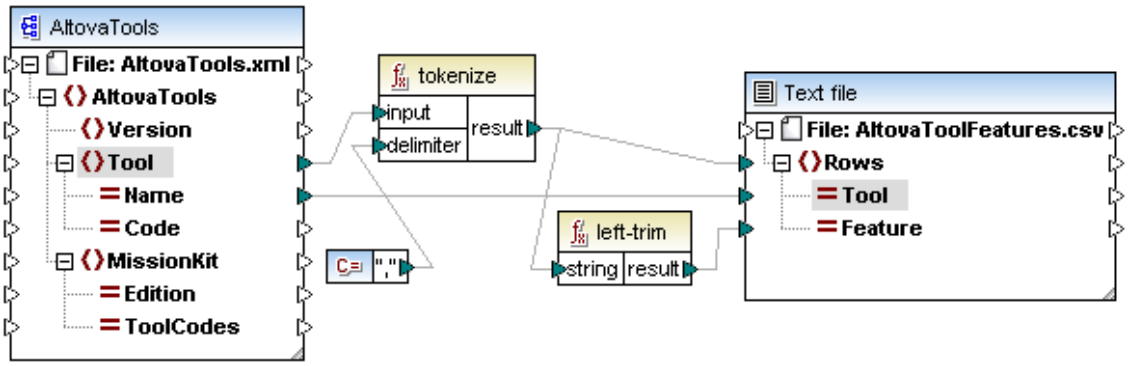
Der Ergebnisstring ist "bcbc"

Beispiel 3
Der Input-String aabaacbca
string1 ist "**ac**"
string2 ist "**ca**"

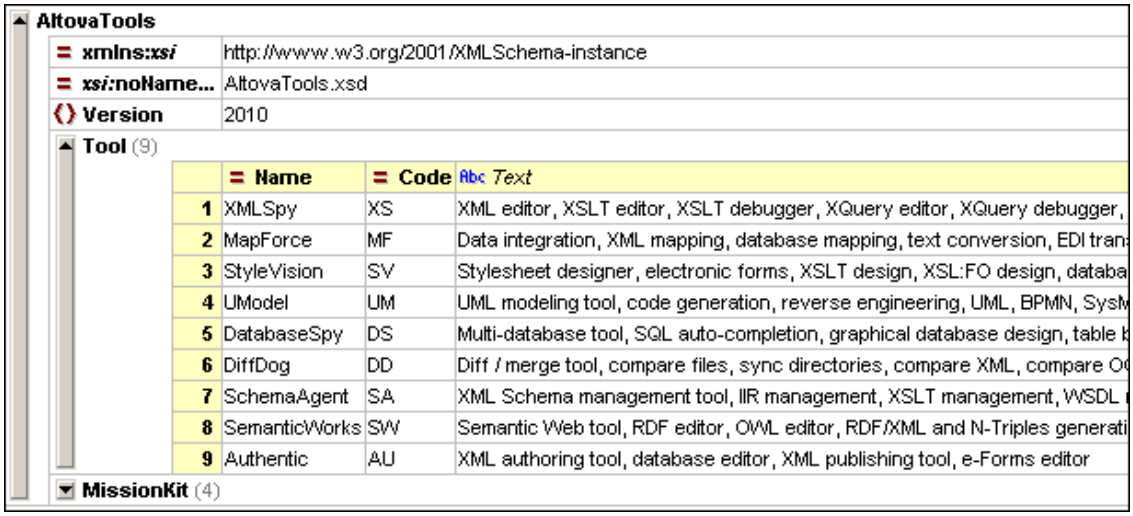
Der Ergebnisstring lautet "ccbccabac"

Tokenize Beispiele

Beispiel **tokenize**:
In der Datei **tokenizeString1.mfd** im Ordner ...\MapForceExamples sehen Sie, wie die tokenize-Funktion verwendet wird.



Unten sehen Sie die XML-Quelldatei. Das Element **Tool** hat zwei Attribute: Name und Code, wobei die Daten des Elements "Tool" aus durch Kommas getrenntem Text bestehen.



AltovaTools

= xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
= xsi:noName...	AltovaTools.xsd
Version	2010

Tool (9)

	= Name	= Code	Abc Text
1	XMLSpy	XS	XML editor, XSLT editor, XSLT debugger, XQuery editor, XQuery debugger,
2	MapForce	MF	Data integration, XML mapping, database mapping, text conversion, EDI tran
3	StyleVision	SV	Stylesheet designer, electronic forms, XSLT design, XSL:FO design, databa
4	UModel	UM	UML modeling tool, code generation, reverse engineering, UML, BPMN, SysM
5	DatabaseSpy	DS	Multi-database tool, SQL auto-completion, graphical database design, table b
6	DiffDog	DD	Diff / merge tool, compare files, sync directories, compare XML, compare O
7	SchemaAgent	SA	XML Schema management tool, IIR management, XSLT management, WSDL
8	SemanticWorks	SW	Semantic Web tool, RDF editor, OWL editor, RDF/XML and N-Triples generati
9	Authentic	AU	XML authoring tool, database editor, XML publishing tool, e-Forms editor

MissionKit (4)

Funktionsweise des Mappings:

- Die tokenize-Funktion erhält Daten aus dem Datenelement **Tool** und teilt die Daten mit Hilfe des Komma-Trennzeichens "**,**" in separate Blöcke auf. D.h., der erste Block ist "XML editor".
- Der result-Parameter wird auf das Datenelement **Rows** in der Zielkomponente gemappt. Dies wird für alle Elemente in der Quelldatei durchgeführt.
- Der result-Parameter wird außerdem auch auf die **left-trim** Funktion gemappt, die aus den einzelnen Blöcken die vorangestellten Leerzeichen entfernt.
- Das Ergebnis des left-trim-Parameters (die einzelnen Blöcke) werden auf das Datenelement **Feature** der Zielkomponente gemappt.
- Als Ausgabedatei der Zielkomponente wurde eine CSV-Datei definiert (AltovaToolFeatures.csv), in der als Feldtrennzeichen das Semikolon verwendet wird (Doppelklicken Sie auf die Komponente, um die Einstellungen zu sehen).

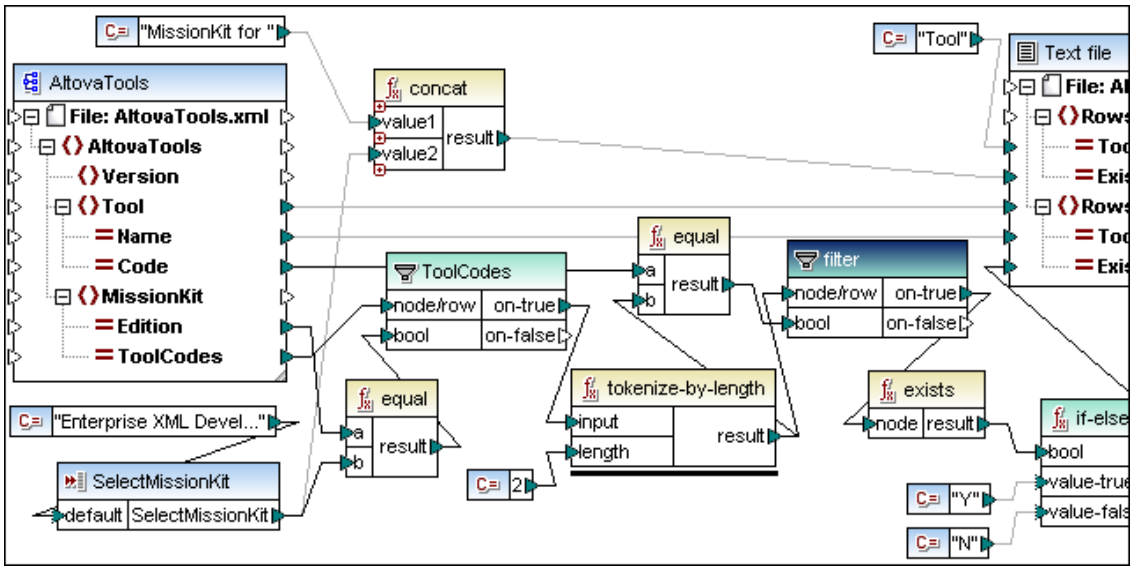
Ergebnis des Mappings:

- Für jedes Tool-Element der Quelldatei
- wird der (Tool) Name auf das Datenelement "Tool" in der Zielkomponente gemappt
- werden die einzelnen Blöcke des aufgeteilten Inhalts von "Tool" an das (Tool Name) Feature-Datenelement angehängt.
- So erhält z.B. des erste Tool, XMLSpy, den ersten Feature-Block "XML editor"
- Dieser Vorgang wird für alle Blöcke des aktuellen Tools wiederholt und anschließend für alle Tools.
- Wenn Sie auf die Schaltfläche "Ausgabe" klicken, sehen Sie das unten gezeigte Ergebnis.

```
1   Tool;Feature
2   XMLSpy;XML editor
3   XMLSpy;XSLT editor
4   XMLSpy;XSLT debugger
5   XMLSpy;XQuery editor
6   XMLSpy;XQuery debugger
7   XMLSpy;XML Schema / DTD editor
8   XMLSpy;WSDL editor
9   XMLSpy;SOAP debugger
10  MapForce;Data integration
11  MapForce;XML mapping
12  MapForce;database mapping
```

 © 2011 Altova GmbH

Beispiel tokenize-by-length
In der Datei **tokenizeString2.mfd** im Ordner ...\MapForceExamples sehen Sie, wie die **tokenize-by-length** Funktion verwendet wird.



Die unten gezeigte XML-Quelldatei ist dieselbe wie die im vorhergehenden Beispiel verwendete. Das Element **MissionKit** hat ebenfalls zwei Attribute: **Edition** und **ToolCodes** aber keinen Inhalt für das Element "MissionKit".

Tool (9)

	Name	Code	Text
1	XMLSpy	XS	XML editor, XSLT editor, XSLT debugger, XQuery editor, XQuery debugger, XML S
2	MapForce	MF	Data integration, XML mapping, database mapping, text conversion, EDI translator,
3	StyleVision	SV	Stylesheet designer, electronic forms, XSLT design, XSL:FO design, database rep
4	UModel	UM	UML modeling tool, code generation, reverse engineering, UML, BPMN, SysML, pro
5	DatabaseSpy	DS	Multi-database tool, SQL auto-completion, graphical database design, table brows
6	DiffDog	DD	Diff / merge tool, compare files, sync directories, compare XML, compare OOXML,
7	SchemaAgent	SA	XML Schema management tool, IIR management, XSLT management, WSDL manag
8	SemanticWorks	SW	Semantic Web tool, RDF editor, OWL editor, RDF/XML and N-Triples generation and
9	Authentic	AU	XML authoring tool, database editor, XML publishing tool, e-Forms editor

MissionKit (4)

	Edition	ToolCodes
1	Enterprise Software Architects	XSMFSVUMDSDDSASW
2	Professional Software Architects	XSMFSVUMDS
3	Enterprise XML Developers	XSMFSVDDSASW
4	Professional XML Developers	XSMFSV

Ziel des Mappings:
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 erhält ihren Standard-Input von einer Konstantenkomponente, in diesem Fall von "Enterprise XML Developers".
- Mit Hilfe der **equal**-Funktion wird der Input-Wert mit dem Wert von "**Edition**" verglichen. Das Ergebnis wird an den **bool**-Parameter des ToolCodes-Filters übergeben.

- 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.

Funktionsweise und Ergebnis der tokenize-by-length Funktion:

- Der ToolCodes **input**-Wert XSMFSVDDSASW wird in mehrere Blöcke bestehend aus je zwei Zeichen aufgeteilt, die durch den **length** Parameter 2 definiert werden. Das Ergebnis sind 6 Blöcke.
- Jeder (in den Parameter b platzierte) Block der equal-Funktion 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 **exist**-Funktion überprüft nun die Datei auf existing/non-existing (existierende/nicht existente) Nodes, die vom **on-true** Parameter der Filter-Komponente an die **exist**-Funktion übergeben werden.

  Existierende Nodes sind jene, für die eine **Übereinstimmung** zwischen dem **ToolCodes**-Block und dem Codewert besteht.

  Nicht existente Nodes sind Fälle, in denen es **keine Übereinstimmung** zwischen einem **ToolCodes**-Block und einem Codewert gibt.

- Die bool-Ergebniss der **exists**-Funktion werden an die **if-else** Funktion übergeben, die ein Y an das Ziel übergibt, wenn der Node vorhanden ist oder ein N, wenn der Node nicht existiert.

Ergebnis des Mappings:

```
1   Tool;MissionKit for Enterprise XML Developers
2   XMLSpy;Y
3   MapForce;Y
4   StyleVision;Y
5   UModel;N
6   DatabaseSpy;N
7   DiffDog;Y
8   SchemaAgent;Y
9   SemanticWorks;Y
10  Authentic;N
11
```

Regular Expressions

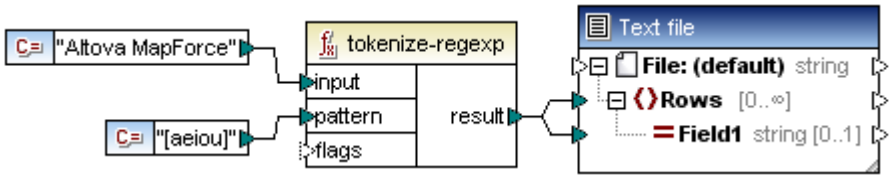
MapForce kann Regular Expressions im **pattern**-Parameter der Funktionen **match-pattern** und **tokenize-regexp** verwenden, um bestimmte Strings des Input-Parameters zu finden.

Die Syntax und Semantik von Regular Expressions für XSLT und XQuery entspricht den in http://www.w3.org/TR/xmlschema-2/ definierten Regeln. Beachten Sie bitte, dass es in der Syntax von Regular Expressions geringfügige Unterschiede zwischen den verschiedenen Programmiersprachen gibt.

 © 2011 Altova GmbH

Terminologie:

input	der String, an dem der regex-Ausdruck ausgeführt wird
pattern	die Regular Expression
flags	optionaler Parameter, um zu definieren, wie die Regular Expression zu interpretieren ist
result	das Ergebnis der Funktion



Tokenize-regexp gibt eine Stringsequenz zurück. Mit der Verbindung des Datenelements "Rows" wird in der Sequenz eine Zeile pro Datenelement erstellt.

regex-Syntax

Literals e.g. Einzelnes Zeichen:
"a" ist z.B. der einfachste regex-Ausdruck. Er findet die erste Instanz des Buchstaben "a" in einem String.

Zeichenklassen []
Dies ist eine Gruppe von Zeichen, die innerhalb von eckigen Klammern steht.

Es wird immer nur jeweils ein einziges der Zeichen innerhalb der eckigen Klammern gefunden.

pattern **[aeiou]**
Findet einen kleingeschriebenen Vokal.

pattern **[mj]ust**
Findet must oder just.

Bitte beachten Sie, dass bei "pattern" die Groß- und Kleinschreibung eine Rolle spielt, daher wird mit einem kleingeschriebenen **a** der Großbuchstabe **A** nicht gefunden.

Zeichenbereich **[a-z]**
Erstellt einen Bereich zwischen den beiden Zeichen. Es wird immer nur jeweils eines der Zeichen gefunden.

pattern **[a-z]**
Findet alle Kleinbuchstaben zwischen a und z.

Negierte Klassen **[^]**
mit dem Zikumflexzeichen als erstem Zeichen nach der öffnenden Klammer wird die Zeichenklasse negiert.

pattern **[^a-z]**
Findet alle Zeichen außerhalb der Zeichenklasse, einschließlich von Zeilenschaltungen

(newline).

Metazeichen "."
Punkt-Metazeichen
steht für **jedes beliebige einzelne** Zeichen (mit Ausnahme von "newline")

pattern **.**
Findet jedes einzelne Zeichen.

Quantifizierer ? + * {}
Quantifizierer definieren, wie oft eine regex-Komponente im Input-String wiederholt werden muss, damit eine Übereinstimmung gefunden wird.

?

null oder eins	der vorhergehende String/Abschnitt ist optional

+

eines oder mehrere	der vorhergehende String/Abschnitt darf ein Mal oder öfter gefunden werden

*****

null oder mehr	der vorhergehende String/Abschnitt darf null Mal oder öfter gefunden werden

{}

min / max Wiederholungen	die Anzahl der Wiederholungen mit denen ein String/ Abschnitt gefunden werden muss z.B. wird mit "mo{1,3}" mo, moo, mooo gefunden.

()
Subpatterns (Untergruppen)
Um Teile eines regex-Ausdrucks zu gruppieren, werden Klammern verwendet.

|
Alternation/oder gestattet das Testen von Unterausdrücken von links nach rechts.
(horse|make) sense - findet "horse sense" oder "make sense"

Flags
Flags sind optionale Parameter, die definieren, wie die Regular Expression interpretiert werden soll. Zum Definieren der Optionen dienen einzelne Buchstaben, d.h., wenn das jeweilige Zeichen vorhanden ist, ist diese Option aktiv. Die Buchstaben können in jeder Reihenfolge vorkommen und auch wiederholt werden.

s
Falls vorhanden, wird der Suchvorgang im "dot-all"-Modus durchgeführt.

Das Metazeichen "**.**" steht für jedes beliebige Zeichen. Wenn der String "hello" und "world" in zwei *verschiedenen* Zeilen vorkommt, wird mit der Regular Expression "hello*world" nur gefunden, wenn das Flag/Zeichen **s** gesetzt wurde.

m
Falls vorhanden, wird der Suchvorgang im mehrzeiligen Modus durchgeführt.

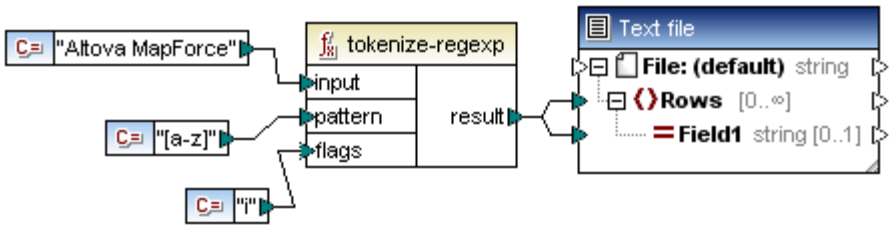
 © 2011 Altova GmbH

Im mehrzeiligen Modus findet das Zirkumflexzeichen **^** den Beginn **jeder** Zeile, d.h. den Beginn des gesamten Strings **und** das erste Zeichen nach dem "newline" Zeichen.

Das Dollarzeichen **$** findet das Ende **jeder** Zeile, d.h. das Ende des gesamten Strings und das Zeichen unmittelbar vor dem "newline" Zeichen.

Newline (Neue Zeile) ist das Zeichen #x0A.

i
Falls vorhanden, wird der Suchvorgang im Modus "Groß-/Kleinschreibung ignorieren" durchgeführt.
Mit der Regular Expression [a-z] plus dem **i** Flag würden alle Buchstaben von a-z und A-Z gefunden.



x
Falls vorhanden, werden Leerzeichen und andere Whitespace-Zeichen vor dem Suchvorgang aus dem Input-String entfernt. Die Whitespace-Zeichen sind #x09, #x0A, #x0D und #x20.

Ausnahme:
Whitespace-Zeichen innerhalb von Zeichenklassen-Ausdrücken werden nicht entfernt z.B. [#x20].

Bitte beachten Sie:
Die komplexen Funktionalitäten der regex-Syntax können sich bei der Generierung von Code in den verschiedenen Programmiersprachen etwas unterschiedlich verhalten. Lesen Sie bitte in der entsprechenden regex-Dokumentation zu Ihrer Sprache nach.

15.5.2 xpath2

XPath2-Funktionen stehen zur Verfügung, wenn entweder XSLT2 oder XQuery ausgewählt ist.

- accessor functions (accessor-Funktionen)
- anyURIfunctions (Alle URI-Funktionen)
- boolean functions (Boolesche Funktionen)
- constructors (Konstruktoren)
- context functions (Kontextfunktionen)
- durations, date and time functions (Zeitdauer-, Datums- und Uhrzeitfunktionen)
- node functions (Node-Funktionen)
- numeric functions (numerische Funktionen)
- QName functions (QName-Funktionen)
- string functions (String-Funktionen)

accessors (Accessor-Funktionen)

Die folgenden accessor-Funktionen stehen zur Verfügung:

base-uri
Die `base-uri` Funktion erhält ein Node-Argument als Input und gibt die URI der XML-Ressource zurück, die den Node enthält. Die Ausgabe hat den Typ `xs:string`. MapForce gibt einen Fehler zurück, wenn kein Input-Node bereitgestellt wird.

Die Ausgabe hat den Typ `xs:string`.

document-uri
Nicht implementiert.

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:lokjalerName`, 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` Funktion (in der Bibliothek der Funktionen im Zusammenhang mit QName).

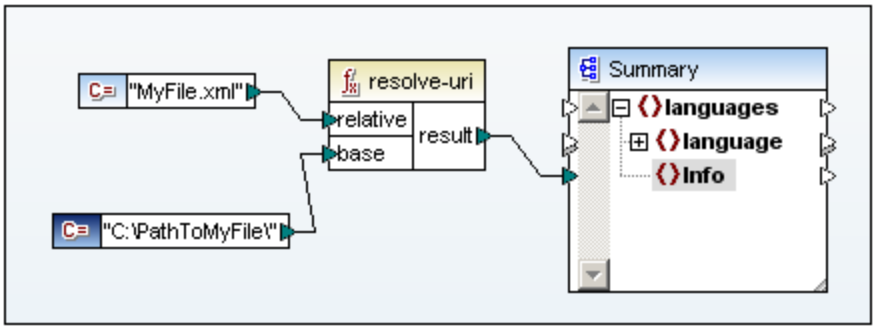
string
Die `string` Funktion funktioniert wie der `xs:string` Konstruktor: Sie konvertiert ihr Argument in `xs:string`. Wenn das Input-Argument ein Wert eines Typs "atomic" ist (z.B. `xs:decimal`), wird dieser Wert in einen Wert vom Typ `xs:string` konvertiert. Wenn das Input-Argument ein Node ist, wird der String-Wert des Node extrahiert. (Der String-Wert eines Node ist eine Verkettung der Werte der untergeordneten Werte des Node.)

anyURI functions (Alle URI-Funktionen)

Die `resolve-uri` Funktion erhält eine URI als erstes Argument (Datentyp `xs:string`) und löst diese anhand der URI im zweiten Argument (Datentyp `xs:string`) auf.

Das Ergebnis (Datentyp `xs:string`) ist eine kombinierte URI. Auf diese Art kann eine relative URI (das erste Argument) durch Auflösung der URI anhand der Basis URI in eine absolute URI konvertiert werden.

 © 2011 Altova GmbH



In der Abbildung oben liefert das erste Argument die relative URI, das zweite Argument die Basis-URI. Die aufgelöste URI ist eine Verkettung der Basis-URI und der relativen URI, also `C:\PfadZuMeinerDatei\MeineDatei.xml`.

Anmerkung: Beide Argumente haben den Datentyp `xs:string`. Die Kombination erfolgt durch Behandlung beider Inputs als Strings. Es gibt daher keine Möglichkeit zu überprüfen, ob die durch diese URI definierten Ressourcen tatsächlich vorhanden sind. MapForce gibt einen Fehler zurück, wenn das zweite Argument fehlt.

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.

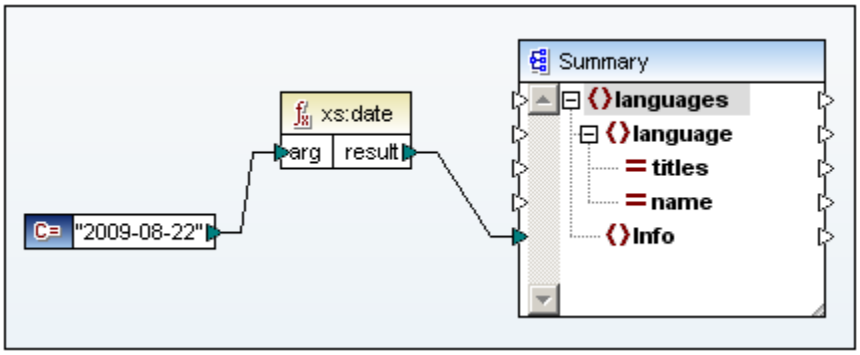
true
Fügt den Booleschen Wert "true" ein.

false
Fügt den Booleschen Wert "false" ein.

constructors (Konstruktoren)

Die Funktionen in der XPath 2.0-Funktionsbibliothek unter der Überschrift "Constructors" konstruieren anhand des Input-Texts bestimmte Datentypen. Normalerweise muss der Input-Text das dem zu konstruierenden Datentyp entsprechende lexikalische Format haben, da die Transformation sonst nicht funktioniert.

Wenn Sie z.B. einen `xs:date` Datentyp erstellen möchten, verwenden Sie die `xs:date` Konstruktorfunktion. Der Input-Text muss das lexikalische Format des `xs:date` Datentyps haben, also `JJJJ-MM-TT` (*Abbildung unten*).



In der Abbildung oben wurde als Input-Argument der Funktion eine String-Konstante (`2009-08-22`) verwendet. Dieser Input hätte auch aus einem Node im Quelldokument abgerufen werden können.

Die `xs:date` Funktion gibt den Input-Text (`2009-08-22`), der den Datentyp `xs:string` (definiert in der *Constant* Komponente) hat, als Ausgabe mit dem Datentyp `xs:date` zurück.

Wenn Sie in einem Funktionskasten den Mauszeiger über das Input-Argument platzieren, wird der erwartete Datentyp des Arguments in einem Popup-Fenster angezeigt.

context functions (Kontextfunktionen)

Die Bibliothek "context functions" enthält Funktionen, die das aktuelle Datum und die Uhrzeit bereitstellen, die Standard-Collation, die vom Prozessor verwendet wird, die Größe der aktuellen Sequenz und die Position des aktuellen Node.

Datums- und Uhrzeit-Funktionen

Die Funktionen `current-date`, `current-time` und `current-dateTime` erhalten kein Argument und geben das aktuelle Datum und/oder die aktuelle Uhrzeit der Systemuhr zurück.

Der Datentyp des Ergebnisses ist von der jeweiligen Funktion abhängig: `current-date` gibt `xs:date` zurück, `current-time` gibt `xs:time` zurück und `current-dateTime` gibt `xs:dateTime` zurück.

default-collation

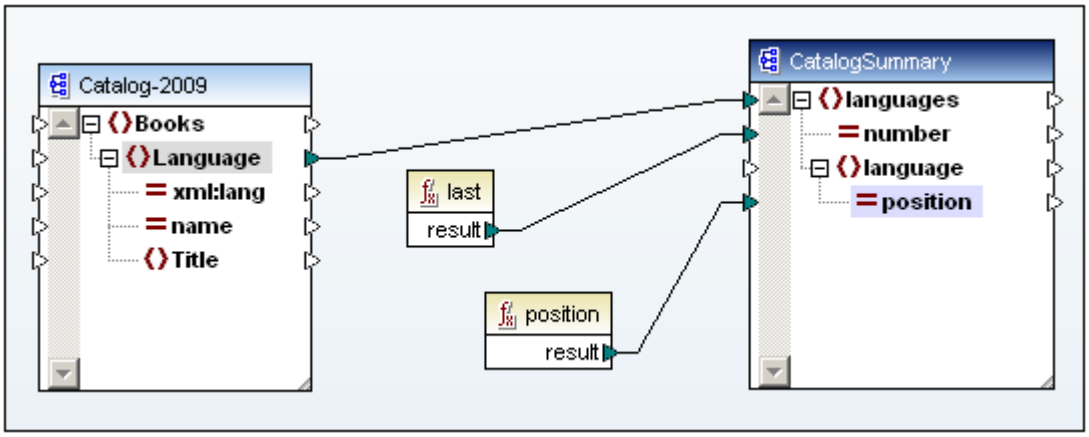
Die Funktion default-collation erhält kein Argument und gibt die Standard-Collation zurück, d.h. die Collation, die verwendet wird, wenn keine Collation für eine Funktion, für die eine solche definiert werden kann, festgelegt ist.

Der Altova XSLT 2.0 Prozessor unterstützt nur die Unicode Codepoint Collation. Vergleiche wie für die Funktionen `fn:max` und `fn:min` basieren auf dieser Collation.

last, position

Die Funktionen `last` und `position` erhalten kein Argument. Die `last` Funktion gibt die Position des letzten Node im Kontext-Nodeset zurück. Die `position` Funktion gibt die Position des aktuellen Node im Nodeset, das gerade verarbeitet wird, zurück.

Das Kontext-Nodeset bei den Nodes, an die die Funktion gerichtet ist, ist das Nodeset, auf das die Funktionen angewendet wird. In der Abbildung unten ist das Nodeset der `Language` Elemente das Kontext-Nodeset für die Funktionen `last` und `position`.



 © 2011 Altova GmbH

Im obigen Beispiel gibt die `last` Funktion die Position des letzten Node des Kontext-Nodeset (des Nodeset der `Language` Elemente) als den Wert des `number` Attributs zurück. Dieser Wert entspricht auch der Größe des Nodeset, da er anzeigt, wie viele Nodes im Nodeset enthalten sind.

Die `position` Funktion gibt die Position des `Language` Node, der gerade verarbeitet wird, zurück. Für jeden `Language` Element-Node wird die Position innerhalb des Nodes der `Langauge` Elemente im Attribut-Node `language/@position` ausgegeben.

Wir empfehlen Ihnen, die Funktionen **position** und **count** aus der **core** Bibliothek zu verwenden.

durations, date and time functions (Zeitdauer-, Datums- und Uhrzeitfunktionen)

Mit Hilfe der Zeitdauer-, Datums- und Uhrzeitfunktionen können Sie das Datum und die Uhrzeit für die Zeitzone anpassen, bestimmte Komponenten aus Datums- und Uhrzeitdaten extrahieren und eine Datums- und Uhrzeit-Einheit von einer anderen subtrahieren.

Die 'Adjust-to-Timezone' Funktionen
Jede dieser Funktionen erhält ein Datum, eine Uhrzeit oder ein Datum- und eine Uhrzeit als erstes Argument und passt die Input-Daten je nach dem Wert des zweiten Arguments 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:

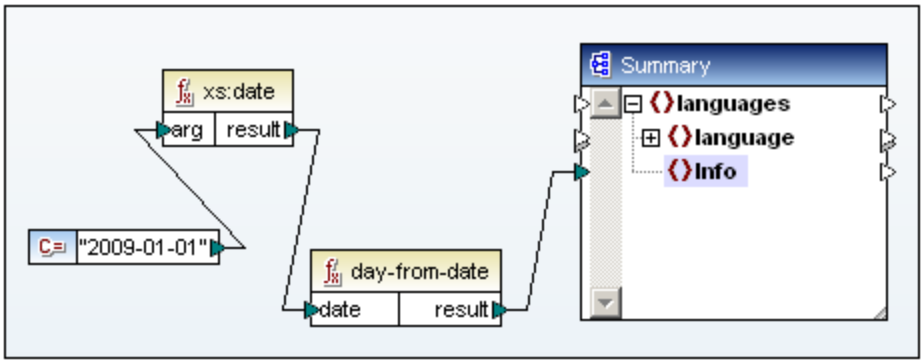
- Das Argument "timezone" (das zweite Argument in der Funktion) ist vorhanden: Das Ergebnis enthält die im zweiten Argument definierte Zeitzone. Die Zeitzone im zweiten Argument wird hinzugefügt.
- Das Argument "timezone" (das zweite Argument in der Funktion) fehlt: Das Ergebnis enthält die implizite Zeitzone, also die Zeitzone des Systems. Die Zeitzone des Systems wird hinzugefügt.
- Das Argument "timezone" (das zweite Argument in der Funktion) ist leer: Das Ergebnis enthält keine Zeitzone.

Wenn das erste Argument eine Zeitzone enthält, (z.B. das Datum `2009-01-01+01:00` oder die Uhrzeit `14:00:00+01:00`), ergeben sich die folgenden Möglichkeiten:

- Das Argument "timezone" (das zweite Argument in der Funktion) ist vorhanden: Das Ergebnis enthält die im zweiten Argument definierte Zeitzone. Die ursprüngliche Zeitzone wird durch die im zweiten Argument definierte Zeitzone ersetzt.
- Das Argument "timezone" (das zweite Argument in der Funktion) fehlt: Das Ergebnis enthält die implizite Zeitzone, also die Zeitzone des Systems. Die ursprüngliche Zeitzone wird durch die Zeitzone des Systems ersetzt.
- Das Argument "timezone" (das zweite Argument in der Funktion) ist leer: Das Ergebnis enthält keine Zeitzone.

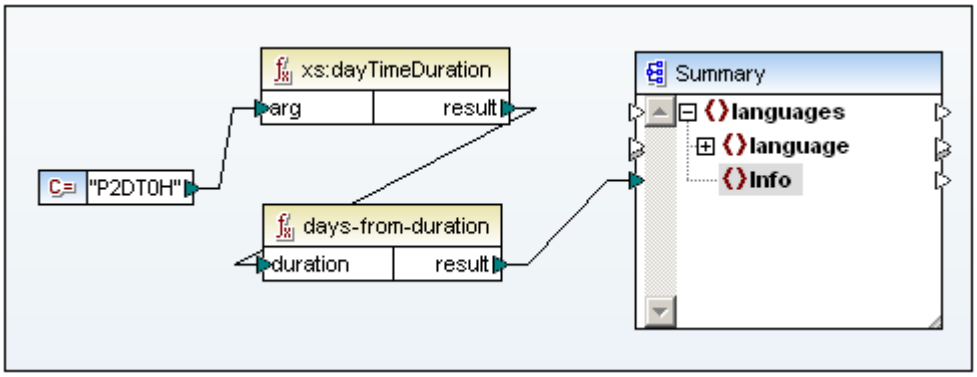
Die 'From' Funktionen
Die einzelnen 'From' Funktionen extrahieren eine bestimmte Komponente aus: (i) Datums- oder Uhrzeitdaten, und (ii) Zeitdauerdaten. Die Ergebnisse haben den Datentyp xs:decimal.

Als Beispiel für die Extraktion einer Komponente aus Datums- oder Uhrzeitdaten wird hier die `day-from-date` Funktion verwendet (*Abbildung unten*).



Das Input-Argument ist ein Datum (`2009-01-01`) vom Typ `xs:date`. Die `day-from-date` Funktion extrahiert die Tageskomponente des Datums (`1`) als einen xs:decimal Datentyp.

