<oXygen/> XML Editor 10.3 User Manual for Eclipse

SyncRO Soft Ltd.

Contributor: Sean Wheller

<oXygen/> XML Editor 10.3 User Manual for Eclipse

SyncRO Soft Ltd.
Contributor: Sean Wheller

Copyright © 2002-2009 SyncRO Soft Ltd. All Rights Reserved.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and SyncRO Soft Ltd., was aware of a trademark claim, the designations have been printed in caps or initial caps. While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Third party software components are distributed in the <oXygen/> installation packages, including the Java Runtime Environment (JRE), DocBook DTD and stylesheets. This product includes software developed by the Apache Software Foundation (http://www.apache.org [http://www.apache.org]): the Apache FOP, Xerces XML Parser and Xalan XSLT . These products are not the property of SyncRO Soft Ltd.. To the best knowledge of SyncRO Soft Ltd. owners of the aforesaid products granted permission to copy, distribute and/or modify the software and its documents under the terms of the Apache Software License, Version 1.1. Other packages are used under the GNU Lesser General Public License. Users are advised that the JRE is provided as a free software, but in accordance with the licensing requirements of Sun Microsystems. Users are advised that SyncRO Soft Ltd. assumes no responsibility for errors or omissions, or for damages resulting from the use of <oXygen/> and the aforesaid third party software. Nor does SyncRO Soft Ltd. assume any responsibility for licensing of the aforesaid software, should the relevant vendors change their terms. By using <oXygen/> the user accepts responsibility to maintain any licenses required by SyncRO Soft Ltd. or third party vendors. Unless SyncRO Soft Ltd. declares in writing that the <oXygen/> license is inclusive of third party licensing.

Table of Contents

Chapter 1. Introduction

Welcome to the User Manual of the <oXygen/> XML Editor 10.3.0 plugin for Eclipse ! This book explains how to use the 10.3.0 version of the <oXygen/> plugin for Eclipse effectively to develop complex XML applications quickly and easily. Please note that this manual assumes that you are familiar with the basic concepts of XML and its related technologies.

The <oXygen/> XML Editor plugin for Eclipse is a cross-platform application for document development using structured mark-up languages such as XML , XSD, Relax NG, XSL, DTD.

<oXygen/> offers developers and authors a powerful Integrated Development Environment. Based on proven Java technology the intuitive Graphical User Interface of the <oXygen/> plugin for Eclipse is easy-to-use and provides robust functionality for editing, project management and validation of structured mark-up sources. Coupled with XSLT and FOP transformation technologies, <oXygen/> supports output to multiple target formats, including: PDF, PS, TXT, HTML and XML.

Key Features and Benefits

The offers the following key features and benefits.

Multiplatform availability: Windows, Mac OS X, Linux, Solaris	Non blocking operations, you can perform validation and transformation operations in background
Visual WYSIWYG XML editing mode based on W3C CSS stylesheets.	Visual DITA Map editor
Closely integration of the DITA Open Toolkit for generating DITA output	Support for latest versions of document frameworks: DocBook and TEI.
Support for XML, XML Schema, Relax NG , Schematron, DTD, NRL schemas, NVDL schemas, XSLT, XSL:FO, WSDL, XQuery, HTML, CSS	
Validate XML Schema schemas, Relax NG schemas, DTDs, Schematron schemas, NRL, NVDL schemas, WSDL, XQuery, HTML and CSS	Manual and automatic validation of XML documents against XML Schema schemas, Relax NG schemas, DTDs, Schematron schemas and NRL, NVDL schemas
Multiple built-in validation engines (Xerces, libxml, MSXML 4.0, MSXML.NET) and support for custom validation engines (Saxon SA, XSV, SQC, Intel® XML Software Suite).	Multiple built-in XSLT transformers (Saxon 6.5, Saxon B, Saxon SA, Saxon.NET, Xalan, libxslt, MSXML 3.0 / 4.0, Microsoft .NET 1.0, Microsoft .NET 2.0), support for custom JAXP transformers and also Intel® XML Software Suite.
Visual schema editor with full and logical model views	Generate HTML documentation from XML Schemas
Ready to use FOP support to generate PDF or PS documents	XInclude support
Context sensitive content assistant driven by XML Schema, Relax NG, DTD, NVDL or by the edited document structure enhanced with schema annotation presenter	New XML document wizards to easily create documents specifying a schema or a DTD
XML Catalog support	Unicode support
Conversions from DTD, Relax NG schema or a set of documents to XML Schema, DTD or Relax NG schema	Syntax coloring for XML, DTD, Relax NG compact syntax, Java, C++, C, PHP, Perl, etc
Easy error tracking - locate the error source by clicking on it	Easy configuration for external FO Processors

Apply XSLT and FOP transformations	XPath search and evaluation support
Preview transformation results as XHTML or XML or in your browser	Support for document templates to easily create and share documents
Import data from a database, Excel, HTML or text file	Convert database structure to XML Schema
Batch validate selected files in project	Canonicalize and sign documents
Configurable actions key bindings	Associate extensions with editors provided by the <oXygen/> plugin.
XSLT Debugger with Backmapping support	XSLT Profiler
XQuery Debugger with Backmapping support	XQuery Profiler
Model View	Attributes View
XQuery 1.0 support	WSDL analysis and SOAP requests support
XSLT 2.0 full support	XPath 2.0 support
Document folding	Spell checking supporting English, German and French including locals
XSLT refactoring actions	Generate large sets of sample XML instances from XML Schema
Pretty-printing of XML files	Drag&drop support
Outline view in sync with a non well-formed document	

About the <oXygen/> User Manual

This User Manual gives a complete overview of the <oXygen/> XML Editor and describes the basic process of authoring, management, validation of structured mark-up documents and their transformation to multiple target outputs. In this manual it is assumed that you are familiar with the use of your operating system and the concepts related to structured mark-up.

The <oXygen/> XML Editor User Manual is comprised of the following parts:

- Chapter 1, *Introduction*: you are reading it.
- Chapter 2, *Installation*: defines the platform and environment requirements of <oXygen/> and instructions for application installation, license installation, starting <oXygen/>, upgrade and uninstall.
- Chapter 3, *Getting started*: provides general orientation and an overview of the <oXygen/>'s editing perspectives.
- Chapter 4, *Editing documents*: explains how to obtain maximum benefit from the editor, project and validation features.
- Chapter 9, *Transforming documents*: explains the considerations for transformation of structured sources to multiple target format and how to obtain maximum benefit.
- Chapter 10, *Querying documents*: explains the support offered by <oXygen/> for querying XML documents.
- Chapter 11, *Debugging XSLT stylesheets and XQuery documents*: explains how to debug XSLT stylesheets or XQuery documents.
- Chapter 12, *Profiling XSLT stylesheets and XQuery documents*: explains how to profile the execution of XSLT stylesheets or XQuery documents.
- Chapter 15, *Importing data*: explains how to import data from a database, an Excel sheet or text file.

- Chapter 16, *Composing Web Service calls*: explains the facilities offered by <oXygen/> for composing and testing WSDL SOAP messages.
- Chapter 17, *Digital signature*: explains how to canonicalize, sign and verify the signature of documents.
- Chapter 19, *Configuring the application*: explains how to configure preferences of the application.

Feedback and input to the <oXygen/> User Manual is welcomed.

Chapter 2. Installation

This section explains platform requirements and installation procedures. It also provides instructions on how to obtain and register a license key, how to perform upgrades and uninstall the application if required.

If you need help at any point during these procedures please send email to `<support@oxygenxml.com>`

Installation Requirements

Platform Requirements

Minimum run-time requirements are listed below.

- Pentium Class Platform
- 256 MB of RAM
- 300 MB free disk space

Operating System, Tools and Environment Requirements

Operating System

Windows	Windows 98 or later.
Mac OS	minimum Mac OS X 10.4
UNIX/Linux	All versions/flavors

Tools

Installation packages are supplied in compressed archives. Ensure you have installed a suitable archive extraction utility with which to extract the archive. The MD5 sum is available on the Download page [http://www.oxygenxml.com/download.html] for every archive. You should check the MD5 sum of the downloaded archive with a MD5 checking tool available on your platform.

Environment Prerequisites

Prior to installation ensure that your installed Eclipse platform is the following:

- the latest stable Eclipse version available at the release date. The current version works with Eclipse 3.4.
- <oXygen/> XML Editor supports only official and stable Java virtual machine versions 1.5.0 and later from Sun Microsystems (available at http://java.sun.com) and from Apple Computer (pre-installed on Mac OS X). For Mac OS X, Java VM updates are available at http://www.apple.com/macosx/features/java/. <oXygen/> XML Editor may work very well with JVM implementations from other vendors but the eventual incompatibilities will not be solved in further <oXygen/> XML Editor releases. <oXygen/> XML Editor does not work with the GNU libgcj Java virtual machine [http://www.oxygenxml.com/forum/ftopic1887.html].

Installation Instructions

Prior to proceeding with the following instructions, please ensure that your system complies with the prerequisites detailed in the installation requirements.

There are two ways of installing the <oXygen/> Eclipse plugin: the Update Site method and the zip archive method.

Procedure 2.1. Eclipse 3.3 plugin installation - the Update Site method

1. Start Eclipse. Choose the menu option: Help / Software Update / Find and Install. Select the checkbox: "Search for new features to install" and press the "Next" button..
2. From the dialog "Update sites to visit" press the button "Add update site" or "New Remote Site".
3. Enter the value `http://www.oxygenxml.com/InstData/Eclipse/site.xml` into the "URL" field of the "New Update Site" dialog. Press the "OK" button.
4. Select the checkbox "oXygen XML Editor" and press the "Next" button.
5. Select the new feature to install "oXygen XML Editor and XSLT debugger" and press the "Next" button in the following install pages. You must accept the Eclipse restart.
6. Paste the license information received in the registration email when prompted. This will happen when you use one of the wizards to create an XML project or document, when you open or create a document associated with <oXygen/> or when accessing the <oXygen/> Preferences.
7. The oXygen XML Editor plugin is installed correctly if you can create an XML project with the New Project wizard of the oXygen XML Editor plugin started from menu File -> New -> Other -> oXygen -> XML Project.

Procedure 2.2. Eclipse 3.4 plugin installation - the Update Site method

1. Start Eclipse. Choose the menu option: Help / Software Updates / Available Software.
2. Press the button "Add Site" in the tab "Available Software" of the dialog "Software Updates".
3. Enter the value `http://www.oxygenxml.com/InstData/Eclipse/site.xml` into the "Location" field of the "Add Site" dialog. Press the "OK" button.
4. Select the checkbox "oXygen XML Editor for Eclipse" and press the Install button.
5. Press the Next button in the following install pages. You must accept the Eclipse restart at the end of the installation.
6. Paste the license information received in the registration email when prompted. This will happen when you use one of the wizards to create an XML project or document, when you open or create a document associated with <oXygen/> or when accessing the <oXygen/> Preferences.
7. The oXygen XML Editor plugin is installed correctly if you can create an XML project with the New Project wizard of the oXygen XML Editor plugin started from menu File -> New -> Other -> oXygen -> XML Project.

Procedure 2.3. Eclipse 3.3 plugin installation - the zip archive method

1. Download [http://www.oxygenxml.com/InstData/Eclipse/com.oxygenxml.editor_10.3.0.zip] the zip archive with the plugin.
2. Unzip the downloaded zip archive in the plugins subdirectory of the Eclipse install directory.

3. Restart Eclipse. Eclipse should display an entry *com.oxygenxml.editor (10.3.0)* in the list available from Window - Preferences - Plug-in Development - Target Platform.

Procedure 2.4. Eclipse 3.4 plugin installation - the zip archive method

1. Download [http://www.oxygenxml.com/InstData/Eclipse/com.oxygenxml.editor_10.3.0.zip] the zip archive with the plugin.

2. Unzip the downloaded zip archive in the `dropins` subdirectory of the Eclipse install directory.

3. Restart Eclipse. Eclipse should display an entry *com.oxygenxml.editor (10.3.0)* in the list available from Window -> Preferences -> Plug-in Development -> Target Platform.

Starting <oXygen/> plugin

The <oXygen/> plugin will be activated automatically by the Eclipse platform when you use one of the <oXygen/> wizards to create an XML project or document, when you open or create a document associated with <oXygen/> or when accessing the <oXygen/> Preferences.

Obtaining and registering a license key

The <oXygen/> XML Editor is not free software and requires a license in order to enable the application.

For demonstration and evaluation purposes a time limited license is available upon request from the <oXygen/> [http://www.oxygenxml.com/register.html] web site. This license is supplied at no cost for a period of 30 days from date of issue. During this period the <oXygen/> XML Editor is fully functional enabling you to test all aspects of the application. Thereafter, the application is disabled and a permanent license must be purchased in order to use the application. For special circumstances, if a trial period of greater than 30 days is required, please contact `<support@oxygenxml.com>`. All licenses are obtained from the <oXygen/> web site [http://www.oxygenxml.com] .

For definitions and legal details of the license types available for <oXygen/> you should consult the End User License Agreement received with the license key and available also on the <oXygen/> website at http://www.oxygenxml.com/eula.html

Note

Starting with version 10.0 <oXygen/> accepts a license key for a newer version in the license registration dialog, e.g. version 10.0 accepts a license key for version 11 or a license key for version 12.

Once you have obtained a license key the installation procedure is described below.

Named User license registration

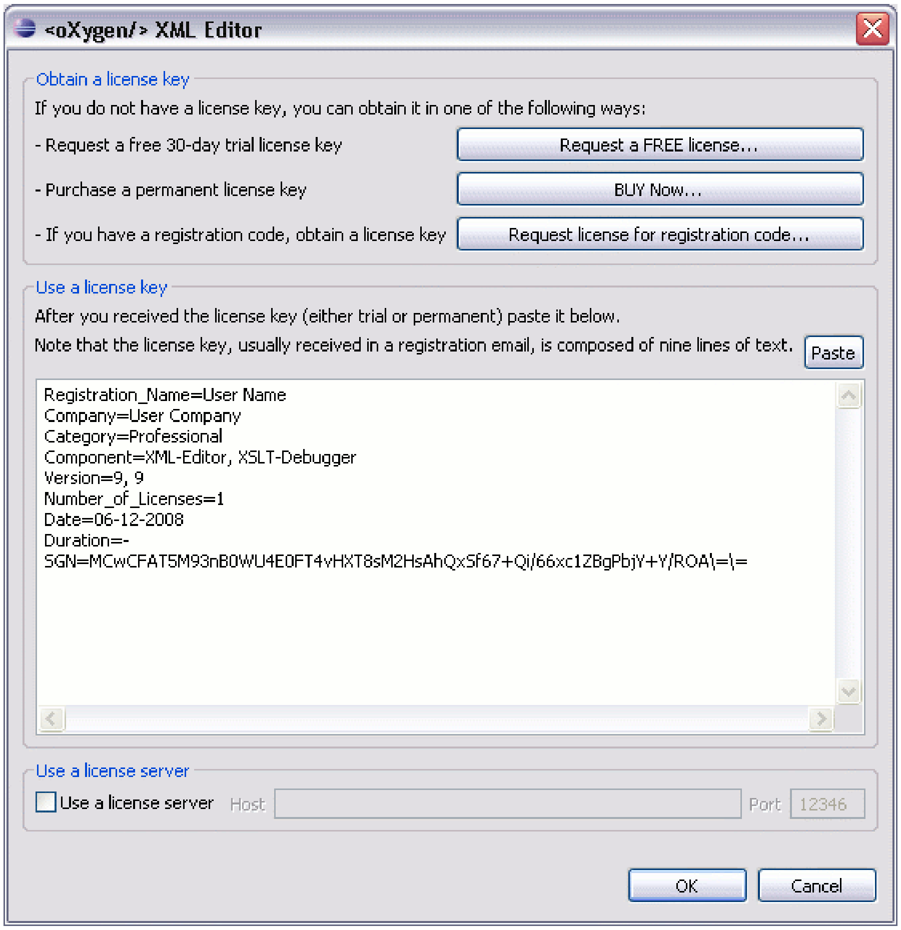
1. Save a backup copy of the message containing the new license key.

2. Start the application.

3. Copy to the clipboard the license text as explained in the message.

4. If this is a new install of the application then it will display automatically the registration dialog when it is started. In the case you already used the application and obtained a new license, go to Window - Preferences - oXygen and press the OK button to make the registration dialog appear.

Figure 2.1. Registration Dialog



If you do not have a license key, you can obtain it in one of the following ways:

- Request a free 30-day trial license key
- Purchase a permanent license key
- If you have a registration code, obtain a license key

After you received the license key (either trial or permanent) paste it below.

Note that the license key, usually received in a registration email, is composed of nine lines of text.

```
Registration_Name=User Name
Company=User Company
Category=Professional
Component=XML-Editor, XSLT-Debugger
Version=9, 9
Number_of_Licenses=1
Date=06-12-2008
Duration=-
SGN=MCwCFAT5M93nB0WU4E0FT4vHXT8sM2HsAhQxSf67+Qi/66xc1ZBgPbjY+Y/ROA\=\=
```

□ Use a license server

5. Paste the license text in the registration dialog, and press OK.

You have the following alternative for the procedure of license install:

Procedure 2.5. Save the license in a text file

1. Save the license key in a file named `licensekey.txt.`

2. Copy the file in the 'lib' folder of the installed plugin. In that way the license will not be asked when <oXygen/> will start.

3. Start Eclipse.

How floating (concurrent) licenses work

If all the floating licenses are used in the same local network the installation procedure of floating licenses is the same as for the Named User licenses. Within the same network the license management is done by communication between the instances of <oXygen/> that are connected to the same local network and that run at the same time. Any new instance of <oXygen/> that is started after the number of running instances is equal with the number of purchased licenses will display a warning message and will disable the open file action.

If the floating licenses are used on machines connected to different local networks a separate license server must be started and the licenses deployed on it.

Procedure 2.6. Floating license server setup

1. Download the license server from one of the download URLs included in the registration email message with your floating license key.

2. Run the downloaded Windows 32 bit installer or Windows 64 bit installer or unzip the all platforms zip archive kit on your server machine. The Windows installer installs the license server as a Windows service, it provides the option to start the Windows service automatically at Windows startup and it creates shortcuts in the Start menu group for starting and stopping the Windows service manually. If you use the zip archive on Windows you have to run the scripts provided in the archive for installing, starting, stopping and uninstalling the server as a Windows service.

3. If you start the server with the script `licenseServer.bat` / `licenseServer.sh` you can leave the default values for the parameters for the licenses folder and server port or you can set these two parameters to other values. The default folder for the floating license file is `[license-server-install-dir]/license` and the default TCP/IP server port is 12346.

   To change the default values of the license server the following parameters have to be used:

   - **-licenseDir** followed by the path of the directory where the license files will be placed;

   - **-port** followed by the port number used to communicate with <oXygen/> instances.

   ## Important

   The license folder must contain a text file called `license.txt` which must contain a single floating license key corresponding to the set of purchased floating licenses. If you have more than one floating license key for the same <oXygen/> version obtained from different purchases please contact us at support@oxygenxml.com to merge your license keys into a single one.

After the floating license server is set up the <oXygen/> application can be started and configured to request a license from it:

Procedure 2.7. Request a floating license from the license server

1. Start Eclipse.

2. Go to *Window* -> *Preferences* -> *oXygen* -> *Register...* . The license dialog is displayed.

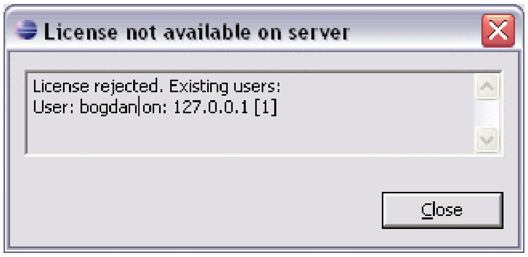
3. Check the *Use a license server* checkbox.

4. Fill-in the *Host* text field with the host name or IP address of the license server.

5. Fill-in the *Port* text field with the port number used for communicating with the license server. Default is 12346.

6. Click the *OK* button. If the maximum number of available licenses was not exceeded a license key is received from the floating license server and registered in <oXygen/> . The license details are displayed in the About dialog opened from menu Help. If the maximum number of licenses was exceeded a warning dialog will pop up letting the user know about the problem.

Figure 2.2. Floating license number exceeded



License rejected. Existing users:
User: bogdan|on: 127.0.0.1 [1]

The error message contains information about the users who requested and successfully received the floating licenses.

How to install the <oXygen/> license server as a Windows service

In order to install the <oXygen/> license server as a Windows service you should run the Windows installer downloaded from the URL provided in the registration email message containing your floating license key.

If you want to install, start and uninstall yourself the server as a Windows service you can run the scripts created in the install folder from a command line console with the install folder of the license server as the current folder (on Windows Vista you have to run the console as Administrator). For installing the Windows service:

```
installWindowsService.bat
```

After installing the server as a Windows service, use the following two commands to start and stop the license server:

```
startWindowsService.bat
```

```
stopWindowsService.bat
```

Uninstalling the Windows service requires the following command:

```
uninstallWindowsService.bat
```

The `installWindowsService.bat` script installs the <oXygen/> license server as a Windows service with the name "oXygenLicenseServer" and accepts two parameters: the path of the folder containing the floating license key files and the local port number on which the server accepts connections from instances of the <oXygen/> XML Editor . The parameters are optional. The default values are:

license	for the license file folder
12555	for the local port number

The JAVA_HOME variable must point to the home folder of a Java runtime environment installed on your Windows system.

The startService.bat script starts the Windows service so that the license server can accept connections from <oXygen/> clients.

The stopService.bat script stops the Windows service. The license server is shut down and it cannot accept connections from <oXygen/> clients.

The uninstallService.bat script uninstalls the Windows service created by the installService.bat script.

When the license server is used as a Windows service the output messages and the error messages cannot be viewed as for a command line script so that they are redirected automatically to the following log files created in the directory where the license server is installed:

outLicenseServer.log	the standard output stream of the server
errLicenseServer.log	the standard error stream of the server

On Windows Vista if you want to start or stop the Windows service with the Start menu shortcut called *Start Windows service* / *Stop Windows service* you have to run the shortcut as Administrator. This is a standard option for running Start menu shortcuts on Windows Vista and is necessary for giving the required permission to the command that starts / stops the Windows service.

How to release a floating license

To release a floating license key so that it can be registered for other user or for the cases when you do not have Internet access (and you own also an individual license to which you want to switch from the floating license), you do not have to disable or to uninstall the <oXygen/> plugin. All you have to do is to go to the main <oXygen/> preferences panel, press the *Register* button, uncheck the *Use a license server* checkbox in the license registration dialog, paste the individual license key and press OK in the dialog. If you only want to stop using the <oXygen/> plugin just uncheck the checkbox and press the OK dialog. This will release the floating license and leave the plugin in the unregistered state.

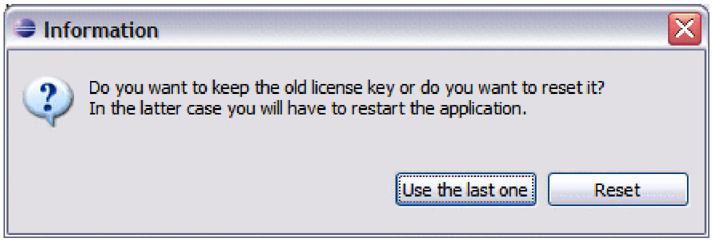
License registration with a registration code

If you have only a registration code and you want to register the associated license key you must request this license key by filling the registration code and other details associated with your license in a request form on the <oXygen/> website. The button **Request license for registration code** in the registration dialog available from menu Window → Preferences+oXygen+Register opens this request form in the default Web browser on your computer.

Unregistering the license key

Sometimes you need to unregister your license key, for example to release a floating license to be used by other user and still use the current <oXygen/> instance with an individual, Named User license, or to transfer your license key to other computer before other user starts using your current computer. This is done by going to Windows → Preferences+oXygen+Register to display the license registration dialog, making sure the text area for the license key is empty and the checkbox *Use a license server* is unchecked, and pressing the OK button of the dialog. This brings up a confirmation dialog in which you select between falling back to the license key entered previously (for the case of releasing a floating license and reverting to the individual license entered previously in the *Register* dialog) and removing your license key from your user account of the computer.

Figure 2.3. Unregister a license key



Do you want to keep the old license key or do you want to reset it?
In the latter case you will have to restart the application.

Upgrading the <oXygen/> application

From time to time, upgrade and patch versions of <oXygen/> are released to provide enhancements that rectify problems, improve functionality and the general efficiency of the application.

This section explains the procedure for upgrading <oXygen/> while preserving any personal configuration settings and customizations.

Procedure 2.8. Upgrade Procedure

1. Uninstall the <oXygen/> plugin (see Uninstall procedure).
2. Follow the Installation instructions.
3. Restart the Eclipse platform.
4. Start the <oXygen/> plugin to ensure that the application can start and that your license is recognized by the upgrade installation.
5. If you are upgrading to a major version, for example from 8.2 to 9.0, then you will need to enter the new license text into the registration dialog that is shown when the application starts.
6. Select Window → Preferences -> Plug-In Development -> Target Platform and next to the *com.oxygenxml.editorcom.oxygenxml.author* list entry you should see the version number of the newest installed plugin. If the previous version was 8.2.0, the list entry should now contain 9.0.0.

Uninstalling the Eclipse plugin

Warning

The following procedure will remove the <oXygen/> XML Editor plugin from your system. It will not remove the Eclipse platform. If you wish to uninstall Eclipse please see its uninstall instructions.

Procedure 2.9. Uninstall Procedure

1. Choose the menu option: Help / Software Update / Manage Configuration and from the list of products select <oXygen/> XML Editor and XSLT Debugger and XSLT Debugger.
2. Select *Disable*
3. Accept the restart of the Eclipse IDE.
4. Again choose the menu option: Help / Software Update / Manage Configuration and from the list of products select <oXygen/> XML Editor and XSLT Debugger .
5. Enable *Show Disabled Features* from the dialog toolbar.
6. From the right section of the displayed window choose *Uninstall.*
7. After the uninstall procedure is complete accept the Eclipse restart.
8. If you wish to completely remove the application directory and any work saved in it, you will have to delete this directory manually. To remove the application configuration and any personal customizations delete the `%APP-`

`DATA%\com.oxygenxml` directory on Windows (usually %APPDATA% has the value [user-home-dir]\Application Data) / `.com.oxygenxml` on Linux from the user home directory.

Performance problems

Large documents

If large documents (more than 10 MB) are edited in and you see that performance slows down considerably after some time then a possible cause is that it needs more memory in order to run properly. You can increase the maximum amount of memory available to plugin by specifying the parameters **-vmargs -Xmx** in the command used to launch the Eclipse platform.

Warning

The maximum amount of memory should not be equal to the physical amount of memory available on the machine because in that case the operating system and other applications will have no memory available.

External processes

The amount of memory allocated for generating PDF output with the built-in Apache FOP processor is controlled by a different setting available in Preferences: Memory available to the built-in FOP. In case of Out Of memory errors this is the setting that must be modified for allowing more memory for the built-in FOP.

For external XSL-FO processors configured in Preferences -> oXygen -> XML -> XSLT/FO/XQuery -> FO Processors and for external XSLT processors configured in Preferences -> oXygen -> XML -> XSLT/FO/XQuery -> Custom Engines the maximum memory must be set in the command line of the tool with a parameter -Xmx set to the Java virtual machine.

Chapter 3. Getting started

Supported types of documents

The <oXygen/> XML Editor provides a rich set of features for working with:

- XML documents and applications
- XSL stylesheets - transformations and debugging
- Schema languages: XML Schema, Relax NG (full and compact syntax), NRL, NVDL, Schematron, DTD
- Querying documents using XPath and XQuery
- Analyzing, composing and testing WSDL SOAP messages
- CSS documents

Getting help

Online help is available at any time while working in <oXygen/> by going to Help → Help Contents → oXygen User Manual for Eclipse

Perspectives

The interface uses standard interface conventions and components to provide a familiar and intuitive editing environment across all operating systems.

In you can work with documents in one of the perspectives:

Editor perspective	Editing of documents is supported by specialized and synchronized editors and views.
XSLT Debugger perspective	XSLT stylesheets can be debugged by tracing their execution step by step.
XQuery Debugger perspective	XQuery transforms can be debugged by tracing their execution step by step.
<oXygen/> Database perspective	Multiple connections to both relational databases and native XML ones can be managed at the same time in this perspective: database browsing, SQL execution, XQuery execution and data export to XML.

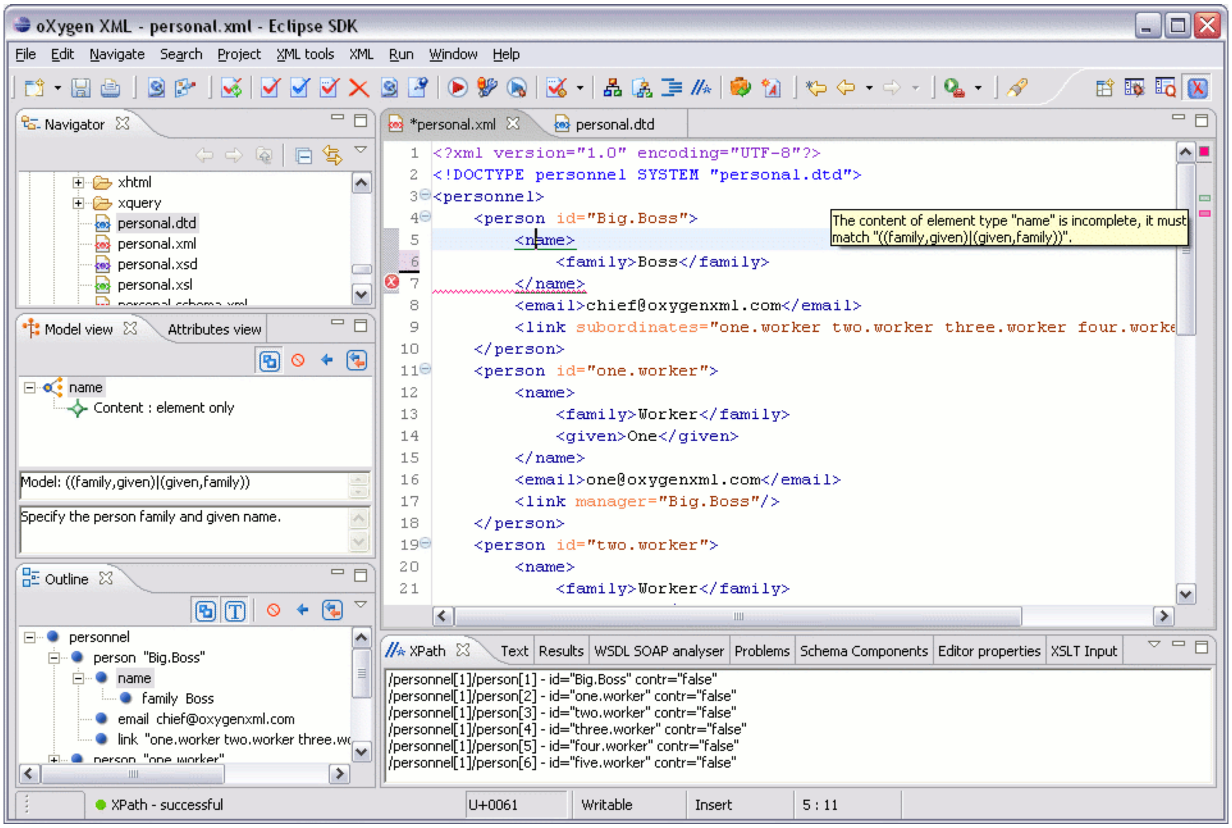
<oXygen/> XML perspective

The <oXygen/> XML perspective is used for editing the content of your documents.

As majority of the work process centers around the Editor panel, other panels can be hidden from view using the expand and collapse controls located on the divider bars.

This perspective organizes the workspace in the following panels:

Figure 3.1. <oXygen/> XML perspective



Model: ((family,given)|(given,family))

Specify the person family and given name.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE personnel SYSTEM "personal.dtd">
<personnel>
    <person id="Big.Boss">
        <name>
            <family>Boss</family>
        </name>
        <email>chief@oxygenxml.com</email>
        <link subordinates="one.worker two.worker three.worker four.worke
    </person>
    <person id="one.worker">
        <name>
            <family>Worker</family>
            <given>One</given>
        </name>
        <email>one@oxygenxml.com</email>
        <link manager="Big.Boss"/>
    </person>
    <person id="two.worker">
        <name>
            <family>Worker</family>
```

The content of element type "name" is incomplete, it must match "((family,given)|(given,family))".

```
/personnel[1]/person[1] - id="Big.Boss" contr="false"
/personnel[1]/person[2] - id="one.worker" contr="false"
/personnel[1]/person[3] - id="two.worker" contr="false"
/personnel[1]/person[4] - id="three.worker" contr="false"
/personnel[1]/person[5] - id="four.worker" contr="false"
/personnel[1]/person[6] - id="five.worker" contr="false"
```

The <oXygen/> custom menu

When the current editor window contains a document associated with <oXygen/> a custom menu is added to the Eclipse menu bar named after the document type: XML, XSL, XSD, RNG, RNC, Schematron, DTD, FO, WSDL, XQuery, HTML, CSS.

The <oXygen/> toolbar buttons

The toolbar buttons added by the <oXygen/> plugin provide easy access to common and frequently used functions. Each icon is a button that acts as a shortcut to a related function.

The editor pane

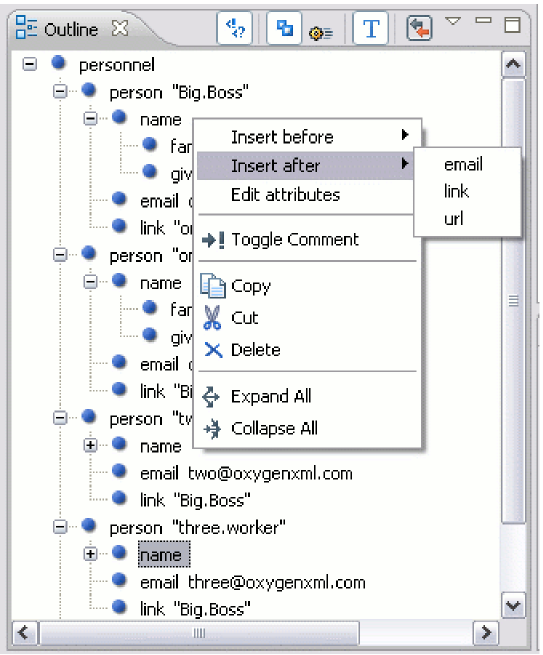
The editor pane is where you edit your documents opened or created by the <oXygen/> Eclipse plugin. You know the document is associated with <oXygen/> from the special icon displayed in the editor's title bar which has the same graphic pattern painted with different colors for different types of documents.

This pane has three different modes of displaying and editing the content of a document available as different tabs at the bottom left margin of the editor panel: text editor, grid editor, CSS-based tagless editor. Navigating between them is as easy as pressing Ctrl + Page Up for switching to the next tab to the left and Ctrl + Page Down for switching to the next tab to the right.

The Outline view

The outline view has the following functions: XML document overview, outliner filters, modification follow-up, document structure change, document tag selection.

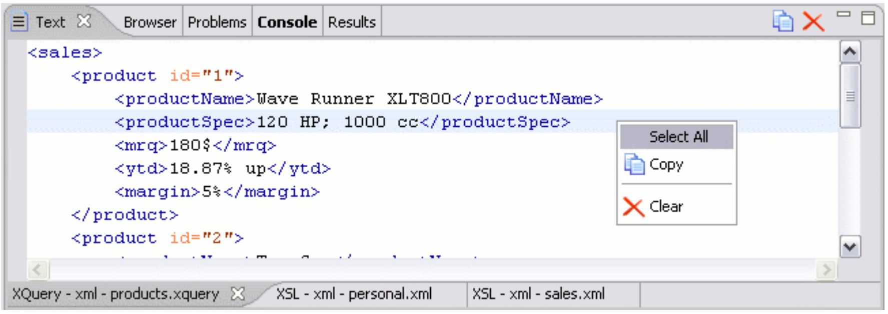
Figure 3.2. The Outline View



The <oXygen/> Text view

The <oXygen/> Text view is automatically showed in the views pane of the Eclipse window to display text output from XSLT transformations, FO processor's info, warning and error messages. It contains a tab for each file with text results displayed in the view.

Figure 3.3. The Text View

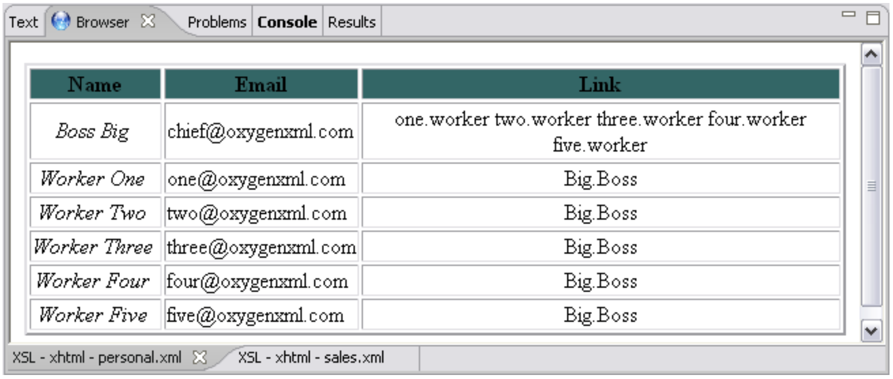


```
<sales>
    <product id="1">
        <productName>Wave Runner XLT800</productName>
        <productSpec>120 HP; 1000 cc</productSpec>
        <mrq>180$</mrq>
        <ytd>18.87% up</ytd>
        <margin>5%</margin>
    </product>
    <product id="2">
```

The <oXygen/> Browser view

The <oXygen/> Browser view is automatically showed in the views pane of the Eclipse window to display HTML output from XSLT transformations. It contains a tab for each file with HTML results displayed in the view.

Figure 3.4. The Browser View

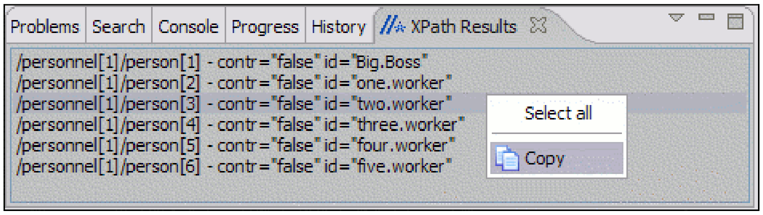


Name	Email	Link
Boss Big	chief@oxygenxml.com	one.worker two.worker three.worker four.worker five.worker
Worker One	one@oxygenxml.com	Big.Boss
Worker Two	two@oxygenxml.com	Big.Boss
Worker Three	three@oxygenxml.com	Big.Boss
Worker Four	four@oxygenxml.com	Big.Boss
Worker Five	five@oxygenxml.com	Big.Boss

The <oXygen/> XPath Results view

The <oXygen/> XPath Results view is automatically showed in the views pane of the Eclipse window to display XPath results.

Figure 3.5. The XPath Results View



```
/personnel[1]/person[1] - contr="false" id="Big.Boss"
/personnel[1]/person[2] - contr="false" id="one.worker"
/personnel[1]/person[3] - contr="false" id="two.worker"
/personnel[1]/person[4] - contr="false" id="three.worker"
/personnel[1]/person[5] - contr="false" id="four.worker"
/personnel[1]/person[6] - contr="false" id="five.worker"
```

Supported editor types

The <oXygen/> Eclipse plugin provides special Eclipse editors identified by the following icons:

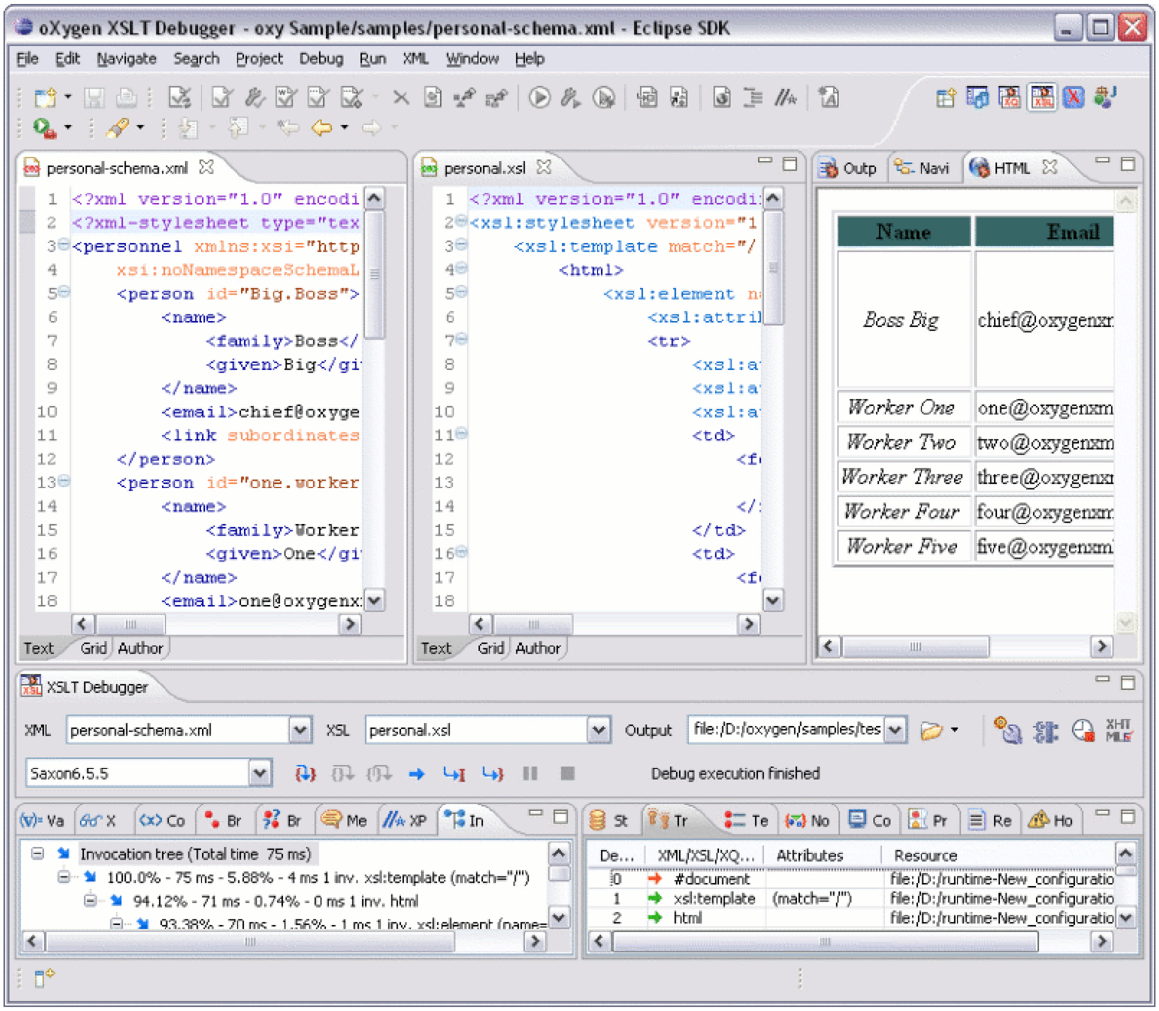
- - The icon for XML documents
- - The icon for XSL stylesheets
- - The icon for XML Schema
- - The icon for Document Type Definition schemas
- - The icon for RELAX NG full syntax schemas
- - The icon for RELAX NG compact syntax schemas
- - The icon for Namespace Routing Language/ Namespace-based Validation Dispatching Language schemas
- - The icon for XSL:FO documents
- - The icon for XQuery documents
- - The icon for WSDL documents

- - The icon for Schematron documents
- - The icon for JavaScript documents
- - The icon for Python documents
- - The icon for CSS documents

<oXygen/> XSLT Debugger Perspective

The XSLT Debugger perspective is used for detecting problems in an XSLT transformation process by executing the process step by step in a controlled environment and inspecting the information provided in different special views. The workspace is organized as an editing area supported by special helper views. The editing area contains editor panels and can be split horizontally or vertically in two stacks of editors: XML editor panels and XSLT editor panels.

Figure 3.6. <oXygen/> XSLT Debugger perspective


oXygen XSLT Debugger - oxy Sample/samples/personal-schema.xml - Eclipse SDK
File Edit Navigate Search Project Debug Run XML Window Help
personal-schema.xml
personal.xsl
Outp Navi HTML
Text Grid Author
XSLT Debugger
Va X Co Br Br Me XP In
St Tr Te No Co Pr Re Ho

personal-schema.xml:

```
<?xml version="1.0" encodi
<?xml-stylesheet type="tex
<personnel xmlns:xsi="http
    xsi:noNamespaceSchemaL
    <person id="Big.Boss">
        <name>
            <family>Boss</
            <given>Big</gi
        </name>
        <email>chief@oxyge
        <link subordinates
    </person>
    <person id="one.worker
        <name>
            <family>Worker
            <given>One</gi
        </name>
        <email>one@oxygenx
```

personal.xsl:

```
<?xml version="1.0" encodi
<xsl:stylesheet version="1
    <xsl:template match="/
        <html>
            <xsl:element n
                <xsl:attri
                <tr>
                    <xsl:a
                    <xsl:a
                    <xsl:a
                    <td>
                        <f

                        </
                    </td>
                    <td>
                        <f

```

Name	Email
Boss Big	chief@oxygenxr
Worker One	one@oxygenxm
Worker Two	two@oxygenxm
Worker Three	three@oxygenx
Worker Four	four@oxygenxm
Worker Five	five@oxygenxml

XML: personal-schema.xml XSL: personal.xsl Output: file:/D:/oxygen/samples/tes

Saxon6.5.5

Debug execution finished

```
Invocation tree (Total time  75 ms)
  100.0% - 75 ms - 5.88% - 4 ms 1 inv. xsl:template (match="/")
    94.12% - 71 ms - 0.74% - 0 ms 1 inv. html
      93.38% - 70 ms - 1.56% - 1 ms 1 inv. xsl:element (name=
```

De...	XML/XSL/XQ...	Attributes	Resource
0	#document		file:/D:/runtime-New_configuratio
1	xsl:template	(match="/")	file:/D:/runtime-New_configuratio
2	html		file:/D:/runtime-New_configuratio

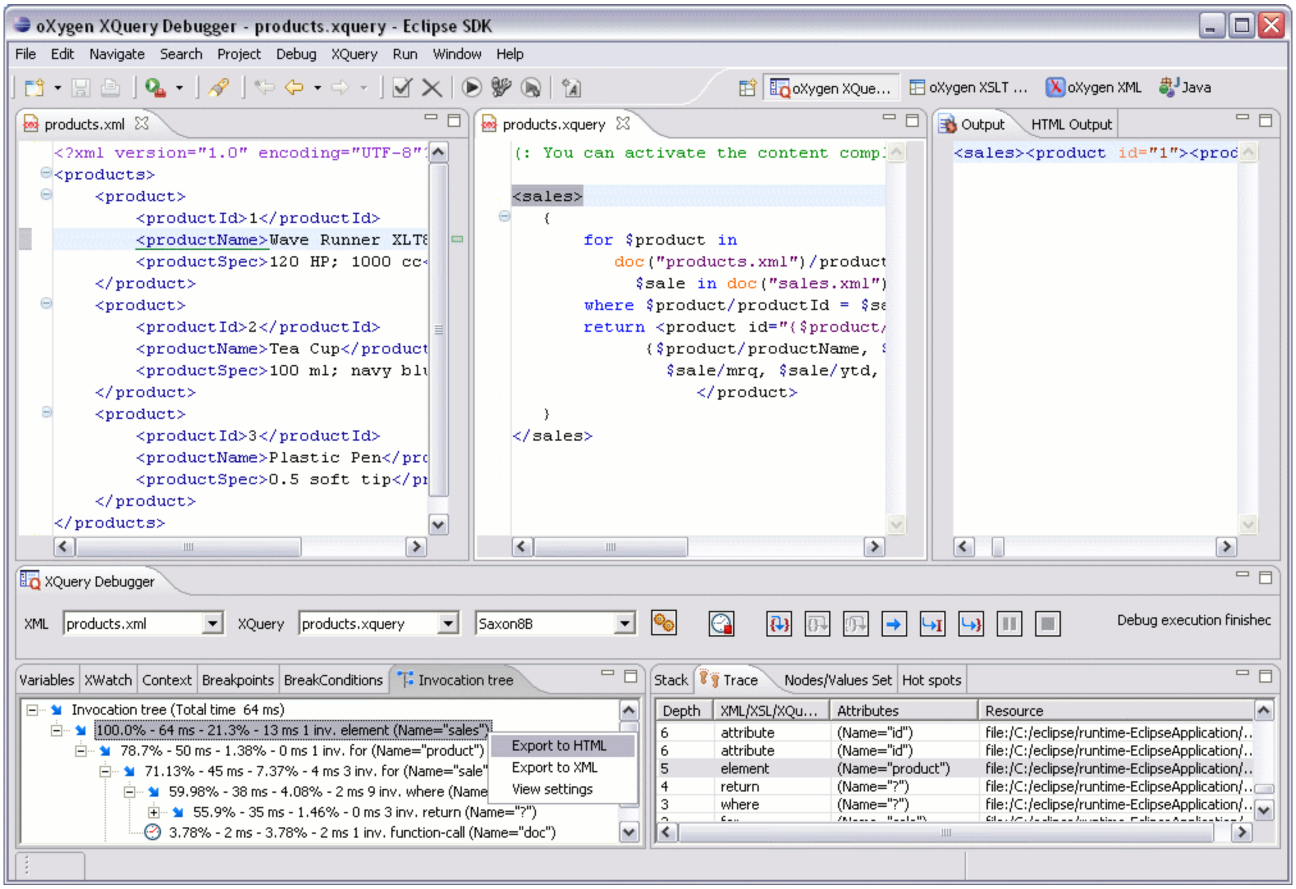
- Source document view - Displays and allows editing of data or document oriented XML files (documents).
- Stylesheet document view - Displays and allows editing of XSL files(stylesheets).

- Output document view - Displays the transformed output that results from the input of a selected document (XML) and selected stylesheet (XSL) to the transformer. The result of transformation is dynamically written as the transformation is processed. There are three types of views for the output: a text view (with XML syntax highlight), an XHTML view and one text view for each xsl:result-document element used in the stylesheet (if it is a XSLT 2.0 stylesheet).
- Control toolbar - Contains all actions needed in order to configure and control the debug process.
- Information views - Distributed in two panes that are used to display various types of information that can be used to understand the transformation process. For each information type there is a corresponding tab. While running a transformation, relevant events are displayed in the various information views. This enables the developer to obtain a clear view of the transformation progress.

<oXygen/> XQuery Debugger Perspective

The XQuery Debugger perspective is similar to the XSLT Debugger perspective. It is used for detecting problems in an XQuery transformation process by executing the process step by step in a controlled environment and inspecting the information provided in different special views. The workspace is organized in:

Figure 3.7. <oXygen/> XQuery Debugger perspective



```
<?xml version="1.0" encoding="UTF-8"?
<products>
    <product>
        <productId>1</productId>
        <productName>Wave Runner XLT8
        <productSpec>120 HP; 1000 cc<
    </product>
    <product>
        <productId>2</productId>
        <productName>Tea Cup</product
        <productSpec>100 ml; navy blu
    </product>
    <product>
        <productId>3</productId>
        <productName>Plastic Pen</prc
        <productSpec>0.5 soft tip</pr
    </product>
</products>
```

```
(: You can activate the content compl

<sales>
   {
        for $product in
           doc("products.xml")/product
             $sale in doc("sales.xml")
        where $product/productId = $sa
        return <product id="{$product/
              {$product/productName, $
                $sale/mrq, $sale/ytd,
                   </product>
   }
</sales>
```

```
<sales><product id="1"><proc
```

XML products.xml XQuery products.xquery Saxon8B

Debug execution finishec

```
Invocation tree (Total time  64 ms)
  100.0% - 64 ms - 21.3% - 13 ms 1 inv. element (Name="sales")
    78.7% - 50 ms - 1.38% - 0 ms 1 inv. for (Name="product")
      71.13% - 45 ms - 7.37% - 4 ms 3 inv. for (Name="sale"
        59.98% - 38 ms - 4.08% - 2 ms 9 inv. where (Name
          55.9% - 35 ms - 1.46% - 0 ms 3 inv. return (Name="?")
        3.78% - 2 ms - 3.78% - 2 ms 1 inv. function-call (Name="doc")
```

Depth	XML/XSL/XQu...	Attributes	Resource
6	attribute	(Name="id")	file:/C:/eclipse/runtime-EclipseApplication/..
6	attribute	(Name="id")	file:/C:/eclipse/runtime-EclipseApplication/..
5	element	(Name="product")	file:/C:/eclipse/runtime-EclipseApplication/..
4	return	(Name="?")	file:/C:/eclipse/runtime-EclipseApplication/..
3	where	(Name="?")	file:/C:/eclipse/runtime-EclipseApplication/..

- Source document view - Displays and allows editing of data or document oriented XML files (documents).
- XQuery document view - Displays and allows editing of XQuery files.

- Output document view - Displays the transformed output that results from the input of a selected document (XML) and selected XQuery document to the XQuery transformer. The result of transformation is dynamically written as the transformation is processed. There are two types of views for the output: a text view (with XML syntax highlight) and an XHTML view.
- Control toolbar - Contains all actions needed in order to configure and control the debug process.
- Information views - Distributed in two panes that are used to display various types of information that can be used to understand the transformation process. For each information type there is a corresponding tab. While running a transformation, relevant events are displayed in the various information views. This enables the developer to obtain a clear view of the transformation progress.

<oXygen/> Database perspective

The Database perspective is similar to the Editor perspective. It allows you to manage a database, offering support for browsing multiple connections at the same time, both relational and native XML databases, SQL execution, XQuery execution and data export to XML.

This perspective offers database specific support for:

- Sleepycat Berkeley DB XML Database
- eXist XML Database
- IBM DB2 (Enterprise edition only)
- JDBC-ODBC Bridge (Enterprise edition only)
- MarkLogic (Enterprise edition only, XQuery support only)
- Microsoft SQL Server 2005 and Microsoft SQL Server 2008 (Enterprise edition only)
- MySQL (Enterprise edition only)
- Oracle 11g (Enterprise edition only)
- PostgreSQL 8.3 (Enterprise edition only)
- Software AG Tamino (Enterprise edition only)
- TigerLogic (Enterprise edition only, XQuery support only)
- Documentum xDb (X-Hive/DB) XML Database (Enterprise edition only)

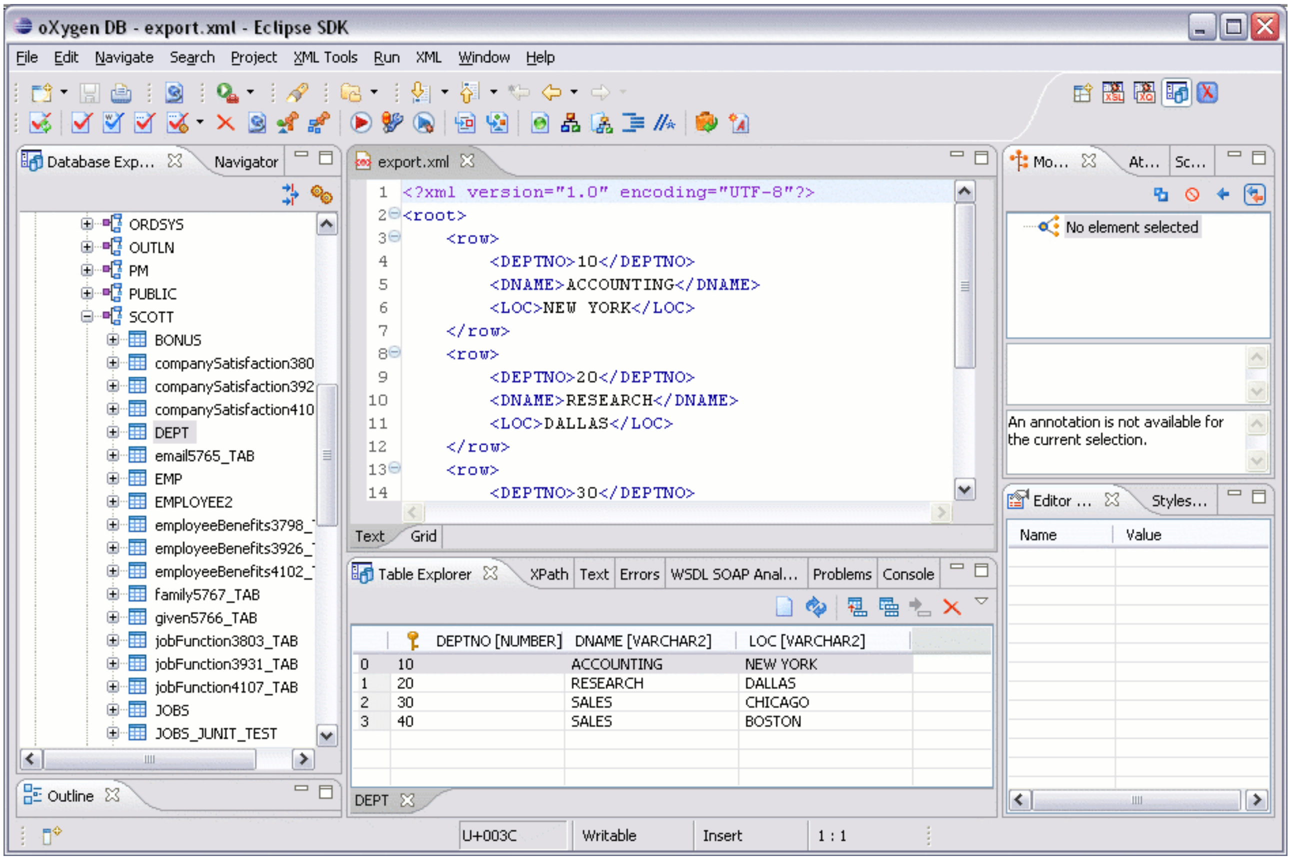
The XML capabilities of the databases marked in this list with "Enterprise edition only" are available only in the Enterprise edition of <oXygen/>. The non-XML capabilities of any database listed here are available also in the Academic and Professional editions of <oXygen/> by registering the database driver as a generic JDBC driver (the *Generic JDBC* type in the list of driver types) when defining the data source for accessing the database in <oXygen/>. The non-XML capabilities are browsing the structure of the database instance, opening a table in the *Table Explorer* view, handling the values from columns of type XML Type as String values. The XML capabilities are: displaying an XML Schema node in the tree of the database structure (for databases with such an XML specific structure) with actions for opening/editing/validating the schemas in an <oXygen/> editor panel, handling the values from columns of type XML Type as XML instance documents that can be opened and edited in an <oXygen/> editor panel, validating an XML instance document added to an XML Type column of a table, etc.

For a detailed feature matrix that compares the Academic, Professional and Enterprise editions of <oXygen/> please go to the <oXygen/> website [http://www.oxygenxml.com/feature_matrix.html].

Note

Only connections configured on relational data sources can be used to import to XML or to generate XML schemas.

Figure 3.8. Database perspective



```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <row>
        <DEPTNO>10</DEPTNO>
        <DNAME>ACCOUNTING</DNAME>
        <LOC>NEW YORK</LOC>
    </row>
    <row>
        <DEPTNO>20</DEPTNO>
        <DNAME>RESEARCH</DNAME>
        <LOC>DALLAS</LOC>
    </row>
    <row>
        <DEPTNO>30</DEPTNO>
```

	DEPTNO [NUMBER]	DNAME [VARCHAR2]	LOC [VARCHAR2]
0	10	ACCOUNTING	NEW YORK
1	20	RESEARCH	DALLAS
2	30	SALES	CHICAGO
3	40	SALES	BOSTON

No element selected

An annotation is not available for the current selection.

Main menu — Provides menu driven access to all the features and functions available within <oXygen/>.

Main toolbar — Provides easy access to common and frequently used functions. Each icon is a button that acts as a shortcut to a related function.

Editor panel — The place where you spend most of your time, reading, editing, applying markup and checking the validity and form of your documents.

Database explorer — Provides browsing support for the configured connections.

Table explorer — Provides table content editing support: insert a new row, delete a table row, cell value editing, export to XML file.

Chapter 4. Editing documents

Working with Unicode

Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. Unicode is an internationally recognized standard, adopted by industry leaders. The Unicode is required by modern standards such as XML, Java, ECMAScript (JavaScript), LDAP, CORBA 3.0, WML, etc., and is the official way to implement ISO/IEC 10646.

It is supported in many operating systems, all modern browsers, and many other products. The emergence of the Unicode Standard, and the availability of tools supporting it, are among the most significant recent global software technology trends. Incorporating Unicode into client-server or multi-tiered applications and websites offers significant cost savings over the use of legacy character sets.

As a modern XML Editor, <oXygen/> provides support for the Unicode standard enabling your XML application to be targeted across multiple platforms, languages and countries without re-engineering. Internally, the <oXygen/> XML Editor uses 16bit characters covering the Unicode Character set.

Opening and saving Unicode documents

On loading documents of the type XML, XSL, XSD and DTD,<oXygen/> receives the encoding of the document from the Eclipse platform. This is then used to instruct the Java Encoder to load support for and save using the code chart specified.

While in most cases you will use UTF-8, simply changing the encoding name will cause the file to be saved using the new encoding. The appendix Unicode Character Encoding provides a matrix that matches common names with Java Names. It also explains what you should type in the XML prolog to cause the document to be saved as the required encoding.

To edit document written in Japanese or Chinese, you will need to change the font to one that supports the specific characters (a Unicode font). For the Windows platform, use of *Arial Unicode MS* or *MS Gothic* is recommended. Do not expect Wordpad or Notepad to handle these encodings. Use Internet Explorer or Word to eventually examine XML documents.

When a document with a UTF-16 encoding is edited and saved in <oXygen/>, the saved document will have a byte order mark (BOM) which will specify the byte order of the document's content. The default byte order is platform dependent. That means that a UTF-16 document created on a Windows platform (where the default byte order mark is *UnicodeLittle*) will have a different BOM than a UTF-16 document created on a Mac OS platform (where the byte order mark is *UnicodeBig*). The byte order and the BOM of an existing document will be preserved by <oXygen/> when the document is edited and saved.

Note

The naming convention used under Java does not always correspond to the common names used by the Unicode standard. For instance, while in XML you will use encoding="UTF-8", in Java the same encoding has the name "UTF8".

Opening and closing documents

As with most editing applications, <oXygen/> lets you open existing documents, save your changes and close them as required.

Creating new documents

<oXygen/> plugin wizards

The New wizard only creates a skeleton document containing the document prolog, a root element and possibly other child elements depending on the options specific for each schema type. If you need to generate full and valid XML instance documents based on an XML Schema schema you should use the XML instance generation tool instead.

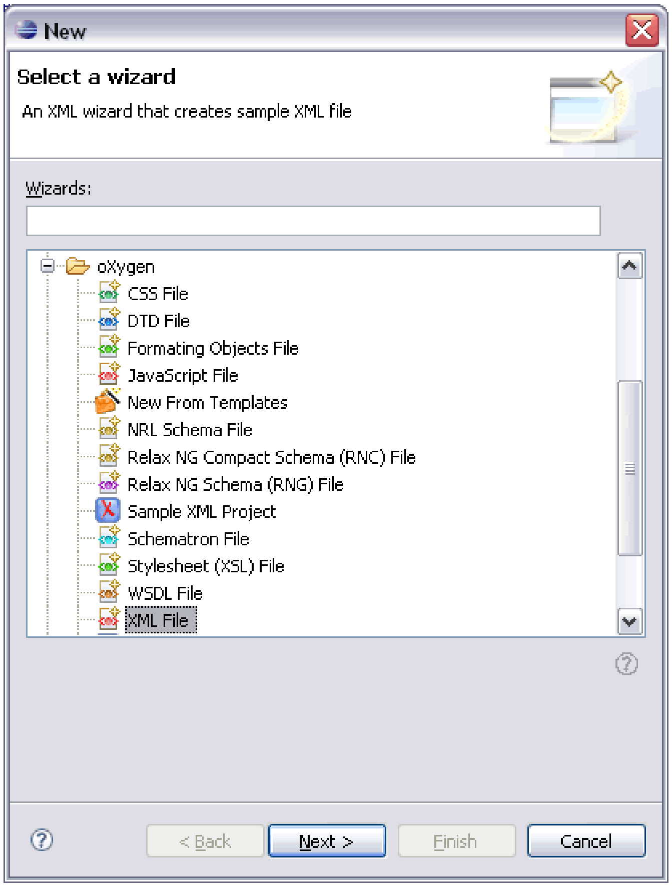
Use the following procedure to create documents.

The <oXygen/> plugin installs a series of Eclipse wizards for easy creation of new documents. Using these wizards you let <oXygen/> fill in details like the system ID or schema location of a new XML document, the minimal markup of a DocBook article or the namespace declarations of a Relax NG schema.

Procedure 4.1. Creating new documents

1. Select File → New → -> Other (**Ctrl+N**) or press the New toolbar button. The New wizard is displayed which contains the supported Document Types: XML, XSL, XML Schema, Document Type Definition, Relax NG Schema, XQuery, Web Services Definition Language, Schematron Schema, CSS File.

   Figure 4.1. The New wizard

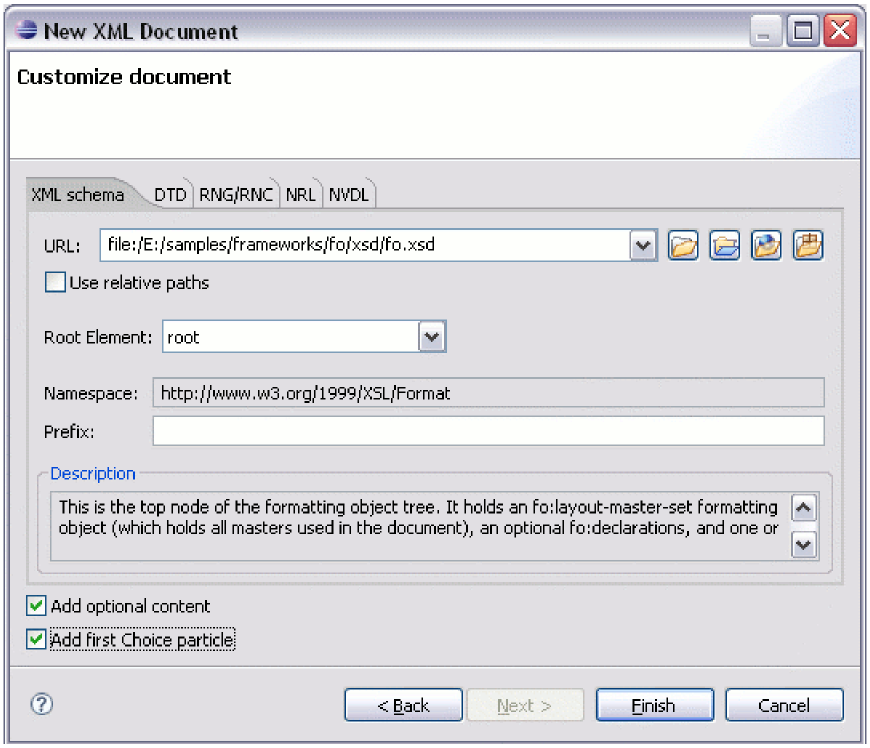
   

2. Select a document type, then click Next. For example if XML was selected the "Create an XML Document" wizard is started.

3. Type a name for the new document and press Next.

4. The Create an XML Document dialog enables definition of a XML Document Prolog using the system identifier of a XML Schema, DTD, Relax NG (full or compact syntax), NRL (Namespace Routing Language) or NVDL (Namespace-based Validation Dispatching Language) schema. As not all XML documents are required to have a Prolog, you may choose to skip this step by clicking OK . If the prolog is required complete the fields as the following.

Figure 4.2. The Create an XML Document Dialog - XML Schema Tab



New XML Document

Customize document

URL: file:/E:/samples/frameworks/fo/xsd/fo.xsd

☐ Use relative paths

Root Element: root

Namespace: http://www.w3.org/1999/XSL/Format

Prefix:

Description

This is the top node of the formatting object tree. It holds an fo:layout-master-set formatting object (which holds all masters used in the document), an optional fo:declarations, and one or

☑ Add optional content

☑ Add first Choice particle

Complete the dialog as follows:

URL

Specifies the location of an XML Schema Document (XSD).

You can also specify an URI if it is solved by the <oXygen/> catalog.

Example 4.1. DITA XSD URI

```
urn:oasis:names:tc:dita:xsd:topic.xsd:1.1
```

Document Root

Populated from the elements defined in the specified XSD, enables selection of the element to be used as document root.

Namespace

Specifies the document namespace.

Prefix

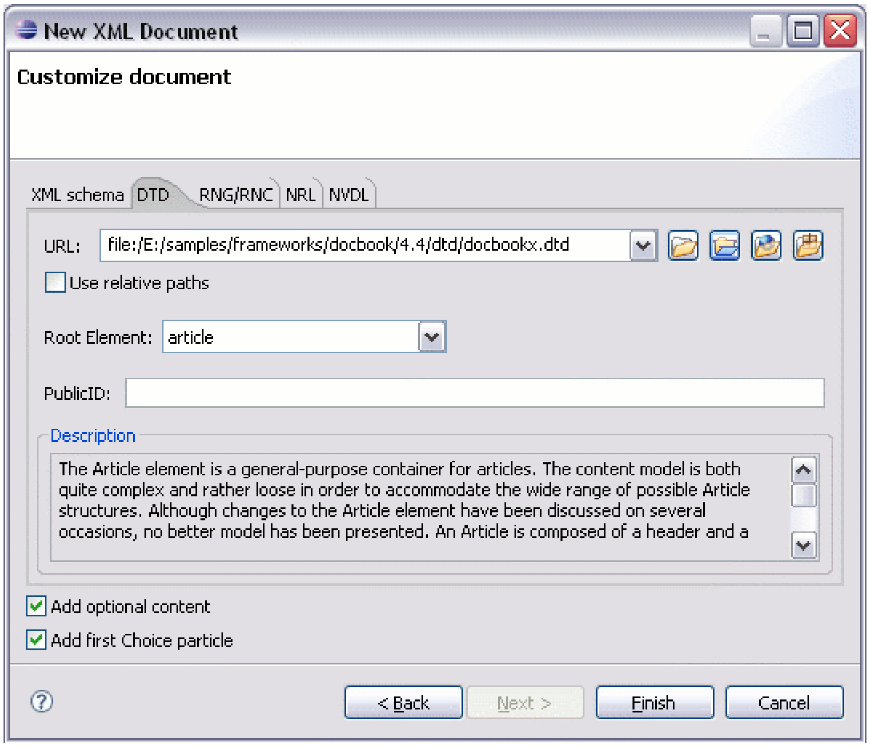
Specifies the prefix for the namespace of the document root.

Description — Shows a small definition for the currently selected element.

Add optional content — If it is selected the elements and attributes that are defined in the XML Schema as optional are generated in the skeleton XML document created in a new editor panel when the OK button is pressed.

Add first Choice particle — If it is selected the first element of an *xs:choice* schema element is generated in the skeleton XML document created in a new editor panel when the OK button is pressed.

Figure 4.3. The Create an XML Document - DTD Tab



URL: file:/E:/samples/frameworks/docbook/4.4/dtd/docbookx.dtd

☐ Use relative paths

Root Element: article

PublicID:

Description

The Article element is a general-purpose container for articles. The content model is both quite complex and rather loose in order to accommodate the wide range of possible Article structures. Although changes to the Article element have been discussed on several occasions, no better model has been presented. An Article is composed of a header and a

☑ Add optional content

☑ Add first Choice particle

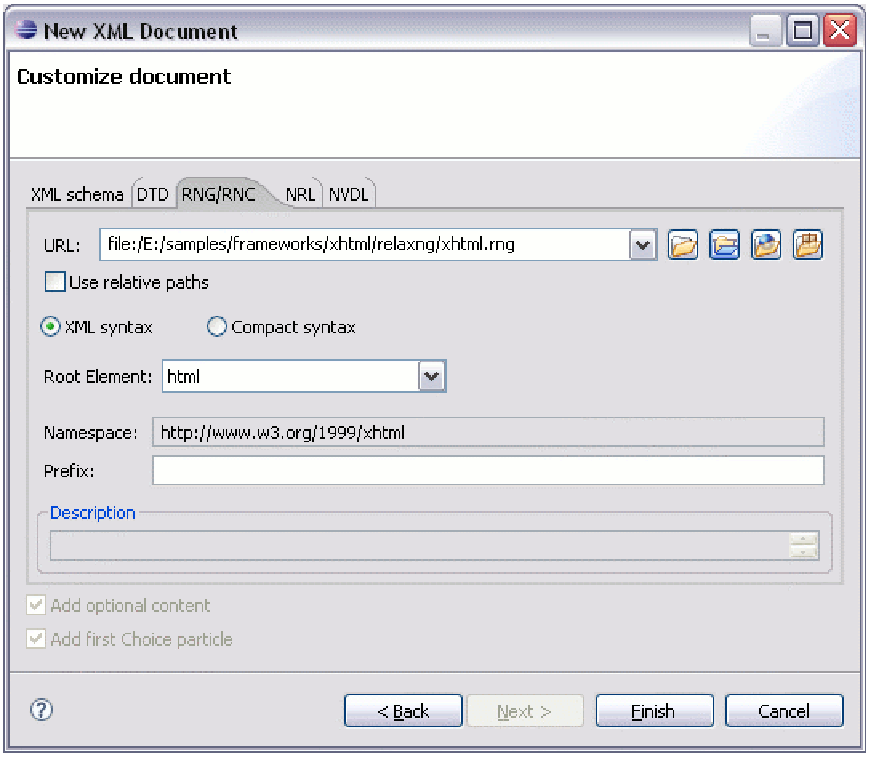
Complete the dialog as follows:

System ID — Specifies the location of a Document Type Definition (DTD).

Document Root — Populated from the elements defined in the specified DTD, enables selection of the element to be used as document root.

Public ID — Specifies the PUBLIC identifier declared in the Prolog.

Figure 4.4. The Create an XML Document - Relax NG Tab



New XML Document

Customize document

URL: file:/E:/samples/frameworks/xhtml/relaxng/xhtml.rng

□ Use relative paths

● XML syntax ○ Compact syntax

Root Element: html

Namespace: http://www.w3.org/1999/xhtml

Prefix:

Description

☑ Add optional content

☑ Add first Choice particle

Complete the dialog as follows:

URL — Specifies the location of a Relax NG schema in XML or compact syntax (RNG/RNC).

XML syntax — When checked the specified URL refers to a Relax NG schema in XML syntax. It will be checked automatically if the user selects a document with the *.rng* extension.

Compact syntax — When checked the specified URL refers to a Relax NG schema in compact syntax. It will be checked automatically if the user selects a document with the *.rnc* extension.

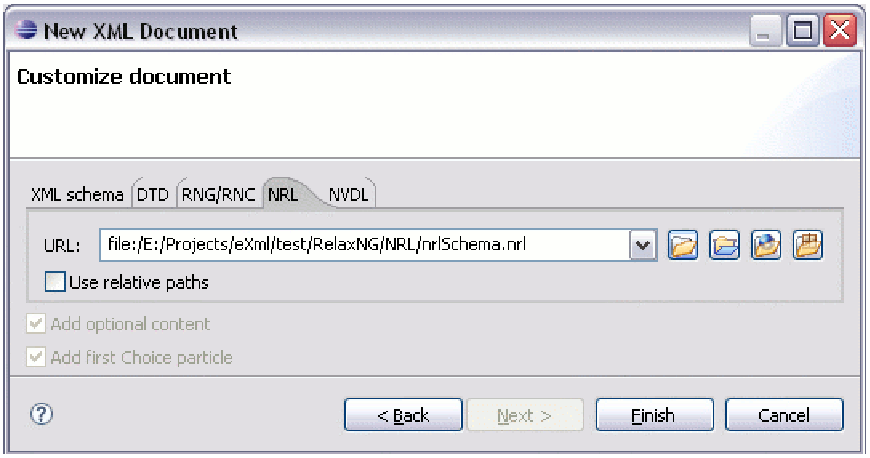
Document Root — Populated from the elements defined in the specified RNG or RNC document, enables selection of the element to be used as document root.

Namespace — Specifies the document namespace.

Prefix — Specifies the prefix for the namespace of the document root.

Description — Shows a small definition for the currently selected element.

Figure 4.5. The Create an XML Document - NRL Tab



Customize document

URL: file:/E:/Projects/eXml/test/RelaxNG/NRL/nrlSchema.nrl

□ Use relative paths

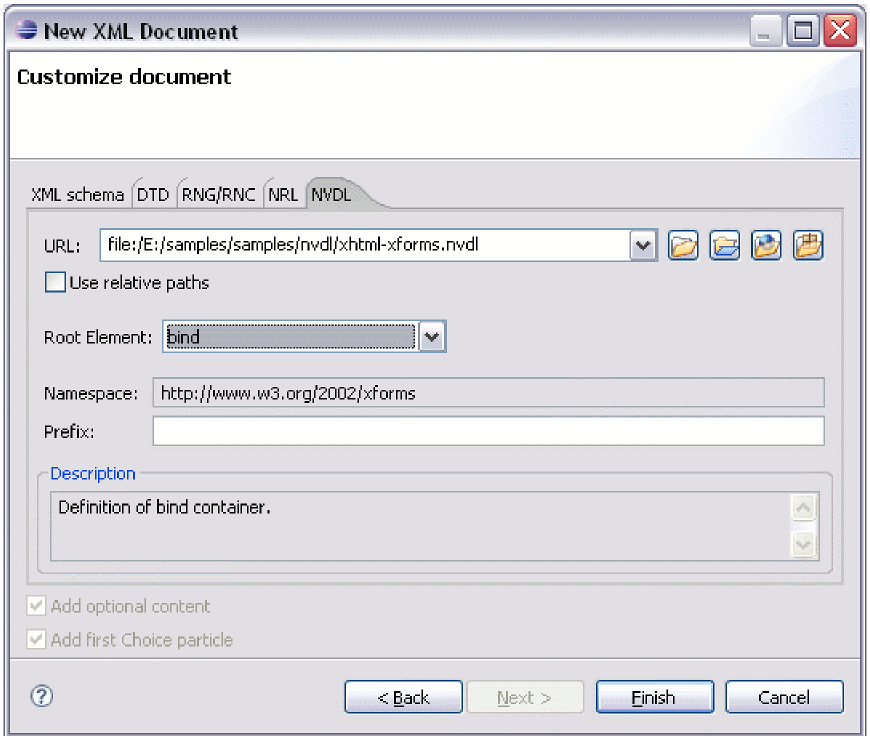
☑ Add optional content

☑ Add first Choice particle

Complete the dialog as follows:

URL Specifies the location of a NRL schema (NRL).

Figure 4.6. The Create an XML Document - NVDL Tab



Customize document

URL: file:/E:/samples/samples/nvdl/xhtml-xforms.nvdl

□ Use relative paths

Root Element: bind

Namespace: http://www.w3.org/2002/xforms

Prefix:

Description

Definition of bind container.

☑ Add optional content

☑ Add first Choice particle

Complete the dialog as follows:

URL Specifies the location of a Namespace-based Validation Dispatching Language schema (NVDL).

Document Root Populated from the elements defined in the specified NVDL document, enables selection of the element to be used as document root.

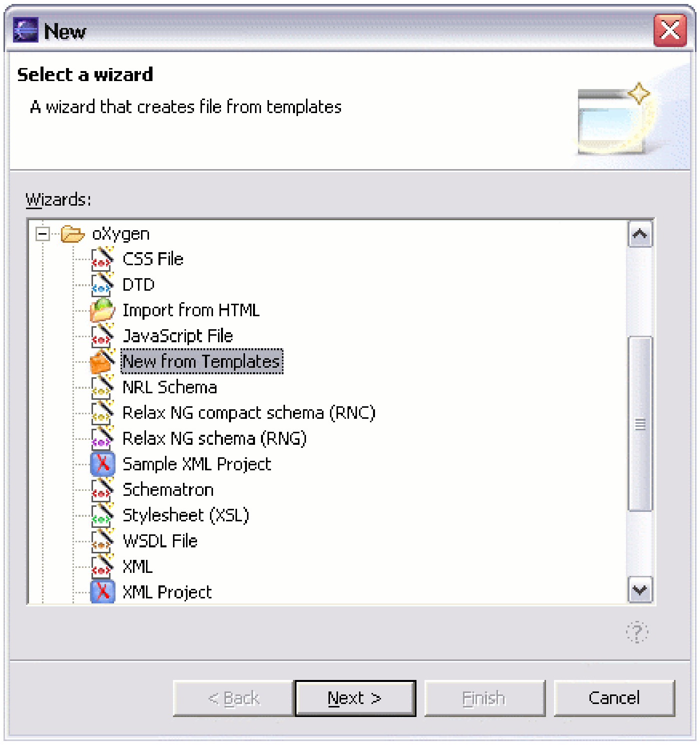
Namespace	Specifies the document namespace.
Prefix	Specifies the prefix for the namespace of the document root.
Description	Shows a small definition for the currently selected element.

Creating Documents based on Templates

Templates are documents containing a predefined structure. They provide starting points on which one can rapidly build new documents that repeat the same basic characteristics. <oXygen/> installs a rich set of templates for a number of XML applications. You may also create your own templates and share them with other users.

The New from Templates wizard enables you to select predefined templates or templates that have already been created in previous sessions or by other users. Open a template using the following options:

Figure 4.7. The New from Templates wizard



New

Select a wizard

A wizard that creates file from templates

Wizards:

- oXygen
  - CSS File
  - DTD
  - Import from HTML
  - JavaScript File
  - New from Templates
  - NRL Schema
  - Relax NG compact schema (RNC)
  - Relax NG schema (RNG)
  - Sample XML Project
  - Schematron
  - Stylesheet (XSL)
  - WSDL File
  - XML
  - XML Project

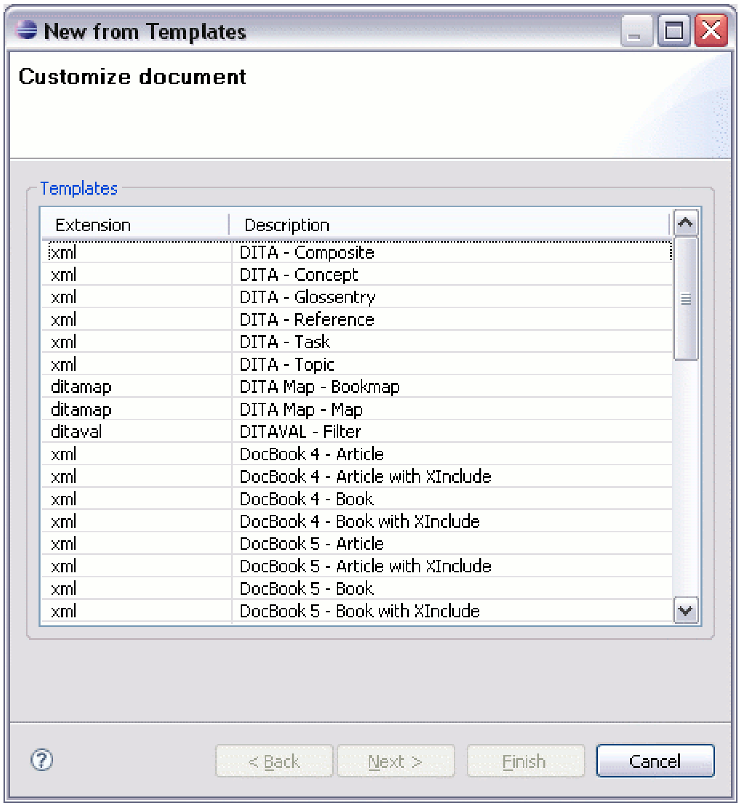
The templates presented in the dialog are:

Document Types templates	Templates supplied with the defined document types.
User defined templates	The user can add template files in the `templates` folder of the <oXygen/> install directory. Also in the option page can be specified a custom templates folder to be scanned.

Procedure 4.2. Creating Documents based on Templates

1. Select File → New → New from Templates The New from templates dialog is displayed.

Figure 4.8. The Templates dialog



Extension	Description
xml	DITA - Composite
xml	DITA - Concept
xml	DITA - Glossentry
xml	DITA - Reference
xml	DITA - Task
xml	DITA - Topic
ditamap	DITA Map - Bookmap
ditamap	DITA Map - Map
ditaval	DITAVAL - Filter
xml	DocBook 4 - Article
xml	DocBook 4 - Article with XInclude
xml	DocBook 4 - Book
xml	DocBook 4 - Book with XInclude
xml	DocBook 5 - Article
xml	DocBook 5 - Article with XInclude
xml	DocBook 5 - Book
xml	DocBook 5 - Book with XInclude

2. Scroll the Templates list and select the required Template Type.

3. Type a name for the new document and press Next.

4. Click Finish. A new document is opened that already contains structure and content provided in the template starting point.

Saving documents

The edited document can be saved with one of the actions:

- File → Save (**Ctrl+S**) to save the current document.

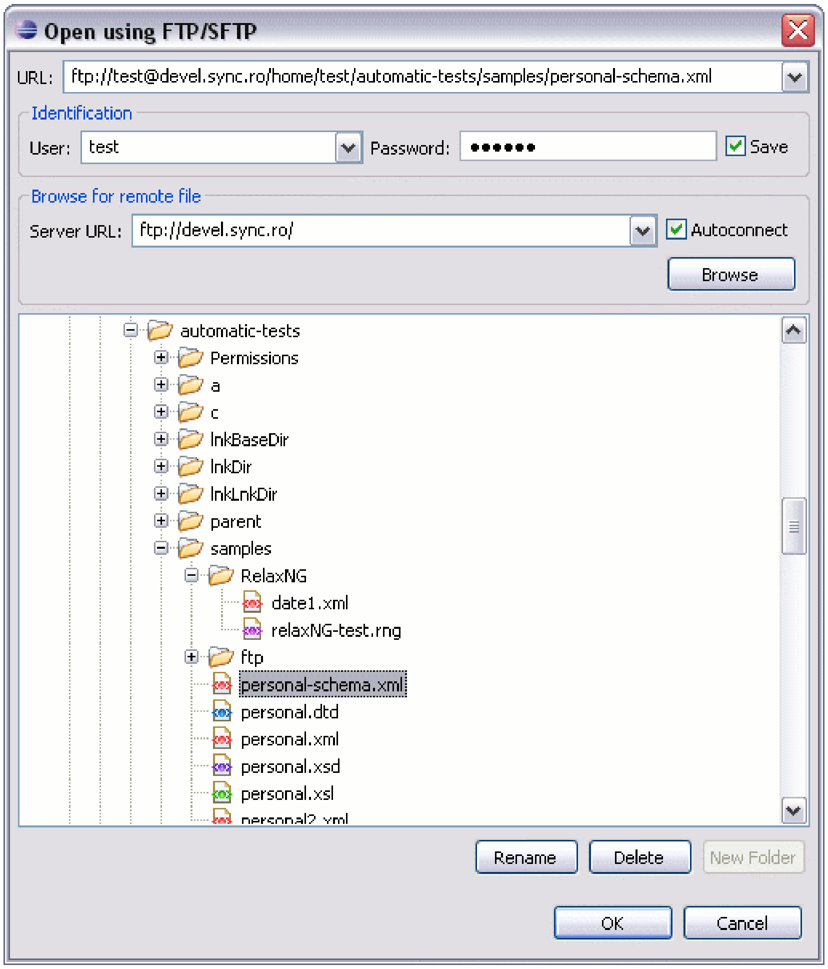
- File → Save As: Displays the Save As dialog, used to name and save an open document to a file; or save an existing file with a new name.

- File → Save All: Saves all open documents.

Opening and Saving Remote Documents via FTP/SFTP

<oXygen/> supports editing remote files, using the FTP, SFTP protocols. The remote opened files can be edited exactly as the local ones. They can be added to the project, and can be subject to XSL and FO transformations.

Figure 4.9. Open URL dialog



Open using FTP/SFTP

URL: ftp://test@devel.sync.ro/home/test/automatic-tests/samples/personal-schema.xml

Identification

User: test Password: •••••• ☑ Save

Browse for remote file

Server URL: ftp://devel.sync.ro/ ☑ Autoconnect

Note

The FTP part is using passive access to the FTP servers. Make sure the server you are trying to connect to is supporting passive connections. Also the UTF-8 encoding is supported and can be configured for communication with the FTP server if the server supports it.

Files can be opened through the Secure FTP (SFTP) protocol using the regular user/password mechanism or using a private key file and a passphrase. The user/password mechanism has precedence so for using the private key and passphrase you have to remove the password from the dialog used to browse the server repository and leave only the user name. The private key file and the passphrase must be set in the SFTP user preferences.

Note

WebDAV access is available only if you check the *Enable the HTTP/WebDAV protocols* option from Window → Preferences+oXygen / Network Configuration page. The proxy settings set in Window → Preferences+General / Network Connections are valid also in the <oXygen/> plugin, that is if an HTTP proxy server, a SOCKS proxy server or a list with excepted host names is set there any HTTP / WebDAV connection made with the <oXygen/> plugin will take into consideration these settings.

To open the remote files, choose from the main menu File → Open URL ... The displayed dialog is composed of several parts:

- The editable combo box, in which it can be specified directly the URL to be opened or saved.

  ### URLs that can be directly opened

  You can type in here an URL like ftp://anonymous@some.site/home/test.xml if the file is accessible through anonymous FTP.

  This combo box is also displaying the current selection when the user changes selection by browsing the tree of folders and files on the server.

- The *Identification* section contains the access credentials. If you want to browse for a file on a server, you have to specify the user and password. This information is bound to the selected URL displayed in the "File URL" combo box, and used further in opening/saving the file. If the check box "Save" is selected, then the user and password are saved between editing sessions. The password is kept encrypted into the options file.

  ### Note

  Your password is well protected. In the case the options file is used on other machine by a user with a different username the password will become unreadable, since the encryption is username dependent. This is also true if you add URLs having user and password to your project.

- The *Browse for remote file* section contains the server combo and the "Autoconnect" check box. Into the server combo it may be specified the protocol , the name or IP of the server .

  ### Server URLs

  When accessing a FTP server, you need to specify only the protocol and the host, like: ftp://server.com, ftp://ftp.apache.org, or if using a nonstandard port: ftp://server.com:7800/ etc.

  By pressing the "Browse" button the directory listing will be shown in the component below. When "Autoconnect" is selected then at every time the dialog is shown, the browse action will be performed.

- The tree view of the documents stored on the server. You can browse the directories, and make multiple selections. Additionally, you may use the "Rename", "Delete", and "New Folder" to manage the file repository.

  The file names are sorted in a case-insensitive way.

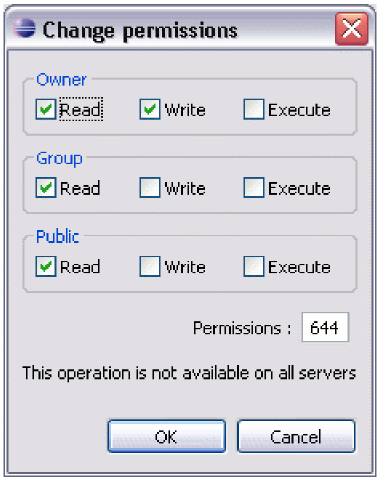
A WebDAV resource can be locked when it is opened in <oXygen/> by checking the option Lock WebDAV files on open to protect it from concurrent modifications on the server by other users. If other user tries to edit the same resource he will receive an error message and the name of the lock owner. The lock is released automatically when the editor for that resource is closed in <oXygen/>.

GZIP compression is handled correctly for the content received/sent from/to a HTTP server or a WebDAV server. The built-in client of <oXygen/> notifies the server when the connection is established that GZIP compression is supported.

Changing file permissions on a remote FTP server

Some FTP servers allow the modification of file permissions on the file system for the files that they serve over the FTP protocol. This feature of the protocol is accessible directly in the FTP file browser dialog by right-clicking on a tree node and selecting the *Change permissions* menu item which brings up the following dialog:

Figure 4.10. FTP server - change file permissions



Change permissions

Owner

☑ Read ☑ Write □ Execute

Group

☑ Read □ Write □ Execute

Public

☑ Read □ Write □ Execute

Permissions : 644

This operation is not available on all servers

The usual Unix file permissions *Read*, *Write* and *Execute* are granted or denied in this dialog for the owner of the file, the group of the owner and the rest of the users. The aggregate number of the current state of the permissions is updated in the *Permissions* text field when a permission is modified with one of the check boxes.

Opening the current document in a Web browser

To open the current document in the default Web browser installed on the computer use the action *Open in browser* available on menu XML and also on the *Document* toolbar. It is useful for seeing the effect of applying an XSLT stylesheet or a CSS stylesheet on a document which specifies the stylesheet using an *xml-stylesheet* processing instruction.

Closing documents

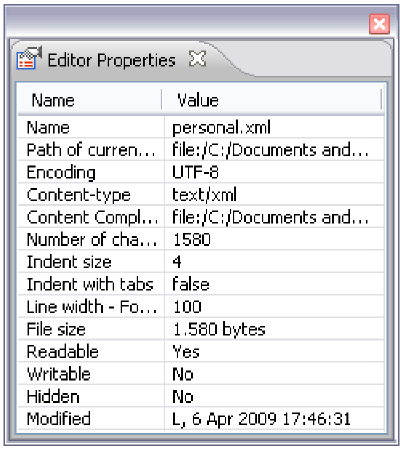
To close documents use one of the following methods:

- File → Close (**Ctrl+F4**) : Closes only the selected tab. All other tab instances remain.
- File → Close All (**Ctrl+Shift+F4**): Closes all opened documents. If a document is modified or has no file, a prompt to save, not to save, or cancel the save operation is displayed.
- Close - accessed by right-clicking on an editor tab: Closes the selected editor.
- Close Other Files - accessed by right-clicking on an editor tab: Closes the other files except the selected tab.
- Close All - accessed by right-clicking on an editor tab: Closes all open editors within the panel.

Viewing file properties

In the Properties view you can quickly access information about the current edited document like the character encoding, full path on the file system, schema used for content completion and document validation, document type name and path, associated transformation scenario, if the file is read-only, document's total number of characters, line width, if indent with tabs is enabled and the indent size. The view can be accessed by going to Window+Show View → Other ...+oXygen+Editor properties

Figure 4.11. The Properties View



Name	Value
Name	personal.xml
Path of curren...	file:/C:/Documents and...
Encoding	UTF-8
Content-type	text/xml
Content Compl...	file:/C:/Documents and...
Number of cha...	1580
Indent size	4
Indent with tabs	false
Line width - Fo...	100
File size	1.580 bytes
Readable	Yes
Writable	No
Hidden	No
Modified	L, 6 Apr 2009 17:46:31

To copy a value from the *Properties View* in the clipboard, for example the full file path, use the *Copy* action available on the right-click menu of the view.

Editing XML documents

Associate a schema to a document

Setting a schema for the Content Completion

In case you are editing document fragments, for instance the chapters from a book each one in a separate file, you can activate the Content Completion for these fragments in two ways:

Setting a default schema

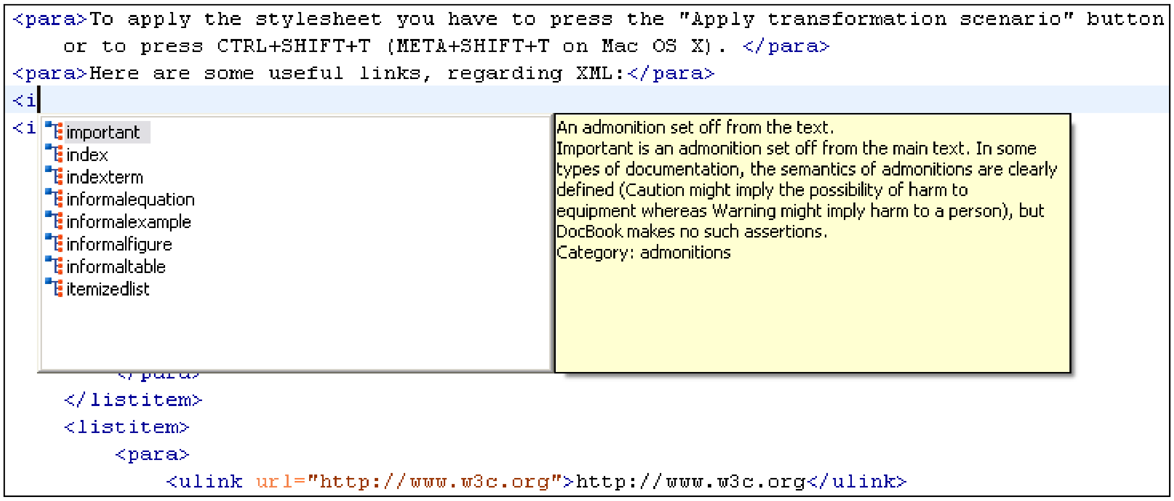
The list of document types available at Options → Preferences -> Document Type Association contains a set of rules for associating a schema with the current document when no schema is specified within the document. The schema is one of the types: XML Schema, XML Schema with embedded Schematron rules, Relax NG, Relax NG with embedded Schematron rules, Schematron, DTD, NRL, NVDL.

The rules are applied in the order they appear in the table and take into account the local name of the root element, the default namespace and the file name of the document.

Important

The editor is creating the Content Completion lists by analysing the specified schema and the current context (the position in the editor). If you change the schema you can observe that the list of tags to be inserted is changing.

Figure 4.12. Content completion driven by a DocBook DTD



