Altova StyleVision 2021 Professional Edition

ユーザーマニュアル

Altova StyleVision 2021 Professional Edition
ユーザーマニュアル

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.

公開日: 2015-2021

(C) 2015-2021 Altova GmbH

目次

1 はじめに 17

1.1 新規機能: バージョン 2021 ... 18
1.1.1 バージョン 2020 ... 18
1.1.2 バージョン 2019 ... 18
1.1.3 バージョン 2018 ... 19
1.1.4 バージョン 2017 ... 19
1.2 製品機能 ... 20
1.3 Altova 製品における Authentic View ... 24
1.4 SPS とは? ... 25
1.5 StyleVision のセットアップ ... 27
1.6 用語解説 ... 28
1.7 このドキュメントについて ... 31

2 ユーザーインターフェイス 33

2.1 メインウィンドウ ... 34
2.1.1 デザインビュー ... 35
2.1.2 Authentic View ... 35
2.1.3 出力ビュー ... 37
2.2 サイドバー ... 39
2.2.1 デザイン概要 ... 42
2.2.2 スキーマツリー ... 44
2.2.3 デザインツリー ... 48
2.2.4 スタイルレポジトリ ... 51
2.2.5 スタイル ... 53
2.2.6 プロパティ ... 54
2.2.7 プロジェクト ... 58
2.2.8 メッセージ ... 59
2.2.9 検索と置換 ... 60

3 クイックスタートチュートリアル 61

3.1 新規 SPS の作成 ... 62
3.2 (XML ソースの)動的なコンテンツの挿入 ... 66
3.3 静的コンテンツの挿入 ... 73
3.4 コンテンツのフォーマット ... 78
3.5 自動計算の使用 ... 84
3.6 条件の使用 ... 88
3.7 グローバルテンプレートとその他のコンテンツの使用 ... 95
3.8 おしまい！ ... 100

4 使用方法の概要 101

4.1 SPS とソース ... 102
4.2 デザインの作成 ... 103
4.3 XSLT と XPath のバージョン ... 104
4.4 Internet Explorer の互換性 ... 105
4.5 SPS と Authentic View ... 107
4.6 StyleVision と Authentic の同期 ... 109
4.7 生成されたファイル ... 110
4.8 StyleVision のプロジェクト ... 112
4.9 StyleVision におけるカタログ ... 117
4.10 StyleVision 内のカタログ ... 121
4.10.1 カタログの仕組み ... 121
4.10.2 StyleVision 内のカタログ構造 ... 122
4.10.3 カタログのカスタム化 ... 123
4.10.4 Windows システムロケーションのための変数 ... 124

5 SPS コンテンツ 126

5.1 テキストとして XML コンテンツを挿入 ... 127
5.1.1 定義済みのフォーマットとともにコンテンツを挿入する ... 129
5.1.2 Authentic View にて要素を追加する ... 130
5.1.3 その他のコンテンツ ... 132
5.2 MS Word コンテンツの挿入 ... 134

5.3 MS Excel コンテンツの挿入 136
5.4 ユーザー定義テンプレート 139
5.5 ユーザー定義要素、XML テキストブロック 142
5.5.1 ユーザー定義要素 142
5.5.2 ユーザー定義 XML テキストブロック 143
5.6 テーブル 145
5.6.1 静的テーブル 146
5.6.2 動的テーブル 148
5.6.3 テーブル内の条件付き処理 152
5.6.4 デザインビュー内のテーブル 153
5.6.5 テーブルのフォーマット 155
5.6.6 行と列の表示 159
5.6.7 CALS/HTML テーブル 161
5.7 リスト 166
5.7.1 静的なリスト 166
5.7.2 動的なリスト 168
5.8 イメージ 171
5.8.1 イメージ: URI とインラインデータ 171
5.8.2 イメージ型 と出力 173
5.8.3 サンプル: イメージのテンプレート 176
5.9 フォームコントロール 178
5.9.1 入力フィールド、複数行の入力フィールド 179
5.9.2 チェックボックス 180
5.9.3 コンボボックス 181
5.9.4 ラジオボタン、ボタン 184
5.10 リンク 186
5.11 バーコード 187
5.12 レイアウト モジュール 191
5.12.1 レイアウトコンテナー 191
5.12.2 レイアウトボックス 194
5.12.3 ライン 198
5.13 変更機能 200

6 SPS 構造 203

6.1 スキーマソース 205
6.1.1 DTD と XML スキーマ 206
6.1.2 DB スキーマ 211
6.1.3 ユーザー定義スキーマ 212
6.2 複数のソースから XML データのマージ 215
6.3 モジュール SPS 218
6.3.1 利用可能なモジュールオブジェクト 219
6.3.2 SPS の作成 222
6.3.3 サンプル: Address Book 226
6.4 テンプレートと デザインフラグメント 231
6.4.1 メインのテンプレート 231
6.4.2 グローバルテンプレート 232
6.4.3 ユーザー定義テンプレート 235
6.4.4 変数テンプレート 238
6.4.5 ノードテンプレート操作 239
6.4.6 デザインフラグメント 242
6.5 XSLT テンプレート 246
6.6 複数の ドキュメントの出力 248
6.6.1 新規 ドキュメントテンプレートの挿入 249
6.6.2 新規ドキュメント テンプレートと デザイン構造 250
6.6.3 新規ドキュメント テンプレートの URL 250
6.6.4 プレビューファイルと出力 ドキュメントファイル 253
6.6.5 ドキュメント プロパティ と スタイル 255

7 高度な機能 257
7.1 自動計算 258
7.1.1 自動計算の編集と移動 258
7.1.2 自動計算によりノードを更新する 261
7.1.3 更新されたノードを使用する自動計算 263
7.1.4 サンプル：注文書 264
7.2 条件 268
7.2.1 条件のセットアップ 268
7.2.2 条件の編集 271
7.2.3 出力をベースにした条件 272

7.2.4 条件と自動計算 273
7.3 条件付き表示 275
7.4 グループ化 277
7.4.1 サンプル: group-by (Persons.sps) 279
7.4.2 サンプル: group-by (Scores.sps) 281
7.5 並べ替え 284
7.5.1 並べ替えのメカニズム 284
7.5.2 サンプル: 複数のソートキーによる並べ替え 286
7.6 パラメーターと変数 290
7.6.1 ユーザー宣言パラメーター 290
7.6.2 デザインフラグメントに対するパラメーター 291
7.6.3 ソースのための SPS パラメータ 294
7.6.4 変数 294
7.6.5 Authentic にて編集可能な変数 296
7.7 目次、参照、ブックマーク 300
7.7.1 目次に含まれるブックマークアイテム 302
7.7.2 目次テンプレートの作成 309
7.7.3 サンプル: シンプルな目次 314
7.7.4 サンプル: 階層的ならびに連続的な目次 317
7.7.5 ドキュメントボディー上の自動番号 321
7.7.6 相互参照 324
7.7.7 ブックマーク と ハイパーリンク 326
7.8 サンプル: 複数の言語の使用 332

8 表示方法と出力 335

8.1 定義済みのフォーマット 336
8.2 出力のエスケープ 338
8.3 値フォーマット (数値データ型のフォーマット) 340
8.3.1 値フォーマットのメカニズム 340
8.3.2 値フォーマットの構文 343
8.4 CSS スタイルを使用する 350
8.4.1 外部スタイルシート 351
8.4.2 グローバルスタイル 354
8.4.3 ローカルスタイル 357

8.4.4 スタイルの値の設定 359
8.4.5 XPath を使用したスタイルプロパティ 362
8.4.6 合成スタイル 364
8.5 Authentic におけるテキストスタイルの柔軟性 368
8.5.1 合成スタイル 368
8.5.2 リッチエディット 370
8.5.3 テキスト状態アイコン 374
8.6 印刷媒体出力のデザイン 377
8.6.1 ドキュメント セクション 378
8.6.2 改ページ 392
8.6.3 フットノート 393
8.6.4 ピクセルの解像度 394
8.6.5 透かし 396

9 追加機能 400
9.1 Altova グローバルリソース 401
9.1.1 グローバル リソースの定義 401
9.1.2 グローバルリソースを使用する 410
9.2 Authentic ノードプロパティ 416
9.3 クリックによる親ノードの置き換え 418
9.4 追加検証 420
9.5 パースされていないエンティティ URI 422
9.6 XSLT、XSL-FO または FO ファイルからの新規作成 424
9.7 ユーザー定義 XPath 関数 427
9.7.1 XPath 関数の定義 429
9.7.2 ノードを検索するための関数の再利用 431
9.7.3 XPath 関数内のパラメーター 432
9.8 日付の使用 441
9.8.1 日付選択の使用 442
9.8.2 日付のフォーマット 443
9.9 スクリプトの使用 446
9.9.1 JavaScript 関数の定義 447
9.9.2 関数をイベントハンドラーとして割り当てる 448
9.9.3 外部 JavaScript ファイル 449

9.10 HTML のインポート 452
9.10.1 HTML のインポートにより新規 SPS を作成 452
9.10.2 スキーマと SPS デザインの作成 454
9.10.3 要素/属性としてテーブルやリストを作成 456
9.10.4 出力の生成 458
9.11 ASPX インターフェイスによる Web アプリケーション 459
9.11.1 サンプル: Windows 7 にて動作する localhost 460
9.12 PXF ファイル:SPS とその関連ファイルのコンテナー 462
9.12.1 PXF ファイルの作成 462
9.12.2 PXF ファイルの編集 466
9.12.3 PXF ファイルのデプロイ 467

10 データベース 469
10.1 データベース と StyleVision 471
10.2 データソースへの接続 473
10.2.1 接続ウィザードの開始 474
10.2.2 データベースドライバー概要 475
10.2.3 ADO 接続の設定 478
10.2.4 ADO.NET 接続のセットアップ 484
10.2.5 ODBC 接続の設定 491
10.2.6 JDBC 接続の設定 494
10.2.7 PostgreSQL 接続のセットアップ 498
10.2.8 SQLite 接続の設定 499
10.2.9 グローバルリソースからの接続の使用 501
10.2.10 データベース接続のサンプル 502
10.3 DB Data Selection 557
10.3.1 非 XML データベース 557
10.3.2 XML データベース 564
10.4 DB スキーマと DB XML ファイル 567
10.5 DB に対する SPS デザイン機能 570
10.5.1 DB フィルター: DB データのフィルタリング 572
10.6 出力ファイルの生成 578
10.7 データベースへの問い合わせ 579
10.7.1 データソース 580

10.7.2 ブラウザーペイン: DB オブジェクトの閲覧 583
10.7.3 クエリペン: 説明と機能 586
10.7.4 クエリペイン: クエリと作業する 589
10.7.5 結果とメッセージ 590

11 Authentic View 592

11.1 Authentic View インターフェイス 593
11.1.1 GUI の概要 593
11.1.2 Authentic View ツールバーアイコン 594
11.1.3 Authentic View メインウィンドウ 595
11.1.4 Authentic View エントリヘルパー 598
11.1.5 Authentic View コンテキストメニュー 601
11.2 Authentic View 内で編集する 604
11.2.1 基本的な編集 604
11.2.2 Authentic View 内のテーブル 608
11.2.3 DB の編集 616
11.2.4 日付と作業する 622
11.2.5 エンティティを定義する 624
11.2.6 Authentic View 内のイメージ 625
11.2.7 Authentic View 内のキーストロール 626

12 Authentic スクリプト 627

12.1 スクリプトエディター 629
12.2 マクロ 630
12.2.1 デザイン要素上のマクロ 630
12.2.2 コンテキストメニューアイテム上のマクロ 632
12.2.3 カスタム ボタン 633
12.3 イベントハンドラー 635
12.4 スクリプトオプション 636

13 処理の自動化 637

13.1 コマンドライン インターフェイス 638
13.1.1 StyleVision 638

13.1.2 StyleVision Server ... 639
13.2 RaptorXML の使用 ... 641
13.2.1 PDF 出力 ... 641
13.3 FlowForce Server を使用した自動化 ... 643
13.4 自動処理の方法 ... 645

14 Visual Studio で動作する StyleVision 646

14.1 Visual Studio プラグインのインストール ... 647
14.2 スタンドアロン版 StyleVision との違い ... 648

15 Eclipse で動作する StyleVision 649

15.1 Eclipse の StyleVision プラグインをインストールする ... 650
15.2 Eclipse Stylevision 内のエントリポイント ... 656

16 メニューコマンドとレファレンス 660

16.1 デザイン ビューのシンボル ... 661
16.2 XPath 式 の編集ダイアログ ... 665
16.2.1 エバリュエーター ... 666
16.2.2 デバッガー ... 669
16.2.3 式ビルダー ... 676
16.3 ツールバー ... 680
16.3.1 書式 ... 682
16.3.2 テーブル ... 683
16.3.3 Authentic ... 685
16.3.4 リッチエディット ... 687
16.3.5 デザイン要素の挿入 ... 687
16.3.6 デザインフィルター ... 690
16.3.7 グローバルリソース ... 691
16.3.8 標準 ... 691
16.4 ファイル メニュー ... 694
16.4.1 新規作成 ... 694
16.4.2 開く、再ロード、閉じる、すべて閉じる ... 700
16.4.3 デザインを保存, すべて保存 ... 705

16.4.4 名前を付けて保存 710
16.4.5 MobileTogether デザインファイルとしてエクスポートする 713
16.4.6 Authentic の XML データを保存、名前を付けて保存 713
16.4.7 生成されたファイルの保存 714
16.4.8 FlowForce へのデプロイ 715
16.4.9 Web デザイン 717
16.4.10 プロパティ 717
16.4.11 印刷プレビュー、印刷 723
16.4.12 最後に開かれたファイル、終了 724
16.5 編集メニュー 726
16.5.1 元に戻す、やり直し、すべて選択 726
16.5.2 検索、次を検索、置き換え 726
16.5.3 スタイルシートのパラメーター 731
16.5.4 マークアップの縮退/展開 732
16.6 プロジェクトメニュー 734
16.6.1 新規プロジェクトの作成、プロジェクトを開く、プロジェクトの再ロード 735
16.6.2 プロジェクトを閉じる、プロジェクトの保存 735
16.6.3 ファイル/グローバル リソース/URL をプロジェクトに追加 736
16.6.4 アクティブなファイル (ならびに関連するファイル) をプロジェクトに追加 738
16.6.5 プロジェクト フォルダーや外部フォルダーをプロジェクトに追加 738
16.7 表示メニュー 741
16.7.1 ツールバーとステータスバー 741
16.7.2 デザインサイドバー 742
16.7.3 デザインフィルター、ズーム 743
16.8 挿入メニュー 744
16.8.1 コンテンツ 744
16.8.2 その他のコンテンツ 745
16.8.3 リッチエディット 745
16.8.4 フォームコントロール 747
16.8.5 DB コントロール 748
16.8.6 自動計算 748
16.8.7 日付選択 749
16.8.8 段落、特殊な段落 750
16.8.9 バーコード 750

16.8.10 イメージ 752
16.8.11 水平線 754
16.8.12 テーブル 755
16.8.13 リスト 755
16.8.14 ブックマーク 757
16.8.15 ハイパーリンク 758
16.8.16 フットノート 760
16.8.17 条件、出力ベースの条件 761
16.8.18 テンプレート 763
16.8.19 ユーザー定義テンプレート 764
16.8.20 変数テンプレート 765
16.8.21 デザインフラグメント 766
16.8.22 レイアウトコンテナー、レイアウトボックス、ライン 766
16.8.23 目次 766
16.8.24 新規ドキュメント 767
16.8.25 ページ/カラム/ドキュメント セクション 767
16.8.26 ユーザー定義アイテム 768
16.9 タグで囲むメニュー 770
16.9.1 テンプレート 770
16.9.2 ユーザー定義テンプレート 771
16.9.3 変数テンプレート 771
16.9.4 段落、特殊な段落 772
16.9.5 リスト 772
16.9.6 ブックマーク と ハイパーリンク 773
16.9.7 条件、出力ベースの条件 774
16.9.8 目次ブックマークと目次レベル 775
16.9.9 新規ドキュメント 775
16.9.10 ユーザー定義要素 776
16.10 テーブル メニュー 778
16.10.1 テーブルの挿入、テーブルの削除 778
16.10.2 テーブルヘッダー/フッターの追加 779
16.10.3 行/列の追加と挿入 779
16.10.4 行/列の削除 780
16.10.5 左/右/下/上のセルを結合 780
16.10.6 水平方向/垂直方向にセルを分割 781

16.10.7 セルの境界とテーブル マークアップの表示 781
16.10.8 テーブル プロパティ 782
16.10.9 CALS/HTML テーブルの編集 782
16.10.10 セル コンテンツの高さ揃え 783
16.11 Authentic メニュー 785
16.11.1 Authentic スクリプトの編集 785
16.11.2 カスタム ツールバーボタン 786
16.11.3 マクロ参照のチェック 790
16.11.4 日付選択の自動追加 790
16.11.5 DB コントロールの自動追加 790
16.11.6 Authentic View の再ロード、XML の検証 791
16.11.7 XML データの新たな行を選択し編集 791
16.11.8 XML エンティティの定義 792
16.11.9 マークアップの表示 793
16.11.10 リッチエディット 794
16.11.11 (動的テーブル) 行コマンド 794
16.12 データベース メニュー 796
16.12.1 データベースへの問い合わせ 796
16.12.2 データベース フィルターの編集とクリア 797
16.13 プロパティ メニュー 799
16.13.1 リストの編集 799
16.13.2 定義済みの値フォーマット文字列 800
16.14 ツールメニュー 802
16.14.1 スペルチェック 802
16.14.2 スペル チェックのオプション 803
16.14.3 グローバルリソース 806
16.14.4 アクティブな構成 807
16.14.5 カスタマイズ 808
16.14.6 ツールバーとウィンドウの復元 813
16.14.7 オプション 813
16.15 ウィンドウ メニュー 816
16.16 ヘルプメニュー 817
16.16.1 目次、インデックス、検索 817
16.16.2 ライセンス登録、注文フォーム、レジストレーション、最新情報のチェック 817

16.16.3 他のコマンド ... 820

17 プログラマーの レファレンス 822

17.1 Scripting Editor ... 823
17.1.1 Creating a Scripting Project ... 823
17.1.2 Built-in Commands ... 835
17.2 Application API ... 845
17.2.1 Overview ... 846
17.2.2 Interfaces ... 865
17.2.3 Enumerations ... 957
17.3 ActiveX Integration ... 963
17.3.1 必要条件 ... 963
17.3.2 ActiveX コントロールをツールボックスに追加する ... 964
17.3.3 アプリケーションレベルの統合 ... 966
17.3.4 ドキュメントレベルの統合 ... 968
17.3.5 ActiveX 統合のサンプル ... 971
17.3.6 コマンド レファレンス ... 988
17.3.7 オブジェクトレファレンス ... 1001

18 付録 1023

18.1 XSLT および XQuery エンジンに関する情報 ... 1024
18.1.1 XSLT 1.0 ... 1024
18.1.2 XSLT 2.0 ... 1024
18.1.3 XSLT 3.0 ... 1026
18.1.4 XQuery 1.0 ... 1026
18.1.5 XQuery 3.1 ... 1029
18.2 XSLT と XPath/XQuery 関数 ... 1030
18.2.1 Altova 拡張関数 ... 1031
18.2.2 その他の拡張関数 ... 1105
18.3 DB により生成された XML スキーマ内のデータ型 ... 1122
18.3.1 ADO ... 1122
18.3.2 MS Access ... 1123
18.3.3 MS SQL Server ... 1123

18.3.4 MySQL ... 1124
18.3.5 ODBC ... 1125
18.3.6 Oracle ... 1126
18.3.7 Sybase ... 1126
18.4 ...技術データ ... 1128
18.4.1 OS とメモリ要件 ... 1128
18.4.2 Altova XML バリデーター ... 1128
18.4.3 Altova XSLT と XQuery エンジン ... 1128
18.4.4 Unicode のサポート ... 1128
18.4.5 インターネットの使用 ... 1129
18.5 ...ライセンス情報 ... 1130
18.5.1 電子的なソフトウェアの配布 ... 1130
18.5.2 ソフトウェアのアクティベーションとライセンスの計測 ... 1130
18.5.3 エンドユーザー使用許諾契約書 ... 1131

インデックス 1132

1 はじめに

Altova StyleVision 2021 Professional Edition は、StyleVision Power Stylesheet の編集を視覚的に行うためのアプリケーションで 64 ビット と32 ビット バージョンがご利用いただけます。StyleVision® は、プラットフォーム更新済みのW indows 7 SP1、W indows 8、W indows 10 と プラットフォーム更新済みのW indows Server 2008 R2 SP1 または以降 で作動します。StyleVision とAltova MissionKit の機能の一部は、Microsoft Office パッケージ（MS Access、MS Excel、MS W ord）、バージョン 2007 または以降のアプリケーションと統合することができます。

StyleVision Power Stylesheet（SPS）を以下の用途に使用することができます:

- XML ドキュメント をW YSIW YG ビューの **Authentic View** にて管理します。**Authentic View** は、Altova XMLSpy 、Altova StyleVision 、Altova Authentic Desktop とAltova Authentic Browserにて利用することができます。この機能により、XML ドキュメントをベースにした電子フォームを簡単に作成することができます。
- **Authentic View** にてデータベース（DB）の編集し、データベースのレポートをHTML と RTF フォーマットにて生成します。
- SPS デザインをベースにしたXSLT スタイルシートを生成します（XSLT 1.0 、XSLT 2.0、とXSLT 3.0 がサポートされます）。XSLT スタイルシートはStyleVision の外部でXML ドキュメントをHTML と（MS W ord などのワードプロセッサーアプリケーションで使用できる）RTF（リッチテキスト）のファイル形式へ変換するために使用することができます。
- StyleVision 内から、HTML とRTF の形式で出力を直接生成します。SPS がデータベースをベースにしている場合、StyleVision は、をベースにしたXML スキーマを各 SPS に対して生成し、スキーマに対して妥当で、DB から得られたデータを含んでいるインスタンスドキュメントを生成することができます。

最終更新日: 2021 年 02 月 25 日

1.1 新規機能: バージョン 2021

バージョン 2021 リリース 2

- 検索と置き換え 機能がデザインビューに搭載されました。デザインテキスト、スタイル、プロパティ、XPath 式内で入力するまたは正規表現を構成する文字列を検索することができます。
- 追加データベースのためのサポート。
- 新規のスキーマ情報にアクセスするためのAltova 拡張関数 。
- Eclipse へのサポート が拡張され、以下のバージョンがカバーされています: 2020-12; 2020-09; 2020-06; 2020-03.

バージョン 2021

- 追加データベースのためのサポート。
- 新規のスキーマ情報にアクセスするためのAltova 拡張関数 。
- Eclipse へのサポート が拡張され、以下をカバーしています: 2020-06; 2020-03; 2019-12; 2019-09.

1.1.1 バージョン 2020

バージョン 2020 リリース 2

- XPath デバッガー内の改善。
- 追加データベースのためのサポート。
- Eclipse 内でのStyleVision の統合へのサポートが Eclipse 4.13 と4.14 に拡張されました。

バージョン 2020

- XPath デバッガー がXPath 式 の編集 ダイアログに追加されました。これによりXPath 式をXML ファイルに対してダイアログ内で直接 テストしてデバッグできます。
- XPath 式 の編集 ダイアログ が簡素化され最適化されました。
- Visual Studio 内でのStyleVision の統合がVisual Studio 2019 に拡張されました。
- Eclipse 内でのStyleVision の統合がEclipse 4.11 と4.12. に拡張されました。

1.1.2 バージョン 2019

バージョン 2019 リリース 3

StyleVision **バージョン 2019r3** の新機能は以下にリストされています。

- The オプションダイアログ (「ツール | オプション」) により自動的に検知されるパスをオーバーライドする Java 仮想マシンパスを設定するためのオプションが提供されます。
- Windows Server 2019 へのサポート。
- Eclipse へのサポートが Eclipse 4.9 と4.10 に拡張されました。

(C) 2015-2021 Altova GmbH

バージョン 2019

StyleVision バージョン 2019 の新規機能とアップデートは以下にリストされています。

- XSLT エンジン内のスピードの向上
- XSLT エンジン内の改善されたスペックの適合性
- 拡大されたデータベースへのサポート: Firebird 3.0、IBM Informix 12.10、MariaDB 10.3、Microsoft SQL Server 2017、MySQL 8.0、PostgreSQL 10 。
- Eclipse 4.8 との統合

1.1.3 バージョン 2018

バージョン 2018 リリース 2

StyleVision バージョン 2018r2 の新規機能とアップデートは以下にリストされています。

- デザインを Altova MobileTogether デザインとしてリッチテキストをしようしてインポートすることができるようになりました。
- Teradata 16 とMariaDB 10 データベースのサポート。
- 高度なネットワークプロキシの設定

バージョン 2018

StyleVision バージョン 2018 の新規機能とアップデートは以下にリストされています。

- XSLT エンジン内の大幅なスピードの向上
- XSLT エンジン内の改善されたスペックの適合性
- Sybase 16、PostgreSQL 9.6、 MySQL 5.7 のためのサポート
- Eclipse 4.7 との統合

1.1.4 バージョン 2017

バージョン 2017 リリース 3

StyleVision バージョン 2017 リリース 3 の新規機能とアップデートは以下にリストされています。

- SPS ファイルから MobileTogether デザインファイルを生成する
- Altova LicenseServer を使用してライセンスを与える
- バグのフィックスと、内部の強化

バージョン 2017

StyleVision バージョン 2017 の新規機能とアップデートは以下にリストされています。

- MS Excel 2007+ ドキュメントをベースにした SPS のサポート
- バグのフィックスと、内部の強化

1.2 製品機能

StyleVision の主な製品機能は、以下にリストされるとおりです:

一般的な 製品機能

StyleVision に搭載された、主な機能を以下に示します:

- Enterprise ならびに Professional Edition は32ビットならびに64ビット版がご利用になれます。
  - StyleVision の機能を StyleVision Server から呼び出すことができます。
  - StyleVision の機能を 外部アプリケーション内に統合 し StyleVision を Visual Studio と Eclipse へ統合することができます。

ソース

SPS は XML スキーマや DTD をベースにデザインすることができます。デザインには XML や CSS ファイルといった、その他のソースファイルを使用することもできます。ソースに関して以下の機能を使用することができます:

- Altova グローバルリソース により、スキーマや XML、CSS といったソースファイルを指定することができます。グローバルリソースメカニズムにより、より素早い開発や試験を行うことが可能になり、ソースデータの素早い変更を行い、その他の Altova アプリケーションに搭載された機能を StyleVision 内部から使用することが可能になります。
- HTML ドキュメントを XML に変換することができます。

インターフェイス

一般的な GUI の機能を以下に示します:

- 複数の SPS デザイン 複数のデザインを同時に開くことができ、そのうちの1つがアクティブな状態となります。各 SPS デザインが異なるタブで表示されます。
- テンプレートフィルター デザインドキュメントの外観をカスタマイズすることができます。現在編集されていないテンプレートの表示を無効にし、編集の効率を向上させることができます。
- デザインビューでのマークアップを隠す: デザインビュー内にあるタグを隠し、縮退させることで、デザインビューのスペースを有効活用することができます。
- SPS のデザインを行う際に、対応するタブをクリックすることで、Authentic View, 出力ビュー、そしてスタイルシートを表示することができます。この機能により出力や XSLT コードを素早く確認し、Authentic View の機能をテストすることができます。
- SPS が XML ソースドキュメント または ソース DB と関連付けられている場合、StyleVision の Authentic View から直接ソースドキュメントを編集することができます。

データベース

次の DB 機能がサポートされています:

- DB レポート を StyleVision で確認することができるほか HTML と RTF Word 形式で保存することができます。
- XML カラムを含む IBM DB2 データベース がサポートされます。
- DB への問い合わせ を StyleVision から直接行うことができます。

出力

インストールされたエディションにより、様々な出力フォーマットがサポートされます。出力に関連する以下の機能がサポートされます:

- XSLT バージョン 1.0、2.0、と3.0 のバージョンがサポートされます。

(C) 2015-2021 Altova GmbH

- Enterprise とProfessional Editions では、複数の出力フォーマット (HTML と RTF) がサポートされ、単一のSPS デザインからこれらのファイルを生成することができます。
- SPS コンポーネントに対して条件をセットすることで、個々の出力形式に対して異なる振る舞いを定義することができます。情報の細かさや異なる出力を柔軟に構成することで、特定の形式に対応した出力を行うことができます。
- XSLT ファイルと出力ファイルの両方を GUI から直接、またはStyleVision Server から、生成、保存することができます。
- Altova により開発された PXF ファイルフォーマット を使用することで、SPS ファイルを関連するソースならびにデータファイルとともに保存することができます。この形式により、SPS ファイル単体ではなく、SPS プロジェクト全体を簡単に受け渡すことができるようになります。
- ASPX インターフェイスによるW eb アプリケーション: この機能により、HTML ウェブページの更新を簡単に行うことができるようになります。StyleVision ではSPS からASPX アプリケーションに必要な全てのファイルが生成されます。ウェブページ(.aspx ファイル)が更新されると、XSLT により(更新箇所を含んだ)XML が動的にウェブページへ変換されます。

SPS デザイン機能

SPS のデザインに関する主な StyleVision の機能を以下に示します:

- SPS には、SPS 内に入力した 静的なテキスト の他 ソースドキュメント から選択された動的なテキストを含めることができます。
- 動的なコンテンツ は、スキーマソースにあるノードをドラッグアンドドロップすることでデザイン内に挿入することができます。デザイン要素(段落、リスト、イメージなど)を挿入した後に、スキーマツリーからXML ノードをデザイン要素へ割り当てることができます。
- 動的なコンテンツ はテキスト、または(入力フィールド または コンボボックス といった) データ入力デバイス として挿入することができます。コンボボックス のようなデータ入力デバイスを挿入した場合、更なる設定を行うことができ、例えば(Authentic View ユーザーにより)選択されるノードの値を、列挙値から得るといったことが可能です。
- メインのテンプレートにより、デザインの構造が指定、管理されます。この構造は グローバルテンプレート と呼ばれるオプションのテンプレート要素にて修正することができます。
- グローバルテンプレート は個々のデータ型に対して作成することができ、各種の型に応じた処理を定義することができます。
- 複数のドキュメントの出力: SPS により生成される出力を複数のドキュメントへ分割することができます。デザインには新規ドキュメントテンプレートが作成され、その中にコンテンツが配置されます。各新規ドキュメントテンプレートにより、異なるドキュメントが出力されます。.
- ユーザー定義テンプレート: 指定されたXPath 条件式により、複数のアイテムに対してテンプレートを生成することができます。これらアイテムは原子型の値またはノードとなり、XPath 条件式により、より詳細なノードの指定を行うことができ、そのセクションに対する条件付けやフィルタリングを行うことができるようになります。
- ユーザー定義要素: この機能はデザイン内の任意の場所に(HTML、XSLT、XSL-FO といった) 表示言語要素を自由に挿入するために開発されました。
- ユーザー定義 XML テキストブロック: XML テキストブロックをデザイン内の任意の場所に挿入することで、これらブロックが、生成されたXSLT スタイルシートの対応する場所にて生成されます。
- デザインフラグメント によりSPS 内にあるテンプレートのモジュール化と再利用を行うことができるようになり、関数の使用と同じ様な方法により、複数のSPS からアクセスすることができるようになります(*モジュール SPS*)を参照ください)。
- SPS モジュール 既にあるモジュールをその他のSPS モジュールへ追加することができ、あるSPS モジュールで定義されたオブジェクトを他のモジュールでも利用することができるようになります。この機能により、複数のSPS にてモジュールオブジェクトを最利用することができるようになり、メンテナンスを簡単に行えるようになります。
- XSLT テンプレート: 生成されたスタイルシートへXSLT ファイルをインポートすることができます。XML インスタンスドキュメント内にあるノードがインポートされたXSLT 内のテンプレートにマッチして、インポートされたテンプレートよりも高い優先度のテンプレートが無い場合、インポートされたテンプレートが使用されます。更に、インポートされたXSLT ファイル内部の名前付きテンプレートを、デザイン内部から呼び出すこともできます。
- XSLT からの新規作成: SPS ファイルをXSLT またはXSL-FO ファイルから作成することができます。XSLT 内にあるテンプレートの構造とスタイルがSPS にて作成され、SPS コンポーネントを修正してコンテンツやフォーマットをSPS に追加することができます。
- ユーザー定義 XPath 関数: ユーザー定義のXPath 関数を作成し、ドキュメント内でXPath 関数の使用が許されている場所から、関数の呼び出しを行うことができます。
- ドキュメント セクション: ドキュメントを複数のセクションに分割することで、各セクションにページプロパティのような、独立したプロパティを与えることができるようになります。この機能により、ドキュメントのパーツごとに異なる方式により表示を行うことが可能になります。
- レイアウトコンテナー: レイアウトコンテナーは、内部にデザイン要素を絶対位置指定により配置することができるブロックです。

- 青写真: レイアウトコンテナー内にフォームのイメージを配置することで、デザインの青写真として使用することができます。青写真を使用することで、既に存在するデザインを正確に再現することが可能になります。
- XMLドキュメントでは、データ構造を繰り返し作成することが簡単に行えます。例えば、ある部署にはお通常複数の従業員が勤務しています。この場合、各従業員のデータがデータ構造の中に収められ、その構造が従業員の数だけ繰り返されることになります。SPS では、このような各データ構造に対する処理 を一度定義して、関連するノード（この例では各従業員ノード）に対して適用することができます。。
- XSLT 2.0 と3.0 SPS 内に複数の内容のテーブルを挿入することができます。
- 繰り返し表示されるデータ構造を動的テーブルとして挿入することもできます。この機能により、テーブル構造に対してループを作成し、テーブル内部にて各行（または カラム）を生成することが可能になります。
- 繰り返される要素は、選択された（1つ以上の） ソートキーにより並べ替え を行うことができ、並べ替えられた要素セットが出力 (HTML と RTF)。に送信されます。
- 変数: 変数をテンプレート上で宣言して、XPath 条件式により指定された値を受け取ることができます。これまで、変数の値は作成されたノードの選択に制限されていました。バージョン 2010 にて導入された変数により、XPath 条件式を変数の値として指定することができるようになりました。
- 共通したデータコンテンツ（例えば属性値の共通値）やその位置をベースに、ノードの グループ化 を行うことができます。
- 条件テンプレート 機能により、XMLドキュメントやシステム環境といった条件が満たされた時に、テンプレートのセットから処理するものを選択することができます。この機能を使用することで、SPS ドキュメントの作成時に、ソースドキュメント内に含まれる情報や未知の情報に対して条件を指定することができるようになります。利用することができる条件は、XPath 1.0 または XPath 2.0 条件式で扱うことのできる条件に限定されます。
- 自動計算 によりソースドキュメントから得られたデータを操作して、その結果を表示することができます。この機能は、数値に対して計算処理を行うような状況（例えば注文書の合計金額を計算する）や、文字列（例えば ハイフンをスラッシュに変更）に対して使用することができ、コンテンツの生成を行うことができます。これらのデータ操作には、XPath 条件式を使用することができます。ネイティブ Java ならびに.NET 関数を、自動計算内のXPath 条件式にて使用することもできます。
- Authentic View にてデータが編集された場合、 自動計算 の結果をソースドキュメントのノードへ渡すことができます。このような処理は（自動計算の値を使った） XML ノードの更新 と呼ばれます。
- 追加検証 を使用することで（スキーマの検証に加えて）ノードにて定義されたXPath 条件式に対して、個々のXMLドキュメントノードを検証することができます。この機能により、Authentic View ユーザーが入力した値が不正である時にメッセージが表示され、そのノードに対して指定されたエラーメッセージにより、問題の詳細を表示することができます。
- バーコード: 新たに追加されたデザインコンポーネントにより、デザイン内にバーコードが簡単に挿入できるようになりました。バーコードイメージを簡単に作成し、出力ドキュメント内に配置することができます。
- 条件付き表示: 幾つかのデザインコンポーネントには条件付き表示プロパティが含まれています。指定された条件が満たされた時に、条件デザインコンポーネントが出力に作成されます。
- イメージ をデザイン内に挿入することができます。イメージのURI は静的（SPS 内で入力）または動的（ソースドキュメント内にあるノードから取得）、または静的と動的パートの組み合わせにより構成することができます。
- インラインデータからのイメージ: XMLドキュメント内のテキストとしてエンコードされたBase-16 やBase-64 から、イメージを生成することができます。この機能により、イメージをテキストとしてソースXMLドキュメント内に直接挿入することができるようになりました。
- 静的ならびに動的な、2種類の リスト を作成することができます 静的なリスト では、各リストアイテムがSPS にて定義され、 動的なリスト では、ノードがリストアイテムとして作成され、ノードに対応する全インスタンスの値がリストのアイテムとして作成されます。
- 静的と動的なリンク をデザイン内へ挿入することができます。ターゲット URI は静的（SPS 内で入力）または動的（ソースドキュメント内にあるノードから取得）、または静的と動的パートの組み合わせにより構成することができます。
- 静的ブックマーク を挿入することができます。これらブックマークは、ハイパーリンクのリンク先となるアンカーとして使用することができます。
- SPS 全体に対してパラメータ を宣言することができます。パラメーターは名前と文字列の値により宣言され、SPS 内部にあるXPath 条件式にて使用することができます。宣言したパラメーター値はデフォルト値として使用され、StyleVision Server にて与えられた値によりオーバーライドすることができます。
- 入力フォーマット 入力フォーマット機能により、数値型のXML スキーマデータ型ノードのコンテンツを、Authentic View に適したかたちや その他の出力の表示を行うためにフォーマットし直すことができます。入力フォーマット機能は、自動計算の結果に対しても使用することができます。
- JavaScript 関数 %SPS%> にて使用することができ、ユーザー定義関数をHTML 出力にて使用することができるようになります。.

- Authentic スクリプト: Authentic View にて、更なる柔軟性と双方向性が提供されます。
- GUI に用意されている数多くの 定義済みHTML フォーマット を利用して、個々のSPS コンポーネントへ適用することができます。
- 用意されている CSS テキストフォーマットやレイアウトプロパティを、スタイルサイドバーから個々のSPS コンポーネントに対して適用することができます。
- 更に、SPSの グローバルなレベルにおける HTML セレクターに対して、または外部 CSS スタイルシートに対して CSS スタイルを定義することができます。これらスタイルルールは Authentic View とHTML 出力に対して適用され、フォーマットならびにレイアウトの柔軟性が大幅に拡張されます。
- Path 条件式を使ってもスタイルの割り当てを行うことができます。この機能により、XML ドキュメントからスタイルプロパティの値を選択し、条件付きでプロパティの値をセットすることができます。
- ページレイアウト(RTF や PDF、バージョン 2007 以上の Word)に対して、方向、余白、ページ番号、ヘッダーやフッターといった数多くの ページレイアウト オプション, が用意されており、SPS にて指定することができます。

関連項目

- ユーザーインターフェイス
- 一般的な使用方法

1.3 Altova 製品における Authentic View

Authentic View は、次の Altova 製品で使用することのできるグラフィカルな XML ドキュメント エディターです:

* Altova XMLSpy
* Altova Authentic Desktop
* Altova Authentic Browser
* Altova StyleVision

StyleVision では、Authentic View へメインウィンドウの Authentic eForm タブからアクセスすることができます。

1.4 SPS とは?

StyleVision Power Stylesheet（またはSPS）は拡張されたXSLT スタイルシートです。:

- XML ドキュメントならびにデータベース Authentic View 内にあるデータエントリーの表示方法を制御します。
- XML ドキュメント変換の出力デザインを指定します。

SPS は、 .sps ファイル拡張子で保存されます。

SPS のデザイン

SPS は、StyleVision 内部で視覚的に作成されます。スキーマ（DTD またはXML スキーマ）をベースにしており、SPS がDB とともに使用される場合、DB 構造からStyleVision により自動的に生成されたXML スキーマをベースにします。SPS のデザインは柔軟性に富んでおり、動的ならびに静的なコンテンツを含めることができます 動的なコンテンツ とは XML ドキュメント またはDB 内にあるデータのことです。静的コンテンツ とは、SPS 内に直接入力されたコンテンツのことです。動的コンテンツはテキストとして直接、または入力フィールドやコンボボックス、テーブルといったコンポーネントの内部に含めることができます。更に、動的なコンテンツを（自動計算機能により）操作することもでき、ソースドキュメント内である条件が満たされた場合に表示するといったことも可能になります。SPS では複数の場所に異なるコンテンツを配置することができ、イメージやハイパーリンク、JavaScript 関数といったその他のコンポーネントを含めることもできます。SPSの各コンポーネントは必要に応じてフォーマットすることができます。

SPS とAuthentic View

完成したSPS がXMLドキュメントやDB に関連付けられた場合、そのXML やDB をAuthentic View で編集することができるようになります。Authentic View は配布されたXMLドキュメントやDB を視覚的に更新する理想的な手段となります。Authentic View により、XMLドキュメントやDB のコンテンツを、複数のユーザーがグラフィカルに編集することができるようになります。StyleVision にて SPS のデザインを行うと、SPS の確認を（そのSPS に対応したAuthentic View で）行うことができます。SPS を使った Authentic View の詳細については、次を参照してください： SPS とAuthentic View

SPS とXSLT スタイルシート

SPS のデザインが完成したら、作成したデザインをベースにしたXSLT スタイルシートを生成することができます。StyleVision ではXSLT 1.0 とXSLT 2.0 とXSLT 3.0 がサポートされており、単一のSPS からHTML、RTF、XSL-FO、そして2007 以降のW ord に対応したXSLT スタイルシートを生成することができます（XSL-FO とW ord 2007 以降はEnterprise Edition でのみサポートされ、RTF はEnterprise とProfessional Edition で、HTML 出力はBasic Edition でのみサポートされます）。生成されたXSLT スタイルシートにより、XSLT スタイルシートの生成に使用されたSPS と同一のSPS をベースにしたXML ドキュメントの変換を行うこともできます。XSLT スタイルシートの使用方法に関する詳細は、生成されたファイルのセクションを参照ください。

SPS と出力

StyleVision では出力（Enterprise Edition ではHTML、RTF、XSL-FO、PDF、Professional ならびにBasic Edision ではHTML）を直接生成することもできます。出力ビュー のタブにより、アクティブな SPS ドキュメントの出力を、StyleVision GUI にて直接表示することができます。目的の出力ファイルは、(i) StyleVision GUI の 「ファイル| 生成されたファイルの保存」 コマンド または StyleVision Server コマンドを呼び出すことで生成することができます。

Altova 製品内のAuthentic View

Authentic View は、次のAltova 製品で使用することのできるグラフィカルな XML ドキュメント エディターです:

* Altova XMLSpy
* Altova Authentic Desktop
* Altova Authentic Browser
* Altova StyleVision

関連項目

- 一般的な使用方法

(C) 2015-2021 Altova GmbH

1.5 StyleVision のセットアップ

Altova StyleVision は、 プラットフォーム更新済みのWindows 7 SP1、Windows 8、Windows 10 上で動作します。Altova Web サイト からStyleVision をダウンロードした後に、実行ファイル(.exe) をダブルクリックして、セットアッププログラムを起動します。セットアッププログラムにより StyleVision を目的の場所へインストールすることができます。Altova XSLT Engines (1.0 と2.0) は、StyleVision 内部に搭載されており、内部的な全ての変換に使用されます。StyleVision とは別にXSLT エンジンをインストールする必要はありません。。

以下のコンポーネントをインストールする必要があります:

- Authentic View 、HTML プレビューそしてデザインビューを使用するために Internet Explorer 5.5 以上が必要になります。Internet Explorer 6.0 以降では XML のサポートが強化されており、インストールが推奨されます。Internet Explorer 9 もサポートされます。
- RTF のプレビューを行うために Microsoft Word 2000 以降が必要になります。Microsoft Word 2003 以降が推奨されます。(Excel テーブルとHTML ページコンテンツなど、Word ドキュメントに貼り付けることのできるコンテンツの) Word ドキュメントからのコピーと貼り付け Word 2007+ が必要になります
- Word 2007 タブにて Word 2007 以降の出力をプレビューするのに、バージョン 2007 以降の Microsoft Word、またはバージョン 2007 以降の Word Viewer が必要になります。Microsoft Word 2003 の互換性パックを使用することで Word 2007 出力のプレビューを行うこともできますが、バージョン 2007 以降の Word ファイルを正しくレンダリングできない場合もあります。

メモ このドキュメンテーションでは、バージョン 2007 以降の Word を、**Word 2007+** と省略して表示します。

メモ 埋込型のプレビューに問題がある場合、StyleVision はプレビュードキュメントを外部アプリケーション(通常 Microsoft Word またはAdobe Reader)にて開くように試みます。埋込型のプレビューに対するエラーメッセージがStyleVision にて表示されます。プレビュードキュメントが外部アプリケーションにて開かれた場合、一時出力ドキュメントを再度生成する前に、外部アプリケーションを終了する必要があります。外部アプリケーションにてドキュメントが開かれている場合、ファイルが他のプロセスにて使われているというエラーメッセージが表示されます。外部アプリケーションによりファイルがロックされるため、SPS デザインを閉じる前に外部アプリケーションを閉じる必要もあります。

RaptorXML

RaptorXML を使用することで、XML ファイルを目的の出力フォーマットへ変換することができます。

関連項目

- 生成されたファイル
- 一般的な使用方法
- 自動化された処理

1.6 用語解説

このセクションでは、StyleVision GUI ならびにそのドキュメンテーションで使用される用語の解説を行います。用語は以下に示されるグループに分けられ、グループ内でアルファベット順に表示されます。

Altova 製品に関連する用語

Altova 製品に関連する用語を以下に示します:

Authentic View 以下のAltova 製品にてXMLドキュメントのエディタービューを利用することができます: Altova XMLSpy; Altova StyleVision; Altova Authentic Desktop; Altova Authentic Browser 。Authentic View ならびにAltova 製品に関する詳細については、Altova W eb サイトを参照ください。

SPS StyleVision Power Stylesheet を省略したもので、このドキュメンテーションではStyleVision により作成されたデザインドキュメントのことを指し、.sps という拡張子により保存されます。詳細については、SPS とは? を参照してください。

グローバルリソース ファイルのセット、フォルダーのセット、そしてデータベースのセットに対するエイリアスです。各リアイアスには構成セットが含まれており、各構成がリソースにマッピングされます。StyleVision にてグローバルリソースが使用される場合、StyleVision におけるアクティブな構成を変更することで、割り当てられたリソースを変更することができます。

一般的な XML の用語

このドキュメンテーションで使用される XML に関する用語を以下に示します:

スキーマ *(英数文字 の を伴う)*スキーマは、一般的なスキーマを指します。StyleVision では(*大文字で表示される*) XML スキーマとDTD がサポートされます。

XML スキーマ XML スキーマ このドキュメンテーションにある(*大文字で表示される*) XML スキーマ とは、W3C の XML スキーマ仕様に準拠したスキーマのことを表します。XML スキーマは(*小文字で表示される*) 全スキーマ のサブセットとなります

URI とURL このドキュメンテーションでは、「ロケーター」としてだけ使用される識別子のことを表し、http スキーマを使用する識別子に対しても、更に適用範囲の広いURI が使用されます。

XSLT とXPath の用語

XSLT 1.0 とXPath 1.0 からXSLT 2.0 ならびにXPath 2.0 にかけて、幾つかの用語が変更されました。例えば、XPath 1.0 でルートノードと呼ばれていたものが、XPath 2.0 ではドキュメントノード と名前が変更されています。このドキュメンテーションでは、XSLT 2.0 ならびにXPath 2.0 に準拠した用語が使用されます。

絶対 XPath コンテキストノードを含んでいるツリーのルートノードから始まるパスの条件式です。StyleVision のダイアログにてパス条件式を入力する場合、絶対 XPath チェックボックスにチェックを入れることで絶対パスとして条件式を入力することができます。このチェックボックスのチェックが外されている場合、コンテキストノードに対して相対的なパスが入力されます。

コンテキストアイテム／コンテキストノード コンテキストアイテムとは、条件式が評価されるアイテム(ノードまたは文字列の値)に相対的なアイテムとなります。コンテキストノードとは、ノードのコンテキストアイテムのことを指します。コンテキストアイテムは条件式内部でも、例えば各ステップの位置やフィルター条件式の中で変更することができます。

カレントノード カレントノードとは現在処理されているノードのことです。サブ式を持たない条件式の場合、カレントノードはコンテキストノード と等価になります。サブ式が存在する条件式の場合ではコンテキストノード が変わることもあります。current()

(C) 2015-2021 Altova GmbH

関数はXSLT 関数で、XPath 関数では無い点に注意してください。

ドキュメント要素 整形式のXMLドキュメントでは、最も外側に位置する要素がドキュメント要素と呼ばれます。ドキュメント要素はドキュメントノード の子となり、整形式のXMLドキュメント内にはドキュメント要素は1つだけ存在することになります。GUI において、ドキュメント要素はルート要素と表示されます。

ドキュメントノード ドキュメントノードにはドキュメント全体が含まれ、ドキュメントのツリー構造におけるルートノードとなります。ルートノードはXPath 条件式では'/' にて表されます。StyleVision のスキーマツリーウィンドウでは、「/ ルート要素」として表示されます。

StyleVision-でのみ使用される用語

StyleVision のメカニズム、コンセプト、コンポーネントを表す用語を以下に示します:

青写真イメージ 青写真イメージは、レイアウトコンテナーの背景イメージとして使用されるもので、通常フォームをスキャンしたものになります。青写真イメージをベースにSPS のデザインを行うことができ、フォームデザインの再度作成を行うことができます。.

動的アイテム XML データソース内から得られたアイテムのことです。動的アイテムはテキスト、テーブル、リスト、そしてイメージや(URI が動的の場合) ハイパーリンクとして作成することができます。

グローバル要素 スキーマツリーウィンドウ内にあるグローバル要素リストに表示される要素のことです。XML スキーマにてグローバル要素として定義された全ての要素は、グローバル要素リストにて表示されます。DTD において全ての要素はグローバル要素となるため、グローバル要素のリストにて表示されます。グローバルテンプレート は、グローバル要素に対してのみ定義されます。

グローバルテンプレート グローバルテンプレートはグローバル要素 に対して定義することができます。グローバルテンプレートが一旦定義されると、ドキュメント内に出現する要素がどこにあろうと、その要素に対してグローバルテンプレートが使用されます。グローバルテンプレートの代わりに、ローカルテンプレートにてグローバル要素を処理することができます。

レイアウトコンテナー レイアウトコンテナーとは、デザイン要素を絶対位置により配置することができるデザインブロックのことです。デザインがフォームをベースとしている場合、レイアウトコンテナーとしてデザインを作成することができ、フォームの要素を絶対位置により配置することができます。それ以外の場合、デザインをフリーハンドで作成し、ドキュメント内にレイアウトコンテナーを配置することができます。

ローカルテンプレート ローカルテンプレートによりメインのテンプレート内にある(グローバルおよび非グローバルの) 要素がどのように処理されるかが定義されます。ローカルテンプレートは、メインのテンプレート内にある特定の要素出現に対して適用されます。ローカルテンプレートとは違い、グローバルテンプレート は、メインのテンプレートにおける要素の全出現に対して適用されます。

メインスキーマ 割り当てられたスキーマソースの１つがメインスキーマとして使用されます。作業用 XML ファイルのドキュメントノードがメインスキーマと関連付けられ、メインのテンプレートの開始点として使用することができます。

メインのテンプレート メインとなるエントリーポイントです。StyleVision において、このテンプレートはドキュメント要素 にマッチして、XSLT プロセッサーにより最初に評価されるものとなります。スキーマツリーウィンドウでは、ドキュメントノードの子として表示されます。メインのテンプレート により、基本的なドキュメント構造が定義されるほか、入力ドキュメントがどのように処理されるかが定義されます。その他にも、ローカルテンプレート を含めることができ、グローバルテンプレートへの参照を行うことができます。

出力 XSLT スタイルシートを伴うXMLドキュメントを処理することで生成された出力のことです。StyleVision により生成することのできる出力ファイルはwould be HTML と RTF 形式となります。 Authentic View は出力とは考えられておらず、StyleVision として独立して記述されます。StyleVision により生成されたXSLT スタイルシートも出力とは考えられておらず、XSLT スタイルシートとして独立して記述されます。

静的アイテム XML データソースではなく、SPS から得られたアイテムのことです。静的アイテムにはテキスト、テーブル、リストのほか、(URI が静的な) イメージやハイパーリンク、ブックマークがあります)。

SPS コンポーネント SPS コンポーネントは、(i) スキーマノード(例えば要素ノード)、(ii) 自動計算 やテキスト文字列といった静的なSPS コンポーネント、または(iii) SPS にて開始ならびに終了タブとして表示される定義済みのフォーマット のどれか

	になります。
テンプレート	ノードやノードのグループを処理するための処理セットとして考えることができます。
テンプレート XML ファイル	テンプレート XML ファイルはStyleVision（Enterprise ならびにProfessional Edition）内のSPS へ関連付けられます。SPS がAuthentic View にて開かれた際に、新たに作成されるSPS を伴ったXML ドキュメントの開始データとして テンプレート XML ファイルが使用されます。テンプレート XML ファイルはSPS がベースとしているスキーマに対して妥当でなければなりません。
ユーザー定義要素	スキーマツリーにてノードでもなければ、定義済みの要素やデザイン要素でもない要素で、ユーザーから指定された要素のことです。属性により要素の指定を行うことができます。
ユーザー定義テンプレート	XPath 条件式にて指定されたシーケンスに対して作成されるテンプレートです。
ユーザー定義 XML テキストブロック	デザイン内の任意の場所にXML テキストブロックを挿入することができます。
作業用 XML/XBRL ファイル	作業 XML/XBRL ファイルはStyleVision 内のSPS に関連付けられたXML データファイルで、StyleVision にてXML ドキュメントの出力やAuthentic View のプレビューを行うために利用されます。作業用 XML/XBRL ファイルを使用しない場合、StyleVision 内のSPS にて動的な XML データが処理されることはありません。2つ以上のグローバル要素が含まれるスキーマをSPS がベースにしている場合、どのグローバル要素がドキュメント要素になるのかが曖昧になってしまいます。作業用 XML/XBRL ファイルを割り当てることで、そのような曖昧性が解決されます（妥当な XML ドキュメントにはドキュメント要素が1つしか含まれないため）。XBRL の機能はEnterprise Edition でのみ利用できることに注意してください。
XML ドキュメント	XML ドキュメント以下のために使用されます: (i) 特定のXML ドキュメント表す、(ii) （XML データドキュメントをSPS で使用するためのデータが含まれるDB ソースを含むXML データソースを表すのに使用されます。

関連項目

- SPS とは?
- 一般的な使用方法

(C) 2015-2021 Altova GmbH

1.7 このドキュメントについて

このドキュメンテーションはStyleVision に同封されているユーザーマニュアルで、StyleVision に内蔵されているヘルプシステムとしてご利用になれるほか、Altova W eb サイトや、印刷用のPDF ファイルをダウンロードすることもできます。

ユーザーマニュアルは、以下のようなセクションにより構成されています:

- はじめに: SPS の説明や、StyleVision の主な機能やコンセプトを紹介します。
- ユーザーインターフェイスの説明: StyleVision GUI の概要を記します。
- チュートリアル:StyleVision に搭載されている機能の使用方法について説明します。
- 使用方法の概要 : SPS の作成に使用されるスキーマソースや、デザイン処理の概要、Authentic View のデプロイやプロジェクトといった、高度な機能の使用方法について記述します。
- SPS ファイルコンテンツ : 静的(スタイルシートベース)かつ動的(XML ドキュメントベース)なコンポーネントの作成方法と、SPS で編集する方法について説明します。
- SPS ファイル構造: SPS ファイルがどのような構造を持ち、どのようにモジュール化されているか説明し、StyleVision のテンプレート処理について記述します。
- SPS ファイルの高度な機能: 計算結果の自動生成や、条件の設定、ユーザー定義された条件のグループ化や並べ替え、そして出力ドキュメントにて目次や相互参照を作成するための方法について説明します。
- SPS ファイルプレゼンテーション: SPS コンポーネントのフォーマットや、レイアウトについて説明します。
- SPS ファイル追加編集機能: SPS を更にパワフルに使用するための機能について説明します。その他のAltova 製品の機能を利用するグローバルリソースや、更なる検証、変数やパラメーターなどについて記述されます。
- SPS ファイルとデータベース: データベースを SPS で使用する方法について記述します。
- Authentic View: XML ドキュメントを Authentic View にて編集する方法について記述します。StyleVision GUI には Authentic View プレビュータブが搭載されており、Authentic View への出力を直ちに確認することができます。
- StyleVisionの統合機能 : StyleVision を他のアプリケーションへ統合する方法について記述します。StyleVision を Visual Studio と Eclipse で使用する方法についても説明します。
- レファレンス :StyleVision で使用されている全てのシンボルやコマンドに関する記述が収められます。
- 付録:Altova XSLT エンジンの情報、DB データ型の XML スキーマ データ型への変換、StyleVision の技術情報、そしてライセンスに関する情報が収められます。

使用方法

まずはStyleVision の概要を知るために、はじめに、ユーザーインターフェイスと使用方法の概要 セクションに関するセクションへ目を通すことをお勧めします。その後、チュートリアルをこなすことで、SPS の作成方法について実際に確認することができます。SPS ファイルのセクション(SPS ファイルコンテンツ、SPS ファイル構造、SPS ファイルの高度な機能、SPS ファイルプレゼンテーション、SPS ファイル追加機能、SPS ファイルとデータベース)では、StyleVision にて用意されている様々な機能の使用方法について確認することができます。レファレンスセクションには全てのツールバーアイコン、デザインシンボル、そしてメニューコマンドが、ツールバーとメニューごとに記述されており、コマンドの確認などを行うために使用することができます。 Authentic View セクションでは、それぞれAuthentic View における編集と、コマンドラインからStyleVision を呼び出す方法について記述されます。

Windows 7、 Windows 8 およびWindows 10 でのファイルパス

このドキュメントのファイルパスはすべてのオペレーティングシステムで同じではありません。以下の通信に注意してください:

- *(マイ) ドキュメント フォルダー:* デフォルトでは以下の場所に存在します。Example ファイルはこのフォルダのサブフォルダに存在します。

Windows 7/8/10	C:\Users\<username>\Documents

- *アプリケーション フォルダー:* アプリケーション フォルダーにはAltova アプリケーションが存在します。デフォルトでの、アプリケーション フォルダのパスは以下の通りです。

Windows 7/8/10	C:\Program Files\Altova\
64-bit OS 上での32 bit バージョン	C:\Program Files (x86)\Altova\

メモ StyleVision は、プラットフォーム更新済みのW indows Server 2008 R2 SP1 または以降 でもサポートされています。

サポート オプション

StyleVision、に関する質問や問題がある場合、以下のサポートオプションがご利用になれます:

1. ヘルプ ファイル(このドキュメンテーション)の確認。ヘルプファイルにはキーワードによる検索の他に、全文検索機能が搭載されています。
2. Altova Web サイト の良くある質問 とディスカッションフォーラム を確認してください。
3. Altovaサポートセンターに連絡してください。

頻繁に使用される略語

このドキュメンテーションでは、以下の省略形が頻繁に使用されます:

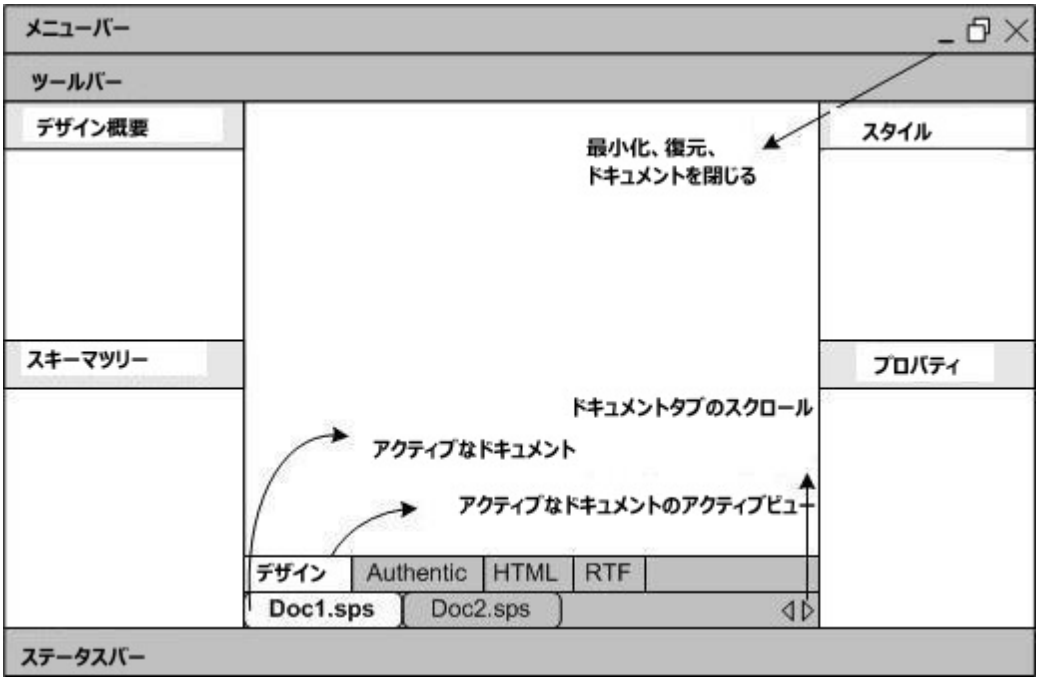
- **SPS**: StyleVision Power Stylesheet
- **DB**: データベース
- **CSS**: カスケーディングスタイルシート
- **FAQ**: 頻繁に寄せられる質問

(C) 2015-2021 Altova GmbH

2 ユーザーインターフェイス

StyleVision GUI は、以下のパーツから構成されます(*下に表示されているユーザーインターフェイスにはすべてのサイドバーは表示されていません*) :

- **メニューバー**: メニューをクリックすることで、そのメニュー内にあるアイテムが表示されます。メニューアイテムについては ユーザーリファレンス メニューバーには最小化、最大化、ドキュメントを閉じるためのボタンが表示されます。
- **ツールバー領域**: 様々な ツールバー ならびにそれらツールバーに対応したコマンドショートカットについては ユーザーリファレンス にて記述されます。
- タブ化された メインウィンドウ, その時開かれている SPS ドキュメントが表示されます。このウィンドウでは SPS デザインの編集 Authentic View のコンテンツを(Authentic eForm タブにて)編集、そして XSLT スタイルシート と出力のプレビューを行うことができます。
- デザインサイドバー: デザイン概要、スキーマツリー、デザインツリー、スタイルレポジトリ、スタイル、プロパティとプロジェクト ウィンドウが表示され、アプリケーションウィンドウへドックすることができるほか、ウィンドウとして表示することもできます。
- **ステータスバー**: アプリケーションのステータス情報が表示されます。64ビットバージョンの StyleVision を使用している場合、アプリケーション名の後に (x64) という文字が文字列されます。32ビットバージョンでこのような文字列は表示されません。



メインウィンドウ と デザインサイドバー の詳細については、このセクションのサブセクションにて記述されます。

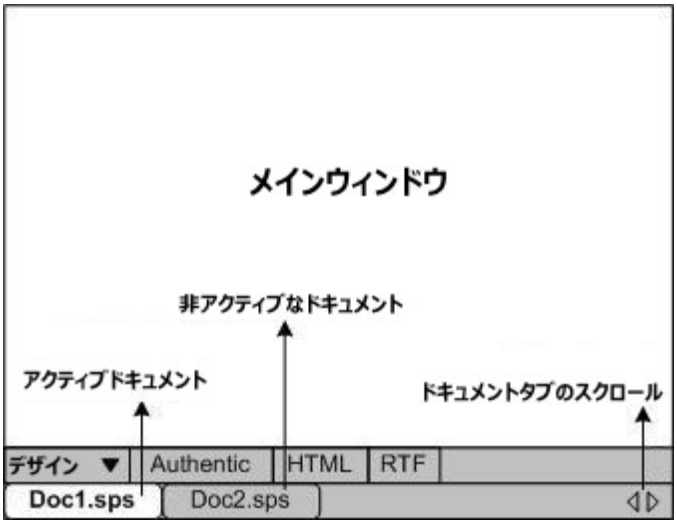
メモ メニューバーならびにツールバーのタイトル部分をドラッグすることで、目的の場所へ移動することができます。

関連項目

- メニューコマンド セクション
- レファレンス セクション | ツールバー

2.1 メインウィンドウ

メインウィンドウにはSPS デザイン、Authentic View 、XSLT スタイルシート、そして出力プレビューが表示されます（以下のイメージを参照）。



メインウィンドウ内のSPS ドキュメント

- StyleVision では複数の SPS ドキュメントを開くことができますが、同時に選択できるアクティブなドキュメントは1つだけです。開かれているドキュメントの名前は、メインウィンドウ下部にあるタブに表示され、アクティブになっているドキュメントのタブがハイライトされたかたちで表示されます。
- タブをクリックして、開かれているドキュメントをアクティブにすることができます。その他にも、ウィンドウメニューにあるオプションを使用することができます。
- 同時に開かれているドキュメントの数が多く、全てのドキュメントタブが表示されない場合、（ドキュメントタブバーの右側にある）スクロールボタンを使用することで、タブのスクロールを行うことができます。
- アクティブなドキュメントを閉じるには、アプリケーションウィンドウの右上にあるメニューバーの、**ドキュメントを閉じる**ボタンをクリックします（または「ファイル| 閉じる」を選択します）。

ドキュメントビュー

以下にあるビューにてドキュメントは表示されます。ビューは同時に1つしか選択することができません。

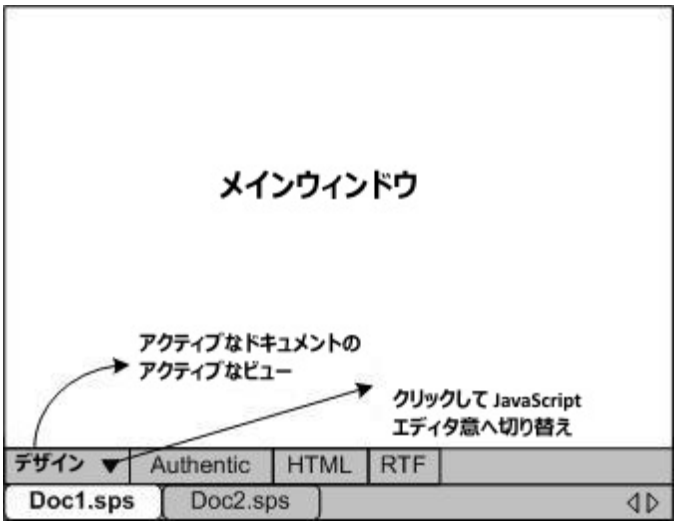
- デザインビュー: SPS のデザインと、SPS で使用するJavaScript 関数の編集を行うことができます。ドロップダウンメニューの矢印をクリックして、デザイン、またはJavaScript を選択することで、ドキュメントのデザインとJavaScript エディターを切り替えることができます。
- Authentic View（Authentic eForm タブ）: 出力 XML ドキュメントをAuthentic View にて直ちに確認することができます（ 作業用 XML ファイルを参照ください）。SPS は動的に作業用 XML ファイルへ適用されるため、Authentic View の機能を試験することができます。
- 出力ビュー（HTML とRTF 出力）: それぞれのフォーマットにて出力されるファイルのプレビューと、出力を生成するために使用されるXSLT スタイルシートのプレビューが表示されます。ドロップダウンメニューの矢印をクリックして、目的の選択を行うことで、出力のプレビューとXSLT スタイルシートの表示を切り替えることができます。

上にあるビューは、メインウィンドウ下部に表示されるタブをクリックすることで切り替えることができます。選択されたビューのタブがハイライトされます。

(C) 2015-2021 Altova GmbH

2.1.1 デザインビュー

SPS のデザインはデザインビューにて行います。デザインビューでは、(i) サイドバー、キーボードからの入力、メニューやツールバーから選択することができるコンテンツ作成ならびに編集機能を使用、または(ii) サイドバーとメニューから選択することができる様々なフォーマット機能を使用することで出力ドキュメントのデザインを作成します。デザインビューに搭載されたこれらの機能について以下で説明します。



デザインビューは、JavaScript エディターまたは(Authentic) スクリプトエディターへ切り替えることもできます。JavaScript エディターでは、SPS にて使用することができる JavaScript 関数 を作成することができます。スクリプトエディターでは、Authentic View に対応したスクリプトを作成、編集することができます。JavaScript エディターや(Authentic) スクリプトエディターへ切り替えを行うには、デザインタブ内にあるドロップダウンボタンをクリックしてドロップダウンメニューから JavaScript または(Authentic) スクリプトエディターを選択します。デザインビューへ切り替え直すには、JavaScript または(Authentic) スクリプトエディターにあるドロップダウンボタンをクリックして、ドロップダウンメニューからデザインを選択します。

デザインビューでは、 メインテンプレート、グローバルテンプレート、ページレイアウトテンプレート、, ページレイアウト テンプレート そしてデザインフラグメントといった複数のテンプレートを SPS 内に収めることができます。ツールバーアイコンとして表示されているテンプレート デザイン フィルターを使用することで、どのテンプレートをデザインビューにて表示するか指定することができます。表示フィルターを使用することで、SPS の最適化と、SPS 間の表示を切り替えることができます。

マークアップタグの表示

マークアップアイコンを使用することで、デザインビューにあるマークアップタグの表示を切り替えることができます(以下を参照)。

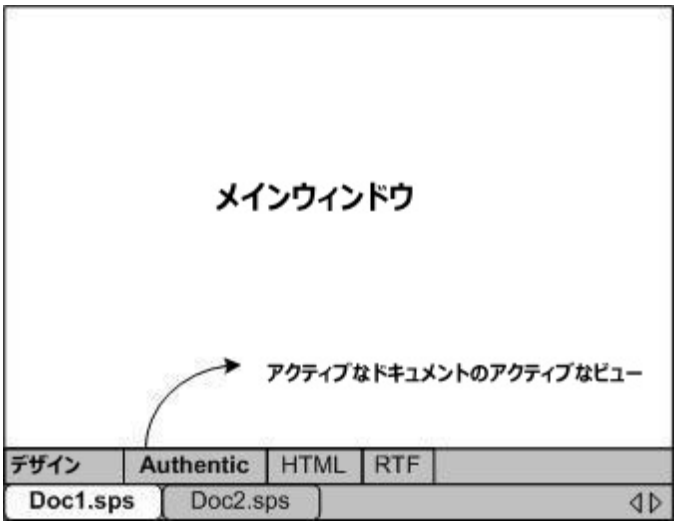
上にあるアイコンのうちどれかを選択することができます。左から、(i) 小さなデザインマークアップ(名前なしのタグ)を表示、(ii) 大きなデザインマークアップ(名前付きタグ)の表示を表します。小さなマークアップが選択された場合、ノードへマウスを移動させた時に、ノードへのパスが表示されます。

2.1.2 Authentic View

メインウィンドウ(Authentic eForm タブ)の**Authentic View** タブでは、Authentic View にて作業用 XML ファイルの閲覧と編集を行うことができます。このビューでは、(i) Authentic XML ドキュメントがどのように表示されるかを確認、そして(ii) SPS により作成された

Authentic View の試験を行うことができます。この機能は、Authentic View の動的な機能を試験するのに特に重要なものとなります。例えば、以下のような状況で、Authentic View の動作を試験することができます:

- 新たな要素や属性を追加する
- 新たな段落やテーブル行を追加する
- 条件テンプレートが影響を受ける値を変更する



Authentic View と作業用 XML ファイル

Authentic View を表示するには、作業用 XML ファイルをアクティブな SPS ドキュメントへ 割り当てる必要があります。作業用 XML ファイルは、SPS がベースにしているスキーマに対して妥当となっている必要があります。

StyleVision により作業用 XML ファイルをベースにした一時的な XML ファイルが作成され、この一時的なファイルがメインウィンドウの Authentic View タブに表示されます。Authentic View にて作成された変更点は一時的な XML ファイルに対して反映されます。(「ファイル | Authentic XML データを保存する」メニューコマンドにより) 明示的に変更点を保存しない限り、作業用 XML ファイルそのものは変更されません。

作業用 XML ファイルが割り当てられていない場合、Authentic View タブをクリックした際に、作業用 XML ファイルを割り当てるように促されます。

Authentic View の制限

メインウィンドウ内の Authentic View は、以下の点を除き XMLSpy ならびに Authentic Desktop にて利用することができる Authentic View と同等の機能を有しています:

- Authentic View 入力ヘルパーを GUI にて利用できません。ノードの挿入や追加を行うには、右クリックしてコンテキストメニューを使用します。
- CALS/HTML テーブルを挿入することができません。
- テキスト状態アイコンを利用することができません。

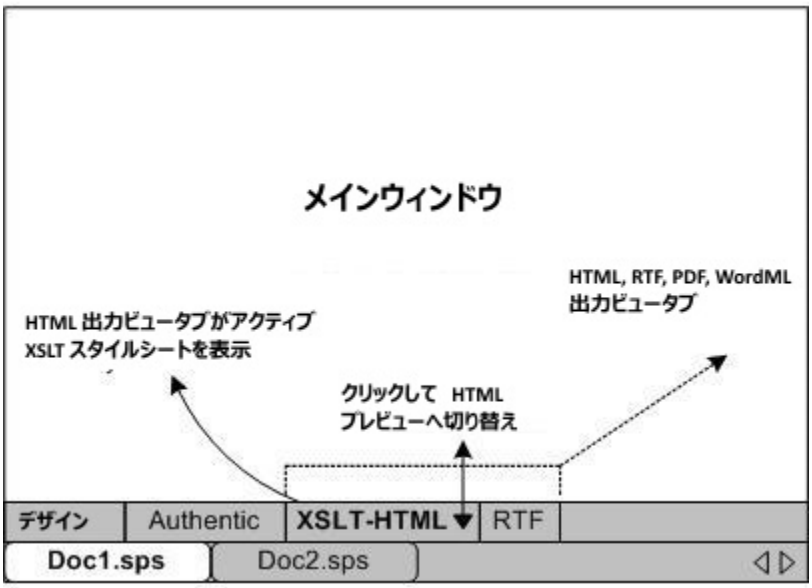
これらの機能をテストするには、XMLSpy または Authentic Desktop に搭載されている Authentic View を使用してください。Authentic View を使用する方法に関する詳細は、Authentic View 内での編集を参照ください。それ以外の情報については、XMLSpy または Authentic Desktop ユーザーマニュアルに含まれている Authentic View チュートリアルを参照ください。

2.1.3 出力ビュー

それぞれ2つの**出力ビュー**タブ搭載されており、HTMLとRTF 出力に対応しています。それぞれの出力ビュータブ（*以下のイメージを参照*）は、以下を表示します: (i) その出力に対応したXSLT スタイルシート、(ii) XSLT スタイルシートにより作業用 XML ファイルを変換することで得られた出力のプレビューが表示されます。

メモ　異なるビューが正確に作動するには、StyleVision のセットアップのセクション説明される通りに各ビューに関連するプログラムをインストールする必要があります。.

出力ビュータブでは、XSLT スタイルシートと出力プレビューの切り替えを出力ビュータブのドロップダウンボタンから行うことができ、目的に応じてXSLT オプションまたは出力プレビューオプションを選択することができます。



XSLT ビュー

ビューには、現在アクティブな SPS により出力フォーマットに対して生成された XSLT スタイルシートが表示されます。XSLT ビューが選択されるたびにスタイルシートは新たに生成されます。

出力ビュータブに表示されるスタイルシートには、行番号や展開/縮退アイコンも表示されます。左側の余白にある + ならびに − アイコンをクリックすることで、要素の展開/縮退を行うことができます。XSLT ビューに表示されたスタイルシートを編集することはできませんが、テキストの検索を行い（「編集 | 検索」を選択）、クリップボードへのコピーを行うことはできます（「編集 | コピー」を使用）。

メモ　SPS に対して使用して得られた出力が表示されます。「ファイル | 生成されたファイルの保存」コマンドを使用することで、個別に生成、保存することができます。

出力 プレビュー

出力 では、作業用 XML ファイルが出力フォーマットに対して生成されたXSLT スタイルシートが表示されます。XSLT-for-HTMLに対して使用して得られた出力が表示されます。出力 プレビュータブ がクリックされるたびに、新たに作成されます。作成されるのは保存バージョンの作業用 XML ファイルで、— Authentic Viewにて編集される一時的なファイルではないことに注意してください。つまり、Authentic View にて作成された変更が出力プレビューに反映されるのは、これらの変更点が作業用 XML ファイルへ保存された後になります（「ファイル | Authentic XML データを保存する」）。

作業用 XML ファイルが割り当てられていない状態で、出力ビュー表示タブにて出力プレビューが表示された場合作業用 XML ファイルを割り当てるように促されます。DB をベースにした SPS の場合、DB がロードされ、そこから一時的な(編集不可能な)XML ファイルが生成され、作業用 XML ファイルとして使用されるため、作業用 XML ファイルを割り当てる必要はありません。

メモ SPS から生成された出力ファイルは、個別に生成され、「「ファイル」生成されたファイルの保存」コマンドを使用することで、ファイルの保存を行うことができます。

関連項目

- StyleVision のセットアップ
- 生成されたファイルの保存
- 自動化された処理

(C) 2015-2021 Altova GmbH

2.2 サイドバー

サイドバー（またはサイドバーウィンドウ）はSPS のデザインを行うためのGUI コンポーネントで、アクティブなビューに関する情報を提供します。以下に示される各サイドバーについては、このセクションのサブセクションにて解説されます。

- デザイン概要
- スキーマツリー
- デザインツリー
- スタイルレポジトリ
- スタイル
- プロパティ
- プロジェクト
- メッセージ
- 検索と置換

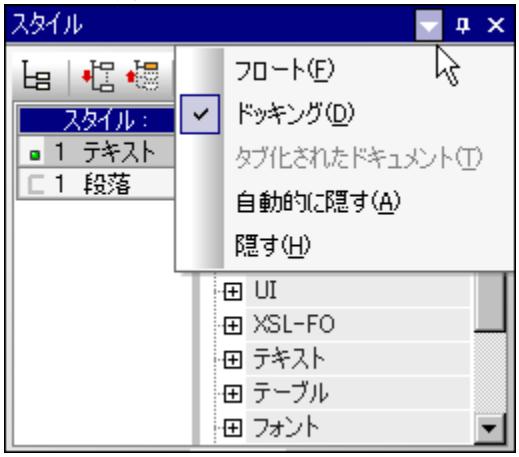
ビューのレイアウト

ビューのレイアウトとは、そのビューでどのサイドバーが利用可能なのかと、これらサイドバーがGUI にてどのように配置されているかのことを指します。レイアウトは個々のビューカテゴリーに対してカスタマイズすることができ、カスタマイズは、(i) ビューにおける個々のサイドバーの表示（表示メニュー、またはサイドバーのタイトルバーを右クリックして、「隠す」を選択する）と、(ii) 必要に応じてGUI にてサイドバーを配置するという2つから構成されます。この方法により定義されたレイアウトは、そのビューカテゴリーに対して保持されます。例えば、デザインビューにてスタイルサイドバーを除く全てのサイドバーが表示されていた場合、次回デザインビューレイアウトが変更されるまで、このレイアウトがデザインビューに対して保持されます。出力プレビュー（HTML、RTF、PDF、Word 出力プレビュー（HTML とRTF プレビュー）に対して定義されたレイアウトは、全ての出力プレビューに対して適用されることに留意してください。ビューカテゴリーは以下のとおりです: (i) ドキュメントが開かれていない、(ii) デザインビュー、(iii) 出力ビューと（iv) Authentic View.

サイドバーウィンドウのドッキングとフロート

サイドバーウィンドウはStyleVision GUI に対してドックするか、画面上にフロート状態にすることができます。ウィンドウをドックするにはタイトルバーをドラッグし、ドラッグする際に表示される矢印にドラッグしてください。内側に表示される矢印へドロップすることで、矢印が表示されたウィンドウの内側に対してドラッグされたウィンドウを配置することができます。外側にある矢印へウィンドウをドロップすることで、矢印の表示されているウィンドウの縁にドッキングが行われます。ウィンドウをフロート状態にするには、(i) タイトルバーをダブルクリックするか、(ii) タイトルバーをドラッグして、画面上に表示される矢印以外の場所でドロップしてください。

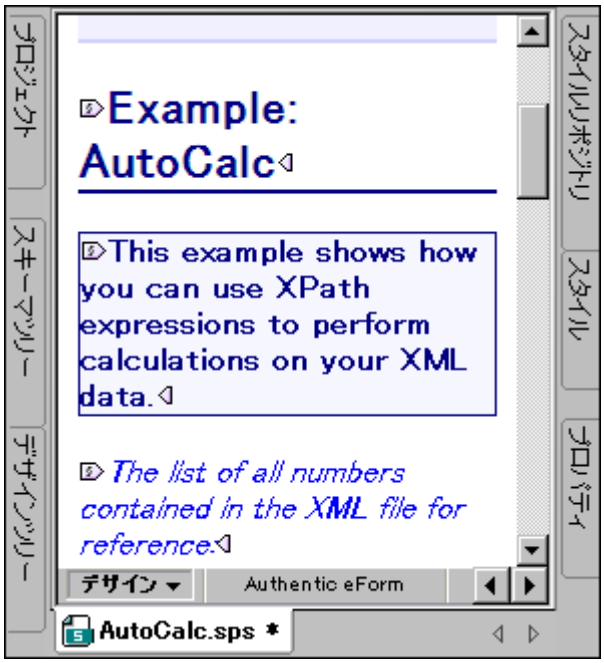
その他にも、以下のメカニズムを使用することができます。ドックされたウィンドウをフロート状態にするには、ドックされたウィンドウの右上にあるメニューボタンをクリックして、フロートを選択します。このメニューへは、ドックされたウィンドウのタイトルバーを右クリックすることでもアクセスすることができます。



フロート状態のウィンドウをドッキングするには、そのウィンドウのタイトルバーを右クリックして、表示されるメニューからドッキングを選択します。最後にドッキングが行われた場所にウィンドウがドッキングされます。

デザインサイドバーウィンドウを自動的に隠す

ドックされたウィンドウは自動的に隠すこともできます。サイドバーウィンドウが自動的に隠れるように設定されている場合、GUI の端にタブというかたちで最小化されます。以下のスクリーンショットでは、6個あるサイドバーが自動的に隠された状態になっています。



マウスカーソルをタブ上に配置することで、メインウィンドウにウィンドウが表示されます。以下にあるスクリーンショットでは、スタイルタブの上にマウスカーソルを配置することで、スタイルサイドバーがメインウィンドウ上に表示されます。

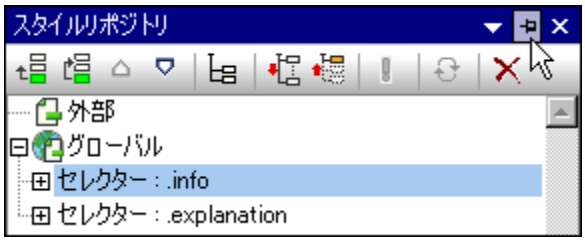
マウスカーソルを表示されたウィンドウの外側へ、またはタブとは別の場所に移動させると、そのウィンドウがタブとして GUI の端に再度収まります。

自動的に隠す機能は、頻繁には使用しないサイドバーを GUI に表示させず、必要な時にだけアクセスするような場合に便利な機能となります。この機能により、デザインサイドバーウィンドウへ簡単にアクセスすることができるとともに、メインウィンドウで更なるスペースを使えるようになります。

ウィンドウを自動的に隠すためには、ドッキングされたウィンドウの右上にある自動的に隠すボタン(ピンのかたちをしたアイコン)をクリックします。それ以外にも、ウィンドウのタイトルバーを右クリックするか、ドッキングされたウィンドウのタイトルバーにあるメニューボタンをクリックすることで表示されるメニューから、自動的に隠すオプションを選択して、この機能を有効にすることができます。

ウィンドウが自動的に隠されるようになります。

特定のウィンドウに対して自動的に隠す機能を無効にするには、マウスカーソルをそのタブへ移動してウィンドウが表示された状態で、自動的に隠すボタンをクリックします。それ以外にも、ウィンドウのタイトルバーを右クリックするか、ドッキングされたウィンドウのタイトルバーにあるメニューボタンをクリックすることで表示されるメニューから、自動的に隠すオプションの選択を無効にすることでも、この機能を無効にすることができます。



メモ　サイドバーウィンドウの自動的に隠す機能が無効になっている場合、そのウィンドウのピンアイコンが下に向かって表示され、有効になっている場合は、ピンの向きが左方向の状態で表示されます。

サイドバーウィンドウを隠す(閉じる)

サイドバーウィンドウが隠されると、GUI 上から直接確認することはできません。サイドバーウィンドウは最大化された状態(ドッキングまたはフロート状態)か、最小化された状態(自動的に非表示にする機能により GUI の端に表示されるタブとして)でのみ表示されます。

ウィンドウを隠すには、ドッキングまたはフロート状態にあるウィンドウの右上に表示されている**閉じる**ボタンをクリックします。それ以外にも、ウィンドウのタイトルバーを右クリックするか、ドッキングされたウィンドウのタイトルバーにあるメニューボタンをクリックすることで表示されるメニューから、**隠す**を選択することができます。

隠されている(閉じられている)状態のウィンドウを再度表示するには、**表示** メニューから目的のデザインサイドバーの名前を選択します。デザインサイドバーウィンドウが、隠された時の位置と状態(ドッキングまたはフロート状態)で再度表示されます。

関連項目

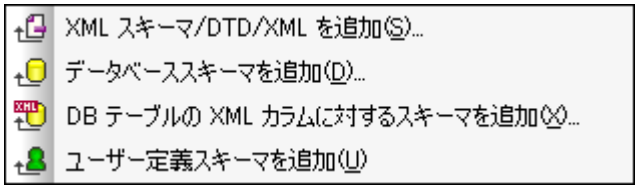
- デザインビュー
- 表示メニュー

2.2.1 デザイン概要

デザイン概要 サイドバーを使用することで、アクティブな SPS に対してスキーマソース、グローバルパラメーター、SPS モジュール、そして CSS ファイルを追加することができます。これらコンポーネントをまとめて表示することで、コンポーネントの集中管理を行うことができます。

スキーマソースの追加

空の SPS へ SPS を追加することができます。ソースのヘッダー以下にある**新しいソースの追加**コマンドをクリックすることで、スキーマソースを追加することができます。この機能により表示されるポップアップメニューにより、: (i) XML スキーマ、DTD、または StyleVision により XML ファイルから生成された XML スキーマ、(ii) により DB から生成されたスキーマ、または (iii) ユーザー定義スキーマ.



作業用 XML ファイルとテンプレート XML ファイル

スキーマが追加されると、そのスキーマがソースアイテム以下に表示されます。2 つのサブアイテムが存在します:

- 作業用 XML ファイル.
- テンプレート XML ファイル.

(C) 2015-2021 Altova GmbH

モジュール、CSS ファイル、パラメータ、XSLT ファイルの追加

モジュール、CSS ファイル、パラメーター、そしてXSLT ファイルセクションの下部にある、**新たに追加する**コマンドをクリックすることで、対応するセクションに新たなアイテムを追加することができます。

デザイン概要の機能

以下の機能は、デザイン概要サイドバーにある各セクション（ソースやパラメーターなど）全てに共通して使用することができます:

- 各セクション名の左側にある三角形の矢印をクリックすることで、そのセクションを展開、または縮退することができます。
- ソース、モジュール、CSS ファイルセクション内にあるファイルセクションにはファイル名だけが表示されます。ファイル名へマウスを移動すると、ファイルのフルパスがポップアップにて表示されます。
- 灰色で表示されているアイテムはインポートされたモジュールを表しており、GUI にてアクティブになっている SPS ファイル内には無いモジュールとなります。
- 各セクションの下部には**新しい <アイテム> を追加**コマンドが含まれており、そのセクションに対して新たなアイテムを追加することができます。例えば、**新しいパラメーターを追加**コマンドをクリックすることで、SPS とデザイン概要のパラメーターリストへ新たなパラメーターが追加されます。
- セクション内にある各アイテムにはコンテキストメニューが含まれており、そのアイテムを右クリックするか、**コンテキストメニュー**アイコン （アイテムの右側にある下向きのアイコン）をクリックすることで表示されます。
- コンテキストメニューに含まれている**削除**アイコンを選択することで、選択されたアイテムを削除することができます。このコマンドは、コンテキストメニューコマンドからも利用することができます。
- コンテキストメニューコマンドにある **XMLSpy でファイルを編集**を選択することで、Altova XMLSpy にてファイルを編集することができます。
- コンテキストメニューコマンドにある**上へ**ならびに**下へ**コマンドは、モジュールセクション内に 複数のモジュールが表示されており、そのうち1つのモジュールが選択されている際に有効になります。選択したコマンドにより、選択されているモジュールの位置を変更することができます。

ソース

ソースセクションのリストには、スキーマと、SPS をベースにしている作業用 XML ファイルとテンプレート XML ファイルが表示されます。（右クリック、またはコンテキストメニューアイコン をクリックすることで表示される）コンテキストメニューへアクセスし、適切な**割り当て...** オプションを選択することで、これらのファイル選択を変更することができます。

モジュール

モジュールセクションには、アクティブな SPS により使用されている SPS モジュールが表示されます。**新しいモジュールを追加**コマンドをクリックし、目的の SPS ファイルを選択することで、新たなモジュールをリストへ追加することができます。

モジュールが表示される順序 には意味があるため、複数のモジュールが表示されている場合、（モジュールのコンテキストメニューからアクセスすることができる）**上へ/下へ**コマンドによりモジュールの順序を変更することができます。コンテキストメニューには、StyleVision にて選択されたコマンドを開くためのコマンドも用意されています。

メモ　デザイン概要サイドバーではモジュールの概要を確認することができ、ファイルレベルでモジュールの管理を行うことができます。様々な モジュールオブジェクト （モジュール内部のオブジェクト）は デザインツリー サイドバー に表示されます。

CSS ファイル

CSS ファイルセクションには、アクティブな SPS により使用されている CSS ファイルが表示されます。**新しい CSS ファイルを追加**コマンドをクリックして、目的の CSS ファイルを選択することで、新たな CSS ファイルがリストへ追加されます。 CSS ファイルが表示される順序 には意味があるため、複数のファイルが表示されている場合、（コンテキストメニューからアクセスすることができる）**上へ/下へ**コマンドが選択可能になります。目的のコマンドをクリックすることで、選択された CSS ファイルを移動することができます。コンテキストメニューには、選択されたモジュールを XMLSpy にて開くためのコマンドも用意されています。

メモ デザイン概要サイドバーでは CSS ファイルの概要を確認することができ、ファイルレベルで CSS ファイルの管理を行うことができます。CSS ファイル内にある様々な CSS ルールは、スタイルレポジトリサイドバーに表示されます。

パラメータ

パラメーターセクションにはSPS 内にあるグローバルパラメーターが表示されます。セクションの下部にある**新たなパラメーターを追加**コマンドを使用することで、新たなパラメーターを追加することができます。パラメーター名や値をダブルクリックすることで、それぞれ名前や値を編集することができます。パラメーターを削除するには、目的のパラメーターを選択して、コンテキストメニューから**削除**コマンドをクリックする必要があります。

XSLT ファイル

XSLT ファイルセクションには、SPS 内にインポートされたXSLT ファイルが表示されます。これらXSLT ファイルに含まれているXSLT テンプレートは、グローバルテンプレートとしてスタイルシートにて利用可能になります。この機能の詳細については、: XSLT テンプレートを参照ください。

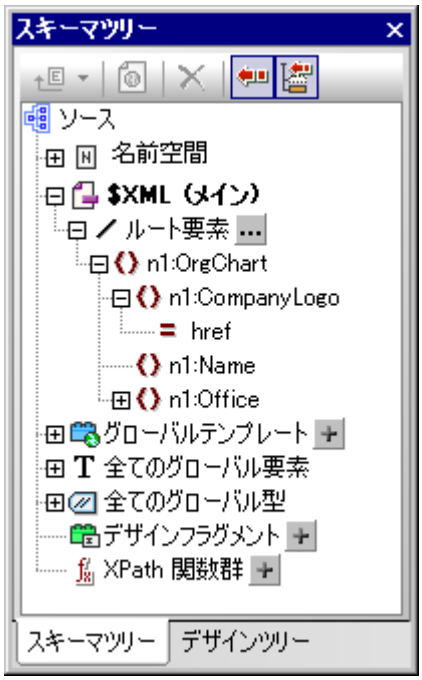
関連項目

- パラメータ
- スキーマツリー
- モジュールSPS
- デザインフラグメント
- スクリプトの使用
- 印刷出力のデザイン

2.2.2 スキーマツリー

スキーマツリーサイドバーでは、以下のような操作を行うことができます (*下のスクリーンショット*) :

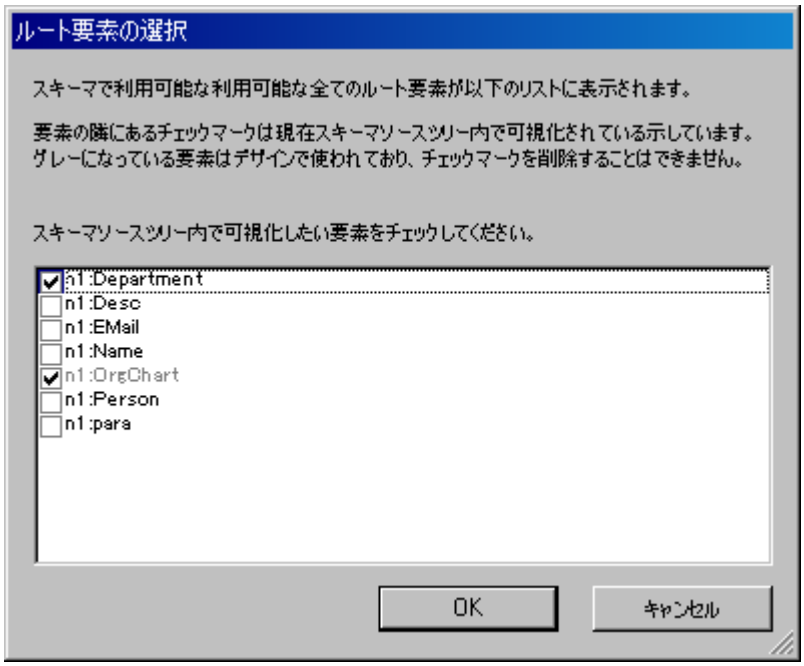
- スキーマに対して複数のルート要素(ドキュメント要素)を選択します。
- スキーマツリーからノード(要素、属性、グローバルタイプ)をドラッグして、デザインへドロップします。これらノードは、出力に含められる XML コンテンツを表します。
- スキーマソース内にあるグローバル要素ならびに型が表示されます。グローバル要素ならびにグローバル型をグローバルテンプレートとして作成することができます。
- SPS にて使用されている全ての名前空間を表示します。
- デザインフラグメントの挿入と編集を行います。
- SPS に対してユーザー定義された XPath 関数 の挿入と編集を行います。



ルート要素

$XML 以下にある各スキーマに対して、選択されたルート要素（またはドキュメント要素））が表示されます。このリストでは選択したスキーマに関連付けられている全てのルート要素が表示されます。各ルート要素を展開すると、モデルツリーが表示されます。これらルート要素内にあるノードから、メインテンプレートのコンテンツが作成されます。メインテンプレートのエントリーポイントはメインスキーマのドキュメントノードで、ドキュメントノードはいつでも変更することができます（その方法については以下を参照ください）。

スキーマに対してルート要素を選択するには、以下の操作を行います。ルート要素アイテムの右側にある選択 ボタンをクリックします。この操作により、ルート要素の選択ダイアログが表示され、スキーマ内に含まれているグローバル要素からルート要素を選択することができます。複数のルート要素を選択することで利用可能になるオプションの説明については、SPS 構造 | スキーマソースを参照ください。



更に、追加した全てのスキーマ内にある全てのグローバル要素 スキーマ が全てのグローバル要素以下に表示されます。各グローバル要素に対しては、グローバルテンプレート を作成することができます。

グローバル要素とグローバル型

グローバル要素ならびにグローバル型は、その他のテンプレートでも再利用することができるグローバルテンプレート を作成するために使用されます。更に、グローバル型はテンプレート内で直接使用することができます。

デザインフラグメント

ドキュメント内にある全てのデザインフラグメント はデザインフラグメントアイテム以下に表示され、デザインフラグメントアイテムを展開することで確認することができます。以下にあるデザインフラグメント機能を利用することができます:

- デザインフラグメントアイテムの追加アイコン をクリックすることで、デザインフラグメントを作成することができます。
- スキーマツリーにあるデザインフラグメントの名前をダブルクリックすることで、デザインフラグメントの名前を編集することができます。
- 各デザインフラグメントの横にあるチェックボックスを有効、または無効にすることで、対応するデザインフラグメントを有効、または無効にすることができます。
- デザインフラグメントをスキーマツリーからデザインをドラッグすることができます。

デザインフラグメントを扱うための情報については、デザインフラグメントのセクションを参照ください。

ユーザー定義 XPath 関数

XPath 関数アイテムの追加アイコン をクリックすることで、ユーザー定義 XPath 関数を追加することができます。XPath 関数を作成すると、その関数がスキーマツリー内に表示されます。XPath 関数をダブルクリックすることで、その関数がダイアログにて開かれ、編集を行うことができるようになります。以下のXPath 関数を利用することができます:

- XPath 関数アイテムの隣にあるチェックボックスにチェックを入れる、またはチェックを外すことで、XPath 関数を有効、または無効にすることができます。

- XPath 関数を右クリックすることでコンテキストメニューが表示され、XPath 関数の名前変更や削除を行うことができます。

XPath 関数の詳細な操作方法については、ユーザー定義 XPath 関数 のセクションを参照ください。

名前空間

SPS 内で使用されている名前空間のリストが、プレフィックスとともに名前空間以下に表示されます。リスト内にある名前空間は、(i) 参照されたスキーマや、スキーマ内部で定義されている名前空間、そして (ii) デフォルトで新たに作成された SPS に追加された名前空間となっています。このようなリストは、XPath 条件式を記述する際に便利な機能となります。更に、xpath-default-ns エントリーの値フィールドをダブルクリックして、名前空間の入力を行うことで、SPS 全体に対する XPath デフォルト名前空間を定義することができます。

メモ　SPS や、SPS から生成された XSLT スタイルシートに対して名前空間を追加する場合、SPS がベースとしている XML スキーマのトップレベルスキーマ要素へ、名前空間を追加する必要があります。

ツールバーとスキーマツリーアイコン

以下のツールバーアイコンは、一般的なスキーマツリーサイドバーコマンドに対するショートカットとなります。

ユーザー定義スキーマ にて、子要素をドキュメント要素に追加、または選択された要素に対して兄弟要素を追加します。アイコンのドロップダウン矢印をクリックすると ユーザー定義スキーマ を構築するその他のコマンドが表示されます。

グローバルテンプレートを作成/削除します。グローバル要素またはグローバル型が選択されている状態で選択可能になります。

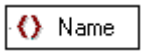
選択されたアイテムを削除します。

同期ツリーを有効にします。有効になっている場合（アイコンの境界線が表示されている場合）、ツリーにてノードを選択することで、(i) デザイン内の対応するノードが選択され、(ii) デザインツリーウィンドウ内にある同期ツリーアイコンが有効になっている場合、デザインツリー内にある対応ノードが選択されます。アイコンが無効になっている場合、デザイン内にある対応するノードが選択されることはありません。ツリーからノードをドラッグして、デザイン内にある目的の場所にドロップすることが難しい場合、このオプションを無効にしてください。

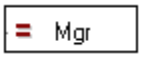
同期オプションが有効になっている場合、スキーマツリーにあるその他のアイテムが自動的に縮退され、デザインにてアイテムが選択されます。このオプションはツリーを同期オプションが有効になっている場合にのみ選択可能になります。

スキーマツリーで使用されるシンボル

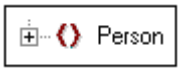
スキーマツリーにおけるシンボルを以下に示します。



要素

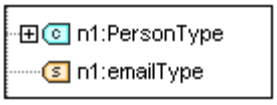


属性



子要素を含む要素。要素をダブルクリックし、左側にある +/- シンボルをクリックすることで、要素の展開/縮退を行うことができます。

DB のフィルタリングを行います。スキーマツリー内にあるトップレベルのデータテーブル要素に対してのみ適用されます。



グローバルの型は型を単純型または複合型となります。複合型はシアン色のアイコンにより表示され、単純型はブラウン色のアイコンにより表示されます。

関連項目

- SPS 構造の作成 | スキーマソース

2.2.3 デザインツリー

デザインツリーサイドバーには、SPS デザインの概要が表示されます。

デザインツリーのルートにはSPS の名前が表示され、マウスポインターをアイコン上に配置すると、ファイルの位置がポップアップで表示されます。デザインツリーにあるアイテムは、以下のカテゴリーに分類されます:

- スクリプト :StyleVision のJavaScript エディターによりSPS に対して定義された全てのJavaScript 関数が表示されます。
- メインのテンプレート: メインテンプレートの詳細な構造が表示されます。
- グローバルテンプレート :現在のSPS 内にあるグローバルテンプレートや、インクルードされた全てのSPS モジュールにあるグローバルテンプレートが表示されます。
- デザインフラグメント : デザイン内にある全てのデザインフラグメントが表示され、デザインフラグメントの作成、編集、名前変更、そして削除を行うことができます。
- XSLT テンプレート: インポートされたXSLT ファイル内にあるXSLT テンプレートが表示されます。
- ユーザー定義 XPath 関数: 独自のXPath 関数を作成、編集、名前の変更、削除することができます。

ツールバーアイコン

以下のツールバーアイコンは、スキーマツリーサイドバーにある一般的なコマンドのショートカットとなります。

デザインに対してデザインフラグメント、メインテンプレート、またはレイアウトアイテムを追加します。アイコンの左側をクリックすることで、デザインフラグメントが追加されます。ドロップダウンを表す矢印をクリックすることで、デザインフラグメントや、その他のレイアウトアイテムを追加するためのリストが表示されます。

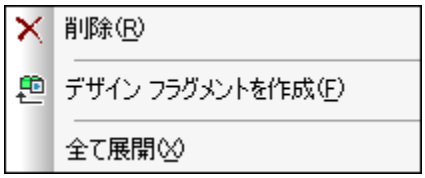
選択されたアイテムを削除します。グローバルテンプレートまたはレイアウトサブツリー内のアイテムが選択されている時に、アイコンが選択可能となります。

同期ツリーを切り替えます。有効になっている場合（アイコンの境界線が表示されている場合）、ツリー内のノードを選択することで、(i) デザイン内の対応するノードが選択され、(ii) デザインツリーウィンドウ内にある同期ツリーアイコンが有効になっている場合、デザインツリー内の対応するノードが選択されます。アイコンが無効になっている場合、デザイン内の対応するノードが選択されることはありません。

同期オプションが有効になっている場合、スキーマツリーにあるその他のアイテムが自動的に縮退され、デザインにてアイテムが選択されます。このオプションはツリーを同期オプションが有効になっている場合にだけ選択可能になります。

デザインツリーの表示を変更する

デザインツリー内にあるアイテムを右クリックすることで表示されるコンテキストメニューから、デザインツリーの表示を変更することができます（以下のスクリーンショットを参照）。



削除コマンドにより、選択されたデザインツリーアイテムをデザインツリーから削除します。**デザインフラグメントを作成**により、デザイン内にデザインフラグメント を作成し、そのアイテムをデザインツリーに追加します。**全て展開**コマンドにより、デザインツリー内にある全てのアイテムを展開することができます。

スクリプトとメインのテンプレート

スクリプトには、インポートされたモジュール内にあるものを含め、デザイン内にある全てのスクリプトが表示されます。メインテンプレートにはメインテンプレートのツリーが表示されます。アイテムを右クリックして、**削除**を選択することで、ツリー内にあるアイテムを削除することができます。

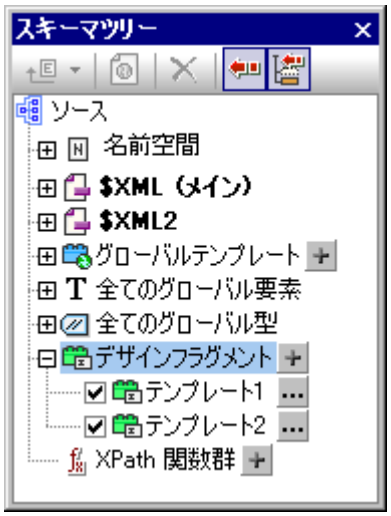
グローバルテンプレート

グローバルテンプレート アイテムには、現在のSPS ならびに追加された全てのSPS モジュール内にある全てのグローバルテンプレートが表示されます。現在のSPS にて定義されているグローバルテンプレートは黒色で表示され、追加されたモジュールにて定義されたグローバルテンプレートはグレーで表示されます。各グローバルテンプレートには、左側にチェックボックスが含まれており、有効/無効を選択することができます。グローバルテンプレートが無効化された場合、デザインから削除されます。

現在のSPS 内にあるグローバルテンプレートをクリックし、ツールバーにある**削除**ボタンをクリックするか、コンテキストメニューから**削除**コマンドを選択することで、そのグローバルテンプレートを削除することができます。デザインとツリーからコンポーネントが削除されます。

デザインフラグメント

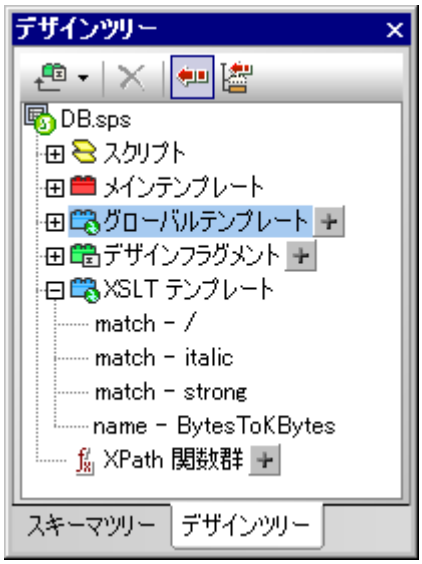
デザインフラグメント アイテムにより、現在のSPS ならびに追加されたSPS モジュール内にあるデザインフラグメントが表示されます。現在のSPS 内にあるデザインフラグメントは黒色で表示され、追加モジュール内で定義されたデザインフラグメントはグレーで表示されます。各デザインフラグメントには、左側にチェックボックスが表示されており、対応するデザインフラグメントを有効/無効にすることができます。現在のSPS 内にあるデザインフラグメントを選択し、削除コマンドをクリックするか、コンテキストメニューから**削除**コマンドを選択することで、そのデザインフラグメントを削除することができます。コンポーネントがデザインとツリーから削除されます。



デザインフラグメントアイテムの右側にある**追加**アイコン をクリックすることで、デザインフラグメントを追加することができます。各デザインフラグメントは展開/縮退可能なノードを備えたツリーとしてデザインされます。デザインフラグメントツリーを選択後に、ツールバーの**削除**ボタンをクリックするか、コンテキストメニューにある**削除**コマンドを選択することで、(現在のSPS にて定義された) 任意のデザインフラグメントツリーを削除することができます。デザインとツリーからコンポーネントが削除されます。

XSLT テンプレート

デザインツリーサイドバーにある XSLT テンプレート以下には、インポートされた XSLT ファイルが表示されます(以下のスクリーンショットを参照)。



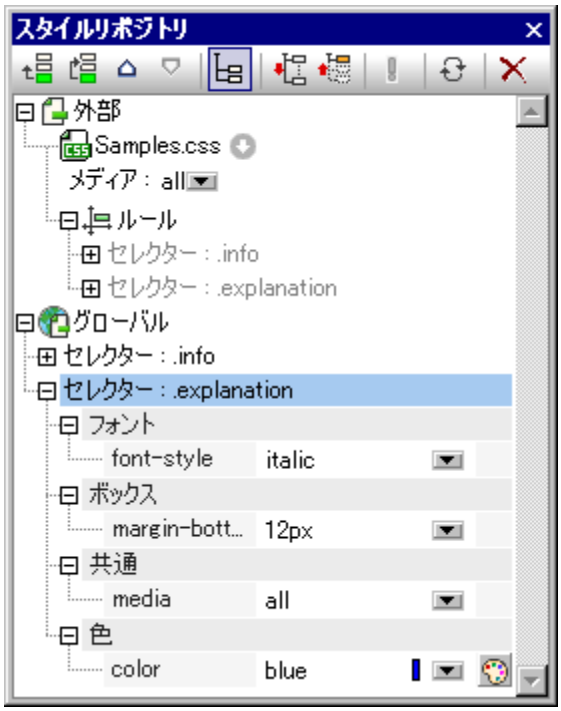
インポートされたXSLT テンプレートには2種類あります: (i) match テンプレート、そして(ii) named テンプレート。デザインツリーでは、これら2種類のテンプレートが、(i) match テンプレートにある select 属性の値、(ii) named テンプレートにある name 属性の値とともにそれぞれ表示されます。XSLT テンプレートに関する詳細については、XSLT テンプレートを参照ください。

関連項目

- デザインフラグメント
- スクリプトの使用

2.2.4 スタイルレポジトリ

スタイルリポジトリ サイドバーでは、外部 CSS スタイルシートの割り当てや、SPS に対してグローバルな CSS スタイルを定義することができます。外部 CSS スタイルシートにあるスタイルルールとグローバルに定義された CSS スタイルは、Authentic View とHTML 出力ドキュメントに対して適用されます。



スタイルリポジトリサイドバーには**グローバル**と**外部**という名前で、2つのツリーが表示されます。外部ツリーにはSPS と関連付けられている外部 CSS スタイルシートのリストが含まれます。グローバルツリーにはSPS と関連付けられた全てのグローバルスタイルが含まれます。

スタイルリポジトリの表示構造は以下のようになっています:

外部
- CSS-1.css（マウスオーバーにより表示されるポップアップにて位置を確認できます）
  - メディア（スタイルリポジトリウィンドウにて定義することができます）
  - ルール（編集不可 - CSS ファイルにて編集する必要があります）
    - セレクター-1
      - プロパティ-1
      - ...
      - プロパティ-N

- ...
- セレクター-N

+ ...
+ CSS-N.css

グローバル

- セレクター-1
  + セレクター-1 プロパティ
- ...
+ セレクター-N

スタイルルールの優先順位

グローバルスタイルルールと外部 CSS スタイルシートに記述されたスタイルルールの両方に、同じドキュメントコンポーネントを表すセレクターが指定されている場合、グローバルスタイルルールが、外部スタイルシートに優先して適用されます。複数のグローバルスタイルルールが同一のドキュメントコンポーネントを指定している場合、最後に記述されたルールが適用されます。同様に、複数の外部スタイルシート内にあるスタイルシートが同一のドキュメントコンポーネントを指定している場合、最後のスタイルシート内にある最後のルールが適用されます。

スタイルレポジトリにてスタイルを管理する

スタイルリポジトリサイドバーでは、ツールバー上にあるアイコンを使用する、またはコンテキストメニュー内にあるアイテムを使用することで、以下の操作を行うことができます:

追加: **追加** アイコン をクリックすると、選択されているツリー(外部またはグローバル)により、新たな外部スタイルシートエントリーを外部ツリーへ追加し、新たなグローバルスタイルエントリーをグローバルツリーへ追加することができます。新規エントリーが、ツリー内にて既に存在するリストへ追加されます。追加コマンドはコンテキストメニューからも選択することができます。外部スタイルシートならびにグローバルスタイルを使用するための詳細な情報については、CSS スタイルと作業 を参照ください。デザイン概要サイドバーからも外部 CSS スタイルシートの追加や削除を行うことができます。

挿入: 挿入アイコン により、(外部ツリー内にて)選択された外部スタイルシートの上部へ新たな外部スタイルシートエントリーを挿入し、(グローバルツリー内にて)選択されたグローバルスタイルエントリーの上部へ新たなグローバルスタイルエントリーを挿入することができます。コマンドはコンテキストメニューからも利用することができます。外部スタイルシートやグローバルスタイルを使用する詳細な情報については、次を参照してください: CSS スタイルと作業.

上へ/下へ: **上へ** アイコン ならびに **下へ** アイコン により、選択された外部スタイルシートやグローバルスタイルを、ツリー内部にてそれぞれ上や下へ移動することができます。これらのコマンドを使用することにより、外部スタイルシートやグローバルスタイルの優先順位を操作することができます。**上へ** ならびに **下へ** コマンドはコンテキストメニューからも利用することができます。スタイルの優先順位を変更する詳細方法については、CSS スタイルと作業.

セレクターのスタイルを表示: 用意されている3つのビューの設定を使用することで、外部スタイルシートやグローバルに定義されたスタイルシートの表示方法を指定することができます。**空でないリスト** 、**全て展開** 、**全て縮退** 、コマンドがツールバーとコンテキストメニューコマンドにて利用することができ、空でないリストオプションを有効にすることで、値とともに定義されているスタイルプロパティだけが表示されるようになります。このオプションが選択されていない場合、全てのスタイルプロパティが表示されます。全て展開ならびに全て縮退設定を、空でないリスト設定と組み合わせることで、選択されたセレクターのスタイル定義を展開/縮退することができます。これらのコマンドはコンテキストメニューからも利用することができます。

重要フラグの切り替え: 重要フラグの切り替えアイコン をクリックすることで、選択された CSS ルールに対して !important という CSS 値をセットします。

すべてを再ロード: **すべてを再ロード** アイコン により、全ての外部 CSS スタイルシートを再ロードします。

リセット: **リセット** アイコン により、選択された外部スタイルシートまたはグローバルスタイルシートを削除することができます。

(C) 2015-2021 Altova GmbH

スタイルレポジトリ内にあるCSS スタイルの編集

スタイルリポジトリでは、以下の編集メカニズムを使用することができます:

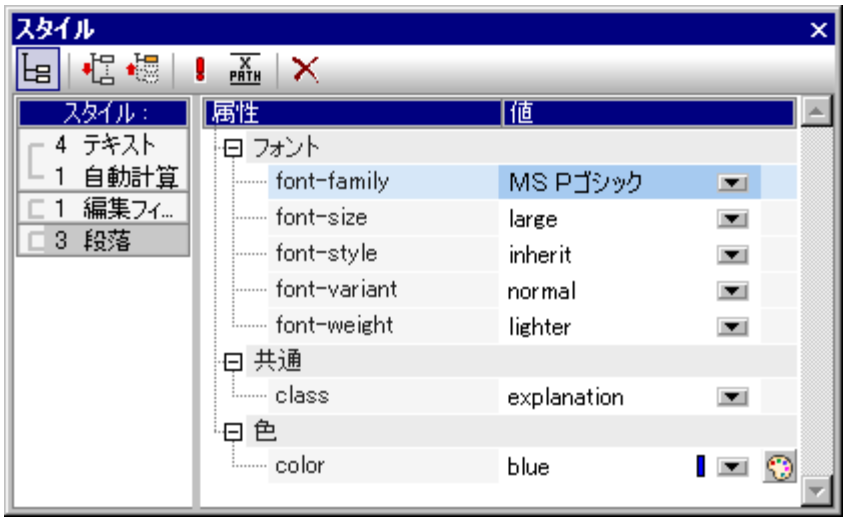
- CSS スタイルシートの追加や削除を行うことができ、各 CSS スタイルシートが適用されるメディアを指定することができます。操作方法については 外部 CSS スタイルシートを参照ください。
- スタイルリポジトリウィンドウでは、グローバルスタイルのセレクターとプロパティを直接編集することができます。操作方法については、CSS スタイルをグローバルに定義するを参照ください。

関連項目

- デザイン概要
- CSS スタイルと作業
- スタイル サイドバー

2.2.5 スタイル

スタイル サイドバーでは、デザインビューにて選択されたSPS コンポーネントに対して、CSS スタイルを**ローカルにて**定義することができます。これはスタイルレポジトリサイドバーにてグローバルに定義されたスタイルとは異なるものです。



スタイルサイドバーは、以下にある2つのパーツから構成されます:

- 左側には、**スタイル**カラムが表示されており、選択されたコンポーネントの型が表示されます。デザインビューでは複数のコンポーネントを同時に選択することができます。選択されたコンポーネントが、コンポーネントの型ごとにスタイルカラムに表示されます。同時に編集することができるのは、表示されているコンポーネントのうち1つだけです。選択されたコンポーネントに属するインスタンスが1つしか無い場合、そのインスタンスがスタイル適用に使用されます。同一コンポーネントの型に属する複数のインスタンスが選択された場合、選択された全てのインスタンスに対してスタイルが適用されます。スタイルは（選択されたコンポーネントの）各インスタンスに対してローカルにて適用されます。特定のインスタンスに対してのみスタイルの適用を行いたい場合、デザインビューにてそのコンポーネントインスンタンスを選択し、スタイルサイドバーにてスタイルの定義を行う必要があります。Shift キーが押下された状態で複数のコンポーネントを

選択することで、連続した複数のコンポーネントを選択することもできます。コンポーネント型の選択に関する詳細については、次を参照してください：CSS スタイルをローカルに定義する。

- 右側には、**スタイル定義ペイン**が表示されており、スタイルカラムにて選択されたコンポーネントの CSS スタイルを定義することができます。スタイル定義ペインは、3つのビューで表示することができます。スタイル定義の詳細については、CSS スタイル値の設定 を参照ください。XPath アイコン により、スタイル値のソースとして XPath 条件式を使用することができます。スタイルプロパティが選択されており、XPath アイコンが有効になっている場合、XPath 条件式をプロパティに対して入力することで、XPath 条件式の値がスタイルプロパティの値として使用されます。これにより、XML ドキュメント内にあるノードの値を、プロパティの値としてランタイムに取得することができます。XPath アイコンが無効になっている場合、静的な値をプロパティの値として入力することができます。

定義表示のオプション

編集のニーズに合わせて、定義の表示を変更することができます。以下に示されるこれらの表示オプションは、ツールバーのボタンとコンテキストメニューから選択することができます。

- **空でないリスト**: このオプションが有効になっている場合、左側のカラムで選択されているコンポーネントにおいて、値が与えられているプロパティだけがアルファベット順で表示されるようになります。オプションが無効になっている場合、すべてのプロパティが表示されます。このオプションにより、特定種類のコンポーネントに対してどのようなプロパティが定義されているか、簡単に確認することができます。選択されたコンポーネント型に対して新たなプロパティを定義するには、このオプションを無効にした状態で、空のプロパティへアクセスする必要があります。

- **すべて展開**: 左側で選択されたコンポーネントに対して、すべてのプロパティを右側のペインに表示します。このオプションは、**空でないリスト**オプションと併用することができます。

- **すべて縮退**: 左側で選択されたコンポーネントに対して、すべてのプロパティを縮退された状態で右側のペインに表示します。このオプションは、**空でないリスト**オプションと併用することができます。

重要フラグの切り替えとツールバーアイコンのリセット

重要フラグの切り替えアイコン をクリックすることで、選択された CSS ルールに対して !important という CSS 値をセットします。

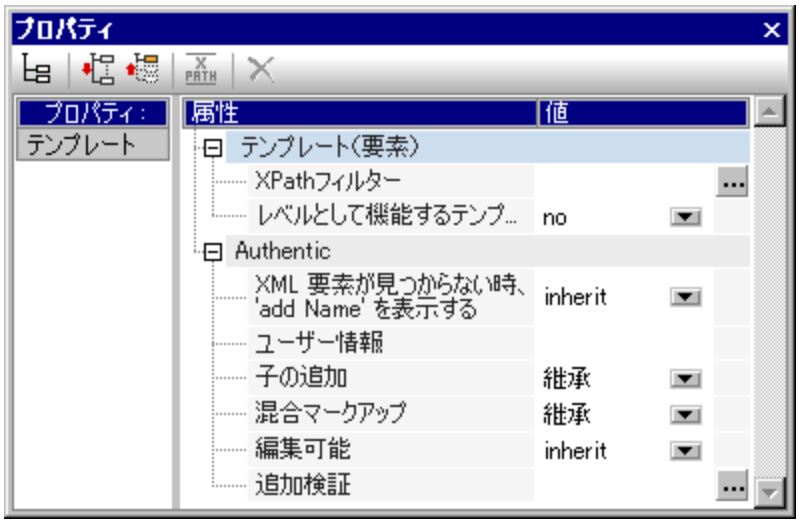
関連項目

- CSS スタイルと作業
- スタイルレポジトリ
- 定義済みのフォーマット

2.2.6 プロパティ

プロパティサイドバーにより、デザインビューにて選択された SPS コンポーネントのプロパティを定義することができます。

(C) 2015-2021 Altova GmbH



プロパティサイドバーは、以下にある2つのパーツから構成されます:

- 左側に表示される**プロパティ**カラムには、選択されているコンポーネントの型が表示されます。同時に編集することができるのは、1つのプロパティだけです。コンポーネントとプロパティをグループ化する方法については、次のセクションを参照してください：コンポーネントとプロパティグループ。
- 右側に表示される**プロパティ定義ペイン**では、プロパティカラムで選択されたコンポーネントに対してコンポーネントプロパティを定義することができます。プロパティ定義ペインは3つのビューにて表示することができます。各プロパティグループに、どのプロパティが収められているかの詳細については、以下の次のセクションを参照してください：プロパティグループ。

定義表示のオプション

編集のニーズに合わせて、定義の表示を変更することができます。以下に示されるこれらの表示オプションは、ツールバーのボタンとコンテキストメニューから選択することができます。

- **空でないリスト**: このオプションが有効になっている場合、左側のカラムで選択されているコンポーネントにおいて、値が与えられているプロパティだけがアルファベット順で表示されるようになります。オプションが無効になっている場合、すべてのプロパティが表示されます。このオプションにより、特定種類のコンポーネントに対してどのようなプロパティが定義されているか、簡単に確認することができます。選択されたコンポーネント型に対して新たなプロパティを定義するには、このオプションを無効にした状態で、空のプロパティへアクセスする必要があります。

- **すべて展開**: 左側で選択されたコンポーネントに対して、すべてのプロパティを右側のペインに表示します。このオプションは、**空でないリスト**オプションと併用することができます。

- **すべて縮退**: 左側で選択されたコンポーネントに対して、すべてのプロパティを縮退された状態で右側のペインに表示します。このオプションは、**空でないリスト**オプションと併用することができます。

ツールバーアイコンのリセット

リセットアイコン をクリックすることで、選択されているプロパティの値をデフォルトに戻します。

コンポーネントとプロパティグループ

選択されているデザインコンポーネントにより、表示されるプロパティグループが変化します。以下のテーブルに、SPS コンポーネントと、それらコンポーネントに含まれるプロパティグループを示します。.

コンポーネント	プロパティグループ
テンプレート	テンプレート; Authentic
コンテンツ	コンテンツ; Authentic; 共通; イベント
テキスト	テキスト; 共通; イベント
自動計算	自動計算; Authentic; 共通; イベント
条件分岐	When
データ入力デバイス	Authentic; 共通; データ入力デバイス; イベント; HTML
イメージ	イメージ; Authentic; 共通; イベント; HTML
リンク	リンク; Authentic; 共通; イベント; HTML
テーブル	テーブル; Authentic; 共通; イベント; HTML; インタラクティブ
パラグラフ	パラグラフ; Authentic; 共通; イベント; HTML

コンポーネントの型に対して以下の点に留意してください:

- テンプレートコンポーネントとは、メインテンプレート、グローバルテンプレート、そしてデザイン内にある全てのスキーマ注釈のことです。
- コンテンツコンポーネントとは content ならびに その他のコンテンツ プレースホルダのことです。ノードのテキストコンテンツや XML ドキュメントから得られたノードが表示されます。
- テキストコンポーネントとは、静的なテキストである文字列のことです。文字列は、テキストコンポーネントを除く全てのコンポーネント間に配置され、空白スペースを含めることもできます。
- データ入力デバイスとは、入力フィールド、複数行の入力フィールド、コンボボックス、チェックボックス、ラジオボタン、そしてボタンことで、プロパティにはデータ入力デバイスと、そのコンテンツを入力することができます。
- テーブルコンポーネントとは、デザイン内にあるテーブル構造のことです。テーブルコンポーネントには、その構成要素がサブコンポーネントとして表示されます。サブコンポーネントには行、カラム、セル、ヘッダー、そしてフッターがあります。
- 段落コンポーネントとは、定義済みのフォーマットのことです。

(C) 2015-2021 Altova GmbH

各プロパティグループの説明を以下のテーブルに示します。

プロパティグループ	説明
自動計算	自動計算が選択されている時に、このオプションが有効になります。値フォーマットプロパティでは、数値や日付データ型といった、自動計算の書式を指定することができます。XPath プロパティでは、自動計算に使用されるXPath 条件式を指定することができます。
Authentic	テンプレート、コンテンツ、自動計算、データ入力デバイス、イメージ、リンク、テーブル、そして段落に対して利用することができるSPS 特有のプロパティです。コンポーネントにより、グループ内で利用可能なプロパティは変化します。詳細についてはAuthentic ノードプロパティ。
共通	共通プロパティグループは、テンプレートと自動計算コンポーネントを除く全てのコンポーネントにて利用可能です。class（クラス名）、dir（書き方の方向）、id（ユニークID）、lang（言語）、title（名前）プロパティをコンポーネントに対して定義することができます。
データ入力デバイス	コンボボックス、チェックボックス、そしてラジオボタンにおける値の範囲を指定します。このプロパティグループは入力フィールドとボタンに対して適用されない点に注意してください。
イベント	以下のクライアントサイドのHTML イベントに対して、JavaScript 関数 を有効にします：： `onclick, ondblclick, onkeydown, onkeypressed, onkeyup, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup.`
HTML	データ入力デバイス、イメージ、リンク、テーブル、パラグラフ のコンポーネント型に対して利用することができます。データ入力デバイスとパラグラフには複数の種類があり、テーブルにはサブコンポーネントが備えられている点に注意してください。これらプロパティは対応するHTML 要素（img、table、p、div など）に対してセットすることができます。そのため、利用可能となるプロパティは、選択されたコンポーネントにより変化します。これらプロパティに対応する値をXPath 条件式により選択することもできます。

更に、イメージ、リンク、パラグラフと他の定義済みのフォーマット と条件分岐に固有のプロパティが用意されています。これらプロパティについては、それぞれのセクションで記述されます。

プロパティの値の設定

プロパティにより、複数の方法でプロパティを入力することができます。

- カラムに値を直接入力する。プロパティを選択し、値カラムをダブルクリックした後に、キーボードから値を入力し、Enter キーを押下するか、GUI をクリックすることで値を確定します。
- プロパティに対応したコンボボックスのドロップダウンリストから、値を選択する。コンボボックスの矢印をクリックして、ドロップダウンのプロパティ値オプションから値を選択します。
- プロパティの値カラムの右側にある編集ボタン ... を使用する。編集ボタンをクリックすることで、そのプロパティのダイアログが表示されます。

共通ならびにHTML グループ内のプロパティによっては、XPath 条件式を使用することでプロパティへ値を与えることができます。XPath アイコン XPATH の有効/無効を切り替えることで、XPath 条件式をプロパティ値のソースとして使用するか指定することができます。プロパティが選択されている状態でXPath アイコンを有効にすると、このプロパティに対してXPath 条件式を入力し、そのXPath 条件式の返り値を、プロパティの値として使用することができます。これにより、プロパティの値としてXML ドキュメント内にある値を、ランタイムにて取得することができるようになります。XPath アイコンが無効になっている場合、プロパティの値として静的な値を入力することができます。詳細については*XPath を使用したスタイルプロパティ*を参照ください。

プロパティ値の修正と削除

プロパティ値を修正し、これまで記述された方法を使用するには、プロパティの値の設定してください。プロパティ値を削除するには、プロパティを選択し、プロパティサイドバーのツールバーにあるリセットアイコン をクリックしてください。

関連項目

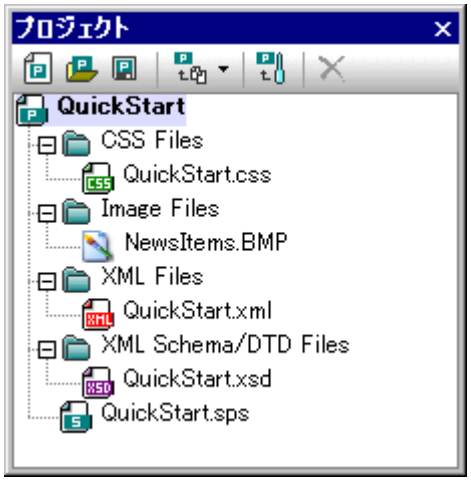
- スタイル

2.2.7 プロジェクト

StyleVision プロジェクトにより、関連するファイルの効率的なグループ化、管理、作業を行うことができます。プロジェクトにファイルが加えられると、SPS をデザインする際にプロジェクトサイドバーから直接ファイルへアクセスすることができるようになります。例えばプロジェクトサイドバーからデザインツリーへSPS ファイルをドラッグすることで、モジュールを作成し、イメージファイルを静的なイメージとしてデザイン内へドロップし、外部スタイルシートとして CSS スタイルシートをスタイルリポジトリへドラッグすることができます。

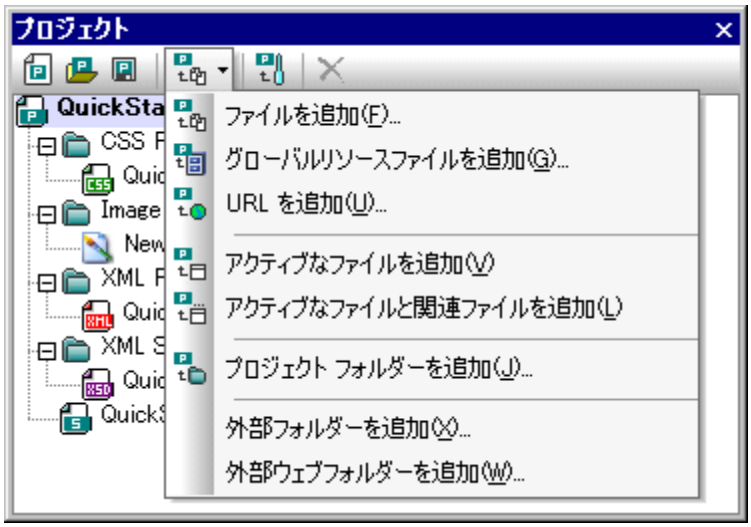
プロジェクトの使用方法に関する詳細については、StyleVision におけるプロジェクトを参照してください。

プロジェクトメニュー内にある全てのコマンドが、現在アクティブなプロジェクトに対して適用されます。新規プロジェクトを作成(「(プロジェクト| 新規プロジェクト)するか、既存のプロジェクトを開く((プロジェクト| プロジェクトを開く)ことで、現在アクティブなプロジェクトを変更することができます。これら2つのコマンドは、プロジェクトサイドバーのツールバーからも選択することができます。新たに作成されたプロジェクトの名前は、保存時(「 (プロジェクト| プロジェクトの保存を選択、または左から3番目に配置されたボタンをクリック)に入力することができます。



ツールバーにある(以下のスクリーンショットでは左から4番目の)**ファイルを追加**ボタンには、以下のスクリーンショットにあるようなコマンドが含まれています。これらコマンドにより、ファイルやフォルダーをアクティブなプロジェクトに追加することができます。詳細については ユーザーリファレンスを参照ください。**アクティブなファイルと関連ファイルを追加**コマンドを選択することで、現在アクティブな SPS に関連した全てのファイルをプロジェクトへ追加することができます。例えば、QuickStart.sps というチュートリアルファイルがアクティブになっている状態でこのコマンドをクリックすると、上のスクリーンショットに表示されている、関連する全てのファイルが追加されます。

(C) 2015-2021 Altova GmbH



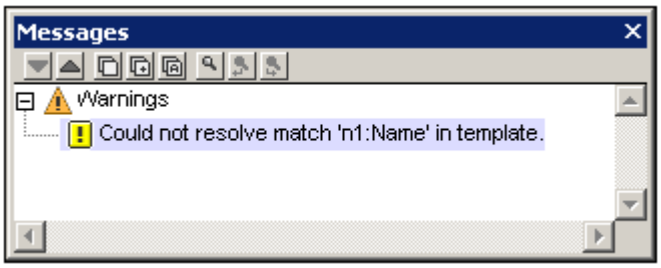
(ツールバーの左から5番目にある) プロジェクトプロパティボタンをクリックすると、プロジェクトサイドバーにて選択されているアイテムにより、(i) プロジェクト、(ii) フォルダー、(iii) ファイルのプロパティダイアログが表示されます。プロジェクトやファイルが選択されている場合、そのファイルの位置がプロパティダイアログに表示されます。フォルダーのプロパティファイルダイアログでは、フォルダーの名前を編集して、フォルダー名とフォルダーに対応したファイル拡張子を指定することができます。指定されたファイル拡張子を持つファイルだけが、そのプロジェクトフォルダーへ表示されます。詳細については[StyleVision におけるプロジェクト](#) を参照ください。

関連項目

- StyleVision におけるプロジェクト
- プロジェクトメニュー

2.2.8 メッセージ

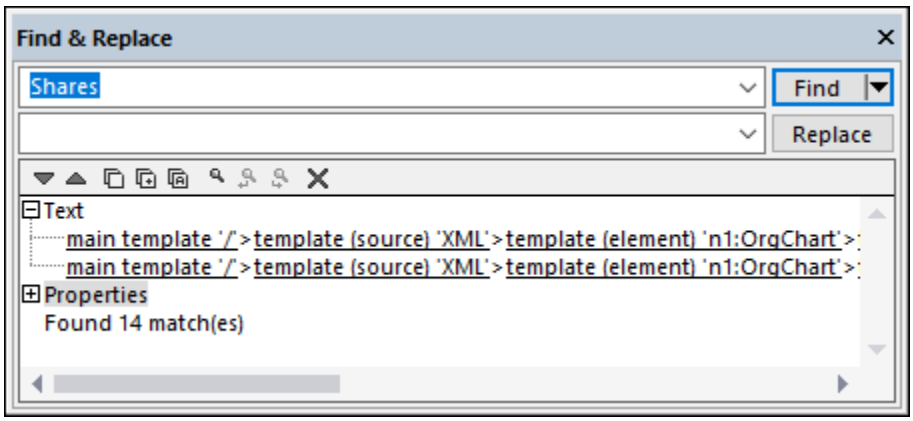
StyleVision が最初に開かれる場合、メッセージサイドバー (*下のスクリーンショット*) がメッセージウィンドウの GUI に表示されます。メッセージサイドバーをオンとオフするには、**表示 | メッセージ**をクリックします。



メッセージサイドバーは、デザインビューとAuthentic View 内の警告を表示します。警告は、不足している作業用 XML ファイル、デザイン構造内のエラーなど、SPS ドキュメントに関連した多くの点に関連してます。Authentic View 内では、警告は、基になるスキーマ、または、追加検証条件に従い、入力された XML データの有効性について説明しています。

2.2.9 検索と置換

検索と置き換えサイドバーによりデザインテキスト、スタイル、プロパティ、XPath 式内で直接入力するまたは正規表現を構成する文字列を検索することができます。デザインビューに検索は適用されます。検索メカニズムはメニューコマンド[編集 | 検索]によりアクセスされる検索ダイアログで使用されるものと同じです。.検索の結果はサイドバー内に表示されます。結果をクリックしてデザイン内の対応するロケーションに移動することができます。検索と置換サイドバーをオンとオフに切り替えるには[表示 | 検索と置換]をクリックします。



検索と置換メカニズムに関する詳細はトピック検索、次を検索、置換の説明を参照してください。

検索

*「検索対象」*フィールドに検索する用語を挿入します。**検索** ボタンのドロップダウン矢印をクリックして 検索オプションを選択します。以下のオプションから選択することができます:

- *選択の箇所:* (i) デザイン内のテキスト (ii) (スタイル サイドバーにて入力される) スタイル内 (iii) (プロパティ サイドバーにて入力される)プロパティ内。Xpath 式内。
- *大文字と小文字/または単語の完全マッチ:* これらはオンとオフに切り替えるオプションです。
- *正規表現:* エントリは正規表現として扱われます。正規表現の使用方法に関しての詳細は検索、次を検索、置換を参照してください。

結果

結果は(i) デザインテキスト (ii) スタイル(iii) プロパティと(iv) Xpath 式のためのグループに整理されます。(*上のスクリーンショットを参照してください*)。各結果アイテムは検索されたアイテムの階層を表示するために構造されています。検索されたアイテムまたは階層内のリンクをクリックしてデザインビュー内のアイテムに移動することができます。

結果ペインは以下の左側からのコマンドのためのアイコンを持っています: 次のおよび前のアイテムに移動する。クリップボードにアイテムをコピーする。結果ペイン内で検索する。結果ペインをクリアする。

置換

置換 ボタンは検索されたすべてのアイテムを *置換* フィールド内の文字列と置き換えます。

3 クイックスタートチュートリアル

このチュートリアルでは、効果的な SPS を作成するにあたり注意すべき点を簡単に解説していきます。最初に SPS の作成とセットアップについて説明し、その後 SPS 内にコンテンツを挿入する方法について、SPS コンポーネントのフォーマット方法、そして、自動計算や条件という、SPS に搭載されている2つのパワフルな機能について説明します。内容を読み進めることで、出力の構造について、また利用可能な機能について効率的な学習を行うことができます。

必要とされるファイル

このチュートリアルで使用されるファイルは、(マイ) ドキュメントフォルダー 以下の `C:\Documents and Settings¥<username>¥My Documents¥Altova¥StyleVision2021¥StyleVisionExamples¥チュートリアル\QuickStart` に収められています:

- `QuickStart.xsd`: SPS のベースとして使用される XML スキーマファイルです。
- `QuickStart.xml`: 作業用 XML ファイルで、内部に含まれているデータが出力プレビューに表示されます。
- `QuickStart.sps`: 既に完成した SPS ファイルで、作成した SPS ファイルと比較することができます。
- `QuickStart.css`: チュートリアルで使用される外部 CSS スタイルシートです。
- `NewsItems.BMP`: SPS にて使用されるイメージファイルです。

チュートリアルの進め方

チュートリアルを最初から読み始め、そのままサブセクションを読み進めてください。チュートリアルを始める前に、XSD ならびに XML ファイルの内容とその構造を確認すると良いでしょう。XSD と XML ファイルを開いたままにしておくことで、チュートリアルを進めながら、それらファイルの内容を確認することができます。`QuickStart.sps` 以外の名前で SPS ドキュメントを保存することで、最初から与えられている SPS ファイルを上書きすること無く、チュートリアルを進めることができます。各ステップを完了した後には、ファイルの保存を忘れないようにしてください。

リンク

- 次のステップ: 新たな SPS の作成とセットアップ

3.1 新規 SPS の作成

このセクションでは、以下の項目について学習します

- 新規 SPS ドキュメントの作成方法。
- SPS に対してスキーマソースを追加する。
- SPS のXSLT バージョンを追加する。
- 作業用 XML ファイル割り当て。
- 出力のエンコードの指定。
- SPS ドキュメントの保存。

このセクション内のファイル

このセクション内で参照されているファイルは以下の場所で見つけることができます: マイドキュメントフォルダー `C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\QuickStart`:

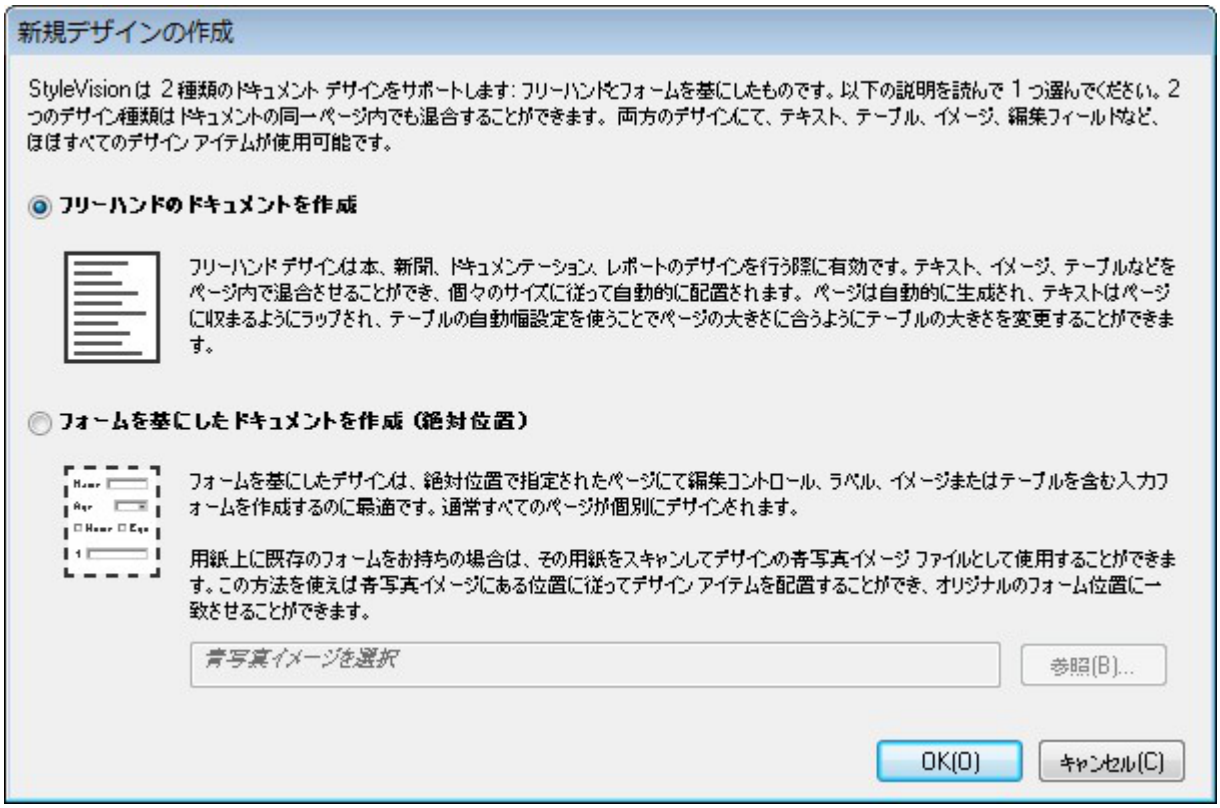
- `QuickStart.xsd` はSPS がベースにされている XML スキーマファイルです。
- `QuickStart.xml` は出力プレビュー内に表示されるデータのソースである作業 XML ファイルです。
- `QuickStart.sps` は完了済みのSPS ファイルです。このファイルを使用して作成したSPS ファイルを比較することができます。

新規 SPS ドキュメントの作成

新規 SPS ドキュメントを作成するには、メニューオプションから**ファイル | 新規作成 | 新規作成(空)** を選択するか、アプリケーションツールバーにある新規アイコン をクリックします。

新規デザインを作成ダイアログボックスが表示され、(i) フリーハンドのドキュメントデザイン、または(ii) フォームをベースにしたドキュメントデザイン(コンポーネントが絶対位置で指定されます)から選択を行うように促されます。

(C) 2015-2021 Altova GmbH

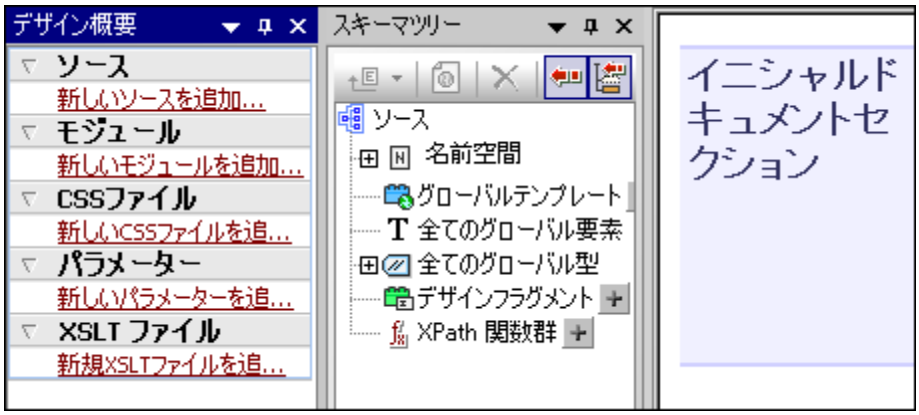


フリーハンドのドキュメントデザインでは、ドキュメントコンテンツが出力メディアオブジェクト（印刷物や画面）にフィットするようにレイアウトされます。ドキュメントコンテンツ内のアイテムは相対的にしか配置することができず、絶対位置を指定することはできません。このオプションは、レポートや記事、書籍などのデザインに適しています。

フォームをベースにしたドキュメントには単一のレイアウトコンテナーが作成され、その中でデザインコンポーネントを絶対位置で指定することができます。レイアウトコンテナーには、絶対位置でレイアウトボックスを配置することができ、各レイアウトボックス内にドキュメントコンテンツを配置することになります。既にあるフォームのデザインを SPS にて再現したい場合、オリジナルのフォームイメージを青写真イメージ として使用することができます。青写真のイメージをレイアウトコンテナーの背景として含めることができ、フォームのデザインに役立てることができます。

ここではフリーハンドのドキュメントを作成します。**フリーハンドのドキュメントを作成**ラジオボタンをクリックして、「OK」をクリックしてください。

SPS1.sps という名前の新規ドキュメントが作成され、デザインビューに表示されます（以下のスクリーンショットを参照）。

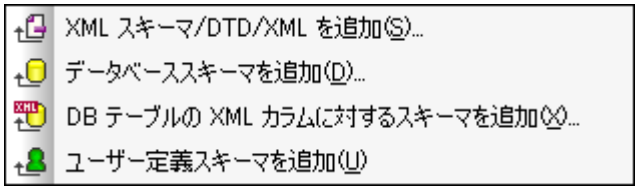


デザインビューにて空のメインテンプレートが表示されます。デザイン概要とスキーマツリーサイドバーにおいてスキーマエントリーは表示されません。

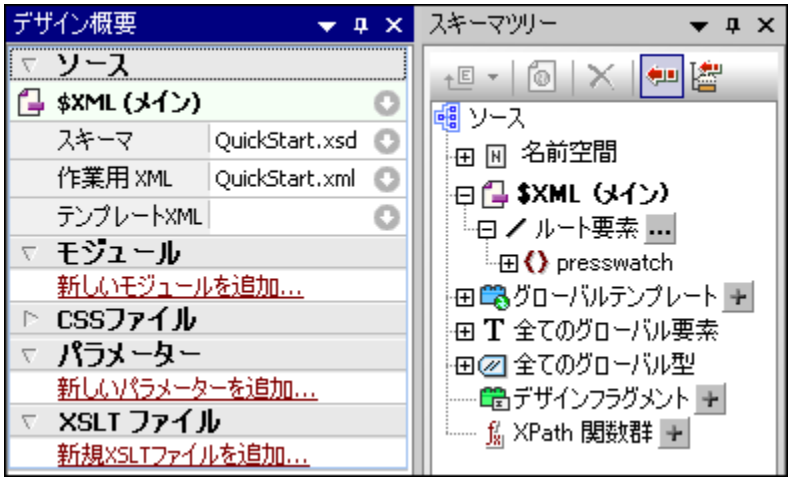
スキーマソースの追加

SPS に対して QuickStart.xsd スキーマを使用します。以下の操作により、スキーマをスキーマソースとして追加することができます:

1. デザイン概要サイドバーから**新しいソースの追加**コマンドをクリックします。メニューが表示されるので、**XML スキーマ/DTD/XML を追加**を選択します。



2. ファイルを開くダイアログボックスが表示されるので、(マイ) ドキュメントフォルダー `C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\QuickStart\QuickStart.xsd` を選択し、**開く**をクリックします。
3. 作業用 XML ファイルを選択するように促されます。ファイルシステムからファイルを選択します。**参照**ボタンをクリックして、(マイ) ドキュメントフォルダー) 内の `C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\QuickStart\QuickStart.xml` を選択し、**開く**をクリックします。スキーマがデザイン概要サイドバー内部のスキーマソースとして追加されます(以下のスクリーンショットを参照)。更に、選択した XML ファイルが、スキーマに対する作業用 XML ファイルとしてデザイン概要に表示されます。



以下の点に注意してください! (i) デザイン概要内にある $XML エントリーに、スキーマや作業用 XML ファイルとテンプレート XML ファイルのリストが表示され、(ii) スキーマツリーサイドバーのルート要素ツリーには、スキーマ内に定義されたグローバル要素 選択した、ルート要素 (ドキュメント要素) が表示されます。このスキーマの場合、グローバル要素 から選択した要素、この場合要素 `presswatch` 、が表示され、(iii) スキーマ内にある グローバル要素 がすべてのグローバル要素ツリーに表示されます。

XSLT バージョンの選択

この例では SPS に対して XSLT 2.0 を使用します。XSLT のバージョンを指定するには、アプリケーションツールバーから XSLT 20 アイコンをクリックします。

作業用 XML ファイルの割り当てまたは変更

これまでの操作によりXML スキーマを SPS へ追加して、作業用 XML ファイルをスキーマへ割り当てました。作業用 XML ファイルにより XML データのソースがSPS へ与えられます。作業用 XML ファイルの割り当て、変更、そして割り当て解除を行うには、デザイン概要サイドバー内にある作業用 XML ファイルの行を右クリック(または右側にあるコンテキストメニューアイコン をクリック)して、コンテキストメニューから目的のコマンドを選択します。 作業用 XML ファイルの割り当てが行われ、デザイン概要にてファイル名を入力することができます。他の操作を行う前に、(マイ) ドキュメントフォルダー内にある `C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\QuickStart\QuickStart.xml` が作業用 XML ファイルとして割り当てられていることを確認してください。

出力のエンコーディングを指定する

オプション ダイアログのデフォルトのエンコードタブ (ツール | オプション) にて HTML エンコード Unicode UTF-8 に設定し、RTF のエンコーディングを UTF-8 にセットすることができます。

SPS ドキュメントの保存

上記のSPS のセットアップを終えた後に、`MyQuickStart.sps` (マイ) ドキュメントフォルダー `C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\QuickStart` フォルダーにて MyQuickStart.sps という名前でドキュメントを保存してください。メニューコマンド ファイル | デザインを保存 」を選択するか、Ctrl+S を押下して、ダイアログにてファイル名を入力します。

リンク

- 次のセクション: 動的なコンテンツの挿入
- チュートリアルスタートページ

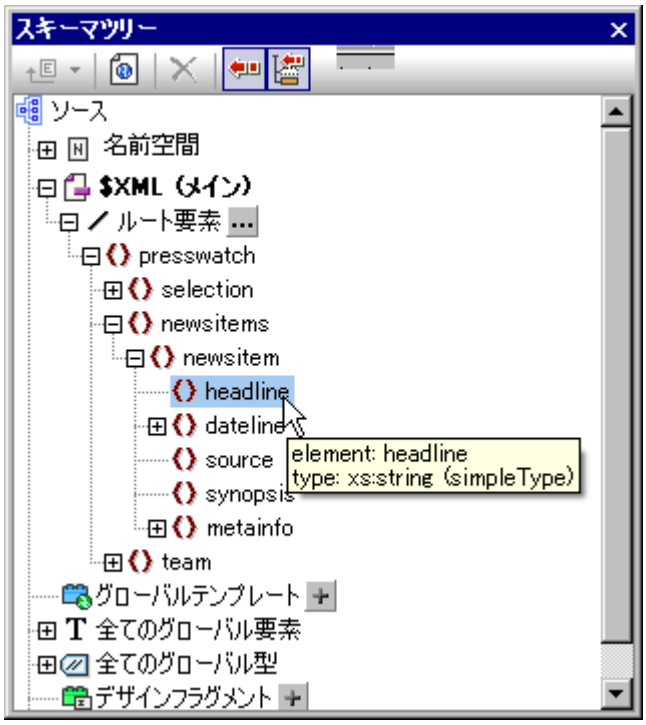
3.2 (XML ソースの)動的なコンテンツの挿入

このセクションでは、XML ドキュメント内にあるノードからデータの挿入を行う方法について説明します。スキーマツリーに表示されている要素や属性ノードをデザインへドラッグすることで、これらノードをコンテンツとして作成することができます。ノードがコンテンツとして作成されると、その中にあるデータが要素の子テキストノードや、その子孫テキストノードのコンテンツが文字列として出力されます。

要素コンテンツの挿入

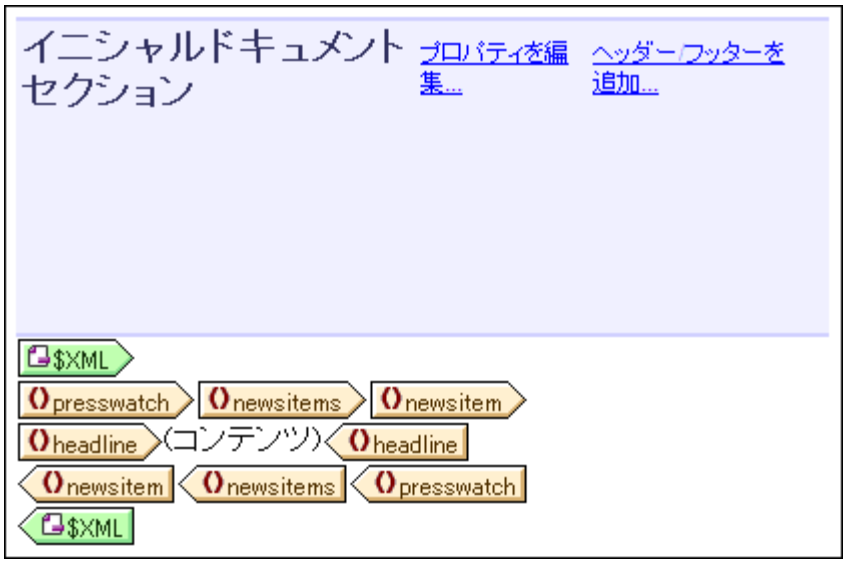
SPS において、以下の操作を行います::

1. スキーマツリーサイドバーにおいて、スキーマツリーを展開して、newsitem 要素の子要素を表示します。(*下のスクリーンショット*).

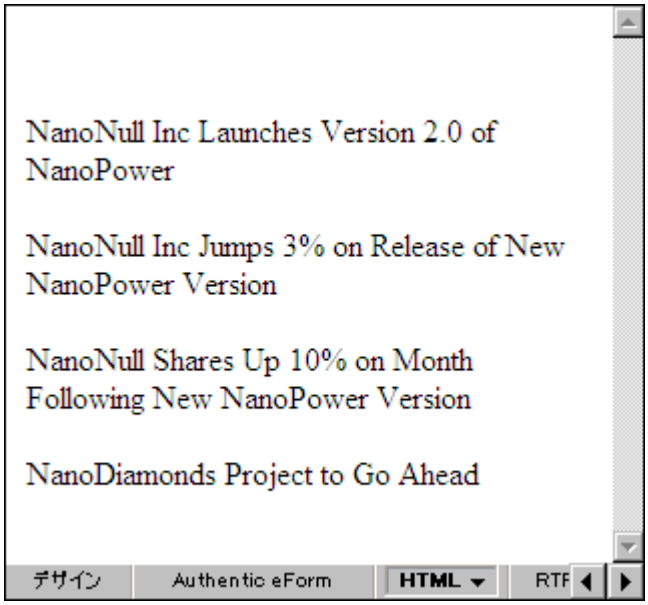


2. headline 要素を選択します(マウスオーバーにより、要素データ型がポップアップで表示されます)。要素をデザインビューへドラッグして、矢印カーソルが挿入状態に変わった状態でメインテンプレートへドロップすることができます。
3. コンテキストメニューから、**コンテンツの作成**を選択します。headline 要素の開始ならびに終了タグが、デザイン内のドロップされた箇所に挿入され、コンテンツプレースホルダが内部に含まれます。headline タグの周りには、その祖先要素のタグが表示されます(スクリーンショットを参照)。

デザインにて、(Enter キーを押下することで)以下のスクリーンショットにあるように改行を挿入します。



HTML タブをクリックして、 HTML 出力のプレビュー(*下のスクリーンショット*) を表示します。HTML プレビューには newsitem の headline 子要素がテキストとして表示されます。



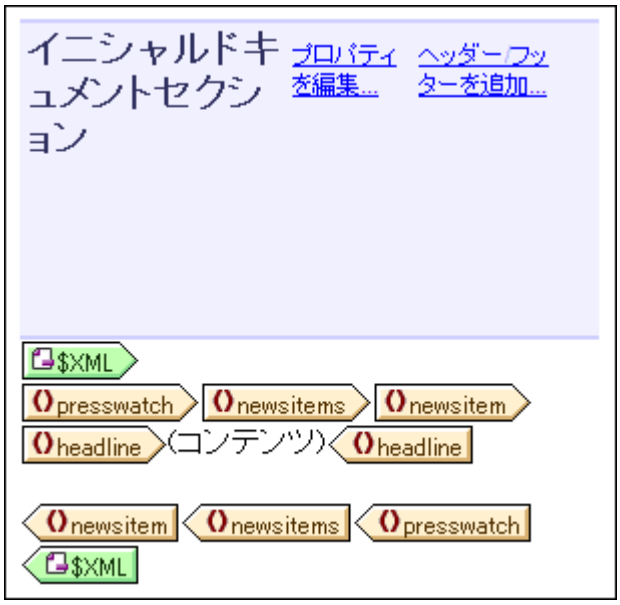
Authentic View や RTF 出力。

メモ デザイン要素の挿入 ツールバー ツールバーにあるコンテンツを挿入をクリックして、デザイン内の位置をクリックし、スキーマセレクターツリーから目的の位置を選択することで、ノードのコンテンツを作成することもできます。

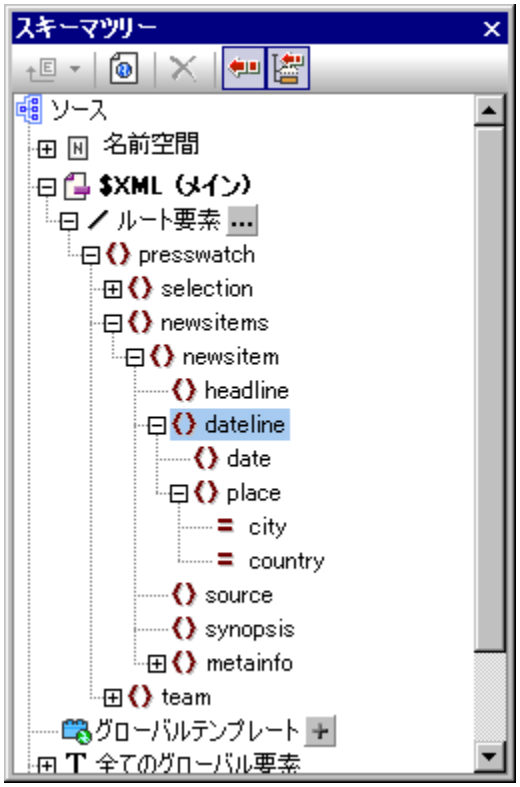
属性コンテンツを挿入する

コンテンツとして要素がデザインへ挿入された場合、属性のコンテンツは自動的に挿入されません。属性ノードをデザインへ挿入することで、属性の値が出力へ表示されるようになります。以下の操作を SPS に対して行なってください:

1. カーソルを headline 要素タグの終端に配置して、Enter キーを押下します。この操作により、空の行が作成されます(スクリーンショットを参照)



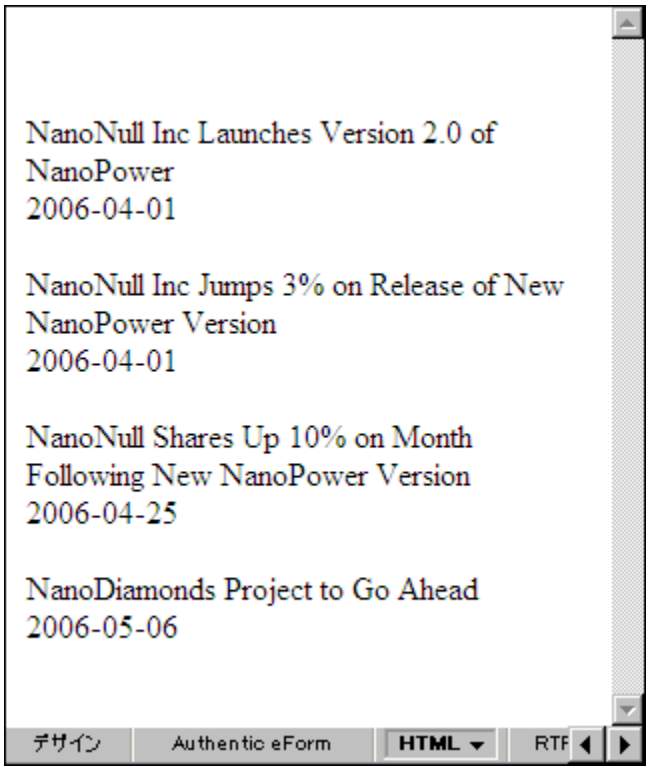
2. スキーマツリーサイドバーにて、dateline 要素を展開します。



dateline 要素には date と place という2つの子要素が含まれており、place 要素には city と country という2つの属性が含まれていることに注意してください。

3. デザイン内の新たに作成した空の行に dateline 要素をドロップします。(*下のスクリーンショット*).

4. HTML プレビュー へ切り替えて、dateline の出力を確認します(*下のスクリーンショット*)。

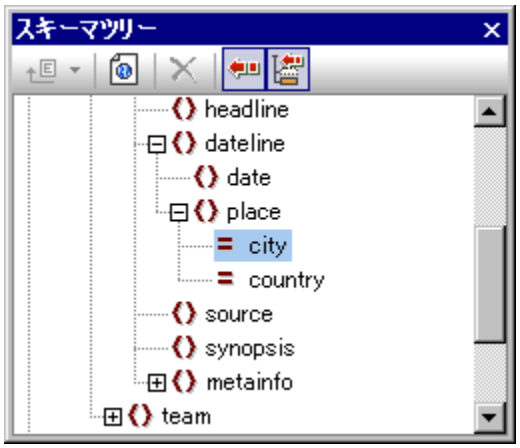


dateline 要素の子である date のコンテンツが出力されていることに注目してください。dateline の子である place に対してコンテンツの出力は行われていません。これは、place のデータが place 要素の属性(city ならびに country 属性)に含まれている一方で、属性の親要素が処理された際に属性のコンテンツは出力されない為です。

5. デザインビューにて、Authentic | 日付選択の自動追加 メニューコマンドのチェックを外すことで、日付選択の自動追加 を無効にします(アイコンの境界線が無くなります)。この操作により、(次の操作で行う)xs:date または

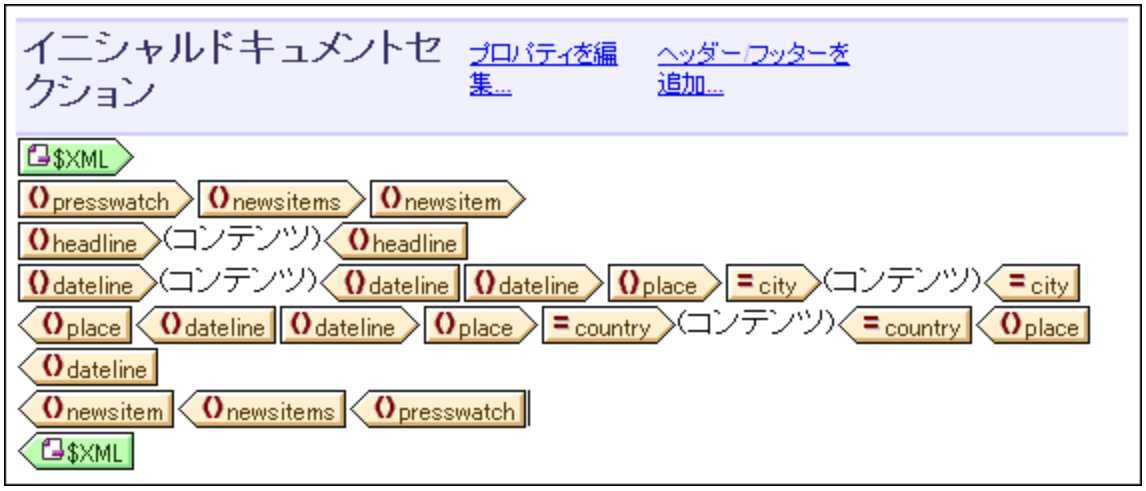
xs:dateTime がデザインへ挿入された際に、日付選択が自動的に挿入されなくなります。スキーマツリーサイドバーから date 要素をドラッグして、dateline 要素の開始ならびに終了タグの間にドロップします。

6. スキーマツリーサイドバー内にある dateline/place 要素の city 属性を選択します(*下のスクリーンショット*)。



7. @city 属性ノードを デザインビュー へドラッグして、date 要素の終了タグ直後にドロップし、コンテンツを作成します。
8. @country ノードを デザインビュー へドロップして、@city 属性の終了タグ直後にドロップし、コンテンツを作成します。.

上の操作を行った後に、SPS デザインは以下のようになっているはずです:

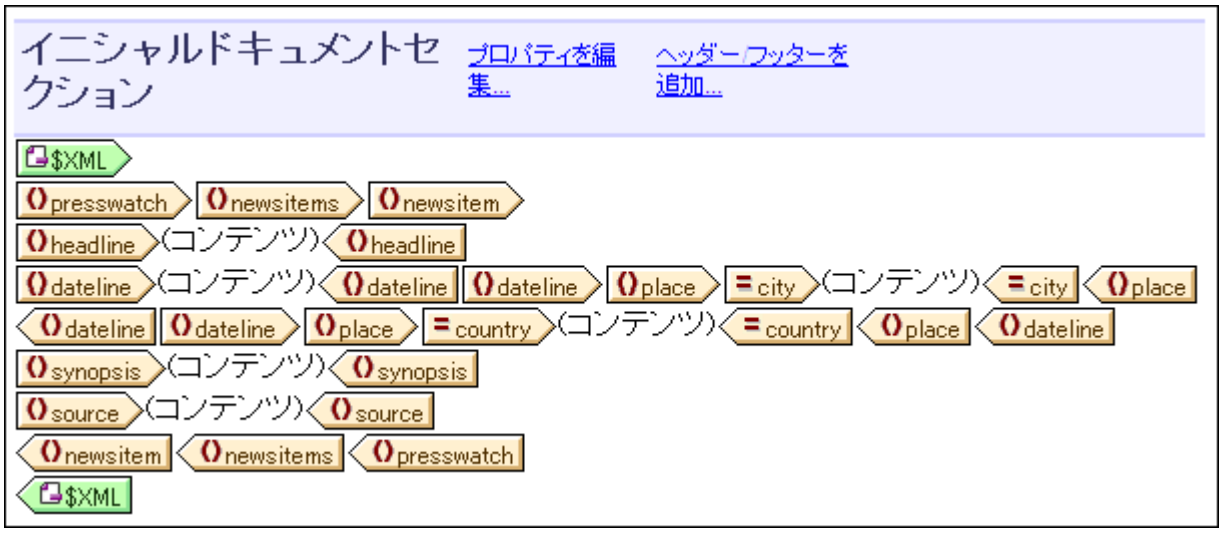


HTML プレビュー は以下のように表示されます::

@city ならびに@country 属性の値が出力に含まれていることに注目してください。

更に動的なコンテンツを追加する

上記の方法により、XML データソースから得られた要素ならびに属性のコンセプトを、デザイン内の任意の場所へ挿入することができます。このセクションを終了する前に、以下のスクリーンショットにあるように synopsis ならびに source 要素をデザインへ追加してください!



synopsis 要素が source 要素の前に配置されていることに注意してください(スキーマ内の順番と異なっています)。synopsis と source 要素をデザインに追加した後に、HTML プレビュー から出力を確認してください。メインのテンプレート 内に配置されたノードの順番により、出力に表示される順番となります(ドキュメントの構造と出力ドキュメントに関する詳細については、 テンプレートとデザインフラグメント を参照ください)。

その他にも、デザイン内に作成されるノードを含む形式(フォーム)について留意する必要があります。HTML プレビューでは、デザイン内にある全てのノードが、テキスト文字列として出力へ送信されます。テキスト文字列として出力する代わりに、テーブルやコンボボックスといった、その他の形式での出力を行うこともできます。このセクションでは、全てのノードを(コンテンツ)として作成したため、これら全てのノードはテキスト文字列として出力されます。条件を使用するセクションでは、コンボボックスとしてノードを作成する方法について学習し グローバルテンプレートとその他のコンテンツの使用 セクションでは、(動的な)テーブルとしてノードを作成する方法について学習します。

次のセクションを始める前に、ファイルを保存するようにしてください。

リンク

- 次のセクション: 静的コンテンツの挿入
- 前のセクション: 新規 SPS の作成とセットアップ
- チュートリアル スタートページ
- SPS ファイル構造

(C) 2015-2021 Altova GmbH

3.3 静的コンテンツの挿入

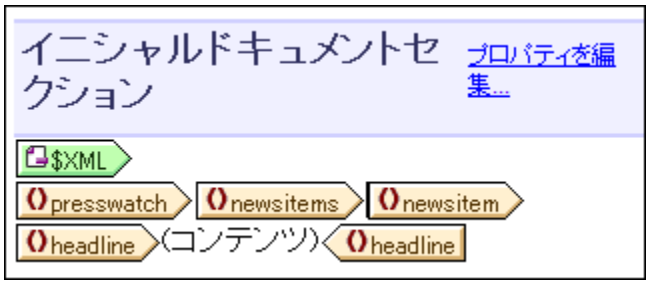
静的コンテンツとは、XML ソースから得られるコンテンツとは違い、デザイン内に直接入力または挿入されるコンテンツのことです。SPS デザインには様々な静的コンポーネントを配置することができます。このチュートリアルセクションでは、以下の静的コンポーネントを挿入します:

- イメージ
- 水平線
- テキスト

静的イメージを挿入する

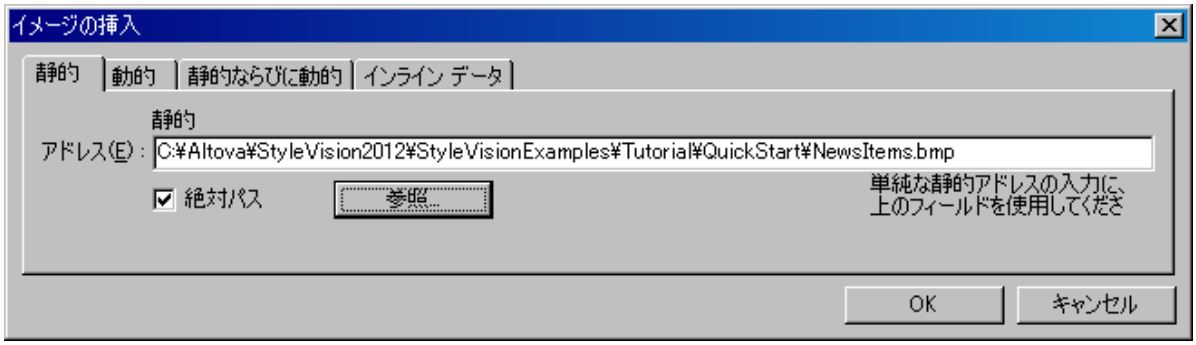
挿入する静的イメージは、(マイ) ドキュメントフォルダー以下の: `C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\QuickStart\NewsItems.BMP` で、ドキュメントのヘッダーとして使用されます。以下の操作により、このイメージをドキュメントの先頭へ挿入することができます:

1. newsitems と newsitem の開始タグの間にカーソルを配置します(以下のスクリーンショットを参照)。



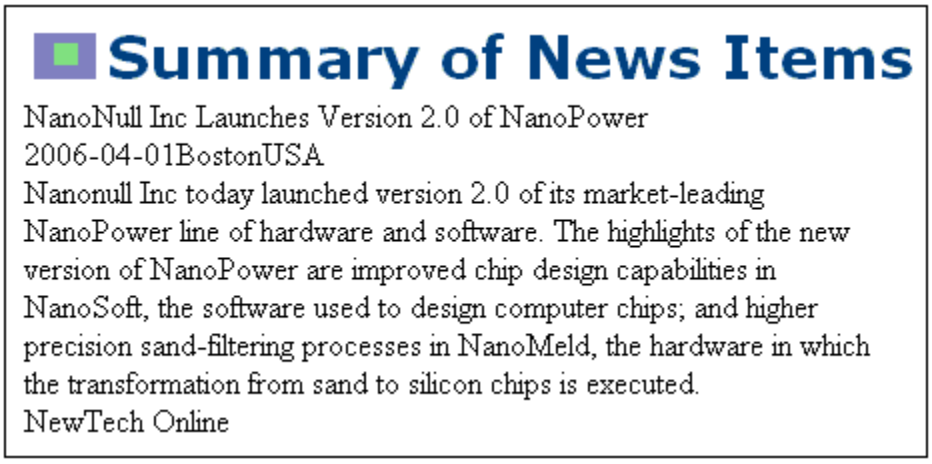
カーソルが newsitems 要素の内側、かつ newsitem 要素の外側に配置されます。そのため、newsitems 要素が処理される際に、1度だけ挿入されます(スキーマ内では newsitems 要素が1度だけ定義されているため)。

2. カーソル位置で右クリックして、挿入 | イメージ を選択します。イメージの挿入ダイアログが表示されます(以下のスクリーンショットを参照)。



3. 静的タブにて絶対パスのチェックボックスをクリックして、NewsItems.BMP を参照ボタンから選択します。
4. 「OK」 をクリックして完了します。

HTML プレビューが以下のように表示されます:



水平線を挿入する

以下の操作を行うことで、ドキュメントのヘッダーとドキュメントのボディーの間に水平線を挿入します:

1. 先ほど挿入した静的イメージの直後にカーソルを配置します。
2. その位置で右クリックして、挿入 | 水平線を選択します。水平線が挿入されます。

水平線のプロパティを以下の方法でセットします:

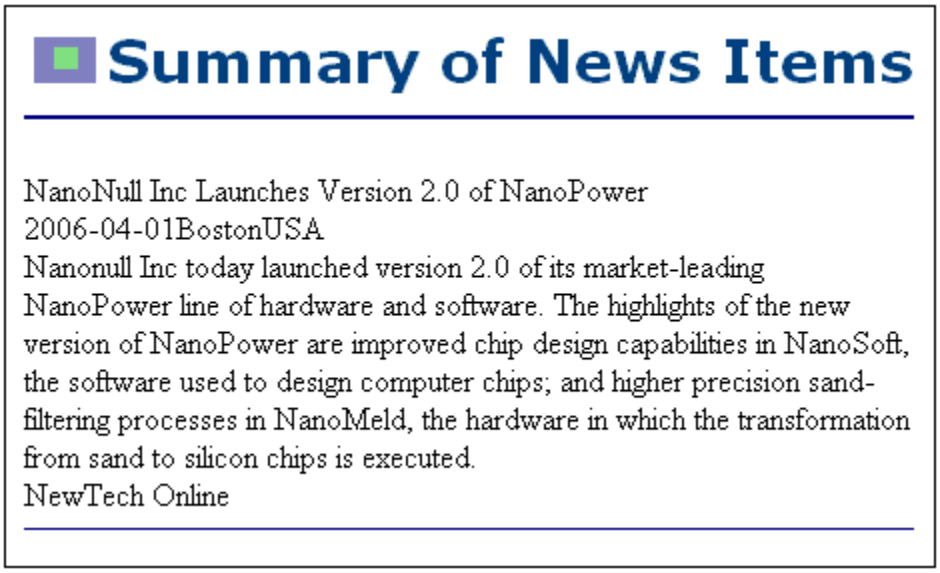
1. デザインビューにて水平線が選択されている状態で、プロパティサイドバーにあるラインコンポーネントを選択し、プロパティの HTML グループを開きます。
2. 水平線に対して color と size プロパティをセットします。
3. デザインビューにてラインが選択されている状態でスタイルサイドバーにあるラインコンポーネントを選択し、プロパティのボックスグループを開きます。margin-botton プロパティの値に12pt をセットします。
4. HTML プレビューにて出力を確認します。

各ニュースアイテムの最後に水平線を挿入することとします。newsitem 要素にある終了タグ直前にカーソルを配置することで、各 newsitem 要素の終端にラインが表示されるようになります。ラインの size プロパティに単位無しの数値を入力することで、水平線の太さを変更することができます(プロパティサイドバーにてラインを選択し、size の値に例えば3 を入力してみてください)。

静的なテキストを挿入する

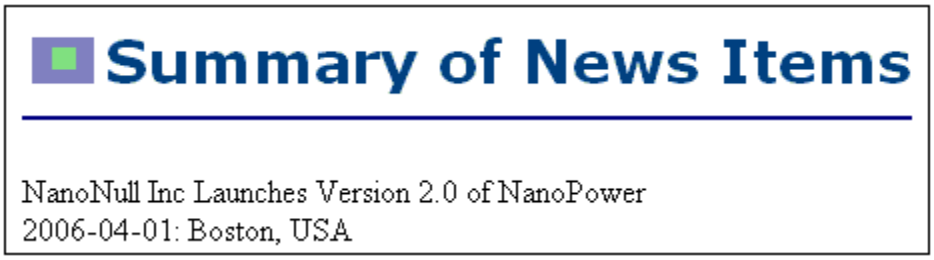
実は(XML ソースから)動的なコンテンツを挿入するセクションで Enter キーを押下することで、改行という静的なテキストが既に追加されています。このセクションでは、静的テキストとしてデザイン内に文字を追加します。

ここまでデザインした SPS の出力結果は、以下のようになります:



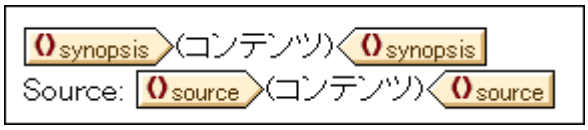
dateline 要素内にdate 要素とplace/@city ならびにplace/@country 属性のコンテンツが(空白スペース無しに)出力されている点に注目してください。空白スペースは静的テキストとしてデザインに追加することができます。デザインにて、date 要素の直後にカーソルを配置して、コロンとスペースを入力します。その後、@city 属性の直後にコンマを入力します(以下のスクリーンショットを参照)。

出力は以下のようになります:



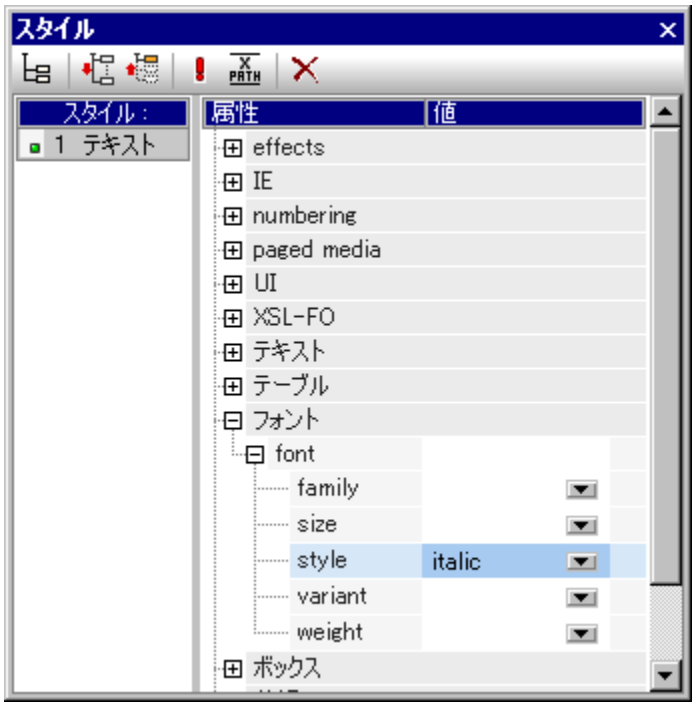
dateline 出力にてコロンとコンマ、そしてスペースが表示されます。これらテキストアイテムは、全てデザインへ直接挿入されたテキストアイテムです。

静的テキストを更に追加します。デザインにて、"Source: " というテキストをsource 要素の開始タグ直前に入力します。

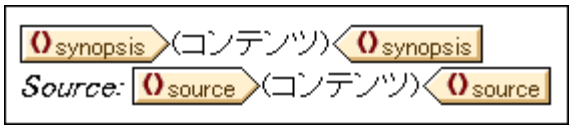


静的なテキストをフォーマットする

静的なテキストをフォーマットするには、目的のテキストをハイライトして、ローカルのスタイルプロパティを指定します。デザインにて入力した "Source: " というテキストをハイライトします。 スタイルサイドバーにて、1 テキストコンポーネントを選択します。以下のスクリーンショットにあるように、プロパティのfont グループを展開して、font-style プロパティにてitalic オプションを選択します。(*下のスクリーンショット*) 。



静的テキスト（ここでは"Source: "）がデザインにてイタリック状態になります:



HTML プレビューの出力は以下のようになります::

ソースアイテムと水平線アイテムの間にある垂直のスペースが狭すぎると思える場合、デザインにてソースと水平線の間に(Enter キーを押下することで)空の行を挿入してください。

操作が終わった後に、ファイルの保存を行います。

このセクションでは、静的コンテンツの挿入とフォーマット方法について学習しました。次のセクションではCSS を使用することにより、デザインコンポーネントをフォーマットする方法について学習します。

リンク

- 次のセクション: コンテンツのフォーマット
- 前のセクション: 動的なコンテンツの挿入
- チュートリアルスタートページ

3.4 コンテンツのフォーマット

StyleVision には、CSS をベースにしたパワフルで柔軟性に富んだスタイルメカニズムが用意されており、デザイン内にあるコンポーネントのフォーマットに使用することができます。StyleVision に搭載されたスタイルメカニズムの重要な点を以下に示します:

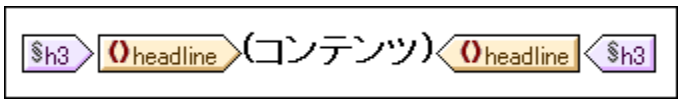
- CSS スタイルルールをブロックコンポーネントならびにインラインコンポーネントに対して定義することができます。
- 定義済みのフォーマット とは承継されるスタイルを含むブロックコンポーネントのことで、コンポーネントのグループに対するラッパーとして使用されます。これら定義済みフォーマットの継承されたスタイルは、各コンポーネントに対してローカルでスタイルを指定することにより、オーバーライドすることができます。これはCSS のカスケード処理に従ったものです。
- デザインではコンポーネントに対するクラス属性を宣言することができ、外部 またはグローバルスタイルルールのセレクターとしてクラスを使用することができます。
- スタイルの指定は3つのレベルで行うことができます。優先順位の低いものから、(i) 外部スタイルシート、(ii) グローバルスタイルルール、と(iii) ローカルスタイルルールとなります。名前をベースとしたセレクター(例えばh1、a、img)などのセレクター型が外部またはグローバルスタイルルールで定義されている場合、それらセレクターは Authentic View とHTML 出力に対してのみ適用され、RTF 出力へは適用されない点に注意してください。クラスセレクターを含むルールは、HTML、RTF、PDF、そして Word 2007+ フォーマットに対して適用されます。.

このセクションでは、以下のことがらを学習します:

- 定義済みのフォーマットを割り当てる
- コンポーネントにクラス属性を割り当てる
- 外部 CSS スタイルシートにてスタイルを定義して そのスタイルシートを SPS のスタイルリポジトリへ追加する
- グローバルスタイルルールを定義する
- 選択された複数デザインコンポーネントに対してローカルスタイルを定義する
- 単一のコンポーネントに対してローカルスタイルを定義する。

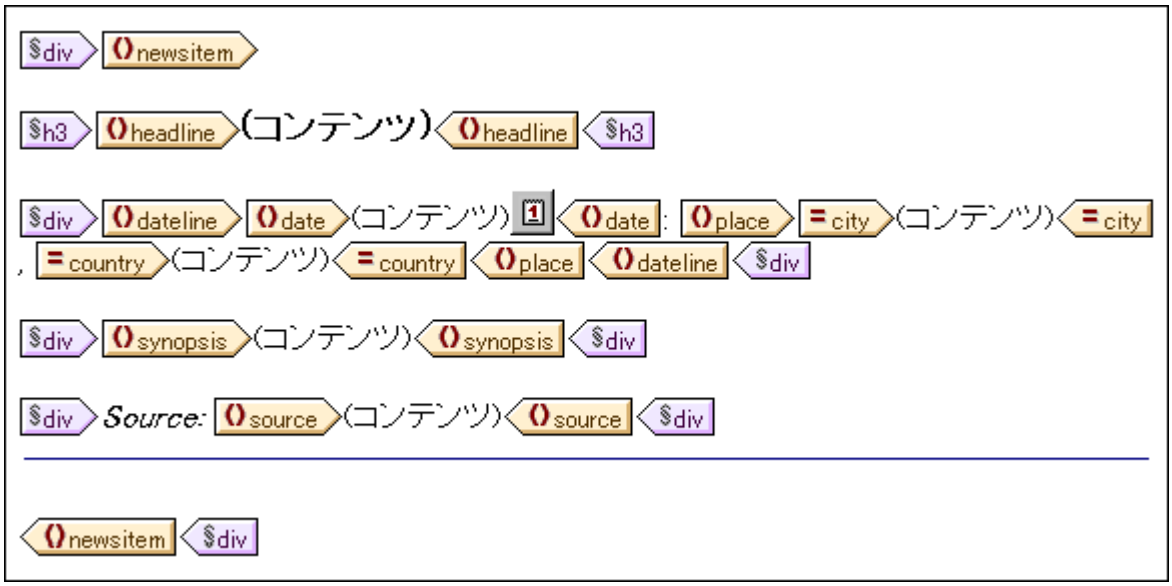
定義済みのフォーマットを割り当てる

定義済みのフォーマット を割り当てることで 定義済みのフォーマットのスタイルを、継承というかたちでコンポーネントへ与えることができます。デザインにおいて headline 要素を選択し、メニューオプションから「**タグで囲む | 特殊な段落 | 見出し3 (h3)**」を選択します。定義済みのフォーマットタグがheadline 要素の周りに作成されます (*下のスクリーンショット*)。



コンテンツのフォントプロパティが変化して、定義済みフォーマットの上と下に空白が追加されます。プロパティ値はh3 定義済みフォーマットから継承されます。

定義済みのフォーマットを使用することで、ブロック内にあるコンポーネントをグループとしてデザインし、それらコンポーネントをブロックやグループとしてフォーマットすることができます。この中でも特にdiv フォーマットを使用することで、空白なしのブロックを作成することができます。デザインにて newsitem、dateline、synopsis、そして source ノードにdiv コンポーネントを割り当てます。デザインは以下にあるスクリーンショットのようになります。静的テキストの"Source: " がsource 要素を含むdiv コンポーネント内に含まれるようになり、newsitem 要素全体がdiv コンポーネント内に含まれます。

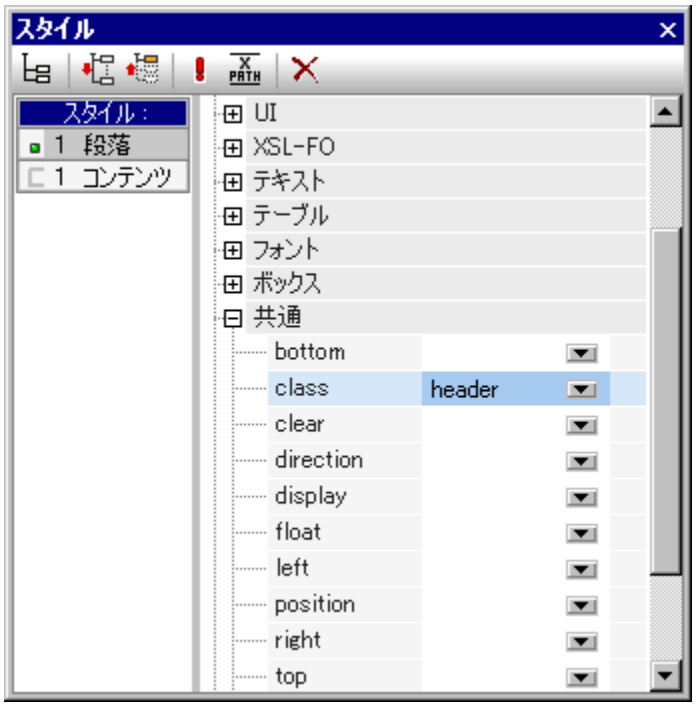


div ブロック内とともに、コンポーネントがグループ化されました。このセクションの終盤 では、グループ化されたコンポーネントのブロックに対してスタイルの割り当てを行います。

コンポーネントをクラス属性に割り当てる

スタイルはコンポーネントのクラスに対して定義することができます。例えば、全てのヘッダーに対して(フォントファミリーや色といった)共通したプロパティを定義することができます。定義を行うには、(i) 共通したプロパティを含んだコンポーネントを割り当て、(ii) そのクラスに対してスタイルプロパティを割り当てる、という操作を行う必要があります。

デザインにて h3 タグを選択し、スタイルサイドバーにて 1 段落 を選択することで、共通プロパティグループを選択します。共通プロパティグループを展開している状態で、class プロパティの値フィールドをダブルクリックして、header という値を入力します。



h3 フォーマットのインスタンスがheader という名前のクラスへ割り当てられます。header クラスに対してスタイルプロパティを定義した場合、これらのプロパティがheader クラスを持っているSPS 内の全てのコンポーネントに対して適用されます。

外部 CSS スタイルシートをスタイルレポジトリへ追加する

外部 CSS スタイルシート内にあるスタイルルールは SPS デザイン内にあるコンポーネントへ適用することができます。外部スタイルシートのルールをコンポーネントに対して適用するためには、スタイルシートをスタイルレポジトリへ追加することができます。スタイルレポジトリサイドバー（デザインビュー内の）にて、以下のような操作を行います:

1. 外部アイテムを選択します。
2. スタイルレポジトリサイドバーのツールバーにある**追加**ボタンをクリックします。ファイルを開くダイアログが表示されます。
3. （マイ）ドキュメントフォルダー以下にある `C:\Documents and Settings\<username>\My Documents¥Altova¥StyleVision2021¥StyleVisionExamples¥チュートリアル\QuickStart\QuickStart.css` を開きます。

スタイルシートがスタイルレポジトリへ追加されます。この段階で関連する記述を以下に示します:

```
.header       {
                 font-family: "Arial", sans-serif;
                 font-weight: bold;
                 color: red;
                }

h3              {
                 font-size: 12pt;
                }
```

header クラスとh3 要素のスタイルルールを組み合わせることで、以下のHTML 出力がheadline 要素に対して作成されます。

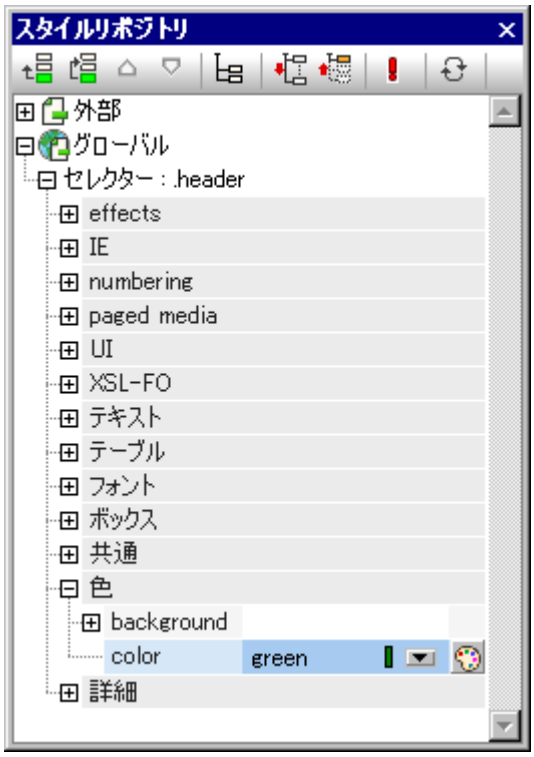
NanoNull Inc Launches Version 2.0 of NanoPower

2006-04-01: Boston, USA
Nanonull Inc today launched version 2.0 of its market-leading NanoPower line of hardware and software. The highlights of the new version of NanoPower are improved chip design capabilities in NanoSoft, the software used to design computer chips; and higher precision sand-filtering processes in NanoMeld, the hardware in which the transformation from sand to silicon chips is executed.
Source: NewTech Online

グローバルスタイル ルールを定義する

CSS セレクターを使用することで、グローバルスタイルルールをSPS 全体に対して定義することができます。ルールはスタイルレポジトリサイドバー内で直接定義します。header クラスに対して、以下のようにグローバルスタイルルールを作成します:

1. デザインビューがアクティブになっている状態で、スタイルレポジトリサイドバー内のグローバルアイテムを選択します。
2. ツールバーの**追加**ボタンをクリックします。この操作により、ワイルドカードセレクター (*) に対して空のルールが作成されます。
3. ワイルドカードの代わりに .header と入力して、セレクターを置き換えます。
4. プロパティの color を展開して、color プロパティ値のドロップダウンリストから green を選択します（以下のスクリーンショットを参照）。



外部スタイルシートでも定義されているプロパティに対してグローバルスタイルルールの定義を行うと、グローバルルールのプロパティ値が優先的に使用されます。そのため、以下のHTML プレビューではヘッドラインのコンテンツが緑色で表示されます。外部スタイルシートにて定義されたその他のプロパティは、(グローバルルールにより上書きされることなく)そのまま使用されます(この例では、font-family とfont-weight)。

NanoNull Inc Launches Version 2.0 of NanoPower

2006-04-01: Boston, USA
Nanonull Inc today launched version 2.0 of its market-leading NanoPower line of hardware and software. The highlights of the new version of NanoPower are improved chip design capabilities in NanoSoft, the software used to design computer chips; and higher precision sand-filtering processes in NanoMeld, the hardware in which the transformation from sand to silicon chips is executed.
Source: NewTech Online

メモ　グローバルスタイルルールはクラスセレクターを使用するため、このルールは、Authentic View とHTML 出力だけではなく、RTF 出力にも適用されます。

複数のコンポーネントに対してローカルスタイルを定義する

複数のコンポーネントに対してローカルスタイルを同時に定義することができます。作成中のデザインにて、ニュースアイテムのテキスト全体にArial フォントを使用したい場合、newsitem 要素を囲んでいるdiv コンポーネントをクリックして、スタイルサイドバーのスタイルカラムにて、「1 段落」を選択し、プロパティのフォントグループ内にあるfont-family にArial を選択します。この操作により、それから派生した形式全てが影響をうけることになります。

デザインにて、dateline、synopsis、source ノードを囲んでいる3つのdiv コンポーネントを(Shift キーを押下した状態で各 div コンポーネントをクリックすることで)選択します。スタイルサイドバーにて「3 段落」を選択し、プロパティのフォントグループにて、font-size を10pt に指定します(h3 コンポーネントは必須項目のfont-size 12pt となっているため、選択されませんでした)。

のスタイルカラムにて「1 段落」を選択します。プロパティのフォントグループにて、font-weight をbold にセットして、font-style をitalic に指定します。プロパティの色グループにて、color をgray に指定します。dateline の出力は以下のようになります: スタイルサイドバーのスタイルカラムにて「1 段落」を選択します。プロパティのフォントグループにて、font-weight をbold にセットして、font-style をitalic に指定します。プロパティの色グループにて、color をgray に指定します。dateline の出力は以下のようになります:

2006-04-01: Boston, USA

div コンポーネントに対して定義されたスタイルが、div コンポーネント内にある静的なテキストに対しても適用されます。

単一のコンポーネントに対してローカルスタイルを定義する

単一のコンポーネントに対して定義されたローカルスタイルは、SPS 内で定義されたより高次のコンポーネントをオーバーライドします。デザインにてheadline 要素を選択し、(スタイルプロパティの色グループ内にあるcolor プロパティに対して)navy 色を割り当てます。ローカルで定義されたプロパティ(color:navy) により、.header クラスに対して定義されたグローバルスタイル(color:green) がオーバーライドされます。

source 要素を囲んでいるdiv コンポーネントを選択します。スタイルサイドバーにて「1 段落」アイテムを選択し、(スタイルプロパティの色グループにある)color プロパティにてgray をセットします。スタイルプロパティのフォントグループにてfont-weight の値をbold へ変更します。これらの値は静的なテキストに対して適用されます。前のセクションにて、font-style のitalic が静的テキストの"Source:" に対して適用されたのを思い出してください。新たなプロパティ(font-weight:bold とcolor:gray)がfont-style:italic プロパティとは別に追加されます。

デザインビューにてsource 要素の(コンテンツ)プレースホルダを選択します。スタイルカラムにて「1 コンテンツ」を選択し、(スタイルプロパティの色グループにある)color プロパティにblack をセットします。その後、プロパティのフォントグループにあるfont-weight に対してnormal を選択します。新たなプロパティがsource 要素のノードに対応したコンテンツプレースホルダノードに適用され、
div コンポーネントに対して定義されたプロパティがオーバーライドされます(以下のスクリーンショットを参照)。

フォーマットを完成する

このセクションにてフォーマットを完成させるにあたり、synopsis 要素のdiv コンポーネントを選択し、ツールバーの定義済みのフォーマット コンボボックスにてp を選択します。この操作によりHTML のp 要素のスタイルがブロックに対して適用されます。HTML プレビューは以下のようになります:

NanoNull Inc Launches Version 2.0 of NanoPower

2006-04-01: Boston, USA

Nanonull Inc today launched version 2.0 of its market-leading NanoPower line of hardware and software. The highlights of the new version of NanoPower are improved chip design capabilities in NanoSoft, the software used to design computer chips; and higher precision sand-filtering processes in NanoMeld, the hardware in which the transformation from sand to silicon chips is executed.

Source: NewTech Online

操作が完了した後に、ファイルを保存してください。

(C) 2015-2021 Altova GmbH

リンク

- 次のセクション: 自動計算の使用
- 前のセクション: 静的コンテンツの挿入
- チュートリアル スタートページ
- CSS スタイルと作業

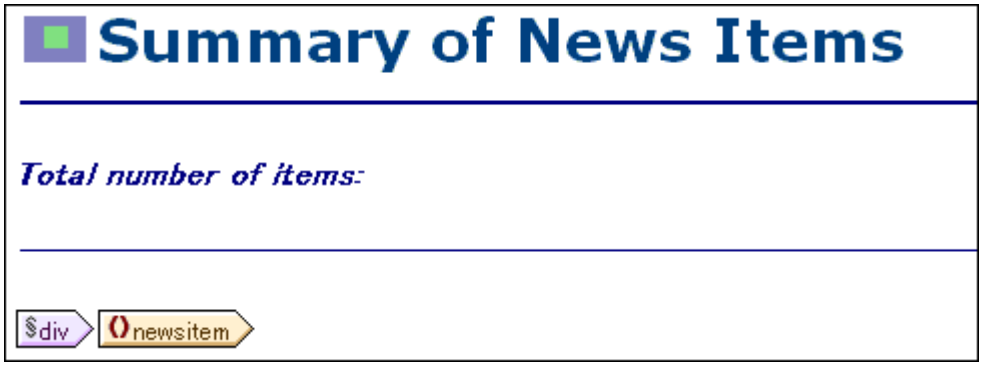
3.5 自動計算の使用

自動計算 は、XML 内にあるデータをベースに更なる情報を提示するためのパワフルなメカニズムです。このセクションでは、デザインに対して、XML ドキュメント内にあるニュースアイテムの総数と、ニュースアイテムによりカバーされた期間という、2つの情報を追加します。どちらの情報も XML ドキュメント内に記述されていませんが、利用可能なデータから計算し、操作することで得ることができます。

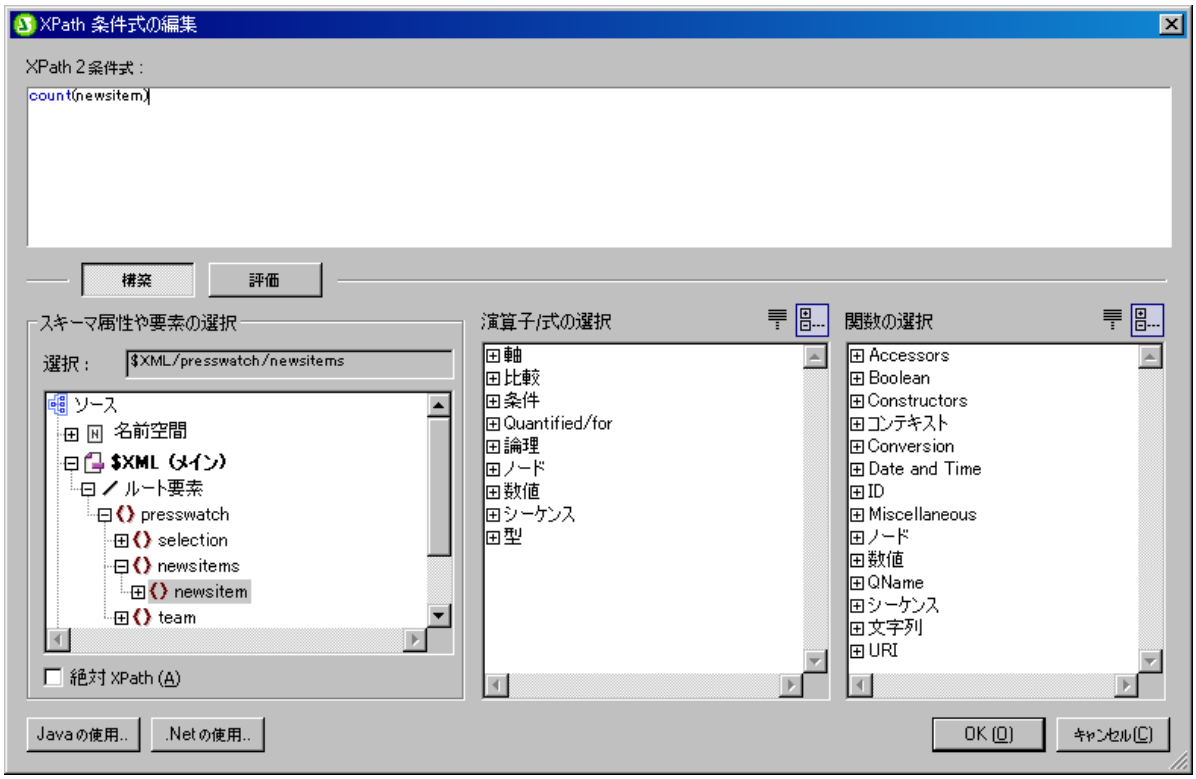
ニュースアイテムのノードを数える

デザインにて、以下を行います:

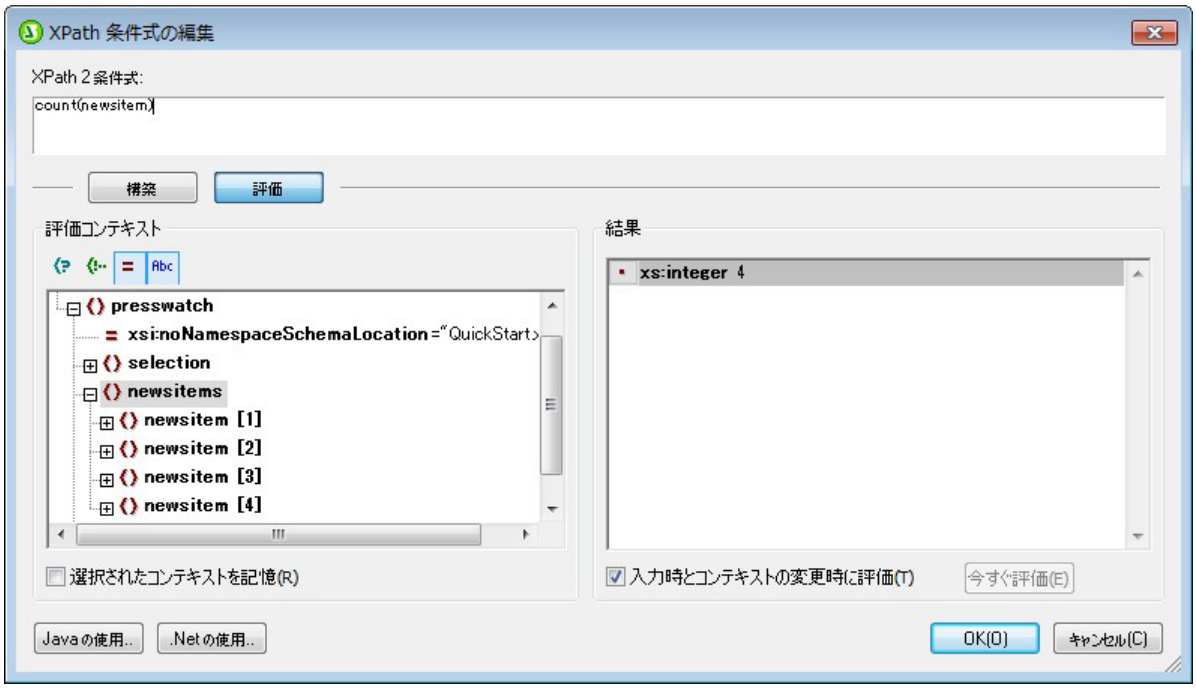
1. 以下のスクリーンショットに示されるように、空白行を挿入します(この場所に自動計算が配置されます)。Return キーを押下して新たな行を追加して、その下に水平線を追加します(以下のスクリーンショットを参照)。



2. 上にスクリーンショットに示されているように、"Total number of news items: " という内容の静的なテキストを入力します。
3. 静的テキストに対してローカルなスタイルルールを適用します。コンテンツのフォーマットに記述されている操作を行います。
4. コロンの後にカーソルを配置して、「**挿入 | 自動計算を挿入 | 値**」を選択します。XPath 式 の編集ダイアログ ダイアログが表示されます(以下のスクリーンショットを参照)。デザインを右クリックして、コンテキストメニューからコマンドを選択することもできます。

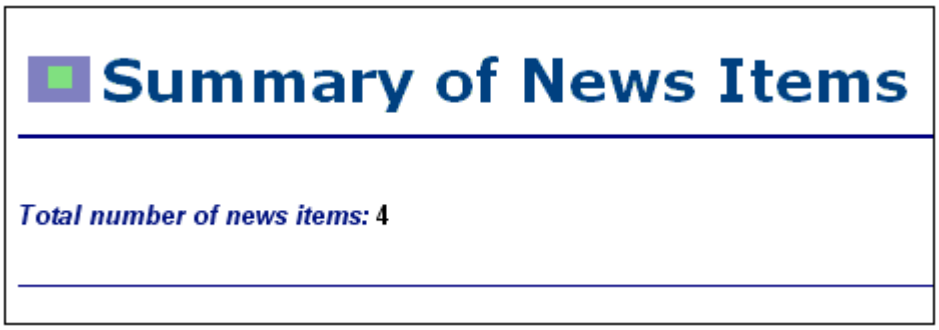


5. スキーマツリーにて newsitems がコンテキストノードとしてハイライトされていることに注目してください。条件式テキストボックスにて、count(newsitem) という条件式を入力するか、条件式テキストボックス以下にある入力ヘルパーペインを使用して条件式を構築することができます。関数のシーケンスグループにある count 関数をダブルクリックすることで条件式ペインに関数が入力されるので、関数のかっこ内にカーソルを配置して、スキーマツリーの newsitem ノードをダブルクリックしてください。**評価**ボタンをクリックすることで、XPath 条件式の戻り値を確認することができます。評価の結果が結果ペインに表示されます（以下のスクリーンショットを参照）。XPath 条件式の編集ダイアログに関する詳細については、次のセクションを参照してください：XPath 式を編集。



6. 「OK」をクリックして完了します。デザイン内のカーソル位置に自動計算が挿入されます（以下のスクリーンショットを参照）。ローカルスタイルにより自動計算をフォーマットすることができます。

HTML 出力 は以下のようになります:



ニュースアイテムによりカバーされた期間を表示する

ニュースアイテムによりカバーされた全ての期間は、一番古い記事の月日と、一番新しい記事の月日により求めることができます。以下にあるような XPath 条件式により、このような処理を行うことができます。以下に示される最初の条件式により、date ノードのコンテンツが単純に出力され、2番目の条件式により date ノードにある年や月だけが出力されるようになります。どちらの条件式も使用することができます。

- `concat(min(//date), ' to ', max(//date)).`
- `concat(month-from-date(min(//date)), '/', year-from-date(min(//date)), ' to ', month-from-date(max(//date)), '/', year-from-date(max(//date)))`

デザインにて、以下に示されるように静的テキストと自動計算が挿入されます。希望するローカルスタイルを適用することができます。

Total number of news items: =(AutoCalc)
Period covered by news items: =(AutoCalc)

HTML プレビューは以下のように表示されます:

Total number of news items: 4
Period covered by news items: 4/2006 to 5/2006

操作が完了した後にファイルを保存してください。

リンク

- 次のセクション: 条件を使用する
- 前のセクション: コンテンツのフォーマット
- チュートリアル スタートページ

3.6 条件の使用

QuickStart.xml では、各 newsitem 要素内にmetainfo 子要素が含まれており、その中に1つ以上のrelevance 子要素を含めることができます。SPS デザインでは、ユニークな relevance 要素値を追加するためのドロップダウンリストを持ったコンボボックスを作成することができます。Authentic View ユーザーがコンボボックスのドロップダウンリストからアイテムを選択すると、そのアイテムを値として XML ドキュメントのノードへ渡すことができます。条件を使用することで、(そのノードをルックアップすることにより) ユーザーの選択内容をテストして、ユーザーの各選択に対して対応する処理や表示を行うことができるようになります。このセクションでは、ユーザーの選択にマッチした relevance 要素が含まれるこれらニュースアイテムを表示するための条件テンプレートを作成します。

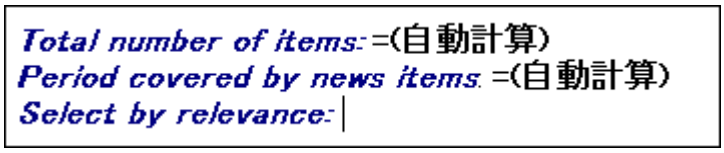
以下のように操作を行います:

1. Authentic View ユーザーが byrelevance 値を選択できる コンボボックスを作成します。コンボボックスのドロップダウンリストに表示される値は、XPath 条件式により取得され、relevance ノードのユニークな値からなるリストを動的に作成します。
2. newsitem 要素の周りに条件を挿入します。この条件により、byrelevance ノードのコンテンツにマッチした relevance 要素を含む全ての newsitem 要素が選択されます。条件の分岐に囲まれたコンテンツのことを、条件テンプレートと呼びます。
3. 条件テンプレート内にて、ニュースアイテムの各 relevance ノードを表示します。
4. byrelevance 要素にマッチした(relevance 要素のリスト内にある)relevance 要素をハイライトします。この操作は、relevance 要素に対する条件を作成し、条件テンプレートに対してフォーマットを適用することで実現できます。
5. newsitem 要素に対する条件内に、全てのニュースアイテムを選択する分岐を挿入します。

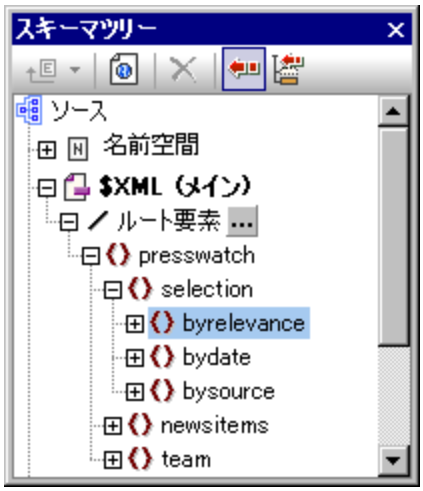
ユニークなノード値を選択するコンボボックスを作成する

XML ドキュメントでは、ユーザーの選択が `/presswatch/selection/byrelevance` 内に含まれます。コンボボックスとして作成するノードです。手順は、以下のとおりです:

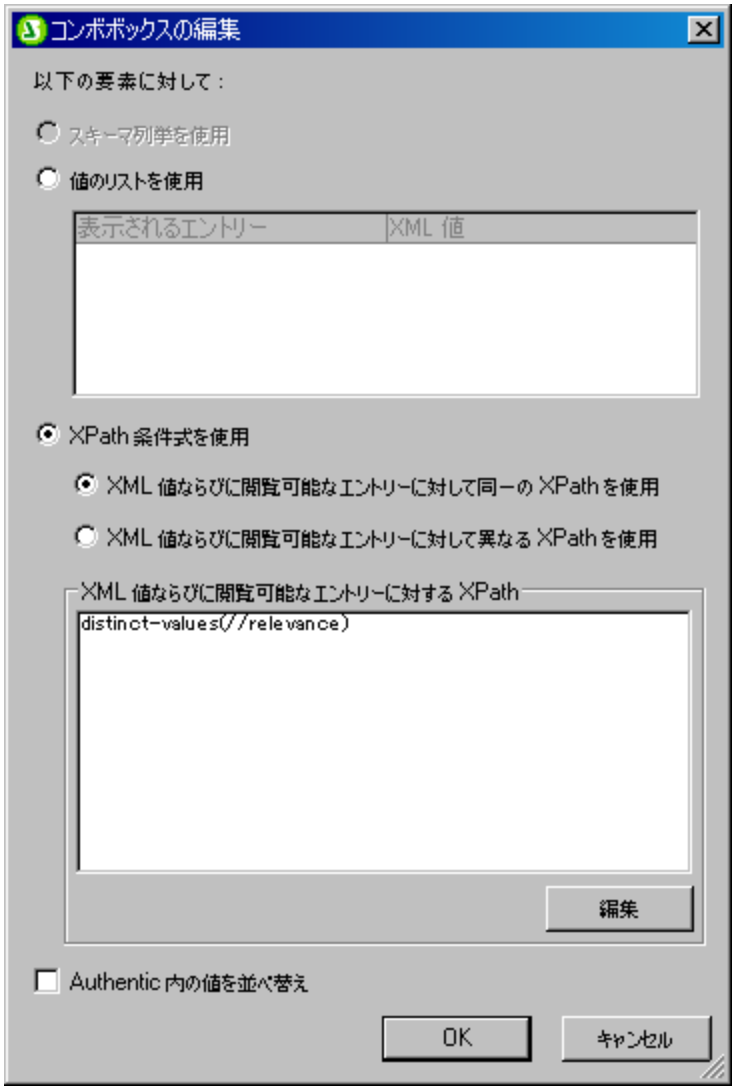
1. ドキュメントの先頭にある2番目の自動計算 の下に"Select by relevance: " という静的テキストを挿入します(以下のスクリーンショットを参照)。



2. (以下のスクリーンショットにて示される) スキーマツリー サイドバー から、byrelevance ノードをドラッグし、先ほど新たに入力した静的テキストの後にドロップします。

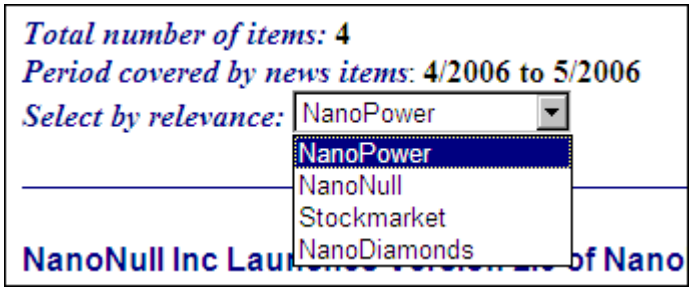


3. コンテキストメニューが表示されるので、コンボボックスの作成を選択します。以下のダイアログが表示されます。



4. コンボボックスの編集ダイアログにて、「XPath 条件式を使用」を選択し、「XML 値ならびに閲覧可能なエントリーに対して同一の XPath を使用」を選択します。XML 値ならびに閲覧可能なエントリーに対する XPath にて、distinct-values(//relevance) という XPath 条件式を入力します。この条件式により、XML ドキュメント内における全ての relevance 要素のユニークな値が選択されます。
5. 「OK」をクリックして完了します。コンボボックスが挿入され、デザインは以下のように表示されます:

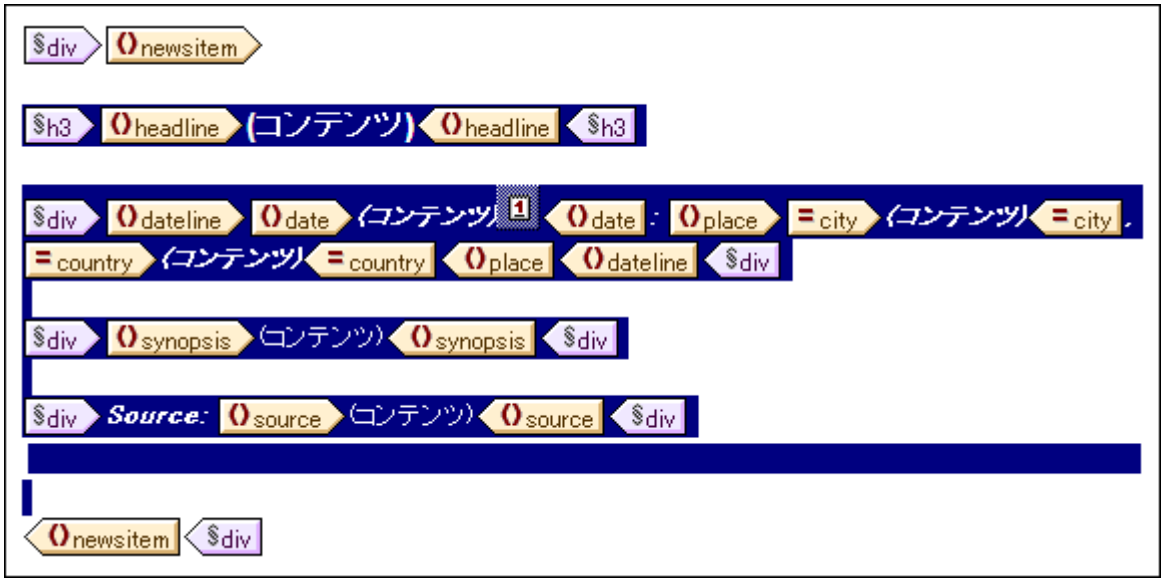
6. Authentic View へ切り替えます。コンボボックスの矢印をクリックすると、全ての relevance ノードにあるユニークな値がリスト表示されます。XML ドキュメントの内容が合致するか確認してみてください。このリストは動的に生成され、XML ドキュメントに新たに加えられた relevance 値も反映されます。



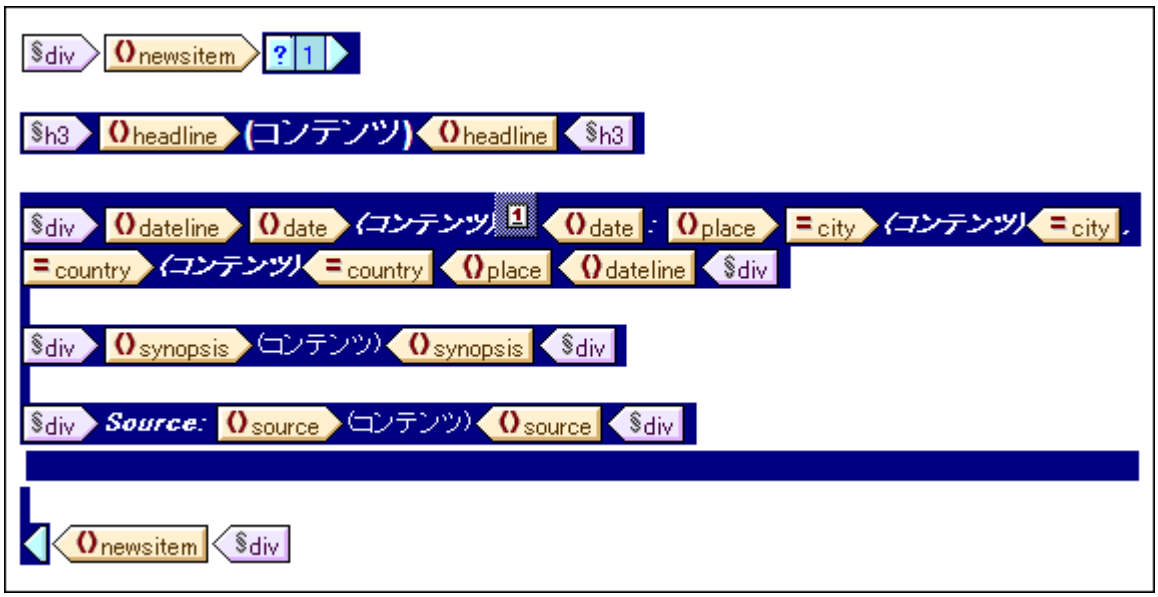
条件を挿入することで選択された relevance に対するニュースアイテムを表示する

条件により、ユーザーにより選択された（そして /presswatch/selection/byrelevance 要素に渡された）値と等しい metainfo/relevance 要素を持った metainfo/relevance 要素を持った newsitem 要素が選択されます。以下のように条件を挿入します:

1. デザインの newsitem におけるコンテンツを選択します（以下のスクリーンショットにおけるハイライト箇所）。



2. メニューコマンド（またはコンテキストメニューコマンド）から「 タグで囲む | 条件 」を選択します。これによりXPath 式の編集ダイアログが表示されます。
3. metainfo/relevance=/presswatch/selection/byrelevance という条件式を入力します。条件式により、現在の newsitem の metainfo/relevance 子孫が、（ユーザーの選択値である）/presswatch/selection/byrelevance に一致する場合に真となります。
4. 「OK」をクリックします。newsitem 要素のコンテンツの周りに条件が作成されます（以下のスクリーンショットを参照）。

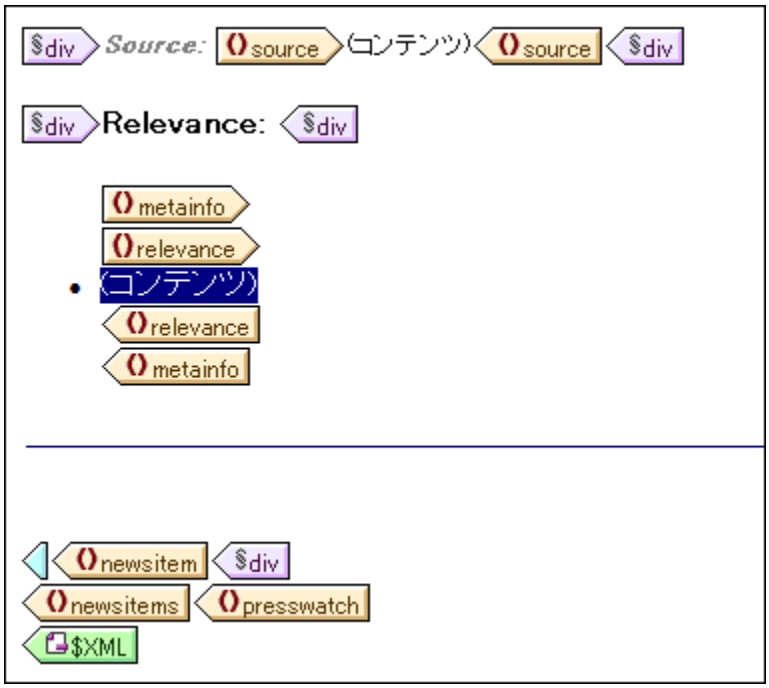


この段階では条件分岐が1つしかない点に注目してください。条件の結果が真となるニュースアイテムが表示され、条件の結果が真とならないニュースアイテムは表示されません。この場合、条件はフィルターとして機能します。このセクションの後半では、2番目の条件分岐を追加します。

relevance ノードをリストとして挿入する

各 newsitem 要素のrelevance ノードを表示するためには、以下の操作によりこれらノードをデザイン内に挿入します（以下のスクリーンショットを参照）：

1. source 要素の div タグと条件テンプレートの終了との間に改行を作成します。
2. "relevance:" という静的テキストを入力し、その回りに定義済みフォーマットの div を作成します。
3. スキーマツリー サイドバーのルート要素ツリーからrelevance 要素をドラッグし、relevance: 静的テキストの下にドロップします。
4. リストを作成します（コンテキストメニューが表示されるので、箇条書きの作成を選択し、目的のフォーマットを選択します）。
5. リストのコンテンツに対してテキストのフォーマットを指定します。デザインは以下のようになります：

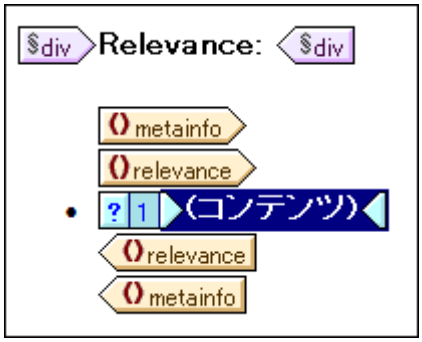


ここでAuthentic Viewにて、コンボボックスの選択肢を変更することでrelevance の選択結果を確認してください。 コンボボックスを使用して選択を変更してください。

選択されたrelevance 要素を太字にする

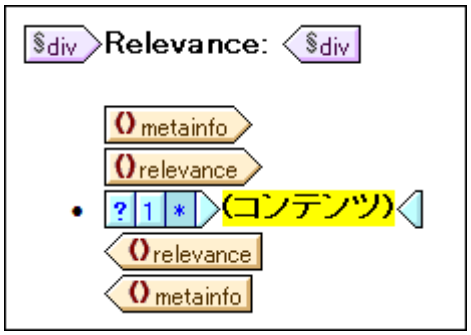
ニュースアイテムによっては、複数のrelevance 要素を含むものもあります。そのような場合、ユーザーの選択に合致したrelevance をハイライトすることで、デザインを改善することができます。以下の操作によりデザインを拡張します:

1. デザイン内の relevance 要素を選択します。
2. .=/presswatch/selection/byrelevance という XPath 条件式とともに、条件を挿入します。この操作により、分岐が1つの条件が作成され、byrelevance 要素にマッチする relevance 要素が選択されます(以下のスクリーンショットを参照)。



3. コンテンツプレースホルダを選択し、ローカルのフォーマットとして(フォントグループから)太字を表す bold と、(色グループから)背景色の background-color に yellow を(スタイルサイドバーにて)指定します。
4. 条件を右クリックして、コンテキストメニューから**条件分岐のコピー**を選択します。
5. XPath 式の編集ダイアログにて、(ダイアログ右上にある)「それ以外」のチェックボックスをクリックします。
6. 「OK」をクリックして完了します。新たな(その他)の条件分岐が作成されます(以下のスクリーンショットを参照)。この条件分岐では、byrelevance 要素にマッチしない全ての relevance 要素が選択されます。

(C) 2015-2021 Altova GmbH



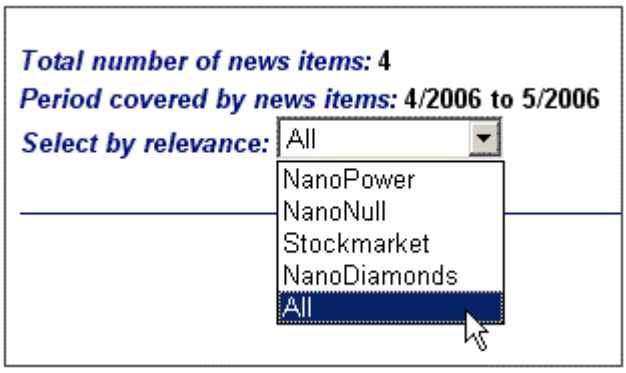
7. その他の条件分岐にあるコンテンツは、最初の条件分岐からのコピーとなります。コンテンツのプレースホルダには太字で黄色の背景色が与えられます。コンテンツプレースホルダを選択し、これらのフォーマット情報を削除してください。

これで、relevance 要素に対して以下のような動作を行う2つの条件分岐を持った条件が作成されました: relevance のコンテンツが/presswatch/selection/byrelevance のコンテンツにマッチすればrelevance のコンテンツが太字に黄色の背景色で表示され、それ以外の場合は、(2番目の条件分岐により) 通常のフォーマットでコンテンツが表示されます。Authentic Viewにて動作を確認してください。

コンボボックスを修正して、2番目の条件分岐を挿入する

コンボボックス内の、Authentic View ユーザーがbyrelevance の値を選択するコンボボックスには、全てのニュースアイテムを選択するオプションが表示されていません。以下の操作により、全てのアイテムを選択するオプションを加えることができます:

1. デザインビューにてコンボボックスを選択します。
2. プロパティサイドバーにてコンボボックスが選択された状態で、(**コンボボックス**グループ内にある) コンボボックス入力値プロパティの**編集**ボタンをクリックします。
3. コンボボックスの編集 ダイアログにて、XPath 条件式を distinct-values(//relevance) から、distinct-values(//relevance), 'All' へ変更します。この操作により、"All" という文字列が XPath 条件式により返されるアイテムのシーケンスに追加されます。
4. Authentic View にてコンボボックスのドロップダウンリストの選択肢を確認します (以下のスクリーンショットを参照)。(*下のスクリーンショット*).



ユーザーが"All" を選択することにより、(All) という値が、ノード /presswatch/selection/byrelevance へ渡されます。byrelevance ノードに "All" という値が含まれる場合、全てのニュースアイテムが表示されるようになります。

ニュースアイテムテンプレートを表示する条件にはmetainfo/relevance=presswatch/selection/byrelevance という分岐だけが与えられています。metainfo/relevance ノードにAll という値は無いため、byrelevance ノードにて All という値が選択されても、ニュースアイテムは表示されません。そこで、All という値に対してマッチする2番目の条件分岐を作成する必要があります。条件の結果が真となった場合に、条件分岐にあるテンプレートが表示されます。以下の操作をおこなってください!

1. デザインビューにて、ニュースアイテムの条件を選択します。
2. 条件を右クリックして、コンテキストメニューから**条件分岐のコピー**を選択します。
3. XPath 式 の編集ダイアログに`/presswatch/selection/byrelevance='All'`と入力します。
4. 「OK」をクリックして完了します。2番目の分岐が作成されます。

2番目の条件分岐には、最初の分岐と同一のコンテンツが含まれます。 ユーザーの選択がAll の場合、、2番目の分岐にはニュースアイテムテンプレートの出力を行います。

操作が完了した状態でファイルを保存してください。

リンク

- 次のセクション: グローバルテンプレートとその他のコンテンツの使用
- 前のセクション: 自動計算の使用
- チュートリアルスタートページ
- 条件

(C) 2015-2021 Altova GmbH

3.7 グローバルテンプレートとその他のコンテンツの使用

グローバルテンプレート を使用することで、グローバルな要素の処理を指定することができます。この機能により、(1つの場所で定義された)グローバルテンプレートを、スタイルシート内の複数の場所で使用することができるようになります。グローバルテンプレートは2種類の方法で使用することができます:

- グローバルテンプレートのルールをローカルテンプレートへコピー。
- (メインテンプレート内にある)ローカルテンプレートから、ローカルにあるノードの処理をグローバルテンプレートへ渡すことができます。グローバルテンプレートが実行されると、メインテンプレートにて処理が再開されます。この場合、グローバルテンプレートが、メインテンプレートから呼び出された、または使用されたことになります。

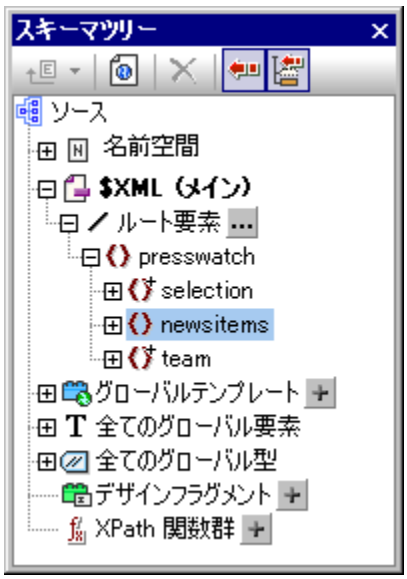
メインテンプレートからグローバルテンプレートを呼び出すには、2つのメカニズムがあります:

- ローカルテンプレートからグローバルテンプレートへの参照。
- メインテンプレート内にあるその他のコンテンツ (rest-of-contents) 処理命令を、現在の要素の子孫要素(つまり現在の要素から見たその他のコンテンツ)へ適用。子孫要素に対応したグローバルテンプレートが存在する場合、その要素に対してグローバルテンプレートが適用されます。そうでない場合、内蔵のテンプレートがその要素に対して適用されます。要素に対する内蔵のテンプレートにより子要素が処理され、要素のテキストコンテンツが出力されます。結果として、子孫要素にある全てのテキストコンテンツが出力されます。要素の属性は出力されないという点に注意してください。

このセクションでは、その他のコンテンツ処理命令とグローバル要素のmember に対するグローバルテンプレートを使用します。

rest-of-contents (その他のコンテンツ)処理命令を挿入する

スキーマの大まかな構造を以下のスクリーンショットに示します。



ドキュメント要素のpresswatch には、(i) selection、(ii) newsitems、(iii) team という3つの子要素が含まれています。これまで作成したメインテンプレートでは/presswatch 要素が処理されます。presswatch 要素内ではnewsitems 要素だけが処理されます。selection ならびにteam 要素は(selection はnewsitems 要素内部で処理されているものの)presswatch 要素内部で処理されません。presswatch へその他のコンテンツ処理命令を挿入することで、selection ならびにteam 要素が処理されるようになります。

newsitems とpresswatch の終了タグ間にカーソルを配置し、メニューコマンドまたはコンテキストメニューから挿入 | その他のコンテンツ を選択することで、デザインへその他のコンテンツ処理命令が挿入されます。その他のコンテンツプレースホルダが挿入されます(以下のスクリーンショットを参照)。

HTML プレビューを確認すると、テキストの文字列が表示されているはずです(以下のスクリーンショットを参照):

内蔵のテンプレートを selection ならびにteam 要素へ適用した結果、この文字列が表示されます。要素に対応した内蔵のテンプレートが子要素の処理を行います。テキストノードに対する内蔵のテンプレートにより、テキストノード内にてテキストが出力されます。これら2つの内蔵テンプレートを組み合わせることで、selection ならびにteam 要素以下にある全ての子孫ノードのテキストコンテンツが出力されます。selection/byrelevance から得られた All というテキストが表示され、team/member 子孫ノードのfirst、last、email 内にあるテキストが、ドキュメント内の順序により出力されます。member のid 属性は出力されていないことに注目してください(属性はmember の子とは見なされません)。

selection に対してグローバルテンプレートを作成する

selection のコンテンツを出力に含める必要は無いため、selection に対して空のグローバルテンプレートを作成し、そのコンテンツが処理されないようにします。以下の操作を行なってください:

1. デザインビューが表示されている状態で、スキーマツリーサイドバー内にある全てのグローバル要素のselection を右クリックします。
2. コンテキストメニューから、**グローバルテンプレートの作成/削除**を選択します。selection に対するグローバルテンプレートが作成されます(以下のスクリーンショットを参照)。

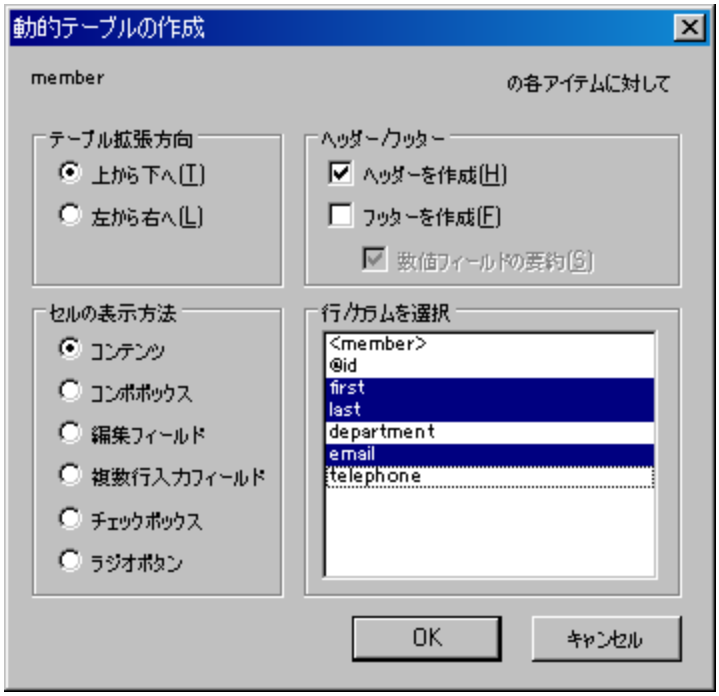
3. グローバルテンプレートにてコンテンツプレースホルダをクリックし、キーボードの **Delete** キーを押下します。コンテンツプレースホルダが削除されます。
4. HTML プレビューをチェックします。All というテキストが出力されたテキストに表示されなくなります(以下のスクリーンショットを参照)。

selection のグローバルテンプレートが空になっているため、selection の子要素が処理されなくなります。

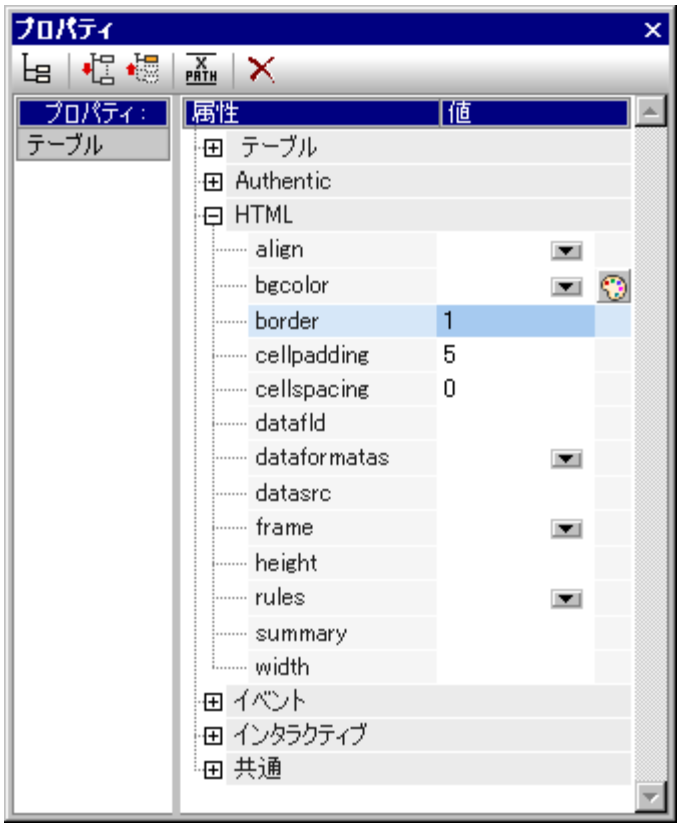
team/member に対してグローバルテンプレートを作成する

ここでメンバーの詳細情報を表示するテーブルを作成します。team 要素に対するグローバルテンプレート内にテーブルを作成します。以下の操作を行なってください:

1. team 要素に対するグローバルテンプレートを作成します（スキーマツリーサイドバーにある全てのグローバル要素にて team を右クリックし、**グローバルテンプレートの作成/削除**を選択します）。
2. 全てのグローバル要素リストにて、team 要素を展開して、member 子要素を（デザイン内の）team グローバルテンプレートへドラッグします。
3. team のグローバルテンプレートへドロップした際にコンテキストメニューが表示されるので、**テーブルの作成**を選択します。動的テーブルの作成ダイアログが表示されます（以下のスクリーンショットを参照）。



4. 属性/要素のリストにて、@id、department、telephone の選択を外し、「OK」をクリックします。動的テーブルが作成されます。
5. カーソルをテーブルボディーのセルに配置して、プロパティサイドバーのテーブルが選択されている状態で、以下のスクリーンショットに示されているようにテーブルのプロパティを変更します。



6. その他の値を、プロパティならびにスタイルサイドバーにて指定します。例えば、ヘッダー行へカーソルを配置した状態で、スタイルの trow を選択し、bgcolor プロパティの値を指定します。静的テキストの文字列であるヘッダーを編集することもできます。team 要素のコンテンツプレースホルダが表示されている場合、コンテンツ部分を削除してください。

HTML プレビューでは、テーブルが以下のように表示されます:

First	Last	Email
Andrew	Bentinck	a.bentinck@nanonull.com
Nadia	Edwards	n.edwards@nanonull.com
John	Edwards	j.edwards@nanonull.com
Janet	Ashe	j.ashe@nanonull.com

リンク

- 次のセクション: おしまい!
- 前のセクション: 条件を使用する
- チュートリアルスタートページ
- 出力の構造

(C) 2015-2021 Altova GmbH

3.8 おしまい！

おめでとうございます！これでチュートリアルは終了です。以下にあるような、SPS を作成する際に最も重要な内容を学習したことになります：

- ドキュメントの構造を作成する方法、（メインのテンプレート とグローバルテンプレート）。
- 動的 と静的 SPS コンポーネントを使用して、動的ならびに静的なコンテンツをデザイン内部へ挿入する方法。
- 外部スタイルシート、グローバルスタイルルール、とローカルスタイルルールにて、CSS スタイルを使用する方法。
- 自動計算 を使用して、XML データ内部の情報を取得する方法。
- 条件 を使用して、XML データのフィルタリングや、XML から得られたデータに従った出力を得る方法。
- グローバルテンプレート とその他のコンテンツを使用する方法。

これら機能の詳細な説明については、以下にあるセクションを参照ください：

- SPS ファイル: コンテンツ
- SPS ファイル: 構造
- SPS ファイル: 高度な機能
- SPS ファイル: プレゼンテーション
- SPS ファイル: 追加機能

上記のセクションでは、このチュートリアルでは触れられなかったその他の StyleVision 機能についても解説されます。

SPSの使用する

SPS が完成すると、以下にある SPS の機能を使用することが可能になります:

- XMLSpy のAuthentic View で、または Authentic Desktop にて、ドキュメントを編集する（Enterprise と Professional エディションには、サイドバーやテキスト状態アイコンが表示されない Authentic View プレビュータブが搭載されます）。これら2つの製品には Authentic View が搭載されており、サイドバーやコンテキストメニューを使用することができます。XMLSpy や Authentic DesktopAuthentic Desktop 内の Authentic View にて QuickStart.xml を編集するには、XML ファイルを MyQuickStart.sps へ関連付け、Authentic View へ切り替えます。
- XML ファイルHTML 出力へ変換する XSLT スタイルシートを生成する。「ファイル| 生成されたファイルの保存」コマンド、またはコマンドラインからXSLT スタイルシートを生成することができます MyQuickStart.sps から XSLT スタイルシートを生成し、スタイルシートを使用して QuickStart.xml の変換を行うことができます。

リンク

- 次のセクション: コンテンツのフォーマット
- 前のセクション: 動的なコンテンツの挿入
- チュートリアル スタートページ

(C) 2015-2021 Altova GmbH

4 使用方法の概要

目的

StyleVision で作成したSPS ドキュメントは、大きく分けて2つの方法で使用することができます:

- Authentic View における XML ソースドキュメントの表示方法を制御し、Authentic View インターフェースから XML ドキュメントやデータベースへのデータ入力を行う。
- HTML とRTF 出力の XSLT スタイルシートを生成する。

SPS を使用することで、XML ドキュメント編集を行い、、HTML とRTF 出力を生成することができます。更に、生成された XSLT スタイルシートは、SPS のそれと同一のスキーマをベースにしたその他の XML ドキュメントを変換するのにも使用することができます。

SPS 作成のステップ

SPS を作成するためのステップ概要を以下に示します:

1. 新たに作成された空の SPS に対してスキーマを割り当てます。スキーマとは、(i) スキーマファイル(DTD または XML スキーマ)、(ii) DB から生成された XML スキーマ(Enterprise ならびに Professional Edition のみ)、(iii) XBRL タクソノミをベースにしたスキーマ(Enterprise Edition のみ)、(iv) ユーザー定義スキーマ(StyleVision にて直接作成) のいずれかになります。この操作はデザイン概要サイドバーにて行うことができます。または、**ファイル | 新規作成** コマンドにより、スキーマから直接新規 SPS を作成することもできます。
2. SPS に作業用 XML ファイルを割り当てます。作業用 XML ファイル により、 Authentic View と出力プレビューを生成する時に SPS にて処理される XML データが与えられます。作業用 XML ファイル が デザイン概要サイドバー にて割り当てられ、新たなスキーマに対する作業用 XML ファイルにより出力を StyleVision でプレビューすることができます。
3. 使用する XSLT のバージョンを選択します。
4. Internet Explorer の互換性 を選択して、インストールされている Internet Explorer のバージョンに合致するようにします。
5. デザインビュー では、様々なデザインコンポーネントにより SPS ドキュメントをデザインすることができます。デザインの作業 はドキュメント構造の作成と、表示プロパティの定義から構成されます。印刷形式の出力 が必要な場合、更に出力形式のプロパティを指定することもできます。
6. Authentic View と出力のテストを行います。デザインへの修正が必要とされる場合、修正を行った後に SPS を再度テストします。
7. XSLT ファイルや出力ファイル が必要とされる場合、それらのファイルを 生成します。
8. 必要な場合は、テンプレート XML ファイルを割り当てます。テンプレート XML ファイル により、SPS を使用した Authentic View で編集可能な新規 XML ドキュメントの初期データを与えることができます。
9. 複数の Authentic View ユーザーが使用できるように、SPS を デプロイ します。

関連項目

- SPS ファイル構造
- データベースとの作業

4.1 SPS とソース

新規 SPS ファイルを作成する

SPS ドキュメントを新たに作成するには、「 ファイル | 新規作成 (Ctrl+N) 」コマンド以下のオプションを選択するか、標準 ツールバーにある**新規デザイン**アイコン をクリックします。新規 SPS ドキュメントが作成され、デザインビューに表示されます。新規ドキュメントには、SPSX.sps という仮の名前が与えられ、X にはアプリケーションが開始されてから作成された新規ドキュメントの順番に応じた整数が与えられます。

新規 SPS ドキュメントを作成した後には SPS のソースファイルを割り当てる必要があります。

SPS に対してソースファイルを割り当てる

SPS に対して割り当てることのできるソースファイルは以下の 3 種類あります：

- スキーマソース
- 作業用 XML ファイル
- テンプレート XML ファイル

これらソースファイルの割り当ては、デザイン概要サイドバーにて行うことができます。割り当てを行う方法については、デザイン概要 のセクションで記述されます。各種ソースファイルに関する重要な事柄を以下に示します。

スキーマソース

デザインドキュメントの構造を作成するには、SPS へ少なくとも1つのスキーマソースファイルを割り当てる必要があります。スキーマソースは、デザイン概要サイドバー内で割り当てられます。スキーマは、XML スキーマファイル（.xsd ファイル）、XML ファイルにより生成された XML スキーマ、DB ファイルにより生成された XML スキーマ、DTD、またはユーザー定義スキーマであることができます。各スキーマのために、任意の作業用 XML ファイルを1つ、そしてオプションのテンプレート XML ファイルを1つ割り当てることができます。

メモ SPS や、SPS から生成された XSLT スタイルシートへ名前空間を追加したい場合、SPS がベースとしている XML スキーマ内にあるトップレベルの schema 要素へ名前空間を追加する必要があります。

作業用 XML ファイル

SPS は、作業用 XML ファイルを割り当てることができます。作業用 XML ファイルから StyleVision 内部の出力プレビューで使用される XML データソースが与えられるため、割り当てられているスキーマに対して妥当である必要があります。作業用 XML ファイルは、デザイン概要サイドバーにて割り当てられます。

テンプレート XML ファイル

SPS は、テンプレート XML ファイルを割り当てることができます。テンプレート XML ファイルにより、SPS が StyleVision 以外の製品に搭載された、Authentic View で開かれた際に作成される、新規 XML ドキュメントの初期データが与えられます。テンプレート XML ファイルは、デザイン概要サイドバーにて割り当てられ、割り当てられたスキーマに対して妥当である必要があります。

関連項目

- スキーマツリー

(C) 2015-2021 Altova GmbH

4.2 デザインの作成

SPS では以下のような事柄を指定することができます:

1. コンテンツ (XML ドキュメントまたは DB から得られた) のうち、どれが出力されるか。更に、コンテンツを直接 SPS へ挿入することで、出力に含めることができます。
2. 出力の構造がどのように行われるか、および
3. 出力の様々なパーツに対して適用される表示 (フォーマット) プロパティの内容 です。

出力のコンテンツ

出力のコンテンツは以下のアイテムから得られます:

1. SPS が適用される XML ドキュメント または DB スキーマツリー サイドバー内の関連するスキーマツリーから目的の XML データノードをドラッグし、SPS 内の目的の場所にドロップすることで、XML ドキュメント から得られたコンテンツが SPS に含まれます。
2. アプリケーションからアクセスすることのできる、拡張された (つまり、StyleVision または Authentic View 製品)。自動計算内にある XPath 2.0 の doc() 関数を使用することで、外部 XML ドキュメントソースから得られたコンテンツへアクセスすることができます。XPath 条件式内にある doc() 関数によりアクセスされた XML ドキュメントが、スキーマソース の関連付けにより参照されている必要はありません。
3. SPS そのもの。テキストならびに (イメージやテーブルといった) その他のコンテンツは、キーボードや GUI に備わっている機能を使用することで、SPS 内に直接挿入することができます。これら入力は XML ドキュメントとは独立したものです。.
4. XPath 条件式により処理された動的 (XML ソースの) データ。処理は通常 自動計算により行われます。
5. HTML 出力に対して、コンテンツ生成のために JavaScript 関数 を使用することができます。

出力の構造

SPS デザインでは、以下の方法により 出力の構造 を指定することができます: (i) XML ドキュメントの構造とは独立した 入力レベルテンプレート (StyleVision のメインのテンプレート) にて出力構造を指定、(ii) (StyleVision のグローバルテンプレート 等の) 様々なノードに対してテンプレートルールを宣言し (iii) これら2つのアプローチを組み合わせたもの。デザインビューでは メインのテンプレート と グローバルテンプレート を使用することで、出力ドキュメントに対して使用する目的の構造を取得することができます モジュール SPS と デザインフラグメント を使用することで、SPS に更なる柔軟性が追加されます。

出力の表示 (フォーマット)

デザインビューでは、CSS スタイルにより表示プロパティがデザインコンポーネントへ適用されます。スタイルの定義はコンポーネントに対してローカルレベルで行うことができるほか、ドキュメントレベルで宣言された HTML セレクターや、外部 CSS スタイルシートにて宣言された HTML セレクターに対して行うことができます。更に 定義済みのフォーマット を使用することで、特定の HTML 要素をコンポーネントに対して適用することができます。表示プロパティの指定に関する詳細については、表示の方法のセクションを参照ください。

関連項目

- SPS ファイル構造
- デザインビュー

4.3 XSLT と XPath のバージョン

SPS とはXSLT スタイルシートと同様の機能を持っています。各 SPS に対して 1.0、2.0、または3.0 のXSLT バージョンを指定する必要があり、適切なツールバーアイコンの XSLT 1.0 または XSLT 2.0 または XSLT 3.0 をクリックすることで選択を行うことができます。この選択により、2つのことが決定されます:

- StyleVision に搭載されているどの XSLT エンジンが変換に使用されるのか。StyleVision にはXSLT 1.0、XSLT 2.0、とXSLT 3.0 という異なるエンジンが搭載されています。
- どの XSLT (1.0、2.0、または3.0) の機能がインターフェース上に表示され、SPS で使用できるようになるのか。例えば、XSLT 3.0 ではXPath 3.0 が使用されるため、(XSLT 1.0 で使用される)XPath 1.0 または(XSLT 2.0 で使用される)XPath 2.0 に比べてより強力な機能を使用することができます。更に、目次機能といった SPS の機能は XSLT 2.0 とXSLT 3.0 でのみ利用することができます。

XSLT 変換

StyleVision におけるXSLT 変換は、(i) インターフェースにて 出力ビュー を生成するため、そして(ii) インターフェイス内部 と StyleVision Server から(HTML とRTF) の出力ファイルの生成と保存 を行うために使用されます。変換に使用されるXSLT エンジン (Altova XSLT 1.0、2.0、または3.0 エンジン) のバージョンは、SPS で選択されたのバージョンにより決定されます。

GUI におけるXSLT の機能

XSLT の各バージョンで利用することのできる機能の大半は、対応するXPath のバージョンに関連しています(XSLT 1.0 のためにXPath 1.0、XSLT 2.0 のためにXPath 2.0、XSLT 3.0 のためにXPath 3.0 が使用されます)。XPath 条件式は StyleVision 内の様々な場所 -特に 自動計算 と 条件テンプレート- で使用されており、インターフェースに備えられている機能を、XPath 条件式の構築に役立てることができます。選択されたXSLT のバージョンに従うかたちで、正しいXPath の機能をインターフェースで自動的に利用することができるようになります。

関連項目

- *生成されたファイル*

(C) 2015-2021 Altova GmbH

4.4 Internet Explorer の互換性

SPS デザインを(デザインビューにて)正確に表示し、プレビューを(Authentic View とHTML プレビュー)出力するには、StyleVision をお使いのコンピューター上にInternet Explorer (IE) をインストールする必要があります。サポートされているIE のバージョンを以下に示します:

- Internet Explorer 5.5 または以降。
- Internet Explorer 6.0 および以上では XML のサポートが改善されているため推奨されます。
- Internet Explorer 9 (IE9) 以上では、サポートされるイメージフォーマットの拡張や、新たな CSS スタイルなど、更に機能が追加されています。デザインにてこれらの新たな機能を使用する場合、IE9 の使用を検討したほうが良いでしょう。

IE9 の機能-StyleVision におけるサポート

以下にあるIE9 以降の機能がStyleVision でサポートされます:

- 新たなイメージフォーマットのサポート: TIFF、JPEG、XR、SVG(SVG ドキュメントは XML 形式で記述されており、SVG 名前空間内にある必要があります)。これらのイメージフォーマットは IE9 では表示されるものの、それより古いバージョンの IE では表示されません。出力にてサポートされているイメージのリストについては、イメージ型 と出力を参照ください。
- 以下にある、(CSS3 を含む)新たな CSS スタイルのサポート。これらスタイルは、Authentic View とHTML 出力に対してのみ適用されます。
  - background-clip
  - background-origin
  - background-size
  - box-sizing
  - box-shadow
  - border-radius (border-*-radius)
  - font-stretch
  - ruby-align
  - ruby-overhang
  - ruby-position
  - overflow-x, overflow-y
  - outline (outline-color, outline-style, outline-width)
  - text-align-last (partial)
  - text-overflow (partial)

- 新たに追加された CSS の長さに関する関数: calc()。
- 新たに追加された CSS の色に関する関数: rgba()、hsl()、hsla() 。
- 新たに追加された CSS の長さユニット: rem、vw、vm、vh、ch。
- IE9 にてサポートされる HTML5 要素を、ユーザー定義要素としてデザインへ挿入することができます。

デザインビューとIE バージョン

プロパティダイアログにてIE のバージョンを指定することで、特定のIE バージョンに互換性のあるかたちでデザインビューを使用することができます。以下の影響が発生します。

- 選択された IE のバージョンでレンダリングすることができる CSS スタイルは全て、StyleVision のスタイルサイドバーに表示されます(IE9 を選択して、サポートされている CSS スタイルとデザインインターフェースで使用する場合、IE9 がインストールされている必要があります)。例えば、IE9 がインストールされている状態で IE9 が互換バージョンとして選択されている場合、IE9 でサポートされている CSS3 をデザインインターフェース内で使用することができます。

- 選択された IE バージョンに対応する HTML 要素は、定義済みのフォーマット または ユーザー定義要素 として入力することができます。HTML 要素は Authentic View や HTML プレビューにて、インストールされた IE のバージョンで行われるのと同様の方法でレンダリングされます。

特定の IE バージョンに対してデザインビューをセットアップする

特定の IE バージョンに対してデザインビューをセットアップするには、メニュー今マンドから「**ファイル | プロパティ**」を選択し、出力タブにて目的の IE (互換) バージョンを選択します。詳細については「ファイル | プロパティ」を参照ください。

古いバージョンの SPS デザインと IE 9 の互換性

StyleVision をお使いのコンピューターに IE 9 以降がインストールされている状態で、古い IE バージョンに対して作成された SPS デザインを開くと、新たなバージョンが StyleVision により検知され、IE 9 互換の状態に変更するか尋ねられます。新たな互換性により、上記にある新たなデザインビューオプションが表示されます。デザインビュー、Authentic View と HTML 出力におけるドキュメントの外観は、(IE 9 での処理方法が異なる**テーブルカラム**を除いて) 変更されることなく表示されます。IE 互換から IE 9 互換への変更が行われた場合、テーブルカラムが期待されているとおりに生成されていることを確認してください。生成が期待通りに行われていない場合、プロパティダイアログにてテーブルカラムのプロパティを修正するか、それまで選択されていた IE の互換バージョンへ設定を変更し直す必要があります。

関連項目

- ファイル | プロパティ
- イメージ型と出力
- CSS スタイルと作業

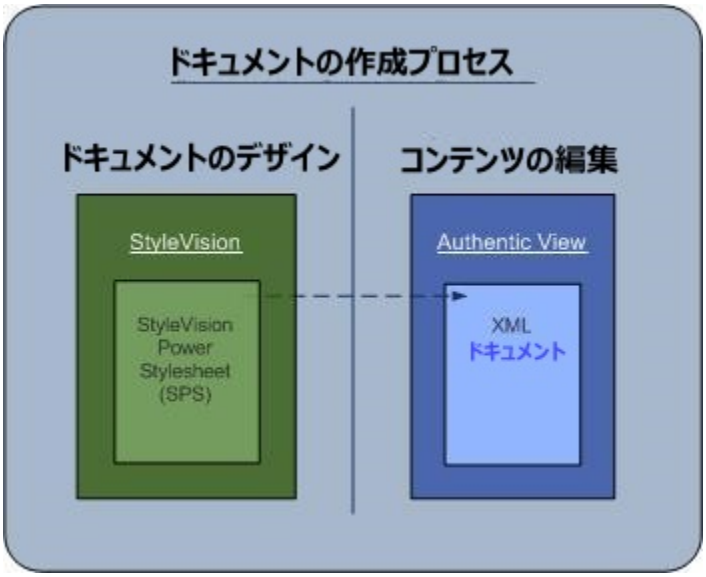
(C) 2015-2021 Altova GmbH

4.5 SPS と Authentic View

StyleVision で作成された SPS を使用することで、Altova 製品のドキュメントビューである Authentic View における XML ドキュメントのデータや、その表示方法を操作することができます。Authentic View を使用することで、XML の詳細を知らないユーザーでも、XML コンテンツの正確な入力や編集を行うことが可能になります。

Authentic View により使用されるドキュメントの作成や編集には、2つの異なるステップが必要になります:

- *ドキュメントのデザイン* : XML ドキュメントの Authentic View は、StyleVision におけるデザインを視覚的に表示したものです。デザインドキュメントとは SPS のことで、SPS は XML ドキュメントの処理だけではなく、Authentic View や最終的な出力フォーマットにおける表示に使用されるほか、Authentic View にて入力されたデータを XML ファイルや DB へ入力するためにも使用されます。

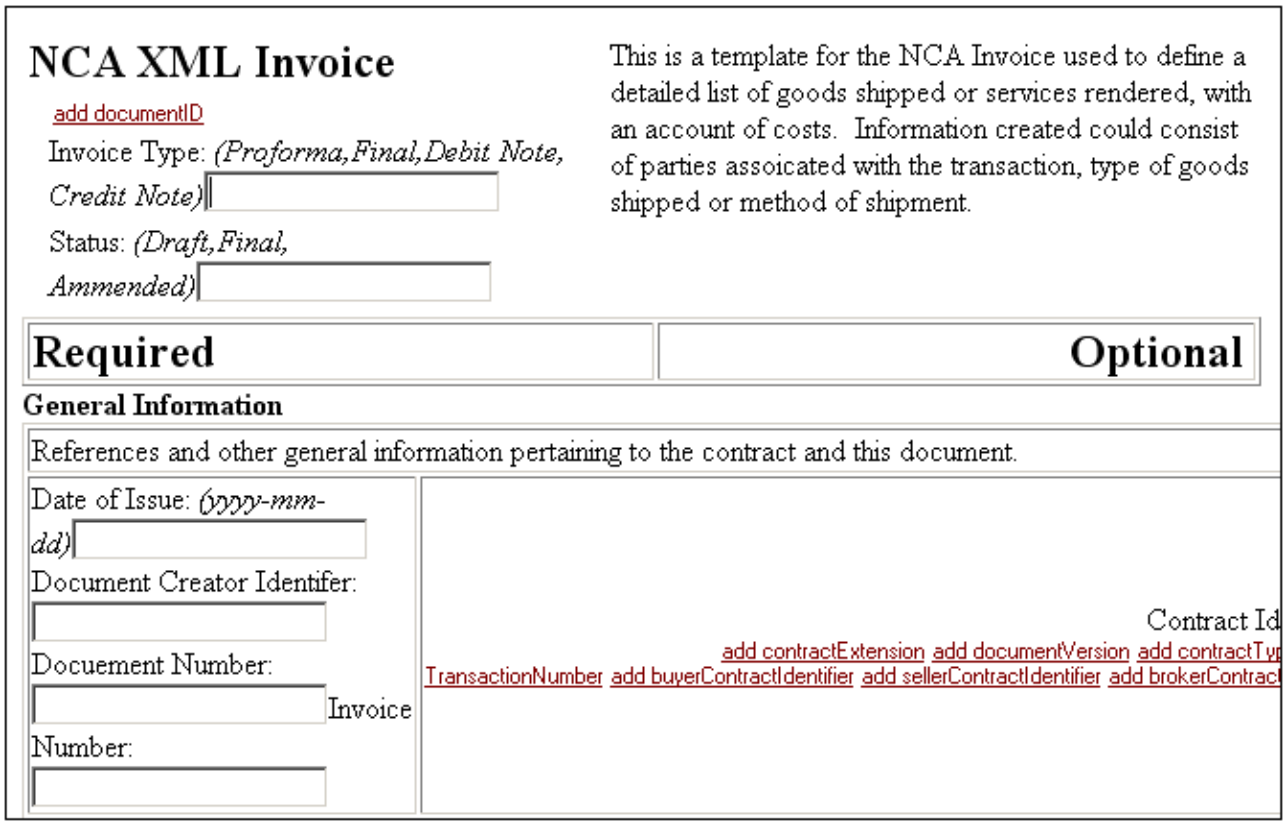


- *コンテンツの編集* : ドキュメントのデザインステップにて作成された SPS は、編集を行う XML ドキュメントへリンクされています（XML ドキュメントは、SPS がベースとしているスキーマに対して妥当でなければなりません）。SPS へリンクされている XML ドキュメントは、その他の Altova 製品にある、Authentic View にて表示されます。にて表示されます。新たな Authentic XML ドキュメントに対して SPS への関連付けを行うことで、テンプレートと SPS で指定されているコントロールを使った Authentic View での編集が可能になります。SPS に関連付けられている既存のドキュメントが開かれた場合、SPS に従い既存のデータが Authentic View にて表示され、 Authentic View にて編集を行うことができるようになります。

Authentic View の使用者が、XML やドキュメントのスキーマ構造について熟知している必要はありません。ドキュメントを Authentic View で表示することにより、技術的な理解を必要とすることなくコンテンツの編集ができるようになります。そのため、ユーザーフレンドリーな Authentic View を備えた SPS を作成することがデザイナーのタスクとなります。Authentic View の使用に関する詳細な情報については、 XMLSpy または Authentic Desktop のユーザーマニュアルにある、Authentic View ドキュメント を参照ください。

業界標準のスキーマのための SPS

Altova の Authentic View パッケージには、数多くの業界標準スキーマに対応した SPS が収められています。ユーザーはこれら標準仕様をベースにした XML ドキュメントを Authentic View にて作成することができます。以下のスクリーンショットは、NCA Invoice 仕様を Authentic View で表示したものの一部です。



アプリケーションフォルダー以下の**`Examples/IndustryStandards`** フォルダーに収められている業界のSPS は、簡単にカスタマイズすることができます。

4.6 StyleVision と Authentic の同期

StyleVision では、バージョンアップが行われるたびに、新たな機能が Authentic View に追加されてきました。以下の点に留意するようにしてください:

- より新しいバージョンの StyleVision で作成された SPS ファイルは、古いバージョンの Authentic View で処理することができない場合があります。
- (より新しいバージョンの StyleVision により作成された) SPS ファイルの機能は、対応するバージョン(またはそれ以降のバージョン)の Authentic View でのみ認識されます。

従って、新しいバージョンの StyleVision により SPS ファイルが作成された場合、デプロイされる Authentic View 製品 のバージョンも、StyleVision のそれと同一にしておく必要があります。例えば、**StyleVision 2008 リリース 2** により SPS ファイルを作成した場合、この SPS ファイルの編集を正しく行うには、**Authentic Desktop 2008 リリース 2** (または同一バージョンのその他の Authentic View 製品)を使用する必要があります。

あるバージョンの StyleVision で作成された SPS は、それ以降のバージョンの Authentic View 製品 により処理することができます。

デプロイされた SPS ファイルが新しいバージョンの StyleVision により修正された場合の同期の方法

SPS が何人もの Authentic View ユーザーに対してデプロイされている状態で、新しいバージョンの StyleVision により新たな Authentic View 機能が SPS へ追加された場合、以下のステップに記される方法で同期を行ってください:

1. 新しいバージョンの Authentic View のライセンスキーをデベロッパーが取得します。
2. 新しいバージョンの StyleVision と Authentic View により、SPS のテストを行います。
3. 新しいバージョンの Authentic View 製品 がすべての Authentic View ユーザーに配布されます。
4. 上にある3つのステップが完了した後に、修正された SPS を Authentic View ユーザーに対して配布します。

関連項目

- Authentic View

4.7 生成されたファイル

StyleVision では、「ファイル | 生成されたファイルの保存」コマンドまたは StyleVision Server を使用し、XSLT スタイルシート と出力ファイルを生成することができます。また、 XML を XSLT を使用して検証、または、変換するには、RaptorXML(+XBRL) Server を使用して行うことすることができます。

StyleVision からは以下のファイルを生成することができます:

- SPS デザインをベースにした XSLT スタイルシート。個別の XSLT スタイルシートは、HTML と RTF 出力のために生成されます。
- 作業用 XML ファイルを処理して生成されるファイルは、SPS から生成された XSLT スタイルシート を使用して SPS 内で割り当てられます。

出力のマークアップは SPS 内部に含まれています。出力に使用されるデータは XML ドキュメントまたは DB 内部に収められています。マークアップとデータは、XSLT スタイルシートにより結び付けられます。StyleVision の出力ビューでは、XSLT スタイルシートと実際の出力の両方をプレビューすることができます。

メモ SPS や、SPS から生成された XSLT スタイルシートに名前空間を追加する場合、SPS がベースとしている XML スキーマのトップレベルにある schema 要素にて名前空間を追加する必要があります。

Altova Web サイト: XML レポート

生成されたドキュメントに関する重要なポイントを以下に示します:

- *HTML 出力とスタイルシート:* (1) 生成された HTML ドキュメントのフォーマットとレイアウトは、StyleVision 内部で確認することのできる そして XML ドキュメントを Authentic View で表示したものとほぼ同一のものになります。(2) HTML ファイル内のデータ入力デバイス(テキスト入力フィールドやチェックボックスなど) では入力が受け付けられることはありません。これらデータ入力デバイスは、Authentic View にて XML データの入力を行うためのもので、HTML フォーマットでも同じように表示されはするものの、データ入力のために使用することはできません。

- *RTF 出力とスタイルシート:* (1) RTF デザインには印刷媒体にて使用される要件を定義する必要があります。これら(例えば表紙のデザインや左/右余白などの) 指定は、プロパティ サイドバーとデザインツリー サイドバー(2) データ入力デバイスが SPS にて使用される場合、これらデバイスは出来る限り RTF ページでもレンダリングされます。(例えばチェックボックスなど) データ入力デバイスを視覚的に表示しにくい場合、代替表示が使用されます。

RTF 出力

XML ドキュメントを SPS から生成された XSLT-for-RTF ファイルにより処理することで、ワンステップで XML ファイルから RTF 出力を生成することができます。RTF 出力のプロパティは SPS にて定義され、出力は RTF プレビューウィンドウにて確認することができます。RTF ファイルの保存を行うには、「ファイル | 生成されたファイルの保存」または StyleVision Server を使って生成を行う必要があります。

メモ 埋込型のプレビューに問題がある場合、StyleVision はプレビュードキュメントを外部アプリケーション(通常 Microsoft Word または Adobe Reader) にて開くように試みます。埋込型のプレビューに対するエラーメッセージが StyleVision にて表示されます。プレビュードキュメントが外部アプリケーションにて開かれた場合、一時出力ドキュメントを再度生成する前に、外部アプリケーションを終了する必要があります。外部アプリケーションにてドキュメントが開かれている場合、ファイルが他のプロセスにて使われているというエラーメッセージが表示されます。外部アプリケーションによりファイルがロックされるため、SPS デザインを閉じる前に外部アプリケーションを閉じる必要もあります。

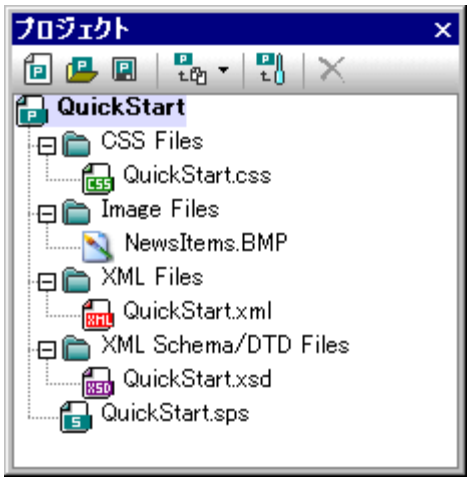
(C) 2015-2021 Altova GmbH

関連項目

- ファイルメニュー| 生成されたファイルの保存

4.8 StyleVision のプロジェクト

互いに関連するファイルはプロジェクト サイド バーに表示されるプロジェクトにて管理することができます(以下のスクリーンショットを参照)。この機能により、SPS のデザインを行なう際に、プロジェクト内のファイルへ簡単にアクセスすることが可能になります。例えば、SPS ファイルをプロジェクト サイド バーからデザインツリーサイド バーへドラッグすることでモジュールを作成し、イメージファイルを静的なイメージとしてデザインへドラッグし、CSS スタイルシートを外部スタイルシートとしてスタイルリポジトリサイド バーへドラッグすることができます。



プロジェクトの作成と保存

「**プロジェクト | 新規プロジェクトの作成**」コマンドにより、新たなプロジェクトが作成されます。新たに作成されたプロジェクトには、異なる種類のファイルを含めるためのフォルダーが複数作成されます(上のスクリーンショットを参照)。フォルダーのプロパティダイアログから、フォルダーに対応するファイルの種類を指定することができます。プロジェクトが初めて保存される時に、プロジェクトの名前が(.svp という拡張子とともに)決定されます。その後プロジェクトの名前を変更するには、Windows ファイルエクスプローラーといったアプリケーションにより、プロジェクトのファイル名を変更する必要があります。

プロジェクトフォルダー

フォルダーはプロジェクト直下に追加することも、プロジェクト内にあるフォルダー内部へ追加することもできます。追加することのできるフォルダーには3種類あります: (i) プロジェクトフォルダー、(ii) 外部フォルダー(参照と選択により追加されます)、(iii) 外部ウェブフォルダー(URL により追加されます)。これらのフォルダーは、それぞれ(i) **プロジェクトにプロジェクトフォルダーを追加**、(ii) **プロジェクトに外部フォルダーを追加**、(iii) **プロジェクトに外部ウェブフォルダーを追加**コマンドを使用することで、メインプロジェクトフォルダーへ追加することができます。メインプロジェクト以下にあるフォルダーやサブフォルダーへこれらのフォルダーを追加するには、フォルダーやサブフォルダーを右クリックすることで表示されるコンテキストメニューから目的のコマンドを選択してください。

プロパティダイアログでは、ファイルの種類を割り当てることができます。プロパティダイアログを表示するには、フォルダーを右クリックして、コンテキストメニューから**プロパティ**を選択してください。

プロパティダイアログでは、フォルダーの名前と、そのフォルダーに対して使用されるファイルの拡張子を編集することができます(各ファイル拡張子はセミコロンにより区切られます)。プロジェクト サイド バーに表示されているフォルダーを選択し、F2 キーを押下することでも、プロジェクトフォ

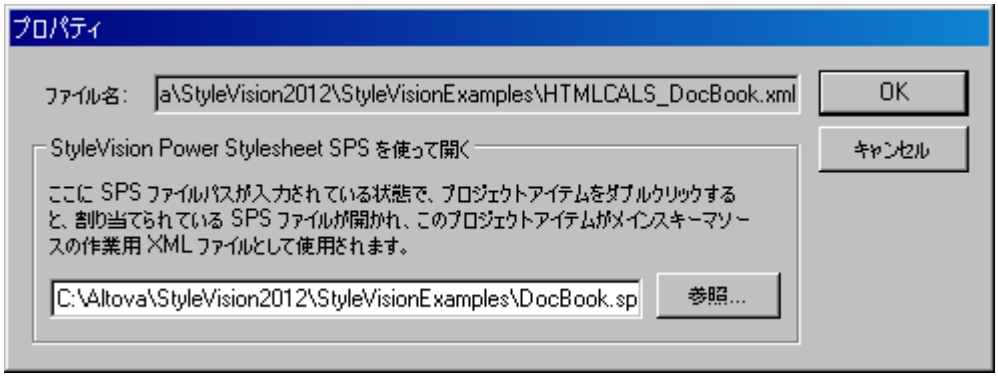
(C) 2015-2021 Altova GmbH

ルダー名を変更することができます。フォルダーに対してファイル拡張子が定義されている場合、該当する拡張子を伴うファイルが「**プロジェクトへファイルを追加**」コマンドにより追加された際に、そのフォルダーへ自動的にファイルが追加されます。複数のフォルダーにて同一のファイル拡張子が定義されている場合、そのファイル拡張子が定義されている(プロジェクト・サイド・バー内の)最初のフォルダーへ、ファイルが追加されます。プロジェクト・サイド・バーでは、ドラッグアンドドロップによりフォルダーの順序を変更することができます。最初のレベル(メインプロジェクトフォルダー直下)にて、フォルダーは(i) プロジェクトフォルダー、(ii) 外部フォルダー、(iii) 外部ウェブフォルダーの順序に固定されます。

プロジェクトファイル

ファイルはメインプロジェクトフォルダーや、サブフォルダーへ追加することができます。プロジェクトメニュー以下にある以下のコマンドを使用することで、ファイルを追加することができます:

- *プロジェクトにファイルを追加* : ファイルを開くウィンドウにて1つ以上のファイルを選択することができます。追加されたファイルは、それぞれ該当する拡張子が定義されている最初のフォルダーへ収められます。
- *プロジェクトにグローバルリソースを追加* : グローバルリソースを介してファイルが追加されます。
- *プロジェクトに URL を追加* : (URL をプロジェクトに追加ダイアログで定義された)URL を介してファイルが追加されます。
- *プロジェクトにアクティブなファイルを追加* : (SPS) 拡張子が定義された最初のプロジェクトフォルダーへ、アクティブな SPS ファイルが追加されます。
- *プロジェクトにアクティブで関係するファイルを追加* : アクティブな SPS ファイルと、スキーマや作業用 XML ファイル、CSS ファイル、静的イメージファイルといった関連するファイルがプロジェクトへ追加されます。これらのファイルは、それぞれ該当するファイル拡張子が定義されたフォルダー内へ収められます。あるデザインと、関連する全てのファイルを素早く取得するのに便利な機能となります。
- *プロパティ* : 任意のプロジェクトファイルが選択されている状態で、プロジェクトウィンドウのツールバーにある**プロパティ**アイコンをクリックするか、コンテキストメニューからプロパティコマンドを選択することで、ファイルの位置が記載されたウィンドウが表示されます(以下のスクリーンショットを参照)。



SPS ファイルが同一プロジェクト内部にある XML ファイルに関連付けられている場合、プロジェクトウィンドウに表示されている XML ファイルをダブルクリックすることで、関連する SPS ファイルが(XML ファイルを作業用 XML ファイルとして)開かれます。同一の SPS に対して複数の XML ファイルが関連付けられている時に、この機能を役立てることができます。この機能により、SPS ファイルと作業用 XML ファイルのブラウジング作業を軽減することができます。

上記のコマンドにより、メインプロジェクトフォルダー内の各フォルダーに定義されたファイル拡張子に従うかたちでファイルが追加されます。特定のフォルダーやサブフォルダーにファイルを追加するには、目的のフォルダーを右クリックして、コンテキストメニューから対応するコマンドを選択して下さい。フォルダー内ではアルファベット順にファイルが並べられます。ファイルをドラッグすることで、別のフォルダーへファイルを移動することができます。ファイルの位置を確認するには、コンテキストメニューから**プロパティ**コマンドを選択して下さい。

グローバルリソース

グローバルリソースをフォルダーへ追加することができます。ファイル型のグローバルリソースは、ファイルリソースに対するエイリアスとなります。1つのエイリアスに対して複数の構成を指定することができ、各構成により異なるファイルリソースを指し示すことが可能です。例えばプロジェクト内部でグローバルリソースが使用される場合、その時点のStyleVision にてアクティブになっている構成により、ターゲットリソースへのリンクが変更されることになります。フォルダー型のグローバルリソースも同様に、その時にアクティブになっている構成に従うかたちで、複数あるターゲット

フォルダーのうちのどれかがリンクされます。フォルダー型のグローバルリソースがデザイン内で使用される場合（例えば作業用 XML ファイルや CSS ファイルを識別するために使用される場合）、フォルダーだけがフォルダー型のグローバルリソースにより指定され、そのフォルダーから目的のファイルまでのパスを別に指定する必要があります。グローバルリソースの使用に関する詳細な情報については、: グローバルリソースの使用を参照ください。

ドラッグ アンド ドロップ

プロジェクト サイドバーでは、フォルダーを他のフォルダーへドラッグし、同一フォルダー上にある別の場所へドラッグすることができます。ファイルを別のフォルダーへドラッグすることはできますが、同一フォルダー内で移動することはできません（ファイルはアルファベット順に並べ替えられます）。その他にも、Windows ファイルエクスプローラーからプロジェクト サイドバーへファイルやフォルダーを直接ドラッグすることができます。

プロジェクトを使用する

プロジェクト内部のファイルは、その種類により様々な方法で使用することができます。プロジェクト サイドバー内にある各種ファイルに対して使用することができるコマンドは、（右クリックにより表示される）コンテキスト メニューから確認することができます。更に、実行可能な場所へファイルをドラッグすることで、関連するコマンドが収められたコンテキスト メニューが表示されます。コマンドによっては、ファイルがドロップされた場所で直接コマンドが実行されます。各種ファイルに対して利用可能なコマンドを以下に示します。

SPS ファイル

- *デザインを開く*: 新たなデザインウィンドウで SPS を開きます。プロジェクトサイドバーからデザインビューへ SPS ファイルをドラッグした際にも、このコマンドを選択することができます。
- *モジュールとしてインポート*: 現在アクティブな SPS に対して SPS をモジュールをインポートします。インポートされたファイルは、デザイン概要のモジュール以下に表示されます（デザイン概要サイドバーのモジュール行へファイルをドラッグすることでも、ファイルをインポートすることができます）。

XML ファイル

- XMLSpy *でファイルを編集*: XML ファイルを XMLSpy で開きます。
- *新規デザインを作成*: 新規 SPS を作成します。SPS に対するスキーマは XML ドキュメントから生成された XML スキーマになります。SPS の作業用 XML ファイルは XML ファイルとなります（デザインビューのメインテンプレートバーへファイルをドラッグすることで、このコマンドを使用することができます）。
- *作業用 XML ファイルとして割り当て*: XML ファイルを SPS の作業用 XML ファイルとして割り当てます（デザイン概要サイドバーの作業用 XML エントリーへファイルをドラッグすることでも、作業用 XML ファイルを追加することができます）。
- *テンプレート XML ファイルとして割り当て*: XML ファイルを SPS のテンプレート XML ファイルとして割り当てます（デザイン概要サイドバーのテンプレート XML エントリーへファイルをドラッグすることでも、テンプレート XML ファイルを追加することができます。）。
- *メモ*: XML ファイルをダブルクリックすると、オプション ダイアログにあるプロジェクトタブの指定に従い、以下の動作のうちのいずれかが実行されます: (i) XMLSpy でファイルを編集、(ii) その XML ファイルをベースにしたデザインを新たに作成、(iii) どちらの動作を行うかユーザーに尋ねる。

XML スキーマ／DTD ファイル

- *XMLSpy でファイルを編集*: XMLSpy にてスキーマファイルを開きます。
- *新規デザインを作成*: 選択されたスキーマをベースにした SPS を新たに作成します（デザインビューのメインテンプレートバーにファイルをドラッグすることでも、選択されたスキーマをスキーマソースとした SPS を新たに作成することができます）。)
- *スキーマファイルとして割り当て*: 選択されたスキーマを、現在アクティブな SPS に対するスキーマソースとして（それまでのスキーマソースを置き換えるかたちで）割り当てます。スキーマを素早く変更するのに便利なコマンドで、例えばスキーマの場所が変更され、間違った割り当てを修正する際に使用することができます（デザイン概要サイドバーのスキーマエントリーへファイルをドラッグすることでも、このコマンドを使用することができます）。
- *メモ* スキーマファイルをダブルクリックすると オプション ダイアログにあるプロジェクトタブの指定に従い、以下の動作のうちのいずれかが実行されます: (i) XMLSpy でファイルを編集、(ii) そのスキーマファイルをベースにしたデザインを新たに作成、(iii) どちらの動作を行うかユーザーに尋ねる。

(C) 2015-2021 Altova GmbH

CSS ファイル

- *XMLSpy でファイルを編集* : XMLSpy で HTML ファイルを開きます。
- *スタイルリポジトリへインポート* : CSS ファイルを、スタイルリポジトリの外部 CSS ファイルへ追加します（スタイルリポジトリサイドバーへファイルドドラッグすることでも、ファイルをスタイルリポジトリへインポートすることができます）。

HTML ファイル

- *XMLSpy でファイルを編集* : XMLSpy で HTML ファイルを開きます。
- *開く* : デフォルトブラウザーで HTML ファイルを開きます。
- *新規デザインを作成* : 新規 SPS を作成し、HTML ドキュメントをベースにしたスキーマを作成することができます（デザインビューへファイルをドラッグすることでも同様の処理が行われます）。

イメージファイル

- *開く* : デフォルトのイメージビューア / 編集アプリケーションでイメージファイルを開きます。
- *デザインにイメージを挿入* : 静的なイメージとして、イメージを SPS 内部に挿入します（ドラッグすることで、目的の場所にイメージを挿入することもできます）。

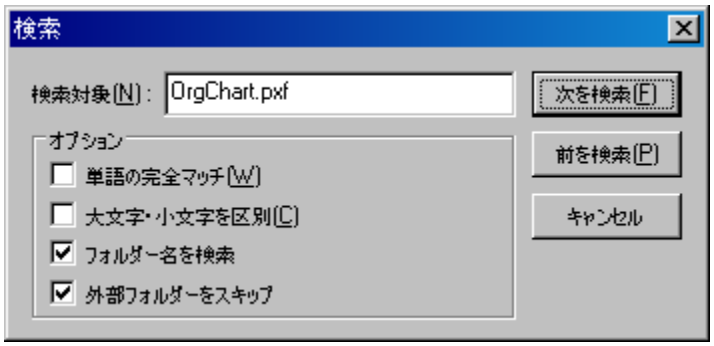
全ての種類

- *含まれているフォルダーを開く* : 選択されたファイルが配置されているフォルダーを、Windows のフォイルエクスプローラーで開きます。
- *切り取り*、*コピー*、*貼り付け*、*削除* : これらのコマンドは、その他の Windows アプリケーションと同じように動作し、選択されたファイルをクリップボードに切り取り/コピーし、クリップボードからファイルを貼り付け、またはファイルを削除することができます。これらのコマンドは複数のファイルが選択された状態でも使用することができます。
- *全て選択* : プロジェクト内のファイルを全て選択します。
- *プロパティ* :ファイルの場所が示されるプロパティダイアログを表示します。

プロジェクトの中を検索

プロジェクトファイルやフォルダーの名前（またはその一部）から検索を行うことができます。検索が成功した場合、ファイルまたはフォルダーが1つずつハイライトされます。

検索を開始するには、プロジェクトサイドバーにてプロジェクトフォルダーを選択し、メニューオプションから「**編集 | 検索**」を選択（または Ctrl+F のショートカットを押下）します。検索ダイアログが表示されるので、検索を行うテキスト文字列を入力し、必要に応じて検索オプションを選択/選択解除します（以下のスクリーンショットを参照）。



以下の検索オプションを使用することができます:

- 「単語の完全マッチ」により、ファイルまたはフォルダー名全体が検索文字列にマッチするようになり、検索の結果を絞り込むことができます。ファイル名の場合、ドットの前にある文字列（但しドットは含まず）も単語として扱われます。
- 大文字と小文字を区別したファイルやフォルダー名の検索を行うことができます。
- フォルダー名を検索の対象に加えることもでき、それ以外の場合はファイルだけが検索されます。
- 外部フォルダー を検索の対象へ加える/外すことができます。外部フォルダーとはシステムまたはネットワーク上に実在するフォルダーのことで、プロジェクト内部に論理的に作成されるプロジェクトフォルダーとは異なるものです。

検索に成功した場合、プロジェクトサイドバーにて最初にマッチしたアイテムがハイライトされます。検索ダイアログに表示されている**次を検索**または**前を検索**ボタンをクリックすることで、マッチする全てのアイテムを確認することができます。

関連項目

- プロジェクトサイドバー
- プロジェクトメニュー

(C) 2015-2021 Altova GmbH

4.9 StyleVision におけるカタログ

StyleVision では、OASIS XML カタログメカニズムのサブセットがサポートされます。カタログ機能により、一般的に使われているスキーマ（やスタイルシートなどのファイル）が、StyleVision によりローカルのユーザーフォルダーから取得されます。これにより全体的な処理スピードが向上し、（ネットワークに接続していない状態でも）ユーザーはオフラインで作業することができるようになり、（URI の変更はカタログファイル内部に限定されるので）ドキュメントのポータビリティーが向上します

StyleVision におけるカタログ機能は以下のように動作します:

RootCatalog.xml

StyleVision が起動すると、検索を行うカタログファイルのリストが含まれる RootCatalog.xml という名前のファイルがロードされます。このファイルを変更し、nextCatalog 要素内にファイルを加えることで、検索を行うファイルを増やすことができます。各カタログファイルが検索され、カタログファイルにて指定されたマッピングにより URI が解決されます。

```
<?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>
```

上記の表示にて、%AltovaCommonFolder% 変数により識別されたフォルダー以下の Schemas ならびに XBRL フォルダー内に、catalog.xml という名前のカタログファイルが存在する点に注目してください（%AltovaCommonFolder% 変数の値は以下のテーブルにて与えられます）。

Altova Common フォルダー内にあるカタログファイルにより、（SVG や W SDL といった）広く使用されるスキーマや XBRL タクソノミの公開ならびにシステム識別子から、ローカルに保存された対応するスキーマのコピーが指し示されます。これらスキーマは、StyleVision のインストール時に Altova Common フォルダーへインストールされます。エラーを起こす可能性があるため、これらファイル内にあるマッピングを複製しないように注意する必要があります。

CoreCatalog.xml、CustomCatalog.xml、とCatalog.xml

上に示される RootCatalog.xml にて、CoreCatalog.xml ならびに CustomCatalog.xml が表示されている点に注目してください:

- CoreCatalog.xml には、Altova Common フォルダー内にあるスキーマを参照するための Altova 特有のマッピングが含まれます。
- CustomCatalog.xml は、独自のマッピングを作成するためのスケルトンファイルです。CustomCatalog.xml には、任意のスキーマに対するマッピングを追加することができますが、Altova Common フォルダー内のカタログファイルで記述されていないものに限られます。OASIS カタログ機能でサポートされている要素を使用するようにしてください（下記参照）。
- Altova Common フォルダーには多くの Catalog.xml ファイルが含まれています。各ファイルは、Altova Common フォルダー内にあるスキーマや XBRL タクソノミのフォルダー内部に収められており、それぞれ公開ならびにシステム識別子により、ローカル上に保存された対応するスキーマのコピーが参照されます。

スキーマカタログファイルとスキーマの場所

RootCatalog.xml ならびにCoreCatalog.xml ファイルはStyleVision アプリケーションフォルダーへインストールされます。CustomCatalog.xml ファイルはMyDocuments/Altova/StyleVision フォルダー以下に配置されます。各 catalog.xml ファイルは対応するスキーマフォルダー内に収められており、これらスキーマフォルダーは、%AltovaCommonFolder%¥Schemas ならびに%AltovaCommonFolder%¥XBRL 以下に配置されます。

シェル環境変数 とAltova 変数

シェル環境変数をnextCatalog 要素内で使用することにより、様々なシステム上の場所を指定することができます(上記の *RootCatalog.xml* を参照)。以下のシェル環境変数がサポートされます:

%AltovaCommonFolder%	C:\Program Files\Altova\Common2021
%DesktopFolder%	現在のユーザーのデスクトップフォルダーへのフルパス
%ProgramMenuFolder%	現在のユーザーのプログラムメニューフォルダーへのフルパス
%StartMenuFolder%	現在のユーザーのスタートメニューフォルダーへのフルパス
%StartUpFolder%	現在のユーザーのスタートアップフォルダーへのフルパス
%TemplateFolder%	現在のユーザーのテンプレートフォルダーへのフルパス
%AdminToolsFolder%	現在のユーザーの管理ツールを保管するファイルシステムフォルダーへのフルパス
%AppDataFolder%	現在のユーザーのアプリケーションデータフォルダーへのフルパス
%CommonAppDataFolder%	全てのユーザーのアプリケーションデータを含むファイルフォルダーへのフルパス
%FavoritesFolder%	現在のユーザーのお気に入りフォルダーへのフルパス
%PersonalFolder%	現在のユーザーの個人用フォルダーへのフルパス
%SendToFolder%	現在のユーザーのSendTo フォルダーへのフルパス
%FontsFolder%	システムフォントフォルダーへのフルパス
%ProgramFilesFolder%	現在のユーザーのProgram Files フォルダーへのフルパス
%CommonFilesFolder%	現在のユーザーのCommon Files フォルダーへのフルパス
%WindowsFolder%	現在のユーザーのW indows フォルダーへのフルパス
%SystemFolder%	現在のユーザーのSystem フォルダーへのフルパス
%LocalAppDataFolder%	ローカル(非ローミング)アプリケーションに対するデータリポジトリとして機能するシステムフォルダーへのフルパス
%MyPicturesFolder%	ピクチャライブラリ(マイピクチャ)フォルダーへのフルパス

(C) 2015-2021 Altova GmbH

カタログの動作方法: DTD

カタログにより、DTD への呼び出しは通常ローカルURI へリダイレクトされます。カタログファイルにより、公開またはシステム識別子から対応するローカルURI へのマッピングが行われます。XML ファイル内のDOCTYPE 宣言が読み込まれた場合、公開またはシステム識別子が、カタログファイルによりローカルリソースへマッピングされます。

広く使用されているスキーマの場合、PUBLIC 識別子が通常既に定義されており、カタログファイル内のURI を適切なローカルコピーへマッピングすることだけが必要になります。XML ドキュメントが解析(パース)されると、その中にあるPUBLIC 識別子が読み取られます。カタログファイル内に目的の識別子がある場合、カタログファイル内の対応するURL がロードされ、その場所からスキーマが読み取られます。例えば、以下にあるSVG ファイルを考えてみましょう:

```
<?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>
```

このドキュメントが読み取られると、そのPUBLIC 識別子に対するカタログが検索されます。カタログファイルには以下のエントリーが含まれます:

```
<catalog>
  ...
    <public publicId="-//W3C//DTD SVG 1.1//EN" uri="schemas/svg/svg11.dtd"/>
  ...
</catalog>
```

この場合、PUBLIC 識別子に対応するマッチが存在するため、SVG DTD へのルックアップがschemas/svg/svg11.dtd URI へリダイレクトされ(このパスはカタログファイルに対して相対的となります)、参照されたローカルファイルがDTD として使用されます。カタログ内のPublic ID に対するマッピングが存在しない場合、XML ドキュメント内のURL (上記の例の場合、http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd)が使用されます。

カタログの動作方法: スキーマ

StyleVision では、カタログによりXML スキーマのリダイレクトを行うことができます。XML インスタンスファイルでは、以下の例にあるように、XML ドキュメントのトップレベルドキュメント要素内にあるxsi:schemaLocation 属性にスキーマへの参照が行われます。例えば、

```
xsi:schemaLocation="http://www.altova.com/schema/orgchart OrgChart.xsd"
```

通常、(上の例にて太字で示されている)属性値のURI には、実際に存在するスキーマの位置が記述されます。カタログによりスキーマが参照された場合、実際に存在するXML スキーマを指し示す必要は無いものの、xsi:schemaLocation 属性の妥当性を満たすために何らかの値を入力する必要があります。属性のURI 部分には、例えばhoge などの値を入力することでも、妥当性を確保することができます。xsi:schemaLocation 属性値の名前空間部分(上の例におけるhttp://www.altova.com/schemas/orgchart)により、カタログ内にスキーマを配置することができます。カタログでは、以下のエントリーにより名前空間部分をベースにしたスキーマの識別が行われます。

```
<uri name="http://www.altova.com/schemas/orgchart" uri="C:\MySchemas\OrgChart.xsd"/>
```

によりサポートされるカタログサブセット

CustomCatalog.xml (又はStyleVision)にて読み取ることができるその他のカタログファイル)にてエントリーを作成する場合、以下にあるOASIS カタログ仕様の要素だけを使用するようにしてください。各要素と、その属性値に関する説明を以下に示します。より詳細な解説に

については、XML カタログの仕様 を参照ください。各要素には xml:base 属性を含めることができ、その要素のベース URI を指定するのに使用することができます。

- `<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"/>`

殆どのスタイルシートに見られるように公開識別子が存在しない場合、system 要素によりシステム識別子を URL へ直接マッピングすることができます。更に uri 要素を使用することで、URI を他の URI へマッピングすることもできます。rewriteURI ならびに rewritsSystem 要素により、それぞれ URI やシステム識別子の書き換えを行うことが可能になります。この機能により、使用されるファイル パスを変更し、ターゲットのフォルダーを変更することが可能になります。これら要素に関するより詳しい情報については、XML カタログの仕様を参照ください。

詳しい説明

カタログに関する詳細は、次を参照してください：XML カタログの仕様。

関連項目

- 生成されたファイル

(C) 2015-2021 Altova GmbH

4.10 StyleVision 内のカタログ

The XML カタログ構造によりローカルフォルダーからファイルを抽出することが可能になり、—変更のためにカタログファイルURI のみが必要になるため、に処理スピード全体が向上し、ドキュメントのポータビリティも向上します。詳細に関しては、カタログの機能のセクションを参照してください。

Altova のXML 製品はDTD とXML スキーマなど共通に使用されるファイルに素早くアクセスしロードするためにカタログメカニズムを使用します。このカタログ構造はカスタム化することが可能で、またユーザーにより拡張可能です。セクション StyleVision 内のカタログ構造 とカタログのカスタム化内で説明されています。セクション Windows システムロケーションのための変数 は共通のシステムのロケーションのためにWindows 変数をリストしています。共通で使用されているフォルダーをロケートするためにこれらの変数はカタログファイル内で使用することができます。

このセクションは次のサブセクションに整理されています:

- カタログの仕組み
- StyleVision 内のカタログ構造
- カタログのカスタム化
- Windows システムロケーションのための変数

これらのカタログの詳細に関してはXML カタログ仕様) を参照してください。

4.10.1 カタログの仕組み

カタログをDTD とXML スキーマをリダイレクトするために使用することができます。両方のケースのメカニズムの後ろにあるコンセプトは同じですが、違いの詳細は以下に説明されている通りです。

DTD

カタログはDTD への呼び出しをローカルのURI にリダイレクトするために使用されます。これはマッピング、カタログファイル、ローカルURI により必要とされるパブリックまたはシステム識別子により達成されます。XML ファイル内のDOCTYPE 宣言が読み取れれると、パブリックまたはシステム識別子が必要とされるローカルリソースをカタログファイルマッピングによりロケートします。

人気のあるスキーマに関してはPUBLIC 識別子は通常定義済みです。このためカタログファイル内のURI のみが正しいローカルコピーに対してPUBLIC 識別子 をマップします。XML ドキュメントが解析されると内部のPUBLIC 識別子が読み取られます。この識別子がカタログファイル内で見つかると、カタログファイル内の対応するURL がルックアップされ、スキーマがこのロケーションから読み取られます。例えば、次のSVG ファイルがStyleVision 内で開かれるとします:

```
<?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>
```

カタログはこのSVG ファイルのPUBLIC 識別子を検索します。カタログファイルは以下のエントリを含んでいます:

```
<catalog>
```

```
  ...
    <public publicId="-//W3C//DTD SVG 1.1//EN" uri="schemas/svg/svg11.dtd"/>
  ...
</catalog>
```

この場合 PUBLIC 識別子のために一致するものがあります。この結果 SVG DTD のためのルックアップが(カタログファイルに相対的な) URL schemas/svg/svg11.dtd にリダイレクトされます。これはSVG ファイルのためにDTD として使用されるローカルファイルです。カタログ内で Public ID のためのマッピングが存在しない場合、XML ドキュメント内のURL が使用されます(上のSVG ファイルのサンプルの場合、これはインターネット URL です: http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd)。

XML スキーマ

StyleVision 内では XML スキーマとカタログを使用することができます。XML インスタンスファイル内で、スキーマへのレファレンスがXML ドキュメントのトップレベルの要素の xsi:schemaLocation 属性内で発生します。例えば、

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

xsi:schemaLocation 属性の値には2 つの部分があります: (上で緑色で表示されている)名前空間のパートと(ハイライトされている) URI パート。名前空間の部分は代替リソースをマップするためにカタログ内で使用されています。例えば、以下のカタログエントリは上のスキーマレファレンスを代替のロケーションのスキーマにリダイレクトします。

```
<uri name="http://www.xmlspy.com/schemas/orgchart" uri="C:\MySchemas\OrgChart.xsd"/>
```

通常 xsi:schemaLocation 属性の値のURI 部分は実際のスキーマのロケーションのパスです。しかしながら、スキーマがカタログを介して参照される場合、URI の部分は実際のXML スキーマをポイントする必要はなく、xsi:schemaLocation 属性の文法の妥当性が保持されるために存在する必要があります。Foo の値は、例えば、属性の値の部分が妥当であるため十分です。

4.10.2 StyleVision 内のカタログ構造

StyleVision が開始されると、(*以下のリスティングで示されるような構造を持った*) RootCatalog.xml というファイルがロードされ、ファイル内部に記述されているカタログファイルが検索されます。このファイルを修正して、検索するカタログファイルを増やすこともできます(各カタログファイルは nextCatalog 要素にて記述されます)。各カタログファイル内に記述されているURI が、ファイル内で指定されたマッピングに従って解決されます。

Listing of RootCatalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
   xmlns:spy="http://www.altova.com/catalog_ext"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd">
  <nextCatalog catalog="%PersonalFolder%/Altova/%AppAndVersionName%/CustomCatalog.xml"/>
  <!-- Include all catalogs under common schemas folder on the first directory level -->
  <nextCatalog spy:recurseFrom="%CommonSchemasFolder%" catalog="catalog.xml"
spy:depth="1"/>
  <nextCatalog spy:recurseFrom="%ApplicationWritableDataFolder%/pkgs/.cache"
catalog="remapping.xml" spy:depth="0"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
</catalog>
```

上のリスティングはカスタムカタログ(named CustomCatalog.xml) と(W 3C XML スキーマとSVG スキーマ)などの一般的に使用されるスキーマをロケートするカタログのセットを参照します

- CustomCatalog.xml は(変数 %PersonalFolder% によりロケートされる) 個人用フォルダーにあります。自身のマッピ

ングを作成するスケルトンファイルです。共通のスキーマフォルダー内のカタログファイルにより指されていない必要とするスキーマのためにマッピングを **CustomCatalog.xml** に追加することができます。OASIS カタログメカニズムのサポートされる要素を使用してこれを行います（*次のセクションを参照*）。

- （共通に使用されるスキーマのセットを含む変数 **%CommonSchemasFolder%** を使用してロケートされる）共通のスキーマフォルダーこれらのスキーマフォルダの内部は保存された対応するスキーマのコピーをポイントする URI に対するパブリックおよびまたはシステム識別子をマップする **catalog.xml** ファイルです。
- **CoreCatalog.xml** は StyleVision アプリケーションフォルダー内にあり、XML ドキュメントの Altova の Authentic ビューを生成するために使用される StyleVision パワースタイルシートなど StyleVision 固有のプロセスにより使用されるスキーマとスタイルシートをロケートするために使用されます。

ロケーション変数
RootCatalog.xml 内で使用されている変数は以下の値を持っています（*上のリスティング*）:

%PersonalFolder%	現在のユーザーの個人的なフォルダー、例えば C:\Users\<name>\Documents
%CommonSchemasFolder%	C:\ProgramData\Altova\Common2021\Schemas
%ApplicationWritableDataFolder%	C:\ProgramData\Altova

カタログファイルとスキーマのロケーション
多種のカタログファイルの場所に注意してください。

- **RootCatalog.xml** と **CoreCatalog.xml** は StyleVision アプリケーションフォルダー内にあります。
- **CustomCatalog.xml** は **MyDocuments\Altova\StyleVision** フォルダー内にあります。
- **Catalog.xml** 入るはそれぞれ特定のスキーマフォルダー内にあり、これらのスキーマフォルダーは共通のスキーマフォルダー内にあります。

4.10.3 カタログのカスタム化

CustomCatalog.xml（または StyleVision により読み取られる他のカタログファイル）内でエントリを作成する場合OASIS カタログ仕様の以下の要素のみを使用してください。下の各要素は属性の値の説明と共にリストされています。詳細に関しては（XML カタログ仕様）を参照してください。各要素はその要素のベース URI を指定するために使用される **xml:base** 属性を取ることができます。

- `<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"/>`

以下の点に注意してください：

- 多くのスタイルシート同様パブリック識別子が存在しない場合、システム識別子を **system** 要素を介して直接 URL にマップすることができます
- uri 要素を使用して URI を他の URI にマップすることができます。
- rewriteURI と rewriteSystem 要素により URI またはシステム識別子の開始の部分をそれぞれ書き換えることが可能になります。これによりファイルパスの開始部分が置き換えられこの結果他のディレクトリをターゲットにすることができます。これらの要素の詳細に関しては XML カタログ仕様 を参照してください。

リリース2014 以降StyleVision はXML Catalogs specification (OASIS Standard V1.1, 7 October 2005) 仕様を厳密に順守しています。この仕様は厳密にパブリックID または システムID）外部識別子ルックアップを（パブリックID または システムID を持たないURI ）URI ルックアップと区別しています。名前空間 URI は単純にパブリックID または システムID ではなくURI と考えられ外部識別子ルックアップではなくURI ルックアップとして使用されるべきです。バージョン2014 以前のStyleVision バージョンではスキーマ名前空間 URI は**<public>** マッピングを使用して訳されてきました。2014 以降から**<uri>** マッピングが使用されています。

v2014 以前： `<public publicID="http://www.MyMapping.com/ref"`
`uri="file:///C:/MyDocs/Catalog/test.xsd"/>`
V-2014 以降: `<uri name="http://www.MyMapping.com/ref"`
`uri="file:///C:/MyDocs/Catalog/test.xsd"/>`

StyleVision が参照されたスキーマを検索する方法

xsi:scemaLocation 属性を介してXML ドキュメント内で参照されるスキーマ（*以下に表示されています*）。
xsi:schemaLocation 属性の値には2 つの部分があります:（緑色で表示されている）名前空間のパートと（ハイライトされている）URI パート。

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

以下はStyleVision が参照されたスキーマを検索するステップです。スキーマは最初のステップでロードされます。

1. **xsi:schemaLocation** 値のURI 部分のためのカタログをルックアップします。**rewriteURI** マッピング内を含みマッピングが検索されるとスキーマのロードのために結果のURI を使用します。
2. **xsi:schemaLocation** 値の名前空間部分のためのカタログをルックアップします。**rewriteURI** マッピング内を含みマッピングが検索されるとスキーマのロードのために結果のURI を使用します。
3. スキーマのロードのために**xsi:schemaLocation** 値のURI 部分を使用します。

XML スキーマ仕様

XML スキーマ仕様情報はStyleVision にビルトされておりXML スキーマ（`.xsd`）ドキュメントの妥当性はこの内部情報に対してチェックされます。XML スキーマドキュメント内では、このためレファレンスはXML スキーマ仕様を定義するスキーマになってはなりません。

`%AltovaCommonSchemasFolder%\Schemas\schema` フォルダー内の**catalog.xml** ファイルは古いXML スキーマ仕様を実装するDTD へのレファレンスを含んできます。XML スキーマドキュメントをこれらのスキーマに対して検証してはなりません。参照されるファイルはドキュメントをこれらの古い奨励に従い作成する場合に編集の目的でStyleVision に入力ヘルパー情報を与えるために含まれています。

4.10.4 Windows システムロケーションのための変数

`nextCatalog` 要素ではシェル環境変数が使用され、システムに関係する場所へのパスが参照されます（*上のRootCatalog.xml の内容を参照*）。以下のシェル環境変数がサポートされます:

`%PersonalFolder%`	現在のユーザーの個人的なフォルダーへのフルパス、例えば`C:\Users\<name>\Documents`
`%CommonSchemasFolder%`	`C:\ProgramData\Altova\Common2021\Schemas`
`%ApplicationWritableDataFolder%`	`C:\ProgramData\Altova`

%AltovaCommonFolder%	C:\Program Files\Altova\Common2021
%DesktopFolder%	現在のユーザーのデスクトップフォルダーへのフルパス。
%ProgramMenuFolder%	現在のユーザーのプログラムメニューフォルダーへのフルパス。
%StartMenuFolder%	現在のユーザーの開始メニューフォルダーへのフルパス。
%StartUpFolder%	現在のユーザーのスタートフォルダーへのフルパス。
%TemplateFolder%	現在のユーザーのテンプレートフォルダーへのフルパス。
%AdminToolsFolder%	現在のユーザーの管理ツールを保管するファイルシステムディレクトリへのフルパス。
%AppDataFolder%	現在のユーザーのアプリケーションデータフォルダーへのフルパス。
%CommonAppDataFolder%	全てのユーザーのアプリケーションデータを含むファイルディレクトリへのフルパス。
%FavoritesFolder%	現在のユーザーの[お気に入り]フォルダへフルパス。
%PersonalFolder%	現在のユーザーの個人フォルダーへのフルパス。
%SendToFolder%	現在のユーザーの送信フォルダーへのフルパス。
%FontsFolder%	システムフォントフォルダーへのフルパス。
%ProgramFilesFolder%	現在のユーザーのプログラムファイルフォルダーへのフルパス。
%CommonFilesFolder%	現在のユーザーの共通ファイルフォルダーへのフルパス。
%WindowsFolder%	現在のユーザーのWindows フォルダーへのフルパス。
%SystemFolder%	現在のユーザーのシステムフォルダーへのフルパス。
%LocalAppDataFolder%	ローカル(非ローミング) アプリケーションのためのデータレポジトリとしての役割を果たすファイルシステムディレクトリへのフルパス。
%MyPicturesFolder%	ピクチャフォルダーへのフルパス。

5 SPS コンテンツ

このセクションでは、XML データコンテンツのデザインを構成する SPS ドキュメントコンポーネントの作成ならびに編集を行う方法について説明します。様々な操作方法が以下のとおり記述されており、それぞれのセクション以下にあるサブセクションにて詳細が記述されます。これらメカニズムはメイン、グローバル、または名前付きテンプレートをデザインするために使用されます。

- テキストとして XML コンテンツを挿入: 関連するノード（要素、属性、型、または CDATA）をドラッグすることで、デザイン内に XML データを挿入し、（コンテンツ）または（その他のコンテンツ）として作成します。
- MS Word コンテンツの挿入
- ユーザー定義テンプレート
- ユーザー定義要素, XML テキストブロック
- テーブルとの作業: テーブルは（i）SPS デザイナーにより、（静的テーブル）や XML ドキュメントのサブ構造を使用することで SPS デザイン内に、または（ii）Authentic View ユーザーにより挿入することができます。
- リストの作成: SPS デザイン内にリスト構造が入力される静的リストや、XML ドキュメントのサブ構造がリストとして作成される動的リストにより、パワフルなデータ並び替え機能を使用することができます。
- イメージの使用: 様々な方法を使用することで、ターゲット URI（静的、動的、これらの混合、パースされていないエンティティ URI）から画像を SPS デザインへ挿入することができます。
- データ入力デバイス（または フォームコントロール）の使用: 入力フィールドやコンボボックスといったデータ入力デバイスにより、Authentic View にて XML データを入力することができます。この機能により、ユーザーへのヘルプや入力の制約が提供されます。XML ドキュメント内にある個々のノードをデータ入力デバイスとして作成することができます。
- リンク
- バーコード
- レイアウト モジュール
- 変更機能: この機能を使用することで、テンプレートにマッチしたノードの選択を行い、別のコンテンツ型へノードを変更することが可能になります。

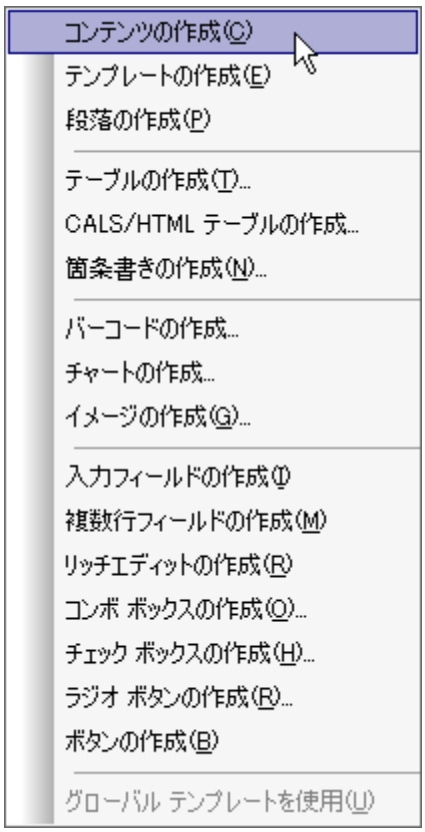
関連項目

- SPS ファイルの高度な機能
- SPS ファイル追加機能

(C) 2015-2021 Altova GmbH

5.1 テキストとして XML コンテンツを挿入

対応するスキーマノードをスキーマツリーウィンドウからドラッグし、デザインにてドロップすることで、XML ドキュメント内にあるノードから得られたデータを、デザインに挿入することができます。スキーマノードをデザインにドロップすると、デザインにてこのノードをどのように作成するのか選択するためのメニューが表示されます（以下のスクリーンショットを参照）。



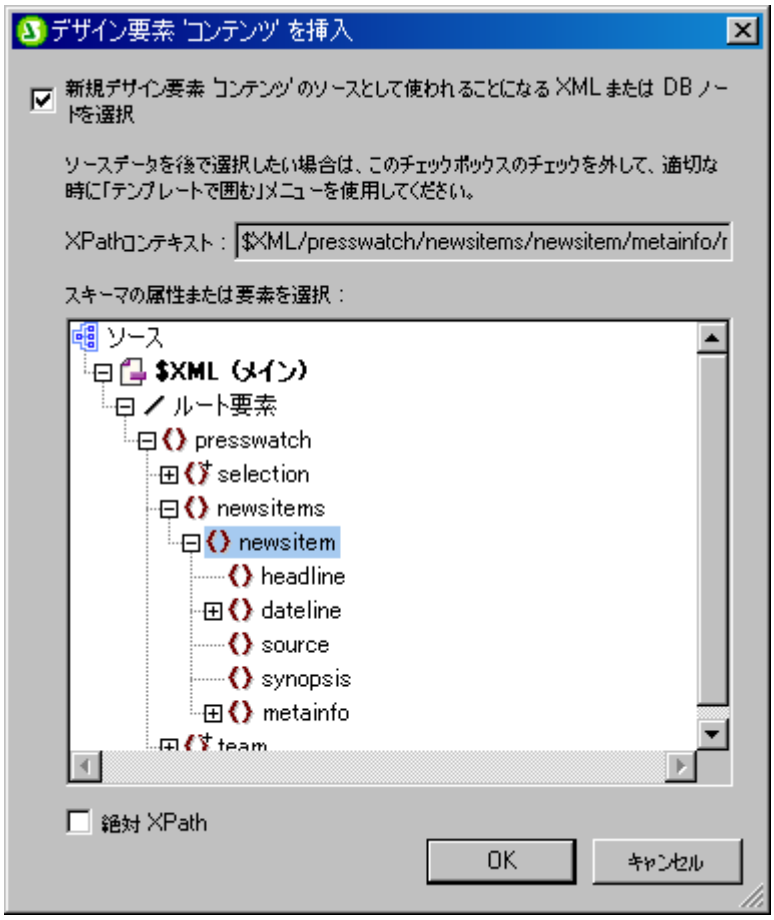
スキーマノードの種類

スキーマツリーサイドバーからデザインへドロップすることができるスキーマノードは、(i) 要素ノード、(ii) 属性ノード、(iii) データ型ノードのいずれかになります。

コンテンツの挿入ツールバーアイコンを使用する

デザイン要素の挿入 ツールバー アイコンを使用することでも、デザイン内にあるノードのコンテンツを挿入することができます。以下の方法によりコンテンツを挿入することができます:

1. コンテンツの挿入アイコンを選択します。
2. コンテンツを挿入するデザイン内の箇所をクリックします。デザイン要素コンテンツを挿入ダイアログが表示されます（*下のスクリーンショット*）。



3. デザイン内における挿入箇所のコンテキストが XPath コンテキストフィールドにて表示されます。コンテンツを作成するノードを選択します。
4. 「OK」をクリックします。コンテンツプレースホルダが作成されます。選択されたノードがコンテキストノードでは無い場合、コンテンツプレースホルダの周りに、選択されたノードへのパスまでのテンプレートタグが作成されます。

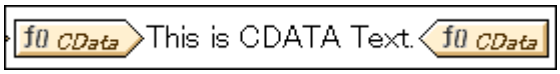
ノードのテキストコンテンツを出力する

ノードのテキストコンテンツを出力するには、ノードをコンテンツとして作成します。ノードがコンテンツとして作成されると、デザインドキュメントにてノードが以下のように表示されます:

上のスクリーンショットでは、Desc 要素がコンテンツとして作成されました。出力ではDesc のテキストコンテンツが表示されます。Desc 要素にBold やItalic といった子要素が含まれる場合、子要素のテキストコンテンツも Desc の一部として出力されます。Desc 要素の属性ノードは子ノードとして見なされないため、属性ノードのコンテンツがDesc コンテンツの一部として出力されることはありません。属性内のコンテンツを処理するには、属性ノードを明示的に挿入する必要があります。

CDATA セクション

CDATA セクションがXML ドキュメント内に存在する場合、それらが出力になります。Authentic View 内では、(メニューコマンド Authentic | マークアップ を使用して)マークアップが切り替えるとタグで表示されます。Authentic View にて(コンテキストメニューから)ドキュメントを編集することで、CDATA セクションをXML ドキュメント内に挿入することができます。



メモ　Authentic View では、CDATA セクションを入力フィールド（テキストボックスならびに複数行のテキストボックス）へ挿入することはできません。Authentic View 内に表示される要素内部にテキストコンテンツコンポーネントとして挿入することはできます。

このセクション

このセクションのサブセクションでは、XML コンテンツをテキストとして挿入するときに考慮するべき、その他の事柄について説明します:

- ノードが挿入された際に、ノードのテキストコンテンツを 定義済みのフォーマットにより直接マークアップ する。
- ソーススキーマの構造により、Authentic View の使用による影響がどのように変化するのか。
- ノード内に明示的に含まれていない子孫ノードを処理に含める 次を参照してください:その他のコンテンツ.

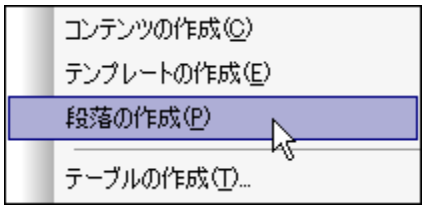
メモ　ノードのコンテンツプレースホルダを削除することで、**空のテンプレートルール**を作成することができます。空のテンプレートルールを使用することで、処理されない（つまり表示させない）ノードを定義することができます。

関連項目

- デザインビューシンボル
- テンプレートとデザインフラグメント

5.1.1　定義済みのフォーマットとともにコンテンツを挿入する

ノードのテキストコンテンツは、StyleVision の定義済みのマークアップとともに、直接挿入することができます。スキーマツリーウィンドウから目的の場所にノードをドラッグアンドドロップして、表示されるメニューから**段落の作成**を選択します（以下のスクリーンショットを参照）。



定義済みのフォーマットタグを選択し、その他の定義済みフォーマットをツールバー内のコンボボックスのフォーマット から選択、またはメニューコマンドの「**挿入 | フォーマット**」を使用することで、定義済みのフォーマットを変更することができます。

プロパティウィンドウ内にある**段落**グループの**段落タイプ**プロパティの値を変更、またはノードテンプレートのコンテキストメニューコマンド「タグで囲む | 特殊な段落」から段落タイプを変更することで、定義済みのフォーマットを変更することができます。

各種の段落には、それぞれ書式に関する高度な機能が含まれています。pre フォーマット型では、改行が空白スペースに変換されること無く、そのまま表示されます。

関連項目

- デザインビューシンボル
- 定義済みのフォーマット

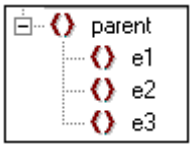
5.1.2 Authentic View にて要素を追加する

デザインにて要素を作成する場合、要素がどのように作成されたかにより、Authentic View にてタブキーが押下された時や**追加...** をクリックするといった、ユーザーのアクションに対するレスポンスが決定されます。ユーザーの手で要素が追加された時に、どの要素がAuthentic View にて作成されるかがここでは重要になります。例えば、(要素サイドバーにある**要素の追加**アイコンをクリックすることで) ユーザーにより要素が挿入された時、どの子要素が自動的に作成されるのかを指定することになります。

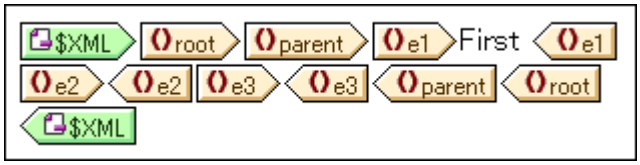
Authentic View は、そこで指定されているスキーマの構造に従うという点に注意する必要があります。Authentic View にてスキーマの構造が正確に実装されることを保証するため、デザインを行う際には、以下に示されるデザインのルールについて留意しておく必要があります。

明確なコンテンツモデル

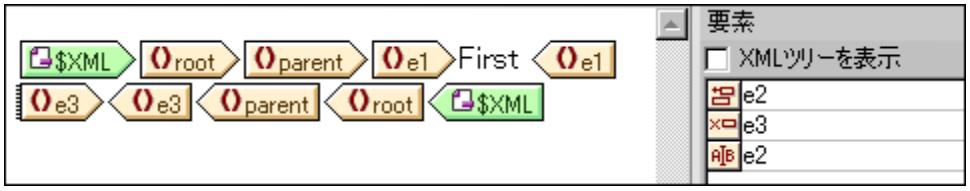
コンテンツモデルを曖昧で無いものにするには、(選択やグループ、代替を持たない) 子要素のシーケンス数を(maxOccurs=1 により) 1に抑えたものにする必要があります。この場合、要素が追加される際に、その子要素のシーケンスも一意に確定するので、自動的に追加されることになります。以下のスクリーンショットでは、3つある子要素は全て必須となっており、一度だけ出現することが許されています。



parent 要素がAuthentic View に挿入されると、その子要素も自動的に挿入されます(以下のスクリーンショットを参照)。タブキーを押下することで、シーケンス内の次要素に移動することができます。



e2 要素が必須では無い場合、parent 要素がAuthentic View に挿入されると、e1 ならびに e3 が自動的に挿入され、要素 e2 ならびに e3 が自動的に挿入されます。e2 が要素サイドバーに表示され、必要な場合は挿入することもできます(以下のスクリーンショットを参照)。e1 内にてタブキーを押下することで e3 へ移動します。



上記のような状況だけが、曖昧性が無いとAuthentic View により判断されます。それ以外の場合は全て、Authentic View には曖昧だと判断され、目的の要素が表示され、選択を促されます。結果としてデザインは、以下に記されるように、単純なルールに従うことになります。

(C) 2015-2021 Altova GmbH

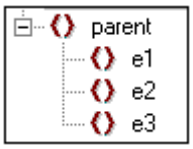
曖昧なコンテンツモデル

XML ドキュメントの編集を行いつつ、Authentic View にて要素を効率的に表示するため、SPS は以下のルールに従う必要があります。

- 子要素の順序はデザインに配置された通りに表示される。
- Authentic View における兄弟要素間の曖昧さを回避するため、目的の順序で必要な全ての子要素をデザインドキュメント内、かつ**単一の親ノード内**に配置する必要があります。兄弟間のリレーションシップを Authentic View でも保持するには、ある親要素内にある各子要素を、複数のインスタンス内に配置しないようにしてください。

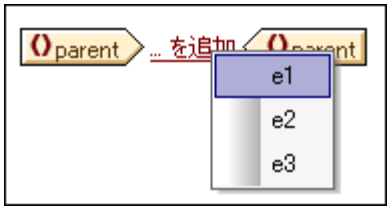
これら2つのルールを以下の例で説明します。

必須子要素のシーケンスから構成される parent 要素のコンテンツモデルを考えます。このコンテンツモデルは上記の曖昧さのないコンテンツモデルと、シーケンスが必須ではないという点を除いて同じものです。この違いのためにシーケンスが出現するかは決定されず、コンテンツモデルは曖昧なものとして判断されます。以下のスクリーンショットに示されるようにドキュメントをデザインした場合、Authentic View にて曖昧さが発生することになります。



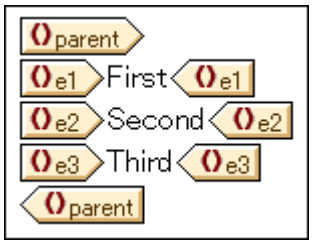
parent 要素は Authentic View にて（シーケンスが必須では無いため）以下のように表示されます。

... を追加ボタンをクリックすると、子要素を含むメニューが表示されます。



これら要素のどれかを選択することで、その要素が挿入されますが、Authentic View によりシーケンスの曖昧さを解決することができないため、残り2つの要素が挿入されることはありません。

コンテンツモデルのデザインを（上のルールに従いつつ）正しく行うには、目的のノードを単一ノード内に正しい順序で明示的に作成します。デザインドキュメントを以下のように変更してください：

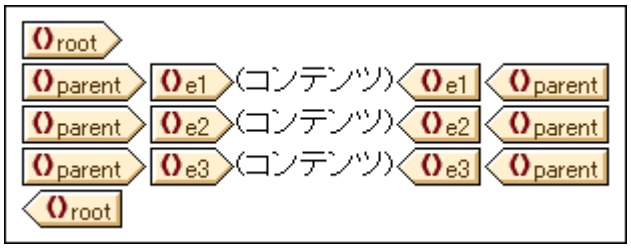


3つある子要素が全て単一の親ノード内部に配置されていることに注目してください。上のデザインは、Authentic View にて以下のように表示されます：

Authentic View ユーザーが「...を追加」をクリックすることで、対応する要素とそのコンテンツが挿入されます。

メモ

- ある要素が複数回出現することが許されている場合、Authentic View にて表示されている出現回数が、スキーマにて許可されている最大出現回数（maxOccurs）に達するまで、サイドバーに許可されている要素が表示され続けます。
- 異なる親ノード内にある各子要素が作成された場合、親子のリレーションシップは他の要素からは独立したものとなるだけでなく、各子要素の位置を特定するため親ノードを行き来する必要が生じるため、処理時間も多くなります。



関連項目

- テキストとして XML コンテンツを挿入
- Authentic View 使用方法

5.1.3 その他のコンテンツ

`rest-of-contents`（その他の）コンテンツプレースホルダは、作成されたテンプレートに対する要素以下にある、その他の子要素全てに対して適用されます。以下の例を考えてみましょう：

- parent 要素には child1 から child4 まで4つの子要素が含まれます。
- parent 要素に対するテンプレート内では、ある処理が child1 と child4 子要素に対して明示的に定義されています。

この場合、child1 と child4 だけが処理されます。child2 ならびに child3 子要素が処理されることはありません。parent の要素に対してその他のコンテンツプレースホルダが挿入された場合、テンプレート内部で明示的に定義された処理ルールにより、child1 ならびに child4 だけではなく、child2 ならびに child3 子要素も処理の対象になります。これら子要素に対してグローバルテンプレートが定義されている場合、グローバルテンプレートが使用されます。そうでない場合、（要素、属性、テキストノードに対して使用される）内蔵のデフォルトテンプレートが適用されます。

重要なメモ その他のコンテンツに対してどのノードが選択されているか留意する必要があります。

(C) 2015-2021 Altova GmbH

- 上の例で記述されたように、全ての子要素ノードならびに子テキストノードは、その他のコンテンツプレースホルダにより選択されます（XML ドキュメント内にある不正な子ノードも処理されます）。
- 属性ノードは選択されません。これらは子ノードとして見なされず、XPath 軸にも含まれません。
- 子要素のグローバルテンプレートが親テンプレートで使用される場合、子要素がローカルで使用されたとは見なされません。そのため、その他のコンテンツプレースホルダにより子要素も選択されることになります。しかし、子要素のグローバルテンプレートが「ローカルのコピー」となっている場合、ローカルで使用されたことになり、子要素がその他のコンテンツにより選択されることはありません。

メモ　**空のテンプレートルール**を作成し、そのノードの（コンテンツ）プレースホルダを削除することができます。空のテンプレートルールにより、処理（出力）を行わないノードを定義することができます。

関連項目

- テキストとして XML コンテンツを挿入
- デザインビューシンボル
- テンプレートとデザインフラグメント

5.2 MS Word コンテンツの挿入

Microsoft Word 2007+ がマシンにインストールされていると、コンテンツをWord ドキュメントから、デザインに**静的コンテンツ**として貼りつけることができます。Word コンテンツは、適切に対応するデザインコンポーネント内挿入され、テキスト書式プロパティは、Word コンテンツから移動されます。ます。例えば、Word 段落ブロック内のテキストコンテンツは、パラグラフのコンポーネント内に挿入され、テキストの書式は保管されます（*下のスクリーンショット参照*）。

Accelerate XML Development

Today, eXtensible Markup Language (XML) technologies play a critical role in all software development projects. XML has received widespread support and adoption in the computer industry because of its simplicity, extensibility, interoperability, and flexibility, all of which stem from its power to represent data independent of programming language, platform, or operating system. In XML-based applications, XML is used alongside complementary technologies such as **XML Schema**, **XSLT**, **XQuery**, **Web services**, and others. Today's developer needs a tool for creating, editing, and debugging these XML-related technologies in an efficient, standards-based manner.

Altova XMLSpy 2007 delivers all the power you need to create the most advanced XML applications, yet at the same time it's flexible enough to allow you to work with XML using the views and options that best suit your specific requirements and preferences. XMLSpy 2007 increases productivity by allowing you to develop higher quality, standards-conformant XML-based applications more quickly than ever before.

Word コンテンツ

§p A Accelerate XML Development A §p

§p §p

§p Today, eXtensible Markup Language (XML) technologies play a critical role in all software development projects. XML has received widespread support and adoption in the computer industry because of its simplicity, extensibility, interoperability, and flexibility, all of which stem from its power to represent data independent of programming language, platform, or operating system. In XML-based applications, XML is used alongside complementary technologies such as **XML Schema**, **XSLT**, **XQuery**, **Web services**, and others. Today's developer needs a tool for creating, editing, and debugging these XML-related technologies in an efficient, standards-based manner. §p

§p §p

§p ***Altova XMLSpy 2007*** delivers all the power you need to create the most advanced XML applications, yet at the same time it's flexible enough to allow you to work with XML using the views and options that best suit your specific requirements and preferences. XMLSpy 2007 increases productivity by allowing you to develop higher quality, standards-conformant XML-based applications more quickly than ever before. §p

§p §p

Word コンテンツは、デザイン内に貼り付けられています。適切な段落フォーマットが適用され、書式は保管されます。

メモ Word コンテンツに加え、Word ドキュメントとして貼り付けることのできるコンテンツは、StyleVision デザイン内に貼り付けることができます。これは、MS Excel テーブルとHTML ページコンテンツを含みます。

(C) 2015-2021 Altova GmbH

メモ　W ord ドキュメント全体から静的コンテンツを含むSPS を作成するには、ファイル | 新規作成 | W ord 2007+ から新規作成 コマンドを使用して新規 SPS を作成します。

サポートされる W ord 機能

次のW ord 構造と書式は、W ord コンテンツがデザインにコピーして貼り付けられた場合サポートされます

- フォーマットされたテキスト
  - *異なるフォント、サイズ、太線、スタイル、テキストの装飾など*
  - *色*
  - *背景色*
  - *テキストの周りの枠*
- 段落
- 改ページ
- 水平線
- ハイパーリンク
- ブックマーク
- テーブル
  - *行の範囲、列の範囲*
  - *フォーマット済みリッチコンテンツ*
  - *ネストされたテーブル*
  - *ヘッダー、フッター*
- リスト、サブリスト
  - *箇条書き: 異なるスタイル*
  - *列挙: 異なるスタイル*
- イメージ

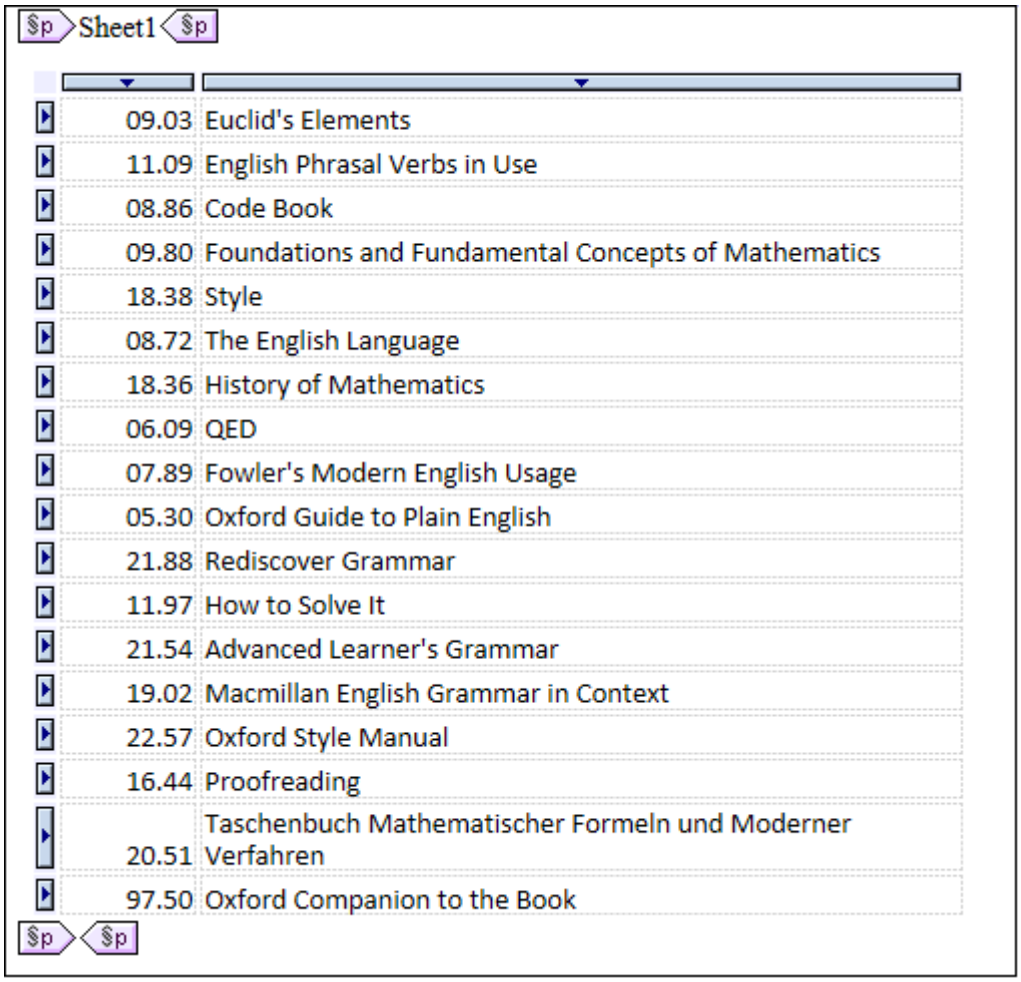
5.3 MS Excel コンテンツの挿入

Microsoft Excel 2007+ がマシンにインストールされている場合、コンテンツを Excel ドキュメントからデザインに**静的コンテンツ**として貼り付けることができます。Excel コンテンツは、静的テーブルおよび他の適切に対応するデザインコンポーネントとして挿入されます。書式 プロパティは保管されます（*下のスクリーンショット参照*）。各 Excel シートは、個別の静的テーブルとして挿入されます。

	A	B
1	09.03	Euclid's Elements
2	11.09	English Phrasal Verbs in Use
3	08.86	Code Book
4	09.80	Foundations and Fundamental Concepts of Mathematics
5	18.38	Style
6	08.72	The English Language
7	18.36	History of Mathematics
8	06.09	QED
9	07.89	Fowler's Modern English Usage
10	05.30	Oxford Guide to Plain English
11	21.88	Rediscover Grammar
12	11.97	How to Solve It
13	21.54	Advanced Learner's Grammar
14	19.02	Macmillan English Grammar in Context
15	22.57	Oxford Style Manual
16	16.44	Proofreading
17	20.51	Taschenbuch Mathematischer Formeln und Moderner Verfahren
18	97.50	Oxford Companion to the Book

Excel シート

(C) 2015-2021 Altova GmbH



テキストの書式が保持された静的テーブルとしてデザインにインポートされた Excel コンテンツ

メモ　Excel コンテンツに加え、Excel ドキュメントとして貼り付けることのできるコンテンツは、StyleVision デザイン内に貼り付けることができます。これは、MS Word コンテンツとHTML ページコンテンツを含みます。

メモ　Excel ドキュメント全体から静的コンテンツを含むSPS を作成するには、 ファイル | 新規作成 | Excel 2007+ から新規作成 コマンドを使用して新規 SPS を作成します。

サポートされる Word 機能

次のWord 構造と書式は、Word コンテンツがデザインにコピーして貼り付けられた場合サポートされます

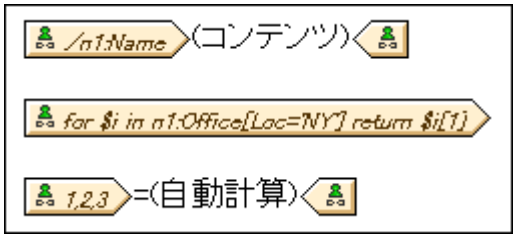
- テーブル
  - *列の範囲、行の範囲*
  - *フォーマット済みパッチコンテンツ*
  - *ネストされたテーブル*
  - *ヘッダー、フッター*
- フォーマットされたテキスト
  - *異なるフォント、サイズ、太線、スタイル、テキストの装飾など*
  - *色*
  - *背景色*
  - *テキストの周りの枠*

- 段落
- *ハイパーリンク*
- リスト、サブリスト
  - *箇条書き: 異なるスタイル*
  - *列挙: 異なるスタイル*
- *イメージ*

(C) 2015-2021 Altova GmbH

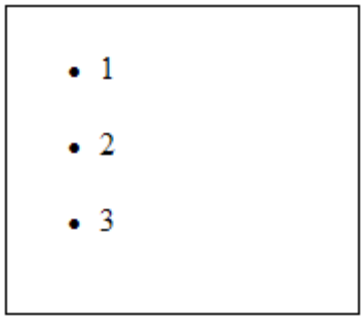
5.4 ユーザー定義テンプレート

ユーザー定義テンプレートとは、XPath 条件式により生成されるアイテムに対して適用されるテンプレートのことです。これらアイテムは原子型の値またはノードとなります。以下のスクリーンショットでは3つのユーザー定義テンプレートが示されており、タグの左側にユーザー定義テンプレートアイコンが示されます。ユーザー定義テンプレートが持つ柔軟性により、テンプレートを作成することができます。ユーザー定義テンプレートにより生成されたコンテンツは Authentic ビューで編集することはできない点にご注意してください。



上に示されるこれら3つのユーザー定義テンプレートに対して定義された XPath 条件式により、以下の処理が行われます:

- ソーススキーマ内のノードを選択します。XPath 条件式を使用することで、全てのコンテキストノードから、任意のスキーマソース内にある任意のノードへアクセスすることができるようになります。曖昧性が無いかたちで StyleVision が定義されたノードを識別することができる場合、テンプレートはユーザー定義テンプレートから(Authentic View における編集が可能な) 通常のテンプレートへ自動的に変換されます。ユーザー定義テンプレートの場合、テンプレートタグの左側にユーザー定義テンプレートを表す緑色のアイコンが表示されます。
- XPath 2.0 と XPath 3.0 の for コンストラクトにより指定された条件を満たすノードを選択します。このようなテンプレートは StyleVision に通常のテンプレート(ユーザー定義テンプレート) を明確に解決することを許可しません。
- 原子型の値のシーケンス(1、2、3)を選択します。原子型の値に対してテンプレートを作成することは可能ですが、このようなテンプレート内でコンテンツプレースホルダを使用することはできません。これは、(コンテンツプレースホルダにより生成される) xsl:apply-template 処理命令が、(原子型の値ではなく) ノードアイテムにしか適用できないためです。そのかわり、箇条書きといったデザイン要素とともに、自動計算を使用することは可能で、以下のスクリーンショットの左側にあるユーザー定義関数により、右側のような出力が生成されます。



メモ　SPS にて XSLT 1.0 が使用されている場合、XPath 条件式の返り値をノードセットにする必要があり、そうでない場合エラーとなります。

XPath を使ったテンプレートノードの選択を行う利点

スキーマノードを XPath 条件式(ユーザー定義テンプレート)により選択することで、XPath のパスセレクターメカニズムを使ったアイテムノードやシーケンスの選択を行うことができるようになり、ノードの選択に対してフィルタリングや条件をセットすることが可能になります。そのため、特定の XML ドキュメントノードを、任意のテンプレートに対して使用することができるようになります。例えば、//Office/Department[@Location="NY"] という XPath 条件式により、NY という値を持った Location 属性が含まれる Department

ノードだけが選択されます。このセクション内にあるその他の例も参照ください。

メモ XPath 条件式に複数のロケーション パスやロケーションステップが含まれる場合、複数のロケーション パスやステップ内に括弧が使用されているかがグループ化や並び替えを行う際に重要になります。例えば、/Org/Office/Dept という XPath 条件式は、(/Org/Office/Dept) という条件式とは異なった方法で処理されます。括弧を伴わない条件式では、各ロケーションステップが繰り返されるのに対して、括弧付きの条件式の場合、全ての Office 要素にある全 Dept 要素が、1 つのノードセットにより返されます。

括弧	実際の XSLT メカニズム	効果

無し	`<xsl:for-each select="Org">`
	`  <xsl:for-each select="Office">`
	`  <xsl:for-each select="Dept">`
	`  ...`
	`  </xsl:for-each>`
	`  </xsl:for-each>`
	`</xsl:for-each>`
		各 Office 要素には独自の Dept が含まれ、グループ化や並び替えは、各 Office 内部で行われる。

有り	`<xsl:for-each`
	`select="/Org/Office/Dept">`
	`  ...`
	`</xsl:for-each>`
		Dept 要素が、全ての Office ならびに Org 要素に対して展開される。

グループ化や並べ替えを行う際に XPath 条件式の評価方法が大きな違いをもたらします。

ユーザー定義テンプレートを挿入する

以下の操作によりユーザー定義テンプレートを挿入することができます:

1. デザイン要素の挿入ツールバーにある**ユーザー定義テンプレートを挿入**アイコンをクリックし、デザイン内のテンプレートを挿入する場所をクリックします。または、テンプレートを挿入する場所を右クリックし、コンテキスト メニューから**ユーザー定義テンプレートを挿入**コマンドを選択します。
2. In the XPath 式を編集 ダイアログにて XPath 条件式を入力し、**OK** をクリックします。XPath 条件式のコンテキスト ノードは、クリックされたノードになる点に注意してください。空のノードテンプレートが作成され、場合によっては結合されたノードが作成されます。ノードが結合されると、ターゲットとなるインスタンスノードが同一のレベルにて選択されるのに対し、ノードが結合されていない場合（つまり複数の階層レベルにて分割されている場合）、ループにより各インスタンスノードが選択されます。グループ化や並べ替えが行われない限り、（結合または分割により）返された両者のノードセットに違いはありません。結合されたノードがグループ化や並べ替えメカニズムでどのように扱われるかの詳細については、: ノードテンプレートオペレーション を参照してください。

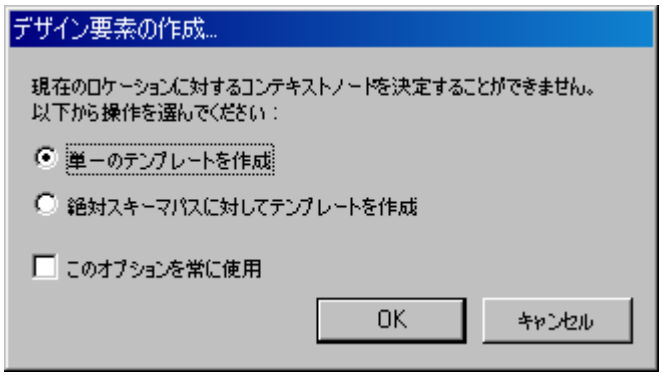
テンプレート マッチの編集

任意の（ユーザー定義または通常の）ノードテンプレートに対する選択は、新たな XPath 条件式により変更することができます。ノードテンプレートのテンプレート マッチを編集するには、ノードテンプレートを右クリックし、**テンプレート マッチの編集**コマンドを選択します。XPath 条件式を編集するためのダイアログが表示されるので、新たなノードを選択する XPath 条件式を入力し、「OK」をクリックします。

(C) 2015-2021 Altova GmbH

ユーザー定義テンプレートにノードを追加する

SPS のスキーマソースコンテキスト内に配置することができないノードまたはシーケンスに対してユーザー定義テンプレートが作成された場合、スキーマツリーから得られたノードがユーザー定義テンプレートへ追加される時に、新たなノードのコンテキストが決定できなくなります。そのため、新たなノードが、(i) 名前で(つまり相対パスで)参照されるのか、または(ii) スキーマソースのルートからのフルパスにより参照されるのが決定するよう促されます(以下のスクリーンショットを参照)。



どのような動作を行うか、確認するのがデフォルトの振る舞いになっています。「ツール| オプション ダイアログ」のデザインタブにて、デフォルトの振る舞いを変更することができます。

関連項目

- SPS ファイル コンテンツ
- ノードテンプレートオペレーション
- ユーザー定義要素

5.5 ユーザー定義要素、XML テキストブロック

ユーザー定義要素 とユーザー定義 XML テキストブロックにより、それぞれ(i) 任意の要素、(ii) 任意のXML テキストブロックをデザインへ挿入することができます。これらの機能により、ソーススキーマならびに StyleVision デザイン要素のパレットから得られたXML 要素やデザイン要素以外の要素を追加することが可能になり、(i) 独自に定義した要素に対するテンプレート(ユーザー定義要素)や、(ii) 独自のオブジェクト(例えばActiveX オブジェクト)を作成する、独立したXML コード(ユーザー定義ブロック)を作成することができるようになります。

ユーザー定義要素とユーザー定義 XML テキストブロックには重要な違いがあります。ユーザー定義要素は(属性を伴う)単一のXML 要素に対するテンプレートノードとしてデザインにて作成され、このテンプレート内にある全てのコンテンツを明示的に作成する必要があります。コンテンツは、SPS で使用することができる様々なデザイン要素により構成されます。ユーザー定義 XML テキストブロックにデザイン要素を含める必要はなく、独立したブロックとして扱われます。ユーザー定義要素は空の状態で作成され、多くのコード行を必要とするオブジェクトを作成する必要はありません。

メモ ユーザー定義要素ならびにユーザー定義テキストブロックは Enterprise エディションの Altova 製品に搭載された Authentic View でのみサポートされます。

関連項目

- ユーザー定義要素
- ユーザー定義 XML テキストブロック
- ユーザー定義テンプレート

5.5.1 ユーザー定義要素

ユーザー定義要素とは、SPS の任意のスキーマソース内には無い要素を出力にて生成することができる要素のことです。この機能により、任意の名前空間(例えばHTML やXSL-FO)にある要素を、デザイン内の任意の場所に挿入することが可能になります。その後、挿入された要素内部にSPS デザイン要素を挿入することができます。

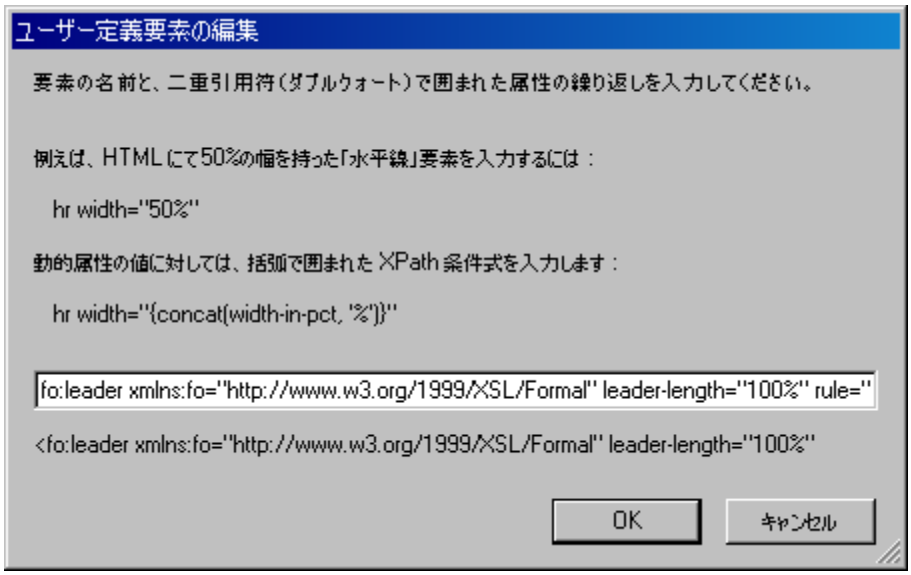
メモ ユーザー定義要素は、Enterprise エディションの Altova 製品に搭載された Authentic View でのみサポートされます。

ユーザー定義要素の挿入

以下の方法によりユーザー定義要素を使用することができます:

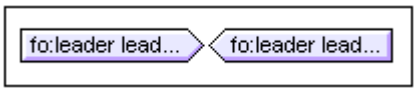
1. デザイン内にてユーザー定義要素を挿入する場所を右クリックします。
2. コンテキストメニューから「**ユーザー定義アイテムを挿入 | ユーザー定義要素**」を選択します。
3. ダイアログが表示されるので、要素名や属性とその値、そして(ドキュメントに要素が含まれない場合は)名前空間宣言を入力します。

(C) 2015-2021 Altova GmbH



上のスクリーンショットでは、leader という名前の XSL-FO 要素が作成されます。fo: というプレフィックスが与えられ、そのプレフィックスが xmlns:fo="http://www.w3.org/1999/XSL/Format" という名前空間宣言に紐付けられています。要素には leader-length や rule-style という属性が含まれており、それぞれ値が与えられています。要素やその属性、そして名前空間宣言は山かっこ("<" ならびに ">")無しで入力してください。

4. 「OK」をクリックして、デザインへ要素を挿入します。開始ならびに終了タグだけを伴う空のテンプレートとして要素がデザイン内に表示されます(以下のスクリーンショットを参照)。



5. その他のテンプレートと同様に、テンプレートへコンテンツを追加することができます。ユーザー定義要素には静的コンテンツや XML ドキュメントから得られた動的コンテンツだけではなく、その他のユーザー定義要素を含めることができます。

メモ　特定の出力でだけ表示を行うように追加されたユーザー定義要素を出力ベースの条件で囲むことにより、その他の出力にて予期しない結果を回避することができます。

関連項目

- ユーザー定義 XML テキストブロック
- ユーザー定義テンプレート

5.5.2　ユーザー定義 XML テキストブロック

ユーザー定義 XML テキストブロックは、SPS から生成された XSLT コード内に挿入することができる XML フラグメントのことです。デザイン要素を追加することができない独立したブロックとして、SPS デザイン内部に配置されます。このような XML テキストブロックは、例えばスタイルシート内の目的の場所に XSLT コードを挿入するといった時に使用されます。

この機能により、XSLT フラグメントやカスタマイズされたコードを、デザイン内に挿入するといった操作を行うことができます。例えば、HTML の SCRIPT 要素内部に ActiveX オブジェクトを挿入することができます。

メモ この機能は、**Enterprise エディションの Altova 製品**（StyleVision、Authentic Desktop、Authentic Browser、XMLSpy）に搭載された Authentic View でのみサポートされます。

ユーザー定義 XML テキストブロックの挿入

以下の操作により XML テキストブロックを挿入することができます:

1. ユーザー定義ブロックを挿入するデザイン内の位置を右クリックします。
2. コンテキストメニューから、「**ユーザー定義アイテムを挿入 | ユーザー定義ブロック**」を選択します。
3. 表示されるダイアログにて、挿入する XML テキストブロックを入力します。XML テキストブロックは整形式でなければならない点に注意してください。

上のスクリーンショットでは、HTML のリストを生成する XML テキストブロックが追加されます。

4. 「**OK**」をクリックして要素をデザインに挿入します。XML テキストブロックがテキストボックスとして表示されます。

メモ 特定の出力でだけ表示を行うように追加されたユーザー定義 XML テキストブロックを出力ベースの条件で囲むことにより、その他の出力にて予期しない結果を回避することができます。

関連項目

- ユーザー定義要素
- ユーザー定義テンプレート

(C) 2015-2021 Altova GmbH

5.6 テーブル

SPS デザインでは、SPS テーブルとCALS/HTML テーブルという、2種類のテーブルを使用することができます。このセクションでは、これらのテーブルを使用する方法について説明します。

SPS テーブル

SPS テーブルは デザインのコンポーネントで、デザイン内にて構成/フォーマットされます。SPS テーブルはデザイン内の任意の場所で作成することができ、作成できるSPS テーブルの数に制限はありません。

SPS テーブルはデータを表示するためだけに使用されるもので、Authentic View と出力 フォーマットにおける文法に従って表示されます。SPS テーブルの構造がテーブルセルのコンテンツはXML ドキュメント内部のノードから得られる場合もありますが、構造そのものが**XML ドキュメント内部のノードで表示されることはありません。**

SPS テーブルには以下の2種類が存在します:

- **静的テーブル**とは、SPS のデザイナーにより作成されるテーブルのことです。テーブルの構造を作成した後に、各セルのコンテンツを個別に定義します。セルのコンテンツはスキーマツリーにある任意の場所から取得することができ、異なる型も扱うことができます。静的テーブルの行は繰り返される構造では無いことに注意してください。繰り返しが行われないため、テーブルは静的と呼ばれており、XML コンテンツからは影響を受けない固定された構造となります。
- **動的テーブル**とは、XML ドキュメントにて繰り返し出現するデータ構造に対応したテーブルのことです。テーブルはサブ構造 –少なくとも1つの子属性または子要素– を持ったスキーマ要素に対して作成することができます。指定されたサブ構造のインスタンスが複数回出現する場合、その要素が繰り返し表示されます。要素の各インスタンスは動的テーブルの行として扱われ、全てまたは一部の子要素や属性がテーブルのカラムになります。そのため、動的テーブルの構造はXML ファイルのコンテンツに依存し、コンテンツにより動的に変化します。

CALS/HTML テーブル

CALS テーブルまたはHTML テーブルのコンテンツモデルは、DTD やスキーマの拡張によりXML ドキュメントにて(CALS またはHTML の仕様に従うかたちで)定義されます。SPS デザインでは、CALS/HTML テーブルがテーブルとして処理されるように指定することができます。この場合、CALS/HTML テーブルを表すXML データ構造から出力フォーマットにおけるテーブルマークアップが生成されます。CALS/HTML テーブルのフォーマットはXML インスタンスドキュメントやSPS (またはそれら両方)にて指定することができます。

HTML テーブルのHTML プレビューを以下に示します:

Name	Phone
John Merrimack	6517890
Joe Concord	6402387

上のテーブルに対応するHTML コードのフラグメントを以下に示します:

```
<テーブル border="1" width="40%">
 <tbody>
  <tr>
   <td>Name</td>
   <td>Phone</td>
  </tr>
  <tr>
   <td>John Merrimack</td>
   <td>6517890</td>
  </tr>
  <tr>
   <td>Joe Concord</td>
   <td>6402387</td>
  </tr>
 </tbody>
```

```
</テーブル>
```

オリジナルのXMLドキュメントは以下のようになります:

```
<phonelist border="1" width="40%">
  <items>
    <person>
      <data>Name</data>
      <data>Phone</data>
    </person>
    <person>
      <data>John Merrimack</data>
      <data>6517890</data>
    </person>
    <person>
      <data>Joe Concord</data>
      <data>6402387</data>
    </person>
  </items>
</phonelist>
```

XMLドキュメント内の要素名がテーブルを表すものでは無いことに注目してください。**テーブルの構造**は、HTMLまたはCALSテーブルモデルに対応している必要があります。以下を参照ください:

- XML要素だけがHTMLカラム要素の<td/>に対応していることに注目してください。
- CALS/HTMLテーブルは、table要素に相当する要素が許可されている場所であれば、XMLドキュメント内の任意の場所に挿入することができます。
- Authentic Viewでは、データをテーブルセルに直接入力することができます。データは対応するCALS/HTMLテーブル要素に対応するコンテンツとして使用されます。
- CALS/HTMLテーブルのフォーマットプロパティはXMLドキュメントから得られるほか、SPSデザインにて指定することもできます。

デザイナーに関する概要

ドキュメントデザイナーは、以下の点に留意してください:

- SPSテーブルの**構造**はSPSにて定義される一方で、CALS/HTMLテーブルの構造はスキーマにて指定され、CALS/HTMLテーブルモデルに従う必要があります。スキーマ内の要素名はCALSまたはHTMLテーブルモデル内の名前と異なるものでも許されます。
- SPSテーブルの**colspan**と**rowspan**はSPSで指定されます。CALS/HTMLテーブルの場合、colspanならびにrowspanはXMLインスタンスドキュメントにて指定されます。
- SPSテーブルの**テーブルフォーマット**はSPSにて指定されます。CALS/HTMLテーブルのフォーマットは、XMLインスタンスドキュメント、またはSPSにて指定されます。

関連項目

- 静的テーブルの作成
- 動的テーブルの作成
- テーブル内の条件付き処理
- CALS/HTMLテーブル

5.6.1 静的テーブル

以下の操作により静的なテーブルを作成することができます:

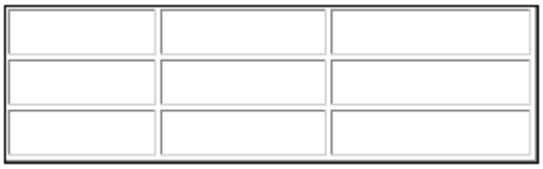
(C) 2015-2021 Altova GmbH

1. メニューアイテムから「**テーブル | テーブルを挿入**」または「**挿入 | テーブルを挿入**」を選択するか、デザイン要素の挿入ツールバーにある**テーブルの挿入** アイコンをクリックします。
2. テーブルの作成ダイアログが表示されます(以下のスクリーンショットを参照)。

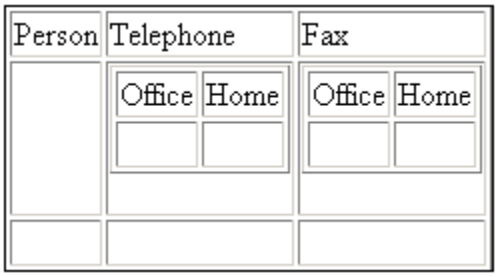
静的テーブル をクリックします。

3. テーブルの挿入ダイアログが表示されるので、テーブルの大きさを入力し、使用することができる全ての幅を使用するかを指定します。

4. 「OK」をクリックします。下に示されるように、指定された次元を持つテーブルが作成されます。



5. StyleVision に備わっている通常の機能を使用することで、テーブルのセルにコンテンツを入力することができます。直接入力されたテキストや、スキーマツリーからドラッグされた要素、またはイメージなどのオブジェクトや入れ子になったテーブルをテーブルに含めることができます。入れ子構造になったテーブルの例を以下に示します。



静的な SPS テーブルは、スキーマ構造のいたるところに散りばめられた XML データを表示し、静的なコンテンツ(XML ソースから与えられてはいないコンテンツ)を表示するのに適したテーブル形式です。

カラム、行、テーブルを削除する

カラム、行、またはテーブルを削除するには、目的のカラム、行、テーブル内にカーソルを移動させ、メニューアイテムの「**テーブル | 行を削除**」、「**テーブル | 列を削除**」、または「**テーブル | テーブルを削除**」を選択します。

ツールバー テーブルの編集 アイコン

デフォルトでツールバーの2段目に配置されているテーブル編集アイコンは、**テーブル** メニューコマンドに対するショートカットとなっています。これらのコマンドを使用することで、静的テーブルの構造を編集し、フォーマットプロパティを割り当てることができます。これらのアイコンは動的 SPS テーブルに対しても使用することができます。CALS/HTML テーブルのフォーマットは別の方法で行われるため、これらのアイコンは **CALS/HTML テーブルに対して使用できません。**

関連項目

- 動的テーブルの作成
- テーブル内の条件付き処理
- SPS デザインビュー内のテーブル
- 静的と動的テーブルのフォーマット
- CALS/HTML テーブル

5.6.2 動的テーブル

以下の操作により動的テーブルを挿入することができます:

1. メニューコマンドからから「**テーブル | テーブルを挿入**」または「**挿入 | テーブルを挿入**」を選択するか、デザイン要素の挿入ツールバーにある**テーブルの挿入** アイコンをクリックします
2. テーブルの作成ダイアログが表示されます(以下のスクリーンショットを参照)。ツールバーにあるテーブルの挿入アイコンをクリックした場合、テーブルを挿入するデザイン内の箇所をクリックした段階でダイアログが表示されます。

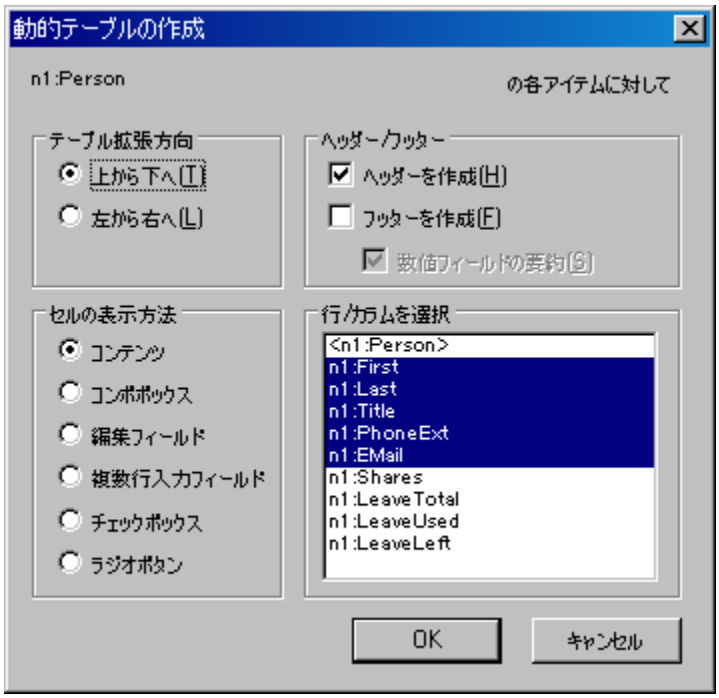
動的テーブルボタンをクリックします。

3. XPath セレクターダイアログが表示されます。XPath コンテキストが、テーブルが挿入される場所になっていることに注目してください。この値をダイアログで変更することはできません。動的テーブルとして作成されるノードを選択してください。以下のスクリーンショットでは、n1:Department がコンテキストノードとなり、n1:Person がテーブルとして作成されるノードとして選択されています。

(C) 2015-2021 Altova GmbH

ユーザー定義 XPath オプションを選択した場合、動的テーブルとして作成されるノードを XPath 条件式により選択することができます。

4. 「OK」をクリックします。動的テーブルの作成ダイアログが表示されます（以下のスクリーンショットを参照）。



5. デザインウィンドウにドラッグされた要素の子要素や属性が表示されます。「行/カラムを選択」では、テーブルのカラムとして使用することができるリストが表示されます。カラムとして使用する子要素や属性を選択してください(上の例では、Shares、LeaveTotal、LeaveUsed、LeaveLeft が選択解除されています)。その他のオプションについては以下で説明されます。設定が完了したら「OK」をクリックします。カラムは子要素や属性に対してだけ作成され、それより下の階層に対しては作成されないことに注意してください。

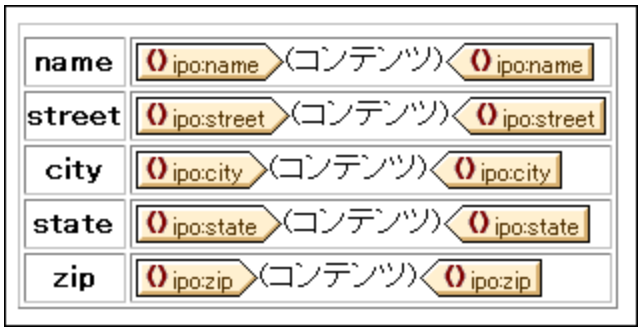
メモ ユーザー定義された XPath を指定して、動的テーブルとして作成するノードを選択すると、ターゲットとなっているノードを曖昧でないかたちで StyleVision により判断することができなくなります。このような場合、テーブルのフィールド(カラム)を選択するための子属性/要素が、動的テーブルの作成ダイアログに表示されず、作成されたテーブルには手動でノードコンテンツを入力する必要があります。ノードコンテンツは、テーブルとして作成される選択ノードの子属性/要素である必要があります。

メモ テーブルとしてスキーマノードを作成する、その他の方法として、スキーマツリーからデザインへノードをドラッグし、ドロップした時にテーブルとして作成することもできます。

縦向きと横向きのテーブル

テーブルが上から下に向けて拡張される場合、テーブルは以下のように表示されます:

テーブルが左から右に拡張される場合、テーブルは以下のように表示されます:



ヘッダーとフッター

カラムと行にはカラムならびに行要素の名前となるヘッダーを与えることができます。カラムヘッダーは各カラムの上部に表示されます。行ヘッダーは、各行の左側に表示されます。ヘッダーを含めるには、動的テーブルの作成ダイアログにある、ヘッダーを作成チェックボックスにチェックを入れます。テーブルの拡張方向に上から下が選択されている状態でヘッダーが作成されると、テーブルボディーの上にヘッダー行が作成されます。テーブルの拡張方向が左から右になっている場合、テーブルボディーの左側にカラムのヘッダーが作成されます。

フッターを含めるには、フッターを作成チェックボックスにチェックを入れます。ヘッダーと同様に、フッターはカラム(カラムの下部に表示)と行(行の右側に表示)に対して作成することができます。数値フィールドの要約チェックボックスにチェックがはいっている場合、数値カラムや行の合計がフッターに表示されます。

テーブル メニューからは、ヘッダーやフッターのセルを統合ならびに分割し、行やカラムの挿入、追加、削除を行うことができます。この機能により、ヘッダーやフッターの構造をとても柔軟に構成することができるようになります。更に、ヘッダーとフッターには条件テンプレートや自動計算を含む任意の型の静的ならびに動的コンテンツを含めることができます。

メモ ヘッダーやフッターは、動的テーブルが定義された際に作成する必要があります。ヘッダーの作成を行うには、動的テーブルの作成ダイアログにある、ヘッダーを作成またはフッターを作成オプションにチェックを入れます。動的テーブル内部で行の追加を行なっても、新たな行が作成されるだけで、ヘッダーやフッターは作成されません。この違いは重要で、ヘッダー/フッターを作成コマンドにより、テーブルを含む各ページの上部または下部にヘッダーやフッターが表示されるようになります。行が挿入または追加された場合、動的テーブルとして作成された要素の各出現がそれぞれの行に表示されます。

入れ子になった動的テーブル

ある動的テーブルに対応する要素の子要素を使用することで、動的テーブル内部に別の(入れ子になった)動的テーブルを追加することができます。入れ子になった動的テーブルの作成は以下のように行うことができます:

1. 要素の子要素がテーブルのカラムとして作成されるような外側の動的テーブルを作成します。
2. デザインビューにある動的テーブルにて、子要素を右クリックします。
3. コンテキストメニューから「**変更 | テーブル**」を選択し、動的テーブルの作成ダイアログを表示します。
4. ネストされる動的テーブルのプロパティを定義します。

動的テーブルを静的テーブル内にネストするには、動的テーブルとして作成される要素を、目的の静的テーブルのセルへドラッグし、ドロップした際にコンテキストメニューから「**テーブルの作成**」を選択してください。

テキストコンテンツを含む要素をテーブルに収める

子要素に対してカラムや行を作成するには、テーブルとして作成された要素内に**子要素ノード**や**属性ノード**が含まれていなければならず、**子テキストノード**は処理されません。テキストノードの内容をカラムに表示するには、例えばTableElement という要素からテーブルを作成する場合、Text という名前の子要素を新たに作成して、TableElement/Text 要素内に目的のテキストノードを挿入します。この操作によりTableElement を動的テーブルとして作成することができます。各行のText にはテキストノードが含まれ、テーブルの各行における内容は、TableElement 要素の出現内容に対応することになります。

テーブルボディーセルのコンテンツ

動的テーブルを作成すると、ノードコンテンツを任意の StyleVision コンポーネントとして作成することができます。上の例では、テーブルボディーのセルがコンテンツとして作成されました。動的テーブルの作成ダイアログでは、セルの表示方法に**コンテンツ**が選択されていました。テーブルはデータ入力デバイスとして作成することもできます。以下の点に留意する必要があります:

- 表示方法の設定は、テーブル全体に対して適用されます。個々のセルの表示を変更するには、テーブルを作成した後に個々のセルを編集する必要があります。セルを右クリックして、コンテキストメニューの「**変更**」から、目的のセルコンテンツ型を選択してください。
- セルを要素コンテンツとして作成し、その要素が派生型である場合、セルの文字列コンテンツと、その派生要素全てが連結されたかたちでセルに表示されます。

カラム、行、テーブルを削除する

カラム、行、テーブルを削除するには、削除するカラム、行、テーブル内にカーソルを配置し、メニューアイテムから「**テーブル | カラムを削除**」、「**テーブル | 行を削除**」、または「**テーブル | テーブルを削除**」をそれぞれ選択してください。テーブルが入れ子になっている場合、「**テーブル | テーブルを削除**」コマンドを使用することで、カーソルが含まれているテーブルを削除することができます。

ツールバー テーブルの編集 アイコン

ツールバーに用意されているテーブル編集アイコンはテーブルメニューコマンドへのショートカットとなります。これらのコマンドを使用することで、動的テーブルの構造やフォーマットプロパティを挿入、削除、そして編集することができます。これらのアイコンは静的テーブルに対しても使用することができます。CALS/HTML テーブルのフォーマットは異なる方法で行われるため、これらのアイコンを**CALS/HTML テーブルへ使用することはできません**。CALS/HTML テーブルは、StyleVision からのみ有効にすることになります。

グローバルテンプレート内の動的テーブルの作成

動的テーブルはグローバルテンプレート内にある要素でも作成することができます。テンプレートの作成は、メインテンプレートの要素と同様の方法で行われます。グローバルテンプレートでは、グローバルテンプレートノードの**派生型要素**に対してしか動的なテーブルを作成することができず、グローバルテンプレート要素そのものに対して作成することはできません。例えば、グローバルテンプレート内にある authors という要素に対して動的なテーブルを作成するには、authors 要素の親要素内に動的テーブルを作成する必要があります。authors 要素のグローバルテンプレート内部にテーブルを作成することはできません。

関連項目

- 静的テーブルの作成
- テーブル内の条件付き処理
- SPS デザインビュー内のテーブル
- 静的と動的テーブルのフォーマット
- CALS/HTML テーブル

5.6.3 テーブル内の条件付き処理

静的ならびに動的なテーブルにある個々のカラムや行、またはカラムや行のヘッダーに対して条件付きの処理をセットすることで、条件の評価結果に従うかたちでカラムや行、またはヘッダーを表示/隠すことができます。条件の結果が true となる場合、カラムや行、ヘッダーが表示されます。

(C) 2015-2021 Altova GmbH

条件付き処理の追加と編集

カラムや行、ヘッダーに対して条件付き処理を追加するには、目的のデザインコンポーネントを右クリックして、**条件処理の編集**を選択します。（下のスクリーンショットでは、左上のカラムヘッダーデザインコンポーネントがハイライトされて表示されています。2番目のカラムデザインコンポーネントは青でアウトラインされています。行のコンポーネントはカラムヘッダーデザインコンポーネントの下に表示されています。）

条件処理の編集を選択することで、XPath 式の編集ダイアログが表示され、条件のXPath 条件式を入力することができます。以下にあるような用途に対して条件付き処理を使用することができます:.

- テーブルのカラムや行に対して false() という XPath 条件式を入力することで、それらを隠し、true() により表示させる。
- カラム内にある値の合計が一定の値を超えた場合にだけ、そのカラムが表示される。
- カラムや行にある全てのセルになんらかの値が与えられている（空でない）場合にだけ、そのカラムや行が表示される。
- 特定のセル値がカラムや行に存在する場合にだけ、そのカラムや行が表示される。

既に作成された条件を編集するには、そのデザインコンポーネントを右クリックして、**条件処理の編集**を選択してください。XPath 式の編集ダイアログが表示され、条件の評価を行うことができます。

条件処理の削除

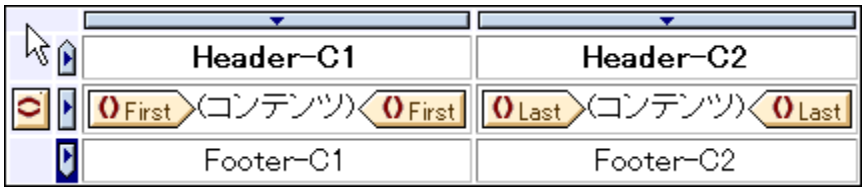
カラムや行、ヘッダーの条件処理を削除するには、目的のデザインコンポーネントを右クリックして、**条件処理のクリア**を選択してください。

関連項目

- 静的テーブルの作成
- 動的テーブルの作成
- SPS デザインビュー内のテーブル
- 静的と動的テーブルのフォーマット
- CALS/HTML テーブル

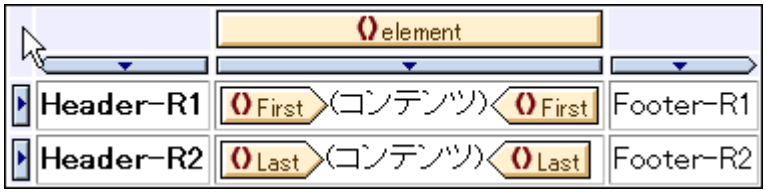
5.6.4　デザインビュー内のテーブル

静的ならびに動的 SPS テーブルのコンポーネントを、テーブルマークアップが有効になっている状態（「**テーブル | テーブルマークアップの表示**」）で以下に示します:



上のスクリーンショットに示されるテーブルは、上から下に展開され、ヘッダーとフッターが含まれているという単純なものです:

- 下向きの矢印を含む長方形により、カラムのサインが示されます。カラムのサインはカラムの上位に配置されます。(そのカラムに対してフォーマットプロパティを割り当てるなどのために)カラム全体を選択するには、そのカラムのサインをクリックします。
- 行は、右向きの矢印が含まれるサインにより示されます。行のサインをクリックして、行全体を選択することができます。
- 上から下に展開されるテーブルでは、ヘッダーとフッターがそれぞれ上向き下向きのアイコンにより表されます(上のスクリーンショットを参照)。左から右へ展開されるテーブルでは、ヘッダーとフッターは、それぞれ左と右方向のアイコンアイコンにより表されます(下のスクリーンショットを参照)。
- テーブル全体を選択するには、テーブルの左上(上、そして下にあるスクリーンショットにてマウスポインターが配置されている場所)をクリックします。
- テーブル行またはカラムが選択されると、その背景が濃い青色でハイライトされます。上のスクリーンショットでは、フッターが選択されています。
- 上から下方向に展開されるテーブルでは、作成されたテーブルの要素がカラム-行グリッドの更に外側にある一番左側に表示されます(上のスクリーンショットを参照)。左から右方向に展開されるテーブルでは、作成されたテーブルの要素が、カラム-行グリッドの上に表示されます(以下のスクリーンショットを参照)。



カラムや行、またはテーブルが選択された後には、スタイルならびにプロパティを、スタイルまたはプロパティウィンドウにてそれぞれセットすることができます。

ドラッグ アンド ドロップ 機能

(静的または動的な)SPS テーブルのカラムと行をドラッグして、同一テーブル内にドロップすることで、位置を変更することができます。

行とカラムをテンプレートのタグで囲む/削除する

行またはカラムのサインを右クリックして、コンテキストメニューから「**タグで囲む | テンプレート**」または「**タグで囲む | ユーザー定義テンプレート**」を選択することで、行やカラムをテンプレートのタグで囲むことができます。その後スキーマツリーからノードを選択、または[ユーザー定義テンプレート](#)に対して XPath 条件式を入力することで、行やカラムの周りにテンプレートが作成されます。

行またはカラムの周りに配置されたテンプレートは、テーブル内にある行やカラムに影響を与えること無く削除することができます。テンプレートタグが選択されている状態で、Delete キーを押下してください。

行やカラムのコンテンツを削除すること無くテンプレートの削除を行い、必要な場合行やカラムを別のテンプレートで囲むといったことを、テンプレートにより実現することができます。[ユーザー定義テンプレート](#) を使用することで、行またはカラム内から(例えば自動計算を使用することで)テンプレートマッチの結果を使用することができます。

(C) 2015-2021 Altova GmbH

関連項目

- テーブルメニュー
- 静的テーブルの作成
- 動的テーブルの作成
- 静的と動的テーブルのフォーマット
- CALS/HTML テーブル
- ノードテンプレートオペレーション
- ユーザー定義テンプレート

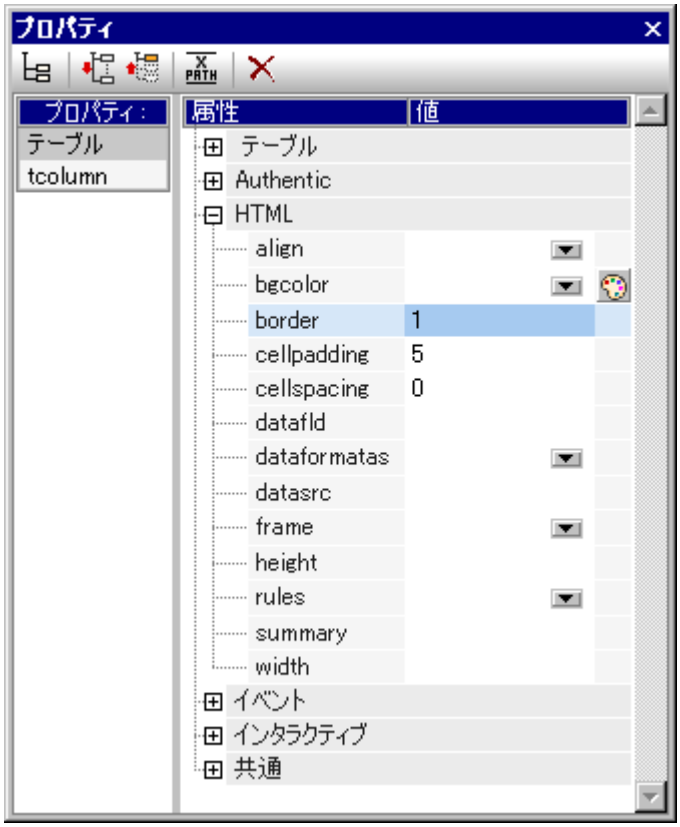
5.6.5 テーブルのフォーマット

以下の機能を使用することで、静的ならびに動的テーブルのフォーマットを行うことができます:

- (プロパティサイドバーにある)HTML テーブルフォーマットプロパティにて
- (スタイルサイドバーにある)CSS(スタイル)プロパティにて

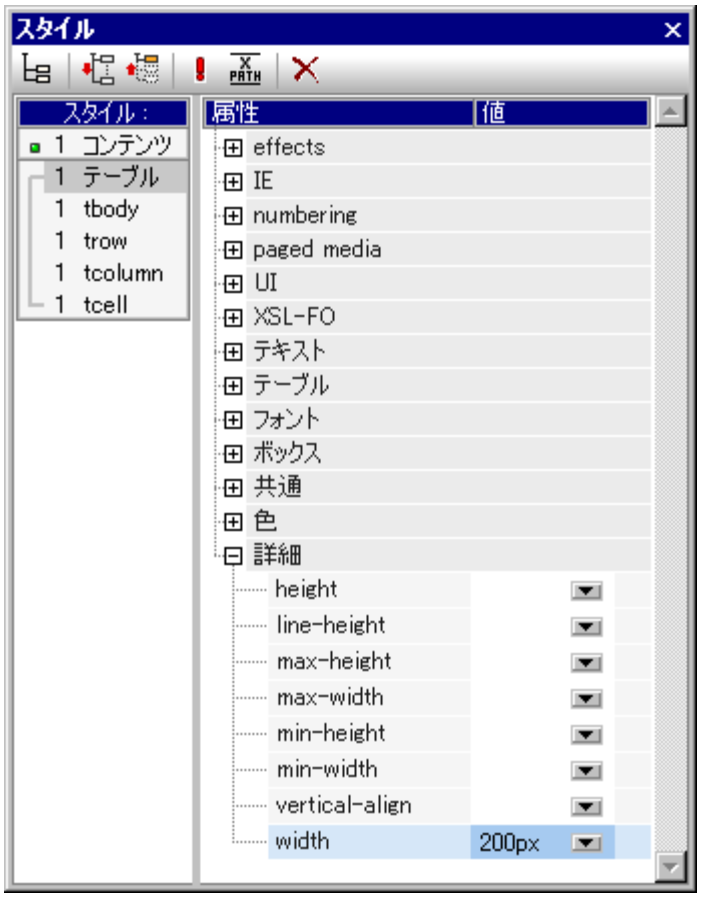
プロパティサイドバー

プロパティサイドバーでは、HTML テーブルフォーマットプロパティを利用することができます(以下のスクリーンショットを参照)。テーブルコンポーネントやサブコンポーネント(ボディー、行、カラム、セル)に適用されるこれらプロパティへは、プロパティの HTML グループからアクセスすることができます。



スタイル サイドバー

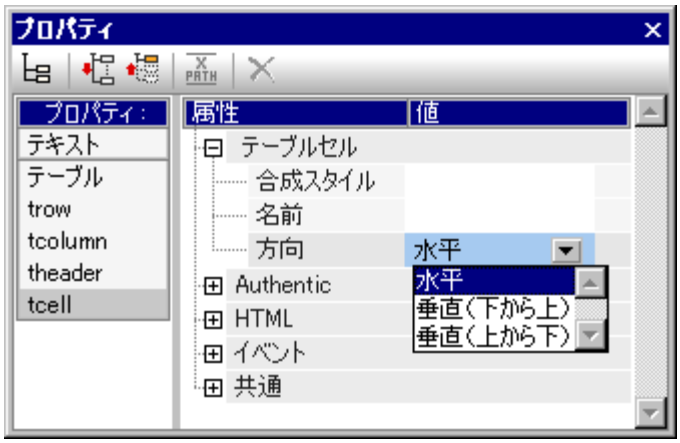
スタイルサイドバーでは、CSS テーブルフォーマットプロパティを利用することができます(以下のスクリーンショットを参照)。テーブルコンポーネントやサブコンポーネント(ボディー、行、カラム、セル)に対して CSS プロパティを使用することができます。



メモ テーブル内にある全てのセルが空になっている場合、Internet Explorer により行が縮退されており、見えなくなっている場合があります。この場合、特定のセルに空白を挿入して対応することができます。

垂直のテキスト

テーブルセル内にあるテキストは、時計回り、または反時計回りに90度回転させることができ、上から下または下から上方向に、テキストを垂直に表示することができます。デザインにてテーブルセル内のコンテンツを選択し、プロパティサイドバーにて tcell を選択します(以下のスクリーンショットを参照)。**テーブルセル**プロパティグループにある**方向**プロパティの値を選択してください。



以下の点に注意してください：

- 出力では回転された状態で表示されますが、デザインでは水平のまま表示されます。
- このプロパティはテキストに対してだけ適用されるべきもので、その他のコンテンツに対して使用しないでください。
- テーブルセル内のテキスト以外にも、テキストボックス内のテキストに同様のプロパティを適用することができます。

プロパティならびにスタイルによるテーブルのフォーマット

フォーマットプロパティは、プロパティサイドバーとスタイルサイドバーの両方で利用することができます。両方のサイドバーで利用することのできるテーブルプロパティを以下のテーブルに示します。

テーブルコンポーネント	プロパティサイドバー	スタイルサイドバー
テーブル	border、frame、rules; cellpadding、cellspacing、bgcolor、height、width（スタイルサイドバーにてheight、width が指定された場合、ここでの値が上書きされます）	ボックススタイル内にあるborders とpadding、詳細グループ以下のheight とwidth（height とwidth の値はプロパティサイドバー内の値を上書きします）、color、font。text styles
ボディー	align、valign	height、vertical-align、color、font、text styles
カラム	align、valign	width、vertical-align、color、font、text styles、box styles
行	align、valign	height、vertical-align、color、font、text styles、box styles
セル	align、valign	height、width、vertical-align、color、font、text styles、box styles

高さと幅

テーブルや行、カラム、セルの高さや幅はスタイルサイドバー（の詳細グループ）にてセットすることができます。マウスを使ってデザインに表示されているテーブルやカラム、または行の大きさが変更されると、スタイルサイドバーの該当するスタイルにて、変更された値が自動的に入力されます。余白が指定されたセルに対してheight やwidth スタイルはサポートされません（row-spanned またはcolumn-spanned）。

テーブルの中央寄せ

テーブルの中央寄せを行うには、プロパティのHTML グループにある align プロパティの値を center にセットします。テーブルが選択されている状態で、「**テーブル | テーブルのプロパティ**」を選択することでも、align プロパティへアクセスすることができます。または、プロパティサイドバーにあるHTML グループからも align プロパティへアクセスすることができます。

PDF 出力にてテーブルの中央寄せを行うには、ご使用のFOP プロセッサーに従った設定を行う必要があります。FO の仕様に従えば、fo:table-and-caption 要素をともなう fo:table によりテーブルを囲み、fo:table-and-caption 要素のtext-align 属性の値にcneter を指定することになります。デザインにテーブルが挿入されただけでは、fo:table-and-caption 属性は作成されませんが、ユーザー定義要素 としてこの要素を追加することができます。Apache FOP プロセッサーを使用している場合、FOP のバージョンによってはfo:table-and-caption 要素がサポートされていないこともあります。この場合、幅が固定されたテーブルを作成し、(「**テーブル | テーブルのプロパティ**」からアクセスすることができる) テーブルプロパティのHTML グループにて width プロパティの値を 4in または 120mm に指定することで、簡単に中央寄せを行うことができます。

交互に表示される行に異なる背景色を与える

以下の方法により、動的テーブルで交互に表示される行に対して背景色をセットすることができます:

1. 背景色を指定する行のアイコンを選択します。テーブルは動的テーブルであり、各要素が行として表示され、テーブルとして作成されている要素に対応する行がデザインには表示されていることに留意してください。
2. 行のサインが選択された状態で、プロパティサイドバーの trow プロパティをクリックします。
3. bgcolor プロパティを選択します。
4. プロパティサイドバーにあるツールバー内にある XPath アイコンをクリックし、XPath 式 の編集ダイアログ にて、以下のような XPath 条件式を入力します:

   ```
   if ( position() mod 2 = 0 ) then "white" else "gray"
   ```

   この XPath 条件式により、偶数行の bgcolor が white (白) になり、奇数行の bgcolor が gray (グレー) となります。

上の方法を拡張することで、さらに複雑なフォーマットを行うこともできます。

動的テーブルの行に番号を付ける

XPath 条件式のposition() 関数を使用することで、動的テーブルの行に番号をつけることができます。テーブル内に行番号を表示するカラムを挿入し、そのカラムのセル内にposition() 関数を含むXPath 条件式が記述された自動計算を挿入します。動的テーブルにおける行に対応した要素はコンテキストノードとなるため、行要素セットにおける各行の位置がposition() 関数により返されます。

PDF 出力におけるテーブルヘッダーとフッター

テーブルがページをまたいで表示される場合、テーブルを含むページ全てにテーブルのヘッダーとフッターが表示されます。以下の点に留意してください:

- フッターに自動計算が含まれる場合、各ページの最後に表示される自動計算には、(それぞれのページに表示されている行ではなく) テーブル全体に対する自動計算の結果が含まれます。
- 個々のページにおいてヘッダーやフッターの表示を無効にすることはできません。例えば、(各ページに表示されるテーブルの最後ではなく) テーブル全体の最後の行にだけフッターを表示することはできません。

改ページが行われるごとに表示されるヘッダーやフッターの表示を回避するには、table 要素のtable-omit-header-at-break やtable-omit-footer-at-break プロパティ(属性) を使用してください。これらのプロパティへは、スタイルサイドバーのXSL-FO グループからアクセスすることができます。改ページにおけるヘッダーやフッターの表示を回避するには、それぞれの属性値をtrue にセットします。デフォルトの値は false になっており、値の指定を行わないと、改ページが行われるたびにヘッダーとフッターが挿入されます。

(C) 2015-2021 Altova GmbH

テーブルセルのコンテンツにハイフンを追加する
テーブルセル内のコンテンツにハイフンを追加するには、対応するブロックにてハイフンを追加してください。

関連項目

- 静的テーブルの作成
- 動的テーブルの作成
- SPS デザインビュー内のテーブル
- CALS/HTML テーブル

5.6.6 行と列の表示

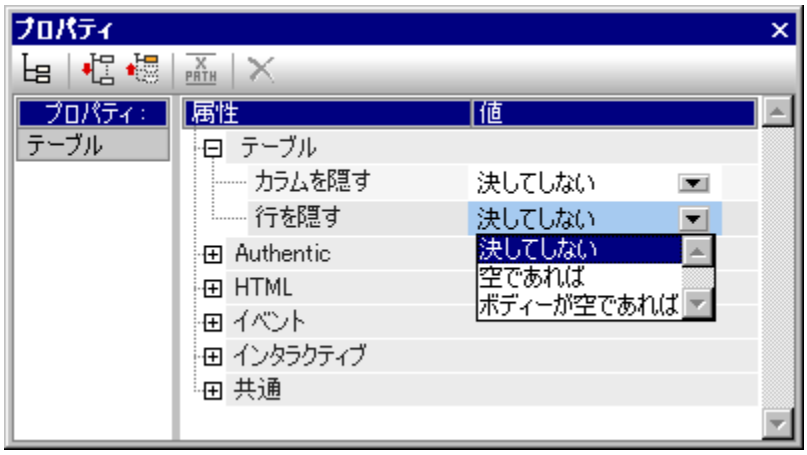
以下に示されるテーブルに対して用意された行ならびにカラム表示オプションは、HTML 出力に対してのみ使用することができます。これらの機能は**Authentic View** にてサポートされておらず、SPS のXSLT バージョンに XSLT 2.0 またはXSLT 3.0 を選択する必要があります。

- 空の行ならびにカラムを自動的に隠す。
- 各カラムに**閉じる**ボタンを配置して、個々のカラムを閉じる。
- 派生したリレーションシップが含まれる行要素に展開/縮退ボタンを表示する。

空の行ならびにカラムをデフォルトで隠す

以下の操作により、HTML 出力にて空の行またはカラムを隠すことができます:

1. デザインビューにて、テーブルまたはその一部（カラム、行、セル）を選択します
2. プロパティサイドバーにて、テーブルのプロパティ以下にあるテーブルグループを選択します（以下のスクリーンショットを参照）。



3. **カラムを隠す**または**行を隠す**プロパティに対する値を選択します。これら2つのプロパティで選択することができるオプションは、**行わない**、**空であれば**、**ボディーが空であれば**、と共通しています。**空であれば**オプションにより（ヘッダーとフッターを含む）全てのカラムまたは行が空であれば、そのカラム/行が隠されます。**ボディーが空であれば**オプションでは、ボディーが空である場合にカラム/行が非表示になります。

メモ　非 XBRL テーブルに（行やカラムのセルが結合された）行またはカラム領域が含まれる場合、空の行やカラムを隠す機能が動作しないこともあります。

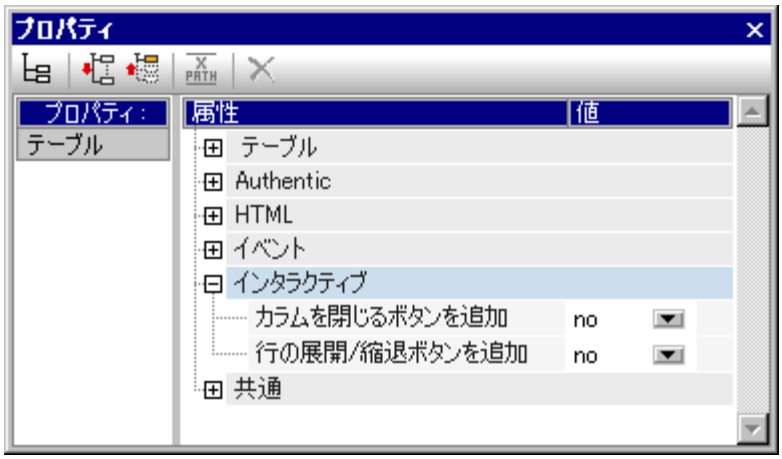
行を展開/縮退するためのカラムを隠す

HTML 出力において、テーブルの各カラムに**閉じる**ボタンを含めるように指定することができます（以下のスクリーンショットを参照）。**閉じる**ボタンをクリックすることで、ユーザーは個々のカラムを隠すことができます。ユーザーによりカラムが隠されると、最初のカラムにプラスシンボルが表示されます（以下のスクリーンショットを参照）。このシンボルをクリックすることで、隠されていた全てのカラムが再度表示されます。

Balance Sheet (in Millions)	2004-09-30	2004-07-01 - 2004-09-30	2003-12-31	2004-01-01
Assets, Total	€21.49		€24.02	
Current Assets, Total	€10.65		€12.32	
Non Current Assets, Total	€10.85		€11.7	
Liabilities and Equity, Total	€21.49		€24.02	
Liabilities, Total	€8.9		€10.79	
Minority Interests				
Equity, Total	€12.59		€13.23	
Issued Capital and Reserves	€12.59		€13.23	

その他にも HTML 出力では、要素名の隣に表示された展開/縮退（プラス/マイナス）シンボルにより、要素の子孫を表示することができます（上のスクリーンショットを参照）。HTML 出力にてこれらのシンボルをクリックすることにより、その行要素が展開または縮退されます。デザインでは、CSS プロパティから個々の行に対してインデントを指定することができます。

これら2つの機能の設定は、テーブルプロパティの**インタラクティブ**グループにて行うことができます（以下のスクリーンショットを参照）。



プロパティにセットすることができる値は（機能を有効にする）yes または（機能を無効にする）no となります。

関連項目

- 静的テーブルの作成
- 動的テーブルの作成
- SPS デザインビュー内のテーブル
- 静的と動的テーブルのフォーマット
- CALS/HTML テーブル

(C) 2015-2021 Altova GmbH

5.6.7 CALS/HTML テーブル

CALS/HTML テーブルではXML による階層構造が表示され、その要素により(i) CALS またはHTML テーブルの構造を定義、(ii) そのテーブルのフォーマットを指定、そして(iii) そのテーブルのコンテンツを含むことができます。XML の構造は、CALS またはHTML テーブルのモデルに正確に対応する必要があります。

以下の方法によりCALS/HTML テーブルを作成することができます:

1. XML 構造をCALS/HTML テーブル構造として定義する
2. テーブルのためのスタイルのフォーマットを指定する
3. SPS デザイン内にCALS/HTML テーブルを挿入する

CALS/HTML テーブルが出力されるようにする

XML ドキュメントにはテーブルの構造とコンテンツを定義するデータ構造を含めることができます。例えば、以下のXML データ構造はHTML テーブルモデルに対応しており、HTML テーブルモデルと同一の要素名が含まれています。

```
<table>
  <tbody>
    <tr>
      <td/>
    </tr>
  </tbody>
</table>
```

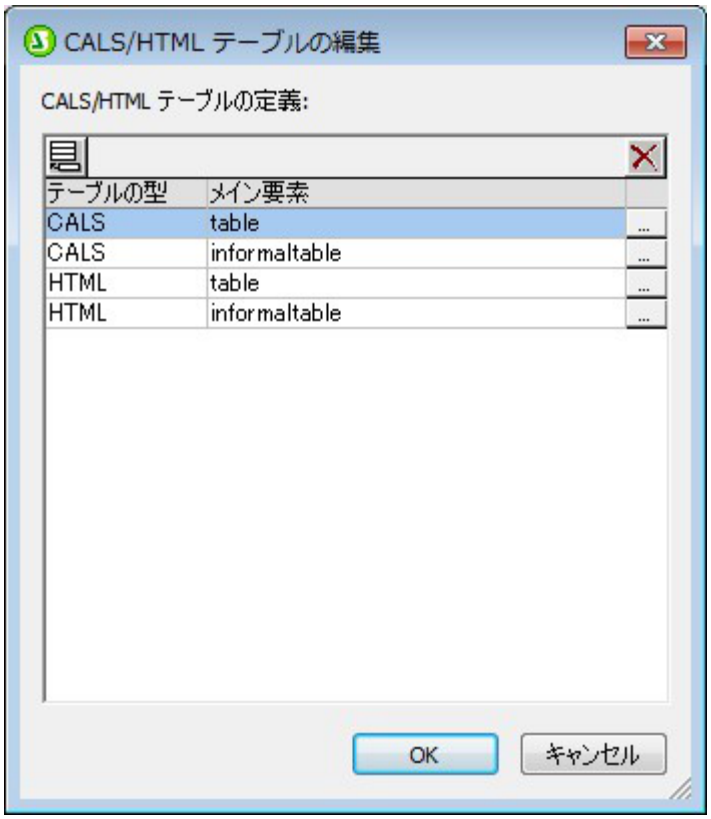
以下の例で示されるように、HTML テーブルモデルと要素名は異なるものの、HTML テーブルモデルの構造に対応したXML データ構造を使用することもできます。

```
<semester>
  <subject>
    <class>
      <student/>
    </class>
  </subject>
</semester>
```

XML ドキュメントにて定義されたこのテーブル構造は、異なる出力フォーマットにてテーブルを直接生成するために使用することができます。テーブルを生成するには、このXML データをCALS またはHTML テーブルとして定義する必要があります。XML データ構造がCALS や(デフォルトの)HTML テーブルとして定義されていない場合、データ構造内にある要素は一般的な非テーブル要素として扱われ、出力ドキュメントにテーブルマークアップが使用されることはありません。

以下の方法によりCALS/HTML テーブルマークアップを生成することができます:

1. メニューオプションから「**テーブル | CALS/HTML テーブルの編集**」を選択します。
2. ポップアップが表示されるので、CALS/HTML テーブルとして使用する XML データ構造を入力します。CALS テーブルモデルに関する情報については、CALS テーブルモデルat OASIS を参照ください。HTML テーブル構造を含むテーブル要素の例については、(プロジェクトウィンドウにある)Example プロジェクトフォルダーの Basic フォルダーに収められている HTMLTable1.sps を参照ください。スキーマ内にある要素を CALS または HTML テーブル要素として有効化するには、ダイアログの左上にある CALS/HTML テーブルボタンをクリックして、**CALS テーブルの追加**または **HTML テーブルの追加**コマンドを選択してください。以下のスクリーンショットでは、table ならびに informaltable が CALS テーブルとして、その他の HTML テーブルとともに作成されています。「**OK**」をクリックして確定してください。



3. 選択された型のテーブルで使われる全ての要素が含まれるダイアログ（HTML テーブルの編集）が表示されます（以下のスクリーンショットを参照）。このダイアログにデフォルトで表示される要素名は、選択されたテーブルモデルの要素名となっています（CALS または HTML）。SPS ファイルスキーマ内に、CALS/HTML テーブルモデルの名前と同じ名前の要素がある場合、それらの名前が黒色で表示されます（以下のスクリーンショットを参照）。表示されている要素名が SPS スキーマ内に存在しない場合、要素名は赤色で表示されます。表示されている要素名をダブルクリックして名前を編集することで、スキーマ内の要素名にマッチさせることができます。

(C) 2015-2021 Altova GmbH

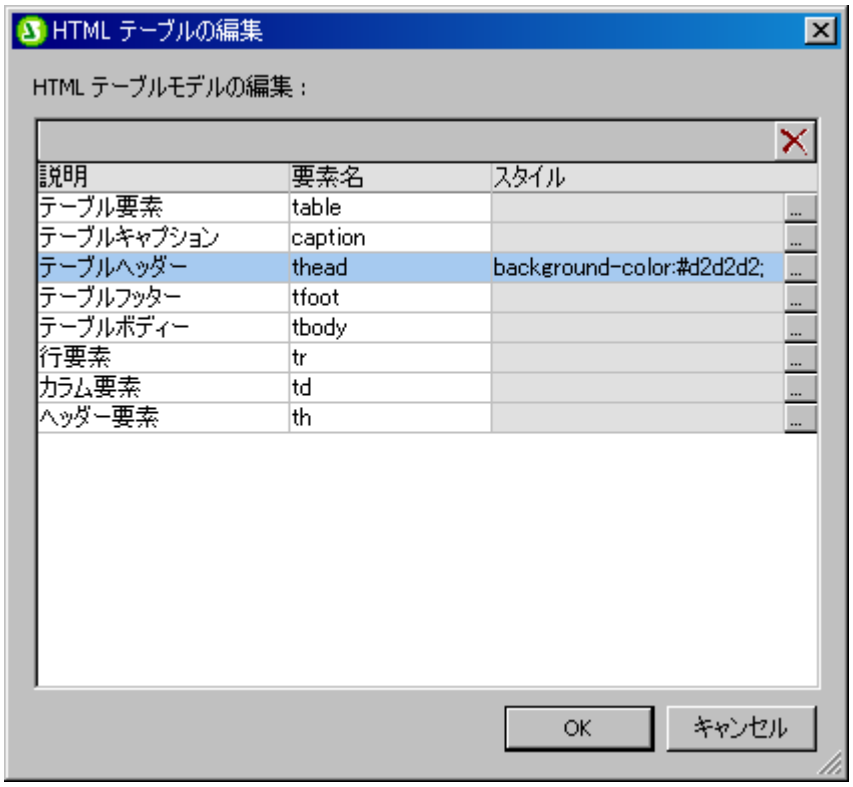
4. 「OK」をクリックして、XML データ構造を CALS または HTML テーブルとして定義します。
5. 入力できる XML データ構造の数に限りはありません（ステップ 2. のスクリーンショットを参照）。同一のメイン要素を CALS と HTML テーブル型の両方に使用することもできます。
6. CALS/HTML テーブルとして使用する XML データ構造の定義を完了したら、「OK」をクリックしてください。

CALS/HTML テーブルが定義され、XML データ構造が CALS/HTML テーブルとして正しく挿入されたら、データ構造がテーブルとして出力に送信されます。CALS/HTML テーブル定義を削除するには、**CALS/HTML テーブルの編集**ダイアログにて削除する定義を選択し、ペインの上部にある**削除**ボタンをクリックしてください。

テーブルのフォーマット

CALS/HTML テーブルは2種類の方法によりフォーマットを指定することができます:

1. ソース XML ドキュメントにあるフォーマット属性から:CALS ならびに HTML テーブルモデルではフォーマット属性を受け取ることができます。ソース XML ドキュメント内にそのような属性が存在する場合、出力のテーブルマークアップにそれらの値が渡されます。
2. CALS テーブルの編集または HTML テーブルの編集ダイアログに表示された個々の要素に対してスタイルを指定（以下のスクリーンショットを参照）。



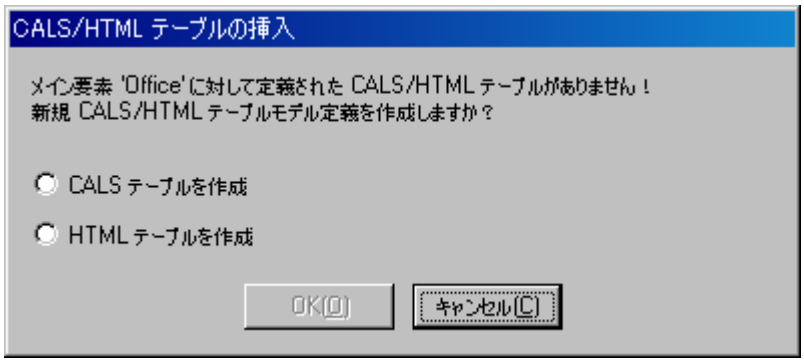
特定の要素に対してスタイルを割り当てるには、その要素のとなりにある**スタイルの追加**ボタンをクリックして、表示される スタイルサイドバーにて目的のスタイルを割り当てます。各スタイルが、独立した CSS 属性としてその要素へ追加されます。style 属性により追加されたスタイルは、個々の CSS 属性により追加されたスタイルよりも優先される点に注意してください。例えば、<thead style="background-color: red" bgcolor="blue"/> という要素では、style="background-color: red" 属性が bgcolor="blue" 属性より優先されます。

CALS/HTML テーブル定義にて割り当てられたスタイルを削除するには、その要素を選択（例えば、上のスクリーンショットにおける thead 要素）して、**削除**ボタンをクリックします。その要素のスタイルが削除されます。

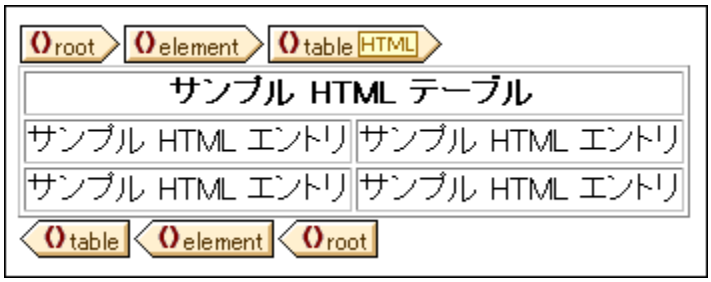
CALS/HTML テーブルをデザインに挿入する

CALS/HTML テーブル構造をデザインへ挿入するには2種類の方法があります:

1. テーブル要素の親を（コンテンツ）としてデザインに挿入する:親のコンテンツが処理される際に、テーブル要素も処理されます。CALS/HTML テーブル出力が有効になっている場合、要素がテーブルとして出力されます。そうでない場合、テキスト形式の出力が行われます。
2. テーブル要素をスキーマツリーからドラッグする:目的の場所にドロップし、**CALS/HTML テーブルの作成**コマンドを選択することで、CALS/HTML テーブルを作成することができます。要素がCALS/HTML テーブルとして定義されていない場合、CALS/HTML テーブルの挿入ダイアログが表示され、CALS または HTML テーブルとして要素を定義することができます（以下のスクリーンショットを参照）。



CALS/HTML テーブルとして要素がデザインで作成されると、CALS/HTML テーブルデザイン要素のプレースホルダがその場所へ挿入されます（以下のスクリーンショットを参照）。



テーブル要素のグローバルテンプレート

以下のテーブル要素のグローバルテンプレート が作成されている場合、それらがCALS/HTML テーブル出力内部で使用されます。CALS テーブル: title、entry。HTML テーブル: caption、th、td。

サンプルファイル

サンプルファイルは（プロジェクトウィンドウ内部の）Examples プロジェクトフォルダー内に収められています。

関連項目

- 静的テーブルの作成
- 動的テーブルの作成

5.7 リスト

SPS にて作成することができるリストには以下の2種類があります:

- 静的なリスト: コンテンツがSPS 内に直接入力されるリストです。リスト構造がXML ドキュメントの構造から動的に得られることはありません。
- 動的なリスト: XML ドキュメントから構造とコンテンツを動的に取得するリストです。

このセクションのサブセクションでは、これら2種類のリストを作成する方法について説明します。

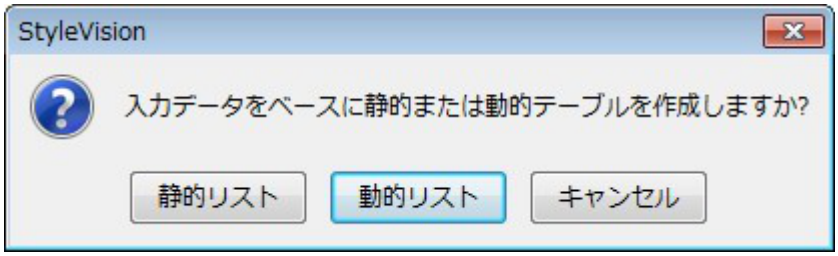
関連項目

- テーブルとの作業

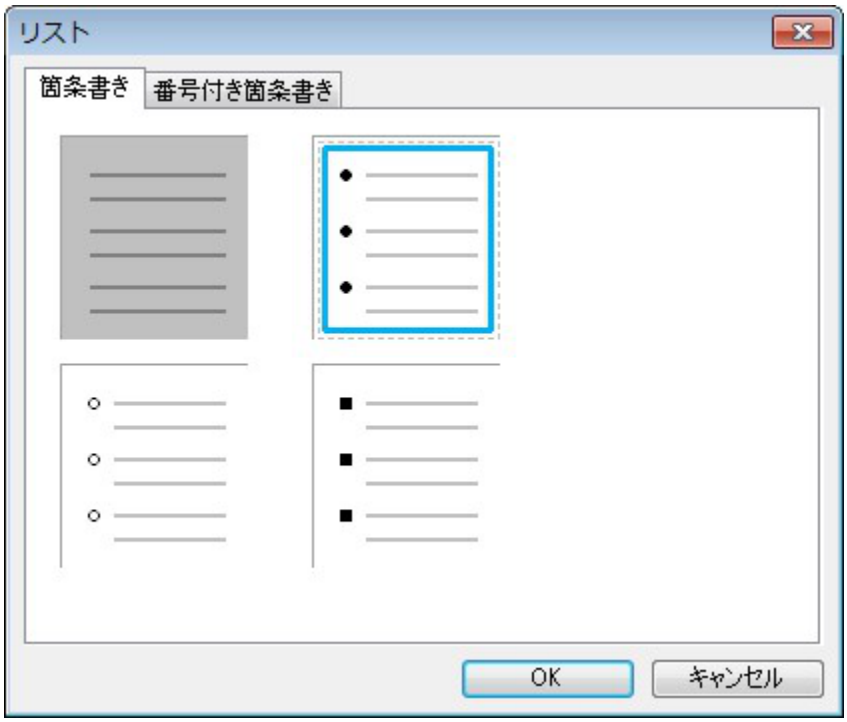
5.7.1 静的なリスト

静的なリストとは、リストアイテムのコンテンツを SPS に直接入力するリストのことです。以下の方法で静的なリストを作成することができます:

1. 静的なリストを作成する場所へカーソルを移動させ、メニューオプションから挿入 | 箇条書きを挿入を選択します(またはデザイン要素の挿入 ツールバー内にある中黒、または番号アイコンをクリックします)。この操作により、作成するリストの種類を選択するためのダイアログが表示されます(以下のスクリーンショットを参照)。



2. **静的リスト**をクリックします。箇条書き、または番号付きのリストを選択するダイアログが表示されます。(*下のスクリーンショット*).



3. 目的のリストアイテムマーカーを選択し、「OK」をクリックします。空のリストアイテムが作成されます。
4. 最初のリストアイテムに使用するテキストを入力します。
5. Enter キーを押下することで、次のリストアイテムを作成することができます。

ネスト構造のリストを作成するには、リストアイテム内にカーソルを配置し、メニューオプションから 挿入 | 箇条書きを挿入 を選択します。その後上記の操作を再度行なってください。

メモ　リストを挿入する箇所にカーソルが配置された状態で、書式 ツールバーにあるリスト（記号）またはリスト（数字）アイコンをクリックすることでも、リストを作成することができます。カーソルの挿入位置に最初のリストアイテムが作成されます。

静的テキストをリストに変換する

以下の操作により、静的テキストをリストに変換することができます:

リストへ変換するテキストをハイライトして、メニューオプションから タグで囲む | リスト を選択します。目的のマーカーを選択した後に「OK」をクリックします。テキスト内に改行が含まれる場合、各行が個別のリストアイテムとして作成されます。テキストの一部だけがハイライトされている場合、そのテキストがリストアイテムとして作成されます。作成することのできるリストの数に制限は無く、必要に応じて Enter キーを押下することで新たなリストを作成することができます。 タグで囲む | リスト コマンドはコンテキストメニューからもアクセスすることができます。

関連項目

- 動的なリスト
- リスト

5.7.2 動的なリスト

動的リストにより、同じ名前を持った兄弟ノードのコンテンツをリストアイテムとして表示することができます。インスタンスの要素がリスト内のアイテムとして表示されます。以下ではそのメカニズムと操作方法について説明します。

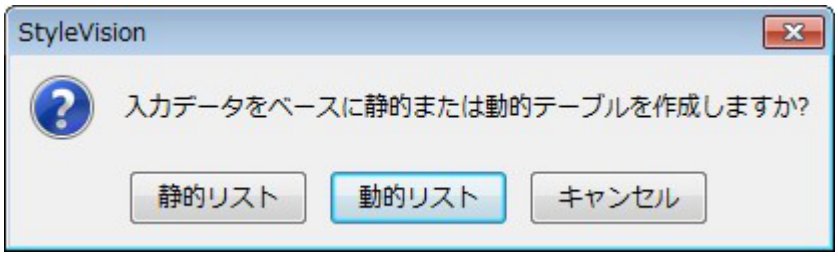
一般的なメカニズム

- 任意の要素をリストとして作成することができます。
- 要素がリストとして作成されると、その要素のインスタンスがリストのアイテムとして作成されます。例えば、department 要素内に複数の person 要素が含まれており、department 内にある全ての person からなるリストを作成する場合、person 要素をリストとして作成する必要があります。
- 要素に対してリストを作成した後に、テキスト、自動計算、動的コンテンツなどの静的または動的コンテンツを挿入することで、リストやリストアイテムの外観またはコンテンツを修正することができます。

動的なリストを作成する

以下の方法により動的リストを作成することができます:

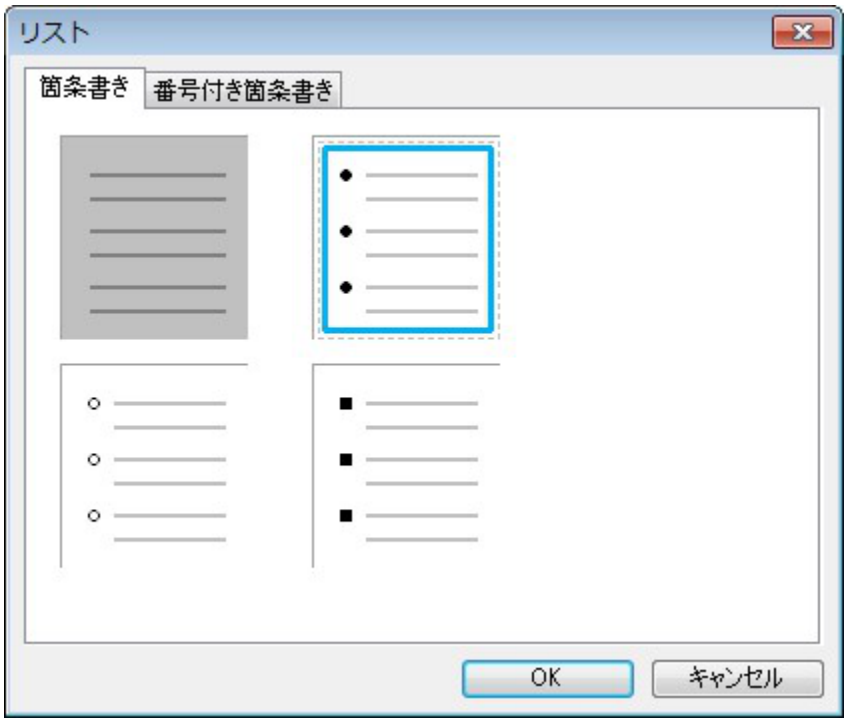
1. 動的リストを挿入する場所にカーソルを移動させ、メニューオプションから 挿入 | 箇条書きを挿入 を選択します。挿入するリストの種類(動的または静的リスト)を選択するダイアログが表示されます(以下のスクリーンショットを参照)。



2. 動的リストをクリックします。XPath セレクターダイアログが表示されます(以下のスクリーンショットを参照)。
3. XPath セレクターダイアログでは、XPath コンテキストが挿入ロケーションのコンテキストになっており、ダイアログ内では変更できない点に注目してください。動的リストとして作成されるノードを選択します。以下のスクリーンショットでは、コンテキストノードが n1:Department となっており、リストのアイテムとして作成されるノードに n1:Person が選択されています。.

ユーザー定義 XPath オプションを選択した場合、動的リストとして作成されるノードを選択する XPath 条件式を入力することができます。「OK」をクリックして、次のステップにて記述されている箇条書きダイアログを表示します。

4. 箇条書きダイアログにて、作成するリストの種類を選択します。箇条書きリストの種類(黒丸や白丸、黒四角)や、番号付き箇条書きを選択することができ、「OK」をクリックすることで、選択した種類のリストが作成されます。



関連項目

- 静的なリスト
- リスト

(C) 2015-2021 Altova GmbH

5.8 イメージ

SPS にてイメージを使用するには2種類の方法があります:

- デザインドキュメント内のイメージ
- Authentic View ツールバーアイコンを使って、XML ドキュメントへマークアップを適用する (カスタムツールバーボタン).

デザインドキュメントへイメージを挿入する場合、(SPS デザイナーにより) SPS 内部でイメージの場所を直接指定することができるほか、XML ドキュメント内にあるノードから取得することができます。イメージの挿入方法については、イメージ URI のセクションを参照ください。異なる出力形式にてサポートされているイメージの種類については、イメージ型 と出力セクションにて記述されます。 レファレンス | Autnetic メニュー | カスタムツールバーボタン のセクションでは、Authentic View で使用することができるツールバーアイコンの定義方法について記述されます。

イメージのプロパティ

イメージはプロパティウィンドウでセットすることができます。デザイン内でイメージを選択し、プロパティサイドバーにて (i) プロパティのカラムにてイメージを選択し、(ii) 目的のプロパティグループを選択し、(iii) 選択されたプロパティグループにある目的のプロパティを選択します。例えば、イメージの高さと幅をセットするには、HTML グループにある height と width プロパティに値をセットします。

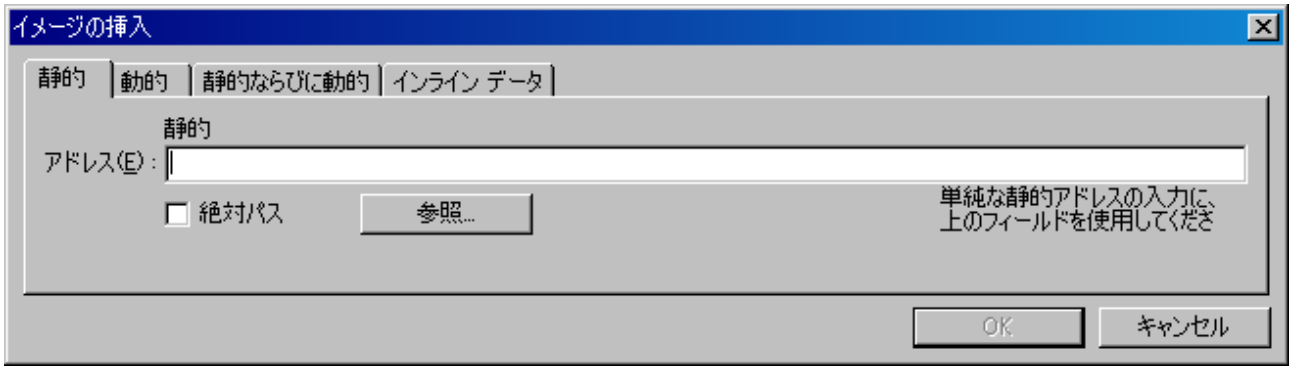
関連項目

- イメージの挿入
- 青写真イメージ

5.8.1 イメージ: URI とインラインデータ

イメージはデザインドキュメント内の任意の場所に挿入することができます。これらのイメージは Authentic View ならびに出力ドキュメントに表示され、デザインビューではイメージの代わりにサムネイルまたはプレースホルダが表示されます。

イメージを挿入するには 挿入 | イメージ メニューコマンドを選択します。イメージの挿入ダイアログが表示されます (以下のスクリーンショットを参照)。



イメージへは2種類の方法によりアクセスすることができます:

- イメージをファイルとして、イメージの挿入ダイアログにて URI を入力することでアクセスすることができます。

- Base-16 または Base-64 テキストとしてイメージをエンコードし、XML ファイル内に配置することができます。

イメージファイルを挿入する

URI を指定することで、イメージファイルがデザイン内に挿入されます。このファイルはランタイムにてアクセスされ、ドキュメント内部に配置されます。イメージの挿入ダイアログでは、3種類の方法でイメージのURI を入力することができます(上のスクリーンショットを参照)。

- 静的タブでは、絶対または相対 URI として URI が直接入力されます。例えば、相対 URI であれば nanonull.gif (以下のセクションを参照ください)となり、絶対 URI であればC:/images/nanonull.gif となります。
- 動的タブでは、(i) 絶対または相対 URI、または (ii) パースされたエンティティ名 を含むノードを選択するXPath 条件式を入力します。例えば、image/@location というエントリーにより、コンテキストノードの子である image 要素の location 属性が選択されます。XML ドキュメント内にある選択されたノードにはイメージ URI が含まれます。非解析対象実体の使用方法については、パースされていないエンティティ URI のセクションを参照ください。
- 静的ならびに動的タブでは、XPath 条件式による動的な箇所を、静的なエントリー(テキスト)により囲むことができます。例えば、静的なプレフィックスを C:/XYZCompany/Personnel/Photos/ として、動的なパートを concat(First, Last)、静的なサフィックスを .png とすることができます。この組み合わせにより、例えば C:/XYZCompany/Personnel/Photos/JohnDoe.png といった絶対 URI が選択されるようになります。

エンコードされたテキストのイメージを挿入する

イメージはBase-16 またはBase-64 でエンコードされたテキストとしてXML ファイル内部に保管することができます。この方法により、個別のファイルへアクセス(リンク)することなく、ソースXML ファイル内にあるイメージへアクセスすることができるようになります。XML ソース内部にてエンコードされたイメージを挿入するには、イメージの挿入ダイアログにあるインラインデータタブを使用します(以下のスクリーンショットを参照)。

XPath 条件式を使用することで、エンコードされたイメージのテキストを含むXML ドキュメント内のノードを指定します。イメージフォーマットコンボボックスから、生成するイメージ形式を選択します(イメージファイルはエンコードされたテキストデータから生成され、このファイルが出力ドキュメントにて使用されます)。エンコーディングコンボボックスでは、ソースXML ファイルで使用されているエンコーディングの形式を指定します。この操作により、エンコードされたテキストを(指定されたエンコード形式により) StyleVision にて正しく読み取ることが可能になります。

(**イメージファイル設定** ボタンからアクセスすることができる) イメージファイル設定ダイアログでは、作成されるイメージファイルに対して名前を与えることができます。名前を与えないことも可能で、その場合 StyleVision によりデフォルトの名前が生成されます。

出力からイメージへのアクセス

出力の形式と作業を行なっている段階により、イメージへのアクセス方法が変わります。以下の点に留意してください:

- お使いのエディションでご利用になれる出力フォーマットを確認してください: (i) Basic エディションでは HTML が、(ii) Professional エディションでは HTML と RTF が利用可能で、(iii) Enterprise エディションでは HTML、RTF、PDF、そして Word 2007+ が利用できます。
- デザインビューとStyleVision などの Altova 製品における Authentic View では、プロパティダイアログ にて、イメージに使用される相対パスを SPS またはXML ファイルに対して相対とするか指定することができます。

(C) 2015-2021 Altova GmbH

- HTML 出力の場合、イメージの URI が HTML ファイルへ渡されることでブラウザーはイメージにアクセスします。そのため、イメージへのパスが相対パスとなっている場合、HTML ファイルの場所から相対的になっている必要があります。StyleVision 内にある HTML プレビューでは、一時的な HTML ファイルが SPS ファイルと同じフォルダー内に作成されます。そのため HTML によるレンダリングを行うには、この場所から相対的となるパスを指定する必要があります。
- RTF 出力の場合、イメージの URI はオブジェクトリンクとして RTF に渡され、RTF ファイルが開かれた時にアプリケーション（通常 MS Word）によりアクセスが行われます。URI が相対パスとなっている場合、そのパスは RTF ファイルから相対的になっている必要があります。StyleVision にて RTF のプレビューを行う場合、SPS ファイルがあるフォルダーに一時的な RTF ファイルが作成されるため、レンダリングを行うにはこの場所から相対的となるパスを指定する必要があります。.
- URI が相対的か絶対的かにかかわらず、レンダリングを行うプロセスからイメージへ物理的にアクセスできる状態になっている必要があります。

イメージのプロパティの編集

イメージを変更するには、デザインビューにてイメージを右クリックして、コンテキストメニューから**URL の編集**を選択します。実体はイメージの挿入ダイアログと等しいイメージの編集ダイアログが表示され、必要な修正を行うことができるようになります。プロパティサイドバーに表示されているイメージグループの URL プロパティからもイメージの編集ダイアログへアクセスすることができます。プロパティのイメージグループには alt プロパティも含まれており、イメージの代替テキストを指定することができます。

イメージを削除する

イメージを削除するには、イメージが選択された状態で、Delete キーを押下します。

関連項目

- イメージ型 と出力
- パースされていないエンティティ URI

5.8.2 イメージ型 と出力

StyleVision にてサポートされているイメージと、StyleVision にてサポートされている出力形式を以下に示します。StyleVision のエディションによりサポートされる出力フォーマットのセットが異なる点に注意してください。Enterprise エディションでは HTML、Authentic、RTF、PDF、Word 2007+ がサポートされるのに対して、Professional エディションでは HTML と Authentic、RTF が、Basic エディションでは HTML がサポートされます。

イメージ型	Authentic	HTML	RTF	PDF	Word 2007+
JPEG	はい	はい	はい	はい	はい
GIF	はい	はい	はい	はい	はい
PNG	はい	はい	はい	はい	はい
BMP	はい	はい	はい	はい	はい
TIFF	はい*	はい*	はい	はい	はい
SVG	はい*	はい*	いいえ	はい	いいえ
JPEG XR	はい	はい	いいえ	いいえ	いいえ

＊下の点に留意してください

以下の点に注意してください:

- デザインビューでは、イメージの場所が静的 URL により与えられている時だけ(つまりSPS に直接入力されたときだけ)イメージが表示されます。
- TIFF ならびに SVN イメージを Authentic View ならびに HTML ビューで表示するには、Internet Explorer 9 以上が必要になります。
- RTF 出力では、TIFF イメージは埋め込みではなくリンクにより表示されるため、大きさの変更を行うことはできません。
- SVG ドキュメントは XML 形式で、SVG 名前空間内に収める必要があります。
- イメージファイルが見つからない場合、FOP によりエラーが返され、PDF は生成されません。
- PDF の生成に FOP が使用される場合、PNG イメージのレンダリングを行うのに JIMI イメージライブラリを FOP からアクセス可能な場所にインストールしておく必要があります。
- FOP による画像の処理に関する詳細については、FOP W eb サイト を参照ください。

サンプルファイル

サンプルファイルの `Images.sps` は以下の場所に収められています:

```
C:\Documents and Settings\<username>\My
      Documents\Altova\StyleVision2021\StyleVisionExamples/Tutorial/Images
```

RTF 出力におけるイメージの大きさ変更

RTF 出力におけるイメージの大きさ変更はJPG ならびにPNG ファイルでのみサポートされています。以下の点に留意してください:

- イメージの大きさ変更は、(XSLT 1.0 **ではなく**) XSLT 2.0 またはXSLT 3.0, を使ったデザインでのみサポートされます。
- イメージのheight ならびにwidth 属性はスタイルサイドバーの詳細グループでのみセットすることができます。プロパティサイドバーのHTML グループにあるheight ならびにwidth 属性は**使用されません**。
- 属性単位(px やcm、in など)だけがサポートされます。パーセント値はサポートされません。
- JPG ならびにPNG イメージはRTF ファイル内に埋め込まれます。この埋め込みはAltova XSLT 2.0 拡張関数により実装されています。

RTF におけるイメージの埋め込み

RTF 出力では、イメージはリンク、またはXSLT 2.0 またはXSLT 3.0 を使用して埋め込みにより表示されます。この設定は各 SPS に対して個別に行われます。以下の方法によりイメージを埋め込むことができます:

1. 目的の SPS がアクティブになっている状態で、(「**ファイル | プロパティ**」メニューオプションから)プロパティダイアログを開きます。
2. イメージタブにて、イメージを埋め込みチェックボックスにチェックを入れます(デフォルトの設定でもチェックされています)。アクティブな SPS では、XSLT のバージョンとして XSLT 2.0 または XSLT 3.0 がセットされていなければなりません。
3. 「**OK**」をクリックして SPS を保存します。アクティブな SPS に対して設定が保存されます。

同様の設定を他のSPS に対して行うには、そのSPS をアクティブにした状態で、同じ操作を行なってください。

イメージを埋め込みチェックボックスにチェックが入っていない場合、(イメージを選択した状態でプロパティサイドバーのURL から選択できる)イメージプロパティにて指定されたイメージファイルへのパスに従いイメージへのリンクが作成されます。パスがどのように解決されるかについては次のセクションを参照してください: イメージURI.

メモ RTF フォーマットでは EMF、JPG、PNG ファイルだけが埋め込みイメージとしてサポートされます。

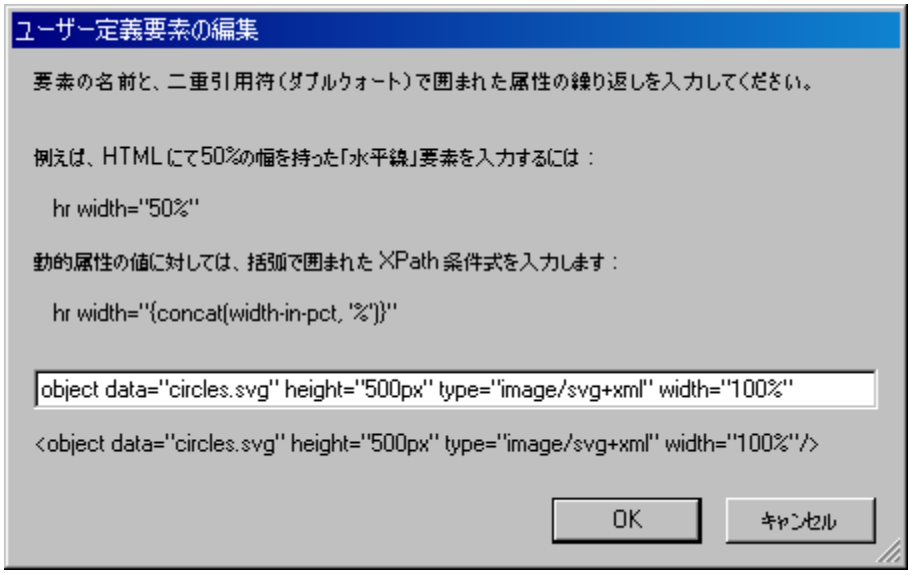
(C) 2015-2021 Altova GmbH

HTML における SVG

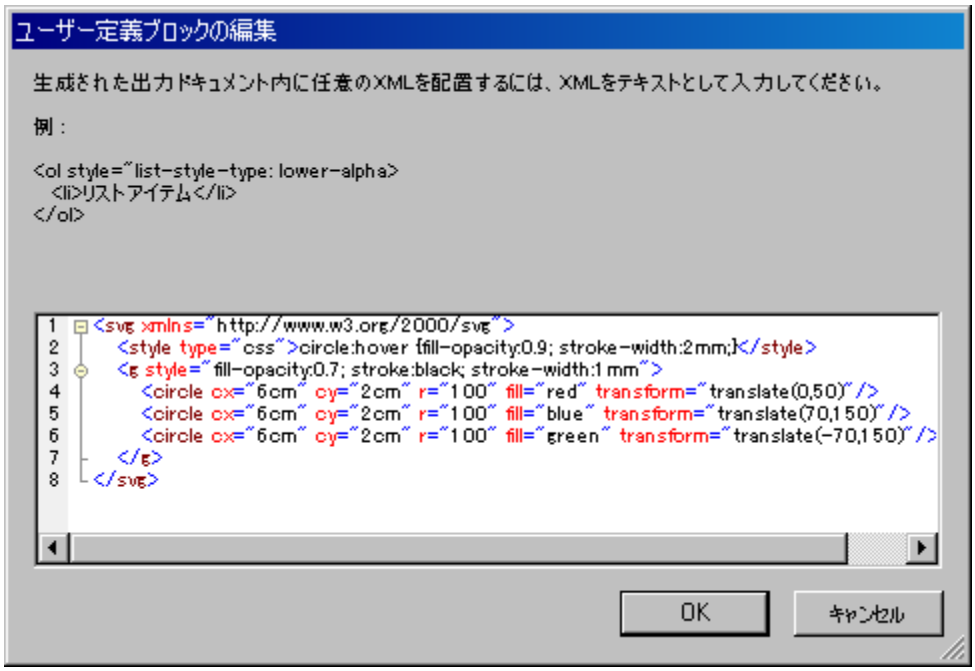
マウスイベントに対するコードが含まれる外部 SVG ファイルがイメージとして使用された場合、イメージ内部で SVG のレンダリングが行われ、インタラクティブ性が失われます。この制約は、オブジェクトとして外部 SVG イメージを扱うか、ユーザー定義された XML ブロックとして SVG コードフラグメントを追加することにより回避することができます。.

SVG をウェブページに含めるには、以下に示される3種類の方法があります。

1. イメージとして挿入された外部 SVG：生成された HTML ファイルにて <img> タグを生成します。インタラクティブ性は失われます。
2. ユーザー定義要素機能によりオブジェクトとして挿入された外部 SVG（以下のスクリーンショットを参照）：type 属性を正しく挿入するようにしてください（例: type="image/svg+xml"）。このように挿入することで、SVG オブジェクトのインタラクティブ性が保たれ、マウスオーバー機能も有効になります。



3. ユーザー定義 XML ブロックにより挿入されたインライン SVG: SVG コードフラグメントの例を以下のスクリーンショットに示します。このように挿入を行うことでインタラクティブ性が保たれます。出力方法が HTML 4.0 または 5.0 の場合、svg 要素を SVG 名前空間に含める必要はありませんが、XHTML を使用する場合、名前空間が必要になります。



関連項目

- イメージ URI

5.8.3 サンプル：イメージのテンプレート

StyleVision パッケージにはStyleVision に搭載されているイメージ機能のデモするためのSPS ファイルが含まれています。ファイルは(マイ ドキュメントフォルダー: `C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\Images\Images.sps` です。Images ドキュメント（Images.xml とImages.sps）は、3つのパーツから構成されます:

- 最初のパートでは Authentic View にてテキスト状態アイコンを使用する方法について記述されます（Enterprise ならびに Professional エディションでのみ作成できます）。XMLSpy の Authentic View 、Authentic Desktop、またはAuthentic Browser にて Images.xml ファイルを開いて、テキスト状態アイコンがどのように使用されるのか確認することができます。テキスト状態アイコンは StyleVision の Authentic プレビューでは確認できず、この機能を StyleVision で確認することはできない点に注意してください。テキスト状態アイコンの作成方法については、このユーザーマニュアルの レファレンス| Autnetic メニュー| カスタム ツールバーボタン のセクションを参照してください。
- 2番目のパートには、StyleVision 出力フォーマットごとにサポートされているイメージフォーマットを表したテーブルが含まれています。デザインビューでは静的 URI をともなうイメージだけが表示されます。このテーブルにあるイメージフォーマットは全て、3番目のパートにて表示されます。
- 3番目のパートでは、StyleVision にてサポートされる全てのイメージが表示されます。Images.sps ファイルを StyleVision にて開いた場合、StyleVision で利用できるプレビューを切り替えることで、どのイメージがそれぞれのプレビューで表示されるのか確認することができます。イメージの場所は XML ノード内に含まれているので、独自のイメージの場所を Authentic View にて入力し、プレビューウィンドウでどのように表示されるか確認することができます。

(C) 2015-2021 Altova GmbH

関連項目

- イメージ URI
- イメージ型 と出力
- カスタム ツール バーボタン

5.9 フォームコントロール

XML ドキュメント内にあるノードは、（入力フィールドやコンボボックスといった）データ入力デバイスとして作成することができます。ドキュメント内にあるノードは、（入力フィールドやコンボボックスといった）データ入力デバイスとして作成することができます。データ入力デバイスは Authentic View における編集を容易にするために用意されています。例えば、入力フィールドを使用することで指定された場所にて Authentic View からのデータ入力を促すことができ、コンボボックスを使用することでユーザーが入力することのできる値の表示や制限を行うことができます。データ入力デバイスにデータが入力されると、要素コンテンツや属性の値として XML ドキュメントにデータが挿入されます。 HTML と RTF 出力では、Authentic View における表示に類似したかたちで、データ入力デバイスがオブジェクトとしてレンダリングされます。データ入力デバイスは XML ドキュメントへの入力しか受け付けておらず、HTML 出力では動作しない点に注意してください。

一般的なメカニズム

StyleVision で使用することのできるデータ入力デバイスのリストを、XML ドキュメントにてどのようにデータが入力されるかの説明とともに、以下に示します。.

データ入力デバイス	XML ファイル内のデータ
入力フィールド（テキストボックス）	ユーザーにより入力されたテキスト
複数行 入力フィールド	ユーザーにより入力されたテキスト
コンボボックス	ユーザーの選択を値へマッピング
チェックボックス	ユーザーの選択を値へマッピング
ラジオボタン	ユーザーの選択を値へマッピング
ボタン	ユーザーの選択を値へマッピング

入力フィールドに入力された値は、XML コンテンツとして XML ドキュメント内に直接入力されます。その他のデータ入力デバイスでは、Authentic View ユーザーの選択が値へマッピングされます。StyleVision では、ユーザーが選択することのできるオプションのリストと、各オプションに対応した値を定義することができます。通常、ダイアログにてオプションと、対応する値を定義することになります。

- 入力フィールド（テキストボックス）
- 複数行 入力フィールド
- コンボボックス
- チェックボックス
- ラジオボタン
- ボタン

一般的な使用方法

以下の操作により、データ入力デバイスを作成することができます:

1. スキーマツリーサイドバーからデザインビューにある目的の場所へノードをドラッグします。
2. コンテキストメニューから、ノードに対して作成するデータ入力デバイスを選択します。
3. データ入力デバイスによってはダイアログが表示されます。ダイアログが表示される場合、必要な情報をダイアログに入力し、「OK」をクリックします。

データ入力デバイスのプロパティを**再度開いて編集**するには、（それを含んでいるノードではなく）そのデータ入力デバイスをクリックして、プロパティサイドバーに表示されているプロパティを編集します。

メモ

- データは、Authentic View 内のデータ入力デバイスのみで作成することができます。。

(C) 2015-2021 Altova GmbH

- 既にあるノードのコンポーネントをデータ入力デバイスへ変更することでも、データ入力デバイスを作成することができます。ノードを右クリックして、**変更**を選択してください。
- HTML とRTF 出力では、ユーザーにより選択されたエントリーが表示されます。HTML ドキュメントに表示されたデータ入力デバイスの値を変更しても、XML ドキュメントやHTML ドキュメント内にある値は変更されません。
- チェックボックスのような、印刷媒体において正しくレンダリングされないデータ入力デバイスは、代替レンダリングが行われます。

関連項目

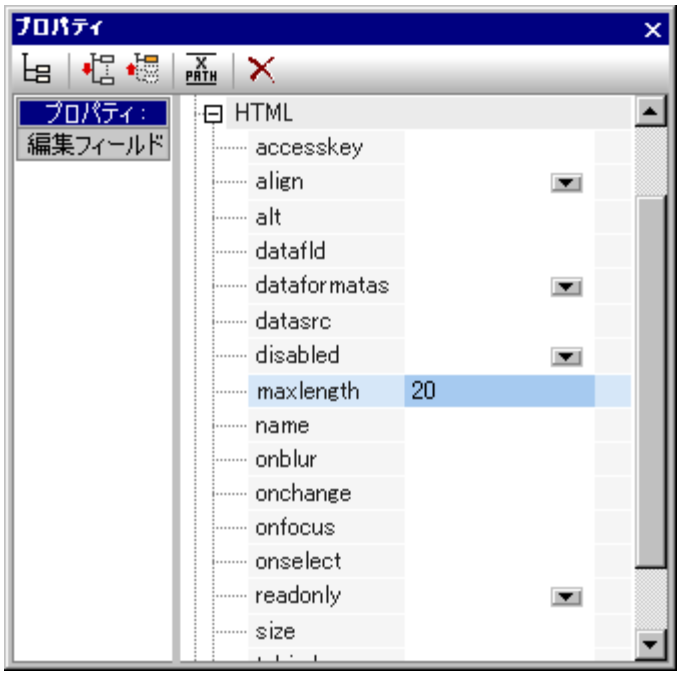
- 入力フィールド、複数行 入力フィールド
- コンボボックス
- チェックボックス
- ラジオボタン、ボタン

5.9.1 入力フィールド、複数行の入力フィールド

スキーマソースウィンドウからデザインビューへノードをドロップすることで、SPS 内に入力フィールドや複数行の入力フィールドを挿入することができます。Authentic View ユーザーにより入力されたテキストが、そのフィールドに対して作成されたXML ノード内に入力されます そのノードのコンテンツが入力フィールドまたは複数行の入力フィールドにて表示されます。

入力フィールドのプロパティを編集する

入力フィールドを選択し、プロパティサイドバーのHTML プロパティを修正することで、入力フィールドのHTML プロパティを修正することができます (*下のスクリーンショット参照*)。



例えば、入力フィールドが選択された状態で、**編集フィールド**を選択し、プロパティのHTML グループ以下にある maxlength プロパティを選択します。maxlength の値フィールドをダブルクリックして、値を入力してください。

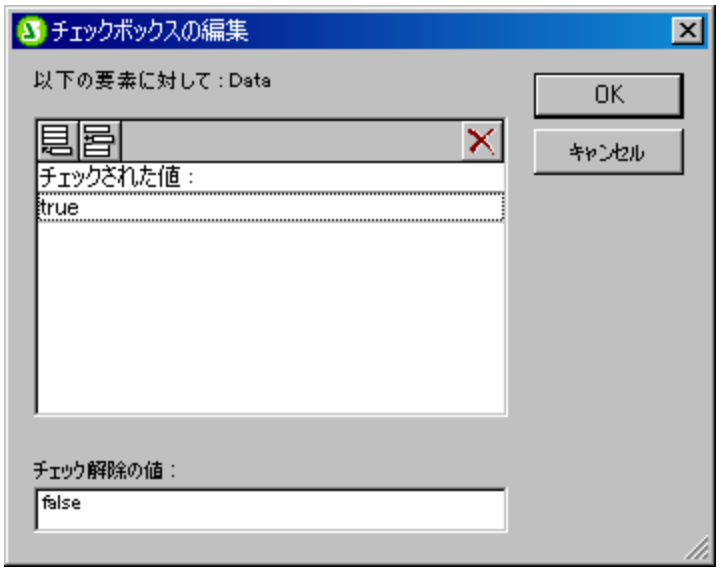
メモ CDATA セクションを入力フィールド（テキストボックスと複数行のテキストボックス）に挿入することはできません。CDATA セクションは、テキストコンテンツコンポーネントとして Authentic View に表示される要素内部だけに入力することができます。

関連項目

- コンボボックス
- チェックボックス
- ラジオボタン、ボタン

5.9.2 チェックボックス

チェックボックスをデータ入力デバイスとして作成することができます。この機能により、ユーザーからの入力を2種類に制限することができます。チェックボックスの編集ダイアログにて XML の値を指定することで、チェックされた時の値と、そうでない時の値をマッピングすることができます。



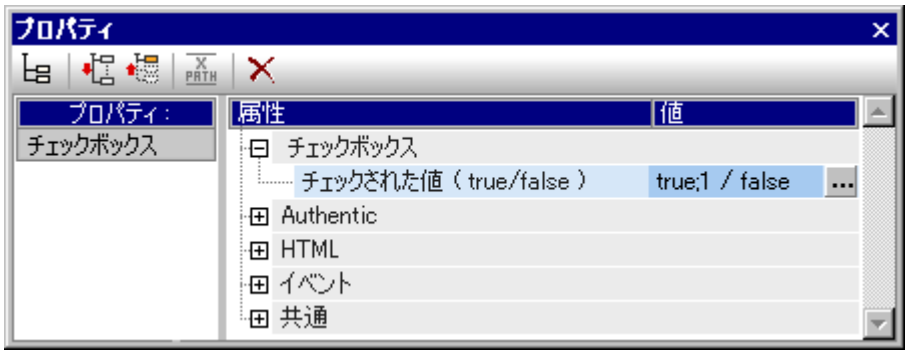
上のスクリーンショットでは、Name という名前の要素がチェックボックスとして作成されました。Authentic View ユーザーがチェックボックスにチェックを入れた場合、true という値が Name 要素の値として入力されます。チェックボックスのチェックが外された場合、（ダイアログにて定義されている通り）false という値が Name の XML 値として入力されます。

メモ 新たな Name（またはチェックボックス）要素が Authentic View にて作成された場合、その XML 値は（チェックされていない時の値ではなく）空となります。チェック解除の値は、チェックボックスが一度チェックされ、その後チェックが外されることで入力されます。ノードに対してデフォルトの値を与えるには、デフォルトの値を含むテンプレート XML ファイルを作成してください。

チェックボックスの編集ダイアログへアクセスする

ノードをチェックボックスとして作成して、新たなチェックボックス作成する時にはチェックボックスの編集ダイアログが表示されます。以下の操作により、チェックボックスの編集ダイアログへアクセスすることができます：

1. デザイン内にあるチェックボックスを選択します。
2. プロパティサイドバーにてチェックボックスアイテムを選択し、プロパティのチェックボックスグループを選択します（以下のスクリーンショットを参照）。



3. チェックされた値プロパティにある編集ボタン ... をクリックします。チェックボックスの編集ダイアログが表示されます。

メモ　チェックボックスが選択された状態でプロパティサイドバーにある HTML プロパティを修正することにより、チェックボックスの HTML プロパティを編集することができます。

関連項目

- 入力フィールド、複数行 入力フィールド
- コンボボックス
- ラジオボタン、ボタン

5.9.3 コンボボックス

コンボボックスを使用することで、Authentic View ユーザーに対してドロップダウンリスト形式のオプションリストを表示することができます。選択されたオプションにマッピングされている値がXML ドキュメントに入力されます。ドロップダウンリストのエントリーからXML 値へのマッピングは、SPS にて指定されます。

マッピングは、以下にあるどれからの方法で、コンボボックスの編集ダイアログにて行うことができます:

- 選択されたノードに対するスキーマの列挙。この場合、(ドロップダウンリストに) 表示されているエントリーがXML の値と同一になります。
- コンボボックスダイアログにて定義されたリストから。表示されるエントリーと、それに対応するXML の値を入力します。
- カレントノードに相対的な XPath 条件式の結果シーケンスから。結果シーケンス内にあるアイテムがドロップダウンリストのエントリーとして表示され、Authentic View ユーザーにより選択されたエントリーがノードの値として入力されます。この機能により、コンボボックスにて動的なエントリーを使用することができるようになります。ここではコンボボックスとして作成されるノードが重要な意味を持ちます。例えば、Namelist という名前の要素にName という要素を任意の数だけ含めることができ、各 Name 要素にはFirst と Last という子要素が含まれている場合を考えてみましょう。. Name 要素をコンボボックスとして作成し、Last 子要素をリストの値として選択した場合、Authentic View では、Name 要素の数だけコンボボックスが作成され、各ドロップダウンリストのエントリーにはLast の値が表示されます。1つのコンボボックスのドロップダウンリスト内に全てのLast 要素を表示するためには、Namelist 要素をコンボボックスとして作成し、XPath 条件式にて Last 要素を選択する必要があります。

コンボボックスの編集ダイアログへアクセスする

ノードをコンボボックスとして作成して、新たなコンボボックスを作成する時には、コンボボックスの編集ダイアログが表示されます。メニューオプションから「**挿入 | フォームコントロールの挿入 | コンボボックス**」を選択することでも、コンボボックスを挿入することができます。以下の操作によりコンボボックスの編集ダイアログへアクセスすることができます。

1. デザイン内にあるコンボボックスを選択します。
2. プロパティサイドバーにてコンボボックスを選択し、プロパティのコンボボックスグループを選択します（以下のスクリーンショットを参照）。

3. コンボボックス入力値プロパティにある編集ボタン ... をクリックします。コンボボックスの編集ダイアログが表示されます。

コンボボックスの編集 ダイアログを使用する

コンボボックスの編集ダイアログを以下に示します。

以下の操作により、コンボボックスのエントリーと値を定義することができます:

1. 対応するラジオボタンをクリックして、エントリーと値を定義する方法を以下から選択します：(i) スキーマの列挙、(ii) 値のリスト、(iii) XPath 条件式を使った値の選択。
2. **スキーマの列挙**を選択した場合、スキーマ内でそのノードに割り当てられている列挙が自動的に、(i) コンボボックスにあるドロップダウンリストのエントリーとして、また (ii) 対応する XML の値として使用されます（以下のスクリーンショットを参照）。ドロップダウンリストの各エントリーには対応する XML 値が割り当てられています。Authentic ユーザーが選択するエントリーに対応した XML の値が XML ファイルに入力されます。エントリーと XML の値は両方ともスキーマの列挙から得られるため、グレーアウトされて編集できないようになっています。

値のリストを選択した場合、コンボボックスのドロップダウンリストに表示されるエントリーと、対応する XML の値を任意の数だけ挿入、追加、編集、削除することができます。これらの編集作業は**値のリストを使用**ラジオボタン以下にあるペインで行うことができます。XPath 条件式を使用することで、表示されるエントリーと XML の値を作成することもできます。XPath 条件式により返されたシーケンス内にあるアイテムが、表示されるエントリーと XML の値として使用されます。ダイアログでは、(i) 同一の XPath 条件式によりエントリーと XML 値を作成する、または (ii) エントリーと XML 値の作成に異なる XPath 条件式を使用することができます。異なる XPath 条件式を使用する場合、インデックスにより2つのシーケンス内にあるアイテムが、表示されるエントリーと XML の値としてマッピングされます。2つのシーケンスに含まれるアイテムの数が異なる場合、エラーが返されます。

3. Authentic View にて表示されるコンボボックスのドロップダウンリストのアイテムをソートするには、**Authentic 内の値を並び替え**チェックボックスにチェックを入れてください。
4. 「OK」をクリックして完了します。

メモ

- XPath 条件式を使用することで、コンボボックスのリスト内に表示されるアイテムを選択し、XML ファイルからの動的なエントリーをコンボボックスに表示することができます。
- コンボボックスのドロップダウンリスト内に表示されるアイテムがスキーマの列挙から得られた場合、デフォルトでアルファベット順に並べ替えられます。XML データファイルからアイテムが得られた場合、デフォルトでドキュメント内にある順序どおりに表示されます。アイテムを DB から得る場合、DB スキーマをメインスキーマとしてセットする必要があります。メインスキーマではない DB からアイテムが得られた場合、XPath 条件式により目的の DB 行を選択するテンプレートを(たとえテンプレートが空があったとしても)デザイン内に含める必要があります。このような場合は更に、目的の行のインスタンスが[すべて取得されている]ようにする必要があります。

関連項目

- 条件を使用する
- 入力フィールド、複数行 入力フィールド
- チェックボックス
- ラジオボタン、ボタン

5.9.4 ラジオボタン、ボタン

ボタンには、通常のボタンとラジオボタンの2種類があります。ラジオボタンにより Authentic View ユーザーからの入力を XML ファイルへ渡すことができます。ボタンにより Authentic View のデータ入力を**行うことはできません**が、HTML 出力にてイベントを開始するときに使用することができます。

ラジオボタン

SPS にラジオボタンを挿入することで、複数のオプションから選択を行うことが可能になります。挿入された各ラジオボタンは、それぞれ XML の値に対応しています。ユーザーが同時に選択することのできるラジオボタンは1つだけで、関連付けられた XML の値がシードの値として入力

されます。各ラジオボタンに対して、(i) ユーザーに対して表示される静的なテキストと、(ii) 各ラジオボタンに対応したXML の値をセットする必要があります。ラジオボタンのXML 値を編集するには、プロパティサイドバーにてラジオボタンアイテムを選択し、ラジオボタンプロパティにある編集ボタンをクリックしてください。ラジオボタンの編集ダイアログが表示されます。

ボタン
ボタンオプションによりボタンが挿入され、ボタン上に表示されるテキストを指定することができます。この機能により、生成されたHTML 出力にてボタンイベントとスクリプトを関連付けることができます。ボタンがXML の値へマッピングされることはなく、Authentic View におけるデータ入力には使用できない点に注意してください。それでも、ボタンの親ノードにある値をAuthentic View ユーザーが選択できるようなデザインを作成することはでき、そのAuthentic View ユーザーが選択した内容により、表示が変更されるようにSPS をデザインすることもできます。

メモ　ラジオボタンやボタンが選択された状態で、プロパティサイドバーにある HTML プロパティを修正することで、ラジオボタンやボタンの HTML プロパティを編集することができます。

関連項目

- 入力フィールド、複数行 入力フィールド
- コンボボックス
- チェックボックス

5.10 リンク

リンク(またはハイパーリンク)を、ドキュメント内のブックマークやウェブページといった外部リソースに対して作成することができます。リンクは動的に生成されたアンカーに対しても作成することができます。StyleVision では、ハイパーリンクに使用される URI を非常に柔軟な方法で作成することができます。
ブックマークとハイパーリンク のセクションでは、静的ならびに動的なブックマークをドキュメント内に作成し、ブックマークや外部ドキュメントへのリンクを作成する方法について記述されます。

関連項目

- ブックマークとハイパーリンク

(C) 2015-2021 Altova GmbH

5.11 バーコード

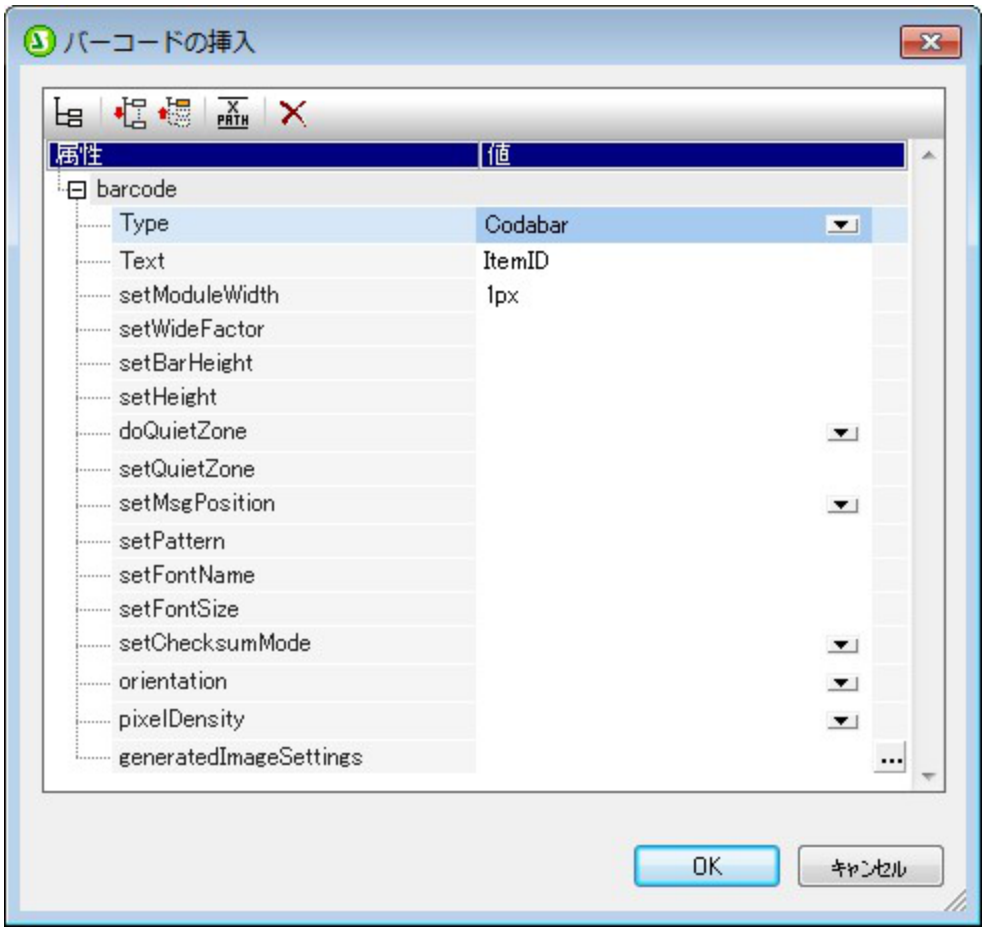
バーコード, バーコードデザイン要素を使用することで、出力ドキュメント内にバーコードを生成することができます（以下のスクリーンショットを参照）。デザインドキュメント内のバーコードを挿入する場所にバーコードデザイン要素を挿入し プロパティを指定します。バーコードデザイン要素はXSLT 2.0 またはXSLT 3.0（ XSLT 1.0 ではない）モードでのみサポートされるということご注意してください。

重要: バーコードを動作させるためには、Java Runtime Environment（JRE） がインストールされている必要があります。JRE はバージョン 1.4 以降で、お使いのシステムにインストールされた StyleVision パッケージのビットバージョン（32-ビットまたは64-ビット）と同一のものでなければなりません。

バーコードを挿入する

以下の方法により、デザイン内部にバーコードを挿入することができます:

1. バーコードを挿入する場所にて右クリックして、**バーコードの挿入**コマンドを選択します。その他にもメニューオプションから「**挿入 | バーコードの挿入**」を選択するか、ツールバーのバーコードアイコンをクリックして、デザインにてバーコードを挿入する場所をクリックします。スキーマツリーからデザインビューへ要素をドラッグアンドドロップして、**バーコードの作成**を選択することもできます。バーコードの挿入ダイアログが表示されます（以下のスクリーンショットを参照）。



2. Type ならびに Text プロパティは必須プロパティとなっている一方、それ以外のプロパティはオプション、または適切なデフォルト値が与えられています。Type プロパティの値はドロップダウンリストから選択することができ、ISBN バーコードを含む EAN-13 や UPC-A といったバーコードの種類を選択することができます。バーコードプロパティについては 下で説明されます 。必須プロパティの値と、目的に応じたオプションのプロパティをセットしてください。XML ファイル内にある値をプロパティの値として使用する場合、XPath 式を入力 することで、アクセスする XML ノードを特定することができます。XPath 条件式を使用するには、目的のプロパティを選択し、プロパティダイアログのツールバーにある XPath ボタンを有効にして、XPath 式 の編集ダイアログにて XPath 条件式を入力します。XPath 条件式はカレントノード内で評価されます。
3. プロパティの設定が完了したら、「OK」をクリックします。バーコードイメージが挿入され、生成されたバーコードを出力プレビューにて確認することができるようになります（以下のスクリーンショットを参照）。

メモ バーコードのイメージは PNG ファイルとして生成されます。

バーコードのプロパティ

以下にあるバーコードのプロパティを指定することができます。Text ならびに Type プロパティは必須となっており、それ以外のプロパティはオプションとなります。バーコードの種類により、利用することのできるプロパティが異なるということに留意してください。

- *Type:* 与えられたテキストを読み取るためのバーコードシステムです（例: EAN または UPC）。

(C) 2015-2021 Altova GmbH

- *Text:* バーコードパターンの生成に使用される値です。
- *SetModuleWidth* : コード内にあるバーの幅を指定します。
- *SetBarHeight:* バーの高さを指定します。
- *SetHeight:* バーコードのイメージの高さを指定します。
- *DoQuietZone:* Yes または No という値により、*SetQuietZone* と *SetVerticalQuietZone* プロパティにより大きさが指定される余白を表示するかが決定されます。
- *SetQuietZone:* バーコードの周りに「クワイエットゾーン」（余白）が表示されるようにします。1次元のバーコードの場合、ここで指定された値が水平方向に対して適用されます。2次元のバーコードの場合、水平ならびに垂直方向に対して値が適用されます。垂直方向の値は、*SetVerticalQuietZone* プロパティに値を指定することで、上書きすることもできます。
- *SetVerticalQuietZone:* 2次元のバーコードに対して、垂直方向の「クワイエットゾーン」（余白）を指定することができます。長さの単位であるミリメートル (mm) を指定する必要があります（例: 2mm）。
- *SetMsgPosition:* バーコードから見て、どこにメッセージテキストが表示されるか指定します。top、bottom、または none （メッセージが生成されません）を指定することができます。
- *SetPattern:* 数字だけの長い文字列といったメッセージテキストを、読みやすくするためのパターンをセットします。パターンの構文は以下に示されます。
- *SetFontName:* テキストを表示するためのフォントを指定します。
- *SetFontSize:* テキストに使用されるフォントサイズを指定します。
- *SetChecksumMode:* 以下の値を指定することができます: (i) *Add* : メッセージにチェックサムが追加されます、(ii) *Check* : バーコードのレンダリング中にチェックサムのチェックが行われます（チェックサムが存在する場合）、(iii) *Ignore* : チェックサムの処理が行われることはありません、(iv) *Auto* : 選択された種類のバーコードにて定義されたデフォルトの振る舞いが有効になります。
- *Orientation:* バーコードの回転を指定することができます。反時計回り方向に90度ごとの回転が行われます。
- *PixelDensity:* バーコードイメージにおけるピクセルの密度（解像度）が指定されます。密度が増えることにより、より鮮明なイメージが生成されます。
- *GeneratedImageSettings:* 生成されたバーコードのイメージファイル名が指定されます。名前が指定されなかった場合、StyleVision により自動的に名前が生成されます。

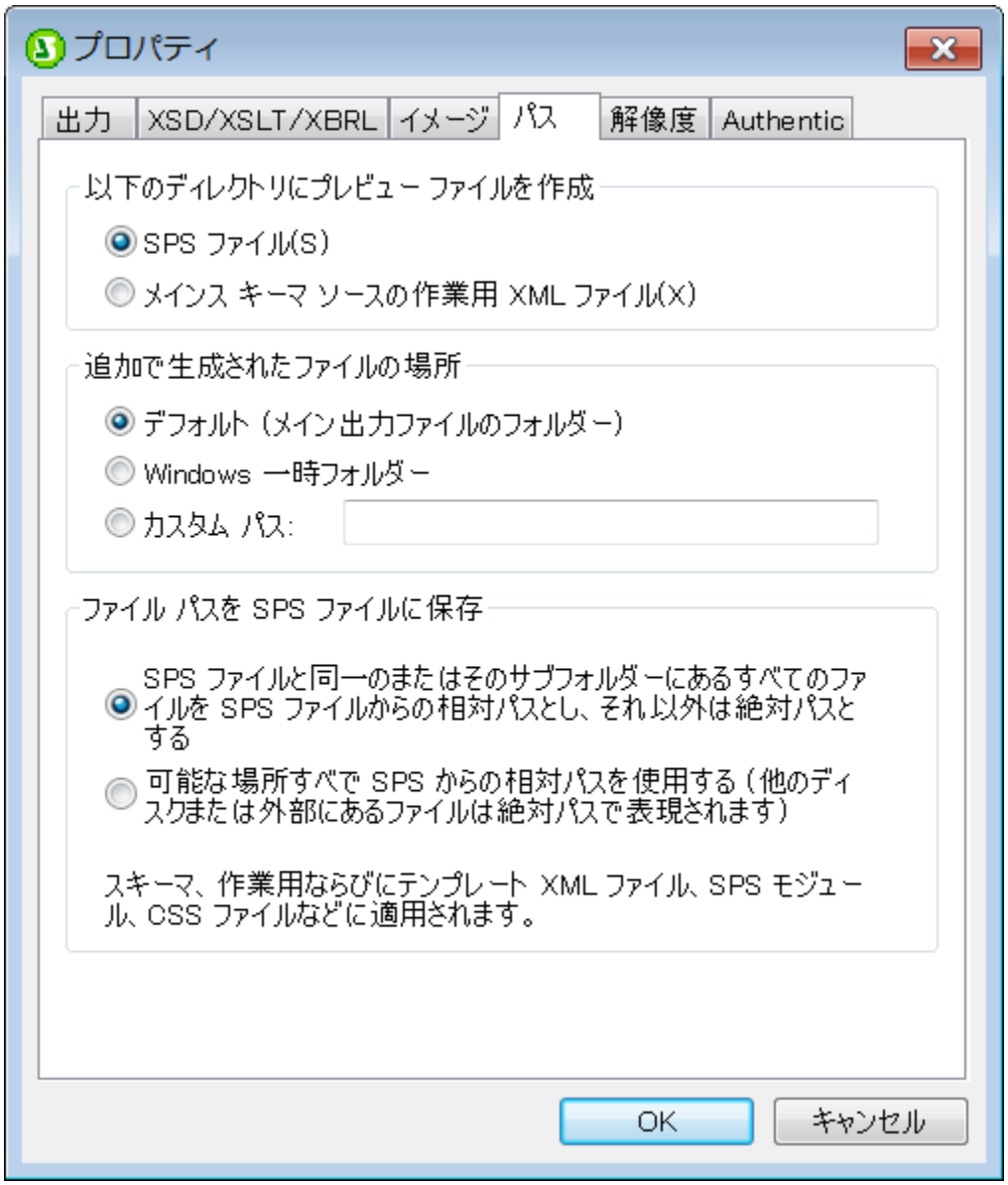
パターンの構文

パターンにより、バーコードに含まれる入力メッセージの文字列をより読みやすいかたちで出力することができます。パターンでは、入力メッセージのテキストがアンダースコアの "_" により示されます。パターン内にあるその他の文字は、出力メッセージテキストの対応する箇所に挿入されます。バックスラッシュ "¥" はエスケープシンボルとなります。例えば "¥?" により "?" という文字が出力メッセージテキストに表示されます（ここでの "?" は全ての文字を表します）。また、"#" 文字により、オリジナルのテキストに含まれていた文字を削除することができます。以下にあるパターンの構文例を参照ください。

入力メッセージテキスト	パターン	出力メッセージテキスト
123456	__ __ __	12 34 56
15032011094655	________ __:__:__ UTC	15\03\2011 09:46:55 UTC
15-03-2011	__#/__#/____	15/03/2011

出力ファイルを生成する

出力にて生成されるバーコードイメージのファイルは、プロパティダイアログのパスタブにて指定された場所に保存されます。プロパティダイアログへは、メニューオプションから「ファイル | プロパティ」を選択することでアクセスすることができます（以下のスクリーンショットを参照）。



プレビューで使用されるバーコードのイメージファイルは、SPS ファイルまたは作業用 XML ファイルが収められているディレクトリに作成されます。これらのファイルは一時的なファイルとして作成され、SPS が閉じられる際に削除されます。メニューオプションの「**ファイル | 生成されたファイルの保存**」により出力が生成された際には任意の場所にバーコードイメージファイルを作成することができます。パスタブにある「追加で生成されたファイルの場所」ペインにて場所を指定することができます(上のスクリーンショットを参照)。

関連項目

- SPS ファイル コンテンツ

(C) 2015-2021 Altova GmbH

5.12 レイアウト モジュール

レイアウト モジュールとは、レイアウトを含むオブジェクトのことです。モジュール全体を SPS デザイン内に挿入することにより、ドキュメント フローにおけるブロックとして作成することができます。デザインの要件に従って、SPS デザイン要素を含む複数のレイアウト ボックスをレイアウト モジュール内に配置することもできます。レイアウト モジュールを使用することで、グラフィカルなデザインアプリケーションと同じような操作で、レイアウトを作成することができます。

レイアウト モジュールを作成するためのステップを以下に示します:

1. レイアウト コンテナーを挿入するレイアウト コンテナーはページ上の幅全体を占めることも、大きさを指定することもできます。デザインの青写真を含めることでデザインガイドとして使用し、(サイド バーにある) スタイルを使用することでレイアウト コンテナーのフォーマットを行うこともできます。
2. レイアウトコンテナー内に1つ以上の レイアウト ボックス を挿入するレイアウト ボックスには複数のデザイン要素(スキーマテキスト、スキーマノード、自動計算、イメージ、リストなど)を含めることができ、(スタイルサイド バーにある) スタイルを使用することで、レイアウト ボックスのフォーマットを行うことができます。レイアウト コンテナー内にあるレイアウト ボックスは、他のレイアウト ボックスに対して相対的に位置を変更し、前面または背面へ移動することができます。
3. ライン の描画やフォーマット、また(レイアウト ボックスやその他のラインといった) レイアウト オブジェクト の前面や背面に移動することができます。

フォームベースのデザイン

新規 SPS の作成 する場合、フリーハンドのデザインか、フォームを基にしたデザインかを選択することができます。フォームを基にしたデザインとは、レイアウト コンテナーにより構成される SPS デザインのことです。

メモ　レイアウトモジュールは、Enterprise エディションの Altova 製品における Authentic View でしかサポートされません。

関連項目

- デザインの作成
- SPS ファイル コンテンツ

5.12.1 レイアウトコンテナー

レイアウト コンテナーは以下のような特徴を備えています:

- ドキュメント内、つまりテンプレート内部に 挿入 することができます。また、ドキュメント デザインを含んでいるコンテナー内に挿入することもできます。
- そのセクションに対して定義されたページの寸法に合わせる(レイアウトコンテナーの「用紙サイズに自動調整」プロパティ)ことができるほか、寸法を指定することができます。詳細については、レイアウト コンテナーのサイズ セクション参照ください。
- レイアウト グリッド と ズーム機能 により、レイアウト コンテナー内にあるオブジェクト の位置を簡単に指定することができます。
- 境界線、背景色、フォントのプロパティといった スタイル プロパティ を、全てのコンテナーに対して指定することができます。
- レイアウト ボックスやラインを含める ことはできますが、その他のデザイン要素を含めることはできません(全てのデザイン要素はレイアウト ボックス内部に配置する必要があります)。
- 青写真 を含めることで、テンプレートを作成するためのガイドとして使用することができます。青写真により表示されたイメージに、正確にマッチするようにデザインの配置を行うことができます。

メモ レイアウトコンテナーは、Enterprise エディションの Altova 製品における Authentic View でしかサポートされません。

レイアウトコンテナーを挿入する

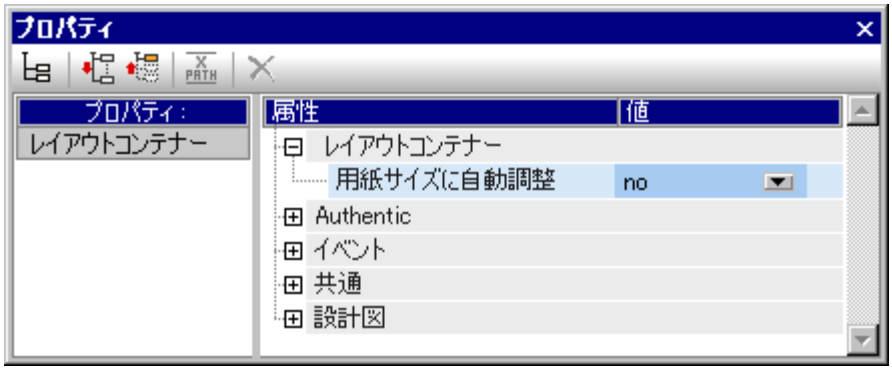
レイアウトコンテナーを挿入するには、デザイン要素の挿入 ツールバーにある**レイアウトコンテナーの挿入** アイコンをクリックした後に、レイアウトコンテナーを挿入する位置をクリックします。レイアウトコンテナーを用紙サイズに自動調整するかを決定するためのダイアログが表示されます。**はい**をクリックすることで、そのドキュメントセクションのページレイアウトプロパティにて定義されたページの寸法と同じ大きさがレイアウトコンテナーに対して与えられます。**いいえ**をクリックした場合、デフォルトサイズの3.5in x 5.0in のレイアウトコンテナーが作成されます。

レイアウトコンテナーは、SPS の作成時にも作成することができます。

レイアウトコンテナーのサイズ

レイアウトコンテナーの大きさは、以下の2つのプロパティにより決定されます:

- 用紙サイズに自動調整プロパティ(プロパティサイドバー、以下のスクリーンショットを参照): yes を選択することで、ドキュメントセクション内にあるページに対して使用される寸法がレイアウトコンテナーでも使用されます。no を選択すると、大きさをカスタマイズすることができるレイアウトコンテナーが作成されます。



- スタイルサイドバーのレイアウトから選択することができる詳細グループの height ならびに width プロパティにより、レイアウトコンテナーの大きさを指定することができます。レイアウトコンテナーの右ならびに下余白をドラッグすることでも、コンテナーの大きさを直接修正することができます。height ならびに width プロパティの値は、用紙サイズに自動調整プロパティの値が no になっている時にだけ有効になる点に注意してください。

レイアウトコンテナー グリッド

レイアウトコンテナー内にグリッドを表示させることで、アイテム間の間隔指定に使用することができます。以下の設定により、グリッドを使用することができます:

- *グリッドを表示/隠す*: デザイン要素の挿入ツールバーにて、グリッドの表示を有効/無効にすることができます。
- *グリッドサイズ*: オプションダイアログのデザインタブ (ツール| オプション) にて、水平ならびに垂直方向の長さを指定することができます。非常に大きな値が入力された場合、グリッドが正確に表示されない場合もあることに注意してください。
- *グリッドへのスナップ*: デザイン要素の挿入ツールバーにある切り替えコマンドにより、グリッドへのスナップ機能を有効または無効にすることができます。グリッドへのスナップ機能が有効になっている場合、レイアウトボックスの左上と、レイアウトボックスの終点がグリッドの線と点に対して配置されます。

ズーム機能

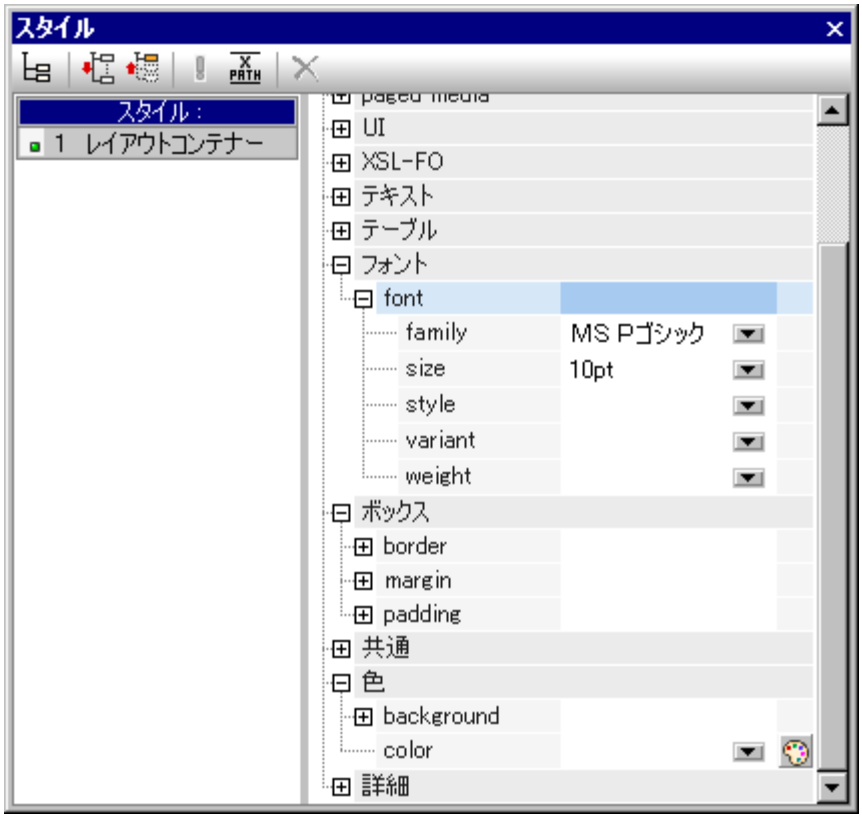
表示を拡大することで、オブジェクトの配置をより正確に行うことができます。(標準ツールバーにある) ズームコンボボックスを使用するか、Ctrl キーを押下しながらマウスホイールを使用することで、拡大率を変更することができます。

(C) 2015-2021 Altova GmbH

レイアウトコンテナー スタイルプロパティ

レイアウトコンテナーに適用することができるスタイルには、以下の2種類があります:

- レイアウトコンテナーだけに適用され、承継されないもの（例: border や background-color プロパティ）。
- レイアウトコンテナー内にあるレイアウトボックスに継承されるもの（例: フォントプロパティ）。



レイアウトコンテナーのスタイルプロパティは、スタイルサイドバーにあるレイアウトコンテナーにてセットされます（上のスクリーンショットを参照）。

レイアウトコンテナーのコンテンツ

レイアウトコンテナー内に含めることができるデザインアイテムは、レイアウトボックスとラインに限られます。その他に、（デザイン要素ではない）青写真をデザインのガイドとしてレイアウトコンテナー内に配置することができます。その他のデザイン要素は全て、レイアウトボックス内に配置する必要があります。

青写真

SPS の作成中に、デザインのガイドとしてレイアウトコンテナー内に青写真を配置することができます。青写真には、レイアウトコンテナーの大きさに等しい大きさのイメージファイルが使用されます。青写真イメージがレイアウトコンテナーよりも小さい場合、デザイン内で指定された位置をオフセットとすることができます（以下のスクリーンショットにある青写真グループプロパティを参照ください）。青写真を使用することで、SPS デザインを青写真デザインで表現することもできます。この機能により、青写真で表示されているとおりにデザイン要素を配置することが可能になります。青写真はデザインビューだけに表示され、出力ビューで表示されることはありません。

（プロパティサイドバーの）レイアウトコンテナープロパティにある青写真のプロパティグループから、青写真のプロパティへアクセスすることができます（以下のスクリーンショットを参照）。

レイアウトコンテナーに表示される青写真の透明度を指定することで、デザインの邪魔にならないようにすることができます。青写真のイメージは、**イメージを表示**プロパティにて、必要に応じて表示/非表示を切り替えることができます。

関連項目

- レイアウトボックス
- ライン

5.12.2 レイアウトボックス

レイアウト内にある全てのデザイン要素(静的テキストやスキーマノード、自動計算、イメージ、リストなど)は、レイアウトボックス内に配置する必要があります。デザイン要素を含んだレイアウトボックスは、必要に応じてレイアウトコンテナーに配置されます。デザイン要素をレイアウトコンテナーに直接配置することはできず、レイアウトボックス内部に配置する必要があります。

このセクションでは以下のサブセクションにより、レイアウトボックスの使用方法について説明します:

- レイアウトボックスの挿入
- レイアウトボックスの選択と移動
- レイアウトボックスのサイズを修正
- レイアウトボックス スタイルプロパティの定義
- レイアウトボックスにコンテンツを挿入
- レイアウトボックスの表示順序

レイアウトボックスの挿入

レイアウトボックスはレイアウトコンテナーの中にだけ挿入することができます。レイアウトボックスを追加するには デザイン要素の挿入 ツールバーにあるレイアウトボックスの追加アイコンをクリックして、レイアウトボックスを挿入するレイアウトコンテナーの場所をクリックします。クリックされた

場所が左上端となるようなかたちでレイアウト ボックスが挿入されます。レイアウト ボックスは透明で境界線を持たず、デフォルト のテキスト が表示されます。

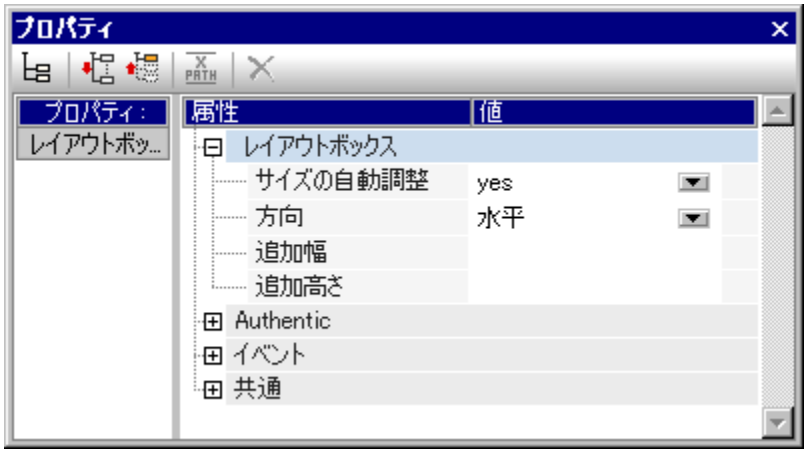
レイアウト ボックスの選択と移動

レイアウト ボックスの左または上側にある境界線までカーソルを移動させ、カーソルが移動状態になった状態でクリックすることで、レイアウト ボックスを選択することができます。マウスボタンを押下しながら、レイアウト ボックスをレイアウト コンテナー内にある他の場所へ移動することもできます。レイアウト ボックスが選択された状態で矢印キーを押下することでも、レイアウト ボックスを移動することができます。レイアウト ボックスが選択されると、関連するプロパティやスタイルがそれぞれのサイド バーに表示されます。

レイアウト ボックスのサイズを修正

各レイアウト ボックスには**サイズの自動調整**という名前のプロパティが含まれています（以下のスクリーンショットを参照）。このプロパティの値を yes にセットすることで、挿入された任意の（マークアップを含む）コンテンツに合わせるかたちでレイアウト ボックスの大きさが自動的に調整されます。**自動調整**の値が no にセットされた場合、コンテンツが挿入されてもレイアウト ボックスの大きさが自動的に変更されることはありません。

右および下側の境界線をドラッグすることで、レイアウト ボックスの大きさを手動で変更することができます。矢印キーを使用することでもレイアウト ボックスの大きさを変更することができます。まず レイアウト ボックスを選択 し、Shift キーを押下しながら、矢印キーを押下することで、レイアウト ボックスを目的の大きさに変更することができます。

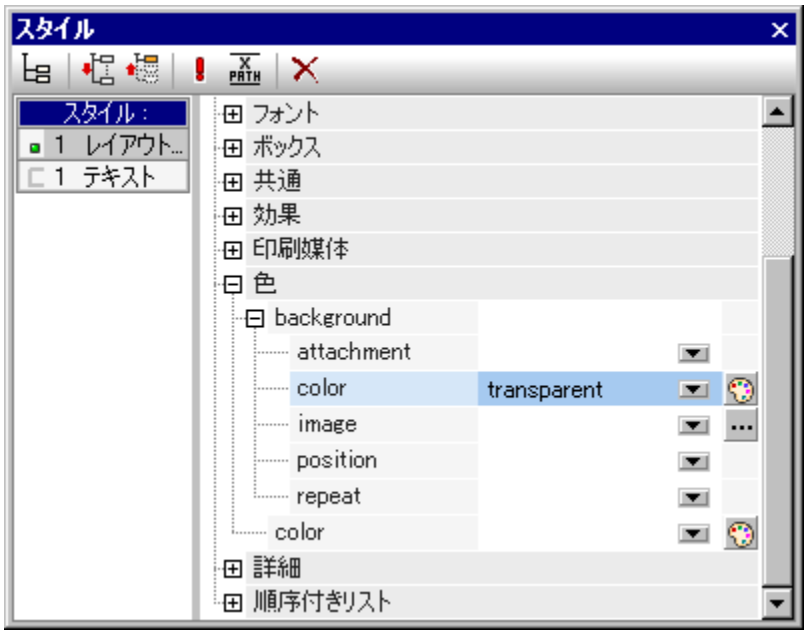


追加高さと**追加幅**プロパティにより、自動調整にて決定される大きさを広げることができます。追加幅と追加高さは、レイアウト ボックスの大きさが手動で変更された際に、新たに計算されます。反対に、これらの値を変更することでも、レイアウト ボックスの大きさを変更することができます。

メモ　レイアウトボックスでは Enter キーを押下することで改行が入力されます。改行を含まないコンテンツが追加されると、その行の長さがレイアウトボックスの幅とともに大きくなり、**追加幅**の値が影響されることになります。

レイアウト ボックス スタイルプロパティ

レイアウト ボックスのスタイルプロパティは、スタイルサイド バーにあるレイアウト ボックスにてセットすることができます（以下のスクリーンショットを参照）。スタイルはレイアウト ボックスの選択時に表示され、編集することができます。

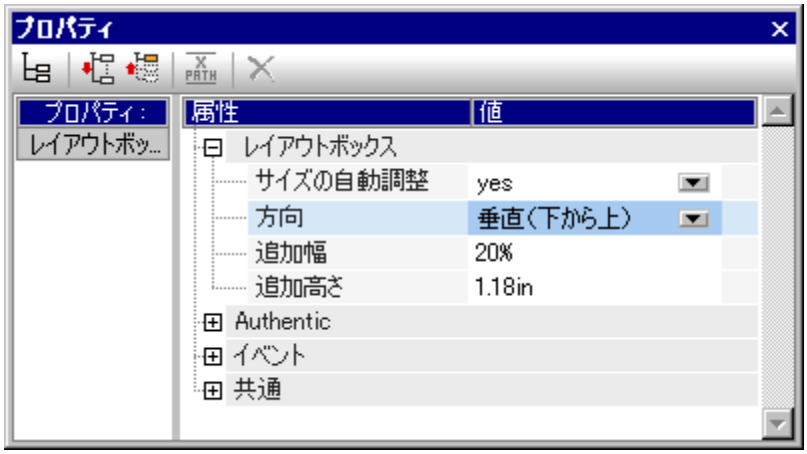


メモ （透明を意味する）transparent という値を、background-color のプロパティコンボボックスから選択することができます（カラーパレットから選択することはできません）。以下で説明されるように、レイアウトボックスが重なり合う際にこの値を使用することができます。

レイアウトボックスにコンテンツを挿入

レイアウトボックスには SPS と同様の方法で任意のデザイン要素を挿入することができます。レイアウトコンテナーまたはLayout Line はレイアウトボックスに挿入できないことに注意してください。留意すべき点を以下に示します:

- コンテキストノードが必要なデザイン要素が挿入された場合、カレントノードがコンテキストノードとして使用されます。レイアウトモジュールの作成に使用されたノードがカレントノードになります。
- レイアウトボックス内のテキストコンテンツは、時計回りまたは反時計回りに90度回転させることができ、上から下、または下から上の方向へテキストを表示することができます。デザインにて回転させるテキストを選択し、プロパティサイドバーにある**レイアウトボックス**グループを選択します。**方向**プロパティにて目的の値を選択してください。

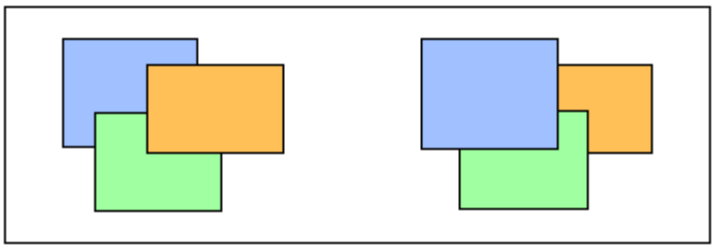


以下の点に注意してください:

- 出力にて文字は回転されますが、デザインではそのように表示されません。
- このプロパティはテーブルセル内のテキストにも適用することができます。

レイアウト ボックスの表示順序

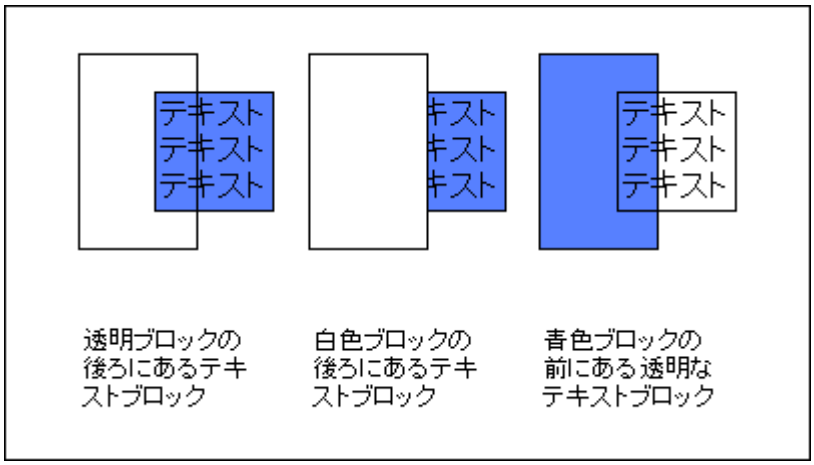
レイアウト ボックスは重ね合わせて表示することができます。あるレイアウト ボックスが別のレイアウト ボックスの上に配置されると、レイアウト ボックスが透明になっていない場合、覆われたレイアウト ボックスが別のレイアウト ボックスにより隠れてしまいます。レイアウト ボックスを重ね合わせることで最前面にあるレイアウト ボックスだけが完全に表示され、その他のレイアウト ボックスは部分的にしか表示されないか、全く表示されなくなります。



レイアウト ボックスのコンテキスト メニューにある**順序** メニューコマンドを使用することで、選択されたレイアウト ボックスを前面または背面へ移動することができます。これらのコマンドを使用することで、レイアウト ボックスを (i) 最も近いボックスに対して移動 (**前面へ移動**ならびに**背面へ移動**コマンド)、または (ii) 全てのボックスに対して移動 (**最前面へ移動**ならびに**最背面へ移動**コマンド) することができます。上のスクリーンショットでは、以下の順序によりボックスが配置されています。

- *左側の図*： オレンジ、緑、青
- *右側の図*： 青、緑、オレンジ

透明 (transparent) な背景 (レイアウト ボックスにデフォルトで使用される背景) をともなうレイアウト ボックスは、特に複数のボックスが透明となっており、更に境界線を持たない際に、他のアイテムに対して順序が変更されていないように見えることもあります。以下のスクリーンショットでは、透明な背景が重ね合わせでどのように表示されるか示されます。



メモ　レイアウト ライン を、レイアウト ボックスと一緒に追加することで、これらラインの順序を他のアイテムに対して移動することができます。

関連項目

- レイアウト コンテナー

- ライン

5.12.3 ライン

ラインは(レイアウト ボックスではなく レイアウト コンテナー内に挿入 した後に、レイアウト コンテナー内で 選択、移動や大きさの変更 を行い、プロパティの割り当てや、レイアウト ボックスならびにラインからなる レイアウト アイテムの表示順序を変更 することができます。

ラインを挿入する

以下の操作により、ラインをレイアウト コンテナーに追加することができます:

1. デザイン要素の挿入 ツールバーにあるラインの挿入アイコンをクリックします。
2. 追加するラインの起点となるレイアウトコンテナー内の位置をクリックします。
3. マウスボタンを離すこと無く、起点から終点までマウスをドラッグします。終点にてマウスボタンを離します。

黒色のラインが挿入されます。両端には起点と終点を表すためのドットが表示されます。

ラインを選択、移動、大きさを変更する

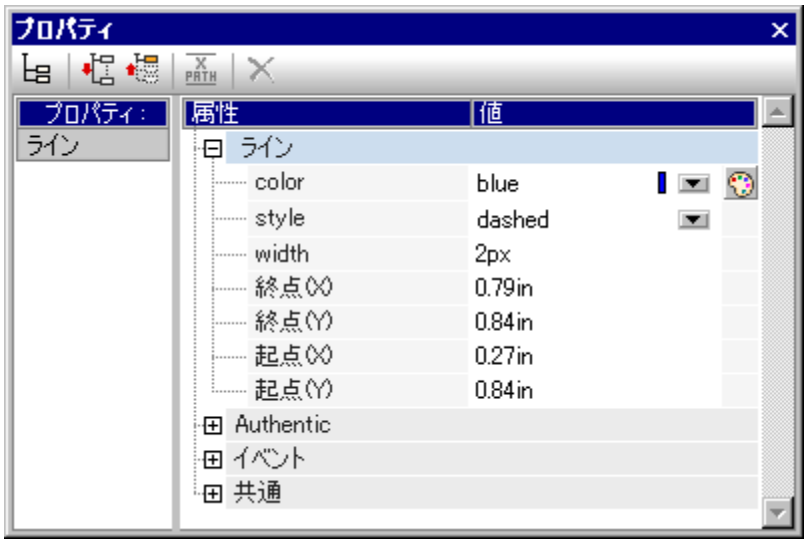
メインウィンドウでは、以下のドラッグアンドドロップ機能を使用することができます:

- ラインを選択するには、ラインの一部をクリックします(ラインの上部にカーソルがある場合、マウスポインターが移動状態になります)。ラインを選択した後には、プロパティがプロパティサイドバーに表示され、編集を開始することができます(以下を参照)。
- ラインを移動するには、ラインが選択されている状態で、目的の場所へドラッグします。ラインが選択されている状態で矢印キーを押下することでも、上下左右へラインを移動することができます。
- ラインの大きさや方向をグラフィカルに変更するには、起点または終点を選択し、大きさや方向を変更するように移動してください。Shift キーを押下しながら矢印キーを使用することでも、ラインの大きさや方向を変更することができます(矢印キーにより、終点が移動されます)。

ラインプロパティ

ラインが選択されている場合、そのプロパティがプロパティサイドバーに表示され、プロパティの編集を行うことができます(以下のスクリーンショットを参照)。ラインを右クリックして、**ラインプロパティを編集** を選択することでも、ラインのプロパティを編集することができます。

(C) 2015-2021 Altova GmbH

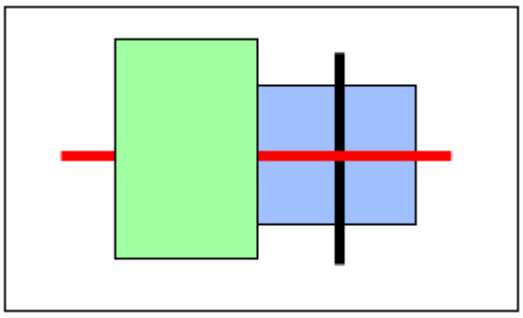


プロパティサイドバーでは、以下のラインプロパティを編集することができます:

- *color*：ラインの色を指定します。デフォルトでは黒色となっています。
- *大きさと位置*：x-軸（水平）ならびに y-軸（垂直）の位置によりラインの起点と終点の位置を指定することができます。レイアウトコンテナーの左上が原点 (x=0, y=0) となります。
- *width*：ラインの太さを指定します。

ラインと表示順序

重ね合わせになっているレイアウトボックスやその他のラインと、あるラインが同じ位置にある場合、そのラインのコンテキストメニューに表示される**順序**メニューコマンドを使用することで、前面や背面へ移動することができます。これらのコマンドを使用することで、ラインを（i）最も近いアイテムに対して移動（**前面へ移動**ならびに**背面へ移動**コマンド）、または（ii）全てのアイテムに対して移動（**最前面へ移動**ならびに**最背面へ移動**コマンド）することができます。



上のスクリーンショットでは、以下の順序で重ね合わせが表示されます：緑色のボックス、赤色のライン、黒色のライン、青色のボックス

関連項目

- レイアウトコンテナー
- レイアウトボックス

5.13 変更機能

テンプレートやテンプレートのコンテンツが選択されている状態で使用することのできる変更機能により、(i) そのテンプレートが適用されるノード、または(ii) そのノードがデザインでどのように作成されるのかを変更することができます。

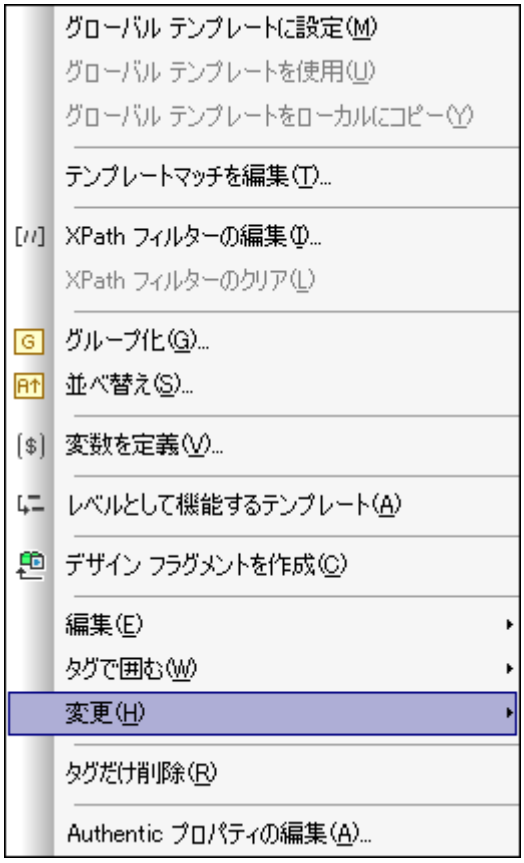
変更機能により変更することができるもの

ノードやそのコンテンツを変更することができます。以下に示されるスクリーンショットのうち、左側のイメージではノードが選択されており、右側のイメージでは、ノード内部にあるコンテンツが選択されています。

上にあるスクリーンショットでは、n1:Name 要素が(コンテンツ) として作成されており、ノードのコンテンツが(コンテンツ) プレースホルダにより示されます。他にも、別種類のコンテンツとしてノードを作成することもできます(例: 入力フィールドやコンボボックス)。その他の種類のコンテンツを選択することもできます。

変更コマンド

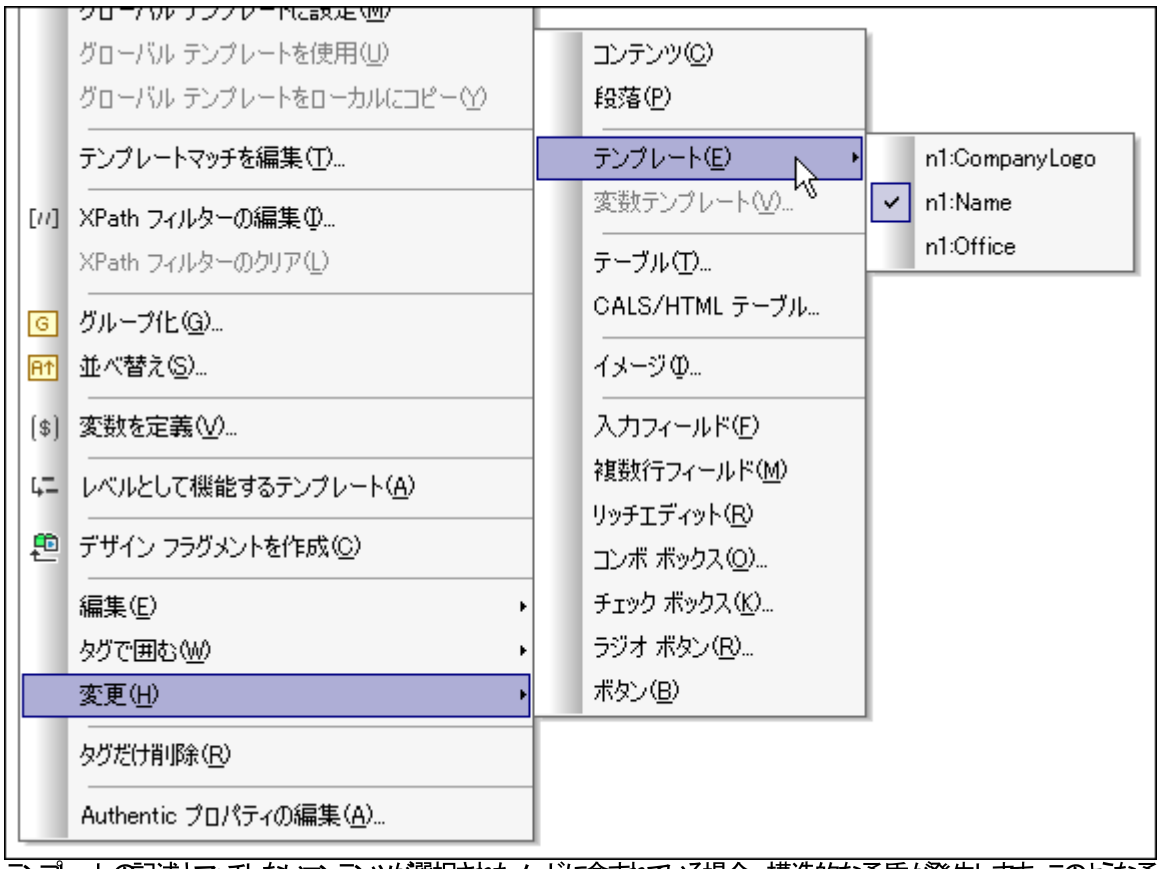
選択されたものを右クリックして、コンテキストメニューから変更コマンドへアクセスすることができます(以下のスクリーンショットを参照)。



(C) 2015-2021 Altova GmbH

テンプレートマッチを編集する

テンプレートが選択されている場合、そのテンプレートが適用されるノードを変更することができます。例えば、スキーマにて要素の名前が変更された際にこの機能を利用することができます。変更コマンドへマウスオーバーして、サブメニューからテンプレートを選択すると、選択されたノードの兄弟要素として挿入することができる全ノードのリストが表示されます。ノードをクリックすることで、そのノードに対してテンプレートが適用されます。

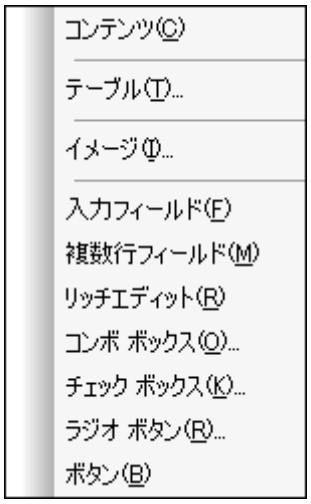


テンプレートの記述とマッチしないコンテンツが選択されたノードに含まれている場合、構造的な矛盾が発生します。このような矛盾が発生すると、ノードタグに赤色の横線が表示され、エラーであることが示されます。

テンプレートマッチを変更することで、（ノードではなく）変数テンプレートをマッチさせることもできます。

ノードのコンテンツ型を変更する

テンプレートやそのコンテンツが選択されている場合、ノードが作成されたコンテンツの型を変更することができます。コンテキストメニューの変更コマンドにマウスオーバーすると、選択されたノードを変更することができるコンテンツの型がサブメニューにて表示されます（以下のスクリーンショットを参照）。



上のスクリーンショットは、コンボボックスが選択されている状態で表示されたものです。

関連項目

- SPS ファイルコンテンツ
- テンプレートとデザインフラグメント

(C) 2015-2021 Altova GmbH

6 SPS 構造

SPS ドキュメントの構造は、入力ならびに出力をベースとしており、以下の項目により制御されます:

- スキーマソース
- モジュールSPS
- テンプレートとデザインフラグメント

入力ベースの構造: スキーマとモジュールSPS ファイル

入力ベースというのは、SPS のソーススキーマにより入力ドキュメントの構造が指定されることで、この構造がSPS ドキュメントの構造となります。例えば、ソーススキーマにて Office 要素のシーケンスが定義されている場合、Office 要素のテンプレートを SPS に含めることができます。処理を行う際に、このテンプレートがソースデータドキュメント内にある各 Office 要素に対して適用されます。

ソースドキュメントの構造によりSPS ファイルのデザインが決定される例は、テーブルでも見ることができます。Office 要素に複数のPerson 子要素が含まれており、各 Person 要素にはName やAddress、Telephone といった子要素のセットが含まれている場合、Person 要素に対してテーブルのテンプレートを作成することがで、各 Person 要素がテーブルにおける行として表示されます(以下のスクリーンショットを参照)。テーブルのカラムは、Person の詳細(Person 要素の子要素)となります。

First	Last	Title (sorted by)
Loby	Matise	Accounting Manager
Frank	Further	Accounts Receivable
Vernon	**Callaby**	Office Manager

Person 要素の構造と複数の `Person` 要素が兄弟ノードであるために、このようなテンプレートが可能になります。テンプレートではPerson 要素に対してデザインされた行が1つ定義されており、この行によりソースドキュメント内にある各 `Person` 要素が処理され、`Person` 要素の数だけ新たな行が作成され、子要素がテーブルのカラムとして表示されます。

各種スキーマソースの使用方法については スキーマソースのセクションにて記述されます。

更に、StyleVision では作成したSPS をモジュールとして別のSPS内部で再利用することができます。この機能により、モジュールを別の構造内部に含めることができるようになります。モジュール内部に含まれているスキーマ構造は、モジュールを含んでいるSPS のスキーマ構造に対して合致するものでなければなりません。モジュールSPS の使用方法については、 モジュールSPS のセクションを参照ください。

出力ベースの構造: テンプレートとデザインフラグメント

スキーマソースにより入力データドキュメントの構造が与えられる一方で、出力ドキュメントのデザインは SPS ドキュメントにより決定されます。このデザインは、メインテンプレートと呼ばれるドキュメントテンプレートに収められます。メインテンプレートにはお通常複数のコンポーネントテンプレートが含まれ、グローバルテンプレートに対する参照が行われます。テンプレートに関する説明は、 テンプレートとデザインフラグメントのセクションにて記述されます。

複数のテンプレートをモジュールとして使用する機能はデザインフラグメントと呼ばれる StyleVision の機能に拡張され、特定の処理を再利用可能なドキュメントフラグメントに対して割り当てることができます。グローバルテンプレートと違い、デザインフラグメントは(i) 複数のテンプレートから構成することが可能で、(ii) 異なる処理が含まれる同一のコンテンツを別々のデザインフラグメントにて作成することができ、状況に応じて、必要な処理を行うことが可能になります。例えば、状況に応じて Email ノードをリンクとして表示し、太字の赤色で Email 要素を表示することが可能になります。デザインフラグメントを2つ用意することで、それぞれ目的の処理を行うことができます。

デザインフラグメントに関する詳細は、デザインフラグメントのセクションにて記述されます。

関連項目

- 使用方法の概要

(C) 2015-2021 Altova GmbH

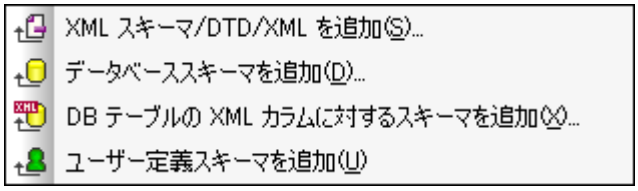
6.1 スキーマソース

スキーマソースはデザインを行う際の出発点ともなるもので、(i) スキーマの選択と、(ii) スキーマにて選択したルート要素によりデザインの構造は影響を受けることになります。

スキーマの選択

以下の操作により、新たな SPS ファイルに対するスキーマの選択を行うことができます:

1. メニューオプションから「**ファイル | 新規作成**」を選択し、サブメニューにある(**新規作成 (空)** を除く)オプションを選択して、スキーマソースを直接選択します。
2. メニューオプションから「**ファイル | 新規作成**」を選択し、サブメニューから「**新規作成 (空)**」を選択します。新規 SPS が作成され、GUI に表示されます。デザイン概要サイドバーにて**新しいソースの追加**コマンドを選択します。スキーマとして追加することのできるものが表示されます(以下のスクリーンショットを参照)。このメニューに表示されている各コマンドについては、このセクションのサブセクションにて記述されます。.



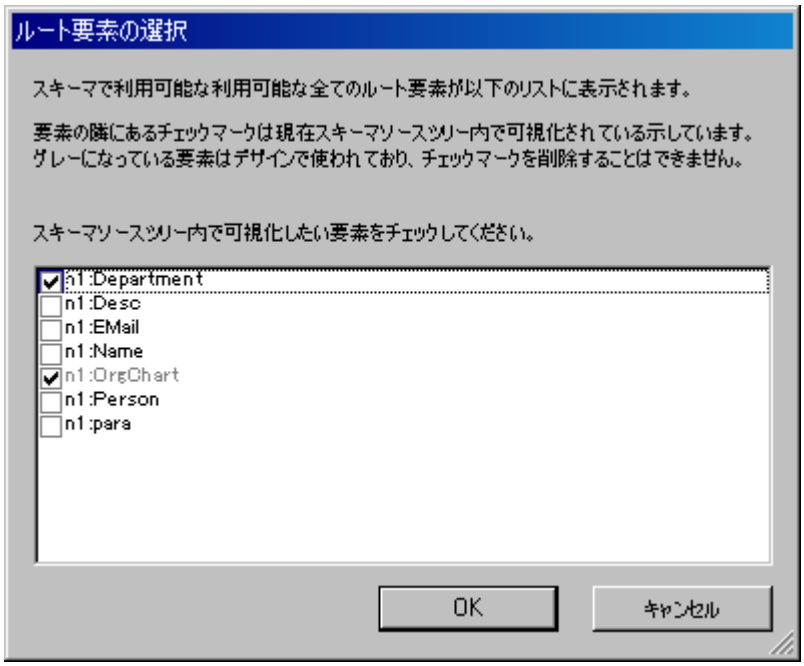
スキーマソースは、ファイル、、DB、またはユーザー定義のスキーマから選択することができます。グローバルテンプレートを使用するか、またグローバルテンプレートとして使用する要素がスキーマにてグローバル要素として定義されているかどうかがここでは重要になります。ファイルから DTD を追加する場合、DTD 内で定義された要素は全てグローバル要素となることに注意してください。ファイルから XML スキーマを追加する場合、どの要素がグローバル要素として定義されているか、そしてスキーマに対して変更が必要な場合には、お使いの XML 環境にてそれができるかどうか調べるようにしてください。DB が選択された場合、インポートを行う際に DB 内にあるどのテーブルをインポートするか決定することができます。この選択により、DB から生成される XML スキーマの構造が決定されます。

メモ 名前空間を SPS へ追加し、SPS から XSLT スタイルシートを生成するには、SPS がベースとしている XML スキーマのトップレベル schema 要素に対して名前空間を追加する必要があります。

ルート要素

スキーマツリーに複数のグローバル要素 が含まれている場合、複数のルート要素 (ドキュメント要素) をデザインで使用するために選択することができます。この機能により、複数のドキュメント要素にマッチするテンプレートを SPS デザインに含めることができます。UniversalSchema.xsd という名前のスキーマをベースにした UniversalSPS.sps にて Element-A と Element-B という2つのルート要素が与えられている場合、Element-A をドキュメント要素とするインスタンスドキュメントと、Element-B をドキュメント要素とする別のインスタンスドキュメントを、単一の SPS から扱うことが可能になります。各 XML インスタンスに対して関係するテンプレートが使用され、そうでないテンプレートは使用されません。これは各 XML インスタンスドキュメントにおいて、そのドキュメント要素にマッチするテンプレートが SPS 内に1つしか無いためです。例えば、ドキュメント要素の /Element-A は、/Element-A を選択するテンプレートにマッチしますが、/Element-B を選択するものにはマッチしません。複数のグローバル要素がスキーマにて定義されている場合、それらのうちのどれかをドキュメント要素としている XML ドキュメントも妥当なものとして認識されます(勿論、そのサブ構造もスキーマに対して妥当でなければなりません)。

複数のルート要素 (ドキュメント要素) を使用するように SPS をセットアップするには、スキーマのルート要素エントリーの右側にある ボタンをクリックしてください。以下に示されるダイアログが表示されます。



ダイアログにはスキーマ内部にある全てのグローバル要素が表示されます。ルート要素 (ドキュメント要素) として使用することができるグローバル要素を選択し、OK をクリックしてください。選択された要素がルート要素リストとして表示され、ルートドキュメント要素として利用できるようになります。これらのドキュメント要素に対してテンプレートを作成することができるようになります。これらのテンプレートは、ルート要素テンプレートの代わりとして使用することができます。XML ドキュメントが SPS とともに処理された場合、代替ルート要素テンプレートのうちの1つだけ－XML ドキュメントのルート（ドキュメント）要素－が使用されます。

Element-A をドキュメント要素とする XML ドキュメントが SPS により処理されると、Element-A にマッチするルートテンプレートが使用され、SPS 内にあるその他全てのルート要素テンプレートは無視されることになります。Element-B をドキュメント要素とするドキュメントが処理された場合、Element-B にマッチするルートテンプレートが使用され、SPS 内にあるその他全てのルート要素テンプレートは無視されることになります。この機能により、異なるルート（ドキュメント）要素を伴う複数の XML ドキュメントに対して同じ SPS を使用することができるようになります。

関連項目

- スキーマツリーサイドバー

6.1.1 DTD と XML スキーマ

SPS は XML スキーマや DTD をベースとすることができるほか、。以下の操作により、XML スキーマや DTD をスキーマソースとして作成することができます:

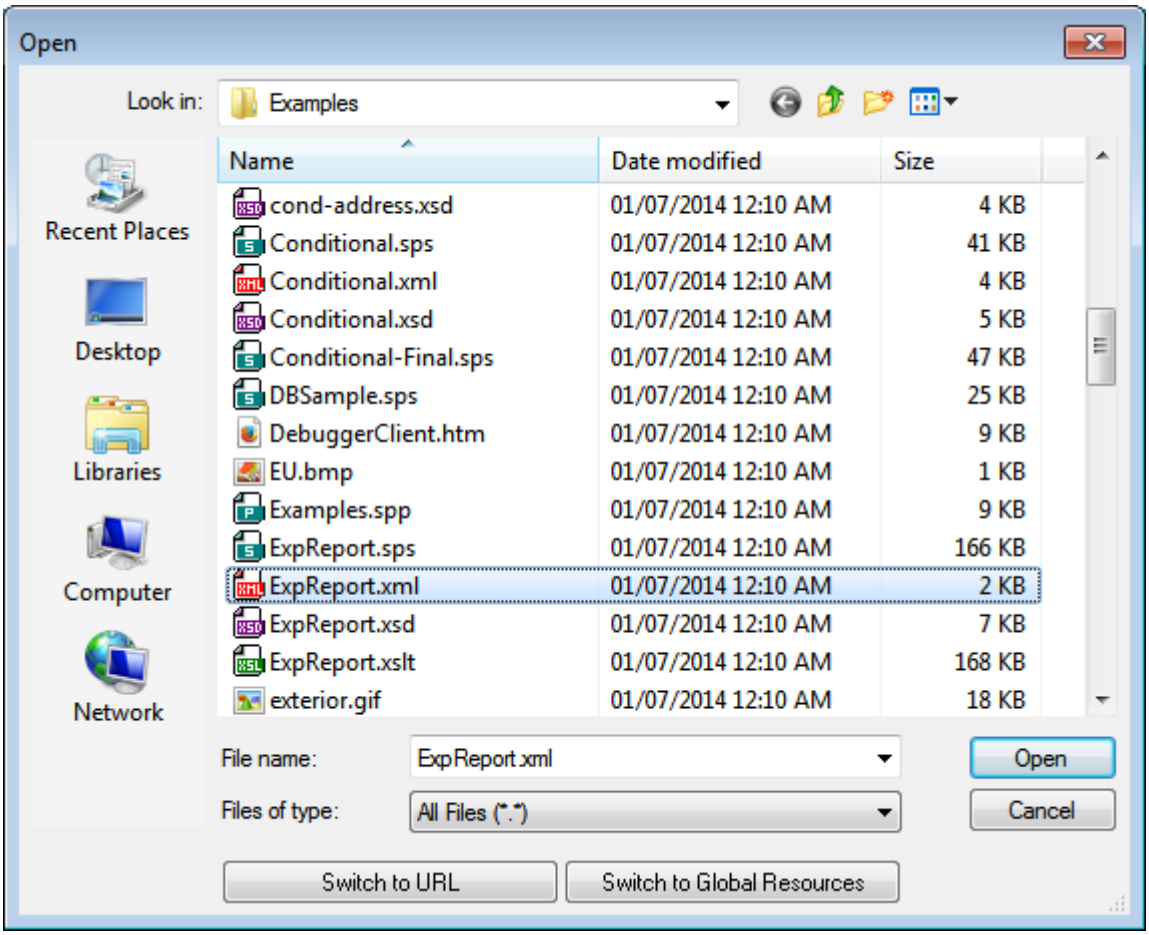
- SPS を作成（「**ファイル | 新規作成 | XML スキーマ/DTD/XML から新規作成**」）することで、XML スキーマや DTD がスキーマソースとして直接作成されます。
- XML スキーマや DTD を、空の SPS

コマンドを選択することにより、XML スキーマや DTD を参照するためのダイアログが表示されます。スキーマが妥当な場合、そのスキーマがスキーマソースとして作成され、スキーマツリーサイドバーにあるスキーマソースツリーに表示されます。また XML ファイルを選択することができます。XML スキーマ（.xsd）または DTD ファイルが XML ファイルに関連付けられている場合、XML スキーマや DTD ファイルがソーススキー

マとしてロードされ、XML ファイルが作業用 XML ファイルとしてロードされます。XML ファイルに関連付けられているスキーマが存在しない場合、XML ファイルの構造とコンテンツから XML スキーマを生成するか、既存のスキーマを選択するように促されます。スキーマの生成を選択した場合、生成されたスキーマがソーススキーマとしてロードされ、XML ファイルが作業用 XML ファイルとしてロードされます。

▼ URL とグローバルリソースを使用してファイルを選択または保存する

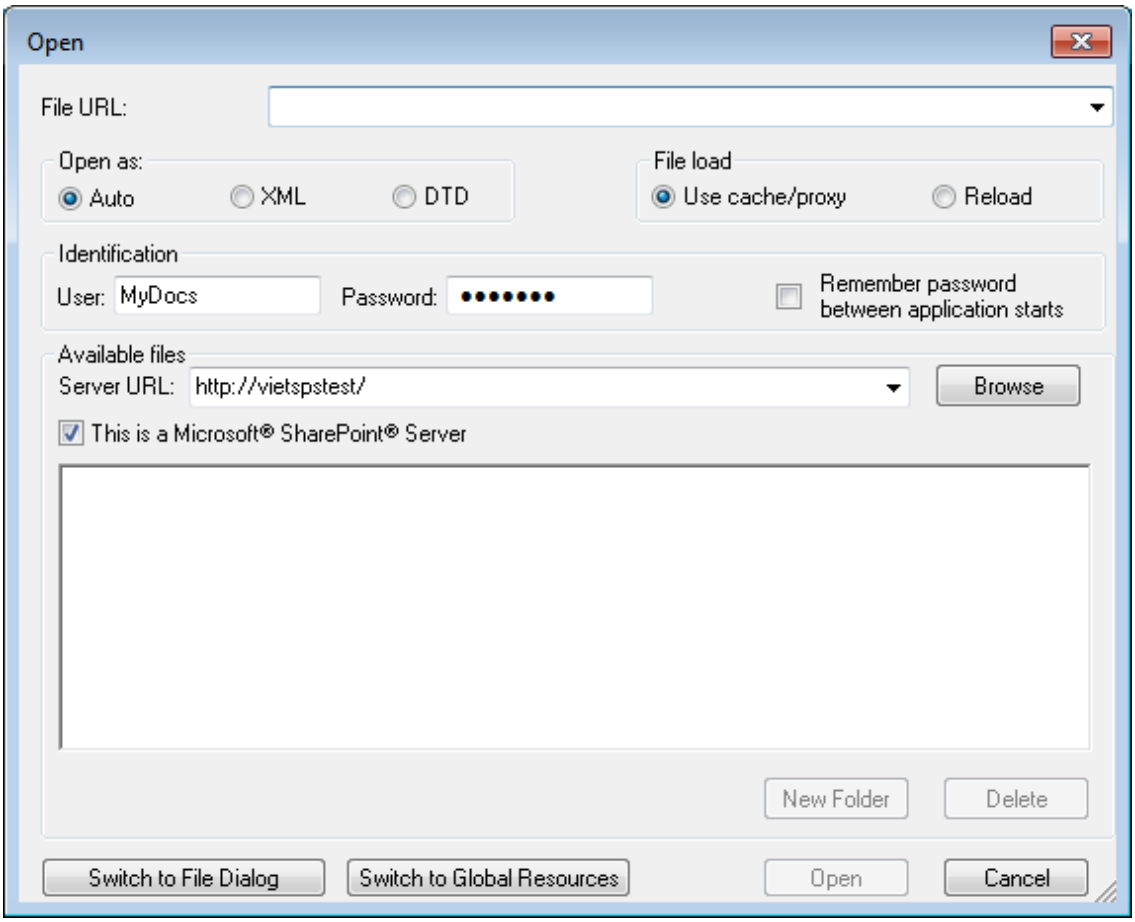
「ファイルを開く」および「ファイルの保存」ダイアログでは、URL またはグローバルリソースを使用して必要なファイルを選択またはファイルを保存することができます（*下のスクリーンショット参照*）。選択プロセスに移動するために「**URL に切り替える**」または「**グローバルリソース**」をクリックしてください。



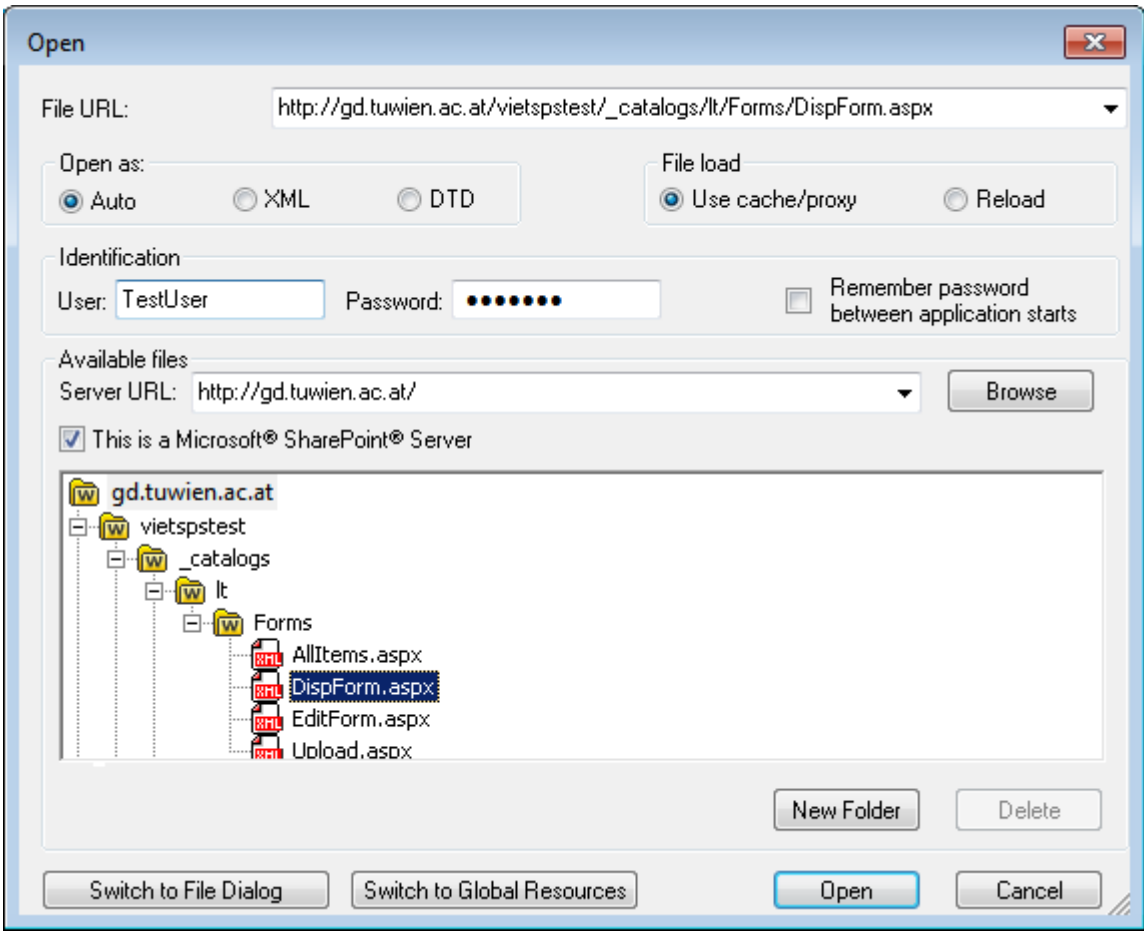
URL を使用してファイルを切り替える

URL から（開くまたは保存する）ファイルを選択するには以下を行ってください：

1. 「**URL に切り替える**」コマンドをクリックします。これにより、開くまたは保存ダイアログの URL モードにスイッチされます。（*下のスクリーンショットは開くダイアログを表示しています*）。



2. アクセスするURL をサーバー *URL* フィールドに入力します（上のスクリーンショット）。サーバーがMicrosoft® SharePoint® Server の場合 *Microsoft® SharePoint® Server* チェックボックスをチェックしてください。この種類のサーバー上にあるファイルと作業する場合、下のMicrosoft® SharePoint® Server メモを参照してください。
3. サーバーがパスワードにより保護されている場合、*ユーザー*と*パスワード*フィールドに入力してください。
4. 「参照」をクリックして、サーバーのディレクトリ構造をナビゲートします。
5. フォルダーツリー内でロードするファイルを参照してクリックします。



ファイルURL がファイルURL フィールドに表示されます（*上のスクリーンショット参照*）。「開く」または「保存」ボタンを使用できるようになります。

6. 「開く」をクリックしてファイルをロードするか、「保存」をクリックして保存します。

以下の点に注意してください：

- W ebDAV をサポートするサーバーとMicrosoft SharePoint Servers のみで参照機能を使用することができます。サポートされるプロトコールはFTP、HTTP およびHTTPS です。
- ファイルを開く際のロード処理を更に管理するには、ローカルのキャッシュまたは（ファイルが以前にロードされている場合スピードを向上する）プロキシのサーバーからファイルをロードします。（ファイルが事前にロードされることにより更に処理スピードを向上することができます）。また、電子出版またはデータベースシステムなどの作業中のファイルを再ロードして、「再ロード」オプションを選択します

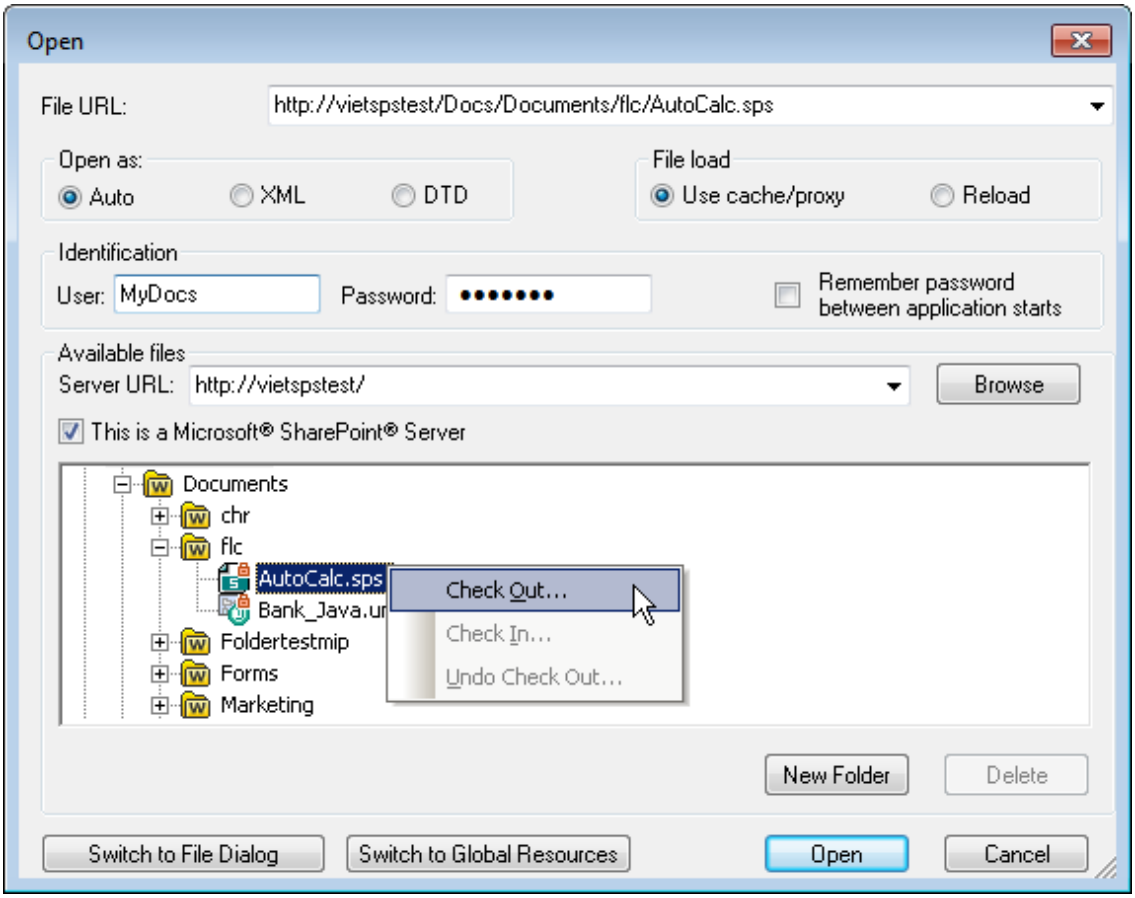
。

▼ Microsoft® SharePoint® Server メモ

Microsoft® SharePoint® Servers に関する以下の点に注意してください：

- 「利用可能なファイル」ペインに表示されるディレクトリ構造では、ファイルアイコンにはファイルのチェックイン/チェックアウトの状

態を表示するシンボルがあります（*下のスクリーンショット*）。



ファイルを右クリックすると、ファイルで使用することのできるコンテキストメニューがポップアップします（*上のスクリーンショット*）。

- 異なるファイルアイコンが以下に表示されています:

	チェックイン状態。
	チェックアウト可能。他のユーザーによりチェックアウト状態。
	チェックアウト不可能。ローカルに対してチェックアウト。

- ファイルをチェックアウトした後、Altova アプリケーションを使用して編集することができ、「ファイル | 保存（Ctrl+S）」を使用して保存することができます。
- 編集されたファイルを「URL を開く」ダイアログのコンテキストメニューからチェックインすることができます（*上のスクリーンショット参照*）または、アプリケーションのメインウィンドウのファイルタブを右クリックすることによりポップアップするコンテキストメニューを使用して（*下のスクリーンショット*）。

- ファイルが他のユーザーによりチェックアウトされている場合、チェックアウトすることができません。
- 他のユーザーによりファイルがチェックアウトされている場合、チェックアウトを行うことはできません。これによりファイルを変更す

ることなくサーバーに戻すことができます。

- 変更することなしにファイルがサーバーへ戻されます。ある Altova 製品でファイルをチェックアウトした場合、他の Altova 製品で同じファイルをチェックアウトすることはできません。この時点で使用できる Microsoft® SharePoint® Server をサポートする Altova アプリケーションのコマンドは、以下の通りです: **「チェックイン」** および **「チェックアウトを元に戻す」** です。

▼ グローバルリソースを使用してファイルを選択または保存する

グローバルリソースを使用してファイルを開くまたは保存する場合、**「グローバルリソース」** をクリックします。グローバルリソースを選択するためのダイアログが表示されます。これらのダイアログは以下のセクションで説明されています: グローバルリソースの使用. グローバルリソースの説明に関してはこのドキュメントの グローバルリソース セクションを参照してください。

XML スキーマにおける anyType データ型

XML スキーマ内にある要素に、XML スキーマの anyType データ型が割り当てられている、またはどのデータ型も割り当てられていない場合、そのスキーマにある全ての要素が使用可能な子要素としてスキーマツリーに表示されます。例えば、email という要素に対してデータ型が割り当てられていない場合、person、address、city、tel といった全てのグローバル要素が、使用可能な子要素としてスキーマツリーに表示されます。例えば xs:string といったデータ型を email 要素に割り当てることで、このような振る舞いを回避することができます。

関連項目

- スキーマツリー サイドバー

6.1.2 DB スキーマ

DB から生成されたスキーマをベースに SPS をデザインすることができるほか、DB ベースのスキーマ(DB スキーマ)をスキーマソースとして使用することができます。以下にあるいずれかの方法により、スキーマソースとして DB スキーマを作成することができます:

- SPS が DB から(**「ファイル | 新規作成 | データベースから新規作成」**または**「ファイル | 新規作成 | DB テーブルの XML カラムから新規作成」**により)直接作成された場合、新規 SPS に対して DB スキーマが生成されます。
- スキーマソースを持たない空の SPS に対して DB スキーマを(デザイン概要サイドバーから)追加することができます。

DB スキーマ生成コマンドにより一時的な XML スキーマが選択された DB から生成され、スキーマソースとして スキーマツリー サイドバー に表示されるほか、XML DB から DB ベースのスキーマ(DB スキーマ)がロードされます。DB という名前の(しかし DB とは無関係の)要素がドキュメント要素として作成され、このドキュメント要素内に DB 構造が作成されます。スキーマの生成過程にて、データベースからインポートするテーブルやデータを選択するように促されます。これらのデータベーステーブルは DB 要素の子要素として XML スキーマ内に作成されるとともに、グローバルテンプレートリスト内のアイテムとして表示されます。

DB から XML スキーマを作成する

DB から XML スキーマを作成するには、2 つのステップを踏むことになります:

- データベースへの接続: このステップでは、(MS Access DB の)DB ファイルへのアクセスや、(MS Access DB 以外の)データベースに対する接続文字列を構築します。. **ADO 接続のみがサポートされていることに注意してください。** これは、ADO がコマンドサポートを形作り、階層順なデータの作成をサポートするからです。
- DB データの選択 : このステップでは、データベースのテーブルやデータ構造が選択されます。非 XML DB の場合、一時的な XML スキーマが生成され、選択されたデータテーブルをベースにした XML ファイルが作成されます。

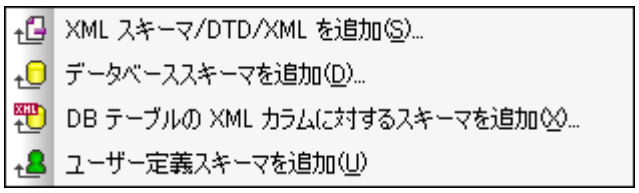
スキーマソースがスキーマツリーに一旦表示されると、SPS のデザインを開始することができます。一時的な XML ファイルは出力プレビュータブがクリックされるたびに作成されます。生成された XML スキーマに従うかたちで XML ファイルは構築され、選択された DB テーブルから得られたデータが含まれます。

6.1.3 ユーザー定義スキーマ

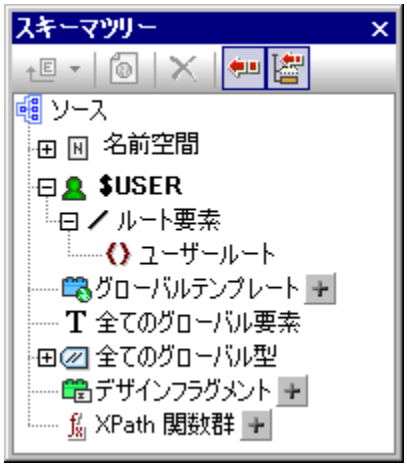
スキーマツリーサイドバーでは、ユーザー定義スキーマを素早く作成することができます。ベースとなるスキーマを持たない XML ドキュメントに対して SPS を作成する際にこの機能を利用することができます。

以下の操作により、スキーマツリーサイドバーにてユーザー定義スキーマを追加、作成することができます:

1. メニューオプションから「**ファイル | 新規作成 | 新規作成 (空)**」を選択します。デザイン概要サイドバーにて、(ソース行以下にある)**新しいソースの追加**コマンドをクリックし、**ユーザー定義スキーマ**を選択します(以下のスクリーンショットを参照)。

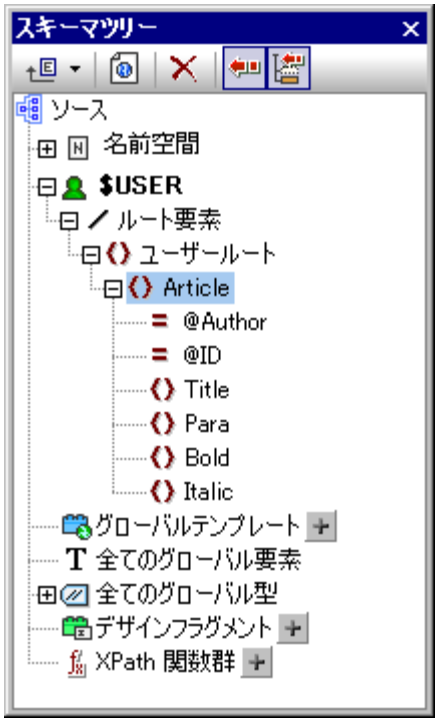


新たなスキーマが作成され、$USER という名前のパラメーターにより表されます(以下のスクリーンショットを参照)。

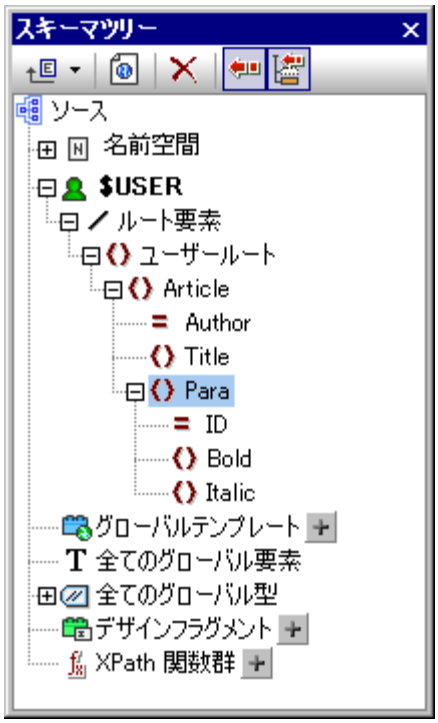


2. ルート要素ツリーには、ユーザールートという名前の ルート要素(ドキュメント要素) が表示されます。
3. ユーザールートをダブルクリックすることで、スキーマがマッチするように XML ドキュメントの ドキュメント要素 の名前を変更することができます。
4. 子要素や属性をドキュメント要素に割り当てるには、ドキュメント要素(ユーザールート)を選択し、 スキーマツリーサイドバーのツールバーにある新規要素の追加アイコン または アイコンの右側にある矢印をクリックして、属性を追加エントリーをクリックします。右クリックして表示されるコンテキストメニューから目的のコマンド

を選択することもできます。要素が選択されている状態で要素の追加や挿入を行うと、選択された要素の前または後に、新たな要素が兄弟要素として追加されます。子要素や子属性を追加することもでき、属性が選択されている状態から、その属性の前または後に他の属性を挿入/追加することができます。新たな要素または属性をツリーに追加した後には、キーボードから名前を入力してください。目的の場所へノードをドラッグすることもできます(次のステップにある記述を参照)。以下のスクリーンショットでは、Article 要素がドキュメント要素となります。Title、Para、Bold、そして Italic 要素と ID ならびに Author 属性が Article の子として追加されています。



5. Bold ならびに Italic 要素と ID 属性を Para の子として移動するには、個々の要素や属性を選択し、Para 要素へドラッグします。マウスポインターが矢印ポインター に変わったところで、ノードをドロップします。ノードが Para の子として作成されます(以下のスクリーンショットを参照)。



6. ドキュメント要素以外の要素が選択されている場合、新たな要素や属性を追加することにより、選択されている要素と同じ階層に新たなノードが追加されます。ノード（要素や属性）を要素ノードへドラッグすることで、そのノードが要素ノードの「子」として作成されます。

ノード名の編集やノードを削除する

要素や属性の名前を編集するには、ノードの名前をダブルクリックして名前の編集を行います。ノードを削除するには、目的のノードが選択されている状態でツールバーにある削除アイコン をクリックするか、コンテキストメニューから**削除**を選択します。

関連項目

- スキーマツリーサイドバー
- デザイン概要サイドバー

(C) 2015-2021 Altova GmbH

6.2 複数のソースから XML データのマージ

XSLT 2.0 または3.0 SPS で使用される XSLT のバージョンとすることで、複数のソース XML ファイルから得られた XML データを組み合わせることができます。

通常、データのマージは ID のような共通したデータをベースに行われます。例えば、従業員 ID により識別される社内の従業員の詳細データ(例: 従業員の詳細情報、給与、出社日)が、複数の XML ファイルにまたがって保管されている場合、これら複数のファイル内に保管されているデータを、従業員 ID をキーとしてマージし、出力ドキュメントにて表示することができます。

メモ Enterprise Edition では、複数のスキーマソースを含めることができ、パラメーター名を使うことで、対応するスキーマから得られた XML ノードを選択することができます(以下のサンプルを参照ください)。Professional ならびに Basic Edition では、XPath 2.0 の doc() 関数を使用することで、目的の XML ファイルと、そのファイル内にある XML ノードを選択することができます。XPath 2.0 の doc() 関数により、外部 XML ドキュメントのドキュメントルートへアクセスすることが可能になり、外部 XML ドキュメント内にあるノードコンテンツを出力に挿入することが可能になります。そのため、doc() 関数を使用する 自動計算 を使うことでも、XML データをマージすることができます(以下のサンプルを参照ください)。

サンプル

(マイ) ドキュメントフォルダー C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples には、サンプル SPS ファイル (MergeData_2_Files.sps) が含まれており、異なるソース XML ファイルのマージの方法が説明されています。SPS は、架空の注文(下にリストされる MergeOrder.xml) からデータを選択します。

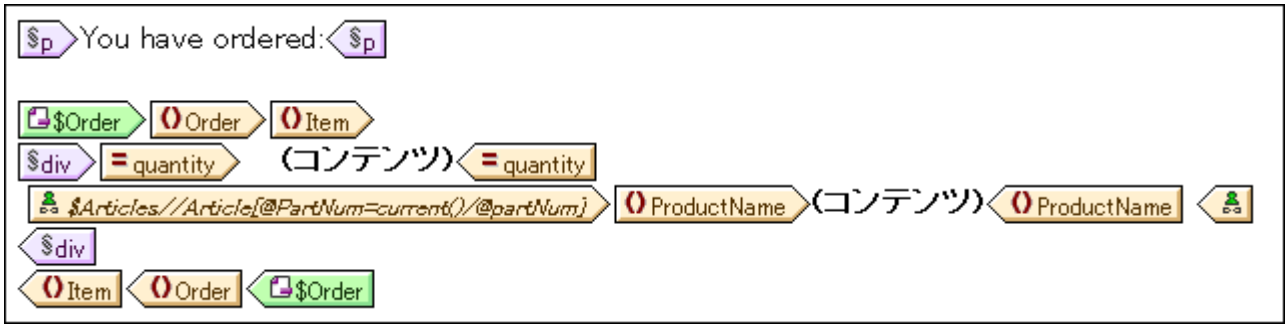
```
<?xml version="1.0" encoding="UTF-8"?>
<Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="MergeOrder.xsd">
   <Item partNum="238-KK" quantity="3" shipDate="2000-01-07" comment="With no inclusions,
please."/>
   <Item partNum="748-OT" quantity="1" shipDate="2000-02-14" comment="Valentine's day
packaging."/>
   <Item partNum="229-OB" quantity="1" shipDate="1999-12-05"/>
   <Item partNum="833-AA" quantity="2" shipDate="1999-12-05" comment="Need this for the
holidays!"/>
</Order>
```

このファイルの /Order/Item/@partNum 属性の値は他のファイル MergeArticles.xml に保管されているカタログから製品を注文するために使用されます(下のリストを参照してください)。

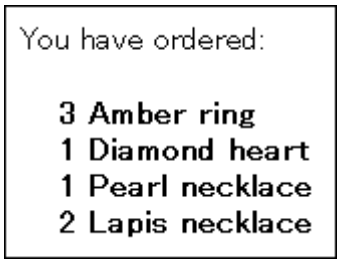
```
<?xml version="1.0" encoding="UTF-8"?>
<Articles xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="MergeArticles.xsd">
   <Article PartNum="833-AA">
      <ProductName>Lapis necklace</ProductName>
      <Price>99.95</Price>
   </Article>
   <Article PartNum="748-OT">
      <ProductName>Diamond heart</ProductName>
      <Price>248.90</Price>
   </Article>
   <Article PartNum="783-KL">
      <ProductName>Uncut diamond</ProductName>
      <Price>79.90</Price>
   </Article>
   <Article PartNum="238-KK">
      <ProductName>Amber ring</ProductName>
      <Price>89.90</Price>
```

```
   </Article>
   <Article PartNum="229-OB">
      <ProductName>Pearl necklace</ProductName>
      <Price>4879.00</Price>
   </Article>
   <Article PartNum="128-UL">
      <ProductName>Jade earring</ProductName>
      <Price>179.90</Price>
</Article>
       ...
</Articles>
```

/Order/Item テンプレート内にユーザー定義テンプレート をセットアップすることで、注文されたパーツの番号により MergeArticles.xml ファイル内の対応する Article 要素が選択され、データのマージが行われます（以下のスクリーンショットを参照）。（/Order/Item コンテキスト内にある）XPath 条件式は $Articles//Article[@PartNum=current()/@partNum] となります。

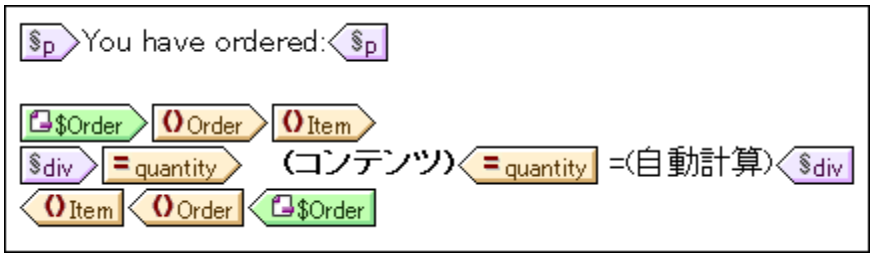


このテンプレートにより、以下のスクリーンショットに示されるような出力が表示されます。



各アイテムの注文数（quantity）が MergeOrder.xml から得られているのに対して、注文されたものの名前は MergeArticles.xml ファイルから得られていることに注目してください。ProductName ノードが /Articles/Article テンプレートのコンテキスト内にて選択されている点にも注目してください。

自動計算 を使用することにより、上で得られたものと同じ結果を得ることができます（以下のスクリーンショットを参照）。スキーマツリーウィンドウから quantity 属性をドラッグすることで、コンテンツとして作成します。その後、以下のスクリーンショットに示される様に自動計算を追加して、以下にある XPath 条件式をその自動計算に与えます。



自動計算の XPath 条件式により、他のスキーマソースのパラメーターや doc() 関数を使用することで、目的のノードを選択することができます:

```
$Articles//Article[@PartNum=current()/@partNum]/製品Name
```

または

```
doc('MergeArticles.xml')//Article[@PartNum=current()/@partNum]/製品Name
```

上にある最初のXPath 条件式が(Enterprise Edition でのみ使用できる)別のXML スキーマを参照しているのに対して、2番目の条件式では(Professional ならびにBasic Edition でも使用することのできる)XPath 2.0 のdoc() 関数を使用しています。

関連項目

- 自動計算

6.3 モジュール SPS

SPS のグローバルテンプレートやデザインフラグメント、JavaScript 関数、そしてページレイアウトアイテムを、他の SPS デザインにて使用することができます。この機能により、以下のようなことが可能になります:

1. 複数の SPS にあるグローバルテンプレートやその他のコンポーネントを再利用することができ、単一のソース編集や出力の一貫性を確保するために使用することができます。
2. SPS をモジュール化して、更に柔軟な構造を作成することが可能になります。

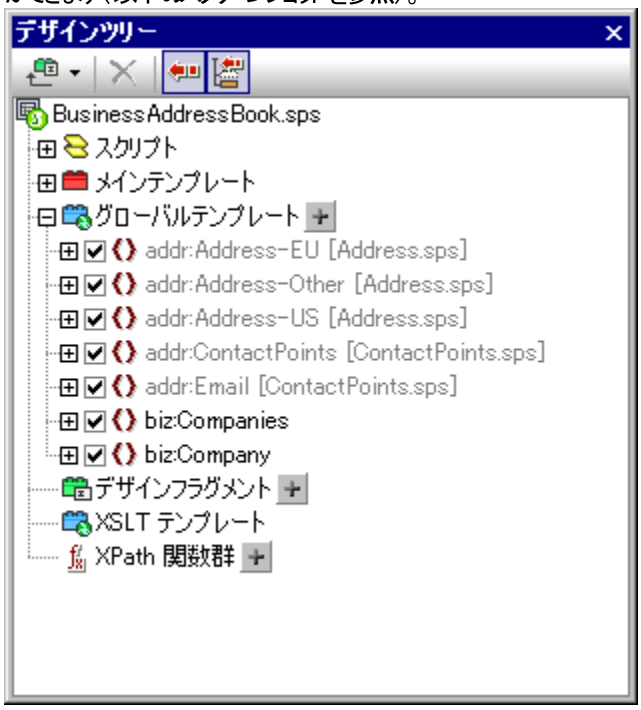
任意の SPS にて、複数の SPS をモジュールとして追加することができます。これらモジュール内にあるコンポーネント(または オブジェクト)のいくつかは、SPS のインポート(や参照)を行う際に利用することができます。

利用が可能なモジュール オブジェクト

利用が可能なモジュール オブジェクト のセクションには、SPS の様々なコンポーネントを SPS にインポートする際の制約と条件が記述されるほか、SPS へのインポートに使用することができないコンポーネントが記されています。追加されたモジュールそのものにモジュールが含まれている場合、これらモジュールは再帰的に SPS へ追加されます。これにより、モジュールを広範囲なデザイン構造や複数のレベルへ展開することが可能になります。

モジュール SPS を作成する

モジュール化された SPS を作成するには、メイン SPS へ目的の SPS をモジュールとして追加 します。追加されたモジュール内部にある全ての JavaScript 関数、グローバルテンプレート、デザインフラグメント、そして XPath 関数が、参照元の SPS にて利用可能となります。これらオブジェクトはデザインツリーにて表示され、チェックボックスへチェックを入れる/外すことで、個々のオブジェクトを有効/無効にすることができます(以下のスクリーンショットを参照)。



これらのオブジェクトはそれぞれのメカニズムに従うかたちで、参照元の SPS にて再利用することができます。グローバルテンプレートは通常有効化するだけで参照元の SPS に適用することができます。デザインフラグメントは、デザインツリーから目的の場所へドラッグする必要があります。JavaScript 関数は、選択されたデザインコンポーネントに対してイベントハンドラーとしてプロパティウィンドウを介して割り当てることができます。(有効化された)XPath 関数は XPath 条件式にて使用することができます。

(C) 2015-2021 Altova GmbH

SPS モジュールの作成と使用方法については、モジュールSPS の作成のセクションを参照ください。

用語解説

SPS が他のモジュール内部で使用されると、後者のモジュールへの追加が行われます。2つのSPS は、それぞれ**追加された SPS モジュール**と、**参照元 SPS モジュール**と呼ばれます。SPS モジュールが追加されると、そのオブジェクトが参照元のSPS モジュールへ追加されます。これらのオブジェクトはモジュールオブジェクトと呼ばれ、以下の種類が存在します: グローバルテンプレート、デザインフラグメント、JavaScript 関数、ページレイアウトアイテム。

関連項目

- デザイン概要
- スキーマソース
- テンプレートとデザインフラグメント

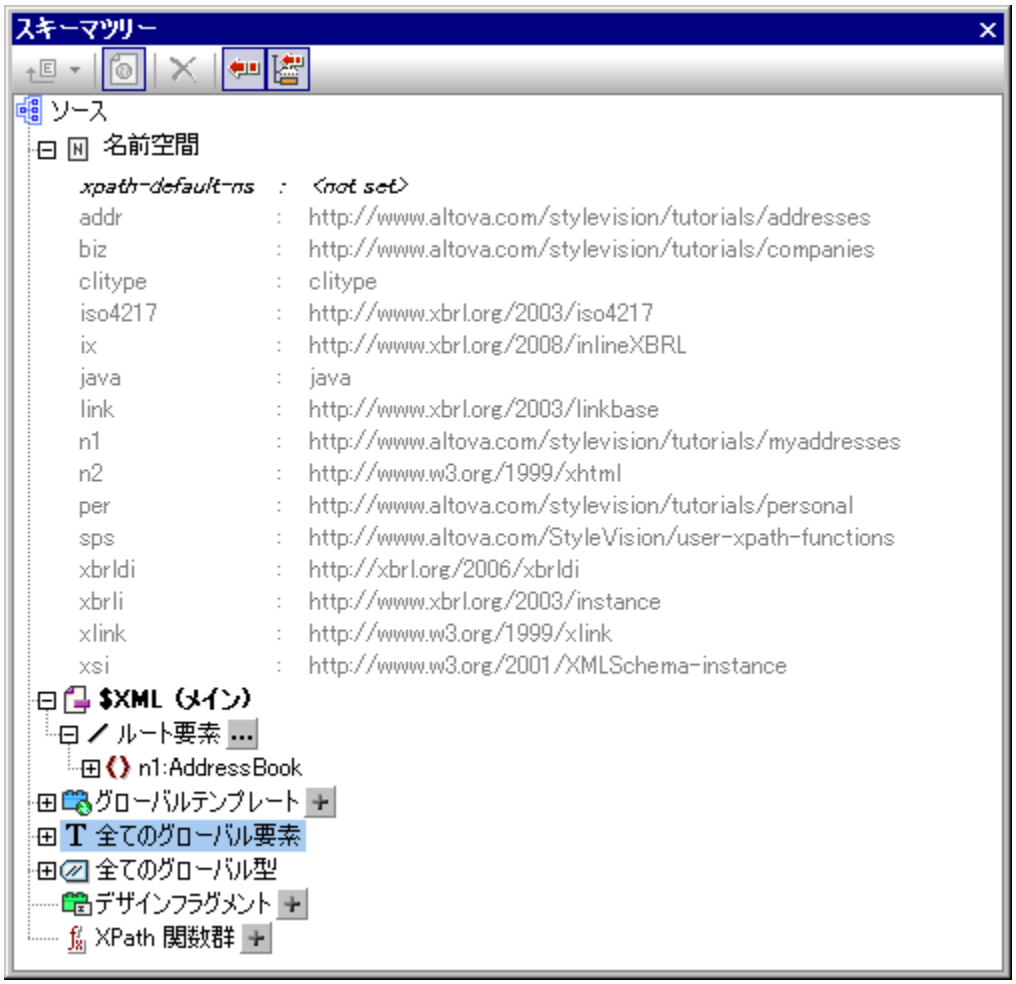
6.3.1 利用可能なモジュールオブジェクト

このセクションでは、追加されたSPS モジュールのうち参照元のSPS モジュールから利用することができるオブジェクトを紹介します。SPS モジュールにおいて、各オブジェクトをどのように利用することができるのか、またその使用方法について説明します。モジュールSPS の作成方法については、次セクションのモジュールSPS の作成を参照ください。セクションの終わりには、参照元のSPS モジュールにて利用することのできないオブジェクトが示されており、モジュールSPS がどのように動作するのか更に理解を深めることができます。

- 名前空間の宣言
- グローバルテンプレート
- デザインフラグメント
- 追加されたモジュール
- スクリプト
- CSS スタイル
- ページレイアウト
- 利用が不可能なモジュールオブジェクト

名前空間の宣言

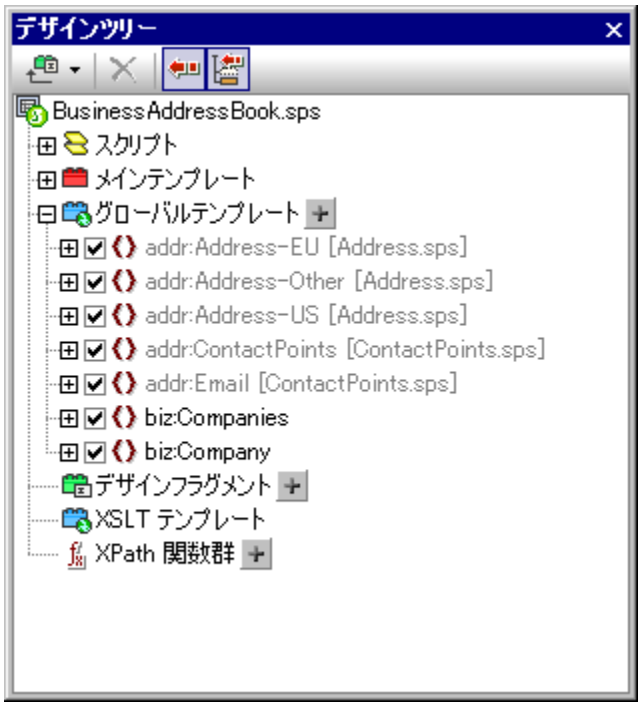
各 SPS には、名前空間 URI とそのプレフィックスのリストが保管されます。SPS がモジュールに追加されると、その中にある名前空間が参照元のSPS にあるスキーマソース内の名前空間と比較されます。追加された SPS 内にある名前空間 URI が、参照元 SPS のスキーマソース内にある名前空間 URI にマッチする場合、参照元 SPS のスキーマソースに使用されているプレフィックスが、追加された SPS の名前空間プレフィックスとして使用されます。追加された SPS 内にある名前空間 URI が、参照元 SPS のスキーマソース内にある名前空間 URI のどれともマッチしない場合、エラーメッセージが表示されます。



上のスクリーンショットでは、あるSPS 内にある様々な名前空間がプレフィックスとともに、スキーマツリーサイドバーに表示されます。これら名前空間の値はソーススキーマから得られるため、編集することはできません。

グローバルテンプレート

追加されたSPS モジュールの グローバルテンプレート は参照元のSPS モジュールにて使用することができ、デザインツリーサイドバーに表示されます(以下のスクリーンショットを参照)。デフォルトでは、追加されたモジュールに表示されているチェックボックスの状態を切り替えることにより、これらのモジュールを有効化/無効化することができます。グローバルテンプレートを作成して、追加されたモジュールから得られたグローバルテンプレートをオーバーライドする場合、グローバルテンプレートエントリーの隣にある + アイコンをクリックしてください。新しいグローバルテンプレートを追加ダイアログが表示されるので、グローバルテンプレートを作成する要素または属性を選択してください。スキーマ内にある目的のノードを選択するXPath 条件式を入力することもできます。「OK」をクリックすることで、新たなグローバルテンプレートと追加されたモジュール内部のグローバルテンプレートのどちらを有効化するのかを選択するように促されます。選択結果により新たに作成されたグローバルテンプレート、または追加されたモジュール内にあるグローバルテンプレートが有効化されます。2つあるグローバルテンプレートのチェックボックスを切り替えることにより、選択の切り替えを行うことができます。



追加されたモジュールのメインテンプレートは使用できない点に注意してください。つまりモジュール化によりテンプレートを再利用する場合、グローバルテンプレートとしてテンプレートを作成する必要があります。特定の要素に対してグローバルテンプレートが定義されていない状況でその要素に対する処理が行われた場合、その要素に対するデフォルトの処理(XSLT の内蔵テンプレート)が使用されます。

デザインフラグメント

追加された SPS 内にあるデザインフラグメント は、参照元の SPS にて利用することができ、SPS は、デザインツリー サイドバーに表示されます(上のスクリーンショットを参照)。デザイン内部にデザインフラグメントを挿入する際には、デザイン内の適切なコンテキスト ノード内へデザインフラグメントが配置されるように注意する必要があります。

追加されたモジュール

追加された SPS モジュール内部に別のモジュールが追加されている場合、それらのモジュールも参照元の SPS から利用することができます。これにより、あるモジュールを追加することで、そのモジュールに追加された全てのモジュールを再帰的に利用することが可能になります。勿論これらのモジュールにより構成されるコンテンツモデルは、参照元 SPS モジュールのソーススキーマに対して妥当なものでなければなりません。モジュールの確認と管理は、デザイン概要サイドバーで行うことができます。

スクリプト

追加された全ての SPS モジュールにあるスクリプトは参照元の SPS にて利用することができ、これらのスクリプトは デザインツリー サイドバー に表示されます。モジュールに追加されたスクリプトはライブラリに集められ、参照元の SPS からプロパティダイアログを介して利用することができるようになります。

CSS スタイル

SPS モジュールに追加されたグローバルスタイルは、グローバルスタイルとして参照元の SPS へ渡され、スタイルレポジトリサイドバー内にルールならびにスタイルルールが表示されます。CSS ファイルは デザイン概要サイドバーにも表示されます。同様に、SPS モジュールへ追加された外部 CSS ファイルも、参照元の SPS にて利用することができます。

ページレイアウト

追加されたモジュールのページレイアウトは参照元のSPS から使用することができ、デザインツリー サイドバーにて表示されます。

参照元のSPS から利用することのできないモジュールオブジェクトオブジェクト

追加されたモジュール内にある以下のオブジェクトを参照元の SPS から利用することはできません:

- *パラメーター定義* : 無視されます。
- *スキーマソース* : 追加されたSPS がベースとしているスキーマソースは無視されます。追加されたSPS 内にあるドキュメント要素のコンテンツモデルは、参照元 SPS のコンテンツモデル内に含める必要があり、そうでない場合、追加されたSPS をモジュールとして正しく使用できないという点に注意してください。必要な場合には、いつでもユーザー定義スキーマを参照元のSPS へ加えることができます。新たなスキーマには、追加されたグローバルテンプレートのコンテンツモデルを含めることができます。
- *作業用 XML ファイルならびにテンプレート XML ファイル* : これらのファイルに対する参照が無視されます。参照元のSPS では、それ自身に与えられた作業用 XML ならびにテンプレート XML ファイルが使用されます。
- *XPath デフォルト名前空間* : これらの値がインポートされたモジュールにてセットされている場合、参照元のSPS にこれらの値が渡されることはありません。

関連項目

- デザイン概要
- モジュールSPS の作成
- スキーマソース
- テンプレートとデザインフラグメント
- スクリプトの使用

6.3.2 SPS の作成

モジュールSPS の作成は、大きく分けて以下のステップにより行われます:

1. 追加するSPS モジュールのデザインと保存。
2. 使用される(つまり参照元の)SPS に対してモジュールを追加。
3. 必要に応じて追加されたオブジェクトを有効/無効にする。
4. 必要に応じて適切なオブジェクトを適用。

追加されるSPS モジュール

他のモジュールへ追加されるSPS を作成する際には、以下の2点に留意する必要があります:

1. 参照元のSPS モジュールにて使用することのできるテンプレートは、グローバルテンプレートだけです。つまり追加されるSPS モジュールにて再利用したいテンプレートは、グローバルテンプレートとして作成する必要があります。
2. 追加される SPS モジュール内で定義されたドキュメント構造は、参照元のSPS のノーススキーマ. により定義されるコンテンツモデル内部で妥当でなければなりません。追加されたテンプレートがSPS のメインスキーマにて定義されたコンテンツモデルに含まれていない場合でも、ユーザー定義スキーマ内で定義することができます。

追加されるSPS モジュールを作成する際には、SPS がベースとしているスキーマは、以下のどれかとなります:

- 参照元 SPS のメインソーススキーマ。この場合、SPS が追加された際に追加されるグローバルテンプレートが、参照元 SPS のメインスキーマにおけるコンテンツモデルの一部となります。そのため、Authentic View におけるこれらグローバルテンプレートの出力は編集可能なものとなります。
- 参照元 SPS のメインスキーマにより定義されたコンテンツモデルの一部となるコンテンツモデルを定義したスキーマ。この場合、グローバルテンプレートが追加されると、参照元 SPS のメインスキーマのコンテンツモデルに対してテンプレートが追加されます。Authentic View におけるこれらグローバルテンプレートの出力は編集可能なものとなります。
- 参照元 SPS のメインスキーマにより定義されるコンテンツモデルの一部とはならないコンテンツモデルを定義したスキーマ。この SPS モジュールが追加されても、参照元 SPS のスキーマにおけるコンテンツモデルの一部となることはありません。それでもユーザー定義スキーマにより、グローバルテンプレートのコンテンツモデルを含むコンテンツモデルが定義される場合、出力の生成に使用することができます。Authentic View におけるこれらグローバルテンプレートの出力は編集可能なものとはなりません。

使用しているスキーマにてコンテンツモデルを定義する場合、使用される名前空間によりノードの展開名が決定されるため、慎重に操作を行う必要があります。

作業用 XML ファイルを使用することで、追加される SPS モジュールの出力をテストすることができます。この作業用 XML ファイルへの参照は、参照元の SPS にて無視されます。

SPS モジュールを追加する

SPS へモジュールを追加するには デザイン概要 にて新たなモジュールを追加コマンドをクリックします（以下のスクリーンショットを参照）。表示されるダイアログにて目的の SPS ファイルを選択し、開くをクリックします。

モジュールが SPS に追加され、デザイン概要のモジュール以下に表示されます。追加された全てのモジュールオブジェクトがデザインツリーサイドバーに表示されるとともに、追加された CSS ファイルはデザイン概要サイドバーにも表示されます。追加されたモジュールそのものが別のモジュールを参照している場合、それら参照されたモジュールも間接的にインポートされ、モジュール以下に灰色で表示されます。間接的にイン

ポートされたモジュールが、どのモジュールによりインポートされたものなのかは、その(間接的にインポートされた)モジュールへマウスオーバーすることで確認することができます。

追加されたモジュールや間接的にインポートされたモジュールを StyleVision で開くには、そのモジュールを右クリックして、コンテキストメニューからモジュールを開くを選択してください。

追加されたモジュールの順序

モジュールが追加される順序と表示される順序は、CSS スタイルの優先順位にとって重要な意味を持ちます。CSS のカスケード順序は、後の方(リストの下部)に定義されたモジュールが、それ以前(リストの上部)のモジュールに対して優先的に決定されます。参照元 SPS モジュール内にある CSS スタイルは、追加されたモジュール内にある全てのスタイルより優先的に使用されます。追加されたモジュールの位置を変更するには、そのモジュールをデザイン概要にて右クリックし、コンテキストメニューから上へまたは下へを選択します。

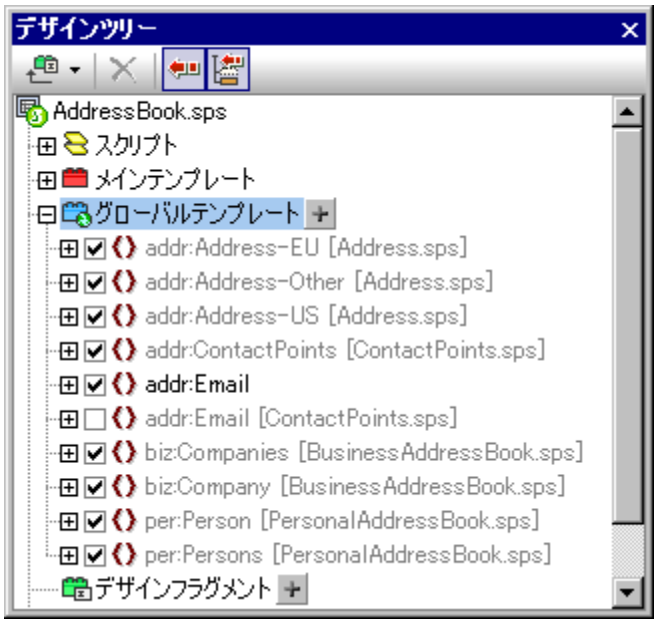
スクリプトやグローバルテンプレート、デザインフラグメント、そしてページレイアウトアイテム間における衝突の解決に、モジュールの順序は意味を持ちません。

ファイルの変更警告

参照元の SPS モジュールが開かれている状態で、追加されたファイル(SPS モジュール、スキーマや作業用 XML ファイル)が変更されると、ファイルの変更ポップアップが表示され、参照元の SPS モジュールに変更を反映するか尋ねられます。

追加されたオブジェクトの有効化と無効化

追加された全てのモジュール内にある全てのモジュールオブジェクトは、(直接的、または間接的に追加されたかにかかわらず)参照元の SPS へ追加され、デザインツリー内の対応する箇所(スクリプト、グローバルテンプレート、デザインフラグメント、XSLT テンプレート、XPath 関数)以下に表示されます。これらオブジェクトの隣にはチェックボックスが表示されており、チェックボックスを有効/無効にすることで、そのオブジェクトを有効/無効にすることができます。オブジェクトを無効にすることで、そのオブジェクトを SPS から実質的に削除することができます。



上のスクリーンショットでは、AddressBook.sps モジュールにて使用されている全てのグローバルテンプレートがグローバルテンプレート以下に表示されています。他のモジュールを介して追加されたモジュールは(それが直接的であれ間接的であれ)グレー色で表示されます。AddressBook.sps 内で直接作成されたオブジェクトは黒色で表示されます。スクリーンショットでは、addr:Email というグローバルテンプレートだけが AddressBook.sps 内に作成され、その他のグローバルテンプレートは全て別のモジュールを介して追加され、オブジェクト名の隣に定義されたファイルの名前が表示されます。

(C) 2015-2021 Altova GmbH

addr:Email という名前のグローバルテンプレートが2つあることに注目してください。1つは参照元のSPS (AddressBook.sps) にて作成されたもので、もう1つは追加されたモジュールのContactPoints.sps にて作成されたものです。複数のグローバルテンプレートが、同じ名前の(名前空間)展開名を持つ場合、そのうちの1つだけを同時に有効にすることができます。チェックボックスにより、有効にするものを選択することができます(その他にも、デザインビューのコンテキストメニューから、グローバルテンプレートを有効にすることもできます)。このメカニズムにより、(i) 参照元 SPS モジュールにて作成したグローバルテンプレートのオーバーライドや、(ii) ある要素に対して適用されるグローバルテンプレートが複数の追加モジュールにて定義されている状況での解決を行う際に便利な機能となります。

現在のSPS にて定義されたグローバルテンプレートは、選択した後に**削除**ボタンをクリックする事で削除することができますが、追加されたモジュールにて定義されたグローバルテンプレートを参照元のSPS にて削除することはできません。削除を行うには、追加されたSPS を開き、そこでグローバルテンプレートを削除する必要があります。

個々のスクリプトならびにデザインフラグメントも、同様の方法で有効化/無効化することができます。

モジュールオブジェクトの適用と使用

参照元のSPS モジュールにて、通常の方法によりテンプレートのデザインを行います。追加されたオブジェクトは、その種類により異なる方法で適用されます。勿論、適用する各モジュールオブジェクトを有効にする必要があります。

グローバルテンプレート
追加されたSPS モジュールからグローバルテンプレート を使用するには、そのグローバルテンプレートが本当に適用されたかどうか確認する必要があります。作成中のデザインにより、以下にある方法のどれかで確認を行うことができます:

- メインテンプレートにて、要素テンプレートがグローバルテンプレートを使用するように指定するか、グローバルテンプレートをローカルにコピーします。デザインにて要素タグを右クリックすることで表示されるコンテキストメニューにて、これらのコマンドを利用することができます。
- メインテンプレートでは、コンテンツやその他のコンテンツプレースホルダによりテンプレートが適用され、関連するグローバルテンプレートが処理されます。

デザインフラグメント
デザインフラグメントを使用するには、デザインツリーからメインテンプレートやグローバルテンプレートテンプレート内にある目的の場所へ、デザインフラグメントをドラッグします。デザインフラグメントが、適切なコンテキストノードへドロップされるようにしてください。詳細については、次を参照してください: デザインフラグメント。

スクリプト
デザインフラグメントを使用するには、デザインツリーからメインテンプレートやグローバルテンプレートテンプレート内にある目的の場所へ、デザインフラグメントをドラッグします。デザインフラグメントが、適切なコンテキストノードへドロップされるようにしてください。詳細については プロパティサイドバーから 特定のイベントに対してセットすることができます.

関連項目

- デザイン概要
- 利用が可能なモジュールオブジェクト
- サンプル: Address Book
- スキーマソース
- テンプレートとデザインフラグメント
- スクリプトの使用

6.3.3 サンプル: Address Book

(マイ) ドキュメントフォルダー 以下にある C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\ModularSPS にはモジュールSPS のサンプルが収められています。このフォルダー内にあるサンプルファイルにより、ビジネス用と個人用の連絡先がモジュール化されているアドレス帳のプロジェクトファイルが構成されます。サンプルにより、モジュール化がどのように実装されるか確認することができるとともに、モジュール化を行う利点を知ることもできます。

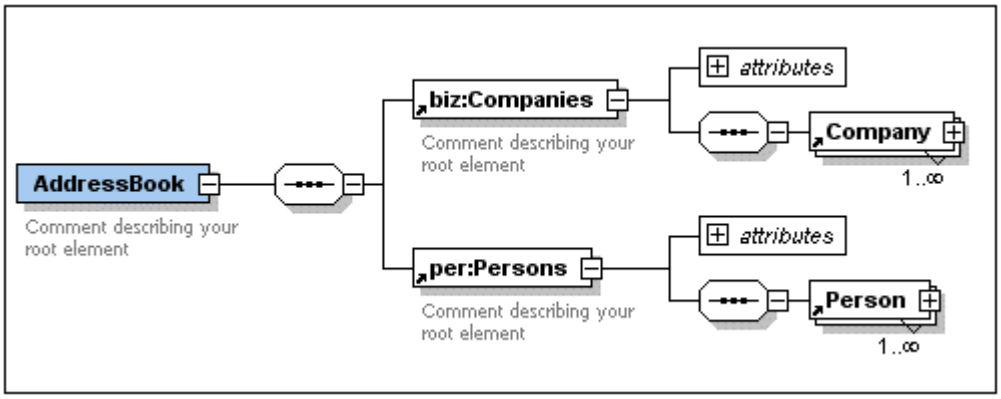
- アドレス帳は、(i) ビジネス用の連絡先と、(ii) 個人用の連絡先という2つのモジュールから構成され、両方とも独立した SPS によりデザインされます。2つのモジュールを合わせることで、合成されたアドレス帳が作られます。ここでは複数のモジュールにより、より大きなデータが構成されます。
- 各モジュール(ビジネス用と個人用の連絡先)のコンテンツモデルはそれぞれ多少違っていますが、住所や電話番号、FAX、電子メールといった情報から構成される ContactPoints モジュールという共通のモジュールを含んでいます。従って、ContactPoints モジュールは2つの連絡先(ビジネス用と個人用)にて共有することができます。モジュール化により、1つのモジュールをその他複数のユニットにおける共通部分として使用することができます。
- 更に、ContactPoints モジュールにより、柔軟性を向上させることができます。サンプルプロジェクトでは、EU (欧州)の住所、US (米国)の住所、またはその他地域の住所に対応した、3つのコンテンツモデルを含む独立した Address モジュールが作成されています。これら3つのコンテンツモデルに対する出力は単一の SPS にて決定されていますが、それぞれの SPS にて細かな定義を行うことも可能です。この場合、モジュールの再利用を簡単に行うことができるため、モジュールを柔軟に使用することが可能になります。

このプロジェクトに関する説明は、以下のようにまとめられています:

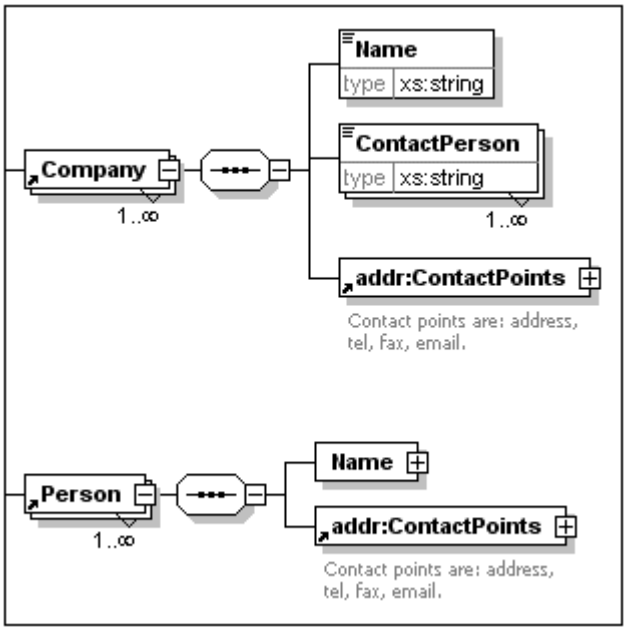
- スキーマファイル
- XML データソース
- SPS ファイル

スキーマファイル

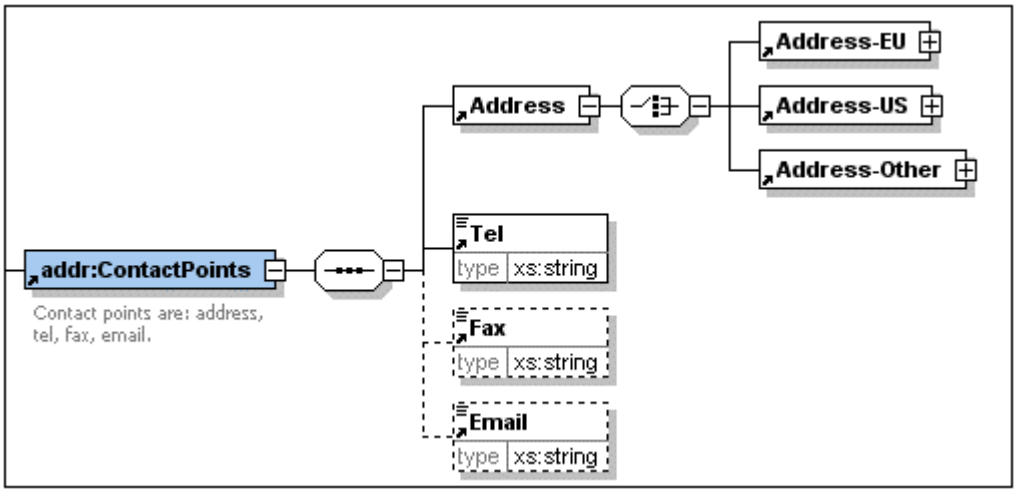
モジュールSPS に対するスキーマを作成する場合、再利用する要素をグローバル要素として作成しなければならない点に留意してください。アドレス帳のスキーマは AddressBook.xsd で、ビジネス用のアドレス帳 (BusinessAddressBook.xsd) と個人用のアドレス帳 (PersonalAddressBook.xsd) をインポートすることで構成されています。BusinessAddressBook.xsd スキーマにより会社用のコンテンツモデルが与えられ、PersonalAddressBook.xsd スキーマにより個人用のコンテンツモデルが与えられます(以下のスクリーンショットを参照)。



両方のスキーマにて連絡先詳細のコンテンツモデルが定義されている ContactPoints.xsd スキーマがインポートされます。



ContactPoints.xsd スキーマにはEU、US、そしてその他（Other）の地域に対応したアドレス帳のコンテンツモデルが定義された Address.xsd スキーマが含まれています（以下のスクリーンショットを参照）。



対象となるスキーマが別の名前空間に属している場合、スキーマが「インポート」されます。対象となるスキーマが同じ名前空間に属している場合、スキーマは「インクルード」されます。

メモ 上に示されるスキーマのスクリーンショットは、Altova XMLSpy のスキーマビューです。

XML データソース

XML データは AddressBook.xml ファイル内に含まれています。このファイルでは、AddressBook 要素内に companies と persons 要素が子要素として含まれています。これら2つの要素のコンテンツモデルは、それぞれ BusinessAddressBook.xsd と PersonalAddressBook.xsd スキーマファイルにて定義されています。

BusinessAddressBook.xsd と PersonalAddressBook.xsd スキーマにそれぞれ対応する XML データファイルも存在します。これら2つの XML ファイル（BusinessAddressBook.xml ならびに PersonalAddressBook.xml）は、対応する SPS ファイルの作業用 XML ファイルとして使用されます。

SPS モジュールと作業用 XML ファイルの対応を以下に示します:

- `AddressBook.xml => AddressBook.sps, ContactPoints.sps, Address.sps`
- `BusinessAddressBook.xml => BusinessAddressBook.sps`
- `PersonalAddressBook.xml => PersonalAddressBook.sps`

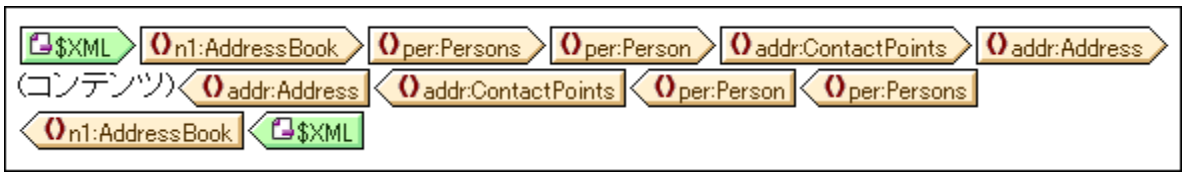
SPS モジュール

最も基本的なモジュール (Address.sps) を最初に説明し、アドレス帳 (AddressBook.sps) を完成させるまでのステップを記述します。全てのSPS モジュールでは AddressBook.xsd をスキーマとして使用しています。

Address.sps

スキーマと作業用 XML ファイルの使用に関するポイントを以下に記します。

- Address.sps では AddressBook.xsd がスキーマとして使用されます。しかし、Address.xsd、ContactPoints.xsd、BusinessAddressBook.xsd、PersonalAddressBook.xsd の全てに Address 要素がグローバル要素として含まれているため、これらスキーマのどれかを使用することもできます。SPS モジュールが別の SPS モジュール内に追加されると、インポートされたモジュールのスキーマが無視されるため、SPS がモジュールとして追加される際にどのスキーマが使用されたかは重要ではなくなります。
- AddressBook.xml が作業用 XML ファイルになります。Address.sps 内にあるメインテンプレートでは、Address 要素だけが処理されるように指定されているということと、Address-EU、Address-US、そして Address-Other のグローバルテンプレートが定義されているだけである点に注目してください。



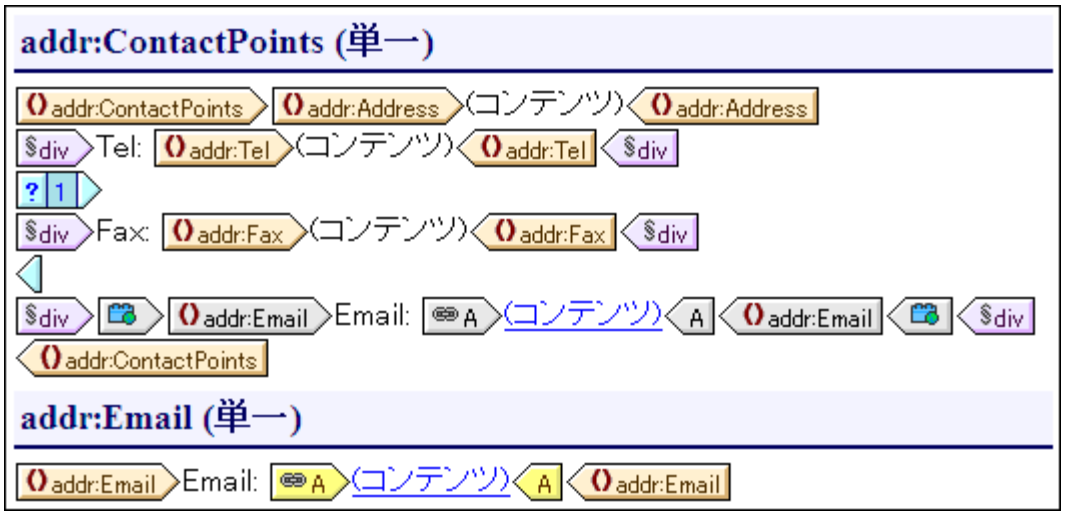
Address 要素だけが処理されるため、出力プレビューにて Address の出力だけが表示されます。Address.sps がモジュールとして使用される場合、グローバルテンプレートが追加され、メインテンプレートが無視されます。

ContactPoints.sps

このSPS では1つのモジュールがインポートされます。グローバルテンプレートが他のグローバルテンプレートやメインテンプレート内部で使用されている点に注目してください。

- ContactPoints.sps では AddressBook.xsd がスキーマとして使用され、AddressBook.xml が作業用 XML ファイルとして使用されます。
- Address.sps がモジュールとして追加されるため、Address-EU、Address-US、Address-Other 要素がグローバルテンプレートとして利用可能になります。
- ContactPoints と Email 要素のグローバルテンプレートが定義されます。ContactPoints の定義に Email のグローバルテンプレートが使用されている点に注目してください(以下のスクリーンショットを参照)。

(C) 2015-2021 Altova GmbH



- プレビューに必要なメインテンプレートでは ContactPoints 要素のグローバルテンプレートが使用されているため、ContactPoints 出力のプレビューが有効になっています。

BusinessAddressBook.sps* と *PersonalAddressBook.sps
プレビューに必要なメインテンプレートでは ContactPoints 要素のグローバルテンプレートが使用されているため、ContactPoints 出力のプレビューが有効になっています。

- これら2つのモジュールでは AddressBook.xsd がスキーマとして使用されます。作業用 XML ファイルには、BusinessAddressBook.xml と PersonalAddressBook.xml がそれぞれ使用されます。
- ContactPoints.sps がモジュールとして使用されます。これにより Address.sps が間接的にインポートされます。これら2つのモジュールにある全てのグローバルテンプレートが参照元の SPS モジュールにて利用可能になります。
- BusinessAddressBook.sps グローバルテンプレートが Companies ならびに Company 要素に対して定義されます。Company の定義ではグローバルテンプレートの ContactPoints が使用されている点に注目してください。
- PersonalAddressBook.sps では、グローバルテンプレートが Person ならびに Persons 要素に対して定義されます。Person の定義ではグローバルテンプレートの ContactPoints が使用されている点に注目してください。

AddressBook.sps
Email 要素に対して2つのグローバルテンプレートが表示されており、そのどちらかを有効にすることができます。

- AddressBook.sps では AddressBook.xsd がスキーマとして使用されます。作業用 XML ファイルには AddressBook.xml が使用されます。
- BusinessAddressBook.sps と PersonalAddressBook.sps がモジュールとして追加され、その結果 ContactPoints.sps と Address.sps が間接的にインポートされます。
- グローバルテンプレートが Email 要素に対して定義されます。つまり Email に対して、ContactPoints.sps と AddressBook.sps から得られた2つのグローバルテンプレートが存在します（以下のスクリーンショットを参照）。

- デザインツリーのグローバルテンプレートでは、どのグローバルテンプレートをアクティブにするか選択することができます。StyleVision で同時に使用することのできる同名のグローバルテンプレートは1つだけで、どちらがアクティブになろうとも、ContactPoints グローバルテンプレートにて使用されます。

- メインテンプレートには、出力のヘッダーに使用される静的コンテンツが含まれています。

関連項目

- デザイン概要
- 利用が可能なモジュールオブジェクト
- モジュールSPS の作成
- スキーマソース
- テンプレートとデザインフラグメント
- スクリプトの使用

(C) 2015-2021 Altova GmbH

6.4 テンプレートと デザインフラグメント

デザインドキュメントはテンプレートにより構成されるため、使用することができるテンプレートの種類を理解することが重要になります。

- *メインテンプレートとグローバルテンプレート*： デザインドキュメントは1つの メインのテンプレート と、オプションとなっている1つ以上 グローバルテンプレートにより構成されます。グローバルテンプレートはメインテンプレートから参照することができます。
- *ノードテンプレート*： メインテンプレートとグローバルテンプレートを構成するテンプレートのことで、ノードテンプレーはスキーマソース内にある ノードテンプレート にマッチします。
- *デザインフラグメント*： 個別に作成され、デザイン内にあるメインテンプレートまたはグローバルテンプレート内で再利用ことができるテンプレートです。

このセクションでは、デザインの構造内部にてテンプレートやデザインフラグメントが果たす役割について説明します。ここで重要になるのはデザイン内の表示プロパティではなく、その構造です。このセクションでは、表紙やヘッダー、フッターといった印刷媒体に対する構造アイテムについても説明されることはありません。これらのアイテムについては、印刷出力のデザイン内にあるセクションを参照ください。

メモ デザインビューでは、メインテンプレート、グローバルテンプレート、ページレイアウト テンプレート、とデザインフラグメントと、複数のテンプレートを、SPS に含めることができます。ツールバーアイコンとして用意されているテンプレート デザイン フィルター機能を使用することで、デザインビューに表示されているテンプレートを管理することができます。

関連項目

- デザインの作成
- スキーマソース
- テンプレートフィルター

6.4.1 メインのテンプレート

メインのテンプレート により、出力の構造が決定されます。つまり、デザインにあるメインテンプレート内の順序が、そのまま Authentic View と出力における表示順序となります。プログラミング用語を使えば手続き型の処理となり、テンプレートの開始点から処理が開始され、最後まで順序どおり処理が行われます。その中で XML ドキュメントから得られたノードが処理されます。これらノードが処理されるテンプレートはローカルテンプレートと呼ばれます。あるローカルテンプレートにおける処理が終わると、メインテンプレート内にある次のコンポーネントへ処理が移ります。ノードの処理を行うために、ノードから グローバルテンプレート への参照が行われることもあります。この場合、そのノードに対してグローバルテンプレートが実行された後に呼び出し元のメインテンプレートへ処理が戻り、次のコンポーネントへ処理が移ります。

メインテンプレートのエントリーポイントはスキーマのドキュメントノード となります。StyleVision ではオプションとして複数のルート要素 (ドキュメント要素) を選択することもでき、メインテンプレート内部で、アクティブな各ドキュメント要素に対して ローカルテンプレートをセットすることができます。処理中に実行されるテンプレートは、処理されている XML インスタンスドキュメントのドキュメント要素に対して定義されたテンプレートとなります。

関連項目

- デザインの作成
- テキストとして XML コンテンツを挿入
- グローバルテンプレート
- デザインフラグメント

- テンプレートフィルター

6.4.2 グローバルテンプレート

グローバルテンプレート は、任意のノードやスキーマ内の型、またはXPath 条件式により指定されたノードに対して定義することができます。

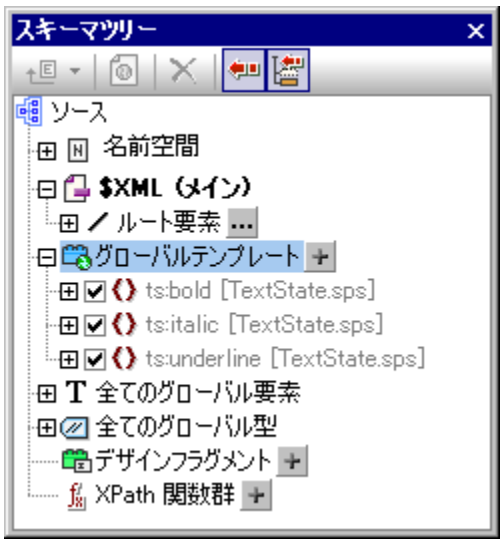
グローバルテンプレートは選択されたノードや型の処理命令に対して適用され、メインのテンプレート、デザインフラグメント、またはその他のグローバルテンプレートから呼び出されます。あるテンプレートが定義され、複数回呼び出されるというのは宣言型の動作ですが、手続き型の処理でも使用されます。このようにして、ある定義を複数回再利用することが可能になります。グローバルテンプレートの呼び出しは、以下のような状況で発生します:

- メインのテンプレート 内にあるノードや型が、(デザイン内のコンポーネントを右クリックし、グローバルテンプレートを選択して)グローバルテンプレートを参照するようにセットされている。
- (コンテンツ) または(その他のコンテンツ) がローカルテンプレート 内にある要素や型内に挿入され、その要素に対するその他のコンテンツや型には、グローバルテンプレート のノードや型が含まれている。

様々な要素や複数の場所でノード(または型)が出現する、または特定の処理命令が全ての出現に対して必要となるような状況で、グローバルテンプレートは便利な機能となります。例えば、chapter、section、appendix、またはblockquote 要素内にあるかどうかにかかわらず、para 要素の書式が常に同じものになるようにすることを考えてみましょう。para 要素に対してグローバルテンプレートを定義して、メインのテンプレート にて para 要素に対するグローバルテンプレートが適用されるように指定することができます。例えば、//chapter/para をメインテンプレート内に追加して、para がグローバルテンプレートを参照するようにし、//chapter/title を追加した後に(コンテンツ) または(その他のコンテンツ) を追加することで、chapter 要素内にあるその他のコンテンツが、利用可能なグローバルテンプレートやデフォルトテンプレートにより処理されるようになります。更に、グローバルテンプレートは複合型(例: アドレスモデルの定義)や単純型(例: xs:decimal)に対しても定義することができます。この場合、その型(複合型または単純型)により定義された出現のうち、グローバルテンプレートを呼び出す全ての出現において、グローバルテンプレートのルールに従った処理が行われます。.

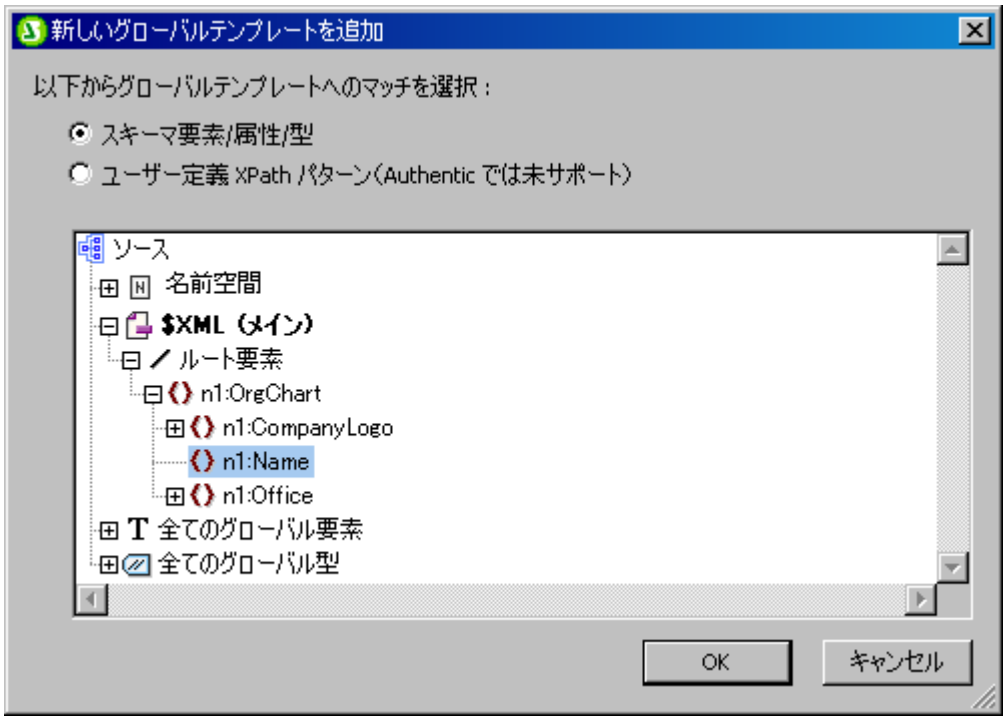
グローバルテンプレートの作成

グローバルテンプレートは、スキーマ内にある任意のノードや型、または XPath 条件式にて指定されたノードに対して作成することができ、スキーマツリーサイドバーにて作成されます(以下のスクリーンショットを参照)。



グローバルテンプレートは以下にあるいずれかの方法により作成することができます:

- スキーマツリーに表示されているグローバルテンプレートアイテムの右側にある**新しいグローバルテンプレートを追加**アイコンをクリックします（上のスクリーンショットを参照）。新しいグローバルテンプレートを追加ダイアログが表示されます（以下のスクリーンショットを参照）。ダイアログに表示されているスキーマツリーから要素や属性、または型を選択することができるほか、XPath 条件式を入力することもできます。XPath により選択されたノードに対して作成されたグローバルテンプレートは Authentic でサポートされない点に注意してください。この操作により、グローバルテンプレートとして作成されなければならないノードが決定されます。「OK」をクリックするとテンプレートが作成され、デザインビューにすでに存在するテンプレートに追加され、編集することができるようになります。スキーマツリーでは、スキーマノードや型アイコンにプラスサインが表示されます。



- スキーマツリーのルート要素以下にあるスキーマノードや型コンポーネントを右クリックして、**グローバルテンプレートの作成/削除**コマンドを選択します。テンプレートが作成され、デザインビューに既に表示されている既存のテンプレート以下に追加されます。スキーマツリーでは、ノードや型のアイコンにプラスサインが表示されます。

- デザインビューのメインテンプレートにあるテンプレートからグローバルテンプレートを作成することもできます。デザインビューやスキーマツリーサイドバーにあるテンプレートを右クリックして、**グローバルテンプレートに設定**を選択します。選択されたテンプレートからグローバルテンプレートが作成され、メインテンプレート内にあるテンプレートが、このグローバルテンプレートを使用するように自動的に定義されます（グローバルテンプレートの使用方法については、下記を参照ください）。

グローバルテンプレートは、メインのテンプレートの下のデザインビュー内に存在します。グローバルテンプレートのために作成されたノード名を含むバーを操作して表示されます。：*（単一）*または*（複合型）*。グローバルテンプレートは、下のスクリーンショットに表示されています。

グローバルテンプレートの処理は、ユーザー定義で、静的と動的であることができます。また、メインのテンプレートの処理のために使用することができます。

グローバルテンプレートの使用

作成したグローバルテンプレートは、同じ修飾名を持ったノードがドキュメントへ挿入された際に使用することができます。ノードがデザイン上にドロップされると表示されるメニューにて、**グローバルテンプレートを使用**を選択してください。その他にも、ローカルテンプレートがデザイン内にあり、同じ修飾名を持ったノードに対してグローバルテンプレートが存在する場合、ローカルテンプレートの代わりにグローバルテンプレートを使用することができます。ローカルテンプレートに対してグローバルテンプレートを使用するには、デザインビューにあるローカルテンプレートを右クリックして、**グローバルテンプレートを使用**コマンドを選択します。グローバルテンプレートが使用される場合、内部に含まれる処理命令が呼ばれ、ランタイムにてローカルテンプレートにより使用されます。

グローバルテンプレートがデザインにて使用される場合、グローバルテンプレートにて XPath のパターンを作成し、ノードセットのフィルタリングを行うことができます。フィルターを作成するには、デザインビューにてグローバルテンプレートのタグを右クリックして、コンテキストメニューから XPath フィルターの編集 を選択します。XPath 式 の編集ダイアログ が表示され、目的の条件式を入力することができるようになります。

再帰的なグローバルテンプレート

グローバルテンプレートは再帰的に作成することも可能で、グローバルテンプレートが自分自身を呼び出すように定義することができます。Authentic View における無限ループを回避するために、呼び出しの深さを制限するためのプロパティに値をセットすることができます。SPS のプロパティダイアログ(「ファイル| プロパティ」)にある Authentic タブ内の**テンプレート呼出し階層上限**プロパティにて値をセットすることができ、この値により、Authentic View 出力を処理する際に再帰的なテンプレート呼び出しの最大値が指定されます。テンプレート呼出し階層上限プロパティで指定された値を超えたテンプレートの呼び出しが行われた場合、エラーが返されます。

グローバルテンプレーをローカルにコピーする

グローバルテンプレートが作成されると、その処理命令をメインテンプレート内にある同名のテンプレートへ直接コピーすることができます。ローカルテンプレートを右クリックして、**グローバルテンプレートをローカルにコピー**を選択します。グローバルテンプレートをローカルにコピーするという処理は、処理命令が一度だけコピーされるという点で(ランタイムにおける)グローバルテンプレートの使用とは異なります。一旦ローカルにコピーされたら、グローバルテンプレートがローカルテンプレートに対して影響を及ぼすことはありません。グローバルテンプレートとローカルテンプレートのどちらか(または両方)をその後修正しても、相手に対して影響を与えることはありません。グローバルテンプレートがローカルテンプレートにより**使用**される場合、グローバルテンプレート側の変更が、ランタイムにてローカルテンプレートに反映されます。

グローバルテンプレートの有効化と無効化

スキーマツリーサイドバーに表示されているグローバルテンプレートのエントリーをチェックすることで、そのグローバルテンプレートが有効化されます。エントリーのチェックを外すことでテンプレートが無効化されます。グローバルテンプレートが有効化された場合、テンプレートが XSLT スタイルシートにて生成されます(グローバルテンプレートが作成されるとデフォルトで有効になります)。無効になっている場合、XSLT スタイルシートにて生成されることはありませんが、SPS デザインには保存されます。

無効になっているグローバルテンプレートを使用するローカルテンプレートでは、(存在しないグローバルテンプレートを参照することはできないため)XSLT のデフォルトテンプレートが使用され、コンテンツやテキストノードの子孫にもその影響は現れます。

有効化/無効化の機能を使用することで、(i) グローバルテンプレートが一時的に必要ない場合でも削除する必要は無く、必要になった時に再度有効化することができ、(ii) インポートされたスタイルシートにあるテンプレートにて名前の衝突が起こった際に、必要ないグローバルテンプレートを一時的に無効化することが可能になります。

グローバルテンプレートの削除

グローバルテンプレートを削除するには、削除するグローバルテンプレートをデザインビューまたはスキーマツリーサイドバーにて右クリックし、**グローバルテンプレートの作成/削除**コマンドを選択します。

(C) 2015-2021 Altova GmbH

単純型のグローバルテンプレートと複合型のグローバルテンプレート

グローバルテンプレートには、単純型と複合型の2種類があります。複合型グローバルテンプレートは下位互換性を保証するためのもので、バージョン 2006 より前の StyleVision にて作成された SPS のグローバルテンプレートにテーブルやリストが含まれている場合、StyleVision 2006 やそれ以降のバージョンでは、通常複合型として、そのグローバルテンプレートが開かれます。

複合型のグローバルテンプレートと単純型のグローバルテンプレートとの違いは、グローバルテンプレートに対して作成されたノードの処理方法にあります。ノードの最初のインスタンスがドキュメントにて見つかった場合、そのノードの直後にある全てのインスタンスが複合型のグローバルテンプレートにより処理されます。単純型のグローバルテンプレートの場合、個々のノードインスタンスが発見されるごとに処理が行われます。

グローバルテンプレートに対して作成された要素ノードの周りに、定義済みのフォーマット または改行が作成されている場合、単純型のグローバルテンプレートは、自動的に複合型のグローバルテンプレートへ変換されるという点に留意してください。これにより、複合型のグローバルテンプレートと同様の処理が行われます。単純型のグローバルテンプレートへ戻したい場合は、(定義済みのフォーマットの外側へノードをドラッグし、定義済みフォーマットを削除することで) 定義済みのフォーマット を削除するか、デザインツリー サイドバー内にあるアイテムを削除することで) 改行を削除する必要があります。単純型のグローバルテンプレートから複合型のグローバルテンプレートへの自動変換を回避するには、単純型のグローバルテンプレートに対して作成された要素のノードタグ内部に定義済みのフォーマット や改行を追加するようにしてください。

モジュール SPS におけるグローバルテンプレート

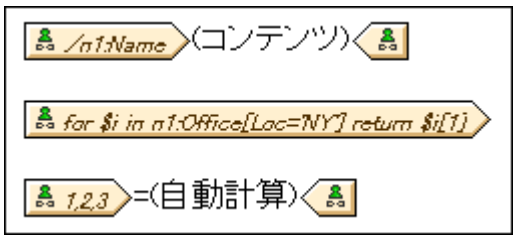
SPS モジュールが別の SPS モジュールへ追加された 場合、追加されたモジュール内にあるグローバルテンプレートを、参照元の SPS にて利用することができます。モジュール SPS の使用に関する詳細については、モジュール SPS のセクションを参照ください。

関連項目

- テキストとして XML コンテンツを挿入
- その他のコンテンツ
- メインのテンプレート
- デザインフラグメント
- テンプレートフィルター
- モジュール SPS

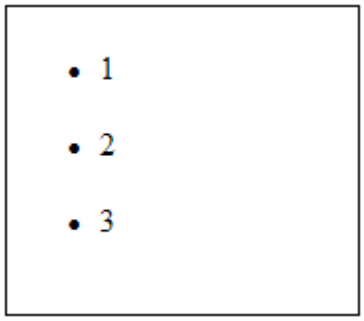
6.4.3 ユーザー定義テンプレート

ユーザー定義テンプレートとは、XPath 条件式により生成されるアイテムに対して適用されるテンプレートのことです。これらアイテムは原子型の値またはノードとなります。以下のスクリーンショットでは3つのユーザー定義テンプレートが示されており、タグの左側にユーザー定義テンプレートアイコンが示されます。ユーザー定義テンプレートが持つ柔軟性により、テンプレートを作成することができます。ユーザー定義テンプレートにより生成されたコンテンツは Authentic ビューで編集することはできない点に注意してください。



上に示されるこれら3つのユーザー定義テンプレートに対して定義された XPath 条件式により、以下の処理が行われます:

- ソーススキーマ内のノードを選択します。XPath 条件式を使用することで、全てのコンテキストノードから、任意のスキーマソース内にある任意のノードへアクセスすることができるようになります。曖昧性が無いかたちで StyleVision が定義されたノードを識別することができる場合、テンプレートはユーザー定義テンプレートから(Authentic View における編集が可能な) 通常のテンプレートへ自動的に変換されます。ユーザー定義テンプレートの場合、テンプレートタグの左側にユーザー定義テンプレートを表す緑色のアイコンが表示されます。
- XPath 2.0 とXPath 3.0 のfor コンストラクトにより指定された条件を満たすノードを選択します。このようなテンプレートは StyleVision に通常のテンプレート(ユーザー定義テンプレート) を明確に解決することを許可しません。
- 原子型の値のシーケンス(1、2、3)を選択します。原子型の値に対してテンプレートを作成することは可能ですが、このようなテンプレート内でコンテンツプレースホルダを使用することはできません。これは、(コンテンツプレースホルダにより生成される)xsl:apply-template 処理命令が、(原子型の値ではなく)ノードアイテムにしか適用できないためです。そのかわり、箇条書きといったデザイン要素とともに、自動計算を使用することは可能で、以下のスクリーンショットの左側にあるユーザー定義関数により、右側のような出力が生成されます。



メモ SPS にて XSLT 1.0 が使用されている場合、XPath 条件式の返り値をノードセットにする必要があり、そうでない場合エラーとなります。

XPath を使ったテンプレートノードの選択を行う利点

スキーマノードを XPath 条件式(ユーザー定義テンプレート)により選択することで、XPath のパスセレクターメカニズムを使ったアイテムノードやシーケンスの選択を行うことができるようになり、ノードの選択に対してフィルタリングや条件をセットすることが可能になります。そのため、特定の XML ドキュメントノードを、任意のテンプレートに対して使用することができるようになります。例えば、//Office/Department[@Location="NY"] という XPath 条件式により、NY という値を持った Location 属性が含まれる Department ノードだけが選択されます。このセクション内にあるその他の例も参照ください。

メモ XPath 条件式に複数のロケーションパスやロケーションステップが含まれる場合、複数のロケーションパスやステップ内に括弧が使用されているかがグループ化や並び替えを行う際に重要になります。例えば、/Org/Office/Dept という XPath 条件式は、(/Org/Office/Dept) という条件式とは異なった方法で処理されます。括弧を伴わない条件式では、各ロケーションステップが繰り返されるのに対して、括弧付きの条件式の場合、全ての Office 要素にある全 Dept 要素が、1 つのノードセットにより返されます。

括弧	実際の XSLT メカニズム	効果

無し	`<xsl:for-each select="Org">`
	`  <xsl:for-each select="Office">`
	`  <xsl:for-each select="Dept">`
	`  ...`
	`  </xsl:for-each>`
	`  </xsl:for-each>`
	`</xsl:for-each>`
		各 Office 要素には独自の Dept が含まれ、グループ化や並び替えは、各 Office 内部で行われる。

有り	`<xsl:for-each select="/Org/Office/Dept">`	Dept 要素が、全ての Office ならびに Org 要素に対して展

	... </xsl:for-each>	開される。

グループ化や並べ替えを行う際にXPath 条件式の評価方法が大きな違いをもたらします。

ユーザー定義テンプレートを挿入する

以下の操作によりユーザー定義テンプレートを挿入することができます:

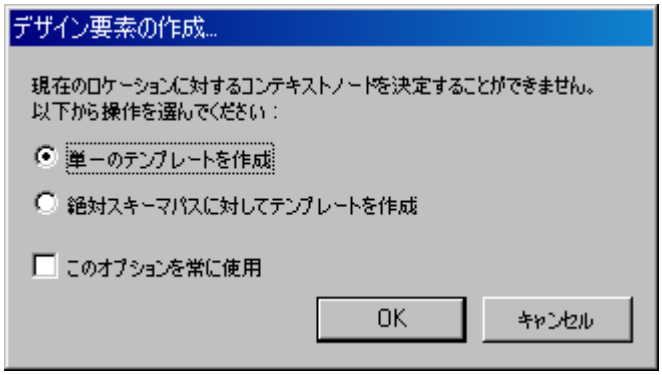
1. デザイン要素の挿入ツールバーにある**ユーザー定義テンプレートを挿入**アイコンをクリックし、デザイン内のテンプレートを挿入する場所をクリックします。または、テンプレートを挿入する場所を右クリックし、コンテキストメニューから**ユーザー定義テンプレートを挿入**コマンドを選択します。
2. In the XPath 式を編集 ダイアログにて XPath 条件式を入力し、**OK** をクリックします。XPath 条件式のコンテキストノードは、クリックされたノードになる点に注意してください。空のノードテンプレートが作成され、場合によっては結合されたノードが作成されます。ノードが結合されると、ターゲットとなるインスタンスノードが同一のレベルにて選択されるのに対し、ノードが結合されていない場合(つまり複数の階層レベルにて分割されている場合)、ループにより各インスタンスノードが選択されます。グループ化や並べ替えが行われない限り、(結合または分割により)返された両者のノードセットに違いはありません。結合されたノードがグループ化や並べ替えメカニズムでどのように扱われるかの詳細については、: ノードテンプレートオペレーション を参照してください。

テンプレートマッチの編集

任意の(ユーザー定義または通常の)ノードテンプレートに対する選択は、新たな XPath 条件式により変更することができます。ノードテンプレートのテンプレートマッチを編集するには、ノードテンプレートを右クリックし、**テンプレートマッチの編集**コマンドを選択します。XPath 条件式を編集するためのダイアログが表示されるので、新たなノードを選択する XPath 条件式を入力し、「OK」をクリックします。

ユーザー定義テンプレートにノードを追加する

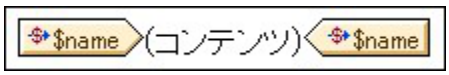
SPS のスキーマソースコンテキスト内に配置することができないノードまたはシーケンスに対してユーザー定義テンプレートが作成された場合、スキーマツリーから得られたノードがユーザー定義テンプレートへ追加される時に、新たなノードのコンテキストが決定できなくなります。そのため、新たなノードが、(i) 名前で(つまり相対パスで)参照されるのか、または(ii) スキーマソースのルートからのフルパスにより参照されるのが決定するよう促されます(以下のスクリーンショットを参照)。



どのような動作を行うか、確認するのがデフォルトの振る舞いになっています。「ツール| オプション ダイアログ」のデザインタブにて、デフォルトの振る舞いを変更することができます。

6.4.4 変数テンプレート

変数テンプレートとは変数に対して使用されるテンプレートで、通常その値が出力されます。メニューオプションの「**挿入 | 変数テンプレートの挿入**」または「**タグで囲む | 変数テンプレート**」を選択することで、カーソル位置に、SPS で定義される変数のテンプレートが挿入されます。変数テンプレートには、デフォルトでコンテンツプレースホルダが含まれており、これにより変数の値が出力されます（以下のスクリーンショットを参照）。変数テンプレートには、必要に応じてその他の（静的ならびに動的な）コンテンツを含めることができるほか、その他のテンプレートと同様に修正を行うこともできます。



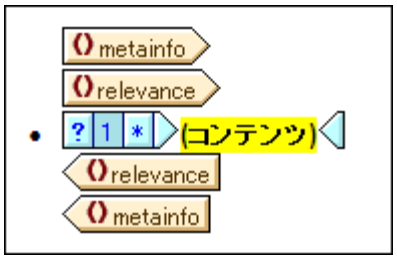
以下の操作により変数テンプレートを挿入することができます:

1. デザイン内のテンプレートを挿入する位置にカーソルを配置します。
2. メニューオプションから「**挿入 | 変数テンプレートの挿入**」を選択します。変数テンプレートの挿入ダイアログが表示されます（以下のスクリーンショットを参照）。

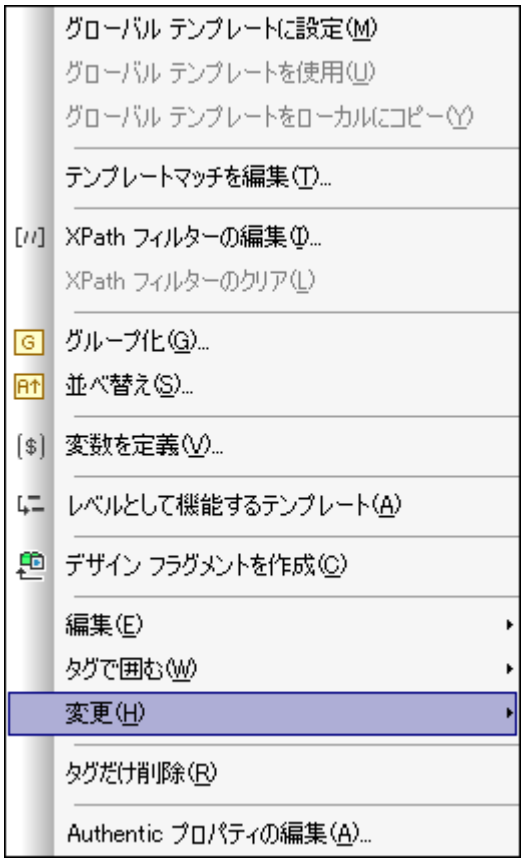
3. ダイアログには、SPS にて定義された ユーザー宣言パラメーターと変数 が表示されます。変数テンプレートに対して追加する変数を選択してください。
4. **OK** をクリックして完了します。

6.4.5 ノードテンプレート操作

ノードテンプレートとは、ノードに対する処理を指定するデザイン内のテンプレートのことです。デザインでは、ベージュ色の開始ならびに終了タグによりノードテンプレートが表示されます(以下のスクリーンショットを参照)。以下のスクリーンショットでは、metainfo とrelevance という2つのノードテンプレートが示されています。*次も参照してください*: *XML ドキュメント内のノード*。



ノードテンプレートにて行うことができる処理の一覧は、そのノードテンプレートの開始または終了タグを右クリックすることで表示されるコンテキストメニューから確認することができます、 *下のスクリーンショット参照*)。



以下ではコンテキストメニュー内にあるコマンドについて説明します:

- グローバルテンプレート
- テンプレートの一致
- XPath フィルター
- グループ化、並べ替え、変数の定義、レベルとして機能するテンプレート

- デザインフラグメントの作成
- タグなし削除
- 編集、タグで囲む、変更
- Authentic プロパティ

以下ではこれらのメニューコマンドについて説明します。ノードテンプレートによっては利用できないコマンドもある点に注意してください。利用できないコマンドはコンテキストメニューにてグレーアウトされます。

グローバルテンプレート：設定、使用、ローカルにコピー

メインテンプレート内にあるノードテンプレートを変更することができるほか、以下のコマンドによりグローバルテンプレートへ割り当てることができます:

- *メイングローバルテンプレートに設定*： スキーマにてグローバル要素として定義されている要素をノードが指し示している場合に使用することができます。ノードテンプレートからグローバルテンプレートが作成されます。メインテンプレート内にあるノードテンプレートではこのグローバルテンプレートが使用され、そのタグが（グローバルテンプレートの使用を表す）グレーで表示されます。
- *グローバルテンプレートを使用*： ノードテンプレートの名前と同じ修飾名でグローバルテンプレートが定義されている場合、ノードテンプレートに対してグローバルテンプレートの処理が使用されます。ノードテンプレートのタグがグレーになります。
- *グローバルテンプレートをローカルにコピー*： ノードテンプレートと同じ修飾名を持ったグローバルテンプレートの処理命令が、直接ノードテンプレートへコピーされます。ノードテンプレートはグローバルテンプレートとは独立したものになります。このコマンドを実行した後に、グローバルテンプレートとは独立したものとしてノードテンプレートを変更することができます。ノードテンプレートからグローバルテンプレートへの参照は行われないため、ノードのタグはベージュ色のままとまります。

に関する詳しい説明は、次のセクションを参照してください： グローバルテンプレート.

テンプレートマッチの編集

テンプレートに対して作成されたノードを、このコマンドにより変更することができます。テンプレートマッチの編集コマンドにより、XPath 式の編集ダイアログが表示され、スキーマ内にある他のノードを選択するXPath 条件式を入力することができます。XPath 条件式を入力することで、テンプレートをユーザー定義テンプレートへ変更することもできます。

XPath フィルターの編集/クリア

XPath フィルターにより、ノードテンプレートが適用されるノードセットのフィルタリングを行うことができます。XPath フィルターはグローバルテンプレートに対して適用することができます。

デフォルトでは、ノードテンプレートが作成されたノードに対応するノード（要素または属性）に対してノードテンプレートが適用されます（スキーマ構造の指定された位置にある同名の出現）。例えば、/Personnel/Office ノードにより、全ての/Personnel/Office 要素が選択されます。Office 要素にて 1 という式を持ったXPath フィルターが作成された場合、述語式がOffice 要素に加えられるのと同様の結果が得られ、全体のXPath 条件式は/Personnel/Office[1] となります。このXPath 条件式により、最初のPersonnel 要素にある最初のOffice 子要素が選択され、その他のOffice 要素は選択されないことになります。

フィルターは、デザイン内にある任意のノードテンプレートならびに複数のノードテンプレートに対して追加することができます。この機能を使って、例えば/Personnel/Office[@country='US']/Person[Title='Manager'] といったXPath 条件式により、社内のUS オフィスにいる管理職全員を選択することができます。この例では、Office とPerson のノードテンプレートにて、それぞれ独立したフィルターが作成されます。

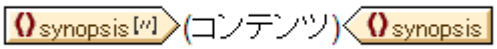
グローバルテンプレートが使用される（呼び出される）場合、XPath フィルターを適用することができます。従って、使用されているグローバルテンプレートのインスタンス全てに対してXPath 条件式を適用することで、ターゲットとなるノードセットを制限することができます。

XPath フィルターをノードに追加するには、ノードテンプレートを右クリックして、**XPath フィルターの編集**を選択します。引用句や角かっこ、デリミターなしでXPath フィルター条件式を入力してください。以下にあるような妥当なXPath 条件式を入力することができます:

(C) 2015-2021 Altova GmbH

- 1
- @country='US'
- Title='Manager'

ノードテンプレートに対して XPath フィルターが追加されると、そのノードテンプレートの開始タグにフィルターシンボルが表示されます。以下のスクリーンショットでは、synopsis ノードテンプレートに対してフィルターが追加されました。



メモ 各ノードテンプレートでは、1つの XPath フィルターがサポートされます。

グループ化、並べ替え、変数の定義、レベルとして機能するテンプレート

これらコマンドのメカニズムの詳細については以下のセクションにて記述されます:

- **グループ化**コマンドにより、選択されたノードテンプレートにより表されるノードのインスタンスがグループ化されます。グループ化のメカニズムについては、グループ化のセクションにて記述されます。
- **並べ替え**コマンドにより、選択されたノードテンプレートにより表されるノードのインスタンスが並べ替えられます。並べ替えのメカニズムについては、並べ替えのセクションにて記述されます。
- **変数の定義**コマンドにより、選択されたノードテンプレートをスコープとした変数を定義することができます。変数の使用方法については、変数のセクションにて記述されます。
- **レベルとして機能するテンプレート**コマンドは、ノードテンプレート上のレベルを作成/削除するためのトグルコマンドです。ドキュメントを階層構造で構築するためには、様々な箇所でレベルを指定する必要があります。この構造は、目次や番号の自動割り当て、そしてテキストの参照を生成するために使用されます。これら機能の詳細については、目次(TOC)と参照.

デザインフラグメントの作成

選択されたテンプレートからデザインフラグメントのテンプレートを作成します。作成されたデザインフラグメントのテンプレートは、デザインの下部にあるデザインフラグメントのテンプレートへ追加され、デザインツリーならびにスキーマツリーへ追加されます。作成された箇所へも、そのデザインフラグメントが適用されます。

(テンプレートや書式) タグだけ削除

このコマンドにより、選択されたテンプレートやフォーマットタグだけが削除され、子孫ノードやそのフォーマットタグが削除されることはありません。あるタグに含まれる子孫ノードなどを削除することなしにフォーマットタグや親要素のタグを削除するために使用することができます(タグが選択された状態で**削除**コマンドが実行されると、これら全てが削除されます)。親要素を削除することで、削除された要素の子孫ノードが不正となる可能性があることに注意してください。不正なノードには取り消し線が表示されます。

編集、タグで囲む、変更

これらコマンドの解説を以下に示します:

- *編集*: Windows で既に馴染み深い、切り取り、コピー、貼付け、そして削除コマンドがサブメニューに表示されます。
- *タグで囲む*: ノードテンプレートを、以下のデザインコンポーネントにより囲むことができます: 段落、特殊な段落、リスト、ハイパーリンク、条件、目次 ブックマークとレベル.
- *変更*: 変更機能により、(i) テンプレートが適用されるノード、または (ii) デザイン内でノードを作成する方法を変更することができます。詳細については、変更機能のセクションを参照ください。

プロパティの編集

Authentic プロパティの編集：このコマンドにより表示されるプロパティサイドバーから、Authentic View 内のインスタンス化されたノードに対して特定のプロパティを割り当てることができます。Authentic プロパティに関する詳細については、Authentic ノードプロパティ.

関連項目

- <% SPS% ファイルコンテンツ
- テンプレートとデザインフラグメント
- XPath ダイアログ

6.4.6 デザインフラグメント

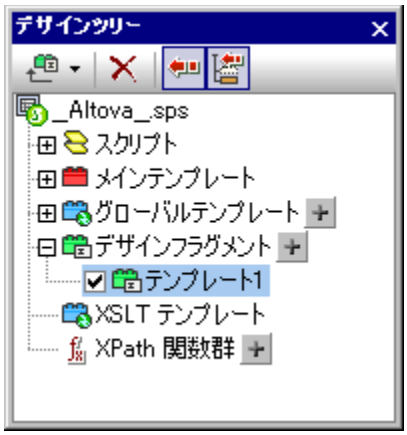
デザインフラグメントは、ドキュメント内の様々な箇所で再利用することができるパーツを作成するために使用することができます。以下に使用するための手順を示します:

1. デザインフラグメントのコンテンツを入力する
2. デザインフラグメントのコンテンツを入力する
3. テンプレート内にある目的の位置にデザインフラグメントを挿入する.

デザインフラグメントのコンテンツを入力する

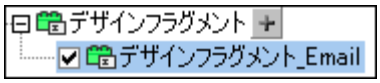
以下の操作によりデザインフラグメントを作成することができます:

1. デザインツリーならびにスキーマツリーにて、ツリー内部にあるデザインフラグメントアイテムの右側に配置されているデザインフラグメントの追加アイコン をクリックします（以下のスクリーンショットを参照）。この操作によりツリーのデザインフラグメントリストにデザインフラグメントアイテムが追加されます（以下のメモも参照ください）。



デザインフラグメントが SPS デザイン内部に作成されます。このテンプレートはデザイン内に既にあるテンプレートへ追加されます。StyleVision のデザインフィルターツールバーにある表示/隠すアイコンによりメインテンプレートとグローバルテンプレートを隠すことで、デザイン内にあるデザインフラグメントだけを確認することができます。デザインフラグメントテンプレートはスキーマツリーにも表示され、素早いアクセスを行うことができます。

2. デザインツリーまたはスキーマツリーにあるデザインフラグメントアイテムをダブルクリックすることで、その名前を編集することができます。目的の名前を入力し、Enter キーを押下してください。編集された名前がデザインツリーとデザイン内部のテンプレートに表示されます。



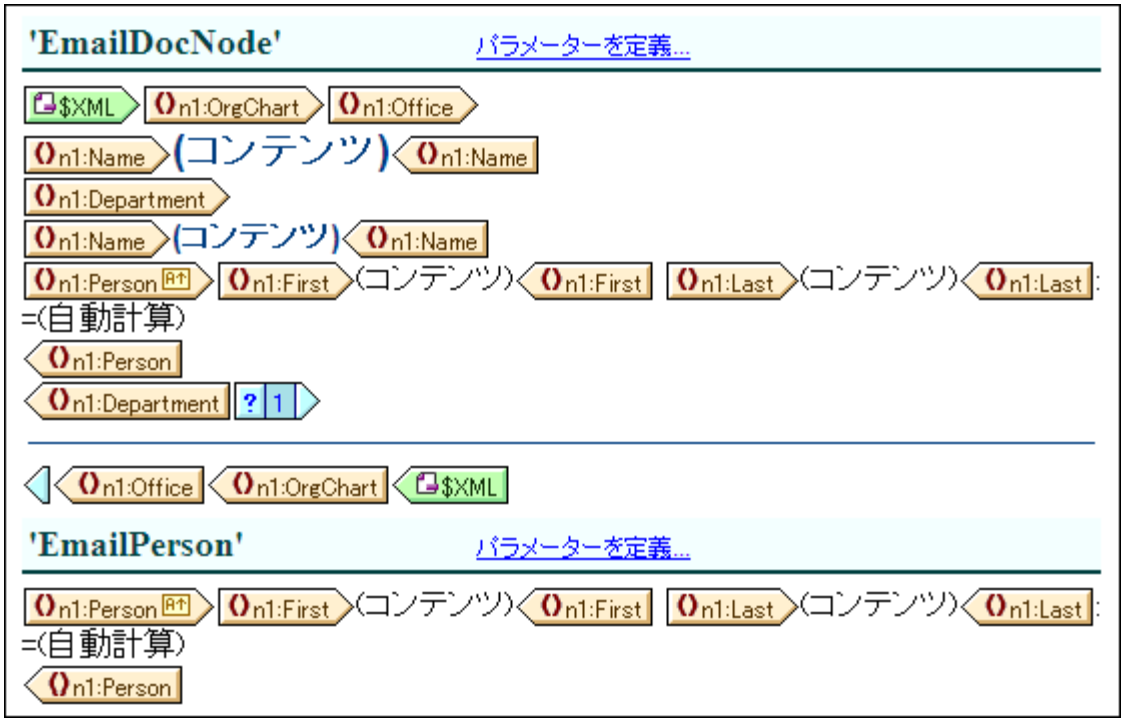
3. デザインにてデザインフラグメントテンプレートのコンテンツを作成します。以下ではその方法について記述されます。

メモ　既存のテンプレートからデザインフラグメントを作成する場合、そのテンプレートを右クリックして、コンテキストメニューから**デザインフラグメントの作成**コマンドを選択します。この操作により、選択されたテンプレートからデザインフラグメントが作成されます。デザインフラグメントのテンプレートがデザイン下部にある既存のデザインフラグメントテンプレートに追加され、デザインツリーとスキーマツリーへ追加されます。この方法によりデザインフラグメントを作成すると、選択されたテンプレートもデザインフラグメントとなり、デザインツリーやスキーマツリーから挿入.

デザインフラグメントのコンテンツを作成する

デザインフラグメントテンプレートのコンテンツは、その他のテンプレートと同様の方法で作成されます。静的コンテンツを挿入するには、カーソルをデザインフラグメントのテンプレート内部へ移動させ目的の静的コンテンツを挿入します。動的コンテンツを挿入するには、目的のスキーマノードをデザインフラグメントのテンプレート内部へドラッグします。

スキーマソースからノードをドラッグするには、(i) グローバル要素ツリーから、または (ii) ルート要素ツリーからノードをドラッグします。これら2つの方法には違いがあり、ノードがグローバル要素ツリーからドラッグされた場合、祖先要素無しに作成されるため、テンプレート内で使用される場合には、親のコンテキスト内部で使用される必要があります（以下のスクリーンショットにある EmailPerson デザインフラグメントを参照）。一方でルート要素ツリーからノードがドラッグされた場合、ドキュメントノードの構造とともにノードが作成され、テンプレート内部の任意の場所で使用することができます（以下のスクリーンショットにある EmailDocNode デザインフラグメントを参照）。



上のスクリーンショットでは、2つのデザインフラグメントテンプレートが示されており、Person 要素に対して同一の出力が生成されます。EmailPerson デザインフラグメントテンプレートでは、グローバル要素の Person を EmailPerson テンプレートへドラッグすることで Person ノードが作成されます。EmailDocNode デザインフラグメントテンプレートでは、ルート要素ツリーから Person ノードがドラッグされ、(ドキュメントノードを表す $XML からの)絶対パスとともにノードが作成されます。

これらデザインフラグメントのテンプレートがメインテンプレートへ挿入される場合、EmailPerson テンプレートが Person ノードの親ノードのコンテキスト内部から呼び出される様にする必要があります。このサンプルは (マイ) ドキュメントフォルダー 以下にある C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\DesignFragments 内のEmail.sps ファイルにて確認することができます。

デザインフラグメントには、デフォルトの値とともにパラメーターを定義することもできます。個々のデザインフラグメントのインスタンスに対して、異なる値をパラメーターへ割り当てることができます。詳細については デザインフラグメントに対するパラメーター を参照ください。

デザインを完了した後には、デザインのコンポーネントがデザインツリーでも示されている点に注目してください。

テンプレート内にある目的の位置にデザインフラグメントを挿入する

デザインフラグメントを挿入するには、デザインツリーまたはスキーマツリーから目的の場所へデザインフラグメントをドラッグします。デザインフラグメントをドロップする場所は、適切なコンテキストとなるような場所でなければなりません。デザインフラグメントのコンテンツがグローバル要素から作成された場合、メインテンプレート内にある適切なコンテキストがノードの親となるようにデザインフラグメントをドラッグする必要があります。上記の デザインフラグメントのコンテンツを作成する を参照ください。

その他にも、デザインフラグメントを挿入する場所を右クリックして、コンテキストメニューから**デザインフラグメントの挿入**を選択することもできます。

メモ デザインフラグメントがメインテンプレート内部で参照されており、デザインフラグメントの名前が後になって変更された場合、メインテンプレート内部にある参照は正しいものではなくなり、XSLT エラーが返されるようになります。問題を修正するには、メインテンプレート内にあるオリジナルの参照を削除して、名前が変更されたデザインフラグメントへの参照を新たに作成してください。

再帰的なデザインフラグメント

デザインフラグメントはそれ自体を呼び出すような、再帰的なかたちで作成することもできます。Authentic View における無限ループを回避するため、呼び出しの階層を制限するプロパティをセットすることができます。SPS のプロパティダイアログ(「ファイル | プロパティ」)にある Authentic タブ内の**テンプレート呼出し階層上限**プロパティにて値をセットすることができ、この値により、Authentic View 出力を処理する際に再帰的なテンプレート呼び出す回数の最大値が指定されます。テンプレート呼出し階層上限プロパティで指定された値を超えたテンプレートの呼び出しが行われた場合、エラーが返されます。

デザインフラグメントの削除

デザインフラグメントを削除するには、デザインツリーにて選択をして、デザインツリーの**削除**ツールバーアイコン をクリックします。

モジュール SPS におけるデザインフラグメント

SPS モジュールが別の SPS モジュールへ追加された 場合、追加されたモジュール内にあるデザインフラグメントも、参照元の SPS にて利用することができるようになります。モジュール SPS に関する詳細については、次のセクションを参照してください: モジュール SPS。

サンプルファイル

サンプル SPS は、(マイ) ドキュメントフォルダーにある、C:\Documents and Settings\<username>\My Documents¥Altova¥StyleVision2021¥StyleVisionExamples¥Tutorial¥DesignFragments にて確認することができます。

関連項目

- テキストとしてXML コンテンツを挿入
- その他のコンテンツ
- メインのテンプレート
- グローバルテンプレート
- テンプレートフィルター
- モジュールSPS

6.5 XSLT テンプレート

XSLT ファイルを SPS へインポートすることで、ファイル内部にあるXSLT テンプレートをグローバルテンプレートとして使用することができるようになります。XML ドキュメントの処理を行なっている際に、XML ノードの1つがインポートされたXSLT テンプレート内のノードへマッチした場合、インポートされたXSLT テンプレートがそのノードへ適用されます。インポートされたXSLT ファイルに名前付きテンプレートが含まれる場合、デザイン内部にそれらを配置することができます。

メモ

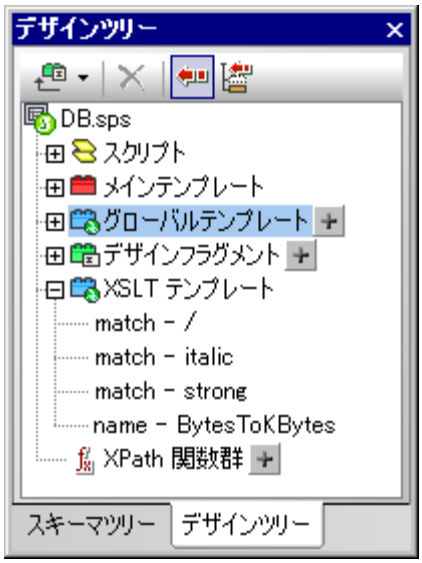
- インポートされた XSLT テンプレートは StyleVision 内部で修正できません。
- XSLT テンプレートは Authentic View でサポートされません。

XSLT ファイルをインポートする

以下の操作によりXSLT ファイルをインポートすることができます:

1. デザイン概要サイドバーにて、**新規 XSLT ファイルを追加...** リンクをクリックします（以下のスクリーンショットを参照）。

2. ファイルを開くダイアログが表示されます。目的の XSLT ファイルを選択し、**開く**ボタンをクリックします。XSLT ファイルがインポートされます。XSLT スタイルシートには xsl:import ステートメントが追加され、デザインツリーサイドバーのXSLT テンプレート以下には、インポートされた XSLT ファイル内部に含まれる XSLT テンプレートが表示されます（以下のスクリーンショットを参照）。



(C) 2015-2021 Altova GmbH

インポートされたXSLT テンプレートには2種類あります: (i) マッチテンプレート(**match** により表示)と(ii) 名前付きテンプレート(**name** により表示)。デザインツリーでは、これら2種類のテンプレートが、それぞれ(i) match テンプレートにある select 属性の値、(ii) 名前付きテンプレートにある name 属性の値とともに表示されます。

Match テンプレート

XML ドキュメントインスタンス内部にあるノードに対してテンプレートが適用され、match テンプレートが適用されるように選択されている場合に match テンプレートが使用されます。XML ノードの修飾名が、インポートされた match テンプレートにマッチする場合に使用されます。同じ修飾名を持つグローバルテンプレートがSPS 内部で作成された場合、インポートされたテンプレートよりも優先順位が高くなるため、そのテンプレートが使用されます。複数のXSLT ファイルがインポートされた場合、最初にインポートされたファイル(XSLT コードの最初に表示されるファイル)の優先順位が最も低くなり、2番目にインポートされたファイルの優先順位が2番目に低いものとなります。

名前付きテンプレート

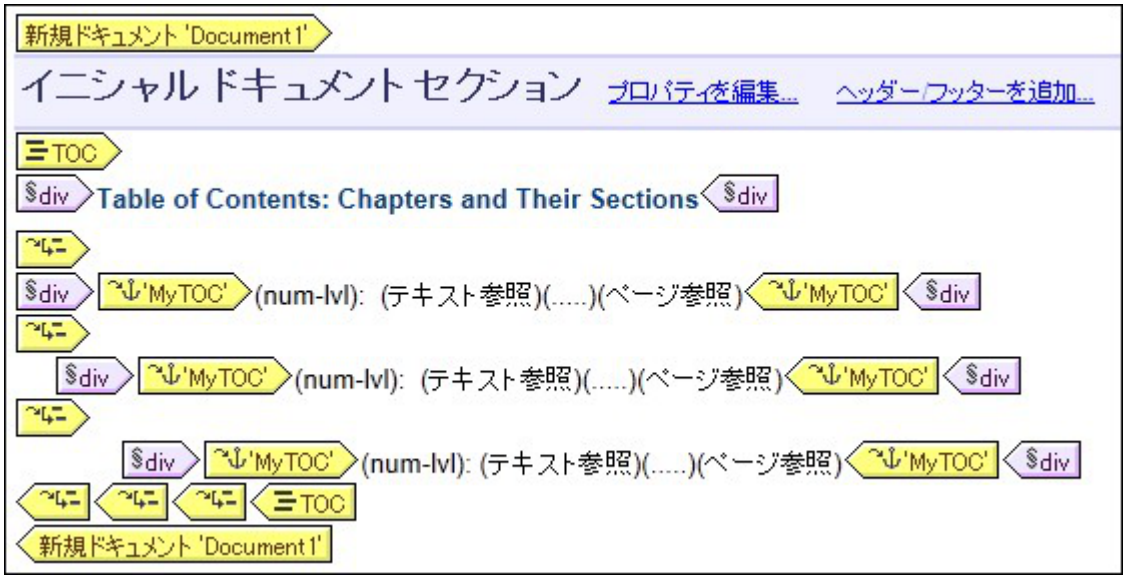
名前付きテンプレートは、デザインツリーからデザイン内の任意の場所へドラッグすることができます。その場所にて名前付きテンプレートを呼び出す xsl:call-template 要素が作成されます(以下のスクリーンショットを参照)。

```
<xsl:call-template name="BoldItalic" />
```

これにより、デザイン内部で指定された場所にて名前付きテンプレートが出力に実装されます。このようなメカニズムは、XML インスタンスドキュメントやXSLT スタイルシートから独立したコンテンツを挿入する際に便利な機能となります。

6.6 複数の ドキュメントの出力

SPS は、メイン出力ドキュメントとその他のドキュメントというかたちで、複数の出力ドキュメントを生成するようにデザインすることもできます。この機能を使用することで、出力のモジュール化を行うことが可能になります。新規ドキュメントテンプレートを挿入することで、出力ドキュメントをデザイン内に作成することができます(以下のスクリーンショットを参照)。各出力ドキュメントのコンテンツは、新規ドキュメントテンプレート内部に配置されます。



新規ドキュメントテンプレートは、ドキュメントデザイン内にある任意の場所で作成することができ、あらゆる階層で出力のモジュール化を行うことができます。例として、世界的な組織内部にある複数の支店が、以下の階層により個別の出力ドキュメントを持つような状況を考えてみましょう: (i) 世界、(ii) 大陸、(iii) 国、(iv) 州/県、(v) 支店。例えば、各支店を個別の出力ドキュメントにて表示し、ある国にある全ての支店を1つの文章にまとめて表示することができます。デザインでは、出力を行う個々の階層レベルにて新規ドキュメントテンプレートを作成する必要があります。適切なドキュメント構造をセットアップする方法については、新規ドキュメント テンプレートとデザイン構造のセクションにて記述されます。

複数の出力ドキュメント機能は、HTML とRTF 書式で使用することができます。Authentic View では、単一のドキュメントにてコンテンツが表示されます。

複数の出力ドキュメントに関する説明は、以下にある項目ごとに記述されます:

- 新規ドキュメントテンプレートの挿入
- 新規ドキュメント テンプレートとデザイン構造
- 新規ドキュメント テンプレートのURL
- 出力ドキュメントファイルのプレビュー
- ドキュメント プロパティとスタイル

関連項目

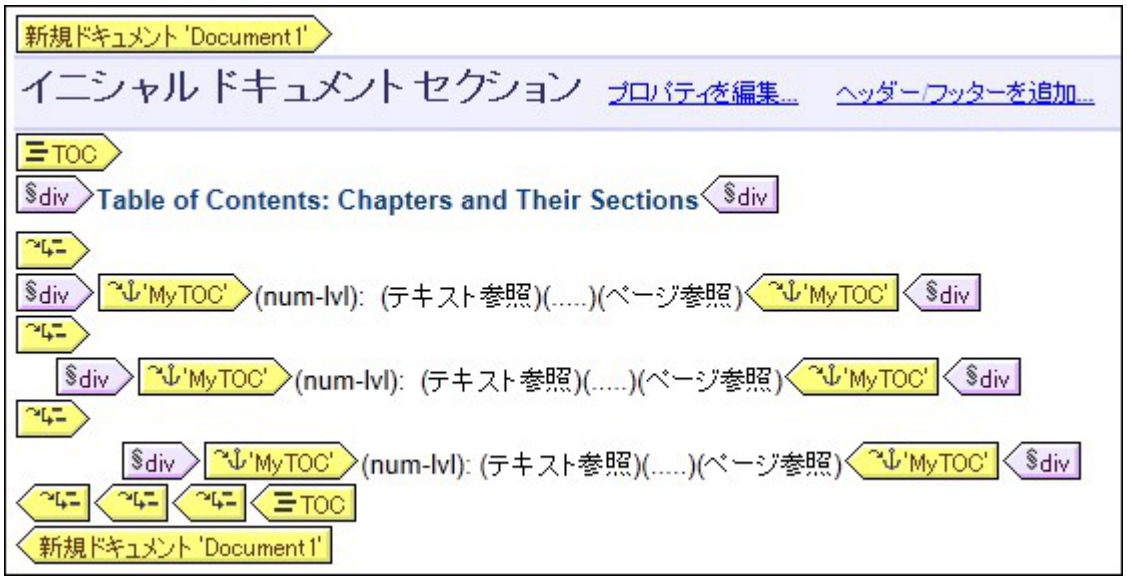
- 出力内のHTML フラグメント

6.6.1 新規 ドキュメントテンプレートの挿入

以下に示される2つの方法により、新規ドキュメントテンプレートをSPS デザイン内部に配置することができます:

- 新規出力ドキュメントテンプレートは、デザイン内の任意の位置へ**挿入**することができます。この場合、テンプレートが挿入された後に新規ドキュメントのコンテンツを追加することになります。新規ドキュメントテンプレートを挿入するには、デザイン内にある目的の位置にカーソルを配置して、メニューオプションから「**挿入 | 新規ドキュメントの挿入**」コマンドを選択するか、その場所を右クリックすることで表示されるコンテキストメニューから、**新規ドキュメントの挿入**を選択します。
- 新規ドキュメントテンプレート内部で**コンテンツを囲む**ことで、新規出力ドキュメントをデザイン内に配置することができます。この場合、新規ドキュメントテンプレートにタグで囲まれたコンテンツが含まれることになります。デザインでは、コンテンツの追加や修正を行うことができます。コンテンツを囲むかたちで新規ドキュメントテンプレートを配置するには、対象となるコンテンツがハイライトされた状態で、メニューオプションから「**タグで囲む | 新規ドキュメント**」コマンドを選択するか、選択したコンテンツを右クリックして、コンテキストメニューから「**タグで囲む | 新規ドキュメント**」を選択します。

コンテンツを含む新規ドキュメントテンプレートを以下のスクリーンショットに示します。



上のスクリーンショットでは、以下の点に注目してください:

1. 新規ドキュメントテンプレートのタグには生成されるドキュメントの URL（パスと名前）が含まれます。出力フォーマットのファイルタイプにより、ファイルの拡張子が自動的に生成されます。例えば、HTML 出力フォーマットの場合、URL にあるファイル名の末尾に .html が加えられます。URL の割り当てについては、新規ドキュメント テンプレートのURL のセクションにて記述されます。
2. 新規ドキュメントテンプレートには、イニシャルドキュメントセクションが1つ含まれます。ドキュメント セクションに記述される方法を使用することで、新たなドキュメントセクションを追加することができます。.

関連項目

- 複数のドキュメントの出力
- 新規ドキュメント テンプレートとデザイン構造
- 新規ドキュメント テンプレートのURL
- 出力ドキュメントファイルのプレビュー

- ドキュメント プロパティとスタイル

6.6.2 新規ドキュメント テンプレートと デザイン構造

複数の出力ドキュメントを作成する場合、ソースドキュメント内にある適切なノードにて個別の新規ドキュメントテンプレートを作成する必要があります。従って、複数の出力ドキュメントをデザインする際には、出力の構造 と入力(ソースXMLドキュメント)の構造 の両方を考慮する必要があります。

メインの出力ドキュメントと追加の出力ドキュメント (出力の構造)

最初の新規ドキュメントテンプレートがデザインへ追加されると、この新規ドキュメントテンプレートの外にある全てのデザインコンテンツが別のドキュメントへ自動的に割り当てられます。この(新規ドキュメントテンプレートとは別の)ドキュメントがメインの出力ドキュメントとなり、StyleVision の出力プレビューではメイン出力ドキュメントとして参照されます。

(メニューオプションの「**ファイル | 生成されたファイルの保存**」コマンドを使用することで)生成された出力ドキュメントでは、メイン出力ドキュメントの名前が、**生成されたファイルの保存**コマンドより出力ドキュメントを生成する際に割り当てられる名前となります。メインドキュメント以外の出力ドキュメントファイルに使用される名前は、それぞれの新規ドキュメントテンプレートのURL にて割り当てられます。

新規ドキュメント テンプレートとソースドキュメントの構造

新規ドキュメントテンプレートが作成される際には、階層構造のどこにそれが作成されるかが重要になります。以下にある2つのケースが考えられます:

1. 新規ドキュメントテンプレート内に作成されたノードが1度だけ処理される。この場合、新規ドキュメントテンプレートも1度だけ処理されます。新規ドキュメントテンプレートの URL にあるファイル名は静的な名前にすることができます。
2. 新規ドキュメントテンプレート内に作成されたノードが複数回処理される。その結果、ノードが処理される回数だけ、新規ドキュメントテンプレートも処理されます。以下のような例を考えてみましょう。Office 要素に(各部門を表す)複数の Department 子要素が含まれています。デザインに表示されている Department ノード内部に新規ドキュメントテンプレートが作成された場合、Department ノードが(Office 要素内にある Department 要素の数だけ)複数回処理されるため、(ソース XML にある Office 要素内の全 Department 要素に対して)新規ドキュメントテンプレートも複数回処理されます。そのため、新規ドキュメントテンプレートの URL プロパティ内にあるファイル名は、動的な名前でなければなりません。静的な名前が使用された場合、各部門を表す Department 要素に対して作成された出力ドキュメントは、全て同じ名前を持つことになります。

関連項目

- 複数のドキュメントの出力
- 新規ドキュメントテンプレートの挿入
- 新規ドキュメント テンプレートのURL
- 出力ドキュメントファイルのプレビュー
- ドキュメント プロパティとスタイル

6.6.3 新規ドキュメント テンプレートの URL

このセクションでは、新規ドキュメントテンプレートのURL をデザイン構造に関連付ける方法 とURL の編集方法 と複数の出力ドキュメント同士でリンクを作成する方法 について説明します。

(C) 2015-2021 Altova GmbH

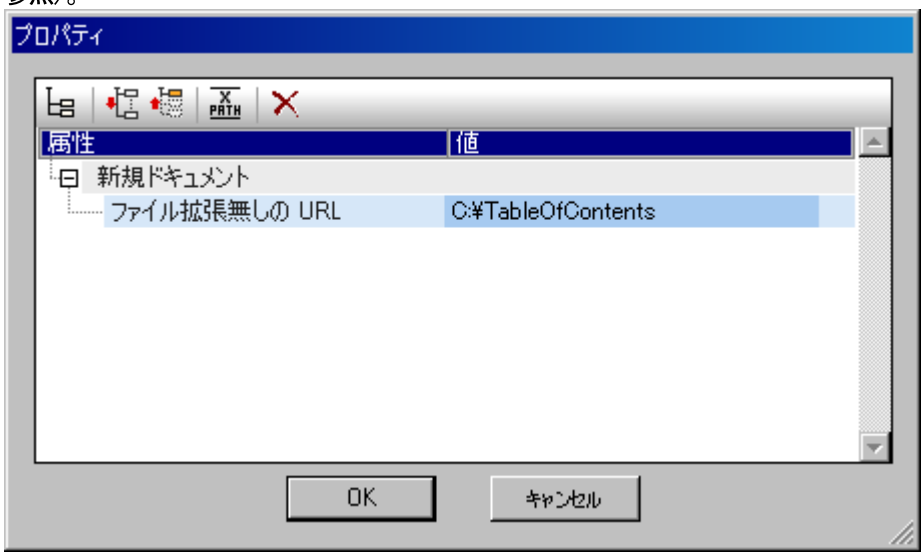
新規ドキュメント テンプレートのURL

新規ドキュメントテンプレートが1度だけ処理される場合（*前のセクション*を参照）、テンプレートのURL プロパティを静的な URL に指定することができます。以下のスクリーンショットでは、ドキュメント要素（$XML）の直下に新規ドキュメントテンプレートが配置されており、処理は1度だけ行われます。URL にはTableOfContents という静的な値が与えられており、この値が出力ドキュメントのファイル名となります。ファイル名のパスにプレフィックスは与えられていないため、メインドキュメントと同じディレクトリにファイルが生成されます（詳細については複数のドキュメント の出力とプレビューを参照ください）。その他にも、URL にパスが含まれる場合、出力ドキュメントがパスにより指定された場所へ保存されます。

その他にも、新規ドキュメントテンプレートが複数回処理されることで複数の出力ドキュメントが生成される場合（*前のセクション*を参照ください）、テンプレートのURL プロパティをXPath 条件式を使った動的な URL とする必要があります。以下のスクリーンショットでは、新規ドキュメントテンプレートのURL にbody/header/para というXPath 条件式が使用されています。新規ドキュメントテンプレートはtopic 要素内部に配置されており、topic 要素が処理されるたびに処理されることになります。topic 要素に対する各イタレーションにて、そのtopic 要素にあるbody/header/para 要素が新規ドキュメントテンプレートのURL として割り当てられることになります。これにより、各 topic 要素に対して新たなドキュメントが作成されることになります。各ドキュメントには、（トピックヘッダーのテキストである）body/header/para 要素から得られた名前が個別に与えられます。.

URL の編集

新規ドキュメントテンプレートがデザインへ追加されると、DocumentX という名前（X は整数）の静的なテキスト文字列をデフォルトとする URL とともに作成されます。URL を編集するには、新規ドキュメントテンプレートを右クリックして、**URL の編集**コマンドを選択します。プロパティダイアログが表示され、**ファイル拡張なしの URL** プロパティのフィールドに値を入力することができます（以下のスクリーンショットを参照）。



静的な URL を入力するには、値フィールドに目的の URL テキストを入力します。動的な URL を入力する場合、値フィールドをクリックしたあとに、プロパティダイアログのツールバーにある XPath ボタンをクリックして、目的の XPath 条件式を入力します。以下の点に注意してください: (i) XPath 条件式のコンテキストノードは新規ドキュメントテンプレートが挿入されたノードになります、(ii) XPath 条件式にプレフィックスを与えるには、（例えば、concat('C:¥MyOutput¥', body¥header¥para) といったかたちで）XPath の concat() 関数を使用してください。この例にある条件式により、C:¥MyOutput¥filename という URL 文字列が生成されます。出力フォーマットに従い、適切なファイル拡張子が自動的に生成されます。

(C) 2015-2021 Altova GmbH

ドキュメントへのリンク

ブックマークとハイパーリンクを使用することで、複数の出力ドキュメント間にリンクを作成することができます。ブックマークは、新規ドキュメントテンプレートのヘッダーを含む新規ドキュメントテンプレート内の任意の場所に配置することができます。その後、他のドキュメントから作成したブックマークへのハイパーリンクを作成することができるようになります。複数回処理されるノード上にブックマークを作成する必要がある場合、ブックマークの名前が動的に生成されるようにする必要があります。そうでない場合(静的なブックマーク名が与えられた場合)、出力に与えられる複数のノードが同じ名前のブックマークを持つことになります。

目次(TOC)もドキュメントへのリンクとして使用することができます。TOC を独立したドキュメント(例えばメインドキュメント)内に配置して、様々な出力ドキュメントに対してリンクを作成し、それぞれの出力ドキュメントから TOC へ戻るリンクを作成することができます。

関連項目

- 複数のドキュメントの出力
- 新規ドキュメントテンプレートの挿入
- 新規ドキュメント テンプレートとデザイン構造
- 出力ドキュメントファイルのプレビュー
- ドキュメント プロパティとスタイル

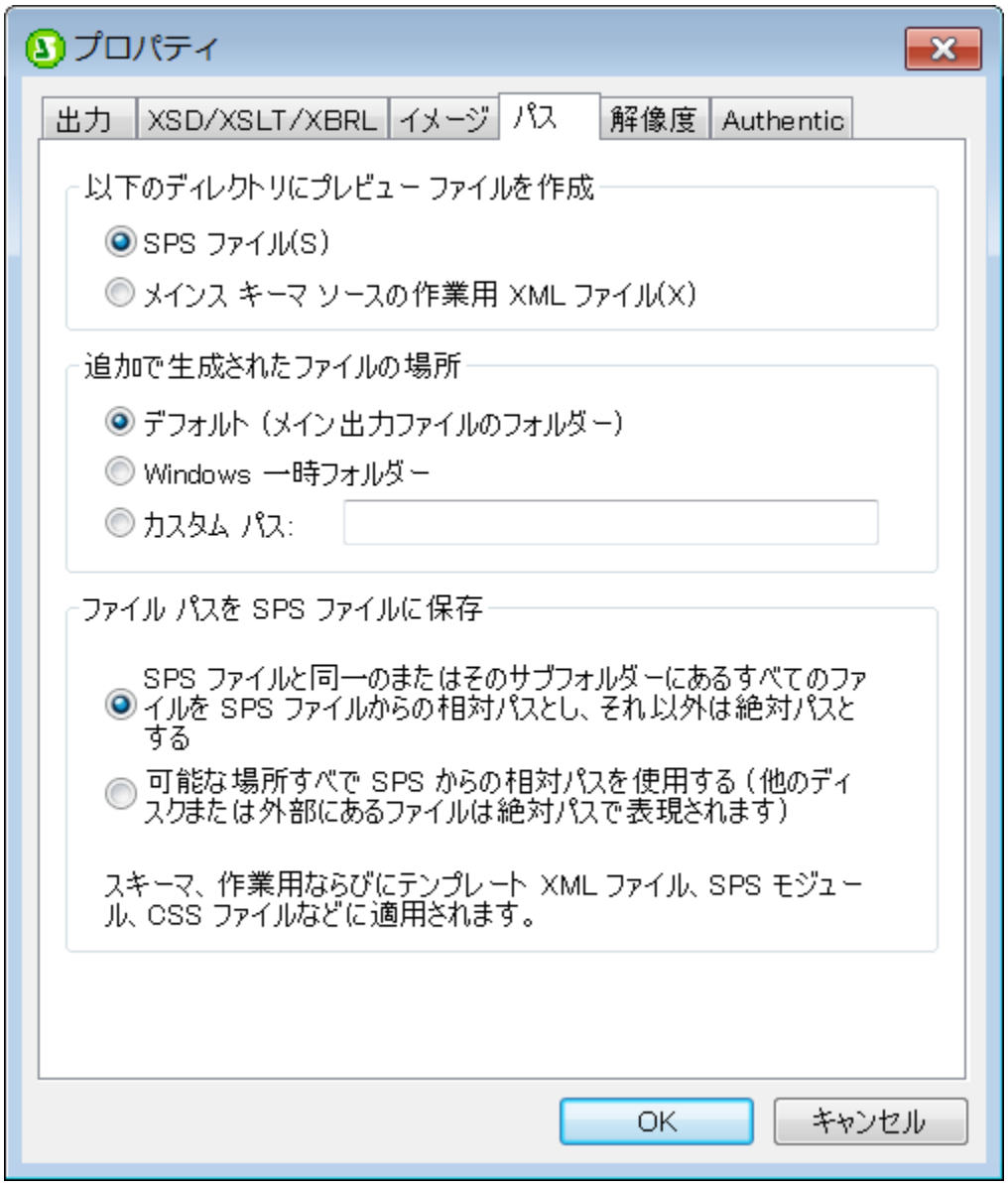
6.6.4 プレビューファイルと出力 ドキュメントファイル

デザインドキュメントの出力プレビューでは、デザインにて個別のドキュメントとして指定された各出力ドキュメントを確認することができます。(以下のスクリーンショットを参照)。

上のスクリーンショットでは、複数の出力ドキュメントを生成するようにデザインされた SPS ドキュメントのプレビューが示されています。各出力ドキュメントへは、(i) 左上に表示されている矢印ボタンをクリックすることで、または(ii) コンボボックスのドロップダウンリストから目的のドキュメントを選択することでアクセスすることができます(上のスクリーンショットを参照)。ドロップダウンリストのアイテムには、URL 全体(パスとファイル名)が表示される点に注目してください。

プレビューファイルの場所

プレビューファイルは、デフォルトで SPS ファイルが作成されるディレクトリにて作成されます。デフォルトの設定は、(メニューオプションの「ファイル | プロパティ」にて表示される)SPS ファイルのプロパティダイアログにある パスタブにて変更することができます(以下のスクリーンショットを参照)。このタブでは、作業用 XML ファイルの場所を指定することができます。新規ドキュメントテンプレートの URL にパスが含まれている場合、このパスで指定された場所が、そのプレビューファイルの格納に使用される場所となります。指定された場所が見つからない場合、エラーが返されます。出力ドキュメント間のリンクを作成する場合、それぞれの出力ファイルがどこで作成されるかを考慮する必要があります。



プロパティダイアログのパスタブでは、出力ドキュメントファイル、イメージやチャートイメージファイルといったプレビューを行うために一時的に使用されるファイルの配置場所を指定することもできます。新規ドキュメントテンプレートのURL にパスが含まれる場合、そこで指定されたパスが使用される点に注意してください。

出力(パス)の生成

以下の操作により、出力ドキュメントファイルを生成することができます:

1. メニューオプションの「**ファイル | 生成されたファイルの保存**」コマンドにマウスを移動させ、目的の出力フォーマットを選択します。
2. 生成されたファイルの保存ダイアログが表示されるので、生成されたファイルを保存するフォルダーを選択します。
3. メインドキュメントファイルの名前を入力し、**保存**ボタンをクリックします。

出力ドキュメントファイル、イメージファイルやチャートのイメージファイルといったその他生成されるファイルの場所が、ポップアップウィンドウにて表示されます。

(C) 2015-2021 Altova GmbH

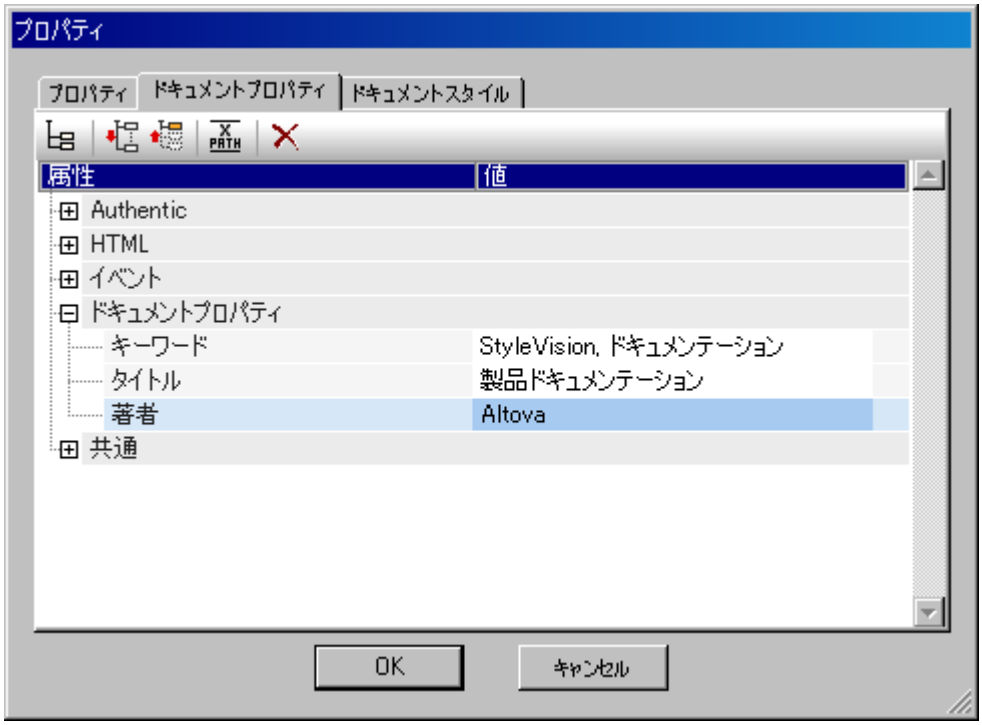
生成されたファイルの保存ダイアログにて選択したフォルダーに、メインドキュメントファイルが保存されます。新規ドキュメントテンプレートにて作成されたその他全てのドキュメントで、URL にパスが含まれていないドキュメントも、メインドキュメントファイルと同じフォルダー内に保存されます。新規ドキュメントテンプレートのURL にてパスが入力された場合、URL にて指定された場所に出力ドキュメントが保存されます。指定されたフォルダーが存在しない場合、エラーが返されます。

関連項目

- 複数のドキュメントの出力
- 新規ドキュメントテンプレートの挿入
- 新規ドキュメント テンプレートとデザイン構造
- 新規ドキュメント テンプレートのURL
- ドキュメント プロパティとスタイル

6.6.5 ドキュメント プロパティ と スタイル

SPS デザインでは、出力を複数のドキュメントへ分割することができます。これらの各ドキュメントには、異なるドキュメントプロパティならびにドキュメントスタイルを割り当てることができます。これらの値は、各ドキュメントのイニシャルドキュメントセクションからアクセスできるプロパティダイアログのドキュメントプロパティならびにドキュメントスタイルタブにて指定することができます。プロパティダイアログへアクセスするには、編集を行うドキュメントのイニシャルドキュメントセクションのタイトルバーにあるプロパティの編集リンクをクリックしてください。ドキュメントプロパティとドキュメントスタイルは、その出力ドキュメント全体に対して適用されます。



ドキュメントプロパティタブでは、プロパティのドキュメントプロパティグループにてドキュメントのメタ情報を入力することができます。このメタ情報は対応する出力ドキュメントと、出力フォーマットに対応したプロパティに保存されます。例えば、HTML 出力フォーマットの場合、HEAD 要素の META タグにてプロパティが保存されます。

ドキュメントスタイルに関する詳細については、CSS プロパティ値の設定のセクションを参照ください。

関連項目

- 複数のドキュメントの出力
- 新規ドキュメントテンプレートの挿入
- 新規ドキュメント テンプレートとデザイン構造
- 新規ドキュメント テンプレートのURL
- 出力ドキュメントファイルのプレビュー

(C) 2015-2021 Altova GmbH

7 高度な機能

SPS デザインの基本的なコンテンツや構造の作成方法については、SPS ファイルコンテンツとSPS ファイル構造 にて記述されます。しかし、ソースデータにあるコンテンツや構造を特定の形式に修正しなければならないことも多々あります。例えば、(個人情報が含まれる) ノードのグループを、苗字といった特定の条件により並べ替えるケースなどが考えられます。他にもデータベース内にある顧客情報を都市ごとにグループ化して、都市ごとの製品売上額を確認するようなケースも考えられます。StyleVision には以下で説明されるような検索機能が搭載されています。

StyleVision のSPS ファイルに搭載された機能を以下に示します:

- 自動計算: 自動計算とはXPath をベースにしたパワフルなメカニズムで、(i) 修正されたデータを出力にて表示するだけでなく、(ii) 自動計算の結果により、XML ドキュメント内にあるノードを更新することができます。
- 条件: データ構造や、XML の値、またはXPath 条件式の結果からなる条件により、テンプレートとテンプレートのコンテンツ処理を行うことができます。
- グループ: XPath 条件式により選択された要素のグループに対して処理を定義することができます。
- 並べ替え: XML 要素のセットを複数のソートキーにより並べ替えることができます。
- パラメーターと変数: グローバルな SPS レベルでデフォルト値を持つパラメーターを宣言することができます。これらの値はランタイムにてオーバーライドすることができ、コマンドラインから値を受け渡すこともできます。SPS 内部では変数を定義することができ、これらの変数を SPS 内部から参照して使用することができます。
- 目次 (TOC) と参照: 目次 (TOC) は全ての出力フォーマットにて、かつドキュメントの様々な場所にて作成することができます。TOC は、最初に目次にて参照されるアイテムを選択し、これらのアイテムを目次にて参照することにより作成されます。参照を使用するその他の機能には、(i) テキスト参照 (ドキュメント内にあるテキストをマークして、ドキュメントの他の場所から参照を行う)、そして (iii) ブックマークとハイパーリンク (ブックマークのキーにより、ハイパーリンクの参照先を作成する) があります。ハイパーリンクは外部のソースに対しても使用することができます。これらの参照メカニズムについても、このセクション以下で説明されます。

関連項目

- SPS ファイルコンテンツ
- SPS ファイル構造
- SPS ファイル追加機能

7.1 自動計算

自動計算機能により、(i) 出力ドキュメント内にある任意の場所で XPath の評価結果を表示し、(ii) オプションとして、XPath の評価結果を使ってメイン XML ドキュメント内にあるノードの更新を(XML ドキュメントが Authentic View にて編集されている時に) 行うことができます。

自動計算機能は、以下のような処理を行う際に便利なメカニズムとなります:

- 動的なデータ値を伴うオペレーションの計算を挿入する: 例えば、Office 要素内にある Employee 要素の数を (count(Employee) により) 数えることができるほか、各 Invoice 要素内にある Price 要素の値の合計を (sum(Price) により) 計算し、Person 要素の FirstName と LastName 要素の値を (concat(FirstName, ' ', LastName) により) 統合することができます。このような操作により、動的に変化する XML ドキュメント内にあるデータから新たなデータを動的に生成して、生成されたデータを出力にて表示することが可能になります。
- ドキュメントのデータ構造から得られた情報を表示する: 例えば、XPath の position() 関数を使用することで、動的テーブルに行番号を動的に挿入し、ドキュメントのセクションに動的な番号を割り振ることができます。この機能を使用することで、動的に変化するドキュメント構造をベースにした情報を自動的に生成することが可能になります。
- 外部 XML ドキュメントから得られたデータを挿入する: XPath 2.0 の doc() 関数を使用することで、外部 XML ドキュメントのドキュメントルートへアクセスすることができるようになり、外部 XML ドキュメントのノードコンテンツを出力内に挿入することが可能になります。.
- メイン XML ドキュメント内にあるノードの値を更新する: 例えば、concat(Title, ' ', FirstName, ' ', LastName) といった XPath 条件式により、Addressee というノードを更新することができます。
- デザイン内部の任意の場所でノードのコンテンツを表示する

関連項目

- 挿入 | 自動計算
- XPath ダイアログ
- XSLT エンジンの情報

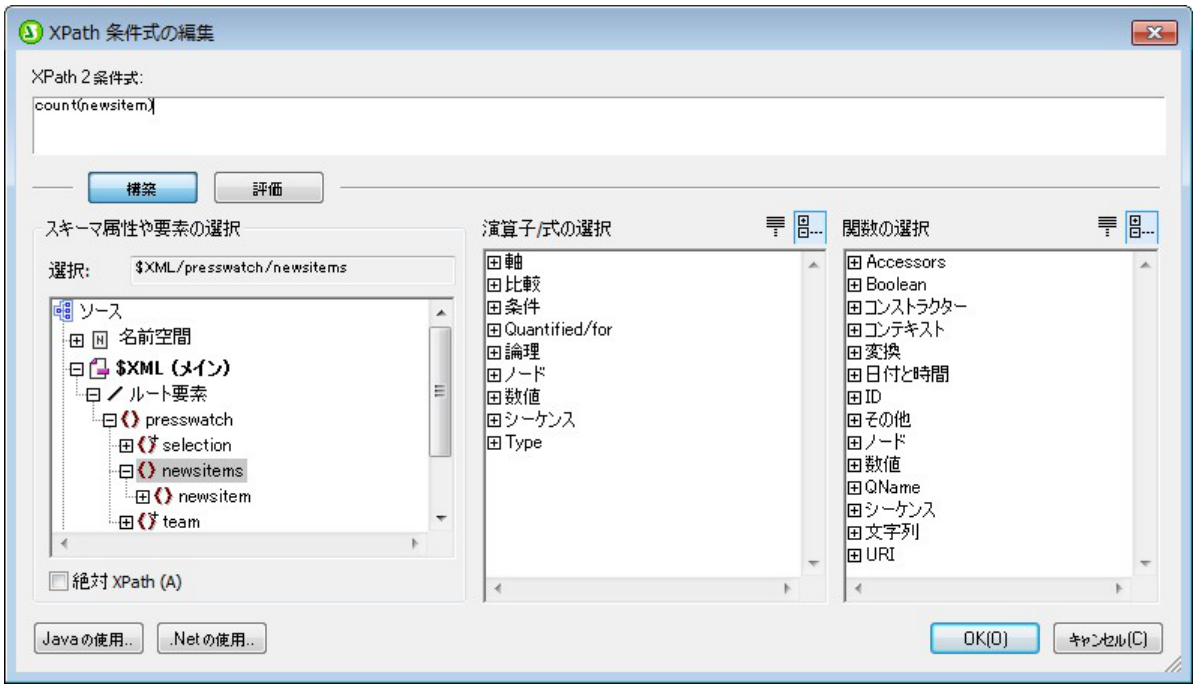
7.1.1 自動計算の編集と移動

自動計算の作成

以下の操作により、自動計算を作成することができます::

1. 自動計算の結果が表示される挿入ポイントへカーソルを配置して、メニューオプションから「**挿入 | 自動計算を挿入**」を選択します。計算結果がプレーンテキストとなる場合、表示されるサブメニューから**値**を選択し、入力フィールド (テキストボックス) 内に結果を表示する場合には**入力フィールド**を、複数行のフィールドに計算結果を表示する場合には**複数行フィールド**を選択します (自動計算の結果は値、または入力フィールド内部に表示されますが、Authentic View における出力であり、編集を行うことはできない点に注意してください)。XPath 条件式の編集ダイアログが表示されます (以下のスクリーンショットを参照)。

(C) 2015-2021 Altova GmbH

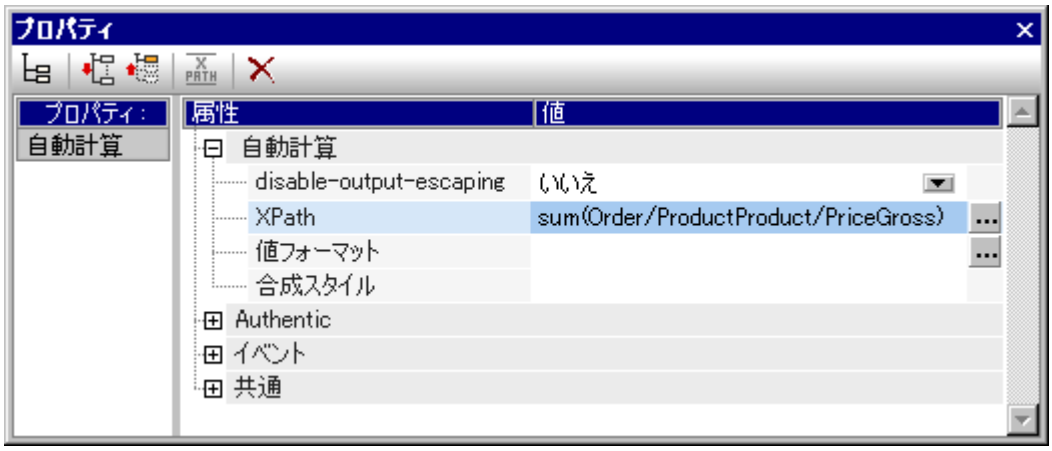


2. 条件式ペインにて、自動計算に使用される XPath 条件式をキーボードから入力します。その他にも、ダイアログの対応するペインにあるノード、演算子、そして関数をダブルクリックすることでも入力を行うことができます。挿入ポイントとなるコンテキストノードにも留意する必要があります。ダイアログが表示される時に、コンテキストノードはスキーマソースツリーにてハイライトされた状態で表示されます(上のスクリーンショットでは、newsitems がハイライトされています)。SPS で使用される XSLT 言語のバージョンに XSLT 1.0 を選択した場合、XPath 1.0 条件式を使用する必要があり、XSLT 2.0 を選択した場合には、XPath 2.0 を使用する必要があります。XPath 条件式の編集ダイアログに関する詳細については、XPath 式を編集セクションを参照ください。
3. オプションとして、XPath 条件式を介することで、自動計算の値を XML ドキュメント内にあるノードへコピーすることができます。自動計算の結果によりノードを更新する方法については、自動計算によりノードを更新するセクションを参照ください。

「OK」ボタンをクリックすることで、XPath 条件式の編集を完了することができます。自動計算のシンボルがデザインビューに表示されます。自動計算の結果を確認するには、Authentic View または、出力 ビュー

自動計算の編集

自動計算のXPath 条件式を編集するには、自動計算が選択された状態でプロパティサイドバーの自動計算グループにある XPath プロパティの編集ボタンをクリックしてください(以下のスクリーンショットを参照)。この操作により、XPath 条件式を編集するための XPath 式の編集ダイアログ (上のスクリーンショット) が表示されます(上のスクリーンショットを参照)。



自動計算のフォーマット

定義済みのフォーマットやCSS スタイルは、通常のテキストと同様の方法で自動計算に対して適用することができます。自動計算を選択し、フォーマットを適用します。更に、プロパティウィンドウの自動計算グループにある入力フォーマットプロパティでは、数値や日時といったデータ型を自動計算の入力フォーマット として指定することができます。

XPath 条件式には改行を含めることもできます。自動計算がpre タグにより囲まれている場合、改行が（ RTF 出力を除く）出力にて表示されます。このような XPath 条件式の例を以下に示します:

```
translate('a;b;c', ';', codepoints-to-string(13))
```

自動計算の移動

自動計算をクリックして選択し、新たな場所へドラッグすることで、自動計算を別の場所に移動することができます。更に、自動計算の切り取りやコピー、貼付けや移動を行うこともできます。新たな場所のコンテキストノードが異なる場合、XPath 条件式を変更する必要がある点に注意してください。

重要なポイント

以下の点に注意してください：

- 自動計算はデザインドキュメント内にある任意の場所に挿入することができます。
- 自動計算を挿入する箇所により、自動計算で使用されるXPath 条件式のコンテキストノードが決定されます。
- Authentic View では、計算に関連する値（つまりXPath 条件式内に含まれているノード）のどれかが変更されるたびに自動計算が再評価されます。
- 自動計算の結果は Authentic View を始めとする全ての出力ビューで編集不可能となります。
- 自動計算の結果を使用することで、XML ドキュメント内にある任意のノードを更新することができます。

関連項目

- 挿入 | 自動計算
- XPath ダイアログ
- XSLT エンジンの情報
- サンプル注文書
- 自動計算によりノードを更新する

(C) 2015-2021 Altova GmbH

7.1.2 自動計算によりノードを更新する

メインXMLドキュメント(Authentic Viewにて編集されるドキュメント)内にあるノードへの自動計算の値(または結果)は、以下の方法によりコピーすることができます。

1. 自動計算の編集と移動に記されている方法により、自動計算を作成します。
2. 自動計算を選択し、プロパティサイドバーの Authentic グループにある Authentic の XML ノードを更新にて編集ボタンをクリックします(以下のスクリーンショットを参照)。

3. XPath セレクターダイアログが表示されるので、更新するノードを選択して、「OK」をクリックします(以下のスクリーンショットを参照)。XPath 条件式に(ドキュメントルートから始まる)絶対値を使用する場合、絶対 XPath のチェックボックスにチェックを入れてください。

他にも、ユーザー定義 XPath ラジオボタンを選択することで、任意の XPath 条件式により更新するノードを選択することができます。

XPath 条件式により選択されるノードは１つでなければなりません。複数のノードがXPath 条件式により選択される場合、ノードの更新は行われません。

重要な情報!

メイン XML ドキュメント内部にあるノードを更新するには、以下の2つの条件を満たす必要があります:

1. 自動計算の XPath 条件式内に、XML ノードに関連した値（つまり Price * 1.2 といった動的な値）を少なくとも1つ含める必要があります。この条件式には Price 要素が含まれており、動的となります。XPath 条件式に静的な値（例えば string("Nanonull, Inc.")）が含まれている場合、XML ノードの更新機能は動作しません。
2. XPath 条件式にて使用される全てのノードは Authentic View にて修正されなければなりません。従って、XPath 条件式が Price*1.2 となっており、自動計算により VATPrice ノードが更新される場合、VATPrice ノードを更新するには Price 要素の更新をAuthentic View にて行う必要があります。

更新の設定とその解除を行う

更新するノードを変更するには、自動計算のAuthentic グループにある Authentic のXML ノードを更新の編集ボタンをクリックして、表示されるXPath セレクターダイアログにて目的のノードを選択します。プロパティの値を削除するには、プロパティウィンドウのツールバーにある削除ボタンをクリックしてください。

自動計算を使った表示と更新されたノードコンテンツによる表示?

条件テンプレートが使用されていない場合、自動計算と更新されたノードのコンテンツの、両方の方法で表示を行うことができます。ノードの更新は自動計算の評価が行われた直後になされるため、他の要素が含まれること無く更新が行われる場合、どちらの方法でも同じ結果を得ることができます。しかし、コンテンツの表示を行うにあたり、複数の方法が存在することに留意するようにしてください。

自動計算を隠す

自動計算を使用した計算を行い、自動計算内に含まれる値によりノードの更新を行う必要がある場合もあります。この場合、以下のようなシナリオが考えられます:

- 結果の表示を1度だけ行う: 自動計算の評価を行うため、それを隠すことができないような状況です。条件テンプレートが加えられない場合では、自動計算を表示して、更新されたノードのコンテンツを表示しないという方法が有効になります。
- 結果の表示を行うこと無く、ノードの更新だけに使用する: このような状況では、自動計算にテキストフォーマットを適用することで、出力メディアでその値が表示されないようにすることができます(例えば、背景色が白色の場所にてテキストの色を白色に指定するなど)。

関連項目

- 自動計算の編集と移動
- 更新されたノードをベースにした自動計算
- サンプル注文書
- XPath ダイアログ

7.1.3 更新されたノードを使用する自動計算

ある自動計算により更新されたノードを使用する(2番目の) 自動計算を作成することができます。以下にある2つの状況が考えられます:

- 2つの自動計算が同一のテンプレート内に含まれている
- 2つの自動計算が異なるテンプレート内に含まれている

同一テンプレート内にある複数の自動計算

2つの自動計算が同じテンプレート内部に配置されている場合、SPS により以下の処理が適用されます:

1. 最初の自動計算の XPath 条件式にて使用されたノードが修正されます。
2. XML ドキュメント内にある全てのノード値が読み取られ、全ての自動計算が処理されます。
3. 最初の自動計算が正常に実行された場合、その XPath 条件式にて指定された XML ノード(ここでは仮に Node-A とします)が更新されます。Node-A をベースにした2番目の自動計算も実行されますが、そこでは更新される前の Node-A の値が使用されます。これは Node-A の値が更新前に読み取られたためで、それ以降読み取られることがなかったためです。
4. ドキュメントが編集されるか、ドキュメントのビューが(Authentic View) とは別のものに)切り替えられると、ノードの値が再度読み取られ、自動計算が実行されます。

5. これにより2番目の自動計算が実行されます。自動計算によりノードの更新が行われる場合、XPath 条件式にて使用されるノードを更新前に変更する必要があります。

Node-A の更新と、Node-A にて更新された値を使った2番目の自動計算の評価により生じるタイムラグはAuthentic View ユーザーにとって分かりにくいものかも知れません。このような状況が発生しないようにするには、2番目の自動計算に含まれるXPath 条件式には、(更新されたノードそのものではなく)最初の自動計算で使用されるXPath 条件式を含めるようにしてください。これにより、2番目の自動計算でも最初の自動計算で使用される入力が使用される様になり、その計算を自動計算内のパーツとして処理することができます。これによりNode-A のコンテンツとは独立したかたちで2番目の自動計算が評価されることになります。

例

種類ごとに税率が異なる製品の販売価格を計算する例を考えてみます。最初の自動計算では(i) 製品本体の値段と、(ii) 付加価値税(VAT) の税率から製品の付加価値税の額を計算し、付加価値税の額を表すノードを更新します。2番目の自動計算では、製品本体の額とVAT の税金を合わせた額が計算され、実際の販売価格を含むノードが更新されます。

- VAT の額を計算する自動計算は NetPrice * VATRate div 100 となります。製品の VAT の税率が入力されると自動計算が実行され、VATAmount ノードが更新されます。
- 製品の販売価格を計算する自動計算がNetPrice + VATAmount となっている場合、更新される前のVATAmount に含まれていた値が自動計算に使用されます。
- しかし、販売価格の計算に使用する自動計算がNetPrice + (NetPrice * VATRate div 100) となっている場合、VATRate の値により販売価格を表すノードが更新されます。更新されたVATAmount ノードは2番目の自動計算とは独立したかたちで更新されます。

詳細な例については、次を参照してください: サンプル:注文書.

異なるテンプレートを使った自動計算

2つの自動計算が異なるテンプレートに配置されている場合、最初の自動計算により更新されたノードを2番目の自動計算にて使用することができます。これは、自動計算の計算とノードの更新がテンプレートごとに別々に行われるためです。この機能がどのように使用されるかは、次を参照してください: サンプル:注文書.

関連項目

- 自動計算の編集と移動
- 自動計算によりノードを更新する
- サンプル:注文書
- XPath ダイアログ

7.1.4 サンプル:注文書

(マイ) ドキュメントフォルダー 以下の`C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\Auto-Calculations\` フォルダーにある Invoice.sps では、以下にあるような自動計算の使用方法を確認することができます:

- ノードの数を数える
- Authentic View ユーザーからの入力に従いノードを選択する
- 自動計算の結果により、ノードのコンテンツを更新する
- ある自動計算における結果を別の自動計算で使用する

サンプルファイルでは、自動計算が黄色の背景色によりハイライトされます(以下のスクリーンショットを参照)。

(C) 2015-2021 Altova GmbH

ノードの数を数える

注文書のサンプルでは、リストにある製品が、顧客により指定されたリスト内にある順序に従い番号付けされます(Product 1 やProduct 2 など)。この番号付けは、自動計算により行われます(以下のスクリーンショットを参照)。

Product 1:	Learning XMLSpy
Net price:	€ 35.00
Category:	Book
VAT:	10%
Price including VAT:	€ 38.5
Product 2:	Scooby Doo's Greatest Hits

この場合、XPath 条件式のposition() 関数により番号を取得することができます。その他にも、それまで処理された兄弟ノードの数を数え、1 を加えることでも番号を取得することができます。この場合 count(preceding-sibling::Product)+1 というXPath 条件式が使用されます。後者の方法は、position() 関数を使用するのが難しいような(または不可能な)状況で使用することができます。製品を削除し、新たな製品を追加することで、サンプルファイル内にある自動計算の動作を確認することができます。

ユーザーからの入力に従いノードを選択する

注文書のサンプルでは、ユーザーがコンボボックスの値を選択することで、製品のカテゴリー(Book、CD、DVD、またはElectronics)が選択されます。この選択結果は、XML ドキュメントの//Product/Category 内に入力されます。自動計算は、この値を使用することでXML ドキュメントへの参照を行い、対応する製品カテゴリーのVAT 税率が取得されます。この自動計算のXPath 条件式は以下のようになります:

```
for $i in Category return /Invoice/Categories/Category[. = $i]/@rate.
```

出力では、自動計算の位置にVAT の税率が表示されます。注文書のサンプルでは、注文書のデータが保管されるXML ドキュメントにルックアップテーブルも保管されます。このようなテーブルは別のドキュメントにて管理することも可能で、その場合、XPath 2.0 のdoc() 関数によりアクセスすることになります。製品のCategory コンボボックスにて製品の種類を変更し、それぞれの種類に対して異なるVAT の値が表示されることで、自動計算の動作を確認することができます(Book=10%、CD=15%、DVD=15%、Electronics=20%)。

自動計算の結果によりノードのコンテンツを更新する

自動計算により取得されるVAT の税率は動的なもので、(Authentic View にて使用するために)一時的にメモリ内部に保管されます。しかし、自動計算の結果はXML ドキュメントのVAT ノード内部に保管することができます。これにより、(i) VAT ノードのコンテンツは自動計算により自動的に入力されるため、ユーザーが入力する必要が無くなり、(ii) ルックアップテーブルが修正されるとInvoice.xml を Authentic で開くことで、その変更がVAT ノードへ反映されます。自動計算によりノードを更新するには、デザインビューにある自動計算を選択し、プロパティウィンドウにあるXML ノードボタンをクリックします(以下のスクリーンショットを参照)。表示されるダイアログにてVAT ノードを選択し、OK をクリックします。

Authentic View にて別の製品カテゴリーがコンボボックスにて選択されると、ルックアップテーブルへの参照が行われ、VAT の税率が自動計算により取得されます。その後、VAT の税率が表示され、VAT ノードが更新されます。

自動計算における結果を別の自動計算で使用する

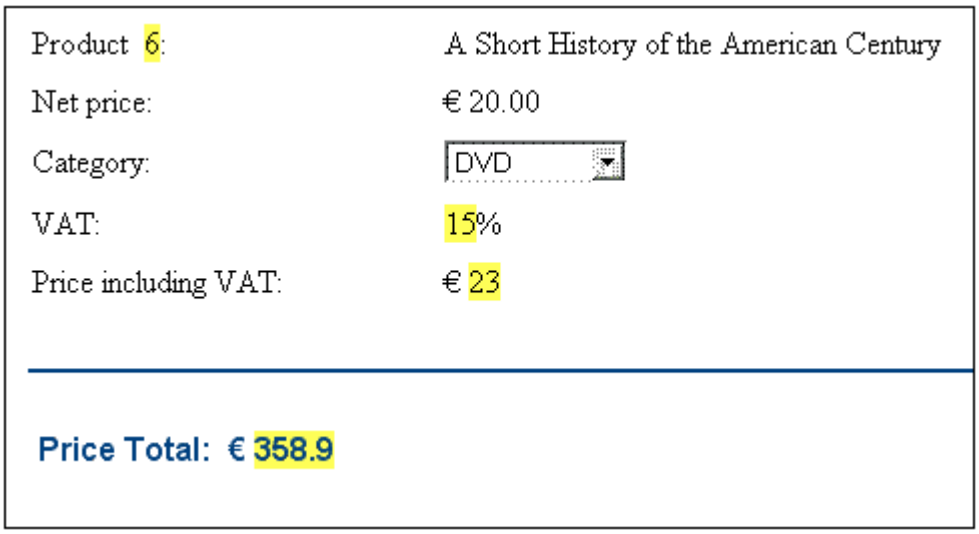
上の例で自動計算により取得された VAT の税率から、各製品に対して販売価格（本体価格 + VAT の税金額）を計算する必要があります。使用する式は以下のようになります:

販売価格 = 本体価格 + *VATの税金額*
VATの税金額 = 本体価格 ＊ *VATの税率 ／ 100* となるため
販売価格 = 本体価格 +（本体価格 ＊ *VATの税率 ／ 100*）

製品の本体価格は PriceNet ノードから取得することができます。VAT の税率は上記の方法を使って自動計算により計算され、この自動計算により、VAT ノードが更新されます。従って、VAT ノードのコンテンツを自動計算に使用することで、販売価格を計算することが可能になります。以下の XPath 条件式を使用することができます:

```
PriceNet  + (PriceNet * (VAT div 100))
```

XPath 条件式は、プロパティウィンドウから確認並びに編集 することができます。本体価格や製品のカテゴリーを変更することで、自動計算が動作していることを確認することができます。製品の（VAT を含む）販売価格が変わることに注目してください。



SPS では、販売価格の自動計算により XML ドキュメント内にある PriceGross ノードが更新されます。

更新されたPriceGross ノードは、別の自動計算を使用することで、購入された全製品の合計金額を計算するために使用することができます。使用されるXPath 条件式はsum(Order/Product/PriceGross) となります。注文書 SPS では、この自動計算によりPriceTotal ノードが更新されます。個々の製品の金額を変更することで、自動計算の動作を確認することができます。

自動計算の演習

それでは2つの自動計算コンポーネントをSPS へ追加してみましょう:

1. 注文全体から割引を計算する自動計算を作成します。注文の合計額が100ユーロ、300ユーロ、または600ユーロを超えた場合、それぞれ 5%、10%、そして12% の割引が適用されるようにします。割引額を(以下のスクリーンショットにあるように)表示して、XML ドキュメントの DiscountAmount ノードを更新するようにしてください。
2. 割引後の価格を計算するための自動計算を作成します。販売価格から割引額を引いた値が割引後の価格となります。実際に請求される価格を(以下のスクリーンショットにあるように)表示して、XML ドキュメントの BillAmount ノードを更新するようにしてください。

これらのコンポーネントをセットアップすることで、以下のスクリーンショットにあるようにAuthentic View の出力が行われます。

Price Total:	€	358.90	**Volume Discounts**
Less Volume Discount:	€	35.89	Over € 100 = 05% Over € 300 = 10%
Bill Amount:	€	323.01	Over € 600 = 12%

(マイ) ドキュメントフォルダー 以下の`C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\Auto-Calculations` フォルダーにあるInvoiceWithDiscounts ファイルにて、これら2つの自動計算を確認することができます。

関連項目

- 自動計算の編集と移動
- 自動計算によりノードを更新する
- 更新されたノードをベースにした自動計算
- XPath ダイアログ

7.2 条件

メインテンプレートとグローバルテンプレートの両方で、デザイン内部の任意の場所に条件を挿入することができます。条件とは1つ以上の分岐から構成されるSPS コンポーネントのことで、各分岐がXPath 条件式により定義されます。例えば、2つの分岐からなる条件を考えてみましょう。最初の条件分岐にあるXPath 条件式により、コンテキストノード上にあるLocation 属性の値が"US" であるかがチェックされます。2番目の条件分岐にあるXPath 条件式では、Location 属性の値が"EU" であるかがチェックされます。各分岐には、テンプレート(条件テンプレート)が含まれています。条件を伴うノードが処理され、結果が真となる最初の条件(条件テンプレート)が処理されます。条件内にある処理が終了すると、その他の分岐にある条件は評価されません。この機能を使用することで、ノードの値により異なるテンプレートを使用することが可能になります。ここで紹介した例では、US とEU という値により異なるテンプレートを使用することができます。

このセクションには、以下のトピックが含まれています:

- 条件のセットアップ: 条件と分岐の作成方法について記述されます。
- 条件の編集: 条件分岐が作成された後に、XPath 条件式を編集する方法について記述されます。
- 特定の出力に対する条件 : 条件を使用することで、異なる出力フォーマットに対して個別の出力を生成する方法について記述されます。
- 条件と自動計算 : 条件と自動計算を組み合わせて使用する際に注意すべき点について記述されます。

関連項目

- クイックスタート チュートリアル: 条件を使用する

7.2.1 条件のセットアップ

以下にあるステップにより、条件を作成することができます:

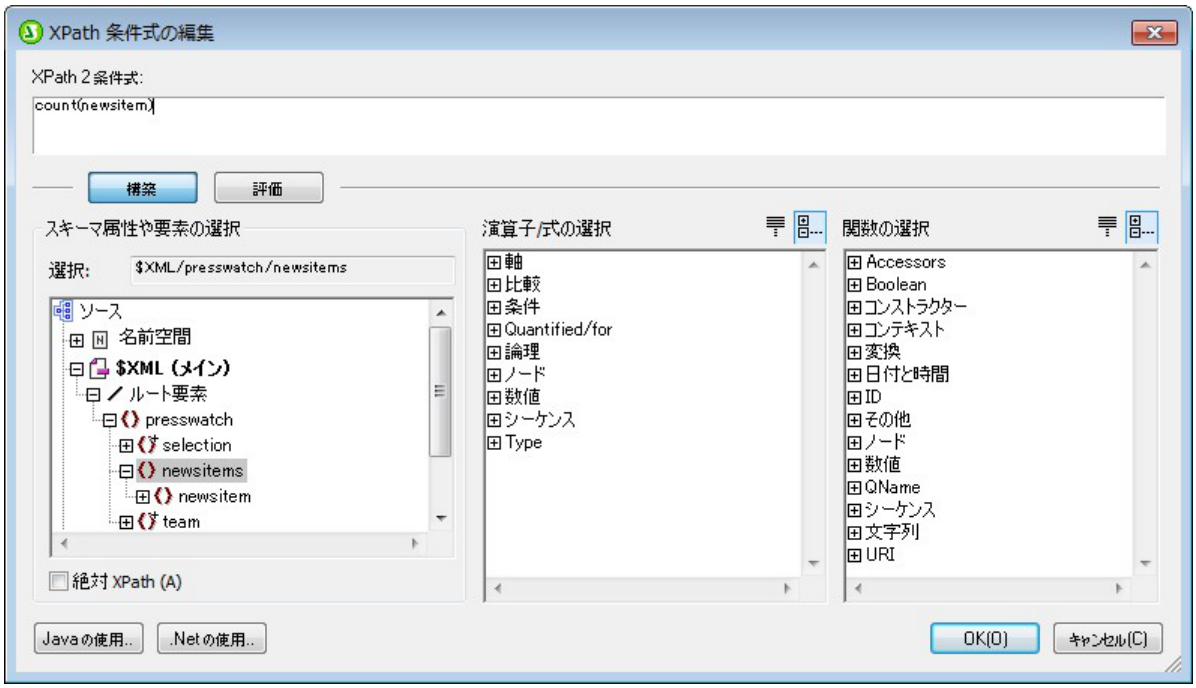
1. 最初の分岐とともに、条件を作成する。
2. 必要に応じて新たな条件分岐を作成する。
3. 条件の各分岐内にテンプレートを作成し、編集する。

最初の分岐と共に条件を作成する

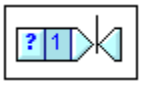
以下の操作により条件をセットアップすることができます:

1. デザイン内の任意の場所にカーソルを配置するか、コンポーネントが選択された状態で、メニューオプションから「**挿入 | 条件の挿入**」を選択します。XPath 式 の編集ダイアログ が表示されます(以下のスクリーンショットを参照)。

(C) 2015-2021 Altova GmbH



2. 条件式ペインでは、その条件分岐に対する XPath 条件式をキーボードから入力することができます。その他にも、ダイアログのペインにあるノード、演算子、または関数をダブルクリックすることで、それらを入力することができます。ノードが挿入されたコンテキストノードについて留意する必要があります。コンテキストノードは、ダイアログが表示された時にスキーマソースツリーにてハイライトされます。
3. 「OK」をクリックして完了します。最初の条件分岐とともに条件が作成され、先程作成した XPath 条件式が最初の条件分岐に使用されます。何も選択されていない状態で条件が挿入されると、最初の分岐が空の状態で表示されます（テンプレートが無い状態で表示されます – 以下のスクリーンショットを参照）。コンポーネントが選択された状態で条件が挿入されると、コンポーネントの周りに条件が作成され、そのコンポーネントが最初の条件分岐におけるテンプレートとなります。

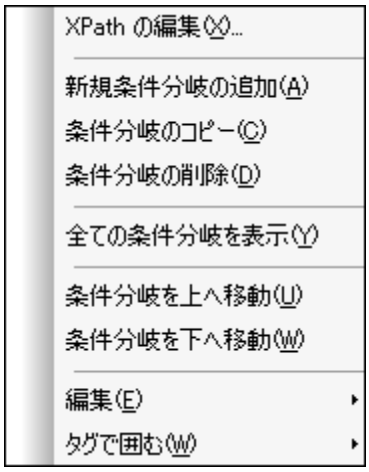


条件全体を選択するには、クエッションマーク (?) が表示されているセルをクリックします。最初の分岐を選択するには、1という番号が表示されているセルをクリックします。

最初の分岐を伴う条件を作成すると、（その分岐内にテンプレートが存在するかにかかわらず）必要な数だけ条件分岐を作成することができます。

新たな条件分岐を作成する

条件分岐は新たに1つずつ作成することができます。新たな分岐はコンテキストメニューから作成することができ、(i) テンプレートが含まれていない状態で作成（**新規条件武器の追加**）、または (ii) 既存のテンプレートをコピーして新たな条件分岐を作成（**条件分岐のコピー**）することができます（以下のスクリーンショットを参照）。



新たな分岐を作成するには、条件分岐のどれかを右クリックして、コンテキストメニューから**新規条件分岐の追加**を選択します。XPath 式 の編集ダイアログ が表示されます。XPath 条件式を入力して「OK」をクリックすると、空の分岐が条件へ新たに加えられます。新たな条件分岐が追加されると、条件に新たなセルが追加され、追加された条件には、追加される前に表示されていた分岐に 1 が加えられた数が表示されます。

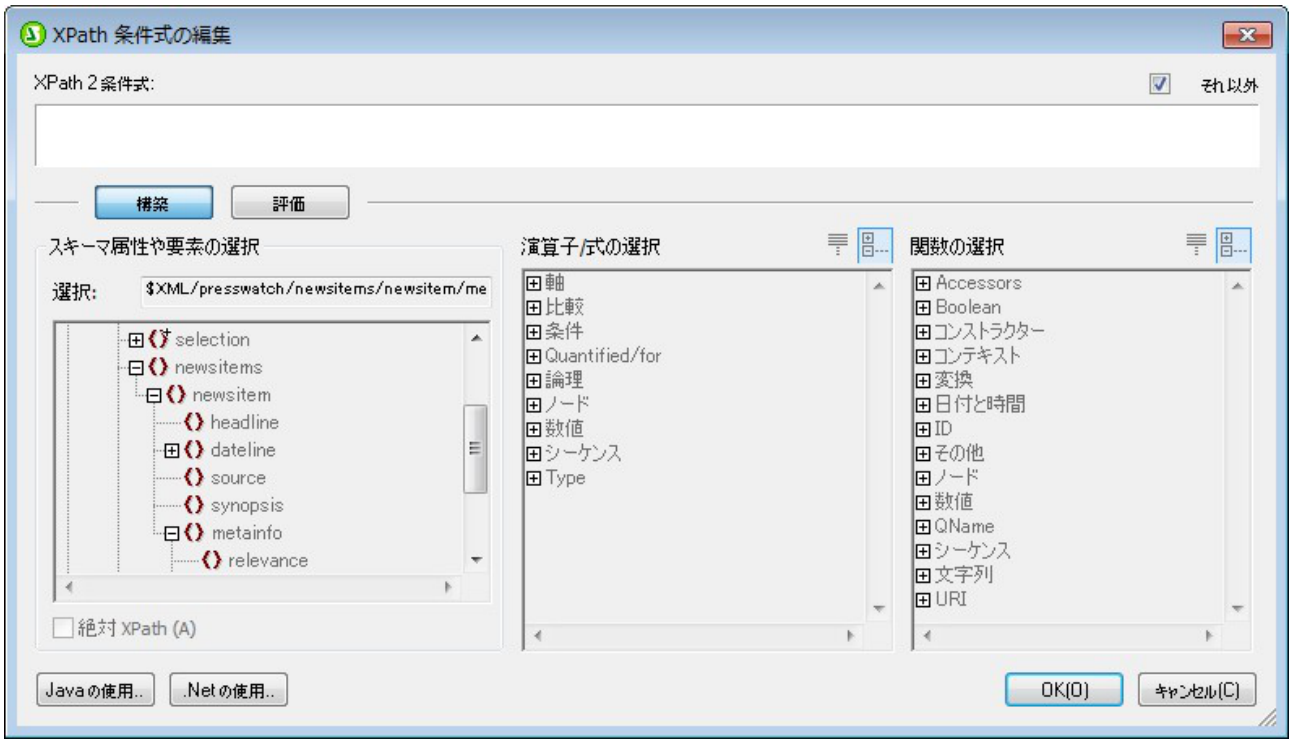
既存の条件分岐からのコピーを作成するには、コピーする条件分岐を右クリックして、コンテキストメニューから**条件分岐のコピー**を選択します。コピーされた条件分岐を含む XPath 条件式とともに、XPath 式 の編集ダイアログ が表示されます。XPath 条件式の編集を終え、「OK」をクリックすると、新たな分岐が条件に追加されます。新たな条件分岐には、コピー元の分岐に含まれていたテンプレートが含まれます。追加された条件には、追加される前に表示されていた分岐に 1 が加えられた数が表示されます。

それ以外の分岐

それ以外の分岐には、他のどの条件分岐も処理されなかった際に処理されるテンプレートが含まれます。その他の条件分岐が定義されていない場合、考えられる全ての事象に条件分岐を対応させるか、どの分岐も処理されないまま条件の処理が終了しても良いようにデザインを作成する必要があります。

その他の条件分岐を挿入するには、上に記されているように**新規条件分岐を追加**するか、**条件分岐のコピー**を行い、XPath 式 の編集ダイアログ にて、それ以外のチェックボックスにチェックを入れます(以下のスクリーンショットを参照)。

(C) 2015-2021 Altova GmbH



分岐の優先度を移動する

条件では、最初に真と評価された分岐が処理され、それ以外の分岐は処理されないため、条件内にある分岐の順序が重要な意味を持つことになります。分岐の位置を変更するには、移動する分岐を選択し、右クリックにより表示されるコンテキストメニューから**条件分岐を上へ移動**または**条件分岐を下へ移動**を選択します。

分岐の削除

条件分岐を削除するには、削除する分岐を選択し、右クリックで表示されるコンテキストメニューから**条件分岐の削除**を選択します。

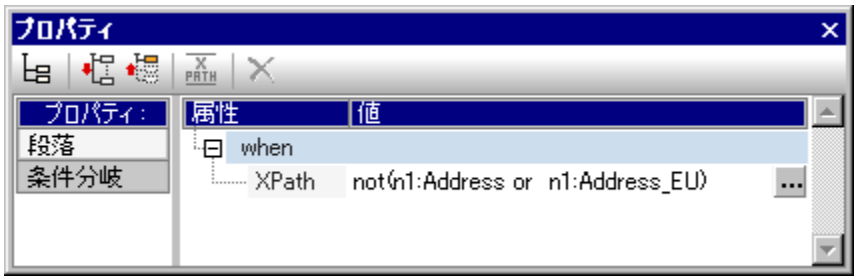
関連項目

- クイックスタート チュートリアル: 条件を使用する
- 条件の編集

7.2.2 条件の編集

以下の操作により、条件分岐のXPath 条件式を編集することができます:

1. （条件ではなく）条件分岐を選択します。
2. プロパティサイドバーにて、条件分岐を選択します（以下のスクリーンショットを参照）。



3. グループの XPath プロパティにある**編集**ボタン ... をクリックします。 XPath 式 の編集ダイアログが表示され、選択した条件分岐のXPath 条件式を編集することができるようになります。

関連項目

- クイックスタート チュートリアル: 条件を使用する
- 条件のセットアップ

7.2.3 出力をベースにした条件

StyleVision のドキュメントデザイン内にある個々のコンポーネントは、（Authentic View、RTF とHTML）出力フォーマットに応じて異なる処理を行うことができます。例えば、Authentic View ではローカルのファイルシステムが指し示され、HTML 出力ではウェブページへ移動することができるリンクを作成する場面を考えてみましょう。この場合、最初の分岐で Authentic View 出力に対するコンテンツが記述され、次の分岐で HTML 出力に対する処理が記述された条件を作成することができます。その他にも、HTML 出力には表示されないテキストを Authentic View の出力に加える場面を考えてみましょう。条件を作成することで、Authentic View 出力の処理だけにテキストを加え、HTML 出力にはそれが表示されないようにすることができます。

メモ ドキュメント内にある個々のパーツやコンポーネントの周りに、目的の出力に対する条件を追加することで、各出力に対して異なるドキュメント構造を簡単に作成することができます。

特定の出力に対して条件を作成する

以下の操作により、特定の出力に対する条件を作成することができます:

1. 出力フォーマットに従って異なる表示を行うコンポーネントをデザインビューにて選択するか、ドキュメントのパーツをハイライトします。
2. 右クリックして表示されるコンテキストメニューから、「**タグで囲む | 出力ベース条件**」を選択します。この操作により、3つの分岐を伴う条件が挿入され、各分岐には同一のコンテンツ（選択されたコンポーネント）が含まれます。各条件分岐が、対応する出力（（Authentic View、RTF または HTML）を表すことになります。各分岐へマウスを移動させるか、選択された分岐の XPath 条件式を（プロパティサイドバーの条件分岐エントリーにて編集ボタンをクリックし）確認することにより、どの分岐がどの出力に対応しているのか確認することができます。
3. 各分岐では、目的の処理をそれぞれ定義することができます。特定の出力フォーマットに対して処理を行いたく無い場合は、そのフォーマットに対応した分岐を削除することができます（その分岐を選択し、**Delete** キーを押下するか、右クリックして表示されるコンテキストメニューから、**条件分岐の削除**を選択します）。

メモ 出力をベースにした条件を最初に作成し、各分岐に対して（静的または動的な）コンテンツを挿入することもできます。まずは出力をベースとして条件をカーソル位置に挿入し、対応する条件分岐内部に必要な静的または動的なコンテンツを挿入していきます。

(C) 2015-2021 Altova GmbH

出力ベースの条件の分岐を編集

出力をベースにした条件のXPath 条件式は、$SV_OutputFormat = 'format' となっており、format の部分には、Authentic、RTF または HTML のどれかが入ります。条件分岐のXPath 条件式は、プロパティサイドバーにある条件分岐エントリーの編集ボタンをクリックします することで編集することができます。例えばAuthentic View HTML の出力フォーマットを組み合わせることで、1つの条件分岐で表すことができるようになります(XPath 条件式は $SV_OutputFormat = 'Authentic' or $SV_OutputFormat = 'HTML' となります)。

更に、(i) 1つまたは複数の分岐を削除し、(ii) 条件にゴ それ以外」の分岐を作成し、(iii) 各分岐の位置を移動することで、分岐の優先度を変更することができます。これらの操作を行う方法については、次を参照してください: 条件のセットアップ と条件の編集。

$SV_OutputFormat パラメータを使用する

各出力に対して生成されたXSLT ファイルには、$SV_OutputFormat がグローバルパラメーターとして作成され、その出力フォーマットに値(つまりAuthentic、RTF またはHTML)が与えられます。このパラメーターへは、別の値を与えることでランタイムにてオーバーライドすることもできます。これにより、例えばランタイムにて選択されるHTML 出力を、通常のHTML 出力とは別に作成することも可能になります。$SV_OutputFormat = 'HTML-1' ならびに $SV_OutputFormat = 'HTML-2' という値を含む条件分岐を作成し、ランタイムにて必要な値(HTML-1 またはHTML-2)をプロセッサーへ与えることで、定義された処理が行われます。

関連項目

- クイックスタート チュートリアル: 条件を使用する
- 条件の編集

7.2.4 条件と自動計算

条件と自動計算を組み合わせて使用する場合、いくつか留意するべきことがあります。重要な点を以下に示します:

- **表示されている条件**(つまり真として選択された分岐)にある自動計算だけが評価されます
- 自動計算は条件の前に評価されます

上記の点をガイダンスとしてまとめたものを以下に示します:

1. 自動計算によりノードが更新され、そのノードが条件に含まれている(条件分岐の XPath 条件式に含まれているか、条件添付がのコンテンツ内に含まれている)場合、自動計算ができる限り条件の外で行われるようにしてください。これにより、自動計算が常に(つまりどの条件分岐が表示されているかに関わらず)表示されるようになり自動計算が行われた際に、ノードが常に更新されるようになります。自動計算が条件分岐の中にあり、その分岐が表示されない場合、ノードも更新されません。
2. 自動計算を条件の内部に配置する必要がある場合、(i) それが条件内にある全ての分岐に配置されるようにして、(ii) 全ての状況に対して条件の分岐が対応できるようにする必要があります。条件テンプレートにてカバーされない事象が仮に存在する場合、(自動計算が表示されているテンプレートに配置されないことにより)自動計算が行われないというリスクが発生します。
3. 条件に応じて異なる自動計算を使用したい場合、全ての分岐において自動計算が使用されていることを確認してください。
4. 条件テンプレートにて条件の順序は重要な意味を持つことに留意してください。最初に真と評価された条件分岐が実行されます。**それ以外**の条件は、定義された条件分岐が全て偽と評価されたときに実行されます。

関連項目

- クイックスタート チュートリアル: 条件を使用する
- 条件の編集
- 自動計算

(C) 2015-2021 Altova GmbH

7.3 条件付き表示

条件付き表示機能により、指定された条件が満たされた時に表示されるデザインコンポーネントを作成することができます。条件付き表示を行うことのできるデザインコンポーネントを以下に示します:

- ハイパーリンク
- ブックマーク
- 新規ドキュメント
- ユーザー定義要素
- 目次 レベル

条件付き表示を指定する

以下の操作により、上にあるデザインコンポーネントを条件付き表示により作成し、その条件を指定することができます:

1. デザイン コンポーネントを選択します。
2. プロパティサイドバーの共通プロパティグループにて、条件付き表示の XPath を編集アイコン ... をクリックします(以下のスクリーンショットを参照)。

3. 表示される XPath 式 の編集ダイアログ にて、そのデザインコンポーネントが出力にて実装されるために必要な条件を入力します。
4. 「OK」をクリックして完了します。

メモ 条件が満たされた場合、そのデザインコンポーネントが実装されます。条件が満たされない場合、デザインコンポーネントを伴わないかたちで、デザインコンポーネントのコンテンツだけが出力されます。例えば、上のスクリーンショットではハイパーリンクに対して条件付き表示が指定されています。条件により、カレントノードが空でないかがチェックされます。ハイパーリンクのテキストは、ハイパーリンクデザインコンポーネントのコンテンツから派生されます(ハイパーリンクの URL はプロパティのハイパーリンクグループにて指定されます)。条件の評価結果が偽となった場合、(ハイパーリンクデザインコンポーネントから得られた)ハイパーリンクのテキストは出力されますが、ハイパーリンクとしてではなくプレーンテキスト(平文)として出力されます。

同様に、その他のデザインコンポーネントにも条件付き表示として作成することができます。条件付き表示の対象となるのはデザインコンポーネントそのもので、そのデザインコンポーネント内にあるコンテンツは条件の評価が真か偽かに関わらず作成されます。

関連項目

- クイックスタート チュートリアル: 条件を使用する
- 条件

(C) 2015-2021 Altova GmbH

7.4 グループ化

グループ化関数は XSLT 2.0 と3.0 SPS にて利用することができ、HTML とRTF 出力に対して適用されます。**グループ化機能は Authentic View 出力にてサポートされません。**

グループ化により、アイテム(大抵の場合ノード)をグループ単位で処理することができるようになります。車の在庫管理を例に考えてみましょう。各車の詳細情報は、car 要素以下に収められます。例えば、car 要素にbrand 属性がある場合、ブランド名により車をグループ化することができます。この機能により、以下のような処理を行うことが可能になります:

あるブランドの車全てをまとめて出力に表示し、その上部にブランド名のヘッダーを表示する。
グループ内部で処理を行うことにより、その処理結果をグループごとで個別に表示する。例えば、各ブランドで利用可能な車種がどれだけあるかを表示することができます。

更に、グループ内に配置されたサブグループ単位で処理を行うこともできます。例えば、各ブランド内にあるモデルや年式ごとにグループ化を行うことができます。

グループ化の条件

アイテムは、(i) 通常ノードの値をチェックするために使用されるグループ化キー、そして(ii) 相対的なアイテムの位置という2つの条件によりグループ化することができます。以下にあるグループ化の条件を使用することができます:

- *group-by* : XPath により定義されたキーをベースにアイテムのグループ化を行います。例えば、car 要素を、brand 属性をベースにグループ化することができます。car 要素に対してグループ化を行い、XPath 条件式により brand 属性が選択されます。
- *group-adjacent* : グループ化のキーと位置情報を組み合わせることでグループ化を行います。同じグループ化のキーを持った隣合わせのアイテムがグループ化の対象になります。あるアイテムにあるグループ化のキー値が、前にあるアイテムの値と違う場合、そのアイテムから新たなグループが作成されます。
- *group-startup-with* : 定義された XPath にマッチするノードが見つかると、新たなグループを作成します。定義された XPath パターンにノードがマッチしない場合、そのノードが現在のグループに追加されます。
- *group-ending-with* : 定義された XPath にマッチするノードが見つかると、そのノードがグループ最後のノードとなります。次に処理されるノードが新たなグループ最初のノードになり、その次に処理されるノードが定義された XPath パターンにマッチしない場合、そのノードがグループに追加されます。

グループの作成

グループは、コンテキストメニューを介してノード、または現在のグループテンプレートにて作成することができます。グループを作成するには、ノードまたは現在のグループテンプレートを右クリックして、コンテキストメニューからグループ化コマンドを選択します。出力グループを作成ダイアログが表示されます(以下のスクリーンショットを参照)。

ダイアログでは、グループ化を有効のチェックボックスにチェックを入れ、必要となるグループ化の種類をラジオボタンから選択します。XPath 条件式を入力することにより、グループ化キーを定義(group-by ならびにgroup-adjacent の場合)することができるほか、マッチを行うためのパターンを入力(group-starting-with ならびにgroup-ending-with の場合)することができます。「OK」をクリックすると、グループセットをアルファベット順(昇順)で並べ替えるか選択するよう促されます。グループセットの並べ替えは後から変更することができるほか、並べ替えを削除することもできます。以下のスクリーンショットでは、ノードと現在のグループテンプレートからなるグループ化の様子が示されます。

上のスクリーンショットでは、person ノードがグループ化され、作成されるグループはアルファベット順に並べ替えられます。例えばperson 要素がdepartment によりグループ化された場合、アルファベットの昇順によりdepartment 要素の並べ替えを行うことができます。

(C) 2015-2021 Altova GmbH

グループの並べ替え

グループの定義を完成させると、アルファベットの昇順によりグループの並べ替えを行うか尋ねられます。グループの並べ替えはいつでも行うことができるほか、ここで作成した並べ替えの設定を削除することもできます。

並べ替えの設定、修正、または削除を行うには、目的のグループ化テンプレートを右クリックして**並べ替え**を選択します。この操作により出力の並べ替えの順序を定義ダイアログが表示されます。このダイアログの使用方法については、並べ替えのセクションを参照ください。グループ化キーによりグループの並べ替えを行うには、並べ替えキーにcurrent-grouping-key() XPath 関数を使用しなければなりません。詳細については、このセクション以下にあるサブセクションを参照ください。

グループ化と並べ替え設定の確認と編集

テンプレートにおけるグループ化ならびに並べ替え設定を確認、または編集するには、テンプレートを右クリックして、**グループ化**または**並べ替え**をそれぞれ選択します。該当するダイアログが表示されるため、設定の確認や修正を行うことができます。

ユーザー定義テンプレート

ユーザー定義テンプレートにより、指定したXPath 条件式により選択されるアイテムに対してテンプレートを適用することができます。ユーザー定義関数のXPath 条件式により選択されたノードをグループ化することもできます。この場合、グループ化はユーザー定義テンプレートに対して適用されます。

関連項目

- 出力の構造
- 並べ替え
- ユーザー定義テンプレート
- サンプル Group-By (Persons.sps)
- サンプル Group-By (Scores.sps)

7.4.1 サンプル: group-by (Persons.sps)

Persons.sps サンプルはPersons.xsd スキーマをベースに作成されておりPersons.xml を作業用 XML ファイルとして使用します。ファイルは(マイ) ドキュメントフォルダー以下にある`C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\Grouping\Persons\`に格納されています。ドキュメント要素のemployees 要素には、任意の数のperson 要素を含めることができ、各 person 要素は、以下に示されるような構造を持ちます:

```
<person first="Vernon" last="Callaby" department="Administration" grade="C"/>
```

デザインではdepartment (部門) によるグループ化を行います。各部門は独立したテーブルにより表示され、部門はアルファベット順にて並べ替えられます。各部門テーブル内部では、person により表される従業員が与えられた権限 (grade) に応じてグループ化が行われ、各 grade では従業員 (person) が苗字 (last) により並べ替えられて表示されます。

グループの作成の手順

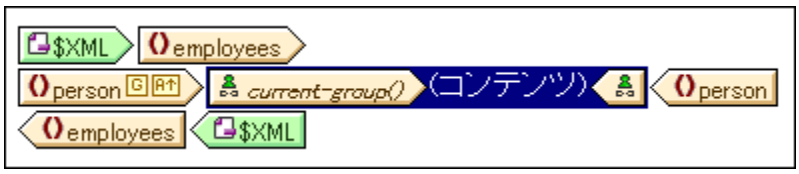
グループを作成するための手順は以下のようになります。まずはgrouping-key をdepartment 属性に対して使用することでperson 要素に対してグループを作成します。この操作によりdepartment 属性の値によるperson 要素のグループ化が行われます。並べ替えが指定された場合、department によるグループをアルファベット順に並べ替えることができます(例えばAdministration が最初に表示され、次にBoard of Directors が表示されます)。部門ごとに独立したテーブルを作成するため、(department grouping-key をベースにした)

現在のグループはテーブルとして作成されます。グループ化された person 要素内部にて、各グループが grade 属性により更にグループ化されるように指定します。

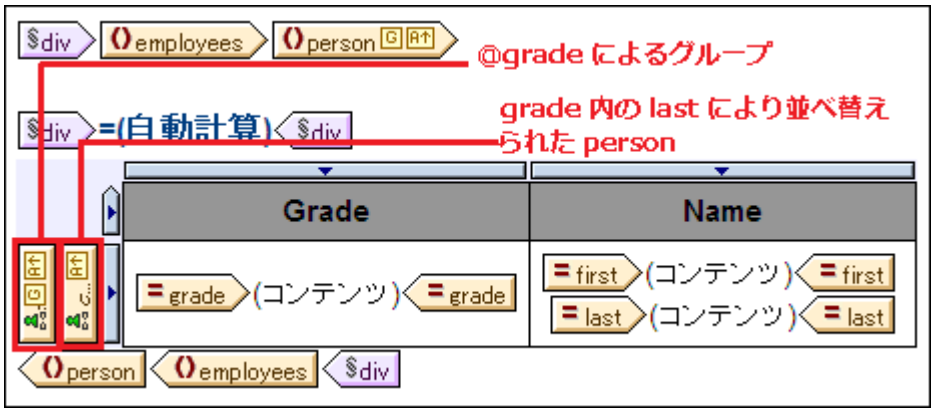
SPS の作成

以下の操作によりデザインを作成します:

1. スキーマツリーから person 要素をドラッグし、コンテンツとして作成します。
2. person 要素のタグを右クリックして、コンテキストメニューからグループ化を選択します。
3. 出力グループを定義ダイアログにて、**group-by** を選択し、マッチテキストボックスにて @department という XPath 条件式を入力し、**はい**をクリックします。
4. グループの並べ替えを行うかを尋ねるダイアログが表示されます。この例ではデフォルトのアルファベット順による並び替えを行うので、**はい**をクリックします(並べ替えはいつでも設定または削除することができます)。
5. 各グループ(この例では department)を独立したテーブル内に作成するため、カレントグループをテーブルとして作成します。以下のスクリーンショットに示されるように current-group() タグを右クリックして、「**変更 | テーブル**」を選択し、子属性の @last と @grade をテーブルのカラムとして選択します。



6. 最初のカラムに @grade が、2番目のカラムに @first と @last ノードが含まれるようにテーブルにあるカラムとセルのコンテンツを修正します(以下のスクリーンショットを参照)。
7. department によりグループ化されたカレントグループにて grade によるグループ化を行うため、current-group() テンプレート上にて grade 属性に対するグループを作成します。デフォルトの並べ替えにより作成の確定を行います。current-group() テンプレートが新たに作成されます(以下のスクリーンショットを参照)。
8. grade により並べ替えられた person のサブグループとなるカレントグループを、last 属性に対して並べ替えます。



9. テーブルのフォーマットをセットします。
10. テーブルの上にテーブルのヘッダーを作成します。個々のテーブルにより部門 (department) が表示されるため、XPath により記述された自動計算をカレントコンテキストで使用することで、部門の名前を動的に取得することができます。XPath 2.0/3.0 の current-grouping-key() 関数を使用することができます。
11. 同様の処理を最初から行い、別の出力を作成します。今度は grade により person のグループ化を行い、その中で department によりグループ化を行います。

テンプレートのグループ化や並べ替えを確認ならびに修正するには、そのテンプレートを右クリックして、コンテキストメニューから**グループ化**または**並べ替え**を選択します。この操作により対応するダイアログが表示され、設定を確認または修正することができます。

関連項目

- 出力の構造
- 並べ替え
- サンプル: Group-By (Scores.sps)

7.4.2 サンプル: group-by (Scores.sps)

Scores.sps サンプルは Scores.xsd スキーマをベースに作成されており、Scores.xml を作業用 XML ファイルとして使用します。ファイルは (マイ) ドキュメントフォルダー 以下にある C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\Grouping\Scores\ に格納されています。ドキュメント要素の results 要素には、任意の数の group 要素と match 要素を含めることができます。各 group 要素には任意の数の team 要素が含まれており、match 要素は以下に示されるような構造を持ちます:

```
<match group="A" date="2007-10-12">
      <team name="Brazil" for="2" points="3"/>
      <team name="Germany" for="1" points="0"/>
</match>
```

デザインは3つのパーツから構成されます (以下のスクリーンショットを参照)。(i) //match/@date によりグループ化された日付ごとの試合結果、(ii) //match/@group によりグループ化されたリーググループごとの試合結果、そして (iii) group 要素から値が動的に生成され、自動計算により必要なデータが自動的に計算されるテーブルにより、リーググループの要約が表示されます。

Match Results: Day-by-Day

2007-10-12

Brazil - Germany	2 - 1
Italy - Holland	2 - 2

2007-10-13

Argentina - France	2 - 0
England - Spain	0 - 0

Match Results: By Group

Group A

Brazil - Germany	2 - 1
Italy - Holland	2 - 2
Brazil - Italy	1 - 2
Germany - Holland	2 - 2
Brazil - Holland	1 - 0
Germany - Italy	1 - 1

Group Tables

Group A

Team	P	W	D	L	F	A	Pts
Brazil	3	2	0	1	4	3	6
Italy	3	1	2	0	5	4	5
Germany	3	0	2	1	4	5	2
Holland	3	0	2	1	4	5	2

グループの作成手順

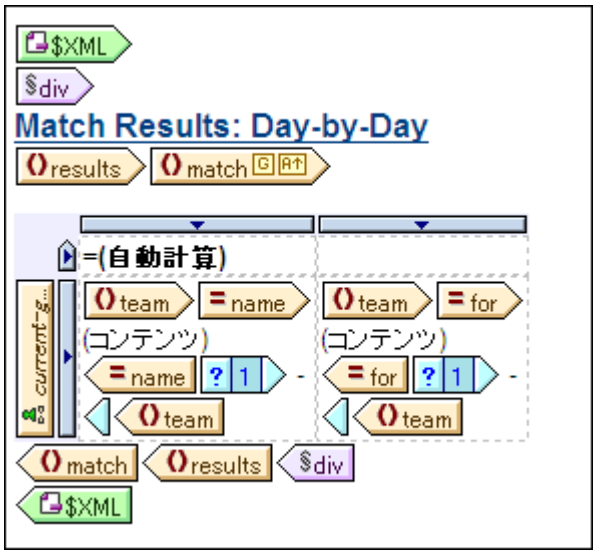
試合結果が表示される2つのセクションでは、日付とトーナメントグループにより試合の結果をグループ化します。データの配置を行うため、日付とトーナメントグループにより作成される各グループのメンバーに対して、境界線が表示されないテーブルを作成します。試合の結果は日付とトーナメントグループごとに異なるテーブルで表示するようにします。Group Tables のセクションでは、group 要素が動的なテーブルとして作成され、自動計算により必要とされるデータの値が計算されます。

(C) 2015-2021 Altova GmbH

SPS の作成

以下の操作によりデザインを作成します:

1. スキーマツリーから /results/match 要素をドラッグし、コンテンツとして作成します。
2. match 要素のタグを右クリックして、コンテキストメニューからグループ化を選択します。
3. 出力グループを定義ダイアログにて、**group-by** を選択し、マッチテキストボックスにて @date という XPath 条件式を入力し、「OK」をクリックします。
4. グループの並べ替えを行うかを尋ねるダイアログが表示されます。この例ではデフォルトのアルファベット順による並べ替えを行うので、**はい**をクリックします(並べ替え条件はいつでも追加または削除することができます)。
5. 各グループ(この例では date)を独立したテーブル内に作成するため、カレントグループをテーブルとして作成します。以下のスクリーンショットに示されるように、current-group() タグを右クリックして、「**変更 | テーブル**」を選択し、子孫ノードの team/@name と team/@for をテーブルのカラムとして選択します。



6. それがカレントグループにおける最後の試合でない場合に出力される、ハイフン (-) を各セルに挿入します。条件テンプレートを作成し、条件には position() != last() を使用します。これにより Brazil - Germany や 2 - 1 といった出力が行われます。
7. ヘッダーに自動計算を追加して、カレントグループのキーを表示するようにします(XPath 条件式には current-grouping-key() を使用します)。
8. 必要に応じてテーブルのフォーマットを行います。
9. トーナメントグループにより試合結果のグループ化を行うには、上記の操作を match の group 属性に対して行います。
10. 各チームの順位が表示されるデザインの最後にあるグループテーブルでは、/results/group 要素を動的テーブルとして作成します。「**テーブル | カラムの追加**」や「**テーブル | カラムの挿入**」コマンドを使うことで、必要に応じてカラムを追加することができます。出力に必要な計算を行う自動計算を各カラムに作成します(白星により 3 ポイント、引き分けで 1 ポイント、黒星で 0 ポイント)。得られた合計得点数により降順でテーブルの並べ替えを行います。これらの結果を得るために使用された XPath 条件式を確認するには、自動計算または並べ替えられたテンプレートを右クリックして、それぞれ **XPath の編集**または**並べ替え**コマンドを選択します。

関連項目

- 出力の構造
- 並べ替え
- サンプル Group-By (Persons.sps)

7.5 並べ替え

並べ替えの機能は HTML とRTF 出力にて使用することができます。**並べ替え機能は Authentic View 出力にてサポートされません。**

並べ替えは、同じ修飾名を持つ兄弟要素ノードのセットに対して、1つ以上のソートキーにより行うことができます。例えば、(Company 要素以下にある)全てのPerson 要素を、Person の子要素であるLastName により並べ替えることができます。ソートキーはノードでなければならず、通常並べ替えが行われる要素ノードの子孫ノード(要素または属性)となります。ここで紹介した例では、LastName がソートキーとなります。

並べ替えの対象となる2つの要素に含まれているソートキーの値が等しい場合、更にソートキーを追加することで、より細かな並べ替えを行うことができます。上で示したPerson 要素の例の場合、最初のソートキーとなるLastNmae に加え、2番目のソートキーとなるFirstName を指定することができます。LastName の値が同じ Person 要素が与えられた場合、FirstName による並べ替えが行われます。これにより、SPS 内で、同じ並べ替え処理に対して複数の並べ替え条件を指定することが可能になります。

並べ替えのセットに対してテンプレートが適用されることで、並べ替えが行われた内容が出力に渡されます。並べ替えはHTML とRTF 出力にてサポートされます。

ユーザー定義テンプレート

ユーザー定義テンプレート を使用することで、指定したXPath 条件式により選択されるアイテムに対してテンプレートを適用することができます。XPath 条件式により選択されたノードを並べ替えることもでき、この場合、並べ替えはユーザー定義テンプレートに対して適用されます。

このセクション

- 並べ替えのメカニズム について記述されます。
- サンプル により、並べ替えがどのように使用されるか紹介します。

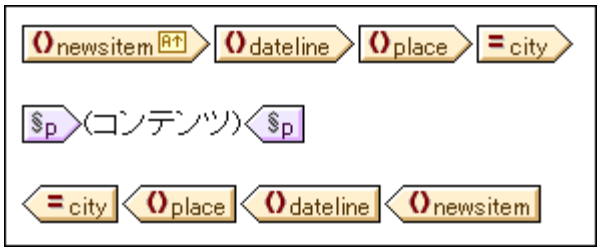
関連項目

- 動的テーブルの作成
- XPath ダイアログ
- ユーザー定義テンプレート

7.5.1 並べ替えのメカニズム

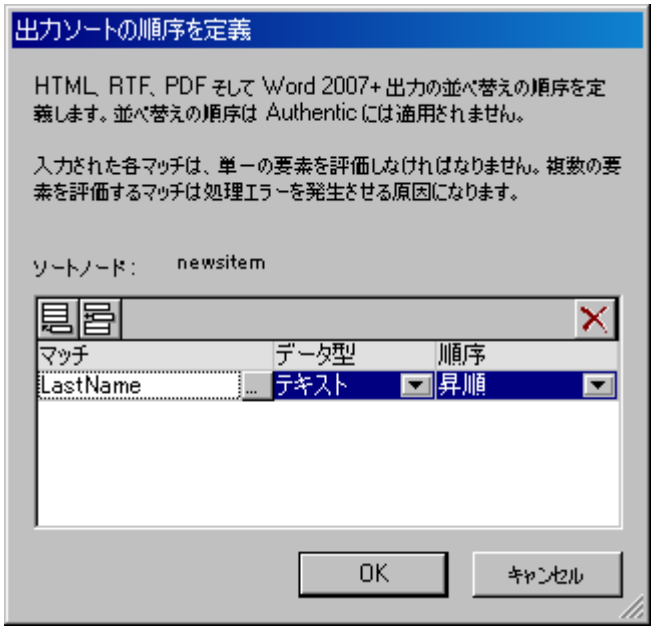
要素のノードに対して並べ替えを行うには、2つのステップを踏む必要があります:

1. デザインビューにて、並べ替えの対象となる要素ノードを選択します。XML ドキュメント内にある**この**要素のインスタンスが並べ替えられることになります。場合によってはどの要素が並べ替えの対象となるのか、すぐには分からないこともあります。例として以下のスクリーンショットに示される構造を考えてみましょう。



各 newsitem には、place 要素を含む dateline 要素が含まれており、place 要素には city 属性が含まれています。全 newsitem 要素以下にある @city ノードをアルファベット順に出力する場合を考えてみましょう。デザインでは、@city ノード（または place、dateline、newsitem 要素）を並べ替えの対象に選択するのでしょうか？@city が選択されると、1つの city ノードだけが並べ替えの対象となります。place または dateline が選択された場合でも、これらの要素は親ノード内に1度しか出現しないため、並べ替えは1つのノードに対してしか行われません。これに対して newsitem が選択された場合、親要素の newsitems 以下に複数の newsitem 要素が出現するため、newsitem に対して並べ替えを行うことになります。

2. 並べ替えを行う要素を選択した後には、その要素を右クリックすることで表示されるコンテキストメニューから**並べ替え**コマンドを選択します。出力ソートの順序を定義ダイアログが表示され、並べ替えの内容を入力することができます（以下のスクリーンショットを参照）。



各並べ替えの設定内容には、(i) ソートキー（マッチカラムにて入力）、(ii) 並べ替えに使用されるソートキーのデータ型（テキストまたは数値）、そして (iii) 並べ替えの順序（昇順または降順）が用意されています。このダイアログに表示される設定内容の順序は重要な意味を持ちます。複数のアイテムが同じ値を持っている場合、並べ替えは最初の行にある条件から最後の行まで順に行われます。並べ替えに使用する条件の数に制限はありません。

並べ替えが使用される例については、次を参照してください： サンプル: 複数のソートキーによる並べ替え.

ユーザー定義テンプレート

ユーザー定義テンプレート を使用することで、指定した XPath 条件式により選択されるアイテムに対してテンプレートを適用することができます。XPath 条件式により選択されたノードを並べ替えることもでき、この場合、並べ替えはユーザー定義テンプレートに対して適用されます。

ソートキーに関するメモ

入力されたXPath 条件式により選択されるソートキーは単一のノードでなければならず、ノードセット（XPath 1.0）やシーケンスアイテム（XPath 2.0 とXPath 3.0）とすることはできません。各要素のキーは、文字列または数値の値に対して解決可能である必要があります。

XSLT 2.0 または3.0 のSPS にてノードのシーケンスがソートキーに対して得られた場合、エラーがXSLT プロセッサーにより返されます。上記のPerson サンプルの場合、../Person/LastName というXPath 条件式では、Person の親要素に含まれている全てのLastName 要素が返されるため、エラーが返されます（複数のPerson 要素が存在する場合）。Person をコンテキストノードとする正しいXPath 条件式は、LastName となります（各 Person 要素には1つのLastName ノードしか存在しないため）。

XSLT 1.0 では、ソートキーのセレクターによりノードセットが得られた場合、最初のノードにあるテキスト値が使用されます。ソートキーに対してXPath 条件式により複数のノードが選択された場合でも、StyleVision によりエラーが返されることはなく、最初のノードにあるテキストが使用されます。しかし、最初に選択されたノードが目的のソートキーではない場合もあります。例えば、上に挙げた../Person/LastName XPath 条件式によりエラーが返されることはありませんが、各要素に対して同じ値（最初のLastName ノードにあるテキスト値）が使用されるため、並べ替えが行われることもありません。この場合、location/@* という形式の条件式を使用することにより、location 子要素にある最初の属性がソートキーとして使用され、並べ替えが行われることになりますが、このような条件式の使用は推奨されておらず、より正確な（単一のノードを選択する）条件式の記述が推奨されます。

関連項目

- サンプル: 複数のソートキーによる並べ替え
- ユーザー定義テンプレート
- 動的テーブルの作成
- XPath ダイアログ

7.5.2 サンプル: 複数のソートキーによる並べ替え

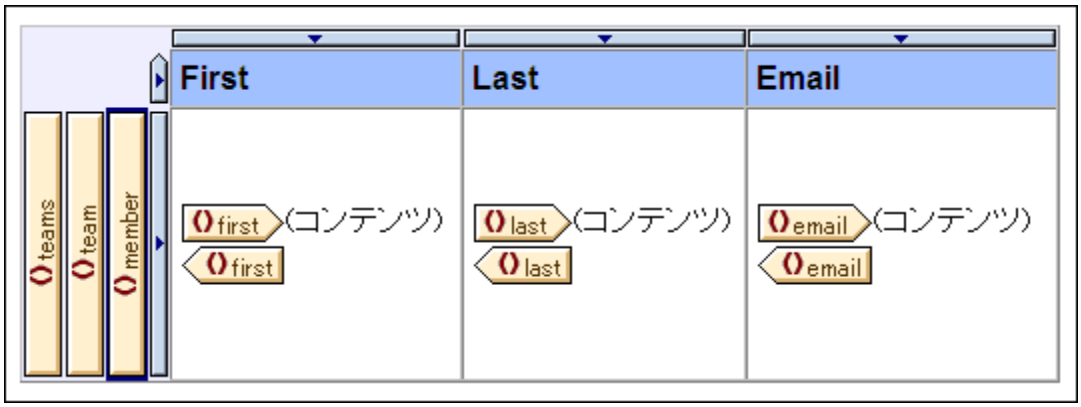
以下にあるシンプルな例では、チームメンバーがテーブルに表示されます（サンプルファイルは（マイ）ドキュメントフォルダーのC:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\Sorting\SortingOnTwoTextKeys.sps にて確認することができます）。テーブルの行には各メンバーの氏名と電子メールアドレスが表示されます。ここでは表示されているメンバーの苗字でまず並べ替えを行い、次に名前による並べ替えを行います。

表示されている名前が並べ替えられていない場合、XML ドキュメント内のmember 要素と同じ順序で名前の表示が行われます（以下にあるHTML 出力のスクリーンショットを参照）。

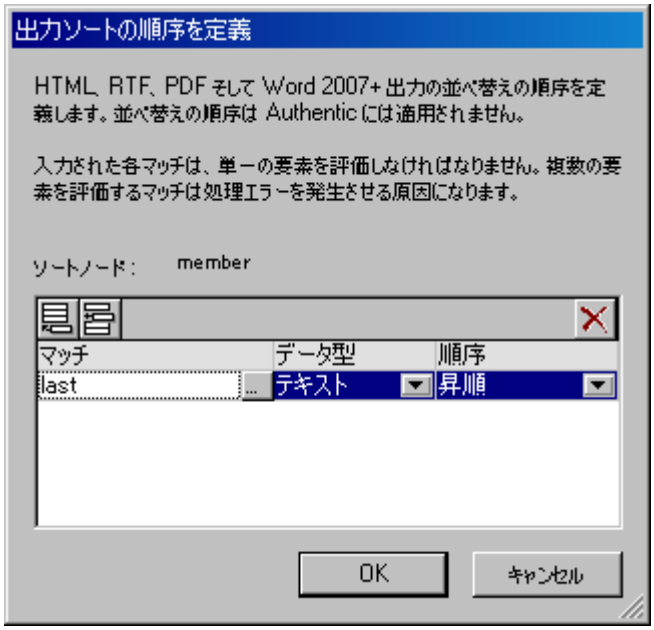
First	Last	Email
Andrew	Bentinck	a.bentinck@nanonull.com
Nadia	Edwards	n.edwards@nanonull.com
John	Edwards	j.edwards@nanonull.com
Janet	Ashe	j.ashe@nanonull.com

(C) 2015-2021 Altova GmbH

デザインビューにて member 要素（以下のスクリーンショットにて青色でハイライトされているノード）を右クリックし、コンテキストメニューから並べ替えコマンドを選択します。



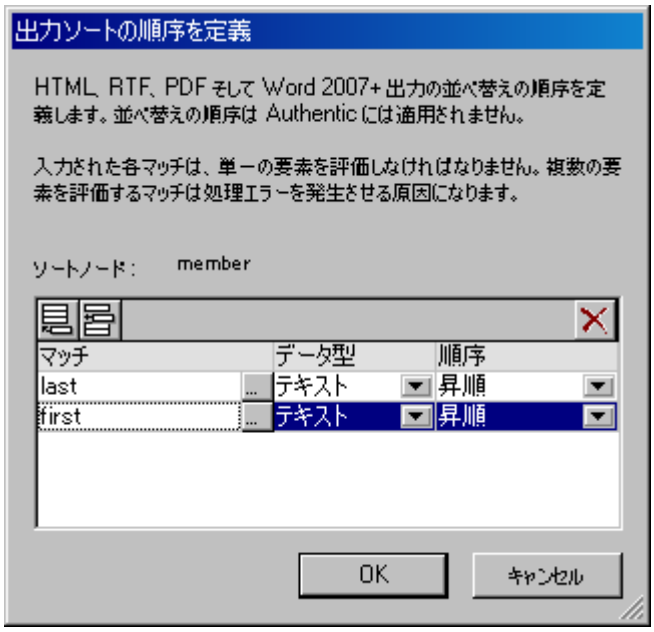
この操作により、出力ソートの順序を定義ダイアログが表示されます（以下のスクリーンショットを参照）。並べ替えに選択された要素の member がソートノードエントリーに表示されます。このノードがソートキーを選択する XPath 条件式のコンテキストノードにもなります。ペインツールバーの左側にある行の追加ボタンをクリックして、最初の並べ替え処理を追加します。追加された行にて XPath 条件式をマッチカラムに入力し、last ノードを選択します。構築ボタン ... をクリックすることでも、XPath 条件式を構築することができます。データ型カラムでは、ソートキーのコンテンツを評価する方法を、テキストと数値から選択することができます。順序カラムでは、並べ替えが行われる方向を、昇順と降順から選択することができます。テキストと昇順を選択し、「OK」をクリックしてください。



デザインビューでは、並べ替えのフィルターを表すアイコンが member タグに表示されます member。チームメンバーの HTML 出力では、以下に示されるように苗字が並べ替えられた状態で表示が行われます。ここで Edwards の並べ替えが正しく行われていないことに注目してください（Nadia が John の前に表示されており、これは XML ドキュメントにおける要素の順序に従ったものです）。名前による並べ替えを行うために、2番目のソートキーを作成します。

First	Last	Email
Janet	Ashe	j.ashe@nanonull.com
Andrew	Bentinck	a.bentinck@nanonull.com
Nadia	Edwards	n.edwards@nanonull.com
John	Edwards	j.edwards@nanonull.com

デザインビューにある member タグを右クリックして、コンテキストメニューから並べ替えコマンドを選択します。last による並べ替えを含む出力ノードの順序を定義ダイアログが表示されます。新たな並べ替えのルールを追加するには、新たな行を追加して、ソートキーに first 要素を入力します（以下のスクリーンショットを参照）。「OK」をクリックして完了します。



HTML 出力では、最初に苗字が、次に名前により並べ替えられた結果が表示されます。

First	Last	Email
Janet	Ashe	j.ashe@nanonull.com
Andrew	Bentinck	a.bentinck@nanonull.com
John	Edwards	j.edwards@nanonull.com
Nadia	Edwards	n.edwards@nanonull.com

(C) 2015-2021 Altova GmbH

関連項目

- 動的テーブルの作成
- XPath ダイアログ
- ユーザー定義テンプレート

7.6 パラメーターと変数

パラメーターと変数はSPS 内部にて宣言ならびに参照することができます。変数は宣言時に値が定義されるのに対して、パラメーターは(ランタイムにてコマンドラインから)渡された値により宣言時に割り振られたオプションのデフォルト値をオーバーライドすることができます。

このセクションでは、パラメーターと変数に対して使用することができる機能について説明します:

- ユーザー宣言パラメーター: ユーザー定義のパラメーターをSPS にて使用する方法について説明します。
- デザインフラグメントに対するパラメーター デザインフラグメントともにパラメーターを使用する方法について記述します。
- ソースに対するSPS パラメーター: スキーマソース(特に作業用 XML ファイルのスキーマ)に対してStyleVision が自動的に定義する特殊な種類のパラメーターについて説明します。これらパラメーターの名前と値はユーザーにとって既知のものであるため、SPS 内部でパラメーターの参照を行い、ランタイムにてコマンドラインから値を渡すことができます。
- 変数: (i) 特定のスコープに対して変数を宣言し、その値を定義し、(ii) 宣言された変数の値を参照し、変数により選択されたノードにてテンプレートを作成することができます。

関連項目

- SPS ファイルコンテンツ
- テンプレートとデザインフラグメント

7.6.1 ユーザー宣言パラメーター

SPS では、ユーザーにより宣言されたパラメーターがデフォルトの文字列値とともにグローバルに使用されます。一旦パラメーターが宣言されると、SPS 内の任意の場所からXPath 条件式により使用することができます。パラメーターのデフォルト値は、StyleVision Server からXSLT スタイルシートに対して新たなグローバル値を渡すことで、個々のXSLT 変換に対してオーバーライドすることができます。

パラメータの使用

ユーザー定義されたパラメーターは、以下の様な状況で使用することができます:

- 特定の値を複数の場所で使用し、複数の計算に対する入力値として使用する この場合、目的の値をパラメーター値として保存して、必要とされている場所や計算にてパラメーターを使用することができます。
- スタイルシートを処理する際に値を渡す: SPS (そしてスタイルシート)では、パラメーターがデフォルトの値とともに使用されます。処理が行われる際に、StyleVision Server から必要な値をパラメーターに対して渡すことができます。

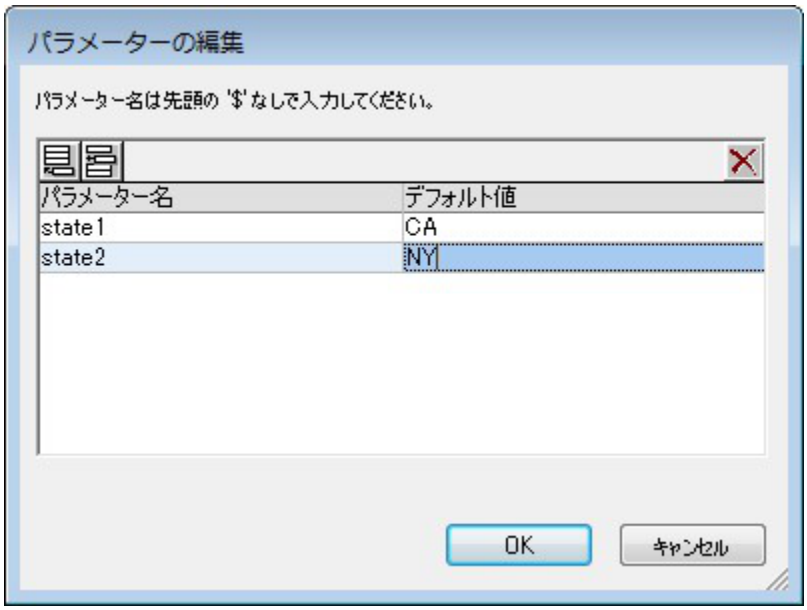
メカニズムの使用方法

ユーザーにより宣言されたパラメーターを SPS にて使用するには以下にある2つのステップを踏む必要があります:

1. 必要なパラメーターを宣言する。
2. 宣言されたパラメーターを参照する。

パラメータの宣言

ユーザーによるパラメーターの宣言と編集はパラメーターの編集ダイアログにて行うことができます(以下のスクリーンショットを参照)。パラメーターの編集ダイアログへは、メニューオプションの「編集 | スタイルシートのパラメーター」コマンド と(「編集 | DB フィルターの編集」により表示される)データベースフィルターの編集ダイアログにあるパラメーターボタンからアクセスすることができます。



パラメーターの名前とデフォルトの値によりパラメーターの宣言を行います。デフォルトの値を与えない場合、デフォルト値を空にします。

以下の操作によりパラメーターを宣言することができます:

1. パラメーターの編集ダイアログにある追加または挿入ボタンをクリックすることで、新たなパラメーターを追加または挿入します。新たな行が表示されます。
2. パラメーターの名前を入力します。パラメーター名の最初の文字にはアルファベットを、それ以外では英数字とアンダースコアを使用することができます。
3. パラメーターのデフォルト値を入力します。入力された文字はテキスト文字列として扱われます。

新規パラメーターの追加や、既存パラメーターの編集は、SPS または Authentic View のどちらかの編集時にいつでも行うことができます。

メモ

- パラメーターの編集ダイアログには、SPS にある全てのユーザー定義パラメーターが表示されます。
- パラメーターは デザイン概要サイドバーでも宣言することができます。

宣言されたパラメーターの参照

XPath 条件式では、パラメーターの前に $ というプレフィックスを配置することで、パラメーターへの参照を行うことができます。例えば、自動計算の XPath 条件式にてパラメーターを参照することができます(例: concat('www.', $company, '.com'))。作成している SPS が DB をベースにしたものである場合、パラメーターを DB フィルターの条件として使用することができます。DB のパラメーターはパラメータの編集 ダイアログにて、宣言ならびに編集されます。

宣言されていないパラメーターを参照するとエラーが返されますが、参照されないパラメーターを宣言してもエラーとはなりません。

7.6.2 デザインフラグメントに対するパラメーター

デザインフラグメントのパラメーターにより、作成したデザインフラグメントに対してパラメーターを定義し、このパラメーターに対してデフォルト値を与えることができます。デザインフラグメントが使用されるデザインの各箇所では異なるパラメーター値を入力することも可能で、各デザインフラグメントの出力を個別に修正することもできます。

例えばEMailAddresses という名前のデザインフラグメントをaltova.com というデフォルト値を持つDomain という名前のパラメーターとともに作成する例を考えてみましょう。このパラメーターを自動計算にて使用することで、従業員の電子メールアドレスを生成することができます。EUのアドレスに対してはEmailAddresses デザインフラグメントにあるDomain パラメーターの値にaltova.eu を使用します。同様の方法で、日本のアドレスに対しては、altova.jp をDomain パラメーターの値として使用します。米国内の従業員が持っているアドレスに対してはDomainの値を変更すること無く、デフォルト値のaltova.com を使用します。

デザインフラグメントに対してパラメーター値を使用するには、以下にある2つのステップを踏むことになります:

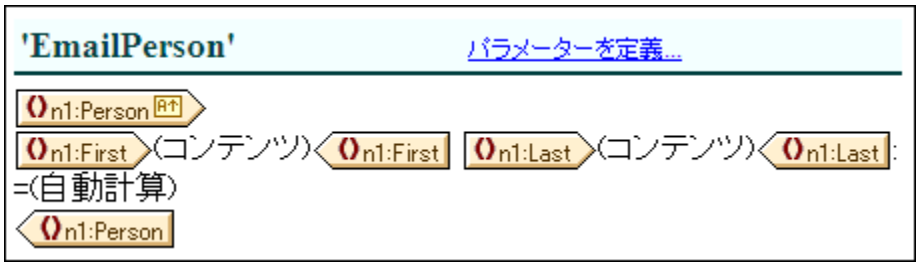
1. デザインフラグメントが作成される場所にてデフォルトの値とともにパラメーターを定義します。
2. デザインフラグメントが使用される場所でパラメーターの値を編集します。

これらのステップについては以下で詳細に説明します。

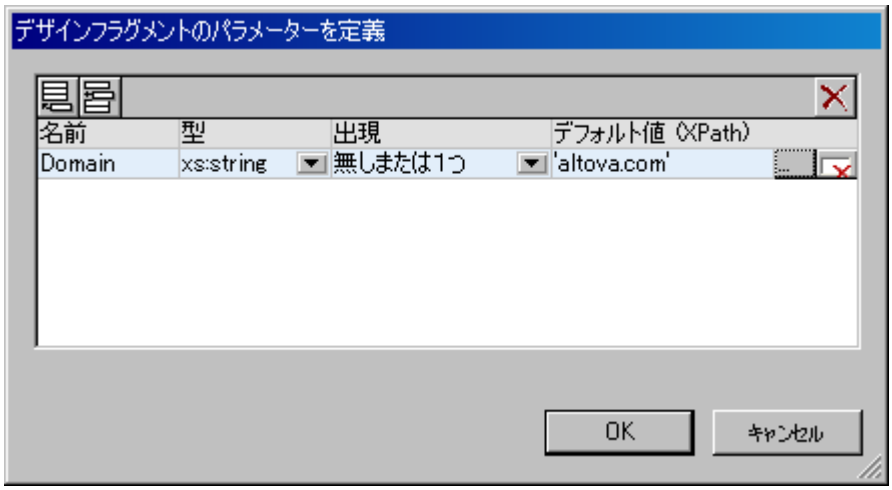
メモ デザインフラグメントに対するパラメーター機能は Enterprise Edision の Altova 製品における Authentic View でのみサポートされます。

パラメーターの定義

各デザインフラグメントに対して割り当てるパラメーターの数に制限はありません。デザインフラグメントのタイトルバーにあるパラメーターを定義リンクをクリックすることでパラメーターを定義できます(以下のスクリーンショットを参照)。



この操作により、デザインフラグメントのパラメーターを定義ダイアログが表示されます(以下のスクリーンショットを参照)。左上にある**追加**ならびに**挿入**アイコンをクリックすることで、パラメーターを入力する行が追加されます。パラメーターの名前、データ型、出現の回数、そしてデフォルト値を入力または選択します。パラメーターの出現属性により、パラメーターのデフォルト値として指定されたXPath 条件式を評価することで得られるアイテムの数を指定することができます。出現属性はオプションとなっており、デフォルトで「無しまたは1つ」にセットされています。パラメーターの数に制限はありません。

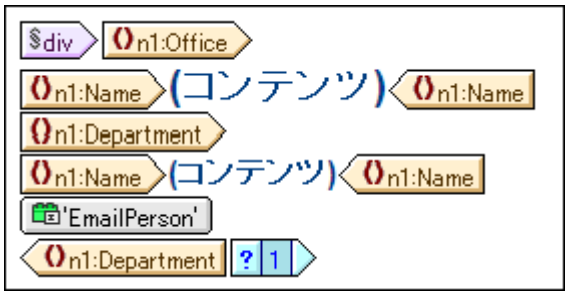


ダイアログには2種類の削除アイコンがあります。各パラメーターエントリーの右側にある削除アイコンをクリックすると、パラメーターのデフォルト値が削除されます。ペイン右上にある削除アイコンをクリックすることで、現在ハイライトされているパラメーターが削除されます。

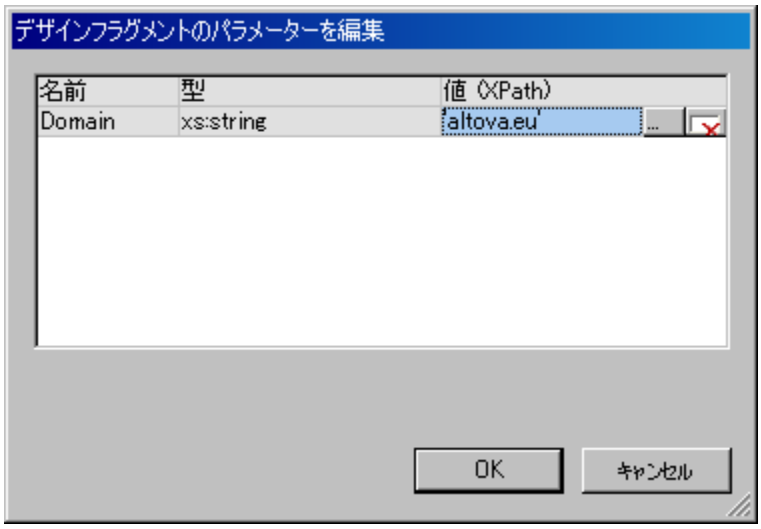
メモ SPS にて XSLT 1.0 が選択されている場合、入力した XPath 条件式によりノードセットが返されなければなりません（そうでない場合エラーが返されます）。

パラメータを使用する

デザインフラグメントを作成すると、デザイン内にある複数の箇所へ（デザインツリーやスキーマツリーからドラッグすることにより）挿入することができます。デザインフラグメントのEmailPerson が n1:Name 要素の後に挿入された状態を以下のスクリーンショットに示します。



デザインフラグメントにてパラメーターが定義すると、デザインフラグメントのインスタンスにてパラメーターの値を編集することができます。デザインフラグメントのインスタンスを右クリックして、**パラメーターを編集**コマンドを選択します。デザインフラグメントのパラメーターを編集ダイアログが表示されます（以下のスクリーンショットを参照）。



このダイアログではパラメーターの値を編集することができます。「OK」をクリックすることで操作を完了します。入力されたパラメーター値がデザインフラグメントのインスタンスにて使用されます。パラメーターの値が編集されていない場合、オリジナルの（デフォルトの）パラメーター値が使用されます。

メモ XSLT 1.0 が選択されている場合、入力した XPath 条件式によりノードセットが返されなければなりません（そうでない場合エラーが返されます）。

関連項目

- ユーザー宣言パラメーター：ドキュメント全体に対して妥当なスタイルシートパラメーターの使用方法について記述

されます。.

7.6.3 ソースのための SPS パラメータ

SPS にはXMLドキュメントがベースとするDTD やXML スキーマ、またはDB から生成されたXML スキーマやDB がベースとしているスキーマを、スキーマソースとして複数個含めることができます。

SPS には、メインとなるスキーマを1つ、そしてオプションのスキーマを複数個含めることができます。新たなスキーマソースが加えられると、そのスキーマに対するパラメーターがStyleVision により自動的に宣言され、そのスキーマに対して割り当てられた作業用 XML ファイルのURI が、パラメーターの値として自動的に割り当てられます。DB の場合、StyleVision により一時的な XML ファイルがDB から生成され、このXML ファイルのドキュメントノードがパラメーターにセットされます。

ソースに対するパラメーターを参照する

各 SPS パラメーターは、そのスキーマに対応するXML ファイルのドキュメントノードを値に持ちます。StyleVision では、各スキーマに対応したXML ファイルは、作業用 XML ファイルまたはDB から生成されたXML ファイルとなります。従って、ソースに対するSPS パラメーターを以下のように利用することができます:

1. SPS 内にある XPath 条件式にて様々なドキュメント内のノード位置を特定する: ドキュメントの位置特定にパラメーターが使用され、ドキュメント内にある目的のノードの位置を特定するために、XPath 条件式が使用されます。例えば、count($XML2//Department/Employee) という条件式により、$XML2 として指定されたスキーマソースの作業用 XML ファイル内にある Department 要素に含まれている Employee 要素の数が返されます。
2. コマンドラインから、他の XML ファイルの URI を SPS のパラメーターとして渡す: 勿論、新たな XML ファイルもパラメーターにより指定されたスキーマをベースにしたものでなければなりません。例えば、FileA.xml と FileB.xml が、SPS にて使用されているスキーマの $XML3 に対して妥当であり、FileA.xml が SPS にて使用される $XML3 スキーマに対して割り当てられた作業用 XML ファイルである場合、コマンドラインからその SPS に対して XSLT 変換を指定する際に、$XML3="FileB.xml" というパラメーターを使用することで FileA.xml の代わりに FileB.xml を使用することができます。コマンドラインでは、メインスキーマを除く全てのソースに対して全ての SPS パラメーターを入力する必要がある点に注意してください。メインスキーマに対応する XML ファイルは XSLT スタイルシートのエントリーポイントであり、変換が行われる XML ファイルとなります。

関連項目

- ユーザー宣言パラメーター

7.6.4 変数

変数は、(i) 変数の宣言 と(ii) 変数の使用により構成されます。

メモ 変数は、Enterprise Edition の Altova 製品にある Authentic View でしかサポートされない点に注意してください。.

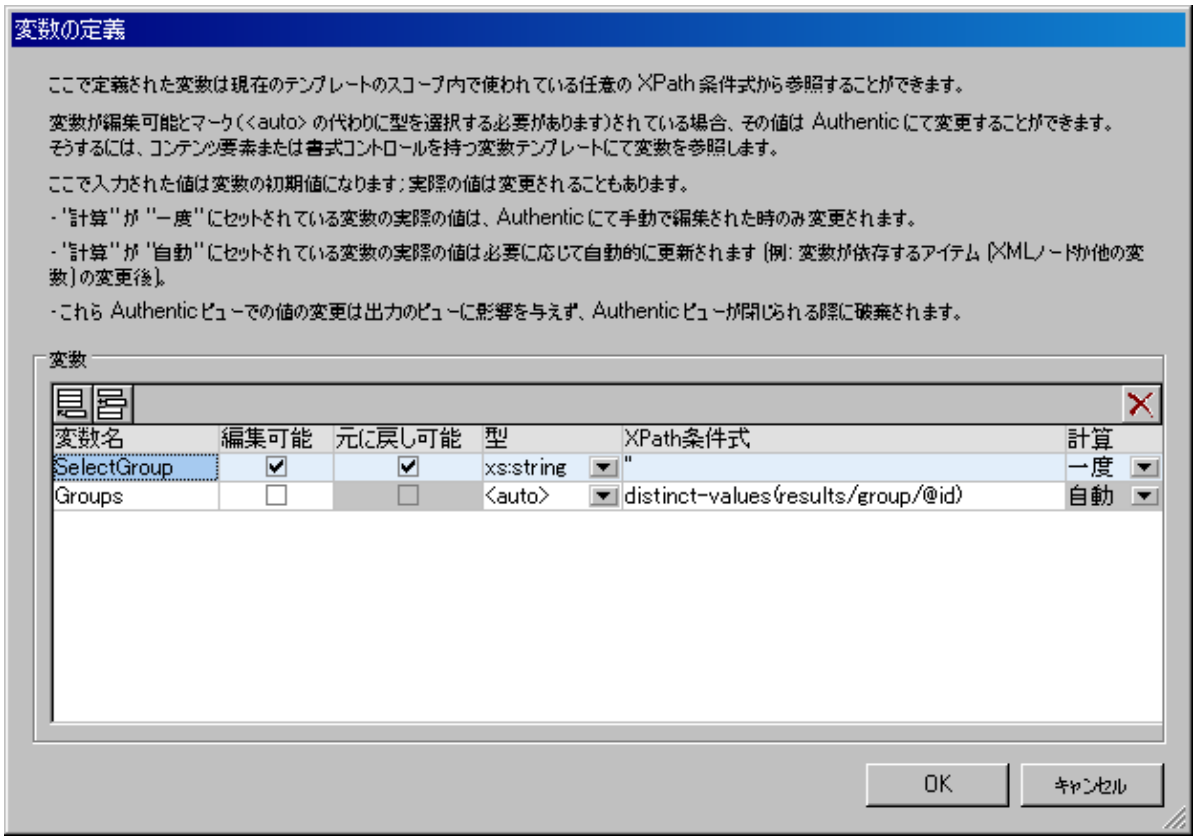
(C) 2015-2021 Altova GmbH

変数の宣言

変数は、デザイン内に含まれているあらゆるテンプレートにて宣言することができます。変数には名前、データ型、そして値が与えられます。更に、Enterprise エディションのAuthentic View にて編集可能かを指定することができます。変数はこのテンプレートをスコープに作成され、その内部で使用することができるようになります。ドキュメント全体をスコープとする変数を宣言するには、ルートテンプレートに対して変数を宣言します。テンプレートにて変数を宣言することで、子孫ノードを特定するためのXPath 条件式をシンプルに作成することができます。

以下の操作により変数を宣言することができます:

1. 変数を作成するノードテンプレートを右クリックして、コンテキストメニューから**変数を定義**を選択します。
2. 変数の定義ダイアログにて、変数ペインの左上にある**変数の追加**アイコンをクリックして、変数の名前を入力します（以下のスクリーンショットを参照）。変数の値は XPath 条件式を介して入力することになります。変数の値に（下のスクリーンショットにある最初の変数のように）文字列を使用したい場合、引用句により文字列を囲んでください。以下のスクリーンショットでは、SelectGroup 変数の値が空の文字列となり、他にもノード名や関数呼び出しを使用することができます。



3. **編集可能**チェックボックスにチェックを入れて変数を編集可能にセットすることで、Authentic View にて変数を編集することが可能になります。。この場合、データ型の値を xs:string のように正しくセットする必要があります。変数が編集可能となっている場合、SPS デザイナーによりセットされたオリジナルの値を Authentic View ユーザーが Authentic View にて変更することができます。変更は（変数の値を編集可能なコンテンツや Authentic View により編集可能なテキストボックスなどにより）変数を直接編集することでも行えますし、変数の XPath 条件式で使用されているノードや値を Authentic View ユーザーが修正することでも行うことができます。
4. 変数が編集可能となっている場合、Authentic View に関連して2つのオプション（**元に戻し可能**ならびに**計算**）が更に有効になります。元に戻しオプションのチェックボックスにチェックを入れることで、変数に対して行われた全ての変更を元に戻すことが可能になります。このオプションにより、Authentic View ユーザーは変数の値を変更前のものに戻すことができるようになります。**計算**オプションの値は一度または自動で、オプションの値が一度にセットされた場合、変数を含むテンプレートが評価される時に、変数の値が一度だけ計算され、ユーザーに

より変数が明示的に変更された時だけ値の変更が行われます。オプションの値が**自動**になっている場合、変数の XPath 条件式で使用されているノードや値が修正された時にも変数の計算が行われます。
5. 追加することのできる変数の数に制限はありませんが、同じスコープ内で既に宣言されている変数の名前を使用することはできません。変数を削除するには、ペインの右上にある**削除**アイコンをクリックしてください。
6. 「**OK**」をクリックすることで完了します。テンプレートタグに $ アイコンが表示され、そのテンプレートに対して変数が宣言されたことを示します。

上の操作では、デザイン内にある各ノードテンプレートに対して変数が作成されます。これら各変数には名前と値が与えられ、宣言されたテンプレート内部がそのスコープとなります。変数を修正するには、変数が作成されたノードテンプレートを右クリックして、コンテキストメニューから**変数を定義**コマンドを選択し、変数の定義ダイアログへアクセスします。

変数の使用

変数は、その変数が作成されたスコープ内で使用することができます。つまり、変数が作成されたテンプレート内部でしかその変数を使用することはできません。変数は Authentic View でのみ編集することができるため、表示内容はユーザーによりコントロールされます。編集された値は SPS が閉じられた時に破棄されます。

変数は任意の XPath 条件式で使用することができ、$ シンボルとともにその名前を使用することで、XPath 条件式内部から参照することができます。例えば、$VarName/Name という XPath 条件式を使うと、VarName という名前の変数により選択されるノードの Name 子要素が選択されます。

XPath 式の編集ダイアログにて XPath 条件式を入力すると、スコープ内にある変数がポップアップにて表示されます（上のスクリーンショットを参照）。ポップアップから変数を選択し、Enter キーを押下することで、その変数への参照が条件式内に追加されます。

関連項目

- SPS ファイルコンテンツ
- テンプレートとデザインフラグメント
- ノードテンプレートオペレーション

7.6.5 Authentic にて編集可能な変数

スコープ内にある変数を Authentic View にて編集することで、Authentic View における表示を変更させることができます。例えば、社内の各支店における従業員のデータを含んだ非常に巨大な XML ドキュメントを使用する場合を考えてみましょう。特定の支店を Authentic View にて選択することで、その支店データだけを Authentic View に表示することができるようになります。

動作方法

Authentic View にて編集可能な変数をセットアップするには、3 つのステップを踏む必要があります:

1. 変数を使用するテンプレート内部で目的の変数を作成します。このテンプレートにより編集のスコープが定義され、変数はスコープとなるテンプレート内部でのみ使用することができるようになります。
2. ユーザー定義されたテンプレートを変数の名前とともに作成します。ユーザー定義されたテンプレートの動的コンテンツには変数の値が含まれます。コンテンツプレースホルダや入力フィールドがユーザー定義テンプレートのコンテンツとしてデザイン内に挿入されると、Authentic View ユーザーは任意のコンテンツを変数の値として入力できるようになります。Authentic View が選択することのできるオプションは、コンボボックスのようなフォームコントロールを挿入することにより制限されます。
3. XPath 条件式にて変数を使用することで、Authentic View の表示を制御することができます。例えば、条件にて変数を使用することで、各分岐にある値により異なる内容を表示することができます。他にも、テンプレートマッチ条件式やテンプレートのフィルターにて変数を使用することができます。

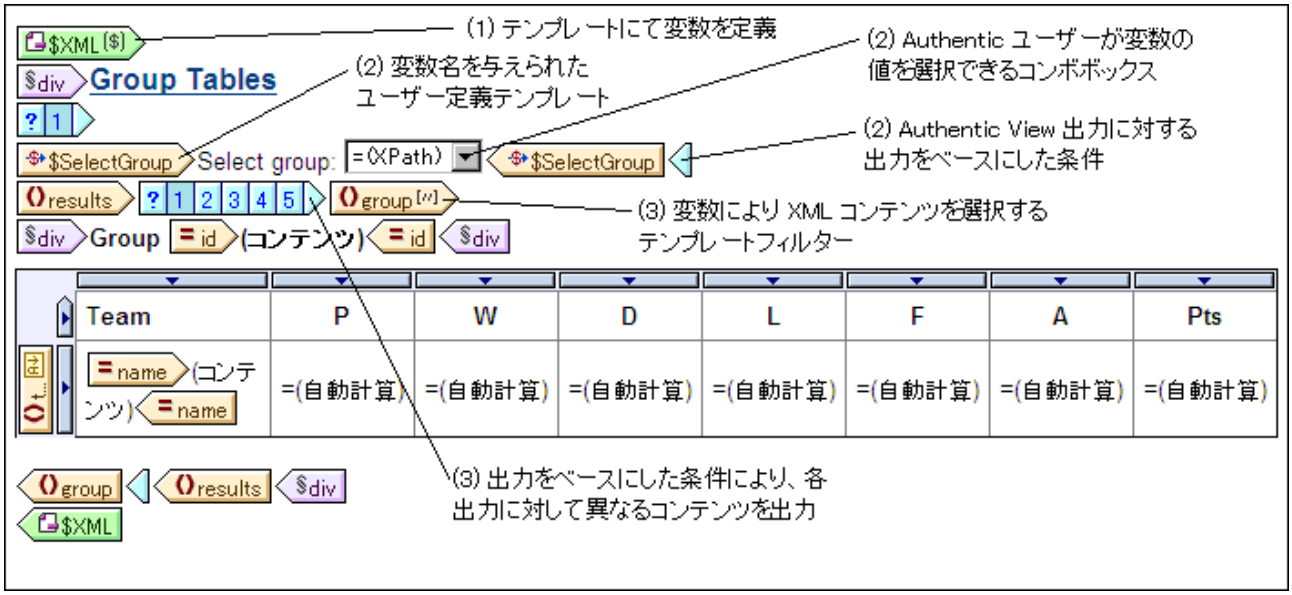
メモ　以下の点に注意してください：

- グループ化や並び替えは Authentic View にてサポートされていないため、これらの機能を含む SPS を Authentic View で表示しても、編集可能な変数を使用することはできません。
- 編集可能な変数機能は Authentic View でのみ使用することができるため、Authentic View のデザインは他の出力とは別に作成する必要があります。このようなデザインは 出力ベースの条件 機能により簡単に作成することができます。

サンプルファイル

(マイ) ドキュメントフォルダー以下の `C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\Grouping\Scores\` にある AuthenticVariables.sps ファイルでは、Authentic View にて編集可能な変数の使用方法を確認することができます。XML ファイルにはトーナメント戦の試合結果が含まれています。トーナメントに参加したチームは、2つのグループに分けられています。編集可能な変数により、表示されるグループが Authentic View にて選択されるようになります。

以下のスクリーンショットでは、SPS デザイン全体が示されます。編集可能な変数が1つ作成され、そのための手順が以下のスクリーンショットでは示されます。スクリーンショット以下ではステップの詳細が示されます。



編集可能な変数をセットアップする際に重要となるステップを以下に示します:

- (ルートテンプレートの) $XML テンプレートにて、SelectGroup という名前の編集可能な変数を xs:string にて定義します。テンプレート全体がこの変数のスコープとなります。

- $XML では、Groups という名前で 編集不可能な変数 が<auto> という型で定義されています。全ての results/group/@id 属性値から、一意の値を動的に取得するためにこの変数は使用されます。これら一意の値は、コンボボックス のドロップダウンリストに表示され、Authentic View ユーザーが表示するグループを選択するのに使用されます。
- ユーザー定義テンプレート が$SelectGroup という名前（編集可能な変数の名前）で作成されます。ユーザー定義テンプレートの場所は、それが編集可能な変数のスコープ内である限り問題となりません。
- ユーザー定義テンプレート内部にはコンボボックスが挿入されます。コンボボックスにある XPath 条件式の$Groups, 'All' によりドロップダウンリストのエントリー値が選択されます。この XPath 条件式により、$Groups 変数内に含まれているアイテムのシーケンスが返され、All というアイテムが、$Groups から返されたシーケンスに追加されます。コンボボックスの All エントリーアイテムは、全てのグループを表示するために使用されます。
- ユーザー定義テンプレートは出力をベースにした条件テンプレート 内に収められており、条件となる出力にはAuthentic View がセットされています。これは編集可能な変数をAuthentic View でのみ使用するためです。.
- 次のステップでは Authentic View ユーザーが選択する編集可能な変数の値を使用します。変数を使用することで、Authentic View ユーザーが選択したグループだけが表示されるようになります。表示は results/group テンプレートに対して フィルターの指定 により行われます。フィルターの XPath 条件式を以下に示します:

```
if ($SelectGroup != 'All') then @id=$SelectGroup else @id
```

  このフィルター条件式により、グループ要素の述語ステップがセットアップされます。$SelectGroup 変数に All 以外の値が含まれている場合、述語ステップは [@id=X] となります（X はコンボボックスにて Authentic View ユーザーが選択した $SelectGroup 変数の値）。このフィルターにより、ユーザーが選択した値を id 属性に含んでいるグループが選択されることになります。$SelectGroup 変数の値が All である場合、任意の id 属性を持ったグループ、つまり全てのグループが選択されます。
- group テンプレートは 出力をベースにした条件 内に収められており、各分岐がそれぞれの出力に対応します。Authentic View の分岐でのみ、group テンプレートにフィルターが適用されます。

Authentic View の出力は以下のようになります:

コンボボックス内のエントリーに注目してください。コンボボックスにて B グループが選択されると、Group B だけが表示されます。

関連項目

- 変数
- ユーザー定義テンプレート
- ノードテンプレートオペレーション
- 条件

(C) 2015-2021 Altova GmbH

7.7 目次、参照、ブックマーク

目次（TOC）やその他の参照メカニズムは、デザインドキュメント内にある目的の場所にてアンカーを作成し、目次やテキスト参照、自動番号シーケンス、またはハイパーリンクから、これらのアンカーに対して参照を行うというかたちで作成されます。

まずはアンカー（またはブックマーク）のメカニズムについて簡単に説明し、目次メカニズム全般について説明します。ブックマークの動作方法を理解することにより、目次メカニズムの理解をより良く行うことができるようになります。

ブックマークのメカニズム

ブックマークには、単純ブックマークと複合ブックマークという2種類のメカニズムが存在します。目次の作成には複合ブックマークが使用されます。

- 単純ブックマークはデザインドキュメント上に作成されます。ブックマークにはユニークな名前が与えられ、参照を行うリンクのターゲットとして使用されます。単純ブックマークメカニズムは、ブックマークとハイパーリンクに対して使用される機能となります。ハイパーリンクはドキュメント外部にあるURL への参照を行うこともができるということに留意してください。
- 目次やドキュメントセクションに対して番号を与えるといった、より複雑な参照を行うには、以下にある2つのステップによりブックマークを作成します:

1. 目次に適合するかたちでデザインドキュメントを階層構造にデザインします。階層構造における各レベルは目次レベルと呼ばれます。ドキュメント構造内部にある各ポイントへ目次レベルを割り当てることにより、構造を作成することができます。目次レベルは他の目次レベル内部にネストすることができ、ドキュメントにて階層的な目次構造を作成することができます。例えば、ある目次レベルを本の章に割り当て、別の目次レベルを章の中にある節に対して割り当てることができます。
2. 目次レベル内部に目次ブックマークを作成することができます。これら目次ブックマークにより、目次内部へ渡される各レベルにおけるドキュメントセクションが識別されます。更に、各目次ブックマークには参照元のコンポーネントに表示されるテキストを指定する必要もあります。

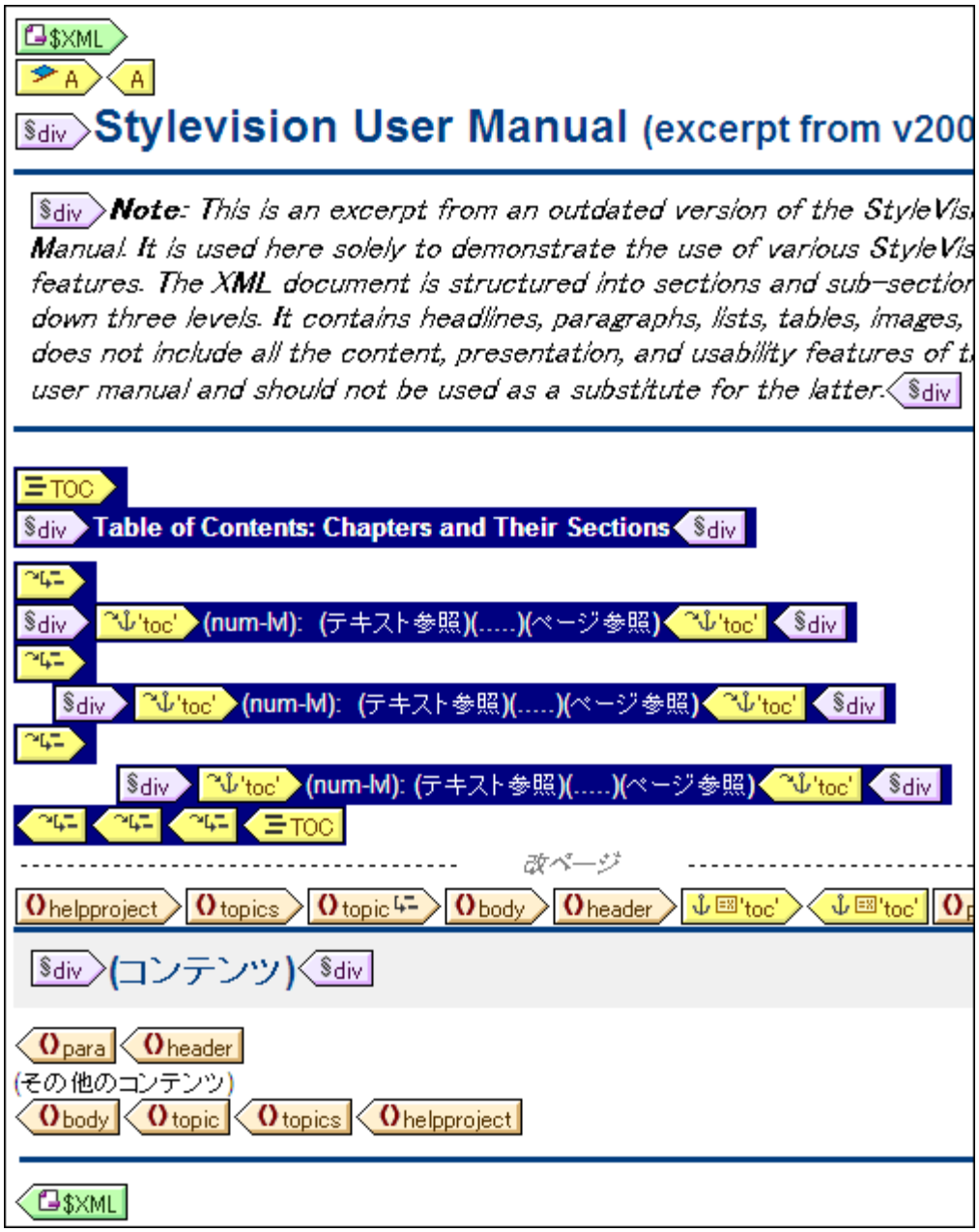
目次レベルならびに目次ブックマークの参照テキストが定義されると、参照元のコンポーネントを含む目次テンプレートをデザインすることができます。

以下にある目次メカニズムでは、目次のメカニズム全般について簡単に説明されます。このセクションでは、利用することのできる参照機能の詳細についても記述されます。

目次メカニズム

SPS で使用されるXSLT のバージョンに（XSLT 1.0 ではなく）XSLT 2.0 またはXSLT 3.0 を選択した場合、目次（TOC）－正確には目次のテンプレート－をデザインにおける任意の場所で作成することができます。

- 目次内部からリンクされるデザイン上のアイテムは、最初にデザイン内部でブックマークしておくことが推奨されます。これらのアイテムは静的コンテンツまたは動的コンテンツとなります。以下にあるスクリーンショットの下半分には、黄色の目次ブックマークタグ目次ブックマークtags 'MyTOC' 'MyTOC' が header タグ内部に表示されており、header アイテムが（目次テンプレートに含めるために）ブックマークされています。
- 目次のためにテンプレートを作成します（以下のスクリーンショットにてハイライトされている箇所を参照）。目次テンプレートには目次のデザインが含まれており、テンプレートはデザインにおける任意の場所に配置することができます。以下のスクリーンショットにて示される例では、ドキュメントの上部近くに目次テンプレートが配置されています。



メモ これら2つのパーツを作成する順序に決まりはなく、同時に作成を行うこともできます。しかし、目次テンプレートを作成する前に目次ブックマークを作成することが推奨されます。

目次は Authentic View や HTML とRTF 出力にて表示されます。他にも、(i) 目次は階層構造またはフラット構造で作成することができ、(ii) デザイン内部に複数の目次を作成することができます。これにより、階層構造によるブックならびに章レベルの目次に加え、図表のリストをフラットな構造で作成することができるようになります。

目次の作成ステップ

目次の作成方法を、以下のステップに示します。最初に目次アイテムを追加して、その後に目次テンプレートを作成します。その他にも目次テンプレートをまず作成して、追加するブックマークアイテムを作成することができます。また、目次テンプレートとアイテムの作成を並行して行うこともできます。

1. XSLT のバージョンに XSLT 2.0 が選択されていることを確認してください。
2. *目次レベル内部のドキュメントの構造*: 目次に複数のレベルが含まれる場合、ネストされた目次レベルの構造をデザインします。目次がフラットな構造になる場合（つまり目次レベルが1つだけの場合）、目次ブックマークを含めることになる目次レベルを少なくとも1つ作成します。
3. *ドキュメントデザインにおける各レベルにて1つ以上の目次ブックマークを作成*: 目次内部に表示される各目次レベル内にあるコンポーネントが、目次ブックマークにより識別されます。
4. *目次レベルの参照を含んでいる目次テンプレートを作成*: 目次テンプレートには、必要な数だけの目次レベル参照を含める必要があります。目次が複数レベルにより構成される場合、目次テンプレート内部にあるレベル参照をネスト化する必要があります。
5. *目次テンプレート内部に目次参照（TOCref）を作成*: 目次テンプレートにて、各レベル参照に対して目次参照をセットします。各目次参照から、ドキュメントの対応する目次レベル内にある目次ブックマークへの参照が行われます。目次参照からは、その他のレベルにある目次ブックマークを参照することもできます。
6. *目次アイテムのフォーマット*: 目次出力における各テキストアイテムは、目次テンプレートの目次参照により生成されます。目次参照の定義により、（階層構造を含む）アイテムの番号付け、目次アイテムのテキスト、そして印刷媒体に対してはページ番号を指定することができます。ドキュメントのメインボディーにある目次ブックマークでは、自動的に割り当てられる番号を定義することもできます。詳細については番号の自動割り当てのセクションを参照ください。

用語解説

インターフェースで使用される目次に関連した主要なコンポーネントの名前を以下のテーブルに示します。テーブルのカラムにて示されるように、**ドキュメントボディー**と（通常ドキュメント上部に実際に表示される目次のデザインである）**目次テンプレート**は個別に理解する必要があります。

- **ドキュメントボディー内部の目次コンポーネント**: 目次テンプレートにて使用されるアイテムをマークするために使用されます。
- **目次テンプレート内部の目次コンポーネント**: ドキュメントボディー内部にてマークされたアイテムを参照するために使用されます。

ドキュメント body	目次 テンプレート
目次レベル: ネスト形式で、ドキュメントの構造を作成します。	**レベル参照（levelrefs）**: ドキュメントボディーにて定義された目次レベルの構造に対応します。対応するレベルにある目次ブックマークへの目次参照を行うことができます。
目次ブックマーク: 名前を与えることにより、目次アイテムとして使用されるドキュメント内のノードを識別します。	**目次参照（TOCrefs）**: 名前により目次ブックマークへの参照を行います。

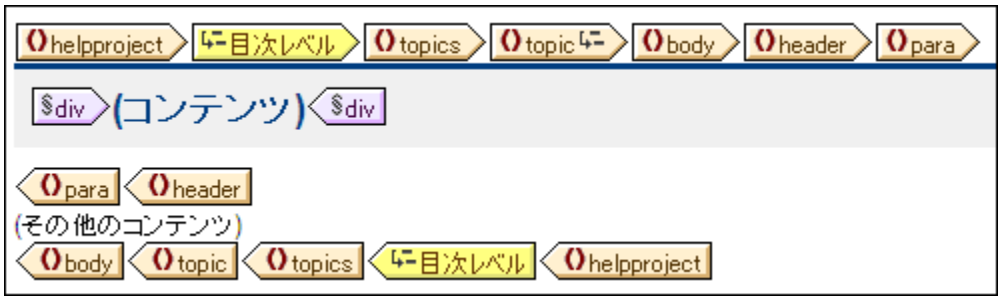
関連項目

- 目次に含まれるブックマークアイテム
- 目次テンプレートの作成
- ブックマークとハイパーリンク

7.7.1 目次に含まれるブックマークアイテム

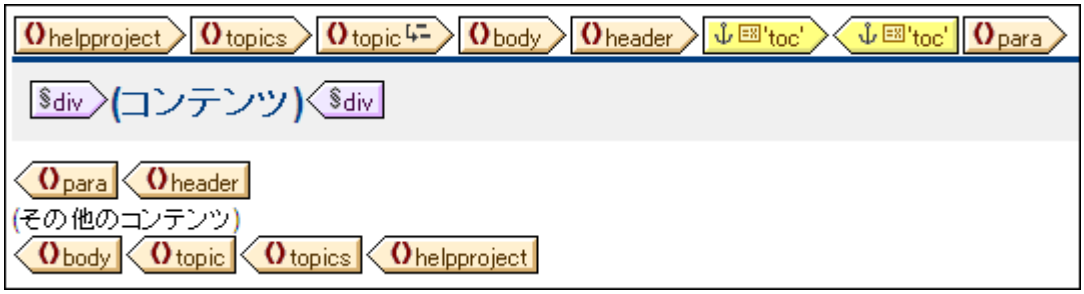
デザイン内にあるブックマークを目次に含めるには、2つのステップを踏む必要があります（ステップの順序に意味はありません）。

1. デザインドキュメントを目次レベルにてネストされた構造になるように構成する： デザインにおける目次レベルは、テンプレート上に、またはデザインコンポーネントの周りに作成することができます。以下のスクリーンショットでは、topic テンプレート にて目次レベルが作成されています。



テンプレートにてレベルが作成されると、テンプレートの開始タグにその旨が表示されます（例：topic）。コンポーネントの周りにレベルが作成された場合、目次レベルタグ 目次レベル 目次レベル により目次レベルが表示されます。上のスクリーンショットでは、topics テンプレートコンポーネントがレベルにより囲まれることになります。これら2つの方法の違いは、レベル内部のデザイン構造 セクションにて説明されます。目次テンプレートの作成された場合、デザインにて作成されたレベルに対応するかたちで目次テンプレートを作成する必要があります。フラットな構造（レベル1つ）の目次においても、デザインには対応するレベルを含める必要があります。

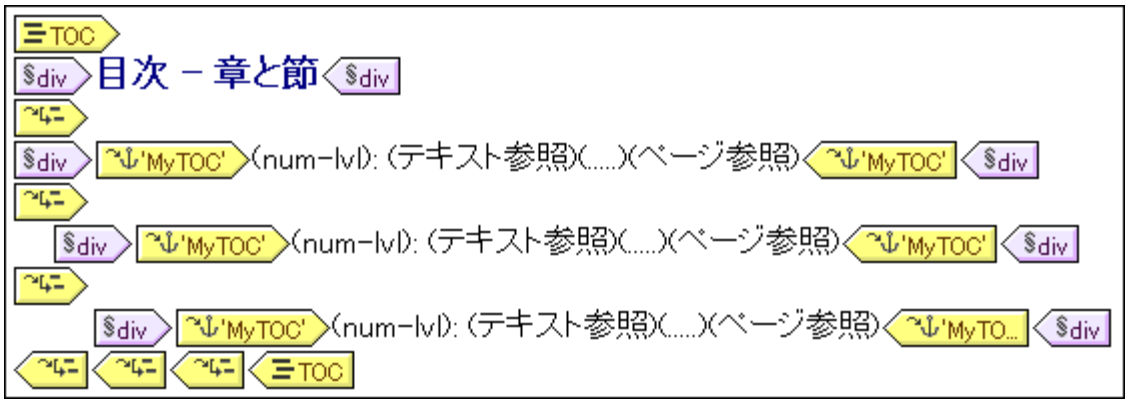
2. 目次ブックマークの作成 ： 名前ならびに目次アイテムテキストを伴うデザインにて目次ブックマークを作成します。目次ブックマークはデザインコンポーネントを囲むかたちでデザインすることもでき、コンポーネントが囲まれない場合、空の要素として表示されます。以下のスクリーンショットでは、デザインコンポーネントを囲んでいない目次ブックマークが示されています。



目次ブックマークはドキュメント内のアンカーとして動作します。上のスクリーンショットでは目次ブックマーク（アンカー）が para 要素インスタンスの開始タグに配置されています。目次ブックマークには、(i) 目次テンプレート内にある目次アイテムを作成する際に目次ブックマークを参照するための名前と、(ii) 対応する目次アイテムのテキストとして使用されるテキスト文字列という、2つの属性が含まれています。これら2つの属性の割り当て方法については、目次ブックマークの作成セクションを参照ください。

目次テンプレートにてブックマークされたアイテムを参照する

（デザインテンプレートのメインボディーに作成されたレベルとは違う）レベル参照という名前のネストされたレベル内にTOC テンプレート は作成されます。各レベル参照 内部には、目次参照 'MyTOC' が挿入されます（以下のスクリーンショットを参照）。レベル参照内にある目次参照からは、（目次ブックマークの名前による）目次ブックマークへの参照が行われます。XML ドキュメント内部の対応するレベルにある指定された名前を持った目次ブックマークが、目次内のアイテムとして作成されます。例えば 'chapters' というタグにより表示される目次参照により、XML ドキュメント内の対応するレベルにある chapters という名前の全目次ブックマークが参照されることになります（目次参照のスコープが current にセットされていた場合）。インスタンス化された目次ブックマークのテキスト属性が目次アイテムのテキストとして使用されます。



上のスクリーンショットに示される目次テンプレートでは、3つのレベル参照がネストされており、各レベル内にある目次参照にはそのレベルの目次アイテムに対するテンプレートが含まれています。例えば、最初のレベル参照にはMyTOC 'MyTOC' という名前の目次ブックマークを参照する目次参照が与えられています。これにより、最初のレベル内にあるMyTOC という名前を持った全ての目次ブックマークが、出力に表示される目次からアクセスできるようになります。同様に、ドキュメントの2番目のレベルにあるMyTOC という名前の目次ブックマークが、目次の2番目のレベルから参照されるようになります。ドキュメント3番目のレベルにあるMyTOC という目次ブックマークが、目次内の3番目のアイテムにより使用されます。

このセクション以下にあるサブセクションでは、(i) デザイン内部でレベルを構築する方法 について、そして(ii) ブックマークの作成方法について説明します。 目次テンプレートの作成 は、目次テンプレートの作成のセクションで説明されています。

関連項目

- 目次(TOC)
- レベル内部のデザイン構造
- 目次ブックマークの作成
- 目次テンプレートの作成

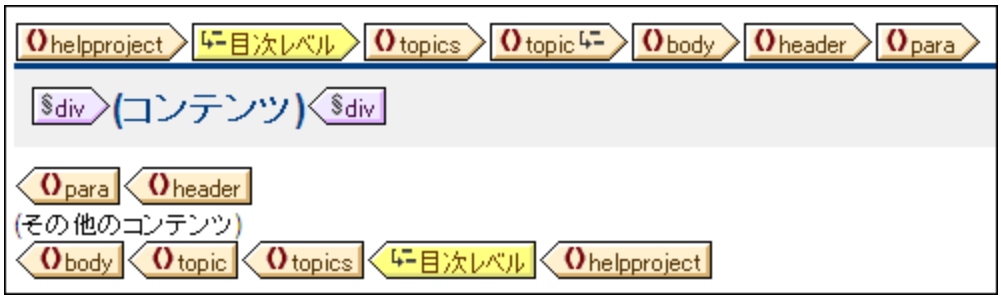
7.7.1.1 目次レベル内部のデザイン構造

目次をデザインするための階層構造は、ネストされたレベルのセットとして指定されます。このような階層構造は、XMLドキュメントの構造に関連しますが、それとは異なるものです。この構造はSPSドキュメントデザイン内部で指定されます。作成する目次テンプレートは、この階層構造に対応するものとなります。フラットな構造(レベルが1つだけの)目次の場合、デザインドキュメントに少なくとも1つのレベルを含める必要があります。ドキュメント内に2つ以上のレベルが存在する場合、これらレベルのどれか、または(1つのレベルとして複数のレベルを統合することにより)複数のレベルに対してフラットな目次を作成することができます。

デザインでは、メインテンプレート、グローバルテンプレート、またはこれらが合わさった内にレベルが作成されます。レベルがどこに作成されるかに関わらず、これらのレベルにより明確な階層構造を構築する必要があります。

レベルの作成

デザイン内部にある各レベルは個別に作成されます。レベルはテンプレートまたはコンポーネントの周りに作成されます。以下にあるスクリーンショットでは、topic テンプレート上に(topic により示される)テンプレートが作成され、topics 要素の周りにも別のレベルが作成されています(目次レベル 目次レベル)。これら2種類のレベルを作成する際の大きな違いは、レベル内に囲むオプション 目次レベル 目次レベル は、テンプレート以外のコンポーネントに対してもレベルを作成することができるという点です。



以下の操作によりレベルを作成することができます:

1. コンポーネント(テンプレートまたはその他のコンポーネント)を選択します。
2. 選択対象を右クリックして、コンテキストメニューから「**レベルとして機能するテンプレート**」(テンプレート作成時に有効になります)または「**タグで囲む | 目次レベル**」を選択します。これらのオプションは、メニューオプションの「**挿入 | 目次を挿入**」以下にある「**目次レベル**」または「**レベルとして機能するテンプレート**」からも選択することができます。

グローバルテンプレート内部のレベル

グローバルテンプレート内部にもレベルをセットすることができます。この場合、グローバルテンプレート内部に作成されたレベルとメインテンプレート内に作成されたレベルの両方により、SPS 実行時に階層構造が定義されるようにする必要があります。以下のスクリーンショットでは、メインテンプレート(topic テンプレート)内に1つのレベルが、そしてグローバルテンプレートのtopic 内部に別のレベルが定義されています。

上のスクリーンショットに示されるコンテンツモデルでは、topic が再帰的な要素となっており、topic 要素内部に、別のtopic 要素を含めることができます。($XML タグにより終了地点が示される)メインテンプレートでは、最初のレベルのtopic にてレベルがセットされています(topic)。メインテンプレートにあるその他のコンテンツにより、header を除く topic/body 以下にある全子要素に対してテンプレートが適用されることが指定されます。つまり、topic/body のtopic 子ノードに対するグローバルテンプレートも処理されることになります。

topic に対するグローバルテンプレートでは、topic テンプレートに対して(topic により示される)レベルがセットされています。このレベルは2番目のレベルのtopic 要素にて出現し、目次では、最初のレベル内部にネストされる2番目のレベルに対応します。このグローバルテンプレートにはその他のコンテンツも含まれており、topic に対するグローバルテンプレートも再帰的な topic 要素に対して適用されることになり、目次階層内にネストされたレベル(3番目のレベル、4番目のレベル、など)が新たに作成されることになります。

デザインを行う際には、デザイン内に作成されたレベルの数に留意する必要があります。目次テンプレートを作成する際には、各レベルにある目次アイテムを明示的に選択し、フォーマットすることになります。

フラット構造の目次におけるレベル

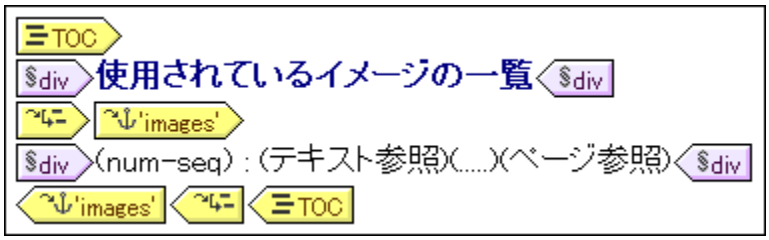
フラットな目次階層構造では、目次アイテムが単一のレベルにて出力されます(使用例: ドキュメント内に表示されるイメージのリスト)。

以下に示される方法により、フラットな構造を取得することができます。

- デザインドキュメントに含める目次レベルを1つだけにする: 目次テンプレートに含まれるレベル参照(そして、その中に含まれる目次参照(TOCref))も、1つだけになります。
- デザインドキュメント内の目次レベルが複数存在する場合: 目次テンプレート内に、参照される目次レベルの位置に対応したレベル参照を配置することになります。ターゲットとなる目次レベルに対応するレベル参照にて、目次参照を1つ配置することになります。
- デザインドキュメントに複数の目次レベルが存在する場合: 目次テンプレート内に配置された目次参照には、ターゲットとなるドキュメントレベル全体をカバーするスコープを指定する必要があります。

ドキュメント内にある全てのイメージのリストを、フラットな構造の目次として表示する例を考えてみましょう。ドキュメントデザインには少なくとも1つのレベルを含める必要があり、このレベルのなかに必要な全ての目次ブックマークを含めることになります。目次テンプレートでは、リストに表示されるイメージが通常と同様の方法で参照されます: (i) 対応する数のレベル参照を作成し、(ii) ターゲットとなる目次レベルに対応するレベル参照にて目次参照を作成します。目次参照には、ターゲットとなる目次レベルにおける目次ブックマークの名前が含まれます。

以下のスクリーンショットにある目次テンプレートでは、images という名前の目次ブックマークを参照する目次参照が含まれているレベル参照が示されます。目次参照のスコープは**現在と下位のレベル**にセットされています。これにより、最初のレベル以下(つまりドキュメント全体)にある image という名前を持った全目次ブックマークが参照されることになります。



デザイン内に複数のレベルが含まれている状況で、例えば2番目のレベルにあるアイテムに対してフラットな目次を作成する必要がある場合、2番目のレベルにだけ目次参照が存在する2つのレベル参照から構成される目次テンプレートを作成することで対応できます(最初のレベルには目次参照を配置しません)。また、目次参照のスコーププロパティにより、与えられた名前のブックマークを検索するデザインドキュメント内のレベルを指定することもできます。

関連項目

- 目次(TOC)
- 目次に含まれるブックマークアイテム
- 目次ブックマークの作成
- 目次テンプレートの作成

(C) 2015-2021 Altova GmbH

- 目次参照のスコーププロパティ

7.7.1.2 目次ブックマークの作成

目次ブックマークはドキュメントデザインの目次レベル内部に作成されます。目次ブックマークはメインテンプレートならびにグローバルテンプレート内部に作成することができます。目次ブックマークは、以下に記される2つの目的のために使用されます:

- デザイン内にある静的ならびに動的コンポーネントを、与えられた名前によりマークします。マーキングはタグにより囲むかたち、または囲まないかたちで行われ、後者の場合は空のタグがデザインに表示されることになります。目次ブックマークの出力は、名前により識別されるアンカーとなります。名前を持ったアンカーが目次(テンプレート)内のアイテムにて参照されることになります。
- 目次ブックマークでは、目次アイテムのテキストとして使用されるテキスト文字列を定義することもできます。このテキスト文字列には、マーカーが配置されているノードの子要素にあるコンテンツ、または XPath 条件式を与えることができます。

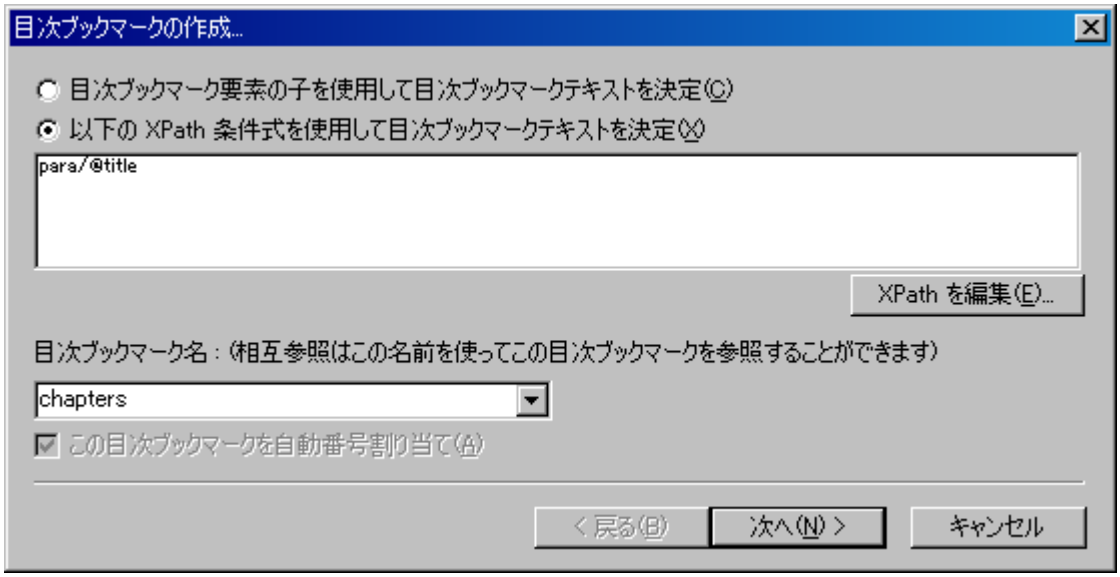
目次ブックマークは、以下にあるいずれかの方法により作成することができます:

- 目次ブックマークの作成 ウィザード を使用する: 目次ブックマークの名前、そのテキストエントリー、自動的に番号が割り当てられるか、表示されるレベルを指定します。
- 空の目次ブックマークを挿入 する: ブックマークのプロパティを、ブックマークの作成後個別に指定することになります。

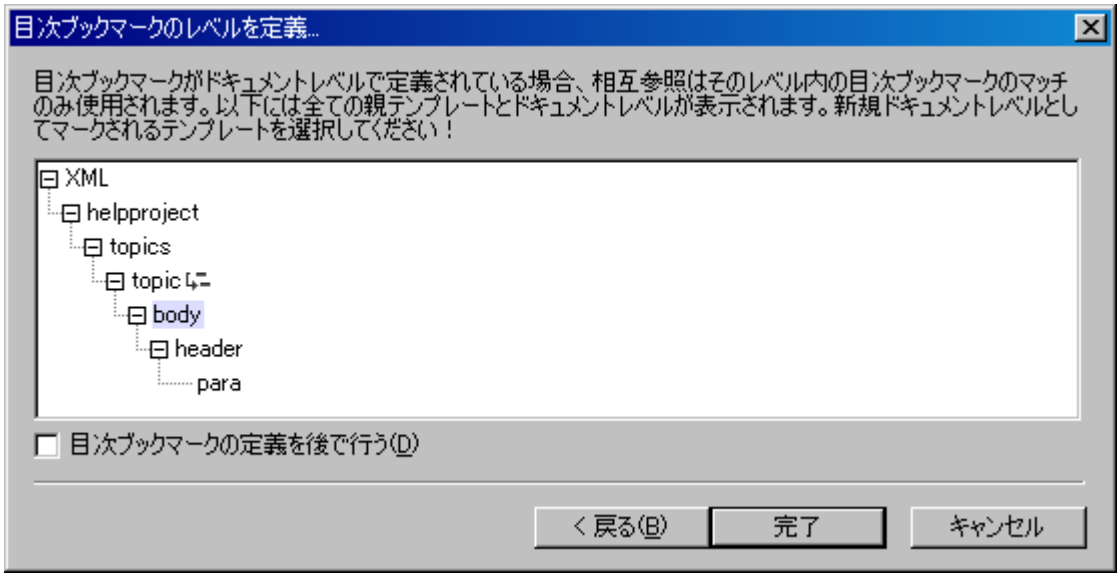
目次ブックマーク ウィザードにより目次ブックマークを作成する

以下に記される様に、目次ブックマークの作成ウィザードにより目次ブックマークを作成することができます:

1. 目次ブックマークを作成するデザイン位置にカーソルを配置します。目次ブックマークにより囲まれるデザインコンポーネントを選択することもできます。
2. 右クリックにより表示されるコンテキストメニューから「**挿入 | 目次を挿入 | 目次ブックマーク(ウィザード)**」を選択するか、目次ブックマークをノードで囲む場合、メニューオプションから「**タグで囲む | 目次ブックマーク(ウィザード)**」を選択します。目次ブックマークの作成ウィザードが表示されます(以下のスクリーンショットを参照)。



3. ウィザードに表示される最初の画面（上のスクリーンショット）では、(i) 目次アイテムのテキスト、(ii) 目次ブックマークの名前、また (iii) 出力にてこの目次ブックマークに番号が与えられるかを指定することができます。テキストのエントリーには、子要素のテキストを使用するように指定することができるほか、XPath 条件式によるテキストの指定を行うことができます。目次ブックマークの名前には、テキストを直接入力することができるほか、定義済みの目次ブックマークの名前をドロップダウンリストから選択することができます。指定を終えた後は**次へ**をクリックしてください。
4. 次のウィザード画面では、目的に応じてテンプレート上に目次レベルを作成することができます。挿入ポイントから見た祖先テンプレートがツリーには表示されます。目次レベルにテンプレートが既に作成されている場合、シンボルによりその旨が示されます。以下のスクリーンショットでは、topic テンプレートの隣にあるシンボルにより、そのテンプレートがレベルとして作成されていることが示されます。祖先テンプレートにて別のレベルを作成する場合、そのテンプレートを選択してください。**定義を後で行う**チェックボックスにチェックを入れることで、レベルの定義を後から行うこともできます。選択を終えた後には**完了**をクリックしてください。目次レベルがテンプレート上に既に存在する場合、**完了**をクリックしてもそのテンプレートに目次レベルが作成されることはありません。



完了をクリックすると、目次ブックマークが挿入ポイントへ作成され、ウィザードにて指定されたテンプレート上に目次レベルが作成されます（テンプレートが指定された場合）。作成された目次ブックマークは、それを含んでいる目次レベル直下に配置されます。例えば目次レベルが目次階層レベルの３番目だとすると、作成された目次ブックマークは３番目の目次レベル内に配置されます。

目次ブックマークの作成

以下の操作により、属性（目次ブックマーク名ならびに目次アイテムテキスト）無しの目次ブックマークを作成することができます:

1. 目次ブックマークを作成するデザイン位置にカーソルを配置します。目次ブックマークにより囲まれるデザインコンポーネントを選択することもできます。
2. 右クリックにより表示されるコンテキストメニューから「**タグで囲む | 目次ブックマーク**」を選択するか、メニューオプションから「**挿入 | 目次を挿入 | 目次ブックマーク**」を選択します。名前やテキストエントリーが含まれない目次ブックマークが挿入されます。これら値の定義方法については、以下にある 編集コマンド を使用してください。

コンポーネントに対して階層的または連続下番号を挿入する

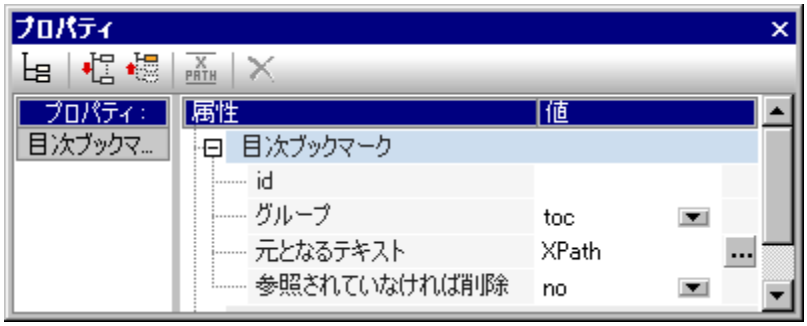
目次ブックマークには出力ドキュメントの（目次ブックマークではなく）メインボディーにおける階層的な、または連続した番号を挿入することができます。番号を挿入する場所を右クリックして、コンテキストメニューから「**目次を挿入 | 階層的な番号付け / 連続的な番号付け**」を選択します。例えば、章の名前を表すテンプレートの周りに自動番号の目次ブックマークを配置することにより、章の名前を表すテンプレートにより生成された全ての章に対して番号が生成されるようになります。

(C) 2015-2021 Altova GmbH

番号は目次レベルの構造をベースとしていることに留意してください。例えば、章のタイトル要素が最初の目次レベルとなっている場合、第4章がその目次レベルにおける4番目のタイトルインスタンスとなるため、そのタイトルには4という番号が与えられます。章の中にある節に対して番号付けが行われる場合、例えば第4章にある第3節には4.3 という番号が与えられます。これは、最初の目次レベル(章)におけるインスタンスが4番目で、2番目の目次レベル(節)におけるインスタンスが3番目となるためです。

目次ブックマークの名前とテキストエントリーを編集

に表示されているプロパティを直接編集するか、目次ブックマークを右クリックして、編集するプロパティを選択し、これらのプロパティを編集することができます。プロパティウィンドウに表示されているプロパティを直接編集するか、目次ブックマークを右クリックして、編集するプロパティを選択し、これらのプロパティを編集することができます。



目次ブックマークには以下のプロパティが含まれています: (i) 目次ブックマーク(グループ)の名前、(ii) ユニークID、(iii) 参照されていないブックマークの削除、(iv) ブックマークのコンテンツまたは XPath 条件式により取得されるテキストエントリーオプション。

関連項目

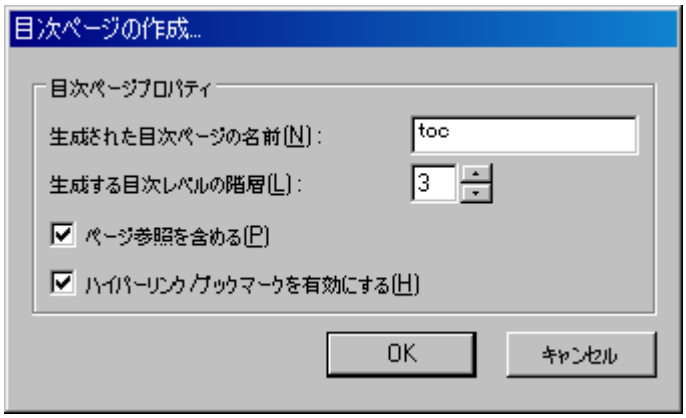
- 目次(TOC)
- 目次に含まれるブックマークアイテム
- レベル内部のデザイン構造
- 目次テンプレートの作成

7.7.2 目次テンプレートの作成

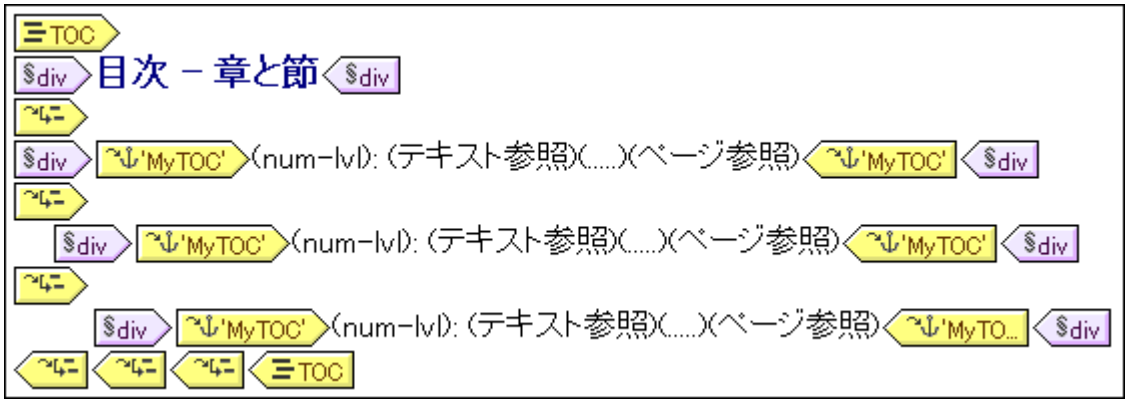
目次テンプレートは出力にて目次を生成するためのテンプレートのことで、SPS デザイン内の任意の場所で作成することができます。複数の目次テンプレートを1つのSPS デザインにて作成することもできます。

目次テンプレートを作成するためのステップを以下に記します:

1. 目次テンプレートを挿入する箇所にカーソルを配置します。
2. メニューオプションから「**挿入 | 目次を挿入 | 目次**」を選択します。目次ページの作成ダイアログが表示されます(以下のスクリーンショットを参照)。右クリックにより表示されるコンテキストメニューからも同じコマンドを選択することができます。



3. ダイアログにて必要な情報を入力します: (i) 目次ページの名前は、デザインドキュメントにある 目次ブックマーク の参照に使用される目次参照名のことです。目次に対して複数のレベル(レベル参照)を選択した場合、同じ目次参照名が全ての名前参照に対して使用されます(個々の目次参照の名前を 後から編集することもできます)。(ii) 指定された 目次レベル参照 (levelrefs) の数により、目次内に含まれるレベル参照の数が決定されます。. (iii) 印刷媒体に対しては、ページ参照(ページ番号)の出力オプションが与えられています。(iv) 目次内のテキストエントリーは、目次ブックマークに対するリンクとして使用することができます。
4. 「OK」をクリックすることで完了します。指定された数のレベル参照とともに目次テンプレートが作成されます。以下のスクリーンショットでは、目次テンプレートのフォーマットが修正されています。



各レベル参照内部には目次参照が含まれており、対応するレベル参照の目次アイテムとなる目次ブックマークを識別するための名前が目次参照には含まれています。各目次参照には目次アイテムに対するデフォルトテンプレートが含まれており、必要な時には いつでも編集することができます。

テンプレート目次テンプレートの編集

以下にある編集オプションを利用することができます:

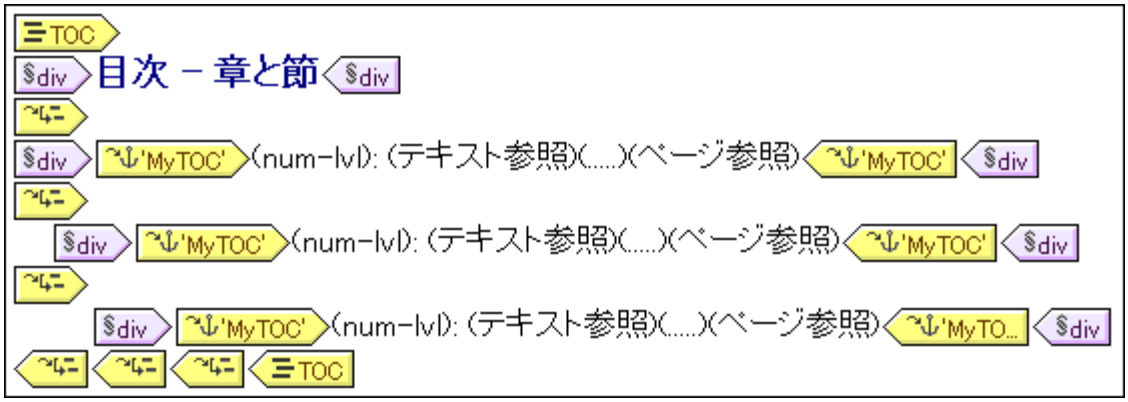
- SPS における別の場所へ目次テンプレートをドラッグすることができます。しかし、この操作によりコンテキストノードが変更された場合、目次テンプレート内部にある XPath 条件式が影響を受ける場合もあります。
- レベル参照 を目次テンプレートの構造へ追加、または削除することができます。
- 個々の目次参照におけるプロパティ (TOCrefs) を編集することができます。目次参照の名前ならびにスコープを変更し、目次参照に対応する目次アイテムがハイパーリンクとして作成されるかを指定することができます。
- 目次参照 を目次テンプレートのレベル参照へ追加、または削除することができます。
- 目次参照内にある目次アイテムは、標準的な StyleVision メカニズムによりCSS プロパティを介したフォーマットを行うことができます。
- イメージ、自動計算、ブロックフォーマットコンポーネントといった標準的な SPS の機能は、目次テンプレートでも使用することができます。

関連項目

- 目次に含まれるブックマークアイテム
- 目次テンプレートにおけるレベル参照 (Levelrefs)
- 目次参照: 名前、スコープ、ハイパーリンク
- 目次アイテムのフォーマット

7.7.2.1 目次テンプレートにおけるレベル参照

目次テンプレート はレベル参照にて構築されます(以下のスクリーンショットを参照)。これらのレベルは目次テンプレートの作成時に作成され、目次ページの作成ダイアログにて指定された数のレベル参照が作成されます。



レベル参照がネストされているということに注目してください。目次テンプレート内にあるレベル参照と、SPS デザイン内にあるレベルが1対1で対応することになります。従って、目次テンプレート上にある最初のレベル参照はSPS デザイン内にある最初のレベルに対応することになり、目次テンプレートにある2番目のレベル参照はSPS デザイン内にある2番目のレベルに対応することになります。目次テンプレートのレベル参照により、SPS 内で指定されたスコープ内にある目次ブックマークが識別されます。例えば目次参照のレベルに対応したドキュメントレベルの目次ブックマークを参照のターゲットとすることができるほか、全ドキュメントレベルにある目次ブックマークをターゲットにし、現在のドキュメントレベルとそれ以下のドキュメントレベルにある目次ブックマークをターゲットとすることができます。

レベル参照の挿入と削除

目次テンプレートが作成されると、レベル参照を目次テンプレートへ挿入し、削除することができます。

コンテンツの周りにレベル参照を挿入するには、レベル参照を作成する目次テンプレート内のコンテンツを選択し、コンテキストメニューまたはメニューオプションから「**タグで囲む | 目次レベル参照**」を選択します。メニューオプションまたはコンテキストメニューから選択することのできる「**挿入 | 目次を参照 | 目次レベル参照**」を選択することで、カーソル位置に空のレベル参照を挿入することもできます。

目次テンプレートからレベル参照を削除するには、削除するレベル参照が選択されている状態で、Delete キーを押下するか、コンテキストメニューから**削除**を選択します。削除されるのはレベル参照だけで、内部にあるコンテンツは削除されないということに注意してください。

関連項目

- 目次テンプレートの作成
- 目次に含まれるブックマークアイテム

- 目次参照：名前、スコープ、ハイパーリンク

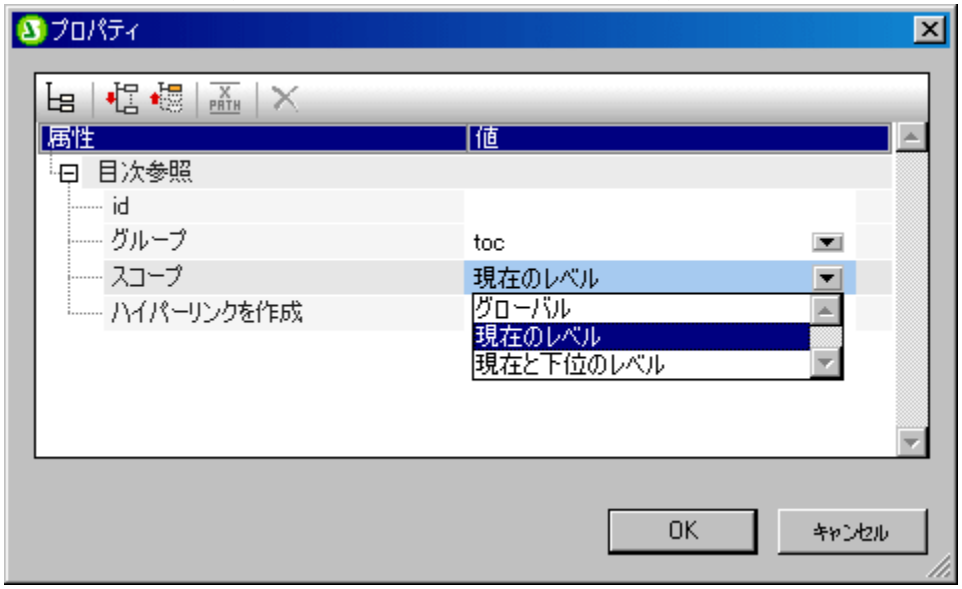
7.7.2.2 目次参照：名前、スコープ、ハイパーリンク

レベル参照内に配置される目次参照には、以下に記される4つのプロパティが与えられています（以下のスクリーンショットを参照）：

- **ハイパーリンクを作成**：yes または no の選択を行うことができ、対応する目次アイテムをハイパーリンクとして作成するか指定することができます。
- **グループ**：目次参照の名前で、指定されたスコープ内にある同名の目次ブックマークを特定するのに使用されます（下記を参照）。該当する目次ブックマークが、目次のレベル参照内に含まれます。
- **ID**：目次参照を一意に特定するための値です。
- **スコープ**： 目次参照が適用されることになる SPS デザイン内部のレベルが指定されます。3つのオプションを利用することができます：(i) グローバル、(ii) 現在のレベル、(iii) 現在と下位のレベル（以下のスクリーンショットを参照）。

目次参照を挿入するには、レベル参照内部にカーソルを配置して、メニューオプションにある「**挿入**」またはコンテキストメニューから、「**目次を挿入 | 目次参照**」を選択してください。

目次参照のプロパティを編集するには、目次テンプレート内にある目次参照タグを右クリックして編集するプロパティ（**ハイパーリンクの作成**、**ID を編集**、**グループを編集**、または**スコープの編集**）を選択します。この操作によりプロパティウィンドウが表示され、選択されたプロパティの値を指定することができます（以下のスクリーンショットを参照）。



目次参照タグが選択されている状態で、プロパティウィンドウにある目次参照プロパティグループから直接プロパティの値を編集することもできます。

関連項目

- 目次に含まれるブックマークアイテム
- 目次テンプレートの作成
- 目次テンプレートにおけるレベル参照（Levelrefs）

(C) 2015-2021 Altova GmbH

7.7.2.3 目次アイテムのフォーマット

目次アイテムには最大4種類の標準コンポーネントに加え、オプションのユーザー指定コンテンツを加えることができます。以下に4種類のコンポーネントを示します（以下のスクリーンショットを参照）：

- （**テキスト参照**）により示される目次アイテムのテキストエントリー
- （....）により示される、テキストエントリーとページ番号の間に示されるリーダー（印刷媒体に対して適用）
- （**ページ参照**）により示される目次アイテムのページ参照（印刷媒体に対して適用）
- （num-lvl）または（num-seq）により示される階層的または連続した番号

目次テンプレートが作成されると、目次参照内にテキストエントリーが自動的に挿入されます。ページ参照を含めるオプションが選択された場合、リーダーとページ参照コンポーネントも追加されます。目次アイテムからコンポーネントの挿入や削除を行うこともできます。コンポーネントを挿入するには、目次アイテム内にある目的の場所にカーソルを配置して、右クリックにより表示されるコンテキストメニューから、「**目次を挿入 | 目次参照 | 入力テキスト / リーダー / ページ参照**」または「**目次を挿入 | 階層的な番号付け / 連続的な番号付け**」を選択します。ネストされたレベルによりデザインが構成されている場合には階層的な番号を使用し、階層構造が存在しない場合（つまり1つのフラットな目次レベルだけが存在する場合）には連続的な番号を使用します。フラットな目次に関する下記の記述を参照ください。コンポーネントを削除するには、そのコンポーネントが選択された状態で**Delete** キーを押下してください。

更に、目次アイテムには静的なコンテンツ（例：テキスト）や動的なコンテンツ（例：自動計算）を挿入することもできます。

フラットな目次におけるレベル

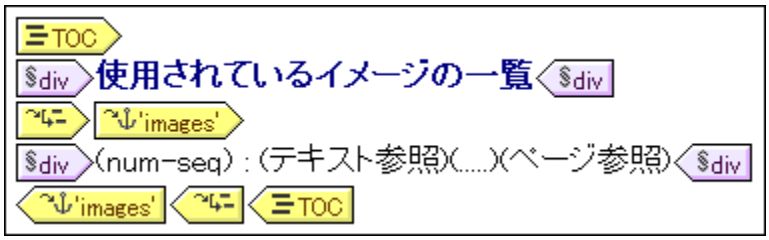
フラットな目次階層構造では、目次アイテムが単一のレベルにて出力されます（使用例：ドキュメント内に表示されるイメージのリスト）。

以下に示される方法により、フラットな構造を取得することができます。

- デザインドキュメントに含める目次レベルを1つだけにする：目次テンプレートに含まれるレベル参照（そして、その中に含まれる目次参照(TOCref)）も、1つだけになります。
- デザインドキュメント内の目次レベルが複数存在する場合：目次テンプレート内に、参照される目次レベルの位置に対応したレベル参照を配置することになります。ターゲットとなる目次レベルに対応するレベル参照にて、目次参照を1つ配置することになります。
- デザインドキュメントに複数の目次レベルが存在する場合：目次テンプレート内に配置された目次参照には、ターゲットとなるドキュメントレベル全体をカバーするスコープを指定する必要があります。

ドキュメント内にある全てのイメージのリストを、フラットな構造の目次として表示する例を考えてみましょう。ドキュメントデザインには少なくとも1つのレベルを含める必要があり、このレベルのなかに必要な全ての目次ブックマークを含めることになります。目次テンプレートでは、リストに表示されるイメージが通常と同様の方法で参照されます：（i）対応する数のレベル参照を作成し、（ii）ターゲットとなる目次レベルに対応するレベル参照にて目次参照を作成します。目次参照には、ターゲットとなる目次レベルにおける目次ブックマークの名前が含まれます。

以下のスクリーンショットにある目次テンプレートでは、images という名前の目次ブックマークを参照する目次参照が含まれているレベル参照が示されます。目次参照のスコープは**現在と下位のレベル**にセットされています。これにより、最初のレベル以下（つまりドキュメント全体）にあるimage という名前を持った全目次ブックマークが参照されることになります。



デザイン内に複数のレベルが含まれている状況で、例えば2番目のレベルにあるアイテムに対してフラットな目次を作成する必要がある場合、2番目のレベルにだけ目次参照が存在する2つのレベル参照から構成される目次テンプレートを作成することで対応できます(最初のレベルには目次参照を配置しません)。また、目次参照のスコーププロパティにより、与えられた名前のブックマークを検索するデザインドキュメント内のレベルを指定することもできます。

書式 目次アイテム

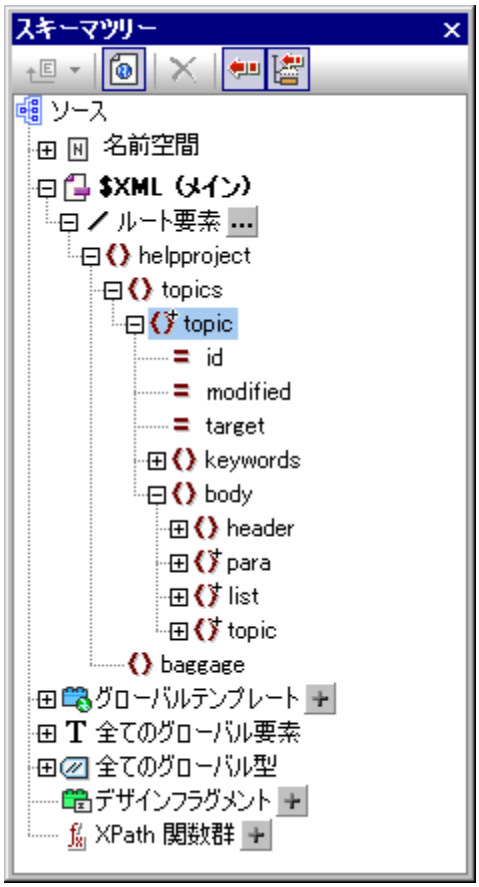
目次アイテムは スタイルサイドバーにて CSS スタイルによるフォーマットを行うことができます。コンポーネントを選択して、スタイルサイドバー スタイルプロパティにてスタイルを指定することにより、個々の目次アイテムコンポーネントのスタイルを指定することができます。

関連項目

- CSS スタイルと作業
- 目次テンプレートの作成

7.7.3 サンプル: シンプルな目次

ChaptersSimple.sps という名前の目次の基本的な機能のデモを行うためのサンプルSPS ファイルが(マイ)ドキュメントフォルダー 以下にある、`C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutoria;\TOC` に収められています。このSPS は、章ベースのドキュメントを記述するためのコンテンツモデルにより構成されるスキーマをベースにしています。ChaptersSimple.sps を StyleVision にて開くと、スキーマツリーウィンドウにてスキーマの構造を確認することができます(以下のスクリーンショットを参照)。次にあるセクションの、 サンプル: 階層ならびに連続的な目次 では、同じスキーマをベースにしたより複雑な目次サンプルについて記述されます。



ドキュメント要素のhelpproject には、子要素にtopics 要素が含まれています。topics 要素にはtopic 要素を任意の数だけ含めることができ、各々のtopic 要素には別のtopic 子孫要素を含めることができます。最初のレベルにあるtopic 要素はドキュメントの章として考えることができ、その子孫要素のtopic 要素を節や小節として考えることができます。

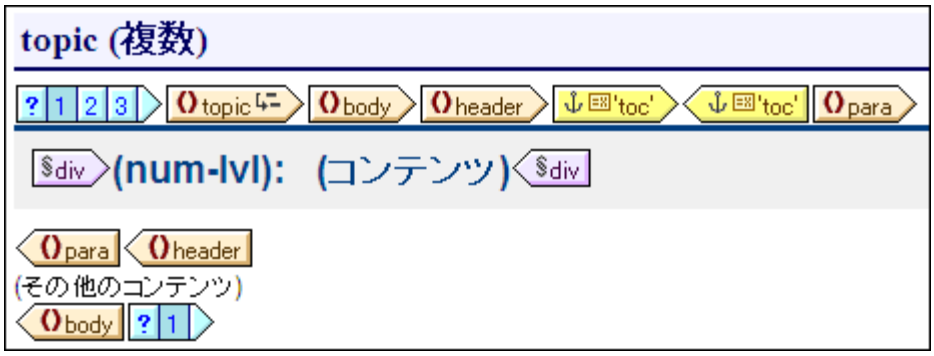
このSPS により、ドキュメントの上部に配置されている目次が作成され、各章の名前(最初のレベルにあるtopic)が表示されます。目次の作成を行うには以下に記されるステップを踏むことになります:

1. 目次レベル内部のデザイン構造: 1つ以上のレベルをデザインドキュメントへ挿入することで、(出力)ドキュメントの構造が作成されます。この階層構造が、目次へ反映される構造となります。このサンプルでは説明を分かりやすくするために、1つの**目次レベル**だけが Topic テンプレートにて作成されています。デザイン内にあるレベルは1つしか無いため、後になって**目次テンプレート**が作成された場合でも、最初のレベルしか意味を持たないことになります。
2. 目次ブックマークの作成: ステップ1 で作成された目次レベル内に**目次ブックマーク**を(デザインドキュメント内部に)作成します。これにより、目次テンプレート内部にある目次参照から、目次ブックマークへの指定を行うことができるようになります。目次ブックマークでは、目次アイテムに表示されるテキストを指定することもできます。
3. 目次テンプレートの作成: ドキュメント内部に目次を作成するためのテンプレートです。デザインドキュメント内にある目次レベルの構造に対応する**レベル参照**により構成されます。例えば、目次テンプレート内に3つのネストされたレベル参照が存在する場合、少なくとも3つのネストされたレベルをデザインドキュメントにも作成する必要があります。このセクションで紹介される例には、デザインドキュメント内にある1つの目次レベルに対応するかたちで、レベル参照が1つだけ作成されています。レベル参照内部には**目次参照**が配置されます。目次参照により、目次にて対応するレベルの目次アイテムが生成されます。

SPS の構造とレベル

SPS におけるデザインの構造を確認してください。(緑色の $XML タグが表示されている) メインテンプレートには目次が含まれていることに注目してください。メインテンプレートにある (その他のコンテンツ) により、グローバルテンプレートならびにデフォルトテンプレートが適用されます。メインテンプレートの外側にあるその他の SPS デザインが、グローバルテンプレートとなります。

目次定義 (デザインにある目次レベルならびに目次ブックマーク) は、topic に対するグローバルテンプレート内に配置されています (以下のスクリーンショットを参照)。このグローバルテンプレートには条件分岐タグが挿入されており、topic 要素の祖先要素がいくつあるかに従って topic 要素の切り分けが行われ、章、節、小節に対して異なる処理が行われることになります。



上のスクリーンショットには、最初の条件分岐 (つまり最初のレベルで章に対する topic 要素) のコンテンツが示されています。topic 要素の開始タグにて目次レベルが作成されている点に注目してください。その他2つの条件分岐では、topic テンプレートに対して条件分岐は作成されておらず、結果として1つの目次レベルにしかドキュメントは割り当てられず、それが (章に対する) 最初のレベルの topic 要素となります。

目次ブックマークの作成

topic 要素の子孫である header 内部 (para 要素の外側) に目次ブックマークが作成されています (以下のスクリーンショットにある黄色のタグ)。目次ブックマークは、トップレベルの各 topic 要素に対するアンカーとして使用されます。

目次ブックマークのプロパティはプロパティサイドバーにて編集することができます (以下のスクリーンショットを参照)。

グループプロパティでは目次ブックマークグループがセットされます。このサンプルの場合、このプロパティに対して MyTOC という値が指定されています。ブックマークが作成されると、ブックマークグループが目次にて参照され、同じレベル内に配置された異なる目次グループを指定するのに使用することができるようになります。ID プロパティにより、作成されたブックマークインスタンスに対してユニークな ID を指定することができま

(C) 2015-2021 Altova GmbH

す。**参照されていなければ削除**プロパティにより、参照されていないブックマークが削除されるようになります。**使用されるテキスト**プロパティにより、目次における目次アイテムのテキストとして使用されるテキストエントリーを指定することができます。ブックマークのコンテンツ(デザインにおけるブックマークの開始ならびに終了タグ間にあるコンテンツ)、またはXPath 条件式からテキストを取得することができます。このサンプルでは、XPath 条件式を使用することで、最初のレベルにあるtopic 要素のヘッダーテキストが返されます。

目次テンプレート

目次テンプレート内部では、最初のレベル参照 (levelref) が挿入されています(以下のスクリーンショットを参照)。このレベル参照は、章を表すtopic 要素にて作成された最初のレベルに対応することになります(上記の**SPS の構造とレベル**を参照ください)。

このレベル参照内部には目次参照 (TOCref) が挿入されます。この目次参照により、(i) MyTOC という名前のブックマークグループ内にあるブックマークの選択(上記の**目次ブックマークの作成**を参照)、そして(ii) スコープ内にあるブックマークの選択が行われます。これらの設定は目次参照が選択されている状態でプロパティサイドバーにて行うことができるほか、デザイン内の目次参照を右クリックして、コンテキストメニューから関連する編集コマンドを選択することで行うことができます。

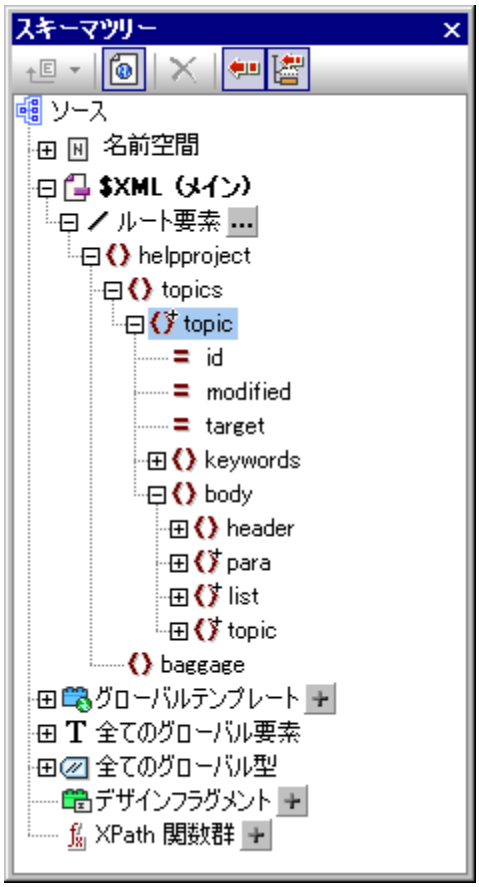
目次アイテムの外観は、目次の目次参照タグにて指定することができます。目次参照タグ内部を右クリックして、コンテキストメニューから挿入するコンポーネントを選択することで、番号のフォーマット、テキスト、リーダー、そしてページ参照を挿入することができます。これらの各コンポーネントをデザインにて選択し、プロパティサイドバーにあるプロパティを修正することで、コンポーネントの編集を行うことができます。

関連項目

- 目次に含まれるブックマークアイテム
- 目次テンプレートの作成

7.7.4 サンプル: 階層的ならびに連続的な目次

Chapters.sps という名前の目次機能のデモを行うためのサンプルSPS ファイルが、(マイ)ドキュメントフォルダー以下にある`C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\TOC` に収められています。このSPS は、章ベースのドキュメントを記述するためのコンテンツモデルにより構成されるスキーマをベースしています。Chapters.sps をStyleVision にて開くと、スキーマツリーウィンドウにてスキーマの構造を確認することができます(以下のスクリーンショットを参照)。



ドキュメント要素のhelpproject には、子要素にtopics 要素が含まれています。topics 要素にはtopic 要素を任意の数だけ含めることができ、各々のtopic 要素には、別のtopic 子要素を含めることができます。最初のレベルにあるtopic 要素をドキュメントの章として考えることができ、その子孫要素のtopic 要素を節や小節として考えることができます。

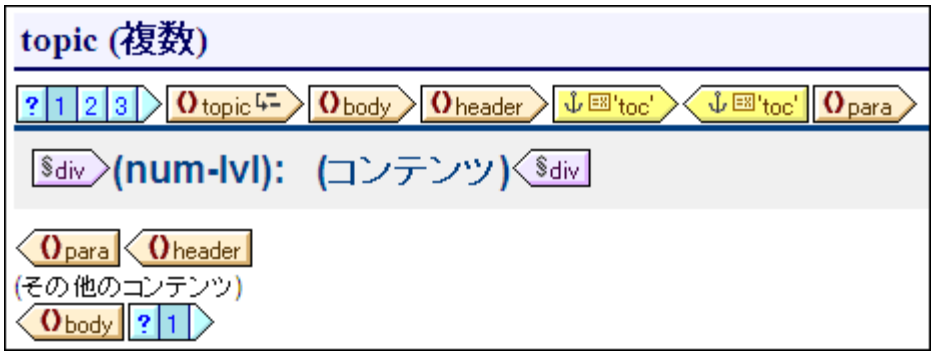
SPS には3つの目次が含まれており、以下の順序でドキュメントの先頭箇所に配置されています:

1. *チャプターの概要*: 各章(最初のレベルにある topics)の名前が表示されます。
2. *チャプターとそのセクション*: 章と、その子孫(最初のレベルにある topics に加え、XML ドキュメントの chapters.xml によれば3つのレベルまである各 topics の子孫構造)が表示されます。
3. *イメージのリスト*:ドキュメント内部に存在する(最初のものを除く)全イメージのファイル名をフラットな構造により表示します。

SPS の構造

目次の詳細を確認する前に、デザインの構造に目を向けてみましょう。(緑色の$XML タグが表示されている)メインテンプレートには目次が含まれていることに注目してください。メインテンプレートにある(その他のコンテンツ) により、グローバルテンプレートならびにデフォルトテンプレートが適用されます。

目次定義はtopic ならびにimage に対するグローバルテンプレートにて行われます。グローバルテンプレート内では、目次レベルがtopic 要素に対して作成されており、目次ブックマークがheader 要素にて定義され、header 子要素内部(para 要素の外側)にて目次ブックマークが作成されています(以下のスクリーンショットを参照)。



topic 要素は再帰的にデザインされているため、目次レベルならびに目次ブックマークも再帰的になります。この再帰構造により、最初の再帰にて下位の目次レベルならびに目次ブックマークが新たに作成されます。この処理は各子孫 topic に対して行われ、下位の目次レベルと、そのレベルにおける目次ブックマークが作成されることになります。ヘッダー(topic タイトル)フォーマットは目次レベルにより異なるため、レベルのテンプレートを条件タグで囲むことで、各レベルに対応した3つの条件分岐を定義しています。各条件分岐により、topic が出現する最初、2番目、そして3番目のレベルに対して異なる処理が行われます。

階層的な番号を表す(num-lvl) がレベル内部に挿入されていることに注目してください。目的の場所を右クリックして、コンテキストメニューから「**目次を挿入 | 階層的な番号付け**」を選択します。これにより、各 topic タイトルに対して(4.2.3 という形式の)階層的な番号付けが行われます。

目次の説明

ドキュメントにて定義された各目次と、その説明を以下に記します。

チャプターの概要: topic に対するグローバルテンプレート内にある目次ブックマークを選択します。プロパティサイドバーでは XPath 条件式を使ったテキストの構築が行われている点に注目してください。使用されるテキストプロパティの値フィールドにあるボタンをクリックすると、定義された XPath 条件式が para となっていることが確認できます。これにより、(header 要素内に目次ブックマークが挿入されているため) header の para 子要素にあるコンテンツが目次アイテムのテキストとして使用されます。

以下のスクリーンショットにて示される目次テンプレートには、レベル参照 (levelref) が1つ含まれており、そのレベル参照内部には MyTOC という名前の目次ブックマークを選択する目次参照 'MyTOC' が含まれています。目次参照のスコープは現在のレベル(つまり最初のレベル)だけとなっており、結果として最初のレベルにある目次アイテムだけが作成されます。

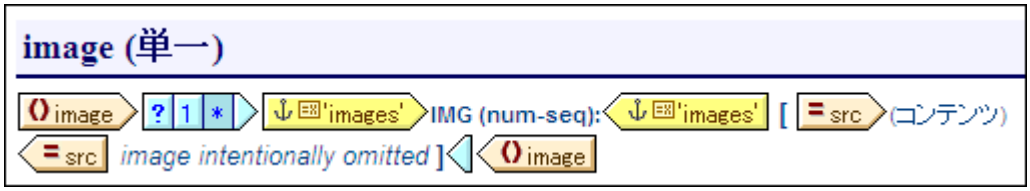
階層的な番号を表示するためのアイテムが定義されている点に注目してください。

チャプターとそのセクションセクション: 以下のスクリーンショットに示される目次では、3つのレベルがネストされたかたちで定義されており、各レベルには現在のレベルをスコープとする目次参照が含まれています。

各目次アイテムには div ブロックが含まれており、インデントなどのフォーマットを各ブロックに対してセットすることができます。

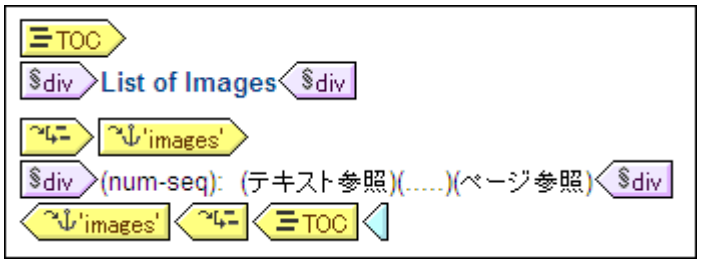
イメージのリスト: フラット構造の目次としてイメージのリストを表示します。まずはインスタンス化されたドキュメントにて、イメージが出現するレベルを考えてみましょう。image 要素は para 要素の子となります。topic 要素にてレベルを作成した場合、ドキュメント内にある最初、2番目、そして3番目のレベル内部にて image 要素が出現することになります。image 要素に対して新たなレベルを作成する必要はありません。

image に対するグローバルテンプレートでは、条件分岐により (i) 最初のイメージと (ii) 別のイメージが個別に処理されることになります。



目次ブックマークが2番目の条件分岐内だけに目次ブックマークが配置されていて、最初の条件分岐の中にあるイメージはブックマークされていないということに注目してください。更に、イメージの連続的な番号付けを表す (num-seq) が「**目次を挿入 | 連続的な番号付け**」により挿入され、(最初のイメージは最初の条件分岐の中にあるため) 2番目のイメージから番号付けが行われます。番号はフォーマットすることもでき、(num-seq) を右クリックして、**フォーマットの編集**を選択するとフォーマットを指定するためのダイアログボックスが表示されます。このサンプルでは 01 という値が定義されており、1桁の数値の前に 0 が挿入されることになります。

今のスクリーンショットで示されるイメージに対する目次テンプレートでは、images という名前のブックマークを識別するための目次参照が1つ配置されており、この目次参照がレベル参照内部に配置されています。目次参照のスコープは「現在と下位のレベル」にセットされています (この値は目次参照の選択時にプロパティウィンドウから編集することができます)。レベル参照により決定された現在のレベルは最初のレベルとなり、そのレベル以下にあるレベルが2番目のレベルとなります。これにより、最初のレベル以下にある全てのイメージが目次内部のアイテムとして選択されます。



選択された番号付けは連続的なものとなるため、フラットなリストに連続した番号が表示されることになります。

関連項目

- 目次に含まれるブックマークアイテム
- 目次テンプレートの作成

7.7.5 ドキュメントボディー上の自動番号

番号の自動割り当て機能を使用することで、複数回出現するノードのインスタンスをメインボディー内部で自動的に番号付けすることができます。例えば複数のChapter 要素が含まれるBook 要素にて番号割り当てを使用することで、個々のChapter 要素に対して番号を自動的に与えることができます。これによりXMLドキュメントの構造をベースにした番号の挿入を簡単に行うことができるようになります。

メモ: 自動的に番号を割り振る機能は、ドキュメントのメインボディー内にある番号を参照しており、番号が目次アイテムのプロパティとして識別される目次内の番号を参照するのでは**ありません**。

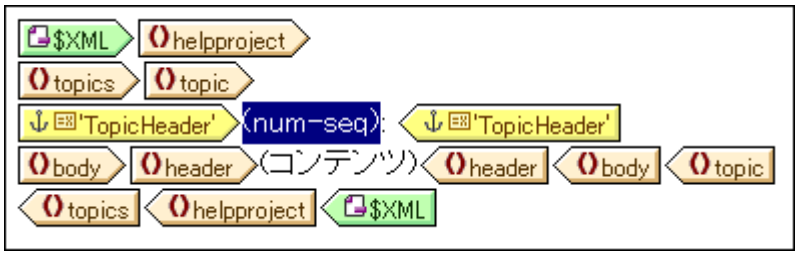
自動的に与えられる番号は連続的なもの(フラット)または階層的なものとなります。連続的な番号の場合、同一レベルにて番号付けが行われることになります。階層的な番号はドキュメントにて作成された目次レベルの階層をベースに番号が与えられ、目次レベルの階層における要素の位置に対して番号が与えられます。

番号には様々な種類のフォーマットを適用することができます。階層的な番号付けの場合、個々の番号トークンが個別にフォーマットされることになります。例えば、3つのトークンから成る番号に対してA.1.i. というフォーマットを与えることにより、それぞれのトークンが異なる番号フォーマットで表示されるようになります。番号のフォーマットは、以下に示されるように、連続的な番号付けと階層的な番号付けにて異なる方法で割り当てられます。

連続的な番号付け(num-seq)

連続的な(またはフラットな)番号付けは、ドキュメントデザイン内部にある目次ブックマークに挿入することができます(以下のスクリーンショットを参照)。以下の操作により連続的な番号を作成することができます:

1. 番号付けを行うノード内部にカーソルを配置し、右クリックにより表示されるコンテキストメニューから「**目次を挿入 | 目次ブックマーク**」を選択します。目次ブックマークが作成されます。以下のスクリーンショットでは topic 要素に対して番号付けを行うため、topic 要素の周りに目次ブックマークが作成されています。topic 要素のどこに番号付けを行うかにより、番号が表示される正確な場所が指定されます。以下にあるスクリーンショットでは、章ヘッダー(タイトル)の直前に番号が配置されています。
2. 目次ブックマークのタグ内部にカーソルを配置して、右クリックにより表示されるコンテキストメニューから「**目次を挿入 | 連続的な番号付け**」を選択します。この操作により連続的な自動番号のプレースホルダ (num-seq) が挿入されます。以下のスクリーンショットでは、目次ブックマークの TopicHeader 内部にある (num-seq) がハイライトされています。



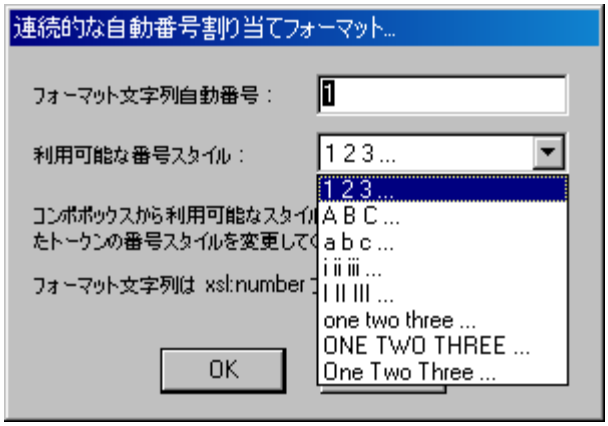
3. 目次ブックマークが目次テンプレート内部から参照される場合、必要に応じて目次ブックマークのプロパティを入力してください。目次ブックマークが連続的な番号付けだけに使用される場合、名前を入力する必要はありません。名前を入力するには、目次ブックマークを右クリックして、**グループを編集**コマンドを選択してください。

上のスクリーンショットで示された例では、連続的な番号がtopic ノードに対してセットされています。これにより、以下のスクリーンショットに示されるように、各 topic 要素に対して連続的な番号が与えられます。XML ドキュメントの階層における topic 要素シーケンスにおける位置をベースに、topic 要素の番号が決定されている点に注目してください。

1: Altova StyleVision 2007
2: About this Documentation
3: Introduction
4: User Interface

メモ: 連続的な番号を他のノードセットでも続けて使用する必要がある場合、両方のノードセット内にある目次ブックマークに対して同じ名前を使用してください。

連続的な番号のフォーマットを行うには、num-seq プレースホルダを右クリックして、**フォーマットの編集**コマンドを選択します。**連続的な自動番号割り当てフォーマット**ダイアログが表示されます(以下のスクリーンショットを参照)。



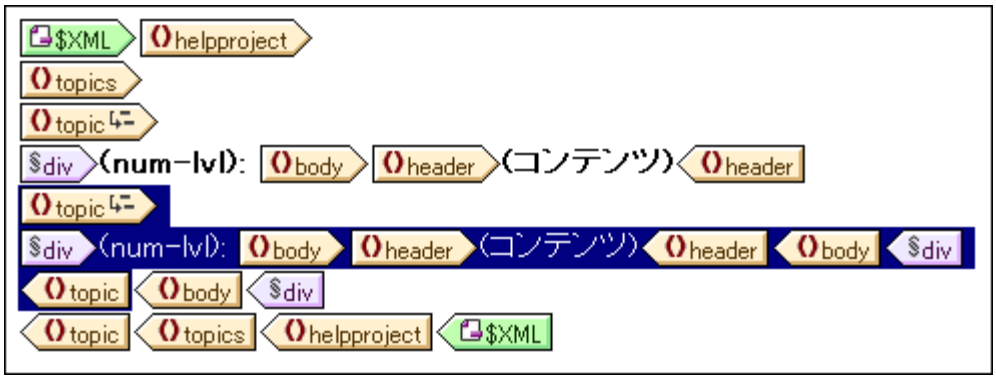
利用可能な番号スタイルのドロップダウンボックスから、目的のフォーマットを選択し、OK をクリックすることで選択されたフォーマットを適用します(上のスクリーンショットを参照)。

構造的な番号付け(num-lvl)

階層的な番号付けは[デザインにおける目次レベル]内に挿入されます。従ってドキュメント内部に階層的な番号を作成するには、ドキュメントの構造を目次レベルに作成する必要があります。[レベル内部のデザイン構造]セクションにおける記述を参照ください。以下の点に留意してください:

- 番号付けされるノード上、またはその内部にレベルを作成する必要があります。

- 番号の階層構造に従うかたちでレベルのネスト化を行う必要があります（以下のスクリーンショットを参照）。
- デザイン上の対応するレベル内部に階層的な番号プレースホルダを挿入する必要があります（以下のスクリーンショットを参照）。



上のスクリーンショットには、2つのレベルが示されています。topic 要素は再帰的に定義されており、2つのtopic 要素に対してレベルが作成されています（ノードタグを右クリックして、**レベルとして機能するテンプレートを選択**します）。上のスクリーンショットでハイライトされているtopic 要素は、別のtopic 要素内にネストされています。これにより、レベルも同様にネストされます。各レベル内には階層的な番号プレースホルダの（num-lvl）が挿入されています（レベル内部を右クリックして、「**目次を挿入 | 階層的な番号付け**」を選択します）。

上のスクリーンショットにあるデザインの出力結果を以下に示します:

1: Altova StyleVision 2007
2: About this Documentation
3: Introduction
3.1: What Is an SPS?
3.2: Product Features
3.3: Setting up StyleVision
4: User Interface
4.1: Main Window
4.2: Design Entry Helpers

最初のレベルが太字により、2番目のレベルが標準文字により表示されます。

階層的な番号のフォーマットを指定するには、（num-lvl）プレースホルダを右クリックして、**フォーマットの編集**コマンドを選択します。階層的自動番号割り当てダイアログが表示されます（以下のスクリーンショットを参照）。

まずはトークンの数をコンボボックスにて選択します。トークンの数は、ドキュメントにおける目次レベルの数に合わせるべきでしょう。各トークンは個別にフォーマットすることができます。ダイアログの中央左にあるリストからフォーマットするトークンを選択してください（上のスクリーンショットでは、2番目のトークンが選択されています）。フォーマットコンボボックスから、目的のフォーマットスタイルを選択します。上のスクリーンショットでは、2番目のトークンに対して小文字のフォーマットが選択されており、それがダイアログ上部にあるテキストボックスの表示に反映されています。更に、レベルの省略ボックスにて番号を選択することで、表示する際に省略するレベルを指定することもできます。

階層的な番号のフォーマット定義は、個々のレベルに対して個別に行われるという点に注意してください。従って、各レベルにある（num-lvl）プレースホルダに対して個別にフォーマットの指定を行う必要があります。

OK をクリックすることで完了します。

関連項目

- 目次（TOC）
- レベル内部のデザイン構造
- 目次ブックマークの作成
- 目次テンプレートの作成

7.7.6 相互参照

相互参照とはドキュメント内における別の箇所に対する参照のことです。SPS では相互参照のターゲットと、そのターゲットに対するリンクという2つのパーツにより相互参照が構築されます。ターゲットをセットするには、目次レベル内部に目次ブックマークを作成します。目次参照内にあるテキスト参照がターゲットへのリンクとなります。テキスト参照によりリンクとなるテキストの出力が生成されます。相互参照は以下にある3つのステップにより構築されることになります。

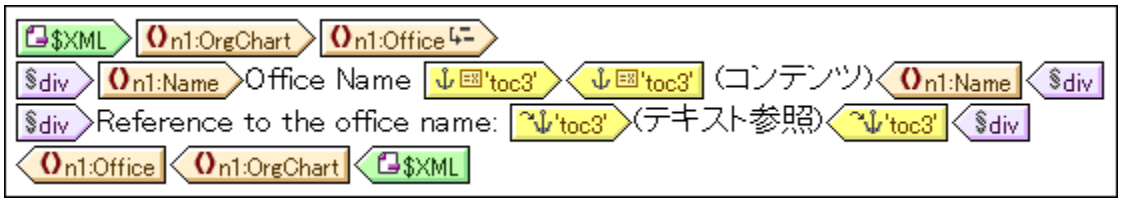
ステップ 1: レベル

レベル内部のデザイン構造のセクションに記されているように、ドキュメントは目次レベルにより分けられます。参照時には目次レベルにより参照のスコープが指定されます。指定されたスコープ内にて指定された名前を持っている目次ブックマークがターゲットとなります。以下のスクリーンショットでは、n1:Office 要素にてレベルが作成されています。

ステップ 2: 目次ブックマークの作成

目的の場所にカーソルを配置して、右クリックにて表示されるコンテキストメニューから「**目次を挿入 | 目次ブックマーク**」を選択することで、目次ブックマークがレベル内部に作成されます。目次ブックマークにはXPath 条件式が与えられており、それにより出力テキストが生成されます。XPath 条件式により、通常ドキュメント内にあるノードが識別され、そのノードのコンテンツが目的のテキストとして使用されます。

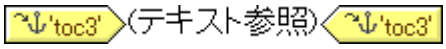
以下にあるスクリーンショットでは、n1:Name 要素内にある目次ブックマーク 'toc3' 'toc3' にtoc3 という名前と、カレントノードを選択するXPath 条件式が与えられています。これにより、n1:Name ノードのコンテンツが出力テキストとして使用されます。



XML ドキュメントが処理されると、全てのn1:Name 要素に対してアンカーが作成されます。このアンカーでは、テキスト参照(相互参照のテキスト)にn1:Name 要素の値が使用されます。

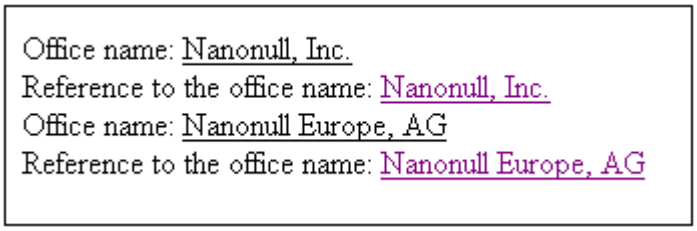
ステップ 3: 目次参照の作成

目次ブックマークにより生成されたアンカーへのリンクを作成するために、(コンテキストメニューの「**目次を挿入 | 目次参照**」により)目次参照が挿入されます。



上のスクリーンショットでは、toc3 という名前の目次参照が、参照先の目次ブックマークと同じ目次レベル(Office レベル)内に配置されています。ここでは目次参照のスコープも指定する必要があります。スコープにより、目次参照に対して与えられた名前と同名の目次ブックマークを検索する目次レベルが指定されることになります。上の例ではスコープに現在のレベルがセットされており、現在のレベルにあるtoc3 という名前の目次ブックマークだけが、参照のターゲットとなります。

上のスクリーンショットでは、n1:Office テンプレートが表示されています。n1:Office ノードが処理されると、n1:Name ノードのコンテンツを出力テキストとするアンカーが作成されます。これは、(目次ブックマークの**使用されるテキスト**プロパティからアクセスすることのできる)目次ブックマークのXPath 条件式により、このノードのコンテンツが出力テキストと指定されているためです。次の行にある目次参照によりtoc3 という名前を持ったアンカーが識別され、テキスト参照コンポーネントによりリンクのテキストが生成されます(以下のスクリーンショットにて紫色で示されているテキスト)。出力は以下のようになります:



上に示された例では、現在のレベルにスコープがセットされています。スコープとして指定することのできるオプションは、この他にも(i) グローバルスコープと(ii) 現在と下位のレベルがあり、これらのオプションを使用することで、デザイン上のその他のレベルにある目次ブックマークをターゲットとすることもできます。

関連項目

- 目次（TOC）
- レベル内部のデザイン構造
- 目次ブックマークの作成
- 目次テンプレートの作成

7.7.7 ブックマーク と ハイパーリンク

SPS ドキュメントでは、デザイン内の任意の場所にブックマークを挿入することができます。これらのブックマークは出力にてアンカーに変換され、ハイパーリンクの参照先として使用することができます。ハイパーリンクはブックマークだけではなく、ウェブページのような外部リソースへの参照を行うこともできます。StyleVision では柔軟なかたちでハイパーリンクに使用されるターゲット URI 構築することができます。

このセクションでは、we describe:

- SPS にて、ブックマーク を挿入する方法。.
- SPS にて、ハイパーリンク を挿入し、ターゲットページへのリンクを作成する方法。

メモ 外部ドキュメントへのリンクは FO の仕様にてサポートされていますが、お使いの FO プロセッサーによりサポートされていないこともあります。FO にてこの機能を使用する場合、使用するプロセッサーにてこの機能がサポートされているか確認してください。

関連項目

- パースされていないエンティティ URI

7.7.7.1 ブックマークの挿入

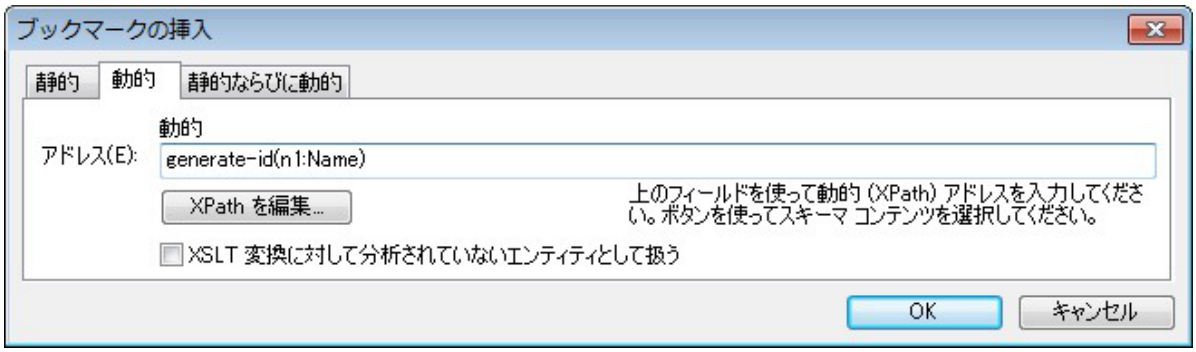
ブックマーク（ハイパーリンク）は、カーソルの配置位置、または SPS コンポーネントの周りと、SPS における任意の場所にて挿入することができます。

ブックマークの挿入ダイアログを使用することで、ブックマークを SPS 内部に作成することができます（以下のスクリーンショットを参照）。このダイアログでは静的または動的なブックマークの名前を定義することができます。動的な名前とは、(i) XML ドキュメントのコンテンツから得られた値、または (ii) XPath 条件式により生成された値のことです。

ブックマークの作成

以下の操作によりブックマークを挿入することができます:

1. ブックマークを作成する箇所にカーソルを配置します。
2. メニューオプションから「 挿入 | ブックマークの挿入」を選択するか、右クリックにより表示されるコンテキストメニューから「**ブックマークを挿入**」を選択します。
3. ブックマークの挿入ダイアログが表示されます（以下のスクリーンショットを参照）。作成されるブックマークが静的なテキストの場合は静的タブを、名前が XML ドキュメントや XPath 条件式から動的に取得される場合は動的タブを、静的ならびに動的なパーツから構成される場合には静的ならびに動的タブを選択してください。以下のスクリーンショットでは、コンテキストノード以下の Name 子要素に対してユニークな名前を持った動的なブックマークが作成されます。



4. 「OK」をクリックすることでブックマークが定義されます。

作成されたブックマークはハイパーリンクから参照することができます。

メモ　デザインにて指定された場所にブックマークは作成されます。繰り返し出現する要素内部が指定された場合、繰り返し出現する各インスタンス要素内部にブックマークが作成されます。静的な名前が与えられた場合、これらブックマーク全てに同じ名前が与えられます。そのため繰り返し出現する要素に対しては、例えばコンテキストノード（ブックマークが作成されたノード）の子要素にあるコンテンツを選択するような動的な名前を指定することが望まれます。動的な方法により選択される複数のインスタンスノードに同じコンテンツが含まれる場合、generated-id() 関数により生成される名前により、ブックマーク名の一意性を保証することができます（以下のスクリーンショットを参照）。ブックマークの参照を行うには、同じ ID をハイパーリンクのhref 値として生成することができます。この場合、generate-id() 関数の前にフラグメント識別子の# を使用するようにしてください。XPath 条件式はconcat('#', generate-id(nodeXXX)) となります。

ブックマークの修正

作成されたブックマークの名前は、ブックマークの編集ダイアログにて修正することができます。以下の操作によりダイアログへアクセスすることができます:

1. デザインにあるブックマークを選択します。
2. プロパティサイドバーのブックマークグループにあるブックマーク名プロパティの編集ボタンをクリックします（以下のスクリーンショットを参照）。上記のブックマークの挿入ダイアログと同じ機能を持ったブックマークの編集ダイアログが表示されます（上のスクリーンショットを参照）。

3. ブックマークの編集ダイアログにて静的、動的、または静的ならびに動的タブにてブックマークの名前を編集してください。

ブックマークの削除

ブックマークを削除するには、デザイン上のブックマークが選択された状態で、Delete キーを押下してください。

関連項目

- ハイパーリンクの定義 : ブックマークへのハイパーリンクへの作成方法について記述されます。
- 挿入 | ブックマーク : ブックマークの挿入について記述されます。

7.7.7.2 ハイパーリンクの定義

ハイパーリンクはテキストやイメージといったSPS コンポーネントの周りに作成されます。ハイパーリンクのターゲットには(i) SPS デザイン内部にあるブックマーク、または(ii) ウェブページや電子メールのメッセージといった外部リソースを使用することができます。このセクションではハイパーリンクのコンテンツ(テキストやイメージなど)について、そしてハイパーリンクのターゲットについて説明します。

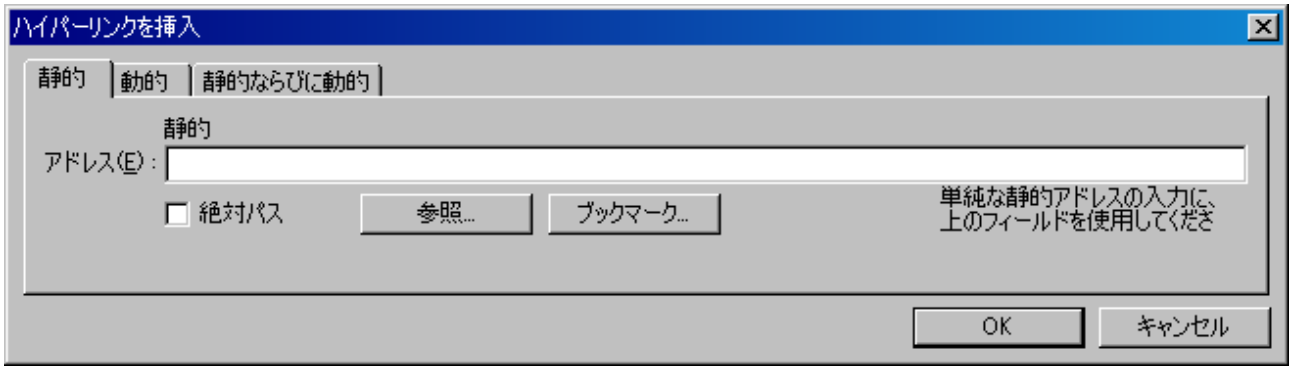
ハイパーリンクの作成

以下の操作によりハイパーリンクを作成することができます:

- (静的または動的な)テキスト、ノード、イメージ、条件テンプレート、自動計算、コンテンツやノードのブロックの周りで: 入力フィールドやコンボボックスといったデータ入力デバイスの周りにハイパーリンクを作成することはできませんが、データ入力デバイスを含んでいるノードや条件テンプレートの周りに作成することはできます。囲まれた対象が、リンクのコンテンツとなります。コンポーネントの周りにハイパーリンクを作成するには、そのコンポーネントが選択されている状態で、メニューオプションから「**タグで囲む | ハイパーリンク**」を選択してください。
- メニューオプションの「**挿入 | ハイパーリンクを挿入**」によりハイパーリンクを挿入: 新たに作成されたハイパーリンクのタグ内部に、リンクのコンテンツを追加する必要があります。

ハイパーリンクのターゲットを定義

タグで囲む | ハイパーリンク または 挿入 | ハイパーリンク によりアクセスすることができるハイパーリンクの挿入ダイアログではハイパーリンクのターゲットを作成することができます(以下のスクリーンショットを参照)。



リンクのターゲットは以下のどれかになります:

- 同一 SPS デザインにある ブックマーク: ターゲット URI はフラグメント識別子となります。
- ブックマークのアンカーに対して動的に作成されたもの : これらのURI もフラグメント識別子となります。
- 外部リソース: URI は(直接入力された)静的なもの、(XML ドキュメント内のノードから得られる)動的なもの、静的ならびに動的なパーツから構成されるもの、またはパースされていないエンティティのどれかとなります。

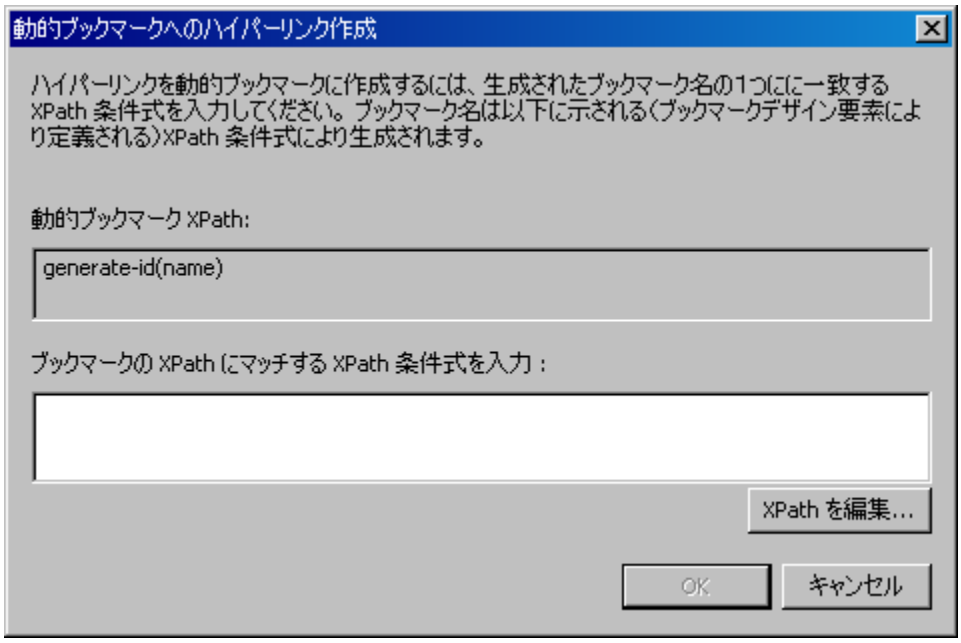
以下ではこれらのターゲットを定義する方法について説明します。ハイパーリンクの挿入/編集ダイアログにて URI を定義して、「OK」をクリックして完了します。

ブックマークへのリンク作成

以下の操作によりブックマークへのリンクを作成することができます:

1. ハイパーリンクの挿入ダイアログにある静的タブにて**ブックマーク**ボタンをクリックします。この操作によりドキュメント内のブックマークを選択するダイアログが表示されます(以下のスクリーンショットを参照)。以下に示されるスクリーンショットには、静的なブックマークが1つ、動的なブックマークが1つ表示されています。

2. 静的なブックマークをターゲット URI として選択するには、静的なブックマークをダブルクリックして「OK」をクリックします。動的なブックマークをダブルクリックした場合、選択された動的なブックマークにマッチする XPath 条件式を入力するように促されます(以下のスクリーンショットを参照)。



動的なブックマーク の実体は、(それ自身がブックマークの名前でない場合)ブックマークの名前を生成する XPath 条件式となります。動的ブックマークへのハイパーリンク作成ダイアログには動的なブックマークの XPath 条件式が表示されており、ターゲットとなるブックマークにマッチする名前を生成する XPath 条件式を構築することができます。「OK」をクリックして完了します。

動的に作成されたID ブックマークへのリンク作成

ブックマークには、動的に生成されたID アンカー を含めることができます。このようなブックマークへのリンクを作成する場合、アンカーはランタイムにて動的に生成されるため、このようなアンカーに対して ハイパーリンク の href 値をどのようにセットするかが問題になります。動的に生成されるアンカーに対しては generate-id() 関数を、ハイパーリンクの href 値の内部で使用することになります。変換が一度だけ行われた場合、generate-id() 関数が同一のノードに対して評価されると、常に同一のID が生成されます。これにより、ブックマークとハイパーリンクにて生成されたID が同一のものとなります。

以下の2点に留意する必要があります:

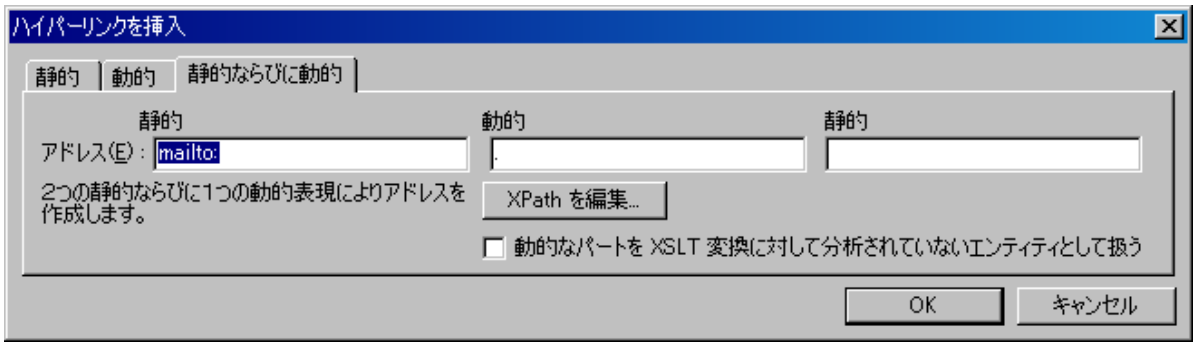
- generate-id() 関数は XPath 条件式として評価する必要があるため、ハイパーリンクのターゲットをセットするにはハイパーリンクの挿入ダイアログにある動的タブを選択する必要があります。
- href 属性で評価される値は(フラグメント識別子を表す) # からはじめる必要があります。従って XPath 条件式は concat('#', generate-id(nodeXXX)) のような形式となります。静的ならびに動的タブにて、# をアドレスの静的パートに入力し、generate-id(nodeXXX) を動的パートに入力することもできます。

外部リソースへのリンク作成

以下の操作により、外部リソースを選択するための URI を構築することができます:

- ハイパーリンクの挿入タブにある静的タブにて URI を直接入力する: 例えば静的タブのアドレス入力フィールドに対して Altova ウェブページ (http://www.altova.com/) へのリンクを直接入力することができます。
- ハイパーリンクの挿入ダイアログにある動的タブの XML ドキュメントソースにてノードを選択する: XML ソース内部にあるノードから、(i) ターゲットとなる URI、または (ii) 目的の URI を値として含んでいる パースされていないエンティティ 名のどれかを表すテキスト文字列を与えることになります。例えば Altova ウェブサイトのアドレスを、ノード内にあるテキスト文字列として含めることができます。
- ハイパーリンクの挿入ダイアログにある静的ならびに動的タブにて、静的ならびに動的なパーツから構成される URI を構築する: この機能は静的な(例えばプロトコルを表す)接頭語や(例えばドメイン名を表す)接尾語を追加する際に使用することができます。例えば mailto: という静的なパーツと //Contact/@email ノ

ードのコンテンツから得られた文字列を組み合わせることにより、電子メールアドレスを構築します（以下のスクリーンショットでは、//Contact/@email ノードのコンテンツプレースホルダへのリンクが作成されており、自分自身のノードを表す self::node() の短縮形 . が使用されています）。



パースされていないエンティティを使用する方法については、パースされていないエンティティ URI のセクションを参照ください。

メモ 外部ドキュメントへのリンクは FO の仕様にてサポートされていますが、お使いの FOP によっては、作成された PDF ドキュメントにて外部へのリンクが動作しない場合があります。

Authentic View における親ノードにあるコンテンツの修正

ハイパーリンクの親ノードにある値は Authentic View ユーザーにより選択することができ、Authentic View ユーザーの選択により表示が変更されるような SPS をデザインすることができます。

ハイパーリンクプロパティの編集

ハイパーリンクを編集するには ハイパーリンクの開始または終了タグを右クリックして、コンテキストメニューから URI の編集を選択します。ハイパーリンクの編集ダイアログが表示されます（上のスクリーンショットを参照）。プロパティウィンドウの ハイパーリンクグループ内にある URL プロパティからも、ハイパーリンクの編集ダイアログへアクセスすることができます。

ハイパーリンクの削除

ハイパーリンクを削除するには、ハイパーリンクの開始または終了タグをクリックして ハイパーリンクを選択し、Delete キーを押下します。ハイパーリンクとそのコンテンツが削除されます。

関連項目

- ブックマークの挿入
- パースされていないエンティティ URI
- 挿入 | ハイパーリンク

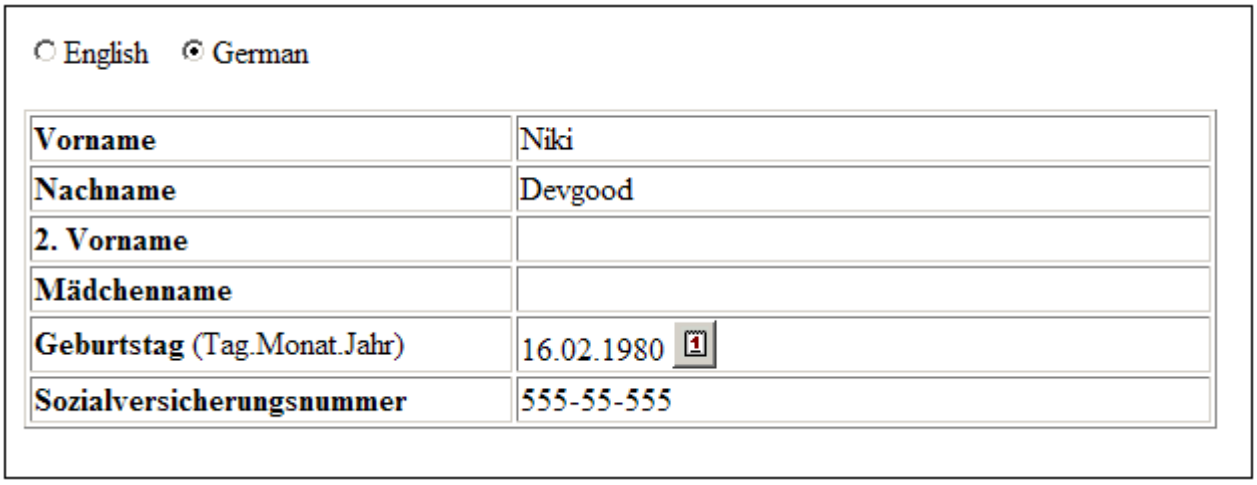
7.8 サンプル: 複数の言語の使用

ドキュメント や Authentic Form には複数の言語を含めたり、ユーザーが使用する言語を選択できるようにしておかなければならないことが数多くあります。StyleVision では、このような機能を実装するための方法が用意されています。以下ではどのようなことが可能なのかが記述されており、StyleVision に同封されている Examples プロジェクトの Multiple Language サンプルにて確認することができます。
Examples プロジェクトは、初めて StyleVision を起動した時に自動的にロードされるデフォルトのプロジェクトで、メニューオプションの「**プロジェクト | プロジェクトを開く**」を選択し、`C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples` フォルダー内にある Examples.svp を選択することでも開くことができます。

変数と条件を使用する

ユーザーが使用する言語を 編集可能な変数に格納します。複数の分岐を持った条件により、選択された言語のコンテンツが表示されます。ユーザーが選択した内容により、対応する条件分岐が選択されます。

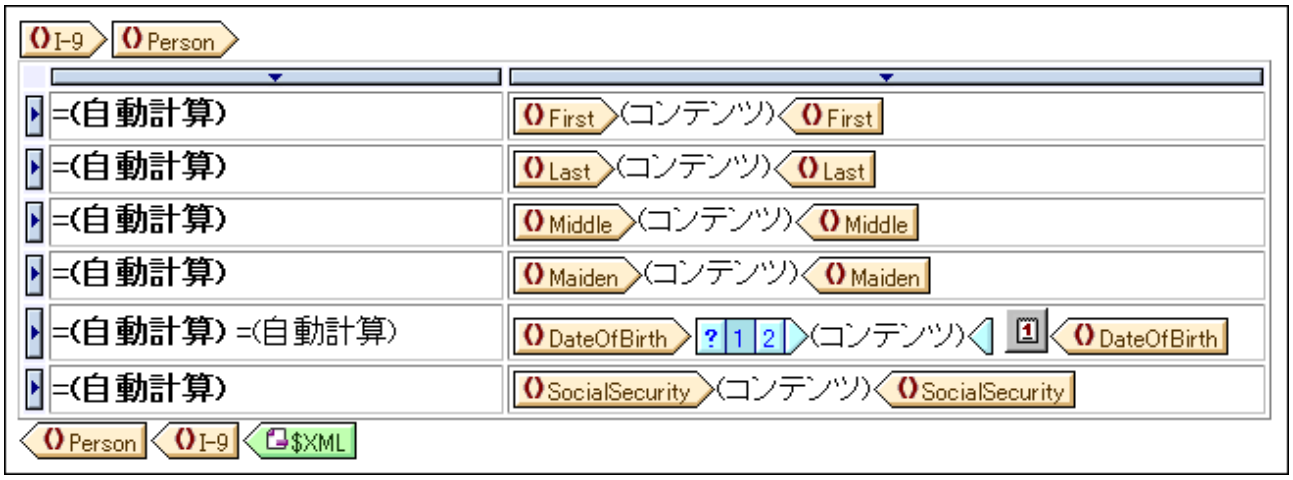
上のスクリーンショットにて、ユーザーの選択が編集可能な変数として入力されている点に注目してください。テーブル内部の条件には2つの分岐が与えられており、変数の値に従った言語の選択が行われます。Authentic View の出力を以下のスクリーンショットにて示します。



上の方法は、ユーザーが使用する言語を選択するような場合に適したものとなります。詳細については、Examples プロジェクトの MultiLangByCondition.sps を参照ください。

パラメータと自動計算の使用

別の例として同じデータを別の言語で使用する状況を考えてみましょう。このような状況ではパラメータの値を使用することで、異なる言語の出力を生成することができます。出力を行う言語の決定は、自動計算などを使うことにより行うことができます。自動計算により、パラメーターの値に従った適切なコンテンツを出力することができます。



上のスクリーンショットでは、以下の様な XPath 条件式が自動計算に使用されています。

```
if ( $Language = 'E' ) then 'First' else
if ( $Language = 'G' ) then 'Vorname' else ''
```

SPS デザイン内にあるグローバルパラメーターの $Language 値を修正するか、ランタイムにてコマンドラインから与えられた値を使用することになります。変換ごとに、同じデータを使った異なる言語での出力を行うことができます。

詳細については Examples プロジェクトの MultiLangByAutoCalc.sps を参照ください。

サンプルファイル

C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples フォルダー内に収められている Examples プロジェクトファイル(Examples.svp) を開くことで、その他のサンプルを確認することができます。

(C) 2015-2021 Altova GmbH

8 表示方法と出力

SPS デザインでは、同一のスタイル機能がコンポーネントに対して定義されます。これらのスタイル情報は、それぞれの出力にて対応するスタイルマークアップへ変換されます。(*Enterprise Edition の Authentic View HTML*、*RTF*、*PDF* および *Word 2007+*、*Professional Edition の Authentic View*、*Basic Edition の HTML と RTF*)表示エフェクト、特にウェブ形式におけるインタラクティブなエフェクト(コンボボックスや JavaScript のイベントハンドラーなど)によっては、(RTF のような)印刷媒体の出力では利用できません。その場合、印刷媒体の出力ではエフェクトが印刷可能な形式で表示されます。印刷媒体の出力に対して、StyleVision では、ページ定義 オプションを使用することができます。ページサイズやページのレイアウト、ヘッダーやフッターといった印刷媒体 オプション コンポーネントのスタイルに対して定義され、RTF の出力のみに使用されます。

SPS コンポーネントのスタイル

SPS コンポーネントの全スタイルにはCSS2 の構文が使用されます。スタイルは外部のスタイルシートにて、SPS 全体に対してグローバルに、そして特定のコンポーネントに対してローカルに定義することができます。CSS2 のカスケード順位がSPS に対して適用されることで、スタイルのデザインにおける柔軟性が飛躍的に向上します。CSS スタイルの使用方法については、このセクションのサブセクションである CSS スタイルと作業 にて記述されます。

スタイルプロパティの値は、スタイルまたはプロパティサイドバーにて直接入力することができるほか、XPath 式を介してセットすることもできます。XPath 条件式を使用することで、(i) プロパティの値を XML ファイルから取得し、(ii) XPath 条件式で記述された条件に従ったかたちでプロパティの値を割り当てることができるようになります。

その他にも、SPS デザインでは特定のHTML 要素をSPS コンポーネントのマークアップとして使用することができます。これら定義済みのフォーマット はHTML 出力に渡されます。このようなマークアップは、SPS コンポーネントのスタイルとしても使用することができます。CSS スタイルが定義済みのフォーマットに対して適用されると、CSS スタイルが定義済みフォーマットのスタイルよりも高い優先度で処理されます。定義済みのフォーマットについては、このセクションのサブセクションである 定義済みのフォーマット にて記述されます。定義済みフォーマットから継承されたスタイルは、RTF 出力にて同等のマークアップへ変換されます。

メモ SPS コンポーネントに対して CSS スタイルを定義する場合、これらスタイルが(RTF の)印刷媒体の出力に対して必ずしも適用されるわけではないという点に留意してください。更に、HTML セレクターが(外部スタイルシートならびにグローバルスタイルルールにて)使用された場合、これらのセレクターが (RTF の)印刷媒体の出力に対して適用されることはありません。このようなセレクターが使用された場合、セレクターの隣にコメントが表示され、RTF 出力に対してそのスタイルが適用されないことが示されます。

印刷媒体の出力をデザインする

StyleVision にて印刷媒体 (RTF の出力やRTF) に対応したXSLT スタイルシートを生成する場合、ページの定義とレイアウト オプション を使用することができます。これらのオプションはコンポーネントのスタイルメカニズムとは別に使用され、このセクションのサブセクションである 印刷出力のデザイン にて記述されます。

関連項目

- 使用方法の概要
- デザインサイドバー

8.1 定義済みのフォーマット

StyleVision では、HTML 要素に対応する数々の定義済みフォーマットを使用することができます（以下のスクリーンショットを参照）。定義済みのフォーマットをデザイン内部にあるコンポーネントに対して適用すると、選択された HTML 上の意味を持つようにコンポーネントがマークアップされます。この操作により2つの効果が期待できます:

- 選択された定義済みのフォーマットに対して、HTML 要素固有のフォーマットが適用されます。
- コンポーネントが段落というコンポーネントタイプに含まれ、コンポーネントタイプに対して ローカルスタイルを定義することが可能になります。

定義済みのフォーマットを割り当てる

メニューオプションの「**挿入 | 特殊な段落を挿入**」から目的のフォーマットを選択するか、ツールバーにあるフォーマットドロップダウンリストから目的のフォーマットを選択することで、定義済みのフォーマットを割り当てることができます。

固有のスタイル

StyleVision にて使用される定義済みフォーマットには、以下にあるスタイルコンポーネントのうちの1つまたは両方が含まれます:

- テキストスタイルコンポーネント
- 余白コンポーネント

例えば、定義済みの段落(p) フォーマットには余白コンポーネントだけが含まれており、選択されたコンポーネントの直前と直後に水平方向の余白が挿入され、テキストのスタイルが適用されることはありません。それとは別に、見出し1(h1) フォーマットにはテキストスタイルコンポーネントと余白コンポーネントの両方が含まれます。

定義済みのフォーマットにおける以下のスタイル情報について留意してください:

- 定義済みフォーマットの余白コンポーネントは全種類のSPS コンポーネントに対して適用されますが、テキストスタイルは適用できる時だけ適用されます。例えば、イメージを選択して 見出し1(h1) の定義済みフォーマットを適用すると余白コンポーネントが適用されるのに対して、テキストスタイルコンポーネントは適用されません。
- 定義済みフォーマットのテキストスタイルコンポーネントはデータ入力デバイスに対して適用されません。
- 同一のコンポーネントに対して適用される定義済みフォーマットは1つだけです。
- 整形済みテキスト(pre) 定義済みフォーマットでは、HTML のpre タブを適用した時と同様のフォーマット結果を得ることができます。テキスト内にある改行ならびに空白スペースはそのまま保持され、表示には(Courier のような) 等角フォントが使用されます。テキストにより構成される段落のように、改行を持たない行の場合、整形済みテキスト(pre) 定義済みフォーマットによりテキストの折り返しが行われることはありません。テキストの折り返しを行う場合、整形済みテキスト(pre wrap) を使用してください。

定義済みのフォーマットのための追加スタイルを定義する

定義済みのフォーマットを選択して スタイルサイドバーからローカルスタイルを適用することで、固有のスタイルとは別の定義を指定することができます。

改行キーと定義済みのフォーマット

Authentic View にて定義済みのフォーマットが適用された要素のコンテンツ内で改行キーが押下されると、その要素のインスタンスとブロックの処理が完了し、その場所に新たな要素インスタンスとブロックが挿入されます。例えば改行キーを押下することで Authentic View ユーザーが新たな段落型の要素を作成するような場合に、この機能を利用することができます。

関連項目

- CSS スタイルをローカルに定義する

8.2 出力のエスケープ

テキスト文字列内にある文字が文字参照またはエンティティ参照として記述されている場合、その文字がエスケープされた状態となります。参照は(文字参照かエンティティ参照かに関わらず)アンパサンド (&) 文字で始まりセミコロン (;) にて終わります。例えば.

- A という文字に対する16進数(または Unicode)の文字参照は `A` となります。
- A という文字に対する10進数の文字参照は `A` となります。
- & という文字に対する HTML(ならびに XML)エンティティ参照は `&` となります。
- & という文字に対する16進数(または Unicode)の文字参照は `&` となります。
- & という文字に対する10進数の文字参照は `&` となります。
- < という文字に対する HTML(ならびに XML)エンティティ参照は `<` となります。

出力のエスケープ

出力のエスケープとは入力にてエスケープされた文字を出力にて表示することを指し、出力にて文字参照またはエンティティ参照として表示される文字のことを出力エスケープされた文字といいます。入力にてエスケープされた文字だけが出力エスケープできるという点に注意してください(以下のテーブルにある例を参照ください)。SPS では以下のものに対して出力エスケープを有効にすることができます:

- 静的テキストのフラグメント
- `contents` プレースホルダ
- 自動計算

プロパティのテキストグループにある disable-output-escaping 属性にて設定を行うことができます。プロパティのデフォルト値は「いいえ」になっており、出力のエスケープが行われることは無く、入力にてエスケープされた文字はデフォルトでエスケープされます(以下のテーブルにある例を参照ください)。

以下の操作により出力のエスケープを無効化することができます:

1. 出力のエスケープの無効化を行う (i) 静的テキスト、または (ii) 静的テキストのフラグメント、(iii) コンテンツプレースホルダ、または (iv) 自動計算を選択します。
2. プロパティサイドバーにて、テキストアイテムのテキストグループを選択し、disable-output-escaping 属性の値として表示される個々の出力または全ての出力を選択します。選択可能な値を以下に示します:

- HTML に: HTML 出力に対して disable-output-escaping の値が「はい」になります。
- Authentic に: Authentic 出力に対して disable-output-escaping の値が「はい」になります。Authentic ビューに対する出力エスケープを無効にする機能は、**Enterprise エディションの Authentic ビュー**(つまり Enterprise エディションのStyleVision、Authentic Desktop、Authentic Browser、XMLSpy)でしか使用できない点に注意してください。.
- RTF に: RTF 出力に対して disable-output-escaping の値が「はい」になります。
- 全てに: 全ての出力に対して disable-output-escaping の値が「はい」になります。

特定の出力(例えばHTML 出力)に対して出力エスケープが無効になっている場合、選択されたテキストはその出力フォーマットにてエスケープされず、その他の出力フォーマットにてエスケープされます。

出力エスケープが無効/有効になっている状態で、テキストがどのように表示されるかを以下のテーブルに示します。

静的テキスト	disable-output-escaping	出力テキスト
`&`	いいえ	`&`
`&`	はい	`&`
`&`	いいえ	`&`

&	はい	&
<	いいえ	<
<	はい	<
A	いいえ	A
A	はい	A
<	いいえ	<
<	はい	<
<	はい	<
&<	はい	&<

メモ disable-output-escaping は、Enterprise エディションの Authentic View でのみサポートされます。

出力フォーマットごとに出力エスケープを無効化する

出力エスケープを無効化したとしても、ある出力にとって意味のあるテキストが別の出力では意味を持たない場合も考えられます。例えば、(ハイライトされた)エスケープ文字が存在する、以下の入力テキストを考えてみましょう:

```
&lt;b&gt;This text is bold.&lt;/b&gt;
```

出力エスケープが無効化されている場合、出力は以下のようになります:

```
<b>This text is bold.</b>
```

HTML 出力に対して出力エスケープが無効化されている状態でこの出力がブラウザーにて閲覧されると、文字列のマークアップがHTML ブラウザーにより処理され、太字でテキストが表示されるようになります:

This text is bold.

しかし、その他の出力フォーマット(例えばPDF)で同じ文字列が出力されても、HTML 出力では意味のあるマークアップが、その他のフォーマットでも意味を持つとは限りません。上記にある例の場合、(出力エスケープを無効にすることで得られる)エスケープされていないテキストは、PDF フォーマットで以下のように表示されます:

```
<b>This text is bold.</b>
```

上記の例で見られるように、出力エスケープを無効にすることで得られる出力テキストは、ある出力フォーマットでコードとして読み取られる一方で、その他の出力では通常のテキストとして読み取られます。disable-output-escaping プロパティを使用する際には、この違いについて常に留意する必要があります。

関連項目

- テキストとしてXML コンテンツを挿入
- 自動計算
- デザインビューシンボル

8.3 値フォーマット（数値データ型のフォーマット）

値フォーマットを使用することで、数値型のXML スキーマデータ型ノード(*下のリスト*)に収められたコンテンツを、元のデータ型を使って表示とは違うフォーマットで表示することができます。例えば、xs:date データ型のノードはYYYY-MM-DD に、+02:00 といったオプションのタイムゾーンコンポーネントが付随した形式になります。値のフォーマットは Authentic View にて表示されるほか、フォーマットの定義によってはHTML とRTF 出力にて表示することもできます。自動計算の結果が数値型のデータ型の表記にマッチする場合、自動計算の結果に対して値のフォーマットを使用することもできます(*下のリスト*) 。

このセクションのサブセクションでは、以下の項目について記述されます:

- 値フォーマットメカニズムの動作方法
- 値フォーマットを定義するための構文

メモ 値フォーマットにより XML ドキュメントに収められているデータの形式が変更されることはありません。妥当な XML ドキュメントでは、そのノードのデータ型に対して適切な形式でデータが常に収められます。値フォーマットは、 Authentic View における表示に対して適用されるほか、オプションとして 出力の表示を行うために使用されます。

値フォーマットが使用できる数値型のデータ型

以下のデータ型に対して値フォーマットを使用することができます:

- xs:decimal ならびに xs:integer: xs:integer から派生した12の内蔵型
- xs:double ならびに xs:float: 値の最小値は 0.000001、最大値は 1,000,000 となります。この範囲に入らない値は（例えば 1.0E7 といった）別の形式で表示され、値のフォーマットを適用することはできません。
- xs:date、xs:dateTime、xs:duration
- xs:gYear、xs:gYearMonth、xs:gMonth、xs:gMonthDay、xs:gDay

関連項目

- 日付のフォーマット

8.3.1 値フォーマットのメカニズム

値フォーマットは以下のアイテムに対して適用することができます:

- SPS にてコンテンツならびに入力フィールドとして表示されている xs:decimal や xs:date といった 数値型のデータ型ノード
- 数値型のデータ型フォーマットの形式に収まる自動計算の結果

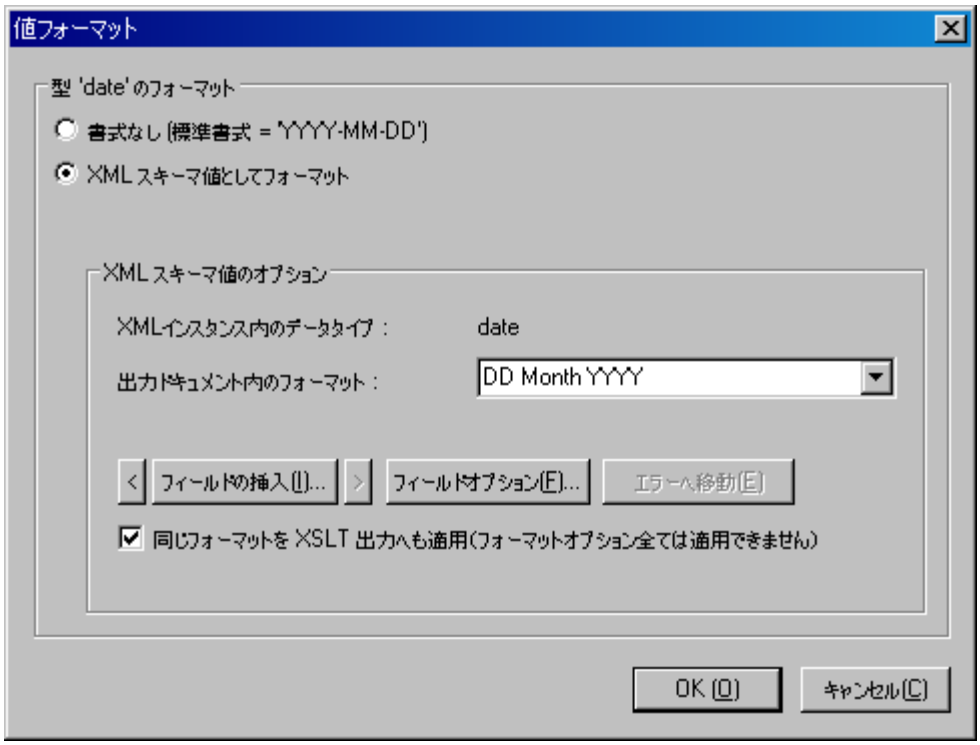
値フォーマットの定義

以下の操作によりSPS に対してノードまたは自動計算を定義することができます:

1. コンテンツプレースホルダやノードの入力フィールド、または自動計算を選択します。

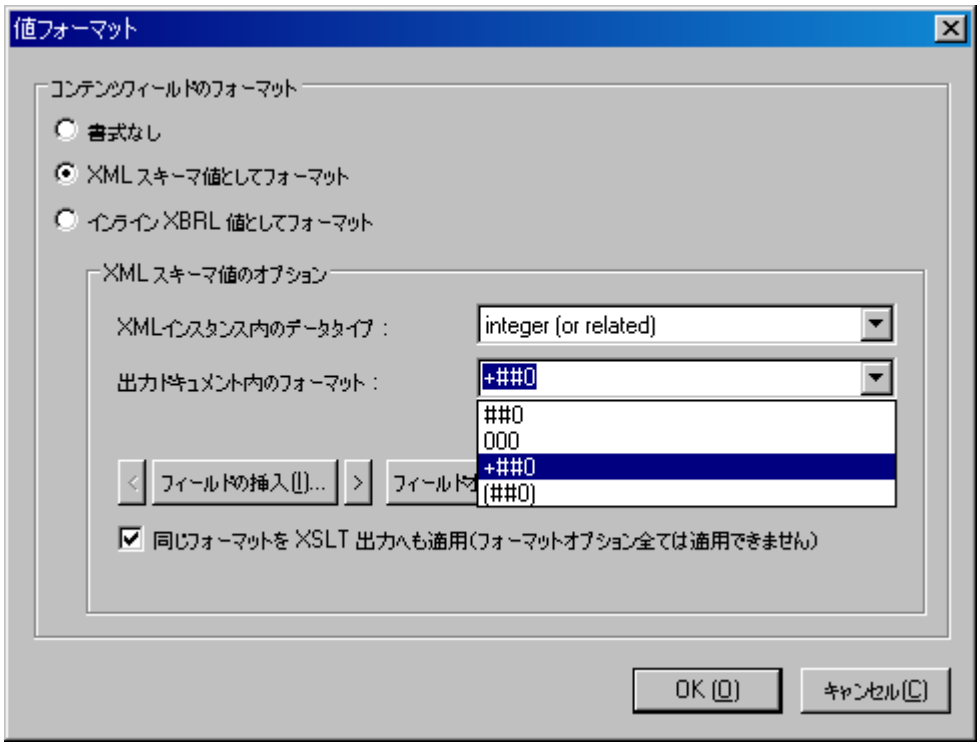
(C) 2015-2021 Altova GmbH

2. プロパティサイドバーにて自動計算グループにある値フォーマットのアイテムを選択します。値フォーマットプロパティにある編集ボタン [...] をクリックします。コンポーネントを右クリックして、コンテキストメニューから**値フォーマットの編集**を選択します。値フォーマットのダイアログが表示されます。選択されたコンポーネントがノードまたは自動計算であるかによりダイアログの表示が変化します。以下のスクリーンショットで示されるスクリーンショットは、xs:date データ型のノードからダイアログを表示したものです。



上のスクリーンショットにて示されるダイアログには、型 'date' のフォーマットが含まれており、xs:date データ型に対する標準的なフォーマットが書式なしのラジオボタンにも表示されています。その他のデータ型のノードに対して、この記述内容は変化します。

選択されたコンポーネントが自動計算の場合、ダイアログは以下の様に表示されます。



3. コンポーネントの値がフォーマットされたかたちで表示するのかを指定することができます。出力をフォーマットされないかたちで表示するには、**書式なし**ラジオボタンを選択してください。**XML スキーマ値としてフォーマット**ラジオボタンを選択することで、値のフォーマットを行うことができます。値のフォーマットを行わない場合、選択されたノードや自動計算結果のデータ型に対して標準的なフォーマットが出力では適用されます。自動計算に対して**XML スキーマ値としてフォーマット**を選択した場合、（ドロップダウンリストから）期待されている自動計算結果のデータ型を選択する必要があります。
4. 値フォーマットの定義を入力します。定義は、(i) ドロップダウンリストから、そのデータ型に対して使用できるオプションを選択する（上のスクリーンショットにある**出力ドキュメント内のフォーマット**を参照ください）、(ii) 入力フィールド内に定義を直接入力する、(iii) **フィールドの挿入**ならびに**フィールドオプション**を使用することで、適切な定義を構築することができます。フォーマットのオプションについては、値フォーマット構文を参照ください。

構文のエラー

構文内にエラーがある場合、以下のような振る舞いが行われます:

- 定義が赤色で表示されます。
- 赤色のエラーメッセージが入力フィールド以下に表示されます。
- 値フォーマットダイアログの「OK」ボタンが無効になります。
- 値フォーマットダイアログの**エラーへ移動**ボタンが有効になります。ボタンをクリックすると、フォーマット定義にて構文エラーが発生した箇所にカーソルが配置されます。

データとデータ型フォーマットのミスマッチ

XML ノードに入力されたデータがそのノードのデータ型の形式にマッチしない、または自動計算の結果が期待されたデータ型の形式にマッチしない場合、フォーマットは未定義となり、正しいかたちで出力で表示されません。

値フォーマットを出力へ適用する

定義した値フォーマットは、Enterprise ならびにProfessional エディションでサポートされる Authentic View に適用されます。

全てではないものの、値フォーマットの定義によっては、HTML o出力へ適用できるものもあります。その他の出力へも値フォーマットを適用するには、同じフォーマットをXMLSpy 出力へ適用チェックボックスにチェックを入れてください。このオプションがチェックされていない場合、Authentic View にて値フォーマットが使用され、その他の出力ではコンポーネントのデータ型で使用される標準的なフォーマットにて値が表示されます。

関連項目

- 値フォーマット構文
- 日付のフォーマット

8.3.2　値フォーマットの構文

値フォーマットの構文は以下のようになります:

([prefix character/s]field[suffix character/s][{field-オプション1,field-option2,...}])+

ここで　prefix character/s とsuffix character/s は、配置とプラス/マイナス記号を表示するかを指定するオプションの指定子で、
field は任意のデータ型依存のフォーマットまたはテキストとなり、
{field-option(s)} は更なるフォーマットオプションを定義するオプションの修飾子となります。

定義構文の説明

値フォーマットの定義は以下のように構築されます:

- 値フォーマットの定義は1つ以上のフィールドから構成されます。例えば、DD Month YYYY には3つのフィールドが含まれます。
- フィールドは連結して使用することもできるほか、空白スペース、ハイフン、コンマ、コロン、ピリオド、または引用句（' または "）に囲まれたテキスト文字列により区切られたかたちで使用することができます。例えば、DD-Month' in the year 'YYYY という定義では、DD と Month というフィールドがハイフンにより区切られ、Month と YYYY フィールドが、引用句で囲まれたテキストにより区切られています。
- フィールドにはオプションの接頭文字や接尾文字を加えることができます（例：<+###,##0.00）。
- フィールドには必要に応じて使用できるフィールドオプションを含めることができます。各フィールドオプションは角かっこ内に含め、フィールドの直後に配置する必要があります。複数のフィールドオプションを使用する場合、コンマ（","）にて分離する必要があります。例えば、DD Month[uc,ro] YYYY という定義では、uc とro がMonth フィールドに対するフィールドオプションとなります。

例

xs:decimal データ型に対する値フォーマットの例を以下に示します:

"$"(##0.00)

出力は以下のようになります:

$ 25.00
$ 25.42
$267.56

xs:date データ型に対する値フォーマットの例を以下に示します:

DD Month{uc,ro} YYYY

ここで、 uc ならびにro は、それぞれMonth フィールドを大文字の読み取り専用にするためのフィールドオプションとなります。

出力は以下のようになります:

24 SEPTEMBER 2003

フィールドタイプ

フィールドタイプは、データのコンポーネントと、そのコンポーネントがどのように表示されるのかを表します。フィールドタイプにおける固有のフォーマットは、接頭部ならびに接尾部にある修飾子やフィールドオプションにより修正することができます。以下のテーブルには利用可能なフィールドタイプが表示されます。値フォーマットダイアログのドロップダウンメニューには、特定の型に使用されるフィールドと、フィールドだけの値フォーマット定義が含まれます。定義を選択し、接頭部の修飾子、接尾部の修飾子、フィールドオプションを必要に応じて加えることになります。

フィールドタイプ	説明
#	指定された場所に対応する数値が無い場合、空白になります
0	指定された場所に対応する数値がない場合、0 になります
.	小数点
,	数値グループのセパレーター
Y	年
y	年(base = 1900)、下記のメモを参照ください
MM	月、2文字分の長さが必要
DD	日、2文字分の長さが必要、
W	週の番号
d	曜日の番号(1 から7)
i	日付の番号(1 から366)
hh	時(0 から23)、2文字分の長さが必要
HH	時(1 から12)、2文字分の長さが必要
mm	分、2文字分の長さが必要
ss	秒、2文字分の長さが必要
AM	AM またはPM
am	am またはpm
AD	AD またはBC
ad	ad またはbc
CE	CE またはBCE
ce	ce またはbce

(C) 2015-2021 Altova GmbH

フィールドタイプ	説明
Weekday	曜日（Sunday, Monday...）
WEEKDAY	曜日（SUNDAY, MONDAY...）
weekday	曜日（sunday, monday...）
Wkd	曜日（Sun, Mon...）
WKD	曜日（SUN, MON...）
wkd	曜日（sun, mon...）
Month	月（January, February...）
MONTH	月（JANUARY, FEBRUARY...）
month	月（january, february...）
Mon	月（Jan, Feb...）
MON	月（JAN, FEB...）
mon	月（jan, feb...）

フィールドの長さと入力値の長さに関するメモ

データコンポーネントの長さに関する以下の点に留意してください:

データフィールドの長さ: MM、DD、HH、hh、mm、ss といったフィールドが使用された場合、定義の中に2文字分の長さを与える必要があります。y または Y フィールドが使用された場合、定義で使用されている y または Y 文字の数により、出力の長さが決定されます。例えば、YYY という文字を指定した場合、2006 に対応した出力の値は 006 になり、YYYYYY という定義の場合、002006 となります。下記にある base 年に関するメモも参照ください。

フィールドの長さを延長: ＊（アスタリスク）を使うことで、意味を持たない数値フィールド（integer や decimal など）の長さを延長することができます。decimal の場合、小数点の前と後の両方で使用することができます。例えば、*0.00* という値フォーマットでは、0 により指定された場所だけではなく、整数部部分と小数点以下にある任意の場所で、数値が空の時には 0 が使用されるようになります。

***Authentic View* における入力の長さ**: Authentic View に表示されるノードのコンテンツは、そのノードに対して定義された値フォーマットをベースにしたものになります。そのため、値フォーマットの定義により許されている数の文字を超えたかたちで Authentic View ユーザーが入力を行うことができなくなります。この機能を使用することで、Authentic View の入力を制限することができます。XML ドキュメントにて**既にある**値の長さが値フォーマットにより指定された長さを超える場合、全体の値が表示されます。

メモ フィールドによりテキストのレンダリングが行われない場合、Windows の地域オプションが原因である場合もあります。例えば Windows の地域言語設定がドイツ語になっている場合、AM/PM フィールドに対して空の文字列が返されます。

接頭部ならびに接尾部の修飾子

接頭部ならびに接尾部の修飾子により、テキスト配置とフィールドのポジティブ/ネガティブ表示を修正することができます。接頭ならびに接尾で使用することのできる修飾子を以下のテーブルに示します。

接頭	接尾	説明

<		左揃えを行い、テキストに対してデフォルトの振る舞いを行います。数値に対しては、デフォルトで右揃えとなっています。
>		数値に対する右揃えを行います。
?		負の数値に対してマイナスのシンボルを付け加えます。数値に対するデフォルトのオプションになります。
<?		負の数値に対してマイナス記号を左寄せして表示します。マイナス記号の後には左寄せされた数値が続きます。
<?>		負の数値に対してマイナス記号を左寄せして表示します。数値は右寄せにより表示されます。
-	-	負の数値に対して、マイナス記号が隣り合わせで表示され、そうでない場合は空白が表示されます。接頭部にて指定されると数値の前にマイナス記号が、接尾部の場合は数値の後に記号が表示されます。
<-	>-	負の数値に対してマイナス記号が表示され、それ以外の場合には空白となります。数値と記号は隣り合って表示され、接頭部の場合は数値の前にマイナス記号が、接尾部の場合は数値の後に記号が表示されます。
<->		負の数値に対してマイナス記号が表示されます。数値は右寄せにより表示されます。
+	+	常にプラスまたはマイナス記号を数値の隣に表示します。接頭部では数値の前に、接尾部では数値の後に表示されます。
<+	>+	常にプラスまたはマイナス記号を数値の隣に表示します。接頭部は左寄せ、接尾部は右寄せで表示が行われます。
<+>		常にプラスまたはマイナス記号を表示します。記号は左寄せで表示され、数値は右寄せで表示されます。
(	)	負の数値の場合かっこが表示され、それ以外の場合は空白が数値の隣に表示されます。
<(		負の数値の場合かっこが表示され、それ以外の場合は空白が数値の隣に左寄せで表示されます。
<(>		負の数値の場合かっこが表示され、それ以外の場合は空白が表示されます。左かっこが左寄せで表示され、その後に数値、そして数値の隣に右かっこが表示されます。
[	]	負の数値の場合かっこが表示され、それ以外の場合は空白が数値の隣に表示されます。
*	*	（接頭部にて）左側に対して、かつ（接尾部にて）右側に対して数値の延長を行います。
_	_	空白スペース。
^	^	オプションで定義された文字の挿入。
	th	英語における序数（1st、2nd、3rd、4th など）。
	TH	英語における序数（1ST、2ND、3RD、4TH など）。

フィールドオプション

フィールドオプションにより、高度な修正をフィールドに対して行うことができます。以下のオプションを使用することができます:

オプション	説明
uc	大文字に変換。
lc	小文字に変換。
left	左寄せ。
right	右寄せ。
ro	（XML にある）値の読み取りだけで、編集を許可しない。
edit	編集可能なフィールド（デフォルトで有効）。
dec=<char>	小数点の文字を指定（デフォルトではピリオド）。
sep=<char>	数値の区切りに使用される文字を指定（デフォルトではコンマ）。
fill=<char>	塗りつぶしに使用する文字を指定します。
base=<year>	年フィールドのbase 年を指定（以下のメモを参照）。
pos	プラス記号だけを表示、負の数値の入力を行うこともできます。

フィールドオプションを使用することで、小数点にコンマを使用して、数値の区切りにピリオドを使用するという欧州式の数値フォーマットを生成することができます（例: 123.456,78）。

上の値を生成するために使用される値フォーマットは以下のようになります: ###,###.##(dec=,,sep=.)。

フィールドには英国式のフォーマットが指定されたままになっている一方で、dec ならびにsep オプションにより数値の小数点と区切り記号が Windows OS の地域設定に従い指定されている点に注目してください（**コントロールパネル | すべてのアイテム | 地域 | フォーマット**）。

Base 年に関するメモ

短い年フォーマット（例: yy やYY）を使用する場合、指定されたbase 年により世紀部分が割愛されます。例えば、base 年フィールドオプションをDD-MM-yy(base=1940) というように、定義にて使用することができます。base 年にて指定された数値の下二桁以上の値がユーザーにより入力された場合、base 年と同じ世紀が使用されます。ユーザーにより入力された値がbase 年にて指定された数値の下二桁未満となる場合、base 年の世紀に1を加えた数値が入力された年の世紀になります。例えば、base=1940 とセットされており、Authentic View ユーザーにより50 が入力された場合、XML ドキュメントに入力される値は1950 となります。23 という数値がユーザーにより入力された場合、2023 という値がXML ドキュメントに入力されます。

以下の点に注意してください：

- 通常２桁の数値が広く使用されますが、１桁の数値や３桁の数値を base 年とともに使用することもできます。
- 短い年フォーマットに対して使用することのできるデータ型は xs:date、xs:dateTime、xs:gYear、そして xs:gYearMonth となります。
- 値フォーマットが自動計算コンポーネントに対してセットされる場合、値フォーマットダイアログにて正しいデータ型が選択されていることを確認してください。選択された日付のデータ型が自動計算の結果に対応している必要があります。
- yy フィールド型が使用される場合、デフォルトの base 年は 1930 となります。明示的に base 年を指定することで、デフォルトの設定がオーバーライドされます。
- YY フィールドタイプは、base 年度をセットすることなく設定され、base 年を指定すること無く YY フィールド型を使用する場合、Authentic View ユーザーは４桁のうち最後の２桁だけを修正することができるようになります。最初の２桁は変更されることなく XML ドキュメントに保持されます。

(C) 2015-2021 Altova GmbH

関連項目

- 値フォーマットのメカニズム
- 日付のフォーマット

8.4 CSS スタイルを使用する

SPS ドキュメントのスタイルはCSS ルールによりデザインされます。スタイルルールは以下のように指定することができます:

- 外部 CSS スタイルシートにて: デザイン概要 サイドバーとスタイルレポジトリ サイドバーから外部 CSS スタイルシートを追加することができます。
- グローバルスタイルにて: SPS の内部ならびにその開始部分で定義が行われます(HTML 出力では、これらグローバルスタイルが head 要素内にある style 子要素にて定義されます)。グローバルスタイルはスタイルリポジトリサイドバーにて定義されます。
- ドキュメントにある個々のコンポーネントに対してローカルで: HTML 出力では、これらのルールが個々のHTML 要素内にある style 属性にて定義されます。ローカルのスタイルルールは スタイル サイドバーにて定義されます。

上記のメカニズムによりスタイルを作成する方法の詳細については、このセクション以下にあるサブセクションにて記述されます。

用語解説

CSS スタイルシートは1つ以上のスタイルルールにより構成されます。ルールは以下のように表示されます:

```
H1 { color: blue }
```

または:

```
H1 { color: blue;
         margin-top: 16px; }
```

各ルールにはセレクター(上の例における H1)と宣言(color: blue)が含まれます。宣言はプロパティ(例えば color)と値(blue)が含まれます。各プロパティと値のペアのことをスタイル定義と言います。StyleVision では、CSS スタイルを スタイル サイドバー(ローカルスタイル)ならびにスタイルレポジトリサイドバー(グローバルスタイル)にて定義されます。

カスケードの順序

CSS ではカスケード順序が適用されます。これにより、評価されるルールの順序は以下のように決定されます:

1. **宣言場所**: 外部スタイルシートの優先度はグローバルスタイルより先低く、グローバルスタイルの優先度はローカルスタイルより先低くなります。外部スタイルシートはインポートされたものとして扱われ、後からインポートされたスタイルシートがより高い優先度を持つようになります。
2. **詳細性**: 同じ要素に対して2つのルールが適用された場合、より詳細なセレクターがより高い優先度を持つようになります。
3. **順序**: 2つのルールが同じ場所から同じ詳細性を持っている場合、スタイルシートの後の方に出現するルールがより高い優先度を持つようになります。スタイルシートがインポートされる場合、インポート元スタイルシートの前に来るものとして処理されます。

モジュール SPS における CSS スタイル

SPS モジュールが別のSPSへ追加された場合、参照元のSPS にあるCSS スタイルが、追加されたモジュール内のスタイルより先高い優先度を持つようになります。複数のモジュールが追加された場合、これらモジュール内にあるCSS スタイルのうち、より下部にあるスタイルが高い優先度を持つようになります。モジュールSPS に関する情報については モジュールSPS のセクションを参照ください。

CSS サポートin Internet Explorer

IE 6.0 より前のInternet Explorer (IE) では、IE 6.0 以降とは異なるCSS ルールの解釈が行われます。従って、デザイナーにとって、どのバージョンのIE に対してデザインを行うのかが重要になります。6.0 以降のIE では新旧両方の解釈を行うことができ、新しいバージョンにて過去の解釈を行うこともできます。IE 6.0 以降で使用される解釈方法はHTML ドキュメントコード内部で指定することができます。SPS では、指定 によりHTML とAuthentic View 出力ドキュメントのスタイルを、Internet Explorer の古い/新しい解釈 によりデザ

インすることができます。選択された解釈方法に従うかたちでCSS スタイルを定義することができます。詳細については プロパティ: CSS サポートを参照ください。

メモ CSS の仕様に関する詳細については、http://www.w3.org/TR/REC-CSS2/ を参照ください。

関連項目

- スタイルレポジトリサイドバー
- スタイルサイドバー
- CSS サポート
- モジュールSPS

8.4.1 外部スタイルシート

このセクションではStyleVision のGUI から外部 CSS スタイルシートを管理する方法について説明します。以下にある項目を説明します:

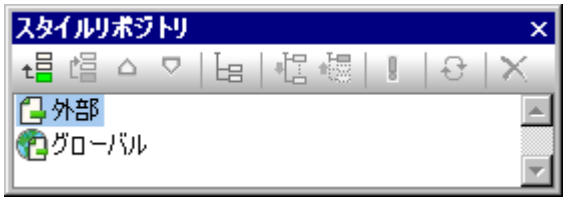
- 外部 CSS スタイルシートをSPS へ追加する
- 外部 CSS スタイルシートにおけるコンテンツの確認と修正を行う
- 優先順位を変更する
- CSS スタイルシートセットと、単一のCSS スタイルシートを切り替える

外部 CSS スタイルシートは、スタイルレポジトリサイドバーとデザイン概要サイドバーという2つのサイドバーにて管理することができます。外部スタイルシートが両方のサイドバーに表示されている場合、あるサイドバーにて行われた変更は、自動的に他のサイドバーにも反映されます。

外部 CSS スタイルシート SPS に追加する

以下の操作により外部 CSS スタイルシートをSPS へ割り当てる事ができます:

1. デザインビューにてスタイルリポジトリウィンドウにある外部アイテムを選択します（以下のスクリーンショットを参照）。



2. スタイルリポジトリツールバーの左側にある**追加**ボタンをクリックします（上のスクリーンショットを参照）
3. ファイルを開くダイアログが表示されるので、目的の CSS ファイルを選択し**開く**ボタンをクリックします。CSS ファイルが外部アイテム以下に追加されます（以下にある表示を参照ください）。
4. ステップ1から3を繰り返すことで、別の外部 CSS スタイルシートを追加することができます。それらの CSS スタイルシートは外部アイテムの、それまでに追加された外部 CSS スタイルシート以下に追加されます。

メモ デザイン概要 サイドバーからも外部 CSS スタイルシートを追加することができます。

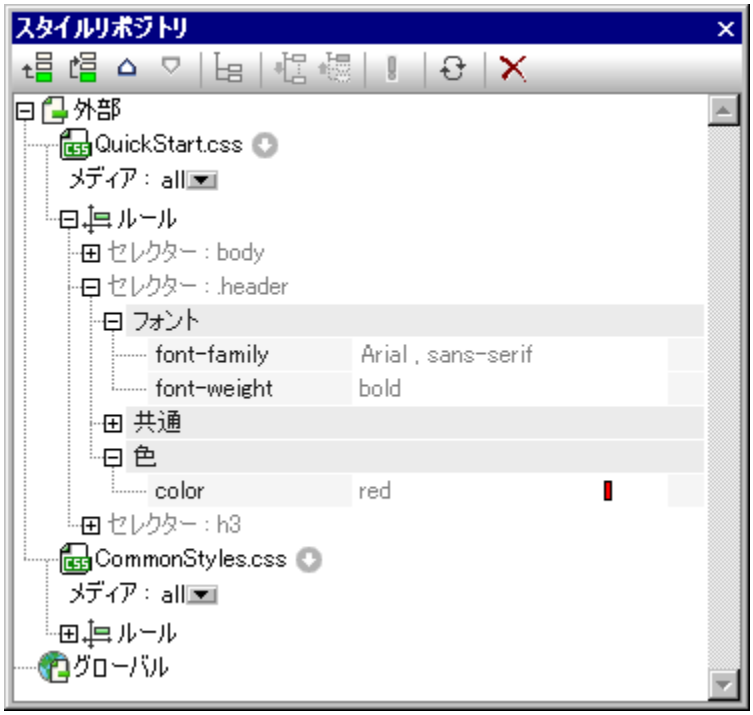
外部 CSS スタイルシートの確認と修正

外部 CSS スタイルシートのツリーは以下のように構成されます(以下にあるスクリーンショットも参照ください):

```
- CSS-1.css  (マウスオーバーによりファイルの位置が表示されます)
  - Media (スタイルリポジトリウィンドウにて定義することができます)
  - Rules (そのまま編集することはできず、CSS ファイル内部で編集を行う必要があります)
    - Selector-1
      - Property-1
      - ...
      - Property-N
    - ...
    - Selector-N
+ ...
+ CSS-N.css
```

スタイルリポジトリウィンドウでは、そのスタイルシートが適用されるメディアを編集することができます。アイテムの右側にある矢印アイコンをクリックして、ドロップダウンリストから目的のメディアを選択してください。外部 CSS スタイルシートに定義されたルールがスタイルリポジトリウィンドウに表示されますが、編集することはできません。各アイテムの左にある + および - シンボルをクリックすることで、ツリー内部にあるセレクターアイテムを展開/縮退することができます(以下のスクリーンショットを参照)。

外部スタイルシートを削除するには、スタイルシートが選択された状態でスタイルリポジトリツールバーにある**リセット**ボタンをクリックしてください。



外部 CSS スタイルシートの優先順位を変更する

スタイルリポジトリウィンドウにて割り当てられた外部 CSS スタイルシートは、@import 命令により HTML 出力ファイルへインポートされます。HTML ファイルでは以下のような記述がなされます:

```
<html>
        <head>
```

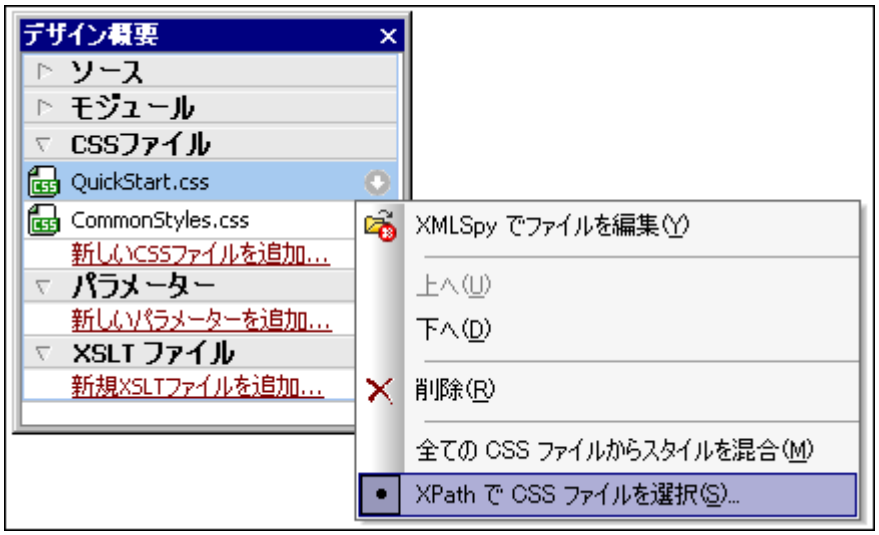
```
            <style>
                <!--
                @import url("ExternalCSS-1.css");
                @import url("ExternalCSS-2.css")screen;
                @import url("ExternalCSS-3.css")print;
                -->
            </style>
        </head>
        <body/>
</html>
```

HTML ファイルにて表示されるファイルの順序は、スタイルリポジトリの外部ツリー以下ならびにデザイン概要サイドバーのCSS ファイルツリー以下に表示されている順序に従います。スタイルリポジトリにてCSS ファイルの順序を変更するには、優先順位を変更するスタイルシートを選択し、スタイルリポジトリツールバーにある上へ ボタンまたは下へ ボタンを使用することで、ツリーにおけるスタイルシートの位置（優先順位）を変更してください。デザイン概要サイドバーではCSS スタイルシートの編集ボタンをクリックして、必要に応じて上へまたは下へコマンドを選択することができます。

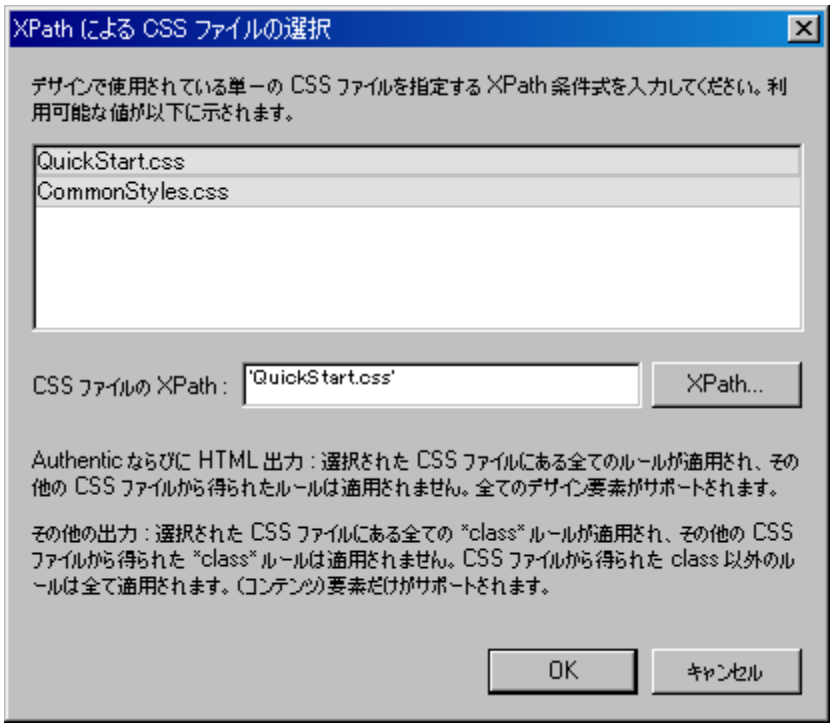
重要: 最も下に表示されているスタイルシートが**最も高いインポート優先度**を持ち、表示されている順序が上がるにつれてスタイルシートのインポート優先度が下がることに注意してください。上に示されている例のインポート優先度は（1） ExternalCSS-3.css、(2) ExternalCSS-2.css、(3) ExternalCSS-1.css の順で高くなります。異なるスタイルシートにて定義された2つのCSS ルールが同じセレクターを持っている場合、より高いインポート優先順位のスタイルシートにあるルールが適用されます。

すべてのCSS ファイルと単一のCSS ファイルを切り替える

(i) 全ての CSS ファイルにあるルールをカスケードルールの優先順位に従い適用する他にも、(ii) 選択された CSS ファイルのルールだけを適用することができます。このオプションはデザイン概要サイドバーから選択することができます（以下のスクリーンショットを参照）。表示されている CSS ファイルの編集ボタンをクリックして、**全ての CSS ファイルからスタイルを混合**コマンド、または **XPath で CSS ファイルを選択**コマンドを選択してください。このオプションはスタイルリポジトリウィンドウでも使用することができます。



XPath で CSS ファイルを選択コマンドをクリックした場合、XPath 条件式を入力するためのダイアログが表示されます（以下のスクリーンショットを参照）。ダイアログ上部のペインに表示されているとおりに、SPS 内にある CSS ファイルのどれかを評価する XPath 条件式を入力する必要があります。ファイル名を文字列として入力する場合、その他全ての XPath 条件式と同様に、文字列をシングルクォートにより囲まなければならない点に注意してください。



以下の点に注意してください:

- *単一の CSS ファイルが選択されている場合* : Authentic ならびに HTML 出力の場合、選択された CSS ファイルにある全てのルールが、ルールをサポートする全てのデザインコンポーネントに対して適用されます。RTF 出力では、選択された CSS ファイルのクラスセレクタールールだけが適用されます。非クラスセレクターのルールは、優先順位に従うかたちで全ての CSS ファイルから適用されます。 RTF出力ではこれらのルールは 次のデザインコンポーネント: 自動計算、(コンテンツ)プレースホルダ、段落(ブロック)コンポーネント、とテーブルセルに対して適用されます。
- *全てのCSS ファイルにあるスタイルルールが使用される場合* : Authentic ならびに HTML 出力の場合、選択された CSS ファイルにある全てのルールが、ルールをサポートする全てのデザインコンポーネントに対して適用されます。RTF 出力では、非クラスセレクタールールだけが、優先順位に従うかたちで全ての CSS ファイルから適用されます。

関連項目

- スタイルレポジトリ
- グローバルスタイル
- ローカルスタイル
- CSS サポート

8.4.2 グローバルスタイル

グローバルスタイルはSPS デザイン全体に対して適用されます。スタイルレポジトリはスタイルレポジトリのグローバル以下に表示されます。これらのスタイルは、CSS ルールとしてAuthentic View と HTML 出力へ渡されます。HTMLドキュメントでは、これらのCSS ルールが/html/head/style 要素内に記述されます。

スタイルリポジトリでは、セレクターとそのセレクターに対応するCSS プロパティから構成されるCSS がグローバルスタイルとなります。以下の2つのステップによりスタイルを作成することができます:

- 新たなスタイルを追加して、それに対する CSS セレクターを宣言する
- そのセレクターに対する CSS プロパティを定義する

サポートされるセレクター

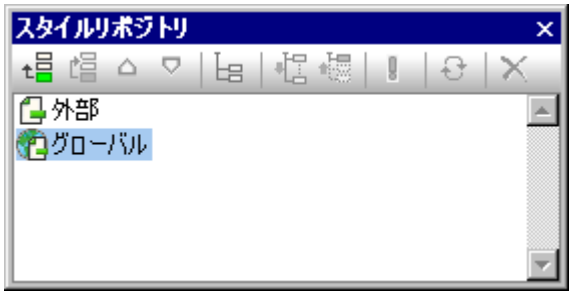
次の セレクター がサポートされます:

- *ユニバーサルセレクター* : * により示されます。
- *型セレクター* : h1 といった要素名を対象にします。
- *属性セレクター* : [class=maindoc] といった属性を対象にします。
- *クラスセレクター* : 例: .maindoc
- *ID セレクター* : 例: #header

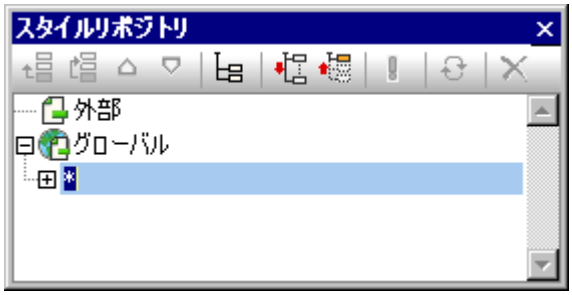
グローバルなスタイルを追加する

以下の操作によりグローバルスタイルを SPS デザインへ追加することができます:

1. デザインビューにてスタイルリポジトリウィンドウにあるグローバルアイテムを選択します(以下のスクリーンショットを参照)。



2. スタイルリポジトリツールバーの左にある**追加**ボタンをクリックします。(全ての HTML 要素が選択される) * セレクターを伴うグローバルスタイルが、グローバルツリー内に挿入されます。新たに挿入されたグローバルスタイルのデフォルトセレクターにはユニバーサルセレクターが使用されます。
3. セレクターをダブルクリックすることで、デフォルトのユニバーサルセレクターを変更することができます。



4. セレクターに対して CSS プロパティをセットします。プロパティをセットする方法については、スタイルの値の設定セクションを参照ください。
5. セレクターに対して CSS プロパティをセットします。プロパティをセットする方法については、

メモ

- スタイルリポジトリウィンドウにある**挿入**ボタンをクリックすることで、グローバルツリーにて選択されているグローバルスタイルの前に新たなグローバルスタイルを挿入することもできます。**追加**ならびに**挿入**コマンドは、グローバルセレクターを右クリックすることでも選択することができます。
- グローバルツリー内にあるアイテムを右クリックして、「**セレクターを追加 | HTML | HTML の要素名**」を選択することで、HTML 要素に対するグローバルスタイルのセレクターを挿入追加することができます。

グローバルスタイルの編集と削除

スタイルセレクターとそのプロパティは、スタイルリポジトリウィンドウにて編集することができます。

- セレクターを編集するには、セレクター名をダブルクリックしてテキストフィールドにて編集を行います。
- スタイルにおけるプロパティ値の定義と編集については、スタイルの値の設定を参照ください。スタイルには3種類の表示方法があり、それらのビューと、ビューの切り替えについては定義ビューのセクションを参照ください。

グローバルスタイルを削除するには、そのスタイルを選択して、スタイルリポジトリツールバーにある**リセット**ボタンをクリックしてください。

グローバルスタイルの優先順位を変更する

スタイルリポジトリウィンドウにて割り当てられたグローバルスタイルは、CSS ルールとして /html/head/style 要素内に配置されます。HTML ファイルでは、これらのルールは以下のように表示されます:

```
<html>
	<head>
		<style>
			<!--
			h1		{ color:blue;
				  font-size:16pt;
				}
			h2		{ color:blue;
				  font-size:14pt;
				}
			.red	{ color:red;}
			.green	{ color:green;}
			.green	{ color:lime;}
			-->
		</style>
	</head>
	<body/>
</html>
```

グローバルスタイルが Authentic View ならびに HTML ドキュメントにて表示される順序は、スタイルリポジトリのグローバルツリーにて表示される順序に対応します。Authentic View ドキュメントにおける順序には意味があり、2つのセレクターにより同じノードが選択される場合、より下部に表示されているセレクターが、より高い優先順位を持つことになります。例えば、上記のルールが HTML ドキュメントに表示されており、HTML ドキュメント内に `<h1 class="green">` という要素と、3つのグローバルスタイルがこの要素に一致する場合: `h1` セレクターと2つの `.green` クラスセレクターを持つ場合。色 `green` を持つ `.green` セレクターの後に発生し、優先順位を与えられるため、and 色 `lime` を持つ `.green` セレクターのプロパティが適用されます。(クラス セレクターはノードセレクターより先高い優先順序が与えられるため、`.green` セレクターには、`h1` セレクターに対する位置に関わらず、`h1` セレクターより先高い優先順位が与えられます)。`h1` スタイルのフォントサイズは、しかしながら、`<h1>` 要素 に一致する高い優先順位を有し、フォントサイズプロパティを有するセレクターが存在しないため、`<h1>` 要素に適用されます。

グローバルスタイルの優先順位を変更するには、そのスタイルが選択された状態で、スタイルリポジトリツールバーにある**上へ**ならびに**下へ**ボタンを使用します。例えば、`.green` グローバルスタイルが `.red` スタイルの前に移動されると、`.red` スタイルの色のプロパティは、`.green` スタイルのプロパティより先高い優先度を持つようになります。

クラスセレクターは型のセレクターより先常に優先されることご注意してください。
つまり、セレクターの順序が `.red` `.green` `h1` `h2` に変更されると、`h1` と `h2` は、`green` のままです。

関連項目

- スタイルレポジトリ
- 外部 CSS スタイルシート
- ローカルスタイル
- CSS サポート

8.4.3 ローカルスタイル

スタイルがローカルで定義されると、コンポーネントに対してスタイルが直接定義されます。ローカルのルールは、グローバルのスタイルルールならびに外部 CSS スタイルシートより先高い優先度を持つことになります。ローカルで定義されるスタイルはCSS スタイルで、フォーマットツールバーまたはスタイルサイドバーにて定義されます(グローバルスタイルの定義と外部 CSS スタイルシートの定義はスタイルレポジトリで行われる点に注意してください)。

フォーマットツールバーを使用してのローカルスタイル指定

デザイン内のコンテンツを選択し、フォーマットツールバーからローカルスタイルを適用することができます(以下のスクリーンショットを参照)。

div や h1、pre といった定義済みのHTML フォーマットに対して、テキストのスタイル、背景色、テキストの配置、ハイパーリンクを適用することができます。詳細については、フォーマットツールバーのセクションを参照ください。

スタイル サイドバーを解してのローカルスタイル指定

以下に示されるステップにより、ローカルにおけるスタイルの定義を行うことができます:

1. スタイルを作成するコンポーネントをデザインビューにて選択します。ノードタグを除く全てのコンポーネントのスタイルを作成することができます。デザインビューにて選択されたコンポーネントが、スタイルサイドバーの**スタイル**ペインに表示されます(以下のスクリーンショットを参照)。以下のスクリーンショットは、コンテンツコンポーネントがデザインビューにて選択された状態で、コンテンツがスタイルカラムに表示されます。

デザインビューにて選択されているコンポーネントが他のコンポーネントを含んでいる場合、選択されたコンポーネント内部にあるその他全てのコンポーネントが、コンポーネントの種類ごとにスタイルサイドバーのスタイルカラムにて表示されます。以下に示されるスクリーンショットではデザインビューにて選択されたコンポーネントに含まれているコンポーネントの種類が表示されています。各種コンポーネントの左には、選択対象内部に存在するインスタンスの数が表示されます。例えば、以下のスクリーンショットでは、16個のテキストコンポーネントと2個の自動計算コンポーネントが選択対象に含まれていることが分かります。あるコンポーネントが選択された状態で Shift キーを押下したまま別のコンポーネントを選択することで、選択された範囲内のコンポーネント全てを選択することができます。

2. デザインビューにて選択を行い、スタイルペインにてスタイルの指定を行う**コンポーネントの種類**を選択します。選択されたコンポーネントの種類が複数個ある場合、定義したスタイルはそれらのコンポーネント全てに対して適用されることになります。例えば、上のスクリーンショットにある**16-テキスト**アイテムを選択した場合、定義したスタイルは16あるテキストコンポーネント全てに対して適用されることになります。このうち4個のテキストコンポーネントだけを個別に定義したい場合、目的のテキストコンポーネントを個別に選択してスタイルの定義を行う必要があります。異なるスタイル値を持ったコンポーネントがデザインビュー

にて選択されている場合、それらの値全てがスタイル定義ペインに表示されます。上のスクリーンショットでは、ある自動計算に normal の font-weight を割り当てられている一方で、別の自動計算には bold font-weight が割り当てられていることが分かります。

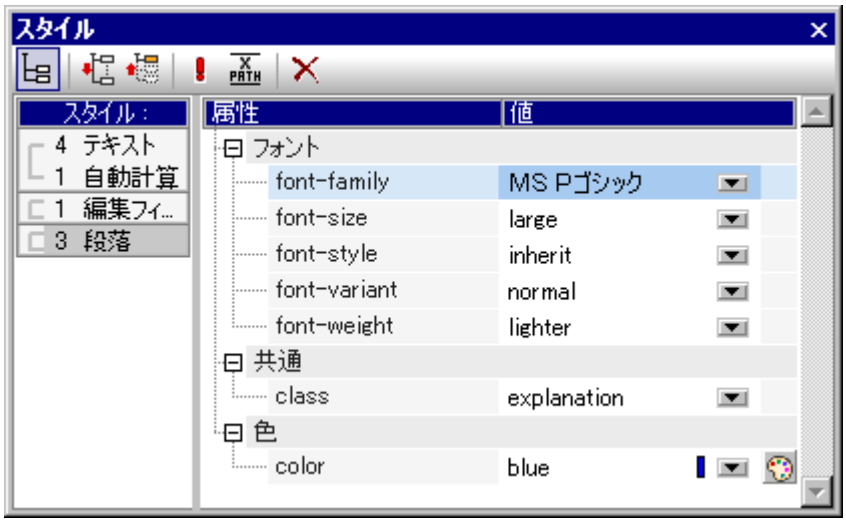
3. **スタイル**ペインにてスタイルを定義するコンポーネントを選択し、スタイル定義ペインにてスタイルの定義を行います。定義方法については、スタイルの値の設定. セクションを参照ください。

関連項目

- スタイルの値の設定 スタイルプロパティの定義方法に関する情報。
- グローバルスタイル グローバルスタイルの使用に関する情報。
- 外部 CSS スタイルシート 外部 CSS スタイルシートの使用に関する情報。
- スタイル サイドバー スタイル サイドバーの説明
- CSS サポート CSS サポートのレベルを調節する出力ドキュメントの設定に関する説明

8.4.4 スタイルの値の設定

スタイルサイドバーのスタイル定義ペイン では、スタイルカラムで選択された種類のコンポーネントに対してスタイルプロパティを定義することができます(以下のスクリーンショットを参照)。スタイルカラムでは、Ctrl キーや Shift キーを押下しながら複数のコンポーネントを選択することもできます。複数種類のコンポーネントが選択されている場合、スタイル定義ペインにて定義した値が、選択されているコンポーネントの全インスタンスに対して適用されることになります。.



スタイルプロパティグループ

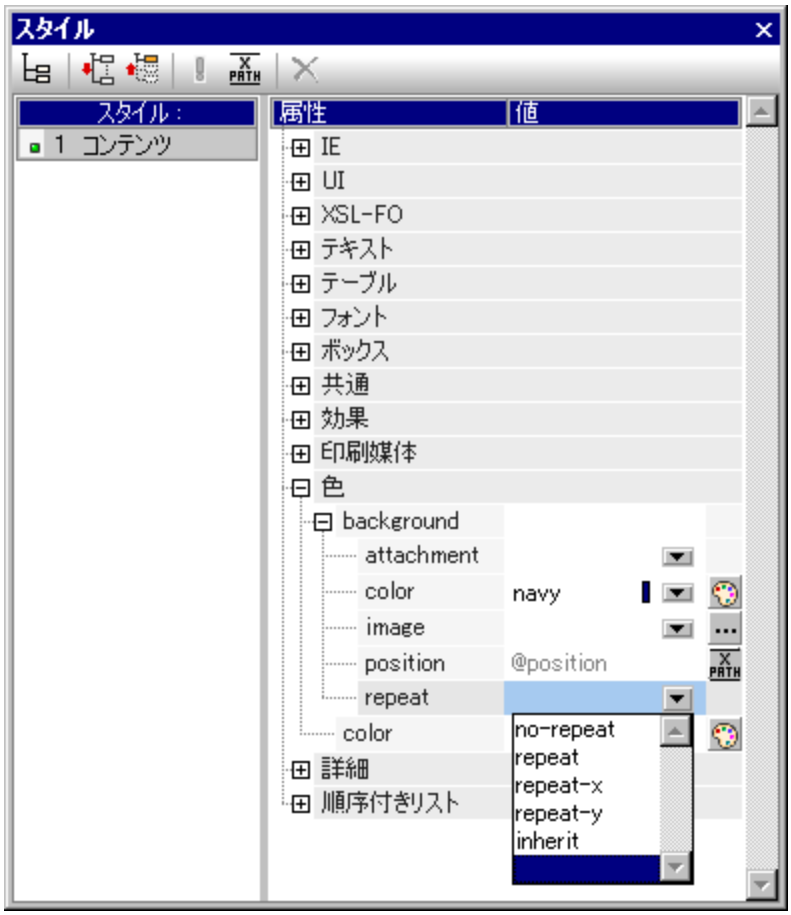
スタイル定義ペインにて編集することのできるスタイルプロパティは、以下のスクリーンショットで示されるように、グループごとに管理されます。

値のあるものを表示, 全て展開 と 全て縮退 ツールバーボタンツールバーボタンを使用することで、プロパティの表示を修正することができます(以下のスクリーンショットを参照)。

スタイル値を入力する

スタイルプロパティの値(スタイル値)は、以下の方法により入力することができます:

- 値カラムに直接入力する: プロパティを選択し、値カラムをダブルクリックします。キーボードにより値を入力し、Enter キーを押下するか、その他の場所をクリックすることで入力された値が確定されます。
- そのプロパティのコンボボックスにあるドロップダウンリストから値を選択する: コンボボックスにあるドロップダウンの矢印をクリックすると、スタイル値のオプションが表示されます。以下のスクリーンショットでは、(background-) repeat プロパティに対するオプションが表示されています。ドロップダウンリストから目的の値を選択してください。
- 値カラムの右側にあるアイコンを使用する: 2種類のアイコンが用意されており、関連するプロパティに対してのみこれらのアイコンは表示されます。((background-)color プロパティにある)カラーパレットは色の選択を行うために使用され、((background-)image プロパティにある)ダイアログアイコンからは、ファイルの参照を行うことができます。



- XPath 式により、スタイルの値を割り当てることもできます。

スタイル値の修正と削除

スタイル値が正確に入力されなかった場合、値が赤色で表示されます。プロパティの値の挿入で説明されているとおりに値の修正を行ってください。

スタイル値を削除（つまり、スタイル値をリセット）するには、スタイルサイドバーのツールバーにあるリセットボタンをクリックしてください。その他にも、プロパティの値カラムをダブルクリックして、Delete または Backspace キーにより値を削除した後に Enter キーを押下することもできます。

関連項目

- スタイルサイドバー
- CSS スタイルをローカルに定義する
- CSS スタイルをグローバルに定義する
- 外部 CSS スタイルシート
- CSS サポート

8.4.5 XPath を使用したスタイルプロパティ

XPath 条件式を使用することでもスタイルを割り当てることができます。この機能により、スタイルプロパティの値を XML データや、XPath 条件式から与えることができます。更に、XPath 2.0/3.0 の doc() 関数を使用することで、アクセス可能な XML ドキュメントにあるノードへのアクセスを行うことが可能になります。スタイルの定義を XML データから取得するだけではなく、XML データのコンテンツや構造により、異なるスタイルを与えることも可能になります。例えば XPath 2.0/3.0 の if...else ステートメントを使用することで、シーケンスにおける要素の位置に従って、異なる背景色を選択することが可能になります。これにより、これらの要素がテーブルで表示される時に、偶数番号と奇数番号の背景色がそれぞれ別の色となります（以下の例を参照）。また、ノードのコンテンツに従うかたちでスタイル値を変更することもできます。

XPath 条件式を使用することができるスタイルプロパティ

XPath 条件式は、以下のスタイルプロパティに対して入力することができます:

- スタイルサイドバーに表示されている全てのプロパティ
- プロパティサイドバーの**共通**、**イベント**、**HTML** グループ内にあるプロパティ

プロパティ値に対する静的モードと動的（XPath）モード

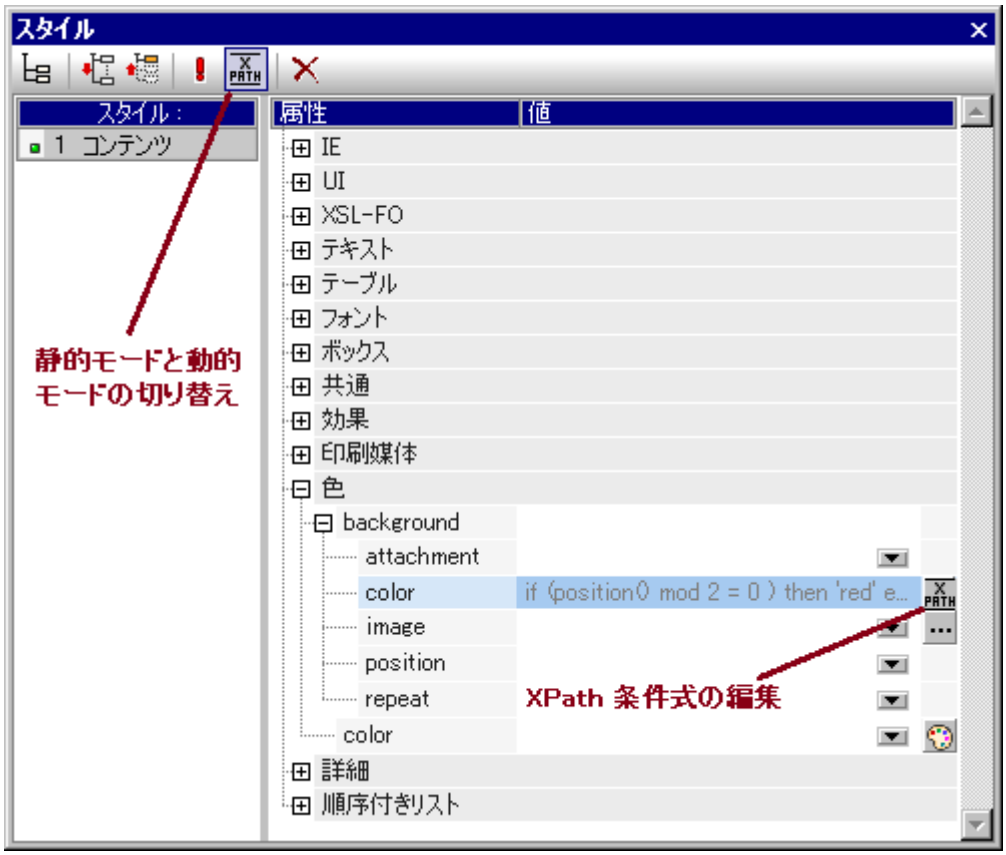
XPath 式の有効化プロパティには、2つのモードが用意されています:

- 静的モード: サイドバーの値カラムにプロパティの値が直接入力されます。例えば、デザインコンポーネントの背景色に対して red という値をサイドバーに直接入力します。
- 動的モード（または XPath モード）: XPath 条件式が入力されます。条件式はランタイムにて評価され、その結果がプロパティの値として入力されます。例えば、デザインコンポーネントの背景色に /root/colors/color1 という XPath 条件式を入力することができます。ランタイムでは、/root/colors/color1 ノードのコンテンツが背景色プロパティの値として使用されます。

静的モードと動的（XPath）モードの切り替え

XPath 条件式を使用することができるプロパティでは、静的モードがデフォルトで選択されています。プロパティを動的（XPath）モードに切り替えるには、そのプロパティを選択してサイドバーのツールバーにある XPath アイコンをクリックします（以下のスクリーンショットを参照）。

(C) 2015-2021 Altova GmbH



静的な値がそのプロパティに対して既に入力されている場合、その値がクリアされ、動的モードへの切り替えが行われます。XPath 式の編集ダイアログが表示されるので、そのプロパティで使用される XPath 条件式を入力します。「OK」をクリックすることで編集を完了します。

プロパティに対して XPath 条件式が入力されると、そのプロパティの値カラムに **XPath 条件式の編集** ボタンが表示されます（上のスクリーンショットを参照）。このボタンをクリックすることで、既に定義された XPath 条件式を編集することができます。静的モードへの切り替えを行うには、ツールバーにある XPath アイコンをクリックしてください。定義された XPath 条件式がクリアされ、静的モードへの切り替えが行われます。

メモ 静的モードと動的モードのうち、同時にアクティブにすることができるのは1つに限られるという点に注意してください。モードの切り替えを行うと、それまでに入力された値がクリアされるため、再度モードの切り替えを行なってもフィールドは空になります（過去に入力した値 / 条件式を取り戻すには、元に戻すコマンドコマンドを使用してください）。SPS を編集した後に再度プロパティを選択しても、過去に選択されたモードにてプロパティは開かれます。

XPath 条件式の作成と編集

XPath の定義は、XPath 式の編集ダイアログにて作成ならびに編集することができます。2つの方法によりこのダイアログへアクセスすることができます:

- （サイドバーのツールバーにあえる XPath アイコンをクリックすることで）静的モードから動的モードへの切り替えを行うと、XPath 式の編集ダイアログが表示されます。プロパティが既に動的モードになっている時にツールバーの XPath アイコンをクリックすると、静的モードへの切り替えが行われ、XPath 条件式の編集ダイアログは表示されないことに注意してください。
- プロパティに対して XPath 条件式が既に定義されている場合、値フィールドにある **XPath 条件式の編集** ボタンをクリックすることで XPath 式の編集ダイアログへアクセスすることができます。ダイアログには、そのプロパティに対して既に定義されている XPath 条件式が表示され、そのまま編集を行うことができます。

XPath 条件式の入力または編集を行った後は、「OK」をクリックして完了してください。

XPath 式により得られる値

XPath 条件式を使用することで、(i) XML ファイルから取得された値をプロパティの値として使用することができるようになり、(ii) 処理されている XML ドキュメントのコンテンツや構造を条件に対して使用することで、適切な値を選択することができるようになります。XPath 条件式からは、以下にある2種類のカテゴリーにより値が返されます:

- *XML ノードコンテンツ*
  XPath 条件式により、(i) SPS により処理されている XML ドキュメント内のノード、または (ii) アクセス可能な任意の XML ドキュメント内にあるノードへアクセスすることができます。例えば、Format/@color という条件式により、コンテキストノードの子要素である Format の color 属性にある値を、この XPath 条件式が定義されたプロパティの値としてセットすることができます。XPath 2.0 の doc() 関数を使用することで、その他の XML ドキュメントにあるノードへもアクセスすることができます。例えば、doc('Styles.xml')//colors/color-3 という XPath 条件式により、Style.xml という XML ドキュメントにある color-3 という要素内の値をプロパティの値として使用することができます。

- *XPath 式*
  XML ドキュメントではなく、XPath 条件式そのものにプロパティの値を入力することもできます。例えば、テーブルの行として出力される要素の背景色を、奇数番目と偶数番目により変更する例を考えてみましょう。このような処理は if (positikon() mod 2 = 0) then 'red' else 'green' といった XPath 2.0 の条件式により実現することができます。この条件式の戻り値は、'red' という文字列か、'green' という文字列になり、これらの文字列(のうちのどちらか)が、この XPath 条件式が定義されたプロパティの値として使用されることになります。この例では文字列リテラルとしてプロパティの値が入力されています。文字列リテラルを使う以外にも、if (position() mod 2 = 0) then doc('Styles.xml')//colors/color-1 else doc('Styles.xml')//colors/color-2 といった条件式を使用することで、XML ドキュメントから値を取得することもできます。'green' という XPath 条件式を定義することで、文字列をそのまま与えることもできますが、その結果は green という静的な値を与えた場合と変わりません。

関連項目

- CSS スタイルと作業
- XPath ダイアログ
- スタイル
- プロパティ
- CSS サポート

8.4.6 合成スタイル

合成スタイルは CSS テキストスタイルプロパティのグループとして表示されており、XML インスタンスドキュメントにおけるノードの属性の値に対応します。更に、スタイルシートに収められている任意の CSS テキストスタイルプロパティグループも合成スタイルとして扱われます。合成スタイルは以下のデザインコンポーネントにて指定することができます:

- 自動計算
- (コンテンツ)プレースホルダ
- 段落(ブロック)デザイン要素
- テーブルセル

合成スタイルの利点

合成スタイルには以下のような利点があります:

- スタイルプロパティは XML のデータで、ユーザーにより編集することができます。 Authentic View では、リッチエディット機能により合成スタイルのグラフィカルな編集を行うことができます。合成スタイルに対してリッチエディットをセットアップする方法の詳細については、セクションAuthentic における合成スタイルを参照してください。
- 上に示されるデザインコンポーネントのスタイルプロパティは、SPS にて割り当てられた XML に収められている複数のプロパティとなります。
- SPS デザインの作成段階にて、SPS デザイナーは要素に関連付けられている複数の合成スタイル間を素早く切り替えることができます。

属性にて合成スタイルを入力する

複数のスタイルプロパティにより構成される合成スタイルは、ソース XML ドキュメントの要素における属性の値として入力されます。例えば、以下に示される XML ソースドキュメントの desc-sytle 属性には、デフォルトの合成スタイルが含まれます:

```
<Desc desc-style="font-family:Verdana; font-size:12pt; color:blue">
```

同一要素に対して複数の合成スタイルを指定することもできます。この場合、各合成スタイルをそれぞれの属性に入力する必要があります:

```
<Desc styleBlue="font-family:Verdana; font-size:12pt; color:blue"
      styleRed ="font-family:Verdana; font-size:12pt; color:red">
```

同一の要素にて複数の合成スタイルを指定することができる場合、デザインコンポーネントの合成スタイルプロパティにて値を入力する際に、合成スタイルの切り替えを行うことができます(以下を参照ください)。

メモ: XML ドキュメントを妥当なものとするために、合成スタイルへアクセスするために使用される属性は、ソーススキーマにて定義する必要があります。

サポートされている CSS テキストスタイルプロパティ

合成スタイルでは、以下の CSS スタイルを使用することができます:

font-family	font-size	font-weight	font-style
color	background-color	text-align	text-decoration

合成スタイル値として属性を設定する

デザインコンポーネントの合成スタイルが属性として定義された場合、Authentic View ユーザーによりこの合成スタイルを編集することができるようになります。Authentic View ユーザーは、デザインコンポーネントのテキスト出力における任意の場所にカーソルを配置して、Authentic View のリッチエディットツールバーから、そのデザインコンポーネントの合成スタイルを編集することができるようになります。

以下の操作により、デザインコンポーネントの合成スタイルとして属性をセットすることができます:

1. デザインビューにて、属性を合成スタイルとして割り当てるデザインコンポーネントを選択します。以下のスクリーンショットでは、Desc 要素の(コンテンツ)プレースホルダが選択されています。

2. コンテンツコンポーネントの合成スタイルプロパティに表示されているコンボボックスに、コンテキスト要素の属性が表示されます（上のスクリーンショットの右下にあるプロパティウィンドウを参照ください）。デザインコンポーネントの合成スタイルとしてセットする属性を選択してください。コンボボックスには空のエントリーも表示されている点に注目してください。合成スタイルを使用しない場合は、空のエントリーを選択してください。この場合、そのデザインコンポーネントの出力に対して、Authentic View におけるリッチエディット機能が無効になります。

Authentic View では、リッチエディットツールバーを使用することで、そのデザインコンポーネントの合成スタイルを修正することができるようになります。

XPath 条件式を合成スタイルの値としてセットする

XPath 条件式を合成スタイルプロパティの値としてセットすることもできます。しかしこの場合、合成スタイルの値が（XML ソースドキュメントではなく）SPS 内部に収められるため、Authentic View にて合成スタイルの編集を行うことができなくなります。

XPath 条件式を合成スタイルプロパティの値としてセットするには、プロパティサイドバーのツールバーにある XPath アイコンをクリックして、表示される XPath ダイアログにて XPath 条件式を入力します。XPath 条件式は属性値テンプレートとして評価され、戻り値は HTML style 属性の値（その他の出力フォーマットでは、それに準ずる値）になります。

例えば、以下に示される n1:Person 要素の（コンテンツ）プレースホルダにて作成された XPath 条件式を考えてみましょう。

```
if (number(n1:Shares) gt 1000) then 'color:red' else 'color:green'
```

この条件式を使用することで、n1:Person 要素に n1:Shares という子要素が含まれており、その数値が 1000 を超える場合には n1:Person 要素の文字が赤色で出力され、それ以外の場合には緑色で出力されるようになります。XPath 条件式により返された値は、HTML style 属性の値として（その他の出力フォーマットでは、それに準ずる値が）出力ドキュメントに渡されます。

SPS により生成された XSLT スタイルシートでは、この XPath 条件式は以下にあるような属性値テンプレートとして評価されます：

```
<span style="{if (number(n1:Shares) gt 1000) then &apos;color:red&apos; else
&apos;color:green&apos;}">
```

HTML 出力では、条件の評価結果により、以下のどちらかが生成されます：

```
<span style="color:red">
```

または

```
<span style="color:green">
```

メモ 属性値テンプレートは、属性の値が XPath 条件式として読み取られる XSLT コンストラクターとなります。波かっこにより区切られ、属性の値を動的に割り当てることができます。

関連項目

- CSS スタイルと作業
- XPath ダイアログ
- スタイル
- プロパティ
- CSS サポート

8.5 Authentic におけるテキストスタイルの柔軟性

SPS は、Authentic View ユーザーもテキストのスタイルを指定することができるようにセットアップすることができます。Authentic View にてテキストフラグメントを選択し、選択されたテキストに対してスタイルプロパティを指定することになります。これらスタイルプロパティは SPS デザイナーの手により行うことができるほか、Authentic View ユーザーにより定義することもできます。どちらの場合でも、Authentic View ユーザーに対してテキストのスタイルを指定するというオプションが与えられることになります。

以下のテキストスタイルオプションが利用可能です:

- 合成スタイル: CSS スタイルプロパティのセット(合成スタイル)を、ソース XML ドキュメントにある要素の属性にて定義することができます。デザインコンポーネントの合成スタイルプロパティ値として属性をセットすることで、Authentic View のリッチエディット 機能による合成スタイルの編集を行うことが可能になります。このメカニズムに関する詳細については、セクション合成スタイルを参照してください。:
- リッチエディット:SPS デザインにて、ある要素がリッチエディットコンポーネントとして作成された場合、Authentic View のリッチエディットスタイルプロパティを使用して、要素とスタイル内部にてテキストフラグメントを選択することができるようになります。リッチエディットにより、Authentic View ユーザーはフォント、フォントの太さ、フォントスタイル、フォントの装飾、フォントサイズ、色、背景色、そしてテキストの配置を指定することができるようになります。詳細については、リッチエディットセクションを参照してください。
- テキスト状態アイコン: SPS デザイナーは要素名に関連付けられた Authentic View ツールバーボタンを作成することができます。このようなツールバーボタンはテキスト状態アイコンと呼ばれます。Authentic View にてテキストフラグメントが選択された状態でテキスト状態アイコンがクリックされると、テキスト状態アイコンに関連付けられた要素が選択されたテキストの回りに作成されます(選択されたテキストの周りに要素の追加がスキーマにより許されている場合だけテキスト状態アイコンは選択可能になります)。結果として、その要素に対して(グローバルテンプレートにて)定義されたテキストフォーマットが、選択されたテキストフラグメントに対して適用されます。

SPS では、これら3つのテキストスタイルオプションを組み合わせて使用することができます。

以下の点に注意してください:

- これら3つのテキストスタイルオプションを使用するには、XML ドキュメントにてスタイルに関連する要素や属性が必要になります。従って、SPS がベースとしているスキーマにて、必要な要素や属性を、目的の階層レベルにて許可する必要があります。
- Authentic View ユーザーにより適用されたテキストスタイルは Authentic View と出力フォーマットの両方に反映されます。

メモ 上記のスタイルオプションは、XML ファイルにて収められているスタイルプロパティから派生したもので、SPS にて割り当てられたスタイルプロパティとは異なるものです。

関連項目

- CSS スタイルと作業
- CSS サポート

8.5.1 合成スタイル

Authentic View 以下のデザインコンポーネントに対して出力全体でスタイル情報を付与することができるようになります:

(C) 2015-2021 Altova GmbH

- 自動計算
- (コンテンツ)プレースホルダ
- 段落(ブロック)デザイン要素
- テーブルセル

以下にメカニズムを説明します:

メモ 合成スタイルに関するより一般的な情報については、合成スタイルのセクションを参照ください。このセクションでは、Authentic View ユーザーが編集することのできる合成スタイルをセットアップする方法について記述されます。

XML 属性に合成スタイルを入力する

(複数のスタイルプロパティにより構成される)デフォルトの合成スタイルは、ソース XML ドキュメント内にある要素の属性値として入力されます。例えば、以下に示される XML ソースドキュメントの desc-style 属性には、デフォルトの合成スタイルが含まれています:

```
<Desc desc-style="font-family:Verdana; font-size:12pt; color:blue">
```

SPS デザイナーは、これらスタイルプロパティを Authentic View にて編集できるように指定することができます。これにより、デザインコンポーネントのテキスト出力に対して Authentic View ユーザーが合成スタイルを指定できるようになります。デザインコンポーネントの合成スタイルが、デフォルトの合成スタイルを含むようにセットすることで、ユーザーによる編集が可能な合成スタイルを作成することができます。

メモ XML ドキュメントを妥当なものとするために、合成スタイルへアクセスするために使用される属性は、ソーススキーマにて定義する必要があります。

サポートされている CSS テキストスタイルプロパティ

合成スタイルでは、以下の CSS スタイルを使用することができます:

font-family	font-size	font-weight	font-style
color	background-color	text-align	text-decoration

デザインコンポーネントの合成スタイル地として属性をセットアップする

デザインコンポーネントの合成スタイルが属性として定義された場合、Authentic View ユーザーによりこの合成スタイルを編集することができるようになります。Authentic View ユーザーは、デザインコンポーネントのテキスト出力における任意の場所にカーソルを配置して、Authentic View のリッチエディットツールバーからそのデザインコンポーネントの合成スタイルを編集することができるようになります。

以下の操作により、属性をデザインコンポーネントの合成スタイルとしてセットすることができます:

1. デザインビューにて、属性を合成スタイルとして割り当てるデザインコンポーネントを選択します。以下のスクリーンショットでは、Desc 要素の(コンテンツ)プレースホルダが選択されています。

2. コンテンツコンポーネントの合成スタイルプロパティに表示されているコンボボックスに、コンテキスト要素の属性が表示されます（上のスクリーンショットの右下にあるプロパティウィンドウを参照ください）。デザインコンポーネントの合成スタイルとしてセットする属性を選択してください。コンボボックスには空のエントリーも表示されている点に注目してください。合成スタイルを使用しない場合は、空のエントリーを選択してください。この場合、そのデザインコンポーネントの出力に対して、Authentic View におけるリッチエディット機能が無効になります。

Authentic View では、リッチエディットツールバーを使用することで、そのデザインコンポーネントの合成スタイルを修正することができるようになります。

関連項目

- Authentic におけるテキスト スタイルの柔軟性
- CSS スタイルと作業
- 合成スタイル

8.5.2 リッチエディット

SPS デザインにてリッチエディットコンポーネントとして要素が作成された場合、Authentic View のリッチエディットスタイルプロパティを使用することで、Authentic View ユーザーによるその要素内にあるテキストフラグメントのマークとスタイルの作成や、テキストの配置といった段落レベルのフォーマットを行うことができるようになります。

リッチエディットにより、Authentic View ユーザーは以下の項目を指定することができます：

- 文字のスタイル：フォント、フォントの太さ、フォントのスタイル、フォントの装飾、フォントサイズ、色、背景色
- 段落のスタイル：テキストの配置

リッチエディットについて以下で説明します：

- リッチエディットのメカニズム
- SPS デザインにある要素をリッチエディットとして作成する

- Authentic View にてリッチエディットを使用する

リッチエディットのメカニズム

Authentic View にてテキストフラグメントが選択され、リッチエディットスタイルが適用された場合、リッチエディットのスタイル要素と、スタイル情報が含まれる属性が選択されたテキストフラグメントの周りに作成されます。Authentic View ユーザーにより選択されたリッチエディットスタイルプロパティが、このスタイル属性の値として挿入されます。

従って、ソース XML ドキュメントに以下のようなテキストフラグメントが存在し:

```
<p> ... Altova StyleVision 2012 features a unique graphical design interface ...
</p>
```

このテキストフラグメントの一部に太字 (bold) のリッチエディットスタイルプロパティが与えられた場合、ソース XML のテキストフラグメントは以下のようになります:

```
<p> ... <span style="font-weight: bold">Altova StyleVision 2012</span> features a
unique graphical design interface ... </p>
```

上の例におけるリッチエディットのスタイル要素は span で、その中にある style 属性にリッチエディットのスタイル情報が含まれています。スタイル情報を格納する要素や属性の名前は固定されておらず、例えば、span 要素の代わりに Style という要素を、style 属性の代わりに css という属性を使用することもできます。この場合、テキストフラグメントのマークアップは以下のようになります:

```
<p> ... <Style css="font-weight: bold">Altova StyleVision 2012</Style> features a
unique graphical design インターフェイス ... </p>
```

スタイル要素や属性がどのような名前であろうとも、**これらスタイル要素ならびに属性は、スキーマにて定義されている必要があり**、スタイルを適用するテキストを含んでいる全ての要素内にて許可されていなければなりません。

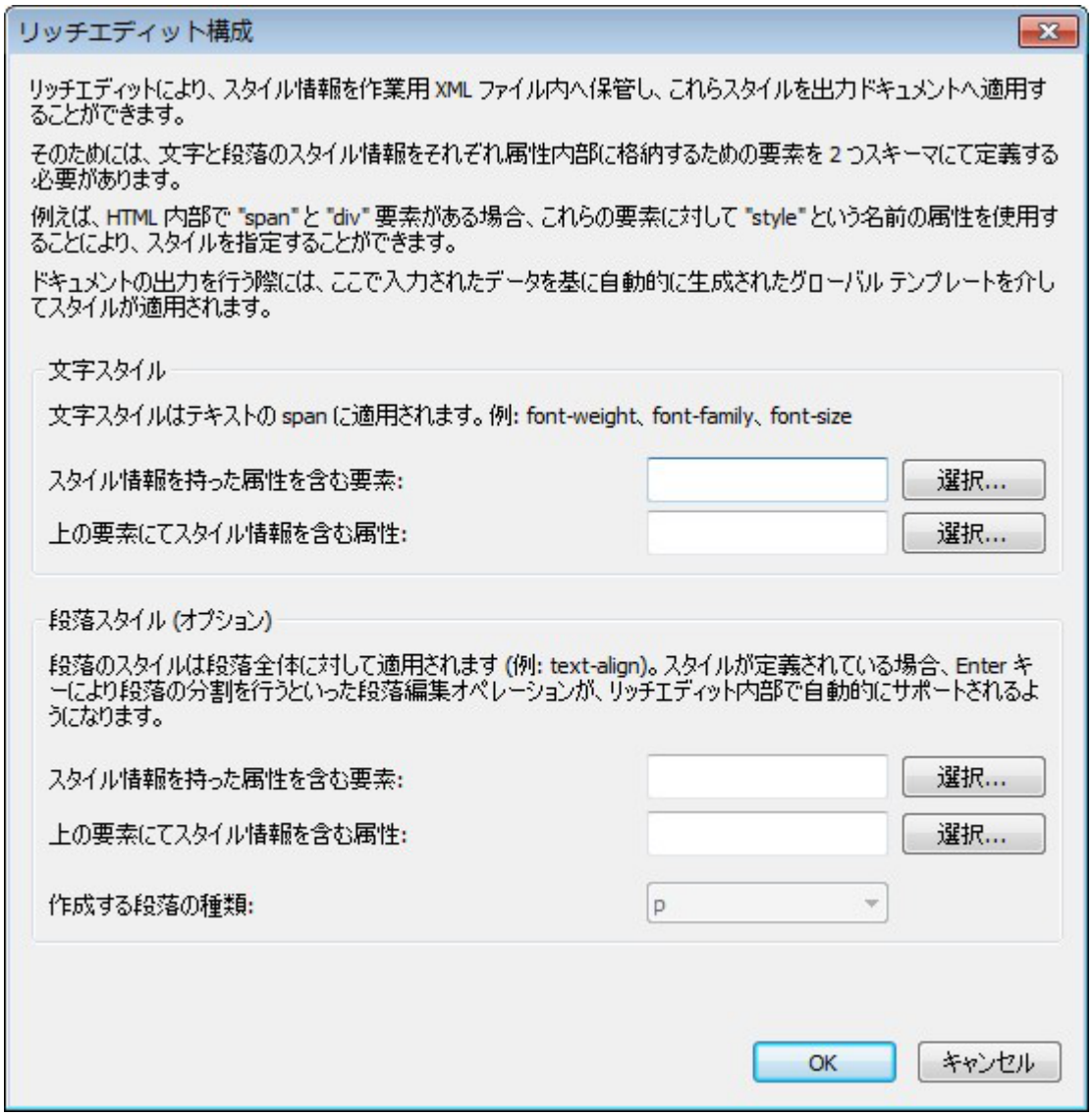
テキストが XSLT スタイルシートにて処理された場合、その出力フォーマットに適したマークアップにてスタイルプロパティが出力へ渡されます。

リッチエディットはブロックレベルのフォーマットに対して適用することもできます(例: テキストの配置)。上記のテキストフラグメントと同様の方法で、ブロックレベルのフォーマットを含む要素ならびに属性を選択してください。リッチエディット構成ダイアログにて段落スタイルが定義された要素を含んだ要素を編集する際に、ブロックレベルのテキスト配置といったブロックレベルのフォーマットプロパティが Authentic View ユーザーに対して利用可能になります。文字のスタイルと同様に、段落のスタイルを適用する要素内部でリッチエディットの段落スタイル要素がスキーマにより許可されている必要があります。

要素をリッチエディットとして作成する

以下の操作により、リッチエディットコンポーネントとして要素を作成することができます:

1. スキーマツリーから要素をドラッグして、デザイン内部の目的の場所にドロップします。
2. 表示されるメニューから**リッチエディットの作成**を選択します。リッチエディットの構成ダイアログが表示されます(以下のスクリーンショットを参照)。ドキュメントに対してリッチエディット文字スタイルが既に作成されている場合、デザイン内のリッチエディットコンポーネントを右クリックして、**リッチエディット要素/属性の設定**を選択する必要があります。



3. リッチエディット構成ダイアログならびに文字スタイルペインにて、リッチエディットスタイルプロパティを含めるスタイル要素とその属性の名前を入力します。**選択**ボタンをクリックすることで、スキーマツリーから目的の要素や属性を選択することもできます。
4. 以下の操作によりブロックレベルのフォーマット（テキストの配置）を有効にすることができます。段落スタイルペインにて、ブロックレベルのフォーマットが含まれる要素と属性を選択します。作成する段落の種類コンボボックスにて、段落に対して使用される定義済みのフォーマットを選択することができます。この定義済みフォーマットが出力へ渡されることになります。
5. 「OK」をクリックして操作を完了します。リッチエディットコンポーネントとして要素が作成され、文字スタイルに対してスタイル要素と同じ名前の**編集不可能**なリッチエディットグローバルテンプレートがデザインに作成されます（以下のスクリーンショットにある Style を参照）。段落スタイルも指定された場合、段落スタイルに対する編集不可能なリッチエディットグローバルテンプレートも作成されます。

以下の点に注意してください:

- リッチエディットコンポーネントが SPS にて初めて作成された際に、編集不可能なリッチエディットグローバルテンプレートが作成されます。その後リッチエディットグローバルテンプレートが新たに作成されることはありません。
- 最初のリッチエディットコンポーネントが作成された後にリッチエディットコンポーネントとして要素が作成された場合には、リッチエディット構成ダイアログは表示されません。リッチエディットコンポーネントは、内部に含まれるリッチエディットのテキストボックスにより表されます(上のスクリーンショットにある chapter 要素を参照)。このリッチエディットテキストボックスは、コンテンツプレースホルダの代わりに表示されます。
- スタイル要素や属性を変更することで、リッチエディットグローバルテンプレートを文字ならびに段落スタイルに対して再構成することができます。リッチエディットコンポーネントを右クリックして、コンテキストメニューから**リッチエディット要素/属性の設定**を選択し、目的の要素/属性名を入力します。(文字ならびに段落スタイルに対する)新たな要素名がリッチエディットグローバルテンプレートのタイトルバーに表示されます。

リッチエディットコンポーネントとして作成された全てのスキーマ要素は、Authentic View のリッチエディットプロパティによりスタイルを指定することができます。

Authentic View にてリッチエディットを使用する

Authentic View にてリッチエディットコンポーネントとして作成された要素内にカーソルが配置されると、リッチエディットツールバーにあるボタンとコントロールが有効になります。リッチエディットコンポーネント以外の場合、これらのアイテムはグレーアウトされます。

文字にスタイルを適用するには、対象となるテキストを選択してください。段落に対してスタイルを適用するには、スタイルを適用する段落内にカーソルを配置してください。その後、リッチエディットツールバーのボタンならびにコントロールにより、目的のスタイルを指定してください。選択されたテキストがスタイル要素のタグに囲まれていない場合、タグにより囲まれることになります。

関連項目

- Authentic におけるテキスト スタイルの柔軟性
- CSS スタイルと作業
- CSS サポート

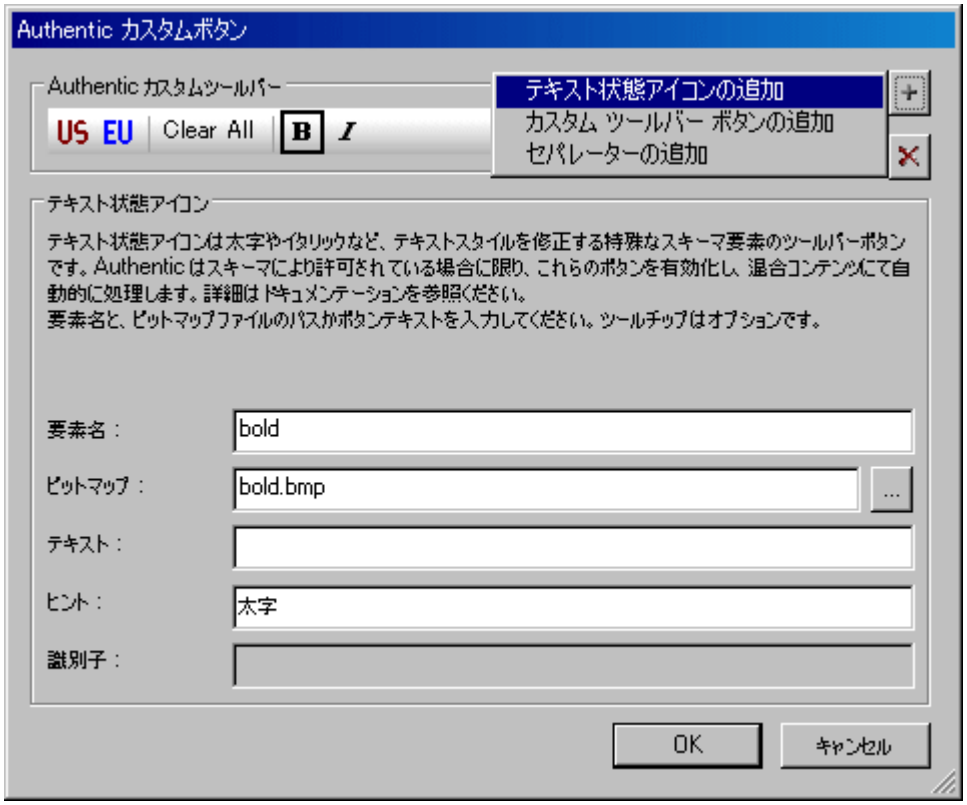
8.5.3 テキスト状態アイコン

テキスト状態アイコンは、要素の名前に関連付けられた Authentic View ツールバーボタンになります。Authentic View にてテキスト状態アイコンがクリックされると、テキスト状態アイコンに割り当てられた要素が、選択されたテキストフラグメントの周りに作成されます。この要素に対して（要素のグローバルテンプレートから）定義されたテキストフォーマットが選択されたテキストフラグメントに対して適用されます。これにより、テキスト状態アイコンを使用したテキストスタイルを Authentic View ユーザーが適用できるようになります。

SPS デザイナーは、テキスト状態アイコンの作成と、テキスト状態アイコンに割り当てられた要素のスタイルを定義することになります。

テキスト状態アイコンの作成

Authentic ツールバーに対するテキスト状態アイコンを作成するには、「**Authentic | ツールバーボタンのカスタマイズ**」コマンドを選択し、Authentic カスタムボタンダイアログの右上にある**追加**ボタンをクリックして、**テキスト状態アイコンの追加**を選択します。

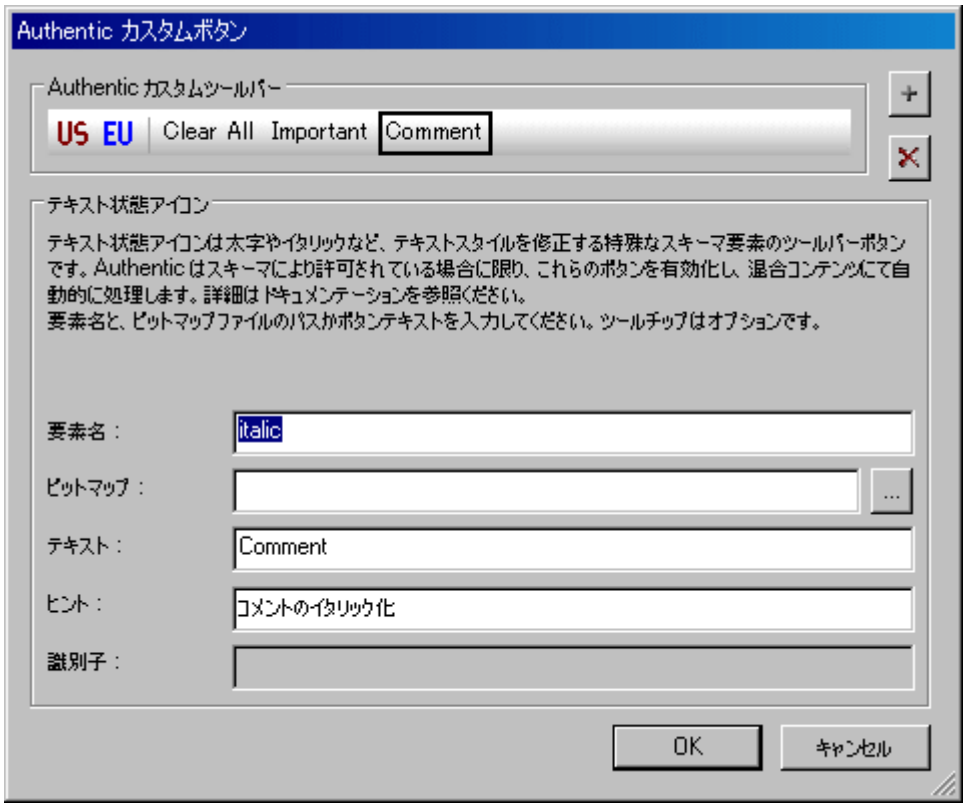


テキスト状態アイコンには以下のパラメーターが用意されています:

- **要素名**: Authentic View にてテキスト状態アイコンがクリックされた時に、選択されたテキストフラグメントを囲む要素です。
- **ビットマップ**: そのテキスト状態アイコンに対して使用されるイメージの場所です。SPS に対する相対パスが表示されます。
- **テキスト**: ビットマップが利用できない場合、このフィールドに入力されたテキストがテキスト状態アイコンのボタンテキストとして使用されます。
- **ヒント**: Authentic View ユーザーがマウスオーバーした際に表示されるガイドです（オプション）。

上の例では、bold と italic 要素を使ったマークアップを行うテキスト状態アイコンに対して、ビットマップのイメージが使用されています。Authentic View がより理解しやすいかたちでビットマップまたはボタンテキストを使用することができます。例えば、重要なテキストフラグメント

にはbold 要素を適用し、コメントにはitalic 要素を使用することができます。以下のスクリーンショットに示されるようにテキスト状態アイコンを作成することができます。



上のスクリーンショットでは、Comment テキスト状態アイコンにitalic 要素に関連付けられている点に注目してください。Authentic View にてテキストが選択されている状態でComment テキスト状態アイコンがクリックされると、そのテキストフラグメントがitalic 要素により囲まれ、italic 要素に対して定義された方法で処理されます（次のセクションを参照ください）

テキスト状態アイコンに関連付けられた要所に対してスタイルルールを定義する

テキスト状態アイコンに関連付けられた要素のスタイルプロパティは、グローバルテンプレートにて定義することができます（以下のスクリーンショットを参照）。

このスクリーンショットには2つの要素（bold とitalic）に対するグローバルテンプレートが示されており、それぞれテキスト状態アイコンに関連付けられています。（コンテンツ）コンポーネントを選択し、目的のフォントスタイルをスタイルサイドバーにて定義することで、スタイルの指定を行うことができます。

これらの要素が Authentic View ユーザーによりテキストフラグメントの周りに作成されると、対応するグローバルテンプレートが（定義されたテキストスタイルとともに）そのテキストフラグメントに対して適用されることになります。.

関連項目

- Authentic におけるテキスト スタイルの柔軟性
- CSS スタイルと作業
- CSS サポート

(C) 2015-2021 Altova GmbH

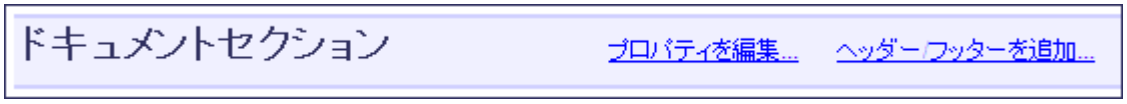
8.6 印刷媒体出力のデザイン

印刷媒体の出力(Enterprise Edition におけるPDF、RTF、Word 2007+ ならびにProfessional Edition におけるRTF)のプロパティは、プロパティサイドバーの**ページレイアウト**プロパティグループにて指定することができます。印刷媒体に対しては以下の項目をデザインすることができます:

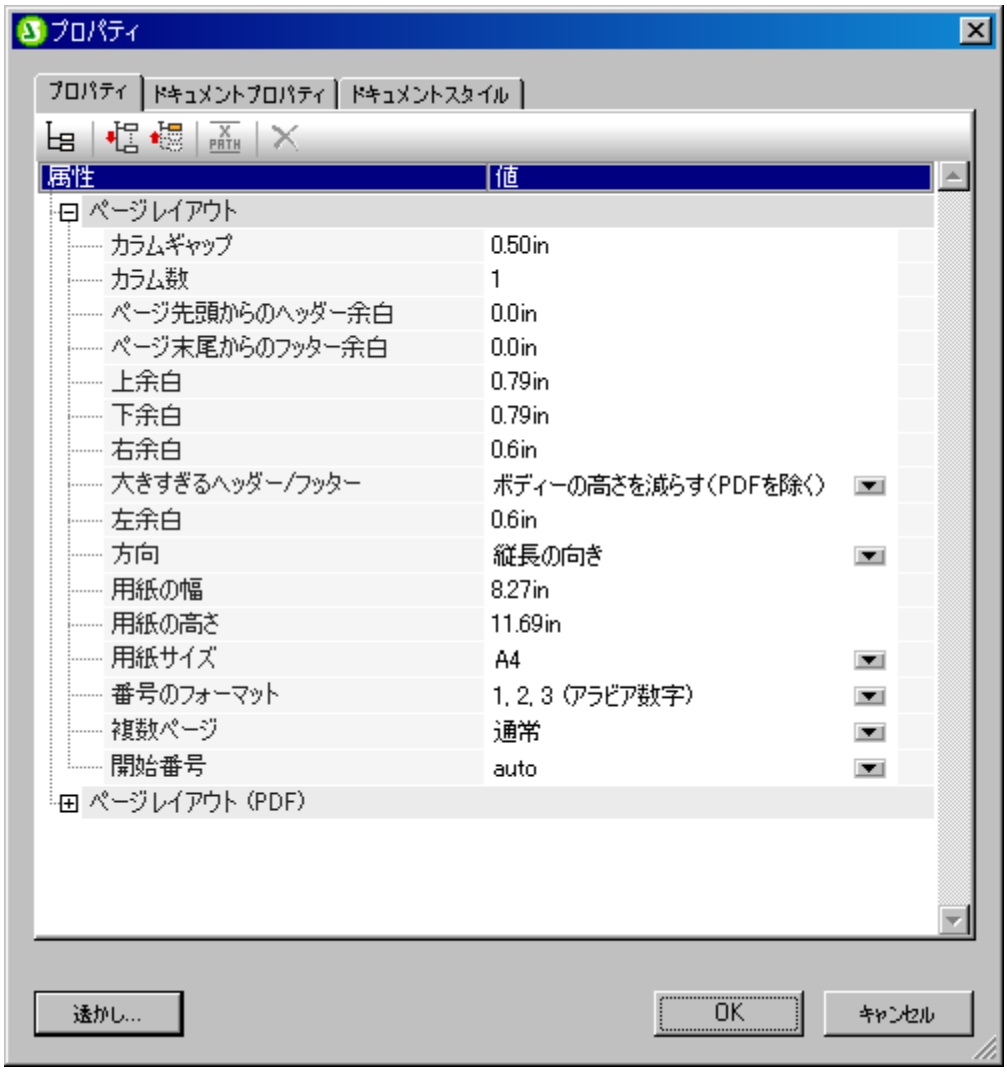
- ドキュメントは複数のセクションに分割することができ、各セクションに異なるページ定義を与えることができます。定義することのできるプロパティを以下に示します。
- ページ寸法(高さと幅)と、ページの方向(縦長か横長か)を定義することができます。
- ページボディに対する余白と、ヘッダーとフッターに与えられる垂直方向の空間を定義することができます。更に、複数のページを見開きとして(つまり余白の設定が奇数と偶数ページで対になるように)指定することができるほか、奇数ページと偶数ページの両方に同じ余白を指定することもできます。
- 各セクションに対してヘッダーとフッターを指定することができます。
- 各セクションに対して個別にページ番号と、開始ページ数を指定することができるほか、セクションをまたいでページ番号を指定することができます。
- 各セクションにて、ページに表示されるカラムの数と幅を指定することができます。

プロパティサイドバー

ページプロパティは、プロパティウィンドウのページレイアウトグループにて、ドキュメントの各セクションに対して個別に定義することができます(以下のスクリーンショットを参照)。各セクションのプロパティへは、イニシャルセクションならびにドキュメントセクションにあるプロパティを編集リンクからアクセスすることができます(以下のスクリーンショットを参照)。



プロパティを編集リンクをクリックすると、プロパティウィンドウが表示されます(以下のスクリーンショットを参照)。



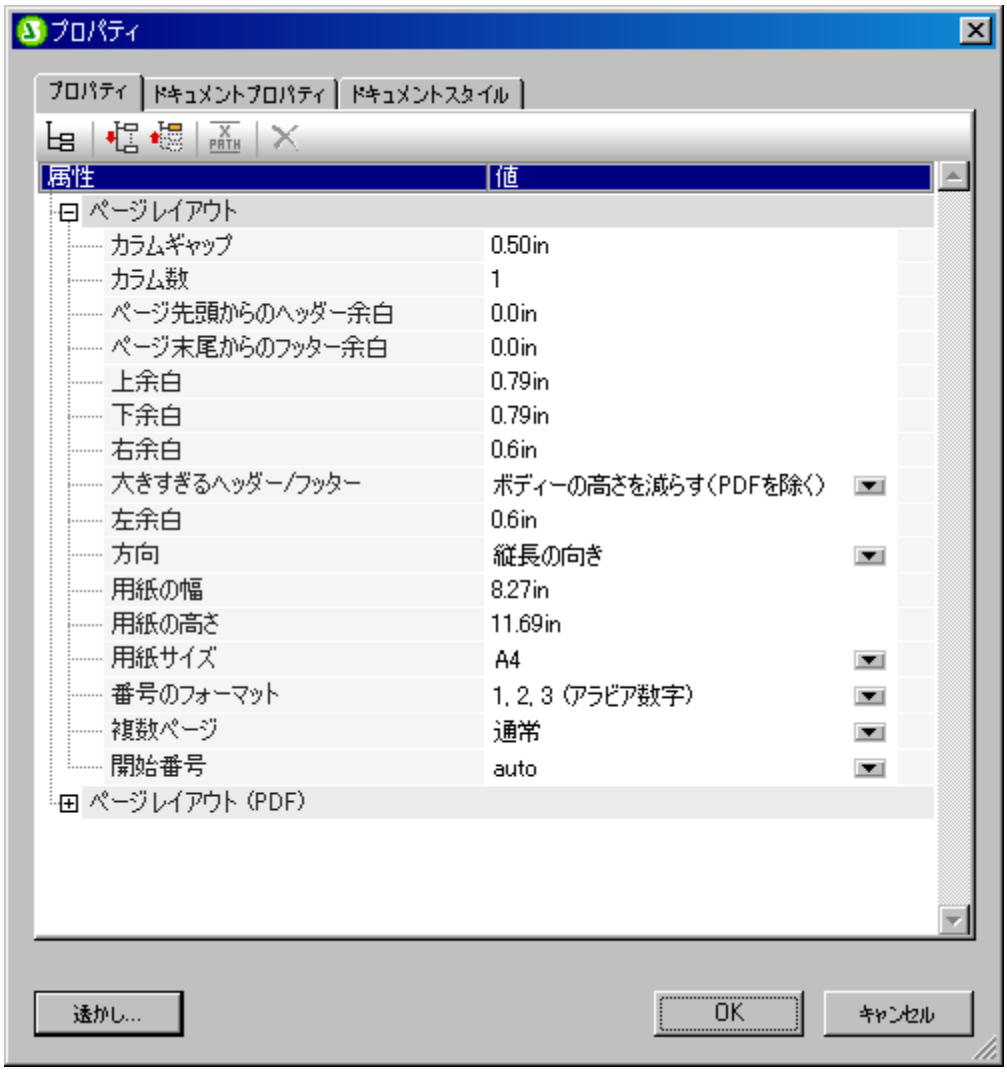
これらプロパティ値をセットする方法については[ページプロパティ](): スタイルの定義方法について記述されます。

関連項目

- [CSS スタイルと作業]()

8.6.1 ドキュメント セクション

SPS には複数のドキュメントセクションを含めることができ、各ドキュメントセクションに独自のページ設定を与えることができます。例えば、テーブルに収められたデータと、そのデータについて説明したテキストを2つのドキュメントセクションとして分割することができます。その他にも、テキストのドキュメントセクションでは縦長の方向で表示が行われ、テーブルのドキュメントセクションでは横長の方向で表示を行うように指定することができます。各ドキュメントセクションに対して[ページプロパティ]()を指定することができます(以下のスクリーンショットを参照)。その他にも、各ドキュメントセクションには独自の[ヘッダーとフッター]()を指定することができます。

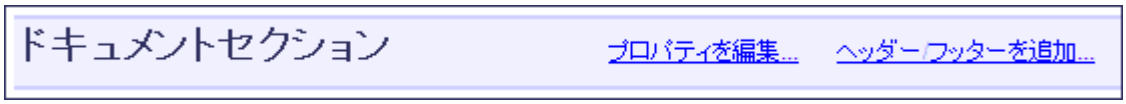


SPS が作成されると、イニシャルドキュメントセクションと呼ばれるドキュメントセクションが1つ作成されます。このドキュメントセクションは、(ドキュメント内にあるセクションが1つか複数あるかに関わらず)ドキュメント最初のセクションとなり、削除することはできません。イニシャルドキュメントセクションのプロパティには、ドキュメント全体に適用されるプロパティとスタイルが含まれています。これらプロパティの詳細については、イニシャルドキュメントセクションのセクションを参照ください。

ドキュメント セクションを挿入する

以下の操作によりドキュメントセクションを挿入することができます:

1. 新たにドキュメントセクションを開始するドキュメント内の場所へカーソルを配置します。
2. (右クリックにより表示される)コンテキストメニューから、「**ページ/カラム/ドキュメントセクションを挿入 | 新規ドキュメントセクション**」を選択します。他にも、「**挿入**」メニューオプションから同様のコマンドを選択することができます。新たなドキュメントセクションがデザイン内に挿入され、ドキュメントセクションタイトルバーが表示されます(以下のスクリーンショットを参照 – ヘッダー/フッターを隠すリンクは、ドキュメントセクションにヘッダーまたはフッターが追加された状態で表示されます)。出力では、ドキュメントセクションごとに新たなページが表示されます。



3. 新規ドキュメントセクションには、ドキュメントセクションが作成された段階のイニシャルドキュメントセクションに割り当てられたページレイアウトプロパティが含まれます。ドキュメントセクションに与えられた ページレイアウトプロパティ は、**プロパティを編集** ハイパーリンクから編集することができます。必要な場合は、 ヘッダーとフッター をドキュメントセクションに追加することもできます(**ヘッダー/フッターを追加** ハイパーリンクを使用)。 ページレイアウトプロパティ と ヘッダーとフッター の定義方法については、このセクション以下にあるサブセクションにて記述されます。ヘッダーやフッターはドキュメントセクション内部に表示されます。デザインにおけるヘッダーとフッターの表示は、**ヘッダー/フッターを隠す** ハイパーリンクにより切り替えることができます。このハイパーリンクは、ヘッダーまたはフッターがドキュメントセクションに追加された後に表示されます。.
4. イニシャルドキュメントセクションの後に新規のページを作成する場合、イニシャルドキュメントセクションのページレイアウトの定義に移動し、セクション区切りを表示するプロパティの値を「*Yes*」に設定します。これにより、イニシャルドキュメントセクションではない最初のページが新規のページとして開始されます。プロパティの値が「*No*」の場合、効果が変更されることに注意してください。この場合、最初の非イニシャルドキュメントセクションは、イニシャルドキュメントセクションの直後に改ページを表示することなく開始されます。イニシャルドキュメントの設定が空である場合は、役に立ちます。例えば、出力を生成しないデザイン処理テンプレートのみを含む場合など。「 *No* 」のプロパティの値は、出力ドキュメントの最初のページが空にならないようにします(これを行わないと改ページを含む空のページがイニシャルドキュメントの後に表示されます)。

以下の点に注意してください:

- XSLT 1.0 が選択された SPS により生成された RTF 出力では、メインテンプレート直下のドキュメントセクションだけが許可されます。この制約は、XSLT 2.0 または XSLT 3.0 が選択された SPS から生成された RTF 出力には当てはまりません。
- 出力ドキュメントでは、ドキュメントセクションごとに新たなページが作成されます。
- ページ余白プロパティは HTML ページに対しても適用されます。
- 複数のドキュメントセクションがデザイン内に存在する場合、ミラー余白プロパティの値と、対応する左余白ならびに右余白プロパティの値はイニシャルドキュメントセクションから取得されます。その他ドキュメントセクションにて指定されたこれらプロパティの値は無視されます。

ドキュメントセクションの削除

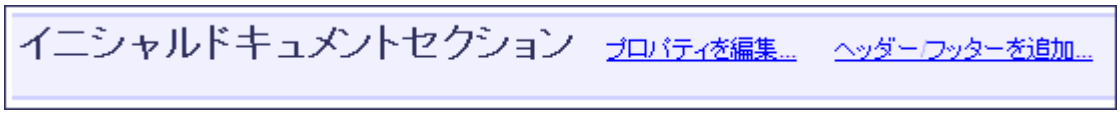
ドキュメントセクションを削除するには、ドキュメントセクションのバーにて**ドキュメントセクション** という文字を右クリックし、コンテキストメニューから「**編集 | 削除**」を選択します。ドキュメントセクションのタイトルバーとともに、そのドキュメントセクションが削除されます。ドキュメントセクションを削除すると、そのドキュメントセクションに対して作成されたページレイアウトプロパティ、ヘッダー、そしてフッターも削除されますが、ドキュメントセクションのコンテンツは削除されません

関連項目

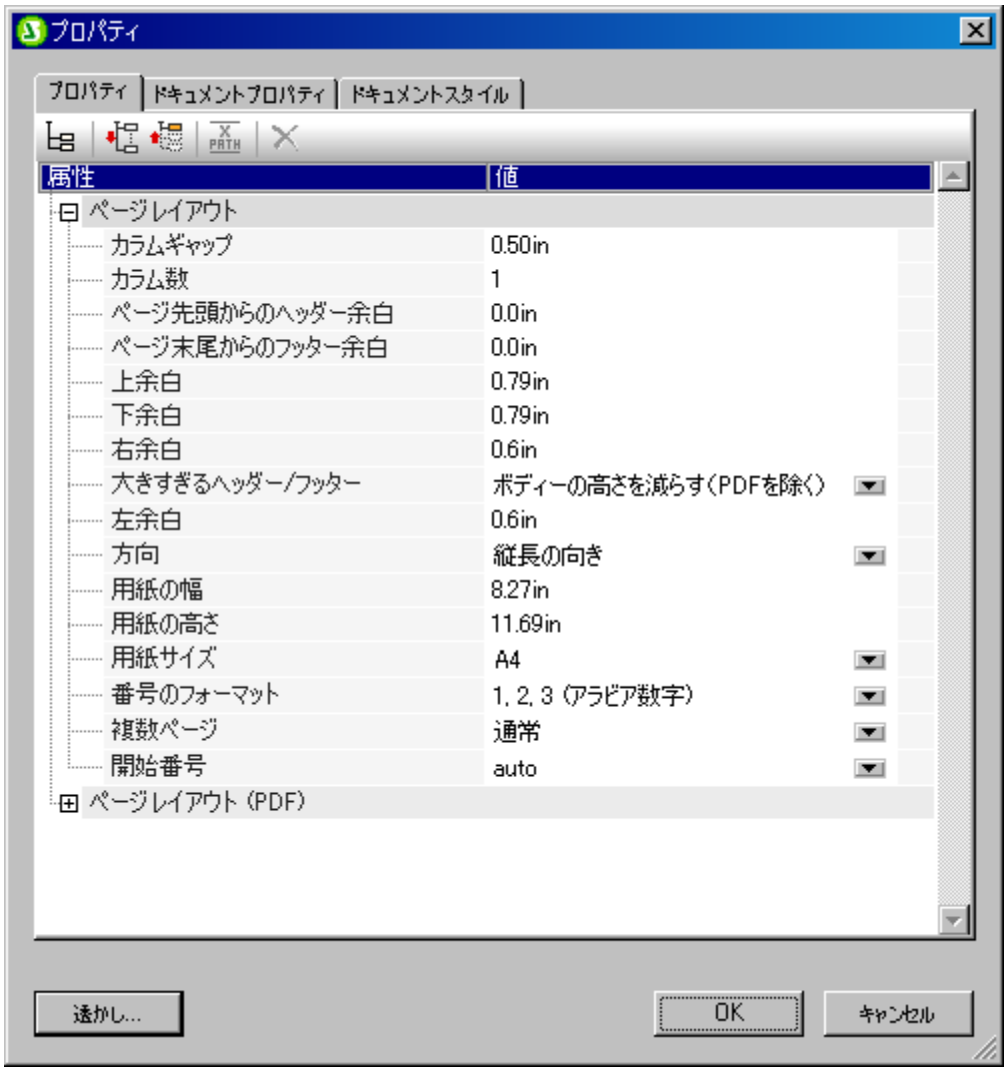
- イニシャルドキュメントセクション
- ページレイアウト プロパティ
- ヘッダーとフッター
- ドキュメント プロパティとスタイル

8.6.1.1 イニシャルドキュメントセクション

その他のドキュメントセクションが存在するかどうかに関わらず、ドキュメント全体のプロパティはイニシャルドキュメントセクションにて定義されます(ドキュメントの最初に表示されるドキュメントセクション - 以下のスクリーンショットを参照)。イニシャルドキュメントセクションにより 表紙 の作成を行うこともできます。



ドキュメントのプロパティを編集するには、イニシャルドキュメントセクションにあるプロパティの編集ハイパーリンクをクリックしてください。イニシャルドキュメントセクションのプロパティダイアログが表示されます(以下のスクリーンショットを参照)。このダイアログには、(i) 基本的なページレイアウトプロパティ、(ii) (HTML) ドキュメントプロパティ、(iii) ドキュメントスタイルという3つのタブが含まれています。



ページレイアウトプロパティ

イニシャルドキュメントセクションのページレイアウトプロパティは、ドキュメント内にある最初のドキュメントセクションに対して適用されます。ドキュメントセクションが1つしか無い場合、これらの設定がドキュメント全体に対して適用されます。ドキュメントセクションが新たに作成されると、その時点でイニシャルドキュメントセクションにあるページレイアウトプロパティの値が使用されます(作成されたドキュメントセクションのプロパティ値は後から編集することができます)。イニシャルドキュメントセクションのプロパティが変更されても、既に作成されたドキュメントセクションのプロパティに対して、これら変更が反映されることはありません。新規ドキュメントセクションの作成には、最新の(つまりその時点での)イニシャルドキュメントセクションにある値が使用されます。個々のページレイアウトプロパティについては、ページレイアウトプロパティのセクションを参照ください。透かしボタンにより、セクションのページのために透かしを定義することができる透かしダイアログが表示されます。それぞれのセクションに対して異なる透かしを定義することができます。

ドキュメントプロパティ

出力 HTML ドキュメントのプロパティは、ドキュメントプロパティタブにて指定されます。

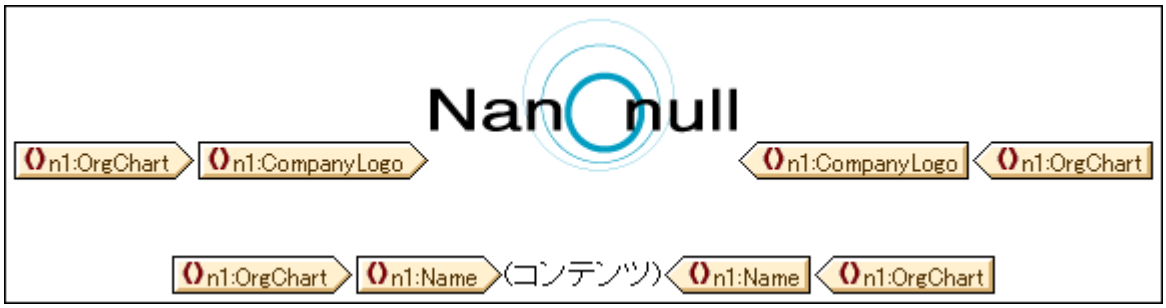
ドキュメント スタイル

ドキュメントスタイルタブにて定義されたスタイルは、ドキュメント全体に対して適用されます。ドキュメント内に複数のドキュメントセクションが定義されている場合、各ドキュメントセクション内にあるデザイン要素がイニシャルドキュメントセクションのプロパティ値を継承することになります。デザイン要素にて継承されたスタイルをオーバーライドするには、個々のデザイン要素にてスタイル値を指定する必要があります。デザイン要素が選択された状態で、スタイルサイドバーにて目的のスタイルを指定してください。

表紙

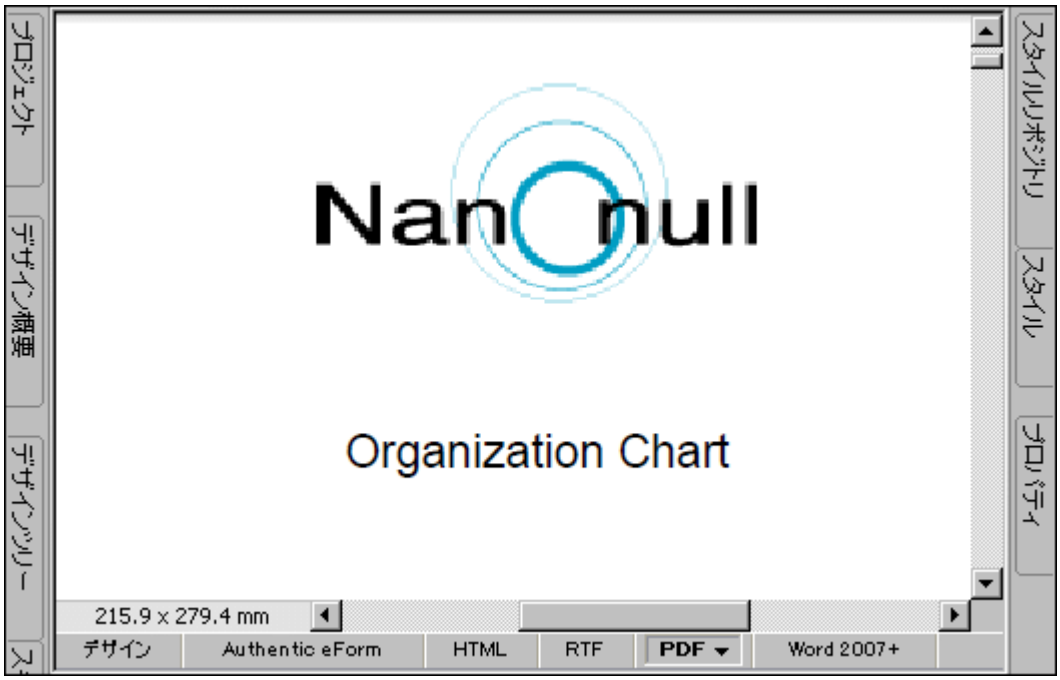
表紙を作成するには、イニシャルドキュメントセクションの最初の方でデザインを行う必要があります。表紙テンプレートの直後に改ページを（「**挿入 | ページ/カラム/ドキュメントセクションを挿入 | 新規ページ**」メニューオプションにて）挿入することで、その他のドキュメントが別ページにて表示されるようにすることができます。表紙のページレイアウトプロパティにある値がその他のページにおける値と異なる場合、イニシャルドキュメントセクション全体を表紙ページとして使用して、それ以降のページを別のドキュメントセクションにより作成する必要があります。

以下に表紙デザインの例を示します:



プレビュー RTF タブをクリックして、プレビューウィンドウにて結果を確認します。

(C) 2015-2021 Altova GmbH



関連項目

- ドキュメント セクション
- ページレイアウト プロパティ
- ヘッダーとフッター
- 改ページ

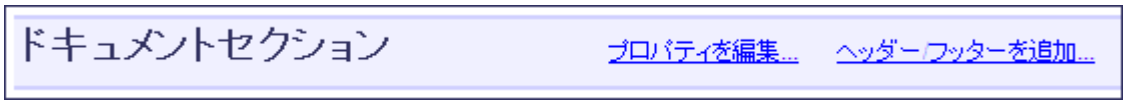
8.6.1.2 ページレイアウト プロパティ

ページプロパティは、ドキュメントセクションのプロパティにあるページレイアウトグループにより、個々のドキュメントセクションに対して個別に割り当てられます。デザイン内にドキュメントセクションが1つしか存在しない場合、そのドキュメントセクションにおけるページレイアウトプロパティが、ドキュメント全体のページプロパティとなります。

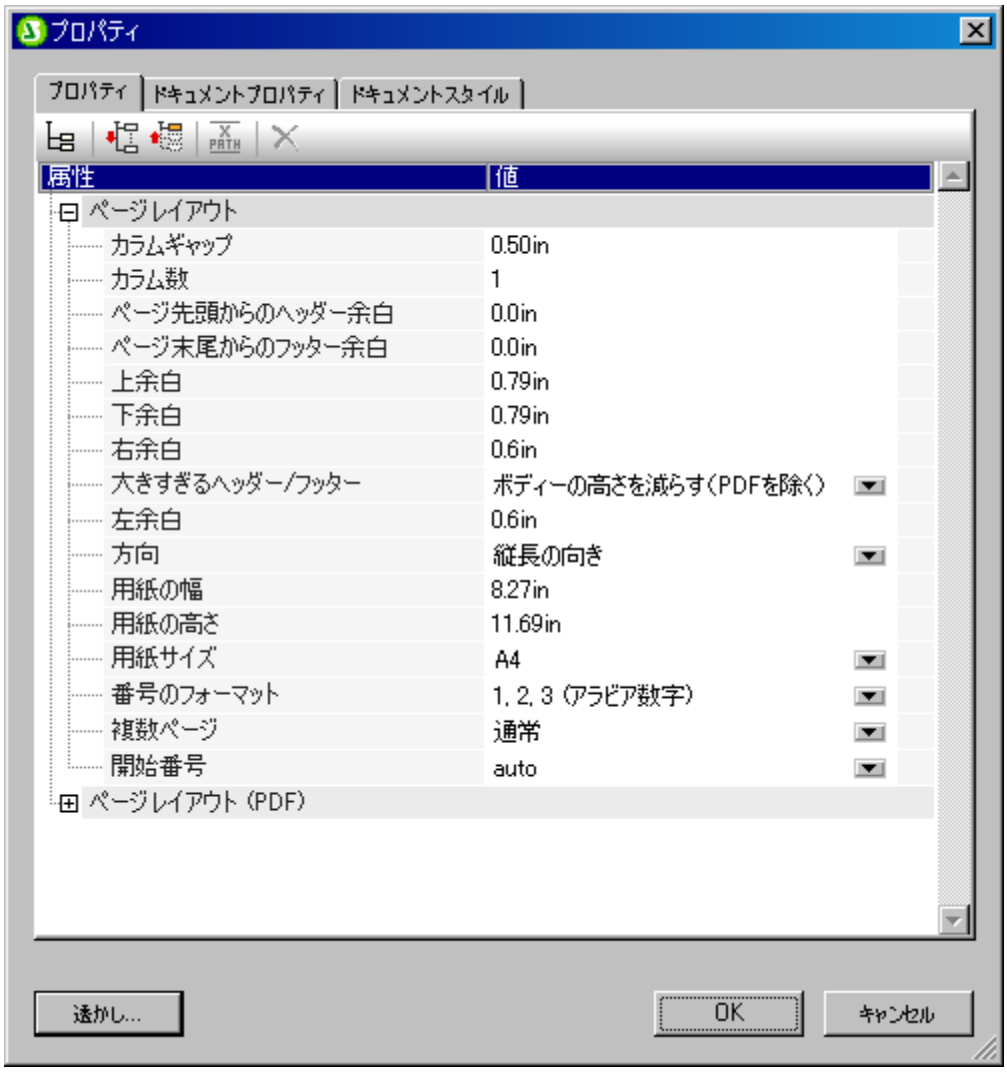
メモ FOP の制限により、出力を生成される際にFOP が使用される場合ページレイアウトプロパティはイニシャルドキュメントセクションでのみ使用することができます。

ドキュメントセクションのページプロパティへアクセスする

ドキュメントセクションのページプロパティへアクセスするには、デザインのイニシャルドキュメントセクション または ドキュメント セクションアイテムアイテムにあるプロパティを編集リンクをクリックします(以下のスクリーンショットを参照)。



この操作によりページレイアウトプロパティを含むプロパティウィドウが表示されます(以下のスクリーンショットを参照)。



この操作によりページレイアウトプロパティを含むプロパティウィンドウが表示されます(以下のスクリーンショットを参照)。

ページサイズ

ドキュメントセクションのページサイズは、(i) 用紙の高さ、(ii) 用紙の幅、(iii) 用紙サイズという3つのプロパティにより決定されます。ページサイズは以下にあるいずれかの方法によりセットすることができます:

- **用紙サイズ**プロパティフィールドのコンボボックスから定義済みのページサイズを選択することができます。この場合、コンボボックスで選択された値に従って、**用紙の高さ**と**用紙の幅**フィールドに値が自動的に入力されます。
- **用紙の高さ**と**用紙の幅**プロパティに独自の値を入力することもできます。この場合、用紙サイズフィールドの値は**カスタムサイズ**となります。

用紙の高さならびに**用紙の幅**プロパティの値として入力することのできる単位は、インチ (in)、センチメートル (cm)、ミリメートル (mm)、パイカ (pc)、ポイント(pt)、ピクセル(px)、ems (em) となります。(i) 単位も一緒に記述する必要がある、(ii) 数値と単位の間にスペースを入れてはいけない、(iii) デフォルトとなる単位は無い、ということに注意してください。無効な単位が入力されると、プロパティの値が赤色で表示されます。

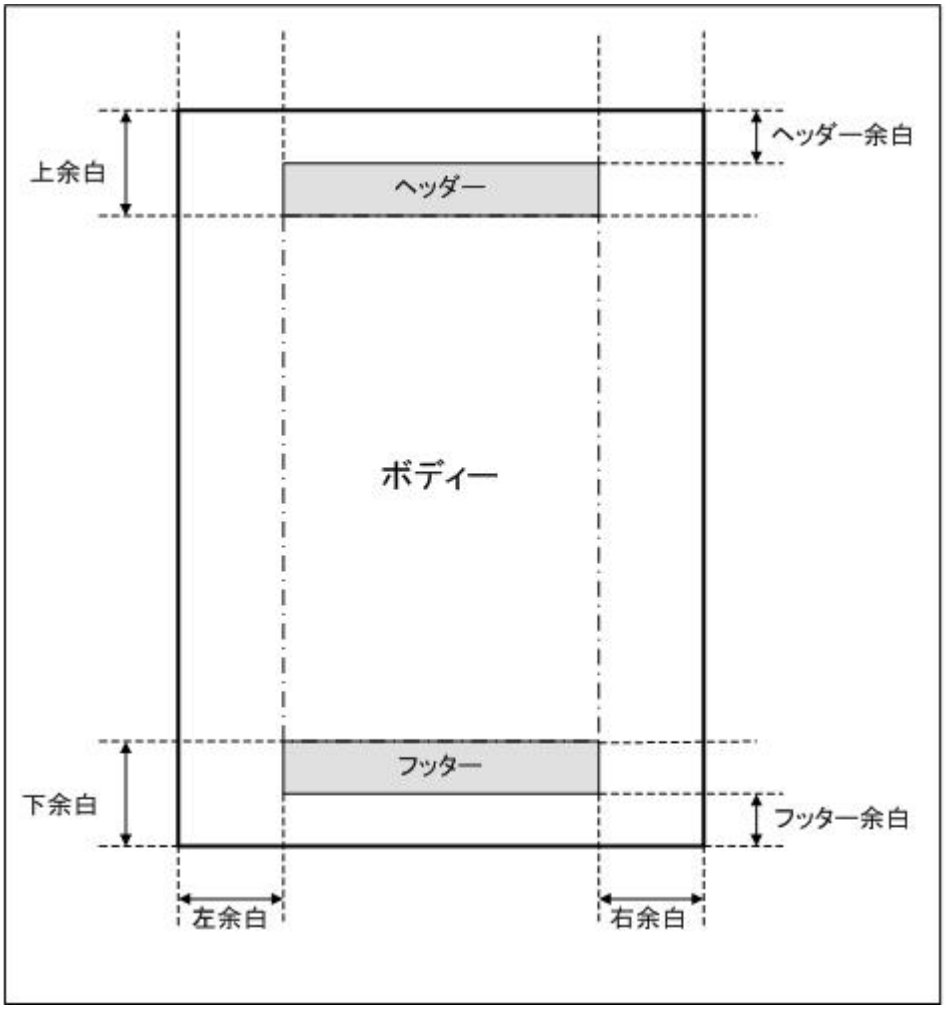
ページ余白

ページ上部、下部、左、右の余白は、それぞれ対応する余白プロパティ(**上余白**、**下余白**、**左余白**、**右余白**)にて定義されます。プロパティフィールドにて値と単位を入力することで、余白を指定することができます。単位にはインチ (in)、センチメートル (cm)、ミリメートル (mm)、パイカ (pc)、ポイント(pt)、ピクセル(px)、ems (em) を使用することができます。(i) 単位も一緒に記述する必要がある、(ii) 数値と単位の間にスペースを入れてはいけない、(iii) デフォルトとなる単位は無い、ということに注意してください。無効な単位が入力されると、プロパティの値が赤色で表示されます。

指定された余白により、ボディーエリアが決定されます(以下の図を参照)。

ヘッダーとフッターの余白

ページ先頭からのヘッダー余白と**ページ末尾からのフッター余白**プロパティにより、ページ先頭からヘッダー上部までの余白と、ページ末尾からフッター下部までの余白がそれぞれ指定されます(以下の図を参照)。



ヘッダーとフッターの高さは、ヘッダーとフッターに含まれるコンテンツにより決定されます。ヘッダーと**ヘッダー余白**の合計が上余白にて指定された値を超えないように注意してください。もし超えてしまう場合、ヘッダーは定義された空間に収まらないということになります。同様に、フッターと**フッター余白**の合計が下余白の値を超えないように注意してください。

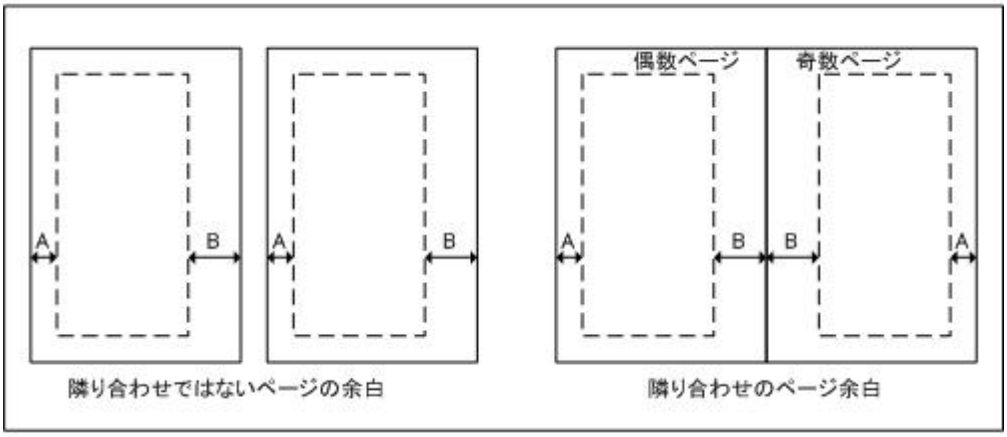
表示されるヘッダーやフッターが割り当てられた空間には収まらない場合、**大きすぎるヘッダー/フッター**プロパティにより RTF 出力におけるヘッダーやフッターの修正を行うことができます。大きすぎるヘッダー/フッタープロパティに**ボディーテキストを重ねる**オプションが選択

された場合、ヘッダー/フッターの内容が、ボディーに重ね合わさるかたちで表示されます。**ボディーの高さを減らす**オプションが選択された場合、ヘッダー/フッターを表示するためにボディーの大きさが削られることになります。

複数のページ設定

複数ページプロパティには2つのオプションが用意されています

- **複数ページ**プロパティに**通常**を指定すると、全ページに対して同一の左余白と右余白が与えられることになります（以下のスクリーンショットを参照）。
- それとは別に、複数ページに**ミラー余白**を指定すると、袋とじとしてドキュメントのページが扱われます（以下のスクリーンショットを参照）。つまり、偶数番号のページ（左側のページ）と、奇数番号のページ（右側のページ）における余白の数値に、向かい合うかたちで同じ値が使用されることになります。



左余白ならびに右余白プロパティに指定された値は、StyleVision により奇数番号のページに対して適用され、それとは逆の値が偶数番号のページに対して適用されます。つまり、奇数番号のページにおける内側の余白が、偶数番号のページにおける外側の余白となります。

メモ SPS デザインに複数のドキュメントセクションが存在する場合、ミラー余白設定の値はイニシャルドキュメントセクションから取得され、左余白と右余白の値も、同様にイニシャルドキュメントセクションから取得されます。その他のドキュメントセクションにあるこれらプロパティの値は無視されることになります。

ページの方向

ページの方向を**縦長**と**横長**から選択することができます。

カラム数

カラムとその幅は、**カラム数**と**カラムギャップ**という2つのプロパティより指定されます。これら2つのプロパティにより、カラムの数と、カラム間におけるスペースがそれぞれ指定されます。従って、ページボディの幅からカラムギャップの合計を引き、カラムの数で割った値がカラムの幅となります。

テキストは、ページ上にあるカラムに順序良く表示されることになります。全てのカラムにコンテンツが表示されたところで新しいページが開始されることになります。デザイン上の希望する位置に新たなカラムを挿入することで、カラム区切りが行われます。カラム区切りを作成するには、目的の場所を右クリックして、コンテキストメニューから「**ページ/カラム/ドキュメントセクションを挿入 | 新規カラム**」を選択してください。

ページ番号

ページ番号の設定は、**番号のフォーマット**と**開始番号**プロパティにて指定することができます。これらのプロパティは以下のように使用されます:

- **番号のフォーマット**プロパティにあるドロップダウンメニューから選択することのできるオプションにより、ページ番号のフォーマットが決定されます(**番号のフォーマット**にて選択された値は ページの合計 にも適用されます)。
- 開始番号プロパティでは、ドキュメントセクションに表示される最初のページ番号を、0 以上の整数でセットすることができます。それまでのドキュメントセクションにあるページ番号を継続して使用する場合、プロパティの値を空白にするか、auto をセットする必要があります。

メモ メニューオプションから「**挿入 | ページ/カラム/ドキュメントセクションを挿入 | ページ番号**」を選択することで、ページ番号のプレースホルダが挿入されます。メニューオプションの「**挿入 | ページ/カラム/ドキュメントセクションを挿入 | 合計ページ**」コマンドにより、合計ページ数を出力ドキュメントへ挿入することができます。

メモ MS Word ではページ番号が自動的に更新されないことがあります。Ctrl+A F9 により、ページ番号を手動で更新することができます。

RTF 出力におけるページ番号

MS W ord に開かれる RTF 出力にてページ番号を表示するには、MS W ord にて(「**選択 | すべて選択**」または Ctrl+A により)ドキュメント全体のコンテンツを選択して、F9 キーを押下する必要があります。この操作により、ページ番号が表示されるようになります。

合計ページ数

ドキュメントにて合計ページ数を表示するには、 合計ページ コマンドを使用してください。

ドキュメント セクションの開始

イニシャルドキュメントセクション以外のドキュメントセクションでは、**セクションの開始**プロパティによりセクションが(偶数ページか奇数ページに関係なく)次のページから開始されるのか、偶数または奇数ページから開始されるのかを指定することができます。例えば、直前のドキュメントセクションが奇数ページで終わり、現在のドキュメントセクションが奇数ページから開始されるように指定されている場合、直前のセクションに続く偶数ページが空白となります。そのページが偶数ページまたは奇数ページになるかはドキュメントにより決定され、ユーザーにより指定されたページ番号とは関係無い点に注意してください。

イニシャルドキュメントセクションの改行

新しいページ内でイニシャルドキュメントセクションのすぐ後でセクションを開始する場合、イニシャルドキュメントセクションのページレイアウトのプロパティに移動し、セクション区切りを表示するプロパティの値を「Yes」に設定します。これにより、(イニシャルドキュメントセクションの後の最初のセクションである)2番目のドキュメントセクションが新しいページに表示されます。このプロパティはイニシャルドキュメントセクションのみのため使用することができ他のセクションでは使用することができません。

プロパティの値が「No」の場合、効果が変更されることに注意してください。この場合、最初の非イニシャルドキュメントセクションは、イニシャルドキュメントセクションの直後に改ページを表示することなく開始されます。イニシャルドキュメントの設定が空である場合は、役に立ちます。例えば、出力を生成しないデザイン処理テンプレートのみを含む場合など。「 No 」のプロパティの値は、出力ドキュメントの最初のページが空にならないようにします(これを行わないと改ページを含む空のページがイニシャルドキュメントの後に表示されます)。

関連項目

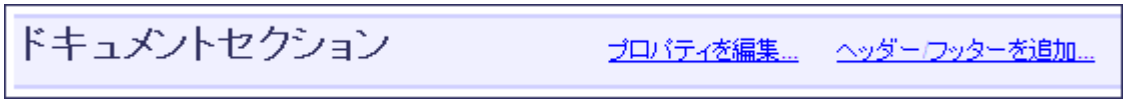
- ドキュメント セクション
- ドキュメント プロパティとスタイル
- ヘッダーとフッター
- 改ページ

8.6.1.3 ヘッダーとフッター: パート1

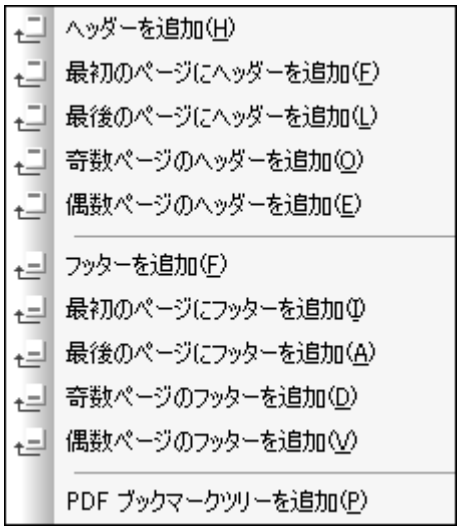
ヘッダーとフッターは、イニシャルドキュメントセクションを含む各ドキュメントセクションに対して追加することができます。

ドキュメントセクションに対してヘッダーやフッターを追加する

ドキュメントセクションに対してヘッダーまたはフッターを追加するには、イニシャルドキュメントセクションのタイトルバーまたはドキュメントセクションタイトルバーにあるヘッダー/フッターを追加リンクをクリックします(以下のスクリーンショットを参照)。



表示されるメニューから、目的のアイテムを選択してください(以下のスクリーンショットを参照)。ヘッダーとフッターは偶数ならびに奇数ページに対して個別に追加することができるほか、全てのページに対して同一のヘッダー/フッターを追加することもできます。更に、ドキュメントセクションにおける最初のページと、最後のページに対して、ヘッダー/フッターテンプレートを個別に作成することもできます(**最初/最後のページにヘッダー/フッターを追加** - PDF 出力でのみ有効)。例えば、ドキュメントセクションで表示される最初のページが表紙となる場合に、空のヘッダー/フッターテンプレートを作成することができます。

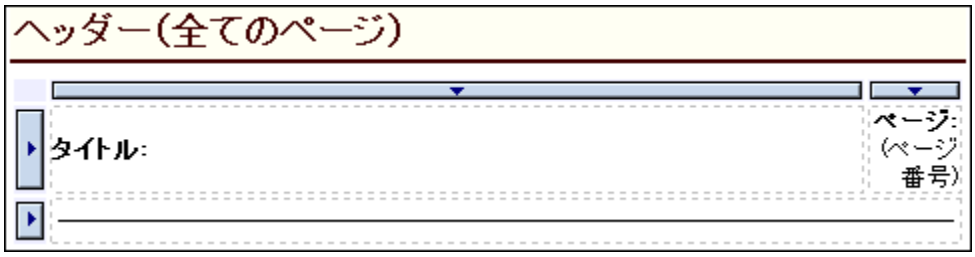


目的のヘッダーやフッターをクリックすることで、デザインビューのドキュメントセクション内にヘッダーまたはフッターのテンプレートが作成されます。イニシャルドキュメントセクション または ドキュメント セクションアイテムのツールバーにあるヘッダー/フッターを隠すリンクをクリックすることで、デザインビューにおけるドキュメントセクションのヘッダー/フッターの表示を切り替えることができます(上のスクリーンショットを参照)。

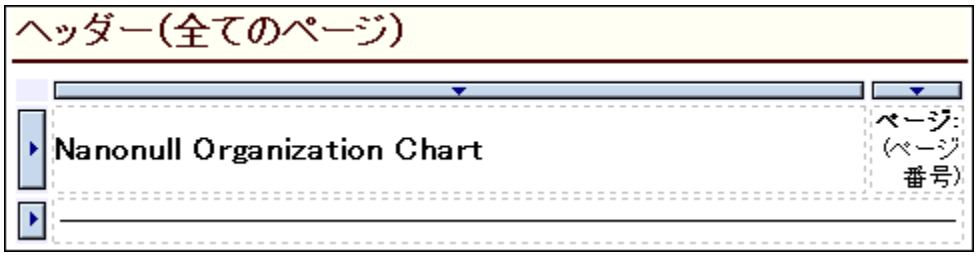
メモ 最後のページにおけるヘッダーならびにフッターは PDF 出力でのみサポートされます。

デザインビューにてヘッダー/フッターをデザインする

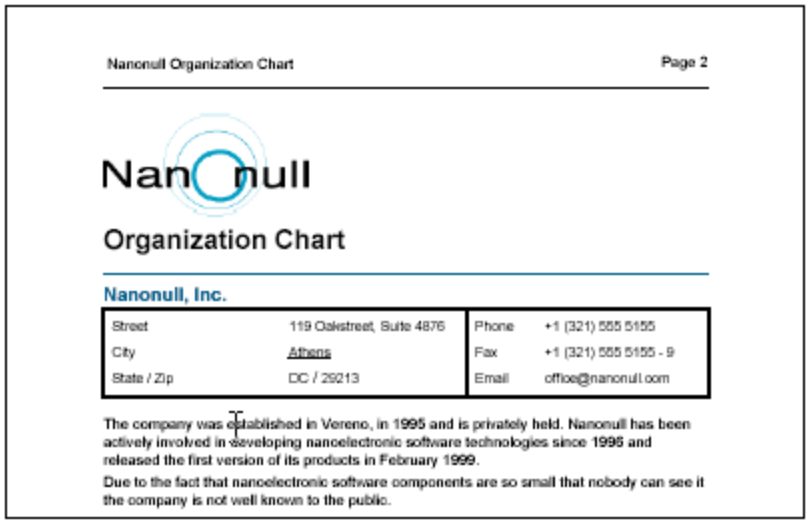
ヘッダー/フッターテンプレートのデザインは、その他のテンプレートと同様の方法で行われます。スキーマツリーからコンポーネントをドラッグし、静的なテキストを入力して、スタイルの設定を定義することができます。ヘッダーが挿入された時のテンプレートの例を以下に示します。



ヘッダーを修正します。静的ならびに動的なコンテンツや、イメージを使用することができます。



プレビューRTF のプレビュータブをクリックすることで、出力結果をプレビューウィンドウにて確認することができます。以下のイメージは組織チャートドキュメントの2ページ目をPDF プレビューウィンドウにて表示したもので、上にある定義に従ってヘッダーが表示されています。RTF 出力にてページ番号を表示するには、「選択 | すべて選択」(またはCtrl+A)によりドキュメント全体のコンテンツを選択して、F9 キーを押下する必要があります。次も参照してください: RTF 出力にてページ番号を表示する



以下の点に注意してください:

- ヘッダーならびにフッターの高さが、上余白(または下余白)からヘッダー余白(またはフッター余白)を引いた値を超えない様にしてください(詳細については、ページレイアウト プロパティ を参照ください)。ヘッダーやフッターの高さは、上余白/下余白とヘッダー余白/フッター余白により決定されます。
- ヘッダー/フッターの定義は、(i) ドキュメントセクションにおける全てのページ、または(ii) ドキュメントセクションにおける奇数または偶数ページに対して個別に行うことができます。更に、最初のページと最後のページに対して独立したヘッダー/フッターを挿入することもできます(詳細についてはヘッダーとフッター: Part 2 を参照ください)。
- ドキュメントセクションのページ番号は、ページレイアウト プロパティで指定した数値から開始されます。

ヘッダーまたはフッターの削除

ヘッダーまたはフッターを削除するには、ヘッダー/フッタータイトルバーを右クリックして、コンテキストメニューから「**編集 | 削除**」を選択してください。

関連項目

- ドキュメント セクション
- ページレイアウト プロパティ
- 改ページ

8.6.1.4 ヘッダーとフッター: パート 2

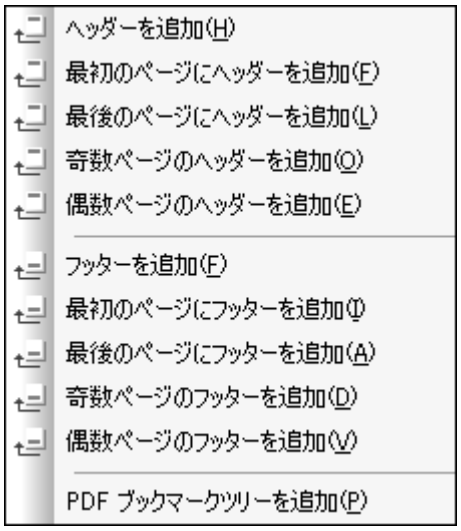
このセクションでは、以下にあるヘッダーとフッターを作成する方法について説明します:

- 偶数ページと奇数ページで異なるヘッダー/フッター
- ドキュメントセクションごとに異なるヘッダー/フッター
- ペイン内部にてヘッダー/フッターのシミュレート
- ヘッダー/フッター

偶数ページと奇数ページで異なるヘッダー/フッターを割り当てる

ドキュメントセクションごとに、偶数ページと奇数ページに対して異なるヘッダー/フッターを割り当てることができます。

偶数ページと奇数ページに対して異なるヘッダーを作成するには、ドキュメントセクションのタイトルバーにある**ヘッダー/フッターを追加**リンクをクリックして、表示されるメニューから**奇数ページのヘッダーを追加**と**偶数ページのヘッダーを追加**を選択します(以下のスクリーンショットを参照)。これにより、奇数ページと偶数ページのヘッダーに対応するテンプレートが2つ作成されます。それぞれのテンプレートにコンテンツを入力してください。



奇数ページと偶数ページに対応したフッターも、上記と同様の方法で作成することができます。

ドキュメント セクションごとに異なるヘッダー/フッターを割り当てる

ヘッダー/フッターは、ドキュメント内のドキュメントセクションごとに作成することができます。イニシャルドキュメントセクションのタイトルバーまたはドキュメントセクションタイトルバーにあるヘッダー/フッターを追加リンクをクリックしてください。上のスクリーンショットに示されているヘッダー/フッターを追加メニューが表示されます。以下の点に注意してください：:

- 奇数ページと偶数ページのヘッダー/フッターを個別に追加することができるほか、ドキュメントセクション内にある全てのページに対して共通したヘッダー/フッターを追加することができます。
- 更に、最初のページ/最後のページのヘッダーやフッターを追加することもできます。これらヘッダー/フッターは、対応するページにて、ドキュメントセクション全体に対して定義されたその他のヘッダー/フッターの代わりに使用されます。
- ドキュメントセクションのページ番号は、直前のドキュメントセクションから継続して、または指定された番号からはじめることができます。
- 合計ページは、(定義されたドキュメントセクションだけではなく)ドキュメント全体の合計ページ数を表示するものです。

ページの内部でヘッダー/フッターのシミュレーションを行う

ヘッダーならびにフッターはレイアウトコンテナー内部に手動で配置することができます。1つのページをデザインコンテナーとしてデザインする際に、この方法を使用することができます。静的テーブル内部に作成されたヘッダーとフッターは、それぞれページの上部と下部に配置されます。複数のページをデザインする場合、改ページ(「**挿入 | ページ/カラム/ドキュメントセクションを挿入 | 新規ページ**」)により分割された複数のレイアウトコンテナーを使用することができます。

承継や合計数をヘッダー/フッターに表示する

ドキュメント内部に複数ページにわたる数値アイテムのリストが含まれており、これら数値アイテムの合計を表示する必要がある場合、各ページのヘッダーやフッターに、そのページにおける小計や、それまでの合計を表示することができます。(マイ)ドキュメントフォルダーの`C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples\Tutorial\Subtotals\` にある Subtotals.sps サンプルでは、それまでの合計がヘッダーとフッターに表示されます。

SPS は以下のようにして作成されています:

- テーブル形式による表示が行われ、印刷媒体に対してテーブルの大きさを自動的に合わせることができないため、ページ内に表示される行の数を指定する必要があります。これらの値は、トップレベル(`$XML` テンプレート)で定義された、`RowsOnFirstPage` ならびに `RowsPerPage` という2つの変数に対して与えられます。
- リストに含まれる合計アイテム数をページごとの行数で割った数(端数切捨て)がページ数となります(最初のページにある行数とは別の行数へも対応することができます)。ページ数は(`$XML` テンプレートで定義された)`CountOfPages` という変数へ収められます。
- ユーザー定義のテンプレートが 1 から `$CountOfPages` のシーケンスに対して作成され、テンプレート内部に静的テーブルが作成されます。このテンプレートでは、各ページにおける最初の行 (`$RowFrom`) と 最後の行 (`$RowTill`) を計算するための変数が定義されています。ユーザー定義のテンプレートによりテーブルの行が生成され、`$RowFrom` と `$RowTill` の値に対応する場所から、XML ファイル内部のアイテム(file 要素)が選択されます。file 要素の位置が `$RowFrom` と `$RowTill` 変数の中に収まる場合、その file 要素に対して行が生成されます。
- それまでの合計は自動計算により求められ、テーブルの上部と下部の行に挿入されます。合計値を計算するための XPath 条件式は、ページの上部と下部にて異なる点に注意してください。
- ヘッダーとフッターが、メインテーブルの上部と下部にそれぞれ作成されます。それまでの合計を生成する自動計算が、ヘッダーならびにフッターテンプレートに挿入されます。
- 各ページの終端に改ページが挿入されます。

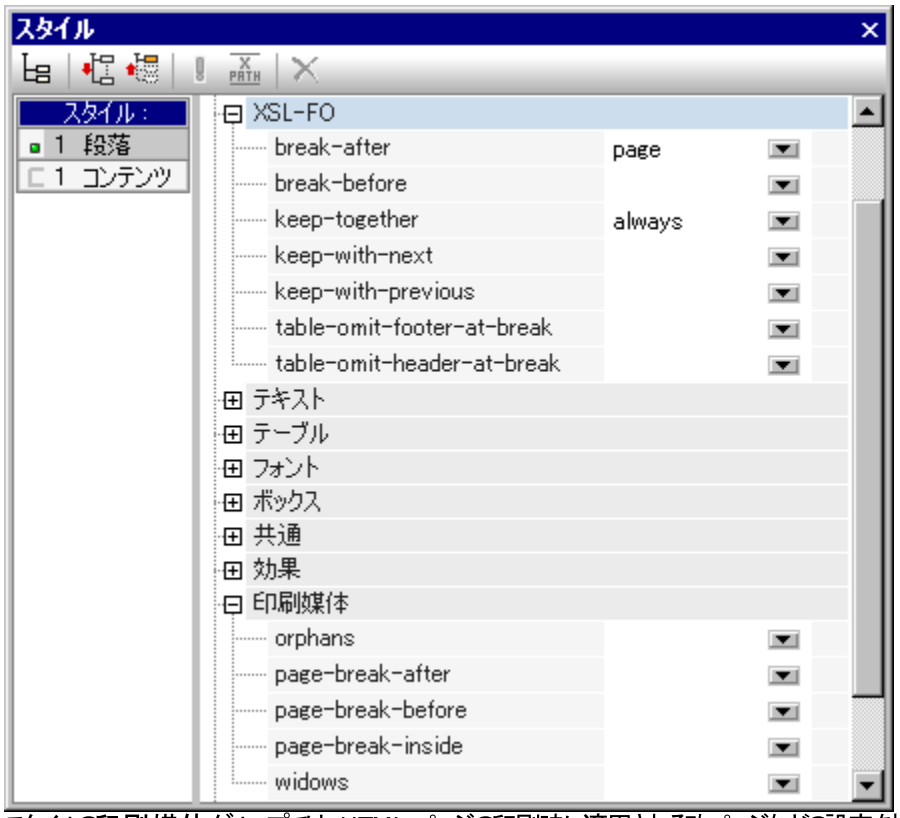
関連項目

- ドキュメント セクション

- ページレイアウト プロパティ
- 改ページ

8.6.2 改ページ

(Enterprise エディションで生成することができる)PDF ドキュメントでは、スタイルの XSL-FO グループにてページやカラムの改ページを指定することができます(以下のスクリーンショットを参照)。このスタイルグループでは、選択されているデザインドキュメントブロック(カーソルが配置されているブロック)の前/後に改ページ/改カラムを配置するかを指定することができます。table-omit プロパティでは、改ページにてテーブルのヘッダーやフッターを表示するか指定することができます。これらプロパティに関するより詳細な情報については、XSL-FO の仕様を参照ください。



スタイルの印刷媒体グループでは、HTML ページの印刷時に適用される改ページなどの設定を指定することができます(上のスクリーンショットを参照)。

関連項目

- ページレイアウト プロパティ: ページプロパティの定義について記述されます。
- ヘッダーとフッター : ヘッダーならびにフッターの使用方法について記述されます。
- スタイル サイドバー

(C) 2015-2021 Altova GmbH

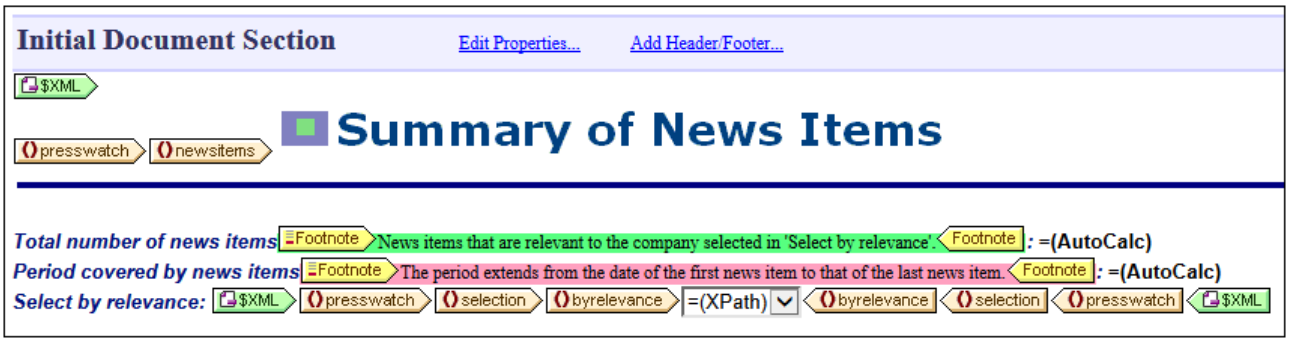
8.6.3 フットノート

フットノートコンポーネントをフットノートの番号を表示したい場所に追加することによりフットノートを挿入することができます（**挿入 | フットノートの挿入**）。フットノートは、ページ番号が与えられ以下の書式で出力することができます（Enterprise Edition では、PDF、RTF、および Word 2007+（**挿入 | フットノートの挿入**）、Professional Edition では、RTF）。

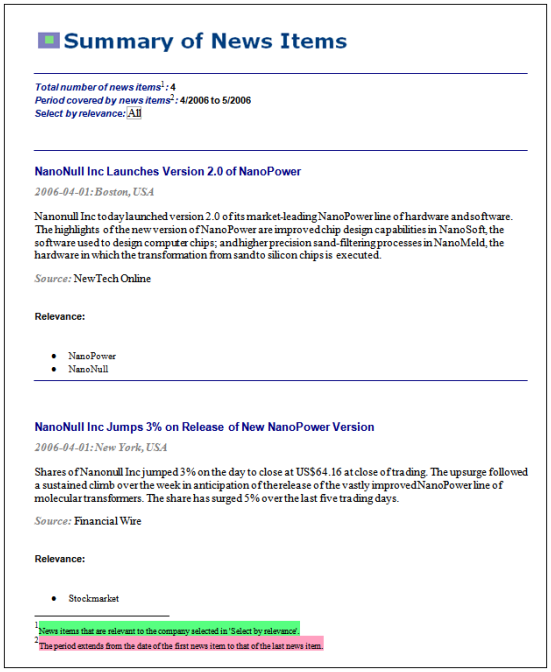
以下の点に注意してください：

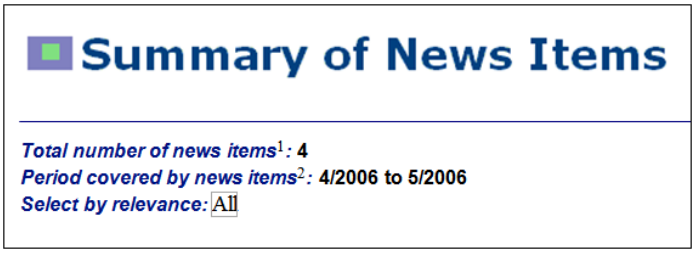
- フットノートのテキストは、フットノートコンポーネントのタグ内におかれる必要があります。フットノートのテキストをフォーマットすることもできます。
- 出力内では、フットノートの番号はフットノートが追加された場所に表示されます。フットノートのテキストは対応するフットノートの番号と共にページの下に表示されます。
- 出力内では、フットノートのテキストがデザイン内のフットノートコンポーネント内のテキストの書式に従いフォーマットされます。
- 出力内では、フットノートは、自動的にドキュメントの終わりまで番号が与えられます。
- In複数の出力ドキュメントの場合、各出力ドキュメントのために番号が与えられます。

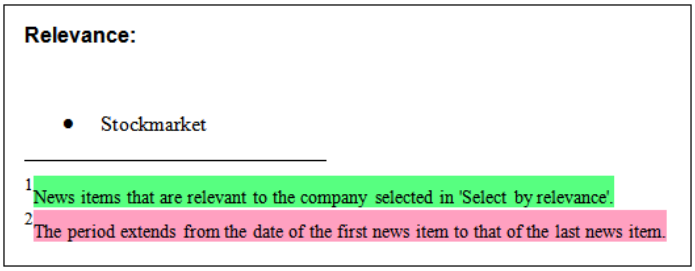
下のスクリーンショットでは、2つのフットノートコンポーネント（**挿入 | フットノートの挿入**）が挿入されています。フットノートのテキストはコンポーネントのタグの中に置かれ、テキストはフォーマットされています。



下のスクリーンショットは出力を表示します。左のスクリーンショットはページ全体を表示し、右のスクリーンショットは、フットノートの番号（上）のフットノートのテキスト（下）拡大を表示しています。







メモ フットノートの番号の書式はサポートされていません。

8.6.4 ピクセルの解像度

SPS にてピクセルを長さの単位に使用する場合、ピクセルにより定義された長さは画面の解像度に対して使用されるということに留意してください。印刷時に表示される長さと画面に表示されるそれは、大きく異なる場合があります。このセクションでは、以下の事柄について説明します:

1. ピクセルにより定義された大きさが、(i) ピクセルにより定義された抽象的な形式と、(ii) 抽象的な値と画面の解像度から得られる実際の大きさという、2つの形式を持つ理由について説明する。
2. この件に対する StyleVision の機能を説明する。

ピクセルからポイントへ

ピクセルにより定義された抽象的な大きさから、実際の大きさを決定する係数を理解する必要があります:

- ピクセルは相対的な単位で、実際に表示される大きさは画面の解像度に影響されます。解像度が高くなるにつれて、(実際に表示される)ピクセルの長さが短くなります。画面の解像度は dpi (dots per inch) という単位で表されます。ここでのドット (dot) は、画面上のピクセルということになります。例えば、画面の解像度が 72 dpi の場合、1インチの線上に72個のピクセル(ドット)が並んで表示されることになります。画面解像度が 96 dpi の場合、1インチに96ピクセルが納められることになります。画面上の1インチに表示されるピクセルの数は、画面の解像度に依存することになります。今日最も一般的な画面解像度は 72 dpi、96 dpi、120 dpi となります。dpi の数値が高くなるにしたがって、表示されるピクセルの大きさが小さくなります。
- ポイントという長さ単位は、絶対的な長さの単位で、印刷業界にて最も広く使用されています。72 ポイントが1インチに相当します。
- 上記のことから、画面解像度が 72 dpi となっている場合のみ、1ピクセルを1ポイントとみなすことができるようになります。その他の画面解像度に対しては、絶対的な大きさを計算することになります。以下のテーブルでは、100ピクセルに対応した(ポイントによる)絶対的な大きさと、画面解像度の対応を示します。

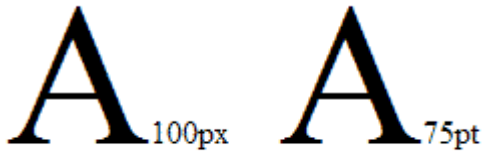
解像度	ピクセル	ポイント	係数
72 dpi	100	100	1.00
96 dpi	100	75	0.75
120 dpi	100	60	0.60

異なる画面解像度に対してピクセルをポイントへ変換するには、1インチに収められるポイントの数 (72) と、1インチに収められるドットの比を表す係数を使用します。異なる解像度に対する変換の係数を上のテーブルに示します。例えば、72 ÷ 96 = 0.75 で、72 ÷ 120 = 0.60 となります。ピクセルの長さを係数で掛けることにより、ポイントという絶対的な長さの単位を求めることができます。96 dpi により表示される100 ピクセルは75ポイントとなります。

72ポイントが1インチとなるため、ポイントの数値を72で割ることにより、インチを単位とする長さを求めることができます。例えば、100ポイントは、100 / 72 = 1.389インチ となります。

画面の解像度と絶対的な長さ

これまでの記述では、画面解像度が72 dpi となっている場合、つまりポイントによる絶対的な長さがピクセル単位で表示されている画面上の長さと同一になる場合だけを取り上げてきました。W indows システムにおける画面の解像度は、通常 72 dpi ではなく 96 dpi となっており、このような画面上に表示される(本来ピクセルにより定義された) ポイント単位の絶対的な長さは、ピクセルの75% となります。例えば、以下に示される StyleVision デザインビューのスクリーンショットは、100px と75pt をそれぞれ表示したものです。



画面の解像度が96 dpi となっているため、表示される文字の高さが等しくなります。この解像度では、100px が、(絶対的な長さ単位の) 75pt と同じ大きさで表示されます。

ピクセルを長さの単位とする場合、(W indows システムでは通常 96 dpi となっている) 画面の解像度を確認するようにしてください。これにより画面に表示されるピクセルの絶対的な長さが決定されます。

印刷出力の解像度

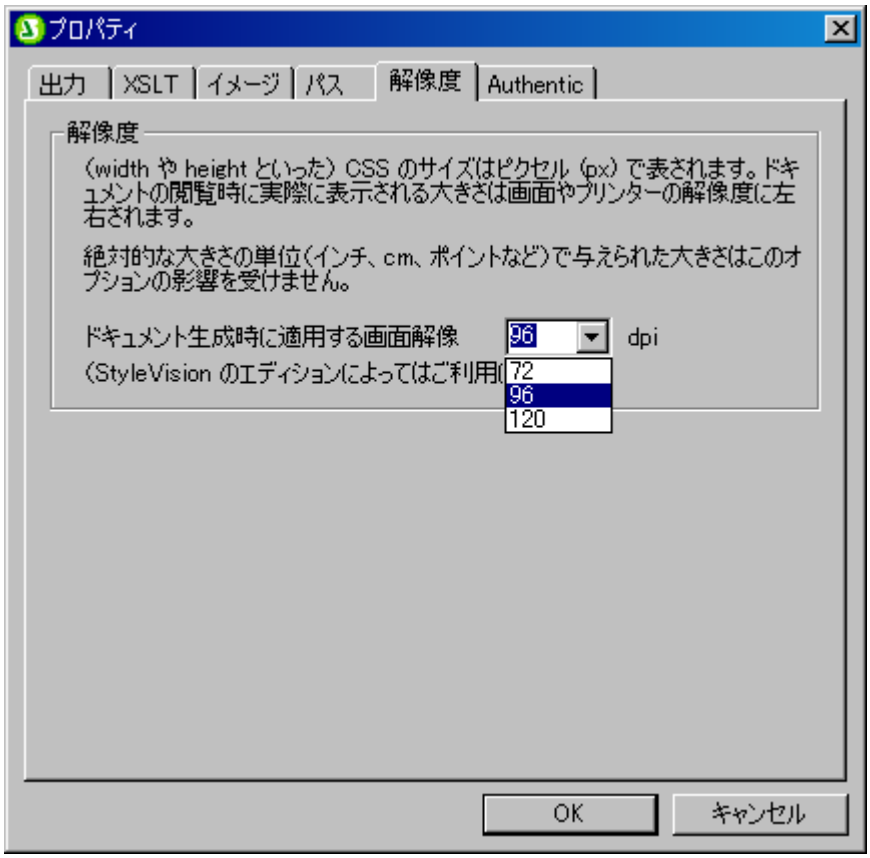
SPS がPDF やRTF、W ord 2007+ といった印刷形式に変換された場合、絶対的な長さの単位ではないピクセルの長さをポイントやインチといった絶対的な長さに変換する必要があります(これはピクセルという単位で印刷を行うことができないためです)。そこで、ピクセルからポイントへの変換を行う際に使用する係数(または画面解像度) を選択する必要があります。

StyleVision では、各 SPS に対して出力のdpi を選択することができます。例えば、96 dpi を選択した場合、100px の文字が75pt の文字として印刷形式ではレンダリングされます。72 dpi を選択した場合、100px の文字が、100pt として出力されます(上のテーブルを参照ください)。選択された値は、ピクセルで定義された全ての長さに対して適用されます。

メモ 絶対的な長さへの変換は HTML 出力に対して行われません。HTML では、元々使用されていたピクセル単位の長さが HTML ファイルへそのまま渡されます。

SPS の印刷出力解像度を選択する

SPS の印刷出力解像度を選択するには、メニューオプションから「**ファイル | プロパティ**」を選択します。プロパティダイアログにて**解像度** タブを選択します(以下のスクリーンショットを参照)。



解像度タブにて、目的の印刷解像度を選択します。表示されている dpi オプションに対して、上記の変換テーブルを使用することができます(上記の**ピクセルからポイントへ**を参照ください)。ピクセルの長さに係数を掛けることで、ポイント単位による絶対的な長さを得ることができます。絶対的な長さへの変換は、印刷出力形式に対してしか適用されないことに注意してください。HTML 出力形式ではオリジナルのピクセルによる定義が保たれます。この設定により画面の解像度が変更されることはありません。

関連項目

- CSS プロパティ値の設定

8.6.5 透かし

透かしにより、ドキュメントセクションの各ページにイメージやテキストを背景として表示することができます。

以下のオプションを利用することができます:

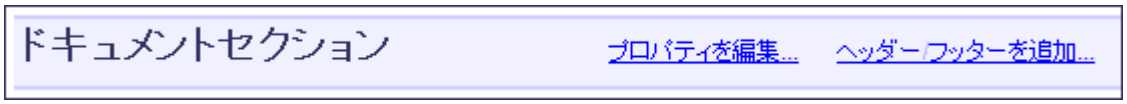
- 各ドキュメントセクションに対して個別の透かしを定義することができます。これにより、セクションごとに異なる透かしが表示されることになります。
- 各ドキュメントセクションに対して、そのセクションの透かしを有効にするための条件を(XPath により)定義する必要があります。例えば、特定の要素にて draft という属性が存在すれば、評価結果が true となる条件式を指定することができます。条件が満たされない、または条件の結果が false() となる場合、透かしは表示されません。
- 透かしにはStyleVision で指定したテキストや選択されたイメージを使用することができます。

- デザインでは、そのページに対応した透かしの場所と表示方法を指定することができます。

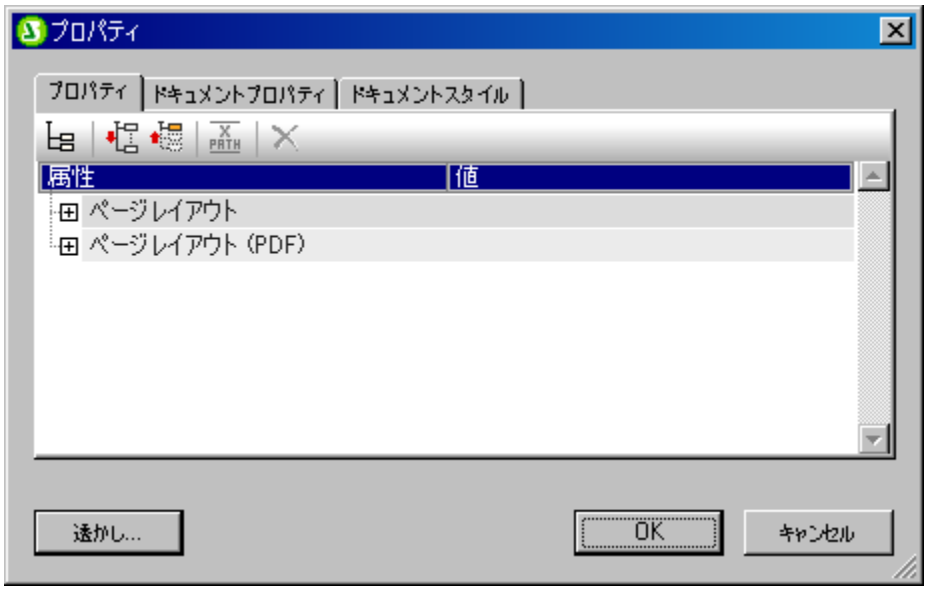
メモ　透かしは印刷媒体の出力に対してのみ使用することができ、HTML や Authentic View には適用されないことに注意してください。

透かしの作成

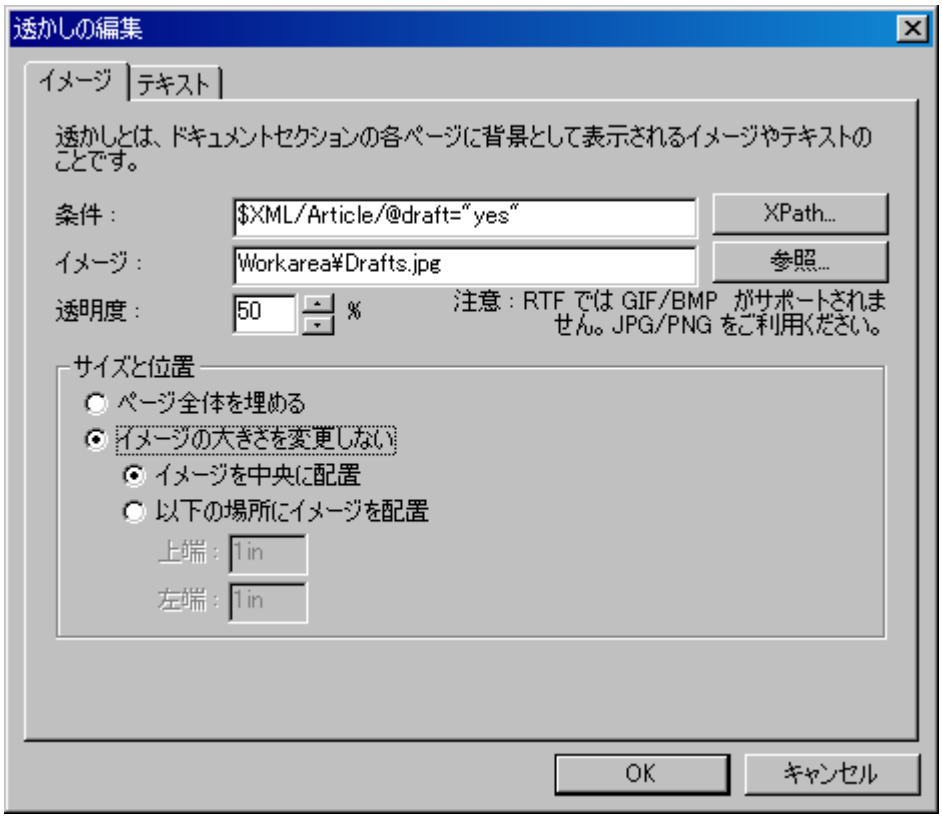
透かしは各ドキュメントセクションに対して個別に作成することができます。透かしを作成（または編集）するには、デザインのイニシャルドキュメントセクションやドキュメントセクションアイテムにあるプロパティを編集リンクをクリックしてください（以下のスクリーンショットを参照）。



表示されるプロパティダイアログ下部にある透かしボタンをクリックしてください（以下のスクリーンショットを参照）。



透かしの編集ダイアログが表示され、そのドキュメントセクションに対して透かしのプロパティを指定することができます（以下のスクリーンショットを参照）。イメージタブとテキストタブにて、透かしのイメージまたはテキストをそれぞれ指定することができます。セクションの透かしにテキストとイメージの両方を使用する場合、それぞれのタブにて設定を指定するようにしてください。この場合、両方の透かしが出力にて作成されます。



イメージ型透かしのパラメーター

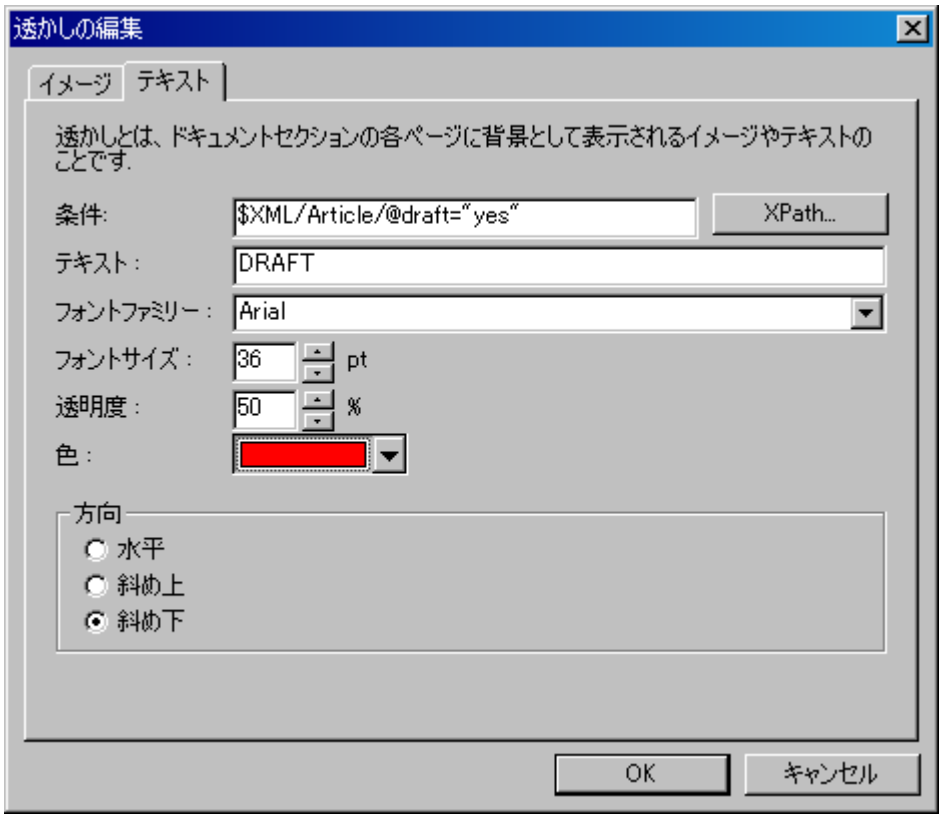
イメージタブには、イメージ型の透かしをそのセクションにて定義するためのパラメーターが含まれています。XPath 条件式により、イメージの透かしを有効にする条件を指定してください。条件を指定しない場合には、**条件**テキストボックスにtrue() と入力してください（これにより条件の評価結果が常にtrue() となります）。条件の評価結果がfalse() となる場合、または条件が指定されていない場合、イメージの透かしは有効になりません。次にイメージの参照を行うことで、イメージへのファイルパスが**イメージ**テキストボックスに入力されます。

イメージの透明度や、イメージの大きさ、ページに配置される場所をセットすることができます。**ページ全体を埋める**オプションを選択することで、ページの幅または高さに合わせてイメージが拡大されます。「**OK**」をクリックすることで確定します。

テキスト型透かしのパラメーター

テキストタブには、テキスト型の透かしをそのセクションにて定義するためのパラメーターが含まれています。XPath 条件式により、テキストの透かしを有効にする条件を指定してください。条件を指定しない場合には、**条件**テキストボックスにtrue() と入力してください（これにより条件の評価結果が常にtrue() となります）。条件の評価結果がfalse() となる場合、または条件が指定されていない場合、テキストの透かしは有効になりません。

(C) 2015-2021 Altova GmbH



次に、透かしとして使用するテキストを入力し、そのフォーマット（フォントファミリー、フォントサイズ、透明度、色）や、表示方向を指定してください。テキストは、指定された方向によりページの端から端までをカバーするように表示されます。「OK」をクリックすることで指定を完了します。

透かしの削除と無効化

透かしを削除または無効にする場合、透かしの編集ダイアログにて XPath 条件式の条件を削除または false() を入力してください。イメージタブとテキストタブでは個別の条件式が使用されることに注意してください。

関連項目

- イニシャルドキュメントセクション
- ページレイアウト プロパティ

9 追加機能

このドキュメンテーションで記述されたコンテンツの編集、構造、高度な機能、様々な表示方法 の他にも、StyleVision には様々な機能が搭載されています。これら機能の詳細については、以下に記されるサブセクションにて記述されます。

- グローバルリソース: グローバルリソースを使用することで、柔軟なリソースの選択を行うことができます。例えば、(ファイルやデータベースのような) 複数のリソースをエイリアスへ割り当てることができます。SPS のソース(XML や XSD など) にエイリアスを使用することで、エイリアスに割り当てられた(複数ある) リソースを簡単に切り替えることができます。
- Authentic ノードプロパティ: XML ドキュメント内にある個々のノードに対して Authentic View 固有のプロパティを割り当てることができます。ノードは読み取り専用として定義することもでき、マウスオーバーされた際にマークアップのタブに対するユーザー情報を表示するのに使用することができます。
- クリックによる親ノードの置き換え: ボタンやハイパーリンクの親ノードに含まれる値を、Authentic View ユーザーにより選択することができます。Authentic View ユーザーの選択に従うかたちで表示が変更されるような SPS をデザインすることができます。
- 追加検証: Boolean 型の値を返す XPath 条件式を使用することで、そのノードに対して入力された値が妥当なものであるか検証することができます。この検証は、ドキュメントがスキーマに対して妥当なものかの検証とは別に行われます。
- 日付との作業: Authentic View ではグラフィカルな日付選択機能を使用することにより、入力された値が XML スキーマフォーマットに従った適切なものであることを保証することができます。
- パースされていないエンティティ URI: XML ドキュメントがベースとしている DTD にパースされていないエンティティを収めることができます。パースされていないエンティティ URI 機能により、イメージやハイパーリンクを URI のターゲット URI として使用することができます。
- スクリプトの使用. StyleVision には JavaScript エディターが搭載されており、JavaScript 関数を定義することができます。これらの関数は SPS における任意の場所からイベントハンドラーとして使用することができ、HTML ドキュメントの出力に対して影響を与えることができます。
- HTML のインポート: HTML ファイルを StyleVision にインポートすることができるほか、HTML ファイルから XML、XSD、そして SPS ファイルを作成することができます。
- XSLT からの新規作成: SPS ファイルを XSLT-for-HTML や XSLT-for-FO テンプレート構造から作成することができるほか、SPS にて XSLT におけるスタイルルールを作成することができます。SPS コンポーネントを修正して、コンテンツやフォーマットを SPS に加えることができます。

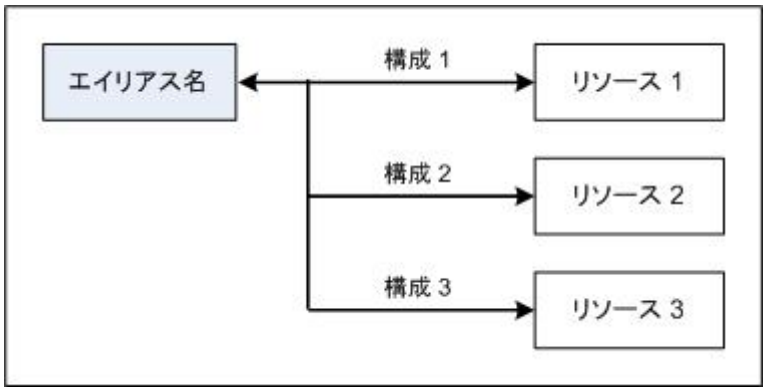
関連項目

- プロパティサイドバー
- Authentic View

(C) 2015-2021 Altova GmbH

9.1 Altova グローバルリソース

Altova グローバルリソースはファイル、フォルダー、そしてデータベースリソースのエイリアス集合のことです。各エイリアスは複数の構成を持ち、各構成を1つのリソースにマッピングすることができます（下のスクリーンショット参照）。そのため、グローバルリソースを入力として使用することで、グローバルリソースを構成とともに切り替えることができます。この操作は、アクティブな構成を選択することのできるGUI 上で簡単に行うことができます。例えば、XMLドキュメントを変換するXSLT スタイルシートがグローバルリソース（エイリアス）により割り当てられている場合、このグローバルリソースに対して複数の構成を作成し、各構成が異なるXSLT ファイルを指すようにすることができます。このようにグローバルリソースを設定すると、構成を切り替えることで変換に使用されるXSLT ファイルを変更することができます。



グローバルリソースはAltova 製品内部におけるリソースの切り替えだけでなく、他のAltova 製品から生成、使用されるリソースに対しても使用することができます。そのため、1つのAltova 製品から他のAltova 製品で使用されるファイルを一括して生成することができます。このような機能により、開発ならびにテストスピードが飛躍的に向上します。例えば、MapForce のマッピングによりXML ファイルを生成し、そのファイルをStyleVision のSPS 作業用XML ファイルとして使用することができます。

Altova グローバルリソースの使用には、2 つのステップがあります:

- グローバルリソースの定義: XML ファイルによりリソースが定義されます。これらのリソースは複数のAltova 製品間で共有することができます。
- グローバルリソースの使用: StyleVision 内部ではファイルパスの代わりにグローバルリソースによりファイルを指定することができます。グローバルリソースを使用すると、StyleVision にてアクティブな構成を切り替えることで、使用するリソースを簡単に変更することができるようになります。

その他のAltova 製品におけるグローバルリソース

現在のところ、以下のAltova 製品でグローバルリソースを使用することができます: XMLSpy、StyleVision、MapForce、Authentic Desktop、MobileTogether Designer およびDatabaseSpy。

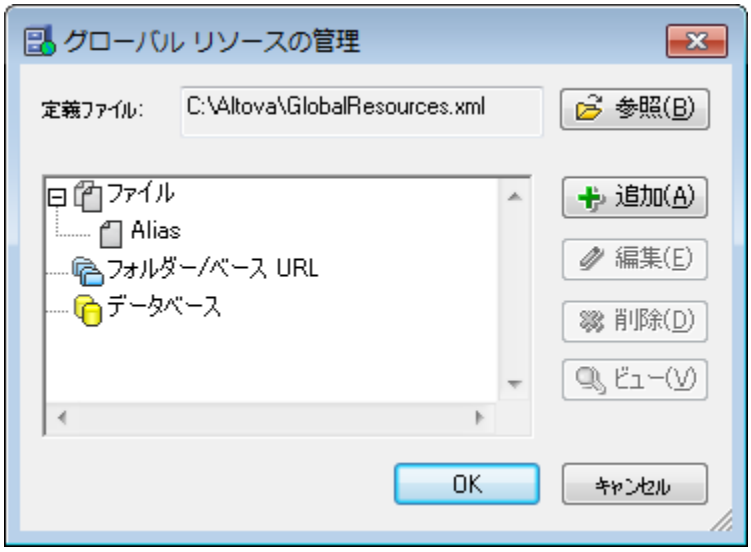
9.1.1 グローバル リソースの定義

Altova グローバルリソースは、以下の方法によりアクセスすることのできるグローバルリソースの管理ダイアログで定義されます:

- メニューコマンドの「**ツール | グローバル リソース**」を選択する。
- グローバルリソースツールバーにある「**グローバル リソースの管理**」アイコンをクリックする（下のスクリーンショット）。

グローバルリソース定義ファイル

グローバルリソースに関する情報はグローバルリソース定義ファイルと呼ばれるXML ファイルに収められています。このファイルは、グローバルリソースの管理ダイアログにて初めてグローバルリソースが定義ならびに保存された際に作成されます（下のスクリーンショットを参照）。



グローバルリソースの管理ダイアログを初めて開くと、グローバルリソース定義ファイルのデフォルトの場所と名前が、定義ファイルテキスト ボックスにて指定されます（上のスクリーンショットを参照）:

```
C:\Users\<username>\My Documents\Altova\GlobalResources.xml
```

このファイルは、すべてのAltova アプリケーションに対するデフォルトのグローバルリソース定義ファイルとしてセットされます。従って、任意のAltova アプリケーションからこのファイルに対して保存されたグローバルリソースは、その他すべてのAltova アプリケーションにて直ちにグローバルリソースとして使用することができることになります。グローバルリソースを定義して、グローバルリソース定義ファイルに保存するには、グローバルリソースの管理ダイアログにてグローバルリソースを追加し、「OK」ボタンをクリックしてください。

既に存在するグローバルリソース定義ファイルを、実行中のAltova アプリケーションにおけるアクティブな定義ファイルとして選択するには、定義ファイル テキスト ボックスの隣にある「参照」ボタンから、そのファイルを選択してください。（上のスクリーンショットを参照）

メモ: グローバルリソース定義ファイルの名前は自由に変更することができ、Altova アプリケーションからアクセスできる任意の場所へ配置することができます。この場合、各アプリケーションにて（定義ファイル テキスト ボックスから）このファイルをアプリケーションのグローバルリソース定義ファイルとして指定する必要があります。複数のAltova 製品にて同一の定義ファイルを指定することで、これらのAltova 製品にてリソースが利用できるようになります。

メモ: 複数のグローバルリソース定義ファイルを作成することもできます。この場合、各 Altova アプリケーションにて同時に選択できるアクティブな定義ファイルは1 つだけで、そのファイルの中に含まれている定義だけがアプリケーションでは利用できるということに注意してください。そのため、必要に応じて同じリソースを複数の定義ファイルで記述する必要が生じることもあります。

グローバルリソースの管理: 追加、編集、削除、保存

グローバルリソースの管理ダイアログでは、選択されたグローバルリソース定義ファイルへグローバルリソースを追加することができるほか、選択されたグローバルリソースを編集または削除することができます。ファイル、フォルダー、そしてデータベースのグループに追加されたグローバルリソースは、グローバルリソース定義ファイルにより管理されます（上のスクリーンショットを参照）。

グローバルリソースを追加するには、「追加」ボタンをクリックして、表示される「**グローバルリソース**」ダイアログにてグローバルリソースを定義します（このセクション以下のサブセクションにある「ファイル」、「フォルダー」、「データベース」の記述を参照ください）。グローバルリソース

を定義して（グローバルリソースの管理ダイアログにある「OK」ボタンをクリックすることで）保存すると、選択されたグローバルリソース定義ファイルのグローバルリソース定義ライブラリに、そのグローバルリソースが追加されます。グローバルリソースはエイリアスにより識別されます。

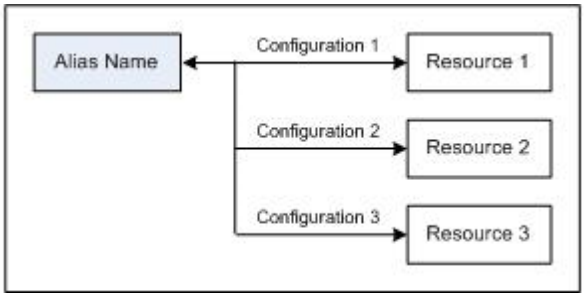
グローバルリソースを編集するには、「**編集**」ボタンをクリックします。関連するグローバルリソースダイアログが表示され、必要な変更を行うことができます（「ファイル」、「フォルダー」、「データベース」サブセクションの記述を参照ください）。

グローバルリソースを削除するには、「**削除**」ボタンをクリックします。

グローバルリソースの追加、編集、または削除を終えた後には、グローバルリソースの管理ダイアログにある「OK」ボタンをクリックして、グローバルリソース定義ファイルに**変更点を保存**するようにしてください。

構成によりグローバルリソースをエイリアス名へ関連付ける

グローバルリソースを定義すると、エイリアス名をリソース（ファイル、フォルダー、データベース）へマッピングすることになります。1 つのエイリアス名から複数のリソースへのマッピングを行うことができ、各マッピングは構成と呼ばれます。そのため、単一のエイリアス名から、複数の構成を介して複数のリソースへの関連付けを行うことができます（以下の図を参照）。



Altova アプリケーションではファイルの代わりにエイリアスを割り当てることができます。アプリケーションのアクティブなグローバルリソース構成を切り替えることで、各エイリアスに割り当てられたリソースを切り替えることができます。例えば Altova XMLSpy アプリケーションにて、MyXML.xml という名前の XML ドキュメントに対して XSLT 変換を行う場合、MyXSLT という名前のエイリアスを使用される XSLT としてグローバルリソースに割り当てることができます。XMLSpy では、アクティブな構成を変更することで、異なる XSLT ファイルを使用することができます。構成-1 にて First.xslt が MyXSLT へマッピングされており、構成-1 がアクティブな構成として選択されている場合、First.xslt が変換に使用されます。これにより、複数の構成を使用することで、単一のエイリアスから複数のリソースへアクセスすることができるようになります。このメカニズムはリソースのテストや比較を行う際に便利な機能となります。更に、グローバルリソースは複数の Altova 製品間で使用することができるため、複数の Altova 製品にてリソースのテストや比較を行うことができます。

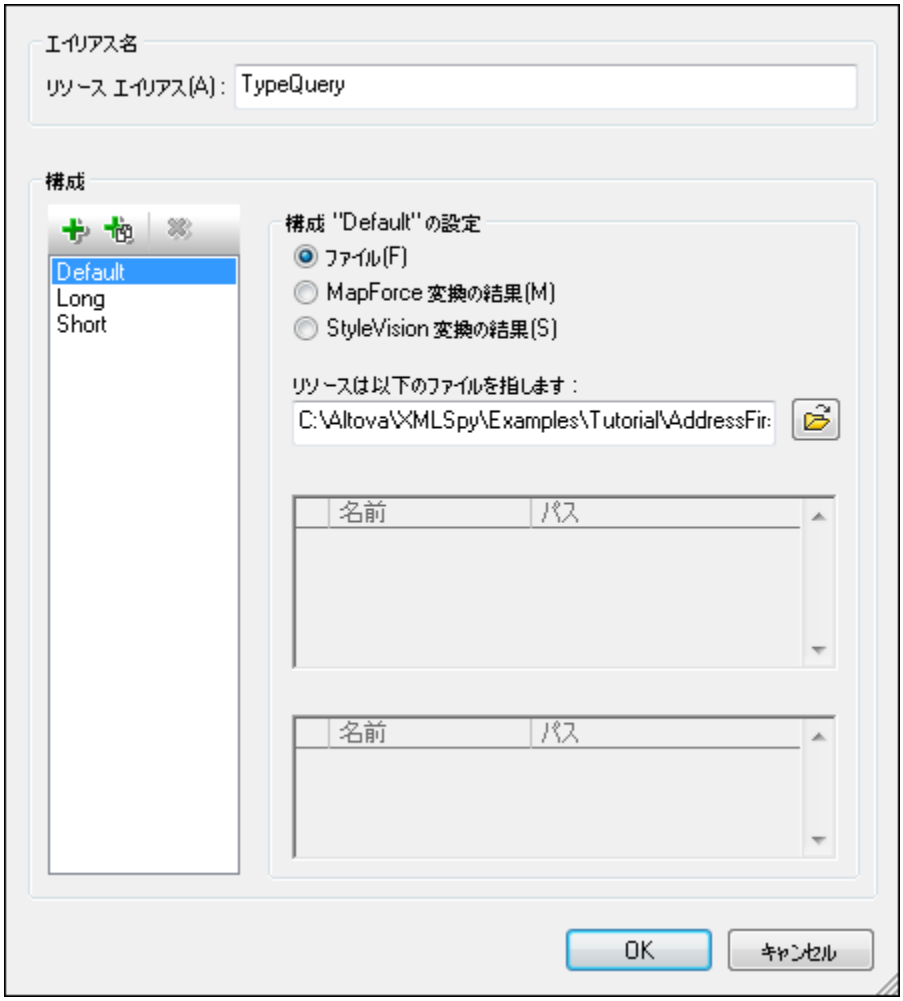
9.1.1.1 ファイル

「グローバルリソースの管理」ダイアログにある「**追加 | ファイル**」コマンドにより、ファイルのグローバルリソースダイアログへアクセスすることができます（以下のスクリーンショットを参照）。グローバルリソースダイアログでは、リソースエイリアステキストボックスにて入力されたエイリアスの構成を定義することができます。以下に記される方法により構成のプロパティを指定して、「OK」をクリックしてエイリアスの定義を保存してください。

エイリアスの定義を保存した後には上記の操作を（「グローバルリソースの管理」ダイアログにある「追加 | ファイル」コマンドを選択するところから）再度行うことで、別のエイリアスを追加することができます。

グローバルリソースダイアログ

エイリアスはグローバルリソースダイアログにて定義されます（以下のスクリーンショットを参照）。



グローバルリソース ダイアログのアイコン

構成の追加: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

構成をコピーして追加: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

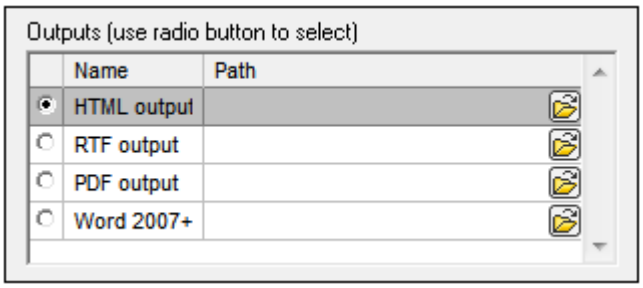
削除: 選択された構成を削除します。

開く: グローバルリソースとして作成されるファイルを参照します。

エイリアスの定義

以下の操作により、エイリアスの名前と構成を定義することができます:

1. *エイリアスに名前を付ける:* リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。
2. *構成を追加する:* 構成ペインには、デフォルトで（名前変更や削除を行うことができない）Default という名前の構成が表示されます。(i) 構成の追加または構成のコピーを追加アイコンをクリックして、(ii) 表示されるダイアログにて**構成名を入力**することで、必要なだけの**構成を追加**することができます。追加された各構成は構成リストに表示されます。上のスクリーンショットでは、Long と Short という名前の構成が構成リストに追加されています。構成のコピーを追加コマンドにより、選択されている構成をコピーして、修正することもできます。
3. *各構成に対してリソースの種類を選択する:* 構成のリストから目的の構成を選択し、構成の設定ペインにて、(i) ファイル、(ii) MapForce 変換の結果、または (iii) StyleVision 変換の結果から、構成のリソースを指定します。対応するラジオ ボタンを選択してください。MapForce または StyleVision 変換の結果が選択された場合、.mfd または .sps ファイルと対応する入力ファイルを使って MapForce または StyleVision により変換が行われ、その変換結果がリソースとなります。
4. *リソースの種類に対するファイルを選択する:* リソースの種類にファイルが選択された場合、リソースのファイル テキスト ボックスにてファイルを選択してください。リソースの種類に変換の結果が選択された場合、ファイルの選択テキスト ボックスにて（MapForce 変換の場合は）.mfd ファイルまたは（StyleVision 変換の場合は）.sps ファイルを選択してください。変換に対して複数の入力または出力が可能な場合、選択するためのオプションが表示されます。例えば、StyleVision 変換の出力オプションが、インストールされている StyleVision のエディションに合わせて表示されます（以下のスクリーンショットでは、Enterprise Edition の出力が示されています）。



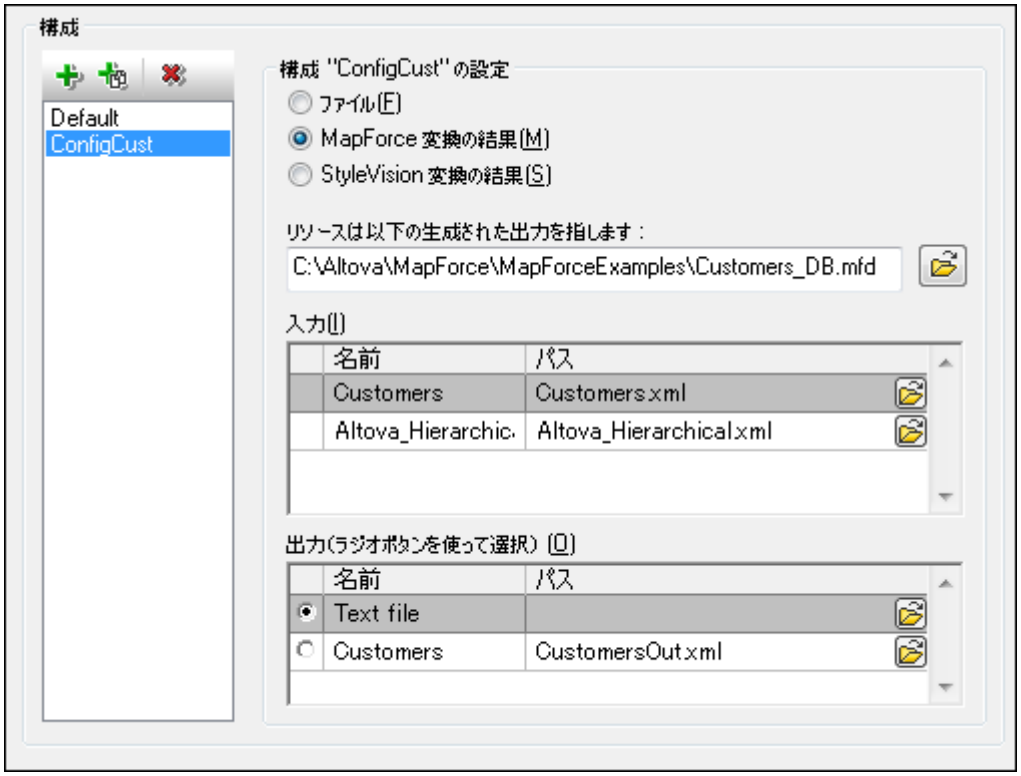
目的のオプションに対応するラジオ ボタンを選択します（上のスクリーンショットでは、「HTML 出力」が選択されています）。変換の結果をリソースとする場合、出力をファイルまたはグローバルリソースとして保存することができます。 アイコンをクリックして、「他のグローバルリソースを選択」または、「参照」を選択してください。どちらのオプションも選択されていない場合、グローバルリソースが使用された際に変換の結果が一時ファイルとしてロードされます。

5. *必要に応じて複数の構成を定義する:* その他の構成を必要な数だけ追加して、各構成に対してリソースを指定します。各構成に対して上記のステップ 3 と 4 を実行してください。エイリアスに対する新たな構成はいつでも追加することができます。
6. *エイリアスの定義を保存する:* 「OK」をクリックしてエイリアスと、すべての構成をグローバルリソースとして保存します。グローバルリソースが、「グローバルリソースの管理」ダイアログにあるファイル以下に表示されます。

MapForce 変換の結果

Altova MapForce では、（既存の）入力ドキュメント スキーマから出力ドキュメント スキーマへのマッピングが行われます。MapForce ユーザーにより作成されたこのマッピングは、MapForce Design (MFD) と呼ばれます。入力スキーマに関連付けられた XML ファイル、テキスト ファイル、データベースは、データ ソースとして使用することができ、MapForce により出力ドキュメント スキーマに対応した出力データ ファイルが生成されます。「MapForce 変換の結果」が出力ドキュメントで、グローバルリソースになります。

MapForce により生成されたデータ ファイルをグローバルリソースとしてセットするには、グローバルリソース ダイアログにて、以下の設定を行う必要があります（以下のスクリーンショットを参照）。



- **.mfd (MapForce Design) ファイル**: 「リソースは以下の生成された出力を指します」テキスト ボックスにて、.mfd ファイルを指定する必要があります。
- **入力データ ファイル**: MFD ファイルが指定されると、そのファイルが解析され、中に含まれる入力スキーマ情報をベースにデフォルトのデータ ファイルが入力ペインに入力されます（上のスクリーンショットを参照）。他のファイルを指定することで、各入力スキーマに対するデフォルト ファイルの選択を変更することができます。
- **出力ファイル**: MFD ドキュメントに複数の出力スキーマが含まれる場合、出力ペインにそれらの出力すべてが表示されます（上のスクリーンショットを参照）。個々の出力スキーマに関連付けられた出力ファイルの場所が MFD ドキュメントにて指定されている場合、その出力スキーマに対するファイルの位置が出力ペインに表示されます。上のスクリーンショットからは、Customers 出力スキーマにデフォルトの XML データ ファイル（CustomersOut.xml）が与えられており、Text ファイル出力 スキーマにはそのような割り当てが行われていないことが理解できます。出力ペインに表示あるデフォルトのファイルの場所を使用することができるほか、自分で指定することができます。MapForce 変換の結果が、選択された出力スキーマがある場所へ保存され、そのファイルがグローバル リソースとして使用されます。

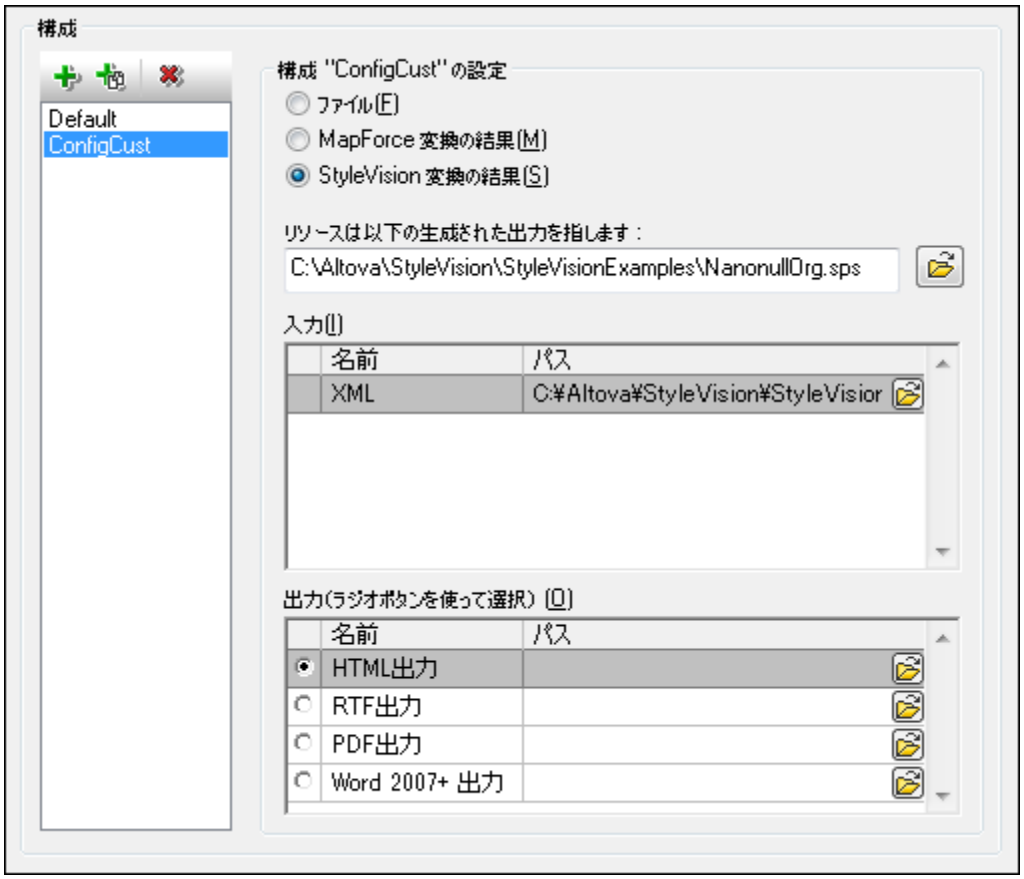
メモ このオプション（MapForce の結果）により、グローバル リソースが呼び出された時に変換が行われるようになります。これで（入力ファイルから得られた）最新のデータがグローバル リソースに含まれることが保証されます。

メモ 変換を行うためには MapForce が使用されるため、この機能を使用するには Altova MapForce をインストールする必要があります。

StyleVision 変換の結果

Altova StyleVision は StyleVision Power Stylesheet（SPS）ファイルを作成するために使用され、これら SPS ファイルにより、XML ドキュメントを様々なフォーマット（HTML、PDF、RTF、Word 2007+ など）の出力ドキュメントへ変換するための XSLT スタイルシートが生成されます。「StyleVision 変換の結果」オプションを選択すると、StyleVision により作成された出力ドキュメントが、選択された構成に関連付けられたグローバル リソースとなります。

グローバル リソース ダイアログにて「StyleVision 変換の結果」オプションを選択すると、以下のファイルを指定する必要があります。



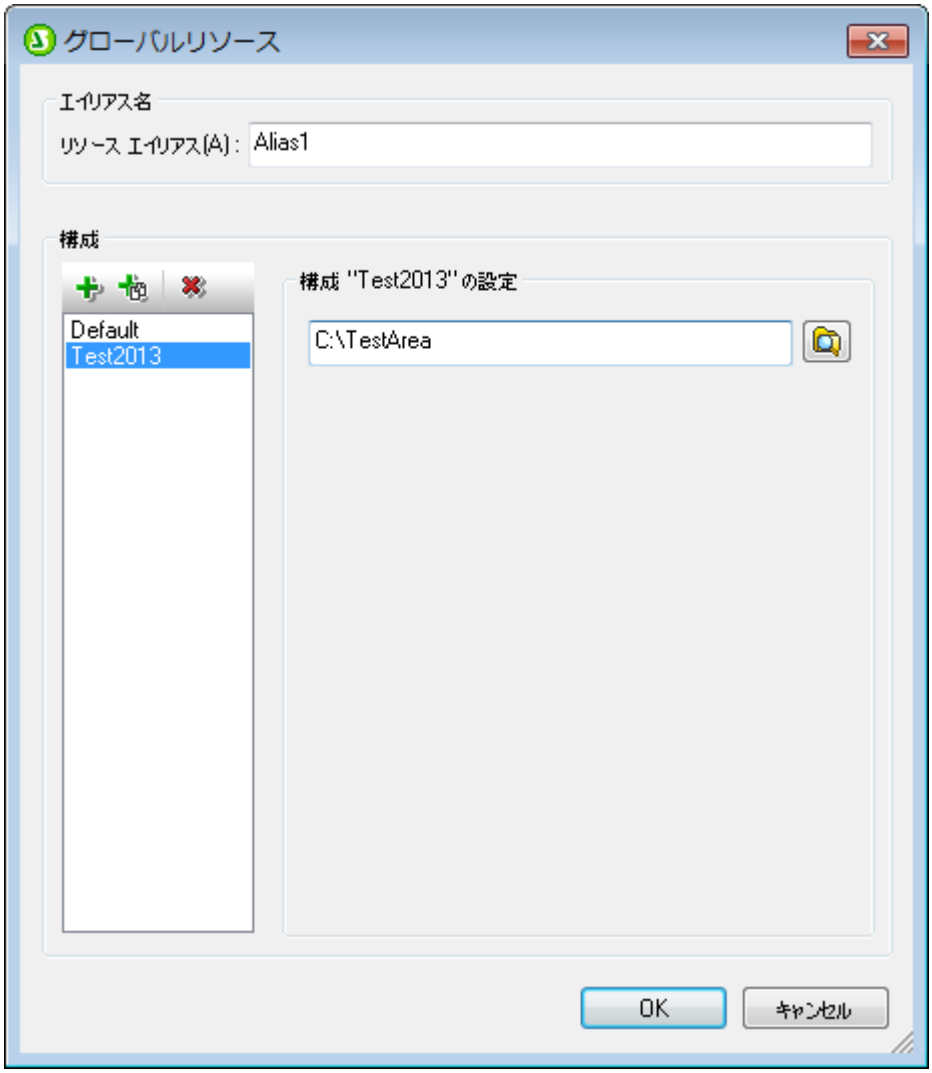
- *.sps* (SPS) ファイル: 「リソースは以下の生成された出力を指します」テキスト ボックスにて、ファイルを指定する必要があります（上のスクリーンショットを参照）。
- **入力ファイル**: 入力ファイルがSPS ファイル内部で既に指定されている場合、SPS ファイルが選択された段階で、そのファイルが入力ペインに表示されます。このエントリーは変更することができるほか、エントリーが指定されていない場合、追加する必要があります。
- **出力ファイル**: 出力ペインにて出力フォーマットを選択し、そのフォーマットに対する出力ファイルの場所を指定します。

メモ このオプション（StyleVision の結果）により、グローバルリソースが呼び出された時に変換が行われるようになります。これで、（入力ファイルから得られた）最新のデータがグローバルリソースに含まれることが保証されます。

メモ 変換を行うためにはStyleVision が使用されるため、この機能を使用するにはAltova StyleVision をインストールする必要があります。

9.1.1.2 フォルダー

フォルダーに対するグローバルリソース ダイアログでは、以下の操作によりフォルダー リソースを追加することができます（以下のスクリーンショットを参照）。



グローバルリソース ダイアログのアイコン

構成の追加: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

構成をコピーして追加: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

削除: 選択された構成を削除します。

開く: グローバルリソースとして作成されるファイルを参照します。

エイリアスの定義

以下の操作によりエイリアス(名前と構成) を定義することができます:

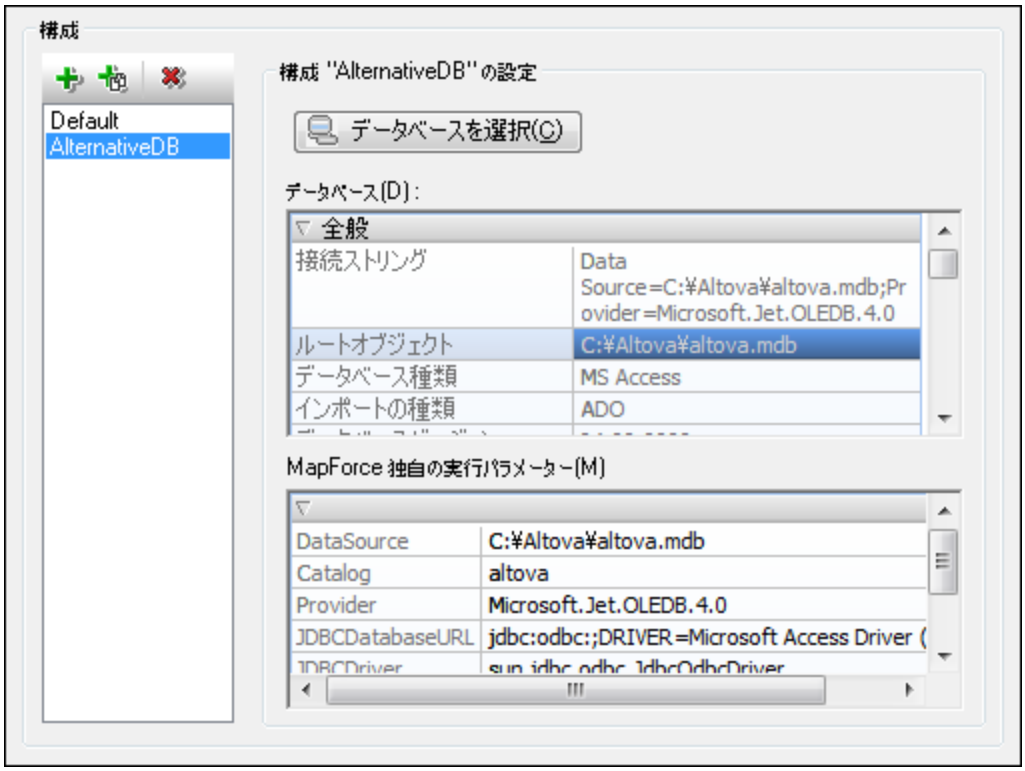
1. **エイリアスに名前を付ける**: リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。

(C) 2015-2021 Altova GmbH

2. **構成を追加する**: 構成ペインには、デフォルトで（名前変更や削除を行うことができない）Default という名前の構成が表示されます。**構成の追加**または**構成のコピー**を追加アイコンをクリックして、表示されるダイアログにて構成名を入力することで、必要なだけの構成を追加することができます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログで構成の名前を入力することで、構成が追加されます。「OK」をクリックすることで新たな構成が構成ペインに表示されます。同じ作業を必要な数だけ繰り返してください。
3. 構成のソースとなるフォルダーを選択する: 構成ペインに表示された構成を選択し、グローバルリソースとして作成するフォルダーを参照します。セキュリティ資格情報がフォルダーにアクセスするために必要とされる場合、ユーザー名とパスワードフィールドでこれらを指定してください。フォルダーにアクセスするためにセキュリティの資格条件が必要な場合、ユーザー名とパスワードフィールドを指定してください。
4. 必要に応じて複数の構成を定義する: その他の構成を必要な数だけ追加して、各構成に対してソースを指定します。各構成に対して上記のステップ 2 と3 を実行してください。エイリアスに対する新たな構成はいつでも追加することができます。
5. エイリアスの定義を保存する: 「OK」をクリックしてエイリアスと、すべての構成をグローバルリソースとして保存します。グローバルリソースが、「グローバルリソースの管理」ダイアログにあるファイル以下に表示されます。

9.1.1.3 データベース

データベースに対するグローバルリソースダイアログでは、以下の操作によりデータベースのソースを追加することができます:



グローバルリソースダイアログのアイコン

構成の追加: 追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

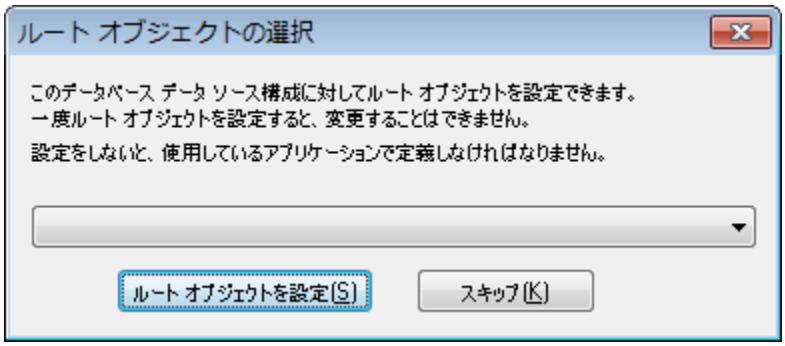
構成をコピーして追加: 選択された構成のコピーとして追加する構成の名前を入力するための、構成の追加ダイアログが表示されます。

削除: 選択された構成を削除します。

エイリアスの定義

以下の操作によりエイリアス(名前と構成) を定義することができます:

1. *エイリアスに名前を付ける*: リソース エイリアス テキスト ボックスにエイリアスの名前を入力します。
2. *構成を追加する*: 構成ペインには、デフォルトで(名前変更や削除を行うことができない) Default という名前の構成が表示されます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログにて構成名を入力することで、必要なだけの構成を追加することができます。構成の追加または構成のコピーを追加アイコンをクリックして、表示されるダイアログで構成の名前を入力することで、構成が追加されます。「OK」をクリックすることで新たな構成が構成ペインに表示されます。同じ作業を必要な数だけ繰り返してください。
3. *構成のリソースとなるデータベースを選択する*: 構成ペインに表示された構成を選択し、「**データベースを選択**」ボタンをクリックします。グローバルリソース接続を作成ダイアログが表示されます。フォルダーにアクセスするためにセキュリティの資格条件が必要な場合、ユーザー名とパスワードフィールドを指定してください。
4. *データベースへ接続する*: 接続ウィザード、既存の接続、 ADO 接続、ODBC 接続、または JDBC 接続を使用して、データベースへの接続を作成します。データベースへの接続のセクションで説明されている接続メソッドの定義を完成します。接続ウィザード、ADO 接続、または、ODBC 接続を使用することができます。 StyleVision から既にデータベースへの接続が存在する場合は、既存の接続アイコンをクリックして、表示される接続リストからデータベースを選択します。
5. *ルート オブジェクトの選択*: ルート オブジェクトを選択することのできるデータベースへ接続した場合、サーバー上にあるルート オブジェクトを選択するためのダイアログが表示されます(以下のスクリーンショットを参照)。ルート オブジェクトを選択して、「**ルート オブジェクトを設定**」をクリックしてください。選択されたルート オブジェクトが、この構成が使用された時にロードされるルート オブジェクトとなります。



ルート オブジェクトを選択しなかった場合(「**スキップ**」ボタンをクリックした場合)、グローバルリソースがロードされた際にルート オブジェクトを選択することになります。

6. *複数の構成を必要に応じて定義する*: 作成されたその他の構成に対してデータベース リソースを指定します(作成された構成に対して上記のステップ 3 から 5 までを繰り返します)。新たな構成をエイリアス定義へ追加することもできます。
7. *エイリアスの定義の保存*: 「OK」をグローバルリソースダイアログ内でクリックし、エイリアスとその構成をグローバルリソースとして保存します。グローバルリソースは、グローバルリソースの管理ダイアログ内でデータベースの下にリストされています。

9.1.2 グローバルリソースを使用する

グローバルリソースには、ファイル型、フォルダー型、そしてデータベース型と、複数の種類が存在します。StyleVision のシナリオによっては、特定のグローバルリソースの使用が許されます。例えば、ファイル型またはフォルダー型のグローバルリソースを、作業用 XML ファイルまたは CSS ファイルに対して使用することができます。また、データベース型のリソースを使用することで、新たに DB をベースとした SPS を作成することができます。 StyleVision で使用することのできる様々なシナリオについて以下で紹介されています: ファイルとフォルダー および データベース。

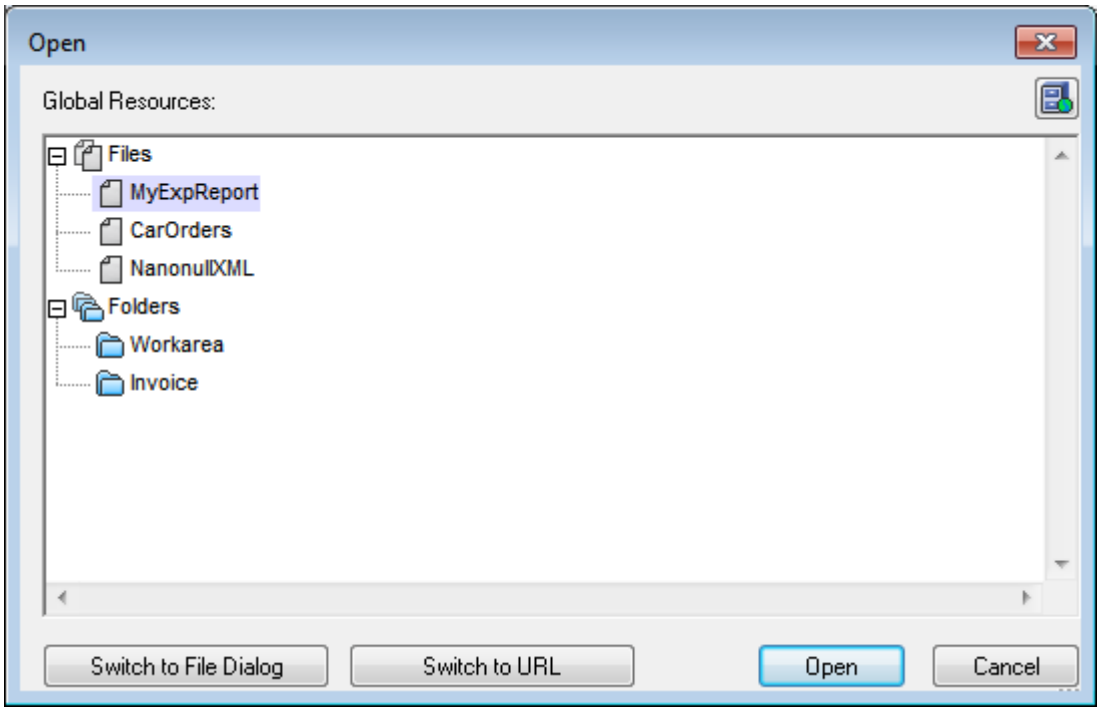
使用するリソースの選択

アプリケーション全体で使用することのできるグローバルリソースの選択、ならびに実際に使用されるグローバルリソースの選択を行うには方法が2つあります:

- 「グローバルリソース」ダイアログにてアクティブなグローバルリソース XML ファイルを選択します。アクティブなグローバルリソース XML ファイル内に収められている利用可能なグローバルリソースの定義は、アプリケーション内の開かれているファイル全てで使用することができます。使用することができるのは、アクティブなグローバルリソース XML ファイルにある定義だけとなります。アクティブなグローバルリソース XML ファイルはいつでも変更することができ、新しいファイルに収められたグローバルリソース定義が、それまで使用されていたアクティブなファイルの内容に置き換えられます。従って、アクティブなグローバルリソース XML ファイルにより、(i) 割り当て可能なグローバルリソースと、(ii) 検索可能なグローバルリソースが決定されます(例えば、あるグローバルリソース XML ファイル内のグローバルリソースに割り当てが行われているが、現在アクティブなグローバルリソース XML ファイルにて、対応する名前のリソースがない場合、割り当てられたグローバルリソースの検索を行うことはできなくなります)。
- 「ツール | アクティブな構成」 または グローバルリソース ツールバーからアクティブな構成 を選択します。メニューコマンド(またはツールバーのドロップダウンリスト)をクリックすることで、全てのエイリアスに対する構成のリストが表示されます。構成を選択することで、その構成がアプリケーションでアクティブな構成となります。この操作により、グローバルリソース(またはエイリアス)が使用される際には、各エイリアスに対して対応するアクティブな構成が適用されます。アクティブな構成は使用される全てのエイリアスに対して適用されます。アクティブな構成と同じ名前の構成をエイリアスが持たない場合、そのエイリアスに対してはデフォルトの構成が適用されます アクティブな構成はリソースの割り当てに影響を与えません。アクティブな構成は、実際にリソースが使用されるときに重要になります。

9.1.2.1 ファイルとフォルダーの割り当て

ファイル型とフォルダー型のグローバルリソースは異なる方法で割り当てられます。以下にある使用できる操作では、「グローバルリソースに切り替え」ボタンにてグローバルリソースを開くダイアログが表示されます(以下のスクリーンショットを参照)。



グローバルリソースの管理: 「グローバルリソースの管理」 ダイアログが表示されます。

ファイル型のグローバルリソースを選択することで、ファイルが割り当てられます。フォルダー型のグローバルリソースを選択することで、ファイルを開くダイアログが表示され、目的のファイルを選択することができるようになります。フォルダーから見て相対的なファイルへのパスが入力されます。従って、同じ名前のファイルを別のフォルダーに配置して、それらのフォルダーをグローバルリソースとして定義することにより、ファイルの選択切り替えを簡単に行うことができるようになります。

ダイアログ下部にあるボタンをクリックすることで、ファイルを開く、またはURL を開くダイアログへの切り替えを行うことができます。ダイアログ右上にある「グローバルリソースの管理」 アイコ からは、「グローバルリソースの管理」 ダイアログへアクセスすることができます。

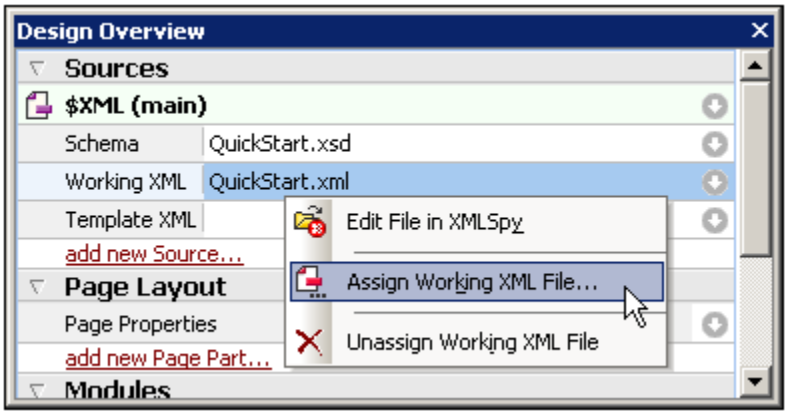
使用できるシナリオ

ファイル型やフォルダー型のグローバルリソースに対して以下のような操作を行うことができます:

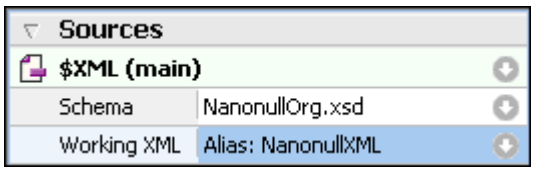
- 変更されたスキーマソース作業用 XML ファイルの追加 およびテンプレート XML ファイル
- グローバルリソースとして保存
- モジュールや CSS ファイルの追加
- グローバルリソースをプロジェクトに追加

スキーマ、作業用 XML ファイル、テンプレート XML ファイル

デザイン概要サイドバーでは、スキーマ、作業用 XML ファイル、テンプレート XML ファイルに対するコンテキストメニューから、グローバルリソースを介してスキーマ や作業用 XML ファイルを割り当てることのできる、ファイルを開くダイアログへアクセスすることができます。「**グローバルリソースにスイッチ**」 ボタンをクリックすることで、StyleVision にて現在アクティブになっているグローバルリソース XML ファイルにて定義された全てのファイル型グローバルリソースを含んだダイアログが表示されます。アクティブなグローバルリソース XML ファイルを選択する方法については、「グローバルリソースの定義」セクションを参照ください。



グローバルリソースがファイルソースとして選択された場合、デザイン概要サイドバーにてその旨が表示されます(以下のスクリーンショットを参照)。



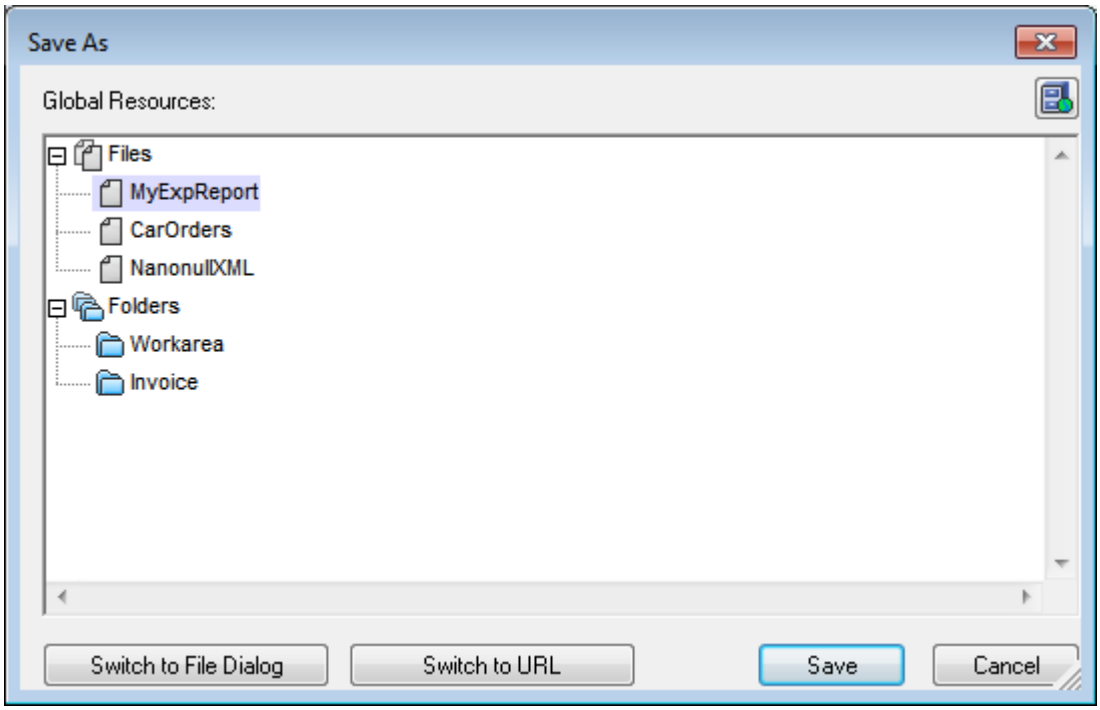
(C) 2015-2021 Altova GmbH

グローバルリソースからモジュールやCSS ファイルを追加

デザイン概要サイドバーにて**新しいモジュール**を追加または**新しいCSS ファイルを追加コマンド**をクリックすると、ファイルを開くダイアログが表示されます。グローバルリソースにスイッチをクリックして、グローバルリソースを選択することができます。

グローバルリソースとして保存

新たに作成されたファイルをグローバルリソースとして保存することができるほか、既存のファイルを開き、グローバルリソースとして保存することができます。メニューオプションから「**ファイル | デザインを保存**」または「**ファイル | 名前を付けて保存**」を選択すると、ファイルの保存ダイアログが表示されます。「**グローバルリソースにスイッチ**」ボタンをクリックすることで、グローバルリソースXML ファイルにて定義されているグローバルリソースのエイリアスを含むダイアログが表示されます。



エイリアスを選択し、「**保存**」をクリックしてください。選択したエイリアスがファイルエイリアスの場合、そのファイルへ直接保存されます。フォルダーエイリアスが選択された場合、保存するファイルの名前を入力するダイアログが表示されます。どちらの場合でも、現在アクティブな構成にて定義された場所にファイルは保存されます。

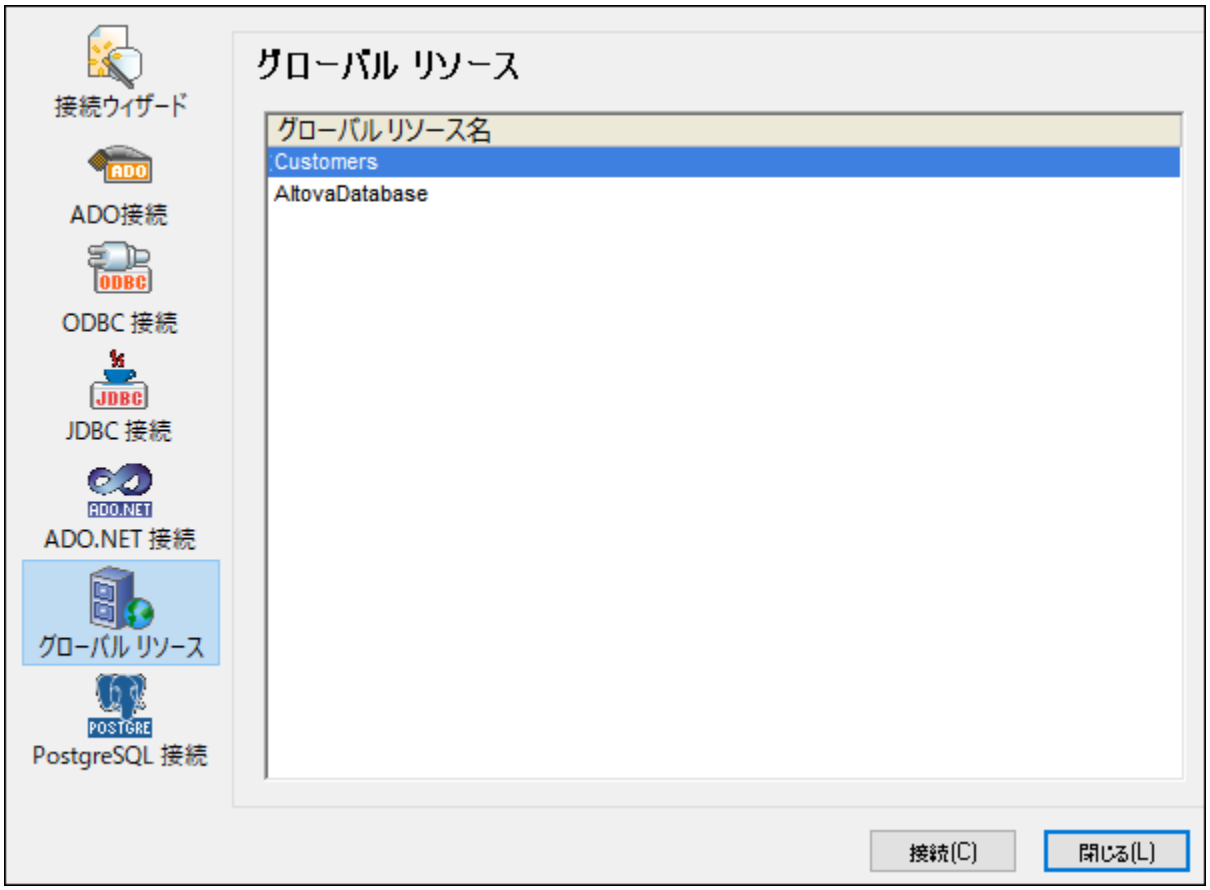
メモ: 各構成は、その構成定義にて指定された場所を指し示すことになります。グローバルリソースとして保存するファイルの拡張子が、構成により指定された場所にあるファイルの拡張子と異なる場合、このグローバルリソースがStyleVision にて開かれた時に編集または検証エラーが発生する可能性があります。これは、StyleVisionが構成の定義にて指定されたファイル形式によりファイルが開かれるためです。

プロジェクトにおけるグローバルリソース

メニューオプションの「**プロジェクト | プロジェクトにグローバルリソースを追加**」を選択することで、現在アクティブなプロジェクトにグローバルリソースを追加することができます。現在アクティブなグローバルリソースXML ファイルにて定義されたグローバルリソースがダイアログにて表示されます。目的のグローバルリソースを選択し、「**OK**」をクリックすると、そのグローバルリソースがプロジェクトに追加されます。グローバルリソースがプロジェクトサイドバーに表示され、その他のファイルと同様に使用することができます。

9.1.2.2 データベースの割り当て

メニューオプションの「ファイル | 新規作成 | データベースから新規作成」コマンドによりデータベース(DB) からSPS を作成する場合、グローバルリソースを使用するオプションを選択することができます(以下のスクリーンショットを参照)。メニューにあるデータベースに関連するコマンドでも、データベース型のグローバルリソースを使用することができます。



データベースを開くダイアログにてグローバルリソースアイコンをクリックすると、現在アクティブなグローバルリソースXML ファイルに定義されている全てのデータベース型グローバルリソースが表示されます。目的のグローバルリソースを選択して、「接続」をクリックします。選択されたグローバルリソースに2つ以上の構成が割り当てられている場合、現在アクティブな構成に対応したグローバルリソースが使用され(「ツール | アクティブな構成」またはグローバルリソースツールバーを確認してください)、接続が確立されます。DB データ選択に記述されているように、使用するデータ構造やデータを選択する必要があります。

9.1.2.3 構成の切り替え

同時にアクティブにすることのできる構成は1つだけで、その設定はアプリケーション全体に対して適用されます。これは、アクティブな構成が、現在開いている全てのファイルやデータソース接続のエイリアスに対して有効であることを意味します。エイリアスにアクティブな構成と同じ名前の構成が割り当てられていない場合、デフォルトの構成がそのエイリアス対して使用されます。構成を変更する例として、XSLT ファイルが複数の構成を持つグローバルリソースを介してXML ドキュメントに割り当てられたと仮定しましょう。各構成は異なるファイルにマップされます。ですから、ファイルの選択はアプリケーションのアクティブな構成としてどの構成が選択されているかにより決まります。

構成の切り替えるには2つの方法があります:

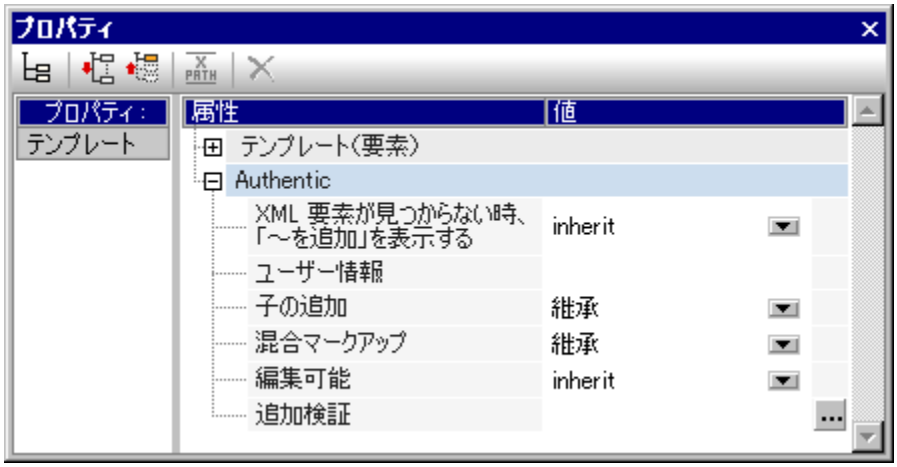
- メニューコマンド「**ツール | アクティブな構成**」を選択すると、グローバルリソースXML ファイル内にある全ての構成がサブメニューに表示されます。目的の構成を選択します。
- グローバルリソースコンボボックスのツールバーから、目的の構成を選択します(以下のスクリーンショットを参照)。

アクティブな構成を切り替えるという方法により、グローバルリソースを介して割り当てられたソースファイルを変更することができます。

9.2 Authentic ノードプロパティ

Authentic ノードプロパティとは、**Authentic View** にて表示するためにセットするプロパティのことです。例えば、XML マークアップとともにノードを表示し、読み取り専用として定義し、カーソルがノード上に配置された際にユーザー情報を表示するといった動作が可能になります。Authentic ノードプロパティは様々な SPS コンポーネントに対してセットすることができます。使用することができるプロパティの種類は、コンポーネントの種類により異なります。

Authentic ノードプロパティの割り当てを行うには、目的のコンポーネントをデザインにて選択します。プロパティサイドバーにある Authentic プロパティグループにて、目的の Authentic ノード設定を指定します（以下のスクリーンショットを参照）。その他にも、ノードテンプレートを右クリックして、**Authentic プロパティの編集**を選択することもできます。



Authentic プロパティの定義

以下のノード設定により、Authentic View における個々のノードの振る舞いを指定することができます。

子の追加
要素のノードを選択した際に、この設定を利用することができます。選択された要素が追加された際に挿入される子要素を定義することができます。全ての子要素、必須要素、挿入する子要素なしのオプションを選択することができます。

追加検証
スキーマに対する XML ドキュメントの検証に加え、個々のノードや自動計算に対して更なる検証を指定することもできます。XPath 条件式を定義することで、ノードや自動計算の XML コンテンツに対して妥当とする範囲を指定することができます。ノードの XML 値が妥当で無い場合、XML 検証が行われた際にエラーとして Authentic View ユーザーへ知らされます。詳細については追加検証 のセクションを参照ください。

表示または取得される DB 行
表示される DB 行と取得される DB 行プロパティでは、Authentic View に表示される DB の行数と、ロードされる DB の行数がそれぞれ定義されます。取得される DB 行プロパティにより、ロードと表示に必要な時間を短縮することができます。

編集可能

ノードが編集可能かどうかを指定します。デフォルトでノードは編集可能となっています。要素、属性、コンテンツのノードを選択した際に、この設定を利用することができます。自動計算の結果は自動計算の XPath 条件式により生成されるため、編集することができず、このオプションを自動計算に対して使用することはできません。

混合マークアップ

要素または属性のノードを選択した際にこの設定を利用することができ、Authentic View の混合マークアップモードにて個々のノードがどのようにマークアップされるかを指定することができます。大きなマークアップ、小さなマークアップ、マークアップを隠す、というオプションを選択することができます。

XML 要素が見つからない時、「～を追加」を表示
選択された要素や属性が存在しない場合、「[要素/属性]を追加」という文字が Authentic View に表示されるかどうかを指定します。デフォルトでは文字が表示されます。要素または属性のノードを選択した際にこの設定を利用することができます。

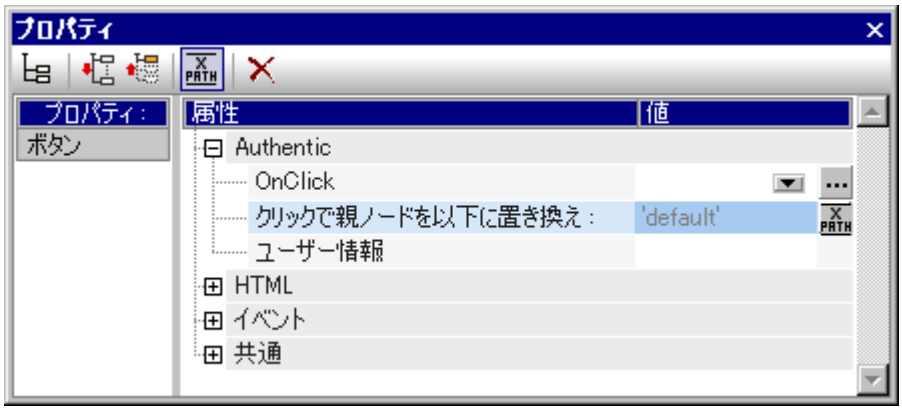
ユーザー情報
ノード上にマウスポインターが配置された時に、テキストボックスに入力されたテキストがツールチップとして表示されます。要素、属性、コンテンツ、自動計算のノードに対してこの機能を利用することができます。要素と属性、そしてコンテンツノードにユーザー情報が与えられている場合、コンテンツノードのユーザー情報が表示されます。

関連項目

- 追加検証
- プロパティサイドバー
- Authentic View 内での編集

9.3 クリックによる親ノードの置き換え

(プロパティサイドバーにある Authentic プロパティグループ以下の) **クリックで親ノードを以下に置き換え** プロパティを使用することで、ボタン または ハイパーリンクの親ノードにあるテキストコンテンツを指定することができます。



Authentic View ユーザーによりデザインコンポーネント (ボタンまたは ハイパーリンク) がクリックされると、SPS デザイナーにより指定されたテキストが、デザインコンポーネントの親ノードの XML コンテンツとして挿入されます。この機能を使用することで、Authentic View ユーザーが行った選択 (ボタンや ハイパーリンクのクリック) により、Authentic View における XML ドキュメントの表示を変更することが可能になります。

使用メカニズム

以下のようにしてこの機能を使用することができます:

1. デザインにて、テキストコンテンツを挿入するノード内部にデザインコンポーネント (ボタンやハイパーリンク) を挿入します。
2. デザインにてデザインコンポーネントが選択された状態で、プロパティサイドバーの**プロパティ**ペインにてそのデザインコンポーネントを選択し、(Authentic プロパティグループ以下にある) **クリックで親ノードを以下に置き換え**のカラムに対して値を入力します (上のスクリーンショットを参照)。値はテキスト値として直接入力することができるほか、XPath 条件式を介して値を入力することもできます (プロパティサイドバーのツールバーにある XPath アイコンをクリックして、XPath 条件式として値を入力することができます)。上のスクリーンショットでは、'default' という XPath 条件式の値が入力されています。Authentic View にてデザインコンポーネントがクリックされると、デザインコンポーネントの親ノードに対して default というテキスト値が入力されます。クリックで親ノードを以下に置き換えプロパティにて default という (XPath 条件式ではない) テキスト値を入力することでも、同様の結果を得ることができます。
3. Authentic View にてそのデザインコンポーネントがクリックされることで、親ノードのコンテンツに入力される値が指定されるため、親ノードのテキストコンテンツをベースにした処理を行うことが可能になります。例えば、条件を使用することにより、親ノードのコンテンツが default となる場合に、デフォルトのフォーマットを適用し、コンテンツが hide となった場合にノードを隠すといった処理を定義することができます。この機能を使用することで、通常であればスクリプトのデプロイが必要になるようなパワフルな処理を行うことが可能になります。

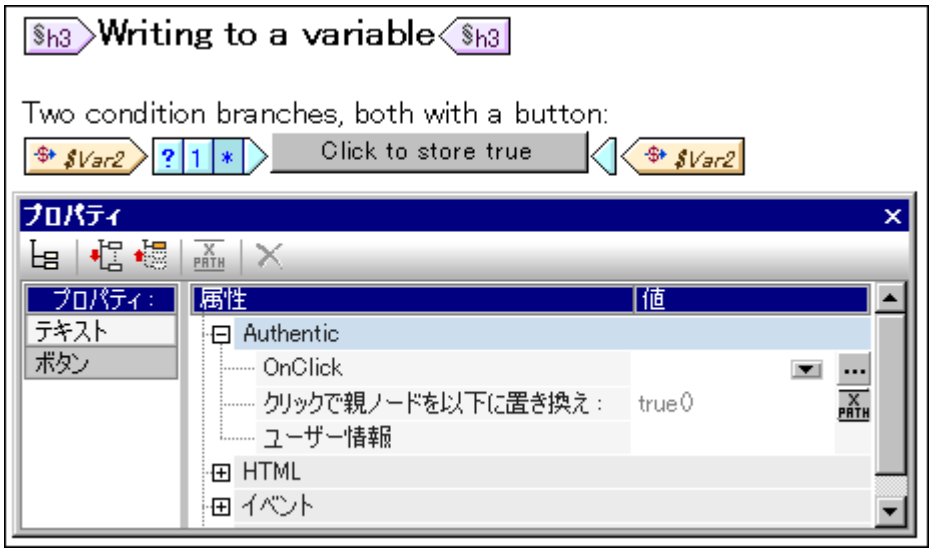
メモ この機能を正しく使用するには、親ノードに対してテキストコンテンツが許可されるようにスキーマにて定義する必要があります。

変数の使用

親ノードは、祖先ノードにおいて定義された変数 として使用することもできます。この場合、**クリックで親ノード以下に置き換え** プロパティにより変数の値を指定することができます。Authentic View にてデザインコンポーネント (ボタン または ハイパーリンク) がクリックされると、そのデザインコンポーネントの**クリックで親ノード以下に置き換え** プロパティで定義された値が変数に代入されます。

簡単な例を以下に示します。変数の Var2 の値が true() にセットされている場合、値に false() をセットするためのボタンが表示されます (以下のスクリーンショットを参照)。この処理は 条件における最初の分岐にて指定されています。

条件にあるそれ以外の分岐では、変数の値に true() をセットするためのボタンが表示される処理が定義されています（以下のスクリーンショットを参照）。変数 Var2 の値（true() または false()）をベースにしたその他の処理を記述することも可能になります。



これにより、Authentic View にて選択されたオプションにより、対応する処理が実行されるような SPS デザインを作成することが可能になります。

関連項目

- ボタンとハイパーリンク
- 条件
- 変数

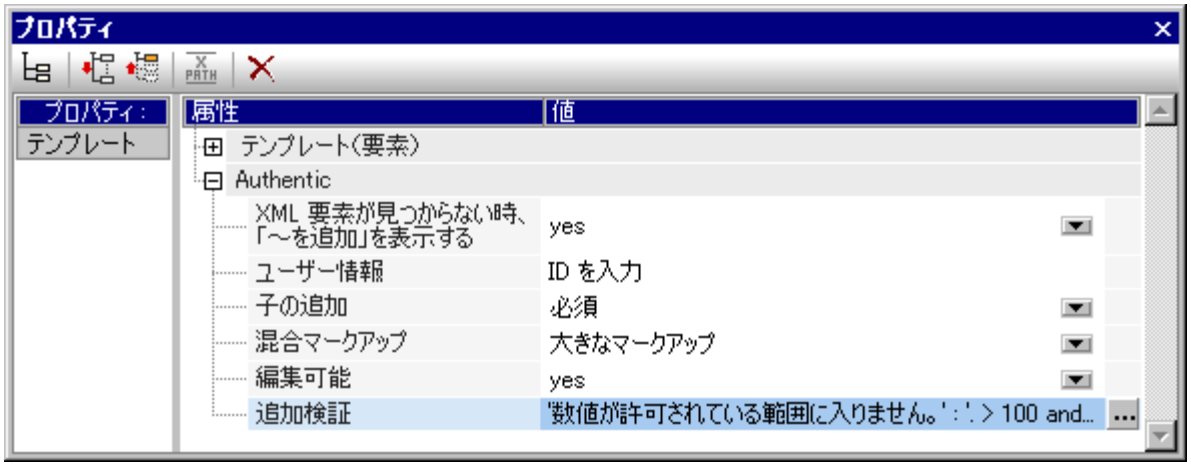
9.4 追加検証

要素、属性、コンテンツ(コンテンツプレースホルダ)、データ入力デバイス、または自動計算ノードのコンポーネントに対して追加検証を設定することができます。XPath 条件式により、ノードや自動計算のXML コンテンツが妥当とする範囲をセットすることができます。定義された範囲に収まらないXML 値は不正な値となります。ノードのXML 値が不正となる場合、(F8 キーを押下することにより)XML ドキュメントの検証を行った際にエラーを表示することでAuthentic View ユーザーへその旨を知らせることができます。表示されるエラーメッセージは、追加検証設定のエラーメッセージフィールドにて入力されたテキストとなります。

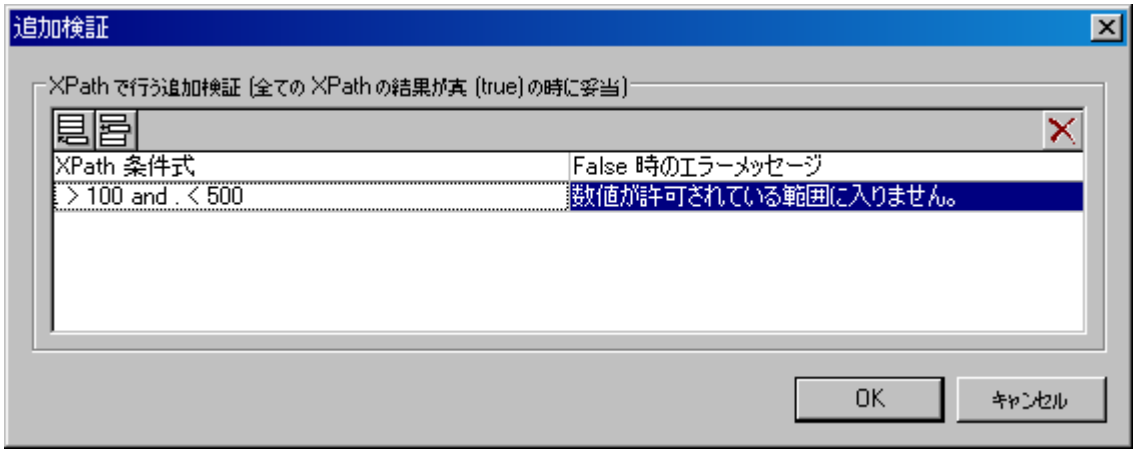
追加検証設定を行う

以下の操作により追加検証をセットすることができます:

1. 追加検証を作成するコンポーネントを選択します。
2. プロパティサイドバーの Authentic プロパティグループにて、追加検証プロパティの**編集**ボタン ... をクリックします(以下のスクリーンショットを参照)。追加検証ダイアログが表示されます。



3. 追加検証ダイアログにて、ペインの左上にある**追加**ボタンをクリックして、追加検証エントリーの行を追加します(以下のスクリーンショットを参照)。



4. XPath 条件式カラムにて、そのコンポーネントにおける XML データが妥当となる値の範囲を XPath 条件式により入力します。
5. データが不正となった際に表示されるエラーメッセージを入力します。

(C) 2015-2021 Altova GmbH

6. 「OK」をクリックして完了します。

関連項目

- プロパティサイドバー

9.5 パースされていないエンティティ URI

DTD を使用していて、その中でパースされていないエンティティを宣言した場合、そのエンティティに関連付けられている URI をSPS 内にあるイメージやハイパーリンクのターゲットとして使用することができます。この機能により、同じ URI をSPS にて複数回使用することが可能になります。この機能では、DTD にあるパースされていないエンティティのURI を出力、すなわち、Authentic View を除く(HTML、RTF へ渡すために、unparsed-entity-uri XSLT 関数が使用されます。

この機能を使用するには、以下の様に、DTD、XML ドキュメント、SPS ドキュメントを適切に編集する必要があります:

1. DTD では、(i) URI と (ii) (エンティティのリソースタイプに StyleVision が指定された) 記述により、パースされていないエンティティを宣言する必要があります。
2. XML ドキュメントでは、パースされていないエンティティの名前を与えることにより、エンティティの参照を行う必要があります。
3. SPS では、動的ノードの値をパースされていないエンティティとして適切にアクセスすることでイメージとハイパーリンクのターゲットとして使用することができます。.

パース去れていないエンティティの宣言と参照と参照

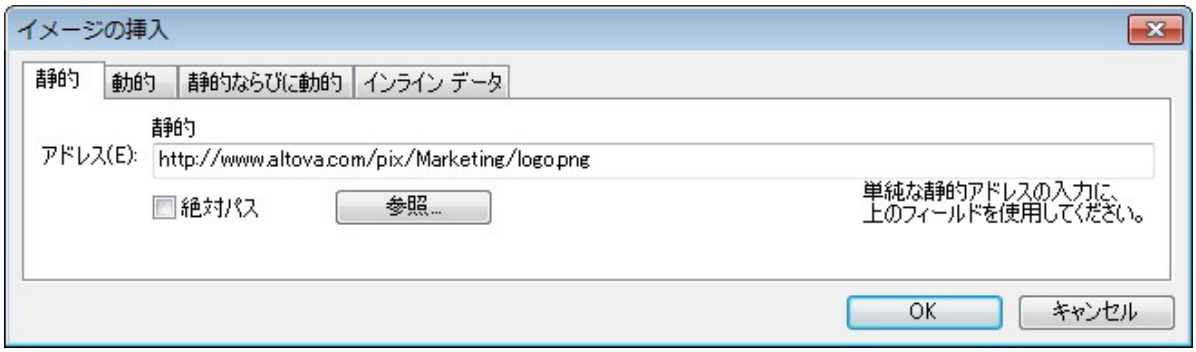
以下にXML ドキュメントの一部を示します。内部 DTD サブセットにはパースされていないエンティティが2つ含まれており、1つにはGIF 表記が、もう1つにはLNK 表記が含まれています。XML コードにある img/@src ならびに link/@href ノードにて名前を与えることにより、パースされていないエンティティへの参照が行われます。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Document SYSTEM "UEURIDoc.dtd" [
<!ENTITY Picture SYSTEM "nanonull.gif" NDATA GIF>
<!ENTITY AltovaURI SYSTEM "http://www.altova.com" NDATA LNK>
]>
<document>
   <header>Example of How to Use Unparsed Entity URIs URI</header>
   <para>...</para>
   <img src="Picture"/>
   <link href="AltovaURI">Link to the Altova Website.</link>
</document>
```

パース去れていないエンティティを使用する SPS イメージとハイパーリンク

以下の操作により、SPS 内でパースされていないエンティ URI を参照するイメージやハイパーリンクを使用することができます:

1. メニューオプションの「**挿入**」から、イメージやハイパーリンクを挿入します。
2. 編集ダイアログにて動的タブプロパティを選択し、パースされていないエンティティの名前を含んでいるノードを選択する XPath 条件式を入力します(以下のスクリーンショットを参照)。上の XML ドキュメントサンプルの場合、XPath 条件式には //img/@src ならびに //link/@href ノードを入力することになります。



3. ダイアログの下部にある「XSLT 変換に対して分析されていないエンティティとして扱う」のチェックボックスにチェックを入れます。これで選択されたノードのコンテンツがパースされていないエンティティとして読み込まれることになります。指定された名前のエンティティが宣言されている場合、そのエンティティに関連付けられている URI が、リソース（イメージまたはハイパーリンク）の指定に使用されます。

スタイルシートの処理が行われる際には、エンティティ名に関連付けられる名前がエンティティ名の代わりに使用されます。

メモ URI が相対的な URI となる場合、XSLT プロセッサーにより DTD のベース URI により絶対 URI への展開が行われます。従って、パースされていないエンティティが相対 URI の 'nanonull.gif' により定義されており、DTD が someFolder 以下に配置されている場合、この URI は file:///c:/someFolder/nanonull.gif となります。

関連項目

- *イメージの使用*
- *ハイパーリンクの定義*

9.6 XSLT、XSL-FO または FO ファイルからの新規作成

SPS デザインは、HTML の出力を行うためのXSLT ファイルや、PDF ならびにFO ファイルを出力するためのXSL-FO コマンドを伴う既存のXSLT ファイルをベースに作成することができます。つまり、1からデザインを作成する代わりに、既存のXSLT ファイルをデザインのひな形として使用することができます。

XSLTステップ からSPS を作成する

以下の操作により、XSLT、XSLT-for-FO、またはFO ファイルからSPS ファイルを作成することができます:

1. メニューコマンドから「**ファイル | 新規作成 | XSLT、XSL-FO または FO ファイルから新規作成...**」を選択します。
2. ファイルを開くダイアログにて、目的のファイルを選択します。
3. 次に表示されるダイアログにて、SPS がベースとして使用するスキーマを選択するよう促されます。目的のスキーマを選択してください。
4. XSLT または FO ファイルの構造やフォーマットをベースにした SPS が作成され、デザインビューに表示されます。
5. あとは通常の方法で SPS を修正することができます。例えば、スキーマツリー からノードをドラッグし、スタイルや表示方法の修正や新たなスタイルの追加、自動計算 と条件テンプレートといったStyleVision の機能を使用することができます。
6. SPS を保存して、作業用 XML ファイルを使用した 多種の出力の書式 のプレビューを行うことができます。更に生成されたファイルの保存 コマンドにより、スタイルシートと出力ファイルの生成 を行うことができます。

サンプル

以下に記されるサンプルは、(マイ) ドキュメントフォルダー 以下にある、`C:\Documents and Settings\<username>\My Documents\Altova\StyleVision2021\StyleVisionExamples/Tutorial/NewFromXSLT` に収められています。このフォルダーには、SimpleExample.xslt、SimpleExample.xsd そして SimpleEample.xml が収められています。

XML ファイルの内容を以下に示します。

XML file used in charts example: YearlySales.xml

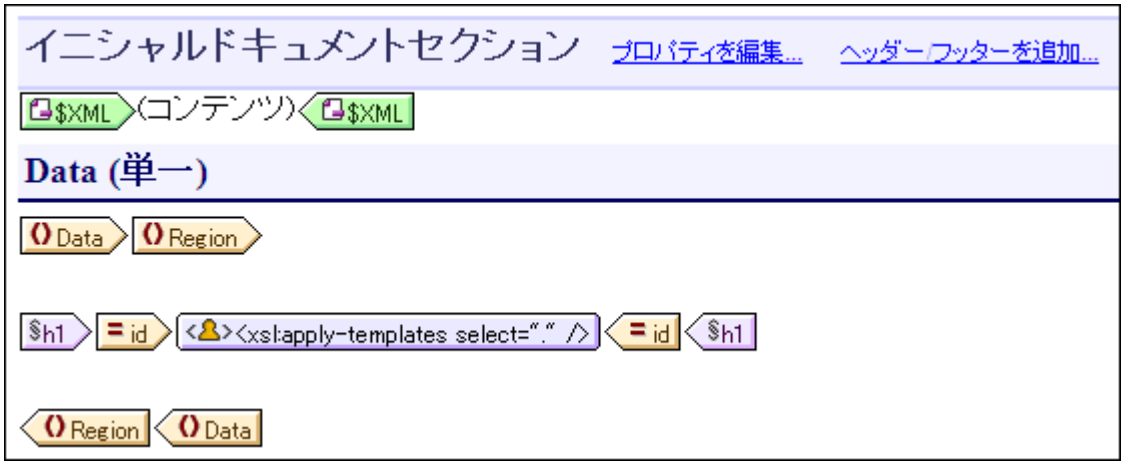
```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
        <ChartType>Pie Chart 2D</ChartType>
        <Region id="Americas">
                <Year id="2005">30000</Year>
                <Year id="2006">90000</Year>
                <Year id="2007">120000</Year>
                <Year id="2008">180000</Year>
                <Year id="2009">140000</Year>
                <Year id="2010">100000</Year>
        </Region>
        <Region id="Europe">
                <Year id="2005">50000</Year>
                <Year id="2006">60000</Year>
                <Year id="2007">80000</Year>
                <Year id="2008">100000</Year>
                <Year id="2009">95000</Year>
                <Year id="2010">80000</Year>
        </Region>
        <Region id="Asia">
                <Year id="2005">10000</Year>
                <Year id="2006">25000</Year>
                <Year id="2007">70000</Year>
```

```
                <Year id="2008">110000</Year>
                <Year id="2009">125000</Year>
                <Year id="2010">150000</Year>
        </Region>
</Data>
```

XSLT ファイルの内容を以下に示します。

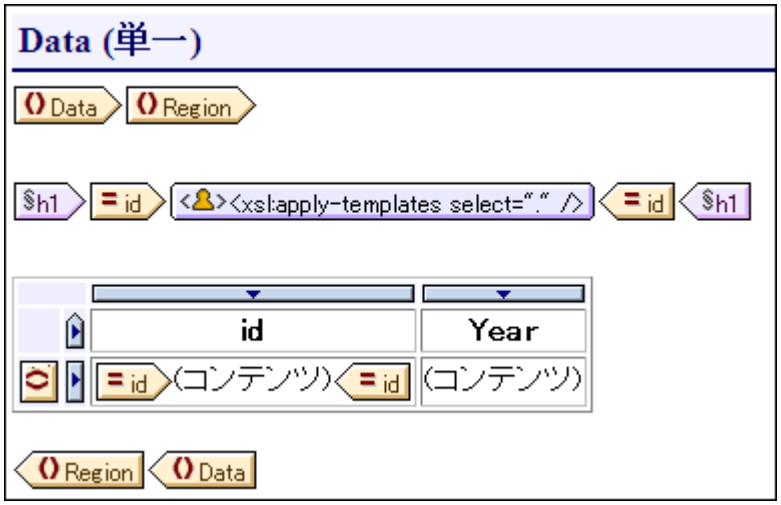
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:fn="
http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
   <xsl:template match="/">
      <html>
         <head>
            <title>Simple Example for New From XSLT</title>
         </head>
         <body>
            <xsl:apply-templates/>
         </body>
      </html>
   </xsl:template>
   <xsl:template match="Data">
      <xsl:for-each select="Region">
         <h1 style="color:red">
            <xsl:apply-templates select="@id"/>
         </h1>
      </xsl:for-each>
   </xsl:template>
</xsl:stylesheet>
```

上記のステップ1から4の操作を行うことにより、以下に示されるような SPS がデザインビューに表示されます:



SPS には2つのテンプレートが作成されている点に注目してください。HTML プレビューに切り替えを行うと、h1 要素のスタイル(color:red;)がSPS へ渡されていることが確認できます。

デザインビューにて h1 要素を選択し、(スタイルサイドバーにある色プロパティグループにて) color の値を black へ変更します。その後スキーマツリーから Year 要素をドラッグして、以下のスクリーンショットで示される場所にてテーブルを作成します。Year ID が最初のカラムに表示されるように、カラムの修正を行います。



ドキュメントのコンテンツ、構造、表示方法プロパティにその他の変更を加え、プレビューにてその出力を確認した後に、生成されたファイルの保存 コマンドを使用することでファイルを保存することができます。

関連項目

- SPS ファイル コンテンツ
- SPS ファイル 構造
- SPS ファイル プレゼンテーション

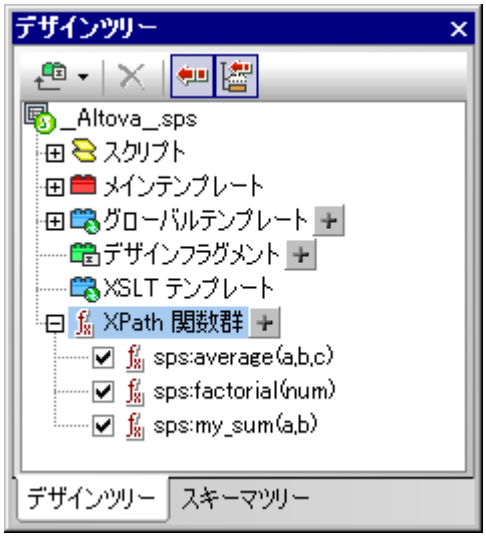
(C) 2015-2021 Altova GmbH

9.7 ユーザー定義 XPath 関数

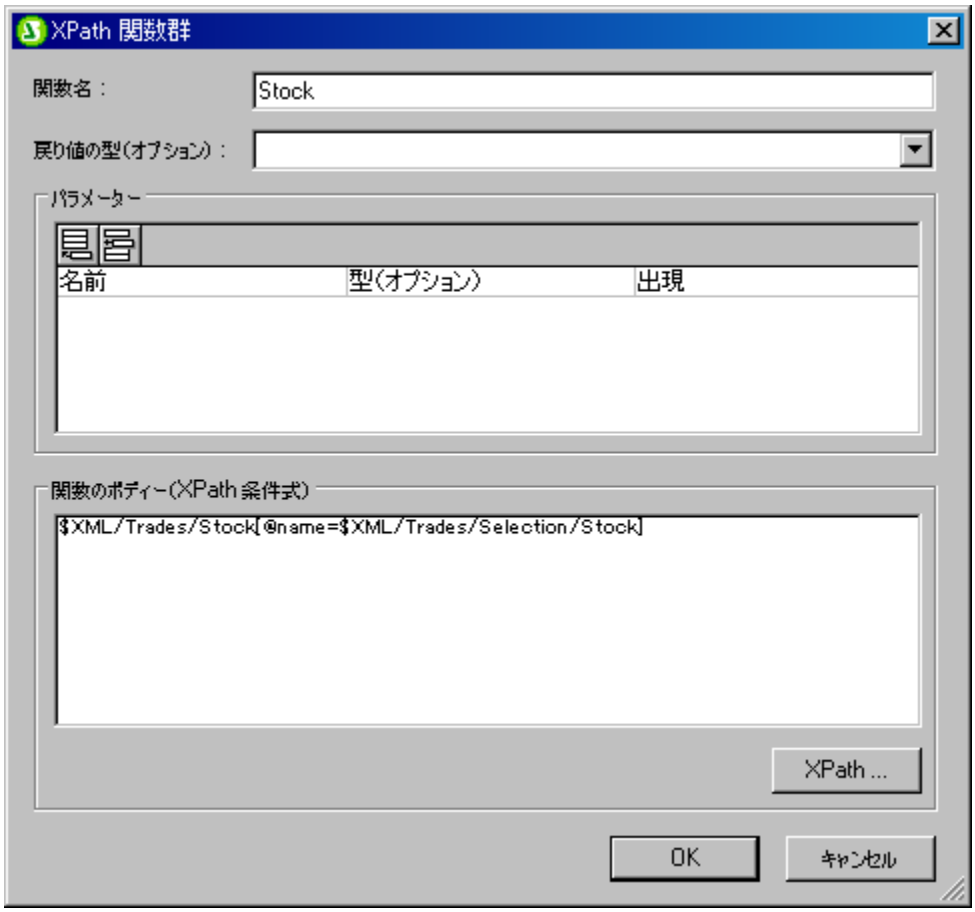
SPS ではカスタマイズされた **XPath 2.0/3.0** 関数を定義することができます。ユーザー定義された XPath 関数は、XPath 条件式を使用することができる任意のデザインコンポーネント（例えば自動計や条件、コンボボックスなど）にて再利用することができます。

ユーザー定義 XPath 関数の定義と編集

ユーザー定義 XPath 関数は、スキーマツリーサイドバーまたはデザインツリーサイドバーにて作成（そして編集）することができます（以下のスクリーンショットを参照）。SPS 内にある全てのユーザー定義 XPath 関数はスキーマツリーならびにデザインツリーサイドバーにある XPath 関数アイテム以下に表示され、両方のサイドバーからアクセスすることができます。



ユーザー定義 XPath 関数を作成するには、XPath 関数アイテムにある アイコンをクリックします。XPath 関数ダイアログが表示されます（以下のスクリーンショットを参照）。既に作成された関数の編集を行うには、サイドバーの XPath 関数リストに表示されているエントリーをダブルクリックします。XPath 関数ダイアログが表示され、関数の定義を編集することができます（以下のスクリーンショットを参照）。



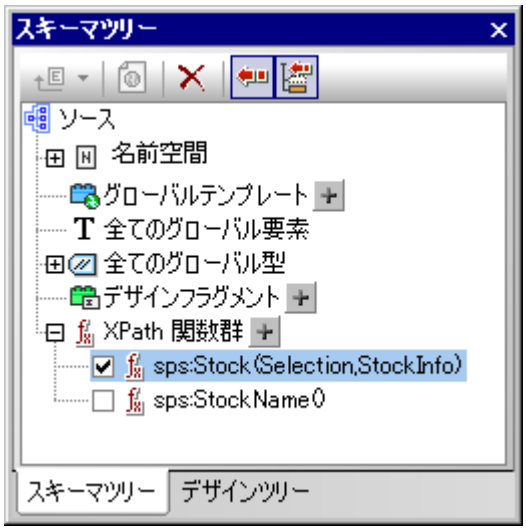
ユーザー定義 XPath 関数が作成されると、その関数をデザインにおける任意の場所で使用することができるようになります。

ユーザー定義 XPath 関数の名前空間

ユーザー定義 XPath 関数は、http://www.altova.com/StyleVision/user-xpath-functions 名前空間にて作成されます。この名前空間にはsps: というプレフィックスが与えられており、ユーザー定義 XPath 関数を呼び出すには、この名前空間プレフィックスを使用する必要があります(例: sps:MyFunction())。

ユーザー定義 XPath 関数の有効化と無効化

ユーザー定義 XPath 関数エントリーの左にあるチェックボックスにチェックを入れる / 外すことで、そのユーザー定義 XPath 関数を、それぞれ有効化 / 無効化することができます(以下のスクリーンショットを参照)。



同じ名前の関数が2つ以上存在するような場合に、この機能を使用することができます。インポートされたSPS モジュール内部に同じ名前の関数が含まれているような時に、同名の関数が表示されることがあります。

ユーザー定義 XPath 関数を呼び出す

ユーザー定義 XPath 関数は、デザイン内にあるXPath 条件式から呼び出すことができます。例えば、上記で定義された sps:MyFunction というユーザー定義 XPath 関数を、自動計算内部にある以下の様な XPath 条件式により呼び出すことができます:

```
sps:MyFunction()/@name.
```

このXPath 条件式は、以下のように評価されます:

1. sps:MyFunction() が評価されます。関数が以下の様に定義されていると仮定しましょう: $XML/Trades/Stock[@name=$XML/Trades/Selection/Stock]。関数が評価されると、/Trades/Selection/Stock 要素のコンテンツにマッチした name 属性の値を持った /Trades/Stock 要素が返されます。
2. ステップ1の結果が関数呼び出し元の XPath 条件式へ戻されます。/Trades/Stock 要素にある name 属性の値が、自動計算の値として返されます。

関数の削除

関数を削除するには、スキーマツリーまたはデザインツリーサイドバーに表示されているXPath 関数を選択して、サイドバーのツールバーにある**アイテムの削除**アイコンをクリックします。または、XPath 関数を右クリックして、コンテキストメニューから**アイテムの削除**を選択することもできます。

9.7.1 XPath 関数の定義

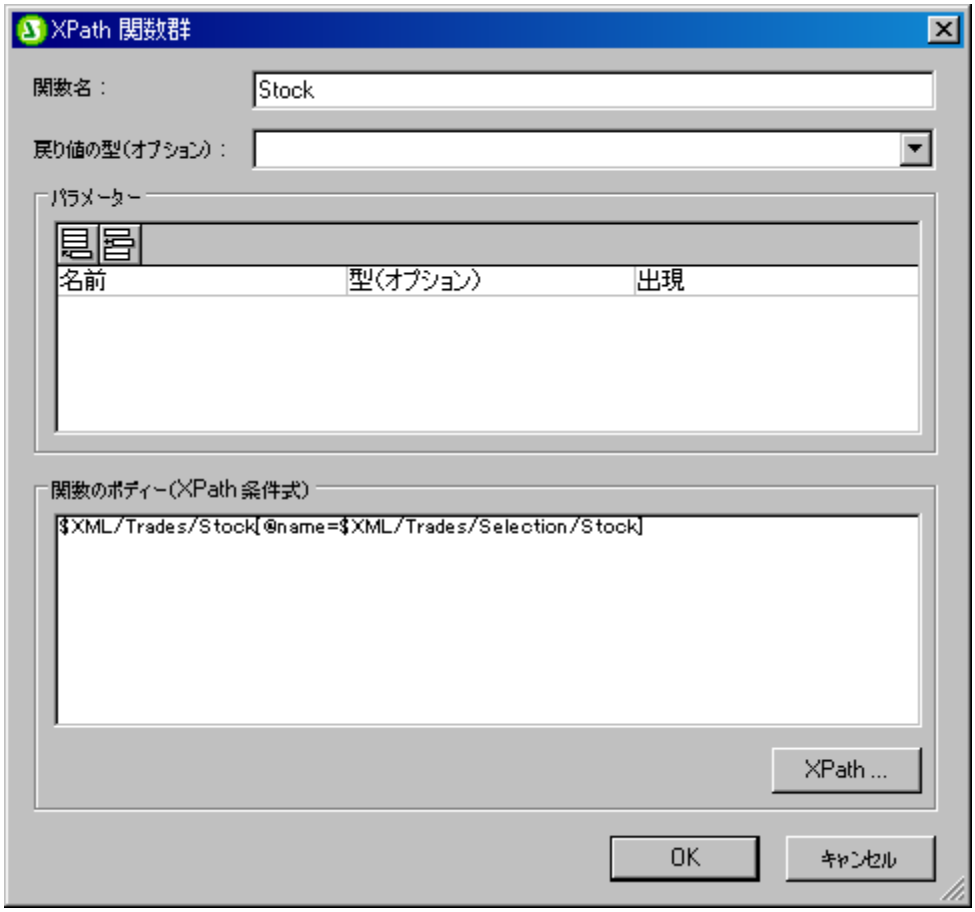
ユーザー定義 XPath 関数には、(i) 名前(テキスト文字列)と、(ii) 定義(XPath 条件式)の両方が必要になります。

更に、関数に対して1つ以上の**引数**(パラメータ)を指定することができます。ユーザー定義 XPath 関数には**戻り値の型**を(戻り値の型コンボボックスのドロップダウンリストから型を選択することにより)オプションとして指定することもできます。データ型を指定することにより、戻り値の値が選択されたデータ型に従ったものなのかをチェックすることができます。選択されたデータ型に戻り値が変換される訳ではない点に注意してください。型のミスマッチが発生した場合、エラーが返されます。戻り値のデータ型が指定されなかった場合、データ型のチェックが行われることはありません。

ユーザー定義 XPath 関数が作成されると、デザイン内の任意の場所で使用することができるようになります。XSLT スタイルシートでは、(以下に示されるように) xsl:stylesheet 要素の子要素である xsl:function 要素として作成されます。

```
<xsl:stylesheet>
   ...
   <xsl:function name="sps:Stock">
       <xsl:sequence select="$XML/Trades/Stock[@name=$XML/Trades/Selection/Stock]"/>
   </xsl:function>
   <xsl:function name="sps:Average" as="xs:decimal">
       <xsl:param name="a" as="xs:integer"/>
       <xsl:param name="b" as="xs:integer"/>
       <xsl:param name="c" as="xs:integer"/>
       <xsl:sequence select="avg( ($a, $b, $c) )"/>
  </xsl:function>
</xsl:stylesheet>
```

以下のスクリーンショットに示される sps:Stock 関数からは、/Trades/Selection/Stock 要素のコンテンツにマッチする name 属性の値を持った /Trades/Stock 要素が返されます。上に示される sps:Average 関数からは、3つの入力パラメーター値の平均が返されます。関数の定義では、avg() という XPath 2.0 の関数が使用されます。戻り値のデータ型には avg() 関数において入力値の xs:integer データ型の値が評価された際に返される xs:decimal が指定されています。戻り値のデータ型を指定することで、値がそのデータ型に従ったものであるかのチェックが行われ、そうでない場合はエラーが返されます。



関数の定義

関数の定義を行うには、スキーマツリーまたはデザインツリーにある XPath 関数アイテムの アイコンをクリックします（上のスクリーンショットを参照）。既に作成された関数を編集するには、サイドバーに表示されている XPath 関数をダブルクリックしてください。関数の名前を入力して、その定義を関数のボディーペインに入力します。必要な場合は、引数の定義も行うことができます（詳細については、パラメーターとシーケンス と パラメーターとノード のセクションを参照ください）。関数の戻り値に対してデータ型を指定することもできます。

XPath 関数を構成する XPath 条件式を記述する際に最も注意すべき点は、XPath 条件式に対して**コンテキストノードが与えられていない**ということです。XPath 条件式によりノードの位置を特定する必要がある場合、以下の方法のどれかによりコンテキストノードを得ることができます:

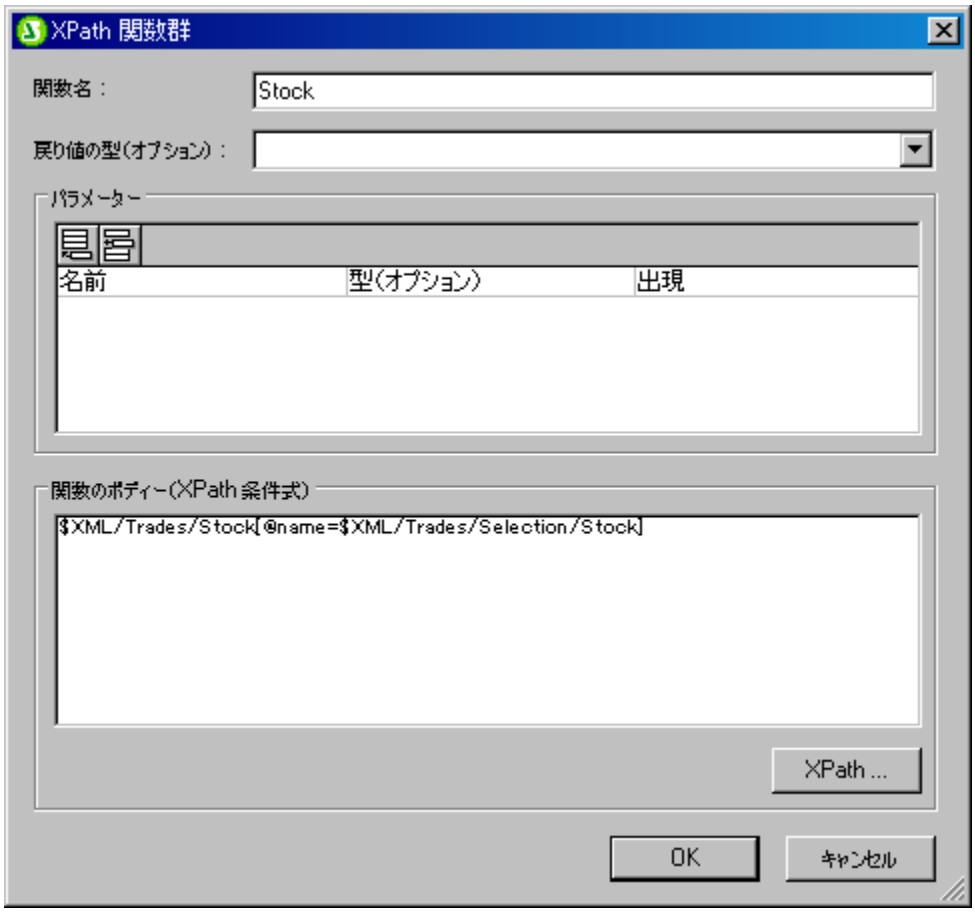
1. XPath 条件式をドキュメントのルートから開始します。ドキュメントルートは、$XML という XPath 条件式により与えられます。例えば、$XML/Trades/Stock[1] という XPath 条件式により、/Trades 要素以下にある最初の Stock 子要素が選択されます。メインスキーマのドキュメントルートを表す $XML という変数は、StyleVision にある全ての SPS デザインにてグローバルに定義されています。
2. 引数にコンテキストノードを与えることができます。詳細については、パラメーターとノード を参照してください。

以下のような状況で、エラーが返されます:

- 引数が定義されているが、関数のボディーで使用されていない場合。
- 関数により返される値のデータ型が、関数にて定義された戻り値の型にマッチしない場合。
- SPS 内にある関数のどれかにエラーが含まれる場合、（エラーを含む関数が呼び出されない場合でも）デザイン全体に対して XSLT エラーが生成されます。この場合、ユーザー定義 XPath 関数のリストにあるチェックボックスのチェックを外すことで、関数を無効化することができます。デザインにて無効化された関数は、デザインから生成された XSLT ドキュメントに含まれません。これにより、エラーを含む XPath 条件式を XSLT に含めず、XSLT エラーを回避することができます。

9.7.2 ノードを検索するための関数の再利用

直前のセクションでは、ノードを特定する XPath 関数を作成する方法について説明しました。以下のスクリーンショットにて示される sps:Stock 関数により、/Trades/Selection/Stock 要素のコンテンツにマッチする name 属性の値を持った /Trades/Stock 要素だけが返されることになります。



XPath 条件式の$XML/Trades/Stock[@ name=$XML/Trades/Selection/Stock] にて定義されている位置を、例えば以下の様にモジュール化することができます:

- $XML/Trades/Stock という定義の sps:Stocks() 関数
- $XML/Trades/Selection/Stock という定義の sps:SelectedStock() 関数

元からあった XPath 条件式は、以下のように書き直すことができます:

```
sps:Stocks()[@name=sps:SelectedStock()]
```

このような方法で XPath 関数を作成することにより、SPS デザイン上の他の場所で作成された XPath 条件式からも関数を再利用することが可能になり、複雑な XPath 条件式の記述を、大幅に簡易化することができます。

9.7.3 XPath 関数内のパラメーター

ユーザー定義 XPath 関数には任意の数のパラメーターを割り当てることができます。関数のパラメーターは XPath 関数ダイアログのパラメーターペインにて定義されます(以下のスクリーンショットを参照)。これらパラメーターは、ユーザー定義 XPath 関数の定義(関数のボディーペイン)にて使用することができます。

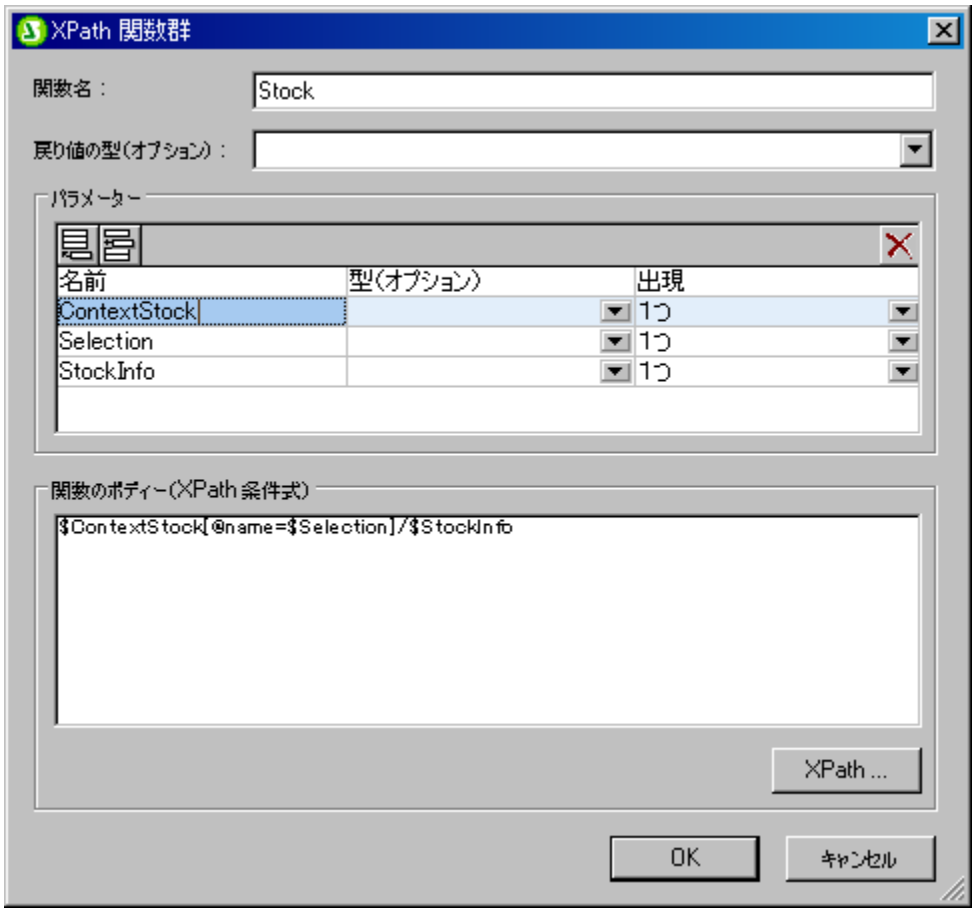
ユーザー定義 XPath 関数のメカニズム

XPath 関数は以下のように動作します:

1. 自動計算などで定義された関数呼び出しにて指定された引数の数は、ユーザー定義関数（以下のスクリーンショットにある、ユーザー定義関数のパラメーターペイン）にて定義されたパラメーターの数にマッチしなければなりません。更に、関数呼び出し内の各引数に対して与えられたアイテムの数も、対応するパラメーターにて定義された出現のオプションにマッチする必要があります。パラメーターに対してデータ型の制約が与えられている場合、引数により与えられた値もそのデータ型にマッチする必要があります。
2. 関数のパラメーターに対して渡された引数は、XPath 関数（以下のスクリーンショットに示される関数ボディー）にて使用されることになります。XPath 条件式を評価した結果は、オプションとして定義することのできる戻り値の型に対してチェックされます。データ型が期待されたものである場合、関数を呼び出した XPath 条件式にてその結果が使用されることになります。

パラメータの順序

関数が呼び出されると、パラメーターペインにて定義された順序に従って呼び出しに使用される引数がパラメーターへ渡されます（以下のスクリーンショットを参照ください）。



上のスクリーンショットにあるようなユーザー定義 XPath 関数の sps:Stock が定義され、以下の XPath 条件式により呼びだされた場合を考えてみましょう:

```
sps:Stock($XML, Node1, Node2)
```

これら3つの引数（$XML、Node1、Node2）は、関数のパラメーターにて定義された順序に従い、それぞれ $contextstock、$Selection、$StockInfo へ渡されます。

関数呼び出しにて記述された各引数はコンマにより分離されていることに注目してください。これにより、コンマにより区切られた関数呼び出し内の各引数が対応するパラメーターへ渡されることになります。

パラメーターペイン内にあるパラメーターの順序は、パラメーターペインにある**追加**、**挿入**、**削除**アイコンを使用することで変更することができます。

パラメータのデータ型

オプションとして、ユーザー定義関数におけるパラメーターのデータ型を定義することができます。データ型が指定された場合、入力された引数のデータ型がパラメーターのデータ型に対してチェックされ、型がマッチしない場合にはエラーが返されます。この機能により、(関数呼び出しの引数から得られる) 入力データをチェックすることができます。

出現

ユーザー定義 XPath 関数の各パラメーターは、シーケンスとして考えることができます。パラメーターの出現プロパティにより、関数呼び出しの引数からそのパラメーターに対してどれだけのアイテムを送信することができるのかを指定することができます。

関数の定義と関数呼び出しの両方において、パラメーターや引数の分離だけではなく、シーケンス内にあるアイテムの分離にもコンマは使用されます。そのため、使用されている文脈から、コンマがパラメーター/引数の分離に使用されているのか、またはシーケンスのアイテムを分離するのに使用されているのかを理解する必要があります。

- パラメーター/引数は、かっこにより関数定義内のパラメーターや関数呼び出し内の引数にあるシーケンスを分離することができます。
- シーケンス内にあるかっこは無視されます。

理解を深めるための例とその説明を以下に示します:

- **パラメーター/引数内部にあるかっこ**: avg() や count() といったいくつかの XPath 2.0 関数は、単一のシーケンスを引数として受け取ります。このシーケンスにて値がコンマにより区切られる場合、または範囲演算子が使用されている場合、シーケンスをかっこで囲むことにより、それらの値がコンマにより区切られた複数のシーケンスではなく、単一のシーケンスであることを明示的に示す必要があります。例えば、avg((count($a), $b, $c)) という関数呼び出しにて、XPath 2.0 関数の avg() は (count($a), $b, $c) という単一のシーケンスを引数として受け取ります。シーケンスのアイテムを列挙することで、3つのアイテムからなるシーケンスが構成されるため、引数は単一の引数として、かっこで囲まれた形式で avg() 関数へ渡す必要があります: avg((**count($a), $b, $c**))。内側に記述されたかっこが無い場合、3つの引数が avg() 関数に対して渡されることになり、(avg() 関数ではシーケンスからなる1つの引数しか期待されていないため) エラーが返されることになります。
- **パラメーター/引数内部にかっこを使用しない**: 上の例と同様に、count() 関数も単一のシーケンスを関数の引数として受け取ります。しかし、この例で示された count($a) という呼び出しでは、コンマによる区切りは行われておらず、変数/パラメーターの $a により値の取得が行われます。そのため引数内部にてかっこを使用する必要はなく、count($a) という表現は正しいものとなります。
- **関数呼び出し内にあるかっことコンマ**: 関数呼び出しでは、(XPath 関数ダイアログのパラメーターペインにて定義された) 対応するパラメーターに対して各引数が使用されるように、かっこを適切に使用する必要があります。例えば、MyAverage() というユーザー定義 XPath 関数の定義にて avg(count($a), $b, $c) という XPath 2.0 条件式が使用された場合、MyAverage((1,2,3),4,5) という関数呼び出しは妥当なものとなります。シーケンスの (1,2,3) がパラメーター $a に対して与えられ、シングルトンシーケンスの 4 と 5 が $b と $c にそれぞれ与えられます。オプションとして、シングルトンシーケンスをかっこで囲むこともできます。MyAverage() から返される値は、この場合 4 となります。

9.7.3.1 パラメータとシーケンス

ここでパラメーターとシーケンスの関係と、パラメーターとシーケンスが XPath 条件式でどのように使用されるかを理解しておくべきでしょう。以下のようにこれらの用語を定義します:

- **シーケンス**は、アトミック値またはノードからなるアイテムにより構成されます。シーケンスのアイテム間にコンマを使用することでシーケンスを構築することができます。

- XPath 2.0 関数は**パラメーター**を受け取るように定義することができます。例えば、関数の定義にて使用される count($a) という XPath 2.0 条件式におけるかっこ内の値 $a が関数のパラメーターとなります。
- **引数**は関数呼び出しにて使用されるアイテムのことです。例えば、count(//Person) という関数呼び出しでは //Person という引数が与えられています。この引数によりノードのシーケンスが count() の引数として与えられることになります。
- substring('styleVisionExamples', 6, 6) という関数の呼び出しでは3つの引数が与えられており、Vision という文字列が返されます。この呼び出しは substring() 関数の定義に対して妥当なものとなります。関数の呼び出しにて複数の引数が与えられた場合、それらはコンマにより区切られることになります。

シーケンスの区切りとして使用されるかっこ

XPath 2.0 条件式を構築する際には、**コンマにより区切られたアイテムや範囲演算子により構成されるシーケンスを区切るためにかっこが使用され、結果としてかっこにより区切られたシーケンスは1つのパラメーターまたは引数として読み取られる**ということに留意してください。

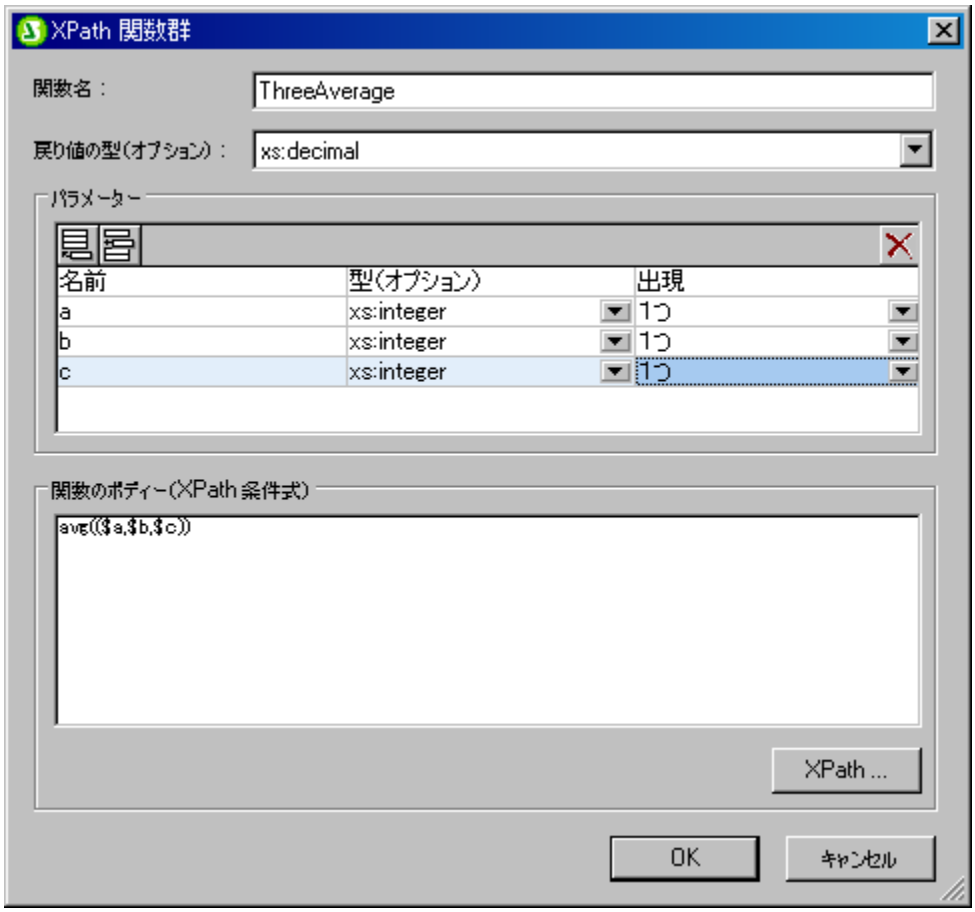
パス条件式は単一のパラメーターまたは単一の引数として読み取られるため、//Person/@salary というパス(ロケーター)条件式の周りでかっこを使用する必要は必ずしもありません。パス条件式は単一のパラメーター/引数からなるシーケンスとなります。

上の説明を補足するための例を以下に示します:

- avg((10, 20, 30)) : XPath 2.0 の avg 関数は、複数のアイテムが含まれているシーケンスを1つ引数として受け取ります。シーケンスに含まれるアイテムはコンマにより区切られているので、単一のシーケンスを区切るために内部のかっこが必要になります。このかっこが無い場合、3つの引数が与えられることになり、関数呼び出しは不正なものとなります(外型の括弧は関数に対する括弧となります)。
- avg(//Person/@salary) : パス条件式により、全 Person 要素のノードにある salary 属性が選択され、その属性値がシーケンスとして返されます。引数が読み取られる前にシーケンスが列挙されることは無いため、括弧をつける必要はありません。単一のパス(ロケーター)条件式が引数となります。パス条件式が評価され、その結果がシーケンスのアイテムとして関数へ与えられます。
- count((10 to 34)) : 範囲演算子による列挙が行われます。引数が読み取られる前に、範囲演算子の 'to' により、コンマにより区切られたアイテムのシーケンスが生成されます。そのため、count() 関数呼び出しの引数にはコンマにより区切られた25個のアイテムが含まれることになります。これらのアイテムを単一のシーケンスとして読み取るためには、括弧を使用する必要があります。括弧が使用されない場合、関数呼び出しに25個の引数が与えられることになり、(count() 関数の引数には単一のシーケンスを使用する必要があるため)関数呼び出しが不正なものとなってしまいます。
- count((10 to 34, 37)) : 内側の括弧により、括弧内部のアイテムにより構成される1つのシーケンスが関数呼び出しに使用されます。この例では26個のアイテムからなる単一のシーケンスが与えられます。
- count(//Person) : 単一の引数を与える場合、シーケンスの区切りとなる括弧を使用する必要はありません。XML ドキュメント内にある //Person ノードを選択するパス条件式が引数として使用され、選択されたノードがシーケンスのアイテムとして渡されます。

XPath 関数にて XPath パラメータを使用する

括弧がユーザー定義 XPath 関数にて使用された場合、(i) 関数呼び出しに使用された引数の数がユーザー定義 XPath 関数のそれとマッチするか、そして(ii) 期待されている型と出現に対して個々の引数が評価されるかを確認する必要があります。



上のスクリーンショットでは、3つのパラメーターが(パラメーターペインにて)定義されており、これらのパラメーターを関数ボディーペインで使用することにより、XPath 関数が定義されています。

パラメーターにて定義された各パラメーターは、それぞれ単一のシーケンスとして考えることができます。出現プロパティでは、それぞれのシーケンスにて許されることになるアイテムの数が指定されます。上のスクリーンショットに示される定義では、各パラメーターがシングルトンシーケンス(つまりアイテムが1つだけ含まれるシーケンス)として定義されています。そのため、関数呼び出しの各引数には、アイテムが1つ含まれるシーケンスが使用されることになります。型プロパティにより、シーケンスのアイテムに対するデータ型が指定されます。

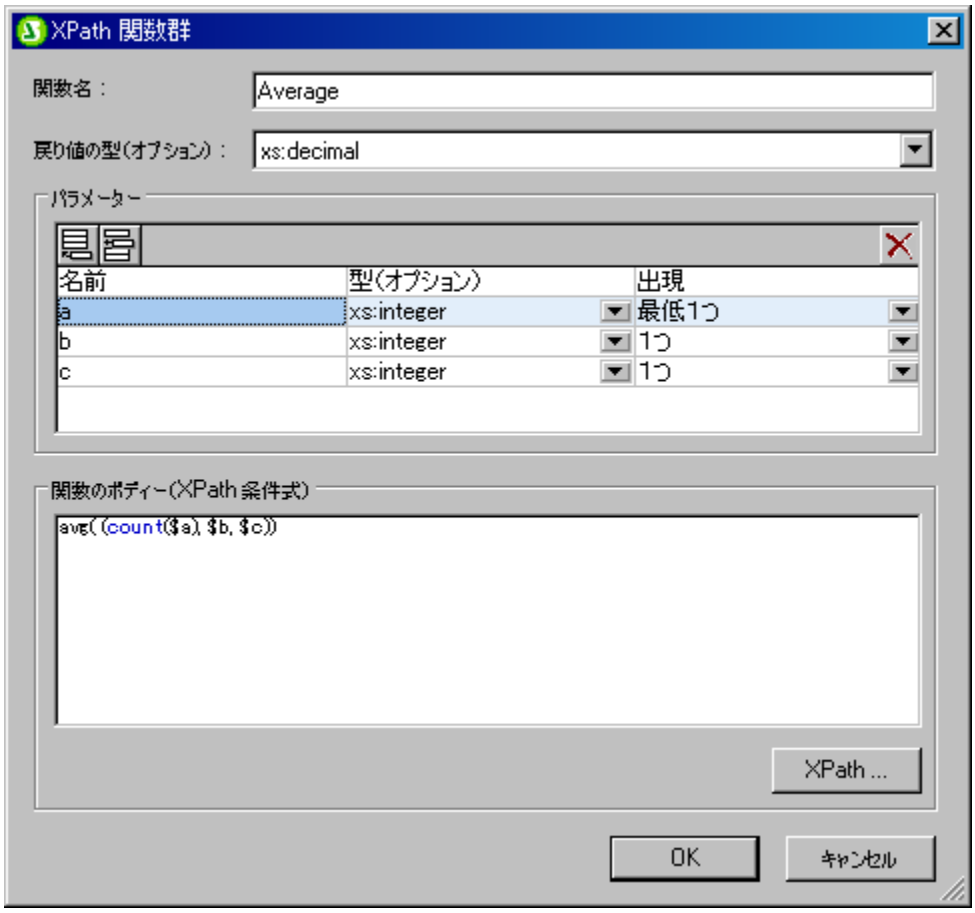
関数ボディーペインにあるXPath 関数の定義では、それぞれ1つのアイテムから構成されるシーケンスがパラメーターにより与えられ、それらの平均が求められます。これらのXPath パラメーターによりシーケンスが構成されるため、これらのパラメーターを括弧で囲み、単一のシーケンスとしてavg() 関数へ引数を渡す必要があります。ランタイムにて与えられる(3つのパラメーターに対応する)関数呼び出しの引数がシングルトンシーケンスでない場合、エラーが返されます。

上のスクリーンショットに示されるXPath 関数のThreeAverage() を呼び出すためのXPath パラメーターの例を以下に示します。デザインビューにて自動計算を挿入し、以下に示されるXPath 条件式を使用することで、結果を確認することができます。関数にはinteger 型の引数が3つ与えられ、それらの平均が返されます。

- sps:ThreeAverage(10,20,30) により 20 が返されます。関数呼び出しには XPath 関数のパラメーターにマッチする妥当な3つの引数が使用されています。
- sps:ThreeAverage((10),(20),(30)) により 20 が返されます。3つの XPath パラメータに対応する、3つの妥当な入力引数が存在します。各入力引数は、(各シーケンスが単一のシーケンスであるために、重複している、しかしながら重複はエラーではない) 括弧で囲まれています

- sps:ThreeAverage((10),20,30) により 20 が返されます。XPath 関数のパラメーターに対応する妥当な引数が3つ与えられています。第１引数が括弧で囲まれており、これは冗長な表現ですがエラーとはなりません。
- sps:ThreeAverage((10,20),(30),(40)) によりエラーが返されます。第１引数がシングルトンシーケンスでは無いため、エラーとなります（$a パラメーターの型プロパティには、「１つ」が指定されています）。
- sps:ThreeAverage((10,20,30)) によりエラーが返されます。引数として与えられているのは括弧により囲まれたシーケンスが１つだけで、XPath 関数のパラメーター数にマッチしません。更に、シーケンスもシングルトンシーケンスではないため、不正となります。

パラメーターの出現プロパティに最低1つがセットされた場合、１つまたは複数のアイテムを含むシーケンスを引数として使用することができます（以下のスクリーンショットを参照）。



上にある定義では、最初のパラメーターに対して1つ以上のアイテムが含まれるシーケンスが定義され、残り2つのパラメーターに対してシングルトンシーケンスが定義されています。最初のパラメーターに対して渡されたアイテムの数を数え、その数と、その他2つのパラメーターに対して渡された数の平均を返すように関数は定義されています。以下の点に注目してください:

- avg() 関数の呼び出しに使われる引数は括弧で囲まれています。これにより、３つのアイテムからなるシーケンスが、引数として avg() へ渡されます。count() 関数から返される値が最初のシーケンスアイテムに、パラメーター $b と $c がそれぞれ２番目と3番目のシーケンスアイテムになります。
- count() 関数呼び出しの引数は単一のシーケンスであることが明らかなので、括弧で囲まれていません。

上のスクリーンショットにて示される Average() XPath 関数を呼び出す際に使用する引数の例を以下に示します:

- sps:Average((1,2),3,4) により 3 が返されます。パラメーターに対応する3つの妥当な引数が与えられています。最初の引数は括弧により囲まれており、他の(シングルトン)シーケンスと区切られています。第1引数に対して count() 関数が処理され、2 という値が返され、その値が avg() 関数に渡されるシーケンスにおける最初のアイテムとして使用されます。
- sps:Average(4,4,4) により 3 が返されます。妥当な引数が3つ与えられています。最初の引数は1つのアイテムにより構成されるシーケンスでも良いため、他の引数と区別するために括弧を使用する必要はありません(対応するパラメーターの出現プロパティを参照ください)。

その他の留意するべき点

以下の点にも留意しておく必要があります:

- XPath 引数の出現に**最低1つ**がセットされている場合、avg(($a)) といった XPath 条件式を関数ボディーとする MyAverage() といった関数を定義することができます。この関数の引数には、1つ以上のアイテムが含まれるシングルトンシーケンスが渡されます。関数は、例えば sps:((2,3,4)) という呼び出しにより 3 という値が返されます。3つのシングルトンシーケンスではなく、単一のシーケンスとして引数を渡すために、引数は括弧で囲む必要があります。
- XPath パラメーターの $a の出現プロパティに**無しまたは1つ**が指定された場合、avg($a, $b, $c) といった XPath 条件式により構成される MyAverage() といった関数を定義することができます。この関数には3つのシーケンスが引数として渡されるだけではなく、最初の引数値を空にすることも可能になります。最初のシーケンスを空にする場合、空のシーケンスを最初の引数値として明示的に渡す必要があります(そうでない場合エラーが返されます)。sps:MyAverage(30, 20, 10) により関数の呼び出しを行うと、20 という値が返されます。sps:MyAverage((), 20, 10) という関数の呼び出しが行われた場合、(最初のシーケンスが空の入力値としてカウントされるため) 15 という値が返されます。それに対して sps:MyAverage(20, 10) という関数の呼び出しでは最初の(空の)シーケンスが明示的に示されておらず、第3引数が与えられていないと認識され、エラーが返されます。

複雑な例

XPath 条件式を再利用するという以外にも、ユーザー定義のXPath 関数を使用することで、XPath 2.0 の関数セットでは提供されていない、複雑でカスタマイズされたXPath 関数を構築することができます。例えば、シングルトンシーケンスを引数として受け取るXPath 条件式を使用することで、階乗関数を簡単に作成することができます。$num という引数が求める階乗の数だとすると、関数を構成するXPath 条件式は以下のようになります:

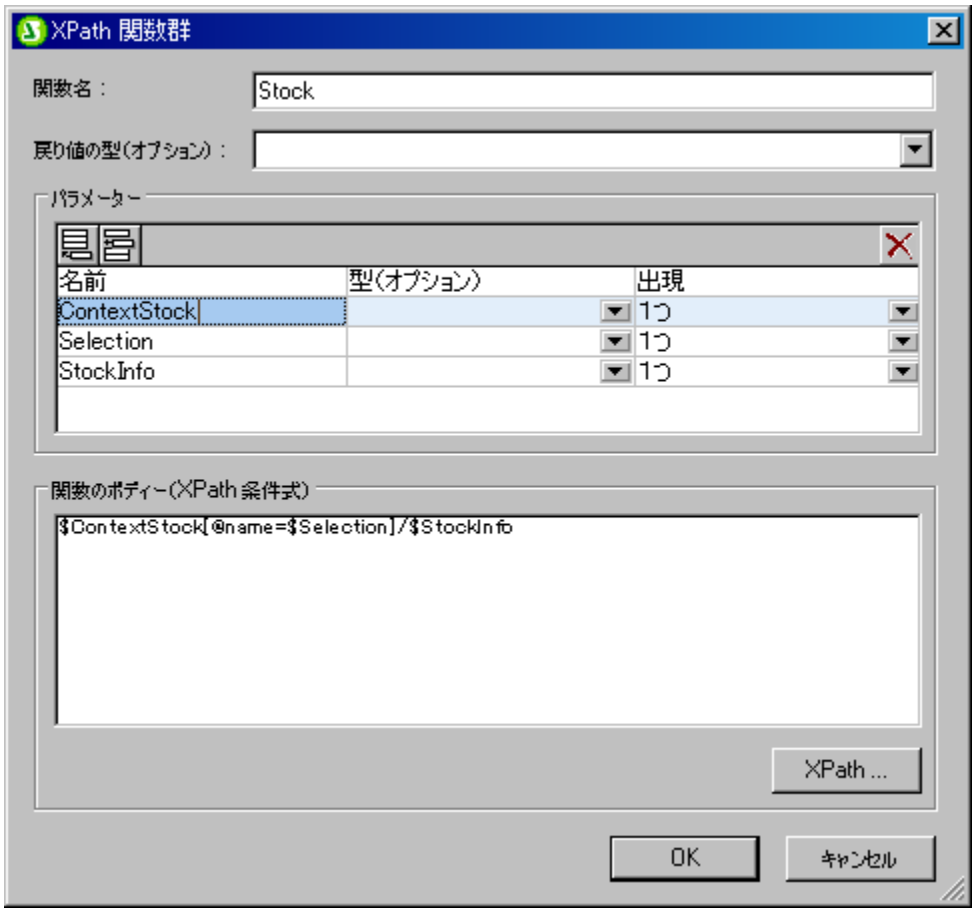
```
if ($num < 2) then 1 else $num * sps:Factorial($num - 1)
```

関数の名前を Factorial() とすると、例えば sps:Factorial(6) という関数呼び出しにより 6 の階乗を求めることができます。

9.7.3.2 パラメータとノード

XPath 関数にてノードを選択するパラメーターを使用する場合、関数がデザイン内のどこから呼ばれるかに関わらず、その関数にはコンテキストノードが与えられていないということに注意する必要があります。関数の定義内(関数ボディーペイン)で使用される XPath 条件式か、XPath 関数を呼び出す際に使用する XPath 条件式にてコンテキストノードを取得することができます。後者の場合、関数呼び出しの引数としてコンテキストノードが与えられることになります。

以下のスクリーンショットに示される、3つのパラメーターとともに定義された Stock() というユーザー定義 XPath 関数を考えてみましょう



関数ボディーにおける定義は $ContextStock[@name=$Selection]/$StockInfo となっており、3つのXPath パラメーターが使用されているものの、コンテキストノードの情報は含まれていません。コンテキストノードの情報は、以下の例にあるように、この関数を呼び出す XPath 条件式から与えられることになります:

```
sps:Stock( $XML/Trades/Stock, $XML/Trades/Selection/Stock, @name )
```

関数呼び出しでは3つの引数が使用されており、コンテキストノードの情報として使用される値とノードの識別を行うための情報が与えられます。以下のXPath 条件式により関数を呼び出し、同様の結果を得ることもできます:

```
sps:Stock( /Trades/Stock, /Trades/Selection/Stock, @name )
sps:Stock( /Trades/Stock, //Selection/Stock, @name )
```

デザインコンポーネントのXPath 条件式では、コンテキストノードが分かっているため、デザインコンポーネントにて使用される関数の呼び出しでは、ドキュメントルートを表す $XML 変数を使用する必要は必ずしもありません。

上に示される関数呼び出しには、定義された3つのXPath パラメーターに対して、それぞれ3つの引数値が与えられていることに注目してください。

- $ContextStock = $XML/Trades/Stock (/Trades/Stock 要素)
- $Selection = $XML/Trades/Selection/Stock (/Trades/Selection/Stock 要素)
- $StockInfo = @name

関数定義におけるXPath 条件式は以下のようになります:

```
$ContextStock[@name=$Selection]/$StockInfo
```

引数が渡されることで、関数定義における XPath 条件式は以下のようになります:

```
$XML/Trades/Stock[@name=$XML/Trades/Selection/Stock]/@name
```

関数に渡されるのはテキスト文字列ではなく、ノードセットであるという点に留意してください。

これにより、XPath パラメーターを介したコンテキストノードの受け渡しが行われ、その後関数にて目的のノードが特定、返されることになります。

(C) 2015-2021 Altova GmbH

9.8 日付の使用

ソースドキュメントに日付の値を受け取るノードが含まれる場合、使用されているXML スキーマにある xs:date や xs:dateTime データ型を使用することで、XPath 2.0/3.0 (*下の例を参照*を参照)。StyleVision では、xs:date または xs:dateTime データ型に対して以下のような機能がサポートされます。

1. グラフィカルな 日付フォーマット を使用することで、Authentic View ユーザーはノードに対して指定されたデータ型のフォーマットに適合する日付の入力を行うことができます。データ入力に使用される日付選択はAuthentic View でのみ利用することができます。
2. 入力フォーマット 機能による、広範囲の日付フォーマット 機能。

These StyleVision の機能については、このセクションのサブセクション(日付選択の使用 と日付のフォーマット)にて記述されます。このセクションでは、日付が関わるような計算を行うXPath 2.0 の条件式について記述されます。

メモ XPath 1.0 では日付ならびに時間データの操作を行うことができません。しかし、XPath 1.0 でも 日付の選択 を使ったデータのインテグリティーと日付フォーマットを使った 日付のフォーマットを保証することができます。

XPath 2.0 における日付の計算

自動計算 におけるXPath 2.0 条件式では、日付形式のデータを操作することができます。XPath 2.0 条件式により行うことのできる例を以下に示します:

- XPath 2.0 関数の current-date() ならびに current-dateTime() を使用することで、現在の日付と日時をそれぞれ取得することができます。
- 日付の計算を行うことができます。例えば、current-date() - DueTime という計算に対して xdt:dayTimeDuration 形式の値を得ることができ、例えば P24D という値により、プラス24日間という期間が表されます。
- XPath 2.0 関数を使用することで、期間から時間単位を取得することができます。例えば、day-from-duration(xdt:dayTimeDuration('P24D')) という条件式により、24 という数値が返されます。

自動計算におけるXPath 2.0 条件式の例を以下に示します。この条件式ではある項目の支払期限が過ぎた場合に、4% の年利に対する日割りの利子額を(単利計算で)計算し、その値と元本の和が返されます。

```
if          (current-date() gt DueDate)
then   (round-half-to-even(InvoiceAmount +
                    (InvoiceAmount*0.04 div 360 *
                          days-from-duration((current-date() - DueDate))), 2))
else   InvoiceAmount
```

このような計算を行うには、DueDate 要素が xs:date といった日付型により定義されており、その要素のコンテンツがYYYY-MM-DD[±HH:MM] という形式(時間部分はオプション)で記述されている必要があります。日付選択を使用することにより、日付が正しい形式で入力されることを保証することができます。

関連項目

- 日付のフォーマット
- 日付選択の使用
- 日付選択の挿入
- 日付選択の自動追加

9.8.1 日付選択の使用

日付選択とは、Authentic View に表示されるカレンダーのことで、選択された日付が xs:dateTime データ型に従った正しい書式で入力されることを保証します。

書式はデータ型に従うかたちで、適切に入力されます。

- xs:date の場合、YYYY-MM-DD[±HH:MM] という書式での入力が行われます（± から始まる時刻部分はオプション）。タイムゾーンコンポーネントの値は日付選択で選択することができます。
- xs:dateTime の場合、エントリーの書式は YYYY-MM-DDTHH:MM:SS[±HH:MM] となります。タイムゾーンのコンポーネント（プレフィックス ±）は、XML スキーマ仕様に従い任意です。ユーザーはタイムゾーンコンポーネントの値を選択することができます。

デザインでの日付選択の挿入と削除

日付選択は、SPS デザイン内にある、(i) xs:date または xs:dateTime データ型で定義された任意のノード、さらに (ii) コンテンツや入力フィールドとしてノードが作成された際に挿入することができます。以下にあるどれかの方法により日付選択を挿入することができます:

- xs:date または xs:dateTime 型のノードが SPS にて作成された時のデフォルト動作として: この動作をデフォルトとするには、メニューオプションから「**Authentic | 日付選択の自動追加**」コマンドを有効にしてください。日付選択の自動追加が有効になっている場合、xs:date または xs:dateTime データ型の要素がコンテンツまたは入力フィールドとして作成された時、またはこれらのコンポーネントへ変更された時に日付選択が挿入されます。
- SPS 内にあるカーソルが、xs:date または xs:dateTime ノード内部に配置されている時: メニューオプションの「**挿入 | 日付選択**」を選択する。カーソルが xs:date または xs:dateTime ノード内部に配置されている場合、「**挿入**」メニューまたはコンテキストメニューからこのコマンドへアクセスすることができます。.

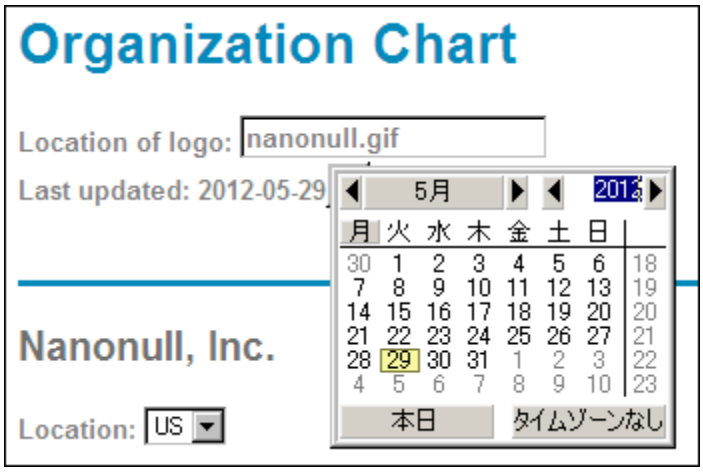
日付選択が挿入されると、日付選択アイコン がその場所に表示されます。日付選択を削除するには、**Delete** または **Backspace** キーを押下してください。

Authentic View にて日付選択を使用する

Authentic View では、日付選択がアイコンとして表示されます（以下のスクリーンショットを参照）。

アイコンをクリックして日付を修正することができます。この操作により日付選択が表示されます（以下のスクリーンショットを参照）。日付選択にある目的の日付を選択することで、新たな日付が入力されます。ノードのデータ型に従うかたちで、正しい書式によりデータが入力されます。

(C) 2015-2021 Altova GmbH



タイムゾーンを入力するには、(デフォルトで「タイムゾーンなし」にセットされている)タイムゾーンボタンをクリックします。ノードのデータ型の書式に従うかたちでタイムゾーンが入力されます(以下のスクリーンショットを参照)。

関連項目

- 日付選択の挿入
- 日付選択の自動追加
- 日付との作業
- 日付のフォーマット

9.8.2 日付のフォーマット

XML ドキュメント内に収められる日付は、そのノードのデータ型にて指定された書式に従って保存されます。例えば、xs:date ノードの値には YYYY-MM-DD[±HH:MM] という書式が使用され、xs:dateTime 型のノードでは YYYY-MM-DDTHH:MM:SS[±HH:MM] というフォーマットが使用されます。デフォルトでは XML 内部にあるデータが Authentic View と出力の両方にてそのまま表示されます。しかし、SPS にて値フォーマット機能を使用することで、出力されたデータとは違った表示を Authentic View にて行うことができるようになり、場合によっては出力の表示も変更することができます。

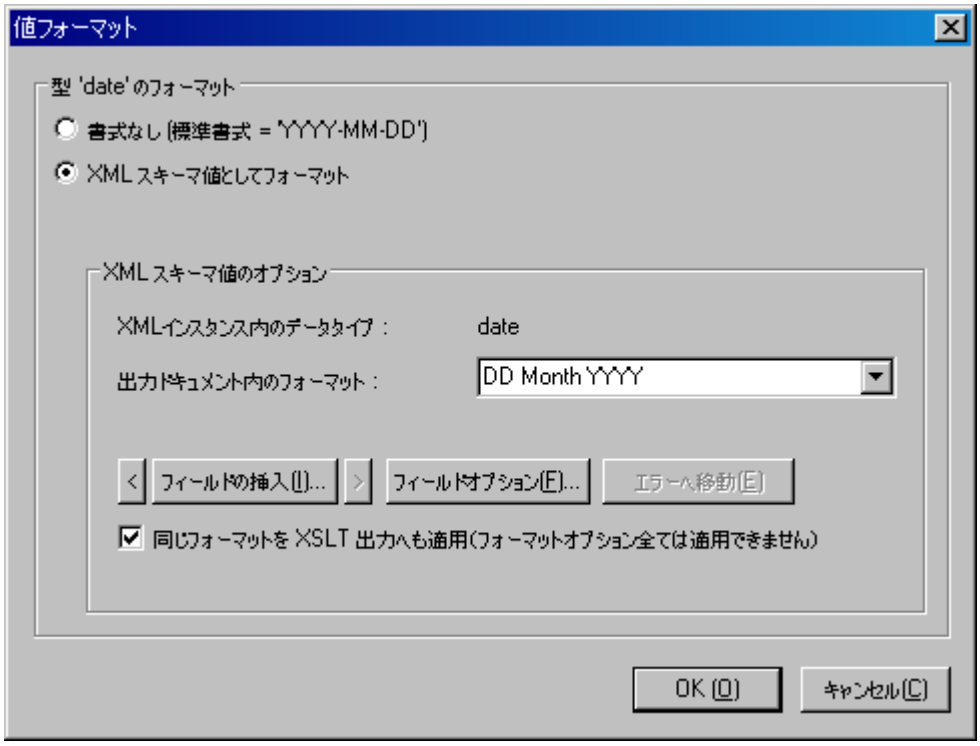
値フォーマットを日付に対して使用することで、以下のデータ型を使ったノードや自動計算に対してカスタムフォーマットを定義することができます:

- xs:date
- xs:dateTime
- xs:duration
- xs:gYear
- xs:gYearMonth
- xs:gMonth
- xs:gMonthDay
- xs:gDay

値フォーマットを使用して日付ノードのフォーマットを行う

以下の操作により、日付ノードにある実際のデータとは異なる日付表示を行うことが可能になります:

1. ノードのコンテンツプレースホルダまたは入力フィールドを選択します。**コンテンツ**、または**入力フィールド**として作成されたノードに対してしか値フォーマットは使用できない点に注意してください。
2. プロパティサイドバーにて**コンテンツ**アイテムを選択し、コンテンツプロパティグループを選択します。値フォーマットプロパティの**編集**ボタン [...] をクリックします。値フォーマットダイアログが表示されます(以下のスクリーンショットを参照)。



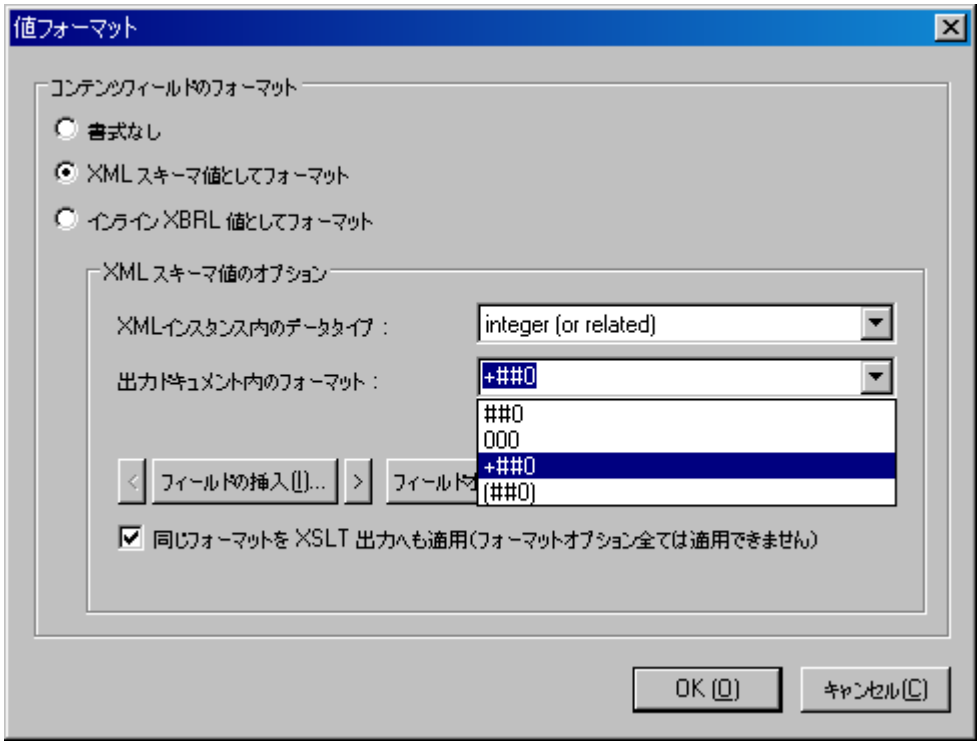
デフォルトでは**書式無し**ラジオボタンが選択されています。
その他のフォーマットを定義するには、**フォーマット**ラジオボタンを選択します。

4. コンボボックスのドロップダウンリストから定義済みの日付フォーマットを選択することができるほか、コンボボックスの入力フィールドにて独自のフォーマットを定義することができます(以下のスクリーンショットを参照)。独自のフォーマットを定義するための構文に関する詳細については 値フォーマット構文 のセクションを参照ください。

値フォーマットによる自動計算のフォーマット

日付フォーマットにより表示された値を、自動計算にて評価することができ、その結果のフォーマットを行うために値フォーマットを使用することができます。以下の操作によりこのような処理を行うことができます:

1. デザイン内にある自動計算を選択します。
2. プロパティサイドバーにてコンテンツアイテムを選択し、自動計算プロパティグループを選択します。値フォーマットプロパティの**編集**ボタン ... をクリックしてください。値フォーマットプロパティのダイアログが表示されます(以下のスクリーンショットを参照)。



   デフォルトでは**書式無し**ラジオボタンが選択されています。
3. その他のフォーマットを定義するには、**フォーマット**ラジオボタンを選択します。
4. コンボボックスのドロップダウンリストから定義済みの日付フォーマットを選択することができるほか、コンボボックスの入力フィールドにて独自のフォーマットを定義することができます(以下のスクリーンショットを参照)。独自のフォーマットを定義するための構文に関する詳細については 値フォーマット構文 のセクションを参照ください。

値フォーマットを出力へ適用する

定義された値フォーマットは Authentic View. 更に、に対して適用されます。その他にも、いくつかの値フォーマット定義を(全てでは無いですが) HTML とRTF 出力に対して適用することができます。**XSLT 出力へも適用**チェックボックスにチェックを入れてください。このオプションがチェックされていないか利用できない場合、Authentic View の出力に対してのみ値フォーマットが適用されます。

関連項目

- 値フォーマット(数値データ型のフォーマット)
- 日付選択の使用

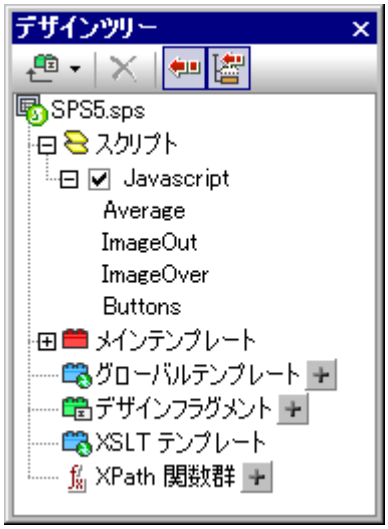
9.9 スクリプトの使用

StyleVision では、各 SPS に対して JavaScript 関数を(デザインビューのタブとして用意されている) JavaScript エディターにて定義することができます。ここで作成された関数は、HTML ドキュメントのヘッダー部分に保管され、HTML ドキュメントのボディー部から呼び出すことができます。以下のような状況で、関数を使用することができます:

- 複数のスクリプトを使用して、複雑な結果を求める。この場合、必要とされているそれぞれのスクリプトを個別の関数として作成し、デザインドキュメントからこれら関数を参照することができます。
- 特定のスクリプトをドキュメント内にある複数の箇所から呼び出す。

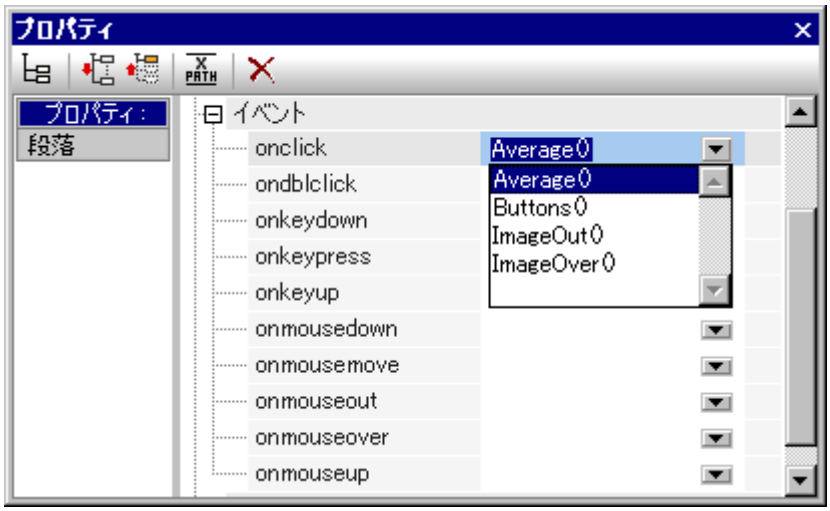
JavaScript エディターにて関数を定義する方法については、JavaScript 関数の定義サブセクションを参照ください。

GUI では、SPS にて定義された全ての JavaScript 関数がデザインツリーウィンドウのスクリプトエントリー以下に表示されます(以下のスクリーンショットを参照)。以下にあるスクリーンショットでは Averate、ImageOut、ImageOver、そして Buttons という4つの JavaScript 関数が表示されており、SPS にて JavaScript が有効になっています。



JavaScript エディターにて定義された関数は、GUI ではイベントハンドラー呼び出しとして扱われます。デザインドキュメント内にあるコンポーネントを選択することで、プロパティサイドバーのイベントプロパティグループにあるイベントハンドラープロパティから、定義されている全ての関数に対する割り当てを行うことができます(*下のスクリーンショット*)。イベントハンドラーに対して JavaScript 関数を割り当てる方法については、関数のイベントハンドラーへの割り当てセクションにて記述されます。

(C) 2015-2021 Altova GmbH



メモ スクリプトは HTML 出力に対してだけ適用され、Authentic View には適用されません。

モジュール SPS 内のスクリプト

SPS モジュールが別の SPS モジュールに追加されると、追加されたモジュール内にあるスクリプトが参照元 SPS でも利用できるようになり、参照元 SPS のコンポーネントに対してプロパティサイドバーからイベントハンドラーとして使用することができるようになります。モジュール SPS の使用に関する詳細については、モジュール SPS のセクションを参照ください。

関連項目

- デザインビュー
- デザインツリー
- モジュール SPS

9.9.1 JavaScript 関数の定義

以下の操作を行うことで、JavaScript 関数を定義することができます:

1. デザインビューにてデザインビューのタブをクリックして、ポップアップから JavaScript を選択し、JavaScript エディターを表示します（以下のスクリーンショットを参照）。

2. JavaScript エディターにて、関数の定義を記述します（以下のスクリーンショットを参照）。

```
1   function DisplayTime()
2   {
3   now = new Date();
4   hours = now.getHours();
5   mins = now.getMinutes();
6   secs = now.getSeconds();
7   result = hours + ":" + mins + ":" + secs;
8   alert(result);
9   }
10
11  function ClearStatus()
12  {
13  window.status="";
14  ]
```

上のスクリーンショットでは、DisplayTime と ClearStatus という2つの JavaScript 関数が定義されています。これらの関数はアクティブな SPS に対して記述されています。HTML ファイルでは、これらの関数が以下にあるようにヘッダー内部に記述されます。

```
<script language="javascript">

<!-- function DisplayTime()
{
        now = new Date();
        hours = now.getHours();
        mins = now.getMinutes();
        secs = now.getSeconds();
        result = hours + "." + mins + "." + secs;
        alert(result)
}

function ClearStatus()
{
        window.status="";
}
-->

</script>
```

これらの関数は HTML ドキュメント内部の任意の場所から呼び出すことができます。StyleVision では、定義された関数をプロパティサイドバーのイベントプロパティグループにあるイベントハンドラープロパティへ割り当てることができます。詳細については 関数のイベントハンドラーへの割り当て のセクションを参照ください。

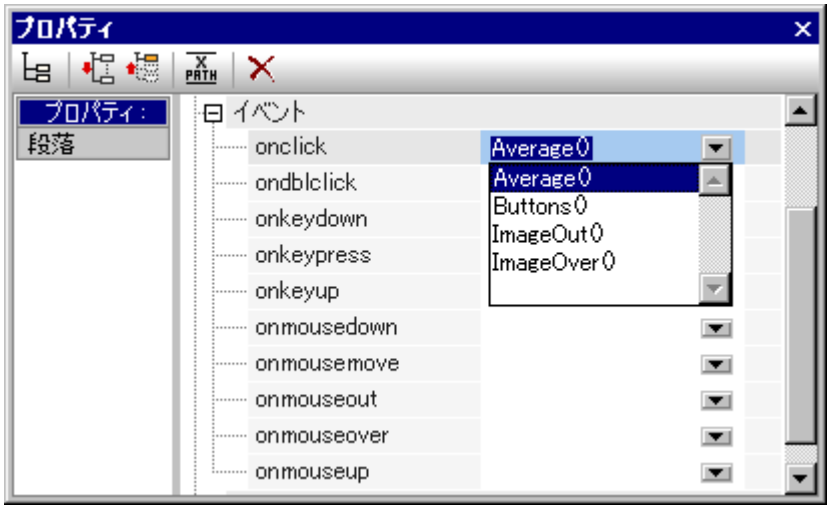
関連項目

- デザインビュー
- イベントハンドラーへ割り当て

9.9.2 関数をイベントハンドラーとして割り当てる

StyleVision GUI では、SPS コンポーネントがHTML にてレンダリングされた際に発生するイベントハンドラーとして JavaScript 関数を割り当てることができます。これらのイベントハンドラーはHTML 出力で使用されます。グローバル関数をイベントハンドラーとして割り当てるこ

とで、(例えばonClick のような)イベントに対してイベントハンドラーをセットすることができます。プロパティサイドバーでは、JavaScript エディターにて定義されたグローバル関数を、イベントプロパティグループにある各イベントのドロップダウンボックスからイベントハンドラーとして利用することができます(以下のスクリーンショットを参照)。



以下の操作により、関数をイベントハンドラーに割り当てることができます:

1. イベントハンドラーを定義する SPS のコンポーネントを選択します。あらゆる種類のノードやコンテンツを(静的か動的かに関わらず)選択することができます。
2. プロパティサイドバーにてイベントプロパティグループを選択します。利用することのできるイベントが、属性カラムに表示されます(上のスクリーンショットを参照)。
3. 目的のイベントに対して、値カラムにあるコンボボックスの矢印をクリックします。JavaScript エディターにて定義された全ての関数が表示されます。
4. ドロップダウンボックスから、そのイベントのイベントハンドラーとして目的の関数を選択します。

イベントハンドラーが定義されたコンポーネントのイベントがHTML 出力にて起動されると、対応するJavaScript 関数が実行されます。

関連項目

- デザインビュー
- JavaScript 関数の定義

9.9.3 外部 JavaScript ファイル

SPS は、以下にある2つの方法により外部 JavaScript ファイルへアクセスすることができます:

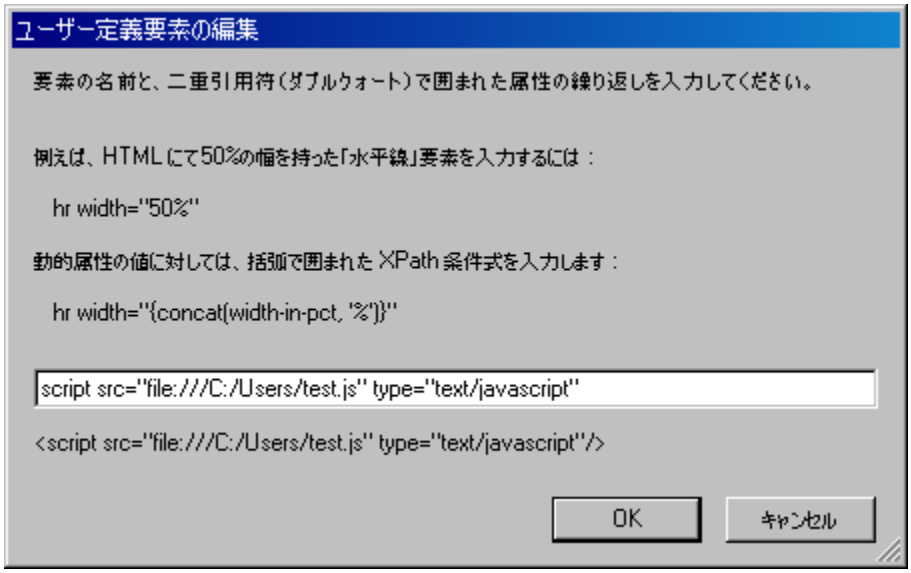
1. ユーザー定義要素またはユーザー定義 XML ブロックを作成する: これらのデザインオブジェクトには外部 JavaScript ファイルへのアクセスを行うSCRIPT 要素を含めることができます。ユーザー定義要素またはユーザー定義 XML ブロックが、デザインの BODY 要素内部に配置されているということを確認してください。これにより HTML 出力の(HEAD 要素ではなく)BODY 要素内部にこれらのオブジェクトが配置されます。
2. JavaScript エディター内にスクリプトを追加する: このようにして作成されたスクリプトは、HTML 出力のHEAD 要素内に配置されます。

ユーザー定義要素とユーザー定義 XML ブロック

ユーザー定義要素 とユーザー定義 XML ブロックを使用することで、外部 JavaScript ファイルへアクセスすることができます。これらのメカニズムを使用することで、外部 JavaScript ファイルへのアクセスを行う SCRIPT 要素を、出力 HTML ドキュメントの BODY 要素内にある任意の場所へ挿入することができます。

以下の操作により、ユーザー定義要素 を挿入することができます:

1. JavaScript ファイルへのアクセスを行う SCRIPT 要素を挿入する場所へカーソルを配置します。
2. 「**挿入**」メニューオプションまたはコンテキストメニューから、ユーザー定義要素を挿入するコマンドを選択します。



3. ダイアログが表示されるため、上のスクリーンショットにあるように、JavaScript ファイルの URL を src 属性の値とする SCRIPT 要素を入力します(例: script type="text/javascript" src="file:///c:/Users/mam/Desktop/test.js")。
4. 「**OK**」をクリックして完了します。

ユーザー定義 XML ブロック を使用することでも同様の結果を得ることができます。ユーザー定義要素に対して記述された上記の方法と違うのは、(i) ユーザー定義 XML ブロック の代わりにユーザー定義要素 が挿入され、(ii) 開始タグと終了タグを含む SCRIPT 要素全体が挿入されるという点です。

JavaScript エディター

JavaScript エディター を使用することで、HTML 出力の HEAD 要素に外部スクリプトを挿入することができます。JavaScript エディターにて作成された関数定義の外側に、以下にあるスクリプトフラグメントを入力してください。

```
var script = ドキュメント.createElement('script');
script.type = 'text/javascript';
script.src = 'file:///c:/Users/Desktop/test.js';
var head = ドキュメント.getElementsByTagName('head')[0];
head.appendChild(script)
```

script.src にある URL にて指定された外部 JavaScript ファイルが、出力 HTML ドキュメントの HEAD 要素からアクセスされます。

関連項目

- JavaScript 関数の定義
- ユーザー定義要素
- ユーザー定義 XML テキストブロック
- イベントハンドラーへ割り当て

9.10 HTML のインポート

StyleVision では、HTML ファイルをベースとしてインポートすることで、以下のファイルを作成することができます:

- インポートされた HTML ファイルのデザインと構造をベースにした SPS ドキュメント。
- HTML ドキュメントコンポーネントがスキーマ要素または属性として作成された XML スキーマ。オプションとして、HTML ドキュメントとは関係の無い要素や属性をユーザー定義スキーマに作成することもできます。
- (i) 作成した XML スキーマの構造をベースにして、(ii) HTML ファイルから得られたコンテンツを含む XML ドキュメント。
- デザインビューのデザインをベースにした XSLT スタイルシート。

HTML-から-XML を作成するステップ

以下に示される手順により、インポートされた HTML ファイルをベースにした XML ファイルの作成を行うことができます:

1. HTML インポートによる新たな SPS の作成 : HTML ファイルが StyleVision へインポートされると、新規 SPS ドキュメントが作成されます。HTML ドキュメントが HTML マークアップタグとともにデザインビューにて表示されます。**ユーザールート**という名前のドキュメント要素とともに、ユーザー定義 XML スキーマがスキーマツリーウィンドウに表示されます。このスキーマが、SPS がベースとするスキーマとなります。この時点でデザインビューに表示されている HTML ドキュメントのコンテンツとマークアップは、静的コンテンツとして SPS に含まれています。
2. スキーマと SPS デザインの作成 : (i) HTML ドキュメントから(スキーマツリーウィンドウの) スキーマツリー上の適切な場所にコンポーネントをドラッグするか、オプションとして (ii) 独自のノードをスキーマツリーへ追加することで、スキーマを作成します。デザインウィンドウでは、スキーマツリーにてノードを構築するために使用された HTML コンテンツが、スキーマノードタグに囲まれて表示されます。対応するスキーマノードが存在しない HTML コンテンツは、スキーマノード無しで表示されます。
3. デザインドキュメントにて、必要に応じてノードのフォーマットや処理ルールの調整、静的コンテンツの追加などを行います。これらの修正は SPS ならびに生成された XSLT においてのみ反映されます。生成されたスキーマや XML ファイルに反映されることはありません。
4. スキーマツリーと SPS デザインの構築が完了した後には、以下の項目の生成と保存 を行うことができます:

- 作成したスキーマツリーに対応する XML スキーマ
- SPS デザイン内にあるコンテンツプレースホルダとともに作成されたスキーマのコンテンツと、スキーマをベースにした構造を伴う XML データファイル
- SPS (.sps ファイル) またはデザインをベースにした XSLT スタイルシート

関連項目

- ユーザー定義スキーマ
- XSLT からの新規作成

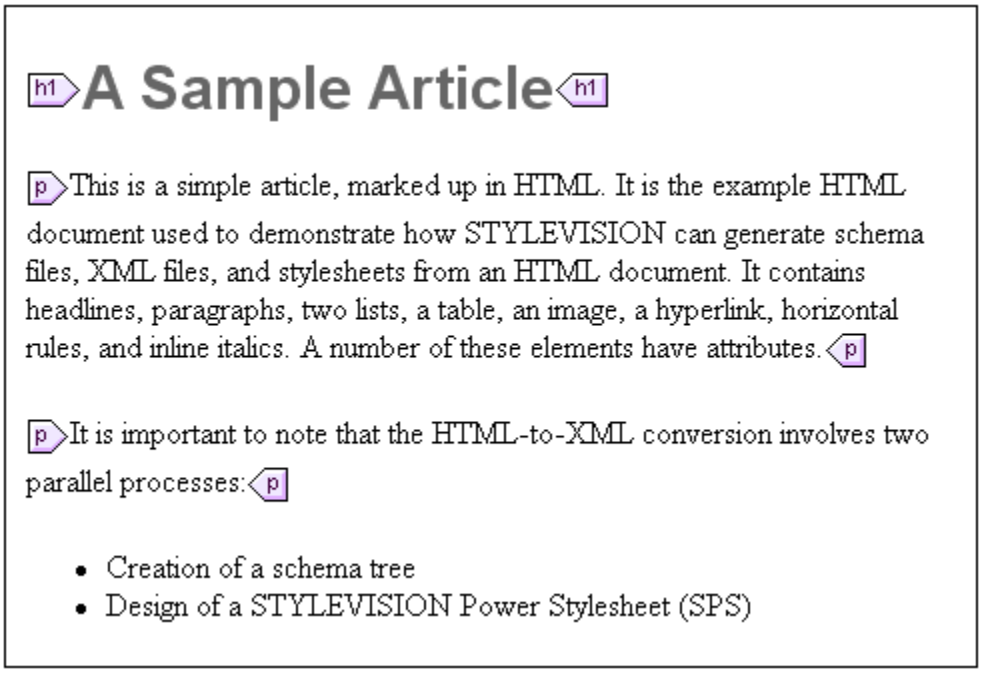
9.10.1 HTML のインポートにより新規 SPS を作成

以下の操作により、HTML ドキュメントから新たな SPS ファイルを作成することができます:

1. メニューオプションから「**ファイル | 新規作成 | HTML ファイルから新規作成**」を選択します。
2. 表示されるファイルを開くダイアログにて、インポートする HTML ファイルを選択し、**開く**ボタンをクリックします。
3. 相対パスを絶対パスへ変換するか尋ねられます。適切なオプションを選択してください。

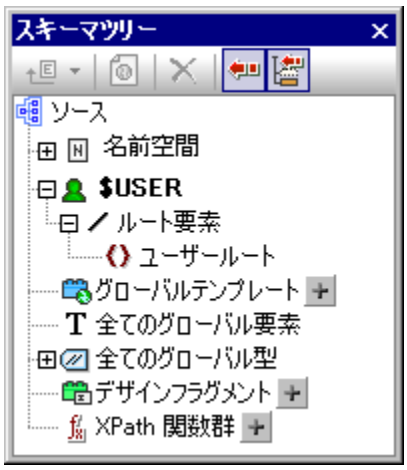
(C) 2015-2021 Altova GmbH

新規 SPS ドキュメントが作成されます。デザインビューには、StyleVision にて使用することのできる定義済み HTML フォーマットのマークアップとともにドキュメントが表示されます（以下のスクリーンショットを参照）。



HTML ドキュメントがメインテンプレート内に表示されていることにご注目してください。グローバルテンプレートは存在しません。

スキーマツリーサイドバーにて、**ユーザールート**という名前のドキュメント要素とともに、ユーザー定義スキーマが作成されます（以下のスクリーンショットを参照）。



全てのグローバル要素のリストにグローバル要素が表示されていないことにご注目してください。

SPS 構造とデザイン

SPS には単一のテンプレート（メインテンプレート）が含まれており、このテンプレートは一時的な内部 XML ドキュメントのドキュメントノードへ適用されます。スキーマツリーウィンドウにて作成されたユーザー定義スキーマの構造に従うかたちで、この XML ドキュメントは作成されます。**この時点で**デザインビューには、メインテンプレート内部の HTML ドキュメントコンポーネントが静的コンポーネントとして SPS 内に表示されます。これら HTML コンポーネントは、Authentic View にて読み取り専用の非 XML コンテンツコンポーネントとして処理されます。

XSLT スタイルシートには、これらのHTML コンポーネントがリテラル結果要素として含まれます。この時点のスキーマには、ドキュメント要素のユーザールートしか含まれておらず、一時的な内部 XML ドキュメントにはドキュメント要素のユーザールートだけが含まれます。

以下にあるいずれかの方法により、ユーザー定義スキーマにて選択した HTML を、要素や属性として作成することができます:

1. 選択対象を要素または属性へ**変換します**。デザインにはノードタグがコンテンツプレースホルダとともに挿入されます。スキーマには要素または属性が作成されます。スキーマでは要素または属性が作成され、XML ドキュメントでは選択対象が XML ドキュメントにて作成されたスキーマノードのテキストコンテンツに変換されます。XML ドキュメントにて作成されたノードのコンテンツは、SPS を介して得られた出力へ動的に挿入されます。
2. 選択対象を要素または属性で**囲みます**。デザインでは選択対象がノードタグにより囲まれ、コンテンツプレースホルダは挿入されません。つまり、選択対象は静的コンテンツとして SPS デザインに表示されます。スキーマでは要素または属性が作成され、XML ドキュメントでは、空のノードが作成されます。スキーマノードタグ内にある静的テキストは出力されるものの、コンテンツプレースホルダがデザイン内部で明示的に挿入されない限り、このノードに対して動的コンテンツが出力されることはありません。

コンテンツプレースホルダにより、XML ドキュメントから得られたデータが(出力にて)表示される場所や(Authentic View にて)編集可能な場所が示されます。

関連項目

- ユーザー定義スキーマ

9.10.2 スキーマと SPS デザインの作成

デザインビューからユーザー定義スキーマへ選択対象をドラッグすることで、スキーマが作成されます。複数のスキーマを一度に作成することはできません。スキーマツリー内のノードへ選択対象をドロップすることで、子または兄弟として新たなノードが作成されます。作成されるノードの種類を要素と属性から選択し、選択対象を新たなノードへ変換するか、囲むことができます。

選択対象

デザインビュー内にある選択対象は以下のどれかとなります:

- HTML ドキュメント内にあるノード
- ノード内にあるテキスト
- ノード間にあるテキスト文字列
- イメージ
- リンク
- テーブル
- リスト
- 上にあるコンポーネントの組み合わせ

このセクションでは、選択対象に対して行われる一般的な処理について記述します。テーブルならびにリストの特別な場合については要素/属性としてテーブルやリストを作成のセクションを参照ください。

選択を行うには、HTML ドキュメントコンポーネントをクリックするかテキスト文字列をハイライトします。複数のコンポーネントを選択するには、ドラッグにより複数のコンポーネントをハイライトします。StyleVision にて複数のコンポーネントがハイライトされた場合、それらコンポーネントの親ノード以下にある全ての子ノードが含まれることに注意してください。

スキーマツリーにおける位置

目的のスキーマツリーノードへ選択対象をドラッグすると、マウスカーソルが以下にあるどちらかのシンボルに変わります。

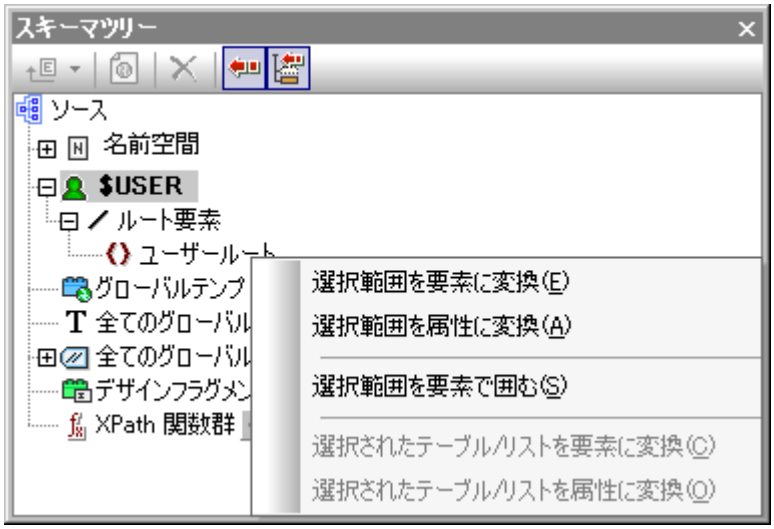
(C) 2015-2021 Altova GmbH

- 新たな兄弟を作成シンボル にてドロップすると、ドロップされたノードの兄弟ノードとして選択対象が作成されます。
- 新たな子を作成シンボル にてドロップすると、ドロップされたノードの子ノードとして選択対象が作成されます。

選択対象をドロップする際に、そのノードの兄弟または子ノードとして選択対象を作成するか決定することになります。

ノード作成方法の切り替え

選択対象をドロップするとコンテキストメニューが表示され、(i) 作成されるノードは要素になるのか属性になるのか、(ii) 選択対象をノードに変換するか、ノードの周りを選択対象で囲むのか、を選択することになります(以下のスクリーンショットを参照)。



以下の点に留意してください:

- 選択対象がノード(要素または属性)に変換される場合、選択対象がコンテンツプレースホルダを含むノードタグに置き換えられます。デザインならびに出力では、選択対象のテキストコンテンツが静的コンテンツから削除されます。出力では、選択対象のテキストが XML ドキュメント内にあるノードの動的コンテンツとして表示されます。
- HTML ノードが XML ノードへ変換されると、HTML ノードタグ内部に XML ノードタグが挿入されます。
- (HTML ノードを含む)選択対象が XML ノードにより囲まれると、選択対象の直前と直後に XML ノードが挿入されます。デザインならびに出力では、選択対象のテキストコンテンツが静的テキストとして保持されます。
- ノードタグは、必要なパス(つまりそのノードを含む祖先ノードタグ)とともに挿入されます。デザイン内のコンテキストノードにより、絶対パスまたは相対パスが作成されます。
- テーブルならびにリストの選択対象からノードを作成する方法については、要素/属性としてテーブルやリストを作成のセクションを参照ください。

スキーマ内のノードを追加、削除する

(HTML 選択対象をベースにしていない)その他のノードを、ユーザー定義スキーマへ追加することができます。ノードを右クリックして、コンテキストメニューから目的のコマンドを選択してください。スキーマツリーサイドバーのツールバーアイコンからも、コマンドを選択することができます。

ノードを削除するには、ノードが選択されている状態で、コンテキストメニューまたはツールバーアイコンから削除コマンドを選択します。ノードが削除されることで、既にあるパスが不正となってしまう場合もある点に注意してください。

デザインの修正

コンポーネントをドラッグして静的ならびに動的コンポーネントを挿入することで、デザインの構造を修正することができます。StyleVision に用意されているスタイル機能を使用することで、スタイルの修正を行うこともできます。

関連項目

- ユーザー定義スキーマ

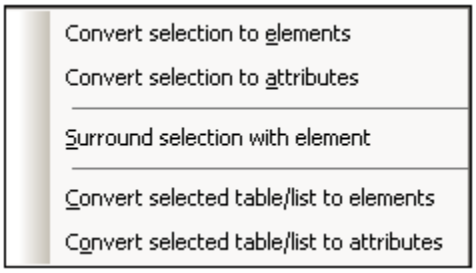
9.10.3 要素/属性としてテーブルやリストを作成

HTML ドキュメント内にあるテーブルならびにリストを XML スキーマの要素または属性へ変換することで、テーブルまたはリストの構造をスキーマ上に保つことができます。

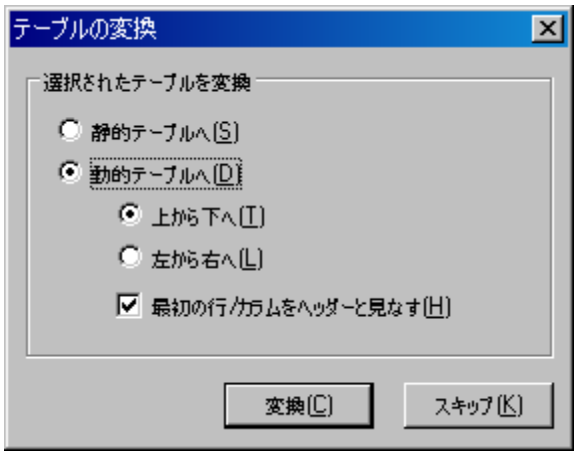
テーブルを要素/属性へ変換する

以下の操作により、テーブルをスキーマノードへ変換することができます:

1. HTML テーブル内部のテキストをハイライトしてテーブルを選択します。
2. スキーマツリー内にあるノードまで選択対象をドラッグして、兄弟または子ノードとして作成します。
3. マウスカーソルが兄弟ノードの作成シンボル ↓ または子ノードの作成シンボル ↴ になった時にノードをドロップします。
4. コンテキストメニューが表示されるので、**選択されたテーブル/リストを要素に変換**または**選択されたテーブル/リストを属性に変換**コマンドを選択します。



5. 表示される変換ダイアログにて、SPS 内に作成されるのが静的テーブルまたは動的テーブルなのかを選択します(以下のスクリーンショットを参照)。



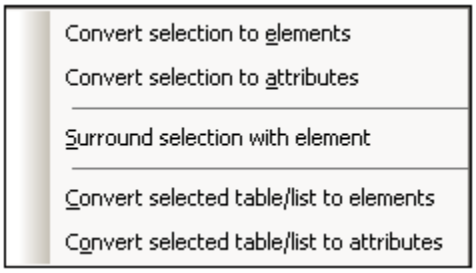
静的テーブルオプションが選択された場合、テーブルの各セルに対してスキーマノードが作成されます。デザインでは、各ノードにコンテンツプレースホルダが挿入されます。テーブルセル内のデータは、一時的な内部 XML ドキュメント(そして生成された XML ドキュメント)へコピーされます。**動的テーブル**オプションは、テーブル内にある全ての行の構造が同一である場合に選択することができます。ノードが SPS にて作成されると、動的テーブルにある行は(全ての行の構造が同一であるため)1つの行により表示されます。テーブルデータは XML ファイルへコピーされます。動的テーブルは上から下方向(垂直方向に行が表示されます)または左から右方向(行のデータがカラムとして表示されます)に展開することができます。最初の行/カラムをヘッダーとして指定すると、カラムヘッダーを含む最初の行が静的テキストとしてデザインに追加され、(ii) スキーマ要素/属性ノードの名前にはヘッダーが使用されます。最初の行/カラムがヘッダーとして指定されなかった場合、ヘッダー行がデザインに含まれることはありません。

6. オプションを選択した後には、**変換**をクリックして完了します。

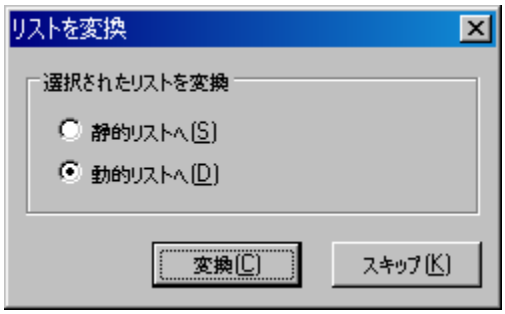
リストを 要素/属性へ変換する

以下の操作により、リストをスキーマノードへ変換することができます:

1. HTML リスト内部のテキストをハイライトしてリストを選択します。
2. スキーマツリー内にあるノードまで選択対象をドラッグして、兄弟または子ノードとして作成します。
3. マウスカーソルが兄弟ノードの作成シンボル ↓ または子ノードの作成シンボル ↳ になった時にノードをドロップします。
4. コンテキストメニューが表示されるので、**選択されたテーブル/リストを要素に変換**または**選択されたテーブル/リストを属性に変換**コマンドを選択します。



5. 表示される変換ダイアログにて、SPS 内に作成されるリストが静的リストまたは動的リストになるかを選択します(以下のスクリーンショットを参照)。



静的リストオプションが選択された場合、各リストアイテムに対してスキーマノードが作成されます。デザインでは、アイテムリストの静的コンテンツとして HTML リストアイテムのテキストが各ノードに挿入されます。**動的リスト**オプションが選択された場合、各リストアイテムがそれぞれ単一のリストアイテムノードとしてデザインに表示され、各リストアイテム要素にはコンテンツプレースホルダが挿入されます。

6. オプションを選択した後には、**変換**をクリックして完了します。

関連項目

- テーブルとの作業: SPS にてテーブルを使用する方法について記述されます。
- リストの作成

9.10.4 出力の生成

SPS を作成した後には、「**ファイル | 生成されたファイルの保存**」コマンドにより以下の出力ファイルを生成することができます:

- 生成されたユーザー定義スキーマ: スキーマツリーサイドバーにて作成したスキーマのことです。
- 生成されたユーザー定義 XML データ: 作成したスキーマをベースに、インポートされた HTML から得られたデータを含む XML ドキュメントのことです。
- HTML とRTF 出力の生成に使用される XSLT スタイルシート。
- HTML とRTF 出力。

(C) 2015-2021 Altova GmbH

9.11 ASPX インターフェイスによる Web アプリケーション

DB またはXML データのHTML レポートをSPS により作成する場合、通常 StyleVision では以下のようにレポートの作成が行われます:

1. ソースデータが DB の場合、完成された SPS が StyleVision にてアクティブになっている状態で、DB から XML ファイルを生成します(ソースデータが XML ファイルの場合、このステップは必要ありません)。
2. SPS から、XSLT-for-HTML ファイルを生成します。
3. 生成された HTML ファイルをサーバー上に配置します。

ウェブアプリケーションの場合、ソース(DB またはXML)データが修正されることでHTML ファイルが古くなることがあります。ウェブサーバー上のHTML ファイルを更新するには、(i) (DB をベースとしてデータの場合)XML ファイルを再度生成し、(ii) 新たな XML ファイルを XSLT-for-HTML により変換し、(iii) 生成されたHTML ファイルをサーバー上へ配置する必要があります。

StyleVision ではHTML ウェブページを素早く更新するための方法が提供されます。この機能によりASPX アプリケーションを自動的に生成することができ、ASPX アプリケーションファイルに必要な全てのファイル(.aspx ファイル、XSLT ファイル、コードファイル)が StyleVision により生成されます。これらのファイルをソースDB ファイルまたはXML ファイル、そしてXSLT-for-HTML とともにサーバー上に配置することで、ウェブインターフェースファイルの.aspx ファイルが更新される度に、(i) (DB をベースとしたデータの場合)DB から新たな XML ファイルが生成され、(ii) サーバー上にあるXSLT-for-HTML ファイルによりXML ファイルが変換され、(iii) 変換の出力がウェブインターフェースページに表示されます。これにより、最新のDB またはXML データがウェブインターフェースページに表示されます。

ASPX ソリューションのファイルを生成する

DB ベースのSPS またはXML ベースのSPS が作成された後には、以下の操作によりASPX ソリューションを作成することができます:

1. StyleVision にて SPS ファイルがアクティブになっている場合、メニューコマンドの「**ファイル | ウェブデザイン | ASPX ウェブアプリケーションを生成**」を選択することで、ASPX ファイルを生成することができます。ASPX アプリケーションファイルが指定されたフォルダーに作成されます。ASPX アプリケーションが生成されたフォルダーには、少なくとも以下のファイルが含まれます:

   - Readme.doc
   - SPSFilename.aspx
   - SPSFilename.xslt
   - SPSFilename.cs

2. DB ファイルまたは XML ファイルを、ASPX アプリケーションが配置されているサーバー上のフォルダーに配置します。.aspx ファイルがアプリケーションのエントリーポイントになります。このファイルが再度読み込まれた場合、表示される DB または XML のデータも更新されます。

メモ XSLT 変換を正しく行うには、Altova のAltovaXML アプリケーション をインストールする必要があります。変換を行う際に問題が発生する場合、AltovaXML のセットアップ方オフが記述されている ReadMe.doc ファイルを参照ください。

動作方法

ASPX アプリケーションを生成したフォルダーには、その他のファイルとともに以下のファイルが含まれます:

- Readme.doc
- SPSFilename.aspx
- SPSFilename.xslt
- SPSFilename.cs

SPSFilename.aspx が出力ドキュメントのURL となります。SPSFilename.aspx により、SPSFilename.cs. に含まれているC# コードが実行され、このC# コードにより(ファイル、またはDB から取得された)XML コンテンツが読み取られ、SPSFilename.xslt ファイルとともに AltovaXML へ渡されます(AltovaXML はAltova のXSLT 変換エンジンで、Altova ウェブサイトからダウンロードすることができます)。

AltovaXML により、与えられたXSLT ファイルを使ったXML コンテンツの変換が行われます。変換によりHTML ドキュメントが作成され、ウェブアプリケーションを介してブラウザーに表示されます。例えばDB に対する変更などによりXML のコンテンツが変更されると、SPSFilename.aspx の表示（またはブラウザー内に表示されているページの更新）を行うことで、最新のデータがデータベースやXML ファイルから取得され、ドキュメントのレンダリングが行われます。

9.11.1 サンプル: Windows 7 にて動作する localhost

以下に記述される操作手順により、ローカルホストにASPX アプリケーションをセットアップします。詳細については、ASPX アプリケーションフォルダーにある Readme.doc ファイルを参照ください。このフォルダーならびにファイルは、SPS ファイルがアクティブな状態でメニューオプションから「**ファイル | ウェブデザイン | ASPX ウェブアプリケーションを生成**」コマンドを選択することにより生成されます。

RaptorXML のインストール

最新バージョンのAltovaXML がインストールされていることを確認してください。AltovaXML はAltova の変換エンジンで、（DB より生成された）XML ファイルの変換に使用されます。チャートの生成といった、StyleVision に搭載された機能のいくつかは、AltovaXML のバージョンによってはサポートされないという点に注意してください。

Internet Information Services（Microsoft Web サーバー）の有効化

Internet Information Service（IIS）が有効化されていない場合、以下の操作を行なってください。ステップ 5 により、IIS が有効になっているか確認することができます。

1. Windows のスタートボタンから「**コントロールパネル | プログラムと機能 | Windows の機能の有効化または無効化**」を選択します。
2. **インターネットインフォメーションサービス**のチェックボックスをクリックします。チェックボックスが部分的に有効となります。
3. インターネットインフォメーションサービス以下にある「**World Wide Web サービス | アプリケーション開発機能 | ASP.NET**」のチェックボックスにチェックを入れます。
4. 「**OK**」をクリックします。処理が完了すると:、C:¥inetpub¥wwwroot という名前のフォルダーが作成されます。これがウェブサーバーのルートフォルダーになります。
5. ブラウザーから localhost へアクセスすることで、IIS のウェルカムスクリーンを確認することができます。

StyleVision にてASPX アプリケーションを生成する

以下の方法によりASPX アプリケーションを生成することができます:

1. データベースならびに SPS ファイルが同一のフォルダーにあることが推奨されます。
2. SPS ファイルが作成された後に、メニューオプションから「**ファイル | ウェブデザイン | ASPX ウェブアプリケーションを生成**」を選択します。
3. 表示されるダイアログにて、C:¥inetpub¥wwwroot 以下にフォルダーを作成し、そのフォルダーを選択します（例: C:¥inetpub¥wwwroot¥Test1）。
4. フォルダーの選択を確定すると、そのフォルダーに <FileName>.aspx、<FileName>_AltovaDataBaseExtractor.cs、Web.config ファイルが生成されます。

メモ C:/inetpub/wwwroot フォルダー以下にファイルを保存する場合、管理者として StyleVision を起動する必要があります。StyleVision を終了して、StyleVision のアイコンを右クリックして、**管理者として実行**を選択してください。

生成されたアプリケーションとASPX を関連付ける

以下の操作により、StyleVision により生成されたアプリケーションが、ASPX により認識されるようになります:

(C) 2015-2021 Altova GmbH

1. Windows のスタートボタンから「**コントロールパネル | 管理ツール | インターネットインフォメーションサービス (IIS) マネージャー**」を選択します。
2. 接続パネルにてフォルダーを展開します（例: Test1）。この時点でのフォルダーアイコンは、標準的な黄色で表示されます。
3. フォルダーのコンテキストメニューから「**アプリケーションへの変換**」を選択します。ダイアログにて「**OK**」をクリックします。フォルダーのアイコンが球形アイコンになります。
4. 接続パネルにあるツリーを展開して、アプリケーションプールを表示し、そのアイテムを選択します。
5. 接続ペイン内にあるアプリケーションプールアイテムを選択することで表示される DefaultAppPool のコンテキストメニューから「**詳細設定**」を選択します。
6. ID プロパティから**カスタムアカウント**を選択し、Windows ユーザー名とパスワードを入力します。
7. **32ビットアプリケーションの有効化**プロパティの値を True にセットします（これによりデータベースドライバーがアクセスできるようになります）。このステップは64ビットの Windows を使用している場合にだけ必要になります。

アプリケーションを実行する

ブラウザーにて localhost/Test1/<FileName>.aspx へアクセスします（ASPX アプリケーションが保存されたフォルダー名が Test1 で、生成元の SPS ファイル名が <FileName>.sps の場合）。変換された HTML がブラウザーに表示されます。ブラウザーに表示されている ASPX ページを再ロードすることで、データベースや XML 内にある最新のデータからページが再度生成されます。

メモ ブラウザーが応答待ち状態になるようであれば、AltovaXML ライセンスファイルが取得されていることを確認してください。

9.12 PXF ファイル：SPS とその関連ファイルのコンテナー

XSLT 2.0 を使用するSPS デザインはPortable XML Form (PXF) ファイルとして保存することができます。PXF フォーマットは SPS デザインを(スキーマファイルやソースXML ファイル、デザインで使用されるイメージファイル、ソースXML を出力フォーマットへ変換する XSLT ファイルといった) 関連するファイルとともにパッケージ化するために、Altova により開発されました。PXF ファイルフォーマットを使用することにより、Authentic View の編集や、Authentic View から出力を生成するために必要な全てのファイルを、1つのファイルとして配布することが可能になります。

このセクションではPXF ファイルの使用方法について、2つのパートに分けて説明します:

- PXF ファイルの作成
- PXF ファイルの編集
- PXF ファイルのデプロイ

メモ

PXF 機能に関する以下の点に注意してください：

- XSLT 2.0 または3.0. によりデザインされた SPS だけがサポートされます。
- XBRL ならびに DB XML カラムスキーマソースはサポートされません。これらのスキーマソースが含まれる場合、保存形式の選択を行うダイアログは表示されません。PXF ファイルを編集するにあたり、これらのスキーマソースを追加することはできません。

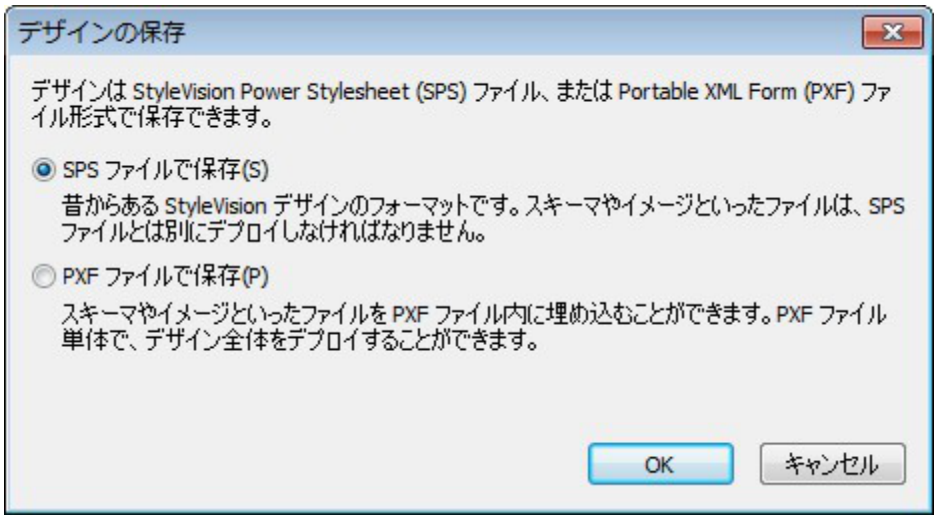
関連項目

- 使用方法の概要
- 名前を付けて保存

9.12.1 PXF ファイルの作成

SPS デザインと関連するファイルを含むPXF ファイルを作成するには、StyleVision にてSPS デザインを開き、メニューオプションから「**ファイル | 名前を付けて保存**」を選択します。デザインの保存ダイアログが表示されます(以下のスクリーンショットを参照)。

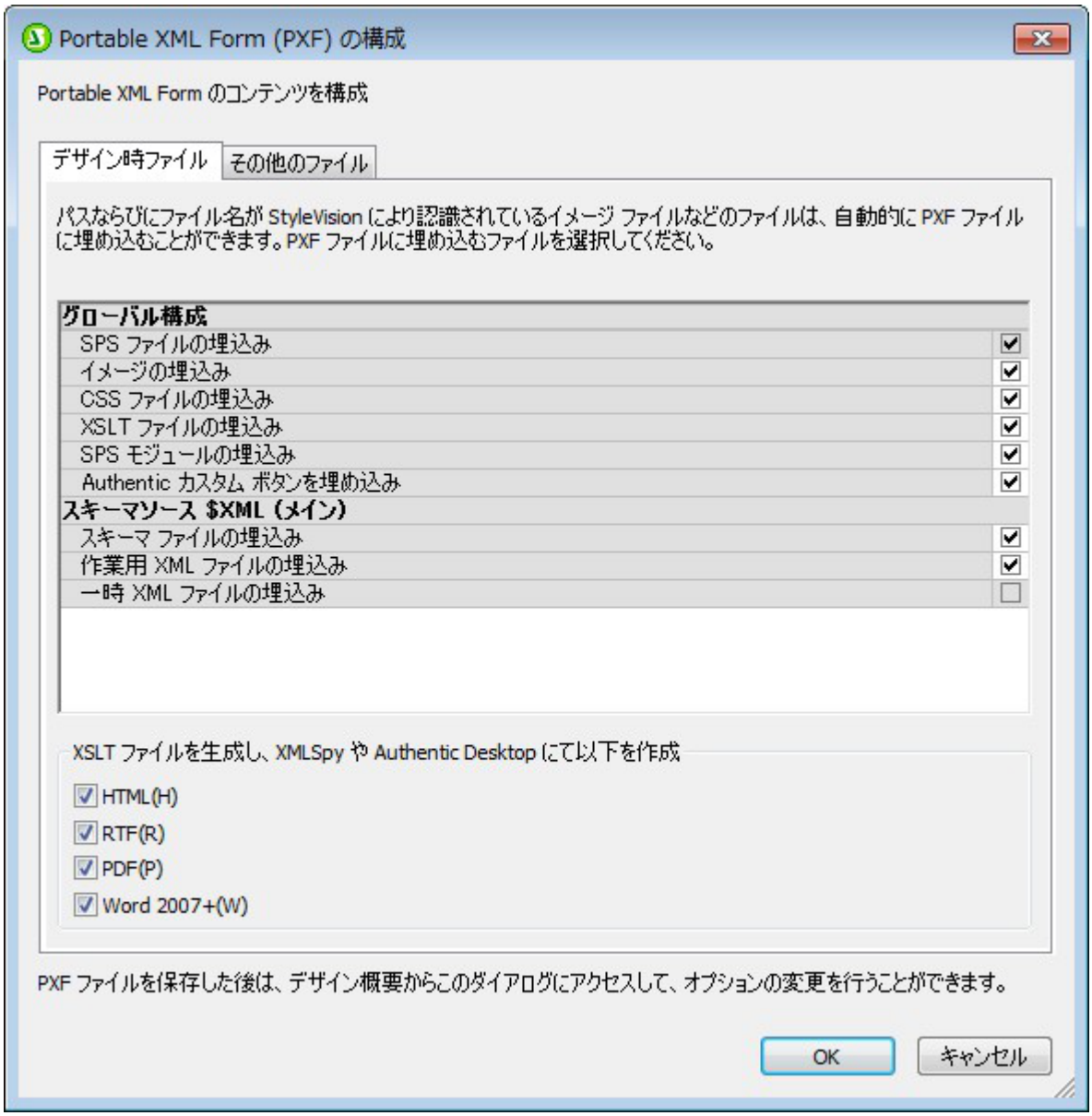
(C) 2015-2021 Altova GmbH



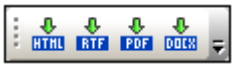
SPS フォーマットはStyleVision デザインに使用される標準のAltova フォーマットです。このセクションではPXF フォーマットに関する説明を行うため、SPS フォーマットについては説明しません。SPS としてファイルを保存する場合、ユーザーレファレンスのセクションにある詳細を参照ください。

PXF として保存する

PXF オプションを選択することにより、W indows システム標準の名前を付けて保存ダイアログが表示されます。ファイルの保存は、デザインを保存 コマンドと同様の方法に加え、PXF ファイルに追加するファイルを選択するためのステップが加わります。PXF ファイル名を指定すると、PXF の構成ダイアログが表示され、埋め込みを行うファイルを選択することができます（以下のスクリーンショットを参照）。



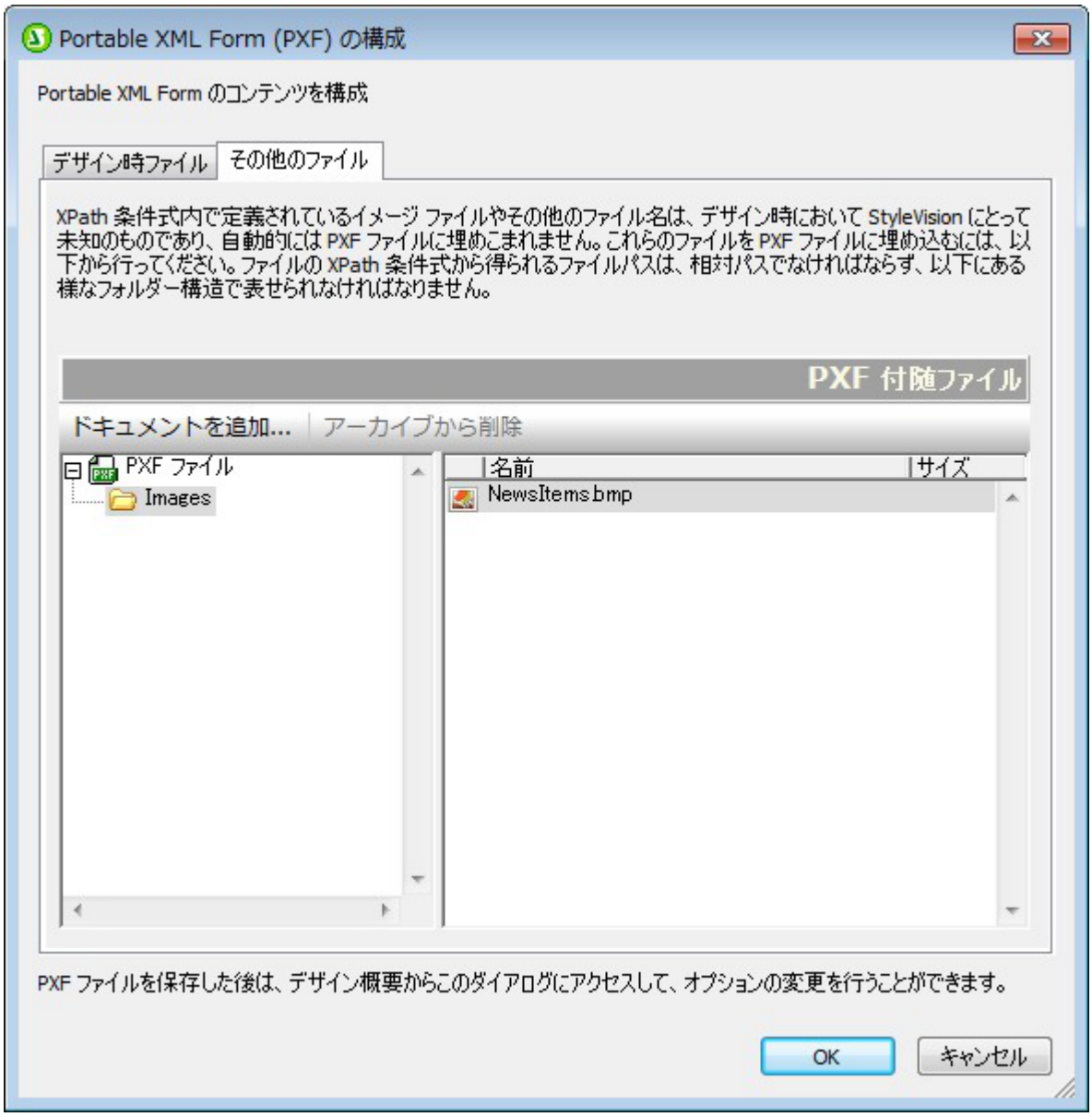
デザイン時ファイルタブのグローバル構成セクションでは、デザインに関連するソースファイルの埋め込みを選択/選択解除することができます。更に、デザインから生成されたXSLT ファイルを埋め込むかを選択することもできます。XSLT ファイルペインにて、埋め込まれたXSLT ファイルが生成する出力フォーマットを選択してください。XSLT ファイルがPXF ファイルに追加され、そのPXF ファイルがAltova 製品のAuthentic View にて開かれた場合、出力フォーマットを生成と閲覧を行うためのツールバーボタンがAuthentic View にて有効になります（以下のスクリーンショットを参照）。



メモ StyleVision の高いバージョンでのみサポートされている（*高から低い順: Enterprise*、*Professional*、*Basic*）出力のためのXSLT ファイルは、 PXF ファイル内で作成され、PXF ファイルが低いバージョンで開かれる場合、PXF ファイルを保存すると、低いバージョンでサポートされる出力のためのXSLT ファイルは、保存されません。PXF ファイルの保存を継続するかを問うプロンプトが表示、または、保存を中断して、サポートされないフォーマットを保管します。

その他のファイルタブでは、デザインには直接含まれていないファイルを指定することができます（以下のスクリーンショットを参照）。例えば、デ

ザイン内部のXPath 条件式により生成されたURL にて参照されるイメージファイルなどを含めることができます。以下のスクリーンショットでは、Images フォルダー以下にあるイメージのNewsItems.bmp がPXF ファイルに含まれるように指定されています。



その他のファイルをPXF ファイルに追加するには、**ドキュメントを追加**ボタンをクリックして目的のファイルを選択します。ファイルを開くダイアログが表示され、SPS ファイルが配置されているフォルダーが開かれます。このフォルダー内にあるファイルまたはその子孫フォルダー内にあるファイルを選択することができます。ファイルがPXF ファイルへ追加されると、そのファイルと対応するフォルダー構造が表示されます。上のスクリーンショットからは、SPS ファイルが配置されているフォルダーにある Images という名前のフォルダー内にNewsItems.bmp というファイルがあることが理解できます。

SPS ファイルが配置されているフォルダーの上位にあるフォルダーからファイルが選択されると、エラーが返されます。

SPS デザインでは、その他のファイルに対する参照を相対パスで、更に付属ファイルのペインに示されているとおりに記述する必要があります。例えば、上のスクリーンショットにある NewsItems.bmp には、Images/NewsItems.bmp という相対パスで参照を行う必要があります。

関連項目

- 使用方法の概要
- 名前を付けて保存

9.12.2 PXF ファイルの編集

PXF ファイルはメニューオプションの ファイル | 開く コマンドにより、StyleVision 内で開き、編集を行うことができます。編集には2種類の方法があります:

- PXF ファイルの構成を編集する。
- SPS や Authentic XML といった個々のファイル内にあるコンテンツを StyleVision にて編集する（イメージや CSS ファイルといったその他のコンポーネントファイルは外部アプリケーションにて編集することができます）。コンポーネントファイルは全て StyleVision にて明示的に更新する必要があります。

PXF の編集を行うエントリーポイント

PXF 構成の編集やPXF ファイルの更新を行うには、デザイン概要サイドバーにあるPXF アイテムを選択します（以下のスクリーンショットを参照）。

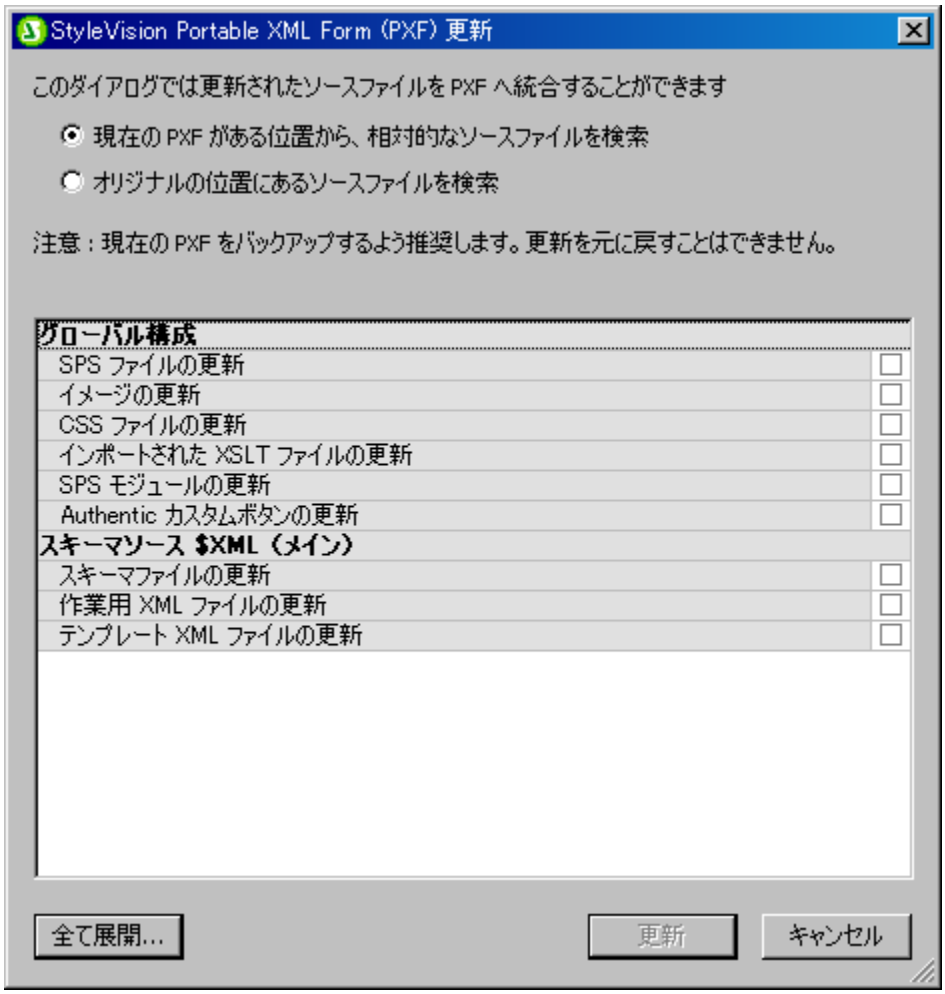
埋め込まれたファイルの構成

デザイン概要サイドバーにある**埋め込まれたファイルの構成**リンクをクリックすると、Portable XML Form (PXF) の構成ダイアログが表示されます。使用することのできるオプションは、PXF ファイルの作成を行う際に使用するものと同じものです。

埋め込まれたファイルの更新

デザイン概要サイドバーにある**埋め込まれたファイルの更新**リンクをクリックすると、Portable XML Form (PXF) の更新ダイアログが表示されます（以下のスクリーンショットを参照）。

(C) 2015-2021 Altova GmbH



ソースファイルの取得を、現在 PXF ファイルが配置されている場所からの相対的な位置により取得するのか、またはオリジナルの場所に従い行うのかを指定することができます。更に、更新するファイルのチェックボックスにチェックを入れて、更新ボタンをクリックすることでファイルの更新を行うことができます。新たな PXF ファイルが作成され、既存の PXF ファイルが上書きされます。そのため、更新を行う前には、オリジナルの PXF ファイルをバックアップすることが強く推奨されます。

関連項目

- 使用方法の概要
- 名前を付けて保存

9.12.3 PXF ファイルのデプロイ

PXF ファイルが作成されると、通常のデータファイルと同じような方法で移動、ダウンロード、コピー、そして保存を行うことができます。PXF ファイルには Authentic View にて必要な全ての XML ファイルを含めることができ、出力ファイルを生成することができるため、このファイルだけで Authentic ユーザーは出力を生成することが可能になります。

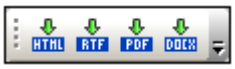
PXF ファイルは Altova 製品の Authentic View にて開く事ができます。XMLSpy にて PXF をどのように使用することができるかの使用シナリオを以下に示します：：

- メニューオプションの「**ファイル | 開く**」コマンドにより PXF ファイルを開きます。埋め込まれた XML ファイルが埋め込まれた SPS ファイルにより Authentic View で表示され、Authentic View にて編集することができます。メニューオプションの「**ファイル | デザインを保存**」コマンドにより、（内部に埋め込まれた XML ファイルが更新された）PXF ファイル内部の変更が保存されます。

- XML ファイルが埋め込まれていない PXF ファイルを「**ファイル | 開く**」コマンドにより開きます。XML ファイルが含まれていない場合、SPS デザインをベースにしたテンプレート XML ファイルが Authentic View にて開かれます。メニューオプションの「**ファイル | デザインを保存**」コマンドを選択することで、この XML ファイルは PXF ファイル内に埋め込まれたファイルとして保存されます。

- Altova 製品の XMLSpy では XML ファイルを PXF ファイルと関連付けることができ、PXF ファイルに埋め込まれた SPS ファイルを Authentic View の編集に使用することができます。関連付けは、メニューオプションの「**Authentic | SPS の割り当て**」コマンドにより、このような処理を行うことができます。変更が保存されると、これら変更点は XML ファイルに対して保存され、PXF ファイルは変更されないままとなります。

- 出力フォーマットの生成に使用される XSLT スタイルシートが PXF ファイル内に埋め込まれた場合、Authentic View ユーザーはそのフォーマットにて出力を生成することができるようになります。ドキュメントの出力は出力生成ツールバーボタンから行うことができます（以下のスクリーンショットを参照）。Authentic View にて表示される出力生成ツールバーのボタンは、その出力に対応する XSLT ファイルが、PXF ファイル内部に含まれる場合だけ有効となります。例えば、HTML と PDF に対する XSLT スタイルシートだけが PXF ファイルに含まれている場合、HTML ならびに PDF 出力のツールバーボタンだけが選択可能となり、その他の（RTF ならびに DocX）出力ボタンは選択不可能状態で表示されます。



メモ PXF ファイルがウェブサーバー上に配置されており、Authentic Browser プラグインにより使用される場合、サーバーにてそのファイルがブロックされていないことを確認する必要があります。PXF (.pxf) ファイルに対して（例えば IIS 管理パネルから）以下のような MIME タイプを追加することで対応できます: `application/x-zip-compressed`.

関連項目

- 使用方法の概要
- 名前を付けて保存

(C) 2015-2021 Altova GmbH

10 データベース

DB をベースにしたSPS が作成された場合、つまりSPS のメインスキーマとしてDB が使用される場合、以下のようにしてSPS を使用することができます:

- Authentic View にて DB を編集する。
- DB の構造をベースにした XML スキーマの生成(DB にスキーマが含まれない場合、IBM DB2 バージョン 9 といった XML DB にだけスキーマは含まれます)。
- DB から得られたデータを含むXML ファイルを生成する(目的のDB データがXML フォーマットではない場合)。
- HTML とRTF 出力を生成するための XSLT スタイルシートを生成する。
- HTML とRTF フォーマットにて(SPS デザインをベースにした)DB レポートを生成する。レポートは レポートは StyleVision にてプレビューすることができます。

DB がSPS におけるスキーマの(メインではない)ソースとなる場合、DB のデータをデザインコンポーネントへ含めることはできるものの、DB そのものを Authentic View で編集することはできません。編集することができるのは、XML ドキュメントまたはメインスキーマに関連付けられているDB となります。

標準的な手順

このセクションでは、StyleVision にてDB を使用する方法について説明します。StyleVision におけるDB の概要を イントロダクションのサブセクションで説明し、DB を使用する方法について順を追って説明します。このセクションでは非 XML DB とXML DB が明示的に区別されており、DB という用語は非 XML DB を表すために使用されます。XML DB はXML DB と常に記述されます。これら2種類のDB では異なる方法によりデータの選択が行われるため、DB の種類についてこのような区別を行う必要があります。

- DB への接続: IBM DB2 バージョン 9 より前のDB を含む、非 XML DB へ接続する方法について記述されます。
- DB データの選択: 非 XML DB とXML DB 内にあるテーブルの構造から、SPS にて使用されるスキーマとXML データがどのように選択されるかについて説明します。
- DB スキーマならびにDB XML ファイル: (非 XML DB の)DB テーブルがロードされると、DB の構造とデータからXML スキーマとXML データファイルがそれぞれ一時的に生成されます。XML DB の場合、スキーマやXML ファイルがStyleVision から生成されることはなく、DB から直接参照されるか、(スキーマの場合は)他の場所から参照されることになります。
- DB フィルター: DB データのフィルタリング: 一時的な XML ファイルへロードされたDB データをフィルタリングすることができます。
- DB のためのSPS デザイン機能: SPS では、DB コントロールやDB クエリといったDB の機能を使用することができます。
- 出力ファイルの生成: StyleVision により生成されたDB レポートに関連したファイル。

サポートされるデータベース

次のデータベースはサポートされています。各データベースの使用できるルート オブジェクトもリストされています。Altova は他のODBC/ADO データベースへのサポートの向上のための努力をしていますが、以下にリストされたデータベースのみで接続とデータ処理のテストに成功しています。使用中のAltova アプリケーションが、64 ビット バージョンの場合、接続されているデータベース特定の64 ビットデータベース ドライバーにアクセスしてください。Altova アプリケーションはIBM iSeries データベースのロジカルファイルをサポートし、ロジカルファイルを表示します。

データベース	メモ
Firebird 2.5.4, 3.0	
IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5	
IBM Db2 for i 6.1, 7.1, 7.2, 7.3, 7.4	ロジカルファイルはサポートされ、ビューとして表示されています。
IBM Informix 11.70, 12.10, 14.10	

データベース	メモ
MariaDB 10.2, 10.3, 10.4, 10.5	
Microsoft Access 2003, 2007, 2010, 2013, 2016, 2019	ドキュメントの作成時（2019 年9 月初旬）Access 2019 のために使用可能な Microsoft Access Runtime が存在しませんでした。Microsoft Access 2016 Runtime がインストールされており、データベースが「大量」のデータ型を使用しない場合のみ Altova 製品から Access 2019 にデータベース接続することができます。
Microsoft Azure SQL Database	SQL Server 2016 codebase
Microsoft SQL Server 2005, 2008, 2012, 2014, 2016, 2017, 2019 Microsoft SQL Server on Linux	
MySQL 5.0, 5.1, 5.5, 5.6, 5.7, 8.0	
Oracle 9i, 10g, 11g, 12c, 18, 19	
PostgreSQL 8.0, 8.1, 8.2, 8.3, 9.0.10, 9.1.6, 9.2.1, 9.4, 9.5, 9.6, 10, 11, 12.1	PostgreSQL 接続は、ODBC または JDBC などのインターフェイス（ドライバ）によりネイティブとドライバーベースの接続でサポートされています。ネイティブな接続はドライバを必要としません。
Progress OpenEdge 11.6	
SQLite 3.33.0	SQLite 接続は、ネイティブとしてサポートされます。SQLite データベースファイルへの直接の接続です。 別途のドライバーは必要ありません。Authentic ビューでは、SQLite からのデータを編集することはできません。Authentic ビューで、SQLite からのデータを保存する場合は、既知の制限についてのメッセージが通知されます。
Sybase ASE 15, 16	
Teradata 16	

関連項目

- SPS とは?

(C) 2015-2021 Altova GmbH

10.1　データベース と StyleVision

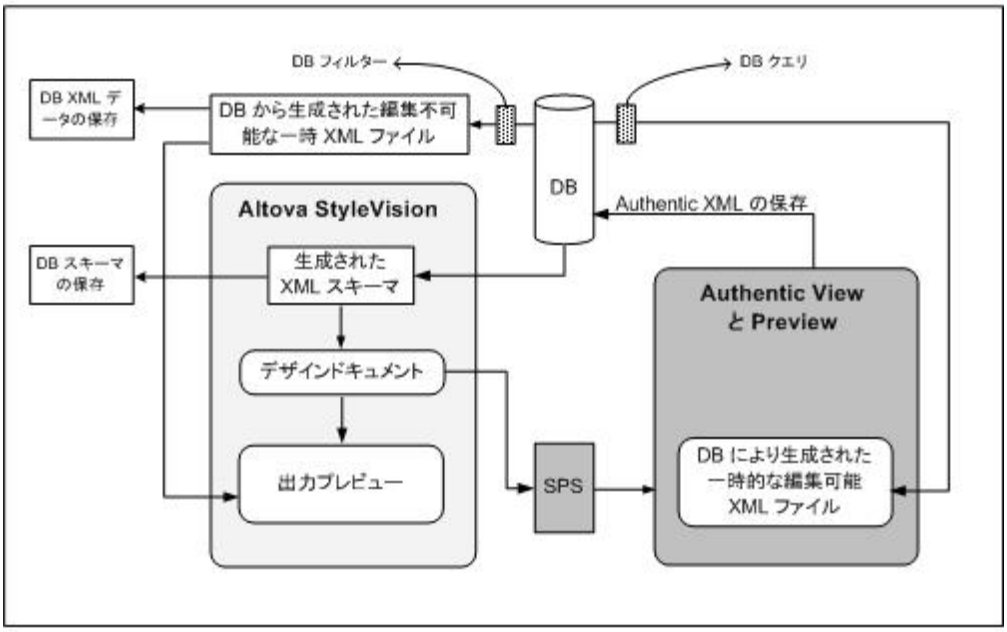
StyleVision ではDB をベースにしたSPS を作成することができます。スタイルシートにより以下に記される処理を行うことができます:

- Authentic View にてDB を編集する
- DB のレポートを生成する

SPS を生成すると、StyleVision にてサポートを確認して、HTML とRTF フォーマットにてファイルを生成することができます。StyleVision により生成された DB に関連するファイルを保存することもできます:

- DB 構造をベースにしたXML スキーマ(スキーマが既に保管されているXML DB にはお適用されません)
- 生成されたスキーマにて定義された構造とDB から得られたコンテンツを伴うXML ファイル(データがXML フォーマットで保管されているXML DB にはお適用されません)
- 生成されたスキーマをベースにデザインされたSPS
- HTML を出力する(SPS のデザインをベースとした)XSLT スタイルシート
- RTF を出力する(SPS のデザインをベースとした)XSLT スタイルシート
- HTML 出力
- RTF 出力

保存されたXML ファイルは適切なXSLT スタイルシートにより処理されます。このような機能と構成により柔軟なレポート生成を行うことができます。



メモ StyleVision にて非 XML DB から生成された XML スキーマならびに XML ファイルの構造やコンテンツを Authentic View に適合するかたちに修正することはできません。これらファイルの構造は、非 XML DB の構造から得られたもので、StyleVision により DB から生成された XML 構造により DB の編集や DB のレポート生成は行われることになります。

DB をベースにした SPS を処理する大まかなメカニズム
DB をベースにした SPS を使用する際のステップを以下に示します。これらのステップでは、DB の編集と、DB から HTML と RTF 形式のレポートを作成するまでがカバーされます。

- StyleVision によるDB への接続。接続を行う際には、XML スキーマとして生成しないデータテーブルのフィルタリングを行うことができます。
- 接続を確立したら、DB の構造をベースにした一時的な XML スキーマ が生成され、StyleVision のスキーマウィンドウにそのスキーマがツリー形式で表示されます。XML DB の場合、(DB 内部またはファイルのある場所に) 既に存在するスキーマが登録されます。
- 一時的な StyleVision-内部 XML ファイルも作成されます。により作成されます。編集不可能なファイルはプレビューならびに生成された XML データファイルのソースとして使用されます(上の図を参照ください)。Authentic View にて表示されるのが編集可能なファイルとなります。Authentic View にてファイルに対して行われた変更が(「**ファイル | Authentic XML データを保存する**」コマンドにより) 保存された場合、DB に対して変更が保存されます。出力ビューが新たにアクセスされた、または XML データファイルが保存された際に、編集不可能な XML ファイルが更新されます。
- StyleVision では、トップレベルフィルタートップレベルフィルターを定義することにより(出力ビューなどで使用される) 編集不可能な XML ファイルへインポートされるデータを制限することができます。
- DB クエリ クエリにより Authentic View に表示されるレコードのリストを制限することができます。この機能は編集時にだけ使用されます。.
- 変更点が DB に保存された場合、出力ビューウィンドウが次回アクセスされた時に編集不可能な XML ファイルが新たな DB のコンテンツとともに更新され、最新状態のファイルがプレビューに表示されます。
- DB をベースにした SPS は、ノードをデザインウィンドウへドラッグアンドドロップし、静的なスタイルシートコンポーネントの挿入、表示プロパティの割り当てといった、標準的なスキーマをベースにした SPS と同様の方法で作成されます。このドキュメンテーションでは、これらのメカニズムについて記述されます。

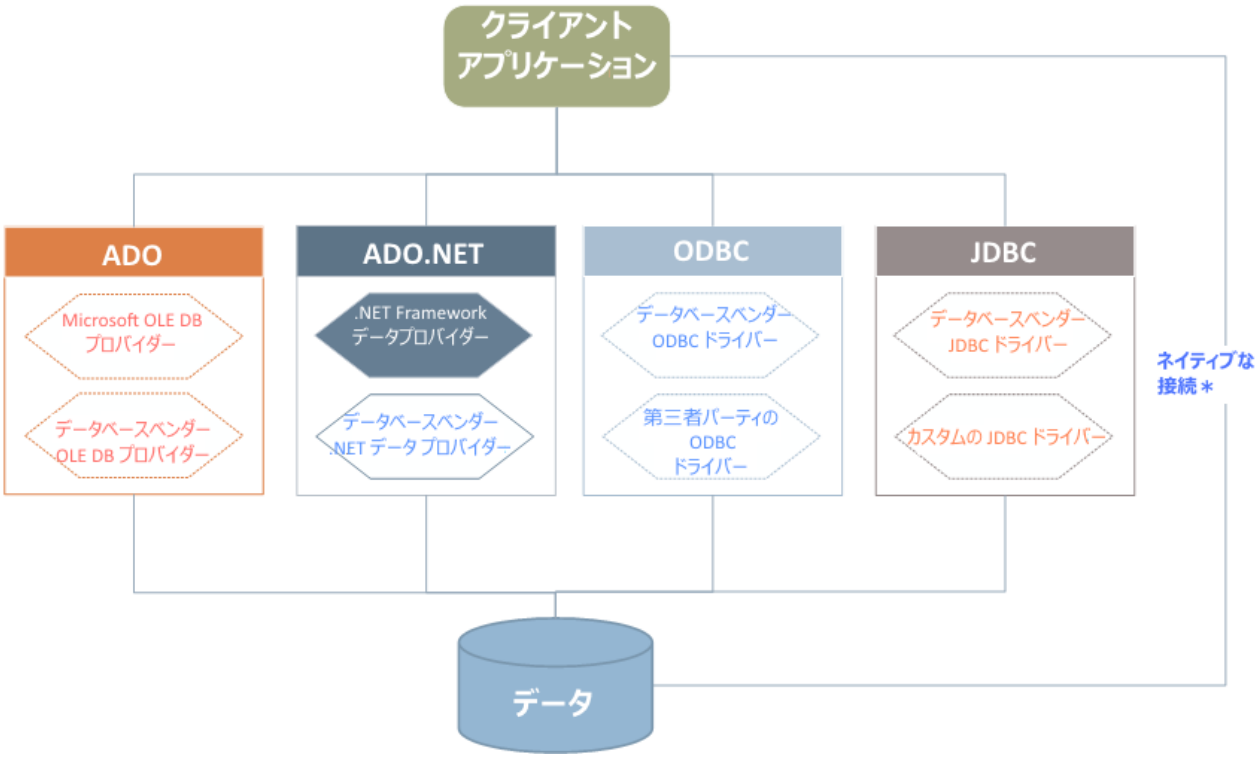
(C) 2015-2021 Altova GmbH

10.2　データソースへの接続

最も簡単なケースとして、データベースは、Microsoft Access、または、SQLite データベースファイルなどのローカルファイルが挙げられます。高度なケースの例としては、データベースは、アプリケーションが接続し、データを消費するオペレーティングシステムと同じオペレーティングシステムを使用するとは限らない、リモート、または、ネットワークデータベースサーバーに存在します。例えば、StyleVision が、Windows で作動し、データ（例えばMySQL）へアクセスするデータベースがLinux マシーンで作動することは可能です。

多種のデータベース型とリモートおよびローカルで作業するには、StyleVision は、データ接続インターフェイスとオペレーティングシステム内で使用することのできる、または、定期的にメジャーなデータベースベンダーによりリリースされるデータベースドライバーに従います。常に進化するデータベース技術の景観の中、このアプローチは、クロスプラットフォームに対する柔軟性および相互運用性に対応します。すなわち、StyleVision により、以下のデータアクセス技術により主要なデータベースにアクセスすることができます

次の図は、最も簡単な方法で、（ジェネリックなクライアントアプリケーションとして表示されている）StyleVision と（データベースサーバー、または、データベースファイルであることができる）データ保管の間で使用することのできるデータ接続のオプションを示しています。



＊直接なネイティブな接続 は、SQLite とPostgreSQL データベースのためにサポートされています。使用中のシステムにインストールするために追加ドライバーは必要ではありません。

上の図で示されるように、StyleVision は、主なデータベース型に次のアクセス技術を使用してアクセスすることができます:

- 基となるOLE DB（Object Linking and Embedding, Database）プロバイダーを使用するADO（Microsoft® ActiveX® Data Objects)
- ADO.NET（データとのインタアクションを有効化するMicrosoft .NET Framework 内で使用することのできるライブラリのセット）
- JDBC（Java データベース接続）
- ODBC（Open データベース接続）

ADO.NET の一部はサポートされていない、または、サポートが制限されている場合があります。次を参照してください: ADO.NET サポートメモ。

選択するデータベースへの接続のインターフェイスは、既存のソフトウェアのインフラストラクチャによります。データのアクセス技術と、接続するデータベースと密に統合することのできる、データベースドライバーを通常選択します。例えば、Microsoft Access 2013 データベースに接続するには、Microsoft Office Access データベース Engine OLE DB プロバイダーなどのネイティブなプロバイダを使用する ADO 接続文字列を作成します。に接続するには、他方、最新の JDBC、ODBC、または ADO.NET インターフェイスを Oracle Web サイトからダウンロードします。

Windows 製品 (Microsoft Access または SQL Server など) のためのドライバーは、Windows オペレーティングシステムで使用可能ですが、他のデータベースの種類では使用することができない可能性があります。メジャーなデータベースベンダーは、対応するデータベースに ADO、ADO.NET、ODBC、または JDBC の組み合わせからクロスプラットフォームアクセスをすることができる、一般使用可能なデータベースクライアントソフトウェアとドライバーを定期的にリリースします。更に、上記の技術のためのサードパーティーのドライバーを使用することができます。多くの場合は、1つ以上の方法で必要なデータベースへ使用中のオペレーティングシステムから接続することができます、この結果、StyleVision からも接続することができます。使用可能な機能、パフォーマンスパラメーター、と既知の問題はデータアクセス技術および使用されるドライバーにより異なります。

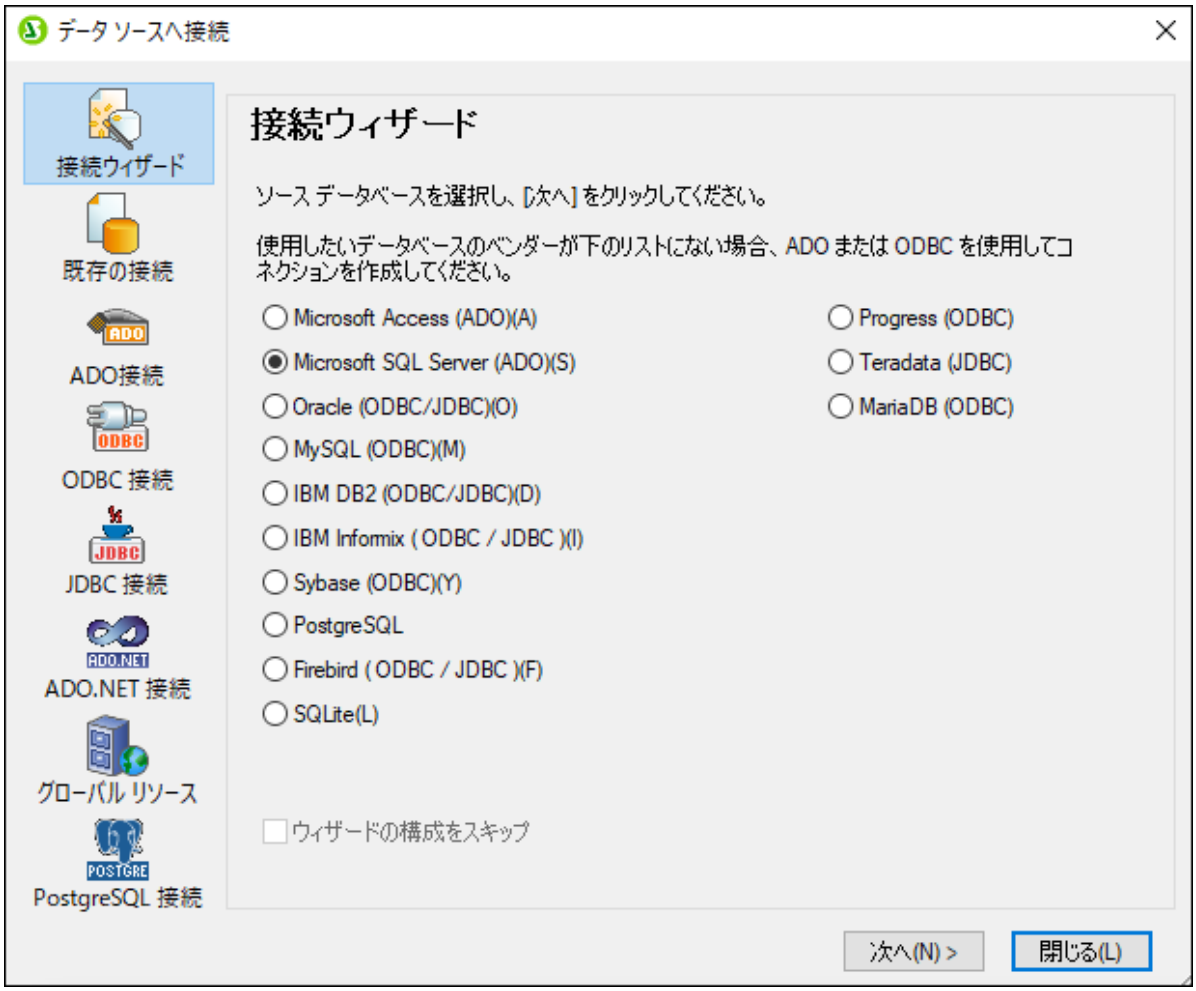
10.2.1 接続ウィザードの開始

StyleVision にはデータソースをセットアップするために必要とするステップを導くウィザードが搭載されています。

ウィザードのステップを開始する前に、データベースの一部の種類は、データベースドライバーまたはデータベースクライアントソフトウェアなどの複数のデータベースの必須条件を個別にインストールまたは構成する必要があることご注意してください。これらは、通常各データベースのベンダーにより提供され、特定の Windows バージョン用に調整されたドキュメントを含みます。データベースの種類別にグループ化されたデータベースドライバーのリストに関しては、をデータベースドライバーの概要参照してください。

データベース接続ウィザードを開始するには次を行ってください:

- 「ファイル」メニューから、「新規作成」をクリックし、「DB から新規作成」をクリックします。



データベース型を選択し「次へ」をクリックすると データベースの種類、技術（ADO, ADO.NET, ODBC, JDBC）と使用されるドライバーにより異なる命令が画面に表示されます。

例えば、各データベースの種類に適用できるサンプルはデータベース接続のサンプル のセクションを参照してください。各データベースへのアクセス技術に関しては以下のトピックを参照してください：

- ADO 接続の設定
- ADO.NET 接続の設定
- ODBC 接続の設定
- JDBC 接続の設定

10.2.2　データベースドライバー概要

以下のテーブルは、特定のデータベースに特定のデータアクセス技術を使用して接続するためによく使用されるデータベースのドライバーのリストです。このリストは、完全または規範的なものではありません。以下に表示されているドライバに加え他のネイティブまたはサードパーティー代替を使用することができます。

Windows オペレーティングシステムの複数のデータベースドライバーが既に使用されている場合がありますが、代替ドライバーをダウンロードする必要があります。データベースの一部に対して、オペレーティングシステムに付随するドライバーよりもデータベースベンダーより提供される最新のドライバーのパフォーマンスの方が良い場合があります。

データベースのベンダーは、個別にダウンロードすることのできるパッケージ、または、バンドルされたデータベースクライアントソフトウェアを提供します。後者の場合、データベースクライアントソフトウェアは、通常、データベースの管理と接続、およびデータベースクライアントとそのコンポーネントのインストールと構成を簡素化するために管理と構成ユーティリティにより構成されています。

データベースクライアントを正確に構成することは、データベースへの接続を成功させるために必須です。データベースをインストールして、使用する前に、各データベースバージョンおよび各 Windows バージョンにより通常異なるデータベースのインストールおよび構成の手順を確認することが奨励されます。

データベースの種類に関して各データアクセステクノロジーの機能と制限を理解するには、特定のデータベース製品のドキュメントを参照し、特定の環境に対しての接続をテストします。よく発生する接続問題を回避するために、以下に注意してください。:

- ADO.NET の一部のプロバイダーはサポートされない、または、サポートが制限されています。次を参照してください: ADO.NET サポートメモ。
- データベースドライバーをインストールする際、Altova アプリケーション（32-ビット または 64-ビット）などの同じプラットフォームが使用されることが奨励されます。例えば、32ビットのAltova アプリケーションが64-ビットオペレーティングシステムで使用される場合、が使用される場合、32-ビットのドライバーをインストールし、32-ビットのドライバーを使用して、データベースの接続をセットアップします。詳細に関しては、使用可能な ODBC ドライバーの表示 を参照してください。
- ODBC データソースをセットアップする際、データソース名（DSN）をユーザー *DSN* の代わりに *システム DSN* として作成することが奨励されます。詳細に関しては、使用可能な ODBC ドライバーの表示 を参照してください。
- When setting
- JDBC データソースをセットアップする際、JRE（Java Runtime Environment）、Java Development Kit（JDK）がインストールされている、またオペレーティングシステムのCLASSPATH 環境変数が構成されていることを確認してください。詳細に関しては、JDBC 接続のセットアップを参照してください。
- データベースベンダーからインストールするドライバーおよびデータベースクライアントソフトウェアのインストールの手順とサポートの詳細は、インストールパッケージのドキュメントを参照してください。公式なまたはサードパーティデータベースドライバーの場合、特定のオペレーティングシステムでの特定のドライバーに適用することのできる最も包括的な情報と構成手順は、通常ドライバーインストールパッケージの一部です。

データベース	インターフェイス	ドライバー
Firebird	ADO.NET	Firebird ADO.NET Data Provider (https://www.firebirdsql.org/en/additional-downloads/)
	JDBC	Firebird JDBC driver (https://www.firebirdsql.org/en/jdbc-driver/)
	ODBC	Firebird ODBC driver (https://www.firebirdsql.org/en/odbc-driver/)
IBM DB2	ADO	IBM OLE DB Provider for DB2
	ADO.NET	IBM Data Server Provider for .NET
	JDBC	IBM Data Server Driver for JDBC and SQLJ
	ODBC	IBM DB2 ODBC Driver
IBM DB2 for i	ADO	• IBM DB2 for i5/OS IBMDA400 OLE DB Provider • IBM DB2 for i5/OS IBMDARLA OLE DB Provider • IBM DB2 for i5/OS IBMDASQL OLE DB Provider

データベース	インターフェイス	ドライバー
	ADO.NET	.NET Framework Data Provider for IBM i
	JDBC	IBM Toolbox for Java JDBC Driver
	ODBC	iSeries Access ODBC Driver
IBM Informix	ADO	IBM Informix OLE DB Provider
	JDBC	IBM Informix JDBC Driver
	ODBC	IBM Informix ODBC Driver
Microsoft Access	ADO	• Microsoft Jet OLE DB Provider • Microsoft Access Database Engine OLE DB Provider
	ADO.NET	.NET Framework Data Provider for OLE DB
	ODBC	• Microsoft Access Driver
MariaDB	ADO.NET	MariaDB のための専用の.NET コネクタが不足している場合、MySQL のためのコネクタ/NET を使用してください（https://dev.mysql.com/downloads/connector/net/).
	JDBC	MariaDB Connector/J (https://downloads.mariadb.org/)
	ODBC	MariaDB Connector/ODBC (https://downloads.mariadb.org/)
Microsoft SQL Server	ADO	• Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) • Microsoft OLE DB Provider for SQL Server (SQLOLEDB) • SQL Server Native Client (SQLNCLI)
	ADO.NET	• .NET Framework Data Provider for SQL Server • .NET Framework Data Provider for OLE DB
	JDBC	• Microsoft JDBC Driver for SQL Server (https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server)
	ODBC	• ODBC Driver for Microsoft SQL Server (https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server)
MySQL	ADO.NET	• Connector/NET (https://dev.mysql.com/downloads/connector/net/)
	JDBC	Connector/J (https://dev.mysql.com/downloads/connector/j/)
	ODBC	Connector/ODBC (https://dev.mysql.com/downloads/connector/odbc/)
Oracle	ADO	• Oracle Provider for OLE DB • Microsoft OLE DB Provider for Oracle
	ADO.NET	Oracle Data Provider for .NET (http://www.oracle.com/technetwork/topics/dotnet/index-085163.html)
	JDBC	• JDBC Thin Driver

データベース	インターフェイス	ドライバー
		• JDBC Oracle Call Interface (OCI) Driver これらのドライバーは、通常、データベースクライアントのインストール中にインストールされます。（Thin ドライバーではない） OCI ドライバーを使用する接続は、Oracle XML DB のコンポーネントを使用しています。
	ODBC	• Microsoft ODBC for Oracle • Oracle ODBC Driver（通常、Oracle データベースクライアントのインストール中にインストールされます）
PostgreSQL	JDBC	PostgreSQL JDBC Driver (https://jdbc.postgresql.org/download.html)
	ODBC	psqlODBC (https://odbc.postgresql.org/)
	Native Connection	使用可能。ネイティブな接続を使用する場合、ドライバーをインストールする必要はありません。
Progress OpenEdge	JDBC	JDBC Connector (https://www.progress.com/jdbc/openedge)
	ODBC	ODBC Connector (https://www.progress.com/odbc/openedge)
SQLite	Native Connection	使用可能。ネイティブな接続を使用する場合、ドライバーをインストールする必要はありません。
Sybase	ADO	Sybase ASE OLE DB Provider
	JDBC	jConnect™ for JDBC
	ODBC	Sybase ASE ODBC Driver
Teradata	ADO.NET	.NET Data Provider for Teradata (https://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata)
	JDBC	Teradata JDBC Driver (https://downloads.teradata.com/download/connectivity/jdbc-driver)
	ODBC	Teradata ODBC Driver for Windows (https://downloads.teradata.com/download/connectivity/odbc-driver/windows)

10.2.3 ADO 接続の設定

Microsoft ActiveX Data Objects（ADO）はOLE DB を介して多種のデータソースにデータベースアクセステクノロジーです。OLE DB はODBC またはJDBC への代替インターフェイスです。COM（コンポーネント オブジェクト モデル）環境内での統一されたアクセスを提供します。ADO は新規のADO.NET 前のバージョンであり、他のデータソースのためにも使用可能ですがMicrosoft Access or SQL Server などのMicrosoft ネイティブなデータベースにアクセスを可能にする方法です。

需要な点は、複数のADO プロバイダー間から選択することができ、一部はダウンロードされワークステーションに使用する前にインストールされる必要があります。SQL Server への接続のために以下のADO プロバイダーを使用することができます:

- Microsoft OLE DB *Driver* for SQL Server (MSOLEDBSQL)

- Microsoft OLE DB *Provider* for SQL Server (SQLOLEDB)
- SQL Server Native Client (SQLNCLI)

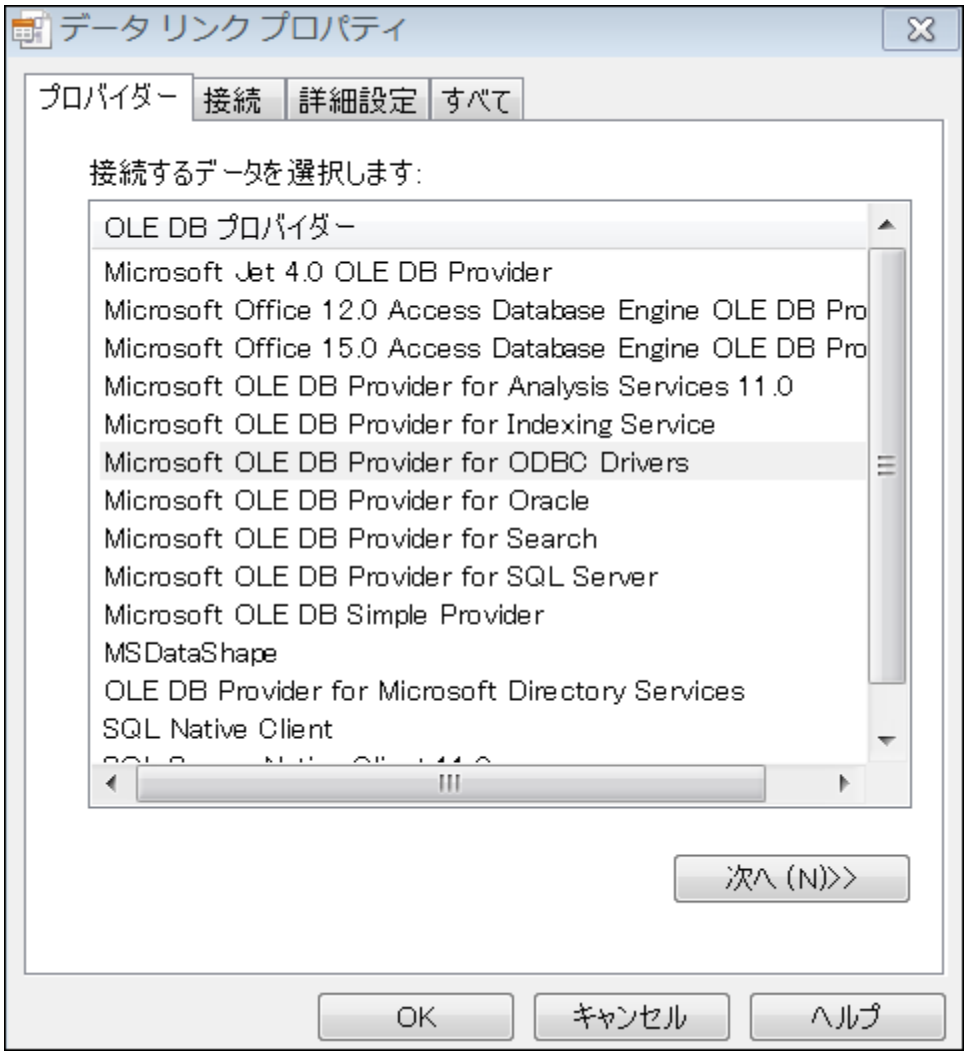
上記でリストされたプロバイダーで奨励されるプロバイダーはMSOLEDBSQL です。https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 からダウンロード可能です。StyleVision（32-ビット または64-ビット）のプラットフォームに一致する必要があることご注意してください。SQLOLEDB とSQLNCLI プロバイダーは非奨励であると考えられています。

Microsoft OLE DB Provider for SQL Server (SQLOLEDB) は共通テーブル式（CTE）とネストされたSELECT ステートメントなどの複雑なクエリのパラメーターバインディングに関する問題があることが知られています。

ADO 接続の設定:

1. データベース接続ウィザードを開始します。
2. 「ADO 接続」をクリックします。

3. 構築 をクリックします。



4. 接続するデータプロバイダーを選択します。下のテーブルは一般的なシナリオをリストしています。

このデータベースに接続する方法...	このプロバイダーを使用する...
Microsoft Access	• **Microsoft Office Access Database Engine OLE DB Provider** (奨励) • **Microsoft Jet OLE DB Provider** **Microsoft Office Access Database Engine OLE DB Provider** がリスト内で使用できない場合 Microsoft Access または Microsoft Access Database Engine Redistributable (https://www.microsoft.com/en-us/download/details.aspx?id=54920) がコンピューターにインストールされていることを確認してください。
SQL Server	• **Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)** が OLE DB プロバイダーのために奨励されています。リストにこのプロバイダーが表示されるためには、プロバイダーが https://docs.microsoft.com/en-

このデータベースに接続する方法…	このプロバイダーを使用する…
	us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 からダウンロードされインストールされる必要があります。 • **Microsoft OLE DB Provider for SQL Server (OLEDBSQL)** • **SQL Server Native Client (SQLNCLI)**
他のデータベース	データベースに適用可能なプロバイダーを選択してください。 視聴中のデータベースへのOLE DB プロバイダーが使用できない場合 データベースベンダーから必要なドライバーをインストールします（ データベースドライバー概要を参照してください）。代わりにADO.NET、ODBC、またはJDBC 接続を設定します。 オペレーティングシステムが必要なデータベースのためにODBC ドライバーを必要とする場合 **Microsoft OLE DB Provider for ODBC Drivers** を使用することもできます。またはODBC 接続を使用してください。

5. プロバイダーを選択後 [次へ] をクリックしてウィザードを完了します。

ウィザードの次のステップは選択されたプロバイダーにより異なります。SQL Server には、データベースサーバーのホスト名、認証メソッド、データベース名、およびデータベースユーザー名とパスワードを提供または選択する必要があります。サンプルに関してはMicrosoft SQL Server（ADO）への接続 セクションを参照ください。Microsoft Access に関しては、データベースファイルへのパスを参照または提供するように問われます。サンプルに関してはMicrosoft Access（ADO）への接続 セクションを参照ください。

初期化プロパティの完全なリスト（接続パラメーター）は接続ダイアログボックスの **全ての** タブで使用可能です。これらのプロパティは選択されたプロバイダーにより異なり、接続を可能にするために明示的に設定される必要がある場合があります。以下のセクションはMicrosoft Access とSQL Server データベースのための基本的な初期化プロパティを構成する方法について説明しています。

- SQL Server データリンクプロパティの設定
- Microsoft Access データリンクプロパティの設定

10.2.3.1 既存の Microsoft Access データベースへの接続

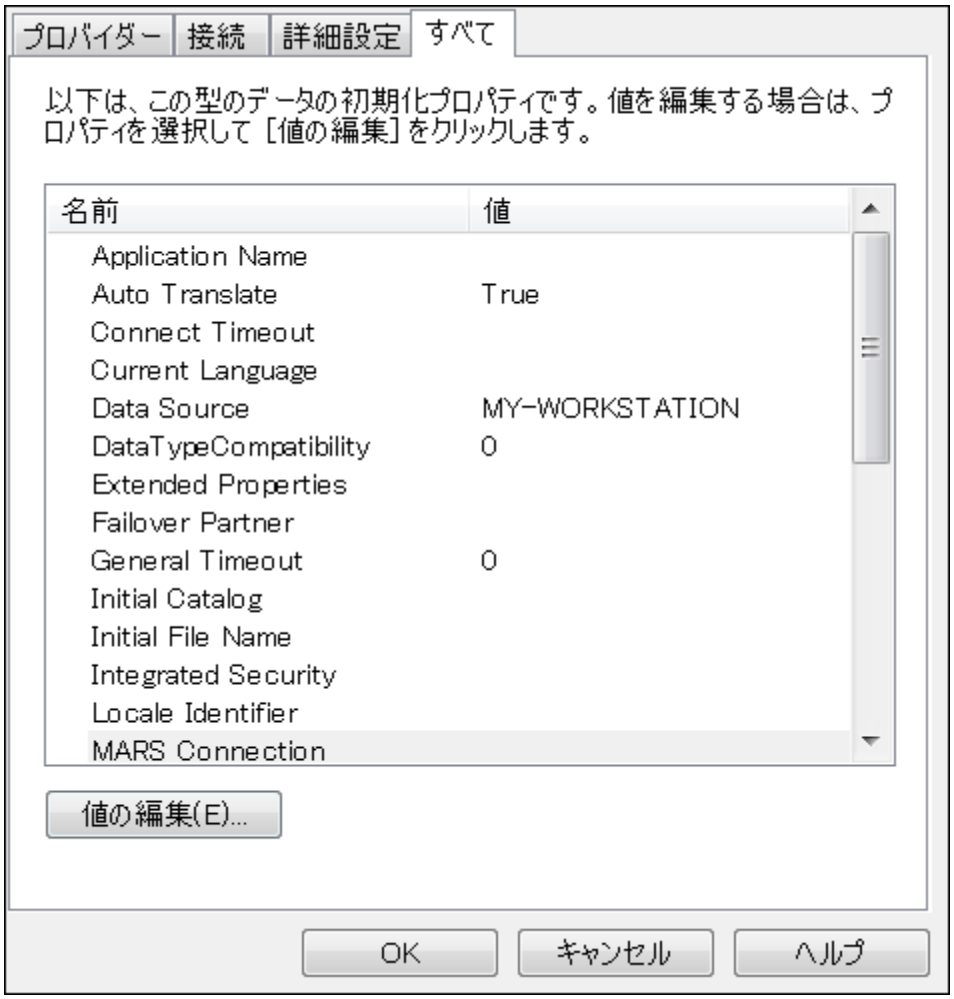
このアプローチは、パスワードで保護されていないMicrosoft Access データベースに接続する場合に適しています。データベースがパスワードにより保護されている場合、Microsoft Access（ADO）への接続 への接続で説明されている通りデータベースパスワードを設定してください。

既存のMicrosoft Access データベースへの接続方法:

1. データベース接続ウィザードを実行します（ データベース接続ウィザードの開始を参照）。
2. **Microsoft Access (ADO)** を選択して、「次へ」をクリックします。
3. データベースファイルを参照、または、（相対または絶対）パスを入力します。
4. 「**接続**」をクリックします。

10.2.3.2 SQL Server データ リンク プロパティの設定

Microsoft SQL Server データベースにADO を介して接続する場合、次のデータリンク プロパティダイアログボックスの全ての タブ内の次の接続プロパティを設定する必要があります。

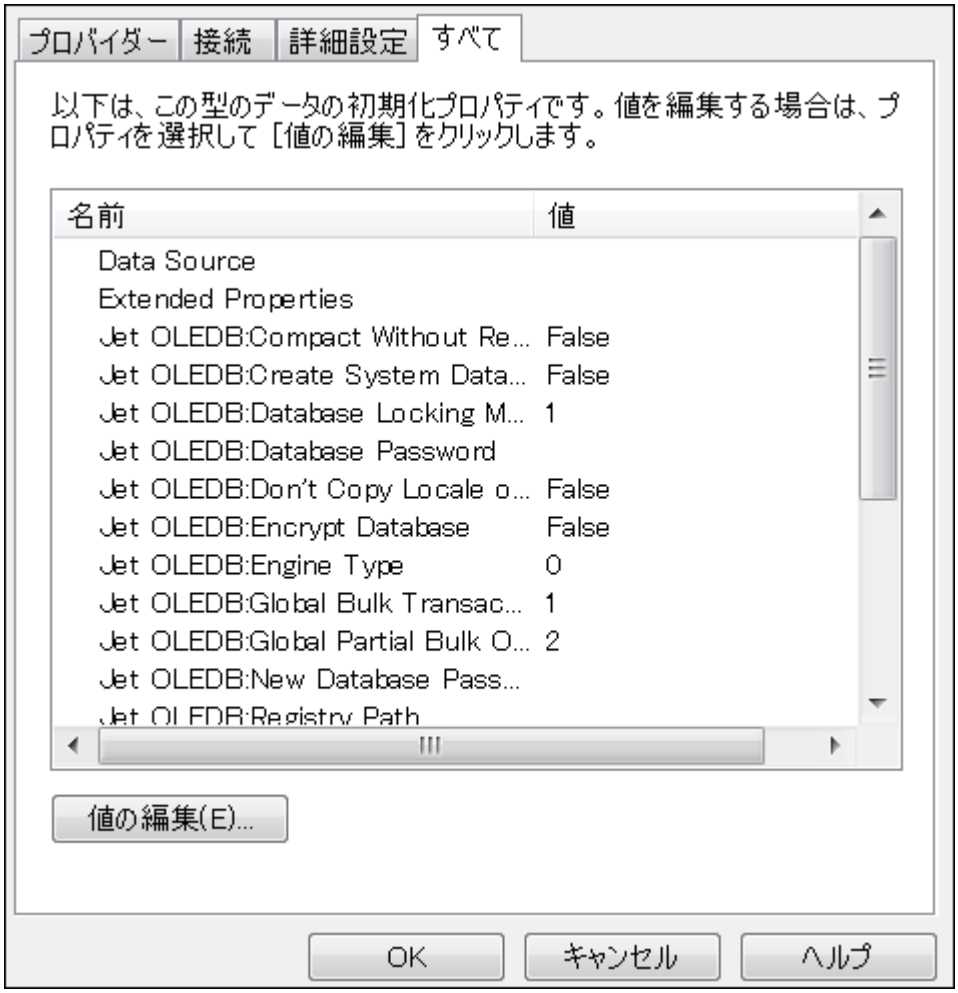


データ リンク プロパティ ダイアログ ボックス

プロパティ	メモ
統合セキュリティ	プロバイダータブで **SQL Server Native Client** がデータプロバイダーに選択された場合は、このプロパティを、空白文字に設定します。
セキュリティ情報を保持	このプロパティを**True** に設定する。

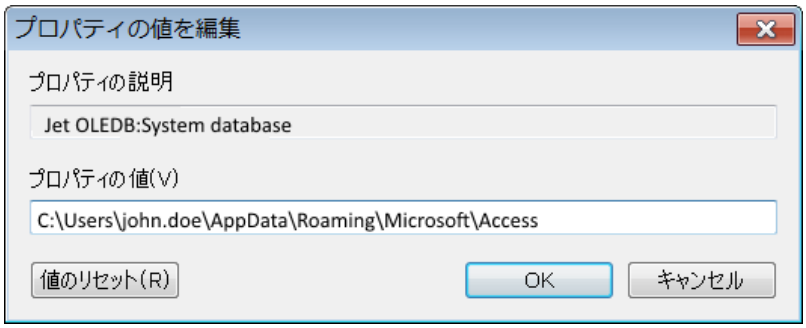
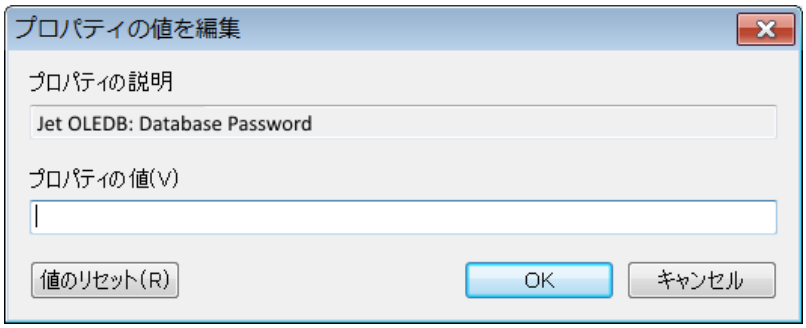
10.2.3.3 Microsoft Access データ リンク プロパティの設定

Microsoft Access データベースにADO を介して接続する場合、次のデータリンク プロパティダイアログボックスの全ての タブ内の次の接続プロパティを設定する必要があります。



データリンクプロパティダイアログボックス

プロパティ	メモ
データソース	このプロパティは、Microsoft Access データベースファイルへのパスを保管します。データベースへの接続問題を回避するために、UNC（Universal Naming Convention 汎用名前付け規則）パスフォーマットの使用が奨励されます。例: `\\anyserver\share$\filepath`
Jet OLEDB: システムデータベース	このプロパティはワークグループ情報ファイルへのパスを保管します。データベースに接続する前に、このプロパティを明示的に設定する必要がある可能性があります。 "ワークグループ情報ファイル" エラーが理由で接続できない場合、ワークグループ情報ファイル（**System.MDW**）をユーザープロファイルに適用できるように設定します（手順は http://support.microsoft.com/kb/305542 を参照してください）、プロパティの値を **System.MDW** ファイルのパスに設定します。

プロパティ	メモ
	
Jet OLEDB: データベース パスワード	データベースがパスワードで保護されている場合は、このプロパティの値をデータベースのパスワードに設定します。 

10.2.4 ADO.NET 接続のセットアップ

ADO.NET は、データベースからのデータを含むデータと作業するためにデザインされた Microsoft .NET フレームワーク ライブラリのセットです。StyleVision から、データベースに接続するには、ADO.NET、Microsoft .NET フレームワーク 4、または、以降を使用する必要があります。下に表示されているとおり、 .NET プロバイダーを選択して、接続文字列を提供し、データベースに ADO.NET を介して接続します。

.NET データプロバイダーは、コマンドを実行する特定の型のデータソースへの接続を可能にするクラスのコネクションです（例えば、SQL Server または Oracle データベースなど）。すなわち、ADO.NET とは、StyleVision などのアプリケーションがデータベースプロバイダーを介してデータベースと作動することを意味します。各 データプロバイダーは特定の型のデータソースと作動するように最適化されています。.NET プロバイダーには以下の2つの種類があります：

1. デフォルトとして Microsoft .NET フレームワークを使用して提供されます。
2. 主要なデータベースベンダーを使用して .NET フレームワークの拡張として提供されます。このような ADO.NET プロバイダーは、個別人ストールされている必要があり、通常それぞれのデータベースベンダーのウェブサイトからダウンロードすることができます。

メモ 特定の ADO.NET プロバイダーは、サポートされていない、または、サポートが制約されています。ADO.NET サポートに関するメモを参照してください。

ADO.NET 接続の作成:

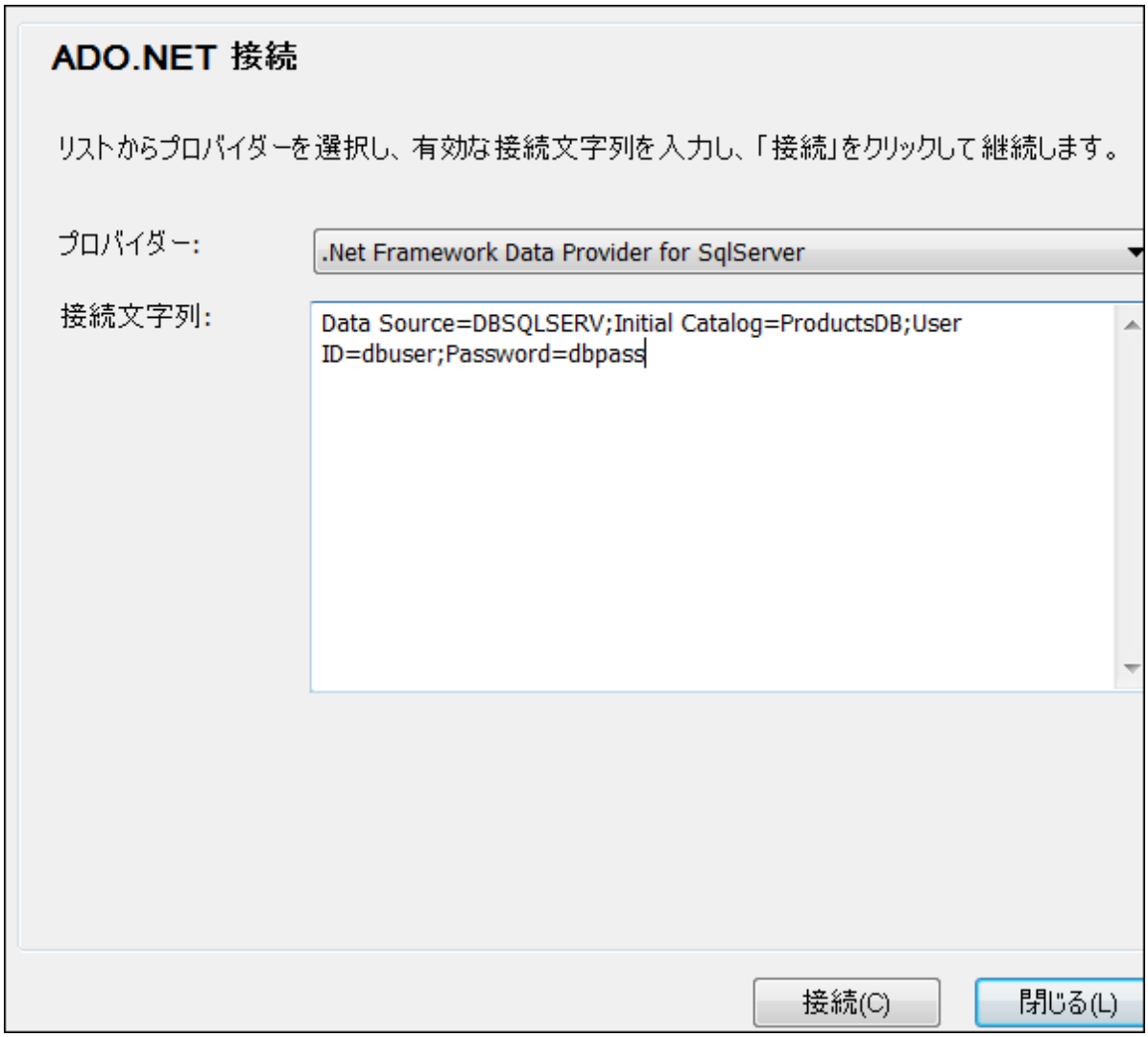
1. データベース接続ウィザードを開始します。
2. 「ADO.NET 接続」をクリックします。

3. .NET データ プロバイダーをリストから選択します。

> .NET フレームワークと共に使用することのできるプロバイダーのリストはデフォルトでプロバイダーのリストに表示されます。ベンダーに特有の.NET データプロバイダーは、使用中のシステムに既にインストールされている場合のみ使用することができます。.NET データ プロバイダーは、データベースベンダーにより提供される、.msi、または、.exe ファイルを実行して、GAC（Global Assembly Cache）内にインストールされる必要があります。

4. データベース接続文字列を入力します。接続文字列は、データベース接続情報をセミコロンで区切られた接続パラメーターのキー値ペアとして定義します。例えば、**`Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;password=dbpass`** などの文字列は、サーバー `DBSQLSERV` 上で、ユーザー名 `dbuser` とパスワード `dbpass` を使用して、SQL Server データベース `ProductsDB` に接続します。接続文字列を、キー値ペアを直接「接続文字列」ダイアログボックスに入力することにより作成することができます。他の接続オプションは Visual Studio で作成することができます（次を参照: Visual Studio 内で接続文字列を作成する）。

> 接続文字列の構文は、プロバイダーリストより選択されたプロバイダーにより異なります。例えば、次を参照してください: サンプル ADO.NET 接続文字列。



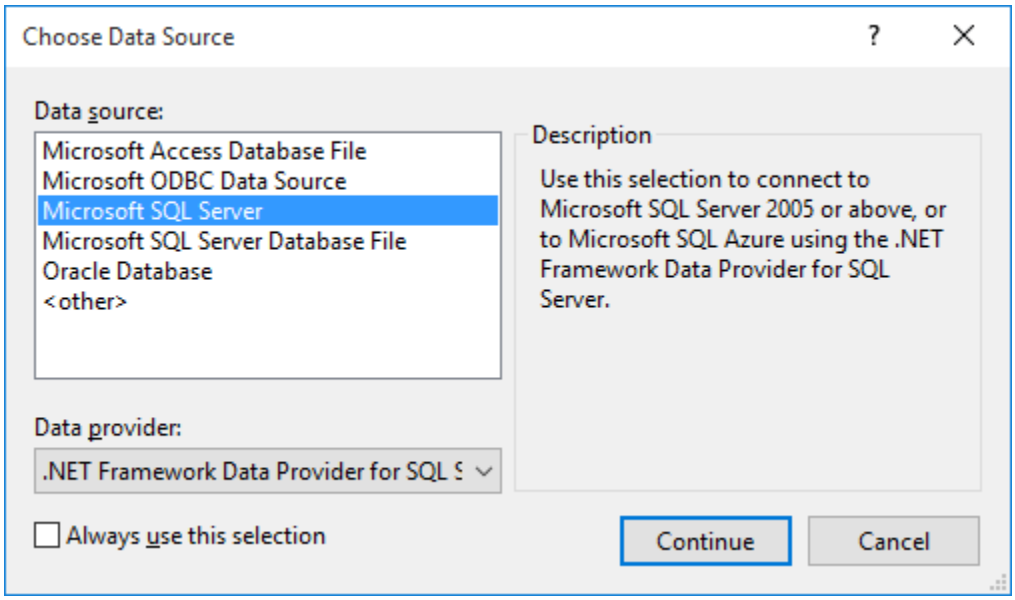
5. 「接続」をクリックします。

10.2.4.1 Visual Studio 内で接続文字列を作成する

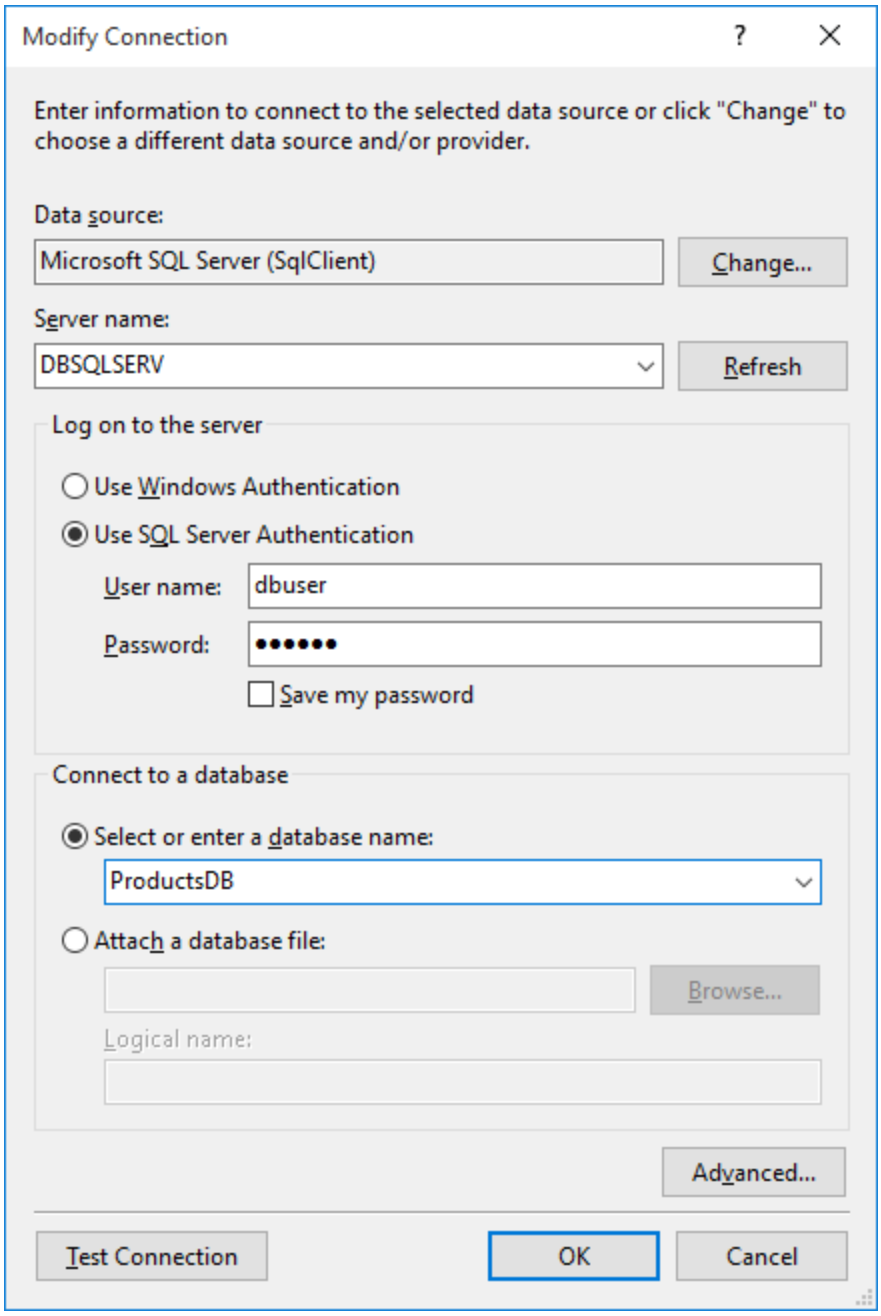
ADO.NET を使用してデータソースに接続するには、有効な接続文字列が必要です。次の手順は、Visual Studio から接続文字列を作成する方法を説明しています。

Visual Studio 内で接続文字列を作成する:

1. 「ツール」メニューから、「データベースへの接続」をクリックします。
2. リストからデータソースを選択します(この例の場合、Microsoft SQL Server)。データプロバイダーは、選択に従い自動的に記入されます。



3. 「接続」をクリックします。



4. データベースへのサーバーホスト名、ユーザー名、およびパスワードを入力します。この例の場合、サーバー `DBSQLSERV` 上のデータベース `ProductsDB` にSQL Server 認証を使用して接続します。

5. 「OK」をクリックします。

データベースへの接続に成功した場合、サーバーエクスプローラーウィンドウに表示されます。サーバーエクスプローラーウィンドウを、メニューコマンド「**表示 | サーバーエクスプローラー**」を使用して表示することができます。データベース接続文字列を取得するには、サーバーエクスプローラーウィンドウ内の接続を右クリックし、「**プロパティ**」を選択します。接続文字列が、Visual Studio プロパティウィンドウに表示されます。StyleVision の「接続文字列」ボックスに文字列を貼り付ける前に、アスタリスク（＊）文字を実際のパスワードと置き換える必要があることに注意してください。

10.2.4.2 サンプル ADO.NET 接続文字列

ADO.NET 接続をセットアップするには、データベース接続ダイアログボックスから ADO.NET プロバイダーを選択し、接続文字列を入力します。（次も参照してください ADO.NET 接続のセットアップ）。異なるデータベースのサンプル ADO.NET 接続文字列は .NET プロバイダーの下にリストされています。

Teradata のための .NET Data

このプロバイダーは、次の Teradata Web サイトからダウンロードすることができます (http://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata)。サンプル接続は以下のとおりです:

```
Data Source=ServerAddress;User Id=user;Password=password;
```

IBM i のための .NET Framework データプロバイダー

このプロバイダーは、*IBM i Access Client Solutions - Windows Application Package* の一部としてインストールされます。サンプル接続文字列は以下のとおりです:

```
DataSource=ServerAddress;UserID=user;Password=password;DataCompression=True;
```

詳細に関しては、インストールパッケージ内に含まれる「.NET プロバイダー技術レファレンス ヘルプファイル」を参照してください。

MySQL のための .NET Framework データプロバイダー

このプロバイダーを MySQL ウェブサイトからダウンロードすることができます（https://dev.mysql.com/downloads/connector/net/）。サンプル接続文字列は以下のとおりです:

```
Server=127.0.0.1;Uid=root;Pwd=12345;Database=test;
```

次も参照してください: https://dev.mysql.com/doc/connector-net/en/connector-net-programming-connecting-connection-string.html

SQL Server のための .NET Framework データプロバイダー

サンプル接続文字列は以下のとおりです:

```
Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;Password=dbpass
```

次も参照してください: https://msdn.microsoft.com/en-us/library/ms254500(v=vs.110).aspx

.NET Framework 4.0 のためのIBM DB2 データプロバイダー10.1.2

```
Database=PRODUCTS;UID=user;Password=password;Server=localhost:50000;
```

メモ このプロバイダーは、通常 IBM DB2 Data Server Client パッケージと共にインストールされます。IBM DB2 Data Server Client パッケージをインストール後 ADO.NET プロバイダーのリスト内でプロバイダーが不足している場合、次の技術メモを参照してください: http://www-01.ibm.com/support/docview.wss?uid=swg21429586。

次も参照してください:
http://www.ibm.com/support/knowledgecenter/en/SSEPGG_10.1.0/com.ibm.swg.im.dbclient.adonet.ref.doc/doc/DB2ConnectionClassConnectionStringProperty.html

.NET (ODP.NET) のための Oracle データプロバイダー

ODP.NET プロバイダーを含むインストールパッケージを Oracle ウェブサイトからダウンロードすることができます (次を参照してください: http://www.oracle.com/technetwork/topics/dotnet/downloads/index.html). サンプル接続文字列は以下のとおりです:

```
Data Source=DSORCL;User Id=user;Password=password;
```

Oracle への接続(ODBC)内で説明されているように **tnsnames.ora** ファイル内で定義された Oracle サービス名をさす DSORCL がデータソース名の場合。

tnsnames.ora ファイル内のサービス名を構成せず接続する場合は、以下のような文字列を使用してください:

```
Data Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host)(PORT=port)))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=MyOracleSID)));User
Id=user;Password=password;
```

次も参照してください: https://docs.oracle.com/cd/B28359_01/win.111/b28375/featConnecting.htm

10.2.4.3 ADO.NET サポートに関するメモ

下のテーブルは StyleVision 内で現在サポートされていない、または、制限のあるサポートのある既知の ADO.NET データベースドライバーがリストされています。

データベース	ドライバー	サポートノート
全てのデータベース	ODBC のための Net Framework **Data Provider**	制限つきサポート。Microsoft Access 接続の際に既知の問題があります。ODBC への直接の接続を代わりに使用することが奨励されます。
	OleDb のための .Net Framework **Data Provider**	制限つきサポート。Microsoft Access 接続の際に既知の問題があります。ADO への直接の接続を代わりに使用することが奨励されます。
Firebird	**Firebird ADO.NET Data Provider**	制限つきサポート。ODBC または JDBC を代わりに使

データベース	ドライバー	サポートノート
		用することが奨励されます。
Informix	.NET Framework 4.0 のための IBM **Informix Data Provider**	サポートされていません。**DB2 Data Server Provider** を代わりに使用してください。
IBM DB2 for i (iSeries)	i5/OS のための .Net Framework **Data Provider**	サポートされていません。*IBM i Access Client Solutions - Windows Application Package* の一部として提供されている **.Net Framework Data Provider for IBM i** を代わりに使用してください。
Oracle	Oracle のための .Net Framework **Data Provider**	制限つきサポート。.NET Framework と共にこのドライバは提供されていますが、Microsoft により使用を奨励されていません。
PostgreSQL	-	このベンダーのための ADO.NET ドライバはサポートされていません。ネイティブな接続を代わりに使用してください。
Sybase	-	このベンダーのための ADO.NET ドライバはサポートされていません。

10.2.5 ODBC 接続の設定

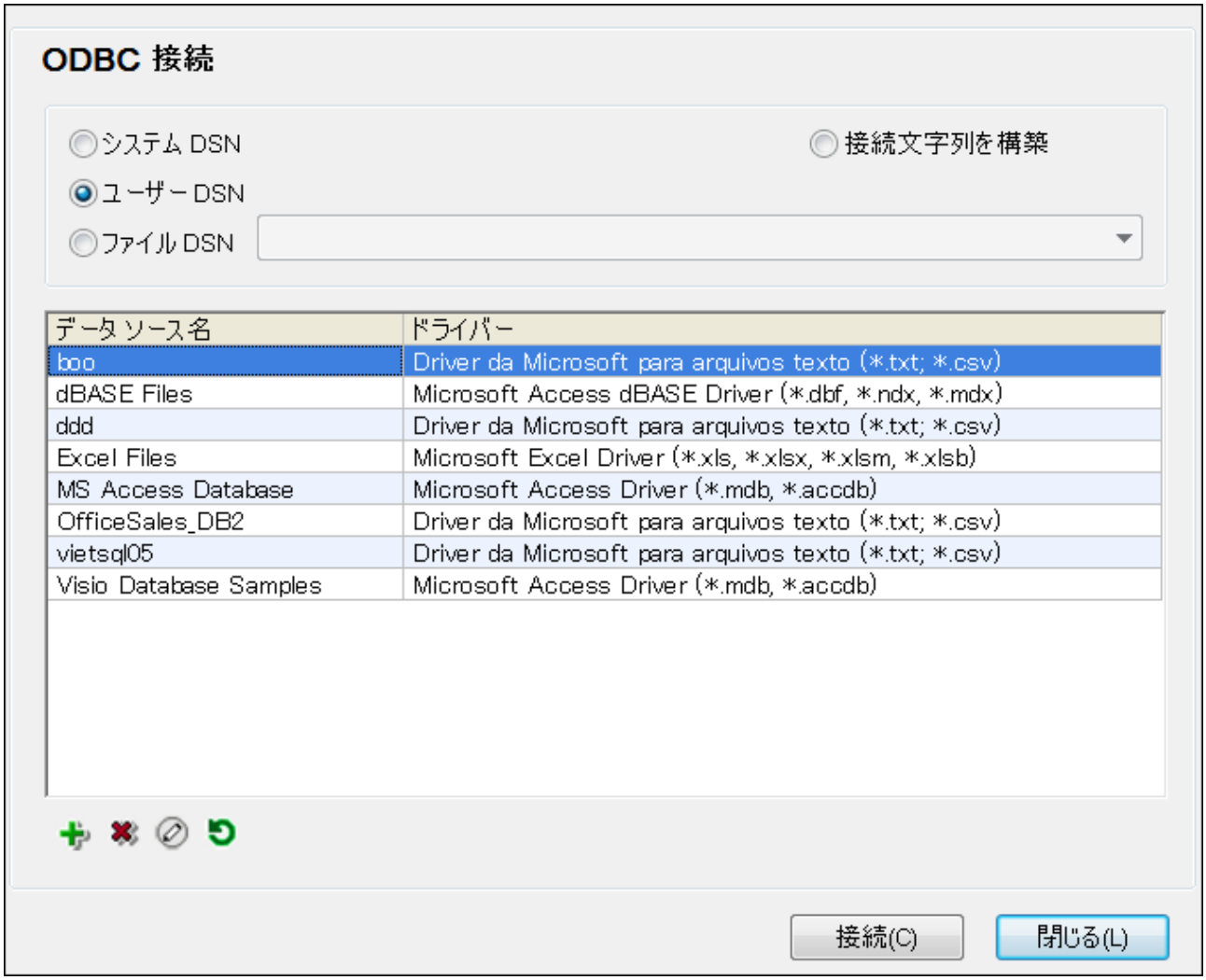
ODBC (Open Database Connectivity) は、一般に使用されているデータアクセス技術で、StyleVision からデータベースに接続することができます。データベースにアクセスするための主要な手段として、またはネイティブな OLE DB- や JDBC-による接続の代替として使用することができます。

ODBC を使用してデータベースに接続するには、最初に ODBC データソース名(DSN) を作成する必要があります。このステップは、他のユーザーまたはオペレーティングシステムにより DSN が既に作成されている場合必要ありません。DSN は、統一された、StyleVision を含むオペレーティングシステム上の ODBC に対応したクライアントアプリのデータベース接続方法を表します。DSN には以下の種類があります:

- ファイル DSN
- ユーザー DSN
- ファイル DSN

オペレーティングシステムで特権を持つすべてのユーザーはシステム データ ソースにアクセスすることができます。作成したユーザーのみがユーザーデータ ソースにアクセスすることができます。*ファイル DSN* を作成した場合、データソースは .dsn 拡張子を持つファイルとして作成され、データソースにより使用されているドライバーがインストールされていることが前提で、他のユーザーと共有することができます。

ODBC 接続ダイアログボックス内の [**ODBC 接続**] をクリックすると、使用中のコンピューターで既に使用することのできる DSN はデータベース接続ダイアログボックスに表示されています。



ODBC 接続ダイアログボックス

データベースにより必要とされるDSN が無い場合、StyleVision データベース接続ウィザードが作成の手助けをします。しかし、W indows オペレーティングシステムに直接作成することもできます。いずれにせよ、続行する前に、使用中のデータベースにODBC ドライバーがオペレーティングシステムのODBC ドライバーリストに適用できることを確認してください。（ 使用可能な ODBC ドライバーの表示を参照 ）。

新規のDSNを使用しての接続:

1. データベース接続ウィザードを開始します。
2. データベース接続ダイアログボックスの［ODBC 接続］をクリックします。
3. システムDSN を作成するには、オペレーティングシステムの管理者権限が必要です。

   > また、StyleVision が管理者として作動している必要があります。

4. 「追加」 をクリックします。

5. ドライバーを選択して、(作成するDSN の種類に応じて) [**ユーザーDSN**] または[**システムDSN**] をクリックします。使用中のデータベースに適用することのできるドライバーがリストされていない場合、データベースベンダーからダウンロードして、インストールしてください。(次を参照: データベースドライバー概要)。
6. ポップアップするダイアログボックスに特定の接続情報を入力してセットアップを完了します。

接続を成功させるには、データベースサーバーのホスト名(またはIP アドレス)、とデータベースのユーザー名およびパスワードを提供する必要があります。他の接続パラメーターがある可能性がありますが、これらのパラメーターはデータベースプロバイダーにより異なります。各接続方法に固有のパラメーターについての詳細は、ドライバープロバイダーのドキュメントを参照してください。一度作成されると、DSN はデータソース名のリストで使用することができます。これによりデータベース接続の詳細をデータベースの接続を希望する際いつでも使用することができます。ユーザーDSN はユーザーDSN のリストに追加されますが、システムDSN はシステムDSN のリストに追加されます。

既存のDSNを使用しての接続:

1. データベース接続ウィザードを開始します。
2. [**ODBC 接続**] をクリックします。
3. 既存のデータソース(ユーザーDSN、システムDSN、ファイルDSN) から種類を選択します
4. 既存 DSN レコードをクリックして、「**接続**」をクリックします。

既存の.dsn ファイルをベースにして接続文字列を構築する:

1. データベース接続ウィザードを開始します。
2. [**ODBC 接続**] をクリックします。
3. [**接続文字列の構築**] を選択して [構築] をクリックします。
4. ファイルDSN を使用して、接続文字列を構築する場合、[**ファイルデータソース**] タブをクリックします。それ以外の場合は、[**マシンデータソース**] タブをクリックします。(システムDSN とユーザーDSN は"マシン" データソースとして知られています。)
5. 必要とされる.dsn ファイルを選択して、「OK」をクリックします。

用意された接続文字列を使用して接続する:

1. データベース接続ウィザードを開始します。
2. [**ODBC 接続**] をクリックします。
3. [**接続文字列の構築**] を選択しています。
4. 接続文字列を与えられたボックスに貼り付け、「**接続**」をクリックします。

10.2.5.1 使用可能な ODBC ドライバーの表示

ODBC データソース管理で使用中のオペレーティングシステムで使用可能な ODBC ドライバーを確認することができます。Windows コントロールパネルの**管理ツール**から ODBC データソース管理 (**Odbcad32.exe**) にアクセスすることができます。64-ビット版オペレーティングシステムには、2 種類のバージョンの実行可能ファイルがあります:

- **C:\Windows\SysWoW64** ディレクトリにある、**Odbcad32.exe** ファイルの32-ビット版バージョン(**C:** が使用中のシステムドライブとして前提されています)。
- **C:\Windows\System32** ディレクトリにある、**Odbcad32.exe** ファイルの64-ビット版バージョン

インストールされている32-ビット版データベースドライバーは、ODBC データソース管理の32-ビット版バージョンで、64-ビット版バージョン内の64-ビット版ドライバーで確認することができます。ですから、対応するバージョンのODBC データソース管理でデータベースドライバーを確認してください。

ODBC データソース管理者

ターゲットデータベースのドライバーがリストに存在しない場合、または、代替ドライバーを追加する場合、データベースのベンダーからダウンロードしてください（データベースドライバー概要 を参照）。ODBC ドライバーがシステムで使用可能になると、ODBC 接続を作成することができます（ODBC 接続のセットアップを参照）。

10.2.6 JDBC 接続の設定

JDBC（Java Database Connectivity）は、Oracle の Java ソフトウェアプラットフォームの一部であるデータベースアクセスインターフェイスです。JDBC 接続は ODBC 接続に比べ、通常より多くリソースを集中的に使用する接続ですが、ODBC で使用することのできない機能を使用することができます。ODBC コネクタにより使用できないデータベース機能を使用する場合 JDBC 接続を使用することが奨励されています。

前提条件

- JRE（Java Runtime Environment）または Java Development Kit（JDK）がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。StyleVision は次のロケーションから Java Virtual Machine（JVM）へのパスを次の順序で決定します: a）アプリケーション オプション内で設定したカスタム JVM パス b）Windows レジストリ内で見つかった JVM パス。c）`JAVA_HOME` 環境変数。
- StyleVision（32-ビット、64-ビット）のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- データベースベンダーからの JDBC ドライバーがインストールされている必要があります。これは使用可能でありデータベースによりサポートされている場合、データベースクライアントインストールの一部として、または、JDBC ライブラリ（.jar ファイル）として個別にダウンロードされている JDBC ドライバーの可能性があります。データベース接続のサンプルも参照してください。
- Oracle データベースに接続する場合は、一部の Oracle ドライバーは特定の JRE バージョンに固有のため、追加コンポーネントと構成が必要な点に注意してください。Oracle 製品のドキュメント（例えば、「JDBC 開発者ガイドおよびリファレンス」）には、各 JDBC ドライバーの構成の手続きの詳細が説明されています。

(C) 2015-2021 Altova GmbH

- CLASSPATH 環境変数は、(.jar ファイルの1つまたは複数の) 使用中のWindows オペレーティングシステムのJDBC ドライバーのパスを含む必要があります。データベースクライアントをインストールする際に、インストーラーはこの変数を自動的に構成する可能性があります (CLASSPATH の構成を参照)。

Windows 資格情報を使用して JDBC を介して SQL Server に接続する方法

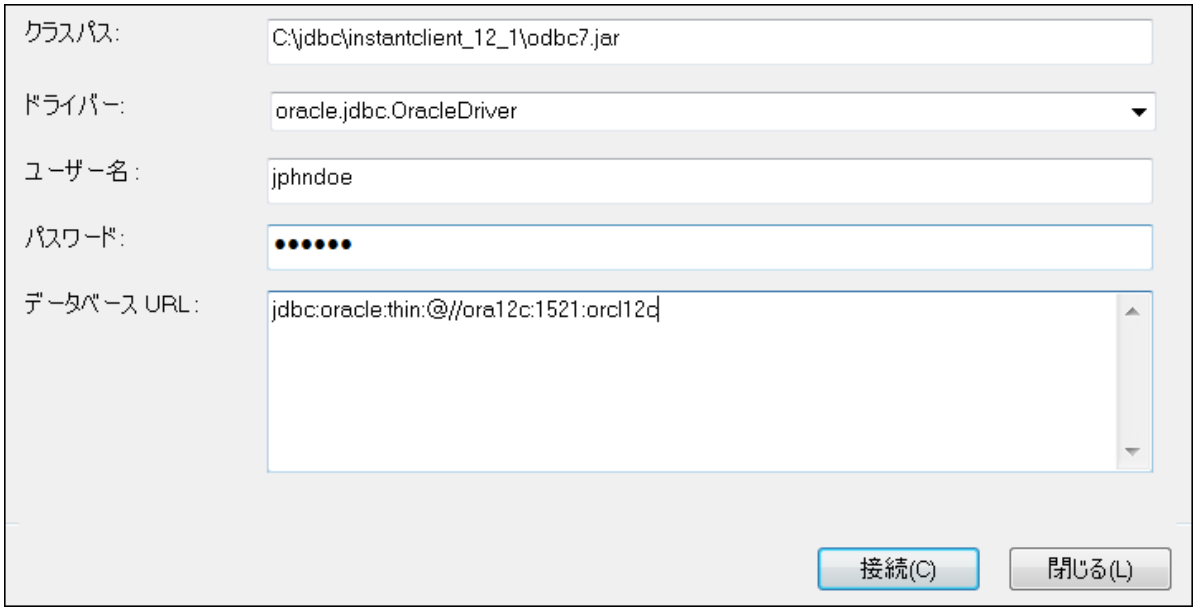
Windows 資格情報 (統合されたセキュリティ) を使用して JDBC を介して SQL Serverに接続する場合、以下に注意してください:

- JDBC ドライバーパッケージに含まれている **sqljdbc_auth.dll** ファイルはシステムPATH 環境変数上のディレクトリにコピーされる必要があります。このようなファイルは2個存在します。1つはx86 プラットフォームのために、もう1つはx64 プラットフォームのために存在します。JDK プラットフォームに対応するPATH が追加されていることを確認してください。
- JDBC 接続文字列はプロパティ **`integratedSecurity=true`** を含む必要があります。このプロパティを異なる場所から追加することができます:
  - データベース接続ウィザードから追加することができます。以下を参照してください。
  - データベースコンポーネント設定から追加することができます。
  - 適応可能な場合、生成済みのJava コード内のデータベース接続文字列の編集をすることにより追加することができます。

詳細に関しては *SQL Server のための Microsoft JDBC ドライバードキュメント*を参照してください。
https://docs.microsoft.com/en-us/sql/connect/jdbc/building-the-connection-url。

JDBC 接続のセットアップ方法

1. データベース接続ウィザードを開始します。
2. 「**JDBC 接続**」をクリックします。
3. 任意で、セミコロンで区切られた.jar ファイルパスのリストを「クラスパス」テキストボックスに入力することもできます。ここで入力される.jar ライブラリは、CLASSPATH 環境変数内で既に定義されているものに加え、環境にロードされます。「クラスパス」テキストボックスの編集が完了すると、ソース.jar ライブラリ内で検索されるJDBC ドライバーは、「ドライバー」リストに自動的に追加されます (次のステップを参照してください)。



4. 「ドライバー」の横でJDBC ドライバーをリストから選択、または、Java クラス名を入力します。CLASSPATH 環境変数により構成されたJDBC ドライバーおよび、「クラスパス」テキストボックス内で検出されたドライバーが含まれます (次を参照: CLASSPATH の構成)。

CLASSPATH 変数内で定義されているJDBC ドライバーのパスを使用することができます。また、データベース接続ダイアログボックスに直接入力された.jar パスはJava Virtual Machine（JVM）によりサポートされています。JVM は接続を構築するためのドライバーを決定します。JVM にロードされるJava クラスを確認し、JDBC ドライバーの競合、および、データベースに接続する際に発生する予期されないエラーを回避するようにしてください。

5. 対応するボックスに、ユーザー名とパスワードを入力します。
6. データベースURL テキスト・ボックス内に、JDBC 接続 URL（文字列）をデータベースの型のとおり入力します。次のテーブルは一般のデータベースの型のためのJDBC 接続 URL（文字列）の構文について説明しています。

データベース	JDBC 接続 URL
Firebird	jdbc:firebirdsql://**<host>**[:**<port>**]/**<database path or alias>**
IBM DB2	jdbc:db2://**hostName**:**port**/**databaseName**
IBM DB2 for i	jdbc:as400://**[host]**
IBM Informix	jdbc:informix-sqli://**hostName**:**port**/**databaseName**:INFORMIXSERVER=**myserver**
MariaDB	jdbc:mariadb://**hostName**:**port**/**databaseName**
Microsoft SQL Server	jdbc:sqlserver://**hostName**:**port**;**databaseName**=name
MySQL	jdbc:mysql://**hostName**:**port**/**databaseName**
Oracle	jdbc:oracle:thin:@**hostName**:**port**:**SID** jdbc:oracle:thin:@//**hostName**:**port**/**service**
Oracle XML DB	jdbc:oracle:oci:@//**hostName**:**port**:**service**
PostgreSQL	jdbc:postgresql://**hostName**:**port**/**databaseName**
Progress OpenEdge	jdbc:datadirect:openedge://**host**:**port**;databaseName=**db_name**
Sybase	jdbc:sybase:Tds:**hostName**:**port**/**databaseName**
Teradata	jdbc:teradata://**databaseServerName**

メモ 上記のフォーマットに対する構文のバリエーションも使用することができます（例えば、データベースURL はポートを除外、またはデータベースのユーザー名とパスワードを含む可能性があります）。詳細はデータベースのベンダーのドキュメントを参照してください。

7. 「接続」をクリックします。

10.2.6.1 CLASSPATH の構成

CLASSPATH 環境変数はクラスと、オペレーティングシステム上の他のソースファイルを検索するためJava Runtime Environment（JRE）またはJava Development Kit（JDK）により使用されます。JDBC を介してデータベースに接続すると、この変数はオペレー

ティングシステム上のJDBC ドライバーへのパス、または一部の場合、使用中のデータベースの種類に固有の追加ライブラリファイルへのパスを含むように構成されなければなりません。

下のテーブルは、CLASSPATH 変数に通常含まれる必要のあるサンプルファイルパスをリストしています。重要なことは、システム上の情報をシステム上のJDBC ドライバーのロケーション、JDBC ドライバー名、およびオペレーティングシステム上に存在するJRE/JDK バージョンをベースにしたこの情報を調整する必要があります。接続問題を避けるために、使用中のオペレーティングシステムにインストールされているJDBC ドライバに適用されるインストールの指示、インストール前とインストール後の構成ステップを確認してください。

データベース	サンプルCLASSPATH エントリ
Firebird	C:\Program Files\Firebird\Jaybird-2.2.8-JDK_1.8\jaybird-full-2.2.8.jar
IBM DB2	C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc.jar;C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc_license_cu.jar;
IBM DB2 for i	C:\jt400\jt400.jar;
IBM Informix	C:\Informix_JDBC_Driver\lib\ifxjdbc.jar;
Microsoft SQL Server	C:\Program Files\Microsoft JDBC Driver 4.0 for SQL Server\sqljdbc_4.0\enu\sqljdbc.jar
MariaDB	**<installation directory>**\mariadb-java-client-2.2.0.jar
MySQL	**<installation directory>**\mysql-connector-java-*version*-bin.jar;
Oracle	**ORACLE_HOME**\jdbc\lib\ojdbc6.jar;
Oracle (with XML DB)	**ORACLE_HOME**\jdbc\lib\ojdbc6.jar;**ORACLE_HOME**\LIB\xmlparserv2.jar;**ORACLE_HOME**\RDBMS\jlib\xdb.jar;
PostgreSQL	**<installation directory>**\postgresql.jar
Progress OpenEdge	%DLC%\java\openedge.jar;%DLC%\java\pool.jar; メモProgress OpenEdge SDK がマシンにインストールされていると想定します。%DLC% は OpenEdge がインストールされているディレクトリです。
Sybase	C:\sybase\jConnect-7_0\classes\jconn4.jar
Teradata	**<installation directory>**\tdgssconfig.jar;**<installation directory>**\terajdbc4.jar

- CLASSPATH 変数の変更は、使用中のコンピュータのJava アプリケーションの振る舞いに影響する可能性があります。続行する前に生じえる影響を把握するために、Java ドキュメントを参照してください。
- 環境変数は、ユーザーまたはシステムであることができます。システム環境変数を変更するには、オペレーティングシステムに対する管理者権限が必要です。
- 環境変数を変更後、設定の変更を有効にするには、プログラムを再起動してください。または、オペレーティングシステムをログオフまたは再起動してください。

Windows 7 でCLASSPATH を構成する:

1. [スタート] メニューを開始して、[コンピューター] を右クリックします。

2. [プロパティ] をクリックします。
3. [システムの詳細設定] をクリックします。
4. [詳細設定] タブから [環境変数] をクリックします。
5. CLASSPATH 変数を、ユーザーまたはシステム環境変数からロケートして、編集をクリックします。CLASSPATH 変数が存在しない場合は、[新規] をクリックして作成します。
6. オペレーティングシステム内で JDBC ドライバーがある場所にパスを含むように変数の値を編集します。JDBC ドライバーパスを他の既に CLASSPATH 変数内にあるパスから区別するためにセミコロン(;)で区切ります。

Windows 10 で CLASSPATH を構成する:

1. Windows キーを押して、環境編集を入力します。
2. [システム環境変数の編集] の提案をクリックします。
3. [環境変数] をクリックします。
4. CLASSPATH 変数を、ユーザーまたはシステム環境変数からロケートして、編集をクリックします。CLASSPATH 変数が存在しない場合は、[新規] をクリックして作成します。
5. オペレーティングシステム内で JDBC ドライバーがある場所にパスを含むように変数の値を編集します。JDBC ドライバーパスを他の既に CLASSPATH 変数内にあるパスから区別するためにセミコロン(;)で区切ります。

10.2.7 PostgreSQL 接続のセットアップ

PostgreSQL データベースへの接続は、ネイティブな接続として、または、ODBC、JDBC、および他のドライバーを使用して確立することができます。ネイティブな接続の利点は、使用中のシステムにインストールする場合、ドライバーを必要としないことです。また、Linux、または、macOS サーバー上での実行のためにファイルをデプロイする場合、ターゲットサーバー上にドライバーをインストールする必要はありません

ネイティブでは内ドライバーを使用しての接続を作成する場合は、以下のトピックを参照してください:

- JDBC 接続の設定
- PostgreSQL (ODBC) への接続

それ以外の場合、PostgreSQL へのネイティブの接続をセットアップする場合、以下の手順に従ってください。継続するには、以下の前提条件が必要です: ホスト名、ポート、データベース名、ユーザー名、およびパスワード。

ネイティブ PostgreSQL 接続のセットアップ:

1. データベース接続ウィザードを開始します.
2. 「PostgreSQL 接続」をクリックします。
3. 対応するボックスに、ホスト(PostgreSQL が同じマシンで作動している場合、*ローカルホスト*)、ポート(通常、5432 ですが、これは任意です)、データベース名、ユーザー名とパスワードを入力します。

4. 「**接続**」をクリックします。

PostgreSQL データベースサーバーが異なるマシンに存在する場合、次の点に注意してください：

- PostgreSQL データベースサーバーは、クライアントからの接続を受け入れるように構成されなければなりません。具体的には、**pg_hba.conf** ファイルがローカルではない接続を許可するように構成される必要があります。第2に、**postgresql.conf** ファイルは、特定された IP アドレスとポートをリッスンするように構成される必要があります。詳細に関しては、PostgreSQL ドキュメントを参照してください（https://www.postgresql.org/docs/9.5/static/client-authentication-problems.html）。
- サーバーマシンがファイアウォールを越えてデザインされたポート（通常、5432）上で接続を受け入れるように構成されている必要があります。例えば、「**コントロールパネル** > **Windows ファイアウォール** > **詳細設定** > **受信の規則**」で設定すると、Windows 上で作動するデータベースサーバーでは、ファイアーウォールを越えてポート 5432 に接続を許可するルールを作成する必要がない場合があります。

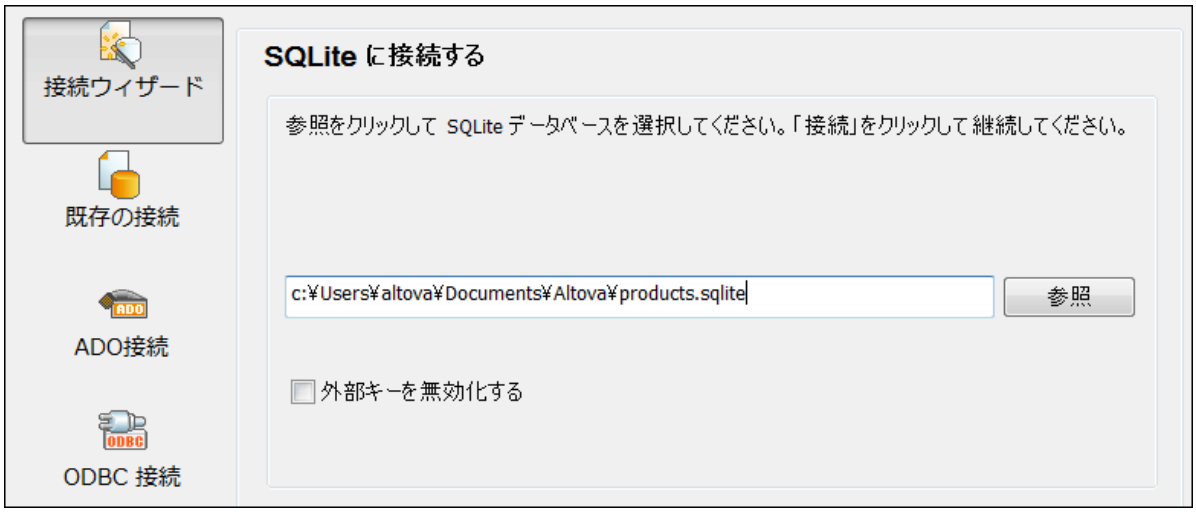
10.2.8 SQLite 接続の設定

SQLite (http://www.sqlite.org) は、ポータブル性とインストールの簡便性が重要な場合理想的な構成のファイルベースの、内蔵のデータ型です。Since SQLite データベースは、StyleVision によりネイティブにサポートされるため、接続のためにドライバーをインストールする必要はありません。

10.2.8.1 既存の SQLite データベースへの接続

既存のSQLite データベースに接続する方法:

1. データベース接続ウィザードを実行します（ データベース接続ウィザードの開始を参照）。
2. **SQLite** を選択して、「次へ」をクリックします。
3. SQLite データベースファイルを参照して、または、（相対または絶対）パスを入力します。「**接続**」ボタンは、データベースファイルへのパスを入力すると有効化されます。



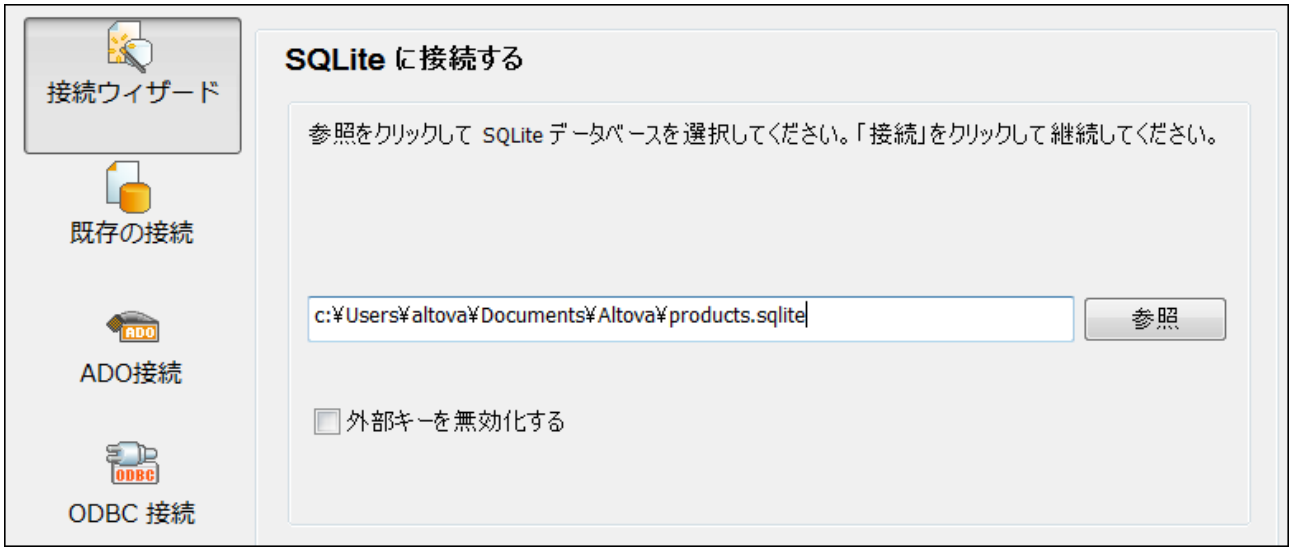
4. 「**外部キーの無効化**」チェックボックスを任意で選択します。次も参照してください: 外部キーの制約。
5. 「**接続**」をクリックします。

10.2.8.2 外部キーの制約

StyleVision, から既存のSQLite データベースに接続する場合、 デフォルトで、外部キーの制約 が有効化されます。外部キーの制約はデータベース内のデータの整合性を保持する手助けを行います。例えば、外部キーが有効化されていると、他のテーブル内に依存関係がある場合、テーブルからレコードを削除することができません。

特定の場合、一時的にこの振る舞いをオーバーライドし、外部キーを無効化する必要がある場合があります。例えば、データ検証エラー無しにデータの列を更新、または、挿入する場合など。SQLite データベースに接続する前に、明示的に外部キーを無効化するには、データベース接続ウィザードの「**外部キーの無効化**」オプションを選択します。

(C) 2015-2021 Altova GmbH



「SQLite への接続」ウィザードページ

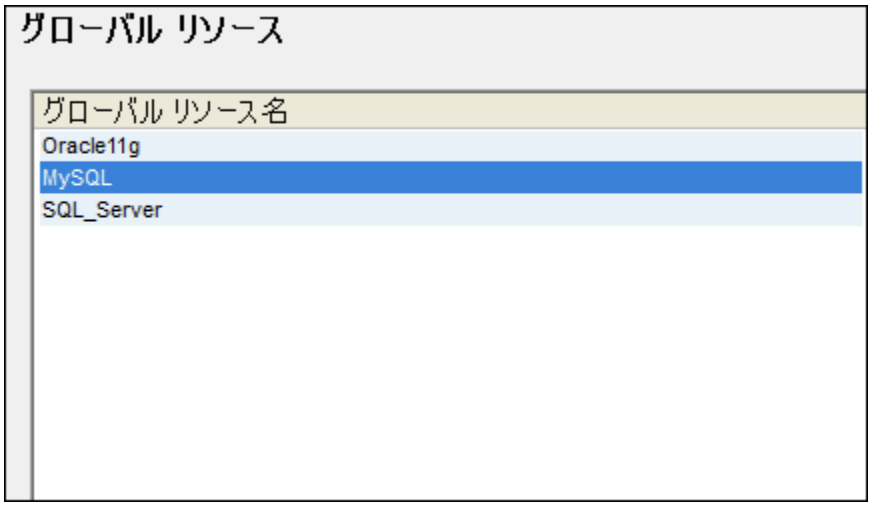
外部キーが無効化されていると、検証チェックのためチェックすることのできなかったデータに対して操作をおこなうことができます。同時に、データベース内に不正確なデータを入れてしまう危険性、または、「無所属の」行を作成する可能性あります。(「無所属の」行の例は、「address」テーブルにリンクされていない「person」テーブル内のアドレスなどが挙げられます。これは、個人名が削除されているにもかかわらず、この個人に関連した住所が削除されていない場合などにこのような行が発生します。)

10.2.9　グローバルリソースからの接続の使用

Altova グローバルリソースはファイル、フォルダー、およびデータベースへのポータブルなレファレンスです。以前にデータベースをグローバルリソースとして既に構成している場合、接続をいつでも再使用することができます(異なる Altova アプリケーション間でも使用可能)。

グローバルリソースからのデータベース接続の使用:

1. データベース接続ウィザードを開始します。
2. [グローバルリソース] をクリックします。グローバルリソースとして使用可能なデータベース接続がリストされます。



3. データベース接続履歴から「**接続**」をクリックします。

ヒント: 各グローバルリソースの追加情報を取得するために、マウスカーソルをグローバルリソースにポイントします。

10.2.10 データベース接続のサンプル

ADO、ODBC、またはJDBC を使用したStyleVision からのデータベースへの接続のためのサンプルがこのセクションには含まれています。ADO.NET 接続サンプルは個別にリストされています サンプルADO.NET 接続文字列 を参照してください。For instructions about establishing a native connection to PostgreSQL とSQLite へのネイティブな接続の作成のための手順は、それぞれ PostgreSQL 接続のセットアップ とSQLite 接続のセットアップ を参照してください。

以下の点に注意してください:

- Windows 構成、ネットワーク環境、およびデータベースクライアント、または、サーバーソフトウェアが各サンプルと異なる場合手順は異なります。
- ほぼすべてのデータベースの型では、1つ以上のデータアクセステクノロジー(ADO、ADO.NET、ODBC、JDBC) またはドライバーを使用して接続することができます。データベース接続パフォーマンス、およびドライバーの機能と制限は、StyleVision 外で構成された選択されたドライバー、(該当する場合)データベースクライアントソフトウェア、および追加接続パラメータなどによって異なります。

10.2.10.1 Firebird (JDBC) への接続

このサンプルではFirebird データベースサーバーへの接続のサンプルの命令について説明されています。

必要条件:

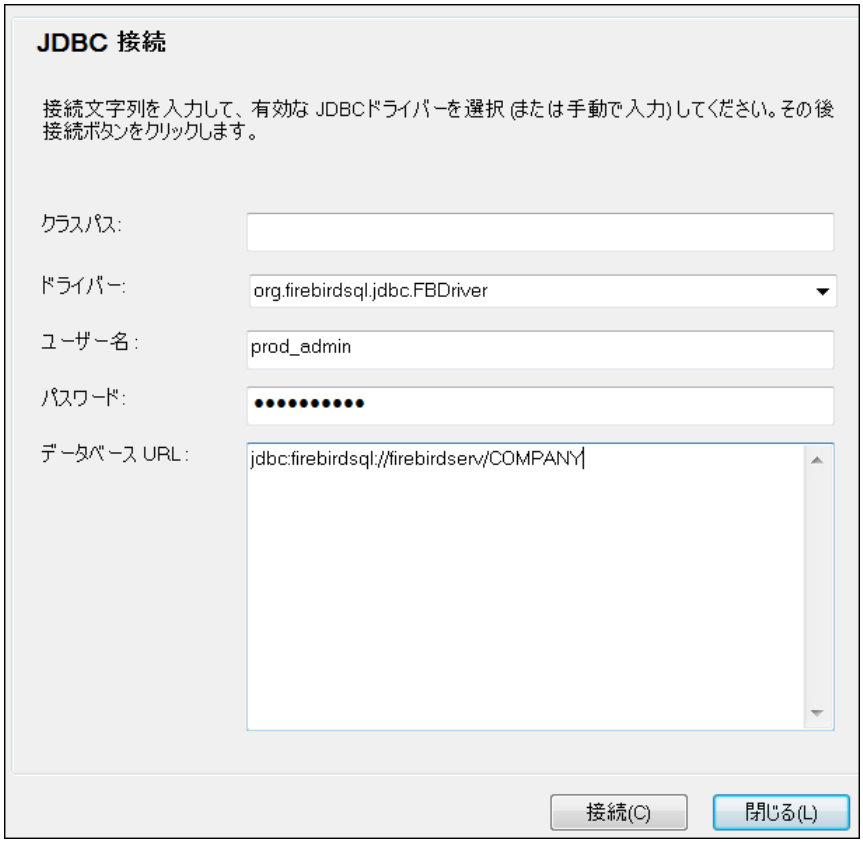
- JRE (Java Runtime Environment) またはJava Development Kit (JDK) がインストールされている必要があります。これはOracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。StyleVision は次のロケーションからJava Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーションオプション内で設定したカスタムJVM パス b) Windows レジストリ内で見つかったJVM パス。c) `JAVA_HOME` 環境変数。
- StyleVision (32-ビット、64-ビット) のプラットフォームがJRE/JDK のプラットフォームに一致することを確認してください。

(C) 2015-2021 Altova GmbH

- Firebird JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります（データベースへの接続を可能にする .jar ファイルのフォームをとります）。ドライバーは Firebird Web サイトからダウンロードすることができます（ http://www.firebirdsql.org/ ）。このサンプルは、*Jaybird 2.2.8* を使用します。
- データベース接続の詳細があります: ホスト、データベース パス、またはエイリアス、ユーザー名、および パスワード。

JDBC を使用して Firebird に接続する

1. データベース接続ウィザードを開始します.
2. 「JDBC 接続」をクリックします。
3. 「クラス パス」の横にデータベースへの接続を与える .jar ファイルへの パスを入力します。必要であれば、セミコロンで区切られた .jar ファイル パスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次の パスを使用して見つけることができます: **C:\jdbc\firebird\jaybird-full-2.2.8.jar**。.jar ファイル パスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラス パス」テキスト ボックスを空白のままにすることができます。（次も参照: CLASSPATH の構成）。
4. 「ドライバー」ボックスから、**org.firebirdsql.jdbc.FBDriver** を選択します。このエントリは、有効な .jar ファイル パスが、「クラス パス」テキスト ボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます（前のステップを参照してください）。



5. 対応するテキスト ボックスに、データベースへのユーザー名と パスワードを入力します。
6. URL テキスト ボックスに、データベースサーバーに適用することのできる ハイライトされた値を置き換える、データベースサーバーの接続文字列を入力します。

```
jdbc:firebirdsql://<host>[:<port>]/<databese path or alias>
```

7. 「接続」をクリックします。

10.2.10.2 Firebird (ODBC) への接続

このサンプルでは、Linux サーバー上で実行される、Firebird 2.5.4 データベースへの接続について説明されています。

必要条件:

- Firebird データベースサーバーは、クライアントからのTCP/IP 接続を受け入れるように構成されています。
- Firebird ODBC ドライバーを使用中のオペレーティングシステムにインストールする必要があります。このサンプルは、を使用します。 Firebird ODBC ドライバー バージョン 2.0.3.154 は、Firebird ウェブサイト (http://www.firebirdsql.org/) からダウンロードすることができます。
- Firebird クライアントを、使用中のオペレーティングシステムにインストールする必要があります。 Firebird 2.5.4 クライアントのためのスタンドアロンのインストーラーが存在しない点に注意してください。また、クライアント は、サーバーのインストール パックの一部です。Firebird サーバー インストール パックをFirebird ウェブサイト(http://www.firebirdsql.org/) からダウンロードすることができます。ダウンロードの際、[スーパークラシック/クラシック または スーパーサーバーのための、Windows 実行ファイル] インストーラーを選択してください。クライアントファイルのみをインストールする場合、[**最小クライアント インストール- サーバーツール無し**] を選択します。

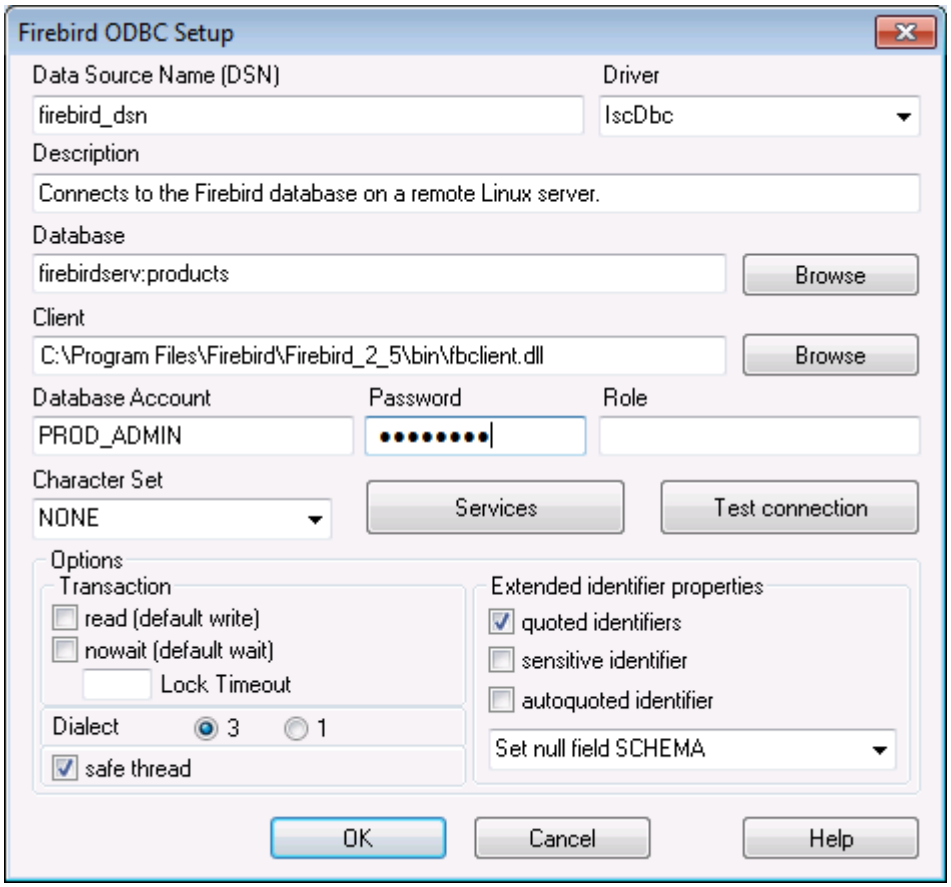
> **重要点:**
>
> - Firebird ODBC ドライバーおよび クライアント (32 ビット または 64 ビット) のプラットフォームは、StyleVision のプラットフォームに適応する必要があります。
> - Firebird クライアントの バージョンは、接続されているFirebird サーバーの バージョンに適応する必要があります。

- データベース接続の詳細 サーバー上の サーバー ホスト名 または IP アドレス、データベース パス(または エイリアス)、ユーザー名、および パスワード。

ODBC を使用して Firebird に接続する:

1. データベース接続ウィザードを開始します。
2. [**ODBC 接続**] をクリックする。
3. [**ユーザー DSN**] を選択して、(または、管理者の特権がある場合、 [**システム DSN**] を選択して、)[**追加**] をクリックします。

4. Firebird ドライバーを選択して、をクリックします。そして [ユーザー DSN] (前のステップでの選択により、または、[システム DSN])。Firebird ドライバーがリスト内に存在しない場合、使用中のオペレーティングシステムにインストールされていることを確認してください (使用可能な ODBC ドライバーの表示 を参照)。



5. データベース接続の詳細を入力します:

データソース名 (DSN)	作成する詳細名とデータソースを入力します。

データベース	サーバーホスト名 または IP アドレスをコロンとデータベース エイリアス(または パス)の後に入力します。このサンプルでは、ホスト名は、`firebirdserv`、データベースエイリアスは、`products` です:
	
	`firebirdserv:products`
	
	データベース エイリアスを使用することにより、サーバー側上では、データベース管理者は、エイリアス製品を構成し、サーバー上の実際の Firebird (.fdb) データベースファイルをポイントします (詳細は、Firebird のドキュメントを参照してください)。
	
	ホスト名の代わりに サーバー IP アドレス、エイリアスの代わりに パス を使用することもできます。ですから、以下のサンプル接続の文字列は有効です:
	
	`firebirdserver:/var/Firebird/databases/butterflies.fdb`
	`127.0.0.1:D:\Misc\Lenders.fdb`
	データベースが Windows マシン上に存在する場合、[参照] をクリックして、Firebird (.fdb) データベースファイルを直接選択します。

クライアント	**fbclient.dll** ファイルへの パスを入力します。デフォルトでは、これはFirebird インストールディレクトリのサブディレクトリです。
データベース アカウント	データベース管理者（このサンプルの場合 PROD_ADMIN）により与えられたデータベースユーザー名を入力します。
パスワード	データベース管理者により与えられた データベース パスワード を入力します。

6. 「OK」をクリックします。

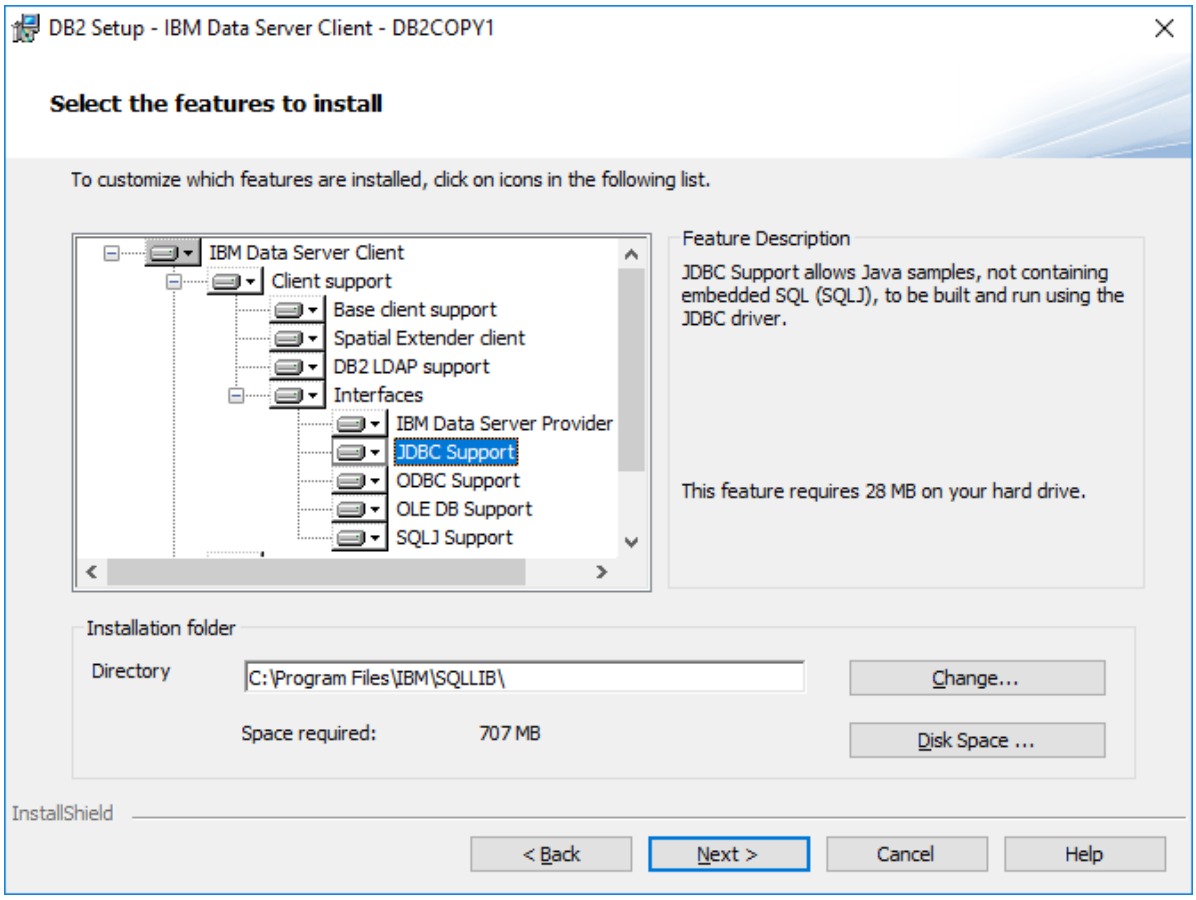
10.2.10.3 IBM DB2 (JDBC) への接続

このサンプルでは、JDBC を使用して **IBM DB2** データベースサーバーに接続する手順のサンプルについて説明されています。

必要条件:

- JRE（Java Runtime Environment）またはJava Development Kit（JDK）がインストールされている必要があります。Oracle JDK またはOracle OpenJDK などのオープンソースビルドであることができます。StyleVision は次のロケーションからJava Virtual Machine（JVM）へのパスを次の順序で決定します: a）アプリケーション オプション内で設定したカスタム JVM パス b）Windows レジストリ内で見つかったJVM パス。c）JAVA_HOME 環境変数。
- StyleVision（32-ビット、64-ビット）のプラットフォームがJRE/JDK のプラットフォームに一致することを確認してください。このサンプルはOracle のOpenJDK 11.0 64-bit を使用し、その結果 StyleVision の64 ビットバージョンを使用します。
- （データベースへの接続を与える1 つ、または、複数の.jar ファイル） JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります。**IBM Data Server Client** バージョン10.1（64-ビット）をインストール後使用することのできるJDBC ドライバーをこのサンプルは使用しています。JDBC ドライバーをインストールするためには、**通常** インストールを選択してください。または、インストールウィザードでこのオプションを明示的に選択してください。

(C) 2015-2021 Altova GmbH

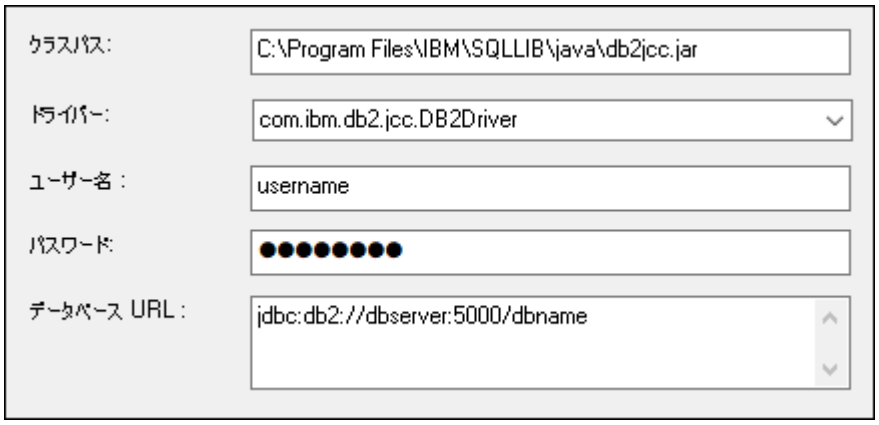


デフォルトのインストールパスを変更していない場合、インストール後必要とされる .jar ファイルは **C:\Program Files\IBM\SQLLIB\java** ディレクトリ内で見つけることができます。

- 以下のデータベース詳細が必要になります: ホスト、データベース、ポート、ユーザー名、および、パスワード。

JDBC を使用して IBM DB2 に接続する:

1. データベース接続ウィザードを開始します.
2. 「JDBC 接続」をクリックします。
3. 「クラスパス」の横に、データベースへの接続を与える .jar ファイルへのパスを入力します。このサンプルは **C:\Program Files\IBM\SQLLIB\java\db2jcc.jar** を参照します。データベースサーバーバージョンにより **db2jcc4.jar** ドライバーを参照する必要がある場合があります。ドライバーの互換性に関しては IBM ドキュメントを参照してください（http://www-01.ibm.com/support/docview.wss?uid=swg21363866）。.jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加している場合クラスパステキストボックスを空にしておくことができます（CLASSPATH の構成を参照してください）。
4. 「ドライバー」ボックスから、**com.ibm.db2.jcc.DB2Driver** を選択します。このエントリは、有効な .jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます（前のステップを参照してください）。



5. データベースユーザーのユーザー名とパスワードを対応するテキストボックスに入力します。
6. JDBC 接続文字列をデータベース URL テキストボックスに入力します。使用中のデータベースサーバーに適用することのできるものと接続の詳細が置き換えられていることを確認してください。

```
jdbc:db2://hostName:port/databaseName
```

7. 「接続」をクリックします。

10.2.10.4 IBM DB2 (ODBC) への接続

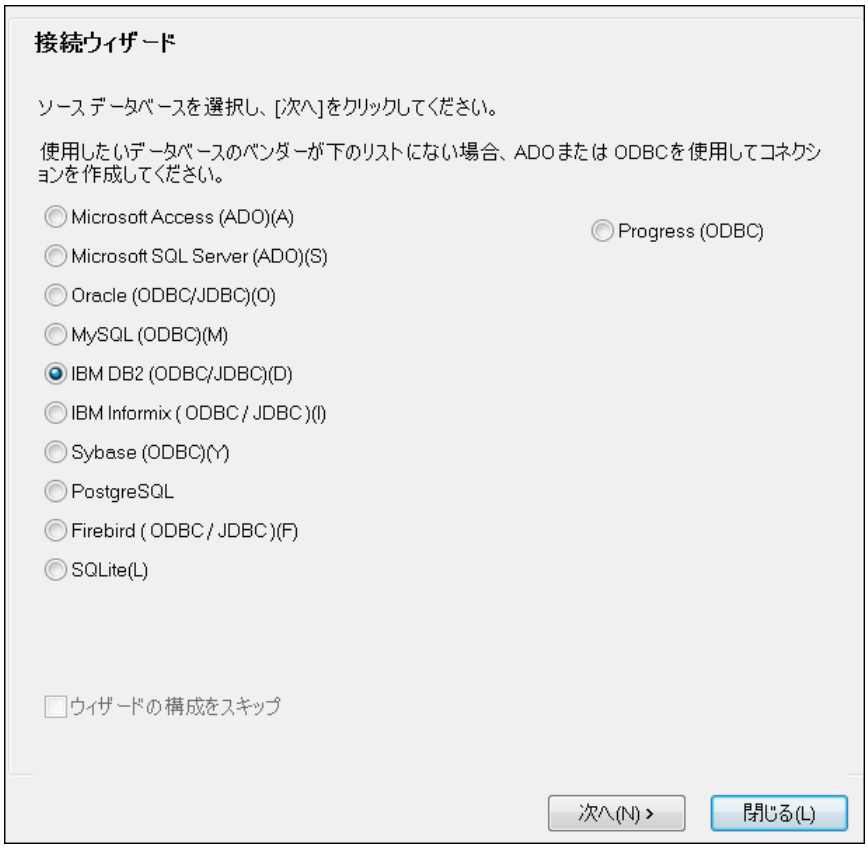
このトピックではODBC を使用した、IBM DB2 への接続のサンプル手順について説明されています。

前提条件:

- IBM Data Server Client は、使用中のオペレーティングシステムにインストールおよび構成される必要があります(このサンプルの場合はIBM Data Server Client 9.7 を使用します)。インストールの手順に関しては、IBM DB2 ソフトウェアにより提供されたドキュメントを確認してください。 IBM Data Server Client インストール後、使用中のコンピューターで ODBC ドライバーが使用可能か確認してください(使用可能な ODBC ドライバーの表示を参照)。
- データベースエイリアスを作成します。複数の作成方法があります:
  - IBM DB2 構成アシスタントから
  - IBM DB2 コマンドラインプロセッサーから
  - ODBC データソースウィザードから(この場合は、下に表示される手順を参照してください)
- データベース接続の詳細: ホスト、データベース、ユーザー名およびパスワード

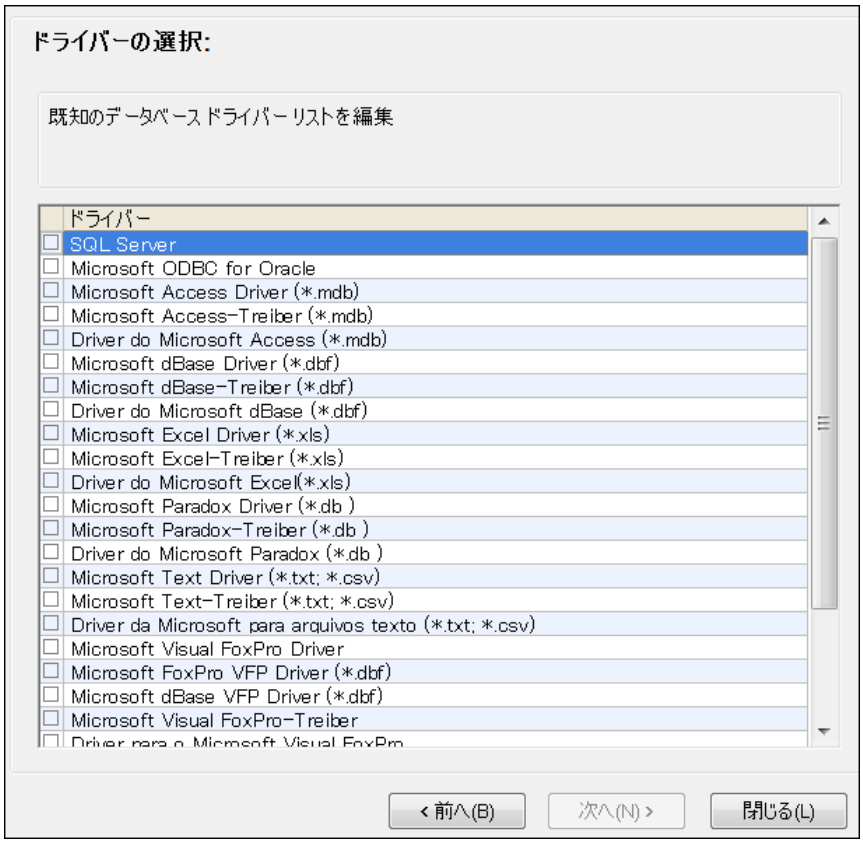
IBM DB2 への接続:

1. データベース接続ウィザードを開始します。**IBM DB2 (ODBC/JDBC)** を選択します。

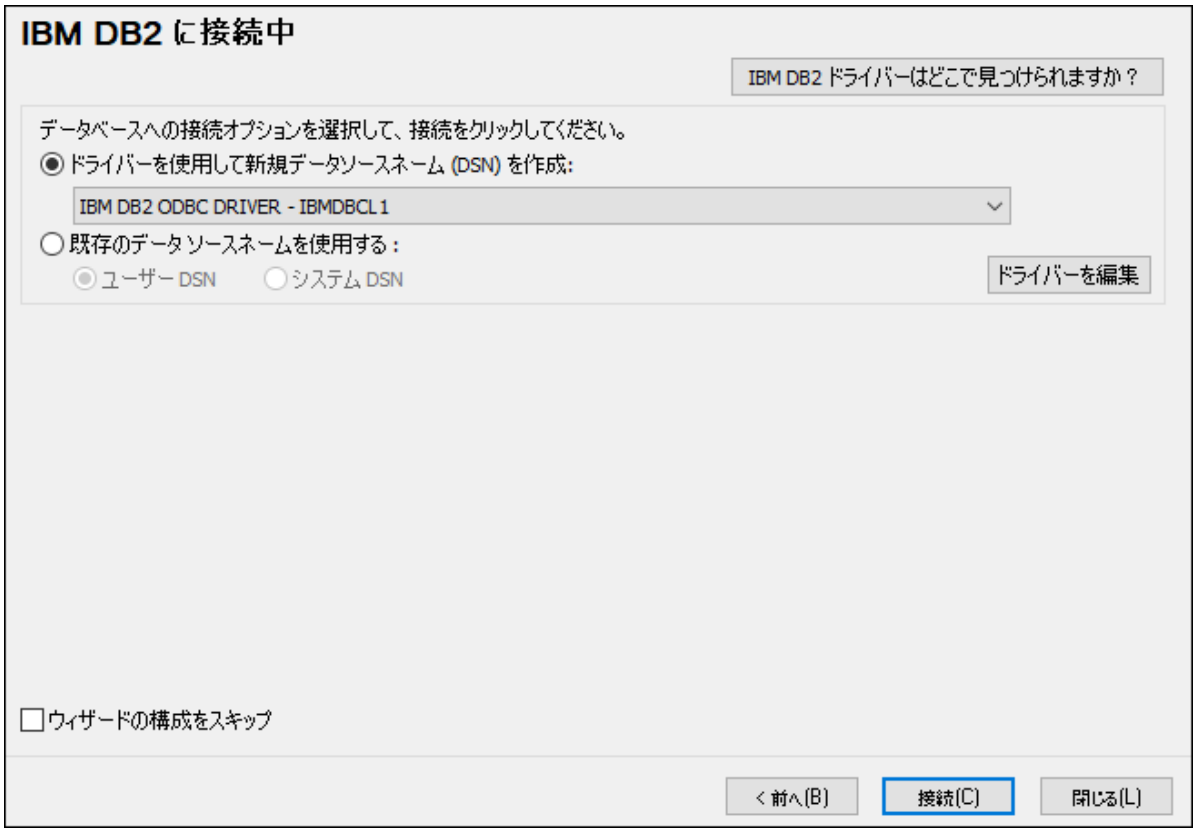


2. 「次へ」をクリックします。

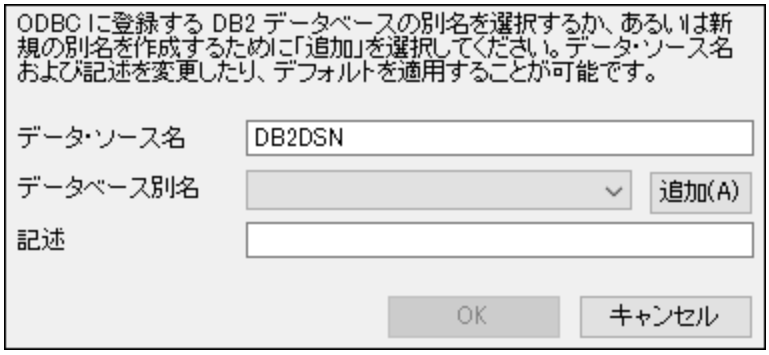
3. **ODBC** を選択して「次へ」をクリックします。データベースのための既知のリストを編集するようにプロンプトされると、IBM DB2 に適用することのできるドライバーを選択して(前提条件を参照)、「次へ」をクリックします。



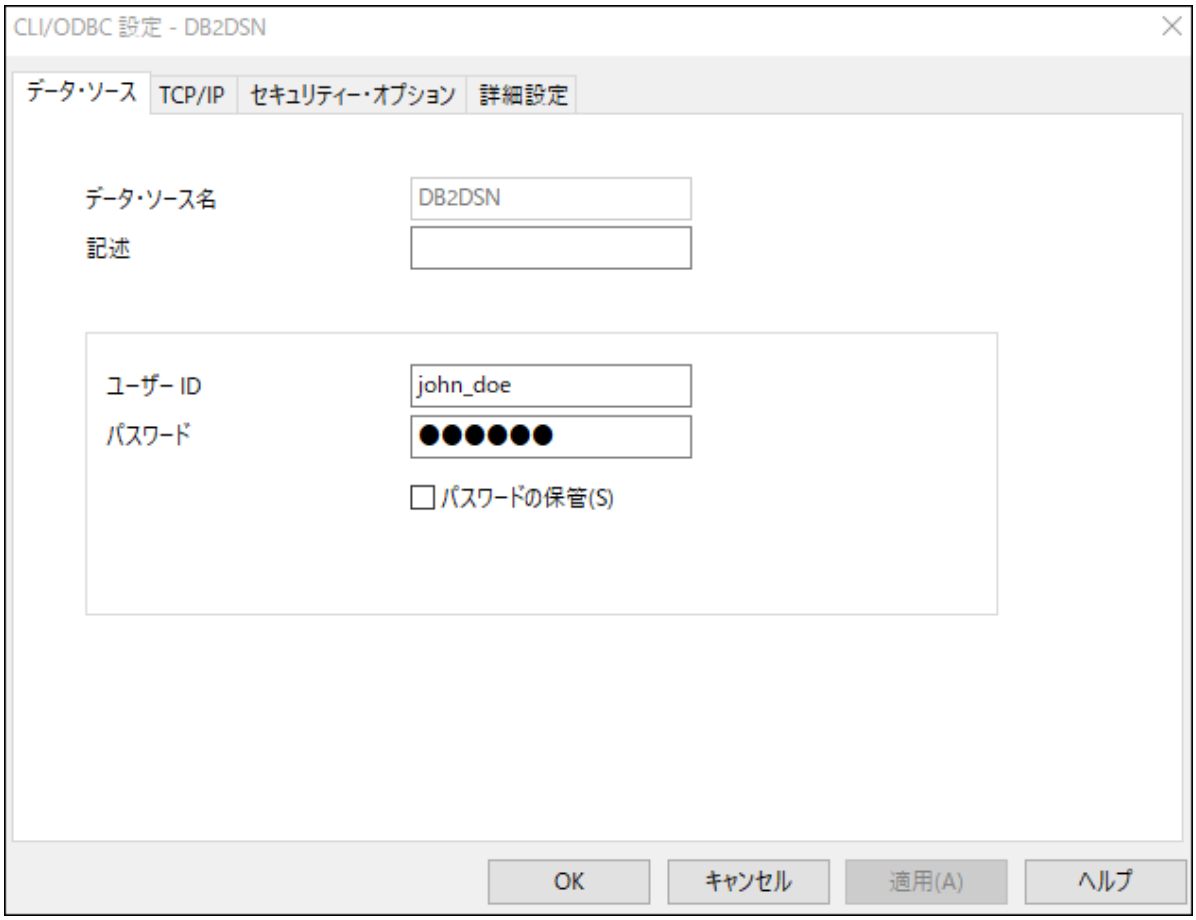
4. リストからIBM DB2 を選択して、「**接続**」をクリックします。（ドライバーで使用可能なリストを編集するには、[**ドライバーの編集**] をクリックして、使用可能なドライバーのリストを編集します。クリックして追加または削除するIBM DB2 ドライバーをチェックまたはチェックの解除をします。ドライバーを選択または選択の解除をします。）



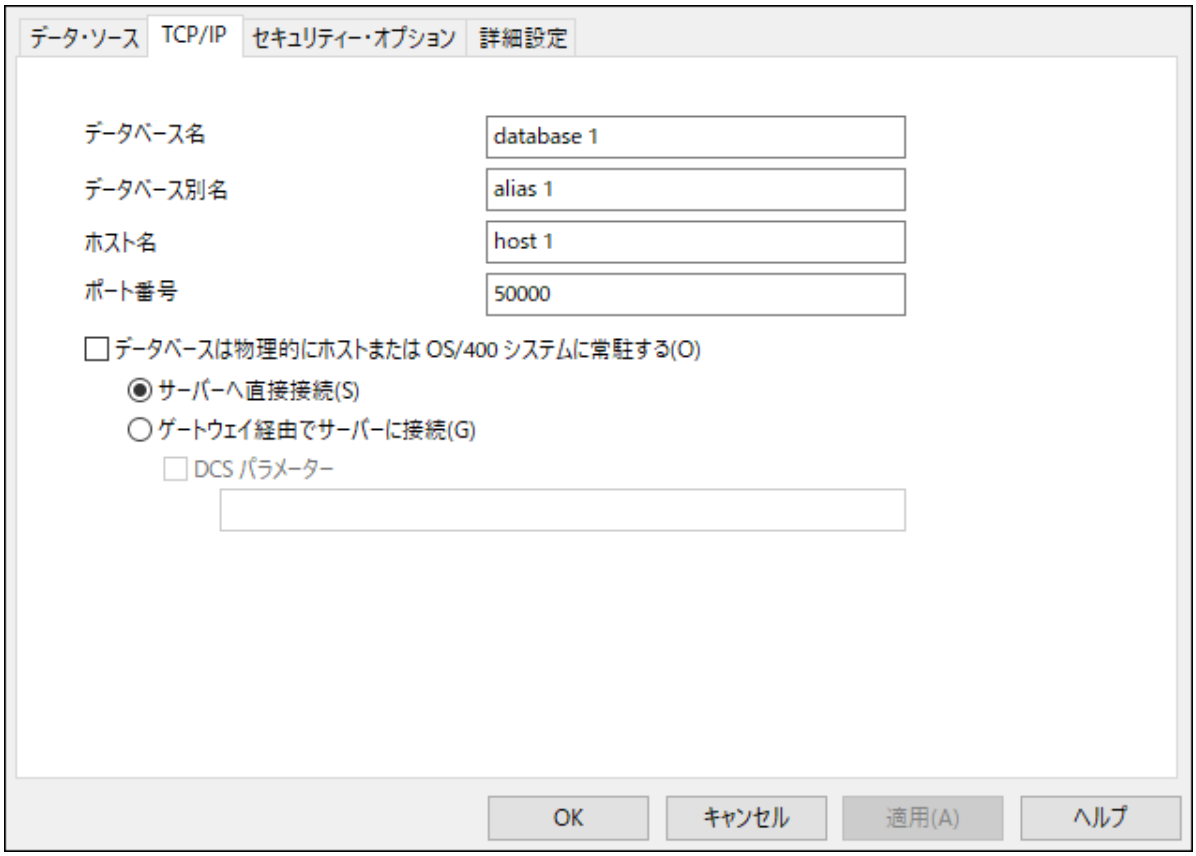
5. ソース名を入力します（このサンプルでは**DB2DSN**）「追加」をクリックします。



6. [データソース] タブで、データベースのユーザー名とパスワードを入力します。



7. **[TCP/IP]** タブで、データベース名、エイリアスの名前、ホスト名とポート名を入力して「OK」をクリックします。



8. ユーザ名とパスワードを再入力して「OK」をクリックします。

(C) 2015-2021 Altova GmbH

10.2.10.5 IBM DB2 for i (JDBC) への接続

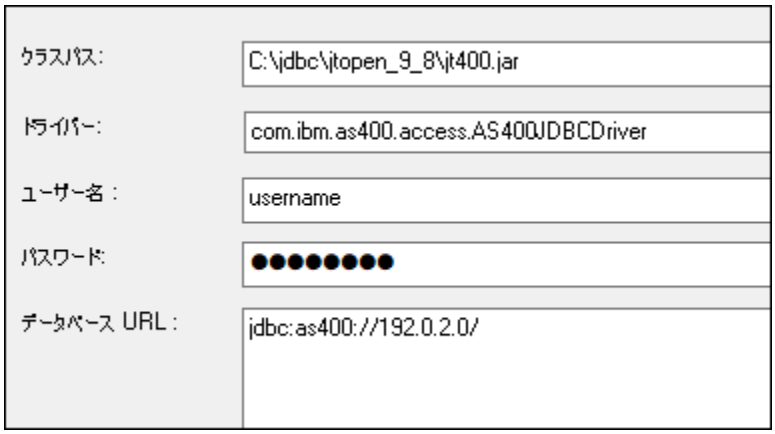
このサンプルでは JDBC を介して **IBM DB2 for i** データベースサーバーに接続する方法について説明されています。

必要条件

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。StyleVision は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- StyleVision (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。このサンプルは Oracle の OpenJDK 11.0 64-bit を使用し、その結果 StyleVision の 64 ビット バージョンを使用します。
- JDBC ドライバー (データベースへの接続を提供する 1 つ、または、複数の .jar ファイル) が使用中のオペレーティングシステム上で使用可能である必要があります。このサンプルは以下のオープンソースを使用します: **Toolbox for Java/JTOpen** バージョン 9.8 (http://jt400.sourceforge.net/)。パッケージをダウンロードし、ローカルディレクトリにアンパックすると、必要な .jar ファイルが **lib** サブディレクトリで使用できるようになります。
- 以下のデータベース接続の詳細が必要です: ホスト、ユーザー名、および パスワード

JDBC を使用して IBM DB2 for i に接続する方法:

1. データベース接続ウィザードを開始します。
2. 「**JDBC 接続**」をクリックします。
3. 「クラス パス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイル パスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます: **C:\Informix_JDBC_Driver\lib\ifxjdbc.jar**。.jar ファイル パスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラス パス」テキスト ボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成)。
4. 「ドライバー」ボックスから、**com.ibm.as400.access.AS400JDBCDriver** を選択します。このエントリは、有効な .jar ファイル パスが、「クラス パス」テキスト ボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます (前のステップを参照してください)。



5. データベースユーザーのユーザー名と パスワードを対応するテキスト ボックスに入力します。
6. JDBC 接続文字列をデータベース URL テキスト ボックスに入力します。**`host`** をデータベースサーバーのホスト名、または、IP アドレスと置き換えてください。