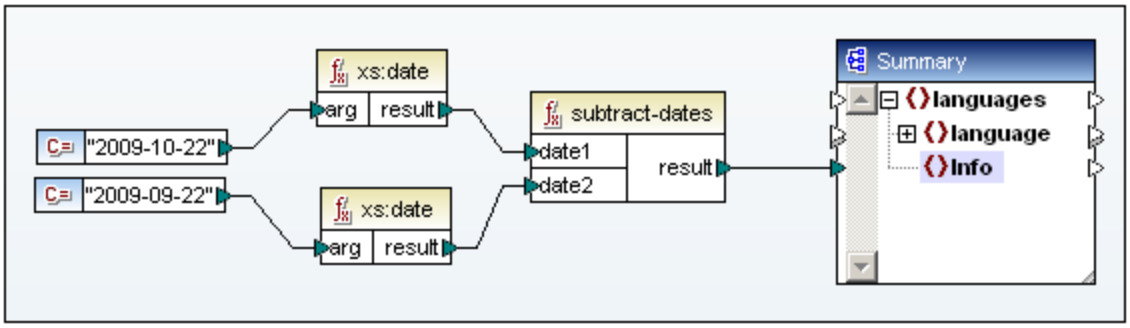
Wenn Zeitkomponenten aus einer Zeitdauer extrahiert werden, muss die Zeitdauer entweder als `xs:yearMonthDuration` (zum Extrahieren von Jahren und Monaten) oder als `xs:dayTimeDuration` (zum Extrahieren von Tagen, Stunden, Minuten und Sekunden) definiert werden. Das Ergebnis hat den Typ `xs:decimal`. In der Abbildung unten sehen Sie `dayTimeDuration` von `P2DT0H` als Input für die Funktion `days-from-duration`. Das Ergebnis ist xs:decimal Datentyp `2`.



Die 'Subtract' Funktionen
Mit Hilfe der Subtraktionsfunktionen können Sie einen Uhrzeitwert von einem anderen subtrahieren und einen Zeitdauerwert errechnen. Die drei Subtraktionsfunktionen sind: `subtract-dates`, `subtract-times`, `subtract-dateTimes`.

In der Abbildung unten sehen Sie, wie ein Datum mit Hilfe der subtract-dates-Funktion von einem anderen Datum subtrahiert wird (`2009-10-22` minus `2009-09-22`). Das Ergebnis ist der dayTimeDuration Wert `P30D`.

 © 2011 Altova GmbH

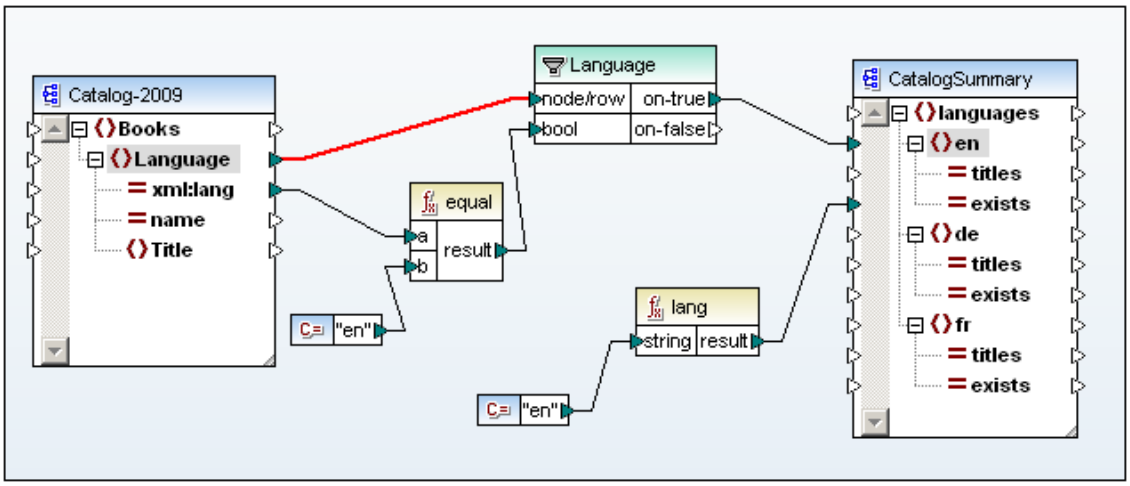


Anmerkung: Wenn Sie den Mauszeiger in einem Funktionskasten über das Input-Argument platzieren, wird der erwartete Datentyp in einem Popup-Fenster angezeigt.

node functions (Node-Funktionen)

Die folgenden Node-Funktionen stehen zur Verfügung:

lang
Die `lang` Funktion erhält ein String-Argument, das einen Sprachcode definiert (wie z.B. `en`). Abhängig davon, ob der Kontext-Node ein `xml:lang` Attribut mit einem Wert hat, der mit dem Argument der Funktion übereinstimmt, wird entweder `true` oder `false` ausgegeben.



Beachten Sie in der Abbildung oben Folgendes:

1. Im Quellschema hat das Element `Language` ein `xml:lang` Attribut.
2. `Language` Nodes werden gefiltert, sodass nur jene `Language` Nodes mit dem `xml:lang` Wert `en` verarbeitet werden (der Filter ist in der `equal` Funktion definiert).
3. Der `Language` Node ist der Kontext-Node an der Stelle, an der im Ausgabedokument das Element `en` erstellt wird.
4. Die Ausgabe der `lang` Funktion (`true` oder `false`) wird an den `en/@exists` Attribut-Node der Ausgabe gesendet. Das Argument der Funktion wird durch die String-Konstante `en` bereitgestellt. Die `lang` Funktion überprüft anschließend, ob der Kontext-Node an dieser Stelle (dem `Language` Element) ein `xml:lang` Attribut mit dem Wert `en` (dem Argument der Funktion) hat. Wenn ja, so wird `true` zurückgegeben, andernfalls `false`.

local-name, name, namespace-uri
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 ist die URI des Namespace an den das Präfix `altova:` gebunden ist (z.B. `http://www.altova.com/examples`).

Jeder 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` Funktion) angewendet.
- ein Argument, das ein Node sein muss: Die Funktion wird auf den angegebenen Node angewendet.

Das Ergebnis dieser insgesamt sechs Varianten ist ein String.

number
Konvertiert einen Input-String in eine Zahl. Konvertiert auch einen Booleschen Input in eine Zahl.

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 einzigen Typen, die in Zahlen konvertiert werden können, sind Boolesche Werte, Strings und andere numerische Typen. Nicht numerische Input-Wert (wie z.B. ein nicht numerischer String) führen zum Ergebnis `NaN` (Not a Number).

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` Funktion) angewendet.
- ein Argument, das ein Node sein muss: Die Funktion wird auf den angegebenen Node angewendet.

numeric functions (numerische Funktionen)

Die folgenden numerischen Funktionen stehen zur Verfügung:

abs
Die `abs` Funktion erhält einen numerischen Wert als Input und gibt seinen absoluten Wert als Dezimalzahl zurück. Wenn das Input-Argument z.B. `-2` oder `+2` ist, so gibt die Funktion den Wert `2` zurück.

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 `2,141`. 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`.

 © 2011 Altova GmbH

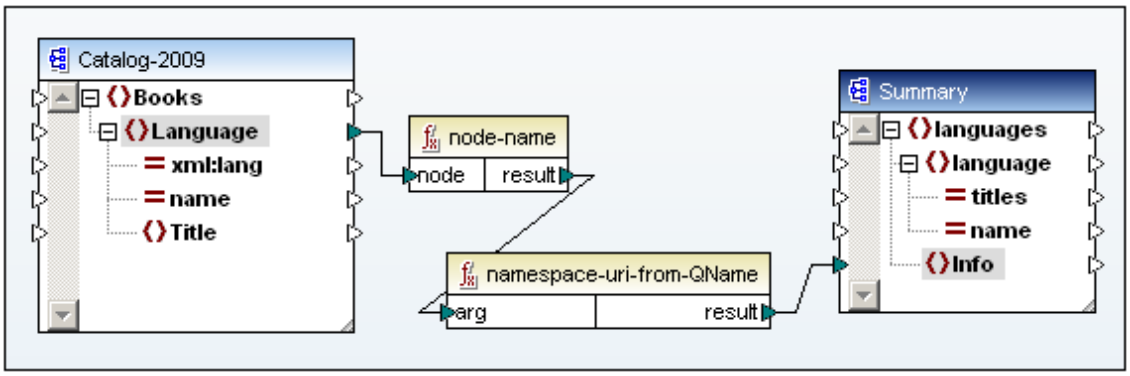
qname-related functions (QName-Funktionen)

Es gibt zwei QName-Funktionen, die ähnlich funktionieren: `local-name-from-QName` und `namespace-uri-from-QName`.

Beide Funktionen erhalten einen erweiterten QName (in Form eines String) als Input-Argument und erzeugen den lokalen Namen bzw. den Namespace-URI Teil des erweiterten QName.

Beachten Sie bitte: Da die Input-Daten der beiden Funktionen Strings sind, kann ein Node nicht direkt mit den Input-Argument-Kästen dieser Funktionen verbunden werden.

Der Node muss zuerst an die `node-name` Funktion geliefert werden, die dann den erweiterten QName ausgibt. Dieser erweiterte QNamee kann dann als Input für die beiden Funktionen verwendet werden (*siehe Abbildung unten*).



Die Ausgabe der beiden Funktionen ist ein String.

string functions (String-Funktionen)

Die folgenden String-Funktionen stehen zur Verfügung:

compare
Die `compare` Funktion erhält zwei Strings als Argumente und vergleicht diese alphabetisch und überprüft, ob diese identisch sind. Wenn *String-1* im Alphabet vor *String-2* (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 *String-1* im Alphabet nach *String-2* (z.B. `B` und `A`) vorkommt, dann gibt die Funktion +1 zurück.

Eine Variante dieser Funktion ermöglicht Ihnen, auszuwählen, welche Collation zum Vergleichen der Strings verwendet werden soll. Wenn keine Collation verwendet wird, wird die Standard-Collation, nämlich die Unicode Codepoint Collation, verwendet. Der Altova Prozessor unterstützt nur die Codepoint Collation.

ends-with
Die `ends-with` Funktion überprüft, ob *String-1* mit *String-2* endet. Wenn ja, gibt die Funktion `true` zurück, andernfalls `false`.

Eine Variante dieser Funktion ermöglicht Ihnen, auszuwählen, welche Collation zum Vergleichen der Strings verwendet werden soll. Wenn keine Collation verwendet wird, wird die Standard-Collation, nämlich die Unicode Codepoint Collation verwendet. Der Altova Prozessor unterstützt nur die Codepoint Collation.

escape-uri
Die `escape-uri` Funktion erhält eine URI als Input für das erste String-Argument und wendet die URI Escape-Konventionen von RFC 2396 auf den String an. Das zweite Boolesche Argument (`escape-reserved`) sollte auf `true()` gesetzt werden, wenn Zeichen mit einer bereits reservierten Bedeutung in URIs mit Escape ausgelassen werden sollen (z.B. "+" oder "/").

Beispiel:

`escape-uri("My A+B.doc", true())` würde `My%20A%2B.doc` zurückgeben
`escape-uri("My A+B.doc", false())` würde `My%20A+B.doc` zurückgeben

lower-case
Die `lower-case` Funktion erhält einen String als Argument und konvertiert alle Großbuchstaben im String in die entsprechenden Kleinbuchstaben.

matches
Die `matches` Funktion überprüft, ob ein bereitgestellter String (das erste Argument) einem regulären Ausdruck (dem zweiten Argument) entspricht. Die Syntax des regulären Ausdrucks muss die Syntax sein, die für die `pattern` Facet von XML-Schema definiert wurde. Die Funktion gibt `true` zurück, wenn der String dem regulären Ausdruck entspricht, und andernfalls `false`.

Die Funktion erhält ein optionales `flags` Argument. Es sind vier Flags definiert (`i`, `m`, `s`, `x`). Es können mehrere Flags verwendet werden: z.B: `imx`. Wenn kein Flag verwendet wird, werden die Standardwerte aller vier Flags verwendet.

Die vier Flags haben die folgende Bedeutung:

- `i` Modus "Groß/Kleinschreibung wird nicht berücksichtigt" verwenden. Die Standardeinstellung ist Groß-/Kleinschreibung berücksichtigen.
- `m` Mehrzeiligen Modus verwenden. In diesem Modus wird der String als mehrzeilig betrachtet, wobei jede Zeile durch ein newline-Zeichen (`x0a`) getrennt wird. Die Metazeichen `^` und `$` kennzeichnen den Beginn und das Ende der einzelnen Zeilen. Die Standardeinstellung ist der String-Modus, in dem der String mit den Metazeichen `^` und `$` 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.

normalize-unicode
Die `normalize-unicode` Funktion normalisiert den Input-String (das erste Argument) entsprechend den Regeln des definierten Normalisierungsprotokolls (zweites Argument). Unterstützt werden die Normalisierungsprotokolle NFC, NFD, NFKC und NFKD.

replace
Die `replace` Funktion erhält den im ersten Argument definierten String als Input, sucht nach den in einem regulären Ausdruck definierten Übereinstimmungen (zweites Argument) und ersetzt diese durch den String im dritten Argument.

 © 2011 Altova GmbH

Die Regeln zum Suchen von Übereinstimmungen sind im matches-Attribut oben definiert. Die Funktion erhält außerdem ein optionales `flags` Argument. Die Flags sind weiter oben in der `matches` Funktion beschrieben.

starts-with
Die `starts-with` Funktion überprüft, ob *String-1* mit *String-2*. beginnt. Falls ja, gibt die Funktion `true` zurück, andernfalls `false`.

In einer Variante dieser Funktion können Sie auswählen, welche Collation zum Vergleichen der Strings verwendet wird. Wenn keine Collation angegeben wird, wird die Standard-Collation, also die Unicode Codepoint Collation verwendet. Der Altova Prozessor unterstützt nur die Unicode Codepoint Collation.

substring-after
Die substring-after Funktion gibt den Teil von *String-1* (dem ersten Argument) zurück, der sich hinter dem Test-String *String-2* (dem zweiten Argument) befindet. Ein optionales drittes Argument definiert, welche Collation beim String-Vergleich verwendet werden soll. Wenn keine Collation angegeben wird, wird die Standard-Collation, also die Unicode Codepoint Collation verwendet. Der Altova Prozessor unterstützt nur die Unicode Codepoint Collation.

substring-before
Die substring-before Funktion gibt den Teil von *String-1* (dem ersten Argument) zurück, der sich vor dem Test-String *String-2* (dem zweiten Argument) befindet. Ein optionales drittes Argument definiert, welche Collation beim String-Vergleich verwendet werden soll. Wenn keine Collation angegeben wird, wird die Standard-Collation, also die Unicode Codepoint Collation verwendet. Der Altova Prozessor unterstützt nur die Unicode Codepoint Collation.

upper-case
Die `upper-case` Funktion erhält einen String als Argument und konvertiert alle Kleinbuchstaben im String in die entsprechenden Großbuchstaben.

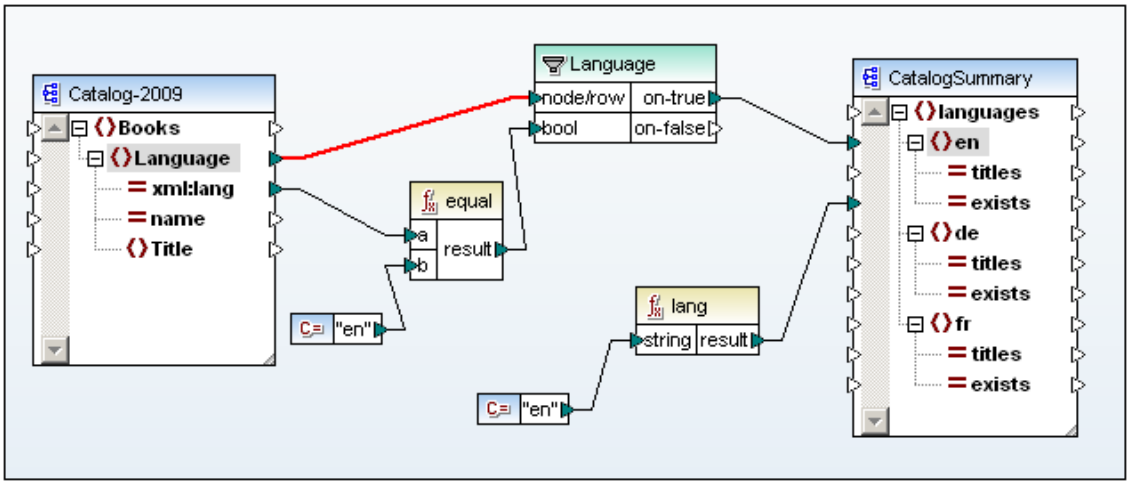
15.5.3 xslt

- xpath functions
  Die Funktionen in der XPath-Funktionsbibliothek sind XPath 1.0 Nodeset-Funktionen.
- xslt functions
  Die Funktionen in der XSLT-Funktionsbibliothek sind XSLT 1.0 Funktionen.

xpath functions (XPath-Funktionen)

Die Funktionen in der XPath-Funktionsbibliothek sind XPath 1.0 Nodeset-Funktionen. Jede dieser Funktionen erhält einen Node oder ein Nodeset als Kontext und gibt Informationen über diesen Node oder dieses Nodeset zurück. Diese Funktionen haben normalerweise:

- einen Kontext-Node (in der Abbildung unten ist der Kontext-Node für die `lang` Funktion das Element "Language" des Quellschemas).
- ein Input-Argument (in der Abbildung unten ist das Input-Argument für die `lang` Funktion die String-Konstante `en`). Die Funktionen `last` und `position` erhalten kein Argument.



lang
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. Beachten Sie in der Abbildung oben bitte Folgendes:

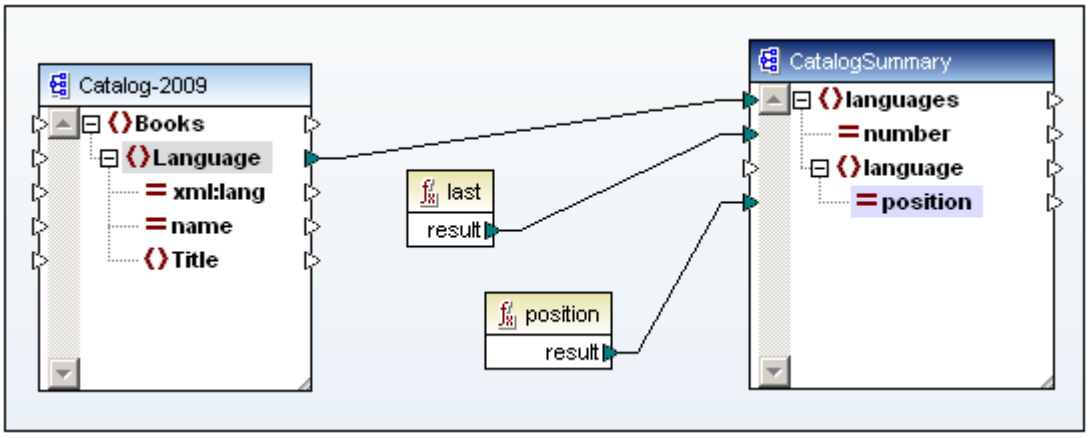
1. Das `Language` Element im Quellschema hat ein `xml:lang` Attribut.
2. `Language` Nodes werden gefiltert, sodass nur diejenigen `Language` Nodes mit dem `xml:lang` Wert `en` verarbeitet werden (der Filtertest ist in der `equal` Funktion definiert).
3. Der `Language` Node ist der Kontext-Node an der Stelle, an der im Ausgabedokument ein `en` Element erstellt wird.
4. Der Ausgabewert der `lang` Funktion (`true` oder `false`) wird an den `en/@exists` Attribut-Node der Ausgabe gesendet. Das Argument der Funktion wird durch die String-Konstante `en` bereitgestellt. Anschließend überprüft die `lang` Funktion, ob der Kontext-Node an dieser Stelle (dem `Language` Element) ein `xml:lang` Attribut mit dem Wert `en` (dem Argument der Funktion) hat. Falls ja, wird `true` zurückgegeben, andernfalls `false`.

last, position
Die Funktionen `last` und `position` erhalten kein Argument. Die `last` Funktion gibt die Position

 © 2011 Altova GmbH

des letzten Node im Kontext-Nodeset zurück. Die `position` Funktion gibt die Position des aktuellen Node im gerade verarbeiteten Nodeset zurück.

Das Kontext-Nodeset bei den Nodes, an die die Funktionen gerichtet sind, ist das Nodeset, auf das die Funktionen angewendet werden. In der Abbildung unten ist das Nodeset von `Language` Elementen das Kontext-Nodeset für die Funktionen `last` und `position`.

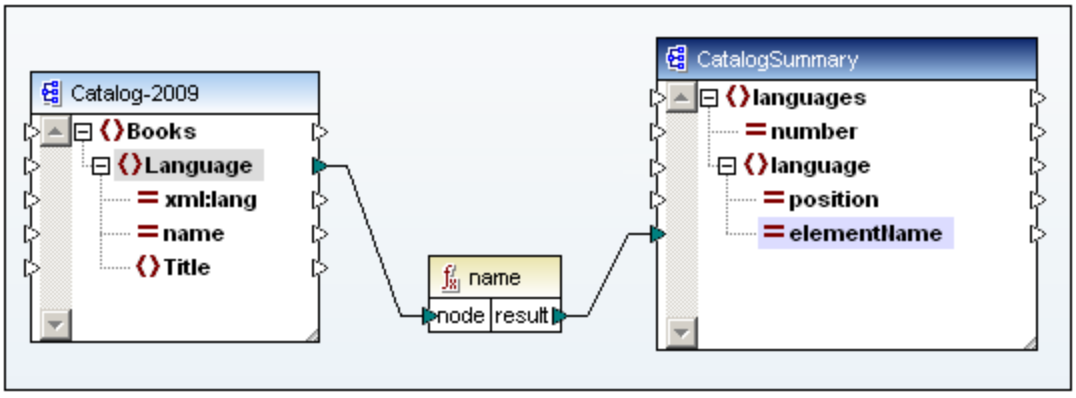


Im Beispiel oben gibt die `last` die Position des letzten Node des Kontext-Nodeset (des Nodeset der `Language` Elemente) als den Wert des `number` Attributs zurück.Dieser Wert entspricht auch der Größe des Nodeset, da er angibt, wie viele Nodes das Nodeset enthält.

Die `position` Funktion gibt die Position des gerade verarbeiteten `Language` Node zurück. Für jeden `Language` Element-Node wird die Position innerhalb des Nodeset der `Langauge` Elemente im `language/@position` Attribute-Node ausgegeben.

name, local-name, namespace-uri
Diese Funktionen werden alle auf dieselbe Art verwendet. Sie geben den Namen, den lokalen Namen bzw. die Namespace URI des Input-Node zurück. In der Abbildung unten sehen Sie, wie diese Funktionen verwendet werden. Beachten Sie, dass kein Kontext-Node definiert ist.

Die name-Funktion gibt den Namen des `Language` Node zurück und gibt diesen an das `language/@elementname` Attribut aus. Wenn das Argument einer dieser Funktionen ein Nodeset anstelle eines einzelnen Node ist, wird der Name (bzw. der lokale Name oder die Namespace URI) des ersten Node im Nodeset zurückgegeben.



Die `name` Funktion gibt den QName des Node zurück; die `local-name` Funktion gibt den lokalen Teil des Names des QName zurück. Wenn der QName eines Node z.B. `altova:MyNode` ist,

dann ist `MyNode` der lokale Name.

Die Namespace URI ist die URI des Namespace, dem der Node angehört. Das Präfix `altova:` kann z.B. so deklariert werden, dass es der folgenden Namespace URI zugeordnet wird: `xmlns:altova="http://www.altova.com/namespaces"`.

Anmerkung: Weitere XPath 1.0 Funktionen finden Sie auch in der "Core" Funktionsbibliothek.

xslt functions (XSLT-Funktionen)

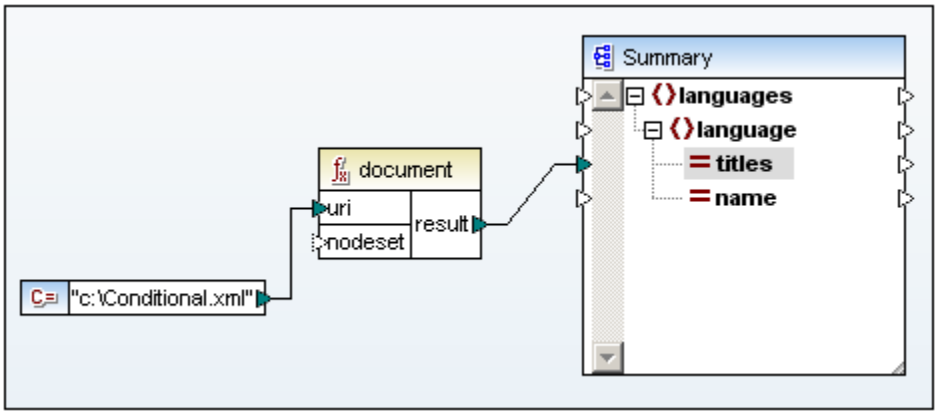
Die Funktionen in der XSLT-Funktionsbibliothek sind XSLT 1.0 Funktionen und werden weiter unten beschrieben. Um eine Funktion zu verwenden, ziehen Sie sie in das Mapping. Wenn Sie den Mauszeiger über den Input-Argumentteil eines Funktionskastens ziehen, wird der erwartete Datentyp des Arguments in einem Popup-Fenster angezeigt.

current

Die current-Funktion erhält kein Argument und gibt den aktuellen Node zurück.

document

Die document Funktion richtet sich (mit dem Argument `uri`; *siehe Abbildung unten*) an ein externes XML-Dokument. Das optionale Nodeset-Argument definiert einen Node, anhand dessen Basis-URI die URI aufgelöst wird, die als erstes Argument bereitgestellt wird, wenn diese URI relativ ist. Das Ergebnis wird in einen Node im Ausgabedokument ausgegeben.



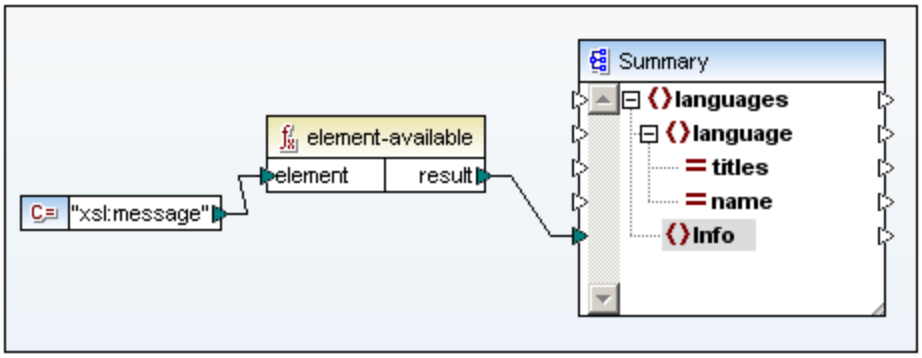
Beachten Sie, dass das `uri` Argument ein String ist, der ein absoluter Pfad sein muss.

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 zugrundeliegenden XSLT-Dokument, das für das Mapping erstellt wird, deklariert sind.

 © 2011 Altova GmbH



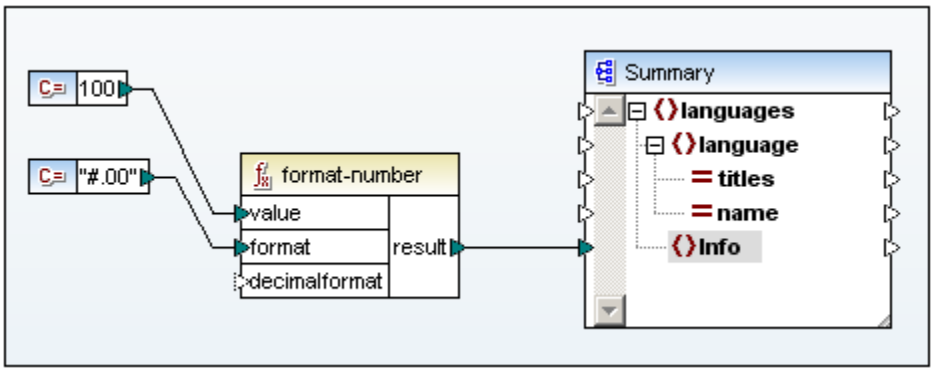
Die Funktion gibt einen Booleschen Wert zurück.

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.

format-number
Die `format-number` Funktion erhält eine Ganzzahl als erstes Argument (`value`) und einen Format-String als zweites Argument (`format`). Das dritte optionale Argument ist ein String, der angibt, welches Dezimalformat verwendet werden soll. Wenn dieses Argument nicht verwendet wird, wird das Standarddezimalformat verwendet.

Dezimalformate werden durch das XSLT 1.0 `decimal-format` Element definiert: Jedes so definierte Dezimalformat kann benannt werden und dieser Name kann als drittes Argument der `format-number` Funktion verwendet werden. Wenn ein Dezimalformat ohne Namen definiert wird, ist es das Standarddezimalformat für die Transformation.



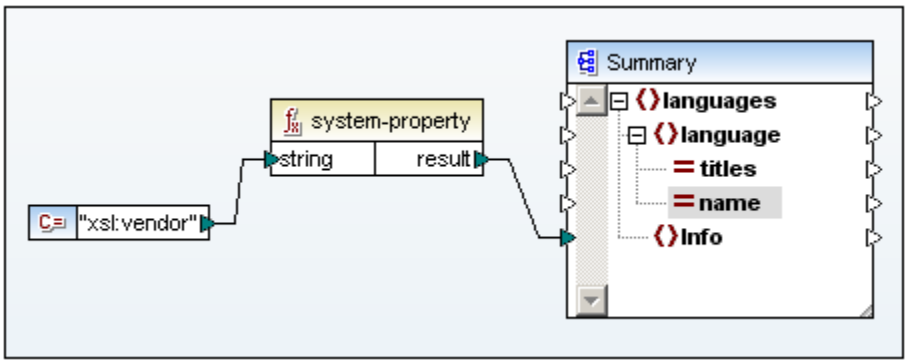
Die Funktion gibt die als String formatierte Zahl zurück.

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.

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 `xsl:prefix`, haben, da dies das Präfix ist, das im zugrundeliegenden XSLT-Stylesheet mit dem XSLT-Namespace verknüpft ist.



unparsed-entity-uri
Wenn Sie eine DTD verwenden, können Sie dafür 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 gerichtet werden, z.B. an einen `href` Node.

© 2011 Altova GmbH

Kapitel 16

Unterstützung für QName

16 Unterstützung für QName

Mit Hilfe von QNames (qualifizierten Namen) können Sie in XML- und XBRL-Instanzdokumenten verwendete Namespace URIs referenzieren und abkürzen. Es gibt zwei Arten von QNames; Namen mit und ohne Präfix.

PrefixedName	Präfix	**:**	LocalPart
UnPrefixedName			LocalPart
			wobei LocalPart ein Element- oder Attributname ist.

<Doc xmlns:**x**="http://myCompany.com">

<x:part>

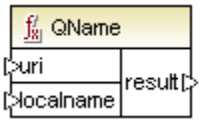
</Doc>

x ist die Namespace-Referenz auf "**http://myCompany.com**" und <x:part> ist daher ein gültiger QName, da:

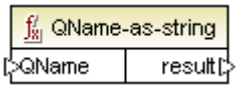
x das Namespace-Präfix ist und
part der LocalPart ist, d.h. der Elementname.

MapForce unterstützt die folgenden QName-Funktionen im Abschnitt **Lang** des Bibliotheksfensters:

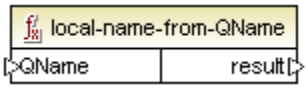
QName
Konstruiert anhand einer Namespace URI und eines lokalen Teils einen QName. Mit Hilfe dieser Funktion können Sie einen QName in einer Zielkomponente erstellen. Der uri- und localname-Parameter können durch eine Konstantenfunktion bereitgestellt werden.



QName-as-string
Konvertiert einen QName in einen String in der Form **{http://myCompany.com}local**.

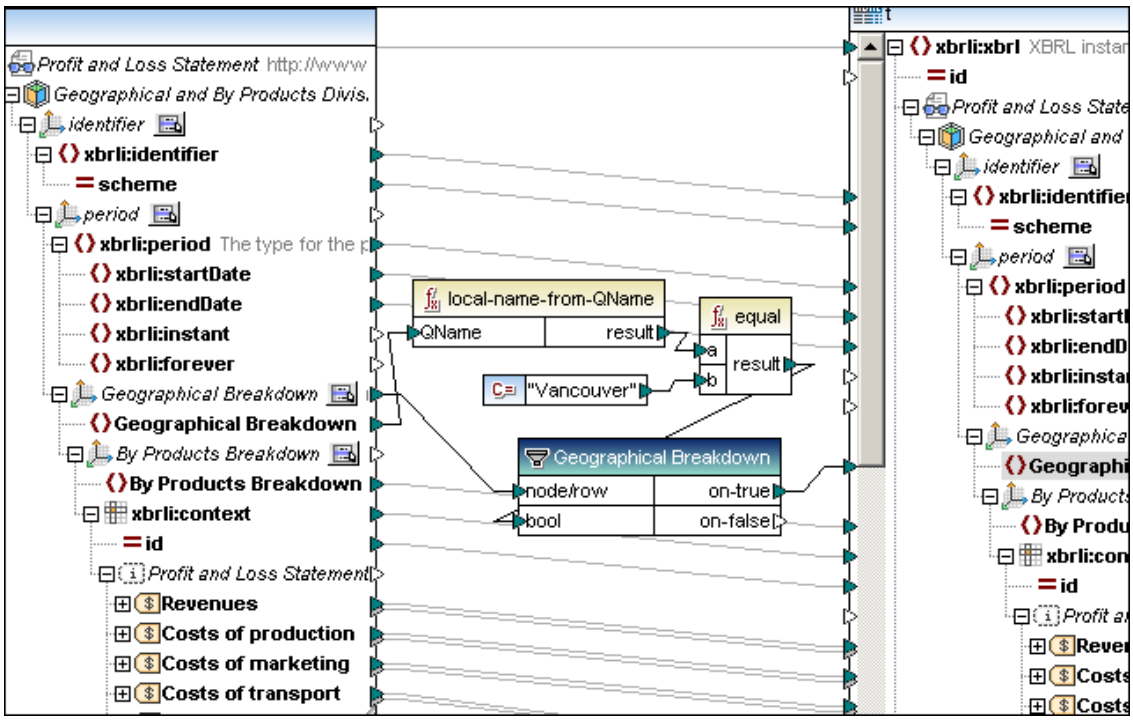


local-name-from-QName
Extrahiert aus einem QName den lokalen Namen.