```
<para>To apply the stylesheet you have to press the "Apply transformation scenario" button
    or to press CTRL+SHIFT+T (META+SHIFT+T on Mac OS X). </para>
<para>Here are some useful links, regarding XML:</para>
<i
<i
      </para>
    </listitem>
    <listitem>
        <para>
            <ulink url="http://www.w3c.org">http://www.w3c.org</ulink>
```

- important
- index
- indexterm
- informalequation
- informalexample
- informalfigure
- informaltable
- itemizedlist

An admonition set off from the text.
Important is an admonition set off from the main text. In some types of documentation, the semantics of admonitions are clearly defined (Caution might imply the possibility of harm to equipment whereas Warning might imply harm to a person), but DocBook makes no such assertions.
Category: admonitions

Adding a Processing Instruction

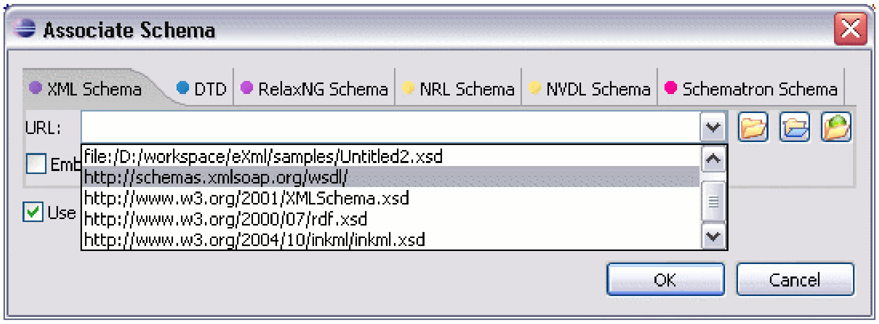
The same effect is obtained by configuring a processing instruction that specifies the schema to be used. The advantage of this method is that you can configure the Content Completion for each file. The processing instruction must be added at the beginning of the document, just after the XML prologue:

```
<?oxygen RNGSchema="file:/C:/work/relaxng/personal.rng" type="xml"?>
```

Select menu Document+XML Document → Associate schema... or click the toolbar button Associate schema to open a dialog for selecting a schema used for Content Completion and document validation. The schema is one of the types: XML Schema (with or without embedded Schematron rules), DTD, Relax NG (with or without embedded Schematron rules), NRL, NVDL, Schematron.

This is a dialog helping the user to easily associate a schema file with the edited document . Enables definition of a XML Document Prolog using the system identifier of a XML Schema, DTD, Relax NG (full or compact syntax) schema, NRL (Namespace Routing Language) schema, NVDL (Namespace-based Validation Dispatching Language) schema or Schematron schema. If you associate an XML Schema with embedded Schematron rules or a Relax NG schema with embedded Schematron rules you have to check the *Embedded Schematron rules* checkbox available for these two types of schemas.

Figure 4.13. Associate schema dialog



Associate Schema

XML Schema | DTD | RelaxNG Schema | NRL Schema | NVDL Schema | Schematron Schema

URL:

file:/D:/workspace/eXml/samples/Untitled2.xsd
http://schemas.xmlsoap.org/wsdl/
http://www.w3.org/2001/XMLSchema.xsd
http://www.w3.org/2000/07/rdf.xsd
http://www.w3.org/2004/10/inkml/inkml.xsd

☐ Emb

☑ Use

OK Cancel

When associating a XML Schema to the edited document if the root element of the document defines a default namespace URI with a "xmlns" attribute the "Associate schema" action adds a xsi:schemaLocation attribute. Otherwise it adds a xsi:noNamespaceSchemaLocation attribute.

The URL combo box contains a predefined set of schemas that are used more often and it also keeps a history of the last used schemas.

<oXygen/> logs the URL of the detected schema in the Status view.

The *oxygen* processing instruction has the following attributes:

RNGSchema specifies the path to the Relax NG schema associated with the current document

type specifies the type of Relax NG schema, is used together with the RNGSchema attribute and can have the value "xml" or "compact".

NRLSchema specifies the path to the NRL schema associated with the current document

NVDLSchema specifies the path to the NVDL schema associated with the current document

SCHSchema specifies the path to the SCH schema associated with the current document

Learning document structure

When working with documents that do not specify a schema, or for which the schema is not known or does not exist, <oXygen/> is able to learn and translate it to a DTD, which in turn can be saved to a file in order to provide a DTD for Content Completion and document validation. In addition to being useful for quick creation of a DTD that will be capable of providing an initialization source for the Content Completion assistant. This feature can also be used to produce DTDs for documents containing personal or custom element types.

When it is opened a document that does not specify a schema <oXygen/> automatically learns the document structure and uses it for Content Completion. To disable this feature uncheck the checkbox Learn on open document from Preferences.

Procedure 4.3. To create a DTD:

1. Open the structured document from which a DTD will be created.
2. Select menu XML → Learn Structure (**Ctrl+Shift+L**) to read the mark-up structure of the current document so that it can be saved as a DTD using the Save Structure option. <oXygen/> will learn the document structure, when finished displaying words Learn Complete in the Message Pane of the Editor Status bar.
3. Select menu Document+XML Document → Save Structure (**Ctrl+Shift+S**) to display the Save Structure dialog, used to name and create DTD documents learnt by the Learn Structure function.

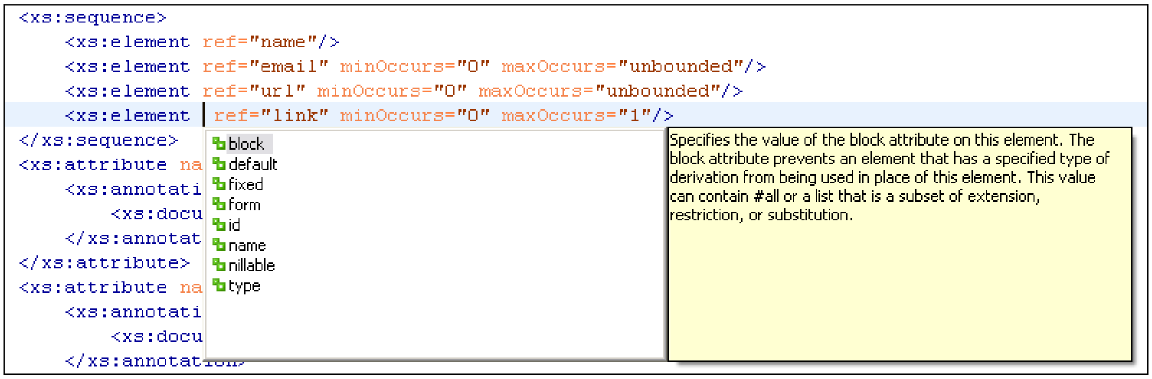
Note

The resulting DTD is only valid for documents containing the elements and structures defined by the document used as the input for creating the DTD. If new element types or structures are defined in a document, they must be added to the DTD in order for successful validation.

Streamline with Content Completion

<oXygen/>'s intelligent Content Completion feature is a content assistant that enables rapid, in-line identification and insertion of structured language elements, attributes and in some cases their parameter options.

Figure 4.14. Content Completion Assistant



```
<xs:sequence>
    <xs:element ref="name"/>
    <xs:element ref="email" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="url" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="link" minOccurs="0" maxOccurs="1"/>
</xs:sequence>
<xs:attribute na
    <xs:annotati
        <xs:docu
    </xs:annotat
</xs:attribute>
<xs:attribute na
    <xs:annotati
        <xs:docu
    </xs:annotation>
```

Specifies the value of the block attribute on this element. The block attribute prevents an element that has a specified type of derivation from being used in place of this element. This value can contain #all or a list that is a subset of extension, restriction, or substitution.

If the Content Completion assistant is enabled in user preferences (the option *Use Content Completion*) it is automatically displayed whenever the < character is entered into a document or by pressing **CTRL+Space** on a partial element or attribute name. Moving the focus to highlight an element and pressing the **Enter** key or the **Tab** key, inserts both the start and end tags of the highlighted element into the document.

The DTD, XML Schema, Relax NG, NRL or NVDL schema used to populate the Content Completion assistant is specified in the following methods, in order of precedence:

- The schema specified explicitly in the document. In this case <oXygen/> reads the beginning of the document and resolves the location of the DTD, XML Schema, Relax NG schema, NRL or NVDL schema.

  ### Note

  Limitation: In case of XML Schema the content completion takes into account only the schema declarations from the root element of the document. If a schema declaration is attached to other element of the XML document it is ignored.

- The default schema rule declared in the Document Type Association preferences panel which matches the edited document.

- For XSLT stylesheets the schema specified in the <oXygen/> Content Completion options.<oXygen/> will read the Content Completion settings when the prolog fails to provide or resolve the location of a DTD, XML Schema, Relax NG or NVDL schema.

- For XML Schemas the schema specified in the <oXygen/> Content Completion options.<oXygen/> will read the Content Completion settings and the specified schema will enhance the content completion inside the *xs:annotation/xs:appinfo* elements of the XML Schema.

After inserting, the cursor is positioned directly before the > character of the start tag, if the element has attributes, in order to enable rapid insertion of any attributed supported by the element, or after the > char of the start tag if the element has no attributes. Pressing the space bar, directly after element insertion will again display the assistant. In this instance the attributes supported by that element will be displayed. If an attribute supports a fix set of parameters, the assistant will display the list of valid parameter. If the parameter setting is user defined and therefore variable, the assistant will be closed to enable manual insertion. The values of the attributes can be learned from the same elements in the current document.

If you press **CTRL + Enter** instead of **Enter** or **Tab** after inserting the start and end tags in the document <oXygen/> will insert an empty line between the start and end tag and the cursor will be positioned between on the empty line on an indented position with regard to the start tag.

If the feature Add Element Content of Content Completion is enabled all the elements that the new element must contain, as specified in the DTD or XML Schema, are inserted automatically in the document. The Content Completion assistant can also add optional content and first choice particle, as specified in the DTD or XML Schema, for the element if the two options are enabled.

The content assistant can be started at any time by pressing **CTRL+Space** The effect is that the context-sensitive list of proposals will be shown in the current position of the caret in the edited document if element, attribute or attribute value insertion makes sense. Such positions are: anywhere within a tag name or at the beginning of a tag name in an XML document, XML Schema, DTD or Relax NG (full or compact syntax) schema, anywhere within an attribute name or at the beginning of an attribute name in any XML document with an associated schema, and within attribute values or at the beginning of attribute values in XML documents where lists of possible values have been defined for that element in the schema associated with the document.

The content of the Content Completion assistant is dependent on the element structure specified in the DTD, XML Schema, Relax NG (full or compact syntax) schema or NRL, NVDL schema associated to the edited document.

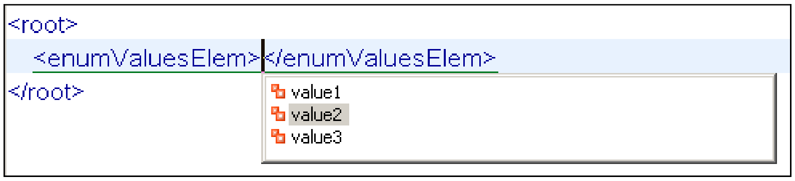
The number and type of elements displayed by the assistant is dependent on the current position of the cursor in the structured document . The child elements displayed within a given element are defined by the structure of the specified DTD, XML Schema, Relax NG (full or compact syntax) schema or NRL, NVDL schema. All elements that can't be child elements of the current element according to the specified schema are not displayed.

Inside Relax NG documents the Content Completion assistant is able to present element values if such values are specified in the Relax NG schema. Also in Relax NG documents the Content Completion assistant presents additional values of type ID for an *anyURI* data type. It presents also pattern names defined in the Relax NG schema as possible values for pattern references. For example if the schema defines an *enumValuesElem* element

```
<element name="enumValuesElem">
    <choice>
        <value>value1</value>
        <value>value2</value>
        <value>value3</value>
    </choice>
</element>
```

in documents based on the schema the Content Completion assistant offers the list of values:

Figure 4.15. Content Completion assistant - element values in Relax NG documents



```
<root>
  <enumValuesElem></enumValuesElem>
</root>
```

If only one element name must be displayed by the content assistant then the assistant is not displayed any more but this only option is automatically inserted in the document at the current cursor position.

If the schema for the edited document defines attributes of type ID and IDREF the content assistant will display for IDREF attributes a list of all the ID values already present in the document for an easy insertion of a valid ID value at the cursor position in the document. This is available for documents that use DTD, XML Schema and Relax NG schema.

Also values of all the *xml:id* attributes are treated as ID attributes and collected and displayed by the content completion assistant as possible values for *anyURI* attributes defined in the schema of the edited document. This works only for XML Schema and Relax NG schemas.

For documents that use an XML Schema or Relax NG schema the content assistant offers proposals for attributes and elements values that have as type an enumeration of tokens. Also if a default value or a fixed value is defined in the XML Schema used in validation for an attribute or element then that value is offered in the content completion window.

If the edited document is not associated with a schema explicitly using the usual mechanisms for associating a DTD or XML Schema with a document or using a processing instruction introduced by the *Associate schema* action the content assistant will extract the elements presented in the pop-up window from the default schema.

If the schema for the document is of type XML Schema, Relax NG (full syntax), NVDL or DTD and it contains element, attributes or attributes values annotations, these will be presented when the content completion window is displayed, if the option *Show annotations* is enabled. Also the annotation is presented in a small tooltip window displayed automatically when the mouse hovers over an element or attribute annotated in the associated schema of the edited document.

In an XML Schema annotations are put in an <xs:annotation> element:

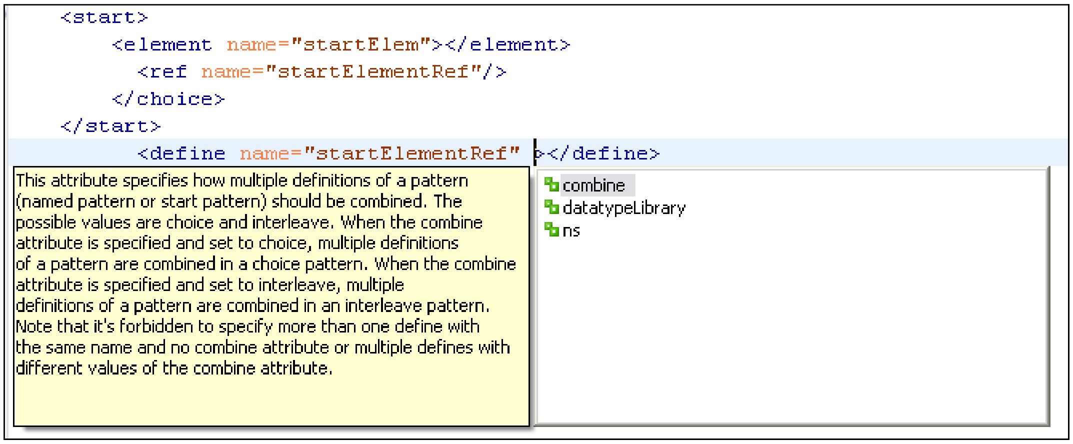
```
<xs:annotation>
    <xs:documentation>
        Description of the element.
    </xs:documentation>
</xs:annotation>
```

If the current element / attribute in the edited document does not have an annotation in the schema and that schema is of the type XML Schema <oXygen/> seeks an annotation in the type definition of the element / attribute or, if no annotation is found there, in the parent type definition of that definition, etc.

In a Relax NG schema any element outside the Relax NG namespace (*http://relaxng.org/ns/structure/1.0*) is handled as annotation and the text content is displayed in the annotation window together with the content completion window:

For NVDL schemas annotations for the elements / attributes in the referred schemas (XML Schema, RNG, etc) are presented

Figure 4.16. Schema annotations displayed at Content Completion



```
    <start>
        <element name="startElem"></element>
          <ref name="startElementRef"/>
        </choice>
    </start>
          <define name="startElementRef" ></define>
```

This attribute specifies how multiple definitions of a pattern (named pattern or start pattern) should be combined. The possible values are choice and interleave. When the combine attribute is specified and set to choice, multiple definitions of a pattern are combined in a choice pattern. When the combine attribute is specified and set to interleave, multiple definitions of a pattern are combined in an interleave pattern. Note that it's forbidden to specify more than one define with the same name and no combine attribute or multiple defines with different values of the combine attribute.

The following HTML tags are recognized inside the text content of an XML Schema annotation: *p*, *br*, *ul*, *li*. They are rendered as in an HTML document loaded in a web browser: *p* begins a new paragraph, *br* breaks the current line, *ul* encloses a list of items, *li* encloses an item of the list.

For DTD <oXygen/> defines a custom mechanism for annotation using comments enabled from the option *Use DTD comments as annotations* . The text of a comment with the following format will be presented on content completion:

```
<!--doc:Description of the element. -->
```

The operation of the Content Completion assistant is configured by the options available in the options group called Content Completion.

Code templates

You can define short names for predefined blocks of code called code templates. The short names are displayed in the content completion window if the word at cursor position is a prefix of such a short name. <oXygen/> comes with a lot of predefined code templates but you can define your own code templates for any type of editor. For more details see the example for XSLT editor code templates.

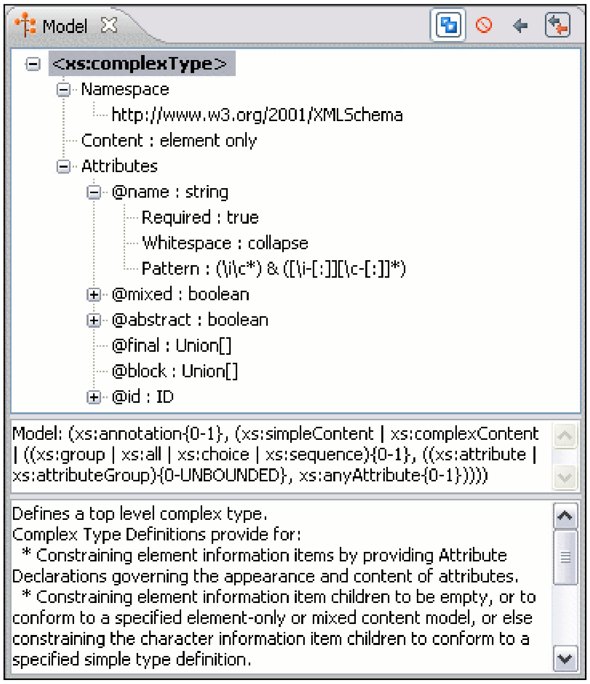
Content Completion helper panels

Information about the current element being edited are also available in the Model panel and Attributes panel, located on the left-hand side of the main window. The Model panel and the Attributes panel combined with the powerful Outline view provide spacial and insight information on the edited document.

The Model panel

The Model panel presents the structure of the current edited tag and tag documentation defined as annotation in the schema of the current document. Open the Model panel from Window → Show View → Other+oXygen+Model view

Figure 4.17. The Model View



```
<xs:complexType>
  Namespace
    http://www.w3.org/2001/XMLSchema
  Content : element only
  Attributes
    @name : string
      Required : true
      Whitespace : collapse
      Pattern : (\i\c*) & ([\i-[:]][\c-[:]]*)
    @mixed : boolean
    @abstract : boolean
    @final : Union[]
    @block : Union[]
    @id : ID
```

Model: (xs:annotation{0-1}, (xs:simpleContent | xs:complexContent | ((xs:group | xs:all | xs:choice | xs:sequence){0-1}, ((xs:attribute | xs:attributeGroup){0-UNBOUNDED}, xs:anyAttribute{0-1}))))

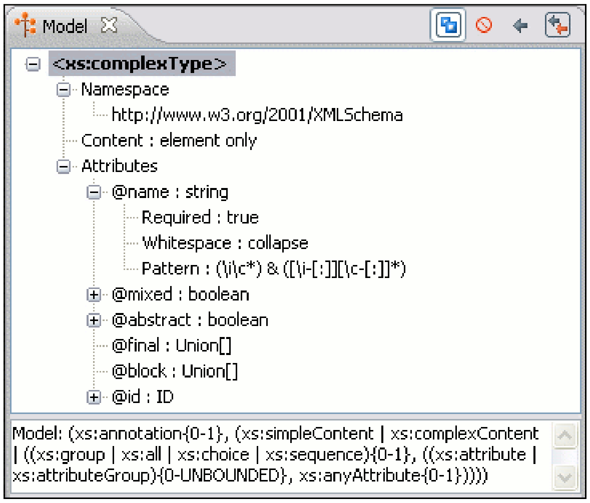
Defines a top level complex type.
Complex Type Definitions provide for:
 * Constraining element information items by providing Attribute Declarations governing the appearance and content of attributes.
 * Constraining element information item children to be empty, or to conform to a specified element-only or mixed content model, or else constraining the character information item children to conform to a specified simple type definition.

The Element Structure panel

The element structure panel shows the structure of the current edited or selected tag in a Tree format.

The information includes the name, model and attributes the currently edited tag may have. The allowed attributes are shown along with any restrictions they might possess.

Figure 4.18. The Element Structure panel



```
<xs:complexType>
    Namespace
        http://www.w3.org/2001/XMLSchema
    Content : element only
    Attributes
        @name : string
            Required : true
            Whitespace : collapse
            Pattern : (\i\c*) & ([\i-[:]][\c-[:]]*)
        @mixed : boolean
        @abstract : boolean
        @final : Union[]
        @block : Union[]
        @id : ID
```

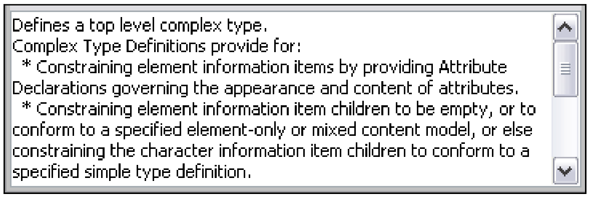
```
Model: (xs:annotation{0-1}, (xs:simpleContent | xs:complexContent
| ((xs:group | xs:all | xs:choice | xs:sequence){0-1}, ((xs:attribute |
xs:attributeGroup){0-UNBOUNDED}, xs:anyAttribute{0-1}))))
```

The Annotation panel

The Annotation panel shows the annotations that are present in the used schema for the currently edited or selected tag.

This information can be very useful to persons learning XML because it has small available definitions for each used tag.

Figure 4.19. The Annotation panel

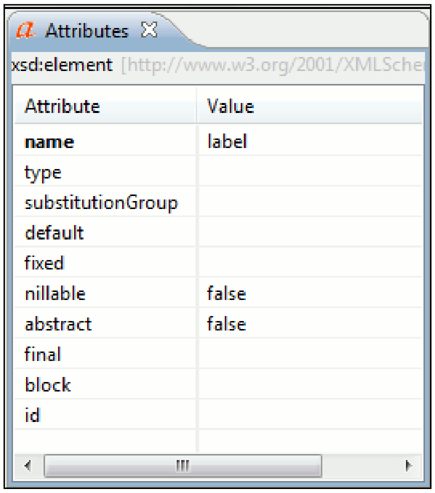


Defines a top level complex type.
Complex Type Definitions provide for:
  * Constraining element information items by providing Attribute Declarations governing the appearance and content of attributes.
  * Constraining element information item children to be empty, or to conform to a specified element-only or mixed content model, or else constraining the character information item children to conform to a specified simple type definition.

The Attributes panel

The Attributes panel presents all the possible attributes of the current element and allows to insert attributes in the current element or change the value of the attributes already used in the element. The attributes already present in the document are painted with a bold font. Clicking on the Value column of a table row will start editing the value of the attribute from the selected row. If the possible values of the attribute are specified as list in the schema associated with the edited document the Value column works as a combo box where you can select one of the possible values to be inserted in the document. The attributes table is sortable, 3 sorting orders being available by clicking on the columns' names. Thus the table's contents can be sorted in ascending order, in descending order or in a custom order, where the used attributes are placed at the beginning of the table as they appear in the element followed by the rest of the allowed elements as they are declared in the associated schema.

Figure 4.20. The Attributes panel

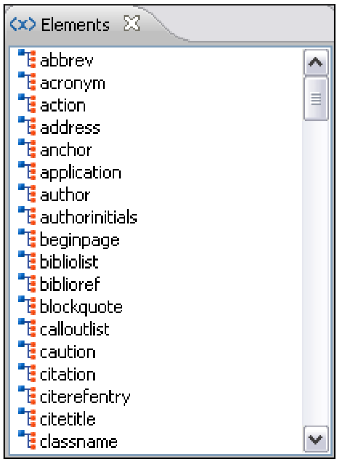


xsd:element [http://www.w3.org/2001/XMLSche

Attribute	Value
name	label
type	
substitutionGroup	
default	
fixed	
nillable	false
abstract	false
final	
block	
id	

The Elements view

Figure 4.21. The Elements View



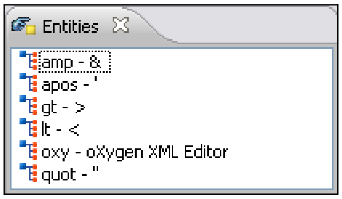
abbrev
acronym
action
address
anchor
application
author
authorinitials
beginpage
bibliolist
biblioref
blockquote
calloutlist
caution
citation
citerefentry
citetitle
classname

Presents a list of all defined elements that you can insert at the current caret position according to the schema used for content completion. Double-clicking any of the listed elements will insert that element in the edited document. All elements from a sequence are presented but the invalid proposals (which cannot be inserted in the current context) are grayed-out.

The Entities View

This view displays a list with all entities declared in the current document as well as built-in ones. Double clicking one of the entities will insert it at the current cursor position.

Figure 4.22. The Entities View



amp - &
apos - '
gt - >
lt - <
oxy - oXygen XML Editor
quot - "

Validating XML documents

The W3C XML specification states that a program should not continue to process an XML document if it finds a validation error. The reason is that XML software should be easy to write, and that all XML documents should be compatible. With HTML it was possible to create documents with lots of errors (like when you forget an end tag). One of the main reasons that HTML browsers are so big and incompatible, is that they have their own ways to figure out what a document should look like when they encounter an HTML error. With XML this should not be possible.

However, when creating an XML document, errors are very easily introduced. When working with large projects or many files, the probability that errors will occur is even greater. Determining that your project is error free can be time consuming and even frustrating. For this reason <oXygen/> provides functions that enable easy error identification and rapid error location.

Checking XML well-formedness

A *Well-Formed XML* document is a document that conforms to the XML syntax rules.

A *Namespace Well-Formed XML* document is a document that is Well-Formed XML and is also namespace-wellformed and namespace-valid.

The XML Syntax rules for Well-Formed XML are:

- All XML elements must have a closing tag.
- XML tags are case sensitive.
- All XML elements must be properly nested.
- All XML documents must have a root element.
- Attribute values must always be quoted.
- With XML, white space is preserved.

The namespace-wellformed rules are:

- All element and attribute names contain either zero or one colon
- No entity names, processing instruction targets, or notation names contain any colons.

The namespace-valid rules are:

- The prefix xml is by definition bound to the namespace name http://www.w3.org/XML/1998/namespace. It MAY, but need not, be declared, and MUST NOT be undeclared or bound to any other namespace name. Other prefixes MUST NOT be bound to this namespace name.

- The prefix xmlns is used only to declare namespace bindings and is by definition bound to the namespace name http://www.w3.org/2000/xmlns/. It MUST NOT be declared or undeclared. Other prefixes MUST NOT be bound to this namespace name.

- All other prefixes beginning with the three-letter sequence x, m, l, in any case combination, are reserved. This means that users SHOULD NOT use them except as defined by later specifications and processors MUST NOT treat them as fatal errors.

- The namespace prefix, unless it is xml or xmlns, MUST have been declared in a namespace declaration attribute in either the start-tag of the element where the prefix is used or in an ancestor element (i.e. an element in whose content the prefixed markup occurs). Furthermore, the attribute value in the innermost such declaration MUST NOT be an empty string.

If you select menu Document+Validate → Check Document Form (**Alt+Shift+V WCmd+Alt+V W**) or click the toolbar button ☑ Check Document Form <oXygen/> checks if your document is *Namespace Well-Formed XML*. If any error is found the result is returned to the Message Panel. Each error is one record in the Result List and is accompanied by an error message. Clicking the record will open the document containing the error and highlight the approximate location.

Example 4.2. Document which is not Well-Formed XML

```
<root><tag></root>
```

When "Check document form" is performed the following error is raised:

```
The element type "tag" must be terminated by the matching end-tag "</tag>"
```

To resolve the error, click in the result list record which will locate and highlight the errors approximate position. Identify which start tag is missing an end tag and insert </tag>.

Example 4.3. Document which is not namespace-wellformed

```
<x::y></x::y>
```

When "Check document form" is performed the following error is raised:

```
Element or attribute do not match QName production: QName::=(NCName':')?NCName.
```

Example 4.4. Document which is not namespace-valid

```
<x:y></x:y>
```

When "Check document form" is performed the following error is raised:

```
The prefix "x" for element "x:y" is not bound.
```

Also the files contained in the current project and selected with the mouse in the Project view can be checked for well-formedness with one action available on the popup menu of the Project view in the *Batch validation* submenu: ☑ Check well form

Validating XML documents against a schema

A *Valid* XML document is a *Well Formed* XML document, which also conforms to the rules of a schema which defines the legal elements of an XML document. The schema type can be: XML Schema, Relax NG (full or compact syntax), Schematron, Document Type Definition (DTD), Namespace Routing Language (NRL) or Namespace-based Validation Dispatching Language (NVDL).

The purpose of the schema is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.

The <oXygen/> ☑ Validate document function ensures that your document is compliant with the rules defined by an associated DTD, XML Schema, Relax NG or Schematron schema. XML Schema or Relax NG Schema can embed Schematron rules. For Schematron it is possible to select the validation phase.

Note

Validation of an XML document against a W3C XML Schema containing a type definition with a *minOccurs* or *maxOccurs* attribute having a value larger than 256 limits the value to 256 and issues a warning about this restriction in the Message panel at the bottom of the <oXygen/> window. Otherwise for large values of the *minOccurs* and *maxOccurs* attributes the validator fails with an OutOfMemory error which practically makes <oXygen/> unusable without a restart of the entire application.

Note

Validation of an XML document against a deeply recursive Relax NG schema may fail with a stack overflow error. It happens very rarely and the cause is the unusual depth of the Relax NG pattern recursion needed to match an element of the document against the schema and the depth exceeds the default stack size allocated by the Java virtual machine. The error can be overcome by simply setting a larger stack size to the JVM at startup using the -Xss parameter, for example -Xss1m.

Note

Validation of an XML document against a W3C XML Schema or Relax NG Schema with embedded ISO Schematron rules allows XPath 2.0 in the expressions of the ISO Schematron rules. This ensures that both XPath 1.0 and XPath 2.0 expressions are accepted in the embedded ISO Schematron rules and are enforced by the validation operation. For embedded Schematron 1.5 rules the version of XPath is set with a user preference.

Note

Validation of an XML document against a Relax NG schema that declares a custom datatype library requires adding the library files to the <oXygen/> classpath.

Marking Validation Errors

A line with a validation error or warning will be marked in the editor panel by underlining the error region with a red color. Also a red sign will mark the position in the document of that line on the right side ruler of the editor panel. The same will happen for a validation warning, only the color will be yellow instead of red.

The ruler on the right of the document is designed to display the errors found during the validation process and also to help the user to locate them more easily. The ruler contains the following areas:

- top area containing a success validation indicator that will turn green in case the validation succeeded or red otherwise.
- middle area where the errors markers are depicted in red. The number of markers shown can be limited by modifying the setting Window → Preferences+oXygen/Editor / Document checking+Maximum number of errors reported per document

Clicking on a marker will highlight the corresponding text area in the editor. The error message is displayed both in the tool tip and in the error area on the bottom of the editor panel.

Status messages from every validation action are logged into the Console view.

Validation Example

Example 4.5. Validation error messages

In our example we will use the case where a DocBook listitem element does not match the rules of the `docbookx.dtd`. In this case running *Validate Document* will return the following error:

```
E The content of element type "listitem" must
match"(calloutlist|glosslist|itemizedlist|orderedlist|segmentedlist|
simplelist|variablelist| caution|important|note|tip|warning|
literallayout|programlisting|programlistingco|screen|
screenco|screenshot|synopsis|cmdsynopsis|
funcsynopsis|classsynopsis|fieldsynopsis| constructorsynopsis|
destructorsynopsis|methodsynopsis|formalpara|para|simpara|
address|blockquote|graphic|graphicco|mediaobject|
mediaobjectco|informalequation| informalexample|
informalfigure|informaltable|equation|example|
figure|table|msgset|procedure|sidebar|qandaset|anchor|
bridgehead|remark|highlights|abstract|authorblurb|epigraph|
indexterm|beginpage)+".
```

As you can see, this error message is a little more difficult to understand, so understanding of the syntax or processing rules for the DocBook XML DTD's "listitem" element is required. However, the error message does give us a clue as to the source of the problem, but indicating that "The content of element type "listitem" must match".

Luckily most standards based DTD's, XML Schema's and Relax NG schemas are supplied with reference documentation. This enables us to lookup the element and read about it. In this case we would want to learn about the child elements of "listitem" and their nesting rules. Once we have correctly inserted the required child element and nested it in accordance with the XML rules, the document will become valid on the next validation test.

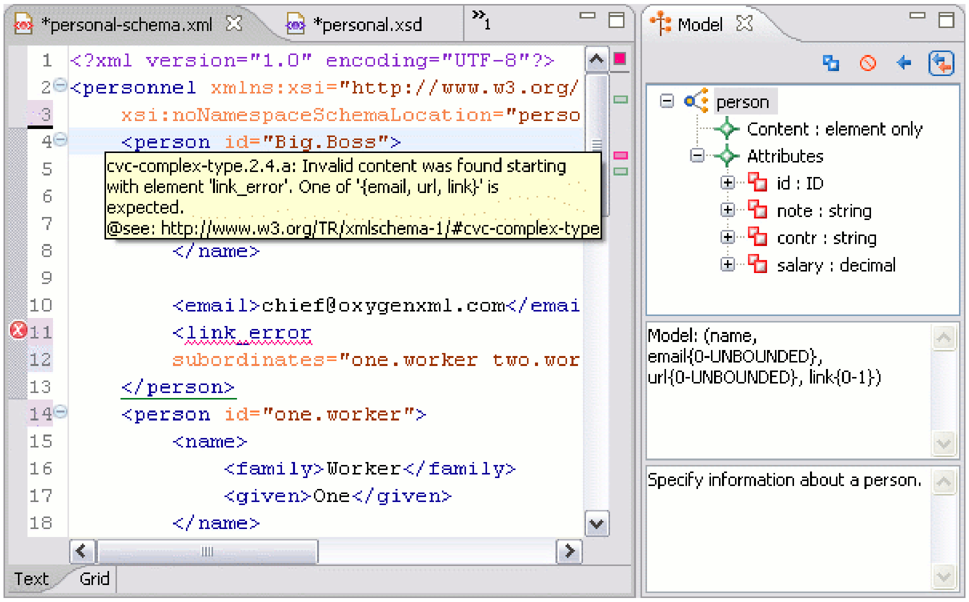
Caching the Schema Used for Validation

If you don't change the active editor and you don't switch to other application the schema associated to the current document is parsed and cached at the first validate action and is reused by the next *Validate document* actions without re parsing it. This increases the speed of the validate action starting with the second execution if the schema is large or is located on a remote server on the Web. To reset the cache and re parse the schema you have to use the Reset cache and validate action.

Validate As You Type

<oXygen/> can be configured to mark validation errors in the edited document as you modify it using the keyboard. If you enable the *Validate as you type* option any validation errors and warnings will be highlighted automatically in the editor panel after the configured delay from the last key typed, with underline markers in the editor panel and small rectangles on the right side ruler of the editor panel, in the same way as for manual validation invoked by the user.

Figure 4.23. Validate as you type on the edited document



```
<?xml version="1.0" encoding="UTF-8"?>
<personnel xmlns:xsi="http://www.w3.org/
    xsi:noNamespaceSchemaLocation="perso
    <person id="Big.Boss">
        </name>

        <email>chief@oxygenxml.com</emai
        <link_error
        subordinates="one.worker two.wor
    </person>
    <person id="one.worker">
        <name>
            <family>Worker</family>
            <given>One</given>
        </name>
```

cvc-complex-type.2.4.a: Invalid content was found starting with element 'link_error'. One of '{email, url, link}' is expected.
@see: http://www.w3.org/TR/xmlschema-1/#cvc-complex-type

person
- Content : element only
- Attributes
  - id : ID
  - note : string
  - contr : string
  - salary : decimal

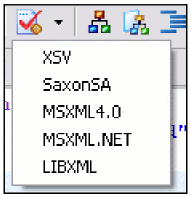
Model: (name, email{0-UNBOUNDED}, url{0-UNBOUNDED}, link{0-1})

Specify information about a person.

Custom validation of XML documents

If you need to validate the edited document with other validation engine than the built-in one you have the possibility to configure external validators as custom validation engines in <oXygen/>. After such a custom validator is properly configured in Preferences it can be applied on the current document with just one click on the External Validation toolbar. The document is validated against the schema declared in the document.

Figure 4.24. External validation toolbar



Some validators are configured by default but they are third party processors which do not support the output message format for linked messages described above:

LIBXML

included in <oXygen/> (Windows edition), associated to XML Editor, able to validate the edited document against XML Schema, Relax NG schema full syntax, internal DTD (included in the XML document) or a custom schema type. XML catalogs support(**--catalogs**) and XInclude processing(**--xinclude**) are enabled by default in the preconfigured LIBXML validator. The **--postvalid** flag is set as default allowing LIBXML to validate correctly the main document even if the XInclude fragments contain IDREFS to ID's located in other fragments.

For validation against an external DTD specified by URI in the XML document the parameter *--dtdvalid ${ds}* must be added manually to the DTD validation command line. ${ds} represents the detected DTD declaration in the XML document.

Note

Known problem: file paths containing spaces are not handled correctly in the LIBXML processor. For example the built-in XML catalog files of the predefined document types (DocBook, TEI, DITA, etc) are not handled by LIBXML if <oXygen/> is installed in the default location on Windows (C:\Program Files) because the built-in XML catalog files are stored in the *frameworks* subdirectory of the installation directory which in this case contains at least a space character.

Note

On Mac OS X if the full path to the LIBXML executable file is not specified in the *Executable path* text field some errors may occur on validation against a W3C XML Schema like:

```
Unimplemented block at ... xmlschema.c
```

These errors can be avoided by specifying the full path to the LIBXML executable file.

Saxon SA

included in <oXygen/>. It is associated to XML Editor and XSD Editor. It is able to validate XML Schema schemas and XML documents against XML Schema schemas. The validation is done according to the W3C XML Schema 1.0 specification or according to the W3C XML Schema 1.1 one. This can be configured in Preferences.

MSXML 4.0

included in <oXygen/> (Windows edition). It is associated to XML Editor, XSD Editor and XSL Editor. It is able to validate the edited document against XML Schema, internal DTD (included in the XML document), external DTD or a custom schema type.

MSXML.NET

included in <oXygen/> (Windows edition). It is associated to XML Editor, XSD Editor and XSL Editor. It is able to validate the edited document against XML Schema, internal DTD (included in the XML document), external DTD or a custom schema type.

XSV

not included in <oXygen/>. A Windows distribution of XSV can be downloaded from: ftp://ftp.cogsci.ed.ac.uk/pub/XSV/XSV31.EXE [ftp://ftp.cogsci.ed.ac.uk/pub/XSV/XSV31.EXE] A Linux distribution can be downloaded from ftp://ftp.cogsci.ed.ac.uk/pub/XSV/XSV-3.1-1.noarch.rpm [ftp://ftp.cogsci.ed.ac.uk/pub/XSV/XSV-3.1-1.noarch.rpm] The executable path is configured already in <oXygen/> for the installation directory `[oXygen-install-dir]/xsv`. If it is installed in a different directory the predefined executable path must be corrected in Preferences. It is associated to XML Editor and XSD Editor. It is able to validate the edited document against XML Schema or a custom schema type.

SQC (Schema Quality Checker from IBM)

not included in <oXygen/>. It can be downloaded from here [http://www.alphaworks.ibm.com/tech/xmlsqc?open&l=xml-dev,t=grx,p=shecheck] (it comes as a .zip file, at the time of this writing SQC2.2.1.zip is about 3 megabytes). The executable path and working directory are configured already for the SQC installation directory `[oXygen-install-dir]/sqc`. If it is

	installed in a different directory the predefined executable path and working directory must be corrected in Preferences. It is associated to XSD Editor.
Intel® XML Software Suite	not included in <oXygen/>. Before you can use this validator you must install the Intel® XML Software Suite and configure it in the <oXygen/> preferences. It is associated to XML Editor and XSD Editor. It can validate an XML Schema schema and an XML document against a DTD or against an XML Schema schema.

Linked output messages of an external engine

The output messages of the validation engines are displayed in an output view at the bottom of the <oXygen/> window. If an output message of the external validation engine (warnings, errors, fatal errors, etc) spans between three to five lines of text and has the following format then the message is linked to a location in the validated document so that a click on the message in the output view highlights the location of the message in an editor panel containing the file referred in the message. This behavior is similar to the linked messages generated by the default built-in validator. The format for linked messages is:

- Type:[F|E|W] (the string "Type:" followed by a letter for the type of the message: fatal error, error, warning - this line is optional in a linked message)
- SystemID: a system ID of a file (the string "SystemID:" followed by the system ID of the file that will be opened for highlighting when the message is clicked in the output message - usually the validated file, the schema file or an included file)
- Line: a line number (the string "Line:" followed by the number of the line that will be highlighted)
- Column: a column number (the string "Column:" followed by the number of the column where the highlight will start on the highlighted line - this line is optional in a linked message)
- Description: the text of the message (the string "Description:" followed by the text content of the message that will be displayed in the output view)

Validation Scenario

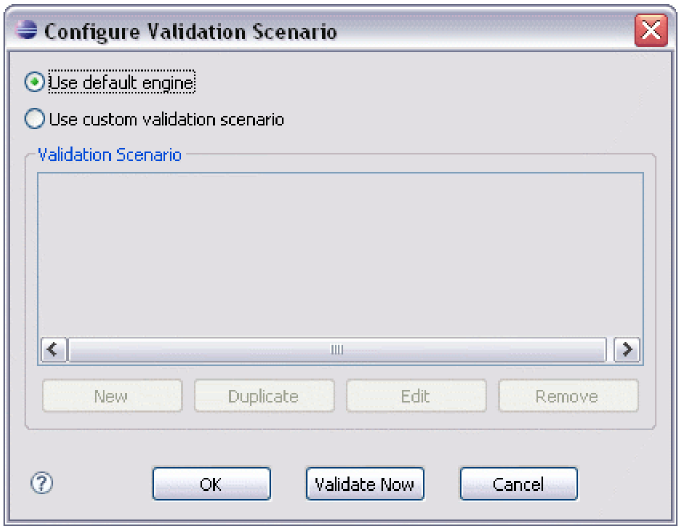
A complex XML document is usually split in smaller interrelated modules which do not make much sense individually and which cannot be validated in isolation due to interdependencies with the other modules. A mechanism is needed to set the main module of the document which in fact must be validated when an imported module needs to be checked for errors.

A typical example is the chunking DocBook XSL stylesheet which has `chunk.xsl` as the main module and which imports a stylesheet module called `param.xsl` which only defines XSLT parameters and other modules called `chunk-common.xsl` and `chunk-code.xsl`. The module `chunk-common.xsl` defines a named XSLT template with the name "chunk" which is called by `chunk-code.xsl`. The parameters defined in `param.xsl` are used in the other modules without being redefined.

Validation of `chunk-code.xsl` as an individual XSLT stylesheet issues a lot of misleading errors referring to parameters and templates used but undefined which are only caused by ignoring the context in which this module is used in real XSLT transformations and in which it should be validated. To validate such a module properly a validation scenario must be defined which sets the main module of the stylesheet and also the validation engine used to find the errors. Usually this is the engine which applies the transformation in order to detect by validation the same errors that would be issued by transformation.

To define a validation scenario first open the *Configure Validation Scenario* dialog. You do this with the *Configure Validation Scenario* action available on the menu XML and on the toolbar of the <oXygen/> plugin. You can use the default engine set in **Preferences**, or use a custom validation scenario. The list of reusable scenarios for documents of the same type as the current document is displayed.

Figure 4.25. Configure Validation Scenarios



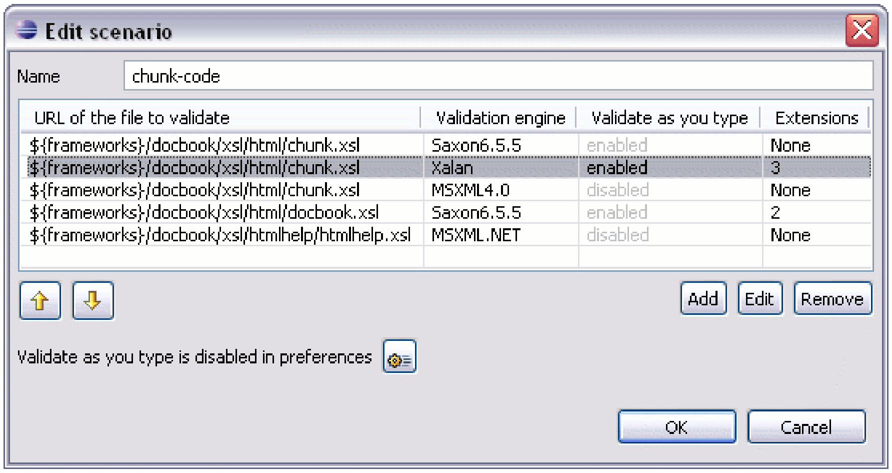
Configure Validation Scenario

● Use default engine

○ Use custom validation scenario

A validation scenario is created or edited in a special dialog opened with the *New* button or with the *Edit* one.

Figure 4.26. Edit a Validation Scenario



Edit scenario

Name: chunk-code

URL of the file to validate	Validation engine	Validate as you type	Extensions
${frameworks}/docbook/xsl/html/chunk.xsl	Saxon6.5.5	enabled	None
${frameworks}/docbook/xsl/html/chunk.xsl	Xalan	enabled	3
${frameworks}/docbook/xsl/html/chunk.xsl	MSXML4.0	disabled	None
${frameworks}/docbook/xsl/html/docbook.xsl	Saxon6.5.5	enabled	2
${frameworks}/docbook/xsl/htmlhelp/htmlhelp.xsl	MSXML.NET	disabled	None

Validate as you type is disabled in preferences

The table columns are:

URL of the file to validate
: The URL of the main module which includes the current module and which is the entry module of the validation process when the current module is validated.

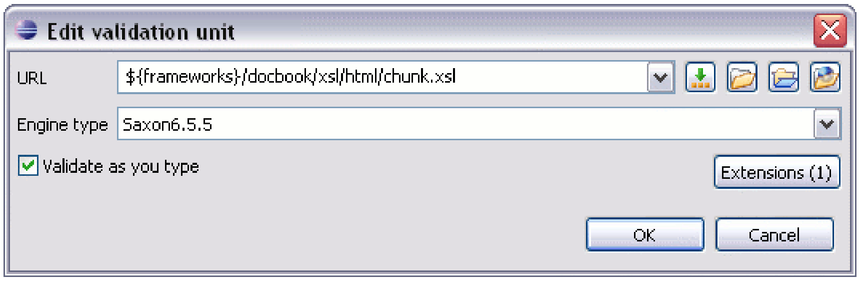
Validation engine
: One of the engines available in <oXygen/> for validation of the type of document to which the current module belongs.

Validate as you type
: If this option is checked then the validation operation defined by this row of the table is applied also by the Validate as you type feature. If the *Validate as you type* feature is disabled in Preferences then this option does not take effect as the Preference setting has higher priority.

Extensions — A list of Java jar files or classes which implement extensions of the language of the current module. For example when the current module is an XSLT stylesheet an extension jar contains the implementation of the XSLT extension functions or the XSLT extension elements used in the stylesheet which includes the current module.

A row of the table is created or edited in the following dialog:

Figure 4.27. Edit a Validation Unit



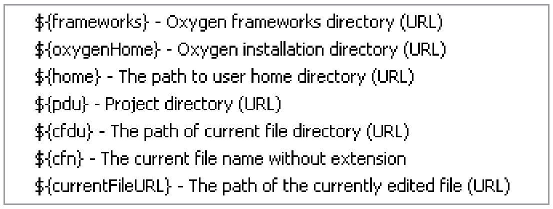
URL ${frameworks}/docbook/xsl/html/chunk.xsl

Engine type Saxon6.5.5

☑ Validate as you type

The components of the dialog are the same as the columns of the table displayed in the scenario edit dialog. The URL of the main module can be specified with the help of a file browser for the local file system (the button), with the help of the Open FTP / SFTP / WebDAV dialog opened by the button or by inserting an editor variable from the following pop-up menu:

Figure 4.28. Insert an editor variable



${frameworks} - Oxygen frameworks directory (URL)
${oxygenHome} - Oxygen installation directory (URL)
${home} - The path to user home directory (URL)
${pdu} - Project directory (URL)
${cfdu} - The path of current file directory (URL)
${cfn} - The current file name without extension
${currentFileURL} - The path of the currently edited file (URL)

A second benefit of a validation scenario is that the stylesheet can be validated with several engines to make sure that it can be used in different environments with the same results. For example an XSLT stylesheet needs to be applied with Saxon 6.5, Xalan and MSXML 4.0 in different production systems.

Other examples of documents which can benefit of a validation scenario are a complex XQuery with a main module which imports modules developed independently but validated in the context of the main module of the query and an XML document in which the master file includes smaller fragment files using XML entity references. In an XQuery validation scenario the default validator of <oXygen/> (Saxon 9) or any connection to a database that supports validation (Oracle 11g, Berkeley DB XML Database, IBM DB2, eXist XML Database, MarkLogic, Microsoft SQL Server, Software AG Tamino, Documentum xDb (X-Hive/DB) XML Database, TigerLogic, MySQL) can be set as validation engine.

Validation Actions in the User Interface

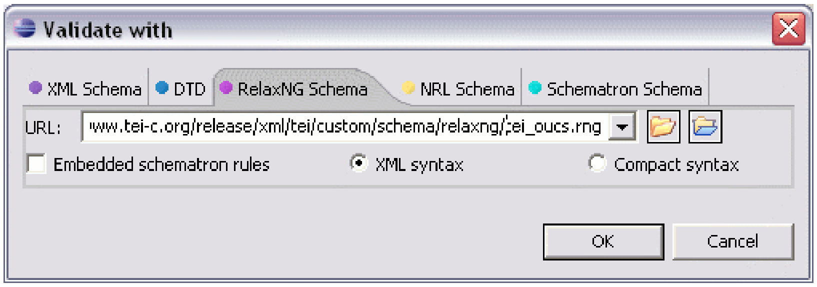
Use one of the actions for validating the current document:

- Select menu XML → Validate Document (**Alt+Shift+V V (Cmd+Alt+V V on Mac OS)**) or click the button Validate Document available in the Validate toolbar to return an error result-list in the Message panel. Mark-up of current document is checked to conform with the specified DTD, XML Schema or Relax NG schema rules. It caches the schema and the next execution of the action uses the cached schema.

- Select menu XML → Reset Cache and Validate or click the button Reset Cache and Validate available in the Validate toolbar to reset the cache with the schema and validate the document. It returns an error result-list in the Message panel. Mark-up of current document is checked to conform with the specified DTD, XML Schema or Relax NG schema rules.

- Select menu XML → External Validation (**Alt+Shift+V E (Cmd+Alt+V E on Mac OS)**) or click the button External Validation available in the Validate toolbar to display the External Validation dialog, used to select the external schemas (XML Schema, DTD, Relax NG, NRL, NVDL, Schematron schema) and to execute the Validation operation on the current document using the selected schemas. Returns an error result-list in the Message panel. Mark-up of current document is checked to conform with the specified schemas rules.

Figure 4.29. The External validation dialog



URL: ww.tei-c.org/release/xml/tei/custom/schema/relaxng/tei_oucs.rng

☐ Embedded schematron rules ● XML syntax ○ Compact syntax

- Select contextual menu of Navigator or Package Explorer view,Batch Validation → Validate to validate all selected files with their declared schemas.

- Select contextual menu of Navigator or Package Explorer view,Batch Validation → Validate With ... to select a schema and validate all selected files with that schema.

- XML → Clear validation markers (**Alt+Shift+V X (Cmd+Alt+V X on Mac OS)**) or click the toolbar button Clear validation markers to clear the error markers added to the Problems view at the last validation of the current edited document.

Also you can select several files in the views like Package Explorer, Navigator and validate them with one click by selecting the action Validate selection or the action Validate selection with ... available from the contextual menu of that view, the submenu Batch Validate.

If there are too many validation errors and the validation process is long you can limit the maximum number of reported errors.

Resolving references to remote schemas with an XML Catalog

When a reference to a remote schema must be used in the validated XML document for interoperability purposes but a local copy of the schema should be actually used for validation for performance reasons the reference can be resolved to the local copy of the schema with an XML catalog. For example if the XML document contains a reference to a remote schema *docbook.rng*

```
<?oxygen RNGSchema="http://www.oasis-open.org/docbook/xml/5.0/rng/docbook.rng"
    type="xml"?>
```

it can be resolved to a local copy with a catalog entry:

```
<system systemId="http://www.oasis-open.org/docbook/xml/5.0/rng/docbook.rng"
    uri="rng/docbook.rng"/>
```

An XML catalog can be used also to map a W3C XML Schema specified with an URN in the xsi:schemaLocation attribute of an XML document to a local copy of the schema. For example if the XML document specifies the schema with:

```
<topic xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1">
```

the URN can be resolved to a local schema file with a catalog entry like:

```
<system systemId="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1"
    uri="topic.xsd"/>
```

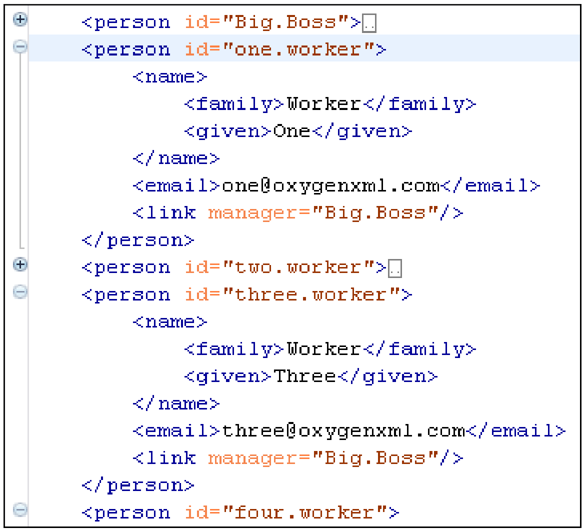
Document navigation

Navigating between XML elements located in various parts of the currently edited document is easy due to several powerful features.

Folding of the XML elements

XML documents are organized as a tree of elements. When working on a large document you can collapse some elements leaving in the focus only the ones you need to edit. Expanding and collapsing works on individual elements: expanding an element leaves the child elements unchanged.

Figure 4.30. Folding of the XML Elements



```
<person id="Big.Boss">□
<person id="one.worker">
    <name>
        <family>Worker</family>
        <given>One</given>
    </name>
    <email>one@oxygenxml.com</email>
    <link manager="Big.Boss"/>
</person>
<person id="two.worker">□
<person id="three.worker">
    <name>
        <family>Worker</family>
        <given>Three</given>
    </name>
    <email>three@oxygenxml.com</email>
    <link manager="Big.Boss"/>
</person>
<person id="four.worker">
```

To toggle the folded state of an element click on the special mark displayed in the left part of the document editor next to the start tag of that element or click on the action Toggle fold (Ctrl+Alt+Y) available from the context menu

Other menu actions related to folding of XML elements are available from the context menu of the current editor:

- Document+Folding+ → Close Other Folds (**Ctrl+NumPad+/**) Fold all the sections except the current element.

- Document+Folding+ → Collapse Child Folds : Fold the sections indented with one level inside the current element.
- Document+Folding+ → Expand Child Folds (**Ctrl+NumPad++ (Cmd+NumPad++)**): Unfold the sections indented with one level inside the current element.
- Document+Folding+ → Expand All (**Ctrl+NumPad+* (Cmd+NumPad+* on Mac OS)**): Unfold all the sections inside the current element.
- Document+Folding+ → Toggle Fold (**Alt+Shift+Y (Cmd+Alt+Y on Mac OS)**): Toggles the state of the current fold.

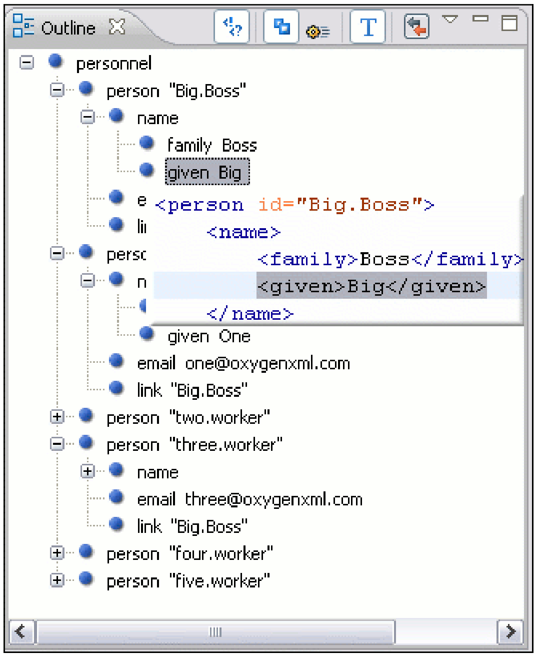
You can use folding by clicking on the special marks displayed in the left part of the document editor.

Outline View

The Outline view has the following available functions:

- the section called “XML Document Overview”
- the section called “Outliner filters”
- the section called “Modification Follow-up”
- the section called “Document Structure Change”
- the section called “Document Tag Selection”

Figure 4.31. The Outline View



```
personnel
  person "Big.Boss"
    name
      family Boss
      given Big
    e
    li
  persc
    n

      given One
    email one@oxygenxml.com
    link "Big.Boss"
  person "two.worker"
  person "three.worker"
    name
    email three@oxygenxml.com
    link "Big.Boss"
  person "four.worker"
  person "five.worker"
```

```
<person id="Big.Boss">
    <name>
        <family>Boss</family>
        <given>Big</given>
    </name>
```

XML Document Overview

The Outline view displays a general tag overview of the current edited XML Document. It also shows the correct hierarchical dependencies between the tag elements, making it easier for the user to be aware of the document's structure and the way tags are nested.

The *Expand all* and *Collapse all* items of the popup menu available on the outline tree enlarge or reduce the set of nodes of the edited document currently visible in the view. The tree expansion action is a faster alternative to mouse clicks on the plus signs of the tree when one wants to access quickly a node deeply nested in the hierarchy of document nodes. When a large number of nodes become expanded and the document structure is not clear any more the collapsing action clears the view quickly by reducing the depth of the expanded nodes to only one child of the currently selected node.

Outliner filters

Show comments/Processing Instructions

Show/Hide Comments and Processing instructions in the outliner.

Show text

Show/Hide additional text content for the displayed elements.

Show attributes

Show/Hide attribute values for the displayed elements.

The displayed attribute values can be changed from the Outline preferences panel.

The content of the Outline view can also be filtered with patterns typed in the text field of the view. The patterns can include the wildcard characters * and ?. If more than one pattern is used they must be separated by comma. Any pattern is a prefix filter, that is a * is appended automatically at the end of every pattern.

Modification Follow-up

When editing, the Outline view dynamically follows the modifications introduced by the user, showing in the middle of the panel the node which is currently being modified .This gives the user better insight on location where in the document one is positioned and how the structure of the document is affected by one's modifications.

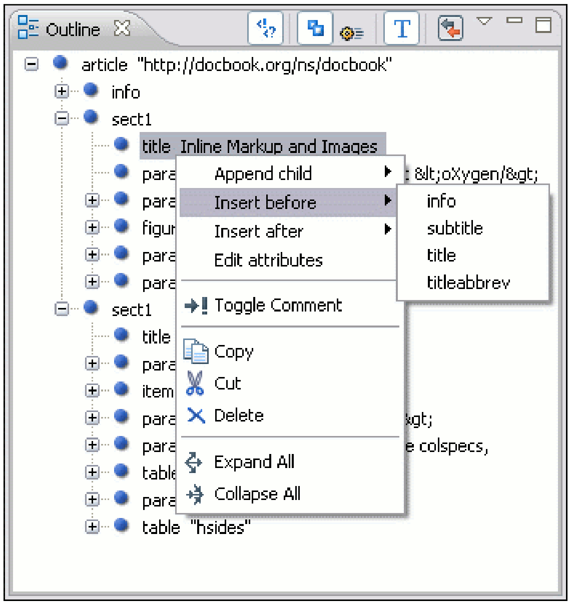
Document Structure Change

Entire XML elements can be moved or copied in the edited document using only the mouse in the Outline view in drag-and-drop operations. If you drag an XML element in the Outline view and drop it on another one in the same panel then the dragged element will be moved after the drop target element. If you hold the mouse pointer over the drop target for a short time before the drop then the drop target element will be expanded first and the dragged element will be moved inside the drop target element after its opening tag. You can also drop an element before or after another element if you hold the mouse pointer towards the upper or lower part of the targeted element. A marker will indicate whether the drop will be performed before or after the target element. If you hold down the CTRL key the performed operation will be copy instead of move.

The drag and drop action in the Outline view can be disabled and enabled from the Preferences dialog.

The popup menu of the Outline tree

Figure 4.32. Popup menu of the Outline tree



The *Append Child*, *Insert Before* and *Insert After* submenus of the outline tree popup menu allow to quickly insert new tags in the document at the place of the element currently selected in the Outline tree. The *Append Child* submenu lists the names of all the elements which are allowed by the schema associated with the current document as child of the current element. The *Insert Before* and *Insert After* submenus of the Outline tree popup menu list the elements which are allowed by the schema associated with the current document as siblings of the current element inserted immediately before respectively after the current element.

Edit attributes for the selected node. A dialog is presented allowing the user to see and edit the attributes of the selected node. See here for more details about editing attributes.

The *Toggle comment* item of the outline tree popup menu is the same item as in the editor popup menu with the same name. It encloses the currently selected element of the outline tree in an XML comment, if the element is not commented, or un comments it if it is commented.

The *Cut*, *Copy* and *Delete* items of the popup menu execute the same actions as the Edit menu items with the same name on the elements currently selected in the outline tree (Cut, Copy, Paste).

Document Tag Selection

The Outline view can also be used to search for a specific tag's location and contents in the edited document. Intuitively, by selecting with the left mouse button the desired tag in the Outline view, the document is scrolled to the position of the selected tag. Moreover, the tag's contents are selected in the document, making it easy to notice the part of the document contained by that specific tag and furthermore to easily copy and paste the tag's contents in other parts of the document or in other documents.

You can also use key search to look for a particular tag name in the Outliner tree.

Grouping documents in XML projects

Large Documents

Let's consider the case of documenting a large project. It is likely to be several people involved. The resulting document can be few megabytes in size. How to deal with this amount of data in such a way the work parallelism would not be affected ?

Fortunately, XML provides a solution for this. It can be created a master document, with references to the other document parts, containing the document sections. The users can edit individually the sections, then apply FOP or XSLT over the master and obtain the result files, let say PDF or HTML.

Two conditions must be fulfilled:

- The master should declare the DTD to be used and the external entities - the sections. A sample document is:

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE book SYSTEM "../xml/docbookx.dtd" [
  <!ENTITY testing SYSTEM "testing.xml" > ]
  >
  <book>
  <chapter> ...
  ```

  At a certain point in the master document there can be inserted the section "testing.xml" entity:

  ... &testing; ...

- The document containing the section must not define again the DTD.

  <section> ... here comes the section content ... </section>

  Note

  The indicated DTD and the element names ("section", "chapter") are used here only for illustrating the inclusion mechanism. You can use any DTD and element names you need.

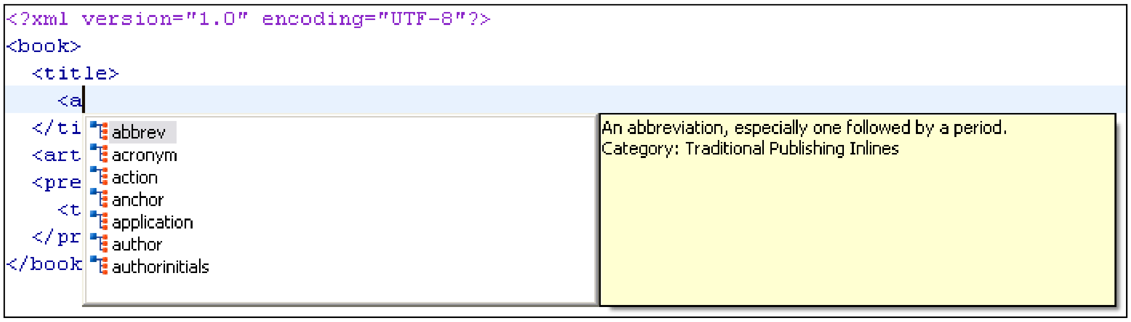
When splitting a large document and including the separate parts in the master file using external entities, only the master file will contain the Document Type Definition (the DTD) or other type of schema. The included sections can't define again the schema because the main document will not be valid. If you want to validate the parts separately you have to use XInclude for assembling the parts together with the master file.

Creating an included part

Open a new document of type XML, with no associated schema.

You can type in the edited document the root element of your section. For example, if you are using DocBook it can be "<chapter></chapter>" or "<section></section>". Now if you are moving the cursor between the tags and press "<", you will see the list of element names that can be inserted.

Figure 4.33. Content Completion list over a document with no schema



```
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <title>
    <a
  </ti
  <art
  <pre
    <t
  </pr
</book
```

An abbreviation, especially one followed by a period.
Category: Traditional Publishing Inlines

Note

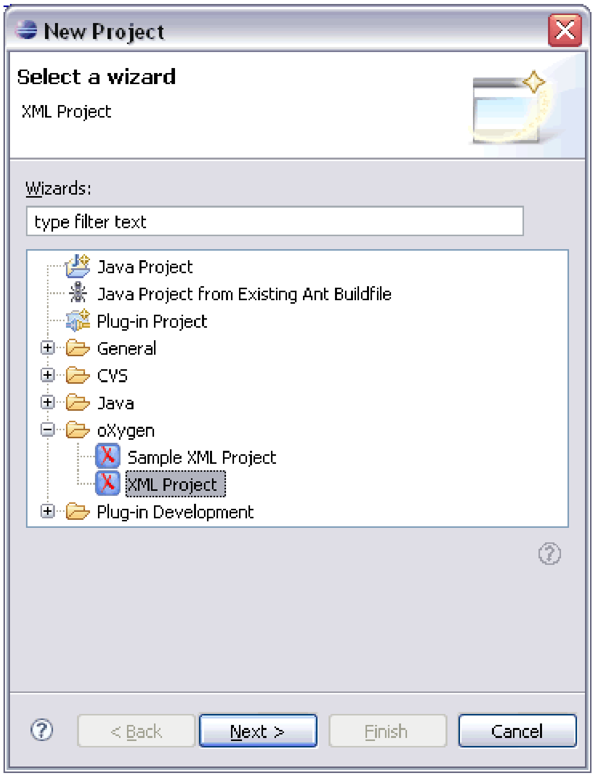
The validation will not work on a included file, as no DTD is set. The validation can be done only from the master file. At this point you can only check the document to be well-formed.

Creating a new project

Procedure 4.4. Create an <oXygen/> XML project

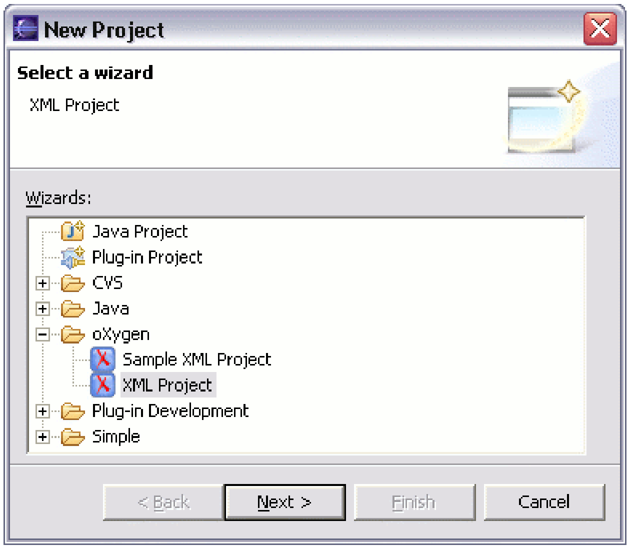
1. Select File → New → -> Other (**Ctrl+N**) or press the New toolbar button. The New wizard is displayed which contains the list entry *XML Project*.

Figure 4.34. The New wizard



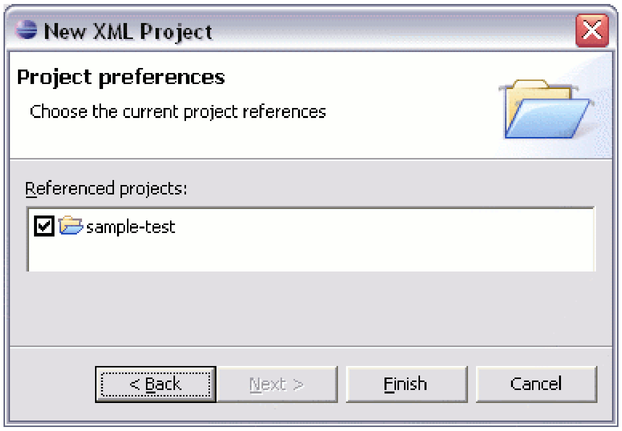
2. Select XML Project in the list of document types and click the Next button.

Figure 4.35. The XML Project wizard - step 1



3. Type a name for the new project and click the Next button.

Figure 4.36. The XML Project wizard - step 2



4. Select other Eclipse projects that you want to reference in the new project and click the Finish button.

The files are organized in a XML project usually as a collection of folders. They are created and deleted with the usual Eclipse actions.

The currently selected files associated to the <oXygen/> pluginin the Package Explorer view can be validated against a schema of type Schematron, XML Schema, Relax NG, NRL, NVDL, or a combination of the later with Schematron with one of the actions *Validate* and *Validate With* ... available on the Batch Validation submenu of the right-click menu of the Package Explorer view. This together with the logical folder support of the project allows you to group your files and validate them very easily.

If the resources from a linked folder in the project have been changed outside the view you can refresh the content of the folder by using the Refresh action from the contextual menu. The action is also performed when selecting the linked resource and pressing **F5** key

You can also use drag and drop to arrange the files in logical folders(but not in linked folders). Also, dragging and dropping files from the project tree to the editor area results in the files being opened.

Including document parts with XInclude

XInclude is a standard for assembling XML instances into another XML document through inclusion. It enables larger documents to be dynamically created from smaller XML documents without having to physically duplicate the content of the smaller files in the main file. XInclude is targeted as the replacement for External Entities. The advantage of using XInclude is that, unlike the entities method, each of the assembled documents is permitted to contain a Document Type Declaration (DOCTYPE Decl.). This means that each file is a valid XML instance and can be independently validated. It also means that the main document to which smaller instances are included can be validated without having to remove or comment the DOCTYPE Decl. as is the case with External Entities. This is makes XInclude a more convenient and effective method for managing XML instances that need to be stand-alone documents and part of a much larger work.

The main application for XInclude is in the document orientated content frameworks such as manuals and Web pages. Employing XInclude enables authors and content managers to manage content in a modular fashion that is akin to Object Orientated methods used in languages such as Java, C++ or C#.

The advantages of modular documentation include: reusable content units, smaller file units that are easier to edited, better version control and distributed authoring.

An example: create a chapter file and an article file in the `samples` folder of the <oXygen/> install folder and include the chapter file in the article file using XInclude.

Chapter file introduction.xml:

```
<?xml version="1.0"?>
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook XML V4.3//EN"
"http://www.oasis-open.org/docbook/xml/4.3/docbookx.dtd">
<chapter>
    <title>Getting started</title>
    <section>
        <title>Section title</title>
        <para>Para text</para>
    </section>
</chapter>
```

Main article file:

```
<?xml version="1.0"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.3//EN"
"http://www.docbook.org/xml/4.3/docbookx.dtd"
[ <!ENTITY % xinclude SYSTEM "../frameworks/docbook/dtd/xinclude.mod">
%xinclude;
]>
<article>
    <title>Install guide</title>
    <para>This is the install guide.</para>
    <xi:include xmlns:xi="http://www.w3.org/2001/XInclude"
                    href="introduction.xml">
      <xi:fallback>
        <para>
          <emphasis>FIXME: MISSING XINCLUDE CONTENT</emphasis>
        </para>
      </xi:fallback>
    </xi:include>
</article>
```

In this example the following is of note:

- The DOCTYPE Decl. defines an entity that references a file containing the information to add the xi namespace to certain elements defined by the DocBook DTD.
- The href attribute of the xi:include element specifies that the `introduction.xml` file will replace the xi:include element when the document is parsed.
- If the `introduction.xml` file cannot be found the parse will use the value of the xi:fallback element - a message to FIXME.

If you want to include only a fragment of other file in the master file the fragment must be contained in a tag having an *xml:id* attribute and you must use an XPointer expression pointing to the *xml:id* value. For example if the master file is:

```
<?xml version="1.0" encoding="UTF-8"?>
<?oxygen RNGSchema="test.rng" type="xml"?>
<test>
    <xi:include href="a.xml" xpointer="a1"
        xmlns:xi="http://www.w3.org/2001/XInclude"/>
</test>
```

and the `a.xml` file is:

```
<?xml version="1.0" encoding="UTF-8"?>
<test>
    <a xml:id="a1">test</a>
</test>
```

after resolving the XPointer reference the document is:

```
<?xml version="1.0" encoding="UTF-8"?>
<?oxygen RNGSchema="test.rng" type="xml"?>
<test>
    <a xml:id="a1" xml:base="a.xml">test</a>
</test>
```

The XInclude support in <oXygen/> is turned on by default. You can toggle it by going to the entry Enable XInclude processing in the menu Window → Preferences+oXygen / XML / XML Parser When enabled <oXygen/> will be able to validate and transform documents comprised of parts added using XInclude.

Working with XML Catalogs

When Internet access is not available or the Internet connection is slow the OASIS XML catalogs [http://www.oasis-open.org/committees/entity/spec.html] present in the list maintained in the XML Catalog Preferences panel will be scanned trying to map a remote system ID (at document validation) or a URI reference (at document transformation) pointing to a resource on a remote Web server to a local copy of the same resource. If a match is found then <oXygen/> will use the local copy of the resource instead of the remote one. This enables the XML author to work on his XML project without Internet access or when the connection is slow and waiting until the remote resource is accessed and fetched becomes unacceptable. Also XML catalogs make documents machine independent so that they can be shared by many developers by modifying only the XML catalog mappings related to the shared documents.

<oXygen/> supports any XML catalog file that conforms to one of:

- the OASIS XML Catalogs Committee Specification v1.1 [http://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html]
- the OASIS Technical Resolution 9401:1997 [http://www.oasis-open.org/specs/a401.htm] including the plain-text flavor described in that resolution

The version 1.1 of the OASIS XML Catalog specification introduces the possibility to map a system ID, a public ID or a URI to a local copy using only a suffix of the ID or URI used in the actual document. This is done using the new catalog elements systemSuffix [http://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html#s.systemsuffix] and uriSuffix [http://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html#s.urisuffix].

An XML catalog can be used also to map a W3C XML Schema specified with an URN in the xsi:schemaLocation attribute of an XML document to a local copy of the schema. For example if the XML document specifies the schema with:

```
<topic xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1">
```

the URN can be resolved to a local schema file with a catalog entry like:

```
<system systemId="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1"
    uri="topic.xsd"/>
```

An XML Catalog file can be created quickly in <oXygen/> starting from the two XML Catalog document templates called *OASIS XML Catalog 1.0* and *OASIS XML Catalog 1.1* and available in the document templates dialog [creating-from-templates].

User preferences related to XML Catalogs can be configured from Window → Preferences +oXygen / XML / XML Catalog

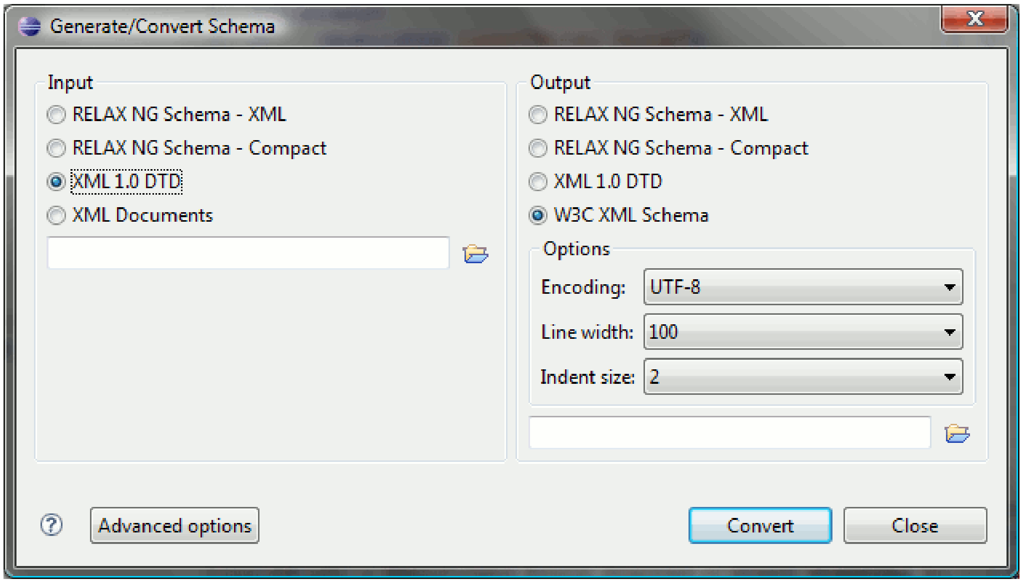
Converting between schema languages

The Generate/Convert Schema allows you to convert a DTD or Relax NG (full or compact syntax) schema or a set of XML files to an equivalent XML Schema, DTD or Relax NG (full or compact syntax) schema. Where perfect equivalence is not possible due to limitations of the target language <oXygen/> will generate an approximation of the source schema.

The conversion functionality is available from XML Tools → Generate/Convert Schema... (**Ctrl+Shift+\ (Cmd+Shift+/ on Mac OS)**) and from the toolbar button Convert to...

A schema being edited can be converted with just one click on a toolbar button if that schema can be the subject of a supported conversion.

Figure 4.37. Convert a schema to other schema language


Generate/Convert Schema

Input
○ RELAX NG Schema - XML
○ RELAX NG Schema - Compact
● XML 1.0 DTD
○ XML Documents

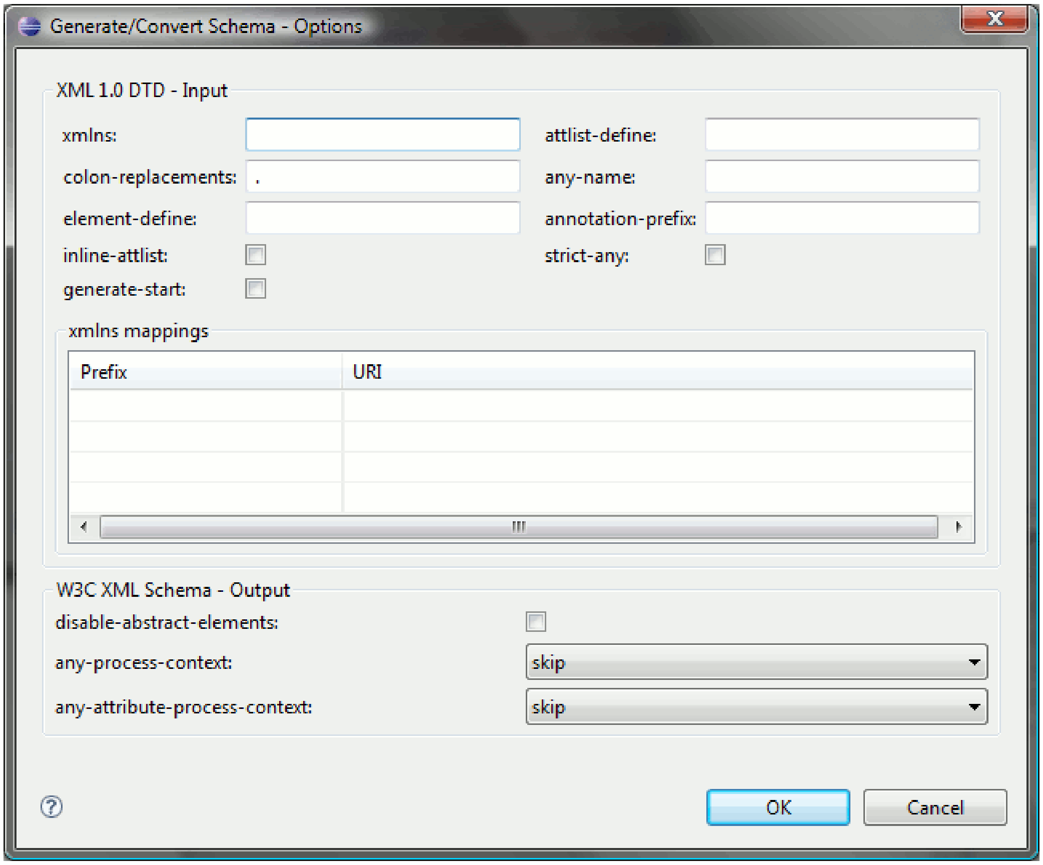
Output
○ RELAX NG Schema - XML
○ RELAX NG Schema - Compact
○ XML 1.0 DTD
● W3C XML Schema

Options
Encoding: UTF-8
Line width: 100
Indent size: 2

The language of the target schema is specified with one of the four radio buttons of the Output panel. The encoding, the maximum line width and the number of spaces for one level of indentation can be also specified in this panel.