```
jdbc:as400://host
```

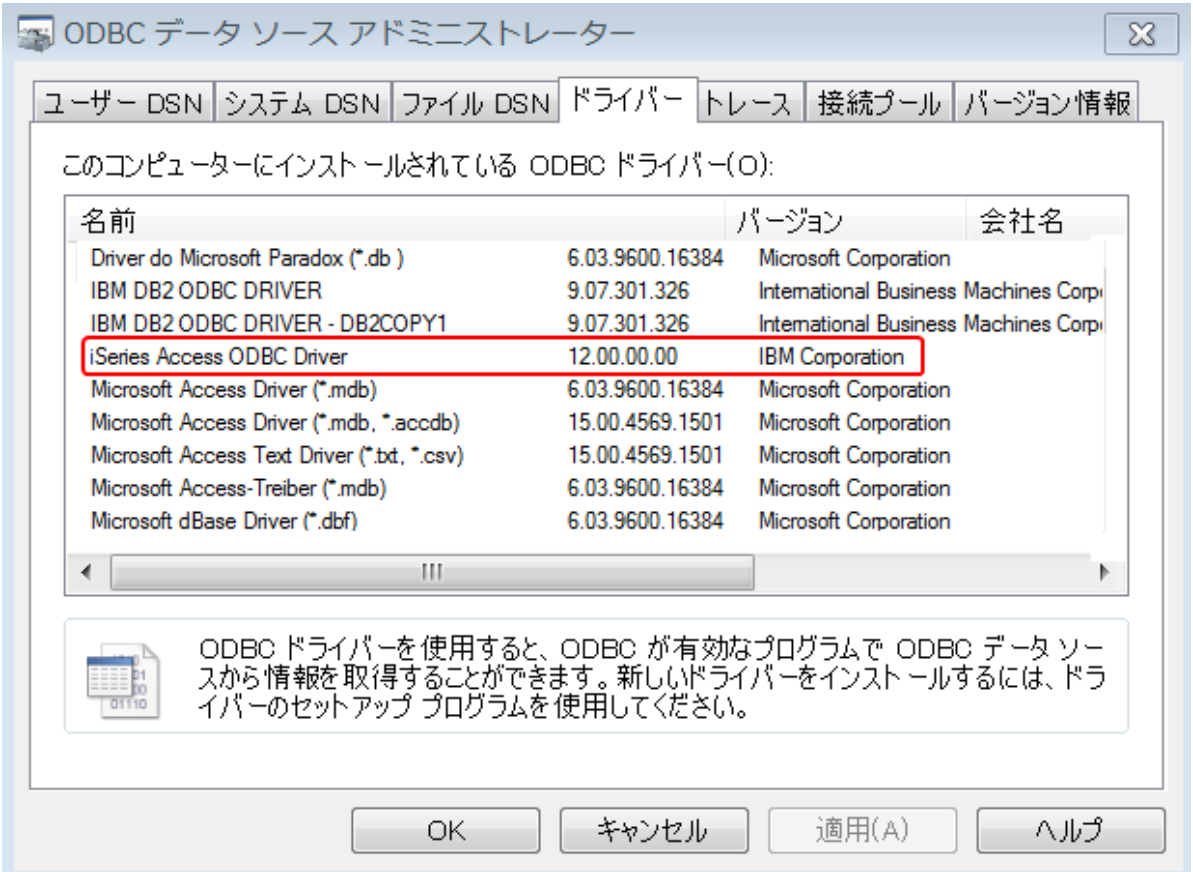
7. 「接続」をクリックします。

10.2.10.6 IBM DB2 for i (ODBC) への接続

このサンプルでは、ODBC を使用して *IBM DB2 for i* データベースへの接続について説明されています。

前提条件:

- *IBM System i Access for Windows* は使用中のオペレーティングシステム(このサンプルでは *IBM System i Access for Windows V6R1M0* を使用しています) にインストールされる必要があります。インストールの手順は、*IBM DB2 for i* ソフトウェアにより提供されているドキュメントを参照してください。インストール後、ODBC ドライバーが使用中のコンピューターで使用可能か確認してください (使用可能なODBC ドライバーの表示を参照)。



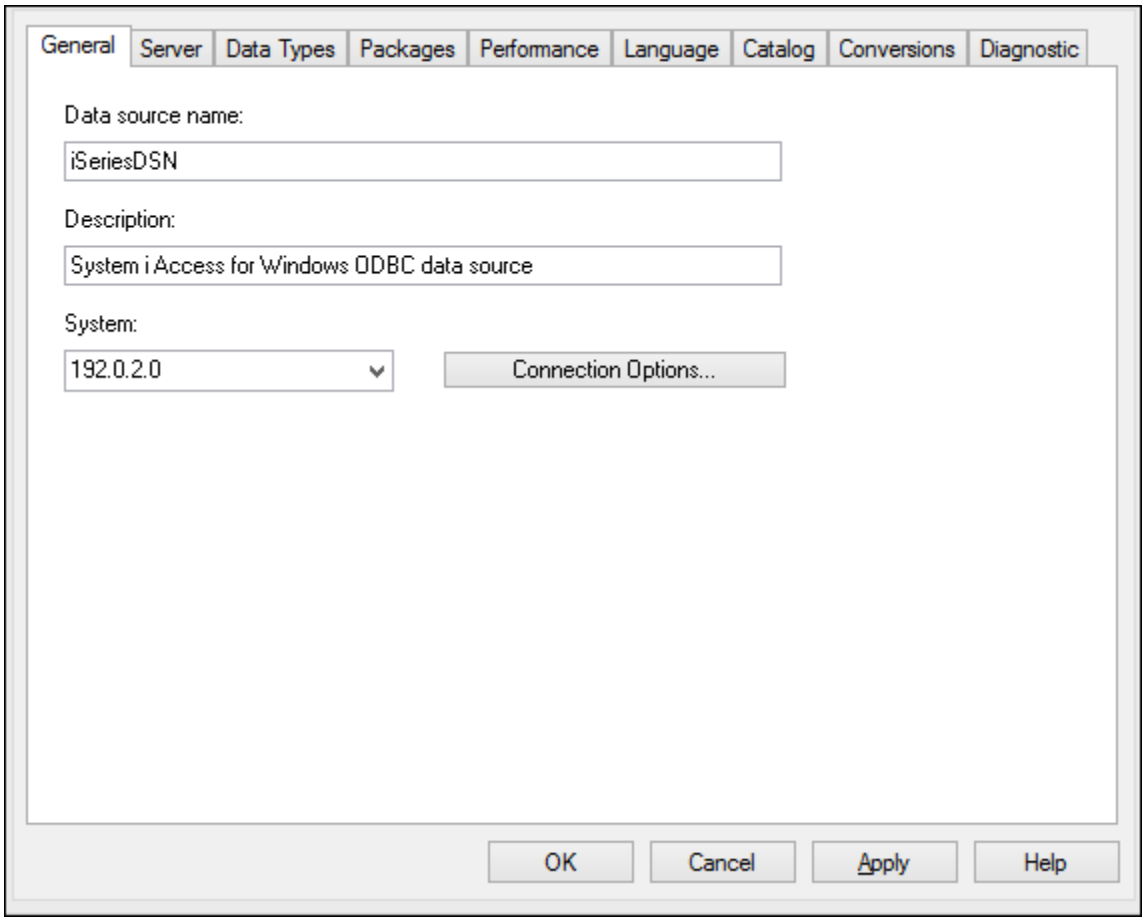
- データベース接続の詳細 データベースサーバーの I.P. アドレス、データベースユーザー名、および パスワード
- *i Navigator* を実行して、新規接続を作成するウィザードの手順を踏んで新規接続を作成してください。システムの指定をプロンプトされると、データベースサーバーのI.P. アドレスを入力してください。接続を作成した後、検証する(接続をクリックして、ファイル > 診断 > 接続の検証 を選択) ことが奨励されます。接続エラーが発生した場合、データベースサーバー管理者に連絡してください。

IBM DB2 for i への接続:

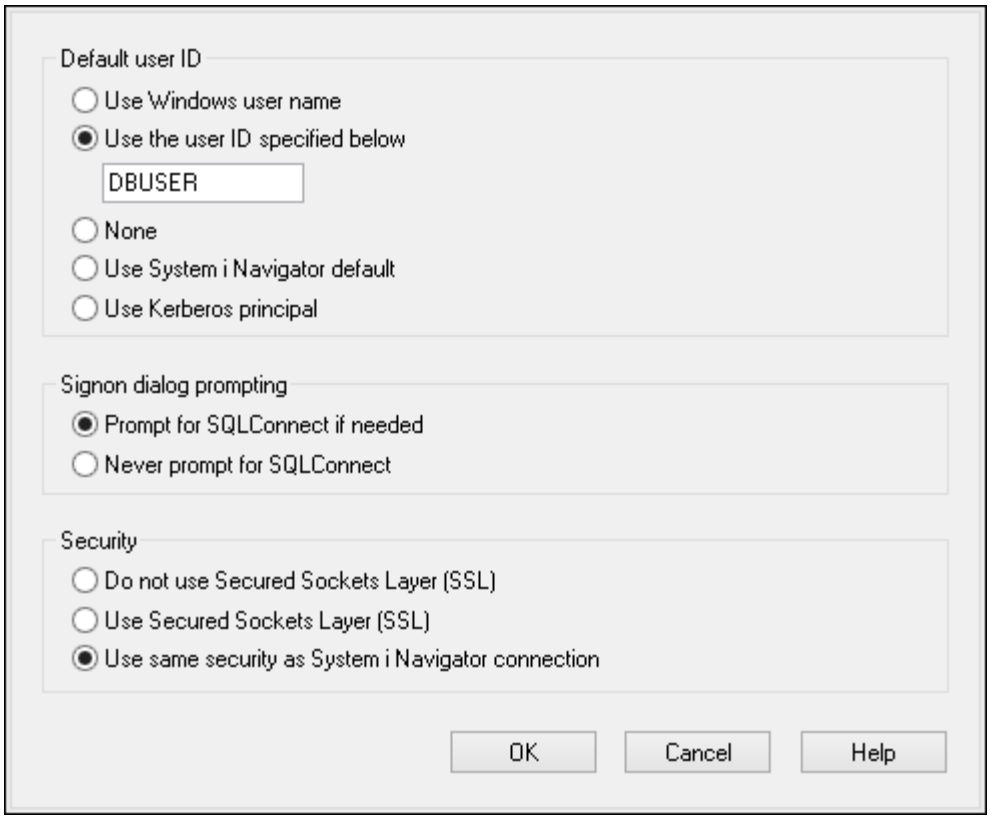
1. データベース接続ウィザードを開始します.

2. [ODBC 接続] をクリックします。
3. [ユーザーDSN] をクリックします。(または、[システムDSN] または、後に続く手順が同じ [ファイルDSN] をクリックします)。
4. 「追加」 をクリックします。
5. **[iSeries Access ODBC Driver]** をリストから選択します。[ユーザーDSN] (または、該当する場合 [システムDSN]) をクリックします。

6. データソース名を入力して、システムコンボボックスから接続を選択します。このサンプルでは、データソース名は **iSeriesDSN** でシステムは **192.0.2.0** です。



7. [接続オプション] をクリックします。[指定されたユーザー ID を使用] を選択して、データベースユーザー（このサンプル内では **DBUSER**）の名前を入力します。



8. 「OK」をクリックします。DSN 内のリストで新しいデータソースを使用することができます。
9. 「接続」をクリックします。
10. データベースのユーザー名とパスワードをプロンプトされると入力して「OK」をクリックします。

10.2.10.7 IBM Informix (JDBC) への接続

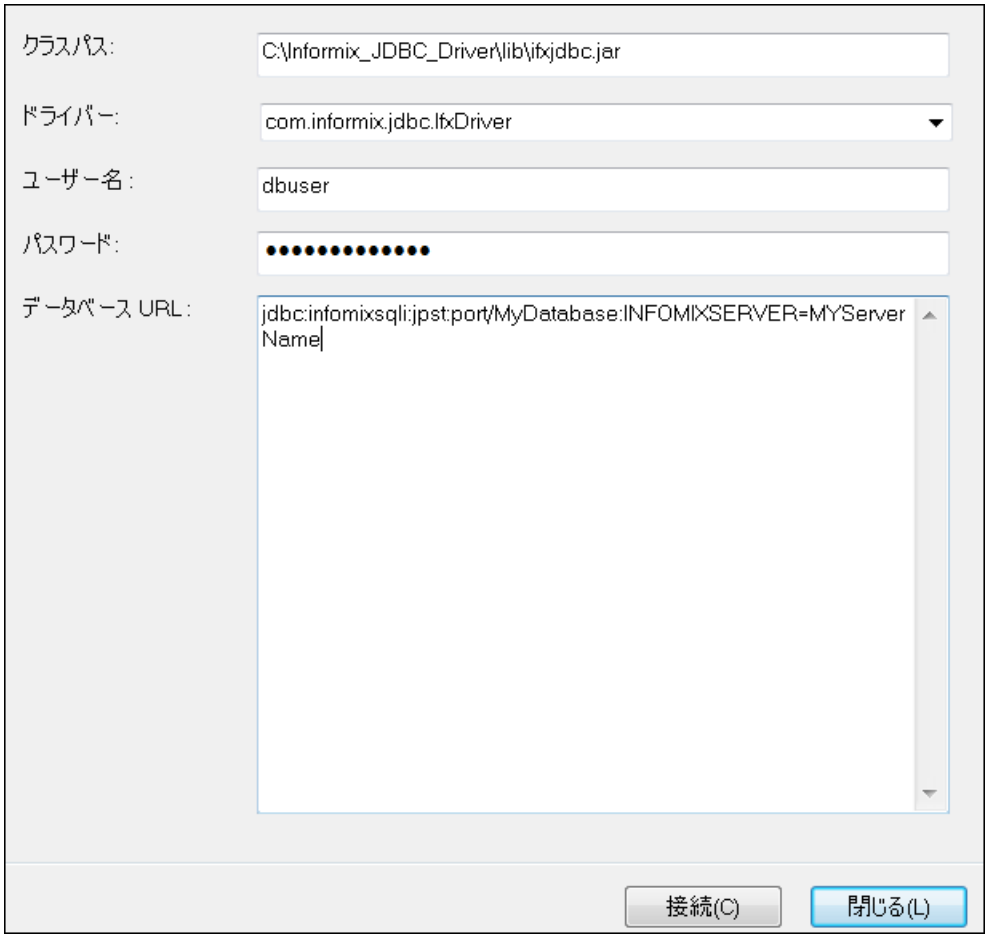
このサンプルでは、JDBC を使用して IBM Informix データベースサーバーに接続する手順のサンプルについて説明されています。

前提条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。StyleVision は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- StyleVision (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- (データベースへの接続を与える1つ、または、複数の .jar ファイル) JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります。この例では、IBM Informix JDBC ドライバー バージョン 3.70 が使用されています。ドライバーのインストールの説明に関しては、ドライバーに付随するドキュメント、または、「IBM Informix JDBC ドライバ プログラマーズガイド」を参照してください。
- データベース接続の詳細: ホスト、サーバーの名前、データベース、ポート ユーザー名、および パスワード

JDBC を使用して IBM Informix に接続する:

1. データベース接続ウィザードを開始します.
2. 「JDBC 接続」をクリックします。
3. 「クラスパス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます: **C:\Informix_JDBC_Driver\lib\ifxjdbc.jar**。.jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成)。
4. 「ドライバー」ボックスから、**com.informix.jdbc.IfxDriver** を選択します。このエントリは、有効な .jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます(前のステップを参照してください)。



5. データベースへのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値を、使用するデータベースサーバーに適用する値と置き換えて、データベース URL テキストボックス内にデータベースサーバーへの接続文字列を入力します。

```
jdbc:informix-sqli://hostName:port/databaseName:INFORMIXSERVER=myserver;
```

7. 「接続」をクリックします。

10.2.10.8 MariaDB (ODBC) への接続

このサンプルでは、ODBC を使用したMariaDB データベースへの接続方法について説明されています。

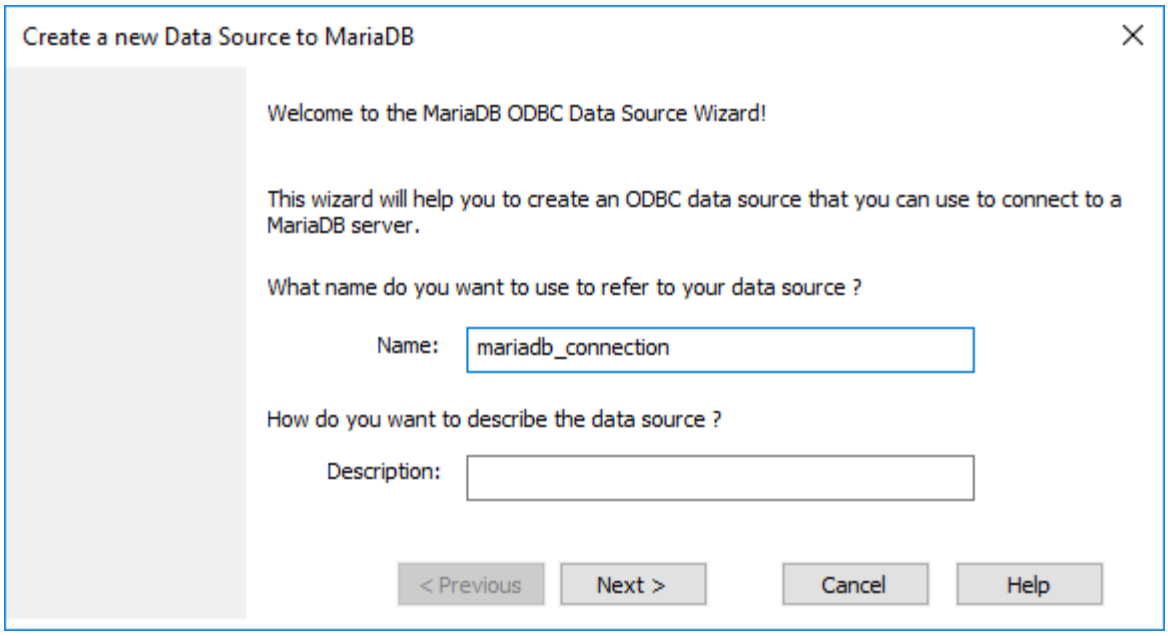
前提条件:

- MariaDB コネクタ/ODBC (https://downloads.mariadb.org/connector-odbc/) がインストールされている必要があります。
- 次のデータベース接続詳細が存在します: ホスト、データベース、ポート、ユーザー名、および、パスワード。

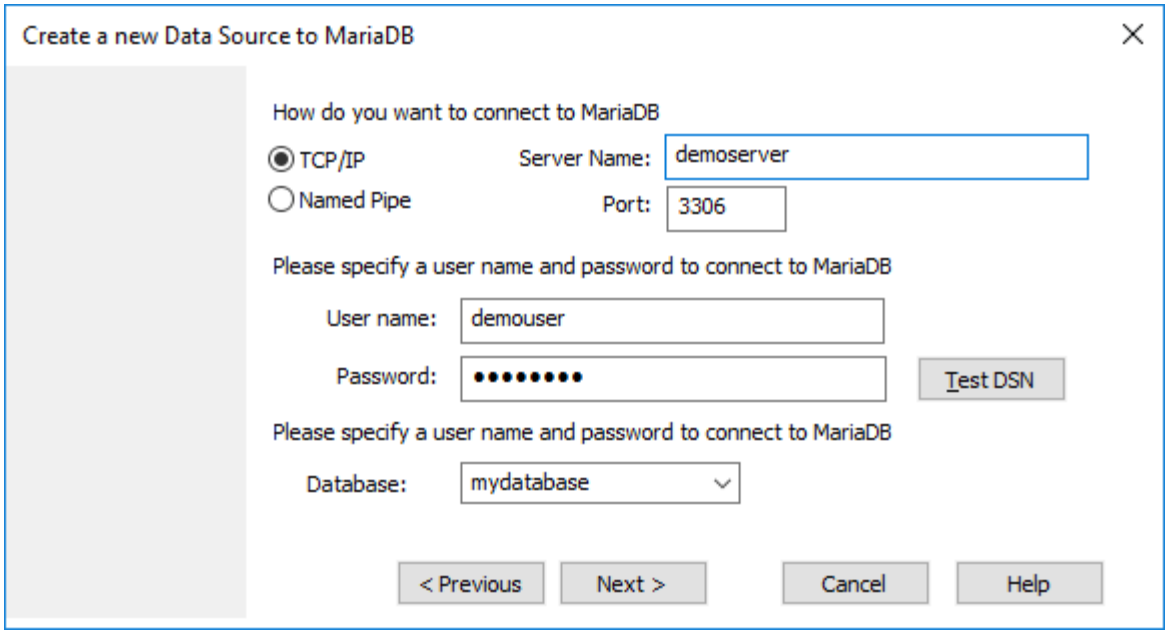
ODBC を使用した MariaDB への接続の方法:

1. データベース接続ウィザードを開始します。
2. 「MariaDB (ODBC)」を選択して、「次へ」をクリックします。

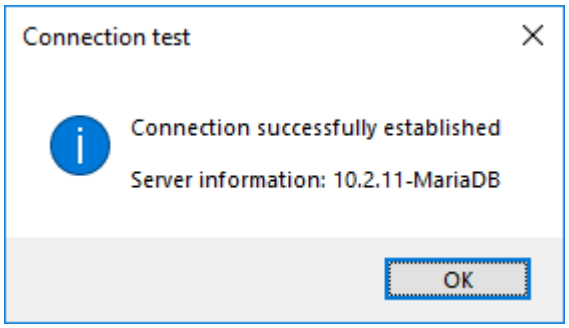
3. 「ドライバーを使用して新規のデータソース名 (DSN) を作成する」を選択し、「MariaDB ODBC 3.0 Driver」を選択します。このようなドライバーがリスト内で使用できない場合、「ドライバーの編集」をクリックし、使用することのできるMariaDB ドライバーを選択します (リストには使用中のオペレーティングシステム上にインストールされているすべてのODBC ドライバーが含まれています)。
4. 「接続」をクリックします。



5. 名前を入力し、任意で、このODBC データソースを将来識別できる詳細を与えます。



6. データベース接続資格情報(TCP/IP サーバー、ユーザー、パスワード)を入力し、データベースを選択して、「Test DSN」をクリックします。接続に成功すると、メッセージが表示されます:



7. 「次へ」をクリックしてウィザードを完了します。他のパラメータが必要になる場合がありますが(例えば、安全な接続を使用して、MariaDB に接続する場合は、SSL 証明書が必要になります)。

メモ　データベースサーバーがリモートの場合、サーバー管理者が遠隔の接続を使用中のマシンの IP アドレスから受け入れるように構成する必要があります。

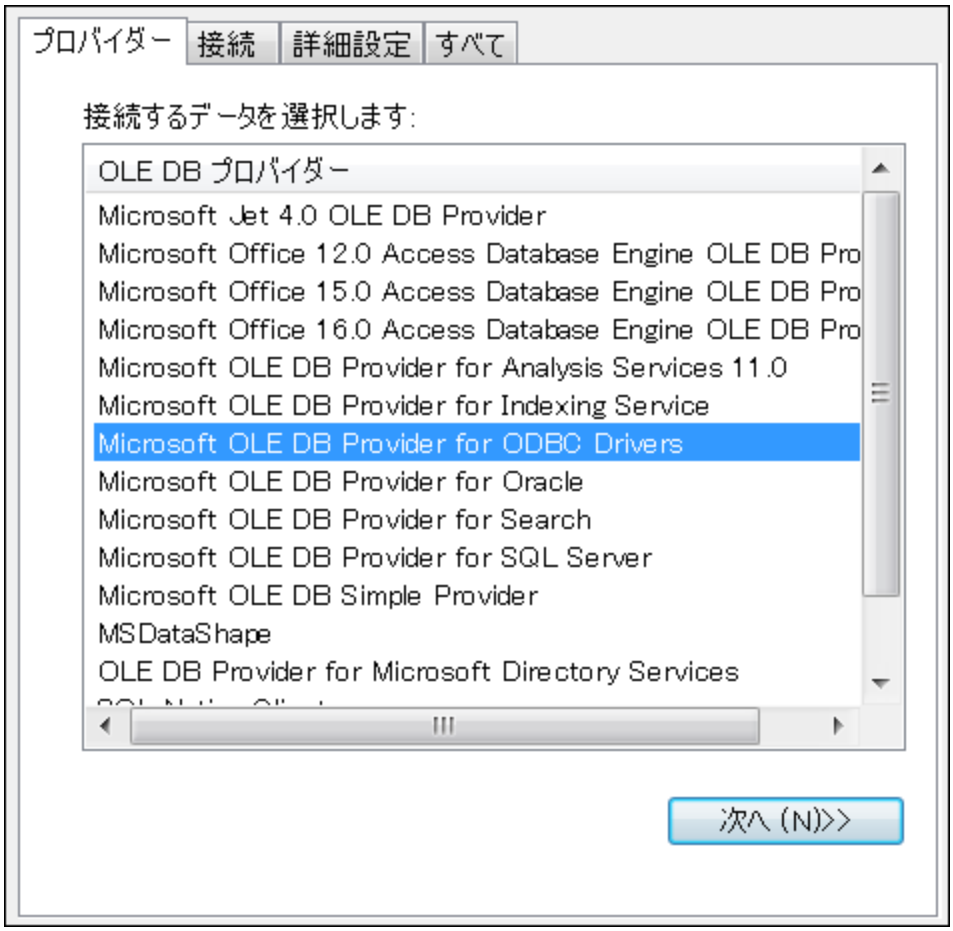
10.2.10.9 Microsoft Access (ADO) への接続

Microsoft Access データベースに接続するには、既存の Microsoft Access データベースに接続するで説明されているとおり、データベースファイルのためのウィザードとブラウザーの手順を踏みます。別の方法としては、このトピックで説明されているとおり、明確的に ADO 接続をセットアップすることです。このアプローチは、データベースがパスワードで保護されている場合、役に立ちます。

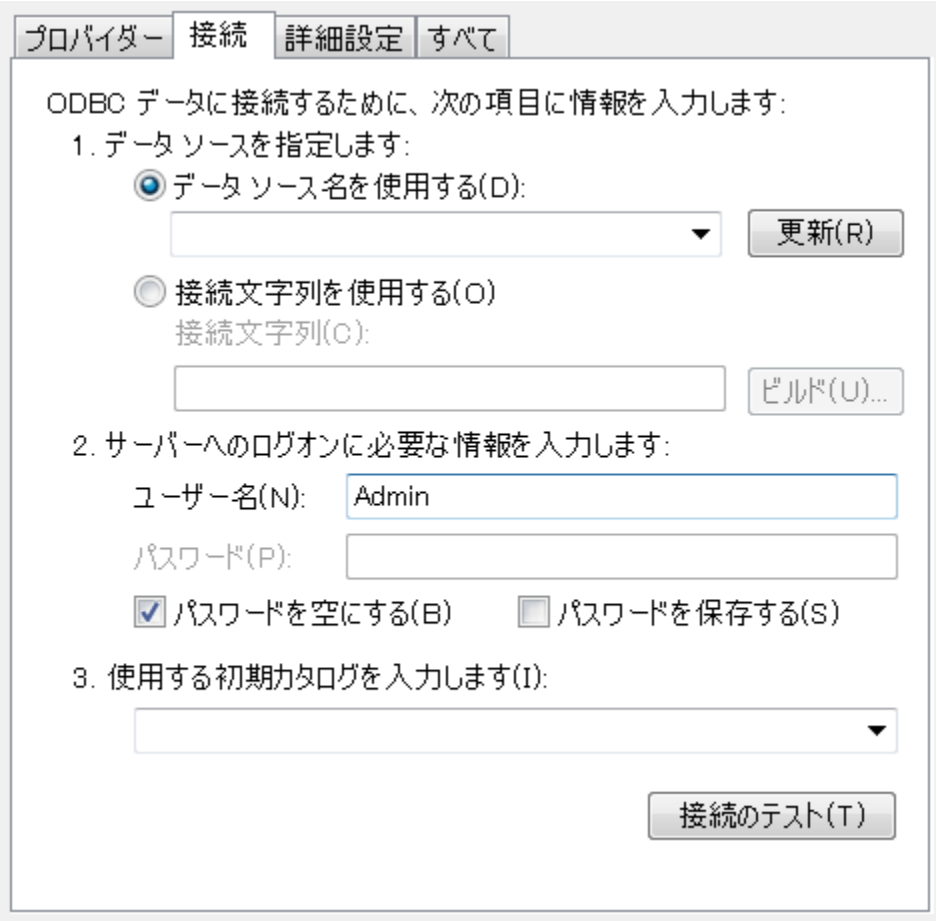
ODBC 接続を使用して、Microsoft Access に接続することもできますが、このシナリオには限度があるため、奨励されません。

パスワードで保護されている Microsoft Access データベースに接続する:

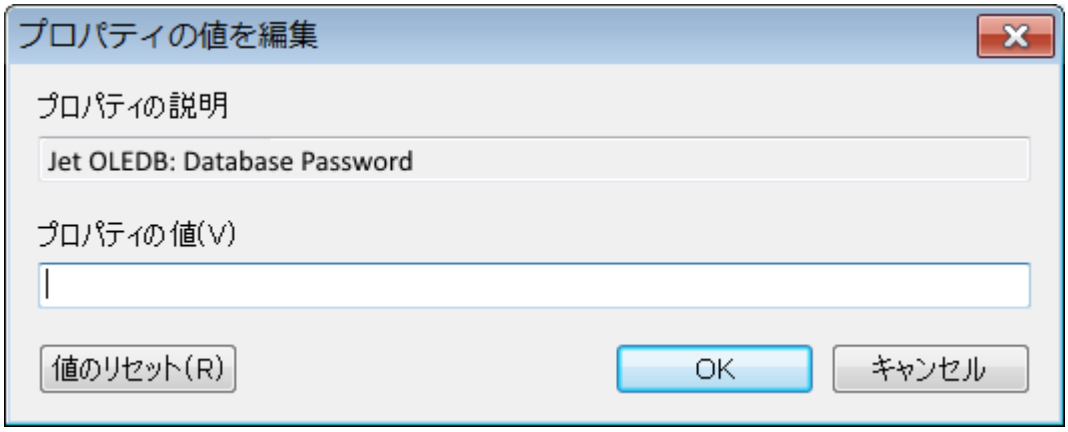
1. データベース接続ウィザードを開始します.
2. [ADO 接続] をクリックします。
3. 構築] をクリックします。



4. **[Microsoft Office 15.0 Access Database Engine OLE DB Provider]** を選択して、「次へ」をクリックします。



5. データソースボックス内でUNC フォーマットのMicrosoft Access ファイルへのパスを入力します。例えば、**myserver** がサーバーの名前であり **mynetworkshare** が共有するネットワークの名前である箇所での**\\myserver\\mynetworkshare\Reports\Revenue.accdb**。
6. 全てタブの**Jet OLEDB:Database Password** パスワード プロパティをダブルクリックし、データベースパスワードをプロパティの値として入力します。



メモ また接続できない場合、ワーキンググループ情報ファイル(**System.MDW**) を検索して、ユーザープロフィールに適用できるようにします。**Jet OLEDB: System database** プロパティの値を**System.MDW** ファイルのパスに設定します。

10.2.10.10 Microsoft SQL Server (ADO) への接続

このサンプルでは、ADO を使用したSQL Server データベースへの接続方法について説明されています。Https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 でダウンロード可能な奨励される **Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)**を使用する場合に適用される説明です。

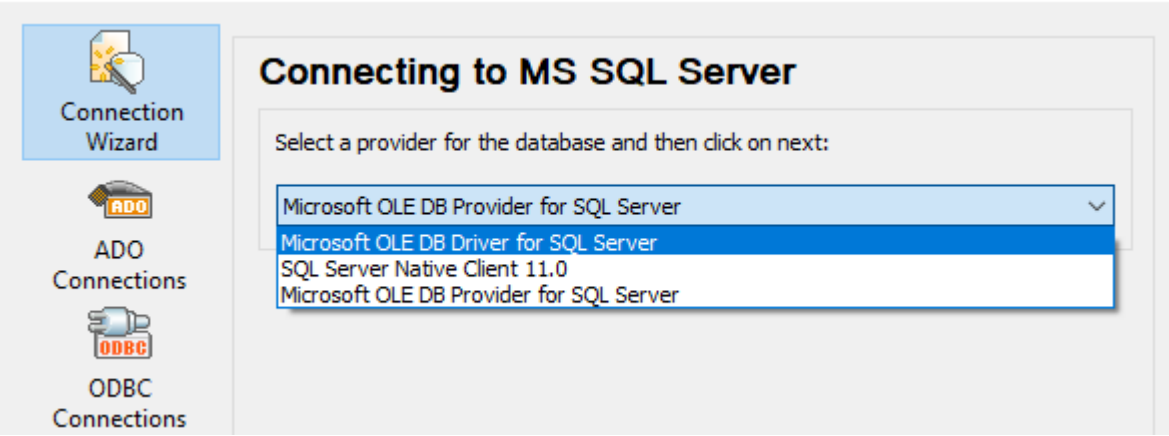
これらの説明を読む前に、ワークステーションに上記のプロバイダーがダウンロードされインストールされていることを確認してください。ADO プロバイダーはStyleVision (32-ビット または 64-ビット)のプラットフォームに一致する必要があることに注意してください。

SQL Server Native Client (SQLNCLI) または **Microsoft OLE DB Provider for SQL Server (SQLOLEDB)** などの他のADO プロバイダーを使用する場合、手順は同様ですが、これらのプロバイダーは奨励されていません。奨励されていないプロバイダーへの接続を成功させるには、SQL Server データリンクプロパティのセットアップで説明されている追加接続プロパティを設定する必要があります。

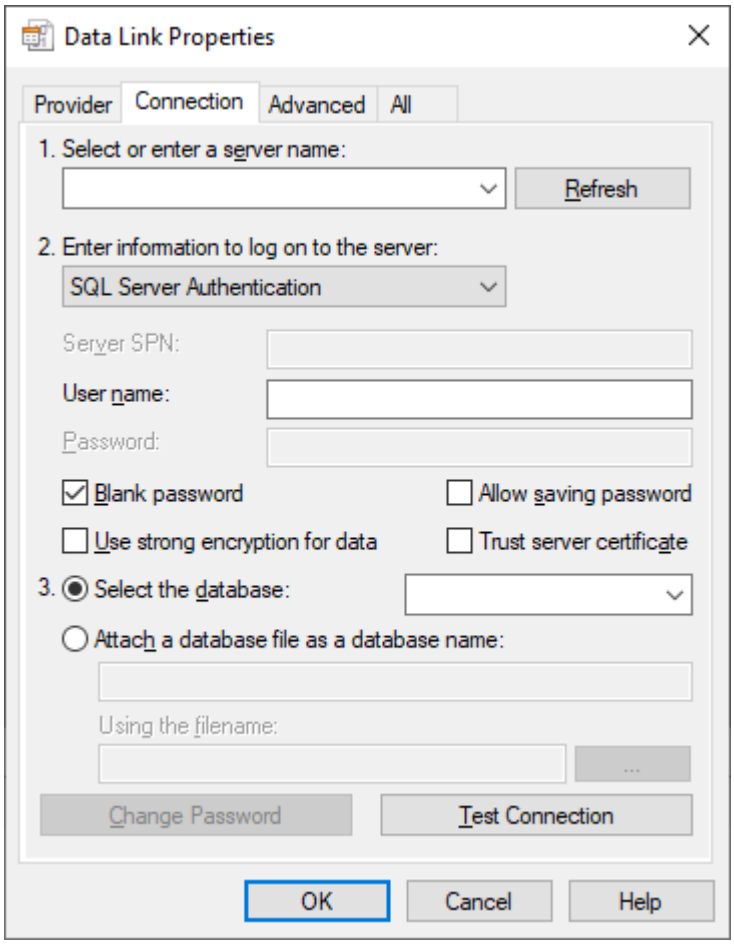
Microsoft OLE DB Provider for SQL Server (SQLOLEDB) は共通テーブル式 (CTE) とネストされたSELECT ステートメントなどの複雑なクエリのパラメーターバインディングに関する問題があることが知られています。

SQL サーバーに接続する方法:

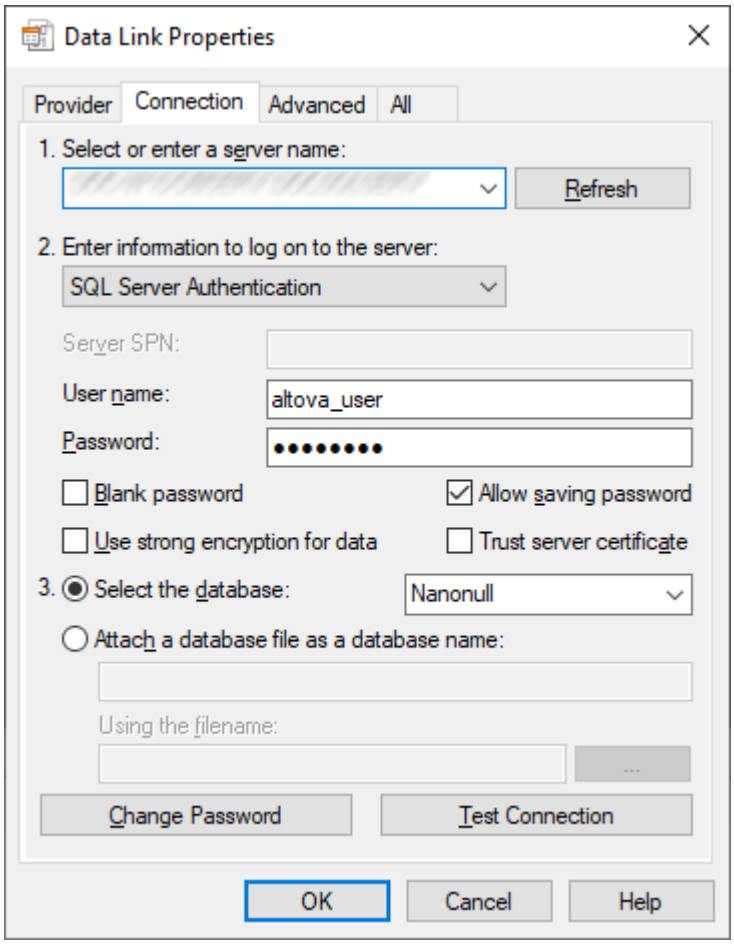
1. データベース接続ウィザードを開始します。
2. **[Microsoft SQL Server (ADO)]** を選択して、[次へ] をクリックします。使用することのできるADO プロバイダーのリストが表示されます。このサンプルでは **Microsoft OLE DB Driver for SQL Server** が使用されています。リストに存在しない場合、上記の通りコンピューターに接続されていることを確認してください。



3. 「次へ」をクリックします。データリンクプロパティダイアログボックスが表示されます。



4. データベースサーバーの名前を選択、または入力します。例えば、**SQLSERV01**。名前を持つSQL Server インスタンスに接続する場合、サーバー名は**SQLSERV01\SOMEINSTANCE** になります。
5. データベースサーバーがWindows ドメイン上の認証されているユーザーの接続を許可している場合、「**Windows 認証**」を選択してください。それ以外の場合は、[**SQL Server 認証**]を選択し、[**空のパスワード**] チェックボックスをクリアし、対応するボックスにデータベース証明書を入力します。
6. **パスワードの保存の許可** チェックボックスを選択し、接続するデータベースを選択します（このサンプル内では、Nanonull です）。



7. この時点で接続をテストする場合は、[**接続のテスト**]をクリックします。これは任意の奨励されるステップです。
8. **[OK]**をクリックします。

10.2.10.11 Microsoft SQL Server (ODBC) への接続

このサンプルでは、ODBC を使用したSQL Server データベースへの接続方法について説明されています。

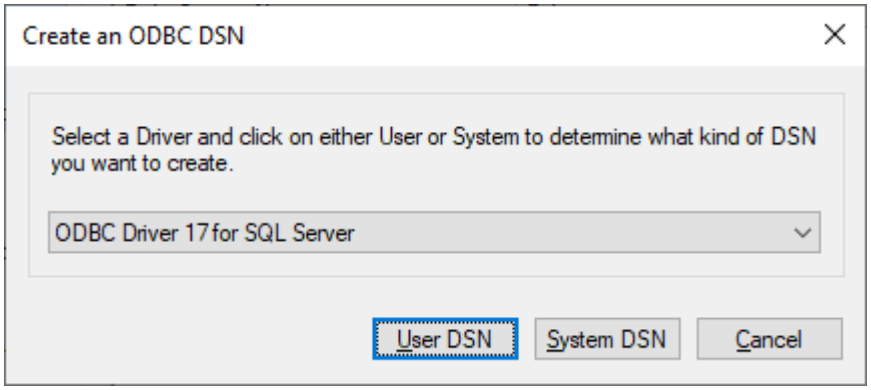
必要条件:

- **Microsoft ODBC Driver for SQL Server** をMicrosoft W eb サイトからダウンロードしてインストールします。https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server を参照してください。このサンプルは**Microsoft ODBC Driver 17 for SQL Server** を**SQL Server 2016** データベースに接続するために使用しています。接続するSQL Server のバージョンにより異なるODBC ドライバーバージョンをダウンロードする必要がある場合があります。使用中のSQL Server によりサポートされるODBC ドライバーバージョンに関する情報は、システムの必要条件を参照してください。

ODBC を使用してのSQL Server への接続:

1. データベース接続ウィザードを開始します。

2. [ODBC 接続] をクリックします。
3. [ユーザーDSN] を選択して、(または、管理者の特権がある場合、 [システムDSN] を選択して、)[追加] をクリックします。
4. ドライバーをリストから選択します。インストール後にのみドライバーはリスト内に表示されることにご注意してください。

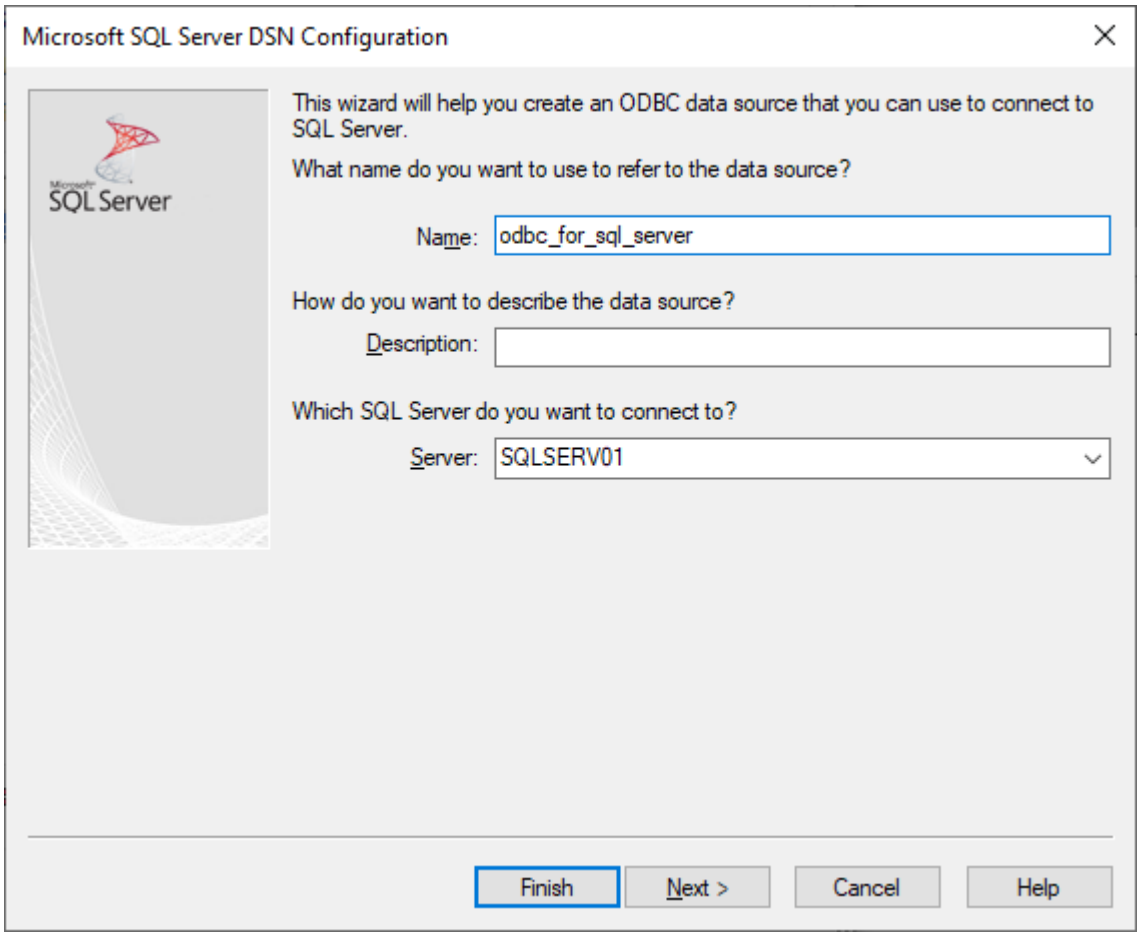


5. [ユーザーDSN] (またはシステムDSN を作成する場合) [システムDSN]をクリックします。

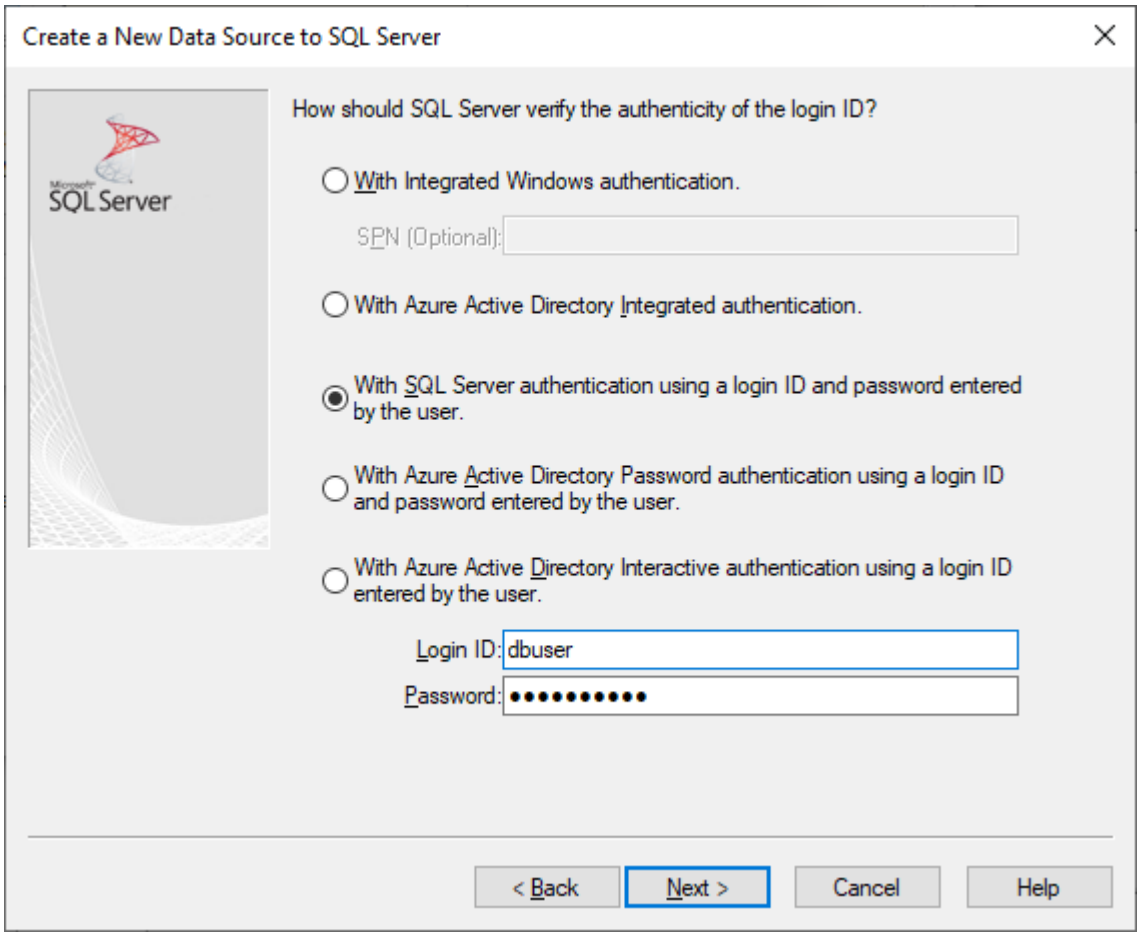
> StyleVision が管理者として作動することを必要とするシステムDSN の作成方法このため**System DSN** を作成するには、ウィザードをキャンセルし、StyleVision が管理者として作動していることを確認し、上記のステップを再度実行します。

6. 名前を入力して、任意でこの接続のIDの説明を入力し、接続するSQL Server (この場合 **SQLSERV01**)をリストから選択します。.

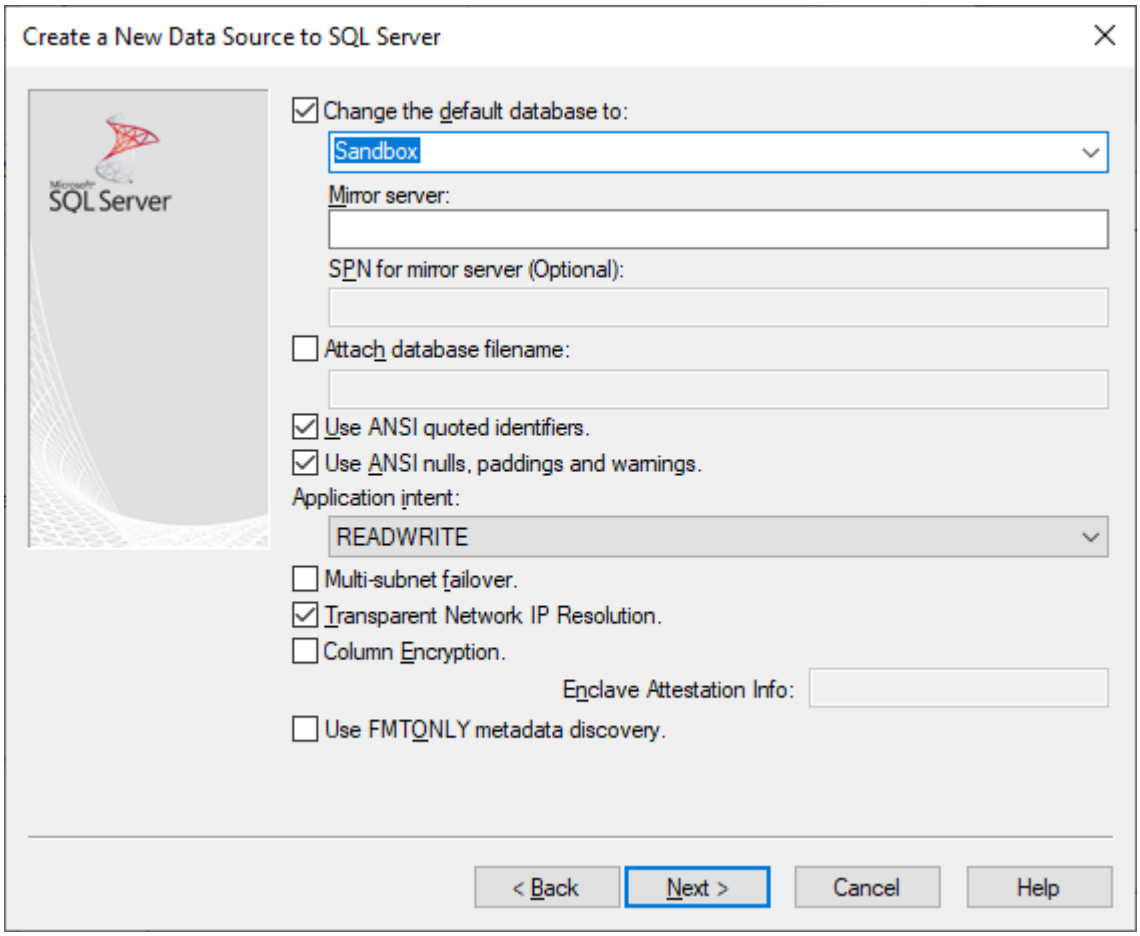
(C) 2015-2021 Altova GmbH



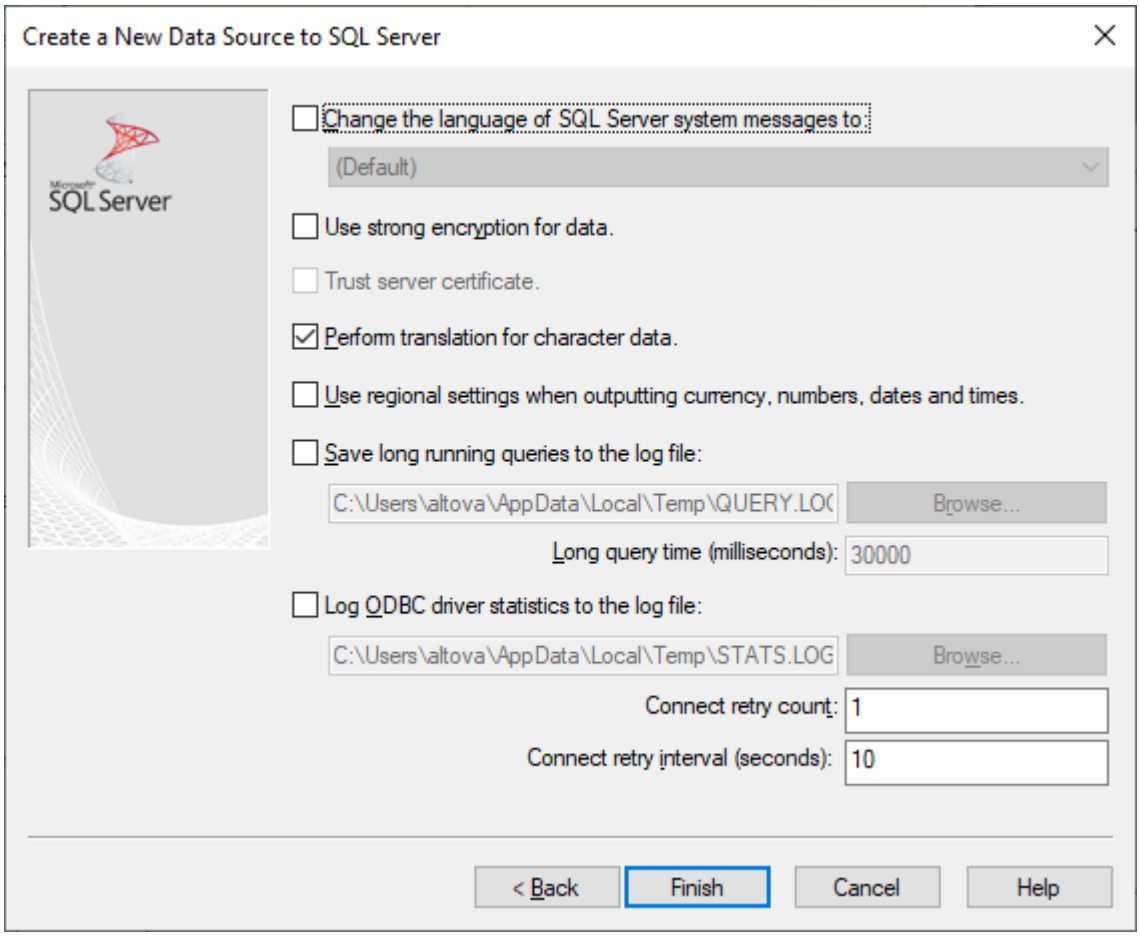
7. データベースサーバーがWindowsドメイン上の認証されているユーザーの接続を許可している場合、「**統合されたWindows認証**」を選択してください。それ以外の場合、適用可能なオプションから1つを選択してください。このサンプルは **SQL Server認証**...を使用しています。これはユーザー名とパスワードが関連するボックスに入力されることを必要としています。



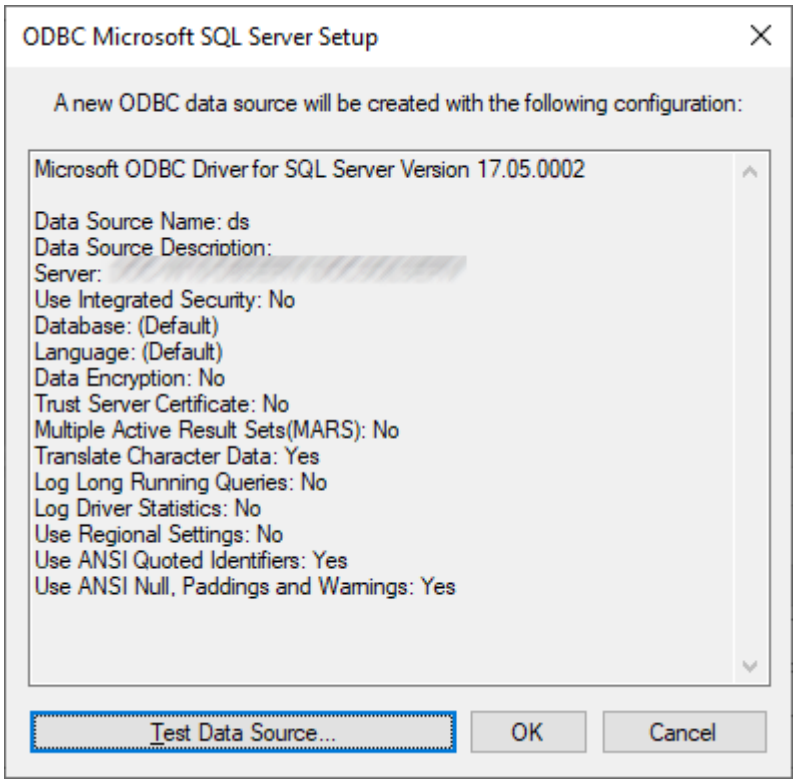
8. オプションで **デフォルトのデータベースを変更** チェックボックスを選択して、接続するデータベースの名前を入力します（このサンプルでは**Sandbox**）。



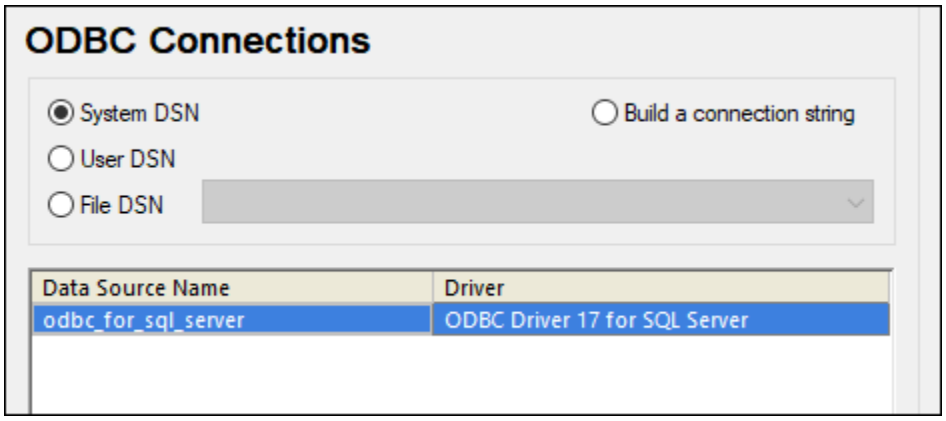
9. ［次へ］をクリックして、任意で、この接続のための追加パラメーターを構成します。



10. 「完了」をクリックします。接続の詳細をリストした確認ダイアログボックスが開かれます。



11. **[OK]**をクリックします。構成された通り ユーザー または システム データソースのリスト内でデータソースが表示されます。例:



10.2.10.12 MySQL (ODBC) への接続

このサンプルではODBC ドライバーを使用して、Windows マシンから MySQL データベースへ接続する方法について説明されています。Windows からは、MySQL ODBC ドライバーを入手することができませんので、個別にダウンロードおよびインストールする必要があります。このサンプルでは、MySQL コネクタ/ODBC 8.0 を使用しています。

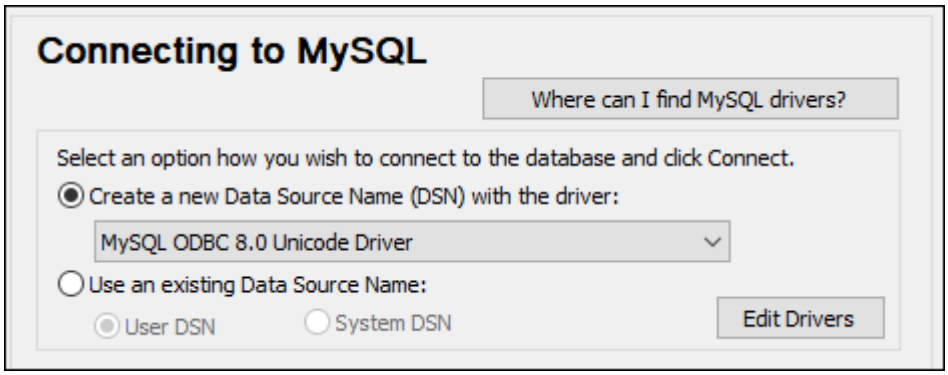
前提条件:

- MySQL ODBC ドライバーをオペレーティングシステム上にインストールする必要があります。使用するデータベースサーバーバージョンのために奨励されるドライバーバージョンのためのMySQLドキュメントを確認してください (https://dev.mysql.com/downloads/connector/odbc/ を確認してください)。
- データベース接続の詳細: ホスト、データベース、ユーザー名およびパスワード

64-ビットプラットフォームのためにMySQL Connector/ODBC をインストールしている場合、64-ビットプラットフォームのためにStyleVision をインストールしてください。

ODBC を介して MySQL に接続する方法:

1. データベース接続ウィザードを開始します。
2. **MySQL (ODBC)** を選択して、「次へ」をクリックします。

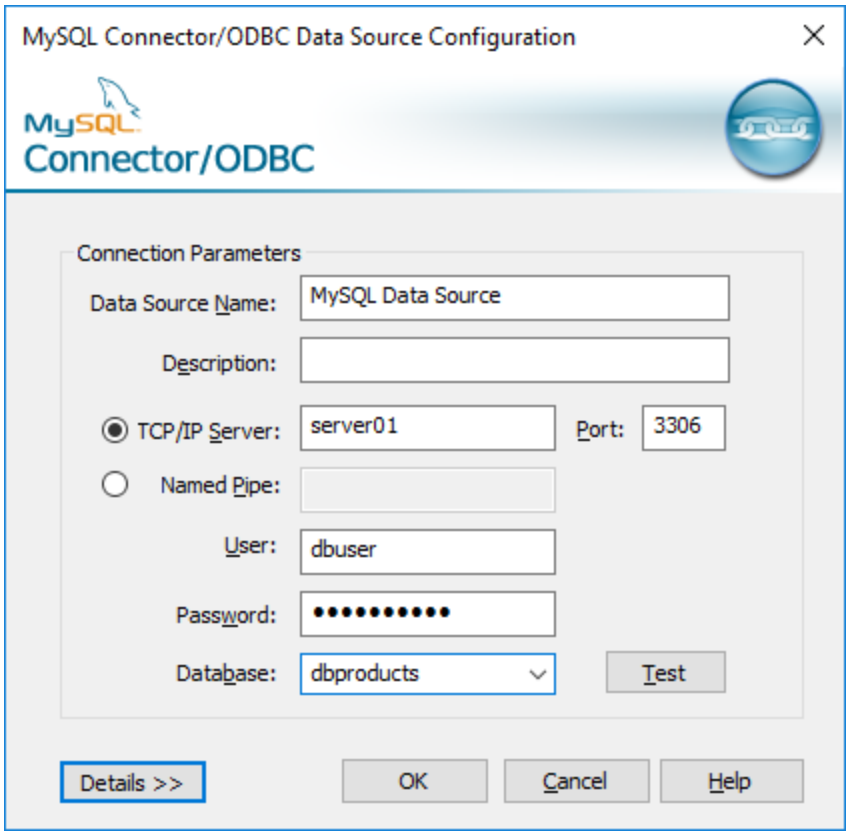


3. [ドライバーを使用して新しいデータソース名 (DSN) を作成] を選択して、MySQLドライバーを選択します。MySQLドライバーがリストにない場合は、[ドライバーを編集] をクリックして、使用可能なMySQLドライバーを選択します (使用中のオペレーティングシステムにインストールされているすべてのODBC ドライバーがリストされています)。

StyleVision 64-ビットをインストールすると、64-ビット ODBC ドライバーがリスト内に表示されます。それ以外の場合、32-ビット ODBC ドライバーが表示されます。使用可能な ODBC ドライバーの確認も参照してください。

4. 「接続」をクリックします。

(C) 2015-2021 Altova GmbH



5. データソース名ボックスの中に、今後このODBC データソースを認識するために役に立つ詳細を入力します。
6. データベース接続資格情報（TCP/IP サーバー、ユーザー、パスワード）を入力して、データベースを選択し、「OK」をクリックします。

メモ　データベースサーバーがリモートの場合、サーバー管理者が使用中のコンピューターのIP アドレスからのリモート接続を承諾するように構成される必要があります。また、[詳細>>] をクリックすると使用することのできる追加パラメーターが表示されます。デフォルトの値を変更する前に、ドライバーのドキュメントを確認してください。

10.2.10.13 Oracle (JDBC) への接続

JDBC インターフェイスを使用して、クライアントマシンから、Oracle データベースサーバーへの接続の方法を説明します。接続は、Oracle Web サイトから使用することのできる、**Oracle Instant Client Package (Basic)** を使用する、Java 接続として作成されます。接続の型の利点は、Oracle Instant Client Package により与えられる Java 環境と .jar ライブラリのみを必要とし、更に複雑なデータベースクライアントをインストールする手間を省くことができる必点です。

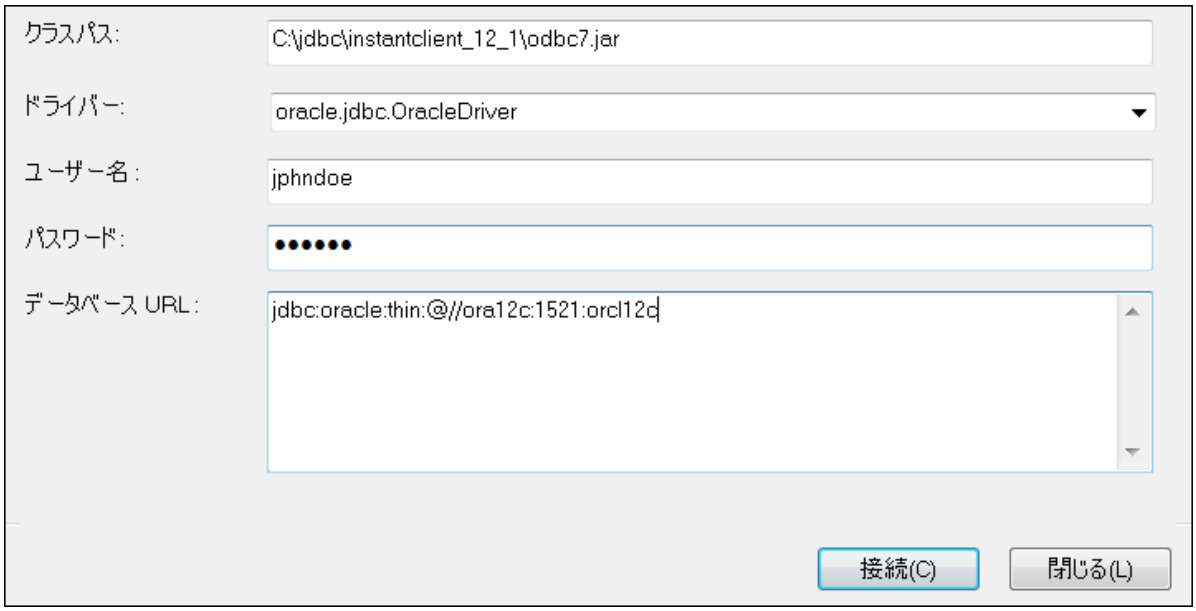
前提条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。StyleVision は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーションオプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- StyleVision (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- **Oracle Instant Client Package (Basic)** が使用中のオペレーティングシステムで使用可能である必要があります。パッケージは、Oracle の公式 Web サイトからダウンロードすることができます。このサンプルは、Windows 32-ビットのための Oracle Instant Client Package バージョン 12.1.0.2.0 、そして結果的に Oracle JDK 32-ビットを使用します。

- データベース接続の詳細があります: ホスト、データベースパス、またはエイリアス、ユーザー名、およびパスワード。

クライアントパッケージを使用して Oracle に接続する:

1. データベース接続ウィザードを開始します。
2. 「JDBC 接続」をクリックします。
3. 「クラスパス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます: **C:\jdbc\instantclient_12_1\ojdbc7.jar**。.jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成)。
4. 「ドライバー」ボックスから、**oracle.jdbc.OracleDriver** または **oracle.jdbc.driver.OracleDriver** を選択します。このエントリは、有効な .jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます(前のステップを参照してください)。
5. データベースへのユーザー名とパスワードを対応するテキストボックスに入力します。



6. ハイライトされた値を、使用するデータベースサーバーに適用する値と置き換えて、データベース URL テキストボックス内にデータベースサーバーへの接続文字列を入力します。

```
jdbc:oracle:thin:@//host:port:service
```

7. 「接続」をクリックします。

10.2.10.14 Oracle (ODBC) への接続

このサンプルでは、ローカルオペレーティングシステム上にインストールされている Oracle データベースクライアントを介して、ネットワークマシン上で StyleVision から Oracle データベースサーバーへの接続のよく使用されている状況について説明されています。

このサンプルでは、StyleVision 内のデータベース接続ウィザードを使用して、ODBC データソース(DSN) のセットアップについて説明されています。既に DSN が作成されている場合は Windows 内の **ODBC データソース管理者**から直接作成しプロンプトされたら選択する事も可能です。ODBC データソースの詳細に関しては、ODBC 接続のセットアップを参照してください。

必要条件:

- (ODBC Oracle ドライバーを含む) Oracle データベースクライアントが使用中のオペレーティングシステムでインストール、および構成されていなければなりません。Oracle データベースクライアントのインストールおよび構成方法は Oracle ソフトウェアに付随するドキュメントを参照してください。
- **tnsnames.ora** ファイルのパスは Oracle ホームディレクトリがインストールされている場所によります。Oracle データベースクライアント 11.2.0 に関しては、デフォルト Oracle ホームディレクトリのパスは以下の通りです:

```
ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = server01)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SID = orcl)
      (SERVER = DEDICATED)
    )
  )
```

tnsnames.ora ファイルのパスは Oracle ホームディレクトリがインストールされている場所によります。Oracle データベースクライアント 11.2.0 に関しては、デフォルト Oracle ホームディレクトリのパスは以下の通りです:

```
C:\app\username\product\11.2.0\client_1\network\admin\tnsnames.ora
```

接続の詳細を貼り付け保存する、または使用可能であれば Oracle *ネット構成アシスタント*ウィザード を実行することにより **tnsnames.ora** ファイルに新規エントリを追加することができます。

ODBC を使用しての Oracle への接続:

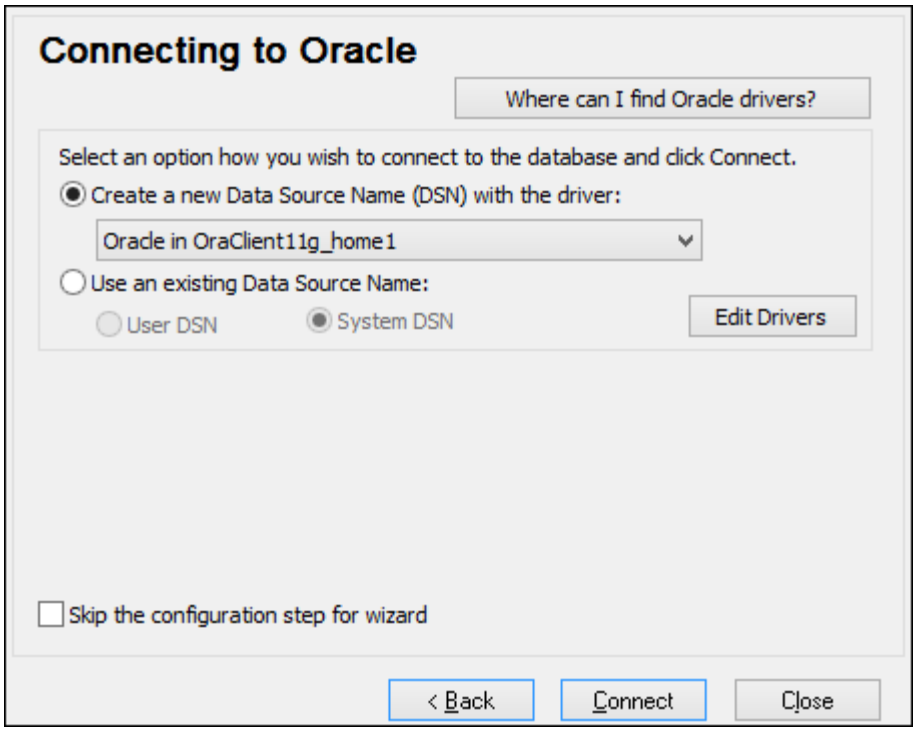
1. データベース接続ウィザードを開始します.
2. **[Oracle (ODBC / JDBC)]** を選択して、「次へ」をクリックします。

3. **[ODBC]** を選択します。

4. [ドライバーの編集] をクリックします。

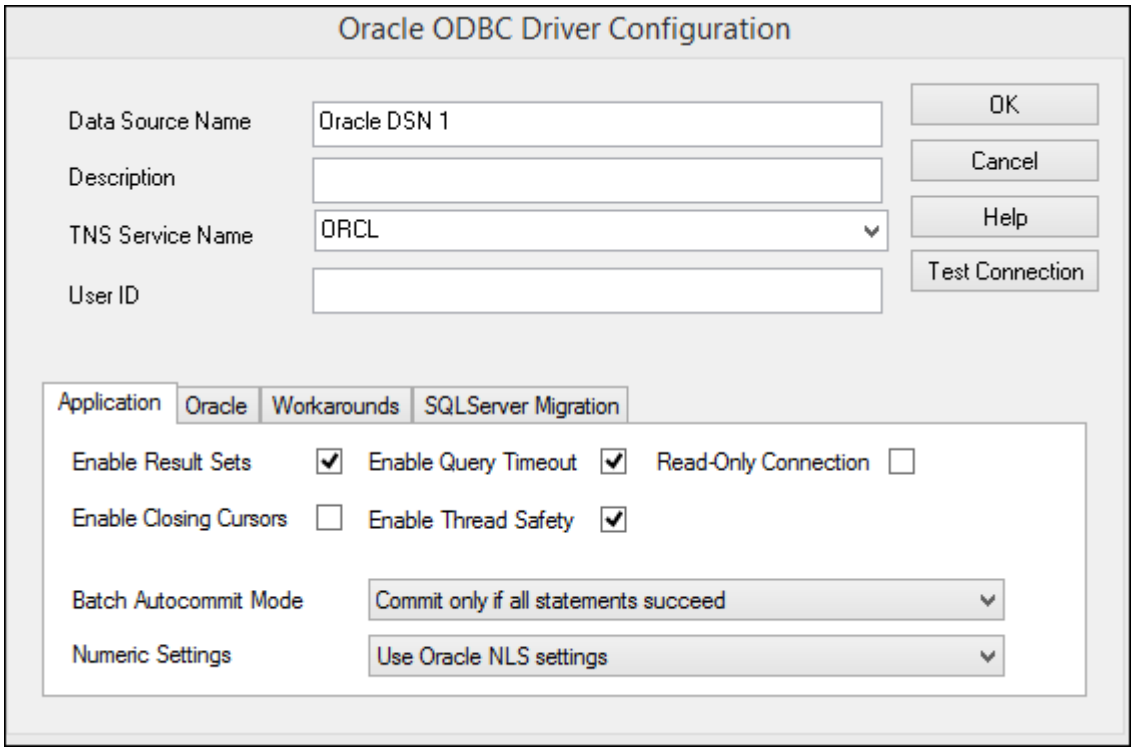
5. 使用するOracleドライバーを選択します（このサンプルでは**Oracle in OraClient11g_home1**です）。Oracle クライアントをインストール後システムで使用可能なOracleドライバーのリストを表示します。
6. [戻る] をクリックします。
7. [**新しいデータソース（DSN）をドライバーを使用して作成**] を選択して、ステップ4 で選択されたOracleドライバーを選択します。

(C) 2015-2021 Altova GmbH

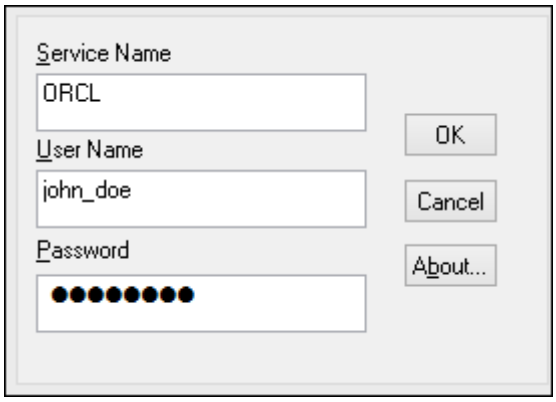


Microsoft ODBC for Oracle という名称のMicrosoft により提供されるドライバーを使用することは避けてください。Microsoft はOracle により提供されるODBC ドライバーを使用することを奨励しています。(以下参照: http://msdn.microsoft.com/en-us/library/ms714756%28v=vs.85%29.aspx)。

8. 「接続」をクリックします。



9. データソース名テキストボックスにデータソースを識別する名前を入力します（このサンプルの場合は**Oracle DSN 1** です）。
10. TNS サービス名ボックスでは、**tnsnames.ora** ファイルで定義されている通りの接続名を入力します。（prerequisites前提条件を参照してください）。このサンプルでは接続名 **ORCL** です。
11. 「OK」をクリックします。



12. データベースのユーザー名とパスワードを入力して、「OK」。をクリックします。

10.2.10.15 PostgreSQL (ODBC) への接続

このトピックではODBC ドライバーを使用した、Windows マシンから PostgreSQL データベースへの接続のサンプル手順について説明されています。Windows からは、 PostgreSQL ODBC ドライバーを入手することができませんので、個別にダウンロードおよびインストールする必要があります。このサンプルでは、公式 Web サイトからダウンロードされたpsqlODBC ドライバー（バージョン 11.0）を使用しています。（データベースドライバー概要も参照してください）。

メモ　（ODBC ドライバー無しで直接 PostgreSQL データベースサーバーに接続するには、次を参照してください：PostgreSQL 接続のセットアップ。

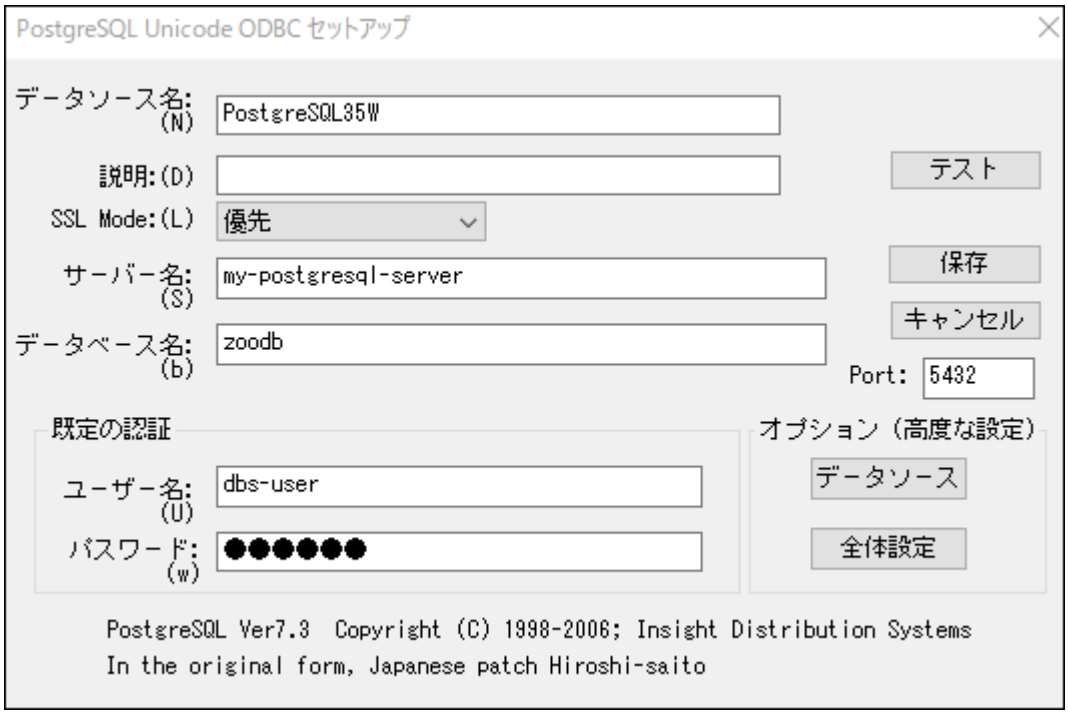
前提条件:

- *psqlODBC* ドライバーを使用中のオペレーティングシステムにインストールする必要があります。
- データベース接続の詳細：ホスト、データベース、ユーザー名およびパスワード

ODBC を使用した PostgreSQL への接続をセットアップする方法:

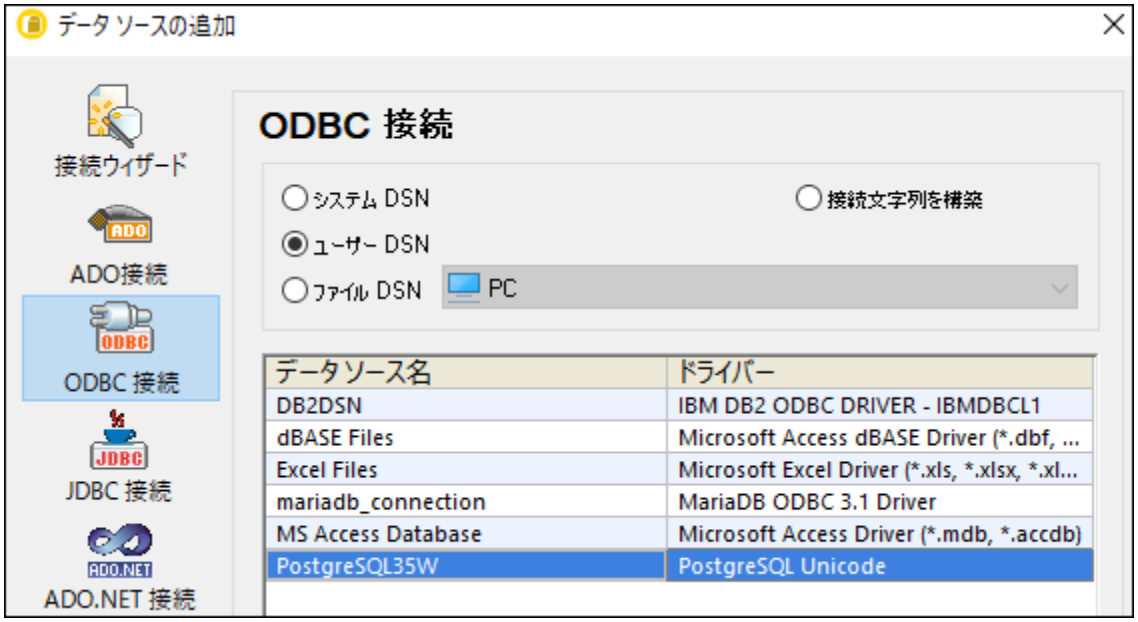
1. データベース接続ウィザードを開始します。
2. **PostgreSQL (ODBC)** をクリックします。
3. **ユーザー DSN** オプションを選択します。
4. 「**新規の DSN の作成**」 をクリックし、ドロップダウンリストからドライバーを選択します。PostgreSQL ドライバーがリスト内で使用できない場合、上記の必要条件で説明されているとおり、PostgreSQL ODBC ドライバーがオペレーティングシステム上にインストールされていることを確認してください。

5. 「**ユーザー DSN**」をクリックします。



6. データベース資格情報を入力し、「保存」をクリックします（これらはデータベースオーナーにより提供されています）。

接続はODBC 接続のリスト内で使用できるようになります。データベースに接続するには、接続をダブルクリックし、「接続」をクリックします。



10.2.10.16 Progress OpenEdge (JDBC) への接続

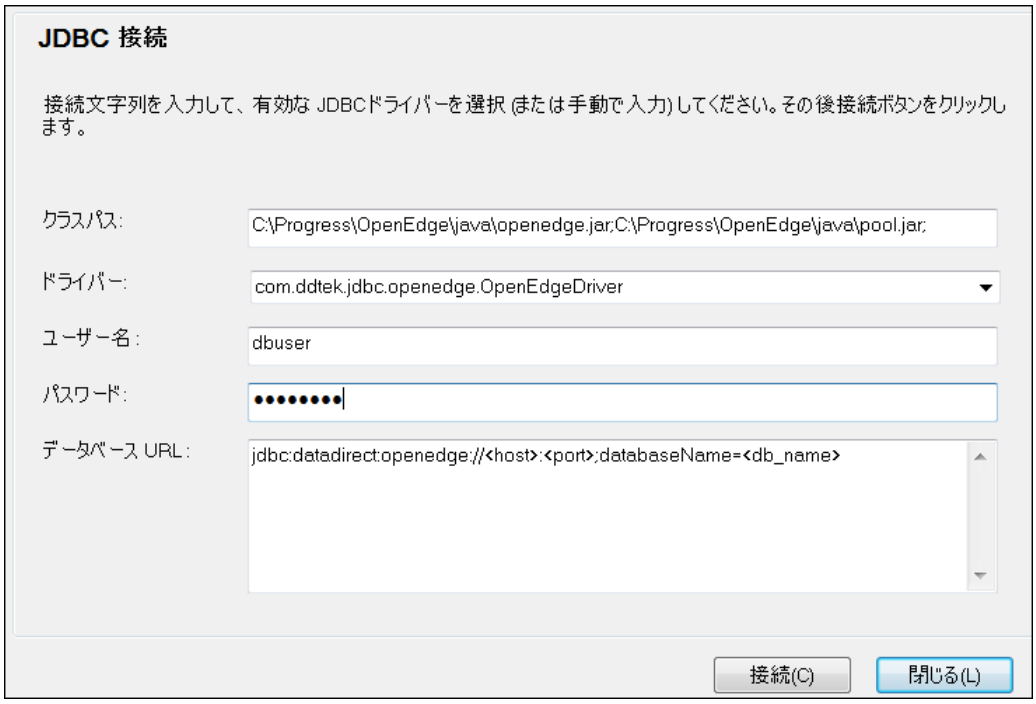
JDBC を使用して Progress OpenEdge 11.6 データベースサーバーへの接続するためのサンプルの手順が説明されています。

前提条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。StyleVision は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- StyleVision (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- オペレーティングシステムの `PATH` 環境変数は、JRE または JDK インストールの `bin` ディレクトリへのパスを含んでいる必要があります。例、`C:\Program Files (x86)\Java\jre1.8.0_51\bin`.
- Progress OpenEdge JDBC ドライバーが使用中のオペレーティングシステムで使用可能である必要があります。このサンプルでは、JDBC 接続は、**C:\Progress\OpenEdge\java** に OpenEdge SDK のインストールの一部として存在する **openedge.jar** と **pool.jar** ドライバーコンポーネントファイルにより与えられています。.
- データベース接続の詳細があります: ホスト、データベース パス、またはエイリアス、ユーザー名、およびパスワード。

JDBC を使用して OpenEdge へ接続する:

1. データベース接続ウィザードを開始します.
2. 「**JDBC 接続**」をクリックします。
3. 「クラス パス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイル パスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます`C:\Progress\OpenEdge\java\openedge.jar;C:\Progress\OpenEdge\java\pool.jar;`。.jar ファイル パスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラス パス」テキスト ボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成)。
4. 「ドライバー」ボックスから、**com.ddtek.jdbc.openedge.OpenEdgeDriver** を選択します。このエントリは、有効な .jar ファイル パスが、「クラス パス」テキスト ボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます (前のステップを参照してください)。



5. データベースへのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値を、使用するデータベースサーバーに適用する値と置き換えて、データベースURL テキストボックス内にデータベースサーバーへの接続文字列を入力します。

```
jdbc:datadirect:openedge://host:port;databaseName=db_name
```

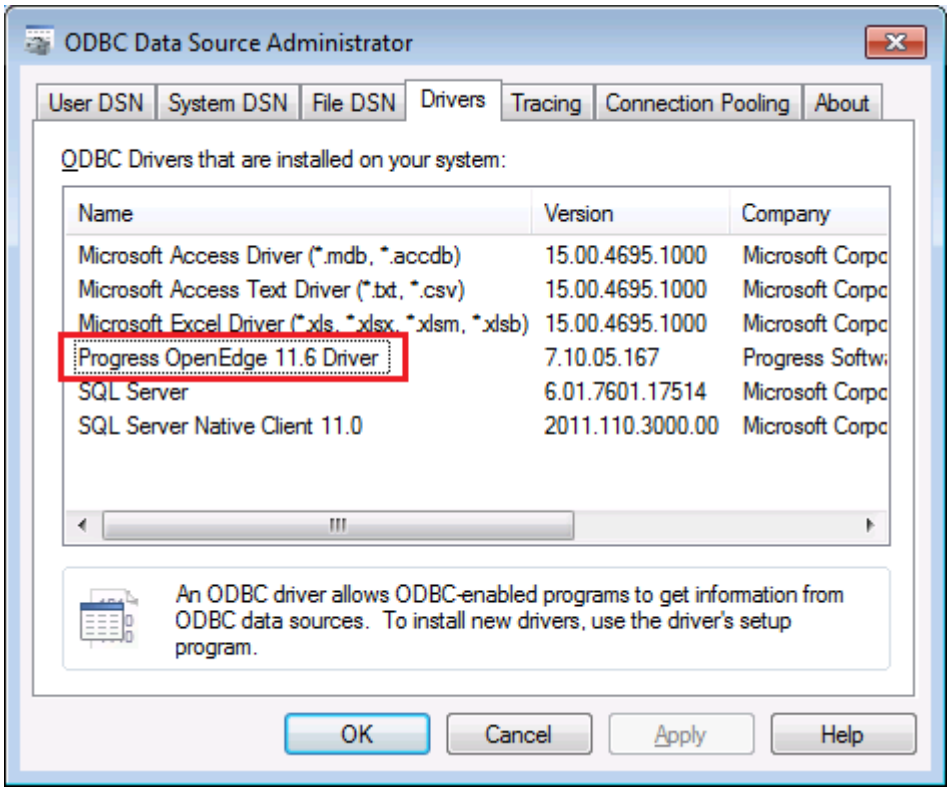
7. 「接続」をクリックします。

10.2.10.17 Progress OpenEdge (ODBC) への接続

このサンプルではProgress OpenEdge 11.6 ODBC ドライバーを使用して、Progress OpenEdge データベースサーバーへの接続の簡単な手順について説明されています。

前提条件:

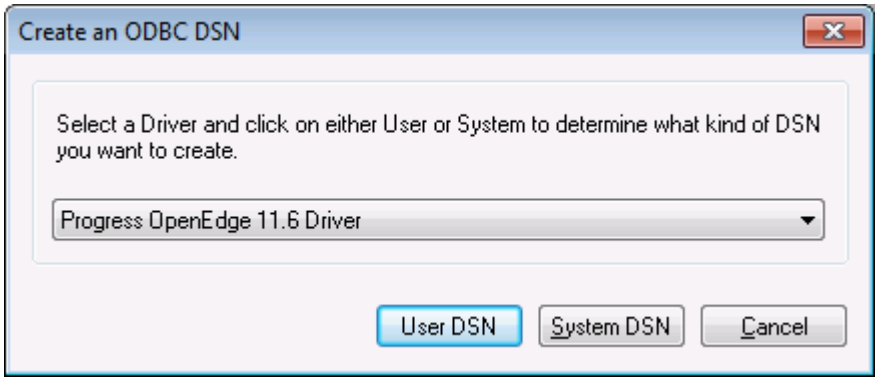
- *Progress OpenEdge ドライバーのための ODBC コネクタ*が使用中のオペレーティングシステムにインストールされている必要があります。Progress OpenEdge ODBC ドライバーは、ベンダーのW eb サイトからダウンロードすることができます(次も参照: データベースドライバーの概要)。StyleVision の32 ビットバージョンを作動する場合は、32 ビット ドライバーをダウンロードし、64 ビットバージョンを作動する場合は、64 ビット ドライバーをダウンロードしてください。インストールの後、ODBC ドライバーが使用中のマシンで使用することができるかを確認してください(次も参照: 使用することのできるODBC ドライバーを確認する)。



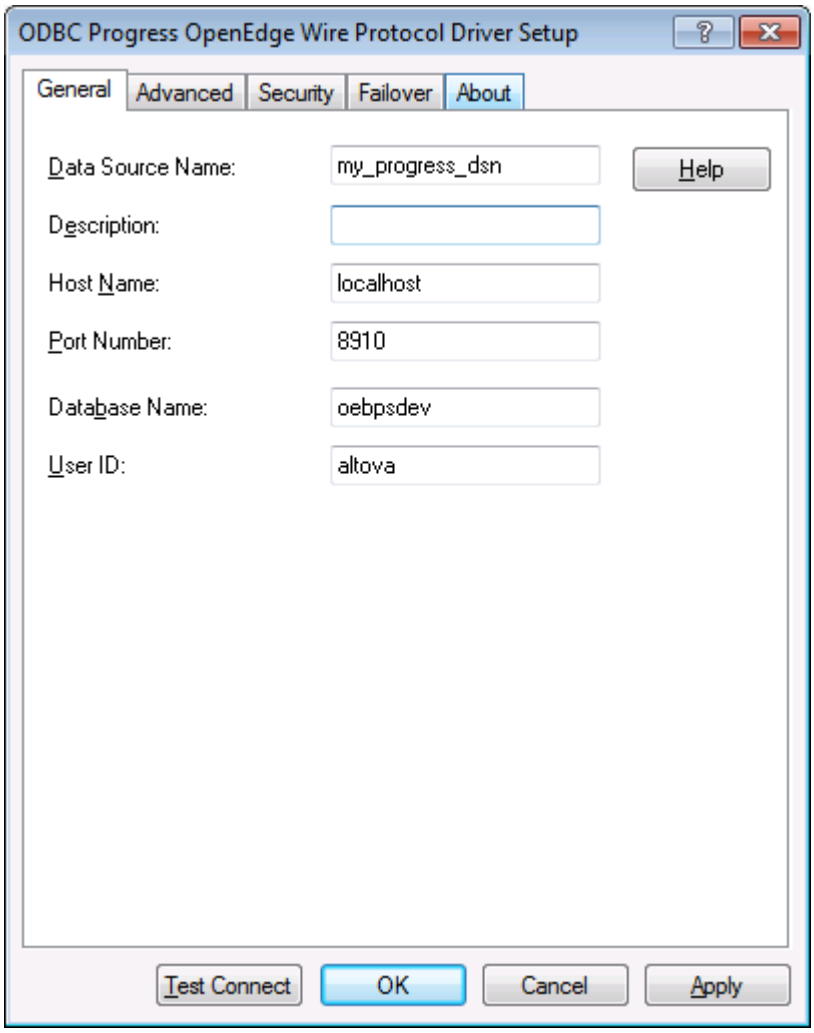
- データベース接続の詳細があります: ホスト名、ポート番号、データベース名、ユーザーID、およびパスワード。

ODBC を使用して Progress OpenEdge に接続する:

1. データベース接続ウィザードを開始します.
2. 「ODBC 接続」をクリックします。
3. 「ユーザーDSN」をクリックします。(または、次の手順が類似している「システムDSN」、または、「ファイルDSN」をクリックします)。
4. 「追加」 をクリックします。
5. 「Progress OpenEdge ドライバー」をリストから選択し、「ユーザーDSN」(または、適用できる場合は、「システムDSN」)をクリックします。

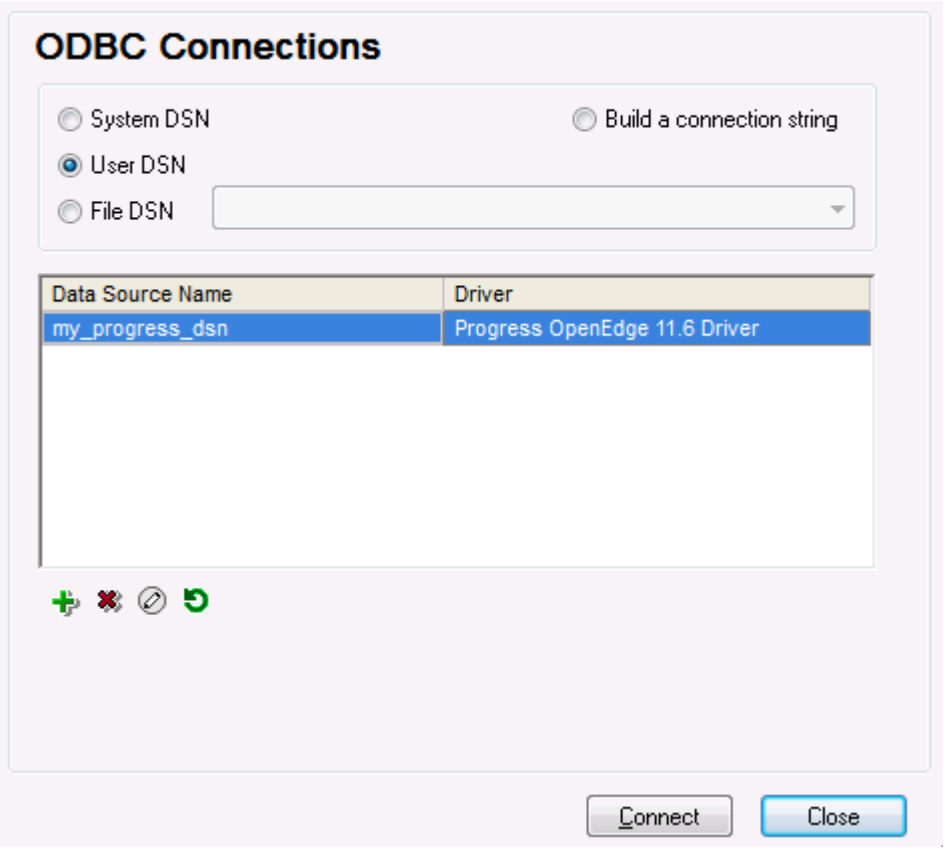


6. データベース接続の資格情報（データベース、サーバー、ポート、ユーザー名、パスワード）を入力し、「OK」をクリックします。入力されたデータを保存する前に、接続を検証するには、「**接続をテストする**」をクリックします。



7. 「OK」をクリックします。新規データソースがODBC データソースのリストに表示されます。

(C) 2015-2021 Altova GmbH



8. 「接続」をクリックします。

10.2.10.18 Sybase (JDBC) への接続

このサンプルでは、JDBC を使用して Sybase データベースサーバーに接続する手順について説明されています。

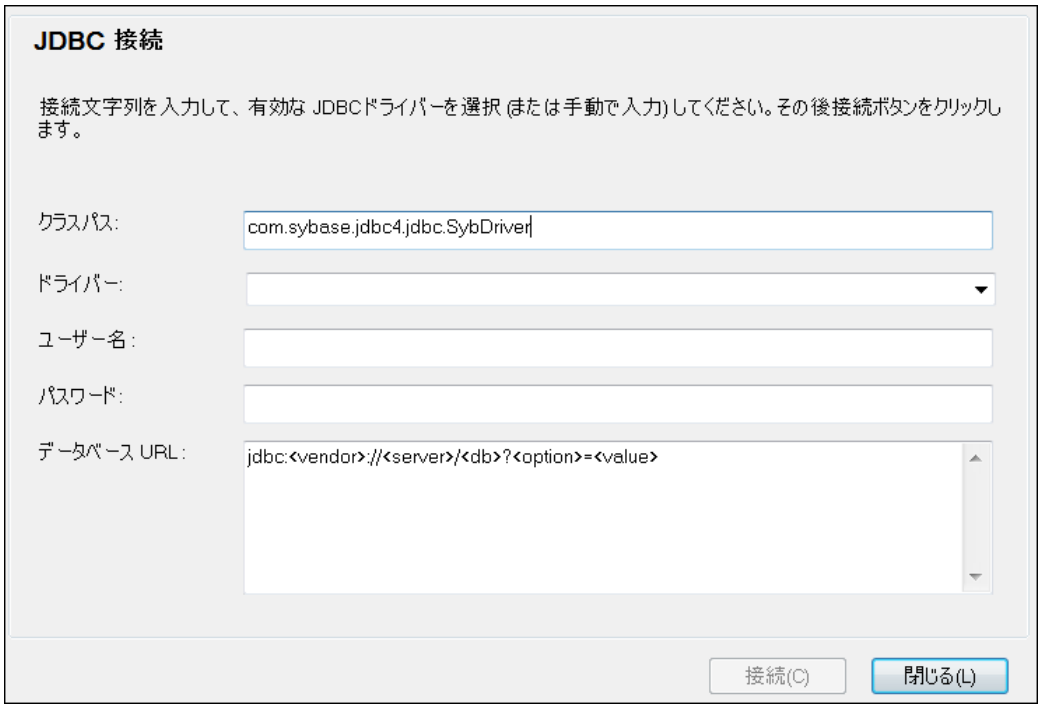
前提条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。StyleVision は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーション オプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- StyleVision (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。
- Sybase *jConnect* コンポーネントがオペレーティングシステムにインストールされている必要があります (このサンプルでは *jConnect 7.0* が使用されており、*Sybase Adaptive Server Enterprise PC Client* インストールの一部としてインストールされています)。データベースクライアントのインストール命令に関しては、Sybase ドキュメントを参照してください。
- 次のデータベース接続詳細が存在します: ホスト、データベース、ポート、ユーザー名、および、パスワード。

JDBC を使用して Sybase へ接続する:

1. データベース接続ウィザードを開始します。
2. 「JDBC 接続」をクリックします。

3. 「クラスパス」の横にデータベースへの接続を与える .jar ファイルへのパスを入力します。必要であれば、セミコロンで区切られた .jar ファイルパスのリストを入力することもできます。このサンプルでは、必要とされる .jar ファイルは次のパスを使用して見つけることができます: **C:\sybase\jConnect-7_0\classes\jconn4.jar**。.jar ファイルパスをオペレーティングシステムの CLASSPATH 環境変数に追加した場合、「クラスパス」テキストボックスを空白のままにすることができます。(次も参照: CLASSPATH の構成)。
4. 「ドライバー」ボックスから、 を選択します。「ドライバー」ボックスから、**com.sybase.jdbc4.jdbc.SybDriver** を選択します。このエントリは、有効な .jar ファイルパスが、「クラスパス」テキストボックス内、または、オペレーティングシステムの CLASSPATH 環境変数内で検出された場合のみ使用することができます(前のステップを参照してください)。



5. データベースのユーザー名とパスワードを対応するテキストボックスに入力します。
6. ハイライトされた値をデータベースサーバーに適用することのできる値と置き換えて、データベース URL テキストボックス内のデータベースサーバーの接続文字列を入力します。

```
jdbc:sybase:Tds:hostName:port/databaseName
```

7. 「**接続**」をクリックします。

10.2.10.19 Teradata (JDBC) への接続

このサンプルでは、JDBC を使用した Teradata データベースへの接続方法について説明されています。

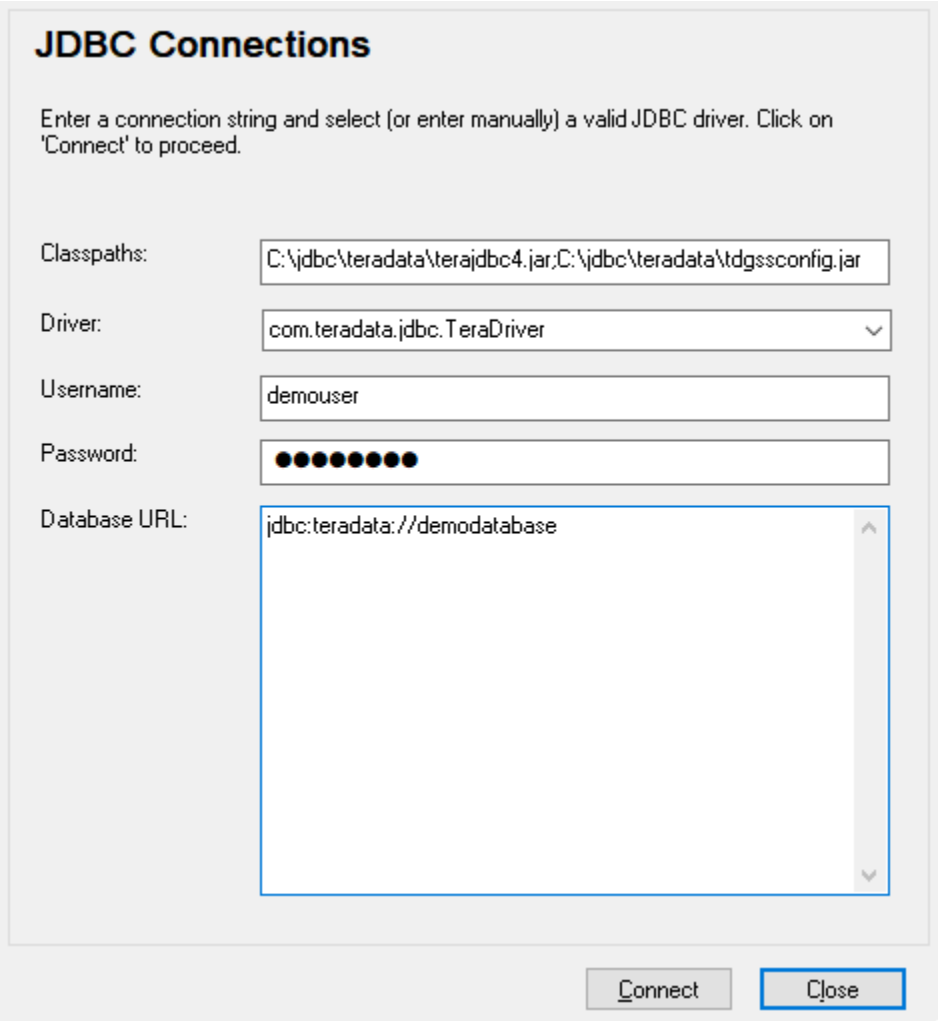
必要条件:

- JRE (Java Runtime Environment) または Java Development Kit (JDK) がインストールされている必要があります。これは Oracle JDK または Oracle OpenJDK などのオープンソースビルドであることができます。StyleVision は次のロケーションから Java Virtual Machine (JVM) へのパスを次の順序で決定します: a) アプリケーションオプション内で設定したカスタム JVM パス b) Windows レジストリ内で見つかった JVM パス。c) `JAVA_HOME` 環境変数。
- StyleVision (32-ビット、64-ビット) のプラットフォームが JRE/JDK のプラットフォームに一致することを確認してください。

- JDBC ドライバー（データベースへの接続を提供する1つまたは複数の.jar ファイル）がオペレーティングシステム上で使用できる必要があります。このサンプルでは、Teradata JDBC ドライバー16.20.00.02 が使用されます。詳細に関しては、次を参照してください：http://downloads.teradata.com/download/connectivity/jdbc-driver。
- 次のデータベース接続詳細が存在します：ホスト、データベース、ポート、ユーザー名、および、パスワード。

JDBC を使用した Teradata への接続の方法:

1. データベース接続ウィザードを開始します。
2. 「JDBC 接続」をクリックします。
3. 「クラスパス」の横に、データベースへの接続を与える.jar ファイルへのパスを入力します。必要な場合、セミコロンで区切られた.jar ファイルパスのリストを入力することができます。このサンプルでは、.jar ファイルは、次のパスの場所に存在します：**C:\jdbc\teradata**。オペレーティングシステムのCLASSPATH 環境変数に.jar ファイルパスが追加されている場合、「クラスパス」テキストボックス空のままにすることができます（次も参照してください：CLASSPATH の構成）。
4. 「ドライバー」ボックス内で、**com.teradata.jdbc.TeraDriver** を選択します。有効な.jar ファイルパスが「クラスパス」テキストボックスで、または、オペレーティングシステムのCLASSPATH 環境変数内に存在すると、使用することができます（前のステップを参照してください）。



5. ユーザー名とパスワードを対応するテキストボックスのデータベース内に入力します。
6. ハイライトされた値をデータベースサーバーと適用することのできる値を置換え、データベースURL テキスト内にデータベースサーバーに接続文字列を入力します。

```
jdbc:teradata://databaseServerName
```

7. 「接続」をクリックします。

10.2.10.20 Teradata (ODBC) への接続

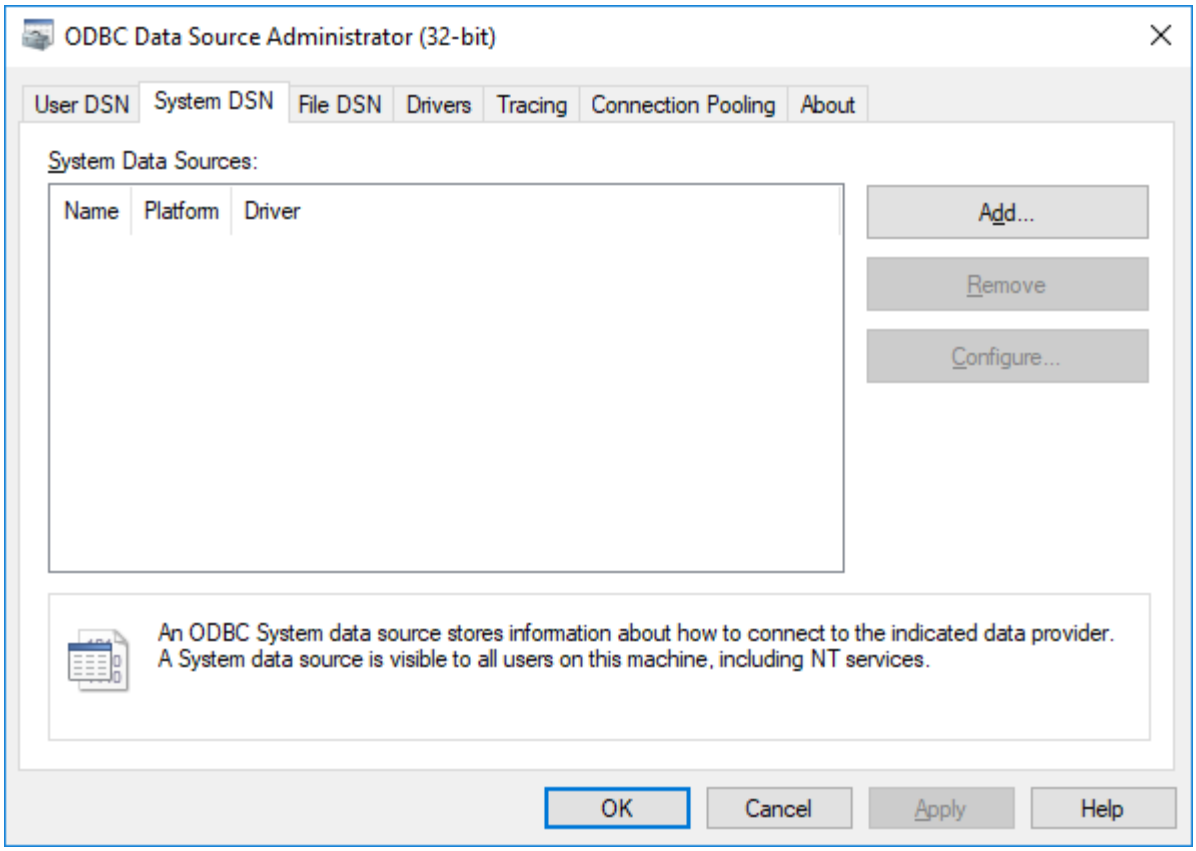
このサンプルでは、ODBC を使用した SQL Teradata データベースへの接続方法について説明されています。

前提条件:

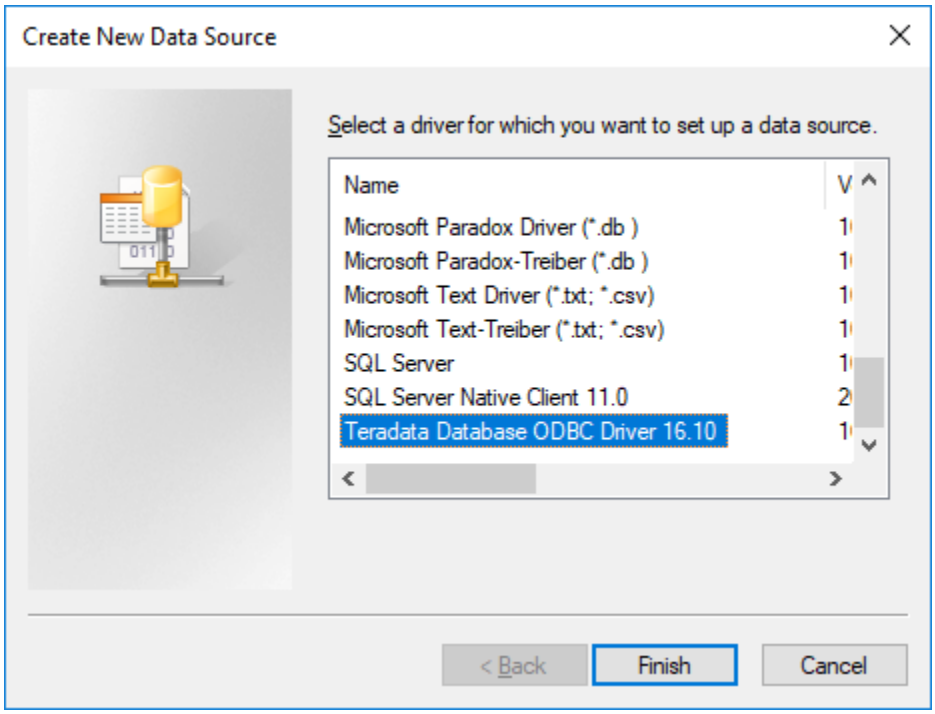
- Teradata ODBC ドライバーがインストールされる必要があります (参照: http://downloads.teradata.com/download/connectivity/odbc-driver/windows。このサンプルは、Windows バージョン 16.20.00 のために ODBC ドライバーを使用しています。
- 次のデータベース接続詳細が存在します: ホスト、ユーザー名、および、パスワード。

ODBC を使用した Teradata への接続の方法:

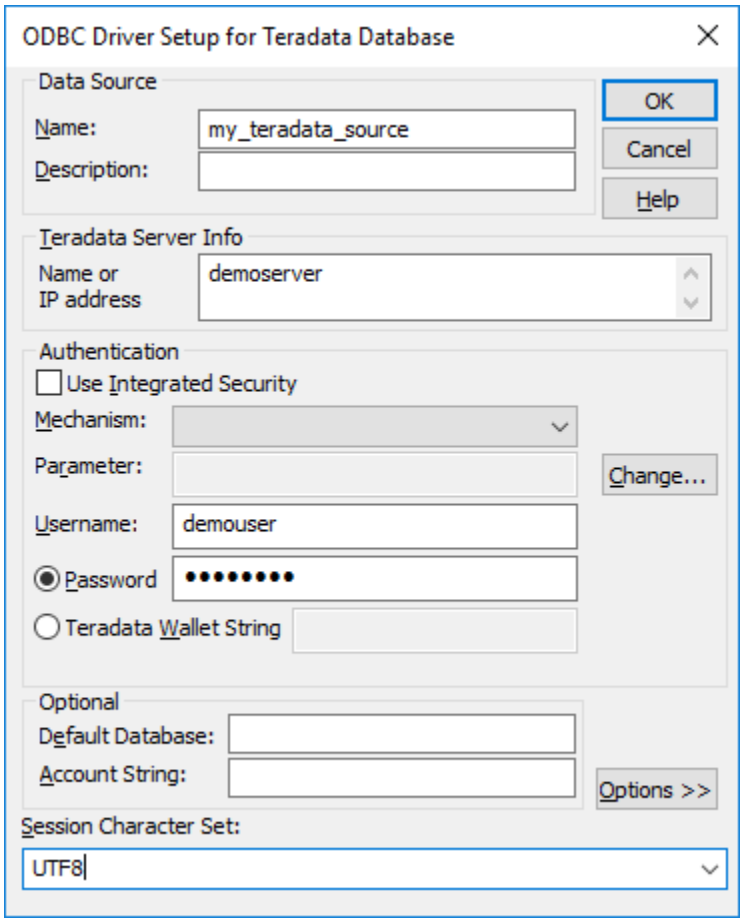
1. **Windows** キーを押して、「ODBC」を入力し、「ODBC データソースのセットアップ (32-ビット)」を提案されるリストから選択します。64-ビット ODBC ドライバーが存在する場合、「ODBC データソースのセットアップ (64-ビット)」を選択し、次のステップ内で 64-ビット StyleVision を使用します。



2. 「System DSN」タブをクリックして、「追加」をクリックします。

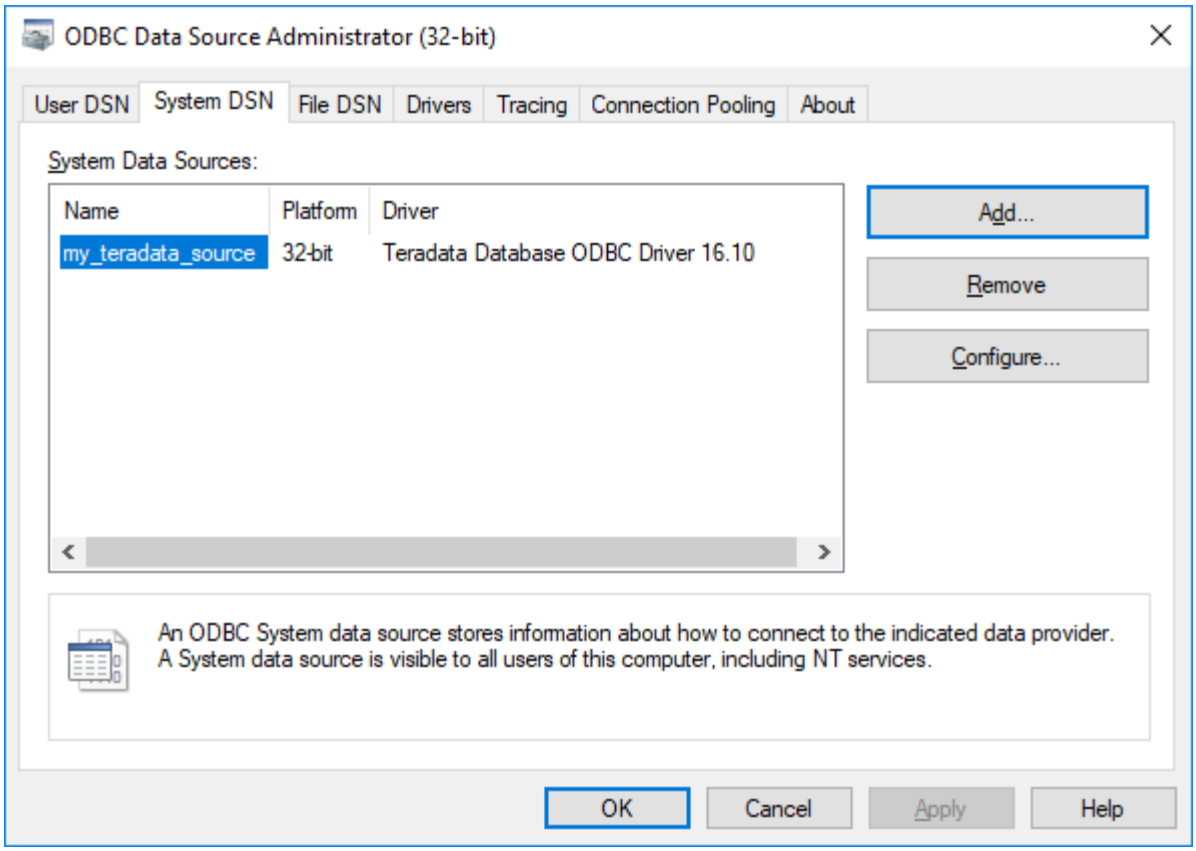


3. 「Teradata データベース ODBC ドライバー」を選択して、「終了」をクリックします。

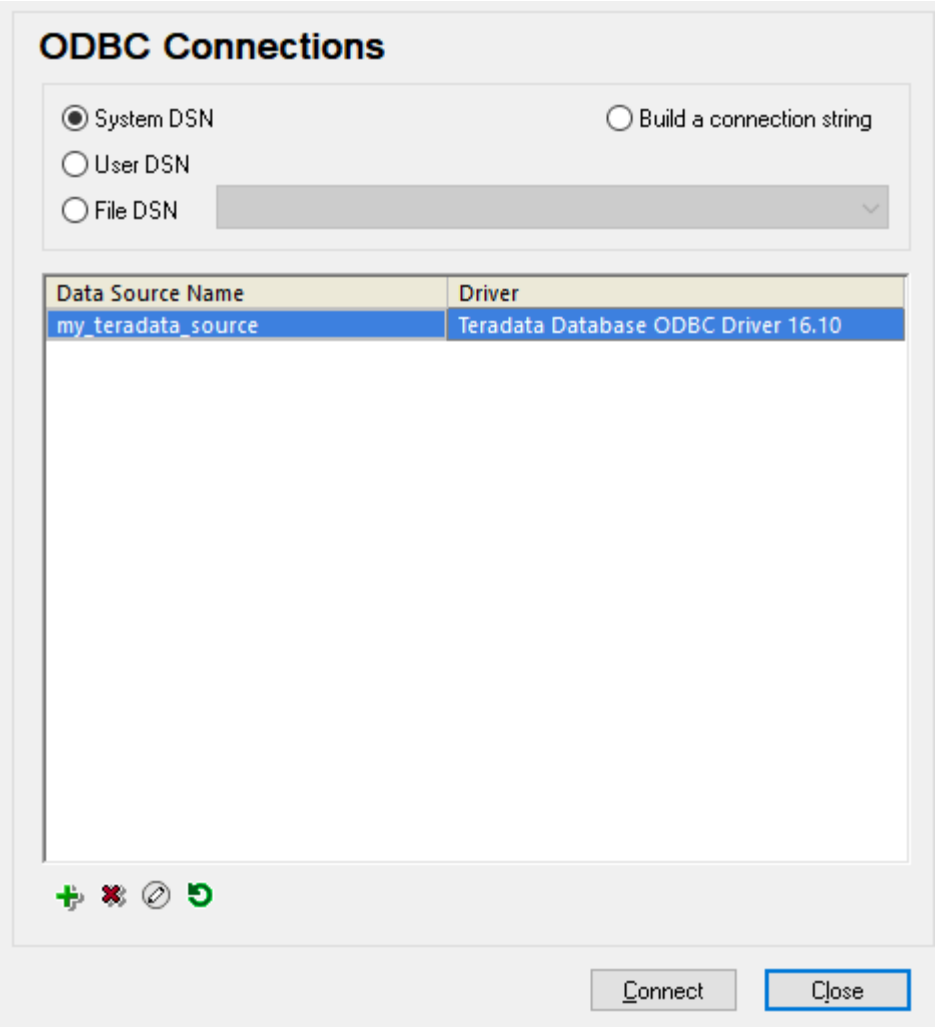


4. 名前を入力し、任意で、このODBC データソースを将来識別できる詳細を与えます。また、データベース接続資格情報（データベースサーバー、ユーザー、パスワード）を入力し、オプションでデータベースを選択します。
5. 「OK」をクリックします。データソースがリスト内に表示されます。

(C) 2015-2021 Altova GmbH



6. StyleVision を作動し、データベース接続ウィザードを開始します。
7. 「ODBC 接続」をクリックします。



8. 「**System DSN**」をクリックし、前に作成されているデータソースを選択し、「**接続**」をクリックします。

メモ　次のエラーが発生した場合: 「ドライバーが無効として返されました（または返しませんでした）SQL_DRIVER_ODBC_VER: 03.80」「ドライバーが無効と返されました」。ODBC クライアントへのパスが使用中のシステムのPATH 環境変数内に存在することを確認してください（例えば、このロケーションにインストールされている場合、**C:\Program Files\Teradata\Client\16.10\bin** ）。このパスが不足している場合、手動で追加します。

10.3 DB Data Selection

SPS で使用されるスキーマならびにXML データを選択する際には、使用するデータベースが(Microsoft Access のような)非 XML DB か(IBM DB2 バージョン 9 以降のような)XML DB であるかにより、1つ以上のテーブル、セル、またはスキーマを選択することになります。ここでは、DB への接続直後に行われるスキーマならびにXML データの選択を取り扱います。

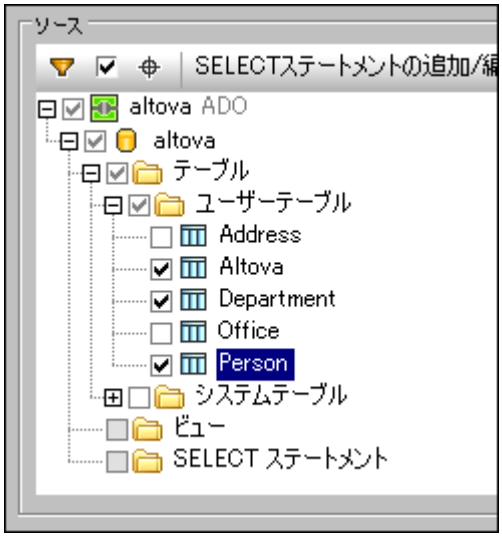
- 非 XML データベースの場合、SPS のデザインに使用されるテーブルを選択することになります。StyleVision により、(i) テーブルの構成をベースにしたスキーマと、(ii) スキーマと選択されたテーブルに含まれているデータをベースにした一時的な XML ファイルが自動的に生成されます。テーブルの選択方法については、DB データの選択 | 非 XML データベースのセクションを参照ください。
- XML データベースの場合、2つのことを行う必要があります。まず目的のXML データが含まれているDB の XML セルを選択します。XML データがSPS の作業用 XML ファイルへロードされます。次に、SPS がベースとするスキーマを選択します。XML データならびにスキーマの選択方法については、DB データの選択 | XML データベースのセクションを参照ください。

10.3.1 非 XML データベース

非 XML データベースに対して接続が作成されると、データベースオブジェクトの挿入ダイアログが表示されます。このダイアログは2つのパーツから構成されます。ダイアログ上部にあるペインにはDB 内にあるテーブルが視覚的に表示され、SPSで使用する目的のテーブルを選択します。ここで選択されたテーブルをベースに、XML スキーマならびにXML データがStyleVision により生成されます。ダイアログ下部にはプレビューペインが表示されており、選択されたテーブルのコンテンツを確認することができます。

ソースペイン

ソースペインにはDB テーブルが視覚的に表示されます(以下のスクリーンショットを参照)。チェックボックスにチェックを入れることで、SPS に必要なテーブルを選択してください。



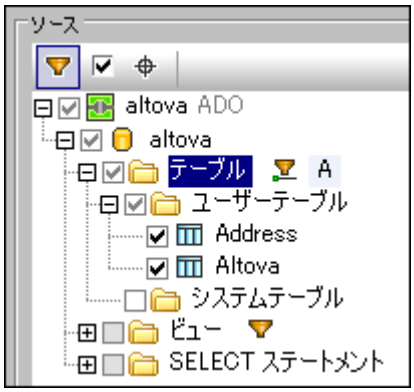
ソースペインのツールバーには3つのアイコンが含まれており、左から順にフォルダーコンテンツのフィルター、チェックされたオブジェクトのみ表示、オブジェクトロケーターとなります。チェックされたオブジェクトのみ表示により、チェックされたテーブルだけを表示することができます。

フォルダーのコンテンツのフィルター

以下の操作により、ソースペインに表示されているオブジェクトのフィルタリングを行うことができます:

1. ソースペインのツールバーにある**フォルダーコンテンツのフィルター**アイコンをクリックします。テーブルフォルダーの隣にフィルターアイコンが表示されます。
2. テーブルフォルダーの隣にあるフィルターアイコンをクリックして、ポップアップメニューからフィルタリングオプションを選択します(以下のスクリーンショットを参照)。ここでは例として**開始文字**を選択します。

3. 表示される入力フィールドにてフィルタリング文字列を入力します(以下のスクリーンショットでは、A という文字がテーブルフォルダーに対して適用されます)。文字の入力と同時にフィルターが適用されます。



オブジェクトロケーター

下記の操作により、ソースペインにあるオブジェクトロケーターを使用することで、特定のデータベースアイテムを発見することができます:

1. ソースペインのツールバーにてオブジェクトロケーターをクリックします。ソースペインの下部にコンボボックスが表示されます。
2. リストの入力フィールドに検索文字列を入力します(例: Altova)。ドロップダウンの矢印をクリックすることで、検索文字列を含む全てのいオブジェクトが表示されます。

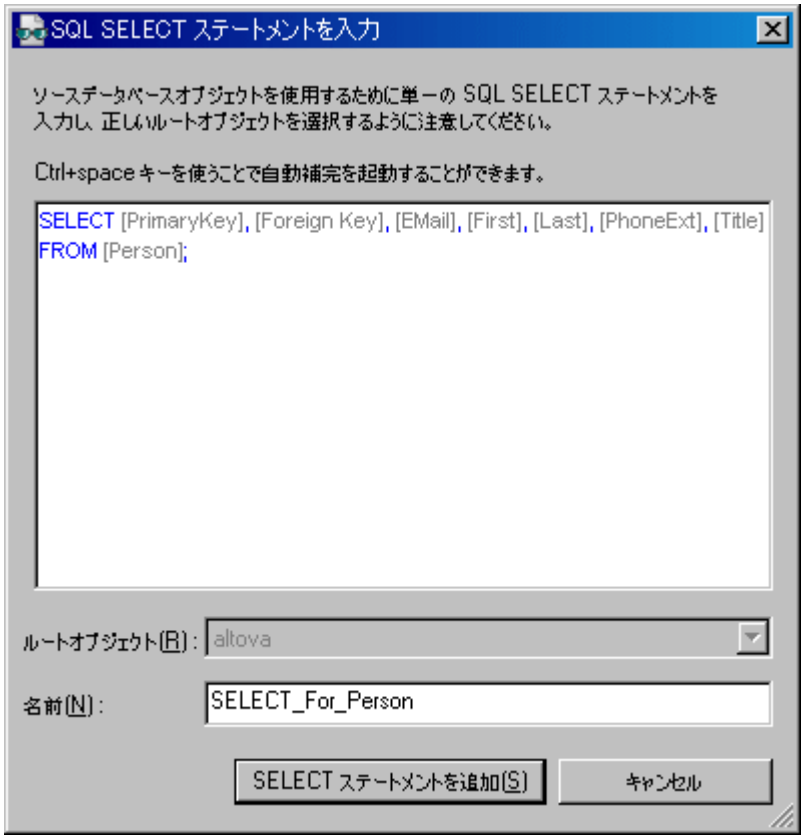
3. リストに表示されたオブジェクトをクリックして、ソースペインにて確認します。

ローカルビューに対してSELECT ステートメントを追加する

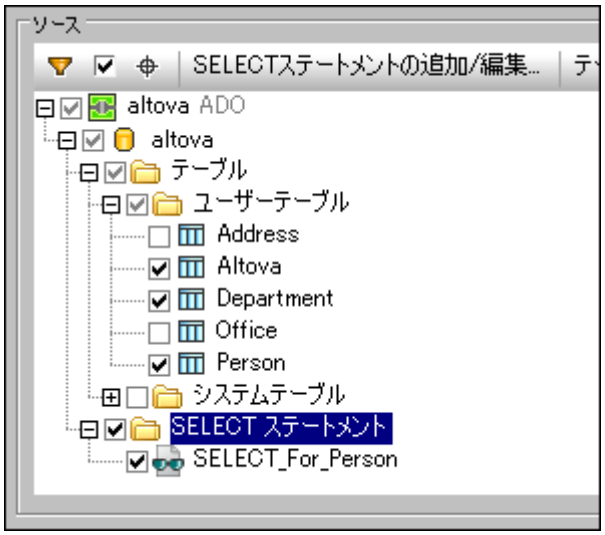
SQL にてSELECT ステートメントにより、ローカルのビューを作成することができます。ローカルのビュー(またはSELECT ステートメント)が含まれるDB の接続からスキーマが生成されると、そのDB に対して生成されたスキーマに各 SELECT ステートメントに対するテーブルが含まれることになります。

以下の操作によりSELECT ステートメントを作成することができます:

1. **SELECT ステートメントの追加/編集**タブをクリックします。SQL SELECT ステートメントの入力ダイアログが表示されます(以下のスクリーンショットを参照)。

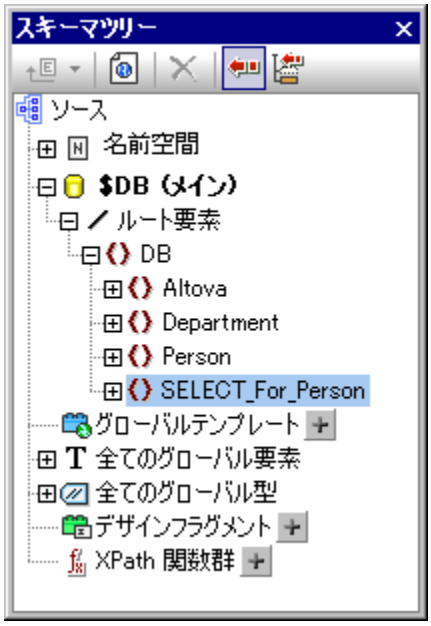


2. SELECT ステートメントを入力します。JDBC を介して Oracle または IBM DB2 データベースへ接続している場合、SELECT ステートメントの終了を表すセミコロンを付け加えないようにしてください。特定のテーブルに対して SELECT ステートメントを作成するには、ソースペインに表示されているテーブルを右クリックして、コンテキストメニューから「**SELECT ステートメントの生成と追加**」を選択してください。
3. **SELECT ステートメントを追加**をクリックしてください。データベースオブジェクトを挿入ダイアログにある SELECT ステートメントのリストに、SELECT ステートメントが追加されます(以下のスクリーンショットを参照)。



メモ JDBC を介して Oracle または IBM DB2 データベースへ接続して SELECT ステートメントの追加/編集コマンドによりデータを取得する場合、SELECT ステートメントに終了を表すセミコロンを付け加えないようにしてください。

データベースオブジェクトを挿入ダイアログにある**完了**ボタンをクリックすると、各 SELECT ステートメントに対してテーブルが作成されます(以下のスクリーンショットを参照)。

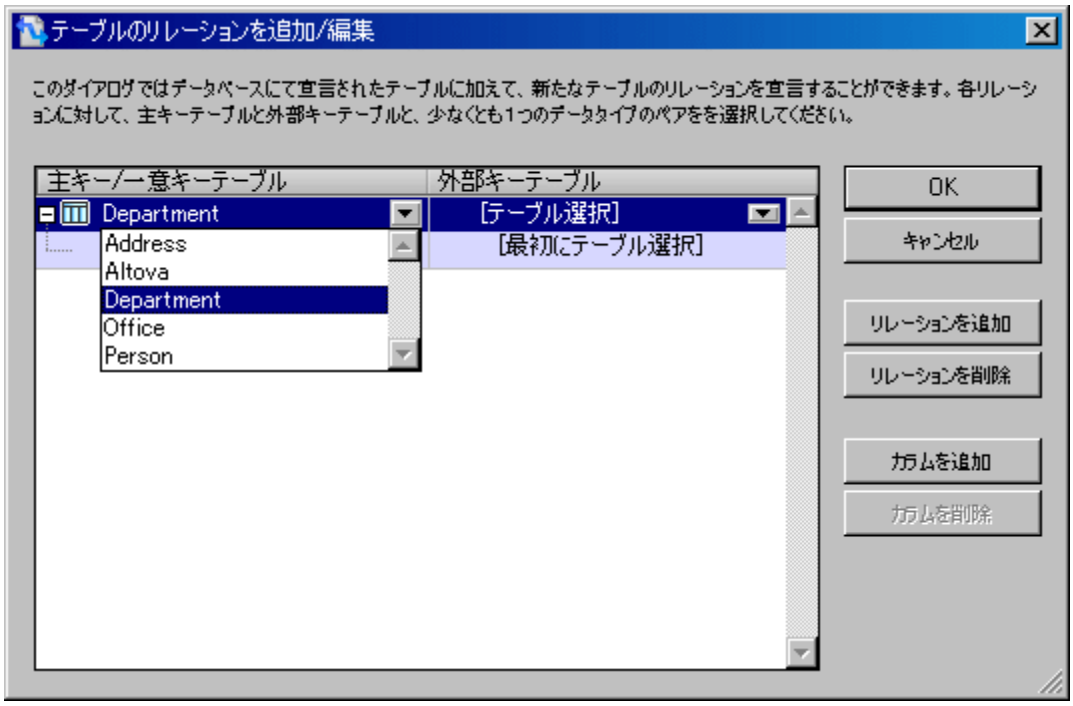


テーブル間におけるローカルリレーション

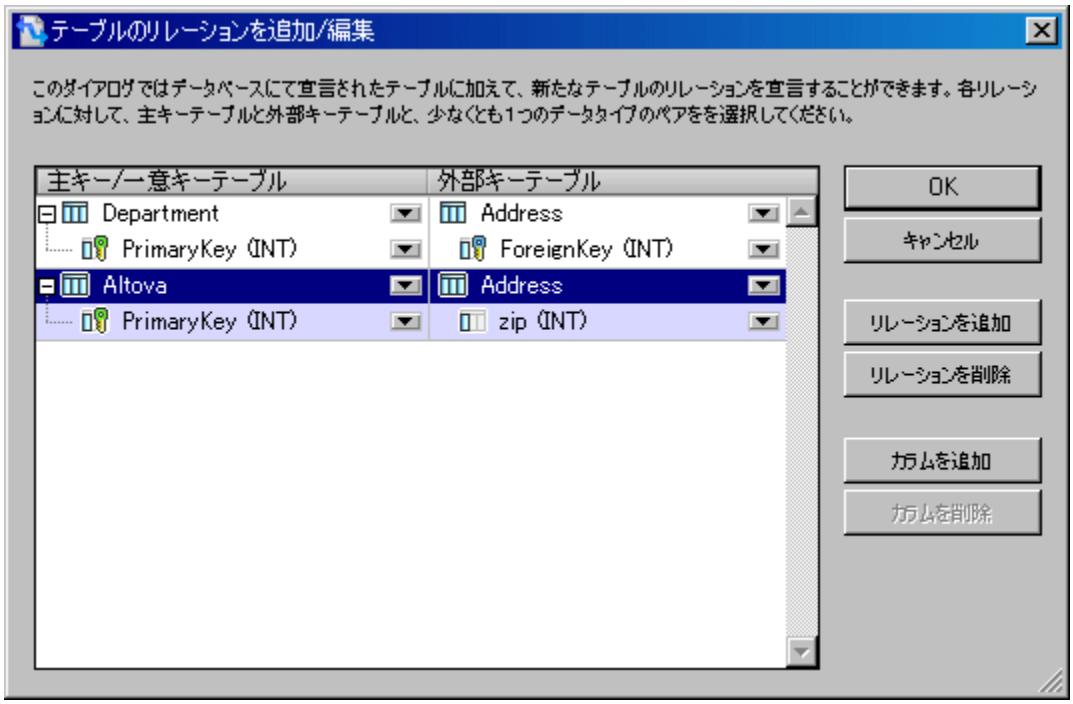
主キー/外部キーのリレーションシップに似たローカルリレーションを2つのテーブル間にて作成することができます。StyleVision 内でリレーションはローカル上で定義されるため、データベースそのものが修正されることはありません。ローカルリレーションは生成されたスキーマに反映されます。

以下の操作によりローカルリレーションを作成することができます:

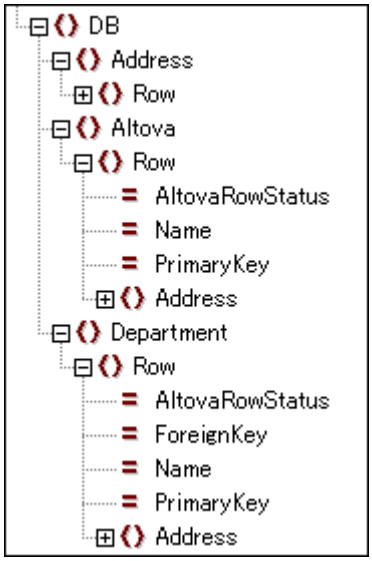
1. データベースオブジェクトを挿入ダイアログにて、テーブルリレーションを追加/編集タブをクリックします。テーブルのリレーションを追加/編集ダイアログが表示されます（以下のスクリーンショットを参照）。
2. **リレーションを追加**ボタンをクリックして、主キー/一意キーカラムにあるテーブル選択コンボボックスのドロップダウンボタンをクリックします（以下のスクリーンショットを参照）。外部キーカラムでもテーブルを選択してください。生成されるリレーションにより、選択された主キー/一意キーカラムにある行のうち、選択された外部キーカラムの行にマッチするものが選択されます。



3. 外部キーテーブルのカラムにマッチさせる主キー/一意キーテーブルのカラムを選択し、外部キーのカラムを選択します。対応するカラムにあるコンボボックスを使用してください。型がマッチしない場合、エラーが表示されます。
4. 必要に応じて新たなローカルリレーションを追加します。必要であれば、ステップ2と3を繰り返します。



5. 「OK」をクリックすることでリレーションの作成を完了します。新たに作成されたリレーションはスキーマの生成時に反映されます。



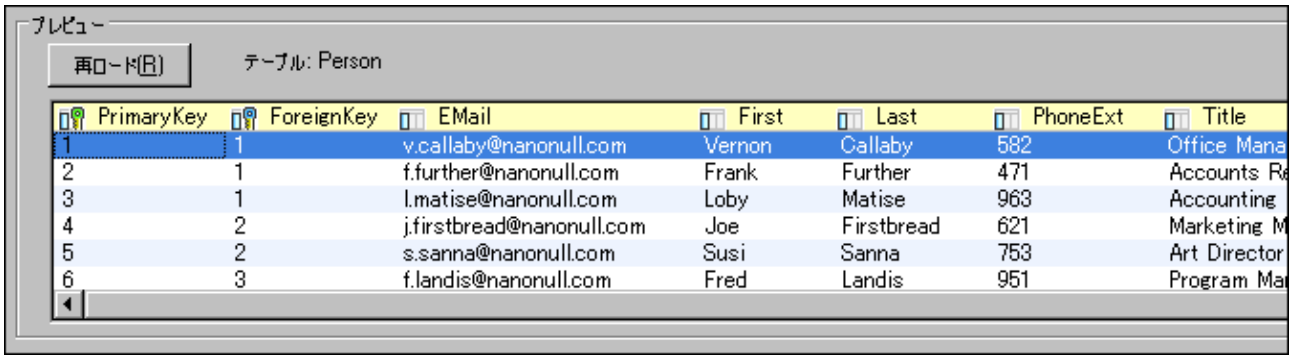
上にある2つのスクリーンショットから、生成されたスキーマでは Altova と Department の両方に Address が含まれていることを確認できます。Department テーブルの主キー値にマッチした外部キー値を持つ Address テーブルの行だけが出力されます。同様に、Altova テーブルの主キー値にマッチした zip カラムの値を持った Address テーブルの行だけが出力されます。

メモ SQL SELECT ステートメント間のローカルリレーションシップはサポートされていません。

(C) 2015-2021 Altova GmbH

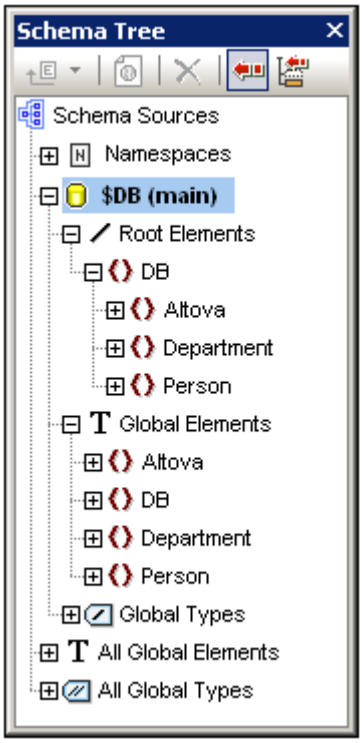
プレビューとプレビューペイン

ソースペインに表示されたテーブルを選択し、プレビューペインにあるプレビューまたは再ロードボタンのどちらからをクリックすることで、テーブルの構造やコンテンツをプレビューすることができます（以下のスクリーンショットを参照）。プレビューペインではテーブルのコンテンツがテーブル形式で表示されます。



DB からXML スキーマならびに作業用 XML ファイルを生成する

SPS にて使用するテーブルを選択し、**完了** ボタンをクリックすることで、XML スキーマの生成とロードが行われます。選択されたDB テーブルの構造に対応したXML スキーマの構造がスキーマツリーウィンドウに表示されます。生成されたスキーマの構造で選択されたテーブルから得られたデータが含まれる作業用 XML ファイルも生成され、出力のプレビューを行うために使用されます。



選択された全てのDB テーブルは、スキーマツリーに表示されるDB ドキュメント要素ならびにグローバル要素以下に作成されます。生成されたXML スキーマの説明については、DB スキーマとDB XML ファイルのセクションを参照ください。DB から既に生成されたXML スキーマをDB フィルターにより修正することはできない点に注意してください。

DB への接続を行い XML スキーマが生成されると、StyleVision に搭載された機能を使ったSPS のデザインを行うことができます。

10.3.2 XML データベース

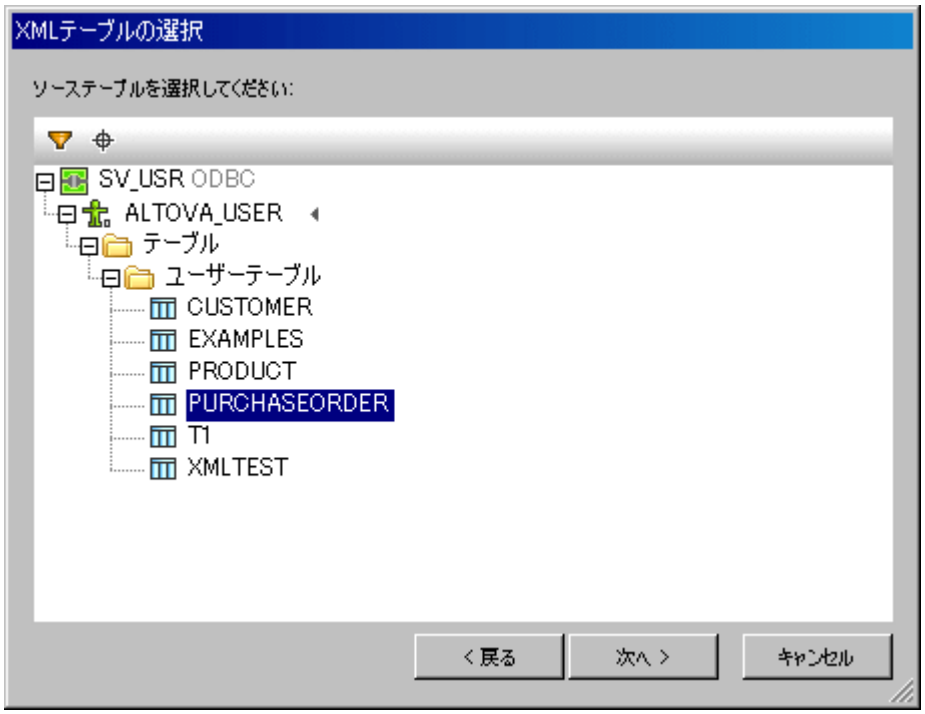
メニューオプションの「ファイル | 新規作成 | DB テーブルの XML カラムから新規作成」コマンドにより表示されるデータベースを開くダイアログにて XML データベース(現在 IBM DB2 XML データベースだけがサポート)への接続を作成した後には、以下にある2つの操作を行う必要があります:

- 目的の XML ドキュメントを含む DB 内にあるセルを選択する。XML ドキュメントが作業用 XML ファイルとして自動的にロードされます。
- SPS に対する XML スキーマを選択する。

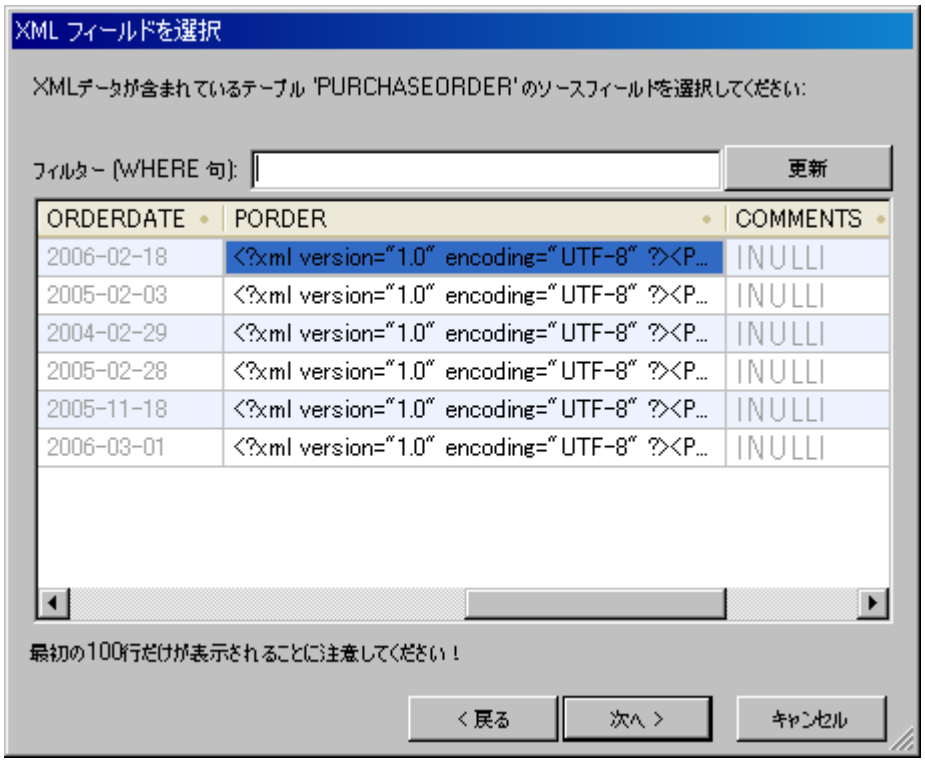
XML セルと作業用 XML ファイルを選択する

IBM DB2 データベースへの接続が確立されると、XML テーブルの選択ダイアログが表示されます(以下のスクリーンショットを参照)。

1. XML テーブルの選択ダイアログにて作業用 XML ファイルとして作成する XML データが含まれているテーブルを選択します。以下のスクリーンショットでは、PURCHASEORDER テーブルが選択されています。



2. 次へをクリックすると XML フィールドを選択ダイアログが表示されます(以下のスクリーンショットを参照)。ペインに表示されている選択のフィルタリングを行うには、SQL WHERE 句を入力し、更新をクリックします。 `WHERE` 句は条件です(`WHERE` キーワードなし、例: `NHE_TEST_TEXT= 'Two'`)



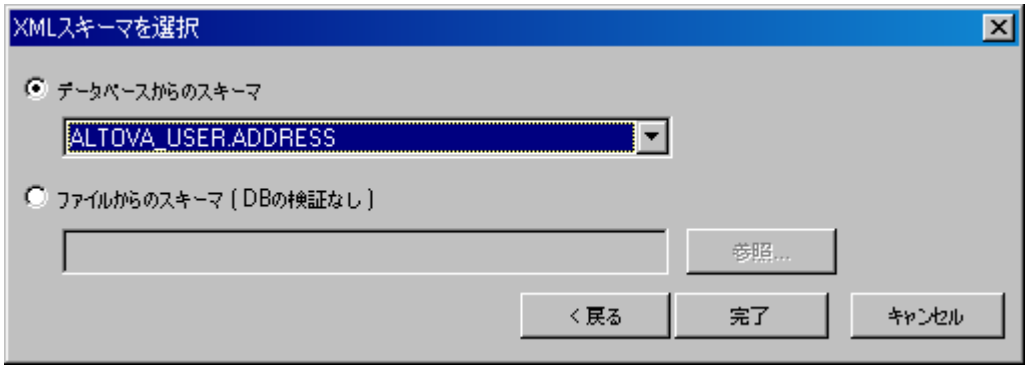
3. 作業用 XML ファイルとして作成する XML データが含まれているセルを選択します。上のスクリーンショットでは選択されたセルが青色でハイライトされています。
4. **次へ**をクリックします。SPS に対して使用されるスキーマを選択する XML スキーマを選択ダイアログが表示されます。以下の記述を参照ください。

メモ JDBC を介して Oracle または IBM DB2 データベースへ接続して、SELECT ステートメントの追加/編集コマンドによりデータを取得する場合、SELECT ステートメントに終了を表すセミコロンを付け加えないようにしてください。

SPS に使用する XML スキーマを選択する

SPS に対して使用されるスキーマは DB 内に含まれている XML スキーマ、または StyleVision にてアクセスすることができる位置にあるスキーマとなります。以下の操作によりスキーマを選択することができます:

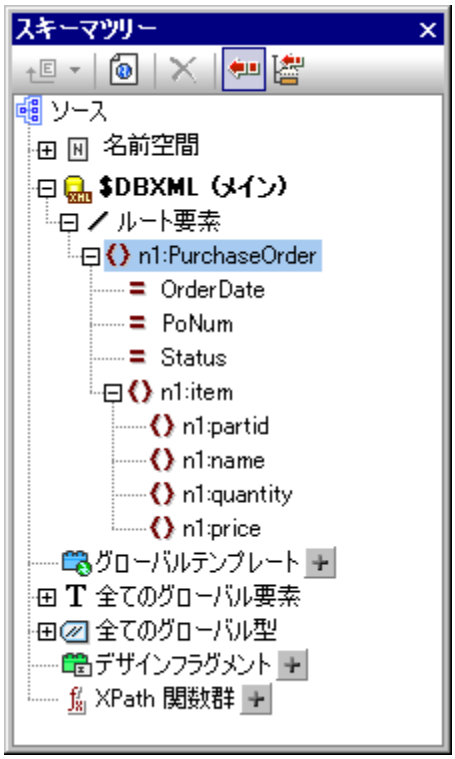
1. XML スキーマを選択ダイアログにて、DB 内に保管されているスキーマ、またはファイルシステム上にあるスキーマを選択するためのラジオボタンを選択してください。非 DB スキーマ（つまり外部ファイルのスキーマ）が選択された場合、DB による検証が行われることは無いという点に注意してください。



2. スキーマを選択します。DB 内に保管されているスキーマはデータベースからのスキーマのドロップダウンリストから選択することができます。外部スキーマはファイルの参照を行うかたちで選択することができます。
3. **完了**をクリックして完了します。

スキーマツリー

XML データとスキーマを選択すると、選択された XML データが作業用 XML ファイルとして作成され、スキーマが SPS にロードされます。以下のスクリーンショットでは、作業用 XML ファイルとスキーマの両方が示されています。



通常の StyleVision 機能により SPS のデザインを行うことができるようになります。作業用 XML ファイル内のデータは Authentic View により編集することができ、変更は DB へ保存されます。

メモ 作業用 XML ファイルは SPS がベースにしているスキーマに対して妥当である必要があります。スキーマの ルート要素 (ドキュメント要素) が XML ドキュメントのルート要素に対応していることも確認してください。

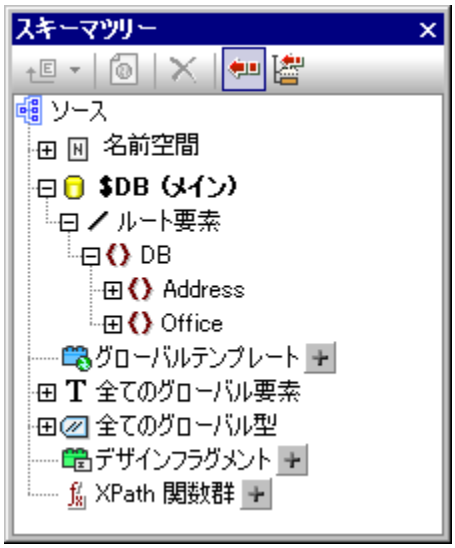
(C) 2015-2021 Altova GmbH

10.4 DB スキーマと DB XML ファイル

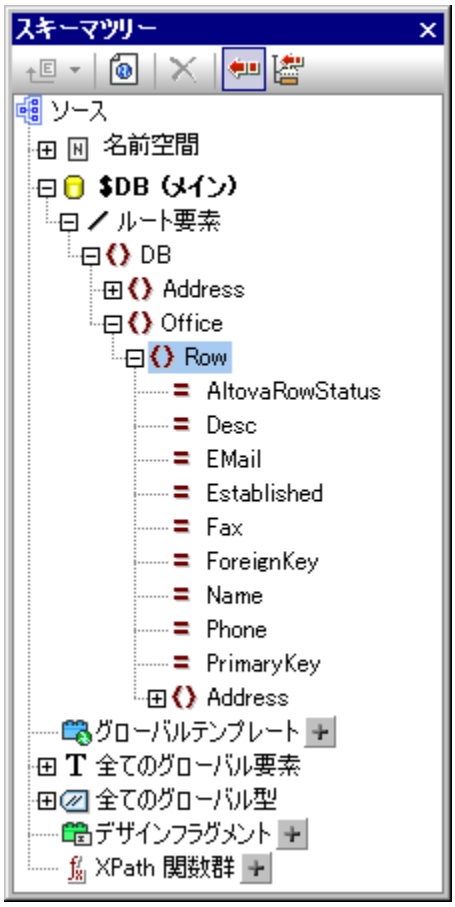
DB XML スキーマ

非 XML データベース(非 XML DB)を StyleVision にロードすると、DB の構造をベースにした XML スキーマが StyleVision により生成され、スキーマツリーウィンドウに表示されます XML DB の場合、(DB またはファイルシステム上にある)既存のスキーマが SPS にて使用されるスキーマとして指定されます。このセクションでは、非 XML DB で使用されるスキーマだけを取り上げます。

XML スキーマは DB という名前のドキュメント要素とともに作成されます。DB 要素には DB 内のトップレベルテーブルに対応した子要素が含まれ、これらトップレベルのテーブル要素はグローバル要素のエントリーとしても作成されます。以下のスクリーンショットにあるトップレベル要素は Address と Office になり、これらの要素は DB 内のテーブルに対応しています。



各々のトップレベルテーブル要素には、数に制限の無い Row という要素が含まれています(以下のスクリーンショットを参照)。各 Row 要素は DB におけるレコードに対応しており、スキーマツリーでは DB テーブルの各行が Row 要素にて表されます。各 Row 要素にはテーブルのフィールドに対応する属性が含まれています。これらの属性のうち AltovaRowStatus という属性は StyleVision により生成されており、行の現在ステータスが保管されます。その他の属性は対応する DB テーブルのフィールドとなります。



メモ XML スキーマの構造は上記のように生成されます。接続時に選択されたテーブルは全てこの構造に含まれることになります。DB フィルターは XML スキーマの構造に影響を与えません。

新たな DB スキーマ構造

バージョン 2005 以降の StyleVision により DB から生成された XML スキーマの構造は、それ以前の StyleVision により生成されたスキーマの構造とは異なるようになります。新たな構造では、データベースの編集を Altova 製品の Authentic View にて行うことが可能になります。これにより、それ以前のバージョンの StyleVision により生成された SPS をバージョン 2005 以降の StyleVision にて開こうとすると、エラーが発生します。StyleVision の DB 編集ならびにレポート機能を使用するには、新しいバージョンの StyleVision にて SPS を再度作成する必要があります。

DB XML データファイル

XML DB への接続を確立し、XML スキーマと XML データを含むカラムを選択すると、作業用 XML ファイルに対して使用される選択されたスキーマとカラムがスキーマツリーウィンドウに表示されます（以下のスクリーンショットを参照）。

DB からは2つの一時的な XML ファイルが生成されます(（次を参照:DB とStyleVision):

- Authentic View にて編集することができる一時 XML ファイル
- （プレビューの表示や出力生成を行うために）作業用 XML ファイルとして使用される編集不可能な XML ファイル

編集可能な一時 XML ファイルはDB がStyleVision へロードされた時に生成され、SPS が作成されていればAuthentic View にて編集を行うこともできます。Authentic View にて使用することができるクエリ機能により、Authentic View の表示内容をフィルタリングすることもできます。Authentic View 内で行われた編集可能なデータに対する全ての変更は、この一時 XML ファイルへ書き込まれ、メニューオプションの「ファイル | Authentic の XML データを保存」コマンドにより、一時的な編集可能 XML ファイルへの変更がDB へ反映されます。

編集不可能な一時 XML ファイルはDB がStyleVision へロードされた時に生成されます。このファイルは作業用 XML ファイルとして、HTML とRTF 出力を生成するために使用されます。Authentic View にて行った変更をプレビューにて確認するには、プレビュー前に編集可能な XML ファイルを保存する必要があります。

メモ

- その他の Authentic View における Authentic View では、DB をベースにした SPS が開かれた際に編集可能な一時ファイルだけが作成されます。Authentic View にて行われた変更はこのファイルへ書き込まれ、ファイルの保存時に XML ファイル内部にある情報がDB へ書き込まれます。
- レポートの生成時には、フィルタリング機能により編集不可能な一時 XML ファイル内にあるデータをフィルタリングすることができます（次を参照:DB フィルターの編集 ）。
- HTML とRTF のプレビューを閲覧するのに作業用 XML ファイルを割り当てる必要は特にありません。ここでは自動的に生成された（編集不可能な）一時 XML ファイルが使用されます。

10.5 DB に対する SPS デザイン機能

DB をベースにした SPS は、スキーマノードをスキーマウィンドウからデザインドキュメントへドラッグし、静的コンテンツをデザインドキュメントへ直接挿入し、デザインコンポーネントに対して適切なフォーマットを適用し、通常のスキーマをベースにした SPS と同様の方法で操作することができます。しかし DB をベースにした SPS を扱う際には、以下の点に注意する必要があります:

DB テーブルに対して動的なテーブルを作成する

以下の操作により DB テーブルに対して動的なテーブルを作成することができます:

1. スキーマツリーにて、動的テーブルとして作成するトップレベルの DB テーブルを選択しデザインへドラッグします。
2. テーブルをドロップしてコンテンツとして作成し、コンテンツプレースホルダを削除します。DB コントロールの自動追加機能が有効になっている場合、デザインは以下のようになります:

3. スキーマツリーにて、動的テーブルとして作成する DB テーブルの Row 要素を選択します。
4. Row 要素を既に作成したテーブルノード内部(上の例では Address)にドラッグします。
5. 要素をドロップして、表示されるメニューからテーブルの作成を選択し、動的テーブルのカラムとして作成する DB フィールドを選択します。DB テーブルが動的テーブルとして作成されます。

メモ DB テーブルは、他のフォーマットでも作成することができます(例: コンテンツ)。

DB コントロールの自動追加

メニューオプションの「**Authentic | DB コントロールの自動追加**」または ツールバーアイコン により、DB テーブルに対するナビゲーションコントロールの自動挿入を有効/無効にすることができます。自動挿入機能が有効になっている場合、DB テーブルがデザインドキュメントへドロップされた際に、DB コントロールも自動的に挿入されます(以下のスクリーンショットを参照)。デフォルトではテーブルタグの直後に挿入が行われます。

これらのコントロールにより、DB テーブルのレコードを、Authentic View にてナビゲーションすることができるようになります。最初の(左端にある)ボタンにより最初のレコードが表示され、2番目のボタンにより直前のレコードが表示され、3番目のボタンをクリックするとレコードへ移動ダイアログが表示されます(以下のスクリーンショットを参照)。4番目のボタンにより次のレコードが表示され、5番目のボタンにより最後のレコードが表示されます。

以下の操作により、デフォルトの場所とは別の場所へコントロールを手動で配置することができます:

1. DB コントロールの自動追加機能を無効にします。
2. コントロールを挿入する場所(DB テーブル要素の開始ならびに終了タグ内)にカーソルを配置します。

3. 右クリックして表示されるコンテキストメニューから「**DB コントロールを挿入 | ナビゲーション**」または「**DB コントロールを挿入 | ナビゲーション + 移動**」を選択します。これらのコマンドは「**挿入**」メニューからも選択することができます。

Authentic View で使用する DB クエリボタンの挿入

DB クエリボタン により、Authentic View にて DB クエリを使用することができます。この機能を使用することで、Authentic View に表示されるレコードの条件を作成することができるようになります。クエリボタンは、個々の DB テーブルに対して以下の範囲内に挿入することができます:

- DB テーブル要素の開始タグと DB テーブル要素（子）の Row 要素の開始タグ
- DB テーブル要素の終了タグと DB テーブル要素（子）の Row 要素の終了タグ

以下の操作により、デザインドキュメントに DB クエリボタンを挿入することができます:

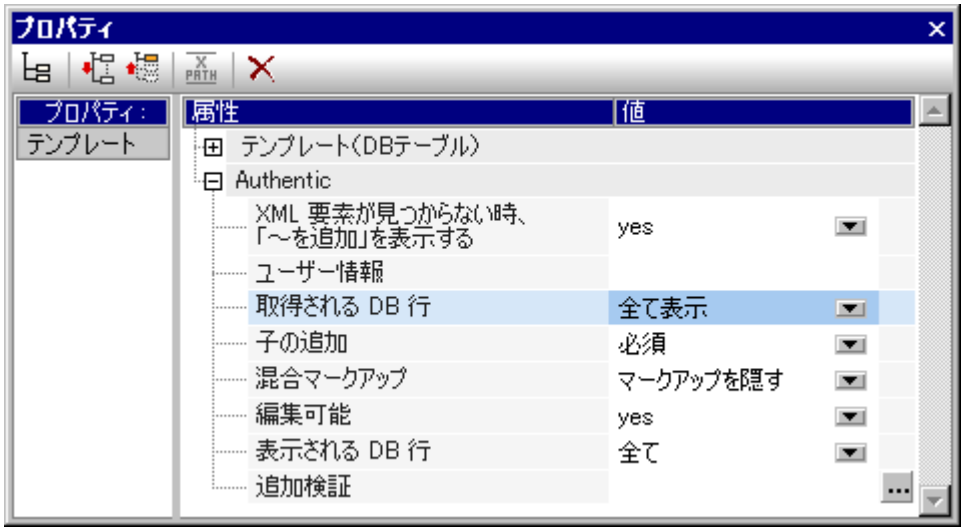
1. 許されている場所（上記参照）でクエリボタンを表示する場所に、カーソルを配置します。
2. 右クリックにより表示されるコンテキストメニュー、またはメニューオプションの「**挿入**」から、「**DB コントロールを挿入 | クエリボタン**」を選択します。デザインドキュメントにクエリボタンが挿入されます。

Authentic View にてクエリボタンがクリックされると、データベースクエリの編集ダイアログが表示されます。このダイアログに関しては Authentic View ドキュメントを参照ください。

表示されるレコードと取得されるレコード

ファイルがロードされた際に取得されるレコードと、Authentic View にて表示されるレコードの数を指定することができます。以下の操作によりこれらの設定を行うことができます:

1. デザインビューにて取得/表示するレコードに対応する Row 要素を選択します。
2. プロパティサイドバーにあるプロパティペインにてテンプレートを選択し、Authentic プロパティグループを選択します。



3. **表示される DB 行**プロパティにてドロップダウンリストから**全て**を選択するか、表示されるレコード数を入力します。
4. **取得される DB 行**プロパティにてドロップダウンリストから全てまたは全て表示を選択するか、取得されるレコードの数を入力します。デフォルトでは全て表示が選択されています。このプロパティにより、DB データがロードされた際に取得されるレコードの数が指定されます。表示される DB 行の値よりも小さな値が指定された場合、表示される DB 行の値が取得される DB 行の値として使用されます。取得される DB 行プロパティにより、最初に取得されるレコードの数を減らして、ロードならびに表示時間を短縮

することが可能になります。Authentic View に配置された行（レコード）ナビゲーションボタンにより、別のレコードの取得と表示を行うこともできます。

メモ

- 表示される行の数は Authentic View に対して適用されます。
- 多くのテーブルが に開かれている場合、開かれているテーブルの数が多すぎるというエラーメッセージが表示されます。デザイン時に表示されるレコードの数を制限することで、開かれているテーブルの数を制限することができます（テーブルはレコードの中にも含まれるため）。ユーザーは、クエリを使用することで Authentic View にロードされるテーブルの数を制限することができます。

AltovaRowStatus

DB から作成された XML スキーマでは、各テーブルに AltovaRowStatus 属性が与えられます。この属性の値は StyleVision により自動的に決定され、初期値が --- の3文字により構成されます。これらの文字は以下に示されるような意味を持ちます。

A	行が追加されました（がDB へはまだ保存されていません）
U,u	行が更新されました（がDB へはまだ保存されていません） DB).
D,d, X	行が削除されました（がDB へはまだ保存されていません）

これらの値は編集されている行の情報を与えるために使用されます。ステータス情報はファイルが保存されるまで保持され、ファイルが保存されると値が初期化（---）されます。

デザインドキュメントコンポーネントの書式

レコードが追加、修正、削除されると、その他のレコードと区別するために追加/修正/削除されたレコードの書式が変更されます。データ型のエラーも赤色により表示されます。外観を使った区別を行う場合、テーブルに対して（手動で）割り当てられる書式がこれらの書式と重複しないようにする必要があります。StyleVision により自動的に割り当てられる書式を以下に示します：：

追加された	A	太字、下線
修正された（更新された）	U, u	下線
削除された	D, d, X	取り消し線
データ型エラー		赤色文字

10.5.1 DB フィルター：DB データのフィルタリング

データベース（DB）から**編集不可能**な一時 XML ファイルへインポートされたデータはフィルタリングすることができます。編集不可能な XML ファイルはレポートの生成に使用され、フィルタリングの効果は HTML と RTF のプレビューでのみ反映され、編集可能な一時 XML ファイルの内容を表示する Authentic プレビューには反映されないことに注意してください。DB フィルターは（ご使用の DB アプリケーションでサポートされている場合）DB 内部で作成することができるほか、SPS（SPS ファイル）内部で作成することができます。SPS では、XML スキーマにおけるトップレベルのデータテーブル（つまり DB コンポーネントの子であるデータテーブル）に対してそれぞれ1つの DB フィルターを作成することができます。DB フィルターが作成または修正されると、DB から得られたデータがその DB に対して生成された編集不可能な XML ファイルへ再ロードされます。DB フィルターを使用することで XML ファイルを望ましい大きさに抑え、レポート生成の処理をより効率的に行うことができます。

(C) 2015-2021 Altova GmbH

メモ　DB フィルターを使用することで、編集不可能な一時 XML ファイルへインポートされたデータを修正することができます。DB フィルターを伴う SPS を保存して、その SPS から XML ファイルを生成すると、DB フィルターの条件に従うかたちで生成された XML ファイルのフィルタリングが行われます。

DB フィルターの作成

1. デザインドキュメントまたはスキーマツリーにて、DB フィルターを作成するデータテーブル要素を選択します（要素の開始または終了タグをクリックするか、スキーマツリーに表示されている要素を選択してください）。
2. メニューオプションの データベース | DB フィルターの編集 を選択するか、ツールバーの アイコンをクリックしてください。以下に示されるダイアログが表示されます。

3. AND を追加 または OR を追加 ボタンをクリックします。空のフィルター条件が追加されます（以下のスクリーンショットを参照）。

4. 条件に使用する値を入力します。条件は、(i) (コンボボックスから選択可能な) フィールド名、(ii) (コンボボックスから選択可能な) 演算子、そして (iii) 直接入力される値により構成されます。条件作成の詳細については、以下にある条件で使用される表現 を参照ください。
5. 別の条件を追加するには、条件の組み合わせに使用する論理演算子に従い、**AND を追加**または**OR を追加**ボタンをクリックして、新たな条件を作成します。論理演算子の詳細については DB フィルターにおける条件の順序替えを参照ください。

条件で使用される表現

DB フィルターの条件は、フィールド名、演算子、そして値により構成されます。**選択可能なフィールド名**は選択されたトップレベルのデータテーブル内にあるカラムで、フィールドの名前はコンボボックスにて選択することができます (上のスクリーンショットを参照)。使用することのできる**演算子**のリストを以下に示します:

=	等しい
<>	等しくない
<	指定された値未満
<=	指定された値以下
>	指定された値より大きい
>=	指定された値以上
LIKE	パターンに一致する
NOT LIKE	パターンに一致しない
IS NULL	空である
NOT NULL	空ではない

IS NULL または NOT NULL が選択されている場合、値フィールドが無効になります。値の入力では引用句などのデリミタが使用されません。入力された値は対応する DB フィールドと同一のフォーマットでなければならず、そうでない場合評価結果は FALSE となります。例えば、Microsoft Access DB にある date データ型のフィールドに対して、StartDate=25/05/2004 という値の条件が与えられた場合、Microsoft Access DB の date データ型のフォーマットは YYYY-MM-DD となっているため、FALSE が返されることになります。

DB フィルターにてパラメーターを使用する

パラメーターの名前を表現の値として入力することもできます。これによりパラメーターの呼び出しが行われ、その値を条件の値として使用することができます。スタイルシートに対して既に宣言されたパラメーターと、将来宣言するパラメーターの両方を入力することができます。パラメーターが宣言されていない状態で値フィールドへの入力が行われた場合、「OK」ボタンが選択不可能な状態になります。

同一の値を複数の条件で使用する、又はコマンドラインから値を渡すような場面でパラメーターを利用することができます（詳細については次を参照: StyleVision Server ）。

表現の値としてパラメーターの名前を入力するには、値入力フィールドに $ を入力し、続けて（空白などをはさまずに）パラメーターの名前を入力します。パラメーターが既に宣言されている場合、エントリーが緑色で表示され、パラメーターがまだ定義されていない場合、エントリーが赤色で表示されます（次を参照: パラメータ）。

パラメータの DB フィルターの編集 ダイアログからパラメーターを宣言する

以下の操作により、パラメーターの編集ダイアログへアクセスしてパラメーターを宣言することができます:

1. データベースフィルターの編集ダイアログにて**パラメーター**ボタンをクリックします。パラメーターの編集ダイアログが表示されます。

2. パラメーターの名前と値を対応するフィールドに入力します。

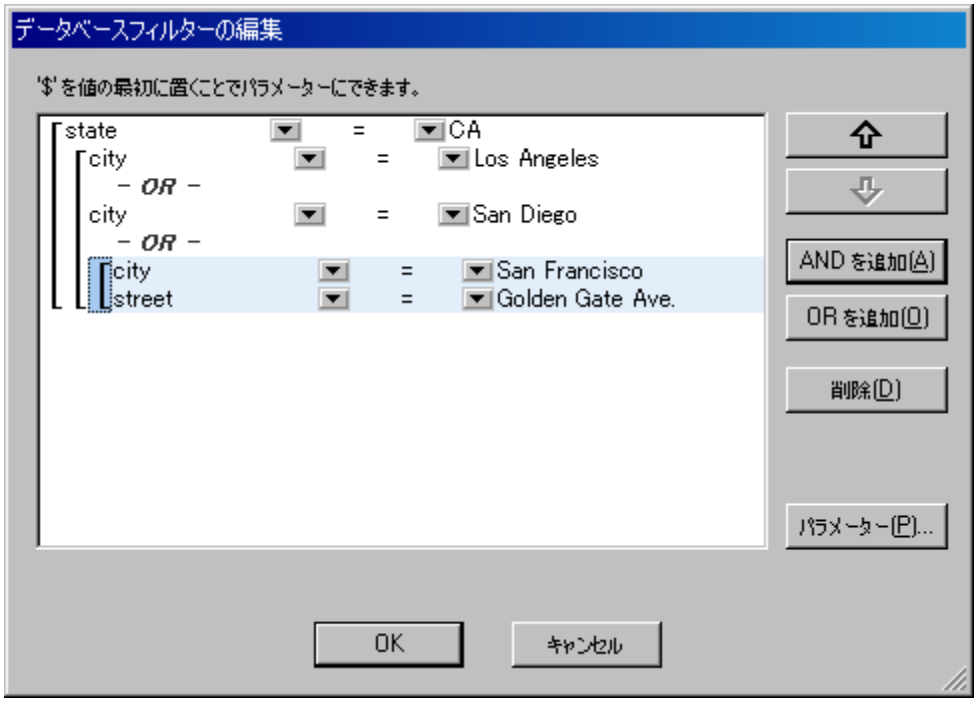
メニューオプションから「**編集 | スタイルシートのパラメーター**」コマンドを選択することでも、パラメーターの編集ダイアログへ直接アクセスしてパラメーターを宣言し、DB パラメーターを編集することができます。

メモ パラメーターの編集ダイアログには、スタイルシートに対して定義された**全ての**パラメーターが表示されます。SPS にて宣言されていないパラメーターを使用するとエラーが返されますが、使用しないパラメーターを宣言してもエラーにはなりません。

データテーブルの要素に対して DB フィルターを作成すると、スキーマツリーにある要素がフィルターシンボルとともに表示されます。以下のスクリーンショットでは、Addresses 要素に対してフィルターシンボルが表示されています。

DB フィルターにおける条件の順序替え

データベースフィルターの編集ダイアログでは、DB フィルターの論理構造と条件間の関係が視覚的に示されます。論理構造の各レベルは角括弧で示されます。隣同士に配置されている条件は AND 演算子(論理積)により組み合わせられ、2つの条件が OR により評価される場合、OR 演算子(論理和)が使用されます。ダイアログの表示により DB フィルターに使用される条件の論理構造を簡単に把握することができます。



上のスクリーンショットに示されている DB フィルターは、以下の様なテキストに書き直すことができます:

State=CA AND (City=Los Angeles OR City=San Diego OR (City=San Francisco AND カスタムer Nr=25))

DB フィルターに示された各条件や条件セットの順序は変更することができます。以下の操作により条件や条件セットを移動することができます:

1. 条件をクリックするか、レベルを表す角括弧をクリックして、条件やレベルを選択する。
2. ダイアログに表示された上矢印または下矢印をクリックする。

以下の点に注意してください:

- 移動方向に同レベルの条件がある場合、これら条件の位置が交換することになります。
- 条件のセット(同じ角括弧以下にある条件)の位置を同一条件内で変更することはできますが、レベルの深さは変更されません。
- 個々の条件は同一レベル内を移動します。隣にある条件が別のレベルに存在する場合、選択された条件は1 つのレベルずつ移動することになります。

条件が選択された状態で削除ボタンをクリックすることで、DB フィルターにある条件を削除することができます。

(C) 2015-2021 Altova GmbH

DB フィルターの編集

DB フィルターを修正するには、メニューオプションから データベース | DB フィルターの編集 を選択します。データベースフィルターの編集ダイアログが表示され、表示された条件の編集、新たな条件の追加、条件の位置変更、または条件の削除を行うことができます。修正後には「OK」をクリックして完了します。DB のデータが StyleVision へ自動的に再ロードされ、DB フィルターへの変更が反映されます。

DB フィルターのクリア(削除)

DB フィルターをクリア(削除)するには、デザインウィンドウまたはスキーマツリーにて DB フィルターを削除する要素を選択し、メニューオプションから データベース | DB フィルターをクリアする を選択します。フィルターが削除され、スキーマツリーに表示された要素名の隣に表示されていたフィルターシンボルが表示されなくなります。

10.6 出力ファイルの生成

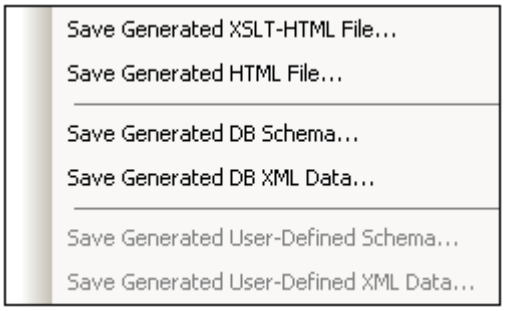
DB をベースにしたSPS が作成されると、SPS に関連したファイルの生成や保存を行うことができるようになります。以下のファイルを生成ならびに保存することができます:

- DB の構造をベースにしたXML スキーマ
- XML スキーマとDB のコンテンツから構成されるXML ファイル
- HTML を出力するXSLT ファイル
- HTML 出力ファイル
- RTF を出力するXSLT ファイル
- RTF 出力ファイル

ファイルの生成と保存はGUI ならびにコマンドラインから行うことができます。

StyleVision GUI からの操作

1. **ファイル** メニューから**生成されたファイルの保存** コマンドを選択します。以下のサブメニューが表示されます。



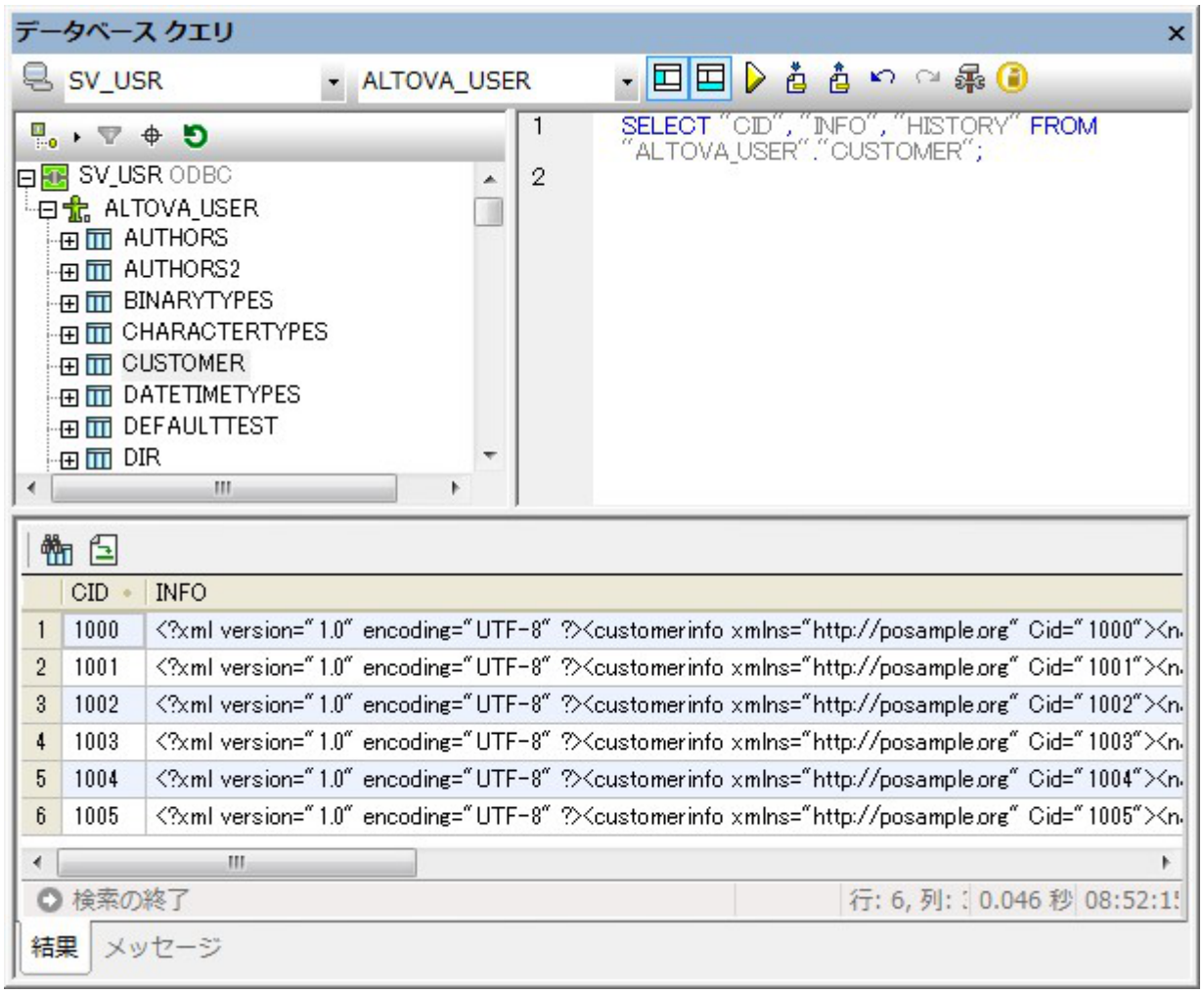
2. 生成するファイルを選択します。名前をつけて保存ダイアログが表示されます。
3. 出力先のフォルダーを選択し、ファイル名を入力した後に「OK」をクリックします。

コマンドラインからの操作

コマンドラインから StyleVision を呼び出し、DB をベースにしたSPS に関連するファイルの生成と保存を行うことができます。XML スキーマやXSLT ファイルだけではなく、DB から得られたXML ファイルや、SPS のデザインをベースにしたHTML とRTF 出力ファイルを保存することができます。

10.7　データベースへの問い合わせ

データベースメニューオプション以下にあるデータベースへ問い合わせコマンドにより、データベースクエリウィンドウが表示されます（以下のスクリーンショットを参照）。クエリウィンドウの表示は、「データベース | データベースへ問い合わせ」コマンドまたはデータベースの問い合わせツールバーアイコン により切り替えることができます。



データベース クエリ ウィンドウの概要

データベースクエリウィンドウは3つのパーツから構成されます:

- 左上にある ブラウザーペイン には、接続情報とデータベーステーブルなどが表示されます。
- 右上にある クエリペイン ではクエリの入力を行います。
- タブ化された 結果/メッセージペイン が表示されます。結果ペインにはクエリの結果が結果グリッドにより表示され、メッセージペインには警告やエラーを含むクエリ実行に関するメッセージが表示されます。

データベースクエリウィンドウ上部にはツールバーが表示されています。ここでは以下に示される2つのツールバーアイコンについて注意してください。その他のツールバーアイコンについては、 クエリペン: 説明と機能 のセクションにて記述されます。

ブラウザーペインの表示を切り替えます。

結果/メッセージペインの表示を切り替えます。

データベースへ問い合わせメカニズムの概要

データベースのクエリは以下のように動作します(詳細についてはこのセクション以下にあるサブセクションを参照ください)。

1. データソースへの接続ウィンドウを介して データベースへの接続が確立 されます。
2. 接続されたデータベースまたはその一部が ブラウザーペインに表示され、必要に応じて表示構成を変更することができます。
3. 問い合わせを行うデータベースに対して適切な構文により記述されたクエリを クエリペインに入力し、クエリが実行されます。
4. クエリの結果 は様々なフィルターを介して確認することができます。

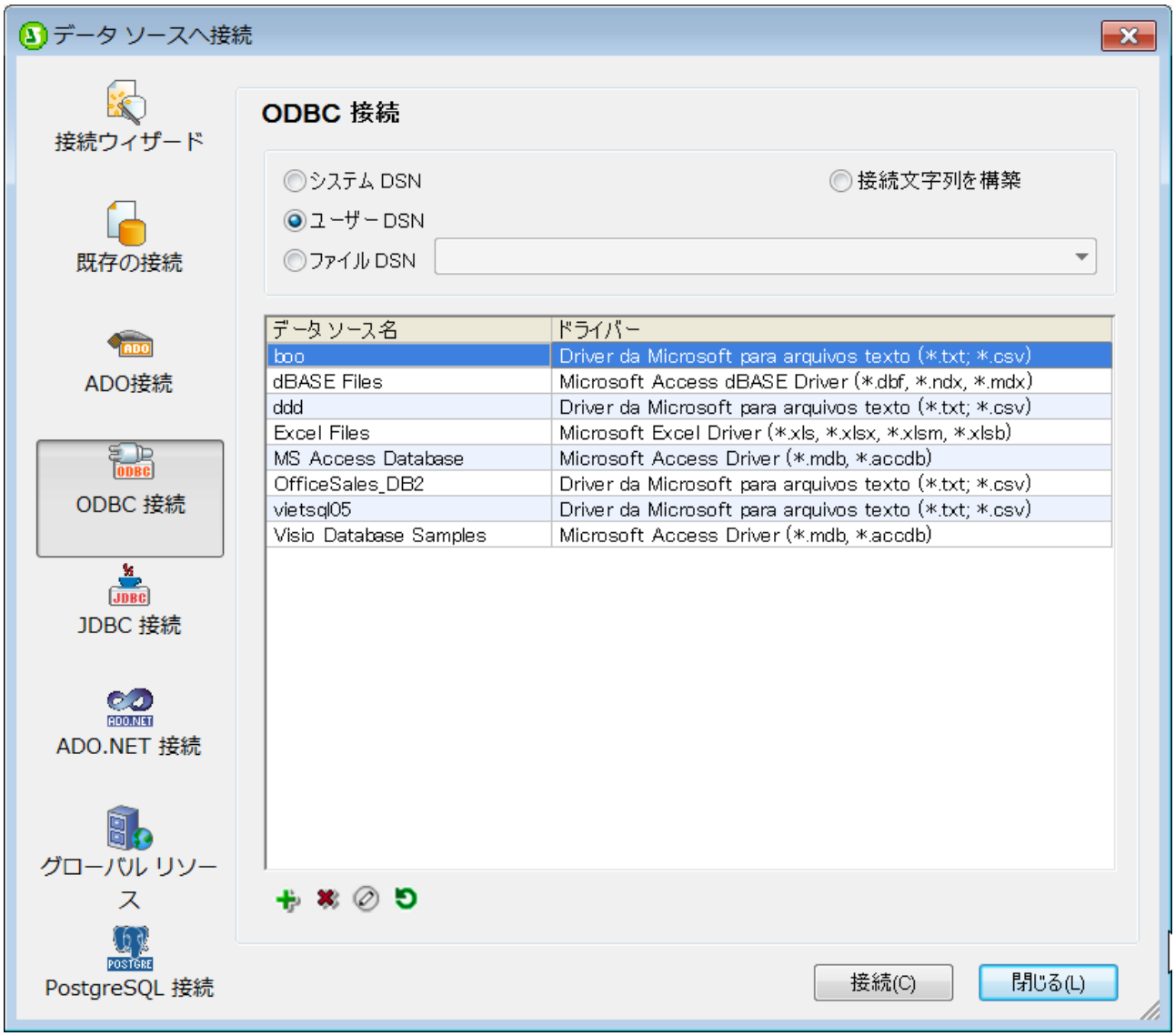
10.7.1 データソース

データベースの問い合わせを行うにあたり、まずは目的のデータベースへ接続する必要があります。このセクションでは以下の項目について記述されます:

- データベースへ接続する
- 複数ある接続から、目的のデータソースならびにルート要素を選択する

データベースへの接続

データベースメニュー以下にある**データベースへ問い合わせ**コマンドを、アプリケーション起動後初めてクリックすると、(データベースへの接続が存在しない場合)データソースへ接続ダイアログが表示され、データベースへの接続を行うことができます(以下のスクリーンショットを参照)。データベースクエリウィンドウ内に配置されたクイック接続アイコン をクリックすることでも、データベースへの接続を行うことができます。接続が既に存在する場合、目的の接続を選択 することができます。



データベースへの接続ダイアログを使ったデータベースへの接続方法については、データソースへの接続セクションを参照ください。

サポートされる データベース

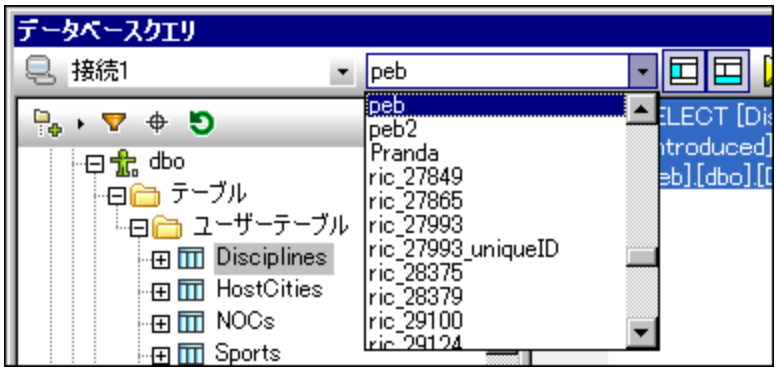
次のデータベースはサポートされています。各データベースの使用できるルート オブジェクトもリストされています。Altova は他の ODBC/ADO データベースへのサポートの向上のための努力をしていますが、以下にリストされたデータベースのみで接続とデータ処理のテストに成功しています。使用中の Altova アプリケーションが、64 ビット バージョンの場合、接続されているデータベース特定の 64 ビット データベース ドライバーにアクセスしてください。Altova アプリケーションは IBM iSeries データベースのロジカル ファイルをサポートし、ロジカル ファイルを表示します。

データベース	メモ
Firebird 2.5.4, 3.0	
IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5	

データベース	メモ
IBM Db2 for i 6.1, 7.1, 7.2, 7.3, 7.4	ロジカルファイルはサポートされ、ビューとして表示されています。
IBM Informix 11.70, 12.10, 14.10	
MariaDB 10.2, 10.3, 10.4, 10.5	
Microsoft Access 2003, 2007, 2010, 2013, 2016, 2019	ドキュメントの作成時（2019 年9 月初旬）Access 2019 のために使用可能な Microsoft Access Runtime が存在しませんでした。Microsoft Access 2016 Runtime がインストールされており、データベースが「大量」のデータ型を使用しない場合のみ Altova 製品から Access 2019 にデータベース接続することができます。
Microsoft Azure SQL Database	SQL Server 2016 codebase
Microsoft SQL Server 2005, 2008, 2012, 2014, 2016, 2017, 2019 Microsoft SQL Server on Linux	
MySQL 5.0, 5.1, 5.5, 5.6, 5.7, 8.0	
Oracle 9i, 10g, 11g, 12c, 18, 19	
PostgreSQL 8.0, 8.1, 8.2, 8.3, 9.0.10, 9.1.6, 9.2.1, 9.4, 9.5, 9.6, 10, 11, 12.1	PostgreSQL 接続は、ODBC または JDBC などのインターフェイス（ドライバ）によりネイティブドライバーベースの接続でサポートされています。ネイティブな接続はドライバを必要としません。
Progress OpenEdge 11.6	
SQLite 3.33.0	SQLite 接続は、ネイティブとしてサポートされます。SQLite データベースファイルへの直接の接続です。 別途のドライバーは必要ありません。Authentic ビューでは、SQLite からのデータを編集することはできません。Authentic ビューで、SQLite からのデータを保存する場合は、既知の制限についてのメッセージが通知されます。
Sybase ASE 15, 16	
Teradata 16	

目的のデータソースを選択する

既存の接続ならびに接続ごとのルート要素は、データベースクエリウィンドウのツールバーに配置された2つのコンボボックスに表示されます（以下のスクリーンショットを参照）。左側のコンボボックスにてデータソースを選択し、右側のコンボボックスからルートオブジェクトを選択することができます。



上のスクリーンショットでは StyleVision DB という名前のデータベースが選択され、このデータベースにて利用することのできるルートオブジェクトのうち、ALTOVA_USER が選択されています。データベースならびにルートオブジェクトはブラウザーペインにて表示されます。

10.7.2　ブラウザーペイン: DB オブジェクトの閲覧

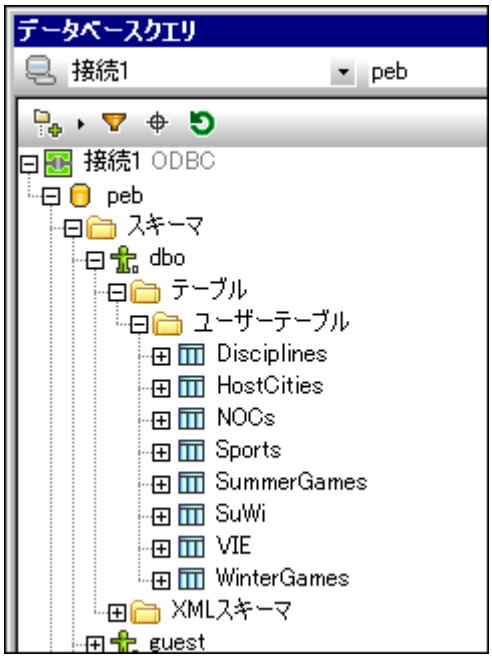
ブラウザーペインでは、選択されたデータベースにあるオブジェクトの概要が表示されます。この概要には、カラムの主キーや外部キーといったデータベースの制約情報も表示されます。IBM DB2 バージョン 9 データベースでは、登録された XML スキーマが別のフォルダー以下に表示されます（以下のスクリーンショットを参照）。

このセクションでは以下の項目について説明します:

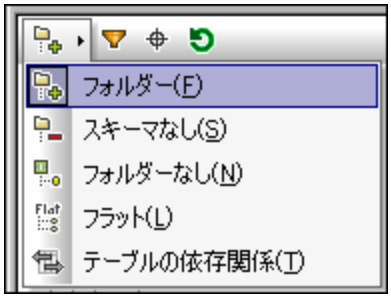
- ブラウザーペインで利用可能な レイアウト
- データベースオブジェクトのフィルター
- データベースオブジェクトの検索 データベースオブジェクト
- アクティブなデータソースにおけるルートオブジェクトの更新

ブラウザーペインレイアウト

デフォルトのフォルダーレイアウトにはデータベースオブジェクトが階層構造で表示されます。アイテムを右クリックすると、選択されたオブジェクトにより異なるコンテキストメニューが表示されます。



ブラウザーのレイアウトを選択するには、ブラウザーペインのツールバーにあるレイアウトアイコンをクリックして、ドロップダウンリストからレイアウトを選択します（以下のスクリーンショットを参照）。選択されたレイアウトによりアイコンが変わります。



選択することのできるレイアウトを以下に示します:

- *フォルダー*：階層ツリーに表示されているオブジェクトの種類により、データベースオブジェクトをフォルダーごとに表示します。これがデフォルトの設定となります。
- *スキーマ無し*：データベーススキーマフォルダーが無いという点を除きフォルダーレイアウトと同じ表示が行われます。テーブルはデータベーススキーマによりカテゴライズされません。
- *フォルダー無し*：フォルダーを使用せずにデータベースオブジェクトを階層構造で表示します。
- *フラット*：データベースオブジェクトを種類ごとに分けて最初の階層レベル以下に表示します。例えば、テーブル内にカラムを表示する代わりに全てのカラムがカラムフォルダー以下に表示されます。
- *テーブルの依存関係*：他テーブルへのリレーションシップによりテーブルのカテゴライズを行います。外部キーが含まれるテーブル、外部キーにより参照されるテーブル、そして他テーブルへのリレーションシップを持たないテーブルにカテゴライズされます。

ユーザーならびにシステムテーブルへテーブルを並び替えるには、フォルダー、スキーマ無し、またはフラットレイアウトへの切り替えを行い、テーブルフォルダーを右クリックして**ユーザーとシステムテーブルの並べ替え**を選択します。ユーザーテーブルならびにシステムテーブルにてテーブルがアルファベット順に並べ替えられます。

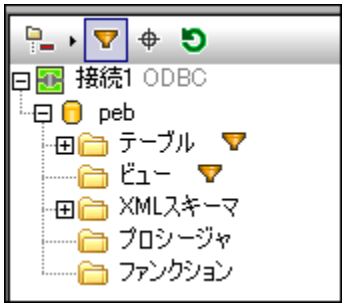
(C) 2015-2021 Altova GmbH

データベース オブジェクトのフィルタリング

ブラウザーペインでは、（フォルダーなしとテーブルの依存関係レイアウトを除き）名前や名前の一部からスキーマ、テーブル、ビューのフィルタリングを行うことができます。文字の入力を行うことでオブジェクトがフィルタリングされます。デフォルトでは大文字と小文字は区別されません。

以下の操作によりブラウザーに表示されたオブジェクトのフィルタリングを行うことができます:

1. ブラウザーペインのツールバーにあるフォルダーコンテンツのフィルターアイコンをクリックします。テーブルならびにビューフォルダーの隣にフィルターアイコンが表示されます（以下のスクリーンショットを参照）。



2. フィルタリングを行うフォルダーの隣にあるフィルターアイコンをクリックして、表示されるポップアップメニューからフィルタリングオプションを選択します（例: 含む）。

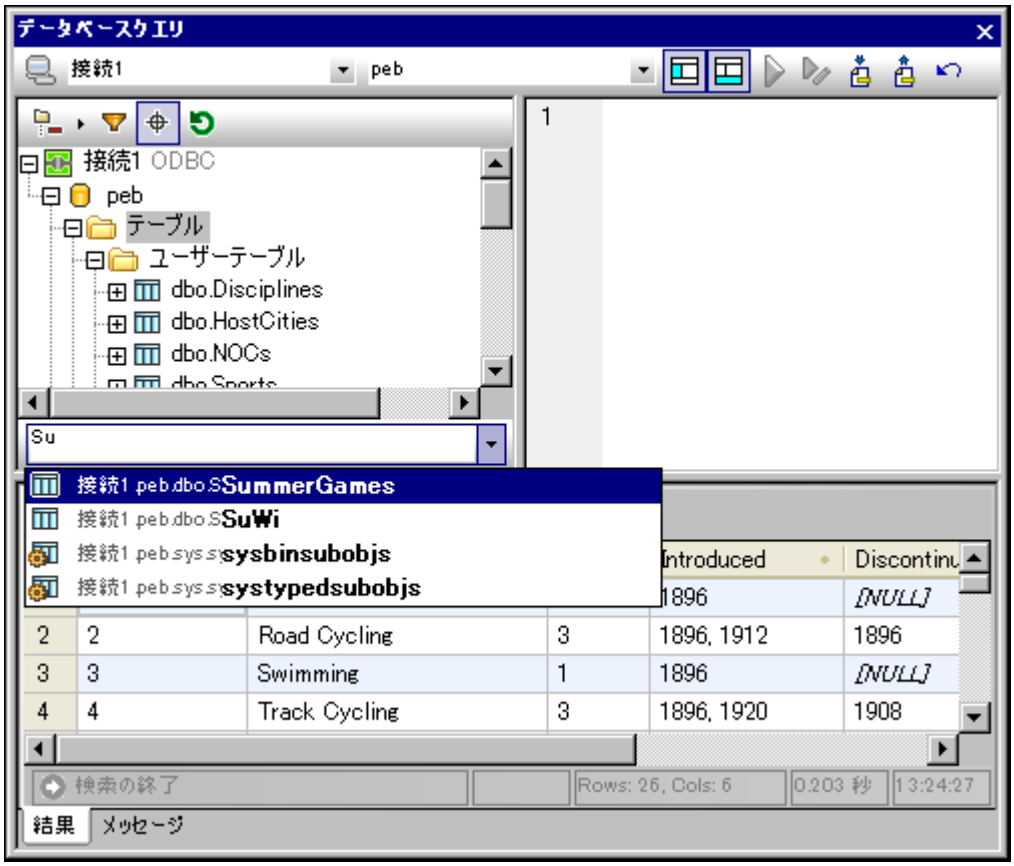
3. 表示される入力フィールドにてフィルタリングの文字列を入力します（以下のスクリーンショットではテーブルフォルダーに対して Su という文字列が入力されています）。入力と同時にフィルターが反映されます。

データベース オブジェクトの検索

名前により特定のデータベースアイテムを検索するには、以下のようにブラウザーペインのオブジェクトロケーターを使用することができます:

1. ブラウザーペインのツールバーにあるオブジェクトロケーターアイコンをクリックします。ブラウザーペイン下部にドロップダウンリストが表示されます。

2. リストの入力フィールドに検索文字列を入力します。ドロップダウンの矢印をクリックすることで、検索文字列を含む全てのオブジェクトが表示されます。



3. リスト内のオブジェクトをクリックすることで、そのオブジェクトがブラウザーペインにて選択されます。

ルートオブジェクトの更新

ブラウザーペインのツールバーにある最新の状態に更新ボタンをクリックすることで、アクティブなデータソースのルートオブジェクトを再描画することができます。

10.7.3 クエリペン: 説明と機能

クエリペインは、選択されたデータベースに対するクエリを入力するためのSQL エディターとして機能します。クエリを入力し、データベースクエリウィンドウにある 実行 コマンド をクリックすることにより、クエリ実行の結果とそのメッセージが結果/メッセージペインにて表示されます。クエリの使用方法については、次のセクションの クエリペン: クエリと作業する を参照ください。このセクションではクエリペインの主な機能について説明します:

- データベースクエリツールバーにおける SQL エディターアイコン
- SQL エディターオプション
- SQL ステートメントの自動補完
- SQL スクリプトの領域定義
- SQL スクリプトに対するコマンドの挿入

- ブックマークの使用

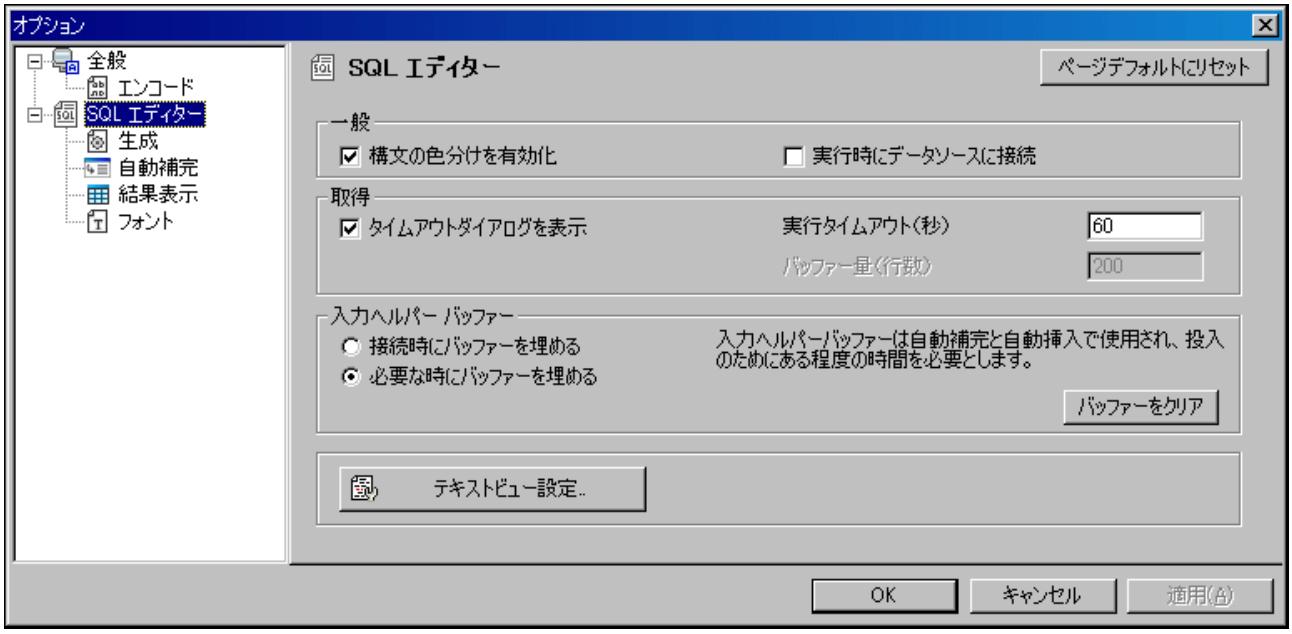
データベース クエリツールバーにおける SQL エディターアイコン

データベースクエリウィンドウのツールバーでは、SQL エディターに対して以下のアイコンを使用することができます:

	実行	現在選択されている SQL ステートメントを実行します。スクリプトに複数のステートメントが含まれており、そのうちどれも選択されていない場合、全てのステートメントが実行されます。
	SQL ファイルのインポート	SQL ファイルを SQL エディターにて開きます。
	SQL ファイルのエクスポート	SQL クエリを SQL ファイルに保存します。
	元に戻す	SQL エディターにて行った変更を元に戻します。
	やり直し	SQL エディターにて元に戻した変更をやり直します。
	オプション	SQL エディターのオプションダイアログが開きます。
	SQL スクリプトを DatabaseSpy で開く	SQL スクリプトを Altova DatabaseSpy で開きます。

SQL エディターオプション

データベースクエリツールバーにあるオプションアイコンをクリックすることで、オプションダイアログが表示されます(以下のスクリーンショットを参照)。左側のペインにてオプションの種類を選択することができ、ページのデフォルトにリセットボタンをクリックすることで、表示されている種類のオプション設定をデフォルトの値に戻すことができます。

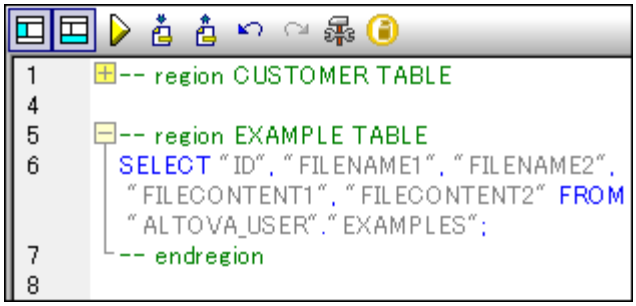


主な設定を以下に示します:

- *全般 ／ エンコード*：新規 SQL ファイルならびにエンコーディングが特定できない既存 SQL ファイルのエンコーディング、そしてバイトオーダーマーク（BOM）の設定を行います。既存の SQL ファイルにおけるエンコーディングが特定できる場合、エンコーディングが変更されることなくファイルは開かれ、保存されます。
- *SQL エディター*：構文の色分けや、実行時のデータソースへの接続を有効/無効にすることができます。クエリの実行にタイムアウトを指定することができ、指定された時間を超えた場合、タイムアウトを変更するためのダイアログを表示することもできます。入力ヘルパーは、自動補機能の一部として表示される入力ヘルパーを指します。SQL ステートメントを入力すると、エディターはコンテキストを考慮した自動補完の提案を表示します。自動表示がオフに設定されている場合、「Ctrl+Spacebar」を押す事によりSQL クエリ内でSQLÖ エディター内で自動補完の提案を要求することができます。入力ヘルパー情報のバッファーはデータソースへの接続時、または、データソースが最初に必要とされる時に入力することができます。テキストビュー設定ボタンは、SQL エディター内のテキストのタブサイズのインデントなどの設定を行うことのできるダイアログを開きます。
- *SQL エディター ／ 生成*：ブラウザーペインからクエリペインへオブジェクトがドラッグされると、SQL ステートメントが生成されます。SQL 生成タブでは、SQL ステートメントの生成に対するオプションをセットすることができます。データベースリストボックスを使用することにより、データベースの種類ごとにステートメントの生成オプションをセットすることができます。**全てのデータベースに適用**チェックボックスを使用することで、現在選択されているオプションが全種類のデータベースに対して適用されます。ステートメントの終端にセミコロンを追加し、エスケープ文字により識別子を囲むオプションをセットすることができます。
- *SQL エディター ／ 結果表示*：結果タブの設定を行うことができます。
- *SQL エディター ／ フォント*：テキストエディターならびに結果ビューのフォントスタイルを指定することができます。

SQL スクリプト内の領域の定義

領域とはSQL スクリプトにて1つのまとまりとしてマーク、宣言されたものです。領域を使用することで、スクリプトの一部分を縮退または展開することができるようになります。領域を他の領域内にネストすることも可能です。領域は、-- region と-- endregion というコメントにより、それぞれ開始/終了することになります。オプションとして領域には名前をつけることもでき、-- region の後に名前が入力されます（以下のスクリーンショットを参照）。



領域を挿入するには、領域内に含めるステートメントが選択された状態で、右クリックして表示されるコンテキストメニューから**領域の挿入**を選択します。縮退/展開可能な領域が作成され、必要ならば名前を付け加えることもできます。上のスクリーンショットにおける行番号に注目してください。領域を削除するには、--region と--endregion のコメントを削除してください。

SQL スクリプト内にコメントを挿入する

SQL スクリプト内にあるテキストはコメントアウトすることができます。コメントアウトされた箇所はスクリプトの実行時にスキップされることになります。

- ブロックをコメントアウトするには、マークされたブロックを右クリックして、ポップアップメニューから「**ブロックコメントを挿入／削除**」を選択します。ブロックコメントを削除するには、マークされたコメントを右クリックして「**ブロックコメントを挿入／削除**」を選択します。
- ある行、または行の一部をコメントアウトするには、行のコメントが始まる箇所にカーソルを配置して、右クリックにて表示されるコンテキストメニューから「**行コメントを挿入／削除**」を選択します。行コメントを削除するには、コメントをマークして右クリックにて表示されるコンテキストメニューから「**行コメントを挿入／削除**」を選択します。

(C) 2015-2021 Altova GmbH

ブックマークの使用

特定の行にブックマークを挿入することで、ドキュメント内のブックマークに対してナビゲーションを行うことができます。ブックマークを挿入するには、ブックマークされる行にカーソルを配置して、右クリックにより表示されるコンテキストメニューから「**ブックマークの挿入／削除**」を選択します。次、または前のブックマークへ移動するにはコンテキストメニューから「**次のブックマークへ移動**」または「**前のブックマークへ移動**」を選択します。ブックマークを削除するには、削除するブックマークがある行にカーソルを配置して、コンテキストメニューから「**ブックマークの挿入／削除**」を選択します。コンテキストメニューにある「**全てのブックマークを削除**」コマンドにより、作成されたブックマークが全て削除されます。

10.7.4　クエリペイン: クエリと作業する

データベースへの接続が確立されている状態では、SQL スクリプトを SQL エディターへ入力し、実行することができます。このセクションでは以下の項目について説明します:

- SQL エディターにて SQL スクリプトを入力する
- データベースクエリウィンドウにてスクリプトを実行する

このセクションでは以下のアイコンが使用されます:

	クエリの実行	現在選択されている SQL ステートメントを実行します。スクリプトに複数のステートメントが含まれており、そのうちどれも選択されていない場合、全てのステートメントが実行されます。
	SQL ファイルのインポート	SQL ファイルを SQL エディターにて開きます。

SQL ステートメントとスクリプトを SQL エディターにて作成する

以下の GUI メソッドを使用することで SQL ステートメントやスクリプトを作成することができます:

- **ドラッグとドロップ**: ブラウザーペインから SQL エディターにオブジェクトをドラッグすることで、そのデータベースオブジェクトを問い合わせるためのステートメントが生成されます。
- **コンテキストメニュー**: ブラウザーペインにてオブジェクトをクリックして、コンテキストメニューから「**SQL エディターで表示 | SELECT**」を選択します。
- **手動による入力**: SQL エディターにて SQL ステートメントを入力します。入力には自動補完機能を使用することができます。
- **SQL スクリプトのインポート**: データベースクエリウィンドウのツールバーにある **SQL ファイルのインポート**アイコンをクリックします。

SQL ステートメントの実行

SQL エディターに入力された SQL ステートメントが複数個ある場合、実行するステートメントを選択し、データベースクエリウィンドウのツールバーにある**実行**アイコンをクリックします。SQL スクリプトにてステートメントが選択されていない場合、スクリプト内に入力された全てのステートメントが実行されます。データベースのデータが取得され結果タブのグリッドに表示されます。実行に関するメッセージがメッセージタブに表示されます。

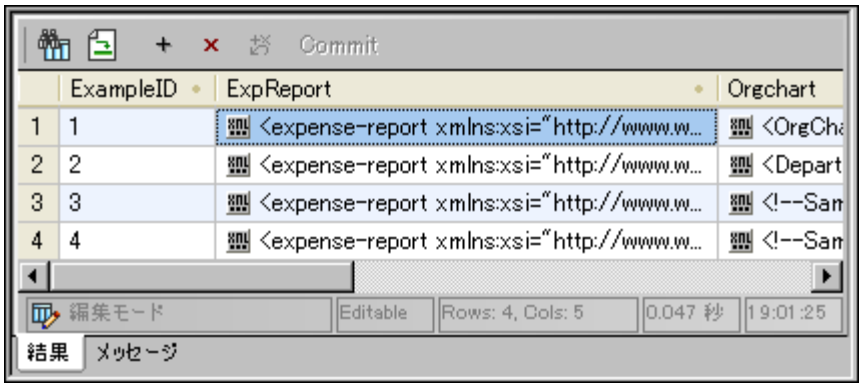
10.7.5 結果とメッセージ

結果/メッセージペインには2つのタブが含まれています:

- 結果タブ にはクエリにより取得されたデータが表示されます。
- メッセージタブ にはクエリの実行に関するメッセージが表示されます。

結果タブ

クエリにより取得されたデータが、結果タブのグリッドに表示されます(以下のスクリーンショットを参照)。



結果タブの適切な場所を右クリックすることにより表示されるコンテキストメニューから、以下のオペレーションを行うことができます:

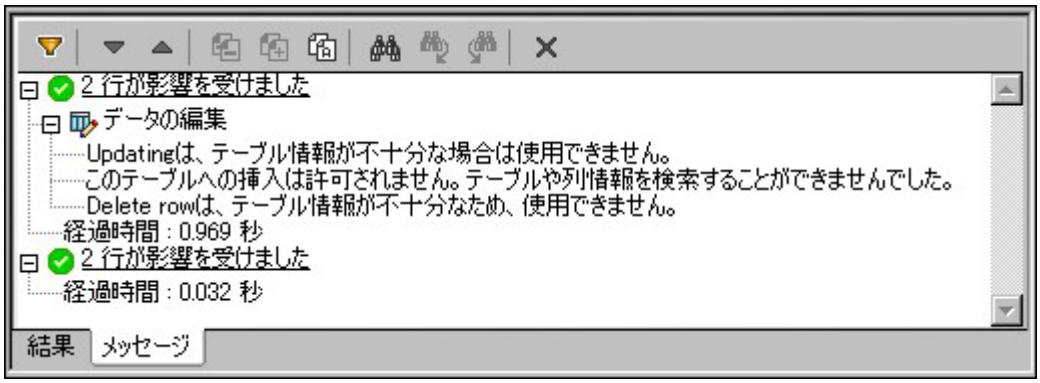
- **カラムによる並び替え** : 並べ替えを行うレコードが含まれているカラムを右クリックして、「**並べ替え | 昇順/降順/デフォルトの復元**」を選択します。
- **クリップボードにコピー** : 最初に (i) データの範囲を選択し、(ii) 選択をコピーします。データを選択するにはいくつかの方法があり、(i) カラムヘッダーや行番号をクリックすることで、カラムや行全体を選択することができるほか、(ii) 個々のセルを選択(Shift や Ctrl キーを使用することで、複数のセルを選択することができます)、(iii) セルを右クリックして、コンテキストメニューから「**選択範囲 | 行/列/全て**」を選択します。選択のマーキングを行い、コンテキストメニューから「**選択されたセルのコピー**」または「**ヘッダーと選択対象のコピー**」を選択します。この操作により、選択対象がクリップボードにコピーされ、他のアプリケーションの処理に渡すことができるようになります。セルとともにヘッダーをコピーするには、**ヘッダーと選択対象のコピー**コマンドを使用してください。

結果タブでは以下のツールバーアイコンを使用することができます:

	ステートメントへ移動	表示されている結果を取得するための SQL エディターにてステートメントがハイライトされます。
	検索	結果ペインにあるテキストを検索します。XML ドキュメントのコンテンツも検索されます。

メッセージタブ

メッセージタブにはそれまでに実行された SQL ステートメントの情報がエラーや警告メッセージとともに表示されます。



メッセージタブのツールバーにはビューのカスタマイズやナビゲーション、クリップボードへのメッセージコピーを行うためのアイコンが含まれています。**フィルター**アイコンにより、特定の種類のメッセージを切り替えることができます。**次**ならびに**前**アイコンにより、選択を移動することができます。子コンポーネントとともに(または含めずに)メッセージをクリップボードへコピーすることもできます。検索語とともに**検索**機能を指定することで、メッセージ内における検索を行うことができます。**クリア**アイコンにより、メッセージペインの内容を破棄することができます。

メモ　これらツールバーアイコンのコマンドは、コンテキストメニューからも利用することができます。

11 Authentic View

Authentic View (*下のスクリーンショット*) ではマークアップ無しでXMLドキュメントが表示されるほか、入力フィールドやコンボボックス、ラジオボタンといったデータ入力機能とともに視覚的に表示されます。ユーザーがAuthentic View にて入力したデータはXML ファイルへ入力されます。

Authentic プレビュー

StyleVision では、SPS の編集を行いながら割り当てられた作業用 XML ファイルのデータを Authentic View にてプレビューすることができます。作業用 XML ファイルがSPS へ割り当てられていない状態でAuthentic View タブをクリックすると、作業用 XML ファイルを割り当てるように促されます。Authentic プレビューではAuthentic View と同様にXMLドキュメントの編集を行うことができ、編集内容は作業用 XML ファイルへ保存されます。このセクションでは Authentic View とAuthentic View 内のドキュメントの編集方法について記述されます。

(C) 2015-2021 Altova GmbH

11.1 Authentic View インターフェイス

Authentic Preview はアクティブなドキュメントのAuthentic タブをクリックすると有効化されます。作業 XML ファイルがSPS に割り当てられていない場合、割り当てるようにプロンプトされます。

このセクションでは以下について説明されています:

- インターフェイスの概要
- Authentic View に特有のツールバーの説明
- メインのAuthentic Preview ウィンドウ
- コンテキストメニューの説明は、XML ドキュメント内のAuthentic View で見つけることができます。

Authentic View 情報の追加ソースは以下のとおりです:

- Authentic View チュートリアルはAuthentic View インターフェイスの使用方法を表示します。このチュートリアルはAltova XMLSpy とAltova Authentic Desktop 製品のドキュメント（チュートリアルのセクションを参照してください）、およびオンラインで見つけることができます。
- Authentic View メニューコマンドの詳細に関しては、次を参照してください: 製品ドキュメントのユーザーレファレンスセクション。

Altova Web サイト: XML コンテンツの編集、XML の作成

11.1.1 GUI の概要

Authentic Preview には、メインウィンドウ内に表示される XML ドキュメントを編集するために使用するメニューコマンド、ツールバーアイコン、とコンテキストメニューがあります。

メニューバー

メニューバー内で使用することのできるメニューは、製品ドキュメントのユーザーレファレンスセクション内で詳しく説明されています。

ツールバー

ツールバー内に表示されるシンボルとアイコンは、Authentic View ツールバーアイコンのセクションで説明されています。

情報ウィンドウ

このウィンドウは現在 Authentic View 内で選択されているノードに関する情報を提供します。

メインウィンドウ

作業 XML ドキュメントが表示され編集されるウィンドウです。セクション Authentic View メインウィンドウ内で詳しく説明されています。

ステータスバー

ステータスバーは現在選択されているノードへのXPath を表示します。StyleVision のAuthentic Preview 内では、現在選択されているノードへのXPath は、現在選択されているノードが灰色で表示されているスキーマツリー内で表示されています。Authentic Preview 内のXPath はステータスバー内では表示されていません。

コンテキストメニュー

メインウィンドウ内で右クリックすると表示されるメニューです。使用することのできるコマンドは、状況に依存した編集コマンドです。すなわち、選択されたノードに関連する構造とコンテンツを操作することができます。このような操作は、ノードの挿入、追加、または、削除、エンティティの追加、またはコンテンツの切り取りと貼り付けを含みます。

11.1.2 Authentic View ツールバーアイコン

Authentic View ツールバー内のアイコンは、コマンドのショートカットです。アイコンの一部は、Windows アプリケーション、または、Altova 製品で既に慣れ親しんだアイコンであり、その他は全く新しいアイコンです。このセクションでは、Authentic View の固有のアイコンについて説明されています。下の説明では、アイコンはグループ化されています。

XML マークアップの表示/非表示

Authentic View 内では、XML 要素、または、属性 の全部（大きなマークアップ）または一部（小さなマークアップ）を名前と共に、または、名前をなしで表示することができます。4 つのマークアップ アイコンがツールバー内に表示され、**Authentic** メニュー内で対応するコマンドを使用できるようになります。

	マークアップを非表示にします。すべての XML タグは既に折りたたまれたものを除いて非表示になります。（展開する通常の方法である）「マークアップモードを非表示にする」内の折りたたまれたタグをクリックすると、ノードのコンテンツが表示されタグが非表示になります。
	小さなマークアップを表示します。XML 要素/属性 タグは名前と共に表示されます。
	大きなマークアップを表示します。XML 要素/属性 タグは名前無しで表示されます。
	複合型マークアップを表示します。StyleVision Power Stylesheet 内では、各 XML 要素、または、属性を表示のために（大きなマークアップ、または、小さなマークアップ）指定することができ、また、すべてを非表示にすることも出来ます。要素の一部がマークアップと共に表示、または、マークアップを隠して表示するように指定されているため、複合型マークアップモードと呼ばれます。複合型マークアップモード内では、Authentic View ユーザーはカスタム化されたマークアップを確認することができます。しかしながら、このカスタム化は StyleVision Power Stylesheet をデザインした個人により作成され、Authentic View ユーザーにより定義することはできません。

動的なテーブル構造の編集

動的な SPS テーブル 内の行はデータ構造の繰り返しです。各行は単一の要素の発生を表示しています。各行は、結果として、次と同じ XML サブ構造を持ちます。

動的な テーブルの編集コマンドは、動的な SPS テーブルの行を操作します。要素の発生の数と順序を変更することができます。しかしながら、個別の要素の発生のサブ構造の変更を意味するため、動的な SPS テーブルの列が編集されます。

動的な テーブル編集コマンドのためのアイコンはツールバー内に表示され、**Authentic** メニュー内でも使用できます。

	テーブルに行を追加する

	テーブルに行を挿入する
	現在のテーブル行を複製する（すなわち、セルコンテンツが複製されます）
	現在の行を上に 1 つ移動する
	現在の行を下に 1 つ移動する
	現在の行を削除する

メモ これらのコマンドは、**動的な SPS テーブル** のみに適用されます。静的な SPS テーブル内で使用することはできません。

DB 行 のナビゲーションアイコン

矢印アイコンは（左側から右側） 最初のレコードに移動する、前のレコードに移動する、レコード # ダイアログに移動する、次のレコードに移動する、最後のレコードに移動する、です。

	このアイコンはクエリを入力することのできる「データベースクエリの変更」ダイアログを開きます。 Authentic View はクエリされたレコードを表示します。

XML データベースの編集

「**編集のための XML データを使用して新規の行を選択する**」コマンドにより、IBM DB2 などの XML DB 内の関連するテーブルから新規の行を選択することができます。この行は、Authentic View 内で表示され、編集することができ、DB に戻し保存することができます。

XML ファイルコマンド

次のアイコン（左側から右側）は下にリストされるコマンドの説明に対応しています:

- *Authentic XML データの保存する*: XML データファイルを保存します。
- *Authentic XML データを名前を付けて保存する*: XML データファイルを他のファイルとして保存します。
- *Authentic View を再ロードする*: 保存された XML データファイルを再ロードします。保存されていない変更は破棄されます。
- *検証する*: XML データファイルを検証します。

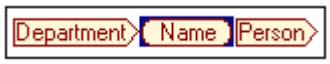
11.1.3 Authentic View メインウィンドウ

Authentic View には4つの表示モードが存在します: 大きなマークアップ、小さなマークアップ、複合型マークアップ、すべての マークアップを隠す。マークアップ情報の多種のレベルをもつドキュメントをこれらのモードで確認することができます。モード間を切り替えるには、**Authentic** メニュー内のコマンド、または、ツールバー内のアイコンを使用します（前のセクションを参照してください: Authentic View ツールバーアイコン）。

大きなマークアップ

タグ内に要素/属性名を持つ要素と属性の開始と終了タグを表示します:

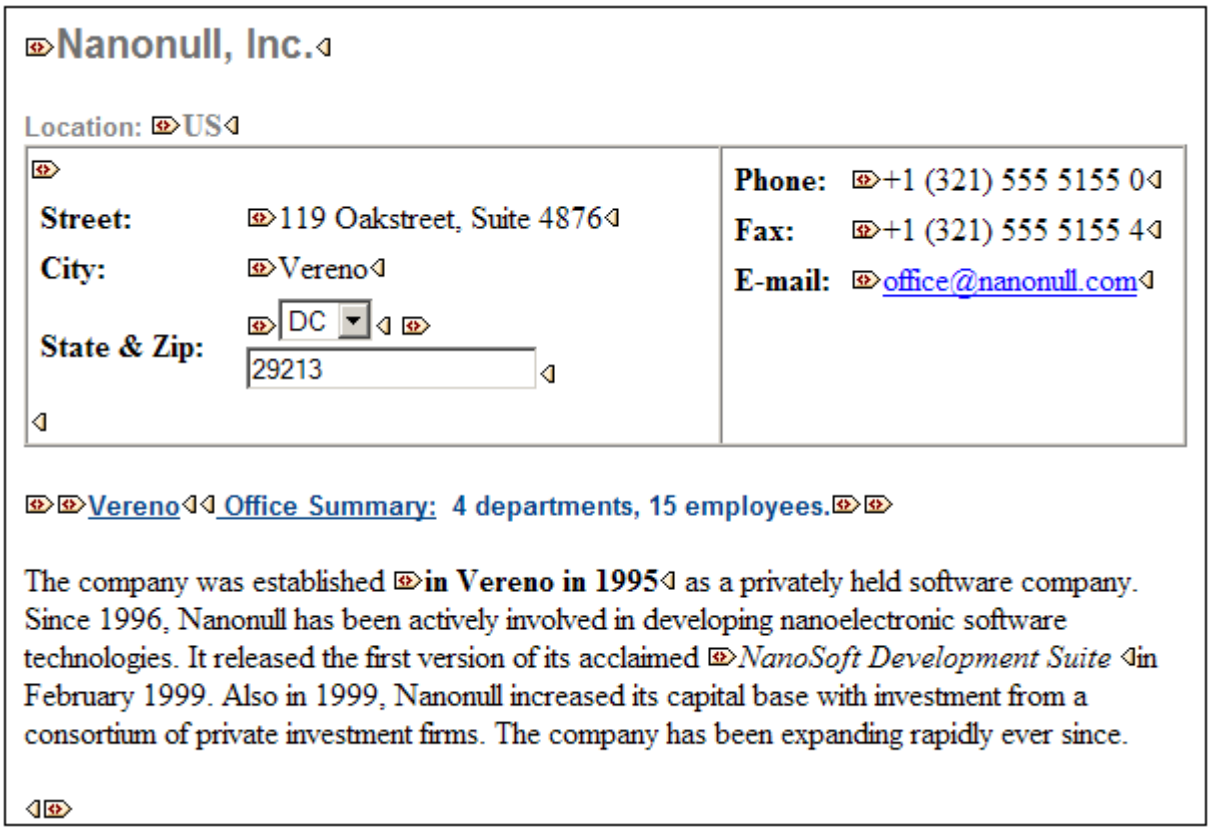
上の図の中では、要素 Name が展開されており、すなわち、開始と終了タグ、および要素のコンテンツが表示されています。要素/属性は、開始、または、終了タグをダブルクリックすることにより、折りたたむことができます。短くされた要素/属性を展開するには、短くされたタグをダブルクリックします。



大きなマークアップ内では、属性は属性の開始と終了タグ内で等号シンボルとして認識されます:

小さなマークアップ

名前の無い要素/属性の開始と終了タグを表示します :



開始タグ内にはシンボルが存在し、終了タグは空であることに注意してください。また、要素タグには角かっこが使用されており、属性タグには等号サインがシンボルとして使用されていることに注意してください（*下のスクリーンショットを参照してください*）。

要素/属性を展開、または、折りたたむには、適切なタグをダブルクリックしてください。下のサンプルは、（青色でハイライトされている）折りたたまれた要素を表示しています。折りたたまれた要素と展開された要素の開始タグの形に注意してください。

複合型マークアップ

複合型マークアップは、カスタム化されたマークアップのレベルを表示しています。StyleVision Power Stylesheet をデザインした個人はドキュメント内の個々の要素/属性のために大きなマークアップ、小さなマークアップ、または、マークアップ無しを決定することができます。Authentic View ユーザーは、内でカスタム化されたマークアップを複合型マークアップ表示モードで確認することができます。

すべての マークアップを非表示にする

すべてのXML マークアップは各されています。Authentic View 内で確認することのできる書式設定は、印刷されたドキュメントの書式であるため、この表示モードはドキュメントのW YSIW YG 表示です。

コンテンツの表示

Authentic View 内では、コンテンツは2 つの方法で表示されています:

- テキスト形式。テキストを入力し、このテキストが要素のコンテンツ、または、属性の値になります。

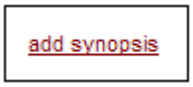
- データ入力デバイス。表示は、入力フィールド（テキストボックス）、複数の入力フィールド、コンボボックス、チェックボックス、または、ラジオボタンのいずれかを含んでいます。入力フィールドと複数の入力フィールドの場合、フィールドに入力するテキストが要素の XML コンテンツ、または、属性の値になります。

  データ入力デバイスの場合、選択はStyleVision Power Stylesheet 内で指定される対応するXML 値を生成します。

  このため、コンボボックス内では、（コンボボックスのドロップダウンリスト内で使用することができる）「許可された選択」選択は「1」の XML 値、または、「許可された選択」、またはその他をマップすることができます。「許可されていない選択」は「0」、「許可されていない選択」、または、その他にマップすることができます。

任意のノード

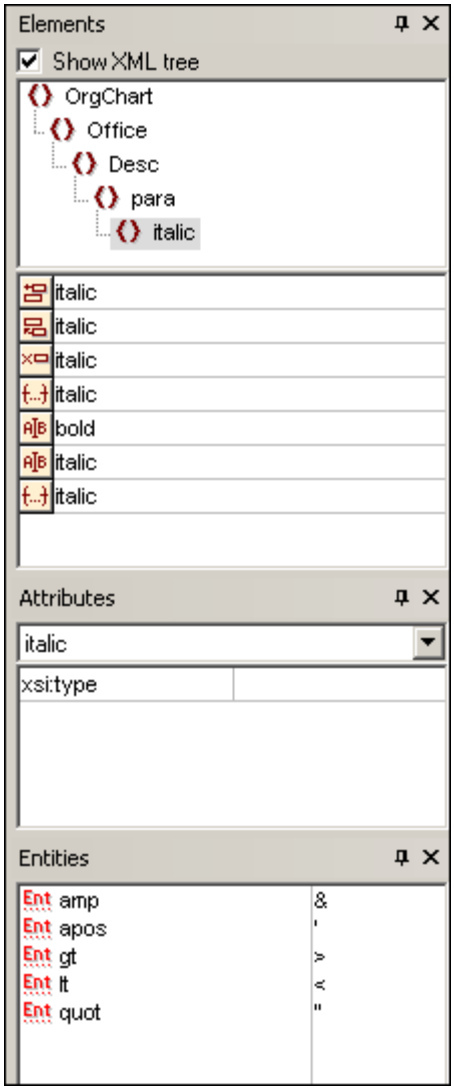
（参照されたスキーマに従い）要素、または、属性が**任意** の場合、型 `add [element/attribute]]` のプロンプトが表示されます:



プロンプトをクリックすると、要素が追加され、カーソルがデータの入力のために表示されます。複数の任意のノードが存在する場合、プロンプト add... が表示されます。プロンプトをクリックすると、任意のノードのメニューが表示されます。

11.1.4 Authentic View エントリヘルパー

Authentic View 内には3つの入力ヘルパーが存在します: 要素、属性、および、エンティティ。Authentic View インターフェイスの右下のウィンドウとしてこれらは表示されています（*下のスクリーンショットを参照してください*）。



要素と属性 入力ヘルパーは状況に依存しています。すなわち、入力ヘルパー内に表示される内容は、ドキュメント内のカーソルの位置により異なります。エンティティの入力ヘルパー内に表示されているエンティティは、状況に依存しません。ドキュメントのために許可されている全てのエンティティはカーソルの場所に関わらず表示されます。

それぞれの入力ヘルパーの説明に関しては下で説明されています。

要素入力ヘルパー

要素入力ヘルパーは2つの部分から構成されています:

- 上の部分には、「XML ツリーを表示」チェックボックスを使用してオンとオフを切り替えることのできるXML ツリーが含まれています。XML ツリーは、現在の要素のドキュメントのルート要素までの祖先を表示しています。XML ツリー内の要素をクリックすると、その要素に対応する要素（このリストの次のアイテム内で説明されているとおり）は要素入力ヘルパーの下の部分に表示されます。
- 挿入することのできるノードのリストを含む下の部分は、Authentic View 内の選択された要素、または、テキストの範囲から挿入、削除またはクリアすることができます。入力ヘルパー内の要素名の左側のアイコンにより入力ヘルパー内にリストされる要素が表示されます。要素入力ヘルパー内に発生するアイコンは意味を説明する詳細と共に下にリストされます。

入力ヘルパーからノードを使用するには、アイコンをクリックします。

要素の後に挿入
入力ヘルパー内の要素は、選択された要素の後に挿入されます。正確な階層レベルに追加されている場合、. 例えば、//sect1/para 要素内にカーソルが存在する場合、sect1 要素を追加すると、新規のsect1 要素が//sect1/para の兄弟の後ではなく、para 要素の親であるsect1 要素の兄弟の後に追加されます。

要素の前に挿入
入力ヘルパー内の要素は選択された要素の前に挿入されます。要素の後に挿入 コマンドと同様、要素は正確な階層レベルとして挿入されます。

要素の削除
要素とコンテンツを削除します。

要素の挿入
入力ヘルパーからの要素を要素内に挿入することもできます。要素内にカーソルが置かれると、その要素の許可されている子要素を挿入することができます。許可されている子要素は要素のみのコンテンツモデルの一部コンテンツモデル、および、複合型 コンテンツモデル（テキストと子要素）となることができます。

テキストの範囲が選択されていると、許可されている子要素がテキスト内で入力ポイントとしてカーソルがポイントされている場所に挿入されます。

- テキストの範囲が選択され、要素が挿入されると、テキストの範囲は挿入された要素のコンテンツになります。
- 要素が入力ポイントに挿入されると、要素はそのポイントに挿入されます。

要素の挿入後、これらのインラインの要素のための表示される（要素入力ヘルパー内の）要素アイコンをクリックすることによりクリアすることができます。テキストの範囲、または、入力ポイントとしてテキスト内にカーソルをポイントするかにより表示されるアイコンが決定されます（下を参照してください）。

要素の適用
（大きなマークアップの表示 ビュー内の開始、および終了タグをクリックすることにより）ドキュメント内の要素を選択する場合、および、要素が他の要素により置き換えられる場合 （例えば、para などの複合型 コンテンツ 要素内では、italic 要素 がbold 要素と置き換えられることができます）、このアイコンは入力ヘルパー内の要素が選択された（オリジナルの）要素に適用することができることを示します。**要素の適用** コマンドを複合型 コンテンツの要素内のテキストの範囲に適用することもできます。テキストの範囲は適用された要素のコンテンツとして作成されます。

- 適用された要素が元の要素の子と**同じ名前を持つ子要素** を持つ場合、また、元の要素内でこの子要素のインスタンスが存在する場合、オリジナルの子要素が新規の要素のコンテンツ内で保持されます。
- 適用された要素が元の要素のインスタント化された子と**同じ名前を持たない子要素** を持つ場合、元の要素のインスタント化された子が新規の要素が持つ子要素、または、要素の兄弟として追加されます。

- 適用された要素が元の要素のコンテンツモデル内に**等価が存在しない子要素**を持つ場合、この子要素は直接作成されず、Authentic View が挿入するオプションを提供します。

要素ではなくテキストの範囲が選択されている場合、要素の選択への適用は、コンテンツとして選択されたテキストの範囲と共にその場所で適用された要素を作成します。カーソルが入力ポイントの場合要素の適用は許可されていません。

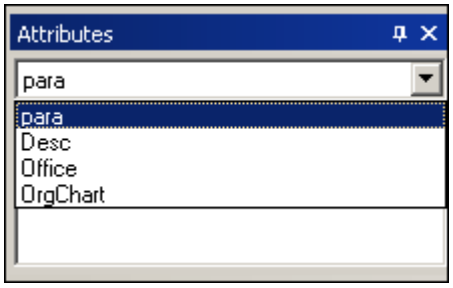
要素をクリアする
このアイコンは、複合型コンテンツの要素内のテキストが選択されると表示されます。アイコンをクリックすると選択されたテキストの範囲の周りから要素をクリアします。

要素をクリアする（挿入ポイントが選択されている場合）
複合型の子-コンテンツ要素である要素内にカーソルが置かれるとこのアイコンが表示されます。アイコンをクリックするとインラインの要素がクリアされます。

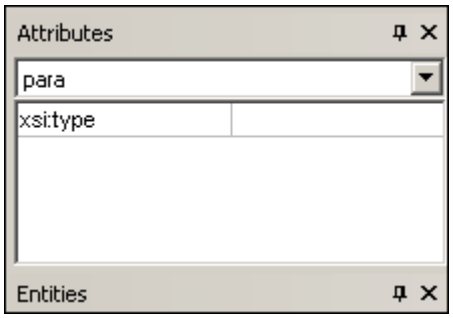
属性入力ヘルパー

属性入力ヘルパーはドロップダウンコンボボックスと属性のリストから構成されています。選択した要素がコンボボックス内に表示されます（開始、または、終了タグをクリック、または、要素コンテンツ内にカーソルをポイントして選択します）。

下に示されている属性入力ヘルパーにはコンボボックス内の `para` 要素が表示されています。コンボボックス内の矢印をクリックとすると、この場合 `OrgChart` である、`para` 要素の**祖先 ドキュメントのルート要素までの祖先** を表示するドロップダウンリストが表示されます。



コンボボックスの下に、要素のための有効化属性のリストが表示されます。この場合、`para` のためのリストが表示されます。要素で属性が必須の場合、太字で表示されます。（下のサンプルでは、内蔵の属性 `xsi:type` 以外の必須の属性は存在しません）。



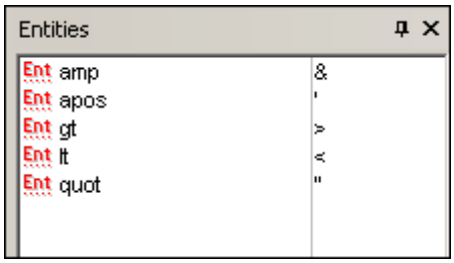
属性のために値を入力するには、属性の値フィールドをクリックして、値を入力します。これにより属性とその値がXMLドキュメント内に作成されます。

以下に注意してください：

- nillable 要素が選択された場合属性入力ヘルパー内に表示される xsi:nil 属性の場合、xsi:nil 属性の値は属性の値のためのドロップダウンリストから許可されている値（true、または、false）の1つを選択して入力することができます。
- xsi:type 属性を属性の値フィールド内をクリックして変更することができ、ドロップダウンリストからリストされた値を選択することができます。リストされた値は、Authentic View ドキュメントがベースとされる XML スキーマ内で定義された使用することのできる抽象型です。

エンティティの入力ヘルパー

エンティティの入力ヘルパーによりドキュメントにエンティティを挿入することができます。エンティティを使用してドキュメントに（企業名など）頻繁に発生する特殊文字、または、テキストフラグメントを挿入することができます。エンティティを挿入するには、エンティティを挿入するテキストにカーソルを置き、エンティティの入力ヘルパー内のエンティティをダブルクリックします。



メモ　内部のエンティティは DTD 内で定義された値を持ちます。外部エンティティは他の XML ファイルなどの外部ソース内に含まれる値を持ちます。内部と外部エンティティはエンティティの入力ヘルパー内にリストされています。エンティティを挿入すると、— 値ではなく — 内部の、または、外部のエンティティが XML テキストに挿入されます。エンティティが内部のエンティティの場合、Authentic View **エンティティの値** を表示します。エンティティが外部エンティティの場合、Authentic View は値ではなくエンティティを表示します。これは、Authentic View 内で表示される外部エンティティである XML ファイルを意味します。コンテンツは Authentic View 表示内のエンティティを置き換えません。

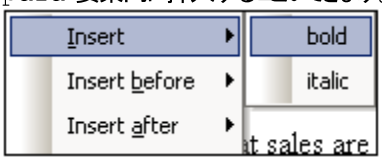
Authentic View 内で自身のエンティティを定義することができ、これらは入力ヘルパー内で表示されます: Authentic View セクション内の編集のエンティティの定義を参照してください

11.1.5　Authentic View コンテキストメニュー

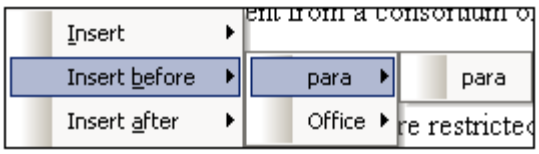
選択されたドキュメントのコンテンツ、選択に関連したコマンドを使用して表示されるコンテキストメニュー、または、カーソルの場所を右クリックします。

要素の挿入

下の図は現在のカーソルの位置で挿入することのできる全ての要素リストである**サブメニューの挿入** を表示しています。**サブメニューの前に挿入** は現在の要素の前に挿入することのできる全ての要素をリストしています。**サブメニューの後に挿入** サブメニューは現在の要素の後に挿入することのできる全ての要素をリストしています。下の図では現在の要素は para 要素です。bold と italic 要素は、現在の para 要素内に挿入することができます。



下で確認することができるように、para と Office 要素は現在の para 要素の前に挿入することができます。



ノードの挿入、置換え、(適用)、およびマークアップの削除 (クリア) コマンドはコンテキストメニュー内、および Authentic View 入力ヘルパー内で使用することができそれぞれのセクションで説明されています。

エンティティの挿入

「エンティティの挿入」コマンドにカーソルをポイントすると、宣言されている全てのエンティティのリスト含むサブメニューが表示されます。エンティティをクリックすると、選択された場所にエンティティが挿入されます。次を参照してください: ドキュメントのためにエンティティを定義する方法に関してはエンティティの定義 を参照してください。

CDATA セクションの挿入

このコマンドは、カーソルがテキスト内にポイントされると有効化されます。クリックするとカーソルの位置で CDATA セクションが挿入されます。CDATA セクションは開始と終了タグにより区切られています。大きな、小さなマークアップをオンにしてこれらのタグを確認することができます。CDATA セクション内では、XML マークアップと解析は無視されます。XML マークアップ文字(アンパサンド、アポストロフィ、より大記号 (>)、より小記号 (<)、および引用符) はマークアップとして扱われず、リテラルとして扱われます。ですから、CDATA セクションは、XML マークアップ文字を持つプログラムコードリスティングなどのテキストのために役にたちます。

ノードの削除

「削除」コマンドにマウスのカーソルをポイントすると、選択されたノードにより構成されたメニューリストとドキュメントを無効化しない)削除可能な祖先すべてが表示されます。削除する要素をクリックします。これは要素、または、削除可能な祖先を簡単に削除する方法です。祖先要素をクリックすると、選択されて要素を含む全ての子孫が削除されます。

クリア

「クリア」コマンドは要素マークアップを選択の周りからクリアします。ノード全体が選択されると、要素 マークアップはノード全体のためにクリアされます。テキストセグメントが選択されると、要素 マークアップがそのテキストセグメントのみのためにクリアされます。

適用

「適用」コマンドはメインウィンドウ内の選択の選択された要素に適用されます。詳細に関しては、次を参照してください: Authentic View 入力ヘルパー。

コピー、切り取り、貼り付け

これらは標準ウィンドウコマンドですが「貼り付け」コマンドはコピーされたテキストを SPS 全体のために指定されたスタイルシートのデザイナーにより XML、または、テキストとして貼り付けます。「XML としてコピー」と「テキストとしてコピー」コマンドの詳細に関しては、次を参照してください: 下記の「貼り付け」コマンドの詳細。

貼り付け

「貼り付け」コマンドは Authentic View XML フラグメントにクリップボードにコピーされた内容を) XML、または、テキストとして張り付けるオプションを提供します。コピーされたフラグメントが XML として貼り付けられると、XML マークアップと共に貼り付けられます。テキストとして貼り付けられると、(XML マークアップではなく)コピーされたフラグメントのテキストコンテンツのみが貼り付けられます。次のシチュエーションが可能です:

- *マークアップタグと共にノード全体* が Authentic View 内でハイライトされ、クリップボードにコピーされます。(i) ノードはこのノードが有効な場所で XML として解析されます。無効な場所では解析されません(ii) ノードがテキストとして解析されると、(マーク

アップではなく)ノードの *テキストコンテンツのみ*が貼り付けられます。テキストを貼り付けることのできるXMLドキュメント内にテキストコンテンツを貼り付けることができます。

- *テキストフラグメント* がAuthentic View 内でハイライトされ、クリップボードにコピーされます。(i)フラグメントが貼り付けられる場所でXML ノードが有効な場合のみ、このフラグメントがXML として貼り付けられる場合、テキストのXML マークアップタグは、テキストフラグメントと共に明示的にコピーされず、テキストと共に貼り付けられます。(ii) フラグメントがテキストとして貼り付けられる場合、テキストが貼り付けられるXMLドキュメント内の場所に貼り付けられます。

メモ　テキストは、テキストが許可されているノードにコピーされます。コピーされたテキストがドキュメントを無効化しないようにご注意してください。コピーされたテキストは以下のようになります:

(i) 新規の場所で構文的に有効な場合(例えば、数字のノード内の数字以外の文字 が無効になります)。
(ii) 以外はノードを無効化する場合(例えば、3つの桁のみ受け入れるノード内の4つの桁はノードを無効化します)。

メモ　解析されたテキストがドキュメントを無効化する場合、テキストが赤い色で表示されます。

削除

「**削除**」コマンドは選択されたノードとそのコンテンツを削除します。ノード内にカーソルをポイント、または、ノードの開始、または、終了タグをクリックすることによりノードは選択されているとして考えられます。

11.2 Authentic View 内で編集する

このセクションはAuthentic View の重要な機能について詳しく説明しています。機能は頻繁に使用されるため、またはは、メカニズム、または、概念について説明が必要なため、これらの機能はこのセクションに含まれています。

セクションでは以下について説明されています:

- 日付の選択は、日付をクリックすると、正確な XML 書式で日付を入力する視覚的なカレンダーです。次を参照してください: 日付の選択。
- エンティティは、特殊文字、またはは、テキスト文字列の短縮形です。これらの特殊文字 、またはは、テキスト文字列を挿入するために、対応するエンティティを挿入することにより、自身のエンティティを定義することができます。 詳細に関しては、次を参照してください: エンティティの定義
- Authentic View で表示することのできる イメージ 。

Authentic View の全ての機能の使用方法を学ぶために、Authentic View チュートリアルを XMLSpy、またはは、Authentic Desktop を使用して行ってください。Authentic View チュートリアルへこれらの製品と共に使用することができます。

Altova W eb サイト: XML コンテンツの編集、XML の作成

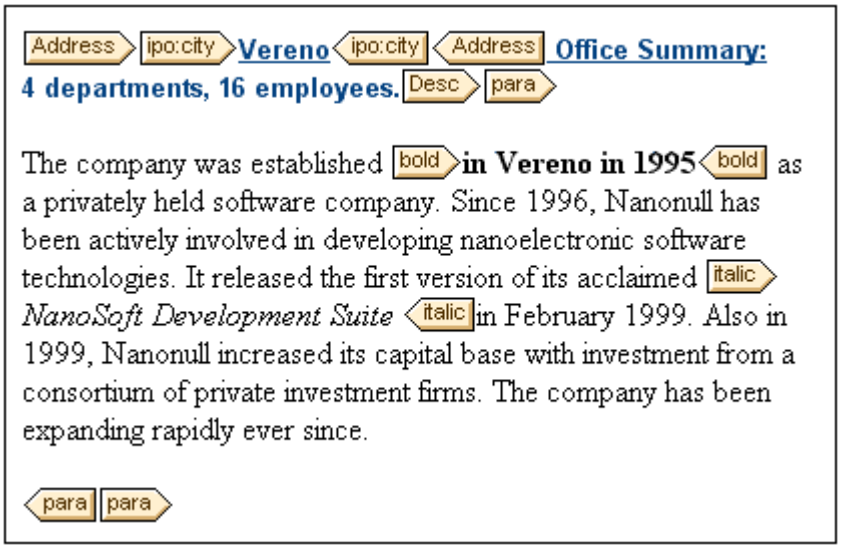
11.2.1 基本的な編集

Authentic View 内で編集を行う場合、XML ドキュメントが編集されます。Authentic View は、ドキュメントの構造的な XML マークアップを非表示にすることができ、ドキュメントのコンテンツのみを表示することができます (*下最初のスクリーンショット*)。 XML の技術的な側面に触れることなく、ドキュメントをノーマルなテキストドキュメントと同様に編集することができます希望する場合、編集中にマークアップをオンに切り替えることもできます (*下の2番目のスクリーンショット*)。

Vereno Office Summary: 4 departments, 16 employees.

The company was established **in Vereno in 1995** as a privately held software company. Since 1996, Nanonull has been actively involved in developing nanoelectronic software technologies. It released the first version of its acclaimed *NanoSoft Development Suite* in February 1999. Also in 1999, Nanonull increased its capital base with investment from a consortium of private investment firms. The company has been expanding rapidly ever since.

XML マークアップの存在しない編集することのできる Authentic View ドキュメント

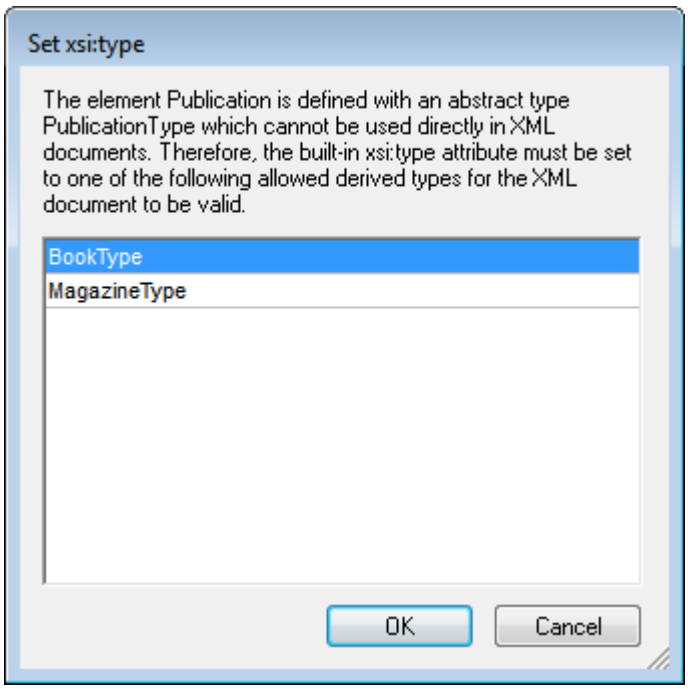


XML マークアップタグが存在する編集することのできる Authentic View ドキュメント

ノードの挿入

新規のノードをAuthentic XMLドキュメントに追加する必要がある場合があります。例えば、新規の`Person`要素はドキュメントのアドレスブックの型に追加される必要があります。このような場合、XML スキーマは新規の要素の追加を許可します。新規のノードを追加する箇所の後にAuthentic View ドキュメント内でノードをクリックするだけです。表示されるコンテキストメニュー内で、必要に応じて「**前に挿入**」、または、「**後に挿入**」を選択します。ドキュメントに挿入することのできるノードはサブメニュー内にリストされます。必要とされるノードをクリックして挿入します。全ての必須の子孫ノードも挿入されます。子孫ノードが任意の場合、クリックすることのできるリンク`Add NodeName`が表示され、任意のノードを追加することができます。

追加されるノードが抽象型の要素の場合、(*下のスクリーンショットで表示されるような*) XML スキーマ内で使用することのできる生成された型のリストを含むダイアログが表示されます。



Publication 要素が追加されると、上のスクリーンショットが表示されます。Publication 要素は抽象的な複合型である型 PublicationType です。2つの複合型 BookType とMagazineType は抽象型 PublicationType から派生します。結果として、Publication 要素がXMLドキュメントに追加されると、Publication の抽象型から派生した2つの具体型の1つが指定される必要があります。新規のPublication 要素はxsi:type 属性に追加されます:

```
<Publication xsi:type="BookType"> ... </Publication>
<Publication xsi:type="MagazineType"> ... </Publication>
...
<Publication xsi:type="MagazineType"> ... </Publication>
```

使用することのできる生成された型の1つを選択し、「OK」をクリックすると以下を行うことができます:

- 選択された生成された型を要素のxsi:type 属性の値として設定します。
- 選択された生成された型のコンテンツモデル内で定義された子孫ノードと共に要素を挿入します。

属性入力ヘルパー内の要素のxsi:type 属性の値を変更することにより、選択された生成された型は後で変更することができます。要素の型がこのように変更されると、前の型のコンテンツモデルのノードは、削除され新しいコンテンツモデルのノードが挿入されます。

テキストの編集

Authentic View ドキュメントは、基本的にテキストとイメージにより構成されています。ドキュメント内のテキストを編集するには、テキストを挿入する箇所にカーソルを置き、編集します。(「**削除**」キーなどの)キーストロークとドラッグアンドドロップ機能を使用して、テキストのコピー、移動、削除を行うことができます。「**Enter**」キーは唯一の例外です。Authentic View ドキュメントは書式設定済みのため、追加のラインやスペースをアイテムの間に追加することができません。Authentic View 内の「**Enter**」キーは、このため、現在編集されている要素の他のインスタンスに追加するためにしようされ、この目的のみに使用することができます。

XML またはテキストとしてコピーする

XML、または、テキストとしてコピーし貼り付けることができます。

- テキストがXML として解析されると、XML マークアップは、ノードのテキスト コンテンツと共に解析されます。XML マークアップは、ノードのコンテンツの一部がコピーされた場合でも、貼り付けられます。マークアップを貼り付けるには、貼り付ける場所でスキーマに従い、貼り付けが許可されている必要があります。
- テキストがテキストとして解析されると、XML マークアップ破壊席されません。

XML 、または、テキストとして貼り付けるには、テキスト(**Ctrl+C**)をコピーし、テキストを貼り付ける箇所を右クリックし、コンテキストメニューコマンド「**貼り付け | XML**」、または、「**貼り付け | テキスト**」. を選択します。ショートカット「**Ctrl+V**」が使用される場合、テキストは、デフォルトのSPS のデフォルトの貼り付けモードで貼り付けられます。SPS のデザイナーにより貼り付けモードは指定することができます。詳細に関しては、次を参照してください: セクション コンテキストメニュー.

更に、ハイライトされテキストを貼り付ける箇所にドラッグします。テキストがドロップされるとポップアップが表示され、テキストをテキストまたは、XML として貼り付けるかが問われます。希望するオプションを選択します。

テキストのフォーマット

XMLドキュメントのシステムの基本的な概念は、プレゼンテーションとコンテンツは区別されるということです。XMLドキュメントにはコンテンツが含まれますが、スタイルシートにはプレゼンテーション(書式設定)が含まれています。Authentic View 内では、XMLドキュメントはスタイルシートにより表示されています。Authentic View 内で表示されるすべての書式設定はスタイルシートにより作成されていることを意味します。太字のテキストが表示されると、スタイルシートにより太字の書式設定が提供されます。リスト書式、または、テーブル書式を表示するリスト、または、テーブルはスタイルシートにより提供されています。Authentic View 内で編集するXMLドキュメントにはコンテンツのみが含まれており、書式設定は含まれていません。スタイルシートに書式設定が含まれています。Authentic View ユーザーは編集するテキストの書式設定をする必要が無い、または、できないことを意味します。コンテンツのみが編集されます。コンテンツに自動的に適用される書式設定は、データのセマンティクスおよび/または構造的な値にリンクされています。例えば、(セマンティックな単位として考えることのできる)電子メールアドレスは電子メールであるため、特定の方法で自動的に書式設定されます。同様に、ヘッドラインもドキュメント(構造的な及び構文的

なユニットを含む内の特定の場所で発生する必要があり、スタイルシートデザイナーがそのヘッドラインのために指定する書式に自動的に書式設定されます。電子メールアドレス、または、ヘッドラインの書式設定を変更することができます。電子メールアドレス、または、ヘッドラインのコンテンツを編集するだけです。

一部の場合、コンテンツは特別に表示される必要があります。例えば、テキスト文字列を太字で表示するなど。すべてのこのような場合、プレゼンテーションはドキュメントの構成要素に結び付けられている必要があります。例えば、表示するテキスト文字を太字で表示するなど、太字で表示することのできるスタイルシートデザイナーによるマークアップにより周りのコンテンツから構造的に区別される必要があります。Authentic View ユーザーとして、このようなテキスト文字列を使用する場合テキスト文字列を適切な要素マークアップで囲む必要があります。これを行うための情報に関しては、次を参照してください: ドキュメントの要素入力ヘルパー セクション内の要素の挿入 コマンド

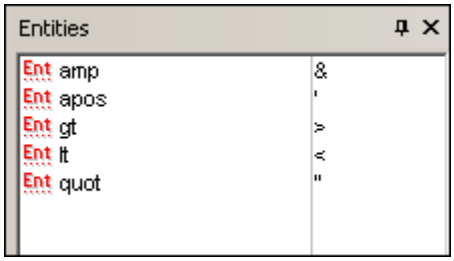
Authentic View 内で RichEdit を使用する

Authentic View 内で、RichEdit ツールバーとして作成された要素の中にカーソルが置かれている場合、ボタンとコントロールは有効化されます(*下のスクリーンショット*)。それ以外の場合、グレーアウトされます。

RichEdit ツールバーのボタンとコントロールを使用して適用するスタイルを指定し、スタイルするテキストを選択します。RichEdit により Authentic View ユーザーはフォント、フォントの大きさ、フォントのスタイル、フォントの飾り、色、テキストの背景色と配置を指定することができます。スタイルされたテキストはスタイル要素のタグに囲まれています。

エンティティの挿入

XML ドキュメント内では、文字の一部はマークアップのために保管されており、ノーマルなテキストでは使用することができません。これらの文字はアンパサンド (&)、アポストロフィ (')、より小記号 (<)、より大記号 (>)、引用符 (") 文字です。これらの文字をデータ内で使用する場合、エンティティの入力ヘルパーを使用して、エンティティレファレンスとして挿入します (*下のスクリーンショット*)。



XML はカスタムのエンティティを作成する機会を与えます。

これらは以下であることができます: (i) キーボードで使用することのできない特殊文字。(ii) ドキュメントのコンテンツ内で再利用するテキスト文字列。(iii) XML データフラグメント、または、(iv) イメージなどの他のソース。Authentic View アプリケーション内でエンティティを自身で定義することができます。定義後、これらのエンティティはエンティティの入力ヘルパー内に表示され、ドキュメントに挿入されます。

CDATA セクションの挿入

CDATA セクションは XML データとして XML パーサーが処理しない XML ドキュメント内のテキストのセクションです。エンティティレファレンスによる特殊文字の置換えを希望しない場合テキストの大きなセクションをエスケープするために使用することができます。例えば、マークアップタグを使用して再生成するプログラムコード、または、XML フラグメントです。要素のコンテンツ内で CDATA セクションは発生することができ、開始と終了で `<![CDATA[` and `]]>` を使用してそれぞれ区切ることができます。この結果、テキスト文字列 `]]>` はセクションの終わりを完了前に示すため CDATA セクションは発生しません。この場合、より大きい文字はエンティティレファレンス(`>`) によりエスケープされるべきです。要素内に CDATA セクションを入力するには、希望する場所にカーソルを置き、右クリックし、コンテキストメニューから CDATA セ

ケションの挿入を選択します。CDATA セクションタグを Authentic View 内で確認するには、マークアップの表示オンに切り替えます。更に、 CDATA セクション内に含まれるテキストをハイライトし、**CDATA セクションの挿入** コマンドを選択します。

メモ CDATA セクションを入力フィールド(すなわち、テキストボックスと複数のテキストボックス)に挿入することはできません CDATA セクションは、Authentic View にテキストコンテンツコンポーネントとして表示される要素内にのみ入力することができます。

リンクの編集

ハイパーリンクは2つの部分から構成されています: リンクテキストとリンクのターゲット。リンクテキストをテキスト内をクリックして編集することにより編集することができます。リンクのターゲットを編集することはできません。((静的な ターゲットアドレス、または、ターゲットアドレスを XML ドキュメント内に含まれるデータを生成することにより)スタイルシートのデザイナーによりリンクのターゲットが設定されます。) Authentic View からは、「Ctrl」を押してをクリックしてリンクのターゲットに移動することができます。(留意点: リンクをクリックするだけで、リンクテキストを編集するためのセットアップを行うことができます)。

11.2.2 Authentic View 内のテーブル

3つのテーブルの種類は、2つのカテゴリに分けることができます: SPS テーブル(静的と動的) とCALS/HTML テーブル。

SPS テーブルには2つの種対があります: 静的と動的。XML ドキュメントがリンクされている StyleVision Power Stylesheet のデザイナーによりSPS テーブルはデザインされます。XML ドキュメント内にSPS テーブルを挿入することはできず、SPS テーブルフィールドにデータを挿入し、動的な SPS テーブルに行を追加、または、削除することができます。SPS テーブルのセクションでは、はこれらのテーブルについて説明されています。

CALS/HTML テーブルは Authentic View のユーザーにより挿入されます。目的は、ドキュメント階層構造内で希望する場所にテーブルを挿入することです。CALS/HTML テーブルとCALS/HTML テーブル編集アイコン の編集機能に関しては、下で説明されています。

11.2.2.1 SPS テーブル

Authentic View 内で2つの種類のSPS テーブルが使用されます: 静的なテーブルと動的なテーブル。

静的なテーブル

静的なテーブルは構造内およびセルのコンテンツ型内で固定されています。Authentic View のユーザーは、テーブルセルにデータを挿入することができますが、これらのテーブルの構造を変更(すなわち、行や列の追加)すること、セルのコンテンツ型の変更することはできません。テキスト内にタイプしてデータを入力、または、チェックボックス、または、ラジオボタンのフォーム内のオプションから選択して、または、コンボボックスのリストから選択してデータを入力することができます。データを入力すると編集することができます。

Nanonull, Inc.			
Street:	119 Oakstreet, Suite 4876	**Phone:**	+1 (321) 555 5155
City:	Vereno	**Fax:**	+1 (321) 555 5155 - 9
State & Zip:	DC 29213	**E-mail:**	office@nanonull.com

メモ 動的なテーブルを編集するためのアイコン、または、コマンドは、静的なテーブルを使用する際に使用することはできません。

動的なテーブル

動的なテーブルには繰り返しデータ構造を表す行が存在します。すなわち、（静的なテーブルの場合は異なりますが）各行は同一のデータ構造を持ちます。結果として、行操作を行うことができます: 行の追加、行の挿入、行を上に移動する、行を下に移動する、行の削除。これらのコマンドは**Authentic** メニューとツール バー内のアイコンとして使用することができます（下に表示される）.

これらのコマンドを使用するには、適切な行内にカーソルをポイントし、必要とするコマンドを選択します。

Administration

First	Last	Title	Ext	EMail	Shares	Leave		
						Total	Used	Left
Vernon	**Callaby**	Office Manager	581	v.callaby@nanonull.com	1500	25	4	21
Frank	Further	Accounts Receivable	471	f.further@nanonull.com	0	22	2	20
Loby	Matise	Accounting Manager	963	l.matise@nanonull.com	add Shares	25	7	18
Employees: 3 (20% of Office, 9% of Company)					**Shares: 1500 (13% of Office, 6% of Company)**			
Non-Shareholders: Frank Further, Loby Matise.								

テーブル内のセルを移動するには、「上」、「下」、「右」、「左」矢印キーを使用します。前方の次のセルに移動する場合は、「タブ」キーを使用します。最後の行の最後のセル内の「タブ」キーを押すと、新規の行が作成されます。

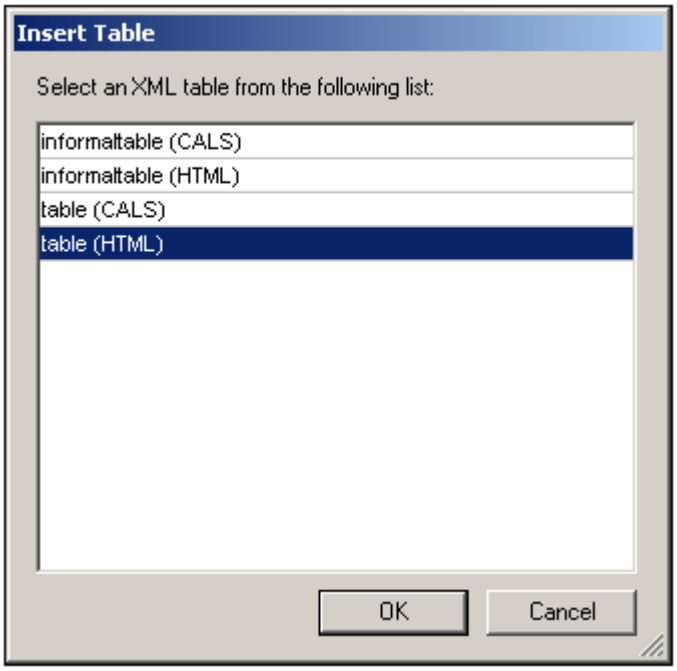
11.2.2.2 CALS/HTML テーブル

一部のXML データ構造がテーブル書式を表示するように指定されているため、Authentic View のユーザーはCALS/HTML テーブルを挿入することができます。CALS/HTML テーブルと作業するには3 つのステップが含まれています: テーブルの挿入。書式設定。データの挿入。CALS/HTML テーブルと作業するためのコマンドはツール バー内のアイコンとして使用することができます（次を参照してください: CALS/HTML テーブル編集アイコン）。

テーブルの挿入

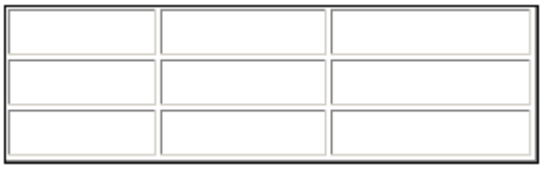
CALS/HTML テーブルを挿入するには、以下を行います:

1. テーブルを挿入する箇所にカーソルをポイントし、アイコンをクリックします。（スキーマによりテーブルを挿入する箇所は決定されることに注意してください）。テーブルの挿入 ダイアログが表示されます（*下のスクリーンショット*）。このダイアログはテーブル構造が定義されているはすべてのXML 要素 データ構造をリストします。例えば、下のスクリーンショットではinformal テーブル要素とテーブル要素はCALS テーブル、および、HTML テーブルと定義されています。



2. 挿入する要素とテーブルモデルを含むエントリを選択し、「OK」をクリックします。
3. 次のダイアログ内では（*下のスクリーンショット*）列と行の数量を選択し、テーブルに追加されるヘッダーおよびまたはフッターを選択し、使用することのできる幅全てをテーブルが使用するかを指定します。完了すると、「OK」をクリックします。

上に示されるダイアログボックス内の仕様のために、以下のテーブルが作成されました。



テーブル メニューコマンドを使用して、列を追加、または、削除、列を作成、列をジョインし分割することができます。開始するには、広義の構造を作成する必要があります。

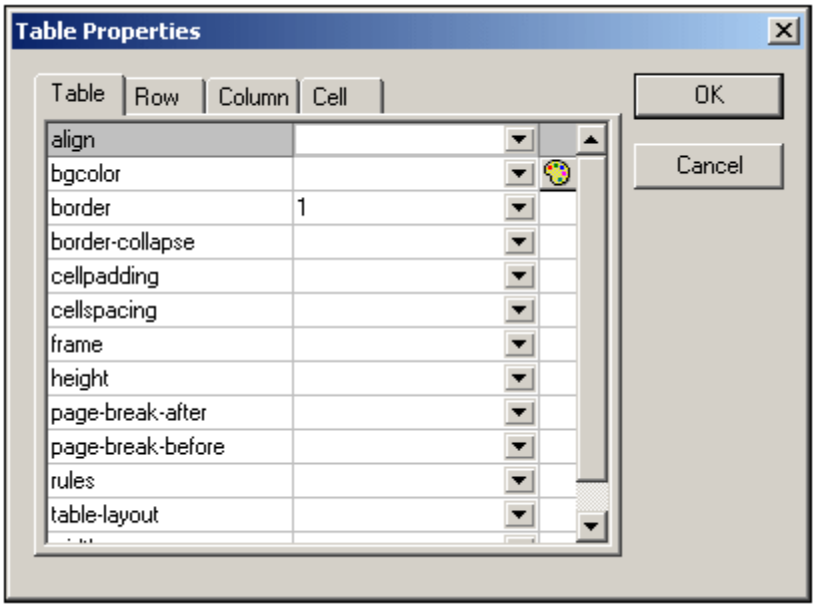
書式設定 テーブルとデータの挿入

テーブル書式設定はドキュメントデザインに既に割り当てられていますが、特定の環境ではテーブル書式設定を変更することができます。これらの環境は、以下のとおりです:

- 多種のテーブル構造要素に対応する要素は属性（基になるXML スキーマ内の）として定義されている適切な CALS、または、HTML テーブルプロパティを持つ必要があります。定義されている属性のみが書式設定のために使用できます。デザイン内で、値がこれらの属性のために設定されている場合、Authentic View 内でこれらの値をオーバーライドすることができます。
- デザイン内では、CSS スタイルを含むスタイル属性を設定してはなりません。CSS スタイルを持つスタイル属性が要素のために指定されている場合、スタイル属性その要素の他の書式設定の属性設定に対して優先順位が与えられます。 この結果、Authentic View 内で指定される書式設定は上書きされます。

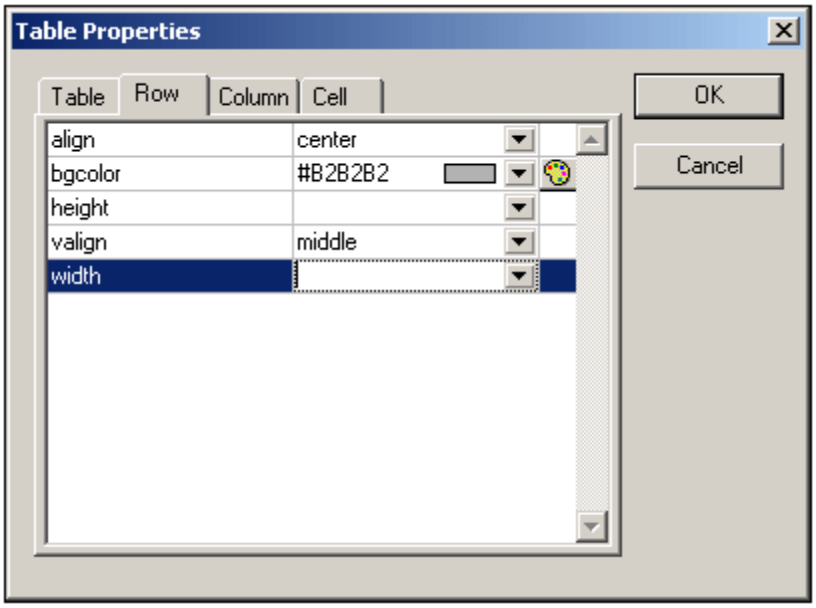
テーブル、行、列、または、セルの書式設定を行うには

1. テーブル （テーブルプロパティ）アイコンにカーソルを置きクリックします。テーブル、または、行、列、または、セルのための書式設定を行うテーブルプロパティダイアログが開かれます（*スクリーンショットを参照してください*）。



2. セルペースとセルパディングプロパティを「0」に設定します。テーブルは以下のようになります:

3. 最初の行にカーソルをポイントし、（テーブルプロパティ）アイコンをクリックします。「行」タブをクリックします。



最初の行がヘッダー行になるため、この行を他の行と区別するために背景色を設定します。上の図で行プロパティが設定されていることに注意してください。列ヘッダーテキストを挿入します。テーブルは以下のようになります:

Name	Telephone	Email

配置は指定されているとおり中央に揃えられています。

4. 「Telephone」列をサブ列「Office」と「Home」に分割する場合、「Telephone」列の幅を水平2つの列に分割する必要があります。最初にしかしながら、サブヘッダー行を作成するためにヘッダーセルの垂直に分割します。「Telephone」セル内にカーソルを置き、 （垂直に分割）アイコンをクリックします。テーブルは以下のようになります:

Name	Telephone	Email

5. 「Telephone」を含むセルの下のセルにカーソルを置き、 （水平に分割）アイコンをクリックします。列ヘッダー「Office」と「Home」を入力します。テーブルは以下のようになります:

Name	Telephone		Email
	Office	Home	

「Telephone」列内の各セルの幅を水平に分割します。

テーブル編集アイコンを使用して、列と行を追加、削除、垂直に整列することもできます。 CALS/HTML テーブル編集アイコンは、CALS/HTML テーブル編集アイコン内で説明されています。

テーブル内のセルを移動する

CALS/HTML テーブル内でセル間を移動するには、「上」、「下」、「右」、「左」矢印キーを使用します。

セルにデータを挿入する

セルにデータを入力するには、セル内にカーソルを置き、データを入力します。

テキストの書式設定

他のテキストXML ドキュメント内同様、CALS/HTML テーブル内のテキストは、 XML 要素、または、属性を使用して書式設定される必要があります。 要素を追加するには、 テキストをハイライトし、要素入力ヘルパー内の必要とされる要素をダブルクリックします。 属性の値を指定するには、テキストフラグメント内にカーソルを置き、属性入力ヘルパー内に必要とされる属性の値 を挿入します。書式設定 ヘッダーテキストを太字に設定すると、テーブルは以下のようになります。

Name	Telephone		Email
	Office	Home	

上のテキストはテキストのハイライトと太字がフォントの書式として指定されているグローバルテンプレートである要素 `strong` のダブルクリックにより書式設定されています。

メモ Authentic View 内で表示されるテキストの書式に関しては、必要とされるテキストのフォーマットを持つグローバルテンプレートが要素のためにStyleVision 内で作成される必要があります。

11.2.2.3 CALS/HTML テーブル編集アイコン

CALS/HTML テーブルを編集するために必要なコマンドは、ツールバー内のアイコンとして使用することができ、下にリストされています。これらのアイコンのために対応するメニューコマンドは存在しません。

CALS/HTML テーブルをいつどのように使用するかについての詳細は、CALS/HTML テーブル。

テーブルの挿入

「テーブルの挿入」コマンドは現在のカーソルの場所にCALS/HTML テーブルを挿入します。

テーブルの削除

「テーブルの削除」コマンドは現在アクティブなテーブルを削除します。

行の追加

「行の追加」コマンドは現在アクティブなテーブルの最後に行を追加します。

列の追加

「列の追加」コマンドは現在アクティブなテーブルの最後に列を追加します。

行の挿入

「行の挿入」コマンドは現在アクティブなテーブル内の現在のカーソルの場所の上に行を追加します。

列の追加

「列の追加」コマンドは現在アクティブなテーブル内の現在のカーソルの場所の左に列を挿入します。

セルを左側にジョインする

「セルを左側にジョインする」コマンドは、現在のセル(現在のカーソルの場所)を左側のセルとジョインします。両方のセルのタグは新規のセル内に留まり、列ヘッダーは変更されず、結合されません。

セルを右側にジョインする

「セルを右側にジョインする」コマンドは、現在のセル(現在のカーソルの場所)を右側のセルとジョインします。両方のセルのコンテンツは新規のセルに連結されます。

下のセルとジョインする

「下のセルとジョインする」コマンドは、現在のセル(現在のカーソルの場所)を下のセルとジョインします。両方のセルのコンテンツは新規のセルに連結されます。

上のセルとジョインする

「上のセルとジョインする」コマンドは、現在のセル(現在のカーソルの場所)を上のセルとジョインします。両方のセルのコンテンツは新規のセルに連結されます。

(C) 2015-2021 Altova GmbH

セルを左右に分割する

「セルを左右に分割する」コマンドは、現在アクティブなセルの右に新規のセルを作成します。両方のセルのサイズは元のセルと同じサイズです。

セルを上下に分割する

「セルを上下に分割する」コマンドは、新規のセルを現在アクティブなセルの下に作成します。

上揃え

このコマンドはセルコンテンツをセルの上に揃えます。

上下中央揃え

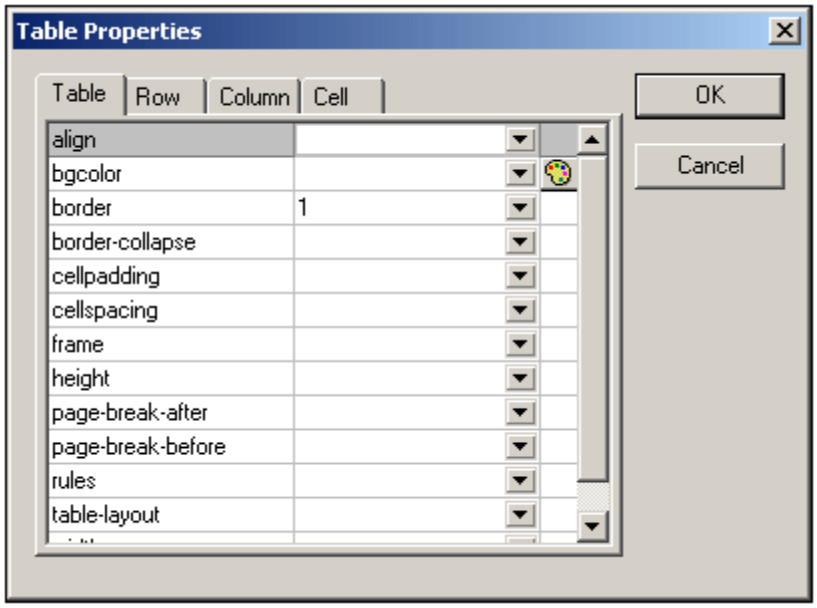
このコマンドは、セルコンテンツを上下中央に揃えます。

下揃え

このコマンドはセルコンテンツをセルの下に揃えます。

テーブルプロパティ

「テーブルプロパティ」コマンドは、テーブルプロパティダイアログボックスを開きます。このアイコンはHTML テーブルのためにのみ有効化することができますが、CALS テーブルのためにクリックすることはできません。



11.2.3 DB の編集

Authentic View 内で、データベース(DB) テーブルを編集し、データを DB に戻すことができます。このセクションには、DB テーブルを編集する際に使用することのできるインターフェイスの説明が含まれています。次の一般的な点について注意してください:

- デザインをコンパクトにまとめるために、Authentic View 内で表示される DB テーブル内のレコードの数量は StyleVision Power Stylesheet のデザイナーにより意図的に制限されている場合があります。このような場合、Authentic View にロードされるレコードの数量を制限することができます。DB テーブル行 ナビゲーションアイコンを使用して DB テーブル内に他のレコードをロードすることができます(*次を参照:* DB テーブルのナビゲート)。
- 特定のレコードを表示するために DB をクエリする ことができます。
- DB レコードを追加、変更、削除し、DB にデータを戻すことができます。次を参照してください: DB テーブルの変更。

メモ Authentic View 内では、SQLite データベースからのデータは編集することができません。Authentic View から SQLite データを保存しようと試みると、メッセージボックスにより既知の制限が通知されます。

11.2.3.1 DB テーブル内をナビゲートする

DB テーブル行をナビゲートするコマンドは、Authentic View ドキュメント内でボタンとして使用することができます。通常、4 – 5 つのボタンを持つナビゲーションパネルが各 DB テーブルに伴います。

矢印アイコンは、左側から右側、内の最初のレコードに移動する、前のレコードに移動する、レコードダイアログに移動する、次のレコードに移動する、最後のレコードに移動する、などがあります(*スクリーンショットを参照してください*)。

DB テーブルをナビゲートするには、必要なボタンを押してください。

XML データベース

IBM DB2 などのXML DB の場合、セル(または、行) には、単一のXML ドキュメントが含まれており、このため、単一の行が Authentic View にロードされます。他の行内のXML ドキュメントをロードする場合は、「Authentic | XML データを持つ新規の行を選択する」メニューコマンドを使用します。

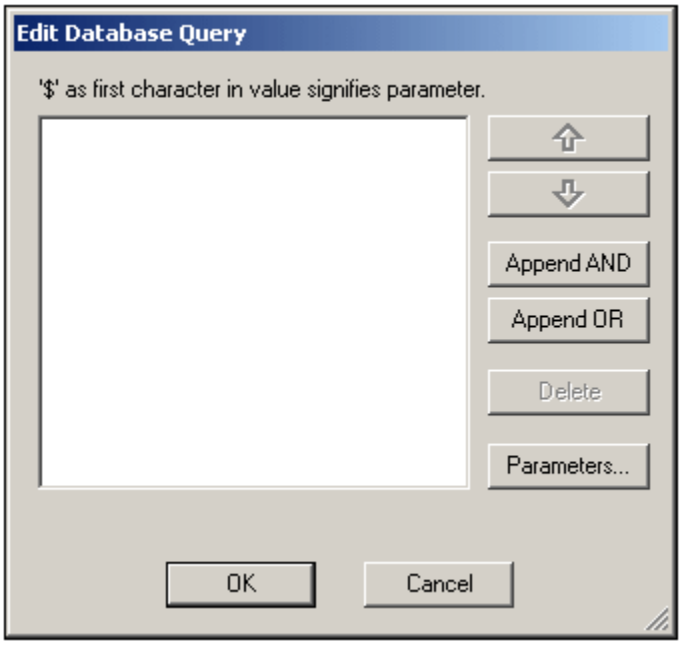
11.2.3.2 DB クエリ

DB クエリにより Authentic View 内に表示されるテーブルのレコードをクエリすることができます。クエリは個別のテーブルのために作成され、1 つのテーブルに 1 つのクエリを作成することができます。クエリの送信時に Authentic View ドキュメント内の変更が保存されない場合、ドキュメント内の全ての変更を保存、または、すべての変更を破棄するかが問われます。他のテーブル内の変更も保存/破棄されます。クエリの送信後、テーブルはクエリの条件を使用して再ロードされます。

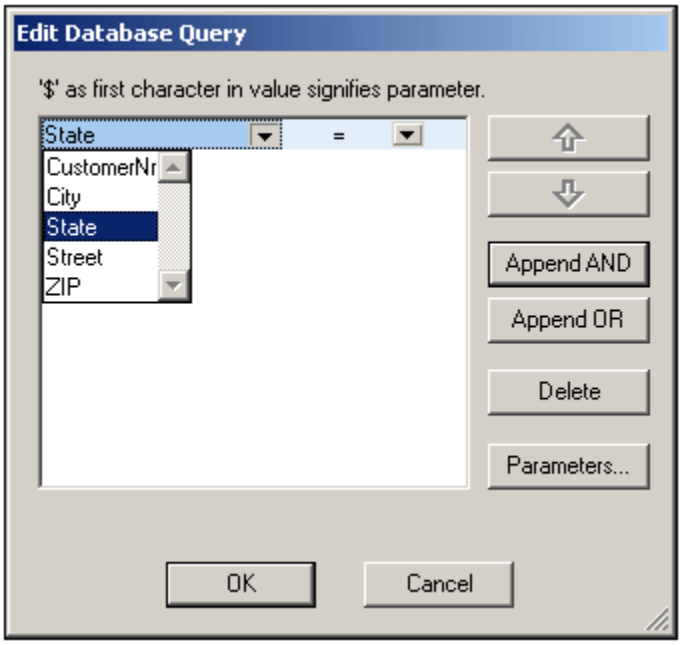
メモ 開かれているテーブルの数量が多すぎることを通知するメッセージが表示されると、開かれているテーブルの数量をクエリを使用してテーブル一部をフィルターし削減することができます。

クエリを作成して送信する方法:

1. データベースクエリの変更 ダイアログを開くために必要とされるテーブルのためのクエリボタン をクリックします(*スクリーンショットを参照してください*)。各 DB テーブルの上、または、下にこのボタンが表示されます。クエリボタンがテーブルに存在しない場合、StyleVision Power Stylesheet のデザイナーはそのテーブルに対して DB クエリ機能を有効化していません。



2. 「AND を追加」、または、「OR を追加」ボタンをクリックします。これにより（下に表示されるように）クエリのための空の条件が追加されます。



3. 条件のために式を入力します。式は以下により構成されています: (i)（関連付けられたコンボボックスで使用することのできる）フィールド名、(ii)（関連付けられたコンボボックスで使用することができます）演算子、(iii)（直接入力することのできる）値。式の構築方法に関しては、必要条件内の式 セクションを参照してください。
4. 他の必要条件を追加する場合、2つの条件をジョインするために使用するロジカルな演算子（AND、または、OR）に従い、「AND を追加」、または、「OR を追加」ボタンをクリックします。新規の必要条件を追加します。ロジカルな演算子の詳細に関しては、次を参照してください: セクション DB クエリ内の必要条件を再度並べ替える。

条件内の式

DB クエリの必要条件内の式はフィールド名、演算子、および値により構成されています。**使用することのできるフィールド名** は、選択されたトップレベルのデータテーブルの子要素です。これらのフィールドの名前は、コンボボックス内にリストされます（*上のスクリーンショットを参照してください*）。「**演算子**」は下にリストされています:

=	等しい記号
<>	等しくない記号
<	より小記号
<=	より小記号、または、等しい記号
>	より大記号
>=	より大記号、または、等しい記号
LIKE	発音が似ている
NOT LIKE	発音が似ていない
IS NULL	空である
NOT NULL	空ではない

IS NULL、または、NOT NULL が選択される場合、値フィールドは無効化されます。**値** は引用符（または、他の区切り文字）無しに入力される必要があります。値は対応するDB フィールドと同じ書式設定を持つ必要があります。それ以外の場合、式は `FALSE` を評価します。例えば、MS Access DB 内の `date` データ型のフィールドのための必要条件が式 `StartDate=25/05/2004` を持つ場

合、式は FALSE を結果として評価します。これは、MS Access DB 内の date データ型は書式 YYYY-MM-DD の書式を持つからです。

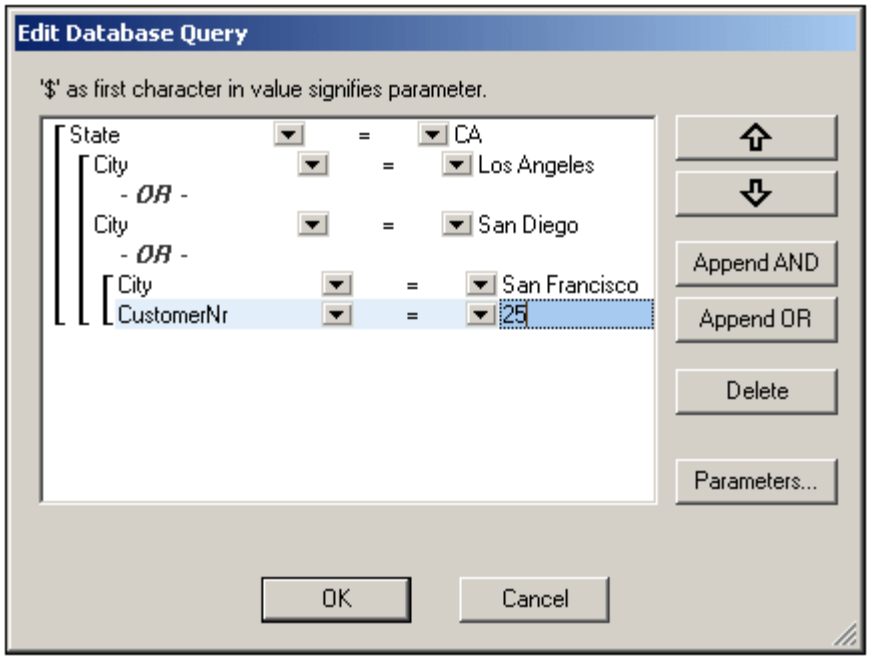
DB クエリを持つパラメーターの使用

クエリを作成する再、式の値としてパラメーターの名前を入力することができます。パラメーターはクエリ内のリテラルの値の代わりに使用することのできる変数です。式内に入力すると、式内で値が使用されます。使用することのできるパラメーターは SPS 内の SPS デザイナーにより定義されており、パラメーターのビューダイアログ内で確認することができます（*下のスクリーンショットを参照してください*）。（出力ドキュメントがコマンドラインでコンパイルされる場合）コマンドラインからパラメーターに値をパスしオーバーライドすることのできる SPS 内のデフォルトの値にパラメーターが割り当てられます。SPS のために定義されたパラメーターをビューするには、データベースクエリの変更ダイアログ内の **パラメーター** ボタンをクリックします。これにより **パラメーターのビュー** ダイアログが開かれます（*スクリーンショットを参照してください*）。

SPS とパラメーター内でスタイルシートのために定義されている **すべての** パラメーターを含むパラメーターのビューダイアログはスタイルシートのデザイン内で編集される必要があります。

DB クエリ内の必要条件を再度並べ替える

DB クエリのロジカルな構造と2つの必要条件、または、必要条件のセット間のルーションシップが視覚的に示されます。ロジカルな構造の各レベルは角かっこで示されています。2つの隣接する必要条件、または、必要条件のセットは AND 演算子を示し、2つの必要条件が OR で区切られている場合、演算子が示されます。必要条件は DB クエリのロジカルな構造のクリアな概要を与えることを目的としています。



上のスクリーンショットで表示されているDB クエリはテキストでは以下の用に表されています:

```
State=CA AND (City=Los Angeles OR City=San Diego OR (City=San Francisco AND
CustomerNr=25))
```

必要条件を移動して、または、必要条件のセットをDB クエリ内の他の条件に相対して上下に移動して、DB クエリの順番を変更することができます。必要条件、または、必要条件のセットを移動するには、以下を行います:

1. クリックして必要条件を選択し、そのレベルを示すかっこをクリックして、レベル全体を選択します。
2. ダイアログ内の「上へ」、または、「下へ」矢印ボタンをクリックします。

次の点について注意してください:

- 進行方向内の隣接する必要条件 が同じレベルの場合、2つの条件は場所を交換することができます。
- 必要条件のセット（すなわち、かっこ内の必要条件）は同じレベル内で位置を変更することができますが、レベルは変更しません。
- 同じレベル内で個別の条件は位置を変更します。隣接する必要条件が更に外向き/内向き（すなわち、同じレベルではない）の場合、選択された必要条件は外向き/内向きに1度に1つのレベル移動します。

DB クエリ内の条件を削除するには、条件を選択して「削除」をクリックします。

DB クエリの変更の方法

DB クエリの変更方法:

1. クエリボタン をクリックします。データベースクエリの変更 ダイアログボックスが開かれます。式の条件内の式を編集し、新規の条件を追加し、条件を並べ替え、DB クエリ内の条件を削除することができます。
2. 「OK」をクリックします。DB からのデータは、DB クエリへの変更が反映されるとAuthentic View に自動的に再ロードされます。

11.2.3.3 DB テーブルの変更

レコードを追加する方法

DB テーブルにレコードを追加する方法:

1. （行を追加するために）DB テーブル行 アイコン内にカーソルを置き、または、（行を入力するために） アイコンをクリックします。これにより一時的な XML ファイル内に新規のレコードが作成されます。
2. 「ファイル | Authentic XML データを保存する...」コマンドをクリックして、DB に新規のレコードを追加します。Authentic View 内では、新規のレコード内の行が DB テーブルに追加されます。このレコードのために AltovaRowStatus が A（追加済み）に設定されます。

新規のレコードのためのデータを入力すると、太字と下線と共に入力されます。これにより、既存のレコードとレコードを区別することができます。既存のレコードがテキストのフォーマット プロパティを使用して書式設定されていない場合、赤字で表示されているデータ型エラーがフラグされます。

「ファイル | Authentic XML データを保存する...」をクリックして新規のレコードは DB に追加されます。新規のレコードが DB 保存されると（ — により示された） AltovaRowStatus フィールドが初期化され、 Authentic View 内で通常のレコードとしてレコードが表示されます。

レコードを変更する方法

レコードを変更するには、必要に応じて DB テーブル内の必要とされる箇所にカーソルを置き、レコードを編集します。表示されるレコードの数量は制限され、必要とされるレコードをナビゲートする必要がある可能性があります（次を参照してください： DB テーブルのナビゲート）。

レコードを変更すると、レコードの全てのフィールド内のエントリに下線が引かれ、このレコードのすべての プライマリインスタンスの AltovaRowStatus は U（更新済み）に設定されます。AltovaRowStatus を持つこのレコードのすべての セカンダリインスタンスは u（小文字）に設定されます。レコードのプライマリとセカンダリインスタンスは DB の構造と生成された XML スキーマに従い定義されます。例えば、Address テーブルが Customer テーブルに含まれる場合、Address テーブルはインスタンス化の 2 つの型内のドキュメントのデザイン内で発生することができます: これは Address テーブル自身が Customer テーブルのインスタンス化内に存在するためです。これらの 2 つの型の片方が変更されると最初に変更される型になります。他の種類としては、1 つ以上の他の型とセカンダリ型が存在する場合があります。データ型エラーは赤い色で表示されフラグされます。

「ファイル | Authentic XML データを保存する...」をクリックして変更は DB に保存されます。変更されたレコードが DB に保存されると、（ — により示された） AltovaRowStatus フィールドが初期化され、Authentic View 内で通常のレコードとしてレコードが表示されます。

以下の点に注意してください：

- Authentic View 内のレコードの単一のフィールドが変更され、データが DB に保存されると、レコード全体が更新されます。
- 日付 値 0001-01-01 が一部の DB ために NULL 値として定義されます。エラーメッセージが発生する場合があります。

レコードを削除する方法

レコードの削除の方法:

1. 削除するレコード内にカーソルを置いて、 アイコンをクリックします。削除されるレコードは取り消し線と共にマークされます。AltovaRowStatus は以下のとおりです: レコードのプライマリインスタンスは D に設定され、セカンダリインスタンスは d に設定され、間接的に削除されるレコードは X に設定されます。間接的に削除されるレコードは個別のテーブル内に保管された削除されたレコード内のフィールドです。例えば、Address テーブルが Customer テーブルに含まれる場合があります。Customer レコードが削除される場合、対応する Address レコードが直接削除されます。Customer テーブル内の Address レコードが削除された場合、Customer テーブル名の Address レコードが主に削除されますが、インスタンス化されている場合、独立した Address テーブル内の同じレコードが 2 番目に削除されます。

2. 「ファイル | Authentic XML データを保存する...」をクリックして変更をDB に保存します。

メモ DB にデータを保存することは、元に戻すコマンドをリセットします。このため、保存前に行われたアクションを元に戻すことはできません。

11.2.4 日付と作業する

Authentic View 内で日付を編集するには2 つの方法があります:

- 日付の選択を使用して日付が入力、または、変更されます。
- 値を入力して日付が入力、または、変更されます。

Authentic View ユーザーが使用するメソッドはSPS 内で定義されています。両方のメソッドはこのセクションの2 つのサブセクションで説明されています。

日付の書式に関するメモ

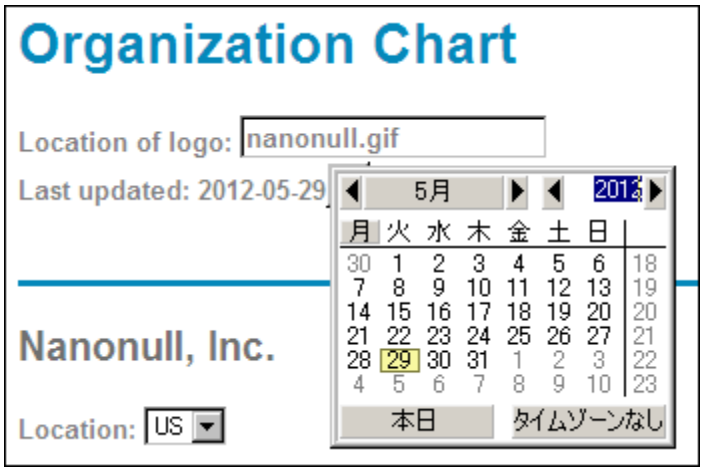
XML ドキュメント内では、日付は複数の日付データ型の1 つに保管することができます。これらのデータ型のそれぞれは、XML ドキュメントが有効であるために日付は特定の文法の書式内で保管される必要があります。例えば、`xs:date` データ型は`YYYY-MM-DD` の書式を必要とします。`xs:date` オード内の日付がこの書式以外の書式で入力されると、XML ドキュメントは無効になります。

正確な書式で日付が入力されるように、SPS デザイナーは視覚的な日付の選択をデザイン内に含むことができます。これにより、日付の選択で選択された日付が、正確な文法の書式であることを確認できます。日付の選択が存在しない場合、Authentic View は正確な文法の書式で日付の挿入を行います。XML ドキュメントの検証により必要とされる文法の書式に関するヒントを得ることができます。

11.2.4.1 日付の選択

XML ドキュメントに標準の書式で日付を挿入するために視覚的なカレンダーである日付の選択が使用されます。ドキュメント内のデータの処理のために標準の書式は重要です。日付の選択 アイコンが変更された日付フィールドの近くに表示されます（*スクリーンショットを参照してください*）。

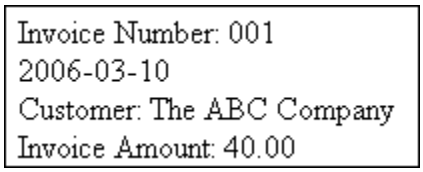
日付の選択を表示するには、日付の選択アイコンをクリックしてください（*スクリーンショットを参照してください*）。



日付を選択するには、希望する日付、付き、または、年度を選択します。XML ドキュメントに日付が挿入され、表示される日付は必要に応じて変更されます。必要な場合は、タイムゾーンも入力してください。

11.2.4.2 テキストのエントリ

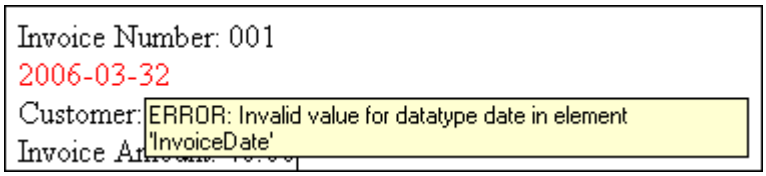
日付の選択が存在しない日付フィールドの場合、新規の値を直接入力することにより日付を編集することができます（*スクリーンショットを参照してください*）。



エラー

以下の種類のエラーがフラグされます:

- 日付を編集し、日付が有効な日付の範囲外になった場合、日付は赤い色で表示されエラーを通知します。マウスのカーソルを無効な日付にポイントすると、エラーメッセージが表示されます（*スクリーンショットを参照してください*）。



- 日付の書式を変更しようとすると、日付は赤い色で表示されエラーを通知します（*スクリーンショットを参照してください。ハイフォンの代わりにスラッシュが使用されています*）。

Invoice Number: 001
2006/03/10
Customer: The ABC Company
Invoice Amount: 40.00

11.2.5 エンティティを定義する

エンティティに関して

ドキュメントがDTD、または、XML スキーマをベースにしているかに関わらず、Authentic View 内で使用するためにエンティティを定義することができます。定義後、これらのエンティティはエンティティの入力ヘルパー内に表示され、コンテキストメニューの「**エンティティの挿入**」サブメニュー内に表示されます。エンティティの入力ヘルパー内のエンティティをダブルクリックすると、カーソル挿入ポイントにエンティティが挿入されます。

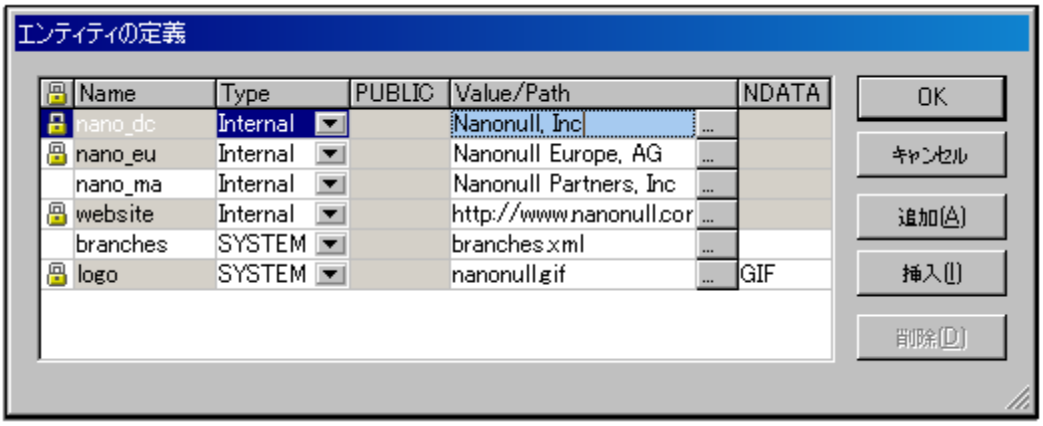
テキスト文字列、XML フラグメント、または、他の外部リソースをドキュメント内の複数の場所で使用する場合、エンティティは役にたちます。必要とされるデータのための短い名前であるエンティティを定義するには、エンティティの定義 ダイアログを使用します。エンティティの定義後、ドキュメント内の複数の場所で使用することができます。これにより時間を節約し、効率よくメンテナンスすることができます。

エンティティの型
ドキュメント内で使用することのできるエンティティには2つの広い型が存在します: XML データ（テキスト文字列、または、XML ドキュメントのフラグメント）である「**解析済みのエンティティ**」、または、非-XML データ（グラフィック、サウンド、または、マルチメディア オブジェクトなどの）バイナリファイルである「**解析されていないエンティティ**」。各 エンティティには名前と値が存在します。解析されたエンティティの場合、エンティティは XML データのためのプレースホルダです。エンティティの値 は XML データ 自身、または、データを含む `.xml` ファイルをポイントする URI です。解析されていないエンティティの場合、エンティティの値は非-XML データ ファイルをポイントする URI です。

エンティティの定義

エンティティを定義する方法:

1. 「**Authentic | XML エンティティの定義....**」をクリックすると、エンティティの定義 ダイアログが開かれます（*下のスクリーンショット*）.



2. 名前フィールドにエンティティ名を入力します。これは、エンティティの入力ヘルパーに表示される名前です。

3. エンティティの型を型フィールドのドロップダウンリストから入力します。次の型を使用することができます: テキストが使用する **Internal** エンティティがXMLドキュメント自身内に保管されています。**PUBLIC** または **SYSTEM** の選択はリソースがXMLファイルの外部にあることを指定し、公開識別子またはシステム識別子を使用してロケートされます。システム識別子はリソースの場所を与えるURI です。公開識別子はプロセッサーがリソースを識別するためのロケーションに影響されない識別子です。公開とシステム識別子の両方を指定すると、公開識別子はシステム識別子、システム識別子を使用して解決します。
4. PUBLIC がType として選択されている場合、PUBLIC フィールド内にリソースの公開識別子を入力します。**Internal** 、または、型としてシステムを選択すると、PUBLIC フィールドは無効化されます。
5. Value/Path フィールド内に、次を入力することができます:

   - エンティティの型がInternal の場合、エンティティの値として使用するテキスト文字列を入力します。エントリに引用符を入力しないでください。入力する引用符はテキスト文字列の一部として扱われます。
   - SYSTEM がエンティティの型である場合、参照ボタンを使用してリソースのURI 、または、ローカルネットワーク上のリソースを入力します。解析されたデータがリソースに含まれている場合、XML ファイルである必要があります（すなわち、ファイルは.xml 拡張子を持つ必要があります）。更に、リソースはGIF ファイルなどのバイナリファイルであることができます。
   - エンティティの型がPUBLIC の場合、このフィールドにシステム識別子を追加で入力する必要があります。

6. NDATA エントリはプロセッサーにこのエンティティを解析せず適切なプロセッサーに送信されるように命令します。NDATA フィールドはエンティティが解析されていないエンティティを示す値を含む必要があります。

ダイアログの機能

エンティティの定義 ダイアログ内で以下の機能を使用することができます:

- エンティティの追加
- エンティティの挿入
- エンティティの削除
- 列ヘッダーをクリックしてエンティティをアルファベット順の値に並べ替えます。1度クリックすると昇順に並べ替え、2度クリックすると降順に並べ替えられます。
- ダイアログボックスと列の幅のサイズを調整します。
- ロック。XMLドキュメント内でエンティティが使用されると、このエンティティはロックされ、エンティティの定義 ダイアログ内で編集することはできません。ロックされたエンティティは、最初の列内でロックシンボルにより表示されます。エンティティをロックすることにより、エンティティに対してXMLドキュメントが有効であることを保証することができます（エンティティが参照されており、定義されていない場合は、ドキュメントは無効になります）。
- エンティティの複製はフラッグされます。

エンティティの制限

- 他のエンティティ内に含まれるエンティティはダイアログ Authentic View、またはXSLT 出力内で解決されません。このようなエンティティのアンパサンド文字は& などのエスケープ文字と共に表示されます。
- イメージファイルではない外部の解析されていないエンティティはAuthentic View 内で解決されません。デザイン内のイメージは外部の解析されていないエンティティを読み取るように定義され、URI がエンティティ名（例: `'logo'`）になるように設定されています。エンティティの定義 ダイアログ内でエンティティ名を（上のスクリーンショット内のロゴエンティティのために設定されているように）イメージファイルのURI に対して解決する値を持つ外部の解析されていないエンティティとして定義することができます（*上のスクリーンショットを参照してください*）。

11.2.6　Authentic View 内のイメージ

Authentic View を使用すると、最終出力ドキュメント（HTML、RTF、PDF およびWord 2007+）内で使用することのできるイメージを指定することができます。イメージの書式の一部は一部の書式、または、アプリケーションでサポートされない場合があることに注意してください。例えば、SVG 書式はPDF 内でサポートされていますが、RTF ではサポートされておらず、HTML 内で表示するためにはブラウザーアドオンが必要になります。イメージ 書式を選択する場合、ドキュメントの出力書式でサポートされている書式を選択してください。

イメージの書式の多数は、多くの出力書式でサポートされています（*下のリストを参照してください*）。

Authentic View はInternet Explorer をベースにしており、Internet Explorer が表示することのできるイメージの書式の大部分を表示することができます。次の一般的なイメージの書式がサポートされています:

- GIF
- JPG
- PNG
- BMP
- W MF（Microsoft ウィンドウ メタファイル）
- EMF（拡張されたメタファイル）
- SVG（PDF 出力のためのみ）

相対的なパス

相対的なパスはSPS ファイルに相対して解決されます。

11.2.7 Authentic View 内のキーストロール

Enter キー

Authentic View 内では、「Enter」キーは、特定のカーソルの場所で追加の要素を追加する際に使用されます。例えば、（スキーマに従い）本のチャプターには複数の段落が含まれており、段落のテキスト内で「Enter」を押すと、新規の段落が現在の段落の後に追加されます。チャプターがタイトルと複数の段落を含むことができる場合（タイトル要素を含む）、段落要素外のチャプター内で「Enter」を押すと、（スキーマにより複数のチャプターが許可されていると想定し）現在のチャプターの後に新規のチャプターが追加されます。

メモ 「Enter」キーは新規のラインを入力しません。これは、段落などのテキスト ノード内にカーソルがある場合でも同じです。

キーボードの使用

入力とナビゲートのためにキーボードを通常の方法で使用することができます:

- 「タブ」キーはカーソルを前に進め、ノードの前後で停止し、ノードコンテンツをハイライトし、静的なコンテンツをステップオーバーし、静的なコンテンツをステップオーバーします。
- `add...` と`add Node` ハイパーリンクは、ノードコンテンツとして考えられ、タブ付けされるとハイライトされます。スペースキーまたは、「Enter」キーを押すと有効化することができます。

(C) 2015-2021 Altova GmbH

12 Authentic スクリプト

Authentic スクリプトにより、SPS デザインが更に柔軟かつ双方向的に機能するようになります。これらのデザインはStyleVision Enterprise ならびにProfessional エディションで作成することができ、Authentic を搭載したEnterprise またはProfessional エディションのAltova 製品にて閲覧することができます。以下のテーブルではAltova 製品ごとにサポートされる機能のリストが示されます。Altova Browser Plugin Trusted バージョンはセキュリティー上の観点から内部スクリプトが無効になっていることに注意してください。

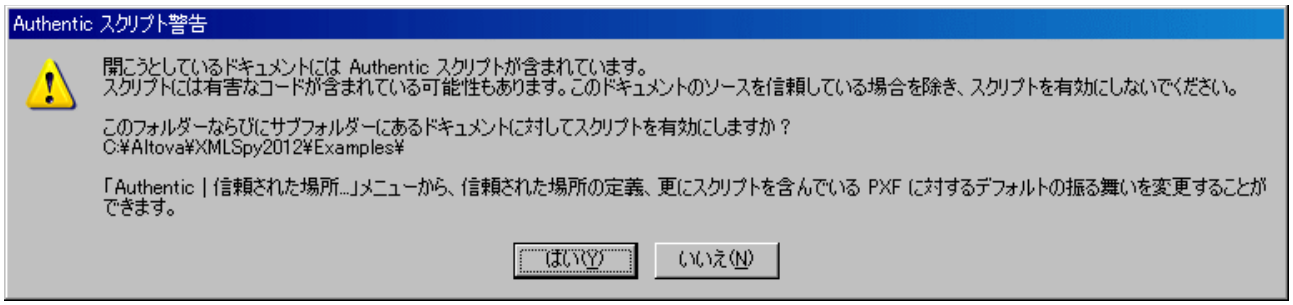
Altova 製品	Authentic スクリプト の作成	Authentic スクリプト
StyleVision Enterprise	はい	はい
StyleVision Professional	はい	はい
StyleVision Basic *	いいえ	いいえ
XMLSpy Enterprise	いいえ	はい
XMLSpy Professional	いいえ	はい
Authentic Desktop Community	いいえ	いいえ
Authentic Desktop Enterprise	いいえ	はい
Authentic Browser Community **	いいえ	いいえ
Authentic Browser Ent Trusted ***	いいえ	はい
Authentic Browser Ent Untrusted	いいえ	はい

* *AuthenticView ビュー無し。*
** *Trusted とUntrusted バージョンの両方。*
*** *スクリプトが存在するのデザインは表示可能。内部マクロの実行やイベントハンドリングは無し。外部イベントのトリガーあります。*

Authentic スクリプトはすべてのAltova 製品で同様の振る舞いをするため、個々の製品に対して特別なコードや設定を指定する必要はありません。

Authentic スクリプト警告ダイアログ

PXF ファイルまたはSPS ベリンクされたXML ファイルにスクリプトが含まれており、(StyleVision 以外のAltova 製品にて) そのファイルが開かれた状態でAuthentic ビューへの切り替えが行われると、警告ダイアログが表示されます(以下のスクリーンショットを参照)。



以下にあるオプションを選択することができます:

- **はい**をクリックすると、ファイルを含んでいるフォルダーが Authentic スクリプトの信頼された場所リストへ追加されます。今後、信頼されたフォルダー内に含まれるファイルを Authentic ビューで開いても、警告ダイアログは表

示されなくなります。信頼された場所のリストへは「**Authentic | 信頼された場所**」メニューコマンドからアクセスならびに編集することができます。
- **いいえ**をクリックすることで、ファイルが含まれているフォルダーを信頼された場所のリストへ追加することなく処理が継続されます。ファイルはスクリプトが無効になった状態で開かれ、ファイルを Authentic ビューで開くたびに Authentic スクリプト警告ウィンドウが表示されます。ファイルが含まれているフォルダーを信頼された場所のリストへ追加するには、「**Authentic | 信頼された場所**」メニューコマンドから信頼された場所のダイアログを開き、フォルダーを追加ならびに必要に応じて変更することができます。

メモ COM インターフェースを介した StyleVision へのアクセスを行う場合、**セキュリティーのチェックが行われず、Authentic スクリプト警告ダイアログは表示されません**（COM インターフェースに関する情報は プログラマーのレファレンス を参照ください）。上記の警告ダイアログは 以外のAuthentic View にて表示されます。SPS デザイナーはこの事に留意

Authentic スクリプトの使用方法

SPS デザインを作成したデザイナーは、以下に示される2つの方法によりAuthentic ドキュメントをインタラクティブなものにすることができます:

- ユーザー定義アクション（マクロ）のためのスクリプトをデザイン要素、ツールバーボタン、とコンテキストメニューアイテムに割り当てます。
- Authentic ビューイベントに反応するデザインイベントハンドラーを追加します。

Authentic ドキュメントをインタラクティブなものにするために必要な全てのスクリプトは、StyleVision GUI（Enterprise ならびに Professional エディション）内で作成することができます。フォーム、マクロ、イベントハンドラーはStyleVision に搭載されたスクリプトエディターにて作成され、これらのスクリプトはSPS 内に保存されます。その後 StyleVision のデザインビューにて、保存されたスクリプトをデザイン要素、ツールバーボタン、コンテキストメニューに割り当てることになります。SPS をベースにしたXML ドキュメントがAuthentic スクリプトをサポートするAltova 製品（上のテーブルを参照）により開かれることで、ドキュメントに与えられた柔軟性や双方向性を使用することができるようになります。

このセクション

このセクションでは、Authentic スクリプトの使用方法について説明します。以下に示されるサブセクションにより構成されます:

- スクリプトエディター: SPS に保管されるスクリプトが含まれる、スクリプトプロジェクトのインターフェースについて記述されます。
- マクロ: マクロとデザイン要素、Authentic ツールバーボタン、そして Authentic コンテキストメニューを関連付ける方法について記述されます。
- イベントハンドラー:イベントハンドラースクリプトの作成と使用について記述されます。
- スクリプトオプション:オプションについて記述されます。

サンプルファイル

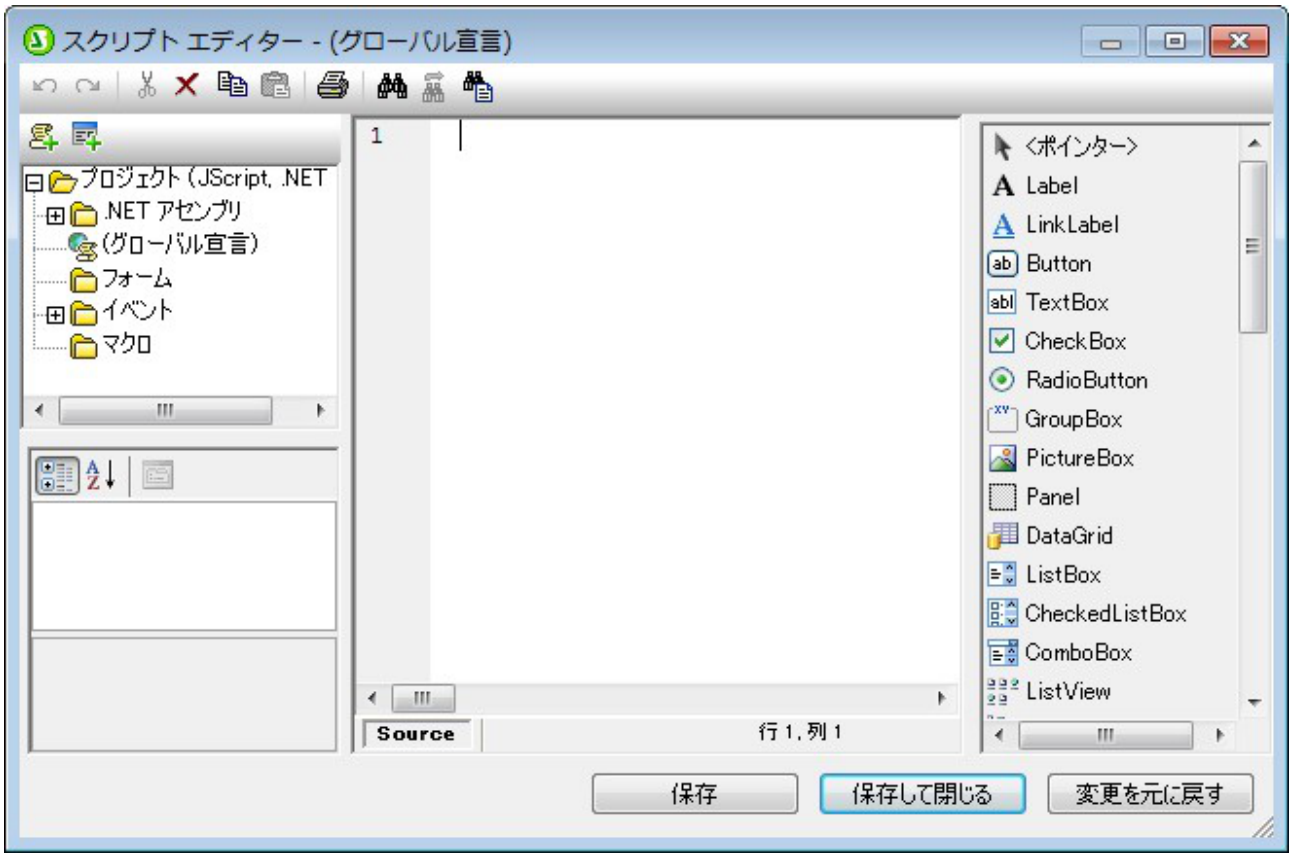
サンプルSPS ファイルは、StyleVision のプロジェクトウィンドウからアクセスすることのできる Examples プロジェクト内の Authentic/Scripting フォルダー以下に収められています。

(C) 2015-2021 Altova GmbH

12.1 スクリプトエディター

StyleVision のスクリプトエディターでは、SPS のフォーム、マクロ、イベントハンドラーに対するスクリプトを追加することができます。

スクリプトエディターはメニューコマンドの「Authentic | Authentic スクリプトの編集」から、またはデザインビュータブのメニュー切り替えスイッチにある Authentic スクリプトアイテムをクリックすることで起動することができます。



以下の方法によりスクリプト言語を変更することができます:

1. プロジェクトツリーの上部にあるプロジェクトアイテムを右クリックします（上のスクリーンショットを参照）。
2. コンテキストメニューから**プロジェクト設定**を選択します。
3. プロジェクト設定ダイアログにて、スクリプト言語として JScript または VBScript を選択し、ターゲットとなる .NET Framework を指定します。選択された Microsoft .NET Framework のクラスや拡張を使用することができるようになり、例えばフォームの作成などを行うことができるようになります。

スクリプトエディターとその機能の詳細については、プログラマーのレファレンスを参照ください。

それぞれの SPS には個別のスクリプトプロジェクトが含まれており、プロジェクトは SPS 内部に保存されます。スクリプトエディターにおける**保存**ボタンをクリックしてもスクリプトプロジェクトが SPS へ保存される訳ではなく、スクリプトプロジェクトに対して行われた変更がメモリ上に保存されるだけとなります。スクリプトプロジェクトに対して行われた変更は、SPS が保存された時にだけ SPS ファイルへ保存されます。

12.2 マクロ

マクロとはユーザーのアクションが発生した際に実行されるプログラムのことです。マクロに引数が与えられることは無く、戻り値が返されることも他のマクロを直接呼び出すこともありません。しかし複数のマクロやイベントハンドラーにより再利用されるコードは、グローバル宣言領域内の関数として宣言することができます。

特定のユーザーアクションに対して割り当てられていないマクロが実行されることはありません。使用することのできるユーザーアクションは、特定のデザイン要素、ツールバーボタンのクリック、ユーザー定義コンテキストメニューアイテムの選択となります。

EventContext プロパティ

SPS デザイン要素は複数の場所に配置することができます。どのようなコンテキストでマクロが実行されるのかを決定するために、AuthenticView インターフェースにはEventContext というコンテキストを保管するためのプロパティが用意されています。EventContext プロパティを介してマクロが開始された場所を問い合わせるXPath への問い合わせを行い、カレントノードでXPath 条件式を評価し、その場所のスコープにて定義された変数へアクセスすることができます。詳細についてはAuthenticView API のセクションを参照ください。

マクロ参照のチェック

意図しないマクロの名前変更や、デザインにて参照が行われているにも関わらずスクリプトプロジェクトがマクロから削除されてしまうことが時として起こりえます。無効な参照のチェックを行う「**Authentic | マクロ参照のチェック**」コマンドを使用することで、このような状況を回避する、またはエラーを回復することができます。参照の置き換えや削除を行うためのダイアログが表示されます。

このセクション

このセクションでは、妥当なユーザーアクションとマクロを関連付ける方法について説明されます:

- デザイン要素上のマクロ
- ツールバーボタンのマクロ
- コンテキストメニューアイテム上のマクロ

12.2.1 デザイン要素上のマクロ

デザイン要素に対するマクロの割り当てはAuthentic プロパティウィンドウから行うことができます。以下のスクリーンショットでは、OnResetAll という名前のマクロが利用可能なマクロの中から選択され、ボタンのOnClick イベントに割り当てられています。

(C) 2015-2021 Altova GmbH

マクロが割り当てられるユーザーアクションは、以下のテーブルにより示される通り、デザイン要素により異なります:

ユーザーアクション	サポートされるデザイン要素	イベントが キャンセル可能か
OnClick	ボタン、イメージ	いいえ
OnBeforeLinkClick	リンク	はい
OnBeforeChange	コンテンツ、その他のコンテンツ、入力フィールド、複数行の入力フィールド、チェックボックス、ラジオボタン、コンボボックス	はい
OnAfterChange	コンテンツ、その他のコンテンツ、入力フィールド、複数行の入力フィールド、チェックボックス、ラジオボタン、コンボボックス	いいえ
OnSetFocus	コンテンツ、その他のコンテンツ、入力フィールド、複数行の入力フィールド、チェックボックス、ラジオボタン、コンボボックス	いいえ
OnKillFocus	コンテンツ、その他のコンテンツ、入力フィールド、複数行の入力フィールド、チェックボックス、ラジオボタン、コンボボックス	いいえ

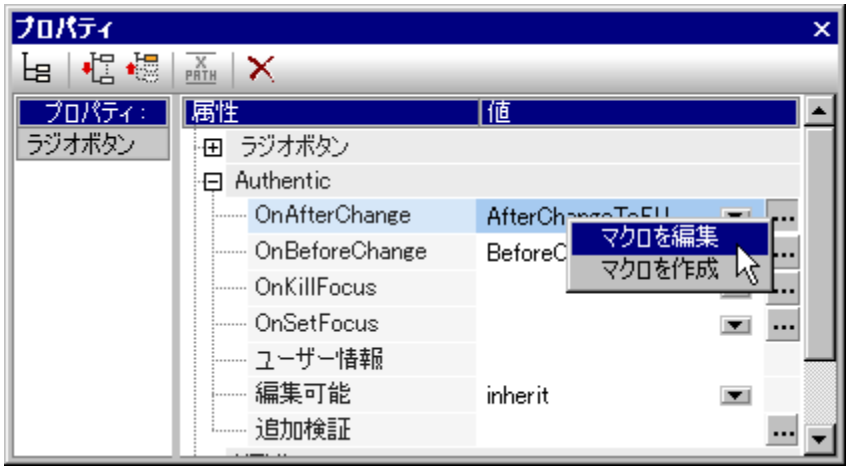
これらのユーザーアクションに割り当てられたマクロはユーザーの入力により呼び出され、元に戻すアクションやAuthentic View の外から行われたアクション(例: COM API 呼び出し)から呼び出されることはありません。マクロがそれ以降のイベント処理をキャンセルするために使用される場合、マクロ内の必要な場所にAuthenticView.DoNotPerformStandardAction() というCOM API 呼び出しを指定してください。例えば、ラジオボタンのOnBeforeChange イベントに対して割り当てられたBeforeChangeToEU というマクロにて、以下の様に使用することができます:

```
if ( !confirm( "Are you sure to change address data to EU?\nAll existing address data
will be deleted!") )
       AuthenticView.DoNotPerformStandardAction();
```

これにより、確認ダイアログでユーザーがキャンセルボタンをクリックした場合、ラジオボタンの値変更を回避することができます。このサンプルはExmples プロジェクト内のAuthentic/Scripting フォルダー内に収められているOnChange.sps にて確認することができます。このファイルではマクロがどのように動作するかを確認することができます。

便利な機能

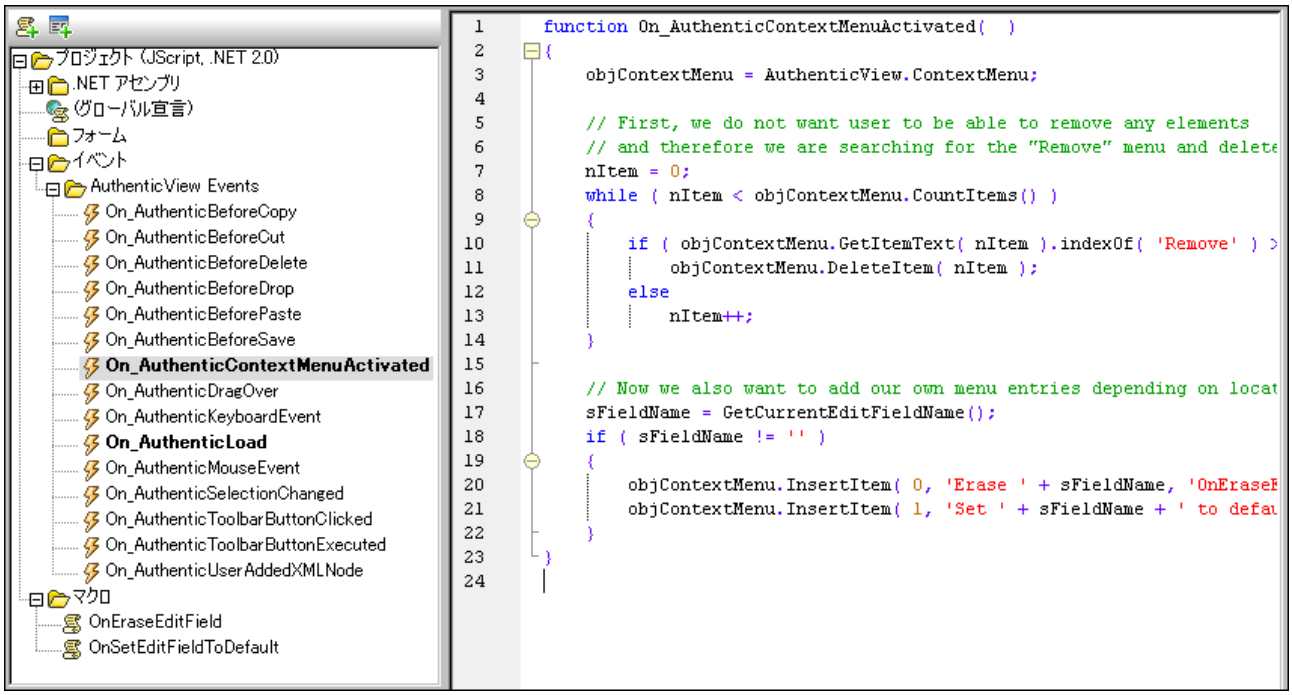
プロパティウィンドウにてマクロが割り当てられた場合、マクロ名の右側にあるマクロの編集ボタンをクリックすることで、スクリプトエディターへ素早くアクセスすることができます(以下のスクリーンショットを参照)。



12.2.2 コンテキストメニューアイテム上のマクロ

Authentic コンテキストメニューに対してもマクロを使用することができます。これによりマクロを実行するカスタムメニューアイテムを追加することができます。

一連の操作の中で GUI が使用されることはありません。On_AuthenticContextMenuActivated イベントに対してイベントハンドラーを追加して、AuthenticView API を介してメニューの操作を行うことになります。このようなサンプルは Examples プロジェクトの Authentic/Scripting フォルダー内にある OnContextMenu.sps にて確認することができます。以下に示されるスクリーンショットでは、On_AuthenticContextMenuActivated イベントハンドラーがスクリプトエディターにて示されます。



On_AuthenticContextMenuActivated イベントに対するイベントハンドラーにより、2つのメニューアイテムが作成されマクロに対して割り当てられます。このサンプルファイルには On-AuthenticLoad イベントに対するイベントハンドラーも含まれており、AuthenticView がロードされた時に入力ヘルパーとマークアップボタンが無効化されます。

```
function On_AuthenticLoad( )
{
    // We are disabling all entry helpers in order to prevent user from manipulating XML tree
    AuthenticView.DisableElementEntryHelper();
    AuthenticView.DisableAttributeEntryHelper();

    // We are also disabling the markup buttons for the same purpose
    AuthenticView.SetToolbarButtonState( 'AuthenticMarkupSmall', authenticToolbarButtonDisabled );
    AuthenticView.SetToolbarButtonState( 'AuthenticMarkupLarge', authenticToolbarButtonDisabled );
    AuthenticView.SetToolbarButtonState( 'AuthenticMarkupMixed', authenticToolbarButtonDisabled );
}
```

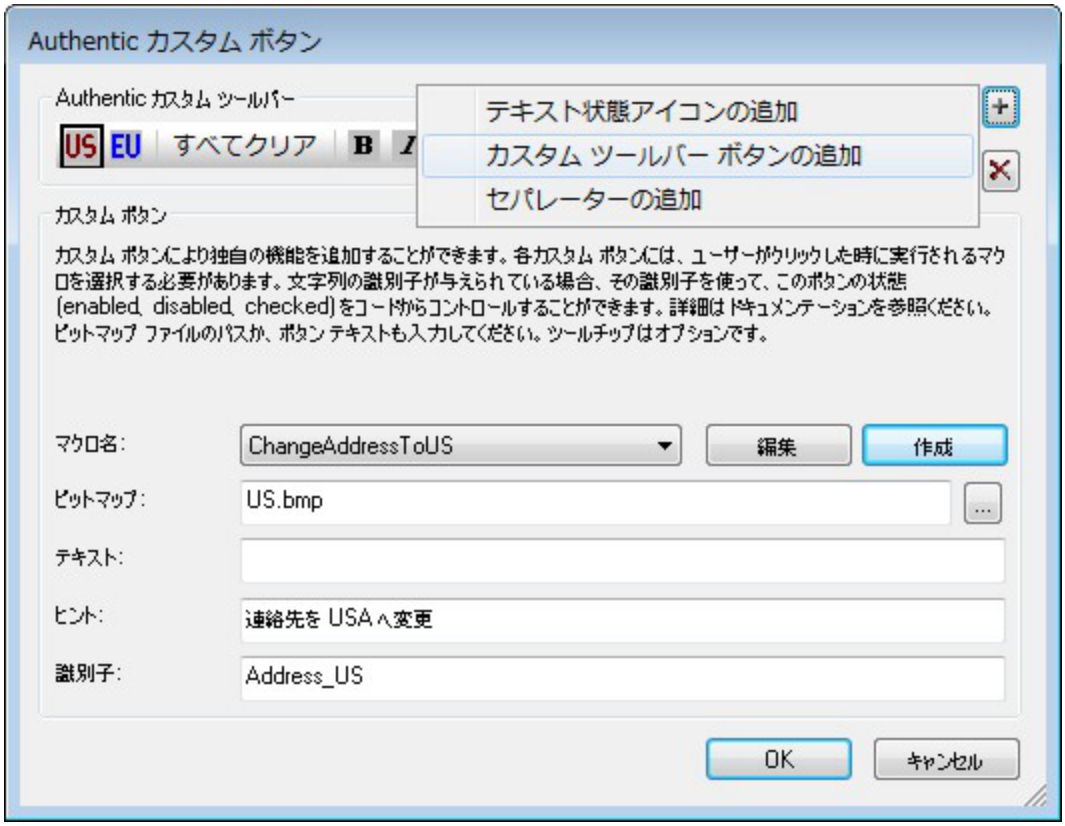

(C) 2015-2021 Altova GmbH

12.2.3　カスタム ボタン

SPS デザイン内では、Authentic ツール バーに追加するためのカスタムボタンを作成し、トリガーするマクロを指定することができます。Authentic ツール バーにカスタムボタンを追加するには、コマンド **Authentic | カスタム ツール バーボタン**を選択し、Authentic カスタム ボタン ダイアログの右上にある、**追加** ボタンをクリックし、カスタムボタンの型を選択してください（*下のスクリーンショット参照*）。

カスタム ボタンの2つの型は以下のとおりです:

- *テキスト状態アイコン* は、テキストのスタイル(太字、または、斜体)を変更し、特定の要素と関連付けられているボタンです。このような要素は、通常複合型コンテンツ型の要素内で発生します（すなわち、テキストと要素の両方を含むことのできる要素）。例えば、テキスト、`bold` 子要素と`italic` 子要素と例えば、を含むことができるため、`para` 要素は、複合型になります。このような場合、（ビットマップイメージである）テキスト状態アイコンを複合型コンテンツ要素の子要素に関連付けることができます ですから、`bold` 要素は、例えば、特定のボタンイメージ（のテキスト状態アイコン)に割り当てられることができます。`bold` 要素のためのグローバルテンプレートは、`bold` 要素に太字の書式を与えるために定義されます。Authentic View 内では、`bold` 要素のためのテキスト状態アイコンは、ツール バーボタンとして表示されます。Authentic View ユーザーが上記の例内の`para` 要素内でテキストを選択し、`bold` 要素のツール バーボタンをクリックすると、選択されたテキストは`bold` 要素タグ内に囲まれ、`bold` 要素の太字書式のグローバルテンプレートがテキストに適用されます。
- *カスタム ツール バーボタン* は特定のマクロに関連付けられます。



カスタム ボタンには以下のパラメーターが与えられます:

- ボタンに使用されるイメージの場所（ビットマップ フィールド）またはボタンのテキスト（テキスト フィールド）
- （テキスト状態アイコンで使用することのできる）*要素名* フィールド内に、テキスト状態アイコンに関連付ける要素名を入力します。

- マクロ名のコンボ ボックスでは、ドロップダウン リストからマクロを選択します。SPS にて保存されたマクロが示されます。[**新規**] ボタンをクリックすると、StyleVision の スクリプトエディターが開かれ、マクロを作成し SPS に保存することができます。[**編集**] ボタンをクリックすると、選択されたマクロが スクリプトエディターで開かれ、編集することができます。
  - Authentic View ユーザーがカスタム ボタンへマウスオーバーした時に表示されるツールヒントをオプションとして、(ツールヒント フィールド内に) 入力することができます。
  - 識別子フィールド内にカスタムボタンの識別子として使用されるテキスト文字列を入力します。デザイナーがボタンの状態を AuthenticView API を使用して管理する場合、この識別子はスクリプトコード内で使用することができます。

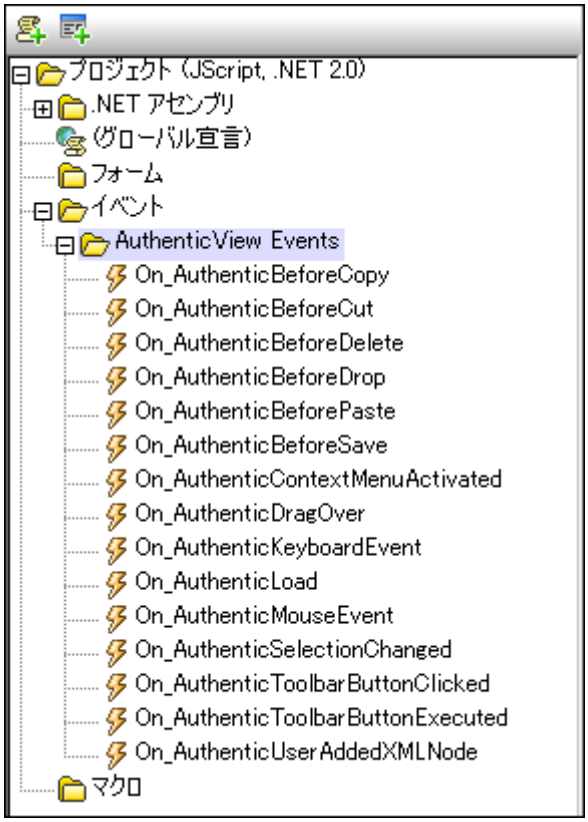
上のスクリーンショットでは、ChangeAddressToUS マクロのカスタム ボタンが示されています。このカスタム ボタンでは US.bmp という名前のイメージが使用されています。ボタンのテキストもフォールバックとして入力することができます。ツールチップにもテキストが入力されており、カスタム ボタンには Address_US という識別子が与えられています。これらの例は Example プロジェクトの Authentic¥Scripting フォルダーにある ToolbarButtons.sps で確認することができます。

(C) 2015-2021 Altova GmbH

12.3 イベントハンドラー

Authentic スクリプトを使用することで、デザイン要素の種類に限定されない一般的なユーザーアクションに反応するイベントハンドラーを定義することが可能になります。外部アプリケーションにある AuthenticView イベントに対して COM リスナーを追加するのと同様のメカニズムが使用されますが、デザイン内部に配置することで、編集がより簡単になります。

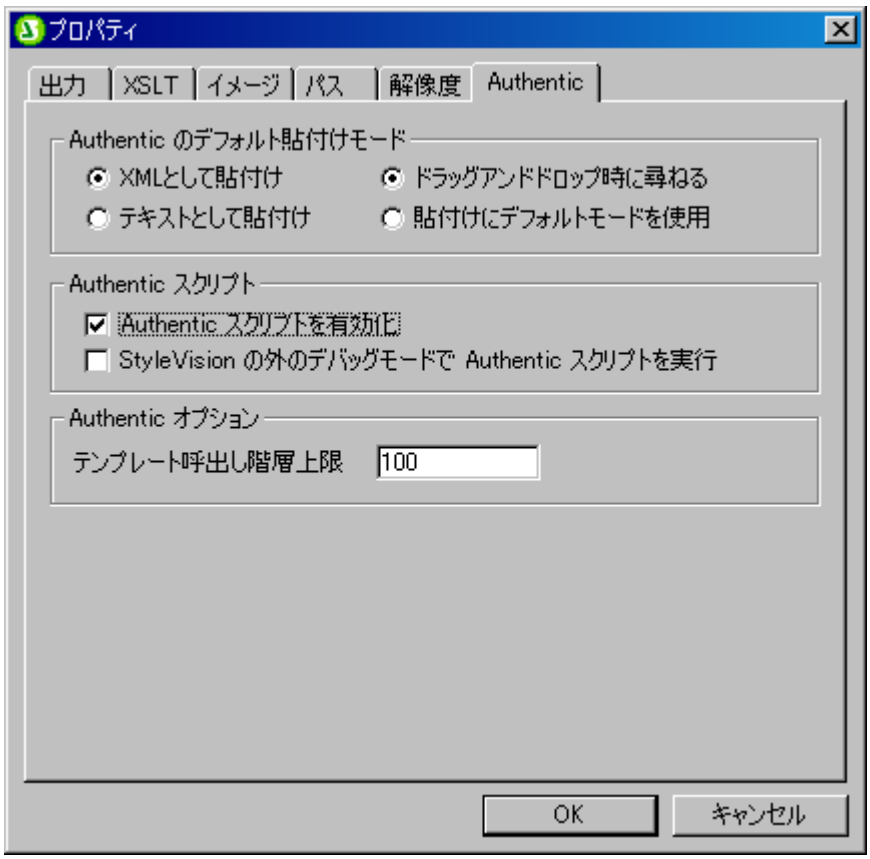
イベントハンドラースクリプトを追加するには、「**Authentic | Authentic スクリプトの編集**」メニューコマンドからスクリプトエディターを起動します。利用可能なイベントハンドラーが、プロジェクトペインの AuthenticView Events ノード以下に表示されます。



イベントをダブルクリックすることで、スクリプトのテンプレートが表示されます。スクリプトペインにてイベントハンドラースクリプトのテンプレートを必要に応じて編集してください。イベントハンドラーの使用例については、コンテキストメニューアイテム上のマクロ を参照してください。

12.4 スクリプトオプション

Authentic スクリプトを使ったインタラクティブなドキュメントを作成するにはSPS ファイルプロパティにてスクリプトを有効化する必要があります。メニューオプションから「ファイル | プロパティ」を選択し、表示されたプロパティダイアログにて Authentic タブを選択して、**Authentic スクリプトの有効化**にチェックを入れてください（以下のスクリーンショットを参照）。



更に、StyleVision の外側（つまりその他の Altova 製品に搭載された Authentic View）からデバッグモードでマクロを実行するか指定することもできます。StyleVision とは違う環境にてマクロの試験やデバッグを行う際にこのオプションを利用することができます。

(C) 2015-2021 Altova GmbH

13 処理の自動化

StyleVision により生成されたXSLT ファイルと出力ファイルを StyleVision の機能と組み合わせることにより、パワフルな自動化を行うことが可能になります。このセクションでは自動化機能について説明します。

StyleVision のファイル生成機能

StyleVision によりSPS デザインが作成されると、使用しているStyleVision のGUI からXSLT ならびに出力ファイルを生成することができるようになります。メニューコマンドの「ファイル | 生成されたファイルの保存」により、以下のファイルを生成することができます:

- HTML とRTF 出力するためのXSLT ファイル。
- HTML とRTF 出力

上のリストから分かるように、StyleVision にて保存することのできるファイルには2種類あります:

1. SPS のデザインにより生成される XSLT ファイル
2. 最終的な出力ファイル（例: HTML）

メモ 更に、ソースにデータベースが使用されている場合、データベースの構造やコンテンツをベースにした XML スキーマならびに XML ファイルを生成することもできます。

最終的な HTML とRTF 出力ファイルを生成するためのプロセスは1つのステップにより実行され、XSLT スタイルシートにより XML ドキュメントが出力フォーマットへ変換されます。

StyleVision Server とRaptorXML: GUI を伴わないファイルの生成

XSLT スタイルシートと目的の出力ファイルを生成するには、StyleVision のGUI（「ファイル | 生成されたファイルの保存」コマンド）の他にも、以下の2つの方法が用意されています:

1. GUI を開かずにStyleVisionのファイル生成 機能を呼び出す StyleVision Server は、多種の出力を作成することができます。
2. RaptorXML は、Altova のXML(+XBRL) バリデーター、とXSLT とXQuery エンジンを含むスタンドアロンのAltova アプリケーションです。RaptorXML 内のXSLT エンジンは、XSLT スタイルシートを使用してXML ドキュメントを処理し書式を出力するためXML の変換に使用することができます。XSLT ファイルを最初に作成すると、RaptorXML により使用されることができます。（RaptorXML は、SPS を入力パラメーターとして受け取らない場合があります）RaptorXML の使用は以下の点で有利です: (i) 大きなファイルの速い変換の結果によりスピードが向上しました。(ii) コマンドライン インターフェイスに加え、RaptorXML は、COM、Java、と.NET のためのインターフェイスを提供し、このため、これらの環境内からの呼び出しが簡単になりました。変換のためのRaptorXML の使用方法は、RaptorXML のサブセクションで説明されています。
3. Altova FlowForce Server ワークフロー内でAltova StyleVision Server を使用して、複数の変換を事前に設定されたトリガーに従い（例えば毎日定刻に）実行することができます。この点については、FlowForce Server を使用した自動化で詳しく説明されています。

13.1 コマンドライン インターフェイス

StyleVision 機能を以下の2つの方法で呼び出すことができます:

- StyleVision 実行可能ファイルを呼び出す。これによりStyleVision のXSLT ファイル生成機能にアクセスすることができます。XSLT ファイルはSPS ファイルから生成されます。
- StyleVision Server を使用して出力ファイル(HTML、など) を生成することができます。関連したファイル(XML、XSD、イメージ、ファイル、などの) 出力ファイルが、パッケージされたSPS ファイルであるPXF ファイルから生成されます。PXF ファイルは、StyleVision から生成されます。

コマンドラインの使用方法

コマンドラインを使用するには2種類の方法があります:

- コマンドラインにコマンドを直接入力し、直ちに実行します。例えばコマンドプロンプトにて StyleVision またはStyleVision Server にコマンドを入力し、**Enter** キーを押下することで、コマンドが実行されます。
- 複数のコマンドをバッチファイルに入力し、以下の例にあるようなバッチ処理を行います:

```
@ECHO OFF
CLS
StyleVision TestEN.sps -outxslt=HTML-EN.xslt
StyleVision TestDE.sps -outxslt=HTML-DE.xslt
StyleVision TestES.sps -outxslt=HTML-ES.xslt
```

バッチファイルが処理されるとコマンドが実行され、ファイルが生成されます。

スケジュール設定により実行される StyleVision

W indows のスケジュールタスクツールを使用することで、指定されたスケジュールに従うかたちでStyleVision コマンド 実行することができます。コマンドを直接入力、または実行するタスクとしてバッチファイルを指定することができます。作成方法に関しては自動処理の方法のセクションを参照ください。

13.1.1 StyleVision

コマンドラインの構文は以下のようになります:

```
StyleVision [<SPS File>] [<options>]
```

where

StyleVision	StyleVision アプリケーションフォルダーに存在するStyleVision を呼び出します。
<SPS File>	SPS ファイルを指定する
<options>	下にリストされるオプションの1つまたは複数のオプション

コマンドが実行されるとStyleVision は(GUI が開かれないなど)警告なしに作動し、必要とされる出力ファイルを生成し閉じられます。エラーまたは警告が発生すると、GUI が開かれ、メッセージボックスに対応するメッセージが表示されます。

メモ StyleVision 内でSPS が正確にロードされるには、SPS が使用するXSD と作業用XML ファイルがSPS 内で指定されている場所に存在する必要があります。

(C) 2015-2021 Altova GmbH

オプション

オプションの入力の順序に決まりはありません。 FO、PDF、と Word 2007+出力に関連したオプションはEnterprise エディションのみで使用することができます。これらのオプションは下の *Enterprise エディションのリスト*で説明されています。

- *XSLT ファイル出力*

`-OutXSLT=<file>`	XSLT-for-HTML を指定されたファイルに書き込む
`-OutXSLRTF=<file>`	XSLT-for-RTF を指定されたファイルに書き込む
`-OutXSLFO=<file>`	XSLT-for-FO を指定されたファイルに書き込む *(Enterprise エディションのみ)*
`-OutXSLWord2007=<file>`	XSLT-for-Word 2007+ を指定されたファイルに書き込む *(Enterprise エディションのみ)*

- *DB データ出力*

`-OutDBXML=<file>`	DB から生成された XML を指定されたファイルに書き込む
`-OutDBchema=<file>`	DB 空生成された XML スキーマを指定されたファイルに書き込む

サンプル