Diese Funktion ist beim Mappen von XBRL-Instanzdokumenten, die Hypercubes enthalten extrem nützlich.

 © 2011 Altova GmbH



Beim Mapping werden jene Facts, bei denen der **lokale Name** des **Inhalts** des explicit Member (d-g:Vancouver) gleich "Vancouver" ist, herausgefiltert. Beachten Sie, dass der Inhalt des Member selbst ein QName ist.

```
<xbrli:context id="Year2007_Vancouver_BeveragesTotal">
  <xbrli:entity>
    <xbrli:identifier scheme="http://www.stockexchange/ticker">ACME</xbrli:identifier>
    <xbrli:segment>
      <xbrldi:explicitMember dimension="d-g:GeographicalBreakdown">d-g:Vancouver</xbrl
      <xbrldi:explicitMember dimension="d-b:ByProductsBreakdown">d-b:BeveragesTotal</x
    </xbrli:segment>
```

Alle Facts, die zur Dimension "GeographicalBreakdown" gehören, werden gefiltert und an die Zielkomponente übergeben.

```
<context id="Year2007_Vancouver_BeveragesTotal">
  <entity>
    <identifier scheme="http://www.stockexchange/ticker">ACME</identifier>
    <segment>
      <explicitMember dimension="d-b:ByProductsBreakdown" xmlns="http://xbrl.org/2006/xbrldi">d-b:Beverages
      <explicitMember dimension="d-g:GeographicalBreakdown" xmlns="http://xbrl.org/2006/xbrldi" />
    </segment>
  </entity>
  <period>
    <startDate>2007-01-01</startDate>
    <endDate>2007-12-31</endDate>
  </period>
</context>
<p:Revenues contextRef="Year2007_Vancouver_BeveragesTotal" unitRef="USD" decimals="0">4</p:Revenues>
<p:CostsOfProduction contextRef="Year2007_Vancouver_BeveragesTotal" unitRef="USD" decimals="0">1</p:Cos
<p:CostsOfMarketing contextRef="Year2007_Vancouver_BeveragesTotal" unitRef="USD" decimals="0">1</p:Cost
<p:ProfitLoss contextRef="Year2007_Vancouver_BeveragesTotal" unitRef="USD" decimals="0">2</p:ProfitLoss>
```

namespace-uri-from-QName
Extrahiert aus einem QName die Namespace URI.

© 2011 Altova GmbH

Kapitel 17

Benutzerreferenz

17 Benutzerreferenz

Der folgende Abschnitt enthält eine Liste aller Menüs und Menüoptionen von MapForce sowie eine kurze Beschreibung davon.

© 2011 Altova GmbH

17.1 Datei

Neu
Erstellen ein neues Mapping-Dokument

Öffnen
Öffnet zuvor gespeicherte Mapping (*.mfd) Dateien.

Bitte beachten Sie:

Ein Mapping, das Funktionen enthält, die nur in einer höheren MapForce Edition verfügbar sind, kann nicht geöffnet werden.

So kann z.B. ein Mapping, das Webservice-Funktionen enthält in der Professional Edition nicht geöffnet werden. Datenbankmappings sind z.B. in Basic Editions nicht möglich.

Speichern
Speichert das aktuelle Mapping unter dem gerade aktiven Dateinamen.

Speichern unter
Speichert das aktuelle Mapping unter einem anderen Namen oder ermöglicht Ihnen, dem Projekt einen neuen Namen zu geben, falls Sie es zum ersten Mal speichern.

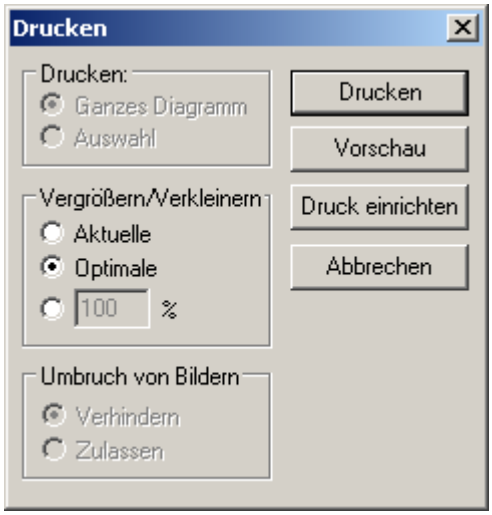
Alles speichern
Speichert alle derzeit geöffneten Mapping-Dateien.

Neu laden
Lädt die aktuell aktive Mapping-Datei neu. Sie werden gefragt, ob Ihre letzten Änderungen verworfen werden sollen.

Schließen
Schließt die aktuell aktive Mapping-Datei. Sie werden gefragt, ob Sie die Datei speichern möchten, bevor sie sie schließen.

Alles schließen
Schließt alle derzeit geöffneten Mapping-Dateien. Sie werden gefragt, ob Sie eine der nicht gespeicherten Mapping-Dateien speichern möchten.

Drucken
Öffnet das Dialogfeld Drucken, über das Sie Ihr Mapping ausdrucken können.



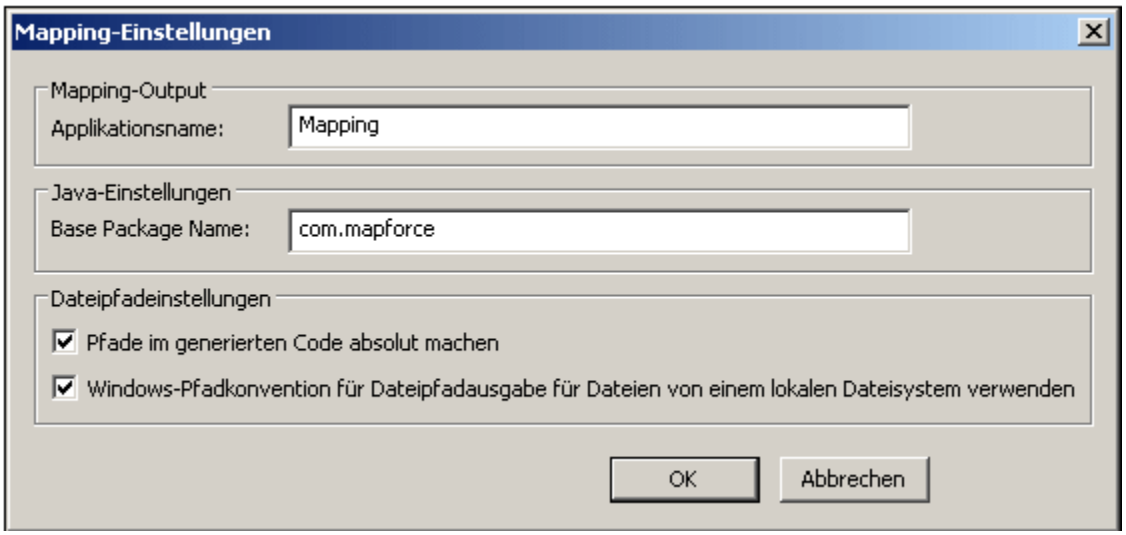
Mit "Aktuelle" wird der aktuell definierte Zoom-Faktor des Mappings beibehalten. Bei Auswahl von "Optimale" wird das Mapping auf Seitengröße vergrößert/verkleinert. Sie können auch einen numerischen Zoom-Faktor angeben. Die Bildlaufleisten der Komponente werden nicht gedruckt. Außerdem können Sie festlegen, ob die Grafik auf mehrere Seiten umbrochen werden soll oder nicht.

Druckvorschau
Öffnet dasselbe Dialogfeld "Drucken" mit denselben Einstellungen wie oben beschrieben.

Druckereinrichtung
Öffnet das Dialogfeld "Druckereinrichtung", in dem Sie den gewünschten Drucker und die Seitenausrichtung usw. einstellen können.

Mapping validieren
Beim Validieren eines Mappings wird überprüft, ob alle Mappings (Konnektoren) gültig sind und entsprechende Warn- oder Fehlermeldungen werden ggf. angezeigt.
Nähere Informationen dazu finden Sie unter "Validieren von Mappings.

Mapping-Einstellungen



Hier sind die dokumentspezifischen Einstellungen definiert. Sie werden in der *.mfd Datei gespeichert.

Mapping Output
Applikationsname: definiert das Präfix für den XSLT1.0/2.0-Dateinamen für die generierten Transformationsdateien.

Dateipfadeinstellungen

Pfade im generierten Code absolut machen
Damit wird sichergestellt, dass der generierte Code mit Mapping-Dateien (*.mfd) aus Versionen vor Version 210 kompatibel ist. Nähere Informationen dazu finden Sie unter Relative und absolute Dateipfade.

Windows-Pfadkonvention für Dateipfadausgabe ... verwenden
Mit Hilfe dieses Kontrollkästchens stellen Sie sicher, dass die Windows-Pfadkonventionen erfüllt werden. Bei der Ausgabe von XSLT2 (und XQuery) wird der gerade verarbeitete Dateiname intern mit der Funktion document-uri abgerufen. Diese Funktion gibt einen Pfad in der Form Datei:// URI für lokale Dateien zurück.

 © 2011 Altova GmbH

Wenn dieses Kontrollkästchen aktiviert ist, wird eine Datei:// URI Pfadspezifikation zur einfacheren Weiterverarbeitung automatisch in einen vollständigen Windows-Dateipfad (z.B. "C:\...") konvertiert.

Code in ausgewählter Sprache generieren
Generiert Code in der aktuell ausgewählten Sprache Ihres Mappings. Die aktuell ausgewählte Sprache wird durch ein markiertes Symbol in der Titelleiste angezeigt: XSLT, XSLT2 .

Code generieren in | XSLT/XSLT2
Mit diesem Befehl wird/werden die für die Transformation aus der/den Quelldatei(en) benötigte(n) 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 können.

Hinweis: Der Name der generierten XSLT-Datei(en) ist im Feld **Applikationsname** des Dialogfelds "Mapping-Output" definiert. Dieses Dialogfeld lässt sich durch Auswahl von **Datei | Mapping-Einstellungen** öffnen.

Dokument generieren
Generiert detaillierte Dokumentation zu Ihren Mapping-Projekten in verschiedenen Ausgabeformaten. Nähere Informationen dazu finden Sie unter Dokumentieren von Mapping-Projekten.

Letzte Dateien
Zeigt eine Liste der zuletzt geöffneten Dateien an.

Letzte Projekte
Zeigt eine Liste der zuletzt geöffneten Projekte an.

Beenden
Beendet die Applikation. Sie werden gefragt, ob Sie nicht gespeicherte Dateien speichern möchten.

17.2 Bearbeiten

Die meisten der Befehle in diesem Menü werden erst aktiv, wenn Sie das Ergebnis eines Mappings im **Ausgabe**-Fenster bzw. z.B. den XSLT-Code 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.

Wiederherstellen
Mit dem Befehl "Wiederherstellen" können Sie zuvor rückgängig gemachte Befehle wiederherstellen. Auf diese Weise können Sie mit Hilfe dieser beiden Befehle den Verlauf Ihres Mappings Schritt für Schritt vor und zurück durchgehen.

Suchen
Dient zum Suchen von bestimmtem Text auf der Registerkarte XSLT, XLST2 oder Ausgabe.

Weitersuchen F3
Sucht nach der nächsten Stelle, an der der Such-String vorkommt.

Ausschneiden/Kopieren/Einfügen/Löschen
Mit den Windows-Standardbearbeitungsbefehlen können Sie beliebige im Mapping-Fenster angezeigte Komponenten oder Funktionen ausschneiden, kopieren usw.

Alles markieren
Markiert alle Komponenten auf dem Register "Mapping" oder den Text/Code auf den Registern XSLT, XSLT2 bzw. Ausgabe.

© 2011 Altova GmbH

17.3 Einfügen

XML Schema / Datei

Fügt eine XML-Schemadatei als Datenquelle oder Zielkomponente in das Mapping-Fenster ein. Sie können XML-Dateien mit einer Schemareferenz auswählen. In diesem Fall wird das referenzierte Schema automatisch eingefügt. Wenn Sie eine XML-Schema-Datei einfügen, werden Sie gefragt, ob Sie eine XML-Instanzdatei bereitstellen möchten, die die Daten für die XSLT-, XSLT2 und Ausgabe enthält. Wenn Sie eine XML-Datei ohne eine Schemareferenz auswählen, werden Sie gefragt, ob automatisch ein XML-Schema dafür generiert werden soll.

17.4 Komponente

Root-Element ändern
Dient zum Ändern des Root-Elements des XML-Instanzdokuments.

Schema-Definition in XMLSpy bearbeiten
Wenn Sie nach Auswahl eines XML-Schemas/ Dokuments auf diese Option klicken, wird eine XML-Schemadatei in der Schema-Ansicht von XMLSpy geöffnet, wo Sie sie bearbeiten können.

Duplikat erzeugen
Fügt eine Kopie/einen Klon des ausgewählten Elements ein und ermöglicht Ihnen dadurch, mehrere Eingabedaten auf dieses Element zu mappen. Duplizierte Datenelemente haben kein Output-Symbol und können nicht als Quellelemente verwendet werden. Ein Beispiel dazu finden Sie im Tutorial im Abschnitt "Duplizieren von Datenelementen".

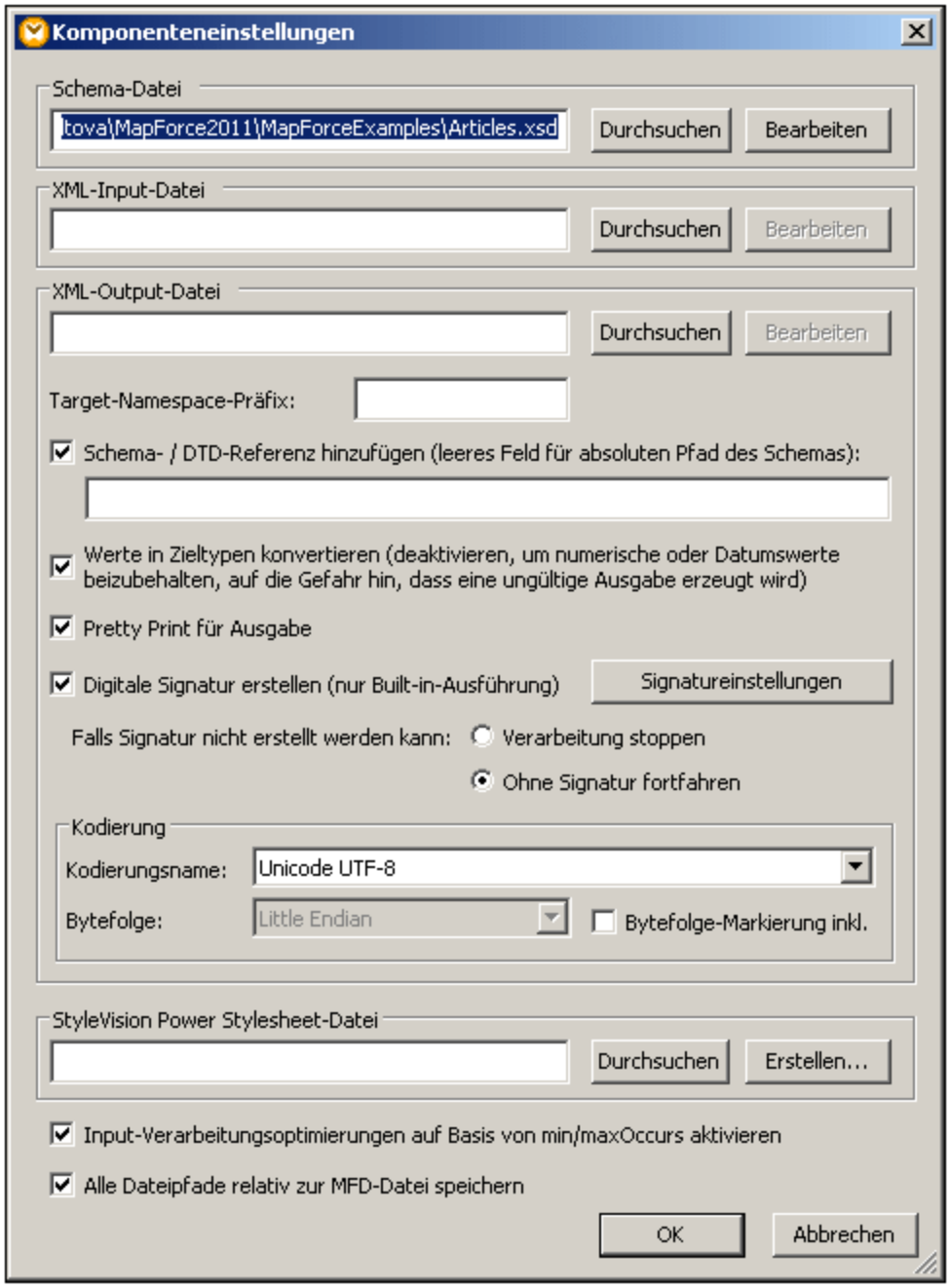
Duplikat löschen
Löscht ein zuvor definiertes Duplikat. Nähere Informationen dazu finden Sie im Tutorial im Abschnitt "Duplizieren von Datenelementen".

Links ausrichten
Richtet die Datenelemente entlang des linken Fensterrands aus.

Rechts ausrichten
Richtet die Datenelemente entlang des rechten Fensterrands aus. Diese Anzeige eignet sich vor allem, um Mappings auf das Zielschema zu erzeugen.

Eigenschaften
Öffnet ein Dialogfeld, in dem die Einstellungen der aktuell ausgewählten Komponente angezeigt werden. Wenn es sich bei der Komponente um eine XML-Schema-Datei handelt, wird das Dialogfeld "Komponenteneinstellungen" geöffnet. Bei einer Textdatei wird das Dialogfeld "Text importieren / exportieren" geöffnet.

© 2011 Altova GmbH



Schema-Datei: Zeigt den Dateinamen und Pfad des Zielschemas an.

XML-Input-Datei: Dient zum Auswählen oder Ändern der XML-Instanz für die aktive Schemakomponente. Dieses Feld wird ausgefüllt, wenn Sie die Schemakomponente zum ersten Mal einfügen und ihr eine XML-Instanzdatei zuweisen.

XML-Output-Datei: Dies ist der Datei- und Pfadname, unter dem die XML-Zielinstanz gespeichert wird, wenn Programmcode generiert und ausgeführt wird. Der Dateiname wird auch als erster Eintrag in der Komponente angezeigt.

Der Eintrag aus dem Feld XML-Input-Instanz wird automatisch in dieses Feld übernommen, wenn Sie die XML-Instanzdatei zuweisen. Wenn Sie der Komponente

keine XML-Instanzdatei zuweisen, so enthält dieses Feld den Eintrag **schemafilenameandpath.xml.**

Target-Namespace-Präfix: Ermöglicht die Eingabe eines Präfixes für den Target-Namespace, wenn es sich um ein Schema / XML-Dokument handelt. Ein Target-Namespace muss im Zielschema definiert werden, damit man ihm hier das Präfix zuweisen kann.

Schema- / DTD-Referenz hinzufügen: Fügt den Pfad der referenzierten Schema-Datei zum Root-Element der XML-Ausgabe hinzu.

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 sowohl eine http:// Adresse als auch einen absoluten oder relativen Pfad eingeben.

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.

Zielwerte in Zieltypen konvertieren: Damit können Sie festlegen, ob beim Mapping die XML-Zielschematypen verwendet werden sollen (Standardeinstellung - aktiviert), oder ob alle Daten, die auf die Zielkomponenten gemappt werden, als **Stringwerte** behandelt werden sollen.

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.

Beachten Sie: 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-Dokument im Ausgabefenster neu, um eine strukturierte Anzeige des Dokuments zu generieren. (Jedes Subelement ist um einen Tabstopp vom übergeordneten Element eingerückt.)

Kodierungseinstellungen

Ab MapForce Version 2008 hat jede Komponente, sowohl Quellen als auch Ziele, ihre eigenen Kodierungseinstellungen. Das bedeutet, dass die *.mfd-Mapping-Dateien keine Standardkodierung haben. Jede Komponente, aus der sich das Mapping zusammensetzt hat ihre eigenen Kodierungseinstellungen. Bei Komponenten handelt es sich immer um XML-, und Text-Komponenten.

Es gibt jedoch eine Standardkodierungseinstellung, die unter **Extras | Optionen** auf dem

 © 2011 Altova GmbH

Register "Allgemein" definiert ist. Sie heißt "Standardkodierung für neue Komponenten" und wird angewendet, wenn neue Komponenten erstellt/eingefügt werden. Wenn Mappings aus früheren Versionen geöffnet werden, wird die Standardkodierungseinstellung verwendet.

Die Gruppe Kodierung besteht aus drei Steuerelementen:

- Auswahlliste zur Auswahl des Kodierungsnamens.
- Auswahlliste zur Definition der Byte-Folge (little endian, big endian).
- Kontrollkästchen "Bytefolge-Markierung inkludieren".

Die Standardeinstellungen sind:

- UTF-8
- little endian (deaktiviert bei UTF-8)
- keine Bytefolge-Markierung.

Bitte beachten Sie:

Die Aktivierung des Kontrollkästchens "Bytefolge-Markierung inkludieren" im Dialogfeld "Komponenteneinstellungen" hat keinerlei Auswirkung auf die Ausgabe von **XSLT 1.0/2.0,** da diese Sprachen BOMs (Byte Order Marks) nicht unterstützen.

Input-Verarbeitungsoptimierungen auf Basis von min/maxOccurs aktivieren
In MapForce Version 2009 gibt es nun eine spezielle Behandlung für 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, kann diese Optimierung in den Komponenteneinstellungen von XML- und EDI-Komponenten deaktiviert werden.

17.5 Verbindung

Idente Sub-Einträge automatisch verbinden
Aktiviert bzw. deaktiviert die Funktion "Idente Sub-Einträge automatisch verbinden" sowie das dazugehörige Symbol in der Symbolleiste.

Einstellungen für "Idente Sub-Einträge verbinden"
Öffnet das Dialogfeld "Einstellungen für 'Idente Sub-Einträge verbinden'", in dem Sie die Verbindungseinstellungen definieren können.

Idente Sub-Einträge verbinden
Mit diesem Befehl können Sie sowohl im Quell- als auch im Zielschema mehrere Konnektoren für Datenelemente **desselben** Namens erzeugen. Die Einstellungen, die Sie in diesem Dialogfeld definieren, werden beibehalten und beim Verbinden zweier Elemente angewendet, wenn in der Titelleiste das Symbol "**Idente Sub-Einträge automatisch verbinden**" aktiv ist. Durch Klicken auf das Symbol können Sie es aktivieren bzw. deaktivieren. Nähere Informationen dazu finden Sie im Abschnitt "Konnektoreigenschaften".

Zielorientiert (Standard)
Ändert den Konnektortyp in Standard-Mapping. Nähere Informationen siehe Quellorientiertes/Mixed Content und Standard-Mapping.

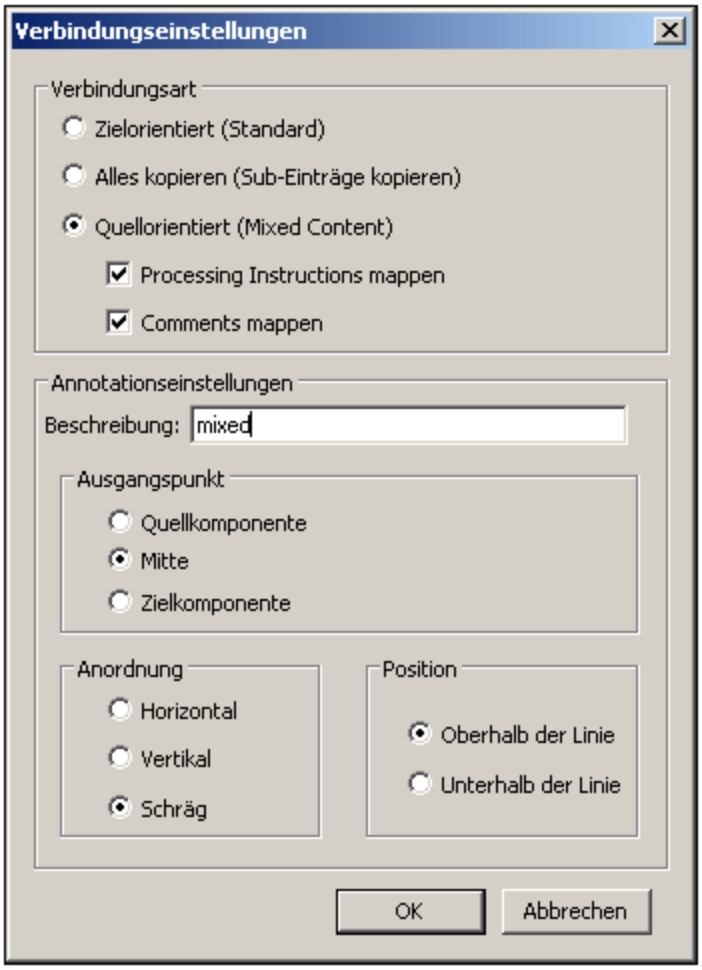
Alles kopieren (Sub-Einträge kopieren)
Erstellt Konnektoren für alle übereinstimmenden Sub-Einträge, wobei jeder der Child-Konnektoren als Substruktur des Parent-Konnektors dargestellt wird. Nähere Informationen siehe ""Alles kopieren"-Verbindungen"

Quellorientiert (Mixed Content)
Ändert den Konnektortyp in quellorientiert / mixed content und gestattet die Auswahl zusätzlicher Elemente für das Mapping. Die zusätzlichen Elemente müssen **Child-Elemente** des gemappten Datenelements in der XML-Quelldatei sein, damit sie gemappt werden können. Nähere Informationen siehe Standardeinstellungen: Mixed Content.

Eigenschaften:
Öffnet ein Dialogfeld, in dem Sie die jeweiligen (mixed content) Einstellungen für den aktuellen Konnektor definieren können. Beachten Sie, dass Optionen, die nicht zur Verfügung stehen, ausgegraut sind.

Bitte beachten Sie, dass diese Einstellungen auch für **complexType-**Elemente gelten, die keine Text-Nodes enthalten!

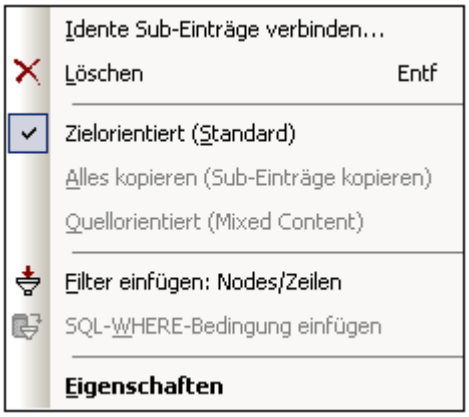
© 2011 Altova GmbH



Annotationseinstellungen:
Einzelne Konnektoren können zur besseren Unterscheidung beschriftet werden.

1. Doppelklicken Sie auf einen Konnektor und geben Sie den Namen des Konnektors in das Feld "Beschreibung" ein. Dadurch werden alle Optionen im Bereich "Annotationseinstellungen" aktiv.
2. In den restlichen Bereichen können Sie die Position und Ausrichtung der Beschriftung definieren.

Konnektor-Kontextmenü:



Idente Sub-Einträge verbinden
Öffnet das Dialogfeld "Idente Sub-Einträge verbinden", in dem Sie die Verbindungseinstellungen ändern und die Datenelemente verbinden können, wenn Sie auf OK klicken.

Löschen
Löscht den markierten Konnektor.

Zielorientiert (Standard)
Ändert den Konnektortyp in Standard-Mapping, Nähere Informationen dazu siehe: "Quellorientiertes/Mixed Content und Standard-Mapping".

Alles kopieren (Sub-Einträge kopieren)
Ändert den Konnektortyp in "Alles kopieren" und verbindet alle Sub-Einträge desselben Namens auf eine grafisch optimierte Weise. Nähere Informationen dazu siehe ""Alles kopieren" Konnektoren".

Quellorientiert (Mixed Content)
Ändert den Konnektortyp in "Quellorientiert / Mixed Content". Nähere Informationen dazu siehe: "Quellorientiertes und Mixed Content Mapping".

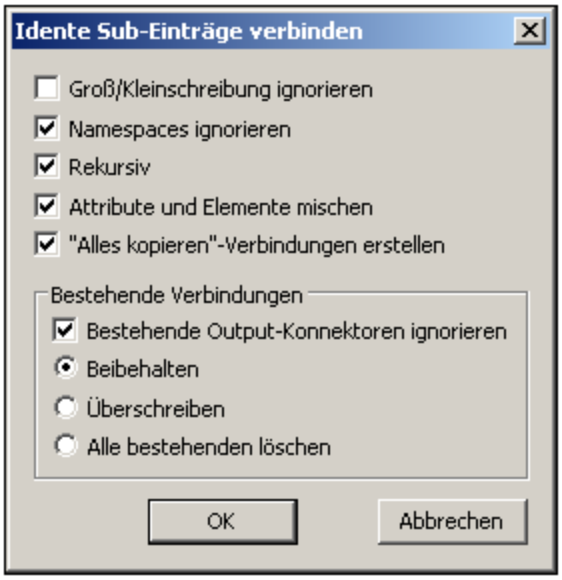
Filter einfügen: Nodes/Zeilen
Fügt eine Filerkomponente in den Konnektor ein. Der Quellkonnektor wird mit dem Parameter "nodes/row" verbunden, der Zielkonnektor mit dem "on-true" Parameter. Nähere Informationen dazu finden Sie unter Filtern - Abrufen dynamischer Daten, Lookup-Tabelle.

Eigenschaften
Öffnet das Dialogfeld "Verbindungseinstellungen", in dem Sie die jeweiligen mixed content-Einstellungen sowie die Einstellungen für Konnektor-Annotationen vornehmen können. Nähere Informationen dazu finden Sie im Referenzteil im Abschnitt Verbindung.

Dialogfeld "Idente Sub-Einträge verbinden"
Mit diesem Befehl können Sie mehrere Konnektoren zwischen Datenelementen erstellen, die in der Quell- und in der Zielkomponente **denselben Namen** haben.

1. Verbinden Sie zwei übergeordnete Datenelemente, die in beiden Komponenten **Child-Elemente** desselben Namens aufweisen.
2. Rechtsklicken Sie auf den Konnektor und wählen Sie die Option **Idente Sub-Einträge verbinden**.

© 2011 Altova GmbH



3. Wählen Sie die weiter unten beschriebenen erforderlichen Optionen aus und klicken Sie auf OK, um das Mapping durchzuführen.

   Für alle Child-Elemente, die denselben Namen haben und den im Dialogfeld definierten Einstellungen entsprechen, werden Mappings erzeugt.

Bitte beachten Sie:
Die hier definierten Einstellungen werden beibehalten und angewendet, wenn in der Titelleiste das Symbol "**Idente Sub-Einträge automatisch verbinden**" aktiv ist. Durch Klicken auf das Symbol können Sie es aktivieren bzw. deaktivieren.

Groß/Kleinschreibung ignorieren:
Ignoriert die Groß- und Kleinschreibung der Child-Elemente.

Namespaces ignorieren:
Ignoriert die Namespaces der Child-Elemente.

Rekursiv:
Nachdem das erste Konnektoren-Set erzeugt wurde, werden auch die Elemente der nächsten untergeordneten Ebene auf identische Namen hin überprüft und ggf. auch für diese Konnektoren erstellt. Nun werden die Child-Elemente dieser untergeordneten Elemente überprüft usw.

Attribute und Elemente mischen:
Damit können Konnektoren zwischen Datenelementen desselben Namens erzeugt werden, selbst wenn es sich um verschiedene Typen handelt, z.B. wenn zwei Datenelemente "Name" existieren, eines jedoch ein Element und das andere ein Attribut ist. Ist die Option aktiviert, wird ein Konnektor zwischen diesen Datenelementen erzeugt.

Bestehende Verbindungen:

Bestehende Ausgabe-Verbindung ignorieren:
Erzeugt **zusätzliche** Konnektoren zu anderen Komponenten, selbst wenn die derzeit vorhandenen Output-Symbole bereits Konnektoren haben.

Beibehalten
Behält bestehende Konnektoren bei.

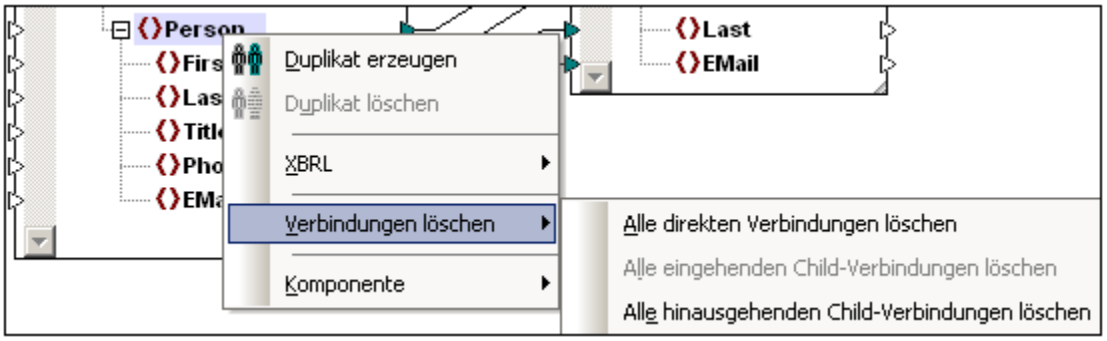
Überschreiben:
Erzeugt Konnektoren gemäß den definierten Einstellungen neu. Bestehende Konnektoren

werden gelöscht.

Alle bestehenden löschen:
Löscht alle bestehenden Konnektoren, bevor neue erzeugt werden.

Löschen von Verbindungen

Konnektoren, die über das Dialogfeld "Idente Sub-Einträge verbinden" oder während des Mappings erstellt wurden, können als Gruppe entfernt werden.



Rechtsklicken Sie auf den Datenelementnamen der Komponente, nicht den Konnektor selbst - in diesem Beispiel auf "Person".
Wählen Sie den Befehl **Verbindungen löschen | Alle... Verbindungen löschen**.

Alle direkten Verbindungen löschen:
Löscht alle Konnektoren, die direkt auf eine andere Quelle- oder Zielkomponente gemappt sind, bzw. auf die eine andere Zielkomponente gemappt ist.

Alle eingehenden Child-Verbindungen löschen:
Ist nur aktiv, wenn Sie mit der rechten Maustaste auf ein Datenelement in einer Zielkomponente geklickt haben. Löscht alle eingehenden Child-Konnektoren.