The conversion can be further fine-tuned by specifying more advanced options available from the Advanced options button. For example if the input is a DTD and the output is an XML Schema the advanced options are:

Figure 4.38. Convert a schema to other schema language - advanced options



Generate/Convert Schema - Options

XML 1.0 DTD - Input

xmlns:

attlist-define:

colon-replacements: .

any-name:

element-define:

annotation-prefix:

inline-attlist: □

strict-any: □

generate-start: □

xmlns mappings

Prefix	URI

W3C XML Schema - Output

disable-abstract-elements: □

any-process-context: skip

any-attribute-process-context: skip

For the Input panel:

xmlns field

specifies the default namespace, that is the namespace used for unqualified element names.

xmlns table

Each row specifies in the prefix used for a namespace in the input schema.

colon-replacement

Replaces colons in element names by the specified chars when constructing the names of definitions used to represent the element declarations and attribute list declarations in the DTD.

element-define

Specifies how to construct the name of the definition representing an element declaration from the name of the element. The specified value must contain exactly one percent character. This percent character is replaced by the name of element (after colon replacement) and the result is used as the name of the definition.

inline-attlist

Specifies not to generate definitions for attribute list declarations and instead move attributes declared in attribute list declarations into the definitions generated for element declarations. This is the default behavior when the output language is XSD.

attlist-define

This specifies how to construct the name of the definition representing an attribute list declaration from the name of the element. The specified value must contain exactly one percent character. This percent character is replaced by the name of element (after colon replacement) and the result is used as the name of the definition.

any-name
: Specifies the name of the definition generated for the content of elements declared in the DTD as having a content model of ANY.

strict-any
: Preserves the exact semantics of ANY content models by using an explicit choice of references to all declared elements. By default, Trang uses a wildcard that allows any element.

generate-start
: Specifies whether Trang should generate a start element. DTDs do not indicate what elements are allowed as document elements. Trang assumes that all elements that are defined but never referenced are allowed as document elements.

annotation-prefix
: Default values are represented using an annotation attribute *prefix:defaultValue* where prefix is the specified value and is bound to http://relaxng.org/ns/compatibility/annotations/1.0 as defined by the RELAX NG DTD Compatibility Committee Specification. By default, Trang will use a for prefix unless that conflicts with a prefix used in the DTD.

For the Output panel:

disable-abstract-elements
: Disables the use of abstract elements and substitution groups in the generated XML Schema. This can also be controlled using an annotation attribute.

any-process-contents
: One of the values: strict, lax, skip. Specifies the value for the processContents attribute of any elements. The default is skip (corresponding to RELAX NG semantics) unless the input format is dtd, in which case the default is strict (corresponding to DTD semantics).

any-attribute-process-contents
: Specifies the value for the processContents attribute of anyAttribute elements. The default is skip (corresponding to RELAX NG semantics).

Formatting and indenting documents (pretty print)

In structured markup languages, the whitespace between elements that is created by use of the **Space bar**, **Tab** or multiple line breaks insertion from use of the **Enter**, is not recognized by the parsing tools. Often this means that when structured markup documents are opened, they are arranged as one long, unbroken line, what seems to be a single paragraph.

While this is perfectly acceptable practice, it makes editing difficult and increases the likelihood of errors being introduced. It also makes the identification of exact error positions difficult. Formatting and Indenting, also called Pretty Print, enables such documents to be neatly arranged, in a manner that is consistent and promotes easier reading on screen and in print output.

Pretty print is in no way associated with the layout or formatting that will be used in the transformed document. This layout and formatting is supplied by the XSL stylesheet specified at the time of transformation.

Procedure 4.5. To format and indent a document:

1. Open or focus on the document that is to be formatted and indented.

2. Select menu XML → Format and Indent (**Ctrl+Shift+F (Cmd+Shift+F on Mac OS)**) or click the toolbar button Format and indent . While in progress the Status Panel will indicate Pretty print in progress. On completion, this will change to Pretty print successful and the document will be arranged.

Note

Pretty Print can format empty elements as an auto-closing markup tag (ex. <a/>) or as a regular tag (ex. <a></a>). It can preserve the order or attributes or order them alphabetically. Also the user may specify a list of elements for which white spaces are preserved exactly as before Pretty print and one with elements for which white space is stripped. These can be configured from Options → Preferences+Editor / Format.

Pretty Print requires that the structured document is *Well-Formed XML*. If the document is not *Well-Formed XML* an error message is displayed. The message will usually indicate that a problem has been found in the form and will hint to the problem type. It will not highlight the general position of the error, to do this run the *well formed* action by selecting Document → Check document form (**Alt+Shift+V W (Cmd+Alt+V W on Mac OS)**).

Important

In XHTML files (XML files which either have the XHTML namespace or the <html> root tag) the JavaScript <script> sections will be formatted according to the JavaScript Format and Indent options and the CSS <style> sections will be formatted according to the CSS Format and Indent options.

Note

If the document is not well-formed because some XML elements contain code in a specific language, for example JavaScript:

```
  <script language="JavaScript" type="text/javascript">
    var javawsInstalled = 0;
    var javaws12Installed = 0;
    var javaws142Installed=0;
    isIE = "false";

    if (navigator.mimeTypes && navigator.mimeTypes.length) {
      x = navigator.mimeTypes['application/x-java-jnlp-file'];
      if (x) {
        javawsInstalled = 1;
        javaws12Installed=1;
        javaws142Installed=1;
      }
    } else {
      isIE = "true";
    }
</script>
```

this code can be enclosed in an XML comment to make the document well-formed before applying the *Format and Indent* action:

```
 <script language="JavaScript" type="text/javascript">
  <!--
    var javawsInstalled = 0;
    var javaws12Installed = 0;
    var javaws142Installed=0;
    isIE = "false";
```

```
      if (navigator.mimeTypes && navigator.mimeTypes.length) {
        x = navigator.mimeTypes['application/x-java-jnlp-file'];
        if (x) {
          javawsInstalled = 1;
          javaws12Installed=1;
          javaws142Installed=1;
        }
      } else {
        isIE = "true";
      }
    -->
  </script>
```

To change the indenting of the current selected text see the action Indent selection .

For user preferences related to formatting and indenting like Detect indent on open and Indent on paste see the corresponding Preferences panel.

XML elements can be excepted from the reformatting performed by the pretty-print operation by including them in the *Preserve space elements (XPath)* list. That means that when the *Format and Indent* (pretty-print) action encounters in the document an element with the name contained in this list the whitespace is preserved inside that element. This is useful when most of the elements must be reformatted with the exception of a few ones which are listed here.

For the situation when whitespace should be preserved in most elements with the exception of a few elements, the names of these elements must be added to the *Strip space elements (XPath)* list.

In addition to simple element names both the *Preserve space elements (XPath)* list and the *Strip space elements (XPath)* one accept a restricted set of XPath expressions for covering a pattern of XML elements with only one expression. The allowed types of expressions are:

//xs:documentation
: the XPath descendant axis can be used only at the beginning of the expression; the namespace prefix can be attached to any namespace, no namespace binding check is performed when applying the pretty-print operation

/chapter/abstract/title
: note the use of the XPath child axis

//section/title
: the descendant axis can be followed by the child axis

The value of an *xml:space* attribute present in the XML document on which the pretty-print operation is applied always takes precedence over the *Preserve space elements (XPath)* and the *Strip space elements (XPath)* lists.

Viewing status information

Status information generated by the Schema Detection, Validation, Validate as you type and Transformation threads are fed into the Console view allowing the user to monitor how the operation is being executed.

Messages contain a timestamp, the name of the thread that generated it and the actual status information. The number of displayed messages in the console view can be controlled from the options panel.

XML editor specific actions

<oXygen/> offers groups of actions for working on single XML elements. They are available from the the context menu of the main editor panel. On Windows the context menu can be displayed with the mouse on a right click or with the keyboard by pressing the special context menu key available on Windows keyboards.

Edit actions

- : Turns on line wrapping in the editor panel if it was off and vice versa. It has the same effect as the Line wrap preference.
- contextual menu of current editor → Toggle comment (**Ctrl + /**): Comment the current selection of the current editor. If the selection already contains a comment the action removes the comment from around the selection. If there is no selection in the current editor and the cursor is not positioned inside a comment the current line is commented. If the cursor is positioned inside a comment then the commented text is uncommented.

Select actions

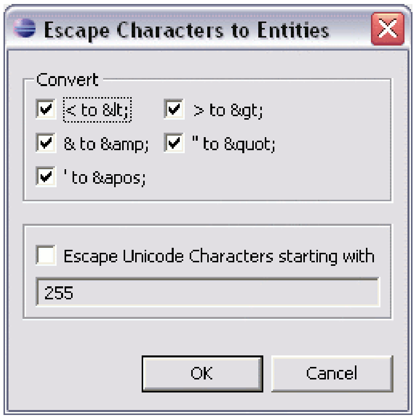
The Select actions are enabled when the caret is positioned inside a tag name.

- contextual menu of current editor+Select → Element: Selects the entire current element;
- contextual menu of current editor+Select → Content: Selects the content of the current element, excluding the start tag and end tag;
- contextual menu of current editor+Select → Attributes: Selects all the attributes of the current element;
- contextual menu of current editor+Select → Parent: Selects the parent element of the current element;

Source actions

- contextual menu of current editor+Source+Escape Selection ... : Escapes a range of characters by replacing them with the corresponding character entities.

Figure 4.39. Escape selection



Escape Characters to Entities

Convert

☑ < to &lt;   ☑ > to &gt;

☑ & to &amp;   ☑ " to &quot;

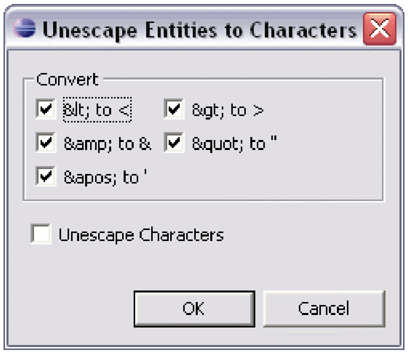
☑ ' to &apos;

☐ Escape Unicode Characters starting with

255

- contextual menu of current editor+Source+Unescape Selection ... : Replaces the character entities with the corresponding characters;

Figure 4.40. Unescape selection



Unescape Entities to Characters

Convert

☑ &lt; to < ☑ &gt; to >

☑ &amp; to & ☑ &quot; to "

☑ &apos; to '

☐ Unescape Characters

- contextual menu of current editor+Source+Indent selection (**Ctrl + I (Cmd + I on Mac OS)**):Corrects the indentation of the selected block of lines.

- contextual menu of current editor+Source+Format and Indent Element (**Ctrl + I**): Pretty prints the element that surrounds the caret position;

- contextual menu of current editor+Source+Import entities list : Shows a dialog that allows you to select a list of files as sources for external entities. The DOCTYPE section of your document will be updated with the chosen entities. For instance, if choosing the file chapter1.xml, and chapter2.xml, the following section is inserted in the DOCTYPE:

  ```
  <!ENTITY chapter1 SYSTEM "chapter1.xml">
  ```

  ```
  <!ENTITY chapter2 SYSTEM "chapter2.xml">
  ```

- Double click on an element or processing instruction - If the double click is done before the start tag of an element or after the end tag of an element then all the element is selected by the double click action. If it is done after the start tag or before the end tag then only the element content without the start tag and end tag is selected.

- contextual menu of current editor → Join and normalize: The action works on the selection. It joins the lines by replacing the *line separator* with a single space character. It also normalizes the whitespaces by replacing a sequence of such characters with a single space.

XML document actions

- contextual menu of current editor → Show Definition : move the cursor to the definition of the current element in the schema associated with the edited XML document (DTD, XML Schema, Relax NG schema, NRL schema).

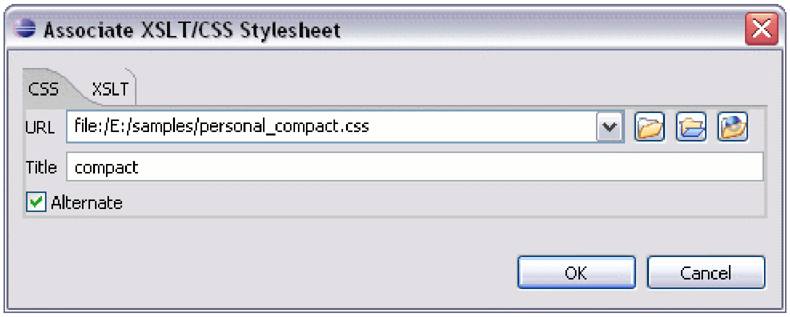
- contextual menu of current editor → Copy XPath (**Ctrl+Shift+.**): Copy XPath expression of current element from current editor to clipboard.

- contextual menu of current editor+Go to the matching tag : Moves the cursor to the end tag that matches the start tag, or vice versa.

- contextual menu of current editor → Go after Next Tag (**Ctrl+Close Bracket**): Moves the cursor to the end of the next tag.

- contextual menu of current editor → Go after Previous Tag (**Ctrl+Open Bracket**): Moves the cursor to the end of the previous tag.
- XML+Associate XSLT/CSS Stylesheet : Inserts an *xml-stylesheet* processing instruction at the beginning of the document referencing either an XSLT or a CSS file depending on the user selection. Either reference is useful for rendering the document in a Web browser when the action *Open in browser* is executed. Referencing the XSLT file is also useful for automatic detection of the transformation stylesheet when there is no scenario associated with the current document

Figure 4.41. Associate XSLT/CSS stylesheet dialog



Associate XSLT/CSS Stylesheet

URL: file:/E:/samples/personal_compact.css

Title: compact

☑ Alternate

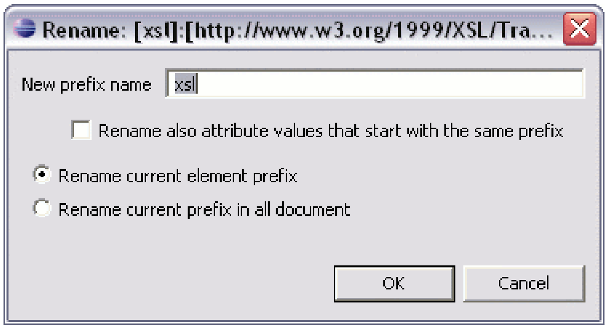
When associating the CSS, the user can also specify the title and if the stylesheet is an alternate one. Setting a **Title** for the CSS makes it the author's preferred stylesheet. Checking the **Alternate** checkbox makes the CSS an alternate stylesheet.

oXygen Author fully implements the W3C recommendation regarding "Associating Style Sheets with XML documents". For more information see: http://www.w3.org/TR/xml-stylesheet/http://www.w3.org/TR/REC-html40/present/styles.html#h-14.3.2

XML Refactoring actions

- context menu of current editor+XML Refactoring+Surround with tag... (**Alt+Shift+E (Cmd+Alt+E on Mac OS)**): Selected Text in the editor is marked with the specified start and end tags.
- context menu of current editor+XML Refactoring+Surround with last <tag> (**Alt+Shift+/ (Cmd+Alt+/ on Mac OS)**): Selected Text in the editor is marked with start and end tags of the last 'Surround in' action.
- context menu of current editor+XML Refactoring+Rename element (**Alt+Shift+R (Cmd+Alt+R on Mac OS)**): The element from the caret position and the elements that have the same name as the current element can be renamed according with the options from the Rename dialog.
- context menu of current editor+XML Refactoring+Rename prefix : The prefix of the element from the caret position and the elements that have the same prefix as the current element can be renamed according with the options from the Rename dialog.

Figure 4.42. Rename Prefix Dialog



Rename: [xsl]:[http://www.w3.org/1999/XSL/Tra...

New prefix name: xsl

□ Rename also attribute values that start with the same prefix

● Rename current element prefix

○ Rename current prefix in all document

Selecting the *Rename current element prefix* option the application will recursively traverse the current element and all its children.

For example, to change the xmlns:p1="ns1" association existing in the current element to xmlns:p5="ns1" just select this option and press OK. If the association xmlns:p1="ns1" is applied on the parent of the current element, then <oXygen/> will introduce a new declaration xmlns:p5="ns1" in the current element and will change the prefix from p1 to p5. If p5 is already associated in the current element with another namespace, let's say ns5, then a dialog showing the conflict will be displayed. Pressing the OK button, the prefix will be modified from p1 to p5 without inserting a new declaration xmlns:p5="ns1". On Cancel no modification is made.

Selecting the "Rename current prefix in all document" option the application will apply the change on the entire document.

To apply the action also inside attribute values one must check the *Rename also attribute values that start with the same prefix* checkbox.

- context menu of current editor+XML Refactoring+Split element : Split the element from the caret position in two identical elements. The caret must be inside the element
- context menu of current editor+XML Refactoring+Join elements (**Alt+Shift+F (Cmd+Alt+F on Mac OS)**): Joins the left and the right elements relative to the current caret position. The elements must have the same name, attributes and attributes values.
- context menu of current editor+XML Refactoring+Delete element tags (**Alt+Shift+, (Cmd+Alt+, on Mac OS)**): Deletes the start tag and end tag of the current element.

Smart editing

Closing tag auto-expansion

If you want to insert content into an auto closing tag like <tag/> deleting the / character saves some keystrokes by inserting a separate closing tag automatically and placing the cursor between the start and end tags: <tag></tag>

Auto-rename matching tag

When you edit the name of the start tag, <oXygen/> will mirror-edit the name of the matching end tag. This feature can be controlled from the *Content Completion* option page.

Auto-breaking the edited line

The *Hard line wrap* option breaks the edited line automatically when its length exceeds the maximum line length defined for the pretty-print operation.

Indent on Enter

The *Indent on Enter* option indents the new line inserted when Enter is pressed.

Smart Enter

The *Smart Enter* option inserts an empty line between the start and end tags and places the cursor in an indented position on the empty line automatically when the cursor is between the start and end tag and Enter is pressed.

Triple click

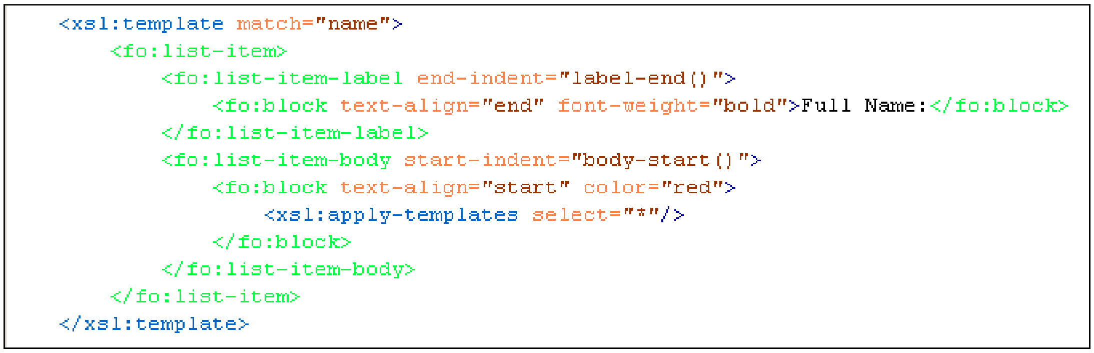
A triple click with the left mouse button selects a different region of text of the current document depending on the position of the click in the document:

- if the click position is inside a start tag or an end tag then the entire element enclosed by that tag is selected
- if the click position is immediately after a start tag or immediately before an end tag then the entire content of the element enclosed by that tag is selected, including all the child elements but excluding the start tag and the end tag of the element
- otherwise the triple click selects the entire current line of text

Syntax highlight depending on namespace prefix

The syntax highlight scheme of an XML file type allows the configuration of a color per each type of token which can appear in an XML file. Distinguishing between the XML tag tokens based on the namespace prefix brings additional visual help in editing some XML file types. For example in XSLT stylesheets elements from different namespaces like XSLT, XHTML, XSL:FO or XForms are inserted in the same document and the editor panel can become cluttered. Marking tags with different colors based on the namespace prefix allows easier identification of the tags.

Figure 4.43. Example of coloring XML tags by prefix



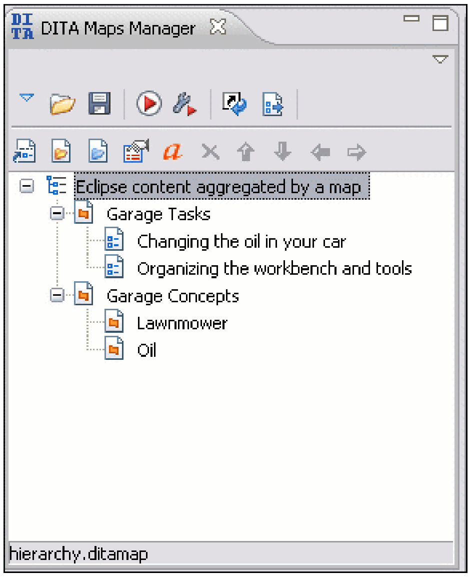
```
<xsl:template match="name">
    <fo:list-item>
        <fo:list-item-label end-indent="label-end()">
            <fo:block text-align="end" font-weight="bold">Full Name:</fo:block>
        </fo:list-item-label>
        <fo:list-item-body start-indent="body-start()">
            <fo:block text-align="start" color="red">
                <xsl:apply-templates select="*"/>
            </fo:block>
        </fo:list-item-body>
    </fo:list-item>
</xsl:template>
```

Editing DITA Maps

DITA Maps organize topics for output to a specific deliverable, including generating navigation files and links to related topics. <oXygen/> provides a special DITA Maps editing view.

The *DITA Maps Manager* view presents maps in a simplified table-of-contents manner allowing the user to easily navigate the referred topics, make changes and perform transformations to various output formats using the DITA-OT framework bundled with <oXygen/>.

Figure 4.44. The DITA Maps Manager view


DITA Maps Manager
Eclipse content aggregated by a map
Garage Tasks
Changing the oil in your car
Organizing the workbench and tools
Garage Concepts
Lawnmower
Oil
hierarchy.ditamap

You can open a map file from Project in the DITA Maps Manager view by right clicking it and choosing *Open in DITA Maps Manager*. The titles of the referenced resources will be resolved dynamically when navigating the tree. After the map is opened in the Manager you can open it in the main editor for further customization using the *Open map in editor* toolbar action.

Tip

If you want to transform your DITA topics to various formats using the DITA Open Toolkit you can open them in the DITA Maps Manager view using the "Open" button located on the internal toolbar and transform them from here.

Note

A map opened from WebDAV can be locked when it is opened in DITA Maps Manager by checking the option Lock WebDAV files on open to protect it from concurrent modifications on the server by other users. If other user tries to edit the same map he will receive an error message and the name of the lock owner. The lock is released automatically when the map is closed from <oXygen/> DITA Maps Manager.

The following general actions can be performed on an opened DITA Map :

Open
: Allows opening the DITA Map in the DITA Maps Manager view. You can also open a DITA Map by dragging it in the DITA Maps Manager from the file system explorer.

Open URL
: Allows opening remote DITA Maps in the DITA Maps Manager view. See Open URL for details.

Save
: Allows saving the currently opened DITA Map.

Apply Transformation Scenario
: Allows the user to start the DITA ANT Transformation scenario associated with the opened map. For more transformation details see here.

Configure Transformation Scenario

Allows the user to configure a DITA ANT Transformation scenario for the opened map. For more transformation details see here.

Refresh References

Sometimes after a topic was edited and its title changed the topic's title needs to be also updated in the DITA Maps manager view. You can use this action to refresh and update titles for all referred topics.

Open map in editor

For complex operations which cannot be performed in the simplified DITA Maps view (like editing a relationship table) you can open the map in the main editing area. See more about editing a map in the main edit area here.

Tip

The additional edit toolbar can be shown by clicking the "Show/Hide additional toolbar" expand button located on the general toolbar.

The following edit actions can be performed on an opened DITA Map:

Insert Topic Reference

Inserts a reference to a topic file. See more about this action here.

Insert Topic Heading

Inserts a topic heading. See more about this action here

Insert Topic Group

Inserts a topic group. See more about this action here.

Edit properties

Edit the properties of a selected node. See more about this action here.

Edit other attributes

Edits all the attributes of a selected node. A dialog is presented allowing the user to see and edit the attributes of the selected node. See here for more details about editing attributes.

Delete

Deletes the selected nodes.

Move Up

Moves the selected nodes in front of their respective previous siblings.

Move Down

Moves the selected nodes after their next respective siblings.

Promote

Moves the selected nodes after their respective parents as a siblings.

Demote

Moves the selected nodes as children to their respective previous siblings.

The contextual menu contains, in addition to the edit actions described above, the following actions:

Open in editor

Open in the editor the resources referred by the selected nodes

Cut, Copy, Paste, Undo, Redo

Common edit actions with the same functionality as those found in the text editor

Paste before, Paste after

Will paste the content of the clipboard before respectively after the selected node.

Append Child/Insert After

Topic reference

Append/Insert a topic reference as a child/sibling of the selected node

	Topic reference to the current edited file	Append/Insert a topic reference to the current edited file as a child/sibling of the selected node
	Topic heading	Append/Insert a topic heading as a child/sibling of the selected node
	Topic group	Append/Insert a topic group as a child/sibling of the selected node

You can also arrange the nodes by dragging and dropping one or more nodes at a time. Drop operations can be performed before, after or as child of the targeted node. The relative location of the drop is indicated while hovering the mouse over a node before releasing the mouse button for the drop.

Drag and drop operations allow you to:

Copy
Select the nodes you want to copy and start dragging them. Before dropping them in the appropriate place, press and hold the **CTRL** key(**META** key on Mac). The mouse pointer should change to indicate that a copy operation will be performed.

Move
Select the nodes you want to move and drag and drop them in the appropriate place.

Promote / Demote
You can move nodes between child and parent nodes which ensures both *Promote* and *Demote* operations.

Tip

You can open and edit linked topics easily by double clicking the references or by right-clicking and choosing "Open in editor". If the referenced file does not exist you will be allowed to create it.

By right clicking the map root element you can open and edit it in the main editor area for more complex operations.

You can decide to open the reference directly in the Author page and keep this setting as a default.

Note

Some of the common actions from the main application menu/toolbar also apply to the DITA Maps Manager when it has focus. These actions are:

File actions	Save, Save As, Save to URL, Save All, Print, Print preview, Close, Close others, Close all
Edit actions	Undo, Redo, Cut, Copy, Paste, Delete

The *Save all* action applies to all editors opened in either <oXygen/> work area or the DITA Maps Manager.

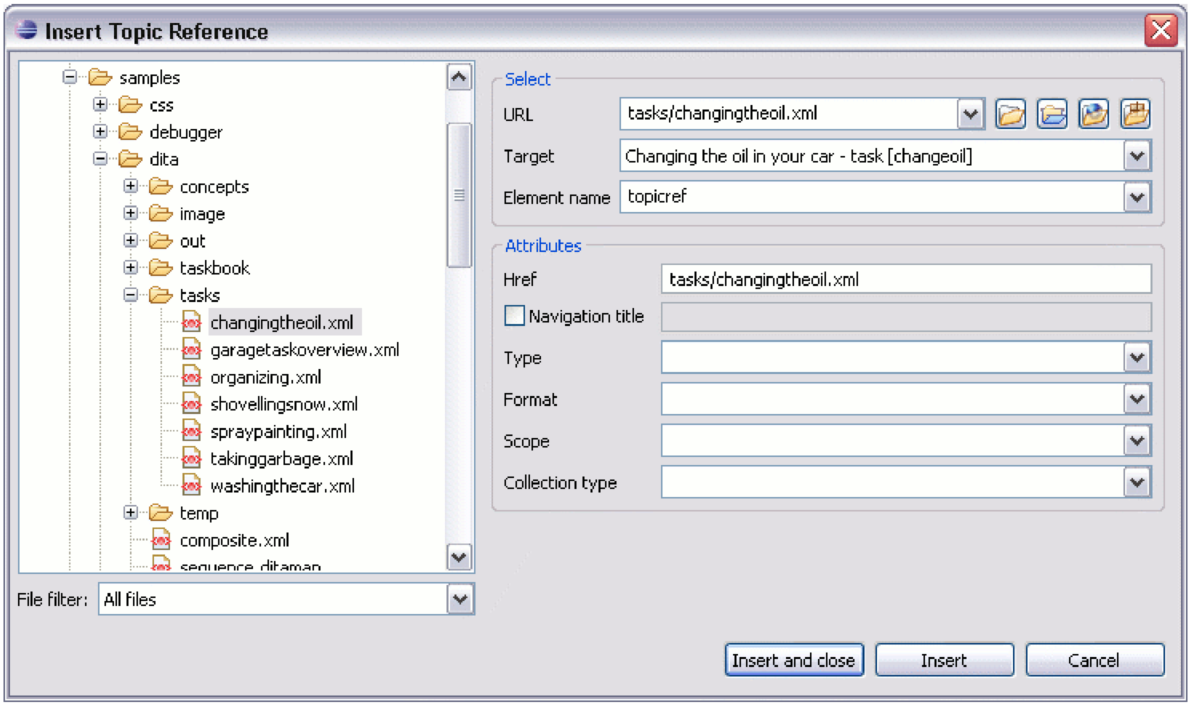
Advanced operations

Inserting a Topic Reference

The *topicref* element identifies a topic (such as a concept, task, or reference) or other resource. A *topicref* can contain other *topicref* elements, allowing you to express navigation or table-of-contents hierarchies, as well as implying relationships between the containing *topicref* and its children. You can set the collection-type of a container *topicref* to determine how its children are related to each other. You can also express relationships among *topicref*'s using group and table structures (using *topicgroup* and *reltable*). Relationships end up expressed as links in the output (with each participant in a relationship having links to the other participants by default).

A reference to a topic file may be inserted both from the toolbar action and the contextual node actions. The same dialog can be used to insert references to maps or links to non-dita files like pdf's.

Figure 4.45. Insert Topic Reference Dialog



Insert Topic Reference

Select

URL: tasks/changingtheoil.xml

Target: Changing the oil in your car - task [changeoil]

Element name: topicref

Attributes

Href: tasks/changingtheoil.xml

□ Navigation title

Type:

Format:

Scope:

Collection type:

File filter: All files

By using the *Insert Topic Reference* Dialog you can easily browse for and select the source topic file. The *Target* combo box shows all available topics that can be targeted in the file. Selecting a target modifies the *Href* value to point to it. The *Format* and *Scope* combos are automatically filled based on the selected file. You can specify and enforce a custom navigation title by checking the *Navigation title* checkbox and entering the desired title.

The file chooser located in the dialog allows you to easily select the desired topic. The selected topic file will be added as a child/sibling of the current selected topic reference. You can easily insert multiple topic references by keeping the dialog opened and changing the selection in the DITA Maps Manager tree. You can also select multiple resources in the file explorer and then insert them all as topic references.

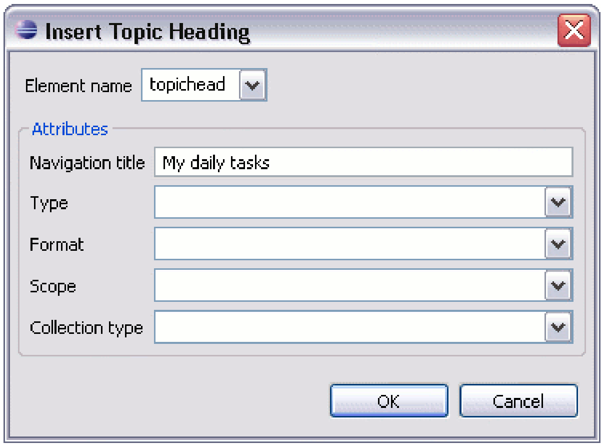
Another easy way to insert a topic reference is to directly drag and drop topic files from the Oxygen Project or the Explorer right in the DITA Maps tree.

Inserting a Topic Heading

The *topichead* element provides a title-only entry in a navigation map, as an alternative to the fully-linked title provided by the *topicref* element.

A topic heading can be inserted both from the toolbar action and the contextual node actions.

Figure 4.46. Insert Topic Heading Dialog



Insert Topic Heading

Element name: topichead

Attributes

Navigation title: My daily tasks

Type:

Format:

Scope:

Collection type:

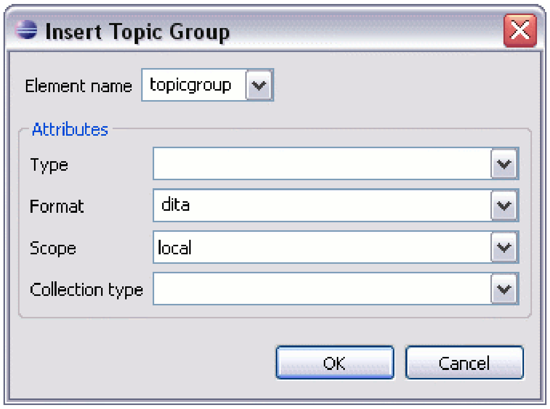
By using the *Insert Topic Heading* Dialog you can easily insert a *topichead* element. The *Navigation title* is required but other attributes can be specified as well from the dialog.

Inserting a Topic Group

The *topicgroup* element identifies a group of topics (such as a concepts, tasks, or references) or other resources. A *topicgroup* can contain other *topicgroup* elements, allowing you to express navigation or table-of-contents hierarchies, as well as implying relationships between the containing *topicgroup* and its children. You can set the collection-type of a container *topicgroup* to determine how its children are related to each other. Relationships end up expressed as links in the output (with each participant in a relationship having links to the other participants by default).

A topic group may be inserted both from the toolbar action and the contextual node actions.

Figure 4.47. Insert Topic Group Dialog



Insert Topic Group

Element name: topicgroup

Attributes

Type:

Format: dita

Scope: local

Collection type:

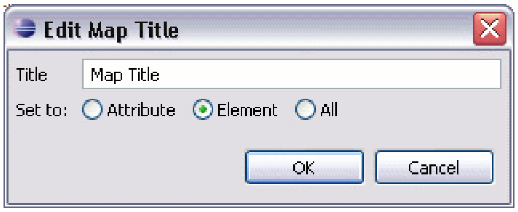
By using the *Insert Topic Group* Dialog you can easily insert a *topicgroup* element. The *Type*, *Format*, *Scope* and *Collection type* attributes can be specified from the dialog.

Edit properties

The *Edit properties* action, available both on the toolbar and on the contextual menu, is used to edit the properties of the selected node. Depending on the selected node, the action will perform the following tasks:

- If a *topicref* element is selected, the action will show a dialog similar with the Insert Topic Reference dialog allowing the editing of some important attributes.
- If a *topichead* element is selected, the action will show a dialog similar with the Insert Topic Heading dialog allowing the editing of some important attributes.
- If a *topicgroup* element is selected, the action will show a dialog similar with the Insert Topic Group dialog allowing the editing of some important attributes.
- If the map's root element is selected then the user will be able to easily edit the map's title using the *Edit Map title* dialog:

Figure 4.48. Edit Map Title Dialog



Title: Map Title

Set to: ○ Attribute ● Element ○ All

By using this dialog you can also specify whether the title will be specified as the *title* attribute to the map or as a *title* element (for DITA-OT 1.1) or specified in both locations.

Transforming DITA Maps

<oXygen/> uses the DITA Open Toolkit (DITA-OT) to transform XML content into an output format. For this purpose both the DITA Open Toolkit 1.4.3 and ANT 1.7 come bundled in <oXygen/>.

Available Output Formats

You can publish DITA-based documents in any of the following formats:

XHTML	DITA Map to XHTML
PDF - DITA OT	DITA Map to PDF using the DITA OT default PDF target
PDF2 - IDIOM FO Plugin	DITA Map to PDF using the DITA OT IDIOM PDF plugin
HTML Help (CHM)	DITA Map to HTML Help. If you have installed HTML Help Workshop, <oXygen/> detect and use this. At the end of the transformation the chm file will be open. Otherwise an error message is displayed, but the hhp(HTML Help Project) file will be generated and you have to compile to obtain a chm file. Note that HTML Help Workshop fails when the files used for transformation contains diacritics in name. This problem appear because the hhp file is generated with UTF8 encoding, and the hhc file read this file with another encoding.
JavaHelp	DITA Map to JavaHelp
Eclipse Help	DITA Map to Eclipse Help
Eclipse Content	DITA Map to Eclipse Content
RTF	DITA Map to Rich Text Format

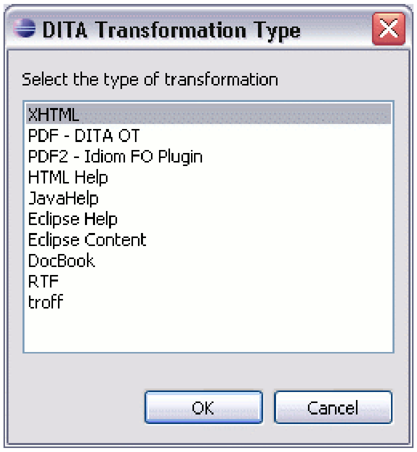
TROFF	DITA Map to Text Processor for Typesetters
Docbook	DITA Map to Docbook

Configuring a DITA transformation

Creating DITA Map transformation scenarios is similar to creating scenarios in the main editing area. See here for more details.

When creating a new scenario you can choose the type of output the DITA-OT ANT scenario will generate:

Figure 4.49. Select DITA Transformation type



Depending on the chosen type of output <oXygen/> will generate values for the default ANT parameters so that you can execute the scenario right away without further customization.

Tip

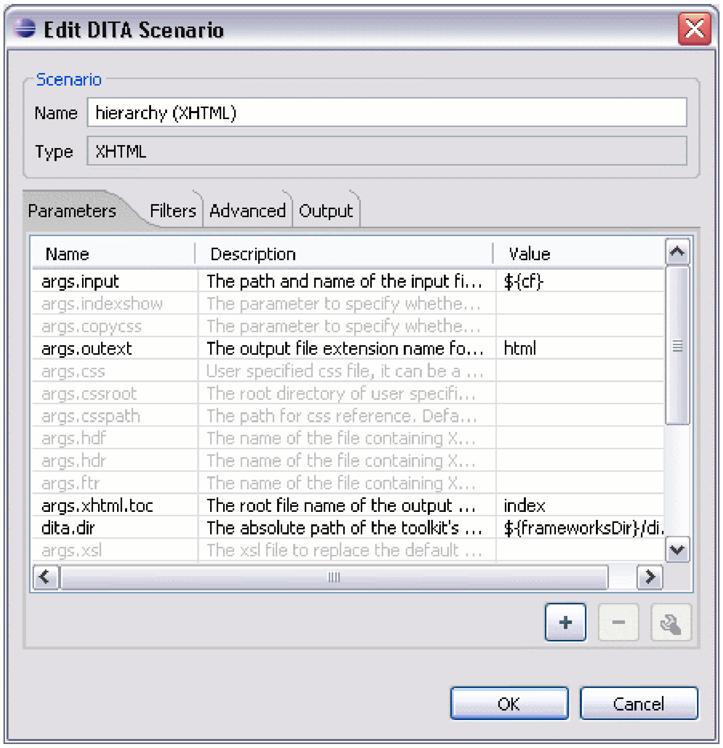
If you want to transform your DITA topics to various formats using the DITA Open Toolkit you can open them in the DITA Maps Manager view using the "Open" button located on the internal toolbar and transform them from here.

Customizing the DITA scenario

The *Parameters* tab

In the Scenario Edit Parameters Tab you can customize all the parameters which will be sent to the DITA-OT build file.

Figure 4.50. Edit DITA Ant transformation parameters



Name: hierarchy (XHTML)

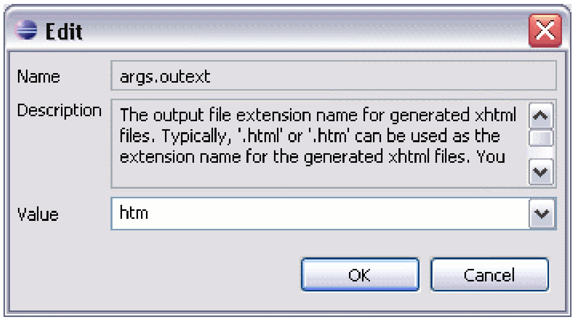
Type: XHTML

Name	Description	Value
args.input	The path and name of the input fi...	${cf}
args.indexshow	The parameter to specify whethe...	
args.copycss	The parameter to specify whethe...	
args.outext	The output file extension name fo...	html
args.css	User specified css file, it can be a ...	
args.cssroot	The root directory of user specifi...	
args.csspath	The path for css reference. Defa...	
args.hdf	The name of the file containing X...	
args.hdr	The name of the file containing X...	
args.ftr	The name of the file containing X...	
args.xhtml.toc	The root file name of the output ...	index
dita.dir	The absolute path of the toolkit's ...	${frameworksDir}/di.
args.xsl	The xsl file to replace the default ...	

All the parameters that can be set to the DITA-OT build files for the chosen type of transformation (eg: XHTML) are listed along with their description. The values for some important parameters are already filled in. You can find more information about each parameter in the DITA OT Documentation [http://dita-ot.sourceforge.net/doc/DITA-antscript.html]

Using the toolbar buttons you can Add, Edit or Remove a parameter.

Figure 4.51. Edit DITA parameters dialog



Name: args.outext

Description: The output file extension name for generated xhtml files. Typically, '.html' or '.htm' can be used as the extension name for the generated xhtml files. You

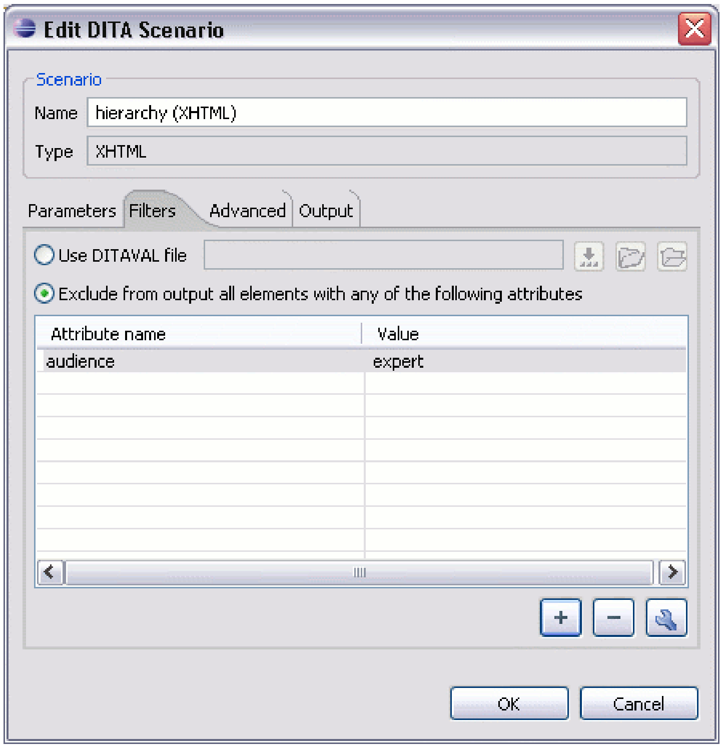
Value: htm

Depending on the parameter type the parameter value will be a simple text field for simple parameter values, a combo box with some predefined values or will have a file chooser and an editor variables selector to simplify setting a file path as value to a parameter.

The *Filters* tab

In the Scenario Filters Tab you can add filters to remove certain content elements from the generated output.

Figure 4.52. Edit Filters tab



Edit DITA Scenario

Scenario

Name: hierarchy (XHTML)

Type: XHTML

○ Use DITAVAL file

● Exclude from output all elements with any of the following attributes

Attribute name	Value
audience	expert

You have two ways in which to define filters:

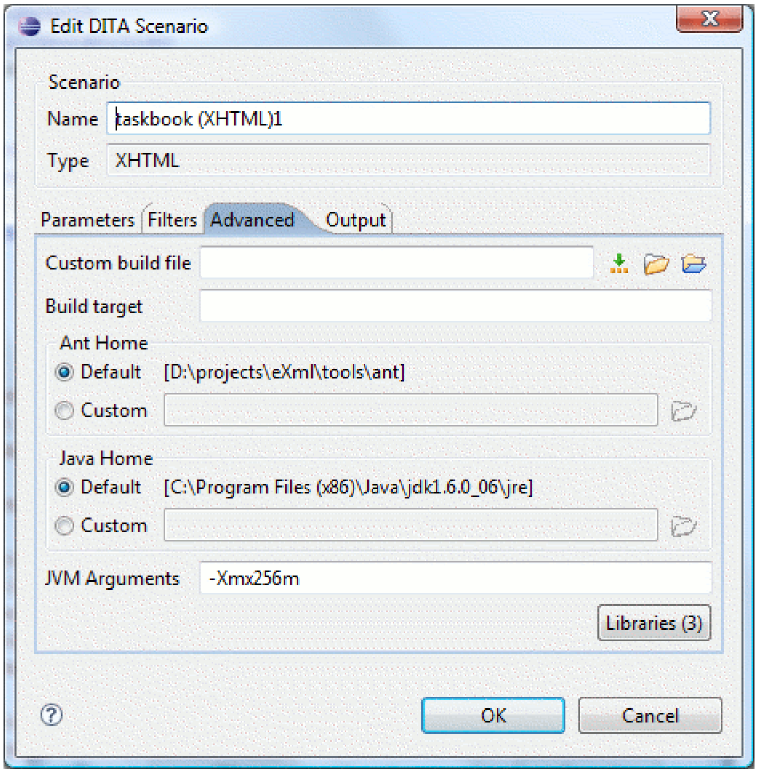
Use DITAVAL file
: If you already have a DITAVAL file associated with the transformed map you can specify the path to it and it will be used when filtering content. You can find out more about constructing a DITAVAL file in the DITA OT Documentation [http://docs.oasis-open.org/dita/v1.1/CD01/langspec/common/about-ditaval.html].

Exclude from output all elements with any of the following attributes
: You can configure a simple list of attribute (name, value) pairs which when present on an element in the input will remove it from output.

The *Advanced* tab

In the Advanced Tab you can specify advanced options for the transformation.

Figure 4.53. Advanced settings tab



Scenario

Name taskbook (XHTML)1

Type XHTML

Custom build file

Build target

Ant Home

● Default [D:\projects\eXml\tools\ant]

○ Custom

Java Home

● Default [C:\Program Files (x86)\Java\jdk1.6.0_06\jre]

○ Custom

JVM Arguments -Xmx256m

You have several parameters that you can specify here:

Custom build file
: If you use a custom DITA-OT build file you can specify the path to the customized build file. If empty, the `build.xml` file from the `dita.dir` directory configured in the *Parameters* tab will be used.

Build target
: You can specify a build target to the build file. By default no target is necessary and the default "init" target is used.

Ant Home
: You can specify a custom ANT installation to run the DITA Map transformation. By default it is the ANT installation bundled with <oXygen/>.

Java Home
: You can specify a custom Java Virtual Machine to run the ANT transformation. By default it is the Java Virtual Machine used by <oXygen/>.

JVM Arguments
: This parameter allows you to set specific parameters to the Java Virtual Machine used by ANT. By default it is set to `-Xmx256m` which means the transformation process is allowed to use 256 megabytes of memory.

Example 4.6. Increasing the memory for the ANT process

Sometimes, when performing a large DITA map transformation you may want to increase the memory allocated to the Java Virtual Machine from the default value (64 MB) to a higher value (256MB). You can do this easily by setting the value '-Xmx256m' without quotes to the "JVM Arguments" text field. In this way you can avoid the Out of Memory (OutOfMemoryError) messages received from the ANT process.

Libraries

You can specify all the additional libraries (jar files or additional class paths) which will be used by the ANT transformer.

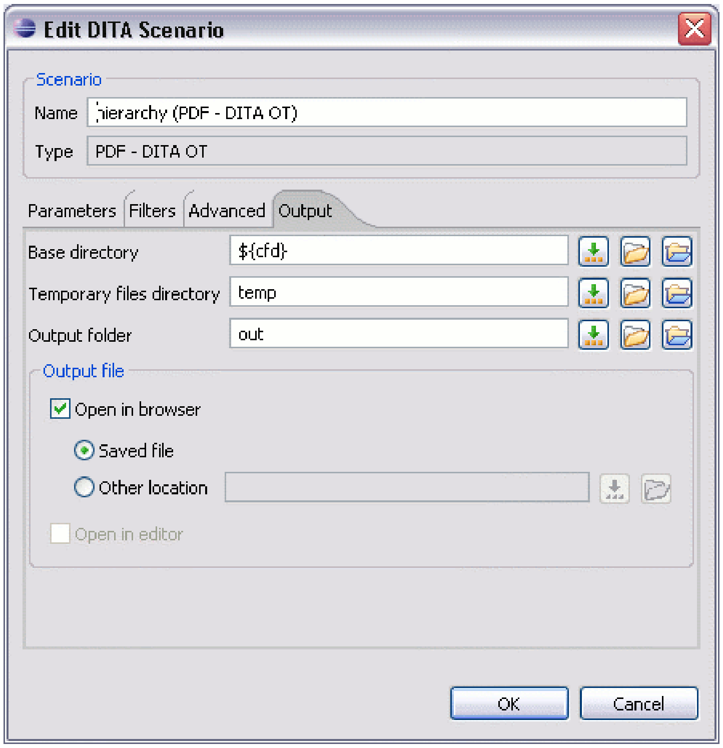
Example 4.7. Additional jars specified for XHTML

For example the additional jars specified for XHTML are the DITA-OT *dost* and *resolver* jars, *xerces* and *saxon* 6 jars.

The *Output* tab

In the Output Tab you can configure options related to the place where the output will be generated.

Figure 4.54. Output settings tab



Edit DITA Scenario

Scenario

Name: hierarchy (PDF - DITA OT)

Type: PDF - DITA OT

Base directory: ${cfd}

Temporary files directory: temp

Output folder: out

Output file

☑ Open in browser

● Saved file

○ Other location

□ Open in editor

You have several parameters that you can specify here:

Base directory

All the relative paths which appear as values in parameters are considered relative to the base directory. The default value is the directory where the transformed map is located.

Temporary files directory

This directory will be used to store pre-processed temporary files until the final output is obtained.

Output folder

The folder where the final output content will be copied.

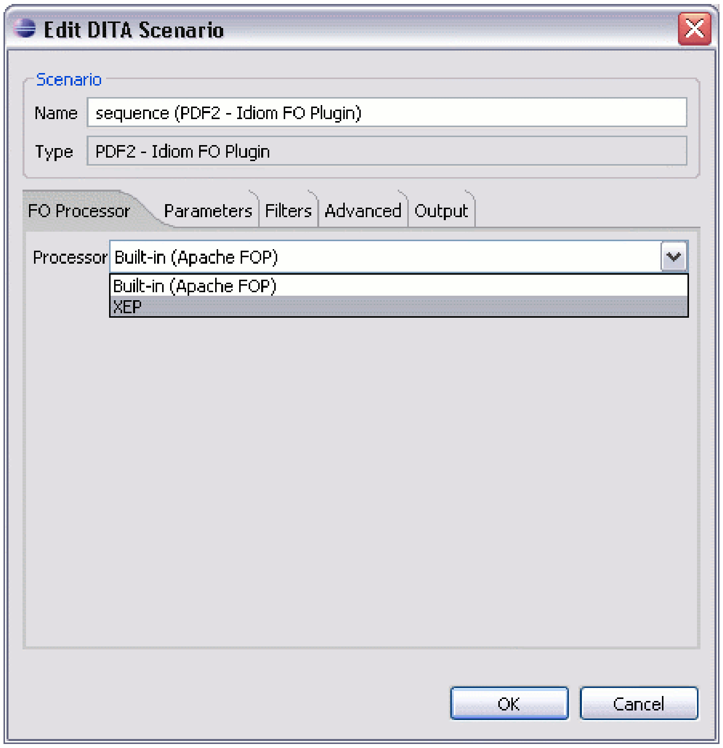
Output file options

The transformation output can then be opened in a browser or even in the editor if specified.

The *FO Processor* tab

This tab appears only when selecting to generate PDF output using the IDIOM FO Plugin and allows you to choose the FO Processor.

Figure 4.55. FO Processor configuration tab



Name: sequence (PDF2 - Idiom FO Plugin)

Type: PDF2 - Idiom FO Plugin

Processor: Built-in (Apache FOP)

You can choose between two processors:

Built-in (Apache FOP)

This processor comes bundled with <oXygen/>. You can find more information about it here.

XEP

The RenderX [http://www.renderx.com/] XEP processor. You can add it very easy from here.

If you select *XEP* in the combo and XEP was already installed in <oXygen/> you can see the detected installation path appear under the combo.

XEP is considered as installed if it was detected from one of the following sources:

XEP was added as an external FO Processor in the <oXygen/> preferences. See here.
The system property "com.oxygenxml.xep.location" was set to point to the XEP executable file for the platform (eg: xep.bat on Windows).
XEP was installed in the `frameworks/dita/DITA-OT/demo/fo/lib` directory of the <oXygen/> installation directory.

Tip

The DITA-OT contributors recommend the use of the IDIOM FO Plugin to transform DITA Maps to PDF as opposed to using the standard PDF target in the DITA-OT framework.

As IDIOM is also bundled with <oXygen/> the *PDF2 - IDIOM FO Plugin* output format should be your first choice in transforming your map to PDF. If you do not have a XEP licence you can transform using Apache FO Processor.

Set a font for PDF output generated with Apache FOP

When a DITA map is transformed to PDF using the Apache FOP processor and it contains some Unicode characters that cannot be rendered by the default PDF fonts then a font that is capable to render these characters must be configured and embedded in the PDF result.

The settings that must be modified for configuring a font for the Apache FOP processor are detailed in this section.

Running a DITA Map ANT transformation

The transformation is run as an external ANT process so you can continue using the application as the transformation unfolds. All output from the process appears in the *DITA Transformation* tab.

Tip

The HTTP proxy settings from <oXygen/> are also used for the ANT transformation so if the transformation fails because it cannot connect to an external location you can check the HTTP/Proxy Configuration.

DITA OT customization support

Support for transformation customizations

You can change all DITA transformation parameters to customize your needs. See here for more details.In addition, you can specify a custom build file, parameters to the JVM and many more for the transformation.

Using your own DITA OT toolkit from <oXygen/>

The DITA-OT toolkit which comes with <oXygen/> is located in the `{INSTALLATION_DIRECTORY}/frameworks/dita/DITA-OT` directory.

You can configure another DITA-OT toolkit directory for use in <oXygen/> To do this you must edit the transformation scenario that you are using and in the Parameters tab change the "dita.dir" parameter to your custom DITA-OT installation directory. Also in the Advanced tab (the Libraries button) you have to add:

- the `dost.jar` and `resolver.jar` libraries as file paths that point to the libraries from your custom DITA-OT installation directory
- the installation directory of your custom DITA-OT and the `lib` subdirectory of that installation directory as directory paths

Using your custom build file

You can specify a custom build file to be used in DITA-OT ANT transformations by editing the transformation scenario that you are using and in the Advanced tab change the *Custom build file* path to point to the custom build file.

Customizing the <oXygen/> Ant tool

The ANT 1.7 tool which comes with <oXygen/> is located in the `{INSTALLATION_DIRECTORY}/tools/ant` directory. Any additional libraries for ANT must be copied to the <oXygen/> ANT `lib` directory.

Example 4.8. Enabling JavaScript in ANT build files

If you are using Java 1.6 to run <oXygen/> the ANT tool should need to additional libraries to process JavaScript in build files.

If you are using Java 1.5 you have to copy the bsf.jar [http://jakarta.apache.org/bsf/] and js.jar [http://www.mozilla.org/rhino/download.html] libraries in the <oXygen/> ANT `lib` directory.

Upgrading to a new version of DITA OT

The DITA OT framework bundled in <oXygen/> is located in the `{INSTALLATION_DIRECTORY}/frameworks/dita/DITA-OT` directory.

Important

There are a couple of modifications made to the DITA OT framework which will be overwritten if you choose to copy the new DITA-OT version over the bundled one:

The DTD's in the framework have been enriched with documentation for each element. If you overwrite you will lose the documentation which is usually shown when hovering an element or in the Model View
The IDIOM FO Plugin comes pre-installed in the bundled DITA-OT framework
Several build files from the IDIOM plugin have been modified to allow transformation using the <oXygen/> Apache Built-in FOP libraries and usage of the <oXygen/> classpath while transforming.

Increasing the memory for the Ant process

You can give custom JVM Arguments to the ANT process. See here for more details.

Resolving topic references through an XML catalog

If you customized your map to refer topics using URI's instead of local paths or you have URI content references in your DITA topic files and you want to resolve the URIs with an XML catalog when the DITA map is transformed then you have to add the catalog to <oXygen/>. The DITA Maps Manager view will solve the displayed topic refs through the added XML catalog and also the DITA map transformations (for PDF output, for XHTML output, etc) will solve the URI references through the added XML catalog.

DITA specializations support

Important

If you are using DITA specializations we recommend that you to activate the *Enable DTD processing in document type detection* checkbox in the Document Type Association page.

Support for editing DITA Map specializations

In addition to recognizing the default DITA map formats: *map* and *bookmap* the DITA Maps Manager can also be used to open and edit specializations of DITA Maps.

All advanced edit actions available for the map like insertion of topic refs, heads, properties editing, allow the user to specify the element to insert in an editable combo. Moreover the elements which appear initially in the combo are all the elements which are allowed to appear at the insert position for the given specialization.

The topic titles rendered in the DITA Maps Manager are collected from the target files by matching the *class* attribute and not a specific element name.

When editing DITA specializations of maps in the main editor the insertions of topic reference, topic heading, topic group and conref actions should work without modification. For the table actions you have to modify each action by hand to insert the correct element name at caret position. You can go to the *DITA Map* document type from the Document Type Association page and edit the table actions to insert the element names as specified in your specialization. See this section for more details.

Support for editing DITA Topic specializations

In addition to recognizing the default DITA topic formats: *topic*, *task*, *concept*, *reference* and *composite*, topic specializations can also be edited in the Author page.

The Content Completion should work without additional modifications and you can choose the tags which are allowed at the caret position.

The CSS styles in which the elements are rendered should also work on the specialized topics without additional modifications.

The toolbar/menu actions should be customized to insert the correct element names if this is the case. You can go to the *DITA* document type from the Document Type Association page and edit the actions to insert the element names as specified in your specialization. See this section for more details.

Editing XML Schemas

An XML Schema describes the structure of an XML document and is used to validate XML document instances against it in order to check that the XML instance conforms to the specified requirements. If the XML instance conforms to the schema then it is said to be valid, otherwise it is invalid.

<oXygen/> has two pages dedicated to editing XML Schema: the usual Text page and the visual Schema editor page.

XML Schema Text Editor

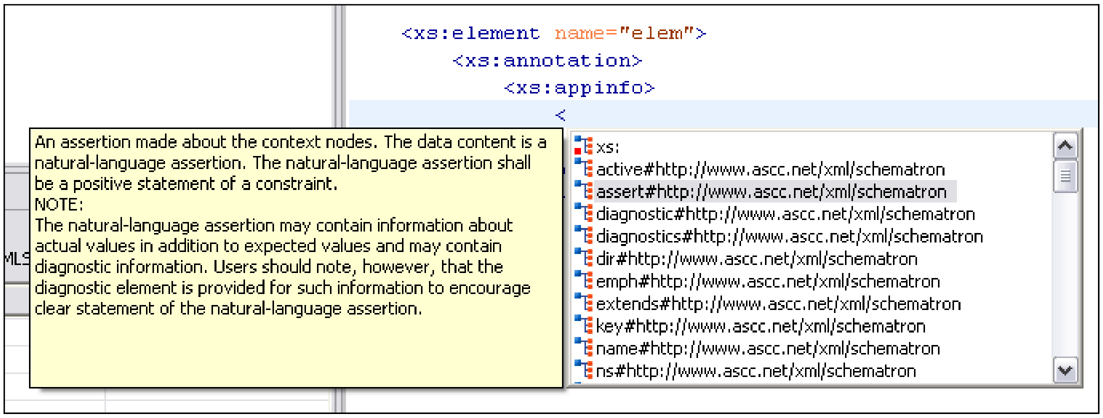
This page presents the usual text view of an XML document synchronized in real time with an outline view. The outline view has two display modes: the mode and the components mode. To activate a side by side source and diagram presentation you have to enable the *Show XML Schema Diagram* checkbox from the Diagram preferences page.

Special content completion features

The editor enhances the content completion of the XML editor inside the *xs:annotation/xs:appinfo* elements of an XML Schema with special support for the elements and attributes from a custom schema(by default ISO Schematron). This content completion enhancement can be configured from the XSD Content Completion preferences page.

If the current XML Schema schema imports or includes other XML Schema schemas then the global types and elements defined in the imported / included schemas are available in the content completion window together with the ones defined in the current file.

Figure 4.56. Schematron support in XML Schema content completion



```
<xs:element name="elem">
    <xs:annotation>
        <xs:appinfo>
            <
```

An assertion made about the context nodes. The data content is a natural-language assertion. The natural-language assertion shall be a positive statement of a constraint.
NOTE:
The natural-language assertion may contain information about actual values in addition to expected values and may contain diagnostic information. Users should note, however, that the diagnostic element is provided for such information to encourage clear statement of the natural-language assertion.

xs:
active#http://www.ascc.net/xml/schematron
assert#http://www.ascc.net/xml/schematron
diagnostic#http://www.ascc.net/xml/schematron
diagnostics#http://www.ascc.net/xml/schematron
dir#http://www.ascc.net/xml/schematron
emph#http://www.ascc.net/xml/schematron
extends#http://www.ascc.net/xml/schematron
key#http://www.ascc.net/xml/schematron
name#http://www.ascc.net/xml/schematron
ns#http://www.ascc.net/xml/schematron

XML Schema actions

- contextual menu of current editor+Schema → Show definition (**Ctrl + Alt + ENTER**): Move the cursor to the definition of the referenced XML Schema item - element, group, simple or complex type. The same action is executed on a double click on a component name in the Schema Outline view. You can define a scope for this action in the same manner you define for Search Declarations

Note

The actions are available when the current editor is of XML Schema type.

XML Schema editor specific actions

The list of actions specific for the XML Schema editor of <oXygen/> is:

- contextual menu of current editor → Show Definition : move the cursor to the definition of the current element in this XSD schema.

Flatten an XML Schema

If an XML Schema is organized on several levels linked by *xs:include* statements sometimes it is more convenient to work on the schema as a single flat file. To flatten schema <oXygen/> recursively adds included files to the master one. That means <oXygen/> replaces the *xs:include* elements with the ones coming from the included files.

This action works at file level not at schema document level so it is available only in Text mode of XML Schema editor. It can be accessed from the XML Schema text editor's contextual menu -> Refactoring -> Flatten Schema. Alternatively you can select one or more schemas in the Project view and invoke the action from the view's contextual menu. In this last case the feedback of the action will be presented in the Information view.

Schema flattening can also be accessed from the command line by running a command line the script, `flattenSchema.bat` on Windows or `flattenSchema.sh` on Mac OS X, Unix/Linux with the input file as the first argument and the output file as the second argument.

In the following example *master.xsd* includes *slave.xsd*. This, in turn, includes *slave1.xsd* which includes both *slave2.xsd* and *slave3.xsd*.

Listing of master.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="tns" xmlns:tns="tns"
    xmlns:b="b" >
  <!-- included elements from slave.xsd -->
  <xs:include schemaLocation="slave.xsd"></xs:include>
  <!-- master.xsd -->
  <xs:element name="element1">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:element2" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Listing of slave.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="tns" xmlns:a="a" xmlns:b="b"
    xmlns:c="c">
  <!-- included elements from slave1.xsd -->
  <xs:include schemaLocation="slave1.xsd"></xs:include>
  <!-- slave  -->
  <xs:element name="element2" xmlns:c="x"/>
</xs:schema>
```

Listing of slave1.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="tns" xmlns:tns="tns"
    blockDefault=" restriction">
  <!-- included elements from slave2.xsd -->
  <xs:include schemaLocation="slave2.xsd"></xs:include>
  <!-- included elements from slave3.xsd -->
  <xs:include schemaLocation="slave3.xsd"></xs:include>
  <!-- slave1  -->
  <xs:element name="element0"/>
  <xs:element name="element7"/>
  <xs:element name="element7Substitute"
      substitutionGroup="tns:element7"
      block="extension"/>
  <xs:element name="element6">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:element7"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="type1">
    <xs:sequence>
      <xs:element ref="tns:element0"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Listing of slave2.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="tns"
  xmlns:tns="tns"
  elementFormDefault="qualified"
  attributeFormDefault="qualified">
  <!-- slave2 -->
  <xs:element name="a"></xs:element>
  <a:element name="element9"
        xmlns:a="http://www.w3.org/2001/XMLSchema">
    <xs:complexType>
      <xs:sequence>
        <!-- This element is from the target namespace -->
        <xs:element name="element3"
              xmlns:b="http://www.w3.org/2001/XMLSchema"/>
        <!-- Element from no namespace -->
        <xs:element name="element4" form="unqualified"/>
        <a:element ref="tns:a"></a:element>
      </xs:sequence>
      <!-- Attribute from the target namespace -->
      <b:attribute name="attr1" type="xs:string"
            xmlns:b="http://www.w3.org/2001/XMLSchema"/>
      <!-- Attribute from the no namespace -->
      <xs:attribute name="attr2" type="xs:string"
            form="unqualified"/>
    </xs:complexType>
  </a:element>
</xs:schema>
```

Listing of slave3.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="tns" finalDefault="restriction"
    xmlns:tns="tns">
  <!-- slave3 -->
  <xs:complexType name="ct1"/>
  <xs:complexType name="ct2" final="extension">
    <xs:complexContent>
```

```
      <xs:extension base="tns:ct1"/>
    </xs:complexContent>
  </xs:complexType>
  <xs:simpleType name="st1" final="union">
    <xs:restriction base="xs:integer"/>
  </xs:simpleType>
  <xs:simpleType name="st2" final="union">
    <xs:restriction base="tns:st1">
      <xs:enumeration value="1"/>
      <xs:enumeration value="2"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="e1" type="tns:c1" final="restriction"/>
  <xs:element name="e2ext" type="tns:c2"
        substitutionGroup="tns:e1"></xs:element>
  <xs:complexType name="c1">
    <xs:sequence>
      <xs:element ref="tns:e1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="c2">
    <xs:complexContent>
      <xs:extension base="tns:c1">
        <xs:sequence>
          <xs:element ref="tns:e1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

Listing of master.xsd after it has been flattened

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="tns" xmlns:a="a"
      xmlns:b="b" xmlns:c="c" xmlns:tns="tns"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- included elements from slave.xsd -->
  <!-- included elements from slave1.xsd -->
  <!-- included elements from slave2.xsd -->
  <!-- slave2 -->
  <xs:element block="restriction" name="a"/>
  <a:element block="restriction" name="element9"
        xmlns:a="http://www.w3.org/2001/XMLSchema">
    <xs:complexType>
      <xs:sequence>
        <!-- This element is from the target namespace -->
        <xs:element block="restriction" form="qualified" name="element3"
          xmlns:b="http://www.w3.org/2001/XMLSchema"/>
        <!-- Element from no namespace -->
        <xs:element block="restriction" form="unqualified"
              name="element4"/>
        <a:element ref="tns:a"/>
      </xs:sequence>
      <!-- Attribute from the target namespace -->
      <b:attribute form="qualified" name="attr1" type="xs:string"
        xmlns:b="http://www.w3.org/2001/XMLSchema"/>
      <!-- Attribute from the no namespace -->
      <xs:attribute form="unqualified" name="attr2" type="xs:string"/>
    </xs:complexType>
  </a:element>
  <!-- included elements from slave3.xsd -->
  <!-- slave3 -->
  <xs:complexType block="restriction" final="restriction" name="ct1"/>
  <xs:complexType block="restriction" final="extension" name="ct2">
    <xs:complexContent>
      <xs:extension base="tns:ct1"/>
    </xs:complexContent>
  </xs:complexType>
  <xs:simpleType final="union" name="st1">
    <xs:restriction base="xs:integer"/>
  </xs:simpleType>
  <xs:simpleType final="union" name="st2">
    <xs:restriction base="tns:st1">
      <xs:enumeration value="1"/>
      <xs:enumeration value="2"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element block="restriction" final="restriction" name="e1"
        type="tns:c1"/>
  <xs:element block="restriction" final="restriction" name="e2ext"
        substitutionGroup="tns:e1"
    type="tns:c2"/>
  <xs:complexType block="restriction" final="restriction"
        name="c1">
    <xs:sequence>
      <xs:element ref="tns:e1"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType block="restriction" final="restriction"
        name="c2">
    <xs:complexContent>
      <xs:extension base="tns:c1">
        <xs:sequence>
          <xs:element ref="tns:e1"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- slave1  -->
  <xs:element block="restriction" name="element0"/>
  <xs:element block="restriction" name="element7"/>
  <xs:element block="extension" name="element7Substitute"
        substitutionGroup="tns:element7"/>
  <xs:element block="restriction" name="element6">
    <xs:complexType>
```

```
      <xs:sequence>
        <xs:element ref="tns:element7"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType block="restriction" name="type1">
    <xs:sequence>
      <xs:element ref="tns:element0"/>
    </xs:sequence>
  </xs:complexType>
  <!-- slave  -->
  <xs:element name="element2" xmlns:c="x"/>
  <!-- master.xsd -->
  <xs:element name="element1">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="tns:element2"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The case of XML Schema redefinitions is also handled as the example below shows.

Listing of master.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:redefine schemaLocation="slave1.xsd">
    <xs:complexType name="tp">
      <xs:complexContent>
        <xs:extension base="tp">
          <xs:choice>
            <xs:element name="el2" type="xs:NCName"/>
            <xs:element name="el3" type="xs:string"/>
          </xs:choice>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:redefine>
  <xs:element name="el" type="tp"/>
</xs:schema>
```

Listing of slave1.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:redefine schemaLocation="slave2.xsd">
    <xs:complexType name="tp">
      <xs:complexContent>
```

```
        <xs:extension base="tp">
          <xs:attribute name="a"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  </xs:redefine>
</xs:schema>
```

Listing of slave2.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="tp">
    <xs:sequence>
      <xs:element name="el" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Listing of master.xsd after it has been flattened>

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="tp">
    <xs:complexContent>
      <xs:extension base="tp_Redefined1">
        <xs:choice>
          <xs:element name="el2" type="xs:NCName"/>
          <xs:element name="el3" type="xs:string"/>
        </xs:choice>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="tp_Redefined1">
    <xs:complexContent>
      <xs:extension base="tp_Redefined0">
        <xs:attribute name="a"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="tp_Redefined0">
    <xs:sequence>
      <xs:element name="el" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="el" type="tp"/>
</xs:schema>
```

The references to the included schema files can be resolved through an XML Catalog.

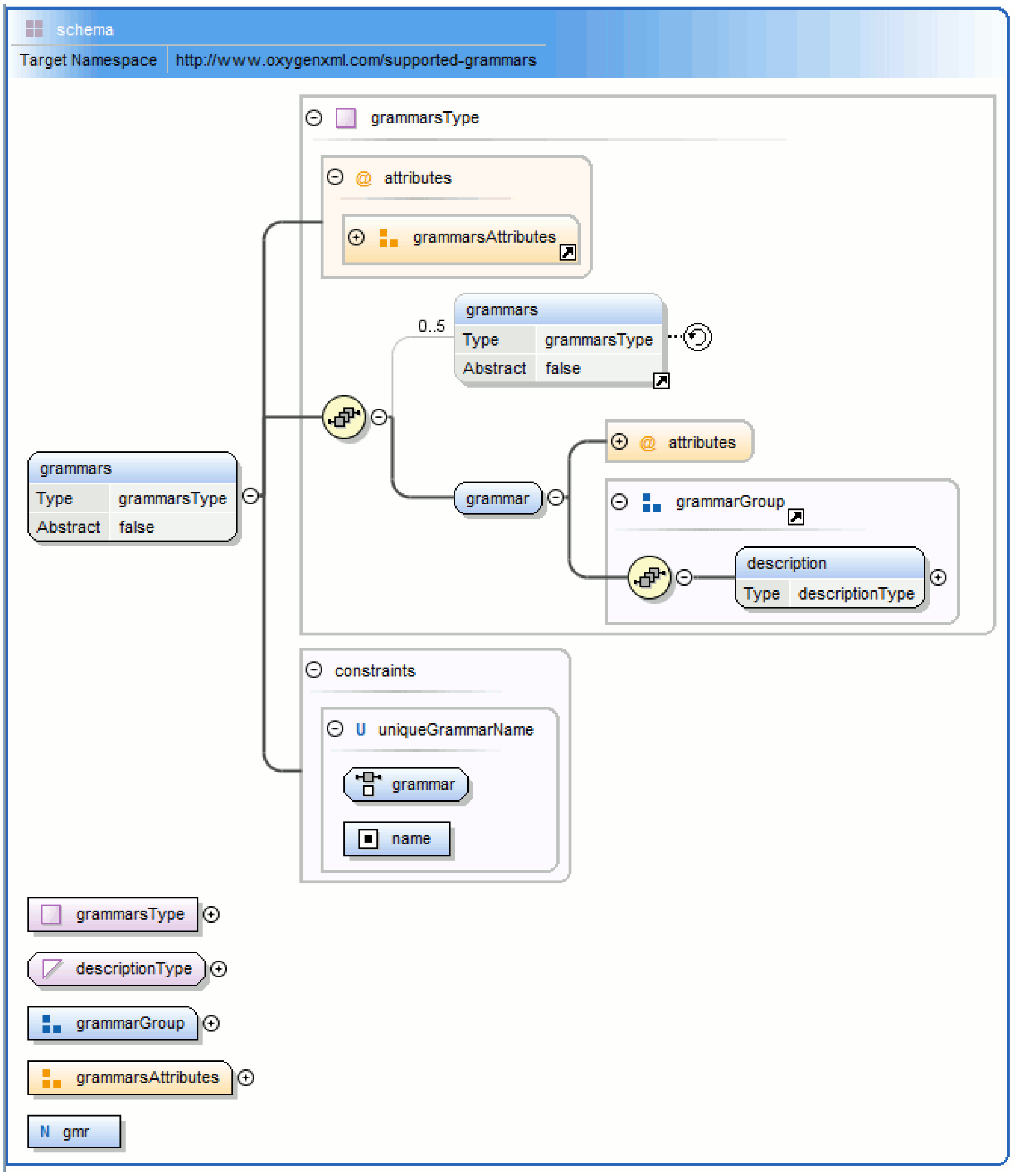
XML Schema Diagram Editor

Introduction

XML Schemas enable document designers to specify the allowed structure and content of an XML document and to check if an XML document is valid.

<oXygen/> provides a simple and expressive Schema Diagram Page for editing XML Schemas. The schema diagram helps both the content authors who want to understand a schema and schema designers who develop complex schemas.

Figure 4.57. XML Schema Diagram



Navigation in the schema diagram

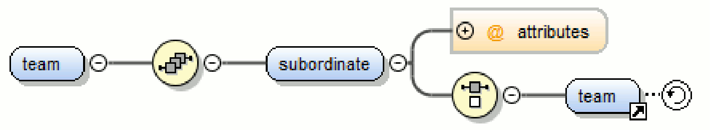
The following editing and navigation features work for all types of schema components:

- Move/refer components in the diagram using drag-and-drop;
- Select consecutive components on the diagram (components from the same level) using the **Shift** key to . You can also make discontinuous selections in the schema diagram using the **Ctrl** key.
- Use **Home/End** to navigate to the first/last component from the same level. Use Ctrl-Home to go to the diagram root and Ctrl-End to go to the last child of the selected component.
- You can easily go back to a previously visited component while moving from left to right. The path will be preserved only if you use the Left Arrow or Right Arrow. For example, if you are on the second attribute from an attribute group and navigate to the attribute group using the Left Arrow, when press the Right Arrow the second attribute will be selected.
- Go back and forward between components viewed or edited in the diagram by selecting them in the *Outline* view: ⬅ Back (go to previous schema component), ➡ Forward (go to next schema component) and ↙ Go to Last Modification (go to last modified schema component); the buttons are available on the *Navigation* toolbar.
- Copy, refer or move global components, attributes, and identity constraints to a different position and from one schema to another using the cut/copy and paste/paste as reference actions;
- Go to the definition of an element or attribute with the action *Show Definition*.
- Search in the diagram using the Find/Replace dialog. You can find/replace components only in the current file scope.
- You can expand and see the contents of the imports/includes/redefines in the diagram but in order to edit components from other schemas the schema for each component will be opened as a separate file in <oXygen/>.

Tip

If an XML Schema referenced by the current opened schema was modified on disk, the change will be detected and you will be asked to refresh the current schema contents.

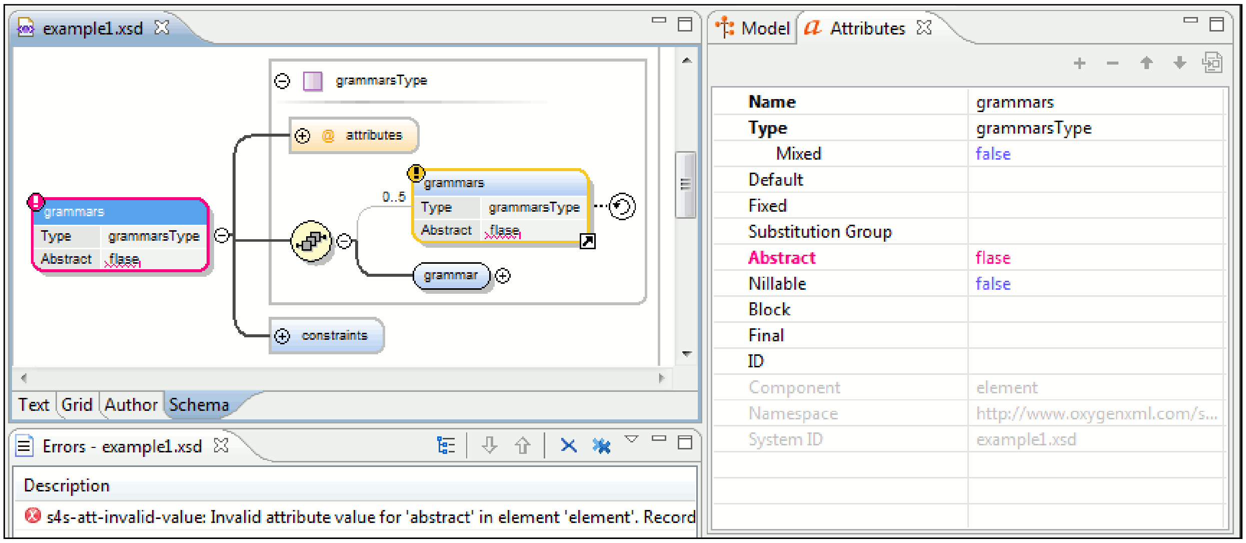
- When a recursive reference is encountered the diagram signals this with a special recurse symbol. Clicking this symbol you can navigate between the diagram components which recurse.



Schema validation

Validation for the Schema Diagram Page is seamlessly integrated in the <oXygen/> validation framework.

Figure 4.58. XML Schema Validation



Name	grammars
Type	grammarsType
Mixed	false
Default	
Fixed	
Substitution Group	
Abstract	flase
Nillable	false
Block	
Final	
ID	
Component	element
Namespace	http://www.oxygenxml.com/s...
System ID	example1.xsd

Description

s4s-att-invalid-value: Invalid attribute value for 'abstract' in element 'element'. Record

Errors are presented by highlighting invalid component properties in the Attributes View and also directly in the diagram if the property is presented. Invalid facets for a component are highlighted in the Facets View.

Components with invalid properties are rendered by default with a red border. You can customize the error colors from the Document checking user preferences. When hovering an invalid component the tooltip will present the validation errors for that component.

When editing a value which is supposed to be a qualified or unqualified XML name you also have as you type validation of the entered value which is very useful to avoid setting not valid XML names for the given property.

If you validate the entire schema using Document → Validate document (**Alt+Shift+V V (Cmd+Alt+V V on Mac OS)**) or the action available on the *Validate* toolbar, all validation errors will be presented. To resolve an error just click (or double click for errors from other schemas) and the corresponding schema component will be display as the diagram root so that you can easily correct the error.

Important

If the schema imports using only the namespace and without specifying the schema location and a catalog is setup mapping the namespace to a certain location both validation and diagram will correctly identify the imported schema.

Tip

If there are unresolved references in your schema a hint will be presented suggesting the use of validation scenarios if the current edited schema is a module.

Schema editing actions

The schema can be edited using drag and drop operations or contextual menu actions.

Drag and drop provides the easiest way to move the existing components to other locations in the schema. For example an element reference can be quickly inserted in the diagram with a drag and drop from the *Outline* view to a compositor in the diagram. Also the components order in an *xs:sequence* can be easily changed using drag and drop. You can easily set the an attribute/element type if this property has not been set by dragging a simple or complex type from the diagram over it. Also you can set the type property for a simple or complex type if the property is not already set by dragging a simple or complex type over it. The type of the mouse pointer will indicate the action which will be performed after drag and drop. Depending on the drop context, the dragged element will be either moved as a child of the drop parent or referred from the parent. If *Ctrl* is pressed, the component will be copied to the destination.

You can edit some schema components directly in the diagram. For these components you can edit the name and the additional properties presented in the diagram. To do this just double click on the value you want to edit. If you want to edit the name of a selected component you can also press Enter. The list of properties which can be displayed for each component can be customized here. When editing references, you can choose from a list of available components. Components from an imported schema for which the target namespace does not have an associated prefix will be displayed as *componentName#targetNamespace* in the list. If the reference is from a target namespace which was not yet mapped you will be prompted to add prefix mappings for the inserted component namespace in the current edited schema.

You can also change the compositor by double-click on it and choose the compositor you want from the proposals list.

There are some components that cannot be edited directly in the diagram: imports, includes, redefines. The editing action can be performed if you double-click or press Enter on an import/include/redefine component. An edit dialog will appear allowing you to customize the directives.

The contextual menu of the Schema page offers the following edit actions:

Show Definition (Ctrl-Shift-Enter)
: Shows the definition for the current selected component. For references this action is available by clicking on the arrow displayed in its bottom right corner.

Open Schema (Ctrl-Shift-Enter)
: Open the selected schema. This action is available for imports, includes and redefines. If the file you try to open does not exist, a warning message will be displayed and you have the possibility to create the file.

Show schema components (Ctrl-Shift-Enter)
: This is the complementary action of *Show Definition* and displays all the schema components. For a global component this action is available by clicking the arrow displayed in its top right corner.

Append
: Offers a list of valid components to append depending on the context. For example to a complex type you can append a compositor, a group, attributes, identity constraints (unique, key, keyref). After a named component was added in the diagram you can set a name for it.

Insert before
: Inserts before the selected component in the schema. The components to be inserted depend on the context. For example, before an import you can insert an import, an include or a redefine. After a named component was added in the diagram you can set a name for it.

Insert after
: Inserts a component after the selected component on the schema. The components to be inserted depend on the context. After a named component was added in the diagram you can set a name for it.

New global
: Inserts a global component in the schema diagram. This action does not depend on the current context.

: If you choose to insert an import you have to specify the url of the imported file, the target namespace and the import id. The same information, excluding the target namespace, is requested for an include or redefine.See the Edit Import dialog for more details

Note

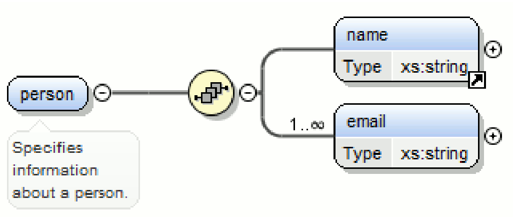
If the imported file has declared a target namespace, the field *Namespace* will be filled automatically.

Extract Global Element

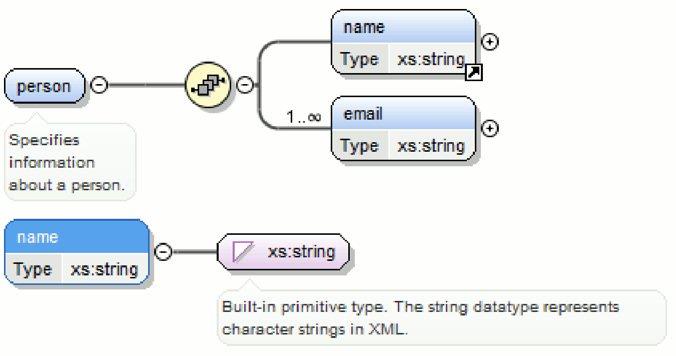
Action available for local elements. A local element is made global and will be replaced with a reference to the global element.

The local element properties that are also valid for the global element declaration are kept.

Example 4.9. Extracting a global element



If you execute **Extract Global Element** on element *name*, the result will be:

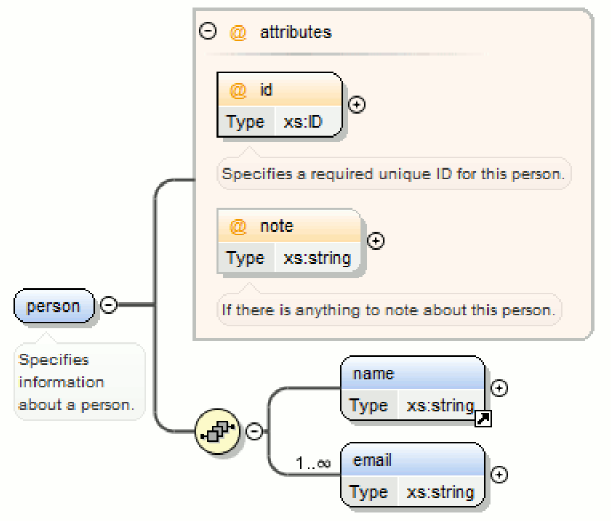

person
Specifies information about a person.
1..∞
name
Type xs:string
email
Type xs:string
name
Type xs:string
xs:string
Built-in primitive type. The string datatype represents character strings in XML.

Extract Global Attribute

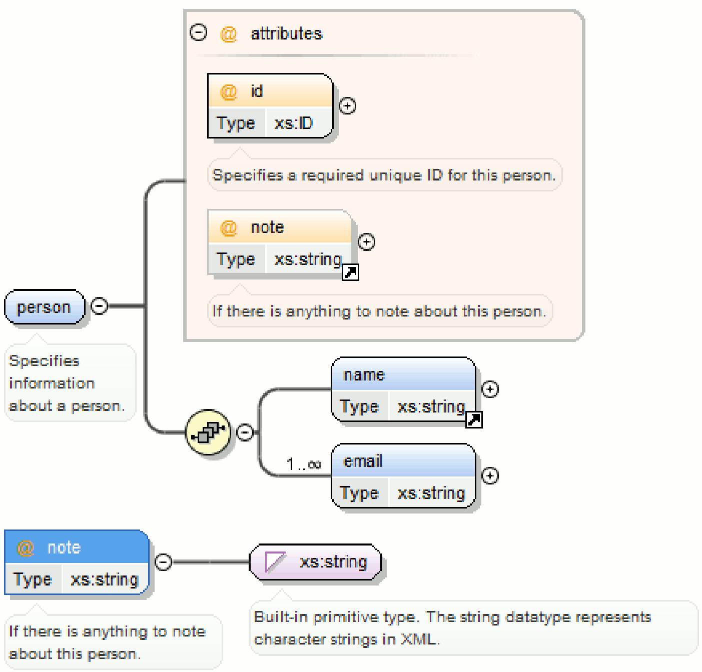
Action available for local attributes. A local attribute is made global and replaced with a reference to the global attribute.

The properties of local attribute that are also valid in the global attribute declaration are kept.

Example 4.10. Extracting a global attribute


attributes
id
Type xs:ID
Specifies a required unique ID for this person.
note
Type xs:string
If there is anything to note about this person.
person
Specifies information about a person.
name
Type xs:string
1..∞
email
Type xs:string

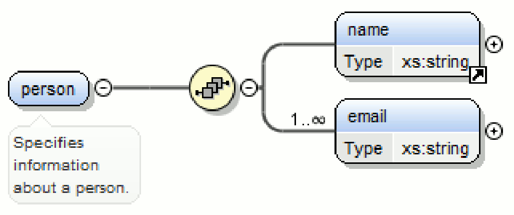
If you execute **Extract Global Attribute** on attribute *note*, the result will be:


attributes
id
Type xs:ID
Specifies a required unique ID for this person.
note
Type xs:string
If there is anything to note about this person.
person
Specifies information about a person.
name
Type xs:string
1..∞
email
Type xs:string
note
Type xs:string
If there is anything to note about this person.
xs:string
Built-in primitive type. The string datatype represents character strings in XML.

Extract Global Group

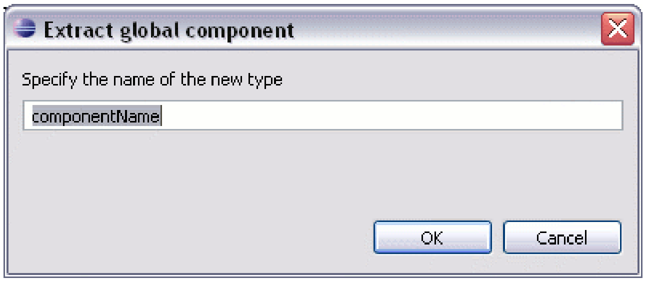
Action available for compositors (sequence, choice, all). This action extracts a global group and makes a reference to it. The action is enabled only if the compositor's parent is not a group.

Example 4.11. Extracting a global group

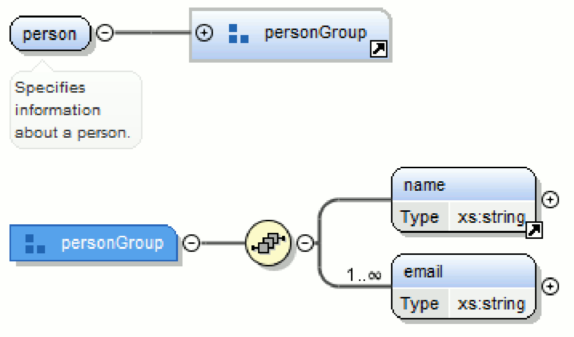


If you execute **Extract Global Group** on the sequence, the *Extract Global component* dialog is shown and you can choose a name for the group.

Figure 4.59. Extract Global Group dialog



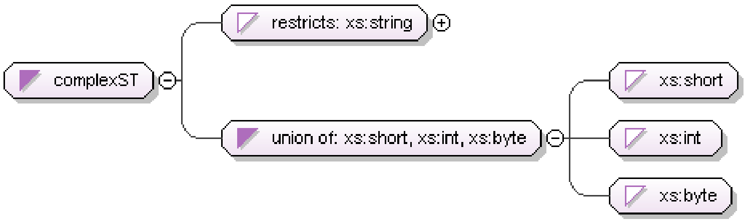
If you type `personGroup`, the result will be:



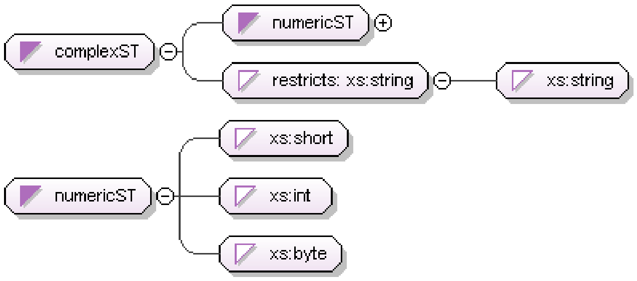
Extract Global Type

Action used to extract an anonymous simple type or an anonymous complex type as global. For anonymous complex types the action is available on the parent element.

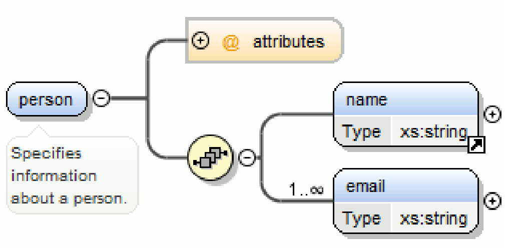
Example 4.12. Extracting a global simple type



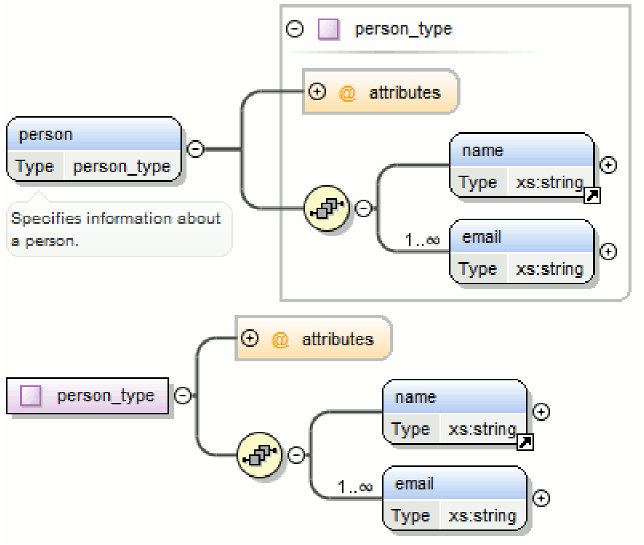
If you use the action on the union and choose `numericST` for the new global simple type name, the result will be:



Example 4.13. Extracting a global complex type



If you execute the action on element *person*, and choose *person_type* for the new complex type name, the result will be:



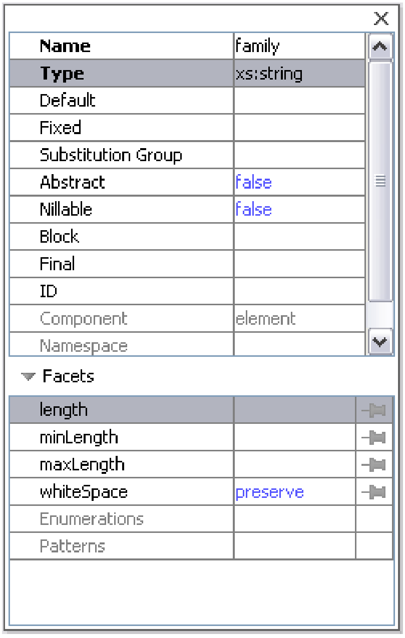
Rename Component

Rename the selected component. Click here for more details.

Edit Attributes... (**Alt+Shift+Enter**)

Allows you to edit the attributes of the selected component in the following dialog:

Figure 4.60. Edit Attributes dialog



Name	family
Type	xs:string
Default	
Fixed	
Substitution Group	
Abstract	false
Nillable	false
Block	
Final	
ID	
Component	element
Namespace	

Facets

length	
minLength	
maxLength	
whiteSpace	preserve
Enumerations	
Patterns	

The attributes presented in this dialog are the same attributes presented in the Attributes View and in the Facets View and the actions that can be performed are the same actions presented for the these views.

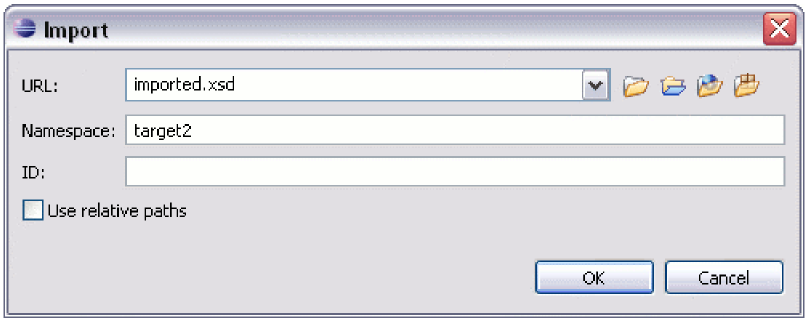
Edit Namespaces...

When performed on the schema root allows you to edit the schema Target namespace and namespace mappings. You can also invoke the action by double-clicking the target namespace property from Attributes View for the schema or by double-clicking the schema component. For details see the Edit Schema Namespaces dialog .

Edit Settings...

For imports, includes and redefines you can edit the schema location, Target namespace (only for imports) and id.

Figure 4.61. The Import dialog



URL: imported.xsd

Namespace: target2

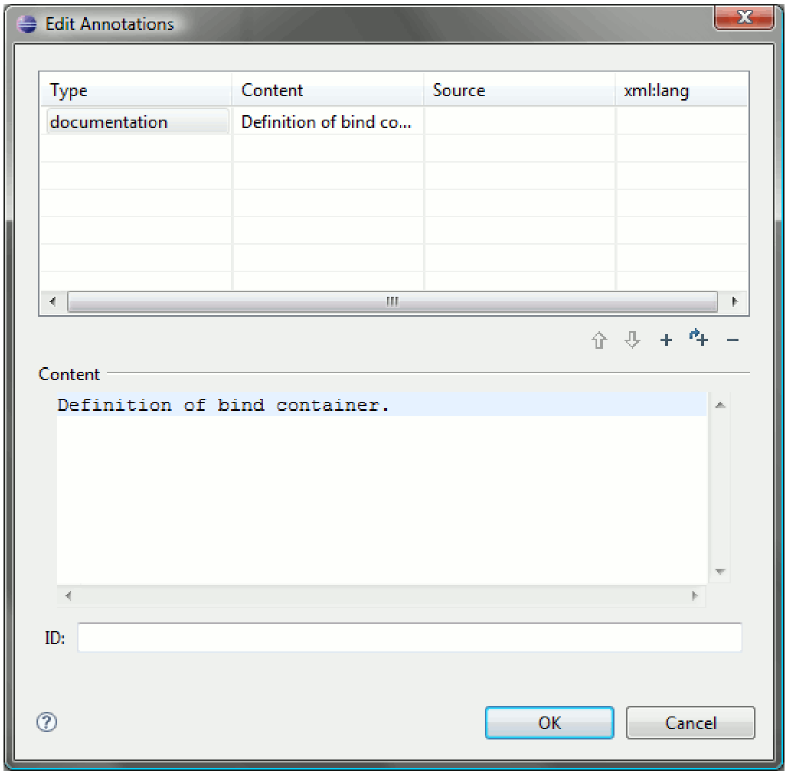
ID:

□ Use relative paths

Edit Annotations...

Allows you to edit the annotation for the selected schema component in the *Edit Annotations* dialog.

Figure 4.62. Edit Annotations dialog



Type	Content	Source	xml:lang
documentation	Definition of bind co...		

Content