```
StyleVision "QuickStart.sps" -outxslt="QuickStartHTML.xslt"
StyleVision "C:\Test\QuickStart.sps" -outxslt="C:\Test\QuickStartHTML.xslt"
```

以下の点に注意してください:

- パスは絶対的、または、相対的でバックスラッシュを使用します。
- ファイル名、または、パスにスペースが含まれる場合、パス全体は引用符で囲まれる必要があります。例: `"c:\My Files\MyXML.xml" or "c:\MyFiles\My XML.xml"`.
- コマンド、パス、フォルダー、およびファイル名は大文字と小文字を区別します。

13.1.2 StyleVision Server

StyleVision Server を、Windows、Linux とMac OS システム上で、コマンドライン インターフェイス(CLI) 上で使用することができ、XML ファイルを出力 HTML、PDF、RTF、とDOCX ドキュメントに変換します。StyleVision Server CLI の`generate` コマンド は、XML ファイルとPXF ファイルを2 つの引数として取り、希望する出力の書式をパラメーターとして取ります。変換のためのXSLT スタイルシートは、入力として提出されているPXF ファイルから取得されます。

RaptorXML Server のCLI ではなく、StyleVision ServerのCLI を使用する利点は、StyleVision Server は、PXF ファイルを入力として取ることができる点です(RaptorXML はXSLT ファイルを入力としてとります)。StyleVision Server はAltova FlowForce ワークフローの一部として使用されることが奨励されます。FlowForce ワークフローは、変換ジョブを設定済みのトリガーに従い

開始します: FlowForce ジョブがトリガーされるとネットワーク内で、複数のファイルは自動的に変換されます。詳細に関しては、FlowForce Server を使用した自動化セクションを参照してください。

StyleVision Server CLI に関する詳しい説明は、次を参照してください: StyleVision Server ドキュメント。

出力ファイル

StyleVision Server は、1 つまたは複数のファイルを指定された PXF ファイルから生成することができます:

- PXF 内で指定された XML と XSLT-for-HTML ファイル、または、代替の XML ファイルを使用した HTML (`.html`) ファイル
- PXF 内で指定された XML と XSLT-for-RTF ファイル、または、代替の XML ファイルを使用した RTF ファイル

(C) 2015-2021 Altova GmbH

13.2 RaptorXML の使用

Altova RaptorXML は、最新の標準と並列計算の標準のための最適化のために構築された Altova の第三世代の超高速な XML と XBRL プロセッサーです。XBRL processing is available only in RaptorXML+XBRL Server. 高度なクロスプラットフォームに対応し、エンジンは今日のマルチコアコンピューターに XML と XBRL データを処理するために高速な処理機能を提供します。

RaptorXML には以下の2 つのエディションがあります:

- ネットワーク上でアクセスすることができ、複数のファイルを1 度に変換することのできる RaptorXML Server エディション.
- ネットワーク上でアクセスすることができ、複数のファイルを1 度に変換することができ、XBRL 検証をサポートする RaptorXML+XBRL Server エディション。

RaptorXML に関する詳しい説明は、次を参照してください: Altova W eb サイト。

一般的な使用例

StyleVision ユーザーにとり最も一般的な RaptorXML の使用例は、XSLT 変換機能です。この機能は以下を行うために通常使用されます:

1. 「ファイル | 生成されたファイルの保存」コマンドを用いて、XSLT スタイルシートは SPS から生成されます。SPS ファイルから XSLT スタイルシートを生成するために RaptorXML を使用することはできないことに注意してください。
2. RaptorXML を使用して、XML ドキュメントを変換するために生成された XSLT スタイルシートが使用されます。RaptorXML を使用して HTML と RTF 出力を生成することができます。

RaptorXML の利点

RaptorXML を使用する利点を以下に示します:

- RaptorXML には、高速な検証と XSLT 変換が搭載されており、大きなファイルを処理するために役に立ちます。
- コマンドライン、COM、Java、.NET インターフェースによる簡単な使用。
- バッチファイルや Windows のタスクスケジューラーにより、処理の自動化やスケジュール処理などを行うことができます。

XML ソースドキュメントから F 出力ドキュメント (HTML など) の生成を自動化するために RaptorXML を使用する方法に関しては、次のセクションを参照してください: 自動処理の方法。

RaptorXML の COM、Java、と .NET インターフェイスの使用を含む RaptorXML に関する詳細は、次を参照してください: RaptorXML ユーザードキュメント。

13.2.1 PDF 出力

XML ドキュメントから PDF 出力を生成するには2 つのステップを踏む必要があります::

1. (RaptorXML などの) XSLT 変換エンジンにより XML ドキュメントの変換を行います。Altova XSLT エンジンといった XSLT 変換エンジンを使用することで変換を行うことができ、FO ドキュメントを生成することができます。
2. (Apache FOP などの) FOP といった FO プロセッサーにより FO ドキュメントを処理することで PDF 出力が生成されます。XSLT 変換から得られた FO を FO プロセッサーへ渡すように StyleVision をセットアップすることで、PDF 生成の結果を StyleVision の PDF プレビューウィンドウに表示、または(「ファイル | 生成されたファイルの保存」コマンドにより)ファイルとして保存することができます。

RaptorXML とPDF

RaptorXML は XSLT 変換に使用されるだけで、FO プロセッサーに対して FO 出力を渡すようなパラメーターは存在しません。そのため、XSLT 変換の（最初の）ステップにより FO ドキュメントが生成されるだけになります。

FO ドキュメントを FO プロセッサーに渡すことで FO から PDF への処理が行われます。このステップを実行するために使用される命令は使用するプロセッサーにより異なります。例えば、Apache FOP プロセッサーの場合、以下のコマンドにより入力 FO ドキュメントと出力 PDF ドキュメントの名前と場所が指定されます:

```
fop -fo input.fo -PDF output.pdf
```

FOP には他のパラメーターも用意されています。詳細については FOP ユーザーリファレンス を参照ください。

FOP とXSLT

FOP オプションにより、入力 XML ファイル、入力 XSLT ファイル、そして出力 PDF ファイルを同時に指定することができます:

```
fop -xml input.xml -xslt input.xslt -PDF output.pdf
```

この場合、FOP は、内蔵のXSLT エンジンを使用し、XML からFO への変換が最初に行われます。その後生成されたFO ドキュメントをFOP へ渡すことで、FO からPDF への変換処理が行われます。

しかしながら、FOP 内蔵のエンジンは、StyleVision とRaptorXML によりサポートされるすべてのXSLT 機能をサポートしない可能性があります。この結果、StyleVision により生成されたXSLT スタイルシート がXML 変換によりFOP の内蔵のXSLT を使用して指定されている場合、エラーが発生する可能性があります。RaptorXML+XBRL のXSLT エンジンを使用して、FO への変換を行い、FO ファイルをFOP へ渡してPDF の生成を行うようにしてください。

PDF のバッチ処理

XSLT 変換の最初のステップのためにRaptorXML を使用して、PDF を生成する素早い簡単な方法です。 2番目のステップである FOP を使用した、FO 処理は、2つのコマンドを組み合わせるバッチファイルを作成します。例:

```
raptorxmlserver xslt --input=Test.xml --output=Test.fo Test.xslt
fop -fo Test.fo -pdf Test.pdf
```

最初のコマンドにより RaptorXML が呼び出され、`test.fo` が出力として生成されます。2番目のコマンドでは、`test.fo` が FOP プロセッサーへ渡され、PDF ファイルの `test.pdf` が作成されます。バッチ処理と、バッチファイルを使った処理の自動化については、次を参照してください: 処理の自動化。

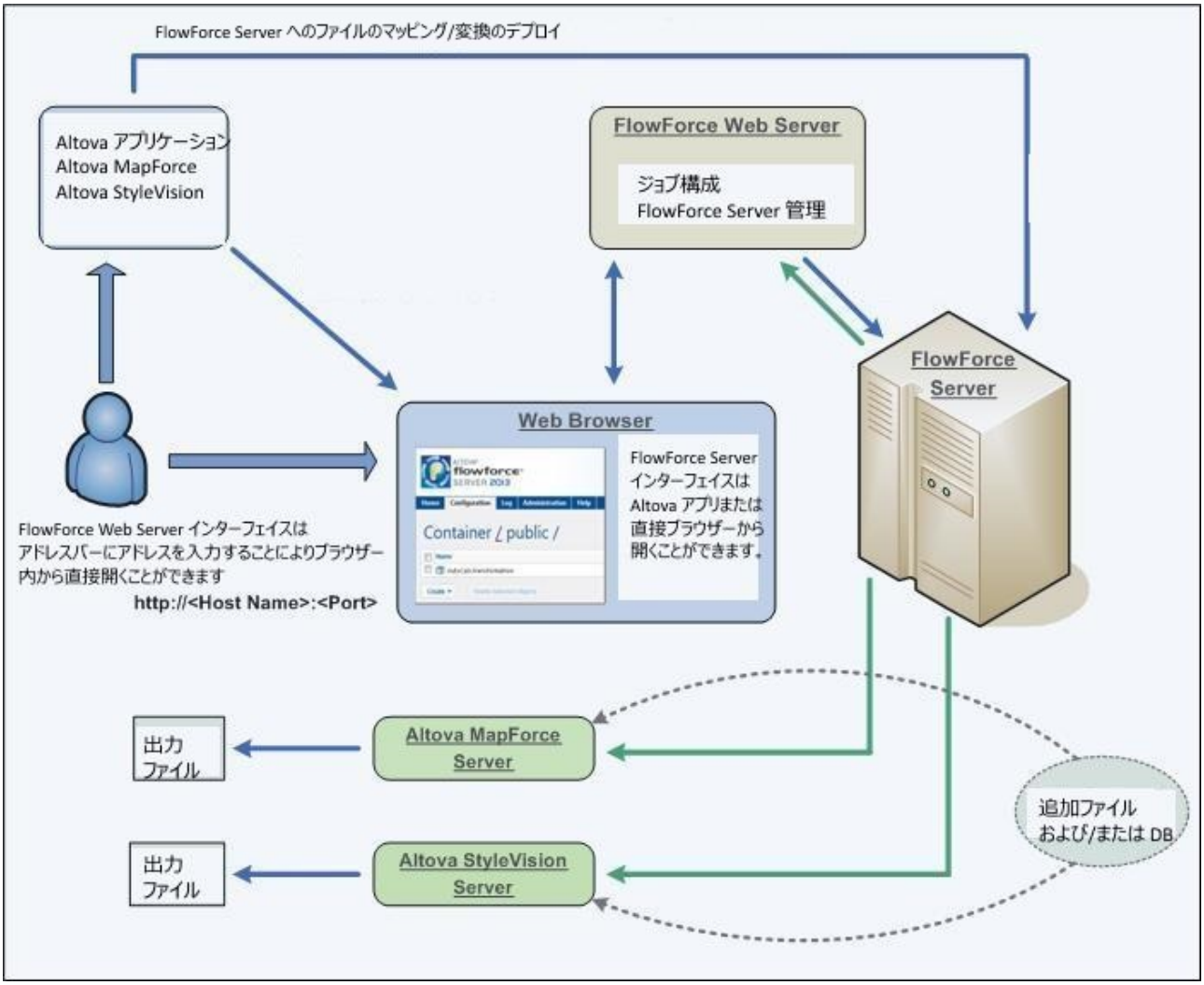
(C) 2015-2021 Altova GmbH

13.3 FlowForce Server を使用した自動化

Windows、Linux、とMac OS システム上で使用することのできるAltova のFlowForce Server を使用して、ネットワーク上で変換を自動化することができます。このプロセスは以下のように作動します:

1. (ファイル | FlowForce へデプロイする コマンドを使用して) StyleVision から、PXF ファイルは .transformation ファイルとして FlowForce Server にデプロイ されます。 .transformation ファイルは、SPS (内でデザインされたとおりに変換を実行するために必要な全てのファイルと情報を含んでいます。下の図では、デプロイは接続ラインにより表示されています。)
2. .transformation ファイルがFlowForce Server にデプロイされた後、.transformation ファイルを使用して、ジョブ定義内で指定されているトリガーに従い、変換を生成するジョブを FlowForce 内でジョブを作成することができます。(トリガーは、例えば、毎日定時にトリガーするように設定することができます。) (下のダイアグラムで表示されているように) StyleVision 、または、HTTP アドレスによりアクセスすることのできる Flow Force ジョブを FlowForce Web Server インターフェイスで作成することができます。FlowForce ジョブの作成に関する詳細は、次を参照してください: FlowForce ドキュメント.
3. 実行時にFlowForce Server は、変換命令と関連するファイルを変換を行う StyleVision Server にパスします (*下の図を参照してください*)。

FlowForce ワークフロー内のStyleVision Server のや区割りは、下の図に表示されています (Altova MapForce マッピングを実行のためにAltova MapForce Server に転送する FlowForce ジョブも作成することができるため、ワークフロー内でのMapForce Server の役割も表示されています)。



FlowForce ジョブにより呼び出されない場合、StyleVision をコマンドラインから呼び出すことができます。使用方法は、StyleVision Server ドキュメントで説明されています。

13.4 自動処理の方法

バッチファイル（ファイル拡張子 .bat と共に保存されるファイル）コマンドラインから実行されるコマンドのシーケンスを含んでいます。バッチファイルが実行されると、バッチファイル内の各コマンドは、最初から始まり順番に処理されます。バッチファイルは次のシチュエーションで役に立ちます:

- 一連のコマンドを自動的に実行する場合（*下を参照*）。
- コマンドが前のコマンドにより生成される入力を必要とする一連の処理コマンドを作成する場合。（例えば、変換の出力として生成される XML ファイルが次の変換の入力として使用される場合）。
- 特定の時間にタスクのシーケンスが実行されることを予定する場合。

コマンドのシーケンスを持つバッチファイル

実行されるコマンドのシーケンスは以下のように入力されます:

```
@ECHO OFF
CLS
StyleVision TestEN.sps -outxslt=HTML-EN.xslt
StyleVision TestDE.sps -outxslt=HTML-DE.xslt
StyleVision TestES.sps -outxslt=HTML-ES.xslt
```

バッチファイルが処理されると、コマンドは実行され、ファイルが生成されます。上記のバッチファイルは、StyleVision を使用して、3 つの XSLT ファイルを SPS ファイルから生成します。

14 Visual Studio で動作する StyleVision

StyleVision をMicrosoft Visual Studio 2010/2012/2013/2015/2017/2019 に統合することができます。この機能により両方の環境の利点を活かすことができ、Visual Studio という開発環境にて高度な SPS ファイルを作成することが可能になります。

このセクションでは、以下にある項目について説明されます。:

- 大まかなインストール処理とVisual Studio へのStyleVision プラグイン統合。
- Visual Studio 版とスタンドアロン版の違い。

関連項目

- StyleVision 統合

(C) 2015-2021 Altova GmbH

14.1 Visual Studio プラグインのインストール

Visual Studio で使える StyleVision プラグインをインストールするには、以下の手順を踏みます:

1. Microsoft Visual Studio 2010/2012/2013/2015/2017/2019 のインストール。
2. StyleVision (Enterprise または Professional Edition) のインストール
3. StyleVision の Microsoft Visual Studio Integration Package をダウンロードして実行。パッケージは www.altova.com にある StyleVision (Enterprise または Professional Edition) ダウンロードページにてご利用になれます。(注意: StyleVision バージョンに対応する Integration Package を使用する必要があります (現在のバージョンは 2021 です)。)

Integration Package をインストールすると、Visual Studio 環境から StyleVision を使用することができるようになります。

関連項目

- StyleVision 統合

14.2 スタンドアロン版 StyleVision との違い

このセクションではVisual Studio のStyleVision プラグインとスタンドアロン版 StyleVision の違いについて説明します。

入力ヘルパー(Visual Studio のツールウィンドウ)

StyleVision の入力ヘルパーはVisual Studio のツールウィンドウからアクセスすることができます。以下の点について留意する必要があります。入力ヘルパーとStyleVision GUI については、ユーザーインターフェイスのセクションを参照ください。

- 入力ヘルパーウィンドウを開発環境内にある任意の場所へドラッグすることができます。
- 入力ヘルパータブを右クリックすることで、インターフェースのカスタマイズを行うことができます。ウィンドウへのドッキング、フロート、自動的に隠すというオプションを選択することができます。

Visual Studio コマンドとして使用される StyleVision コマンド

StyleVision コマンドの一部はVisual Studio GUI にてVisual Studio コマンドとして表示されます::

- *元に戻す*、*やり直し* : これらの Visual Studio コマンドは Visual Studio 環境における全てのアクションに対して適用されます。
- *プロジェクト*: StyleVision のプロジェクトは Visual Studio のプロジェクトとして管理されます。
- *カスタマイズ ツールバー*, *カスタマイズ コマンド*: カスタマイズダイアログ(「**ツール | カスタマイズ**」)に含まれているツールバーとコマンドタブには Visual Studio のコマンドと StyleVision のコマンドの両方が収められています。
- *表示* : 表示メニューにある **StyleVision** オプションでは入力ヘルパーウィンドウやサイドバーの表示切り替え、編集ビューの切り替え、編集ガイドの切り替えを行うことができます。
- ***StyleVision ヘルプ*** : Visual Studio ヘルプメニューのサブメニューとして StyleVision のヘルプメニューが表示されます。

関連項目

- StyleVision 統合

(C) 2015-2021 Altova GmbH

15 Eclipse で動作する StyleVision

Eclipse はプラグインというかたちで様々な種類のアプリケーションを統合することができるオープンソースのフレームワークです。

Eclipse のStyleVision プラグインにより、Eclipse 2020-12, 2020-09, 2020-06, 2020-03 プラットフォームからStyleVision の機能へアクセスすることが可能になります。Eclipse はW indows プラットフォームで利用することができます。このセクションでは、Eclipse に対応したStyleVision のプラグインをインストールする方法について、そしてStyleVision パースペクティブ をセットアップする方法について記述されます。セットアップが完了すると、StyleVision GUI のコンポーネントとStyleVision メニューコマンドがEclipse GUI にて使用できるようになります。

関連項目

- StyleVision 統合

15.1 Eclipse の StyleVision プラグインをインストールする

Eclipse 用のStyleVision プラグインをインストールする前に、以下が既にインストールされていることを確認してください:

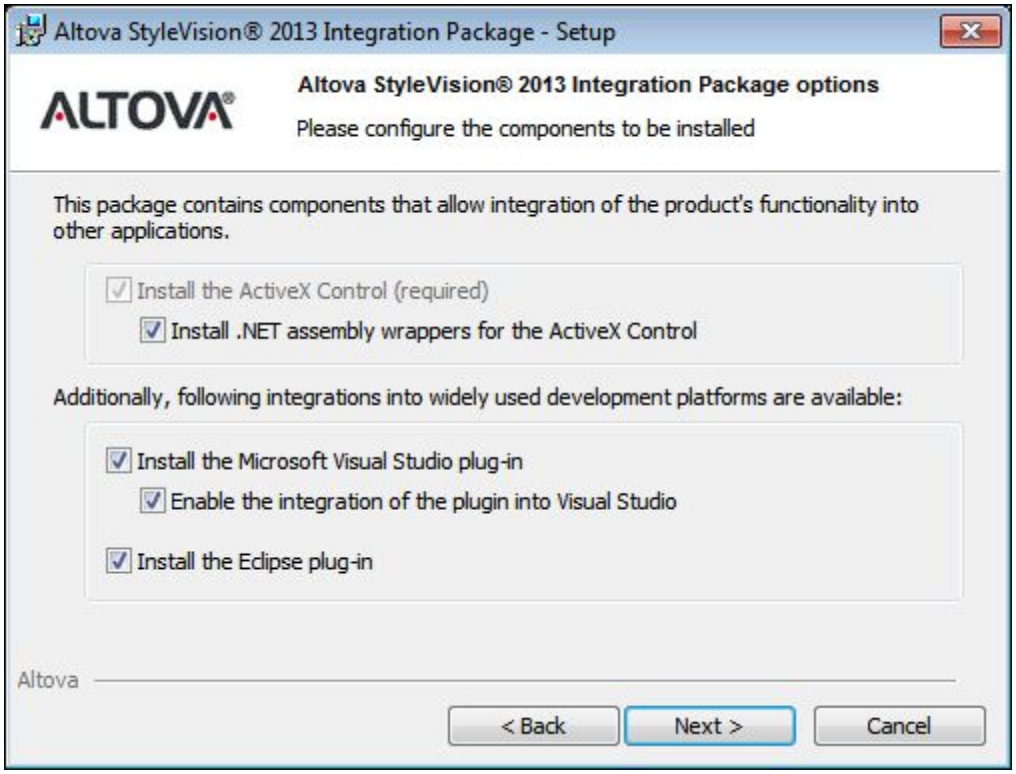
- StyleVision Enterprise またはProfessional Edition。
- Eclipse プラットフォーム2020-12, 2020-09, 2020-06, 2020-03 。StyleVision のバージョン(32-ビットトまたは64-ビット)と等しいバージョンのEclipse(32ビットまたは64ビット)をインストールしてください。
- Eclipse のバージョンとStyleVision (32-ビット または64-ビット) に一致するJava JRE/JDK 。例えば、Eclipse 64-ビットとStyleVision 64-ビットを使用する場合、64-ビット JRE/JDK をインストールしてください。Eclipse により必要とされる特定のJava バージョンはEclipse インストールの前提条件をチェックしてください。

これらがインストールされた状態で、StyleVision Integration Package に収められているEclipse 用のStyleVision プラグインをインストールすることができます(下記を参照)。

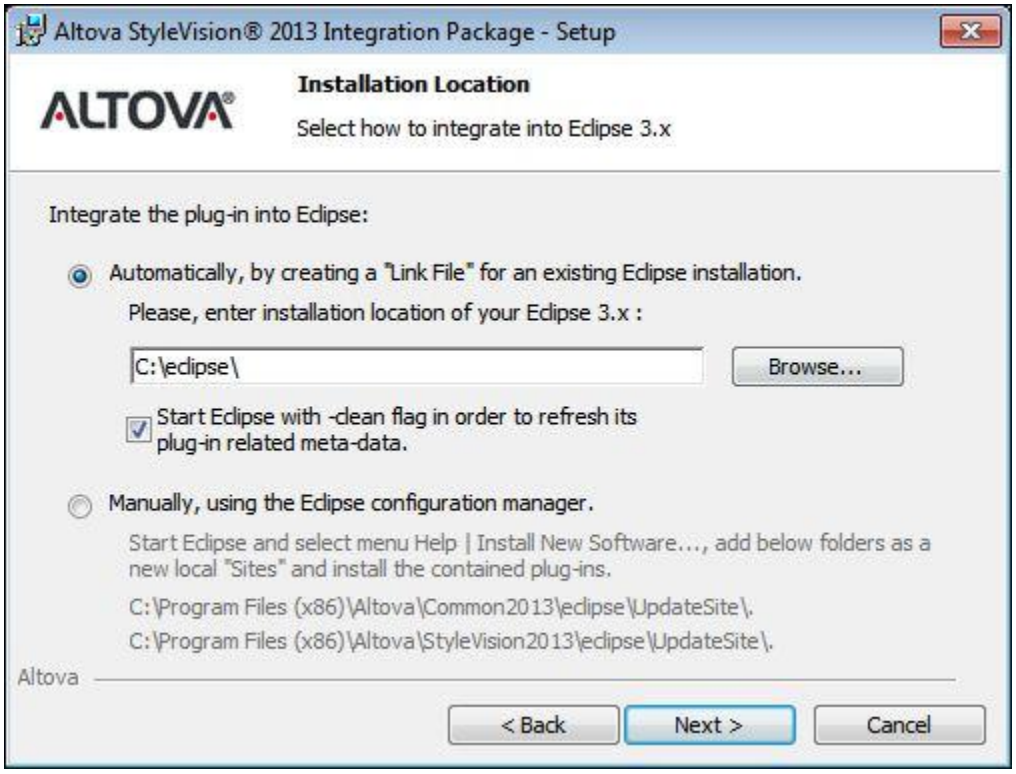
StyleVision Integration Package

Eclipse 用のStyleVision プラグインはStyleVision Integration Package の中に含まれており、以下のようにStyleVision Integration Package のインストーラーからインストールすることができます:

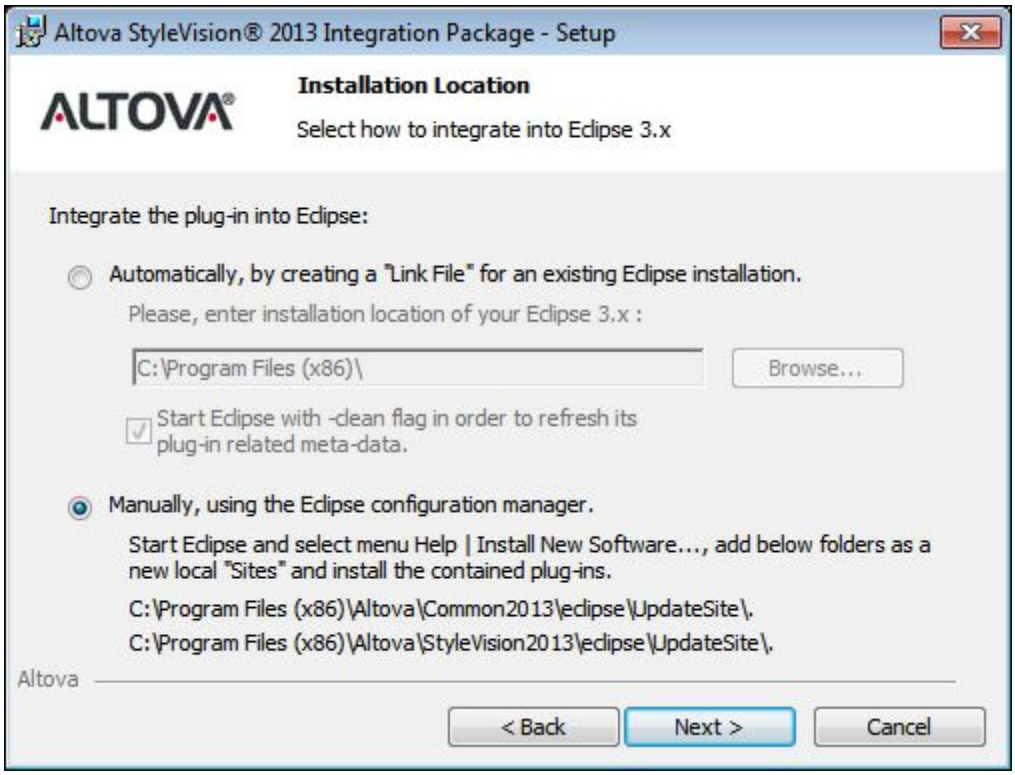
1. StyleVision (Enterprise またはProfessional Edition)、JRE、そしてEclipse が既にインストールされていることを確認してください(上記参照)。
2. Altova ウェブサイトのコンポーネントダウンロードページから、StyleVision Integration Package をダウンロード、インストールします。このインストールでは重要なステップが2つあり、それぞれステップ 3 と4 で説明されます。
3. StyleVision Integration Package のインストールでは、Eclipse 用のStyleVision プラグインをインストールするかどうかを尋ねられます(以下のスクリーンショットを参照)。オプションにチェックを入れて、次へをクリックしてください。



4. その次に表示される(以下のスクリーンショットの)ダイアログにて、インストールウィザードによりEclipse へStyleVision プラグインの統合を行う(自動)か、ご自身の手で(Eclipse GUI から)Eclipse 用のStyleVision プラグインを統合することができます(手動)。



ここではインストールウィザードによる統合(自動)を選択することを推奨します。*「自動」*オプションをチェックし Eclipse 実行可能ファイル(`eclipse.exe`)が存在する場所を参照します 。それらの操作を完了後、次へボタンを押下します。手動で StyleVision プラグインをインストールする場合は、(*以下のスクリーンショットにあるように*) *手動*オプションを選択してください。Eclipse 内部から手動で統合する方法の命令のための下のセクションを参照してください。

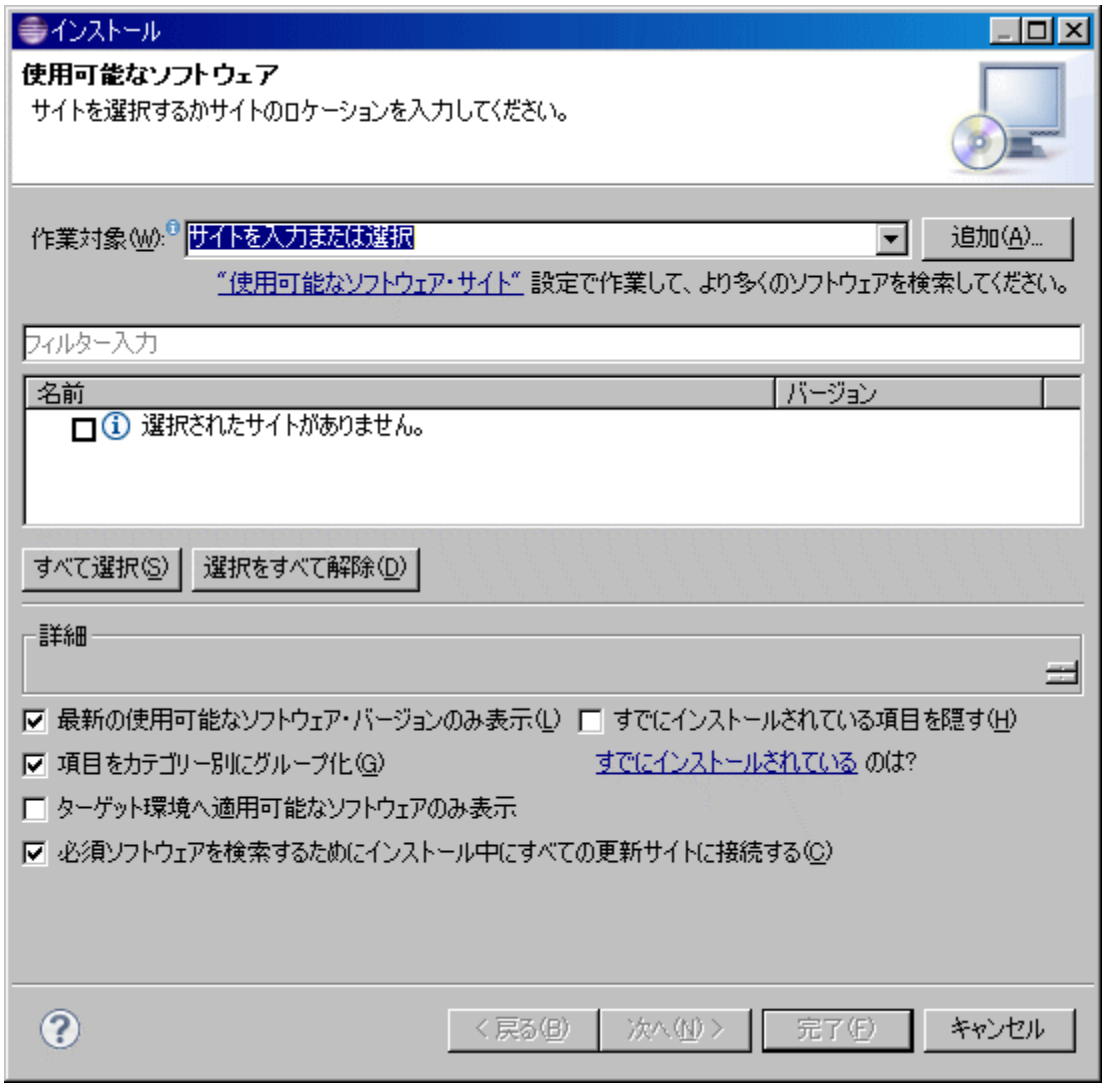


5. インストールを完了します。自動的な統合をセットアップしている場合 Eclipse のためのStyleVision プラグインはに統合され、Eclipse を次回起動する際に使用できるようになります。

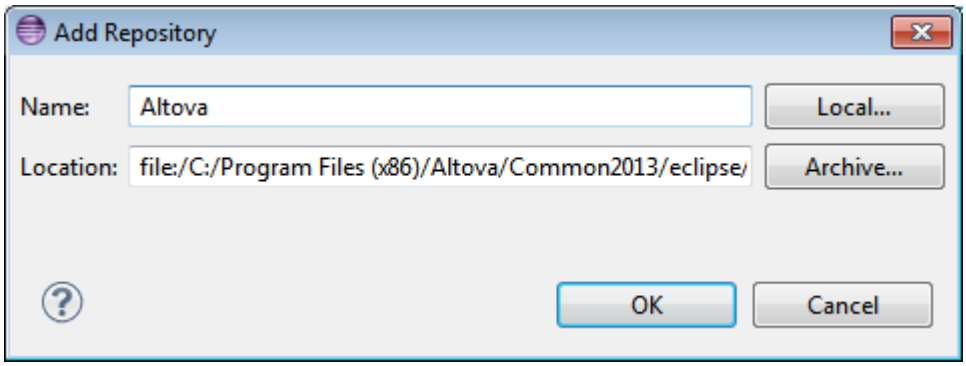
Eclipse 内部で StyleVision プラグインを手動で統合する

Eclipse のためのStyleVision プラグインを手動で統合する場合、for do the following:

1. Eclipse にてメニューオプションから「**ヘルプ | 新規ソフトウェアのインストール**」を選択します。
2. ポップアップされるリポジトリの追加ダイアログ（*下のスクリーンショット*）で **追加**ボタンクリックします。



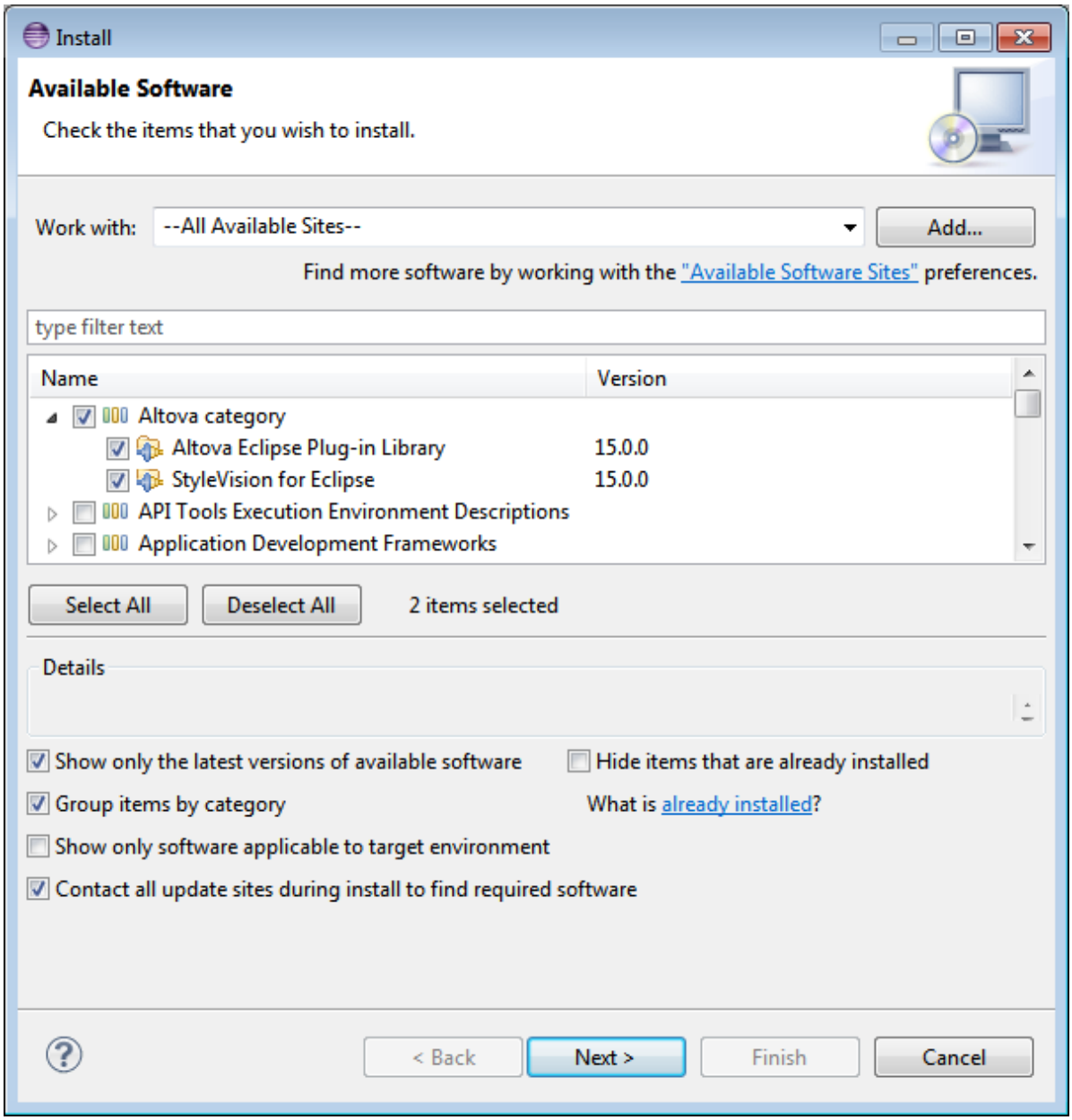
3. ポップアップされるリポジトリの追加ダイアログ（*下のスクリーンショット*）で **ローカル** ボタンクリックします。
4. フォルダー `c:\Program Files\Altova\Common2021\eclipse\UpdateSite` を参照します。('Altova' などの) サイトの名前を入力し、「OK」をクリックします。



5. ステップ 2 から 4 を繰り返します、今度は、フォルダー `c:\Program Files\Altova\MapForce2021\eclipse\UpdateSite` を選択し、'Altova StyleVision' などの名前を与えま

す。

6. *インストールダイアログの作業対象コンボボックスにて ―すべての使用可能なサイト― オプションを選択します（以下のスクリーンショットを参照）。これにより使用可能なすべてのプラグインは下のペイン内に表示されます。Altova カテゴリフォルダーのトップレベルにコンボボックスをチェックします（下のスクリーンショットを参照してください）。「次へ」ボタンをクリックします。*



7. インストール詳細画面にてインストールするアイテムを確認します。**次へ** をクリックします。
8. 表示されるライセンスのレビュー画面にて、**使用条件の条項に同意します** を選択し、**完了** ボタンをクリックしてインストールを行います（StyleVision Enterprise または Professional Edition ライセンスに加えて StyleVision プラグインのライセンスへ新たに同意する必要はありません）。

アイコンが表示されないなどの問題が発生した場合は、-clean プラグ付きで Eclipse を起動してみてください。

現在インストールされているバージョン

現在インストールされている StyleVision プラグインのバージョンを確認するには、Eclipse のメニューオプションから「**ヘルプ | Eclipse について**」を選択します。そこから StyleVision アイコンを選択してください。

関連項目

- StyleVision 統合

15.2 Eclipse Stylevision 内のエントリポイント

次のエントリポイントを使用して、Eclipse では以下の方法により StyleVision の機能へアクセスすることができます:

- StyleVision パースペクティブ: StyleVision の GUI 機能が Eclipse の GUI に表示されます。
- StyleVision メニューとツールバー

StyleVision パースペクティブ

Eclipse におけるパースペクティブとは、特定の機能に結び付けられた GUI ビューのことです。Eclipse 用の StyleVision プラグインが Eclipse に統合されると、StyleVision パースペクティブも自動的に作成されます。このパースペクティブは、編集ビュー、メニュー、入力ヘルパー、そしてサイドバーなど、StyleVision の GUI 要素を含むパースペクティブとなります。

StyleVision と関連付けられている種類のファイル(例えば .sps) が開かれた時、このファイルを StyleVision パースペクティブで編集することができます。同様に、他の種類のファイルも Eclipse の別のパースペクティブで開くことができます。更に、アクティブなファイルに対しては、パースペクティブを変更することができ、異なる環境で同じファイルの編集を行うことができます。従って、パースペクティブには2つの利点があります:

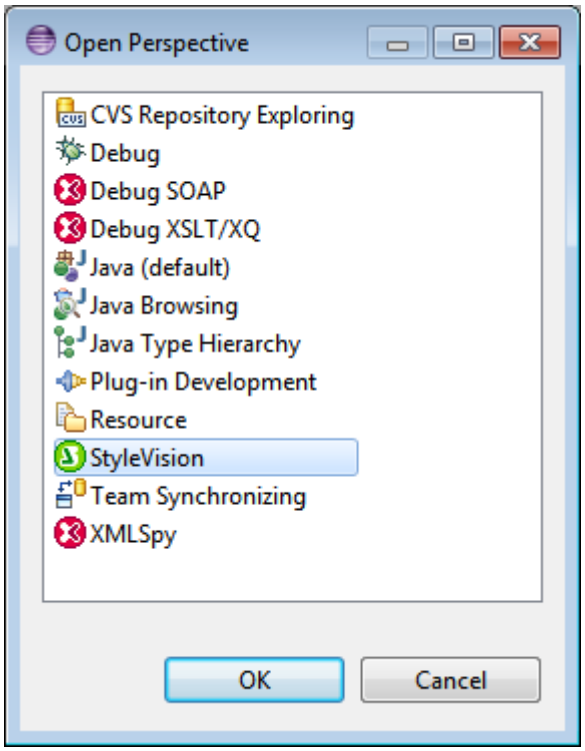
1. アクティブなファイルに対して、作業環境を素早く変更することができる。
2. 新たな開発環境を開くこと無くファイルの切り替えを行うことができる(パースペクティブに結び付けられた環境も有効になります)。

StyleVision パースペクティブを使った作業には以下のようなものがあります:

- StyleVision パースペクティブへの切り替え
- StyleVision パースペクティブの環境設定
- StyleVision パースペクティブのカスタマイズ

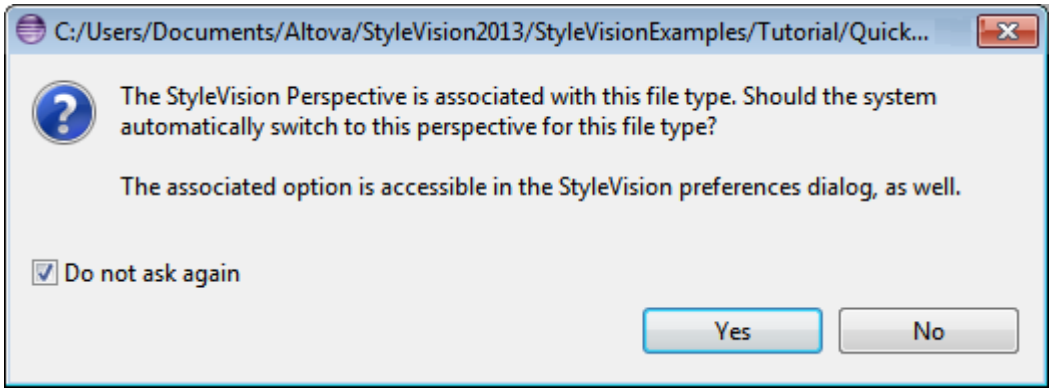
StyleVision パースペクティブへの切り替え

Eclipse にて、**メニューコマンド**から「**ウィンドウ | パースペクティブ | パースペクティブを開く | その他**」を選択します。表示されるダイアログボックス内で(*下のスクリーンショット*) select **StyleVision**を選択して[開く]をクリックします。



空のドキュメント、またはアクティブなドキュメントに対して StyleVision パースペクティブが開かれます。この方法でユーザーはパースペクティブの切り替えを行うことができます。目的のパースペクティブを、パースペクティブを開くサブメニュー（のその他アイテムの上）に表示させることで、パースペクティブにより素早くアクセスすることができます。

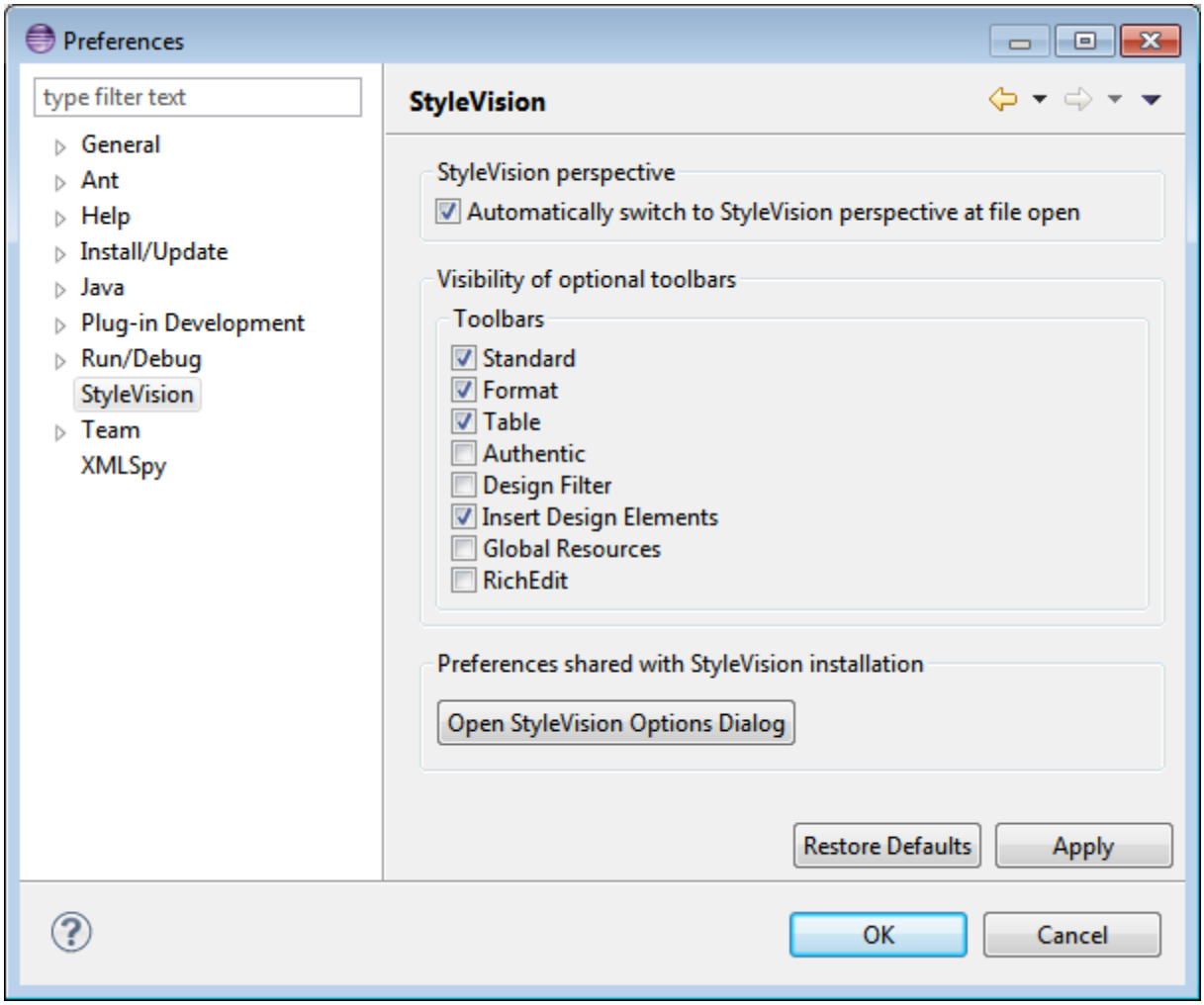
この設定はカスタマイズダイアログにて行うことができます。ファイルが開かれた時、またはアクティブになったときにパースペクティブを自動的に切り替えることもできます。そのファイルが最初に開かれたときには、ファイルの種類に関連付けられているアプリケーションのパースペクティブが自動的に開かれ、このファイルの種類に対して、このパースペクティブをデフォルトのパースペクティブとして割り当てるか尋ねられます。



この種類のファイルを開くたびに尋ねられたくない場合は、*今後このメッセージを表示しない* にチェックを入れ「OK」をクリックします。

StyleVision パースペクティブの環境設定

パースペクティブの優先順位には以下が含まれています: StyleVision パースペクティブの環境設定パースペクティブの設定には、(i) 関連付けられた種類のファイルが開かれたときに、自動的にパースペクティブを切り替える、（*上の参照*）(ii) 各 StyleVision ツールバーを含めるオプションが含まれます。設定ダイアログ（*下のスクリーンショット*）にアクセスするには、[**ウィンドウ | 設定**] メニューコマンドを選択します。



左のペインにあるパースペクティブのリストから StyleVision を選択し、必要な設定を選択します。「OK」ボタンをクリックして完了します。

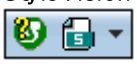
StyleVision パースペクティブのカスタマイズ

カスタマイズオプションを使うことで、ショートカットやコマンドをパースペクティブに加えることができます。(*以下のスクリーンショットにある*) パースペクティブのカスタマイズダイアログにアクセスするには、(*StyleVision パースペクティブ*の場合は) パースペクティブをアクティブにして、メニューコマンドから[**ウィンドウ | パースペクティブ | パースペクティブのカスタマイズ**]を選択します。
ツールバー可視性とメニュー可視性タブ内では、度のツールバーとメニューが表示されるかを指定することができます。アクションセットの使用可能タブでは、親メニューとツールバーにアクションセットを追加することができます。アクショングループ可用性タブでは、アクションセットを親メニューやツールバーへ追加することができます。アクショングループを有効にするには、対応するチェックボックスにチェックを入れてください。カスタマイズパースペクティブダイアログのショートカットタブでは、サブメニューのショートカットをセットすることができます。サブメニューコンボボックスから目的のサブメニューを選択し、ショートカットのカテゴリを選択し、パースペクティブに追加したいショートカットにチェックを入れます。[**適用 & 閉じる**]をクリックしてカスタマイズを完了すると変更が反映されます。

StyleVision メニューとツールバー

The **StyleVision** メニューにはStyleVision により認識されるドキュメントの型が現在 Eclipse 内で開かれていない場合でも関連性のあるコマンドが含まれています。StyleVision のスタンダードバージョンではコマンドの一部は「ファイル」メニュー内に存在しています。

StyleVision ツールバーには次のボタンが含まれています（*下のスクリーンショット参照*）。



これらは左側から：(i) StyleVision ヘルプを開く (ii) (**StyleVision** メニューからアクセスすることの代替として) StyleVision コマンドにアクセス

関連項目

- StyleVision 統合

16 メニューコマンドとレファレンス

このセクションでは StyleVision ツールバー、デザインビューのシンボル、そしてメニューコマンドについて説明します。セクションは以下のように構成されます:

- デザインビューで使用されているシンボルの説明。
- XPath 式の編集ダイアログの説明。
- すべてのツールバーとそれらのアイコン、そしてツールバーの表示をカスタマイズする方法。
- すべてのメニューコマンド。

ユーザーリファレンスのセクションでは個々のコマンドに関する記述がなされている一方で、関連するセクションでは StyleVision の機能で使用されているメカニズムの詳細について説明されます。メカニズムは以下のグループにより構成されています:

- SPS ファイルコンテンツ
- SPS ファイル構造
- SPS ファイルの高度な機能
- SPS ファイル表示方法
- SPS ファイル追加機能
- SPS ファイルとデータベース

コマンドラインの使用方法に関しては、次を参照してください: コマンドライン インターフェイス: StyleVisionBatch

照関連項目

- ユーザーインターフェイス
- チュートリアルのクリックスタート

(C) 2015-2021 Altova GmbH

16.1 デザイン ビューのシンボル

SPS のデザインには通常、複数種類のコンポーネントが含まれ、各コンポーネントはそれぞれ異なるシンボルでデザイン内に表示されます。これらのシンボルは以下のようにグループ分けされます。:

- XML ドキュメント内のノード
- XML ドキュメント コンテンツ
- データ入力デバイス
- 定義済みのフォーマット
- XPath オブジェクト
- URI オブジェクト

これらコンポーネントに対して以下の操作を行うことができます:

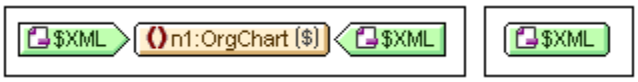
- ドラッグ アンド ドロップにより移動する
- (i) 編集メニューのコマンド、または (ii) 標準的な Windows のショートカットにより切り取り、コピー、貼り付け、または削除する
- フォーマットを適用する
- 右クリックすることで、コンテキスト メニューを表示する

XML ドキュメント内のノード

XML ドキュメント内にある要素や属性は SPS デザインドキュメント内でタグとして表示されます。各ノードには開始タグと終了タグが含まれます。開始タグまたは終了タグをダブルクリックすることでそのノードが縮退され、ノードが縮退されると、ノード内部に含まれているすべてのコンテンツが隠されます。縮退されたノードをダブルクリックすることで、ノードが展開され、内部に含まれているコンテンツが表示されます。

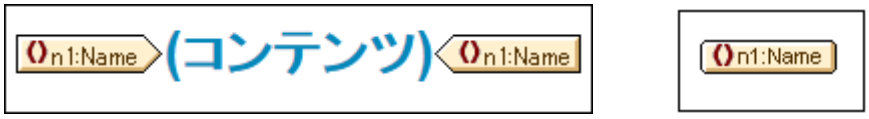
ノードには以下の種類が存在します:

- *ドキュメントノード*



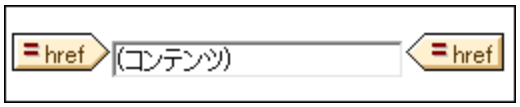
$XML により示されるドキュメント ノードは XML ドキュメント全体を表します。スキーマ ソースが XML ドキュメントに関連付けられている場合には緑色の $XML タグが表示され、スキーマ ソースがデータベースに関連付けられている場合には黄色の $DB タグが表示されます。上のスクリーンショットでは、OrgChart を含むドキュメント ノードが左側に、右側には縮退されたドキュメント ノードが示されています。.

- *要素ノード*



挿入された場所に祖先ノードが存在しない場合、すべての祖先ノードとともに要素ノードが挿入されます。上のスクリーンショットでは、展開 (左側) ならびに縮退 (右側) された Name という要素ノードが示されます。

- *属性ノード*



挿入された場所に祖先ノードが存在しない場合、すべての祖先ノードとともに**属性ノード**が挿入されます。属性ノードには = という頭文字が表示されます。上のスクリーンショットでは、href 属性ノードが展開されたかたち（左側）ならびに縮退されたかたち（右側）で示されます。

ノードはデザイン内にノードテンプレートとして含まれます。デザインに含むことのできる異なる種類のテンプレートに関しては、次のセクションを参照してください：, テンプレートとデザインフラグメント.

XML ドキュメント コンテンツ

XML ドキュメント コンテンツは 2 種類のプレースホルダにより表示されます:

- `(contents)`
- `(rest-of-contents)`

contents プレースホルダは単一ノードのコンテンツを表し、ノード内に含まれるすべてのテキスト コンテンツが出力されます。ノードが属性ノードまたはテキストのみの要素ノードの場合、ノードの値が出力されます。ノードが混合コンテンツや要素コンテンツだけで構成される要素ノードの場合、子孫ノードのテキスト コンテンツが出力されます。XSLT の用語に置き換えると、**contents** プレースホルダは、select 属性の値によりノードが指定された xsl:apply-templates 要素と等しい振る舞いを行います。

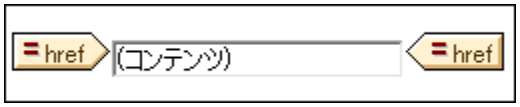
メモ **contents** プレースホルダが要素ノードに適用されると、その要素の属性に含まれる値は出力されません。属性ノードの値を出力するには、属性を（メインまたはグローバル）テンプレートに明示的に含める必要があります。

rest-of-contents プレースホルダにより、カレント ノード以下にあるその他の子要素に対してテンプレートが適用されます。この場合、各子要素に対して適用されるテンプレートはグローバルテンプレート（その要素に対して定義されている場合）、または要素のデフォルト テンプレート（テキストのみの要素からはテキストだけを出力し、子要素のテンプレートを適用）となります。例えば、title、author、isbn、pubdate という子要素を含んだ book という要素を考えてみましょう。book の定義により title 子要素だけが出力されると指定された場合、この定義が処理されることで、その他の子要素（author、isbn、pubdate）は出力されないことになります。しかし、book の定義内の title 要素直後に**その他のコンテンツ** プレースホルダが含まれている場合、その他の子要素（author、isbn、pubdate）、グローバルテンプレート（その要素に対する定義が存在する場合）または要素のデフォルト テンプレートも適用されることになります。

データ入力デバイス

デザインにてデータ入力デバイスを配置することで、Authentic View ユーザーによる XML ドキュメントの正確な編集と、妥当なデータの入力がサポートされます。ノードには以下のデータ入力デバイスを割り当てることができます:

- *入力フィールド*（単一行と複数行）



- *コンボボックス*

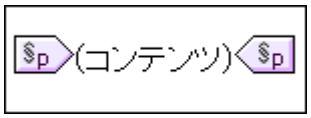
- *チェックボックス*

- *ラジオボタン*

展開されたタグはダブルクリックすることで縮退することができますし、縮退されているタグも、ダブルクリックにより展開することができます。データ入力デバイスの細かな使用方法については、データ入力デバイスを参照ください。

定義済みのフォーマット

定義済みフォーマットは薄い紫色のタグで表示され、ダブルクリックにより展開/縮退することができます。



上のスクリーンショットでは、定義済みのフォーマット p (para) が展開された状態 (左側) と縮退された状態 (右側) で示されています。定義済みのフォーマットを適用するには、目的のアイテムを (そのアイテムをクリックするかテキストをマーキングすることで) ハイライトして、定義済みのフォーマットを挿入します。。

XPath オブジェクト

StyleVision ではXPath 条件式を使用するために2 つのメカニズムが用意されています。

- *条件テンプレート*

条件タグは青色により表示されます。開始タグにはセルが含まれており、左端のセルには疑問符が含まれています。その他のセルには (i) 個々の when 条件に対応した 1 からはじまる数値か、(ii) その他の条件を表すオプションのアスタリスクが表示されます。条件分岐はクリックにより選択でき、選択された条件分岐の番号が開始タグにてハイライトされ、その分岐に対応するテンプレートが条件の開始タグと終了タグの間に表示されます。選択された条件分岐の XPath 条件式もデザイン ツリーでハイライトされます。条件のタグを縮退/展開することはできません。

- *自動計算*

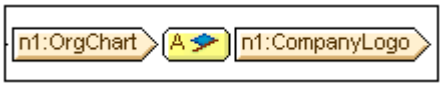
自動計算はデザイン ビューにて =(自動計算) というオブジェクトにて表示されます (上のスクリーンショットを参照)。選択された自動計算の XPath 条件式がデザイン ツリーにてハイライトされ、プロパティ サイドバーから[プロパティサイドバー自動計算の編集をおこなうダイアログへアクセス](#)することができます。

URI オブジェクト

デザイン内部へ挿入することのできる URI をベースにしたオブジェクトには、以下の3 種類があります:

- *イメージ*
  StyleVision からアクセスできるイメージが SPS デザインへ挿入されると、イメージがデザイン ビューに表示されます。イメージへアクセスできない場合、SPS はイメージのプレースホルダによりマークされます。

- *ブックマーク (アンカー)*



ブックマークは A という文字の入った黄色のタグにより表示されます (上のスクリーンショットを参照)。メニューから [**挿入 | ブックマークの挿入**] を選択することで空のブックマークが追加されるほか、コンテンツが選択された状態で [**タグで囲む | ブックマーク**] のメニュー オプションを選択することで、ブックマークを挿入することができます。アンカー タグは展開 (左側のスクリーンショット) ならびに縮退 (右側のスクリーンショット) することができます。

- *リンク*

ブックマークは A という文字の入った黄色のタグにより表示されます (上のスクリーンショットを参照)。メニューから [**挿入 | ブックマークの挿入**] を選択することで空のブックマークが追加されるほか、コンテンツが選択された状態で [**タグで囲む | ブックマーク**] のメニュー オプションを選択することで、ブックマークを挿入することができます。アンカー タグは展開 (左側のスクリーンショット) ならびに縮退 (右側のスクリーンショット) することができます。

関連項目

- [ツールバー](#)
- [デザイン サイドバー](#)
- [コンテンツ編集処理](#)

(C) 2015-2021 Altova GmbH

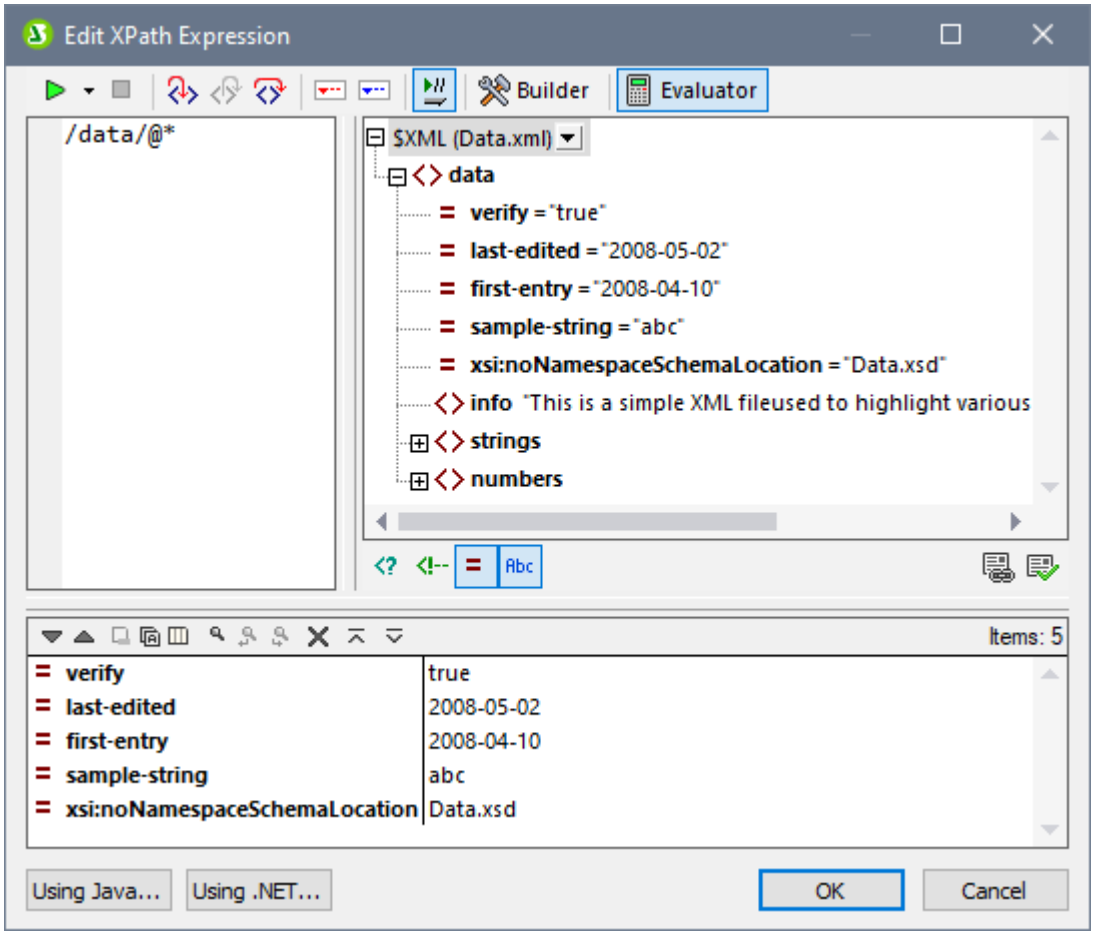
16.2 XPath 式 の編集ダイアログ

XPath 式 の編集ダイアログ（*下のスクリーンショット*）はXPath 式 の編集を作成、テストするために使用されます。条件付きの処理 または スタイル と プロパティ の値のために式を入力する場合など XPath 式が入力されるデザインビュー内のすべての箇所で使用することができます。

ダイアログはSPS（XSLT 1.0 のためのXPath 1.0、XSLT 2.0 のためのXPath 2.0、XSLT 3.0 のためのXPath 3.1）のXSLT バージョン に対応するXPath バージョン を自動的にサポートします。XPath バージョンを切り替えるには、SPS のXSLT バージョンを切り替えます。

ダイアログのレイアウト

XPath 式 の編集には以下のペインが含まれています（*下のスクリーンショットを参照*）: (i) 式ペイン（*左上*）(ii) ソースペイン（*右上*）(iii) 結果ペイン（*下*）。ビルダーモードでは結果ペインは入力ヘルパーペインにより置き換えられています。

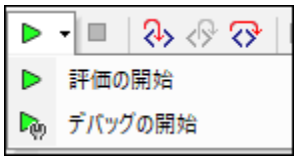


評価モードとデバッグモード

XPath 式 の編集ダイアログ には2 つのモードが存在します:

- XPath 式が割り当てられた作業 XML ファイルに対して評価される評価モード。**式ペイン** に式は入力され、結果は**結果ペイン** に表示されます。結果内のノードをクリックしてダイアログのソースペイン内のノードに移動することができます。
- 割り当てられた作業 XML ファイルに適用し XPath 式をデバッグすることのできるデバッグモード 。ブレークポイントとトレースポイントを設定し、評価をステップバイステップで行うことができます。各ステップで変数のコンテンツを確認することができ、評価の追加アスペクトをチェックすることのできるカスタムのウォッチ式を設定することができます。で

ウィンドウのツールバーの左角にある「**評価/デバッグの開始**」ドロップダウンメニュー内から適切なコマンドを選択し、2 つのモード間を切り替えることができます *(下のスクリーンショットを参照)*.



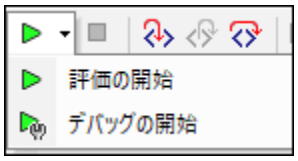
2 つのモードの使用方法はこのセクションのサブセクション内で説明されています。

XPath/XQuery 式ビルダー

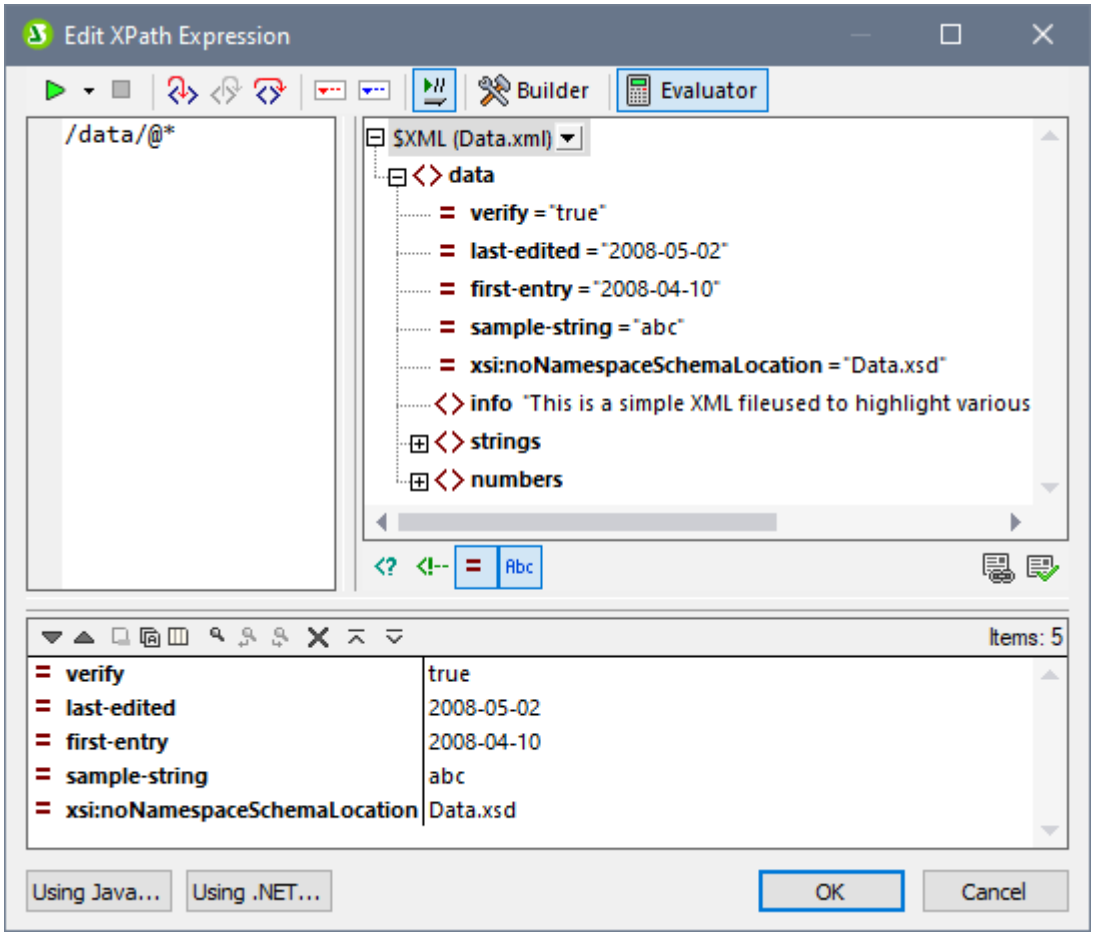
両方のモード内で、式ビルダーを使用して構文的に正しい式を作成することができます。メインツールバー の「**ビルダーモード**」ボタンを使用して式ビルダーを切り替えます。

16.2.1 エバリュエーター

「**評価/デバッグの開始**」ドロップダウンメニュー内で「**評価の開始**」を選択し評価モードを選択します *(下のスクリーンショットを参照)*.



評価モード内で**エバリュエーター** ボタンをクリックします *(下のスクリーンショットを参照)*。エバリュエーターには次のペインが存在します *(下のスクリーンショットを参照)*: (i) 式ペイン *(左上)* (ii) ソースペイン *(右上)* (iii) 結果ペイン *(下)*。



XPath 式とその評価

XPath 式は式ペイン内に入力されます。結果ペイン内に評価の結果は表示されます *(上のスクリーンショット)*。

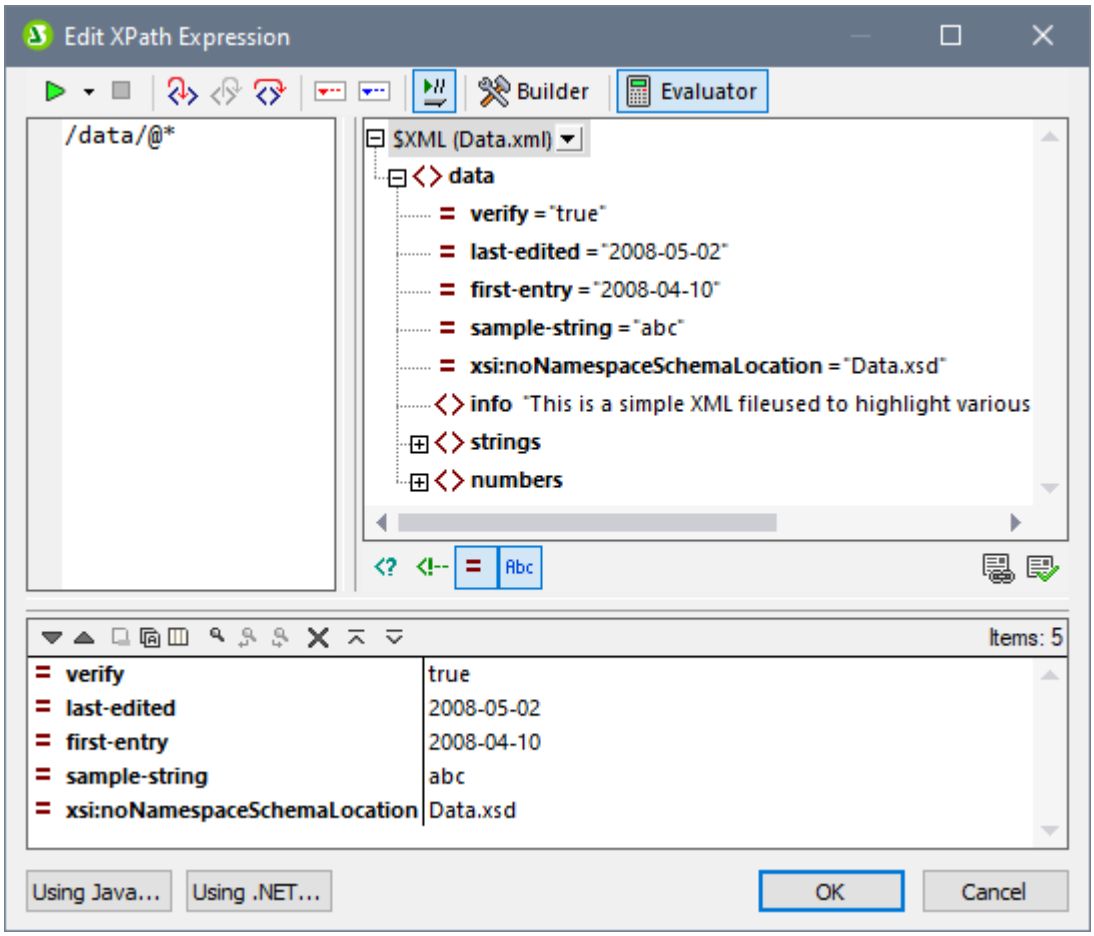
以下の点に注意してください：

- XML ファイルに対して式を評価するには、そのファイルは SPS のソースの一つのファイルと作業 として割り当てられる必要があります。
- (**入力中に評価**」iアイコンをツールバーから選択すると)，結果は式の入力中に表示されます。または、(ツールバーの左上の)「**評価/デバッグの開始**」(F5) をクリックすると表示されます。
- ソースツリー内のノードのXPath ロケーター式を入力するにはソースペイン内のノードをダブルクリックします。
- ソースペイン内では、以下の表示を切り替えることができます: (i) 処理命令 (ii) コメント (iii) 属性 (iv) 要素。ソースペインの下のボタンを使用してこれを行ってください。
- コンテキストノードは式が作成されるデザインコンポーネントを指します。式のコンテキストノードとして他のノードを設定するには以下を行います: (i) ソースペイン内のノードを選択します (ii) (ソースペインの下にある)**評価コンテキスト** をクリックします。このコンテキストノードを式のために保存するには、(ソースペインの下にある)「**評価コンテキストを記憶する**」コマンドボタンを切り替えてください。式のために実際のコンテキストノードは現在のデザインコンポーネントのコンテキストノードになり、ランタイムで使用されるコンテキストノードになります。
- XPath 式内で Java と.NET プログラミング言語の関数を使用することができます。ダイアログの下の「**Java の使用**」と「**.NET の使用**」ボタンはXPath式内のJava と.NET 拡張関数の使用法に関する説明をする情報ボックスを表示します。詳細に関してはこのドキュメントの拡張関数セクションを参照してください。

- （より読みやすくするために）式を複数行で表示するには **リターン** キーを使用します。
- 式フィールド内のテキストのサイズを大きくする/小さくするには、式フィールドをクリックし、**Ctrl** を押して、スクロールホイールを使用します。*結果ペイン内にもこの機能が適用されることに注意してください。*
- ノードのロケーター パス式を手動で挿入する代わりに、以下を行うことができます: (i) ロケーター パスを挿入する XPath 式内の場所にカーソルをポイントします (ii) ソースツリー内でターゲットとするノード内にカーソルをポイントします。式内の選択されたノードのロケーター パスを入力します。ロケーター パスはドキュメントのルートノードから始まる絶対パスです。

結果ペイン

結果ペインは下のスクリーンショット内で表示されています。結果ペイン固有ののツール バーが存在することに注意してください。



結果ペインには以下の機能が存在します:

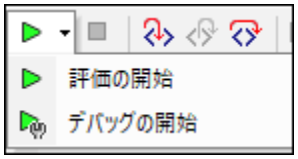
- 結果リストは2つの列により構成されています: (i) ノード名、またはデータ型 (ii) ノードのコンテンツ。
- XPath 式が（要素または属性などの）ノードを返す場合 ノードのコンテンツ全体がノードの値として表示されるかを選択することができます。これを行うには、*完全な結果を表示する*をオンに切り替えてください。
- 式により生成されるリテラルとは異なり、結果に（テキスト ノードを含む）ノードが含まれる場合、結果ペイン内でそのノードをクリックすると、ソースツリー内の XML ドキュメント内の関連するノードがハイライトされます。
- 結果サブラインの列、または、値列の両方をコピーすることができます。全ての列をコピーするには、サブラインを右クリックし「**全ての列を含みコピーする**」をオンに切り替えます。（または、結果ペインのツール バー内のアイコンを使用してコマンドをオンとオフに切り替えることもできます。）コピーするサブラインを右クリックし、（そのサブラインのために）「**サブラインをコピーする**」または「**全てをコピーする**」（全てのサブラインのために）を選択します。

結果ペインのツールバー
結果ペインのツールバーにはナビゲート、検索、およびコピー機能を提供するアイコンが含まれています。これらのアイコンは、左から順番に以下で説明されています。対応するコマンドは結果リストアイテムのコンテキストメニュー内で使用することができます。

アイコン	動作
次、前	結果リストからそれぞれ次のおよび前のアイテムを選択します。
クリップボードに選択されたテキストラインをコピーする	クリップボードに選択された結果アイテムの値カラムをコピーします。全てのカラムをコピーするには、全てのカラムを含みコピーするコマンドに切り替えてください（*下を参照*）。
全てのメッセージをクリップボードにコピーする	空の値を含む値カラムのすべての結果アイテムをクリップボードにコピーします。各アイテムは異なるラインにコピーされます。
全てのカラムを含みコピーする	(i) 全てのカラム、または(ii) 値カラムのコピー間を切り替えます。カラムの区切りは単一のスペースです。
検索する	結果リスト内で特別文字を含む文字列を検索する検索ダイアログを開きます。
前を検索する	検索ダイアログ内に最後に入力された用語の前の発生を検索します。
次を検索する	検索ダイアログ内に最後に入力された用語の次の発生を検索します。
クリア	結果のリストをクリアする
複数のラインの結果を単一のラインに折りたたむ	結果アイテムの値カラムが複数ラインのテキストを含む場合、（新しいラインまたは文字を含む場合）、複数ラインと単一ラインの表示を切り替えることができます。
完全な結果を表示する	ノードの値としてノードのコンテンツ全体を表示します。

16.2.2　デバッガー

デバッガーにより XML ファイルと作業 のコンテキスト内で XPath 式をデバッグすることができます。「**評価/デバッグの開始**」ドロップダウンメニュー内で「**デバッグの開始**」にアクセスします（*下のスクリーンショット*）。これによりデバッグモードに設定されます。式を作成するためのヘルプのための）ビルダーと（式をデバッグするための）エバリュエーター間を切り替えることができます。デバッグを開始するために「**評価/デバッグの開始**」（F5）をクリックします。



式の挿入後（デバッグモードにモードが切り替えられていることを確認して）「**評価/デバッグの開始**」（F5）をクリックしてデバッグを開始することができます。.

デバッグモードをセットアップするためのボタン

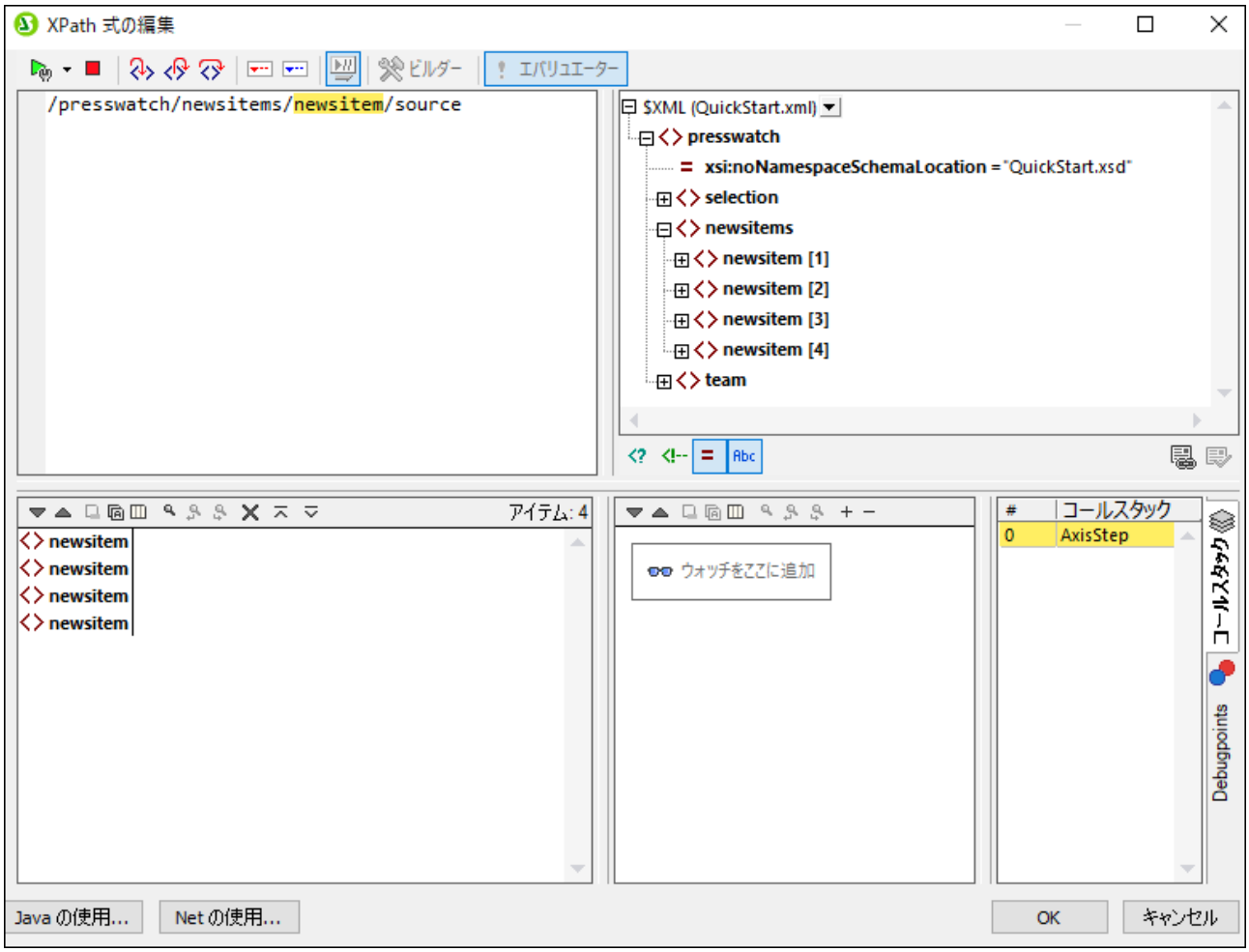
	評価/デバッグの開始（F5）	デバッガーを開始します。

	ビルダーに切り替える	式の作成を手助けするコンテキストを考慮する入力ヘルパーを提供する式ビルダーモードに切り替えます。
	入力中に評価する	式の挿入中に式の評価を切り替えます。

デバッグモードのレイアウト

デバッグモード内では、結果ペインに2つのペインが追加されます（*下のスクリーンショットを参照*）:

- ペイン内に個別のタブを持つコールスタックとデバッグポイントペイン
- ウォッチ式と変数が同時に同じペインに表示される変数とウォッチ式ペイン



デバッガーモードには以下の機能が搭載されています:

- XPath 式がどのように評価されるかをステップごとに確認するためにXPath 評価プロセスにステップインすることができます。ステップイン（F11）ツールバーボタンを使用して、これを行います。各評価ステップで、現在評価されている式の一部が黄色でハイライ

トされており、そのステップの評価の結果は結果ペインに表示されます（*上のスクリーンショット*）。例えば、上のスクリーンショットでは、 `book` 要素のすべての `section` 子孫要素が選択されています。

- 評価を一時停止する箇所でブレークポイントを設定し、これらの箇所で結果をチェックします。ブレークポイントでのみ一時停止し評価をステップスルーすることができます。「**デバッグの開始**」（F5）ツールバーボタンをこのために使用します。**ステップイン**（F11）を使用したすべてのステップで一時停止するより先素早く行うことができます。
- トレースポイントとしてマークされるステップでの結果を確認するようにトレースポイントを設定します。（ブレークポイント以外）評価は一時停止しませんが、結果ペイン内のリスト内にトレースポイントの結果は表示されます。
- ウォッチ式を使用して（評価のドキュメントデータまたはアスペクトなどの）情報をチェックすることができます。この機能はブレークポイントで役に立ちます。
- 変数とウォッチ式ペイン内に表示されるスコープ内の値を含む変数。
- 評価ステップのプロセッサーコールはコールスタックとデバッグポイントペインのコールスタックタブ内で表示されます。
- ブレークポイントとトレースポイントが設定されると、コールスタックとデバッグポイントペインのデバッグポイントタブ内で表示されます。

これらの機能の情報に関しては、下の詳細を参照してください。

デバッガーの実行

XPath 式のデバッグのための大まかなステップは以下の通りです:

1. XPath 式を式ペインに入力します。
2. ブレークポイント、またはトレースポイントを設定します。ブレークポイントは評価を停止するポイントを指します。トレースポイントは記録される評価内のポイントを指します。トレースポイントは従って評価結果のトレース可能なパスを提供します。
3. 「**デバッガーの開始**」をクリックすると、式内にブレークポイントがマークされていない限り、評価が最後のステップの前の段階まで行われます。「**デバッガーの開始**」を繰り返しクリックしてブレークポイントを進め評価の最後まで移動します。
4. ステップイン/アウト/オーバー機能を使用して評価をステップバイステップに行います。

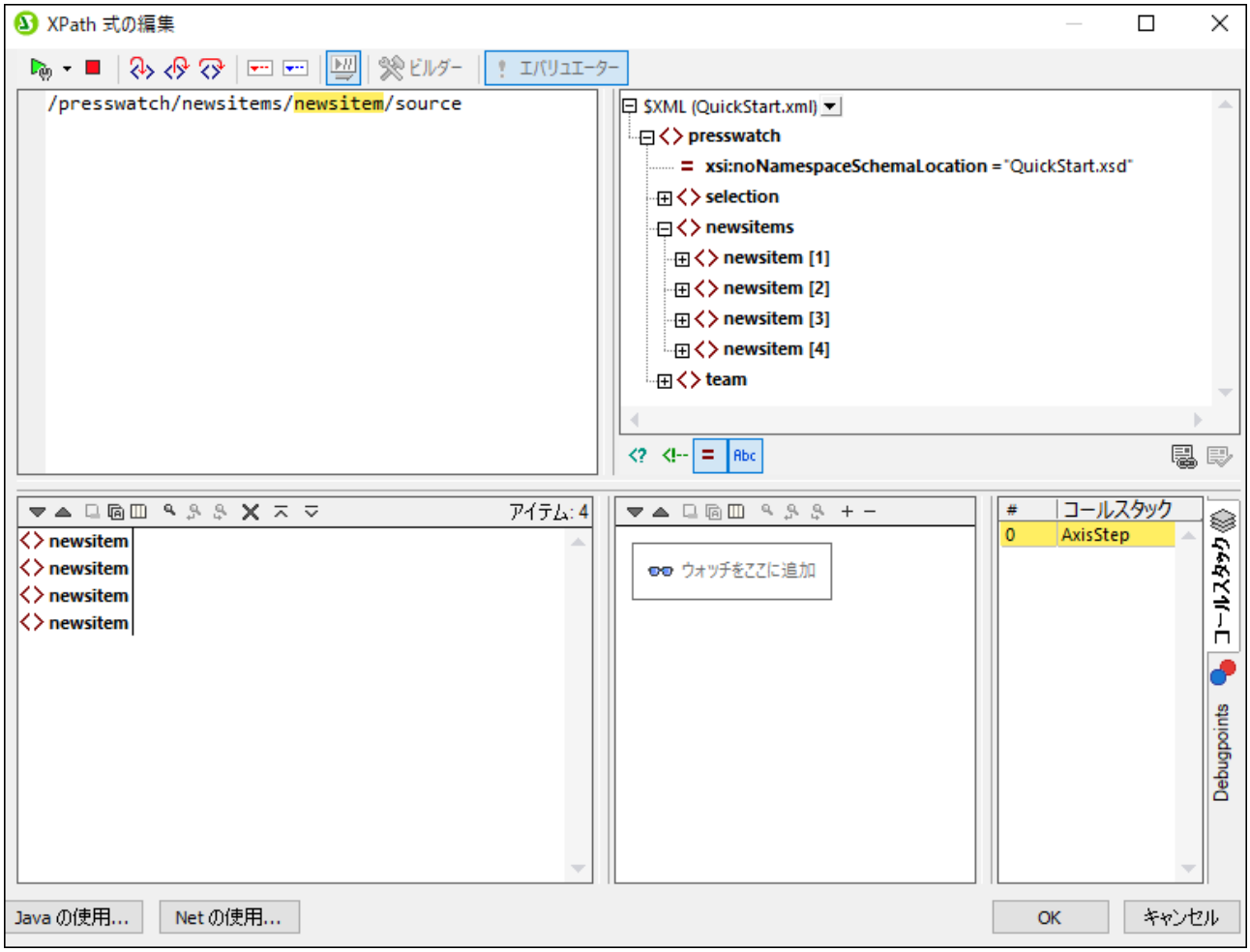
デバッグのためのボタン

	デバッガーの開始（F5）	デバッガーを開始します。評価はブレークポイントのために停止する以外最後まで行われます。
	デバッガーの停止（Shift+F5）	評価を停止しデバッガーを停止します。
	ステップイン（F11）	評価を継続し、ステップごとに評価を行います。
	ステップアウト（Shift+F11）	現在の評価ステップをステップアウトし、親ステップに移動します。
	ステップオーバー（Ctrl+F11）	子孫ステップにステップオーバーします。
	ブレークポイントの挿入/削除（F9）	カーソルがポイントする箇所の式ステップでブレークポイントを挿入/削除します。
	トレースポイントの挿入/削除 (Shift+F9)	カーソルがポイントする箇所の式ステップでトレースポイントを挿入/削除します。

ステップイン、アウト、およびオーバーの評価ステップ

*ステップイン*機能により評価をステップバイステップで行うことができます。このコマンドの各クリックにより評価の次のステップに移動することができます。現在のステップは式内でハイライトされ表示されます（*下のスクリーンショットを参照*）。*ステップアウト*機能により現在のステップとして高いレベルのステップに移動することができまが、*ステップオーバー*機能は低いレベルのステップにステップオーバーし、同じレベルの次のステップに移動することができます。*ステップ*機能を下のスクリーンショット内で表示されている式を使用して、3つの*ステップ*ボタンを押してそれぞれの仕組みを試すことができます。

下のスクリーンショットはロケーターステップ `newsitem` に到達すると一時停止される際の評価を表示しています。この時点では結果は4つの `newsitem` ノードを表示しています。



ブレークポイント

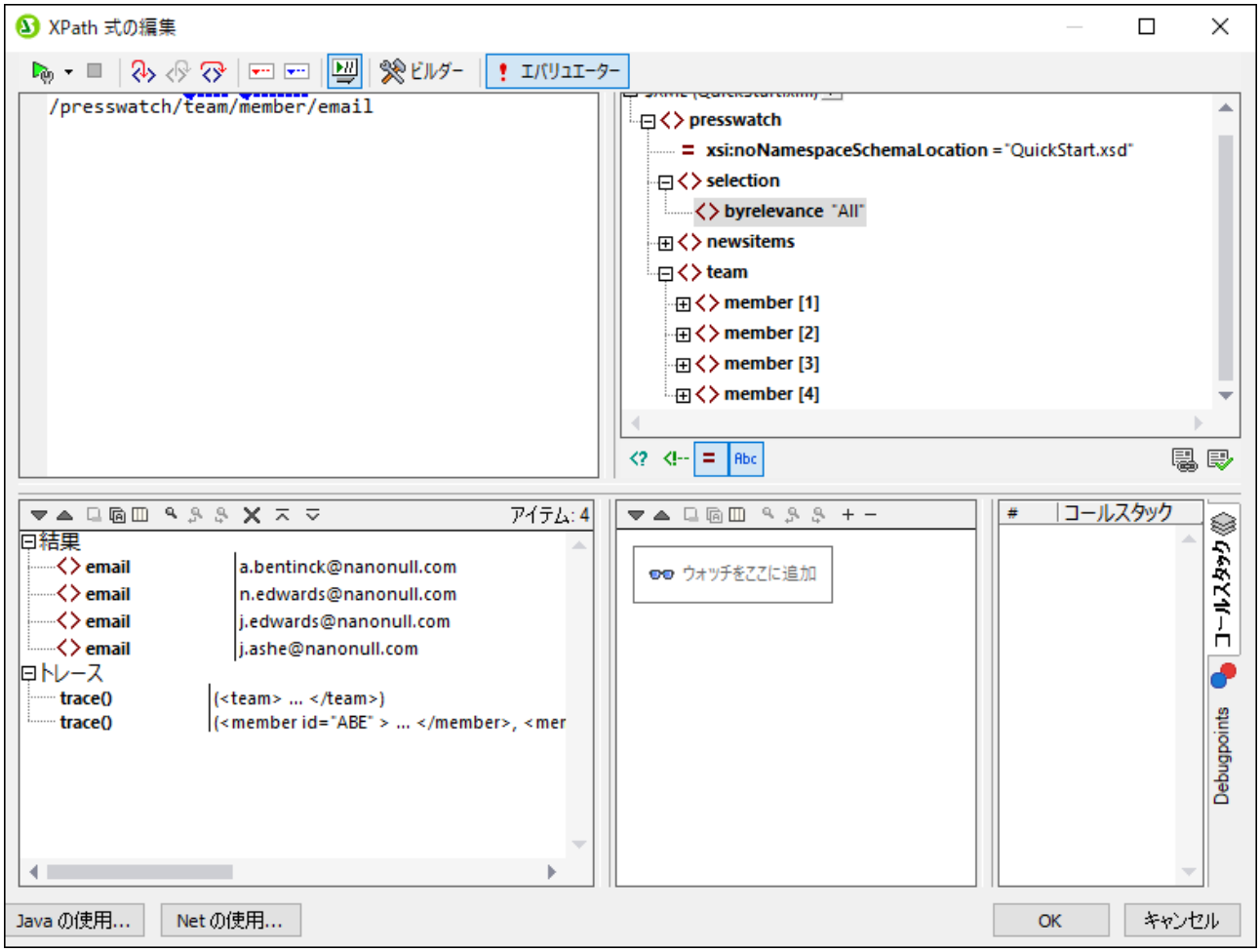
ブレークポイントは「**デバッガーの開始**」により開始された後にデバッガーを停止するためのポイントです。ブレークポイントでデバッガーが停止すると、結果をチェックして、**ステップイン**機能を使用して評価の次のステップの結果を表示することができます。ブレークポイントを設定するには、式内のブレークポイントを設定する箇所にカーソルをポイントし、「**ブレークポイントの挿入/削除**」(F9) ツールバーボタンをクリックします。ブレークポイントは赤い点線のオーバーラインでマークされています。ブレークポイントを削除するには、選択し「**ブレークポイント/削除**」**(F9)**をクリックします。

下の[デバッグポイント](#) も参照してください。

トレースポイント

トレースポイントは結果が記録されるポイントです。これらの結果は *結果* タブのトレースツリーで表示されています (*下のスクリーンショットを参照*)。これにより式の特定の部分のすべての評価を確認することができます。例えば、下のスクリーンショット内では、トレースポイントが `team` ノードと`member` ノード上で設定されています。これらのトレースポイントでの結果はトレースツリー内で表示されています。

トレースポイントを設定するには、トレースポイントを設定するポイントにカーソルをポイントし、ツールバーボタン「**トレースポイントの挿入/削除**」(Shift+F9) をクリックします。トレースポイントは青い点線のオーバーラインでマークされています (*下のスクリーンショットを参照*)。トレースポイントを削除するには、選択し「**トレースポイントの挿入/削除**」(F9)をクリックします。

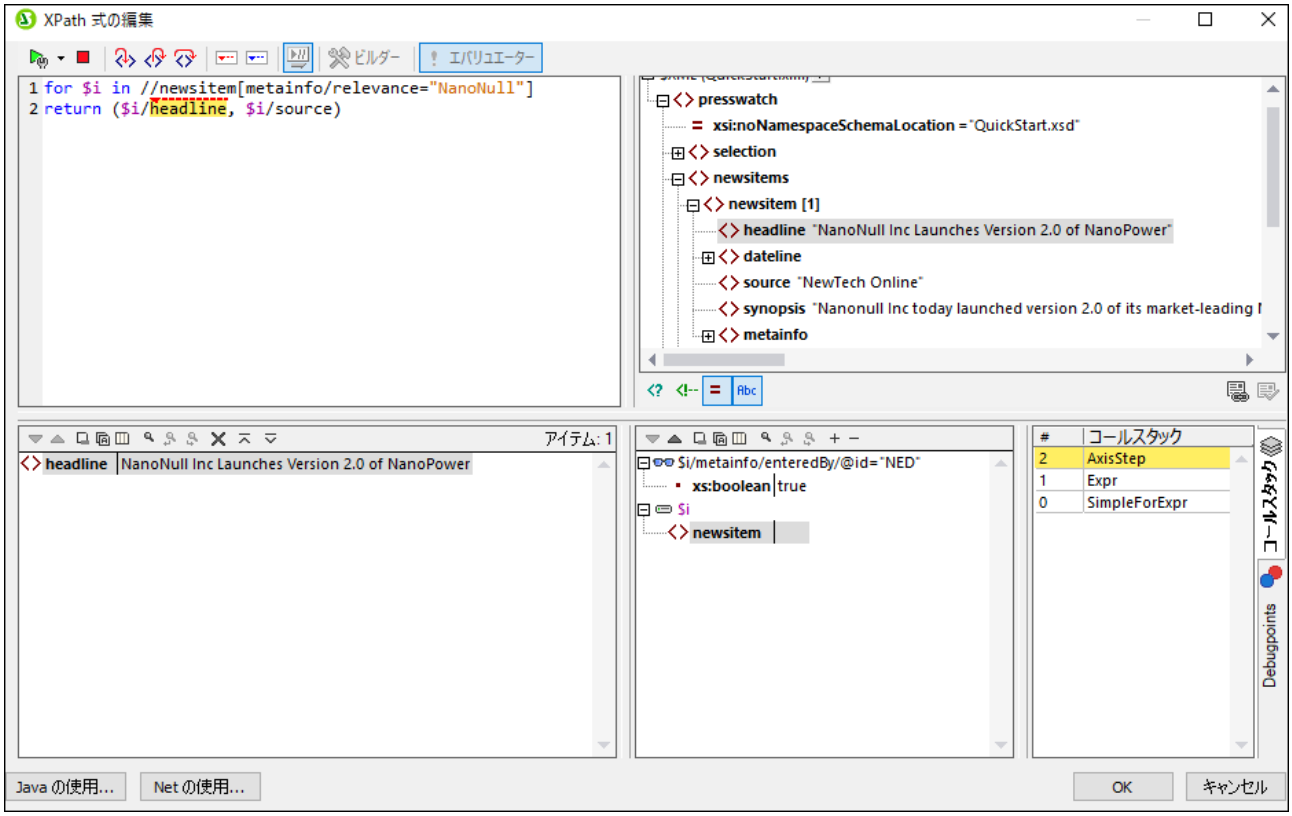


メモ　ブレークポイントとトレースポイントが式の一部に設定されている場合、赤色と青色の交互の点線によりオーバーラインが作成されます。

下の[デバッグポイント](#) も参照してください。

変数、ウォッチ式、コールスタック

変数とウォッチ式は変数とウォッチ式ペイン（のスクリーンショットの中央のペイン）内に表示されます。



変数
式内で宣言されており、現在の評価ステップ内のスコープに存在する編集は相対する現在の値と共に表示されています。例えば、上のスクリーンショットでは、`headline` 上のブレークポイントで処理が一時停止されています。`$i` 変数はこの評価ステップではスコープ内です。ですから、上のスクリーンショット内で `$i` が最初のノードである現在の値と共に表示されています。

ウォッチ式
ウォッチ式は評価の開始前、または評価の一時停止中に挿入することのできる式です。以下の目的のために使用することができます:

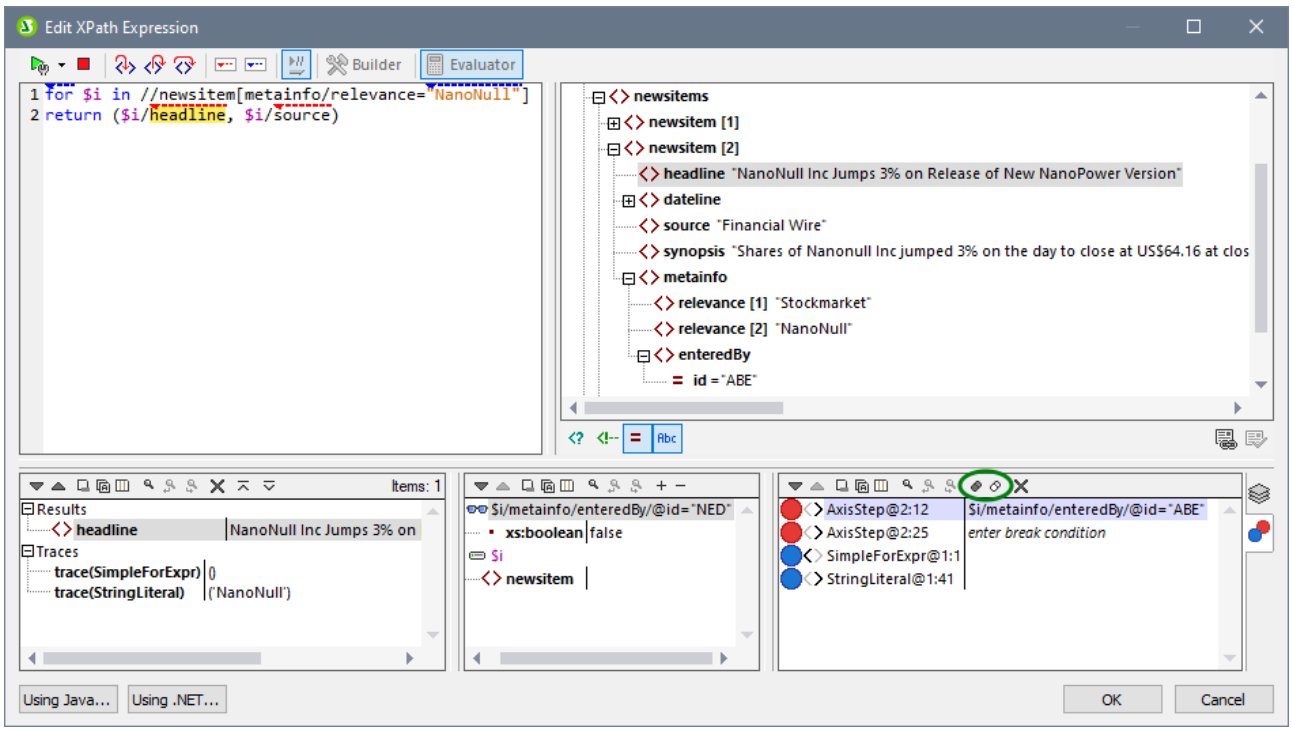
- 特定の条件をテストする場合。上のスクリーンショット内のサンプルでは、ウォッチ式 `$i/metainfo/enteredBy/@id="NED"` が `NED` の ID を持つチームメンバーによりこの新規のアイテムが入力されているかをテストするために使用されています。結果 `true` は最初のニュースアイテムがこの条件が満たしているかを示しています。
- 特定のコンテキスト内のデータを検索する場合。例えば、`Company` 要素コンテキストでは、ターゲットドキュメント内で企業のカスタマーコードを検索するウォッチ式 `@id` を入力することができます。
- 追加データを生成する場合。例えば、新規のアイテムの総数を示す適切な文字列が生成されます。

ウォッチ式を挿入するにはペインのツールバー内の*(上のスクリーンショット内で赤枠で囲まれている)*「**ウォッチエントリの追加**」をクリックし、式を挿入し、終了すると「`Enter`」をクリックしてください。ウォッチ式を削除するには、ツールバー内で「**選択されたウォッチエントリの削除**」をクリックしてください。デバッグ中に式が(例えば、変数の1つが範囲以外など)何らかの理由で正確に評価されない場合、ウォッチ式は赤色で表示されます。

コールスタック
コールスタックとデバッグポイントペインの*コールスタックタブ*はデバッグ内のそのポイントまでのプロセッサーの呼び出しを表示します*(上のスクリーンショット内の右下のペイン)*。現在のプロセッサーの呼び出しは黄色でハイライトされています。現在の評価ステップに直接導いた呼び出しのみが表示されることに注意してください。

デバッグポイント

コールスタックとデバッグポイントペインのデバッグポイントタブは式内で設定された(赤い丸印で表示される)ブレークポイントと(青い丸印で表示される)トレースポイントを表示しています(*下のスクリーンショット内の右下のペイン*)。各ダイアログポイントはラインと文字番号と共にリストされています。例えば、`AxisStep@2:12` は式ペイン内の式のライン 2 と文字 12 にデバッグポイントが存在することを意味します。



以下の機能に注意してください:

- ブレークポイントのためにデバッグポイントペイン内の *ブレークポイントの条件* をダブルクリックして**ブレークの条件** を入力し、条件のために式を入力し **Enter** を押しますます。条件が `true` と評価された場合のみそのブレークポイントは有効化されます。例えば、上のスクリーンショットではブレーク条件 `$i/metainfo/enteredBy/@id="ABE"` はid `ABE` を持つチームメンバーにより入力されたニュースアイテムのそれぞれのヘッドライン上のブレークポイントを有効化します。スクリーンショットはこのブレークポイントで一時中止されている評価を示しています(子のブレークポイントでのウォッチ式が `false` を返すことに注意してください。)
- 対応するツールバーボタンをクリックして全てのデバッグポイントを有効化/無効化することができます: **全てのデバッグポイントの有効化** と **全てのデバッグポイントの無効化** デバッグポイント(*上のスクリーンショット内で緑で囲まれているボタン*)。デバッグポイントが無効化されていると、有効化されるまで全ての評価のためにデバッグポイントは無効化されています。
- 対応するコンテキストメニューのために個別のブレークポイントを有効化/無効化することができます。

ペイン内のツールバーコマンド

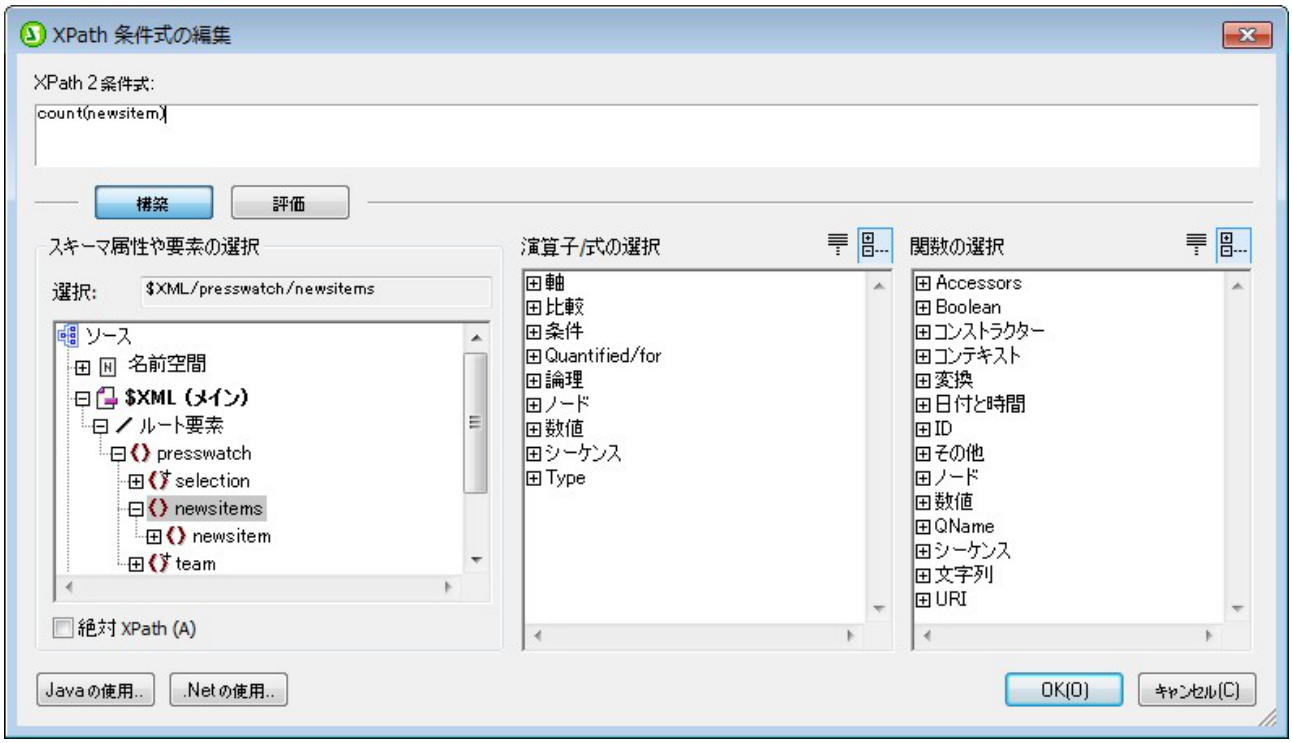
デバッグモード内の式の編集ダイアログのペインにはナビゲート、検索、およびコピーの機能を提供するボタンが含まれています(*上のスクリーンショット*)。これらのボタンは左側から下のテーブル内で説明されています。相対するコマンドもリストされるアイテムのコンテキストメニュー内で使用することができます。

アイコン	動作
次、前	結果リストからそれぞれ次のおよび前のアイテムを選択します。

クリップボードに選択されたテキストラインをコピーする	クリップボードに選択された結果アイテムの値カラムをコピーします。全てのカラムをコピーするには、全てのカラムを含みコピーするコマンドに切り替えてください（*下を参照*）。
全てのメッセージをクリップボードにコピーする	空の値を含む値カラムのすべての結果アイテムをクリップボードにコピーします。各アイテムは異なるラインにコピーされます。
全てのカラムを含みコピーする	(i) 全てのカラム、または(ii) 値カラムのコピー間を切り替えます。カラムの区切りは単一のスペースです。
検索する	結果リスト内で特別文字を含む文字列を検索する検索ダイアログを開きます。
前を検索する	検索ダイアログ内に最後に入力された用語の前の発生を検索します。
次を検索する	検索ダイアログ内に最後に入力された用語の次の発生を検索します。
クリア	結果のリストをクリアする
複数のラインの結果を単一のラインに折りたたむ	結果アイテムの値カラムが複数ラインのテキストを含む場合、（新しいラインまたは文字を含む場合）、複数ラインと単一ラインの表示を切り替えることができます。
完全な結果を表示する	ノードの値としてノードのコンテンツ全体を表示します。

16.2.3 式ビルダー

XPath 条件式の編集ダイアログ内にある [構築] ボタンがクリックされると、XPath 条件式を構築するための入力ヘルパーペインが表示されます。これら入力ヘルパー内にあるエントリーをダブルクリックすることで、テキスト ボックスのカーソル位置にそのアイテムが挿入されます。



入力ヘルパーペインには以下の3 種類があります:

- XPath 式内に要素と属性を挿入するためのスキーマツリー。*相対的な XPath* チェックボックスがチェックされていると(作成されているXPath 式内のデザイン内のノードである)選択されているノードへのロケーション パスはコンテキスト ノードに対して相対的に挿入されます。コンテキスト ノードはスキーマツリーペインの下に表示されています。絶対 XPath 式はドキュメント ルートから始まり、*Relative相対的な XPath* チェックボックスがチェックされていない場合選択されたノードのために使用されます。
- 演算子と式のための入力ヘルパーペイン。以下が含まれます: (i) 軸 (`ancestor::`, `parent::`, など) (ii) 演算子 (例えば、`eq` と`div`) および(iii) 式 (`for # in # return #`, など)。ペインのアイテムはアルファベット順、または関数カテゴリ別にグループ分けすることができます。ペインのタイトル バー内のドロップダウンメニューから *階層的* または *フラット* を選択してオプションを選択します。
- アルファベット順、または関数別にグループ分けされたアクティブな XPath バージョンの関数を持つ入力ヘルパー。ペインのタイトル バー内のドロップダウンメニューから *階層的* または *フラット* を選択してオプションを選択します。 The *Names/Types* オプションにより関数の引数を選択することができ、名前、またはデータ型として表示することを選択できます。

ビルダーの機能

- 両方のペイン内のアイテムに関する詳細を表示するには、マウスをアイテムの上にかざします。
- 各関数は、自身の(引数、引数のデータ型、関数の出力のデータ型などである)署名と共にリストされています。
- 署名は、名前、または、関数の引数と出力のデータ型を利用してリストされています。ペインのタイトル バーのドロップダウンメニューから*名前* または *型* を選択します。
- ペイン(演算子、条件式、または、関数) 内のアイテムをダブルクリックして、式内のカーソルの箇所にアイテムを挿入します。関数はプレースホルダ(`#` シンボル)により示されている引数を使用して挿入されます。
- (i) XPath 条件式の編集フィールド内でテキストが選択されており、(ii) プレースホルダを含む関数の条件式が挿入のためにダブルクリックされている場合、選択されていたテキストがプレースホルダの代わりに挿入されます。

条件式内に関数を挿入後、関数名の上にマウスをポイントすると、関数の署名と関数の説明をしたテキストが表示されます。同じ名前を持つ関数のために異なる署名が存在する場合、これらはオーバーロード要素として下の部分に示されます。カーソルを関数のかっこ内にかざし、Ctrl+Shift+ Space キー を押すと、その関数名の多種のオーバーロードの署名を確認することができます。

XPath 式の作成

XPath 式 の編集ダイアログは次の方法でXPath 式の作成を手助けします。

- *コンテキスト ノードとスキーマツリー*
  *ソース*ペイン内の*選択* テキスト ボックスはすぐにコンテキスト ノードを表示します。式はこのコンテキスト ノード内の場所に挿入され、コンテキストとしてこのノードと共に評価されます。

- *スキーマツリーからのノードの挿入*
  ソース ペイン内で、スキーマ全体が表示されます。スキーマツリー内のノードをダブルクリックし、XPath 式に挿入します。If the *Relative XPath* チェックボックスがチェックされていると、選択されているノードはコンテキスト ノードに相対的なロケーション パス式と共に挿入されます。

- *名前空間の情報*
  スキーマ属性や要素ペインにあるスキーマ ツリーには名前空間アイテムが含まれます。このアイテムを展開すると、スタイルシート内部で宣言されたすべての名前空間が表示されます。この情報は XPath 条件式にて使用する名前空間の接頭辞を確認するために使用することができます。

- *XPath 軸、演算子、式の挿入*
  演算子/式の選択ペインでは、SPS で選択されている XPath の軸 (ancester:: や parent:: など)、演算子 (eq や div など)、または式 (for # in # return # など) が表示されます。表示方式は、アルファベット順と、機能によりアイテムがグループ分けされた階層構造から選択することができます。軸、演算子、または 式を XPath 条件式に挿入するには、目的のアイテムをダブルクリックしてください。

- *XPath 関数の挿入*

関数の選択 ペインは、XPath 関数をアルファベット順に、または、機能別のグループ別にリストしています。（ペインの上のアイコンをクリックして、2つの異なる表示方法を切り替えてください）。各 関数は自身の署名と共にリストされています。関数に1つ以上の署名が存在する場合、その関数は署名の数だけリストされます。署名内の引数は、カンマで区切られており、各引数には、発生各ごとに発生インジケーターを有します（? はゼロのシーケンス、または、指定された型の1つのアイテムを示し、* は、ゼロのシーケンス、または、指定された型の1つ以上のアイテムを示します）。引数は、名前、またはデータ型ごとに表示されます。ペインの上のThe arguments can be displayed as names or as datatypes; select *Names* or *Types* in the title bar of the pane.各関数は、その関数の戻り型を指定します。例: `=> date ?` は、期待されるデータ型は、無しのシーケンス、または、`date` アイテムであることを示しています。関数の上にマウスをかざすと、関数の短い説明が表示されます。

- Java ならびに .NET 拡張を XPath 条件式に使用することで、これらプログラミング言語に搭載された関数へアクセスすることができます。ダイアログ下部にある [**Java**] ならびに [.NET] ボタンをクリックすると、XPath 条件式内部で Java ならびに .NET 拡張関数を使用するための方法が示されます。この機能の詳細については、このドキュメンテーションの 拡張関数 を参照ください。

直接のテキストエントリ中のインテリジェントな編集

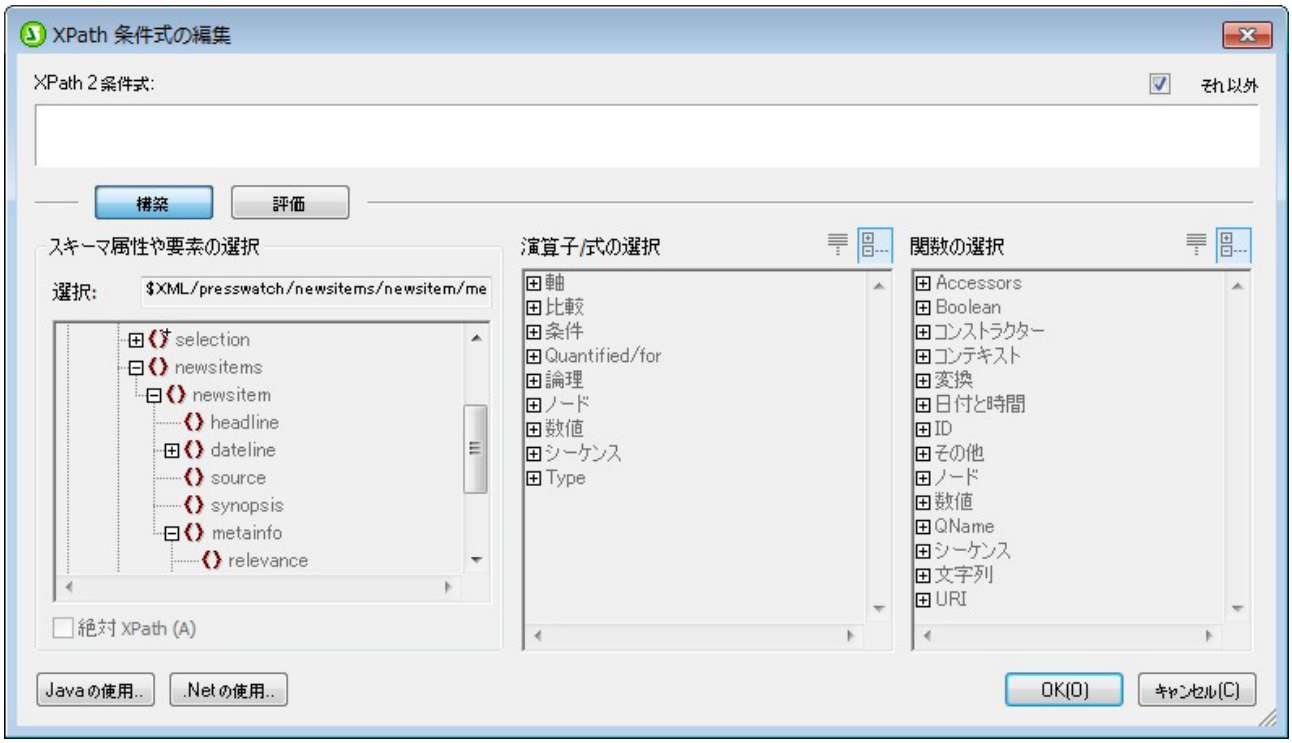
条件式テキスト ボックスへ条件式を直接入力する場合、その時点で利用することのできるオプションがポップアップで表示されます（以下のスクリーンショットを参照）。

これらは要素 XPath 関数とXPath 軸を含みます。「**上**」と「**下**」キーを使用してオプションを上下に移動します。オプションを選択して、式に挿入する場合 **Enter** を押します。

それ以外の場合チェックボックス

条件テンプレートの条件分岐にそれ以外の条件を追加することができる場合、[**それ以外**] 切り替えが表示されます（以下のスクリーンショットの赤い矢印を参照）。それ以外の分岐が存在しない条件テンプレートにチェック ボックスは表示されるほか、すでにそれ以外の条件分岐が含まれる条件テンプレートを編集する際にも [**それ以外**] チェック ボックスは表示されます。[**それ以外**] チェック ボックスにチェックを入れ、[OK] ボタンをクリックすることで、条件テンプレートにそれ以外の条件分岐が追加されます。[**それ以外**] チェック ボックスにチェックを入れると、構築モードにあるすべての入力ヘルパーが無効になります。これは、それ以外の条件分岐に与えられる条件はなく、XPath 条件式を構築する必要がないためです。

(C) 2015-2021 Altova GmbH



[それ以外] 条件分岐を使用する詳細については、「次を参照してください：条件テンプレート。

16.3 ツールバー

ツールバーでは様々な StyleVision コマンドが用意されています。ツールバーは以下のように構成されます:

- 書式
- テーブル
- Authentic
- リッチエディット
- デザイン要素の挿入
- デザインフィルター
- グローバルリソース
- 標準

各ツールバーのアイコンは、対応するコマンドの簡単な説明とともに、このセクション以下にあるサブセクションに記されます。

ツールバーの位置

ツールバーはフロート状態にして画面上の自由な位置に配置することができるほか、GUI のウィンドウ内部に配置することができます。ツールバーは通常 GUI の上端にあるメニューバー以下に配置されますが、ウィンドウの両端または下部に配置することもできます。

以下の操作により、ツールバー領域にツールバーを配置することができます:

1. ツールバーのハンドル (ツールバーがツールバー領域に配置されている場合) またはそのタイトル バー (ツールバーがフロート状態の場合) をクリックします。
2. ツールバーを目的の場所までドラッグし、ドロップします。目的の場所にツールバーが存在しない場合、ウィンドウの端にツールバーを移動させることで、新たなツールバーが表示されます。

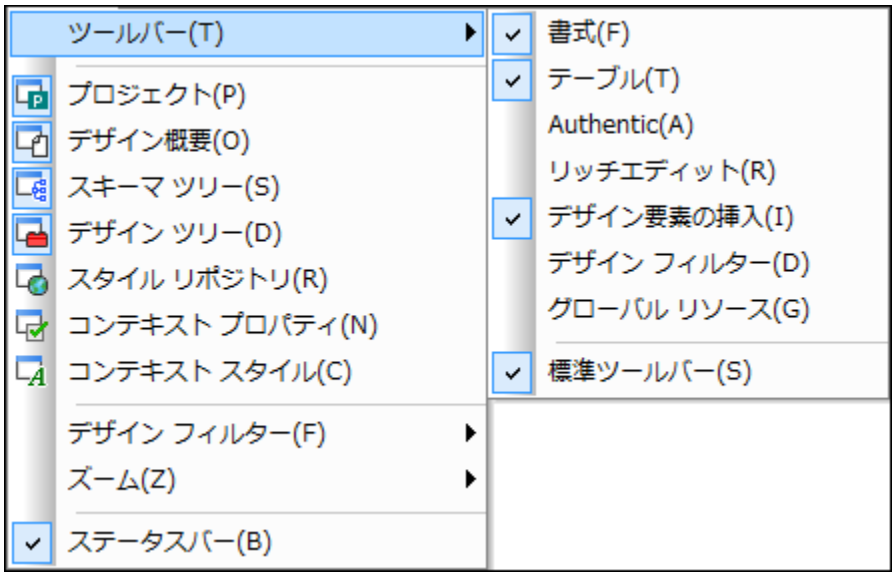
ツールバーをフロート状態にするには、ツールバー領域の外にツールバーをドラッグし、ウィンドウの端以外の画面上の任意の位置でドロップします。

ツールバーを表示/隠す

以下にある 3 つの方法で、個々のツールバーの表示を切り替えることができます:

- [**表示 | ツールバー**] メニューにて、ツールバーから対応するツールバーを選択または選択解除します。.

(C) 2015-2021 Altova GmbH



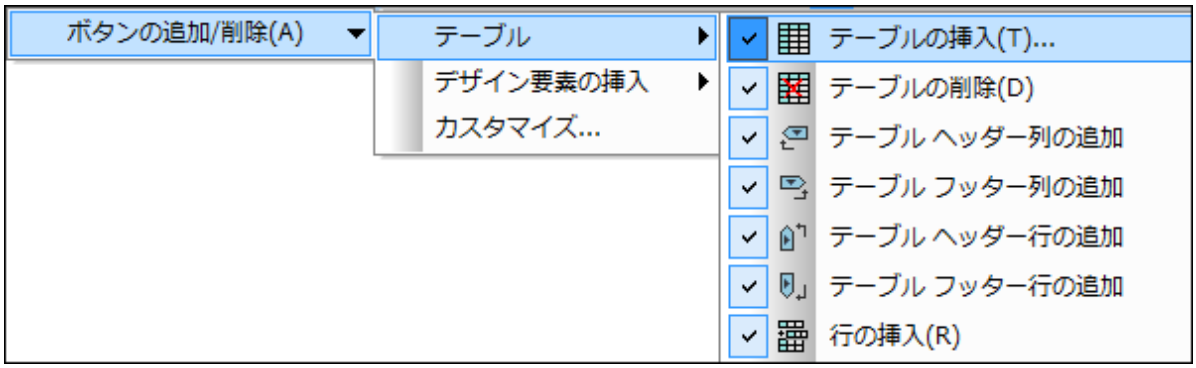
- ツールバーの領域を右クリックして、コンテキスト メニューから個々のツールバーの表示を切り替えます（以下のスクリーンショットを参照）。

- (ツール | カスタマイズ)メニューにて表示されるカスタマイズ ダイアログ のツールバータブにて、ツールバーのチェック ボックスをクリックすることで、個々のツールバーの表示を切り替えます。完了した後には[**閉じる**] ボタンによりダイアログを閉じることができます。

ツールバーボタンを追加/削除

ツールバーにある個々のボタンは追加または削除することができます。以下の操作により、ツールバーからボタンを追加または削除することができます:

1. 追加または削除するボタンに関連付けられたツールバーにて、(ツールバーがツールバー領域内に配置されている場合) [**カスタマイズ**] ボタンまたは (ツールバーがフロート状態の場合) [**ツールバーのオプション**] ボタンをクリックします。[**カスタマイズ**] ボタンはツールバーの右端または下端に表示されている矢印のことです。[**ツールバー オプション**] ボタンは、フロート状態のツールバー右側に表示されます。
2. 表示される [**ボタンの追加/削除**] メニュー上にマウス カーソルを移動させます。そのツールバー領域に表示されているツールバーの名前と、[**カスタマイズ**] メニュー アイテムが表示されます (以下のスクリーンショットを参照)。



3. 追加または削除するツールバーボタンを含んでいるツールバーにマウスを移動させます（上のスクリーンショットを参照）。
4. 表示されるメニューにて、追加または削除するツールバー ボタンの名前をクリックします。
5. [カスタマイズ] アイテムをクリックすると、カスタマイズ ダイアログが表示されます。

ボタンのリスト以下にある [**ツールバーをリセット**] アイテムにより、そのツールバーを StyleVision. をインストールした状態に戻すことができます。リセットされると、そのツールバーにあるすべてのボタンが表示されます。

メモ ツールバーに含まれるボタンと、そのツールバーとの関連付けを解除することはできません。上に示されたツールバーのボタンを表示/隠す動作は GUI に対してのみ適用されます。

関連項目

- ユーザーインターフェイス

16.3.1 書式

[**書式**] ツールバーはデザイン ビューで表示され、一般的なインラインならびにブロックのフォーマット プロパティを、SPS のデザインにて選択されたアイテムへ割り当てるためのコマンドが含まれています。

定義済みの HTML フォーマット
ドロップダウン リストから選択された HTML フォーマットがデザイン ビューの選択対象に適用されます。例えば div を選択することで、HTML のブロック (div) 要素がデザイン ビューにて現在選択されている要素の周りに適用されます HTML フォーマットは RTF 出力のために対応する RTF プロパティに変換されます。

テキスト プロパティ
対応するボタンをクリックすることで、太字、イタリック、下線そして取り消し線のインライン テキスト プロパティが、現在デザイン ビューで選択されている対象に対して適用されます。ツールバーボタンからは、スタイル、フォント サイズ、フォントの色と背景色を指定することもできます。

配置
配置プロパティ（左寄せ、中央揃え、右寄せ、両端揃え）を、デザイン ビューの選択対象に直接適用します。

(C) 2015-2021 Altova GmbH

リスト
カーソルの挿入ポイントにリストを挿入することができるほか、SPS の選択対象をリストへ変換することができます。

ハイパーリンク

カーソルの挿入ポイントにハイパーリンクを挿入します。このコマンドの詳細な使用方法については、ハイパーリンクを参照ください。

関連項目

- ツールバー
- 書式とスタイル
- 入力フォーマット

16.3.2 テーブル

[テーブル] ツールバーはデザインビューで表示され、静的ならびに動的なテーブルの構造やフォーマットを指定するコマンドが収められています。テーブルツールバー内にあるこれらのコマンドを以下のスクリーンショットに示します。

行と列の操作
SPS のテーブル（静的ならびに動的）に対して行または列の挿入、追加、削除を行うことができます。行または列は現在のカーソル位置の直前に挿入、または直後に追加されます。カーソルが配置されている行/列を削除することもできます。これらの操作は[**行／列の挿入**]、[**行／列の追加**]、[**行／列の削除**] ボタンにより行われます。[**テーブル ヘッダー行／列の追加**] または[**テーブルフッター行／列の追加**] により、テーブルのヘッダーやフッターを追加することもできます。

セルの操作

現在カーソルが配置されているSPS テーブルセルを、隣接する4 つのセルの1 つに結合することができます。結合操作はHTML のテーブルにおけるspan 操作と似たものです。[**右／左／上／下のセルを結合**] ボタン、がこれらの操作のために使用されます。更に、[**セルを水平方向に分割**] または[**セルを垂直方向に分割**] ボタンにより、カーソルが配置されている．SPS テーブルのセルを、それぞ

れ水平または垂直方向に分割することもできます。SPS テーブルのセル内部にあるコンテンツは、上揃え、中央揃え、下揃えすることができ、[**セルの境界を表示**] ボタンをクリックすることで、セルの境界線を表示または隠すことができます。

テーブルの操作、プロパティ、表示

静的または動的なテーブル内部にカーソルが配置された状態で テーブルの削除 をクリックすることで、そのテーブルが削除されます。テーブルのマークアップ表示は、[**テーブル マークアップの表示**] ボタンにより切り替わります。[**テーブルのプロパティ**] コマンドにより、そのテーブルのプロパティを定義するためのダイアログが表示されます。

関連項目

- ツールバー
- テーブル
- テーブルメニュー

16.3.3 Authentic

Authentic ツールバー には、Authentic View をカスタマイズするため、そして Authentic View にて XML ドキュメントを編集するためのコマンドが含まれています。これらのコマンドを以下のスクリーンショットに示します（ツールバー右の**カスタマイズ** ボタンをクリックすると、カスタマイズメニューのためのAuthentic ツールバーボタンの追加と削除を使用することができます）。

Authentic View ユーザーはこれらすべての機能を利用することができるほか、SPS のデザイン時にも Authentic View ユーザーが利用できる機能を使った SPS のテストを行うことができます。

XML ドキュメントの検証、保存、再ロード
XML ドキュメントを Authentic View にて編集する際に、[**XML の検証**] ボタンを使用することで、作業用 XML ファイルの妥当性をチェックすることができます。編集された内容は [**Authentic の XML データを保存**] ボタンにて作業用 XML ファイルへ保存することができます。XML ドキュメントを再ロードすることで、前回保存したバージョンへ戻すこともできます。

XML データの新たな行を選択して編集
このコマンドは、SPS が XML DB をベースにしている場合にのみ有効になります。コマンドにより、XML の列を Authentic View へロードして編集することができます。詳細については[コマンドの詳細](#) を参照ください。

Authentic View のマークアップタグ
Authentic View では、マークアップ タグの表示をカスタマイズすることができます。マークアップ タグを表示しない ([**マークアップを隠す**])、ノード名を表示する ([**大きなマークアップの表示**])、ノード名を表示しない ([**小さなマークアップの表示**])、または個々のノードに対して異なるオプションを指定する ([**混合マークアップの表示**]) ことができます。

Authentic View 内の動的テーブルを編集
Authentic View では、動的なテーブルに対して行オペレーションを行うことができます。行の挿入、追加、複製、移動、そして削除を、対応するボタン ([**行の挿入**]、[**行の追加**]、[**行の複製**]、[**行を上に移動**]、[**行を下に移動**]、[**行の削除**]) により行うことができます。

XML エンティティの定義
Authentic View にて作業用 XML ファイルを編集中にエンティティを定義することができます。[**XML エンティティの定義**] ボタンをクリックすることで、
XML エンティティの定義ダイアログが表示されます (使用方法については *Authentic | エンティティの定義* を参照ください)。

(C) 2015-2021 Altova GmbH

関連項目

- ツールバー
- Authentic メニュー

16.3.4 リッチエディット

リッチエディット ツールバーには Authentic View にてテキストのスタイルプロパティにてテキストをマークアップするためのコマンドが含まれます。

Authentic View にて、リッチエディットコンポーネントとして作成された要素の中にカーソルが配置されている場合、リッチエディット ツールバー内にあるボタンやコントロールが有効になります（以下のスクリーンショットを参照）。それ以外の場合、コマンドはグレーアウトされます。

リッチエディットにより、フォント、フォントの太さ、フォント スタイル、フォントの装飾、フォントの大きさ、色、背景色、そしてテキストの配置を Authentic View ユーザーが指定できるようになります。スタイルを適用したいテキストを Authentic View で選択し、リッチエディット ツールバーにあるボタンやコントロールから、適用するスタイルを指定してください。スタイルが適用されたテキストは スタイル要素のタグにより囲まれることになります。

関連項目

- ツールバー
- リッチエディット
- Authentic メニュー

16.3.5 デザイン要素の挿入

デザイン要素の挿入 ツールバー ツールバーにはSPS デザイン内部へデザイン要素を挿入するコマンドや、関連するコマンドに対応したアイコンが含まれています。ツールバーアイコンから挿入することのできるデザイン要素を以下のスクリーンショットに示します。ツールバーには3 種類のアイテムが収められています:

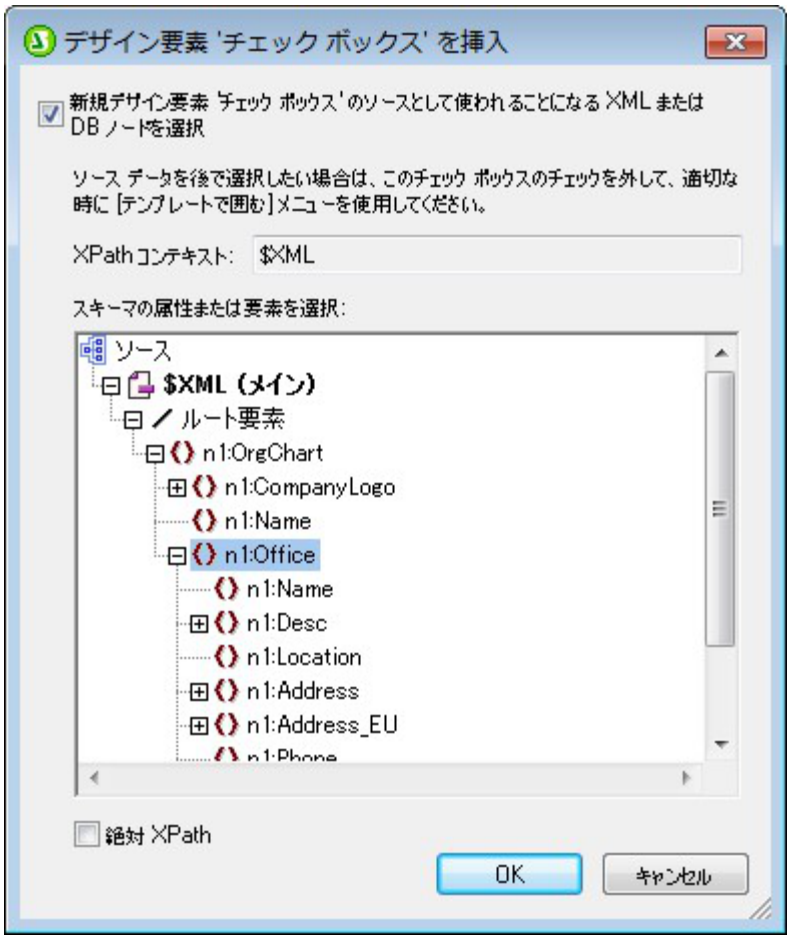
1. デザイン要素, :コンテキスト ノードに影響を受けます（ツールバーにある殆どの要素です）
2. レイアウト要素 :ノード コンテキストとは独立して使用できます
3. グリッドに関連する操作 :デザインの補助に使用されます

デザイン要素

デザイン要素は[**挿入**] メニューからも利用することができる、コンテキスト ノードに影響を受けるノードです。以下の操作により、ツールバーアイコンからデザイン要素を挿入することができます:

1. 挿入する要素のツールバー アイコンを選択します。
2. 要素が挿入されるデザイン上の場所をクリックします。選択された要素のためのデザイン要素の挿入ダイアログが表示されます (*下のスクリーンショット*)。ダイアログにはコンテキスト ノードがハイライトされた状態のスキーマツリーが表示されます。コンテキスト ノードとは、デザイン要素を挿入するために指定したカーソル位置の外側にあるノードのことです。

(C) 2015-2021 Altova GmbH



3. 現在選択されているコンテキスト ノード内部にデザイン要素を挿入するには、[OK] をクリックしてください。他のコンテキスト ノードを選択するには、スキーマ ツリーにて目的のコンテキスト ノードを選択し、[OK] をクリックしてください。
4. 自動計算といったデザイン要素の場合、自動計算を定義するといった操作を更に行う必要があります。そうでない場合 (例えばユーザー定義テンプレートの挿入など)、デザイン要素の挿入ダイアログがスキップされます。この場合、XPath 式 の編集ダイアログ といった、別のダイアログが表示されます。必要な情報を入力し、ダイアログの [OK] ボタンをクリックしてください。

挿入されたデザインの種類により、ステップ 3 または ステップ 4 でデザイン要素が挿入されます。

レイアウト要素

デザイン要素を挿入ツールバーには、(i) レイアウト コンテナー、(ii) レイアウト ボックス、そして (iii) ラインを挿入するための、3 つのレイアウト要素コマンドが収められています。レイアウト ボックスとラインはレイアウト コンテナー内部だけに挿入することができることに注意してください。

レイアウト コンテナーを挿入するには、[**レイアウト コンテナーの挿入**] アイコンを選択し、レイアウト コンテナーを挿入するデザイン上の場所をクリックしてください。レイアウト コンテナーの大きさについて尋ねられ、選択によりレイアウト コンテナーが挿入されます。レイアウト ボックスを挿入するには、[**レイアウト ボックスの挿入**] アイコンをクリックし、レイアウト ボックスを挿入するレイアウト コンテナー上の場所をクリックしてください。レイアウト ボックスが挿入されます。レイアウト ボックス内部をクリックすると、キーボードによる文字の入力をはじめることができます。ラインを挿入するには、[**ラインの挿入**] アイコンをクリックして、レイアウト コンテナー内にあるラインの開始点までカーソルを移動してください。クリックによりラインの開始点がセットされ、目的の終了点までカーソルをドラッグしてください。終了点でカーソルをドロップすることで、指定された開始点から終了点までのラインが挿入されます。

レイアウト コンテナーやレイアウト ボックスの大きさを変更するには、レイアウト コンテナーまたはレイアウト ボックスの右端または下端までカーソルを移動させ、境界線をドラッグしてください。レイアウト ボックスを移動するには、レイアウト ボックスの上端または左端までカーソルを移動させ、カーソルの表示が十字に変わったところでドラッグによりレイアウト ボックスを新たな場所へ移動してください。

グリッドに関連する操作

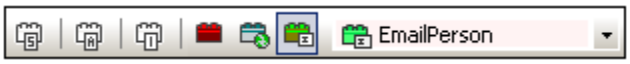
[**グリッドの表示**] コマンドにより、グリッドの表示/非表示を切り替えることができます。[**グリッドへのスナップ**] コマンドが有効になっている場合、レイアウト コンテナー内部で作成されたレイアウト ボックスやラインが、グリッド線やグリッド線の交差点に対してスナップされます。グリッドのプロパティはオプションダイアログ（[**ツール | オプション**]）のデザイン タブにてセットすることができます。

関連項目

- ツールバー
- 挿入メニュー
- レイアウトコンテナー

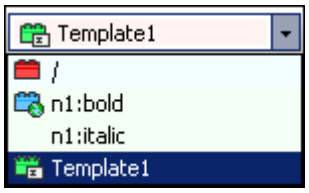
16.3.6 デザインフィルター

デザインフィルター ツールバー（*下のスクリーンショット*）には、デザインに表示されるテンプレートのフィルタリングを行うためのコマンドが収められています。ツールバーにある各アイコンについて以下で説明します。.



アイコン	コマンド	説明
	選択されたテンプレートだけを表示	選択されたテンプレートだけを表示します。テンプレート内にカーソルを配置してこのアイコンをクリックすることで、そのテンプレートだけが表示されます。
	全種類のテンプレートを表示	SPS 内にあるすべてのテンプレート（メイン、グローバル、名前付き、レイアウト）が表示されます。
	インポートされたテンプレートを表示	インポートされたテンプレートの表示を有効/無効にします。
	メイン テンプレートを表示/隠す	メイン テンプレートの表示を有効/無効にします。
	グローバル テンプレートを表示/隠す	グローバルテンプレートの表示を有効/無効にします。
	デザイン フラグメントを表示/隠す	デザイン フラグメントの表示を有効/無効にします。

デザイン フィルターコンボ ボックスにはSPS にある全テンプレートのリストが表示されます（以下のスクリーンショットを参照）。



コンボボックスからテンプレートを選択することで、そのテンプレートがデザインにて選択されます。コンボボックスによりデザイン内にある目的のテンプレートへ素早くアクセスすることができ、複数のテンプレートがデザイン内部に存在する場合、その中でもテンプレートの一部が表示されていないような状況で利用することができます。

関連項目

- ツールバー
- デザインツリー

16.3.7　グローバルリソース

グローバルリソース ツールバー では、(i) アプリケーションに対してアクティブな構成を選択し、(ii) Altova グローバルリソース ダイアログへアクセスすることができます（以下のスクリーンショットを参照）。

コンボボックスのドロップダウンリストに示されたオプションからアクティブな構成を選択してください。グローバルリソースの管理アイコンをクリックすることで、Altova グローバルリソース ダイアログが表示されます。

関連項目

- Altova グローバルリソース

16.3.8　標準

標準 ツールバー には重要なファイルや編集機能に関するコマンドが含まれています。簡単な説明とともにこれらのアイコンを以下に示します。コマンド名をクリックすることで、リファレンス セクションにあるコマンドの詳細な説明を確認することができます。

アイコン	コマンド	ショートカット	説明
	XML スキーマ/DTD/XML から新規作成	**Ctrl+N**	スキーマをベースにした SPS ドキュメントを新たに作成します。ドロップダウンをクリックすることで、DB や HTML ドキュメントをベースにした SPS や、空の SPS を作成することができます。

アイコン	コマンド	ショートカット	説明
	開く	Ctrl+O	既存のSPS ドキュメントを開きます。
	再ロード		最後に保存された状態のSPS を再ロードします。
	デザインを保存	Ctrl+S	アクティブな SPS ドキュメントを保存します。
	すべて保存	Ctrl+Shift+S	開かれているすべてのSPS ドキュメントを保存します。
	印刷	Ctrl+P	作業用 XML ファイルのAuthentic View を印刷します。
	印刷プレビュー		作業用 XML ファイルのAuthentic View の印刷プレビューを表示します。
	切り取り	Shift+Del	選択対象を切り取り、クリップボードにコピーします。
	コピー	Ctrl+C	選択対象をクリップボードにコピーします。
	貼り付け	Ctrl+P	クリップボードの内容をカーソル位置へ貼り付けます。
	削除	Del	選択対象を削除します。
	元に戻す	Alt+ Backspace	編集された変更を元に戻します。元に戻せる回数に制限はありません。
	やり直し	Ctrl+Y	元に戻した操作をやり直します。
	検索	Ctrl+F	Authentic View や出力ビューにあるテキストの検索を行います。
	次を検索	F3	検索されたテキストが次に出現する箇所を検索します。
	ズーム		デザイン ビューのズーム倍率をセットします。
	小さなデザイン マークアップの表示		小さなマークアップ タグが表示されるようにセットします。
	大きなデザイン マークアップの表示		大きなマークアップ タグが表示されるようにセットします。
	XSLT 1.0		XSLT 1.0 がスタイルシート言語として設定されます。
	XSLT 2.0		XSLT 2.0 がスタイルシート言語として設定されます。
	XSLT 3.0		XSLT 3.0 がスタイルシート言語として設定されます。
	スペルチェック		SPS ドキュメントにてスペルチェックを行います。
	DB クエリの切り替え		データベースクエリダイアログを表示/非表示します。

(C) 2015-2021 Altova GmbH

関連項目

- ツールバー

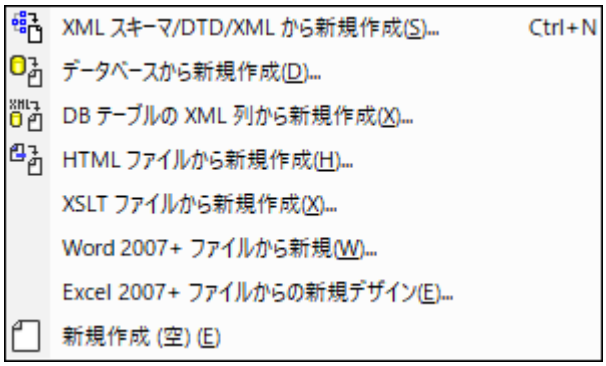
16.4 ファイル メニュー

ファイル メニューには、SPS や関連するファイルを操作するためのコマンドが収められています。以下のコマンドを利用することができます:

- 新規作成 様々なソースから SPS を新たに作成します。
- 開く、再ロード、閉じる、すべて閉じる ファイルを開く、アクティブなファイルを閉じる、または再ロードします。
- デザインの保存、名前をつけて保存、すべて保存 アクティブな SPS や、開かれているすべての SPS ファイルを保存するためのコマンドです。
- MobileTogether デザインファイルとしてエクスポートする を使用して、アクティブな SPS ファイルから MobileTogether デザインを生成することができます。
- Authentic の XML データを保存、名前を付けて保存 Authentic View にて有効になるコマンドで、作業用 XML ファイル に対してなされた変更を保存します。
- 生成されたファイルの保存 SPS により生成することのできる出力ファイルを保存します。
- Web デザイン ASPX アプリケーションを動作させるのに必要なすべてのファイルを、指定されたフォルダーに生成します。
- プロパティ 出力ドキュメントのエンコーディング、ブラウザーの CSS 互換モード、Authentic View における相対イメージ パスの解決方法、RTF (Enterprise ならびに Professional エディションのみ) や Word 2007+ (Enterprise エディションのみ) 出力にてイメージを埋め込むのか、またはリンクするのかの指定を行います。
- 印刷プレビュー、印刷 Authentic View と出力ビューにて有効になります。プレビューに表示されている内容を印刷するためのコマンドです。
- 最後に開かれたファイル、終了 それぞれ最後に開いたファイルを開く、プログラムを終了するためのコマンドです。

16.4.1 新規作成

[**新規作成**] コマンド上へマウス カーソルを配置すると、SPS ドキュメントを新たに作成するためのサブメニューが表示されます (以下のスクリーンショットを参照):



- XML スキーマや DTD、XML ファイルから生成された XML スキーマをベースにした SPS ファイルを新たに作成します ([**XML スキーマ/DTD/XML から新規作成**])。選択されたスキーマが デザイン概要サイドバーへ追加され、(スキーマツリー サイドバーにある) スキーマ ツリーにスキーマの構造が表示されます。デザインビュー では、空のテンプレートとともにSPSが作成されます。新規 SPS はURL やグローバルリソースを介したファイルからも作成することができます (下記参照)。
- 選択された DB から生成された XML スキーマをベースにした SPS ファイルを新たに作成します ([**データベースから新規作成**]、または [**DB テーブルの XML カラムから新規作成**])。データベースへの接続方法については「 データベースへの接続とSPS のセットアップ を参照ください。空のメイン テンプレートとともに デザインビュー に SPS が作成されます。

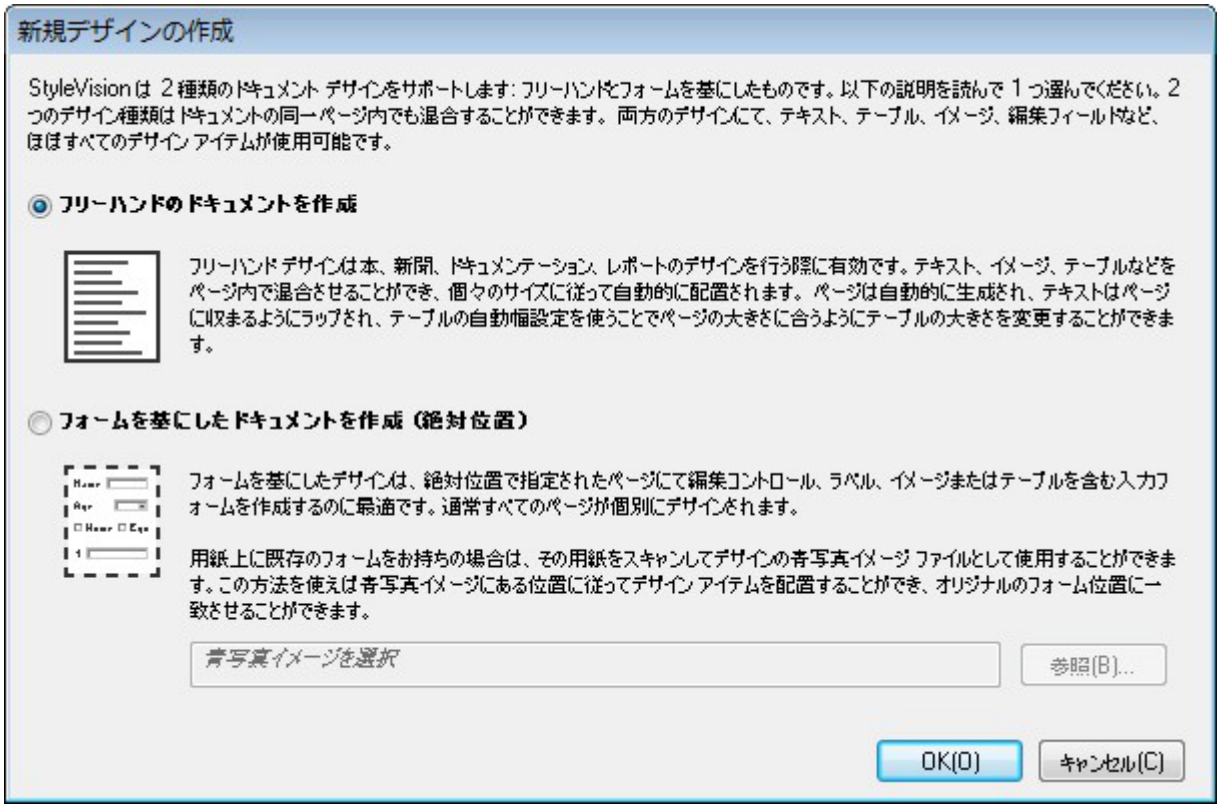
- HTML ファイルからノードごとに作成したユーザー定義のスキーマをベースに、新規 SPS を作成します（[**HTML ファイルから新規作成**]）。ユーザー定義のスキーマがデザイン概要サイドバーとスキーマツリー サイドバーへ追加されます。単一のドキュメント要素（ルート要素）を持ったツリーがスキーマツリーに表示され、HTML ファイルがデザインビューに表示されます。
- XSLT-for-HTML または XSLT-for-FO ファイル、FO ファイルから SPS を作成することができます。XSLT のテンプレート構造ならびにスタイルが SPS にて作成されます。SPS コンポーネントを修正し、コンテンツやフォーマットを SPS に追加することができます。詳細については「 XSLT からの新規作成 」を参照ください。
- デザインの静的テキストとして MS Word ドキュメントのコンテンツを含む新規 SPS。
- デザインの静的テキストとして MS Excel ドキュメントのコンテンツを含む新規 SPS。
- 空の SPS を新たに作成します（[**新規作成（空）**]）。デザイン概要サイドバーやスキーマ ツリーにスキーマは追加されず、デザインビューに空のメイン テンプレートが作成されます。

メモ　ファイルや DB のリソースを指定するために グローバルリソース を使用することもできます。

デザインの種類を選択する

（XSD とXML）といったソースファイルを選択すると、新規デザインの作成ダイアログが表示されます。

新規デザインを作成ダイアログボックスが表示され、(i) フリーハンドのドキュメントデザイン、または(ii) フォームをベースにしたドキュメントデザイン(コンポーネントが絶対位置で指定されます)から選択を行うように促されます。



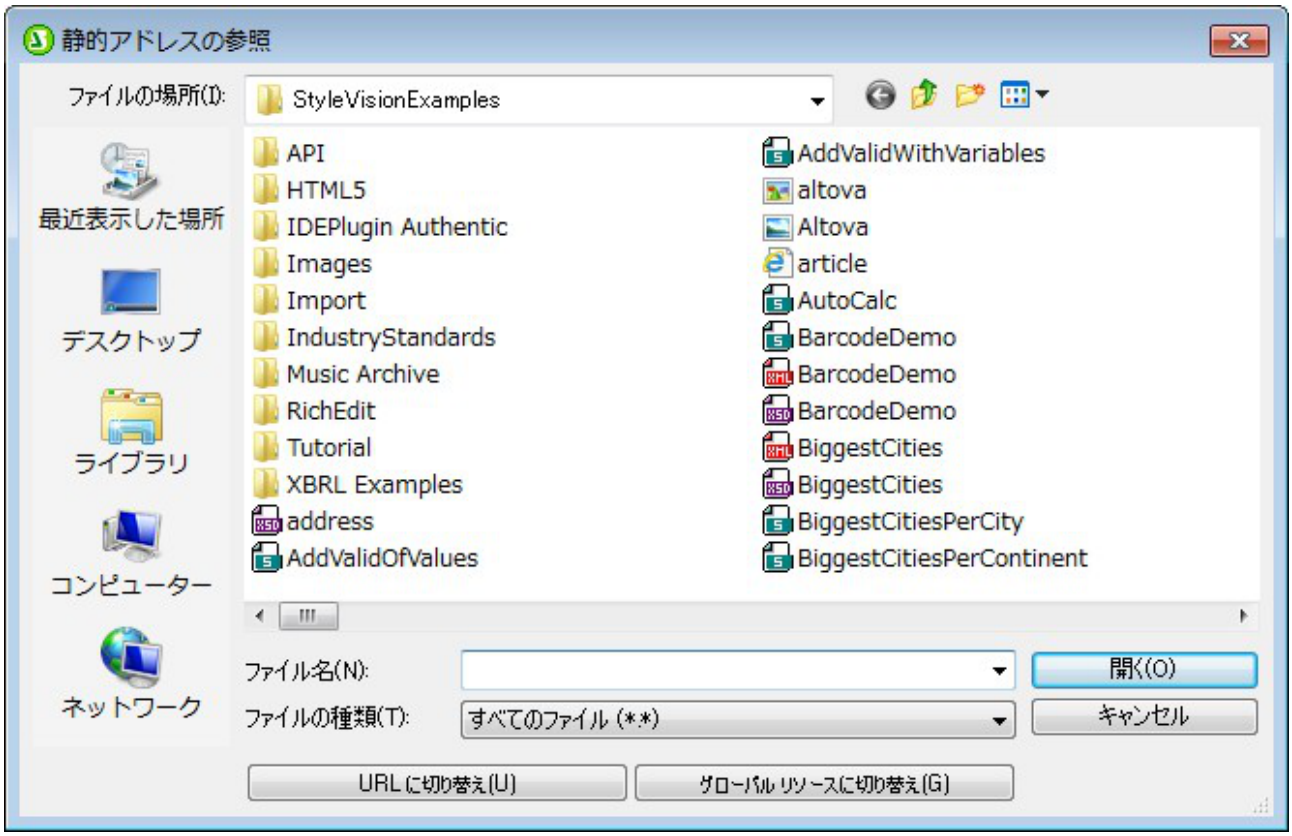
フリーハンドのドキュメントデザインでは、ドキュメントコンテンツが出力メディアオブジェクト（印刷物や画面）にフィットするようにレイアウトされます。ドキュメントコンテンツ内のアイテムは相対的にしか配置することができず、絶対位置を指定することはできません。このオプションは、レポートや記事、書籍などのデザインに適しています。

フォームをベースにしたドキュメントには単一のレイアウトコンテナーが作成され、その中でデザインコンポーネントを絶対位置で指定することができます。レイアウトコンテナーには、絶対位置でレイアウトボックスを配置することができ、各レイアウトボックス内にドキュメントコンテンツを配置することになります。既にあるフォームのデザインを SPS にて再現したい場合、オリジナルのフォームイメージを青写真イメージ として使用する

ことができます。青写真のイメージをレイアウトコンテナーの背景として含めることができ、フォームのデザインに役立てることができます。

URL と グローバルリソースを介してファイルを選択する

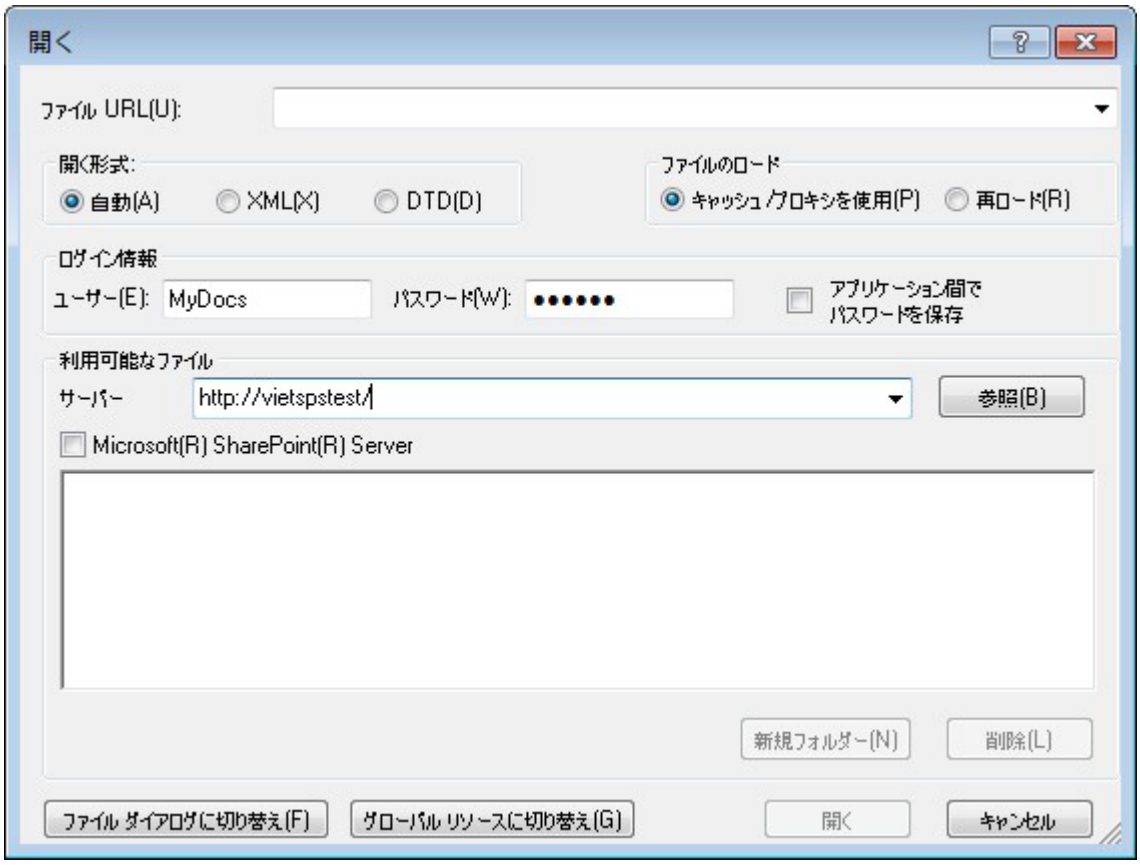
ファイルを開くならびにファイルを保存ダイアログによっては、URL やグローバルリソースにより目的のファイルを選択することができます（以下のスクリーンショットを参照）。[URL にスイッチ]、または [グローバルリソースにスイッチ] を選択することで、別の選択プロセスが表示されます。



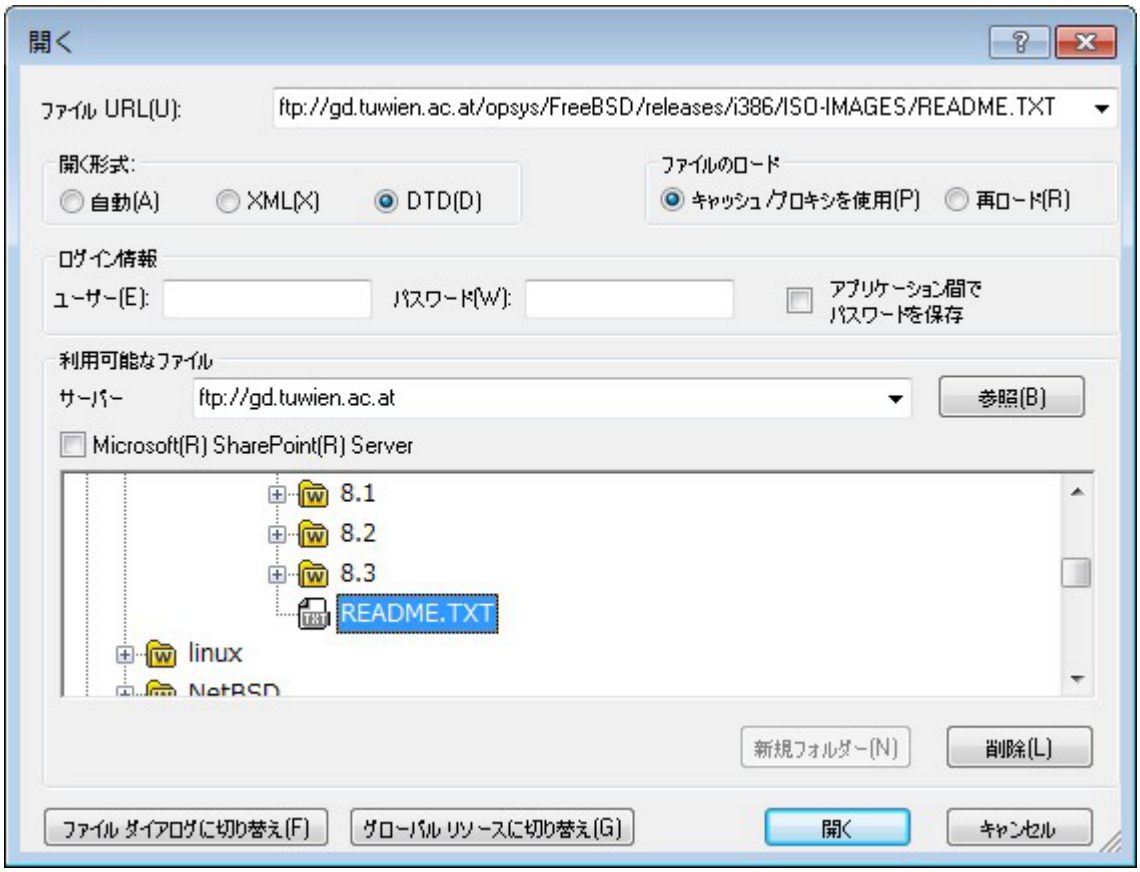
URL によりファイルを選択する

以下の操作により URL を介したファイルの選択を行うことができます:

1. [URL にスイッチ] コマンドをクリックします。URL モードのファイルを開くダイアログが表示されます（以下のスクリーンショットを参照）。



2. サーバーURL フィールドにアクセスするURL を入力します（上のスクリーンショットを参照）。サーバーがMicrosoft(R) SharePoint(S) Server の場合、Microsoft(R) SharePoint(R) Server チェックボックスにチェックを入れてください。この種類のサーバーを利用する場合、下記のMicrosoft(R) SharePoint(R) Server に関するメモを参照ください。
3. サーバーがパスワードを要求する場合、[ユーザー] と[パスワード] のフィールドに必要な情報を入力してください。
4. [参照] をクリックして、サーバーのディレクトリ構造を確認することができます。
5. フォルダーツリーにてロードするファイルを選択します。



ファイルURL が、ファイルURL フィールドに表示されます（上のスクリーンショットを参照）。この段階で、[開く] ボタンが選択可能になります。

6. [開く] ボタンをクリックして、ファイルをロードします。開かれたファイルがメイン ウィンドウに表示されます。

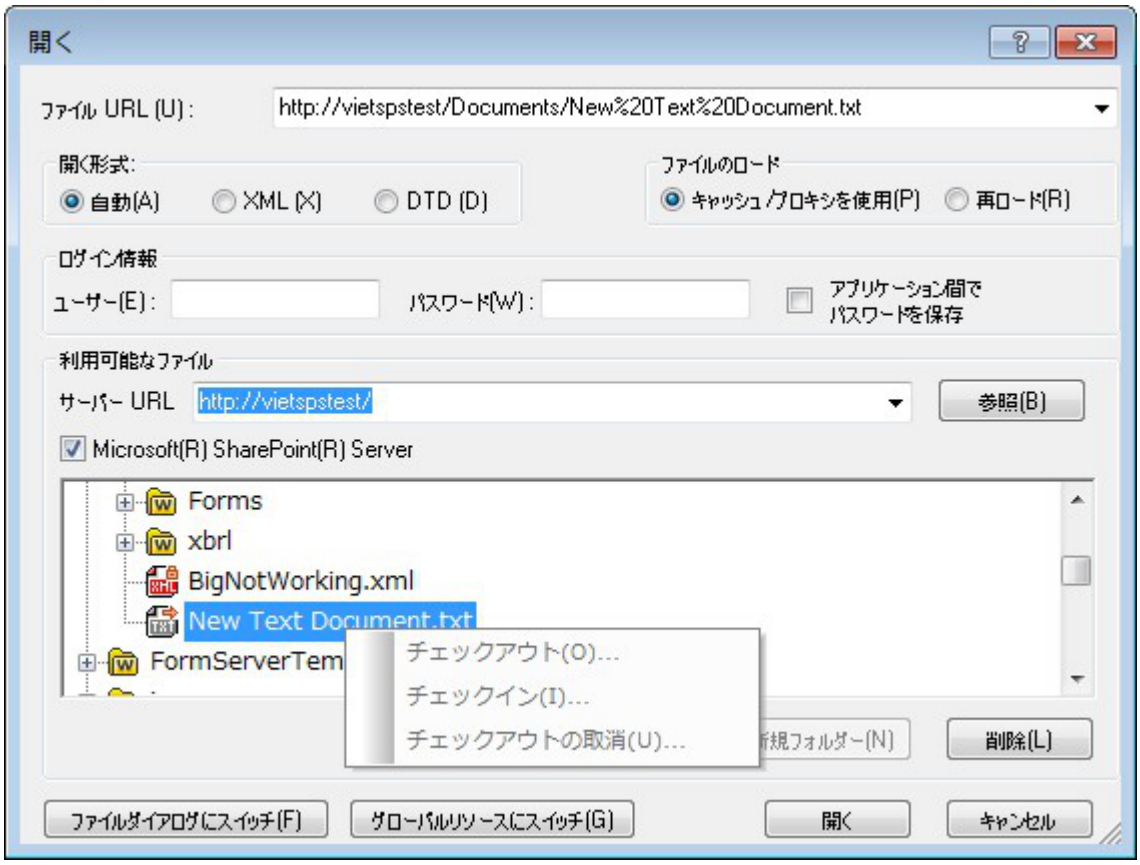
メモ 参照機能はW ebDAV ならびにMicrosoft SharePoint Server をサポートしているサーバーでのみ利用することができます。サポートされているプロトコルはFTP、HTTP、そしてHTTPS になります。

メモ ローカル キャッシュやプロキシ サーバーを使用したファイルのロードを行うこともできます（以前ファイルがロードされた場合、ロードのスピードが劇的に向上します）。その他にも、（データベース システムなどを使用している場合）ファイルを再ロードする必要に迫られることもあります。その場合[再ロード] オプションを選択してください。

Microsoft® SharePoint® Server メモ

Microsoft® SharePoint® Servers に関する以下の点に留意してください：

- 利用可能なファイル ペインに表示されるディレクトリ構造では、ファイルのチェック イン/チェック アウト状態がファイル アイコンのシンボルとして表示されます（以下のスクリーンショットを参照）。



ファイルを右クリックして表示されるコンテキスト メニューには、そのファイルに対して利用可能なコマンドが表示されます。

- ファイルのアイコンを以下に示します:

	チェック イン状態。チェック アウト可能。
	他のユーザーによりチェック アウト状態。チェック アウト不可能。
	ローカルに対してチェック アウト。編集後にチェック イン可能。

- ファイルがチェック アウトされている状態で、Altova アプリケーションにて編集した後は[**ファイル | 保存** (Ctrl+S)] により保存することができます。
- URL を開くダイアログではコンテキスト メニューのほか、アプリケーションのメイン ウィンドウにあるファイル タブを右クリックすることで表示されるコンテキスト メニューから、編集されたファイルをチェック インすることができます。

- 他のユーザーによりファイルがチェック アウトされている場合、チェック アウトを行うことはできません。
- ファイルがローカルでチェック アウトされている場合、コンテキスト メニューから[**チェック アウトの取消**] を選択することで、チェック アウトを取り消すことができます。変更することなしにファイルがサーバーへ戻されます。
- ある Altova 製品でファイルをチェック アウトした場合、他の Altova 製品で同じファイルをチェック アウトすることはできません。ファイルは既にチェック アウトされた状態だと認識されます。Microsoft® SharePoint® Server をサポートしている Altova アプリケーションで行うことができる操作は、[**チェックイン**] と[**チェックアウトの取消**] のみになります。

グローバルリソースを介してファイルを開き、保存する
グローバルリソースを介してファイルを開く、または保存したい場合、[グローバルリソースにスイッチ] をクリックします。グローバルリソースを選択するためのダイアログが表示されます。これらのダイアログについては、グローバルリソースの使用に関するセクションを参照ください。グローバル グローバルリソース のセクションを参照ください。

関連項目

- SPS とソース
- スキーマソース
- スキーマツリー サイドバー

16.4.2 開く、再ロード、閉じる、すべて閉じる

[開く] (Ctrl+O) コマンド により、既存のSPS または PXF ファイルを開くことができます。Windows システムの ダイアログを開くダイアログ が表示され、.sps または .pxf 拡張子のファイルを選択することができます。

[再ロード] コマンドにより、ディスクに保存されている SPS ファイルを開き直します。ファイルが最後に保存されてからなされたすべての変更が失われることになります。作業用 XML ファイルも再ロードされ、外部的に変更された作業用 XML ファイルも更新されます。

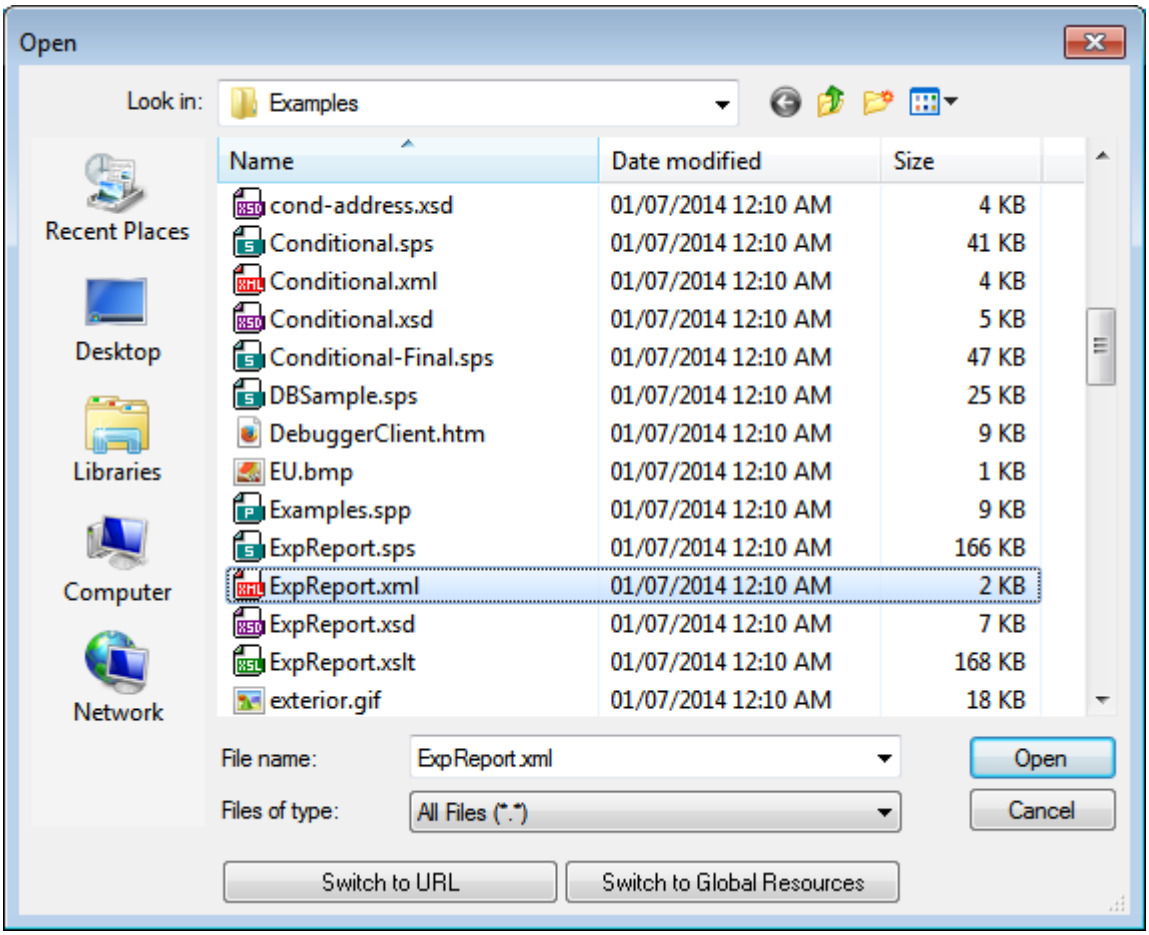
[閉じる] コマンドにより、現在アクティブな SPS ドキュメントが閉じられます。複数のファイルを同時に開くことはできるものの、アクティブなファイルは同時に1 つしかないことに注意してください。 メインウィンドウ の右上にある閉じる ボタンをクリックすることでも、アクティブなドキュメントを閉じることができます。ドキュメント内に保存されていない変更点が存在する場合、これらの変更を保存するか尋ねられます。

[すべて閉じる] コマンドにより、開かれているすべての SPS ドキュメントが閉じられます。開かれているドキュメント内に保存されていない変更点がある場合、これらの変更を保存するか尋ねられます。

▼ URL とグローバルリソースを使用してファイルを選択または保存する

「ファイルを開く」 および「ファイルの保存」 ダイアログでは、URL または グローバルリソースを使用して必要なファイルを選択または ファイルを保存することができます (*下のスクリーンショット参照*)。選択プロセスに移動するために「URL に切り替える」 または「グローバルリソース」 をクリックしてください。

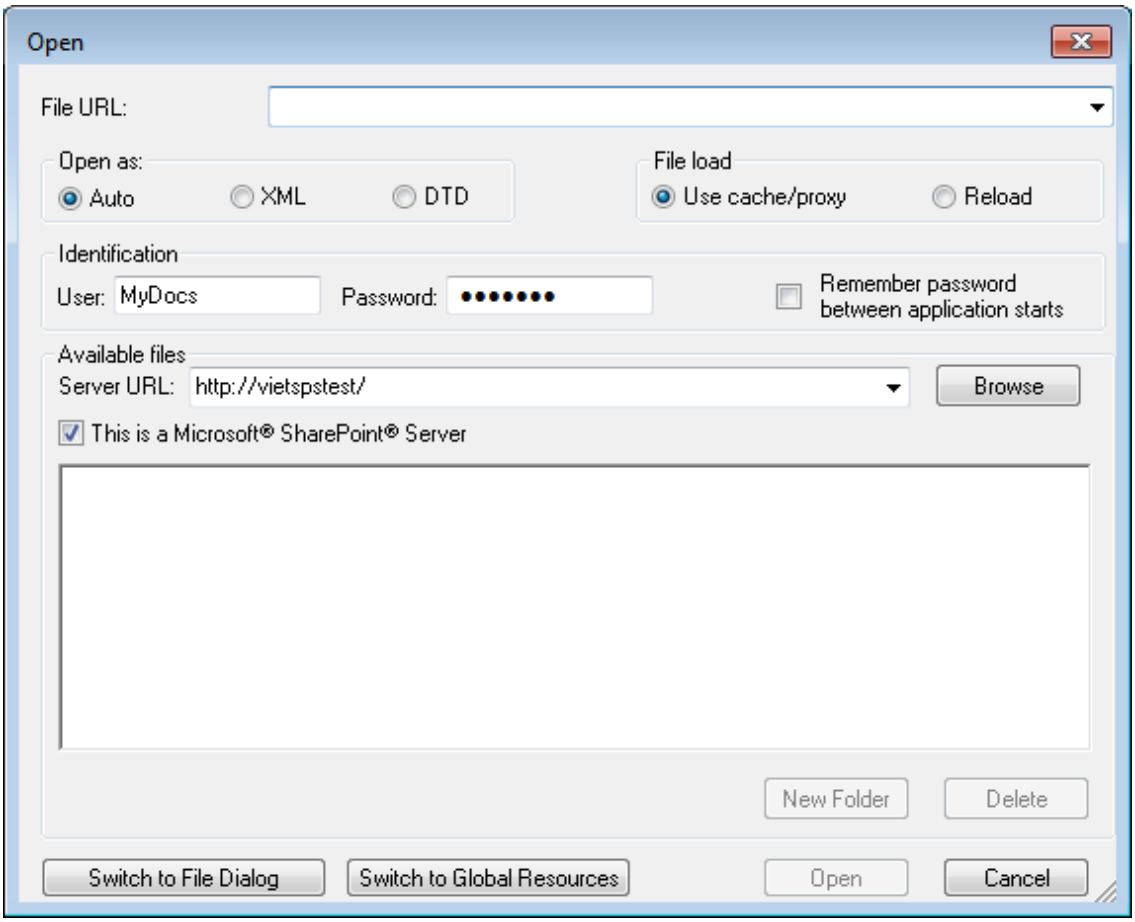
(C) 2015-2021 Altova GmbH



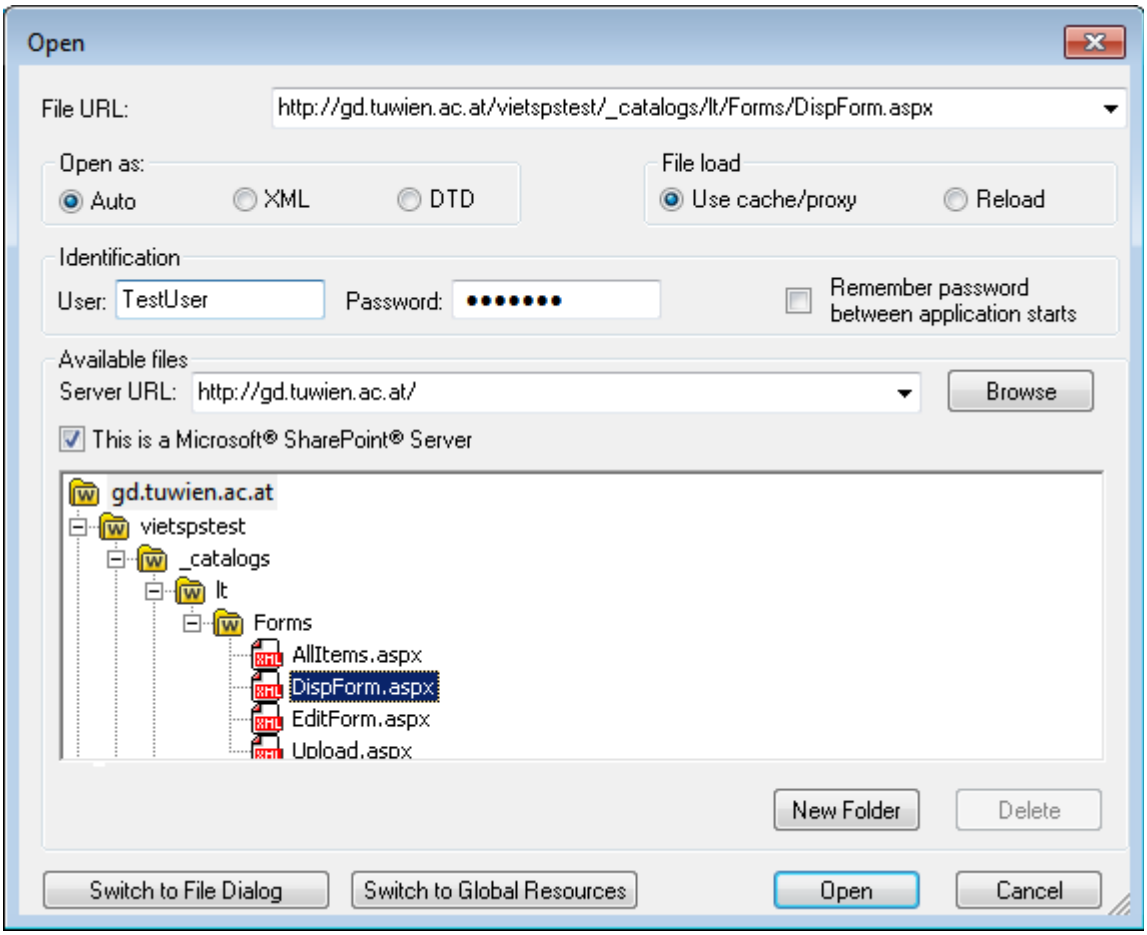
URL を使用してファイルを切り替える

URL から(開くまたは保存する)ファイルを選択するには以下を行ってください:

1. 「URL に切り替える」コマンドをクリックします。これにより、開くまたは保存ダイアログの URL モードにスイッチされます。(*下のスクリーンショットは開くダイアログを表示しています*)。



2. アクセスするURL をサーバー *URL* フィールドに入力します（上のスクリーンショット）。サーバーがMicrosoft® SharePoint® Server の場合 *Microsoft® SharePoint® Server* チェックボックスをチェックしてください。この種類のサーバー上にあるファイルと作業する場合、下のMicrosoft® SharePoint® Server メモを参照してください。
3. サーバーがパスワードにより保護されている場合、*ユーザー*と*パスワード*フィールドに入力してください。
4. 「参照」をクリックして、サーバーのディレクトリ構造をナビゲートします。
5. フォルダーツリー内でロードするファイルを参照してクリックします。



ファイルURL がファイルURL フィールドに表示されます（*上のスクリーンショット参照*）。「開く」または「保存」ボタンを使用できるようになります。

6. 「開く」をクリックしてファイルをロードするか、「保存」をクリックして保存します。

以下の点に注意してください：

- W ebDAV をサポートするサーバーとMicrosoft SharePoint Servers のみで参照機能を使用することができます。サポートされるプロトコールはFTP、HTTP およびHTTPS です。
- ファイルを開く際のロード処理を更に管理するには、ローカルのキャッシュまたは（ファイルが以前にロードされている場合スピードを向上する）プロキシのサーバーからファイルをロードします。（ファイルが事前にロードされることにより更に処理スピードを向上することができます）。また、電子出版またはデータベースシステムなどの作業中のファイルを再ロードして、「再ロード」オプションを選択します

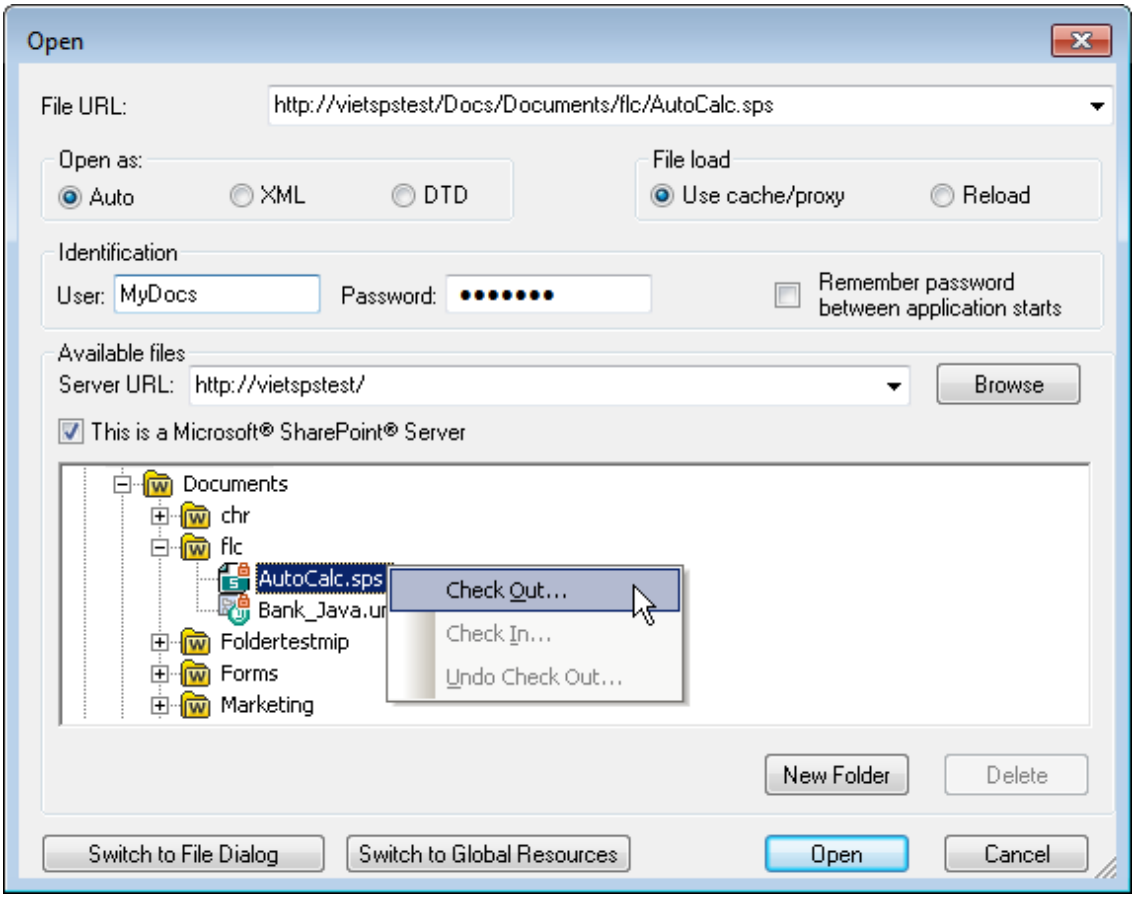
。

▼ Microsoft® SharePoint® Server メモ

Microsoft® SharePoint® Servers に関する以下の点に注意してください：

- 「利用可能なファイル」ペインに表示されるディレクトリ構造では、ファイルアイコンにはファイルのチェックイン/チェックアウトの状

態を表示するシンボルがあります（*下のスクリーンショット*）。



ファイルを右クリックすると、ファイルで使用することのできるコンテキスト メニューがポップアップします（*上のスクリーンショット*）。

- 異なるファイルアイコンが以下に表示されています:

	チェック イン状態。
	チェック アウト可能。他のユーザーによりチェック アウト状態。
	チェック アウト不可能。ローカルに対してチェック アウト。

- ファイルをチェックアウトした後、Altova アプリケーションを使用して編集することができ、「**ファイル | 保存 (Ctrl+S)**」を使用して保存することができます。
- 編集されたファイルを「URL を開く」ダイアログのコンテキスト メニューからチェックインすることができます（*上のスクリーンショット参照*）または、アプリケーションのメインウィンドウのファイルタブを右クリックすることによりポップアップするコンテキスト メニューを使用して（*下のスクリーンショット*）。

- ファイルが他のユーザーによりチェックアウトされている場合、チェックアウトすることができません。
- 他のユーザーによりファイルがチェック アウトされている場合、チェック アウトを行うことはできません。これによりファイルを変更す

ることなくサーバーに戻すことができます。

- 変更することなしにファイルがサーバーへ戻されます。ある Altova 製品でファイルをチェックアウトした場合、他の Altova 製品で同じファイルをチェックアウトすることはできません。この時点で使用できる Microsoft® SharePoint® Server をサポートする Altova アプリケーションのコマンドは、以下の通りです: **「チェックイン」** および **「チェックアウトを元に戻す」** です。

▼ グローバルリソースを使用してファイルを選択または保存する

グローバルリソースを使用してファイルを開くまたは保存する場合、**「グローバルリソース」** をクリックします。グローバルリソースを選択するためのダイアログが表示されます。これらのダイアログは以下のセクションで説明されています: グローバルリソースの使用. グローバルリソースの説明に関してはこのドキュメントの グローバルリソース セクションを参照してください。

関連項目

- ファイル| 新規作成
- メインウィンドウ

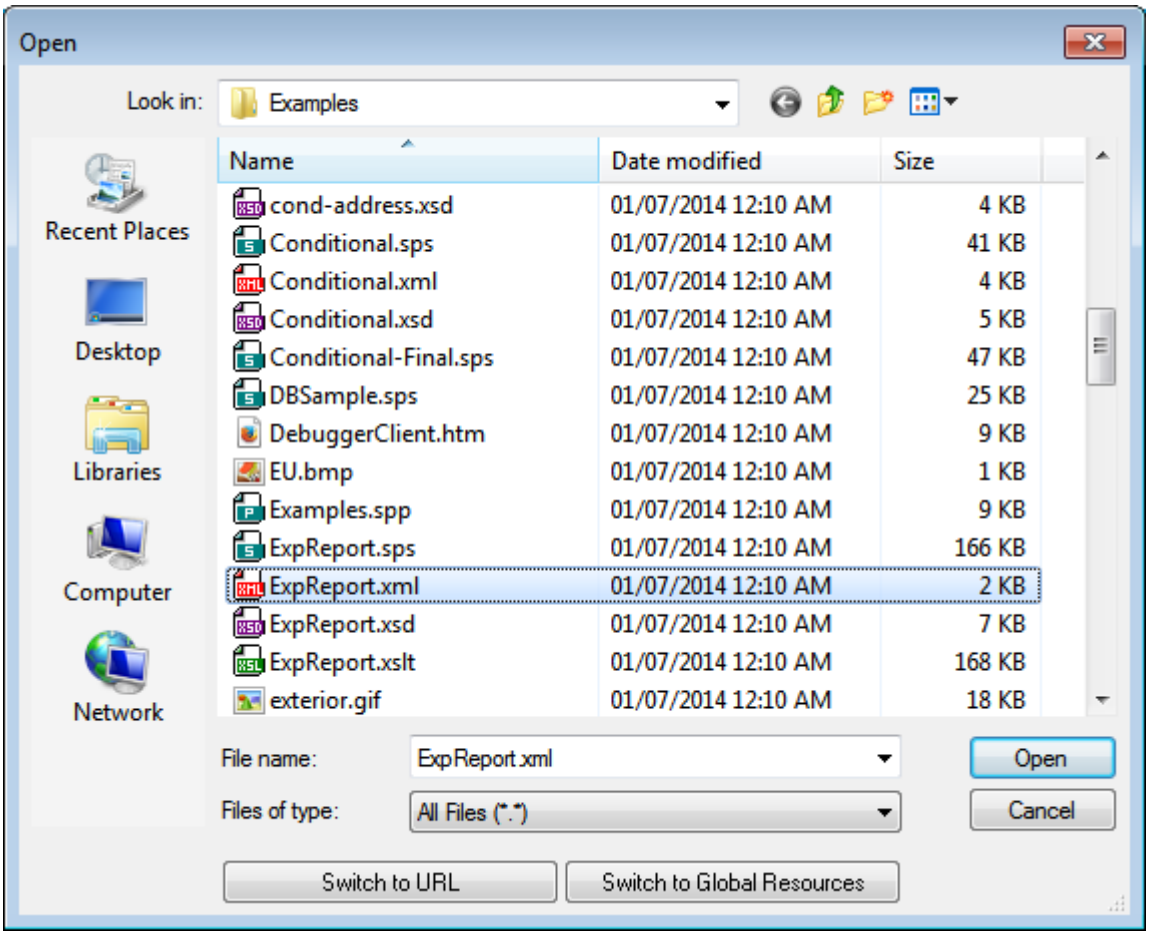
16.4.3 デザインを保存, すべて保存

[デザインを保存] (Ctrl+S) コマンド により、現在開かれているファイルを SPS ファイル（拡張子 .sps）として保存します。

[すべて保存] (Ctrl+Shift+S) コマンド により、開かれているすべての SPS ドキュメントが保存されます。

▼ URL とグローバルリソースを使用してファイルを選択または保存する

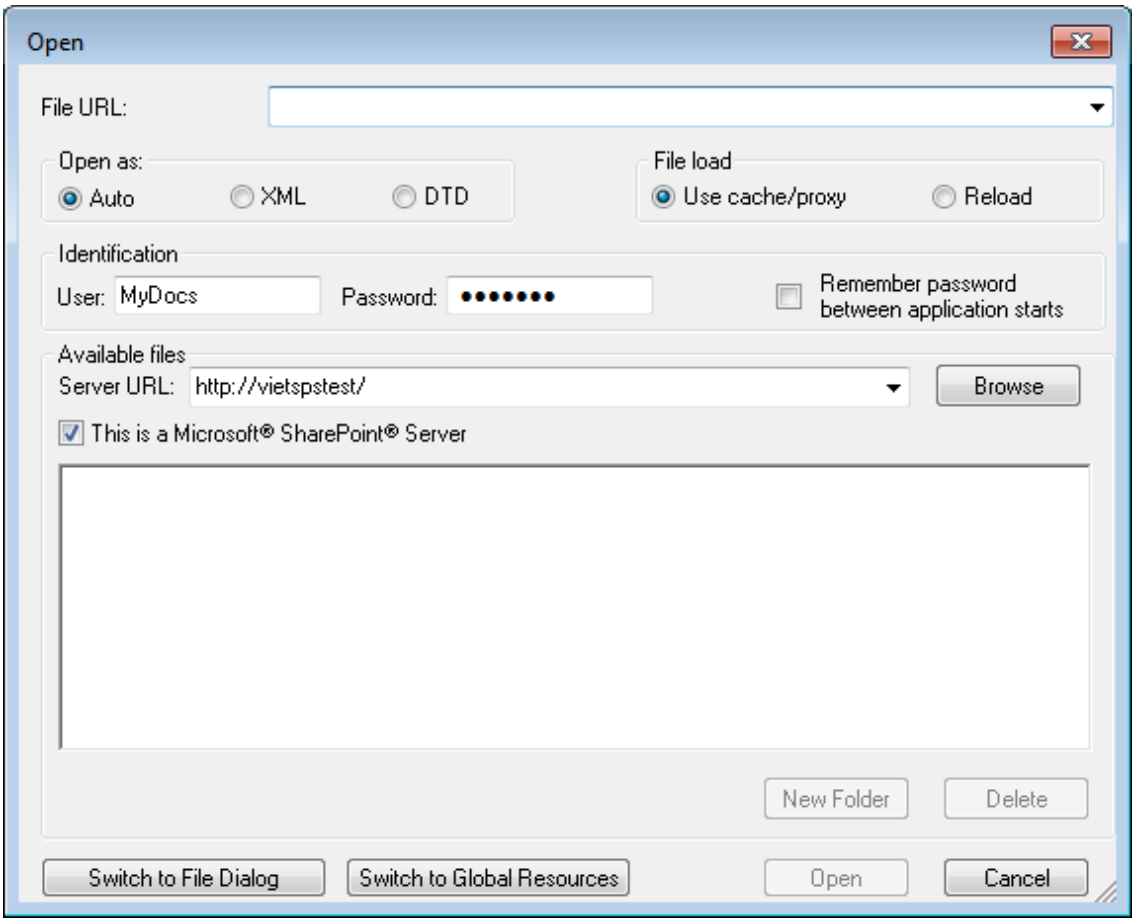
「ファイルを開く」および「ファイルの保存」ダイアログでは、URL またはグローバルリソースを使用して必要なファイルを選択またはファイルを保存することができます（*下のスクリーンショット参照*）。選択プロセスに移動するために **「URL に切り替える」** または **「グローバルリソース」** をクリックしてください。



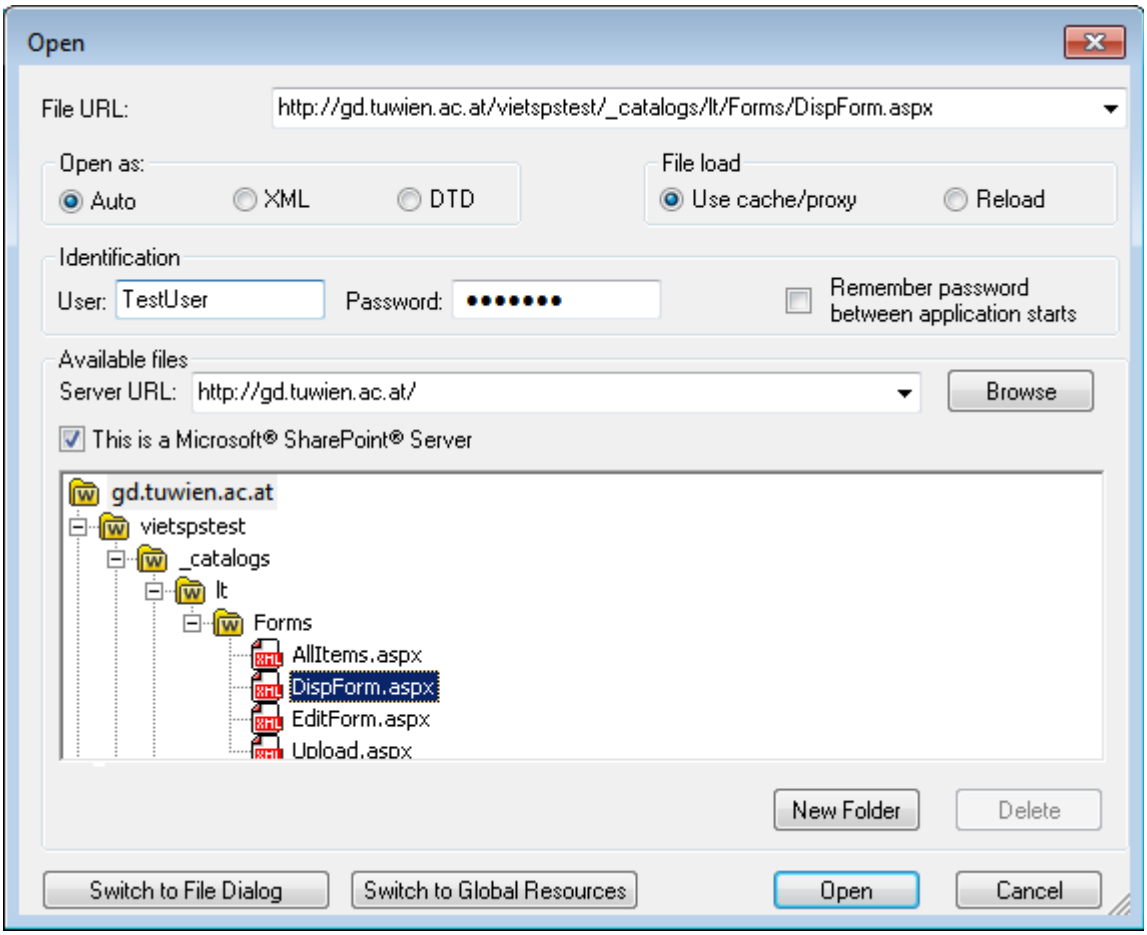
URL を使用してファイルを切り替える

URL から(開くまたは保存する)ファイルを選択するには以下を行ってください:

1. 「URL に切り替える」コマンドをクリックします。これにより、開くまたは保存ダイアログの URL モードにスイッチされます。(*下のスクリーンショットは開くダイアログを表示しています*)。



2. アクセスするURL をサーバー *URL* フィールドに入力します（上のスクリーンショット）。サーバーがMicrosoft® SharePoint® Server の場合 *Microsoft® SharePoint® Server* チェックボックスをチェックしてください。この種類のサーバー上にあるファイルと作業する場合、下のMicrosoft® SharePoint® Server メモを参照してください。
3. サーバーがパスワードにより保護されている場合、*ユーザー*と*パスワード*フィールドに入力してください。
4. 「参照」をクリックして、サーバーのディレクトリ構造をナビゲートします。
5. フォルダーツリー内でロードするファイルを参照してクリックします。



ファイルURL がファイルURL フィールドに表示されます（*上のスクリーンショット参照*）。「開く」または「保存」ボタンを使用できるようになります。

6. 「開く」をクリックしてファイルをロードするか、「保存」をクリックして保存します。

以下の点に注意してください：

- W ebDAV をサポートするサーバーとMicrosoft SharePoint Servers のみで参照機能を使用することができます。サポートされるプロトコールはFTP、HTTP およびHTTPS です。
- ファイルを開く際のロード処理を更に管理するには、ローカルのキャッシュまたは（ファイルが以前にロードされている場合スピードを向上する）プロキシのサーバーからファイルをロードします。（ファイルが事前にロードされることにより更に処理スピードを向上することができます）。また、電子出版またはデータベースシステムなどの作業中のファイルを再ロードして、「再ロード」オプションを選択します

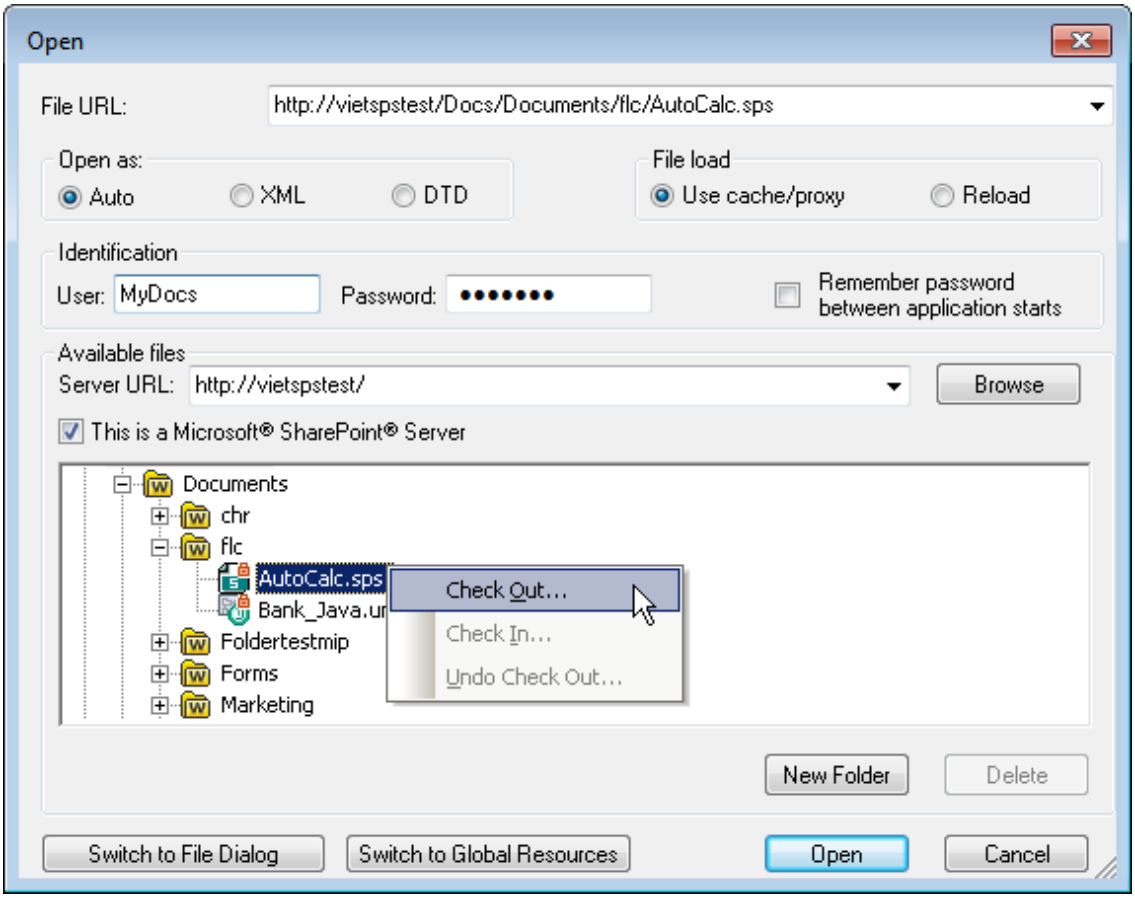
。

▼ Microsoft® SharePoint® Server メモ

Microsoft® SharePoint® Servers に関する以下の点に注意してください：

- 「利用可能なファイル」ペインに表示されるディレクトリ構造では、ファイルアイコンにはファイルのチェックイン/チェックアウトの状

態を表示するシンボルがあります（*下のスクリーンショット*）。



ファイルを右クリックすると、ファイルで使用することのできるコンテキスト メニューがポップアップします（*上のスクリーンショット*）。

- 異なるファイルアイコンが以下に表示されています:

	チェック イン状態。
	チェック アウト可能。他のユーザーによりチェック アウト状態。
	チェック アウト不可能。ローカルに対してチェック アウト。

- ファイルをチェックアウトした後、Altova アプリケーションを使用して編集することができ、「ファイル | 保存（Ctrl+S）」を使用して保存することができます。
- 編集されたファイルを「URL を開く」ダイアログのコンテキスト メニューからチェックインすることができます（*上のスクリーンショット参照*）または、アプリケーションのメインウィンドウのファイルタブを右クリックすることによりポップアップするコンテキスト メニューを使用して（*下のスクリーンショット*）。

- ファイルが他のユーザーによりチェックアウトされている場合、チェックアウトすることができません。
- 他のユーザーによりファイルがチェック アウトされている場合、チェック アウトを行うことはできません。これによりファイルを変更す

ることなくサーバーに戻すことができます。

- 変更することなしにファイルがサーバーへ戻されます。あるAltova 製品でファイルをチェックアウトした場合、他のAltova 製品で同じファイルをチェックアウトすることはできません。この時点で使用できるMicrosoft® SharePoint® Server をサポートするAltova アプリケーションのコマンドは、以下の通りです: **「チェックイン」** および **「チェックアウトを元に戻す」** です。

▼ グローバルリソースを使用してファイルを選択または保存する

グローバルリソースを使用してファイルを開くまたは保存する場合、**「グローバルリソース」** をクリックします。グローバルリソースを選択するためのダイアログが表示されます。これらのダイアログは以下のセクションで説明されています: グローバルリソースの使用. グローバルリソースの説明に関してはこのドキュメントのグローバルリソースセクションを参照してください。

関連項目

- ファイル | 閉じる

16.4.4 名前を付けて保存

[**名前を付けて保存**] コマンドにより、デザインをSPS ファイルまたはPXF (Portable XML Form) ファイルとして保存することができます。コマンドをクリックすると、デザインの保存ダイアログが表示されます (以下のスクリーンショットを参照)。目的のフォーマットを選択して、[OK] をクリックしてください。

SPS フォーマットは StyleVision デザインのAltova 標準フォーマットです。**PXF フォーマット**は、デザイン内に含まれるすべてのファイル (スキーマファイル、XML ファイル、イメージファイル、生成されたXSLT スタイルシートなど) を内包したAltova フォーマットで、Authentic View にてデザインを開き、デザインをベースにしたHTML およびまたはRTF 出力を行うのに必要なすべてのファイルを簡単にやり取りすることが可能になります。

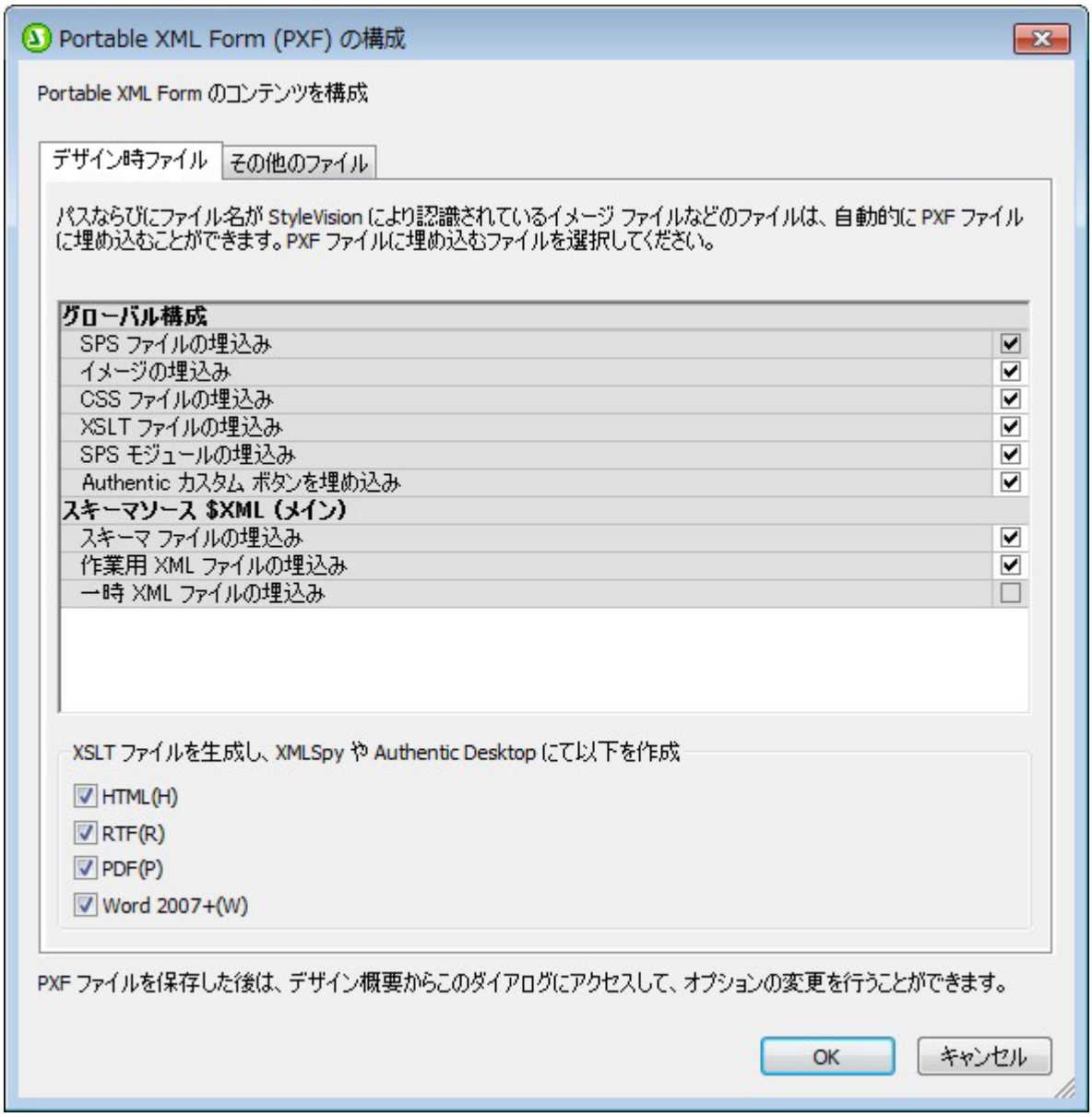
(C) 2015-2021 Altova GmbH

SPS として保存する

SPS オプションを選択すると、W indows システムのファイルを保存ダイアログが表示されます。保存を行うための操作は「デザインを保存コマンド」コマンドに記述されます。名前を付けて保存コマンドにより、既に保存されたファイルとは異なるファイル名でデザインを保存することができます。

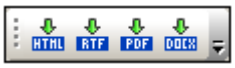
PXF として保存する

PXF オプションを選択することにより、W indows システム標準の名前を付けて保存ダイアログが表示されます。ファイルの保存は、デザインを保存 コマンドと同様の方法に加え、PXF ファイルに追加するファイルを選択するためのステップが加わります。PXF ファイル名を指定すると、PXF の構成ダイアログが表示され、埋め込みを行うファイルを選択することができます（以下のスクリーンショットを参照）。



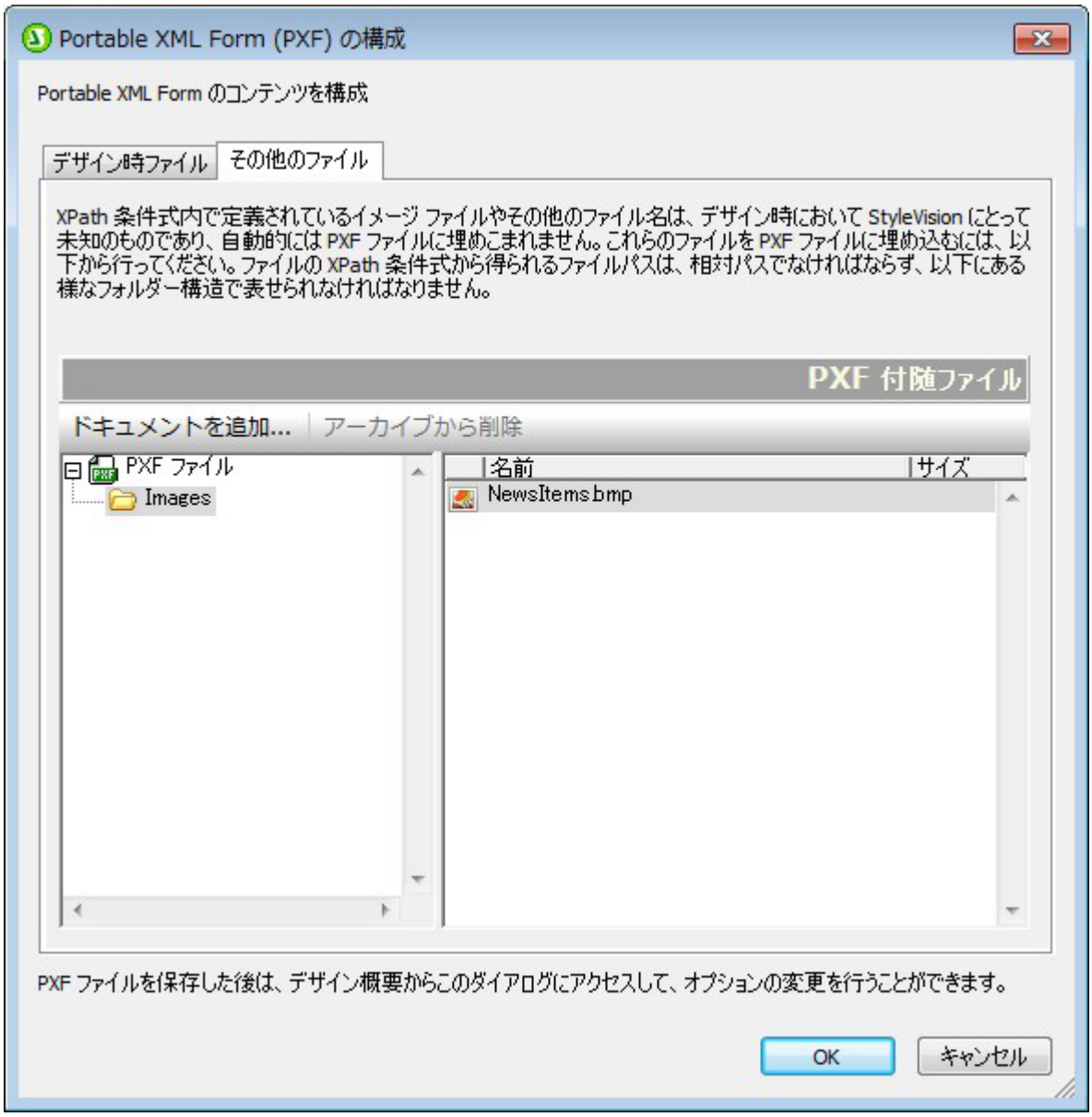
デザイン時ファイルタブのグローバル構成セクションでは、デザインに関連するソースファイルの埋め込みを選択/選択解除することができます。更に、デザインから生成された XSLT ファイルを埋め込むかを選択することもできます。XSLT ファイルペインにて、埋め込まれた XSLT ファイル

が生成する出力フォーマットを選択してください。XSLT ファイルがPXF ファイルに追加され、そのPXF ファイルがAltova 製品のAuthentic View にて開かれた場合、出力フォーマットを生成と閲覧を行うためのツールバーボタンがAuthentic View にて有効になります(以下のスクリーンショットを参照)。



メモ StyleVision の高いバージョンでのみサポートされている(*高から低い順: Enterprise*、*Professional*、*Basic*)出力のためのXSLT ファイルは、 PXF ファイル内で作成され、PXF ファイルが低いバージョンで開かれる場合、PXF ファイルを保存すると、低いバージョンでサポートされる出力のためのXSLT ファイルは、保存されません。PXF ファイルの保存を継続するかを問うプロンプトが表示、または、保存を中断して、サポートされないフォーマットを保管します。

その他のファイルタブでは、デザインには直接含まれていないファイルを指定することができます(以下のスクリーンショットを参照)。例えば、デザイン内部のXPath 条件式により生成されたURL にて参照されるイメージファイルなどを含めることができます。以下のスクリーンショットでは、Images フォルダー以下にあるイメージのNewsItems.bmp がPXF ファイルに含まれるように指定されています。



その他のファイルを PXF ファイルに追加するには、**ドキュメントを追加**ボタンをクリックして目的のファイルを選択します。ファイルを開くダイアログが表示され、SPS ファイルが配置されているフォルダーが開かれます。このフォルダー内にあるファイルまたはその子孫フォルダー内にあるファイルを選択することができます。ファイルが PXF ファイルへ追加されると、そのファイルと対応するフォルダー構造が表示されます。上のスクリーンショットからは、SPS ファイルが配置されているフォルダーにある Images という名前のフォルダー内に NewsItems.bmp というファイルがあることが理解できます。

SPS ファイルが配置されているフォルダーの上位にあるフォルダーからファイルが選択されると、エラーが返されます。

SPS デザインでは、その他のファイルに対する参照を相対パスで、更に付属ファイルのペインに示されているとおりに記述する必要があります。例えば、上のスクリーンショットにある NewsItems.bmp には、Images/NewsItems.bmp という相対パスで参照を行う必要があります。

関連項目

- デザインの保存、互換性、すべて保存
- PXF ファイル:SPS とその関連ファイルのコンテナー

16.4.5 MobileTogether デザインファイルとしてエクスポートする

このコマンドは、Altova MobileTogether デザインファイルをアクティブな SPS デザインから生成します。A MobileTogether デザインファイルはモバイルデバイスのために MobileTogether アプリ内でソリューションを実行するために使用されます。例えば、MobileTogether ソリューションは、スマートフォンなどのモバイルデバイスで、データベースの内容を閲覧し編集するために開くことができます。MobileTogether ソリューションは Altova MobileTogether Designer 内でデザインすることができます。このコマンドは、MobileTogether Designer 内で編集することができる、SPS デザインを MobileTogether デザインに変換することができます。詳細に関しては、MobileTogether ウェブページを参照してください。オプションダイアログ（「ツール | オプション」）の MobileTogether Design タブ内で変換オプションを使用することができます。

メモ　SPS デザインの全ての機能が MobileTogether デザイン内にコレスポンデンスを持つとは限りません。このコマンドを実行下の地、MobileTogether Designer 内で生成されたファイルをレビューし、必要に応じて変更してください。以下のデザイン機能は MobileTogether デザインにエクスポートされることは既知ではありません（i） StyleVision デザイン内でリストされている最初の項目以降のソース(Enterprise 機能)。（ii） グローバルテンプレート (ii) モジュール。

16.4.6 Authentic の XML データを保存、名前を付けて保存

Authentic View では、SPS に関連付けられた作業用 XML ファイルまたは DB を編集することができます。[**Authentic の XML データを保存**] コマンドにより、変更された箇所が作業用 XML ファイルや DB に対して保存されます。StyleVision における XML ファイルの編集のほかにも、Altova XMLSpy や Authentic Desktop の Authentic View にて XML ドキュメントや DB を編集することができます。

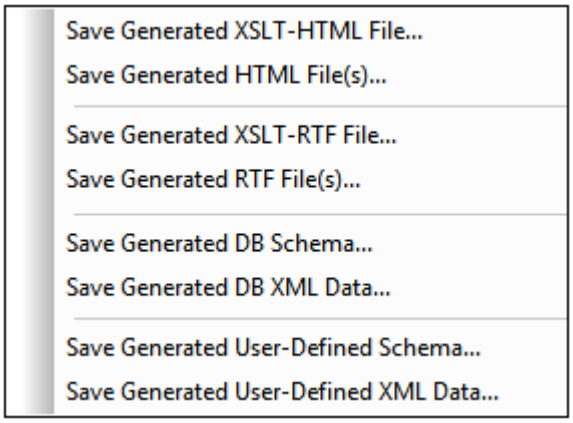
Authentic の XML データを名前を付けて保存 コマンドにより、Authentic XML ドキュメントを別のファイルに保存することができます。

関連項目

- Authentic View

16.4.7 生成されたファイルの保存

[生成されたファイルの保存] コマンドにあるサブメニューには、以下のファイルを保存するためのオプションが含まれています（以下のスクリーンショットを参照）。生成されたファイルの利用方法については、使用方法 | 生成されたファイルを参照ください。



生成されたXSLT-HTML ファイルの保存

[生成されたXSLT-HTML ファイルの保存] コマンドにより、 からHTML 出力を行うXSLT ファイルが生成されます。このXSLT ファイルにより、XML ドキュメントからHTML への変換を行うことができます。

生成されたHTML ファイルの保存

[生成されたHTML ファイルの保存] コマンドにより、HTML ファイルが生成されます。デザインにて複数のドキュメントの出力 が指定されている場合、複数のHTML ファイルが生成されます。この操作には2 つの入力ファイルが必要になります:

- 現在アクティブな SPS ファイルに割り当てられた作業用 XML ファイル。作業用 XML ファイルが割り当てられていない場合、[生成された HTML ファイルの保存] コマンドは選択不可能になります。DB をベースにしたSPS の場合、自動的に生成される編集不可能な XML ファイルが作業用 XML ファイルとなり、作業用 XML ファイルは手動で割り当てられません（次を参照:DB スキーマとDB XML ファイルを参照ください）。
- 現在アクティブな SPS ファイルから自動的に生成されるXSLT ファイル。

生成されたXSLT-RTF ファイルの保存

[生成されたXSLT-RTF ファイルの保存] コマンドにより、 からRTF 出力を行うXSLT ファイルが生成されます。このXSLT ファイルにより、XML ドキュメントからRTF への変換を行うことができます。

生成されたRTF ファイルの保存

[生成されたRTF ファイルの保存] コマンドによりRTF ファイルが生成されます。デザインにて複数のドキュメントの出力 が指定されている場合、複数のRTF ファイルが生成されます。この操作には2 つの入力ファイルが必要になります:

- 現在アクティブな SPS ファイルに割り当てられた作業用 XML ファイル。作業用 XML ファイルが割り当てられていない場合、コマンドが選択不可能になります。DB をベースにしたSPS の場合、自動的に生成される編集不可能な XML ファイルが作業用

XML ファイルとなり、作業用 XML ファイルは手動で割り当てられません（（次を参照:DB スキーマとDB XML ファイルを参照ください）。
- 現在アクティブな SPS ファイルから自動的に生成される XSLT ファイル。

生成されたデータベース スキーマの保存

DB をベースにした SPS を作成するため DB へ接続すると、DB の構造をベースにした一時的な XML スキーマが StyleVision により生成、ロードされます。[生成されたデータベース スキーマの保存] コマンドにより、この生成された XML スキーマを保存することができます。XML DB の場合、スキーマ ファイルは生成されず、DB やその他の場所にあるスキーマが使用されます。このコマンドは XML DB に対して選択できません。

生成されたデータベース XML データの保存

[生成されたデータベース XML データの保存] コマンドにより、XML 構造内の DB から得られたデータを含む XML ファイルが、DB から生成された XML スキーマの構造に従うかたちで生成、保存されます。StyleVision Power Stylesheet にて DB フィルターが定義されている場合、これらフィルターがデータのインポートに対して適用されます。XML DB の場合、スキーマ ファイルは生成されず、DB の XML 列やその他の場所にあるスキーマが使用されます。このコマンドは XML DB に対して選択できません。

生成されたユーザー定義スキーマの保存

SPS にユーザー定義スキーマが含まれる場合に、このコマンドが有効になります。スキーマ ツリー サイド バーにて作成したスキーマが、.xsd 拡張子を持つ XML スキーマとして保存されます。

生成されたユーザー定義 XML データの保存

ユーザー定義のスキーマに対応する、インポートされた HTML ファイル内にあるデータが XML ファイルとして保存されます。対応するデータは XML スキーマ ノードとして作成された（デザイン ビューの）HTML ドキュメント内にあるノードとなります。

16.4.8 FlowForce へのデプロイ

[**FlowForce へのデプロイ**] コマンドにより、.transformation ファイルを Altova FlowForce Server へデプロイすることができます。.transformation ファイルには SPS にてデザインされた変換を行うのに必要なすべてのファイルと情報が含まれます。.transformation ファイルを FlowForce Server へデプロイすると、.transformation ファイルを使って、ジョブ定義にて指定されたトリガーに従うかたちで変換を生成するためのジョブを、FlowForce にて作成することができます。FlowForce ジョブの作成方法については、FlowForce のドキュメンテーションを参照ください。

.transformation ファイルが Portable XML Format（PXF）ファイルから生成されます。そのため、[**FlowForce へのデプロイ**] コマンドは、PXF ファイルがアクティブの状態でのみ使用することができます。SPS ファイルがアクティブの場合でも、[**FlowForce へのデプロイ**] コマンドはアクティブになりますが、SPS ファイルを PXF ファイルとして保存するよう促されます。SPS ファイルから PXF ファイルを作成するには、[**ファイル | 名前を付けて保存**] コマンドを選択して、保存するファイルの形式に PXF を選択してください。

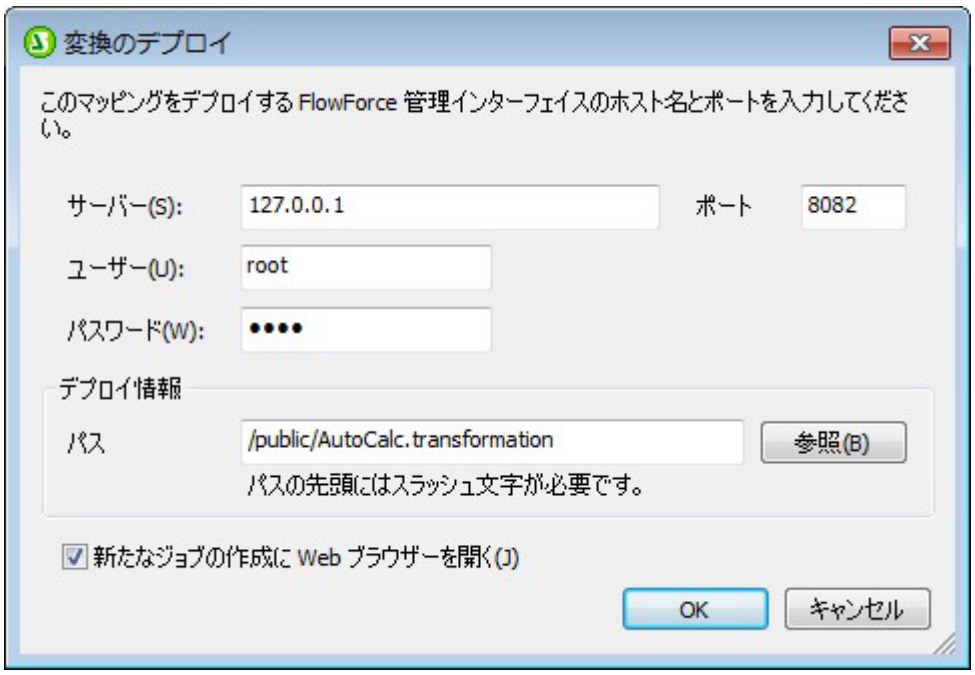
以下の点に注意してください：

- PXF ファイル形式で保存することで、（イメージ ファイルなどの）外部ファイルを内部に含めることができます。変換に必要な外部ファイルが PXF ファイル内部に含まれていない場合、外部ファイルを FlowForce Server 上に保存する必要があります。外部ファイルは（FlowForce のジョブ定義にて指定された）作業ディレクトリからアクセスされるため、作業用ディレクトリからこれらのファイルが正しくアクセスできるように、作業ディレクトリに対して相対的に配置する必要があります。
- StyleVision の変換が行われる FlowForce のジョブが実行されると、ジョブが StyleVision Server へ渡され、PXF ファイルのコンテンツが StyleVision Server によりジョブのパラメーターにより指定された作業用ディレクトリへ展開されます。この展開が行われる際にファイル名が重複しないことを保証するため、PXF ファイル内に含まれているファイルと同じ名前のファイルが作業用ディレクトリに含まれていないことを確認する必要があります。

[**FlowForce へのデプロイ**] コマンドを実行する前に、ライセンスされた Altova FlowForce Server と Altova StyleVision Server が正常に動作していることを確認してください。FlowForce Server のセットアップ方法に関する情報は、Altova FlowForce のドキュメンテーションを参照ください（StyleVision Server は FlowForce Server に含まれています）

デプロイコマンド

[**FlowForce へのデプロイ**] コマンドにより、変換のデプロイ ダイアログが表示されます（以下のスクリーンショットを参照）。



このダイアログでは、以下の項目を指定することができます:

- FlowForce Web Server (FlowForce Server ではないことに注意してください) のアドレスとポート番号や、FlowForce Server へのアクセス情報 (ユーザー名とパスワード)。
- 変換ファイルのファイル名とそのファイルが保存される FlowForce Server 上の場所。ファイル パスは FlowForce Server のルート ディレクトリを表すスラッシュ (/) から始める必要があります。
- ファイルが最後に保存された後にデザインが変更されると、[**デプロイ前に変更点を保存**] チェック ボックスが選択可能になります。変更を保存するには、このチェック ボックスを有効にしてください。
- SSL により暗号化された接続によりマッピングをデプロイする場合、「**SSL を使用**」チェックボックスを選択します。これは、FlowForce Server が既に SSL 接続を受け入れるように構成されているものの想定しています。詳細に関しては、FlowForce Server ドキュメントを参照してください。

[OK] をクリックすることで、.transformation ファイルが FlowForce Server 上の指定された場所にデプロイされます。[**新たなジョブの作成に Web ブラウザーを開く**] チェックボックスにチェックを入れた場合、ブラウザーが開かれ、デプロイ ステップにより作成されたジョブを編集することができます（上のスクリーンショットを参照）。

StyleVision Server の複数のバージョン

`.transformation` ファイルをデプロイするサーバ上に、FlowForce Server 管理下で StyleVision Server の複数のバージョンが作動している場合、(Windows サーバーのみに適用可能)、「*Select StyleVision Server の選択*」ダイアログ内で、このマッピングが実行される Stylevision Server バージョンを指定してください。使用するバージョンを手動で選択することが、また、自動的に最も適したバージョンを選択することもできます。

(C) 2015-2021 Altova GmbH

FlowForce Server 管理下で作動する各 StyleVision Server バージョンのためにFlowForce Server インストールディレクトリに .tool ファイルが含まれる場合、このダイアログが表示されます。デフォルトでは、StyleVision Server をFlowForce Server の一部としてインストールする際、StyleVision Server .tool ファイルが自動的にこのディレクトリに追加されます。FlowForce 内に.tool ファイルが保存されるパスは、以下のとおりです: **C:\Program Files\Altova\FlowForceServer2021\tools**。FlowForce Server 管理下で実行する StyleVision Serverの追加バージョンが存在する場合、.tool ファイルが上記のディレクトリに手動で追加される必要がある場合があります。StyleVision Server の .tool ファイルは以下で検索することができます: **C:\Program Files\Altova\StyleVisionServer2021\etc**.

メモ　FlowForce Server の使用方法については、FlowForce ドキュメント のドキュメンテーションを参照ください。

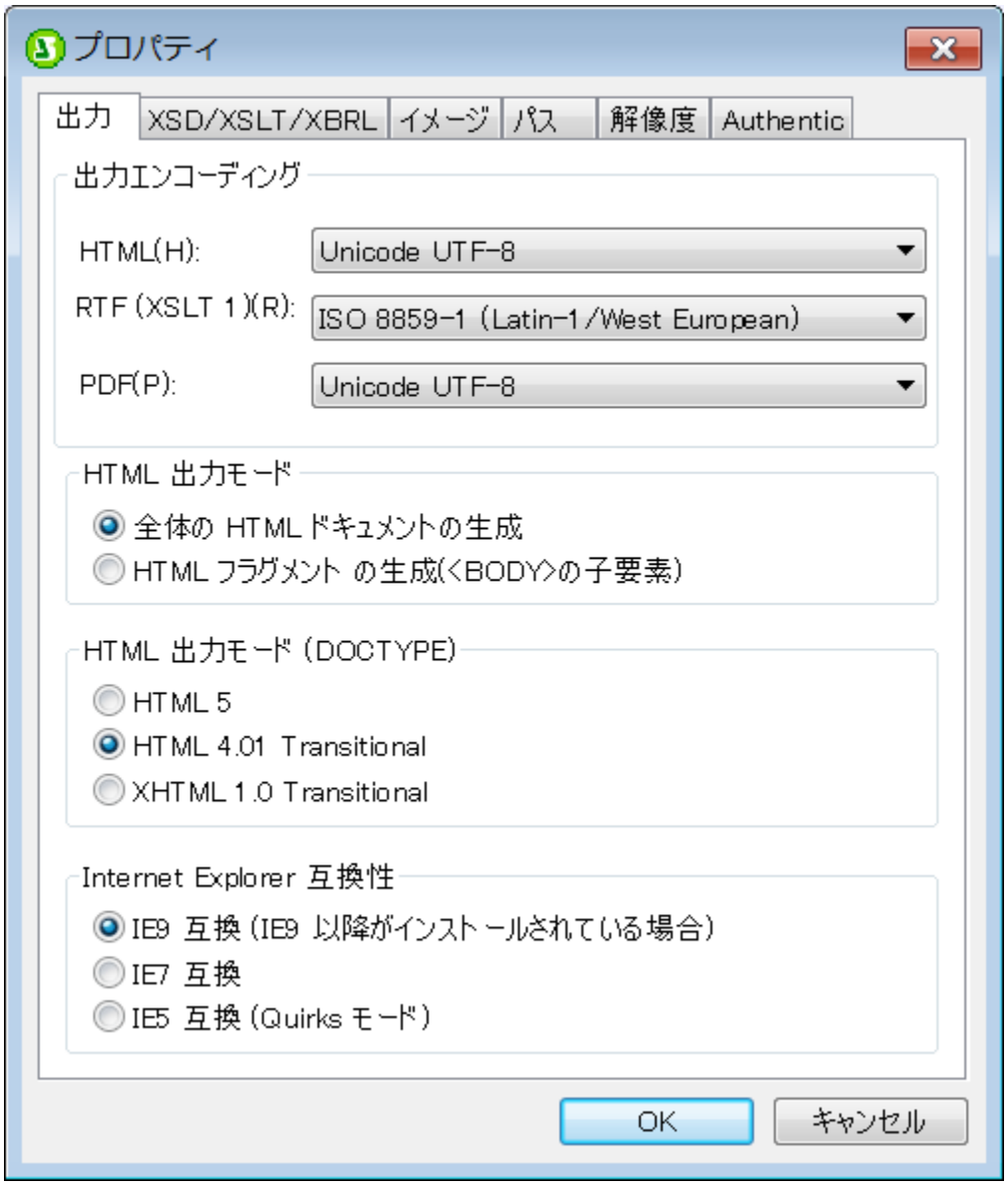
16.4.9　Web デザイン

[**Web デザイン**] コマンドには[**ASPX Web アプリケーションの生成**] コマンドがサブメニューに含まれています。後者のコマンドにより、ASPX アプリケーションを動作させるために必要なすべてのファイルが、指定されたフォルダー内部に生成されます。ブラウザーにより C# コードの出力ドキュメントである ASPX ファイルが読み取られると、ASPX パッケージ内にある XSLT ファイルにより動的に変換されたソースデータベースや XML ファイルのデータが表示されます。変換処理の出力ドキュメントである ASPX ファイルが、ソースデータベースや XML ファイルの最新データとともに更新されます。

に関する詳しい説明は、次を参照してください: ASPX インターフェイスによる Web アプリケーション。

16.4.10　プロパティ

プロパティ コマンドにより、プロパティダイアログ SPS のプロパティをセットするためのプロパティダイアログが表示されます



出力

出力タブでは以下のプロパティをセットすることができます:

- **出力エンコーディング**: 出力エンコーディング ペインでは、出力ドキュメントのエンコーディングを選択することができます。このダイアログにてエンコーディングを変更することで、現在アクティブな SPS のエンコーディングが変更されます。デフォルトのエンコード を後に作成されたすべてのSPS ドキュメントのために指定することができます。これはオプションダイアログのエンコードタブ内で行うことができます。

- *HTML 出力モード*: HTML ドキュメント全体、または、HTML body 要素の子要素が出力であるかを選択することができます。子要素は、他に、すなわち同じレベルでの、並行した出力で、全ての子孫を回帰的に含みます。この結果、出力ドキュメントはHTML コードのフラグメントであることができます。

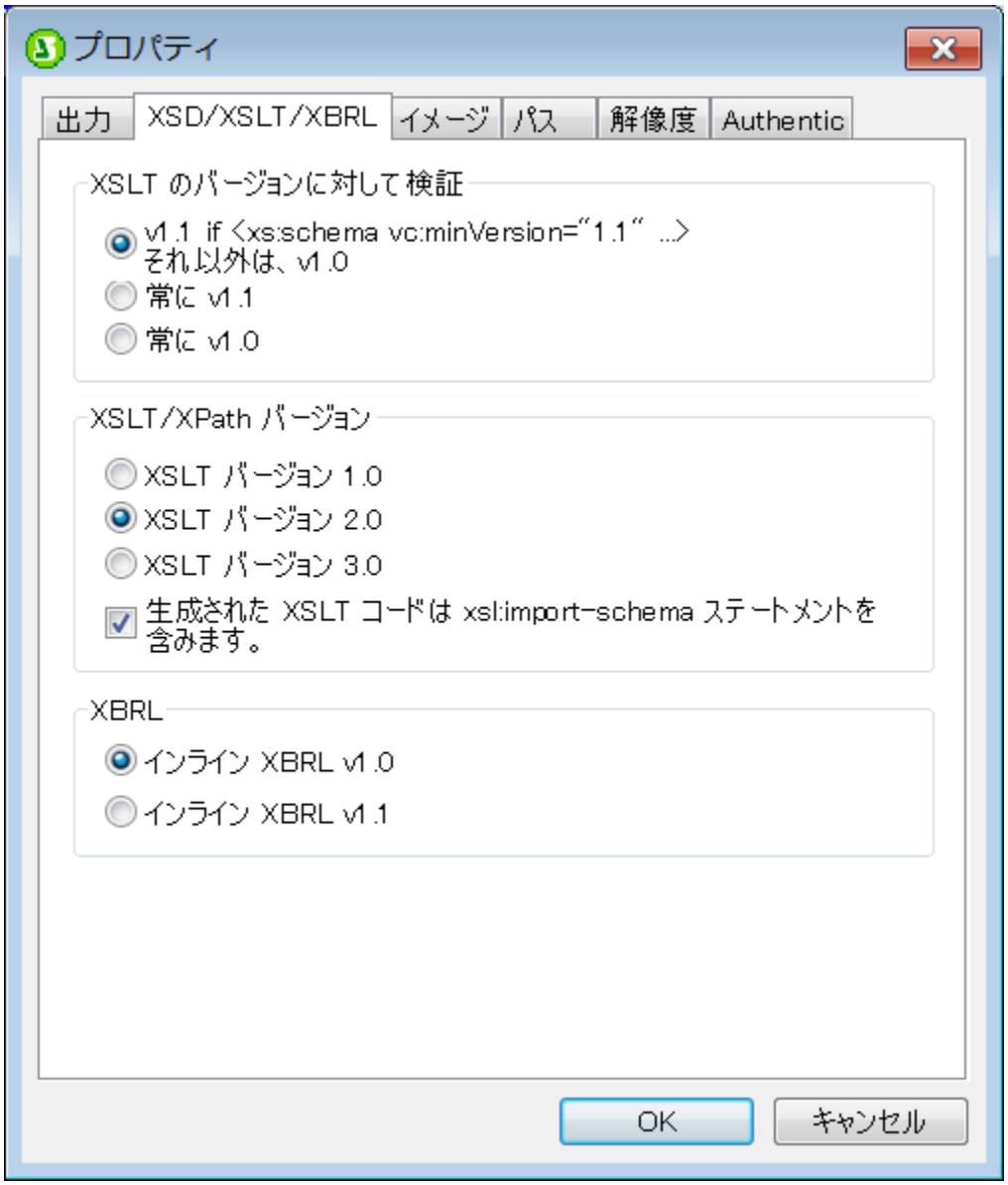
- *HTML 出力モード (DOCTYPE):* 生成される HTML 出力を、HTML5、HTML 4.01 Transitional ドキュメント、または XHTML 1.0 Transitional ドキュメントから選択することができます。この設定は SPS ドキュメントの作成または編集中に変更することもできます。

- *Internet Explorer の互換性 と CSS サポート:* IE 6 前のInternet Explorer (IE) バージョン内のCSS サポートは、一部のアスペクトで不十分な箇所がある可能性があります。CSS サポートは、拡張されており、IE 6 内では修正されており、更にIE 7、IE 9 および以降では改良され拡張されています。

  SPS 内では、プロパティダイアログ内で互換性のあるモードを選択することができます (*上のスクリーンショット*)。IE 5、IE 7、またはIE 9 を選択することができます。(IE 9 互換性を適用するには、IE 9 または以降がインストールされている必要があります)。IE サポートの指定されたレベルは、すぐにAuthentic View とHTML プレビューで使用することができます。新規のSPS ドキュメントは、IE7 互換性を選択して作成します。Altova StyleVision の古いバージョンで作成されたはSPS ドキュメントを(プロパティダイアログで選択された)必要な互換性モードで保存することができます。

XSD/XSLT

このタブでは、XML 検証のために使用されるXSD バリデーターおよびSPS 内で使用されるXSLT バージョンを指定することができます。



StyleVision には、XSD 1.0 バリデーターとXSD 1.1 バリデーターが搭載されています。以下のオプションを選択することができます:

- XSD ドキュメントの `/xs:schema/@vc:minVersion` 属性 が `1.1` に設定されている場合、XSD 1.1 バリデーターを使用します。それ以外の場合、XSD 1.0 バリデーターを使用します。
- 常にXSD 1.1 バリデーターを使用する。
- 常にXSD 1.0 バリデーターを使用する。

このタブからアクティブなドキュメントのためにXSLT バージョンを選択します。*xsl:import-schema 宣言* オプションの使用を確認すると、XSLT 2.0 と3.0 仕様の `xsl:import-schema` 要素がStyleVision により生成されたXSLT ドキュメント内に含まれます。XML ドキュメント内に `xsi:schemaLocation` 属性が存在しない場合、スキーマからデータが読み取られるようにするには、このオプションが選択されていることが奨励されます。

イメージ

このタブでは、RTF 出力にて、イメージをリンクするのか埋め込むのかを選択することができます。XSLT 2.0 が選択されている場合、埋め込みオプションを利用することができます。設定は各 SPS に対して個別に行われます。以下の操作によりイメージの埋め込みを行うことができます:

1. 目的の SPS がアクティブな状態で、プロパティ ダイアログ ([**ファイル | プロパティ**]) を開きます。
2. (デフォルトの設定がチェックされている) *RTF および Word 2007+ チェックボックス のための*[イメージを埋め込み] チェックボックスにチェックを入れます (デフォルトの設定ではチェックが入っています)。アクティブな SPS の XSLT バージョンとして XSLT 2.0 が選択されている場合だけ、イメージの埋め込みは行われます。.
3. [OK] をクリックして、SPS を保存します。アクティブな SPS に対して設定が保存されます。

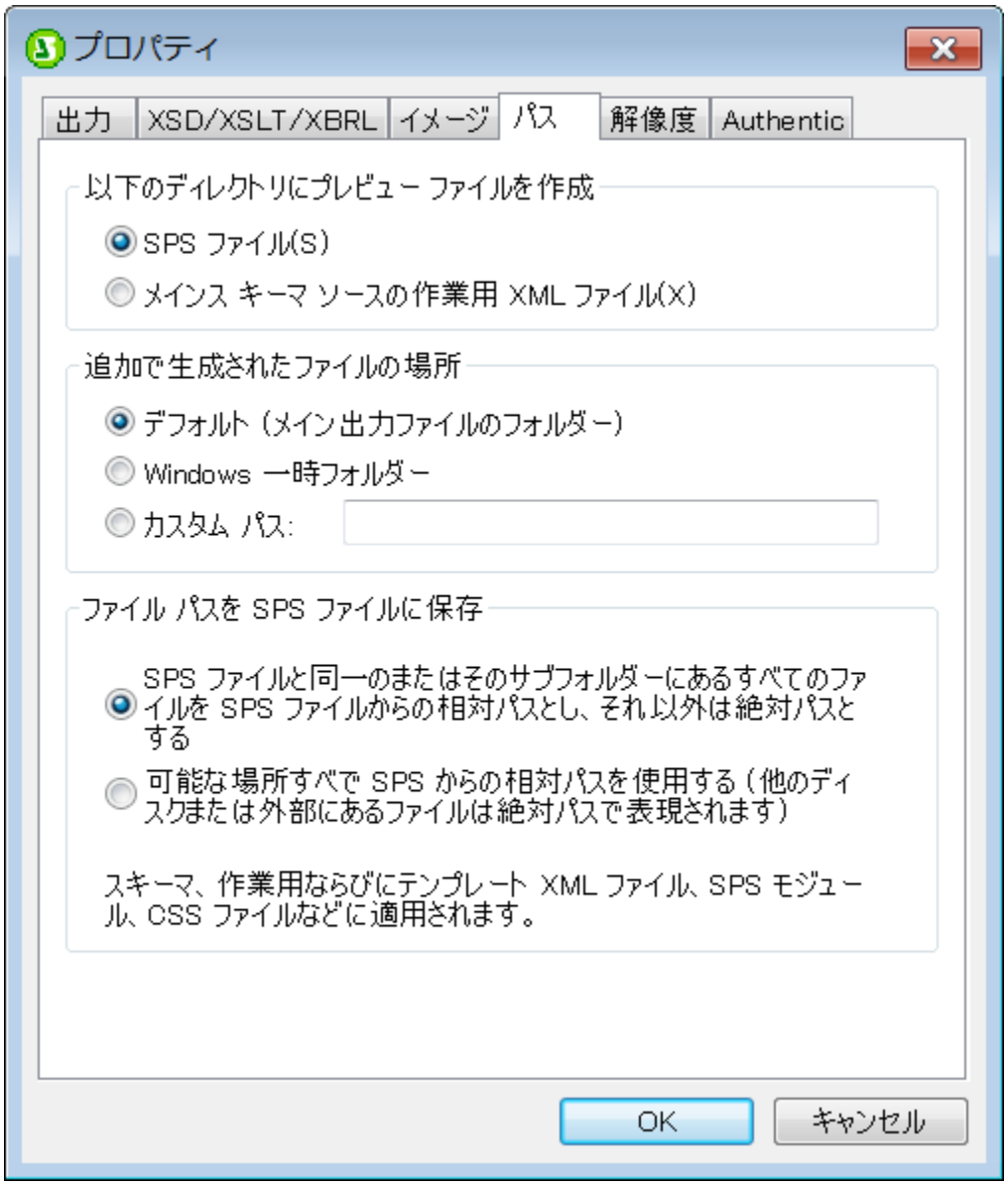
[イメージを埋め込み] チェック ボックスにチェックが入っていない場合、イメージ プロパティにて指定されたイメージ ファイル パスに従い、イメージへのリンクが作成されます (イメージを選択し、プロパティ入力ヘルパーにて URL を選択します)。パスの解決方法については、イメージ URI のセクションを参照ください。

PXF ファイルを保存するためにこの設定を選択する必要があります。

メモ　RTF フォーマットでは EMF、JPG、PNG ファイルでのみイメージの埋め込みがサポートされます。

パス

このタブの設定では、SPS ファイルにより作成される様々なファイルへのデフォルトのパスと、SPS ファイル内部で保存されたパスを指定することができます。



[パス] タブにて指定可能な値を以下に示します:

- SPS ファイルと同一の場所、またはメイン スキーマ ソースの作業用 XML ファイルと同一の場所にプレビュー ファイルを作成する。
- イメージ ファイルやバーコード イメージ ファイル、チェート イメージ ファイルなどの生成されたファイルが作成される場所。
- SPS に含まれるファイルのパスを、SPS ファイルのディレクトリ以下にファイルが含まれる時にだけ相対パスとするか、またはファイルが SPS ファイルのディレクトリ ツリー以外に存在する場合にも相対パスとするのかの指定。

解像度

解像度では、デザイン上でピクセルとして指定された大きさが、画面以外の (印刷媒体の) 長さ単位に変更される時の倍率を指定することができます。コンボ ボックスのドロップダウンから 72dpi、96dpi、120dpi を選択することができます。dpi の値が大きくなるにしたがい、1 ピクセルにより表示される実際の長さが小さくなります。ピクセルからポイントへの変換比率は、それぞれ 1.0、0.75、0.6 となります。詳細については ピクセルの解像度を参照ください。

Authentic

以下のAuthentic プロパティが利用可能です:

- **Authentic のデフォルト貼り付けモード**: クリップボードへコピーされた内容が Authentic View へデフォルトでどのように貼り付けられるのかを選択します。マークアップを含む内容が XML として貼り付けられると、マークアップとともに貼り付けが行われます。そうでない場合、マークアップ無しでノードのテキスト コンテンツだけが貼り付けられます。Authentic View では、挿入ポイントにて右クリックして、コンテキスト メニューから [**貼り付けの形式 | XML**] または [**貼り付けの形式 | テキスト**] を選択することにより、デフォルトの設定とは異なる形式の貼り付けを行うこともできます。デフォルトの貼り付けモードは、SPS ドキュメントの編集中いつでも変更することができます。

  Authentic View にテキストがドラッグ アンド ドロップされた時に、テキストの貼り付け方式をユーザーが選択するか、デフォルトの貼り付けモードが使用されるのかを選択することができます。貼り付けのモードをユーザーの選択に委ねる場合、[**ドラッグ アンド ドロップ時に尋ねる**] ラジオ ボタンを選択し、ユーザーへの通知なしにデフォルトの貼り付けモードを使用するには、[**貼り付けにデフォルト モードを使用**] ラジオ ボタンを選択してください。

- **Authentic スクリプト**: SPS とともに保存されたスクリプトに対するオプションで、(i) スクリプトを有効にするかどうか、(ii) スクリプト内にあるイベントを処理または無効化するか、(iii) スクリプト内にあるマクロを StyleVision 外 (つまりその他の Altova 製品に搭載された Authentic View) からデバッグ モードで実行できるようにするかを選択することができます。StyleVision 環境の外部でマクロの試験やデバッグを行う際に、この機能を使用することができます。

- **テンプレートの呼び出し階層上限**: Authentic View 出力にて再帰的に行えるテンプレートの最大呼び出し回数を指定します。テンプレート呼び出しの回数が、ここで指定された値を超えると、エラーが返されます。この設定により、終わりのないループを回避することができます。

関連項目

- ツール | オプション
- ピクセルの解像度

16.4.11　印刷プレビュー、印刷

[**印刷プレビュー**] コマンド はデザイン ビューならびにAuthentic View にて有効になります (Authentic View はEnterprise ならびにProfessional エディションでサポートされます)。印刷プレビュー コマンドにより、(デザイン ビューがアクティブの場合は) SPS デザイン、または (Authentic View がアクティブの場合は) 作業用 XML ファイルのAuthentic View のプレビューが含まれるウィンドウが表示されます。プレビューは、画面上に表示されている内容にしたがい、タグなしで、またはタグとともに表示されます。

印刷プレビュー ウィンドウでは、ページ上部にあるツール バー、またはページ下部にあるナビゲーション アイコンから、以下の操作を行うことができます。印刷プレビュー ツール バーにあるコマンドを左から順に以下に示します。

- 印刷ボタンを使用して、ページを印刷する。
- ページの方向を垂直または水平から選択する。

- [**ページ設定**] ボタンをクリックすることで、ページ設定ダイアログを表示する。
- ヘッダーとフッターを印刷対象に含める/対象から外す。
- ページの幅、またはページの高さが画面全体に収まるように表示を切り替える。
- 画面上に表示されるページ数を指定する。
- ズーム倍率を選択することで、表示されているプレビュー ページの倍率係数を指定する。

プレビューのページ ナビゲーションを行うには、プレビュー画面の下部にあるページ ナビゲーション ボタンを使用するか、ページ テキスト ボックスにページ番号を入力してください。

Authentic View ならびに出力プレビュータブでは が有効になります。選択されたビューの作業用 XML ファイルが、ビューに対して指定されたページのセットアップにしたがって印刷されます。Authentic View のページ セットアップは印刷プレビュー ウィンドウからアクセスすることのできる [**ページ設定**] ダイアログにて行うことができます。

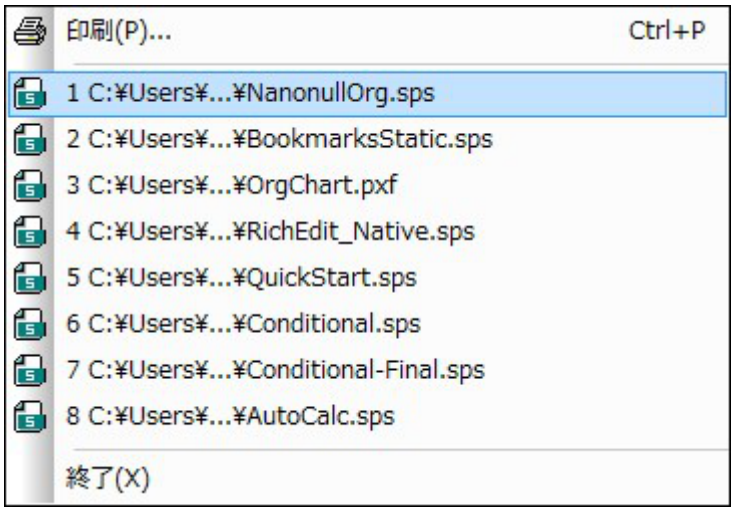
メモ 背景色やイメージを印刷プレビューに表示するには、(i) Internet Explorer の [**ツール**] メニュー以下にある [**インターネット オプション**] を選択し、[**詳細設定**] タブをクリックし、(ii) 設定ボックスの [**印刷**] グループ以下にある [**背景の色とイメージを印刷する**] チェック ボックスにチェックを入れ、(iii) [**OK**] をクリックします。

関連項目

- ファイル| プロパティ

16.4.12 最後に開かれたファイル、終了

最後に使用されたファイルのリストが、ファイル名やパス情報とともに表示されます。エントリーの 1 つをクリックすることで、そのファイルが新たなタブにてメイン ウィンドウに開かれます。



キーボードの Alt+F により [**ファイル**] メニューを開き、対応する数値キーを押下することで、そのファイルを開くことができます。例えば、1 を押下することで、リストの先頭にあるファイルが開かれ、2 を押下することで、2 番目のファイルが開かれます。

終了] コマンドによりStyleVision を終了します。保存されていないファイルが開かれている場合、これらのファイルを保存するか尋ねられます。

関連項目

- メインウィンドウ

16.5 編集メニュー

[編集] メニューには SPS とAuthentic View ドキュメンの編集をサポートするためのコマンドが収められています。[切り取り] (Ctrl+X) や[コピー] (Ctrl+C)、[貼り付け] (Ctrl+V)、[削除] (Del) といった標準的な編集コマンドの他にも、以下のコマンドを利用することができます:

- 元に戻す、やり直し、すべて選択 : 編集内容を元に戻す、元に戻した内容をやり直す、SPS のコンテンツすべてを選択します。
- 検索、次を検索、置き換: SPS や、Authentic View、とXSLT スタイルシート プレビュー内にあるテキストの検索、Authentic View 内にあるテキストの置き換えを行います。
- スタイルシートのパラメーター: SPS に対してグローバルに宣言されたパラメーターの編集をします。
- マークアップの縮退/展開: SPS デザイン コンポーネント タグを縮退/展開します。

コマンドへは、コンポーネントやカーソルの挿入位置を右クリックすることで表示されるコンテキスト メニューからもアクセスできます。更にいくつかのコマンドは、キーボードのショートカットやツールバーのアイコンからも利用できます。開かれているドキュメントのビューやカーソルの位置により利用できないコマンドはグレーアウトされます。

関連項目

- ツールバー

16.5.1 元に戻す、やり直し、すべて選択

[元に戻す] (Ctrl+Z) コマンド により、編集した内容を元に戻すことができます。元に戻す操作で使用できる回数に上限はなく、行ったすべての操作はドキュメントが開かれた状態まで元に戻すことができます。

[やり直し] (Ctrl+Y) コマンド により、それまで元に戻された操作をやり直すことができます。[元に戻す] と[やり直し] コマンドを使用することで、コマンドの履歴を行き来することができます。

[すべて選択] コマンドにより、デザインドキュメント ウィンドウにあるすべてのコンテンツが選択されます。

関連項目

- ツールバー

16.5.2 検索、次を検索、置き換え

検索 (Ctrl+F) コマンドにより デザインビュー、JavaScript エディター、Authentic View、とHTML-のための-XSLT とRTF-のための-XSLT スタイルシート内で用語または用語の一部の検索が可能になります

デザインビュー、HTML プレビュー、とAuthentic View

検索 コマンドをデザインビュー、HTML プレビュー またはAuthentic View をクリックすると検索と置換 サイドバーが有効化されます。

以下の点に注意してください:

- デザインビューでは静的なデータが検索されますが、ノード名は検索されません。
- Authentic View 内では静的な (XSLT) 入力からのテキストではなく動的なデータ (XML データ) が検索されます。
- エントリを単語全体と一致するには、[単語の完全マッチ]をチェックしてください。例えば、soft のエントリは用語 `soft` のみを検索します。例えば `software` 内 `soft` のは検索されません。
- 用語の一部を一致させるには、[単語の完全マッチ]チェックボックスのチェックを解除してください。これにより例えば `soft` と `software` を入力することができます。
- 大文字と小文字を区別して検索するには、[大文字と小文字を区別]チェックボックスのチェックを解除します。これにより `soft` のエントリで `Soft` を検索できるようになります。

XSLT-for-HTML、XSLT-for-RTF、 とJavaScript Editor

XSLT-for-HTML、XSLT-for-RTF または JavaScript エディタータブ内の **検索** コマンドをクリックすると以下のダイアログが表示されます:

以下のオプションから選択することができます:

- *大文字と小文字を区別する* 切り替えられると、大文字と小文字を区別する検索がオンに切り替えられます (`Address` は `address` とは異なります)。
- *単語単位で検索*: テキスト内の文字のみが一致されます。例えば、入力文字列 `fit` に対して、 *単語の完全マッチ* がオンにされていると、単語 `fit` のみが検索文字列に一致します。 in `fitness` 内の `fit` は一致しません。
- *正規 表現* オンに切り替えられると、検索用語は、正規表現として読み取られます。下の正規表現の使用方法の説明に関しては *正規表現* を参照してください。
- *アンカーの検索*検索用語が入力されると、ドキュメント内の一致がハイライトされ、一致の内の1つが現在の選択としてマークされます。*アンカーの検索* の切り替えは、最初の選択がカーソルの位置に対して相対的かを決定します。*アンカーの検索* がオンに切り替えられると、現在選択されている一致が選択され、現在のカーソルの場所の次の一致に一致します。*アンカーの検索* がオフに切り替えられていると、現在選択されている一致はドキュメントの最初から数えて最初の一致が一致します。
- *選択範囲内での検索*: オンに切り替えられると、現在のテキストの選択範囲をロックし検索を選択されている範囲に制限します。それ以外の場合、ドキュメント全体が検索されます。テキストの新しい範囲を選択する前に、*選択範囲内の検索* オプションをオフに切り替えて現在の選択範囲のロックを解除します。

次を検索コマンド

次を検索 (F3) コマンド はリクエストされたテキストの次の発生を検索するために最後に使用された検索コマンドを繰り返します。検索機能の使用方法に関しては 検索 を参照してください。

置換 (Ctrl+H)

デザインビュー、JavaScript エディター、および Authentic View (*not は Basic エディションではサポートされていません*) 内で **置換** コマンドを有効化することができ、テキスト文字列を検索し、他のテキスト文字列と置き換えることができます。

正規表現の使用

正規表現 (regex) を使用して、テキスト文字列を検索することができます。これを行うには、最初に *正規表現* オプションをオンに切り替えます (*上記参照*)。これは検索用語フィールド内のテキストが正規表現として評価されるように指定します。次に正規表現と検索フィールドに

入力します。正規表現の作成をヘルプするために、検索用語フィールドの右にある **正規表現 ビルダー** ボタンをクリックします。これにより、検索用語フィールド内のテキストが正規表現として評価されます。メタ文字の以下の短い説明を参照してください。

正規表現メタ文字
正規表現 メタ文字のリストは以下の通りです。

.	任意の文字を一致する。これは単一の文字のプレースホルダーです。
\(	一致をタグした範囲の開始をマークする。
\)	一致をタグした範囲の終了をマークする。
\(abc**\)**	**\(** と**\)** メタ文字は、タグされた式の開始と終了をマークします。一致する箇所を、後で参照する(バックレファレンス)目的のためにタグ(「記録」)する場合、役に立つ可能性があります。9つまでの式をタグ付けすることができます(検索、または、置換フィールド内で後に参照することができます)。 例えば、**\(the\) \1** は文字列 the に一致します。この式は、以下のように説明することができます: 前に一致しているタグ付けされた箇所へバックレファレンスが後に続く、スペース文字が後に続く文字列「the」に一致させる(そしてタグ付けされた箇所として記録する)。
\n	n が 1 から 9 までの整数の値を持つことができる **変数** の場合。式は、置き換える際にタグ付けされた1番目から9番目までの箇所を参照することができます。例えば、検索文字列が Fred**\([1-9]\)**XXX で、置換文字列が Sam**\1**YYY の場合、検索文字列内に、明示的に番号1とインデックスされている1つのタグされた式が存在することを意味し、また、置換文字列内では、タグ付けされた式は、**\1** と参照されています。検索と置換式が Fred2XXX に対して適用されている場合、Sam2YYY が生成されます。
\<	単語の先頭に一致。
\>	単語の末尾に一致。
\x	このコンテキスト以外では特別な意味を持つ、文字 x が使用できるようになります。例えば、\[は、文字セットの始まりではなく、[として解釈されます。
[...]	文字のセットを示します。例えば、[abc] は、a、b または c のいずれかを意味します。範囲も使用することができます: 例えば、[a-z] をすべての小文字のために使用できます。
[^...]	セット内の文字の補完です。例えば、[^A-Za-z] は、アルファベット文字を除く全ての文字を意味します。
^	行頭に一致。(セットの内部で使用される以外の場合。上を参照してください)。
$	行末に一致。例: A+$ が1つ以上の行末の A を検索する場合。
*	0 回以上一致。例えば、Sa*m は、Sm、Sam、Saam、Saaam に一致するなど。
+	1 回以上一致。例えば、Sa+m は、Sam、Saam、Saaam に一致するなど。

特別文字の表記
以下の式に注意してください。

\r	キャリッジリターン(CR)。CR (**\r**) または LF (**\n**) を使用して新規のラインを検索または作成します。
\n	改行(LF)。CR (**\r**) または LF (**\n**) を使用して新規のラインを検索または作成します。
\t	タブ文字
****	正規表現内に表示される文字をエスケープするために使用してください。例: **\\\n**

正規表現サンプル

このサンプルは、正規表現を使用してテキストを検索し置き換える方法について説明しています。多くの場合、テキストの検索と置換えは複雑ではなく、正規表現を必要としません。しかしながら、通常の検索と置換え操作ではテキストを操作できないインスタンスが存在する場合が

あります。例えば、数千行からなるXMLファイル内の1つのオペレーション内の特定の要素を、囲まれたコンテンツに影響を与えずに、名前を書き換える必要がある場合が挙げられます。もう1つの例は、要素の複数の属性の順序を変更する場合です。この場合は、手動で行う際に必要とされる手間を省くため、正規表現の使用が役に立ちます。

サンプル1: 要素の名前変更

サンプルXMLコードリストには、ブックのリストが含まれています。このサンプルの目的を、の各ブックの `<Category>` 要素を `<Genre>` と置き換えることとします。この目的を達成するために正規表現を使用します。

```
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="books.xsd">
   <book id="1">
      <author>Mark Twain</author>
      <title>The Adventures of Tom Sawyer</title>
      <category>Fiction</category>
      <year>1876</year>
   </book>
   <book id="2">
      <author>Franz Kafka</author>
      <title>The Metamorphosis</title>
      <category>Fiction</category>
      <year>1912</year>
   </book>
   <book id="3">
      <author>Herman Melville</author>
      <title>Moby Dick</title>
      <category>Fiction</category>
      <year>1851</year>
   </book>
</books>
```

必要条件を満たすには、以下のステップを踏んでください：

1. 検索と置換ダイアログボックスを開くには、**Ctrl+H** を押します。
2. **正規表現の使用** .* をクリックします。
3. 検索フィールドに次のテキストを入力します: `<category>(.+)</category>` 。この正規表現は、すべての `category` 要素に一致し、ハイライトされます。

(既知ではない)各要素の内部のテキストに一致するには、タグ付けされた条件式 `(.+)` が使用されています。タグ付けされた条件式 `(.+)` は、「1つまたは複数の文字 `.+` の発生をマッチし、このマッチを記録します。」を意味します。次のステップで表示されているように、後でタグ付けされた条件式を参照する必要があります。

4. 置換フィールド内に、次のテキストを入力します: `<genre>\1</genre>` 。この正規表現置換えのテキストを定義します。検索フィールドから既にタグ付けされた条件式にバックレファレンスするために `\1` が使用されていることに注意してください。すなわち、このコンテキストでは、`\1` 「現在一致する<category> 要素の内部のテキスト」を意味します。
5. **全てを置き換える** クリックして、結果を確認します。すべての `category` 要素が、このサンプルの目的である `genre` に名前を変更されました。

サンプル2: 属性の順序を変更する

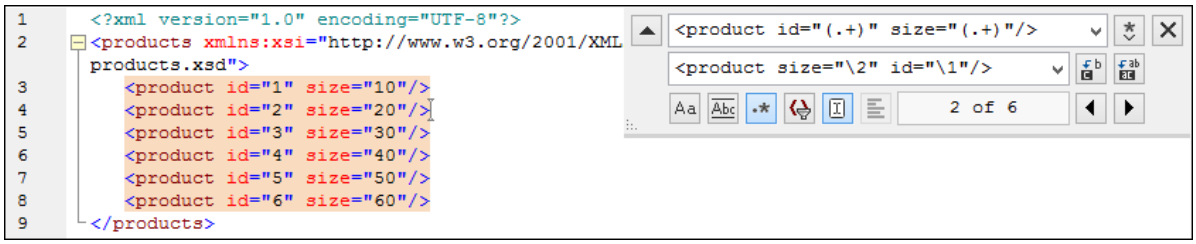
下のサンプルXML コードリストには製品のリストが含まれています。各製品には以下の2つの属性が存在します: `id` と `size`。各 `product` 要素内の `id` と `size` 属性の順序の変更をこのサンプルの目的とします。(すなわち、`size` 属性が `id` の前に来るように変更します)。この条件を満たすために正規表現を使用することができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<products xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="products.xsd">
   <product id="1" size="10"/>
   <product id="2" size="20"/>
   <product id="3" size="30"/>
   <product id="4" size="40"/>
   <product id="5" size="50"/>
   <product id="6" size="60"/>
</products>
```

必要条件を満たすには、以下のステップを踏んでください:

1. 検索と置換ダイアログボックスを開くには、**Ctrl+H** を押します。
2. **正規表現の使用** をクリックします。
3. 検索フィールドに次のテキストを入力します: `<product id="(.+)" size="(.+)"/>` 。この正規表現は、XML ドキュメント内の製品要素に一致します。(既知ではない)各属性の値に一致するためには、タグ付けされた条件式 `(.+)` が2度使用されていることに注意してください。タグ付けされた条件式 `(.+)` は、各属性の(`.+` である文字の1度、または複数回の発生である)値に一致します。

4. 検索フィールドに次のテキストを入力します: `<product size="\2" id="\1"/>` 。この正規表現には、一致した製品の要素のテキストの代替テキストが含まれています。`\1` と`\2` の2つの参照が使用されていることに注意してください。これらは、検証フィールドからタグ付けされた条件式に対応しています。すなわち、`\1` は、「属性 id の値」を意味し、`\2` は「属性 size の値」を意味します。



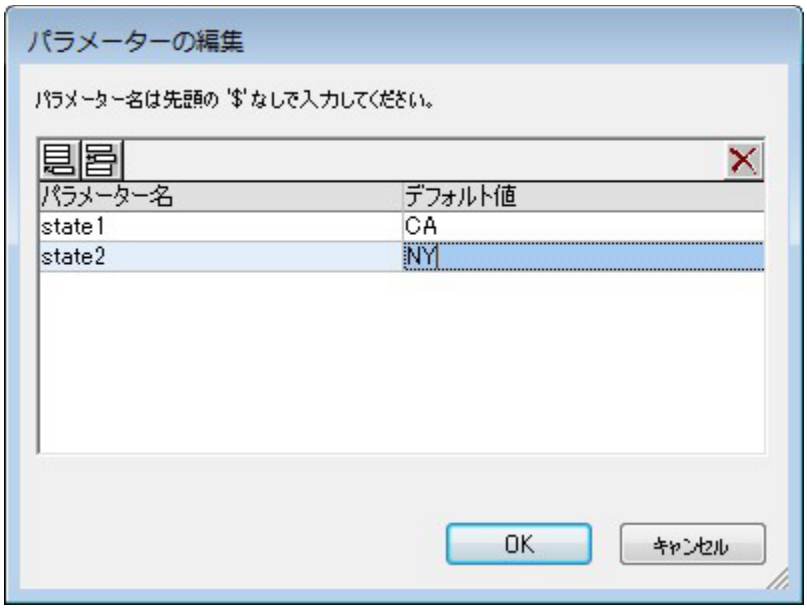
6. **全てを置き換える** をクリックして、結果を確認します。すべての product 要素は属性 size が属性の id 前に置かれるように更新されました。

関連項目

- ツールバー

16.5.3 スタイルシートのパラメーター

スタイルシートのパラメーター コマンド により、パラメーターと、そのデフォルト値を宣言ならびに編集することができます。コマンドはデザインビューならびにAuthentic View で利用することができます。このコマンドを選択すると、以下に示されるパラメーターの編集ダイアログが表示されます。



以下の点に留意してください:

- スタイルシート全体 と DB フィルターにて使用されるパラメーターを挿入、追加、削除することができます。

- パラメーター名はアルファベット文字により開始する必要があり、パラメーター名にはアルファベットと数値、そしてアンダースコアを使用することができます。
- パラメーターの編集ダイアログには SPS 内にあるすべてのユーザー定義パラメーターが含まれます。
- パラメーターはデザイン概要サイドバーでも宣言することができます。

関連項目

- パラメータ
- DB フィルター: DB データのフィルタリング

16.5.4 マークアップの縮退/展開

[**マークアップの縮退/展開**] コマンドは選択されたタグを縮退または展開するための切り替えコマンドです。コマンドはあらゆる種類のノードや定義済みのフォーマット、SPS メカニズムに対して使用することができます。タグを縮退/展開するには、そのタグをダブルクリックしてください。展開された終了タグをダブルクリックすることでも、タグは縮退されます。

一連のタグが縮退される様子を以下のスクリーンショットに示します。縮退されたタグをダブルクリックすることで、タグが展開されます。

タグを縮退することで、編集のニーズに従い最適な表示を行うことができます。

関連項目

- シンボル
- テンプレートフィルター

16.6 プロジェクトメニュー

[プロジェクト] メニューでは、プロジェクトの作成、構成、そして修正を行うことができます（以下のスクリーンショットを参照）。プロジェクトのセットアップやプロジェクト内に含まれるファイルの指定、各ファイルの種類にしたがって個別のフォルダーにて管理するといった操作を行うことができます。プロジェクトはプロジェクト サイドバーにて視覚的に表示され、SPS デザインで使用するためファイルへのアクセスも簡単に行うことができます。

[プロジェクト] メニューには以下のコマンドが含まれています。各コマンドの詳細については、このセクションのサブセクションにて説明されます：

- 新規プロジェクトの作成: 新規プロジェクトを作成します。
- プロジェクトを開く: 既存のプロジェクトを開きます。
- プロジェクトの再ロード: プロジェクト サイドバーに表示されたプロジェクトを再ロードします。
- プロジェクトを閉じる: プロジェクト サイドバーに表示されているプロジェクトを閉じます。
- プロジェクトの保存: プロジェクトの保存と名前付けを行います。
- プロジェクトにファイルを追加: プロジェクト サイドバーに表示されているプロジェクトにファイルを追加します。
- プロジェクトにグローバルリソースを追加: プロジェクト サイドバーに表示されているプロジェクトにAltova グローバルリソースを追加します。
- プロジェクトにURL を追加: プロジェクト サイドバーに表示されているプロジェクトへ URL を介してファイルを追加します。
- プロジェクトにアクティブなファイルを追加: 現在アクティブな SPS ファイルをプロジェクトへ追加します。
- プロジェクトにアクティブならびに関係するファイルを追加: 現在アクティブな SPS ファイルと、関係するファイルを追加します。
- プロジェクトにプロジェクト フォルダーを追加: プロジェクト サイドバーに表示されているプロジェクトへフォルダーを追加します。
- プロジェクトに外部フォルダーを追加: プロジェクト サイドバーのプロジェクトへローカルフォルダーを追加します。

(C) 2015-2021 Altova GmbH

- プロジェクトにWeb フォルダーを追加: プロジェクト サイドバーのプロジェクトへURL を介してフォルダーを追加します。

最近使用したプロジェクト

プロジェクト メニューの下部には、最後に使用されたプロジェクトの名前が最大 9 件まで表示され、これらのファイルへ素早くアクセスすることができます。

ドラッグ アンド ドロップ

プロジェクト ウィンドウでは、あるフォルダーを別のフォルダーや同一フォルダーにある別の場所へドラッグすることができます。ファイルを他のフォルダーへドラッグすることはできますが、同一フォルダー内で移動することはできません（フォルダー内のファイルはアルファベット順に並び替えられます）。更に、ファイルやフォルダーをWindows ファイル エクスプローラーからプロジェクト サイドバーへドラッグすることもできます。

関連項目

- StyleVision におけるプロジェクト
- プロジェクト サイドバー

16.6.1　新規プロジェクトの作成、プロジェクトを開く、プロジェクトの再ロード

新規プロジェクトの作成 コマンド によりプロジェクトを新たに作成することができます。プロジェクト サイドバーに既にプロジェクトが開かれている場合、そのプロジェクトが新たに作成されたプロジェクトにより置き換えられます。開かれているプロジェクトに保存されていない変更点が含まれる場合、変更をプロジェクトに保存するか尋ねられます。[**新規プロジェクトの作成**] コマンドでは新たなプロジェクトが作成されるだけで、プロジェクトを保存するには、プロジェクトの保存 コマンドを明示的に使用する必要があります。

プロジェクトを開く コマンド により、既存のプロジェクトが開かれ、プロジェクト サイドバーに表示されます。プロジェクト サイドバーに既にプロジェクトが開かれている場合、そのプロジェクトが新たに作成されたプロジェクトにより置き換えられます。開かれているプロジェクトに保存されていない変更点が含まれる場合、プロジェクトを置き換える前に、変更をプロジェクトに保存するか尋ねられます。

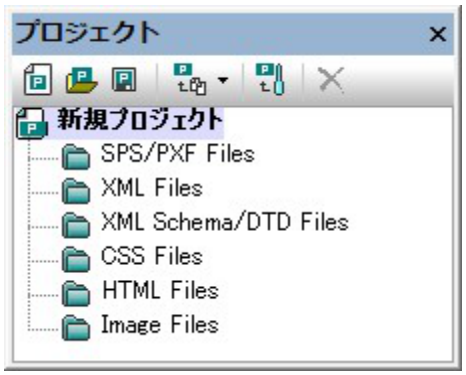
プロジェクトの再ロード コマンド により、現在開かれているプロジェクトを再ロードします。このコマンドは、複数のユーザーが同時に作業するような環境で、別の人によりプロジェクトが変更された時に利用することができます。

関連項目

- StyleVision におけるプロジェクト
- プロジェクト サイドバー

16.6.2　プロジェクトを閉じる、プロジェクトの保存

[**プロジェクトを閉じる**] コマンドにより、アクティブなプロジェクトが閉じられます。プロジェクト内部に保存されていない変更点が存在する場合、プロジェクトを閉じる前に変更するか尋ねられます。保存されていない変更を含んでいるプロジェクトには、プロジェクト サイドバーに表示されたプロジェクト名の隣にアスタリスクが表示されます。



[プロジェクトの保存] コマンドにより、現在開かれているプロジェクトが保存されます。プロジェクトの名前は最初に保存される際に指定されることに注意してください。ファイル名の変更は、StyleVision 外部でしか行えず、例えばWindows ファイルエクスプローラーによりファイルの名前変更を行うことになります。

関連項目

- StyleVision におけるプロジェクト
- プロジェクトサイドバー

16.6.3 ファイル/グローバル リソース/URL をプロジェクトに追加

プロジェクトにファイルを追加

[プロジェクトにファイルを追加] コマンドにより、アクティブなプロジェクトに対してファイルを追加します。コマンドを選択するとファイルを開くダイアログが表示されるので、プロジェクトに追加するファイル、またはファイルのグループを指定してください。選択されたファイルが各サブフォルダーにて定義されたファイル拡張子にしたがい、プロジェクト フォルダーのサブフォルダー内部に追加されます。同じ種類の拡張子が複数のフォルダーに対して定義されている場合、そのファイル拡張子にマッチした(プロジェクト サイドバーにある) 最初のフォルダーへファイルが追加されます。

メイン プロジェクト フォルダー内部にあるフォルダーやサブフォルダーへファイルを追加するには、そのフォルダーを右クリックして、[ファイルを追加] コマンドを選択し、目的のファイルを選択してください。

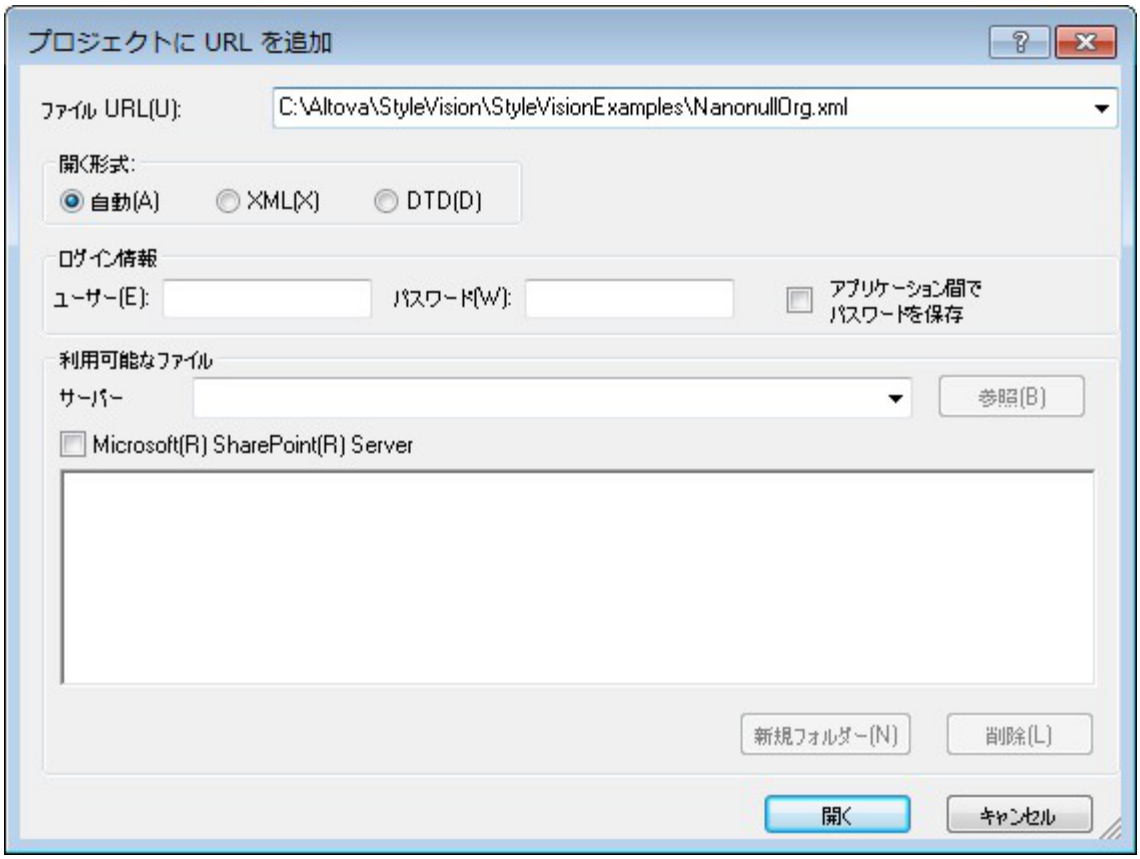
プロジェクトにグローバルリソースを追加

[プロジェクトにグローバル リソースを追加] コマンドにより、現在アクティブなグローバルリソースXML ファイル内にあるグローバルリソースを示すダイアログが表示され、これらリソースの中からアクティブなプロジェクトへ追加する対象を選択することができます。目的のグローバルリソースを選択して、[OK] をクリックしてください。

プロジェクトにURL を追加

[プロジェクトに URL を追加] コマンドにより、現在のプロジェクトへURL が追加されます。プロジェクト内にURL を含めることで、その URL により指し示されるオブジェクトがプロジェクトへ追加されます。[URL をプロジェクトに追加] ダイアログが表示されます(以下のスクリーンショットを参照)。

(C) 2015-2021 Altova GmbH



ファイル URL (file:¥ プロトコルとともに、またはプロトコルなしで) やサーバー URL を入力することができます。サーバー URL の場合、ユーザー名とパスワードを入力して、サーバー URL の入力を行なってください。[参照] をクリックすることで、[利用可能なファイル] に表示されるリストから目的のファイルを選択することができます。

URL はフォルダーやメイン プロジェクト フォルダーのサブフォルダーへ追加することもできます。プロジェクト フォルダーを右クリックして、[URL を追加] コマンドを選択してください。[URL をプロジェクトに追加] ダイアログが表示されます。

ドラッグ アンド ドロップ

ファイルを他のフォルダーへドラッグすることはできますが、同一フォルダー内で移動することはできません (フォルダー内のファイルはアルファベット順に並び替えられます)。更に、ファイルやフォルダーを Windows ファイル エクスプローラーからプロジェクト サイドバーへドラッグすることもできます。

ファイル、リソース、URL の削除

ファイル、Altova リソース、または URL を削除するには、そのオブジェクトをプロジェクト サイドバーで右クリックして、コンテキスト メニューから [削除] コマンドを選択してください。

関連項目

- StyleVision におけるプロジェクト
- プロジェクト サイドバー

16.6.4 アクティブなファイル（ならびに関連するファイル）をプロジェクトに追加

プロジェクトにアクティブなファイルを追加

[**プロジェクトにアクティブなファイルを追加**] コマンドにより、アクティブな SPS ファイルが現在のプロジェクトへ追加されます。このファイルは、.sps ファイル拡張子が定義されている最初のフォルダーへ追加されます。SPS だけではなく、関連付けられたスキーマや作業用 XML ファイル、テンプレート XML、CSS、イメージファイルなども追加したい場合、（以下に記される）[**プロジェクトにアクティブならびに関係するファイルを追加**] コマンドを使用してください。アクティブなファイルをメイン プロジェクト内にあるフォルダーやサブフォルダーへ追加するには、そのフォルダーを右クリックして [**アクティブなファイルを追加**] コマンドを選択してください。

プロジェクトにアクティブならびに関係するファイルを追加

[**プロジェクトにアクティブならびに関係するファイルを追加**] コマンドにより、現在アクティブな SPS ファイルと関連付けられたスキーマファイルや、存在する場合には作業用 XML ファイル、テンプレート XML、CSS、イメージファイルが、それぞれのファイル拡張子に関連付けられた最初のフォルダーへ追加されます。アクティブなファイルと関係するファイルをメイン プロジェクト フォルダー内にあるフォルダーやサブフォルダーへ追加するには、目的のフォルダーを右クリックして [**アクティブなファイルと関連ファイルを追加**] コマンドを選択してください。

ファイルの削除

ファイルを削除するは、プロジェクト サイドバーにてそのファイルを右クリックして、コンテキスト メニューから [**削除**] を選択してください。

関連項目

- StyleVision におけるプロジェクト
- プロジェクト サイドバー

16.6.5 プロジェクト フォルダーや外部フォルダーをプロジェクトに追加

プロジェクトにプロジェクト フォルダーを追加

[**プロジェクトにプロジェクト フォルダーを追加**] コマンドにより、新たなフォルダーが現在のプロジェクトへ追加されます。コマンドを選択するとプロパティ ダイアログが表示され、そのフォルダーに関連付けられたファイル拡張子を入力することができます（ファイル拡張子はセミコロンにより区切ることができます）。そのフォルダーに関連付けられたファイル拡張子のファイルがプロジェクトに追加されると、そのフォルダーへファイルが自動的に追加されます。新たに追加されたプロジェクト フォルダーは、プロジェクト サイドバー内にあるプロジェクト フォルダーのリストへ追加されます。

プロジェクト フォルダーのサブフォルダーを作成するには、サブフォルダーを作成するフォルダーを右クリックして、表示されるコンテキスト メニューから [**プロジェクト フォルダーを追加**] コマンドを選択します。プロパティ ダイアログにてフォルダー名とフォルダーのファイル拡張子を入力してください。

(C) 2015-2021 Altova GmbH

プロジェクトに外部フォルダーを追加

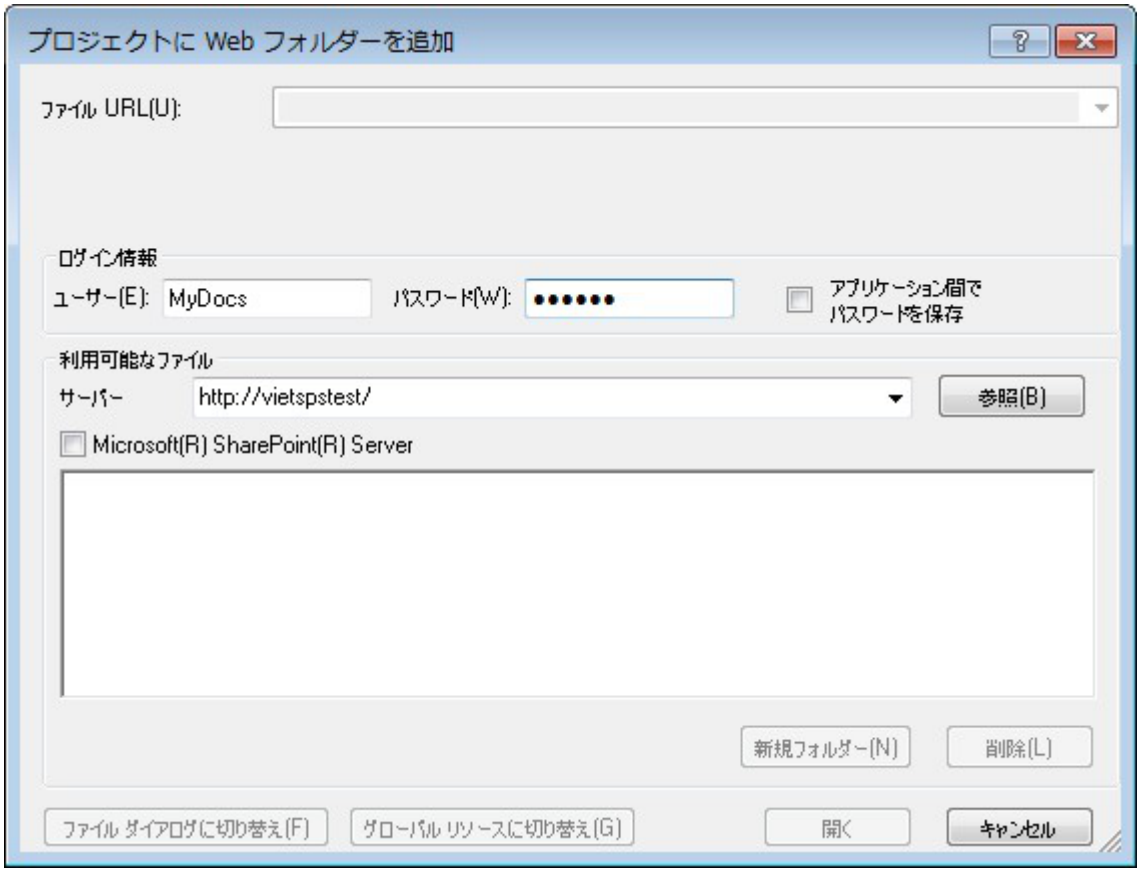
[**プロジェクトに外部フォルダーを追加**] コマンドにより、外部フォルダーが現在のプロジェクトへ追加されます。このコマンドによりローカルまたはネットワーク上にあるフォルダーと、内部に含まれるすべてのコンテンツが現在のプロジェクトへ追加されます。追加された外部フォルダーは展開または縮退することができます。外部フォルダーをプロジェクト フォルダーのサブフォルダーとして追加するには、プロジェクト フォルダーを右クリックして、コンテキスト メニューから [**外部フォルダーを追加**] コマンドを選択してください。

プロジェクトにWeb フォルダーを追加

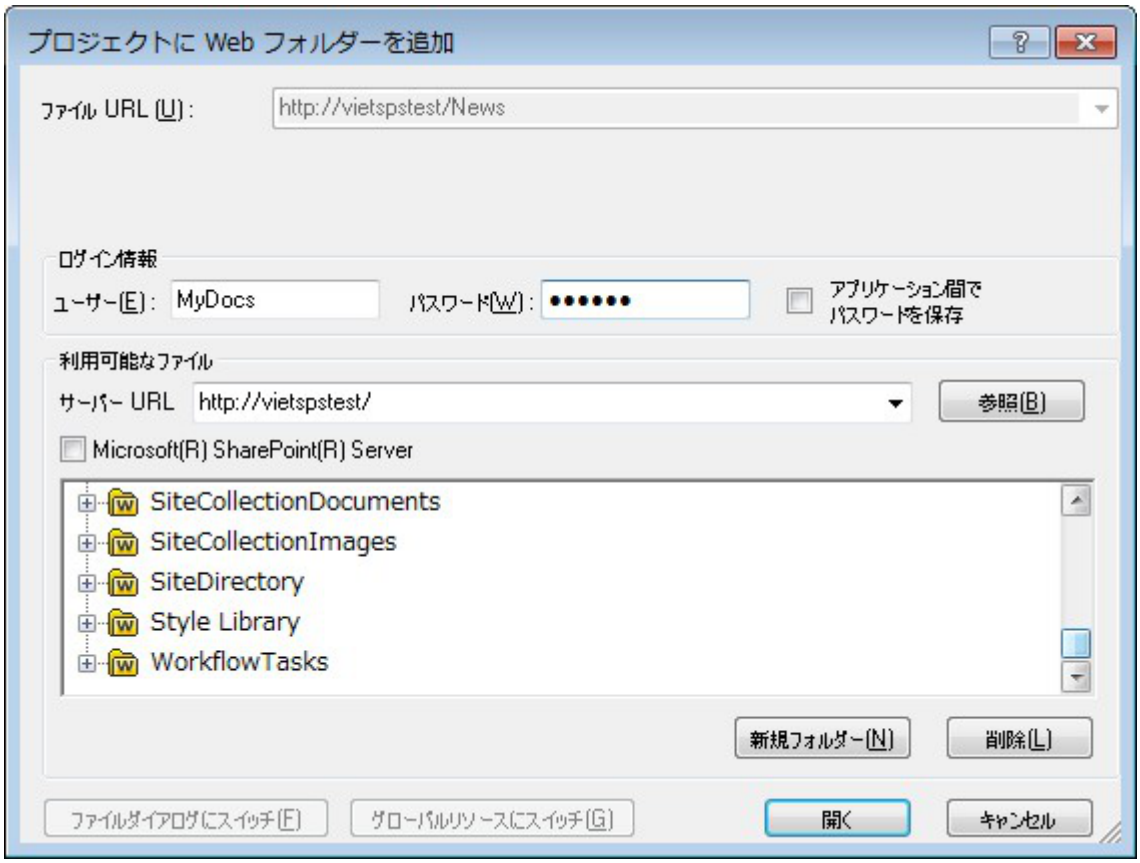
[**プロジェクトに外部 Web フォルダーを追加**] コマンドにより、URL を介して新規外部フォルダーが現在のプロジェクトへ追加されます。追加された外部フォルダーは展開または縮退することができます。外部 Web フォルダーをプロジェクト フォルダーのサブフォルダーとして追加するには、プロジェクト フォルダーを右クリックして、コンテキスト メニューから [**外部 Web フォルダーを追加**] コマンドを選択してください。

コマンドをクリックすると、[プロジェクトにWeb フォルダーを追加] ダイアログが表示されます（以下のスクリーンショットを参照）。以下の操作を行なってください：

1. [サーバー URL] フィールドをクリックして、サーバー URL を入力し、ログイン ID を [ユーザー] ならびに [パスワード] に入力します。



2. [**参照**] をクリックしてサーバーへ接続すると、利用可能なフォルダーが表示されます。



3. プロジェクトに追加するフォルダーをクリックすると、[OK] ボタンが選択可能になります。フォルダー名とサーバー URL がファイル URL フィールドに表示されます。
4. [OK] をクリックするとフォルダーがプロジェクトへ追加されます。
5. プラス アイコンをクリックすると、フォルダーのコンテンツを確認できます。
6. Web フォルダーに対して表示されるファイルの種類を定義するには、そのフォルダーを右クリックしてコンテキスト メニューから [**プロパティ**] を選択し、目的のファイル拡張子を入力してください。

ドラッグ アンド ドロップ

ファイルを他のフォルダーへドラッグすることはできますが、同一フォルダー内で移動することはできません。ファイルやフォルダーを W indows ファイル エクスプローラーからプロジェクト サイド バーへドラッグすることもできます。

フォルダーの削除

ファイルを削除するは、プロジェクト サイド バーにてそのファイルを右クリックして、コンテキスト メニューから [**削除**] を選択してください。

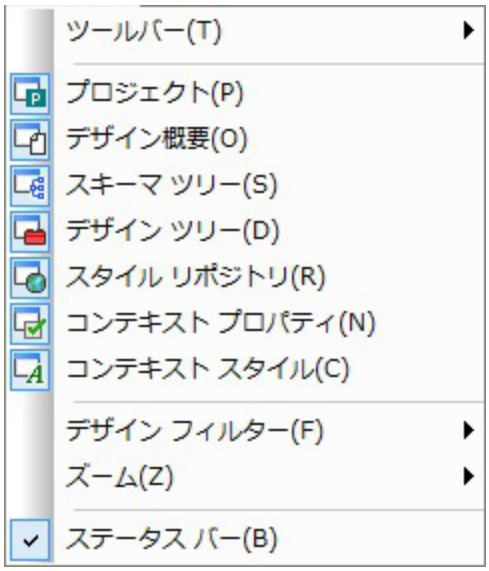
関連項目

- StyleVision におけるプロジェクト
- プロジェクト サイド バー

(C) 2015-2021 Altova GmbH

16.7 表示メニュー

[表示] メニューには、GUI の外観を変更し、GUI コンポーネントの表示を切り替えるためのコマンドが含まれています（以下のスクリーンショットを参照）。個々のツールバー、デザイン サイドバー、デザイン フィルター、ステータス バーの表示を切り替えることができます。



関連項目

- ユーザーインターフェイス
- ツールバー

16.7.1 ツールバーとステータスバー

[ツールバー] アイテム上部にカーソルを配置すると、複数のツールバーの表示を切り替えるためのサブメニューが表示されます（以下のスクリーンショットを参照）。

有効になっているツールバーが表示されます。上のスクリーンショットでは、すべてのツールバーが表示されています。ツールバーをクリックすることで、対応するツールバーの表示を切り替えることができます。ツールバーの詳細については、次のセクションを参照してください：レファレンス| ツールバー。

ステータスバー

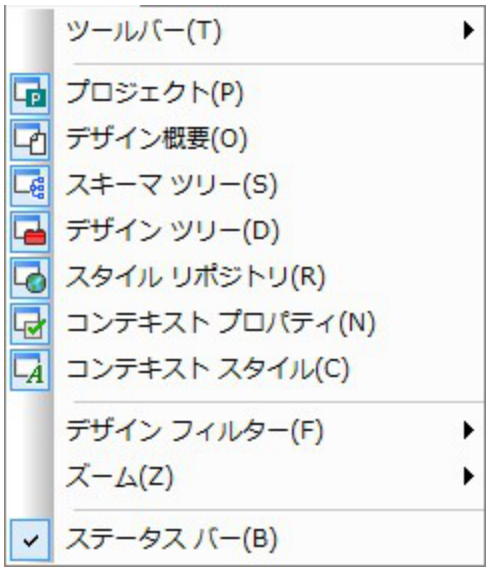
アプリケーション ウィンドウの下部に配置されるステータス バーの表示は、[ステータス バー] コマンドにより切り替えることができます。

関連項目

- ツールバー
- ツール| カスタマイズ

16.7.2 デザインサイドバー

[表示] メニューコマンドには各サイドバーの表示を切り替えるためのコマンドが含まれています（以下のスクリーンショットを参照）。



サイドバーが有効になっている（コマンドのアイコンにフレームがついている）場合、そのサイドバーがGUI に表示されます。必要に応じてサイドバーの表示を切り替えてください。このコマンドは、非表示状態のサイドバーを再度表示するのにも使用することができます。指定されたサイドバーの設定はビュー（デザインビュー、Authentic View、Authentic View、出力ビュー）に関連付けられており、あるビューにて行われた変更は、新たに変更されるまで、そのビューに対して保持されます。

関連項目

- ユーザーインターフェイス| デザインサイドバー

(C) 2015-2021 Altova GmbH

16.7.3 デザインフィルター、ズーム

デザインフィルター

[デザイン フィルター] メニューアイテムでは、デザイン ビューに表示されるテンプレートをフィルタリングするためのコマンドがサブメニューに収められています。デザインが非常に長い、または複数のテンプレートが含まれているような状況で利用できます。デザイン フィルター メカニズムにより、表示するテンプレートの種類を指定することができます。以下のフィルター オプションが利用できます:

アイコン	コマンド	説明
	選択されたテンプレートだけを表示	選択されたテンプレートだけを表示します。テンプレート内にカーソルを配置してこのアイコンをクリックすることで、そのテンプレートだけが表示されます。
	全種類のテンプレートを表示	SPS 内にあるすべてのテンプレート（メイン、グローバル、名前付き、レイアウト）が表示されます。
	インポートされたテンプレートを表示	インポートされたテンプレートの表示を有効/無効にします。
	メイン テンプレートを表示/隠す	メイン テンプレートの表示を有効/無効にします。
	グローバル テンプレートを表示/隠す	グローバル テンプレートの表示を有効/無効にします。
	デザイン フラグメントを表示/隠す	デザイン フラグメントの表示を有効/無効にします。

これらのコマンドは デザインフィルター ツール バーのツール バー アイコンとして利用することもできます。

ズーム

[ズーム] コマンドにより拡大率をサブメニューから選択することができます。（標準ツール バーにある）ズーム コンボ ボックスから、拡大や縮小、ズームの倍率を指定することができるほか、Ctrl キーとマウスのスクロール ホイールにより、拡大/縮小を行うこともできます。

関連項目

- ツール バー | デザインフィルター

16.8 挿入メニュー

[**挿入**] メニューには、様々なデザイン コンポーネントをSPS へ挿入するためのコマンドが含まれています。これらコマンドのいくつかは ツールバーアイコン として利用することができます。更に、[**挿入**] メニュー コマンドはSPS デザインを右クリックすることで表示されるコンテキスト メニューからも利用できます。コンテキスト メニューでは、SPSのカーソル位置に挿入できないコマンドはグレー アウトされます。

メモ [**挿入**] コマンドはSPS の構築に使用されるため、これらのコマンドはデザイン ビューでのみ利用できます。

関連項目

- ユーザーインターフェイス
- ツールバー

関連項目:

コンテンツの編集の手順 :[**挿入**] メニューにより挿入することのできる様々なコンポーネントの使用方法について詳細に記述されます。
ツールバー: 個々のツールバーについて詳細に記述されます。
タグで囲むメニュー: 選択対象を囲むかたちでデザイン コンポーネントを付与するコマンドについて記述されます。

16.8.1 コンテンツ

[**コンテンツ**] コマンドにより、カーソル位置に(コンテンツ) プレースホルダが挿入されます。(コンテンツ) プレースホルダは要素ノードと属性ノード内部に挿入することができ、そのノード以下にあるすべての子が処理されることになります。

- そのノードが要素ノードの場合、ノードの子要素ノードとテキスト ノードが処理されます。子要素ノードの処理にはグローバルテンプレートが使用され、それ以外の場合、要素に対する内蔵のテンプレート ルールが使用されます。テキスト ノードの処理を行うには、テキスト ノードに対する内蔵のテンプレート ルールを使用します。これにより、要素に対する内蔵のテンプレート ルールからすべての子孫テキスト ノードにおけるテキストが出力されます。属性の親要素や祖先要素に対して定義されているグローバルテンプレートが存在しない限り、または(コンテンツ) プレースホルダやその他のコンテンツレンダリングにより明示的に属性の出力が指定されていないかぎり、(コンテンツ) プレースホルダでは、属性の値が出力されないということに注意してください。
- そのノードが属性ノードの場合、属性の子テキスト ノードに対する内蔵のテンプレートが使用されます。このテンプレートでは、テキスト ノードのテキストが出力され、属性の値が出力されることになります。

ノードのタグにカーソルを配置して、右クリックにより表示されるコンテキスト メニューから[**挿入 | コンテンツを挿入**] を選択するか、デザイン要素の挿入 ツールバーにある[**コンテンツの挿入**] アイコンをクリックして、要素を挿入するデザイン上の箇所をクリックすることで、(コンテンツ) プレースホルダを挿入することができます。

コンテンツのスタイル

定義済みのフォーマットやスタイル サイドバーにあるプロパティを使用することで、(コンテンツ) プレースホルダのフォーマットを指定することができます。このフォーマットはデザインならびに出力で確認することができるほか、ノードのコンテンツに適用されます。

(C) 2015-2021 Altova GmbH

コンテンツの置換え

スキーマツリーから別のノードが(コンテンツ) プレースホルダを含んでいるノードへドロップされると、既存の(コンテンツ) プレースホルダが新しいノードにより置き換えられます。

コンテンツの削除

(コンテンツ) プレースホルダを選択し、キーボードのDelete キーを押下することで削除することができます。

メモ ノードの (コンテンツ) プレースホルダを削除することで、**空のテンプレート ルール**を作成することができます。空のテンプレート ルールにより、テンプレートが適用されないノード (つまり出力を行わないノード) を定義することができます。

関連項目

- テキストとしてXML コンテンツを挿入
- 出力の構造
- その他のコンテンツ

16.8.2 その他のコンテンツ

[**その他のコンテンツ**] コマンドにより、そのノードに対する(その他のコンテンツ) が挿入されます。このプレースホルダは、そのノードの**使用されていない子ノード**にあるコンテンツを表しており、要素以下で使用されていない要素やテキスト ノードに対して適用されるXSLT のxsl:apply-templates ルールに対応します。テンプレートは子属性に対して適用されない点に注意してください。要素タグ内部を右クリックして、コンテキスト メニューから[**その他のコンテンツを挿入**] を選択することで、(その他のコンテンツ) プレースホルダを挿入することもできます。

(その他のコンテンツ) プレースホルダを使用することで、ある子要素を特定の方法で処理して、その兄弟要素に対してテンプレートを適用することができます。テンプレートを兄弟要素に適用することで、兄弟要素が処理されないという可能性を排除することができます。これにより、ドキュメント階層の下部にある要素を処理することができるようになります。

(その他のコンテンツ) プレースホルダを選択し、キーボードのDelete キーを押下することで削除することができます。

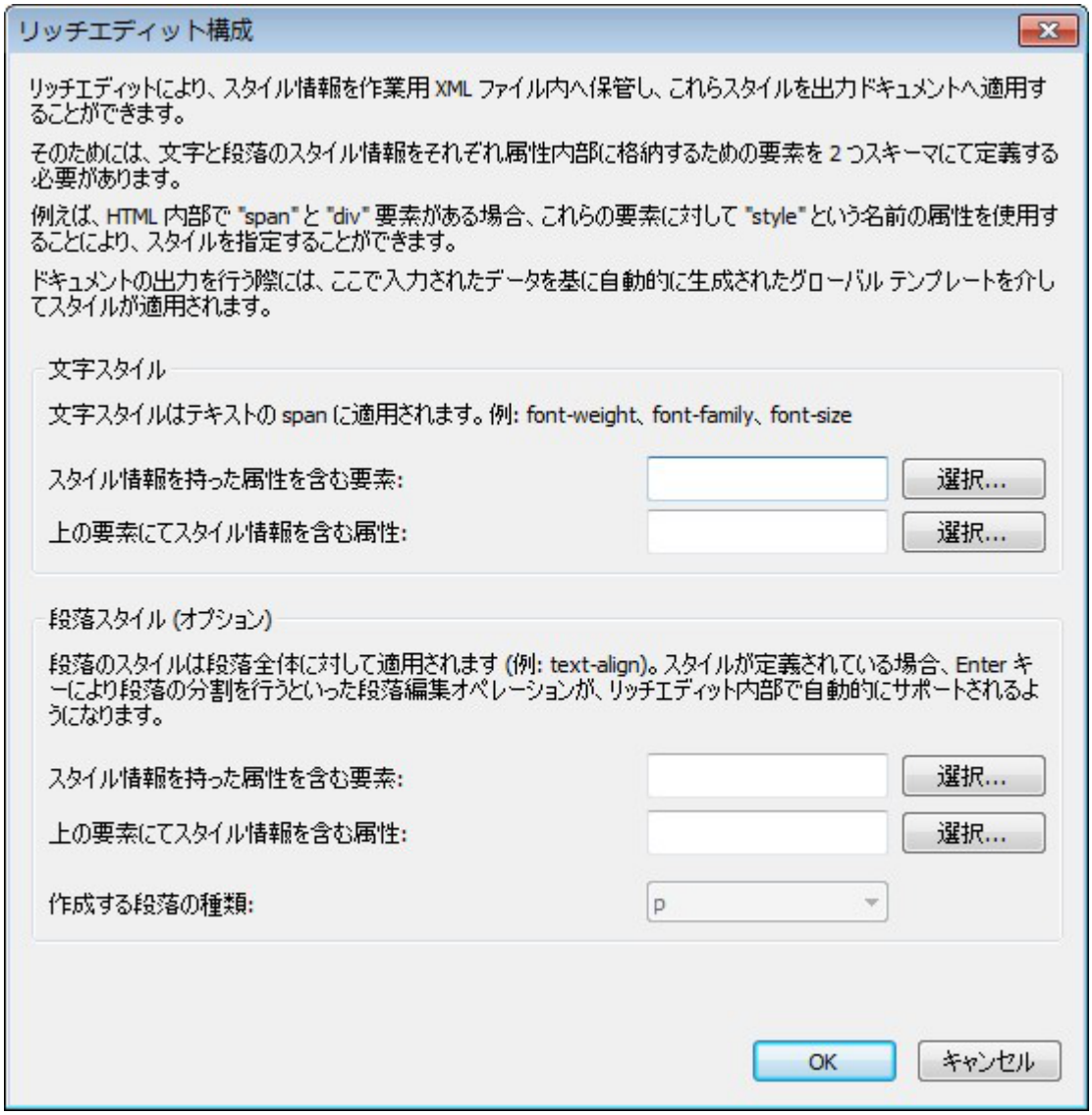
関連項目

- テキストとしてXML コンテンツを挿入
- 出力の構造
- その他のコンテンツ
- コンテンツ

16.8.3 リッチエディット

[**リッチエディット**] コマンドにより、カーソルの位置にリッチエディット コンポーネントが挿入されます。リッチエディット コンポーネントに対して、リッチエディット構成ダイアログが表示されます(以下のスクリーンショットを参照)。リッチエディット構成はドキュメント内にあるすべてのリッチエ

ディット コンポーネントに対して有効になります。リッチエディット コンポーネントが挿入され、これ以降リッチエディット構成ダイアログは表示されません。



リッチエディット構成ダイアログでは、リッチエディット スタイルプロパティを含むスタイル要素とその属性の名前を入力します。目的の要素と属性をスキーマツリーから選択することができます。対応する[選択] ボタンをクリックしてスキーマツリーを開いてください。選択を終えた後に[OK] をクリックすることでリッチエディット コンポーネントが作成され、スタイル要素の名前を含んだ編集不可能なリッチエディット グローバルテンプレートがデザイン上に作成されます(以下のスクリーンショットを参照)。

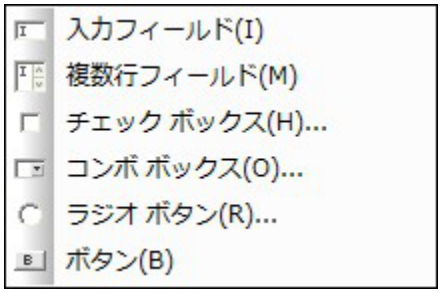
リッチエディットの機能に関する詳細については、Authentic におけるテキスト スタイルの柔軟性を参照ください。

関連項目

- リッチエディット ツールバー
- リッチエディット
- Authentic メニュー

16.8.4　フォームコントロール

[フォーム コントロール] コマンドへマウスを移動すると、各種フォームコントロール(データ入力デバイス)を挿入するためのコマンドが表示されます(以下のスクリーンショットを参照)。



[フォーム コントロール] コマンドへマウスを移動すると、各種フォームコントロールデータ入力デバイスの使用のセクションを参照ください。フォームコントロールを作成すると、そのコントロールを選択し、プロパティ サイドバーにてプロパティの編集を行うことができます。

デザインの挿入ポイントを右クリックして、コンテキスト メニューから [**挿入 | フォーム コントロールの**] を選択するか、デザイン要素の挿入 ツールバーにあるフォームコントロールアイコンをクリックして、デザイン上の位置をクリックすることで、デザイン上にフォームコントロールを挿入することができます。

メモ　CDATA セクション (テキスト ボックスや複数行のテキスト ボックス) を入力フィールドに挿入することはできません。CDATA セクションは、Authentic View に表示される要素内に、テキスト コンテンツ コンポーネントとして表示されます。

関連項目

- データ入力デバイスの使用
- プロパティ

16.8.5 DB コントロール

[DB コントロール] コマンド上にマウスを移動すると、Authentic View ユーザーがAuthentic View のレコード表示やナビゲーション、DB への問い合わせを Authentic View で行うためのコントロールを挿入するためのコマンドが表示されます。これらのコントロールをデザインへ挿入すると、Authentic Viewドキュメントの対応する位置にコントロールが表示されます。

以下にコマンドのリストを示します。

- ナビゲーション
- ナビゲーション + 移動
- クエリボタン

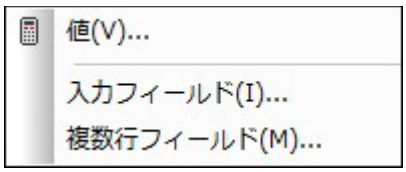
これらコントロールの作成方法や処理方法については、次のセクションを参照してください：DB のためのSPS デザイン機能.

関連項目

- DB のためのSPS デザイン機能
- データベースとの作業

16.8.6 自動計算

[自動計算] を使うことで、XPath 条件式を使った値の計算を行うことができます。自動計算が挿入された箇所に値は表示されます。自動計算はSPS 内部へテキスト、入力フィールド、または複数行の入力フィールドとして挿入されます。自動計算を挿入する箇所にカーソルを配置して、右クリックするか [挿入] メニューコマンドを使用してください。[挿入 | 自動計算] にマウスカーソルを移動すると、どのように自動計算が挿入されるかを選択するためのメニューが表示されます（以下のスクリーンショットを参照）。デザイン要素の挿入 ツールバーにある自動計算アイコンを使用することもできます。

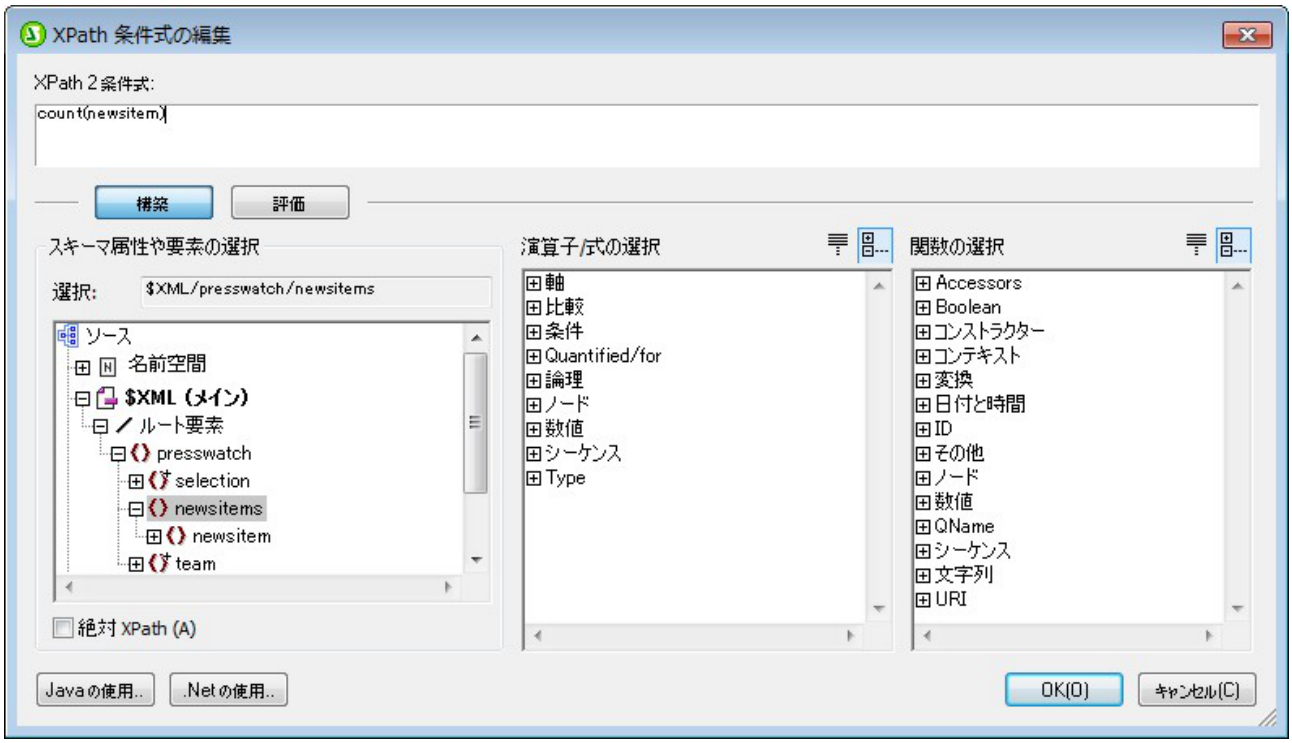


自動計算の値が Authentic View と出力ドキュメントに表示されることになります。

自動計算に対する XPath 条件式

自動計算の挿入方法を選択すると、（以下のスクリーンショットに示される）XPath 式 の編集ダイアログ（*下のスクリーンショット*） ダイアログが表示されます。

(C) 2015-2021 Altova GmbH



構築されている条件式のコンテキスト ノードが、左側に表示されたペイン内のスキーマ ツリーでハイライトされます。XPath 条件式をテキスト ボックスへ直接入力することができるほか、(3 つあるペインのうちいずれかの) アイテムをダブルクリックすることでアイテムを挿入することができます。左側に表示されたペインのスキーマ ツリーから挿入されたノードは、([絶対 XPath] にチェックが入れられていない場合) コンテキスト ノードに対して相対的、または([絶対 XPath] にチェックが入っている場合) 絶対的なものになります。

XPath 条件式を作成すると、[OK] をクリックすることで自動計算が挿入されます。

自動計算によりXML ノードを更新する

XML ドキュメント内にあるノードを自動計算の結果により更新することができます。更新方法については、自動計算によるノードの更新のセクションを参照ください。

関連項目

- XPath 式の編集ダイアログ
- 自動計算

16.8.7　日付選択

[日付選択] コマンドにより、カーソル位置に日付選択が挿入されます。このコマンドは、カーソルが xs:date または xs:dateTime ノード内部にあり、要素が(コンテンツ) または入力フィールドとして作成されている時にのみ有効になります。

関連項目

- 日付選択の使用
- 日付選択の自動追加
- 日付との作業
- 日付のフォーマット

16.8.8 段落、特殊な段落

[段落] コマンド § により、HTML にて段落を表す <p> 要素が選択されたコンポーネントの周りに挿入されます。ノード全体を（タグをクリックすることで）選択、または静的なテキストを選択することで、コンポーネントの選択は行われます。カーソルが静的なテキスト内部に配置されている場合、その場所に段落要素（の開始および終了タグ）が挿入されます。段落は、デザイン要素の挿入 ツール バーにある [段落の挿入] アイコンでも挿入することができます。

[特殊な段落] コマンドにより、選択されたノードに対して定義済みのフォーマットを割り当てることができます。ツール バーのコンボ ボックスから、定義済みのフォーマットを選択することができます。

各種段落にはそれぞれ個別のフォーマット機能が含まれています。pre フォーマットでは、改行が空白スペースに変換されることなく、そのまま出力されることに注意してください。

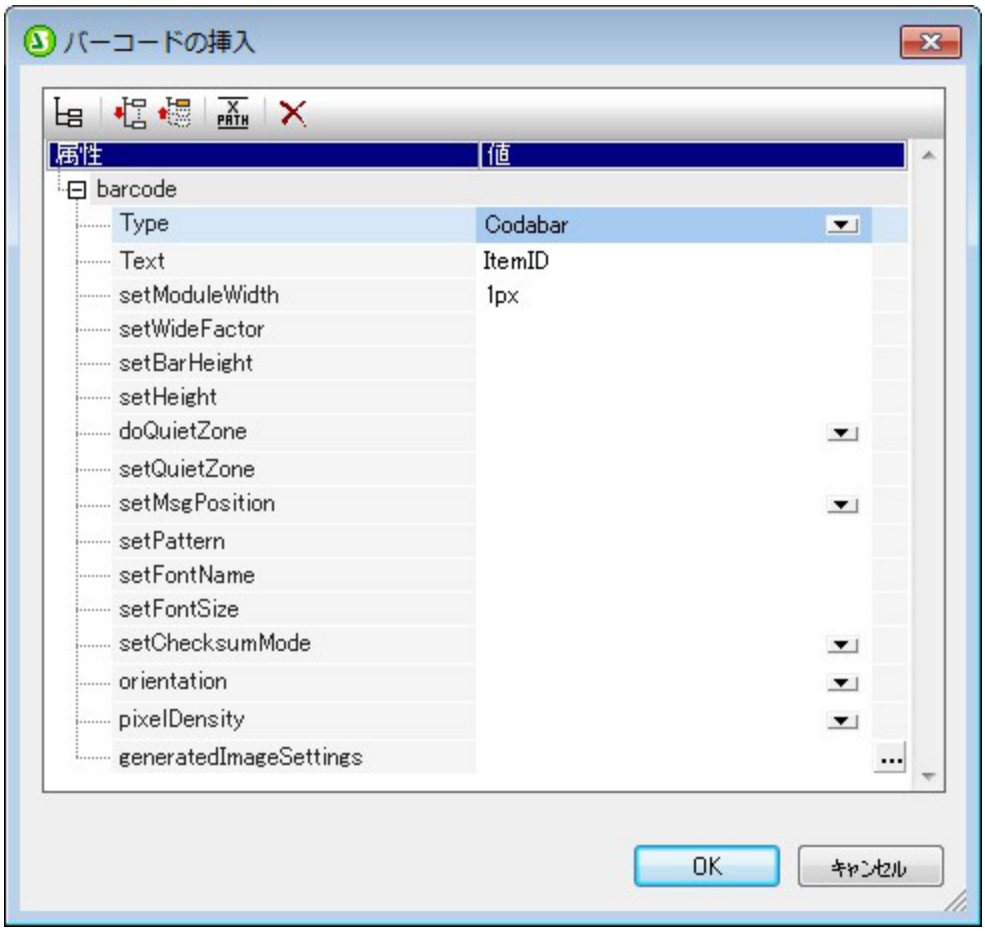
関連項目

- 定義済みのフォーマット
- CSS スタイルと作業

16.8.9 バーコード

[バーコードの挿入] コマンドにより、バーコードの挿入ダイアログが表示されます（以下のスクリーンショットを参照）。

(C) 2015-2021 Altova GmbH



Type と Text のプロパティは必須項目となっており、その他の項目はオプションになっているか、デフォルトの値が与えられています。必須項目と、その他必要な項目に値を入力し、[OK] をクリックしてください。指定されたバーコードがデザインへ挿入されます（以下の例では ISBN が挿入されています）。バーコードのプロパティに関する詳細については、「 バーコードのセクションを参照ください。

重要な情報: バーコードを動作させるには、Java Runtime Environment（JRE）をインストールする必要があります。バージョン 1.4 以降かつお使いのシステムにインストールされている StyleVision のビット バージョン（32 ビットまたは 64 ビット）に対応した JRE が必要です。

関連項目

- バーコード
- SPS ファイル コンテンツ

16.8.10 イメージ

[イメージ] コマンドにより、挿入するイメージを指定するためのイメージの挿入ダイアログが表示されます（以下のスクリーンショットを参照）。デザイン要素の挿入 ツールバーにあるイメージの挿入アイコンをクリックすることでも、イメージの挿入ダイアログが表示されます。

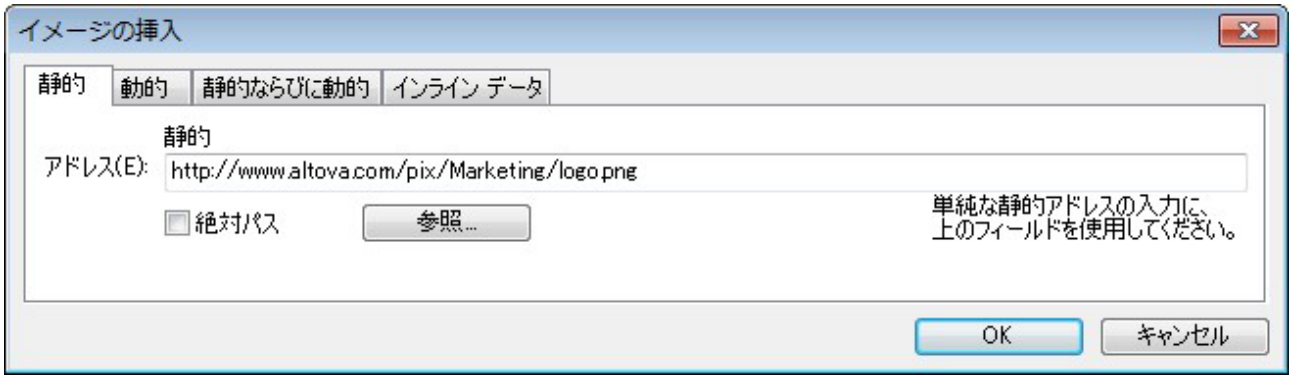
イメージの挿入ダイアログには4 つのタブが含まれており、それぞれ異なる方法によりイメージの場所を指定することができます:

- 静的: イメージ URI を直接入力します。
- 動的: XML ドキュメントから得られた URI を使用、または XPath 条件式により生成されたイメージを使用します。
- 静的ならびに動的: 静的ならびに動的な方法を組み合わせます。
- インライン データ: Base-16 または Base-64 にてエンコードされたとして XML ファイル内部に収められたイメージを選択します。

それぞれのタブの詳細を以下に記します。

静的

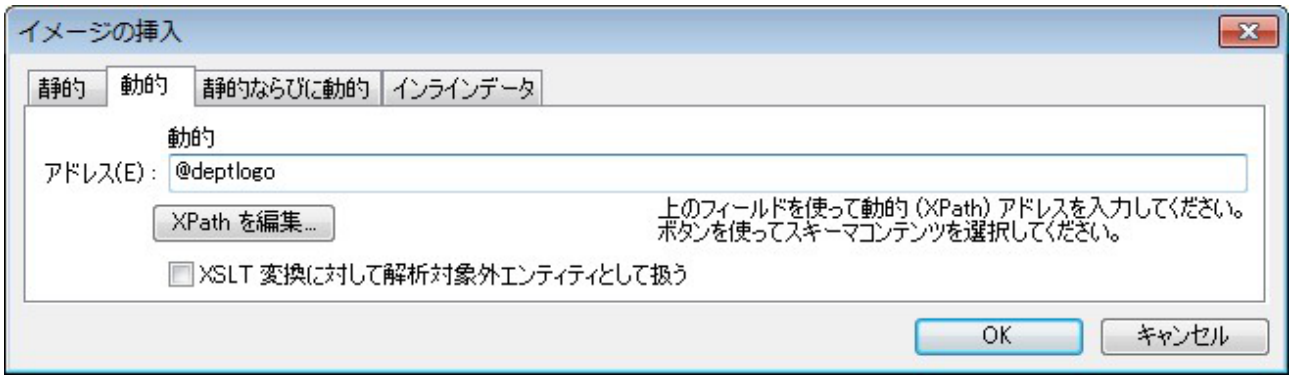
イメージのURI がアドレス フィールドに直接入力されます（以下のスクリーンショットを参照）。以下のスクリーンショットでは、http://www.altova.com/pix/Marketing/logo.png というURI が使用されています。



URI は絶対パスまたは相対パスとして入力することができます（[相対パス] チェック ボックスにより選択）。相対 URI が入力された場合、そのパスはSPS ファイルが配置されている場所に対して相対的になります。[参照] ボタンをクリックして、イメージ ファイルの場所を指定することでも、（絶対または相対パスの）URI を入力することもできます。

動的

XPath 条件式によりイメージのURI が返されます。以下のスクリーンショットでは、@deptlogo というXPath 条件式が使用されています。これはイメージのURI がコンテキスト ノードのdeptlogo 属性内に含まれていることを意味しています。作成されたイメージを含んでいるノードがコンテキスト ノードになります。



[XPath を編集] ボタンをクリックすることで、XPath 式構築が表示されます。ダイアログに含まれるスキーマ ツリーでは、コンテキスト ノードがハイライトされます。

SPS がDTD をベースにしており、解析対象外エンティティを使用している場合、イメージのURI を参照する解析対象外エンティティを使用することができます。[XSLT 変換に対して解析対象外エンティティとして扱う] チェック ボックスにチェックを入れて、解析対象外エンティティが含まれているノードを選択するXPath 条件式を入力してください。解析対象外エンティティの使用方法については、: パースされていないエンティティURI のセクションを参照してください。

静的と動的

静的ならびに動的なメカニズムの両方を使用することでURI を生成します。

コンテキスト ノードにあるdeptname 属性にMarketing という値が含まれる場合、上のスクリーンショットにある設定で作成されるイメージのURI は、http://www.altova.com/pix/Marketing/logo.png となります。動的な部分においては XPath 式構築 を使用することができます。

インラインデータ

イメージはBase-16 またはBase-64 によりエンコードされたテキストとしてXML ファイル内部に含めることができます。イメージの挿入ダイアログに入力されたXPath 条件式により、エンコードされたテキストが含まれるノードが選択されます。[エンコーディング] コンボ ボックスにより、ソースXML にて、StyleVision が使用されているエンコーディングを正確に読み取ることができます。[イメージ フォーマット] コンボ ボックスでは、生成するイメージ ファイルのフォーマットが指定されます（イメージ ファイルはエンコードされたテキスト データから生成され、このファイルが出力ドキュメントにて使用されます）。

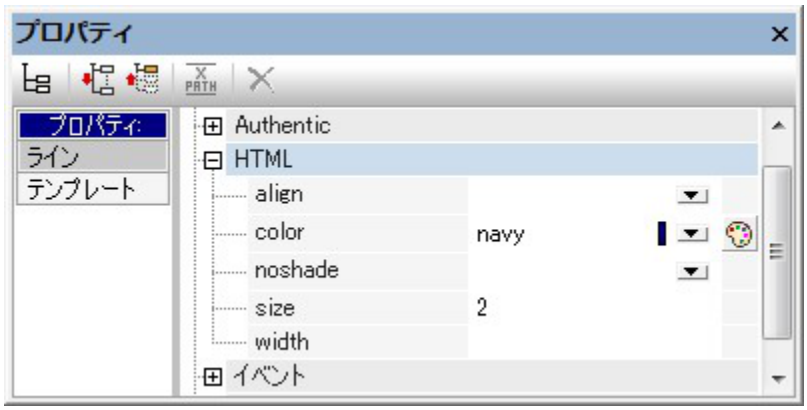
[イメージ ファイル設定] ボタンをクリックすることでアクセスすることのできるイメージ ファイル設定ダイアログでは、作成されるイメージ ファイルに名前を与えることができます。ファイル名を与えないというオプションを選択することもでき、その場合 StyleVision により自動的に名前が生成されます。

関連項目

- イメージの使用

16.8.11 水平線

[水平線] コマンドにより、カーソル挿入位置に水平線が挿入されます。SPS コンポーネントが選択された状態でこのコマンドを選択することはできません。水平線のプロパティをセットするには、デザイン上で水平線を選択し、プロパティ サイドバーのラインを選択した後、HTML グループ内にあるプロパティに値を指定してください。



以下のプロパティを指定することができます: color (色)、size (大きさ)、width (幅)、alignment (配置)、noshade (影なし)。

関連項目

- プロパティ サイドバー
- CSS スタイルと作業

(C) 2015-2021 Altova GmbH

16.8.12　テーブル

[**テーブルの挿入**] コマンドを選択すると、テーブルの作成ダイアログが表示されます（以下のスクリーンショットを参照）。

静的なテーブルと動的なテーブルから、目的のテーブルに対応するボタンをクリックしてください。各種テーブルの操作方法については、静的 SPS テーブルと動的 SPS テーブルのセクションを参照ください。

テーブルは、[**テーブル | テーブルの挿入**] メニューコマンドや、デザイン要素の挿入ツールバーにあるテーブルの挿入アイコン からも作成することができます。

関連項目

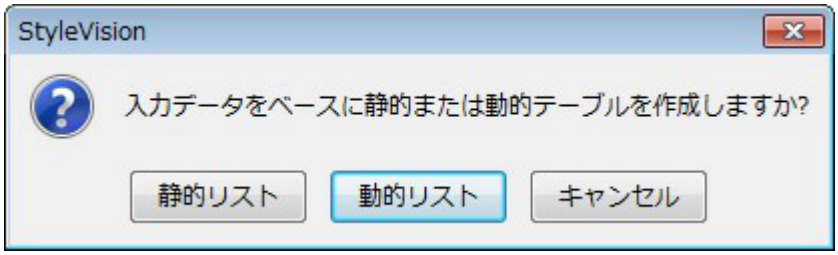
- 静的テーブルの作成
- 動的テーブルの作成
- CALS/HTML テーブル

16.8.13　リスト

[**リストの挿入**] コマンドにより、静的または動的なリストを作成することができます。静的なリストのリスト アイテムがSPS に入力され、動的なリストのアイテムがXML ドキュメントの兄弟ノードから得られます。

以下の操作によりリストを作成します:

1. リストを挿入する箇所にカーソルを配置して、[**リストの挿入**] コマンドを選択します。静的または動的なリストを作成するためのダイアログが表示されます（以下のスクリーンショットを参照）。



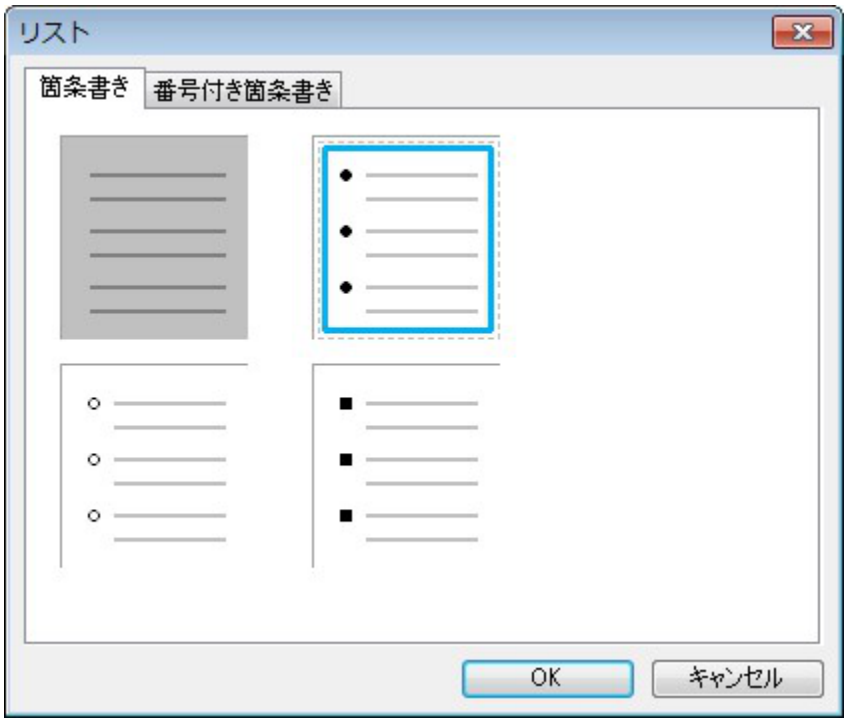
[**静的なリスト**] をクリックすると、ステップ 3 にあるリストの種類を選択するダイアログが表示されます。
[**動的なリスト**] をクリックすると、XPath セレクター ダイアログが表示されます。

2. XPath セレクター ダイアログでは、XPath コンテキストが挿入位置のコンテキストとなっており、このダイアログでは変更できないことに注目してください。動的なリストとして作成されるノードを選択してください。下のスクリーンショットでは、n1:Department がコンテキスト ノードとなっており、n1:Person ノードがリストとして作成されるノードとして選択されています。これにより、n1:Person ノードのコンテンツがリストのアイテムとして作成されます。

ユーザー定義 XPath オプションを選択した場合、動的なテーブルとして作成されるノードを選択する XPath 条件式を入力します。[OK] をクリックすることで、次のステップで説明されるリストのダイアログが表示されます。

3. リスト ダイアログにて、作成するリストの種類を選択します。複数種類の箇条書きマークと、番号を選択することができます。[OK] をクリックすると、選択されたアイテムのマークとともにリストが作成されます。

(C) 2015-2021 Altova GmbH



メモ　リストを挿入する箇所にカーソルが配置されている状態で、デザイン要素の挿入 ツールバーにある[箇条書きリストを挿入]アイコンをクリックすることでも、静的なリストを作成することができます。スキーマツリーにあるノードをデザイン上へドラッグすることでも、動的なリストを作成することができます。

関連項目

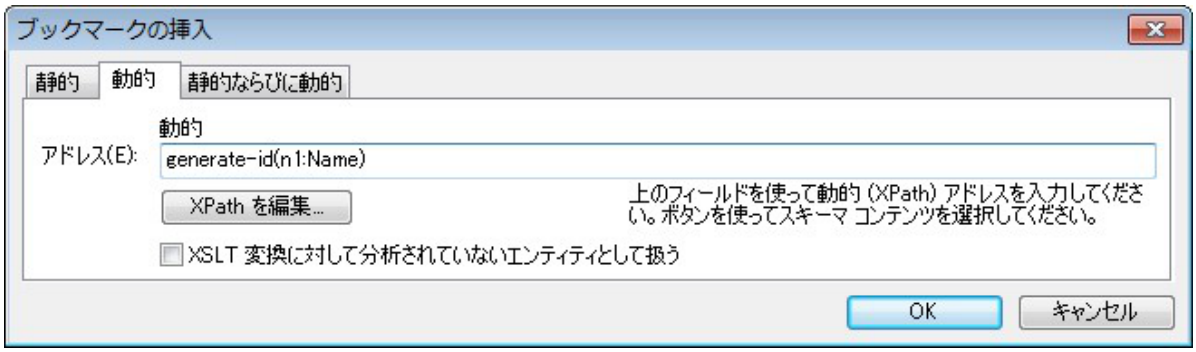
- リストの作成
- タグで囲む | リスト

16.8.14　ブックマーク

[ブックマーク] コマンドにより、SPS 内の任意の場所へブックマーク（またはアンカー）を挿入することができます。ブックマークへはハイパーリンクから参照することができます。

以下の操作によりブックマークを挿入することができます。:

1. ブックマークを作成する位置にカーソルを配置します。
2. [ブックマークの挿入] を選択するか、右クリックで表示されるコンテキスト メニューから[ブックマークの挿入] を選択します。ブックマークの挿入ダイアログが表示されます。



3. 「ブックマークの挿入」ダイアログにて、ブックマークの名前を静的なものにするのか(静的)、XML ドキュメントから動的に取得されるのか(動的)、または静的なパーツと動的なパーツから構成されるのか(静的ならびに動的)により、対応するタブを選択します。上のスクリーンショットでは、コンテキスト ノードの各 Name 子要素に対応したユニークな ID が名前として含まれる動的なブックマークが作成されます。

[OK] をクリックします。ブックマークが定義されます。

メモ デザインにて指定された場所にブックマークは作成されます。繰り返し出現する要素内部が指定された場合、繰り返し出現する各インスタンス要素内部にブックマークが作成されます。静的な名前が与えられた場合、これらブックマーク全てに同じ名前が与えられます。そのため繰り返し出現する要素に対しては、例えばコンテキスト ノード(ブックマークが作成されたノード)の子要素にあるコンテンツを選択するような動的な名前を指定することが望まれます。動的な方法により選択される複数のインスタンスノードに同じコンテンツが含まれる場合、generated-id() 関数により生成される名前により、ブックマーク名の一意性を保証することができます(以下のスクリーンショットを参照)。ブックマークの参照を行うには、同じ ID を ハイパーリンクの href 値として生成することができます。この場合、generate-id() 関数の前にフラグメント識別子の # を使用するようにしてください。XPath 条件式は concat('#', generate-id(nodeXXX)) となります。

ブックマークは作成した後にも、名前を編集することができます。ブックマークを右クリックして、コンテキスト メニューから [**ブックマーク名の編集**] コマンドを選択してください。プロパティ サイドバーのブックマーク グループ以下にあるブックマーク名プロパティからも、ブックマーク名の属性を変更することができます。

ブックマークを削除

ブックマークをデザイン上で選択し、Delete キーを押すことでブックマークを削除することができます。

関連項目

- ブックマークの挿入
- ハイパーリンクの定義

16.8.15 ハイパーリンク

[**ハイパーリンク**] コマンドにより、出力ドキュメント (HTML または RTF) 内にある任意の場所から、出力ドキュメント内にあるアンカー、または、外部ドキュメントやドキュメント フラグメントへのリンクを挿入することができます。 リンクは出力ドキュメントにだけ作成され、Authentic View 内部でのリンク作成は行えないことにご注意してください。

以下の操作によりハイパーリンクを挿入することができます

1. ハイパーリンクは既存のデザイン コンポーネントの周りに作成することができるほか、ドキュメントの任意の場所へ挿入することができます（この場合、その後リンク内部のテキストを挿入することになります）。ハイパーリンクへ変更する SPS コンポーネントやテキスト フラグメントを選択するか、リンクを挿入する箇所へカーソルを移動してください。
2. ツール バーにあるハイパーリンク アイコンをクリックするか、メニューから［**挿入 | ハイパーリンクの挿入**］を選択、または挿入箇所を右クリックして、コンテキスト メニューから［**ハイパーリンクの挿入**］を（またはデザイン コンポーネントが選択されている場合には［**タグで囲む | ハイパーリンク**］）選択します。デザイン要素の挿入 ツール バーにある**ハイパーリンクの挿入** アイコンを使用することでもハイパーリンクを挿入することができます。
3. ハイパーリンク ダイアログの挿入 ダイアログにて、リンク先のドキュメントやドキュメント フラグメントを指定してください。ドキュメント フラグメント（ドキュメント内にあるブックマーク）へのリンクを作成する場合、# シンボルを追加する必要があります。ハイパーリンクの URI は以下の形式で指定することができます:

- 静的なアドレスとして: ［**参照**］ボタンにより選択された HTML ファイル、または［**ブックマーク**］ボタンにより選択されたデザイン ドキュメント内にあるブックマークが直接入力されます。例: http://www.altova.com（静的な Web ページ URI）、U:¥documentation¥index.html（［**参照**］ボタンにより選択）、#top_of_page（［**ブックマーク**］ボタンにより入力）。
- 動的なアドレスとして: XML ドキュメント内にあるノードを指定することで、そのノード内にあるアドレスがリンク先のアドレスとして使用されます。例えば、//otherdocs/doc1 というノードが指定されます。generate-id() 関数によりブックマークの名前が生成されている場合、ハイパーリンクの href も同じ generate-id() 関数により生成することが推奨されます。詳細については「ハイパーリンクの定義を参照してください。
- 静的ならびに動的なテキストを組み合わせてアドレスとする: 静的なテキストと XML ドキュメント ノードを指定します。例: www.altova.com -- department/name -- #intropara。

4. ［**OK**］をクリックすることで、ハイパーリンクが作成されます。

メモ　動的なハイパーリンクのエントリーとしてノードを指定する際には、XPath 条件式の編集ダイアログにある［絶対 XPath］チェック ボックスにチェックを入れることで、絶対 XPath 条件式を入力することができます。チェック ボックスにチェックが入れられていない場合、XPath が挿入されるハイパーリンク内のノードに対して相対的になります。

パースされていないエンティティの使用

DTD をスキーマとして使用している場合、ハイパーリンク アドレスの動的な部分には、DTD 内にある解析対象外実体に対して宣言された URI を使用することができます。解析対象外実体の使用方法の詳細については、次を参照してください: パースされていないエンティティ URI の使用 を参照してください。

ハイパーリンクの編集

ハイパーリンクの作成後には href の値を編集することができます。ハイパーリンクを右クリックして、コンテキスト メニューから［**URL の編集**］を選択してください。プロパティ サイドバーにある［**ハイパーリンク**］グループ以下にある URL 属性からも URL を編集することができます。

ハイパーリンクの削除

ハイパーリンクがデザイン内で選択されている状態で **Delete** キーを押すことで、そのハイパーリンクを削除することができます。

関連項目

- ハイパーリンクの定義
- パースされていないエンティティ URI
- ブックマークの挿入

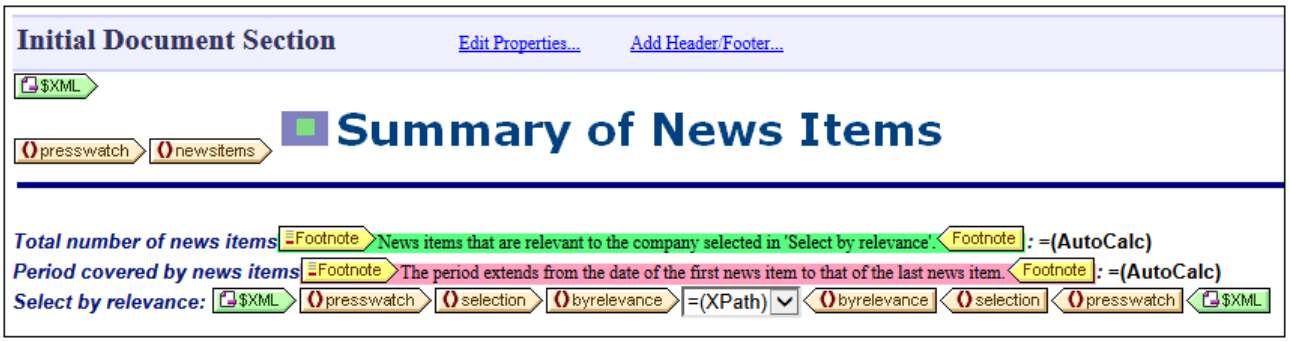
16.8.16 フットノート

フットノートコンポーネントをフットノートの番号を表示したい場所に追加することによりフットノートを挿入することができます（**挿入 | フットノートの挿入**）。フットノートは、ページ番号が与えられ以下の書式で出力することができます（Enterprise Edition では、PDF、RTF、および Word 2007+（**挿入 | フットノートの挿入**）、Professional Edition では、RTF）。

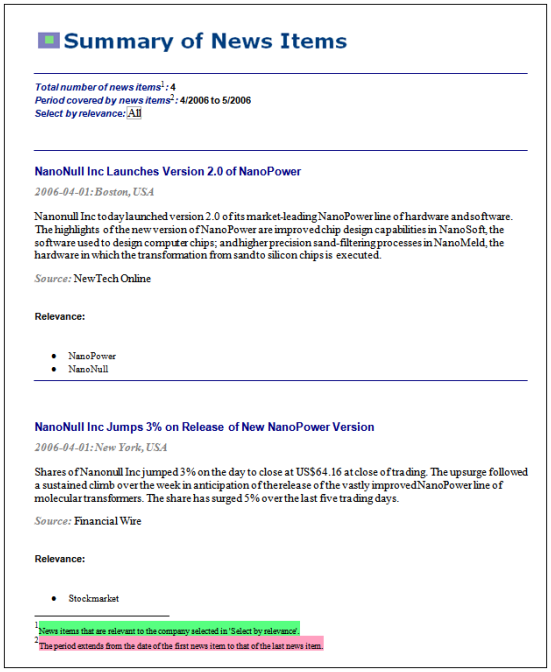
以下の点に注意してください：

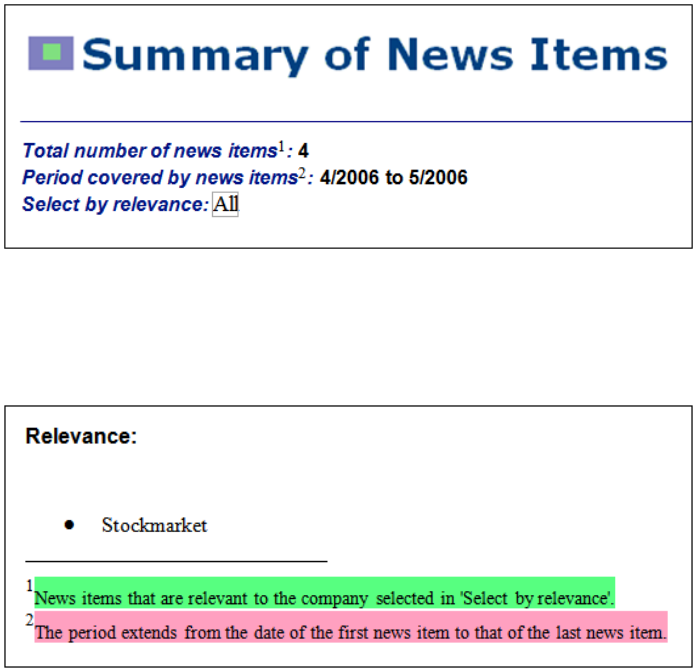
- フットノートのテキストは、フットノートコンポーネントのタグ内におかれる必要があります。フットノートのテキストをフォーマットすることもできます。
- 出力内では、フットノートの番号はフットノートが追加された場所に表示されます。フットノートのテキストは対応するフットノートの番号と共にページの下に表示されます。
- 出力内では、フットノートのテキストがデザイン内のフットノートコンポーネント内のテキストの書式に従いフォーマットされます。
- 出力内では、フットノートは、自動的にドキュメントの終わりまで番号が与えられます。
- In複数の出力ドキュメントの場合、各出力ドキュメントのために番号が与えられます。

下のスクリーンショットでは、2つのフットノートコンポーネント（**挿入 | フットノートの挿入**）が挿入されています。フットノートのテキストはコンポーネントのタグの中に置かれ、テキストはフォーマットされています。



下のスクリーンショットは出力を表示します。左のスクリーンショットはページ全体を表示し、右のスクリーンショットは、フットノートの番号（上）のフットノートのテキスト（下）拡大を表示しています。





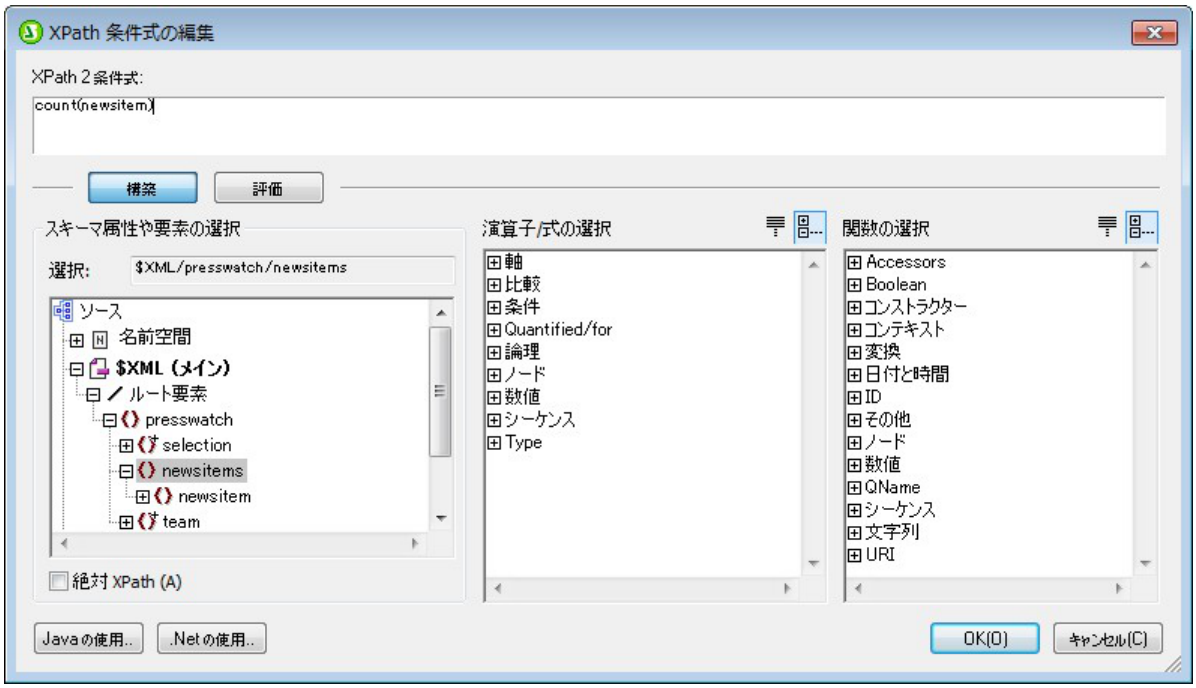
メモ　　フットノートの番号の書式はサポートされていません。

16.8.17　条件、出力ベースの条件

[**条件**] コマンドにより、カーソル位置または選択対象の周りに条件を挿入することができます。条件は 1 つ以上の分岐から構成され、それぞれの分岐内部に処理ルールが含まれることになります。これにより、異なる処理ルールのセットをそれぞれの分岐に対して指定することができます。例えば、ノードのコンテンツが Stop という文字列の場合にはノードのコンテンツを赤色に、Go という文字列の場合には緑色に、どちらでもない場合には黒色で出力されるような指定を行うことができます。

以下の操作により条件を挿入することができます

1. デザイン上の目的の位置にカーソルを配置するか、条件分岐内部に配置するコンポーネントを選択します。
2. メニューコマンドの [**挿入 | 条件**] を選択するか、右クリックにより表示されるコンテキスト メニューから [**挿入 | 条件**] を選択します。
3. 表示される XPath 式 の編集ダイアログ にて XPath 条件式を入力します（以下のスクリーンショットを挿入）。



構築されている条件式で使用されるコンテキスト ノードが、左端のペインにあるスキーマ ツリーにてハイライトされます。テキスト ボックスにXPath 条件式を直接入力することができるほか、ペイン内にあるアイテムをダブルクリックすることで、そのアイテムを挿入することもできます。左端にあるスキーマ ツリーから挿入されたノードは、コンテキスト ノードに対して相対的（[絶対 XPath] チェック ボックスにチェックが入っていない場合）、または絶対的（[絶対 XPath] チェック ボックスにチェックが入っている場合）に挿入されます。

4. [**OK**] をクリックすることで条件の挿入を完了します。入力されたXPath 条件式に対して適用される分岐とともに条件が作成されます。

出力ベース条件

このコマンドにより、カーソル位置または選択対象の周りに出力ベースの条件を挿入することができます。条件の分岐（Authentic View、RTF、またはHTML）が、それぞれ異なる出力を表します。分岐タグ上部にマウスを移動するか（プロパティ サイドバーの条件分岐エントリーにある 編集ボタンをクリックして）選択された分岐のXPath 条件式をチェックすることで、どの分岐がどの出力に対応しているかを確認することができます。出力ベースの条件がカーソル挿入位置に作成されると、分岐がすべて空の条件が作成され、各分岐に対してコンテンツを挿入することになります。出力ベースの条件がコンポーネントの周りに作成されると、選択されていたコンポーネントがすべての分岐内部に含まれることになります。出力ベースの条件に関する詳細については、出力をベースした条件を参照ください。出力ベースの条件は、一般的な条件と同じように編集、移動、削除することができます。

条件分岐の XPath 条件式を編集する

分岐のXPath 条件式を編集するには、デザイン ビューにて分岐を選択し、プロパティ サイドバーから [条件分岐 | 条件トリガー] を選択し、XPath アイテムの編集ボタン ... をクリックしてください。XPath 条件式の編集ダイアログが表示され、条件式を編集することができます（上のスクリーンショットを参照）。

(C) 2015-2021 Altova GmbH

条件分岐の追加、分岐の順序変更、分岐の削除
新規条件分岐の追加、分岐の順序変更、そして分岐を削除するには、目的の分岐を右クリックして、コンテキスト メニューから目的のアイテムを選択してください。

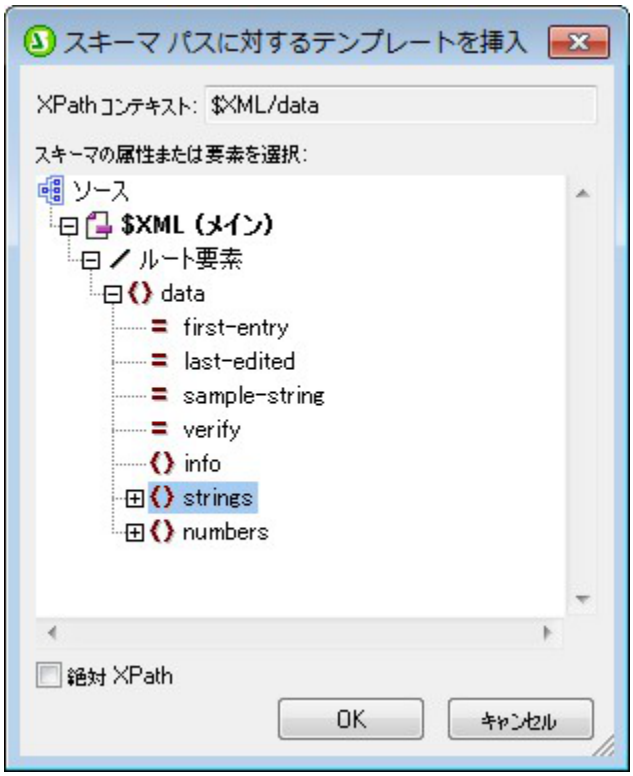
関連項目

- 条件

16.8.18　テンプレート

[テンプレート] コマンドにより、カーソルの挿入位置に選択されたスキーマ ツリー ノードに対する空のテンプレートを挿入します。以下の操作によりテンプレートを挿入できます:

1. テンプレートを挿入するデザイン上の位置にカーソルを配置します。
2. [挿入 | テンプレート] コマンドを選択します。スキーマ パスに対するテンプレートの挿入ダイアログが表示されます (以下のスクリーンショットを参照)。



3. XPath コンテキスト フィールドには、カーソルの挿入位置におけるコンテキスト ノードが表示されており、このノードが作成されるテンプレートのコンテキスト ノードとなります。作成するテンプレートのノードを選択してください。上のスクリーンショットでは、strings ノードが作成されるテンプレートのノードとして選択されています。
4. [OK] をクリックして完了します。

選択されたノードに対する空のテンプレートが作成されます（下のスクリーンショットでは、strings ノードに対して空のテンプレートが作成されています）。

関連項目

- テキストとして XML コンテンツを挿入
- 出力の構造
- デザイン要素の挿入

16.8.19 ユーザー定義テンプレート

[**ユーザー定義テンプレート**] コマンドにより、ユーザーが XPath 条件式により指定したノードを選択する空のテンプレートが作成されます。以下の操作によりユーザー定義のテンプレートを挿入することができます:

1. テンプレートを挿入するデザイン上の位置にカーソルを配置します。
2. **挿入 | ユーザー定義テンプレート** コマンドをクリックすると XPath 式 の編集ダイアログが表示されます。
3. 目的のノードを選択する XPath 条件式を入力します。(i) XPath 条件式はユーザー定義テンプレート内部のノードをコンテキストとして評価され、(ii) XPath 条件式はドキュメントやその他の XML ドキュメント内にあるあらゆるノードを選択することができます。
4. XPath 条件式を入力した後に [OK] をクリックして完了します。

目的のノードに対する空のユーザー定義テンプレートが作成されます。

詳細については、次のセクションを参照してください: SPS ファイル: コンテンツ | ユーザー定義テンプレート.