Alle hinausgehenden Child-Verbindungen löschen:
Ist nur aktiv, wenn Sie mit der rechten Maustaste auf ein Datenelement in einer Quellkomponente geklickt haben. Löscht alle hinausgehenden Child-Konnektoren.

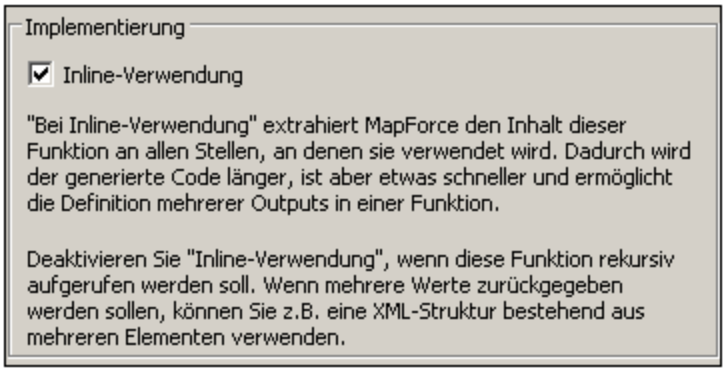
 © 2011 Altova GmbH

17.6 Funktion

Benutzerdefinierte Funktion erstellen...
Erstellt eine neue benutzerdefinierte Funktion. Bei Auswahl dieser Option wird eine leere benutzerdefinierte Funktion erstellt, in die Sie die benötigten Komponenten einfügen können. Bei Definition einer solchen Funktion wird automatisch eine einzige Ausgabekomponente eingefügt. Eine benutzerdefinierte Funktion kann nur eine Output-Komponente haben, es sei denn, sie ist als inline-Funktion definiert. Nähere Informationen dazu finden Sie unter "So erstellen Sie eine benutzerdefinierte Funktion von Grund auf neu".

Benutzerdefinierte Funktion von Auswahl erstellen:
Erstellt auf Basis der aktuell im Projektfenster ausgewählten Elemente eine neue benutzerdefinierte Funktion. Nähere Informationen dazu finden Sie unter "Hinzufügen benutzerdefinierte Funktionen".

Funktion - Einstellungen
Öffnet das Einstellungsdialogfeld für die derzeit aktive benutzerdefinierte Funktion, wo Sie die aktuellen Einstellungen ändern können. Auf diese Art können Sie den Typ der benutzerdefinierten Funktion ändern, d.h. doppelklicken Sie auf die Titelleiste einer benutzerdefinierten Funktion, um den Inhalt zu sehen und wählen Sie anschließend diese Menüoption, um den Typ zu ändern.

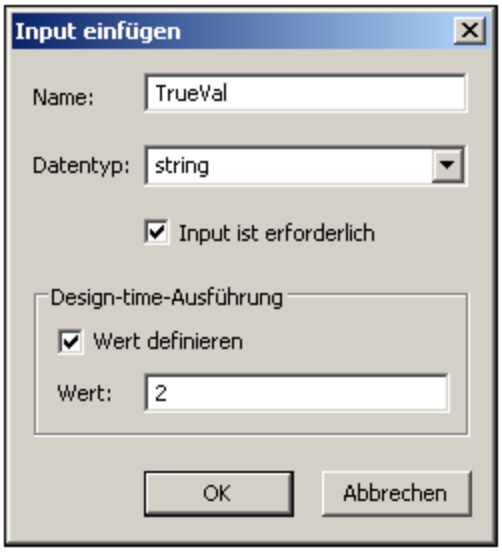


Funktion entfernen
Löscht die aktuell aktive benutzerdefinierte Funktion vom Register dieses Namens, während Sie an bestehenden benutzerdefinierten Funktionen arbeiten. D.h. dies funktioniert nur bei bestehenden benutzerdefinierten Funktionen, wenn Sie deren Inhalt anzeigen.

Es erscheint eine Meldung, die Sie daran erinnert, dass Instanzen ungültig werden können. Sie listet die Bibliotheken auf, in denen die benutzerdefinierte Funktion vorkommt.

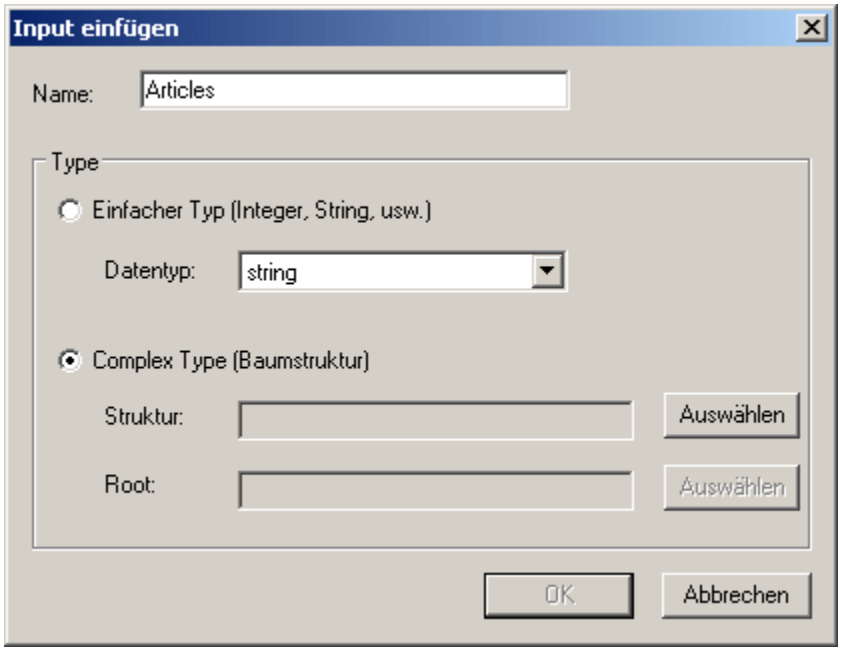
Input-Komponente einfügen
Fügt eine "Input"-Komponente in das Mapping oder in eine benutzerdefinierte Funktion ein.

Wenn Sie sich auf der Registerkarte für das Haupt-Mapping befinden, wird das unten abgebildete Dialogfeld angezeigt. Bei dieser Art von Input-Komponente können Sie bei der Ausführung des kompilierten Mappings über die Befehlszeile einen **Parameter** definieren. Nähere Informationen dazu finden Sie unter "Befehlszeile - Definieren von Input/Output-Dateien ".



Wenn Sie sich auf dem Register für die benutzerdefinierte Funktion befinden, wird das unten gezeigte Dialogfeld angezeigt. Dieser Typ der Input-Komponente gestattet Ihnen:

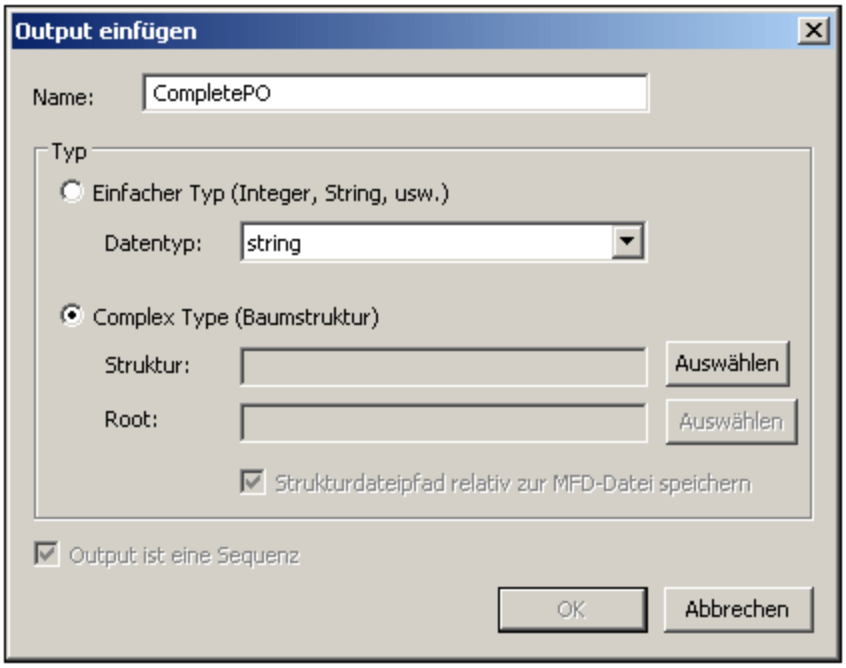
- einfache Inputs zu definieren
- komplexe Inputs zu definieren z.B. Schema-Strukturen



Output-Komponente einfügen
Fügt eine "Output"-Komponente in eine benutzerdefinierte Funktion ein. Auf einem benutzerdefinierten Funktionsregister wird das unten gezeigte Dialogfeld angezeigt. Mit diesem Typ der Input-Komponente können Sie Folgendes definieren:

- einfache Outputs
- komplexe Outputs z.B. Schema-Strukturen

 © 2011 Altova GmbH


Output einfügen
Name:
CompletePO
Typ
Einfacher Typ (Integer, String, usw.)
Datentyp:
string
Complex Type (Baumstruktur)
Struktur:
Auswählen
Root:
Auswählen
Strukturdateipfad relativ zur MFD-Datei speichern
Output ist eine Sequenz
OK
Abbrechen

17.7 Ausgabe

Mit der ersten Gruppe von Optionen (**XSLT 1.0, XSLT 2.0, usw.)** können Sie die Zielsprache definieren, in der Ihr Code erzeugt werden soll.

Beachten Sie, dass der Built-In-Ausführungsprozessor eine Vorschau auf das Ergebnis des Mappings bietet, wenn Sie auf die Schaltfläche "Ausgabe" klicken, und nicht zum Generieren von Programmcode verwendet werden kann.

Ausgabedatei validieren
Validiert die erzeugte XML-Datei anhand des referenzierten Schemas.

Generierte Ausgabedatei speichern
Speichert die aktuell auf dem Register "Ausgabe" angezeigten Daten.

Alle Ausgabedateien speichern
Speichert alle von dynamischen Mappings generierten Ausgabedateien. Nähere Informationen dazu finden Sie unter Dynamische Dateinamen - Input / Output.

Ausgabedatei neu generieren
Generiert das aktuelle Mapping vom Ausgabefenster aus neu.

Lesezeichen einfügen/löschen
Fügt im Fenster Ausgabe an der Cursorposition ein Lesezeichen ein.

Nächstes Lesezeichen
Navigiert im Fenster Ausgabe zum nächsten Lesezeichen.

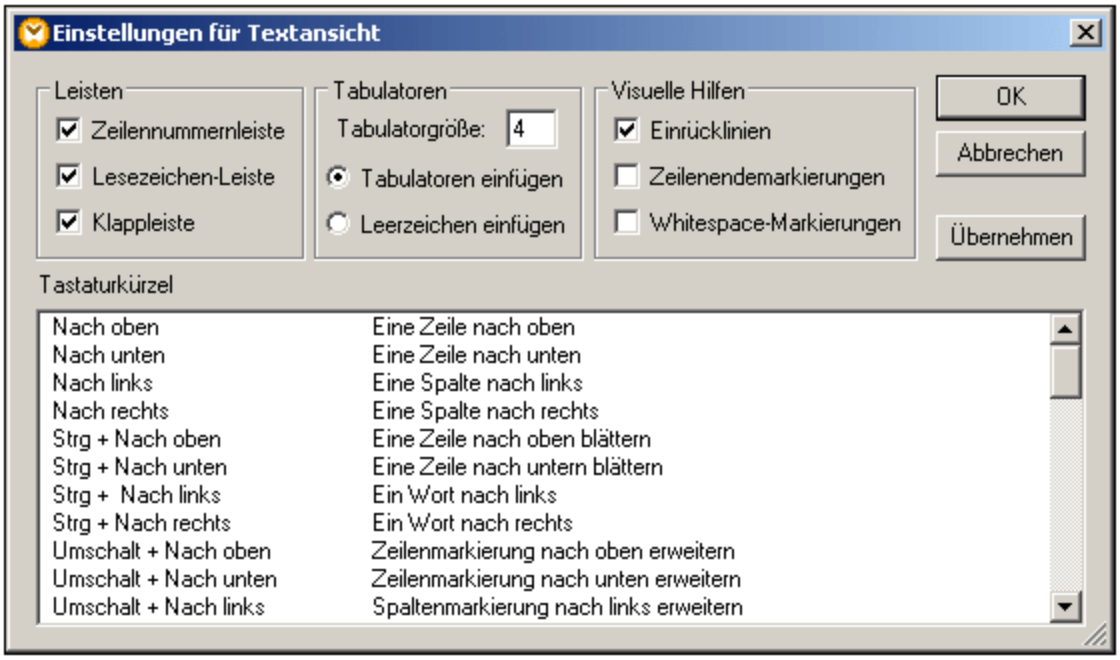
Vorhergehendes Lesezeichen
Navigiert im Fenster Ausgabe zum vorhergehenden Lesezeichen.

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. Die einzelnen Child Nodes werden durch ein einziges Tabulatorzeichen vom übergeordneten Node eingerückt.

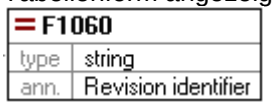
Einstellungen für Textansicht
Dient zum Anpassen der Texteinstellungen im Fenster Ausgabe und zeigt die aktuell definierten Tastaturkürzel an, die im Fenster gelten.

© 2011 Altova GmbH


Einstellungen für Textansicht
Leisten
Zeilennummernleiste
Lesezeichen-Leiste
Klappleiste
Tabulatoren
Tabulatorgröße: 4
Tabulatoren einfügen
Leerzeichen einfügen
Visuelle Hilfen
Einrücklinien
Zeilenendemarkierungen
Whitespace-Markierungen
OK
Abbrechen
Übernehmen
Tastaturkürzel
Nach oben Eine Zeile nach oben
Nach unten Eine Zeile nach unten
Nach links Eine Spalte nach links
Nach rechts Eine Spalte nach rechts
Strg + Nach oben Eine Zeile nach oben blättern
Strg + Nach unten Eine Zeile nach untern blättern
Strg + Nach links Ein Wort nach links
Strg + Nach rechts Ein Wort nach rechts
Umschalt + Nach oben Zeilenmarkierung nach oben erweitern
Umschalt + Nach unten Zeilenmarkierung nach unten erweitern
Umschalt + Nach links Spaltenmarkierung nach links erweitern

17.8 Ansicht

Annotationen anzeigen
Zeigt XML-Schema-Annotationen im Komponentenfenster an.
Wenn auch das Symbol "Datentypen anzeigen" aktiv ist, werden beide Informationen in Tabellenform angezeigt.



Datentypen anzeigen
Zeigt die Schema-Datentypen für jedes Element/Attribut an.
Wenn auch das Symbol "Annotationen anzeigen" aktiv ist, werden beide Informationen in Tabellenform angezeigt.

Bibliothek in Titelleleiste anzeigen
Zeigt den Namen der Bibliothek im Funktionstitel an.

Tipps anzeigen
Zeigt einen Tooltip mit erklärendem Text an, wenn Sie den Mauszeiger über einer Funktion platzieren.

Ausgewählte Komponenten-Konnektoren anzeigen
Wechselt zwischen zwei Ansichten:

- Anzeige aller Mapping-Konnektoren oder
- Anzeige nur jener Konnektoren, die sich auf die aktuell ausgewählten Komponenten beziehen.

Quell- und Zielkonnektoren anzeigen
Wechselt zwischen zwei Ansichten:

- Anzeige von Konnektoren, die mit der aktuell ausgewählten Komponente **direkt** verbunden sind oder
- Anzeige von Konnektoren, die mit der aktuell ausgewählten Komponente verknüpft sind und von der Quellkomponente ausgehen und zur Zielkomponente führen.

Vergrößern/Verkleinern
Öffnet das Dialogfeld "Vergrößern/Verkleinern", in dem Sie entweder einen numerischen Zoom-Faktor eingeben oder den Zoom-Faktor mit Hilfe des Schiebereglers interaktiv einstellen können.

Zurück
Geht auf den Registern der gerade geöffneten Mappings zurück.

Vorwärts
Geht auf den Registern der gerade geöffneten Mappings vorwärts.

Statusleiste
Blendet die unterhalb des Fensters "Meldungen" angezeigte Statusleiste ein oder aus.

© 2011 Altova GmbH

Bibliotheksfenster
Blendet das Bibliotheksfenster, das alle Bibliotheksfunktionen enthält, ein oder aus.

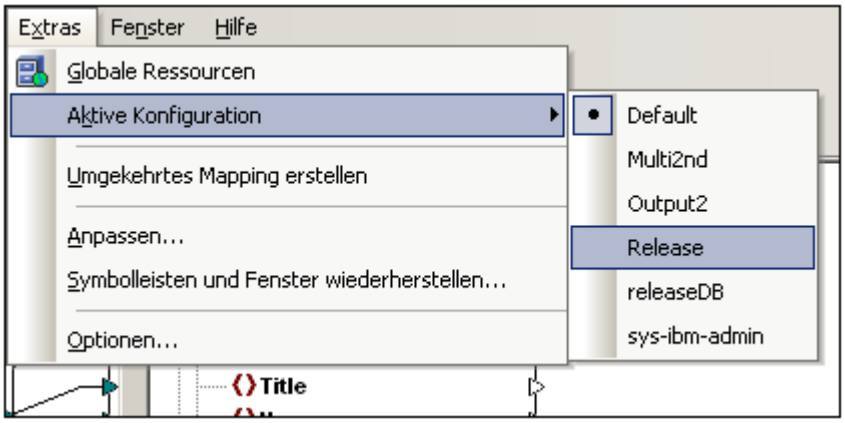
Meldungen
Blendet das Fenster "Meldungen" ein oder aus. Bei Generierung von Code wird das Validierungsfenster automatisch aktiviert, um das Ergebnis der Validierung anzuzeigen.

Übersichtsfenster
Blendet das Übersichtsfenster ein oder aus. Ziehen Sie das Rechteck mit der Maus, um in Ihrer Mapping-Projektansicht zu navigieren.

17.9 Extras

Globale Ressourcen
Öffnet das Dialogfeld "Globale Ressourcen verwalten", in dem Sie globale Ressourcen zur XML-Datei für globale Ressourcen hinzufügen, bearbeiten und daraus löschen können. Nähere Informationen finden Sie unter Globale Ressourcen - Eigenschaften.

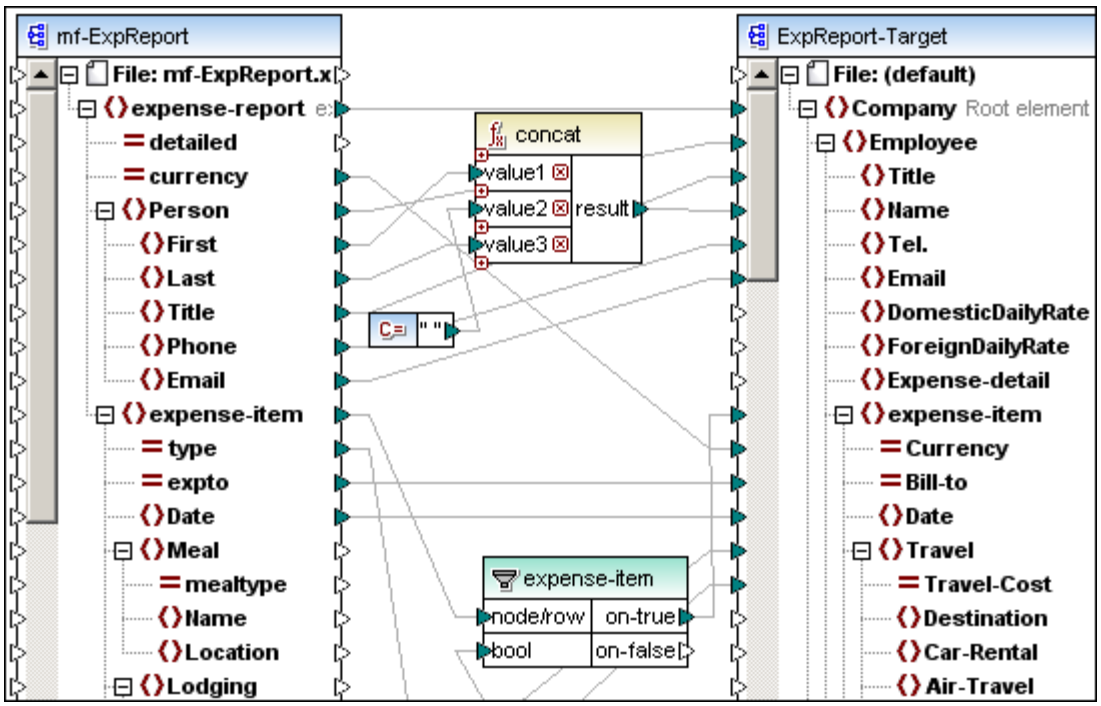
Aktive Konfiguration
Dient zum Auswählen/Wechseln der aktuell aktiven globalen Ressource aus einer Liste aller Ressourcen/Konfigurationen in den globalen Ressourcen. Wählen Sie die gewünschte Konfiguration im Untermenü aus.



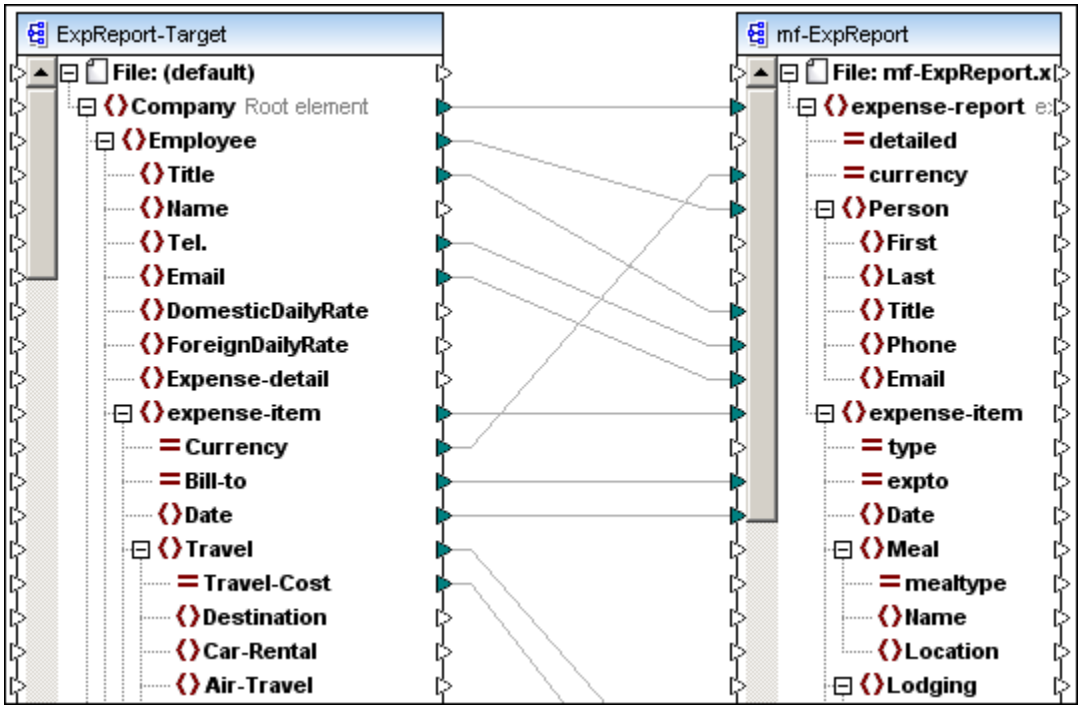
Umgekehrtes Mapping erstellen
Erstellt anhand des gerade in MapForce aktiven Mappings, das die Basis für ein neues Mapping bilden soll, ein "umgekehrtes" Mapping. Beachten Sie, dass das Ergebnis nicht als vollständiges Mapping gedacht ist. Nur die direkten Verbindungen zwischen Komponenten werden im umgekehrten Mapping beibehalten. Das Mapping wird höchstwahrscheinlich nicht gültig sein oder kann nicht ausgeführt werden, wenn Sie auf die Schaltfläche "Ausgabe" klicken, ohne das Mapping vorher manuell zu bearbeiten.

Beispiel: **Tut-ExpReport.mfd** in Ordner ...\MapForceExamples\Tutorial:

© 2011 Altova GmbH



Ergebnis eines umgekehrten Mappings:



Allgemein:

- Die Quellkomponente wird zur Zielkomponente und die Zielkomponente wird zur Quellkomponente.
- Wenn der Komponente eine XML-Input- und eine XML-Output-Instanzdatei zugewiesen wurde, so werden die beiden gegeneinander ausgetauscht.

Beibehaltene Verbindungen

- direkte Verbindungen zwischen Komponenten.
- direkte Verbindungen zwischen Komponenten in einem verketteten Mapping
- die Art der Verbindung: Standard, gemischter Inhalt, Alles kopieren

- Einstellungen für die Weiterleitung einer Komponente
- Datenbankkomponenten bleiben unverändert

Gelöschte Verbindungen

- Verbindungen über Funktionen, Filter werden zusammen mit der Funktion gelöscht
- benutzerdefinierte Funktionen
- Webservice-Komponenten

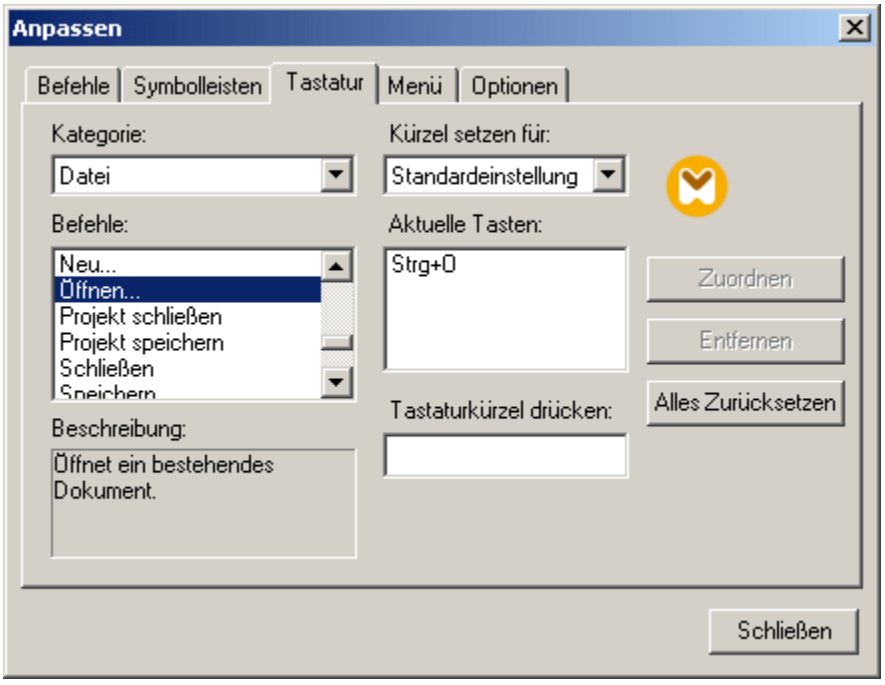
Symbolleisten und Fenster wiederherstellen
Setzt die Symbolleisten, Eingabehilfefenster, angedockten Fenster usw. wieder auf ihre Standardeinstellung zurück. Damit die Änderungen wirksam werden, muss MapForce neu gestartet werden.

Anpassen...
Mit Hilfe des Befehls "Anpassen" können Sie MapForce Ihren Bedürfnissen entsprechend anpassen.

Auf dem Register **Tastatur** können Sie Tastenkürzel für jeden beliebigen MapForce Befehl definieren oder ändern.

So weisen Sie einem Befehl ein neues Tastenkürzel zu:

1. Wählen Sie den Befehl **Extras | Anpassen** und klicken Sie auf das Register "Tastatur".
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. Klicken Sie in das Textfeld **Tastaturkürzel drücken** und drücken Sie die Tasten, mit denen der Befehl aufgerufen werden soll.



 © 2011 Altova GmbH

Die Tasten werden sofort im Textfeld angezeigt. Wenn das Tastaturkürzel bereits vergeben ist, wird diese Funktion unterhalb des Textfelds angezeigt.

5. Klicken Sie auf die Schaltfläche **Zuweisen,** um das Tastaturkürzel zuzuweisen. Das Tastaturkürzel wird nur im Listenfeld "Aktuelle Tasten" angezeigt. (Um den Eintrag im Textfeld "Tastaturkürzel drücken" zu löschen, drücken Sie eine der Steuerungstasten: **Strg, ALT** oder **Umschalt**).

So heben Sie die Zuweisung eines Tastaturkürzels auf oder löschen es:

1. Klicken Sie im Listenfeld "Aktuelle Tasten" auf das gewünschte Tastaturkürzel.
2. Klicken Sie auf die Schaltfläche **Entfernen**.
3. Klicken Sie zur Bestätigung auf die Schaltfläche **Schließen.**.

Kürzel setzen für:

Derzeit keine Funktion.

Liste der aktuellen Tastaturkürzel-Zuweisungen:

Tastaturkürzel nach Taste	
F1	Menü "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 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
Strg + N	Datei neu
Strg + O	Datei öffnen
Strg + S	Datei speichern
Strg + P	Datei drucken
Strg + A	Alles auswählen
Strg + X	Ausschneiden
Strg + C	Kopieren
Strg + V	Einfügen
Strg + Z	Rückgängig
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
Eingabetaste	Bestätigt eine Auswahl

Tastaturkürzel im Fenster "Ausgabe"	
Strg + F2	Lesezeichen einfügen/löschen
F2	Nächstes Lesezeichen
Umschalt + F2	Vorhergehendes Lesezeichen
Strg + Umschalt + F2	Alle Lesezeichen löschen

Tastaturkürzel zum Vergrößern/Verkleinern	
Strg + Mausrad vorwärts	Vergrößern
Strg + Mausrad zurück	Verkleinern
Strg + 0 (Null)	Zoomfaktor zurücksetzen

Optionen
Öffnet das Dialogfeld "Optionen" in dem Sie folgende Möglichkeiten haben:

- Hinzufügen oder Löschen von benutzerdefinierten XSLT-Funktionen.
- Definition **allgemeiner** Einstellungen wie z.B. Standard-Ausgabekodierung für neue Komponenten auf dem Register "Allgemein".
- Definieren, welche Meldungen reaktiviert werden sollen.

Register "Bibliotheken":

- Dient zum Hinzufügen oder Löschen von benutzerdefinierten XSLT- oder Programmiersprachenbibliotheken/-funktionen zu MapForce.

Register "Allgemein":

- Gibt an, ob das Logo, die Copyright-Informationen usw. beim Start und/oder Drucken angezeigt werden soll.
- Richtet Komponenten oder Funktionen beim Ziehen mit der Maus an anderen Komponenten aus.
- Aktivieren/Deaktivieren des Farbverlaufs im Hintergrund
- Beschränken des Annotationstexts in Komponenten auf X Zeilen. Gilt auch für SELECT-Anweisungen, die in einer Komponente zu sehen sind.
- Definition der Standardzeichencodierung für neue Komponenten
- Definition der Standardausgabekodierung für neue Komponenten
- Ausführungs-Timeout für das Register "Ausgabe" bei Anzeige der Vorschau des Mapping-Ergebnisses.
- Gibt an, ob die Ausgabe in **temporäre** Dateien erfolgen soll (Standardeinstellung) oder **direkt** auf die Festplatte geschrieben werden soll, wenn Sie auf die Schaltfläche/das Register "Ausgabe" klicken.

  Achtung: Wenn Sie die Option "Direkt in die endgültigen Output-Dateien schreiben" aktivieren, werden Ausgabedateien überschrieben, ohne dass Sie dies vorher bestätigen müssen.

- Beschränkt die Ausgabe auf X Millionen Zeichen, wenn Sie den Code mit dem Built-In-Ausführungsprozessor generieren. Der Built-In-Ausführungsprozessor ist das

 © 2011 Altova GmbH

einzige Ziel, das XML, CSV und FLF Streaming unterstützt.

Register **Meldungen**:
Über dieses Register können Sie Meldungsfelder, die Sie zu einem früheren Zeitpunkt durch Aktivieren des Kontrollkästchens "Nicht mehr fragen" deaktiviert haben wieder aktivieren.

17.10 Fenster

Überlappend
Mit diesem Befehl ordnen Sie alle offenen Dokumentfenster an, sodass sie übereinander gestapelt und überlappend angezeigt werden.

Horizontal anordnen
Mit diesem Befehl ordnen Sie alle offenen Dokumente **horizontal nebeneinander** an, sodass alle Dokumentfenster gleichzeitig sichtbar sind.

Vertikal anordnen
Mit diesem Befehl ordnen Sie alle offenen Dokumente **vertikal nebeneinander** an, sodass alle Dokumentfenster gleichzeitig sichtbar sind.

1
2
In dieser Liste werden alle offenen Fenster angezeigt, sodass Sie schnell zwischen ihnen wechseln können.
Sie können sich auch mit Strg-TAB oder Strg F6 durch die offenen Fenster schalten.

© 2011 Altova GmbH

17.11 Hilfe

Im Menü **Hilfe** finden Sie Befehle zum Aufrufen der Bildschirmhilfe zu MapForce, Befehle zum Abrufen von Informationen über MapForce und Links zu den Support-Seiten auf der Altova Website. Außerdem enthält das Menü **Hilfe** das Registrierungsdialogfeld, über das Sie nach Erwerb des Produkts den Lizenz-Keycode eingeben können.

Die Beschreibung der Befehle des Menüs **Hilfe** ist in die folgenden Unterabschnitte gegliedert:

- Inhaltsverzeichnis, Index, Suchen
- Registrierung, Bestellformular
- Weitere Befehle

17.11.1 Inhaltsverzeichnis, Index, Suchen

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 gute Übersicht über das gesamte Hilfedokument. Wenn Sie auf einen Eintrag im Inhaltsverzeichnis klicken, gelangen Sie zu diesem Thema.

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.

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. Das Hilfesystem führt eine Volltextsuche am gesamten Hilfedokument durch und gibt eine Trefferliste zurück. Doppelklicken Sie auf einen Eintrag, um das Thema anzuzeigen.

© 2011 Altova GmbH

17.11.2 Aktivierung, Bestellformular, Registrieren, Updates

Software-Aktivierung
Nachdem Sie Ihre Altova-Software heruntergeladen haben, können Sie sie entweder mit Hilfe eines kostenlosen Evaluierungs-Keycode oder eines käuflich erworbenen permanenten Lizenzkeycode aktivieren.