```
Definition of bind container.
```

ID:

You can perform the following operations in the dialog:

- **Edit all appinfo/documentation items for a specific annotation**. All appinfo/documentation items for a specific annotation are presented in a table and can be easily edited. Information about an annotation item includes: type(documentation/appinfo), content, source(optional, specify the source of the documentation/appinfo element) and xml:lang. The content of a documentation/appinfo item can be edited in the Content area below the table.
- **Insert/Insert before/Remove documentation/appinfo.** + allows you to insert a new annotation item (documentation/appinfo). You can add a new item before the item selected in table by press the + button. Also you can delete the selected item using the − button.
- **Move items up/down** To do this use the ⇧ and ⇩ buttons.
- **Insert/Insert before/Remove annotation**. Available for components that allow multiple annotations like schemas or redefines.
- **Specify an ID for the component annotation**. The ID is optional.

Note

For imported/included components which do not belong to the current edited schema the dialog presents the annotation as read-only and you will have to open the schema where the component is defined in order to edit its annotation.

Note

Annotations are by default rendered under the component's graphical representation. When you have a reference to a component with annotations, these annotations will be presented in the diagram also below the reference component. The *Edit Annotations* action invoked from the contextual menu will edit the annotations for the reference. If the reference component does not have annotations you can edit the annotations of the referred component by double-clicking on the annotations area. Otherwise you can edit the referred component annotations only if you go to the definition of the component.

Cut (Ctrl-X)

Cut the selected component(s).

Copy (Ctrl-C)

Copy the selected components(s).

Paste (Ctrl-V)

Paste the component(s) from the clipboard as children of the selected component.

Paste as Reference

Create references to the copied component(s). If not possible a warning message will be displayed.

Remove (**Delete**)

Remove the selected component(s).

Optional

Can be performed on element/attribute/group references, local attributes, elements, compositors and element wildcards. The *minOccurs* property is set to *0* and the *use* property for attributes is set to *optional*.

Unbounded

Can be performed on element/attribute/group references, local attributes, elements, compositors and element wildcards. The *maxOccurs* property is set to *unbounded* and the *use* property for attributes is set to *required*.

Reference

Can be performed on local elements or attributes. This action makes a reference to a global element or attribute.

Search References

Searches all references of the item found at current cursor position in the defined scope if any. Click here for more details.

Search References in...

Searches all references of the item found at current cursor position in the specified scope. Click here for more details.

Search Occurrences in File

Searches all occurrences of the item found at current cursor position in the current file. Click here for more details.

Component Dependencies

Allows you to easily see the dependencies for the current selected component. Click here for more details.

Resource Hierarchy — Allows you to easily see the hierarchy for the current selected resource. Click here for more details.

Resource Dependencies — Allows you to easily see the dependencies for the current selected resource. Click here for more details.

Save as Image... — Save the diagram as image.

Generate Sample XML Files — Generate XML files using the current opened schema. The selected component will be the XML document root. See more on Generate Sample XML Files section.

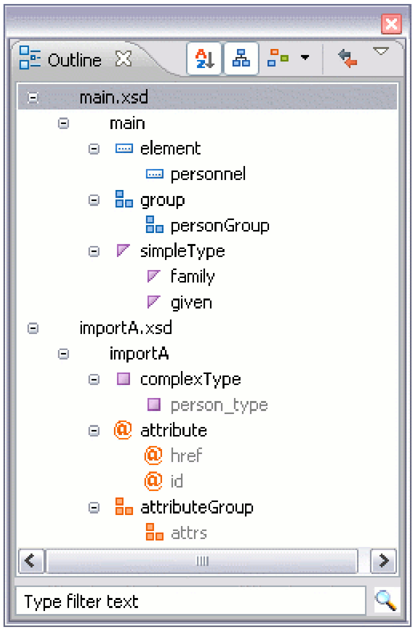
Show Annotations — Depending on the state (check/uncheck) the annotations are presented or hidden in the diagram.

Options... — Show the Schema preferences panel.

The Schema Outline View

The Schema Outline View presents all the global components grouped by their location, namespace or type. If hidden, you can open it from Window → Show View → Other → oXygen → Outline.

Figure 4.63. The Outline View for XML Schema



The Outline View provides the following options:

Sort — Allows you to sort alphabetically the schema components.

Show imported/included — Show also the components from imported/included schemas.

Grouping Options — Allows you to group the components by location, namespace or type. When grouping by namespace, the main schema target namespace is the first presented in the Outline view.

Selection update on caret move

Allows a synchronization between Outline View and schema diagram. The selected view from the diagram will be also selected in the Outline View.

The following contextual menu actions are available:

Remove (**Delete**)

Remove the selected item from the diagram.

Search References ()

Searches all references of the item found at current cursor position in the defined scope if any. Click here for more details.

Search References in...

Searches all references of the item found at current cursor position in the specified scope. Click here for more details.

Component Dependencies

Allows you to easily see the dependencies for the current selected component. Click here for more details.

Rename Component

Rename the selected component. Click here for more details.

If you know the component name, you can search for it by typing its name in the filter text field located in the bottom of the view or directly on the tree structure.

Tip

The search filter is case insensitive. The following wildcards are accepted:

- ***** - any string
- **?** - any character
- **,** -patterns separator

If no wildcards are specified, the string to search will be searched as a partial match (similar to ***textToFind***).

Note

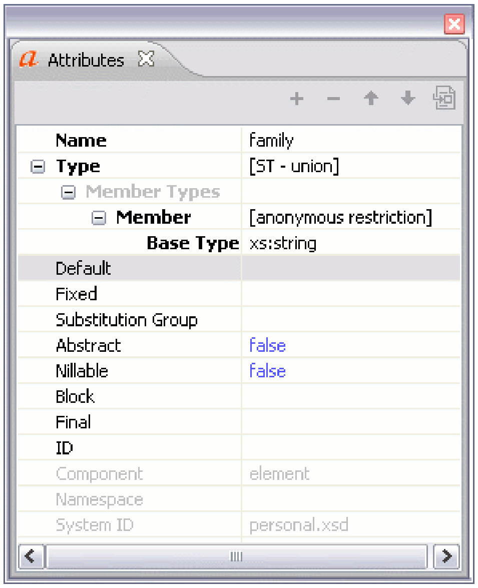
In the Text page the Outline has contextual actions like: Edit Attributes, Cut, Copy, Delete.

In the Text page you can switch between the current outline and the standard Outline View by pressing the button. Your decision will be applied to all new schema editors opened after this operation.

The Attributes view

The Attributes View presents the properties for the selected component in the schema diagram. For details about available properties for each schema component see the properties of schema components. If hidden, you can open it from Window → Show View → Other → oXygen → Attributes.

Figure 4.64. The Attributes view



Name	family
⊟ **Type**	[ST - union]
⊟ Member Types	
⊟ **Member**	[anonymous restriction]
Base Type	xs:string
Default	
Fixed	
Substitution Group	
Abstract	false
Nillable	false
Block	
Final	
ID	
Component	element
Namespace	
System ID	personal.xsd

The default value of a property is presented in the Attributes View with blue foreground. The properties that can't be edited are rendered with gray foreground. A non-editable category which contains at least one child is rendered with bold. Bold properties are properties with values set explicitly to them.

Properties for components which do not belong to the current edited schema are read-only but if you double-click them you can choose to open the corresponding schema and edit them.

You can edit a property by double-clicking on by pressing Enter. For most properties you can choose valid values from a list or you can specify another value. If a property has an invalid value or a warning, it will be highlighted in the table with the corresponding foreground color. By default properties with errors are presented with red and the properties with warnings with yellow. You can customize the error colors from the Document checking user preferences.

For imports, includes and redefines properties are not edited directly in the Attributes View. A dialog will be shown allowing you to specify properties for them.

The schema namespace mappings are not presented in Attributes View. You can view/edit these by choosing **Edit Namespaces** from the contextual menu on the schema root. See more in the Edit schema namespaces section.

The Attributes View has five actions available on the toolbar and also on the contextual menu:

+ Add
: Allows you to add a new member type to an union's member types category.

− Remove
: Allows you to remove the value of a property.

↑ Move Up
: Allows you to move up the current member to an union's member types category.

↓ Move Down
: Allows you to move down the current member to an union's member types category.

Copy
: Copy the attribute value.

Show Definition
: Show the definition for the selected type.

Edit Facets

Allows you to edit the facets for a simple type.

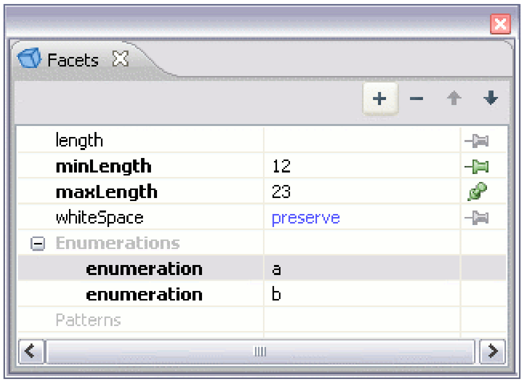
Note

If the selected component is a reference to a component defined in another schema, most properties will be read-only and the actions will be disabled.

The Facets view

The Facets View presents the facets for the selected component if available. If hidden, you can open it from Window → Show View → Other → oXygen → Facets.

Figure 4.65. The Facets view



length		
minLength	12	
maxLength	23	
whiteSpace	preserve	
⊟ Enumerations		
enumeration	a	
enumeration	b	
Patterns		

The default value of a facet is presented in the Facets View with blue. The facets that can't be edited are rendered with gray. The grouping categories (eg: *Enumerations* and *Patterns*) are not editable. If these categories contain at least one child they are rendered with bold. Bold facets are facets with values set explicitly to them.

Important

Usually inherited facets are presented as default in the Facets view but if patterns are inherited from a base type and also specified in the current simple type only the current specified patterns will be presented. You can see the effective pattern value obtained by combining the inherited and the specified patterns as a tooltip on the Patterns category.

Facets for components which do not belong to the current edited schema are read-only but if you double-click them you can choose to open the corresponding schema and edit them.

You can edit a facet by double-clicking or by pressing Enter.For some facets you can choose valid values from a list or you can specify another value. If a facet has an invalid value or a warning, it will be highlighted in the table with the corresponding foreground color. By default facets with errors are presented with red and the facets with warnings with yellow. You can customize the error colors from the Document checking user preferences.

The Facets View has four toolbar actions available also on the contextual menu:

+ Add

Allows you to add a new enumeration or a new pattern.

− Remove

Allows you to remove the value of a facet.

⬆ Move Up

Allows you to move up the current enumeration/pattern in Enumerations/Patterns category.

⬇ Move Down	Allows you to move down the current enumeration/pattern in Enumerations/Patterns category.
Copy	Copy the attribute value.
Open in XML Schema Regular Expressions Builder	Allows you to open the pattern in the XML Schema Regular Expressions Builder

Facets can be fixed to prevent a derivation from modifying its value. To fix a facet value just press the pin button.

Note

If the selected component is a reference to a component defined in another schema, the facets will be read-only and the actions will be disabled.

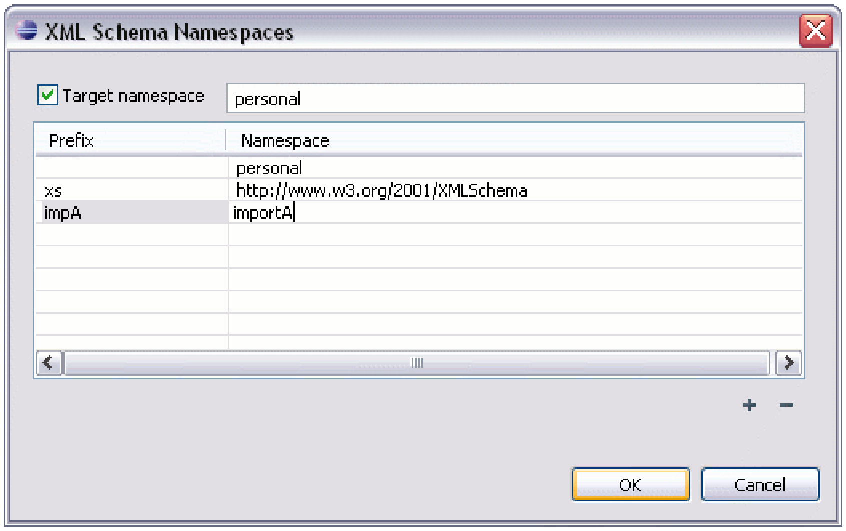
Editing patterns

You can edit regular expressions either be hand or you can right click, choose *Open in XML Schema Regular Expression Builder* and have a full-fledged XML Schema Regular Expression builder to guide you in testing and constructing the pattern.

Edit Schema Namespaces

You can use this dialog to easily set a Target namespace and define namespace mappings for a newly created XML Schema. In the Schema page these namespaces can be modified anytime by choosing **Edit Namespaces** from the contextual menu. Also you can do that by double-clicking on the schema root in the diagram.

Figure 4.66. XML Schema Namespaces



XML Schema Namespaces

☑ Target namespace: personal

Prefix	Namespace
	personal
xs	http://www.w3.org/2001/XMLSchema
impA	importA

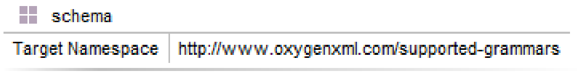
The XML Schema Settings dialog allows you to edit the following information:

- **Target namespace** The Target namespace of the schema.
- **Prefixes** The dialog shows a table with namespaces and the mapped prefixes. You can add a new prefix mapping or remove an already existing one.

Schema Components

Definitions for all XML Schema components are presented together with the symbols used to represent them in the diagram and tables with information about the displayed properties.

xs:schema



schema

Target Namespace	http://www.oxygenxml.com/supported-grammars

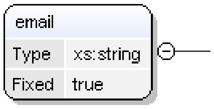
Defines the root element of a schema. A schema document contains representations for a collection of schema components, e.g. type definitions and element declarations, which have a common target namespace. See more info at http://www.w3.org/TR/xmlschema-1/#element-schema.

Schema by default displays the *targetNamespace* property when rendered.

Table 4.1. xs:schema properties

Property Name	Description	Possible Values
Target Namespace	The schema target namespace.	Any URI
Element Form Default	Determining whether local element declarations will be namespace-qualified by default.	qualified, unqualified, [Empty] Default value is unqualified.
Attribute Form Default	Determining whether local attribute declarations will be namespace-qualified by default.	qualified, unqualified, [Empty] Default value is unqualified.
Block Default	Default value of the 'block' attribute of xs:element and xs:complexType.	#all, extension, restriction, substitution, restriction extension, restriction substitution, extension substitution, restriction extension substitution, [Empty]
Final Default	Default value of the 'final' attribute of xs:element and xs:complexType.	#all, restriction, extension, restriction extension, [Empty]
Version	Schema version	Any token
ID	The schema id	Any ID
Component	The edited component name.	Not editable property.
SystemID	The schema system id	Not editable property.

xs:element



email

Type	xs:string
Fixed	true

Defines an element. An element declaration is an association of a name with a type definition, either simple or complex, an (optional) default value and a (possibly empty) set of identity-constraint definitions. See more info at http://www.w3.org/TR/xmlschema-1/#element-element.

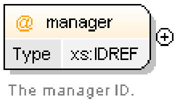
An element by default displays the following properties when rendered in the diagram: *default*, *fixed*, *abstract* and *type*. When referenced or declared locally, the element graphical representation also contains the value for the *minOccurs* and *maxOccurs* properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the element are drawn using dotted lines if the element is optional.

Table 4.2. xs:element properties

Property Name	Description	Possible Values	Mentions
Name	The element name. Always required.	Any NCName for global or local elements, any QName for element references.	If missing, will be displayed as '[element]' in diagram.
Is Reference	When set, the local element is a reference to a global element.	true/false	Appears only for local elements.
Type	The element type.	All declared or built-in types. In addition, the following anonymous types are available: [ST-restriction], [ST-union], [ST-list], [CT-anonymous], [CT-extension SC], [CT-restriction SC], [CT-restriction CC], [CT-extension CC].	For all elements. For references, the value is set in the referred element.
Base Type	The extended/restricted base type.	All declared or built-in types	For elements with complex type, with simple or complex content.
Mixed	Defines if the complex type content model will be mixed.	true/false	For elements with complex type.
Content	The content of the complex type.	simple/complex	For elements with complex type which extends/restricts a base type. It is automatically detected.
Content Mixed	Defines if the complex content model will be mixed.	true/false	For elements with complex type which has a complex content.
Default	Default value of the element. A default value is automatically assigned to the element when no other value is specified.	Any string	The fixed and default attributes are mutually exclusive.
Fixed	A simple content element may be fixed to a specific value using this attribute. A fixed value is also automatically assigned to the element and you cannot specify another value.	Any string	The fixed and default attributes are mutually exclusive.
Min Occurs	Minimum number of occurrences of the element.	A numeric positive value. Default value is 1	Only for references/local elements
Max Occurs	Maximum number of occurrences of the element.	A numeric positive value. Default value is 1	Only for references/local elements
Substitution Group	Qualified name of the head of the substitution group to which this element belongs.	All declared elements	For global and reference elements

Property Name	Description	Possible Values	Mentions
Abstract	Controls whether the element may be used directly in instance XML documents. When set to true, the element may still be used to define content models, but it must be substituted through a substitution group in the instance document.	true/false	For global elements and element references
Form	Defines if the element is "qualified" (i.e., belongs to the target namespace) or "unqualified" (i.e., doesn't belong to any namespace).	unqualified/qualified	Only for local elements
Nillable	When this attribute is set to true, the element can be declared as nil using an xsi:nil attribute in the instance documents.	true/false	For global elements and element references
Block	Controls whether the element can be subject to a type or substitution group substitution. '#all' blocks any substitution, 'substitution' blocks any substitution through substitution groups and 'extension'/'restriction' block any substitution (both through xsi:type and substitution groups) by elements or types, derived respectively by extension or restriction from the type of the element. Its default value is defined by the 'blockDefault' attribute of the parent xs:schema.	#all, restriction, extension,substitution, extension restriction, extension substitution, restriction substitution, restriction extension substitution	For global elements and element references
Final	Controls whether the element can be used as the head of a substitution group for elements whose types are derived by extension or restriction from the type of the element. Its default value is defined by the 'finalDefault' attribute of the parent xs:schema.	#all, restriction, extension, restriction extension, [Empty]	For global elements and element references
ID	The component id.	Any id	For all elements.
Component	The edited component name.	Not editable property.	For all elements.
Namespace	The component namespace.	Not editable property.	For all elements.
System ID	The component system id.	Not editable property.	For all elements.

xs:attribute



Defines an attribute. See more info at http://www.w3.org/TR/xmlschema-1/#element-attribute.

An attribute by default displays the following properties when rendered in the diagram: *default*, *fixed*, *use* and *type*. Connectors to the attribute are drawn using dotted lines if the attribute use is optional. The attribute name is stroked out if prohibited.

Table 4.3. xs:attribute properties

Property Name	Description	Possible Value	Mentions
Name	Attribute name. Always required.	Any NCName for global/local attributes, all declared attributes' QName for references.	For all local or global attributes. If missing, will be displayed as '[attribute]' in the diagram.
Is Reference	When set, the local attribute is a reference.	true/false	For local attributes.
Type	Qualified name of a simple type.	All global simple types and built-in simple types. In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for creating anonymous simple types more easily.	For all attributes. For references, the type is set to the referred attribute.
Default	Default value. When specified, an attribute is added by the schema processor (if it is missing from the instance XML document) and it is given this value. The default and fixed attributes are mutually exclusive.	Any string	For all local or global attributes. For references the value is from the referred attribute.
Fixed	When specified, the value of the attribute is fixed and must be equal to this value. The default and fixed attributes are mutually exclusive.	Any string	For all local or global attributes. For references the value is from the referred attribute.
Use	Possible usage of the attribute. Marking an attribute "prohibited" is useful to exclude attributes during derivations by restriction.	optional, required, prohibited	For local attributes
Form	Specifies if the attribute is qualified (i.e., must have a namespace prefix in the instance XML document) or not. The default value for this attribute is specified by the 'attributeFormDefault' attribute of the xs:schema document element.	unqualified/qualified	For local attributes.
ID	The component id.	Any id	For all attributes.
Component	The edited component name.	Not editable property.	For all attributes.
Namespace	The component namespace.	Not editable property.	For all attributes.
System ID	The component system id.	Not editable property.	For all attributes.

xs:complexType

person_type

Defines a top level complex type.

Complex Type Definitions provide for:

- Constraining element information items by providing Attribute Declarations governing the appearance and content of attributes.
- Constraining element information item children to be empty, or to conform to a specified element-only or mixed content model, or else constraining the character information item children to conform to a specified simple type definition.
- Using the mechanisms of Type Definition Hierarchy to derive a complex type from another simple or complex type.
- Specifying post-schema-validation infoset contributions for elements.
- Limiting the ability to derive additional types from a given complex type.
- Controlling the permission to substitute, in an instance, elements of a derived type for elements declared in a content model to be of a given complex type.

See more info at http://www.w3.org/TR/xmlschema-1/#element-complexType.

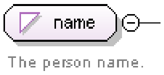
Tip

A complex type which is a base type to another type will be rendered with yellow background.

Table 4.4. xs:complexType properties

Property Name	Description	Possible Values	Mentions
Name	The name of the complex type. Always required.	Any NCName	Only for global complex types. If missing, will be displayed as '[complexType]' in diagram.
Base Type Definition	The name of the extended/restricted types.	Any from the declared simple or complex types.	For complex types with simple or complex content.
Derivation Method	The derivation method.	restriction/ extension	Only when base type is set. If the base type is a simple type, the derivation method is always extension.
Content	The content of the complex type.	simple/ complex	For complex types which extend/restrict a base type. It is automatically detected.
Content Mixed	Specifies if the complex content model will be mixed.	true/false	For complex contents.
Mixed	Specifies if the complex type content model will be mixed.	true/false	For global and anonymous complex types.
Abstract	When set to 'true', this complex type cannot be used directly in the instance documents and needs to be substituted using an 'xsi:type' attribute.	true/false	For global and anonymous complex types.
Block	Controls whether a substitution (either through a 'xsi:type' or substitution groups) can be performed for a complex type, which is an extension or a restriction of the current complex type. This attribute can only block such substitutions (it cannot "unblock" them), which can also be blocked in the element definition. The default value is defined by the 'blockDefault' attribute of xs:schema.	all, extension, restriction, extension restriction, [Empty]	For global complex types.
Final	Controls whether the complex type can be further derived by extension or restriction to create new complex types.	all, extension, restriction, extension restriction, [Empty]	For global complex types.
ID	The component id.	Any id	For all complex types.
Component	The edited component name.	Not editable property.	For all complex types.
Namespace	The component namespace.	Not editable property.	For all complex types.
System ID	The component system id.	Not editable property.	For all complex types.

xs:simpleType



Defines a simple type. A simple type definition is a set of constraints on strings and information about the values they encode, applicable to the normalized value of an attribute information item or of an element information item with no element children. Informally, it applies to the values of attributes and the text-only content of elements. See more info at http://www.w3.org/TR/xmlschema-1/#element-simpleType.

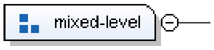
Tip

A simple type which is a base type to another type will be rendered with yellow background.

Table 4.5. xs:simpleType properties

Name	Description	Possible Values	Scope
Name	Simple type name. Always required.	Any NCName.	Only for global simple types. If missing, will be displayed as '[simpleType]' in diagram.
Derivation	The simple type category: restriction, list or union.	restriction,list or union	For all simple types.
Base Type	A simple type definition component. Required if derivation method is set to restriction.	All global simple types and built-in simple types. In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for easily create anonymous simple types.	For global and anonymous simple types with the derivation method set to restriction.
Item Type	A simple type definition component. Required if derivation method is set to list.	All global simple types and built-in simple types(from schema for schema). In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for easily create anonymous simple types.	For global and anonymous simple types with the derivation method set to list. Derivation by list is the process of transforming a simple datatype (named the item type) into a whitespace-separated list of values from this datatype. The item type can be defined inline by adding a simpleType definition as a child element of the list element, or by reference, using the itemType attribute (it is an error to use both).
Member Types	Category for grouping union members.	Not editable property.	For global and anonymous simple types with the derivation method set to union.
Member	A simple type definition component. Required if derivation method is set to union.	All global simple types and built-in simple types(from schema for schema). In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for easily create anonymous simple types.	For global and anonymous simple types with the derivation method set to union. Deriving a simple datatype by union merges the lexical spaces of several simple datatypes (called member types) to create a new simple datatype. The member types can be defined either by reference (through the memberTypes attribute) or embedded as simple datatype local definitions in the xs:union element. Both styles can be mixed.
Final	Blocks any further derivations of this datatype (by list, union, derivation or all).	#all, list, restriction, union, list restriction, list union, restriction union. In addition, [Empty] proposal is present for set empty string as value.	Only for global simple types.
ID	The component id.	Any id.	For all simple types
Component	The name of the edited component.	Not editable property.	Only for global and local simple types
Namespace	The component namespace.	Not editable property.	For global simple types.
System ID	The component system id.	Not editable property.	Not present for built-in simple types..

xs:group



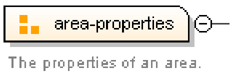
Defines a group of elements to be used in complex type definitions. See more info at http://www.w3.org/TR/xmlschema-1/#element-group.

When referenced the graphical representation also contains the value for the *minOccurs* and *maxOccurs* properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the group are drawn using dotted lines if the group is optional.

Table 4.6. xs:group properties

Property Name	Description	Possible Values	Mentions
Name	The group name. Always required.	Any NCName for global groups, all declared groups for reference.	If missing, will be displayed as '[group]' in diagram.
Min Occurs	Minimum number of occurrences of the group.	A numeric positive value. Default value is 1.	Appears only for reference groups.
Max Occurs	Maximum number of occurrences of the group.	A numeric positive value. Default value is 1.	Appears only for reference groups.
ID	The component id.	Any id	For all groups.
Component	The edited component name.	Not editable property.	For all groups.
Namespace	The component namespace.	Not editable property	For all groups.
System ID	The component system id.	Not editable property.	For all groups.

xs:attributeGroup

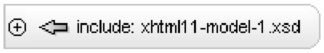


Defines an attribute group to be used in complex type definitions. See more info at http://www.w3.org/TR/xmlschema-1/#element-attributeGroup.

Table 4.7. xs:attributeGroup properties

Property Name	Description	Possible Values	Mentions
Name	Attribute group name. Always required.	Any NCName for global attribute groups, all declared attribute groups for reference.	For all global or referred attribute groups. If missing, will be displayed as '[attributeGroup]' in diagram.
ID	The component id.	Any id	For all attribute groups.
Component	The edited component name.	Not editable property.	For all attribute groups.
Namespace	The component namespace.	Not editable property.	For all attribute groups.
System ID	The component system id.	Not editable property.	For all attribute groups.

xs:include

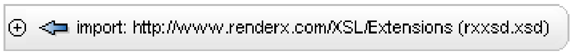


Adds multiple schemas with the same target namespace to a document. See more info at http://www.w3.org/TR/xmlschema-1/#element-include.

Table 4.8. xs:include properties

Property Name	Description	Possible Values
Schema Location	Included schema location.	Any URI
ID	Include ID.	Any ID
Component	The component name.	Not editable property.

xs:import



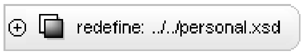
import: http://www.renderx.com/XSL/Extensions (rxxsd.xsd)

Adds multiple schemas with different target namespace to a document. See more info at http://www.w3.org/TR/xmlschema-1/#element-import.

Table 4.9. xs:import properties

Property Name	Description	Possible Values
Schema Location	Imported schema location	Any URI
Namespace	Imported schema namespace	Any URI
ID	Import ID	Any ID
Component	The component name	Not editable property.

xs:redefine



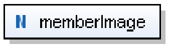
redefine: ../../personal.xsd

Redefines simple and complex types, groups, and attribute groups from an external schema. See more info at http://www.w3.org/TR/xmlschema-1/#element-redefine.

Table 4.10. xs:redefine properties

Property Name	Description	Possible Values
Schema Location	Redefine schema location.	Any URI
ID	Redefine ID	Any ID
Component	The component name.	Not editable property.

xs:notation



N memberImage

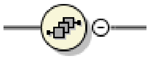
Describes the format of non-XML data within an XML document. See more info at http://www.w3.org/TR/xmlschema-1/#element-notation.

Table 4.11. xs:notation properties

Property Name	Description	Possible values	Mentions
Name	The notation name. Always required.	Any NCName.	If missing, will be displayed as '[notation]' in diagram.
System Identifier	The notation system identifier.	Any URI	Required if public identifier is absent, otherwise optional.
Public Identifier	The notation public identifier.	A Public ID value	Required if system identifier is absent, otherwise optional.
ID	The component id.	Any ID	For all notations.
Component	The edited component name.	Not editable property.	For all notations.
Namespace	The component namespace.	Not editable property.	For all notations.
System ID	The component system id.	Not editable property.	For all notations.

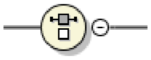
xs:sequence, xs:choice, xs:all

Figure 4.67. An *xs:sequence* in diagram



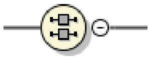
xs:sequence specifies that the child elements must appear in a sequence. Each child element can occur from 0 to any number of times. See more info at http://www.w3.org/TR/xmlschema-1/#element-sequence.

Figure 4.68. An *xs:choice* in diagram



xs:choice allows only one of the elements contained in the declaration to be present within the containing element. See more info at http://www.w3.org/TR/xmlschema-1/#element-choice.

Figure 4.69. An *xs:all* in diagram



xs:all specifies that the child elements can appear in any order. Each child element can occur 0 or 1 time. See more info at http://www.w3.org/TR/xmlschema-1/#element-all.

The compositor graphical representation also contains the value for the *minOccurs* and *maxOccurs* properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the compositor are drawn using dotted lines if the compositor is optional.

Table 4.12. xs:sequence, xs:choice, xs:all properties

Property Name	Description	Possible Values	Mentions
Compositor	Compositor type.	sequence, choice, all.	'all' is only available as a child of a group or complex type.
Min Occurs	Minimum occurrences of compositor.	A numeric positive value. Default is 1.	The property is not present if compositor is 'all' and is child of a group.
Max Occurs	Maximum occurrences of compositor.	A numeric positive value. Default is 1.	The property is not present if compositor is 'all' and is child of a group.
ID	The component id.	Any ID	For all compositors.
Component	The edited component name.	Not editable property.	For all compositors.
System ID	The component system id.	Not editable property.	For all compositors.

xs:any

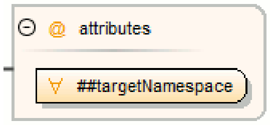
Enables the author to extend the XML document with elements not specified by the schema. See more info at http://www.w3.org/TR/xmlschema-1/#element-any.

The graphical representation also contains the value for the *minOccurs* and *maxOccurs* properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the wildcard are drawn using dotted lines if the wildcard is optional.

Table 4.13. xs:any properties

Property Name	Description	Possible Values
Namespace	The list of allowed namespaces. The namespace attribute expects a list of namespace URIs. In this list, two values have a specific meaning: '##targetNamespace' stands for the target namespace, and '##local' stands for local attributes (without namespaces).	##any, ##other, ##targetNamespace, ##local or anyURI
Process Contents	Type of validation required on the elements allowed for this wildcard.	skip, lax, strict
Min Occurs	Minimum occurrences of any	A numeric positive value. Default is 1.
Max Occurs	Maximum occurrences of any	A numeric positive value. Default is 1.
ID	The component id.	Any ID.
Component	The name of the edited component.	Not editable property.
System ID	The component system id.	Not editable property.

xs:anyAttribute

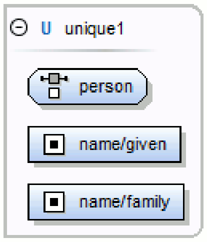


Enables the author to extend the XML document with attributes not specified by the schema. See more info at http://www.w3.org/TR/xmlschema-1/#element-anyAttribute.

Table 4.14. xs:anyAttribute properties

Property Name	Description	Possible Value
Namespace	The list of allowed namespaces. The namespace attribute expects a list of namespace URIs. In this list, two values have a specific meaning: '##targetNamespace' stands for the target namespace, and '##local' stands for local attributes (without namespaces).	##any, ##other, ##targetNamespace, ##local or anyURI
Process Contents	Type of validation required on the elements allowed for this wildcard.	skip, lax, strict
ID	The component id.	Any ID.
Component	The name of the edited component.	Not editable property.
System ID	The component system id.	Not editable property.

xs:unique

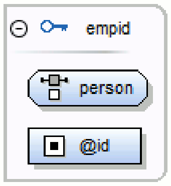


Defines that an element or an attribute value must be unique within the scope. See more info at http://www.w3.org/TR/xmlschema-1/#element-unique.

Table 4.15. xs:unique properties

Property Name	Description	Possible Values
Name	The unique name. Always required.	Any NCName.
ID	The component id.	Any ID.
Component	The edited component name.	Not editable property.
Namespace	The component namespace.	Not editable property.
System ID	The component system id.	Not editable property.

xs:key

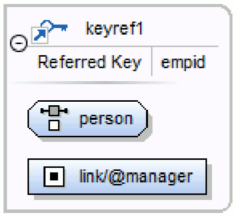


Specifies an attribute or element value as a key (unique, non-nullable, and always present) within the containing element in an instance document. See more info at http://www.w3.org/TR/xmlschema-1/#element-key.

Table 4.16. xs:key properties

Property Name	Description	Possible Value
Name	The key name. Always required.	Any NCName.
ID	The component id.	Any ID.
Component	The edited component name.	Not editable property.
Namespace	The component namespace.	Not editable property.
System ID	The component system id.	Not editable property.

xs:keyRef



Specifies that an attribute or element value correspond to those of the specified key or unique element. See more info at http://www.w3.org/TR/xmlschema-1/#element-keyref.

A keyref by default displays the *Referenced Key* property when rendered.

Table 4.17. xs:keyRef properties

Property Name	Description	Possible Values
Name	The keyref name. Always required.	Any NCName.
Referred Key	The name of referred key	any declared element constraints.
ID	The component id.	Any ID.
Component	The edited component name.	Not editable property.
Namespace	The component namespace.	Not editable property.
System ID	The component system id.	Not editable property.

xs:selector

Specifies an XPath expression that selects a set of elements for an identity constraint. See more info at http://www.w3.org/TR/xmlschema-1/#element-selector.

Table 4.18. xs:selector properties

Property Name	Description	Possible Values
XPath	Relative XPath expression identifying the element on which the constraint applies.	An XPath expression.
ID	The component id.	Any ID.
Component	The edited component name.	Not editable property.
System ID	The component system id.	Not editable property.

xs:field

Specifies an XPath expression that specifies the value used to define an identity constraint. See more info at http://www.w3.org/TR/xmlschema-1/#element-field.

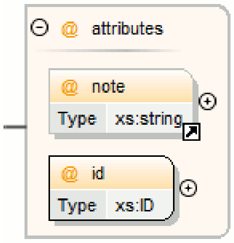
Table 4.19. xs:field properties

Property Name	Description	Possible Values
XPath	Relative XPath expression identifying the field(s) composing the key, key reference, or unique constraint.	An XPath expression.
ID	The component id.	Any ID.
Component	The edited component name.	Not editable property.
System ID	The component system id.	Not editable property.

Constructs used to group schema components

Some schema components are grouped in containers so that they can be more easily identified and classified.

Attributes

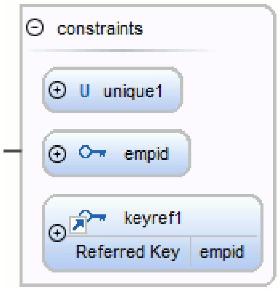


Groups all attributes and attribute groups belonging to a complex type.

Table 4.20. Attributes properties

Property Name	Description	Possible Values
Component	The element for which the attributes are displayed.	Not editable property.
System ID	The component system id.	Not editable property.

Constraints

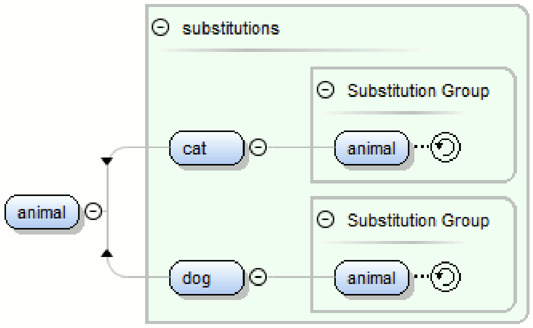


Groups all constraints (xs:key, xs:keyRef or xs:unique) belonging to an element.

Table 4.21. Attributes properties

Property Name	Description	Possible Values
Component	The element for which the constraints are displayed.	Not editable property.
System ID	The component system id.	Not editable property.

Substitutions



Groups all elements which can substitute the current element.

Table 4.22. Attributes properties

Property Name	Description	Possible Values
Component	The element for which the substitutions are displayed.	Not editable property.
System ID	The component system id.	Not editable property.

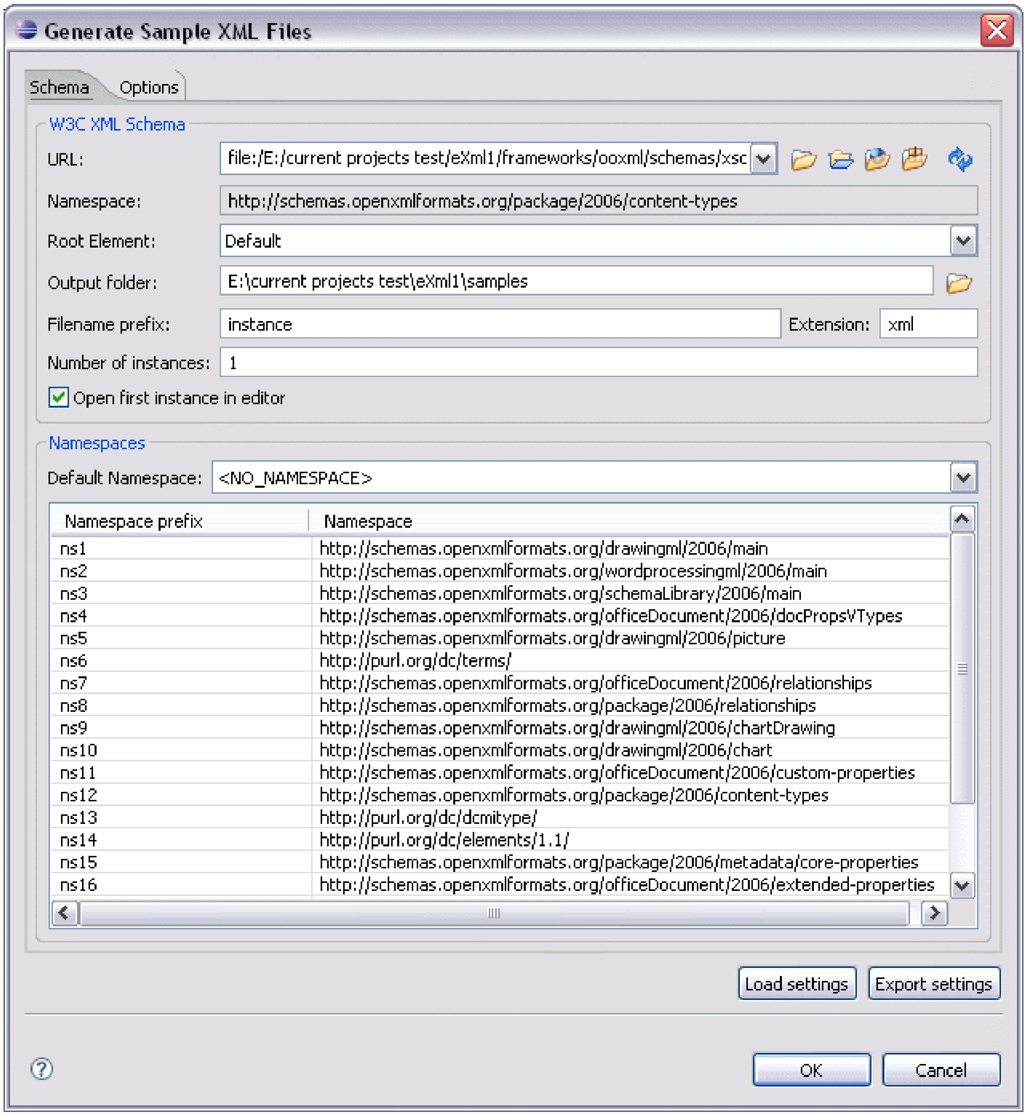
Create an XML Schema from a relational database table

To create an XML Schema from the structure of a relational database table use the special wizard available in the *Tools* menu.

XML Schema Instance Generator

To generate sample XML files from an XML Schema use the *Generate Sample XML Files...* dialog. It is opened with the action XML Tools → Generate Sample XML Files.... The action is available also on the contextual menu from the schema page.

Figure 4.70. The Generate Sample XML Files dialog



URL: file:/E:/current projects test/eXml1/frameworks/ooxml/schemas/xsc

Namespace: http://schemas.openxmlformats.org/package/2006/content-types

Root Element: Default

Output folder: E:\current projects test\eXml1\samples

Filename prefix: instance Extension: xml

Number of instances: 1

☑ Open first instance in editor

Default Namespace: <NO_NAMESPACE>

Namespace prefix	Namespace
ns1	http://schemas.openxmlformats.org/drawingml/2006/main
ns2	http://schemas.openxmlformats.org/wordprocessingml/2006/main
ns3	http://schemas.openxmlformats.org/schemaLibrary/2006/main
ns4	http://schemas.openxmlformats.org/officeDocument/2006/docPropsVTypes
ns5	http://schemas.openxmlformats.org/drawingml/2006/picture
ns6	http://purl.org/dc/terms/
ns7	http://schemas.openxmlformats.org/officeDocument/2006/relationships
ns8	http://schemas.openxmlformats.org/package/2006/relationships
ns9	http://schemas.openxmlformats.org/drawingml/2006/chartDrawing
ns10	http://schemas.openxmlformats.org/drawingml/2006/chart
ns11	http://schemas.openxmlformats.org/officeDocument/2006/custom-properties
ns12	http://schemas.openxmlformats.org/package/2006/content-types
ns13	http://purl.org/dc/dcmitype/
ns14	http://purl.org/dc/elements/1.1/
ns15	http://schemas.openxmlformats.org/package/2006/metadata/core-properties
ns16	http://schemas.openxmlformats.org/officeDocument/2006/extended-properties

Complete the dialog as follows:

URL
Schema's URL. Last used URLs are displayed in the drop-down box.

Namespace
Displays the namespace of the selected schema.

Document root
After the list is selected, a list of elements is displayed in the combo box. The user should choose the root of the XML documents to be generated.

Output folder
Path to the folder where the generated XML instances will be saved.

Filename prefix and Extension
Generated files' names have the following format: prefixN.extension, where *prefix* and *extension* are specified by the user and *N* represents an incremental number from 0 up to *Number of instances - 1*.

Number of instances
The number of XML files to be generated.

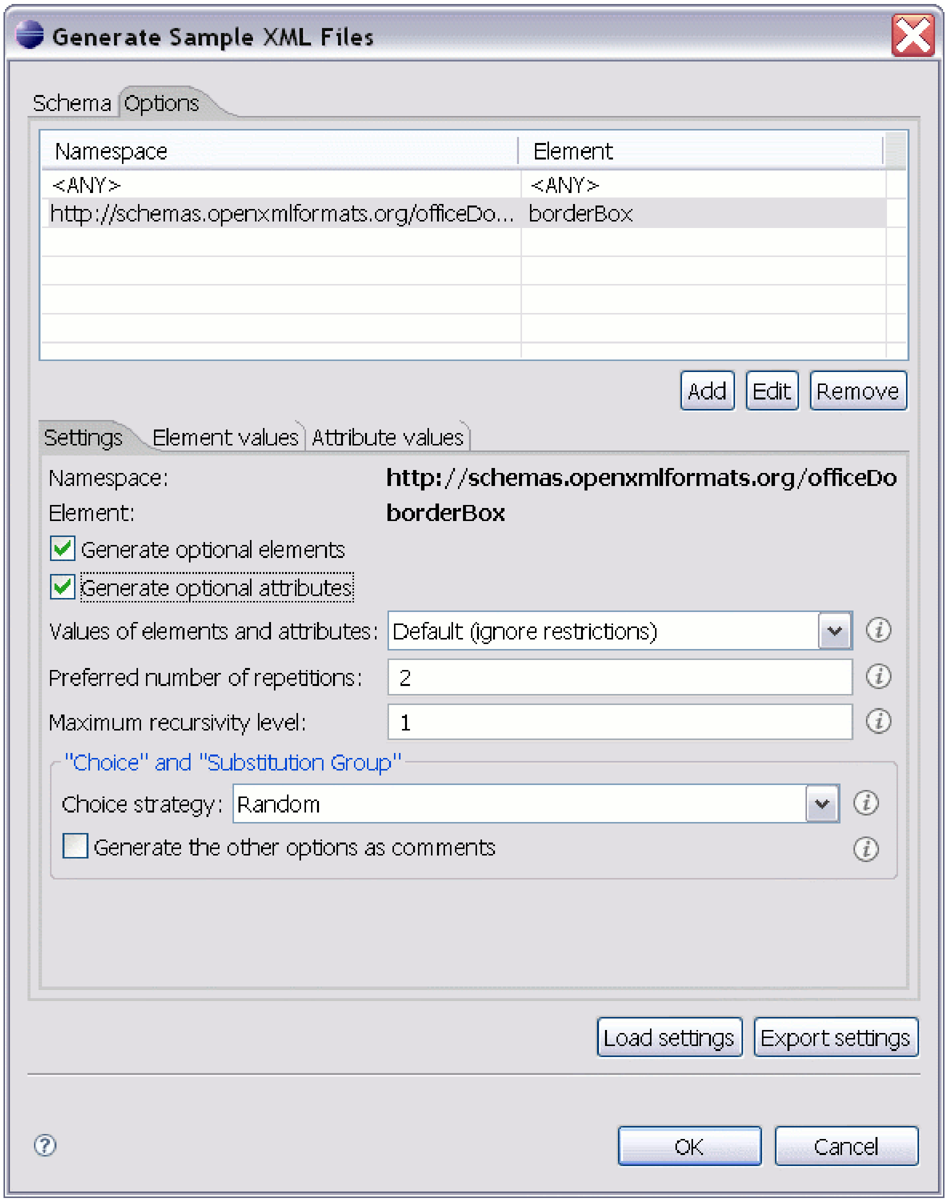
Open first instance in editor — When checked, the first generated XML file will be opened in editor.

Namespaces — Here the user can specify the default namespace as well as the proxies (prefixes) for namespaces.

Load settings / Export settings — The current settings can be saved for further usage with the **Export settings** button, and reloaded when necessary with the **Load settings** button.

The *Options* tab becomes active only after the URL field is filled-in and a schema is detected. It allows the user to set specific options for different namespaces and elements.

Figure 4.71. The Generate Sample XML Files dialog



Namespace	Element
<ANY>	<ANY>
http://schemas.openxmlformats.org/officeDo...	borderBox

Namespace: **http://schemas.openxmlformats.org/officeDo**

Element: **borderBox**

☑ Generate optional elements

☑ Generate optional attributes

Values of elements and attributes: Default (ignore restrictions)

Preferred number of repetitions: 2

Maximum recursivity level: 1

"Choice" and "Substitution Group"

Choice strategy: Random

☐ Generate the other options as comments

Namespace / Element table — Allows the user to define settings for:

	• All elements from all namespaces. This is the default setting and it can also be accessed from Options -> Preferences -> XML / XML Instance Generator. • All elements from a specific namespace. • A specific element from a specific namespace.	
Settings	Generate optional elements	When checked, all elements will be generated, including the optional ones (having the *minOccurs* attribute set to 0 in the schema).
	Generate optional attributes	When checked, all attributes will be generated, including the optional ones (having the *use* attribute set to *optional* in the schema.)
	Values of elements and attributes	Controls the content of generated attributes and elements. Several choices are available: • None - No content is inserted; • Default - Inserts a default value depending of data type descriptor of the respective element/attribute. The default value can be either the data type name or an incremental name of the attribute or element (according to the global option from the *XML instance generator* preferences page). Please note that type restrictions are ignored for this option for generating the values of elements and attributes. For example if an element is of a type that restricts an *xs:string* with the *xs:maxLength* facet in order to allow strings with a maximum length of 3 the XML instance generator tool may generate string element values longer than 3 characters. If you need to generate valid values please use the *Random* option. • Random - Inserts a random value depending of data type descriptor of the respective element/attribute.
	Preferred number of repetitions	Allows the user set the preferred number of repeating elements related with minOccurs and maxOccurs defined in XML Schema.

- If the value set here is between minOccurs and maxOccurs, that value will be used;
- If the value set here is less than minOccurs, the minOccurs value will be used;
- If the value set here is greater than maxOccurs, that value will be used.

Maximum recursivity level

Option to set the maximum allowed depth of the same element in case of recursivity.

Choice strategy

Option to be used in case of xs:choice or substitutionGroup. The possible strategies are:

- First - the first branch of xs:choice or the head element of substitutionGroup will be always used;
- Random - a random branch of xs:choice or a substitute element or the head element of a substitutionGroup will be used.

Generate the other options as comments

Option to generate the other possible choices or substitutions (for xs:choice and substitutionGroup). These alternatives will be generated inside comments groups so you can uncomment them and use later. Use this option with care (for example on a restricted namespace and element) as it may generate large result files.

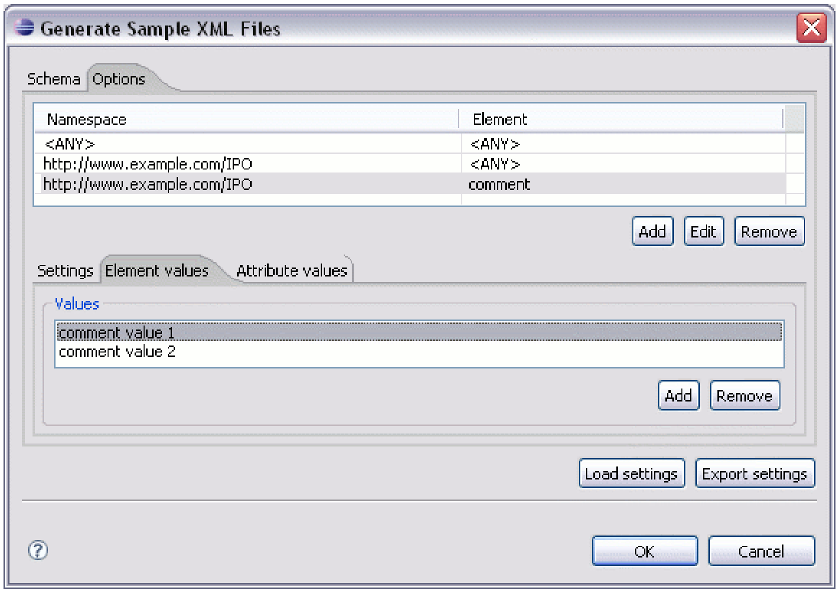
Load settings / Export settings

The current settings can be saved for further usage with the **Export settings** button, and reloaded when necessary with the **Load settings** button.

Element values

The *Element values* tab allows you to add values that will be used to fill the content of elements. If there are more than one value, then the values will be used in a random order.

Figure 4.72. The Element values tab



Generate Sample XML Files

Namespace	Element
<ANY>	<ANY>
http://www.example.com/IPO	<ANY>
http://www.example.com/IPO	comment

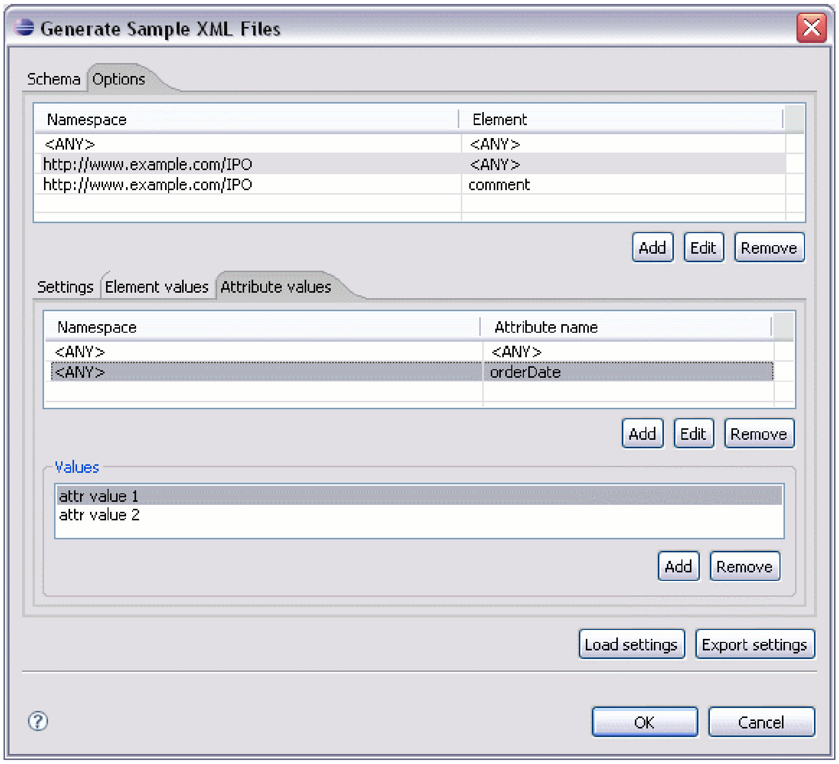
Values

comment value 1
comment value 2

Attribute values

The *Attribute values* tab allows you to add values that will be used to fill the attributes. If there are more than one value, then the values will be used in a random order.

Figure 4.73. The Attribute values tab



Generate Sample XML Files

Namespace	Element
<ANY>	<ANY>
http://www.example.com/IPO	<ANY>
http://www.example.com/IPO	comment

Namespace	Attribute name
<ANY>	<ANY>
<ANY>	orderDate

Values

attr value 1
attr value 2

Running the XML instance generator from command line

The XML instance generator tool can be used also from command line by running the script called `xmlGenerator.bat` (on Windows) / `xmlGenerator.sh` (on Mac OS X / Unix / Linux) located in the <oXygen/> installation folder. The parameters can be set once in the dialog, exported to an XML file on disk with the button "Export settings" and reused from command line. With the exported settings file you can generate the same XML instances from the command line as from the dialog:

```
xmlGenerator.sh -cfgFile myConfigurationFile.xml
```

The script can be integrated in an external batch process launched from the command line. The command line parameter of the script is the relative path to the exported XML settings file. The files which are specified with relative paths in the exported XML settings will be made absolute relative to the directory from where the script is run.

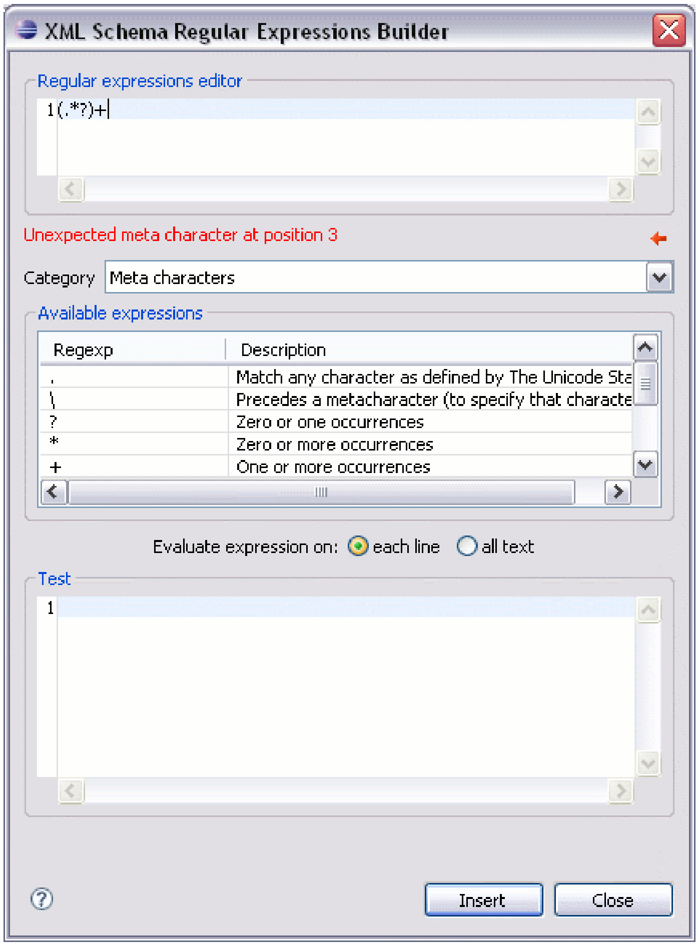
Example 4.14. Example of an XML configuration file saved with *Export settings* button

```
<settings>
    <schemaSystemId>http://www.w3.org/2001/XMLSchema.xsd</schemaSystemId>
    <documentRoot>schema</documentRoot>
    <outputFolder>D:\projects\output</outputFolder>
    <filenamePrefix>instance</filenamePrefix>
    <filenameExtension>xml</filenameExtension>
    <noOfInstances>1</noOfInstances>
    <openFirstInstance>true</openFirstInstance>
    <defaultNamespace>&lt;NO_NAMESPACE></defaultNamespace>
    <element namespace="&lt;ANY>" name="&lt;ANY>">
        <generateOptionalElements>false</generateOptionalElements>
        <generateOptionalAttributes>false</generateOptionalAttributes>
        <valuesForContentType>DEFAULT</valuesForContentType>
        <preferredNumberOfRepetitions>2</preferredNumberOfRepetitions>
        <maximumRecursivityLevel>1</maximumRecursivityLevel>
        <choicesAndSubstitutions strategy="RANDOM"
                generateOthersAsComments="false"/>
        <attribute namespace="&lt;ANY>"
                name="&lt;ANY>">
            <attributeValue>attrValue1</attributeValue>
            <attributeValue>attrValue2</attributeValue>
        </attribute>
    </element>
    <element namespace="&lt;NO_NAMESPACE>"
            name="&lt;ANY>">
        <generateOptionalElements>true</generateOptionalElements>
        <generateOptionalAttributes>true</generateOptionalAttributes>
        <valuesForContentType>DEFAULT</valuesForContentType>
        <preferredNumberOfRepetitions>2</preferredNumberOfRepetitions>
        <maximumRecursivityLevel>1</maximumRecursivityLevel>
        <choicesAndSubstitutions strategy="RANDOM"
                generateOthersAsComments="true"/>
        <elementValue>value1</elementValue>
        <elementValue>value2</elementValue>
        <attribute namespace="&lt;ANY>"
                name="&lt;ANY>">
            <attributeValue>attrValue1</attributeValue>
            <attributeValue>attrValue2</attributeValue>
        </attribute>
    </element>
</settings>
```

XML Schema regular expressions builder

To generate XML Schema regular expressions use the action XML Tools → XML Schema Regular Expressions Builder It will open a dialog which allows you to build and test regular expressions.

Figure 4.74. XML Schema regular expressions builder dialog



XML Schema Regular Expressions Builder
Regular expressions editor

```
1 (.*?)+
```

Unexpected meta character at position 3

Category: Meta characters

Regexp	Description
.	Match any character as defined by The Unicode Sta
\	Precedes a metacharacter (to specify that characte
?	Zero or one occurrences
*	Zero or more occurrences
+	One or more occurrences

Evaluate expression on: ● each line ○ all text

The dialog contains the following sections:

- *Regular expressions editor* - allows you edit the regular expression to be tested and used. Content completion is available and presents a list with all the predefined expressions. It is accessible by pressing **Ctrl + Space**.
- If the edited regular expression is not correct, an error message that contain the position where the error was detected, will be display. If you click on the error message or on the button ← , the error will be highlight inside the regular expression for easily correct them.
- *Category* combo box - here you can choose from several categories of predefined expressions. The selected category influences the displayed expressions in the *Available expressions* table.
- *Available expressions* table - it consists of two columns. The first one presents the regular expressions, the second displays a short description of the expressions. The set of expressions depend on the category selected in the previous combo box. You can add an expression in the *Regular expressions editor* by double-clicking on the expression row in the table You will notice that in the case of *Character categories* and *Block names* the expressions are also listed in complementary format. For example: *\p{Lu}* - Uppercase letters; *\P{Lu}* - Complement of: Uppercase letters.
- *Evaluate expression on* radio buttons - there are available two options: *Evaluate expression on each line* and *Evaluate expression on all text* . If the first option is selected the edited expression will be applied on each line from the *Test* area. If the second option is selected the expression will be applied on the whole text.

- *Test* area - it is a text editor which allows you to enter a text sample on which the regular expression will be applied. The matches of the expression will be highlighted.

After editing and testing your regular expression you can insert it in the current editor. The *Insert* button will become active when an editor is opened in the background and there is an expression in the *Regular expressions editor*.

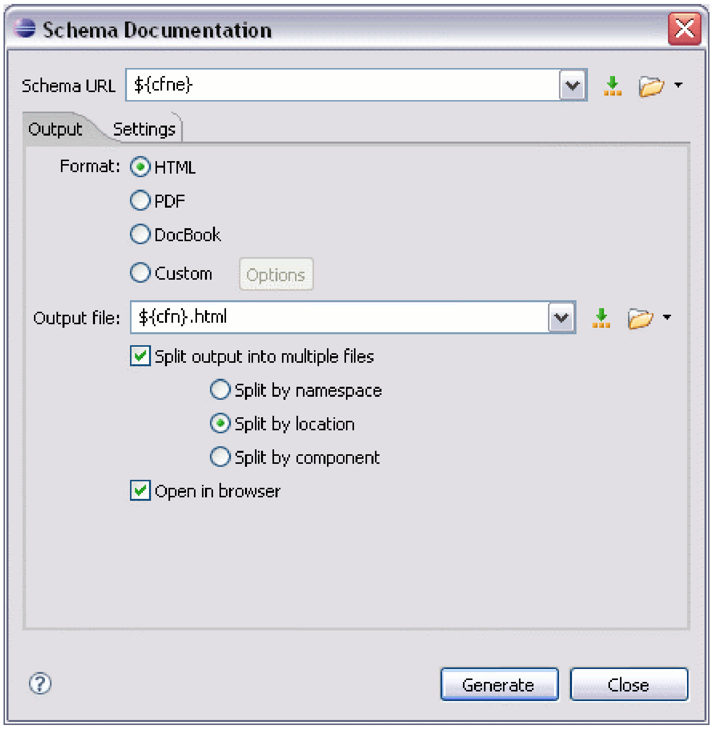
The regular expression builder cannot be used to insert regular expressions in the grid version or the schema version of a document editor. Accordingly the *Insert* button of the dialog will be disabled if the current document is edited in grid mode.

Generating documentation for an XML Schema

<oXygen/> can generate detailed documentation for the components of an XML Schema in HTML, PDF and DocBook XML formats similar with the Javadoc documentation for the components of a Java class. You can select the components and the level of detail. The components are hyperlinked in both HTML and DocBook documents.

To generate documentation for an XML Schema document use the dialog *Schema Documentation*. It is opened with the action XML Tools → Generate Documentation → Schema Documentation... . It can be also opened from the *Navigator* contextual menu: Generate Schema DocumentationThe dialog enables the user to configure a large set of parameters for the process of generating the documentation.

Figure 4.75. The Output panel of the Schema Documentation dialog



Schema URL: ${cfne}

Format: ● HTML ○ PDF ○ DocBook ○ Custom

Output file: ${cfn}.html

☑ Split output into multiple files

○ Split by namespace

● Split by location

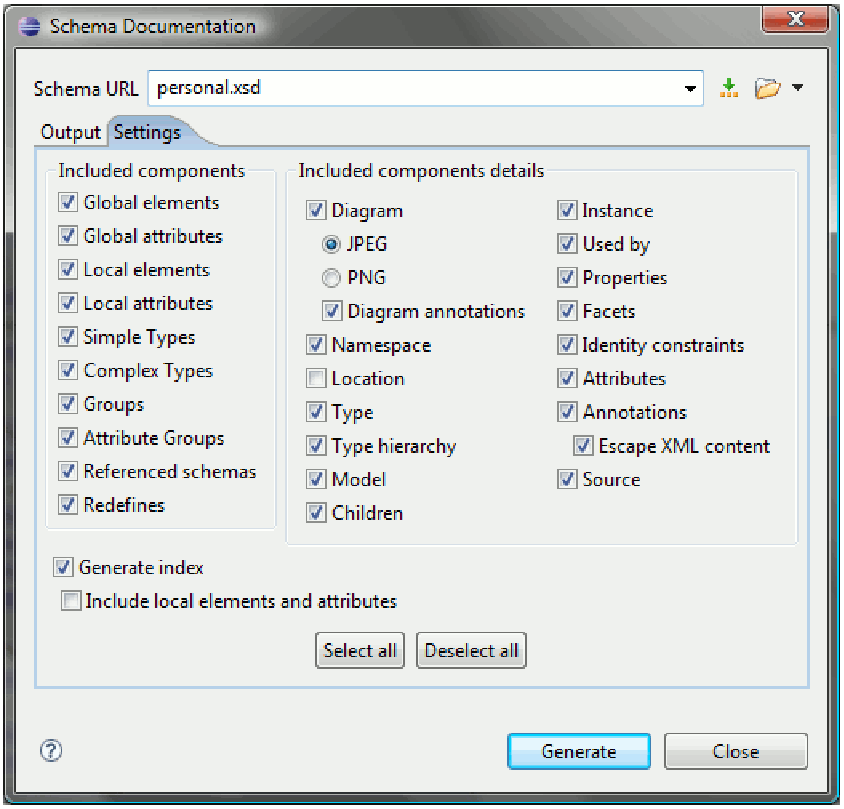
○ Split by component

☑ Open in browser

The *Schema URL* field of the dialog panel must contain the full path to the XML Schema (XSD) file you want to generate documentation for. The schema may be a local or a remote one. You can specify the path to the schema using the editor variables.

You can choose to split the output into multiple files by namespace, location or component.

Figure 4.76. The Settings panel of the Schema Documentation dialog



Schema URL: personal.xsd

Included components
- ☑ Global elements
- ☑ Global attributes
- ☑ Local elements
- ☑ Local attributes
- ☑ Simple Types
- ☑ Complex Types
- ☑ Groups
- ☑ Attribute Groups
- ☑ Referenced schemas
- ☑ Redefines

Included components details
- ☑ Diagram
  - ● JPEG
  - ○ PNG
  - ☑ Diagram annotations
- ☑ Namespace
- ☐ Location
- ☑ Type
- ☑ Type hierarchy
- ☑ Model
- ☑ Children
- ☑ Instance
- ☑ Used by
- ☑ Properties
- ☑ Facets
- ☑ Identity constraints
- ☑ Attributes
- ☑ Annotations
  - ☑ Escape XML content
- ☑ Source

☑ Generate index

☐ Include local elements and attributes

When you generate documentation for a schema you can choose what components to include in the output (global elements, global attributes, local elements, local attributes, simple types, complex types, group, attribute groups, referenced schemas, redefines) and the details to be included in the documentation:

- **Diagram** Show the diagram for each component. You can choose the image format to use for the diagram section.
- **Diagram annotations** The option controls whether or not the annotations of the components presented in the diagram sections should be included.
- **Namespace** Show the namespace for each component.
- **Location** Show the schema location for each component.
- **Type** Show the type of the component if it is not an anonymous one.
- **Type hierarchy** Show the types hierarchy
- **Model** Show the model (sequence, choice, all) presented in BNF form. form. For *xs:all* the model the children are separated by space. For *xs:sequence* the children are separated by comma, for *xs:choice* by |. You can easily check if an element is required or optional.
- **Children** Show the list of all the children of the component
- **Instance** Show an XML instance generated based on each schema element.
- **Used by** Show the list of all the components that refer the component sorted by component type and name.

- **Properties** Show some properties for the component.
- **Facets** Show the facets for each simple type
- **Identity constraints** Show the identity constraints for each element. For each constraint there are presented the name, the type (unique, key, keyref), the refer attribute, the selector and field(s).
- **Attributes** Show the attributes for the component. For each attribute there are presented the name, the type, the fixed or default value, the use and annotation.
- **Annotations** Show the annotations for the component. If you choose **Escape XML Content**, the XML tags are present in the annotations.
- **Source** Show the text schema source for each component.
- **Generate index** Create an index with the components included in the documentation.
- **Include local elements and attributes** If checked, local elements and attributes are included in the documentation index.

These options are persistent between sessions.

Generate documentation in HTML format

The HTML documentation contains images corresponding to the schema definitions as the ones displayed by the schema diagram editor. These images are divided in clickable areas which are linked to the definitions of the clicked names of types or elements. The documentation of a definition includes a *Used By* section with links to the other definitions which refer to it. If the **Escape XML Content** option is unchecked, the HTML or XHTML tags used inside the *xs:documentation* elements of the input XML Schema for formatting the documentation text (for example <b>, <i>, <u>, <ul>, <li>, etc.) are rendered in the generated HTML documentation.

The generated images format is PNG. The image of an XML Schema component contains the graphical representation of that component as it is rendered in the Schema Diagram panel of the <oXygen/>'s XSD editor panel.

Figure 4.77. Schema documentation example

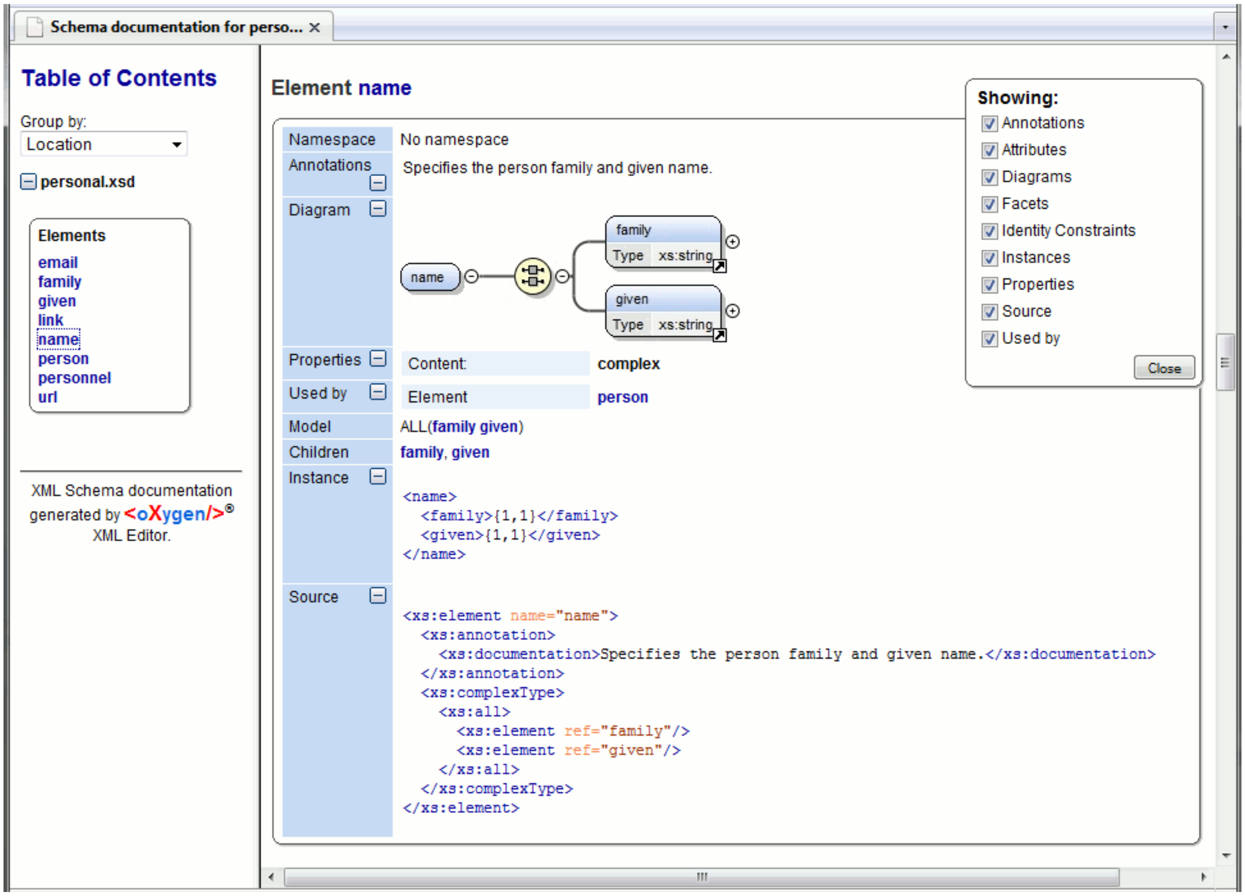


Table of Contents

Group by: Location

⊟ personal.xsd

Elements

- email
- family
- given
- link
- name
- person
- personnel
- url

XML Schema documentation generated by <oXygen/>® XML Editor.

Element name

Namespace	No namespace
Annotations ⊟	Specifies the person family and given name.
Diagram ⊟	
Properties ⊟	Content: **complex**
Used by ⊟	Element **person**
Model	ALL(**family given**)
Children	**family**, **given**
Instance ⊟	
Source ⊟	

Instance:

```
<name>
  <family>{1,1}</family>
  <given>{1,1}</given>
</name>
```

Source:

```
<xs:element name="name">
  <xs:annotation>
    <xs:documentation>Specifies the person family and given name.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:all>
      <xs:element ref="family"/>
      <xs:element ref="given"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

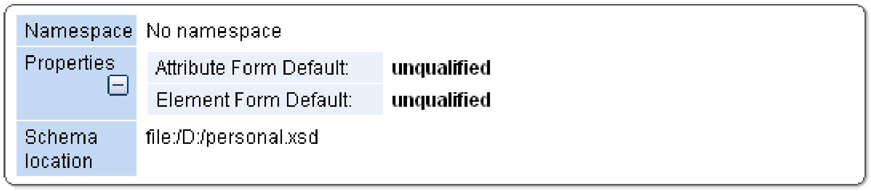
Showing:

- ☑ Annotations
- ☑ Attributes
- ☑ Diagrams
- ☑ Facets
- ☑ Identity Constraints
- ☑ Instances
- ☑ Properties
- ☑ Source
- ☑ Used by

Close

The generated documentation include a table of contents. The contents can be grouped by namespace, location or component type. After the table of contents there is presented some information about the main schema, the imported, included and redefined schemas. This information consists in the schema target namespace, the schema properties (attribute form default, element form default, version) and the schema location.

Figure 4.78. Information about a schema

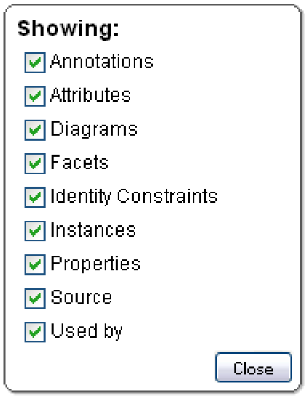


Namespace	No namespace
Properties ⊟	Attribute Form Default: **unqualified** Element Form Default: **unqualified**
Schema location	file:/D:/personal.xsd

If you choose to split the output into multiple files, the table of contents will be displayed in the left frame. The contents will be grouped in the same mode. If you split the output by location, each file contains a schema description and the components that you have chosen to include. If you split the output by namespace, each file contains information about schemas from that namespace and the list with all included components. If you choose to split the output by component, each file will contain information about a schema component.

After the documentation is generated you can collapse details for some schema components. This can be done using the *Showing* view

Figure 4.79. The Showing view

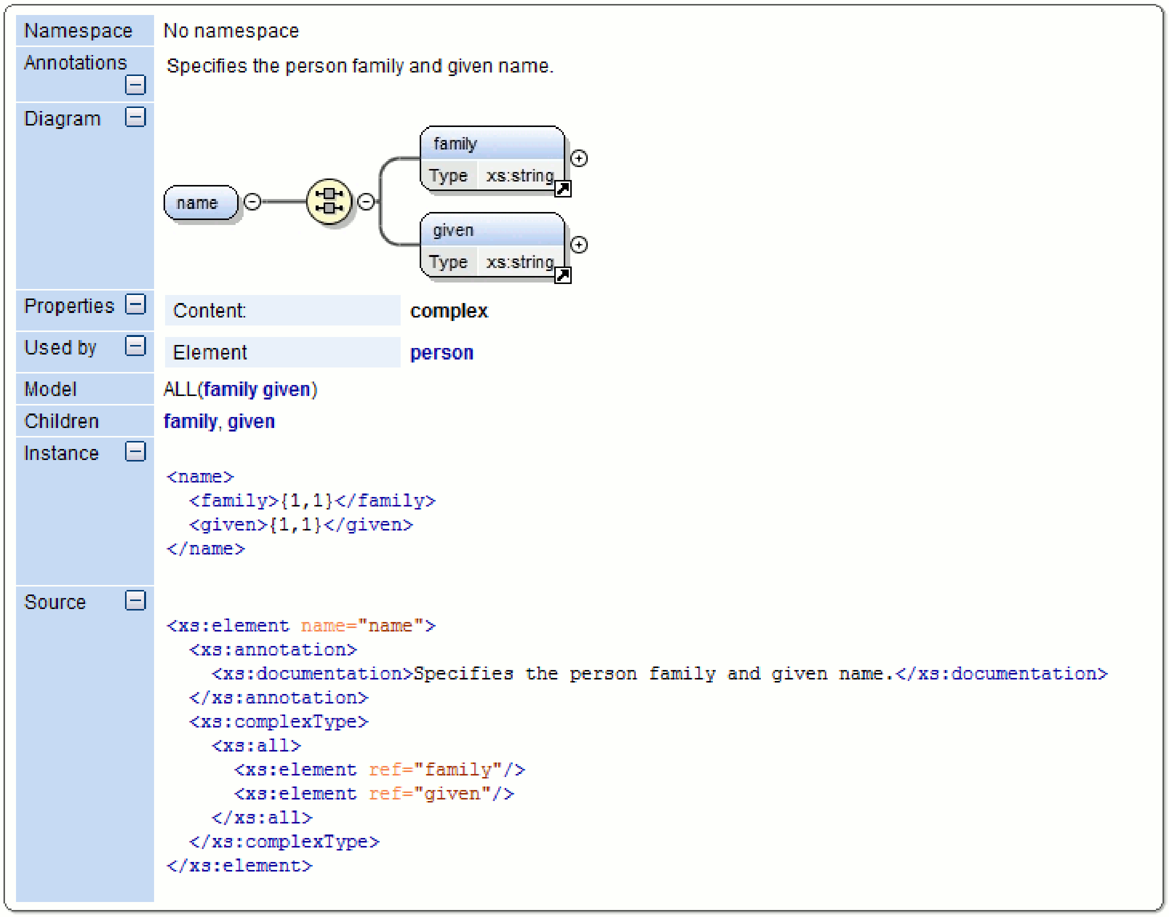


Showing:

- ☑ Annotations
- ☑ Attributes
- ☑ Diagrams
- ☑ Facets
- ☑ Identity Constraints
- ☑ Instances
- ☑ Properties
- ☑ Source
- ☑ Used by

For each component included in the documentation the section presents the component type follow by the component name. For local elements and attributes the name of the component is specified as *parent name/component name*. You can easily go to the parent documentation by clicking on the parent name.

Figure 4.80. Documentation for a schema component



Namespace: No namespace

Annotations: Specifies the person family and given name.