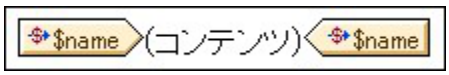
関連項目

- SPS ファイル: コンテンツ | ユーザー定義テンプレート
- デザイン要素の挿入
- 挿入 | テンプレート
- タグで囲む | ユーザー定義テンプレート

(C) 2015-2021 Altova GmbH

16.8.20　変数テンプレート

変数テンプレートとは変数に対して使用されるテンプレートで、通常その値が出力されます。メニューオプションの「**挿入 | 変数テンプレートの挿入**」または「**タグで囲む | 変数テンプレート**」を選択することで、カーソル位置に、SPS で定義される変数のテンプレートが挿入されます。変数テンプレートには、デフォルトでコンテンツプレースホルダが含まれており、これにより変数の値が出力されます（以下のスクリーンショットを参照）。変数テンプレートには、必要に応じてその他の（静的ならびに動的な）コンテンツを含めることができるほか、その他のテンプレートと同様に修正を行うこともできます。



以下の操作により変数テンプレートを挿入することができます:

1. デザイン内のテンプレートを挿入する位置にカーソルを配置します。
2. メニューオプションから「**挿入 | 変数テンプレートの挿入**」を選択します。変数テンプレートの挿入ダイアログが表示されます（以下のスクリーンショットを参照）。

3. ダイアログには、SPS にて定義された ユーザー宣言パラメーターと変数 が表示されます。変数テンプレートに対して追加する変数を選択してください。
4. **OK** をクリックして完了します。

関連項目

- 挿入 | テンプレート
- SPS ファイル: コンテンツ | ユーザー定義テンプレート

16.8.21 デザインフラグメント

[デザイン フラグメント] コマンドには、デザインに現在表示されているすべてのデザイン要素がサブメニューとして含まれています。サブメニューに表示されるデザイン フラグメントをクリックすることで、クリックされたアイテムが挿入位置に挿入されます。

関連項目

- デザインフラグメント

16.8.22 レイアウトコンテナー、レイアウトボックス、ライン

[挿入 | レイアウト コンテナー] コマンドにより、デザイン内部の任意の場所にレイアウト コンテナーを挿入することができます。レイアウト コンテナーにはレイアウト ボックスやラインを挿入することができるほか、これらの挿入コマンドはレイアウト コンテナーの選択中に有効になります。

デザイン要素の挿入 ツールバーに含まれている対応するアイコンにより、レイアウト コンテナー、レイアウト ボックス、そしてラインを挿入することができます。ツールバーアイコンにより挿入を行うには、目的のツールバーアイコンをクリックして、レイアウト アイテムを挿入するデザイン上の位置をクリックしてください。

レイアウト モジュールの詳細な記述や、デザインへの挿入ならびに使用方法については、次のセクションを参照してください：レイアウト モジュール。

関連項目

- レイアウト モジュール
- ツールバー | デザイン要素の挿入

16.8.23 目次

[目次] コマンドにマウスを移動させると、目次（TOC）テンプレート、目次ブックマーク、目次に対するデザインドキュメント構造を作成するためのコマンドが表示されます。

コマンドのリストを以下に示します。リストにある対応するリンクをクリックすることで、対応する目次コンポーネントの使用方法が記述されたセクションにて、それぞれのコマンドの詳細を確認することができます。

- 目次の挿入
- 目次ブックマーク
- 目次ブックマーク（ウィザード）
- 目次参照
- 目次参照 | 入力テキスト/リーダー/ページ参照
- 構造的な番号付け
- 連続的な番号付け
- レベル

(C) 2015-2021 Altova GmbH

- レベル参照
- レベルとして機能するテンプレート

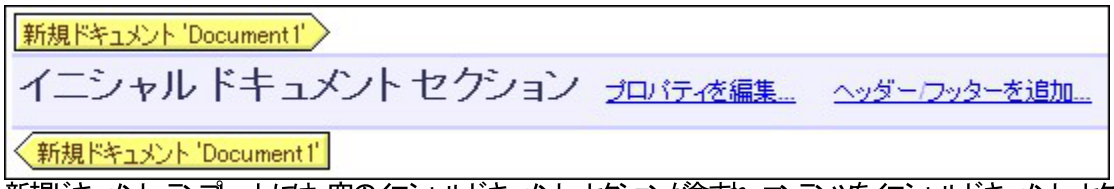
メモ　これらのコマンドは、デザイン内部の適切な箇所を右クリックすることで表示されるコンテキスト メニューからもアクセスすることができます。

関連項目

- 目次（TOC）

16.8.24　新規ドキュメント

[**新規ドキュメントの挿入**] コマンドにより、新規ドキュメント テンプレートがカーソル挿入位置に挿入されます（以下のスクリーンショットを参照）。



新規ドキュメント テンプレートには、空のイニシャルドキュメント セクションが含まれ、コンテンツをイニシャルドキュメント セクションへ入力することができるようになります。必要な場合には、[**挿入 | ページ/カラム/ドキュメント セクションの挿入 | 新規ドキュメント セクション**] コマンドを使用することで、ドキュメント セクションをイニシャルドキュメント セクションへ追加することもできます。

新規ドキュメント テンプレートにより、新たなドキュメントが出力に作成されます。出力は複数の出力ドキュメントから構成されることになります。

複数の出力ドキュメントを処理する方法の詳細については次のセクションを参照してください：複数のドキュメントの出力。

関連項目

- 複数のドキュメントの出力
- 新規ドキュメントテンプレートの挿入
- 新規ドキュメント テンプレートとデザイン構造
- ドキュメント プロパティとスタイル

16.8.25　ページ/カラム/ドキュメント セクション

[**ページ/カラム/ドキュメント セクション**] 以下にあるコマンドにより、ページ媒体出力にてページ区切り（HTML 印刷 とRTF 出力）とページ番号（RTF 出力）。これら項目の挿入はカーソルの挿入位置でのみ行うことができます。

新規ページ

[**新規ページ**] を選択することで、カーソル位置にページ区切りが挿入されます。デザイン ウィンドウ上にページ区切りを表す破線が表示されます。ページの区切りはHTML 出力のブラウザー表示に影響を与えないものの、HTML ファイルが印刷された際にページが区切られます。RTF 出力の場合、指定された場所にページの区切りが挿入されます。

ページ番号

[**ページ番号**] により、RTF出力におけるページ番号が出力されます。ページ番号がドキュメントのどこに挿入されたかにより、ページ番号はブロックとして（独立した行に）またはインラインとして（ドキュメント テキスト内部に）表示されます。例えば、ページ番号が段落要素内部に挿入された場合、段落内部にインラインとしてページ番号が出力されます。それとは別に、例えばページ番号が2 つの要素間に挿入された場合、独立したブロックとしてページ番号が出力されます。

合計ページ

[**ページ合計**] により、ページの合計数がPDF 出力に挿入されます。ページ合計はヘッダーやフッターなどを含む、ドキュメント デザイン内の任意の場所へ挿入することができ、ページの番号付けに利用することができます。例えば、Pages: （ページ番号）/（ページ合計） というデザインにより、Pages: 1/25 といった出力を得ることができます。

新規カラム

セクション内で表示されるカラム（段組） の数は、そのセクションの ページプロパティ で指定することができます。出力では、複数のカラムにより構成されたページに、1 つずつテキストが埋められていくことになります。新規カラムをデザインへ挿入することにより、新たなカラムからテキストが出力されるようになります。ドキュメント内部に新規カラムを挿入するには、新規カラムを挿入する箇所にカーソルが配置されている状態で、コンテキスト メニューからもアクセスすることのできる [**新規カラム**] コマンドを選択してください。

新規ドキュメント セクション

ドキュメントはイニシャルドキュメント セクションと、オプションのセクションから構成されます。各 セクション にはそれぞれページ プロパティが含まれています。ドキュメント内に新規セクションを挿入するには、新規セクションが挿入されたデザイン内の位置にカーソルを移動して、コンテキスト メニューからもアクセスすることのできる [**新規ドキュメント セクション**] コマンドを選択してください。

、ページ番号、合計ページの削除

改ページ、ページ番号、合計ページを削除するには、プレースホルダを選択し、Delete キーを押下してください。

関連項目

- ファイル | 印刷

16.8.26 ユーザー定義アイテム

入力 | ユーザー定義 アイテム コマンドへマウスオーバーすることで、ユーザー定義要素 または ユーザー定義 XML テキスト ブロック を挿入するためのコマンドが表示されます。これらコンポーネントの使用方法については、SPS ファイル コンテンツ | ユーザー定義要素、XML テキスト ブロックのセクションを参照ください。

関連項目

- ユーザー定義要素
- タグで囲む | ユーザー定義要素
- ユーザー定義 XML テキストブロック
- ユーザー定義テンプレート

16.9 タグで囲むメニュー

[タグで囲む] メニューには、デザイン上の選択対象を様々なデザイン コンポーネントにより囲むためのコマンドが収められています。これらコマンドのいくつかは、コンポーネントをデザインへ挿入するためのツール バーアイコン からもアクセスすることができます（同様のコマンドは挿入メニューメニューからもアクセスできます）。[タグで囲む] コマンドは、SPS デザインにて選択対象を右クリックすることで表示されるコンテキスト メニューからもアクセスすることができます。SPS のカーソル位置で利用することのできないコマンドは、メニューとコンテキスト メニューで無効になります。

メモ [タグで囲む] コマンドはSPS を構築するためのコマンドであるため、デザイン ビューでのみ利用可能となります。

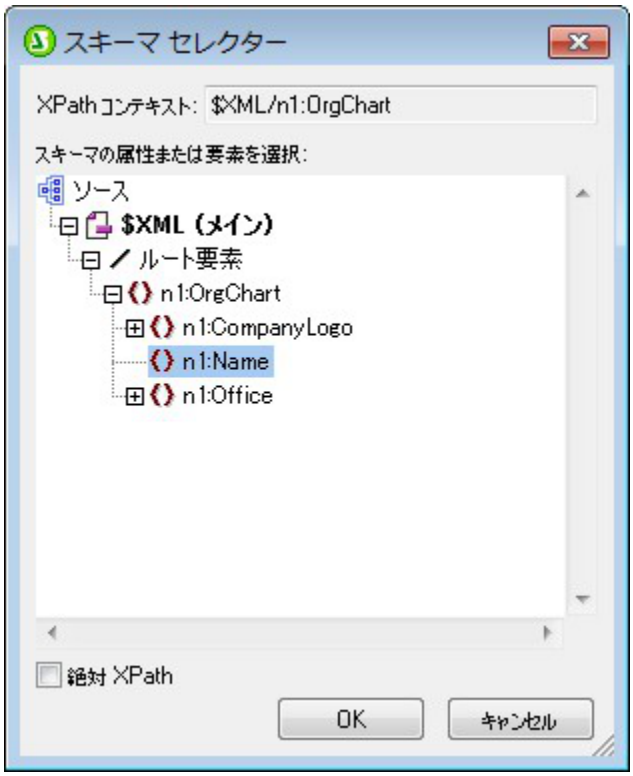
関連項目

- コンテンツの編集の手順
- ツールバー

16.9.1 テンプレート

[タグで囲む | テンプレート] コマンドにより、選択されたデザイン コンポーネントやテキストが、選択されたスキーマ ツリー ノードにより囲まれます。以下の操作を行なってください：

1. テンプレートにより囲むデザイン コンポーネントやテキストを選択します。
2. [タグで囲む | テンプレート] コマンドを選択します。スキーマ セレクター ダイアログが表示されます（以下のスクリーンショットを参照）。



3. XPath コンテキスト フィールドには選択対象のコンテキスト ノードが表示されます。テンプレートとして作成するノードを選択してください。上のスクリーンショットでは、n1:Name ノードがテンプレートのノードとして選択されています。
4. [OK] により操作を完了します。

選択されたノードに対するテンプレートが、選択対象の周りに作成されます。

関連項目

- テキストとして XML コンテンツを挿入
- 出力の構造
- デザイン要素の挿入

16.9.2 ユーザー定義テンプレート

タグで囲む | ユーザー定義テンプレート] コマンドにより、選択対象を XPath 条件式によりユーザーが指定したノードに対するテンプレートで囲みます。以下の操作によりユーザー定義テンプレートを挿入することができます:

1. ユーザー定義テンプレートにより囲むデザイン上のコンポーネントを選択します。
2. [**タグで囲む | ユーザー定義テンプレート**] コマンドを選択します。XPath 式を編集 ダイアログが表示されます。
3. 目的のノードを選択する XPath 条件式を入力します。(i) XPath 条件式はユーザー定義テンプレートが作成されたノードのコンテキスト内で評価され、(ii) XPath 条件式はドキュメント内部や別の XML ドキュメントにある任意のノードを選択することができることに留意してください。
4. XPath 条件式を入力した後に [OK] をクリックして完了します。

ターゲットのノードに対するユーザー定義テンプレートが選択対象の周りに作成されます。

に関する詳しい説明は、次のセクションを参照してください: SPS ファイル 構造 | テンプレートとデザインフラグメント | 変数テンプレート.

関連項目

- SPS ファイル コンテンツ | ユーザー定義テンプレート
- デザイン要素の挿入
- 挿入 | テンプレート
- 挿入 | ユーザー定義テンプレート

16.9.3 変数テンプレート

[**タグで囲む | 変数テンプレート**] コマンドにより、SPS デザイン内で定義された変数に対するテンプレートを選択範囲の周りに作成します。

1. 変数テンプレートを周りに作成するデザイン内のコンポーネントを選択します。
2. [**タグで囲む | 変数テンプレート**] コマンドを選択します。変数テンプレートダイアログにより囲む ダイアログが表示されます。

3. ダイアログのリストから、作成するテンプレートに使用する変数を選択します。
4. [OK] をクリックして完了します。

選択対象の周りに変数テンプレートが作成されます。

に関する詳しい説明は、次のセクションを参照してください：, SPS ファイル: 構造 | テンプレートとデザインフラグメント | 変数テンプレート.

関連項目

- 挿入 | テンプレート
- SPS ファイル: コンテンツ | ユーザー定義テンプレート

16.9.4 段落、特殊な段落

[段落] コマンド によりHTMLの段落 <p> 要素が選択されたコンポーネントの周りに作成されます。静的テキストが選択されている状態、または（タグをクリックすることで）ノード全体を選択することで、コンポーネントが選択されているものとみなされます。カーソルが静的なテキスト内部に配置されている場合、段落要素（の開始および終了タグ）がその位置に挿入されます。デザイン要素の挿入 ツールバーにある**段落の挿入**アイコンを使っても挿入することができます

[特殊な段落] コマンドでは、選択されたノードに定義済みのフォーマットを割り当てることができます。利用可能な定義済みフォーマットはツールバーのコンボボックスからも選択することができます。

各種段落にはそれぞれフォーマットが与えられています。pre フォーマットでは、改行が空白スペースに正規化されることなく、そのまま改行として出力されることに注意してください。

関連項目

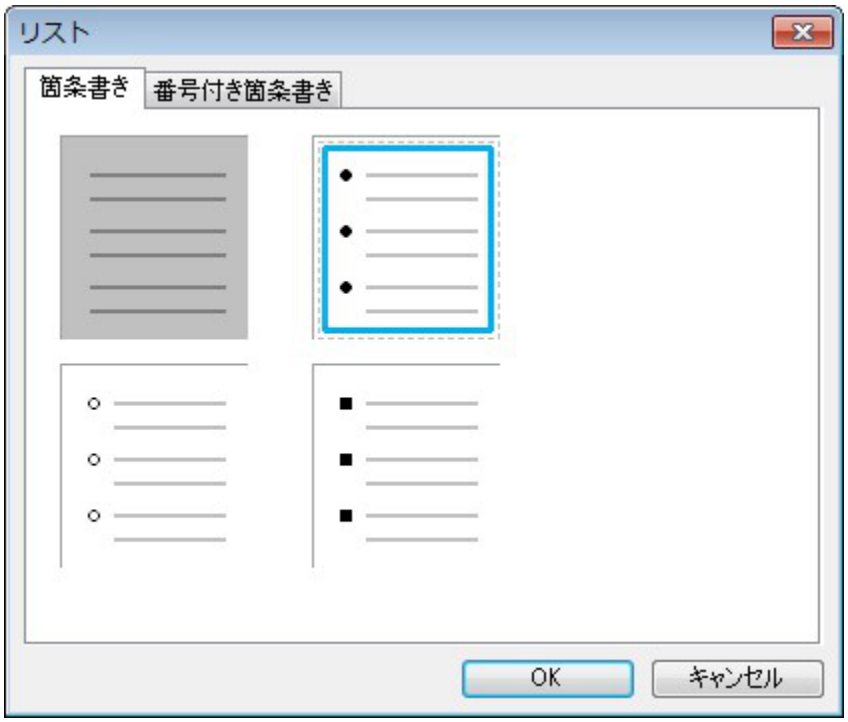
- 定義済みのフォーマット
- CSS スタイルと作業

16.9.5 リスト

[タグで囲む | リスト] コマンドにより、選択対象の周りに静的なリストとリスト アイテムが作成されます。選択対象に（Enter キーを押下することで挿入される）改行が含まれている場合、改行により区切られた各テキストが個別のリスト アイテムとして作成されます。

このコマンドが選択されると、リスト ダイアログが表示されます（以下のスクリーンショットを参照）。

(C) 2015-2021 Altova GmbH



リスト アイテムのマーカーを選択し、[OK] をクリックすることでリストが作成されます。選択対象内にある改行の数だけリスト アイテムが作成されます。Enter キーを押下することで、新たなリスト アイテムがリストに追加されます。

メモ　静的なコンテンツを選択し、デザイン要素の挿入 ツール バーにあるリスト アイコンをクリックすることで同様の結果を得ることもできます。

関連項目

- リストの作成
- タグで囲む | リスト

16.9.6　ブックマーク と ハイパーリンク

[タグで囲む | ブックマーク] ならびに [タグで囲む | ハイパーリンク] コマンドは、SPS デザイン内でテキストやコンポーネントが選択された状態で有効になります。これらのコマンドにより、ブックマークとハイパーリンクが選択範囲の周りにそれぞれ作成されます。ブックマークとハイパーリンクの処理と、その作成方法については、「高度な機能 | 目次、参照、ブックマーク」.

関連項目

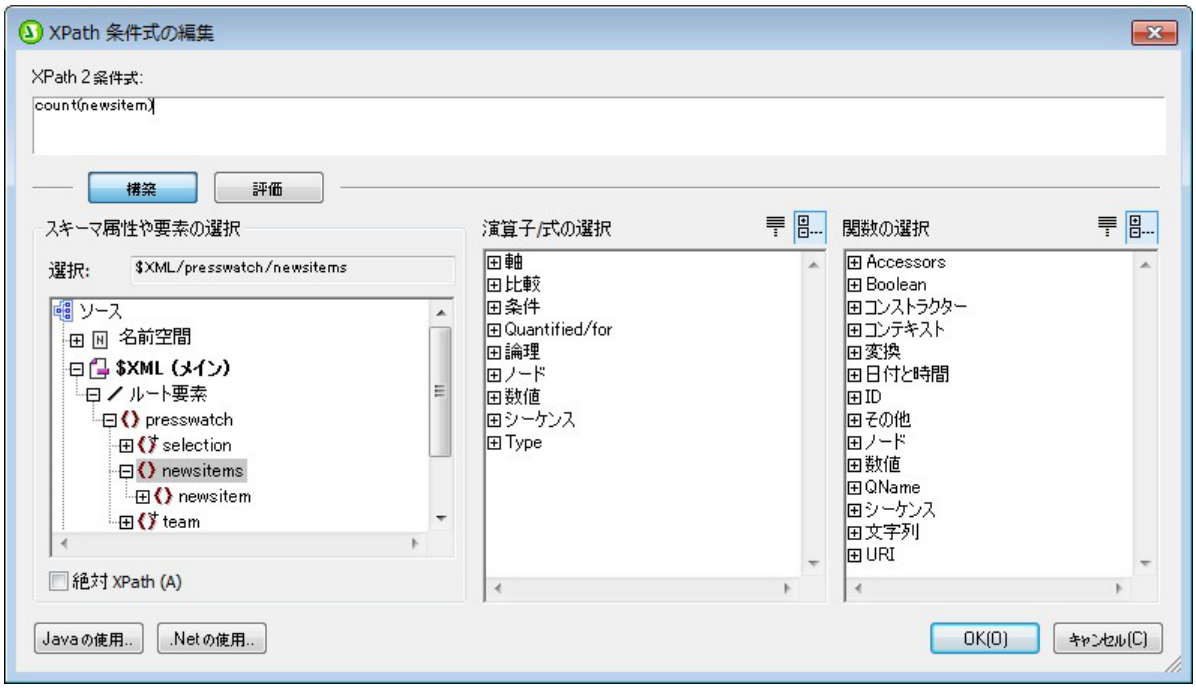
- ブックマークの挿入
- ハイパーリンクの定義

16.9.7 条件、出力ベースの条件

[条件] コマンドにより、カーソル位置または選択対象の周りに条件を挿入することができます。条件は1 つ以上の分岐から構成され、それぞれの分岐内部に処理ルールが含まれることになります。これにより、異なる処理ルールのセットをそれぞれの分岐に対して指定することができます。例えば、ノードのコンテンツがStop という文字列の場合にはノードのコンテンツを赤色に、Go という文字列の場合には緑色に、どちらでもない場合には黒色で出力されるような指定を行うことができます。

以下の操作により条件を挿入することができます

1. デザイン上の目的の位置にカーソルを配置するか、条件分岐内部に配置するコンポーネントを選択します。
2. メニューコマンドの[**挿入 | 条件**] を選択するか、右クリックにより表示されるコンテキスト メニューから[**挿入 | 条件**] を選択します。
3. 表示される XPath 式 の編集ダイアログ にて XPath 条件式を入力します（以下のスクリーンショットを挿入）。



構築されている条件式で使用されるコンテキスト ノードが、左端のペインにあるスキーマ ツリーにてハイライトされます。テキスト ボックスにXPath 条件式を直接入力することができるほか、ペイン内にあるアイテムをダブルクリックすることで、そのアイテムを挿入することもできます。左端にあるスキーマ ツリーから挿入されたノードは、コンテキスト ノードに対して相対的（[**絶対 XPath**] チェック ボックスにチェックが入っていない場合）、または絶対的（[**絶対 XPath**] チェック ボックスにチェックが入っている場合）に挿入されます。

4. [**OK**] をクリックすることで条件の挿入を完了します。入力されたXPath 条件式に対して適用される分岐とともに条件が作成されます。

出力ベース条件

このコマンドにより、カーソル位置または選択対象の周りに出力ベースの条件を挿入することができます。条件の分岐 (Authentic View、RTF、またはHTML) が、それぞれ異なる出力を表します。分岐タグ上部にマウスを移動するか（プロパティ サイドバーの条件分岐エントリーにある 編集ボタンをクリックして） 選択された分岐のXPath 条件式をチェックすることで、どの分岐がどの出力に対応しているかを確認することができます。出力ベースの条件がカーソル挿入位置に作成されると、分岐がすべて空の条件が作成され、各分岐に対してコンテンツを挿入することになります。出力ベースの条件がコンポーネントの周りに作成されると、選択されていたコンポーネントがすべての分岐内部に含まれ

ることになります。出力ベースの条件に関する詳細については、出力をベースにした条件を参照ください。出力ベースの条件は、一般的な条件と同じように編集、移動、削除することができます。

条件分岐の XPath 条件式を編集する

分岐のXPath 条件式を編集するには、デザイン ビューにて分岐を選択し、プロパティ サイドバーから [条件分岐 | 条件トリガー] を選択し、XPath アイテムの編集ボタン ... をクリックしてください。XPath 条件式の編集ダイアログが表示され、条件式を編集することができます（上のスクリーンショットを参照）。

条件分岐の追加、分岐の順序変更、分岐の削除

新規条件分岐の追加、分岐の順序変更、そして分岐を削除するには、目的の分岐を右クリックして、コンテキスト メニューから目的のアイテムを選択してください。

関連項目

- 条件

16.9.8 目次ブックマークと目次レベル

デザイン内にあるコンポーネントが選択されている状態から、関連する目次（TOC）コンポーネントを作成することができます。目次コマンドのリストを以下に示します。リンクをクリックすることで、その目次コンポーネントのセクションから使用方法の詳細を確認することができます。

- 目次ブックマーク
- 目次ブックマーク（ウィザード）
- レベル
- レベル参照

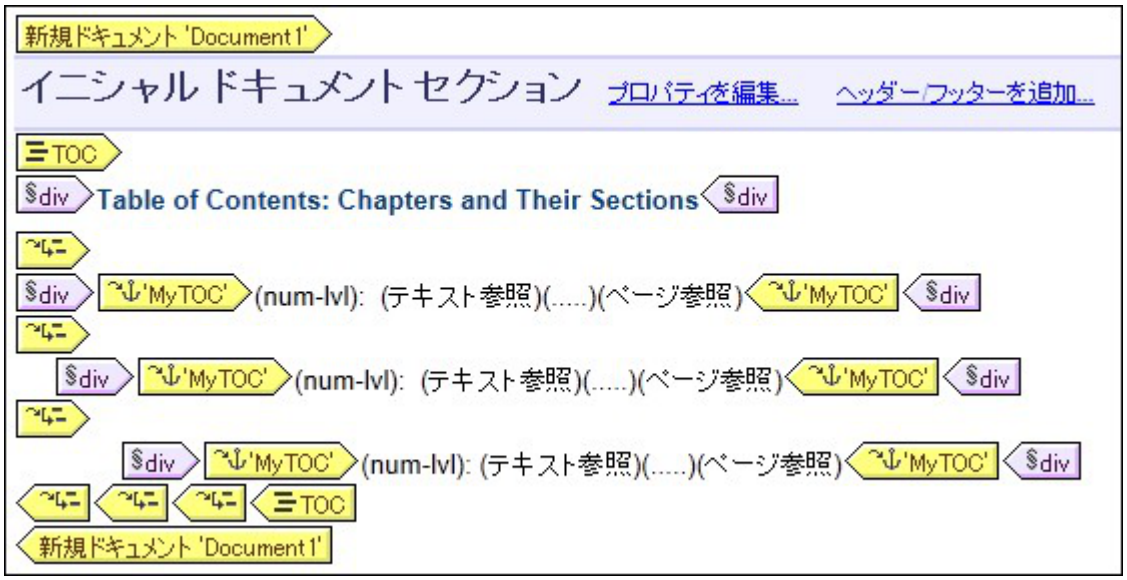
メモ　これらのコンポーネントは、デザイン内の適切な箇所をクリックすることで表示されるコンテキスト メニューのコマンドとしても利用することができます。

関連項目

- 目次（TOC）

16.9.9 新規ドキュメント

[**新規ドキュメント**] コマンドにより、現在の選択対象が新たなドキュメント テンプレートにより囲まれます（以下のスクリーンショットを参照）。



新規ドキュメント テンプレートにはイニシャルドキュメント セクションが含まれており、内部にはコマンドが選択された際の選択対象が含まれます。上のスクリーンショットでは、目次デザイン コンポーネントが新規ドキュメント テンプレートにより囲まれており、イニシャルドキュメント セクションの中にコンテンツを入力することができます。必要な場合には、[**挿入 | ページ/カラム/ドキュメント セクションの追加**] コマンドによりドキュメント セクションをイニシャルドキュメント セクションへ追加することもできます。

新規ドキュメント テンプレートにより、新たなドキュメントが出力に作成されます。これにより、出力が複数の出力ドキュメントから構成されるようになります。

複数の出力ドキュメントの処理方法については、複数のドキュメントの出力のセクションを参照ください。

関連項目

- 複数のドキュメントの出力
- 新規ドキュメントテンプレートの挿入
- 新規ドキュメント テンプレートとデザイン構造
- ドキュメント プロパティとスタイル

16.9.10 ユーザー定義要素

[**タグで囲む | ユーザー定義要素**] コマンドにより、デザイン内の選択対象の周りにユーザー定義要素 が作成されます。ユーザー定義要素の使用方法については、「 SPS ファイル: コンテンツ| ユーザー定義要素のセクションを参照ください。

関連項目

- ユーザー定義要素
- タグで囲む| ユーザー定義要素
- ユーザー定義 XML テキストブロック
- ユーザー定義テンプレート

16.10 テーブル メニュー

[テーブル] メニューには、静的または動的なテーブルを挿入し、それらテーブルの構造やプロパティを変更するためのコマンドが含まれています。テーブルの構造は行や列の追加、挿入、削除、結合、分割により変更することができます。テーブルや個々の列、行、セルのプロパティはCSS スタイルとテーブルやサブコンポーネントのHTML プロパティにより定義することができます。

テーブルコマンドは[テーブル] メニューや、テーブルツールバーにあるアイコンからアクセスすることができます。カーソルの位置により利用できるテーブルコマンドが変化します。テーブルの挿入コマンドをクリックすることで、SPS 内の任意の位置にてテーブルを挿入することができます。テーブルの構造を編集するには、適切なセル、列、または行にカーソルを移動させ、目的の編集コマンドを選択してください。フォーマット プロパティを編集するには、適切なセル、列、行、またはテーブルにカーソルを移動させ、スタイル サイドバーまたはプロパティ サイドバーにてテーブルコンポーネントに対するプロパティを定義してください。

テーブル メニューでは以下のコマンドを使用できます:

- テーブルの挿入、テーブルの削除
- テーブルヘッダー/フッターの追加
- 行/列の追加と挿入
- 行/列の削除
- 左、右、下、上にあるセルとの結合
- 水平方向/垂直方向にセルを分割
- セルの境界とテーブル マークアップの表示
- テーブル プロパティ
- セル コンテンツの垂直配置

ヘッダーとフッター

動的なテーブルを作成した場合、ヘッダーやフッターをテーブルに追加するか指定することができます(フッターはテーブルが上から下方向に大きくなる場合だけ利用することができます)。静的テーブルでは、テーブルの上部と下部に手動で行を追加することにより、それぞれヘッダーとフッターを作成することができます。静的ならびに動的なテーブル内にあるヘッダーとフッターの構造は、セルの分割と統合により修正することができます。

テーブル内のナビゲーション
Tab キーや矢印キーにより、テーブル セル間を移動することができます。

セル コンテンツの追加
セルにはあらゆる種類のSPS コンポーネントを挿入することができます。コンポーネントのフォーマットは標準的なフォーマット ツールにより行われます。

関連項目

- テーブルツールバー
- テーブルとの作業

16.10.1 テーブルの挿入、テーブルの削除

テーブルの挿入 コマンド により、空のテーブルがデザインへ挿入されます。このコマンドを選択することで、作成するテーブルを静的と動的から選択することができます。

- 静的テーブルを選択すると、テーブルの大きさ（行と列の数）を選択するダイアログが表示されます。
- 動的なテーブルを選択すると、XPath セレクター ダイアログが表示され、動的テーブルとして作成されるノードを選択することになります。[OK] をクリックすると、テーブルの各アイテム フィールドに表示される子ノードを選択する動的テーブルの作成ダイアログが表示されます。詳細については、動的テーブルの作成を参照ください。

テーブルの構造は、行や列を追加、挿入、削除することにより、テーブルの作成後も変更することができます。

テーブルの削除 コマンド により、カーソルが配置されている静的または動的なテーブルが削除されます。

関連項目

- テーブルツールバー
- テーブルとの作業

16.10.2　テーブルヘッダー/フッターの追加

テーブルヘッダーは、（テーブルボディー上部の）ヘッダー行または（テーブルボディー左側の）ヘッダー列として表示されます（マークアップとしては、ヘッダー列がテーブルボディー内部に含まれることもあります）。同様に、テーブルフッターは（テーブルボディー下部の）フッター行または（テーブルボディー右側の）フッター列として表示されます（マークアップとしては、フッターがテーブルボディー内部に含まれることもあります）。

メモ　HTML 出力ではテーブル ヘッダーが th 要素内に含まれるため、太字で表示されます（th 要素が bold フォーマットを継承しているため）。

HTML 出力ではテーブル ヘッダーが th 要素内に含まれるため、太字で表示されます（th 要素が bold フォーマットを継承しているため）。

テーブル ヘッダー列の追加： テーブル ボディーの左にヘッダー列を追加します。

テーブル フッター列の追加： テーブル ボディーの右にフッター列を追加します。.

テーブル ヘッダー行の追加： テーブル ボディーの上にヘッダー行を追加します。

テーブルフッター行の追加: テーブルボディーの下にフッター行を追加します。

関連項目

- テーブルツールバー
- テーブルとの作業

16.10.3　行/列の追加と挿入

[**行の追加**] コマンド により、静的または動的なテーブル内にあるカーソルから見て行が追加されます。

[**行の挿入**] コマンド により、カーソルから見て行が挿入されます。このコマンドは静的と動的なテーブルの両方で使用することができます。

[**列の追加**] コマンド により、静的または動的なテーブル内にあるカーソルから見て列が追加されます。

[**列の挿入**] コマンド により、カーソルから見て列が挿入されます。このコマンドは静的と動的なテーブルの両方で使用することができます。.

関連項目

- テーブルツールバー
- テーブルとの作業

16.10.4 行/列の削除

[**行の削除**] コマンド により、カーソルが配置されているテーブルの行が削除されます。このコマンドは、静的ならびに動的なテーブルの両方に対して適用されます。

[**列の削除**] コマンド により、カーソルが配置されているテーブルの列が削除されます。このコマンドは、静的ならびに動的なテーブルの両方に対して適用されます。

関連項目

- テーブルツールバー
- テーブルとの作業

16.10.5 左/右/下/上のセルを結合

[**左のセルを結合**] コマンド により、カーソルが配置されているセルと、その左にあるセルが結合されます。両方のセルのコンテンツが連結されて新しいセルに表示されます。両方のセルにあるすべてのプロパティ値が新たなセルに渡されます。このコマンドは静的ならびに動的なテーブルで使用することができます。

[**右のセルを結合**] コマンド により、カーソルが配置されているセルと、その右にあるセルが結合されます。両方のセルのコンテンツが連結されて新しいセルに表示されます。両方のセルにあるすべてのプロパティ値が新たなセルに渡されます。このコマンドは静的ならびに動的なテーブルで使用することができます。

[**下のセルを結合**] コマンド により、カーソルが配置されているセルと、その下にあるセルが結合されます。両方のセルのコンテンツが連結されて新しいセルに表示されます。両方のセルにあるすべてのプロパティ値が新たなセルに渡されます。このコマンドは静的ならびに動的なテーブルで使用することができます。

(C) 2015-2021 Altova GmbH

[上のセルを結合] コマンド により、カーソルが配置されているセルと、その上にあるセルが結合されます。両方のセルのコンテンツが連結されて新しいセルに表示されます。両方のセルにあるすべてのプロパティ値が新たなセルに渡されます。このコマンドは静的ならびに動的なテーブルで使用することができます。

関連項目

- テーブルツールバー
- テーブルとの作業

16.10.6　水平方向/垂直方向にセルを分割

[セルを左右に分割] コマンド により、カーソルが配置されているセルの右隣に新たなセルが作成されます。オリジナルのセル内にあるコンテンツは変更されません。オリジナルのセルにて定義されたすべてのプロパティは、新たなセルにもコピーされます。このコマンドは静的ならびに動的なテーブルで利用することができます。

[セル上下に分割] コマンド により、カーソルが配置されているセルの下に新たなセルが作成されます。オリジナルのセル内にあるコンテンツは変更されません。オリジナルのセルにて定義されたすべてのプロパティは、新たなセルにもコピーされます。このコマンドは静的ならびに動的なテーブルで利用することができます。

関連項目

- テーブルツールバー
- テーブルとの作業

16.10.7　セルの境界とテーブル マークアップの表示

[セル境界の表示] や [テーブル マークアップの表示] コマンドを有効にすることで、セルの境界線やテーブル行と列のマークアップが表示されます。これら 2 つのオプションを有効にすることにより、テーブルの構造を理解するのに役立てることができます。無効にすることで、更に正確なテーブルの表示を行うことができます。

[セル境界の表示] コマンドにより、border の値が 0 になっているテーブルにも、境界 (border) を表示することができます。

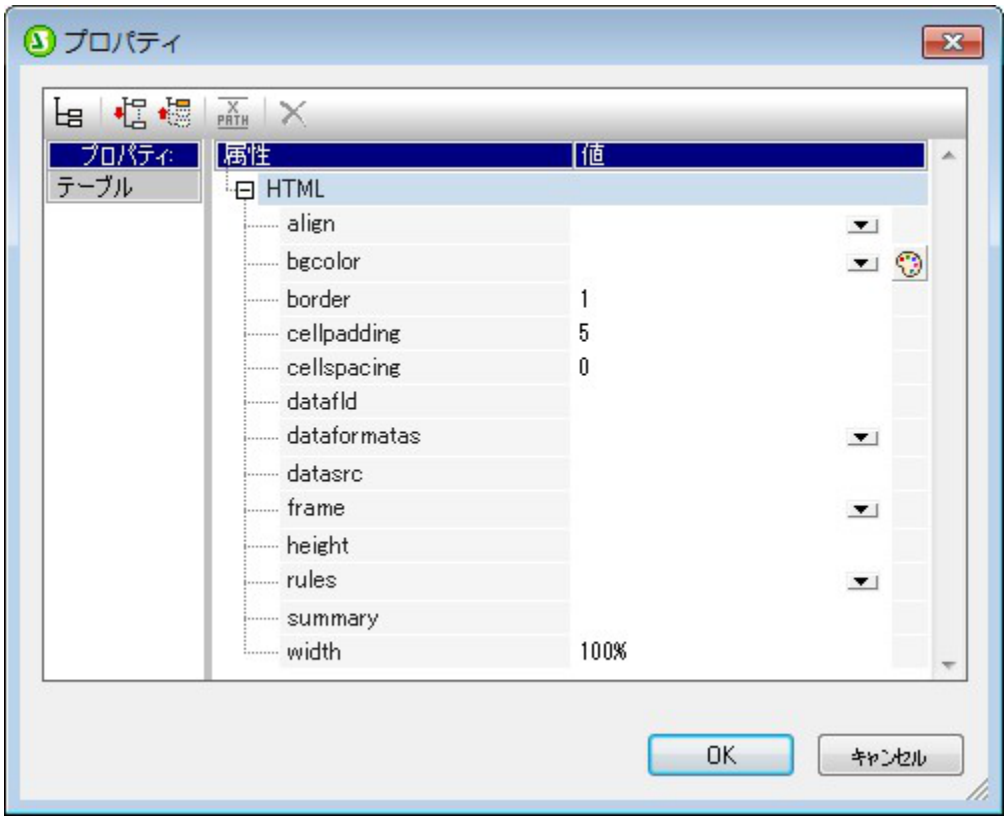
[テーブル マークアップの表示] コマンドにより、青色で示される行と列のマークアップが表示されます。

関連項目

- テーブルツールバー
- テーブルとの作業

16.10.8 テーブル プロパティ

[テーブル プロパティ] コマンドは、静的または動的 テーブル内部にカーソルが配置されている時に有効になります。コマンドをクリックすることで、テーブルコンポーネントが選択されたプロパティサイドバーが表示されます（以下のスクリーンショットを参照）。



テーブルのプロパティを編集して、[OK] ボタンにより完了します。

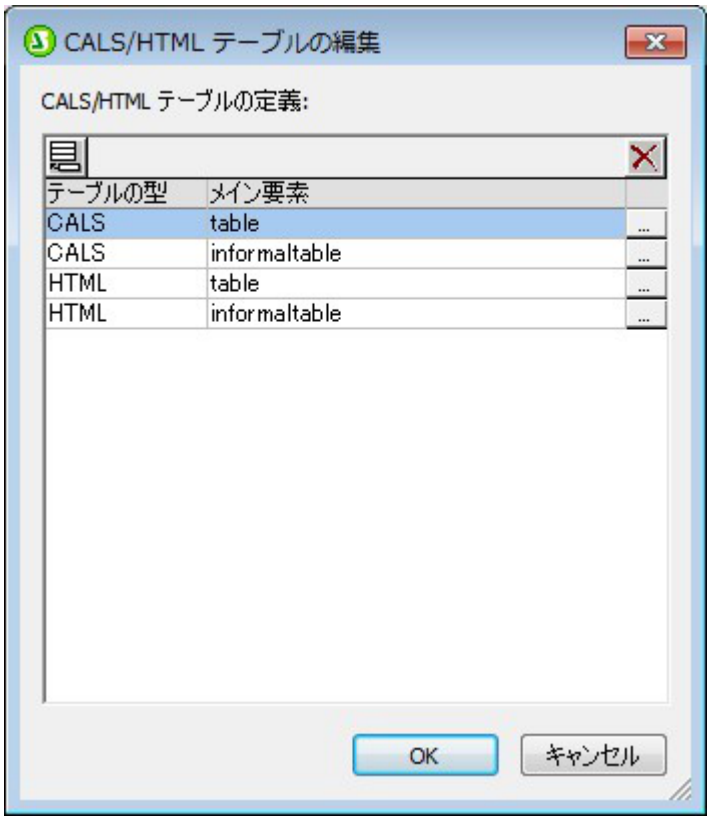
関連項目

- テーブルツールバー
- テーブルとの作業

16.10.9 CALS/HTML テーブルの編集

[CALS/HTML テーブルの編集] コマンドにより、テーブルとして出力にて生成されるCALS テーブルモデルまたはHTML テーブルモデル形式のXMLドキュメントのデータを定義することができます。出力フォーマットのテーブルマークアップがXMLドキュメントから直接得られるほか、更なるテーブルフォーマット スタイルをSPS にて追加することができます。

このコマンドを選択することで、CALS/HTML テーブルの編集ダイアログが表示されます（以下のスクリーンショットを参照）。



CALS/HTML テーブルに関する詳細については、次のセクションを参照してください: テーブル.

関連項目

- テーブルとの作業
- 静的テーブルの作成
- 動的テーブルの作成

16.10.10　セル コンテンツの高さ揃え

テーブルツールバーには、セルコンテンツ高さ配置を行うコマンド アイコンが収められています。セル内にカーソルが配置された状態で、目的のアイコンをクリックしてください。

上揃え: セルのコンテンツ高さをセル上部に揃えます。

中央揃え: セルのコンテンツ高さをセル中央に揃えます。

下揃え: セルのコンテンツ高さをセル下部に揃えます。

関連項目

- テーブルツールバー
- テーブルとの作業

(C) 2015-2021 Altova GmbH

16.11 Authentic メニュー

[Authentic] メニューに収められたコマンドにより、以下の処理を行うことができます:

- SPS により表示される XML ドキュメントのAuthentic View 部分をカスタマイズ
- StyleVision の Authentic View プレビューにてドキュメントの編集を行う。

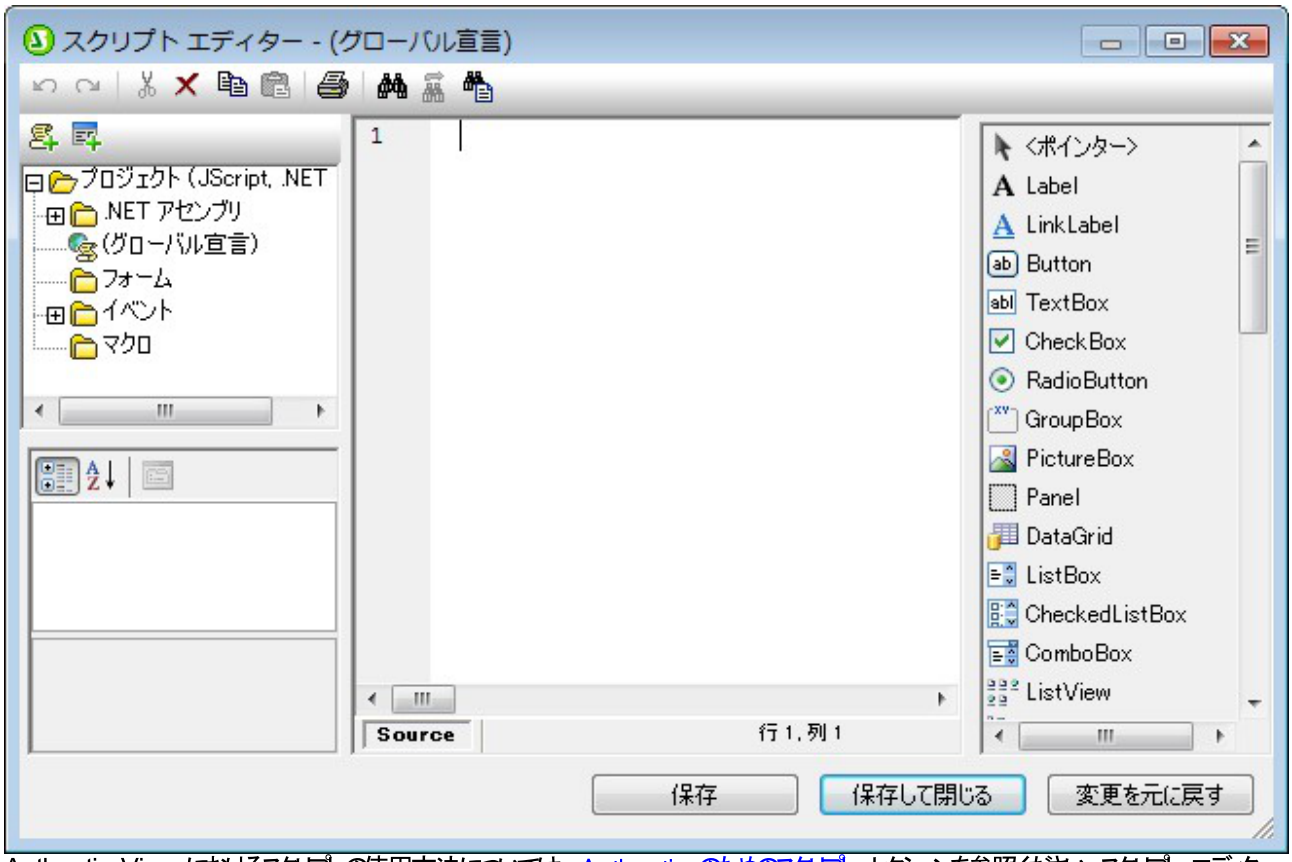
Authentic 内のメニューは、下にリストされるとおりです

- Authentic スクリプトの編集
- カスタム ツール バーボタン
- マクロ参照のチェック
- 日付選択の自動追加
- DB コントロールの自動追加
- 再ロードとXML の検証
- XML データの新たな行を選択し編集
- XML エンティティの定義
- マークアップ コマンド
- (動的テーブル) 行コマンド

これら各コマンドの詳細については、このセクションのサブセクションを参照ください。

16.11.1 Authentic スクリプトの編集

[Authentic スクリプトの編集] コマンドにより、Authentic View のフォームやイベント、マクロを作成するためのStyleVision のスクリプトエディターが表示されます (以下のスクリーンショットを参照)。



Authentic View におけるスクリプトの使用方法については、Authentic のためのスクリプト セクションを参照ください。スクリプト エディターの操作方法については、プログラマーのレファレンスのScriptingセクションを参照ください。

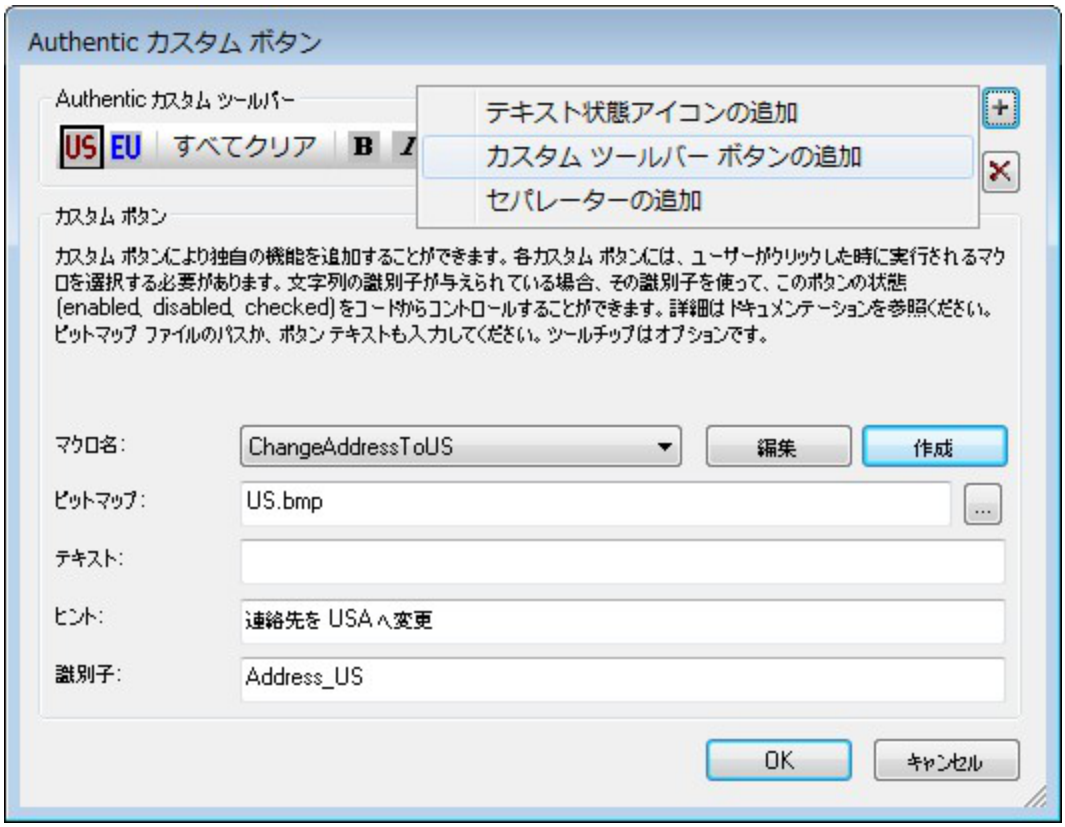
関連項目

- スクリプトエディター
- Authentic のためのスクリプト
- SPS とAuthentic View

16.11.2 カスタム ツールバーボタン

[ツールバー ボタンのカスタマイズ] コマンドを選択すると、Authentic ツールバーをデザインするためのAuthentic カスタムボタンが表示されます。Authentic ツールバーがSPS とともに保存されると、Enterprise またはProfessional Edition のAltova 製品にあるAuthentic View にて、そのSPS に関連付けられたXML ファイルを編集する際に、これらツールバーが表示されます。

メモ Authentic View が搭載されている Altova 製品は、Enterprise ならびに Professional Edition の XMLSpy、StyleVision 、そして Authentic Desktop と Authentic Browser プラグインです。



ボタンの追加

ボタンやセパレーターを Authentic ツールバーに追加するには、ダイアログ右上にある追加 ([+]) ボタンをクリックします。(i) テキスト状態のアイコン、(ii) マクロを実行するためのカスタム ボタン、または (iii) ツールバーにあるボタンをグループに分けるためのセパレーターを追加することができます。テキスト状態アイコン と カスタム ボタン に関する詳細は、以下を参照ください。

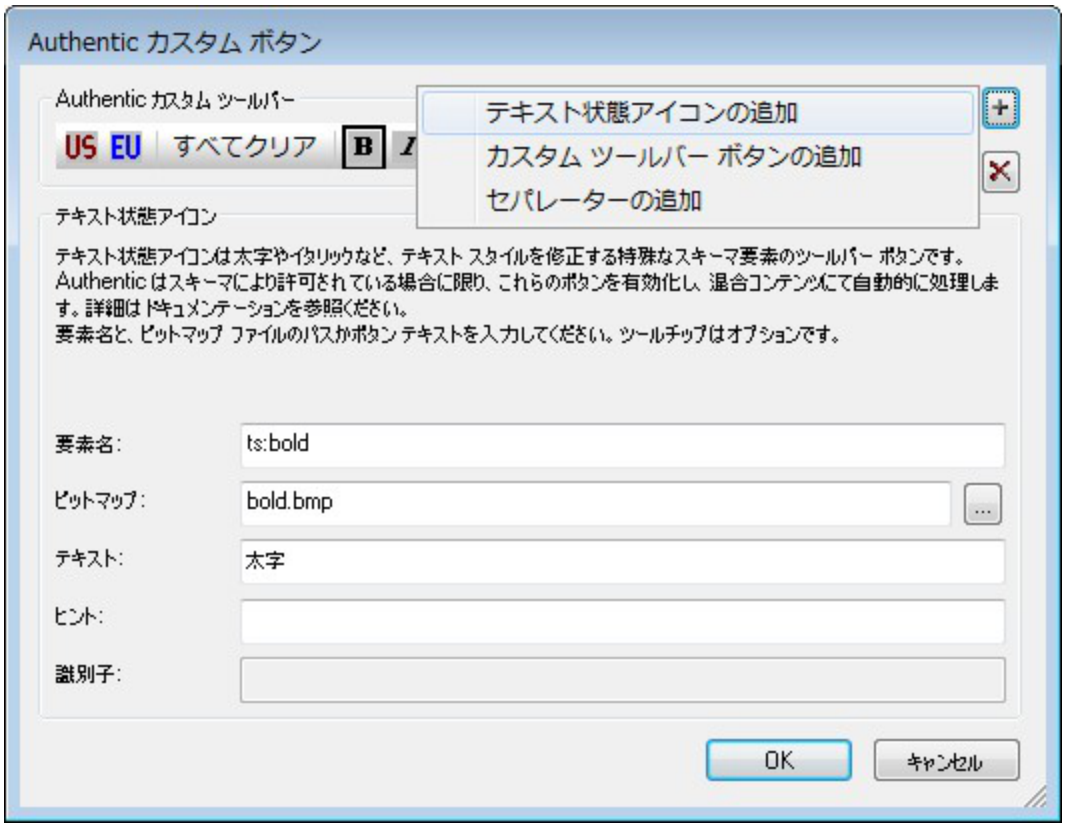
ボタン移動とボタンの削除

ボタンやセパレーターをドラッグすることで、そのボタンやセパレーターをツールバー内にある別の場所へ移動することができます。ボタンやセパレーターを選択し、ダイアログ右上にある削除ボタンをクリックすることで、そのボタンやセパレーターが削除されます。

テキスト状態アイコン

テキスト状態アイコンによりグローバル要素に対するアイコンが定義されます。Authentic View ユーザーが Authentic View ドキュメント内にあるテキストを選択し、テキスト状態アイコンをクリックすると、アイコンにより定義された要素が、選択されたテキストの周りに挿入されます。テキスト状態アイコンは、太字やイタリックといったインラインのフォーマットを行うために通常利用されます。

テキスト状態アイコンを Authentic ツールバーへ追加するには、Authentic カスタム ボタン ダイアログの右上にある追加ボタンをクリックして、[**テキスト状態アイコンの追加**] を選択します。作成されるテキスト状態アイコンに対応する XML 要素の名前を入力し、ボタンのビットマップ イメージを選択するか、ボタンのテキストを入力してください。オプションとして、Authentic View ユーザーがテキスト状態アイコンへマウスオーバーした時に表示されるツールチップを入力することができます。[OK] をクリックするとボタンが Authentic ツールバーへ追加されます。

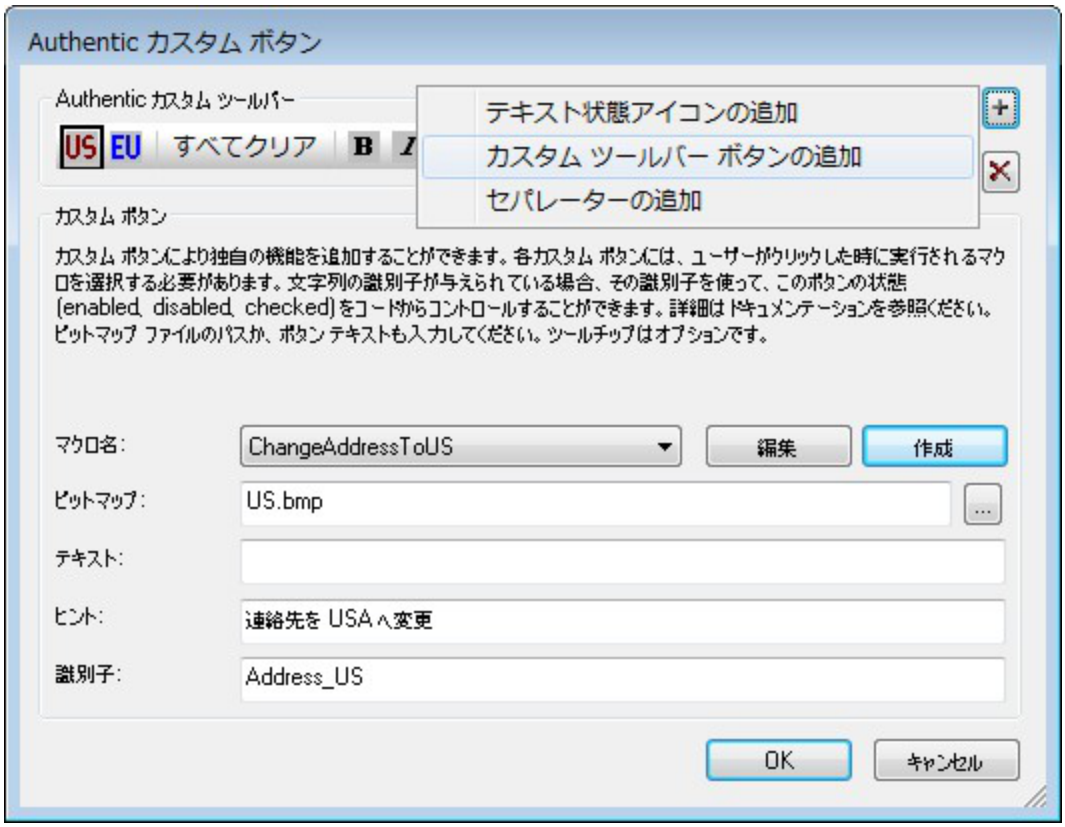


上のスクリーンショットでは、bold 要素に対するテキスト状態アイコンが表示されています。このテキスト状態アイコンにはbold.bmp という名前のイメージが使用されます。入力されているボタンのテキストはフォールバック（バックアップ）として使用されます。上のスクリーンショットで定義されているテキスト状態アイコンに対してイメージファイルが見つからない場合、Bold というテキストがボタンのテキストとして使用されます。

カスタム ボタン

Authentic View ユーザーがAuthentic ツールバーにあるカスタムボタンをクリックすると、マクロが実行されます。SPS デザインではカスタムボタンを作成して、実行されるマクロを指定することができます。

Authentic ツールバーへカスタムボタンを追加するには、Authentic カスタム ボタン ダイアログの右上にある追加ボタンをクリックして、[**カスタム ツールバー ボタンの追加**] を選択してください。



カスタム ボタン take 次のパラメータ

- ボタンに使用されるイメージの場所 (ビットマップ フィールド) またはボタンのテキスト (テキスト フィールド)
- マクロ名のコンボ ボックスでは、ドロップダウン リストからマクロを選択します。SPS にて保存されたマクロが示されます。[新規] ボタンをクリックすると、StyleVision のスクリプトエディターが開かれ、マクロを作成し SPS に保存することができます。[編集] ボタンをクリックすると、選択されたマクロがスクリプトエディターで開かれ、編集することができます。
  - オプションとして、Authentic View ユーザーがカスタム ボタンへマウスオーバーした時に表示されるツールチップを入力することができます。
  - 識別子フィールドでは、カスタム ボタンの識別子として使用されるテキスト文字列を入力します。この識別子はコードのスクリプト化で使用することができます。

上のスクリーンショットでは、ChangeAddressToUS マクロのカスタム ボタンが示されています。このカスタム ボタンでは US.bmp という名前のイメージが使用されています。ボタンのテキストもフォールバックとして入力することができます。ツールチップにもテキストが入力されており、カスタム ボタンには Address_US という識別子が与えられています。これらの例は Example プロジェクトの Authentic¥Scripting フォルダーにある `ToolbarButtons.sps` で確認することができます。

関連項目

- スクリプトエディター
- Authentic のためのスクリプト
- SPS と Authentic View

16.11.3 マクロ参照のチェック

「マクロ参照のチェック」コマンドにより、ツールバーボタンとスクリプトからマクロへの参照が正しいかチェックされます。無効な参照が見つかった場合にはメッセージが表示されます。

関連項目

- スクリプトエディター
- Authentic のためのスクリプト
- SPS とAuthentic View

16.11.4 日付選択の自動追加

日付選択の自動追加機能の有効/無効が切り替えられます。日付選択の自動追加機能が有効になっている場合、コンテンツまたは入力フィールドとして作成された xs:date または xs:dateTime データ型の要素すべてに対して、(コンテンツ) プレースホルダまたは入力フィールドの後に日付選択が挿入されます。

関連項目

- 日付選択の使用
- 日付選択の挿入
- 日付との作業
- 日付のフォーマット

16.11.5 DB コントロールの自動追加

DB コントロールの自動追加機能の有効/無効が切り替えられます。

DB コントロールの自動追加が有効になっている場合、DB テーブル要素がデザインへドロップされると、(以下に示される) DB コントロールパネルが、そのDB テーブル要素のRow 子要素の直前に挿入されます。

DB コントロールパネルにより、Authentic View ユーザーは Authentic View に表示されるDB テーブルの行の範囲を変更することができます。最初の(左隅にある) ボタンにより、最初のレコードが表示され、2 番目のボタンにより直前のレコードが、3 番目のボタンにより表示するレコードを指定するためのダイアログが表示され(以下のスクリーンショットを参照)、4 番目のボタンにより次のレコードを表示、最後の(右隅にある) ボタンにより最後のレコード

DB コントロールの自動追加が無効になっている場合、デザインドキュメントへDB テーブルがドロップされてもDB コントロール パネルは挿入されません。

メモ DB テーブルの開始と終了タグ間にカーソルが配置されている状態で、[**挿入 | DB コントロール**] 以下にあるサブメニュー コマンドを選択することにより、ナビゲーション ボタンを手動で挿入することもできます。このサブメニューからは、4 個のボタンからなるナビゲーションと、それらのボタンにレコードへ移動ダイアログを呼び出すためのボタンが加えられたナビゲーション ボタンを挿入することができます。

関連項目

- DB のためのSPS デザイン機能

16.11.6 Authentic View の再ロード、XML の検証

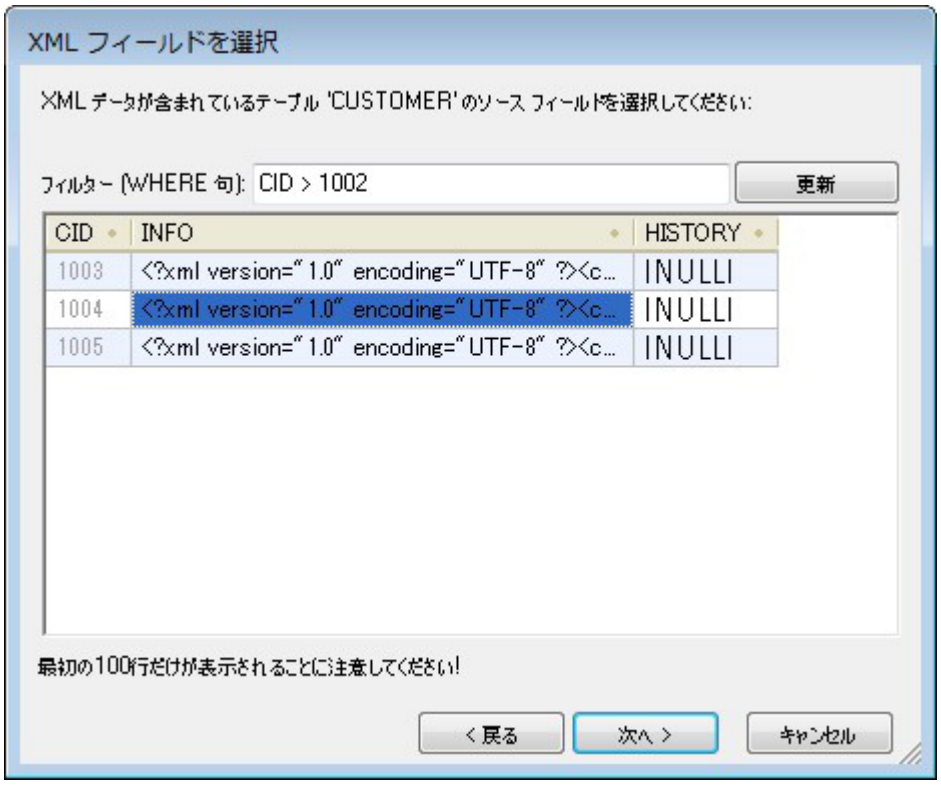
[**再ロード**] コマンドにより、Authentic XML データ ファイルが再ロードされます。ファイルがStyleVision とは別のソフトウェアにより修正された際に、または別のユーザーが他のマシンから作業している場合に、このコマンドを役立てることができます。

[**XML の検証**] (F8) コマンド により、XML ファイルを関連付けられたスキーマに対して検証します。StyleVision のXSD 1.0 またはXSD 1.1 バリデーターが使用されるかは、プロパティ ダイアログ 内で指定することもできます。個々のノードに対して入力されたその他の必須項目もチェックされます。検証チェックの結果はメッセージ ボックスにて表示されます。

16.11.7 XML データの新たな行を選択し編集

[**XML データから新たな行を選択し編集**] コマンドにより、IBM DB2 といったXML DB 内の関連するテーブルから新たな行を選択することができます。Authentic View に表示された行は、編集後にDB へ保存し直すことができます。

XML DB がXML データ ソースとして使用された場合、XML データ列のセルに含まれているXML ドキュメントがXML データとしてAuthentic View に表示されます。[**XML データから新たな行を選択し編集**] コマンドにより、同じ XML 列にある別のセル (行) からXML ドキュメントを選択することができます。このコマンドを選択すると、XML 列が含まれているテーブルが表示されるXML フィールドの選択ダイアログが表示されます (以下のスクリーンショットを参照)。



フィルタリングにより表示されるテーブルを絞り込むことができます。SQL のWHERE 句の内容をフィルターには入力してください（WHERE キーワードを除いた条件で、例えばCID > 1002 という形式になります）。[更新] ボタンをクリックすることで、表示内容が更新されます。フィルタリングされた結果の例を上のスクリーンショットに示します。目的のXML ドキュメントが含まれているセルを選択し、[OK] をクリックしてください。選択されたセル（行）のXML ドキュメントがAuthentic View へロードされます。

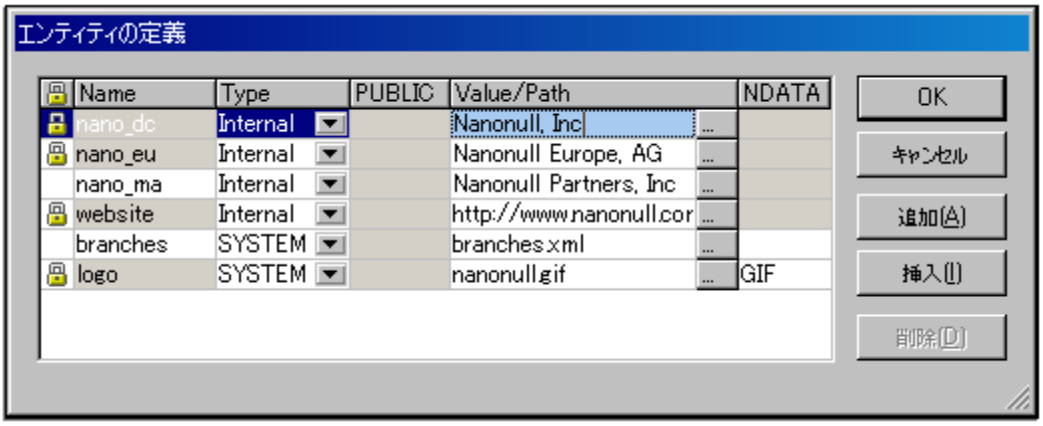
16.11.8 XML エンティティの定義

[XML エンティティの定義] コマンドは Authentic View でのみ利用することができます。Authentic View の[エンティティの定義] コマンドにより、XML ドキュメントへ追加するエンティティを定義することができます。エンティティが定義されると、右クリックにより表示されるコンテキスト メニューから[エンティティの挿入] を選択し、挿入するエンティティの名前を選択することで、定義されたエンティティをXML ドキュメントへ挿入することができます。

このコマンドにより定義されたエンティティは、以下にある 3 種類のいずれかになります:

- 内部解析対象実体: ドキュメント内部に頻繁に現れるテキスト文字列がエンティティの値となります。エンティティを使用することで、すべての出現が定義された値に展開されます。
- 外部解析対象実体: エンティティの各出現を置き換える外部 XML ファイルとなります。外部 XML ファイルのURI がエンティティの値として使用されます。
- 外部対象外実体: エンティティが処理された際に呼ばれる外部リソースとなります。エンティティの値は外部リソースのURI となります。

コマンドをクリックすると、エンティティの定義ダイアログが表示されます（以下のスクリーンショットを参照）



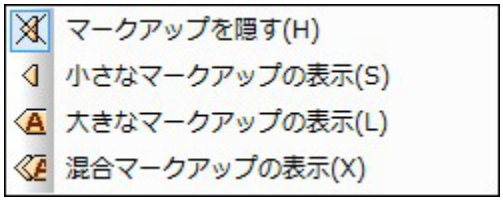
ダイアログこのダイアログの使用方法に関しては、次を参照してください：Authentic View ドキュメント内のエンティティの定義

関連項目

- パースされていないエンティティ URI

16.11.9　マークアップの表示

[**マークアップの表示**] コマンドにはAuthentic XML ドキュメント内に含まれるマークアップを制御するためのオプションがサブメニューとして含まれています。4 つのマークアップ コマンドにより、マークアップを隠し、様々なマークアップの表示モードを選択することができます。



マークアップにより、Authentic View にてノードのタグがどのように表示されるかが指定されます。同時に複数のオプションを選択することはできず、常にどれか1 つのオプションが選択されている必要があります。[**マークアップを隠す**] オプションを選択すると、タグの表示が行われません。[**小さなマークアップを表示**] オプションにより、名前なしの開始ならびに終了タグが表示されます。[**大きなマークアップを表示**] オプションでは、開始タグと終了タグが対応するノード名とともに表示されます。[**混合マークアップを表示**] オプションでは、Authentic ノードプロパティにて個々のノードに対して指定されたマークアップの設定が適用されます。個々のノードのデフォルト マークアップ プロパティはマークアップを隠すになっているため、ノードプロパティにて マークアップの表示を指定しない限り、ノードのマークアップは表示されません。

関連項目

- Authentic ノードプロパティ
- テーブルとの作業
- Authentic View インターフェイス

16.11.10 リッチエディット

[リッチエディット] コマンドへマウスオーバーすることで、リッチエディット コマンドが含まれるサブメニューが表示されます（以下のスクリーンショットを参照）。このサブメニュー内にあるコマンドはAuthentic View でリッチエディット コンポーネントとして作成された要素内部にカーソルが配置されている場合のみ有効になります。

リッチエディット コマンドがクリックされると、リッチエディットのテキスト スタイル プロパティ メニューが選択されているテキストに対して適用されます。Authentic View ユーザーは選択されたテキストのフォント、フォントの太さ、フォント スタイル、フォントの飾り、フォント サイズ、色、背景色、そして配置を指定することができます。
に関する詳しい説明は、about リッチエディット feature in context、次を参照してください：Authentic におけるテキスト スタイルの柔軟性.

関連項目

- リッチエディット ツールバー
- リッチエディット
- Authentic メニュー

16.11.11 （動的テーブル）行コマンド

Authentic View にてカーソルが動的テーブル内部に配置されている状態では、（動的テーブルの）**行コマンド**が有効になり動的テーブルの行を操作することができます。行の追加、挿入、複製、削除を行うことができるほか、選択されている行を上または下へ移動することができます。動的テーブルの行では固定されたデータ構造が使用されるため、Authentic View ユーザーは動的テーブルにより表示されているデータのエンテキスト内にあるデータ構造を操作していることになります。

行の内部にカーソルを配置することでカーソルは選択されます。選択された行の前や後に空の行を挿入または追加することができます。行の複製により、選択された行とコンテンツのコピーが、選択された行の後に作成されます。選択されている行を上下に移動することもできます。

関連項目

- テーブルツールバー
- テーブルとの作業
- Authentic View インターフェイス

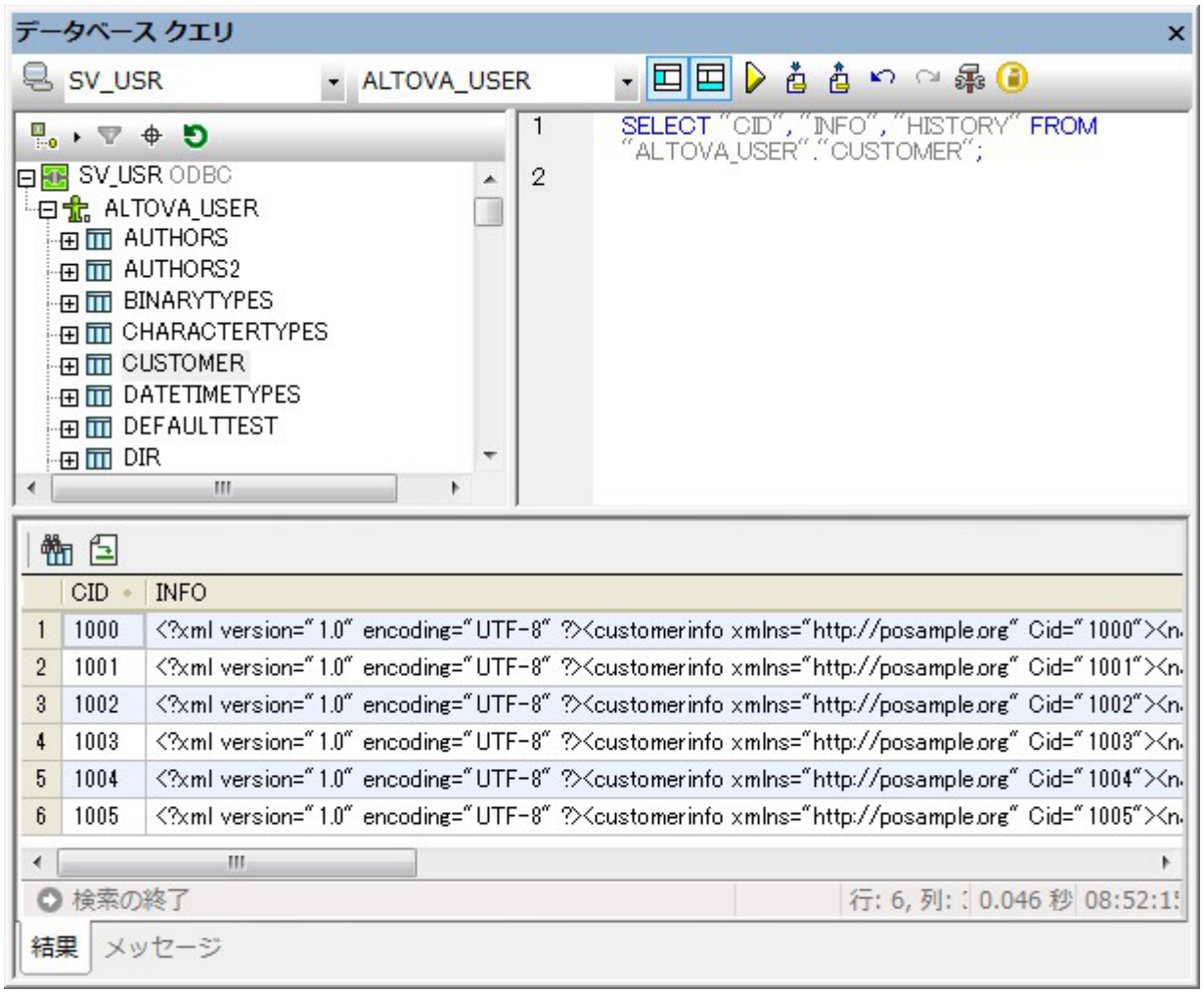
16.12 データベース メニュー

[データベース] メニューには接続されたデータベースへの問い合わせや、接続されたデータベースへ適用されるフィルターの編集や、フィルターをクリアするためのコマンドが収められています。

- データベースへ問い合わせ は、データベースへの接続処理 を行い、データベース クエリ ウィンドウを開きます。
- データベース フィルターの編集とクリア は、それぞれデータベース フィルターの編集ダイアログへアクセスし、データベース フィルターをクリアします。

16.12.1 データベースへの問い合わせ

[データベースへ問い合わせ] コマンド により、データベースへの接続を行い、クエリ（問い合わせ）を作成する データベース クエリ ウィンドウ が表示されます（以下のスクリーンショットを参照）。データベース クエリ ウィンドウの使用 データベースへ問い合わせのセクションを参照ください。



[データベースへの問い合わせ] コマンドにより、データベース クエリ ウィンドウの表示を切り替えることができます。

関連項目

- DB とStyleVision
- データベースへの接続処理
- データベースへ問い合わせ
- データ型の変換: DB からXSD へ

16.12.2　データベース フィルターの編集とクリア

[データベース フィルターの編集] コマンド により、データベーステーブルに対するフィルターを作成ならびに編集することができます。データベースフィルターにより、選択されたデータベーステーブルからインポートされ、表示されるデータが決定されます。データベースフィルターは1 個以上の条件により構成されます。条件の指定には、(= や> などの) 演算子と(テキストや数値などの) 値が組み合わせられます。条件はAND やOR などの論理式により組み合わせることもできます。

以下の操作によりデータベースフィルターを作成ならびに編集することができます:

1. データベースフィルターの作成や編集を行うトップレベルのデータ テーブル要素を選択します。デザイン ビューの要素タグを選択するか、スキーマ ツリーにある要素名を選択してください。
2. [データベース | データベース フィルターの編集] コマンドを選択するか、ツール バーにある対応するコマンドをクリックします。データベースフィルターの編集ダイアログが表示されます。

3. [AND を追加] や[OR を追加] ボタンにより条件を追加することができます。矢印ボタンにより条件を上下に移動することができます。[削除] ボタンにより条件を削除することができます。
4. データベースフィルターの条件を指定します。各条件は、フィールド名、演算子、そして値という3 つのパーツから構成されます。フィールド名と演算子の選択はコンボ ボックスより行うことができます。条件の値はキーボードにより入力、または($ 文字に続く パラメーターから選択することができます。

データベース フィルターのクリア コマンド

[**データベース フィルターのクリア**] コマンド により、確認後にフィルターが削除されます。

関連項目

- DB とStyleVision
- データベースへの接続処理
- データベースへ問い合わせ
- データ型の変換: DB からXSD へ

(C) 2015-2021 Altova GmbH

16.13　プロパティ メニュー

[プロパティ] メニューには、リストの挿入と入力フォーマット 機能に対してデータ型フォーマットを定義するためのコマンドが収められています。以下のサブセクションではそれぞれのコマンドについて説明します:

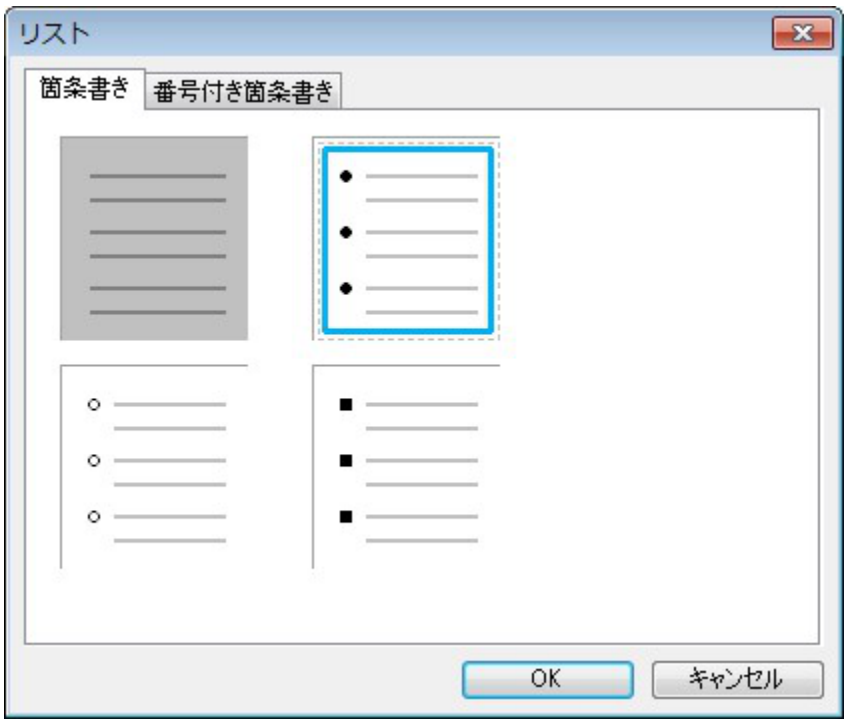
- リスト コマンド: リストが挿入されます。
- 定義済みのフォーマット 文字列 コマンド: 与えられた SPS に対して数値のデータ型を定義します。

関連項目

- プロパティ サイドバー

16.13.1　リストの編集

[リスト] コマンドにより、リストがカーソル位置に挿入されます。コマンドを選択すると、リストのスタイルを選択するためのリスト ダイアログが表示されます（以下のスクリーンショットを参照）。数値を伴うリストの場合、最初に表示される番号を指定することもできます。



関連項目

- リストの作成

16.13.2 定義済みの値フォーマット文字列

数値、date、time、dateTime、または duration のデータ型により定義された（コンテンツ）プレースホルダ、入力フィールド、自動計算には、値フォーマット ダイアログによりカスタム フォーマットを割り当てることができます。値フォーマット ダイアログでは、フォーマットを直接作成、または定義済みのフォーマットをリストから選択することができます。

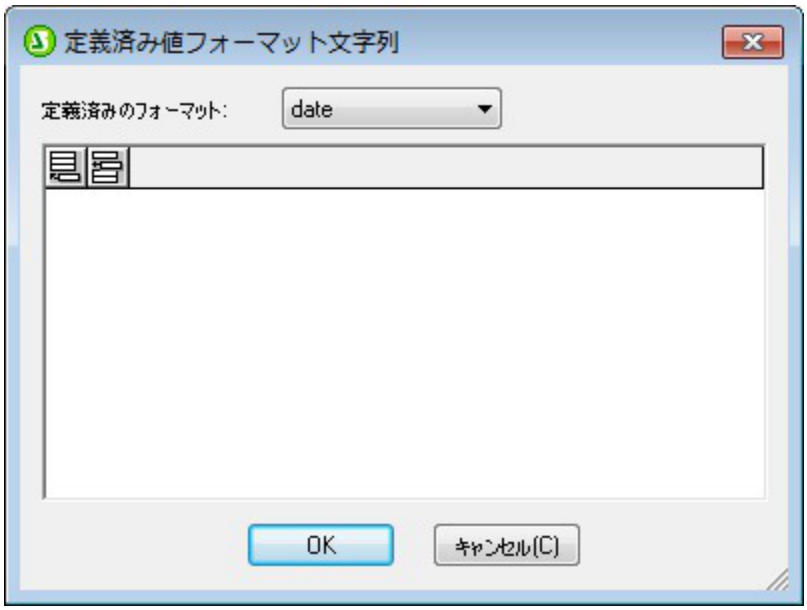
ドロップダウンリスト内で使用することのできる定義済みのフォーマットには次の2種類があります:

- StyleVision に最初から与えられている定義済みフォーマット。
- [**定義済みの値フォーマット文字列**] コマンドを使って、ユーザーにより定義済みフォーマット。ユーザーが定義済みの値フォーマットを作成すると、（アプリケーション全体にではなく）現在開かれている SPS に対してこれらのフォーマットが作成されます。定義済みの値フォーマットを作成した後には、SPS ファイルを保存することで、次回そのファイルが開かれた時にも変更が反映されます。

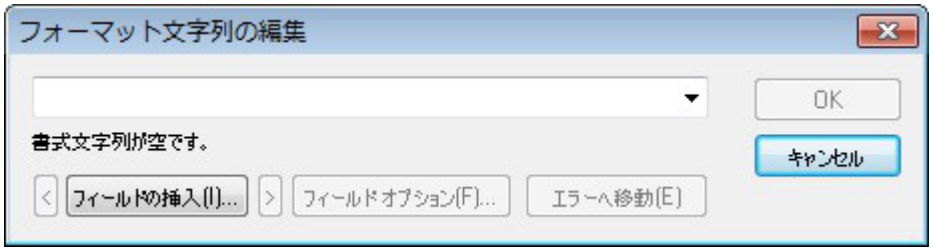
定義済みの値フォーマット文字列を作成する

定義済みの値フォーマット文字列にはデータ型が与えられます。以下の操作により定義済みの値フォーマット文字列を作成することができます:

1. [**プロパティ | 定義済みの値フォーマット文字列**] を選択します。以下のダイアログが表示されます



2. コンボボックスのドロップダウンリストからデータ型を選択し、追加または挿入アイコンをクリックします。フォーマット文字列の編集ダイアログが表示されます。



(C) 2015-2021 Altova GmbH

コンボ ボックスの矢印をクリックすると、そのデータ型に対する StyleVision から得られた定義済みフォーマットのリストが表示されます（以下のスクリーンショットを参照）。

リストからフォーマットを選択して修正することができるほか、入力フィールドへ直接フォーマットを入力することができます。フォーマットを定義するための構文については、値フォーマット のセクションを参照ください。[**フィールドの挿入**] または [**フィールド オプション**] ボタンにより構文を入力することもできます。

3. フォーマットの指定を終えた後には [OK] をクリックして SPS ファイルを保存してください。フォーマット文字列がそのデータ型に対する定義済みのフォーマット リストへ追加され、選択された要素が対応するデータ型の場合（その SPS ファイルの）値フォーマット ダイアログのオプション内に表示されるようになります。

メモ

- 追加することのできるカスタム フォーマット文字列の数に制限はありません。
- 定義済みのフォーマット文字列ダイアログに表示されたフォーマット文字列の順序により、値フォーマット ダイアログに表示されるフォーマット文字列の順序が決定されます。カスタマイズされたフォーマット文字列は最初から同封されている定義済みのフォーマットの上に表示されます。
- カスタム フォーマット文字列を編集するには、定義済みフォーマット文字列ダイアログに表示されたエントリーをダブルクリックしてください。
- カスタム フォーマット文字列を削除するには、定義済み値フォーマット文字列ダイアログにてそのエントリーを選択し、**削除**アイコンをクリックしてください。

関連項目

- 値のフォーマット（数値データ型のフォーマット）

16.14 ツールメニュー

[ツール] メニューにはスペルチェックオプションと、StyleVision をカスタマイズするためのオプションが収められています。そ

ツールメニューのサブメニューには以下のコマンドが収められています:

- スペルチェック
- スペルチェックのオプション
- グローバルリソース
- アクティブな構成
- カスタム化
- オプション

関連項目

- StyleVision のセットアップ

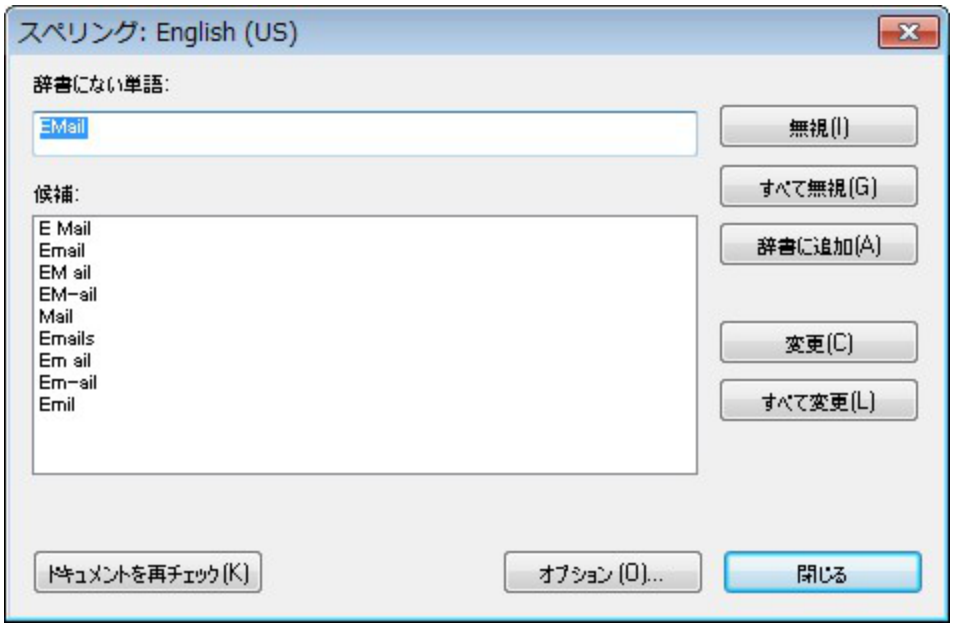
16.14.1 スペルチェック

[スペル チェック] コマンドにより、SPS (デザインビュー) または Authentic View 内のドキュメントにてスペルチェックを実行することができます。スペルチェッカーに内蔵された言語辞書から使用する言語を選択することができます (以下のメモを参照ください)。

メモ Altova ソフトウェアに同封されている内蔵の辞書は Altova の好みにより選択されたものではなく、MPL、LGPL 、または BSD ライセンスといった商用ソフトウェアにおける再配布が認められているライセンス下で利用可能かどうかに大きく依存しております。オープンソースの辞書は数多く存在しますが、GPL ライセンスなどの、より制約の多いライセンス下で配布されています。これらの辞書は、http://www.altova.com/dictionaries。からダウンロード可能なインストーラーにてご利用いただくことができます。辞書に対して適用されるライセンスの条件に同意するか、また辞書がお使いのコンピューターにあるソフトウェアに適したものなのかの判断はお客様に委ねられます。

このコマンドをクリックすることで、以下に示されるダイアログが表示されます。ダイアログでは選択された辞書に存在しない単語が、ドキュメント内で出現する順序に従い [辞書にない単語] フィールドに 1 つずつ表示され、デザインビューでは対応する単語がハイライトされます。

(C) 2015-2021 Altova GmbH



[候補] ペインに表示されたエントリーを選択し、[**変更**] または [**すべて変更**] をクリックすることで、ハイライトされたインスタンス、またはすべてのインスタンスがそれぞれ変更されます（候補リストにある単語をダブルクリックすることで、未知の単語が置き換えられます）。また、[**無視**] ボタンをクリックすることで未知の単語を無視することができるほか、[**すべて無視**] ボタンにより、ドキュメント内にあるその未知の単語すべてのインスタンスを無視されます。[**辞書に追加**] ボタンにより、未知の単語が辞書へ追加されます。未知の単語を辞書へ追加することで、スペルチェッカーはその単語が正しいものであると認識するようになり、辞書に存在しない次の単語が表示されます。[**ドキュメントの再チェック**] ボタンによりドキュメントを最初からチェックすることができるほか、[**閉じる**] ボタンによりダイアログを閉じることができます。

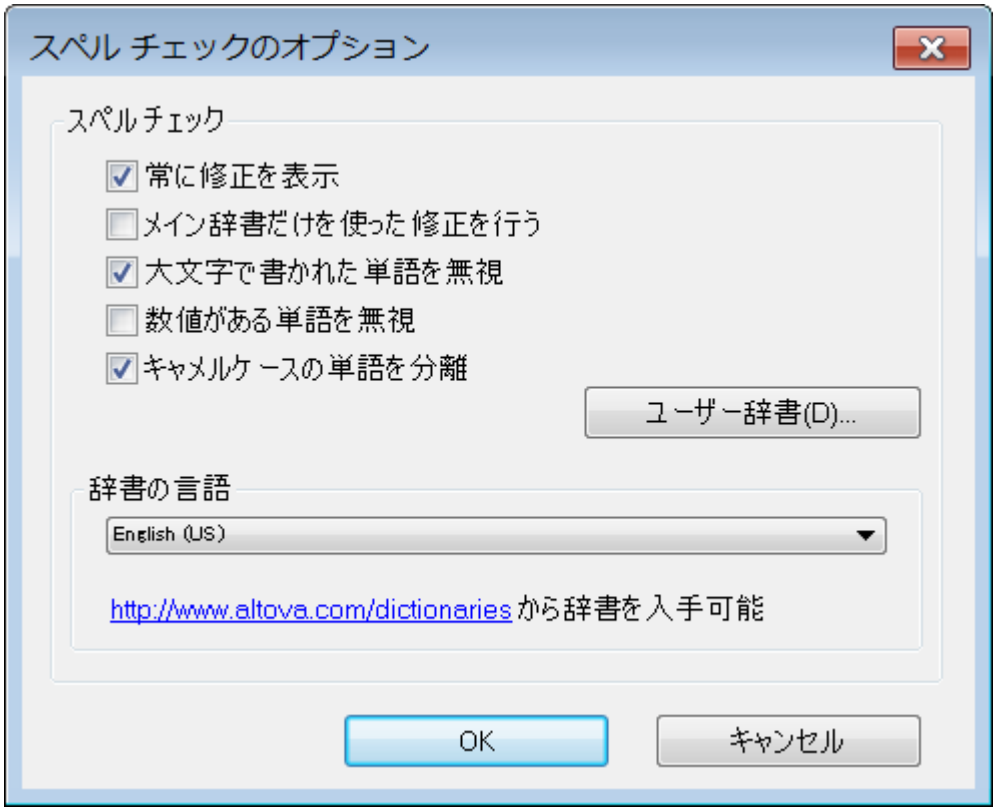
オプション ボタンは、スペル チェックに対して指定することのできる スペルチェック オプション ダイアログを開きます。

関連項目

- スペルチェック オプション

16.14.2　スペル チェックのオプション

[**スペル チェックのオプション**] コマンドにより、スペルチェックのオプションを指定するためのダイアログが表示されます（以下のスクリーンショットを参照）。



常に修正を表示:
このオプションを選択することで、ユーザー辞書とそれ以外の辞書から得られた単語の候補が候補リスト ボックスに表示されます。このオプションを無効にすると、候補が表示されなくなります。

メイン辞書だけを使った修正を行う:
このオプションを選択することで、言語辞書（メイン辞書）だけが使用されます。提案にユーザー辞書は使用されません。「**ユーザー辞書**」ボタンも無効化され、ユーザー辞書の編集を行うことができなくなります。

大文字で書かれた単語を無視:
このオプションを選択することで、すべて大文字で記述された単語がチェックの対象から外されます。

数値を伴う単語を無視:
このオプションを選択することで、数値を含んだすべての単語が無視されます。

キャメルケースの単語を分離:
単語内に大文字の文字が含まれる単語をキャメルケースの単語といいます。例えば"CamelCase" という単語では、"Case" の"C" が大文字で示されているため、キャメルケースの単語として認識されます。キャメルケースの単語は通常辞書に含まれていないため、スペルチェッカーによりエラーとして認識されます。*キャメルケースの単語*オプションにより、キャメルケースの単語を大文字からはじまるパーツに分離して、個々のパーツをチェックすることによりこの問題を回避することができます。このオプションはデフォルトでチェックされています。

辞書の言語:
このコンボ ボックスではスペルチェッカーにて使用される辞書の言語が選択されます。デフォルトの選択はEnglish (US) となっています。Altova Web サイトからはその他の言語の辞書を無料でダウンロードいただけます。

スペル チェッカーで使用される辞書を追加する
辞書の言語は、.aff ファイルと.doc ファイルという2 つのHunspell 辞書ファイルから構成されます。言語辞書は、以下の場所にある

Lexicons フォルダーへインストールされます。全ての言語の辞書は次の場所にある Lexicons フォルダー内にインストールされます: C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons.

Lexicons フォルダー内部には、異なる言語に対応した辞書のファイルが、<言語名>¥辞書ファイル> という構成で各フォルダーに収められます。例えば、2つの(English (British) とEnglish (US)) 英語辞書が以下のように配置されます:

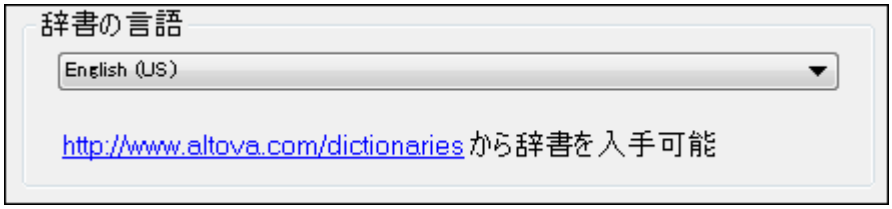
```
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)
\en_GB.aff
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (British)
\en_GB.dic
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (US)\en_US.dic
C:\ProgramData\Altova\SharedBetweenVersions\SpellChecker\Lexicons\English (US)
\en_US.dic
```

スペルチェッカーオプション ダイアログでは、「辞書の言語」コンボボックスにあるドロップダウンリストに利用可能な辞書の言語が表示されます。Lexicons フォルダー以下にある言語サブフォルダーの名前がリストには表示されます。例えば上にある英語辞書の場合、English (British) とEnglish (US) という名前がリストに表示されます。

インストールされた辞書はコンピューターを使っているすべてのユーザー、そして(32 ビットか64 ビットかにかかわらず) 複数のAltova 製品により共有されます。

スペルチェッカーで使用される辞書は以下に示される2 種類の方法で追加することができ、どちらの方法でもファイルをシステムに登録する必要はありません

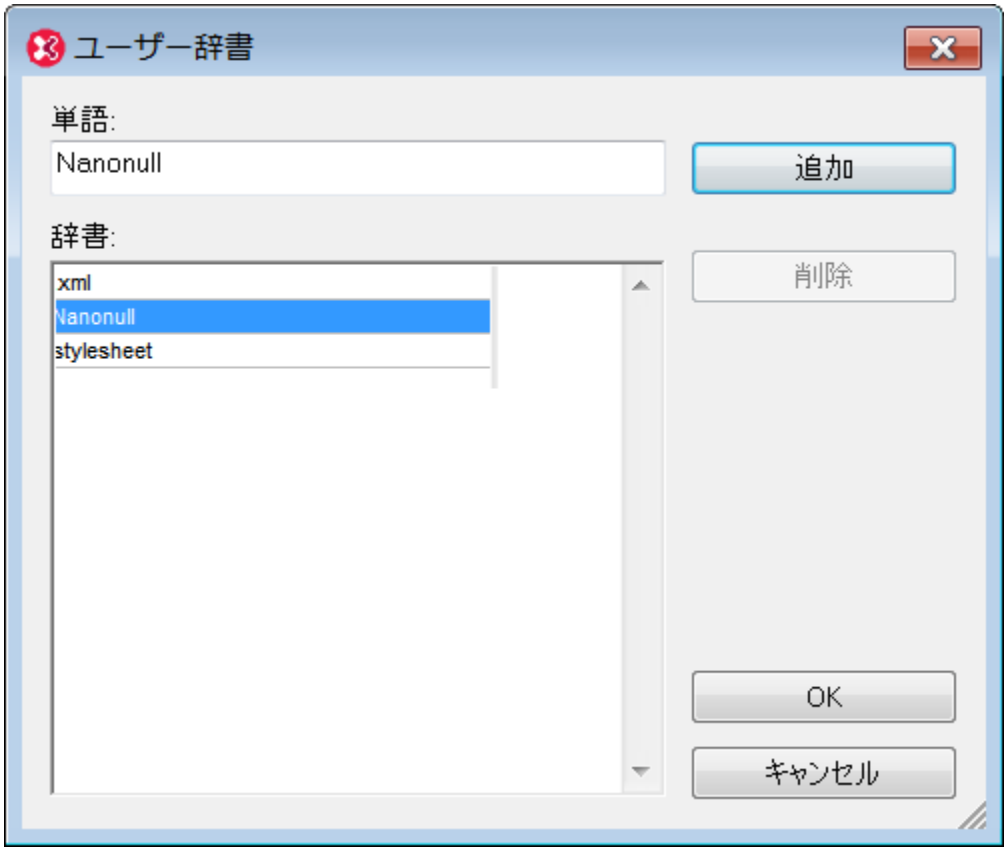
- Hunspell 辞書をLexicons フォルダーの新規サブフォルダーへ追加する。Hunspell 辞書は、例えば http://wiki.services.openoffice.org/wiki/Dictionaries や http://extensions.services.openoffice.org/en/dictionaries.といった場所からダウンロードすることができます(OpenOffice ではOXT という拡張子のZIP アーカイブが配布されているため、ファイル拡張子を .zip へ変換して、回答した .aff と.dic ファイルを ファイルをLexicons フォルダー以下にある言語フォルダーへコピーしてください。Hunspell 辞書はMyspell 辞書をベースにしているため、Myspell 辞書を使用することもできます)
- 複数の言語辞書をコンピューターの適切な場所へインストールするAltova 辞書インストーラーを使用する。スペルチェックのオプションダイアログにあるリンクをクリックすることで、インストーラーをダウンロードすることができます(以下のスクリーンショットを参照)。



メモ　辞書に対して適用されるライセンスの条項に同意するか、そしてお使いのコンピューターにおける辞書の使用が適切なものかの判断は、お客様へ委ねられます。

ユーザー辞書を使用する

コンピューター上の各ユーザーには、ユーザーにより許可された単語を収めるためのユーザー辞書が与えられます。スペルチェックを行うと、言語辞書とユーザー辞書に含まれている単語のリストに対してドキュメント内の単語がチェックされます。ユーザー辞書ダイアログでは、ユーザー辞書に単語を追加、または削除することができます(*以下のスクリーンショットを参照*)。スペルチェックのオプションダイアログにて「ユーザー辞書」ボタンをクリックすることで、この辞書へアクセスすることができます(*このセクションの2番目のスクリーンショットを参照してください*)。



ユーザー辞書へ単語を追加するには、単語フィールドに単語を入力し、「追加」ボタンをクリックしてください。入力された単語がアルファベット順で辞書ペインに追加されます。辞書から単語を削除するには、目的の単語を辞書ペインにて選択し、「削除」ボタンをクリックしてください。その単語が辞書ペインから削除されます。ユーザー辞書ダイアログにおける編集を終えた後には「OK」をクリックすることで、変更点がユーザー辞書へ保存されます。

スペルチェックの際に単語をユーザー辞書へ追加することもできます。 スペルチェックダイアログ は取るアクションのためのプロンプトをポップアップします。「辞書に追加」ボタンをクリックすると、未知の単語がユーザー辞書へ追加されます。

ユーザー辞書は以下の場所にあります: `C:\Users\<user>\Documents\Altova\SpellChecker\Lexicons\user.dic`

関連項目

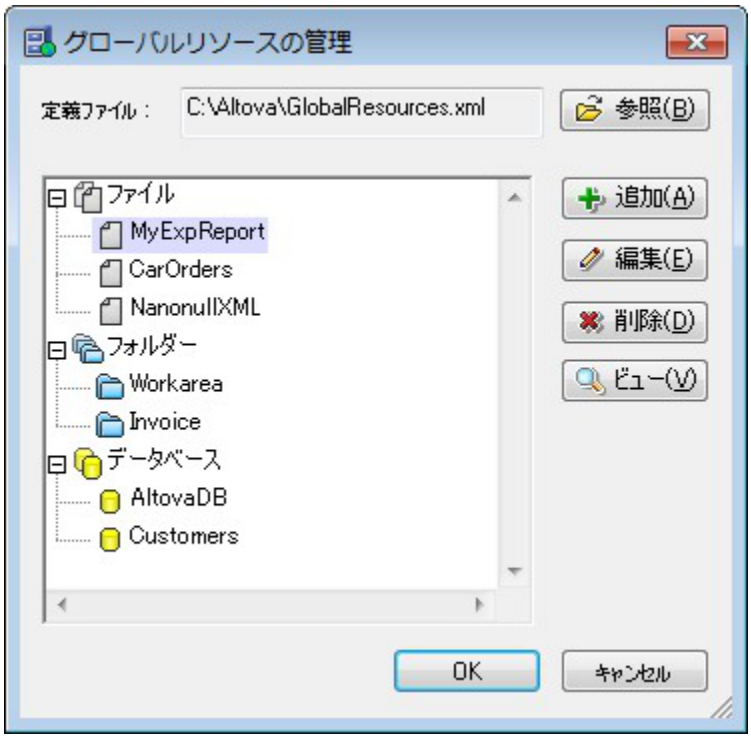
- スペルチェック

16.14.3 グローバルリソース

[グローバル リソース] コマンドにより、以下の操作を行うためのAltova グローバルリソースダイアログが表示されます（以下のスクリーンショットを参照）。

- グローバル リソースを使用するための Altova グローバル リソース XML ファイルを指定する。

- ファイル、フォルダー、データベース グローバル リソース（またはエイリアス）を追加する。
- 各グローバル リソース（エイリアス）に対して様々な構成を指定する。各構成により、特定のリソースがマッピングされます。



グローバル リソースを定義する方法の詳細については、グローバルリソースの定義のセクションを参照ください。

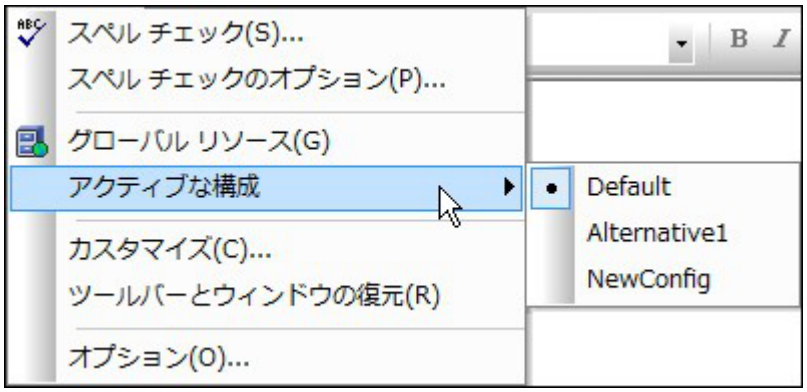
メモ　Altova グローバル リソース ダイアログへは、グローバルリソースツールバー（[**表示 | ツールバー| グローバル リソース**]）からもアクセスすることができます。

関連項目

- Altova グローバルリソース
- ツールバーとステータスバー

16.14.4　アクティブな構成

[**アクティブな構成**] メニュー アイテムには、現在アクティブな グローバルリソースXML ファイルにて定義されたすべての構成が、サブメニューとして含まれています（以下のスクリーンショットを参照）。



現在アクティブな構成は中黒 (•) により示されます。上のスクリーンショットでは、Default という名前の構成がアクティブになっています。構成を選択することで、アクティブな構成を変更することができます。

メモ アクティブな構成は グローバルリソースツールバー([**表示 | ツールバー | グローバル リソース**]) からも選択することができます。

関連項目

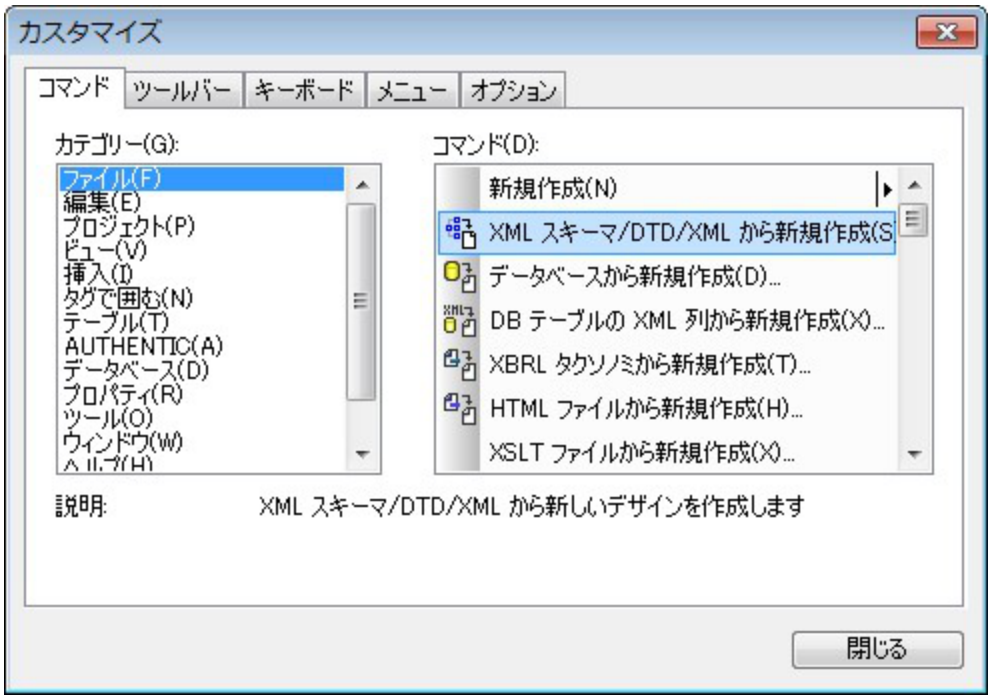
- Altova グローバルリソース
- ツールバーとステータスバー

16.14.5 カスタマイズ

[カスタマイズ] コマンドにより、特定のニーズに対して StyleVision をカスタマイズすることができます。

コマンド タブ
カスタマイズ ダイアログの [**コマンド**] タブでは、個々のコマンドをメニュー バーやツール バーに配置することができます。

(C) 2015-2021 Altova GmbH



メニューバーやツールバーへ**コマンドを追加**するには、コマンド タブのコマンド ペイン内にあるコマンドを選択し、選択対象をメニューバーまたはツールバーへドラッグしてください。コマンドを配置することができる場所にはI が表示され、ドロップすることでコマンドがその位置に配置されます。コマンドを配置することができない場所では、X マークが表示されます。コマンドに関連付けられたアイコンが存在する場合、ツールバーにはそのアイコンが表示され、そうでない場合にはテキスト形式のコマンドが作成されます。メニューバーまたはツールバーへコマンドを追加すると、コマンドを右クリックすることで表示されるメニューから、その外観を変更することができます。

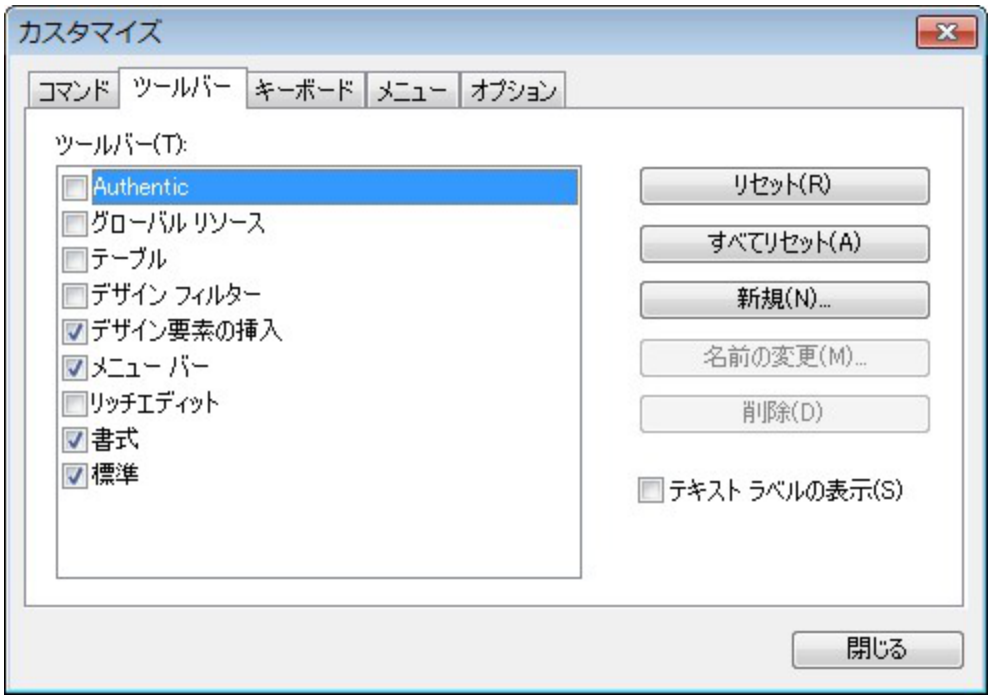
メニューバーやツールバーの**アイテムを削除**するには、カスタマイズ ダイアログが開かれている状態で削除するアイテムを右クリックし、[**削除**] を選択してください。

メモ

- 上記のカスタマイズはアプリケーションに対して適用され、StyleVision にてドキュメントが開かれているかどうかにかかわらず適用されます。
- StyleVision がインストールされたときの状態にメニューやツールバーをリセットするには、ツールバータブを選択し、[**リセット**] または [**すべてリセット**] ボタンをクリックしてください。

ツールバータブ

[**ツールバー**] タブでは、特定のツールバーを有効化または無効化して、ツールバーアイテムのテキスト ラベルを表示、メニューバーやツールバーをインストール時の状態へリセットすることができます。



StyleVision のインターフェイスにはメニュー バーとオプションのツール バー (Authentic、デザイン フィルター、フォーマット、標準、テーブル、目次) が表示されます。

各ツール バーはコマンドのグループとして分けることができます。コマンドは、コマンド タブが開かれている状態でツール バーに追加することができます。ツール バーをドラッグすることで、画面上の好きな場所へドッキングすることができるほか、ツール バーのタイトル バーをダブルクリックすることでツール バーのドッキング状態を切り替えることができます。

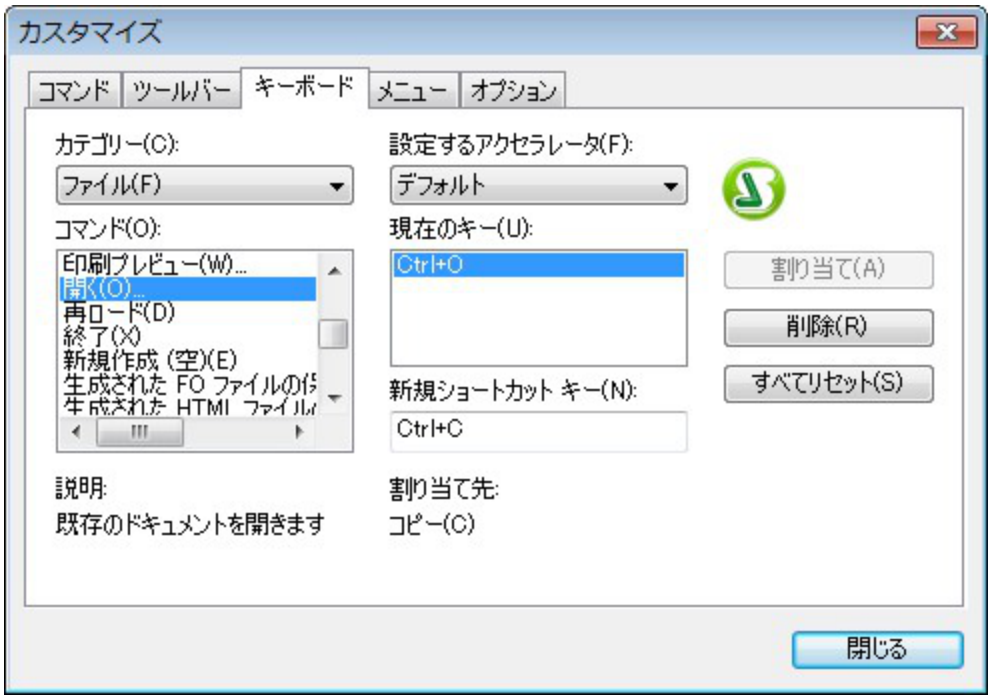
カスタマイズ ダイアログのツール バー タブでは、チェック ボックスをクリックすることで、ツール バーを表示/隠すことができます。ツール バーを (ツール バー タブにて) 選択し、[**テキスト ラベルを表示**] チェック ボックスをクリックすることで、そのツール バーに含まれているアイテムのテキスト ラベルが表示されます。[**リセット**] ボタンをクリックすることで、選択されたツール バーを StyleVision がインストールされた際の状態へリセットすることもできます。[**すべてリセット**] ボタンをクリックすると、すべてのツール バーとメニュー バーがリセットされます。

メニュー バーに関する注意点
メニュー バーにコマンドを追加し、アイテムをメニュー バーから削除することができます: 上記のコマンドを参照してください。StyleVision のインストール時の状態にメニュー バーをリセットするには、カスタマイズ ダイアログのツール バータブ内のメニュー バーを選択し、**リセット** ボタンをクリックします。(**すべてをリセットする** ボタンをクリックするとツール バーをリセットします)。

キーボード タブ

[**キーボード**] タブでは、任意の StyleVision コマンドに対してキーボード ショートカットを定義または変更することができます。

(C) 2015-2021 Altova GmbH



コマンドショートカットをコマンドへ割り当てる

1. カテゴリーコンボボックスから目的のコマンドがあるカテゴリーを選択します。
2. コマンド リスト ボックスにてショートカットを割り当てるコマンドを選択します。
3. [新規ショートカット キー] 入力フィールドをクリックして、コマンドを起動するためのショートカット キーを押下します。[新規ショートカット キー] 入力フィールドにショートカットが表示されます。入力されたショートカットが他のコマンドへ既に割り当てられている場合、入力フィールドの下にそのコマンド名が表示されます（例えば上のスクリーンショットでは、Ctrl+C が[コピー] コマンドへ既に割り当てられているため、ファイルを開くコマンドへの割り当てを行うことはできません）。入力したショートカット キーを取り消すには、Ctrl、Alt、またはShift キーのいずれかを押下してください。
4. [**割り当て**] ボタンをクリックすることで、入力されたショートカットがコマンドに対して割り当てられます。ショートカットが[現在のキー] リスト ボックスに表示されます。

ショートカットの割り当てを変更（または削除）する

1. 削除するショートカットが割り当てられているコマンドを選択します。
2. 削除するショートカットを[現在のキー] リスト ボックスにてクリックします。
3. 選択可能になった[**削除**] ボタンをクリックします。

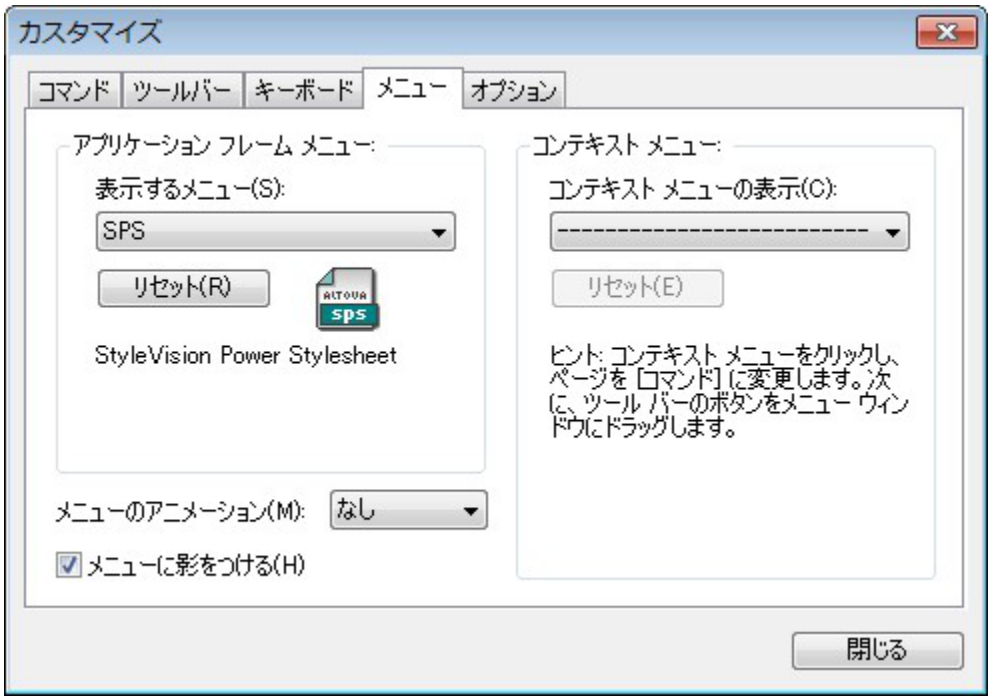
キーボード の割り当てをすべてリセットする

1. [**すべてリセット**] ボタンをクリックすることで、インストール時のオリジナルショートカットへ割り当てがリセットされます。すべてのキーボード割り当てを本当にリセットしても良いかの確認を行うダイアログが表示されます。
2. [はい] をクリックすることで、すべてのショートカット割り当てがリセットされます。

設定するアクセラレーター
現在機能は割り当てられていません。

メニュータブ

[**メニュー**] タブではメイン メニューや（右クリックにより表示される）コンテキスト メニューをカスタマイズすることができます。メニュー バーには、ドキュメントが開かれている時に表示されるデフォルト メニューとSPS ドキュメントが開かれている時に表示されるSPS メニューがあります。



メニューをカスタマイズする

1. カスタマイズするメニュー バーを選択します（スクリーンショットではSPS メニューが選択されています）。
2. [コマンド] タブをクリックして、目的のコマンドをメニュー バーへドラッグしてください。

メニューからコマンドを削除する

1. アプリケーションフレームメニューペインから、（ドキュメントが開かれていない状態で使用可能なメニューを表示する）*デフォルト* または（ドキュメントが開かれている状態で使用可能なメニューを表示する）*SPS* を選択します。
2. カスタマイズダイアログが開かれている状態で（i）アプリケーションのメニュー バーから削除するメニュー、または（ii）これらのメニューの一つから削除するコマンドを選択します。
3. または（i）メニューからメニュー バー、またはメニューコマンドをドラッグ、または（ii）メニューを右クリック、または「**削除**」を選択します。

メニュー バーをリセットする

1. アプリケーションメニューペインのコンボボックス内でリセットするメニューエントリを選択します。
2. メニュー名の下にある [**リセット**] ボタンをクリックします。メニュー バーをリセットしてもよいか確認するプロンプトが表示されます。

コンテキストメニューをカスタマイズする（右クリックメニュー）

1. コンテキスト メニューのコンボ ボックスから目的のコンテキスト メニューを選択します。
2. [**コマンド**] タブをクリックして、表示されたコンテキスト メニューへコマンドをドラッグします。

コンテキストメニューからコマンドを削除する

1. コマンドまたはコマンドを表すアイコンを右クリックします。
2. ポップアップ メニューから [**削除**] オプションを選択します。または、コンテキストメニューからコマンドをドラッグして、マウスポインターの下にチェック マークが表示されたらすぐドロップします。

コンテキスト メニューをリセットする

1. コンテキスト メニューのコンボ ボックスから目的のコンテキスト メニューを選択します。
2. コンテキスト メニュー名の下にある [**リセット**] ボタンをクリックします。リセットしてもよいか確認するプロンプトが表示されます。

コンテキスト メニュー ウィンドウを閉じる

- タイトル バーの右側にある閉じるアイコンをクリックするか、

- カスタマイズ ダイアログ ボックスの [閉じる] ボタンをクリックしてください。

メニューのアニメーション
メニューのアニメーション オプションにより、メニューがクリックされたときの表示方法を指定することができます。メニューのアニメーションにあるドロップダウン リストから、目的のオプションを選択してください。

メニューに影をつける
メニューの周りに影をつけたい場合にこのオプションを選択してください。すべてのメニューに影が加えられます。

オプション タブ

[オプション] タブでは、ツール バーにあるその他の機能をカスタマイズすることができます。

[ツール バーに画面のヒントを表示] オプションを有効にすることで、ツール バー アイテムのヒントが表示されます。画面のヒント オプションには、ショートカットも画面に表示するかを指定するためのサブオプションが与えられています。

関連項目
- ツール バー
- ビュー | ツール バー
- ユーザーインターフェイス

16.14.6 ツール バーとウィンドウの復元

このコマンドにより、ツール バー、ウィンドウ、入力ヘルパー、そのた GUI コンポーネントが、デフォルトの状態に戻されます。変更を反映するには、StyleVision を再起動する必要があります。

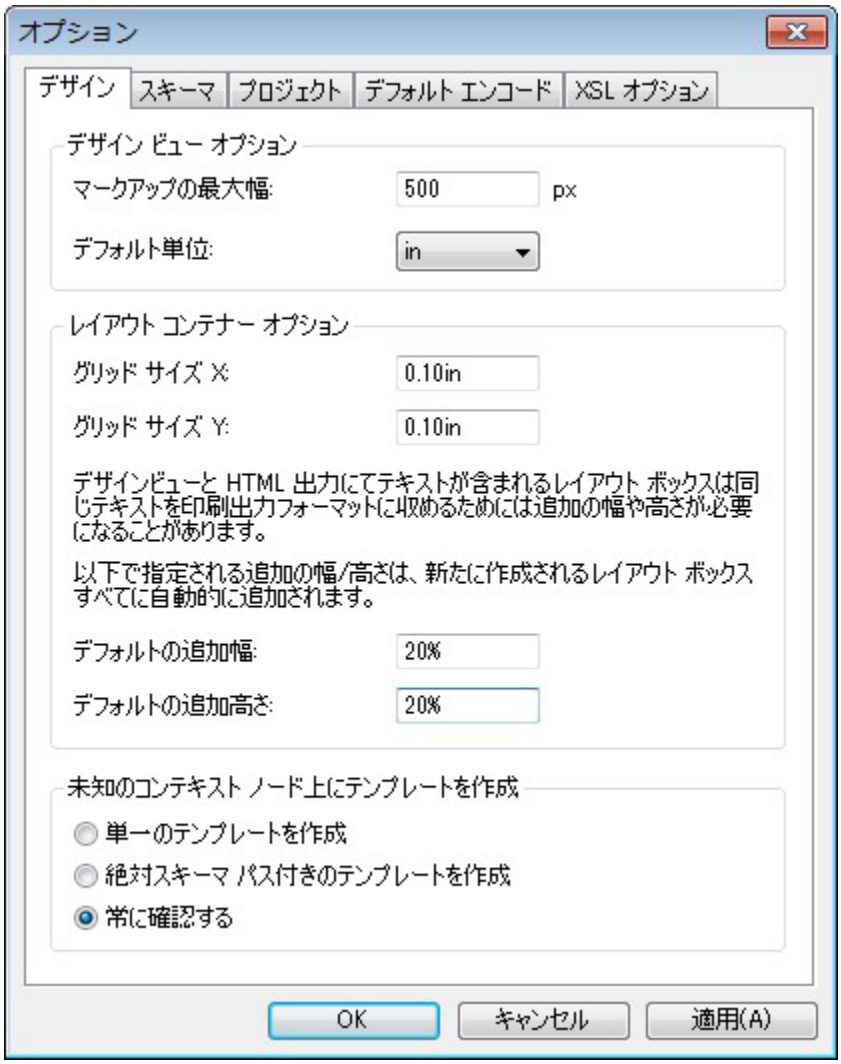
関連項目
- ツール バー
- ビュー | ツール バー
- ユーザーインターフェイス

16.14.7 オプション

[オプション] コマンドにより、HTML 出力 ファイルのエンコードを指定するためのダイアログが表示されます。

デザインビュー オプション

デザイン タブでは、アプリケーション全体で使用される一般的なデザイン オプションをセットすることができます (以下のスクリーンショットを参照)。



以下のオプションをセットすることができます:

- マークアップの(ピクセル) 最大幅。ピクセル数として正の整数を入力してください。
- レイアウト コンテナーのグリッド サイズ。指定された長さが、対応するグリッドの軸に表示されるグリッド間の幅となります。
- レイアウト ボックスのデフォルト追加幅と追加高さ。入力された長さが出力フォーマットにてテキストのレンダリングを行うために必要な追加長さとしてレイアウト ボックスに追加されます。これらの値はパーセンテージとして、または絶対長さ単位として入力することができます。
- コンテキスト ノードが未知の場所にノード テンプレートが作成された時のデフォルトの振る舞い。このオプションは、デザインのスキーマ ソース内にあるコンテキスト内部に配置することができないアイテムに対してテンプレートが作成されるようなユーザー定義テンプレートに適用されます。このようなユーザー定義テンプレート内部でノードが作成されると、(i) その名前だけで、または(ii) スキーマ ルートからのフル パスとともにノードを作成することができます。デフォルトの振る舞いとして、これらオプションのどれかを選択することができるほか、このような状況でユーザーへの確認を求めるように指定することができます。デフォルトの選択は[常に確認する] です。

スキーマオプション

スキーマ ツリーでは、要素や属性をアルファベット順に表示することができます。スキーマ タブにて対応するチェック ボックスにチェックを入れてください(以下のスクリーンショットを参照)。デフォルトでは、属性がアルファベット順で、要素がスキーマの構造に従うかたちで表示されるようになっています。

プロジェクトオプション

プロジェクト サイドバーにて XML ファイルや XSD ファイルがダブルクリックされると、オプション ダイアログのプロジェクト タブにてセットされたオプションにしたがって、以下のアクションのいずれかが実行されます（以下のスクリーンショットを参照）: (i) ファイルを XMLSpy で編集する、(ii) 選択されたファイルをベースに新たなデザインを作成する、(iii) 実行するアクションを選択するよう確認する。

デフォルトのエンコード

[デフォルト エンコーディング] タブでは、各種出力にて使用されるデフォルトのエンコーディングを個別にセットすることができます。エンコーディングにより各種文字セットに対するコードポイントが指定されます。各コンボボックスのドロップダウン リストには、エンコーディング オプションのリストが表示されます。各種出力にて使用するエンコーディングを選択し、[OK] をクリックしてください。以降新規作成されるすべての SPS にて使用されるようになります。

XSLT-for-HTML では、以下の位置に出力エンコーディング情報が登録されます:

- スタイルシートの xsl:output 要素にある encoding 属性:
```
<xsl:output version="1.0" encoding="UTF-8" indent="no" omit-xml-declaration="no"
media-type="text/html" />
```
- HTML ヘッダーの content-type meta 要素にある charset 属性:
```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

XSLT-for-RTF では、スタイルシートの xsl:output 要素にある encoding 属性に出力エンコーディング情報が登録されます:
```
<xsl:output version="1.0" encoding="ISO-8859-1" indent="no" method="text" omit-xml-
declaration="yes" media-type="text/rtf" />
```

メモ これらの設定は新規 SPS にて使用されるデフォルトのエンコーディングで、このダイアログから現在開かれている SPS のエンコーディングを変更することはできません。現在開かれている SPS のエンコーディングを変更するには、ファイル| プロパティ コマンドを使用してください。

MobileTogether デザイン

SPS ファイルを MobileTogether デザイン（MTD）ファイルに変換するために下にリストされたオプションを使用することができます。（変換は MobileTogether デザインファイルとしてエクスポートする コマンドを使用して行われます）。

- （例えば太字または斜線の要素を含む パラグラフ要素）インラインフォーマットを含む構文テキストと パラグラフレベルのテキストはデフォルトで、テキストがブロックレベルのテキストのフォーマットを持つ箇所で単一のラベルに変換されます。各ラベルが元のテキストフラグメントに対応するフォーマットを持つ箇所で、この種類のテキストを複数の MTD ラベルに分割するオプションを設定することができます。
- SPS にブランチを持つ条件が存在する場合、MTD への変換はデフォルトで 条件の全てのブランチを変換し、MTD ドキュメント内の他のシーケンスの 1 つ下に置かれます。条件（と全てのブランチ）の変換を抑制することを任意で選択することができます。

16.15 ウィンドウ メニュー

[**ウィンドウ**] メニューには、StyleVision ウィンドウがGUI に表示される方法を指定するためのコマンドが収められています（[重ねて表示]、[上下に並べて表示]、[左右に並べて表示]）。ウィンドウを最大化するには、そのウィンドウの最大化ボタンをクリックしてください。

[**ウィンドウ**] メニューには、StyleVision ウィンドウがGUI に表示される方法を指定するためのコマンドが収められています（[重ねて表示]、[上下に並べて表示]、[左右に並べて表示]）。ウィンドウを最大化するには、そのウィンドウの最大化ボタンをクリックしてください。

更に、現在開かれているすべてのドキュメント名が表示されます。他のウィンドウをアクティブにするには、対応するドキュメント名をクリックしてください。

ウィンドウダイアログ

ドキュメントのリスト下部には[ウィンドウ] ダイアログへアクセスするためのコマンドが用意されています。このエントリーをクリックすることで[ウィンドウ] ダイアログが開かれ、現在開かれているウィンドウのリストと選択されたウィンドウに対して適用することのできるコマンドが表示されます（名前をクリックすることで、そのウィンドウが選択されます）。

警告: [ウィンドウ] ダイアログから抜けるには、[OK] をクリックして、[**ウィンドウを閉じる**] ボタンを**クリックしないように**してください。[ウィンドウを閉じる] ボタンをクリックすると、ウィンドウ ダイアログにて現在選択されているウィンドウが閉じられます。

関連項目

- ユーザーインターフェイス

(C) 2015-2021 Altova GmbH

16.16 ヘルプメニュー

StyleVision[ヘルプ] メニューにはStyleVision のヘルプ マニュアルへアクセスするためのコマンドやStyleVision に関する情報へアクセスするためのコマンド、そして Altova W eb サイト上のサポート ページへのリンクが収められています。ヘルプ メニューには登録ダイアログも含まれており、購入した製品のライセンス キーコードを入力することもできます。

[ヘルプ] メニューのコマンドについては、以下のサブセクションで記述されます:

- 目次、インデックス、検索
- ライセンス登録、注文フォーム、レジストレーション、最新情報のチェック
- 他のコマンド

16.16.1 目次、インデックス、検索

▼ 目次

▤ *説明*

ヘルプウィンドウの左側のペインに目次を表示した、StyleVision の画面上のヘルプマニュアルを開きます。目次はヘルプドキュメント全体の概要を表示しています。目次のエントリをクリックしてトピックに移動することができます。

▼ インデックス

▤ *説明*

ヘルプウィンドウの左側のペインにキーワード インデックスを表示したStyleVision の画面上のヘルプマニュアルを開きます。目次はヘルプドキュメント全体の概要を表示しています。インデックスはキーワードをリストし、キーワードをダブルクリックすることでトピックへ移動することができます。キーワードが 1 つ以上のトピックにリンクされている場合は、トピックのリストが表示されます。

▼ 検索

▤ *説明*

ヘルプウィンドウの左側のペインに検索ダイアログを表示したStyleVision の画面上のヘルプマニュアルを開きます。単語を検索するには、入力フィールドに検索対象を入力して、(i) 「Return」 を押す、または(ii) 「**トピックのリスト**」をクリックします。を押します。ヘルプシステムは、ヘルプドキュメント全体で全文検索を行いヒットしたリストを返します。アイテムを表示するためにはアイテムをダブルクリックします。

16.16.2 ライセンス登録、注文フォーム、レジストレーション、最新情報のチェック

▼ ソフトウェアのライセンス認証

▤ *説明*

Altova 製品ソフトウェアをダウンロードすると、無料評価キーまたは購入されたライセンスキーを使用して、製品にライセンスを供与、または、ライセンスの認証を行うことができます。

- *無料 評価ライセンス*初めて製品のダウンロードとインストールを行うと、ソフトウェアライセンス認証ダイアログが表示さ

れます。ダイアログでは無料 評価 ライセンスをリクエストすることができます。ユーザーの名前、所属会社名、そして電子メールアドレスを表示されるダイアログに入力し **リクエスト** をクリックします。ライセンスファイルが入力された電子メールアドレスに送信されます。この手順には数分を要します。ライセンスファイルを適切な場所に保存します。**リクエスト** をクリックすると、リクエストダイアログの下に入力フィールドが表示されます。このフィールドはライセンスファイルの パスを取ります。ライセンスファイルを参照 またはライセンスファイルへの パスを入力し「OK」をクリックします。(「ソフトウェアのライセンス認証ダイアログ」内で「**新規のライセンスをアップロードする**」をクリックしてライセンスファイルへの パスを入力するダイアログにアクセスすることができます。)ソフトウェアは30日の間アンロックされます。

- *永続的なライセンスキー*: ソフトウェアライセンス認証ダイアログには永続的なライセンスキーを購入するためのボタンが含まれています。このボタンをクリックすると、製品の永続的なライセンスキーを購入することのできるAltova オンラインショップに移動することができます。受信する電子メールはライセンスデータを含むライセンスファイルの書式で送信されています。3つの種類の永続的なライセンスが存在します: インストール済み、同時使用ユーザー、名前を持つユーザー。*インストール済みのライセンス* は単一のコンピューター上のソフトウェアのロックを解除します。*インストール済みのライセンス を* N 台のコンピューターのために購入するとN 台までのコンピューター上でソフトウェアを使用することができます。*同時使用ユーザーライセンス* はN 人の同時使用ユーザーにN 人のユーザーに同時にソフトウェアを使用することを許可します 。(10N 台のコンピューターにソフトウェアをインストールすることができます)。 *名前を持つユーザーライセンス*は*特定のユーザー*が5台の異なるコンピューター上でソフトウェアを使用することを許可します。ソフトウェアのライセンスを認証するには「**新規のライセンスをアップロードする**」をクリックして表示されるダイアログ内でライセンスファイルを参照、またはライセンスファイルの パスを入力し「OK」をクリックします。

メモ 複数のユーザーライセンスのために各ユーザーは各自の名前を入力するようにプロンプトされます。

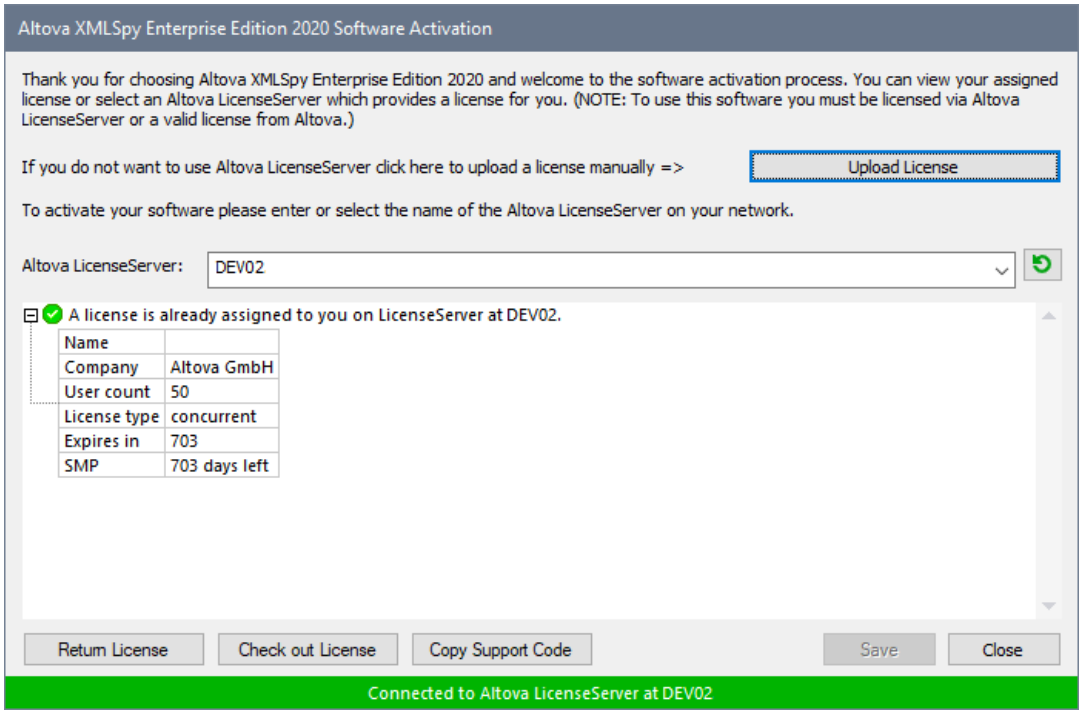
> *ライセンスの電子メールとAltova 製品へのライセンス供与(有効化) の異なる方法*
> Altova から受信するライセンス電子メールにはライセンスファイルが添付ファイルとして含まれています。ライセンスファイルは `.altova_licenses` ファイル拡張子を有しています。
>
> Altova 製品のライセンスを認証するには、以下のうち1つ行ってください:
>
> - 適切な場所にライセンスファイル(`.altova_licenses`) を保存し、ライセンスファイルをダブルクリックし、表示されるダイアログに必要な情報を入力し、「**キーの適用**」をクリックして完了します。
> - ライセンスファイル(`.altova_licenses`) を適切な場所に保存します。Altova 製品内では、「**ヘルプ | ライセンス登録**メニューコマンドを選択し、**新規のライセンスをアップロード**します。ライセンスファイルへの パスを入力し「OK」をクリックします。
> - 適切な場所にライセンスファイル(`.altova_licenses`) を保存し、Altova LicenseServer のライセンスプールにアップロードします。以下を行うことができます: (i) 製品のソフトウェアライセンス認証ダイアログからAltova 製品からライセンスを取得します。*(以下を参照)*または(ii) Altova LicenseServer から製品へのライセンスを割り当てます。*LicenseServer の使用の詳細に関しては、下記のトピックを参照してください。*

ソフトウェアライセンス認証ダイアログ(*下のスクリーンショット*) は、「**ヘルプ | ソフトウェア アクティベーション**」をクリックすることにより常にアクセスすることができます。

以下の方法によりソフトウェアをアクティブ化することができます:

- *ソフトウェアライセンス認証ダイアログでライセンスを登録する方法*。ダイアログ内で、「**新規のライセンスをアップロード**」をクリックして、ライセンスファイルを参照し選択します。「OK」をクリックしてライセンスファイルへの パスを確認します(複数のユーザーライセンスの場合は個人の名前です)。「**保存**」をクリックして完了します。
- *ネットワーク上のAltova LicenseServer を使用してライセンス供与する方法*: ネットワーク上のAltova LicenseServer を使用してライセンスを取得するにはソフトウェアのライセンスの認証ダイアログの下にある **Altova LicenseServer を使用する**をクリックします。使用するLicenseServer がインストールされているマシンを選択します。License Servers の自動検知はLAN 上で配信が送信されることを意味します。これらの配信がサブネッ

トに制限されているため、License Server は自動検知のためのクライアントマシンと同じサブネット上に存在する必要があります。自動検知が作動しない場合、サーバーの名前を入力します。Altova LicenseServer はライセンスプール内で Altova 製品のためのライセンスを有している必要があります。LicenseServer プール内に存在する場合、ソフトウェアライセンス認証ダイアログ内に表示されます（*Altova XMLSpy 内のダイアログで表示されている例を参照してください*）。「**保存**」をクリックしてライセンスを取得します。



マシン固有のライセンスが LicenseServer からインストールされると、7 日間は、LicenseServer に戻すことができません。7 日過ぎると、（「**ライセンスを戻す**」をクリックして）マシンのライセンスを LicenseServer に戻すことができ、このライセンスは、他のクライアントにより LicenseServer から取得することができます。LicenseServer 管理者は、LicenseServer の Web UI を使用して、取得されたライセンスの割り当てを解除することができます。ライセンスの返却は、マシン固有のライセンスのみに適用され、現在使用中のライセンスには適用されないことに注意してください。

ライセンスのチェックアウト

ライセンスが製品マシン上に保管されるように、ライセンスをライセンスプールから 30 日間チェックアウトすることができます。これにより、オフラインで作業することが可能になります。この機能はとても役に立ちます。Altova LicenseServer にアクセスできない環境（例えば、旅行中に Altova 製品がインストールされたラップトップコンピューターで作業する）場合などが挙げられます。ライセンスはチェックアウトされていますが、LicenseServer は、ライセンスが使用中と表示し、ライセンスは他のマシンで使用することができません。ライセンスはチェックアウトの期間が終わると自動的にチェックインされた状態を戻します。または、チェックアウトされたライセンスはソフトウェアのライセンスの認証ダイアログのボタンを使用して「**チェックイン**」することができます。

ライセンスをチェックアウトするには以下をおこないます：（i）ソフトウェアのライセンスの認証ダイアログで「**ライセンスのチェックアウト**」をクリックします（*上のスクリーンショット参照*）。（ii）ライセンスのチェックアウトダイアログ内から、チェックアウトの期間を選択し、「**チェックアウト**」をクリックします。ライセンスがチェックアウトされます。ライセンスのチェックアウト後2 つの状態が発生します：（i）ソフトウェアのライセンス認証ダイアログは時刻およびチェックアウトの期限を含むチェックアウトに関する情報を表示します。（ii）ダイアログ内の「**ライセンスのチェックアウト**」ボタンは「**チェックイン**」ボタンに変更されます。「**チェックイン**」ボタンをクリックして、ライセンスをチェックインすることができます。チェックアウト期間の期限が切れると、ライセンスは自動的にチェックイン状態に戻されるため、選択したチェックアウトの期間がオフラインで作業する期間をカバーするように確認してください。

メモ ライセンスのチェックアウトを可能にするには、LicenseServer 上でチェックアウト機能が有効化されている必要があります。チェックアウトを試みる際この機能が有効化されていない場合、エラーメッセージが表示されます。この場合、LicenseServer 管理者に連絡してください。

サポートコードのコピー
「**サポートコードのコピー**」をクリックして、ライセンスの詳細をクリップボードにコピーしてください。これはオンラインサポートフォームを使用してサポートをリクエストする際に必要なデータです。

Altova LicenseServer を使用することにより、IT 管理者は、リアルタイムでネットワーク上の全てのライセンスの概要、および、クライアントの割り当てと、クライアントのライセンスの使用状況を確認することができます。LicenseServer を使用する利点は、ですから、多数の Altova ライセンスを管理することのできる管理機能です。Altova LicenseServer は、Altova Web サイトで無料で提供されています。Altova LicenseServer および Altova LicenseServer を使用したライセンスの供与に関する詳細は、Altova LicenseServer ドキュメントを参照してください。

注文フォーム

説明

ソフトウェア製品のライセンス許与バージョンを注文する準備が整っている場合、(*前のセクション参照*) ソフトウェアライセンス認証ダイアログ内の「**永久ライセンスの購入**」ボタン、または「**注文フォーム**」コマンドを使用して Altova オンラインショップに移動して注文することができます。

登録

説明

Altova 製品登録ページをブラウザーのタブに表示します。Altova ソフトウェアを登録することにより、最新の製品の情報が得られます。

更新のチェック

説明

Altova サーバーに接続して、より新しいバージョンの製品が利用可能かどうかチェックし、その結果を表示します。

16.16.3 他のコマンド

サポートセンター

説明

インターネット上にある Altova サポートセンターへのリンクとなっています。サポートセンターには FAQ やディスカッションフォーラムが含まれており、問題の解決方法を探り、Altova の技術サポートスタッフへアクセスすることができます(現在英語のみの提供となります)。

WEB 上の FAQ

説明

インターネット上にある Altova の FAQ へのリンクとなっています。FAQ データベースは Altova のサポートスタッフにより常時更

新されています。

コンポーネントのダウンロード

説明

インターネット上にあるAltova のコンポーネントダウンロードセンターへのリンクとなっています。このリンク先から様々なコンポーネントソフトウェアをダウンロードして、Altova 製品とともに使用することができます。ソフトウェアコンポーネントはXSLT やXSL-FO プロセッサーからアプリケーションサービスプラットフォームまで、幅広く提供されています。コンポーネントダウンロードセンターにてご利用になれるソフトウェアは、通常無料でご利用になれます。

インターネット上のStyleVision

説明

インターネット上にあるAltova Web サイトへのリンクとなっています。Altova Web サイトでは、StyleVision や関連するテクノロジーについて確認することができます。

StyleVision について

説明

スプラッシュ画面と製品のバージョン番号が表示されます。StyleVision の64 ビットバージョンを使用している場合、これはアプリケーション名の後のサフィックス（x64）により示されています。32 ビットバージョンにはサフィックスは存在しません。

17 プログラマーの レファレンス

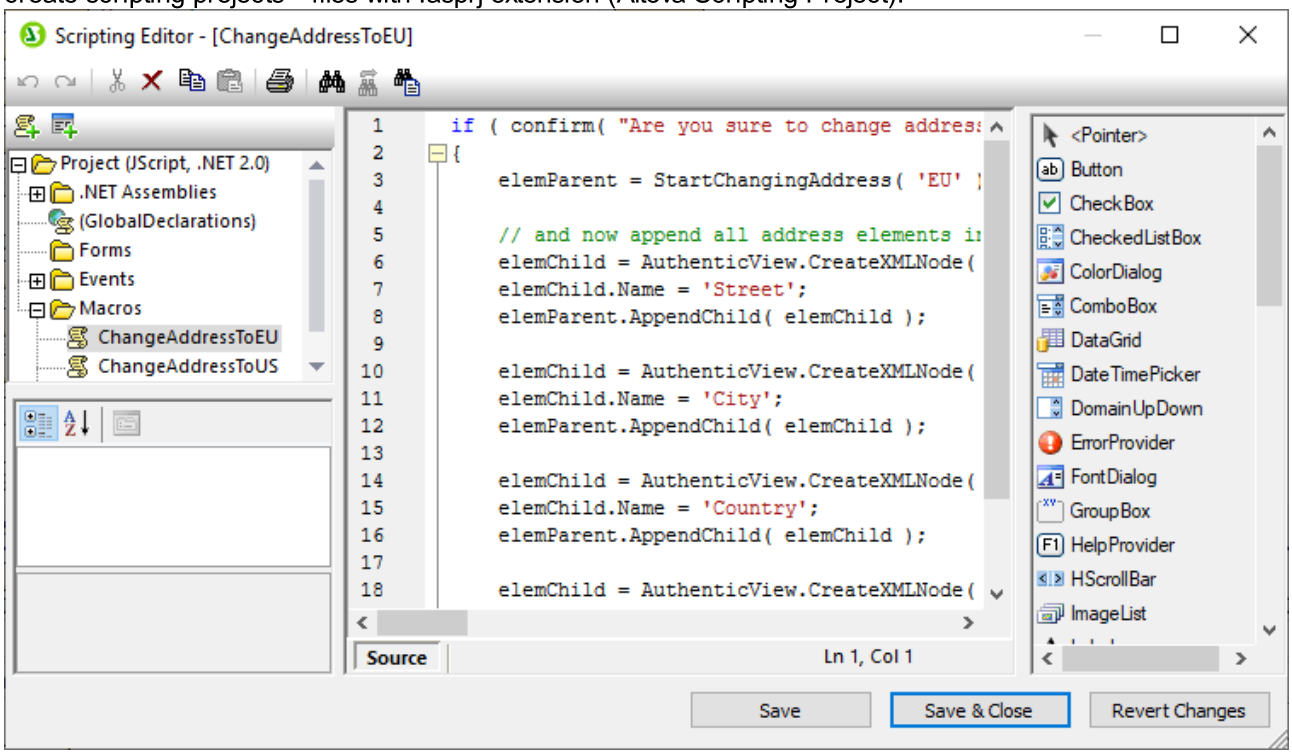
StyleVision を自動化サーバーとして使用する
StyleVision は自動化することのできるサーバーです。これは、他のアプリケーションにプログラムすることのできる(自動化クライアントと称される)オブジェクトを公開します。この結果、自動化クライアントは、自動化サーバーにより使用することのできるオブジェクトと機能に直接アクセスすることができます。 StyleVision の機能を活用することができるため、これはクライアントの自動化に取り有益です。例えば、自動化クライアントは、StyleVision を使用して、XSLT ファイルを SPS から生成することができます。開発者は、ですから、アプリケーションを StyleVision の既存の機能を使用して機能を向上することができます。

StyleVision のプログラムすることのできるオブジェクトは、COM API である **StyleVision API**, を使用して自動化されたクライアントで使用することができます。ドキュメント使用することのできる全てのオブジェクトに関する情報はこのドキュメントで紹介されています(次を参照: アプリケーション API)。

(C) 2015-2021 Altova GmbH

17.1 Scripting Editor

Scripting Editor is a development environment built into StyleVision from where you can customize the functionality of the Authentic View with the help of JScript or VBScript scripts. For example, you can add a new toolbar button that performs a custom task such as text formatting, or you can have StyleVision display a notification message box each time when an action is taken in the Authentic View. To make this possible, you create scripting projects—files with .asprj extension (Altova Scripting Project).



Scripting Editor

Scripting projects are portable; you can create them as part of the .sps (StyleVision Power Stylesheet) file. An Altova application that consumes the .sps file will be able to execute the macros inside the project—this can be not only StyleVision, but also XMLSpy or Authentic Desktop.

> Scripting Editor requires .NET Framework 2.0 or later to be installed before StyleVision is installed.

17.1.1 Creating a Scripting Project

All scripts and scripting information created in the Scripting Editor are stored in Altova Scripting Projects (.asprj files). A scripting project may contain macros, application event handlers, and forms (which can have their own event handlers). In addition, you can add global variables and functions to a "Global Declarations" script—this makes such variables and functions accessible across the entire project. To start a new project:

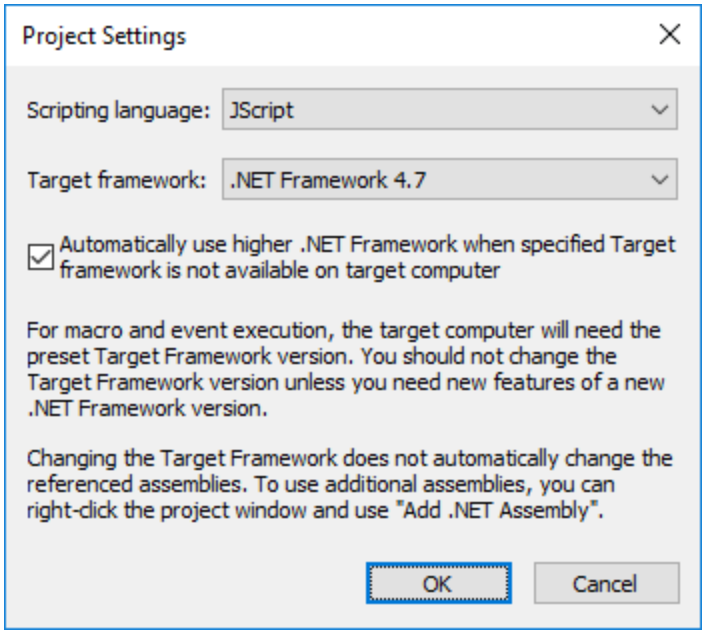
1. Create or open the .sps file that the scripting project should be part of.
2. Do one of the following:
   - On the **Authentic** menu, click **Authentic Scripts**.

- Click the **Design** tab at the base of the main window, and then select **Authentic Script**.

Each .sps file can have a single scripting project assigned to it. To open an existing scripting project associated with an .sps file, follow the same instructions as above. This opens the Scripting Editor window. As you edit the scripting project, you can temporarily save changes in memory, by clicking the **Save** button at the base of Scripting Editor . To save the scripting project to disk, first click **Save & Close** in Scripting Editor, and then save the .sps file.

The languages supported for use in a scripting project are JScript and VBScript (not to be confused with Visual Basic, which is not supported). These scripting engines are available by default on Windows and have no special requirements to run. You can select a scripting language as follows:

1. Right-click the **Project** item in the upper-left pane, and select **Project settings** from the context menu.
2. Select a language (JScript or VBScript), and click **OK**.



From the Project settings dialog box above, you can also change the target .NET Framework version. This is typically necessary if your scripting project requires features available in a newer .NET Framework version. Note that any clients using your scripting project will need to have the same .NET Framework version installed (or a later compatible version).

By default, a scripting project references several .NET assemblies, like `System`, `System.Data`, `System.Windows.Forms`, and others. If necessary, you can import additional .NET assemblies, including assemblies from .NET Global Assembly Cache (GAC) or custom .dll files. You can import assemblies as follows:

1. Statically, by adding them manually to the project. Right-click **Project** in the top-left pane, and select **Add .NET Assembly** from the context menu.
2. Dynamically, at runtime, by calling the `CLR.LoadAssembly` command from the code.

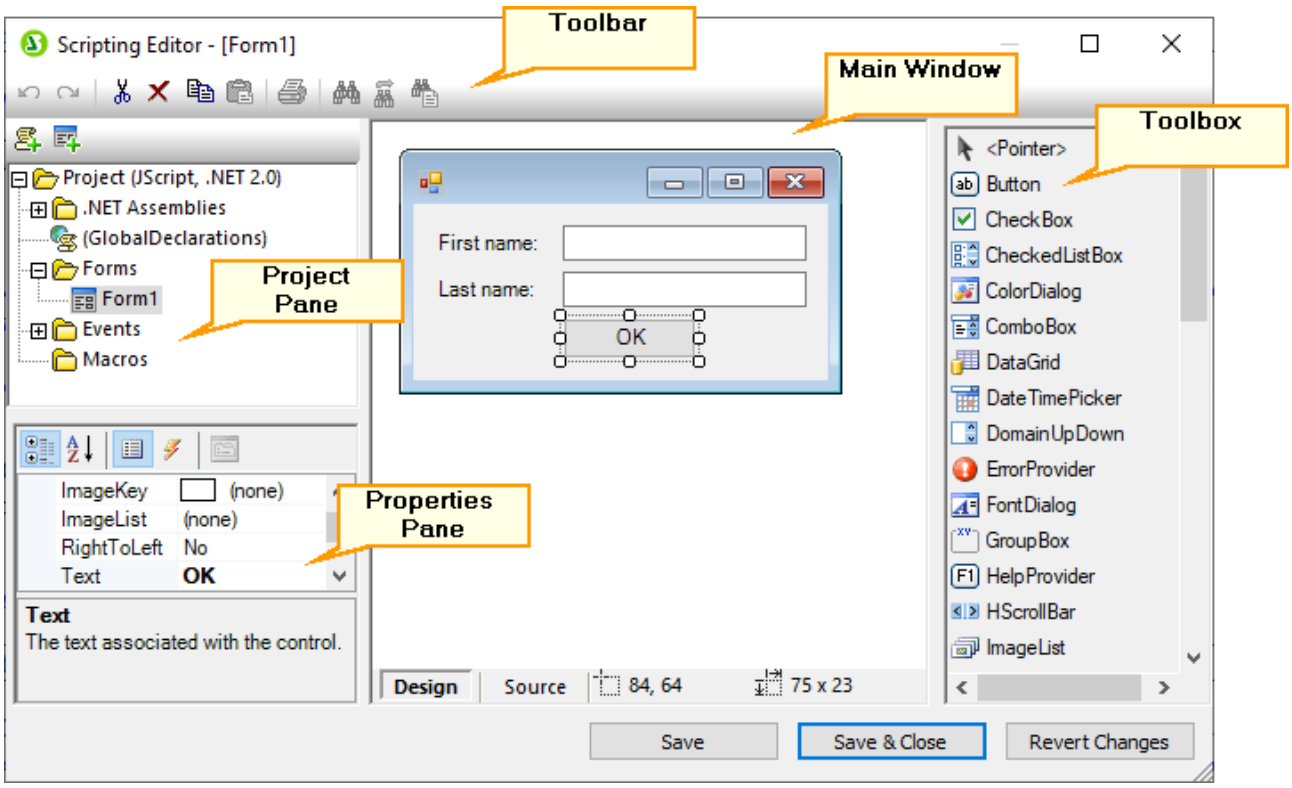
The next sections focus on the parts that your scripting project may need: global declarations, macros, forms, and events. For further information about scripting and examples of scripting projects, see Authentic Scripting.

(C) 2015-2021 Altova GmbH

17.1.1.1 Overview of the Environment

The Scripting Editor consists of the following parts:

- Toolbar
- Project pane
- Properties pane
- Main window
- Toolbox



The scripting project is saved when you save the .sps file of which the scripting project is part of. The **Save** button illustrated above creates only a temporary in-memory copy.

Toolbar

The toolbar includes the **Undo (Ctrl+Z)** and **Redo (Ctrl+Y)** commands and editor commands (**Copy**, **Cut**, **Delete**, **Paste**). When editing source code, the **Find** and **Replace** commands are additionally available, as well as the **Print** command.

Project pane

The project pane helps you view and manage the structure of the project. A scripting project consists of several components that can work together and may be created in any order:

- A *"Global Declarations"* script. As the name suggests, this script stores information available globally across the project. You can declare in this script any variables or functions that you need to be available in all forms, event handler scripts, and macros.
- *Forms.* Forms are typically necessary to collect user input, or provide some informative dialog boxes. A form is invoked by a call to it either within a function (in the Global Declarations script) or directly in a macro.
- *Events.* The "Events" folder displays StyleVision application events provided by the COM API. To write a script that will be executed when an event occurs, double-click any event, and then type the handling code in the editor. The application events should not be confused with form events; the latter are handled at form level, as further detailed below.
- *Macros*. A macro is a script that can be invoked either on demand from a context menu or be executed automatically when StyleVision starts. Macros do not have parameters or return values. A macro can access all variables and functions declared in the Global Declarations script and it can also display forms.

Right-click any of the components to see the available context menu commands and their shortcuts. Double-click any file (such as a form or a script) to open it in the main window.

The toolbar buttons provide the following quick commands:

New macro Adds a new macro to the project, in the **Macros** directory.

New form Adds a new form to the project, in the **Forms** directory.

Properties pane

The Properties pane is very similar to the one in Visual Studio. It displays the following:

- Form properties, when a form is selected
- Object properties, when an object in a form is selected
- Form events, when a form is selected
- Object events, when an object in a form is selected

To switch between the properties and events of the selected component, click the **Properties** or **Events** buttons, respectively.

The **Categorized** and **Alphabetical** icons display the properties or events either organized by category or organized in ascending alphabetical order.

When a property or event is selected, a short description of it is displayed at the bottom of the Properties pane.

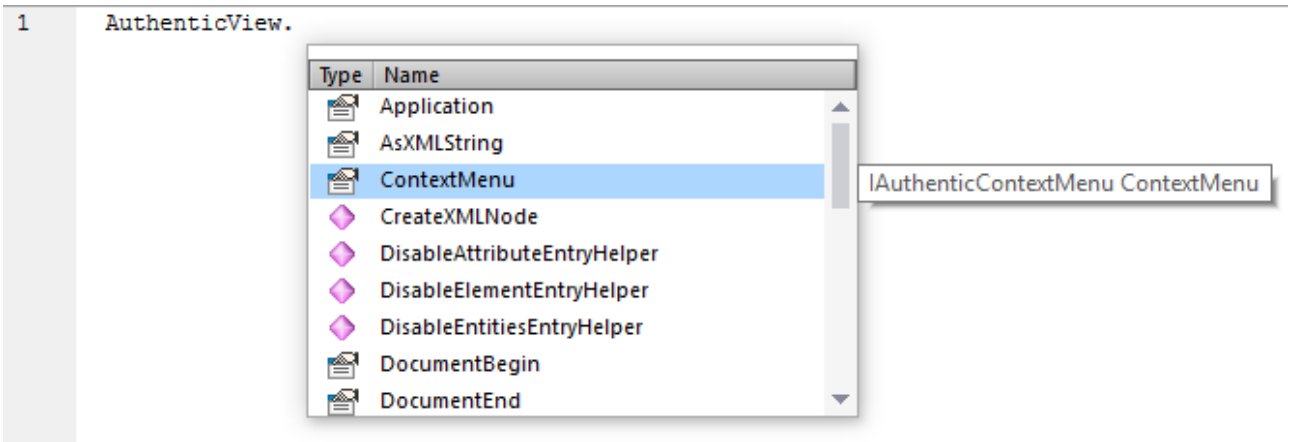
Main window

The main window is the working area where you can enter source code or modify the design of the form. When editing forms, you can work in two tabs: the **Design** tab and the **Source** tab. The **Design** tab shows the layout of the form, while the **Source** tab contains the source code such as handler methods for the form events.

The source code editor provides code editing aids such as syntax coloring, source code folding, highlighting of starting and ending braces, zooming, autocompletion suggestions, bookmarks.

(C) 2015-2021 Altova GmbH

Autocompletion suggestions
JScript and VBScript are untyped languages, so autocompletion is limited to COM API names and StyleVision built-in commands. The full method or property signature is shown next to the autocompletion entry helper.



Placing the mouse over a known method or property displays its signature (and documentation if available), for example:
The auto-completion entry helper is normally shown automatically during editing, but it can also be obtained on demand by pressing **Ctrl+Space**.

Bookmarks
- To set or remove a bookmark, click inside a line, and then press **Ctrl+F2**
- To navigate to the next bookmark, press **F2**
- To navigate to the previous bookmark, press **Shift+F2**
- To delete all bookmarks, press **Ctrl+Shift+F2**

Zooming in/out
- To zoom in or out, hold the **Ctrl** key pressed and then press the "+" or "-" keys or rotate the mouse wheel.

Text view settings
To trigger text settings, right-click inside the editor, and select **Text View Settings** from the context menu.

Font settings
To change the font, right-click inside the editor, and select **Text View Font** from the context menu.

Toolbox

The Toolbox contains all the objects that are available for designing forms, such as buttons, text boxes, combo boxes, and so on.

To add a Toolbox item to a form:

1. Create or open a form and make sure that the **Design** tab is selected.
2. Click the Toolbox object (for example, **Button**), and then click at the location in the form where you wish to insert it. Alternatively, drag the object directly onto the form.

Some objects such as `Timer` are not added to the Form but are created in a tray at the bottom of the main window. You can select the object in the tray and set properties and event handlers for the object from the Properties pane. For an example of handling tray components from the code, see Handling form events.

You can also add registered ActiveX controls to the form. To do this, right-click the Toolbox area and select **Add ActiveX Control** from the context menu.

17.1.1.2 Global Declarations

The "Global Declarations" script is present by default in any scripting project; you do not need to create it explicitly. Any variables or functions that you add to this script are considered global across the entire project. Consequently, you can refer to such variables and functions from any of the project's macros and events. The following is an example of a global declarations script that imports the `System.Windows.Forms` namespace into the project. To achieve that, the code below invokes the `CLR.Import` command built into Scripting Editor.

```
// import System.Windows.Forms namespace for all macros, forms and events:
CLR.Import( "System.Windows.Forms" );
```

Note: Every time a macro is executed or an event handler is called, the global declarations are re-initialized.

17.1.1.3 Macros

Macros are scripts that contain JScript (or VBScript, depending on your project's language) statements, such as variable declarations and functions.

If your projects should use macros, you can add them as follows: right-click inside the Project pane, select **Add Macro** from the context menu, and then enter the macro's code in the main form. The code of a macro could be as simple as an alert, for example:

```
alert("Hello, I'm a macro!");
```

More advanced macros can contain variables and local functions. Macros can also contain code that invokes forms from the project. The listing below illustrates an example of a macro that shows a form. It is assumed that this form has already been created in the "Forms" folder and has the name "SampleForm", see also Forms.

```
// display a form
ShowForm( "SampleForm" );
```

In the code listing above, `ShowForm` is a command built into Scripting Editor. For reference to other similar commands that you can use to work with forms and .NET objects, the Built-in Commands.

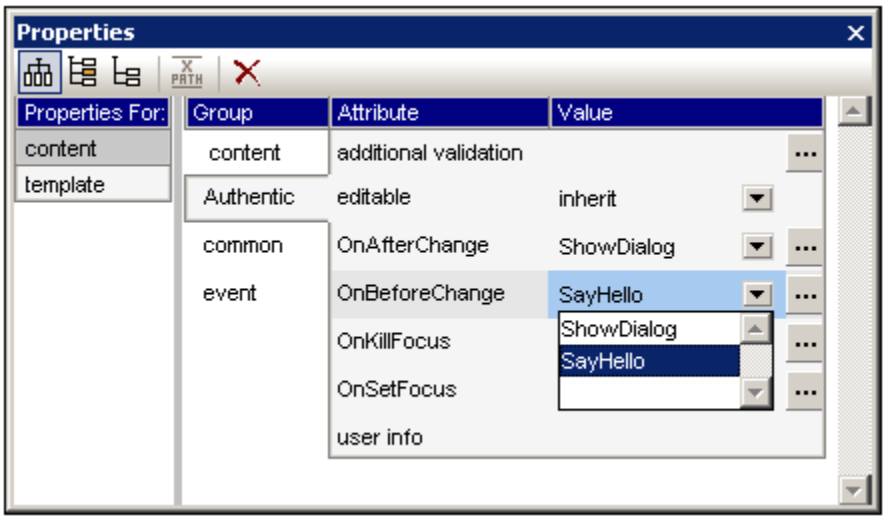
Only one macro can be run at a time. After a macro (or event) is executed, the script is closed and global variables lose their values.

To run a macro directly in Script Editor, click **Run Macro** . To debug a macro using the Visual Studio debugger, click **Debug Macro** .

Running macros

To run a macro, you need to first bind the macro to an Authentic event. The macro will run when that Authentic event is triggered. For example, say you wish to run a macro before editing text in Authentic. You must go about this as follows:

1. In Design View, select the editable content for which the macro is to be set.
2. In the Properties sidebar (*screenshot below*), select the *Authentic* group of properties of the relevant design component.



3. Select the Authentic event that will trigger the macro and open the dropdown list of this event's *Value* combo box. All the macros defined in the Scripting Project of the SPS will be listed. Select the macro you wish to associate with this function.
4. In Authentic View, when any Authentic event that has a macro assigned to it is triggered, then the macro will be executed.

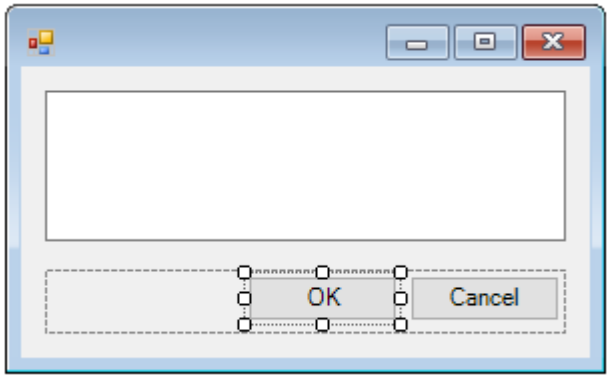
See the section, Authentic Scripting >> Macros, for a detailed description of how to use macros in a StyleVision design document.

17.1.1.4 Forms

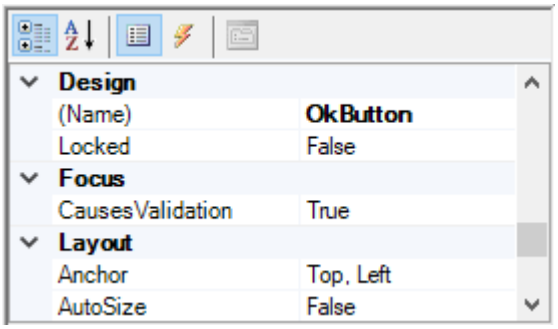
Forms are particularly useful if you need to collect input data from users or display data to users. A form can contain miscellaneous controls to facilitate this, such as buttons, check boxes, combo boxes, and so on.

To add a form, right-click inside the Project pane, and then select **Add Form** from the context menu. To add a control to a form, drag it from the Toolbox available to the right side of Scripting Editor and drop it onto the form.

You can change the position and size of the controls directly on the form, by using the handles that appear when you click any control, for example:



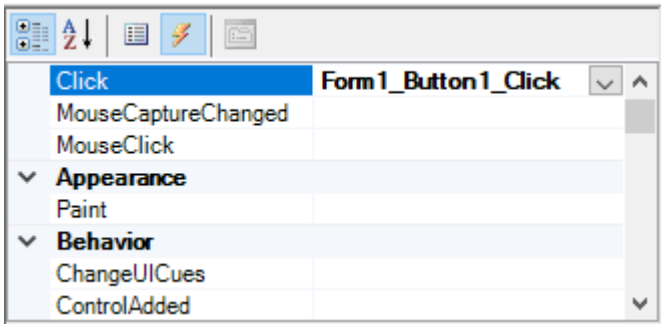
All form controls have properties that you can easily adjust in the Properties pane. To do this, first select the control on the form, and then edit the required properties in the Properties pane.



Handling form events

Each form control also exposes various events to which your scripting project can bind. For example, you might want to invoke some StyleVision COM API method whenever a button is clicked. To create a function that binds to a form event, do the following:

1. In the Properties pane, click **Events** .
2. In the **Action** column, double-click the event where you need the method (for example, in the image below, the handled event is "Click").



You can also add handler methods by double-clicking a control on the form. For example, double-clicking a button in the form design generates a handler method for the "Click" event of that button.

Once the body of the handler method is generated, you can type code that handles this event, for example:

```
//Occurs when the component is clicked.
function MyForm_ButtonClick( objSender, e_EventArgs )
{
   alert("A button was clicked");
}
```

To display a work-in-progress form detached from the Scripting Editor, right-click the form, and select **Test Form** from the context menu. Note that the **Test Form** command just displays the form; the form's events (such as button clicks) are still disabled. To have the form react to events, call it from a macro, for example:

```
// Instantiate and display a form
ShowForm( "SampleForm" );
```

Accessing form controls

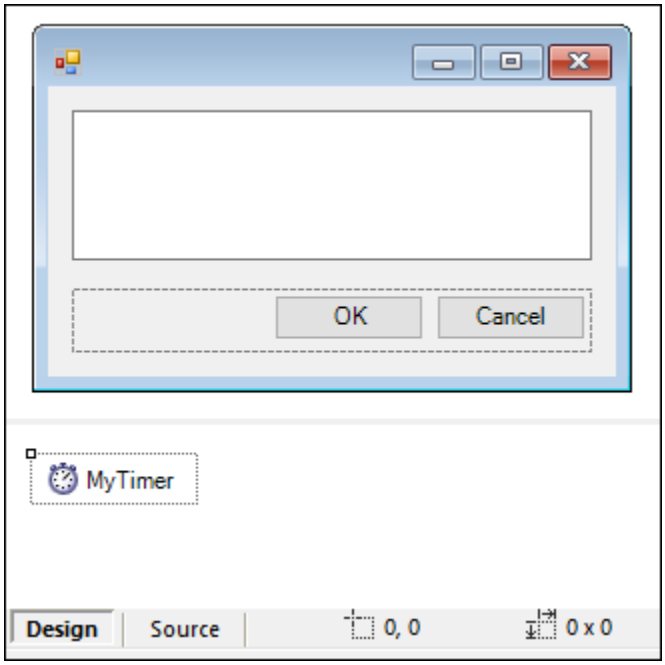
You can access any components on a form from your code by using field access syntax. For example, suppose there is a form designed as follows:

```
// MyForm
//   ButtonPanel
//      OkButton
//      CancelButton
//   TextEditor
//      AxMediaPlayer1
// TrayComponents
//   MyTimer
```

The code below shows how to instantiate the form, access some of its controls using field access syntax, and then display the form:

```
// Instantiate the form
var objForm = CreateForm("MyForm");
// Disable the OK button
objForm.ButtonPanel.OkButton.Enabled = false;
// Change the text of TextEditor
objForm.TextEditor.Text = "Hello";
// Show the form
objForm.ShowDialog();
```

When you add certain controls such as timers to the form, they are not displayed on the form; instead, they are shown as tray components at the base of the form design, for example:



To access controls from the tray, use the `GetTrayComponent` method on the form object, and supply the name of the control as argument. In this example, to get a reference to `MyTimer` and enable it, use the following code:

```
var objTimer = objForm.GetTrayComponent("MyTimer");
objTimer.Enabled = true;
```

For ActiveX Controls, you can access the underlying COM object via the OCX property:

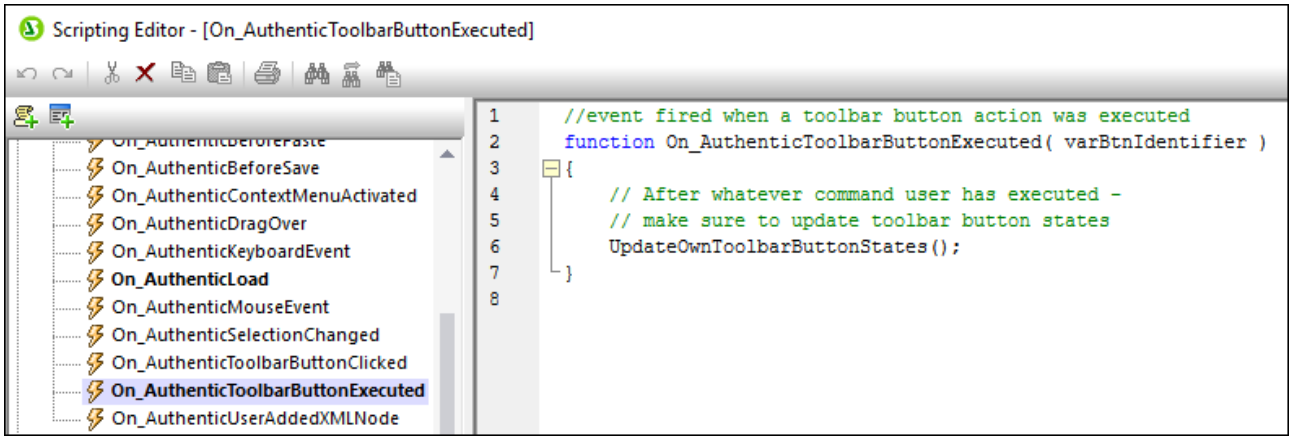
```
var ocx = lastform.AxMediaPlayer1.OCX; // get underlying COM object
ocx.enableContextMenu = true;
ocx.URL = "mms://apasf.apa.at/fm4_live_worldwide";
```

17.1.1.5 Events

Your scripting project may optionally include scripts that handle events in the Authentic View, such as cutting, copying, or pasting text, or opening a context menu. These events are provided by the StyleVision COM API, and you can find them in the "Events" folder of your scripting project. Note that these events are StyleVision-specific, as opposed to form events.

To create an event handler script, right-click an event, and select **Open** from the context menu (or double-click the event). The event handler script is displayed in the main window, where you can start editing it. For example, the event handler illustrated below calls a function when a toolbar button is executed. The function definition is in the Global Declarations script, so it is available to the event handler script.

(C) 2015-2021 Altova GmbH



The name of the event handler function must not be changed; otherwise, the event handler script will not be called.

You can optionally define local variables and helper functions within event handler scripts, for example:

```
var local;

function On_AuthenticLoad(  )
{
   local = "OnAuthenticLoad";
   Helper();
}

function Helper()
{
   alert("I'm a helper function for " + local);
}
```

17.1.1.6 JScript Programming Tips

Below are a few JScript programming tips that you may find useful while developing a scripting project in StyleVision Scripting Editor.

Out parameters

Out parameters from methods of the.NET Framework require special variables in JScript. For example:

```
var dictionary =
CLR.Create("System.Collections.Generic.Dictionary<System.String,System.String>");
dictionary.Add("1", "A");
dictionary.Add("2", "B");

// use JScript method to access out-parameters
var strOut = new Array(1);
```

```
if ( dictionary.TryGetValue("1", strOut) ) // TryGetValue will set the out parameter
  alert( strOut[0] ); // use out parameter
```

Integer arguments

.NET Methods that require integer arguments should not be called directly with JScript number objects which are floating point values. For example, instead of:

```
var objCustomColor = CLR.Static("System.Drawing.Color").FromArgb(128,128,128);
```

use:

```
var objCustomColor =
CLR.Static("System.Drawing.Color").FromArgb(Math.floor(128),Math.floor(128),Math.floor(
128));
```

Iterating .NET collections

To iterate .NET collections, the JScript Enumerator as well as the .NET iterator technologies can be used, for example:

```
// iterate using the JScript iterator
var itr = new Enumerator( coll );
for ( ; !itr.atEnd(); itr.moveNext() )
  alert( itr.item() );

// iterate using the .NET iterator
var itrNET = coll.GetEnumerator();
while( itrNET.MoveNext() )
  alert( itrNET.Current );
```

.NET templates

.NET templates can be instantiated as shown below:

```
var coll = CLR.Create( "System.Collections.Generic.List<System.String>" );
```

or

```
CLR.Import( "System" );
CLR.Import( "System.Collections.Generic" );
var dictionary = CLR.Create( "Dictionary<String,Dictionary<String,String>>" );
```

.NET enumeration values

.NET enumeration values are accessed as shown below:

```
var enumValStretch = CLR.Static( "System.Windows.Forms.ImageLayout" ).Stretch;
```


(C) 2015-2021 Altova GmbH

Enumeration literals

The enumeration literals from the StyleVision API can be accessed as shown below (there is no need to know their numerical value).

```
objExportXMIFileDlg.XMIType = eXMI21ForUML23;
```

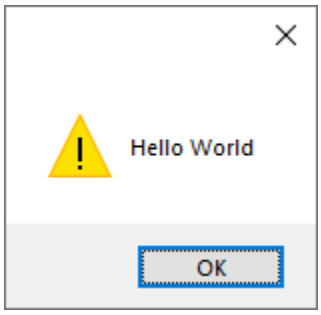
17.1.2 Built-in Commands

This section provides reference to all the commands you can use in the StyleVision Scripting Editor.

- alert
- confirm
- CLR.Create
- CLR.Import
- CLR.LoadAssembly
- CLR.ShowImports
- CLR.ShowLoadedAssemblies
- CLR.Static
- CreateForm
- doevents
- lastform
- prompt
- ShowForm
- watchdog

17.1.2.1 alert

Displays a message box that shows a given message and the "OK" button. To proceed, the user will have to click "OK".



Signature

For JScript, the signature is:

```
alert(strMessage : String) -> void
```

For VBScript, the signature is:

```
MsgBox(strMessage : String) -> void
```

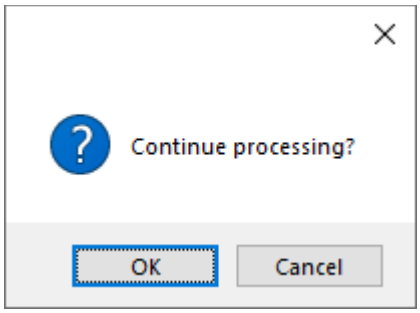
Example

The following JScript code displays a message box with the text "Hello World".

```
alert("Hello World");
```

17.1.2.2 confirm

Opens a dialog box that shows a given message, a confirmation button, and a cancel button. The user will have to click either "OK" or "Cancel" to proceed. Returns a Boolean that represents the user's answer. If the user clicked "OK", the function returns **true**; if the user clicked "Cancel", the function returns **false**.



Signature

```
confirm(strMessage : String) -> result : Boolean
```

Example (JScript)

```
if ( confirm( "Continue processing?" ) == false )
  alert("You have cancelled this action");
```

Example (VBScript)

```
If ( confirm( "Continue processing?" ) = false ) Then
  MsgBox("You have cancelled this action")
End If
```

17.1.2.3 CLR.Create

Creates a new .NET object instance of the type name supplied as argument. If more than one argument is passed, the successive arguments are interpreted as the arguments for the constructor of the .NET object. The return value is a reference to the created .NET object

Signature

```
CLR.Create(strTypeNameCLR : String, constructor arguments ... ) -> object
```

Example

The following JScript code illustrates how to create instances of various .NET classes.

```
// Create an ArrayList
var objArray = CLR.Create("System.Collections.ArrayList");
// Create a ListViewItem
var newItem = CLR.Create( "System.Windows.Forms.ListViewItem", "NewItemText" );
// Create a List<string>
var coll = CLR.Create( "System.Collections.Generic.List<System.String>" );
// Import required namespaces and create a Dictionary object
CLR.Import( "System" );
CLR.Import( "System.Collections.Generic" );
var dictionary = CLR.Create( "Dictionary< String, Dictionary< String, String > >" );
```

17.1.2.4 CLR.Import

Imports a namespace. This is the scripting equivalent of C# `using` and VB.Net `imports` keyword. Calling `CLR.Import` makes it possible to leave out the namespace part in subsequent calls like `CLR.Create()` and `CLR.Static()`.

Note: Importing a namespace does not add or load the corresponding assembly to the scripting project. You can add assemblies to the scripting project dynamically (at runtime) in the source code by calling CLR.LoadAssembly.

Signature

```
CLR.Import(strNamespaceCLR : String) -> void
```

Example

Instead of having to use fully qualified namespaces like:

```
if ( ShowForm( "FormName" ) == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
{
    var sName = lastform.textboxFirstName.Text + " " + lastform.textboxLastName.Text;
    CLR.Static( "System.Windows.Forms.MessageBox" ).Show( "Hello " + sName );
}
```

One can import namespaces first and subsequently use the short form:

```
CLR.Import( "System.Windows.Forms" );

if ( ShowForm( "FormName" ) == CLR.Static( "DialogResult" ).OK )
{
    var sName = lastform.textboxFirstName.Text + " " + lastform.textboxLastName.Text;
    CLR.Static( "MessageBox" ).Show( "Hello " + sName );
}
```

17.1.2.5 CLR.LoadAssembly

Loads the .NET assembly with the given long assembly name or file path. Returns Boolean **true** if the assembly could be loaded; **false** otherwise.

Signature

```
CLR.LoadAssembly(strAssemblyNameCLR : String, showLoadErrors : Boolean) -> result :
Boolean
```

Example

The following JScript code attempts to set the clipboard text by loading the required assembly dynamically.

```
// set clipboard text (if possible)
// System.Windows.Clipboard is part of the PresentationCore assembly, so load this
assembly first:
if ( CLR.LoadAssembly( "PresentationCore, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35", true ) )
{
    var clipboard = CLR.Static( "System.Windows.Clipboard" );
    if ( clipboard != null )
        clipboard.SetText( "HelloClipboard" );
}
```

17.1.2.6 CLR.ShowImports

Opens a message box that shows the currently imported namespaces. The user will have to click "OK" to proceed.

Signature

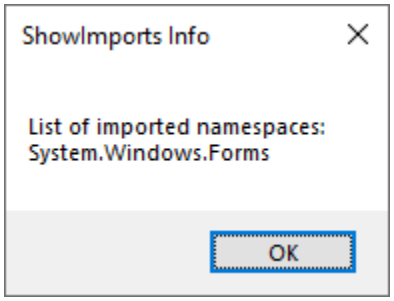
```
CLR.ShowImports() -> void
```

Example

The following JScript code first imports a namespace, and then displays the list of imported namespaces:

(C) 2015-2021 Altova GmbH

```
CLR.Import( "System.Windows.Forms");
CLR.ShowImports();
```



17.1.2.7 CLR.ShowLoadedAssemblies

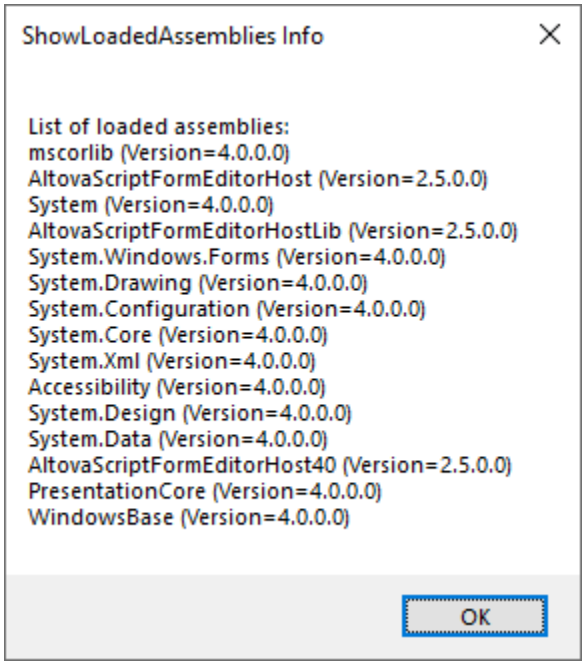
Opens a message box that shows the currently loaded assemblies. The user will have to click "OK" to proceed.

Signature

```
CLR.ShowLoadedAssemblies() -> void
```

Example

```
CLR.ShowLoadedAssemblies();
```



17.1.2.8 CLR.Static

Returns a reference to a static .NET object. You can use this function to get access to .NET types that have no instances and contain only static members.

Signature

```
CLR.Static(strTypeNameCLR : String) -> object
```

Example (JScript)

```
// Get the value of a .NET Enum into a variable
var enumValStretch = CLR.Static( "System.Windows.Forms.ImageLayout" ).Stretch

// Set the value of the Windows clipboard
var clipboard = CLR.Static( "System.Windows.Clipboard" );
clipboard.SetText( "HelloClipboard" );

// Check the buttons pressed by the user on a dialog box
if ( ShowForm( "FormName" ) == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
    alert( "ok" );
else
    alert( "cancel" );
```

17.1.2.9 CreateForm

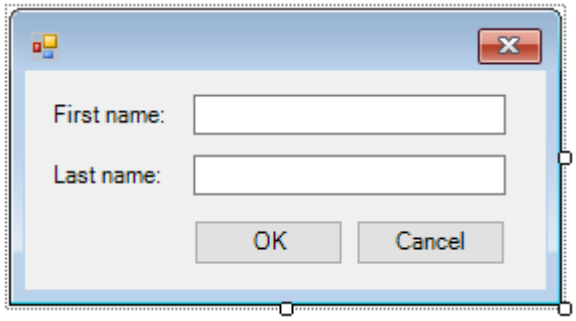
Instantiates the `Form` object identified by the name supplied as argument. The form must exist in the "Forms" folder of the scripting project. Returns the form object (`System.Windows.Forms.Form`) corresponding to the given name, or `null` if no form with such name exists.

Signature

```
CreateForm (strFormName : String) -> System.Windows.Forms.Form | null
```

Example

Let's assume that a form called "FormName" exists in the scripting project.



The following JScript code instantiates the form with some default values and displays it to the user.

```
var myForm = CreateForm( "FormName" );
if ( myForm != null )
{
   myForm.textboxFirstName.Text = "Daniela";
   myForm.textboxLastName.Text = "Heidegger";
   var dialogResult = myForm.ShowDialog();
}
```

The `dialogResult` can subsequently be evaluated as follows:

```
if ( dialogResult == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
    alert( "ok" );
else
    alert( "cancel" );
```

Note: The code above will work only if the **DialogResult** property of the "OK" and "Cancel" buttons is set correctly from the Properties pane (for example, it must be **OK** for the "OK" button).

17.1.2.10 doevents

Processes all Windows messages currently in the message queue.

Signature

```
doevents() -> void
```

Example (JScript)

```
for ( i=0; i < nLongLastingProcess; ++i )
{
    // do long lasting process

    doevents(); // process Windows messages; give UI a chance to update
}
```

17.1.2.11 lastform

This is a global field that returns a reference to the last form object that was created via `CreateForm()` or `ShowForm()`.

Signature

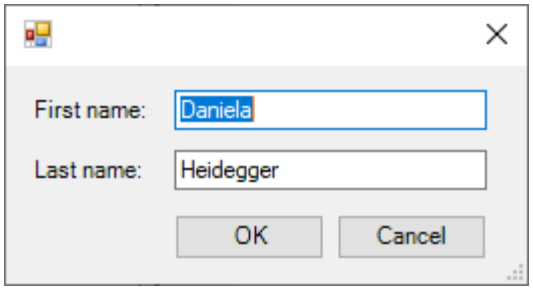
```
lastform -> formObj : System.Windows.Forms.Form
```

Example

The following JScript code shows the form "FormName" as a dialog box.

```
CreateForm( "FormName" );
if ( lastform != null )
{
    lastform.textboxFirstName.Text = "Daniela";
    lastform.textboxLastName.Text = "Heidegger";
    var dialogResult = lastform.ShowDialog();
}
```

The values of both textbox controls are initialized with the help of `lastform`.



17.1.2.12 prompt

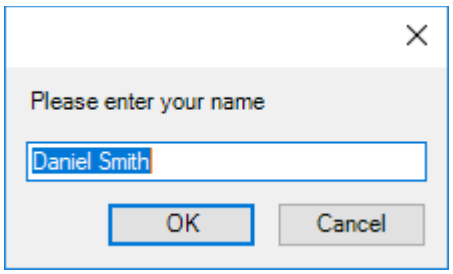
Opens a dialog box that shows a message and a textbox control with a default answer. This can be used to let the user input a simple string value. The return value is a string that contains the textbox value or null if the user selected "Cancel".

Signature

```
prompt(strMessage : String, strDefault : String) -> val : String
```

Example

```
var name = prompt( "Please enter your name", "Daniel Smith" );
if ( name != null )
    alert( "Hello " + name + "!" );
```



(C) 2015-2021 Altova GmbH

17.1.2.13 ShowForm

Instantiates a new form object from the given form name and immediately shows it as dialog box. The return value is an integer that represents the generated `DialogResult` (`System.Windows.Forms.DialogResult`). For the list of possible values, refer to the documentation of the `DialogResult` Enum (https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.dialogresult?view=netframework-4.8).

Signature

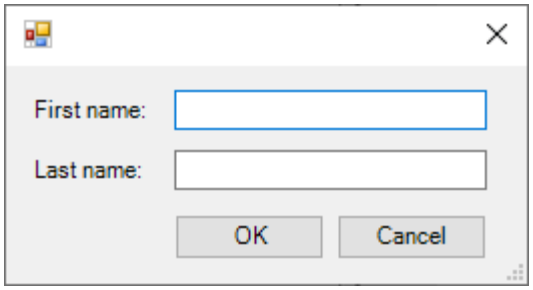
```
ShowForm(strFormName : String) -> result : Integer
```

Example

The following JScript code

```
var dialogResult = ShowForm( "FormName" );
```

Shows the form "FormName" as a dialog box:



The `DialogResult` can subsequently be evaluated, for example:

```
if ( dialogResult == CLR.Static( "System.Windows.Forms.DialogResult" ).OK )
    alert( "ok" );
else
    alert( "cancel" );
```

Note: The code above will work only if the **DialogResult** property of the "OK" and "Cancel" buttons is set correctly from the Properties pane (for example, it must be **OK** for the "OK" button).

17.1.2.14 watchdog

Long running CPU-intensive scripts may ask the user if the script should be terminated. The `watchdog()` method is used to disable or enable this behavior. By default, the watchdog is enabled.

Calling `watchdog(true)` can also be used to reset the watchdog. This can be useful before executing long running CPU-intensive tasks to ensure they have the maximum allowed script processing quota.

Signature

```
watchdog(bEnable : boolean) -> void
```

Example

```
watchdog( false );  // disable watchdog - we know the next statement is CPU intensive
but it will terminate for sure
doCPUIntensiveScript();
watchdog( true );   // re-enable watchdog
```


(C) 2015-2021 Altova GmbH

17.2 Application API

The COM-based API of StyleVision (also called the Application API from now on) enables other applications to use the functionality of StyleVision.

StyleVision and its Application API follow the common specifications for automation servers set out by Microsoft. It is possible to access the methods and properties of the Application API from common development environments, such as those using C#, C++, VisualBasic, and Delphi, and with scripting languages like JScript and VBScript.

Execution environments for the Application API

The Application API can be accessed from the following execution environments:

- External programs (described below and in the Overview part of this section)
- From within the built-in Scripting Editor of StyleVision. For a description of the scripting environment, see the section, Scripting Editor.
- Via an ActiveX Control, which is available if the integration package is installed. For more information, see the section ActiveX Integration.

External programs

In the Overview part of this section, we describe how the functionality of StyleVision can be accessed and automated from external programs.

Using the Application API from outside StyleVision requires an instance of StyleVision to be started first. How this is done depends on the programming language used. See the section, Programming Languages, for information about individual languages.

Essentially, StyleVision will be started via its COM registration. Then the `Application` object associated with the StyleVision instance is returned. Depending on the COM settings, an object associated with an already running StyleVision can be returned. Any programming language that supports creation and invocation of COM objects can be used. The most common of these are listed below.

- JScript script files have a simple syntax and are designed to access COM objects. They can be run directly from a DOS command line or with a double click on Windows Explorer. They are best used for simple automation tasks.
- C# is a full-fledged programming language that has a wide range of existing functionality. Access to COM objects can be automatically wrapped using C#..
- C++ provides direct control over COM access but requires relatively larger amounts of code than the other languages.
- Java: Altova products come with native Java classes that wrap the Application API and provide a full Java look-and-feel.
- Other programming languages that make useful alternatives are: Visual Basic for Applications, Perl, and Python.

Programming points

The following limitations must be considered in your client code:

- Be aware that if your client code crashes, instances of StyleVision may still remain in the system.

- Don't hold references to objects in memory longer than you need them, especially those from the `XMLData` interface. If the user interacts between two calls of your client, then there is no guarantee that these references are still valid.
- Don't forget to disable dialogs if the user interface is not visible.

17.2.1 Overview

This overview of the Application API is organized as follows:

- The Object Model describes the relationships between the objects of the Application API.
- Programming Languages explains how the most commonly used programming languages (JScript, VBScript, C#, and Java) can be used to access the functionality of the Application API. Code listings from the example files supplied with your application package are used to describe basic mechanisms.

17.2.1.1 Object Model

The starting point for every application which uses the Application API is the `Application` object. This object contains general methods like import/export support and references to the open documents and any open project.

The `Application` object is created differently in various programming languages. In scripting languages such as JScript or VBScript, this involves calling a function which initializes the application's COM object. For examples, see the Programming Languages section.

The chart below shows the links between the main objects of the Application API

```
Application
|
| -- Documents
|     |
|     | -- Document
|     |     |
|     |     | -- Methods for assigning working XML file, saving files
|     |     | -- Methods for saving generated files
|     |     | -- Methods for activating, saving, closing SPS file
```

In addition there are objects for schema sources and properties, as well as methods for AuthenticView events.

Once you have created an `Application` object you can start using the functionality of StyleVision.

17.2.1.2 Programming Languages

Programming languages differ in the way they support COM access. A few examples for the most frequently used languages (*links below*) will help you get started. The code listings in this section show how basic functionality can be accessed. The files in the `API` subfolder of the `Examples` folder can be used to test this functionality:

 (C) 2015-2021 Altova GmbH

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\StyleVision\2021\%APPNAME%>Examples`

JScript

The JScript listings demonstrate the following basic functionality:

- Start application or attach to a running instance
- Simple document access
- Iteration

VBScript

VBScript is different than JScript only syntactically; otherwise it works in the same way. For more information, refer to the JScript examples.

C#

C# can be used to access the Application API functionality. The code listings show how to access the API for certain basic functionality.

- Start StyleVision: Starts StyleVision, which is registered as an automation server, or activates StyleVision if it is already running.
- Open OrgChart.pxf: Locates one of the example documents installed with StyleVision and opens it. If this document is already open it becomes the active document.
- OnDocumentOpened Event On/Off: Shows how to listen to StyleVision events. When turned on, a message box will pop up after a document has been opened.
- Open ExpReport.xml: Opens another example document.
- Shutdown StyleVision: Stops StyleVision.

Java

The StyleVision API can be accessed from Java code. The Java sub-section of this section explains how some basic StyleVision functionality can be accessed from Java code. It is organized into the following sub-sections:

- Mapping Rules for the Java Wrapper
- Example Java Project
- Application Startup and Shutdown
- Simple Document Access
- Iterations
- Event Handlers

17.2.1.2.1 JScript

This section contains listings of JScript code that demonstrate the following basic functionality:

- Start application or attach to a running instance

- Simple document access
- Iteration

Example files

The code listings in this section are available in example files that you can test as is or modify to suit your needs. The JScript example files are located in the `JScript` subfolder of the API Examples folder:

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\StyleVision\2021\%APPNAME%>Examples`

The example files can be run in one of two ways:

- *From the command line:* Open a command prompt window, change the directory to the path above, and type the name of one of the example scripts (for example, `Start.js`).
- *From Windows Explorer:* In Windows Explorer, browse for the JScript file and double-click it.

The script is executed by Windows Script Host that is packaged with Windows operating system. For more information about Windows Script Host, refer to MSDN documentation (https://msdn.microsoft.com).

17.2.1.2.1.1 Start Application

The JScript below starts the application and shuts it down. If the COM object of the 32-bit StyleVision cannot be found, the code attempts to get the COM object of the 64-bit application; otherwise, an error is thrown. If an instance of the application is already running, the running instance will be returned.

メモ 32-ビット StyleVision では、登録された名前、またはCOM オブジェクトのプログラム識別子（ProgId）は以下の通りです: `MapForce.ApplicationStyleVision.Application`。64-ビット StyleVision では、名前は以下の通りです: MapForce_x64.ApplicationStyleVision_x64.Application。プログラムの呼び出しは自身のレジストリハイブ、またはグループ（32-bit または64-bit）内のCLASSES レジストリエントリにアクセスすることに注意してください。ですから、標準のコマンドプロンプトと上 64-bit Windows のWindows Explorerを使用してスクリプトを実行すると、64-ビット StyleVision. を指す 64-ビットレジストリエントリにアクセスされます。この理由のため、StyleVision 32-ビットと64-ビットの両方がインストールされている場合、32-ビット StyleVision. を呼び出すために特別な処理が必要になります。例えば、スクリプトホストがプログラムを呼び出す場合以下を行います:

1. ディレクトリを **C:\Windows\SysWOW64** に変更します。
2. コマンドライン上で **wscript.exe** を入力し実行するスクリプトパスを以下のように入力します:

```
wscript.exe "C:\Users\...
\Documents\Altova\StyleVision2021\StyleVisionExamples\API\JScript\start.js"
```

```
// Initialize application's COM object. This will start a new instance of the
application and
// return its main COM object. Depending on COM settings, the main COM object of an
already
// running application might be returned.
try {   objStyleVision = WScript.GetObject("", "StyleVision.Application");   }
catch(err) {}
```

```
if( typeof( objStyleVision ) == "undefined" )
{
   try   {   objStyleVision = WScript.GetObject("", "StyleVision_x64.Application")   }
   catch(err)
   {
      WScript.Echo( "Can't access or create StyleVision.Application" );
      WScript.Quit();
   }
}

// if newly started, the application will start without its UI visible. Set it to
visible.
objStyleVision.Visible = true;

WScript.Echo(objStyleVision.Edition + " has successfully started. ");

objStyleVision.Visible = false; // will shutdown application if it has no more COM
connections
//objStyleVision.Visible = true;   // will keep application running with UI visible
```

The JScript code listed above is available in the sample file `Start.js` (see Example Files).

17.2.1.2.1.2 Simple Document Access

After you have started the application as shown in Start Application, you will most likely want to programmatically open a document in order to work with it. The JScript code listing below illustrates how to open two documents from the StyleVision Examples folder.

```
// Locate examples via USERPROFILE shell variable. The path needs to be adapted to
major release versions.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objStyleVision.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\My
Documents\\Altova\\StyleVision" + majorVersionYear + "\\StyleVisionExamples\\";

// Tell StyleVision to open two documents. No dialogs
objDoc1 = objStyleVision.Documents.OpenDocument(strExampleFolder + "OrgChart.pxf");
objStyleVision.Documents.OpenDocument(strExampleFolder +
"BiggestCitiesPerContinent.sps");
```

The JScript code listed above is available in the sample file `DocumentAccess.js` (see Example Files).

17.2.1.2.1.3 Iteration

The JScript listing below shows how to iterate through the open documents. It is assumed that you have already started the application and opened some documents as shown in the previous sections.

```
// go through all open documents using a JScript Enumerator
for (var iterDocs = new Enumerator(objStyleVision.Documents); !iterDocs.atEnd();
iterDocs.moveNext())
{
   objName = iterDocs.item().Name;
   WScript.Echo("Document name: " + objName);
}

// go through all open documents using index-based access to the document collection
for (i = objStyleVision.Documents.Count; i > 0; i--)
   objStyleVision.Documents.Item(i).Close();
```

The JScript code listed above is available in the sample file `DocumentAccess.js` (see Example Files).

17.2.1.2.2 C#

The C# programming language can be used to access the Application API functionality. You could use Visual Studio 2010/2012/2013/2015/2017/2019 to create the C# code, saving it in a Visual Studio project. Create the project as follows:

1. In Microsoft Visual Studio, add a new project using **File | New | Project**.
2. Add a reference to the StyleVision Type Library by clicking **Project | Add Reference**. The Add Reference dialog appears. Browse for the StyleVision Type Library component, which is located in the StyleVision application folder, and add it.
3. Enter the code you want.
4. Compile the code and run it.

Example C# project

Your StyleVision package contains an example C# project, which is located in the `API\C#` subfolder of the `Examples` folder :

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\StyleVision\2021\%APPNAME%>Examples`

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019. The code listing below shows how basic application functionality can be used. This code is similar to the example C# project in the API Examples folder of your application package, but might differ slightly.

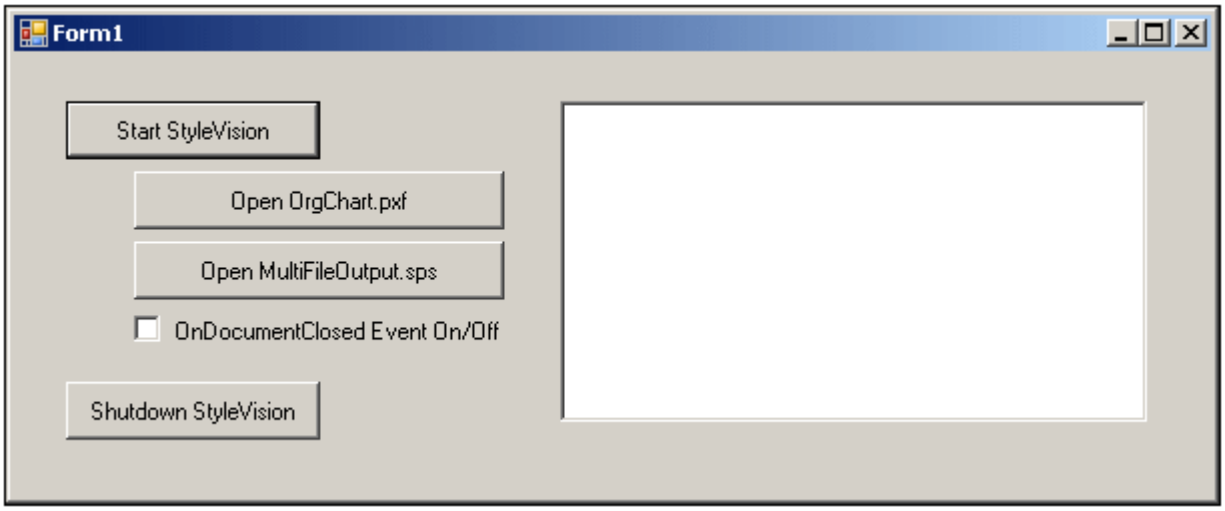
Platform configuration

If you have a 64-bit operating system and are using a 32-bit installation of StyleVision, you must add the x86 platform in the solution's Configuration Manager and build the sample using this configuration. A new x86 platform (for the active solution in Visual Studio) can be created in the New Solution Platform dialog (**Build | Configuration Manager | Active solution platform | <New…>**).

What the code listing below does

The example code listing below creates a simple user interface (*screenshot below*) with buttons that invoke basic StyleVision operations:

(C) 2015-2021 Altova GmbH



- Start StyleVision: Starts StyleVision, which is registered as an automation server, or activates the application if it is already running.
- Open OrgChart.pxf: Locates one of the example documents installed with StyleVision and opens it. If this document is already open it becomes the active document.
- Open MultiFileOutput.sps: Opens another example document.
- Shut down StyleVision: Stops StyleVision.

You can modify the code (of the code listing below or of the example C# project in the API Examples folder) in any way you like and run it.

Compiling and running the example

In the API Examples folder, double-click the file `AutomateStyleVision_VS2008.sln` or the file `AutomateStyleVision_VS2010.sln` (to open in Visual Studio 2010/2012/2013/2015/2017/2019). Alternatively the file can be opened from within Visual Studio (with **File | Open | Project/Solution**). To compile and run the example, select **Debug | Start Debugging** or **Debug | Start Without Debugging**.

Code listing of the example

Given below is the C# code listing of the basic functionality of the form (`Form1.cs`) created in the `AutomateStyleVision` example. Note that the code listed below might differ slightly from the code in the API Examples form.The listing below is commented for ease of understanding. Parts of the code are also presented separately in the sub-sections of this section, according to the Application API functionality they access.

The code essentially consists of a series of handlers for the buttons in the user interface shown in the screenshot above.

```
namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
```

```
        // An instance of StyleVision is accessed via its automation interface.
        StyleVisionLib.Application StyleVision;

        // Location of examples installed with StyleVision
        String strExamplesFolder;

        private void Form1_Load(object sender, EventArgs e)
        {
            // Locate examples installed with StyleVision.
            // REMARK: You might need to adapt this if you have a different major version
of the product.
            strExamplesFolder = Environment.GetEnvironmentVariable("USERPROFILE") + "\\My
Documents\\Altova\\StyleVision2012\\StyleVisionExamples\\";
        }

        // Handler for the "Start StyleVision" button
        private void StartStyleVision_Click(object sender, EventArgs e)
        {
            if (StyleVision == null)
            {
                Cursor.Current = Cursors.WaitCursor;

                // If there is no StyleVision instance, create one and make it visible.
                StyleVision = new StyleVisionLib.Application();
                StyleVision.Visible = true;

                Cursor.Current = Cursors.Default;
            }
            else
            {
                // If a StyleVision instance is already running, make sure it's visible.
                if (!StyleVision.Visible)
                    StyleVision.Visible = true;
            }
        }

        // Handler for the "Open OrgChart.pxf" button
        private void openOrgChart_Click(object sender, EventArgs e)
        {
            // Make sure there's a running StyleVision instance, and that it's visible
            StartStyleVision_Click(null, null);

            // Open a sample files installed with the product.
            StyleVision.Documents.OpenDocument(strExamplesFolder + "OrgChart.pxf");
            updateListBox();
        }

        // Handler for the "Open MultiFileOutput.sps" button
        private void openMultiFileOutput_Click(object sender, EventArgs e)
        {
            // Make sure there's a running StyleVision instance, and that it's visible
            StartStyleVision_Click(null, null);

            // Open one of the sample files installed with the product.
            StyleVision.Documents.OpenDocument(strExamplesFolder +
"MultiFileOutput.sps");
            updateListBox();
        }

        // Handler for the "Shutdown StyleVision" button
        // Shut down the application instance by explicitly releasing the COM object.
        private void shutdownStyleVision_Click(object sender, EventArgs e)
```

```
        {
            if (StyleVision != null)
            {
                // Allow shut-down of StyleVision by releasing UI
                StyleVision.Visible = false;

                // Explicitly release the COM object
                try
                {
                    int i =
System.Runtime.InteropServices.Marshal.ReleaseComObject(StyleVision);
                    while
(System.Runtime.InteropServices.Marshal.ReleaseComObject(StyleVision) > 0) ;
                }
                finally
                {
                    // Disallow subsequent access to this object.
                    StyleVision = null;
                }
            }
        }

        delegate void addListBoxItem_delegate(string sText);
        // Called from the UI thread
        private void addListBoxItem(string sText)
        {
            listBoxMessages.Items.Add(sText);
        }
        // Wrapper method to call UI control methods from a worker thread
        void syncWithUIthread(Control ctrl, addListBoxItem_delegate methodToInvoke,
String sText)
        {
            // Control.Invoke: Executes on the UI thread, but calling thread waits for
completion before continuing.
            // Control.BeginInvoke: Executes on the UI thread, and calling thread doesn't
wait for completion.
            if (ctrl.InvokeRequired)
                ctrl.BeginInvoke(methodToInvoke, new Object[] { sText });
        }

        // Event handler for OnDocumentClosed event
        private void handleOnDocumentClosed(StyleVisionLib.Document i_ipDocument)
        {
            String sText = "";

            if (i_ipDocument.Name.Length > 0)
                sText = "Document " + i_ipDocument.Name + " was closed!";

            // Synchronize the calling thread with the UI thread because
            // COM events are triggered from a working thread
            addListBoxItem_delegate methodToInvoke = new
addListBoxItem_delegate(addListBoxItem);
            // Call syncWithUIthread with the following arguments:
            // 1 - listBoxMessages - list box control to display messages from COM events
            // 2 - methodToInvoke  - a C# delegate which points to the method which will
be called from the UI thread
            // 3 - sText           - the text to be displayed in the list box
            syncWithUIthread(listBoxMessages, methodToInvoke, sText);
        }
```

```
        private void updateListBox()
        {
            // Iterate through all open documents
            listBoxMessages.Items.Clear();


            for (int i = 1; i <= StyleVision.Documents.Count; i++)
            {
                StyleVisionLib.Document doc = StyleVision.Documents[i];

                if (doc != null)
                {
                    if (checkBoxEventOnOff.Checked)
                        doc.OnDocumentClosed += new
StyleVisionLib._IDocumentEvents_OnDocumentClosedEventHandler(handleOnDocumentClosed);
                    else
                        doc.OnDocumentClosed -= new
StyleVisionLib._IDocumentEvents_OnDocumentClosedEventHandler(handleOnDocumentClosed);

                    listBoxMessages.Items.Add(doc.Name);
                    StyleVisionLib.SchemaSources sources = doc.SchemaSources;

                    for (int j = 1; j <= sources.Count; j++)
                    {
                        StyleVisionLib.SchemaSource source = sources[j];

                        if (source != null)
                        {
                            listBoxMessages.Items.Add("\tSchema      file name : " +
source.SchemaFileName + "\n");
                            listBoxMessages.Items.Add("\tWorking XML file name : " +
source.WorkingXMLFileName + "\n");
                            listBoxMessages.Items.Add("\tIs main schema source : " +
source.IsMainSchemaSource + "\tType name : " + source.TypeName + "\n");
                        }
                    }
                }
            }
        }
    }
}
```

17.2.1.2.2.1 Add Reference to StyleVision API

Add the application's type library as a reference in a .NET project as follows: With the .NET project open, click **Project | Add Reference**. Then browse for the type library, which is called `StyleVision.tlb`, and is located in the StyleVision application folder.

Then declare a variable to access the StyleVision API:

```
     // An instance of StyleVision is accessed via its automation interface.
        StyleVisionLib.Application StyleVision;
```

17.2.1.2.2.2 Application Startup and Shutdown

In the code snippets below, the methods `StartStyleVision_Click` and `ShutdownStyleVision_Click` are those assigned to buttons in the AutomateStyleVision example that, respectively, start up and shut down the application. This example is located in the C# subfolder of the API Examples folder (*see the file Form1.cs*):

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\StyleVision\2021\%APPNAME%>Examples`

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

Starting StyleVision

The following code snippet from the AutomateStyleVision example shows how to start up the application.

```
        // Handler for the "Start StyleVision" button
        private void StartStyleVision_Click(object sender, EventArgs e)
        {
            if (StyleVision == null)
            {
                Cursor.Current = Cursors.WaitCursor;

                // If there is no StyleVision instance, create one and make it visible.
                StyleVision = new StyleVisionLib.Application();
                StyleVision.Visible = true;

                Cursor.Current = Cursors.Default;
            }
            else
            {
                // If an instance of StyleVision is already running, make sure it's
visible
                if (!StyleVision.Visible)
                    StyleVision.Visible = true;
            }
        }
```

Shutting down StyleVision

The following code snippet from the AutomateStyleVision example shows how to shut down the application.

```
        // Handler for the "Shutdown StyleVision" button
        // Shut down the application instance by explicitly releasing the COM object.
        private void shutdownStyleVision_Click(object sender, EventArgs e)
        {
            if (StyleVision != null)
            {
                // Allow shut-down of StyleVision by releasing UI
                StyleVision.Visible = false;

                // Explicitly release the COM object
                try
                {
                    int i =
System.Runtime.InteropServices.Marshal.ReleaseComObject(StyleVision);
```

```
                    while
(System.Runtime.InteropServices.Marshal.ReleaseComObject(StyleVision) > 0) ;
                }
                finally
                {
                    // Disallow subsequent access to this object.
                    StyleVision = null;
                }
            }
        }
```

17.2.1.2.2.3 Opening Documents

The code snippets below (from the AutomateStyleVision example) show how two files are opened via two separate methods assigned to two buttons in the user interface. Both methods use the same Application API access mechanism: Documents.OpenDocument(string).

The AutomateStyleVision example (*see the file Form1.cs*) is located in the C# subfolder of the API Examples folder:

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\StyleVision\2021\%APPNAME%>Examples`

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

Code snippet

```
       // Handler for the "Open OrgChart.pxf" button
        private void openOrgChart_Click(object sender, EventArgs e)
        {
            // Make sure there's a running StyleVision instance, and that it's visible
            StartStyleVision_Click(null, null);

            // Open a sample files installed with the product.
            StyleVision.Documents.OpenDocument(strExamplesFolder + "OrgChart.pxf");
            updateListBox();
        }

        // Handler for the "Open MultiFileOutput.sps" button
        private void openMultiFileOutput_Click(object sender, EventArgs e)
        {
            if (StyleVision == null)
                StartStyleVision_Click(null, null);

            // Open one of the sample files installed with the product.
            StyleVision.Documents.OpenDocument(strExamplesFolder +
"MultiFileOutput.sps");
            updateListBox();
        }
```

The file opened last will be the active file.

17.2.1.2.2.4 Events

The code snippet below (from the AutomateStyleVision example) lists the code for two event handlers. The AutomateStyleVision example (*see the file Form1.cs*) is located in the C# subfolder of the API Examples folder:

Windows 7、Windows 8、Windows 10	C:\Users\<username>\Documents\Altova\StyleVision\2021\%APPNAME%>Examples

You can compile and run the project from within Visual Studio 2010/2012/2013/2015/2017/2019.

Code snippet

```
        delegate void addListBoxItem_delegate(string sText);
        // Called from the UI thread
        private void addListBoxItem(string sText)
        {
            listBoxMessages.Items.Add(sText);
        }
        // Wrapper method to call UI control methods from a worker thread
        void syncWithUIthread(Control ctrl, addListBoxItem_delegate methodToInvoke,
String sText)
        {
            // Control.Invoke: Executes on the UI thread, but calling thread waits for
completion before continuing.
            // Control.BeginInvoke: Executes on the UI thread, and calling thread doesn't
wait for completion.
            if (ctrl.InvokeRequired)
                ctrl.BeginInvoke(methodToInvoke, new Object[] { sText });
        }

        // Event handler for OnDocumentClosed event
        private void handleOnDocumentClosed(StyleVisionLib.Document i_ipDocument)
        {
            String sText = "";

            if (i_ipDocument.Name.Length > 0)
                sText = "Document " + i_ipDocument.Name + " was closed!";

            // Synchronize the calling thread with the UI thread because
            // COM events are triggered from a working thread
            addListBoxItem_delegate methodToInvoke = new
addListBoxItem_delegate(addListBoxItem);
            // Call syncWithUIthread with the following arguments:
            // 1 - listBoxMessages - list box control to display messages from COM events
            // 2 - methodToInvoke  - a C# delegate which points to the method which will
be called from the UI thread
            // 3 - sText           - the text to be displayed in the list box
            syncWithUIthread(listBoxMessages, methodToInvoke, sText);
        }



        private void updateListBox()
        {
            // Iterate through all open documents
```

```
            listBoxMessages.Items.Clear();


            for (int i = 1; i <= StyleVision.Documents.Count; i++)
            {
                StyleVisionLib.Document doc = StyleVision.Documents[i];

                if (doc != null)
                {
                    if (checkBoxEventOnOff.Checked)
                        doc.OnDocumentClosed += new
StyleVisionLib._IDocumentEvents_OnDocumentClosedEventHandler(handleOnDocumentClosed);
                    else
                        doc.OnDocumentClosed -= new
StyleVisionLib._IDocumentEvents_OnDocumentClosedEventHandler(handleOnDocumentClosed);

                    listBoxMessages.Items.Add(doc.Name);
                    StyleVisionLib.SchemaSources sources = doc.SchemaSources;

                    for (int j = 1; j <= sources.Count; j++)
                    {
                        StyleVisionLib.SchemaSource source = sources[j];

                        if (source != null)
                        {
                            listBoxMessages.Items.Add("\tSchema      file name : " +
source.SchemaFileName + "\n");
                            listBoxMessages.Items.Add("\tWorking XML file name : " +
source.WorkingXMLFileName + "\n");
                            listBoxMessages.Items.Add("\tIs main schema source : " +
source.IsMainSchemaSource + "\tType name : " + source.TypeName + "\n");
                        }
                    }
                }
            }
        }
```

17.2.1.2.3 Java

The Application API can be accessed from Java code. To allow accessing the StyleVision automation server directly from Java code, the libraries listed below must reside in the `classpath`. They are installed in the folder: `JavaAPI` in the StyleVision application folder.

- `AltovaAutomation.dll`: a JNI wrapper for Altova automation servers (`AltovaAutomation_x64.dll` in the case of 64-bit versions)
- `AltovaAutomation.jar`: Java classes to access Altova automation servers
- `StyleVisionAPI.jar`: Java classes that wrap the StyleVision automation interface
- `StyleVisionAPI_JavaDoc.zip`: a `Javadoc` file containing help documentation for the Java API

Note: In order to use the Java API, the DLL and Jar files must be on the Java Classpath.

Example Java project

An example Java project is supplied with your product installation. You can test the Java project and modify and use it as you like. For more details of the example Java project, see the section, Example Java Project.

Rules for mapping the Application API names to Java

The rules for mapping between the Application API and the Java wrapper are as follows:

- ***Classes and class names***
  For every interface of the StyleVision automation interface a Java class exists with the name of the interface.

- ***Method names***
  Method names on the Java interface are the same as used on the COM interfaces but start with a small letter to conform to Java naming conventions. To access COM properties, Java methods that prefix the property name with `get` and `set` can be used. If a property does not support write-access, no setter method is available. Example: For the `Name` property of the `Document` interface, the Java methods `getName` and `setName` are available.

- ***Enumerations***
  For every enumeration defined in the automation interface, a Java enumeration is defined with the same name and values.

- ***Events and event handlers***
  For every interface in the automation interface that supports events, a Java interface with the same name plus '`Event`' is available. To simplify the overloading of single events, a Java class with default implementations for all events is provided. The name of this Java class is the name of the event interface plus '`DefaultHandler`'. For example:
  `Application`: Java class to access the application
  `ApplicationEvents`: Events interface for the Application
  `ApplicationEventsDefaultHandler`: Default handler for `ApplicationEvents`

Exceptions to mapping rules

There are some exceptions to the rules listed above. These are listed below:

Interface	Java name
`Document`, method `SetEncoding`	`setFileEncoding`
`AuthenticView`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Goto`	`gotoElement`
`AuthenticRange`, method `Clone`	`cloneRange`

This section

This section explains how some basic StyleVision functionality can be accessed from Java code. It is organized into the following sub-sections:

- Example Java Project
- Application Startup and Shutdown
- Simple Document Access
- Iterations
- Event Handlers

17.2.1.2.3.1 Example Java Project

The StyleVision installation package contains an example Java project, located in the the `API\Java` subfolder of the `Examples` folder :

Windows 7、Windows 8、Windows 10	`C:\Users\<username>\Documents\Altova\StyleVision\2021\%APPNAME%>Examples`

This folder contains Java examples for the StyleVision API. You can test it directly from the command line using the batch file `BuildAndRun.bat`, or you can compile and run the example project from within Eclipse. See below for instructions on how to use these procedures.

File list

The Java examples folder contains all the files required to run the example project. These files are listed below. If you are using a 64-bit version of the application, some filenames contain `_x64` in the name. These filenames are indicated with `(_x64)`.

`AltovaAutomation(_x64).dll`	Java-COM bridge: DLL part
`AltovaAutomation.jar`	Java-COM bridge: Java library part
`StyleVisionAPI.jar`	Java classes of the StyleVision API
`RunStyleVision.java`	Java example source code
`BuildAndRun.bat`	Batch file to compile and run example code from the command line prompt. Expects folder where Java Virtual Machine resides as parameter.
`.classpath`	Eclipse project helper file
`.project`	Eclipse project file
`StyleVision_JavaDoc.zip`	`Javadoc` file containing help documentation for the Java API

What the example does

The example starts up StyleVision and performs a few operations, including opening and closing documents. When done, StyleVision stays open. You must close it manually.

- Start StyleVision: Starts StyleVision, which is registered as an automation server, or activates StyleVision if it is already running.
- Open OrgChart.pxf: Locates one of the example documents installed with StyleVision and opens it.
- Iteration and Changing the View Mode: Shows how to iterate through open documents.
- Event Handling: Shows how to handle StyleVision events.
- Shut down StyleVision: Shuts down StyleVision.

You can modify the example in any way you like and run it.

Running the example from the command line

To run the example from the command line, open a command prompt window, go to the Java folder of the API Examples folder (*see above for location*), and then type:

```
buildAndRun.bat "<Path-to-the-Java-bin-folder>"
```


(C) 2015-2021 Altova GmbH

The Java binary folder must be that of a JDK 1.5 or later installation on your computer. Press the **Return** key. The Java source in `RunStyleVision.java` will be compiled and then executed.

Loading the example in Eclipse

Open Eclipse and use the **Import | Existing Projects into Workspace** command to add the Eclipse project file (`.project`) located in the Java folder of the API Examples folder (*see above for location*). The project `RunStyleVision` will then appear in your Package Explorer or Navigator. Select the project and then the command **Run as | Java Application** to execute the example.

Note: You can select a class name or method of the Java API and press F1 to get help for that class or method.

Java source code listing

The Java source code in the example file `RunStyleVision.java` is listed below with comments.

```
01 // Access general JAVA-COM bridge classes
02 import com.altova.automation.libs.*;
03
04 // Access StyleVision Java-COM bridge
05 import com.altova.automation.StyleVision.*;
06 import com.altova.automation.StyleVision.Enums.ENUMApplicationStatus;
07
08 /**
09  * A simple example that starts the COM server and performs a View operations on it.
10  * Feel free to extend.
11  */
12 public class RunStyleVision
13 {
14   public static void main(String[] args)
15   {
16     // An instance of the application.
17     Application stylevision = null;
18
19     // Instead of COM error-handling, use Java exception mechanism.
20     try
21     {
22       // Start StyleVision as COM server.
23       stylevision = new Application();
24
25       ENUMApplicationStatus status = ENUMApplicationStatus.eApplicationRunning;
26       do{
27         // Check the application status
28         status = stylevision.getStatus();
29         System.out.println("status : " + status + "\n");
30       } while (status != ENUMApplicationStatus.eApplicationRunning);
31
32       // COM servers start up invisible, so we make the server visible
33       stylevision.setVisible(true);
34
35       // Locate samples installed with the product.
36       String strExamplesFolder = System.getenv("USERPROFILE") + "\\My Documents\
\Altova\\StyleVision2012\\StyleVisionExamples\\";
37
38       // Open two files from the product samples.
39       stylevision.getDocuments().openDocument(strExamplesFolder + "OrgChart.pxf");
```

```
40       stylevision.getDocuments().openDocument(strExamplesFolder +
"BiggestCitiesPerContinent.sps");
41
42       // Iterate through all open documents
43       for (Document doc:stylevision.getDocuments())
44       {
45         System.out.println("Document name : " + doc.getName() + "\n");
46         SchemaSources sources = doc.getSchemaSources();
47
48         for (int i = 1; i <= sources.getCount(); i++)
49         {
50           SchemaSource source = sources.getItem(i);
51           System.out.println("\tSchema      file name : " + source.getSchemaFileName()
+ "\n");
52           System.out.println("\tWorking XML file name : " +
source.getWorkingXMLFileName() + "\n");
53           System.out.println("\tIs main schema source : " +
source.getIsMainSchemaSource() + "\tType name : " + source.getTypeName() + "\n");
54         }
55       }
56       // The following lines attach to the document events using a default
implementation
57       // for the events and override one of its methods.
58       // If you want to override all document events it is better to derive your
listener class
59       // from DocumentEvents and implement all methods of this interface.
60       Document doc = stylevision.getActiveDocument();
61       doc.addListener(new DocumentEventsDefaultHandler()
62       {
63         @Override
64         public void onDocumentClosed(Document i_ipDoc) throws AutomationException
65         {
66           System.out.println("Document " + i_ipDoc.getName() + " requested closing.");
67         }
68       });
69       doc.close();
70       doc = null;
71
72       System.out.println("Watch StyleVision!");
73     }
74     catch (AutomationException e)
75     {
76       e.printStackTrace();
77     }
78     finally
79     {
80       // Make sure that StyleVision can shut down properly.
81       if (stylevision != null)
82         stylevision.dispose();
83
84       // Since the COM server was made visible and still is visible, it will keep
running
85       // and needs to be closed manually.
86       System.out.println("Now close StyleVision!");
87     }
88   }
89 }
```

17.2.1.2.3.2 Application Startup and Shutdown

The code listings below show how the application can be started up and shut down.

Application startup

Before starting up the application, the appropriate classes must be imported (*see below*).

```
01 // Access general JAVA-COM bridge classes
02 import com.altova.automation.libs.*;
03
04 // Access StyleVision Java-COM bridge
05 import com.altova.automation.StyleVision.*;
06 import com.altova.automation.StyleVision.Enums.ENUMApplicationStatus;
07
08 /**
09  * A simple example that starts the COM server and performs a View operations on it.
10  * Feel free to extend.
11  */
12 public class RunStyleVision
13 {
14   public static void main(String[] args)
15   {
16     // An instance of the application.
17     Application stylevision = null;
18
19     // Instead of COM error-handling, use Java exception mechanism.
20     try
21     {
22       // Start StyleVision as COM server.
23       stylevision = new Application();
24
25       ENUMApplicationStatus status = ENUMApplicationStatus.eApplicationRunning;
26       do{
27         // Check the application status
28         status = stylevision.getStatus();
29         System.out.println("status : " + status + "\n");
30       } while (status != ENUMApplicationStatus.eApplicationRunning);
31
32       // COM servers start up invisible, so we make the server visible
33       stylevision.setVisible(true);
34
35 ...
36     }
37   }
38 }
```

Application shutdown

The application can be shut down as shown below.

```
1 {
2       // Make sure that StyleVision can shut down properly.
3       if (stylevision != null)
4         stylevision.dispose();
5
6       // Since the COM server was made visible and still is visible, it will keep
running
7       // and needs to be closed manually.
```

```
8       System.out.println("Now close StyleVision!");
9 }
```

17.2.1.2.3.3 Simple Document Access

The code listing below shows how to open a document.

```
1  // Locate samples installed with the product.
2  String strExamplesFolder = System.getenv("USERPROFILE") + "\\My Documents\\Altova\
\StyleVision2012\\StyleVisionExamples\\";
3
4  // We open two files form the product samples.
5  stylevision.getDocuments().openDocument(strExamplesFolder + "OrgChart.pxf");
6  stylevision.getDocuments().openDocument(strExamplesFolder +
"BiggestCitiesPerContinent.sps");
```

17.2.1.2.3.4 Iterations

The listing below shows how to iterate through open documents.

```
01 // Iterate through all open documents
02 for (Document doc:stylevision.getDocuments())
03   {
04      System.out.println("Document name : " + doc.getName() + "\n");
05      SchemaSources sources = doc.getSchemaSources();
06
07      for (int i = 1; i <= sources.getCount(); i++)
08      {
09        SchemaSource source = sources.getItem(i);
10        System.out.println("\tSchema      file name : " + source.getSchemaFileName() +
"\n");
11        System.out.println("\tWorking XML file name : " +
source.getWorkingXMLFileName() + "\n");
12        System.out.println("\tIs main schema source : " +
source.getIsMainSchemaSource() + "\tType name : " + source.getTypeName() + "\n");
13      }
14   }
```

17.2.1.2.3.5 Event Handlers

The listing below shows how to listen for and use events.

```
01 // The following lines attach to the document events using a default implementation
02 // for the events and override one of its methods.
03 // If you want to override all document events it is better to derive your listener
class
04 // from DocumentEvents and implement all methods of this interface.
05 Document doc = stylevision.getActiveDocument();
06 doc.addListener(new DocumentEventsDefaultHandler()
```

```
07 {
08   @Override
09   public void onDocumentClosed(Document i_ipDoc) throws AutomationException
10   {
11     System.out.println("Document " + i_ipDoc.getName() + " requested closing.");
12   }
13 });
14 doc.close();
15 doc = null;
```

17.2.2 Interfaces

This chapter contains the reference of the StyleVision Type Library.

17.2.2.1 Application

Methods

Quit

Properties
Application
Parent

ActiveDocument
Documents

Status
MajorVersion
MinorVersion
Edition
IsAPISupported
ServicePackVersion

Description
Application is the root for all other objects. It is the only object you can create by CreateObject (VisualBasic) or other similar COM related functions.

Example

```
Dim objSpy As Application
Set objSpy = CreateObject("XMLSpy.Application")
```

17.2.2.1.1 Events

17.2.2.1.1.1 OnShutDown

Event: OnShutDown()

Description
Sent just before StyleVision shuts down.

17.2.2.1.2 ActiveDocument

Property: ActiveDocument as Document

Description
Reference to the active document. If no document is open, ActiveDocument is null (nothing).

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.1.3 Application

Property: Application as Application (read-only)

Description
Accesses the StyleVision application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.1.4 Documents

Property: Documents as Documents

Description
Collection of all open documents.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

17.2.2.1.5 Edition

Property: Edition as String

Description
Returns the edition of the application, for example `Altova StyleVision Enterprise Edition` for the Enterprise edition.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.1.6 IsAPISupported

Property: IsAPISupported as Boolean

Description
Returns whether the API is supported in this version or not.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.1.7 MajorVersion

Property: MajorVersion as Integer

Description
Returns the application version's major number, for example `15` for 2013 versions, and `16` for 2014 versions..

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.1.8 MinorVersion

Property: MinorVersion as Integer

Description
Returns the application version's minor number.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.1.9 NewDocument

Method: NewDocument as Document

Return Value
None

Description
Creates a new empty document based on the previous template.

Errors
1000 The application object is invalid.
1005 Error when creating a new document
1006 Cannot create document

17.2.2.1.10 OpenDocument

Method: OpenDocument(strFileName as *String*) as Document

Return Value
None

Description
Opens an existing SPS file.

Errors
1000 The application object is invalid.
1002 Invalid file extension.
1003 Error when opening document.
1004 Cannot open document.

17.2.2.1.11 Parent

Property: Parent as Application (read-only)

Description
Accesses the StyleVision application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

17.2.2.1.12 Quit

Method: Quit()

Return Value
None

Description
This method terminates StyleVision. All modified documents will be closed without saving the changes. This is also true for an open project.

If StyleVision was automatically started as an automation server by a client program, the application will not shut down automatically when your client program shuts down if a project or any document is still open. Use the Quit method to ensure automatic shut-down.

Errors
1111 The application object is no longer valid.

17.2.2.1.13 ServicePackVersion

Property: ServicePackVersion as Long

Description
Returns the Service Pack version number of the application. Eg: 1 for 2010 R2 SP1

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.1.14 Status

Property: Status as ENUMApplicationStatus

Description
Returns the current status of the running application.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.1.15 Visible

Property: Visible as VARIANT_BOOL

Description
Sets or gets the visibility attribute of StyleVision.

Errors

1110 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.2 AppOutputLine

Represents a message line. Its structure is more detailed and can contain a collection of child lines, thereby forming a tree of message lines.

Properties and Methods

Properties to navigate the object model:

Application
Parent

Line access:

GetLineSeverity
GetLineSymbol
GetLineText
GetLineTextEx
GetLineTextWithChildren
GetLineTextWithChildrenEx

A single AppOutputLine consists of one or more sub-lines.
Sub-line access:

GetLineCount

A sub-line consists of one or more cells.
Cell access:

GetCellCountInLine
GetCellIcon
GetCellSymbol
GetCellText
GetCellTextDecoration
GetIsCellText

Below an AppOutputLine there can be zero, one, or more child lines which themselves are of type AppOutputLine, which thus form a tree structure.

Child lines access:

ChildLines

(C) 2015-2021 Altova GmbH

17.2.2.2.1 Application

Property: Application as Application (read-only)

Description
Retrieves the application's top-level object.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.2 ChildLines

Property: ChildLines as AppOutputLines (read-only)

Description
Returns a collection of the current line's direct child lines.

Errors
4100 The application object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.3 GetCellCountInLine

Method: GetCellCountInLine (*nLine* as Long) as Long

Description
Gets the number of cells in the sub-line indicated by *nLine* in the current AppOutputLine.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.4 GetCellIcon

Method: GetCellIcon (*nLine* as Long, *nCell* as Long) as Long

Description
Gets the icon of the cell indicated by *nCell* in the current AppOutputLine's sub-line indicated by *nLine*.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.5 GetCellSymbol

Method: GetCellSymbol (*nLine* as Long, *nCell* as Long) as AppOutputLineSymbol

Description
Gets the symbol of the cell indicated by *nCell* in the current AppOutputLine's sub-line indicated by *nLine*.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.6 GetCellText

Method: GetCellText (*nLine* as Long, *nCell* as Long) as String

Description
Gets the text of the cell indicated by *nCell* in the current AppOutputLine's sub-line indicated by *nLine*.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.7 GetCellTextDecoration

Method: GetCellTextDecoration (*nLine* as Long, *nCell* as Long) as Long

Description
Gets the decoration of the text cell indicated by *nCell* in the current AppOutputLine's sub-line indicated by *nLine*.
It can be one of the ENUMAppOutputLine_TextDecoration values.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.8 GetIsCellText

Method: GetIsCellText (*nLine* as Long, *nCell* as Long) as String

(C) 2015-2021 Altova GmbH

Description
Gets the text of the cell indicated by *nCell* in the current AppOutputLine's sub-line indicated by *nLine*.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.9 GetLineCount

Method: GetLineCount () as Long

Description
Gets the number of sub-lines the current line consists of.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.10 GetLineSeverity

Method: GetLineSeverity () as Long

Description
Gets the severity of the line. It can be one of the ENUMAppOutputLine_Severity values:

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.11 GetLineSymbol

Method: GetLineSymbol () as AppOutputLineSymbol

Description
Gets the symbol assigned to the whole line.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.12 GetLineText

Method: GetLineText () as String

Description
Gets the contents of the line as text.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.13 GetLineTextEx

Method: GetLineTextEx (*psTextPartSeperator* as String, *psLineSeperator* as String) as String

Description
Gets the contents of the line as text using the specified part and line separators.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.14 GetLineTextWithChildren

Method: GetLineTextWithChildren () as String

Description
Gets the contents of the line including all child and descendant lines as text.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.15 GetLineTextWithChildrenEx

Method: GetLineTextWithChildrenEx (*psPartSep* as String, *psLineSep* as String, *psTabSep* as String, *psItemSep* as String) as String

Description
Gets the contents of the line including all child and descendant lines as text using the specified part, line, tab and item separators.

Errors

(C) 2015-2021 Altova GmbH

4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.2.16 Parent

Property: Parent as AppOutputLines (read-only)

Description
The parent object according to the object model.

Errors
4100 The object is no longer valid.
4101 Invalid address for the return parameter was specified.

17.2.2.3 AppOutputLines

Represents a collection of `AppOutputLine` message lines.

Properties and Methods
Properties to navigate the object model:

`Application`
`Parent`

Iterating through the collection:

`Count`
`Item`

17.2.2.3.1 Application

Property: Application as Application (read-only)

Description
Retrieves the application's top-level object.

Errors
4000 The object is no longer valid.
4001 Invalid address for the return parameter was specified.

17.2.2.3.2 Count

Property: Count as Integer (read-only)

Description
Retrieves the number of lines in the collection.

Errors
4000 The object is no longer valid.
4001 Invalid address for the return parameter was specified.

17.2.2.3.3 Item

Property: Item (*nIndex* as Integer) as AppOutputLine(read-only)

Description
Retrieves the line at nIndex from the collection. Indices start with 1.

Errors
4000 The object is no longer valid.
4001 Invalid address for the return parameter was specified.

17.2.2.3.4 Parent

Property: Parent as AppOutputLine (read-only)

Description
The parent object according to the object model.

Errors
4000 The object is no longer valid.
4001 Invalid address for the return parameter was specified.

17.2.2.4 AppOutputLineSymbol

An `AppOutputLineSymbol` represents a link in an `AppOutputLine` message line which can be clicked in the StyleVision Messages window. It is applied to a cell of an `AppOutputLine` or to the whole line itself.

Properties and Methods
Properties to navigate the object model:

`Application`

(C) 2015-2021 Altova GmbH

Parent

Access to AppOutputLineSymbol methods:

GetSymbolHREF
GetSymbolID
IsSymbolHREF

17.2.2.4.1 Application

Property: Application as Application (read-only)

Description
Retrieves the application's top-level object.

Errors
4200 The object is no longer valid.
4201 Invalid address for the return parameter was specified.

17.2.2.4.2 GetSymbolHREF

Method: GetSymbolHREF () as String

Description
If the symbol is of type URL, returns the URL as a string.

Errors
4200 The object is no longer valid.
4201 Invalid address for the return parameter was specified.

.

17.2.2.4.3 GetSymbolID

Method: GetSymbolHREF () as Long

Description
Gets the ID of the symbol.

Errors
4200 The object is no longer valid.
4201 Invalid address for the return parameter was specified.

17.2.2.4.4 IsSymbolHREF

Method: IsSymbolHREF () as Boolean

Description
Indicates if the symbol is of kind URL.

Errors
4200 The object is no longer valid.
4201 Invalid address for the return parameter was specified.

17.2.2.4.5 Parent

Property: Parent as Application (read-only)

Description
The parent object according to the object model.

Errors
4200 The object is no longer valid.
4201 Invalid address for the return parameter was specified.

17.2.2.5 AuthenticContextMenu

The context menu interface provides the means for the user to customize the context menus shown in Authentic. The interface has the methods listed in this section.

17.2.2.5.1 CountItems

Method: CountItems`()` nItems as long

Return Value
Returns the number of menu items.

Errors
2501 Invalid object.

17.2.2.5.2 DeleteItem

Method: DeleteItem`(IndexPosition as long)`

(C) 2015-2021 Altova GmbH

Return Value
Deletes the menu item that has the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

17.2.2.5.3 GetItemText

Method: GetItemText`(IndexPosition as long)` MenuItemName as string

Return Value
Gets the name of the menu item located at the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

17.2.2.5.4 InsertItem

Method: InsertItem`(IndexPosition as long, MenuItemName as string, MacroName as string)`

Return Value
Inserts a user-defined menu item at the position in the menu specified in the first parameter and having the name submitted in the second parameter. The menu item will start a macro, so a valid macro name must be submitted.

Errors
2501 Invalid object
2502 Invalid index
2503 No such macro
2504 Internal error

17.2.2.5.5 SetItemText

Method: SetItemText`(IndexPosition as long, MenuItemName as string)`

Return Value
Sets the name of the menu item located at the index position submitted in the first parameter.

Errors
2501 Invalid object
2502 Invalid index

17.2.2.6 AuthenticEventContext

The `EventContext` interface gives access to many properties of the context in which a macro is executed.

17.2.2.6.1 EvaluateXPath

Method: EvaluateXPath (strExpression as string) as strValue as string

Return Value
The method evaluates the XPath expression in the context of the node within which the event was triggered and returns a string.

Description
`EvaluateXPath()` executes an XPath expression with the given event context. The result is returned as a string, in the case of a sequence it is a space-separated string.

Errors

2201	Invalid object.
2202	No context.
2209	Invalid parameter.
2210	Internal error.
2211	XPath error.

17.2.2.6.2 GetEventContextType

Method: GetEventContextType `()` Type as `AuthenticEventContextType` enumeration

Return Value
Returns the context node type.

Description
`GetEventContextType` allows the user to determine whether the macro is in an XML node or in an XPath atomic item context. The enumeration `AuthenticEventContextType` is defined as follows:

authenticEventContextXML,
authenticEventContextAtomicItem,
authenticEventContextOther

If the context is a normal XML node, the `GetXMLNode()` function gives access to it (returns `NULL` if not).

Errors

2201	Invalid object.
2202	No context.
2209	Invalid parameter.

17.2.2.6.3 GetNormalizedTextValue

Method: GetNormalizedTextValue() strValue as string

Return Value
Returns the value of the current node as string

Errors
2201 Invalid object.
2202 No context.
2203 Invalid context
2209 Invalid parameter.

17.2.2.6.4 GetVariableValue

Method: GetVariableValue(strName as string) strValue as string

Return Value
Gets the value of the variable submitted as the parameter.

Description
GetVariableValue gets the variable's value in the scope of the context.

```
nZoom = parseInt( AuthenticView.EventContext.GetVariableValue( 'Zoom' ) );
if ( nZoom > 1 )
{
    AuthenticView.EventContext.SetVariableValue( 'Zoom', nZoom - 1 );
}
```

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2209 Invalid parameter

17.2.2.6.5 GetXMLNode

Method: GetXMLNode() Node as XMLData object

Return Value
Returns the context XML node or NULL

Errors
2201 Invalid object.
2202 No context.

2203 Invalid context
2209 Invalid parameter.

17.2.2.6.6 IsAvailable

Method: IsAvailable`()` as Boolean

Return Value
Returns true if `EventContext` is set, false otherwise.

Errors
2201 Invalid object.

17.2.2.6.7 SetVariableValue

Method: SetVariableValue`(strName as string, strValue as string)`

Return Value
Sets the value (second parameter) of the variable submitted in the first parameter.

Description
`SetVariableValue` sets the variable's value in the scope of the context.

```
nZoom = parseInt( AuthenticView.EventContext.GetVariableValue( 'Zoom' ) );
if ( nZoom > 1 )
{
    AuthenticView.EventContext.SetVariableValue( 'Zoom', nZoom - 1 );
}
```

Errors
2201 Invalid object.
2202 No context.
2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2207 Variable read-only
2208 No modification allowed

17.2.2.7 AuthenticRange

The first table lists the properties and methods of AuthenticRange that can be used to navigate through the document and select specific portions.

Properties	Methods

(C) 2015-2021 Altova GmbH

Application	Clone	MoveBegin
FirstTextPosition	CollapsToBegin	MoveEnd
FirstXMLData	CollapsToEnd	NextCursorPosition
FirstXMLDataOffset	ExpandTo	PreviousCursorPosition
LastTextPosition	Goto	Select
LastXMLData	GotoNext	SelectNext
LastXMLDataOffset	GotoPrevious	SelectPrevious
Parent	IsEmpty	SetFromRange
	IsEqual	

The following table lists the content modification methods, most of which can be found on the right/button mouse menu.

Properties	Edit operations	Dynamic table operations
Text	Copy	AppendRow
	Cut	DeleteRow
	Delete	DuplicateRow
	IsCopyEnabled	InsertRow
	IsCutEnabled	IsFirstRow
	IsDeleteEnabled	IsInDynamicTable
	IsPasteEnabled	IsLastRow
	Paste	MoveRowDown
		MoveRowUp

The following methods provide the functionality of the Authentic entry helper windows for range objects.

Operations of the entry helper windows		
Elements	**Attributes**	**Entities**
CanPerformActionWith	GetElementAttributeValue	GetEntityNames
CanPerformAction	GetElementAttributeNames	InsertEntity
PerformAction	GetElementHierarchy	
	HasElementAttribute	
	IsTextStateApplied	
	SetElementAttributeValue	

Description
AuthenticRange objects are the 'cursor' selections of the automation interface. You can use them to point to any cursor position in the Authentic view, or select a portion of the document. The operations available for AuthenticRange objects then work on this selection in the same way, as the corresponding operations of the user interface do with the current user interface selection. The main difference is that you can use an arbitrary number of AuthenticRange objects at the same time, whereas there is exactly one cursor selection in the user interface.

To get to an initial range object use AuthenticView.Selection, to obtain a range corresponding with the current cursor selection in the user interface. Alternatively, some trivial ranges are accessible via the read/only properties AuthenticView.DocumentBegin, AuthenticView.DocumentEnd, and AuthenticView.WholeDocument. The most flexible method is AuthenticView.Goto, which allows navigation to a specific portion of the document within one call. For more complex selections, combine the above with the various navigation methods on range objects listed in the first table on this page.

Another method to select a portion of the document is to use the position properties of the range object. Two positioning systems are available and can be combined arbitrarily:

- **Absolute** text cursor positions, starting with position 0 at the document beginning, can be set and retrieved for the beginning and end of a range. For more information see FirstTextPosition and LastTextPosition. This method requires complex internal calculations and should be used with care.

- The **XMLData** element and a text position inside this element, can be set and retrieved for the beginning and end of a range. For more information see FirstXMLData, FirstXMLDataOffset, LastXMLData, and LastXMLDataOffset. This method is very efficient but requires knowledge of the underlying document structure. It can be used to locate XMLData objects and perform operations on them otherwise not accessible through the user interface.

Modifications to the document content can be achieved by various methods:

- The Text property allows you to retrieve the document text selected by the range object. If set, the selected document text gets replaced with the new text.
- The standard document edit functions Cut, Copy, Paste and Delete.
- Table operations for tables that can grow dynamically.
- Methods that map the functionality of the Authentic entry helper windows.
- Access to the XMLData objects of the underlying document to modify them directly.

17.2.2.7.1 AppendRow

Method: AppendRow()as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a new row at the end of the selected table. The selection of the range is modified to point to the beginning of the new row. The function returns *true* if the append operation was successful, otherwise *false*.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' -------------------------------------------------------
' Scripting environment - VBScript
' Append row at end of current dynamically growable table
' -------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
        objRange.AppendRow
        ' objRange points to beginning of new row
        objRange.Select
End If
```

17.2.2.7.2 Application

Property: Application as Application (read-only)

Description
Accesses the StyleVision application object.

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.

17.2.2.7.3 CanPerformAction

Method: CanPerformAction (*eAction* as SPYAuthenticActions, *strElementName* as String) as Boolean

Description
CanPerformAction and its related methods enable access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content, without having to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If the location can be found, the method returns *True*, otherwise it returns *False*.

HINT: To find out all valid element names for a given action, use CanPerformActionWith.

Errors

2001	The authentic range object or its related view object is no longer valid.
2005	Invalid address for the return parameter was specified.
2007	Invalid action was specified.

Examples
See PerformAction.

17.2.2.7.4 CanPerformActionWith

Method: CanPerformActionWith (*eAction* as SPYAuthenticActions, *out_arrElementNames* as Variant)

Description
PerformActionWith and its related methods, enable access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content without having to know exactly where the modification will take place.

This method returns an array of those element names that the specified action can be performed with.

HINT: To apply the action use CanPerformActionWith.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples
See PerformAction.

17.2.2.7.5 Clone

Method: Clone() as AuthenticRange

Description
Returns a copy of the range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.6 CollapsToBegin

Method: CollapsToBegin() as AuthenticRange

Description
Sets the end of the range object to its begin. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

17.2.2.7.7 CollapsToEnd

Method: CollapsToEnd() as AuthenticRange

Description
Sets the beginning of the range object to its end. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.8 Copy

Method: Copy() as Boolean

Description
Returns *False* if the range contains no portions of the document that may be copied.
Returns *True* if text, and in case of fully selected XML elements the elements as well, has been copied to the copy/paste buffer.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.9 Cut

Method: Cut() as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted.
Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document and saved in the copy/paste buffer.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.10 Delete

Method: Delete() as Boolean

Description
Returns *False* if the range contains portions of the document that may not be deleted.
Returns *True* after text, and in case of fully selected XML elements the elements as well, has been deleted from the document.

Errors

2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.11 DeleteRow

Method: DeleteRow() as Boolean

Description

If the beginning of the range is inside a dynamic table, this method deletes the selected row. The selection of the range gets modified to point to the next element after the deleted row. The function returns *true,* if the delete operation was successful, otherwise *false*.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' -------------------------------------------------------
' Scripting environment - VBScript
' Delete selected row from dynamically growing table
' -------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we are in a table
If objRange.IsInDynamicTable Then
        objRange.DeleteRow
End If
```

17.2.2.7.12 DuplicateRow

Method: DuplicateRow() as Boolean

Description

If the beginning of the range is inside a dynamic table, this method inserts a duplicate of the current row after the selected one. The selection of the range gets modified to point to the beginning of the new row. The function returns *true* if the duplicate operation was successful, otherwise *false*.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' -------------------------------------------------------
' Scripting environment - VBScript
' duplicate row in current dynamically growable table
```


(C) 2015-2021 Altova GmbH

```
' -------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
        objRange.DuplicateRow
        ' objRange points to beginning of new row
        objRange.Select
End If
```

17.2.2.7.13 EvaluateXPath

Method: EvaluateXPath (strExpression as string) strValue as string

Return Value
The method returns a string

Description
`EvaluateXPath()` executes an XPath expression with the context node being the beginning of the range selection. The result is returned as a string, in the case of a sequence it is a space-separated string. If XML context node is irrelevant, the user may provide any node, like `AuthenticView.XMLDataRoot`.

Errors

2001	Invalid object
2005	Invalid parameter
2008	Internal error
2202	Missing context node
2211	XPath error

17.2.2.7.14 ExpandTo

Method: ExpandTo (*eKind* as SPYAuthenticElementKind), as AuthenticRange

Description
Selects the whole element of type eKind, that starts at, or contains, the first cursor position of the range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Range expansion would be beyond end of document.
2005	Invalid address for the return parameter was specified.

17.2.2.7.15 FirstTextPosition

Property: FirstTextPosition as Long

Description
Set or get the left-most text position index of the range object. This index is always less or equal to LastTextPosition. Indexing starts with 0 at document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decrementing the test position by 1 has the same effect as the cursor-left key.

If you set FirstTextPosition to a value greater than the current LastTextPosition, LastTextPosition gets set to the new FirstTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors

2001	The authentic range object, or its related view object is not valid.
2005	Invalid address for the return parameter was specified.
2006	A text position outside the document was specified.

Examples

```
' ---------------------------------------
' Scripting environment - VBScript
' ---------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

' let's create a range that selects the whole document
' in an inefficient way
Dim objRange
' we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

' let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
	MsgBox "Test using direct text cursor positioning was ok"
Else
	MsgBox "Ooops!"
End If
```

17.2.2.7.16 FirstXMLData

Property: FirstXMLData as XMLData

Description
Set or get the first XMLData element in the underlying document that is partially, or completely selected by the range. The exact beginning of the selection is defined by the FirstXMLDataOffset attribute.

Whenever you set FirstXMLData to a new data object, FirstXMLDataOffset gets set to the first cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set FirstXMLData /

FirstXMLDataOffset selects a position greater then the current LastXMLData / LastXMLDataOffset, the latter gets moved to the new start position.

HINT: You can use the FirstXMLData and LastXMLData properties to directly access and manipulate the underlying XML document in those cases where the methods available with the AuthenticRange object are not sufficient.

Errors

2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2008 Internal error
2009 The XMLData object cannot be accessed.

Examples

```
' ----------------------------------------------
' Scripting environment - VBScript
' show name of currently selected XMLData element
' ----------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objXmlData
Set objXMLData = objAuthenticView.Selection.FirstXMLData
' authentic view adds a 'text' child element to elements
' of the document which have content. So we have to go one
' element up.
Set objXMLData = objXMLData.Parent
MsgBox "Current selection selects element " & objXMLData.Name
```

17.2.2.7.17 FirstXMLDataOffset

Property: FirstXMLDataOffset as Long

Description

Set or get the cursor position offset inside FirstXMLData element for the beginning of the range. Offset positions are based on the characters returned by the Text property, and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on screen. Although the data offset is based on this text, there only two valid offset positions, one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.

- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If FirstXMLData / FirstXMLDataOffset selects a position after the current LastXMLData / LastXMLDataOffset, the latter gets moved to the new start position.

Errors

2001 The authentic range object, or its related view object is not valid.

2005 Invalid offset was specified.
Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
	objRange.Select()
Else
	MsgBox "Oops"
End If
```

17.2.2.7.18 GetElementAttributeNames

Method: GetElementAttributeNames (*strElementName* as String, *out_arrAttributeNames* as Variant)

Description

Retrieve the names of all attributes for the enclosing element with the specified name. Use the element/attribute pairs, to set or get the attribute value with the methods GetElementAttributeValue and SetElementAttributeValue.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid address for the return parameter was specified.

Examples

See SetElementAttributeValue.

17.2.2.7.19 GetElementAttributeValue

Method: GetElementAttributeValue (*strElementName* as String, *strAttributeName* as String) as String

Description
Retrieve the value of the attribute specified in strAttributeName, for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames, or HasElementAttribute.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid attribute name was specified.
Invalid address for the return parameter was specified.

Examples
See SetElementAttributeValue.

17.2.2.7.20 GetElementHierarchy

Method: GetElementHierarchy (*out_arrElementNames* as Variant)

Description
Retrieve the names of all XML elements that are parents of the current selection. Inner elements get listed before enclosing elements. An empty list is returned whenever the current selection is not inside a single XMLData element.

The names of the element hierarchy, together with the range object uniquely identify XMLData elements in the document. The attributes of these elements can be directly accessed by GetElementAttributeNames, and related methods.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.21 GetEntityNames

Method: GetEntityNames (*out_arrEntityNames* as Variant)

Description
Retrieve the names of all defined entities. The list of retrieved entities is independent of the current selection, or location. Use one of these names with the InsertEntity function.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples
See: GetElementHierarchy and InsertEntity.

17.2.2.7.22 GetVariableValue

Method: GetVariableValue`(strName as string) strVal as string`

Return Value
Gets the value of the variable named as the method's parameter.

Errors

2001	Invalid object.
2202	No context.
2204	No such variable in scope
2205	Variable cannot be evaluated
2206	Variable returns sequence
2209	Invalid parameter

17.2.2.7.23 Goto

Method: Goto (*eKind* as SPYAuthenticElementKind, *nCount* as Long, *eFrom* as SPYAuthenticDocumentPosition) as AuthenticRange

Description
Sets the range to point to the beginning of the nCount element of type eKind. The start position is defined by the parameter eFrom.

Use positive values for nCount to navigate to the document end. Use negative values to navigate to the beginning of the document. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before begin of document.
2005	Invalid element kind specified.
	Invalid start position specified.
	Invalid address for the return parameter was specified.

17.2.2.7.24 GotoNext

Method: GotoNext (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description

(C) 2015-2021 Altova GmbH

Sets the range to the beginning of the next element of type eKind. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document word-by-word
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False

On Error Resume Next
While Not bEndOfDocument
	objRange.GotoNext(spyAuthenticWord).Select
	If ((Err.number - vbObjecterror) = 2003) Then
		bEndOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise  ' forward error
	End If
Wend
```

17.2.2.7.25 GotoNextCursorPosition

Method: GotoNextCursorPosition() as AuthenticRange

Description

Sets the range to the next cursor position after its current end position. Returns the modified object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2005 Invalid address for the return parameter was specified.

17.2.2.7.26 GotoPrevious

Method: GotoPrevious (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description
Sets the range to the beginning of the element of type eKind which is before the beginning of the current range. The method returns the modified range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2004 Target lies before beginning of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document tag-by-tag
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
Dim bBeginOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
	objRange.GotoPrevious(spyAuthenticTag).Select
	If ((Err.number - vbObjecterror) = 2004) Then
		bBeginOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise  ' forward error
	End If
Wend
```

17.2.2.7.27 GotoPreviousCursorPosition

Method: GotoPreviousCursorPosition() as AuthenticRange

Description
Set the range to the cursor position immediately before the current position. Returns the modified object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2004 Target lies before begin of document.
2005 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

17.2.2.7.28 HasElementAttribute

Method: HasElementAttribute (*strElementName* as String, *strAttributeName* as String) as Boolean

Description
Tests if the enclosing element with name strElementName, supports the attribute specified in strAttributeName.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid address for the return parameter was specified.

17.2.2.7.29 InsertEntity

Method: InsertEntity (*strEntityName* as String)

Description
Replace the ranges selection with the specified entity. The specified entity must be one of the entity names returned by GetEntityNames.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Unknown entry name was specified.

Examples

```
' -------------------------------------------------------
' Scripting environment - VBScript
' Insert the first entity in the list of available entities
' -------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we get the names of all available entities as they
' are shown in the entry helper of XMLSpy
Dim arrEntities
objRange.GetEntityNames arrEntities

' we insert the first one of the list
If UBound(arrEntities) >= 0 Then
	objRange.InsertEntity arrEntities(0)
Else
	MsgBox "Sorry, no entities are available for this document"
End If
```

17.2.2.7.30 InsertRow

Method: InsertRow() as Boolean

Description
If the beginning of the range is inside a dynamic table, this method inserts a new row before the current one. The selection of the range gets modified to point to the beginning of the newly inserted row. The function returns *true* if the insert operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

Examples

```
' ------------------------------------------------------------
' Scripting environment - VBScript
' Insert row at beginning of current dynamically growing table
' ------------------------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' check if we can insert something
If objRange.IsInDynamicTable Then
        objRange.InsertRow
        ' objRange points to beginning of new row
        objRange.Select
  End If
```

17.2.2.7.31 IsCopyEnabled

Property: IsCopyEnabled as Boolean (read-only)

Description
Checks if the copy operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.32 IsCutEnabled

Property: IsCutEnabled as Boolean (read-only)

Description
Checks if the cut operation is supported for this range.

Errors

(C) 2015-2021 Altova GmbH

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.33 IsDeleteEnabled

Property: IsDeleteEnabled as Boolean (read-only)

Description
Checks if the delete operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.34 IsEmpty

Method: IsEmpty() as Boolean

Description
Tests if the first and last position of the range are equal.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.35 IsEqual

Method: IsEqual (*objCmpRange* as AuthenticRange) as Boolean

Description
Tests if the start and end of both ranges are the same.

Errors
2001 One of the two range objects being compared, is invalid.
2005 Invalid address for a return parameter was specified.

17.2.2.7.36 IsFirstRow

Property: IsFirstRow as Boolean (read-only)

Description
Test if the range is in the first row of a table. Which table is taken into consideration depends on the extent of the range. If the selection exceeds a single row of a table, the check is if this table is the first element in an embedding table. See the entry helpers of the user manual for more information.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.37 IsInDynamicTable

Method: IsInDynamicTable() as Boolean

Description
Test if the whole range is inside a table that supports the different row operations like 'insert', 'append', duplicate, etc.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.38 IsLastRow

Property: IsLastRow as Boolean (read-only)

Description
Test if the range is in the last row of a table. Which table is taken into consideration depends on the extent of the range. If the selection exceeds a single row of a table, the check is if this table is the last element in an embedding table. See the entry helpers of the user manual for more information.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.39 IsPasteEnabled

Property: IsPasteEnabled as Boolean (read-only)

Description
Checks if the paste operation is supported for this range.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.40 IsSelected

Property: IsSelected as Boolean

(C) 2015-2021 Altova GmbH

Description
Returns true() if selection is present. The selection range still can be empty: that happens when e.g. only the cursor is set.

17.2.2.7.41 IsTextStateApplied

Method: IsTextStateApplied (*i_strElementName* as String) as Boolean

Description
Checks if all the selected text is embedded into an XML Element with name i_strElementName. Common examples for the parameter i_strElementName are "strong", "bold" or "italic".

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.42 LastTextPosition

Property: LastTextPosition as Long

Description
Set or get the rightmost text position index of the range object. This index is always greater or equal to FirstTextPosition. Indexing starts with 0 at the document beginning, and increments with every different position that the text cursor can occupy. Incrementing the test position by 1, has the same effect as the cursor-right key. Decreasing the test position by 1 has the same effect as the cursor-left key.

If you set LastTextPosition to a value less then the current FirstTextPosition, FirstTextPosition gets set to the new LastTextPosition.

HINT: Use text cursor positions with care, since this is a costly operation compared to XMLData based cursor positioning.

Errors
2001 The authentic range object, or its related view object is not valid.
2005 Invalid address for the return parameter was specified.
2006 A text position outside the document was specified.

Examples
```
' ----------------------------------------
' Scripting environment - VBScript
' ----------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

nDocStartPosition = objAuthenticView.DocumentBegin.FirstTextPosition
nDocEndPosition = objAuthenticView.DocumentEnd.FirstTextPosition

' let's create a range that selects the whole document
```

```
' in an inefficient way
Dim objRange
' we need to get a (any) range object first
Set objRange = objAuthenticView.DocumentBegin
objRange.FirstTextPosition = nDocStartPosition
objRange.LastTextPosition = nDocEndPosition

' let's check if we got it right
If objRange.isEqual(objAuthenticView.WholeDocument) Then
	MsgBox "Test using direct text cursor positioning was ok"
Else
	MsgBox "Oops!"
End If
```

17.2.2.7.43 LastXMLData

Property: LastXMLData as XMLData

Description
Set or get the last XMLData element in the underlying document that is partially or completely selected by the range. The exact end of the selection is defined by the LastXMLDataOffset attribute.

Whenever you set LastXMLData to a new data object, LastXMLDataOffset gets set to the last cursor position inside this element. Only XMLData objects that have a cursor position may be used. If you set LastXMLData / LastXMLDataOffset, select a position less then the current FirstXMLData / FirstXMLDataOffset, the latter gets moved to the new end position.

HINT: You can use the FirstXMLData and LastXMLData properties to directly access and manipulate the underlying XML document in those cases, where the methods available with the AuthenticRange object are not sufficient.

Errors

2001	The authentic range object, or its related view object is not valid.
2005	Invalid address for the return parameter was specified.
2008	Internal error
2009	The XMLData object cannot be accessed.

17.2.2.7.44 LastXMLDataOffset

Property: LastXMLDataOffset as Long

Description
Set or get the cursor position inside LastXMLData element for the end of the range.

Offset positions are based on the characters returned by the Text property and start with 0. When setting a new offset, use -1 to set the offset to the last possible position in the element. The following cases require specific attention:

- The textual form of entries in Combo Boxes, Check Boxes and similar controls can be different from what you see on the screen. Although, the data offset is based on this text, there only two valid offset positions,

one at the beginning and one at the end of the entry. An attempt to set the offset to somewhere in the middle of the entry, will result in the offset being set to the end.
- The textual form of XML Entities might differ in length from their representation on the screen. The offset is based on this textual form.

If LastXMLData / LastXMLDataOffset selects a position before FirstXMLData / FirstXMLDataOffset, the latter gets moved to the new end position.

Errors

2001	The authentic range object, or its related view object is not valid.
2005	Invalid offset was specified.
	Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Select the complete text of an XMLData element
' using XMLData based selection and ExpandTo
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' first we use the XMLData based range properties
' to select all text of the first XMLData element
' in the current selection
Dim objRange
Set objRange = objAuthenticView.Selection
objRange.FirstXMLDataOffset = 0  ' start at beginning of element text
objRange.LastXMLData = objRange.FirstXMLData  ' select only one element
objRange.LastXMLDataOffset = -1  ' select till its end

' the same can be achieved with the ExpandTo method
Dim objRange2
Set objRange2 = objAuthenticView.Selection.ExpandTo(spyAuthenticTag)

' were we successful?
If objRange.IsEqual(objRange2) Then
	objRange.Select()
Else
	MsgBox "Ooops"
End If
```

17.2.2.7.45 MoveBegin

Method: MoveBegin (*eKind* as SPYAuthenticElementKind, *nCount* as Long) as AuthenticRange

Description

Move the beginning of the range to the beginning of the nCount element of type eKind. Counting starts at the current beginning of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The end of the range stays unmoved, unless the new beginning would be larger than it. In this case, the end is moved to the new beginning. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before beginning of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

17.2.2.7.46 MoveEnd

Method: MoveEnd (*eKind* as SPYAuthenticElementKind, *nCount* as Long) as AuthenticRange

Description

Move the end of the range to the begin of the nCount element of type eKind. Counting starts at the current end of the range object.

Use positive numbers for nCount to move towards the document end, use negative numbers to move towards document beginning. The beginning of the range stays unmoved, unless the new end would be less than it. In this case, the beginning gets moved to the new end. The method returns the modified range object.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2003 Target lies after end of document.
2004 Target lies before begin of document.
2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

17.2.2.7.47 MoveRowDown

Method: MoveRowDown() as Boolean

Description

If the beginning of the range is inside a dynamic table and selects a row which is not the last row in this table, this method swaps this row with the row immediately below. The selection of the range moves with the row, but does not otherwise change. The function returns *true* if the move operation was successful, otherwise *false*.

Errors

2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.48 MoveRowUp

Method: MoveRowUp() as Boolean

(C) 2015-2021 Altova GmbH

Description
If the beginning of the range is inside a dynamic table and selects a row which is not the first row in this table, this method swaps this row with the row above. The selection of the range moves with the row, but does not change otherwise. The function returns *true* if the move operation was successful, otherwise *false*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.49 Parent

Property: Parent as AuthenticView (read-only)

Description
Access the view that owns this range object.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.50 Paste

Method: Paste() as Boolean

Description
Returns *False* if the copy/paste buffer is empty, or its content cannot replace the current selection.

Otherwise, deletes the current selection, inserts the content of the copy/paste buffer, and returns *True*.

Errors
2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.7.51 PerformAction

Method: PerformAction (*eAction* as SPYAuthenticActions, *strElementName* as String) as Boolean

Description
PerformAction and its related methods, give access to the entry-helper functions of Authentic. This function allows easy and consistent modification of the document content without a need to know exactly where the modification will take place. The beginning of the range object is used to locate the next valid location where the specified action can be performed. If no such location can be found, the method returns *False*. Otherwise, the document gets modified and the range points to the beginning of the modification.

HINT: To find out element names that can be passed as the second parameter use CanPerformActionWith.

Errors

2001 The authentic range object, or its related view object is no longer valid.
2005 Invalid address for the return parameter was specified.
2007 Invalid action was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Insert the innermost element
' ---------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' we determine the elements that can be inserted at the current position
Dim arrElements()
objRange.CanPerformActionWith spyAuthenticInsertBefore, arrElements

' we insert the first (innermost) element
If UBound(arrElements) >= 0 Then
	objRange.PerformAction spyAuthenticInsertBefore, arrElements(0)
	' objRange now points to the beginning of the inserted element
	' we set a default value and position at its end
	objRange.Text = "Hello"
	objRange.ExpandTo(spyAuthenticTag).CollapsToEnd().Select
Else
	MsgBox "Can't insert any elements at current position"
End If
```

17.2.2.7.52 Select

Method: Select()

Description

Makes this range the current user interface selection. You can achieve the same result using:
'*objRange.Parent.Selection = objRange*'

Errors

2001 The authentic range object or its related view object is no longer valid.

Examples

```
' ----------------------------------------
' Scripting environment - VBScript
' ----------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' set current selection to end of document
objAuthenticView.DocumentEnd.Select()
```


(C) 2015-2021 Altova GmbH

17.2.2.7.53 SelectNext

Method: SelectNext (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description
Selects the element of type eKind after the current end of the range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2003	Target lies after end of document.
2005	Invalid element kind specified.
	Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document word-by-word
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentBegin
Dim bEndOfDocument
bEndOfDocument = False

On Error Resume Next
While Not bEndOfDocument
	objRange.SelectNext(spyAuthenticWord).Select
	If ((Err.number - vbObjecterror) = 2003) Then
		bEndOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise  ' forward error
	End If
Wend
```

17.2.2.7.54 SelectPrevious

Method: GotoPrevious (*eKind* as SPYAuthenticElementKind) as AuthenticRange

Description
Selects the element of type eKind before the current beginning of the range. The method returns the modified range object.

Errors

2001	The authentic range object, or its related view object is no longer valid.
2004	Target lies before begin of document.

2005 Invalid element kind specified.
Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Scan through the whole document tag-by-tag
' ---------------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

Dim objRange
Set objRange = objAuthenticView.DocumentEnd
Dim bBeginOfDocument
bBeginOfDocument = False

On Error Resume Next
While Not bBeginOfDocument
	objRange.SelectPrevious(spyAuthenticTag).Select
	If ((Err.number - vbObjecterror) = 2004) Then
		bBeginOfDocument = True
		Err.Clear
	ElseIf (Err.number <> 0) Then
		Err.Raise  ' forward error
	End If
Wend
```

17.2.2.7.55 SetElementAttributeValue

Method: SetElementAttributeValue (*strElementName* as String, *strAttributeName* as String, *strAttributeValue* as String)

Description

Set the value of the attribute specified in strAttributeName for the element identified with strElementName. If the attribute is supported but has no value assigned, the empty string is returned. To find out the names of attributes supported by an element, use GetElementAttributeNames, or HasElementAttribute.

Errors

2001 The authentic range object or its related view object is no longer valid.
2005 Invalid element name was specified.
Invalid attribute name was specified.
Invalid attribute value was specified.

Examples

```
' ---------------------------------------------
' Scripting environment - VBScript
' Get and set element attributes
' ---------------------------------------------
Dim objRange
' we assume that the active document is open in authentic view mode
```

```
Set objRange = Application.ActiveDocument.AuthenticView.Selection

' first we find out all the elements below the beginning of the range
Dim arrElements
objRange.GetElementHierarchy arrElements

If IsArray(arrElements) Then
	If UBound(arrElements) >= 0 Then
		' we use the top level element and find out its valid attributes
		Dim arrAttrs()
		objRange.GetElementAttributeNames arrElements(0), arrAttrs

		If UBound(arrAttrs) >= 0 Then
			' we retrieve the current value of the first valid attribute
			Dim strAttrVal
			strAttrVal = objRange.GetElementAttributeValue (arrElements(0), arrAttrs(0))
			msgbox "current value of " & arrElements(0) & "//" & arrAttrs(0) & " is: " & strAttrVal

			' we change this value and read it again
			strAttrVal = "Hello"
			objRange.SetElementAttributeValue arrElements(0), arrAttrs(0), strAttrVal
			strAttrVal = objRange.GetElementAttributeValue (arrElements(0), arrAttrs(0))
			msgbox "new value of " & arrElements(0) & "//" & arrAttrs(0) & " is: " & strAttrVal
		End If
	End If
End If
```

17.2.2.7.56 SetFromRange

Method: SetFromRange (*objSrcRange* as AuthenticRange)

Description
Sets the range object to the same beginning and end positions as objSrcRange.

Errors
2001 One of the two range objects, is invalid.
2005 Null object was specified as source object.

17.2.2.7.57 SetVariableValue

Method: SetVariableValue`(strName as string, strValue as string)`

Return Value
Sets the value (second parameter) of the variable named in the first parameter.

Errors
2201 Invalid object.
2202 No context.

2204 No such variable in scope
2205 Variable cannot be evaluated
2206 Variable returns sequence
2207 Variable read-only
2208 No modification allowed

17.2.2.7.58 Text

Property: Text as String

Description
Set or get the textual content selected by the range object.

The number of characters retrieved are not necessarily identical, as there are text cursor positions between the beginning and end of the selected range. Most document elements support an end cursor position different to the beginning cursor position of the following element. Drop-down lists maintain only one cursor position, but can select strings of any length. In the case of radio buttons and check boxes, the text property value holds the string of the corresponding XML element.

If the range selects more then one element, the text is the concatenation of the single texts. XML entities are expanded so that '&' is expected as '&'.

Setting the text to the empty string, does not delete any XML elements. Use Cut, Delete or PerformAction instead.

Errors
2001 The authentic range object or its related view object is no longer valid.
2005 Invalid address for a return parameter was specified.

17.2.2.8 AuthenticView

Properties	Methods	Events
Application	Goto	OnBeforeCopy
AsXMLString	IsRedoEnabled	OnBeforeCut
DocumentBegin	IsUndoEnabled	OnBeforeDelete
DocumentEnd	Print	OnBeforeDrop
Event	Redo	OnBeforePaste
MarkupVisibility	Undo	
Parent	UpdateXMLInstanceEntities	
Selection		OnMouseEvent
XMLDataRoot		OnSelectionChanged
WholeDocument		

Description
AuthenticView and its child objects AuthenticRange and AuthenticDataTransfer provide you with an interface for Authentic View, which allow easy and consistent modification of

document contents. These interfaces replace the following interfaces which are marked now as **obsolete:**

OldAuthenticView (old name was DocEditView)
AuthenticSelection (old name was DocEditSelection, superseded by AuthenticRange)
AuthenticEvent (old name was DocEditEvent)

AuthenticView gives you easy access to specific features such as printing, the multi-level undo buffer, and the current cursor selection, or position.

AuthenticView uses objects of type AuthenticRange to make navigation inside the document straight-forward, and to allow for the flexible selection of logical text elements. Use the properties DocumentBegin, DocumentEnd, or WholeDocument for simple selections, while using the Goto method for more complex selections. To navigate relative to a given document range, see the methods and properties of the AuthenticRange object.

17.2.2.8.1 Events

17.2.2.8.1.1 OnBeforeCopy

Event: OnBeforeCopy() as Boolean

Scripting environment - VBScript:
```
Function On_AuthenticBeforeCopy()
	' On_AuthenticBeforeCopy = False  ' to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticBeforeCopy()
{
	// return false;  /* to disable operation */
}
```

17.2.2.8.1.2 OnBeforeCut

Event: OnBeforeCut() as Boolean

Scripting environment - VBScript:
```
Function On_AuthenticBeforeCut()
	' On_AuthenticBeforeCut = False  ' to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticBeforeCut()
{
	// return false;  /* to disable operation */
}
```

17.2.2.8.1.3 OnBeforeDelete

Event: OnBeforeDelete() as Boolean

Scripting environment - VBScript:
```
Function On_AuthenticBeforeDelete()
	' On_AuthenticBeforeDelete = False  ' to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticBeforeDelete()
{
	// return false;  /* to disable operation */
}
```

17.2.2.8.1.4 OnBeforeDrop

Event: OnBeforeDrop (*i_nXPos* as Long, *i_nYPos* as Long, *i_ipRange* as AuthenticRange, *i_ipData* as cancelBoolean

Scripting environment - VBScript:
```
Function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
	' On_AuthenticBeforeDrop = False  ' to disable operation
End Function
```

Scripting environment - JScript:
```
function On_AuthenticBeforeDrop(nXPos, nYPos, objRange, objData)
{
	// return false;  /* to disable operation */
}
```

Description
This event gets triggered whenever a previously dragged object gets dropped inside the application window. All event related information gets passed as parameters.

The first two parameters specify the mouse position at the time when the event occurred. The parameter *objRange* passes a range object that selects the XML element below the mouse position. The value of this parameter might be *NULL*. Be sure to check before you access the range object. The parameter *objData* allows to access information about the object being dragged.

Return *False* to cancel the drop operation. Return *True* (or nothing) to continue normal operation.

17.2.2.8.1.5 OnBeforePaste

Event: OnBeforePaste (*objData* as Variant, *strType* as String) as Boolean

Scripting environment - VBScript:
```
Function On_AuthenticBeforePaste(objData, strType)
	' On_AuthenticBeforePaste = False  ' to disable operation
End Function
```

 (C) 2015-2021 Altova GmbH

Scripting environment - JScript:

```
function On_AuthenticBeforePaste(objData, strType)
{
        // return false;  /* to disable operation */
}
```

Description

This event gets triggered before a paste operation gets performed on the document. The parameter *strType* is one of "TEXT", "UNICODETEXT" or "IUNKNOWN". In the first two cases *objData* contains a string representation of the object that will be pasted. In the later case, *objData* contains a pointer to an IUnknown COM interface.

Return *True* (or nothing) to allow paste operation. Return *False* to disable operation.

17.2.2.8.1.6 OnBeforeSave

Event: OnBeforeSave (SaveAs flag) as Boolean

Description: `OnBeforeSave` gives the opportunity to e.g. warn the user about overwriting the existing XML document, or to make the document read-only when specific circumstances are not met. The event will be fired before the file dialog is shown. (Please note, that the event fires when saving the XML document, and not when saving the SPS design in StyleVision.)

17.2.2.8.1.7 OnLoad

Event: OnLoad ()

Description: `OnLoad` can be used e.g. to restrict some AuthenticView functionality, as shown in the example below:

```
function On_AuthenticLoad(  )
{
    // We are disabling all entry helpers in order to prevent user from manipulating XML tree
    AuthenticView.DisableElementEntryHelper();
    AuthenticView.DisableAttributeEntryHelper();

    // We are also disabling the markup buttons for the same purpose
    AuthenticView.SetToolbarButtonState( 'AuthenticMarkupSmall', authenticToolbarButtonDisabled );
    AuthenticView.SetToolbarButtonState( 'AuthenticMarkupLarge', authenticToolbarButtonDisabled );
    AuthenticView.SetToolbarButtonState( 'AuthenticMarkupMixed', authenticToolbarButtonDisabled );
}
```

In the example the status of the Markup Small, Markup Large, Markup Mixed toolbar buttons are manipulated with the help of button identifiers. See complete list.

17.2.2.8.1.8 OnMouseEvent

Event: OnMouseEvent (*nXPos* as Long, *nYPos* as Long, *eMouseEvent* as SPYMouseEvent, *objRange* as AuthenticRange) as Boolean

Scripting environment - VBScript:

```
Function On_AuthenticMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
        ' On_AuthenticMouseEvent = True  ' to cancel bubbling of event
End Function
```

Scripting environment - JScript:

```
function On_AuthenticMouseEvent(nXPos, nYPos, eMouseEvent, objRange)
{
        // return true;  /* to cancel bubbling of event */
}
```

Description
This event gets triggered for every mouse movement and mouse button Windows message.

The actual message type and the mouse buttons status, is available in the *eMouseEvent* parameter. Use the bit-masks defined in the enumeration datatype SPYMouseEvent to test for the different messages, button status, and their combinations.

The parameter *objRange* identifies the part of the document found at the current mouse cursor position. The range object always selects a complete tag of the document. (This might change in future versions, when a more precise positioning mechanism becomes available). If no selectable part of the document is found at the current position, the range object is *null*.

17.2.2.8.1.9 OnSelectionChanged

Event: OnSelectionChanged (*objNewSelection* as AuthenticRange)

Scripting environment - VBScript:

```
Function On_AuthenticSelectionChanged (objNewSelection)
End Function
```

Scripting environment - JScript:

```
function On_AuthenticSelectionChanged (objNewSelection)
{
}
```

Description
This event gets triggered whenever the selection in the user interface changes.

17.2.2.8.1.10 OnToolbarButtonClicked

Event: OnToolbarButtonClicked (Button identifier)

Description: `OnToolbarButtonClicked` is fired when a toolbar button was clicked by user. The parameter button identifier helps to determine which button was clicked. The list of predefined button identifiers is below:

- AuthenticPrint
- AuthenticPrintPreview
- AuthenticUndo
- AuthenticRedo
- AuthenticCut
- AuthenticCopy
- AuthenticPaste
- AuthenticClear
- AuthenticMarkupHide
- AuthenticMarkupLarge
- AuthenticMarkupMixed
- AuthenticMarkupSmall
- AuthenticValidate
- AuthenticChangeWorkingDBXMLCell
- AuthenticSave
- AuthenticSaveAs
- AuthenticReload
- AuthenticTableInsertRow
- AuthenticTableAppendRow
- AuthenticTableDeleteRow
- AuthenticTableInsertCol
- AuthenticTableAppendCol
- AuthenticTableDeleteCol
- AuthenticTableJoinCellRight
- AuthenticTableJoinCellLeft
- AuthenticTableJoinCellAbove
- AuthenticTableJoinCellBelow
- AuthenticTableSplitCellHorizontally
- AuthenticTableSplitCellVertically
- AuthenticTableAlignCellContentTop
- AuthenticTableCenterCellVertically
- AuthenticTableAlignCellContentBottom
- AuthenticTableAlignCellContentLeft
- AuthenticTableCenterCellContent
- AuthenticTableAlignCellContentRight
- AuthenticTableJustifyCellContent
- AuthenticTableInsertTable
- AuthenticTableDeleteTable
- AuthenticTableProperties
- AuthenticAppendRow
- AuthenticInsertRow
- AuthenticDuplicateRow
- AuthenticMoveRowUp
- AuthenticMoveRowDown
- AuthenticDeleteRow
- AuthenticDefineEntities
- AuthenticXMLSignature

For custom buttons the user might add his own identifiers. Please, note that the user must take care, as the identifiers are not checked for uniqueness. The same identifiers can be used to identify buttons in the `Set/GetToolbarState()` COM API calls. By adding code for different buttons, the user is in the position to completely redefine the AuthenticView toolbar behavior, adding own methods for table manipulation, etc.

17.2.2.8.1.11 OnToolbarButtonExecuted

Event: OnToolbarButtonExecuted (Button identifier)

Description: `OnToolbarButtonClicked` is fired when a toolbar button was clicked by user. The parameter button identifier helps to determine which button was clicked. See the list of predefined button identifiers.

`OnToolbarButtonExecuted` is fired after the toolbar action was executed. It is useful e.g. to add update code, as shown in the example below:

```
//event fired when a toolbar button action was executed
function On_AuthenticToolbarButtonExecuted( varBtnIdentifier )
{
	// After whatever command user has executed - make sure to update toolbar button states
	UpdateOwnToolbarButtonStates();
}
```

In this case `UpdateOwnToolbarButtonStates` is a user function defined in the Global Declarations.

17.2.2.8.1.12 OnUserAddedXMLNode

Event: OnUserAddedXMLNode (XML node)

Description: `OnUserAddedXMLNode` will be fired when the user adds an XML node as a primary action. This happens in the situations, where the user clicks on

- auto-add hyperlinks (see example OnUserAddedXMLNode.sps)
- the Insert…, Insert After…, Insert Before… context menu items
- Append row, Insert row toolbar buttons
- Insert After…, Insert Before… actions in element entry helper (outside StyleVision)

The event doesn't get fired on Duplicate row, or when the node was added externally (e.g. via COM API), or on Apply (e.g. Text State Icons), or when in XML table operations or in DB operations.

The event parameter is the XML node object, which was added giving the user an opportunity to manipulate the XML node added. An elaborate example for an event handler can be found in the `OnUserAddedXMLNode.sps` file.

17.2.2.8.2 Application

Property: Application as Application (read-only)

Description
Accesses the StyleVision application object.

 (C) 2015-2021 Altova GmbH

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.8.3 AsXMLString

Property: AsXMLString as String

Description
Returns or sets the document content as an XML string. Setting the content to a new value does not change the schema file or sps file in use. If the new XMLString does not match the actual schema file error 2011 gets returned.

Errors
2000 The authentic view object is no longer valid.
2011 AsXMLString was set to a value which is no valid XML for the current schema file.

17.2.2.8.4 ContextMenu

Property: ContextMenu() as ContextMenu

Description
The property `ContextMenu` gives access to customize the context menu. The best place to do it is in the event handler `OnContextMenuActivated`.

Errors
2000 Invalid object.
2005 Invalid parameter.

17.2.2.8.5 CreateXMLNode

Method: CreateXMLNode (*nKind* as SPYXMLDataKind) as XMLData

Return Value
The method returns the new XMLData object.

Description
To create a new XMLData object use the CreateXMLNode() method.

Errors
2000 Invalid object.
2012 Cannot create XML node.

17.2.2.8.6 DisableAttributeEntryHelper

Method: DisableAttributeEntryHelper ()

Description
DisableAttributeEntryHelper () disables the attribute entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

17.2.2.8.7 DisableElementEntryHelper

Method: DisableElementEntryHelper ()

Description
DisableElementEntryHelper () disables the element entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

17.2.2.8.8 DisableEntityEntryHelper

Method: DisableEntityEntryHelper ()

Description
DisableEntityEntryHelper () disables the entity entry helper in XMLSpy, Authentic Desktop and Authentic Browser plug-in.

Errors
2000 Invalid object.

17.2.2.8.9 DocumentBegin

Property: DocumentBegin as AuthenticRange (read-only)

Description
Retrieve a range object that points to the beginning of the document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

17.2.2.8.10 DocumentEnd

Property: DocumentEnd as AuthenticRange (read-only)

Description
Retrieve a range object that points to the end of the document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.8.11 DoNotPerformStandardAction

Method: DoNotPerformStandardAction ()

Description
`DoNotPerformStandardAction()` serves as cancel bubble for macros, and stops further execution after macro has finished.

Errors
2000 Invalid object.

17.2.2.8.12 EvaluateXPath

Method: EvaluateXPath (XMLData as XMLData, strExpression as string) strValue as string

Return Value
The method returns a string

Description
`EvaluateXPath()` executes an XPath expression with the given XML context node. The result is returned as a string, in the case of a sequence it is a space-separated string.

Errors
2000 Invalid object.
2005 Invalid parameter.
2008 Internal error.
2013 XPath error.

17.2.2.8.13 Event

Property: Event as AuthenticEvent (read-only)

Description

This property gives access to parameters of the last event in the same way as OldAuthenticView.event does. Since all events for the scripting environment and external clients are now available with parameters this Event property should only be used from within IDE-Plugins.

Errors

2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.8.14 EventContext

Property: EventContext() as EventContext

Description

`EventContext` property gives access to the running macros context. See the `EventContext` interface description for more details.

Errors

2000 Invalid object.

17.2.2.8.15 GetToolbarButtonState

Method: GetToolbarButtonState (`ButtonIdentifier` as string) as `AuthenticToolbarButtonState`

Return Value

The method returns `AuthenticToolbarButtonState`

Description

`Get/SetToolbarButtonState` queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above). One usage is to disable toolbar buttons permanently. Another usage is to put `SetToolbarButtonState` in the `OnSelectionChanged` event handler, as toolbar buttons are updated regularly when the selection changes in the document.

Toolbar button states are given by the listed enumerations.

The default state means that the enable/disable of the button is governed by AuthenticView. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

Errors

2000 Invalid object.
2005 Invalid parameter.
2008 Internal error.
2014 Invalid button identifier.

(C) 2015-2021 Altova GmbH

17.2.2.8.16 Goto

Method: Goto (*eKind* as SPYAuthenticElementKind, *nCount* as Long, *eFrom* as *SPYAuthenticDocumentPosition*) as *AuthenticRange*

Description
Retrieve a range object that points to the beginning of the *nCount* element of type *eKind*. The start position is defined by the parameter *eFrom*. Use positive values for *nCount* to navigate to the document end. Use negative values to navigate towards the beginning of the document.

Errors

2000	The authentic view object is no longer valid.
2003	Target lies after end of document.
2004	Target lies before beginning of document.
2005	Invalid element kind specified. The document position to start from is not one of *spyAuthenticDocumentBegin* or *spyAuthenticDocumentEnd*. Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------
' Scripting environment - VBScript
' ----------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

On Error Resume Next
Dim objRange
' goto beginning of first table in document
Set objRange = objAuthenticView.Goto (spyAuthenticTable, 1, spyAuthenticDocumentBegin)
If (Err.number = 0) Then
        objRange.Select()
Else
        MsgBox "No table found in document"
End If
```

17.2.2.8.17 IsRedoEnabled

Property: IsRedoEnabled as Boolean (read-only)

Description
True if redo steps are available and Redo is possible.

Errors

2000	The authentic view object is no longer valid.
2005	Invalid address for the return parameter was specified.

17.2.2.8.18 IsUndoEnabled

Property: IsUndoEnabled as Boolean (read-only)

Description
True if undo steps are available and Undo is possible.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.8.19 MarkupVisibility

Property: MarkupVisibility as SPYAuthenticMarkupVisibility

Description
Set or get current visibility of markup.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid enumeration value was specified.
Invalid address for the return parameter was specified.

17.2.2.8.20 Parent

Property: Parent as Document (read-only)

Description
Access the document shown in this view.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.8.21 Print

Method: Print (*bWithPreview* as Boolean, *bPromptUser* as Boolean)

Description
Print the document shown in this view. If *bWithPreview* is set to *True*, the print preview dialog pops up. If *bPromptUser* is set to *True*, the print dialog pops up. If both parameters are set to *False*, the document gets printed without further user interaction.

(C) 2015-2021 Altova GmbH

Errors
2000 The authentic view object is no longer valid.

17.2.2.8.22 Redo

Method: Redo() as Boolean

Description
Redo the modification undone by the last undo command.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.8.23 Selection

Property: Selection as AuthenticRange

Description
Set or get current text selection in user interface.

Errors
2000 The authentic view object is no longer valid.
2002 No cursor selection is active.
2005 Invalid address for the return parameter was specified.

Examples

```
' ----------------------------------------
' Scripting environment - VBScript
' ----------------------------------------
Dim objAuthenticView
' we assume that the active document is open in authentic view mode
Set objAuthenticView = Application.ActiveDocument.AuthenticView

' if we are the end of the document, re-start at the beginning
If (objAuthenticView.Selection.IsEqual(objAuthenticView.DocumentEnd)) Then
        objAuthenticView.Selection = objAuthenticView.DocumentBegin
Else
        ' objAuthenticView.Selection = objAuthenticView.Selection.GotoNextCursorPosition()
        ' or shorter:
         objAuthenticView.Selection.GotoNextCursorPosition().Select
End If
```

17.2.2.8.24 SetToolbarButtonState

Method: SetToolbarButtonState (`ButtonIdentifier` as string, `AuthenticToolbarButtonState` state)

Description
`Get/SetToolbarButtonState` queries the status of a toolbar button, and lets the user disable or enable the button, identified via its button identifier (see list above). One usage is to disable toolbar buttons permanently. Another usage is to put `SetToolbarButtonState` in the `OnSelectionChanged` event handler, as toolbar buttons are updated regularly when the selection changes in the document.

Toolbar button states are given by the listed enumerations.

The default state means that the enable/disable of the button is governed by AuthenticView. When the user sets the button state to enable or disable, the button remains in that state as long as the user does not change it.

Errors
2000 Invalid object.
2008 Internal error.
2014 Invalid button identifier.

17.2.2.8.25 Undo

Method: Undo() as Boolean

Description
Undo the last modification of the document from within this view.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.8.26 UpdateXMLInstanceEntities

Method: UpdateXMLInstanceEntities()

Description
Updates the internal representation of the declared entities, and refills the entry helper. In addition, the validator is reloaded, allowing the XML file to validate correctly. Please note that this may also cause schema files to be reloaded.

Errors
The method never returns an error.

Example
```
// ----------------------------------------
// Scripting environment - JavaScript
// ----------------------------------------
```

 (C) 2015-2021 Altova GmbH

```
if(Application.ActiveDocument && (Application.ActiveDocument.CurrentViewMode == 4))
{
	var objDocType;
	objDocType = Application.ActiveDocument.DocEditView.XMLRoot.GetFirstChild(10);

	if(objDocType)
	{
		var objEntity = Application.ActiveDocument.CreateChild(14);
		objEntity.Name = "child";
		objEntity.TextValue = "SYSTEM \"child.xml\"";
		objDocType.AppendChild(objEntity);

		Application.ActiveDocument.AuthenticView.UpdateXMLInstanceEntities();
	}
}
```

17.2.2.8.27 WholeDocument

Property: WholeDocument as AuthenticRange (read-only)

Description
Retrieve a range object that selects the whole document.

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.8.28 XMLDataRoot

Property: XMLDataRoot as XMLData (read-only)

Description
Returns or sets the top-level XMLData element of the current document. This element typically describes the document structure and would be of kind spyXMLDataXMLDocStruct, spyXMLDataXMLEntityDocStruct or spyXMLDataDTDDocStruct..

Errors
2000 The authentic view object is no longer valid.
2005 Invalid address for the return parameter was specified.

17.2.2.9 Document

The `Document` interface has the following methods and properties:

Methods

- Activate
- Application

- AssignWorkingXMLFile
- Close
- Save
- SaveAs
- Saved
- SaveGeneratedFOFile
- SaveGeneratedFOFileEx
- SaveGeneratedHTMLFile
- SaveGeneratedHTMLFileEx
- SaveGeneratedPDFFile
- SaveGeneratedPDFFileEx
- SaveGeneratedRTFFile
- SaveGeneratedRTFFileEx
- SaveGeneratedWord2007File
- SaveGeneratedWord2007FileEx
- SaveGeneratedXSLTFOFile
- SaveGeneratedXSLTFOFileEx
- SaveGeneratedXSLTHTMLFile
- SaveGeneratedXSLTHTMLFileEx
- SaveGeneratedXSLTRTFFile
- SaveGeneratedXSLTRTFFileEx
- SaveGeneratedXSLTWord2007File
- SaveGeneratedXSLTWord2007FileEx

Properties

- Application
- FullName
- Name
- Parameters
- Parent
- Path
- SchemaSources

17.2.2.9.1 Events

17.2.2.9.1.1 OnDocumentClosed

Event: OnDocumentClosed(*Document* as Document)

Description
This event gets fired as a result of closing a document.

(C) 2015-2021 Altova GmbH

17.2.2.9.1.2 OnModifiedFlagChanged

Event: OnModifiedFlagChanged(as Boolean)

Description
Returns true if the Modified flag has been changed.

17.2.2.9.2 Activate

Method: Activate ()

Description
Activate document frame.

Errors
1200 Document object is invalid.
1201 Invalid input parameter.

17.2.2.9.3 Application

Property: Application as Application (read-only)

Description
Accesses the StyleVision application object.

Errors
140011 The object is no longer valid.
11
140711 Invalid address for the return parameter was specified.
00

17.2.2.9.4 AssignWorkingXMLFile

Method: AssignWorkingXMLFile (*strWorkingXMLFileName* as String, *strSchemaSourceName* as String)

Description
Assigns the Working XML File by supplying its URI as a string and the Schema Source Name as a string. The Schema Source Name is the name assigned to the schema in the SPS; (the default schema name in an SPS created new in StyleVision is: `XML`).

Errors
1200 The document object is invalid.
1201 Invalid input parameter.
1203 Error assigning Working XML File
1404 Missing XML Schema or DTD

17.2.2.9.5 Close

Method: Close (*bDiscardChanges* as Boolean)

Description
To close the document call this method. If bDiscardChanges is true and the document is modified, the document will be closed but not saved.

Errors
1400 The object is no longer valid.
1401 Document needs to be saved first.

17.2.2.9.6 FullName

Property: FullName as String

Description
This property can be used to get or set the full file name - including the path - to where the document gets saved. The validity of the name is not verified before the next save operation.

This property makes the methods GetPathName and SetPathName obsolete.

Errors
1400 The document object is no longer valid.
1402 Empty string has been specified as full file name.

17.2.2.9.7 GetPathName (obsolete)

Superseded by Document.FullName

```
// ----- javascript sample -----
// instead of:
// strPathName = Application.ActiveDocument.GetPathName();
// use now:
strPathName = Application.ActiveDocument.FullName;
```

Method: GetPathName() as String

Description
The method GetPathName gets the path of the active document.

See also Document.SetPathName (obsolete).

(C) 2015-2021 Altova GmbH

17.2.2.9.8 Name

Property: Name as String (read-only)

Description
Use this property to retrieve the name - not including the path - of the document file. To change the file name for a document use the property FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

17.2.2.9.9 Parameters

Property: Parameters as Parameters (read-only)

Description
Reference to the current `Parameters` object.

Errors
1200 Document object is invalid.
1201 Invalid input parameter.

17.2.2.9.10 Parent

Property: Parent as Documents (read-only)

Description
Access the parent of the document object.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

Property: Parent as Application (read-only)

Description
Access the StyleVision application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.9.11 Path

Property: Path as String (read-only)

Description
Use this property to retrieve the path - not including the file name - of the document file. To change the file name and path for a document use the property FullName.

Errors
1400 The document object is no longer valid.
1407 Invalid address for the return parameter was specified.

17.2.2.9.12 Save

Method: Save()

Description
The method writes any modifications of the document to the associated file.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1403 Error when saving file, probably the file name is invalid.

17.2.2.9.13 SaveAs

Method: SaveAs (*strFileName* as String)

Description
Save the document to the file specified.

Errors
1400 The document object is no longer valid.
1407 An empty file name has been specified.
1403 Error when saving file, probably the file name is invalid.

17.2.2.9.14 Saved

Property: Saved as Boolean (read-only)

Description
This property can be used to check if the document has been saved after the last modifications.

Errors
1400 The document object is no longer valid.

(C) 2015-2021 Altova GmbH

1407 Invalid address for the return parameter was specified.

17.2.2.9.15 SaveGeneratedFOFile

Method: SaveGeneratedFOFile (*strFileName* as String)

Description
Saves the generated file to the location specified.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.16 SaveGeneratedFOFileEx

Method: SaveGeneratedFOFileEx (*strFileName* as String, *pbError* as Variant) as AppOutputLines

Description
Saves the generated file to the location specified and gives error description.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.17 SaveGeneratedHTMLFile

Method: SaveGeneratedHTMLFile (*strFileName* as String)

Description
Saves the generated file to the location specified.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.18 SaveGeneratedHTMLFileEx

Method: SaveGeneratedHTMLFileEx (*strFileName* as String, *pbError* as Variant) as AppOutputLines

Description
Saves the generated file to the location specified and gives error description.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.19 SaveGeneratedPDFFile

Method: SaveGeneratedPDFFile (*strFileName* as String)

Description
Saves the generated file to the location specified.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.20 SaveGeneratedPDFFileEx

Method: SaveGeneratedPDFFileEx (*strFileName* as String, *pbError* as Variant) as AppOutputLines

Description
Saves the generated file to the location specified and gives error description.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.21 SaveGeneratedRTFFile

Method: SaveGeneratedRTFFile (*strFileName* as String)

Description
Saves the generated file to the location specified.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.22 SaveGeneratedRTFFileEx

Method: SaveGeneratedRTFFileEx (*strFileName* as String, *pbError* as Variant) as AppOutputLines

(C) 2015-2021 Altova GmbH

Description
Saves the generated file to the location specified and gives error description.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.23 SaveGeneratedWord2007File

Method: SaveGeneratedWord2007File (*strFileName* as String)

Description
Saves the generated file to the location specified.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.24 SaveGeneratedWord2007FileEx

Method: SaveGeneratedWord2007FileEx (*strFileName* as String, *pbError* as Variant) as AppOutputLines

Description
Saves the generated file to the location specified and gives error description.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.25 SaveGeneratedXSLTFOFile

Method: SaveGeneratedXSLTFOFile (*strFileName* as String)

Description
Saves the generated file to the location specified.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.26 SaveGeneratedXSLTFOFileEx

Method: SaveGeneratedXSLTFOFileEx (*strFileName* as String, *pbError* as Variant) as AppOutputLines

Description
Saves the generated file to the location specified and gives error description.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.27 SaveGeneratedXSLTHTMLFile

Method: SaveGeneratedXSLTHTMLFile (*strFileName* as String)

Description
Saves the generated file to the location specified.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.28 SaveGeneratedXSLTHTMLFileEx

Method: SaveGeneratedXSLTHTMLFileEx (*strFileName* as String, *pbError* as Variant) as AppOutputLines

Description
Saves the generated file to the location specified and gives error description.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.29 SaveGeneratedXSLTRTFFile

Method: SaveGeneratedXSLTRTFFile (*strFileName* as String)

Description
Saves the generated file to the location specified.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

(C) 2015-2021 Altova GmbH

17.2.2.9.30 SaveGeneratedXSLTRTFFileEx

Method: SaveGeneratedXSLTRTFFileEx (*strFileName* as String, *pbError* as Variant) as AppOutputLines

Description
Saves the generated file to the location specified and gives error description.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.31 SaveGeneratedXSLTWord2007File

Method: SaveGeneratedXSLTWord2007File (*strFileName* as String)

Description
Saves the generated file to the location specified.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.32 SaveGeneratedXSLTWord2007FileEx

Method: SaveGeneratedXSLTWord2007FileEx (*strFileName* as String, *pbError* as Variant) as AppOutputLines

Description
Saves the generated file to the location specified and gives error description.

Errors
1201 Invalid document object.
1204 Cannot generate output file.

17.2.2.9.33 SchemaSources

Property: SchemaSources as SchemaSources (read-only)

Description
Reference to the current `SchemaSources` object.

Errors

1200 Document object is invalid.
1201 Invalid input parameter.

17.2.2.9.34 SetPathName (obsolete)

Superseded by Document.FullName

```
// ----- javascript sample -----
// instead of:
// Application.ActiveDocument.SetPathName("C:\\myXMLFiles\\test.xml");
// use now:
Application.ActiveDocument.FullName = "C:\\myXMLFiles\\test.xml";
```

Method: SetPathName (*strPath* as String)

Description
The method SetPathName sets the path of the active document. SetPathName only copies the string and does not check if the path is valid. All succeeding save operations are done into this file.

17.2.2.10 Documents

The `Documents` interface has the following methods and properties:

Methods

- ActiveDocument
- Item
- NewDocument
- OpenDocument

Properties

- Application
- Count
- Parent

17.2.2.10.1 ActiveDocument

Property: ActiveDocument as Document

Description
Reference to the active document. If no document is open, ActiveDocument is null (nothing).

(C) 2015-2021 Altova GmbH

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.
1600 Invalid `Documents` object
1601 Invalid input parameter

17.2.2.10.2 Application

Property: Application as Application (read-only)

Description
Access the StyleVision application object.

Errors

1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.10.3 Count

Property: Count as long

Description
Count of open documents.

Errors

1600 Invalid `Documents` object
1601 Invalid input parameter

17.2.2.10.4 Item

Method: Item (*n* as long) as Document

Description
Gets the document with the index n in this collection. Index is 1-based.

Errors

1600 Invalid `Documents` object
1601 Invalid input parameter

17.2.2.10.5 NewDocument

Method: NewDocument() as Document

Return Value
None

Description
Creates a new empty document based on the previous template.

Errors
1000 The application object is invalid.
1005 Error when creating a new document
1006 Cannot create document
1600 Invalid `Documents` object
1601 Invalid input parameter

17.2.2.10.6 OpenDocument

Method: OpenDocument(strFileName as *String*) as Document

Return Value
None

Description
Opens an existing SPS file.

Errors
1000 The application object is invalid.
1002 Invalid file extension.
1003 Error when opening document.
1004 Cannot open document.
1600 Invalid `Documents` object
1601 Invalid input parameter

17.2.2.10.7 Parent

Property: Parent as Application (read-only)

Description
Access the StyleVision application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

17.2.2.11 Parameter

The `Parameter` interface has the following properties:

Properties

- Application
- Name
- Parent
- Value

17.2.2.11.1 Application

Property: Application as Application (read-only)

Description
Access the StyleVision application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.11.2 Name

Property: Name as String (read-only)

Errors
1600 Invalid `Documents` object
1601 Invalid input parameter

17.2.2.11.3 Parent

Property: Parent as Application (read-only)

Description
Access the StyleVision application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.11.4 Value

Property: Value as String (read-only)

Errors
1600 Invalid `Documents` object
1601 Invalid input parameter

17.2.2.12 Parameters

The `Parameters` interface has the following properties:

Properties

- Application
- Count
- Item
- Parent

17.2.2.12.1 Application

Property: Application as Application (read-only)

Description
Access the StyleVision application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.12.2 Count

Property: Count as long

Description
Count of parameters.

Errors
1600 Invalid `Parameters` object
1601 Invalid input parameter

(C) 2015-2021 Altova GmbH

17.2.2.12.3 Item

Method: Item (*n* as long) as Document

Description
Gets the document with the index n in this collection. Index is 1-based.

Errors
1600 Invalid `Documents` object
1601 Invalid input parameter

17.2.2.12.4 Parent

Property: Parent as Application (read-only)

Description
Access the StyleVision application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.13 SchemaSource

The `SchemaSource` interface has the following properties:

Properties

- Application
- IsMainSchemaSource
- Name
- Parent
- SchemaFileName
- TemplateFileName
- Type
- TypeName
- WorkingXMLFileName

17.2.2.13.1 Application

Property: Application as Application (read-only)

Description
Access the StyleVision application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.13.2 IsMainSchemaSource

Property: IsMainSchemaSource as Boolean (read-only)

Description
Returns true if schema source is the main schema source.

Errors
1400 Invalid schema source object.
1401 Invalid parameter.

17.2.2.13.3 Name

Property: Name as String (read-only)

Description
Use this property to retrieve the name of the schema source.

Errors
1400 The schema source object is not valid.
1401 Invalid parameter.

17.2.2.13.4 Parent

Property: Parent as `SchemaSource` (read-only)

Description
The parent object according to the object model.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.13.5 SchemaFileName

Property: SchemaFileName as String

(C) 2015-2021 Altova GmbH

Description
Use this property to retrieve the name of the Schema File.

Errors
1400 Invalid schema source object.
1401 Invalid parameter.
1403 Missing XMLSchema or DTD
1406 Error assigning schema File

17.2.2.13.6 TemplateFileName

Property: TemplateFileName as String

Description
Use this property to retrieve the name of the Working XML File.

Errors
1400 Invalid schema source object.
1401 Invalid parameter.
1403 Missing XMLSchema or DTD
1407 Error assigning Template XML File

17.2.2.13.7 Type

Property: Type as ENUMSchemaSourceType (read-only)

Description
Use this property to retrieve the type of the schema source.

Errors
1400 Invalid schema source object.
1401 Invalid parameter.

17.2.2.13.8 TypeName

Property: TypeName as String (read-only)

Description
Use this property to retrieve the type of the schema source.

Errors
1400 Invalid schema source object.
1401 Invalid parameter.

17.2.2.13.9 WorkingXMLFileName

Property: WorkingXMLFileName as String

Description
Use this property to retrieve the name of the Working XML File.

Errors
1400 Invalid schema source object.
1401 Invalid parameter.
1403 Missing XMLSchema or DTD
1203 Error assigning Working XML File

17.2.2.14 SchemaSources

The `SchemaSources` interface has the following properties:

Properties

- Application
- MainSchemaSource
- Parent

17.2.2.14.1 Application

Property: Application as Application (read-only)

Description
Access the StyleVision application object.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.14.2 Count

Property: Count as long (read-only)

Description
Count of schema sources.

(C) 2015-2021 Altova GmbH

Errors
1300 Invalid `SchemaSources` object
1301 Invalid input parameter

17.2.2.14.3 Item

Method: Item (*n* as long) as Document

Description
Gets the document with the index n in this collection. Index is 1-based.

Errors
1300 Invalid `SchemaSources` object
1301 Invalid input parameter

17.2.2.14.4 MainSchemaSource

Property: MainSchemaSource as SchemaSource (read-only)

Errors
1300 Invalid `SchemaSources` object
1301 Invalid input parameter

17.2.2.14.5 Parent

Property: Parent as SchemaSources (read-only)

Description
The parent object according to the object model.

Errors
1111 The application object is no longer valid.
1100 Invalid address for the return parameter was specified.

17.2.2.15 XMLData

Properties
Kind
Name
TextValue

HasChildren
MayHaveChildren
Parent

Methods
GetFirstChild
GetNextChild
GetCurrentChild

InsertChild
InsertChildAfter
InsertChildBefore
AppendChild

EraseAllChildren
EraseChild
EraseCurrentChild

IsSameNode

CountChildren
CountChildrenKind

GetChild
GetChildAttribute
GetChildElement
GetChildKind
GetNamespacePrefixForURI

HasChildrenKind
SetTextValueXMLEncoded

Description
The XMLData interface provides direct XML-level access to a document. You can read and directly modify the XML representation of the document. However, please, note the following restrictions:

- The XMLData representation is only valid when the document is shown in grid view or authentic view.
- When in authentic view, additional XMLData elements are automatically inserted as parents of each visible document element. Typically this is an XMLData of kind spyXMLDataElement with the Name property set to 'Text'.
- When you use the XMLData interface while in a different view mode you will not receive errors, but changes are not reflected to the view and might get lost during the next view switch.

Note also:

- Setting a new text value for an XML element is possible if the element does not have non-text children. A text value can be set even if the element has attributes.
- When setting a new text value for an XML element which has more than one text child, the latter will be deleted and replaced by one new text child.
- When reading the text value of an XML element which has more than one text child, only the value of the first text child will be returned.

(C) 2015-2021 Altova GmbH

17.2.2.15.1 AppendChild

Declaration: AppendChild (*pNewData* as XMLData)

Description
AppendChild appends pNewData as last child to the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1505 Invalid XMLData kind was specified.
1506 Invalid address for the return parameter was specified.
1507 Element cannot have Children
1512 Cyclic insertion - new data element is already part of document
1514 Invalid XMLData kind was specified for this position.
1900 Document must not be modified

Example

```
Dim objCurrentParent As XMLData
Dim objNewChild As XMLData

Set objNewChild = objSpy.ActiveDocument.CreateChild(spyXMLDataElement)
Set objCurrentParent = objSpy.ActiveDocument.RootElement

objCurrentParent.AppendChild objNewChild

Set objNewChild = Nothing
```

17.2.2.15.2 CountChildren

Declaration: CountChildren as long

Description
CountChildren gets the number of children.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.

17.2.2.15.3 CountChildrenKind

Declaration: CountChildrenKind (*nKind* as SPYXMLDataKind) as long

Description
CountChildrenKind gets the number of children of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.

17.2.2.15.4 EraseAllChildren

Declaration: EraseAllChildren

Description
EraseAllChildren deletes all associated children of the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1900 Document must not be modified

Example
The sample erases all elements of the active document.

```
Dim objCurrentParent As XMLData

Set objCurrentParent = objSpy.ActiveDocument.RootElement
objCurrentParent.EraseAllChildren
```

17.2.2.15.5 EraseChild

Method: EraseChild (Child as XMLData)

Description
Deletes the given child node.

Errors
1500 Invalid object.
1506 Invalid input xml
1510 Invalid parameter.

17.2.2.15.6 EraseCurrentChild

Declaration: EraseCurrentChild

Description
EraseCurrentChild deletes the current XMLData child object. Before you call EraseCurrentChild you must initialize an internal iterator with XMLData.GetFirstChild. After deleting the current child, EraseCurrentChild increments the internal iterator of the XMLData element. No error is returned when the last child gets erased

(C) 2015-2021 Altova GmbH

and the iterator is moved past the end of the child list. The next call to EraseCurrentChild however, will return error 1503.

Errors

1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object, or the iterator points past the last child.
1900 Document must not be modified

Examples

```
// ----------------------------------------
// XMLSpy scripting environment - JScript
// erase all children of XMLData
// ----------------------------------------
// let's get an XMLData element, we assume that the
// cursor selects the parent of a list in grid view
var objList = Application.ActiveDocument.GridView.CurrentFocus;

// the following line would be shorter, of course
//	objList.EraseAllChildren ();

// but we want to demonstrate the usage of EraseCurrentChild
if ((objList != null) && (objList.HasChildren))
{
	try
	{
		objEle = objList.GetFirstChild(-1);
		while (objEle != null)
			objList.EraseCurrentChild();
			// no need to call GetNextChild
	}
	catch (err)
		// 1503 - we reached end of child list
		{ if ((err.number & 0xffff) != 1503) throw (err); }
}
```

17.2.2.15.7 GetChild

Declaration: GetChild (*position* as long) as XMLData

Return Value

Returns an XML element as XMLData object.

Description

GetChild() returns a reference to the child at the given index (zero-based).

Available with TypeLibrary version 1.5

Errors

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

17.2.2.15.8 GetChildAttribute

Method: GetChildAttribute (strName as string) child as XMLData object (`NULL` on error)

Description
Retrieves the attribute having the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

17.2.2.15.9 GetChildElement

Method: GetChildElement (strName as string, nIndex as long) child as XMLData object (`NULL` on error)

Description
Retrieves the Nth child element with the given name.

Errors
1500 Invalid object.
1510 Invalid parameter.

17.2.2.15.10 GetChildKind

Declaration: GetChildKind (*position* as long, *nKind* as SPYXMLDataKind) as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetChildKind() returns a reference to a child of this kind at the given index (zero-based). The position parameter is relative to the number of children of the specified kind and not to all children of the object.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

(C) 2015-2021 Altova GmbH

17.2.2.15.11 GetCurrentChild

Declaration: GetCurrentChild as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetCurrentChild gets the current child. Before you call GetCurrentChild you must initialize an internal iterator with XMLData.GetFirstChild.

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

17.2.2.15.12 GetFirstChild

Declaration: GetFirstChild (*nKind* as SPYXMLDataKind) as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetFirstChild initializes a new iterator and returns the first child. Set nKind = -1 to get an iterator for all kinds of children.
REMARK: The iterator is stored inside the XMLData object and gets destroyed when the XMLData object gets destroyed. Be sure to keep a reference to this object as long as you want to use GetCurrentChild, GetNextChild or EraseCurrentChild.

Errors
1500 The XMLData object is no longer valid.
1501 Invalid XMLData kind was specified.
1504 Element has no children of specified kind.
1510 Invalid address for the return parameter was specified.

Example
See the example at XMLData.GetNextChild.

17.2.2.15.13 GetNamespacePrefixForURI

Method: GetNamespacePrefixForURI (strURI as string) strNS as string

Description
Returns the namespace prefix of the supplied URI.

Errors
1500 Invalid object.
1510 Invalid parameter.

17.2.2.15.14 GetNextChild

Declaration: GetNextChild as XMLData

Return Value
Returns an XML element as XMLData object.

Description
GetNextChild steps to the next child of this element. Before you call GetNextChild you must initialize an internal iterator with XMLData.GetFirstChild.

Check for the last child of the element as shown in the sample below.

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1510 Invalid address for the return parameter was specified.

Examples

```
' ---------------------------------------------
' VBA code snippet - iterate XMLData children
' ---------------------------------------------
On Error Resume Next
Set objParent = objSpy.ActiveDocument.RootElement

'get elements of all kinds
Set objCurrentChild = objParent.GetFirstChild(-1)

Do
        'do something useful with the child

        'step to next child
        Set objCurrentChild = objParent.GetNextChild
Loop Until (Err.Number - vbObjectError = 1503)
```

```
// ---------------------------------------
// XMLSpy scripting environment - JScript
// iterate through children of XMLData
// ---------------------------------------
try
{
        var objXMLData = ... // initialize somehow
        var objChild = objXMLData.GetFirstChild(-1);

        while (true)
```

```
        {
                // do something usefull with objChild

                objChild = objXMLData.GetNextChild();
        }
}
catch (err)
{
        if ((err.number & 0xffff) == 1504)
                ; // element has no children
        else if ((err.number & 0xffff) == 1503)
                ;       // last child reached
        else
                throw (err);
}
```

17.2.2.15.15 GetTextValueXMLDecoded

Method: GetTextValueXMLDecoded ()as string

Description
Gets the decoded text value of the XML.

Errors
1500 Invalid object.
1510 Invalid parameter.

17.2.2.15.16 HasChildren

Declaration: HasChildren as Boolean

Description
The property is true if the object is the parent of other XMLData objects. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

17.2.2.15.17 HasChildrenKind

Declaration: HasChildrenKind (*nKind* as SPYXMLDataKind) as Boolean

Description
The method returns true if the object is the parent of other XMLData objects of the specific kind.

Available with TypeLibrary version 1.5

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

17.2.2.15.18 InsertChild

Declaration: InsertChild (*pNewData* as XMLData)

Description
InsertChild inserts the new child before the current child (see also XMLData.GetFirstChild, XMLData.GetNextChild to set the current child).

Errors
1500 The XMLData object is no longer valid.
1503 No iterator is initialized for this XMLData object.
1505 Invalid XMLData kind was specified.
1506 Invalid address for the return parameter was specified.
1507 Element cannot have Children
1512 Cyclic insertion - new data element is already part of document
1514 Invalid XMLData kind was specified for this position.
1900 Document must not be modified

17.2.2.15.19 InsertChildAfter

Method: InsertChildAfter (Node as XMLData, NewData as XMLData)

Description
Inserts a new XML node (supplied with the second parameter) after the specified node (first parameter).

Errors
1500 Invalid object.
1506 Invalid input xml
1507 No children allowed
1510 Invalid parameter.
1512 Child is already added
1514 Invalid kind at position

17.2.2.15.20 InsertChildBefore

Method: InsertChildBefore (Node as XMLData, NewData as XMLData)

Description

(C) 2015-2021 Altova GmbH

Inserts a new XML node (supplied with the second parameter) before the specified node (first parameter).

Errors

1500 Invalid object.
1506 Invalid input xml
1507 No children allowed
1510 Invalid parameter.
1512 Child is already added
1514 Invalid kind at position

17.2.2.15.21 IsSameNode

Declaration: IsSameNode (*pNodeToCompare* as XMLData) as Boolean

Description

Returns true if pNodeToCompare references the same node as the object itself.

Errors

1500 The XMLData object is no longer valid.
1506 Invalid address for the return parameter was specified.

17.2.2.15.22 Kind

Declaration: Kind as SPYXMLDataKind

Description
Kind of this XMLData object. This property is read-only.

Errors

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

17.2.2.15.23 MayHaveChildren

Declaration: MayHaveChildren as Boolean

Description
Indicates whether it is allowed to add children to this XMLData object.
This property is read-only.

Errors

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

17.2.2.15.24 Name

Declaration: Name as String

Description
Used to modify and to get the name of the XMLData object.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

17.2.2.15.25 Parent

Declaration: Parent as XMLData

Return value
Parent as XMLData object. Nothing (or NULL) if there is no parent element.

Description
Parent of this element. This property is read-only.

Errors
1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

17.2.2.15.26 SetTextValueXMLEncoded

Method: SetTextValueXMLEncoded (*strVal* as String)

Description
Sets the encoded text value of the XML.

Errors
1500 Invalid object.
1513 Modification not allowed.

17.2.2.15.27 TextValue

Declaration: TextValue as String

Description
Used to modify and to get the text value of this XMLData object.

Errors

(C) 2015-2021 Altova GmbH

1500 The XMLData object is no longer valid.
1510 Invalid address for the return parameter was specified.

17.2.3 Enumerations

This is a list of all enumerations used by the StyleVision API. If your scripting environment does not support enumerations use the number-values instead.

17.2.3.1 ENUMApplicationStatus

Description
Enumeration to specify the current Application status.

Possible values:

eApplicationRunning	= 0
eApplicationAfterLicenseCheck	= 1
eApplicationBeforeLicenseCheck	= 2
eApplicationConcurrentLicenseCheckFailed	= 3
eApplicationProcessingCommandLine	= 4

17.2.3.2 ENUMAppOutputLine_Severity

Description
Enumeration values to identify the severity of an AppOutputLine.

Possible values:

eSeverity_Undefined	= -1
eSeverity_Info	= 0
eSeverity_Warning	= 1
eSeverity_Error	= 2
eSeverity_CriticalError	= 3
eSeverity_Success	= 4
eSeverity_Summary	= 5
eSeverity_Progress	= 6
eSeverity_DataEdit	= 7
eSeverity_ParserInfo	= 8
eSeverity_PossibleInconsistencyWarning	= 9
eSeverity_Message	= 10
eSeverity_Document	= 11
eSeverity_Rest	= 12
eSeverity_NoSelect	= 13
eSeverity_Select	= 14
eSeverity_Autoinsertion	= 15
eSeverity_GlobalResources_DefaultWarning	= 16

17.2.3.3 ENUMAppOutputLine_TextDecoration

Description
Enumeration values for the different kinds of text decoration of an AppOutputLine.

Possible values:

eTextDecorationDefault	= 0
eTextDecorationBold	= 1
eTextDecorationDebugValues	= 2
eTextDecorationDB_ObjectName	= 3
eTextDecorationDB_ObjectLink	= 4
eTextDecorationDB_ObjectKind	= 5
eTextDecorationDB_TimeoutValue	= 6
eTextDecorationFind_MatchingString	= 7
eTextDecorationValidation_Speclink	= 8
eTextDecorationValidation_ErrorPosition	= 9
eTextDecorationValidation_UnkownParam	= 10

17.2.3.4 ENUMSchemaSourceType

Description
Enumeration to specify the source schema type: XML Schema, DTD, DB, DB cell, User-Defined, or XBRL.

Possible values:

eSchemaSourceType_XSDorDTD	= 0
eSchemaSourceType_DB	= 1
eSchemaSourceType_DBCell	= 2
eSchemaSourceType_User	= 3
eSchemaSourceType_XBRL	= 4

17.2.3.5 ENUMSchemaType

Description
Enumeration to specify the schema type: W3C XML Schema or DTD.

Possible values:

eSchemaTypeW3CSchema	= 0
eSchemaTypeDTD	= 1

(C) 2015-2021 Altova GmbH

17.2.3.6 SPYAuthenticActions

Description
Actions that can be performed on AuthenticRange objects.

Possible values:

spyAuthenticInsertAt	= 0
spyAuthenticApply	= 1
spyAuthenticClearSurr	= 2
spyAuthenticAppend	= 3
spyAuthenticInsertBefore	= 4
spyAuthenticRemove	= 5

17.2.3.7 SPYAuthenticDocumentPosition

Description
Relative and absolute positions used for navigating with AuthenticRange objects.

Possible values:

spyAuthenticDocumentBegin	= 0
spyAuthenticDocumentEnd	= 1
spyAuthenticRangeBegin	= 2
spyAuthenticRangeEnd	= 3

17.2.3.8 SPYAuthenticElementKind

Description
Enumeration of the different kinds of elements used for navigation and selection within the AuthenticRange and AuthenticView objects.

Possible values:

spyAuthenticChar	= 0
spyAuthenticWord	= 1
spyAuthenticLine	= 3
spyAuthenticParagraph	= 4
spyAuthenticTag	= 6
spyAuthenticDocument	= 8
spyAuthenticTable	= 9
spyAuthenticTableRow	= 10
spyAuthenticTableColumn	= 11

17.2.3.9 SPYAuthenticMarkupVisibility

Description
Enumeration values to customize the visibility of markup with MarkupVisibility.

Possible values:

spyAuthenticMarkupHidden	= 0
spyAuthenticMarkupSmall	= 1
spyAuthenticMarkupLarge	= 2
spyAuthenticMarkupMixed	= 3

17.2.3.10 SPYAuthenticToolbarButtonState

Description
Authentic toolbar button states are given by the following enumeration:

Possible values:

authenticToolbarButtonDefault	= 0
authenticToolbarButtonEnabled	= 1
authenticToolbarButtonDisabled	= 2

17.2.3.11 SPYMouseEvent

Description
Enumeration type that defines the mouse status during a mouse event. Use the enumeration values as bitmasks rather then directly comparing with them.

Examples

```
' to check for ctrl-leftbutton-down in VB
If (i_eMouseEvent = (XMLSpyLib.spyLeftButtonDownMask Or XMLSpyLib.spyCtrlKeyDownMask)) Then
        ' react on ctrl-leftbutton-down
End If

' to check for double-click with any button in VBScript
If (((i_eMouseEvent And spyDoubleClickMask) <> 0) Then
        ' react on double-click
End If
```

Possible values:

spyNoButtonMask	= 0
spyMouseMoveMask	= 1
spyLeftButtonMask	= 2
spyMiddleButtonMask	= 4
spyRightButtonMask	= 8
spyButtonUpMask	= 16

(C) 2015-2021 Altova GmbH

```
spyButtonDownMask          = 32
spyDoubleClickMask         = 64
spyShiftKeyDownMask        = 128
spyCtrlKeyDownMask         = 256
spyLeftButtonDownMask      = 34   // spyLeftButtonMask | spyButtonDownMask
spyMiddleButtonDownMask    = 36   // spyMiddleButtonMask | spyButtonDownMask
spyRightButtonDownMask     = 40   // spyRightButtonMask | spyButtonDownMask
spyLeftButtonUpMask        = 18   // spyLeftButtonMask | spyButtonUpMask
spyMiddleButtonUpMask      = 20   // spyMiddleButtonMask | spyButtonUpMask
spyRightButtonUpMask       = 24   // spyRightButtonMask | spyButtonUpMask
spyLeftDoubleClickMask     = 66   // spyRightButtonMask | spyButtonUpMask
spyMiddleDoubleClickMask   = 68   // spyMiddleButtonMask | spyDoubleClickMask
spyRightDoubleClickMask    = 72   // spyRightButtonMask | spyDoubleClickMask
```

17.2.3.12 SPYValidateXSDVersion

Description
Enumeration values that select what XSD version to use. The XSD version that is selected depends on both (i) the presence/absence—and, if present, the value—of the `/xs:schema/@vc:minVersion` attribute of the XSD document, and (ii) the value of this enumeration.

`spyValidateXSDVersion_1_0` selects `XSD 1.0` if `vc:minVersion` is absent, or is present with any value.
`spyValidateXSDVersion_1_1` selects `XSD 1.1` if `vc:minVersion` is absent, or is present with any value.
`spyValidateXSDVersion_AutoDetect` selects `XSD 1.1` if `vc:minVersion=1.1`. If the `vc:minVersion` attribute is absent, or is present with a value other than `1.1`, then `XSD 1.0` is selected.

Possible values

```
spyValidateXSDVersion_AutoDetect      = 0
spyValidateXSDVersion_1_1             = 1
spyValidateXSDVersion_1_0             = 2
```

17.2.3.13 SPYValidateErrorFormat

Description
Enumeration values that select the format of the error message.

Possible values

```
spyValidateErrorFormat_Text           = 0
spyValidateErrorFormat_ShortXML       = 1
spyValidateErrorFormat_LongXML        = 2
```

17.2.3.14 SPYXMLDataKind

Description
The different types of XMLData elements available for XML documents.

Possible values:

spyXMLDataXMLDocStruct	= 0
spyXMLDataXMLEntityDocStruct	= 1
spyXMLDataDTDDocStruct	= 2
spyXMLDataXML	= 3
spyXMLDataElement	= 4
spyXMLDataAttr	= 5
spyXMLDataText	= 6
spyXMLDataCData	= 7
spyXMLDataComment	= 8
spyXMLDataPI	= 9
spyXMLDataDefDoctype	= 10
spyXMLDataDefExternalID	= 11
spyXMLDataDefElement	= 12
spyXMLDataDefAttlist	= 13
spyXMLDataDefEntity	= 14
spyXMLDataDefNotation	= 15
spyXMLDataKindsCount	= 16

(C) 2015-2021 Altova GmbH

17.3 ActiveX Integration

このセクションで説明されている StyleVision ユーザーインターフェイスと ActiveX コントロール機能を使用するアプリケーションに統合することができます。 ActiveX 技術により C++、C#、VB.NET、HTML などの統合のために使用される広範囲の言語が有効化されます (HTML 内に統合されている ActiveX コンポーネントは、Microsoft Internet Explorer バージョンと ActiveX をサポートすると共に作動する必要があります)。全てのコンポーネントはフル OLE コントロールです。Java への統合はラッパークラスにより与えられます。

ActiveX コントロールを、カスタムコードに統合するには、StyleVision Integration Package をインストールしてください (次を参照してください: http://www.altova.com/ide_integration.html)。すべての StyleVision を最初に、そして、StyleVision Integration Package そして他の必要条件を言語とプラットフォームに応じてインストールしてください (次を参照してください: 必要条件)。

統合の 2 つの異なるレベル間を柔軟に選択することができます: アプリケーションレベルとドキュメントレベル.

アプリケーションレベルの統合は、(メニュー、ツールバー、ペインを含む) StyleVision のフルインターフェイスを ActiveX コントロールをカスタムアプリケーションとして埋め込むことを意味します。例えば、多くのシナリオでは、カスタムアプリケーションを、StyleVision グラフィカルユーザーインターフェイスを埋め込む 1 つのフォームから構成することができます。

ドキュメントレベルの統合より先このアプローチより先簡単に実装することができますが、カスタム必要条件に従いを柔軟的に StyleVision グラフィカルユーザーインターフェイスを構成するためには適切ではない可能性があります。ドキュメントレベルの統合は、アプリケーションに StyleVision を 1 つ埋め込むことを意味します。メイン StyleVision コントロールのみではなく、メインドキュメント エディターウィンドウの実装も含まれます。また、オプションで、追加ウィンドウの実装も含まれます。このアプローチは GUI の構成に柔軟性を与えますが、選択された言語により ActiveX コントロールとの高度なインタラクションが必要とされます。

セクション アプリケーションレベルの統合 と ドキュメントレベルの統合 は、それぞれのレベルの主要なステップに関して説明しています。ActiveX 統合のサンプル セクションは、C#、HTML、および、Java のサンプルを提供します。これらのサンプルを確認することにより、正しい選択が直ぐに行うことができます。オブジェクトレファレンス セクションは、プロパティとメソッドに加え統合のために使用できるすべての COM オブジェクトに付いて説明しています。

StyleVision を Visual Studio プラグインとして使用するための情報に関しては、次を参照してください: Visual Studio 内の StyleVision。

17.3.1 必要条件

StyleVision ActiveX コントロールをカスタムアプリケーションに統合するには、以下が使用中のコンピューターにインストールされている必要があります:

- StyleVision
- StyleVision Integration Package は、次からダウンロードできます: http://www.altova.com/ide_integration.html

64 ビット ActiveX コントロールを統合するには、StyleVision の 64 ビット バージョンと StyleVision Integration Package をインストールする必要があります。Visual Studio を使用して、Microsoft .NET プラットフォームの下で開発されたアプリケーションに関しては、StyleVision の 32 ビット と 64 ビット バージョンと StyleVision Integration Package について下で説明されているとおりインストールします。

メモ Windows XP と Windows Server 2003 で統合パッケージが正確に作動するためには、再頒布可能パッケージを手動でインストールしてください: https://www.microsoft.com/en-US/download/details.aspx?id=48145.

Microsoft .NET (C#, VB.NET) with Visual Studio

StyleVision ActiveX コントロールを、Microsoft .NET の下で開発された 32 ビット アプリケーションに統合するには、以下を使用中のコンピューターにインストールしてください：

- Microsoft .NET Framework 4.0 または以降
- Visual Studio 2010/2012/2013/2015/2017/2019
- StyleVision 32 ビットとStyleVision Integration Package 32 ビット
- ActiveX コントロールがVisual Studio ツールボックスに追加されている必要があります（次を参照してください：ActiveX コントロールをツールボックスに追加する).

64 ビット ActiveX コントロールを統合する場合、次の必要条件が上記に加え満たされる必要があります:

- StyleVision 32 ビットとStyleVision Integration Package 32 ビット をインストールしてください（Visual Studio は32 ビットで作動するため、32 ビット ActiveX コントロールを Visual Studio デザイナーに与える必要があります。）
- StyleVision 64 ビットとStyleVision Integration Package 64 ビット をインストールしてください（実際の64 ビット ActiveX コントロールをランタイムにカスタムアプリケーションに与えます。）
- Visual Studio 内で、64 ビットビルド更新を作成し、この更新を使用してアプリケーションをビルドします。サンプルに関しては、次を参照してください：サンプルC# ソリューションの実行。

Java

Eclipse 開発環境を使用して、StyleVision ActiveX コントロールを Java アプリケーションに統合するには、次を使用中のコンピューターにインストールしてください：

- Java Runtime Environment（JRE）または Java Development Kit（JDK）7 または以降
- Eclipse
- StyleVision and StyleVision Integration Package

メモ StyleVision ActiveX コントロールの64 ビット バージョンを作動するには、Eclipse の64 ビット バージョン、および、StyleVision とStyleVision Integration Package の64 ビット バージョン ofを使用してください。

クライアントコンピューター上での StyleVision 統合とデプロイ

.NET アプリケーションを作成し他のクライアントに配布する場合は、クライアントコンピューター上に次をインストールしてください：

- StyleVision
- StyleVision Integration Package
- カスタムの統合コード、または、アプリケーション。

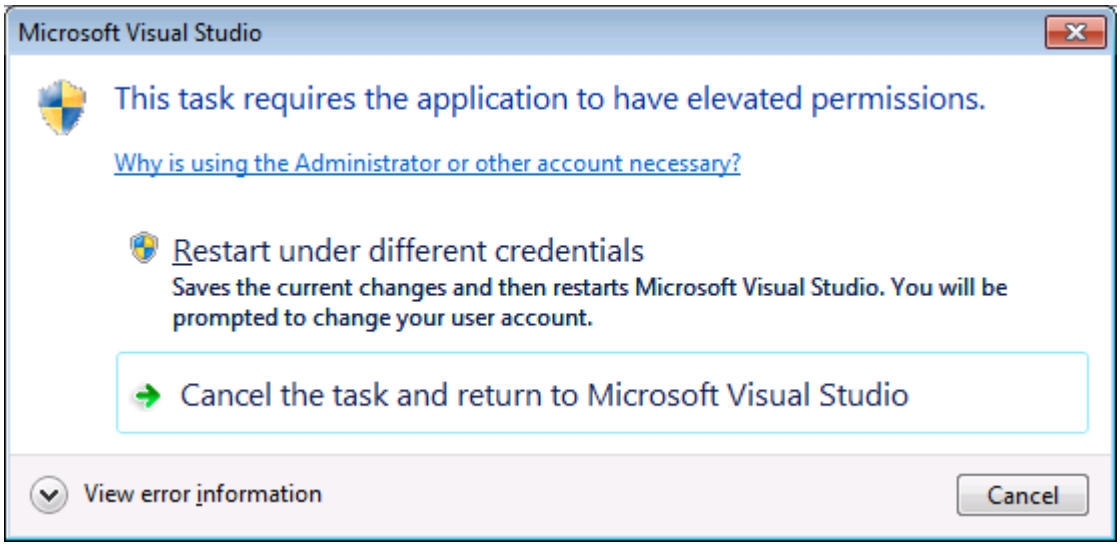
17.3.2 ActiveX コントロールをツールボックスに追加する

Visual Studio を使用して開発されたアプリケーション内で、StyleVision ActiveX コントロールを使用するには、コントロールが最初に Visual Studio ツールボックスに追加される必要があります:

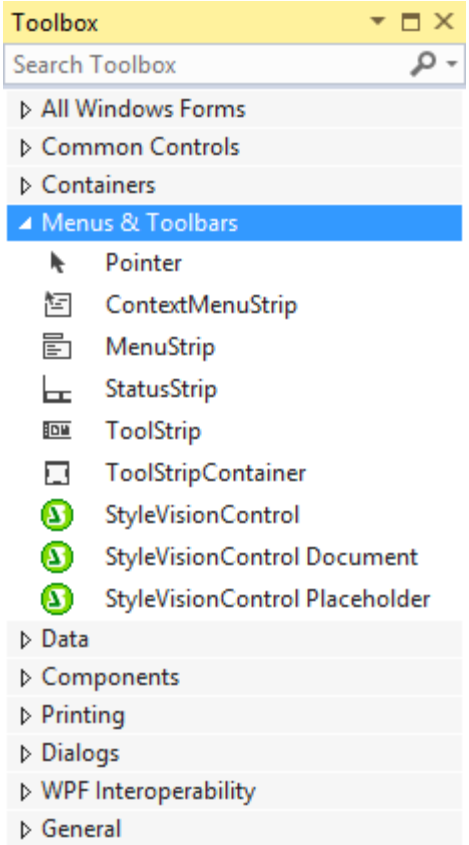
1. Visual Studio の「ツール」メニューから、「ツールボックスアイテムの選択」をクリックします。
2. 「COM コンポーネント」タブ、から、StyleVisionControl, StyleVisionControl ドキュメント、および、StyleVisionControl プレースホルダーの横のチェックボックスを選択します。

上記のコントロールが使用することができない場合、下のステップに従います:

1. 「COM コンポーネント」タブから「参照」をクリックします。**StyleVisionControl.ocx** を StyleVision インストールからフォルダーを選択します。StyleVision Integration Package をインストールすることを忘れないでください。それ以外の場合、このファイルを使用することはできません。次を参照してください: 必要条件.
2. を更に高いパーミッションと共に再起動するようにプロンプトされると、「異なる資格情報下で再起動する」をクリックします。



上記のステップが完了すると、StyleVision ActiveX コントロールは、Visual Studio ツールボックス内で使用できるようになります。



メモ　アプリケーションレベルの統合では、**StyleVisionControl** ActiveX コントロールのみが使用されます（次を参照してください: アプリケーションレベルの統合）。 StyleVisionControl ドキュメント とStyleVisionControl プレースホルダー コントロールは、ドキュメントレベルの統合のために使用されます（次を参照してください: ドキュメントレベルの統合）。

17.3.3 アプリケーションレベルの統合

アプリケーションレベルの統合により使用中のアプリケーションのウィンドウにStyleVision のインターフェイスを埋め込むことができます。 この種類の統合により、すべてのメニュー、ツールバー、ステータスバー、ドキュメントウィンドウ、および、ヘルパーウィンドウを含むStyleVision のユーザーインターフェイスを使用することができます。アプリケーションのユーザーインターフェイスのカスタム化はStyleVision が提供する機能に限られています。これには、ヘルパーウィンドウの整理とサイズ調整、及びメニューとツールバーのカスタム化が含まれています。

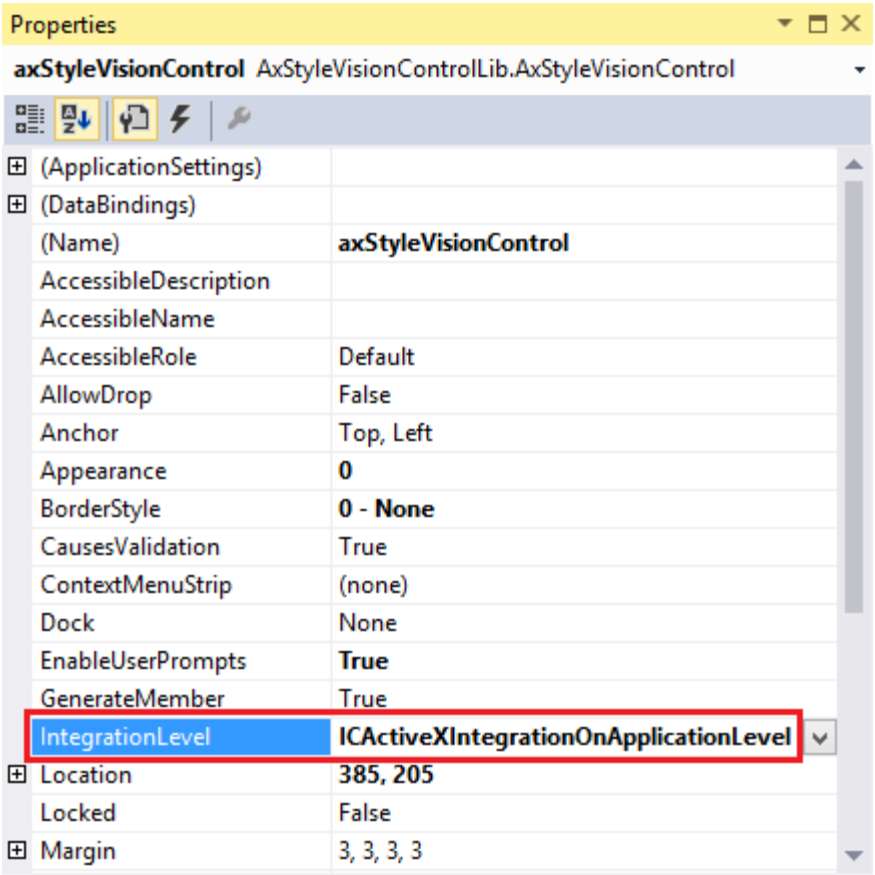
アプリケーションレベルで統合する場合、統合する必要のある唯一のActiveX コントロールは`StyleVisionControl` です。`StyleVisionControlDocument` または`StyleVisionControlPlaceHolder` ActiveX コントロールをインスタンス化、または、アクセス使用としないでください。

初期化、または、StyleVision の振る舞いを自動化するには、`StyleVisionControl`.のために説明されているプロパティ、メソッド、とイベントを使用してください。StyleVision 機能への複雑なアクセスのために`StyleVisionControl.application` を使用することを考慮してください。

StyleVision アプリケーションをHTML ページ、に埋め込むサンプルに関しては、次を参照してください: アプリケーションレベルのHTML 統合。

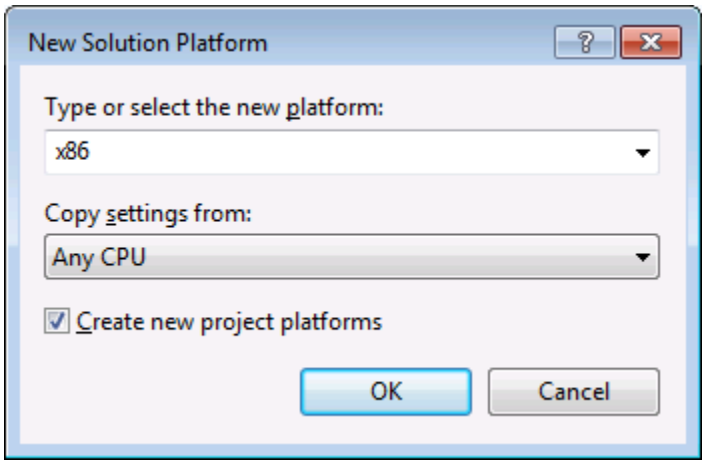
Visual Studio を使用する C# または VB.NET での StyleVision ActiveX コントロールをアプリケーションレベルで統合する基本的なアプリケーションを作成するステップは、以下のとおりです:

1. すべての必要条件が満たされていることを確認してください（次を参照してください: 必要条件）.
2. 新規の Visual Studio Windows Forms プロジェクトを新規の空のフォーム共に作成してください。
3. まだ行っていない場合、ActiveX コントロールをツールボックスに追加します（次を参照してください: ActiveX コントロールをツールボックスに追加する）。
4. ツールボックスから、**StyleVisionControl** をドラッグして、新規のフォームにドロップしてください。
5. フォームで **StyleVisionControl** を選択し、プロパティウィンドウ内で **IntegrationLevel** プロパティを **ICActiveXIntegrationOnApplicationLevel** に設定します。



6. ビルド（x86、x64）するプラットフォームに一致するビルドプラットフォームの更新を作成します。ビルドの更新の作成の方法は以下のとおりです:

   a. Visual Studio 内のソリューションを右クリックし、「**更新 マネージャー**」を選択します。
   b. 「**アクティブなソリューションプラットフォーム**」から、「**新規作成...**」を選択し、x86 または x64 更新を選択します（このサンプルでは、**x86**）。



Visual Studio 内でソリューションをビルドして実行する準備が整うと、ターゲットプラットフォーム(x86、x64)に一致する更新をしようしてビルドすることを確認してください。

17.3.4 ドキュメントレベルの統合

アプリケーションレベルの統合に比べ、ドキュメントレベルの統合複雑ですが、ActiveX コントロール。を使用することによりアプリケーションに StyleVision 機能を柔軟的に埋め込むことができます。このアプローチにより、コードは StyleVision ユーザーインターフェイスの次のパーツに部分的にアクセスすることができます:

- ドキュメントの編集ウィンドウ
- プロジェクトウィンドウ
- デザイン概要ウィンドウ
- スキーマツリーウィンドウ
- デザインツリーウィンドウ
- スタイルウィンドウ
- スタイルレポジトリウィンドウ
- プロパティウィンドウ
- メッセージウィンドウ

アプリケーションレベルの統合 で説明されているとおり、ActiveX アプリケーションレベルの統合のためには、**StyleVisionControl** のみが必要とされる唯一のコントロールです。しかしながら、ActiveX ドキュメントレベルの統合の場合は、機能 StyleVision が次の ActiveX コントロールにより与えられます:

1. **StyleVisionControl**
2. StyleVisionControl ドキュメント
3. StyleVisionControl プレースホルダ

これらのコントロールは StyleVision のアプリケーション インストールフォルダーで使用することのできる **StyleVisionControl.ocx** ファイルにより与えられます。Visual Studio を使用して ActiveX 統合を開発すると、これらのコントロールを Visual Studio ツールボックスに追加する必要があります(次を参照してください: ActiveX コントロールをツールボックスに追加する)。

ドキュメントレベルでの ActiveX コントロールのアプリケーションへの統合の基本のステップは、以下のとおりです:

1. 最初に、アプリケーション内で **StyleVisionControl** をインスタンス化します。「 **StyleVisionControl** ドキュメント」と「 **StyleVisionControl** プレースホルダ」コントロールへのサポートが有効化されるために、このコントロールの統合は必須で

す。IntegrationLevel プロパティを「ICActiveXIntegrationOnDocumentLevel」（または「1」）に設定することは重要です。ユーザーに対してコントロールを非表示にするには、「Visible」プロパティを「False」に設定します。

メモ ドキュメントレベルで統合する場合、未知の結果が生成される可能性があるため、**StyleVisionControl** の「開く」メソッドを使用しないでください。**StyleVisionControl** ドキュメント と**StyleVisionControl** プレースホルダ の対応するメソッドを代わりに使用して開いてください。

2. **StyleVisionControl** ドキュメント のインスタンスを少なくとも1つアプリケーション内に作成します。このコントロールは、アプリケーションにStyleVision のドキュメントの編集ウィンドウを与え、必要であれば、複数回インスタンス化することができます。

   既存のファイルを開くために、メソッド「開く」を使用します。ドキュメントに関連した機能にアクセスするには、メソッドとプロパティ「ドキュメント」を介してアクセスすることのできる「パス」と「保存」を使用します。

メモ コントロールは、読み取り専用モードをサポートしません。プロパティ「読み取り専用」の値は無視されます。

3. 任意で、アプリケーションに、アプリケーションで使用することのできる(ドキュメント ウィンドウ以外の)各追加ウィンドウのために **StyleVisionControl** プレースホルダ コントロールを追加することができます。

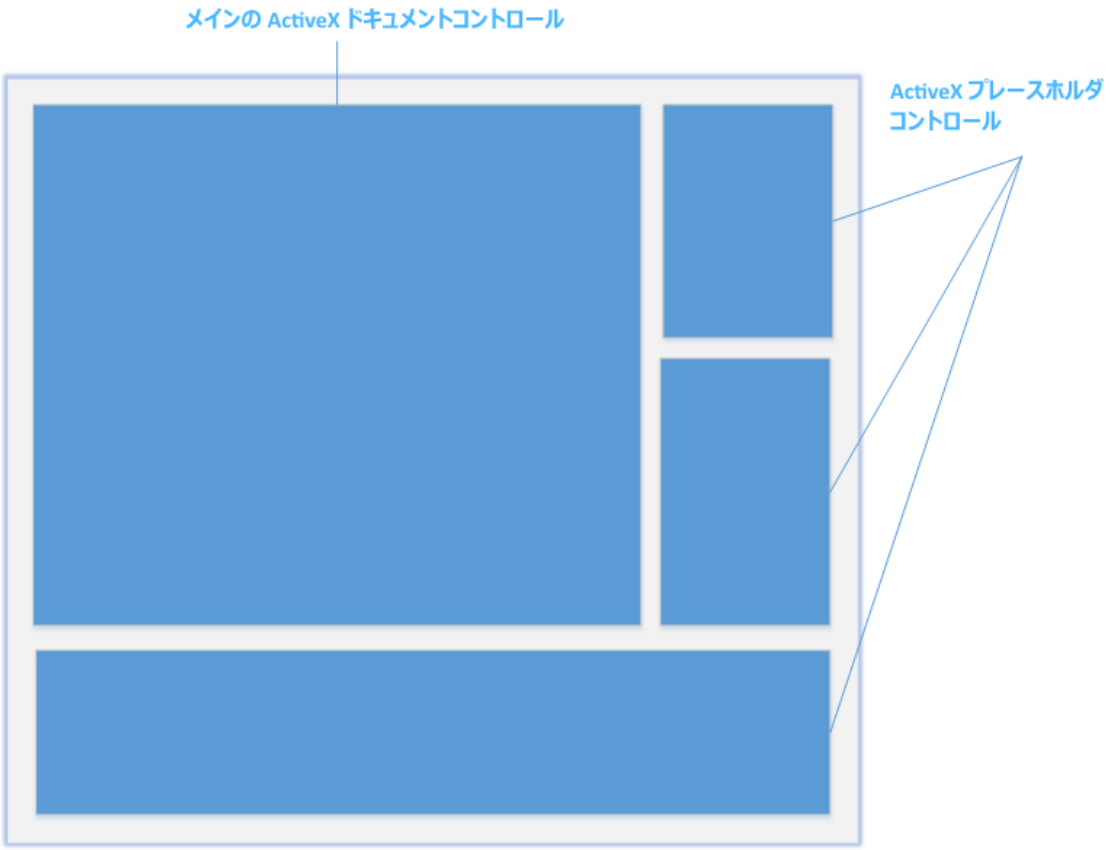
   StyleVisionControl プレースホルダ のインスタンスによりアプリケーションウィンドウに選択して StyleVision の追加ウィンドウを埋め込むことができます。ウィンドウの種類（例えば、プロジェクトウィンドウ）はプロパティ「PlaceholderWindowID」により定義することができます。有効なウィンドウの識別子のためには、次を参照してください: StyleVisionControlPlaceholderWindow.