- **Kostenloser Evaluierungs-Keycode.** Wenn Sie die Software zum ersten Mal starten, wird das Dialogfeld "Software-Aktivierung" angezeigt. Es enthält eine Schaltfläche, über die Sie einen kostenlosen Evaluierungs-Keycode anfordern können. Geben Sie in das Dialogfeld, das daraufhin angezeigt wird, Ihren Namen, den Namen Ihrer Firma und Ihre E-Mail-Adresse ein und klicken Sie auf "Jetzt anfordern!". Der Evaluierungs-Keycode wird an die von Ihnen angegebene E-Mail-Adresse gesendet und sollte in wenigen Minuten eintreffen. Geben Sie den Schlüssel nun im Dialogfeld "Software-Aktivierung" in das Feld Key-Code ein und klicken Sie auf OK, um Ihr Altova-Produkt zu aktivieren. Die Software wird für einen Zeitraum von 30 Tagen aktiviert.
- **Permanenter Lizenz Keycode.** Das Dialogfeld "Software-Aktivierung" enthält eine Schaltfläche, über die Sie einen permanenten Lizenz-Keycode erwerben können. Wenn Sie auf diese Schaltfläche klicken, gelangen Sie zum Altova Online Shop, wo Sie einen permanenten Lizenz-Keycode für Ihr Produkt erwerben können. Bei Ihrem permanenten Lizenz-Keycode handelt es sich entweder um eine Einzelbenutzerlizenz oder um eine Mehrfachlizenz. Beide Arten werden per E-Mail zugesandt. Eine *Einzelbenutzerlizenz* enthält Ihre Lizenzdaten, Ihren Namen, die Firma, E-Mail-Adresse und Keycode. Eine *Mehrfachlizenz* enthält Ihre Lizenzdaten, ihren Firmennamen und den Keycode. Beachten Sie, dass gemäß Ihrer Lizenzvereinbarung auf den Computern Ihres Unternehmens nur so viele Versionen der Altova-Software installiert werden dürfen, wie lizenziert wurden (Mehrplatzlizenz)

Anmerkung: Stellen Sie bitte sicher, dass die von Ihnen in das Registrierungsdialogfeld eingegebenen Daten genau mit denen in Ihrer Lizenz-E-Mail übereinstimmen. Geben Sie außerdem bei Mehrplatzlizenzen den Namen des jeweiligen Benutzers in das Feld "Name" ein.

Das Dialogfeld "Software-Aktivierung" kann jederzeit über den Befehl **Hilfe | Software-Aktivierung** aufgerufen werden.

Bestellformular
Sobald Sie eine lizenzierte Version von MapForce bestellen möchten, klicken Sie im Dialogfeld "Software-Aktivierung" (*siehe oben)* auf die Schaltfläche **Permanenten Key-Code erwerben...** um zum sicheren Online-Shop von Altova weitergeleitet zu werden.

Registrierung
Wenn Sie Ihre Altova-Software nach der Aktivierung das erste Mal starten, erscheint ein Dialogfeld, in dem Sie gefragt werden, ob Sie Ihr Produkt registrieren möchten. Es gibt in diesem Dialogfeld drei Schaltflächen:

- **OK:** Leitet Sie zum Registrierungsformular weiter
- **Später erinnern:** Ruft ein Dialogfeld auf, in dem Sie auswählen können, wann Sie das nächste Mal erinnert werden möchten.
- **Abbrechen:** Schließt das Dialogfeld und unterdrückt die Anzeige in Zukunft. Wenn Sie das Produkt zu einem späteren Zeitpunkt registrieren möchten, verwenden Sie den Befehl **Hilfe | Registrieren**

Auf Updates überprüfen
Überprüft, ob am Altova Server eine neuere Version Ihres Produkts vorhanden ist und zeigt

eine entsprechende Meldung an.

© 2011 Altova GmbH

17.11.3 Weitere Befehle

Der Befehl **Support Center** ist ein Link zum Altova Support Center im Internet. Im Support Center finden Sie Antworten auf häufig gestellte Fragen, Diskussionsforen, in denen Sie Software-Probleme besprechen können und ein Formular, um unsere Mitarbeiter vom technischen Support zu kontaktieren.

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.

Der Befehl **Komponenten downloaden** 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.

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.

Der Befehl **MapForce Training** ist ein Link zur Online Training-Seite auf der Altova Website. Hier können Sie aus Online-Kursen, die von ausgebildeten Altova Experten gehalten werden, auswählen.

Mit dem Befehl **Über** MapForce wird das Willkommensfenster und die Versionsnummer Ihres Produkts angezeigt

Kapitel 18

Anhänge

18 Anhänge

Diese Anhänge enthalten technische Informationen über MapForce und wichtige Lizenzinformationen. Die einzelnen Anhänge sind in die folgenden Unterabschnitte gegliedert:

Technische Daten

- Betriebssystem und erforderlicher Arbeitsspeicher
- Altova XML Parser
- Altova XSLT- und XQuery-Prozessor
- Unicode-Unterstützung
- Internet-Verwendung
- Lizenzüberwachung

Lizenzinfomationen

- Electronic Software Distribution
- Copyright
- Endbenutzer-Lizenzvereinbarung (EULA)

© 2011 Altova GmbH

18.1 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.

18.1.1 XSLT 1.0-Prozessor: Implementierungsinformationen

Der Altova XSLT 1.0-Prozessor ist in die Altova XML-Produkte XMLSpy, StyleVision, Authentic und MapForce integriert. Außerdem steht er im kostenlosen AltovaXML-Paket zur Verfügung. Im Altova XSLT 1.0-Prozessor sind die W3C-Spezifikationen implementiert und entsprechen der XSLT 1.0 Recommendation vom 16. November 1999 und der XPath 1.0 Recommendation vom 16. November 1999. Im Folgenden sind Einschränkungen und implementierungsspezifisches Verhalten aufgelistet.

Einschränkungen

- Die Elemente **`xsl:preserve-space`** und **`xsl:strip-space`** werden nicht unterstützt.
- Wenn das Attribut `method` von `xsl:output` auf HTML gesetzt ist, oder wenn standardmäßig HTML output ausgewählt ist, werden Sonderzeichen in der XML- oder XSLT-Datei direkt als Sonderzeichen in die HTML-Dokument eingefügt und nicht als HTML-Zeichenreferenzen in der Ausgabe. So wird z.B. das Zeichen ` ` (die Dezimalzeichenreferenz für ein geschütztes Leerzeichen) nicht als ` ` in den HTML-Code eingefügt sondern direkt als geschütztes Leerzeichen.

Behandlung der Implementierung von Nur-Whitespace Nodes in XML-Quelldokumenten
Aus den XML-Daten (und somit dem XML-Infoset), die an den Altova XSLT 1.0-Prozessor übergeben werden, werden boundary-whitespace-only Text Nodes entfernt. (Ein boundary-whitespace-only Text Node ist ein Textnode, der nur Whitespaces enthält und der zwischen zwei Elementen innerhalb eines mixed-content-Elements vorkommt). Dies kann sich auf den Wert auswirken, der von den Funktionen `fn:position()`, `fn:last()` und `fn:count()` zurückgegeben wird.

In jeder Node-Auswahl, bei der auch Text Nodes ausgewählt werden, sind normalerweise auch Boundary-whitespace-only Text Nodes enthalten. Da jedoch beim XML Infoset, das von den Altova-Prozessoren verwendet wird, boundary-whitespace-only Text Nodes entfernt werden, sind diese Nodes im XML Infoset nicht vorhanden. Folglich unterscheidet sich die Größe der Auswahl und die Nummierung der Nodes von einer, in der diese Text Nodes enthalten sind. Aus diesem Grund können sich die Ergebnisse der Funktionen `fn:position()`, `fn:last()` und `fn:count()` von denen anderer Prozessoren unterscheiden.

Am häufigsten tritt die Situation, dass boundary-whitespace-only Text Nodes als gleichrangige Elemente anderer Elemente überprüft werden, dann auf, wenn `xsl:apply-templates` verwendet wird, um Templates anzuwenden. Wenn die Funktionen `fn:position()`, `fn:last()` und `fn:count()` in Patterns mit einer Namensüberprüfung verwendet werden (z.B. `para[3], kurz für para[position()=3]`), sind boundary-whitespace-only Nodes irrelevant, da nur die benannten Elemente (`para` im obigen Beispiel) ausgewählt werden. (Beachten Sie jedoch, dass boundary-whitespace-only Nodes in Patterns, in denen ein Platzhalterzeichen wie z.B. `*[10]` verwendet wird, sehr wohl relevant sind).

Hinweis: Wenn der boundary-whitespace-only Text Node in der Ausgabe benötigt wird, fügen Sie das erforderliche Whitespace-Zeichen in eines der benachbarten Child-Elemente ein. So erzeugt z.B. das XML-Fragment:

```
<para>This is <b>bold</b> <i>italic</>.</para>
```

bei Verarbeitung mit dem XSLT Template

```
<xsl:template match="para">
   <xsl:apply-templates/>
</xsl:template>
```

folgendes Resultat:

```
This is bolditalic.
```

Um in der Ausgabe ein Leerzeichen zwischen `bold` und `italic` zu erhalten, fügen Sie entweder in das Element `<b>` oder `<i>` in der XML-Quelle ein Leerzeichen ein. z.B.:

```
<para>This is <b>bold</b> <i> italic</i>.</para> or
<para>This is <b>bold&#x20;</b> <i>italic</i>.</para> or
<para>This is <b>bold</b><i>&#x20;italic</i>.</para>
```

Wenn eines der oben genannten `para`-Elemente mit demselben XSLT Template verarbeitet wird, erhalten Sie folgendes Ergebnis:

```
This is bold italic.
```

18.1.2 XSLT 2.0-Prozessor: Implementierungsinformationen

Der Altova XSLT 2.0-Prozessor ist in die Altova XML-Produkte XMLSpy, StyleVision, Authentic und MapForce integriert. Außerdem steht er im kostenlosen AltovaXML-Paket zur Verfügung. In diesem Abschnitt werden die implementierungsspezifischen Aspekte des Prozessors beschrieben. Im ersten Teil des Abschnitts finden Sie allgemeine Informationen über den Prozessor. Im Anschluss daran finden Sie eine Liste der implementierungsspezifischen Aspekte der XSLT 2.0-Funktionen.

Informationen über das implementierungsspezifische Verhalten von XPath 2.0-Funktionen finden Sie im Abschnitt XPath 2.0- und XQuery 1.0-Funktionen.

Allgemeine Informationen

Der Altova XSLT 2.0-Prozessor entspricht der XSLT 2.0 Recommendation vom 23 January 2007. Beachten Sie bitte die folgenden allgemeinen Informationen über den Prozessor.

Rückwärtskompatibilität

Der Altova 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 von Altova XML 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.

Bei allen anderen Altova-Produkten wird Rückwärtskompatibilität nie benötigt, da automatisch der entsprechende Prozessor für die Transformation ausgewählt wird. Angenommen, Sie legen fest, dass ein bestimmtes XML-Dokument in XMLSpy mit einem XSLT 1.0 Stylesheet verarbeitet werden soll. Bei Aufruf des Transformationsbefehls wählt XMLSpy zur Transformation automatisch den XSLT 1.0-Prozessor aus.

Anmerkung:
Diese Auswahl basiert auf dem Wert im Attribut `version` des Stylesheet-Elements `stylesheet` oder `transform`.

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"
```

 © 2011 Altova GmbH

```
   ...
</xsl:stylesheet>
```

Beachten Sie bitte die folgenden Punkte:

- Der Altova XSLT 2.0-Prozessor verwendet als Default Functions 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-Konstrukturoen 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.
- In den Candidate Recommendations vom 23 January 2007 wurden die Datentypen untypedAtomic und duration (dayTimeDuration und yearMonthDuration), die sich zuvor im XPath Datatypes Namespace befanden (normalerweise mit dem Präfix xdt:) in den XML-Schema-Namespace verschoben.
- 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 Altova XSLT 2.0-Prozessor ist schemafähig.

Whitespaces im XML-Dokument
Standardmäßig entfernt der Altova XSLT 2.0-Prozessor alle boundary whitespaces aus boundary-whitespace-only Nodes im XML-Quelldokument. Dies wirkt sich auf die Werte aus, die die Funktionen `fn:position()`, `fn:last()`, `fn:count()` und `fn:deep-equal()` zurückgeben. Nähere Informationen dazu finden Sie in den Abschnitten über XPath 2.0- und XQuery 1.0-Funktionen unter Nur-Whitespace Nodes im XML-Quelldokument.

Hinweis: Wenn in der Ausgabe ein boundary-whitespace-only-Text Node erforderlich ist, fügen Sie das gewünschte Whitespace-Zeichen in eines der beiden benachbarten Child-Elemente ein. So erzeugt z.B. das XML-Fragment:

```
<para>This is <b>bold</b> <i>italic</>.</para>
```

bei Verarbeitung mit dem XSLT Template

```
<xsl:template match="para">
   <xsl:apply-templates/>
</xsl:template>
```

folgendes Resultat:

```
This is bolditalic.
```

Um in der Ausgabe ein Leerzeichen zwischen `bold` und `italic` zu erhalten, fügen Sie entweder in das Element `<b>` oder `<i>` in der XML-Quelle ein Leerzeichen ein. z.B.:

```
<para>This is <b>bold</b> <i> italic</>.</para> oder
<para>This is <b>bold&#x20;</b> <i>italic</>.</para> oder
<para>This is <b>bold</b><i>&#x20;italic</>.</para>
```

Wenn nun ein solches XML-Fragment mit demselben XSLT Template verarbeitet wird, erhalten Sie folgendes Ergebnis:

```
This is bold italic.
```

XSLT 2.0-Elemente und -Funktionen
Einschränkungen und implementierungsspezifisches Verhalten von XSLT 2.0-Elementen und -Funktionen werden im Abschnitt XSLT 2.0-Elemente und -Funktionen aufgelistet.

XPath 2.0-Funktionen
Implementierungsspezifisches Verhalten von XPath 2.0-Funktionen wird im Abschnitt XPath 2.0- und XQuery 1.0-Funktionen aufgelistet.

XSLT 2.0-Elemente und -Funktionen

Einschränkungen
Die Elemente **`xsl:preserve-space`** und **`xsl:strip-space`** werden nicht unterstützt.

Implementierungsspezifisches Verhalten
Im Folgenden finden Sie eine Beschreibung, wie der Altova 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 2.0, XPath 2.0 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 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 Kodierungen `x-binarytobase16` und `x-binarytobase64` unterstützt.

Anmerkung: Die folgenden Kodierungswerte, die in früheren Versionen von AltovaXML implementiert waren, werden nun nicht mehr verwendet: `base16tobinary`, `base64tobinary`, `binarytobase16` und `binarytobase64`.

© 2011 Altova GmbH

18.1.3 Erweiterungen

Es gibt in Programmiersprachen wie Java und C# eine Reihe von fertigen Funktionen, die nicht als XPath 2.0 / XQuery 1.0-Funktionen oder XSLT 2.0-Funktionen zur Verfügung stehen. Ein gutes Beispiel für solche Funktionen 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 erweitern und die Erstellung von Stylesheets wäre viel einfacher.

Die Altova-Prozessoren (XSLT 1.0, XSLT 2.0 und XQuery 1.0), die in einer Reihe von Altova-Produkten verwendet werden, unterstützen die Verwendung von Erweiterungsfunktionen in Java und .NET. Zusätzlich unterstützen die Altova XSLT-Prozessoren MSXSL Scripts für XSLT 1.0 und 2.0.

Beachten Sie, dass Erweiterungsfunktionen immer von XPath-Ausdrücken aus aufgerufen werden. In diesem Abschnitt wird beschrieben, wie Sie Erweiterungsfunktionen und MSXSL Scripts in Ihren XSLT Stylesheets und XQuery-Abfragen verwenden können.

Haupsächlich 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) installiert sein oder es muss Zugriff auf eine solche bestehen.

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
- Java: Statische Methoden und statische Felder
- Java: Instanzmethoden und Instanzfelder
- Datentypen: XSLT/XQuery in Java
- Datentypen: Java in XSLT/XQuery

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>
```

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 und Benutzerdefinierte Jar-Dateien beschrieben. Pfade zu Klassendateien, die sich nicht im aktuellen Verzeichnis befinden, und Pfade zu allen JAR-Dateien müssen jedoch angegeben werden.

Anmerkung: Wenn Sie einen Namespace zu einem XSLT-Stylesheet hinzufügen möchten, das anhand eines in SPS in StyleVision generiert wurde, muss der Namespace zum `schema`-Element der obersten Ebene des XML-Schemas, auf dem das SPS basiert, hinzugefügt werden. Beachten Sie, dass die folgende Namespace Deklaration `xmlns:java="java"` standardmäßig automatisch in jedem SPS erstellt wird, das in StyleVision entworfen wurde.

Benutzerdefinierte Klassendateien

Wenn der Zugriff über eine Klassendatei erfolgt, gibt es zwei Möglichkeiten:

 © 2011 Altova GmbH

- Die Klassendatei befindet sich in einem Paket. Die XSLT-oder XQuery-Datei befindet sich im selben Ordner wie das Java-Paket.
- Die Klassendatei befindet sich nicht in einem Paket. Die XSLT-oder XQuery-Datei befindet sich im selben Ordner wie die Klassendatei.
- Die Klassendatei befindet sich in einem Paket. Die XSLT-oder XQuery-Datei befindet sich in irgendeinem beliebig gewählten Ordner.
- Die Klassendatei befindet sich nicht in einem Paket. Die XSLT-oder XQuery-Datei befindet sich in irgendeinem beliebig gewählten Ordner.

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 befindet sich nicht in einem Paket, die XSLT/XQuery-Datei befindet sich im selben Ordner wie die Klassendatei

Im Beispiel unten wird die Methode `getVehicleType()` der Klasse `Car` class des Pakets `com.altova.extfunc` aufgerufen. Die Klassendatei `Car` class befindet sich im folgenden Ordner: `JavaProject/com/altova/extfunc`. 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` class 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
`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 befindet sich nicht in einem Paket, die XSLT/XQuery-Datei befindet sich in einem beliebigen Ordner

Im Beispiel unten wird die Methode `getCarColor()`der Klasse `Car` class 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. Der Pfad der Klassendatei wird in der Namespace-URI als Abfragestring definiert. 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

 © 2011 Altova GmbH

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.

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>
</xsl:template>

<xsl:template match="car"/>

</xsl:stylesheet>
```

Anmerkung: Wenn ein Pfad über eine Erweiterungsfunktion angegeben wird, wird er zum ClassLoader hinzugefügt.

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, dann gibt die Java-Erweiterungsfunktion eine Sequenz zurück, bei der es sich um einem XPath/XQuery-Datentyp handelt. Wenn das Ergebnis eines Kontruktoraufrufs 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*):
  ```
  <xsl:value-of select="date:toString(date:new())" xmlns:date="
  java:java.util.Date" />
  ```

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"
```

 © 2011 Altova GmbH

```
                    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>
```

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:

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` bereitszustellen.

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 einige Fällen ist es nicht möglich, auf Basis der verfügbaren Infomationen 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 Datentypes erhält: `mymethod(double)`.

 © 2011 Altova GmbH

- 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.

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.

.NET-Erweiterungsfunktionen

Wenn Sie 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 Konstruktur, eine Eigenschaft oder Methode (statische oder Instanz) in einer .NET-Klasse aufzurufen.

Eine Eigenschaft einer .NET-Klasse wird mit der Syntax `get_PropertyName()` aufgerufen.

Dieser Abschnitt ist in die folgenden Unterabschnitte gegliedert:

- .NET: Konstruktoren
- .NET: Statische Methoden und statische Felder
- .NET: Instanzmethoden und Instanzfelder
- Datentypen: XSLT/XQuery in .NET
- Datentypen: .NET in XSLT/XQuery

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 Klasen 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

 © 2011 Altova GmbH

```
       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.

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.

.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, 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) auf den erforderlichen XPath-Datentyp gecastet 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*):
  ```
  <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 Boolschen Ausdruck konvertiert werden:
- ```
  <xsl:value-of select="xs:integer(data:get_Month(date:new(2008, 4, 29)))
  " xmlns:date="clitype:System.DateTime" />
  ```

.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 Methodenmane 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="
```

 © 2011 Altova GmbH
```

```
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>
```

.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 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.

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 einige Fällen ist es nicht möglich, auf Basis der verfügbaren Infomationen 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 Datentypes 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.

 © 2011 Altova GmbH

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.

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.

MSXSL Scripts 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.

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>
```


© 2011 Altova GmbH

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.

Altova Erweiterungsfunktionen

Altova Erweiterungsfunktionen befinden sich im Namespace **`http://www.altova.com/xslt-extension`** und werden in diesem Abschnitt mit dem Präfix **`altova:`**gekennzeichnet, das an den oben angegebenen Namespace gebunden ist.

In der aktuellen Version Ihres Altova-Produkts werden die folgenden Erweiterungsfunktionen 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 übr die Unterstützung für Altova Erweiterungsfunktionen in der jeweiligen Release zu erhalten, schlagen Sie bitte in der Dokumentation der jeweils aktuellen Release nach.

Allgemeine Funktionen

- **`altova:evaluate()`**
- **`altova:distinct-nodes()`**
- **`altova:encode-for-rtf()`**
- **`altova:xbrl-labels()`**
- **`altova:xbrl-footnotes()`**
- **altova:generate-auto-number()**
- **`altova:reset-auto-number()`**
- **`altova:get-temp-folder()`**

Diagrammfunktionen (Nur Enterprise und Reporting Edition)

Die Altova-Erweiterungsfunktionen für Diagramme werden nur in der Enterprise und der Reporting Edition von Altova Produkten unterstützt und gestatten die Erstellung von Diagrammen anhand von XML-Daten. Die Diagrammfunktionen wurden in zwei Gruppen unterteilt:

- Funktionen zum Generieren und Speichern von Diagrammen
- Funktionen zum Erstellen von Diagrammen

Ein dritter Abschnitt enthält eine Liste der Diagrammdaten-XML-Struktur, anhand der

Diagramme generiert werden können und zum Schluss finden Sie im XSLT-Beispieldokument ein Beispiel, wie Diagrammfunktionen verwendet werden können, um anhand von XML-Daten Diagramme zu generieren.

Allgemeine Funktionen

In der aktuellen Version Ihres Altova-Produkts werden die folgenden Erweiterungsfunktionen 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.

- **altova:evaluate()**
- **altova:distinct-nodes()**
- **altova:encode-for-rtf()**
- **altova:xbrl-labels()**
- **altova:xbrl-footnotes()**
- **altova:generate-auto-number()**
- **altova:reset-auto-number()**
- **altova:get-temp-folder()**

altova:evaluate()

Die `altova:evaluate()` Funktion erhält einen XPath-Ausdruck als obligatorisches Argument, der als String übergeben wird, und gibt das Resultat des ausgewerteten Ausdrucks zurück.

```
altova:evaluate(XPathExp as xs:string)
```

Beispiel:

```
altova:evaluate('//Name[1]')
```

Beachten Sie im obigen Beispiel, dass der Ausdruck `//Name[1]` durch Einschließen in einfache Anführungszeichen als String übergeben wird. Die Funktion `altova:evaluate` gibt den Inhalt des ersten `Name`-Elements im Dokument zurück.

Die Funktion `altova:evaluate` kann zusätzliche (optionale) Argumente erhalten. Diese Argumente sind die Werte der einzelnen Variablen und haben die Namen `p1`, `p2`, `p3`... `pN`, die im XPath-Ausdruck verwendet werden können.

```
altova:evaluate(XPathExp as xs:string [, p1value ... pNvalue])
```

wobei

- die Variablennamen die Form `pX` haben müssen, wobei `X` eine Ganzzahl ist
- die Reihenfolge der Argumente der Funktion vom zweiten Argument an der Reihenfolge der Variablen mit den Namen `p1` bis `pN` entspricht. Daher wird das zweite Argument der Wert der Variablen `p1`, das dritte Argument der der Variablen `p2`, usw.
- die Werte der Variablen den Typ `item*` haben müssen

Beispiel:

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate( $xpath, 10, 20, 'hi' )" />
```

 © 2011 Altova GmbH

Gibt "hi 20 10" zurück

Beachten Sie im obigen Beispiel folgende Punkte:

- Das zweite Argument des Ausdrucks `altova:evaluate` ist der der Variablen `$p1` zugewiesene Wert, das dritte Argument ist das der Variblen `$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..

Im Folgenden ein weiteres Beispiel für die Verwendung:

```
<xsl:variable name="xpath" select="'$p1'" />
<xsl:value-of select="altova:evaluate( $xpath, //Name[1] )" />
```
Gibt den Wert des ersten `Name` *Elements aus.*

```
<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`)

Variablen können in der Erweiterungsfunktion `altova:evaluate()` wie im Beispiel unten gezeigt verwendet werden:

- 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*

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.

`altova:distinct-nodes()`
Die `altova:distinct-nodes()` Funktion 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-equal` durchgeführt.

```
altova:distinct-nodes( $arg as node()* )  as node()*
```

`altova:encode-for-rtf()`
Die `altova:encode-for-rtf()` Funktion konvertiert den Input-String in Code für RTF.

```
altova:encode-for-rtf( $inputstr as xs:string?,
$preserveallwhitespace as xs:boolean,
$preservenewlines as xs:boolean) as xs:string
```

Whitespaces und neue Zeilen werden gemäß dem für die entsprechenden Parameter definierten Booleschen Wert beibehalten.

`altova:xbrl-labels()`
Die `altova:xbrl-labels()` Funktion 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.

```
altova:xbrl-labels( $name as xs:QName, $file as xs:string )  as node()*
```

`altova:xbrl-footnotes()`
Die `altova:footnotes()` Funktion erhält einen Node als Input-Argument und gibt die durch den Input-Node referenzierte Gruppe der XBRL-Fußnoten-Nodes zurück.