Diagram:

Properties:

Content:	**complex**

Used by:

Element	**person**

Model: ALL(**family given**)

Children: **family**, **given**

Instance:

```
<name>
  <family>{1,1}</family>
  <given>{1,1}</given>
</name>
```

Source:

```
<xs:element name="name">
  <xs:annotation>
    <xs:documentation>Specifies the person family and given name.</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:all>
      <xs:element ref="family"/>
      <xs:element ref="given"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Generate documentation in PDF, DocBook or a custom format

Schema documentation can be also generated in PDF, DocBook or a custom format. You can choose the format from the Schema Documentation Dialog. For the PDF and DocBook formats, the option to split the output in multiple files is disabled.

For PDF the documentation is generated in DocBook format and after that a transformation using the FOP processor is applied to obtain the PDF file. If there are errors during the transformation using the Apache FOP these are presented. To configure the FOP processor see the FO Processors preferences page.

If you generate the documentation in DocBook format you can apply a transformation scenario on the output file, for example one of the scenarios proposed by <oXygen/> (*DocBook PDF* or *DocBook HTML*) or configure your own scenario for it.

For the custom format you can specify your stylesheet to transform the intermediary XML generated in the documentation process. You have to write your stylesheet based on the schema `xsdDocSchema.xsd` from `{INSTALATION_DIRECTORY}/frameworks/schema_documentation`.

Generating documentation from the command line

You can export the settings of the Schema Documentation dialog to an XML file by pressing the "Export settings" button. With the exported settings file you can generate the same documentation from the command line by running the script `schemaDocumentation.bat` (on Windows) / `schemaDocumentation.sh` (on Mac OS X / Unix / Linux) located in the <oXygen/> installation folder. The script can be integrated in an external batch process launched from the command line.

The command line parameter of the script is the relative path to the exported XML settings file. The files which are specified with relative paths in the exported XML settings will be made absolute relative to the directory from where the script is run.

Example 4.15. Example of an XML configuration file

```
<serialized>
    <map>
        <entry>
            <String xml:space="preserve">xsd.documentation.options</String>
            <xsdDocumentationOptions>
                <field name="outputFile">
                    <String xml:space="preserve">${cfn}.html</String>
                </field>
                <field name="splitMethod">
                    <Integer xml:space="preserve">1</Integer>
                </field>
                <field name="openOutputInBrowser">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="format">
                    <Integer xml:space="preserve">1</Integer>
                </field>
                <field name="customXSL">
                    <null/>
                </field>
                <field name="deleteXMLFiles">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeIndex">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeGlobalElements">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeGlobalAttributes">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeLocalElements">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeLocalAttributes">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeSimpleTypes">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeComplexTypes">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeGroups">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeAttributesGroups">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeRedefines">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="includeReferencedSchemas">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsDiagram">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsNamespace">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsLocation">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsType">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsTypeHierarchy">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsModel">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsChildren">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsInstance">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsUsedby">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsProperties">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsFacets">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsAttributes">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsIdentityConstr">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsEscapeAnn">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsSource">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
                <field name="detailsAnnotations">
                    <Boolean xml:space="preserve">true</Boolean>
                </field>
            </xsdDocumentationOptions>
```

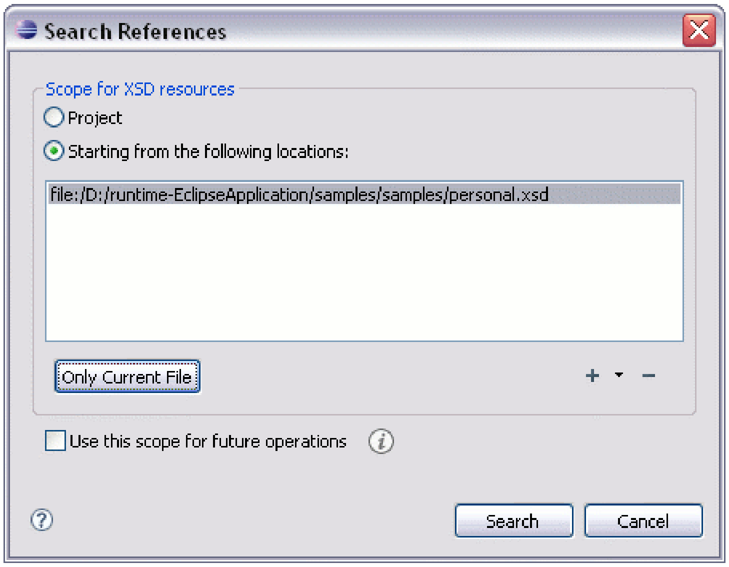
```
        </entry>
    </map>
</serialized>
```

Searching and refactoring actions

All the following actions can be applied on attribute, attributeGroup, element, group, key, unique, keyref, notation, simple or complex types:

- XSD+ → References (**Alt+Shift+S R (Cmd+Alt+S R on Mac OS)**): Searches all references of the item found at current cursor position in the defined scope if any. If a scope is defined but the current edited resource is not part of the range of resources determined by this a warning dialog will be shown and you have the possibility to define another scope, otherwise you have to define a search scope. You can define the search references scope in the following dialog:

  Figure 4.81. Search References dialog

  Search References

  Scope for XSD resources

  ○ Project

  ● Starting from the following locations:

  file:/D:/runtime-EclipseApplication/samples/samples/personal.xsd

  □ Use this scope for future operations

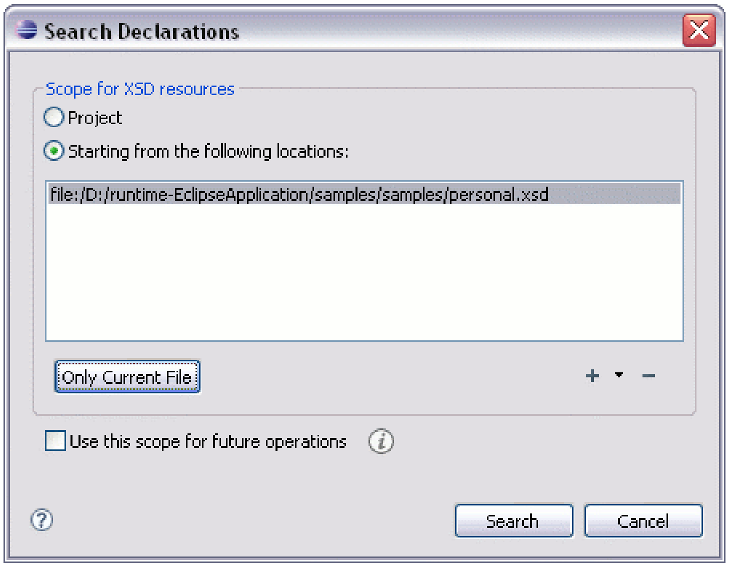
  A search scope may include the project or a collection of files and directories that you have to specify.

  You can decide to automatically use the defined scope for future operations by checking the corresponding checkbox.

- contextual menu of current editor+Search → References in...: Searches all references of the item found at current cursor position in the file(s) that you specify when define a scope in the dialog above.

- XSD+ → Declarations (**Alt+Shift+S D (Cmd+Alt+S D on Mac OS)**): Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined but the current edited resource is not part of the range of resources determined by this a warning dialog will be shown and you have the possibility to define another scope, otherwise you have to define a search scope. You can define the search declarations scope in the following dialog:

Figure 4.82. Search Declarations dialog



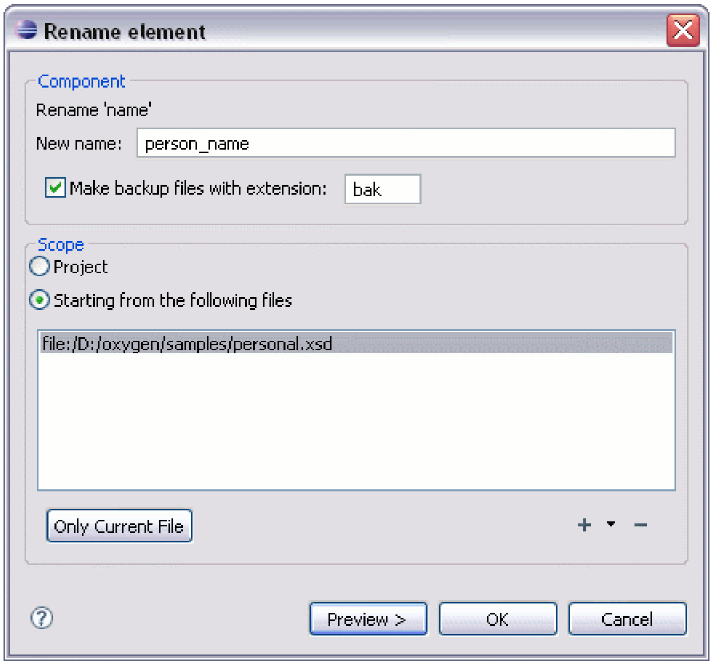
Search Declarations
Scope for XSD resources
○ Project
● Starting from the following locations:
file:/D:/runtime-EclipseApplication/samples/samples/personal.xsd
Only Current File
+ ▾ −
□ Use this scope for future operations
Search
Cancel

A search scope may include the project or a collection of files and directories that you have to specify.

Action is not available in Schema page.

- contextual menu of current editor+Search → Declarations in...: Searches all declarations of the item found at current cursor position in the file(s) that you specify when define a scope in the dialog above. Action is not available in Schema page.
- XSD → Occurrences in File (**Alt+Shift+S O (Cmd+Alt+S O on Mac OS)**): Searches all occurrences of the item at the caret position in the currently edited file.
- contextual menu of current editor+Rename Component...: Rename the selected component. You have to specify the new name for the component and the file(s) affected by the modification in the following dialog:

Figure 4.83. Rename component dialog



Rename element

Component

Rename 'name'

New name: person_name

☑ Make backup files with extension: bak

Scope

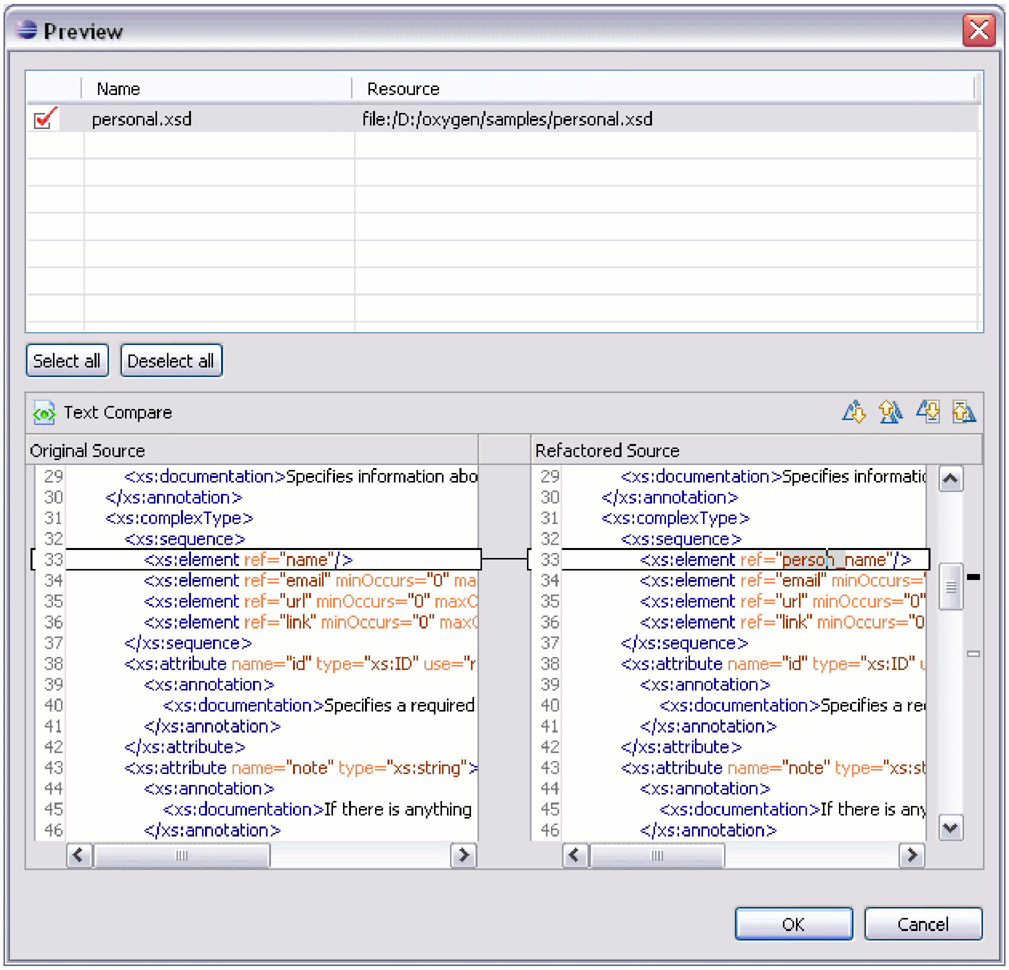
○ Project

● Starting from the following files

file:/D:/oxygen/samples/personal.xsd

You have the possibility to view the files affected by the rename component action if click on preview button. The changes will be shown in the following preview dialog:

Figure 4.84. Preview dialog



	Name	Resource
☑	personal.xsd	file:/D:/oxygen/samples/personal.xsd

Original Source

```
      <xs:documentation>Specifies information abo
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element ref="email" minOccurs="0" ma
        <xs:element ref="url" minOccurs="0" maxC
        <xs:element ref="link" minOccurs="0" maxC
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" use="r
        <xs:annotation>
          <xs:documentation>Specifies a required
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="note" type="xs:string">
        <xs:annotation>
          <xs:documentation>If there is anything
        </xs:annotation>
```

Refactored Source

```
      <xs:documentation>Specifies informati
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="person_name"/>
        <xs:element ref="email" minOccurs="
        <xs:element ref="url" minOccurs="0"
        <xs:element ref="link" minOccurs="0
      </xs:sequence>
      <xs:attribute name="id" type="xs:ID" u
        <xs:annotation>
          <xs:documentation>Specifies a re
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="note" type="xs:st
        <xs:annotation>
          <xs:documentation>If there is any
        </xs:annotation>
```

Resource Hierarchy/Dependencies View

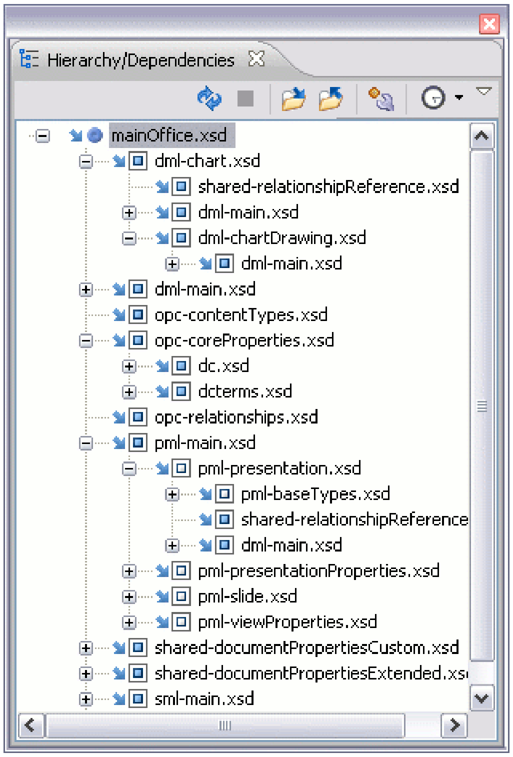
The Resource Hierarchy/Dependencies view allows you to easily see the hierarchy/dependencies for a schema. You can open the view from Window → Show View → Other → oXygen → Resource Hierarchy/Dependencies.

This view is useful for example when you want to start from an XML Schema (XSD) file and build and review the hierarchy of all the other XSD files that are imported, included or redefined in the given XSD file. Also the same view is able to build the inverse tree structure, that is the structure of all other XSD files that import, include or redefine the given XSD file. The scope of the search is configurable: the current Oxygen project, a set of local folders, etc.

The view can build similar tree structures for a RELAX NG schema, a NVDL schema or an XSLT stylesheet.

The build process for the hierarchy view is started with the action **Resource Hierarchy** available on the contextual menu.

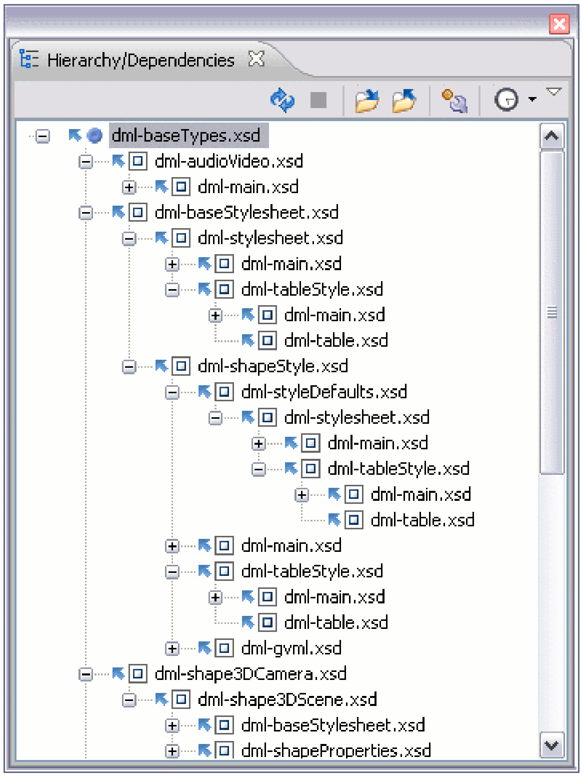
Figure 4.85. Resource Hierarchy/Dependencies view - hierarchy for mainOffice.xsd



```
mainOffice.xsd
    dml-chart.xsd
        shared-relationshipReference.xsd
        dml-main.xsd
        dml-chartDrawing.xsd
            dml-main.xsd
    dml-main.xsd
    opc-contentTypes.xsd
    opc-coreProperties.xsd
        dc.xsd
        dcterms.xsd
    opc-relationships.xsd
    pml-main.xsd
        pml-presentation.xsd
            pml-baseTypes.xsd
            shared-relationshipReference
            dml-main.xsd
        pml-presentationProperties.xsd
        pml-slide.xsd
        pml-viewProperties.xsd
    shared-documentPropertiesCustom.xsd
    shared-documentPropertiesExtended.xs
    sml-main.xsd
```

The build process for the dependencies view is started with the action **Resource Dependencies** available on the contextual menu.

Figure 4.86. Resource Hierarchy/Dependencies view - dependencies for dml-baseTypes.xsd



```
dml-baseTypes.xsd
    dml-audioVideo.xsd
        dml-main.xsd
    dml-baseStylesheet.xsd
        dml-stylesheet.xsd
            dml-main.xsd
            dml-tableStyle.xsd
                dml-main.xsd
                dml-table.xsd
        dml-shapeStyle.xsd
            dml-styleDefaults.xsd
                dml-stylesheet.xsd
                    dml-main.xsd
                    dml-tableStyle.xsd
                        dml-main.xsd
                        dml-table.xsd
            dml-main.xsd
            dml-tableStyle.xsd
                dml-main.xsd
                dml-table.xsd
            dml-gvml.xsd
    dml-shape3DCamera.xsd
        dml-shape3DScene.xsd
            dml-baseStylesheet.xsd
            dml-shapeProperties.xsd
```

In the Resource Hierarchy/Dependencies view you have several actions in the toolbar:

Refresh the hierarchy/dependencies structure.

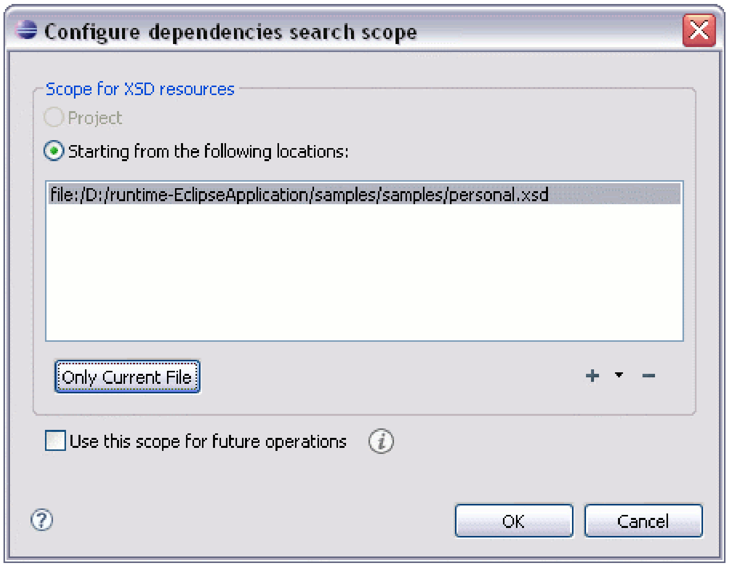
Allows you to stop the hierarchy/dependencies computing.

Allows you to choose a schema to compute the hierarchy structure.

Allows you to choose a schema to compute the dependencies structure.

Allows you to configure a scope for compute the dependencies structure in the following dialog:

Figure 4.87. Configure dependencies search scope



Configure dependencies search scope

Scope for XSD resources

○ Project

● Starting from the following locations:

file:/D:/runtime-EclipseApplication/samples/samples/personal.xsd

□ Use this scope for future operations

You can decide to automatically use the defined scope for future operations by checking the corresponding checkbox.

Allows you to repeat a previous dependencies computation.

On the contextual menu you have also some actions like:

- **Open** Open the schema. Also you can open the schema by a double-click on the hierarchy/dependencies structure.
- **Copy location** Copy the location of the schema.
- **Show Resource Hierarchy** Show the hierarchy for the selected schema.
- **Show Resource Dependencies** Show the dependencies for the selected schema.
- **Expand All** Expand all the children of the selected schema from the hierarchy/dependencies structure.
- **Collapse All** Collapse all the children of the selected schema from the hierarchy/dependencies structure.

Tip

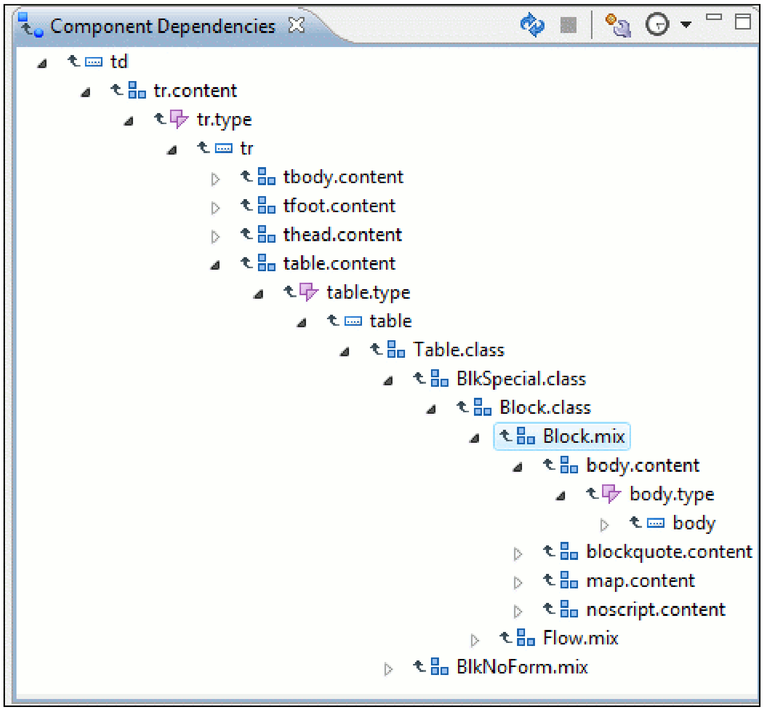
When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon

Component Dependencies View

The Component Dependencies view allows you to easily see the dependencies for a selected schema component. You can open the view from Window → Show View → Other → oXygen → Component Dependencies.

If you want to see the dependencies of a schema component just select the desired schema component in the editor and choose the **Component Dependencies** action from the contextual menu. The action is available for all named components (element, attribute, etc).

Figure 4.88. Component Dependencies view - hierarchy for xhtml11.xsd



```
td
  tr.content
    tr.type
      tr
        tbody.content
        tfoot.content
        thead.content
        table.content
          table.type
            table
              Table.class
                BlkSpecial.class
                  Block.class
                    Block.mix
                      body.content
                        body.type
                          body
                      blockquote.content
                      map.content
                      noscript.content
                    Flow.mix
                BlkNoForm.mix
```

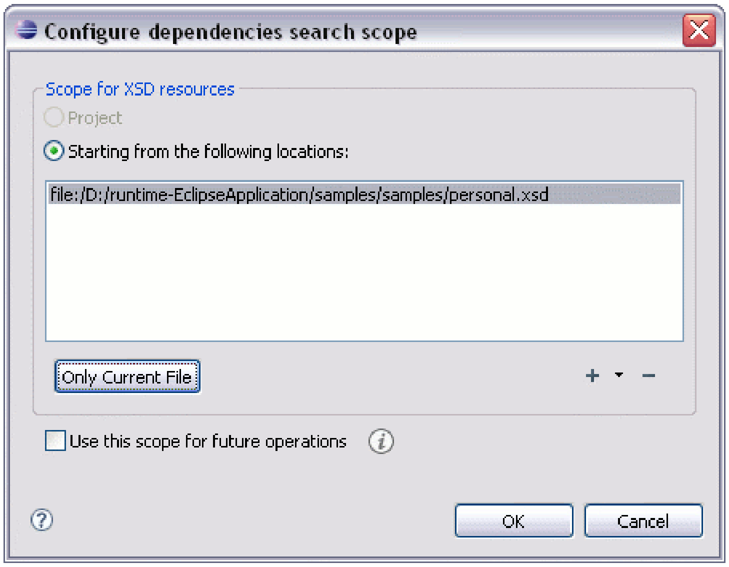
In the Component Dependencies view you have several actions in the toolbar:

Refresh the dependencies structure.

Allows you to stop the dependencies computing.

Allows you to configure a search scope for compute the dependencies structure in the following dialog:

Figure 4.89. Configure dependencies search scope



Scope for XSD resources

○ Project

● Starting from the following locations:

file:/D:/runtime-EclipseApplication/samples/samples/personal.xsd

□ Use this scope for future operations

You can decide to automatically use the defined scope for future operations by checking the corresponding checkbox.

Allows you to repeat a previous dependencies computation.

On the contextual menu you have also some actions like:

- **Go to First Reference** selects the first reference of the referred component from the current selected component in the dependencies tree.
- **Go to Component** Shows the definition of the current selected component in the dependencies tree.

Tip

If a component contains multiple references to another a small table is shown containing all references.

When a recursive reference is encountered it is marked with a special icon

Linking between development and authoring

The Author page is available on the XML Schema editor allowing to edit the annotations visually and presenting a really nice and compact view of the XML Schema, with support for links on included/imported schemas. Embedded Schematron is also supported. See more details here.

Editing Relax NG schemas

<oXygen/> provides a special type of editor for Relax NG schemas. This editor presents the usual text view of an XML document synchronized in real time with an outline view. The outline view has two display modes: the standard outline mode and the components mode.

Relax NG schema diagram

Introduction

<oXygen/> provides a simple, expressive and easy to read Schema Diagram View for Relax NG schemas.

With this new feature you can easily develop complex schemas, print them on multiple pages or save them as JPEG, PNG and BMP images. It helps both schema authors in developing the schema and content authors that are using the schema to understand it.

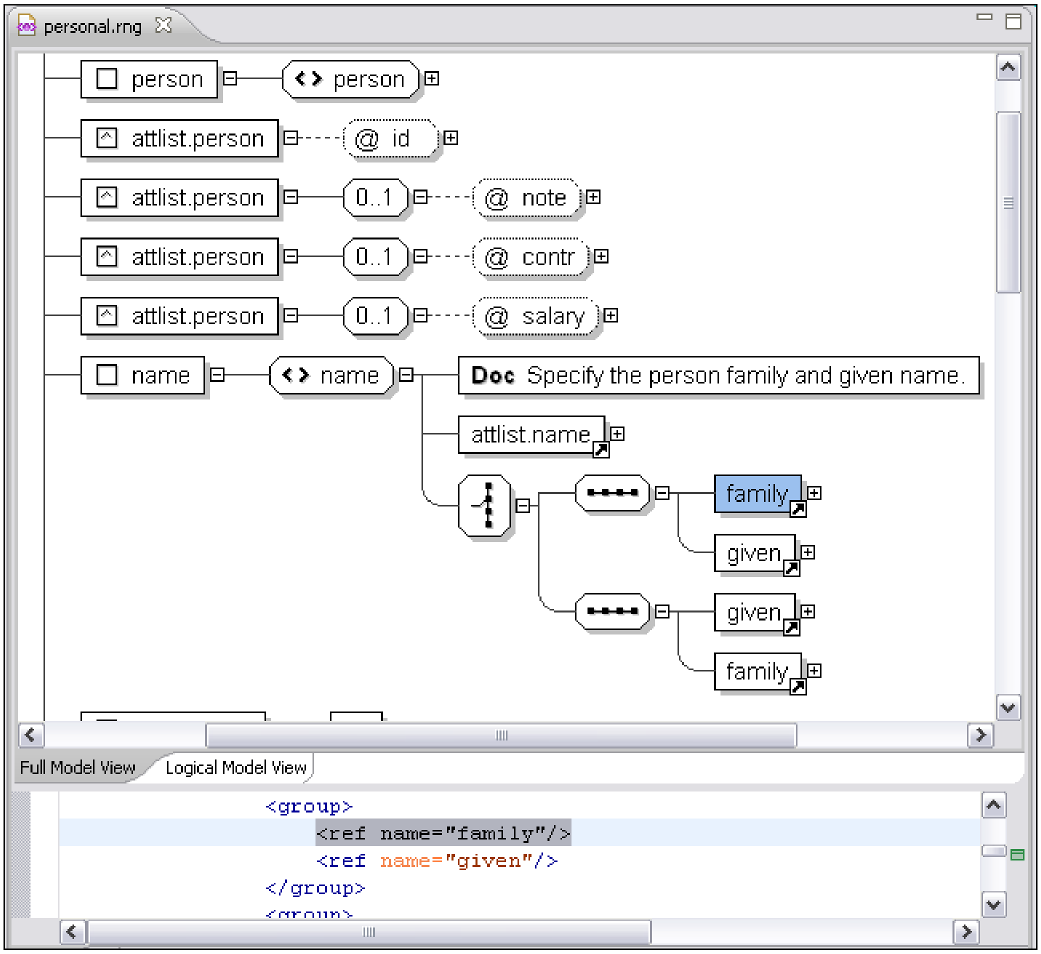
<oXygen/> is the only XML Editor to provide a side by side source and diagram presentation and have them synchronized in real-time:

- the changes you make in the Editor will immediately be visible in the Diagram (no background parsing).
- changing the selected element in the diagram will select the underlaying code in the source editor.

Full model view

When you create a new schema document or open an existing one the Editor Panel is divided in two sections: one containing the Schema Diagram and the second the source code. The Diagram View has two tabbed panes offering a Full Model View and a Logical Model View.

Figure 4.90. Relax NG schema editor - full model view



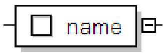
```
        <group>
            <ref name="family"/>
            <ref name="given"/>
        </group>
        <group>
```

The following references can be expanded in place: patterns, includes and external references. This coupled with the synchronization support makes the schema navigation easy.

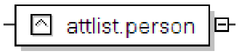
All the element and attribute names are editable: double-click on any name to start editing it.

The symbols used in the schema diagram

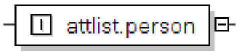
The Full Model View renders all the Relax NG Schema patterns with intuitive symbols:



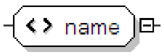
a *define* pattern with the *name* attribute having the value equal to the string from the rectangle



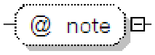
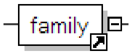
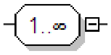
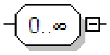
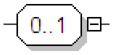
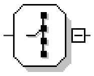
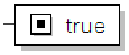
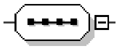
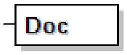
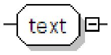
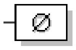
a *define* pattern with the *combine* attribute having the value *interleave* and the *name* attribute having the value equal to the string from the rectangle



a *define* pattern with the *combine* attribute having the value *choice* and the *name* attribute having the value equal to the string from the rectangle



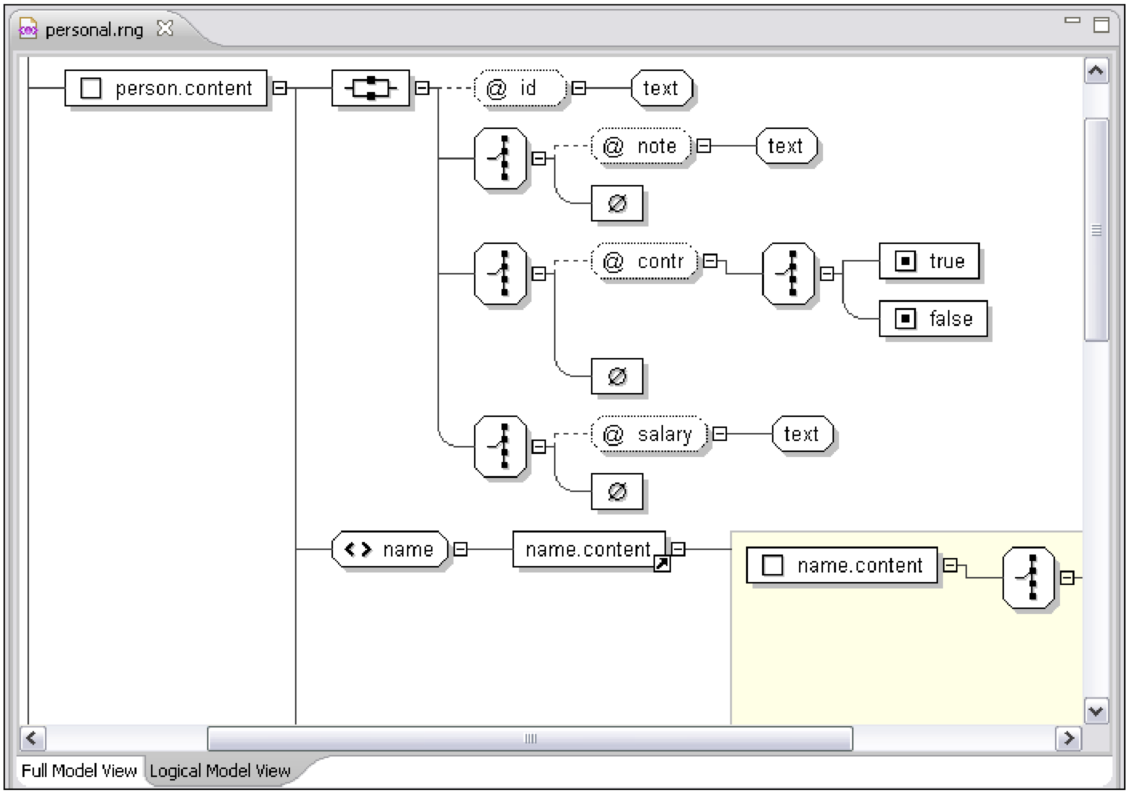
an *element* pattern with the *name* attribute having the value equal to the string from the rectangle

	an *attribute* pattern with the *name* attribute having the value equal to the string from the rectangle
	a *ref* pattern with the *name* attribute having the value equal to the string from the rectangle
	a *oneOrMore* pattern
	a *zeroOrMore* pattern
	an *optional* pattern
	a *choice* pattern
	a *value* pattern, used for example inside a *choice* pattern
	a *group* pattern
	a pattern from the Relax NG Annotations namespace (http://relaxng.org/ns/compatibility/annotations/1.0) which is treated as a documentation element in a Relax NG schema
	a *text* pattern
	an *empty* pattern

Logical model view

The Logical Model View presents the compiled schema which is a single pattern. The patterns that form the element content are defined as a top level pattern with a generated name. The name is generated depending of the name class of the elements.

Figure 4.91. Logical Model View for a Relax NG schema



Actions available in the diagram view

The contextual menu offers some actions:

- **Append child** Append a child to the selected component.
- **Insert Before** Insert a component before the selected component.
- **Insert After** Insert a component after the selected component.
- **Edit attributes** Edit the attributes of the selected component.
- **Remove** Remove the selected component
- **Show only the selected component** Depending on its state(selected/not selected), the selected component is the single component shown in the diagram or all the diagram components are shown.
- **Show Annotations** Depending on its state (selected/not selected), the documentation nodes are shown or hidden.
- **Auto expand to references** This option controls how the schema diagram is automatically expanded. For instance if you select it and then edit a top level element or you make a refresh, the diagram will be expanded until it reaches referred components. If this is left unchecked, only the first level of the diagram is expanded, showing the top level elements.

  For large schemas, the editor disables this option automatically.
- **Collapse Children** Collapse the children of the selected view

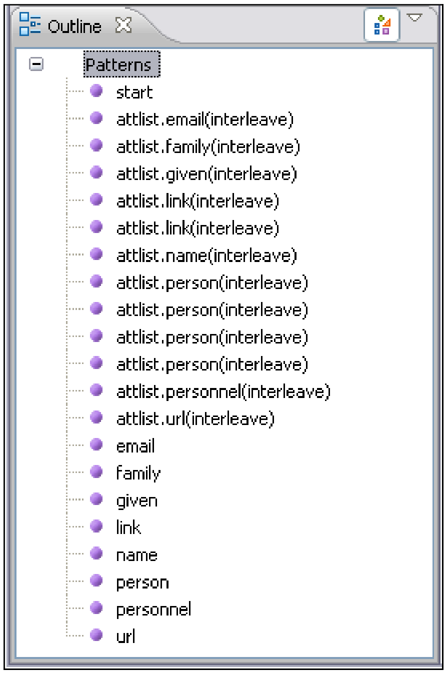
- **Expand Children** Expand the children of the selected view.
- **Print Selection...** Print the selected view.
- **Save Selection as Image...** Save the current selection as JPEG, BMP or PNG Image.
- **Refresh** Refreshes the Schema Diagram according to the changes in your code (changes in your imported documents or those that are not reflected automatically in the compiled schema).

If the schema is not valid you will see an error message in the Logical Model View instead of the diagram.

Relax NG Outline view

The Relax NG Outline View presents a list with the patterns that appear in the diagram in both the Full Model View and Logical Model View cases. It allows a quick access to a component by knowing its name. It can be opened from Window → Show View → Other → oXygen → Outline. You can switch to the standard outline by pressing the button.

Figure 4.92. Outline view for Relax NG



Outline

Patterns
- start
- attlist.email(interleave)
- attlist.family(interleave)
- attlist.given(interleave)
- attlist.link(interleave)
- attlist.link(interleave)
- attlist.name(interleave)
- attlist.person(interleave)
- attlist.person(interleave)
- attlist.person(interleave)
- attlist.person(interleave)
- attlist.personnel(interleave)
- attlist.url(interleave)
- email
- family
- given
- link
- name
- person
- personnel
- url

Relax NG editor specific actions

The list of actions specific for the Relax NG (full syntax) editor of <oXygen/> is:

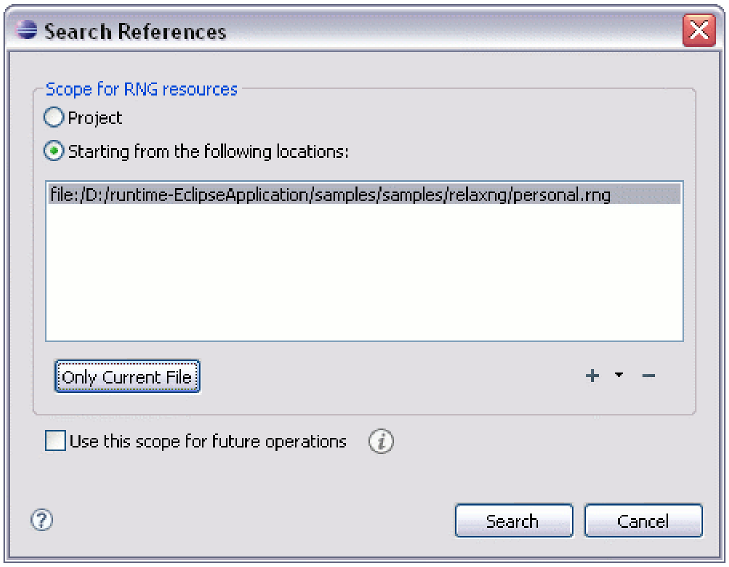
- contextual menu of current editor → Show Definition : move the cursor to the definition of the current element in this Relax NG (full syntax) schema.

Searching and refactoring actions

All the following actions can be applied on *ref* and *parentRef* parameters only.

- RNG+ → References (**Alt+Shift+S R (Cmd+Alt+S R on Mac OS)**): Searches all references of the item found at current cursor position in the defined scope if any. If a scope is defined but the current edited resource is not part of the range of resources determined by this a warning dialog will be shown and you have the possibility to define another scope, otherwise you have to define a search scope. You can define the search references scope in the following dialog:

  Figure 4.93. Search References dialog

  Search References

  Scope for RNG resources

  ○ Project

  ● Starting from the following locations:

  file:/D:/runtime-EclipseApplication/samples/samples/relaxng/personal.rng

  □ Use this scope for future operations

  A search scope may include the project or a collection of files and directories that you have to specify.

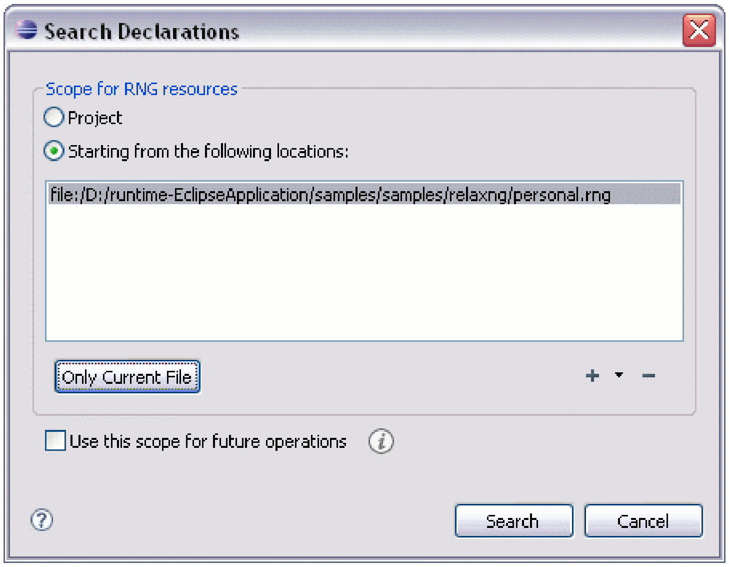
  You can decide to automatically use the defined scope for future operations by checking the corresponding checkbox.

- contextual menu of current editor+Search → References in...: Searches all references of the item found at current cursor position in the file(s) that you specify when define a scope in the dialog above.

All the following actions can be applied on named *define* parameters only.

- RNG+ → Declarations (**Alt+Shift+S D (Cmd+Alt+S D on Mac OS)**): Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined but the current edited resource is not part of the range of resources determined by this a warning dialog will be shown and you have the possibility to define another scope, otherwise you have to define a search scope. You can define the search declarations scope in the following dialog:

Figure 4.94. Search Declarations dialog



○ Project

● Starting from the following locations:

file:/D:/runtime-EclipseApplication/samples/samples/relaxng/personal.rng

□ Use this scope for future operations

A search scope may include the project or a collection of files and directories that you have to specify.

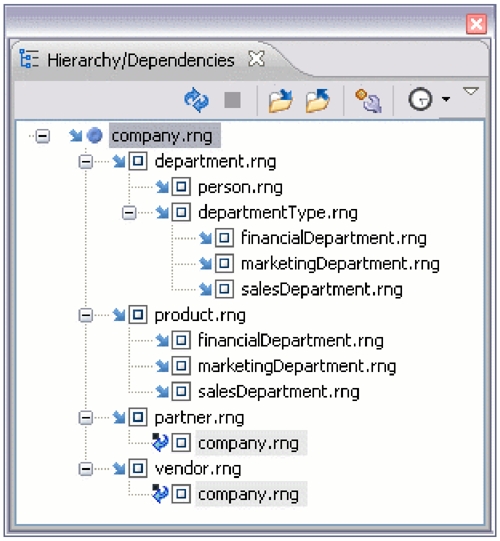
- contextual menu of current editor+Search → Declarations in...: Searches all declarations of the item found at current cursor position in the file(s) that you specify when define a scope in the dialog above.
- RNG → Occurrences in File: Searches all occurrences of the item at the caret position in the currently edited file.
- contextual menu of current editor+Refactoring+Rename Component...: Rename the selected component. You have to specify the new name for the component and the file(s) affected by the modification as described for XML Schema

Resource Hierarchy/Dependencies View

The Resource Hierarchy/Dependencies view allows you to easily see the hierarchy/dependencies for a schema. You can open the view from Window → Show View → Other → oXygen → Resource Hierarchy/Dependencies.

If you want to see the hierarchy of a schema just select the desired schema in the project view and choose **Resource Hierarchy** from the contextual menu.

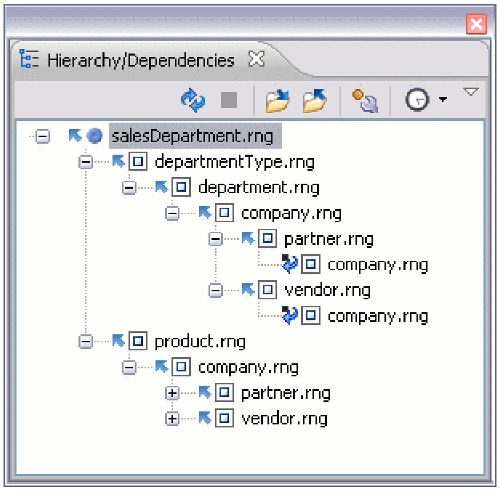
Figure 4.95. Resource Hierarchy/Dependencies view - hierarchy for company.rng



- company.rng
  - department.rng
    - person.rng
    - departmentType.rng
      - financialDepartment.rng
      - marketingDepartment.rng
      - salesDepartment.rng
  - product.rng
    - financialDepartment.rng
    - marketingDepartment.rng
    - salesDepartment.rng
  - partner.rng
    - company.rng
  - vendor.rng
    - company.rng

If you want to see the dependencies of a schema just select the desired schema in the project view and choose **Resource Dependencies** from the contextual menu.

Figure 4.96. Resource Hierarchy/Dependencies view - dependencies for salesDepartment.rng



- salesDepartment.rng
  - departmentType.rng
    - department.rng
      - company.rng
        - partner.rng
          - company.rng
        - vendor.rng
          - company.rng
  - product.rng
    - company.rng
      - partner.rng
      - vendor.rng

In the Resource Hierarchy/Dependencies view you have several actions in the toolbar:

Refresh the hierarchy/dependencies structure.

■ Allows you to stop the hierarchy/dependencies computing.

Allows you to choose a schema to compute the hierarchy structure.

Allows you to choose a schema to compute the dependencies structure.

Allows you to configure a scope for compute the dependencies structure.

Allows you to repeat a previous dependencies computation.

On the contextual menu you have also some actions like:

- **Open** Open the schema. Also you can open the schema by a double-click on the hierarchy/dependencies structure.
- **Copy location** Copy the location of the schema.
- **Show Resource Hierarchy** Show the hierarchy for the selected schema.
- **Show Resource Dependencies** Show the dependencies for the selected schema.
- **Expand All** Expand all the children of the selected schema from the hierarchy/dependencies structure.
- **Collapse All** Collapse all the children of the selected schema from the hierarchy/dependencies structure.

Tip

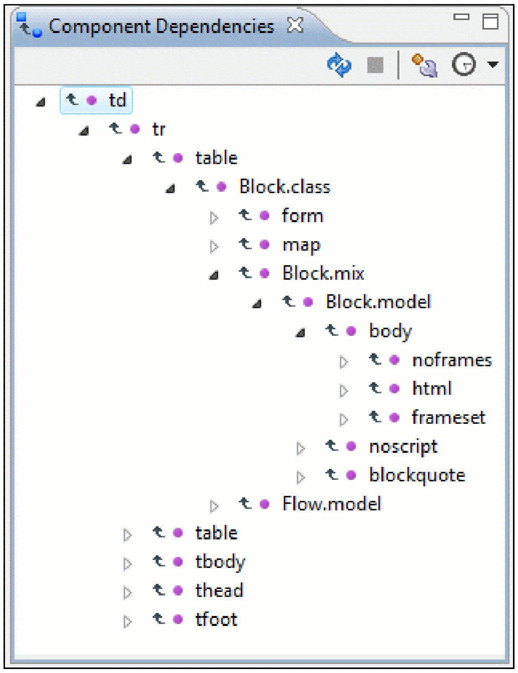
When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon

Component Dependencies View

The Component Dependencies view allows you to easily see the dependencies for a selected RelaxNG component. You can open the view from Window → Show View → Other → oXygen → Component Dependencies.

If you want to see the dependencies of a RelaxNG component just select the desired component in the editor and choose the **Component Dependencies** action from the contextual menu. The action is available for all named defines.

Figure 4.97. Component Dependencies view - hierarchy for xhtml.rng



In the Component Dependencies view you have several actions in the toolbar:

Refresh the dependencies structure.

Allows you to stop the dependencies computing.

Allows you to configure a search scope for compute the dependencies structure in the following dialog:

You can decide to automatically use the defined scope for future operations by checking the corresponding checkbox.

Allows you to repeat a previous dependencies computation.

On the contextual menu you have also some actions like:

- **Go to First Reference** selects the first reference of the referred component from the current selected component in the dependencies tree.
- **Go to Component** Shows the definition of the current selected component in the dependencies tree.

Tip

If a component contains multiple references to another a small table is shown containing all references.

When a recursive reference is encountered it is marked with a special icon

Configuring a custom datatype library for a RELAX NG Schema

A RELAX NG schema can declare a custom datatype library for the values of elements in the instance XML documents. The datatype library must be implemented in Java and must implement the interface specified on the www.thaiopensource.com website. [http://www.thaiopensource.com/relaxng/pluggable-datatypes.html]

The jar file containing the custom library and any other dependent jar file must be added to the classpath of the application, that is the jar files must be added to the folder `[Oxygen-plugin-folder]/lib` and a line *<library name="lib/custom-library.jar"/>* must be added for each jar file to the file `[Oxygen-plugin-folder]/plugin.xml` in the <runtime> element.

The Eclipse platform must be restarted for loading the custom library.

Linking between development and authoring

The Author page is available on the Relax NG schema presenting the schema very similar with the Relax NG compact syntax. It links to imported schemas and external references. Embedded Schematron is also supported. See more details here.

Editing NVDL schemas

When a complex XML document is composed by combining elements and attributes from different namespaces and the schemas which define these namespaces are not even developed in the same schema language then it is difficult to specify in the document all the schemas which must be taken into account for validation of the XML document or for offering content completion when the document is edited. In this case a NVDL (Namespace Validation Definition Language) schema can be used which allows to combine and interleave multiple schemas of different types (W3C XML Schema, RELAX NG schema, Schematron schema) in the same XML document.

<oXygen/> provides a special type of editor for NVDL schemas. This editor presents the usual text view of an XML document synchronized in real time with an outline view. The outline view has two display modes: the standard outline mode and the components mode.

NVDL schema diagram

Introduction

<oXygen/> provides a simple, expressive and easy to read Schema Diagram View for NVDL schemas.

With this new feature you can easily develop complex schemas, print them on multiple pages or save them as JPEG, PNG and BMP images. It helps both schema authors in developing the schema and content authors that are using the schema to understand it.

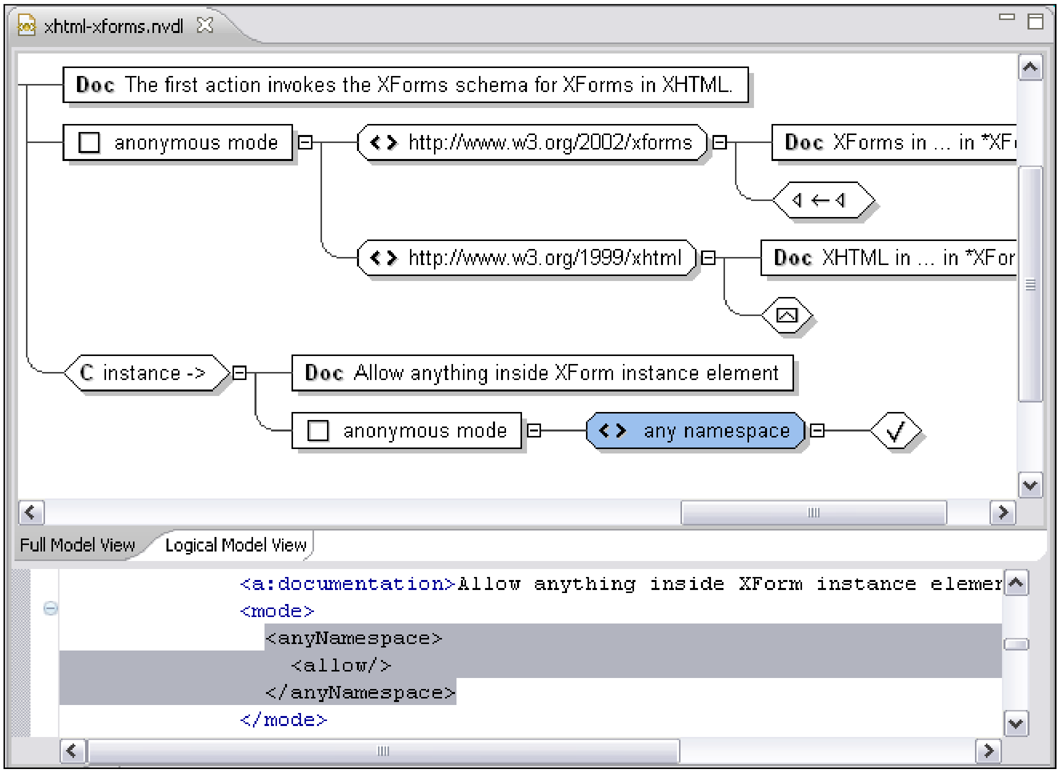
<oXygen/> is the only XML Editor to provide a side by side source and diagram presentation and have them synchronized in real-time:

- the changes you make in the Editor will immediately be visible in the Diagram (no background parsing).
- changing the selected element in the diagram will select the underlaying code in the source editor.

Full model view

When you create a new schema document or open an existing one the Editor Panel is divided in two sections: one containing the Schema Diagram and the second the source code. The Diagram View has two tabbed panes offering a Full Model View and a Logical Model View. The Logical Model View is not available for NVDL.

Figure 4.98. NVDL schema editor - full model view


xhtml-xforms.nvdl
Doc The first action invokes the XForms schema for XForms in XHTML.
anonymous mode
http://www.w3.org/2002/xforms
Doc XForms in ... in *XF
http://www.w3.org/1999/xhtml
Doc XHTML in ... in *XFor
C instance ->
Doc Allow anything inside XForm instance element
anonymous mode
any namespace
Full Model View
Logical Model View

```
            <a:documentation>Allow anything inside XForm instance elemen
            <mode>
              <anyNamespace>
                <allow/>
              </anyNamespace>
            </mode>
```

The Full Model View renders all the NVDL elements with intuitive icons. This coupled with the synchronization support makes the schema navigation easy.

Double click on any diagram component in order to edit its properties.

Actions available in the diagram view

The contextual menu offers some actions:

- **Show only the selected component** Depending on its state(selected/not selected), the selected component is the single component shown in the diagram or all the diagram components are shown.
- **Show Annotations** Depending on its state (selected/not selected), the documentation nodes are shown or hidden.
- **Auto expand to references** This option controls how the schema diagram is automatically expanded. For instance if you select it and then edit a top level element or you make a refresh, the diagram will be expanded until it reaches referred components. If this is left unchecked, only the first level of the diagram is expanded, showing the top level elements.

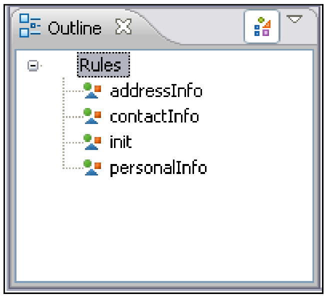
  For large schemas, the editor disables this option automatically.
- **Collapse Children** Collapse the children of the selected view
- **Expand Children** Expand the children of the selected view.
- **Print Selection...** Print the selected view.
- **Save Selection as Image...** Save the current selection as JPEG Image.
- **Refresh** Refreshes the Schema Diagram according to the changes in your code (changes in your imported documents or those that are not reflected automatically in the compiled schema).

If the schema is not valid you will see an error message in the Logical Model View instead of the diagram.

NVDL Outline view

The NVDL Outline View presents a list with the named or anonymous rules that appear in the diagram. It allows a quick access to a rule by knowing its name. It can be opened from Window → Show View → Other → oXygen → Outline

Figure 4.99. Outline view for NVDL



NVDL editor specific actions

The list of actions specific for the NVDL editor of <oXygen/> is:

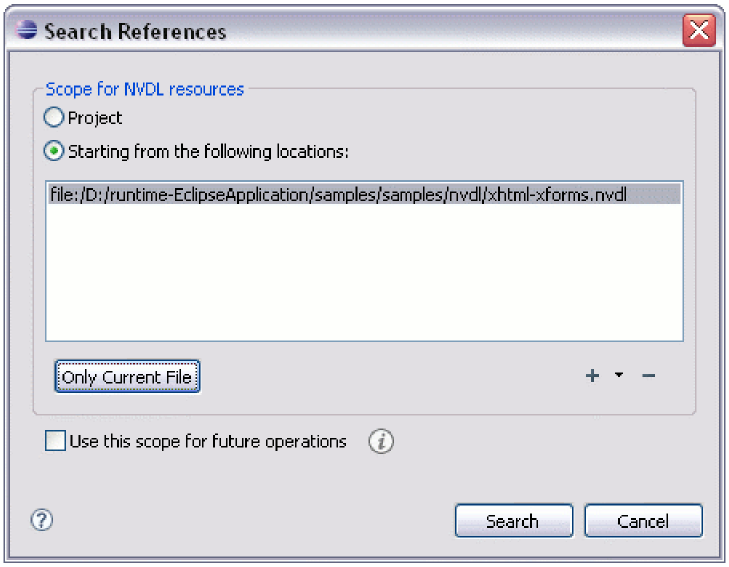
- contextual menu of current editor → Show Definition : move the cursor to its definition in the schema used by NVDL to validate it.

Searching and refactoring actions

All the following actions can be applied on mode *name*, *useMode* and *startMode* attributes only.

- NVDL+ → References (**Alt+Shift+S R (Cmd+Alt+S R on Mac OS)**): Searches all references of the item found at current cursor position in the defined scope if any. If a scope is defined but the current edited resource is not part of the range of resources determined by this a warning dialog will be shown and you have the possibility to define another scope, otherwise you have to define a search scope. You can define the search references scope in the following dialog:

  Figure 4.100. Search References dialog

  Search References

  Scope for NVDL resources

  ○ Project

  ● Starting from the following locations:

  file:/D:/runtime-EclipseApplication/samples/samples/nvdl/xhtml-xforms.nvdl

  Only Current File

  ☐ Use this scope for future operations

  A search scope may include the project or a collection of files and directories that you have to specify.

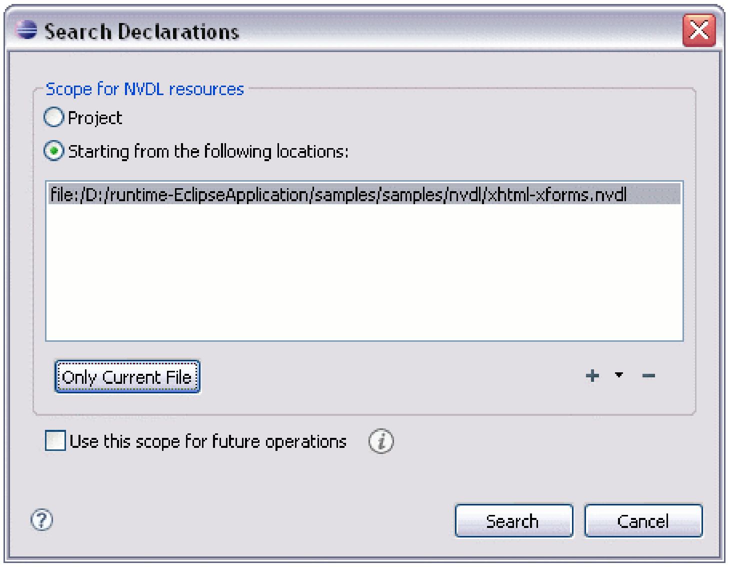
  You can decide to automatically use the defined scope for future operations by checking the corresponding checkbox.

- contextual menu of current editor+Search → References in...: Searches all references of the item found at current cursor position in the file(s) that you specify when define a scope in the dialog above.

All the following actions can be applied on named *define* parameters only.

- NVDL+ → Declarations (**Alt+Shift+S D (Cmd+Alt+S D on Mac OS)**): Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined but the current edited resource is not part of the range of resources determined by this a warning dialog will be shown and you have the possibility to define another scope, otherwise you have to define a search scope. You can define the search declarations scope in the following dialog:

Figure 4.101. Search Declarations dialog



Search Declarations

Scope for NVDL resources

○ Project

● Starting from the following locations:

file:/D:/runtime-EclipseApplication/samples/samples/nvdl/xhtml-xforms.nvdl

□ Use this scope for future operations

A search scope may include the project or a collection of files and directories that you have to specify.

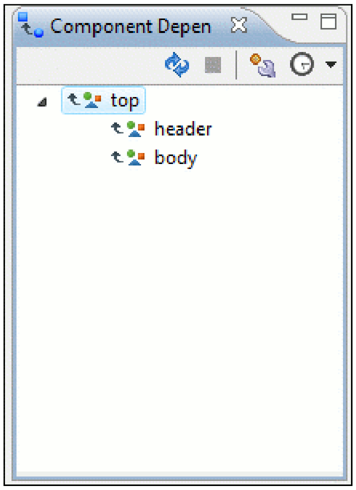
- contextual menu of current editor+Search → Declarations in...: Searches all declarations of the item found at current cursor position in the file(s) that you specify when define a scope in the dialog above.
- NVDL → Occurrences in File (**Ctrl+Shift+U**): Searches all occurrences of the item at the caret position in the currently edited file.
- contextual menu of current editor+Refactoring+Rename Component...: Rename the selected component. You have to specify the new name for the component and the file(s) affected by the modification as described for XML Schema

Component Dependencies View

The Component Dependencies view allows you to easily see the dependencies for a selected NVDL named mode. You can open the view from Window → Show View → Other → oXygen → Component Dependencies.

If you want to see the dependencies of a NVDL mode just select the desired component in the editor and choose the **Component Dependencies** action from the contextual menu. The action is available for all named modes.

Figure 4.102. Component Dependencies view - hierarchy for test.nvdl



In the Component Dependencies view you have several actions in the toolbar:

Refresh the dependencies structure.

🟥 Allows you to stop the dependencies computing.

Allows you to configure a search scope for compute the dependencies structure in the following dialog:

You can decide to automatically use the defined scope for future operations by checking the corresponding checkbox.

Allows you to repeat a previous dependencies computation.

On the contextual menu you have also some actions like:

- **Go to First Reference** selects the first reference of the referred component from the current selected component in the dependencies tree.
- **Go to Component** Shows the definition of the current selected component in the dependencies tree.

Tip

If a component contains multiple references to another a small table is shown containing all references.

When a recursive reference is encountered it is marked with a special icon

Linking between development and authoring

The Author page is available on the NVDL scripts editor presenting them in a compact and easy to understand representation. See more details here.

Editing XSLT stylesheets

<oXygen/> provides special support for developing XSLT 1.0 / 2.0 stylesheets.

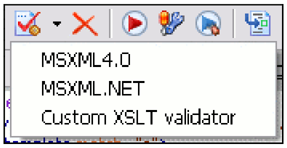
Validating XSLT stylesheets

Validation of XSLT stylesheets documents is performed with the help of an XSLT processor configurable from user preferences according to the XSLT version: 1.0 or 2.0. For XSLT 1.0 the options are: Xalan, Saxon 6.5.5, Saxon 9 B, Saxon 9 SA, MSXML 4.0, MSXML.NET, a JAXP transformer specified by the main Java class. For XSLT 2.0 the options are: Saxon 9 B, Saxon 9 SA, a JAXP transformer specified by the main Java class.

Custom validation of XSLT stylesheets

If you need to validate an XSLT stylesheet with other validation engine than the built-in ones you have the possibility to configure external engines as custom XSLT validation engines in <oXygen/>. After such a custom validator is properly configured in Preferences it can be applied on the current document with just one click on the External Validation toolbar. The document is validated against the schema declared in the document.

Figure 4.103. External validation toolbar



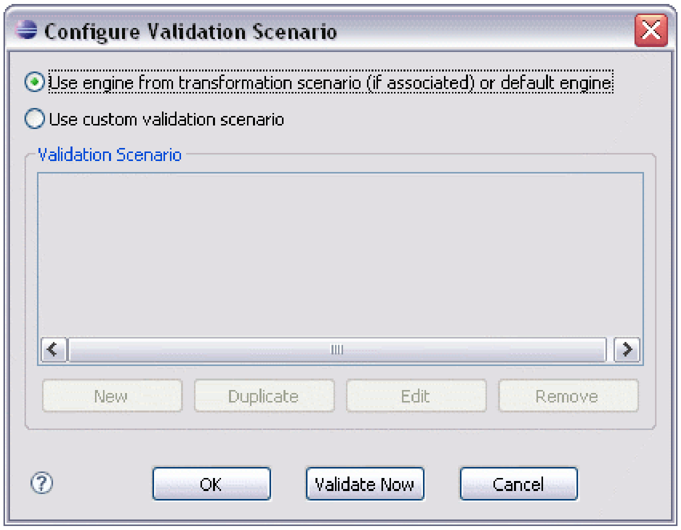
There are two validators configured by default:

MSXML 4.0 included in <oXygen/> (Windows edition). It is associated to the XSL Editor type in Preferences.

MSXML.NET included in <oXygen/> (Windows edition). It is associated to the XSL Editor type in Preferences.

Associate a validation scenario

Figure 4.104. Configure Transformation Dialog


Configure Validation Scenario
Use engine from transformation scenario (if associated) or default engine
Use custom validation scenario
Validation Scenario
New
Duplicate
Edit
Remove
OK
Validate Now
Cancel

Content Completion in XSLT stylesheets

The content completion assistant adds special features for editing XSLT stylesheets.

Inside XSLT templates of an XSLT stylesheet the content completion presents also all the elements allowed in any context by the schema associated to the result of applying the edited stylesheet. That schema is defined by the user in the Content Completion / XSL preferences. There are presented all the elements because in a template there is no context defined for the result document so the user is allowed to insert any element defined by the schema of the result document.

The content completion window lists the template modes and the names of templates, variables and parameters defined in imported and included XSLT stylesheets together with the ones defined in the current stylesheet.

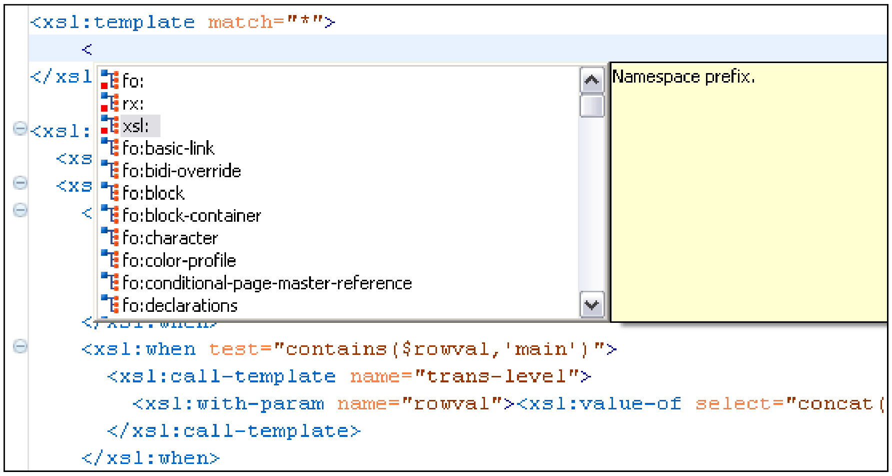
The extension functions built in to the Saxon product are presented in the content completion list if the Saxon namespace (http://saxon.sf.net [] for version 2.0 or http://icl.com/saxon for version 1.0) are mapped to a prefix and one of the following conditions are true:

- if the edited file has a transformation scenario that use as transformation engine Saxon 6.5.5 (for version 1.0), Saxon 9.1.0.7 B or Saxon 9.1.0.7 SA (for version 2.0)
- if the edited file has a validation scenario that use as validation engine Saxon 6.5.5 (for version 1.0), Saxon 9.1.0.7 B or Saxon 9.1.0.7 SA (for version 2.0)
- if the validation engine specified in Options is Saxon 6.5.5 (for version 1.0), Saxon 9.1.0.7 B or Saxon 9.1.0.7 SA (for version 2.0)

Namespace prefixes in scope for the current context are presented at the top of the content completion window to speed the insertion of prefixed elements into the document.

For the common namespaces like XSL namespace (http://www.w3.org/1999/XSL/Transform), XML Schema namespace (http://www.w3.org/2001/XMLSchema) or Saxon namespace (http://icl.com/saxon for version 1.0, http://saxon.sf.net/ for version 2.0) , <oXygen/> provides an easily mode to mapped them by propose a prefix for these namespaces.

Figure 4.105. Namespace prefixes in the content completion window



```
<xsl:template match="*">
    <
</xsl
<xsl:
  <xs
  <xs
    <
    </xsl:when>
    <xsl:when test="contains($rowval,'main')">
      <xsl:call-template name="trans-level">
        <xsl:with-param name="rowval"><xsl:value-of select="concat(
      </xsl:call-template>
    </xsl:when>
```

Content Completion in XPath expressions

In XSLT stylesheets the content completion assistant provides all the features available in the editor for XML documents and also adds some enhancements. In XPath expressions used in attributes of XSLT stylesheets elements like *match*, *select* and *test* it offers XPath functions, XSLT functions, XSLT axes and user defined functions. If a transformation scenario was defined and associated to the edited stylesheet the content completion assistant computes and presents elements and attributes based on the input XML document selected in the scenario and on the current context in the stylesheet. The associated document is displayed in the XSLT/XQuery input view.

Content Completion for XPath expressions is started:

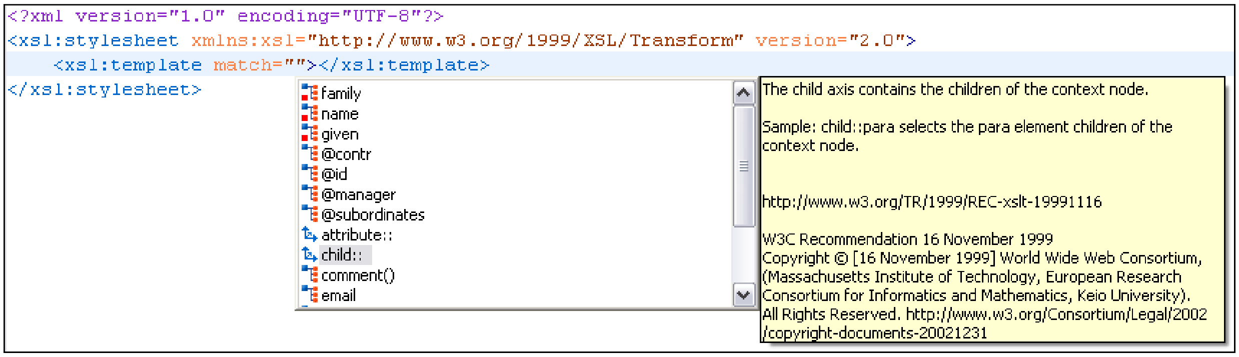
- on XPath operators detected in one of the *match*, *select* and *test* attributes of XSLT elements: ", ', /, //, (, [, |, :, ::, $
- for attribute value templates of non XSLT elements, that is the '{' character is detected as the first character of the attribute value
- on request if the combination CTRL + Space is pressed inside an edited XPath expression

The items presented in the content completion window are dependent on the context of the current XSLT element, the XML document associated with the edited stylesheet in the transformation scenario of the stylesheet and the XSLT version of the stylesheet (1.0 or 2.0). For example if the document associated with the edited stylesheet is:

```
<personnel>
    <person id="Big.Boss">
        <name>
            <family>Boss</family>
            <given>Big</given>
        </name>
        <email>chief@oxygenxml.com</email>
        <link subordinates="one.worker"/>
    </person>
    <person id="one.worker">
        <name>
            <family>Worker</family>
            <given>One</given>
        </name>
        <email>one@oxygenxml.com</email>
        <link manager="Big.Boss"/>
    </person>
</personnel>
```

and you enter an element *xsl:template* using the content completion assistant the *match* attribute is inserted automatically, the cursor is placed between the quotes and the XPath content completion assistant automatically displays a popup window with all the XSLT axes, XPath functions and elements and attributes from the XML input document that can be inserted in the current context. The set of XPath functions depends on the XSLT version declared in the root element - *xsl:stylesheet* (1.0 or 2.0).

Figure 4.106. Content Completion in the *match* attribute



```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
    <xsl:template match=""></xsl:template>
</xsl:stylesheet>
```

The child axis contains the children of the context node.

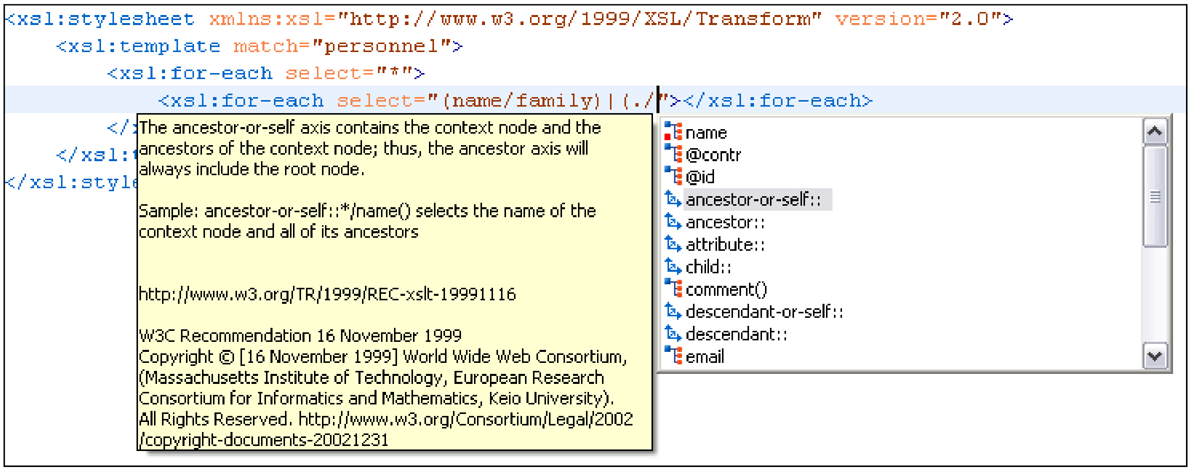
Sample: child::para selects the para element children of the context node.

http://www.w3.org/TR/1999/REC-xslt-19991116

W3C Recommendation 16 November 1999
Copyright © [16 November 1999] World Wide Web Consortium, (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231

If the cursor is inside the *select* attribute of an *xsl:for-each*, *xsl:apply-templates*, *xsl:value-of* or *xsl:copy-of* element the content completion proposals are dependent of the path obtained by concatenating the XPath expressions of the parent XSLT elements *xsl:template* and *xsl:for-each* like the following figure shows:

Figure 4.107. Content Completion in the *select* attribute



```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
    <xsl:template match="personnel">
        <xsl:for-each select="*">
            <xsl:for-each select="(name/family)|(./"></xsl:for-each>
        </
    </xsl:t
</xsl:styl
```

The ancestor-or-self axis contains the context node and the ancestors of the context node; thus, the ancestor axis will always include the root node.

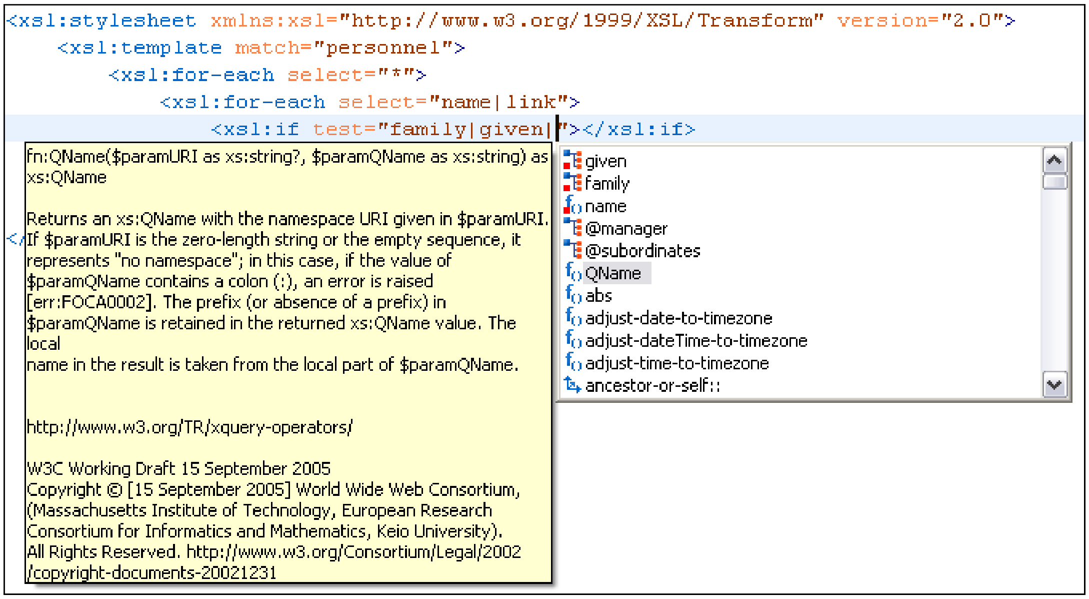
Sample: ancestor-or-self::*/name() selects the name of the context node and all of its ancestors

http://www.w3.org/TR/1999/REC-xslt-19991116

W3C Recommendation 16 November 1999
Copyright © [16 November 1999] World Wide Web Consortium, (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231

Also XPath expressions typed in the *test* attribute of an *xsl:if* or *xsl:choose / xsl:when* element benefit of the assistance of the content completion.

Figure 4.108. Content Completion in the *test* attribute



```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
    <xsl:template match="personnel">
        <xsl:for-each select="*">
            <xsl:for-each select="name|link">
                <xsl:if test="family|given|"></xsl:if>
</
```

fn:QName($paramURI as xs:string?, $paramQName as xs:string) as xs:QName

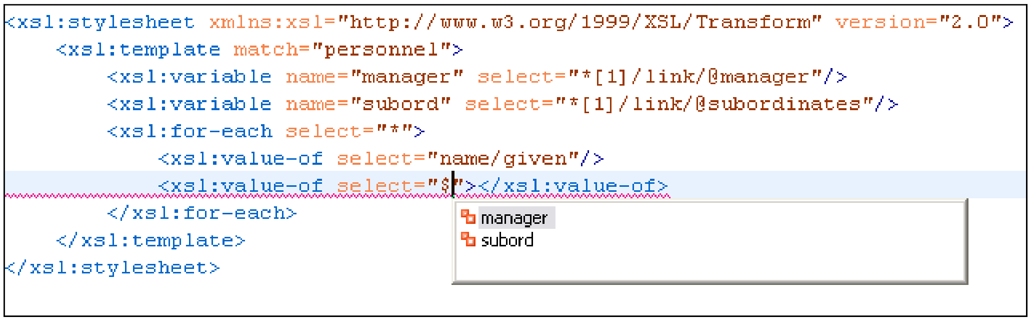
Returns an xs:QName with the namespace URI given in $paramURI. If $paramURI is the zero-length string or the empty sequence, it represents "no namespace"; in this case, if the value of $paramQName contains a colon (:), an error is raised [err:FOCA0002]. The prefix (or absence of a prefix) in $paramQName is retained in the returned xs:QName value. The local
name in the result is taken from the local part of $paramQName.

http://www.w3.org/TR/xquery-operators/

W3C Working Draft 15 September 2005
Copyright © [15 September 2005] World Wide Web Consortium, (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231

XSLT variable references are easier to insert in XPath expressions with the help of the content completion popup triggered by the $ character which signals the start of such a reference in an XPath expression.

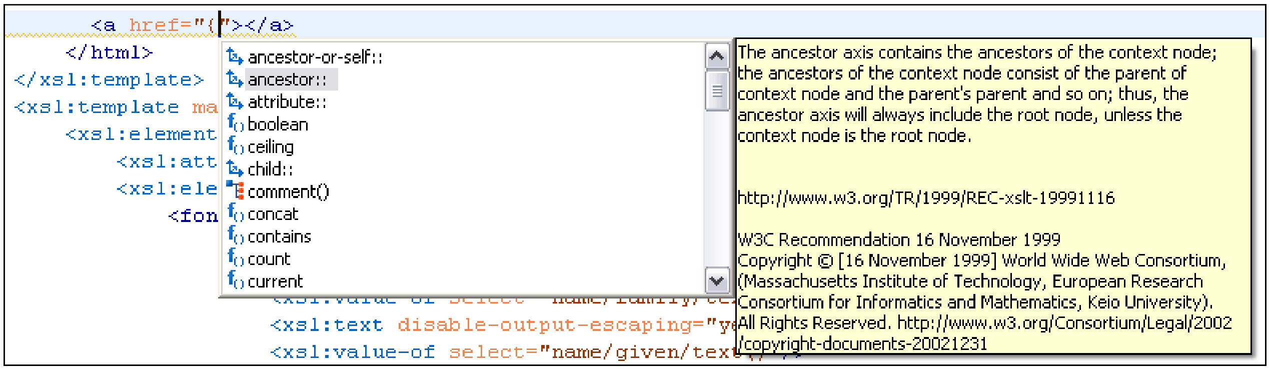
Figure 4.109. Content Completion in the *test* attribute



```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
    <xsl:template match="personnel">
        <xsl:variable name="manager" select="*[1]/link/@manager"/>
        <xsl:variable name="subord" select="*[1]/link/@subordinates"/>
        <xsl:for-each select="*">
            <xsl:value-of select="name/given"/>
            <xsl:value-of select="$"></xsl:value-of>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

The same content completion assistant is available also in attribute value templates of non XSLT elements if the '{' character is the first one in the value of the attribute.

Figure 4.110. Content Completion in attribute value templates



```
      <a href="{"></a>
    </html>
</xsl:template>
<xsl:template ma
    <xsl:element
        <xsl:att
        <xsl:ele
            <fon
                <xsl:text disable-output-escaping="y
                <xsl:value-of select="name/given/text
```

The ancestor axis contains the ancestors of the context node; the ancestors of the context node consist of the parent of context node and the parent's parent and so on; thus, the ancestor axis will always include the root node, unless the context node is the root node.

http://www.w3.org/TR/1999/REC-xslt-19991116

W3C Recommendation 16 November 1999
Copyright © [16 November 1999] World Wide Web Consortium, (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All Rights Reserved. http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231

Tooltip Helper for the XPath Functions Arguments

When editing the arguments of an XPath/XSLT function, <oXygen/> keeps track of the current entered argument by displaying a tooltip above the function containing the function signature. The currently edited argument is displayed in bold.

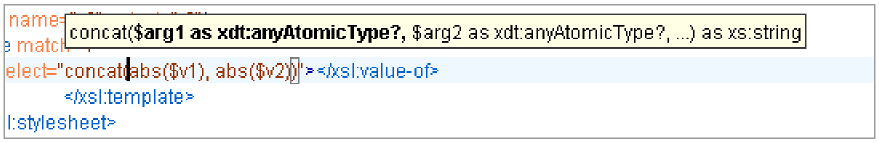
When moving the caret through the expression, the tooltip is updated to reflect the argument that is found at the caret position.

Let's consider the following example. We are concatenating the absolute value of two variables: v1 and v2.

```
<xsl:template match="/">
    <xsl:value-of select="concat(abs($v1), abs($v2))"></xsl:value-of>
</xsl:template>
```

When moving the caret before the first "abs" function, the editor will identify that it represent the first argument of the "concat" function, and will show in bold that the first argument is named "$arg1" and is of type "xdt:anyAtomicType" and it is optional. The function takes also other arguments, having the same type, and returns a "xs:string".

Figure 4.111. XPath Tooltip Helper - Identify the concat function first argument

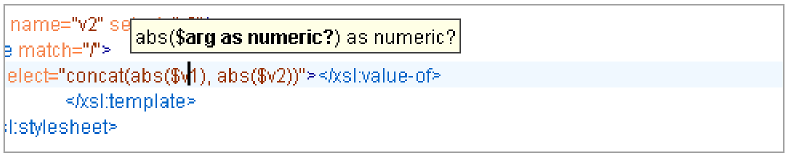


```
name=
e match
elect="concat(abs($v1), abs($v2))"></xsl:value-of>
        </xsl:template>
l:stylesheet>
```

concat(**$arg1 as xdt:anyAtomicType?,** $arg2 as xdt:anyAtomicType?, ...) as xs:string

Moving the caret on the first variable "$v1", the editor identifies the "abs" as context function and shows its signature:

Figure 4.112. XPath Tooltip Helper - Identify the abs function argument

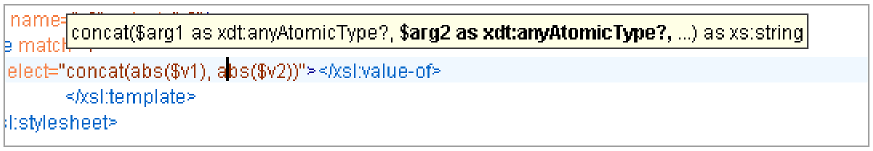


```
name="v2" se
e match="/">
elect="concat(abs($v1), abs($v2))"></xsl:value-of>
        </xsl:template>
:l:stylesheet>
```

abs(**$arg as numeric?**) as numeric?

Further, clicking on the second "abs" function name, the editor detects that it represents the second argument of the "concat function". It redisplays the correct tooltip, displaying the second argument in bold.

Figure 4.113. XPath Tooltip Helper - Identify the concat function second argument



```
name=
e match
elect="concat(abs($v1), abs($v2))"></xsl:value-of>
        </xsl:template>
:l:stylesheet>
```

concat($arg1 as xdt:anyAtomicType?, **$arg2 as xdt:anyAtomicType?,** ...) as xs:string

The tooltip helper is present also in the XPath Toolbar and the XPath Builder.

Code templates

When the content completion is invoked by pressing **CTRL+Space** it also presents a list of code templates specific to the type of the active editor. Such a code template provides a shortcut for inserting a small document fragment at the current caret position. <oXygen/> comes with a large set of ready-to use templates for XSL and XML Schema documents.

Example 4.16. The XSL code template called Template-Match-Mode

Typing **t** in an XSL document and selecting **tmm** in the content assistant pop-up window will insert the following template at the caret position in the document:

```
<xsl:template match="" mode="">

</xsl:template>
```

Other templates can be easily defined by the user. Also the code templates can be shared with other users.

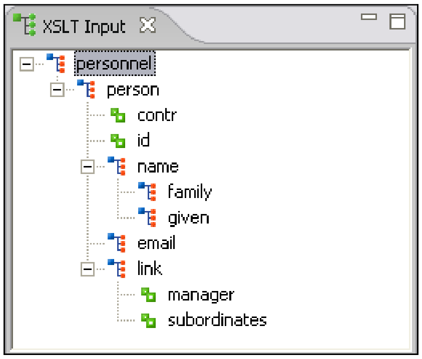
The XSLT/XQuery Input View

The structure of the XML document associated to the edited XSLT stylesheet , or of the source documents of the edited XQuery is displayed in a tree form in a view called *XSLT/XQuery Input*. The tree nodes represent the elements of the documents.

The XSLT Input View

If you click on a node, the corresponding template from the stylesheet will be highlighted. A node can be dragged and dropped in the editor area for quickly inserting *xsl:template*, *xsl:for-each* or other XSLT elements with the *match / select / test* attribute already filled with the correct XPath expression referring to the dragged tree node and based on the current editing context of the drop spot.

Figure 4.114. XSLT input view



For example for the following XML document

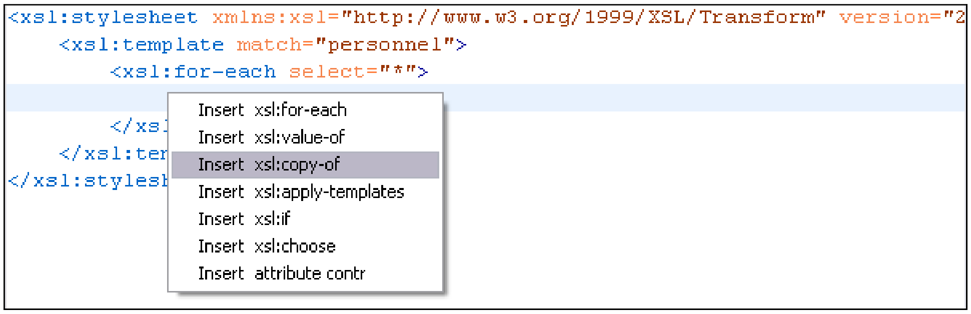
```
<personnel>
    <person id="Big.Boss">
        <name>
            <family>Boss</family>
            <given>Big</given>
        </name>
        <email>chief@oxygenxml.com</email>
        <link subordinates="one.worker"/>
    </person>
    <person id="one.worker">
        <name>
            <family>Worker</family>
            <given>One</given>
        </name>
        <email>one@oxygenxml.com</email>
        <link manager="Big.Boss"/>
    </person>
</personnel>
```

and the following XSLT stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        version="2.0">
    <xsl:template match="personnel">
        <xsl:for-each select="*">

        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

if you drag the *given* element and drop it inside the *xsl:for-each* element a popup menu will be displayed.