メモ 各ウィンドウ識別子のためには **StyleVisionControl** プレースホルダ」のみを使用してください。

   StyleVision プロジェクトウィンドウを選択するプレースホルダーコントロールのために、追加メソッドを使用することができます。「OpenProject」を使用して、StyleVision プロジェクトをロードすることができます。StyleVision 自動化インターフェイスからプロパティプロジェクトとメソッドを使用して、他のプロジェクトに関連したオペレーションを操作します。

例えば、Visual Studio を使用したC# またはVB.NET 内では、StyleVision ActiveX コントロールをドキュメントレベルで統合する基本的なアプリケーションの作成ステップは、以下にリストされるものに類似しています。必要に応じて適用は複雑になる可能性があります。しかしながら、ActiveX ドキュメントレベルの統合のための必要条件として、下の命令を理解することは重要です。

1. 新規の空のフォームと共に新規のVisual Studio Windows Forms プロジェクトを作成します。
2. まだ行っていない場合、ActiveX コントロールをツールボックスに追加します（次を参照してください: ActiveX コントロールをツールボックスに追加する）。
3. ツールボックスから **StyleVisionControl** をドラッグして新規のフォームにドラッグします。
4. **StyleVisionControl** の「IntegrationLevel」プロパティを「ICActiveXIntegrationOnDocumentLevel」に、と「Visible」プロパティを「False」に設定します。コードから、または、「プロパティ」ウィンドウから行うことができます。
5. StyleVisionControl ドキュメント をツールボックスからフォームにドラッグします。 このコントロールは、使用中のアプリケーションにStyleVision のメインウィンドウを与え、ドキュメントのためにサイズを調整することができます。
6. 代わりに、フォームに1つまたは複数のStyleVisionControl プレースホルダ コントロールを追加することができます（アプリケーションが必要とする各追加ウィンドウの型など。例えば、「プロジェクト」ウィンドウなど）。このような追加プレースホルダーコントロールをメインドキュメントコントロールの下、または、左右に通常配置します。例:



7. 各「StyleVisionControl プレースホルダ」コントロールの「PlaceholderWindowID」プロパティを有効なウィンドウ識別子に設定してください。有効な値のリストに関しては、次を参照してください: StyleVisionControlPlaceholderWindow。
8. アプリケーションにコマンドを以下のように追加します(少なくとも、ドキュメントを開き、保存し、閉じる必要があります)。

StyleVision コマンドをクエリする

ドキュメントレベルで統合する場合は、StyleVision メニューまたはツールバーをアプリケーションで使用することはできません。代わりに、必要とするコマンドを取得し、状態を確認し、プログラム的に実行することができます:

- 使用することのできるすべてのコマンドを取得するには、**StyleVisionControl** の `CommandsList` プロパティを使用します。
- メニュー構造により整理されているコマンドを取得するには、メイン `Menu` プロパティを使用します。
- 表示される順番に整理されているコマンドを取得するには、`Toolbars` プロパティを使用します。
- StyleVision にコマンドを送信するには、`Exec` メソッドを使用します。
- コマンドが現在有効化されているか、または、無効化されているかクエリするために、`QueryStatus` メソッドを使用します。

これにより StyleVision コマンドをアプリケーションのメニューとツールバーに柔軟的に統合することができます。

StyleVision のインストールは、StyleVision 内で使用されているコマンドラベルを与えます。GIF 書式内のアイコンのために StyleVision インストールのフォルダー「<ApplicationFolder>¥Examples¥ActiveX¥Images」を参照してください。fコマンド名に対応するファイル名は、コマンド参照 セクション内にリストされています。

一般的な注意点

StyleVision の振る舞いを自動化するには、**StyleVisionControl**、StyleVisionControl ドキュメント と StyleVisionControl プレースホルダ内で説明されている プロパティ、メソッド、とイベントを使用してください。

StyleVision 機能への高度なアクセスに関しては、次の プロパティを考慮してください:

- StyleVisionControl.Application
- StyleVisionControlDocument.Document
- StyleVisionControlPlaceHolder.Project

これらのプロパティは StyleVision 自動化インターフェイス(StyleVisionAPI)へのアクセスを提供します

メモ ドキュメントを開くには、適切なドキュメントコントロールの StyleVisionControlDocument.Open または StyleVisionControlDocument.New を使用してください。 プロジェクトを開くには、StyleVision プロジェクトウィンドウを埋め込むプレースホルダ コントロール上の StyleVisionControlPlaceHolder.OpenProject を常に使用してください。

異なるプログラミング環境内で必要なコントロールをインスタンス化しアクセスする方法を説明するサンプルに関しては、次を参照してください: ActiveX 統合のサンプル。

17.3.5 ActiveX 統合のサンプル

このセクションは、異なるコンテナー環境とプログラム言語を使用する StyleVision ドキュメントレベルの統合のサンプル、を含みます(HTML セクションには、追加でアプリケーションレベルの統合のサンプルが含まれます)。全てのサンプルのソースコードは、StyleVision インストール内のフォルダー <ApplicationFolder>\Examples\ActiveX で使用することができます。

17.3.5.1 C#

C# and Visual Studio のための基本の ActiveX 統合サンプルソリューションは、フォルダー <ApplicationFolder>\Examples\ActiveX\C# 内にあります。ソースコードをコンパイルする前に、サンプルを実行して、全ての必要条件が満たされていることを確認してください(次を参照してください: サンプル C# ソリューションの実行)。

17.3.5.1.1 サンプル C# ソリューションの実行

フォルダー **<ApplicationFolder>\Examples\ActiveX\C#** 内で使用することのできるサンプル Visual Studio ソリューションは、StyleVision ActiveX コントロールの使用方法について説明しています。このソリューションをビルドして実行する前に、次のステップに注意してください:

ステップ 1: 必要条件をチェックする

Visual Studio 2010 または以降は、サンプルソリューションを開く必要があります。必要条件の完全なリストは、次を参照してください: 必要条件。

ステップ 2: 書き込みのパーミッションを保有するディレクトリにサンプルをコピーする

Visual Studio を管理者として作動することを回避するには、デフォルトの場所から実行する代わりに、書き込みのパーミッションを持つディレクトリへのソースコードをコピーします。

ステップ 3: すべての必要とされるコントロールプロパティをチェックし設定する

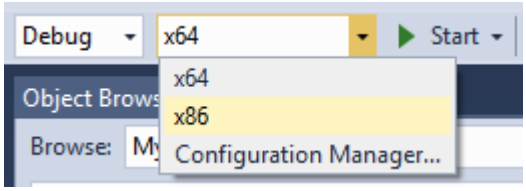
サンプルアプリケーションは、StyleVisionControlDocument の1つのインスタンスとStyleVisionControlPlaceHolder コントロールの複数のインスタンスを含んでいます。下のテーブルに表示されているとおりにこれらのコントロールの次のプロパティが設定されていることを確認してください:

コントロール名	プロパティ	プロパティ値
axStyleVisionControl	IntegrationLevel	ICActiveXIntegrationOnDocumentLevel
axStyleVisionControlHelperWndSchemaSource	PlaceholderWindowID	2
axStyleVisionControlHelperWndStyleRepository	PlaceholderWindowID	4
axStyleVisionControlHelperWndContextStyle	PlaceholderWindowID	6
axStyleVisionControlHelperWndContextProperties	PlaceholderWindowID	5
axStyleVisionControlHelperWndProject	PlaceholderWindowID	0
axStyleVisionControlHelperWndDesignOverview	PlaceholderWindowID	1
axStyleVisionControlHelperWndDesignTree	PlaceholderWindowID	3

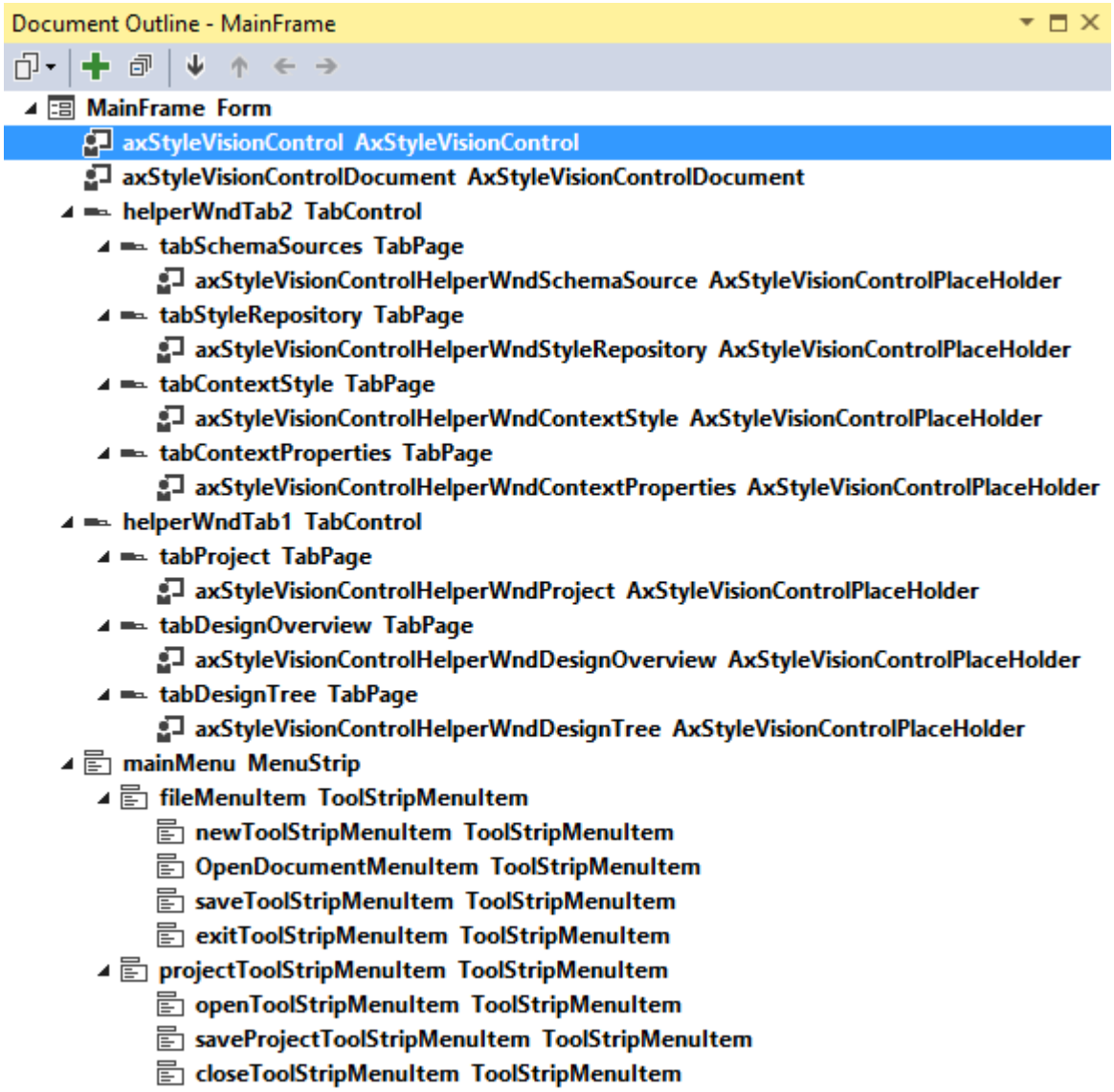
ActiveX コントロールのプロパティをビュー、または、設定する方法は、以下のとおりです:

1. デザイナーウィンドウ内で **MDIMain.cs** フォームを開きます。

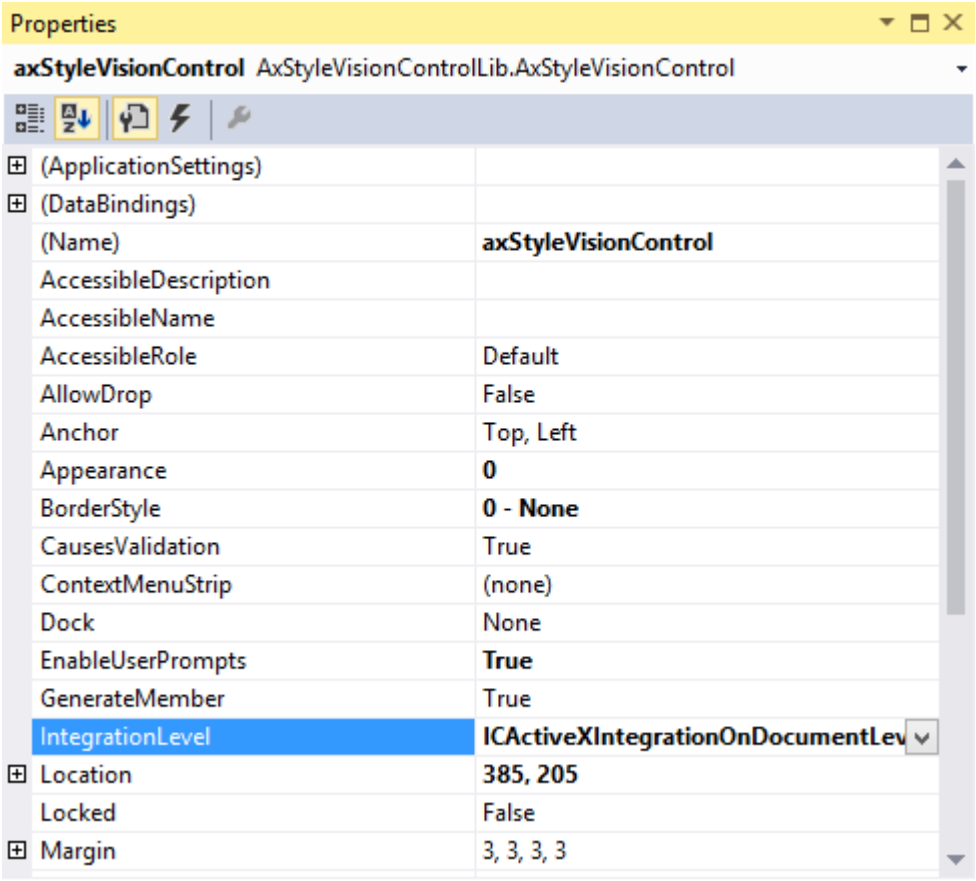
メモ 64 ビット Windows では、デザイナーウィンドウで開く前にVisual Studio ソリューションを「x86」にビルドを更新する必要がある可能性があります。64 ビットアプリケーションとしてサンプルをビルドする場合は、次を参照してください: 必要条件.



2. Visual Studio のドキュメント概要 ウィンドウを開きます(「表示」 メニューから、「他のウィンドウ | ドキュメントの概要」をクリックします)。



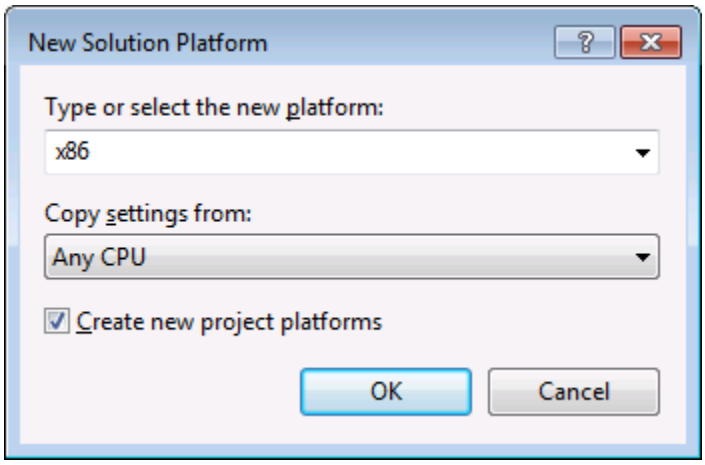
3. ドキュメント概要 ウィンドウ内のActiveX コントロールをクリックし、プロパティ ウィンドウ内の必要とされるプロパティを編集します。例:



ステップ 4: ビルドプラットフォームを設定する

- ビルド(x86、x64)するプラットフォームに一致するビルドプラットフォームの更新を作成します。ビルドの更新の作成の方法は以下のとおりです:

  a. Visual Studio 内のソリューションを右クリックし、「 更新マネージャー」を選択します。
  b. 「アクティブなソリューションプラットフォーム」から、「新規作成...」を選択し、x86 または x64 更新を選択します(このサンプルでは、**x86**)。



Visual Studio 内でソリューションをビルドして実行する準備が整うと、ターゲットプラットフォーム(x86、x64) に一致するように更新をしようしてビルドすることを忘れないでください。それ以外の場合、ランタイムエラーが起こる可能性があります。

17.3.5.2 HTML

このセクションのコードリストは、アプリケーションレベルとドキュメントレベルでのStyleVisionControl の統合方法を示しています。全てのサンプルのためのソースコードはStyleVision インストールのフォルダー`<ApplicationFolder>\Examples\ActiveX\HTML` 内にあります。サンプルは、32 ビット アプリケーションとして作動する場合、Internet Explorer 内でのみ作動することができます。

メモ 64 ビット モード上で作動する場合、nternet Explorer 10 または以降は、ActiveX コントロールをロードしません。

17.3.5.2.1 アプリケーションレベルの HTML 統合

このサンプルは、アプリケーションレベルでのHTML ページへのStyleVision コントロールの簡単な統合について説明しています。統合に関しては、次のセクションで説明されています:

- HTML コード内でStyleVisionControl を開始します。
- ボタンがドキュメントをロードするように実装し、コード生成タスクを自動化します。
- アプリケーションイベントのためのアクションを定義する。

このサンプルのためのコードは、StyleVision インストール内の次の場所で使用することができます:
`<ApplicationFolder>\Examples\ActiveX\HTML\StyleVisionActiveX_ApplicationLevel.htm`.

17.3.5.2.1.1 コントロールインスタンスを作成する

HTML `Object` タグは、StyleVisionControl のインスタンスを作成するために使用されます。`Classid` は、StyleVisionControl です。幅と高さがウィンドウのサイズを指定します。アプリケーションレベルがデフォルトのため、追加のパラメーターは必要ありません。

```
<OBJECT id="objStyleVisionControl"
              Classid="clsid:559db547-71fc-435d-aa8e-7faa321d0c6e"
              width="1000"
              height="700"
```

```
            VIEWASTEXT>
</OBJECT>
```

17.3.5.2.1.2 デフォルトのドキュメントを開くボタンを追加する

一部のタスクの自動化のシンプルなサンプルとして、ページにボタンを追加します:

```
<input type="button" value="Open Marketing Expenses" onclick="BtnOpenMEFile()">
```

クリックされると、定義済みドキュメントがStyleVisionControl 内で開かれます。異なるインストール上でサンプルを作動するため、StyleVisionControl に対して相対的なファイルを検索するためにメソッドを使用します。

```
<SCRIPT ID="Javahandlers" LANGUAGE="javascript">
// ---------------------------------
// open a pre-defined document
function BtnOpen()
{
       if(strPath.value.length > 0)
       {
              var absolutePath = MakeAbsolutePath(strPath.value);
              var objDoc = objStyleVisionControl.Open(absolutePath);
              if (objDoc == null)
                     alert("Unable to locate " + absolutePath + " at: " +
objStyleVisionControl.BaseHref);
       }
       else
       {
              alert("Please set path for the document first!");
              strPath.focus();
       }
}
</SCRIPT>
```

17.3.5.2.1.3 現在のドキュメントの有効性をチェックする

現在のドキュメントの有効性を次のスクリプトを使用してチェックすることができます:

```
// check validity of current document.
// if validation fails, show validation result in alert box .
function BtnTest()
{
       // get top-level object of automation interface
       var objApp = objStyleVisionControl.Application;

       // get the active document
       var objDocument = objApp.ActiveDocument;

       if ( objDocument == null )
              alert( "no active document found" );
       else
       {
              // define as arrays to support their usage as return parameters
              var errorText = new Array(1);
              var errorPos = new Array(1);
```

```
            var badData = new Array(1);

            var valid = objDocument.IsValid(errorText, errorPos, badData);

            if (! valid)
            {
                      // compose the error description
                      var text = errorText;

                      // access that XMLData object only if filled in
                      if (badData[0] != null)
                             text += "(" + badData[0].Name + "/" + badData[0].TextValue
+ ")";


                      alert("Validation error[" + errorPos + "]: " + text);
            }
            else
                   alert("Docuent is valid");
      }
}
```

17.3.5.2.1.4 カスタムイベントへの接続

サンプルは、プリンシパルを表示するためにStyleVisionControl カスタムイベントのための2つのイベントのコールバックを実装します:

```
<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentOpened" of StyleVisionControl object  -->
<SCRIPT LANGUAGE="javascript">
       function objStyleVisionControl::OnDocumentOpened( objDocument )
       {
       // alert("Document '" + objDocument.Name + "' opened!");
       }
</SCRIPT>

<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentClosed" of StyleVisionControl object  -->
<SCRIPT LANGUAGE="javascript">
       function objStyleVisionControl::OnDocumentClosed( objDocument )
       {
       // alert("Document '" + objDocument.Name + "' closed!");
       }
</SCRIPT>
```

17.3.5.2.2 ドキュメントレベルの HTML 統合

このサンプルは、ドキュメントレベルでHTML ページにStyleVision コントロールを統合する方法について説明しています。次のトピックについて説明されています:

- HTML コード内でStyleVisionControl ActiveX コントロールオブジェクトをインスタンス化します。
- StyleVision ファイルの編集を許可する StyleVisionControlDocument ActiveX コントロールをインスタンス化します。
- StyleVisionControl プロジェクトウィンドウのためのStyleVisionControlPlaceHolder をインスタンス化します。
- StyleVision ヘルパーウィンドウをホストするためにStyleVisionControlPlaceHolder をインスタンス化します。

- 頻繁に使用される StyleVision コマンドのためのシンプルなカスタマーツール バーを作成します。
- StyleVision のCOM 自動化インターフェイスを使用するボタンを追加します。
- コマンドボタンを更新するためのイベント ハンドラーを使用します。

このサンプルは、StyleVision インストールの<ApplicationFolder>\Examples\ActiveX\HTML\ フォルダー内のファイル StyleVisionActiveX_ApplicationLevel.htm 内で使用することがあります。

17.3.5.2.2.1 StyleVisionControl をインスタンス化する

StyleVisionControl HTML OBJECT タグ は、StyleVisionControl のインスタンスを作成するために使用されます。クラスID は StyleVisionControl です。ユーザーインターフェイスのためのコントロール間ジェージャーとして使用される幅と高さは、0 に設定されています。統合レベルは OBJECT タグ内で パラメーターとして指定されています。

```
<OBJECT id=""
        Classid="clsid:559db547-71fc-435d-aa8e-7faa321d0c6e"
        width="0"
        height="0"
        VIEWASTEXT>
   <PARAM NAME="IntegrationLevel" VALUE="1">
</OBJECT>
```

17.3.5.2.2.2 エディターウィンドウの作成

HTML OBJECT タグ は、編集ウィンドウを埋め込むために使用されます。追加のカスタム パラメーターは、新規のからのマッピングを使用してコントロールが初期化されるように指定します。

```
<OBJECT id="objDoc1"
        Classid="clsid:8EE4E77E-C793-468B-A0D8-5D6334CD6986"
        width="600"
        height="500"
        VIEWASTEXT>
    <PARAM NAME="NewDocument">
</OBJECT>
```

17.3.5.2.2.3 プロジェクトウィンドウの作成

HTML OBJECT タグ は、StyleVisionControlPlaceHolder ウィンドウを作成するために使用されます。最初の 追加カスタム パラメーターは、StyleVision プロジェクト ウィンドウを表示するためのプレースホルダを定義します。2番目の パラメーターは、StyleVision インストールで提供されたサンプルプロジェクト の1 つをロードします（以下の場所で見つけることができます: <yourusername>/MyDocuments フォルダー）。

```
<OBJECT id="objProjectWindow"
        Classid="clsid:E6ECBA9C-0E01-4FD5-98C3-C08B3D4824BC"
        width="200"
        height="200"
        VIEWASTEXT>
    <PARAM name="PlaceholderWindowID" value="-1">
          <PARAM name="FileName" value="StyleVisionExamples/Examples.svp">
```

```
</OBJECT>
```

17.3.5.2.2.4 ヘルパーウィンドウのためにプレースホルダを作成する

HTML OBJECT タグは、異なる StyleVision ヘルパーウィンドウをホストすることのできる StyleVisionControlPlaceHolder ActiveX コントロールをインスタンス化するために使用されます。最初から、ヘルパーウィンドウは表示されません。サンプルファイルを参照してください。

```
<OBJECT id="objEHWindow"
        Classid="clsid:E6ECBA9C-0E01-4FD5-98C3-C08B3D4824BC"
        width="200"
        height="200"
        VIEWASTEXT>
       <PARAM name="PlaceholderWindowID" value="-1">
</OBJECT>
```

3 つのボタンにより表示される実際のウィンドウを切り替えることができます。JavaScript 実行ボタンをクリックすることにより、プロパティ PlaceHolderWindowID を定義されている対応する値に設定します。

```
<input type="button" value="New File" onclick="BtnNewFile(objDoc1)">
<input type="button" value="Open file" onclick="BtnOpenFile(objDoc1)">
<input type="button" value="Save File" onclick="BtnSaveFile(objDoc1)">
<input type="button" value="Open project" onclick="BtnOpenProject(objDoc1)">
<input type="button" value="Save project" onclick="BtnSaveProject()">
<input type="button" value="Close project" onclick="BtnCloseProject()">
```

17.3.5.2.2.5 必須のイベントハンドラー

既に開かれているファイルのためなどの必須のイベントハンドラー:

```
<SCRIPT LANGUAGE="javascript">
       function objStyleVisionControl::OnOpenedOrFocused( strFilePath,
bOpenWithThisControl, bFileAlreadyOpened)
       {
              alert("Opening...");
              if (!bFileAlreadyOpened)
              DoOpenFile(objDoc1, strFilePath);
       }
</SCRIPT>
```

17.3.5.3 Java

StyleVision ActiveX コンポーネントに Java コードからアクセスすることができます。Java 統合は下にリストされるライブラリで提供されています。これらのライブラリは、StyleVision と StyleVision Integration Package の両方をインストールした後、StyleVision インストール内のフォルダー <ApplicationFolder>\Examples\JavaAPI で使用することができます(次を参照してください: 必要条件)。

- AltovaAutomation.dll: Altova 自動サーバーのための JNI ラッパー(StyleVision の 32 ビット インストールの場合)

- AltovaAutomation_x64.dll: Altova 自動サーバーのためのJNI ラッパー(StyleVision の64 ビット インストールの場合)
- AltovaAutomation.jar: Altova 自動サーバーにアクセスします
- StyleVisionActiveX.jar: StyleVision ActiveX インターフェイスをラップする Java クラス
- StyleVisionActiveX_JavaDoc.zip: Javadoc インターフェイスのためのヘルプドキュメントを含む Javadoc ファイル

メモ Java ActiveX 統合を使用するには、.dll と.jar ファイルが Java クラス検索パス内に含まれている必要があります。

サンプル Java プロジェクト
サンプル Java プロジェクトは製品のインストールにより提供されます。Java プロジェクト をテストし、使用用途に応じて変更することができます。詳細に関しては、次を参照してください: サンプル Java プロジェクト。

JavaRules に ActiveX コントロール名をマッピングする際のルール
ActiveX コントロールのドキュメントに関しては、次を参照してください: オブジェクトレファレンス。オブジェクトの名前変換は Java の場合、他の言語と比較して、若干異なります。具体的には、ActiveX コントロールと Java ラッパー間のマッピングのためのルールは、以下のとおりです:

- *クラスとクラス名*
  StyleVision ActiveX インターフェイスの全てのコンポーネントのために、コンポーネントの名前を持つ Java クラスが存在します。

- *メソッド名*
  Javadoc インターフェイス上のメソッド名は、COM インターフェイス上で使用される名前と同じですが、Java 名前変換に準拠する小文字で開始します。COM プロパティにアクセスするには、プロパティ名をプレフィックスする get と set から始まる Java メソッド使用することができます。ふプロパティが書き込みをサポートしない場合、setter メソッドを使用することはできません。サンプル: StyleVisionControl の IntegrationLevel プロパティ、Java メソッド getIntegrationLevel と setIntegrationLevel を使用することができます。

- *列挙*
  ActiveX インターフェイス内で定義される全ての列挙のために、Java 列挙は同じ名前と値を使用して定義されます。

- *イベント とイベント ハンドラー*
  イベントをサポートする自動化インターフェイス内の全てのインターフェイスのために、同じ名前と'イベント' を使用した Javadoc インターフェイスを使用することができます。単一のイベントのオーバーロードを簡素化するために、全てのイベントのためのデフォルトの実装を使用する Java クラスが提供されます。この Java クラスの名前は、イベント インターフェイスの名前と 'DefaultHandler' です。例:
  StyleVisionControl:アプリケーションにアクセスするための Java クラス
  StyleVisionControlEvents:StyleVisionControl のためのイベント インターフェイス
  StyleVisionControlEventsDefaultHandler:StyleVisionControlEvents のためのデフォルトのハンドラー

マッピングルールの例外
上記のルールの例外の一部は、以下のとおりです。これらは下にリストされています:

インターフェイス	Java 名
StyleVisionControlDocument, method New	newDocument
AuthenticView, method Goto	gotoElement
AuthenticRange, method Goto	gotoElement

AuthenticRange, method Clone	cloneRange

このセクション
このセクションは、Java コードからアクセスすることのできる基本のStyleVision ActiveX 機能の一部を紹介しています。次のサブセクションに整理されています:

- サンプルJava プロジェクト
- ActiveX コントロールの作成
- コントロール内のデータのロード
- 基本のイベントハンドリング
- メニュー
- UI アップデートイベントハンドリング
- StyleVision ノードツリーの作成

17.3.5.3.1 サンプル Java プロジェクト

StyleVision インストールパッケージには、アプリケーションフォルダー: <ApplicationFolder>\Examples\ActiveX\Java\ の ActiveX サンプルフォルダーにあるサンプルJava プロジェクトが含まれています。

Java サンプルは、Java. を使用して作成された共有されるデスクトップ アプリケーション内にStyleVisionControl を統合する方法を示しています。

バッチファイルBuildAndRun.bat を使用してコマンドラインからテストすることができ、また、Eclipse 内からサンプルプロジェクトをコンパイルして実行することもできます。これらのプロシージャの使用方法を参照してください。

ファイルリスト
Java サンプルフォルダーには、サンプルプロジェクトを作動するために必要な全てのファイルが含まれています。これらのファイルは下にリストされています:

.classpath	Eclipse プロジェクトヘルパーファイル
.project	Eclipse プロジェクトファイル
AltovaAutomation.dll	Java-COM ブリッジ: DLL パート (32-ビットインストール用)
AltovaAutomation_x64.dll	Java-COM ブリッジ: DLL パート (64-ビットインストール用)
AltovaAutomation.jar	Java-COM ブリッジ: Java ライブラリパート
BuildAndRun.bat	コンパイルするバッチファイル、および、コマンドラインプロンプトからサンプルコードを実行します。Java 仮想マシンがパラメータとして存在する箇所にフォルダーが期待されます。
StykeVisionActiveX.jar	StyleVision ActiveX コントロールのJava クラス
StyleVisionActiveX_JavaDoc.zip	Java API のためのヘルプドキュメントを含むJavadoc ファイル
StyleVisionContainer.java	Java サンプルソースコード

StyleVisionContainerEventHandler.java	Java サンプルソースコード
StyleVisionDialog.java	Java サンプルソースコード

サンプルの内容

サンプルは、AW T フレームウィンドウ内でStyleVision ドキュメント エディターウィンドウ、StyleVision プロジェクトウィンドウ、StyleVision スキーマツリーウィンドウとStyleVision デザイン概要ウィンドウを表示します。StyleVision のためにメインメニューを読み取ります。、同じ構造を持つAW T メニューを作成します。このメニューまたはプロジェクトを使用して、ドキュメントエディター内でファイルを開き作業することができます。

サンプルを必要に応じて変更することができます.

コードリスト内で次の特定の機能が説明されています:

- ActiveX コントロールの作成: 自動サーバーとして考えられているStyleVision を返します。または、作動中の場合、StyleVision をアクティブ化します。
- コントロール内のデータのロード: StyleVision にインストールされているサンプルドキュメントの1つを検索し開きます。
- 基本のイベントハンドリング: すべての開かれているドキュメントのビューをテキストビューに変えます。コードも開かれているドキュメント内での反復の方法を示しています。
- メニュー: アクティブなドキュメントを検証し、メッセージボックス内に結果を表示します。コードは出力パラメータの使用方法について説明しています。
- UI アップデートイベントハンドリング: StyleVision イベントの処理方法について説明しています。
- StyleVision ノードツリーの作成: StyleVision ノードツリーの作成とモーダルアクティブ化について説明しています。

サンプルフォルダーへのパスの更新

与えられているサンプルを作動する前に、**StyleVisionContainer.java** ファイルを編集します。具体的には、StyleVision サンプルファイルが使用中のオペレーティングシステム上で保存されている実際のフォルダーを参照する次のパスを確認します:

```
// Locate samples installed with the product.
final String strExamplesFolder = System.getenv( "USERPROFILE" ) + "\\My Documents\
\Altova\\StyleVision2021\\StyleVisionExamples\\";
```

コマンドラインからサンプルを実行する

コマンドラインからサンプルを作動します:

1. すべての必要条件が満たされていることを確認してください(次を参照してください: 必要条件).
2. コマンドプロンプトウィンドウを開き、サンプルJava プロジェクト フォルダーに対して現在のディレクトリを変更し、次を入力します:

```
buildAndRun.bat "<Path-to-the-Java-bin-folder>"
```

3. 「Enter」を押します。

StyleVisionContainer.java 内のJava ソースがコンパイルされ、実行されます。

Eclipse 内でサンプルをコンパイルし実行する

サンプルJava プロジェクトをEclipse にインポートする

1. すべての必要条件が満たされていることを確認してください（次を参照してください: 必要条件).
2. ファイル メニュー から、「インポート」をクリックします。
3. 「ワークスペースで既存のプロジェクトを開く」を選択し、次の場所にあるEclipse プロジェクトファイルを参照します: <ApplicationFolder>¥Examples¥ActiveX¥Java¥。このフォルダーでは書き込みのアクセスがないため、インポートダイアログボックス上の「ワークプレースにプロジェクトをコピーする」チェックボックスを選択することが奨励されます。

サンプルアプリケーションを作動するには、Package Explorer 内のプロジェクトを右クリックし、コマンド「実行 | Java アプリケーション」を選択します。

コード内のコメントを使用するとJava API クラスのためのヘルプを使用することができ、とEclipse 内でJavadoc ビューを有効化するには、メニューコマンド「Window | ビューの表示 | JavaDoc」を選択してください。

17.3.5.3.2 ActiveX コントロールの作成

下のコードリストは、ActiveX コントロールの作成方法を示しています。コンストラクターは、Java ラッパーオブジェクトを作成します。カンバスにより派生したオブジェクトをパネル、またはフレームに追加すると、ラップされたActiveX の生成がトリガーされます。

```
01   /**
02    * StyleVision manager control - always needed
03    */
04   public static StyleVisionControl         styleVisionControl = null;
05
06   /**
07    * StyleVisionDocument editing control
08    */
09   public static StyleVisionControlDocument     styleVisionDocument = null;
10
11   /**
12    * Tool windows - StyleVision place-holder controls
13    */
14   private static StyleVisionControlPlaceHolder   styleVisionProjectToolWindow = null;
15   private static StyleVisionControlPlaceHolder   styleVisionDesignToolWindow = null;
16   private static StyleVisionControlPlaceHolder   styleVisionSchemaTreeToolWindow = null;
17
18   // Create the StyleVision ActiveX controls. First should be StyleVisionControl
     // determining that we want to place document controls and place-holder
19   // controls individually. It gives us full control over the menu, as well.
20      styleVisionControl = new StyleVisionControl(
        ICActiveXIntegrationLevel.ICActiveXIntegrationOnDocumentLevel.getValue() );
21
22      styleVisionDocument = new StyleVisionControlDocument();
23      frame.add( styleVisionDocument, BorderLayout.CENTER );
24
25
26   // Create a project window and open the sample project in it
27      styleVisionProjectToolWindow = new StyleVisionPlaceHolder(
        StyleVisionControlPlaceholderWindow.StyleVisionControlProjectWindowToolWnd.getValue() );
28      styleVisionProjectToolWindow.setPreferredSize( new Dimension( 200, 200 ) );
```

17.3.5.3.3 コントロール内のデータのロード

下のコードリストは、ActiveX コントロール内にデータをロードを表示する方法を表示しています。

```
1    // Locate samples installed with the product.
```

```
2       final String strExamplesFolder = System.getenv( "USERPROFILE" ) +
        "\\My Documents\\Altova\\StyleVision2021\\StyleVisionExamples\\";
3       styleVisionProjectToolWindow.openProject( strExamplesFolder + "Examples.svp" );
```

17.3.5.3.4 基本のイベントハンドリング

下のコードリストは、基本のイベントの処理方法について表示しています。StyleVisionControl の open メソッド、または、メニューまたはプロジェクトツリーを使用してファイルを開こうとすると、onOpenedOrFocused イベントは、アタッチ済みのイベントハンドラーに送信されます。このイベントのための基本の処理により、StyleVisionDocumentControl の open メソッドを呼び出すことによりファイルを開きます。

```
01       // Open the PXF file when button is pressed
02       btnOpenPxf.addActionListener( new ActionListener() {
03         public void actionPerformed(ActionEvent e) {
04           try {
05             styleVisionControl.open( strExamplesFolder + "OrgChart.pxf" );
06           } catch (AutomationException e1) {
07             e1.printStackTrace();
08           }
09         }
10       } );
11        public void onOpenedOrFocused( String i_strFileName, boolean i_bOpenWithThisControl,
boolean i_bFileAlreadyOpened ) throws AutomationException
12   {
13     // Handle the New/Open events coming from the Project tree or from the menus
14     if ( !i_bFileAlreadyOpened )
15     {
16       // This is basically an SDI interface, so open the file in the already existing document
control
17       try {
18         StyleVisionContainer.initStyleVisionDocument();
19         StyleVisionContainer.styleVisionDocument.open( i_strFileName );
20       } catch (Exception e) {
21         e.printStackTrace();
22       }
23     }
24   }
```

17.3.5.3.5 メニュー

下のコードリストは、メニューアイテムの作成方法を表示しています。 各 StyleVisionCommand オブジェクトは、コマンドのID へのセットされている ActionCommand を持つ、対応する MenuItem オブジェクト、を取得します。全てのメニューアイテムにより生成されているアクションは、(閉じられたメカニズムのを再度解釈するなどの) 特定の処理を行う同じ関数により処理されます。または、exec メソッドを呼び出すことにより StyleVisionControl オブジェクトに実行を委任することができます。メニューの作成中にフィルされる menuMap オブジェクトは、後で使用されます (次のセクションを参照してください: UI アップデートイベントハンドリング).

```
01
02       // Load the file menu when the button is pressed
03       btnMenu.addActionListener( new ActionListener() {
04         public void actionPerformed(ActionEvent e) {
05           try {
06             // Create the menubar that will be attached to the frame
07             MenuBar mb = new MenuBar();
08             // Load the main menu's first item - the File menu
```

```
09              StyleVisionCommand xmlSpyMenu =
styleVisionControl.getMainMenu().getSubCommands().getItem( 0 );
10              // Create Java menu items from the Commands objects
11              Menu fileMenu = new Menu();
12              handlerObject.fillMenu( fileMenu, xmlSpyMenu.getSubCommands() );
13              fileMenu.setLabel( xmlSpyMenu.getLabel().replace( "&", "" ) );
14              mb.add( fileMenu );
15              frame.setMenuBar( mb );
16              frame.validate();
17            } catch (AutomationException e1) {
18              e1.printStackTrace();
19            }
20            // Disable the button when the action has been performed
21            ((AbstractButton) e.getSource()).setEnabled( false );
22          }
23        } ) ;
24    /**
25     * Populates a menu with the commands and submenus contained in a StyleVisionCommands object
26     */
27    public void fillMenu(Menu newMenu, StyleVisionCommands styleVisionMenu) throws
AutomationException
28    {
29      // For each command/submenu in the xmlSpyMenu
30      for ( int i = 0 ; i < styleVisionMenu.getCount() ; ++i  )
31      {
32        StyleVisionCommand styleVisionCommand = styleVisionMenu.getItem( i );
33        if ( styleVisionCommand.getIsSeparator() )
34          newMenu.addSeparator();
35        else
36        {
37          StyleVisionCommands subCommands = styleVisionCommand.getSubCommands();
38          // Is it a command (leaf), or a submenu?
39          if ( subCommands.isNull() || subCommands.getCount() == 0 )
40          {
41            // Command -> add it to the menu, set its ActionCommand to its ID and store it in the
menuMap
42            MenuItem mi = new MenuItem( styleVisionCommand.getLabel().replace( "&", "" ) );
43            mi.setActionCommand( "" + styleVisionCommand.getID() );
44            mi.addActionListener( this );
45            newMenu.add( mi );
46            menuMap.put( styleVisionCommand.getID(), mi );
47          }
48          else
49          {
50            // Submenu -> create submenu and repeat recursively
51            Menu newSubMenu = new Menu();
52            fillMenu( newSubMenu, subCommands );
53            newSubMenu.setLabel( styleVisionCommand.getLabel().replace( "&", "" ) );
54            newMenu.add( newSubMenu );
55          }
56        }
57      }
58    }
59      /**
60     * Action handler for the menu items
61     * Called when the user selects a menu item; the item's action command corresponds to the
command table for XMLSpy
62     */
63    public void actionPerformed( ActionEvent e )
64    {
65      try
66      {
67        int iCmd = Integer.parseInt( e.getActionCommand() );
68        // Handle explicitly the Close commands
69        switch ( iCmd )
```

```
70        {
71          case 57602:       // Close
72          case 34050:       // Close All
73            StyleVisionContainer.initStyleVisionDocument();
74            break;
75          default:
76            StyleVisionContainer.styleVisionControl.exec( iCmd );
77            break;
78        }
79      }
80      catch ( Exception ex )
81      {
82        ex.printStackTrace();
83      }
84
85    }
```

17.3.5.3.6 UI アップデートイベントハンドリング

下のコードリストは、UI-Update イベントハンドラーの作成方法を表示しています。

```
01    /**
02     * Call-back from the StyleVisionControl.
03     * Called to enable/disable commands
04     */
05    @Override
06    public void onUpdateCmdUI() throws AutomationException
07    {
08      // A command should be enabled if the result of queryStatus contains the Supported (1) and
Enabled (2) flags
09      for ( java.util.Map.Entry<Integer, MenuItem> pair : menuMap.entrySet() )
10
      pair.getValue().setEnabled( StyleVisionContainer.styleVisionControl.queryStatus( pair.getKey()
) > 2 );
11    }
12
13    /**
14     * Call-back from the StyleVisionControl.
15     * Usually called while enabling/disabling commands due to UI updates
16     */
17    @Override
18    public boolean onIsActiveEditor( String i_strFilePath ) throws AutomationException
19    {
20      try {
21        return
StyleVisionContainer.styleVisionDocument.getDocument().getFullName().equalsIgnoreCase( i_strFilePath
);
22      } catch ( Exception e ) {
23        return false;
24      }
25    }
```

17.3.5.3.7 StyleVision ドキュメントのプロパティのリスト

下のリストは、StyleVision ドキュメント（スキーマ、パラメーターなど）のプロパティがツリー内のノードとしてロードされるかを表示しています。

```
001 //access StyleVision Java-COM bridge
002 import com.altova.automation.StyleVision.*;
003
004 // access AWT/Swing components
005 import java.awt.*;
006 import javax.swing.*;
007 import javax.swing.tree.*;
008
009 /**
010  * A simple example of a tree control loading the structure from a StyleVision Document object.
011  * The class receives a Document object, loads its nodes in a JTree, and prepares
012  * for modal activation.
013  *
014  * Feel free to modify and extend this sample.
015  *
016  * @author Altova GmbH
017  */
018 class StyleVisionDialog extends JDialog
019 {
020   /**
021    * The tree control
022    */
023   private JTree myTree;
024
025   /**
026    * Root node of the tree control
027    */
028   private DefaultMutableTreeNode top ;
029
030   /**
031    * Constructor that prepares the modal dialog containing the filled tree control
032    * @param xml   The data to be displayed in the tree
033    * @param parent  Parent frame
034    */
035   public StyleVisionDialog( Document doc, Frame parent )
036   {
037     // Construct the modal dialog
038     super( parent, "Data tree", true );
039     // Arrange controls in the dialog
040     top = new DefaultMutableTreeNode("root");
041     myTree = new JTree(top);
042     setContentPane( new JScrollPane( myTree ) );
043     // Build up the tree; expand all nodes, hide root node
044     fillTree( top, doc );
045     for(int i = 0 ; i < myTree.getRowCount() ; ++i )
046       myTree.expandRow( i );
047     myTree.setRootVisible( false );
048   }
049
050   /**
051    * Loads the nodes of an XML element under a given tree node
052    * @param node  Target tree node
053    * @param doc Source XML element
054    */
055   private void fillTree( DefaultMutableTreeNode node, Document doc)
056   {
057     try
058     {
059       DefaultMutableTreeNode titleNode = new DefaultMutableTreeNode( "SchemaSources" ) ;
060       for (SchemaSource schema : doc.getSchemaSources())
061       {
062         String nodeText = "$" + schema.getName() ;
063         if ( schema.getIsMainSchemaSource() )
064           nodeText += " (main)";
```

```
065         DefaultMutableTreeNode newNode = new DefaultMutableTreeNode(nodeText ) ;
066         node.add( titleNode ) ;
067         titleNode.add( newNode );
068         String attribute = schema.getSchemaFileName() ;
069         if ( attribute.length() > 0 )
070         {
071           DefaultMutableTreeNode subNode = new DefaultMutableTreeNode( "SchemaFile" ) ;
072           subNode.add( new DefaultMutableTreeNode( attribute ) );
073           newNode.add( subNode );
074         }
075         attribute = schema.getWorkingXMLFileName() ;
076         if ( attribute.length() > 0 )
077         {
078           DefaultMutableTreeNode subNode = new DefaultMutableTreeNode( "Working XML" ) ;
079           subNode.add( new DefaultMutableTreeNode( attribute ) );
080           newNode.add( subNode );
081         }
082         attribute = schema.getTemplateFileName() ;
083         if ( attribute.length() > 0 )
084         {
085           DefaultMutableTreeNode subNode = new DefaultMutableTreeNode( "Template" ) ;
086           subNode.add( new DefaultMutableTreeNode( attribute ) );
087           newNode.add( subNode );
088         }
089       }
090
091       titleNode = new DefaultMutableTreeNode( "Parameters" ) ;
092       for (Parameter param : doc.getParameters())
093       {
094         DefaultMutableTreeNode newNode = new DefaultMutableTreeNode( "$" + param.getName() );
095         node.add( titleNode ) ;
096         titleNode.add( newNode );
097         DefaultMutableTreeNode subNode = new DefaultMutableTreeNode( "Value" ) ;
098         subNode.add( new DefaultMutableTreeNode( param.getValue() ) );
099         newNode.add( subNode );
100       }
101     }
102     catch (Exception e)
103     {
104       e.printStackTrace();
105     }
106   }
107
108 }
```

17.3.6 コマンド レファレンス

このセクションは、StyleVision 内で使用することのできる全てのメニューコマンドの名前と識別子をリストしています。各サブセクションは StyleVision. コマンドテーブルの対応するトップレベルメニューからのコマンドが以下のとおりに整理されています:

- 「メニューアイテム」列は、コマンドのメニューテキストを StyleVision で表示される順番に表示します。コマンドの機能を簡単に識別することができます。
- 「コマンド名」列は、StyleVision インストールディレクトリの **ActiveX\Images** フォルダーから同じ名前を持つアイコンを取得するために使用される文字列を指定します。
- 「ID」列は、このコマンドを実行またはクエリするメソッドの引数として与えられる列の数値識別子を表示します。

コマンドを実行するには、StyleVisionControl.Exec または StyleVisionControlDocument.Exec メソッドを利用します。コマンドのステータスをクエリするには、StyleVisionControl.QueryStatus または StyleVisionControlDocument.QueryStatus メソッドを使用します。

インストールされたStyleVision のエディションにより、コマンドの一部がサポートされない場合があります。

17.3.6.1 「ファイル」メニュー

「ファイル」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
XML スキーマ/DTD/XML から新規作成...	IDC_FILE_NEW_FROM_SCHEMA	37579
データベースから新規作成...	IDC_FILE_NEW_FROM_DB	37577
DB テーブルのXML 列から新規作成...	IDC_FILE_NEW_FROM_DBCELL	37861
XBRL タクソノミから新規作成...	IDC_FILE_NEW_FROM_XBRL_TAXONOMY	37912
HTML ファイルから新規作成...	IDC_FILE_NEW_FROM_HTML	37578
XSLT ファイルから新規作成...	IDC_FILE_NEW_FROM_XSLT	39737
W ord 2007+ ファイルから新規...	IDC_FILE_NEW_FROM_DOCX	39814
Excel 2007+ ファイルからの新規デザイン...	IDC_FILE_NEW_FROM_EXCEL	36054
新規作成 (空)	IDC_FILE_NEW_EMPTY	37576
開く...	ID_FILE_OPEN	57601
再ロード	IDC_FILE_RELOAD	36002
閉じる	ID_FILE_CLOSE	57602
全て閉じる	ID_FILE_CLOSE_ALL	37777
デザインの保存	ID_FILE_SAVE	57603
名前を付けて保存	ID_FILE_SAVE_AS	57604
全て保存する	ID_FILE_SAVE_ALL	37778
MobileTogether デザインファイルとしてエクスポート...	IDC_FILE_EXPORT_AS_MTD	36055
Authentic のXML データを保存	ID_SAVE_AUTHENTIC_XML	32860
Authentic のXML データに名前を付けて保存	ID_SAVE_AUTHENTIC_XML_AS	32812
生成されたXSLT-HTML ファイルの保存	IDC_SAVE_GEN_XSLT_HTML	37642
生成されたHTML ファイルの保存	IDC_SAVE_GEN_HTML	37636
生成されたXSLT-RTF ファイルの保存	IDC_SAVE_GEN_XSLT_RTF	37643
生成されたRTF ファイルの保存	IDC_SAVE_GEN_RTF	37638
生成されたXSLT-FO ファイルの保存	IDC_SAVE_GEN_XSLT_FO	37641

メニューアイテム	コマンド名	ID
生成されたFO ファイルの保存	IDC_SAVE_GEN_FO	37635
生成されたPDF ファイルの保存	IDC_SAVE_GEN_PDF	37637
生成されたXSLT-W ord 2007+ ファイルの保存	IDC_SAVE_GEN_XSLT_WORDML	37523
生成されたW ord 2007+ ファイルの保存	IDC_SAVE_GEN_WORDML	37529
生成されたデータベース スキーマの保存	IDC_SAVE_GEN_DB_SCHEMA	37633
生成されたデータベースXML データの保存	IDC_SAVE_GEN_DB_XML	37634
生成されたユーザー定義スキーマの保存	IDC_SAVE_GEN_USERDEF_SCHEMA	37639
生成されたユーザー定義 XML データの保存	IDC_SAVE_GEN_USERDEF_XML	37640
FlowForce へデプロイ	IDC_STYLEVISIONGUI_DEPLOY_TO_FLOWFORCE	39787
ASPX W eb アプリケーションの生成	IDC_SAVE_SCRIPT_FOR_DBEXTRACT_AND_XSLTGEN_ASPX	39755
プロパティ...	IDC_FILE_PROPERTIES	36027
印刷プレビュー	ID_FILE_PRINT_PREVIEW	57609
印刷	ID_FILE_PRINT	57607
最近使ったファイル	ID_FILE_MRU_FILE1	57616
終了	ID_APP_EXIT	57665

17.3.6.2 「編集」メニュー

「編集」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
元に戻す	ID_EDIT_UNDO	57643
やり直し	ID_EDIT_REDO	57644
切り取り	ID_EDIT_CUT	57635
コピー	ID_EDIT_COPY	57634
貼り付け	ID_EDIT_PASTE	57637
削除	ID_EDIT_CLEAR	57632
検索	ID_EDIT_FIND	57636

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
次を検索	ID_EDIT_FINDNEXT	36802
置換	ID_EDIT_REPLACE	57641
スタイルシート パラメーター...	IDC_EDIT_PARAMETERS	37573
マークアップの縮退/展開	IDC_SPSGUI_COLLAPSE_EXPAND_MARKUP	36804
全て選択	ID_EDIT_SELECT_ALL	57642

17.3.6.3 「プロジェクト」メニュー

「プロジェクト」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
新規のプロジェクト	IDC_ICPROJECTGUI_NEW	37200
プロジェクトを開く	IDC_ICPROJECTGUI_OPEN	37201
プロジェクトを再ロードする	IDC_ICPROJECTGUI_RELOAD	37202
プロジェクトを閉じる	IDC_ICPROJECTGUI_CLOSE	37203
プロジェクトを保存する	IDC_ICPROJECTGUI_SAVE	37204
プロジェクトにファイルを追加する	IDC_ICPROJECTGUI_ADD_FILES_TO_PROJECT	37205
グローバルリソースをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_GLOBAL_RESOURCE_TO_PROJECT	37239
URL をプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_URL_TO_PROJECT	37206
アクティブなファイルをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_ACTIVE_FILE_TO_PROJECT	37208
アクティブならびに関係するファイルをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_ACTIVE_AND_RELATED_FILES_TO_PROJECT	37209
プロジェクトフォルダーをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_FOLDER_TO_PROJECT	37210
外部フォルダーをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_EXT_FOLDER_TO_PROJECT	37211
外部 W eb フォルダーをプロジェクトに追加する	IDC_ICPROJECTGUI_ADD_EXT_URL_FOLDER_TO_PROJECT	37212
最近使ったプロジェクト	IDC_ICPROJECTGUI_RECENT	37224

17.3.6.4 「表示」メニュー

「表示」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
ダミーエントリ	ID_VIEW_TOOLBARS	37807
プロジェクト	ID_VIEW_PROJECT	57682
デザイン概要	ID_VIEW_DESIGNOVERVIEW	37883
スキーマソリー	ID_VIEW_SCHEMASOURCES	37805
デザインソリー	ID_VIEW_DESIGNTREE	37804
スタイルレポジトリ	ID_VIEW_STYLEREPOSITORY	37806
コンテキストプロパティ	ID_VIEW_CONTEXTPROPERTY_COMMON	37802
コンテキストスタイル	ID_VIEW_CONTEXTPROPERTY_STYLES	37803
1 つのテンプレートを表示	IDC_DESIGN_FILTER_ONE	37704
すべてのテンプレート タイプの表示	IDC_DESIGN_FILTER_ALL	37700
インポートされたテンプレートの表示	IDC_DESIGN_FILTER_IMPORTED	37870
メイン テンプレートの表示/非表示	IDC_DESIGN_FILTER_MAIN	37701
グローバルテンプレートの表示/非表示	IDC_DESIGN_FILTER_MATCH	37702
デザイン フラグメントの表示/非表示	IDC_DESIGN_FILTER_NAMED	37703
ズームイン	IDC_SPSGUI_ZOOM_IN	36805
ズームアウト	IDC_SPSGUI_ZOOM_OUT	36806
ズーム500%	IDC_SPSGUI_ZOOM_500	36807
ズーム400%	IDC_SPSGUI_ZOOM_400	36808
ズーム200%	IDC_SPSGUI_ZOOM_200	36809
ズーム150%	IDC_SPSGUI_ZOOM_150	36810
ズーム100%	IDC_SPSGUI_ZOOM_100	36811
ズーム75%	IDC_SPSGUI_ZOOM_75	36812
ズーム50%	IDC_SPSGUI_ZOOM_50	36813
ステータスバー	ID_VIEW_STATUS_BAR	59393

(C) 2015-2021 Altova GmbH

17.3.6.5 「挿入」メニュー

「挿入」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
コンテンツの挿入	IDC_INSERT_CONTENTS	37846
その他のコンテンツを挿入	IDC_INSERT_REST_OF_CONTENTS	37617
リッチエディットの挿入	IDC_INSERT_RICHEDIT	39773
入力フィールド	IDC_INSERT_EDITFIELD	37854
複数の入力フィールド	IDC_INSERT_MULTILINEEDITFIELD	37855
チェックボックス...	IDC_INSERT_CHECKBOX	37857
コンボボックス...	IDC_INSERT_COMBOBOX	37856
ラジオボタン...	IDC_INSERT_RADIOBUTTON	37858
ボタン	IDC_INSERT_BUTTON	37859
ナビゲーション	IDC_INSERT_DBNAVIGATION	37603
ナビゲーション + 移動	IDC_INSERT_DBNAVIGATIONGOTO	37604
クエリボタン	IDC_INSERT_DBQUERY	37605
値...	IDC_INSERT_AUTOCALC_VALUE	37596
入力フィールド...	IDC_INSERT_AUTOCALC_FIELD	37594
複数の入力フィールド...	IDC_INSERT_AUTOCALC_MULTILINE_FIELD	37595
日付選択の挿入	IDC_INSERT_DATEPICKER	37602
段落の挿入	IDC_INSERT_PARAGRAPH	37847
アドレス	IDC_INSERT_ADDRESS	37843
ブロック(div)	IDC_INSERT_BLOCK	37528
ブロック引用	IDC_INSERT_BLOCKQUOTE	37527
中央揃え	IDC_INSERT_CENTER	37534
フィールドセット	IDC_INSERT_FIELDSET	37574
整形済みテキスト	IDC_INSERT_FORMATTED	37580
書式設定、ラッピング済みテキスト	IDC_INSERT_FORMATTED_WRAP	37876
見出し 1 (h1)	IDC_INSERT_HEADING1	37585

メニューアイテム	コマンド名	ID
見出し 2 (h2)	IDC_INSERT_HEADING2	37586
見出し 3 (h3)	IDC_INSERT_HEADING3	37587
見出し 4 (h4)	IDC_INSERT_HEADING4	37588
見出し 5 (h5)	IDC_INSERT_HEADING5	37589
見出し 6 (h6)	IDC_INSERT_HEADING6	37590
バーコードの挿入...	IDC_INSERT_BARCODE	40221
チャートの挿入...	IDC_INSERT_CHART	40223
イメージの挿入...	IDC_INSERT_IMAGE	37593
水平線の挿入	IDC_INSERT_HORIZONTAL_LINE	37606
テーブルの挿入...	IDC_INSERT_TABLE	40212
リストの挿入...	IDC_INSERT_FORMAT_BULLETS	37848
ブックマークの挿入...	IDC_INSERT_BOOKMARK	37530
ハイパーリンクの挿入...	IDC_INSERT_HYPERLINK	37849
注釈の挿入	IDC_INSERT_FOOTNOTE	39797
条件の挿入...	IDC_INSERT_CONDITION	37850
出力ベースの条件の挿入	IDC_INSERT_CONDITION_PER_OUTPUT	37851
テンプレートの挿入...	IDC_INSERT_TEMPLATE	40216
ユーザー定義テンプレートの挿入...	IDC_INSERT_USER_DEFINED_TEMPLATE	40184
値テンプレートの挿入...	IDC_INSERT_VARIABLE_TEMPLATE	37531
ダミーエントリ	IDC_INSERT_NAMED_TEMPLATE	39300
レイアウトコンテナーの挿入	IDC_INSERT_LAYOUT_CONTAINER	40213
レイアウトボックスの挿入	IDC_INSERT_LAYOUT_BOX	40214
ラインの挿入	IDC_INSERT_SHAPE_LINE	40215
目次	IDC_INSERT_TOC_WIZARD	37618
目次ブックマーク	IDC_INSERT_MARKER	37608
目次ブックマーク(ウィザード)...	IDC_INSERT_MARKER_WIZARD	37609
目次参照	IDC_INSERT_REF	37613
入力テキスト	IDC_INSERT_REF_ENTRY_TEXT	37614

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
リーダー	IDC_INSERT_REF_LEADER	37615
ページ参照	IDC_INSERT_REF_PAGE_REFERENCE	37616
階層的な番号付け	IDC_INSERT_AUTO_NUMBER_LEVEL	37598
連続的な番号付け	IDC_INSERT_AUTO_NUMBER_FLAT	37597
目次レベル	IDC_INSERT_LEVEL	37607
目次レベル参照	IDC_INSERT_REFLEVEL	37860
レベルと機能するテンプレート	IDC_ASSIGN_LEVEL	37524
新規ドキュメントの挿入	IDC_INSERT_RESULT_DOCUMENT	39756
新規ページ	IDC_INSERT_PAGE_BREAK	37610
ページ番号	IDC_INSERT_PAGE_NUMBER	37611
ページ合計	IDC_INSERT_PAGE_TOTAL	37612
新規カラム	IDC_INSERT_COLUMN_BREAK	37864
新規ドキュメントセクション	IDC_INSERT_DOCUMENT_SECTION	37951
XBRL テーブルをタクソノミから挿入する...	IDC_INSERT_XBRL_TABLE_FROM_TAXONOMY	39813
コンテキスト	IDC_INSERT_XBRL_CONTEXT	37942
アイテム	IDC_INSERT_XBRL_ITEM	39793
期間	IDC_INSERT_XBRL_PERIOD	37909
期間開始	IDC_INSERT_XBRL_PERIOD_START	37947
期間終了	IDC_INSERT_XBRL_PERIOD_END	37948
識別子	IDC_INSERT_XBRL_IDENTIFIER	37910
Unit	IDC_INSERT_XBRL_UNITASPECT	39789
UnitRef	IDC_INSERT_XBRL_UNITREF	37946
フットノート	IDC_INSERT_XBRL_FOOTNOTE	37911
XDT ではないセグメント	IDC_INSERT_XBRL_NONXDTSEGMENT_ASPECT	39790
XDT ではないシナリオ	IDC_INSERT_XBRL_NONXDTSCENARIO_ASPECT	39791
XML ノード	IDC_INSERT_XBRL_XMLNODE	39792

メニューアイテム	コマンド名	ID
iXBRL 非表示データ	IDC_INSERT_IXBRL_HIDE	39721
iXBRL 分数	IDC_INSERT_IXBRL_FRACTION	39723
ユーザー定義要素	IDC_STYLEVISIONGUI_INSERT_USERXMLELEM	37908
ユーザー定義 ブロック	IDC_STYLEVISIONGUI_INSERT_USERXMLTEXT	37920

17.3.6.6 「囲む」メニュー

「囲む」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
テンプレート...	IDC_ENCLOSE_WITH_TEMPLATES	40180
ユーザー定義 テンプレート...	IDC_ENCLOSE_WITH_USER_DEFINED_TEMPLATES	40217
変数テンプレート...	IDC_ENCLOSE_WITH_VARIABLE_TEMPLATES	37520
段落	IDC_ENCLOSE_PARAGRAPH	37627
アドレス	IDC_ENCLOSE_ADDRESS	40194
ブロック(div)	IDC_ENCLOSE_BLOCK	40195
ブロック引用	IDC_ENCLOSE_BLOCKQUOTE	40196
中央揃え	IDC_ENCLOSE_CENTER	40197
フィールドセット	IDC_ENCLOSE_FIELDSET	40198
整形済みテキスト	IDC_ENCLOSE_FORMATTED	40199
書式設定、ラッピング済みテキスト	IDC_ENCLOSE_FORMATTED_WRAP	40200
見出し 1 (h1)	IDC_ENCLOSE_HEADING1	40201
見出し 2 (h2)	IDC_ENCLOSE_HEADING2	40202
見出し 3 (h3)	IDC_ENCLOSE_HEADING3	40203
見出し 4 (h4)	IDC_ENCLOSE_HEADING4	40204
見出し 5 (h5)	IDC_ENCLOSE_HEADING5	40205
見出し 6 (h6)	IDC_ENCLOSE_HEADING6	40206

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
箇条書きと段落番号...	IDC_ENCLOSE_FORMAT_BULLETS	37581
ブックマーク...	IDC_ENCLOSE_BOOKMARK	40207
ハイパーリンク...	IDC_ENCLOSE_HYPERLINK	37620
条件...	IDC_ENCLOSE_CONDITION	37599
出力ベース条件	IDC_ENCLOSE_CONDITION_PER_OUTPUT	37600
目次ブックマーク	IDC_ENCLOSE_MARKER	40210
目次ブックマーク(ウィザード)...	IDC_ENCLOSE_MARKER_WIZARD	40211
目次レベル	IDC_ENCLOSE_LEVEL	40208
目次レベル参照	IDC_ENCLOSE_REFLEVEL	40209
新規ドキュメント	IDC_ENCLOSE_RESULT_DOCUMENT	39758
ユーザー定義要素...	IDC_ENCLOSE_WITH_USER_DEFINED_XML_ELEMENT	40222

17.3.6.7 「テーブル」メニュー

「テーブル」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
テーブルの挿入...	IDC_INSERT_TABLE	40212
テーブルの削除	IDC_TABLE_DELETE	37658
テーブル ヘッダー列の追加	IDC_TABLE_ADD_HEADERCOL	37888
テーブル フッター列の追加	IDC_TABLE_ADD_FOOTERCOL	37889
テーブル ヘッダー行の追加	IDC_TABLE_ADD_HEADERROW	37900
テーブル フッター行の追加	IDC_TABLE_ADD_FOOTERROW	37901
行の追加	IDC_TABLE_APPEND_ROW	37657
列の追加	IDC_TABLE_APPEND_COL	37656
行の挿入	IDC_TABLE_INSERT_ROW	37664
列の挿入	IDC_TABLE_INSERT_COL	37662
行の削除	IDC_TABLE_DELETE_ROW	37660
列の削除	IDC_TABLE_DELETE_COL	37659

メニューアイテム	コマンド名	ID
左のセルを結合	IDC_TABLE_JOIN_LEFT	37666
右のセルを結合	IDC_TABLE_JOIN_RIGHT	37667
下のセルを結合	IDC_TABLE_JOIN_DOWN	37665
上のセルを結合	IDC_TABLE_JOIN_UP	37668
セルを水平方向に分割	IDC_TABLE_SPLIT_HORZ	37670
セルを垂直方向に分割	IDC_TABLE_SPLIT_VERT	37671
セル境界の表示	IDC_TABLE_SHOW_ZERO_BORDER	37669
テーブルマークアップの表示	IDC_TABLE_SHOW_UICELLS	37887
テーブルプロパティ...	IDC_TABLE_EDIT_PROPERTIES	37661
CALS/HTML テーブルの編集	IDC_EDIT_XMLTABLES	37533

17.3.6.8 「Authentic」メニュー

「Authentic」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
Authentic スクリプトの編集	IDC_EDIT_AUTHENTIC_SCRIPTPROJECT	37601
ツールバーボタンのカスタマイズ...	IDC_TEXT_STATE_ICONS	37672
マクロの参照をチェックする...	IDC_AUTHENTIC_CHECK_MACRO_REFERENCES	39751
日付選択の自動追加	IDC_TOGGLE_DATEPICKER_AUTOINSERT	37674
DB コントロールの自動追加	IDC_TOGGLE_DBCONTROL_AUTOINSERT	37675
Authentic ビューを再ロード	IDC_AUTHENTICGUI_RELOAD	32800
XML の検証	IDC_VALIDATE	32954
XML データの新たな行を選択し編集	IDC_CHANGE_WORKING_DB_XML_CELL	32861
XML 署名...	IDC_AUTHENTICGUI_XMLSIGNATURE	32862
XML エンティティを定義	IDC_DEFINE_ENTITIES	32805
マークアップを隠す	IDC_MARKUP_HIDE	32855
小さなマークアップの表示	IDC_MARKUP_SMALL	32858
大きなマークアップの表示	IDC_MARKUP_LARGE	32856

(C) 2015-2021 Altova GmbH

メニューアイテム	コマンド名	ID
混合マークアップの表示	IDC_MARKUP_MIXED	32857
太字	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEBOLD	32813
イタリック	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEITALIC	32814
下線	IDC_AUTHENTICGUI_RICHEDIT_TOGGLEUNDERLINE	32815
取り消し線	IDC_AUTHENTICGUI_RICHEDIT_TOGGLESTRIKETHROUGH	32816
前景色	IDC_AUTHENTICGUI_RICHEDIT_COLOR_FOREGROUND	32824
背景色	IDC_AUTHENTICGUI_RICHEDIT_COLOR_BACKGROUND	32830
左寄せ	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_LEFT	32818
中央揃え	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_CENTER	32819
右寄せ	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_RIGHT	32820
Justify	IDC_AUTHENTICGUI_RICHEDIT_ALIGN_JUSTIFY	32821
行の追加	IDC_ROW_APPEND	32806
行の挿入	IDC_ROW_INSERT	32809
行の複製	IDC_ROW_DUPLICATE	32808
行を上に移動	IDC_ROW_MOVE_UP	32811
行を下に移動	IDC_ROW_MOVE_DOWN	32810
行の削除	IDC_ROW_DELETE	32807

17.3.6.9 「データベース」メニュー

「データベース」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
データベースのクエリ	ID_VIEW_DBQUERY	37818

メニューアイテム	コマンド名	ID
DB フィルターの編集	IDC_EDIT_DBFILTERS	37571
DB フィルターをクリア	IDC_CLEAR_DBFILTERS	37547

17.3.6.10 「プロパティ」メニュー

「プロパティ」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
箇条書きと段落番号の編集...	IDC_EDIT_BULLETS_AND_NUMBERING	37839
設定済みの値フォーマット文字列	IDC_FORMAT_PREDEFINES	37582

17.3.6.11 「ツール」メニュー

「ツール」メニューには以下のコマンドが含まれています:

メニューアイテム	コマンド名	ID
スペル...	IDC_SPSGUI_SPELL_CHECK	36800
スペルオプション...	IDC_SPSGUI_SPELL_OPTIONS	36801
グローバルリソース	IDC_GLOBALRESOURCES	37401
	IDC_GLOBALRESOURCES_SUBMENUENTRY 1	37408
カスタマイズ...	IDC_APP_TOOLS_CUSTOMIZE	32959
オプション...	IDC_TOOLS_OPTIONS	37676

17.3.6.12 「ヘルプ」メニュー

「ウィンドウ」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
重ねて表示	ID_WINDOW_CASCADE	57650
上下に並べて表示	ID_WINDOW_TILE_HORZ	57651
左右に並べて表示	ID_WINDOW_TILE_VERT	57652
アイコンを整理して表示	ID_WINDOW_ARRANGE	57649

(C) 2015-2021 Altova GmbH

17.3.6.13 「ヘルプ」メニュー

「ヘルプ」メニューには以下のコマンドが含まれます:

メニューアイテム	コマンド名	ID
目次	ID_HELP_FINDER	32966
インデックス	ID_HELP_INDEX	32967
検索	ID_HELP_SEARCH	32969
ライセンスの登録	IDC_ACTIVATION	32970
注文フォーム	ID_HELP_ORDERFORM	32971
登録	ID_HELP_REGISTRATION	32972
最新情報のチェック	IDC_CHECK_FOR_UPDATES	32973
StyleVision 製品の比較	IDC_PRODUCT_COMPARISON	32955
サポートセンター	ID_HELP_SUPPORTCENTER	32961
W eb のFAQ	ID_HELP_FAQ	32962
コンポーネントのダウンロード	ID_HELP_COMPONENTS	32963
インターネット上のStyleVision	ID_HELP_INTERNET	32964
StyleVision トレーニング	ID_HELP_TRAINING	32965
StyleVision について	ID_APP_ABOUT	57664

17.3.7 オブジェクトレファレンス

オブジェクト:
StyleVisionCommand
StyleVisionCommands
StyleVisionControl
StyleVisionControlDocument
StyleVisionControlPlaceHolder

標準のStyleVision 機能へのアクセスを与えるには、**StyleVision 自動化インターフェイス** のオブジェクトにもアクセスすることができます。詳細に関しては、次を参照してください: StyleVisionControl.Application, StyleVisionControlDocument.Document と StyleVisionControlPlaceHolder.Project 。

17.3.7.1 StyleVisionCommand

プロパティ:
ID
Label
Name
IsSeparator
ToolTip
StatusText
Accelerator
SubCommands

詳細:
コマンドオブジェクトは次のオプションであることができます: 実行可能なコマンド, コマンド コンテナー(例えば、メニュー、サブメニュー、または、ツールバー)、または、メニューセパレーター。現在の Command オブジェクト内に保管される情報の種類を決定するには、ID、IsSeparator、と SubCommands プロパティをクエリしてください。

コマンドオブジェクト	使用可能な場合...
実行可能なコマンド	• ID はゼロよりも大きい場合 • IsSeparator が false の場合 • SubCommands が空白の場合
コマンド コンテナー	• ID はゼロの場合 • IsSeparator が true の場合 • SubCommands が Command オブジェクトを含む場合
セパレーター	• ID はゼロの場合 • IsSeparator が true の場合

17.3.7.1.1 Accelerator

プロパティ: Accelerator を string として

詳細:
コマンドのために定義されたアクセラレータキーを返します。コマンドにアクセラレーターキーが割り当てられていない場合、このプロパティは、空の文字列を返します。 アクセラレータキーの文字列の表示は次のフォーマットを取ります:

```
[ALT+][CTRL+][SHIFT+]key
```

Windows プラットフォーム SDK 関数 GetKeyNameText を使用して key が変換された箇所。

17.3.7.1.2 ID

プロパティ: ID を long として

詳細:

(C) 2015-2021 Altova GmbH

このプロパティは、コマンドの一意の識別子を取得します。コマンドのID は(Exec を使用して) コマンドを実行するために、または、(QueryStatus を使用して) ステータスをクエリするために必須です。コマンドが他のコマンド (例えば、トップレベルのメニュー) のためのコンテナー、または、セパレーターの場合、ID は0 です。

17.3.7.1.3 IsSeparator

プロパティ: IsSeparator を boolean として

詳細:
コマンドオブジェクトがメニューセパレーターの場合、の場合、プロパティは、 true を返します。それ以外の場合は、 false を返します。次も参照してください: コマンド.

17.3.7.1.4 Label

プロパティ: Label を string として

詳細:
このプロパティは、StyleVision のグラフィカルなユーザーインターフェイス内に表示されるとおりのコマンドのテキストを取得します。コマンドがセパレーターの場合、「ラベル」は、空の文字列です。このプロパティは、関連したGUI テキストを持たない、一部のツールバーコマンドのために空の文字列を返します。

17.3.7.1.5 Name

プロパティ: Name を string として

詳細:
このプロパティは一意のコマンド名を取得します。使用できる箇所で、この値はコマンドのアイコンファイルを取得するために使用できます。使用することのできるアイコンファイルは、使用中のStyleVision インストールのフォルダー **<ApplicationFolder>\Examples\ActiveX\Images** 内にあります。

17.3.7.1.6 StatusText

プロパティ: Label を string として

詳細:
コマンドが選択されると、ステータステキストは、StyleVision のステータスバー内で表示されるテキストです。他のコマンドのセパレーターまたは、コンテナオではないコマンドオブジェクトに対してのみ適用することができます。それ以外の場合、プロパティは空の文字列です。

17.3.7.1.7 SubCommands

プロパティ: SubCommands を Commands として

詳細:
SubCommands プロパティは、現在のコマンドのサブコマンドである Command オブジェクトのコレクションを取得します。プロパティは、他のコマンド(メニュー、サブメニュー、または、ツールバー)のためのコンテナーであるコマンドにのみ適用されます。このようなコンテナーコマンドは、0 に設定された ID を持ち、および、 false に設定された IsSeparator プロパティを持ちます。

17.3.7.1.8 ToolTip

プロパティ: ToolTip を string として

詳細:
このプロパティは、各コマンドのためのヒントを表示するテキストを取得します。コマンドがツールのヒントのテキストを持たない場合、プロパティは、空の文字列を返します。

17.3.7.2 StyleVisionCommands

プロパティ:
Count
Item

詳細:
StyleVisionControl のコマンドレベルとID へのアクセスを取得するための Command オブジェクトのコレクションです。これらのコマンドは、Exec メソッドを使用して実行することができます。 また、ステータスをQueryStatusを使用してクエリすることができます。

17.3.7.2.1 Count

プロパティ: Count を long として

詳細:
このコレクションのレベル上の Command オブジェクトの数。

17.3.7.2.2 Item

プロパティ: Item (n を long として) をCommandとして

詳細:
このコレクション内のインデックス n を持つコマンドを取得します。インデックスは 1 ベースです。

17.3.7.3 StyleVisionControl

プロパティ:
IntegrationLevel
Appearance
Application
BorderStyle
CommandsList
EnableUserPrompts
MainMenu
Toolbars

(C) 2015-2021 Altova GmbH

メソッド:
Open
Exec
QueryStatus

イベント:
OnUpdateCmdUI
OnOpenedOrFocused
OnCloseEditingWindow
OnFileChangedAlert
OnContextChanged
OnDocumentOpened
OnValidationWindowUpdated

このオブジェクトは完全な ActiveX コントロールで、StyleVision ライブラリがアプリケーションレベルモードで使用されている場合、表示されます。

17.3.7.3.1 プロパティ

以下のプロパティが定義されています:

IntegrationLevel
EnableUserPrompts
Appearance
BorderStyle

コマンドに関連したプロパティ:
CommandsList
MainMenu
Toolbars

StyleVisionAPI へのアクセス:
Application

17.3.7.3.1.1 Appearance

プロパティ: Appearance を short として

ディスパッチ ID : -520

詳細:
0 に対して等しくない値は、コントロールの周りでクライアントエッジを表示します。デフォルトの値は 0 です。

17.3.7.3.1.2 Application

プロパティ: Application を Application として

ディスパッチ ID : 1

詳細:
Application プロパティは、完全な StyleVision 自動サーバーAPI の Application オブジェクトへのアクセスを与えます。プロパティは読み取り専用です

17.3.7.3.1.3 BorderStyle

プロパティ: BorderStyle を short として

ディスパッチ ID : -504

詳細:
1 の値は、薄いボーダーを持つコントロールを表示します。デフォルトの値は 0 です。

17.3.7.3.1.4 CommandsList

プロパティ: CommandList を Commands として (読み取り専用)

ディスパッチ ID : 1004

詳細:
このプロパティは、StyleVisionControl を持つ定義された全てのコマンドのフラットリストを返します。メニュー構造に従い、コマンドを整理するには、MainMenu を使用します。ツールバーコマンドを取得するには、Toolbars を使用します。

```
public void GetAllStyleVisionCommands()
{
    // Get all commands from the StyleVision ActiveX control assigned to the current
form
    StyleVisionControlLib.StyleVisionCommands commands =
this.axStyleVisionControl1.CommandList;
    // Iterate through all commands
    for (int i = 0; i < commands.Count; i++)
    {
      // Get each command by index and output it to the console
      StyleVisionControlLib.StyleVisionCommand cmd =
axStyleVisionControl1.CommandList[i];
      Console.WriteLine("{0} {1} {2}", cmd.ID, cmd.Name, cmd.Label.Replace("&", ""));
    }
}
```

C# サンプル

(C) 2015-2021 Altova GmbH

17.3.7.3.1.5 EnableUserPrompts

プロパティ: `EnableUserPrompts` をboolean として

ディスパッチ ID : 1006

詳細:
このプロパティを *false* に設定すると、コントロール内のユーザープロンプトを無効化します。デフォルトの値は *true* です。

17.3.7.3.1.6 IntegrationLevel

プロパティ: `IntegrationLevel` を`ICActiveXIntegrationLevel` として

ディスパッチ ID : 1000

詳細:
The `IntegrationLevel` プロパティ determines コントロールのオペレーションモード。次も参照してください: アプリケーションレベルの統合 とドキュメントレベルの統合 。

メモ `StyleVisionControl` オブジェクトの作成後すぐに、このプロパティを設定することは重要です。

17.3.7.3.1.7 MainMenu

プロパティ: `MainMenu` を Command として(読み取り専用)

ディスパッチ ID : 1003

詳細:
このプロパティは、 StyleVisionControl メインメニュー内で使用することのできる構造の情報とコマンドを `Command` オブジェクトとして提供します。オブジェクトは、 StyleVision の使用することのできる全てのサブメニューを含んでいます(例「ファイル」、「編集」、「表示」など)。サブメニューオブジェクトにアクセスするには、`MainMenu` プロパティの `SubCommands` プロパティを使用します。各サブメニューは `Command` オブジェクトでもあります。各サブメニューのために、対応する子コマンドセパレーターを取得するために、 `SubCommands` プロパティを使用して更に反復することができます(このテクニックが使用される可能性があります。例えば、アプリケーションメニューをプログラム的に作成するなど)。メニューコマンドの一部は、 `SubCommands` プロパティを持つ他のメニューコマンドのためのコンテナー(「親」)としての役割を果たします。全てのメニューコマンドの構造をプログラム的に取得するには、.再帰的な関数を作成する必要があります。

```
public void GetStyleVisionMenus()
{
    // Get the main menu from the StyleVision ActiveX control assigned to the current
form
    StyleVisionControlLib.StyleVisionCommand mainMenu =
this.axStyleVisionControl1.MainMenu;

    // Loop through entries of the main menu (e.g. File, Edit, etc.)
    for (int i = 0; i < mainMenu.SubCommands.Count; i++)
    {
      StyleVisionControlLib.StyleVisionCommand menu = mainMenu.SubCommands[i];
```

```
      Console.WriteLine("{0} menu has {1} children items (including separators)",
menu.Label.Replace("&", ""), menu.SubCommands.Count);
    }
}
```

C# サンプル

17.3.7.3.1.8 Toolbars

プロパティ: `Toolbars` を Commands として (読み取り専用)

ディスパッチ ID : 1005

詳細:
このプロパティは、StyleVisionControl ツールバーの構造の情報を `Command` オブジェクトとして与えます。`Command` オブジェクトには、StyleVision の使用することのできる全てのツールバーが含まれています。ツールバーにアクセスするには、`Toolbars` プロパティの `SubCommands` プロパティを使用します。各ツールバーは、`Command` オブジェクトでもあります。各ツールバーのために、コマンドを使用するために `SubCommands` プロパティを更に反復します (例えば、アプリケーションのツールバープログラム的に作成するためなどに このテクニックは使用される場合があります)。

```
public void GetStyleVisionToolbars()
{
    // Get the application toolbars from the StyleVision ActiveX control assigned to
the current form
    StyleVisionControlLib.StyleVisionCommands toolbars =
this.axStyleVisionControl1.Toolbars;

    // Iterate through all toolbars
    for (int i = 0; i < toolbars.Count; i++)
    {
      StyleVisionControlLib.StyleVisionCommand toolbar = toolbars[i];
      Console.WriteLine();
      Console.WriteLine("The toolbar \"{0}\" has the following commands:",
toolbar.Label);

      // Iterate through all commands of this toolbar
      for (int j = 0; j < toolbar.SubCommands.Count; j++)
      {
         StyleVisionControlLib.StyleVisionCommand cmd = toolbar.SubCommands[j];
         // Output only command objects that are not separators
         if (!cmd.IsSeparator)
         {
            Console.WriteLine("{0}, {1}, {2}", cmd.ID, cmd.Name, cmd.Label.Replace("&",
 ""));
         }
      }
    }
}
```

C# サンプル

(C) 2015-2021 Altova GmbH

17.3.7.3.2 メソッド

以下のメソッドが定義されます:

Open
Exec
QueryStatus

17.3.7.3.2.1 Exec

Method: Exec (nCmdID を long として) を boolean として

ディスパッチ ID : 6

詳細:
このメソッドは、ID nCmdID を持つ StyleVision コマンドを呼び出し、コマンドが実行可能な場合、メソッドは、true を返します。使用可能な全てのコマンドのリストを取得するには、CommandsList を使用してください。.コマンドの状態を取得するには、QueryStatus を使用してください。

17.3.7.3.2.2 Open

Method: Open (strFilePath を string として) を boolean として

ディスパッチ ID : 5

詳細:
メソッドの結果は、引数 strFilePath 内でパスされる拡張子により異なります。ファイル拡張子が .sps の場合、新規ドキュメントが開かれます。ファイル拡張子が .svp の場合、対応するプロジェクトが開かれます。異なるファイル拡張子がメソッドにパスされると、コントロールはファイルを新規のコンポーネントとしてアクティブなドキュメントにロードします。

ドキュメントレベルの統合モード内でコントロール使用する際、このメソッドを使用して、ドキュメントまたはプロジェクトをロードしないでください。代わりに、StyleVisionControlDocument.Open と StyleVisionControlPlaceHolder.OpenProject を使用してください。

17.3.7.3.2.3 QueryStatus

Method: QueryStatus (nCmdID を long として) を long として

ディスパッチ ID : 7

詳細:
QueryStatus は、nCmdID により指定されたコマンドのチェックされた/チェックが解除されたステータスを返します。このステータスはビットマスクとして返されます。

ビット	値	名前	意味
0	1	サポートされている	コマンドがサポートされている場合設定します。
1	2	有効化されている	コマンドが有効化されている場合設定します(実行可能)。

2	4	チェックされている コマンドがチェックされている場合設定します。

これは、QueryStatus が0 コマンドを返すと、ID は、有効な StyleVision コマンドとして認識されないことを意味します。QueryStatus が1 または5 の値を返すとコマンドは無効化されます。

17.3.7.3.3 Events

StyleVisionControl ActiveX コントロールは次の接続ポイントイベントを提供します:

OnUpdateCmdUI
OnOpenedOrFocused
OnCloseEditingWindow
OnFileChangedAlert
OnContextChanged

OnDocumentOpened
OnValidationWindowUpdated

17.3.7.3.3.1 OnCloseEditingWindow

イベント: OnCloseEditingWindow (i_strFilePath を String として) を boolean として

ディスパッチ ID: 1002

詳細:
StyleVision が既に開かれているドキュメントを閉じる場合、このイベントはトリガーされます。このイベントへの応答として、クライアントは *i_strFilePath* に関連するエディターウィンドウを閉じます。このイベントが *true* を返すと、クライアントがドキュメントを閉じたことを指します。特定の処理が必要とされておらず、StyleVisionControl がエディターを閉じようと試み、関連したドキュメントコントロールを破棄する場合、クライアント は、*false* を返すことができます。

17.3.7.3.3.2 OnContextChanged

イベント: OnContextChanged (i_strContextName を String として、i_bActive を bool として) を bool として

ディスパッチ ID: 1004

詳細:
このイベントは、is not used in StyleVision

17.3.7.3.3.3 OnDocumentOpened

イベント: OnDocumentOpened (objDocument を Document として)

ディスパッチ ID: 1

詳細:

(C) 2015-2021 Altova GmbH

このイベントはドキュメントが開かれる都度トリガーされます。引数 objDocument は、StyleVision 自動化インターフェイス、からの Document オブジェクトであり、ドキュメントの詳細をクエリするために使用することができ、または、追加オペレーションを行います。ドキュメントレベルで統合される場合、イベント StyleVisionControlDocument.OnDocumentOpened を代わりに使用することが奨励されます。

17.3.7.3.3.4 OnFileChangedAlert

イベント： OnFileChangedAlert (i_strFilePath を String として) を bool として

ディスパッチ ID： 1001

詳細：
StyleVisionControl を使用してファイルがロードされると、isこのイベントはトリガーされます。他のアプリケーションによりハードディスク上で変更されます。イベントを処理すると、クライアントは、true を返します。または、StyleVision が通常の方法で処理すると、false を返します。例えば、ユーザーに再ロードを促すなど。

17.3.7.3.3.5 OnLicenseProblem

イベント： OnLicenseProblem (i_strLicenseProblemText を String として)

ディスパッチ ID： 1005

詳細：
StyleVisionControl がこのコントロールのための有効なライセンスを使用できないことを検知すると、このイベントはトリガーされます。制限されているユーザーファイセンスの場合、コントロールが初期化されると発生する可能性があります。インテグレーターはこのイベントをしようして、このコントロールの機能へのアクセスを無効化することができます。このイベントから戻ると、コントロールは機能へのアクセスをブロックします（例、コントロール内の空のウィンドウを表示し、リクエストに対してエラーを返すなど）。

17.3.7.3.3.6 OnOpenedOrFocused

イベント： OnOpenedOrFocused (i_strFilePath を String として、i_bOpenWithThisControl を bool として)

ディスパッチ ID： 1000

詳細：
アプリケーションレベルで統合する場合、このイベントはクライアントにドキュメントが開かれたこと、またはStyleVision によりアクティブに設定されたことを通知します。

ドキュメントレベルで統合する場合、このイベントは、クライアントにドキュメントウィンドウをファイル i_strFilePath で開くように命令します。ファイルが既に開かれている場合、対応するドキュメントウィンドウをアクティブなウィンドウにしてください。

i_bOpenWithThisControl が true の場合、内部アクセスが必要とされるため、ドキュメントは、StyleVisionControl により開かれる必要があります。それ以外の場合、ファイルは、異なるエディターで開くことができます。

17.3.7.3.3.7 OnToolWindowUpdated

イベント: `OnToolWindowUpdated(pToolWnd` を long として)

ディスパッチ ID: 1006

詳細:
ツールウィンドウが更新されると、このイベントはトリガーされます。

17.3.7.3.3.8 OnUpdateCmdUI

イベント: `OnUpdateCmdUI ()`

ディスパッチ ID: 1003

詳細:
頻繁に呼び出され、`StyleVisionControl.QueryStatus` を使用して StyleVision コマンドの状況をチェックする良い機会をインテグレーターに与えます。このコールバック内で long オペレーションを実行しないでください。

17.3.7.3.3.9 OnValidationWindowUpdated

イベント: `OnValidationWindowUpdated` ()

ディスパッチ ID: 3

詳細:
新しい情報と共に検証出力ウィンドウが更新されると、このイベントはトリガーされます。

17.3.7.4 StyleVisionControlDocument

プロパティ:
Appearance
BorderStyle
Document
IsModified
Path
ReadOnly

メソッド:
Exec
New
Open
QueryStatus
Reload
Save
SaveAs

(C) 2015-2021 Altova GmbH

イベント:
OnDocumentOpened
OnDocumentClosed
OnModifiedFlagChanged
OnContextChanged
OnFileChangedAlert
OnActivate

StyleVisionControl がドキュメントレベルモード内で統合されている場合、各ドキュメントは、型 StyleVisionControlDocument のオブジェクト内で表示されています。StyleVisionControlDocument には、1度に1つのドキュメントのみが含まれていますが、場合によっては異なるファイルを表示するために使用される場合があります。

このオブジェクトは完全な ActiveX コントロールです。

17.3.7.4.1 プロパティ

以下のプロパティが定義されています:

ReadOnly
IsModified
Path
Appearance
BorderStyle

StyleVisionAPI へのアクセス:
Document

17.3.7.4.1.1 Appearance

プロパティ: Appearance を short として

ディスパッチ ID : -520

詳細:
0 に対して等しくない値は、ドキュメント コントロールの周りにクライアントエッジを表示します。デフォルトの値は 0 です。

17.3.7.4.1.2 BorderStyle

プロパティ: BorderStyle を short として

ディスパッチ ID : -504

詳細:
1 の値は、薄いボーダー持つコントロールを表示します。デフォルトの値は 0 です。

17.3.7.4.1.3 Document

プロパティ: `Document` をドキュメントとして

ディスパッチ ID : 1

詳細:
`Document` プロパティは、StyleVision 自動サーバーAPI の`Document` オブジェクト へのアクセスを与えます。このインターフェイスは、コントロール内にロードされるドキュメントと使用することのできる追加機能を与えます。プロパティは読み取り専用です

17.3.7.4.1.4 IsModified

プロパティ: `IsModified` を boolean として(読み取り専用)

ディスパッチ ID : 1006

詳細:
ドキュメントコンテンツが最後に開かれてから変更された場合、`IsModified` は *true* を返します。再ロードし、オペレーションを保存します。それ以外の場合は、`false` を返します。

17.3.7.4.1.5 Path

プロパティ: `Path` を string として

ディスパッチ ID : 1005

詳細:
コントロール内にロードされたドキュメントのフルパス名に設定、または、取得します。

17.3.7.4.1.6 ReadOnly

プロパティ: `ReadOnly` を boolean として

ディスパッチ ID : 1007

詳細:
このプロパティを使用すると、ドキュメントの読み取り専用モードをオンとオフに切り替えることができます。`ReadOnly` が `true` の場合、変更はできないことに注意してください。

(C) 2015-2021 Altova GmbH

17.3.7.4.2 メソッド

以下のメソッドが定義されています:

ドキュメント ハンドリング:
New
Open
Reload
Save
SaveAs

コマンド ハンドリング::
Exec
QueryStatus

17.3.7.4.2.1 Exec

メソッド: Exec (nCmdID を long として) を boolean として

ディスパッチ ID : 8

詳細:
このメソッドは、ID nCmdID を持つ StyleVision コマンドを呼び出し、コマンドが実行可能な場合、メソッドは、true を返します。現在アクティブなドキュメントがアプリケーションで使用できる場合のみ、このメソッドを呼び出すことができます。

メニュー構造に従いコマンドを整理するには、StyleVisionControl の MainMenu プロパティを使用します。ツールバーコマンドを取得するには、StyleVisionControl の Toolbars プロパティを使用します。

17.3.7.4.2.2 New

メソッド: New () を boolean として

ディスパッチ ID : 1000

詳細:
このメソッドはコントロール内で新規ドキュメントを初期化します。

17.3.7.4.2.3 Open

メソッド: Open (strFileName を string として) を boolean として

ディスパッチ ID : 1001

詳細:
Open は、ファイル strFileName を新規のドキュメントとして、コントロールにロードします。

17.3.7.4.2.4 QueryStatus

メソッド: QueryStatus (nCmdID を long として) を long として
ディスパッチ ID: 9

詳細:
QueryStatus は、nCmdID により指定されたコマンドのチェックされた/チェックが解除されたステータスを返します。このステータスはビットマスクとして返されます。

ビット	値	名前	意味
0	1	サポートされている	コマンドがサポートされている場合設定します。
1	2	有効化されている	コマンドが有効化されている場合設定します(実行可能)。
2	4	チェックされている	コマンドがチェックされている場合設定します。

これは、QueryStatus が 0 コマンドを返すと、ID は、有効な StyleVision コマンドとして認識されないことを意味します。QueryStatus が 1 または 5 の値を返すとコマンドは無効化されます。アプリケーション内で現在アクティブなドキュメントが存在する場合、クライアントは、ドキュメントコントロールの QueryStatus メソッドを呼び出します。

17.3.7.4.2.5 Reload

メソッド: Reload () を boolean として

ディスパッチ ID: 1002

詳細:
Reload ファイルシステムからドキュメントコンテンツを更新する。

17.3.7.4.2.6 Save

メソッド: Save () を boolean として

ディスパッチ ID: 1003

詳細:
Save は、現在のドキュメントをロケーション Path に保存する。

17.3.7.4.2.7 SaveAs

メソッド: SaveAs (strFileName を string として) を boolean として

ディスパッチ ID: 1004

(C) 2015-2021 Altova GmbH

詳細:
SaveAs は Path を *strFileName* に設定し、ドキュメントをこの場所に保存します。

17.3.7.4.3 Events

StyleVisionControlDocument ActiveX コントロールは次の接続ポイントイベントを提供します:

OnDocumentOpened
OnDocumentClosed
OnModifiedFlagChanged
OnContextChanged
OnFileChangedAlert
OnActivate
OnSetEditorTitle

17.3.7.4.3.1 OnActivate

イベント: OnActivate ()

ディスパッチ ID: *1005*

詳細:
ドキュメントコントロールが有効化されると、フォーカスのある、ユーザー入力として使用することのできるこのイベントがトリガーされます。

17.3.7.4.3.2 OnContextChanged

イベント: OnContextChanged (i_strContextName を String として、i_bActive を bool として) を bool として

ディスパッチ ID: *1004*

詳細: なし

17.3.7.4.3.3 OnDocumentClosed

イベント: OnDocumentClosed (objDocument を Document として)

ディスパッチ ID: *1001*

詳細:
このイベントはドキュメントがこのコントロールにロードされると閉じられます。引数 objDocument は、StyleVision 自動化インターフェイスからの Document オブジェクトです、で慎重に使用してください。

17.3.7.4.3.4 OnDocumentOpened

イベント: `OnDocumentOpened` (`objDocument` を Document として)

ディスパッチ ID: 1000

詳細:
このイベントは、このコントロール内でドキュメントが開かれるとトリガーされます。引数 `objDocument` は、StyleVision 自動化インターフェイスからの `Document` オブジェクトです。ドキュメントの詳細をクエリするために使用することができ、または、追加オペレーションを行います。

17.3.7.4.3.5 OnDocumentSaveAs

イベント: `OnContextDocumentSaveAs` (`i_strFileName` を String として)

ディスパッチ ID: 1007

詳細:
このドキュメントが新しい名前で内部で保存される場合このイベントがトリガーされます。

17.3.7.4.3.6 OnFileChangedAlert

イベント: `OnFileChangedAlert` () を bool として

ディスパッチ ID: 1003

詳細:
このドキュメントにロードされると、このイベントはトリガーされます。コントロールはハードディスク上で他のアプリケーションに変更されます。イベントを処理すると、クライアントは、true を返します。または、StyleVision が通常の方法で処理すると、false を返します。例えば、ユーザーに再ロードを促すなど。

17.3.7.4.3.7 OnModifiedFlagChanged

イベント: `OnModifiedFlagChanged` (`i_bIsModified` を boolean として)

ディスパッチ ID: 1002

詳細:
このイベントはドキュメントの状態を、変更された、および、変更されていない状態に切り替えると、トリガーされます。ドキュメントコンテンツが元のコンテンツと異なる場合、パラメーター *i_bIsModifed* is *true* それ以外の場合は、`false` を返します。

17.3.7.4.3.8 OnSetEditorTitle

イベント: OnSetEditorTitle ()

(C) 2015-2021 Altova GmbH

ディスパッチ ID : 1006

詳細:
このイベントは、含まれているドキュメントが内部で名前を与えられると挙げられます。

17.3.7.5 StyleVisionControlPlaceHolder

すべてのプロジェクト プレースホルダーウィンドウに使用することのできるプロパティ:
PlaceholderWindowID

プロジェクト プレースホルダーウィンドウのためのプロパティ:
Project

プロジェクト プレースホルダーウィンドウのためのメソッド:
OpenProject
CloseProject

StyleVisionControlPlaceHolder コントロールは、概要、ライブラリまたはプロジェクトウィンドウなどの追加の StyleVision ウィンドウを表示するために使用されます。他の ActiveX コントロールと同様に使用することができ、クライアントアプリケーション内に配置することができます。

17.3.7.5.1 プロパティ

以下のプロパティが定義されています:

PlaceholderWindowID

StyleVisionAPI へのアクセス:
Project

17.3.7.5.1.1 Label

プロパティ: Label を String (読み取り専用)として

ディスパッチ ID : 1001

詳細:
このプロパティはプレースホルダのタイトルへのアクセスを与えます。プロパティは読み取り専用です

17.3.7.5.1.2 PlaceholderWindowID

プロパティ: PlaceholderWindowID を StyleVisionControlPlaceholderWindow として

ディスパッチ ID : 1

詳細:

このプロパティを使用すると、コントロールのクライアントエリア内で表示されるStyleVision ウィンドウが通知されます。PlaceholderWindowID は、StyleVisionControlPlaceholderWindow 列挙の有効な値に設定することができます。コントロールはすぐに状態を変更し、すぐに新規のStyleVision ウィンドウを表示します。

17.3.7.5.1.3 Project

プロパティ: Project Project として(読み取り専用)

ディスパッチ ID: 2

詳細:
Project プロパティは、StyleVision 自動サーバーAPI のProject オブジェクト へのアクセスを与えます。このインターフェイスは、コントロールへロードされるプロジェクトに使用される追加機能を与えます。プレースホルダウィンドウが StyleVisionXProjectWindow (=3)の値を持つPlaceholderWindowID の場合のみ、プロパティは有効なプロジェクトインターフェイスを返します。プロパティは読み取り専用です

17.3.7.5.2 メソッド

以下のメソッドが定義されています:

OpenProject
CloseProject

17.3.7.5.2.1 OpenProject

メソッド: OpenProject (strFileName を string として) を boolean として

ディスパッチ ID: 3

詳細:
OpenProject はstrFileName を新規のプロジェクトとしてコントロールにロードします。プレースホルダウィンドウが XMLSpyXProjectWindow (=3)と異なるPlaceholderWindowID を持つ場合、メソッドは失敗します。

17.3.7.5.2.2 CloseProject

メソッド: CloseProject ()

ディスパッチ ID: 4

詳細:
CloseProject はプロジェクトにロードされたコントロールを閉じます。プレースホルダウィンドウがXMLSpyXProjectWindow (=3)と異なるPlaceholderWindowID を持つ場合、メソッドは失敗します。

(C) 2015-2021 Altova GmbH

17.3.7.5.3 イベント

StyleVisionControlPlaceholder ActiveX コントロールは次の接続ポイントイベントを提供します:

OnModifiedFlagChanged

17.3.7.5.3.1 OnModifiedFlagChanged

イベント: `OnModifiedFlagChanged(i_bIsModified` を boolean として)

ディスパッチ ID: 1

詳細:
このイベントは StyleVisionXProjectWindow (=3) の `PlaceholderWindowID` を持つプレースホルダコントロールのためのみにトリガーされます。プロジェクトのコンテンツが変更された、または、変更されていない状態の間で変更があった場合、イベントは実行されます。プロジェクトコンテンツが元のコンテンツと異なる場合、パラメーター *i_bIsModifed* は *true* を返し、それ以外の場合は、`false` を返します。

17.3.7.5.3.2 OnSetLabel

イベント: `OnSetLabel(i_strNewLabel` を string として)

ディスパッチ ID: 1000

詳細:
プレースホルダウィンドウのタイトルが変更されると、挙げられます。

17.3.7.6 列挙

以下の列挙が定義されています:

ICActiveXIntegrationLevel
StyleVisionControlPlaceholderWindow

17.3.7.6.1 ICActiveXIntegrationLevel

StyleVisionControl の IntegrationLevel プロパティのための可能な値。

```
ICActiveXIntegrationOnApplicationLevel  = 0
ICActiveXIntegrationOnDocumentLevel     = 1
```

17.3.7.6.2 StyleVisionControlPlaceholderWindow

この列挙にはサポートされる追加 StyleVision ウィンドウのリストが含まれています。

```
StyleVisionControlNoToolWnd             = -1
StyleVisionControlProjectWnd            = 0
StyleVisionControlDesignOverviewWnd     = 1
StyleVisionControlSchemaSourcesWnd      = 2
StyleVisionControlDesignTreeWnd         = 3
StyleVisionControlStyleRepositoryWnd    = 4
StyleVisionControlContextPropertiesWnd  = 5
StyleVisionControlContextStylesWnd      = 6
StyleVisionControlMessageWnd            = 7
StyleVisionControlDBQueryWnd            = 8
```


(C) 2015-2021 Altova GmbH

18 付録

この付録には(i) StyleVision にて使用される XSLT エンジンの情報、(ii) データベース データ型から XML スキーマ データ型への変換情報、(iii) StyleVision の技術情報、(iv) StyleVision のライセンス情報について記述されます。各情報は以下のサブセクションに含められます:

XSLT エンジンの情報

StyleVision にて出力を生成するために使用される Altova XSLT エンジンの実装に関する情報が記されます。

- Altova XSLT 1.0 エンジン
- Altova XSLT 2.0 エンジン
- Altova XSLT 3.0 エンジン
- XSLT と XPath/XQuery 関数

DatatypesDB2XSD

DB フィールドが XML ノードへ変換されると、DB データ型が XML スキーマ データ型へ変換されます。この付録では以下のソース データベースに対するマッピングが記されます。

- MS Access
- MS SQL Server
- MySQL
- Oracle
- ODBC
- ADO
- Sybase

技術情報

StyleVision に関する技術的な情報が記述されます。

- OS とメモリの要件
- Altova XML パーサー
- Altova XSLT と XQuery エンジン
- Unicode のサポート
- インターネットの接続

ライセンス情報

StyleVision の配布とライセンスに関する情報が含まれます。

- 電子的なソフトウェアの配布
- ライセンス メータリング
- 著作権
- 使用許諾契約書

18.1 XSLT および XQuery エンジンに関する情報

StyleVision のXSLT およびXQuery エンジンは、W 3C 仕様に従っています、ですから、XMLSpy の以前のバージョン内のAltova エンジンより先厳密です。この結果、以前のエンジンで無視されていた小さなエラーが、StyleVision によりエラーとして挙げられます。

例えば

- パス演算子の結果がノードと非-ノードを両方含む場合、型エラー(`err:XPTY0018`) です。
- パス式 `E1/E2` 内の`E1` がノードのシーケンスを評価しない場合、型エラー(`err:XPTY0019`) です。

この種類のエラーが発生した場合、XSLT/XQuery ドキュメントまたはインスタンスドキュメントを必要に応じて修正してください。

このセクションは、エンジンの実装固有の機能を仕様別に整理して説明します。

- XSLT 1.0
- XSLT 2.0
- XSLT 3.0
- XQuery 1.0
- XQuery 3.1

18.1.1 XSLT 1.0

StyleVision のXSLT 1.0 エンジン は、W orld W ide W eb Consortium(ワールド・ワイド・ウェブ・コンソーシアム) (W 3C) の1999 年 11 月 16 日版のXSLT 1.0 勧告 および1999 年 11 月 16 日版のXPath 1.0 勧告 に準拠します。実装に関しての以下の情報にご注意してください。

実装についての注意点

`xsl:output` の`method` 属性がHTMLに設定された場合、または、がHTML 出力 デフォルトで選択されている場合、XML または XSLT ファイル内の特殊文字はHTML ドキュメントにHTML 文字参照として出力内に挿入されます。例えば、文字 U+00A0 (ブレーク無しのスペースのための16 進数レファレンス) がHTML コード内に文字の参照 (` ` or ` `) 、または、エンティティ参照 ` ` として挿入されます

18.1.2 XSLT 2.0

このセクション:

- エンジン適合性
- 下位互換性
- 名前空間
- スキーマ認識
- 実装固有の振る舞い

(C) 2015-2021 Altova GmbH

適合性

StyleVision のXSLT 2.0 エンジンは、W orld W ide W eb Consortium（ワールド・ワイド・ウェブ・コンソーシアム）（W 3C）の2007 年 1 月 23 日版のXSLT 2.0 勧告 および2010 年 12 月 14 日版のXPath 2.0 勧告に準拠します。

下位互換性

XSLT 2.0 エンジン は下位互換性を有します。XSLT 2.0 エンジンの 下位互換性が有効になるのは、 XSLT 1.0 スタイルシートを処理するためにXSLT 2.0 エンジン が使用される際です。XSLT 1.0 エンジン と下位互換性を持つXSLT 2.0 エンジンにより作成される出力に違いがあるかもしれないことに注意してください。

名前空間

XSLT 2.0 スタイルシートは、 XSLT 2.0. プレフィックス内で使用することのできる型コンストラクタ および関数を使用するため、以下の名前空間を宣言する必要があります。 下のリストは通常使用されるリスト です。希望する場合は、代替プレフィックスを使用することもできます。

名前空間	プレフィックス	名前空間 URI
XML スキーマ型	`xs:`	`http://www.w3.org/2001/XMLSchema`
XPath 2.0 関数	`fn:`	`http://www.w3.org/2005/xpath-functions`

通常これらの名前空間は、以下のリスト で表示されるように`xsl:` スタイルシートまたは、 `xsl:transform` 要素で宣言されます:

```
<xsl:スタイルシート 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:スタイルシート>
```

次の点に注意してください:

- XSLT 2.0 エンジンは、（上のテーブルでリストされている） XPath 2.0 およびXQuery 1.0 関数 名前空間 を**デフォルト の関数名前空間**として使用します。 XPath 2.0 およびXSLT 2.0 関数をプレフィックス無しでスタイルシート内で使用することができます。XPath 2.0 関数 名前空間 をスタイルシート内でプレフィックスと共に宣言すると、 割り当てられた宣言内でプレフィックスを追加して使用することができます。
- XML スキーマ 名前空間から型コンストラクタと型を使用する場合、名前空間 宣言内で使用された、プレフィックスを使用して型コンストラクタを呼び出さなければなりません（例えば、 `xs:date`）。
- XPath 2.0 関数の一部は、XML スキーマ データ型と同じ名前を保有します。 例えば、 XPath 関数 `fn:string` および `fn:boolean` のためには、同じロケーション名: `xs:string` および`xs:boolean` を持つ、XML スキーマ データ型 が存在します。ですから、 XPath 式 `string('Hello')`を使用する場合、 式は、 `xs:string('Hello')` ではなくて `fn:string('Hello')` として検証します。

スキーマ認識

XSLT 2.0 エンジンは、スキーマを認識します。ですから、ユーザー定義 スキーマ型 および`xsl:validate` 命令を使用することができます。

実装固有の振る舞い

以下は、XSLT 2.0 エンジンが、特定のXSLT 2.0 関数の振る舞いの実装-特定のアスペクトをどのように扱うかの説明です。

xsl:result-document
追加してサポートされるエンコードは以下の通りです（Altova-固有）: `x-base16tobinary` および `x-base64tobinary`.

function-available
インスコープ関数の使用をテストする関数（XSLT、XPath、および拡張関数）。

unparsed-text
`href` 属性は、以下を受け入れます（i）ベース-uri フォルダー内のファイルの相対パス、および（ii） 相対パスを持つまたは持たない `file://` プロトコール。追加してサポートされるエンコードは以下の通りです（Altova-固有）: `x-binarytobase16` および `x-binarytobase64`.

unparsed-text-available
`href` 属性は、以下を受け入れます（i）ベース-uri フォルダー内のファイルの絶対パス、および（ii）絶対パスを持つまたは持たない `file://` プロトコール。追加してサポートされるエンコードは以下の通りです（Altova-固有）: `x-binarytobase16` および `x-binarytobase64`.

メモ RaptorXML の先行製品であるAltovaXML で実装されていた以下のエンコード値は使用しないでください: `base16tobinary`, `base64tobinary`, `binarytobase16` and `binarytobase64`.

18.1.3 XSLT 3.0

StyleVision のXSLT 3.0 エンジンは、W orld W ide W eb Consortium（ワールド・ワイド・ウェブ・コンソーシアム）（W 3C）の2017 年 6 月 8 日版 XSLT 3.0 勧告候補 および2017 年 3 月 21 日版 XPath 3.1 勧告候補 に準拠します。

XSLT 3.0 エンジンは、XSLT 2.0 エンジンと同様の実装固有の機能を搭載しています。更に、以下のXSLT 3.0 機能へのサポートを含みます。更に次の一連の新規のXSLT 3.0 機能をサポートします: XPath/XQuery 3.1 関数とオペレーターとXPath 3.1 仕様。

メモ 任意のストリーミングの機能は現在サポートされていません。streamable 属性の値に関係なく、ドキュメント全体がメモリにロードされ、使用することのできるメモリが十分な場合は処理されます。64-ビット アプリでは、これは問題になりません。thisメモリが問題の場合は、システムに解決策を追加する必要があります。

18.1.4 XQuery 1.0

このセクション:

- エンジン適合性
- スキーマ認識
- エンコード
- 名前空間
- XML ソースと検証
- 静的および動的な型のチェック
- ライブラリモジュール
- 外部関数

- 照合順序
- 数値データの精度
- XQuery 命令サポート

適合性

StyleVision のXQuery 1.0 エンジンは、W orld W ide W eb Consortium（ワールド・ワイド・ウェブ・コンソーシアム）（W 3C）の 2010 年 12 月 14 日版のXQuery 1.0 勧告に準拠します。XQuery 標準は、多数の機能の実装についての裁量を提供します。下には、XQuery 1.0 エンジンがどのようにこれらの機能を実装するかについて説明するリストが下に挙げられています。

スキーマ認識

XQuery 1.0 エンジン はスキーマを認識します。

エンコード

UTF-8 およびUTF-16 文字のエンコードは、サポートされています。

名前空間

以下の名前空間 URI と関連するバインドは定義済みです。

名前空間	プレフィックス	名前空間 URI
XML スキーマ型	xs:	http://www.w3.org/2001/XMLSchema
スキーマインスタンス	xsi:	http://www.w3.org/2001/XMLSchema-instance
内蔵の関数	fn:	http://www.w3.org/2005/xpath-functions
Local 関数	local:	http://www.w3.org/2005/xquery-local-functions

次の点に注意してください：

- XQuery 1.0 エンジンは、上にリストされたプレフィックスを名前空間に対応するバウンドとして認識します。
- Since the 上にリストされた内蔵の関数 名前空間は、XQuery内のデフォルトの関数です。内蔵の関数 が呼び出される際、名前空間、fn: プレフィックス を使用する必要はありません。（例えば、string("Hello") がfn:string 関数を呼び出す場合。）しかし、プレフィックス fn: は、クエリプロローグ内で名前空間を宣言することなく内蔵の関数を呼び出す時に使用することができます。（サンプル: fn:string("Hello")）.
- クエリプロローグ内でdefault function 名前空間 式を宣言することにより、デフォルトの関数 名前空間をすることにより変更することができます。
- XML スキーマ 名前空間空の型を使用する場合、プレフィックス xs: は、名前空間を明確に宣言することなく、また、これらのプレフィックスをクエリプロローグ内でバインドすることなく使用することができます。（サンプル: xs:date および xs:yearMonthDuration。）XML スキーマ 名前空間のために、他のプレフィックスを使用する場合は、クエリプロローグ内で明確に宣言されている必要があります。（サンプル: declare 名前空間 alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04").）
- untypedAtomic, dayTimeDuration, およびyearMonthDuration データ型 が23 January 2007 のCR 共に、XPath データ型 名前空間からXML スキーマ 名前空間へ移動されていることに注意してください。ですから以下となります: xs:yearMonthDuration。

関数のための名前空間、型コンストラクタ、ノード テスト、が間違って割り当てられている場合、エラーが発生します。しかし、一部の関数はスキーマデータ型と同じ名前を持つことに注意してください。 例: fn:string および fn:boolean。(xs:string および xs:boolean は宣言されています) 名前空間 プレフィックス、関数または 型コンストラクタが使用されるか決定します。

XML ソースドキュメントと検証

XQuery 1.0 エンジンを使用して、実行される XQuery ドキュメント内の XML ドキュメントは、整形式である必要があります。しかし、XML スキーマに従い有効である必要はありません。XML ファイルが外部スキーマと関連付けられ、また有効な場合、ポスト スキーマ検証情報が XML データのために生成され、クエリ検証のために使用されます。

静的および動的な型のチェック

静的分析フェーズは、外部レファレンスの存在（例、モジュールのため）、呼び出された関数と変数が定義済みであるか、など構文などのクエリのアスペクトをチェックします。静的分析フェーズでエラーが検知されると、実行は停止されます。

クエリが実際に作動中にランタイム中に動的な型チェックは実行されます。型がオペレーションの必要条件と整合性を持たない場合、エラーが報告されます。例えば、式 `xs:string("1") + 1` は、エラーを返します。型 `xs:string` のオペランドを足し算のオペレーションが実行できないためです。

ライブラリモジュール

ライブラリモジュールは、再利用のため関数と変数を保管します。XQuery 1.0 エンジンは単一の外部 XQuery ファイルに保管されているモジュールをサポートします。このようなモジュール ファイルはプロローグにターゲット名前空間に関連するモジュール宣言を含む必要があります。 以下はモジュールサンプルです:

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns:webaddress() { "http://www.altova.com" };
```

すべての関数および変数は、モジュールに関連した名前空間に属するモジュール内で宣言されています。モジュールはクエリプロローグ内の `import module` ステートメントを使用して XQuery ファイルにインポートする際に使用されます。`import module` ステートメントは、ライブラリモジュールファイル内で直接宣言された、関数と変数のみをインポートします。例:

```
import module namespace modlib = "urn:module-library" at "modulefilename.xq";
if        ($modlib:company = "Altova")
then       modlib:webaddress()
else       error("No match found.")
```

外部関数

外部関数 は、サポートされていません。例えば、`external` キーワードを仕様する式など。以下参照:

```
declare function hoo($param as xs:integer) as xs:string external;
```

照合順序

デフォルトの照合順序は、Unicode コードポイントをベースにした文字列を比較する Unicode-コードポイント照合順序です。その他にサポートされる照合順序は、ICU 照合順序 はここ にリストされるとおりです。特定の照合順序を使用する場合、サポートされる照合順序のリスト 内に与えられているとおり URI を提供します。`fn:max` と `fn:min` 関数を含む文字列の比較は、指定された照合順序により行われます。 照合順序オプションが指定されていない場合、デフォルトの Unicode-コードポイント照合順序 が使用されます。

数値データの精度

- `xs:integer` データ型には任意の精度があり、表記できる桁数に制限はありません。
- `xs:decimal` データ型には小数点の後に 20 桁の制限があります。
- `xs:float` と `xs:double` データ型には 15 桁の精度の制限があります。

(C) 2015-2021 Altova GmbH

XQuery 命令サポート

`Pragma` 命令 は、サポートされていません。発生した場合、無視されフォールバックの式が検証されます。

18.1.5 XQuery 3.1

StyleVision のXQuery 3.1 エンジンは、World Wide Web Consortium（ワールド・ワイド・ウェブ・コンソーシアム）（W3C）の 2017 年 3 月 21 日版のXQuery 3.1 候補勧告に準拠し、またXPath およびXQuery 関数 3.1.へのサポートを含みます。XQuery 3.1 仕様は、3.0 仕様のスーパーセットです。XQuery 3.1 エンジンは、ですかXQuery 3.0 機能をサポートします。

実装固有の特性はXQuery 1.0同様です。

18.2 XSLT と XPath/XQuery 関数

このセクションでは、XPath および/または XQuery 式で使用することのできる、Altova 拡張関数と他の拡張関数をリストします。Altova 拡張関数は Altova の XSLT および XQuery エンジンで使用することができ、W3C 標準で定義された関数ライブラリで使用することのできる機能に追加して機能を提供します。

一般的な情報

以下の一般的な情報に注意してください：

- W3C 仕様により定義されているコア関数ライブラリの関数は、関数の呼び出しにプレフィックスは必要ありません。これは、XSLT および XQuery エンジンが、XPath/XQuery 関数仕様で指定されている `http://www.w3.org/2005/xpath-functions` プレフィックス無しの関数をデフォルト関数の名前空間に属するものとして読み込むためです。
- 関数において、各アイテムが引数となるようなシーケンスが期待されており、2つ以上のアイテムがシーケンスにより呼び出された場合、エラーが返されます。
- 全ての比較は Unicode コードポイントコレクションを使用することで行われます。
- QName の結果は `[prefix:]localname` という形式でシリアル化されます。

xs:decimal の精度

精度とは、数値内にある桁数のことで、仕様では少なくとも18桁が求められます。`xs:decimal` 型に結果が収められる除算の場合、端数処理を行うことなく精度は小数点以下の19桁になります。

黙示的なタイムゾーン

2つの `date`、`time`、または `dateTime` 値を比較する場合、比較する値のタイムゾーンを明らかにする必要があります。値の中にタイムゾーンが明示的に与えられていない場合、黙示的なタイムゾーンが使用されます。黙示的なタイムゾーンはシステムクロックから取得され `implicit-timezone()` 関数によりその値をチェックすることができます。

照合順序

デフォルトの照合順序は、Unicode コードポイントをベースに文字列を比較する Unicode コードポイント照合順序です。エンジンは Unicode 照合アルゴリズムを使用しています。他のサポートされる照合順序は下にリストされる ICU 照合順序 です。使用するには、サポートされる照合順序のリストの URI を提供してください（*下のテーブル*）。`max` と `min` 関数を含む、文字列の比較は、指定された照合順序に沿って行われます。照合順序オプションが指定されていない場合、デフォルトの Unicode コードポイント照合順序が使用されます。

言語	URI
`da`: デンマーク語	`da_DK`
`de`: ドイツ語	`de_AT, de_BE, de_CH, de_DE, de_LI, de_LU`
`en`: 英語	`en_AS, en_AU, en_BB, en_BE, en_BM, en_BW, en_BZ, en_CA, en_GB, en_GU, en_HK, en_IE, en_IN, en_JM, en_MH, en_MP, en_MT, en_MU, en_NA, en_NZ, en_PH, en_PK, en_SG, en_TT, en_UM, en_US, en_VI, en_ZA, en_ZW`
`es`: スペイン語	`es_419, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GQ, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE`
`fr`: フランス語	`fr_BE, fr_BF, fr_BI, fr_BJ, fr_BL, fr_CA, fr_CD, fr_CF, fr_CG, fr_CH, fr_CI, fr_CM, fr_DJ, fr_FR, fr_GA, fr_GN, fr_GP, fr_GQ, fr_KM, fr_LU, fr_MC, fr_MF, fr_MG, fr_ML, fr_MQ, fr_NE, fr_RE, fr_RW, fr_SN, fr_TD, fr_TG`

(C) 2015-2021 Altova GmbH

it: イタリア語	it_CH, it_IT
ja: 日本語	ja_JP
nb: ノルウェー語 (ブークモール)	nb_NO
nl: オランダ語	nl_AW, nl_BE, nl_NL
nn: ノルウェー語 (ニーノシュク)	nn_NO
pt: ポルトガル語	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: ロシア語	ru_MD, ru_RU, ru_UA
sv: スウェーデン語	sv_FI, sv_SE

名前空間軸
名前空間軸はXPath 2.0 にて廃止されましたが、名前空間軸の使用はサポートされています。XPath 2.0 メカニズムにより名前空間情報へアクセスするには、`in-scope-prefixes()`、`namespace-uri()`、`namespace-uri-for-prefix(` 関数を使用してください。

18.2.1 Altova 拡張関数

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間、`http://www.altova.com/xslt-extensions`** に収められており、**`altova:`** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

W 3C のXPath/XQuery 関数仕様で定義された関数は、以下で使用することができます: (i) XSLT 子アンテキスト内のXPath 式と、(ii) XQuery 文書内のXQuery 式。このドキュメントでは、前者(XSLT 内のXPath) のコンテキストで使用することのできる関数を、**XP** シンボルと共に表示し、と称します。後者(XQuery) で使用することのできる関数はの**XQ** シンボルと共に表示され、XQuery 関数と共に作業することができます。W 3C のXSLT 仕様は、XPath/XQuery 関数の仕様ではなく、XSLT 文書内のXPath 式でも使用することのできる関数を定義します。これらの関数は、**XSLT** シンボルと共に表示され、XSLT 関数と称されます。関数を使用することのできるXPath/XQuery およびXSLT のバージョンは、関数の詳細に記載されています (*下のシンボルを参照してください*)。
XPath/XQuery およびXSLT 関数ライブラリからの関数は、プレフィックス無しでリストされています。Altova 拡張関数などの、他のライブラリからの関数はプレフィックスと共にリストされています。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

XSLT 関数

XSLT 関数はXSLT 2.0 の`current-group()` や`key()` 関数と同様に、XSLT コンテキストにて使用することができます。(例えば、XQuery コンテキストなどの) 非-XSLT コンテキストでは使用することができません。XBRL に対するXSLT 関数は、XBRL をサポートするエディションのAltova 製品でのみ使用することができます。

XPath/XQuery 関数

XPath/XQuery 関数は、 XSLT コンテキスト、 XQuery 関数のXPath 式で使用することができます:

- 日付/時刻
- 位置情報
- イメージに関連した
- 数値
- シーケンス
- 文字列
- その他

バーコード関数

Altova バーコード拡張関数はバーコードを生成し、スタイルシートを介して生成された出力に配置することができます。

18.2.1.1 XSLT 関数

XSLT 拡張関数 はXSLT コンテキスト内のXPath 式にて使用することができます。(例えば、XQuery コンテキストなどの) 非-XSLT コンテキストでは使用することができません。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**`http://www.altova.com/xslt-extensions`** に収められており、**`altova:`** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

標準関数

▼ distinct-nodes [altova:]

`altova:distinct-nodes(`*`node()*`***`)`** を **`node()*`** とする XSLT1 XSLT2 XSLT3
入力として1つ以上のノードを必要とし、同じセットから重複した値を持つノードを除いたノードを返します。XPath/XQuery 関数 `fn:deep-equal` を使用して比較を行うことができます。

⊟ *サンプル*

- **`altova:altova:distinct-nodes`**`(country)` は重複した値を持つものを除く、全ての子 `country` ノード返します。

evaluate [altova:]

altova:evaluate(XPathExpression *as xs:string*[, **ValueOf$p1, ... ValueOf$pN])** **XSLT1** **XSLT2** **XSLT3**

XPath 式を必要とし、必須引数として文字列をパスします。評価された式の出力を返します。例えば **altova:evaluate('//Name[1]')** は、ドキュメント内の最初の Name 要素のコンテンツを返します。式 //Name[1] は、一重引用符を使用することにより、文字列としてパスされます。

altova:evaluate 関数は、オプションとして追加の引数を持つことができます。これらの引数は、p1, p2, p3... pN の名前を持つスコープ内の変数の値です。使用に関して以下の点に注意してください：(i) 変数は、X が整数である箇所のフォーム pX の名前と共に定義される必要があります。(ii) altova:evaluate 関数の引数は、(*上の署名参照*) 2 番目の引数からは、数値順の変数のシーケンスに対応した引数のシーケンス変数の値を与えます: p1 to pN: 第 2 引数は変数 p1 の値で、第 3 引数は、変数 p2 の値です。(iii) 変数の値は型 item* である必要があります。

サンプル

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 'hi')" />
```

outputs "hi 20 10"

上のリストに関して、以下の点に注意してください：

- altova:evaluate 式の第 2 引数は、変数 $p1 に割り当てられた値で、第三の引数は変数 $p2 に割り当てられた値です。
- 関数の第 4 番目の引数は、引用符による囲いで表示された文字列の値です。
- xs:variable 要素の select 属性は、XPath 式を提供します。この式は xs:string の型である必要があり、一重引用符で囲まれています。

変数の使用方法を更に説明するサンプル

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
  ```
  最初の Name *要素の出力値*

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, '//Name[1]')" />
  ```
  Outputs "//Name[1]"

altova:evaluate() 拡張関数は、XSLT スタイルシート内の XPath 式が動的に評価される必要のあるシチュエーションで役に立ちます。例えば、ユーザーが並べ替えの必要条件をリクエストする場合、このシチュエーションは属性 UserReq/@sortkey に保管されます。スタイルシートでは、以下の式が使用できます: <xsl:sort select="**altova:evaluate(../UserReq/@sortkey)**" order="ascending"/>。altova:evaluate() 関数は、コンテキストノードの親の UserReq 子要素の sortkey 属性を読み込みます。sortkey 属性の値が Price の場合、Price は、altova:evaluate() 関数により返され、select 属性 <xsl:sort select="**Price**" order="ascending"/> の値になります。この sort 命令が、Order という要素のコンテキスト内で発生する場合、Order 要素は Price の子の値に従い並べ替えられます。また、@sortkey の値が、Date の場合、Order 要素は、Date の子の値に従い並べ替えられます。ですから、Order の並べ替えの条件は、ランタイムでの sortkey 属性から選択されます。これは、以下の式などでは達成することはできません: <xsl:sort select="**../UserReq/@sortkey**" order="ascending"/>。上の場合、並べ替え条件は sortkey 属性自身であり、Price または Date（または、現在の sortkey のコンテンツ）ではありません。

メモ 静的なコンテキストは、変数以外以外で、呼び出し環境の名前空間、型、機能、を含みます。ベース URI とデフォルトの名前空間は継承されます。

追加サンプル

- 静的な変数: `<xsl:value-of select="$i3, $i2, $i1" />`
  3 つの変数の値を出力します。

- 動的な変数を持つ動的 XPath 式:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />
  ```
  "30 20 10" を出力します。

- 動的な変数を持たない XPath 式:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath)" />
  ```
  出力エラー: $p3 に対して定義されている変数はありません。

encode-for-rtf [altova:]

altova:encode-for-rtf(input *as xs:string*, **preserveallwhitespace** *as xs:boolean*, **preservenewlines** *as xs:boolean*) を **xs:string** とする **XSLT2** **XSLT3**
RTF のためのコードに入力文字列を変換します。空白と新しい行は、それぞれの引数により指定される boolean の値に基づき保管されます。

[トップ]

XBRL 関数

Altova XBRL 関数は XBRL をサポートする Altova 製品のエディションのみで使用することができます。

xbrl-footnotes [altova:]

altova:xbrl-footnotes(*node()***)** を **node()*** とする **XSLT2** **XSLT3**
ノードを入力引数として必要とし、入力ノードに参照される XBRL フットノート ノードを返します。

xbrl-labels [altova:]

altova:xbrl-labels(*xs:QName*, *xs:string***)** を **node()*** とする **XSLT2** **XSLT3**
以下の 2 つの入力引数を必要とします: ノード名とノードを含むタクソノミファイルロケーション。関数は、入力ノードと関連した XBRL ラベルノードを返します。

[トップ]

(C) 2015-2021 Altova GmbH

18.2.1.2 XPath/XQuery 関数: 日付と時刻

Altova の日付/時刻拡張関数はXPath とXQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の**XPath 3.0** および**XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、`http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

▼ 機能によりグループ化

- xs:dateTime に期間を追加して、xs:dateTimeを返す
- xs:date に期間を追加して、xs:date を返す
- xs:time に期間を追加して、return xs:time を返す
- フォーマットと期間の取得
- 現在の日付/時刻を生成する関数からタイムゾーンを削除する
- 期間から日付、時刻、分数、および、秒数を返す
- 日付から整数を週の曜日として返す
- 日付から週数を整数として返す
- 各型の構文コンポーネントから日付、時刻、期間 の型を構築する
- 文字列入力から 日付、日付時刻 または 時刻 を構築する
- 年齢に関連した関数

▼ アルファベット順にグループ化

altova:add-days-to-date
altova:add-days-to-dateTime
altova:add-hours-to-dateTime
altova:add-hours-to-time
altova:add-minutes-to-dateTime
altova:add-minutes-to-time
altova:add-months-to-date
altova:add-months-to-dateTime
altova:add-seconds-to-dateTime
altova:add-seconds-to-time
altova:add-years-to-date
altova:add-years-to-dateTime
altova:age
altova:age-details
altova:build-date
altova:build-duration
altova:build-time
altova:current-dateTime-no-TZ
altova:current-date-no-TZ

altova:current-time-no-TZ
altova:date-no-TZ
altova:dateTime-no-TZ
altova:days-in-month
altova:hours-from-dateTimeDuration-accumulated
altova:minutes-from-dateTimeDuration-accumulated
altova:seconds-from-dateTimeDuration-accumulated
altova:format-duration
altova:parse-date
altova:parse-dateTime
altova:parse-duration
altova:parse-time
altova:time-no-TZ
altova:weekday-from-date
altova:weekday-from-dateTime
altova:weeknumber-from-date
altova:weeknumber-from-dateTime

[トップ]

xs:dateTime に期間を追加する XP3.1 XQ3.1

これらの関数は**xs:dateTime** に期間を追加し、**xs:dateTime** を返します。xs:dateTime 型はCCYY-MM-DDThh:mm:ss.sss のフォーマットです。これはxs:date とxs:time フォーマットの連結で、T により区切られています。タイムゾーンサフィックス+01:00 (for example)は任意です。

add-years-to-dateTime [altova:]

altova:add-years-to-dateTime(DateTime *as xs:dateTime*, **Years** を *xs:integer*)
asxs:dateTime とする XP3.1 XQ3.1
日付までの期間を年数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される年数です。結果はxs:date 型です。

サンプル

- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) は2024-01-15T14:00:00 を返します。
- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -4) は2010-01-15T14:00:00 を返します。

add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(DateTime *as xs:dateTime*, **Months** を *xs:integer*)
asxs:dateTime とする XP3.1 XQ3.1
xs:dateTime に月数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられたxs:dateTime に追加される月数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) 2014-11-15T14:00:00 を返します。
- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -2) 2013-11-15T14:00:00 を返します。

add-days-to-dateTime [altova:]

(C) 2015-2021 Altova GmbH

altova:add-days-to-dateTime(DateTime *as xs:dateTime*, **Days** *as xs:integer*) を **xs:dateTime** とする **XP3.1** **XQ3.1**
xs:dateTime に日数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される日数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) は 2014-01-25T14:00:00 を返します。
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8) は 2014-01-25T14:00:00 を返します。

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(DateTime *as xs:dateTime*, **Hours** *as xs:integer*) を **xs:dateTime** とする **XP3.1** **XQ3.1**
xs:dateTime に時間数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される時間数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) は 2014-01-15T23:00:00 を返します。
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) は 2014-01-15T05:00:00 を返します。

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(DateTime *as xs:dateTime*, **Minutes** *as xs:integer*) を **xs:dateTime** とする **XP3.1** **XQ3.1**
xs:dateTime に分数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される分数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) 2014-01-15T14:55:00 を返します。
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) 2014-01-15T14:05:00 を返します。

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(DateTime *as xs:dateTime*, **Seconds** を *xs:integer*) **asxs:dateTime** とする **XP3.1** **XQ3.1**
xs:dateTime に秒数での期間を追加します（*下のサンプル参照*）。第 2 の引数は、第 1 の引数として与えられた xs:dateTime に追加される秒数です。結果は xs:dateTime 型です。

サンプル

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) 2014-01-15T14:00:30 を返します。
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) 2014-01-15T14:00:05 を返します。

[トップ]

xs:date に期間を追加する XP3.1 XQ3.1

これらの関数は**xs:date** に期間を追加し、**xs:date** を返します。xs:date 型はCCYY-MM-DD フォーマットです。

add-years-to-date [altova:]

altova:add-years-to-date(Date *as xs:date*, **Years** *as xs:integer*) を**xs:date** とする XP3.1 XQ3.1

日付までの期間を年数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される年数です。結果は xs:date 型です。

サンプル

- **altova:add-years-to-date**(xs:date("2014-01-15"), 10) は2024-01-15 を返します。
- **altova:add-years-to-date**(xs:date("2014-01-15"), -4) は2010-01-15 を返します。

add-months-to-date [altova:]

altova:add-months-to-date(Date *as xs:date*, **Months** *as xs:integer*) を**xs:date** とする XP3.1 XQ3.1

日付までの期間を月数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される月数です。結果は xs:date 型です。

サンプル

- **altova:add-months-to-date**(xs:date("2014-01-15"), 10) 2014-11-15 を返します。
- **altova:add-months-to-date**(xs:date("2014-01-15"), -2) 2013-11-15 を返します。

add-days-to-date [altova:]

altova:add-days-to-date(Date *as xs:date*, **Days** *as xs:integer*) を**xs:date** とする XP3.1 XQ3.1

日付までの期間を日数で表示します。第 2 の引数は第 1 の引数として与えられたxs:date に追加される日数です。結果は xs:date 型です。

サンプル

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10) は 2014-01-25 を返します。
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) は2014-01-07 を返します。

[トップ]

フォーマットと期間の取得 XP3.1 XQ3.1

これらの関数は入力 **xs:duration** または**xs:string** を解析し、それぞれan **xs:string** または**xs:duration** を返します。

format-duration [altova:]

altova:format-duration(Duration *as xs:duration*, **Picture** *as xs:string*) **asxs:string** とする XP3.1 XQ3.1

第 1 の引数として提出された期間を、第 2 の引数として提出された文字列によりをフォーマットします。出力は、文字列によりフォーマットされたテキスト文字列です。

(C) 2015-2021 Altova GmbH

サンプル

- **altova:format-duration**(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") は"Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7" を返します。
- **altova:format-duration**(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") は"Months:03 Days:02 Hours:02 Minutes:53" を返します。

parse-duration [altova:]

altova:parse-duration(InputString *as xs:string*, **Picture** *as xs:string*) を**xs:duration** とする **XP3.1** **XQ3.1**
パターン化された文字列を最初の引数として、文字を第 2 の引数とします。入力文字列は文字をベースに解析され、xs:duration が返されます。

サンプル

- **altova:parse-duration**("Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") は "P2DT2H53M11.7S" を返します。
- **altova:parse-duration**("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") は "P3M2DT2H53M" を返します。

[トップ]

xs:time に期間を追加する **XP3.1** **XQ3.1**

これらの関数は**xs:time** に期間を追加し、 **xs:time** を返します。 **xs:time** 型はhh:mm:ss.sss 構文フォームです。文字 Z は協定世界時（UTC）を表します。他のタイムゾーンは、UTC との差異を +hh:mm または -hh:mm のフォーマットで表示しています。タイムゾーンの値が無い場合は、UTC ではない未知のタイムゾーンとして見なされます。

add-hours-to-time [altova:]

altova:add-hours-to-time(Time *as xs:time*, **Hours** *as xs:integer*) を**xs:time** とする **XP3.1** **XQ3.1**
日付までの期間を時間数で表示します。第 2 の引数は第 1 の引数として与えられた xs:time に追加される時間数です。結果は xs:time 型です。

サンプル

- **altova:add-hours-to-time**(xs:time("11:00:00"), 10) は21:00:00 を返します。
- **altova:add-hours-to-time**(xs:time("11:00:00"), -7) は04:00:00 を返します。

add-minutes-to-time [altova:]

altova:add-minutes-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) を**xs:time** とする **XP3.1** **XQ3.1**
日付までの期間を分数で表示します。第 2 の引数は第 1 の引数として与えられた xs:date に追加される分数です。結果は xs:date 型です。

サンプル

- **altova:add-minutes-to-time**(xs:time("14:10:00"), 45) 14:55:00 を返します。

- **altova:add-minutes-to-time**(xs:time("14:10:00"), -5) 14:05:00 を返します。

add-seconds-to-time [altova:]

altova:add-seconds-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) を **xs:time** とする **XP3.1** **XQ3.1**
時間までの期間を秒数で表示します。第 2 の引数は、第 1 の引数として与えられた xs:time に追加される秒数です。結果は xs:time 型です。第 2 のコンポーネントは 0 から 59.999 の範囲であることができます。

サンプル

- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20) は 14:00:20 を返します。
- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20.895) は 14:00:20.895 を返します。

[トップ]

日付/時刻データ型からタイムゾーンの部分を削除する **XP3.1** **XQ3.1**

これらの関数は、現在の **xs:dateTime**、**xs:date**、または **xs:time** 値からそれぞれタイムゾーンを削除します。xs:dateTime と xs:dateTimeStamp の差異は、後者のタイムゾーンが必要な場合です。(前者の場合は任意です。) xs:dateTimeStamp 値のフォーマットは CCYY-MM-DDThh:mm:ss.sss±hh:mm または CCYY-MM-DDThh:mm:ss.sssZ です。、日付と時刻が xs:dateTimeStamp としてシステムクロックから読み込まれる場合、 current-dateTime-no-TZ() 関数がタイムゾーンを削除するために使用されます。

current-date-no-TZ [altova:]

altova:current-date-no-TZ() を **xs:date** とする **XP3.1** **XQ3.1**
この関数は引数を必要としません。current-date() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、xs:date の値を返します。

サンプル

現在の date が **2014-01-15+01:00** の場合:

- **altova:current-date-no-TZ**() は 2014-01-15 を返します。

current-dateTime-no-TZ [altova:]

altova:current-dateTime-no-TZ() を **xs:dateTime** とする **XP3.1** **XQ3.1**
この関数は引数を必要としません。current-dateTime() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、xs:dateTime の値を返します。

サンプル

現在の dateTime が **2014-01-15T14:00:00+01:00** の場合:

- **altova:current-dateTime-no-TZ**() は 2014-01-15T14:00:00 を返します。

current-time-no-TZ [altova:]

altova:current-time-no-TZ() を **xs:time** とする **XP3.1** **XQ3.1**
この関数は引数を必要としません。current-time() (システムクロックによる現在の時刻) のタイムゾーンの部分を削除し、

(C) 2015-2021 Altova GmbH

xs:time の値を返します。

サンプル

現在のtime が14:00:00+01:00 の場合:

- altova:current-time-no-TZ() は14:00:00 を返します。

date-no-TZ [altova:]

altova:date-no-TZ(InputDate *as xs:date*) をxs:date とする XP3.1 XQ3.1
この関数はxs:date 引数を必要とし、タイムゾーンの部分を削除し、xs:date の値を返します。日付が変更されていない点に注意してください。

サンプル

- altova:date-no-TZ(xs:date("2014-01-15+01:00")) は2014-01-15 を返します。

dateTime-no-TZ [altova:]

altova:dateTime-no-TZ(InputDateTime *as xs:dateTime*) をxs:dateTime とする XP3.1 XQ3.1
この関数はxs:dateTime 引数を必要とし、タイムゾーンの部分を削除し、xs:dateTime の値を返します。日付が変更されていない点に注意してください。

サンプル

- altova:dateTime-no-TZ(xs:date("2014-01-15T14:00:00+01:00")) は2014-01-15T14:00:00 を返します。

time-no-TZ [altova:]

altova:time-no-TZ(InputTime *as xs:time*) をxs:time とする XP3.1 XQ3.1
この関数はxs:time 引数を必要とし、タイムゾーンの部分を削除し、xs:time 値を返します。時刻は変更されていないことに注意してください。

サンプル

- altova:time-no-TZ(xs:time("14:00:00+01:00")) は14:00:00 を返します。

[トップ]

期間から日数、時間数、分数、および、秒数を返す XP3.1 XQ3.1

これらの関数は、期間から、月内の日数、時間数、分数、秒数をそれぞれ返します。

days-in-month [altova:]

altova:days-in-month(Year *as xs:integer*, Month *as xs:integer*) asxs:integer XP3.1 XQ3.1
は指定された月内の日数 を返します。Year とMonth 引数を使用して月を指定することができます。

サンプル

- altova:days-in-month(2018, 10) は31 を返します。
- altova:days-in-month(2018, 2) は28 を返します。
- altova:days-in-month(2020, 2) は29 を返します。

hours-from-dayTimeDuration-accumulated

altova:hours-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*) **asxs:integer**
XP3.1 **XQ3.1**

DayAndTime 引数 (which is of type xs:duration) により提供される期間内の時間の総数を返します。Day と Time コンポーネント内の時間は、整数である結果に追加されます。60 分として新規の一時間は考えられます。ネガティブな時間の値で期間の結果をナビゲートすることができます。

サンプル

- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5D") は5日間の時間の総計である 120 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H") は5日間の時間の総計プラス2時間である 122 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H60M") は5日間の時間の総計プラス2時間プラス60分である 123 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H119M") は5日間の時間の総計プラス2時間とプラス119分である 123 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H120M") は5日間の時間の総計プラス2時間プラス120分である 124 を返します。
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("-P5DT2H") は -122 を返します。

minutes-from-dayTimeDuration-accumulated

altova:minutes-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*)
asxs:integer **XP3.1** **XQ3.1**

DayAndTime 引数 (which is of type xs:duration) により提出される期間内の分数の総数を返します Day と Time コンポーネントは、追加され、整数である結果に追加されます。ネガティブなマイナスの値で期間の結果をナビゲートします。

サンプル

- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT60M") は を返します。60
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H") は一時間内の分数である 60 を返します。
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H40M") は100 を返します。
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("P1D") は一日の秒数の総数である 1440 を返します。
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("-P1DT60M") は を返します。-1500

seconds-from-dayTimeDuration-accumulated

altova:seconds-from-dayTimeDuration-accumulated(DayAndTime *as xs:duration*)
asxs:integer **XP3.1** **XQ3.1**

(xs:duration の型である)DayAndTime 引数 により提出された期間内の秒数の総数を返します。Day と Time コンポーネント内の秒数は、整数である結果に追加されます。ネガティブな秒の値で期間をナビゲートします。

サンプル

- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1M") は一分内の秒数である 60 を返します。

(C) 2015-2021 Altova GmbH

- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H") は一時間内の秒数の総数である 3600 を返します。
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H2M") は3720 を返します。
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("P1D") は一日の秒数の総数である 86400 を返します。
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("-P1DT1M") は -86460 を返します。

xs:dateTime または xs:dateReturn から週数を返す XP3.1 XQ3.1

これらの関数は、xs:dateTime または xs:date から曜日を(整数として)を返します。曜日は(米国式フォーマットを使用して) 1 から 7 と番号づけられています。 日曜=1 と番号付けられます。欧州のフォーマットでは月曜 (=1) として番号付けられます。日曜=1 である米国フォーマットは整数 0 がフォーマットを表示するために使用できる箇所で設定することができます。

weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *as xs:dateTime*) を **xs:integer** とする XP3.1 XQ3.1

時刻付きの日付を単一の引数として、この日付の曜日を正数として返します。曜日は 日曜=1 から開始して番号を付けます。(月曜=1 とする) ヨーロッパのフォーマットが必要な場合、この関数の他の署名を使用します (*下の次の署名を参照*)。

サンプル

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00")) は月曜を表示する 2 を返します。

altova:weekday-from-dateTime(DateTime *as xs:dateTime*, **Format** *as xs:integer*) **asxs:integer** とする XP3.1 XQ3.1

時間月の日付を最初の引数として、この日付の曜日を正数として返します。曜日は 月曜=1 から開始して番号を付けます。第 2 の引数(整数)が 0 の場合、日曜=1 から開始して、曜日は 1 から 7 と番号を付けます。第 2 の引数が 0 以外の整数の場合、月曜=1 です。第 2 の引数がない場合、関数はこの関数の他の署名を持つと読み込まれます (*前の次の署名を参照*)。

サンプル

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 1) は月曜を表示する 1 を返します。
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 4) は月曜を表示する 1 を返します。
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 0) は月曜を表示する 2 を返します。

weekday-from-date [altova:]

altova:weekday-from-date(Date *as xs:date*) を **xs:integer** とする XP3.1 XQ3.1

日付を単一の引数として、この日付の曜日を正数として返します。曜日は 日曜=1 から開始して番号を付けます。(月曜=1 とする) ヨーロッパのフォーマットが必要な場合、この関数の他の署名を使用します (*下の次の署名を参照*)。

サンプル

- **altova:weekday-from-date**(xs:date("2014-02-03+01:00")) は月曜を表示する 2 を返します。

altova:weekday-from-date(Date *as xs:date*, **Format** *as xs:integer*) を **xs:integer** とする XP3.1 XQ3.1

日付を最初の引数とし、この日付の曜日を正数として返します。曜日は 月曜=1 から開始して番号を付けます。第 2 (フォーマット) 引数が 0, の場合、日曜=1 から開始し、曜日は 1 から 7 で番号付けられます。第 2 引数が整数で 0 以外の場合、月曜=1 です。第 2 の引数がない場合、関数はこの関数の他の署名を持つと読み込まれます (*前の署名を参照*)。

サンプル

- **altova:weekday-from-date**(xs:date("2014-02-03"), 1) は月曜を示す 1 を返します。
- **altova:weekday-from-date**(xs:date("2014-02-03"), 4) は月曜を示す 1 を返します。
- **altova:weekday-from-date**(xs:date("2014-02-03"), 0) は月曜を示す 2 を返します。

[トップ]

xs:dateTime または xs:date から週数を返す XP2 XQ1 XP3.1 XQ3.1

これらの関数は週数(整数として)を xs:dateTime から xs:date. から返します。週の番号付けは、米国、欧州、イスラムのカレンダーフォーマットで使用することができます。週の始まりが異なるため、週数の番号付けは、カレンダーのフォーマットにより異なります。(米国フォーマットでは、日曜、欧州フォーマットでは月曜、イスラムフォーマットでは土曜が週の開始日です)。

weeknumber-from-date [altova:]

altova:weeknumber-from-date(Date *as xs:date*, **Calendar** *as xs:integer*) を **xs:integer** とする XP2 XQ1 XP3.1 XQ3.1

正数として提出された Date 引数の週数を返します。第 2 の引数(カレンダー)は続くカレンダーシステムを指定します。
サポートされる カレンダー の値は次のとおりです:

- 0 = US 米国のカレンダー(週の始まりは日曜日)
- 1 = ISO 標準、 欧州のカレンダー(週の始まりは月曜日)
- 2 = イスラムのカレンダー(週の始まりは土曜日)

デフォルトは 0 です。

サンプル

- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 0) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 1) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 2) は 13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23")) は 13 を返します。

上のサンプル(2014-03-23) の date の曜日は日曜日です。米国およびイスラムのカレンダーは欧州カレンダーのこの日付よりも一週間先です。

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(DateTime *as xs:dateTime*, **Calendar** *as xs:integer*) を **xs:integer** とする XP2 XQ1 XP3.1 XQ3.1

正数として提出された DateTime 引数の週数を返します。第 2 の引数(カレンダー)は続くカレンダーシステムを指定します。
サポートされる カレンダー の値は次のとおりです:

- 0 = US 米国のカレンダー(週の始まりは日曜日)
- 1 = ISO 標準、 欧州のカレンダー(週の始まりは月曜日)
- 2 = イスラムのカレンダー(週の始まりは土曜日)

Default is 0.

サンプル

- altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 0) は13 を返します。
- altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 1) は13 を返します。
- altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 2) は13 を返します。
- altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00")) は13 を返します。

上のサンプル(2014-03-23T00:00:00) のdateTime の曜日は日曜日です。米国およびイスラムのカレンダーは欧州カレンダーのこの日付より先一週間先です。

[トップ]

各型の構文コンポーネントから日付、時刻、期間の型を構築する XP3.1 XQ3.1

関数は、xs:date, xs:time またはxs:duration の構文コンポーネントを入力引数とし、引数を結合させて対応するデータ型を構築します。

build-date [altova:]

altova:build-date(Year *as xs:integer*, **Month** *as xs:integer*, **Date** *as xs:integer*) を **xs:date** とする XP3.1 XQ3.1

第 1 、第 2 、第3 の引数は、それぞれ、年、月、日を表します。xs:date 型の値を構築するため結合されます。整数の値は、特定の日付の一部の適正な範囲内である必要があります。例えば第 2 の引数 (月の部分) は12 以上であってはなりません。

サンプル

- altova:build-date(2014, 2, 03) は2014-02-03 を返します。

build-time [altova:]

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) を**xs:time** とする XP3.1 XQ3.1

第 1 、第 2 、第3 引数はそれぞれ時間数 (0 から23)、分数 (0 から59)、および秒数 (0 から59) の値です。これらの値は、xs:time 型の値を作成するために結合されます。整数の値は、それぞれの部分の正しい範囲内である必要があります。例えば、秒 (分) 引数は、59 以上であってはなりません。値にタイムゾーンを追加するには、この関数の他の署名を使用してください (*次の署名を参照*)。

サンプル

- altova:build-time(23, 4, 57) は23:04:57 を返します。

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*, **TimeZone** *as xs:string*) を**xs:time** とする XP3.1 XQ3.1

第 1 、第 2 、第 3 引数はそれぞれ時間数 (0 から23)、分数 (0 から59)、および秒数 (0 から59) の値です。第四の引数は、値の一部としてタイムゾーンを与えます。4 つの引数が結合され、xs:time 型の値を構築します。例えば、第 2 の引数 (分数) は59 以上であってはなりません。

サンプル

- **altova:build-time**(23, 4, 57, '+1') は23:04:57+01:00 を返します。

build-duration [altova:]

altova:build-duration(Years *as xs:integer*, **Months** *as xs:integer*) を **xs:yearMonthDuration** とする **XP3.1** **XQ3.1**

xs:yearMonthDuration 型の値を構築するためには2 つの引数が必要です。最初の引数は期間の 年数 値の部分を与え、第 2 の引数は、月数 値の部分を与えます。第 2 の引数（月数）が12 同じまたはより大きくなると、整数は12 により分割されます。商は、年数 の部分を表す 最初の引数に加算され、（除算の）残りは月の部分は 月数の部分を表すために使用されます。期間の xs:dayTimeDuration 型を作成するには、次の署名を参照してください。

サンプル

- **altova:build-duration**(2, 10) はP2Y10M を返します。
- **altova:build-duration**(14, 27) はP16Y3M を返します。
- **altova:build-duration**(2, 24) はP4Y を返します。

altova:build-duration(Days *as xs:integer*, **Hours** *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) を**xs:dayTimeDuration** とする **XP3.1** **XQ3.1**

xs:dayTimeDuration 型の値を構築するためには4 つの引数が必要です。最初の引数は期間の 日数 値の部分で、第 2 、第 3 、第 4 引数は、それぞれ期間の時間数、分数、秒数 値のです。これらの3 つの 時間 引数は、次の高い単位の値に加算され、結果は期間の全体の値を計算するために使用されます。例えば、72 秒は、1M+12S（1 分 と12 秒）に変換され、この値が期間全体の値を計算するために使用されます。xs:yearMonthDuration 型の期間を構築するためには、次の署名を参照してください。

サンプル

- **altova:build-duration**(2, 10, 3, 56) はP2DT10H3M56S を返します。
- **altova:build-duration**(1, 0, 100, 0) はP1DT1H40M を返します。
- **altova:build-duration**(1, 0, 0, 3600) はP1DT1H を返します。

[トップ]

文字列入力から 日付、日付時刻 または 時刻 を構築する **XP2** **XQ1** **XP3.1** **XQ3.1**

関数は、文字列を引数として、xs:date、xs:dateTime または xs:time データ型を構築します。文字列は、データ型のコンポーネントを提出されたパターン引数をベースにして分析されます。

parse-date [altova:]

altova:parse-date(Date *as xs:string*, **DatePattern** *as xs:string*) を**xs:date** とする **XP2** **XQ1** **XP3.1** **XQ3.1**

Date 入力文字列を**xs:date** の値として返します。第二の引数である**DatePattern** は、入力文字列のパターン（コンポーネントのシーケンス）を指定します。**DatePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネントセパレータと共に説明されています。下のサンプルを参照してください。

D 日付
M 月
Y 年

DatePattern のパターンは、**Date** のパターンに一致する必要があります。出力は**xs:date** 型であるため、出力は常に **YYYY-MM-DD** 構文フォーマットになります。

サンプル

- **altova:parse-date**(xs:string("06-03-2014"), "[D]-[M]-[Y]") は2014-03-06 を返します。
- **altova:parse-date**(xs:string("06-03-2014"), "[M]-[D]-[Y]") は2014-03-06 を返します。
- **altova:parse-date**("06/03/2014", "[M]/[D]/[Y]") は2014-03-06 を返します。
- **altova:parse-date**("06 03 2014", "[M] [D] [Y]") は2014-03-06 を返します。
- **altova:parse-date**("6 3 2014", "[M] [D] [Y]") は2014-03-06 を返します。

parse-dateTime [altova:]

altova:parse-dateTime(DateTime *as xs:string*, **DateTimePattern** *as xs:string*) を **xs:dateTime** とする **XP2** **XQ1** **XP3.1** **XQ3.1**

DateTime 入力文字列を**xs:dateTime** の値として返します。第二の引数である**DateTimePattern** は、入力文字列のパターン（コンポーネントのシーケンス）を指定します。**DateTimePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネント セパレータ と共に説明されています。下のサンプルを参照してください。

D	日
M	月
Y	年
H	時間
m	分
s	秒

DateTimePattern のパターンは、**DateTime** のパターンに一致する必要があります。出力は**xs:dateTime** 型であるため、出力は常に**YYYY-MM-DDTHH:mm:ss** 構文フォーマットになります。

サンプル

- **altova:parse-dateTime**(xs:string("06-03-2014 13:56:24"), "[D]-[M]-[Y] [H]:[m]:[s]") は2014-03-06T13:56:24 を返します。
- **altova:parse-dateTime**("time=13:56:24; date=06-03-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") は2014-03-06T13:56:24 を返します。

parse-time [altova:]

altova:parse-time(Time *as xs:string*, **TimePattern** *as xs:string*) を**xs:time** とする **XP2** **XQ1** **XP3.1** **XQ3.1**

Time 入力文字列を**xs:time** の値として返します。第二の引数である**TimePattern** は、入力文字列のパターン（コンポーネントのシーケンス）を指定します。**TimePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネント セパレータ と共に説明されています。下のサンプルを参照してください。

H	時間
m	分
s	秒

TimePattern のパターンは、**Time** のパターンに一致する必要があります。出力は**xs:time** 型であるため、出力は常に**YYYY-HH:mm:ss** 構文フォーマットになります。

サンプル

- **altova:parse-time**(xs:string("13:56:24"), "[H]:[m]:[s]") は13:56:24 を返します。
- **altova:parse-time**("13-56-24", "[H]-[m]") は13:56:00 を返します。
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") は13:56:24 を返します。

- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") は13:56:24 を返します。

[トップ]

年齢に関連した関数 XP3.1 XQ3.1

関数は計算された年齢(i) 入力引数の日付と現在の日付の期間 (ii) 2 つの入力引数の日付の期間 を返します。**altova:age** 関数は、年齢を年数で返します、**altova:age-details** は年齢を年数、月数、日数からなる3 つの整数のシーケンスで返します。

age [altova:]

altova:age(StartDate *as xs:date***)** を**xs:integer** とする XP3.1 XQ3.1

引数として提出された開始日から(システムクロックから取得された)現在の日付 までの日数を数えて、あるオブジェクトの年齢の年数である正数を返します。入力引数が1 年より先大きい場合、または一年の場合、返される値は負の数です。

サンプル

If the current date is **2014-01-15**:

- **altova:age**(xs:date("2013-01-15")) は1 を返します。
- **altova:age**(xs:date("2013-01-16")) は0 を返します。
- **altova:age**(xs:date("2015-01-15")) は-1 を返します。
- **altova:age**(xs:date("2015-01-14")) は0 を返します。

altova:age(StartDate *as xs:date*, **EndDate** *as xs:date***)** を**xs:integer** とする XP3.1 XQ3.1

最初の引数として提出された開始日から第 2 の引数の終了日 までの日数を数えて、あるオブジェクトの年齢の年数である正数を返します。最初の引数が1 年または第 2 の引数より先後の日付の場合、返される値は負の数です。

サンプル

If the current date is **2014-01-15**:

- **altova:age**(xs:date("2000-01-15"), xs:date("2010-01-15")) は10 を返します。
- **altova:age**(xs:date("2000-01-15"), current-date()) は、現在の日付が2014-01-15 の場合 14 を返します。
- **altova:age**(xs:date("2014-01-15"), xs:date("2010-01-15")) は-4 を返します。

age-details [altova:]

altova:age-details(InputDate *as xs:date***)** を(**xs:integer**)* とする XP3.1 XQ3.1

引数と(システムクロックから取得された)現在の日付として提出された日付の間の年数、月数、日数の3 つの整数を返します。返された years+months+days の合計は、2 つの日付 (入力の日付と現在の日付) の時間差です。 入力の日付は現在の日付より先早いまたはお遅い値をもつことができますが、入力の日付が早いか遅いかは返される値で示されていません。戻される値は常に正の数です。

サンプル

現在の日付が**2014-01-15** の場合:

- **altova:age-details**(xs:date("2014-01-16")) は(0 0 1)を返します。
- **altova:age-details**(xs:date("2014-01-14")) は(0 0 1) を返します。
- **altova:age-details**(xs:date("2013-01-16")) は(1 0 1) を返します。
- **altova:age-details**(current-date()) は(0 0 0)を返します。

(C) 2015-2021 Altova GmbH

altova:age-details(Date-1 *as xs:date*, **Date-2** *as xs:date*) を(**xs:integer)*** とする **XP3.1** **XQ3.1**
二つの引数間の年数、月数、日数の3 つの整数を返します。 返された years+months+days の合計は、2 つの入力の日付の時間差です。最初の引数として提出される二つの日付はどちらが速くても、または遅くてもかまいません。返される値は入力の日付が現在の日付より早いかまたは遅いかを示しません。戻される値は常に正の数です。

サンプル

- **altova:age-details**(xs:date("2014-01-16"), xs:date("2014-01-15")) は (0 0 1) を返します。
- **altova:age-details**(xs:date("2014-01-15"), xs:date("2014-01-16")) は (0 0 1)を返します。

[トップ]

18.2.1.3 XPath/XQuery 関数: 位置情報

以下の位置情報 XPath/XQuery 拡張関数は、StyleVision の現在のバージョンによりサポートされています。また、次で使用することができます: (i) XSLT コンテキスト内のXPath 式、または(ii) XQuery ドキュメント内のXQuery 式。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

format-geolocation [altova:]

altova:format-geolocation(Latitude *as xs:decimal*, **Longitude** *as xs:decimal*, **GeolocationOutputStringFormat** *as xs:integer*) を**xs:string** とする **XP3.1** **XQ3.1**
緯度と経度を最初の2 つの引数とし、位置情報を文字列として出力します。第 3 の引数 **GeolocationOutputStringFormat** は、位置情報出力文字列のフォーマットです。出力文字列のフォーマットを識別するために、正数の値 1 から 4 を使用します。(*下の「位置情報出力文字列フォーマット」を参照してください*)。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西)です。

メモ image-exif-data 関数とExif メタデータの @Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:format-geolocation**(33.33, -22.22, 4) はxs:string "33.33 -22.22" を返します。

- **altova:format-geolocation**(33.33, -22.22, 2) はxs:string "33.33N 22.22W" を返します。
- **altova:format-geolocation**(-33.33, 22.22, 2) はxs:string "33.33S 22.22E" を返します。
- **altova:format-geolocation**(33.33, -22.22, 1) はxs:string "33°19'48.00"S 22° 13'12.00"E" を返します。

位置情報出力文字列フォーマット:

与えられた緯度と経度は下にリストされる出力フォーマットによりフォーマットされます。希望するフォーマットは整数 ID (1 から 4) により識別されます。緯度の値の範囲は+90 から-90 (北 から 南) です。経度の値の範囲は+180 から-180 (東 から 西)です。

1
度、分、10 進の秒、方角のサフィックス付き (N/S, E/W) **D°M'S.SS"N/S D°M'S.SS"E/W** *サンプル:* **33°55'11.11"N 22°44'66.66"W**

2
10 進の度、方角のサフィックス付き (N/S, E/W) **D.DDN/S D.DDE/W** *サンプル:* **33.33N 22.22W**

3
度、分、10 進の秒、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。 **+/-D°M'S.SS" +/-D°M'S.SS"** *サンプル:* **33°55'11.11" -22°44'66.66"**

4
10 進の度、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。 **+/-D.DD +/-D.DD** *サンプル:* **33.33 -22.22**

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

(C) 2015-2021 Altova GmbH

parse-geolocation [altova:]

altova:parse-geolocation(GeolocationInputString *as xs:string*) を **xs:decimal+** とする **XP3.1 XQ3.1**

GeolocationInputString 引数を解析して、位置情報の緯度と経度（この通りの順番）を 2 つの xs:decimal アイテムのシーケンスとして返します。位置情報入力文字列が提供されることのできるフォーマットは以下のリストの通りです。

メモ image-exif-data 関数と Exif メタデータの @Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:parse-geolocation**("33.33 -22.22") は 2 つの xs:decimals (33.33, 22.22) のシーケンスを返します。
- **altova:parse-geolocation**("48°51'29.6""N 24°17'40.2""") は 2 つの xs:decimals (48.8582222222222, 24.2945) のシーケンスを返します。
- **altova:parse-geolocation**('48°51''29.6"N 24°17''40.2"') は 2 つの xs:decimals (48.8582222222222, 24.2945) のシーケンスを返します。
- **altova:parse-geolocation**(**image-exif-data**(//MyImages/Image20141130.01)**/@Geolocation**) は 2 つの xs:decimals のシーケンスを返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度（この通りの順番）を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が 1 つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルとしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い (") でハイライトされており、エスケープした単位インジケーターは青い ("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き (N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-

D.DD
サンプル: 33.33 -22.22

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-km [altova:]

altova:geolocation-distance-km(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) を**xs:decimal** とする **XP3.1** **XQ3.1**
2 つの位置情報の間の距離をキロメートルで計算します。位置情報入力文字列を提供することのできるフォーマットは下にリストされています。緯度の値の範囲は+90 から-90 (北 から 南) です。経度の値の範囲は+180 から-180 (東 から 西)です。

メモ image-exif-data 関数とExif メタデータの@Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-distance-km**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""")
  はxs:decimal 4183.08132372392 を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は+90 から-90 (N からS)。経度の値の範囲は+180 から-180 (E からW)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルとしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き(N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

(C) 2015-2021 Altova GmbH

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き (+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-mi [altova:]

altova:geolocation-distance-mi(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) を **xs:decimal** とする **XP3.1** **XQ3.1**
2 つの位置情報の間の距離をマイルで計算します。位置情報入力文字列を提供することのできるフォーマットは下にリストされています。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西)です。

メモ image-exif-data 関数とExif メタデータの @Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-distance-mi**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""") は xs:decimal 2599.40652340653 を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォー

マットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は+90 から-90 (N から S)。経度の値の範囲は+180 から-180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き(N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き(N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き(+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocations-bounding-rectangle [altova:]

altova:geolocations-bounding-rectangle(Geolocations を *xs:sequence* として

GeolocationOutputStringFormat を *xs:integer* として) を**xs:string** として **XP3.1** **XQ3.1**
最初の引数として文字列のシーケンスを取ります。シーケンスの各文字列が位置情報です。関数はそれぞれが、最初の引数に送信されたすべての位置情報を含むために最適にサイズ調整された長方形の左上と右下の位置情報の座標である2つの文字列のシーケンスを返します。位置情報入力文字列が表示される書式は以下にリストされています(*「位置情報入力文字列書式」を参照してください*)。緯度の値は+90 から-90 への(N からS への) の範囲です。経度の値は+180 から-180 への(E からW への) の範囲です。

関数の2番目の引数は出力シーケンス内の2つの位置情報文字列の書式を指定します。引数はそれぞれの値が異なる位置情報文字列書式を識別する1 から4 の整数の値を取ります(*下記の「位置情報出力文字列書式」を参照してください*)。

メモ image-exif-data 関数とExif メタデータの属性は入力文字列を提供するために使用することができます。

サンプル

- **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832"), 1) はシーケンス("51°30'33.804"N 0°7'5.952"W", "48°12'51.67116"N 16°22'14.61576"E")を返します。
- **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832", "42.5584577 -70.8893334"), 4) はシーケンス("51.50939 -70.8893334", "42.5584577 16.3707266")を返します

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度(この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は+90 から-90 (N からS)。経度の値の範囲は+180 から-180 (E からW)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き(N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き(N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き(+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-

D.DD
サンプル: 33.33 -22.22

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

位置情報出力文字列フォーマット:

与えられた緯度と経度は下にリストされる出力フォーマットによりフォーマットされます。希望するフォーマットは整数 ID (1 から 4) により識別されます。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西) です。

1
度、分、10 進の秒、方角のサフィックス付き (N/S, E/W) D°M'S.SS"N/S D°M'S.SS"E/W *サンプル:* 33°55'11.11"N 22°44'66.66"W

2
10 進の度、方角のサフィックス付き (N/S, E/W) D.DDN/S D.DDE/W *サンプル:* 33.33N 22.22W

3
度、分、10 進の秒、プレフィックスサイン付き (+/-); (N/E) のためのプラスサインは任意です。 +/-D°M'S.SS" +/-D°M'S.SS" *サンプル:* 33°55'11.11" -22°44'66.66"

4
10 進の度、プレフィックスサイン付き (+/-); (N/E) のためのプラスサインは任意です。 +/-D.DD +/-D.DD *サンプル:* 33.33 -22.22

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 Geolocationを標準 Exif メタデータ タグから生成します。Geolocation は、4 つのExif タグの連結です: 単位の追加された (*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

(C) 2015-2021 Altova GmbH

geolocation-within-polygon [altova:]

altova:geolocation-within-polygon(Geolocation *as xs:string*, **((PolygonPoint** *as xs:string*)+)) を **xs:boolean** とする **XP3.1** **XQ3.1**

PolygonPoint 引数により説明されている **Geolocation**（最初の引数）が多角形のエリア内に存在するかを決定します。もし、PolygonPoint 引数が（最初と最後のポイントが同じ場合に作成される）閉じられたフィギュアを作成しない場合、フィギュアを閉じるために、最初のポイントが明示的に最後のポイントとして追加されます。全ての引数（Geolocation および PolygonPoint+）は、（*下にリストされるフォーマットの*）位置情報入力文字列により提供されます。Geolocation 引数が多角形エリア内にある場合、関数は true(); を返します。その他の場合は false() を返します。緯度の値の範囲は +90 から -90（北 から 南）です。経度の値の範囲は +180 から -180（東 から 西）です。

メモ image-exif-data 関数と Exif メタデータの @Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24", "58 -32"))** は true() を返します。
- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24"))** は true() を返します。
- **altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48° 51'29.6""N 24°17'40.2""")))** は true() を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度（この通りの順番）を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルとしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い（"）でハイライトされており、エスケープした単位インジケーターは青い（""）でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き（N/S, E/W）
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き（+/-）; （**N/E**）のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き（N/S, E/W）
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き（+/-）; （**N/E**）のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き（N/S, E/W）
  D.DDN/S D.DDW/E

サンプル: 33.33N 22.22W

- 10 進の度、プレフィックスサイン付き(+/-); (N/S E/W)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: 33.33 -22.22

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 Geolocationを標準 Exif メタデータ タグから生成します。Geolocation は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-within-rectangle [altova:]

altova:geolocation-within-rectangle(Geolocation *as xs:string*, **RectCorner-1** *as xs:string*, **RectCorner-2** *as xs:string*) を **xs:boolean** とする **XP3.1** **XQ3.1**
長方形の対格の角を指定する第 2 及び第 3 引数(RectCorner-1、RectCorner-2)により説明されている Geolocation (最初の引数) が長方形のエリア内に存在するかを決定します。全ての引数(Geolocation、RectCorner-1 および RectCorner-2) は、(*下にリストされるフォーマットの*) 位置情報入力文字列により提供されます。Geolocation 引数が長方形エリア内にある場合、関数は true(); を返します。その他の場合は false() を返します。緯度の値の範囲は +90 から -90 (北 から 南) です。経度の値の範囲は +180 から -180 (東 から 西)です。

メモ image-exif-data 関数とExif メタデータの @Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-within-rectangle("33 -22", "58 -32", "-48 24")** は true() を返します。
- **altova:geolocation-within-rectangle("33 -22", "58 -32", "48 24")** は false() を返します。
- **altova:geolocation-within-rectangle("33 -22", "58 -32", "48°51'29.6""S 24° 17'40.2""")** は true() を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (N から S)。経度の値の範囲は +180 から -180 (E から W)。

メモ 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、

それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルとしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10 進の秒、方角のサフィックス付き(N/S, E/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'55.25"W**

- 度、分、10 進の秒、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'55.25"**

- 度、分、10 進の分、方角のサフィックス付き (N/S, E/W)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10 進の分、プレフィックスサイン付き(+/-); (**N/E**) のためのプラスサインは任意です。+/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き(N/S, E/W)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き(+/-); (**N/S E/W**)) のためのプラスサインは任意です。+/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif 属性: 位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つのExif タグの連結です: 単位の追加された(*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

[トップ]

18.2.1.4 XPath/XQuery 関数: イメージに関連

以下のイメージに関連したXPath/XQuery 拡張関数は、StyleVision の現在のバージョンによりサポートされています。また、次で使用することができます: (i) XSLT コンテキスト内のXPath 式、または(ii) XQuery ドキュメント内のXQuery 式。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova 拡張関数名前空間、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

suggested-image-file-extension [altova:]

altova:suggested-image-file-extension(Base64String *as string*) を **string?** とする **XP3.1** **XQ3.1**
イメージファイルのBase64 エンコード を引数として、イメージのファイル拡張子を、イメージのBase64エンコード内の記録として返します。返された値は、エンコード内で使用することのできるイメージ型情報を基にしたヒントです。この情報が使用できない場合、空の文字列が返されます。この関数は、Base64 イメージをファイルとして保存し、適切なファイル拡張子を動的取得する際に役に立ちます。

サンプル

- **altova:suggested-image-file-extension**(/MyImages/MobilePhone/Image20141130.01) は 'jpg' を返します。
- **altova:suggested-image-file-extension**($XML1/Staff/Person/@photo) は '' を返します。

上のサンプルでは、関数の引数として与えられたノードはBase64 エンコードイメージを含むと仮定します。最初のサンプルは jpg をファイルの型および拡張子として取得します。二番目のサンプルでは、与えられたBase64 エンコードは使用できる拡張子の情報を提供しません。

mt-transform-image [altova:]

altova:mt-transform-image(Base64Image *as Base64BinaryString*, **Size** *as item()+*, **Rotation** *as xs:integer*, **Quality** *as xs:integer*) を **Base64BinaryString** とする **XP3.1** **XQ3.1**
Base64-エンコード イメージを最初の引数として、変換されたBase64-エンコード イメージを返します。第 2、第 3、第 4 引数は変換された以下のイメージ パラメーターです: サイズ、回転、およびクオリティ。

- サイズ引数には3 つのサイズ変更のオプションがあります。

(X,Y)	絶対ピクセルの値。アスペクト率は保持されません。高さと幅は自動的にイメージの長いおよび短い辺に逢わされるため、高さと幅の指示は関係ありません。2 つの整数アイテムのシーケンスとして値が入力されます。かっこが必要です。
X	X をピクセルの新しい長い辺として、イメージの縦横比のサイズ変更が行われます。アスペクト率は保持されま

	す。値は整数で引用符なしで入力されます。
'X%'	元のディメンションの与えられたパーセンテージにイメージのサイズを変更します。値は文字列として引用符を付けて入力される必要があります。

- 回転は以下の値を持つことができます: 90、180、270、-90、-180、-270。これらの値は回転の度数です。正の値はイメージを時計回りに回転させます。負の値はイメージを反時計回りに回転させます。 Altova Exif 属性 **OrientationDegree** を使用して、イメージの現在の回転の度数(0, 90, 180, 270) をイメージのExif **Orientation** タグから取得することができます。ですが、**OrientationDegree** 属性はデータの**Orientation** タグから取得されるため、Exif データ内に**Orientation** タグが存在する場合のみ使用することができます。(*下の OrientationDegree 説明を参照してください*)。
- クオリティは、0 から100 の値で、JPEG 圧縮のIJG クオリティスケールの値を参照していますが、クオリティのパーセンテージのインジケーターではありません。サイズとクオリティのどちらかを優先すると、もう一方の優先度が下がります。フルカラーソースのイメージでは、75 が通常最高値と見なされています。もし、75 が満足のいく結果をもたらさない場合は、値を上げてください。

メモ Exif データが元のイメージに存在する場合、変換時に削除され、変換されたイメージにはExif データは存在しません。

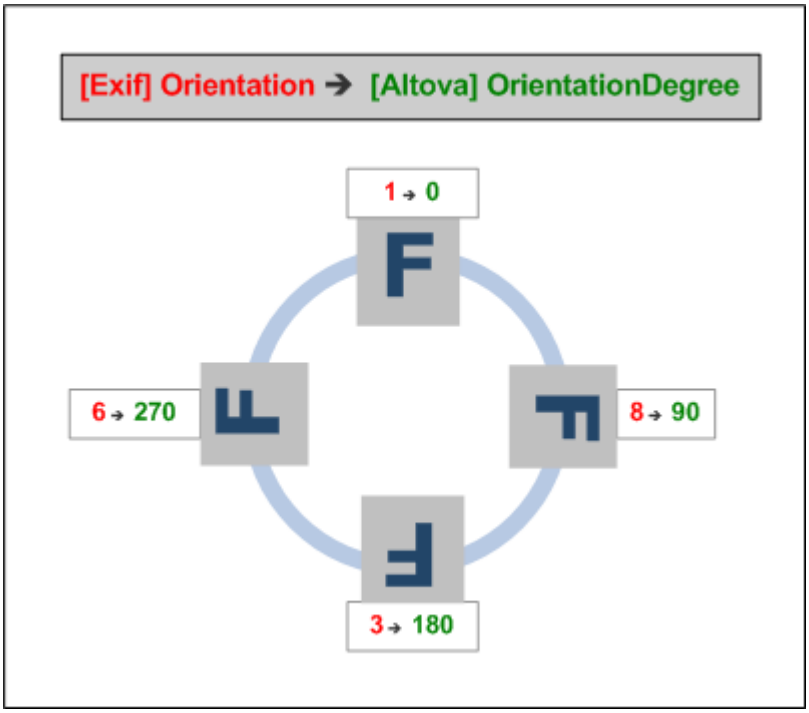
サンプル

- **mt-transform-image**(Images/Image[@id='43'], '50%', 90, 75)
  関数は、43 の@id 値を持つイメージ子孫ノード内でBase64-エンコード 文字列として保管されているイメージを入力とします。関数は変換されたイメージを返します。変換されたイメージは50% まで、サイズ変更され、時計回りに90 度回転され、75 のクオリティレベルを与えられます。
- **mt-transform-image**(Images/Image[@id='43'], 400, 90, 75)
  関数は前のサンプルと同じ結果を出しますが、長い辺は400 ピクセルの特定の値に設定されています。元のイメージのアスペクト率は保持されます。
- **mt-transform-image**(Images/Image[@id='43'], (400, 280), image-exif-data($XML1/$XML1/Images/ReferenceImage)/@OrientationDegree, 75)
  このサンプルは、前のサンプルと同じイメージを選択し、同じクオリティ値(75) を設定します。イメージのサイズは400x280 ピクセルに設定されています。回転値は、ノード内のBase64-エンコード イメージの**@OrientationDegree** 属性から得られます。

Altova Exif 属性: OrientationDegree

Altova XPath/XQuery エンジンはカスタム属性 **OrientationDegree** をExif メタデータタグ **Orientation** から生成します。

OrientationDegree は標準Exif タグ Orientation を正数の値(1、8、3 または6) からそれぞれ対応する度数の値(0, 90, 180, 270) へ下の図に示されているように変換します。2、4、5、7 のOrientation 値の変換はないことに注意してください。(これらの向きはイメージ1 を垂直方向中央軸で反転して、2 の値を持つイメージを取得します。そして、このイメージを90-度ごとに時計回りにジャンプさせそれぞれ7、4、および5 の値を取得します。)。



標準 Exif メタタグ

- ImageWidth
- ImageLength
- BitsPerSample
- Compression
- PhotometricInterpretation
- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription

(C) 2015-2021 Altova GmbH

- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio

- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

image-exif-data [altova:]

altova:image-exif-data(Base64BinaryString *as string*) を **element?** とする **XP3.1** **XQ3.1**

Base64 エンコード JPEG イメージを引数として、イメージの Exif メタデータを含む **Exif** という名の要素を返します。Exif メタデータは `Exif` 要素の属性の値ペアとして作成されます。属性名は、Base64 エンコード内で検出された Exif データタグです。Exif 仕様タグのリストは以下の通りです。ベンダー特有のタグが Exif データ内に存在する場合、タグとその値も属性値のペアとして返されます。標準 Exif メタデータタグに追加して（*下のリスト参照*）Altova 特有の属性値のペアもまた生成されます。これらの Altova Exif 属性は以下のとおりです。

(C) 2015-2021 Altova GmbH

サンプル

- 属性にアクセスするには、以下の関数を使用します:
  `image-exif-data(//MyImages/Image20141130.01)/@GPSLatitude`
  `image-exif-data(//MyImages/Image20141130.01)/@Geolocation`
- 全ての属性にアクセスするには、以下の関数を使用します:
  `image-exif-data(//MyImages/Image20141130.01)/@*`
- 全ての属性の名前にアクセスするには、以下の式を使用します:
  `for $i in image-exif-data(//MyImages/Image20141130.01)/@* return name($i)`
  関数により返される属性の名前を検出するために役に立ちます。

Altova Exif 属性: 位置情報

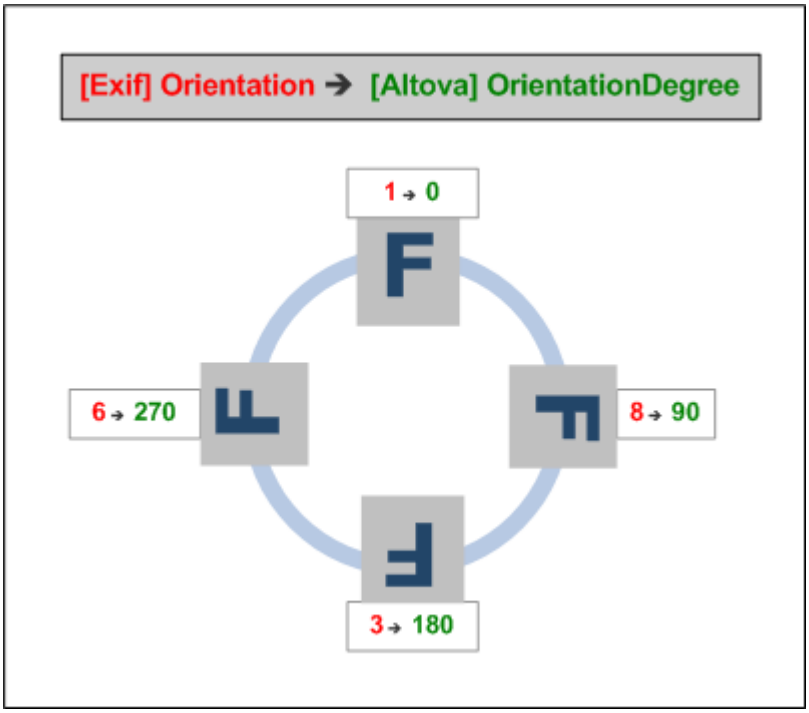
Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータ タグから生成します。**Geolocation** は、4 つの Exif タグの連結です: 単位の追加された (下のテーブル参照) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef 。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

Altova Exif 属性: OrientationDegree

Altova XPath/XQuery エンジンはカスタム属性 **OrientationDegree** を Exif メタデータタグ **Orientation** から生成します。

OrientationDegree は標準 Exif タグ Orientation を正数の値 (1、8、3 または 6) からそれぞれ対応する度数の値 (0, 90, 180, 270) へ下の図に示されているように変換します。2、4、5、7 の Orientation 値の変換はないことに注意してください。(これらの向きはイメージ 1 を垂直方向中央軸で反転して、2 の値を持つイメージを取得します。そして、このイメージを 90-度ごとに時計回りにジャンプさせそれぞれ 7、4、および 5 の値を取得します。)。



標準 Exif メタデータ タグのリスト

- ImageWidth
- ImageLength
- BitsPerSample
- Compression
- PhotometricInterpretation
- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription

(C) 2015-2021 Altova GmbH

- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio

- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

[トップ]

18.2.1.5 XPath/XQuery 関数: 数値

Altova の数値拡張関数は XPath と XQuery 内で使用することができ、データを更に処理するための追加機能を提供します。このセクションの関数は Altova の **XPath 3.0** と **XQuery 3.0** エンジンと使用することができます。XPath/XQuery コンテキストで使用することができます。

(C) 2015-2021 Altova GmbH

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova 拡張関数名前空間、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

自動付番関数

generate-auto-number [altova:]

altova:generate-auto-number(ID *as xs:string*, **StartsWith** *as xs:double*, **Increment** *as xs:double*, **ResetOnChange** *as xs:string*) を xs:integer とする XP1 XP2 XQ1 XP3.1 XQ3.1

関数が呼び出される度に番号を生成します。関数が初めて呼び出される際に生成される最初の番号は StartsWith 引数により指定されます。その後の関数の呼び出しは新しい番号を生成します。この番号は前に生成された番号を、Increment 引数で指定された値ごとにインクリメントします。実質的には、altova:generate-auto-number 関数は、ID 引数により指定されたカウンターを作成し、このカウンターが関数が呼び出されるごとにインクリメントされます。ResetOnChange 引数の値が、前の関数呼び出しから変更されると、生成される番号の値が StartsWith 値にリセットされます。自動付番は altova:reset-auto-number 関数を使用してリセットすることができます。

サンプル

- **altova:generate-auto-number**("ChapterNumber", 1, 1, "SomeString") は、関数が呼び出される度に番号を 1 つ返します。1 から始まり、関数が呼び出される度に 1 ずつインクリメントします。その後の呼び出しの 4 番目の"SomeString"引数がである限り、インクリメントは継続されます。4 番目の引数の値が変更されると、(ChapterNumber と呼ばれる) カウンターは 1 にリセットされます。ChapterNumber の値も altova:reset-auto-number 関数の呼び出しによりの altova:reset-auto-number("ChapterNumber") ようにリセットされます。

reset-auto-number [altova:]

altova:reset-auto-number(ID *as xs:string*) XP1 XP2 XQ1 XP3.1 XQ3.1

この関数は、ID 引数内で名づけられた自動付番カウンターの番号をリセットします。 引数内のカウンター名を作成した関数の引数により指定された数値にリセットされます。altova:generate-auto-number

サンプル

- **altova:reset-auto-number**("ChapterNumber") は、altova:generate-auto-number 関数により作成された ChapterNumber という名の自動付番カウンターをリセットします。
- ChapterNumber を作成した altova:generate-auto-number 関数の StartsWith 引数の値に数値をリセットします。

[[トップ]]

数値関数

hex-string-to-integer [altova:]

altova:hex-string-to-integer(HexString *as xs:string*) を **xs:integer** とする **XP3.1** **XQ3.1**
10 進のシステム(10進)内の正数の16 進の同値の文字列引数を必要とし、10 進の整数を返します。

サンプル

- **altova:hex-string-to-integer**('1') は1 を返します。
- **altova:hex-string-to-integer**('9') は9 を返します。
- **altova:hex-string-to-integer**('A') は10 を返します。
- **altova:hex-string-to-integer**('B') は11 を返します。
- **altova:hex-string-to-integer**('F') は15 を返します。
- **altova:hex-string-to-integer**('G') はエラーを返します。
- **altova:hex-string-to-integer**('10') は16 を返します。
- **altova:hex-string-to-integer**('01') は1 を返します。
- **altova:hex-string-to-integer**('20') は32 を返します。
- **altova:hex-string-to-integer**('21') は33 を返します。
- **altova:hex-string-to-integer**('5A') は90 を返します。
- **altova:hex-string-to-integer**('USA') はエラーを返します。

integer-to-hex-string [altova:]

altova:integer-to-hex-string(Integer *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
正数を引数として必要とし、文字列として自身のベース16 の同値を返します。

サンプル

- **altova:integer-to-hex-string**(1) は'1' を返します。
- **altova:integer-to-hex-string**(9) は'9' を返します。
- **altova:integer-to-hex-string**(10) は'A' を返します。
- **altova:integer-to-hex-string**(11) は'B' を返します。
- **altova:integer-to-hex-string**(15) は'F' を返します。
- **altova:integer-to-hex-string**(16) は'10' を返します。
- **altova:integer-to-hex-string**(32) は'20' を返します。
- **altova:integer-to-hex-string**(33) は'21' を返します。
- **altova:integer-to-hex-string**(90) は'5A' を返します。

[[トップ]]

数値フォーマット関数

[[トップ]]

18.2.1.6 XPath/XQuery 関数: スキーマ

下にリストされる Altova 拡張関数はスキーマの情報を返します。以下は関数の詳細 (i) サンプル(ii) スキーマコンポーネントのリストとパースペクティブのプロパティです。 Altova の**XPath 3.0** と**XQuery 3.0** エンジンと共に使用することができ XPath/XQuery コンテキスト内で見つけることができます。

スキーマドキュメントからのスキーマ情報
関数 **altova:schema** には以下の2つの引数が存在します: 1つのゼロ引数と他の2つの引数。ゼロ引数関数はスキーマ全体を返します。この開始点からスキーマ内をナビゲートし必要とするスキーマコンポーネントをロケートします。2-引数関数は自身のQNameにより識別される特定のコンポーネントの型を返します。両方の場合、戻り値は関数です。返されたコンポーネント内をナビゲートするには、特定のコンポーネントのプロパティを選択する必要があります。プロパティが非動的アイテム(すなわちコンポーネントの場合) このコンポーネントのプロパティをさらに選択してナビゲートすることが可能です。選択されたプロパティが動的なアイテムの場合、アイテムの値が返され、ナビゲートを更に行うことはできません。

メモ Xquery 式ではスキーマは明示的にインポートされる必要があります。XPath 式では、スキーマは処理環境にインポートされる必要があります。例えば、XSLT には**xslt:import** 命令を使用してインポートします。

XML ノードからのスキーマ情報
関数 **altova:type** はXMLドキュメントのノードを送信しノードの型情報をPSVI から返します。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

▼ Schema (引数無し)

altova:schema() as (function(xs:string) as item()*)? **XP3.1** **XQ3.1**
schema コンポーネント全体を返します。**schema** コンポーネントを **schema** コンポーネントのプロパティの一つを選択して更にナビゲートすることができます。

- このプロパティがコンポーネントの場合、このコンポーネントのプロパティの1つを選択して更に深く他のステップをナビゲートすることができます。このステップは更にスキーマをナビゲートするために繰り返すことができます。
- コンポーネントが動的な値の場合、動的な値が返され更にナビゲートすることはできません。

schema コンポーネントのプロパティ:

```
"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"
```

(**schema** 以外の)他のすべてのコンポーネント型のプロパティが下にリストされています。

メモ Xquery 式ではスキーマは明示的にインポートされる必要があります。XPath 式では、スキーマは処理環境にインポートされる必要があります。例えば、XSLT には**xslt:import** 命令を使用してインポートします。

⊟ *サンプル*

- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema() ("type definitions") return $typedef ("name")` はスキーマ内のすべての単純型または複合型の名前を返します
- `import schema "" at "C:\Test\ExpReport.xsd"; altova:schema() ("type definitions")[1]("name")` はスキーマ内の単純型または複合型の最初の名前を返します

コンポーネントとそのプロパティ

Assertion

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Assertion"
test	XPath プロパティレコード	

Attribute Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Declaration"
name	文字列	属性のローカル名
target namespace	文字列	属性の名前空間 URI
type definition	Simple Type または Complex Type	
scope	プロパティを持つ関数 (″class″:″Scope″, "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group)	
value constraint	存在する場合、プロパティを持つ関数 (″class″: ″Value Constraint″、″variety″: ″fixed″ または″default″, ″value″: atomic value, ″lexical form″: string。″value″ プロパティは namespace-sensitive 型のために使用することはできないことに注意してください。	
inheritable	ブール値	

Attribute Group Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Group Definition"
name	文字列	属性グループのローカル名
target namespace	文字列	属性グループの名前空間 URI
attribute uses	(Attribute Use) のシーケンス	
attribute wildcard	任意の属性のワイルドカード	

Attribute Use

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Use"
required	ブール値	属性が必要な場合はtrue 、任意の場合はfalse
value constraint	Attribute Declaration を参照	
inheritable	ブール値	

Attribute Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

Complex Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
base type definition	Complex Type Definition	
final	文字列のシーケンス ("restriction"|"extension")	
context	空のシーケンス(not implemented)	
derivation method	文字列 (″restriction″|″extension″)	
abstract	ブール値	
attribute uses	Attribute Use のシーケンス	
attribute wildcard	任意の属性のワイルドカード	
content type	プロパティを持つ関数: (″class″:″Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited	文字列のシーケンス	

substitutions	("restriction"|"extension")	
assertions	アサーションのシーケンス	

Element Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
type definition	Simple Type または Complex Type	
type table	プロパティを持つ関数 (″class″:″Type Table″, "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type)	
scope	プロパティを持つ関数 (″class″:″Scope″, "variety": ("global"|"local"), "parent": optional Complex Type)	
value constraint	Attribute Declaration を参照	
nillable	ブール値	
identity-constraint definitions	Identity Constraint のシーケンス	
substitution group affiliations	Element Declaration のシーケンス	
substitution group exclusions	文字列のシーケンス ("restriction"|"extension")	
disallowed substitutions	文字列のシーケンス ("restriction"|"extension"|"substitution")	
abstract	ブール値	

Element Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

Facet

プロパティ名	プロパティ型	プロパティ値

kind	文字列	ファセットの名前、例えば、"minLength" または "enumeration"
value	ファセットによる	ファセットの値
fixed	ブール値	
typed-value	ファセットの列挙のみ、Array(xs:anyAtomicType	列挙値を含む配列。それぞれがアトミック値のシーケンスである場合があります。(メモ列挙ファセットに関しては、実際の型にかかわらず "value" 値プロパティは文字列のシーケンスです)

Identity Constraint

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Identity-Constraint Definition"
name	文字列	制約のローカル名
target namespace	文字列	制約の名前空間 URI
identity-constraint category	文字列 ("key"|"unique"|"keyRef")	
selector	XPath プロパティレコード	
fields	XPath プロパティレコード のシーケンス	
referenced key	(keyRef のためのみ): Identity Constraint	対応するキー制約

Model Group

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group"
compositor	文字列 ("sequence"|"choice"|"all")	
particles	Particle のシーケンス	

Model Group Definition

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group Definition"
name	文字列	モデルグループのローカル名
target namespace	文字列	モデルグループの名前空間 URI
model group	Model Group	

Notation

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Notation Declaration"
name	文字列	表記のローカル名

target namespace	文字列	表記の名前空間 URI
system identifier	anyURI	
public identifier	文字列	

Particle

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Particle"
min occurs	整数	
max occurs	整数または文字列("unbounded")	
term	Element Declaration、Element Wildcard、または ModelGroup	

Simple Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Simple Type Definition"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
final	文字列のシーケンス ("restriction"|"extension"|"list"|"union")	
context	含まれるコンポーネント	
base type definition	Simple Type	
facets	Sequence of Facet	
fundamental facets	(実装されていない)空のシーケンス	
variety	文字列 ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(リスト型のみのための) Simple Type	
member type definitions	(ユニオン型のための) Simple Type のシーケンス	

Type Alternative

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Type Alternative"
test	XPath プロパティレコード	
type definition	Simple Type または Complex Type	

XPath Property Record

プロパティ名	プロパティ型	プロパティ値

(C) 2015-2021 Altova GmbH

namespace bindings	プロパティを持つ関数のシーケンス(″prefix″: string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	XPath 式の静的ベースURI
expression	文字列	文字列としてのXPath 式

▼ Schema (2つの引数)

altova:schema(ComponentKind *as xs:string*, **Name** *as xs:QName*) **as (function(xs:string) as item()*)?** **XP3.1** **XQ3.1**

2番目の引数内で与えられている名前と同じ名前を持つ最初の引数内で指定されているコンポーネントの型を返します。コンポーネントのプロパティの一つを選択して更にナビゲートすることができます。

- このプロパティがコンポーネントの場合、このコンポーネントのプロパティの1つを選択して更に深く他のステップをナビゲートすることができます。このステップは更にスキーマをナビゲートするために繰り返すことができます。
- コンポーネントが動的な値の場合、動的な値が返され更にナビゲートすることはできません。

メモ Xquery 式ではスキーマは明示的にインポートされる必要があります。XPath 式では、スキーマは処理環境にインポートされる必要があります。例えば、XSLT には**xslt:import** 命令を使用してインポートします。

サンプル

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  altova:schema("element declaration", xs:QName("OrgChart"))("type definition")
  ("content type")("particles")[3]!.("term")("kind")
  ```
  は3番目の`particles` コンポーネントの `kind` プロパティを返します。 このparticles コンポーネントは`OrgChart` の `Qname` を持つ要素宣言の子孫です。

- ```
  import schema "" at "C:\Test\ExpReport.xsd";
  let $typedef := altova:schema("type definition", xs:QName("emailType"))
  for $facet in $typedef ("facets")
  return [$facet ("kind"), $facet("value")]
  ```
  は各 **emailType** コンポーネントの各 **facet** にそのファセットの型と値を含む列挙を返します。

コンポーネントとそのプロパティ

Assertion

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Assertion"
test	XPath プロパティレコード	

Attribute Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Declaration"
name	文字列	属性のローカル名

target namespace	文字列	属性の名前空間 URI
type definition	Simple Type または Complex Type	
scope	プロパティを持つ関数 (″class″:″Scope″, "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group)	
value constraint	存在する場合、プロパティを持つ関数 (″class″: ″Value Constraint″、″variety″: ″fixed″ または″default″, ″value″: atomic value, ″lexical form″: string。″value″ プロパティは namespace-sensitive 型のために使用することはできないことに注意してください。	
inheritable	ブール値	

Attribute Group Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Group Definition"
name	文字列	属性グループのローカル名
target namespace	文字列	属性グループの名前空間 URI
attribute uses	(Attribute Use) のシーケンス	
attribute wildcard	任意の属性のワイルドカード	

Attribute Use

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Use"
required	ブール値	属性が必要な場合は true 、任意の場合は false
value constraint	Attribute Declaration を参照	
inheritable	ブール値	

Attribute Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

(C) 2015-2021 Altova GmbH

Complex Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
base type definition	Complex Type Definition	
final	文字列のシーケンス ("restriction"|"extension")	
context	空のシーケンス(not implemented)	
derivation method	文字列 ("restriction"|"extension")	
abstract	ブール値	
attribute uses	Attribute Use のシーケンス	
attribute wildcard	任意の属性のワイルドカード	
content type	プロパティを持つ関数: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	文字列のシーケンス ("restriction"|"extension")	
assertions	アサーションのシーケンス	

Element Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
type definition	Simple Type または Complex Type	
type table	プロパティを持つ関数 ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type)	
scope	プロパティを持つ関数 ("class":"Scope", "variety": ("global"|"local"), "parent": optional Complex Type)	
value constraint	Attribute Declaration を参照	
nillable	ブール値	
identity-constraint	Identity Constraint のシーケンス	

definitions		
substitution group affiliations	Element Declaration のシーケンス	
substitution group exclusions	文字列のシーケンス ("restriction"|"extension")	
disallowed substitutions	文字列のシーケンス ("restriction"|"extension"|"substitution")	
abstract	ブール値	

Element Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

Facet

プロパティ名	プロパティ型	プロパティ値
kind	文字列	ファセットの名前、例えば、″minLength″ または″enumeration″
value	ファセットによる	ファセットの値
fixed	ブール値	
typed-value	ファセットの列挙のみ、Array(xs:anyAtomicType	列挙値を含む配列。それぞれがアトミック値のシーケンスである場合があります。(メモ 列挙ファセットに関しては、実際の型にかかわらず″value″ 値プロパティは文字列のシーケンスです)

Identity Constraint

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Identity-Constraint Definition"
name	文字列	制約のローカル名
target namespace	文字列	制約の名前空間 URI
identity-constraint category	文字列 (″key″|″unique″|″keyRef″)	
selector	XPath プロパティレコード	
fields	XPath プロパティレコード のシーケンス	

(C) 2015-2021 Altova GmbH

referenced key	(keyRef のためのみ): Identity Constraint	対応するキー制約

Model Group

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group"
compositor	文字列 ("sequence"|"choice"|"all")	
particles	Particle のシーケンス	

Model Group Definition

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group Definition"
name	文字列	モデルグループのローカル名
target namespace	文字列	モデルグループの名前空間 URI
model group	Model Group	

Notation

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Notation Declaration"
name	文字列	表記のローカル名
target namespace	文字列	表記の名前空間 URI
system identifier	anyURI	
public identifier	文字列	

Particle

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Particle"
min occurs	整数	
max occurs	整数または文字列("unbounded")	
term	Element Declaration、Element Wildcard、または ModelGroup	

Simple Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Simple Type Definition"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
final	文字列のシーケンス	

	("restriction"|"extension"|"list"|"union")	
context	含まれるコンポーネント	
base type definition	Simple Type	
facets	Sequence of Facet	
fundamental facets	(実装されていない)空のシーケンス	
variety	文字列 ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(リスト型のみのための) Simple Type	
member type definitions	(ユニオン型のための) Simple Type のシーケンス	

Type Alternative

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Type Alternative"
test	XPath プロパティレコード	
type definition	Simple Type または Complex Type	

XPath Property Record

プロパティ名	プロパティ型	プロパティ値
namespace bindings	プロパティを持つ関数のシーケンス("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	XPath 式の静的ベース URI
expression	文字列	文字列としての XPath 式

型

altova:type(Node *as item?*) **as (function(xs:string) as item()*)? XP3.1 XQ3.1**
関数 **altova:type** は XML ドキュメントの要素または属性ノードを送信しノードの型情報を PSVI から返します。

メモ スキーマが参照可能になるために XML ドキュメントはスキーマ宣言を持つ必要があります。

サンプル

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type
  ```
  はノードの型情報を含む関数を返します。

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type ("kind")
  ```


(C) 2015-2021 Altova GmbH

はノードの型のコンポーネント（単純型または複合型）を取りコンポーネントの`kind` property の値を返します。

コンポーネントとそのプロパティ

- Assertion

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Assertion"
test	XPath プロパティレコード	

- Attribute Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Declaration"
name	文字列	属性のローカル名
target namespace	文字列	属性の名前空間 URI
type definition	Simple Type または Complex Type	
scope	プロパティを持つ関数（"class":"Scope", "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group)	
value constraint	存在する場合、プロパティを持つ関数（"class": "Value Constraint"、"variety": "fixed" または "default", "value": atomic value, "lexical form": string。"value" プロパティは namespace-sensitive 型のために使用することはできないことに注意してください。	
inheritable	ブール値	

- Attribute Group Declaration

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Group Definition"
name	文字列	属性グループのローカル名
target namespace	文字列	属性グループの名前空間 URI
attribute uses	(Attribute Use) のシーケンス	
attribute wildcard	任意の属性のワイルドカード	

- Attribute Use

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Attribute Use"
required	ブール値	属性が必要な場合は true 、任意の場合は false

value constraint	Attribute Declaration を参照	
inheritable	ブール値	

Attribute Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

Complex Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
base type definition	Complex Type Definition	
final	文字列のシーケンス ("restriction"|"extension")	
context	空のシーケンス(not implemented)	
derivation method	文字列 (″restriction″|″extension″)	
abstract	ブール値	
attribute uses	Attribute Use のシーケンス	
attribute wildcard	任意の属性のワイルドカード	
content type	プロパティを持つ関数: (″class″:″Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	文字列のシーケンス ("restriction"|"extension")	
assertions	アサーションのシーケンス	

Element Declaration

(C) 2015-2021 Altova GmbH

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Complex Type"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
type definition	Simple Type または Complex Type	
type table	プロパティを持つ関数 (″class″:″Type Table″, "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type)	
scope	プロパティを持つ関数 (″class″:″Scope″, "variety": ("global"|"local"), "parent": optional Complex Type)	
value constraint	Attribute Declaration を参照	
nillable	ブール値	
identity-constraint definitions	Identity Constraint のシーケンス	
substitution group affiliations	Element Declaration のシーケンス	
substitution group exclusions	文字列のシーケンス ("restriction"|"extension")	
disallowed substitutions	文字列のシーケンス ("restriction"|"extension"|"substitution")	
abstract	ブール値	

- Element Wildcard

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Wildcard"
namespace constraint	プロパティを持つ関数 (″class″: ″Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings"	
process contents	文字列 (″strict″|″lax″|″skip″)	

- Facet

プロパティ名	プロパティ型	プロパティ値
kind	文字列	ファセットの名前、例えば、″minLength″ または″enumeration″
value	ファセットによる	ファセットの値
fixed	ブール値	

typed-value	ファセットの列挙のみ、Array(xs:anyAtomicType	列挙値を含む配列。それぞれがアトミック値のシーケンスである場合があります。（メモ列挙ファセットに関しては、実際の型にかかわらず "value" 値プロパティは文字列のシーケンスです）

Identity Constraint

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Identity-Constraint Definition"
name	文字列	制約のローカル名
target namespace	文字列	制約の名前空間 URI
identity-constraint category	文字列（"key"|"unique"|"keyRef"）	
selector	XPath プロパティレコード	
fields	XPath プロパティレコード のシーケンス	
referenced key	(keyRef のためのみ): Identity Constraint	対応するキー制約

Model Group

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group"
compositor	文字列（"sequence"|"choice"|"all"）	
particles	Particle のシーケンス	

Model Group Definition

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Model Group Definition"
name	文字列	モデルグループのローカル名
target namespace	文字列	モデルグループの名前空間 URI
model group	Model Group	

Notation

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Notation Declaration"
name	文字列	表記のローカル名
target namespace	文字列	表記の名前空間 URI
system identifier	anyURI	
public identifier	文字列	

Particle

(C) 2015-2021 Altova GmbH

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Particle"
min occurs	整数	
max occurs	整数または文字列("unbounded")	
term	Element Declaration、Element Wildcard、または ModelGroup	

Simple Type

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Simple Type Definition"
name	文字列	(匿名の場合は空)型のローカル名
target namespace	文字列	(匿名の場合は空)型の名前空間 URI
final	文字列のシーケンス ("restriction"|"extension"|"list"|"union")	
context	含まれるコンポーネント	
base type definition	Simple Type	
facets	Sequence of Facet	
fundamental facets	(実装されていない)空のシーケンス	
variety	文字列 ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(リスト型のみのための) Simple Type	
member type definitions	(ユニオン型のための) Simple Type のシーケンス	

Type Alternative

プロパティ名	プロパティ型	プロパティ値
kind	文字列	"Type Alternative"
test	XPath プロパティレコード	
type definition	Simple Type または Complex Type	

XPath Property Record

プロパティ名	プロパティ型	プロパティ値
namespace bindings	プロパティを持つ関数のシーケンス("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	XPath 式の静的ベース URI
expression	文字列	文字列としての XPath 式

18.2.1.7 XPath/XQuery 関数: シーケンス

Altova のシーケンス拡張関数はXPath とXQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の**XPath 3.0** および**XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3.1
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3.1

attributes [altova:]

altova:attributes(AttributeName *as xs:string*) を**attribute()*** とする XP3.1 XQ3.1
入力引数 AttributeName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、attribute:: 軸に対して行われます。これは、コンテキストノードが親要素ノードである必要があることを意味します。

サンプル

- **altova:attributes**("MyAttribute") はMyAttribute()* を返します。

altova:attributes(AttributeName *as xs:string*, **SearchOptions** *as xs:string*)
asattribute()* XP3.1 XQ3.1
入力引数 AttributeName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、attribute:: 軸に対して行われます。コンテキストノードが親要素ノードである必要があります。第 2 引数はオプションのフラッグを含みます。使用することのできるフラッグは以下の通りです:
r = 正規表現検索を切り替えます; AttributeName は正規表現検索文字列である必要があります;
f = このオプションが指定されている場合、AttributeName は完全一致を提供します。それ以外の場合は、AttributeName は属性名に部分的に一致するとその属性を返します。例えば **f** が指定されていない場合、MyAtt は MyAttribute を返します。
i = 大文字と小文字を一致させる検索に切り替えます。
p = 検索に名前空間プレフィックスを含みます。AttributeName は、名前空間プレフィックスを含みます。例えば altova:MyAttribute。
フラッグは順序に関わりなく書き込むことができます。無効なフラッグはエラーを生成します。1 つまたは複数のフラッグを省略することができます。空の文字列は許可されていますが、引数を1 つしか持たない関数と同じ効果を持ちます(*前の署名*)、ですが、空のシーケンスは第 2 の引数としては許可されていません。

サンプル

- **altova:attributes**("MyAttribute", "rfip") はMyAttribute()* を返します。

- **altova:attributes**("MyAttribute", "pri") はMyAttribute()* を返します。
- **altova:attributes**("MyAtt", "rip") はMyAttribute()* を返します。
- **altova:attributes**("MyAttributes", "rfip") は不一致を返します。
- **altova:attributes**("MyAttribute", "") はMyAttribute()* を返します。
- **altova:attributes**("MyAttribute", "Rip") 認識されないフラッグエラーを返します。
- **altova:attributes**("MyAttribute",) 見つからない第 2 引数を返します。

elements [altova:]

altova:elements(ElementName *as xs:string*) を**element()*** とする **XP3.1** **XQ3.1**
入力引数 ElementName で与えられた名前と同じローカル名を持つすべての要素を返します。検索は大文字と小文字を区別して child:: 軸に対して実行されます。コンテキストノードは、検索される要素の親ノードである必要があります。

サンプル

- **altova:elements**("MyElement") はMyElement()* を返します。

altova:elements(ElementName *as xs:string*, **SearchOptions** *as xs:string*) **aselement()*** **XP3.1** **XQ3.1**
入力引数 ElementName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、child:: 軸に対して行われます。コンテキストノードが親要素ノードである必要があります。第 2 引数はオプションのフラッグを含みます。使用することのできるフラッグは以下の通りです:
r = 正規表現検索を切り替えます; ElementName は正規表現検索文字列である必要があります;
f = このオプションが指定されている場合、ElementName は完全一致を提供します。それ以外の場合は、ElementName は属性名に部分的に一致するとその属性を返します。例えば **f** が指定されていない場合、MyElem はMyElement を返します。
i = 大文字と小文字を一致させる検索に切り替えます。
p = 検索に名前空間プレフィックスを含みます。ElementName は、名前空間プレフィックスを含みます。例えば altova:MyElement。
フラッグは順序に関わりなく書き込むことができます。無効なフラッグはエラーを生成します。1 つまたは複数のフラッグを省略することができます。空の文字列は許可されていますが、引数を1 つしか持たない関数と同じ効果を持ちます(*前の署名*)、ですが、空のシーケンスは第 2 の引数としては許可されていません。

サンプル

- **altova:elements**("MyElement", "rip") はMyElement()* を返します。
- **altova:elements**("MyElement", "pri") はMyElement()* を返します。
- **altova:elements**("MyElement", "") はMyElement()* を返します。
- **altova:attributes**("MyElem", "rip") はMyElement()* を返します。
- **altova:attributes**("MyElements", "rfip") 不一致を課wします。
- **altova:elements**("MyElement", "Rip") 認識されないフラッグエラーを返します。
- **altova:elements**("MyElement",) 見つからない第 2 引数を返します。

find-first [altova:]

altova:find-first((Sequence *as item()**), **(Condition(Sequence-Item** *as xs:boolean*)) を **item()?** とする **XP3.1** **XQ3.1**
この関数は2 つの引数を必要とします。最初の引数は1 つ、または1 つ以上のデータ型のアイテムのシーケンスです。第 2 の引数 Condition は、(1 のアリティを持つ) 1 つの引数 を必要とし、boolean を返す XPath 関数に対する参照です。
Condition で参照された関数の代わりに、**Sequence** の各アイテムが提出されます。(*注意:* この関数は1 つの引数のみを必要とします。) **Condition** 内の関数に**true()**と評価させる最初の **Sequence** アイテムは、**altova:find-first**、反復の終了の結果として返されます。

サンプル

- **altova:find-first**(5 to 10, function($a) {$a mod 2 = 0}) は xs:integer 6 を返します。**Condition** 引数は、**$a** という名のインライン関数を宣言し、定義します。XPath 3.0 インライン関数 **function()** を参照します。**altova: Sequence** 引数内の各アイテムでは、**find-first** がパスされ、代わりに、**$a** を入力値とします。入力値は関数定義($a mod 2 = 0)内の条件に対してテストされます。この条件を満たす最初の入力値が **altova:find-first**（この場合は 6）の結果として返されます。

- **altova:find-first**((1 to 10), (function($a) {$a+3=7})) は xs:integer 4 を返します。

更なるサンプル
ファイル **C:\Temp\Customers.xml** が存在する場合:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は xs:string C:\Temp\Customers.xml を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** が存在する場合:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は xs:string http://www.altova.com/index.html を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** も存在しない場合:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) 結果無しを返します。

上のサンプルについての注意点

- XPath 3.0 関数 doc-available は URI として使用され、ドキュメントノードが提出された URI で検出される場合 true を返する単一の引数を必要とします。（ですから、提出された URI でのドキュメントは XML ドキュメントである必要があります。）
- doc-available 関数は、altova:find-first の第 2 引数である、**Condition** で使用することができます。これは、1 つの引数（アリティ=1）のみを必要とするからであり、item() を入力（ URI として使用される文字列）として、boolean の値を返すからです。
- doc-available 関数は、参照されているだけで、呼び出されていない点に注意してください。アタッチされている #1 サフィックスは関数が 1 つのアリティであることを表示するためです。doc-available#1 の意味は以下のとおりです: アリティ=1 を持つ doc-availabe() 関数を使用し、最初のシーケンスの各アイテムの代わりに、単一引数として パスします。この結果、 2 つの文字列の各自つは、文字列をURIとして使用し、URIにドキュメントノードが存在するかテストする **doc-available()** にパスされます。1 つが各当する場合、**doc-available()** 関数は true() を評価し、シーケンス内のその文字列のインデックス ポジションは、 **altova:find-first** 関数の結果として返されます。doc-available() 関数に関しての注意点: 相対パスは、デフォルトで関数がロードされる XML ドキュメントの現在のベース URI に対して相対的に解決されます。

find-first-combination [altova:]

altova:find-first-combination((Seq-01 *as item()**), **(Seq-02** *as item()**), **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean*)) を **item()*** とする **XP3.1** **XQ3.1**
この関数は 3 つの引数を必要とします:

- 最初の2 つの引数 Seq-01 とSeq-02, は、1 つまたは1 つ以上のデータ型のアイテムです。
- 第 3 の引数 Condition は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 とSeq-02 のアイテムは指定された組み合わせ(各シーケンスからの1 つずつにアイテムで構成されるペア)で、Condition 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

Condition 関数にtrue() と評価するように指示する最初のペアはaltova:find-first-combination の結果として返されます。以下の点に注意してください: (i) Condition 関数が提出された引数ペア内で繰り返され、true()を評価しない場合、altova:find-first-combination は、結果を返しません; (ii) altova:find-first-combination の結果が常に(データ型の)アイテムのペアである場合またはアイテムでない場合。

サンプル

- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) はxs:integers (11, 21) のシーケンスを返します。
- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) はxs:integers (11, 22) のシーケンを返します。
- altova:find-first-combination(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) は xs:integers (11, 23) のシーケンを返します。

find-first-pair [altova:]

altova:find-first-pair((Seq-01 *as item()**), (Seq-02 *as item()**), (Condition(Seq-01-Item, Seq-02-Item *as xs:boolean*)) をitem()* とする XP3.1 XQ3.1
この関数は3 つの引数を必要とします:

- 最初の2 つの引数 Seq-01 とSeq-02, は、1 つまたは1 つ以上のデータ型のアイテムです。
- 第 3 の引数 Condition は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 とSeq-02 のアイテムは指定された組み合わせで、Condition 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Condition 関数にtrue() と評価するように指示する最初のペアはaltova:find-first-pair の結果として返されます。以下の点に注意してください: (i) Condition 関数が提出された引数ペア内で繰り返され、true()を評価しない場合、altova:find-first-pair は、結果を返しません; (ii) altova:find-first-combination の結果が常に(データ型の) アイテムのペアである場合またはアイテムでない場合。

サンプル

- altova:find-first-pair(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) は xs:integers (11, 21) のシーケンスを返します。
- altova:find-first-pair(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) は*結果無し*を返します。

上の2 つのサンプルに表示される通り、ペアの順序は以下の通りです: (11, 21) (12, 22) (13, 23)...(20,

30) 。(33 を返す指示されたペアがないため) この理由で第 2 のサンプルは *結果無し*を返します。

find-first-pair-pos [altova:]

altova:find-first-pair-pos((Seq-01 *as item()**)**, (Seq-02** *as item()**)**, (Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean*)**)** を**xs:integer** とする **XP3.1** **XQ3.1**
この関数は3 つの引数を必要とします:

- 最初の2 つの引数 **Seq-01** と**Seq-02**, は、1 つまたは1 つ以上のデータ型のアイテムです。
- 第 3 の引数 **Condition** は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 と**Seq-02** のアイテムは指定された組み合わせで、**Condition** 内の関数の引数として パスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Condition 関数に**true()**を評価させる、最初に指示されたペアのインデックス ポジションは、は**altova:find-first-pair-pos** の結果として返されます。関数が提出された引数ペア内で繰り返され、**true()**を一度も評価しない場合、**altova:find-first-pair-pos** *結果無し*が返します。

サンプル

- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) は 1 を返します
- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) は*結果無し*を返します。

上の2 つのサンプルに表示される通り、ペアの順序は以下の通りです: (11, 21) (12, 22) (13, 23)...(20, 30)。最初のサンプルでは、最初のペアは**Condition** 関数に**true()**を評価させ、シーケンス内のインデックス ポジションには**1** が返されます。第 2 のサンプルでは、 33 を返すペアがないため、*結果無し*が返します。

find-first-pos [altova:]

altova:find-first-pos((Sequence *as item()**)**, (Condition(Sequence-Item** *as xs:boolean*)**)** を**xs:integer** とする **XP3.1** **XQ3.1**
この関数は2 つの引数を必要とします。最初の引数は1 つ、または1 つ以上のデータ型のアイテムのシーケンスです。第 2 の引数 Condition は、(1 のアリティを持つ) 1 つの引数 を必要とし、boolean を返す XPath 関数に対する参照です。Condition で参照された関数の代わりに、**Sequence** の各アイテムが提出されます。(*注意:* この関数は1 つの引数のみを必要とします。) **Condition** 内の関数に**true()**と評価させる最初の **Sequence** アイテムは、**altova:find-first-pos** の結果として返された**Sequence** 内インデックス ポジションを持ちます。

サンプル

- **altova:find-first-pos**(5 to 10, function($a) {$a mod 2 = 0}) は**xs:integer** 2 を返します。
  Condition 引数は、**$a** という名のインライン関数を宣言し、定義します。XPath 3.0 インライン関数 **function()**を参照します。 **Sequence** 引数内の各アイテムでは、**find-first-pos** がパスされ、代わりに、**$a** を入力値とします。入力値は関数定義($a mod 2 = 0)内の条件に対してテストされます。この条件を満たす最初の入力値が **altova:find-first-pos**((シーケンス内で条件を満たす最初の値である6 がシーケンスのインデックス位置 2 にあるためこの場合は2、) の結果として返されます。

- **altova:find-first-pos**((2 to 10), (function($a) {$a+3=7})) は xs:integer 3 を返します。

更なるサンプル
ファイル **C:\Temp\Customers.xml** が存在する場合:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は 1 を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** が存在する場合:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) は 2 を返します。

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** も存在しない場合:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) 結果無しを返します。

上のサンプルについての注意点

- XPath 3.0 関数 doc-available は URI として使用され、ドキュメントノードが提出された URI で検出される場合 true を返する単一の引数を必要とします。(ですから、提出された URI でのドキュメントは XML ドキュメントである必要があります。)
- doc-available 関数は、altova:find-first-pos の第 2 引数である、**Condition** で使用することができます。これは、1 つの引数 (アリティ=1) のみを必要とするからであり、item() を入力(URI として使用される文字列) として、boolean の値を返すからです。
- doc-available 関数は、参照されているだけで、呼び出されていない点に注意してください。アタッチされている #1 サフィックスは関数が 1 つのアリティであることを表示するためです。doc-available#1 の意味は以下のとおりです: アリティ=1 を持つ doc-availabe() 関数を使用し、最初のシーケンスの各アイテムの代わりに、単一引数として パスします。 この結果、 2 つの文字列の各自つは、文字列をURIとして使用し、URIにドキュメントノードが存在するかテストする **doc-available()** にパスされます。1 つが各当する場合、**doc-available()** 関数は true() を評価し、シーケンス内のその文字列のインデックス ポジションは、 **altova:find-first-pos** 関数の結果として返されます。doc-available() 関数に関しての注意点: 相対パスは、デフォルトで関数がロードされる XML ドキュメントの現在のベース URI に対して相対的に解決されます。

for-each-attribute-pair [altova:]

altova:for-each-attribute-pair(Seq1 *as element()?*, **Seq2** *as element()?*, **Function** *as function()*) **asitem()*** **XP3.1** **XQ3.1**

ペアの一つの属性が最初の要素を取得し、もう一つの属性が2番目の要素から取得された箇所で、最初の2つの引数は、2つの要素、および、属性ペアを作成するために使用された属性を識別します。要素ペアは同じ名前を持つことをベースに選択され、ペアは名前別に、アルファベット順にセットに並べ替えられます。属性のために他の要素上に対応する属性が存在しない場合、ペアのジョイントは解除されます。これは、一つのメンバーにより飲み構成されることを意味します。関数アイテム(3番目の引数 Function) はアイテムのシーケンスである出力であるペアのシーケンス内のペア(ジョイント、ジョイントの解除) に個別に適用されます。

サンプル

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, function($a, $b){$a+b}) は以下 を返します。...

```
(2, 4, 6) if
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(2, 4, 6) if
<Test-A att2="2" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />

(2, 6) if
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />
```

メモ: 結果 (2, 6) は次のアクションにより取得されます: (1+1, ()+2, 3+3, 4+()) 。演算子の一つが空のシーケンスの場合、アイテム 2 と 4 の場合同様、結果は空のシーケンスになります。

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, concat#2) は以下を返します。

```
(11, 22, 33) if
<Test-A att1="1" att2="2" att3="3" />
<Test-B att1="1" att2="2" att3="3" />

(11, 2, 33, 4) if
<Test-A att4="4" att1="1" att3="3" />
<Test-B att3="3" att2="2" att1="1" />
```

for-each-combination [altova:]

altova:for-each-combination(FirstSequence *as item()**, **SecondSequence** *as item()**, **function($i,$j){$i || $j}) を item()* とする** **XP3.1** **XQ3.1**

第 3 の引数として与えられた、インライン関数を返します。FirstSequence 内の各 $i を SecondSequence内の各 $j と結合します。出力はこれらのアイテムのシーケンスです。

サンプル

- **altova:for-each-combination**(('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j}) は ('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3') を返します。

for-each-combination [altova:]

altova:for-each-combination(FirstSequence *as item()**, **SecondSequence** *as item()**, **Function($i,$j){$i || $j}) as item()*** **XP3.1** **XQ3.1**

最初の2 つの引数内の2 つのシーケンスのアイテムは最初のシーケンスの各アイテムが2 番目のシーケンスになるように結合されるように結合されます。3 番目の引数として与えられる関数は結果のシーケンス内の組み合わせに適用され、アイテムのシーケンスである出力内で出力されます（*サンプル参照*）。

サンプル

- **altova:for-each-combination**(('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j}) は ('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3') を返します。

▼ substitute-empty [altova:]

altova:substitute-empty(FirstSequence *as item()**, **SecondSequence** *as item()*) を **item()*** とする **XP3.1** **XQ3.1**
FirstSequence が空の場合、SecondSequence を返します。FirstSequence が空でない場合、FirstSequence を返します。

サンプル

- **altova:substitute-empty**((1,2,3), (4,5,6)) は (1,2,3) を返します。
- **altova:substitute-empty**((), (4,5,6)) は (4,5,6) を返します。

18.2.1.8 XPath/XQuery 関数: 文字列

Altova の文字列拡張関数は XPath と XQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の **XPath 3.0** および **XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数は XPath/XQuery 式で使用することができ、XPath、XQuery、および XSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は **Altova 拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内の XPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内の XPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内の XQuery 式で使用):	**XQ1** **XQ3.1**

▼ camel-case [altova:]

altova:camel-case(InputString *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
入力文字列を **InputString** をキャメルケースで返します。文字列は(空白スペースのショートカットである) 正規表現 **'\s'** を使用して分析されます。空白文字または連続する空白文字のシーケンスの後の最初の非空白スペース文字は大文字です。出力文字列の最初の文字は大文字です。

サンプル

- **altova:camel-case**("max") Max を返します。
- **altova:camel-case**("max max") Max Max を返します。
- **altova:camel-case**("file01.xml") File01.xml を返します。
- **altova:camel-case**("file01.xml file02.xml") File01.xml File02.xml を返します。
- **altova:camel-case**("file01.xml file02.xml") File01.xml File02.xml を返します。
- **altova:camel-case**("file01.xml -file02.xml") File01.xml -file02.xml を返します。

altova:camel-case(InputString *as xs:string*, **SplitChars** *as xs:string*, **IsRegex** *as xs:boolean*) を **xs:string** とする **XP3.1** **XQ3.1**
SplitChars を使用して、次の大文字をトリガーする文字を決定し、入力文字列を **InputString** キャメルケースに変換しま

す。 **SplitChars** は、**IsRegex = true()** の場合、または、**IsRegex = false()** の場合、プレーン文字は正規表現として使用されます。出力文字列の最初の文字は大文字です。

サンプル

- **altova:camel-case**("setname getname", "set|get", true()) setName getName を返します。
- **altova:camel-case**("altova\documents\testcases", "\", false()) Altova\Documents\Testcases を返します。

char [altova:]

altova:char(Position *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**

xs:string.Position に対してのコンテキスト アイテムの値を変換することにより得られた文字列内の Position 引数により指定されたポジションにある文字を含む文字列を返します。引数により提出されたインデックスに文字が存在しない場合、結果文字列は空です。

サンプル

コンテキスト アイテムが **1234ABCD** の場合:

- **altova:char**(2) は 2 を返します。
- **altova:char**(5) は A を返します。
- **altova:char**(9) は空の文字列を返します。
- **altova:char**(-2) は空の文字列を返します。

altova:char(InputString *as xs:string*, **Position** *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**

引数として提出された文字列内の Position 引数により指定されたポジションでの文字を含む文字列を返します。Position 引数により提出されたインデックスに文字が存在しない場合、結果文字列は空です。

サンプル

- **altova:char**("2014-01-15", 5) は - を返します。
- **altova:char**("USA", 1) は U を返します。
- **altova:char**("USA", 10) は空の文字列を返します。
- **altova:char**("USA", -2) は空の文字列を返します。

create-hash-from-string[altova:]

altova:create-hash-from-string(InputString *as xs:string*) **as xs:string** **XP2** **XQ1** **XP3.1** **XQ3.1**

altova:create-hash-from-string(InputString *as xs:string*, **HashAlgo** *as xs:string*) **as xs:string** **XP2** **XQ1** **XP3.1** **XQ3.1**

HashAlgo 引数により指定されているハッシュアルゴリズムを使用して InputString からハッシュ文字列を生成します。次のハッシュアルゴリズムは（大文字、または、小文字で指定されている可能性があります: **MD5**, **SHA-224**, **SHA-256**, **SHA-384**, **SHA-512**。（最初の署名を参照してください） 2番目の引数が指定されていない場合、**SHA-256** ハッシュアルゴリズムが使用されます。

サンプル

- **altova:create-hash-from-string**('abc') は **SHA-256** ハッシュアルゴリズムを使用して生成されたハッシュ文字列 を返します。
- **altova:create-hash-from-string**('abc', 'md5') は **MD5** ハッシュアルゴリズムを使用して生成されたハッシュ文字列 を返します。
- **altova:create-hash-from-string**('abc', 'MD5') は **MD5** ハッシュアルゴリズムを使用して生成された

ハッシュ文字列 を返します。

first-chars [altova:]

altova:first-chars(X-Number *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
xs:string に対するコンテキストアイテムの値を変換することにより得られた最初の X-Number 文字を含む文字列を返します。

サンプル

コンテキストアイテムが **1234ABCD** の場合:

- **altova:first-chars**(2) は 12 を返します。
- **altova:first-chars**(5) は 1234A を返します。
- **altova:first-chars**(9) は 1234ABCD を返します。

altova:first-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
InputString 引数として提出された文字列の最初の文字を含む文字列を返します。

サンプル

- **altova:first-chars**("2014-01-15", 5) は 2014- を返します。
- **altova:first-chars**("USA", 1) は U を返します。

format-string [altova:]

altova:format-string(InputString *as xs:string*, **FormatSequence** *as item()**) **as xs:string** **XP3.1** **XQ3.1**
入力文字列（最初の引数）には、配置パラメーター（%1, %2, etc）が含まれています。各パラメーターは、（2番目の引数として提出されている）フォーマットシーケンス内の対応するポジションでロケートされる文字列アイテムと置き換えられます。フォーマットシーケンス内の最初のアイテムは、配置パラメーター %1 を置換、二番目のアイテムは %2 を置き換えます。関数は、書式設定された代替を持つ文字列を返します。配置パラメーターに文字列が存在しない場合は、配置パラメーターが返されます。これは、配置パラメーターのインデックスが書式シーケンス内のアイテムの数よりも大きい場合発生します。

サンプル

- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe')) は "Hello Jane, John, Joe" を返します。
- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe', 'Tom')) は "Hello Jane, John, Joe" を返します。
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe', 'Tom')) は "Hello Jane, John, Tom" を返します。
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe')) は "Hello Jane, John, %4" を返します。

last-chars [altova:]

altova:last-chars(X-Number *as xs:integer*) を **xs:string** とする **XP3.1** **XQ3.1**
xs:string に対してのコンテキストアイテムの値の変換より取得された文字列の最後の X-Number 文字を含んでいる文字列を返します。

サンプル

コンテキストアイテムが **1234ABCD** の場合:

- **altova:last-chars**(2) は CD を返します。
- **altova:last-chars**(5) は 4ABCD を返します。
- **altova:last-chars**(9) は 1234ABCD を返します。

altova:last-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) **as xs:string** とする **XP3.1** **XQ3.1**
引数として提出された文字列の最後の X-Number 文字を含んでいる文字列を返します。

サンプル

- **altova:last-chars**("2014-01-15", 5) は 01-15 を返します。
- **altova:last-chars**("USA", 10) は USA を返します。

pad-string-left [altova:]

altova:pad-string-left(StringToPad *as xs:string*, **StringLength** *as xs:integer*, **PadCharacter** *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
PadCharacter 引数は1文字です。文字列の左側にパッドされ、この数が、StringLength 引数の整数の値と等しくなるように StringToPad の文字の数を増やします。StringLength 引数は任意の整数の値（正数または負数）を持つことができますが、パディングは、StringLength の値が StringToPad 内の文字数より多い場合の場合のみ発生します。もし、StringToPad が StringLength の値より多くの文字数も持つ場合、StringToPad は変更されません。

サンプル

- **altova:pad-string-left**('AP', 1, 'Z') は 'AP' を返します。
- **altova:pad-string-left**('AP', 2, 'Z') は 'AP' を返します。
- **altova:pad-string-left**('AP', 3, 'Z') は 'ZAP' を返します。
- **altova:pad-string-left**('AP', 4, 'Z') は 'ZZAP' を返します。
- **altova:pad-string-left**('AP', -3, 'Z') は 'AP' を返します。
- **altova:pad-string-left**('AP', 3, 'YZ') は[パッド文字が長すぎます] エラーを返します。

pad-string-right [altova:]

altova:pad-string-right(StringToPad *as xs:string*, **StringLength** *as xs:integer*, **PadCharacter** *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
PadCharacter 引数は1文字です。文字列の右側にパッドされ、この数が、StringLength 引数の整数の値と等しくなるように StringToPad の文字の数を増やします。StringLength 引数は任意の整数の値（正数または負数）を持つことができますが、パディングは、StringLength の値が StringToPad 内の文字数より多い場合の場合のみ発生します。もし、StringToPad が StringLength の値より多くの文字数も持つ場合、StringToPad は変更されません。

サンプル

- **altova:pad-string-right**('AP', 1, 'Z') を 'AP' を返します。
- **altova:pad-string-right**('AP', 2, 'Z') を 'AP' を返します。
- **altova:pad-string-right**('AP', 3, 'Z') を 'APZ' を返します。
- **altova:pad-string-right**('AP', 4, 'Z') を 'APZZ' を返します。
- **altova:pad-string-right**('AP', -3, 'Z') を 'AP' を返します。
- **altova:pad-string-right**('AP', 3, 'YZ') は[パッド文字が長すぎます] エラーを返します。

repeat-string [altova:]

altova:repeat-string(InputString *as xs:string*, **Repeats** *as xs:integer*) を **xs:string** とする

(C) 2015-2021 Altova GmbH

XP2 XQ1 XP3.1 XQ3.1
最初の InputString 引数により構成される文字列、Repeats 回繰り返してを生成します。

サンプル

- altova:repeat-string("Altova #", 3) は "Altova #Altova #Altova #" を返します。

substring-after-last [altova:]

altova:substring-after-last(MainString *as xs:string*, CheckString *as xs:string*) を xs:string とする XP3.1 XQ3.1
CheckString が MainString 内で検出された場合、MainString 内の CheckString が発生した後のサブ文字列が返されます。MainString 内で CheckString が検出されない場合、空の文字列が返されます。CheckString が空の文字列の場合、MainString 全体が返されます。一度以上発生する場合、CheckString の最後の発生の後の サブ文字列が返されます。

サンプル

- altova:substring-after-last('ABCDEFGH', 'B') は 'CDEFGH' を返します。
- altova:substring-after-last('ABCDEFGH', 'BC') は 'DEFGH' を返します。
- altova:substring-after-last('ABCDEFGH', 'BD') は '' を返します。
- altova:substring-after-last('ABCDEFGH', 'Z') は '' を返します。
- altova:substring-after-last('ABCDEFGH', '') は 'ABCDEFGH' を返します。
- altova:substring-after-last('ABCD-ABCD', 'B') は 'CD' を返します。
- altova:substring-after-last('ABCD-ABCD-ABCD', 'BCD') は '' を返します。

substring-before-last [altova:]

altova:substring-before-last(MainString *as xs:string*, CheckString *as xs:string*) を xs:string とする XP3.1 XQ3.1
CheckString が MainString 内で検出された場合、MainString 内の CheckString が発生する前のサブ文字列が返されます。MainString 内で CheckString が一度以上発生する場合、CheckString の最後の発生の前の サブ文字列が返されます。

サンプル

- altova:substring-before-last('ABCDEFGH', 'B') は 'A' を返します。
- altova:substring-before-last('ABCDEFGH', 'BC') は 'A' を返します。
- altova:substring-before-last('ABCDEFGH', 'BD') は '' を返します。
- altova:substring-before-last('ABCDEFGH', 'Z') は '' を返します。
- altova:substring-before-last('ABCDEFGH', '') は '' を返します。
- altova:substring-before-last('ABCD-ABCD', 'B') は 'ABCD-A' を返します。
- altova:substring-before-last('ABCD-ABCD-ABCD', 'ABCD') は 'ABCD-ABCD-' を返します。

substring-pos [altova:]

altova:substring-pos(StringToCheck *as xs:string*, StringToFind *as xs:string*) を xs:integer とする XP3.1 XQ3.1
StringToCheck 内での StringToFind の最初の発生の文字位置を整数として返します。StringToCheck の最初の文字は、位置 1 にあります。StringToFind が StringToCheck 内で発生しない場合、整数 0 が返されます。第 2 またはその後の StringToCheck, 発生を確認するには、この関数の次の署名を確認してください。

サンプル

- altova:substring-pos('Altova', 'to') は 3 を返します。

- **altova:substring-pos**('Altova', 'tov') は 3 を返します。
- **altova:substring-pos**('Altova', 'tv') は 0 を返します。
- **altova:substring-pos**('AltovaAltova', 'to') は 3 を返します。

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*, **Integer** *as xs:integer*) を **xs:integer** とする **XP3.1** **XQ3.1**
StringToCheck 内での StringToFind の最初の発生の文字位置を返します。Integer 引数により与えられた、文字位置から StringToFind の検索が開始されます。この位置の前の文字サブ文字列は検索されません。しかし、返された整数は、全体文字列 StringToCheck の検索された文字列の位置です。この署名は、StringToCheck 内で複数回発生する、第 2 または後の発生を検索する際に役に立ちます。 StringToFind が StringToCheck 内で発生しない場合、整数 0 が返されます。

サンプル

- **altova:substring-pos**('Altova', 'to', 1) は 3 を返します。
- **altova:substring-pos**('Altova', 'to', 3) は 3 を返します。
- **altova:substring-pos**('Altova', 'to', 4) は 0 を返します。
- **altova:substring-pos**('Altova-Altova', 'to', 0) は 3 を返します。
- **altova:substring-pos**('Altova-Altova', 'to', 4) は 10 を返します。

trim-string [altova:]

altova:trim-string(InputString *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
この関数は xs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

- **altova:trim-string**(" Hello World ")) は "Hello World" を返します。
- **altova:trim-string**("Hello World ")) は "Hello World" を返します。
- **altova:trim-string**(" Hello World")) は "Hello World" を返します。
- **altova:trim-string**("Hello World")) は "Hello World" を返します。
- **altova:trim-string**("Hello World")) は "Hello World" を返します。

trim-string-left [altova:]

altova:trim-string-left(InputString *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
この関数は xs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

- **altova:trim-string-left**(" Hello World ")) は "Hello World " を返します。
- **altova:trim-string-left**("Hello World ")) は "Hello World " を返します。
- **altova:trim-string-left**(" Hello World")) は "Hello World" を返します。
- **altova:trim-string-left**("Hello World")) は "Hello World" を返します。
- **altova:trim-string-left**("Hello World")) は "Hello World" を返します。

trim-string-right [altova:]

altova:trim-string-right(InputString *as xs:string*) を **xs:string** とする **XP3.1** **XQ3.1**
この関数は xs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされた xs:string を返します。

サンプル

(C) 2015-2021 Altova GmbH

- **altova:trim-string-right**(" Hello World ")) は" Hello World" を返します。
- **altova:trim-string-right**("Hello World ")) は"Hello World" を返します。
- **altova:trim-string-right**(" Hello World")) は" Hello World" を返します。
- **altova:trim-string-right**("Hello World")) は"Hello World" を返します。
- **altova:trim-string-right**("Hello World")) は"Hello World" を返します。

18.2.1.9 XPath/XQuery 関数: その他

XPath/XQuery 拡張関数の以下の一般的な目的は現在のStyleVision バージョンでサポートされており(i) XSLT コンテキスト内のXPath 式 または(ii) Xquery ドキュメント内のXQuery 式内で使用することができます。。

関数の名前指定と言語の適用性に関するメモ

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数はAltova **拡張関数名前空間**、**http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3.1**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3.1**

decode-string [altova:]

altova:decode-string(Input *as xs:base64Binary*) **as xs:string** **XP3.1** **XQ3.1**
altova:decode-string(Input *as xs:base64Binary*, **Encoding** *as xs:string*) **as xs:string** **XP3.1** **XQ3.1**
指定されたエンコードを使用して送信されたbase64Binary 入力を文字列にデコードします。エンコードが指定されていない場合UTF-8 エンコードが使用されます。以下のエンコードがサポートされます: US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4

サンプル

- **altova:decode-string**($XML1/MailData/Meta/b64B) はbase64Binary 入力をUTF-8 エンコード済み文字列として返します。
- **altova:decode-string**($XML1/MailData/Meta/b64B, "UTF-8") はbase64Binary 入力をUTF-8-エンコード済み文字列として返します。
- **altova:decode-string**($XML1/MailData/Meta/b64B, "ISO-8859-1") はbase64Binary 入力をISO-8859-1 エンコード済み文字列として返します。

encode-string [altova:]

altova:encode-string(InputString *as xs:string*) **as xs:base64Binaryinteger** **XP3.1** **XQ3.1**
altova:encode-string(InputString *as xs:string*, **Encoding** *as xs:string*) **as**

xs:base64Binaryinteger XP3.1 XQ3.1
与えらえている場合指定されているエンコードを使用して送信された文字列をエンコードします。エンコードが与えられていない場合 UTF-8 エンコードが使用されます。エンコードされた文字列はbase64Binary 文字に変換され、エンコードされたbase64Binary 値が返されます。最初にUTF-8 エンコードがサポートされ、サポートは以下のエンコードに拡張されます: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

サンプル

- **altova:encode-string**("Altova") はUTF-8 エンコード済み文字列 "Altova" の等価である base64Binary を返します。
- **altova:encode-string**("Altova", "UTF-8") はUTF-8 エンコード済み文字列 "Altova" の等価であるbase64Binary のを返します。

get-temp-folder [altova:]

altova:get-temp-folder() を xs:string とする XP2 XQ1 XP3.1 XQ3.1
この関数は引数を必要としません。この関数は現在のユーザーの一時的なフォルダーへのパスを返します。

サンプル

- **altova:get-temp-folder**() は、マシーン上で、xs:string として C:\Users\<UserName>\AppData\Local\Temp\ と類似したパスを返します。

generate-guid [altova:]

altova:generate-guid() asxs:string XP2 XQ1 XP3.1 XQ3.1
ユニークな文字列 GUID 文字列を生成します。

サンプル

- **altova:generate-guid**() は(例えば) 85F971DA-17F3-4E4E-994E-99137873ACCD を返します。

high-res-timer [altova:]

altova:high-res-timer() asxs:double XP3.1 XQ3.1
秒数でシステムハイれぞリューションタイマーの値を返します。ハイれぞリューションタイマーがシステム内で存在すると、必要とされる場合、高度に正確な時間の計算を有効化します(例えば、アニメーションと正確なコード実行の時間の決定などが例として挙げられます)。この関数は、システムのタイマーに精度を与えます。

サンプル

- **altova:high-res-timer**() は '1.16766146154566E6' などを返します。

parse-html [altova:]

altova:parse-html(HTMLText *as xs:string*) **asnode()** XP3.1 XQ3.1
HTMLText 引数は、HTML ドキュメントのテキストを含む文字列です。関数は、文字列からHTML ツリーを作成します。提供された文字列は、HTML 要素を含む、または、含まない場合があります。いずれの場合でも、ツリーのルート要素は、**HTML** と名付けられます。提出された文字列内のHTML コードが有効な HTMLであることを確認することが奨励されます。.

サンプル

- **altova:parse-html**("<html><head/><body><h1>Header</h1></body></html>") は提供された文字列からHTML ツリーを作成します。

▼ sleep[altova:]

altova:sleep(Millisecs *as xs:integer*) **asempty-sequence()** **XP2** **XQ1** **XP3.1** **XQ3.1**
Millisecs により与えられるミリ秒で示される期間のための実行を延期します。

⊟ *サンプル*

- **altova:sleep**(1000) は1000 ミリ秒のための実行を延期します。

[[トップ]]

18.2.1.10 バーコード関数

Altova XSLT エンジンはサードパーティーJava ライブラリを使用してバーコードを作成します。使用されるクラスやpublic メソッドを以下に記します。クラスは *<ProgramFilesFolder>*\Altova\Common2021\jar フォルダーにある AltovaBarcodeExtension.jar という パッケージに収められています。

使用される Java ライブラリは*<ProgramFilesFolder>*¥Altova¥Common2021¥jar フォルダーのサブフォルダー以下に収められています：

- barcode4j\barcode4j.jar（ウェブサイト: http://barcode4j.sourceforge.net/)
- zxing\core.jar（ウェブサイト: http://code.google.com/p/zxing/)

それぞれのフォルダーにはライセンスファイルも収められています。

Java 仮想マシン

バーコード関数を使用するには、Java 仮想マシンは使用中のマシンで使用することができます。マシンへのパスは以下で説明される通りです。

- Altova デスクトップ製品を使用している場合、Altova アプリケーションは、Java 仮想マシンへのパスを以下の順序で読み取ることにより自動的に検知しようとします：(i) Windows レジストリ(ii) **JAVA_HOME** 環境変数。アプリケーションのオプションダイアログ内にカスタムパスを追加することもできます。このエントリは自動的に検知された他のJava VM パス以上の優先順位を有します。
- Altova サーバー製品がWindows マシン上で作動している場合、Java 仮想マシンへのパスはWindows レジストリから最初に読み取られます。成功しない場合、**JAVA_HOME** 環境変数が使用されます。
- Altova サーバー製品を作動する場合、および、Linux またはmacOS マシン上でサーバー製品を作動する場合、Java 仮想マシンへのパスが**JAVA_HOME** 環境変数内に保管されていることを確認してください。**\bin\server** または **\bin\client** ディレクトリ内の **jvm.dll** ファイルを指している必要があります。

com.altova.extensions.barcode パッケージ

殆どの種類のバーコード生成には com.altova.extensions.barcode パッケージが使用されます。

以下のクラスが使用されます:

```
public class BarcodeWrapper
      static BarcodeWrapper newInstance( String name, String msg, int dpi, int
orientation, BarcodePropertyWrapper[] arrProperties )
```

```
double getHeightPlusQuiet()
double getWidthPlusQuiet()
org.w3c.dom.Document generateBarcodeSVG()
byte[] generateBarcodePNG()
String generateBarcodePngAsHexString()
```

public class **BarcodePropertyWrapper** *動的に使用されるバーコードプロパティを保管するために使用されます*

```
BarcodePropertyWrapper( String methodName, String propertyValue )
BarcodePropertyWrapper( String methodName, Integer propertyValue )
BarcodePropertyWrapper( String methodName, Double propertyValue )
BarcodePropertyWrapper( String methodName, Boolean propertyValue )
BarcodePropertyWrapper( String methodName, Character propertyValue )
String getMethodName()
Object getPropertyValue()
```

public class **AltovaBarcodeClassResolver** *により、*
org.krysalis.barcode4j.DefaultBarcodeClassResolver *にて登録されたクラスに加え、* qrcode *に対して*
com.altova.extensions.barcode.proxy.zxing.QRCodeBean *クラスが登録されます。*

com.altova.extensions.barcode.proxy.zxing パッケージ

QRCode バーコードの生成には com.altova.extensions.barcode.proxy.zxing パッケージが使用されます。

以下のクラスが使用されます:

class **QRCodeBean**

- org.krysalis.barcode4j.impl.AbstractBarcodeBean を拡張します
- com.google.zxing.qrcode.encoder に対して AbstractBarcodeBean インターフェースを作成します。

```
void generateBarcode(CanvasProvider canvasImp, String msg)
void setQRErrorCorrectionLevel(QRCodeErrorCorrectionLevel level)
BarcodeDimension calcDimensions(String msg)
double getVerticalQuietZone()
double getBarWidth()
```

class QRCodeErrorCorrectionLevel QRCode に対するエラー訂正レベル

```
static QRCodeErrorCorrectionLevel byName(String name)
"L" = ~7% correction
"M" = ~15% correction
"H" = ~25% correction
"Q" = ~30% correction
```

XSLT の例

以下に XSLT スタイルシートにてバーコード関数を使用する XSLT の例を示します

```
<?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:altova="http://www.altova.com"
   xmlns:altovaext="”http://www.altova.com/xslt-extensions”
   xmlns:altovaext-barcode="java:com.altova.extensions.barcode.BarcodeWrapper"
   xmlns:altovaext-barcode-
property="java:com.altova.extensions.barcode.BarcodePropertyWrapper">
```

```
   <xsl:output method="html" encoding="UTF-8" indent="yes"/>
   <xsl:template match="/">
      <html>
         <head><title/></head>
         <body>
            <img alt="barcode" src="{altovaext:get-temp-folder()}barcode.png"/>
         </body>
      </html>
      <xsl:result-document
         href="{altovaext:get-temp-folder()}barcode.png"
         method="text" encoding="base64tobinary" >
         <xsl:variable name="barcodeObject"
            select="altovaext-barcode:newInstance(&apos;Code39&apos;,string(&apos;some
value&apos;),
            96,0, (altovaext-barcode-property:new( &apos;setModuleWidth&apos;, 25.4 div
96 * 2 ) ) )"/>
         <xsl:value-of select="xs:base64Binary(xs:hexBinary(string(altovaext-
barcode:generateBarcodePngAsHexString($barcodeObject)) ))"/>
      </xsl:result-document>
   </xsl:template>
</xsl:stylesheet>
```

18.2.2 その他の拡張関数

Java や C# などのプログラミング言語には、XPath 2.0 / XQuery 関数、または XSLT 2.0 関数として利用できない関数がいくつかあります。そのような関数の良い例として、Java で利用することのできる sin() や cos() といった数学関数があります。XSLT スタイルシートや XQuery のクエリにてこれらの関数が利用できるのであれば、スタイルシートやクエリの適用範囲を大幅に拡張することができ、スタイルシート作成タスクの負担が大幅に軽減されます。Altova 製品で使用されている Altova エンジン(XSLT 1.0、XSLT 2.0、XQuery 1.0)では、Java や .NET および MSXSL scripts for XSLT における拡張関数の使用がサポートされます。 このセクションでは、拡張機能および XSLT スタイルシート内で MSXSL スクリプト。を使用する方法について記述します。使用できる拡張関数は以下のように構成されます:

- Java 拡張関数
- .NET 拡張関数
- XSLT に対する MSXSL スクリプト

記述の中では特に、(i) 関連するライブラリ内の関数がどのように呼ばれるか、(ii) 関数呼び出しを行う際に入力として使用される引数を変換するのにどのようなルールが適用され、return により値が返される際にどのような変換ルールが適用されるのか(XSLT/XQuery データオブジェクトに対する関数の結果)について説明されます。

必要条件

拡張関数のサポートを有効にするには、XSLT 変換や XQuery の実行を行うコンピューターに Java Runtime Environment (Java 関数にアクセスする場合)ならびに .NET Framework 2.0 以上(.NET 関数にアクセスする場合)がインストールされている、またはアクセスできる環境が整っている必要があります。

18.2.2.1 Java 拡張関数

Java 拡張関数は、XPath または XQuery 条件式にて使用することができるほか、Java のコンストラクターを呼び出したり、Java の(静的またはインスタンス)メソッドを呼び出すことができます。

Java クラスのフィールドは、引数を持たないメソッドとして扱われます。フィールドは静的またはインスタンスとして存在することができます。フィールドへのアクセス方法については、静的とインスタンスの両方について、以下のサブセクションにて記述されます。

このセクションは以下のサブセクションにより構成されます:

- Java : コンストラクター
- Java : 静的メソッドと静的フィールド
- Java : インスタンスメソッドとインスタンスフィールド
- データ型 : XPath/XQuery から Java へ
- データ型 : Java から XPath/XQuery へ

以下の点に注意してください

- Altova デスクトップ製品を使用している場合、Altova アプリケーションは、Java 仮想マシンへのパスを(以下の順序で)読み取ることにより自動的に検知しようとします : (i) Windows レジストリ(ii) **JAVA_HOME** 環境変数。アプリケーションのオプションダイアログ内にカスタムパスを追加することもできます。このエントリは自動的に検知された他の Java VM パス以上の優先順位を有します。
- Altova サーバー製品が Windows マシン上で作動している場合、Java 仮想マシンへのパスは Windows レジストリから最初に読み取られます。成功しない場合、**JAVA_HOME** 環境変数が使用されます。
- Altova サーバー製品を作動する場合、および、Linux または macOS マシン上でサーバー製品を作動する場合、Java 仮想マシンへのパスが **JAVA_HOME** 環境変数内に保管されていることを確認してください。**\bin\server** または **\bin\client** ディレクトリ内の **jvm.dll** ファイルを指している必要があります。

拡張関数のフォーム

XPath/XQuery 条件式における拡張関数では、`prefix:fname()` の形式を取る必要があります。

- `prefix:` 部により拡張関数が Java 関数として認識されます。`java:` から始まる URI のスコープ内の名前空間宣言に拡張関数を関連付けることで Java 関数であるという認識が行われます。名前空間の宣言により、例えば `xmlns:myns="java:java.lang.Math"` という Java クラスが特定されます。名前空間の宣言は、(コロン無しの) `xmlns:myns="java"` という形式で、Java クラスの識別子を拡張関数にある `fname()` 部の左型に配置することでも行うことができます。
- `fname()` 部により、呼び出されている Java メソッドが識別され、メソッドの引数が提供されます(以下の例を参照ください)。`prefix:` 部にて識別された名前空間 URI が Java クラスを識別できなかった場合、Java クラスの識別はクラスの前にくる `fname()` 部にて行うことになり、ピリオドによりクラスから分離されることになります(以下にある2番目の XSLT サンプルを参照)。

メモ 呼び出されるクラスはコンピューターのクラスパス上にある必要がありあす。

XSLT サンプル

以下に静的メソッドを呼び出す2つのサンプルを示します。最初のサンプルでは、クラス名 (`java.lang.Math`) が名前空間 URI に加えられており、`fname()` へ加えることはできません。2 番目のサンプルでは、`prefix:` 部に `java:` が与えられており、`fname()` 部にてクラスとメソッドが識別されます。

```
<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)" />
```


(C) 2015-2021 Altova GmbH

拡張関数内にあるメソッド名（上の例では cos() ）は、名前付き Java クラス（上の例では java.lang.Math ）の public な静的メソッドの名前に一致する必要があります。

XQuery サンプル

以下に XSLT のサンプルに似た XQuery のサンプルを示します:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

ユーザー定義された Java クラス

独自の Java クラスやメソッドを作成した場合、(i) JAR ファイル（または class ファイル）を介してこれらクラスファイルへアクセスしているか、(ii) これら（JAR または class）ファイルが、カレントディレクトリ（XSLT や XQuery ドキュメントが存在するディレクトリ）に配置されているかにより、これらクラスの呼び出し方法が変わってきます。これらファイルの特定方法については、ユーザー定義クラスファイルならびにユーザー定義 JAR ファイルを参照ください。カレントディレクトリには無いクラスファイルや JAR ファイルへのパスは指定しなければならないことご注意してください。

メモ StyleVision を使って SPS から生成された XSLT スタイルシートへ名前空間を加えるには、SPS がベースにしている XML スキーマのトップレベルにある schema 要素へ名前空間を追加する必要があります。StyleVision にて作成された全ての SPS では、デフォルトで xmlns:java="java" という名前空間宣言が自動的に作成されることご注意してください。

18.2.2.1.1 ユーザー定義のクラスファイル

アクセスがクラスファイルを介したものである場合、4 つのケースが考えられます:

- クラスファイルはパッケージである。XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている。（下のサンプルを参照）
- クラスファイルはパッケージでは無い。XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている。（下のサンプルを参照）
- クラスファイルはパッケージである。XSLT または XQuery ファイルがランダムな場所に収められている。（下のサンプルを参照）
- クラスファイルはパッケージである。XSLT または XQuery ファイルがランダムな場所に収められている。（下のサンプルを参照）

クラスファイルがパッケージではなく、XSLT または XQuery ドキュメントと同じ場所に収められているケースを考えてみましょう。この場合、フォルダー内の全クラスを発見することができるため、ファイルの場所を指定する必要はありません。クラスの識別を行う構文は以下のようになります:

```
java:classname
```

ここで、

java: によりユーザー定義の Java 関数が呼ばれていることが示されます（デフォルトでカレントディレクトリにある Java クラスがロードされます）。

classname は目的となるメソッドのクラスが含まれているクラスの名前です。

クラスが名前空間 URI にて識別され、名前空間がメソッド呼び出しにて使用されます。

クラスファイルがパッケージで、XSLT/XQuery ファイルが Java パッケージと同じ場所に収められている

以下の例では、com.altova.extfunc パッケージにある Car クラスの getVehicleType() メソッドが呼び出されています。com.altova.extfunc パッケージは JavaProject という名前のフォルダーに置かれており、XSLT ファイルも同じフォルダーに配置されています。

```
<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>
```

クラスファイルが参照され、XSLT/XQuery ファイルがクラスファイルと同じフォルダーに収められている

下のサンプルでは Car クラスの getVehicleType() メソッドが呼び出されます。以下を述べることができます: (i) Car クラスは次のフォルダー内にあります: JavaProject/com/altova/extfunc 。(ii) このフォルダーは下のサンプルの現在のフォルダー内に存在します。XSLT ファイルもフォルダー JavaProject/com/altova/extfunc 内にあります。

```
<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>
```

クラスファイルがパッケージされXSLT/XQuery ファイルがランダムな場所に収められている

以下の例では、com.altova.extfunc パッケージにある Car クラスの getVehicleColor() メソッドが呼び出されています。com.altova.extfunc パッケージは JavaProject という名前のフォルダーに置かれており、XSLT ファイルが任意の場所に配置されています。この場合、以下のような構文でパッケージの場所をクエリ文字列として URI 内にて指定する必要があります:

```
java:classname[?path=uri-of-package]
```

ここで、

java: によりユーザー定義の Java 関数が呼ばれていることを表します。
uri-of-package は Java パッケージの URI です。

classname は目的のメソッドが含まれているクラス名です。

クラスは名前空間 URI により特定され、名前空間がメソッド呼び出しのプレフィックスで使用されます。以下の例ではカレントディレクトリ以外にあるクラスファイルへのアクセスを行うことができます。

```
<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>
```

クラスファイルがパッケージではなく、XSLT/XQuery ファイルがランダムな場所に収められている

以下の例では、com.altova.extfunc パッケージにある Car クラスの getCarColor() メソッドが呼び出されています。com.altova.extfunc パッケージは JavaProject という名前のフォルダーに置かれており、XSLT ファイルが任意の場所に配置されています。以下のような構文で、クラスファイルの場所をクエリ文字列として URI 内にて指定する必要があります:

```
java:classname[?path=uri-of-classfile]
```

ここで

java: によりユーザー定義の Java 関数が呼ばれていることを表します。
uri-of-classfile は Java パッケージの URI です。
classname は目的のメソッドが含まれているクラス名です。

クラスは名前空間 URI により特定され、名前空間はメソッド呼び出しのプレフィックスで使用されます。以下の例ではカレントディレクトリ以外にあるクラスファイルへのアクセスを行うことができます。

```
<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>
```

メモ パスが外部関数により与えられている場合、ClassLoader によりパスが追加されます。

18.2.2.1.2 ユーザー定義の JAR ファイル

JAR ファイル経由でアクセスが行われた場合、以下の構文により JAR ファイルの URI を指定する必要があります:

```
xmlns:classNS="java:classname?path=jar:uri-of-jarfile!/"
```

クラスの識別を行う名前空間 URI のプレフィックスを使用してメソッドが呼び出されます: `classNS:method()`

上の例に対する説明は以下のとおりです:

`java:` 関数が呼び出されていることを表します。
`classname` がユーザー定義されたクラスの名前になります。
`?` はクラス名とパスを分離するために使用されます。
`path=jar:` により、JAR ファイルへのパスが与えられていることを示します。
`uri-of-jarfile` は JAR ファイルの URI となります。
`!/` は、終了を表すデリミタとなります。
`classNS:method()` により、メソッドの呼び出しが行われます。

その他にも、メソッド名とともにクラス名を与えることができます。構文の例を以下に示します:

```
              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()
```

以下に JAR ファイルを使った Java 拡張関数を呼び出す XSLT サンプルを記します:

```
<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>
```

メモ: 拡張関数によりパスが与えられている場合、ClassLoader にパスが追加されます。

18.2.2.1.3 Java:コンストラクター

拡張関数を使用することで Java コンストラクターを呼び出すことができます。`new()` により全てのコンストラクターを呼び出すことができます。

(C) 2015-2021 Altova GmbH

Java コンストラクターの呼び出し結果を、黙示的にXPath/XQuery データ型へ変換できる場合、Java 拡張関数により XPath/XQuery データ型のシーケンスが返されます。Java コンストラクターの呼び出し結果がXPath/XQuery データ型へ変換できない場合、値を返すクラス名でラップしたJava オブジェクトがコンストラクターにより作成されます。例えば、 `java.util.Date` クラスに対するコンストラクターが呼び出された場合（`java.util.Date.new()`）、`java.util.Date` を持ったオブジェクトが返されます。返されたオブジェクトのレキシカルフォーマットは、XPath データ型のレキシカルフォーマットにマッチしない場合もあり、目的のXPath データ型に対するレキシカルフォーマットへ値の変換を行い、その後目的のXPath データ型へ変換を行う必要があります。

コンストラクターにより作成されたJava オブジェクトにより2つのことが行えます:

- 変数への割り当てを行うことができます:
```
<xsl:variable name="currentdate" select="date:new()"
xmlns:date="java:java.util.Date" />
```
- 拡張関数への受け渡しを行うことができます（*インスタンスメソッドならびにインスタンスフィールド*を参照ください）:
```
<xsl:value-of select="date:toString(date:new())"
xmlns:date="java:java.util.Date" />
```

18.2.2.1.4 Java：静的メソッドと静的フィールド

静的メソッドは、Java 名ならびにメソッドの引数により直接呼び出すことができます。`E` や `PI` といった定数の静的フィールド（引数を持たないメソッド）は、引数を指定することなくアクセスすることができます。

XSLT の例

静的メソッドならびにフィールドを呼び出す例を以下に示します:

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos( jMath:PI() )" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:E() * jMath:cos(3.14)" />
```

上の拡張関数は `prefix:fname()` という形式を使用していることに注意してください。3つの例にあるプレフィックスは全てjMath: となっており、このプレフィックスは、`java:java.lang.Math` という名前空間 URI に関連付けられています。名前空間 URI はjava: で開始しなければなりません。上の例では、クラス名（`java.lang.Math`）を含むように拡張されています。拡張関数のfname() 部は（`java.lang.Math` のような）public クラスにマッチする必要があり、その後 public な静的メソッドが引数とともに続くか（例: `cos(3.14)`）、public な静的フィールドが続きます（例: PI()）。

上の例では、クラス名が名前空間 URI に含まれています。クラス名が名前空間 URI に含まれていない場合、以下の例にあるように、拡張関数の `fname()` 部にて追加する必要があります:

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

XQuery の例

XQuery における似たようなサンプルを以下に示します:

```
<cosine xmlns:jMath="java:java.lang.Math">
```

```
        {jMath:cos(3.14)}
    </cosine>
```

18.2.2.1.5 Java：インスタンスメソッドとインスタンスフィールド

メソッド呼び出しの第一引数としてパースされる Java オブジェクトが、インスタンスメソッドには与えられています。このような Java オブジェクトは通常、拡張関数を使うことで作成される(例：コンストラクターの呼び出し)か、スタイルシートパラメーター/変数により作成されます。以下に XSLT サンプルを示します:

```
<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>
```

上の例では、ノード `enrollment/@type` の値が以下のように作成されます:

1. `java.util.Date` クラスに対するオブジェクトがコンストラクター(`date:new()` コンストラクター)とともに作成されます。
2. `jlang.Object.getClass` メソッドの引数として Java オブジェクトがパースされます。
3. `getClass` メソッドにより得られたオブジェクトが `jlang.Object.toString` メソッドの引数としてパースされます。

結果(`@type` の値)は `java.util.Date` を持った文字列となります。

インスタンスフィールドは、引数としてインスタンスフィールドへ渡される Java オブジェクトではないという点で、理論的にはインスタンスメソッドと異なります。パラメーターや変数がその代わりに引数として渡されますが、パラメーター/変数そのものに Java オブジェクトから返された値が含まれている場合もあります。例えば、`CurrentDate` パラメーターには `java.util.Date` クラスのコンストラクターから返された値が含まれます。この値は引数として、`date:toString` インスタンスメソッドへ渡され、`/enrollment/@date` の値として使用されます。

18.2.2.1.6 データ型：XPath/XQuery から Java へ

XPath/XQuery 条件式内部から Java 関数が呼び出された場合、複数ある同名の Java クラスのうち、どのクラスが呼び出されたのか決定するのに、関数へ渡される引数のデータ型が重要になります。

Java では、以下のルールが適用されます:

- 同名の Java メソッドが2つ以上あり、それぞれが違う数の引数を受け取る場合、呼び出しに使用されている引数の数に一番マッチするメソッドが選択されます。
- XPath/XQuery の文字列、数値、boolean データ型は、黙示的に対応する Java データ型へ変換されます(以下のリストを参照)。与えられた XPath/XQuery 型が2つ以上の Java 型へ変換できる場合(例: `xs:integer`)、選択されたメソッドにて宣言されている Java 型が使用されます。例えば、呼び出された Java メソッドが `fx(decimal)` で、与えられた XPath/XQuery データ型が `xs:integer` の場合、`xs:integer` が Java の `decimal` データ型へ変換されます。

以下のテーブルに、XPath/XQuery の文字列、数値、boolean 型から Java データ型への黙示的な変換リストを示します。

xs:string	java.lang.String
xs:boolean	boolean (プリミティブ型), java.lang.Boolean
xs:integer	int, long, short, byte, float, double, ならびに、java.lang.Integer のようなこれらのラッパークラス
xs:float	float (プリミティブ型), java.lang.Float, double (プリミティブ型)
xs:double	double (プリミティブ型), java.lang.Double
xs:decimal	float (プリミティブ型), java.lang.Float, double(プリミティブ型), java.lang.Double

上のリストにある XML スキーマデータ型(ならびに XPath や XQuery で使用されているデータ型)のサブタイプも、対応する祖先のサブタイプとして Java のデータ型へ変換されます。

場合によっては、与えられた情報から正しい Java メソッドを選択することができない場合もあります。例えば、以下のような場合を考えてみましょう:

- 与えられた引数が 10 という値を持った xs:untypedAtomic 型で、mymethod(float) メソッドへ渡されるのを意図している。
- しかし、そのクラスにはは別のデータ型を取る mymethod(double) というメソッドも存在する。
- メソッド名が同じで、与えられた型 (xs:untypedAtomic) も float と double の両方に変換することができるため、xs:untypedAtomic が float ではなく double に変換される可能性もある。
- 結果として、意図したメソッドは選択されず、予期しない動作結果がを招く可能性がある。この問題を回避するには、意図したメソッドを使用するユーザー定義のメソッドを別の名前で新たに作成する必要があります。

上のリストでカバーされていない型(例: xs:date)は変換されず、エラーとなります。しかし場合によっては、Java コンストラクターを使用して、目的の Java データ型を作成することが可能だということも留意してください。

18.2.2.1.7 データ型:Java から XPath/XQuery へ

Java メソッドにより値が返され、値のデータ型が文字列、数値、または boolean 型の場合、対応する XPath/XQuery 型への変換が行われます。例えば、Java の java.lang.Boolean や boolean データ型は xsd:boolean へ変換されます。

関数から返された一次元配列は、シーケンス(sequence)に展開されます。2 次元以上の配列は変換されることが無いため、ラップして使用するべきでしょう。

Java オブジェクトや文字列、数値、boolean 以外のデータ型がラップされて返された場合、最初に(例えば toString といった)Java メソッドを使用して Java オブジェクトを文字列へ変換することで、目的の XPath/XQuery 型への変換を行います。XPath/XQuery では、文字列を目的となる型のレキシカルフォーマットへ変換し、目的の型への変換を(例えば cast as 式を使用することで)行うことができます。

18.2.2.2 .NET 拡張関数

.NET プラットフォームにて作業を行なっている場合、.NET 言語(例えばC#)で記述された拡張関数を使用することができます。.NET 拡張関数はXPath やXQuery 条件式内部から使用することができ、.NET クラス内部にあるコンストラクターや(static または インスタンス変数)プロパティを呼び出すことができます。

`get_PropertyName()` 構文を使用することにより.NET クラスのプロパティを呼び出すことができます。

このセクションは、以下のサブセクションにより構成されています:

- .NET コンストラクター
- .NET:静的メソッドならびに静的フィールド
- .NET:インスタンスメソッドとインスタンスフィールド
- データ型:XPath/XQuery から.NET へ
- データ型:.NET からXPath/XQuery へ

拡張関数のフォーム

XPath/XQuery 条件式にある拡張関数は、`prefix:fname()` の形式を取る必要があります。

- `prefix:` 部は、呼び出されている.NET クラスを特定するURI となります。
- `fname()` 部により、.NET クラス内にあるコンストラクター、プロパティ、または(静的または インスタンス)メソッドが特定され、必要な場合は引数が与えられます。
- URI は`clitype:` で開始する必要があり、これにより関数が.NET 拡張関数であることが認識されます。
- 拡張関数の `prefix:fname()` 形式は、システムクラスならびにコードされたアセンブリとともに使用することもできます。しかしクラスをロードする必要がある場合、必要な情報が含まれるパラメーターが必要になります。

パラメーター

アセンブリをロードするには以下のパラメーターを使用してください:

`asm`	ロードするアセンブリの名前。
`ver`	バージョン番号(ピリオドにより分離された最大4桁の整数)。
`sn`	アセンブリ厳密名のキートークン(16新数の数値)。
`from`	ロードするアセンブリ(DLL) の場所を特定するURI。URI が相対パスの場合、XSLT やXQuery ドキュメントに対して相対的になります。このパラメーターが指定された場合、その他のパラメーターが無視されます。
`partialname`	アセンブリ名の一部。`Assembly.LoadWith.PartialName()` へ渡され、アセンブリへのロードが試みられます。 `partialname` が指定された場合、その他のパラメーターが無視されます。
`loc`	例えば`en-US` というロケール。デフォルトは`neutral` です。

アセンブリがDLL からロードされる場合、`from` パラメータが使用して、`sn` パラメーターは使用しないでください。アセンブリがグローバルアセンブリキャッシュ(GAC) からロードされる場合、`sn` パラメータを使用して `from` パラメーターは使用しないでください。

最初のパラメーターの前に疑問符 (?) を挿入し、パラメーター同士はセミコロンで分離する必要があります。パラメーター名へ値を受け渡すには、統合符号 (=) を使用します(以下の例を参照ください)。

名前空間宣言の例

XSLT において、システムクラス `System.Environment` を特定する名前空間宣言の例を以下に示します:

```
xmlns:myns="clitype:System.Environment"
```

XSLT において、ロードするクラスを `Trade.Forward.Scrip` として特定する名前空間宣言の例を以下に示します:

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

XQuery において、システムクラス `MyManagedDLL.testClass` を特定する名前空間宣言の例を以下に示します。2つのケースが考えられます:

1. アセンブリが GAC からロードされた場合:
```
declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
      ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
```
2. アセンブリが DLL からロードされた場合(完全参照と一部の参照):
```
      declare namespace cs="clitype:MyManagedDLL.testClass?from=file:///C:/Altova
      Projects/extFunctions/MyManagedDLL.dll;

declare namespace cs="clitype:MyManagedDLL.testClass?from=MyManagedDLL.dll;
```

XSLT の例

システムクラス `System.Math` 内の関数を呼び出すための完全な XSLT の例を以下に示します:

```
<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>
```

`math` 要素にある名前空間宣言により、`math:` プレフィックスと `clitype:System.Math` URI が関連付けられます。URI の最初にある `clitype:` により、それ以降の記述がシステムクラスまたはロードされたクラスを特定するものであることが示されます。XPath 条件式にある math: プレフィックスにより、拡張関数が URI (そしてクラス) `System.Math` に関連付けられます。拡張関数により、`System.Math` クラス内のメソッドが特定され、必要な場所に引数が与えられます。

XQuery の例

上の XSLT に対する例と同様の XQuery 例を以下に示します:

```
<math xmlns:math="clitype:System.Math">
   {math:Sqrt(9)}
```

```
</math>
```

上のXSLT と同様に、名前空間宣言により.NET クラス(この場合はシステムクラス) が特定されます。XQuery 式により呼び出されるメソッドが特定され、引数が与えられます。

18.2.2.2.1 .NET コンストラクター

拡張関数を使用することで、.NET コンストラクターを呼び出すことができます。`new()` により全てのコンストラクターを呼び出すことができます。クラス内に2つ以上のコンストラクターがある場合、与えられた引数の数が最もマッチするコンストラクターが選択されます。与えられた引数に対してマッチするコンストラクターが見つからない場合、"`No constructor found'`" エラーが返されます。

XPath/XQuery データ型を返すコンストラクター

.NET コンストラクター呼び出しの結果が XPath/XQuery データ型へ黙示的に変換することができる場合、.NET 拡張関数から XPath/XQuery データ型のシーケンスが返されます。

.NET オブジェクトを返すコンストラクター

.NET コンストラクター呼び出しの結果がXPath/XQuery データ型へ適切に変換できない場合、値を返すクラス名でラップした.NET オブジェクトがコンストラクターにより作成されます。例えば、`System.DateTime` クラスのコンストラクターが(`System.DateTime.new()` により) 呼ばれた場合 `System.DateTime` 型を持つオブジェクトが返されます。

返されたオブジェクトのレキシカルフォーマットは、目的のXPath データ型と違っている場合があります。その場合、返された値を: (i) 目的の XPath データ型のレキシカルフォーマットへ変換し、(ii) 目的のXPath データ型へキャストする必要があります。

コンストラクターにより作成された.NET オブジェクトに対して3つのことを行うことができます:

- 変数内で使用することができます:
```
<xsl:variable name="currentdate" select="date:new(2008, 4, 29)"
xmlns:date="clitype:System.DateTime" />
```
- 拡張関数へ渡すことができます(*インスタンスメソッドとインスタンスフィールド* を参照ください):
```
<xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
xmlns:date="clitype:System.DateTime" />
```
- 文字列、数値、またはboolean へ変換することができます:
```
<xsl:value-of select="xs:integer(date:get_Month(date:new(2008, 4, 29)))"
xmlns:date="clitype:System.DateTime" />
```

18.2.2.2.2 .NET : 静的メソッドと静的フィールド

メソッド名と引数を与えることで、静的メソッドを直接呼び出すことができます。呼び出しに使用される名前は、クラス内にある public static メソッドと完全に一致する必要があります。関数の呼び出しに使用されたメソッド名と引数の数にマッチするものがクラス内に複数ある場合、与えられた引数が評価され、最もマッチするものが選択されます。マッチする結果が得られない場合、エラーが返されます。

メモ: .NET クラス内にあるフィールドは、引数を持たないメソッドとしてみなされます。プロパティは `get_PropertyName()` 構文により呼び出されます。

例

1つの引数とともにメソッド (System.Math.Sin(arg)) を呼び出す XSLT サンプルを以下に示します:

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

(引数なしのメソッドとしてみなされる) フィールド (System.Double.MaxValue() を呼び出す XSLT サンプルを以下に示します:

```
<xsl:value-of select="double:MaxValue()" xmlns:double="clitype:System.Double"/>
```

(get_PropertyName() 構文を使って) プロパティ (System.String()) を呼び出す XSLT サンプルを以下に示します:

```
<xsl:value-of select="string:get_Length('my string')"
xmlns:string="clitype:System.String"/>
```

1つの引数とともにメソッド ((System.Math.Sin(arg)) を呼び出す XQuery サンプルを以下に示します:

```
<sin xmlns:math="clitype:System.Math">
   { math:Sin(30) }
</sin>
```

18.2.2.2.3 .NET : インスタンスメソッドとインスタンスフィールド

インスタンスメソッドには、メソッド呼び出しの第一引数として .NET オブジェクトが渡されます。通常この .NET オブジェクトは、拡張関数 (例えばコンストラクター呼び出し) またはスタイルシート パラメーター/変数により作成されます。以下に XSLT の例を示します:

```
<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>
```

上の例では、System.DateTime コンストラクター(new(2008, 4, 29)) が、System.DateTime 型の.NET オブジェクトの作成に使用されます。このオブジェクトは、最初にreleasedate 変数の値として、次にSystem.DateTime.ToString() メソッドの引数として作成されます。System.DateTime.ToString() インスタンスメソッドは、System.DateTime コンストラクターの(new(2008, 4, 29)) における引数として2度呼び出されます。これらインスタンスにおいて releasedate 変数が.NET オブジェクトを取得するのに使用されます。

インスタンスメソッドとインスタンスフィールド

インスタンスメソッドとインスタンスフィールドの違いは理論的なものです。インスタンスメソッドでは.NET オブジェクトが直接引数に渡され、インスタンスフィールドではパラメーターや変数が(.NET オブジェクトそのものを含めることはできるものの) 代わりに渡されます。例えば上の例では、releasedate 変数に.NET オブジェクトが含まれており、この変数が2番目のdate 要素コンストラクターにて ToString() の引数として渡されます。そのため、最初のdate 要素にある ToString() インスタンスがインスタンスメソッドであるのに対し、2番目はインスタンスフィールドとしてみなされます。両方のインスタンスで求められる結果は等価です。

18.2.2.2.4 データ型:XPath/XQuery から .NET へ

.NET 拡張関数がXPath/XQuery 条件式内部で使用された場合、複数ある.NET メソッドのうち、どれが呼び出されたのか決定するのに関数の引数に使用されるデータ型が重要になります。

.NET では、以下のルールが適用されます:

- クラス内に同名のメソッドが2つ以上ある場合、呼び出しに使用された引数の数がマッチするメソッドだけが、呼び出される関数の候補に狭められます。
- XPath/XQuery の文字列、数値、boolean データ型は黙示的に対応する.NET データ型へ変換されます(以下のリストを参照)。与えられたXPath/XQuery 型が2つ以上の.NET 型へ変換できる場合(例: xs:integer)、選択されたメソッドにて宣言されている.NET 型が使用されます。例えば、呼び出された.NET メソッドがfx(fouble) で、与えられたXPath/XQuery データ型がxs:integer の場合、xs:integer が.NET のdouble データ型へ変換されます。

以下のテーブルに、XPath/XQuery の文字列、数値、boolean 型から.NET データ型へ行われる黙示的な変換リストを示します。

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

上のリストにあるXML スキーマ型(ならびにXPath やXQuery で使用されているデータ型) のサブタイプも、対応する祖先のサブタイプとして.NET のデータ型へ変換されます。

場合によっては、与えられた情報から正しい.NET メソッドを選択することができない場合もあります。例えば、以下のような場合を考えてみましょう:

- 与えられた引数が1つあるxs:untypedAtomic 値で、mymethod(float) メソッドへ渡されるのを意図している。
- しかし、そのクラスには別のデータ型をとるmymethod(double) というメソッドも存在する。

- メソッド名が同じで、与えられた型（xs:untypedAtomic）も float と double の両方に変換することができるため、xs:untypedAtomic が float ではなく double に変換される可能性もある。
- 結果として、意図したメソッドは選択されず、予期しない動作結果がを招く可能性がある。この問題を回避するには、意図したメソッドを使用するユーザー定義のメソッドを別の名前で新たに作成する必要があります。

上のリストでカバーされていない型（例: xs:date）は変換されずエラーとなります。

18.2.2.2.5 データ型：.NET から XPath/XQuery へ

.NET メソッドにより値が返される際に値のデータ型が文字列、数値、または boolean 型の場合、対応する XPath/XQuery 型への変換が行われます。例えば、.NET の decimal データ型は xsd:decimal へ変換されます。

.NET オブジェクトや文字列、数値、boolean 以外のデータ型が返された場合、最初に（例えば System.DateTime.ToString() といった）.NET メソッドを使用して .NET オブジェクトを文字列へ変換します。XPath/XQuery では、文字列を目的となる型のレキシカルフォーマットへ変換し、目的の型への変換を（例えば cast as 式を使用することで）行うことができます。

18.2.2.3 XSLT に対する MSXSL スクリプト

<msxsl:script> 要素にはユーザー定義の関数や変数が含まれており、XSLT スタイルシート内の XPath 条件式内部から呼び出しを行うことができます。<msxsl:script> はトップレベル要素で、<xsl:stylesheet> または <xsl:transform> の子要素である必要があります。

<msxsl:script> 要素は urn:schemas-microsoft-com:xslt 名前空間内に存在する必要があります（以下を参照ください）。

スクリプト言語と名前空間

ブロック内で使用されるスクリプト言語は <msxsl:script> 要素の language 属性にて指定され、XPath 条件式における関数の呼び出しに対して使用される名前空間は implements-prefix 属性により特定されます（以下を参照）。

```
<msxsl:script language="scripting-language" implements-prefix="user-namespace-prefix">

   function-1 or variable-1
   ...
   function-n or variable-n

</msxsl:script>
```

<msxsl:script> 要素は Windows Scripting Runtime を使ったやりとりを行うため、お使いのコンピューターにインストールされた言語だけしか <msxsl:script> 要素では使用することができません。**MSXSL スクリプトを使用するには .NET Framework 2.0 以上のプラットフォームをインストールする必要があります。**結果として <msxsl:script> 言語から .NET スクリプト言語を使用することができます。

HTML の <script> 要素における language 属性と同じ値が language 属性では受理されます。language 属性が指定されていない場合、Microsoft JScript がデフォルトとして想定されます。

implements-prefix 属性には名前空間スコープ内で宣言されたプレフィックスが与えられます。通常この名前空間は関数ライブラリのために予約されたユーザーの名前空間となります。<msxsl:script>要素内で定義された全ての関数ならびに変数は、implements-prefix 属性にて指定されたプレフィックスで特定される名前空間に収められます。XPath 条件式内部から関数が呼ばれる場合、完全修飾関数名が同じ名前空間内に関数として定義されていなければなりません。

サンプル

<msxsl:script> 要素内で定義された関数を使用する XSLT スタイルシートの例を以下に示します:

```
<?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>
```

データ型

スクリプトブロックとのやりとりに使用される パラメーターの値は XPath データ型に限定されます。スクリプトブロック内にある関数にてやりとりされるデータや変数に、この制限はありません。

アセンブリ

msxsl:assembly 要素を使用することで、アセンブリをスクリプト内部へインポートすることができます。アセンブリは名前や URI により特定されます。アセンブリのインポートは、コンパイル時に行われます。以下に msxsl:assembly 要素の簡単な使用例を示します:

```
<msxsl:script>
       <msxsl:assembly name="myAssembly.assemblyName" />
       <msxsl:assembly href="pathToAssembly" />

       ...

</msxsl:script>
```


(C) 2015-2021 Altova GmbH

アセンブリ名は、以下のような完全な名前でも::

```
"system.Math, Version=3.1.4500.1 Culture=neutral PublicKeyToken=a46b3f648229c514"
```

`"myAssembly.Draw"` のような短い名前でも指定できます。

名前空間

msxsl:using 要素により名前空間の宣言を行うことができます。これにより、スクリプト内において名前空間無しでアセンブリクラスを使用することができ、タイピングの手間を軽減することができます。以下に `msxsl:using` 要素の簡単な使用例を示します:

```
<msxsl:script>
       <msxsl:using namespace="myAssemblyNS.NamespaceName" />

       ...

</msxsl:script>
```

`namespace` 属性の値は名前空間の名前となります。

18.3 DB により生成された XML スキーマ内のデータ型

XML スキーマがデータベース(DB) から生成される場合、そのDB 固有のデータ型は、XML スキーマデータ型に変換されます。DB データ型のXML スキーマ データ型へのマッピングは、下に挙げられるDB で使用されます。下のリストから使用するDB を選択してください。

- ADO
- MS Access
- MS SQL Server
- MySQL
- ODBC
- Oracle
- Sybase

18.3.1 ADO

XML スキーマがADO データベース(DB) から生成される場合、ADO DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

ADO データ型	XML スキーマ データ型
adGUID	xs:ID
adChar	xs:string
adWChar	xs:string
adVarChar	xs:string
adWVarChar	xs:string
adLongVarChar	xs:string
adWLongVarChar	xs:string
adVarWChar	xs:string
adBoolean	xs:boolean
adSingle	xs:float
adDouble	xs:double
adNumeric	xs:decimal
adCurrency	xs:decimal
adDBTimeStamp	xs:dateTime
adDate	xs:date
adBinary	xs:base64Binary
adVarBinary	xs:base64Binary
adLongVarBinary	xs:base64Binary
adInteger	xs:Integer

(C) 2015-2021 Altova GmbH

adUnsignedInt	xs:unsignedInt
adSmallInt	xs:short
adUnsignedSmallInt	xs:unsignedShort
adBigInt	xs:long
adUnsignedBigInt	xs:unsignedLong
adTinyInt	xs:byte
adUnsignedTinyInt	xs:unsignedByte

18.3.2 MS Access

XML スキーマがMS Access データベース(DB) から生成される場合、MS Access DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

MS Access データ型	XML スキーマ データ型
GUID	xs:ID
char	xs:string
varchar	xs:string
memo	xs:string
bit	xs:boolean
Number(single)	xs:float
Number(double)	xs:double
Decimal	xs:decimal
Currency	xs:decimal
Date/Time	xs:dateTime
Number(Long Integer)	xs:integer
Number(Integer)	xs:short
Number(Byte)	xs:byte
OLE Object	xs:base64Binary

18.3.3 MS SQL Server

XML スキーマがMS SQL Server データベース(DB) から生成される場合、MS SQL Server DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

MS SQL Server データ型	XML スキーマ データ型
uniqueidentifier	xs:ID

char	xs:string
nchar	xs:string
varchar	xs:string
nvarchar	xs:string
text	xs:string
ntext	xs:string
sysname	xs:string
bit	xs:boolean
real	xs:float
float	xs:double
decimal	xs:decimal
money	xs:decimal
smallmoney	xs:decimal
datetime	xs:dateTime
smalldatetime	xs:dateTime
binary	xs:base64Binary
varbinary	xs:base64Binary
image	xs:base64Binary
integer	xs:integer
smallint	xs:short
bigint	xs:long
tinyint	xs:byte

18.3.4 MySQL

XML スキーマがMySQL データベース(DB) から生成される場合、MySQL DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

MySQL データ型	XML スキーマ データ型
char	xs:string
varchar	xs:string
text	xs:string
tinytext	xs:string
mediumtext	xs:string
longtext	xs:string

(C) 2015-2021 Altova GmbH

tinyint(1)	xs:boolean
float	xs:float
double	xs:double
decimal	xs:decimal
datetime	xs:dateTime
blob	xs:base64Binary
tinyblob	xs:base64Binary
mediumblob	xs:base64Binary
longblob	xs:base64Binary
smallint	xs:short
bigint	xs:long
tinyint	xs:byte

18.3.5 ODBC

XML スキーマが ODBC データベース（DB）から生成される場合、ODBC DB データ型は下のテーブルの通り XML スキーマデータ型に変換されます。

ODBC データ型	XML スキーマ データ型
SQL_GUID	xs:ID
SQL_CHAR	xs:string
SQL_VARCHAR	xs:string
SQL_LONGVARCHAR	xs:string
SQL_BIT	xs:boolean
SQL_REAL	xs:float
SQL_DOUBLE	xs:double
SQL_DECIMAL	xs:decimal
SQL_TIMESTAMP	xs:dateTime
SQL_DATE	xs:date
SQL_BINARY	xs:base64Binary
SQL_VARBINARY	xs:base64Binary
SQL_LONGVARBINARY	xs:base64Binary
SQL_INTEGER	xs:integer
SQL_SMALLINT	xs:short
SQL_BIGINT	xs:long

SQL_TINYINT	xs:byte

18.3.6 Oracle

XML スキーマがOracle データベース(DB) から生成される場合、Oracle DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

Oracle データ型	XML スキーマ データ型
ROWID	xs:ID
CHAR	xs:string
NCHAR	xs:string
VARCHAR2	xs:string
NVARCHAR2	xs:string
CLOB	xs:string
NCLOB	xs:string
NUMBER (with check constraint applied)*	xs:boolean
NUMBER	xs:decimal
FLOAT	xs:double
DATE	xs:dateTime
INTERVAL YEAR TO MONTH	xs:gYearMonth
BLOB	xs:base64Binary

* データ型 NUMBER の列にチェック制約が適用されている場合、またチェック制約が0 または1 の値をチェックする場合、この列の NUMBER データ型は、 xs:boolean のXML スキーマデータ型に変換されます。このメカニズムは、生成されたXML スキーマ内にxs:boolean データ型を生成する際に役に立ちます。

18.3.7 Sybase

XML スキーマがSybase データベース(DB) から生成される場合、Sybase DB データ型は下のテーブルの通りXML スキーマデータ型に変換されます。

Sybase データ型	XML スキーマ データ型
char	xs:string
nchar	xs:string
varchar	xs:string
nvarchar	xs:string
text	xs:string

(C) 2015-2021 Altova GmbH

sysname-varchar(30)	xs:string
bit	xs:boolean
real	xs:float
float	xs:float
double	xs:double
decimal	xs:decimal
money	xs:decimal
smallmoney	xs:decimal
datetime	xs:dateTime
smalldatetime	xs:dateTime
timestamp	xs:dateTime
binary<=255	xs:base64Binary
varbinary<=255	xs:base64Binary
image	xs:base64Binary
integer	xs:integer
smallint	xs:short
tinyint	xs:byte

18.4 技術データ

このセクションは、ソフトウェアの技術面に関する役に立つ背景情報を含んでいます。以下のように整理されています:
OS とメモリ要件Altova XML バリデーターAltova XSLT とXQuery エンジンUnicode のサポートインターネットの使用

18.4.1 OS とメモリ要件

オペレーティングシステム

Altova ソフトウェア アプリケーションは、以下のプラットフォームでご使用いただけます:
プラットフォーム更新済みのW indows 7 SP1、W indows 8、W indows 10プラットフォーム更新済みのW indows Server 2008 R2 SP1 または以降 **メモリ**
ソフトウェアがC++ で書かれているため、Java Runtime Environment をダウンロードする必要はなく、Java ベースのアプリケーションに比べ、通常少ないメモリを必要とします。しかしながら、各ドキュメントは完全に解析するため、また、ビューと編集の速度を向上するためにメモリにダウンロードされます。メモリの要件は、ドキュメントのサイズを増やします。

メモリ要件は、制限のない「元に戻す」履歴により影響を受けます。大きなドキュメントの大きなセクションの切り取り、貼り付け操作を繰り返し行うと、使用できるメモリがすぐに消費されます。

18.4.2 Altova XML バリデーター

XML ドキュメントを開くと、アプリケーションは、内蔵のXML バリデーターを使用して、指定されている場合、スキーマに対して整形式をチェックし、ツリーとインフォセットを作成します。XML バリデーターは、ドキュメントを編集する際にインテリジェントな編集ヘルプを提供し、発生する検証エラーを表示するために使用されます。

内蔵のXML バリデーターは、W 3C のXML スキーマ1.0 と1.1 仕様の最終勧告を実装しています。. New developments recommended by the W 3C XML スキーマ作業グループにより勧告される新しい項目は、XML バリデーターに継続的に組み込まれるため、Altova 製品は最高水準の開発環境を届けることができます。

18.4.3 Altova XSLT と XQuery エンジン

Altova 製品 は、Altova XSLT 1.0、2.0、および3.0 エンジンとAltova XQuery 1.0 と3.1 エンジンを使用しています。各エンジンのためのドキュメントと実装に固有の振る舞いに関しては、製品で使用されるエンジンの各ドキュメントの付属書（エンジン情報）で確認することができます。

メモ Altova MapForce は、XSLT 1.0、2.0 およびXQuery 1.0 エンジンを使用したコードを生成します。

18.4.4 Unicode のサポート

Altova XML 製品は、Unicode を完全にサポートします。XML ドキュメントを編集するには、ドキュメント内で使用されているUnicode 文字をサポートするフォントが必要です。

フォントの多くは、Unicode 範囲全体の特定のサブセットを含む場合があり、このため、通常は対応する表記システムをターゲットとします。テキストの一部が、文字化けして表示された場合、理由としては、選択されたフォントが必要とする字形を含まない場合があげられます。です

から、特に、異なる言語、または、異なる言語システムのXMLドキュメントを編集する場合、範囲全体をカバーするフォントを使用することが役にたちます。典型的なUnicode フォントは、Windows PC のArial Unicode MS で確認することができます。

アプリケーションフォルダーの/Examples フォルダー内で、異なる言語システムで表記された次の文章を含むUnicodeUTF-8.html という XHTML ファイルを確認してください:
When the world wants to talk, it speaks Unicode Wenn die Welt miteinander spricht, spricht sie Unicode 世界的に話すなら、Unicode です。) XHTML ファイルを開くと、Unicode の可能性を確認することができ、使用中の PC の使用することのできるフォントによりサポートされている表記システムが表示されます。

18.4.5 インターネットの使用

Altova アプリケーションは、次の状況でインターネット 接続を開始します:

- 登録ダイアログ(「**ヘルプ | ソフトウェアのライセンス認証**」)内の「評価キーコードをリクエスト」をクリックした場合、登録ダイアログボックス内の3つのフィールドが通常のhttp (ポート 80)接続を使用し、サーバーに転送され、無料の評価キーが顧客に通常の SMTP 電子メールを使用して送り返されます。
- Altova 製品の一部では、インターネットからファイルを開くことができます(「**ファイル | 開く | URL に切り替える**」)。この場合、ドキュメントは、次のプロトコルメソッドと接続の1つを使用して取得されます: HTTP (通常、ポート 80)、FTP (通常、ポート 20/21)、HTTPS (通常、ポート 443)。HTTP サーバーをポート 8080 で作動することもできます(URL ダイアログ内で、サーバー名とコロンの後にポートを指定します)。
- XML スキーマ、または、DTD を参照するXMLドキュメント、と、URL により指定されているドキュメントを開くと、参照されているスキーマドキュメントは、HTTP 接続(ポート 80) またはURL により指定されている他のプロトコル(上のポイント 2 参照)により抽出されます。XML ファイルが検証されている場合、スキーマドキュメントも抽出されます(オプションダイアログのファイルタブ内の(「**ツール | オプション**」))。アプリケーションに命令している場合、ドキュメントが開かれると検証が自動的に行われる場合もあります。
- WSDL とSOAP を使用するAltova アプリケーションでは、Web サービスを使用する接続は、WSDLドキュメントにより定義されています。
- XMLSpy 内で、「**電子メールで送信**」コマンドを使用する場合、(「**ファイル | 電子メールで送信**」) 現在選択されている範囲、または、ファイルは、ユーザーのマシンにインストールされているMAPI コンプライアント電子メールプログラムにより送信されます。
- ソフトウェアの荒いセンス認証 とLiveUpdate の一部として、Altova ソフトウェア使用許諾書内で更に詳しい説明を確認することができます。

18.5 ライセンス情報

このセクションには以下の内容が含まれています:

- ソフトウェアの配布 に関する情報
- ソフトウェアのアクティベーションとライセンスの計測
- ソフトウェアの使用に関する使用許諾契約書

本製品を使用する前に、上記の情報をよくお読みください。ソフトウェアのインストール時に上記のすべての条件に同意したとみなされ、お客様は上記の条件に拘束されることを同意したとみなされます。

Altova ライセンスの内容を確認するには、Altova Web サイト の Altova 法的な情報のページ に移動してください。

18.5.1 電子的なソフトウェアの配布

この製品は電子的なソフトウェアの配布により利用することが可能で、この配布方法により、以下のユニークなメリットがあります:

- 購入を決定する前に、無料でソフトウェアを試用することができます。*(Note: Altova MobileTogether Designer に対してライセンスを無料で割り当てることができます)。*
- Once ソフトウェアの購入を決定した際には、Altova Web サイト にて注文を行います。すぐにライセンス登録された製品の使用を開始することができます。
- オンラインにて注文を行うと、常に最新のソフトウェアをご利用いただけます。
- 製品パッケージには包括的なヘルプシステムが画面上に表示されます。最新バージョンのユーザーマニュアルは https://www.altova.com/ja/ 上にあり、(i) HTML フォーマットによる閲覧、ならびに (ii) PDF フォーマットのダウンロードと印刷に対応しております。

30 日間の評価期間

この製品をダウンロードした後は、最大で30日の間無料で製品の評価を行うことができます。20日間を超えた頃から、製品がライセンス登録されていないことがソフトウェアにより表示されます。このメッセージはアプリケーションが起動されるたびに表示され、30日間を超えてプログラムを使用するには、キーコードを含むライセンスファイルから提供される製品のライセンスを購入します。ライセンスファイルを製品のソフトウェアアクティベーションダイアログにアップロードして、製品をアンロックします。

https://shop.altova.com/ でライセンスを購入することができます

組織内でソフトウェアの評価を行う

評価版のソフトウェアを組織内のネットワークにて配布したい場合、またはインターネットに接続されていないコンピューターにてソフトウェアを使用する場合、どのような状態でも改変されていないことを条件に、セットアッププログラムだけの配布を行うことが可能です。ソフトウェアインストーラーへアクセスした人は、例外なく30日間の評価ライセンスキーコードをリクエストして、試用期間が経過した後は、製品を使い続けるためにライセンスの購入を行う必要があります。

18.5.2 ソフトウェアのアクティベーションとライセンスの計測

Altova のソフトウェアアクティベーションの一部として、ソフトウェアにより内部ネットワークまたはインターネットへの接続を行い、インストール時、登録時、Altova により使用されるライセンスサーバーの更新やライセンスの正当性を検証することで、ソフトウェアの不正な使用を防ぎ、顧客サービスを向上するため、ライセンスに関する情報を送信することがあります。アクティベーションにより、オペレーティングシステムや IP アドレス、

(C) 2015-2021 Altova GmbH

日付/時刻、ソフトウェアのバージョン、コンピュータの名前などのライセンスに関する情報が、お使いのコンピューターとAltova ライセンスサーバー間にてやり取りされます。

お使いのAltova 製品にはライセンス計測モジュールが内蔵されており、エンドユーザー使用許諾契約書の意図しない違反を防ぎます。お使いの製品はシングルユーザーまたはマルチユーザーとしてインストールされており、ライセンス計測モジュールにより、ライセンスされている数を超えたユーザーが同時に製品を使用することが無いことが保証されます。

このライセンス計測技術により、ローカルエリア接続（LAN）において、別々のコンピューター間で動作しているアプリケーションインスタンス間の通信が行われます。

シングルライセンス

ライセンス計測プロセスの一部としてアプリケーションが起動すると、ソフトウェアにより短いデータグラムがブロードキャストにより送信され、同一のネットワークセグメントにある他のコンピューターにてプログラムが動作していないかのチェックが行われます。応答が無い場合は、アプリケーションの他インスタンスから送信される信号に応えるため、ポートが開かれます。

マルチユーザーライセンス

同一のLAN 内にて2つ以上のアプリケーションインスタンスが使用された場合、スタートアップ時に、これらインスタンス間において通信が行われます。これらのインスタンス間にてキーコードのやりとりが行われ、購入された数のライセンスを超えてインスタンスが起動しないように保証することができます。このようなライセンス計測システムはUNIX やデータベース開発ツールにて広く使用されているもので、Altova ユーザーはリーズナブルな価格にて同時使用マルチユーザーライセンスを購入することができます。

弊社はアプリケーションのデザインも行っており、少数の小さなネットワークパケットを送信することで、ネットワークに対する負荷を最小限に抑えております。Altova により使用される2799番 TCP/IP ポートはIANA により公式登録されており（詳細は（IANA W eb サイト (http://www.iana.org/) を参照ください）、弊社のライセンス計測モジュールも既にテストされたものです。

ファイヤーウォールを使用している場合、2799番ポートにて Altova 製品が動作しているコンピューター同士が通信しているのに気づかれるかも知れません。その他の手段によりライセンス使用許諾書の内容が守られることを保証できる限り、組織間の異なるグループにおいてこのようなトラフィックをブロックすることは勿論可能です。

証明書に関するメモ

Altova アプリケーションはHTTPS を介して Altova ライセンスサーバー（`link.altova.com`）に通信します。この通信のためにAltova は登録済みのSSL 証明書を使用します。（例えば、社内IT 部署または外部エージェンシーによりこの証明書が置き換えられている場合、使用中のAltova アプリケーションは接続が安全でないことを警告します。Altova アプリケーションを開始するために代替の証明書を使用することができますが、自己責任で行ってください。*安全ではない接続* の警告メッセージが表示されると、証明書の発行元を確認して（Altova 証明書の代替証明書の使用の継続または停止を決定することができる）社内IT チームと相談してください。

（例えば、クライアントマシンへの、または、クライアントマシンへの通信を監視するために）自身の証明書の使用が必要な場合 Altova の無料管理ソフトウェアである Altova LicenseServer を使用中のネットワークにインストールすることが奨励されます。このセットアップでは、Altova LicenseServer はAltova との通信のためにAltova 証明書の使用を許可しつつクライアントマシンが所属機関の証明書の使用を継続することができます。

18.5.3 エンドユーザー使用許諾契約書

- Altova エンドユーザー使用許諾契約書: http://www.altova.com/ja/legal/eula
- Altova プライバシーポリシー: http://www.altova.com/ja/privacy

インデックス

(

(テーブルの)セル,
左右に分割, 781
上下に分割, 781
(テーブルの)行,
HTML 出力内の展開/折りたたみ, 159
(テーブルの)列,
HTML 出力の非表示, 159

.

.docx (Enterprise Edition のみ), 20, 37
.NET,
StyleVision スタンドアロンとの違い, 648
StyleVision の違い, 648
StyleVision の統合, 646
.NET 拡張関数,
XPath 条件式内, 665, 676
XSLT と XQuery, 1114
インスタンスメソッド、インスタンスフィールド, 1117
コンストラクター, 1116
サポート、Authentic View 内, 665, 676
データ型変換、.NET から XPath/XQuery へ, 1119
データ型変換、XPath/XQuery から .NET へ, 1118
概要, 1114
静的メソッド、静的フィールド, 1116
.NET 内のXSLT と XQuery のための拡張機能,
.NET 拡張関数を参照する, 1114

A

ActiveX,
アプリケーションレベルの統合, 966
ドキュメントレベルの統合, 968
統合の必要条件, 963
ActiveX コントロール,
Visual Studio ツールボックスへの追加, 964
ADO,
データ接続インターフェイスとして, 473
接続のセットアップ, 478
ADO.NET,
接続のセットアップ, 484
Altova XML パーサー,
について, 1128
Altova ウェブサイト, 820
Altova エンジン,
Altova 製品内で, 1128
Altova グローバルリソース,
グローバルリソース, 401
グローバルリソース を参照する, 401
Altova ソフトウェアの注文, 817
Altova 拡張関数,
チャート関数 (チャート関数を参照), 1031
AltovaRowStatus,
DB-ベースの SPS 内, 570
AltovaXML,
と FOP, 641
API,
documentation, 845
overview, 846
Application,
ActiveDocument, 866
Application, 866
Documents, 866
Parent, 868
Quit, 869
ASP.NET アプリケーション, 459
ASPX web アプリケーション, 459
Authentic Browser, 24
Authentic Desktop, 24
Authentic View, 609
Altova 製品内の, 24
GUI の概要, 593
SPS テーブル, 608
StyleVision の新規の バージョンと同期, 109
XML DB 内の編集 データ, 791
XML テーブル, 609
XML テーブルアイコン, 613
XML テーブルの使用, 609
XML/テキストとして貼り付け, 601
ツールバーアイコン, 594
テーブル (SPS と XML), 608
テキストの書式設定, 594
テキストの範囲, 601
と SPS, 107
と 作業用 XML ファイル, 35

Authentic View, 609
と 標準産業 スキーマ, 107
ドキュメントの表示, 595
ドキュメント作成の過程, 107
のためのツールバー ボタ, 685
マークアップの表示, 594, 595
メインウィンドウ, 595
主要な機能の使用方法, 604
詳細, 35
入力ヘルパー, 598
編集の保存, 713
Authentic View 内に表示される記録,
設定, 570
Authentic XML, 592
Authentic XML データ コマンドの保存, 713
Authentic スクリプト, 627, 629
内のマクロ, 630
Authentic ツールバー, 786
Authentic ツールバーのためのカスタムボタン, 786
Authentic ノードプロパティ, 416
Authentic メニュー, 785
マークアップの表示, 594
動的なテーブルの編集, 594
AuthenticRange,
AppendRow, 884
Application, 885
CanPerformAction, 885
CanPerformActionWith, 886
Close, 886
CollapsToBegin, 886
CollapsToEnd, 887
Copy, 887
Cut, 887
Delete, 887
DeleteRow, 888
DuplicateRow, 888
ExpandTo, 889
FirstTextPosition, 889
FirstXMLData, 890
FirstXMLDataOffset, 891
GetElementAttributeNames, 892
GetElementAttributeValue, 893
GetElementHierarchy, 893
GetEntityNames, 893
Goto, 894
GotoNext, 894
GotoNextCursorPosition, 895
GotoPrevious, 895
GotoPreviousCursorPosition, 896
HasElementAttribute, 897
InsertEntity, 897
InsertRow, 898
IsCopyEnabled, 898
IsCutEnabled, 898
IsDeleteEnabled, 899
IsEmpty, 899
IsEqual, 899
IsFirstRow, 899
IsInDynamicTable, 900
IsLastRow, 900
IsPasteEnabled, 900
IsTextStateApplied, 901
LastTextPosition, 901
LastXMLData, 902
LastXMLDataOffset, 902
MoveBegin, 903
MoveEnd, 904
MoveRowDown, 904
MoveRowUp, 904
Parent, 905
Paste, 905
PerformAction, 905
Select, 906
SelectNext, 907
SelectPrevious, 907
SetElementAttributeValue, 908
SetFromRange, 909
Text, 910
AuthenticView, 924
Application, 916
AsXMLString, 917
DocumentBegin, 918
DocumentEnd, 919
Event, 919
Goto, 921
IsRedoEnabled, 921
IsUndoEnabled, 922
MarkupVisibility, 922
OnBeforeCopy, 911
OnBeforeCut, 911
OnBeforeDelete, 912
OnBeforeDrop, 912
OnBeforePaste, 912
OnMouseEvent, 914
OnSelectionChanged, 914
Parent, 922

AuthenticView, 924
Print, 922
Redo, 923
Selection, 923
Undo, 924
WholeDocument, 925
XMLDataRoot, 925

C

C#,
StyleVision の統合, 971
CALS/HTML テーブル, 161, 782
CDATA セクション, 127
Authentic View 内に挿入, 604
Class ID,
StyleVision 統合, 975
COM-API,
documentation, 845
CoreCatalog.xml, 122
CSS スタイル,
クイックスタートチュートリアル内, 78
スタイルも参照, 53
モジュール SPS 内, 222
CSS スタイルシート,
スタイルも参照, 351
に適用されたメディア, 351
外部スタイルシート, 351
外部の優先順序のインポート, 351
CSS ファイル,
デザイン 概要 サイドバー内の管理, 42
CustomCatalog.xml, 122

D

Database connection,
setup examples, 502
DB, 567, 616, 617
Authentic View 内で表示される記録, 416
Authentic View 内のクエリ, 616
Authentic View 内の編集, 616, 621
Authentic View内のテーブルのナビゲート, 616
DB クエリ内のパラメーター, 617
SPS (非-XML DB) のために作業用 XML データを選択, 557
SPS (XML DB)のために作業用 XML データを選択する, 564
SPS のために 作業用 XML データを選択する, 557
SPS のためにスキーマを選択(XML DB), 564
SPS のためにスキーマを選択(非-XML DB), 557
SPS のためにスキーマを選択する, 557
StyleVision 内での作業, 469
StyleVision 内の作業メカニズム, 471
XML データ ファイル, 567
XML ファイル と HTML/PDF 出力 の生成, 578
クエリの作成, 617
のためのスキーマ ファイル, 567
生成された XML スキーマ ファイル, 471
生成された XML データ ファイル, 471
内の Authentic Viewの表示のフィルター, 617
DB デフォルト, 572
DB クエリ ボタン,
SPS 内に挿入, 570
DB クエリ内のコメント, 586
DB クエリ内のブックマーク, 586
DB クエリ内の自動補完, 586
DB クエリ内の領域, 586
DB コントロール, 790
自動挿入, 570
挿入のためのコマンド, 748
DB スキーマ (Enterprise と Professional Edition), 211
DB テーブル移動 コントロール, 790
DB パラメーター,
作成と編集, 731
使用方法, 572
DB フィルター,
XML ファイルのためにデータをフィルターする, 572
クリア, 797
データ型内, 572
作成と変更, 572
編集, 797
DB-ベースの SPS ためのスキーマ, 557
DB-ベースの SPS のための XML ファイル, 564
DB-ベースの SPS のためのスキーマ, 564
Document,
Application, 927
Close, 928
FullName, 928
GetPathName, 928
Name, 929
Path, 930
Save, 930
SaveAs, 930

Document,
Saved, 930
SetPathName, 936
Documents,
Count, 937
Item, 937
DPI, 717
と 印刷 出力, 394
とピクセルにより定義された長さ, 394
DTD,
SPS ソースとして, 206
パースされていないエントリの宣言, 422
DTD とカタログ, 121

E

Eclipse のための StyleVision プラグイン,
インストール, 650
Eclipse プラットフォーム,
StyleVision パースペクティブ, 656
と StyleVision, 649
と StyleVision 統合パッケージ, 650
Eclipse 内のStyleVision パースペクティブ, 656
Enter キー, 626
Enter キー参照, 626
使用の効果, 626
Enumerations,
SPYAuthenticActions, 959
SPYAuthenticDocumentPosition, 959
SPYAuthenticElementKind, 959
SPYAuthenticMarkupVisibility, 960
SPYMouseEvent, 960
SPYXMLDataKind, 962
Event, 911, 912, 914
Excel テーブル コンテンツ,
デザインにコピーして張り付ける, 134

F

Firebird,
ODBC を使用しての接続, 504
FO プロセッサー (Enterprise Edition),
セットアップ, 27
FO 変換, 641

G

GUI,
ドキュメントビュー内, 34
メイン ウィンドウ の, 34
詳細, 33
複数の ドキュメント内, 34

H

HTML,
StyleVision の統合, 977
HTML インポート, 452
HTML テーブルの, 456
HTML リストの, 456
SPS からファイルの作成, 458
SPS デザイン, 454
スキーマ 構造, 454
新規の SPS の作成, 452
HTML から XML への変換, 452
HTML サンプル,
of StyleVisionControl 統合, 975
StyleVisionControl 統合, 975, 976, 977
HTML テーブル, 161, 782
HTML ページ コンテンツ,
デザインにコピーして張り付ける, 134
HTML 出力, 110
とイメージサポート, 173

I

IBM DB2,
JDBC を介した接続, 506
ODBC を使用しての接続, 508
IBM DB2 for i,
ODBC を使用しての接続, 516
IBM Informix,
connecting through JDBC, 515
JDBC を使用しての接続, 518
IE 9,
Internet Explorer 互換性を参照, 105
Internet Explorer 互換性, 105

J

Java, 979
Java と .NET 関数 (Enterprise Edition のみ),
自動計算内, 258
Java 拡張関数,
XPath 条件式, 665, 676
XPath 条件式 ダイアログの編集, 665, 676
XPath 条件式内, 665, 676
XSLT と XQuery, 1105
インスタンスメソッド、インスタンスフィールド, 1112
コンストラクター, 1110
サポート、Authentic View 内, 665, 676
データ型変換、 Java から XPath/XQuery へ, 1113
データ型変換、XPath/XQuery から Java へ, 1112
ユーザー定義の JAR ファイル, 1110
ユーザー定義のクラスファイル, 1107
概要, 1105
静的メソッド、静的フィールド, 1111
JavaScript,
スクリプトを参照, 446
JavaScript Editor, 447
JavaScript エディター, 446
デザインビュー内, 35
JDBC,
Teradata への接続, 550
データ接続インターフェイスとして, 473
接続の設定 (Windows), 494
JRE,
Eclipse のための StyleVision プラグイン, 650
JScript,
scripting with StyleVision, 823

L

Line,
レイアウト コンテナー内, 766

M

Macros,
developing, 823, 828
MariaDB,
ODBC を使用した接続, 520
Microsoft Access,
ADO を介しての接続, 478, 522
Microsoft Office 2007 (Enterprise Edition のみ), 20, 37
Microsoft SQL Server,
ADO を介しての接続, 525
ODBC を使用しての接続, 527
MobileTogether デザイン,
エクスポート, 713
MS Word ドキュメントコンテンツ,
デザインにコピーして張り付ける, 134
MSXSL スクリプト内の拡張関数, 1119
msxsl:script, 1119
MySQL,
ODBC を使用しての接続, 533

O

ODBC,
MariaDB への接続, 520
Teradata への接続, 552
データ接続インターフェイスとして, 473
接続のセットアップ, 491
ODBC ドライバー,
使用可能か確認する, 491
Office Open XML (Enterprise Edition のみ), 20, 37
OLE DB,
データ接続インターフェイスとして, 473
OOXML (Enterprise Edition のみ), 20, 37
Oracle データベース,
JDBC を使用しての接続, 535
ODBC を使用しての接続, 536
OS,
Altova 製品のための, 1128
Otherwise 条件 分岐, 268
Overview,
of XMLSpy API, 846

P

Parent, 929
PDF,
定義 ページ プロパティ, 383

PDF 出力,
ページ番号が付いたメディア出力を参照, 388
PDF 出力 (Enterprise Edition), 110
とイメージサポート, 173
PDF 内のページ プロパティ, 383
PostgreSQL,
ODBC を使用しての接続, 542
直接 (ネイティブ)接続, 496
直接接続する(ネイティブ), 498
Progress OpenEdge データベース,
JDBC を使用しての接続, 545
ODBC を使用しての接続, 546
PXF ファイル, 462
デプロイ, 467
作成, 462
編集, 466
名前をつけて保存, 710

R

RaptorXML, 637
と FOP, 641
RaptorXML 内のカタログ, 122
Return キー,
Enter キーを参照, 626
RootCatalog.xml, 122
RTF 出力,
ページ番号が付いたメディア出力を参照, 388
RTF 出力 (Enterprise Edition), 110
とイメージサポート, 173

S

Scripting Editor,
overview, 823, 825
SPS,
と Authentic View (Enterprise と Professional Edition), 25
と DB, 469
と StyleVision, 25
と XSLT スタイルシート, 25
開く, 700
概要, 25
再ロード, 700
閉じる, 700
SPS テーブル,
静的なテーブルも参照, 145
動的なテーブルの編集, 594
動的なテーブルも参照, 145
SPS テーブル 内の Authentic View,
使用方法, 608
SPS デザインの構造, 203
SPS デザイン概要, 103
SPS と Authentic View, 107
SPS のためのソース ファイル, 102
SPS ファイル 構造, 203
SPS 内の XPath バージョン, 104
SPS 内の XSLT バージョン, 104
SQL Server,
ADO.NET を使用して接続する, 484
SQL エディター,
クエリ内で作成, 589
データベース クエリ ウィンドウ内, 586
詳細, 586
SQL サーバー,
ADO を介しての接続, 478
SQLite,
外部キーの無効化, 500
接続のセットアップ (Windows), 500
StyleVision,
Authentic と同期, 109
ユーザー マニュアル, 17
製品の機能, 20
統合, 963
StyleVision Power Stylesheet,
SPS を参照, 17
StyleVision について, 820
StyleVision に関する良くある質問, 820
StyleVision のセットアップ, 27
StyleVision のためのサポート, 820
StyleVision のための技術サポート, 820
StyleVision の新規リリース,
StyleVision と同期, 109
StyleVision 統合,
のサンプル, 975, 976, 977
StyleVision 統合パッケージ, 647, 650
StyleVisionBatch, 27, 637
StyleVisionControl, 1004
C# を使用した統合, 971
HTML を使用下統合, 977
アプリケーションレベルでのサンプル 統合, 975, 976, 977
アプリケーションレベルのサンプル 統合, 976
オブジェクトレファレンス, 1001

StyleVisionControl, 1004
ドキュメント, 963
ドキュメントレベルでの統合のサンプルサンプル, 971
StyleVisionControl ドキュメント, 1012
StyleVisionControl プレースホルダ, 1019
StyleVisionコマンド,
StyleVisionControl, 1002
StyleVisionControl 内, 1004
Sybase,
JDBC を使用しての接続, 549

T

Teradata,
JDBC を使用した接続, 550, 552
terminate, 869
Toolbar buttons,
adding and removing, 681

U

Unicode のサポート,
Altova 製品内で, 1128
URI,
パースされていないエントリの保持, 422

V

VBScript,
scripting with StyleVision, 823
Visual Studio,
StyleVision ActiveX コントロールをツールボックスへ追加, 964
Visual Studio .Net,
と StyleVision, 646
VS .NET,
と StyleVision 統合パッケージ, 647
VS .NET のための StyleVision プラグイン,
インストール, 647

W

Windows,
Altova 製品のためのサポート, 1128
Word 2007 (Enterprise Edition のみ), 20, 37
Word ドキュメントコンテンツ,
デザインにコピーして張り付ける, 134
WordML (Enterprise Edition のみ), 20, 37

X

XML,
デザインに挿入, 143
XML DB,
新規 XML データ行のロード, 616
新規の データ行を Authentic View にロード, 791
XML スキーマ と DTD,
SPS ソースとして, 206
XML データ,
SPS デザイン内に挿入, 127
複数のソースからマージ, 215
XML データ DB-ベースの SPS ための, 557
XML テーブル (Enterprise と Professional Edition), 145
XML テーブル 内の Authentic View,
編集のためのアイコン, 613
XML テーブル内の Authentic View,
使用方法, 609
XML ドキュメントコンテンツ,
デザインビュー内のシンボル, 661
XML ドキュメントノード,
デザインビュー内のシンボル, 661
XML の検証,
Authentic View 内, 791
XML パーサー,
について, 1128
XML ファイル,
DB からの データ, 578
XMLData,
AppendChild, 947
EraseAllChildren, 948
EraseCurrentChild, 948
GetChild, 949
GetChildKind, 950

XMLData,
GetCurrentChild, 951
GetFirstChild, 951
GetNextChild, 952
HasChildren, 953
HasChildrenKind, 953
InsertChild, 954
IsSameNode, 955
Kind, 955
MayHaveChildren, 955
Name, 956
Parent, 956
TextValue, 956
XMLSpy, 24
XMLSpy API,
documentation, 845
overview, 846
XMLSpy コマンドテーブル, 988
XMLSpyLib, 845
Application, 865
AuthenticRange, 882
AuthenticView, 910
Document, 925
Documents, 936
XMLData, 945
XPath,
複数の ドキュメント内でのノードの検索, 294
XPath 1.0,
と 日付, 441
XPath 2.0,
と 日付, 441
XPath ダイアログ,
デバッグ式, 669
式のテスト, 666
詳細, 665, 676
XPath フィルター,
グローバル テンプレート上, 232
XPath 演算子,
XPath ダイアログ内, 665, 676
XPath 関数,
XPath ダイアログ内, 665, 676
ユーザー定義, 427
XPath 条件式 ダイアログの編集,
XPath ダイアロ を参照, 676
XPath ダイアログを参照, 665
XQuery,
拡張関数, 1105
XQuery プロセッサー,
Altova 製品内で, 1128
xs:date,
と 日付選択, 442
xs:dateTime,
と 日付選択, 442
XSLT,
デザインにコードフラグメントを挿入, 143
拡張関数, 1105
XSLT インポート, 424
XSLT から SPS は, 424
XSLT から新規の, 424
XSLT スタイルシート プレビュー,
出力ビュー内, 37
XSLT テンプレート, 48
SPS へインポートする, 246
デザイン 概要 サイドバー内の管理, 42
XSLT テンプレートのインポート,
SPS へ, 246
XSLT と XQuery のためのJava 拡張関数,
Java 拡張関数を参照する, 1105
XSLT と XQuery のための拡張関数, 1105
XSLT のパースされていないエントリの URI 関数, 422
XSLT バージョン,
SPS のための設定, 691
XSLT プロセッサー,
Altova 製品内で, 1128
XSLT 変換, 641
XSLT 要素,
デザインにコードとして挿入する, 142
XSLT/XQuery 内のスクリプト,
拡張関数で参照する, 1105

Z

アクティブおよび関連するファイルのプロジェクトへの追加, 738
アクティブな ファイルをプロジェクトに追加する, 738
アクティブな 構成, 807
イニシャル ドキュメント セクション, 380
イベントハンドラー,
関数を割り当てる, 448
イメージ,
SPS 内の使用の概要, 171
URI の指定, 171
イメージも参照, 171
クイックスタートチュートリアル内, 73
サポートされている型, 173

イメージ,
サンプル ファイル, 176
デザインビュー内のシンボル, 661
デザイン内に挿入のためのコマンド, 752
とパースされていないエンティティ URI, 171
パースされていないエントリを使用して検索, 422
出力の表示の評価, 171
イメージ書式設定,
内の Authentic View, 625
インターネットの使用,
Altova 製品内で, 1129
インターフェイス,
GUI を参照, 33
ウィンドウ メニュー, 816
エイリアス,
グローバルリソース参照, 401
エンコード,
出力ファイルのための, 813
エンコード コマンド, 717
エンティティ,
Authentic View 内に挿入, 604
Authentic View 内の定義, 604
内の Authentic View の定義, 624
エンティティ 参照,
と 出力 エスケープ, 338
エンティティ入力ヘルパー,
Authentic View, 598
エンドユーザー使用許諾契約書, 1130, 1131
エントリ,
Authentic View 内の定義, 685, 792
URI ホルダーとして使用, 422
パースされていない, 422
オブジェクトロケーター,
データベース クエリ ウィンドウ内, 583
カスタム化 StyleVision, 808
カスタム辞書,
SPS スペルチェックのための, 803
カタログ ファイル, 117
カタログと環境変数, 124
カタログのカスタム化, 123
カタログファイル, 121
カタログメカニズムの概要, 121
カタログ内で使用される環境変数, 122
カバーページ, 380
キーコード,
Altova ソフトウェアのための, 817
キープ, 392
キーボードショートカット,
コマンドのカスタム化, 808
クイックスタートチュートリアル,
CSS スタイル, 78
testing Authentic View (Enterprise と Professional Edition), 100
XSLT スタイルシートの生成, 100
イメージ, 73
クラス 属性, 78
グローバル テンプレート, 95
コンテンツ プレースホルダ, 66
コンボボックス, 88
その他のコンテンツ, 95
はじめに, 61
リスト, 88
作成 新規の SPS ドキュメント, 62
自動計算, 84
条件, 88
新規の SPS ドキュメントのセットアップ, 62
水平線, 73
静的コンテンツ, 73
静的テキスト, 73
定義済みフォーマット, 78
動的コンテンツ, 66
必要な ファイル, 61
クエリ,
Authentic View内の DB 表示, 617
データベース クエリを参照, 579
クエリ データベース,
データベース クエリを参照, 579
クエリ データベース コマンド, 579
クエリ ペイン,
データベース クエリ ウィンドウ内, 586
クエリ ボタン,
SPS 内に挿入, 570
クラス 属性,
クイックスタートチュートリアル内, 78
グループ化, 277
サンプルによりグループ化 (Scores.sps), 281
サンプルによりグループ化(Persons.sps), 279
グローバル スタイル,
スタイルを参照, 354
グローバル テンプレート, 231, 232
クイックスタートチュートリアル内, 95
その他のコンテンツへの効果, 132
グローバル リソース,
定義, 401
グローバル リソース XML ファイル, 401
グローバルリソース,

グローバルリソース,
ダイアログ, 806
ツールバー, 691
ツールバーを使用して構成を選択する, 691
データベースの型の定義, 409
ファイルの型の定義, 403
フォルダーの型の定義, 407
構成の変更, 414
使用, 410, 411, 414
グローバルリソースの,
構成, 401
グローバルリソースの構成, 414
グローバル型,
テンプレート内, 232
コピー コマンド, 726
コマンド,
カスタム化, 808
コマンド ラインのユーティリティ, 27
コマンドライン, 637
と パラメーター, 290
と 計画されたタスク, 645
コンテキストノード,
XPath ダイアログ内, 665, 676
コンテンツ,
デザイン内に挿入のためのコマンド, 744
コンテンツ プレースホルダ,
クイックスタートチュートリアル内, 66
コンテンツとしてノードを挿入, 127
コンテンツ モデル,
デザイン上の曖昧さの効果, 130
コンテンツ 編集 プロシージャ, 126
コントロールから,
挿入のためのメニューコマンド, 747
コンポーネントダウンロードセンター,
Altova ウェブサイトで, 820
コンボボックス, 181
クイックスタートチュートリアル内, 88
サブトータル,
ヘッダー と フッター内, 390
サポートオプション, 31
ショートカット,
キーボードのためのカスタム化, 808
シンプルな グローバル テンプレート, 232
スキーマ,
SPS ソースとして, 206
SPS のための DB から, 211
カタログを介したルックアップ, 123
ユーザー定義, 212
スキーマ 構造,
と SPS デザイン, 130
スキーマソース, 102, 694
SPS のための選択, 205
サイドバー ウィンドウ, 44
ソースの変更, 294
デザイン 概要 サイドバー内の管理, 42
と ルート 要素 (ドキュメント 要素), 205
の概要, 48
複数のSPS (Enterprise Edition), 205
複数のソース と XPath, 294
複数のソース とノードの検索, 294
スキーマソース ウィンドウ,
デザイン入力ヘルパーも参照, 39
スキーマツリーオプション, 813
スキーマとカタログ, 121
スキーマの追加, 694
スクリーンの解像度,
とピクセルにより定義された長さ, 394
スクリーン上の ヘルプ,
のインデックス, 817
検索, 817
目次, 817
スクリプト,
Authentic View 内, 627, 629, 630
JavaScript 関数, 448
SPS 内の使用, 446
デザイン ツリー内, 446
と JavaScript 関数, 446
の概要, 48
定義 JavaScript 関数, 447
スクリプト Editor, 629
スクロール ボタン,
メイン ウィンドウ内, 34
スタイル,
CSS ルールを合成, 364
SPS に CSS スタイルシートを割り当てる, 351
SPS 内の定義 グローバル スタイル, 354
StyleVision 内で作業, 350
XML データから, 362
XPath 条件式, 362
XPath 条件式を使用して, 362
カスケードの順序, 350
サイドバー ウィンドウ, 53
スタイルの優先順位, 354
デザイン入力ヘルパーも参照, 39
と スタイル, 362
とプロパティグループ, 53

スタイル,
の優先順位, 51
の用語, 350
割り当てられた外部スタイルシートのためのメディア, 351
定義, 53
定義 ローカル スタイル, 357
複数を合成, 364
スタイル レポジトリ,
サイドバー ウィンドウ, 51
デザイン入力ヘルパーも参照, 39
と グローバル スタイル, 354
と 外部CSS スタイルシート, 351
スタイルシート,
CSS スタイルシートを参照, 351
XSLT スタイルシートを参照, 351
ステータスバー, 741
スペルチェック,
StyleVision 内, 802
スペルチェック オプション,
SPS のための, 803
セクション,
SPS デザイン内, 378
イニシャル ドキュメント セクション, 380
と ページ レイアウト, 378
削除, 378
新規の セクションのためのデフォルトのプロパティ, 380
挿入, 767
セル,
SPS テーブル内の結合, 780
セルコンテンツの上下揃え,
SPS 内, 783
セルの結合,
SPS テーブル内, 780
その他のコンテンツ, 132
クイックスタートチュートリアル内, 95
デザイン内に挿入のためのコマンド, 745
と グローバル テンプレート, 232
ソフトウェアの比較,
ダウンロードのための, 820
ソフトウェアライセンス認証, 817
ソフトウェア製品ライセンス, 1131
ダイアログのカスタマイズ,
StyleVision のためのカスタマイズ, 741
タグ,
展開 と 縮退, 732
タグで囲むメニュー, 770
チェックボックス, 180
ツールバー, 680
Authentic ツールバー, 685
GUI の配置, 680
カスタマイズ, 741
テーブル ツールバー, 683
フォーマット ツールバー, 682
リッチエディット, 687
移動, 33
再設定, 680
追加/移動アイコン, 680
標準 ツールバー, 691
表示のオンとオフ, 680
表示のオンとオフの切り替え, 741
ツールバーとウィンドウの復元, 813
ツールバーを使用してデザイン要素を挿入する,
デザイン要素 ツールバーの挿入, 687
ツールメニュー, 802
データ-エントリデバイス, 178
デザインビュー内のシンボル, 661
挿入のためのメニューコマンド, 747
データベース,
DB を参照, 469, 616
とグローバルリソース, 414
編集のためのツールバー ボタン, 691
データベース (Enterprise と Professional Edition),
DB を参照 , 17
データベース クエリ,
DB クエリ ウィンドウ内のブラウザーペイン, 583
クエリ のためにに接続する', 580
クエリの作成, 589
の結果, 590
メッセージペイン, 590
データベース クエリ ウィンドウ, 579
ビューのオフとオンの切り替え, 796
データベースドライバー,
概要, 475
データベース接続,
ウィザードの開始, 474
グローバルリソースからの再使用, 501
設定, 473
データ型,
DB フィルター内, 572
テーブル,
Authentic View 内の編集, 794
HTML ドキュメントからインポートされた, 456
navigating, 778
PDF 内のヘッダー と フッター, 155
SPS 内の削除, 778
StyleVision 内での境界の表示非表示, 781

テーブル,
から行を削除, 780
から列を削除, 780
セルコンテンツ, 778
セルコンテンツの上下揃え, 783
セルの結合, 780
フォーマット, 155
ヘッダー と フッター, 778
ヘッダー と フッターの追加, 779
概要, 145
空の列の非表示, 159
行の挿入, 779
行の追加, 779
行の展開/折りたたみ, 159
作成, 755
条件付きの処理, 152
静的な テーブルの挿入, 778
静的なテーブル, 146
代替の行のためのスタイル, 362
動的な(SPS) テーブルの編集, 594
動的なテーブルの作成, 148
編集 プロパティの, 782
列の挿入, 779
列の追加, 779
列を非表示にするボタン, 159
テーブル (SPS),
プロパティの編集, 683
編集のためのツールバー ボタン, 683
テーブル セルコンテンツの配置,
SPS 内, 783
テーブル ダイアログの選択,
DB-ベースの SPS, 564
DB-ベースの SPS ための, 557
テーブル メニュー, 778
テーブル 内の Authentic View,
SPS (静的と動的) テーブルの使用, 608
XML テーブルの編集のためのアイコン, 613
テーブルセルの分割,
左右, 781
上下, 781
テーブル内の Authentic View,
XML テーブルの使用, 609
テキスト,
Authentic View 内の書式設定, 604
Authentic View 内の編集, 604
テキストの範囲,
内の Authentic View, 601
テキスト参照, 324
テキスト状態アイコン, 374, 786
デザイン コマンドの保存, 705
デザイン ツリー,
サイドバー ウィンドウ, 48
と モジュール SPS, 222
デザイン テンプレートのためのフィルター,
スイッチオン と オフ, 743
デザイン フィルター,
スイッチオン と オフ, 743
デザイン フラグメント,
挿入, 766
デザイン 概要,
サイドバー ウィンドウ, 42
デザイン 構造, 203
デザインツリー,
デザイン入力ヘルパーも参照, 39
デザインビュー,
SPS デザイン内のシンボル, 661
と JavaScript エディター, 35
マークアップの表示, 35
詳細, 35
デザインビュー内のシンボル,
XML ドキュメントコンテンツの, 661
XML ドキュメントノードの, 661
イメージの, 661
データ-エントリデバイスの, 661
ハイパーリンクの, 661
ブックマーク (アンカー)の, 661
自動計算の, 661
条件付きのテンプレートの, 661
定義済みフォーマットの, 661
デザインビュー内のテーブル,
タグで囲まれた と 削除 テンプレート, 153
表示, 153
デザインビュー内のマークアップタグ, 35
デザインビュー内の入力ヘルパー,
表示のオンとオフの切り替え, 742
デザインフラグメント, 242
デザイン入力ヘルパー,
自動的に非表示, 39
詳細, 39
非表示, 39
表示のオンとオフの切り替え, 742
デザイン入力ヘルパー ウィンドウ,
ドッキング, 39
フロート, 39
デザイン要素, 687
デフォルトの ユーザー 辞書,

デフォルトの ユーザー 辞書,
SPS スペルチェックのための, 803
テンプレート,
タグで囲まれた, 770
テーブル 行と列の移動, 153
テーブル 行と列をタグで囲む, 153
と一致するノードの変更, 200
のツリー, 48
ビューのオンとオフの切り替え, 743
挿入, 763
テンプレート XML ファイル (Enterprise と Professional Edition), 102
定義, 28
テンプレート フィルター, 690
テンプレート一致 コマンドの編集, 139
ドキュメント,
開く と 閉じる, 34
概要, 31
ドキュメント スタイル, 255
ドキュメント プロパティ, 255
ドキュメント 要素 (ルート 要素参照), 205
ドキュメントノード,
定義, 28
ドキュメントビュー,
GUI 内, 34
ドキュメントレベル,
XMLSpy の統合のサンプル, 971
ドキュメント要素,
定義, 28
ドッキング,
デザイン入力ヘルパー ウィンドウ, 39
ノード,
として作成されているノードの変更, 200
ノードテンプレート,
と XPath フィルター, 239
と グローバル テンプレート, 239
ユーザー定義, 139
子テンプレートに変換, 239
操作, 239
ノードテンプレート上の XPath フィルター, 239
ノードテンプレート上のフィルター, 239
ノードのためのテンプレート,
ノードテンプレートを参照, 239
ノードの更新,
自動計算の値を使用して, 263
ノードの更新 (Enterprise と Professional Edition),
自動計算の値を使用して, 261
自動計算結果を使用して, 258
バーコード, 187, 750
パーサー,
Altova 製品に内蔵の, 1128
ハイパーリンク, 186, 326
タグで囲まれた, 773
デザインビュー内のシンボル, 661
デザイン内に挿入のためのコマンド, 758
とパースされていないエントリ, 328
ハイパーリンクを使用して検索, 422
ブックマークにリンク, 328
移動 と削除, 328
外部リソースにリンク, 328
作成と編集, 328
バッチ ファイル,
と 計画されたタスク, 645
パラメーター, 290
DB クエリ内, 617
DB フィルター内, 572
SPS 内, 290
スキーマソースのため, 294
デザイン 概要 サイドバー内の管理, 42
デザインフラグメントのための, 291
と Authentic View, 290
と コマンドライン, 290
ユーザー定義 パラメーターの概要, 48
一般的な説明, 290
作成と編集, 731
複数の ドキュメント内でのノードの検索, 294
パラメーター ダイアログの編集, 731
パラメーター デフォルト,
DB フィルター内, 572
バリデーター,
Altova 製品内で, 1128
ピクセル,
と 印刷メディアの長さ, 717
とスクリーンの解像度, 717
ピクセルにより定義された長さ,
と ページ付けされたメディア, 394
ビュー,
GUI レイアウト, 39
ビューの変更,
Authentic View, 594
ファイル,
最後に開かれたファイルを開く, 724
ファイル DSN,
セットアップ, 491
ファイル メニュー, 694
コマンドの終了, 724

ファイル メニュー, 694
ファイル | Authentic XML データの保存, 713
ファイル | エンコード, 717
ファイル | デザインの保存, 705
ファイル | 印刷, 723
ファイル | 印刷プレビュー, 723
ファイル | 開く, 700
ファイル | 新規の, 694
ファイル | 生成された ファイルの保存, 714
ファイル | 閉じる, 700
ファイル | 名前をつけて保存, 710
ファイル 変更の通知,
モジュール SPS 内, 222
ファイルコマンド, 726
フィルター,
テンプレート選択して確認するための, 690
フィルター (DB のための),
クリア, 797
編集, 797
フォーマット,
ツールバー ボタン, 682
テーブルのための, 155
テキスト プロパティ, 682
テキストの配置, 682
プロシージャの概要, 335
リスト, 682
数字フィールドの, 340
挿入時のノード, 129
定義済みの HTML フォーマット, 682
表示も参照, 335
フォーマット 設定,
入力 フォーマットのための定義, 800
フォーマット 番号,
自動番号, 321
フォーム-ベースの デザイン, 191, 694
ブックマーク, 186, 326
タグで囲まれた, 773
デザイン内に挿入のためのコマンド, 757
作成と編集, 326
削除, 326
ブックマーク (アンカー),
デザインビュー内のシンボル, 661
フッター,
テーブル内, 155
テーブル内に追加, 779
フッター と ヘッダー,
サブトータルを含む, 390
ページ番号が付いたメディア出力内, 388
番号付けられた媒体の出力内, 390
ブラウザーペイン,
データベース クエリ ウィンドウ内, 583
プラットフォーム,
Altova 製品のための, 1128
プレビューの問題, 27
フロート,
デザイン入力ヘルパー ウィンドウs, 39
プロジェクト,
とドラッグアンドドロップ, 734
の詳細, 112
使用方法, 112
プロジェクト オプション, 813
プロジェクト サイドバー, 58
プロジェクト メニュー, 734
アクティブおよび関連するファイルのプロジェクトへの追加, 738
アクティブなファイルのプロジェクトへの追加, 738
プロジェクト コマンドに URL を追加する, 736
プロジェクトにファイルを追加するコマンド, 736
プロジェクトフォルダーをプロジェクトに追加する, 738
プロジェクに Altova リソースを追加するト コマンド, 736
開く コマンド, 735
外部フォルダー/ウェブフォルダーをプロジェクトに追加するコマンド, 738
再ロード コマンド, 735
新規の コマンド, 735
閉じるコマンド, 735
保存 コマンド, 735
プロジェクトに URL を追加する, 736
プロジェクトにファイルを追加する, 736
プロジェクトの再ロード, 735
プロジェクトの保存, 735
プロジェクトフォルダーをプロジェクトに追加, 738
プロジェクトを閉じる, 735
プロジェクに Altova リソースを追加する, 736
プロセッサー,
ダウンロードのための, 820
プロパティ,
Authentic View 内のためのノード, 416
SPS テーブルの, 683, 782
サイドバー ウィンドウ, 54
デザイン入力ヘルパーも参照, 39
と プロパティグループ, 54
定義, 54
プロパティ メニュー, 799
箇条書きと段落番号, 799
プロパティ入力ヘルパー,
イベントグループ, 448

ページ, 383
PDF 出力のためのサイズの設定, 383
PDF 出力のための余白の設定, 383
PDF 出力の番号付け, 383
デザインのためのコマンド, 767
ページ総数(Enterprise Edition), 767
ページ番号(Enterprise Edition), 767
ページ番号が付いたメディア,
のためのデザイン, 377
プロパティ, 377
ページ サイズ, 383
ページ 定義, 383
ページ番号が付いた, 383
ヘッダー と フッター, 388
余白, 383
ページ付けされたメディア,
とピクセルにより定義された長さ, 394
ベース年度,
入力 フォーマット内, 340
ヘッダー,
テーブル内, 155
テーブル内に追加, 779
ヘッダー と フッター,
サブトータルを含む, 390
ページ番号が付いたメディア出力内, 388
号付けられた媒体の出力内, 390
ヘルプ,
スクリーン上のヘルプ参照, 817
ヘルプ メニュー, 817
ボタン, 184
マークアップ,
Authentic View 内, 793
内の Authentic View, 594, 595
マークアップの非表示, 35, 594
マークアップの表示, 35
マークアップを隠す, 595
メイン スキーマ, 231
メイン スキーマ (Enterprise Edition のみ), 44
メイン テンプレート, 231
定義, 28
メッセージサイドバー, 59
メニュー,
カスタム化, 808
メニュー バー,
移動, 33
メニューの挿入, 744
箇条書きと段落番号, 755
挿入 | イメージ, 752
挿入 | コンテンツ, 744
挿入 | その他のコンテンツ, 745
挿入 | デザイン フラグメント, 766
挿入 | ハイパーリンク, 758
挿入 | ブックマーク, 757
挿入 | ページ, 767
挿入 | 自動計算, 748
挿入 | 条件, 761
挿入 | 水平線, 754
挿入 | 段落, 750
挿入 | 特殊な 段落, 750
挿入 | 日付選択, 749
メモリ要件, 1128
モジュール,
デザイン 概要 サイドバー内の管理, 42
モジュール SPS,
SPS モジュールに追加する, 222
サンプル プロジェクト, 226
と CSS スタイル, 219, 222
と スキーマソース, 219, 222
と スクリプト, 219
と テンプレート XML ファイル, 219
と ファイル 変更の通知, 222
と モジュールオブジェクト, 219
と 作業用 XML ファイル, 219
と 名前空間宣言, 219
と作業, 222
概要, 218
作成, 222
追加する SPS モジュール, 222
優先順位の順序の効果, 222
有効化と無効化, 222
やり直しコマンド, 726
ユーザー DSN,
セットアップ, 491
ユーザー インターフェイス,
GUI を参照, 33
ユーザー マニュアル,
スクリーン上のヘルプも参照, 817
ユーザー参照, 660
ユーザー情報, 416
ユーザー定義 XML テキストブロック, 143
ユーザー定義 XPath 関数, 427
ユーザー定義 スキーマ, 212
ユーザー定義 テキストブロック, 768
ユーザー定義 テンプレート, 139
タグで囲まれた, 771
挿入, 764

ユーザー定義 要素, 142, 768, 776
ユーザー定義テキストブロック, 142
ユーザー定義要素, 142
ライセンス, 1131
Altova ソフトウェアのための, 817
情報, 1130
ライセンス計測,
Altova 製品にて, 1130
ラジオ ボタン, 184
リスト, 166, 772
HTML ドキュメントからインポートされた, 456
クイックスタートチュートリアル内, 88
タグで囲まれた, 772
リスト (静的 と 動的), 755
リスト プロパティ, 799
リッチエディット, 370
Authentic View 内の定義, 794
挿入, 745
リッチエディット ツールバー, 687
リンク,
Authentic View 内の, 604
ハイパーリンクを参照, 186, 326
ルート 要素, 44
ルート 要素 (ドキュメント 要素参照),
スキーマのために選択, 205
と スキーマソース, 205
レイアウト,
GUI 内のビュー, 39
レイアウト コンテナー, 191, 766
レイアウト コンテナー と 要素, 687
レイアウト ボックス, 766
レイアウト モジュール,
作成のためのステップ, 191
レイアウトのための青写真, 191
レイアウトボックス, 194
ローカル スタイル,
スタイルを参照, 357
ローカル テンプレート, 231
一時的な 出力 ドキュメント, 27
一般的な使用方法, 101
印刷 コマンド, 723
印刷 出力,
ページ番号が付いたメディア, 377
印刷 出力のための列, 383
印刷プレビュー コマンド, 723
箇条書きと段落番号, 166, 168, 755, 799
タグで囲まれた, 772
改ページ, 392, 767
開く,
最後に開かれた ファイル, 724
開く (SPS) コマンド, 700
開く プロジェクト, 735
外部キー,
SQLite 内での無効化, 500
外部フォルダー/
ウェブフォルダーをプロジェクトに追加する, 738
環境変数, 124
関数,
in XPath 内でユーザーにより定義されている, 427
機能,
StyleVision, 20
技術データ, 1128
境界,
SPS テーブルの, 781
型-ベースの テンプレート, 232
計画されたタスク,
StyleVisionBatch コマンドの作成, 645
StyleVisionBatch バッチ ファイル, 645
検索, 60
正規表現の使用, 726
元に戻すコマンド, 726
構成,
グローバル リソースの, 807
行,
SPS テーブルに挿入, 779
SPS テーブルに追加, 779
SPS 内のテーブルから削除, 780
合成 スタイル, 364
Authentic View のための, 368
最後に開かれた ファイル, 724
作業用 XML ファイル, 44, 102
Authentic View 内の検証, 791
と Authentic View, 35
と 出力ビュー, 37
印刷, 723
印刷プレビュー, 723
定義, 28
作成 新規の SPS ドキュメント,
クイックスタートチュートリアル内, 62
削除,
DB フィルター内, 572
SPS 内のテーブル, 778
SPS 内のテーブルから行を削除, 780
SPS 内のテーブルから列を削除, 780
削除 コマンド, 726
使用方法, 101

視覚的な書式設定,
内の Authentic View, 625
試用期間,
Altova ソフトウェア製品の試用, 1130
次を検索コマンド, 726
自動化された 処理, 637
自動計算, 258
Java と :NET 関数 (Enterprise Edition のみ), 258
XML ドキュメント内のノードを更新する, 261
XML ドキュメント内のノードを値を使用して更新する, 263
クイックスタートチュートリアル内, 84
サンプル, 281
サンプル ファイル, 264
デザインビュー内のシンボル, 661
デザイン内に挿入のためのコマンド, 748
と 出力 エスケープ, 338
と 条件, 273
ノードを値で更新する, 748
移動, 258
作成、 編集、フォーマット, 258
使用方法, 258
自動計算の結果をベースにして, 263
日付結果のフォーマット, 443
非表示, 261
自動的なノードの番号付け, 321
自動的に非表示,
デザイン入力ヘルパーの機能, 39
自動番号, 321
辞書,
SPS スペルチェックのための, 803
式,
DB フィルター条件内, 572
実行中の総数,
ヘッダー と フッター内, 390
終了 コマンド, 724
出力 エスケープ, 338
出力 エンコード, 717
出力 ドキュメントのエンコード, 717
出力 ドキュメントのスタイル, 255
出力 ドキュメントのプロパティ, 255
出力-エスケープの無効化, 338
出力ビュー,
詳細, 37
出力ファイル,
DB-ベースの <%SV-PS%> から, 578
生成, 110
出力-ベースの 条件, 272
処理ユニットとしての型,
グローバル テンプレート内, 232
小さなマークアップ, 595
小さなマークアップの表示, 594
小数,
のフォーマット, 340
省略,
ユーザーマニュアル内で使用されている, 31
上下のテキスト,
テーブル セル内, 155
レイアウトボックス内, 194
条件,
DB フィルター内, 572
クイックスタートチュートリアル内, 88
セットアップ, 268
タグで囲まれた, 774
デザイン内に挿入のためのコマンド, 761
と 自動計算, 273
出力の差異のための, 272
出力-ベースの, 272
編集, 271
条件付きのテンプレート, 761
デザインビュー内のシンボル, 661
条件を参照, 268
条件付きの表示, 275
新規の SPS ドキュメントのセットアップ,
クイックスタートチュートリアル内, 62
新規の コマンド, 694
新規の ドキュメント テンプレート, 248
URL, 250
と デザイン 構造, 250
挿入, 249
新規の プロジェクト, 735
新規機能, 18
水平線,
クイックスタートチュートリアル内, 73
デザイン内に挿入のためのコマンド, 754
数字,
のフォーマット, 340
数字フィールド,
フォーマットの, 340
整数,
のフォーマット, 340
正規表現,
検索と置換の条件式, 726
生成された ファイルの保存コマンド, 714
生成されたファイル, 110
製品の機能,
リスト, 20

静的コンテンツ,
クイックスタートチュートリアル内, 73
静的テキスト,
クイックスタートチュートリアル内, 73
と 出力 エスケープ, 338
静的な (SPS) テーブル 内の Authentic View,
使用方法, 608
静的な テーブル,
SPS 内に挿入, 683
挿入, 778
編集のためのツールバー ボタン, 683
静的な リスト, 166, 755
静的なテーブル, 145
SPS テーブルも参照してください。, 146
テーブルも参照, 155
切り取りコマンド, 726
選択されたノードに対しての XPath, 593
全て選択コマンド, 726
挿入,
SPS テーブル内の行, 779
SPS テーブル内の列, 779
相互参照, 324
属性入力ヘルパー,
Authentic View, 598
代替の 処理, 418
大きなマークアップ, 595
大きなマークアップの表示, 594
段落,
タグで囲まれた, 772
デザイン内に挿入のためのコマンド, 750
値フォーマット, 340
置換, 60
正規表現の使用, 726
置換コマンド (Enterprise と Professional エディション), 726
著作権に関する情報, 1130
追加,
SPS 内のテーブルの行, 779
SPS 内のテーブルの列, 779
追加検証, 420
追加編集 プロシージャ, 400
定義済みの フォーマット 設定,
入力 フォーマットのための, 800
定義済みフォーマット,
デザインビュー内のシンボル, 661
デザイン内に挿入のためのコマンド, 750
ノードの挿入, 129
定義済みフォーマットの割り当て,
クイックスタートチュートリアル内, 78
貼り付け,
XML, 601
XML として, 604
テキスト, 601
テキストとして, 604
貼り付けコマンド, 726
統合,
アプリケーション内の StyleVision, 963
透かし, 396
動的コンテンツ,
クイックスタートチュートリアル内, 66
動的な (SPS) テーブル 内の Authentic View,
使用方法, 608
動的な テーブル,
編集のためのツールバー ボタン, 683
動的な リスト, 166, 168, 755
動的なテーブル, 145
Authentic View 内の編集, 794
SPS テーブルも参照, 148
テーブルも参照, 155
と グローバル テンプレート, 148
ネスとされた動的なテーブル, 148
ヘッダー と フッター, 148
追加された/挿入された行との差異, 148
編集, 594
特殊な 段落,
タグで囲まれた, 772
デザイン内に挿入のためのコマンド, 750
内の Authentic View テーブル,
使用方法, 608
日付,
SPS 内での使用, 441
XPath 2.0 を使用したデータ操作のサンプル, 441
と 日付選択, 442, 749, 790
のフォーマット, 340
フォーマットの, 443
手動での変更, 623
日付の選択,
内の Authentic View の使用, 622
日付選択,
Authentic View 内で使用, 442
SPS 内に挿入, 442
デザイン内に挿入のためのコマンド, 749
デフォルトで日付ノードに追加する, 790
と XSD データ型, 442
使用方法の詳細, 442
日付の構文フォーマット, 442

日付選択の自動追加, 790
入力 フィールド, 179
入力 フォーマット,
定義 フォーマット 設定のための, 800
日付の, 443
背景情報, 1128
配布,
Altova ソフトウェア製品, 1130
Altova ソフトウェア製品の配布, 1130
非表示,
デザイン入力ヘルパーの機能, 39
表示,
フォーマット、フォーマットも参照, 335
プロシージャの概要, 335
表示メニュー, 741
評価キー,
Altova ソフトウェアのための, 817
評価機関,
Altova ソフトウェア製品, 1130
付録, 1023
複合型の グローバル テンプレート, 232
複合型マークアップ, 416, 595
複合型マークアップの表示, 594
複数の 出力-ドキュメント, 248
と 出力 プレビュー, 253
ファイルの場所, 253
リンクする, 250
複数のドキュメントの出力, 767
複数の言語サンプル, 332
複数の入力 フィールド, 179
文字 参照,
と 出力 エスケープ, 338
並べ替え, 284
グループとグループ内, 277, 279, 281
サンプル ファイル, 286
並べ替え メカニズム, 284
並べ替えキー, 284
並べ替えキー, 284
閉じる (SPS) コマンド, 700
変更コマンド, 200
変数, 290, 294
Authentic View 内の編集, 296
Authentic を使用して値を推定する, 418
変数 テンプレート, 238
タグで囲まれた, 771
挿入, 765
編集することのできる 変数, 296
編集メニュー, 726
保存,
作業用 XML ファイル, 713
法的な情報, 1130
名前が付けられたテンプレート, 231
名前の追加, 416
名前空間,
SPS に追加, 44, 110
SPS へ追加, 102, 205
SPS 内の, 44
の概要, 48
目次, 300
サンプル, シンプルな, 314
サンプル、階層ならびに連続的な, 317
メニュー コマンド, 766
含まれるアイテムのためのマーク付け, 302
使用方法の概要, 300
目次 を参照, 300
目次 アイテム,
フォーマット, 313
構築, 313
目次 テンプレート,
フォーマット, 313
レベル参照, 311
構成, 311
作成と編集, 309
参照レベル, 311
目次 ブックマーク, 302
タグで囲まれた, 775
と レベル, 307
のためのウィザード, 307
作成, 307
目次 レベル, 302, 304
タグで囲まれた, 775
目次参照, 312
目次参照を参照, 312
優先順位,
のスタイル, 51
用語,
StyleVision 内で使用されている, 28
要素,
Authentic View と SPS 内に追加, 130
ユーザー定義, 142
要素 テンプレート,
ユーザー定義, 142
要素入力ヘルパー,
Authentic View, 598
列,
SPS テーブルに追加, 779

列,
SPS テーブル内に挿入, 779
SPS 内のテーブルから削除, 780
強制改行, 767
挿入, 767
列挙,
StyleVisionControl, 1021
連続した マークアップ, 35
曖昧な,
コンテンツの, 130