```
altova:footnotes( $arg as node() )  as node()*
```

`altova:generate-auto-number(id als xs:string, start-with als xs:integer, increment als xs:integer, reset-on-change als xs:string)`
Generiert eine Reihe von Zahlen mit der definierten ID. Die Anfangszahl (Ganzzahl) und die Inkrementierung werden definiert.

`altova:reset-auto-number(id als xs:string)`
Diese Funktion setzt die automatische Nummerierung der mit dem ID-Argument definierten

 © 2011 Altova GmbH

automatischen Nummerierung zurück. Die Reihe wird auf den Anfangsganzzahlwert der Reihen zurückgesetzt (siehe `altova:generate-auto-number` oben).

`altova:get-temp-folder als xs:string`
Ruft den temp-Ordner ab.

18.2 Technische Daten

Dieses Kapitel enthält nützliche Hintergrundinformationen zu den technischen Aspekten Ihrer Software. Es ist in die folgenden Abschnitte gegliedert:

- OS- und Arbeitsspeicheranforderungen
- Altova XML Parser
- Altova XSLT- und XQuery-Prozessor
- Unicode-Unterstützung
- Internet-Verwendung

© 2011 Altova GmbH

18.2.1 OS- und Arbeitsspeicheranforderungen

Betriebssystem
Die Altova-Software-Applikationen sind:

- 32-Bit Windows-Applikationen für Windows XP Windows Server 2003 und 2008, Windows Vista und Windows 7 oder
- 64-Bit-Applikationen für Windows Vista und Windows 7

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.

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.

18.2.2 Altova XML Parser

Wenn Sie ein XML-Dokument öffnen, verwendet die Applikation den integrierten Validierungsparser (den Altova XML Parser), um das Dokument auf Wohlgeformtheit zu prüfen, es gegen ein Schema zu validieren (falls eines angegeben wurde) und Baumstrukturen und Infosets zu erstellen. Der Altova XML Parser dient auch dazu, beim Editieren von Dokumenten intelligente Eingabehilfen zur Verfügung zu stellen und etwaige Validierungsfehler dynamisch anzuzeigen.

Im integrierten Altova XML Parser ist die Final Recommendation der W3C XML Schema Spezifikation implementiert. Neue Entwicklungen, die von der XML Schema-Arbeitsgruppe empfohlen werden, werden ständig in den Altova Parser integriert, sodass Ihnen mit Altova-Produkten immer eine Entwicklungsumgebung auf dem neuesten Stand der Technik zur Verfügung steht.

© 2011 Altova GmbH

18.2.3 Altova XSLT- und XQuery-Prozessor

Die Altova Produkte arbeiten mit dem Altova XSLT 1.0-Prozessor, dem Altova XSLT 2.0-Prozessor und dem Altova XQuery-Prozessor. Nähere Informationen zu diesen Prozessoren finden Sie im Anhang 1 zu dieser Dokumentation im Abschnitt über Prozessoren.

Diese drei Prozessoren stehen auch als eigenständige Applikationen in Form des AltovaXML -Pakets zum kostenlosen Download von der Altova Website zur Verfügung. Die Dokumentation zur Verwendung der Prozessoren wird mit dem AltovaXML-Paket mitgeliefert.

18.2.4 Unicode-Unterstützung

Unicode ist ein 16-Bit-Zeichensatzstandard (der auf 32-Bit erweiterbar ist), der vom Unicode-Konsortium definiert wurde und der eine eindeutige Zahl für jedes Zeichen zur Verfügung stellt, und zwar:

- plattformunabhängig
- programmunabhängig
- unabhängig von der Sprache

Im Prinzip arbeiten Computer nur mit Zahlen. Sie speichern Buchstaben und andere Zeichen unter einer Zahl, die sie diesen zuweisen. Vor der Entwicklung von Unicode gab es hunderte von verschiedenen Codiersystemen für die Zuweisung dieser Zahlen. Es gab keine einzige Codierung, die Platz für genügend Zeichen bot: So werden z.B. in der Europäischen Union mehrere verschiedene Codierungen benötigt, um alle ihre Sprachen abzudecken. Selbst für eine einzige Sprache wie z.B. Englisch reichte eine einzelne Codierung nicht aus, um alle gebräuchlichen Buchstaben, Satzzeichen und technischen Symbole abzudecken.

Diese Codierungssysteme standen früher miteinander in Konflikt, d.h. zwei Codierungen verwendeten dieselbe Zahl für zwei verschiedene Zeichen oder verschiedene Zahlen für dasselbe Zeichen. Jeder Computer (v.a. Server) muss viele verschiedene Codierungen unterstützen. Wenn aber Daten zwischen verschiedenen Codierungen oder Plattformen ausgetauscht werden, besteht immer das Risiko, dass Fehler entstehen.

Unicode ist die Lösung für diese Probleme!
Unicode enthält eine eindeutige Zahl für jedes Zeichen, egal für welche Plattform, welches Programm oder welche Sprache. Der Unicode-Standard wurde von branchenführenden Unternehmen wie Apple, HP, IBM, JustSystems, Microsoft, Oracle, SAP, Sun, Base und vielen anderen übernommen.

Unicode ist für moderne Standards wie XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML, usw. erforderlich und ist das offizielle Verfahren zum Implementieren von ISO/IEC 10646. Unicode wird von vielen Betriebssystemen, allen modernen Browsern und vielen anderen Produkten unterstützt. Die Entwicklung des Unicode-Standards und die Verfügbarkeit von Tools, die diesen unterstützen, gehören zu den wichtigsten globalen Software Technologie-Trends in letzter Zeit.

Die Integration von Unicode in Client-Server oder Multi-Tier-Applikationen und Websites bietet beträchtliche Kostenvorteile im Vergleich zur Verwendung herkömmlicher Zeichensätze. Dank Unicode kann ein einziges Software-Produkt oder eine einzige Website für mehrere Plattformen, Sprachen und Länder konzipiert werden, ohne dass es neu entwickelt werden muss. Daten können dadurch durch viele verschiedene Systeme übertragen werden, ohne dass dabei Fehler auftreten.

Windows XP

Die XML-Produkte von Altova bieten vollständige Unicode-Unterstützung. Um ein XML-Dokument von einem System mit nicht lateinischer Zeichensatzcodierung bearbeiten zu können, benötigen Sie allerdings auch 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. Aus diesem Grund kann es vorkommen, dass ein XML-Dokument "nicht druckbare" Zeichen enthält, da die gewählte Schriftart die erforderlichen Glyphen nicht enthält. Es ist daher manchmal nützlich, wenn man eine Schriftart hat, die den gesamten

 © 2011 Altova GmbH

Unicode-Bereich abdeckt - v.a. wenn Sie XML-Dokumente aus der ganzen Welt editieren.

Die größte uns bekannte Universalschriftart ist "Arial Unicode MS", eine Schriftart, die von Agfa Monotype für Microsoft entwickelt wurde. Diese Schriftart enthält mehr als 50.000 Glyphen und deckt das gesamte im Unicode 2.1.-Standard definierte Zeichenspektrum ab. Die Schriftart ist in Microsoft Office 2000 enthalten und beansprucht 23 MB.

Wir empfehlen Ihnen dringend, diese Schriftart auf Ihrem System zu installieren und mit der Applikation zu verwenden, wenn Sie öfter Dokumente bearbeiten, die in unterschiedlichen Schriftsystemen verfasst wurden. Diese Schriftart wird bei Verwendung der Standardinstallation von Microsoft Office nicht installiert, Sie können jedoch die Option "Benutzerdefinierte Installation" wählen, um sie zu installieren.

Im Ordner "Examples" finden Sie außerdem eine neue XHTML-Datei mit dem Namen `Unicode-UTF8.html`, die den Satz "Wenn die Welt miteinander spricht, spricht sie Unicode" in vielen verschiedenen Sprachen und Schriftsystemen enthält ("When the world wants to talk, it speaks Unicode") (世界的に話すなら、Unicode です。) - Diese Zeile stammt aus der 10. Unicode-Konferenz von 1997 und ist ein schönes Beispiel für die Bedeutung von Unicode für den XML-Standard. Wenn Sie diese 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.

Rechts nach Links-Schreibrichtung

Beachten Sie bitte, dass selbst unter Windows NT 4.0 Texte aus Schriftsystemen mit einer Schreibrichtung von rechts nach links (z.B. Hebräisch oder Arabisch) nur in den Ländern korrekt wiedergegeben werden, in denen solche Schriftsysteme tatsächlich verwendet werden. Dies liegt daran, dass nur die hebräische und arabische Version von Windows NT die Wiedergabe und Bearbeitung von Text in Rechts-links-Schreibweise auf Betriebssystemebene unterstützen.

18.2.5 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.
- Falls Sie den URL-Modus des Dialogfelds "Öffnen" verwenden, um ein Dokument direkt über eine URL zu öffnen (**Datei | Öffnen | Zu URL wechseln**), wird dieses Dokument über eine HTTP-Verbindung (Port 80) aufgerufen. (*Diese Funktionalität steht in XMLSpy und Authentic Desktop zur Verfügung.)*
- 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 es ebenfalls über eine HTTP-Verbindung (Port 80) aufgerufen, sobald Sie das XML-Dokument validieren. Dies kann auch automatisch beim Öffnen eines Dokuments vorkommen, wenn Sie die Applikation so konfiguriert haben, dass Dateien beim Öffnen automatisch validiert werden (Register "Datei" des Dialogfelds **Extras | Optionen**).
- Wenn Sie den Befehl "Als Mail senden..." (**Datei | Als Mail senden**) 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 in diesem Handbuch und der Altova-Software-Lizenzvereinbarung beschriebenen Software-Aktivierung und beim Live-Update..

© 2011 Altova GmbH

18.3 Lizenzinformationen

Dieser Anhang enthält die folgenden Informationen:

- Informationen über den Vertrieb dieses Software-Produkts
- Informationen im Zusammenhang mit dem intellektuellen Eigentum dieses Software-Produkts
- Die Endbenutzer-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.

18.3.1 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 testen, bevor Sie sich zu einem Kauf entscheiden.
- Wenn Sie sich entschieden haben, die Software zu kaufen, können Sie Ihre Bestellung online abgeben und das Produkt steht Ihnen in Minutenschnelle im vollen Lizenzumfang zur Verfügung.
- Sie erhalten immer die neueste Version unserer Software
- Die Software enthält ein umfassendes Online-Hilfesystem. 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 weiterverwenden möchten, müssen Sie eine Software-Produktlizenz erwerben, die Sie in Form eines Keycodes erhalten, der in das Dialogfeld "Software-Aktivierung" eingegeben werden muss, um das Produkt zu aktivieren. Sie können Ihre Lizenz online in unserem Web-Shop unter Altova Shop 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, sollten 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.

Nähere Informationen finden Sie in der Altova Software-Lizenzvereinbarung am Ende dieses Handbuchs.

© 2011 Altova GmbH

18.3.2 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 Datagram 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.

Mehrfachlizenz
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 http://www.isi.edu/in-notes/iana/assignments/port-numbers) und unser Lizenzüberwachungsmodul basiert auf einer bewährten und getesteten 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 zwischen verschiedenen Gruppen in Ihrem Unternehmen natürlich blockieren, solange Sie mit anderen Mitteln sicherstellen können, dass Ihre Lizenzvereinbarung eingehalten wird.

Sie werden außerdem auch feststellen, dass Ihr Altova Produkt viele nützliche Funktionen enthält, die Ihre Internet-Verbindung verwenden, diese stehen jedoch nicht mit der Lizenzüberwachungsfunktion in Zusammenhang.

18.3.3 Rechte am geistigen Eigentum

Die Software und alle Kopien, deren Anfertigung Ihnen von Altova gestattet ist, sind das geistige Eigentum von Altova und dessen Lieferanten und stehen in deren Besitz. Struktur, Organisation und Code der Software stellen wertvolle Betriebsgeheimnisse und vertrauliche Informationen von Altova und dessen Lieferanten dar. Die Software ist durch gesetzliche Bestimmungen urheberrechtlich geschützt. Diese gesetzlichen Bestimmungen beinhalten (ohne darauf beschränkt zu sein) das Urheberrecht der Vereinigten Staaten, internationale Verträge und das in den Ländern, in denen die Software genutzt wird, geltende Recht. Altova behält sich das Eigentum an allen Patenten, Urheberrechten, Branchengeheimnissen, Warenzeichen und sonstigen geistigen Eigentumsrechten, die hinsichtlich der Software bestehen, vorbehält. Das Eigentumsrecht von Altova erstreckt sich auch auf alle Bilder, Fotografien, Animationen, Videos, Audioaufzeichnungen, Musikstücke, Texte und „Applets“, die Teil der Software sind, und alle dazugehörigen Unterlagen in gedruckter Form. Mitteilungen über geltend gemachte Verstöße gegen geistige Eigentumsrechte sind an den Copyright Agenten von Altova zu richten (nähere Angaben dazu finden Sie auf der Website von Altova).

Altova Software enthält Software-Komponenten Dritter, die ebenfalls urheberrechtlich geschützt sind. Unter anderem sind diese Urheberrechte ausführlich unter http://www.altova.com/legal_3rdparty.html aufgelistet.

Alle anderen Namen oder Warenzeichen sind das Eigentum ihrer jeweiligen Inhaber.

© 2011 Altova GmbH

18.3.4 Altova Endbenutzer-Lizenzvereinbarung

DIES IST EIN RECHTSGÜLTIGES DOKUMENT -- BITTE BEWAHREN SIE ES SORGFÄLTIG AUF

ALTOVA® ENDBENUTZER-LIZENZVERTRAG

Lizenzgeber:
Altova GmbH
Rudolfsplatz 13a/9
A-1010 Wien
Österreich

Wichtig – Bitte sorgfältig lesen. Benutzerinformation:

Dieser Endbenutzer-Lizenzvertrag („Lizenzvertrag“) ist ein rechtsgültiger Vertrag, der zwischen Ihnen und Altova GmbH („Altova“) abgeschlossen wurde. Bevor Sie die von Altova bereitgestellte Software („Software“) sowie die dazugehörige Dokumentation (u.a. Unterlagen in gedruckter Form, Online-Dateien oder Dokumentation in elektronischer Form („Dokumentation“) verwenden, lesen Sie bitte dieses Dokument sorgfältig durch. Indem Sie auf die untenstehenden Schaltflächen „Ich akzeptiere“ und „Weiter“ klicken, oder indem Sie die Software installieren oder auf sonstige Weise nutzen, stimmen Sie zu, dass Sie die Bedingungen dieses Lizenzvertrags sowie die Datenschutzbestimmungen („Datenschutzbestimmungen“) von Altova, inklusive, jedoch nicht ausschließlich, die nachfolgend festgelegten Garantieausschlüsse, Haftungsbeschränkungen sowie Datennutzungs- und Kündigungsregelungen als rechtsverbindlich anerkennen, auch wenn Sie die Software nicht käuflich erwerben. Sie stimmen zu, dass diese Vereinbarung in demselben Umfang gültig und durchsetzbar ist, wie ein von Ihnen persönlich unterzeichneter Vertrag. Wenn Sie den Bedingungen dieses Lizenzvertrags nicht zustimmen, sind Sie nicht zur Verwendung der Software berechtigt und müssen alle heruntergeladenen Kopien, die sich in Ihrem Besitz oder unter Ihrer Kontrolle befinden, vernichten. Während des Installationsvorganges können Sie eine Kopie dieses Lizenzvertrages zum Zeitpunkt Ihrer Zustimmung ausdrucken. Darüber hinaus können Sie unsere Website unter http://www.altova.com/de/eula besuchen, um eine Kopie dieses Lizenzvertrags herunterzuladen und auszudrucken. Die Datenschutzbestimmungen können unter http://www.altova.com/de/privacy eingesehen werden.

1. SOFTWARELIZENZ

(a) Lizenzgewährung.

(i) Sofern Sie diesem Lizenzvertrag zugestimmt haben, gewährt Ihnen Altova eine nicht ausschließliche und nicht übertragbare (mit Ausnahme der nachfolgenden Bestimmungen) Lizenz, ohne das Recht zur Vergabe von Sublizenzen, zur Installation und Verwendung der Software auf einem kompatiblen Einzelplatzrechner oder einer Workstation, wobei die maximale Zahl an Computern, die im Lizenzumfang enthalten sind, nicht überschritten werden darf. Vorbehaltlich der Beschränkungen gemäß Abschnitt 1 (c), dürfen Sie eine Kopie der Software auf mehr als einem Ihrer kompatiblen Einzelplatzrechnern oder Workstations installieren und verwenden, wenn Sie eine Named-User-Lizenz (Lizenz zur benutzerdefinierten Nutzung) erworben haben. Vorbehaltlich der Beschränkungen gemäß den Abschnitten 1(d) und 1(e) darf die Software gleichzeitig im Netzwerk verwendet werden. Die maximale Zahl an Computern und/oder Benutzern sowie die Art der Lizenz, d.h. Einzelplatzlizenz, Named-User-Lizenz oder Parallellizenz, wird beim Kauf der Software festgelegt und genau angegeben. Einzelplatzlizenzen sind als fixe Lizenzen und nicht als Parallellizenzen zu verwenden, d.h. es ist nicht erlaubt, die Lizenz für einen Benutzer zu deinstallieren, um die Lizenz für einen anderen Benutzer zu installieren und diese Lizenz anschließend wieder zu deinstallieren und wieder für den ursprünglichen Benutzer zu

installieren. Einzelplatzlizenzen sollten statisch sein. Dessen ungeachtet sind permanente Uminstallationen einer Einzelplatzlizenz gestattet (wenn z.B. ein Mitarbeiter nicht mehr für das Unternehmen arbeitet und sein Rechner nicht mehr benötigt wird). Während des Evaluierungszeitraums (wie nachfolgend definiert) darf die Software nur von einem Benutzer und auf nur einem (1) Einzelplatzrechner oder einer Workstation installiert und genutzt werden. Wenn Sie die Lizenz im Rahmen eines Pakets von Altova-Softwareprodukten („Produktpaket“) erworben haben, von denen Sie nicht jedes Produkt einzeln installiert haben, gelten die Bestimmungen dieses Lizenzvertrags für die Verwendung der gesamten Software, die im Lieferumfang des Produktpakets enthalten ist.

(ii) Wenn Sie eine Lizenz für SchemaAgent erworben haben, dann gelten die Bestimmungen dieses Lizenzvertrags zur Nutzung der Software für den darin inkludierten SchemaAgent-Server („SchemaAgent-Server“). Dabei dürfen Sie den SchemaAgent-Server jedoch ausschließlich in Verbindung mit der Altova-Software und nur für die in der dazugehörigen Dokumentation beschriebenen Zwecke verwenden.

(iii) Wenn Sie eine Lizenz für Software erworben haben, die es dem Benutzer ermöglicht den Quellcode zu generieren, dann sind Sie durch die Ihnen im Rahmen dieser Lizenz erteilten Genehmigung zur Installation und Verwendung der Software auch berechtigt, den Quellcode zu generieren, und zwar entweder (i) anhand von Modulen der Altova Library, die in der Software inkludiert sind (der auf diese Weise generierte Code wird als „beschränkter Quellcode“ bezeichnet), oder (ii) anhand Ihrer eigenen Schemata oder Mappings (der anhand Ihres eigenen Quellmaterials generierte Code wird als „unbeschränkter Quellcode“ bezeichnet). Zusätzlich zu den durch diesen Lizenzvertrag eingeräumten Rechten gewährt Ihnen Altova eine nicht ausschließliche, nicht übertragbare und eingeschränkte Lizenz, den gesamten generierten Code (bestehend aus dem beschränkten und unbeschränkten Quellcode) in eine ausführbare Objektcodeform zu übersetzen und diese zu verwenden, zu kopieren, weiterzugeben oder zu lizenzieren. Sie sind nur dann befugt, den beschränkten Quellcode ab- oder weiterzugeben, weiterzulizenzieren, zu verkaufen oder an Dritte zu übertragen, wenn die betreffende Partei bereits in Form eines gesonderten Lizenzvertrags mit Altova zur Verwendung des beschränkten Quellcodes berechtigt ist. Ungeachtet anderslautender Bestimmungen in diesem Lizenzvertrag dürfen Sie die Altova-Library-Module oder den beschränkten Quellcode (oder sonstiges geistiges Eigentum von Altova enthalten in oder in Verbindung mit den Altova- Library-Modulen oder dem beschränkten Quellcode) nicht in solcher Art und Weise weitergeben, in andere Software einbetten, mit anderer Software verbinden oder auf andere Weise verwenden, die den beschränkten Quellcode den Bestimmungen des „Copylefts“, der freien Software oder einer Open Source Lizenz unterwerfen würde, welche die Offenlegung des beschränkten Quellcodes oder der Altova-Library- Module erforderlich machen würde. Altova behält sich alle übrigen Rechte an der Software vor. Bei den Bestandteilen von UModel, für die ein Reverse Engineering Ihres eigenen oder eines anderen Quellcodes, den Sie rechtmäßig erworben haben, zulässig ist, stellt eine solche Verwendung durch Sie keinen Verstoß gegen diesen Lizenzvertrag dar. Soweit dies nicht in Abschnitt 1(h) ausdrücklich gestattet wird, ist ein Reverse Engineering der Software explizit untersagt.

(iv) Falls beschränkter Quellcode in Form ausführbaren Objektcodes inkludiert wird, verpflichten Sie sich, folgende Informationen (1) in den Willkommensbildschirm, oder falls nicht vorhanden, in einen oder mehrere der durch den Endbenutzer jederzeit aufrufbaren Bildschirme und (2) - in elektronischer und/oder gedruckter Form in die Dokumentation zu inkludieren: "Teile dieses Programms wurden unter Verwendung von Altova® [Name der Altova Software, z.B. MapForce® 2011] entwickelt und enthalten Bibliotheken, die im Eigentum der Altova GmbH (Copyright © 2007-2011 Altova GmbH (www.altova.com) stehen."

(b) Verwendung auf einem Server zur Installation und Verwendung von SchemaAgent. Sie sind berechtigt, ein (1) Exemplar der Software auf einem Dateiserver ausschließlich innerhalb Ihres internen Netzwerks zu installieren und von dort auf die maximale Zahl von betrieblich genutzten Computern innerhalb dieses Netzwerks, die in der Lizenz angegeben ist, herunterzuladen und zu installieren. Wenn Sie eine Lizenz für SchemaAgent

erworben haben, können Sie SchemaAgent auf jedem Server oder jeder Workstation installieren und in Verbindung mit Ihrer Software verwenden. Jede sonstige Netzwerknutzung ist unzulässig; dies umfasst insbesondere die direkte Verwendung oder die Nutzung der Software über Befehle, Daten oder Anweisungen von oder an einen Computer, der nicht Teil Ihres internen Netzwerks ist, die Verwendung für Internet- oder Webhosting-Dienste sowie die Verwendung durch einen Benutzer, der nicht unter einer gültigen Lizenz von Altova zur Verwendung dieses Exemplars der Software berechtigt ist.

(c) Benutzerdefinierte Nutzung (Named-Use). Wenn Sie eine Lizenz der „benutzerdefinierten Version" der Software erworben haben, sind Sie berechtigt, die Software auf bis zu fünf (5) kompatiblen Einzelplatzrechnern oder Workstations zu installieren, auf denen Sie der primäre Nutzer sind. Dadurch können Sie von einem Computer zu einem anderen je nach Bedarf wechseln, vorausgesetzt, dass die Software von Ihnen als benutzerdefinierter Nutzer jeweils nur ein (1) Mal gleichzeitig verwendet wird. Wenn Sie mehrere benutzerdefinierte Lizenzen erworben haben, erhält jeder individuell definierte Nutzer einen gesonderten Lizenzschlüssel.

(d) Parallellizenz für die Verwendung im selben physischen Netzwerk oder am selben Unternehmensstandort. Wenn Sie eine Parallellizenz Software-Version erworben haben, können Sie die Software auf mehreren kompatiblen und betrieblich genutzten Computern installieren, jedoch höchstens zehnmal (10-mal) die maximale Zahl der Benutzer, vorausgesetzt, dass nur die maximale Zahl von Benutzern die Software wirklich gleichzeitig nutzen sowie weiters vorausgesetzt, dass die Computer, auf welchen die Software installiert ist, in dasselbe physische Computer Netzwerk eingebunden sind. Die maximale Zahl der Parallelbenutzer wird beim Kauf der Software festgelegt. Sofern die Parallelbenutzer Version der Software an mehr als einem Standort oder in mehr als einem Netzwerk verwendet werden soll, sind für jedes physisch getrennte Netzwerk sowie für jeden physisch getrennten Standort separate Parallelbenutzer Lizenzen erforderlich, wobei die Einschränkungen hinsichtlich der maximalen Zahl der Benutzer und der Anzahl der Nutzer, die die Software benutzen, anwendbar sind. Für den Fall, dass ein Computer nicht in dasselbe physische Netzwerk eingebunden ist, ist für diesen eine lokal installierte Benutzerlizenz oder eine Parallellizenz für die Verwendung in einer virtuellen Umgebung erforderlich. Be- und Einschränkungen hinsichtlich der Nutzung von Parallelbenutzer Lizenzen auf Heimcomputern sind in Abschnitt 1(g) geregelt.

(e) Parallellizenz für die Verwendung in einer virtuellen Umgebung. Wenn Sie eine Parallellizenz erworben haben, sind Sie berechtigt, ein Exemplar der Software auf einem Terminalserver (Microsoft Terminal Server, Citrix Metaframe, usw.), einem Anwendungsvirtualisierungsserver (Microsoft App-V, Citrix XenApp, VMWare ThinApp, usw.) oder einer virtuellen Maschine in Ihrem Intranet zu installieren, damit einzelne Benutzer in Ihrer Organisation von einem anderen Computer über einen Terminalserver, einen Anwendungsvirtualisierungsserver oder eine virtuelle Maschine auf die Software zugreifen und diese nutzen können. Jedoch darf die Zahl der Benutzer, die über einen solchen Netzwerk- oder Terminalserver bzw. eine virtuelle Maschine zum selben Zeitpunkt auf die Software zugreifen, die maximale Zahl der im Lizenzumfang enthaltenen User nicht überschreiten. Außerdem darf die Gesamtanzahl der Benutzer, die die Software über den Terminal Server, den Anwendungsvirtualisierungsserver oder eine virtuelle Maschine verwenden, die sechs (6)-fache Anzahl der lizenzierten Benutzer nicht übersteigen. Folglich gelten die in Abschnitt 1(d) festgelegten Einschränkungen betreffend die Anzahl der Installationen und die Voraussetzung, dass die Verwendung über dasselbe physische Netzwerk erfolgen muss, nicht für Terminal Server-, Anwendungsvirtualisierungsserver-Umgebungen oder Umgebungen auf virtullen Maschinen. Sie müssen in einer virtuellen Umgebung mit Hilfe geeigneter Maßnahmen sicherstellen, dass nicht mehr als die maximale Anzahl der in der Lizenz vorgesehenen Benutzer gleichzeitig auf die Software zugreifen. Altova gibt über die Performance der Altova-Software in einer Umgebung, in der Terminalserver, Anwendungsvirtualisierungserver oder virtuelle Maschinen verwendet werden, keinerlei Zusicherungen oder Gewährleistungen ab. Die Anwendung der Altova-Software auf die vorstehend beschriebene Weise ist

ausdrücklich von der eingeschränkten Garantie, die durch Abschnitt 5 dieses Lizenzvertrags eingeräumt wird, ausgeschlossen. Ebenso wird kein technischer Support bei Problemen geleistet, die aufgrund der Anwendung in einer solchen Umgebung auftreten.

(f) Kopien für Sicherungs- und Archivierungszwecke. (iii) Sie dürfen eine (1) Kopie der Software für Sicherungs- und eine Kopie für Archivierungszwecke herstellen, sofern diese nicht auf einem Computer installiert oder verwendet werden. Außerdem müssen bei Kopien zu Sicherungs- und Archivierungszwecken alle im Original enthaltenen urheber- und patentrechtlichen Angaben sowie alle anderen Angaben hinsichtlich geistiger Eigentumsrechte von Ihnen reproduziert werden und in der jeweiligen Kopie der Software unverändert enthalten sein. Die Befugnis zum Anlegen einer Sicherungs- und Archivierungskopie darf nur an Dritte übertragen werden, wenn gleichzeitig eine Übertragung aller Rechte an der Software gemäß Abschnitt 3 erfolgt.

(g) Nutzung auf dem Privat-PC (privater und nicht-kommerzieller Gebrauch). Um sich mit der Software, ihren Features und Funktion besser vertraut zu machen, sind Sie als vorrangiger Benutzer des Computers, auf dem die Software installiert wurde, berechtigt, ein weiteres Exemplar der Software zum persönlichen und nicht kommerziellen Gebrauch („pnk“) auf einem (1) einzigen Ihrer Heimcomputer (z.B. Ihrem Laptop oder PC) zu installieren. Dieses pnk-Exemplar darf zu keinerlei kommerziellen oder umsatzgenerierenden Geschäftsaktivitäten verwendet werden. Dazu gehören insbesondere die Arbeit von zu Hause aus, jedwede Form von Telearbeit oder andere Arbeit, die im Zusammenhang mit der Verwendung der Software steht. Das pnk-Exemplar der Software darf nicht gleichzeitig auf auf einem Privat-PC und dem eigentlich dafür vorgesehenen Primärcomputer verwendet werden.

(h) Freischaltcodes, Upgrades und Updates. Bevor Sie die Software kaufen bzw. im Rahmen Ihrer Registrierung für den Evaluierungszeitraum von dreißig (30) Tagen erhalten Sie einen Evaluierungs-Freischaltcode. Wenn Sie sich in der Folge dazu entscheiden, die Software von Altova GmbH oder einem autorisierten Vertriebshändler zu kaufen, erhalten Sie einen uneingeschränkten Freischaltcode. Mit diesem können Sie die Software nach dem Evaluierungszeitraum aktivieren. Ohne ausdrückliche schriftliche Genehmigung von Altova dürfen Sie an Freischaltcodes keine Lizenzen einräumen und Freischaltcodes nicht reproduzieren oder vertreiben. Wenn die von Ihnen in Lizenz erworbene Software ein Upgrade oder Update ist, ersetzt das Upgrade oder Update das sie zuletzt heruntergeladen oder installiert haben, die zuvor lizenzierte Kopie der Software. Durch das betreffende Upgrade oder Update und den dazugehörigen Freischaltcode wird keine zweite Lizenz für die Software gewährt, und Sie dürfen die Kopie des Upgrades oder Updates nicht zusätzlich zu der Kopie der Software verwenden, die dadurch ersetzt wird und dessen Lizenz beendet wurde.

(i) Eigentumsrecht. Das Eigentumsrecht an der Software wird Ihnen nicht übertragen. Alle Exemplare der Software und alle von Ihnen angefertigte Kopien verbleiben – vorbehaltlich der Ihnen durch diesen Lizenzvertrag eingeräumten Rechte – im Eigentum von Altova. Dokumente, Dateien, Stylesheets, generierter Programmcode (dies beinhaltet auch den unbeschränkten Quellcode) und Schemata, die Sie unter Verwendung der Software anlegen oder erstellen, sind in Übereinstimmung mit der dazugehörigen Dokumentation und den Bestimmungen dieses Lizenzvertrags Ihr Eigentum außer diese wurden mit einer Testsoftware gemäß Abschnitt 4 dieser Vereinbarung erstellt. In diesem Fall wird Ihnen nur eine beschränkte Lizenz zur Nutzung sämtlicher Ergebnisse, welche einen generierten Programmcode (inklusive eines unbeschränkten Quellcodes) enthalten, wie Java, C++, C#, VB.NET oder XSLT und verbundene Projektdateien und Build Scripts sowie generierte XML, XML Schematas, Dokumentationen, UML Diagrammen und Datenbankstrukturen, für die 30 Tage dauernde Testperiode eingeräumt **.**

(j) Reverse Engineering. Sofern dies nicht ausdrücklich nach geltendem Recht der Europäischen Union gestattet ist, sind Sie nicht berechtigt, an der Software Reverse Engineering durchzuführen, die Software zu dekompilieren, zu disassemblieren, oder auf andere Weise zu versuchen, den Quellcode, zugrunde liegende Konzepte, Techniken für

Benutzerschnittstellen oder Algorithmen der Software direkt oder indirekt zu entschlüsseln. Davon ausgenommen ist lediglich das ausdrücklich nach geltendem Recht der Europäischen Union gestattete Maß an Dekompilierung, das erforderlich ist, um die Funktionsfähigkeit der Software mit anderen Softwareprogrammen zu erzielen. Dies ist Ihnen jedoch erst dann gestattet, wenn Sie zuerst bei Altova die dafür notwendigen Informationen angefordert haben und Altova Ihnen diese nicht zur Verfügung gestellt hat. Altova ist berechtigt, für die Bereitstellung solcher Informationen angemessene Bedingungen aufzustellen und eine angemessene Entschädigung zu verlangen. Die von Altova zur Verfügung gestellten Informationen bzw. alle von Ihnen im Sinne der hier festgelegten Bestimmungen rechtmäßig erworbenen Informationen dürfen nur zu den hierin angeführten Zwecken verwendet und keinesfalls an Dritte weitergegeben oder zur Entwicklung von Software genutzt werden, die der Software, die Gegenstand dieses Lizenzvertrags ist, in wesentlichen Aspekten ähnlich ist. Informationen im Sinne dieses Absatzes können von Benutzern aus der Europäischen Union bei der Kundendienstabteilung von Altova angefordert werden.

(k) Sonstige Beschränkungen. Sie sind nicht berechtigt, die Software (zur Gänze oder teilweise) zu verleihen, zu vermieten, per Leasing zur Verfügung zu stellen, weiterzulizenzieren, weiterzugeben oder auf sonstige Weise Dritten zu überlassen, es sei denn, dies ist durch die Bestimmungen in Abschnitt 3 oder an anderer Stelle in diesem Lizenzvertrag ausdrücklich gestattet. Sofern dies nicht ausdrücklich in diesem Vertrag gestattet ist, dürfen Sie die Software nicht kopieren. Alle Kopien, zu deren Anfertigung Sie berechtigt sind, müssen die im Original enthaltenen urheber- und patentrechtlichen Angaben sowie alle anderen Angaben hinsichtlich geistiger Eigentumsrechte unverändert enthalten. Die Software darf nicht verändert, adaptiert oder übersetzt werden. Sie dürfen weder direkt noch indirekt ein Pfand- oder Sicherungsrecht an der Software bestellen oder die Bestellung eines solchen zulassen. Es ist Ihnen nicht gestattet, wissentlich eine Handlung vorzunehmen, durch die die Software öffentlich zugänglich wird, oder die Software in einer Computerumgebung zu verwenden, die nicht in diesem Lizenzvertrag angegeben ist.