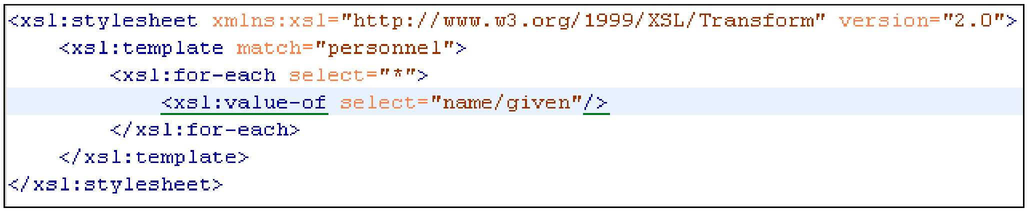
Figure 4.115. XSLT Input drag and drop popup menu



```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2
    <xsl:template match="personnel">
        <xsl:for-each select="*">

        </xsl
    </xsl:te
</xsl:styles
```

Select for example *Insert xsl:value-of* and the result document will be:

Figure 4.116. XSLT Input drag and drop result

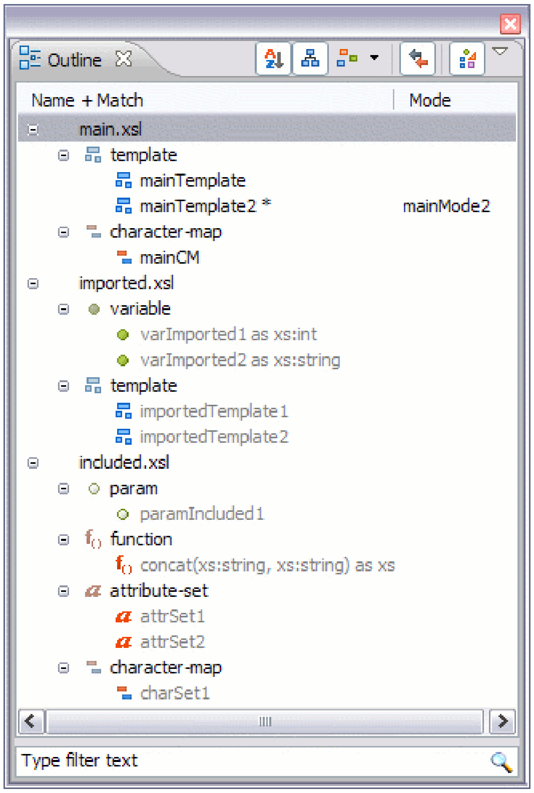


```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
    <xsl:template match="personnel">
        <xsl:for-each select="*">
            <xsl:value-of select="name/given"/>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

The XSLT Outline View

The XSLT Outline View present the list of all the components (templates, attribute-sets, character-maps, variables, functions) from both the edited stylesheet and its imports/includes. It can be opened from Window → Show View → Other → oXygen → Outline.

Figure 4.117. The XSLT Outline View



Name + Match	Mode
main.xsl	
template	
mainTemplate	
mainTemplate2 *	mainMode2
character-map	
mainCM	
imported.xsl	
variable	
varImported1 as xs:int	
varImported2 as xs:string	
template	
importedTemplate1	
importedTemplate2	
included.xsl	
param	
paramIncluded1	
function	
concat(xs:string, xs:string) as xs	
attribute-set	
attrSet1	
attrSet2	
character-map	
charSet1	

Type filter text

The XSLT Outline View provide some actions to easily navigate inside a stylesheet:

Sort Allows you to alphabetically sort the stylesheet components.

Show imported/included Allows you to show also the components from imported/included stylesheets.

Grouping options The stylesheet components can be grouped by location, type and mode.

Selection update on caret move Allows a synchronization between Outline View and source document. The selection in the outline view can be synchronized with the caret's moves or the changes in the XSLT editor. Selecting one of the components from the outline view also selects the corresponding item in the source document.

Components mode Allows you to switch between the current outline and the standard Outline View. Your preference for specific or standard outline will be applied to all new xslt editors opened after this operation.

The following contextual menu actions are available:

Remove (**Delete**)
Remove the selected item from the stylesheet.

Search References ()
Searches all references of the item found at current cursor position in the defined scope if any. Click here for more details.

Search References in...
Searches all references of the item found at current cursor position in the specified scope. Click here for more details.

Component Dependencies
Allows you to easily see the dependencies for the current selected component. Click here for more details.

Rename Component	Rename the selected component. Click here for more details.

The stylesheet components information are presented in two columns: the first column present the *name* and *match* attributes, the second column the *mode* attribute. If you know the component name, match or mode, you can search it in the outline view by typing one of these information in the filter text field from the bottom of the view or directly on the tree structure. When you type de component name, match or mode in the filter text field you can switch to the tree structure using the arrow keys of the keyboard, Enter, Tab, Shift-Tab. To switch from tree structure to the filter text field you can use Tab, .

Tip

The search filter is case insensitive. The following wildcards are accepted:

- ***** - any string
- **?** - any character
- **,** -patterns separator

If no wildcards are specified, the string to search will be searched as a partial match (similar to ***textToFind***).

On the XSLT outline view you have some contextual actions like: Edit Attributes, Cut, Copy, Delete.

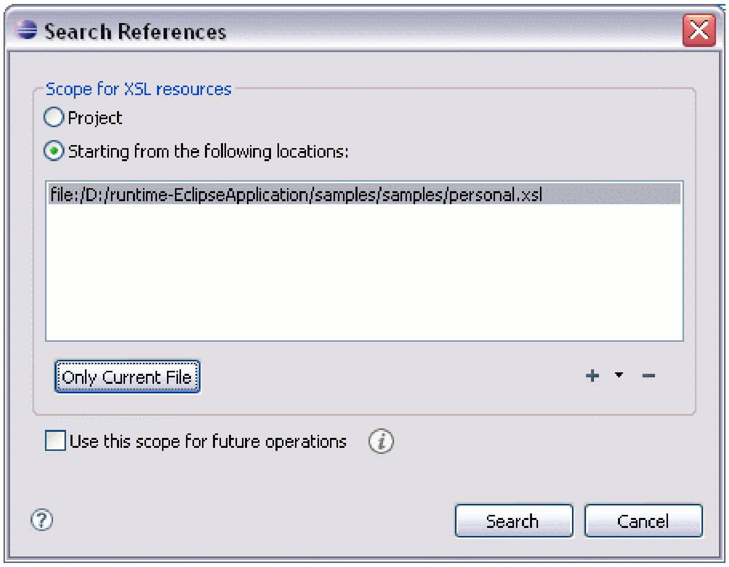
Finding XSLT references and declarations

Note

All the following actions can be applied on named templates, attribute sets, functions, decimal formats, keys, variables or parameters only. In case they are applied on other items, a warning message will pop-up.

- XSL+ → References (**Alt+Shift+S R (Cmd+Alt+S R on Mac OS)**): Searches all references of the item found at current cursor position in the defined scope if any. If a scope is defined but the current edited resource is not part of the range of resources determined by this a warning dialog will be shown and you have the possibility to define another scope, otherwise you have to define a search scope. You can define the search references scope in the following dialog:

Figure 4.118. Search References dialog



Scope for XSL resources

○ Project

● Starting from the following locations:

file:/D:/runtime-EclipseApplication/samples/samples/personal.xsl

□ Use this scope for future operations

A search scope may include the project or a collection of files and directories that you have to specify.

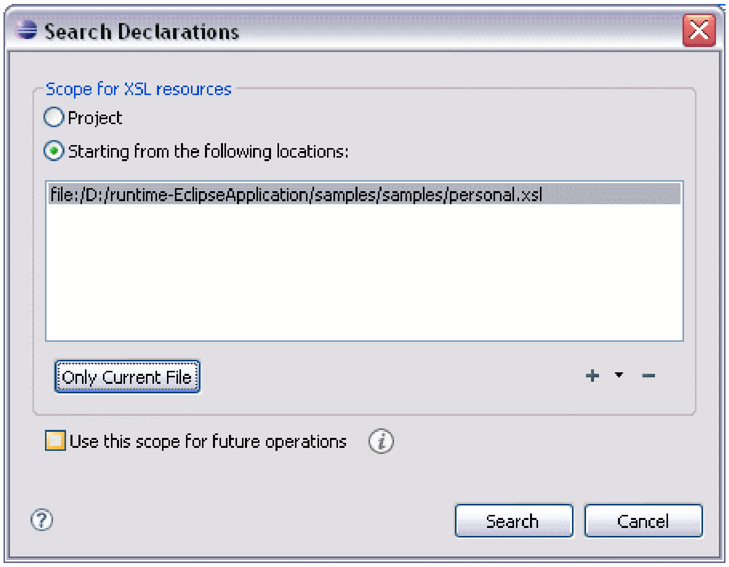
Note

For faster access, a shortcut to this action is also added in the XSL References toolbar.

You can decide to automatically use the defined scope for future operations by checking the corresponding checkbox.

- contextual menu of current editor+Search → References in...: Searches all references of the item found at current cursor position in the file(s) that you specify when define a scope in the dialog above.
- XSL+ → Declarations (**Alt+Shift+S D (Cmd+Alt+S D on Mac OS)**): Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined but the current edited resource is not part of the range of resources determined by this a warning dialog will be shown and you have the possibility to define another scope, otherwise you have to define a search scope. You can define the search declarations scope in the following dialog:

Figure 4.119. Search Declarations dialog



Scope for XSL resources

○ Project

● Starting from the following locations:

file:/D:/runtime-EclipseApplication/samples/samples/personal.xsl

☐ Use this scope for future operations

A search scope may include the project or a collection of files and directories that you have to specify.

Note

For faster access, a shortcut to this action is also added in the XSL References toolbar.

- contextual menu of current editor+Search → Declarations in...: Searches all declarations of the item found at current cursor position in the file(s) that you specify when define a scope in the dialog above.
- XSL → Occurrences in file (**Alt+Shift+S O (Cmd+Alt+S + O on Mac OS)**): Searches all occurrences of the item at the caret position in the currently edited file.
- XSL → Show Definition (**Ctrl+Alt+Enter (Cmd+Shift+Enter on Mac OS)**): Moves the cursor to the location of the definition of the current item.

XSLT refactoring actions

- XSL+ → Create template from selection...: Opens a dialog that allows the user to specify the name of the new template to be created. The possible changes to be performed on the document can be previewed prior to altering the document. After pressing OK, the template is created and the selection is replaced by a

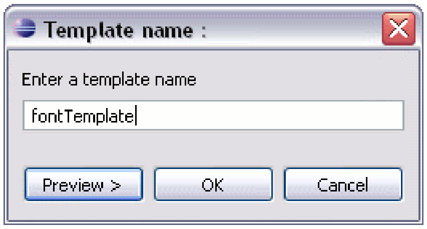
  ```
  xsl:call-template
  ```

  instruction referring the just created template.

  ### Note

  The selection must contain wellformed elements only.

Figure 4.120. Create template from selection dialog



Template name :

Enter a template name

fontTemplate

- XSL+ → Create stylesheet from selection...: Creates a separate stylesheet and replaces the selection with a

  ```
  xsl:include
  ```

  instruction referring the just created stylesheet.

  ### Note

  The selection must contain a well formed top level element.

- XSL → Extract attributes as xsl:attributes...: Extracts the attributes from the selected element and represents each of them with a

  ```
  xsl:attribute
  ```

  instruction.

  For example from the following element

  ```
  <person id="Big{test}Boss"/>
  ```

  you would obtain

  ```
  <person>
      <xsl:attribute name="id">
          <xsl:text>Big</xsl:text>
          <xsl:value-of select="test"/>
          <xsl:text>Boss</xsl:text>
      </xsl:attribute>
  </person>
  ```

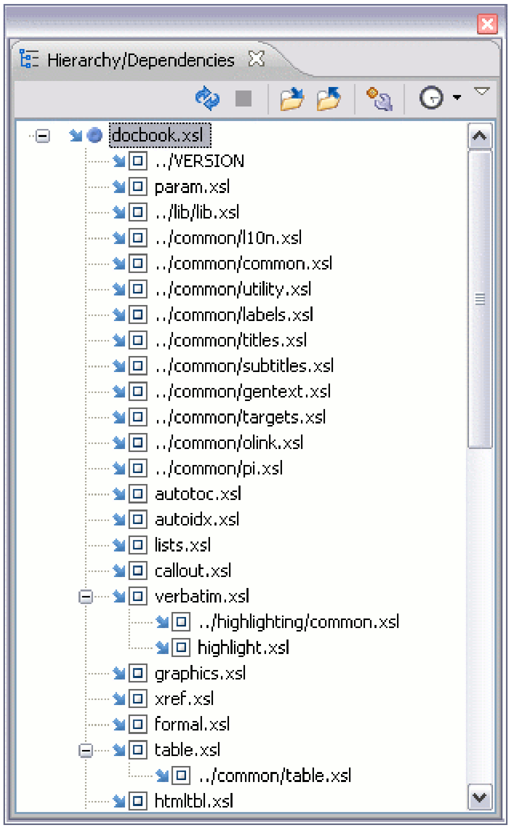
- contextual menu of current editor+Refactoring+Rename Component...: Rename the selected component. You have to specify the new name for the component and the file(s) affected by the modification as described for XML Schema

Resource Hierarchy/Dependencies View

The Resource Hierarchy/Dependencies view allows you to easily see the hierarchy/dependencies for a stylesheet. You can open the view from Window → Show View → Other → oXygen → Resource Hierarchy/Dependencies.

If you want to see the hierarchy of a stylesheet just select the desired stylesheet in the project view and choose **Resource Hierarchy** from the contextual menu.

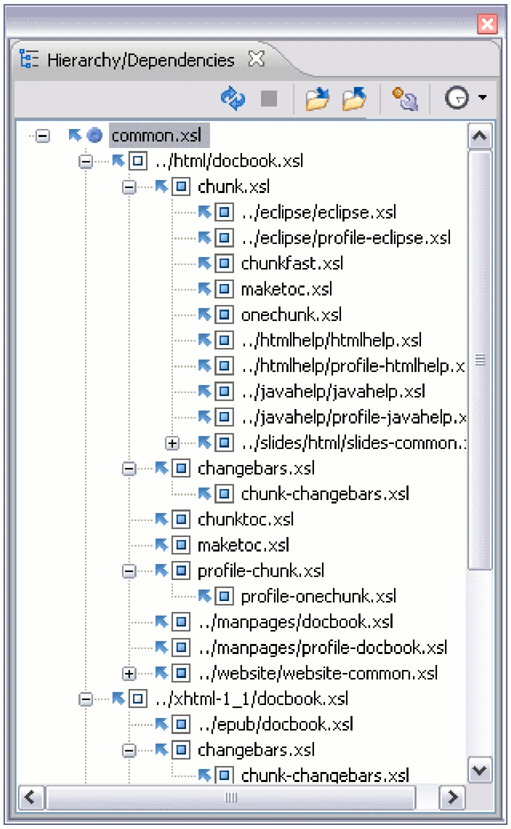
Figure 4.121. Resource Hierarchy/Dependencies view - hierarchy for docbook.xsl



```
docbook.xsl
    ../VERSION
    param.xsl
    ../lib/lib.xsl
    ../common/l10n.xsl
    ../common/common.xsl
    ../common/utility.xsl
    ../common/labels.xsl
    ../common/titles.xsl
    ../common/subtitles.xsl
    ../common/gentext.xsl
    ../common/targets.xsl
    ../common/olink.xsl
    ../common/pi.xsl
    autotoc.xsl
    autoidx.xsl
    lists.xsl
    callout.xsl
    verbatim.xsl
        ../highlighting/common.xsl
        highlight.xsl
    graphics.xsl
    xref.xsl
    formal.xsl
    table.xsl
        ../common/table.xsl
    htmltbl.xsl
```

If you want to see the dependencies of a stylesheet just select the desired stylesheet in the project view and choose **Resource Dependencies** from the contextual menu.

Figure 4.122. Resource Hierarchy/Dependencies view - dependencies for common.xsl



```
common.xsl
    ../html/docbook.xsl
        chunk.xsl
            ../eclipse/eclipse.xsl
            ../eclipse/profile-eclipse.xsl
            chunkfast.xsl
            maketoc.xsl
            onechunk.xsl
            ../htmlhelp/htmlhelp.xsl
            ../htmlhelp/profile-htmlhelp.x
            ../javahelp/javahelp.xsl
            ../javahelp/profile-javahelp.x
            ../slides/html/slides-common.:
        changebars.xsl
            chunk-changebars.xsl
        chunktoc.xsl
        maketoc.xsl
        profile-chunk.xsl
            profile-onechunk.xsl
        ../manpages/docbook.xsl
        ../manpages/profile-docbook.xsl
        ../website/website-common.xsl
    ../xhtml-1_1/docbook.xsl
        ../epub/docbook.xsl
        changebars.xsl
            chunk-changebars.xsl
```

In the Resource Hierarchy/Dependencies view you have several actions in the toolbar:

Refresh the hierarchy/dependencies structure.

Allows you to stop the hierarchy/dependencies computing.

Allows you to choose a schema to compute the hierarchy structure.

Allows you to choose a schema to compute the dependencies structure.

Allows you to configure a scope for compute the dependencies structure.

Allows you to repeat a previous dependencies computation.

On the contextual menu you have also some actions like:

- **Open** Open the schema. Also you can open the schema by a double-click on the hierarchy/dependencies structure.
- **Copy location** Copy the location of the schema.
- **Show Resource Hierarchy** Show the hierarchy for the selected schema.
- **Show Resource Dependencies** Show the dependencies for the selected schema.
- **Expand All** Expand all the children of the selected schema from the hierarchy/dependencies structure.

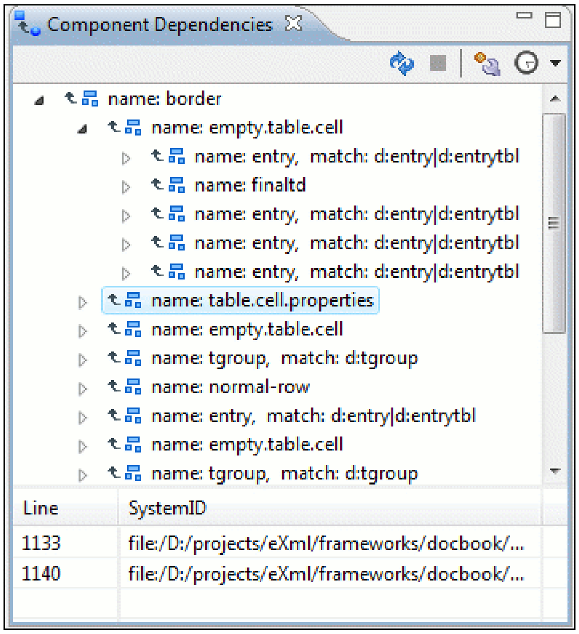
- **Collapse All** Collapse all the children of the selected schema from the hierarchy/dependencies structure.

Component Dependencies View

The Component Dependencies view allows you to easily see the dependencies for a selected XSLT component. You can open the view from Window → Show View → Other → oXygen → Component Dependencies.

If you want to see the dependencies of an XSLT component just select the desired component in the editor and choose the **Component Dependencies** action from the contextual menu. The action is available for all named components (templates, variables, parameters, attribute sets, keys, etc).

Figure 4.123. Component Dependencies view - hierarchy for table.xsl



```
name: border
    name: empty.table.cell
        name: entry,  match: d:entry|d:entrytbl
        name: finaltd
        name: entry,  match: d:entry|d:entrytbl
        name: entry,  match: d:entry|d:entrytbl
        name: entry,  match: d:entry|d:entrytbl
    name: table.cell.properties
    name: empty.table.cell
    name: tgroup,  match: d:tgroup
    name: normal-row
    name: entry,  match: d:entry|d:entrytbl
    name: empty.table.cell
    name: tgroup,  match: d:tgroup
```

Line	SystemID
1133	file:/D:/projects/eXml/frameworks/docbook/...
1140	file:/D:/projects/eXml/frameworks/docbook/...

In the Component Dependencies view you have several actions in the toolbar:

Refresh the dependencies structure.

🟥 Allows you to stop the dependencies computing.

Allows you to configure a search scope for compute the dependencies structure in the following dialog:

You can decide to automatically use the defined scope for future operations by checking the corresponding checkbox.

Allows you to repeat a previous dependencies computation.

On the contextual menu you have also some actions like:

- **Go to First Reference** selects the first reference of the referred component from the current selected component in the dependencies tree.
- **Go to Component** Shows the definition of the current selected component in the dependencies tree.

Tip

If a component contains multiple references to another a small table is shown containing all references.

When a recursive reference is encountered it is marked with a special icon

Linking between development and authoring

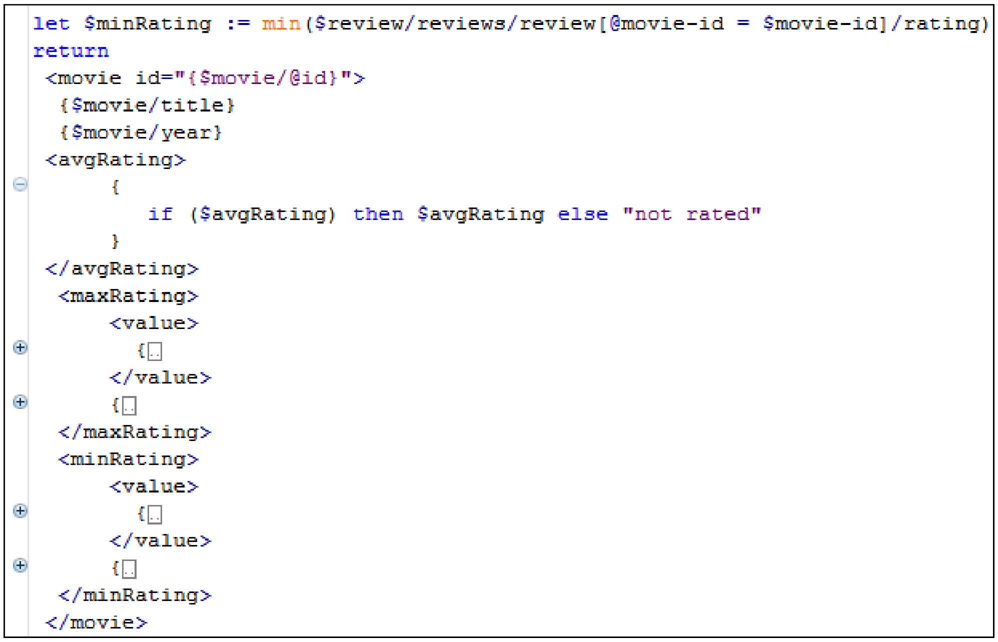
The Author page is available for the XSLT editor presenting the stylesheets in a nice visual rendering. See more details here.

Editing XQuery documents

Folding in XQuery documents

In a large XQuery document the instructions enclosed in the '{' and '}' characters can be collapsed so that only the needed instructions remain in focus. The same folding features available for XML documents are also available in XQuery documents.

Figure 4.124. Folding in XQuery documents

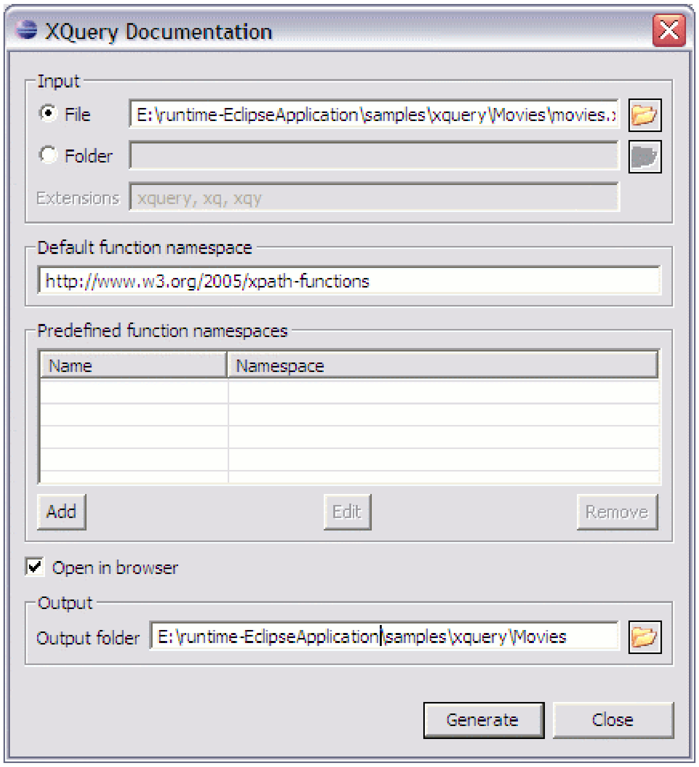


```
let $minRating := min($review/reviews/review[@movie-id = $movie-id]/rating)
return
 <movie id="{$movie/@id}">
  {$movie/title}
  {$movie/year}
 <avgRating>
      {
         if ($avgRating) then $avgRating else "not rated"
      }
 </avgRating>
  <maxRating>
      <value>
        {..
      </value>
      {..
  </maxRating>
  <minRating>
      <value>
        {..
      </value>
      {..
  </minRating>
 </movie>
```

Generating HTML Documentation for an XQuery Document

To generate HTML documentation for an XQuery document similar with the Javadoc documentation for Java classes use the dialog *XQuery Documentation*. It is opened with the action XML Tools → Generate Documentation → XQuery Documentation... . . It can be also opened form the Navigator contextual menu:Generate XQuery Documentation. The dialog enables the user to configure a set of parameters of the process of generating the HTML documentation. The parameters are:

Figure 4.125. The XQuery Documentation dialog



● File: E:\runtime-EclipseApplication\samples\xquery\Movies\movies.x

○ Folder:

Extensions: xquery, xq, xqy

Default function namespace: http://www.w3.org/2005/xpath-functions

☑ Open in browser

Output folder: E:\runtime-EclipseApplication\samples\xquery\Movies

Input

The *Input* panel allows the user to specify either the *File* or the *Folder* which contains the files for which to generate the documentation. One of the two text fields of the *Input* panel must contain the full path to the XQuery file. Extensions for the XQuery files contained in the specified directory can be added as comma separated values. Default there are offered xquery, xq, xqy.

Default function namespace

Optional URI for the default namespace for the submitted XQuery if it exists.

Predefined function namespaces

Optional engine dependent, predefined namespaces that the submitted XQuery refers to. They allow the conversion to generate annotation information to support the presentation component's hypertext linking if the predefined modules have been loaded into the local xqDoc XML repository.

Open in browser

When checked, the generated documentation will be opened in an external browser.

Output

Allows the user to specify where the generated documentation will be saved on disk.

Editing CSS stylesheets

<oXygen/> provides special support for developing CSS stylesheet documents.

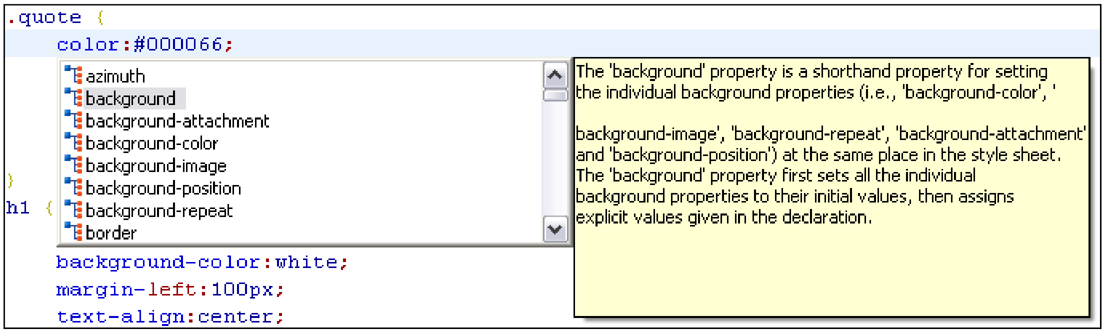
Validating CSS stylesheets

<oXygen/> includes a built-in CSS validator integrated with the general validation support. This brings the usual validation features to CSS stylesheets.

Content Completion in CSS stylesheets

A content completion assistant similar to the one of XML documents offers the CSS properties and the values available for each property. It is activated on the **CTRL + Space** shortcut and it is context sensitive when it is invoked for the value of a property.

Figure 4.126. Content Completion in CSS stylesheets



```
.quote {
    color:#000066;
}
h1 {
    background-color:white;
    margin-left:100px;
    text-align:center;
```

The 'background' property is a shorthand property for setting the individual background properties (i.e., 'background-color', '

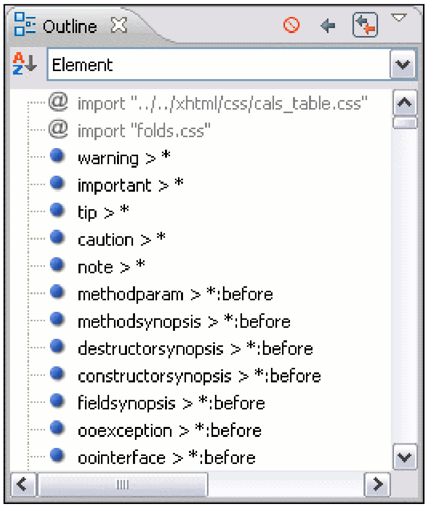
background-image', 'background-repeat', 'background-attachment' and 'background-position') at the same place in the style sheet. The 'background' property first sets all the individual background properties to their initial values, then assigns explicit values given in the declaration.

The properties and the values offered as proposals are dependent on the CSS Profile selected in the Options → Preferences+CSS Validator page, Profile combo box. The CSS 2.1 set of properties and property values is used for most of the profiles, excepting CSS 1 and CSS 3 for which specific proposal sets are used.

CSS Outline View

The *CSS Outline View* presents the import declarations of other stylesheet files and all the selectors defined in the current CSS document. The selector entries can be presented in the order they appear in the document or sorted by element name used in the selector or the entire selector string representation. The selection in the outline view can be synchronized with the caret moves or the changes made in the stylesheet document. When selecting an entry from the outline view the corresponding import or selector will be highlighted in the CSS editor.

Figure 4.127. CSS Outline View



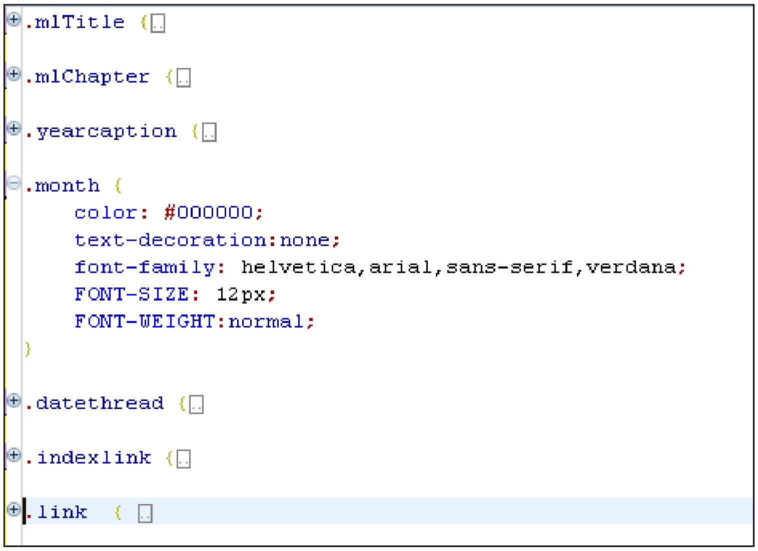
```
@ import "../../xhtml/css/cals_table.css"
@ import "folds.css"
● warning > *
● important > *
● tip > *
● caution > *
● note > *
● methodparam > *:before
● methodsynopsis > *:before
● destructorsynopsis > *:before
● constructorsynopsis > *:before
● fieldsynopsis > *:before
● ooexception > *:before
● oointerface > *:before
```

The selectors presented in the *CSS Outline View* can be quickly found using *key search*. When you press a sequence of character keys while the focus is in the outline view the first selector that starts with that sequence will be selected.

Folding in CSS stylesheets

In a large CSS stylesheet document some styles may be collapsed so that only the needed styles remain in focus. The same folding features available for XML documents are also available in CSS stylesheets.

Figure 4.128. Folding in CSS stylesheets



```
.mlTitle {

.mlChapter {

.yearcaption {

.month {
    color: #000000;
    text-decoration:none;
    font-family: helvetica,arial,sans-serif,verdana;
    FONT-SIZE: 12px;
    FONT-WEIGHT:normal;
}

.datethread {

.indexlink {

.link  {
```

Formatting and indenting CSS stylesheets (pretty print)

If the edited CSS stylesheet becomes unreadable because of the bad alignment of the text lines the pretty-print operation available for XML documents is also available for CSS stylesheets. It works in the same way as for XML documents and is available as the same menu and toolbar action.

Other CSS editing actions

The CSS editor type offers a reduced version of the popup menu available in the XML editor type, that means only the folding actions,the edit actions and a part of the source actions (only the actions *To lower case*, *To upper case*, *Capitalize lines*).

Changing the user interface language

<oXygen/> comes with the user interface translated in English, French, German, Italian, Japanese and Dutch. If you want to use <oXygen/> in other language you have to translate all the messages and labels available in the user interface (menu action names, button names, checkbox texts, view titles, error messages, status bar messages, etc.) and provide a text file with all the translated messages to <oXygen/> in the form of a Java properties file. Such a file contains pairs of the form message key - translated message displayed in the user interface. In order to install the new set of translated messages you must copy this file to the [oXygen-install-folder]/lib folder, restart <oXygen/> and set the new language in the <oXygen/> preferences. You can get the keys of all the messages that must be translated from the properties file containing the English translation used in <oXygen/>. To get this file contact us at support@oxygenxml.com.

Handling read-only files

If a file marked as read-only by the operating system is opened in <oXygen/> you will not be able to make modifications to it regardless of the page the file was opened in. You can check out the read-only state of the file by looking in the Properties view. If you modify the file's properties from the operating system and the file becomes writable you will be able to make modifications to it on the spot without having to reopen it.

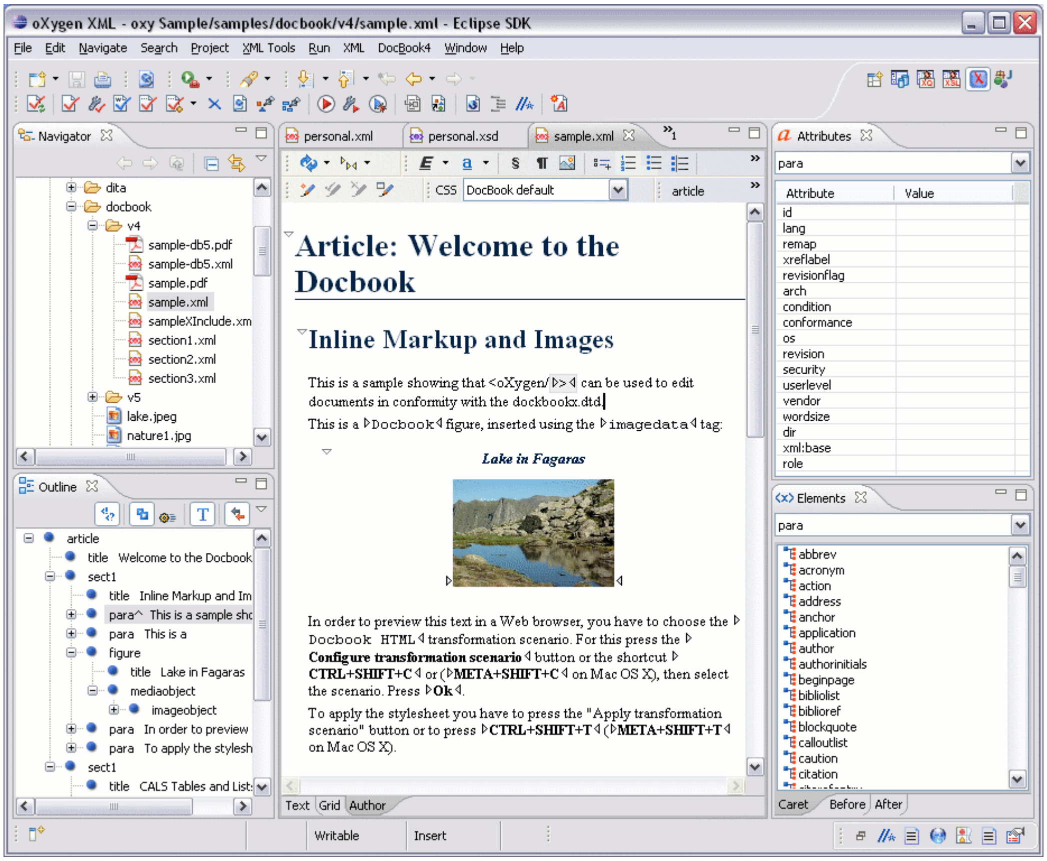
Chapter 5. Authoring in the tagless editor

Authoring XML documents without the XML tags

Once the structure of the XML document and the required restrictions on the elements and attributes are fixed with an XML schema the editing of the document is easier in a WYSIWYG (what-you-see-is-what-you-get) editor in which the XML markup is not visible.

This tagless editor is available as the Author mode of the XML editor. The Author mode is activated by pressing the Author button at the bottom of the editing area where the mode switches of the XML editor are available: Text, Grid and Author (see the following screenshot). The Author mode renders the content of the XML document visually based on a CSS stylesheet associated with the document. Many of the actions and features available in Text mode are also available in Author mode.

Figure 5.1. oXygen Author Editor



Article: Welcome to the Docbook

Inline Markup and Images

This is a sample showing that <oXygen/> can be used to edit documents in conformity with the dockbookx.dtd.

This is a Docbook figure, inserted using the imagedata tag:

Lake in Fagaras

In order to preview this text in a Web browser, you have to choose the Docbook HTML transformation scenario. For this press the **Configure transformation scenario** button or the shortcut **CTRL+SHIFT+C** or (**META+SHIFT+C** on Mac OS X), then select the scenario. Press **Ok**.

To apply the stylesheet you have to press the "Apply transformation scenario" button or to press **CTRL+SHIFT+T** (**META+SHIFT+T** on Mac OS X).

The tagless rendering of the XML document in the Author mode is driven by a CSS stylesheet which conforms to the version 2.1 of the CSS specification [http://www.w3.org/TR/CSS21/] from the W3C consortium. Also some CSS 3 features like namespaces and custom extensions of the CSS specification are supported.

The CSS specification is convenient for driving the tagless rendering of XML documents as it is an open standard maintained by the W3C consortium. A stylesheet conforming to this specification is very easy to develop and edit in <oXygen/> as it is a plain text file with a simple syntax.

The association of such a stylesheet with an XML document is also straightforward: an *xml-stylesheet* XML processing instruction with the attribute *type="text/css"* must be inserted at the beginning of the XML document. If it is an XHTML document, that is the root element is a **html** element, there is a second method for the association of a CSS stylesheet: an element **link** with the **href** and **type** attributes in the **head** child element of the **html** element as specified in the CSS specification [http://www.w3.org/TR/REC-html40/present/styles.html#h-14.3.2].

There are two main types of users of the Author mode: *developers* and *content authors*. A *developer* is a technical person with advanced XML knowledge who defines the framework for authoring XML documents in the tagless editor. Once the framework is created or edited by the developer it is distributed as a deliverable component ready to plug into the application to the content authors. A *content author* does not need to have advanced knowledge about XML tags or operations like validation of XML documents or applying an XPath expression to an XML document. He just plugs the framework set up by the developer into the application and starts editing the content of XML documents without editing the XML tags directly.

The framework set up by the developer is called *document type* and defines a type of XML documents by specifying all the details needed for editing the content of XML documents in tagless mode: the CSS stylesheet which drives the tagless visual rendering of the document, the rules for associating an XML schema with the document which is needed for content completion and validation of the document, transformation scenarios for the document, XML catalogs, custom actions available as buttons on the toolbar of the tagless editor.

The tagless editor comes with some ready to use predefined document types for XML frameworks largely used today like DocBook, DITA, TEI, XHTML.

The Content Author role

A content author edits the content of XML documents in tagless mode disregarding the XML tags as they are not visible in the editor. If he edits documents conforming to one of the predefined types he does not need to configure anything as the predefined document types are already configured when the application is installed. Otherwise he must plug the configuration of the document type into the application. This is as easy as unzipping an archive directly in the *frameworks* subfolder of the application's install folder.

In case the edited XML document does not belong to one of the document types set up in Preferences you can specify the CSS files to be used by inserting an *xml-stylesheet* processing instructions. You can insert the processing instruction by editing the document or by using the Associate XSLT/CSS stylesheet action.

The syntax of such a processing instruction is:

```
<?xml-stylesheet type="text/css" media="media type" title="title"
href="URL" alternate="yes|no"?>
```

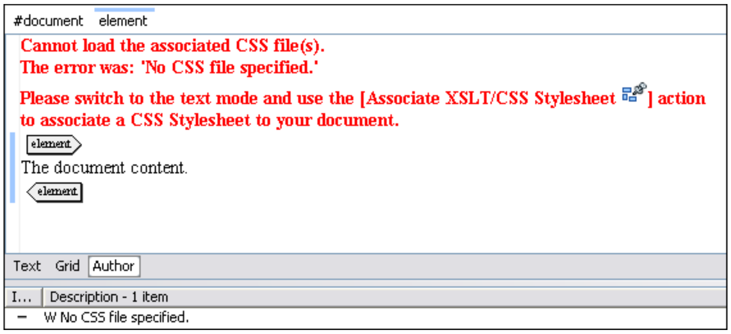
You can read more about associating a CSS to a document, the syntax and the use of the *xml-stylesheet* processing instruction in the section Author CSS Settings.

When the document has no CSS association or the referred stylesheet files cannot be loaded a default one will be used. A warning message will also be displayed at the beginning of the document presenting the reason why the CSS cannot be loaded.

Note

In general it is recommended to associate a CSS while in Text mode so that the whitespace normalization rules specified in the stylesheets will be properly applied when switching to Author mode.

Figure 5.2. Document with no CSS association default rendering



Cannot load the associated CSS file(s).
The error was: 'No CSS file specified.'

Please switch to the text mode and use the [Associate XSLT/CSS Stylesheet] action to associate a CSS Stylesheet to your document.

The document content.

W No CSS file specified.

Author views

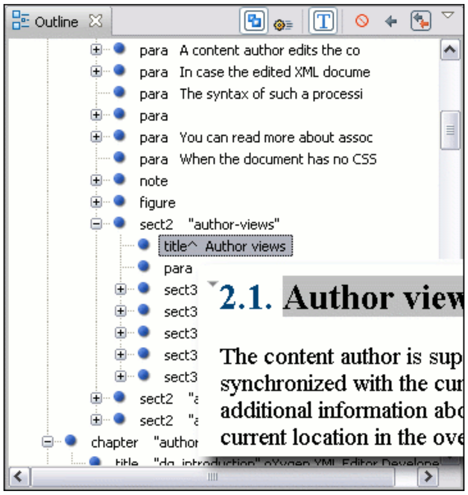
The content author is supported by special views which are automatically synchronized with the current editing context of the editor panel and which present additional information about this context thus helping the author to see quickly the current location in the overall document structure and the available editing options.

Outline view

The Outline view has the following available functions:

- the section called “XML Document Overview”
- the section called “Modification Follow-up”
- the section called “Document Structure Change”

Figure 5.3. The Outline View



```
para   A content author edits the co
para   In case the edited XML docume
para   The syntax of such a processi
para
para   You can read more about assoc
para   When the document has no CSS
note
figure
sect2   "author-views"
  title^  Author views
  para
  sect3
  sect3
  sect3
  sect3
  sect3
sect2   "a
sect2   "a
chapter   "author
title   "dg_introduction" oXygen XML Editor Develope
```

2.1. Author view

The content author is sup
synchronized with the cur
additional information abo
current location in the ove

XML Document Overview

The Outline view displays a general tag overview of the current edited XML Document. It also shows the correct hierarchical dependencies between the tag elements, making it easier for the user to be aware of the document's structure and the way tags are nested. It also allows the user to insert or delete nodes using pop-up menu actions.

Modification Follow-up

When editing, the Outline view dynamically follows the modifications introduced by the user, showing in the middle of the panel the node which is currently being modified. This gives the user a better insight on location inside the document and how the structure of the document is affected by one's modifications.

Document Structure Change

Entire XML elements can be moved or copied in the edited document using only the mouse in the Outline view in drag-and-drop operations. If you drag an XML element in the Outline view and drop it on another one in the same panel then the dragged element will be moved after the drop target element. If you hold the mouse pointer over the drop target for a short time before the drop then the drop target element will be expanded first and the dragged element will be moved inside the drop target element after its opening tag. You can also drop an element before or after another element if you hold the mouse pointer towards the upper or lower part of the targeted element. A marker will indicate whether the drop will be performed before or after the target element. If you hold down the CTRL key after dragging, there will be performed a copy operation instead of a move one.

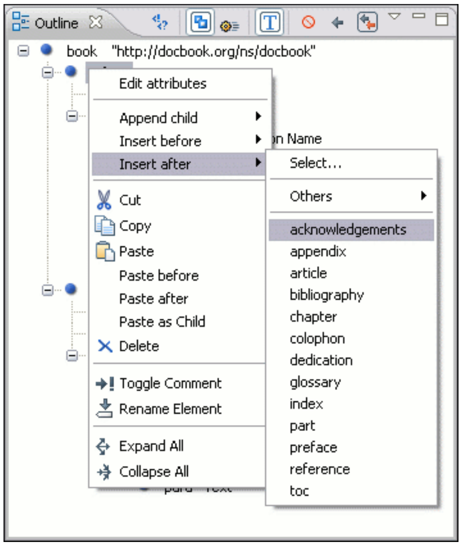
The drag and drop action in the Outline view can be disabled and enabled from the Preferences dialog.

Tip

You can select and drag multiple nodes in the Author Outliner tree.

The popup menu of the Outline tree

Figure 5.4. Popup menu of the Outline tree



Edit attributes for the selected node. A dialog is presented allowing the user to see and edit the attributes of the selected node. See here for more details about editing attributes.

The *Append child*, *Insert before* and *Insert after* submenus of the outline tree popup menu allow to quickly insert new tags in the document at the place of the element correctly selected in the Outline tree. The *Append child* submenu lists the names of all the elements which are allowed by the schema associated with the current document as child of the current element. The effect is the same as typing the '<' character and selecting an element name from the popup menu offered by the content completion assistant. The *Insert before* and *Insert after* submenus of the Outline tree popup menu list the elements which are allowed by the schema associated with the current document as siblings of the current element inserted immediately before respectively after the current element.

The *Cut*, *Copy* and *Delete* items of the popup menu execute the same actions as the Edit menu items with the same name on the elements currently selected in the outline tree (Cut, Copy, Paste). You can insert a well-formed element before, after or as a child of the currently selected element by accessing the *Paste before*, *Paste after* or *Paste as Child* actions.

The *Toggle Comment* item of the outline tree popup menu encloses the currently selected element of the outline tree in an XML comment, if the element is not commented, or removes the comment if it is commented.

Using the *Rename Element* action the element from the caret position and the elements that have the same name as the current element can be renamed according with the options from the Rename dialog.

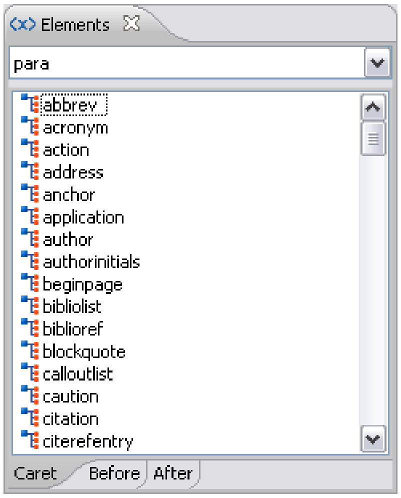
The *Expand All/Collapse All* actions expand/collapse the selection and all its children.

Tip

You can Copy/Cut or Delete multiple nodes in the Outliner by using the contextual menu after selecting all the nodes in the tree.

Elements view

Figure 5.5. The Elements View



Presents a list of all defined elements that you can insert in your document. All elements from a sequence are presented but the invalid proposals (which cannot be inserted in the current context) are grayed-out. The upper part of the view features a combo box that contains the current element's ordered ancestors. Selecting a new element in this combo box will update the list of the allowed elements in *Before* and *After* tabs.

Three tabs present information relative to the caret location:

- *Caret* shows a list of all the elements allowed at the current caret location. Double-clicking any of the listed elements will insert that element at the caret position.

- *Before* shows a list of all elements that can be inserted before the element selected in the combo box. Double-clicking any of the listed elements will insert that element before the element at the caret position.

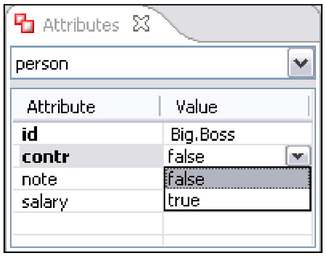
- *After* shows a list of all elements that can be inserted after the element selected in the combo box. Double-clicking any of the listed elements will insert that element after the element at the caret position.

Double clicking an element name in the list surrounds the current selection in the editor panel with the start tags and end tags of the element. If there is no selection just an empty element is inserted in the editor panel at the cursor position.

Attributes view

The Attributes panel presents all the possible attributes of the current element allowed by the schema of the document and allows to insert attributes in the current element or change the value of the attributes already used in the element. The attributes already present in the document are painted with a bold font. Default values are painted with an italic font. Clicking on the Value column of a table row will start editing the value of the attribute from the selected row. If the possible values of the attribute are specified as list in the schema associated with the edited document the Value column works as a combo box where you can select one of the possible values to be inserted in the document. The attributes table is sortable by clicking on the column names. Thus the table's contents can be sorted in ascending order, in descending order or in a custom order, where the used attributes are placed at the beginning of the table as they appear in the element followed by the rest of the allowed elements as they are declared in the associated schema.

Figure 5.6. The Attributes View



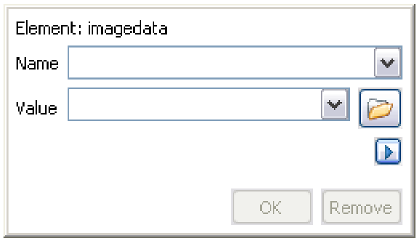
Attribute	Value
id	Big.Boss
contr	false
note	
salary	

A combo box located in the upper part of the view allows you to edit the attributes of the ancestors of the current element.

The contextual menu of the view allows you to insert a new element (*Add* action) or delete an existing one (*Delete* action). Delete action can be invoked on a selected table entry by pressing *DEL* or *BACKSPACE*.

The attributes of an element can be edited also in place in the editor panel by pressing the shortcut Alt + Enter which pops up a small window with the same content of the Attributes view. In the initial form of the popup only the two text fields Name and Value are displayed, the list of all the possible attributes is collapsed.

Figure 5.7. Edit attributes in place

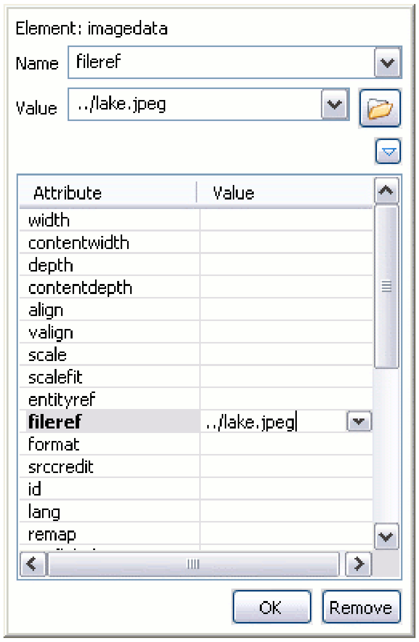

Element: imagedata

Name

Value

The small arrow button next to the Cancel button expands the list of possible attributes allowed by the schema of the document as in the Attributes panel.

Figure 5.8. Edit attributes in place - full version



Element: imagedata

Name: fileref

Value: ../lake.jpeg

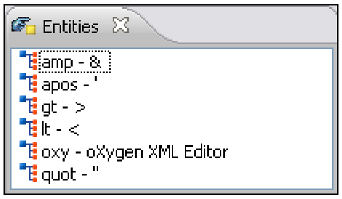
Attribute	Value
width	
contentwidth	
depth	
contentdepth	
align	
valign	
scale	
scalefit	
entityref	
fileref	../lake.jpeg
format	
srccredit	
id	
lang	
remap	

The Name field auto-completes the name of the attribute: the complete name of the attribute is suggested based on the prefix already typed in the field as the user types in the field.

Entities view

This view displays a list with all entities declared in the current document as well as built-in ones. Double clicking one of the entities will insert it at the current cursor position.

Figure 5.9. The Entities View



- amp - &
- apos - '
- gt - >
- lt - <
- oxy - oXygen XML Editor
- quot - "

The Author editor

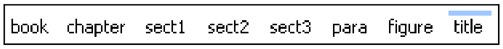
In order to view the XML file in Author view, the XML document must be associated with a CSS file that defines the way the XML file is rendered. The document can be edited as text, the XML markup being hidden by default.

Navigating the document content

Fast navigating the document content can be done using the **Tab**/**Shift** + **Tab** for advancing forward / backwards. The caret will be moved to the next/previous editable position. Entities and hidden elements will be skipped.

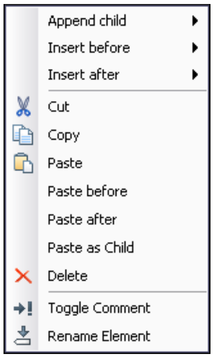
A left-hand side stripe paints a vertical thin light blue bar indicating the vertical span of the element found at caret position. Also a top stripe indicates the path from document root to the current element.

Figure 5.10. Top stripe in Editor view



The last element is also highlighted by a thin light blue bar for easier identification. Clicking one element from the top stripe selects the entire element in the Editor view.

Figure 5.11. The top stripe pop-up menu



The *Append child*, *Insert before* and *Insert after* submenus allow to quickly insert new tags in the document at the place of the selected element. The *Append child* submenu lists the names of all the elements which are allowed by the schema associated with the current document as child of the current element. The effect is the same as typing the '<' character and selecting an element name from the popup menu offered by the content completion assistant. The *Insert before* and *Insert after* submenus list the elements which are allowed by the schema associated with the current document as siblings of the current element inserted immediately before respectively after the current element.

The *Cut*, *Copy*, *Paste* and *Delete* items of the popup menu execute the same actions as the Edit menu items with the same name on the elements currently selected in the stripe (Cut, Copy, Paste, Delete). The styles of the copied content is preserved by the *Cut* and *Copy* operations, for example the *display:block* property or the tabular format of the data from a set of table cells. The *Paste before*, *Paste after* and *Paste as Child* actions allow the user to insert an well-formed element before, after or as a child of the currently selected element.

The *Toggle Comment* item of the outline tree popup menu encloses the currently selected element of the top stripe in an XML comment, if the element is not commented, or removes the comment if it is commented.

Using the *Rename Element* action the selected element and the elements that have the same name as the current element can be renamed according with the options from the Rename dialog.

When working on a large document the **folding support** can be used to collapse some elements content leaving in focus only the ones you need to edit. Foldable elements are marked with a small triangle painted in the upper left corner. Hovering with the mouse pointer over that marker, the entire content of the element is highlighted by a dotted border for quick identification of the foldable area.

When working on a suite of documents that refer to one another(references, external entities, XInclude, DITA conref, etc), the **linking support** is useful for navigating between the documents. In the predefined customizations that are bundled with <oXygen/> links are marked with an icon representing a chain link: 🔗 . When hovering with the mouse pointer over the marker, the mouse pointer will change to indicate that the link can be followed and a tooltip will present the destination location. Clicking on a followable link will result in the referred resource being opened in an editor. The same effect can be obtained by using the action *Open file at caret* when the caret is in a followable link element.

To position the cursor at the beginning or at the end of the document you can use *Ctrl+Home* and *Ctrl+End*, respectively.

Displaying the markup

In Author view, the amount of displayed markup can be controlled using the following dedicated actions:

- Full Tags with Attributes - displays full name tags with attributes for both block level as well as in-line level elements.
- Full Tags - displays full name tags without attributes for both block level as well as in-line level elements.
- Block Tags - displays full name tags for block level elements and simple tags without names for in-line level elements.
- Inline Tags - displays full name tags for in-line level elements, while block level elements are not displayed.
- Partial Tags - displays simple tags without names for in-line level elements, while block level elements are not displayed.
- No Tags - none of the tags is displayed. This is the most compact mode.

The default tags display mode can be configured in the Author options page. However, if the document opened in Author editor does not have an associated CSS stylesheet, then the *Full Tags* mode will be used.

Block-level elements are those elements of the source document that are formatted visually as blocks (e.g., paragraphs), while the inline level elements are distributed in lines (e.g., emphasizing pieces of text within a paragraph, in-line images, etc). The graphical format of the elements is controlled from the CSS sources via the *display* property.

Bookmarks

A position in a document can be marked with a bookmark. Later the cursor can go quickly to the marked position with a keyboard shortcut or with a menu item. This is useful for easy navigation in a large document or for working on more than one document at a moment when the cursor must move between several marked positions.

A bookmark can be placed with one of the menu items available on the menu Edit → Bookmarks → Create or with the menu item Edit → Bookmarks → Bookmarks Quick Creation or with the keyboard shortcuts associated with these menu items and visible on the menu Edit → Bookmarks. A bookmark can be removed when a new bookmark is placed in the same position as an old one or with the action Edit → Bookmarks → Remove All. The cursor can go to a bookmark with one of the actions available on the menu Edit → Bookmarks → Go to.

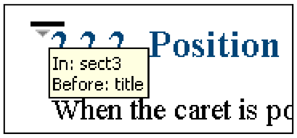
Position information tooltip

When the caret is positioned next to an element tag, a tooltip will be shown for a couple of seconds displaying the position of the caret relative to the current element context.

Here are the common situations that can be encountered:

- The caret is positioned before the first children of the current node.

Figure 5.12. Before first child



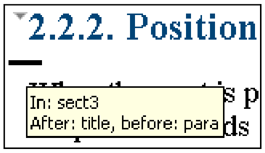
2.2.2. Position

In: sect3
Before: title

When the caret is pc

- The caret is positioned between the start and end of two sibling nodes.

Figure 5.13. Between two siblings

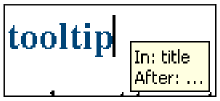


2.2.2. Position

In: sect3
After: title, before: para

- The caret is positioned after the last child of the current node.

Figure 5.14. After last child

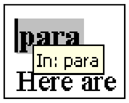


tooltip

In: title
After: ...

- The caret is positioned in an empty node.

Figure 5.15. Empty node



para

In: para

Here are

The nodes in the previous cases are displayed in the tooltip window using their names. When one of them is a text node it will be presented using "..." sequence.

You can deactivate this feature by unchecking Options → Preferences+Editor / Author+Show caret positioned info checkbox. Even if this option is disabled, you can trigger the display of the position tooltip by pressing Shift+F2.

Note

The position information tooltip is not displayed if one of the modes *Full Tags with Attributes* or *Full Tags* is selected.

Displaying referred content

The referred content (entities, XInclude, DITA conref, etc) will be resolved and displayed by default. You can control this behavior from the Author options page.

The referred resources are loaded and displayed inside the element or entity that refers them, however the displayed content cannot be modified directly.

When the referred resource cannot be resolved, an error will be presented inside the element that refers them instead of the content.

If you want to make modifications to the referred content, you must open the referred resource in an editor. The referred resource can be opened quickly by clicking on the link (marked with the icon 🔗) which is displayed before the referred content. The referred resource is resolved through the XML Catalog set in **Preferences**.

To update the displayed referred content so that it reflects the latest modifications of the referred resource, you can use the Refresh references action. Please note that the content of the expanded external entities can only be refreshed by using the Reload action.

Finding and replacing text

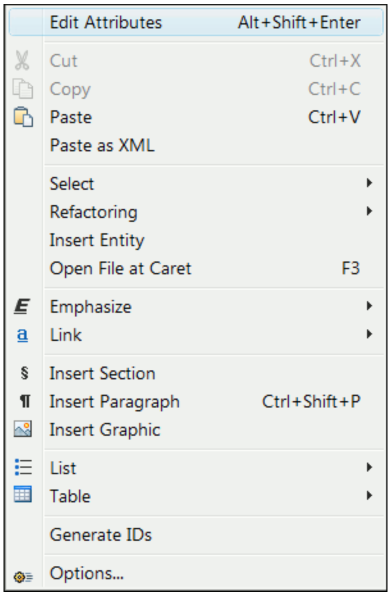
The Find/Replace dialog can be used in the Author page in the same way as in the Text page.

These limitations can be compensated by using the Find All Elements dialog.

Contextual menu

More powerful support for editing the XML markup is offered via actions included in the contextual menu. Two types of actions are available: **generic actions**(actions that not depends on a specific document type) and **document type actions**(actions that are configured for a specific document type).

Figure 5.16. Contextual menu



The generic actions are:

- **Cut**, **Copy**, **Paste** - common edit actions with the same functionality as those found in the text editor.
- **Paste As XML** - similar to **Paste** operation, except that the clipboard's content is considered to be XML.
- **Select** - contains the following actions:
  - **Select -> Select Element** - selects the entire element at the current caret position.

    - **Select -> Select Content** - selects the entire content of the element at the current caret position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

    - **Select -> Select Parent** - selects the parent of the element at the current caret position.

    ### Note

    You can select an element by triple clicking inside its content. If the element is empty you can select it by double clicking it.

- **Refactoring** - contains a series of actions designed to alter the document's structure:

    - **Toggle Comment** - encloses the currently selected text in an XML comment, or removes the comment if it is commented;

    - **Split Element** - splits the content of the closest element that contains the caret's position. Thus, if the caret is positioned at the beginning or at the end of the element, the newly created sibling will be empty;

    - **Join Elements** - joins two adjacent elements that have the same name. The action is available only when the caret position is between the two adjacent elements. Also, joining two elements can be done by pressing the Delete or Backspace keys and the caret is positioned between the boundaries of these two elements.

    - **Surround with Tag...** - selected text in the editor is marked with the specified tag.

    - **Surround with '<Tag name>'** - selected text in the editor is marked with start and end tags of the last '**Surround with Tag...**' action.

    - **Rename Element** - the element from the caret position and the elements that have the same name as the current element can be renamed according with the options from the Rename dialog.

    - **Delete Element Tags** - deletes the tags of the closest element that contains the caret's position. This operation is also executed if the start or end tags of an element are deleted by pressing the *Delete* or *Backspace* keys.

- **Insert Entity** - allows the user to insert a predefined entity or a character entity. Surrogate character entities (range #x10000 to #x10FFFF) are also accepted.

    Character entities can be entered in one of the following forms:

    - #<*decimal value*> - e.g. #65

    - &#<*decimal value*>; - e.g. &#65;

    - #x<*hexadecimal value*> - e.g. #x41

    - &#x<*hexadecimal value*>; - e.g. &#x41;

- **Open File at Cursor** - opens in a new editor panel the file with the name under the current position of the caret in the current document. If the file does not exist at the specified location the error dialog that is displayed contains a Create new file action which displays the **New** file dialog. This allows you to choose the type or the template for the file. If the action succeeds, the file is created with the referred location and name and is opened in a new editor panel. This is useful when you decide first on the file name and after that you want to create it in the exact location specified at the current cursor position.

Document type actions are specific to some document type. Examples of such actions can be found in section Predefined document types.

Editing XML in <oXygen/> Author

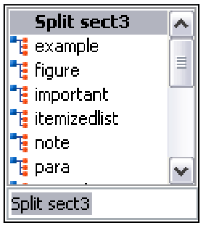
Editing the XML markup

One of the most useful feature in Author editor is the content completion support. The fastest way to invoke it is to press Ctrl + Space (on *Mac OS X* the shortcut is Meta + Space).

Content completion window offers the following types of actions:

- inserting allowed elements for the current context according to the associated schema, if any;
- inserting element values if such values are specified in the schema for the current context;
- inserting new undeclared elements by entering their name in the text field;
- inserting CDATA sections, comments, processing instructions.

Figure 5.17. Content completion window



If you press *Enter* the displayed content completion window will contain as first entry the *Split <Element name>* item. Selecting it splits the content of the closest element that contains the caret's position. Thus, if the caret is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

If the caret is positioned inside a space preserve element the first choice in the content completion window is *Enter* which inserts a new line in the content of the element. If there is a selection in the editor and you invoke content completion, a *Surround with* operation can be performed. The tag used will be the selected item from the content completion window.

Joining two elements. You can choose to join the content of two sibling elements with the same name by using the Join elements action from the editor contextual menu.

The same action can be triggered also in the next situations:

- The caret is located before the end position of the first element and *Delete* key is pressed.
- The caret is located after the end position of the first element and *Backspace* key is pressed.
- The caret is located before the start position of the second element and *Delete* key is pressed.
- The caret is located after the start position of the second element and *Backspace* key is pressed.

In either of the described cases, if the element has no sibling or the sibling element has a different name, *Unwrap* operation will be performed automatically.

Unwrapping the content of an element You can unwrap the content of an element by deleting its tags using the Delete element tags action from the editor contextual menu.

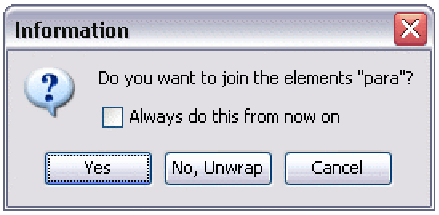
The same action can be triggered in the next situations:

- The caret is located before the start position of the element and *Delete* key is pressed.
- The caret is located after the start position of the element and *Backspace* key is pressed.
- The caret is located before the end position of the element and *Delete* key is pressed.
- The caret is located after the end position of the element and *Backspace* key is pressed.

Removing all the markup of an element You can remove the markup of the current element and keep only the text content with the action Remove All Markup available on the submenu **Refactoring** of the contextual menu and on the toolbar **XML Refactoring**.

When you press *Delete* or *Backspace* in the presented cases a dialog will be displayed allowing you to choose between *Join* or *Unwrap* operations. If the current element is empty, no dialog will be presented and the element tags will be deleted.

Figure 5.18. Join/Unwrap dialog



Information

Do you want to join the elements "para"?

☐ Always do this from now on

When you click on a marker representing the start or end tag of an element, the entire element will be selected. The contextual menu displayed when you right-click on the marker representing the start or end tag of an element contains *Append child*, *Insert Before* and *Insert After* submenus as first entries.

Editing the XML content

Entire sections or chunks of data can be moved or copied by using the *Drag and Drop* support. The following situations can be encountered:

- when both the drag and drop sources are Author pages, an well-formed XML fragment is transferred. The section will be balanced before dropping it by adding matching tags when needed.
- when the drag source is the Author page but the drop target is a text based editor only the text inside the selection will be transferred as it is.
- the text dropped from another text editor or another application into the Author page will be inserted without changes.

The font size of the current WYSIWYG-like editor can be increased and decreased on the fly with the same actions as in the Text editor:

Ctrl-NumPad+ or Ctrl-+ or Ctrl-mouse wheel	increase font size
Ctrl-NumPad- or Ctrl-- or Ctrl-mouse wheel	decrease font size

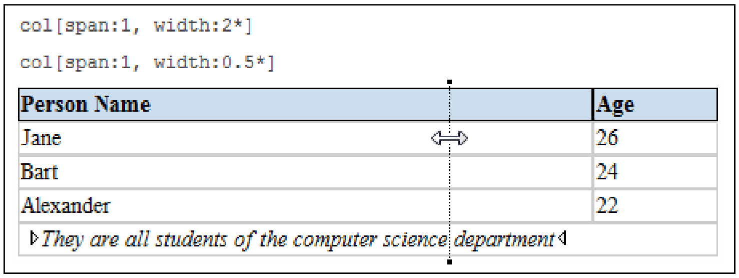
Ctrl-NumPad0 or Ctrl-0	restore font size to the size specified in Preferences

Removing the text content of the current element You can remove the text content of the current element and keep only the markup with the action Remove Text available on the submenu **Refactoring** of the contextual menu and on the toolbar **XML Refactoring**. This is useful when the markup of an element must be preserved, for example a table structure but the text content must be replaced.

Table layout and resizing

The support for editing data in tabular form can manage table width and column width specifications from the source document. The specified widths will be considered when rendering the tables and when visually resizing them using mouse drag gestures. These specifications are supported both in fixed and proportional dimensions. The predefined frameworks (DITA, DocBook and XHTML) already implement support for this feature. The layout of the tables from these types of documents takes into account the table width and the column width specifications particular to them. The tables and columns widths can be visually adjusted by dragging with the mouse their edges and the modifications will be committed back into the source document.

Figure 5.19. Resizing a column in <oXygen/> Author editor



```
col[span:1, width:2*]
col[span:1, width:0.5*]
```

Person Name	Age
Jane	26
Bart	24
Alexander	22

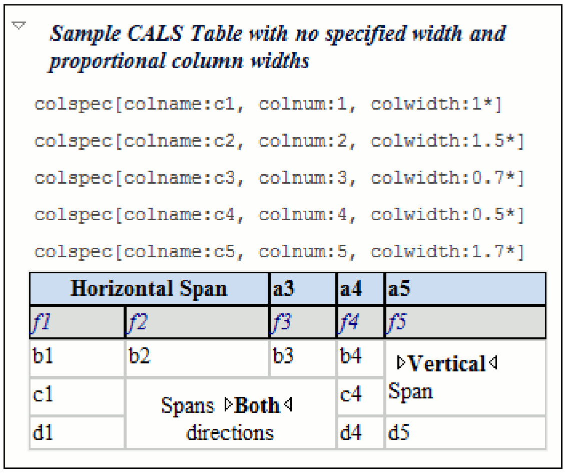
They are all students of the computer science department

DocBook

The DocBook table layout supports two models: CALS and HTML.

In the CALS model only column widths can be specified by using the `colwidth` attribute of the associated `colspec` element. The values can be fixed or proportional.

Figure 5.20. CALS table



Sample CALS Table with no specified width and proportional column widths

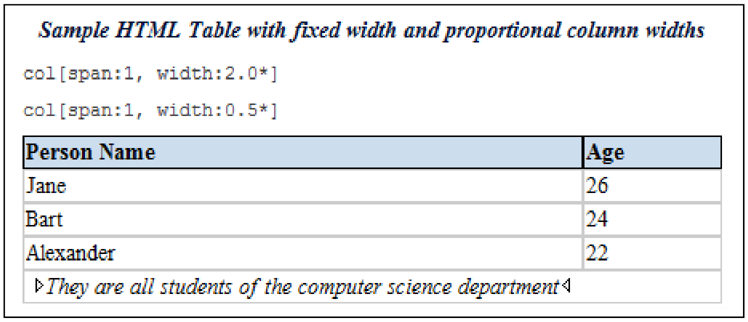
```
colspec[colname:c1, colnum:1, colwidth:1*]
colspec[colname:c2, colnum:2, colwidth:1.5*]
colspec[colname:c3, colnum:3, colwidth:0.7*]
colspec[colname:c4, colnum:4, colwidth:0.5*]
colspec[colname:c5, colnum:5, colwidth:1.7*]
```

Horizontal Span		a3	a4	a5
f1	*f2*	*f3*	*f4*	*f5*
b1	b2	b3	b4	**Vertical** Span
c1	Spans **Both** directions		c4	
d1			d4	d5

XHTML

The HTML table model accepts both table and column widths by using the width attribute of the table element and the col element associated with each column. The values can be represented in fixed units, proportional units or percentages.

Figure 5.21. HTML table



Sample HTML Table with fixed width and proportional column widths

col[span:1, width:2.0*]

col[span:1, width:0.5*]

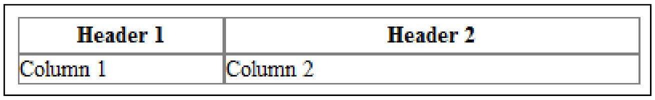
Person Name	Age
Jane	26
Bart	24
Alexander	22
They are all students of the computer science department	

DITA

The DITA table layout accepts CALS tables and simple tables.

The simple tables accept only relative column width specifications by using the `relcolwidth` attribute of the simpletable element.

Figure 5.22. DITA simple table



Header 1	Header 2
Column 1	Column 2

Refreshing the content

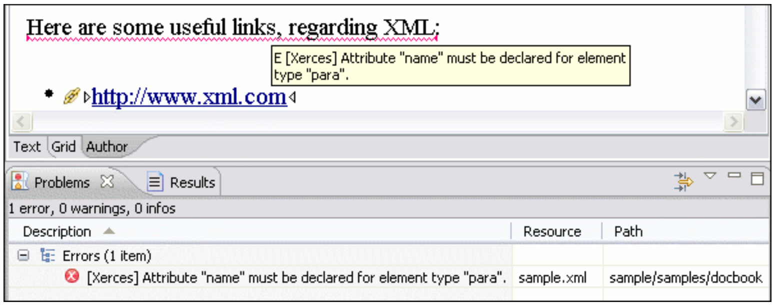
On occasion you may need to reload the content of the document from the disk or reapply the CSS. This can be performed by using the **Reload** action.

For refreshing the content of the referred resources you can use the action **Refresh references**. This action affects the displayed referred content, such as: references, XInclude, DITA conref, etc. However, this action will not refresh the expanded external entities, to refresh those you will need to use the Reload action.

Validation and error presenting

You can validate or check the XML form of the documents while editing them in Author Editor. Validate as you type as well as validate on request operations are available. Author editor offers validation features and configuring possibilities similar to text editor. You can read more about checking the XML form of documents in section Checking XML form. A detailed description of the document validation process and its configuration is described in section Validating Documents.

Figure 5.23. Error presenting in <oXygen/> Author editor


Here are some useful links, regarding XML:

E [Xerces] Attribute "name" must be declared for element type "para".

- http://www.xml.com

1 error, 0 warnings, 0 infos

Description	Resource	Path
Errors (1 item)		
[Xerces] Attribute "name" must be declared for element type "para".	sample.xml	sample/samples/docbook

A fragment with a validation error or warning will be marked in the editor panel by underlining the error region with a red color. The same will happen for a validation warning, only the color will be yellow instead of red.

Status messages from every validation action are logged into the Console view.

Whitespace handling

There are several major aspects of white-space handling in the <oXygen/> Author editor when opening documents or switching to Author mode, saving documents or switching from Author mode to another one and editing documents.

Open documents

When deciding if the white-spaces from a text node are to be preserved, normalized or stripped, the following rules apply:

- If the text node is inside an element context where the *xml:space="preserve"* is set then the white-spaces are preserved.
- If the CSS property *white-space* is set to *"pre"* for the node style then the white-spaces are preserved.
- If the text node contains other non-white-space characters then the white-spaces are normalized.
- If the text node contains only white-spaces:
  - If the node has a parent element with the CSS *display* property set to *inline* then the white-spaces are normalized.
  - If the left or right sibling is an element with the CSS *display* property set to *inline* then the white-spaces are normalized.
  - If one of its ancestors is an element with the CSS *display* property set to *table* then the white-spaces are striped.
  - Otherwise the white-spaces are ignored.

Save documents

The Author editor will try to format and indent the document while following the white-space handling rules:

- If text nodes are inside an element context where the *xml:space="preserve"* is set then the white-spaces are written without modifications.
- If the CSS property *white-space* is set to *"pre"* for the node style then the white-spaces are written without any changes.

- In other cases the text nodes are wrapped.

Also, when formatting and indenting an element that is not in a *space-preserve* context, additional *Line Separators* and white-spaces are added as follows:

- Before a text node that starts with a white-space.
- After a text node that ends with a white-space.
- Before and after CSS *block* nodes.
- If the current node has an ancestor that is a CSS *table* element.

Editing documents

You can insert *space* characters in any text nodes. *Line breaks* are permitted only in *space-preserve* elements. Tabs are marked in the space-preserve elements with a little marker.

Note

CDATA sections, *comments*, *processing instructions* have by default the *white-space* CSS property set to *"pre"* unless overridden in the CSS file you are using. Also they are considered to be *block* nodes.

Minimize differences between versions saved on different computers

The number of differences between versions of the same file saved by different content authors on different computers can be minimized by imposing the same set of formatting options when saving the file, for all the content authors. An example for a procedure that minimizes the differences is:

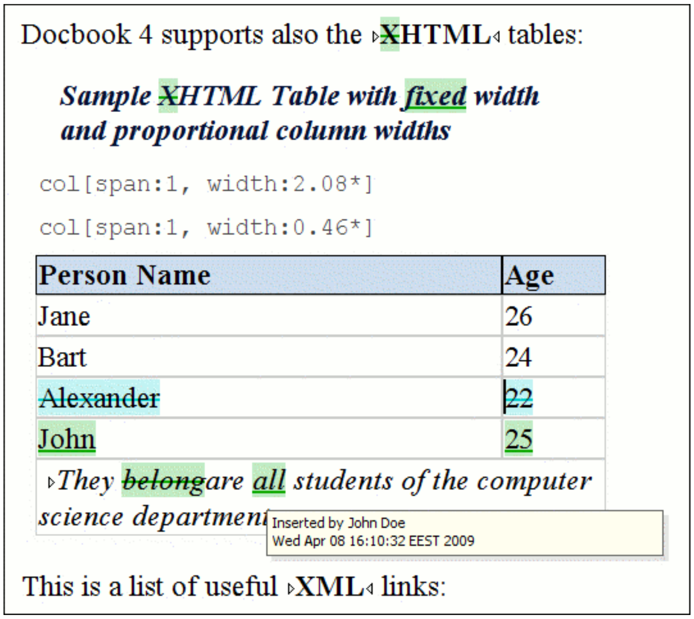
1. Create an <oXygen/> project file that will be shared by all content authors.
2. Set your own preferences in the following panels of the Preferences dialog: Editor / Format and Editor / Format / XML.
3. Save the preferences of these two panels in the <oXygen/> project by selecting the button *Project Options* in these two panels.
4. Save the project and commit the project file to your versioning system so all the content authors can use it.
5. Make sure the project is opened in the *Project* view and open your XML files in the Author mode and save them.
6. Commit the saved XML files to your versioning system.

When other content authors will change the files only the changed lines will be displayed in your diff tool instead of one big change that does not allow to see the changes between two versions of the file.

Change Tracking

Track Changes is a way to keep track of the changes you make to a document. You can activate change tracking for the current document by choosing Edit+Track Changes or by clicking the Track Changes button located on the Author toolbar. When *Track Changes* is enabled your modifications will be highlighted using a distinctive color. The name of the author who is currently making changes and the colors can be customized from the Track Changes preferences page.

Figure 5.24. Change Tracking in <oXygen/> Author



Docbook 4 supports also the ▹~~X~~HTML◃ tables:

Sample ~~X~~HTML Table with fixed width and proportional column widths

```
col[span:1, width:2.08*]

col[span:1, width:0.46*]
```

Person Name	Age
Jane	26
Bart	24
~~Alexander~~	~~22~~
John	25

▹*They ~~belong~~are all students of the computer science department*

Inserted by John Doe
Wed Apr 08 16:10:32 EEST 2009

This is a list of useful ▹XML◃ links:

When hovering a change the tooltip will display information about the author and modification time.

If the selection in the Author contains track changes and you Copy it the clipboard will contain the selection with all the changes *accepted*. This filtering will happen only if the selection is not entirely inside a tracked change.

Tip

For each change the author name and the modification time are preserved. The changes are stored in the document as processing instructions and they do not interfere with validating and transforming it.

Managing changes

You can review the changes made by you or other authors and then accept or reject them using the Track Changes toolbar buttons or the similar actions from the Edit menu.

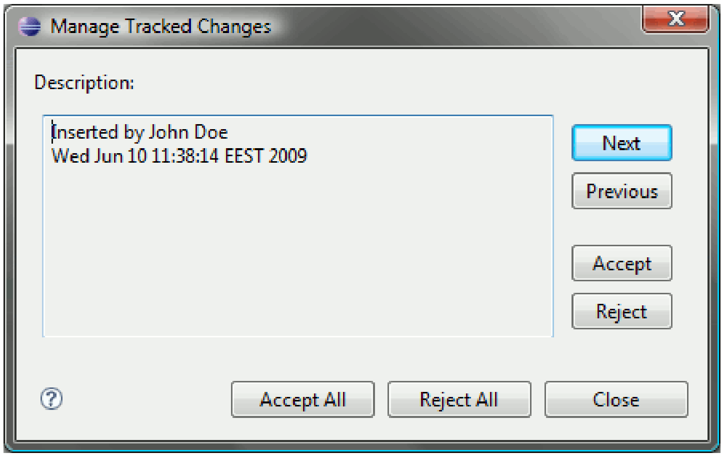
Track Changes
Enable or disable track changes for the current document.

Accept Change
Accept the change located at the caret position. For an insert change this means keeping the inserted text and for a delete change this means removing the content from the document. The action is also available in the Author page contextual menu.

Reject Change
Reject the change located at the caret position. For an insert change this means removing the inserted text and for a delete change this preserving the original content from the document. The action is also available in the Author page contextual menu.

Manage Tracked Changes
This is a way to find and manage all changes in the current document.

Figure 5.25. Manage Tracked Changes



Inserted by John Doe
Wed Jun 10 11:38:14 EEST 2009

The dialog offers the following actions:

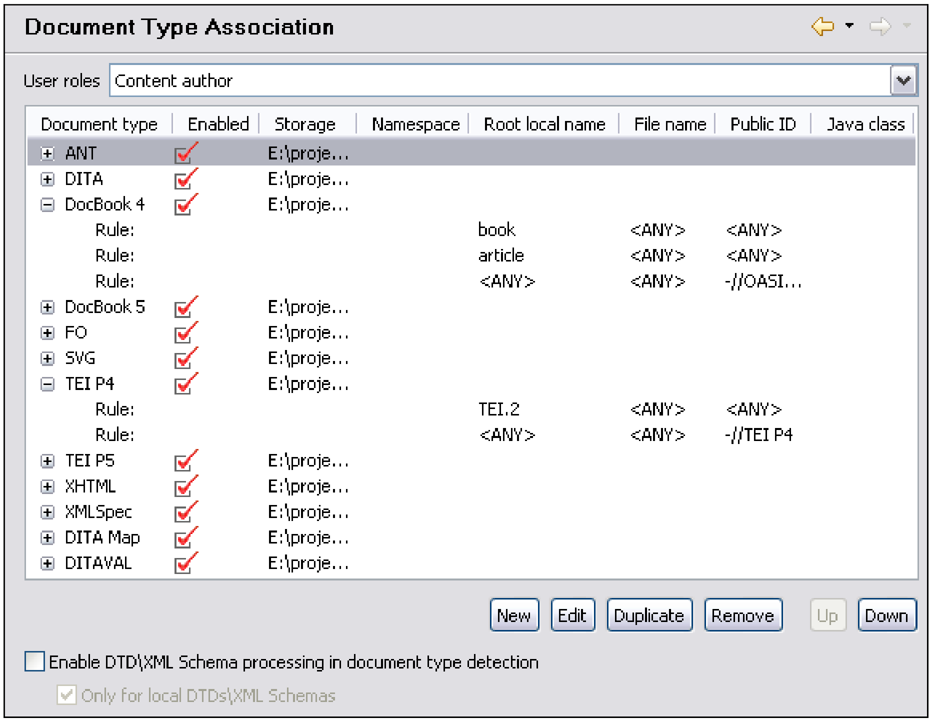
Next	Find the next change in the document.
Previous	Find the previous change in the document.
Accept	Accept the current change.
Reject	Reject the current change.
Accept All	Accept all changes in the document.
Reject All	Reject all changes in the document.

Chapter 6. Predefined document types

A document type is associated to an XML file according to its defined rules and it specifies many settings used to improve editing the category of XML files it applies for. These settings include specifying a default grammar used for validation and content completion, default scenarios used for transformation, specifying directories with file templates, specifying catalogs and a lot of settings which can be used to improve editing in the Tagless editor.

The tagless editor comes with some predefined document types already configured when the application is installed on the computer. These document types describe well-known XML frameworks largely used today for authoring XML documents. Editing a document which conforms to one of these types is as easy as opening it or creating it from one of the predefined document templates which also come with the application.

Figure 6.1. Document Type preferences page



Document Type Association

User roles: Content author

Document type	Enabled	Storage	Namespace	Root local name	File name	Public ID	Java class
⊞ ANT	☑	E:\proje...					
⊞ DITA	☑	E:\proje...					
⊟ DocBook 4	☑	E:\proje...					
Rule:				book	<ANY>	<ANY>	
Rule:				article	<ANY>	<ANY>	
Rule:				<ANY>	<ANY>	-//OASI...	
⊞ DocBook 5	☑	E:\proje...					
⊞ FO	☑	E:\proje...					
⊞ SVG	☑	E:\proje...					
⊟ TEI P4	☑	E:\proje...					
Rule:				TEI.2	<ANY>	<ANY>	
Rule:				<ANY>	<ANY>	-//TEI P4	
⊞ TEI P5	☑	E:\proje...					
⊞ XHTML	☑	E:\proje...					
⊞ XMLSpec	☑	E:\proje...					
⊞ DITA Map	☑	E:\proje...					
⊞ DITAVAL	☑	E:\proje...					

☐ Enable DTD\XML Schema processing in document type detection

☑ Only for local DTDs\XML Schemas

The DocBook V4 document type

DocBook is a very popular set of tags for describing books, articles, and other prose documents, particularly technical documentation.

Association rules

A file is considered to be a DocBook document when either of the following occurs:

- root element name is a *book* or *article*;
- public id of the document contains *-//OASIS//DTD DocBook XML*.

Schema

The schema used for DocBook documents is in *${frameworks}/docbook/dtd/docbookx.dtd*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

Author extensions

The CSS file used for rendering DocBook content is located in *${frameworks}/docbook/css/docbook.css*.

Specific actions for DocBook documents are:

- **B** Bold emphasized text - emphasizes the selected text by surrounding it with *<emphasis role="bold"/>* tag.
- *I* Italic emphasized text - emphasizes the selected text by surrounding it with *<emphasis role="italic"/>* tag.
- U Underline emphasized text - emphasizes the selected text by surrounding it with *<emphasis role="italic"/>* tag.

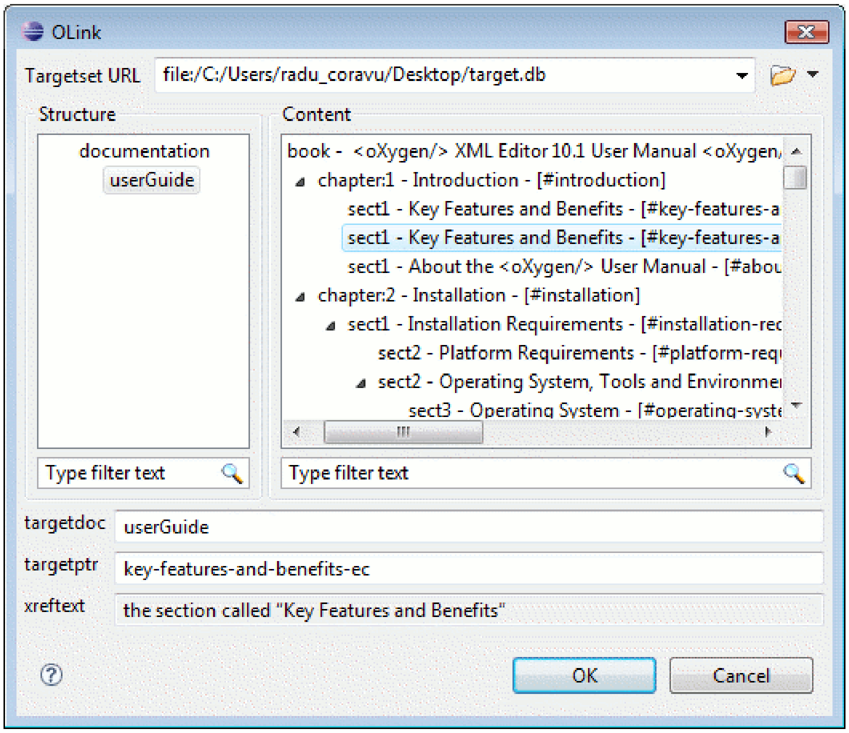
  ### Note

  For all of the above actions if there is no selection then a new 'emphasis' tag with specific role will be inserted. These actions are available in any document context.

  These actions are grouped under the *Emphasize* toolbar actions group.

- link - inserts a hypertext link.
- ulink - inserts a link that address its target by means of an URL (Universal Resource Locator).
- olink - inserts a link that address its target indirectly, using the `targetdoc` and `targetptr` values which are present in a `Targetset` file.

Figure 6.2. Insert OLink Dialog



Targetset URL: file:/C:/Users/radu_coravu/Desktop/target.db

documentation
userGuide

book - <oXygen/> XML Editor 10.1 User Manual <oXygen/
chapter:1 - Introduction - [#introduction]
sect1 - Key Features and Benefits - [#key-features-a
sect1 - Key Features and Benefits - [#key-features-a
sect1 - About the <oXygen/> User Manual - [#abou
chapter:2 - Installation - [#installation]
sect1 - Installation Requirements - [#installation-rec
sect2 - Platform Requirements - [#platform-requ
sect2 - Operating System, Tools and Environme
sect3 - Operating System - [#operating-syste

targetdoc: userGuide

targetptr: key-features-and-benefits-ec

xreftext: the section called "Key Features and Benefits"

After you choose the `Targetset` URL the structure of the target documents is presented. For each target document (`targetdoc`) the content is displayed allowing for easy identification of the `targetptr` for the `olink` element which will be inserted. You can use the Search fields to quickly identify a target. If you already know the values for the `targetdoc` and `targetptr` you can insert them directly in the corresponding fields. You have also the possibility to edit an olink using the action **Edit OLink** available on the contextual menu. The action make sense only if the dialog was already displayed with a proper `Targetset`.

- uri - inserts an URI element. The URI identifies a Uniform Resource Identifier (URI) in content.
- xref - inserts a cross reference to another part of the document. The initial content of the xref is automatically detected from the target.

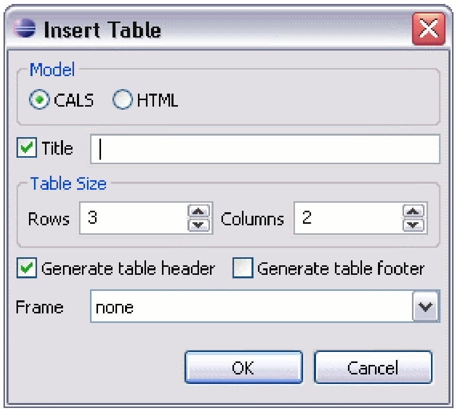
Note

These actions are grouped under the *Link* toolbar actions group.

- § Insert Section - inserts a new section/subsection in the document, depending on the current context. For example if the current context is *sect1* then a *sect2* will be inserted and so on.
- ¶ Insert Paragraph - inserts a new paragraph depending on the current context. If current context is a paragraph context(one of the ancestors of the element at caret position is 'para') then a new paragraph will be inserted after the paragraph at caret. Otherwise a new paragraph is inserted at caret position.
- Insert Graphic - inserts a graphic object at the caret position. This is done by inserting either <*figure*> or <*inlinegraphic*> element depending on the current context. The following graphical formats are supported: GIF, JPG, JPEG, BMP, PNG and SVG.

- Insert Ordered List - inserts an ordered list with one list item.
- Insert Itemized List - inserts an itemized list with one list item.
- Insert Variable List - inserts a DocBook variable list with one list item.
- Insert List Item - inserts a new list item for in any of the above three list types.
- Insert Table - opens a dialog that allows you to configure the table to be inserted.

Figure 6.3. Insert Table Dialog



Insert Table

Model: ● CALS ○ HTML

☑ Title

Table Size: Rows 3 Columns 2

☑ Generate table header ☐ Generate table footer

Frame: none

The dialog allows the user to configure the number of rows and columns of the table, if the header and footer will be generated and how the table will be framed. Also, *CALS* or *HTML* table model can be selected.

Note

Unchecking the *Title* checkbox an 'informaltable' element will be inserted.

- Insert Row - inserts a new table row with empty cells. The action is available when the caret position is inside a table.
- Insert Column - inserts a new table column with empty cells after the current column. The action is available when the caret position is inside a table.
- Insert Cell - inserts a new empty cell depending on the current context. If the caret is positioned between two cells, a new one will be inserted at caret's position. If the caret is inside a cell, then the new one will be created after the current cell.
- Delete Column - deletes the table column where the caret is located.
- Delete Row - deletes the table row where the caret is located.
- Join Row Cells - joins the content of the selected cells. The operation is available if the selected cells are from the same row and they have the same row span. The action is also available when the selection is missing, but the caret is positioned between two cells.

- Join Cell Above - joins the content of cell from current caret position with that of the cell above it. Note that this action works only if both cells have the same column span.

- Join Cell Below - joins the content of cell from current caret position with that of the cell below it. Note that this action works only if both cells have the same column span.

- Split Cell To The Left - splits the cell from current caret position in two, inserting a new empty table cell to the left. Note that this action works only if the current cell spans over more than one column. The column span of the source cell will be decreased with one.

- Split Cell To The Right - splits the cell from current caret position in two, inserting a new empty table cell to the right. Note that this action works only if the current cell spans over more than one column. The column span of the source cell will be decreased with one.

- Split Cell Above - splits the cell from current caret position in two, inserting a new empty table cell above. Note that this action works only if the current cell spans over more than one row. The row span of the source cell will be decreased with one.

- Split Cell Below - splits the cell from current caret position in two, inserting a new empty table cell below. Note that this action works only if the current cell spans over more than one row. The row span of the source cell will be decreased with one.

Note

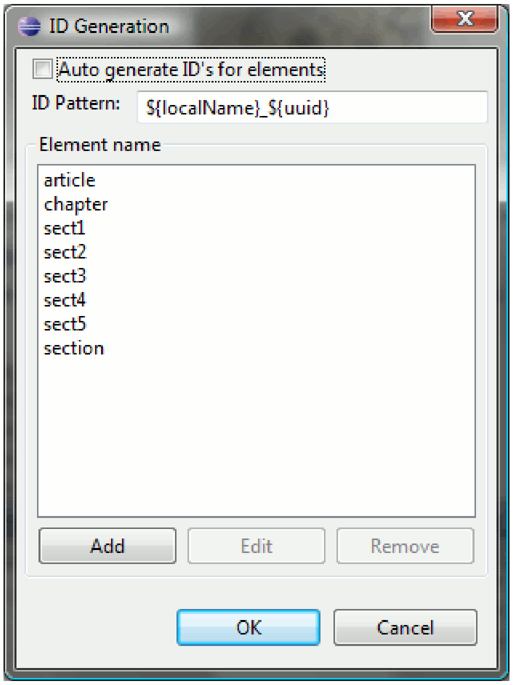
DocBook v4 supports only CALS table model. HTML table model is supported in DocBook v5.

Caution

Column specifications are required for table actions to work properly.

- Generate IDs -allows you to generate ID for the current selection or for the element at caret position if the element appears in **ID Generation** dialog.

Figure 6.4. ID Generation dialog



In this dialog you can specify the elements for which <oXygen/> should generate an ID. You can choose to automatically generate an ID for these elements by selecting Auto generate ID's for elements. You can choose a pattern for the generated ID using the field ID Pattern. If the element already has an ID, this ID is preserved.

All actions described above are available in the contextual menu, main menu (**Docbook4** submenu) and in the **Author custom actions** toolbar.

Templates

Default templates are available for DocBook 4. They are stored in **${frameworksDir}/docbook/templates/Docbook 4** folder and they can be used for easily creating a book or article with or without XInclude.

These templates are available when creating new documents from templates.

Docbook 4 - Article	New Docbook 4 Article
Docbook 4 - Article with XInclude	New Docbook 4 XInclude-aware Article
Docbook 4 -Book	New Docbook 4 Book
Docbook 4 -Book with XInclude	New Docbook 4 XInclude-aware Book

Catalogs

The default catalog is stored in **${frameworksDir}/docbook/catalog.xml**.

Transformation Scenarios

The following default transformation scenarios are available:

- **DocBook4 -> DocBook5 Conversion** - converts a DocBook4-compliant document to DocBook5;
- **DocBook HTML** - transforms a DocBook document into a HTML document;
- **DocBook PDF** - transforms a DocBook document into a PDF document.
- **DocBook HTML - chunk** - transforms a DocBook document in multiple HTML documents.

The DocBook V5 document type

Customization for DocBook V.5 is similar with that for DocBook V.4 with the following exceptions:

Association rules

A file is considered to be a DocBook V.5 document when the namespace is 'http://docbook.org/ns/docbook'.

Schema

DocBook v5 documents use a RelaxNG and Schematron schema located in *${frameworks}/docbook/5.0/rng/docbookxi.rng*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

Author extensions

DocBook 5 extensions contain all DocBook 4 extensions plus support for HTML table.

Templates

Default templates are available for DocBook 5. They are stored in **${frameworksDir}/docbook/templates/Docbook 5** folder and they can be used for easily creating a book or article with or without XInclude.

These templates are available when creating new documents from templates.

Docbook 5 - Article	New Docbook 5 Article
Docbook 5 - Article with XInclude	New Docbook 5 XInclude-aware Article
Docbook 5 -Book	New Docbook 5 Book
Docbook 5 -Book with XInclude	New Docbook 5 XInclude-aware Book

Catalogs

The default catalog is stored in **${frameworksDir}/docbook/5.0/catalog.xml**.

Transformation Scenarios

The following default transformation scenarios are available:

- **DocBook HTML** - transforms a DocBook document into HTML document;
- **DocBook PDF** - transforms a DocBook document into a PDF document.
- **DocBook HTML - chunk** - transforms a DocBook document in multiple HTML documents.

The DocBook Targetset document type

This document type is provided to edit or create a targetset file which is used to resolve cross references with olinks.

Association rules

A file is considered to be a DocBook Targetset document when the root name is 'targetset'.

Schema

DocBook Targetset documents use a DTD and schema located in *${frameworks}/docbook/xsl/common/targetdatabase.dtd*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

Author extensions

Templates

A default template is available for DocBook Targetset. It is stored in **${frameworksDir}/docbook/templates/Targetset** folder and can be used for easily creating a targetset.

This template is available when creating new documents from templates.

Docbook Targetset - Map	New Targetset Map

The DITA Topics document type

The Darwin Information Typing Architecture (DITA) is an XML-based architecture for authoring, producing, and delivering technical information. It divides content into small, self-contained topics that can be reused in different deliverables. The extensibility of DITA permits organizations to define specific information structures and still use standard tools to work with them.

Association rules

A file is considered to be a dita topic document when either of the following occurs:

- root element name is one of the following: *concept*, *task*, *reference*, *dita*, *topic*;
- public id of the document is one of the public id's for the elements above.
- the root element of the file has an attribute named "DITAArchVersion" attribute from the "http://dita.oasis-open.org/architecture/2005/" namespace. This enhanced case of matching is only applied when the *Enable DTD processing* option from the Document Type Detection option page is enabled.

Schema

The default schema used for DITA topic documents is located in *${frameworks}/dita/dtd/ditabase.dtd*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

Author extensions

The CSS file used for rendering DITA content is located in *${frameworks}/dita/css/dita.css*.

Specific actions for DITA topic documents are:

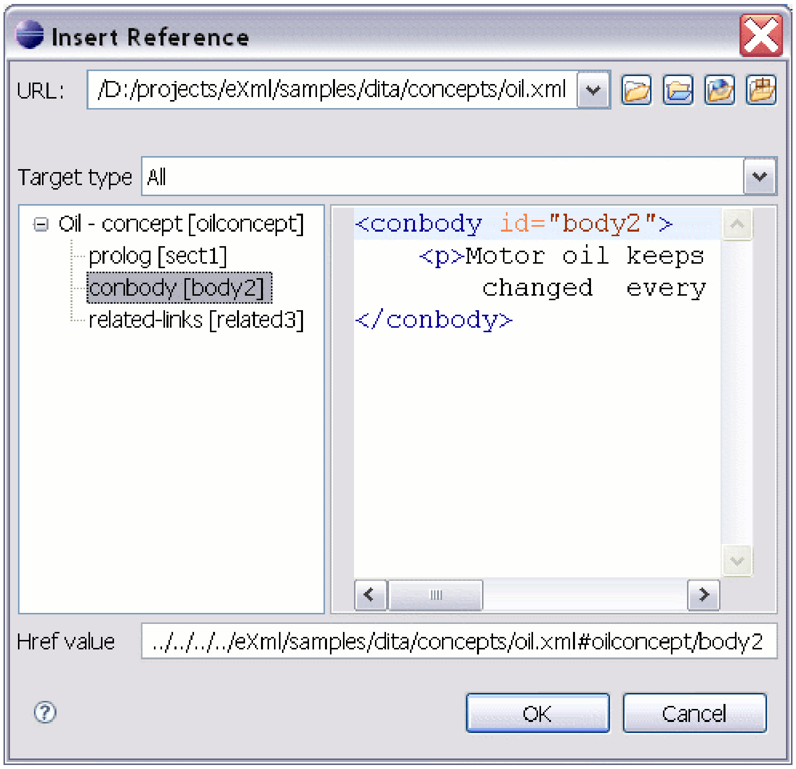
- **B** Bold - surrounds the selected text with *b* tag.
- *I* Italic - surrounds the selected text with *i* tag.
- U Underline - surrounds the selected text with *u* tag.

Note

For all of the above actions if there is no selection then a new specific tag will be inserted. These actions are available in any document context.

- Cross Reference - inserts an *xref* element with the value of attribute *format* set to "dita". The target of the *xref* is selected in a dialog which lists all the IDs available in a file selected by the user.

Figure 6.5. Insert a cross reference in a DITA document



URL: /D:/projects/eXml/samples/dita/concepts/oil.xml

Target type: All

```
⊟ Oil - concept [oilconcept]
    prolog [sect1]
    conbody [body2]
    related-links [related3]
```

```
<conbody id="body2">
    <p>Motor oil keeps
        changed  every
</conbody>
```

Href value: ../../../../eXml/samples/dita/concepts/oil.xml#oilconcept/body2

- File Reference - inserts an *xref* element with the value of attribute *format* set to "xml".
- Web Link - inserts an *xref* element with the value of attribute *format* set to "html", and *scope* set to "external".

- Related Link to Topic - inserts a *link* element inside a *related-links* parent.
- Related Link to File - inserts a *link* element with the *format* attribute set to "xml" inside a *related-links* parent.
- Related Link to Web Page - inserts a *link* element with the attribute *format* set to "html" and *scope* set to "external" inside a *related-links* parent.

  Note

  The actions for inserting references described above are grouped inside *link* toolbar actions group.

- § Insert Section/Step - inserts a new section/step in the document, depending on the current context. A new section will be inserted in either one of the following contexts:

  - section context, when the value of 'class' attribute of the current element or one of its ancestors contains 'topic' or 'section'.
  - topic's body context, when the value of 'class' attribute of the current element contains 'topic/body'.

  A new step will be inserted in either one of the following contexts:

  - task step context, when the value of 'class' attribute of the current element or one of its ancestors contains 'task/step'.
  - task steps context, when the value of 'class' attribute of the current element contains 'task/steps'.

- ¶ Insert Paragraph - inserts a new paragraph depending on the current context. If current context is a paragraph context (the value of 'class' attribute of the current element or one of its ancestors contains 'topic/p') then a new paragraph will be inserted after this paragraph. Otherwise a new paragraph is inserted at caret position.

- Insert Concept - inserts a new concept. Concepts provide background information that users must know before they can successfully work with a product or interface. This action is available in one of the following contexts:

  - concept context, one of the current element ancestors is a *concept*. In this case an empty *concept* will be inserted after the current *concept*.
  - concept or dita context, current element is a *concept* or *dita*. In this case an empty *concept* will be inserted at current caret position.
  - dita topic context, current element is a *topic* child of a *dita* element. In this case an empty *concept* will be inserted at current caret position.
  - dita topic context, one of the current element ancestors is a dita's *topic*. In this case an empty *concept* will be inserted after the first *topic* ancestor.

- Insert Task - inserts a new task. Tasks are the main building blocks for task-oriented user assistance. They generally provide step-by-step instructions that will enable a user to perform a task. This action is available in one of the following contexts:

  - task context, one of the current element ancestors is a *task*. In this case an empty *task* will be inserted after the last child of the first *concept*'s ancestor.
  - task context, the current element is a *task*. In this case an empty *task* will be inserted at current caret position.
  - topic context, the current element is a *dita*'s *topic*. An empty *task* will be inserted at current caret position.

  - topic context, one of the current element ancestors is a *dita*'s *topic*. An empty *task* will be inserted after the last child of the first ancestor that is a *topic*.

- Insert Reference - inserts a new reference in the document. A reference is a top-level container for a reference topic. This action is available in one of the following contexts:

  - reference context, one of the current element ancestors is a *reference*. In this case an empty *reference* will be inserted after the last child of the first ancestor that is a *reference*.

  - *reference* or *dita* context, the current element is either a *dita* or a *reference*. An empty *reference* will be inserted at caret position.

  - topic context, the current element is *topic* descendant of *dita* element. An empty *reference* will be inserted at caret position.

  - topic context, the current element is descendant of *dita* element and descendant of *topic* element. An empty *reference* will be inserted after the last child of the first ancestor that is a *topic*.

- Insert Graphic - inserts a graphic object at the caret position. This is done by inserting either `<figure>` or `<inlinemediaobject>` element depending on the current context.. The following graphical formats are supported: GIF, JPG, JPEG, BMP, PNG and SVG.

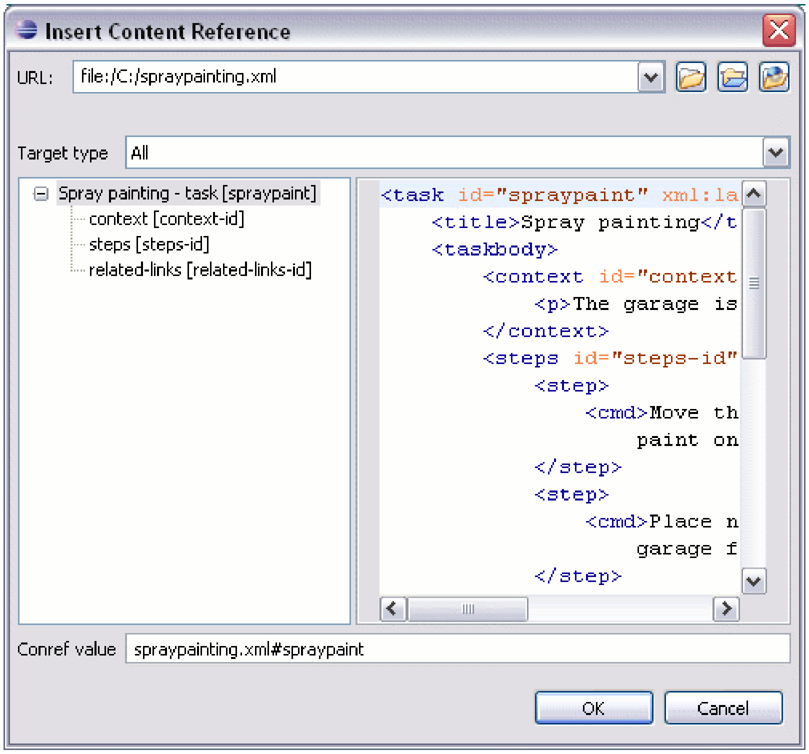
- Insert Content Reference - inserts a content reference at the caret position.

  The DITA conref attribute provides a mechanism for reuse of content fragments. The conref attribute stores a reference to another element and is processed to replace the referencing element with the referenced element. The element containing the content reference acts as a placeholder for the referenced element. The identifier for the referenced element must be either absolute or resolvable in the context of the referencing element. See here [http://docs.oasis-open.org/dita/v1.0/archspec/conref.html] for more details.

  <oXygen/> will display the referred content of a DITA conref if it can resolve it to a valid resource. If you use URI's instead of local paths and you have a catalog used in the DITA OT transformation you can add the catalog to <oXygen/> and if the URI's can be resolved the referred content will be displayed.

  A content reference is inserted with the action *Insert a DITA Content Reference* available on the toolbar *Author custom actions* and on the menu DITA → Insert.

Figure 6.6. Insert Content Reference Dialog



URL: file:/C:/spraypainting.xml

Target type All

- Spray painting - task [spraypaint]
  - context [context-id]
  - steps [steps-id]
  - related-links [related-links-id]

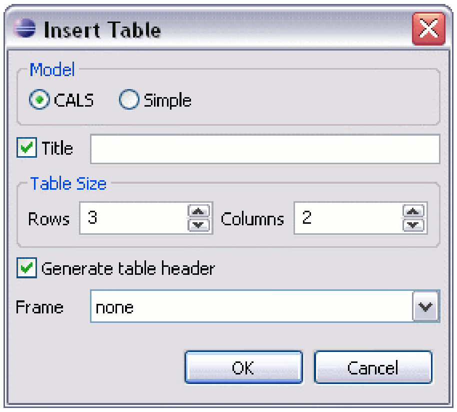
```
<task id="spraypaint" xml:la
    <title>Spray painting</t
    <taskbody>
        <context id="context
            <p>The garage is
        </context>
        <steps id="steps-id"
            <step>
                <cmd>Move th
                    paint on
            </step>
            <step>
                <cmd>Place n
                    garage f
            </step>
```

Conref value spraypainting.xml#spraypaint

In the URL chooser you can choose the file from which you want to reuse content. Depending on the *Target type* filter you will see a tree of elements which can be referred (which have id's). For each element the XML content is shown in the preview area. The *Conref value* is computed automatically for the selected tree element. After pressing OK an element with the same name as the target element and having the attribute *conref* with the value specified in the *Conref value* field will be inserted at caret position.

- Replace conref with content - Replace the content reference fragment at caret position with the referenced content. This action is useful when you want to make changes to the content but decide to keep the referenced fragment unchanged.
- Insert Ordered List - inserts an ordered list with one list item.
- Insert Unordered List - inserts an unordered list with one list item.
- Insert List Item - inserts a new list item for in any of the above two list types.
- Insert Table - opens a dialog that allows you to configure the table to be inserted.

Figure 6.7. Insert Table Dialog



Model

● CALS ○ Simple

☑ Title

Table Size

Rows 3 Columns 2

☑ Generate table header

Frame none

The dialog allows the user to configure the number of rows and columns of the table, if the header will be generated, if the title will be added and how the table will be framed.

- Insert Row - inserts a new table row with empty cells. The action is available when the caret position is inside a table.
- Insert Column - inserts a new table column with empty cells after the current column. The action is available when the caret position is inside a table.
- Insert Cell - inserts a new empty cell depending on the current context. If the caret is positioned between two cells, a new one will be inserted at caret's position. If the caret is inside a cell, then the new one will be created after the current cell.
- Delete Column - deletes the table column where the caret is located.
- Delete Row - deletes the table row where the caret is located.
- Join Row Cells - joins the content of the selected cells. The operation is available if the selected cells are from the same row and they have the same row span. The action is also available when the selection is missing, but the caret is positioned between two cells.
- Join Cell Above - joins the content of cell from current caret position with that of the cell above it. Note that this action works only if both cells have the same column span.
- Join Cell Below - joins the content of cell from current caret position with that of the cell below it. Note that this action works only if both cells have the same column span.
- Split Cell To The Left - splits the cell from current caret position in two, inserting a new empty table cell to the left. Note that this action works only if the current cell spans over more than one column. The column span of the source cell will be decreased with one.
- Split Cell To The Right - splits the cell from current caret position in two, inserting a new empty table cell to the right. Note that this action works only if the current cell spans over more than one column. The column span of the source cell will be decreased with one.

- Split Cell Above - splits the cell from current caret position in two, inserting a new empty table cell above. Note that this action works only if the current cell spans over more than one row. The row span of the source cell will be decreased with one.

- Split Cell Below - splits the cell from current caret position in two, inserting a new empty table cell below. Note that this action works only if the current cell spans over more than one row. The row span of the source cell will be decreased with one.

Note

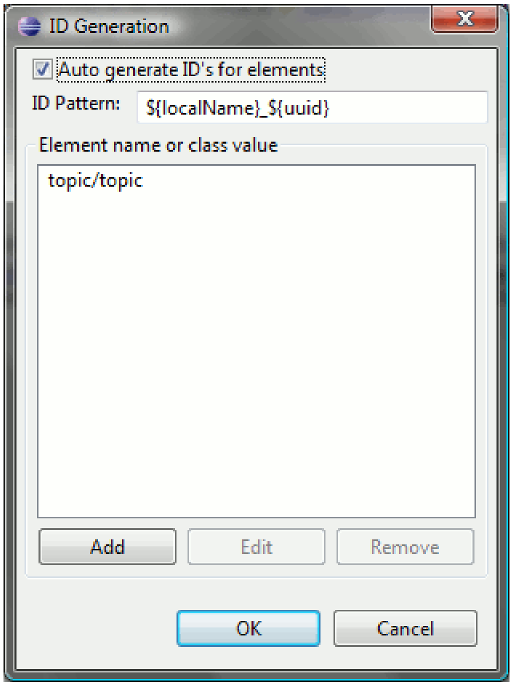
DITA supports CALS table model similar with DocBook document type in addition to the *simpletable* element specific for DITA.

Caution

Column specifications are required for table actions to work properly.

- Generate IDs - allows you to generate ID for the current selection or for the element at caret position if the element appears in **ID Generation** dialog.

Figure 6.8. ID Generation dialog



☑ Auto generate ID's for elements

ID Pattern: ${localName}_${uuid}

Element name or class value

topic/topic

In this dialog you can specify the elements for which <oXygen/> should generate an ID. You can choose to automatically generate an ID for these elements by selecting Auto generate ID's for elements. You can choose a pattern for the generated ID using the field ID Pattern. If the element already has an ID, this ID is preserved.

All actions described above are available in the contextual menu, main menu (**DITA** submenu) and in the **Author custom actions** toolbar.

Templates

Default templates available for DITA topics are stored in **${frameworksDir}/dita/templates/topic** folder. They can be used for easily creating a DITA's *concept*, *reference*, *task* or *topic*.

These templates are available when creating new documents from templates.

DITA - Composite	New DITA Composite
DITA - Concept	New DITA Concept
DITA - Glossentry	New DITA Glossentry
DITA - Reference	New DITA Reference
DITA - Task	New DITA Task
DITA - Topic	New DITA Topic

Catalogs

The default catalog is stored in **${frameworks}/dita/catalog-dita.xml**.

Transformation Scenarios

The following default transformation scenarios are available for DITA Topics:

- **DITA Topic to DocBook** - converts a DITA topic document into a DocBook document;
- **DITA Topic to HTML** - transforms a DITA topic document into HTML document;
- **DITA to PDF** - transforms a DITA document into a PDF document.

The DITA MAP document type

DITA maps are documents that collect and organize references to DITA topics to indicate the relationships among the topics. They can also serve as outlines or tables of contents for DITA deliverables and as build manifests for DITA projects.

Maps allow scalable reuse of content across multiple contexts. They can be used by information architects, writers, and publishers to plan, develop, and deliver content.

Association rules

A file is considered to be a dita map document when either of the following occurs:

- root element name is one of the following: *map*, *bookmap*;
- public id of the document is *-//OASIS//DTD DITA Map* or *-//OASIS//DTD DITA BookMap*.
- the root element of the file has an attribute named "class" which contains the value "map/map" and a "DITAArchVersion" attribute from the "http://dita.oasis-open.org/architecture/2005/" namespace. This enhanced case of matching is only applied when the *Enable DTD processing* option from the Document Type Detection option page is enabled.

Schema

The default schema used for DITA Map documents is located in *${frameworks}/dita/DITA-OT/dtd/map.dtd*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

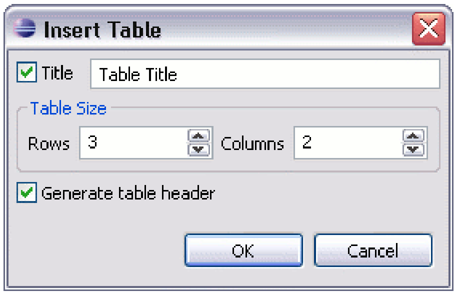
Author extensions

The CSS file used for rendering DocBook content is located in *${frameworks}/dita/css/dita.css*.

Specific actions for DITA Map documents are:

- Insert Topic Reference - inserts a reference to a topic. You can find more information about this action here.
- Insert Topic Heading - inserts a topic heading. You can find more information about this action here.
- Insert Topic Group - inserts a topic group. You can find more information about this action here.
- Insert Content Reference - inserts a content reference at the caret position. See more about this action here [215].
- Insert Table - opens a dialog that allows you to configure the relationship table to be inserted.

  Figure 6.9. Insert Relationship Table Dialog

  Insert Table

  ☑ Title: Table Title

  Table Size

  Rows: 3 Columns: 2

  ☑ Generate table header

  The dialog allows the user to configure the number of rows and columns of the relationship table, if the header will be generated and if the title will be added.

- Insert Row - inserts a new table row with empty cells. The action is available when the caret position is inside a table.
- Insert Column - inserts a new table column with empty cells after the current column. The action is available when the caret position is inside a table.
- Insert Cell - inserts a new empty cell depending on the current context. If the caret is positioned between two cells, a new one will be inserted at caret's position. If the caret is inside a cell, then the new one will be created after the current cell.
- Delete Column - deletes the table column where the caret is located.
- Delete Row - deletes the table row where the caret is located.

- Join Row Cells - joins the content of the selected cells. The operation is available if the selected cells are from the same row and they have the same row span. The action is also available when the selection is missing, but the caret is positioned between two cells.

- Join Cell Above - joins the content of cell from current caret position with that of the cell above it. Note that this action works only if both cells have the same column span.

- Join Cell Below - joins the content of cell from current caret position with that of the cell below it. Note that this action works only if both cells have the same column span.

- Split Cell To The Left - splits the cell from current caret position in two, inserting a new empty table cell to the left. Note that this action works only if the current cell spans over more than one column. The column span of the source cell will be decreased with one.

- Split Cell To The Right - splits the cell from current caret position in two, inserting a new empty table cell to the right. Note that this action works only if the current cell spans over more than one column. The column span of the source cell will be decreased with one.

All actions described above are available in the contextual menu, main menu (**DITA** submenu) and in the **Author custom actions** toolbar.

Templates

Default templates available for DITA Maps are stored in **${frameworksDir}/dita/templates/map** folder. They can be used for easily creating a DITA *map* and *bookmap* files.

These templates are available when creating new documents from templates.

DITA Map - Bookmap	New DITA Bookmap
DITA Map - Map	New DITA Map

Catalogs

The default catalog is stored in **${frameworks}/dita/catalog-dita.xml**.

Transformation Scenarios

To configure and run DITA-OT ANT transformations on a DITA Map you have to open it in the DITA Maps Manager view.

The XHTML document type

The Extensible HyperText Markup Language, or XHTML, is a markup language that has the same depth of expression as HTML, but also conforms to XML syntax.

Association rules

A file is considered to be a XHTML document when the root element name is a *html*.

Schema

The schema used for these documents is located in *${frameworks}/xhtml/dtd/xhtml1-strict.dtd*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

CSS

The default CSS options for the XHTML document type are set to merge the CSSs specified in the document with the CSSs defined in the XHTML document type.

Author extensions

The CSS file used for rendering XHTML content is located in *${frameworks}/xhtml/css/xhtml.css*.

Specific actions are:

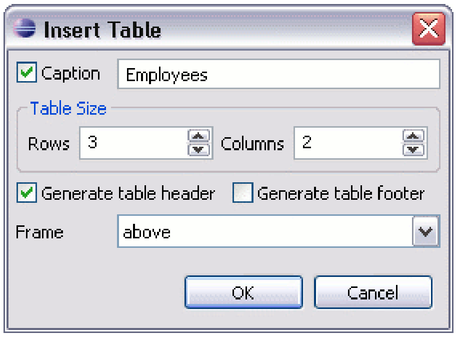
- **B** Bold - changes the style of the selected text to *bold* by surrounding it with *b* tag.
- *I* Italic - changes the style of the selected text to *italic* by surrounding it with *i* tag.
- U Underline - changes the style of the selected text to *underline* by surrounding it with *u* tag.

  ## Note

  For all of the above actions if there is no selection then a new specific tag will be inserted. These actions are available in any document context.

- H Headings - groups actions for inserting *h1*, *h2*, *h3*, *h4*, *h5*, *h6* elements.
- ¶ Insert Paragraph - inserts a new paragraph depending on the current context. If current context is a paragraph context (one of the ancestors of the element at caret position is *p*) then a new paragraph will be inserted after the paragraph at caret. Otherwise a new paragraph is inserted at caret position.
- Insert Graphic - inserts a graphic object at the caret position. This is done by inserting an *img* element regardless of the current context. The following graphical formats are supported: GIF, JPG, JPEG, BMP, PNG and SVG.
- Insert Ordered List - inserts an ordered list (*ol* element) with one list item (*li* child element).
- Insert Unordered List - inserts an unordered list (*ul* element) with one list item (*li* child element).
- Insert Definition List - inserts a definition list (*dl* element) with one list item (a *dt* child element and a *dd* child element).
- Insert List Item - inserts a new list item for in any of the above two list types.
- Insert Table - opens a dialog that allows you to configure the table to be inserted.

Figure 6.10. Insert Table Dialog



Insert Table

☑ Caption: Employees

Table Size

Rows: 3
Columns: 2

☑ Generate table header ☐ Generate table footer

Frame: above

The dialog allows the user to configure the number of rows and columns of the table, if the header and footer will be generated and how the table will be framed.

- Insert Row - inserts a new table row with empty cells. The action is available when the caret position is inside a table.
- Insert Column - inserts a new table column with empty cells after the current column. The action is available when the caret position is inside a table.
- Insert Cell - inserts a new empty cell depending on the current context. If the caret is positioned between two cells, a new one will be inserted at caret's position. If the caret is inside a cell, then the new one will be created after the current cell.
- Delete Column - deletes the table column where the caret is located.
- Delete Row - deletes the table row where the caret is located.
- Join Row Cells - joins the content of the selected cells. The operation is available if the selected cells are from the same row and they have the same row span. The action is also available when the selection is missing, but the caret is positioned between two cells.
- Join Cell Above - joins the content of cell from current caret position with that of the cell above it. Note that this action works only if both cells have the same column span.
- Join Cell Below - joins the content of cell from current caret position with that of the cell below it. Note that this action works only if both cells have the same column span.
- Split Cell To The Left - splits the cell from current caret position in two, inserting a new empty table cell to the left. Note that this action works only if the current cell spans over more than one column. The column span of the source cell will be decreased with one.
- Split Cell To The Right - splits the cell from current caret position in two, inserting a new empty table cell to the right. Note that this action works only if the current cell spans over more than one column. The column span of the source cell will be decreased with one.

- Split Cell Above - splits the cell from current caret position in two, inserting a new empty table cell above. Note that this action works only if the current cell spans over more than one row. The row span of the source cell will be decreased with one.

- Split Cell Below - splits the cell from current caret position in two, inserting a new empty table cell below. Note that this action works only if the current cell spans over more than one row. The row span of the source cell will be decreased with one.

All actions described above are available in the contextual menu, main menu (**XHTML** submenu) and in the **Author custom actions** toolbar.

Templates

Default templates are available for XHTML. They are stored in **${frameworksDir}/xhtml/templates** folder and they can be used for easily creating basic XHTML documents.

These templates are available when creating new documents from templates.

XHTML - 1.0 Strict	New Strict XHTML 1.0
XHTML - 1.0 Transitional	New Transitional XHTML 1.0
XHTML - 1.1 DTD Based	New DTD-based XHTML 1.1
XHTML - 1.1 DTD Based + MathML 2.0 + SVG 1.1	New XHTML 1.1 with MathML and SVG insertions.
XHTML - 1.1 Schema based	New XHTML 1.1 XML Schema based.

Catalogs

There are three default catalogs for XHTML document type: *${frameworks}/xhtml/dtd/xhtmlcatalog.xml*, *${frameworks}/xhtml11/dtd/xhtmlcatalog.xml* and *${frameworks}/xhtml11/schema/xhtmlcatalog.xml*.

Transformation Scenarios

The following default transformation scenarios are available for XHTML:

- **XHTML to DITA concept** - converts an XHTML document to a DITA concept document;
- **XHTML to DITA reference** - converts an XHTML document to a DITA reference document;
- **XHTML to DITA task** - converts an XHTML document to a DITA task document;
- **XHTML to DITA topic** - converts an XHTML document to a DITA topic document;

The TEI P4 document type

The Text Encoding Initiative (TEI) Guidelines is an international and interdisciplinary standard that enables libraries, museums, publishers, and individual scholars to represent a variety of literary and linguistic texts for online research, teaching, and preservation.

Association rules

A file is considered to be a TEI P4 document when either of the following occurs:

- the root's local name is **TEI.2**
- the document's public id is **-//TEI P4**

Schema

The DTD schema used for these documents is located in *${frameworks}/tei/tei2xml.dtd*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

Author extensions

The CSS file used for rendering TEI P4 content is located in *${frameworks}/tei/xml/tei/css/tei_oxygen.css*.

Specific actions are:

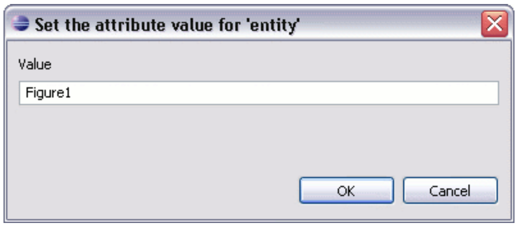
- **B** Bold - changes the style of the selected text to *bold* by surrounding it with *hi* tag and setting the *rend* attribute to *bold*.
- *I* Italic - changes the style of the selected text to *italic* by surrounding it with *hi* tag and setting the *rend* attribute to *italic*.
- U Underline - changes the style of the selected text to *underline* by surrounding it with *hi* tag and setting the *rend* attribute to *ul*.

  ### Note

  For all of the above actions if there is no selection then a new specific tag will be inserted. These actions are available in any document context.

- § Insert Section - inserts a new section/subsection, depending on the current context. For example if the current context is *div1* then a *div2* will be inserted and so on.
- ¶ Insert Paragraph - inserts a new paragraph depending on the current context. If current context is a paragraph context (one of the ancestors of the element at caret position is *p*) then a new paragraph will be inserted after the paragraph at caret. Otherwise a new paragraph is inserted at caret position.
- Insert Image - inserts a graphic object at the caret position. The following dialog is displayed allowing the user to specify the *entity* that refers the image itself:

Figure 6.11. Insert image entity dialog



Set the attribute value for 'entity'

Value

Figure1

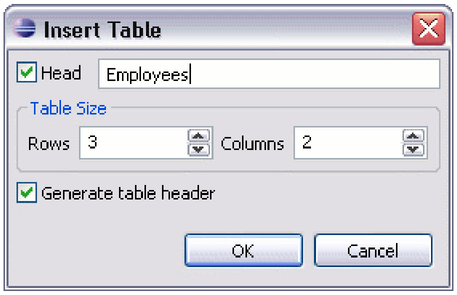
- Insert Ordered List - inserts an ordered list (*list* element with *type* attribute set to *ordered*) with one list item (*item* element).

- Insert Itemized List - inserts an unordered list (*list* element with *type* attribute set to *bulleted*) with one list item (*item* element).

- Insert List Item - inserts a new list item for in any of the above two list types.

- Insert Table - opens a dialog that allows you to configure the table to be inserted.

Figure 6.12. Insert Table Dialog



Insert Table

☑ Head Employees

Table Size

Rows 3 Columns 2

☑ Generate table header

The dialog allows the user to configure the number of rows and columns of the table and if the header will be generated.

- Insert Row - inserts a new table row with empty cells. The action is available when the caret position is inside a table.

- Insert Column - inserts a new table column with empty cells after the current column. The action is available when the caret position is inside a table.

- Insert Cell - inserts a new empty cell depending on the current context. If the caret is positioned between two cells, a new one will be inserted at caret's position. If the caret is inside a cell, then the new one will be created after the current cell.

- Delete Column - deletes the table column where the caret is located.

- Delete Row - deletes the table row where the caret is located.

- Join Row Cells - joins the content of the selected cells. The operation is available if the selected cells are from the same row and they have the same row span. The action is also available when the selection is missing, but the caret is positioned between two cells.

- Join Cell Above - joins the content of cell from current caret position with that of the cell above it. Note that this action works only if both cells have the same column span.

- Join Cell Below - joins the content of cell from current caret position with that of the cell below it. Note that this action works only if both cells have the same column span.

- Split Cell To The Left - splits the cell from current caret position in two, inserting a new empty table cell to the left. Note that this action works only if the current cell spans over more than one column. The column span of the source cell will be decreased with one.

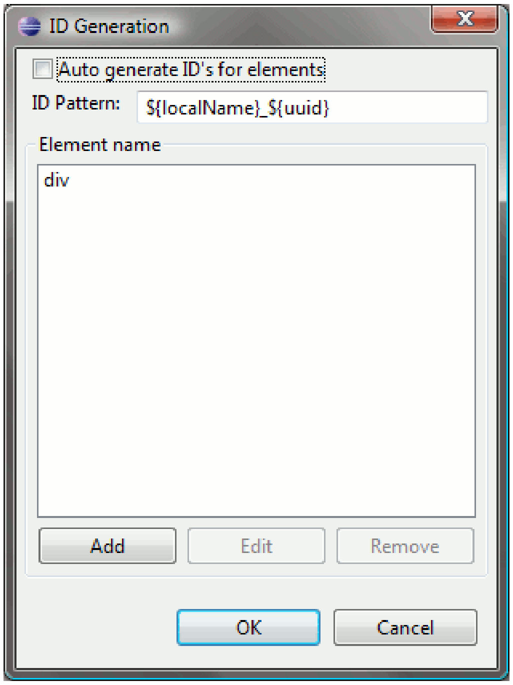
- Split Cell To The Right - splits the cell from current caret position in two, inserting a new empty table cell to the right. Note that this action works only if the current cell spans over more than one column. The column span of the source cell will be decreased with one.

- Split Cell Above - splits the cell from current caret position in two, inserting a new empty table cell above. Note that this action works only if the current cell spans over more than one row. The row span of the source cell will be decreased with one.

- Split Cell Below - splits the cell from current caret position in two, inserting a new empty table cell below. Note that this action works only if the current cell spans over more than one row. The row span of the source cell will be decreased with one.

- Generate IDs - allows you to generate ID for the current selection or for the element at caret position if the element appears in **ID Generation** dialog.

Figure 6.13. ID generation dialog



ID Generation

□ Auto generate ID's for elements

ID Pattern: ${localName}_${uuid}

Element name

div

In this dialog you can specify the elements for which <oXygen/> should generate an ID. You can choose to automatically generate an ID for these elements by selecting Auto generate ID's for elements. You can choose a pattern for the generated ID using the field ID Pattern. If the element already has an ID, this ID is preserved.

All actions described above are available in the contextual menu, main menu (**TEI P4** submenu) and in the **Author custom actions** toolbar.

Templates

Default templates are available for XHTML. They are stored in **${frameworksDir}/tei/templates/TEI P4** folder and they can be used for easily creating basic TEI P4 documents.

These templates are available when creating new documents from templates.

TEI P4 - Lite	New TEI P4 Lite.
TEI P4 - New Document	New TEI P4 standard document.

Catalogs

There are two default catalogs for TEI P4 document type: *${frameworks}/tei/xml/teip4/schema/dtd/catalog.xml* and *${frameworks}/tei/xml/teip4/custom/schema/dtd/catalog.xml*.

Transformation Scenarios

The following default transformations are available:

- **TEI HTML** - transforms a TEI document into a HTML document;
- **TEI P4 -> TEI P5 Conversion** - convert a TEI P4 document into a TEI P5 document;

- **TEI PDF** - transforms a TEI document into a PDF document.

The TEI P5 document type

Customization for TEI P5 is similar with that for TEI P4 with the following exceptions:

Association rules

A file is considered to be a TEI P5 document when the namespace is *http://www.tei-c.org/ns/1.0*.

Schema

The RNG schema used for these documents is located in *${frameworks}/tei/xml/tei/custom/schema/relaxng/tei_allPlus.rng*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

Author extensions

The CSS file used for rendering TEI P5 content and custom actions are the same with those configured for TEI P4.

Templates

Default templates are available for TEI P5. They are stored in **${frameworksDir}/tei/templates/TEI P5** folder and they can be used for easily creating basic TEI P5 documents.

These templates are available when creating new documents from templates.

TEI P5 - All	New TEI P5 All.
TEI P5 - Bare	New TEI P5 Bare.
TEI P5 - Lite	New TEI P5 Lite.
TEI P5 - Math	New TEI P5 Math.
TEI P5 - Speech	New TEI P5 Speech.
TEI P5 - SVG	New TEI P5 with SVG extensions.
TEI P5 - XInclude	New TEI P5 XInclude aware.

Catalogs

XML catalogs used for TEI P4 are used also for TEI P5.

Transformation Scenarios

The following default transformations are available:

- **TEI P5(experimental) HTML** - transforms a TEI document into a HTML document;
- **TEI P5(experimental) PDF** - transforms a TEI document into a PDF document.

The MathML document type

Mathematical Markup Language (MathML) is an application of XML for describing mathematical notations and capturing both its structure and content. It aims at integrating mathematical formulae into World Wide Web documents.

<oXygen/> offers support for editing and validating MathML 2.0 documents.

Association rules

A file is considered to be a MathML document when the root element name is a *math* or it's namespace is `http://www.w3.org/1998/Math/MathML`.

Schema

The schema used for these documents is located in *${frameworks}/mathml2/dtd/mathml2.dtd*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

Templates

Default templates are available for MathML. They are stored in the **${frameworksDir}/mathml2/templates** folder.

These templates are available when creating new documents from templates.

MathML - Equation   Simple MathML template file.

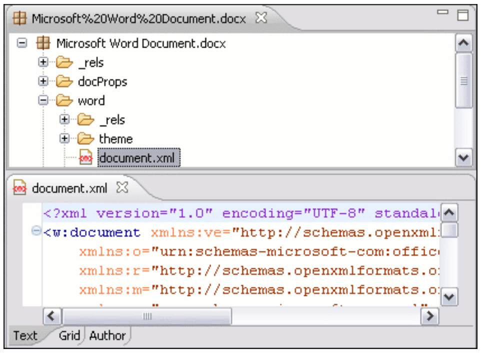
The Microsoft Office OOXML document type

Office Open XML (also referred to as OOXML or OpenXML) is a free and open Ecma [http://www.ecma-international.org/publications/standards/Ecma-376.htm] international standard document format, and a proposed ISO/IEC standard for representing spreadsheets, charts, presentations and word processing documents.

OOXML uses a file package conforming to the Open Packaging Convention. This format uses the ZIP file format and contains the individual files that form the basis of the document. In addition to Office markup, the package can also include embedded files such as images, videos, or other documents.

<oXygen/> offers support for editing, transforming and validating documents composing the OOXML package directly through the archive support.

Figure 6.14. Editing OOXML packages in <oXygen/>



```
<?xml version="1.0" encoding="UTF-8" standal
<w:document xmlns:ve="http://schemas.openxml
    xmlns:o="urn:schemas-microsoft-com:offic
    xmlns:r="http://schemas.openxmlformats.o
    xmlns:m="http://schemas.openxmlformats.o
```

Association rules

A file is considered to be an OOXML document when it has one of the following namespaces:

- `http://schemas.openxmlformats.org/wordprocessingml/2006/main`
- `http://schemas.openxmlformats.org/package/2006/content-types`
- `http://schemas.openxmlformats.org/drawingml/2006/main`
- `http://schemas.openxmlformats.org/package/2006/metadata/core-properties`
- `http://schemas.openxmlformats.org/package/2006/relationships`
- `http://schemas.openxmlformats.org/presentationml/2006/main`
- `http://schemas.openxmlformats.org/officeDocument/2006/custom-properties`
- `http://schemas.openxmlformats.org/officeDocument/2006/extended-properties`
- `http://schemas.openxmlformats.org/spreadsheetml/2006/main`
- `http://schemas.openxmlformats.org/drawingml/2006/chart`

Schema

The NVDL schema used for these documents is located in *${frameworks}/ooxml/schemas/main.nvdl*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory. The schema can be easily customized to allow user defined extension schemas for use in the OOXML files. See the Markup Compatibility and Extensibility [http://www.ecma-international.org/news/TC45_current_work/Office%20Open%20XML%20Part%205%20-%20Markup%20Compatibility%20and%20Extensibility.pdf] Ecma PDF document for more details.

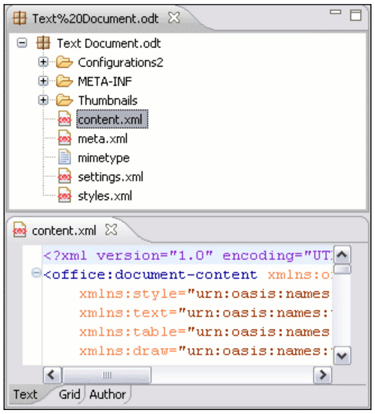
The Open Office ODF document type

The OpenDocument format (ODF) is a free and open file format for electronic office documents, such as spreadsheets, charts, presentations and word processing documents. The standard [http://www.oasis-open.org/committees/office/] was developed by the Open Office XML technical committee of the Organization for the Advancement of Structured Information Standards (OASIS) consortium and based on the XML format originally created and implemented by the OpenOffice.org office suite.

A basic OpenDocument file consists of an XML document that has <document> as its root element. OpenDocument files can also take the format of a ZIP compressed archive containing a number of files and directories; these can contain binary content and benefit from ZIP's lossless compression to reduce file size. OpenDocument benefits from separation of concerns by separating the content, styles, metadata and application settings into four separate XML files.

<oXygen/> offers support for editing, manipulating and validating documents composing the ODF package directly through the archive support.

Figure 6.15. Editing ODF packages in <oXygen/>



```
<?xml version="1.0" encoding="UT
<office:document-content xmlns:o
    xmlns:style="urn:oasis:names
    xmlns:text="urn:oasis:names:
    xmlns:table="urn:oasis:names
    xmlns:draw="urn:oasis:names:
```

Association rules

A file is considered to be an ODF document when it has the following namespace: `urn:oasis:names:tc:opendocument:xmlns:office:1.0`

Schema

The RelaxNG schema used for these documents is located in *${frameworks}/odf/schemas/OpenDocument-schema-v1.1.rng*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

The OASIS XML Catalog document type

The OASIS [http://www.oasis-open.org/committees/entity/spec-2001-08-06.html] XML catalog is a document describing a mapping between external entity references or URI's and locally-cached equivalents. You can read more about using catalogs in <oXygen/> here.

Association rules

A file is considered to be an XML Catalog document when it has the following namespace: `urn:oasis:names:tc:entity:xmlns:xml:catalog` or when its root element name is `catalog`.

Schema

The OASIS 1.1 XSD schema used for these documents is located in *${frameworks}/xml/catalog1.1.xsd*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

The XML Schema document type

This document type is used to associated CSS stylesheets to an XML Schema so it can be visualized in the Author page.

Association rules

A file is considered to be an XML Schema document when the root name is 'schema' and namespace is 'http://www.w3.org/2001/XMLSchema'.

Author extensions

The following CSS alternatives are proposed for visualizing XML Schemas in the Author page.

${frameworks}/xmlschema/schema-main.css	Documentation - representation of XML Schema optimized for editing and viewing documentation.
${frameworks}/xmlschema/schemaISOSchematron.css	XMLSchema+ISOSchematron - representation of XML Schema with embedded ISO Schematron rules.
${frameworks}/xmlschema/schemaSchematron.css	XMLSchema+Schematron - representation of XML Schema with embedded Schematron rules.
${frameworks}/xmlschema/default.css	XMLSchema+Schematron - representation of XML Schema for general editing.

The RelaxNG document type

This document type is used to associated CSS stylesheets to an RelaxNG file so it can be visualized in the Author page.

Association rules

A file is considered to be an RelaxNG document when the namespace is 'http://relaxng.org/ns/structure/1.0'.

Author extensions

The following CSS alternatives are proposed for visualizing RelaxNG schemas in the Author page.

${frameworks}/relaxng/relaxng-main.css	Relax NG - representation of Relax NG optimized for editing in the Author mode.
${frameworks}/relaxng/relaxngISOSchematron.css	RelaxNG+ISOSchematron - representation of RelaxNG with embedded ISO Schematron rules.
${frameworks}/relaxng/relaxngSchematron.css	RelaxNG+Schematron - representation of RelaxNG with embedded Schematron rules.

The NVDL document type

This document type is used to associated CSS stylesheets to a NVDL file so it can be visualized in the Author page.

Association rules

A file is considered to be a NVDL document when the namespace is 'http://purl.oclc.org/dsdl/nvdl/ns/structure/1.0'.

Author extensions

The following CSS is proposed for visualizing NVDL schemas in the Author page.

${frameworks}/nvdl/nvdl.css	Representation of Relax NG optimized for editing in the Author mode.

The Schematron document type

This document type is used to associated CSS stylesheets to a Schematron file so it can be visualized in the Author page.

Association rules

A file is considered to be a Schematron document when the namespace is 'http://purl.oclc.org/dsdl/schematron'.

Author extensions

The following CSS is proposed for visualizing Schematron schemas in the Author page.

${frameworks}/schematron/iso-schematron.css	Representation of Schematron optimized for editing in the Author mode.

The Schematron 1.5 document type

This document type is used to associated CSS stylesheets to a Schematron 1.5 file so it can be visualized in the Author page.

Association rules

A file is considered to be a Schematron 1.5 document when the namespace is 'http://www.ascc.net/xml/schematron'.

Author extensions

The following CSS is proposed for visualizing Schematron 1.5 schemas in the Author page.

${frameworks}/schematron/schematron15.css	Representation of Schematron 1.5 optimized for editing in the Author mode.

The XSLT document type

This document type is used to associated CSS stylesheets to an XSLT stylesheet file so it can be visualized in the Author page.

Association rules

A file is considered to be a XSLT document when the namespace is 'http://www.w3.org/1999/XSL/Transform'.

Author extensions

The following CSS is proposed for visualizing XSLT stylesheets in the Author page.

${frameworks}/xslt/xslt.css	Representation of XSLT optimized for editing in the Author mode.

The XMLSpec document type

XMLSpec is a markup language for W3C specifications and other technical reports.

Association rules

A file is considered to be an XMLSpec document when the root name is 'spec'.

Schema

XMLSpec documents use a RelaxNG schema located in *${frameworks}/xmlspec/schema/xmlspec.rng*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

Author extensions

Templates

Default templates are available for XMLSpec. They are stored in **${frameworksDir}/xmlspec/templates** folder and they can be used for easily creating an XMLSpec.

These templates are available when creating new documents from templates.

XMLSpec - New Document	New XMLSpec document

Catalogs

The default catalog is stored in **${frameworks}/xmlspec/catalog.xml**.

Transformation Scenarios

The following default transformation scenarios are available:

- **XMLSpec PDF** - transforms an XMLSpec document into PDF document;
- **XMLSpec HTML** - transforms an XMLSpec document into HTML document;
- **XMLSpec HTML Diff** - produces "color-coded" HTML from *diff* markup;
- **XMLSpec HTML Slices** - produces "chunked" HTML specifications;

The FO document type

FO describes the formatting of XML data for output to screen, paper or other media.

Association rules

A file is considered to be an FO document when the it's namespace is `http://www.w3.org/1999/XSL/Format`.

Schema

FO documents use a XML Schema located in *${frameworks}/fo/xsd/fo.xsd*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

Author extensions

Transformation Scenarios

The following default transformation scenarios are available:

- **FO PDF** - transforms an FO document into PDF document;

The EAD document type

EAD Document Type Definition (DTD) is a standard for encoding archival finding aids using Extensible Markup Language (XML). The standard is maintained in the Network Development and MARC Standards Office of the Library of Congress (LC) in partnership with the Society of American Archivists.

Association rules

A file is considered to be a FO document when the it's namespace is `urn:isbn:1-931666-22-9` or it's public ID is `//DTD ead.dtd (Encoded Archival Description (EAD) Version 2002)//EN`.

Schema

EAD documents use a Relax NG Schema located in *${frameworks}/ead/rng/ead.rng*, where *${frameworks}* is a subdirectory of the <oXygen/> install directory.

Author extensions

Templates

Default templates are available for EAD. They are stored in **${frameworksDir}/ead/templates** folder and they can be used for easily creating an EAD document.

These templates are available when creating new documents from templates.

EAD - NWDA Template 2008-04-08	New EAD document

Catalogs

The default catalog is stored in **${frameworks}/ead/catalog.xml**.

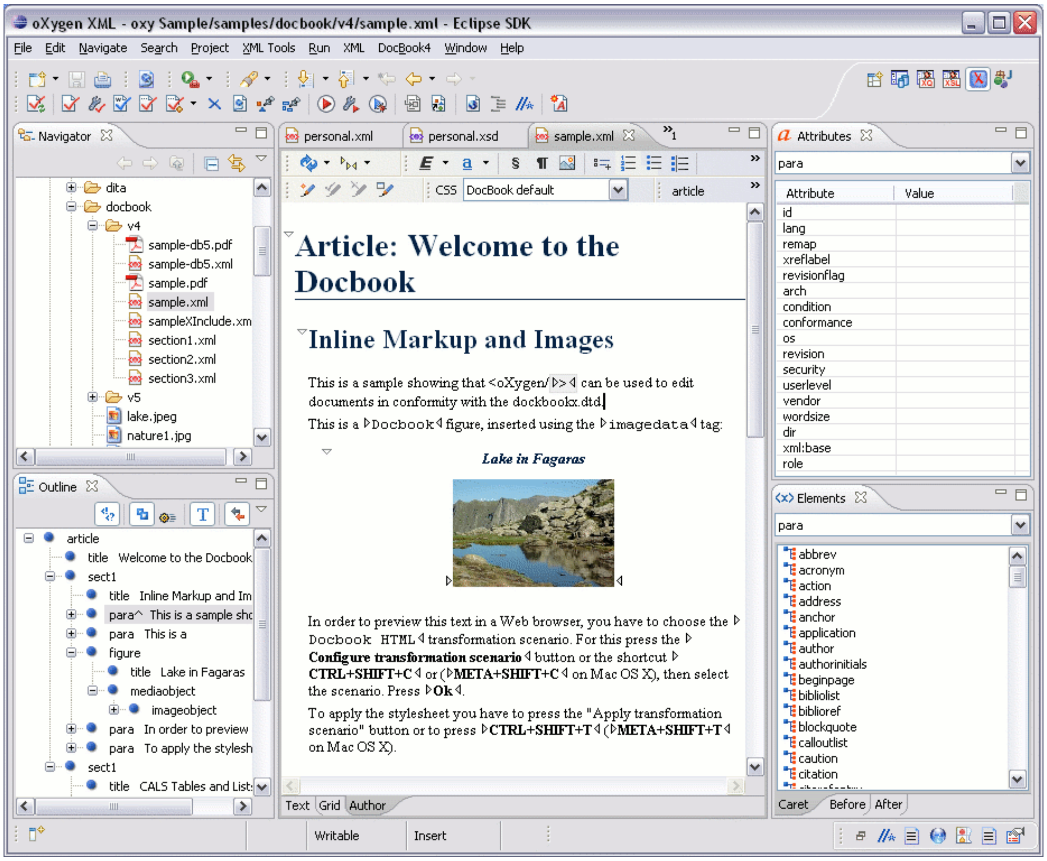
Chapter 7. <oXygen/> XML Editor Developer Guide

Introduction

Starting with version 9, <oXygen/> adds extensive support for customization.

The Author mode from <oXygen/> was designed for bridging the gap between the XML source editing and a friendly user interface. The main achievement is the fact that the Author combines the power of the source editing and the intuitive interface of a text editor.

Figure 7.1. oXygen Author Editor



Article: Welcome to the Docbook

Inline Markup and Images

This is a sample showing that <oXygen/▷>◁ can be used to edit documents in conformity with the dockbookx.dtd.

This is a ▷Docbook◁ figure, inserted using the ▷imagedata◁ tag:

Lake in Fagaras

In order to preview this text in a Web browser, you have to choose the ▷Docbook HTML◁ transformation scenario. For this press the ▷**Configure transformation scenario**◁ button or the shortcut ▷**CTRL+SHIFT+C**◁ or (▷**META+SHIFT+C**◁ on Mac OS X), then select the scenario. Press ▷**Ok**◁.

To apply the stylesheet you have to press the "Apply transformation scenario" button or to press ▷**CTRL+SHIFT+T**◁ (▷**META+SHIFT+T**◁ on Mac OS X).

Although <oXygen/> comes with already configured frameworks for DocBook, DITA, TEI, XHTML, you might need to create a customization of the editor to handle other types of documents. For instance in the case you have a collection of XML document types used to define the structure of the documents that are used in your organisation and you want them visually edited by people who are not experienced in using XML.

There are several ways to customize the editor:

1. Create a CSS file defining styles for the XML elements the user will work with, and create XML files that refer the CSS through an `xml-stylesheet` processing instruction.

2. Fully configure a document type association. This involves putting together the CSS files, the XML schemes, actions, menus, etc, bundling them and distributing an archive. The CSS and the GUI elements are settings of the <oXygen/> Author. The other settings like the templates, catalogs, transformation scenarios are general settings and are enabled whenever the association is active, no matter the editing mode (Text, Grid or Author).

We will discuss both approaches in the following sections.

Simple Customization Tutorial

XML Schema

Let's consider the following XML Schema, `test_report.xsd` defining a report with results of a testing session. The report consists of a title, few lines describing the test suite that was run and a list of test results, each with a name and a boolean value indicating if the test passed or failed.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="report">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="title"/>
                <xs:element ref="description"/>
                <xs:element ref="results"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="description">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element name="line">
                    <xs:complexType mixed="true">
                        <xs:sequence minOccurs="0"
                            maxOccurs="unbounded">
                            <xs:element name="important"
                              type="xs:string"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="results">
        <xs:complexType>
            <xs:sequence maxOccurs="unbounded">
                <xs:element name="entry">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="test_name"
                              type="xs:string"/>
                            <xs:element name="passed"
```

```
                               type="xs:boolean"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Our use-case is that several users are testing a system and must send report results to a content management system. The Author customization should provide a visual editor for this kind of documents.

Writing the CSS

We have to define a set of rules describing how the XML document is to be rendered into the <oXygen/> Author. This is done using Cascading Style Sheets or CSS on short. CSS is a language used to describe how an HTML or XML document should be formatted by a browser. CSS is widely used in the majority of websites.

Note

For more information regarding CSS, please read the specification http://www.w3.org/Style/CSS/. A tutorial is available here : http://www.w3schools.com/css/css_intro.asp

The elements from an XML document are displayed in the layout as a series of boxes. Some of the boxes contain text and may flow one after the other, from left to right. These are called in-line boxes. There are also other type of boxes that flow one below the other, like paragraphs. These are called block boxes.

For example consider the way a traditional text editor arranges the text. A paragraph is a block, because it contains a vertical list of lines. The lines are also blocks. But any block that contains inline boxes is arranging its children in a horizontal flow. That is why the paragraph lines are also blocks, but the traditional "bold" and "italic" sections are represented as inline boxes.

The CSS allows us to specify that some elements are displayed as tables. In CSS a table is a complex structure and consists of rows and cells. The "table" element must have children that have "table-row" style. Similarly, the "row" elements must contain elements with "table-cell" style.

To make it easy to understand, the following section describes the way each element from the above schema is formatted using a CSS file. Please note that this is just one from an infinite number of possibilities of formatting the content.

report

This element is the root element of the report document. It should be rendered as a box that contains all other elements. To achieve this we set its display type to **block**. Additionally we are setting some margins for it. The CSS rule that matches this element is:

```
report{
    display:block;
    margin:1em;
}
```

title

The title of the report. Usually titles have a larger font. We should use also the **block** display - the next elements will be placed below it, and change its font to double the size of the normal text.

```
title {
    display:block;
    font-size:2em;
}
```

description

This element contains several lines of text describing the report. The lines of text are displayed one below the other, so the description will have the same **block** display. To make it standout we are changing its background.

```
description {
    display:block;
    background-color:#EEEEFF;
    color:black;
}
```

line

A line of text in the description. We do not define a specific aspect for it, just indicating that the display should be **block**.

```
line {
    display:block;
}
```

important

The `important` element defines important text from the description. Because it can be mixed with text, its display property must be set to **inline**. To make it easier to spot, we will emphasize its text.

```
important {
    display:inline;
    font-weight:bold;
}
```

results

The `results` element shows the list of test_names and the result for each one. To make it easier to read, we choose to display it as a **table** with a green border and margins.

```
results{
    display:table;
    margin:2em;
    border:1px solid green;
}
```

entry

An item in the results element. Because we chose the results to be a table, the entry is the row in the table. Thus, the display is **table-row**.

```
entry {
    display:table-row;
}
```

test_name, passed

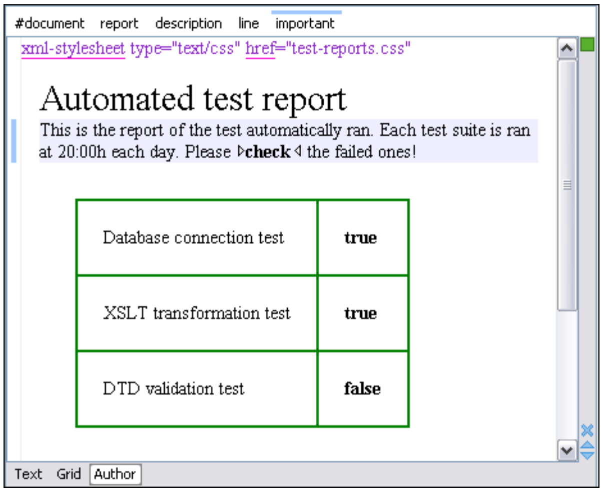
The name of the individual test, and its result. They are cells in the results table with display set to **table-cell**. Padding and a border are added to emphasize the table grid.

```
test_name, passed{
    display:table-cell;
    border:1px solid green;
    padding:20px;
}

passed{
    font-weight:bold;
}
```

The full content of the CSS file `test_report.css` is:

```
report {
    display:block;
    margin:1em;
}

description {
    display:block;
    background-color:#EEEEFF;
    color:black;
}

line {
    display:block;
}

important {
    display:inline;
    font-weight:bold;
}

title {
    display:block;
    font-size:2em;
}

results{
    display:table;
    margin:2em;
    border:1px solid green;
}

entry {
    display:table-row;
}

test_name, passed{
    display:table-cell;
    border:1px solid green;
    padding:20px;
}

passed{
    font-weight:bold;
}
```

Figure 7.2. A report opened in the Author



xml-stylesheet type="text/css" href="test-reports.css"

Automated test report

This is the report of the test automatically ran. Each test suite is ran at 20:00h each day. Please ▷**check**◁ the failed ones!

Database connection test	**true**
XSLT transformation test	**true**
DTD validation test	**false**

The XML Instance Template

Now we have the XML Schema and the CSS file. Based on these files, the <oXygen/> Author can help the content author in loading, editing and validating the test reports. We have to create an XML file template, a kind of skeleton, that the users can use as a starting point for creating new test reports.

The template must be generic enough and refer the XML Schema file and the CSS stylesheet. This is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="test_report.css"?>
<report xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:noNamespaceSchemaLocation="test_report.xsd">
    <title>Test report title</title>
    <description>
        <line>This is the report
                  <important>description</important>.</line>
    </description>
    <results>
        <entry>
            <test_name>Sample test1</test_name>
            <passed>true</passed>
        </entry>
        <entry>
            <test_name>Sample test2</test_name>
            <passed>true</passed>
        </entry>
    </results>
</report>
```

The processing instruction `xml-stylesheet` associates the CSS stylesheet to the XML file. The href pseudo attribute contains the URI reference to the stylesheet file. In our case the CSS is in the same directory as the XML file.

The next step is to place the XSD file and the CSS file on a web server and modify the template to use the HTTP URLs, like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css"
        href="http://www.mysite.com/reports/test_report.css"?>
<report xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation=
        "http://www.mysite.com/reports/test_report.xsd">
    <title>Test report title</title>
    <description>
.......
```

The alternative is to create an archive containing the `test_report.xml`, `test_report.css` and `test_report.xsd` and send it to the content authors.

Advanced Customization Tutorial - Document Type Associations

<oXygen/> Author is highly customizable. Practically you can associate an entire class of documents (grouped logically by some common features like namespace, root element name or filename) to a bundle consisting of a CSS stylesheets, validation schemas, catalog files, templates for new files, transformation scenarios and even custom actions. This is called a **Document Type Association**.

Creating the Basic Association

In this section we will create a **Document Type Association** for a set of documents. As an example we will create a light documentation framework, similar to DocBook and create a complete customization of the Author editor.

You can find the complete files that were used in this tutorial in the Example Files Listings.

First step. XML Schema.

Our documentation framework will be very simple. The documents will be either `articles` or `books`, both composed of `sections`. The `sections` may contain `titles`, `paragraphs`, `figures`, `tables` and other `sections`. To complete the picture, each section will include a `def` element from another namespace.

The first schema file:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.oxygenxml.com/sample/documentation"
    xmlns:doc="http://www.oxygenxml.com/sample/documentation"
    xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts"
    elementFormDefault="qualified">
    <xs:import namespace=
    "http://www.oxygenxml.com/sample/documentation/abstracts"
     schemaLocation=
    "abs.xsd"/>
```

The namespace of our documents will be `http://www.oxygenxml.com/sample/documentation`. The namespace of the `def` element is `http://www.oxygenxml.com/sample/documentation/abstracts`.

Now let's define the structure of the sections. They all start with a title, then have the optional `def` element then either a sequence of other sections, or a mixture of paragraphs, images and tables.

```
<xs:element name="book" type="doc:sectionType"/>
<xs:element name="article" type="doc:sectionType"/>
<xs:element name="section" type="doc:sectionType"/>

<xs:complexType name="sectionType">
    <xs:sequence>
        <xs:element name="title" type="xs:string"/>
        <xs:element ref="abs:def" minOccurs="0"/>
        <xs:choice>
            <xs:sequence>
                <xs:element ref="doc:section" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:choice maxOccurs="unbounded">
                <xs:element ref="doc:para"/>
                <xs:element ref="doc:image"/>
                <xs:element ref="doc:table"/>
            </xs:choice>
        </xs:choice>
    </xs:sequence>
</xs:complexType>
```

The paragraph contains text and other styling markup, such as bold (`b`) and italic (`i`) elements.

```
<xs:element name="para" type="doc:paragraphType"/>

<xs:complexType name="paragraphType" mixed="true">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="b"/>
        <xs:element name="i"/>
    </xs:choice>
</xs:complexType>
```

The `image` element has an attribute with a reference to the file containing image data.

```
<xs:element name="image">
    <xs:complexType>
        <xs:attribute name="href" type="xs:anyURI" use="required"/>
    </xs:complexType>
</xs:element>
```

The `table` contains a header row and then a sequence of rows (`tr` elements) each of them containing the cells. Each cell has the same content as the paragraphs.

```
 <xs:element name="table">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="header">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="td" maxOccurs="unbounded"
                            type="doc:paragraphType"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="tr" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="td" type="doc:tdType"
                             maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>


<xs:complexType name="tdType">
    <xs:complexContent>
        <xs:extension base="doc:paragraphType">
            <xs:attribute name="row_span" type="xs:integer"/>
            <xs:attribute name="column_span" type="xs:integer"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
```

The `def` element is defined as a text only element in the imported schema `abs.xsd`:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace=
     "http://www.oxygenxml.com/sample/documentation/abstracts">
    <xs:element name="def" type="xs:string"/>
</xs:schema>
```

Now that we defined our XML data structure, let's start styling it...

Second step. The CSS.

If you read the Simple Customization Tutorial then you already have some basic notions about creating simple styles. Our document contains elements from different namespaces, so we will use CSS Level 3 extensions supported by the <oXygen/> layout engine to associate specific properties with that element.

Note

Please note that the CSS Level 3 is a standard under development, and has not been released yet by the W3C. However, it addresses several important issues like selectors that are namespace aware and values for the CSS properties extracted from the attributes of the XML documents. Although not (yet) conforming with the current CSS standard these are supported by the <oXygen/> Author.

Defining the General Layout.

We are now creating the basic layout of the rendered documents.

Elements that are stacked one on top of the other are: `book`, `article`, `section`, `title`, `figure`, `table`, `image`. These elements are marked as having `block` style for display. Elements that are placed one after the other in a flowing sequence are: `b`, `i`. These will have `inline` display.

```
/* Vertical flow */
book,
section,
para,
title,
image,
ref {
    display:block;
}

/* Horizontal flow */
b,i {
    display:inline;
}
```

Important

Having `block` display children in an `inline` display parent, makes <oXygen/> Author change the style of the parent to `block` display.

Styling the `section` Element.

The title of any section must be bold and smaller than the title of the parent section. To create this effect we have to create a sequence of CSS rules. The `*` operator matches any element, so we can use it to match titles having progressive depths in the document.

```
title{
    font-size: 2.4em;
    font-weight:bold;
}
* * title{
    font-size: 2.0em;
}
* * * title{
    font-size: 1.6em;
}
* * * * title{
    font-size: 1.2em;
}
```

Note

CSS rules are combined as follows:

- All the rules that match an element are kept as a list. The more specific the rule is, the further it will be placed to the end of the list.
- If there is no difference in the specificity of the rules, they are placed in the list in the same order as they appear in the CSS document.
- The list is then iterated, and all the properties from the rules are collected, overwriting the already collected values from the previous rules. That is why the font-size is changed depending on the depth of the element, while the font-weight property remains unchanged - no other rule is overwriting it.

It's useful to have before the title a constant text, indicating that it refers to a section. This text can include also the current section number. To achieve we have to use the `:before` and `:after` pseudo elements, plus the CSS counters.

We declare a counter named `sect` for each `book` or `article`. The counter is set to zero at the beginning of each such element:

```
book,
article{
    counter-reset:sect;
}
```

The `sect` counter is incremented with each `section`, that is the a direct child of a `book` or an `article` element.

```
book > section,
article > section{
    counter-increment:sect;
}
```

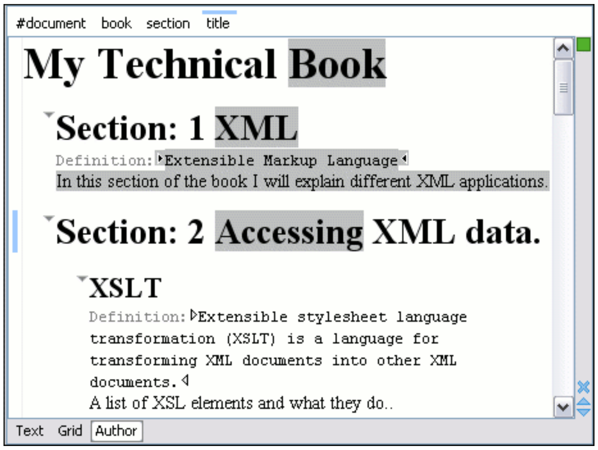
The "static" text that will prefix the section title is composed of the constant "Section ", followed by the decimal value of the `sect` counter and a dot.

```
book > section > title:before,
article > section > title:before{
    content: "Section " counter(sect) ". ";
}
```

To make the documents easy to read, we will add a margin to the sections. In this way the higher nesting level, the larger the left side indent. The margin is expressed relatively to the parent bounds:

```
section{
    margin-left:1em;
    margin-top:1em;
}
```

Figure 7.3. A sample of nested sections and their titles.



My Technical Book

Section: 1 XML

Definition: Extensible Markup Language

In this section of the book I will explain different XML applications.

Section: 2 Accessing XML data.

XSLT

Definition: Extensible stylesheet language transformation (XSLT) is a language for transforming XML documents into other XML documents.

A list of XSL elements and what they do..

In the above screenshot you can see a sample XML document rendered by our CSS. The selection "avoids" the text that is generated by the CSS "content" property. This happens because the CSS generated text is not present in the XML document and is just a visual aid.

Styling the `table` Element.

There are standard CSS properties used to indicate what elements are tables, table rows and table cells. What CSS is missing is the possibility to indicate the cell spanning. <oXygen/> Author offers support for adding an extension to solve this problem. This will be presented in the next chapters.

The table in this example is a simple one. The header must be formatted in a different way than the ordinary rows, so it will have a background color.

```
table{
    display:table;
    border:1px solid navy;
    margin:1em;
    max-width:1000px;
    min-width:150px;
}

table[width]{
  width:attr(width, length);
}

tr, header{
    display:table-row;
}

header{
    background-color: silver;
    color:inherit
}

td{
  display:table-cell;
  border:1px solid navy;
  padding:1em;
}
```

Note

Children elements with `block` or `table-caption` display placed at the beginning or the end of an element displayed as a `table`, will be grouped and presented as blocks at the top or the bottom of the table.

Note

Mixing elements having `table-cell`, `table-group`, `table-row`, etc.. display type with others that have `block` or `inline` display or with text content breaks the layout of the table. In such cases the table is shown as a `block`.

Note

Having child elements that do not have `table-cell` or `table` display in a parent with `table-row` display breaks the table layout. In this case the `table` display is supported for the children of the `table-row` element in order to allow sub-tables in the parent table.

Note

<oXygen/> Author can automatically detect the spanning of a cell, without the need to write a Java extension for this.

This happens if the span of the cell element is specified using the **colspan** and **rowspan** attributes, just like in HTML, or **cols** and **rows** attributes.

For instance, the following XML code:

```
  <table>
        <tr>
            <td>Cell 1.1</td>
            <td>Cell 1.2</td>
            <td>Cell 1.3</td>
        </tr>
        <tr>
            <td>Cell 2.1</td>
            <td colspan="2" rowspan="2">
            Cell spanning 2 rows and 2 columns.
            </td>
        </tr>
        <tr><td>Cell 3.1</td></tr>
  </table>
```

using the CSS:

```
table{
   display: table;
}
tr{
   display: table-row;
}
td{
    display: table-cell;
}
```

is rendered correctly:

Table 7.1. Built-in Cell Spanning

<table>
<tr><td>Cell 1.1</td><td>Cell 1.2</td><td>Cell 1.3</td></tr>
<tr><td>Cell 2.1</td><td rowspan="2" colspan="2">Cell spanning 2 rows and 2 columns</td></tr>
<tr><td>Cell 3.1</td></tr>
</table>

Because in our schema the `td` tag has the attributes **row_span** and **column_span** that are not automatically recognized by <oXygen/> Author, we will implement a Java extension which will provide information about the cell spanning. See the section Configuring a Table Cell Span Provider.

Because the column widths are specified by the attributes **width** of the elements `customcol` that are not automatically recognized by <oXygen/> Author, it is necessary to implement a Java extension which will provide information about the column widths. See the section Configuring a Table Column Width Provider.

Styling the Inline Elements.

The "bold" style is obtained by using the `font-weight` CSS property with the value `bold`, while the "italic" style is specified by the `font-style` property:

```
b {
    font-weight:bold;
}

i {
    font-style:italic;
}
```

Styling Elements from other Namespace

In the CSS Level 1, 2, and 2.1 there is no way to specify if an element X from the namespace Y should be presented differently from the element X from the namespace Z. In the upcoming CSS Level 3, it is possible to differentiate elements by their namespaces. <oXygen/> Author supports this CSS Level 3 functionality. For more information see the Namespace Selectors section.

To match our `def` element we will declare its namespace, bind it to the *abs* prefix, and then write a CSS rule:

```
@namespace abs "http://www.oxygenxml.com/sample/documentation/abstracts";

abs|def{
    font-family:monospace;
    font-size:smaller;
}
abs|def:before{
    content:"Definition:";
    color:gray;
}
```

Styling images

The CSS 2.1 does not specify how an element can be rendered as an image. To overpass this limitation, <oXygen/> Author supports a CSS Level 3 extension allowing to load image data from an URL. The URL of the image must be specified by one of the element attributes.

Note

<oXygen/> Author recognizes the following image file formats: JPEG, GIF, PNG and SVG. The oXygen Author for Eclipse does not render the SVG files.

```
image{
    display:block;
    content: attr(href, url);
    margin-left:2em;
}
```

Our `image` element has the required attribute `href` of type `xs:anyURI`. The `href` attribute contains an image location so the rendered content is obtained by using the function:

```
attr(href, url)
```

Important

The first argument is the name of the attribute pointing to the image file. The second argument of the `attr` function specifies the type of the content. If the type has the `url` value, then <oXygen/> identifies the content as being an image. If the type is missing, then the content will be the text representing the attribute value.

Important

<oXygen/> Author handles both absolute and relative specified URLs. If the image has an *absolute* URL location (e.g: "http://www.oasis-open.org/images/standards/oasis_standard.jpg") then it is loaded directly from this location. If the image URL is *relative* specified to the XML document (e.g: "images/my_screenshot.jpg") then the location is obtained by adding this value to the location of the edited XML document.

An image can also be referenced by the name of a DTD entity which specifies the location of the image file. For example if the document declares an entity **graphic** which points to a JPEG image file:

```
<!ENTITY graphic SYSTEM "depo/keyboard_shortcut.jpg" NDATA JPEG>
```

and the image is referenced in the XML document by specifying the name of the entity as the value of an attribute:

```
<mediaobject>
    <imageobject>
        <imagedata entityref="graphic" scale="50"/>
    </imageobject>
</mediaobject>
```

The CSS should use the functions **url**, **attr** and **unparsed-entity-uri** for displaying the image in the Author mode:

Note

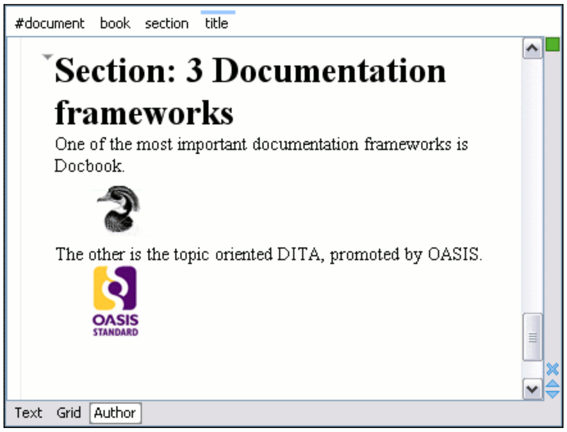
Note that the scale attribute of the imagedata element will be considered without the need of a CSS customization and the image will be scaled accordingly.

```
imagedata[entityref]{
    content: url(unparsed-entity-uri(attr(entityref)));
}
```

To take into account the value of the `width` attribute of the `imagedata` and use it for resizing the image, the CSS can define the following rule:

```
imagedata[width]{
  width:attr(width, length);
}
```

Figure 7.4. Samples of images in Author



Section: 3 Documentation frameworks

One of the most important documentation frameworks is Docbook.

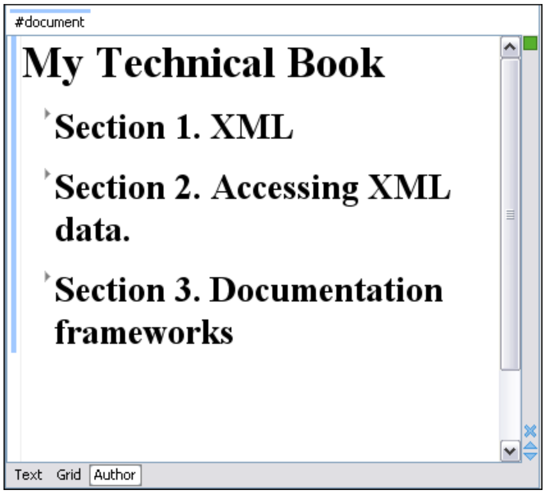
The other is the topic oriented DITA, promoted by OASIS.

Marking elements as foldable

You can specify what elements are collapsible. The collapsible elements are rendered having a small triangle icon in the top left corner. Clicking on this icon hides or shows the children of the element. In our case, we will mark the `section` elements as foldable. We will leave only the `title` child elements visible.

```
section{
    foldable:true;
    not-foldable-child: title;
}
```

Figure 7.5. Folded Sections


#document
My Technical Book
Section 1. XML
Section 2. Accessing XML data.
Section 3. Documentation frameworks
Text
Grid
Author

Marking elements as links

You can specify what elements are links. The text content specified in the `:before` pseudo element will be underlined. When hovering the mouse over that content the mouse pointer will change to indicate that it can follow the link. Clicking on a link will result in the referred resource being opened in an editor. In our case we will mark the `link` elements as links with the `href` attribute indicating the referred location.

```
link[href]:before{
    display:inline;
    link:attr(href);
    content: "Click to open: " attr(href);
}
```

Note

If you plan to use IDs as references for links, the value of the link property should start with a sharp sign(#). This will ensure that the default link target reference finder implementation will work and clicking on the link will send you to the indicated location in the document. For more details about the link target reference finder read the section Configuring a Link target reference finder.

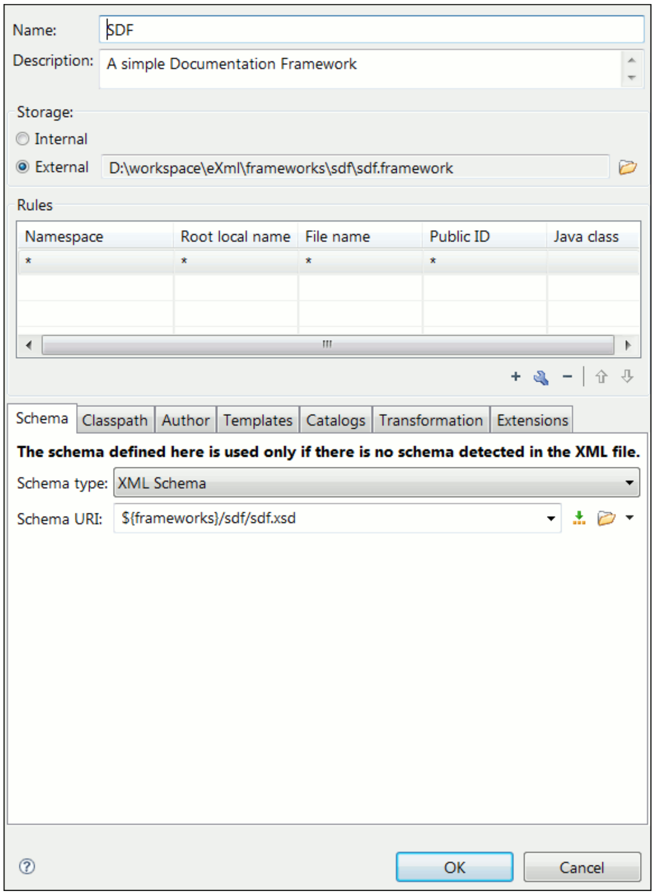
Example 7.1. IDs as references for links

```
link[linkend]:before{
    display:inline;
    link: "#" attr(linkend);
    content: "Click to open: " attr(linkend);
}
```

Third Step. The Association.

Now that we have the XML Schema and the CSS stylesheet for the documents we intend to edit, we can proceed to create a distributable framework package for our content authors.

Figure 7.6. The Document Type Dialog



Name: SDF

Description: A simple Documentation Framework

Storage:

○ Internal

● External D:\workspace\eXml\frameworks\sdf\sdf.framework

Rules

Namespace	Root local name	File name	Public ID	Java class
*	*	*	*	

The schema defined here is used only if there is no schema detected in the XML file.

Schema type: XML Schema

Schema URI: ${frameworks}/sdf/sdf.xsd

Organizing the Framework Files

First create a new folder called `sdf` (from "Simple Documentation Framework") in `{oXygen_installation_directory}/frameworks`. We will use this folder to store all files related to our documentation framework. Let's organise it a bit, by creating the following folder structure:

```
oxygen
  frameworks
     sdf
       schema
       css
```

Important

The `frameworks` directory is the container where all the oXygen framework customizations are located.

Each subdirectory contains files related to a specific type of XML documents: schemas, catalogs, stylesheets, CSS files, etc.

Distributing a framework means delivering a framework directory.

Important

We assume you have the right to create files and folder inside the oXygen installation directory. If you do not have this right, you will have to install another copy of the program in a folder you have access to, the home directory for instance, or your desktop. You can download the "all platforms" distribution from the oXygen website and extract it in the chosen folder.

To test your framework distribution you will need to copy it in the `frameworks` directory of the newly installed application and start oXygen by running the provided start-up script files.

We should copy the created schema files `abs.xsd` and `sdf.xsd`, `sdf.xsd` being the master schema, to the `schema` directory and the CSS file `sdf.css` to the `css` directory.

Association Rules

We must specify when oXygen should use the files created in the previous section by creating a document type association. Open the Document Type dialog by following the procedure:

1. Open the Options Dialog, and select the Document Types Association option pane.

2. Select the **Developer** user role from the **User role** combo box at the top of the dialog. This is important, because it will allow us to save the document type association in a file on disk, instead of <oXygen/> options.

3. Click on the **New** button.

In the displayed dialog, fill in the following data:

Name

Enter **SDF** - This is the name of the document type.

Description

Enter **Simple Documentation Framework** - This is a short description helping the other users understand the purpose of the Document Type.

Storage

The storage refers to the place where the Document Type settings are stored. Internal means the Document Types are stored in the default <oXygen/> preferences file. Since we want to share the Document Type to other users, we must select External, and choose a file.

The file must be in the `{oXygen_installation_directory}/frameworks/sdf` directory. A possible location is **/Users/{user_name}/Desktop/oxygen/frameworks/sdf/sdf.framework**. The framework directory structure will be:

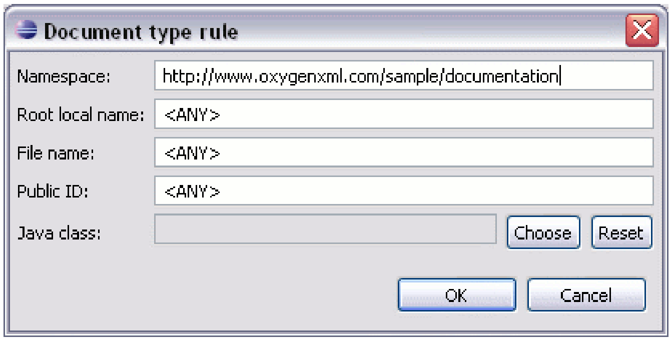
```
oxygen
  frameworks
     sdf
       sdf.framework
       schema
         sdf.xsd
       css
         sdf.css
```

Rules

If a document opened in <oXygen/> matches one of the rules defined for the Document Type, then it is activated.

Press the + Add button from the Rules section. Using the newly displayed dialog, we add a new rule that matches documents with the root from the namespace: `http://www.oxygenxml.com/sample/documentation`. The root name, file name or PublicID are not relevant.

Figure 7.7. Editing a rule



Document type rule

Namespace: http://www.oxygenxml.com/sample/documentation

Root local name: <ANY>

File name: <ANY>

Public ID: <ANY>

Java class:

A document matches a rule when it fulfills the conditions imposed by each field of the rule:

Namespace

the namespace of the root element declared in the XML documents of the current document type. A value of ANY_VALUE matches any namespace in an XML document. Value may contain wildcards(*, ?) and editor variables. Multiple values separated by comma(,) are accepted.

Root local name

The local name of the root element of the XML documents of the current document type. A value of ANY_VALUE matches any local name of the root element. Value may contain wildcards(*, ?) and editor variables. Multiple values separated by comma(,) are accepted.

File name

The file name of the XML documents of the current document type. A value of ANY_VALUE matches any file name. Value may contain wildcards(*, ?) and editor variables. Multiple values separated by comma(,) are accepted.

Public ID

The public ID of the XML documents of the current document type (for a document validated against a DTD). A value of ANY_VALUE matches any public ID. Value may contain wildcards(*, ?) and editor variables. Multiple values separated by comma(,) are accepted.

Java class

The full name of a Java class that has access to all root element attributes and the above 4 values in order to decide if the document matches the rule.

Java API: Rules implemented in Java

An alternative to the rule we defined for our association is to write the entire logic in Java.

1. Create a new Java project, in your IDE.

   Create the `lib` directory in the Java project directory and copy there the `oxygen.jar` file from the `{oXygen_installation_directory}/lib`. The `oxygen.jar` contains the Java interfaces we have to implement and the available Author API needed to access its features.

2. Create the class `simple.documentation.framework.CustomRule`. This class must implement the `ro.sync.ecss.extensions.api.DocumentTypeCustomRuleMatcher` interface.

The interface defines two methods: `matches`, and `getDescription`.

1. The `matches` method is the one that is invoked when the edited document must be checked against the document type association. It takes as arguments the root local name, its namespace, the document location URI, the PublicID and the root element attributes. It must return `true` when the document matches the association.
2. The `getDescription` method returns a description of the rule.

Here is the implementation of these two methods. The implementation of `matches` is just a Java equivalent of the rule we defined earlier.