Sie haben die geltenden gesetzlichen Bestimmungen und die Anweisungen von Altova in Bezug auf die Benutzung der Software einzuhalten. Sie sind verpflichtet, Ihre Mitarbeiter und Vertreter, die Zugriff auf die Software haben, von den in diesem Lizenzvertrag enthaltenen Beschränkungen in Kenntnis zu setzen und diese auf sie zu überbinden.

(l) ES WIRD KEINE GARANTIE ODER GEWÄHRLEISTUNG FÜR DIE FEHLERFREIHEIT DER SOFTWARE ABGEGEBEN ODER GEWÄHRT UND VON ALTOVA KEINERLEI HAFTUNG DIESBEZÜGLICH ÜBERNOMMEN. UNABHÄNGIG VON IRGENDWELCHEN SUPPORTLEISTUNGEN NACH IRGENDEINEM TECHNISCHEN STANDARD IST DIE SOFTWARE KEINESFALLS FÜR DIE VERWENDUNG IN ODER IM ZUSAMMENHANG MIT KERNKRAFTANLAGEN, FLUGGERÄTENAVIGATION, KOMMUNIKATIONSSYSTEMEN ODER LUFTVERKEHRSKONTROLLEINRICHTUNGEN, MEDIZINISCHEN GERÄTEN ODER LEBENSERHALTUNGSSYSTEMEN, MEDIZINISCHEN ODER DIE GESUNDHEITSVORSORGE BETREFFENDEN ANWENDUNGEN, ODER SONSTIGEN ANWENDUNGEN KONZIPIERT, BEI DENEN FEHLER DER SOFTWARE ODER FEHLER BEI DER DATENVERARBEITUNG ZU TODESFÄLLEN, PERSONENSCHÄDEN ODER SCHWEREN SACH- ODER UMWELTSCHÄDEN FÜHREN KÖNNTEN. SIE ERKLÄREN SICH DAMIT EINVERSTANDEN, DASS SIE DIE ALLEINIGE VERANTWORTUNG FÜR DIE EIGNUNG UND ZWECKMÄSSIGKEIT DER SOFTWARE UND ALLE MIT DER SOFTWARE ERSTELLTEN ODER VERARBEITETEN DATEN FÜR DEN BEABSICHTIGTEN VERWENDUNGSZWECK TRAGEN UND SIE WERDEN ALTOVA UND DEREN GESCHÄFTSFÜHRER SOWIE ANGESTELLTE HINSICHTLICH ALLER ANSPRÜCHE, FORDERUNGEN UND KLAGEN DRITTER, DIE AUF DIE EIGNUNG UND ZWECKMÄSSIGKEIT DER VON IHNEN VERWENDETEN SOFTWARE ODER DIE VON DER SOFTWARE ERSTELLTEN DATEN GEGRÜNDET SIND, SCHAD- UND KLAGLOS HALTEN.

2. GEISTIGE EIGENTUMSRECHTE

Anerkennung der Rechte von Altova. Die Software und alle Kopien, deren Anfertigung Ihnen von Altova gestattet ist, sind das geistige Eigentum von Altova und dessen Lieferanten und stehen in deren Besitz. Struktur, Organisation und Code der Software stellen wertvolle Betriebsgeheimnisse und vertrauliche Informationen von Altova und dessen Lieferanten dar. Die Software ist durch gesetzliche Bestimmungen urheberrechtlich geschützt. Diese gesetzlichen Bestimmungen beinhalten (ohne darauf beschränkt zu sein) das Urheberrecht der Vereinigten Staaten, internationale Verträge und das in den Ländern, in denen die Software genutzt wird, geltende Recht. Sie anerkennen, dass sich Altova das Eigentum an allen Patenten, Urheberrechten, Branchengeheimnissen, Warenzeichen und sonstigen geistigen Eigentumsrechten, die hinsichtlich der Software bestehen, vorbehält. Das Eigentumsrecht von Altova erstreckt sich auch auf alle Bilder, Fotografien, Animationen, Videos, Audioaufzeichnungen, Musikstücke, Texte und „Applets“, die Teil der Software sind, und alle dazugehörigen Unterlagen in gedruckter Form. Sie dürfen keine Handlungen vornehmen, die sich nachteilig auf die geistigen Eigentumsrechte von Altova an der Software auswirken. Warenzeichen dürfen nur in Übereinstimmung mit den anerkannten Standards für die Verwendung von Warenzeichen (einschließlich der namentlichen Nennung der Warenzeicheninhaber) verwendet werden. Die Verwendung von Warenzeichen ist nur zur Kennzeichnung von Druckmaterialien, die mit der Software hergestellt wurden, gestattet. Es entstehen Ihnen daraus keinerlei Eigentumsrechte an dem betreffenden Warenzeichen. XMLSpy®, Authentic®, StyleVision®, MapForce®, UModel®, DatabaseSpy®, DiffDog®, SchemaAgent®, SemanticWorks®, MissionKit®, Markup Your Mind®, Nanonull™ und Altova® sind (in mehreren Ländern eingetragene) Warenzeichen von Altova GmbH. Unicode und das Logo von Unicode sind Warenzeichen von Unicode, Inc. Windows, Windows XP, Windows Vista und Windows 7 sind Warenzeichen von Microsoft. W3C, CSS, DOM, MathML, RDF, XHTML, XML und XSL sind (in mehreren Ländern eingetragene) Warenzeichen des World Wide Web Consortium (W3C). Marken des W3C sind von den Dachinstitutionen des Konsortiums (MIT, INRIA und Keio) eingetragen und stehen in deren Eigentum. Sofern dies nicht ausdrücklich hierin festgelegt ist, entstehen Ihnen aus diesem Lizenzvertrag keinerlei geistigen Eigentumsrechte an der Software. Mitteilungen über geltend gemachte Verstöße gegen geistige Eigentumsrechte sind an den Copyright Agent von Altova zu richten (nähere Angaben dazu finden Sie auf der Website von Altova).

3. ÜBERTRAGUNGSBESCHRÄNKUNG

Ungeachtet der vorstehenden Bestimmungen dürfen Sie alle Ihre Rechte zur Verwendung der Software an eine andere natürliche oder juristische Person unter der Voraussetzung übertragen, dass: (a) Sie den vorliegenden Lizenzvertrag, die Software und sonstige Soft- oder Hardware, die mit der Software, die Gegenstand dieses Lizenzvertrags ist, geliefert, verpackt oder auf dieser vorinstalliert ist, einschließlich aller Kopien, Updates und früherer Versionen sowie aller Exemplare von Fontsoftware, die in andere Formate konvertiert wurde, an die betreffende Person übertragen; (b) Sie keine Kopien, einschließlich Sicherungskopien und sonstiger, auf einem Computer gespeicherter Kopien, zurückbehalten; (c) der Empfänger von Altova einen persönlichen Freischaltcode von Altova erhält; und (d) der Empfänger die Bestimmungen dieses Lizenzvertrags sowie alle anderen Bestimmungen akzeptiert, zu denen Sie die Softwarelizenz rechtmäßig erworben haben. Ungeachtet der vorstehenden Ausführungen dürfen Sie keine Schulungs-, Vorab- oder Musterkopien der Software an Dritte übertragen.

4. ZUSÄTZLICHE BEDINGUNGEN FÜR VORAB- UND TESTVERSIONEN

Wenn es sich bei dem von Ihnen mit dieser Lizenz erworbenen Produkt um eine unverkäufliche Vorabversion oder um Beta-Software („Pre-Release-Software“) handelt, gelten die in diesem Abschnitt enthaltenen Bedingungen. Diese gelten außerdem für alle Test- und/oder Demoversionen von Altova-Software („Testsoftware“) und bleiben so lange gültig, bis Sie eine Lizenz erwerben. Wenn Bestimmungen dieses Abschnitts im Widerspruch zu anderen, in

diesem Lizenzvertrag enthaltenen Bestimmungen stehen, dann gehen die Bedingungen dieses Abschnitts hinsichtlich Pre-Release-Software und Testsoftware allen anderen Bestimmungen dieses Lizenzvertrags vor, soweit dies für die Auflösung des Widerspruchs erforderlich ist. Sie erkennen an, dass es sich bei der Pre-Release-Software um eine Vorabversion handelt, die nicht das endgültige Produkt von Altova darstellt, und in der Fehler, Funktionsstörungen oder andere Probleme auftreten können, die möglicherweise zu einem System- oder Hardwareabsturz oder -fehler bzw. zu Datenverlust führen. DIE PRE-RELEASE-SOFTWARE UND/ODER TESTSOFTWARE WIRD IHNEN DAHER **OHNE GEWÄHR FÜR DEREN VERWENDUNG ODER LEISTUNGSMERKMALE** ZUR VERFÜGUNG GESTELLT, UND ALTOVA ÜBERNIMMT DIESBEZÜGLICH KEINERLEI AUSDRÜCKLICHEN ODER STILLSCHWEIGENDEN GEWÄHRLEISTUNGS- ODER HAFTUNGSVERPFLICHTUNGEN. SOFERN UND SOWEIT FÜR PRE-RELEASE- UND/ODER TESTSOFTWARE KEIN HAFTUNGSSAUSSCHLUSS, SONDERN NUR EINE HAFTUNGSBESCHRÄNKUNG GESETZLICH ZULÄSSIG IST, BESCHRÄNKT SICH DIE HAFTUNG VON ALTOVA UND DESSEN LIEFERANTEN AUF INSGESAMT FÜNFZIG US-DOLLAR (50 USD). Wenn die Testversion mit einer Time-out-Funktion ausgestattet ist, kann sie nach Ablauf des Evaluierungszeitraums nicht mehr verwendet werden. Mit dem Ablauf dieses Zeitraums endet Ihre Lizenz automatisch, wenn Sie sie nicht verlängern. Ihre Lizenz zur Nutzung sämtlicher mit der Testsoftware erstellten Ergebnisse, welche einen generierten Progammcode (inklusive den unbeschränkten Quellcode) wie Java, C++, C#, VB.NET oder XSLT und verbundene Projektdateien und Build Scripts sowie generierte XML, XML Schematas, Dokumentationen, UML Diagramme und Datenbankstrukturen enthalten, erlischt automatisch nach Ablauf der vorgesehenen Testperiode. Die Lizenz zum Gebrauch dieser Ergebnisse lebt nach dem Kauf einer Lizenz für die Software, die Sie getestet und mit der Sie das Ergebnis erstellt haben, wieder auf. Der Zugriff auf Dateien, die mit der Testsoftware erstellt wurden, erfolgt auf Ihr alleiniges Risiko. Sie bestätigen, dass Altova Ihnen weder zugesichert noch garantiert hat, die Pre-Release-Software zu einem späteren Zeitpunkt anderen Personen anzukündigen oder zur Verfügung zu stellen. Sie bestätigen ferner, dass Altova sich Ihnen gegenüber weder ausdrücklich noch stillschweigend verpflichtet hat, die Pre-Release-Software öffentlich anzukündigen oder einzuführen, und dass Altova kein Produkt anbieten muss, das der Pre-Release-Software ähnlich oder mit ihr kompatibel ist. Sie bestätigen folglich, dass jegliche Forschungs- oder Entwicklungsarbeit, die Sie in Bezug auf die Pre-Release-Software oder damit verbundene Produkte ausführen, in Ihr alleiniges Risiko fällt. Wenn Altova Sie dazu auffordert, werden Sie für die Dauer dieses Lizenzvertrags Altova Rückmeldung bezüglich Ihrer Tests und der Pre-Release-Software geben, einschließlich entsprechender Berichte über Fehler und Funktionsstörungen. Wenn Sie die Pre-Release-Software im Rahmen einer gesonderten schriftlichen Vereinbarung erhalten haben, unterliegt Ihre Verwendung der Software den darin enthaltenen Bedingungen. Es ist Ihnen nicht gestattet, die Pre-Release-Software unterzulizenzieren, per Leasing zur Verfügung zu stellen, zu verleihen, zu vermieten, weiterzugeben oder auf sonstige Weise Dritten zu überlassen. Sie verpflichten sich, bei Erhalt einer späteren unveröffentlichten Version der Pre-Release-Software bzw. bei Erhalt einer kommerziellen Vollversion von Altova (entweder als Einzelprodukt oder als Teil eines Gesamtprodukts) die zuvor von Altova erhaltene Pre-Release-Software zurückzugeben oder zu vernichten und den Bestimmungen des Lizenzvertrags für die betreffende spätere Version der Pre-Release-Software Folge zu leisten.

5. EINGESCHRÄNKTE GARANTIE UND HAFTUNGSBESCHRÄNKUNG

(a) Eingeschränkte Garantie und Garantieansprüche der Kunden. Altova garantiert der Person/Rechtspersönlichkeit, die ursprünglich die Lizenz für die Verwendung der Software gemäß den Bestimmungen dieses Lizenzvertrags erworben hat, dass (i) die Software über einen Zeitraum von neunzig (90) Tagen nach Erhalt im Wesentlichen in Übereinstimmung mit der dazugehörigen Dokumentation funktioniert, und (ii) die von Altova zur Verfügung gestellten Supportleistungen im Wesentlichen auf die in Abschnitt 6 dieses Vertrags niedergelegte Weise erfolgen. In einigen Ländern sind Beschränkungen über die Dauer einer stillschweigenden Garantie nicht zulässig, so dass die obigen Beschränkungen und Ausschlüsse eventuell für Sie nicht zutreffen. Im größtmöglichen rechtlich zulässigen Maß sind stillschweigende Garantien in

Bezug auf die Software (sofern solche existieren) auf neunzig (90) Tage beschränkt. Die gesamte Haftung von Altova und dessen Lieferanten, sowie Ihre einzigen Garantieansprüche, sind nach dem Ermessen von Altova auf eine der beiden folgenden Optionen beschränkt: (i) Erstattung des Kaufpreises (wenn zutreffend), oder (ii) Reparatur oder Austausch der Software, die unter die eingeschränkte Garantie von Altova fällt und die unter Vorlage einer Kopie des Kaufbelegs bei Altova reklamiert wird. Diese eingeschränkte Garantie gilt nicht, wenn die Funktionalität der Software durch ein Versehen, durch Missbrauch, falsche Anwendung, Trojaner, Viren oder einen sonstigen schädlichen externen Code beeinträchtigt wurde. Die Garantie für jede Ersatzsoftware erstreckt sich auf die Restdauer des ursprünglichen Garantiezeitraums oder auf dreißig (30) Tage, je nachdem, welcher Zeitraum länger ist. Die eingeschränkte Garantie gilt nicht für Test- und/oder Pre-Release-Software.

(b) Keine weiteren Garantien, Haftungsausschluss. MIT AUSNAHME DER VORSTEHEND ANGEFÜHRTEN EINGESCHRÄNKTEN GARANTIE UND DEN DIESBEZÜGLICHEN GARANTIEANSPRÜCHEN BESTEHEN SEITENS ALTOVA ODER DESSEN LIEFERANTEN KEINERLEI WEITEREN GARANTIEVERPFLICHTUNGEN. ALTOVA UND DESSEN LIEFERANTEN ÜBERNEHMEN KEINE GEWÄHR FÜR DIE ANWENDUNG ODER DIE ERGEBNISSE AUS DER NUTZUNG DER SOFTWARE. MIT AUSNAHME DER VORSTEHEND ANGEFÜHRTEN EINGESCHRÄNKTEN GARANTIE SOWIE IM HINBLICK AUF ALLE ANDEREN GEWÄHRLEISTUNGEN, BEDINGUNGEN, ZUSICHERUNGEN ODER ANSPRÜCHE, DIE NACH DER FÜR IHR LAND GELTENDEN RECHTSORDNUNG NICHT AUSGESCHLOSSEN ODER EINGESCHRÄNKT WERDEN KÖNNEN, SCHLIESSEN ALTOVA UND DESSEN LIEFERANTEN ALLE ANDEREN AUSDRÜCKLICHEN ODER STILLSCHWEIGENDEN GEWÄHRLEISTUNGEN, BEDINGUNGEN, ZUSICHERUNGEN UND ANSPRÜCHE AUS, DIE SICH AUS DEM GESETZ, DER RECHTSPRAXIS, EINEM GEWOHNHEITSRECHT, EINEM HANDELSBRAUCH ODER AUS SONSTIGEN GRÜNDEN ERGEBEN. ALTOVA UND DESSEN LIEFERANTEN SCHLIESSEN IM GRÖSSTMÖGLICHEN RECHTLICH ZULÄSSIGEN UMFANG ALLE ANDEREN AUSDRÜCKLICHEN UND STILLSCHWEIGENDEN GARANTIEN UND GEWÄHRLEISTUNGEN AUS. DIES BEINHALTET UNTER ANDEREM GARANTIEN IM HINBLICK AUF MARKTGÄNGIGKEIT, EIGNUNG FÜR EINEN BESTIMMTEN ZWECK, ZUFRIEDENSTELLENDE QUALITÄT, INFORMATIONSGEHALT ODER –RICHTIGKEIT, UNGESTÖRTE NUTZUNG, EIGENTUMSRECHT UND NICHTVERLETZUNG VON RECHTEN DRITTER UND DIE ERBRINGUNG ODER NICHTERBRINGUNG VON SUPPORTLEISTUNGEN IN BEZUG AUF DIE SOFTWARE. MIT DIESER EINGESCHRÄNKTEN GARANTIE WERDEN IHNEN BESTIMMTE RECHTE EINGERÄUMT. UNTER UMSTÄNDEN BESITZEN SIE NOCH ANDERE RECHTE; DIE JA NACH LAND/RECHTSORDNUNG UNTERSCHIEDLICH SEIN KÖNNEN.

(c) Haftungseinschränkung. SOWEIT DIE ANWENDBAREN GESETZE DIES ZULASSEN, HAFTEN ALTOVA ODER DESSEN LIEFERANTEN, AUCH WENN EINE IM RAHMEN DER GARANTIE DURCHGEFÜHRTE ABHILFEMASSNAHME IHREN WESENTLICHEN ZWECK NICHT ERFÜLLT, IN KEINEM FALL FÜR KONKRETE, ZUFÄLLIG ENTSTANDENE, MITTELBARE, UNMITTELBARE ODER FOLGESCHÄDEN JEGLICHER ART (INSBESONDERE SCHÄDEN AUS ENTGANGENEM GEWINN, GESCHÄFTSUNTERBRECHUNGEN, VERLUST VON GESCHÄFTSINFORMATIONEN ODER ANDEREN FINANZIELLEN VERLUSTEN), DIE DURCH DIE NUTZUNG ODER DIE UNMÖGLICHKEIT DER NUTZUNG DER SOFTWARE ODER DIE NICHTERBRINGUNG VON SUPPORTLEISTUNGEN ENTSTANDEN SIND, SELBST WENN ALTOVA AUF DIE MÖGLICHKEIT EINES SOLCHEN SCHADENS HINGEWIESEN WURDE. IN JEDEM FALL IST DIE GESAMTHAFTUNG VON ALTOVA GEMÄSS DIESEM LIZENZVERTRAG AUF DIE HÖHE DES FÜR DAS SOFTWAREPRODUKT ENTRICHTETEN BETRAGES BESCHRÄNKT. Da es in einigen Ländern und Rechtsordnungen nicht zulässig ist, die Haftung für Schäden auszuschließen oder zu beschränken, gilt die obige Einschränkung unter Umständen nicht für Sie. In solchen Ländern und Rechtsordnungen gilt die Beschränkung der Haftung durch Altova im größtmöglichen rechtlich zulässigen Umfang, und der Ausschluss bzw. die Beschränkung der in diesem Lizenzvertrag beinhalteten Garantien und Haftungen hat keine Auswirkung auf

die Konsumentenschutzrechte von Personen, die Waren auf andere Weise als im Zuge ihrer geschäftlichen Tätigkeit erwerben. Der vorstehende Haftungssauschluss/die vorstehende Haftungsbeschränkung sind wesentliche Bestandteile des zwischen Ihnen und Altova abgeschlossenen Lizenzvertrags.

(d) Ansprüche in Zusammenhang mit Urheberrechtsverletzungen. Altova wird Sie gegenüber allen Forderungen, Prozessen oder Verfahren schad- und klaglos halten bzw. alle Ansprüche, Prozesse oder Verfahren beilegen, die Dritte mit dem Argument gegen Sie erheben, dass der Inhalt der Software gegen ein Urheberrecht verstößt oder ein geistiges oder sonstiges Eigentumsrecht verletzt, das durch das Recht der Vereinigten Staaten oder der Europäischen Union geschützt ist (insgesamt als „Ansprüche" bezeichnet). Dies erfolgt – soweit nicht ausdrücklich etwas Anderes festgelegt ist – jedoch nur insoweit, als der betreffende Anspruch sich direkt aus der Verwendung der Software ergibt und nach Maßgabe der in Abschnitt 5 dieses Vertrags festgelegten Beschränkungen. Altova ist von jedem Anspruch innerhalb von zehn (10) Geschäftstagen, nachdem Sie erstmals davon benachrichtigt wurden, in Kenntnis zu setzen. Außerdem haben Sie mit Altova in angemessenem Umfang bei der Abwehr solcher Ansprüche zu kooperieren und Altova dabei zu unterstützen, ohne dass Sie dafür Kosten geltend machen können. Das Recht auf Entscheidungen in Bezug auf solche Ansprüche liegt allein bei Altova (dies beinhaltet auch, ohne darauf beschränkt zu sein, die Auswahl der Rechtsberater und das Recht, für Sie einen Vergleich zu den von Altova für zweckmäßig erachteten Bedingungen einzugehen). Sie können auf eigene Kosten einen Rechtsberater hinzuziehen und an den Verfahrens- oder Vergleichsverhandlungen teilnehmen. Altova kommt bis zu einer Höhe von insgesamt maximal dem Kaufpreis der Software für die Schäden, Kosten und Anwaltsgebühren auf, zu deren Bezahlung Sie in Zusammenhang mit solchen Ansprüchen verpflichtet werden (oder die Sie aufgrund eines Vergleichs zu entrichten haben), soweit diese nicht von einer Versicherung oder dritten Partei übernommen werden. Ist oder wird die Software Gegenstand von aufgrund einer Urheberrechtsverletzung vorgebrachten Ansprüchen, oder wird ihre Verwendung untersagt, oder ist es nach Ansicht des Rechtsberaters von Altova wahrscheinlich, dass ein solcher Umstand eintritt, so wird Altova versuchen, eine Beilegung herbeizuführen, indem alle wirtschaftlich vertretbaren Anstrengungen unternommen werden, um die Software zu modifizieren oder eine Lizenz für die weitere Verwendung der Software zu erwerben. Wenn es nach Ansicht des Rechtsberaters von Altova nicht möglich ist, den bevorstehenden oder bereits vorgebrachten Anspruch bzw. die Verfügung, mit der die Verwendung der Software untersagt wurde, durch angemessene Abänderung oder den Erwerb einer Lizenz beizulegen, so kann Altova diesen Lizenzvertrag ohne negative Konsequenzen für Altova beenden und Ihnen anteilig alle bereits an Altova entrichteten Gebühren zurückerstatten.
MIT AUSNAHME DER VORSTEHEND ANGEFÜHRTEN BESTIMMUNGEN OBLIEGEN ALTOVA KEINERLEI HAFTUNGSVERPFLICHTUNGEN FÜR ANSPRÜCHE IN ZUSAMMENHANG MIT URHEBERRECHTSVERLETZUNGEN. Diese Haftungsverpflichtung gilt nicht für Urheberrechtsverletzungen, die ausschließlich auf vom Kunden eingebrachte Elemente zurückzuführen sind.

6. SUPPORT UND WARTUNG

Zusätzlich zu der Version des Softwareprodukts, für das Sie eine Lizenz erhalten haben, bietet Altova eine Reihe von optionalen „Support- & Wartungspaketen" („*Software & Maintenance Package"*, „SMP") an, die Sie zusätzlich zu Ihrer Lizenz für die Software erwerben können. Der für das jeweilige SMP geltende Supportzeitraum (wie nachfolgend definiert) wird beim Kauf des SMP festgelegt. Die Support- und Wartungsservices und die Upgrades, die Ihnen zur Verfügung stehen, hängen davon ab, ob Sie ein SMP erwerben bzw. für welche Version des SMP Sie sich entscheiden.

(a) Wenn Sie kein SMP erwerben, erhalten Sie beim Kauf lediglich die Software, danach jedoch keinerlei Wartungsreleases oder Updates. Obwohl die Möglichkeit besteht, dass Altova Ihnen in Einzelfällen Wartungsreleases als kostenlose Zusatzleistung zur Verfügung stellt, sind in diesen Releases keine neuen Produktfeatures, die über das beim Kauf der Software bestehende Maß hinausgehen, beinhaltet. In jedem Fall erhalten Sie dreißig (30) Tage lang ab

Kauf der Software (der „Supportzeitraum" im Sinne dieses Absatzes 6(a)) kostenlosen technischen Support von Altova. Außerdem kann Altova in Einzelfällen auch während des dreißig (30) Tage dauernden Evaluierungszeitraums technischen Support als kostenlose Zusatzleistung zur Verfügung stellen. Technischen Support erhalten Sie ausschließlich über ein webbasiertes Supportformular, wobei es keine garantierte Reaktionszeit gibt.

(b) Mit dem Erwerb eines SMP haben Sie während des dafür geltenden Supportzeitraums **Anspruch auf die von Ihnen erworbene Produktversion** sowie auf alle Wartungsreleases und Updates für diese Produktversion, die während des für Sie geltenden Supportzeitraumsfreigegeben werden. Während des Supportzeitraums Ihres SMP erhalten Sie auch Upgrades auf die entsprechende Produktversion der Software, mit der auf eine höhere Version der Software, für die Sie die Lizenz erworben haben, gewechselt wird und die während Ihres Supportzeitraums freigegeben werden. Die einzelnen Upgradeeditionen, auf die Sie innerhalb Ihres Supportzeitraums Anspruch haben, sind in dem von Ihnen erworbenen SMP im Detail angeführt. Software, die als gesondertes Produkt eingeführt wird, ist nicht im SMP enthalten. Wartungsreleases, Updates und Upgrades können neue Features enthalten, dies muss aber nicht der Fall sein. Darüber hinaus erhalten Sie während des Supportzeitraums bevorzugten technischen Support von Altova, und zwar ausschließlich über ein webbasiertes Supportformular. Altova wird alle wirtschaftlich vertretbaren Anstrengungen unternehmen, um alle Anfragen per E-Mail innerhalb von achtundvierzig (48) Stunden während der Geschäftszeiten (Montag bis Freitag, 8.00 bis 22.00 Uhr UTC, ausgenommen Feiertage in Österreich und den USA) zu beantworten. Außerdem wird Altova sich in angemessenem Umfang darum bemühen, Workarounds für Fehler, die bei der Software aufgetreten sind, zur Verfügung zu stellen.

Während des Supportzeitraums können Sie Altova Fehler oder Defekte in der Software melden. Wenn nach Altovas Einschätzung ein reproduzierbarer schwerwiegender Fehler vorliegt, der die Verwendung und Funktionalität der Software erheblich beeinträchtigt, wird Altova wirtschaftlich vertretbaren Anstrengungen unternehmen um Korrekturen oder provisorische Lösungen anzubieten, die in zukünftigen Updates oder Wartungsreleases enthalten sind. Diese Updates oder Wartungsreleases werden von Altova von Zeit zu Zeit zur Verfügung gestellt.

Es steht im Ermessen von Altova, einen schriftlichen Nachweis über von Ihnen festgestellte Fehler oder Funktionsstörungen zu verlangen oder Beispieldateien anzufordern, aus denen das aufgetretene Softwareproblem hervorgeht. In einem solchen Fall müssen Sie die angeforderten Nachweise oder Dateien, aus denen ausreichend detailliert hervorgeht, in welchen Aspekten Fehler bei der Software auftreten, so schnell wie möglich per E-Mail, Fax oder Expresspost mit Zustellung am nächsten Tag an Altova übermitteln. Bei der Diagnose oder Analyse von Fehlern haben Sie in zumutbarem Rahmen mit Altova zu kooperieren. Fehlerbehebungen können in Wartungsreleases, Updates oder neuen Hauptversionen der Software enthalten sein. Altova ist nicht verpflichtet, unwesentliche Fehler, d.h. Fehler, die die Benutzung der Software nach dem Ermessen von Altova nicht wesentlich beeinträchtigen, zu beheben Sie erhalten von Altova unter keinen Umständen Beratung, Unterstützung oder Hilfestellung, die allgemeiner Natur ist und nicht in unmittelbarem Zusammenhang mit der Software steht.

Bei der Aktualisierung der Software kann es unter Umständen vorkommen, dass vor der Installation auch Software aktualisiert werden muss, die nicht unter diesen Lizenzvertrag fällt. Für Updates von Betriebssystem- und Anwendungssoftware, die nicht ausdrücklich Gegenstand dieses Lizenzvertrags ist, sind ausschließlich Sie verantwortlich. Solche Updates sind nicht im Umfang dieser Lizenz enthalten und werden nicht von Altova zur Verfügung gestellt. Die Erfüllung der in diesem Abschnitt 6 festgelegten Verpflichtungen durch Altova versteht sich unter der Bedingung, dass Sie die Software ordnungsgemäß verwenden und diesen Lizenzvertrag ausnahmslos jederzeit einhalten. Altova ist nicht verpflichtet, technischen Support zu leisten, wenn nach Ansicht von Altova die Fehlfunktion der Software auf einen der folgenden Gründe zurückzuführen ist: (i) Fehler, die durch die Verlegung der Software auf einen anderen Rechner oder Speicherort hervorgerufen werden; (ii) Änderungen, Modifikationen oder Versuche, die Software abzuwandeln, die von Altova nicht schriftlich

genehmigt wurden; (iii) äußere Einflüsse auf die Software, wie z.B. Naturkatastrophen, Stromausfälle, Stromschwankungen oder Computerausfälle; (iv) Ihr Versäumnis die Software auf dem von Altova festgelegten Release Level zu halten; oder (v) Nutzung der Software ohne vorherige Genehmigung durch Altova zusammen mit einer anderen Software. Sie alleine sind dafür verantwortlich: (i) allen Betriebs- und Fehlerbehebungsanleitungen von Altova Folge zu leisten, Altova unverzüglich von Fehlern oder Defekten an der Software zu informieren und Altova eine genaue Beschreibung dieser Fehler und/oder Defekte zu liefern; (ii) für den Schutz Ihrer vertraulichen Informationen zu sorgen; (iii) Datensicherungssysteme und -abläufe für die Wiederherstellung verlorener oder geänderter Dateien, Daten oder Programme einzurichten und anzuwenden.

7. AKTIVIERUNG DER SOFTWARE, UPDATES UND LIZENZÜBERWACHUNG

(a) Lizenzüberwachung. Altova verfügt über ein integriertes Lizenzüberwachungsmodul, das Ihnen hilft, unbeabsichtigte Verletzungen dieses Lizenzvertrags zu vermeiden. Dabei kann Altova zur Lizenzüberwachung von verschiedenen installierten Versionen der Software Ihr internes Netzwerk verwenden.

(b) Aktivierung der Software. **Die Software von Altova kann Ihr internes Netzwerk und Ihre Internetverbindung verwenden, um Angaben über Ihre Lizenz im Zuge der Installation, Registrierung, Benutzung oder Aktualisierung an einen Altova-Lizenzserver zu übertragen und diese zu verifizieren, um auf diese Weise eine nicht lizenzierte oder illegale Verwendung der Software zu verhindern und den Kundenservice von Altova weiter zu verbessern. Die Aktivierung erfolgt über einen Austausch von lizenzbezogenen Daten zwischen Ihrem Computer und dem Altova-Lizenzserver. Sie stimmen dieser Vorgangsweise von Altova zu und verpflichten sich, allen diesbezüglichen Vorgaben Folge zu leisten. Sie erklären sich damit einverstanden, dass die Verwendung von Freischaltcodes, welche nicht von Altova erstellt werden oder wurden und nicht rechtmäßig von Altova oder einem dazu berechtigten Wiederverkäufer erworben wurden, zum Zweck der Aktivierung oder Nutzung der Software die gewerblichen Schutzrechte von Altova sowie die Bestimmungen dieses Lizenzvertrages verletzt. Sie erklären sich weiters damit einverstanden, dass Versuche mit dem Zweck der Umgehung oder Deaktivierung der urheberrechtlichen Schutzmaßnahmen oder Lizenzmanagementsystemen von Altova die gewerblichen Schutzrechte von Altova sowie die Bestimmungen dieses Lizenzvertrages verletzen. Altova behält sich ausdrücklich das Recht vor, alle rechtlich verfügbaren sowie angemessenen Mittel zu ergreifen, um derartige Praktiken zu verhindern und entgangenen Gewinn, Schäden und Kosten zurückerstattet zu bekommen.**

(c) LiveUpdate. Altova stellt Ihnen ein neues LiveUpdate Benachrichtigungsservice zur Verfügung, welches kostenlos ist. Altova kann Ihr internes Netzwerk und Ihre Internetverbindung verwenden, um Angaben über Ihre Lizenz an einen LiveUpdate-Server von Altova zu übertragen, um Ihre Lizenz in gewissen Zeitabständen zu verifizieren und festzustellen, ob ein Update für Sie verfügbar ist.

(d) Verwendung von Daten. Der gesamte Wortlaut der Datenschutzbestimmungen von Altova kann unter http://www.altova.com/de/privacy eingesehen werden. Die Datenschutzbestimmungen sind durch Bezugnahme Teil dieses Lizenzvertrags. Durch Ihre Zustimmung zu den Bestimmungen dieses Lizenzvertrags bzw. durch Benutzung der Software erklären Sie sich damit einverstanden, dass Altova für die in diesem Lizenzvertrag und/oder in den Datenschutzbestimmungen (in der jeweils geltenden Fassung) genannten Zwecke Daten erhebt, verarbeitet und weitergibt. Altova behält sich das Recht vor, diese Bestimmung des Lizenzvertrags und/oder der Datenschutzbestimmungen jederzeit zu ändern. Wir legen Ihnen nahe, die auf der Website von Altova veröffentlichten Datenschutzbestimmungen von Zeit zu Zeit erneut durchzulesen.

(e) Hinweis für Benutzer in Europa. Bitte beachten Sie, dass die in Absatz 7(d) beschriebenen Informationen von Altova, Inc., einem Unternehmen mit Sitz in Beverly, Massachusetts, USA, oder seinen Tochterunternehmen, Zweigniederlassungen oder weltweit ansässigen autorisierten Partnern zum Zweck der Datenverarbeitung, Analyse und Überprüfung nach außerhalb der EU transferiert werden können. Sie werden darauf hingewiesen, dass in den USA ein Datenschutzmodell zur Anwendung kommt, das teils auf Gesetzen, teils auf Regierungsverordnungen und zum Teil auf Selbstregulierung beruht. Des Weiteren werden Sie darauf hingewiesen, dass der Rat der Europäischen Union befunden hat, dass dieses amerikanische Modell den in Artikel 25 der Datenrichtlinie der Europäischen Union (Richtlinie 95/46/EC, 1995 O.J. (L 281) 31) festgelegten Datenschutzbestimmungen nicht "in ausreichendem Ausmaß" Rechnung trägt. Laut Artikel 26 der Datenrichtlinie der Europäischen Union, dürfen persönliche Daten dann von der Europäischen Union in ein Drittland übertragen werden, wenn die jeweilige Person ihre Zustimmung zur Übertragung derartiger Informationen eindeutig gegeben hat, unabhängig davon, in welchem Ausmaß diese Daten in anderen Ländern geschützt sind. Durch Ihre Zustimmung zu dieser Software-Lizenzvereinbarung gestatten Sie die Übertragung aller derartiger Informationen an die USA sowie deren Verarbeitung gemäß dieser Software-Lizenzvereinbarung und Altovas Datenschutbestimmungen.

8. DAUER UND BEENDIGUNG

Dieser Lizenzvertrag wird folgendermaßen beendet: (a) indem Sie Altova eine schriftliche Benachrichtigung von der Vertragsbeendigung übermitteln; (b) durch Altova in Form einer diesbezüglichen schriftlichen Benachrichtigung, wenn Sie gegen diesen Lizenzvertrag verstoßen und diesen Verstoß nicht innerhalb von 10 (zehn) Tagen, nachdem Altova Ihnen dies mitgeteilt hat, berichtigen; oder (c) auf Verlangen eines autorisierten Vertriebshändlers von Altova, wenn Sie eine Lizenzzahlung nicht entrichten oder sonstige Zahlungsverpflichtungen nicht einhalten. Der Lizenzvertrag einer früheren Version der Software, für die Sie ein Upgrade oder ein Update durchgeführt haben, wird mit Ihrer Zustimmung zu dem Lizenzvertrag, der für das Upgrade oder Update gilt, automatisch beendet. Bei Beendigung des Lizenzvertrags dürfen Sie die gesamte Software, die Gegenstand dieser Lizenz ist, nicht mehr verwenden, müssen alle Kopien, die in Ihrem Besitz oder Einflussbereich stehen, vernichten, und müssen in zumutbarem Rahmen alle von Altova geforderten Maßnahmen ergreifen, um sicherzustellen, dass keine Kopien der Software in Ihrem Besitz oder Einflussbereich verbleiben. Die in den Abschnitten 1(h), 1(i), 1(j), 1(k), 2, 5(b), 5(c), 5(d), 7(d), 7(e), 9, 10 und 11 niedergelegten Bestimmungen bleiben auch nach Beendigung dieses Lizenzvertrags weiterhin aufrecht.

9. EINGESCHRÄNKTE RECHTE UND EXPORTBESCHRÄNKUNGEN

Die Entwicklung der Software wurde ausschließlich privat finanziert. Bei der Software handelt es sich um kommerzielle Computersoftware, die mit **eingeschränkten Rechten** ausgestattet ist. Die Verwendung, Vervielfältigung oder Weitergabe der Software durch die US-Regierung, eine Behörde oder einen Kooperationspartner der US-Regierung unterliegt den Beschränkungen im Rahmen dieses Vertrags sowie den Beschränkungen nach FAR 12.211 und 12.212 (48 C.F.R. §12.211 und 12.212) bzw. DFARS 227.7202 (48 C.F.R. §227-7202). Dabei wird eine Lizenz für kommerzielle Computersoftware und kommerzielle Computerdokumentation an die US-Regierung als Endnutzer ausschließlich in Form einer Lizenz für kommerziell genutzte Güter erteilt, weshalb es sich bei den damit verbundenen Rechten um dieselben Rechte handelt, die allen anderen Endnutzern im Rahmen dieses Lizenzvertrags gewährt werden. Beim Hersteller handelt es sich um Altova GmbH, Rudolfsplatz, 13a/9, A-1010 Wien, Österreich/EU. Die Software und/oder Dokumentation darf ausschließlich unter Einhaltung aller anwendbaren Exportbestimmungen der Vereinigten Staaten von Amerika sowie des Landes, in dem Sie die Software erhalten haben, verwendet, exportiert oder wiedereingeführt werden. Insbesondere darf die Software und/oder Dokumentation nicht (i) in ein Land exportiert oder wiedereingeführt werden, über das die Vereinigten Staaten ein Embargo verhängt haben, oder einem Staatsangehörigen oder Bewohner eines solchen Landes überlassen werden; oder (ii) einer Person überlassen werden, die auf der Liste der Specially Designated Nationals des U.S.

Treasury Department oder dem Table of Denial Orders des U.S. Department of Commerce verzeichnet sind. Indem Sie die Software benutzen, erklären Sie, dass Sie weder in einem dieser Länder ansässig sind noch seiner Kontrolle unterliegen, noch ein Staatsangehöriger oder Bewohner eines dieser Länder sind, noch auf einer der vorstehend erwähnten Listen genannt werden.

10. SOFTWARE DRITTER

Die dieser Lizenz unterliegende Software kann Software Dritter enthalten, für die ergänzende Vermerke und/oder Nutzungsbedingungen gelten. Diese Vermerke und ergänzenden Nutzungsbedingungen für die Software Dritter können über unsere Website unter http://www.altova.com/legal_3rdparty.html eingesehen werden und sind durch Bezugnahme Teil dieses Lizenzvertrags. Indem Sie Ihre Zustimmung zu den Bedingungen dieses Vertrags erteilen, stimmen Sie auch automatisch allen ergänzenden Bestimmungen, die möglicherweise darin enthalten sind, zu.

11. ALLGEMEINE BESTIMMUNGEN

Wenn Sie sich in der Europäischen Union befinden und die Software in der Europäischen Union und nicht in den Vereinigten Staaten verwenden, unterliegt dieser Lizenzvertrag dem Recht der Republik Österreich (unter Ausschluss von dessen Verweisungsnormen und der UN-Kaufrechtskonvention). Sie erklären sich ausdrücklich damit einverstanden, dass alle Streitigkeiten oder Konflikte mit Altova, die in Zusammenhang mit Ihrer Verwendung der Software stehen, in die alleinige Zuständigkeit des Handelsgerichts Wien fallen. Sie erklären sich weiters ausdrücklich damit einverstanden, dass Sie bezüglich solcher Streitigkeiten oder Konflikte der persönlichen Zuständigkeit des Handelsgerichts Wien unterstellt sind.

Wenn Sie sich in den Vereinigten Staaten befinden und die Software in den Vereinigten Staaten verwenden, unterliegt dieser Lizenzvertrag dem Recht des Commonwealth of Massachusetts, USA (unter Ausschluss von dessen Verweisungsnormen und der UN-Kaufrechtskonvention). Sie erklären sich ausdrücklich damit einverstanden, dass alle Streitigkeiten oder Konflikte mit Altova, die in Zusammenhang mit Ihrer Verwendung der Software stehen, in die alleinige Zuständigkeit der einzel- und bundesstaatlichen Gerichte im Bundesstaat Massachusetts fallen. Sie erklären sich weiters ausdrücklich damit einverstanden, dass Sie bezüglich solcher Streitigkeiten oder Konflikte der persönlichen Zuständigkeit der einzel- und bundesstaatlichen Gerichte im Bundesstaat Massachusetts unterstellt sind.

Wenn Sie sich nicht in der Europäischen Union oder den Vereinigten Staaten befinden und die Software nicht in den Vereinigten Staaten verwenden, unterliegt dieser Lizenzvertrag dem Recht der Republik Österreich (unter Ausschluss von dessen Verweisungsnormen und der UN-Kaufrechtskonvention). Sie erklären sich ausdrücklich damit einverstanden, dass alle Streitigkeiten oder Konflikte mit Altova, die in Zusammenhang mit Ihrer Verwendung der Software stehen, in die alleinige Zuständigkeit des Handelsgerichts Wien fallen. Sie erklären sich weiters ausdrücklich damit einverstanden, dass Sie bezüglich solcher Streitigkeiten oder Konflikte der persönlichen Zuständigkeit des Handelsgerichts Wien unterstellt sind. Die Anwendung von Verweisungsnormen einer Rechtsordnung sowie des UN-Abkommens zum internationalen Warenkauf (CISG) auf diesen Lizenzvertrag wird ausdrücklich ausgeschlossen.

Dieser Lizenzvertrag enthält die gesamte Vereinbarung zwischen den Parteien in Bezug auf den Vertragsgegenstand und tritt an die Stelle aller diesbezüglichen früheren mündlichen oder schriftlichen Vereinbarungen zwischen den Parteien. Benachrichtigungen oder sonstige Mitteilungen im Rahmen dieses Lizenzvertrags müssen schriftlich erfolgen und gelten als ordnungsgemäß übermittelt, wenn sie per Einschreiben mit Rückschein oder per Kurierdienst mit Zustellung am nächsten Tag an die auf der Website von Altova angegebene Adresse (wenn Altova der Empfänger ist) bzw. an die in den Aufzeichnungen von Altova für Sie eingetragene Adresse (wenn Sie der Empfänger sind) oder aber an eine andere, von den Vertragspartnern festgelegte und auf die vorstehend beschriebene Weise bekannt gegebene Adresse geschickt

werden. Dieser Lizenzvertrag ist für die Vertragspartner verbindlich und geht auf ihre Erben, persönlichen und rechtlichen Vertreter, verbundenen Unternehmen, Rechtsnachfolger und zulässigen Abtretungsempfänger über. Die Nichtdurchsetzung oder Nichtausübung von Rechten oder Rechtsmitteln unter diesem Lizenzvertrag durch die Vertragspartner stellt keinen Verzicht auf ein solches Recht oder Rechtsmittel dar und beeinträchtigt in keiner Weise das Recht des betreffenden Vertragspartners, ein solches Recht oder Rechtsmittel sowie alle anderen Rechte oder Rechtsmittel aus diesem Lizenzvertrag später durchzusetzen bzw. auszuüben. Eine Änderung dieses Lizenzvertrags ist nur Form einer schriftlich niedergelegten Vereinbarung möglich, die von beiden Vertragspartnern unterzeichnet wird. Wenn eine Zuwiderhandlung gegen die Bestimmungen dieses Lizenzvertrags durch einen der Vertragspartner erfolgt ist oder droht, so kann der jeweils andere Vertragspartner alle ihm zustehenden Rechte und Rechtsmittel geltend machen. Jeder Vertragspartner ist ordnungsgemäß befugt und ermächtigt, in diesen Lizenzvertrag einzutreten und die daraus erwachsenden Verpflichtungen zu erfüllen. Sollte eine Bestimmung dieses Lizenzvertrags für unwirksam, rechtswidrig oder undurchführbar erklärt werden, so wird dadurch die Wirksamkeit der übrigen Bestimmungen nicht berührt, und dieser Lizenzvertrag bleibt im größtmöglichen rechtlich zulässigen Umfang wirksam und gültig. Die Vertragspartner haben die vorstehenden Vertragsbedingungen zur Kenntnis genommen und erklären sich ausdrücklich damit einverstanden.

Letzte Aktualisierung: 2011-10-01

 © 2011 Altova GmbH

Index

(

(Standard),
mehrere Input- / Output-Komponenten, 54

.

.NET Erweiterungsfunktionen,
Datentypkonvertierungen, .NET in XPath/XQuery, 437
Datentypkonvertierungen, XPath/XQuery in .NET, 436
für XSLT und XQuery, 431
Instanzmethoden, Instanzfelder, 435
Konstruktoren, 433
statische Methoden, statische Felder, 434
Übersicht, 431

<

<dynamic>,
Datei-Input / Output, 54

A

abs, 366
Absolut,
Pfade - Vorteile/Nachteile, 172
Access,
MS Access-Unterstützung, 254
add, 343
Adjust-to-Timezone, 363
Aggregat,
Funktionen, 203
Aggregieren,
Funktion, 318
Aktion,
Symbol "Tabellenaktionen", 264
Aktualisieren,
Datenbankaktion, 264
Alles kopieren,
Konnektoren auflösen / löschen, 90
und Filter, 90
Verbindungen, 90
Altova Erweiterungen,
Diagrammfunktionen (siehe Diagrammfunktionen), 439
Altova Erweiterungsfunktionen,
allgemeine Funktionen, 440
Diagrammfunktionen (siehe Diagrammfunktionen), 440
Altova Website, 413
Altova XML Parser,
Info, 446
Altova XSLT 1.0-Prozessor,
Einschränkungen und implementierungsspezifisches Verhalten, 418
Altova XSLT 2.0-Prozessor,
allgemeine Informationen, 420
Informationen, 420
Altova-Prozessoren,
in Altova-Produkten, 447
Anführungszeichen,
und Befehlszeilenparameter, 213
Ansicht, 400
anyURI,
Funktionen, 360
Applikationsarbeitsablauf,
bei Verwendung globaler Ressourcen, 149
Arabisch, 449
Arbeitsablauf,
bei Verwendung globaler Ressourcen, 149
starten - globale Ressource, 153
Arbeitsspeicher-Anforderungen, 445
Arial Unicode MS, 448
ATTLIST,
DTD Namespace URIs, 231
Aufrufen,
Template, 312
Ausgabe,
benutzerdefiniert wenn bool = false, 306
Built-in Ausführungsprozessor, 120
Daten von Ausgabe speichern, 41
Fenster, 120
Fenster - Zoom-Faktor, 120
Parameter, 306
Pseudo-SQL, 256
Schemapfad zur Ausgabe hinzufügen, 386
validieren, 122

© 2011 Altova GmbH

Ausgabe,
XML validieren, 120
Ausgabe neu generieren, 256
Ausgabesprache, 398
Ausrichten,
Komponenten im Mapping-Fenster, 21
Ausrichtungshilfslinien,
Komponenten automatisch ausrichten, 68
Ausschneiden,
Parent/Child-Konnektoren, 95
Automatisch mappen,
Child-Elemente, 95
Automatisch verbinden,
Child-Elemente, 95
Sub-Einträge, 78
Automatische Ausrichtung,
Hilfslinien, 21
Komponenten im Mapping, 68
avg, 327

B

base-uri, 360
Basis,
Typ - derived types, 237
Bearbeiten, 384
Bedingung,
erweiterbares IF-Else, 385
Befehlszeile,
dynamische Input-Dateinamen, 164
Input-Parameter, 167, 213
Parameter, 211
Parameter und Eingabewerte, 213
Parameter und Vorschaueinstellungen, 215
Befehlszeilenparameter,
Wildcards in Anführungszeichen, 213
Beispiel,
rekursives benutzerdefiniertes Mapping, 239
Benutzerdefiniert,
Ausgabe wenn bool = false, 306
Bibliothek, 66, 117
Funkionen - Typ ändern, 277
Funktion, 66, 117
Funktion - Standard, 289
Funktionen, 277
Funktionen - komplexe, 294
importieren/exportieren, 277
komplexe Input-Komponenten, 295
lookup-Funktionen, 289
löschen, 277
verschachtelte Funktionen, 306
XSLT 2.0 Funktionen, 317
XSLT-Funktionen, 312
Benutzerdefinierte,
Funktion - Restrictions, 277
Funktionen & mffs, 277
Benutzerdefinierte Funktion,
inline, 287
rekursiv, 241
Benutzerdefinierten,
Funktionstyp ändern, 277
Benutzerhandbuch,
siehe auch Bildschirmhilfe, 410
Bestellen der Altova-Software, 411
Betriebssysteme für Altova-Produkte, 445
Bibliothek,
benutzerdefiniert, 66, 117
benutzerdefinierte Funktionen importieren, 277
definieren, 276
Funktion, 66, 117
Funktion suchen, 66, 117
Funktionsreferenz, 326
XPath2, 66, 117
XSLT 2.0-Funktionen hinzufügen, 317
XSLT-Funktionen hinzufügen, 312
Bibliotheken,
und benutzerdefinierte Funktionen, 277
Bildschirmhilfe,
Index, 410
Inhaltsverzeichnis, 410
Suchen, 410
BOM,
Byte Order Mark, 386
Bool,
Ausgabe wenn false, 306
Input-Nodes vergleichen, 206
boolean, 330
Builder,
benutzerdefinierte Funktion, 277
Built-in,
Ausführungsprozessor - Schaltfläche, 120
Bytefolge-Markierung,
in Komponenteneinstellungen, 386

© 2011 Altova GmbH

C

ceiling, 343
char-from-code, 349
Child-Datenelemente,
löschen, 95
Child-Elemente,
automatisch verbinden, 95
Code,
Inline-Funktionen & Codegröße, 287
Schemnamen daraus löschen, 386
code-from-char, 349
Codegenerierung,
Eingabeparameter, 213
Pfade absolut machen, 172
und absoluter Pfad, 37
und Eingabeparameter, 213
Version der Wrapper-Klasse, 402
von verketteten Mappings, 110
compare, 367
Compute when,
Variable, 184
concat, 349
constructors,
xs:ENTITY, 361
contains, 349
Context,
override, 222
Copyright-Informationen, 451
Core,
Bibliotheksfunktionen, 327
count, 327
count()-Funktion,
in XPath 1.0, 418
count, sum, avg,
Aggregatfunktion, 203
CSV,
Streaming, 120
current, 372
current-date, 362
current-dateTime, 362
current-time, 362

D

Date,
XSLT 2.0 Konstruktor, 317
Datei, 381
globale Ressourcen definieren, 138
Input aus mehreren Dateien / Output in mehrere Dateien, 54
Katalog, 233
Ressourcenkonfiguration hinzufügen, 138
Datei:,
(Standard), 54
Datenelement in Komponente, 54
Dateien,
dynamische Dateinamen als Input, 167
mehrere Dateien anhand einer Datenbank, 169
Daten,
filtern, 32
Daten beibehalten,
bei Verwendung von Wertezuordnung, 198
über Wertezuordnung übergeben, 198
Daten konsolidieren,
XML-Dateien zusammenführen, 209
Daten übergeben,
über Wertezuordnung unverändert übergeben, 198
Daten unverändert beibehalten,
über Wertezuordnung übergeben, 198
Datenbank,
DB ändern, 386
mappen auf, 255
mehrere Quellen, 386
Schemanamen aus Code entfernen, 386
Unterstützung, 254
XQuery, 254
Datenbankaktion,
Aktualisieren, 264
einfügen, 256
Datenbankunterstützung, 58
Datenelement,
duplizieren, 48
fehlendes, 99
Datenelemente,
Schema - Mapping, 25
Datentyp,
explizit - implizit, 317
Default,

© 2011 Altova GmbH

Default,
Eingabewert, 306
Default Functions Namespace,
in XSLT 2.0 Stylesheets, 420
default-collation, 362
Definitionsdatei,
globalresource.xml, 136
Derived,
types - verwenden / mappen auf, 237
distinct-values, 348
divide, 343
document, 372
document-uri, 360
DoTransform bat,
XML transformieren, 41
DTD,
Quelle und Ziel, 231
Duplizieren,
Eingabeelement, 48
Konnektor mittels Strg duplizieren, 78
Dynamisch,
Dateinamen als Input-Parameter, 167
Input-Dateien zur Laufzeit, 164
und Unterstützung mehrerer Dateien, 162

E

EDI,
validieren, 122
Eigenschaften,
Wertezuordnungstabelle, 201
Eine einzige Zieldatei,
mehrere Quellen, 209
Einfügen, 385
Datenbankaktion, 256
Einführung in MapForce, 3
Eingabemethoden, 448
Eingabeparameter,
Befehlszeile, 213
und Codegenerierung, 213
Einmal berechnen,
Variable, 184
Einstellungen,
Komponente ändern, 386
Element,
in Zielelement konvertieren, 386
rekursives Element in XML-Schema, 239
element-available, 372
Endbenutzer-Lizenzvereinbarung, 451, 455
ends-with, 367
Entfernen,
"Alles kopieren"-Verbindungen, 90
Schemanamen, 386
Verbindung, 95
equal, 341
equal-or-greater, 341
equal-or-less, 341
Ersetzen,
fehlenden Node, 218
Erstellen,
benutzerdefinierte Funktion, 277
Funktion, 28
Erweitern,
Funktionsparameter, 66, 117
Erweiterungsfunktionen für XSLT und XQuery, 423
Erweiterungsfunktionen in .NET für XSLT und XQuery,
siehe .NET Erweiterungsfunktionen, 431
Erweiterungsfunktionen in Java für XSLT und XQuery,
siehe Java-Erweiterungsfunktionen, 423
Erweiterungsfunktionen in MSXSL Scripts, 437
escape-uri, 367
Evaluierungs-Keycode,
für Ihre Altova-Software, 411
Evaluierungszeitraum,
bei Altova Software-Produkten, 454
für Altova-Software-Produkte, 452
von Altova Software-Produkten, 451
Examples,
Tutorial-Ordner, 18
Exist,
Mappen fehlender Nodes mittels not-exist, 220
exists, 345
Node-Test, 218
Explizit,
Datentyp, 317
Exportieren,
benutzerdefinierte Funktion, 277
Expression,
regular, 356
Extras, 402

© 2011 Altova GmbH

F

false, 361
Fehlende Datenelemente, 99
Fehlende Nodes,
Mappen fehlender Nodes, 220
Fehler,
Validierung, 122
Filter,
"Alles kopieren"-Konnektor, 90
Daten, 32
Komplement, 32
Komponente - Tipps, 192
Parent-Datenelemente mappen, 192
Prioritätskontext, 192
verketten - vermeiden, 192
Filtern,
Zusammenführen von XML-Dateien, 209
FLF,
Streaming, 120
floor, 343
format-date-time, 330
format-number, 330, 372
Fragen und Antworten zu MapForce, 413
Fragezeichen,
fehlende Datenelemente, 99
From, 363
function-available, 372
Funktion, 395
Aggregatfunktion, 203
benutzerdefiniert, 277
benutzerdefinierte, 66, 117
benutzerdefinierte - Typ ändern, 277
benutzerdefinierte exportieren, 277
benutzerdefinierte Funktion, 241
benutzerdefinierte Standardfunktion, 289
benutzerdefinierte XSLT 2.0-Funktion hinzufügen, 317
benutzerdefinierte XSLT-Funktion hinzufügen, 312
Bibliothek, 66, 117
core Bibliothek, 327
definieren, 276
erweiterbare, 66, 117
erweiterbares IF-Else, 385
hinzufügen, 66, 117
in Bibliothek suchen, 66, 117
inline, 287
Input als Parameter, 213
komplexe - inline, 287
Konvertierung - Boolesche, 206
lookup-Funktion, 289
Restrictions in benutzerdefinierter, 277
summieren, 318
Typ von benutzerdefinierter Funktion ändern, 277
verschachtelte benutzerdefinierte, 306
visueller Function Builder, 277
xpath2 Bibliothek, 360
XQuery, 66, 117
xslt-Bibliothek, 370
Funktionen,
benutzerdefinierte importieren, 277
mappen auf, 28
Referenzteil, 326
siehe unter XSLT 2.0-Funktionen, 422

G

generate-id, 372
Generieren,
Code & Inline-Funktionen, 287
mehrere Java-Ziele, 41
mehrere XSLT-Ziele, 41
XML-Schema automatisch generieren, 20
get-fileext, 339
get-folder, 339
Global, 448
Globale Ressource,
aktivieren, 143
Arbeitsablauf, 149
Arbeitsablauf starten, 153
Eigenschaften, 156
einer Komponente zuweisen, 141
Konfiguration kopieren, 138
Ordner als, 145
Standardkonfiguration, 138
Globale Ressourcen,
Definitionsdatei, 136
Ressourcendatei definieren, 138
Symbolleiste, 137
Globalresource.xml,
Ressourcendefinition, 136
greater, 341

© 2011 Altova GmbH

Größe anpassen,
Komponente an optimale Größe, 20
group-adjacent, 348
group-by, 348
group-ending-with, 348
group-starting-with, 348
Gruppen,
Schleifen und Hierarchien, 125

H

Health Level 7,
Beispiel, 274
Hebräisch, 449
Hierarchien,
Schleifen und Gruppen, 125
Hilfe, 409
siehe Bildschirmhilfe, 410
Hintergrundinformationen, 444
Hinzufügen,
benutzerdefinierte Funktionen, 277
globale Ressourcendatei, 138
Schemapfad, 386
Hiragana, 448
HL7 2.6 auf 3.x,
Beispiel, 274
HTML,
Register im Mapping-Fenster, 64

I

IF-Else,
erweiterbar, 385
Implementierungsspezifisches Verhalten,
von XSLT 2.0.Funktionen, 422
Implizit,
Datentyp, 317
Importieren,
benutzerdefinierte Funktionen, 277
Include,
XSLT, 312
XSLT 2.0, 317
in-context,
Parameter, 327
Inline, 277
benutzerdefinierte Funktionen, 287
Funktionen und Codegröße, 287
Input,
als Befehlszeilenparameter, 213
Befehlszeilenparameter, 215
Boolesche Nodes vergleichen, 206
mehrere Dateien, 164
optionale Parameter, 306
Standardwert, 306
XML-Instanz, 386
Input-Komponente,
Standardwert, 215
Input-Parameter,
dynamisch, 164
und dynamische Dateinamen, 167
Input-Symbol,
Mapping, 78
Input-Verarbeitung aktivieren,
Optimierung, 386
Installation,
Ordner examples, 18
Instanz,
XML-Input-Instanz, 386
XML-Output-Instanz, 386
Internet-Verwendung,
in Altova-Produkten, 450
ISO/IEC 10646, 448
is-xsi-nil, 345
Iteration,
Prioritätskontext, 208

J

Java,
mehrere Ziele, 37
mehrere Ziele generieren, 41
Java Erweiterungsfunktionen,
Datentypkonvertierungen, Java in Xpath/XQuery, 431
Instanzmethoden, Instanzfelder, 429
Konstruktoren, 428
statische Methoden, statische Felder, 428
Java extension functions,
datatype conversions, XPath/XQuery to Java, 430
Java-Erweiterungsfunktionen,
benutzerdefinierte JAR-Dateien, 427

© 2011 Altova GmbH

Java-Erweiterungsfunktionen,
benutzerdefinierte Klassendateien, 424
für XSLT und XQuery, 423
Übersicht, 423

K

Kanji, 448
Katakana, 448
Katalogdateien, 233
Keycodes,
für Ihre Altova-Software, 411
Kodierung,
Bytefolge-Markierung, 386
Komponenteneinstellungen, 386
Komplement,
Komplement Nodeset, 32
Komplex,
benutzerdefinierte Funktion, 294
komplexe benutzerdefinierte Input-Komponenten, 295
Komplexe,
Funktion - inline, 287
Komponente,
Datenbank, 58
Datenbank ändern, 386
Einstellungen ändern, 386
gelöschte Datenelemente, 99
globale Ressource zuweisen, 141
Input - Standardwert, 215
Input aus mehreren Dateien / Output in mehrere Dateien, 53
Input-Verarbeitung aktivieren, 386
Kodierungseinstellungen, 386
mehrere Input- / Output-Komponenten, 164
mehrere Inputs / Outputs, 162
optimale Größe, 20
Pretty-Print-Ausgabe, 386
Komponenten,
Input aus mehreren Dateien / Output in mehrere Dateien, 54
Verarbeitungsreihenfolge, 132
Komponenten-Download-Center,
auf der Altova Website, 413
Konfiguration,
bestehende kopieren, 138
wechseln - globale Ressource, 143
zu globaler Ressource hinzufügen, 138
Konnektor,
Eigenschaften, 95
Mappen mit, 78
mit Strg kopieren, 78
Popup, 78
untergeordnete Verbindungen inkludieren, 95
Konnektoren,
Alles kopieren, 90
Konnektor-Symbol,
Popup, 78
Konstante,
als Standardwert, 215
Konstruktor,
XSLT2, 317
Konstruktoren,
xs:ENTITY, 361
Kontext,
Priorität, 66, 117
Prioritätskontext, 208
Konvertieren,
in Zielschema, 386
Konvertierung,
Funktionen - Boolesche, 206
Typüberprüfung, 232
Kopieren,
bestehenden Konnektor mit Strg an eine andere Stelle kopieren, 78

L

lang, 365, 370
last, 362, 370
last()-Funktion,
in XPath 1.0, 418
less, 341
Lizenz, 455
für Ihre Altova-Software, 411
Informationen, 451
Lizenzüberwachung,
bei Altova-Produkten, 453
local-name, 365, 370
logical-and, 341
logical-not, 341
logical-or, 341
Lokale,
Schemas - Katalogdateien, 233
Lookup-Tabelle,

© 2011 Altova GmbH

Lookup-Tabelle,
Eigenschaften, 201
Mappen fehlender Nodes, 220
Wertezuordnungstabelle, 194
Löschen,
"Alles kopieren" Verbindungen, 90
benutzerdefinierte Funktionen, 277
Löschung - fehlende Datenelemente, 99
Verbindungen, 95
lower-case, 367

M

main-mfd-filepath, 339
manespace-uri, 365
Mapforce,
Einführung, 3
Prozessor, 120
Terminologie, 12
Übersicht, 10
MapForce-Prozessor,
Built-in Ausführungsprozessor, 120
Mappen,
Daten auf Datenbanken, 255
Mapping,
Anzahl der Verbindungen, 78
Child-Elemente, 95
Eigenschaften, 95
Fenster - automatische Ausrichtung, 21
Konnektor, 78
quellorientiert - Mixed Content, 82
Schemaelemente, 25
Standard-Mapping, 88
Struktur validieren, 122
Tutorial, 18
typorientiert, 90
Validierung, 122
Verarbeitungsreihenfolge, 132
zieloriertiert, 88
Mapping methods,
standard, 81
target-driven, 81
Mapping-Fenster,
automatische Ausrichtung, 68
Stylevision-Register, 64
Markierte Datenelemente,
fehlende Datenelemente, 99
matches, 367
max, 327
Mehrere,
mehrere Zielschemas anzeigen, 41
Quellschemas, 386
Tabellenaktionen, 264
Ziele und Java, 37
Zielschemas, 37
Mehrere Dateien,
Input / Output, 164
Input aus / Output in, 53
verarbeiten (Tutorial), 54
Mehrere Inputs,
Elemente, 48
Mehrere Quellen,
auf ein einziges Zielelement, 43
in einer einzigen Zieldatei zusammenführen, 209
Mehrere XML-Dateien,
anhand einer einzigen XML-Quelle, 169
Menü,
Ansicht, 400
Ausgabesprache, 398
Bearbeiten, 384
Datei, 381
Einfügen, 385
Extras, 402
Funktion, 395
Hilfe, 409
Verbindung, 390
mfd-filepath, 339
MFF,
und benutzerdefinierte Funktionen, 277
min, 327
min, max,
Aggregatfunktion, 203
minOccurs/maxOccurs,
Optimierung der Input-Verarbeitung, 386
Mixed,
Content mappen, 82
quellorientiertes Mapping, 82
Standard-Mapping, 88
modulus, 343
MS Access,
Unterstützung v 2000 und 2003, 254
MS Access 2000,
Unterstützung, 58
MS SQL Server,

© 2011 Altova GmbH

MS SQL Server,
Unterstützung, 58
MSXML 6.0,
Bibliothek, 402
msxsl:Script, 437
Multi,
Dateiunterstützung - Sprachen, 162
Input / Output, 162
multiply, 343
MyDokuments,
Ordner examples, 18

N

name, 365, 370
Named,
Template - Namespaces, 312
Named Template,
Nodes summieren, 318
Namespace,
named Template, 312
Namespace URI,
DTD, 231
Namespaces,
im XSLT 2.0 Stylesheet, 420
namespace-uri, 370
Nicht lateinischen Schriftsysteme, 448
node, 365
Boolesche vergleichen, 206
Mappen fehlender Nodes, 220
mehrere summieren, 318
Position, 222
testen, 218
node-name, 360
Nodeset,
Komplement, 32
normalize-space, 349
normalize-unicode, 367
Not exist,
Mappen fehlender Nodes, 220
not-equal, 341
not-exists, 345
Null,
nillable, 250
xsi:nil, 250
number, 330, 365

O

Optimale Größe,
Doppelklick auf die Schaltfläche "Größe anpassen", 20
Optimierung,
Input-Verarbeitung aktivieren, 386
Optional,
Eingabeparameter, 306
Oracle,
Unterstützung, 58
Ordner,
als globale Ressource, 145
Output,
mehrere Dateien, 164
XML-Instanz, 386
Output-Symbol,
Mapping, 78
Override,
context, 222

P

Parameter,
Ausgabe, 306
Befehlszeile, 211, 213
in Funktionen erweitern, 66, 117
in-context, 327
Input - dynamisch, 164
Input-Funktion, 213, 215
mittels Wildcards, 213
optionale, 306
Standardwert, 215
und Codegenerierung, 213
Parent,
Mappen und Filter, 192
Parent-Kontext,
Variable, 184
parse-dateTime, 330
parse-number, 330
Parser,
in Altova-Produkte integrierter, 446
Partner-Software, 413
Pfad,

© 2011 Altova GmbH

Pfad,
absoluter bei Generierung von Code, 37
Pfade,
im generierten Code absolut machen, 172
Plattformen für Altova-Produkte, 445
position, 345, 362, 370
node / Kontext, 222
position() function,
in XPath 1.0, 418
Pretty Print,
in Ausgabekomponente, 386
Priorität,
Funktion, 66, 117
und Filter, 192
Prioritätskontext,
definieren, 208
Prozessor,
Mapforce, 120
Prozessoren,
zum Downloaden, 413

Q

qname-related,
Funktionen, 367
Quelldatei,
in mehrere Zieldateien aufteilen, 169
Quellorientiertes Mapping,
Mixed Content Mapping, 82
Quellorientiertes Mapping im Vgl. zu Standard-Mapping, 88

R

Raster,
Komponenten ausrichten, 68
Rechtliches, 451
Referenz, 380
Funktionen in MapForce, 326
Regex, 356
Register,
Stylevision-Ausgaberegister, 73
Registrierung der Altova-Software, 411
Regular expressions, 356
Reihenfolge,
der Verarbeitung von Komponenten, 132
Rekursiv,
Aufrufe in Funktionen, 287
benutzerdefinierte Funktion, 241
benutzerdefiniertes Mapping, 239
Relativ,
Pfade - Vorteile, 172
remove-fileext, 339
remove-folder, 339
replace, 367
replace-fileext, 339
resolve-filepath, 339
resolve-uri, 360
Ressource,
Eigenschaften einer globalen Ressource, 156
Ordner, 145
Rest einfügen,
Tabellenaktion, 264
Root,
Element des Ziels, 205
Root-Element,
neues erstellen, 95
round, 343
round-half-to-even, 366
round-precision, 343
RTF,
Register im Mapping-Fenster, 64
Rückwärtskompatibilität,
des XSLT 2.0-Prozessors, 420

S

Schema,
Anzeigen mehrerer Ziele, 41
automatisch von XML-Datei generieren, 20
Datenbank als Quelle, 58
Katalogdateien, 233
mehrere Quellen, 386
mehrere Ziele, 37
Name, aus generiertem Code löschen, 386
Pfad zur Ausgabe hinzufügen, 386
rekursive Elemente, 239
Schemanamen,
aus Tabellennamen entfernen, 386
Schleifen,
Gruppen und Hierarchien, 125

© 2011 Altova GmbH

Schriftart, 448
Schriftsystem, 448
Schriftsysteme, 449
Scripts in XSLT/XQuery,
siehe Erweiterungsfunktionen, 423
Sequenz,
Postition, 222
set-empty, 348
set-xsi-nil, 345
Shortcut,
Tastatur, 402
Software-Produktlizenz, 455
Speichern,
Daten im Ausgabefenster, 41
XML-Ausgabe, 120
Sprachen,
und dynamische Unterstützung mehrerer, 162
SQL,
Ausgabe neu generieren, 256
Befehle im Ausgabe-Fenster, 256
Pseudo-SQL im Fenster Ausgabe-Vorschau, 256
SQL-Skript erneut ausführen,
Ausgabe neu generieren, 256
Standard,
benutzerdefinierte Funktion, 289
Konfiguration - Globale Ressource, 138
mapping method, 81
Mixed Content Mapping, 88
Parameter - Input-Komponente, 215
Standardbibliothek,
XSLT-Bibliothek, 66, 117
Standard-Mapping im Gegensatz zu quellorientiertem Mapping, 88
starts-with, 349, 367
Streaming,
XML, CSV und FLF, 120
string, 330, 360
string-join, 327
string-length, 349
Stylesheet,
Stylevision Power Stylesheet Register, 73
Stylevision,
Mapping-Ausgaberegister, 73
Register im Mapping-Fenster, 64
verkettetes Mapping - Endkomponente, 105, 110
Sub-Einträge,
automatisch verbinden, 78
substitute-missing, 345
substitute-missing-with-xsi-nil, 345
substring, 349
substring-after, 349, 367
substring-before, 349, 367
Subtract, 343, 363
Suchen,
Funktion in Bibliothek, 66, 117
XSLT - Register Ausgabevorschau, 384
sum, 327
Summieren,
Nodes in XSLT 1.0, 318
Support für <MapForce, 413
Symbolleiste,
globale Ressource, 137
system-property, 372

T

Tabelle,
Lookup - Wertezuordnung, 194
Schemanamen entfernen, 386
Tabellenaktionen - mehrere, 264
Tabellen,
Symbol "Tabellenaktionen", 264
Tabellenaktion,
Rest einfügen, 264
Target-driven mapping, 81
Tastatur,
Kürzel, 402
Tastaturkürzel,
Zoom-Faktor Fenster "Ausgabe", 120
Tastenkürzel,
Shortcuts, 402
Technische Informationen, 444
Technischer Support für MapForce, 413
Teminologie, 12
Template,
aufrufen, 312
named - summieren, 318
Test,
Node-Test, 218
Tipps, 188
Tokenize, 349, 353
Tokenize-by-length, 349, 353
tokenize-regexp, 349
Transformationsergebnis,

© 2011 Altova GmbH

Transformationsergebnis,
globale Ressourcen, 149
Transformieren,
DoTransform.bat, 41
Input-Daten - Wertezuordnung, 194
translate, 349
Treiber,
JDBC, 386
true, 361
Tutorial, 18
Ordner examples, 18
Typ,
in Zieltyp konvertieren, 386
Typen,
derived types - xsi:type, 237
Typkonvertierung,
überprüfen, 232
Typorientiert,
Verbindungen, 90

U

Über MapForce, 413
Überprüfen,
Typ überprüfen, 232
Übersicht über MapForce, 10
UCS-2, 448
UCS-4, 448
Unicode,
Konsortium, 448
Standard, 448
Unterstützung, 448
Unicode-Zeichen, 448
Universalschriftart, 448
unparsed-entity-uri, 372
Untergeordnete,
Verbindungen - inkludieren, 95
Unterstützung,
Datenbanken, 254
upper-case, 367
URI,
in DTDs, 231
UTF-8, 448

V

Validator,
in Altova-Produkten, 446
Validieren,
Daten im Ausgabefenster, 41
Mapping-Projekt, 122
Output-Daten, 122
XML, 120
Verarbeitungsreihenfolge,
von Komponenten in einem Mapping, 132
Verbindung, 390
Eigenschaften, 95
löschen, 95
Parent/Child-Konnektoren verschieben, 95
Verbindungen,
typorientiert, 90
Verketten,
Filter - vermeiden, 192
Verkettetes,
Mapping - Codegenerierung, 110
Verkettetes Mapping,
Endkomponente mittels StyleVision anzeigen, 105, 110
Verschachtelt,
benutzerdefinierte Funktionen, 306
Verschieben,
Parent/Child-Konnektoren, 95
Version,
Kompatibilität der Wrapper Klassen, 402
Vertrieb,
von Altova Produken, 454
von Altova Software-Produkten, 451
von Altova-Software-Produkten, 452
Verwenden,
Globale Ressourcen, 143
Visual Studio,
unterstützte Versionen - Codegenerierung, 402
Visueller Function Builder, 277
Von rechts nach links, 449
Vorschau,
Mapforce-Prozessor, 120
Wert der Input-Komponente, 215

© 2011 Altova GmbH

W

Warnung,
Validierung, 122
Wechseln,
Konfiguration - globale Ressource, 143
Wert,
default, 306
Input-Parameter - Vorschau, 215
Wert definieren,
Input-Komponente - Vorschau, 215
Wertezuordnung,
Daten unverändert übergeben, 198
Lookup-Tabelle, 194
Lookup-Tabelle - Eigenschaften, 201
Whitespace im XML-Dokument,
Behandlung durch den Altova XSLT 2.0-Prozessor, 420
Whitespace Nodes im XML-Dokument,
und Behandlung durch den XSLT 1.0-Prozessor, 418
Wiedergabe, 449
Wildcards,
Verwendung von Anführungszeichen in der Befehlszeile, 213
Windows,
Unterstützung für Altova-Produkte, 445
Windows 2000, 448
Windows NT, 448
Word2007+,
Register im Mapping-Fenster, 64
Wrapper,
Klassen, 402
Wrapper-Klassen,
Versionskompatibilität, 402

X

Xerces,
Bibliotheken, 402
XML, 448
Ausgabe speichern, 120
Ausgabe validieren, 120
Streaming, 120
XML-Schema von XML-Datei generieren, 20
XML Parser,
Info, 446
XML-Dateien,
anhand einer einzigen XML-Quelle, 169
XML-Instanz,
absoluter Pfad, 37
Input, 386
Output, 386
XML-Schema,
automatisch generieren, 20
xpath, 370
mehrere Nodes summieren, 318
xpath2,
Bibliothek, 66, 117
Bibliotheksfunktionen, 360
XQuery,
Erweiterungsfunktionen, 423
Funktionen, 66, 117
und Datenbanken, 254
XQuery-Prozessor,
in Altova-Produkten, 447
xs:,
constuctors, 361
Konstruktoren, 361
xs:date, 362
xs:time, 362
Xsi:nil,
nillable, 250
xsi:type,
mappen auf derived types, 237
xsl:preserve-space, 418
xsl:strip-space, 418
XSLT,
benutzerdefinierte Funktionen hinzufügen, 312
Bibliotheksfunktionen, 370
Erweiterungsfunktionen, 423
generieren (Tutorial), 36
mehrere Ziele generieren, 41
Register im Mapping-Fenster, 64
Standardbibliothek, 66, 117
Template Namespace, 312
XSLT 1.0-Prozessor,
Einschränkungen und implementierungsspezifisches Verhalten, 418
XSLT 2.0,
benutzerdefinierte Funktionen hinzufügen, 317
XSLT 2.0 Stylesheet,
Namespace-Deklarationen, 420
XSLT 2.0-Funktionen,

© 2011 Altova GmbH

XSLT 2.0-Funktionen,
implementierungsspezifisches Verhalten, 422
spezifische Funktionen siehe unter fn:, 422
XSLT 2.0-Prozessor,
allgemeine Informationen, 420
Informationen, 420
XSLT2.0,
Date-Konstruktor, 317
XSLT-Prozessoren,
in Altova-Produkten, 447

Z

Zeichen-Entities,
in der HTML-Ausgabe von XSLT-Transformationen, 418
Ziel,
mehrere Schemas, 37
mehrere Schemas anzeigen, 41
Root-Element, 205
Zieldatei,
mehrere Zieldateien anhand einer einzigen Quelldatei, 169
Zielelement,
mehrere Quellen mappen, 43
Zielorientiert,
Mapping, 88
Zoom,
Faktor im Ausgabefenster, 120
Zusammenführen,
mehrere Input-Dateien, 164
XML -Dateien, 209
Zuweisen,
Globale Ressource zu einer Komponente, 141

© 2011 Altova GmbH