```
public boolean matches(
  String systemID,
  String rootNamespace,
  String rootLocalName,
  String doctypePublicID,
  Attributes rootAttributes) {

 return "http://www.oxygenxml.com/sample/documentation"
  .equals(rootNamespace);
}

public String getDescription() {
 return "Checks if the current Document Type Association"
 + " is matching the document.";
}
```

The complete source code is found in the Example Files Listings, the Java Files section.

3. Package the compiled class into a *jar* file. Here is an example of an ANT script that packages the `classes` directory content into a *jar* archive named `sdf.jar`:

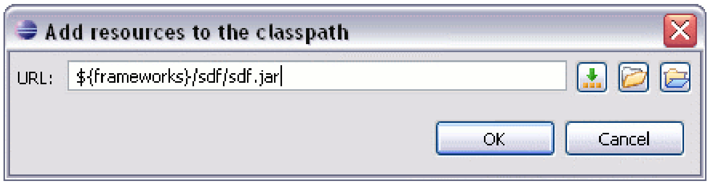
```
<?xml version="1.0" encoding="UTF-8"?>
<project name="project" default="dist">
    <target name="dist">
   <jar destfile="sdf.jar" basedir="classes">
    <fileset dir="classes">
     <include name="**/*"/>
     </fileset>
   </jar>
    </target>
</project>
```

4. Copy the `sdf.jar` file into the `frameworks/sdf` directory.
5. Add the `sdf.jar` to the Author classpath. To do this select **SDF** Document Type from the **Document Type Association** options page and press the Edit button.

   Select the Classpath tab in the lower part of the dialog.

   Press the + Add button . In the displayed dialog enter the location of the jar file, relative to the <oXygen/> `frameworks` directory. If you are in the process of developing the extension actions you can also specify a path to a directory which holds compiled Java classes.

Figure 7.8. Adding a classpath entry



Add resources to the classpath

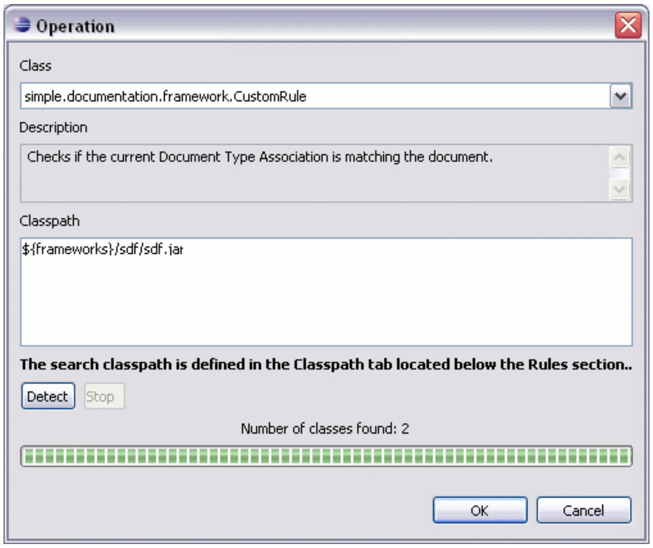
URL: ${frameworks}/sdf/sdf.jar

6. Clear the rules we defined before by using the − Remove button.

   Press the + Add button from the Rules section.

   Press the Choose button that follows the Java class value. The following dialog is displayed:

Figure 7.9. Selecting a Java association rule.



Operation

Class: simple.documentation.framework.CustomRule

Description: Checks if the current Document Type Association is matching the document.

Classpath: ${frameworks}/sdf/sdf.jar

The search classpath is defined in the Classpath tab located below the Rules section..

Number of classes found: 2

To test the association, open the `sdf.xml` sample and validate it.

Schema Settings

In the dialog for editing the Document Type properties, in the bottom section there are a series of tabs. The first one refers to the schema that is used for validation of the documents that match the defined association **Rules**.

Important

If the document refers a schema, using for instance a `DOCTYPE` declaration or a `xsi:schemaLocation` attribute, the schema from the document type association will not be used when validating.

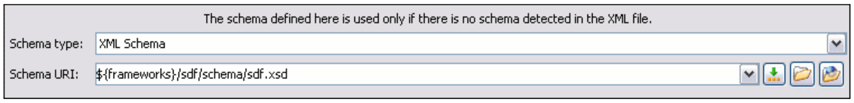
Schema Type
: Select from the combo box the value **XML Schema**.

Schema URI
: Enter the value `${frameworks}/sdf/schema/sdf.xsd`. We should use the `${frameworks}` editor variable in the schema URI path instead of a full path in order to be valid for different <oXygen/> installations.

Important

The `${frameworks}` variable is expanded at the validation time into the absolute location of the directory containing the frameworks.

Figure 7.10. The Schema panel



The schema defined here is used only if there is no schema detected in the XML file.

Schema type: XML Schema

Schema URI: ${frameworks}/sdf/schema/sdf.xsd

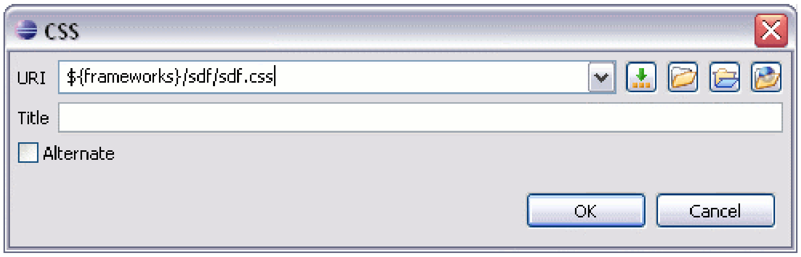
Author CSS Settings

Select the Author tab from the Document Type edit dialog. By clicking on the CSS label in the right part of the tab the list of associated CSS files is shown.

Here you can also specify how should the CSSs defined in the document type be treated when there are CSSs specified in the document(with `xml-stylesheet` processing instructions). The CSSs from the document can either replace the CSSs defined in the document type association or merge with them.

Add the URI of the CSS file `sdf.css` we already defined. We should use the `${frameworks}`editor variable in the file path.

Figure 7.11. CSS settings dialog



URI: ${frameworks}/sdf/sdf.css

Title:

□ Alternate

The Title text field refers to a symbolic name for the stylesheet. When adding several stylesheets with different titles to a Document Type association, the content author can select what CSS will be used for editing from the **Author CSS Alternatives** toolbar.

This combo-box from the toolbar is also populated in case your XML document refers CSS files directly using `xml-stylesheet` processing instructions, and the processing instructions define titles for the CSS files.

Note

The CSS settings dialog allows to create a *virtual*`xml-stylesheet` processing instructions. The CSS files defined in the Document Type Association dialog and the `xml-stylesheet` processing instructions from the XML document are processed together, as being all a list of processing instructions.

<oXygen/> Author fully implements the W3C recommendation regarding "Associating Style Sheets with XML documents". For more information see: http://www.w3.org/TR/xml-stylesheet/http://www.w3.org/TR/REC-html40/present/styles.html#h-14.3.2

Testing the Document Type Association

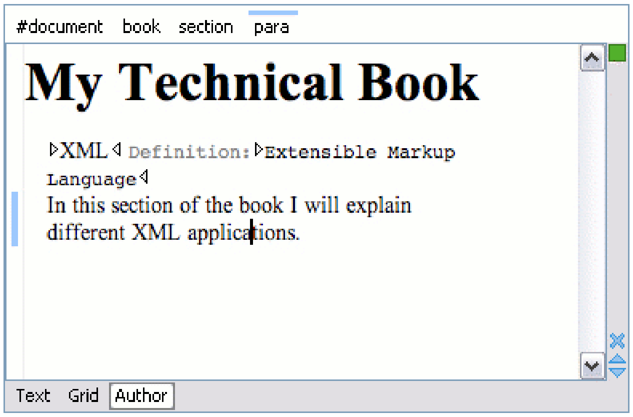
To test the new Document Type create an XML instance that is conforming with our Simple Document Format. We will not specify an XML Schema location directly in the document, using an `xsi:schemaLocation` attribute; <oXygen/> will detect instead its associated document type and use the specified schema.

```
<book xmlns="http://www.oxygenxml.com/sample/documentation"
      xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts">

    <title>My Technical Book</title>
    <section>
        <title>XML</title>
        <abs:def>Extensible Markup Language</abs:def>
        <para>In this section of the book I will
           explain different XML applications.</para>
    </section>
</book>
```

When trying to validate the document there should be no errors. Now modify the `title` to `title2`. Validate again. This time there should be one error:

```
  cvc-complex-type.2.4.a: Invalid content was found starting with element
  'title2'. One of '{"http://www.oxygenxml.com/sample/documentation":title}'
  is expected.
```

Undo the tag name change. Press on the Author button at the bottom of the editing area. <oXygen/> should load the CSS from the document type association and create a layout similar to this:



My Technical Book

XML Definition: Extensible Markup Language

In this section of the book I will explain different XML applications.

Packaging and Deploying

Using a file explorer, go to the <oXygen/> `frameworks` directory. Select the `sdf` directory and make an archive from it. Move it to another <oXygen/> installation (eventually on another computer). Extract it in the `frameworks` directory. Start <oXygen/> and test the association as explained above.

If you create multiple document type associations and you have a complex directory structure it might be easy from the deployment point of view to use an <oXygen/> all platforms distribution. Add your framework files to it, repackage it and send it to the content authors.

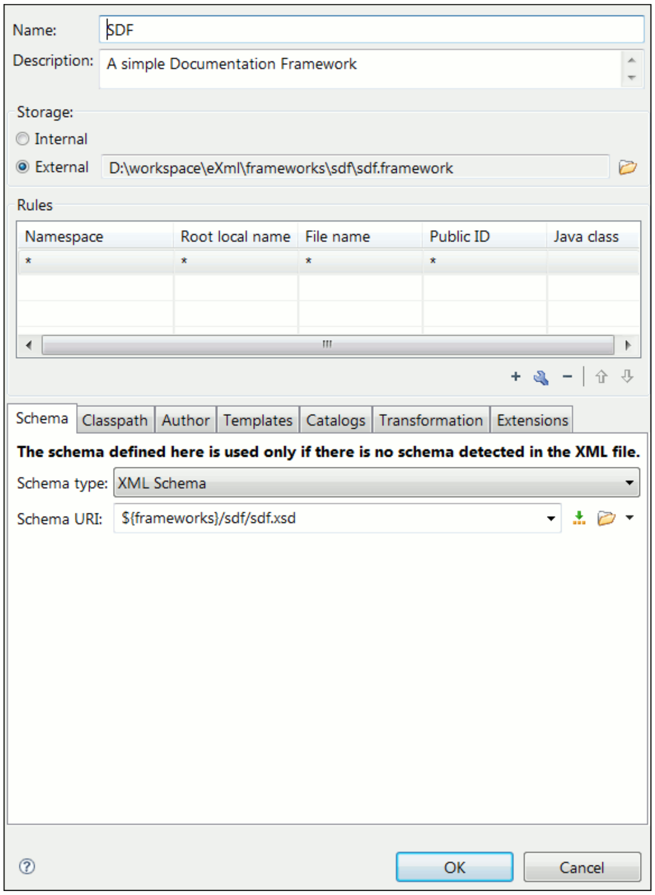
Warning

When deploying your customized sdf directory please make sure that your sdf directory contains the sdf.framework file (that is the file defined as External Storage in Document Type Association dialog shall always be stored inside the sdf directory). If your external storage points somewhere else <oXygen/> will not be able to update the Document Type Association options automatically on the deployed computers.

Author Settings

You can add a new *Document Type Association* or edit the properties of an existing one from the Options+Preferences+Document Type Association option pane. All the changes can be made into the *Document type* edit dialog.

Figure 7.12. The Document Type Dialog



Name: SDF

Description: A simple Documentation Framework

Storage:

○ Internal

● External D:\workspace\eXml\frameworks\sdf\sdf.framework

Rules

Namespace	Root local name	File name	Public ID	Java class
*	*	*	*	

The schema defined here is used only if there is no schema detected in the XML file.

Schema type: XML Schema

Schema URI: ${frameworks}/sdf/sdf.xsd

Configuring Actions, Menus and Toolbars

The <oXygen/> Author toolbars and menus can be changed to provide a productive editing experience for the content authors. You can create a set of actions that are specific to a document type.

In our example, the `sdf` framework, we created the stylesheet and the validation schema. Now let's add some actions for inserting a `section` and a `table`. To add a new action, follow the procedure:

1. Open the Options Dialog, and select the Document Types Association option pane.

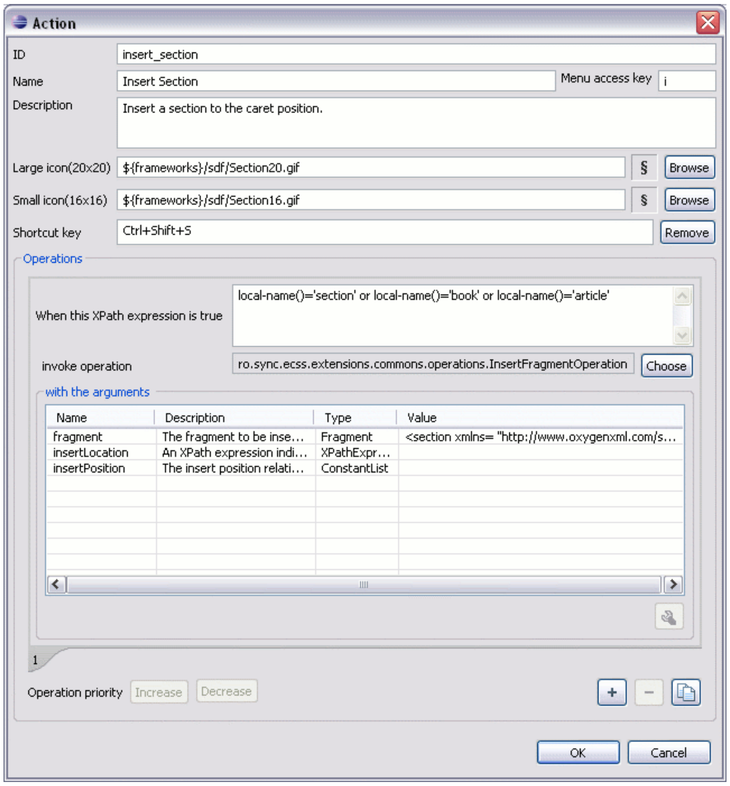
2. In the lower part of the Document Type Association dialog, click on the Author tab, then select the Actions label.

3. To add a new action click on the + Add button.

The Insert Section Action

This paragraph describes how you can define the action for adding a section. We assume the icon files § `Section16.gif` for the menu item and § `Section20.gif` for the toolbar, are already available. Although we could use the same icon size for both menu and toolbar, usually the icons from the toolbars are larger than the ones placed in the menus. These files should be placed in the `frameworks/sdf` directory.

Figure 7.13. The Action Edit Dialog



ID: insert_section

Name: Insert Section

Menu access key: i

Description: Insert a section to the caret position.

Large icon(20x20): ${frameworks}/sdf/Section20.gif

Small icon(16x16): ${frameworks}/sdf/Section16.gif

Shortcut key: Ctrl+Shift+S

When this XPath expression is true: local-name()='section' or local-name()='book' or local-name()='article'

invoke operation: ro.sync.ecss.extensions.commons.operations.InsertFragmentOperation

with the arguments

Name	Description	Type	Value
fragment	The fragment to be inse...	Fragment	<section xmlns= "http://www.oxygenxml.com/s...
insertLocation	An XPath expression indi...	XPathExpr...	
insertPosition	The insert position relati...	ConstantList	

Operation priority

ID — An unique identifier for the action. You can use **insert_section**.

Name — The name of the action. It is displayed as a tooltip when the action is placed in the toolbar, or as the menu item name. Use **Insert section**.

Menu access key — On Windows, the menu items can be accessed using (ALT + letter) combination, when the menu is visible. The letter is visually represented by underlining the first letter from the menu item name having the same value. Since the name is **Insert section**, we can use as a menu access key the letter **s**.

Description — You can add a short description for the action. In our case **Adds a section element** will suffice.

Large icon (20x20) — The path to the file that contains the toolbar image for the action. A good practice is to store the image files inside the framework directory. This way we can use the editor variable`${frameworks}` to make the image file relative to the framework location. Insert **${frameworks}/sdf/Section20.gif**

Note

If the images are bundled in a jar archive together with some Java operations implementation for instance, it might be convenient for you to refer the images not by the file name, but by their relative path location in the class-path.

If the image file `Section20.gif` is located in the directory `images` inside the jar archive, you can refer to it by using **/images/Section20.gif**. The jar file must be added into the Classpath list.

Small icon (16x16)

The path to the file that contains the menu image. Insert **${frameworks}/sdf/Section16.gif**

Shortcut key

A shortcut key combination for triggering the action. To define it, click in the text field and press the desired key combination. We can choose **Ctrl+Shift+s**.

Note

The shortcut is enabled only by adding the action to the main menu of the Author mode which contains all the actions that the author will have in a menu for the current document type.

At this time the action has no functionality added to it. Next we must define how this action operates. An action can have multiple operation modes, each of them activated by the evaluation of an XPath version 2.0 expression.

Note

The XPath expression of an operation mode is evaluated relative to the **current element**. The current element is the one where the caret is positioned. In fact there is hierarchy of elements containing the caret position, but we are considering only the closest one. A simple expression like:

```
title
```

is a relative one and checks if the current element has a "title" child element. To check that the current element is a "section" we can use the expression:

```
local-name()='section'
```

Note

<oXygen/> Author determines the operation to be executed by iterating through the defined operation modes. The first operation whose XPath expression "matched" the current document context gets executed, while the others are being ignored. Make sure you order correctly your operations by placing the ones with more specific XPath selectors before the ones having more generic selectors.

For instance the expression

```
person[@name='Cris' and @age='24']
```

is more specific than

```
person[@name='Cris']
```

The action mode using the first expression must be placed before the one using the second expression in the action modes list.

We decide that we can add sections only if the current element is either a `book`, `article`, or another `section`.

XPath expression

Set the value to:

```
local-name()='section' or local-name()='book' or
 local-name()='article'
```

Invoke operation

A set of built-in operations is available. A complete list is found in the Author Default Operations section. To this set you can add your own Java operation implementations. In our case, we'll use the **InsertFragmentOperation** built-in operation, that inserts an XML fragment at the caret position.

Configure the arguments by setting the following values:

fragment

```
<section xmlns=
"http://www.oxygenxml.com/sample/documentation">
     <title/>
</section>
```

insertLocation

Leave it empty. This means the location will be the element at the caret position.

insertPosition

Select "Inside".

The Insert Table Action

We will create an action that inserts into the document a table with three rows and three columns. The first row is the table header. Similarly to the insert section action, we will use the **InsertFragmentOperation**.

We assume the icon files `Table16.gif` for the menu item and `Table20.gif` for the toolbar, are already available. We place these files in the `frameworks/sdf` directory.

The action properties:

ID	You can use **insert_table**.
Name	Insert **Insert table**.
Menu access key	Enter the **t** letter.
Description	We can use **Adds a section element**.
Toolbar icon	Use **${frameworks}/sdf/Table20.gif**
Menu icon	Insert **${frameworks}/sdf/Table16.gif**
Shortcut key	We can choose **Ctrl+Shift+t**.

Now let's set up the operation the action uses.

XPath expression

Set it to the value

```
true()
```

Note

true() is equivalent with leaving this field empty.

Invoke operation

We'll use **InsertFragmentOperation** built-in operations that inserts an XML fragment at the caret position.

Configure its arguments by setting the values:

fragment

```
<table xmlns=
"http://www.oxygenxml.com/sample/documentation">
  <header><td/><td/><td/></header>
  <tr><td/><td/><td/></tr>
  <tr><td/><td/><td/></tr>
</table>
```

insertLocation

In our example will always add tables at the end of the section that contains the caret position. Use:

```
ancestor::section/*[last()]
```

insertPosition

Select "After".

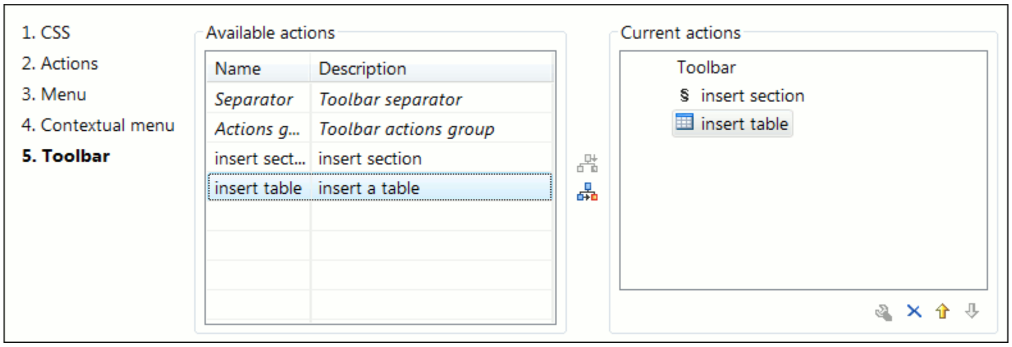
Configuring the Toolbar

Now that we have defined the two actions we can add them to the toolbar.

The first thing to check is that the toolbar Author custom actions should be displayed when switching to the **Author** mode: Right click in the application window upper part, in the area that contains the toolbar buttons and check Author custom actions in the displayed menu if it is unchecked.

Open the Document Type edit dialog for the **SDF** framework and select on the Author tab. Next click on the Toolbar label.

Figure 7.14. Configuring the Toolbar



1. CSS
2. Actions
3. Menu
4. Contextual menu
5. **Toolbar**

Available actions

Name	Description
Separator	*Toolbar separator*
Actions g...	*Toolbar actions group*
insert sect...	insert section
insert table	insert a table

Current actions

Toolbar
- insert section
- insert table

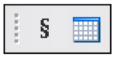
The panel is divided in two sections: the left side contains a list of actions, while the right one contains an action tree, displaying the list of actions added in the toolbar. The special entry called *Separator* allows you to visually separate the actions in the toolbar.

Select the Insert section action in the left and the Toolbar label in the right, then press the Add as child button.

Now select the Insert table action in the left and the Insert section in the right. Press the Add as sibling button.

When opening a **Simple Documentation Framework** test document in Author mode, the toolbar below will be displayed at the top of the editor.

Figure 7.15. Author Custom Actions Toolbar



Configuring the Main Menu

Defined actions can be grouped into customized menus in the <oXygen/> menu bar. For this open the Document Type dialog for the **SDF** framework and click on the Author tab. Next click on the Menu label.

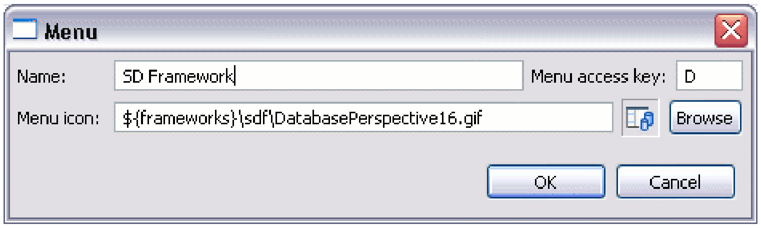
In the left side we have the list of actions and some special entries:

Submenu Creates a submenu. You can nest an unlimited number of menus.

Separator Creates a separator into a menu. In this way you can logically separate the menu entries.

In the right side we have the menu tree, having the Menu entry as root. To change its name click on this label to select it, then press the Edit button. Enter **SD Framework** as name, and **D** as menu access key.

Figure 7.16. Changing the Name of the Menu



Name: SD Framework

Menu access key: D

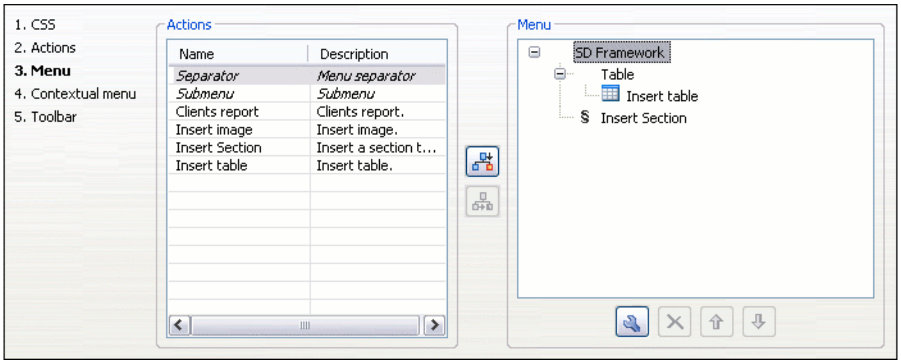
Menu icon: ${frameworks}\sdf\DatabasePerspective16.gif

Select the Submenu label in the left an the SD Framework label in the right, then press the Add as child button. Change the submenu name to Table, using the Edit button.

Select the Insert section action in the left and the Table label in the right, then press the Add as sibling button.

Now select the Insert table action in the left and the Table in the right. Press the Add as child button.

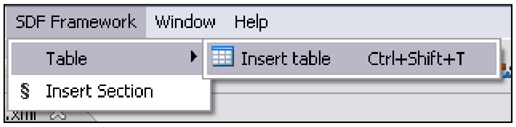
Figure 7.17. Configuring the Menu



Name	Description
Separator	*Menu separator*
Submenu	*Submenu*
Clients report	Clients report.
Insert image	Insert image.
Insert Section	Insert a section t...
Insert table	Insert table.

When opening a **Simple Documentation Framework** test document in Author mode, the menu we created is displayed in the editor menu bar, between the Debugger and the Document menus. In the menu we find the Table submenu and the two actions:

Figure 7.18. Author Menu



Note

The shortcut of an action defined for the current document type is enabled only if the action is added to the main menu. Otherwise the author can run the action only from the toolbar.

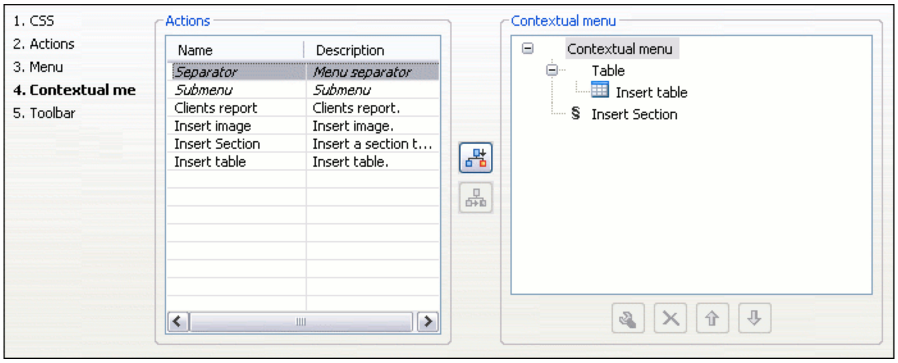
Configuring the Contextual Menu

The contextual menu is shown when you right click (on Mac OS X it is used the combination **ctrl** and mouse click) in the Author editing area. In fact we are configuring the bottom part of the menu, since the top part is reserved for a list of generic actions like Copy, Paste, Undo, etc..

Open the Document Type dialog for the **SDF** framework and click on the Author tab. Next click on the Contextual Menu label.

Follow the same steps as explained above in the Configuring the Main Menu, except changing the menu name - the contextual menu has no name.

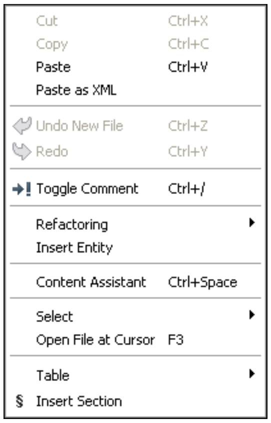
Figure 7.19. Configuring the Contextual Menu



Name	Description
Separator	*Menu separator*
Submenu	*Submenu*
Clients report	Clients report.
Insert image	Insert image.
Insert Section	Insert a section t...
Insert table	Insert table.

To test it, open the test file, and click to open the contextual menu. In the lower part there is shown the Table sub-menu and the Insert section action:

Figure 7.20. Author Contextual Menu



Author Default Operations

Below are listed all the operations and their arguments.

InsertFragmentOperation

Inserts an XML fragment at the current cursor position. The selection - if there is one, remains unchanged. The fragment will be inserted in the current context of the cursor position. That means that if the current XML document uses some namespace declarations then the inserted fragment must use the same declarations. The inserted fragment will not be copied and pasted to the cursor position, but the namespace declarations of the fragment will be adapted if needed to the existing namespace declarations of the XML document. Examples of namespace adjusting when the fragment is inserted and the descriptions of the arguments are described here.

InsertOrReplaceFragmentOperation

Similar to **InsertFragmentOperation**, except it removes the selected content before inserting the fragment.

InsertOrReplaceTextOperation

Inserts a text. It removes the selected content before inserting the text section.

text — The text section to insert.

SurroundWithFragmentOperation

Surrounds the selected content by a fragment. Since the fragment can have multiple nodes, the surrounded content will be always placed in the first leaf element. If there is no selection, the operation will simply insert the fragment at the caret position. The arguments are described here.

SurroundWithTextOperation

The surround with text operation takes two arguments, two text values that will be inserted before and after the selected content. If there is no selected content, the two sections will be inserted at the caret position. The arguments of the operation are:

header

The text that will be placed before the selection.

footer

The test that will be placed after the selection.

The arguments of `InsertFragmentOperation`

fragment

The value for this argument is a text. This is parsed by the <oXygen/> Author as it was already in the document at the caret position. You can use entities references declared in the document and it is namespace aware. The fragment may have multiple roots.

Note

You can use even namespace prefixes that are not declared in the inserted fragment, if they are declared in the document where the insertion is done. For clarity, we recommend always to prefix and declare namespaces in the inserted fragment!

Note

If there are namespace declarations in the fragment that are identical to the in the document insertion context, the namespace declaration attributes are removed from the fragment elements.

Example 7.2. Prefixes that are not bound explicitly

For instance, the fragment:

```
<x:item id="dty2"/>
&ent;
<x:item id="dty3"/>
```

Can be correctly inserted in the document: ('|' marks the insertion point):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE x:root [
    <!ENTITY ent "entity">
]>

<x:root xmlns:x="nsp">
  |
</x:root>
```

Result:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE x:root [
    <!ENTITY ent "entity">
]>
<x:root xmlns:x="nsp">
    <x:item id="dty2"/>
    &ent;
    <x:item id="dty3"/>
</x:root>
```

Example 7.3. Default namespaces

If there is a default namespace declared in the document and the document fragment does not declare a namespace, the elements from the fragment are considered to be in **no namespace**.

For instance the fragment:

```
<item id="dty2"/>
<item id="dty3"/>
```

Inserted in the document:

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="nsp">
|
</root>
```

Gives the result document:

```
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="nsp">
    <item xmlns="" id="dty2"/>
    <item xmlns="" id="dty3"/>
</root>
```

insertLocation An XPath expression that is relative to the current node. It selects the reference node for the fragment insertion.

insertPosition One of the three constants: "Inside", "After", or "Before" , showing where the insertion is made relative to the reference node selected by the **insertLocation**. "Inside" has the meaning of the first child of the reference node.

The arguments of `SurroundWithFragmentOperation`

fragment The XML fragment that will surround the selection.

Example 7.4. Surrounding with a fragment

Let's consider the fragment:

```
<F>
   <A></A>
   <B>
     <C></C>
   </B>
</F>
```

And the document:

```
<doc>
  <X></X>
  <Y></Y>
  <Z></Z>
<doc>
```

Considering the selected content that is to be surrounded is the sequence of elements `X` and `Y`, then the result is:

```
<doc>
   <F>
      <A>
         <X></X>
         <Y></Y>
      </A>
      <B>
        <C></C>
      </B>
   </F>
  <Z></Z>
<doc>
```

Because the element `A` was the first leaf in the fragment, it received the selected content. The fragment was then inserted in the place of the selection.

Java API - Extending Author Functionality through Java

<oXygen/> Author has a built-in set of operations covering the insertion of text and XML fragments (see the Author Default Operations) and the execution of XPath expressions on the current document edited in Author mode. However, there are situations in which we need to extend this set. For instance if you need to enter an element whose attributes

should be edited by the user through a graphical user interface. Or the users must send the selected element content or even the whole document to a server, for some kind of processing or the content authors must extract pieces of information from a server and insert it directly into the edited XML document. Or you need to apply an XPath expression on the current Author document and process the nodes of the result nodeset.

In the following sections we are presenting the Java programming interface (API) available to the developers. You will need the Oxygen Author SDK [http://www.oxygenxml.com/InstData/Editor/Developer/oxygenAuthorSDK.zip] available on the <oXygen/> website [http://www.oxygenxml.com/developer.html] which includes the source code of the Author operations in the predefined document types and the full documentation in Javadoc format of the public API available for the developer of Author custom actions.

The next Java examples are making use of AWT classes. If you are developing extensions for the <oXygen/> XML Editor plugin for Eclipse you will have to use their SWT counterparts.

We assume you already read the Configuring Actions, Menus, Toolbar section and you are familiar with the <oXygen/> Author customization. You may find the XML schema, CSS and XML sample in the Example Files Listings.

Warning

Make sure the Java classes of your custom Author operations are compiled with the same Java version that is used by . Otherwise the classes may not be loaded by the Java virtual machine. For example if you run with a Java 1.5 virtual machine but the Java classes of your custom Author operations are compiled with a Java 1.6 virtual machine then the custom operations cannot be loaded and used by the Java 1.5 virtual machine.

Example 1. Step by Step Example. Simple Use of a Dialog from an Author Operation.

Let's start adding functionality for inserting images, in our **Simple Documentation Framework** (shortly SDF). The images are represented by the `image` element. The location of the image file is represented by the value of the `href` attribute. In our Java implementation we will show a dialog with a text field, in which the user can enter a full URL, or he can browse for a local file.

1. Create a new Java project, in your IDE.

   Create the directory `lib` in the Java project directory and copy in it the `oxygen.jar` file from the `{oXygen_installation_directory}/lib` directory. The `oxygen.jar` contains the Java interfaces we have to implement and the API needed to access the Author features.

2. Create the class `simple.documentation.framework.InsertImageOperation`. This class must implement the `ro.sync.ecss.extensions.api.AuthorOperation` interface.

   The interface defines three methods: `doOperation`, `getArguments` and `getDescription`.

   1. The `doOperation` method is invoked when the action is performed either by pressing the toolbar button, selecting the menu item or through the shortcut. It takes as arguments an object of type `AuthorAccess` and a map or argument names and values.

   2. The `getArguments` method is used by <oXygen/> when the action is configured, it returns the list of arguments (name and type) that are accepted by the operation.

   3. The `getDescription` method is also used by <oXygen/> when the operation is configured and its return value describes what the operation does.

   Here is the implementation of these three methods.

   ```
   /**
   * Performs the operation.
   ```

```
 */
public void doOperation(
               AuthorAccess authorAccess,
               ArgumentsMap arguments)
 throws IllegalArgumentException,
                  AuthorOperationException {

 JFrame oxygenFrame = (JFrame) authorAccess.getParentFrame();
 String href = displayURLDialog(oxygenFrame);
 if (href.length() != 0) {
     // Creates the image XML fragment.
     String imageFragment =
        "<image xmlns='http://www.oxygenxml.com/sample/documentation' href='"
        + href + "'/>";

     // Inserts this fragment at the caret position.
     int caretPosition = authorAccess.getCaretOffset();
     authorAccess.insertXMLFragment(imageFragment, caretPosition);
 }
}

/**
 * Has no arguments.
 *
 * @return null.
 */
public ArgumentDescriptor[] getArguments() {
 return null;
}

/**
 * @return A description of the operation.
 */
public String getDescription() {
 return "Inserts an image element. Asks the user for a URL reference.";
}
```

The complete source code of our operation is found in the Example Files Listings, the Java Files section.

Important

Make sure you always specify the namespace of the inserted fragments.

3. Package the compiled class into a jar file. An example of an ANT script that packages the `classes` directory content into a jar archive named `sdf.jar` is listed below:

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="project" default="dist">
    <target name="dist">
   <jar destfile="sdf.jar" basedir="classes">
    <fileset dir="classes">
```

```
     <include name="**/*"/>
     </fileset>
   </jar>
    </target>
</project>
```

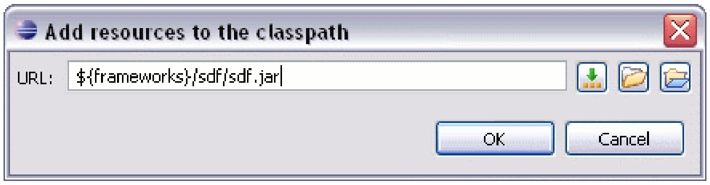
4. Copy the `sdf.jar` file into the `frameworks/sdf` directory.

5. Add the `sdf.jar` to the Author class path. To do this, Open the options Document Type Dialog, select **SDF** and press the Edit button.

   Select the Classpath tab in the lower part of the dialog.

   Press the + Add button . In the displayed dialog enter the location of the jar file, relative to the <oXygen/> `frameworks` directory:

   ### Figure 7.21. Adding a classpath entry

   URL: ${frameworks}/sdf/sdf.jar

6. Let's create now the action which will use the defined operation. Click on the Actions label.

   We assume the icon files `Image16.gif` for the menu item and `Image20.gif` for the toolbar are already available. Place these files in the `frameworks/sdf` directory.

   Define the action properties:

ID	An unique identifier for the action. Use **insert_image**.
Name	The name of the action. Use **Insert image**.
Menu access key	Use the **i** letter.
Description	Enter the text **Inserts an image**.
Toolbar icon	Enter here: **${frameworks}/sdf/Image20.gif**
Menu icon	Enter here: **${frameworks}/sdf/Image16.gif**
Shortcut key	We will use: **Ctrl+Shift+i**.

   Now let's set up the operation.

   We are adding images only if the current element is a `section`, `book` or `article`.

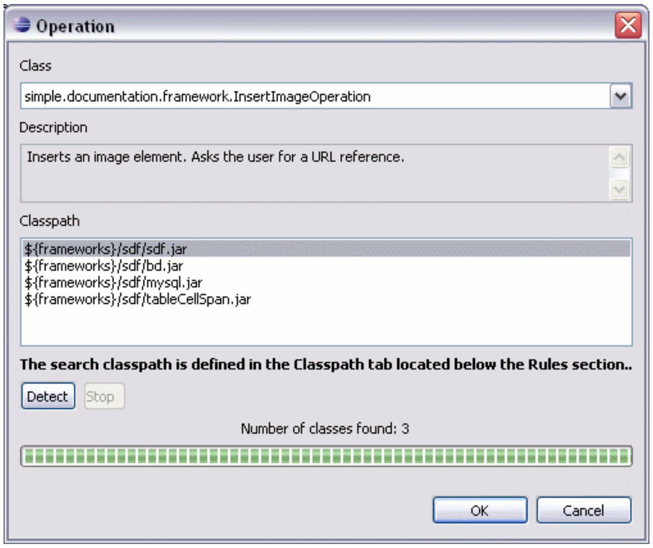
   XPath expression Set the value to:

   ```
   local-name()='section' or local-name='book'
    or local-name='article'
   ```

Invoke operation

In this case, we'll use our Java operation we defined earlier. Press the Choose button, then select `simple.documentation.framework.InsertImageOperation`.

Figure 7.22. Selecting the Operation



Class: simple.documentation.framework.InsertImageOperation

Description: Inserts an image element. Asks the user for a URL reference.

Classpath:

```
${frameworks}/sdf/sdf.jar
${frameworks}/sdf/bd.jar
${frameworks}/sdf/mysql.jar
${frameworks}/sdf/tableCellSpan.jar
```

The search classpath is defined in the Classpath tab located below the Rules section..

Number of classes found: 3

This operation has no arguments.

7. Add the action to the toolbar, using the Toolbar panel.

To test the action, you can open the `sdf.xml` sample, then place the caret inside a `section` between two `para` elements for instance. Press the button associated with the action from the toolbar. In the dialog select an image URL and press Ok. The image is inserted into the document.

Figure 7.23. Dialog Displayed by the Insert Image Operation

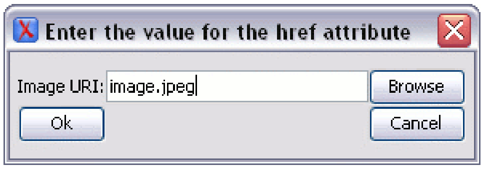


Image URI: image.jpeg

Example 2. Operations with Arguments. Report from Database Operation.

In this example we will create an operation that connects to a relational database and executes an SQL statement. The result should be inserted in the edited XML document as a `table`. To make the operation fully configurable, it will have arguments for the *database connection string*, the *user name*, the *password* and the *SQL expression*.

1. Create a new Java project, in your IDE.

   Create the directory `lib` in the Java project directory and copy in it the `oxygen.jar` file from the `{oXygen_installation_directory}/lib` directory.

2. Create the class `simple.documentation.framework.QueryDatabaseOperation`. This class must implements the `ro.sync.ecss.extensions.api.AuthorOperation` interface.

```
import ro.sync.ecss.extensions.api.ArgumentDescriptor;
import ro.sync.ecss.extensions.api.ArgumentsMap;
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.AuthorOperation;
import ro.sync.ecss.extensions.api.AuthorOperationException;

public class QueryDatabaseOperation implements AuthorOperation{
```

Let's define the arguments of the operation. For each of them we will use a `String` constant representing the argument name:

```
private static final String ARG_JDBC_DRIVER ="jdbc_driver";
private static final String ARG_USER ="user";
private static final String ARG_PASSWORD ="password";
private static final String ARG_SQL ="sql";
private static final String ARG_CONNECTION ="connection";
```

We must describe each of the argument name and type. To do this implement the `getArguments` method which will return an array of argument descriptors:

```
public ArgumentDescriptor[] getArguments() {
  ArgumentDescriptor args[] = new ArgumentDescriptor[] {
    new ArgumentDescriptor(
      ARG_JDBC_DRIVER,
      ArgumentDescriptor.TYPE_STRING,
      "The name of the Java class that is the JDBC driver."),
    new ArgumentDescriptor(
      ARG_CONNECTION,
      ArgumentDescriptor.TYPE_STRING,
      "The database URL connection string."),
    new ArgumentDescriptor(
      ARG_USER,
      ArgumentDescriptor.TYPE_STRING,
      "The name of the database user."),
    new ArgumentDescriptor(
      ARG_PASSWORD,
      ArgumentDescriptor.TYPE_STRING,
      "The database password."),
    new ArgumentDescriptor(
      ARG_SQL,
      ArgumentDescriptor.TYPE_STRING,
      "The SQL statement to be executed.")
  };
  return args;
 }
```

These names, types and descriptions will be listed in the Arguments table when the operation is configured.

When the operation is invoked, the implementation of the `doOperation` method extracts the arguments, forwards them to the method that connects to the database and generates the XML fragment. The XML fragment is then inserted at the caret position.

```
public void doOperation(AuthorAccess authorAccess, ArgumentsMap map)
   throws IllegalArgumentException, AuthorOperationException {
```

```
  // Collects the arguments.
  String jdbcDriver =
   (String)map.getArgumentValue(ARG_JDBC_DRIVER);
  String connection =
   (String)map.getArgumentValue(ARG_CONNECTION);
  String user =
   (String)map.getArgumentValue(ARG_USER);
  String password =
   (String)map.getArgumentValue(ARG_PASSWORD);
  String sql =
   (String)map.getArgumentValue(ARG_SQL);

  int caretPosition = authorAccess.getCaretOffset();
  try {
   authorAccess.insertXMLFragment(
     getFragment(jdbcDriver, connection, user, password, sql),
     caretPosition);
  } catch (SQLException e) {
   throw new AuthorOperationException(
     "The operation failed due to the following database error: "
     + e.getMessage(), e);
  } catch (ClassNotFoundException e) {
   throw new AuthorOperationException(
     "The JDBC database driver was not found. Tried to load ' "
     + jdbcDriver + "'", e);
  }
 }
```

The `getFragment` method loads the JDBC driver, connects to the database and extracts the data. The result is a `table` element from the `http://www.oxygenxml.com/sample/documentation` namespace. The `header` element contains the names of the SQL columns. All the text from the XML fragment is escaped. This means that the '<' and '&' characters are replaced with the '&lt;' and '&amp;' character entities to ensure the fragment is well-formed.

```
private String getFragment(
  String jdbcDriver,
  String connectionURL,
  String user,
  String password,
  String sql) throws
   SQLException,
   ClassNotFoundException {

      Properties pr = new Properties();
      pr.put("characterEncoding", "UTF8");
      pr.put("useUnicode", "TRUE");
      pr.put("user", user);
      pr.put("password", password);

      // Loads the database driver.
      Class.forName(jdbcDriver);
      // Opens the connection
      Connection connection =
         DriverManager.getConnection(connectionURL, pr);
      java.sql.Statement statement =
         connection.createStatement();
      ResultSet resultSet =
         statement.executeQuery(sql);

      StringBuffer fragmentBuffer = new StringBuffer();
      fragmentBuffer.append(
        "<table xmlns=" +
        "'http://www.oxygenxml.com/sample/documentation'>");

      //
      // Creates the table header.
      //
      fragmentBuffer.append("<header>");
      ResultSetMetaData metaData = resultSet.getMetaData();
      int columnCount = metaData.getColumnCount();
      for (int i = 1; i <= columnCount; i++) {
          fragmentBuffer.append("<td>");
          fragmentBuffer.append(
            xmlEscape(metaData.getColumnName(i)));
          fragmentBuffer.append("</td>");
      }
      fragmentBuffer.append("</header>");

      //
      // Creates the table content.
      //
      while (resultSet.next()) {
          fragmentBuffer.append("<tr>");
          for (int i = 1; i <= columnCount; i++) {
              fragmentBuffer.append("<td>");
              fragmentBuffer.append(
                xmlEscape(resultSet.getObject(i)));
              fragmentBuffer.append("</td>");
          }
          fragmentBuffer.append("</tr>");
      }

      fragmentBuffer.append("</table>");

      // Cleanup
      resultSet.close();
      statement.close();
      connection.close();
      return fragmentBuffer.toString();
}
```

The complete source code of our operation is found in the Example Files Listings, the Java Files section.

3. Package the compiled class into a jar file.

4. Copy the jar file and the JDBC driver files into the `frameworks/sdf` directory.

5. Add the jars to the Author class path. For this, Open the options Document Type Dialog, select **SDF** and press the Edit button.

   Select the Classpath tab in the lower part of the dialog.

6. Click on the Actions label.

   The action properties are:

   ID

   An unique identifier for the action. Use **clients_report**.

   Name

   The name of the action. Use **Clients Report**.

   Menu access key

   Use the letter **r**.

   Description

   Enter the text **Connects to the database and collects the list of clients**.

   Toolbar icon

   Enter here: **${frameworks}/sdf/TableDB20.gif**

   We assume the image `TableDB20.gif` for the toolbar action is already present in the `frameworks/sdf` directory.

   Menu icon

   Leave empty.

   Shortcut key

   We will use: **Ctrl+Shift+c**.

   Let's set up the operation. The action will work only if the current element is a `section`.

   XPath expression

   Set the value to:

   ```
   local-name()='section'
   ```

   Invoke operation

   In this case, we'll use our Java operation we defined earlier. Press the Choose button, then select `simple.documentation.framework.QueryDatabaseOperation`.

   Once selected, the list of arguments is displayed.

   In the figure below the first argument, *jdbc_driver*, represents the class name of the MySQL JDBC driver.

   The connection string has the URL syntax : *jdbc://<database_host>:<database_port>/<database_name>*.

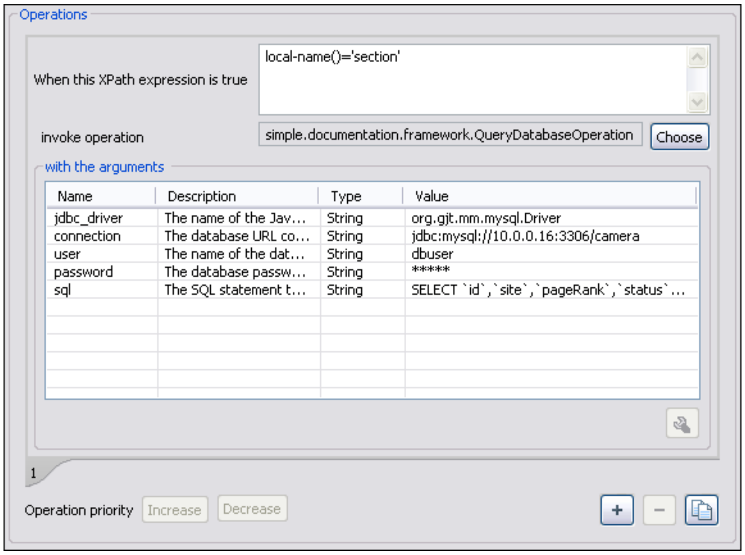
   The SQL expression used in the example is:

   ```
   SELECT userID, email FROM users
   ```

   but it can be any valid SELECT expression which can be applied to the database.

7. Add the action to the toolbar, using the Toolbar panel.

Figure 7.24. Java Operation Arguments Setup



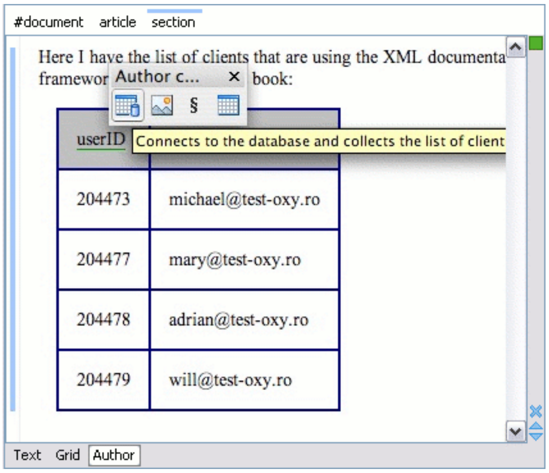
When this XPath expression is true: local-name()='section'

invoke operation: simple.documentation.framework.QueryDatabaseOperation

Name	Description	Type	Value
jdbc_driver	The name of the Jav...	String	org.gjt.mm.mysql.Driver
connection	The database URL co...	String	jdbc:mysql://10.0.0.16:3306/camera
user	The name of the dat...	String	dbuser
password	The database passw...	String	*****
sql	The SQL statement t...	String	SELECT `id`,`site`,`pageRank`,`status`...

To test the action you can open the `sdf.xml` sample place the caret inside a `section` between two `para` elements for instance. Press the Create Report button from the toolbar. You can see below the toolbar with the action button and sample table inserted by the Clients Report action.

Figure 7.25. Table Content Extracted from the Database



Here I have the list of clients that are using the XML documenta
framewor book:

Connects to the database and collects the list of client

userID	
204473	michael@test-oxy.ro
204477	mary@test-oxy.ro
204478	adrian@test-oxy.ro
204479	will@test-oxy.ro

Configuring New File Templates

We will create a set of document templates that the content authors will use as starting points for creating new *Simple Document Framework* books and articles.

Each of the Document Type Associations can point to a directory usually named `templates` containing the file templates. All the files that are found here are considered templates for the respective document type. The template name is taken from the name of the file, and the template kind is detected from the file extension.

Create the `templates` directory into the `frameworks/SDF` directory. The directory tree for our documentation framework is now:

```
oxygen
  frameworks
     sdf
       schema
       css
       templates
```

Now let's create in this `templates` directory two files, one for the *book* template and another for the *article* template.

The `Book.xml` file:

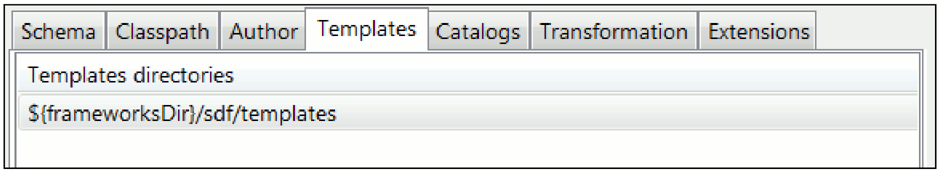
```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.oxygenxml.com/sample/documentation"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts">
    <title>Book Template Title</title>
    <section>
        <title>Section Title</title>
        <abs:def/>
        <para>This content is copyrighted:</para>
        <table>
            <header>
                <td>Company</td>
                <td>Date</td>
            </header>
            <tr>
                <td/>
                <td/>
            </tr>
        </table>
    </section>
</book>
```

The `Article.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<article
    xmlns="http://www.oxygenxml.com/sample/documentation"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <title></title>
    <section>
        <title></title>
        <para></para>
        <para></para>
    </section>
</article>
```

Open the Document Type dialog for the **SDF** framework and click on the Templates tab. Enter in the Templates directory text field the value `${frameworksDir}/sdf/templates`. As we already seen before, it is recommended that all the file references made from a Document Type Association to be relative to the `${frameworksDir}` directory. Binding a Document Type Association to an absolute file (e.g: "C:\some_dir\templates") makes the association difficult to share between users.

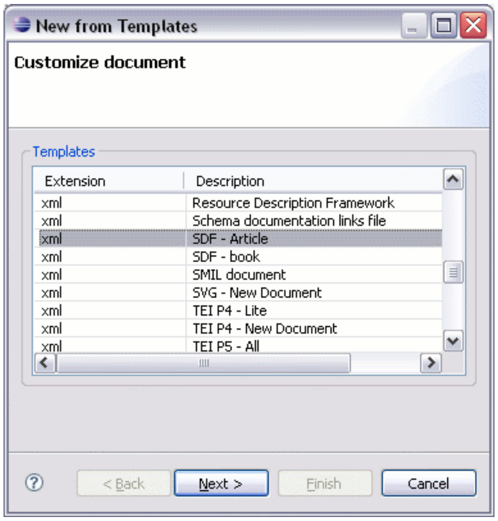
Figure 7.26. Setting the templates directory



Templates directories
${frameworksDir}/sdf/templates

To test the templates settings, press the File/New menu item to display the New dialog. The names of the two templates are prefixed with the name of the Document Type Association, in our case **SDF**. Selecting one of them should create a new XML file with the content specified in the template file.

Figure 7.27. Templates displayed in the New Dialog.



Extension	Description
xml	Resource Description Framework
xml	Schema documentation links file
xml	SDF - Article
xml	SDF - book
xml	SMIL document
xml	SVG - New Document
xml	TEI P4 - Lite
xml	TEI P4 - New Document
xml	TEI P5 - All

Configuring XML Catalogs

You can add catalog files to your Document Type Association using the Catalogs tab from the Document Type dialog.

Important

<oXygen/> XML Editor collects all the catalog files listed in the installed frameworks. No matter what the Document Type Association matches the edited file, all the catalog mappings are considered when resolving external references.

Important

The catalog files settings are available for all editing modes, not only for the **Author** mode.

In the XML sample file for **SDF** we did not used a `xsi:schemaLocation` attribute, but instead we let the editor use the schema from the association. However there are cases in which we must refer for instance the location of a schema file from a remote web location. In such cases the catalog may be used to map the web location to a local file system entry.

In the following section we will present an use-case for the XML catalogs, by modifying our `sdf.xsd` XML Schema file from the Example Files Listings.

The `sdf.xml` file refers the other file `abs.xsd` through an `import` element:

```
<xs:import namespace=
 "http://www.oxygenxml.com/sample/documentation/abstracts"
 schemaLocation="abs.xsd"/>
```

The `schemaLocation` attribute references the `abs.xsd` file located in the same directory. What if the file was on the web, at the `http://www.oxygenxml.com/SDF/abs.xsd` location for instance? In this case the attribute value will be:

```
<xs:import namespace=
 "http://www.oxygenxml.com/sample/documentation/abstracts"
 schemaLocation="http://www.oxygenxml.com/SDF/abs.xsd"/>
```

There is a problem with this approach. What happens if an Internet connection is not available? How will we check our document for errors if a part of the schema is not available? The answer is to create a catalog file that will help the parser locate the missing piece containing the mapping:

```
http://www.oxygenxml.com/SDF/abs.xsd -> ../local_path/abs.xsd
```

To do this create a new XML file called `catalog.xml` and save it into the `{oXygen_installation_directory}/frameworks/sdf` directory. The content of the file should be:

```
<?xml version="1.0"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
    <system
      systemId="http://www.oxygenxml.com/SDF/abs.xsd"
      uri="schema/abs.xsd"/>
</catalog>
```

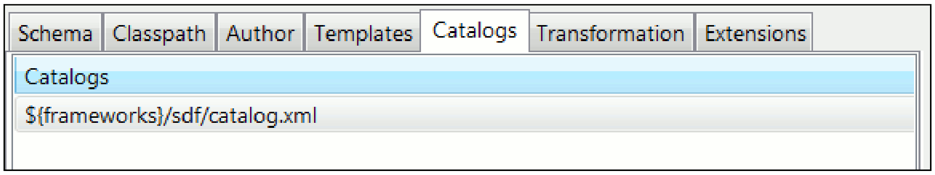
This means that all the references to `http://www.oxygenxml.com/SDF/abs.xsd` must be resolved to the `abs.xsd` file located in the `schema` directory.

Note

The references in the XML catalog files are relative to the directory that contains the catalog.

Save the catalog file and modify the `sdf.xsd` file by changing its `import` element, then add the catalog to the Document Type association. You can do this in the **Catalogs** tab by pressing the New button. Enter `${frameworks}/sdf/catalog.xml` in the displayed dialog.

Figure 7.28. Adding Catalogs to the Document Type Association



Catalogs

${frameworks}/sdf/catalog.xml

To test the catalog settings, restart <oXygen/> and try to validate a new sample **Simple Documentation Framework** document. There should be no errors.

Configuring Transformation Scenarios

When distributing a framework to the users, it is a good idea to have the transformation scenarios already configured. This would help the content authors publish their work in different formats. Being contained in the **Document Type Association** the scenarios can be distributed along with the actions, menus, toolbars, catalogs, etc.

In the following section we will create a transformation scenario for our framework.

Create the directory `xsl` in the directory `frameworks/sdf`. The directory structure for our documentation framework should be:

```
oxygen
  frameworks
     sdf
       schema
       css
       templates
       xsl
```

Create the `sdf.xsl` file in the `xsl` directory. The complete content of the `sdf.xsl` file is found in the Example Files Listings.

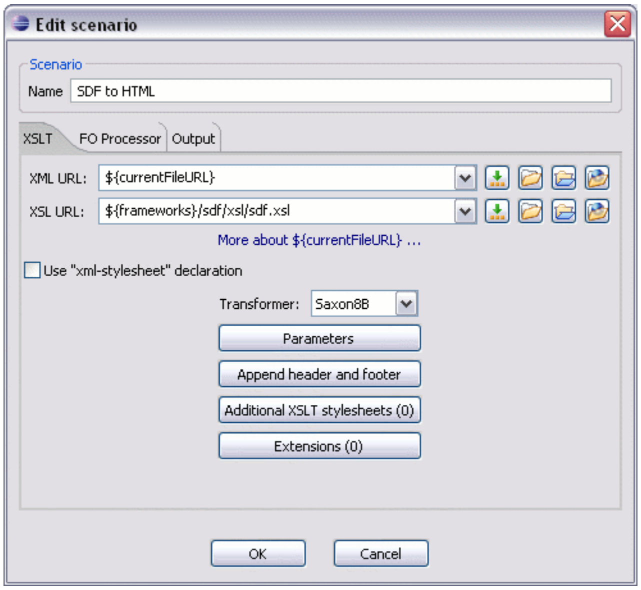
Open the Options/Preferences/Document Type Associations. Open the Document Type dialog for the **SDF** framework then choose the Transformation tab. Click on the New. In the Edit Scenario dialog, fill the following fields:

Name	The name of the transformation scenario. Enter *SDF to HTML*.
XSL URL	`${frameworks}/sdf/xsl/sdf.xsl`
Transformer	Saxon 9B.

Change to the Output tab. Change the fields:

Save as	`${cfd}/${cfn}.html` This means the transformation output file will have the name of the XML file and the *html* extension and will be placed in the same directory.
Open in browser	Enable this option.
Saved file	Enable this checkbox.

Figure 7.29. Configuring a transformation scenario



Edit scenario

Scenario

Name: SDF to HTML

XML URL: ${currentFileURL}

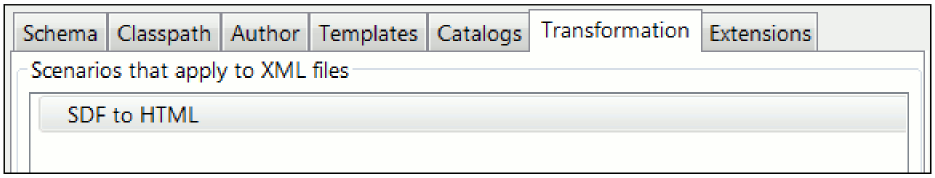
XSL URL: ${frameworks}/sdf/xsl/sdf.xsl

☐ Use "xml-stylesheet" declaration

Transformer: Saxon8B

Now the scenario is listed in the Transformation tab:

Figure 7.30. The transformation tab

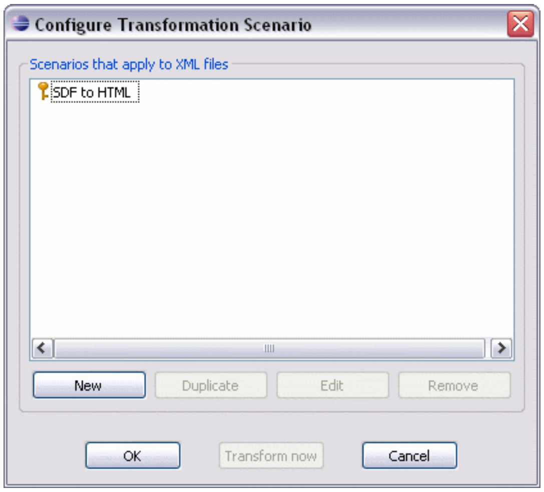


Scenarios that apply to XML files

SDF to HTML

To test the transformation scenario we created, open the **SDF** XML sample from the Example Files Listings. Click on the ▶Apply Transformation Scenario button. The Configure Transformation Dialog is displayed. Its scenario list contains the scenario we defined earlier *SDF to HTML*. Click on it then choose Transform now. The HTML file should be saved in the same directory as the XML file and opened in the browser.

Figure 7.31. Selecting the predefined scenario


Configure Transformation Scenario
Scenarios that apply to XML files
New
Duplicate
Edit
Remove
OK
Transform now
Cancel

🔑 SDF to HTML

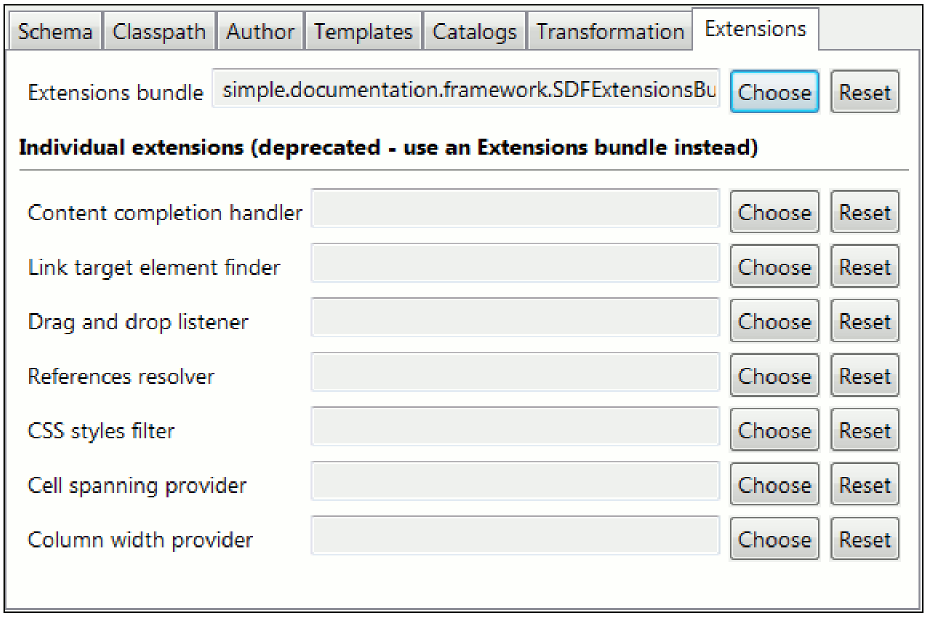
Note

The key 🔑 symbol indicates that the scenario is read-only. It has this state because the scenario was loaded from a Document Type Association. The content authors can still change parameters and other settings if they are duplicating the scenario and edit the duplicate. In this case the copy of the scenario is created in the user local settings.

Configuring Extensions

You can add extensions to your Document Type Association using the Extensions tab from the Document Type dialog.

Figure 7.32. Configure extensions for a document type



Extensions bundle: simple.documentation.framework.SDFExtensionsBu

Configuring an Extensions Bundle

Starting with <oXygen/> 10.3 version a single bundle was introduced acting as a provider for all other extensions. The individual extensions can still be set but this practice is being discouraged and the single provider should be used instead.

The extensions bundle is represented by the `ro.sync.ecss.extensions.api.ExtensionsBundle` class. The provided implementation of the `ExtensionsBundle` is instantiated when the rules of the Document Type Association defined for the custom framework match a document opened in the editor. Therefor references to objects which need to be persistent throughout the application running session must not be kept in the bundle because the next detection event can result in creating another `ExtensionsBundle` instance.

1. Create a new Java project, in your IDE.

   Create the `lib` directory in the Java project directory and copy in it the `oxygen.jar` file from the `{oXygen_installation_directory}/lib` directory.

2. Create the class `simple.documentation.framework.SDFExtensionsBundle` which must extend the abstract class `ro.sync.ecss.extensions.api.ExtensionsBundle`.

   ```
   public class SDFExtensionsBundle extends ExtensionsBundle {
   ```

   A **Document Type ID** and a short description should be defined first by implementing the methods `getDocumentTypeID` and `getDescription`. The Document Type ID is used to uniquely identify the current framework. Such an ID must be provided especially if options related to the framework need to be persistently stored and retrieved between sessions.

   ```
    public String getDocumentTypeID() {
         return "Simple.Document.Framework.document.type";
    }

    public String getDescription() {
         return "A custom extensions bundle used for the Simple Document" +
                       "Framework document type";
    }
   ```

   In order to be notified about the activation of the custom Author extension in relation with an opened document an `ro.sync.ecss.extensions.api.AuthorExtensionStateListener` should be implemented. The **activation** and **deactivation** events received by this listener should be used to perform custom initializations and to register/remove listeners like `ro.sync.ecss.extensions.api.AuthorListener`, `ro.sync.ecss.extensions.api.AuthorMouseListener` or `ro.sync.ecss.extensions.api.AuthorCaretListener`. The custom author extension state listener should be provided by implementing the method `createAuthorExtensionStateListener`.

   ```
    public AuthorExtensionStateListener createAuthorExtensionStateListener() {
         return new SDFAuthorExtensionStateListener();
    }
   ```

   The `AuthorExtensionStateListener` is instantiated and notified about the activation of the framework when the rules of the Document Type Association match a document opened in the Author editor page. The listener is notified about the deactivation when another framework is activated for the same document, the user switches to another page or the editor is closed. A complete description and implementation of an `ro.sync.ecss.extensions.api.AuthorExtensionStateListener` can be found in the Implementing an Author Extension State Listener.

Customizations of the content completion proposals are permitted by creating a schema manager filter extension. The interface that declares the methods used for content completion proposals filtering is `ro.sync.contentcompletion.xml.SchemaManagerFilter`. The filter can be applied on elements, attributes or on their values. Responsible for creating the content completion filter is the method `createSchemaManagerFilter`. A new `SchemaManagerFilter` will be created each time a document matches the rules defined by the Document Type Association which contains the filter declaration.

```
public SchemaManagerFilter createSchemaManagerFilter() {
     return new SDFSchemaManagerFilter();
}
```

A detailed presentation of the schema manager filter can be found in Configuring a Content completion handler section.

The <oXygen/> Author supports link based navigation between documents and document sections. Therefor, if the document contains elements defined as links to other elements, for example links based on the **id** attributes, the extension should provide the means to find the referred content. To do this an implementation of the `ro.sync.ecss.extensions.api.link.ElementLocatorProvider` interface should be returned by the `createElementLocatorProvider` method. Each time an element pointed by a link needs to be located the method is invoked.

```
public ElementLocatorProvider createElementLocatorProvider() {
     return new DefaultElementLocatorProvider();
}
```

The section that explains how to implement an element locator provider is Configuring a Link target element finder.

The drag and drop functionality can be extended by implementing the `ro.sync.exml.editor.xmleditor.pageauthor.AuthorDnDListener` interface. Relevant methods from the listener are invoked when the mouse is dragged, moved over, or exits the author editor page, when the drop action changes, and when the drop occurs. Each method receives the `DropTargetEvent` containing information about the drag and drop operation. The drag and drop extensions are available on Author page for both <oXygen/> Eclipse plugin and standalone application. The Text page corresponding listener is available only for <oXygen/> Eclipse plugin. The methods corresponding to each implementation are: `createAuthorAWTDndListener`, `createTextSWTDndListener` and `createAuthorSWTDndListener`.

```
public AuthorDnDListener createAuthorAWTDndListener() {
     return new SDFAuthorDndListener();
}
```

For more details about the Author drag and drop listeners see the Configuring a custom Drag and Drop listener section.

Another extension which can be included in the bundle is the reference resolver. In our case the references re represented by the **ref** element and the attribute indicating the referred resource is **location**. To be able to obtain the content of the referred resources we will have to implement a Java extension class which implements the `ro.sync.ecss.extensions.api.AuthorReferenceResolver`. The method responsible for creating the custom references resolver is `createAuthorReferenceResolver`. The method is called each time a document opened in an Author editor page matches the Document Type Association where the extensions bundle is defined. The instantiated references resolver object is kept and used until another extensions bundle corresponding to another Document Type is activated as result of the detection process.

```
public AuthorReferenceResolver createAuthorReferenceResolver() {
     return new ReferencesResolver();
}
```

A more detailed description of the references resolver can be found in the Configuring a References Resolver section.

To be able to dynamically customize the default CSS styles for a certain `AuthorNode` an implementation of the `ro.sync.ecss.extensions.api.StylesFilter` can be provided. The extensions bundle method responsible for creating the `StylesFilter` is `createAuthorStylesFilter`. The method is called each time a document opened in an Author editor page matches the document type association where the extensions bundle is defined. The instantiated filter object is kept and used until another extensions bundle corresponding to another Document Type is activated as result of the detection process.

```
public StylesFilter createAuthorStylesFilter() {
     return new SDFStylesFilter();
}
```

See the Configuring CSS styles filter section for more details about the styles filter extension.

In order to edit data in custom tabular format implementations of the `ro.sync.ecss.extensions.api.AuthorTableCellSpanProvider` and `the ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider` interfaces should be provided. The two methods from the `ExtensionsBundle` specifying these two extension points are `createAuthorTableCellSpanProvider` and `createAuthorTableColumnWidthProvider`.

```
public AuthorTableCellSpanProvider createAuthorTableCellSpanProvider() {
     return new TableCellSpanProvider();
}

public AuthorTableColumnWidthProvider
         createAuthorTableColumnWidthProvider() {
     return new TableColumnWidthProvider();
}
```

The two table information providers are not reused for different tables. The methods are called for each table in the document so new instances should be provided every time. Read more about the cell span and column width information providers in Configuring a Table Cell Span Provider and Configuring a Table Column Width Provider sections.

If the functionality related to one of the previous extension point does not need to be modified then the developed `ExtensionsBundle` should not override the corresponding method and leave the default base implementation to return **null**.

3. Package the compiled class into a jar file.
4. Copy the jar file into the `frameworks/sdf` directory.
5. Add the jar file to the Author class path.
6. Register the Java class by clicking on the Extensions tab. Press the Choose button and select from the displayed dialog the name of the class: `SDFExtensionsBundle`.

The complete source code of the `SDFExtensionsBundle` implementation is found in the Example Files Listings, the Java Files section.

Implementing an Author Extension State Listener

The `ro.sync.ecss.extensions.api.AuthorExtensionStateListener` implementation is notified when the Author extension where the listener is defined is activated or deactivated in the Document Type detection process.

```
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.AuthorExtensionStateListener;

public class SDFAuthorExtensionStateListener implements
  AuthorExtensionStateListener {
  private AuthorListener sdfAuthorDocumentListener;
  private AuthorMouseListener sdfMouseListener;
  private AuthorCaretListener sdfCaretListener;
  private OptionListener sdfOptionListener;
```

The **activation** event received by this listener when the rules of the Document Type Association match a document opened in the Author editor page, should be used to perform custom initializations and to register listeners like `ro.sync.ecss.extensions.api.AuthorListener`, `ro.sync.ecss.extensions.api.AuthorMouseListener` or `ro.sync.ecss.extensions.api.AuthorCaretListener`.

```
 public void activated(AuthorAccess authorAccess) {
   // Get the value of the option.
   String option = authorAccess.getOptionsStorage().getOption(
               "sdf.custom.option.key", "");
   // Use the option for some initializations...

   // Add an option listener.
   authorAccess.getOptionsStorage().addOptionListener(sdfOptionListener);

   // Add author document listeners.
   sdfAuthorDocumentListener = new SDFAuthorListener();
   authorAccess.getDocumentController().addAuthorListener(
               sdfAuthorDocumentListener);

   // Add mouse listener.
   sdfMouseListener = new SDFAuthorMouseListener();
   authorAccess.getEditorAccess().addAuthorMouseListener(sdfMouseListener);

   // Add caret listener.
   sdfCaretListener = new SDFAuthorCaretListener();
   authorAccess.getEditorAccess().addAuthorCaretListener(sdfCaretListener);

   // Other custom initializations...

 }
```

The *`authorAccess`* parameter received by the `activated` method can be used to gain access to Author specific actions and informations related to components like the editor, document, workspace, tables, change tracking a.s.o.

If options specific to the custom developed Author extension need to be stored or retrieved, a reference to the `OptionsStorage` can be obtained by calling the `getOptionsStorage` method from the author access. The same object can be used to register `OptionListener` listeners. An option listener is registered in relation with an option **key** and will be notified about the value changes of that option.

An AuthorListener can be used if events related to the Author document modifications are of interest. The listener can be added to the AuthorDocumentController. A reference to the document controller is returned by the getDocumentController method from the author access. The document controller can also be used to perform operations involving document modifications.

To provide access to Author editor component related functionality and informations, the author access has a reference to the AuthorEditorAccess that can be obtained when calling the getEditorAccess method. At this level AuthorMouseListener and AuthorCaretListener can be added which will be notified about mouse and caret events occurring in the Author editor page.

The **deactivation** event is received when another framework is activated for the same document, the user switches to another editor page or the editor is closed. The deactivate method is typically used to unregister the listeners previously added on the activate method and to perform other actions. For example options related to the deactivated author extension can be saved at this point.

```
 public void deactivated(AuthorAccess authorAccess) {
   // Store the option.
   authorAccess.getOptionsStorage().setOption(
               "sdf.custom.option.key", optionValue);

   // Remove the option listener.
   authorAccess.getOptionsStorage().removeOptionListener(sdfOptionListener);

   // Remove document listeners.
   authorAccess.getDocumentController().removeAuthorListener(
               sdfAuthorDocumentListener);

   // Remove mouse listener.
   authorAccess.getEditorAccess().removeAuthorMouseListener(sdfMouseListener);

   // Remove caret listener.
   authorAccess.getEditorAccess().removeAuthorCaretListener(sdfCaretListener);

   // Other actions...

 }
```

Configuring a Content completion handler

You can filter or contribute to items offered for content completion by implementing the ro.sync.contentcompletion.xml.SchemaManagerFilter interface.

```
import java.util.List;

import ro.sync.contentcompletion.xml.CIAttribute;
import ro.sync.contentcompletion.xml.CIElement;
import ro.sync.contentcompletion.xml.CIValue;
import ro.sync.contentcompletion.xml.Context;
import ro.sync.contentcompletion.xml.SchemaManagerFilter;
import ro.sync.contentcompletion.xml.WhatAttributesCanGoHereContext;
import ro.sync.contentcompletion.xml.WhatElementsCanGoHereContext;
import ro.sync.contentcompletion.xml.WhatPossibleValuesHasAttributeContext;

public class SDFSchemaManagerFilter implements SchemaManagerFilter {
```

You can implement the various callbacks of the interface either by returning the default values given by <oXygen/> or by contributing to the list of proposals. The filter can be applied on elements, attributes or on their values. Attributes filtering can be implemented using the `filterAttributes` method and changing the default content completion list of `ro.sync.contentcompletion.xml.CIAttribute` for the element provided by the current `ro.sync.contentcompletion.xml.WhatAttributesCanGoHereContext` context. For example, the `SDFSchemaManagerFilter` checks if the element from the current context is the `table` element and add the `frame` attribute to the `table` list of attributes.

```
 public List<CIAttribute> filterAttributes(List<CIAttribute> attributes,
       WhatAttributesCanGoHereContext context) {
   // If the element from the current context is the 'table' element add the
   // attribute named 'frame' to the list of default content
      // completion proposals
   ContextElement contextElement = context.getParentElement();
   if ("table".equals(contextElement.getQName())) {
     CIAttribute frameAttribute = new CIAttribute();
     frameAttribute.setName("frame");
     frameAttribute.setRequired(false);
     frameAttribute.setFixed(false);
     frameAttribute.setDefaultValue("void");
     attributes.add(frameAttribute);
   }
   return attributes;
 }
```

The elements that can be inserted in a specific context can be filtered using the `filterElements` method. The `SDFSchemaManagerFilter` uses this method to replace the `td` child element with the `th` element when `header` is the current context element.

```
public List<CIElement> filterElements(List<CIElement> elements,
   WhatElementsCanGoHereContext context) {
   // If the element from the current context is the 'header' element remove
   // the 'td' element from the list of content completion proposals and add
   // the 'th' element.
   ContextElement contextElement = context.getElementStack().peek();
   if ("header".equals(contextElement.getQName())) {
     for (Iterator<CIElement> iterator = elements.iterator();
               iterator.hasNext();) {
       CIElement element = iterator.next();
        // Remove the 'td' element
       if ("td".equals(element.getQName())) {
            elements.remove(element);
            break;
          }
        }
     // Insert the 'th' element in the list of content completion proposals
     CIElement thElement = new SDFElement();
     thElement.setName("th");
     elements.add(thElement);
   }
  return elements;
 }
```

The elements or attributes values can be filtered using the filterElementValues or filterAttributeValues methods.

The complete source code of the SDFSchemaManagerFilter implementation is found in the Example Files Listings, the Java Files section.

Configuring a Link target element finder

The link target reference finder represents the support for finding references from links which indicate specific elements inside an XML document. This support will only be available if a schema is associated with the document type.

If you do not define a custom link target reference finder, the DefaultElementLocatorProvider implementation will be used by default. The interface which should be implemented for a custom link target reference finder is ro.sync.ecss.extensions.api.link.ElementLocatorProvider. As an alternative, the ro.sync.ecss.extensions.commons.DefaultElementLocatorProvider implementation can also be extended.

The used ElementLocatorProvider will be queried for an ElementLocator when a link location must be determined (when user clicks on a link). Then, to find the corresponding (linked) element, the obtained ElementLocator will be queried for each element from the document.

The DefaultElementLocatorProvider implementation

The DefaultElementLocatorProvider implementation offers support for the most common types of links:

- links based on ID attribute values
- XPointer element() scheme

The method getElementLocator determines what ElementLocator should be used. In the default implementation it checks if the link is an XPointer element() scheme otherwise it assumes it is an ID. A non-null IDTypeVerifier will always be provided if a schema is associated with the document type.

The link string argument is the "anchor" part of the of the URL which is composed from the value of the link property specified for the link element in the CSS.

```
public ElementLocator getElementLocator(IDTypeVerifier idVerifier,
               String link) {
  ElementLocator elementLocator = null;
  try {
    if(link.startsWith("element(")){
      // xpointer element() scheme
      elementLocator = new XPointerElementLocator(idVerifier, link);
    } else {
      // Locate link element by ID
      elementLocator = new IDElementLocator(idVerifier, link);
    }
  } catch (ElementLocatorException e) {
    logger.warn("Exception when create element locator for link: "
        + link + ". Cause: " + e, e);
  }
  return elementLocator;
}
```

The `XPointerElementLocator` implementation

The `XPointerElementLocator` is an implementation of the abstract class `ro.sync.ecss.extensions.api.link.ElementLocator` for links that have one of the following XPointer element() scheme patterns:

element(`elementID`) locate the element with the specified id

element(`/1/2/3`) A child sequence appearing alone identifies an element by means of stepwise navigation, which is directed by a sequence of integers separated by slashes (/); each integer n locates the nth child element of the previously located element.

element(`elementID/3/4`) A child sequence appearing after a `NCName` identifies an element by means of stepwise navigation, starting from the element located by the given name.

The constructor separates the id/integers which are delimited by slashes(/) into a sequence of identifiers (an XPointer path). It will also check that the link has one of the supported patterns of the XPointer element() scheme.

```
public XPointerElementLocator(IDTypeVerifier idVerifier, String link)
                      throws ElementLocatorException {
  super(link);
  this.idVerifier = idVerifier;

  link = link.substring("element(".length(), link.length() - 1);

  StringTokenizer stringTokenizer = new StringTokenizer(link, "/", false);
  xpointerPath = new String[stringTokenizer.countTokens()];
  int i = 0;
  while (stringTokenizer.hasMoreTokens()) {
    xpointerPath[i] = stringTokenizer.nextToken();
    boolean invalidFormat = false;

    // Empty xpointer component is not supported
    if(xpointerPath[i].length() == 0){
      invalidFormat = true;
    }

    if(i > 0){
      try {
        Integer.parseInt(xpointerPath[i]);
      } catch (NumberFormatException e) {
        invalidFormat = true;
      }
    }

    if(invalidFormat){
      throw new ElementLocatorException(
        "Only the element() scheme is supported when locating XPointer links."
        + "Supported formats: element(elementID), element(/1/2/3),
              element(elemID/2/3/4).");
    }
    i++;
  }

  if(Character.isDigit(xpointerPath[0].charAt(0))){
    // This is the case when xpointer have the following pattern /1/5/7
    xpointerPathDepth = xpointerPath.length;
  } else {
    // This is the case when xpointer starts with an element ID
    xpointerPathDepth = -1;
    startWithElementID  = true;
  }
}
```

The method `startElement` will be invoked at the beginning of every element in the XML document(even when the element is empty). The arguments it takes are

`uri`	the namespace URI, or the empty string if the element has no namespace URI or if namespace processing is disabled
`localName`	the local name of the element
`qName`	the qualified name of the element
`atts`	the attributes attached to the element. If there are no attributes, it will be empty.

The method returns `true` if the processed element is found to be the one indicated by the link.

The `XPointerElementLocator` implementation of the `startElement` will update the depth of the current element and keep the index of the element in its parent. If the `xpointerPath` starts with an element ID then the current element ID is verified to match the specified ID. If this is the case the depth of the XPointer is updated taking account of the depth of the current element.

If the XPointer path depth is the same as the current element depth then the kept indices of the current element path are compared to the indices in the XPointer path. If all of them match then the element has been found.

```
public boolean startElement(String uri, String localName,
        String name, Attr[] atts) {
  boolean linkLocated = false;
  // Increase current element document depth
  startElementDepth ++;

  if (endElementDepth != startElementDepth) {
    // The current element is the first child of the parent
    currentElementIndexStack.push(new Integer(1));
  } else {
    // Another element in the parent element
    currentElementIndexStack.push(new Integer(lastIndexInParent + 1));
  }

  if (startWithElementID) {
    // This the case when xpointer path starts with an element ID.
    String xpointerElement = xpointerPath[0];
    for (int i = 0; i < atts.length; i++) {
      if(xpointerElement.equals(atts[i].getValue())){
        if(idVerifier.hasIDType(
            localName, uri, atts[i].getQName(), atts[i].getNamespace())){
          xpointerPathDepth = startElementDepth + xpointerPath.length - 1;
          break;
        }
      }
    }
  }

  if (xpointerPathDepth == startElementDepth){
    // check if xpointer path matches with the current element path
    linkLocated = true;
    try {
      int xpointerIdx = xpointerPath.length - 1;
      int stackIdx = currentElementIndexStack.size() - 1;
      int stopIdx = startWithElementID ? 1 : 0;
      while (xpointerIdx >= stopIdx && stackIdx >= 0) {
        int xpointerIndex = Integer.parseInt(xpointerPath[xpointerIdx]);
        int currentElementIndex =
          ((Integer)currentElementIndexStack.get(stackIdx)).intValue();
        if(xpointerIndex != currentElementIndex) {
          linkLocated = false;
          break;
        }

        xpointerIdx--;
        stackIdx--;
      }

    } catch (NumberFormatException e) {
      logger.warn(e,e);
    }
  }
  return linkLocated;
}
```

The method `endElement` will be invoked at the end of every element in the XML document (even when the element is empty).

The `XPointerElementLocator` implementation of the `endElement` updates the depth of the current element path and the index of the element in its parent.

```
public void endElement(String uri, String localName, String name) {
  endElementDepth = startElementDepth;
  startElementDepth --;
  lastIndexInParent = ((Integer)currentElementIndexStack.pop()).intValue();
}
```

The `IDElementLocator` implementation

The `IDElementLocator` is an implementation of the abstract class `ro.sync.ecss.extensions.api.link.ElementLocator` for links that use an **id**.

The constructor only assigns field values and the method `endElement` is empty for this implementation.

The method `startElement` checks each of the element's attribute values and when one matches the link, it considers the element found if one of the following conditions is satisfied:

- the qualified name of the attribute is `xml:id`

- the attribute is of type ID

The type of the attribute is checked with the help of the method IDTypeVerifier.hasIDType.

```
public boolean startElement(String uri, String localName,
        String name, Attr[] atts) {
  boolean elementFound = false;
  for (int i = 0; i < atts.length; i++) {
    if (link.equals(atts[i].getValue())) {
      if("xml:id".equals(atts[i].getQName())) {
        // xml:id attribute
        elementFound = true;
      } else {
        // check if attribute has ID type
        String attrLocalName =
          ExtensionUtil.getLocalName(atts[i].getQName());
        String attrUri = atts[i].getNamespace();
        if (idVerifier.hasIDType(localName, uri, attrLocalName, attrUri)) {
          elementFound = true;
        }
      }
    }
  }

  return elementFound;
}
```

Creating a customized link target reference finder

If you need to create a custom link target reference finder you can do so by following these steps.

Create the class which will implement the ro.sync.ecss.extensions.api.link.ElementLocatorProvider interface. As an alternative, your class could extend ro.sync.ecss.extensions.commons.DefaultElementLocatorProvider, the default implementation.

As a start point you can use the source code of the DefaultElementLocatorProvider implementation which is found in the Example Files Listings, the Java Files section. There you will also find the implementations for XPointerElementLocator and IDElementLocator .

Configuring a custom Drag and Drop listener

You can add your own drag and drop listener implementation of ro.sync.ecss.extensions.api.DnDHandler. You can choose from three interfaces to implement depending on whether you are using the framework with the <oXygen/> Eclipse plugin or the standalone version or if you want to add the handler for the Text or Author pages.

Table 7.2. Interfaces for the DnD listener

Interface	Description
`ro.sync.exml.editor.xmleditor.pageauthor.AuthorCustomDnDHandler`	Receives callbacks from the <oXygen/> standalone application for Drag And Drop in Author
`com.oxygenxml.editor.editors.author.AuthorDnDListener`	Receives callbacks from the <oXygen/> Eclipse plugin for Drag And Drop in Author
`com.oxygenxml.editor.editors.TextDnDListener`	Receives callbacks from the <oXygen/> Eclipse plugin for Drag And Drop in Text

Configuring a References Resolver

We need to provide a handler for resolving references and obtain the content they refer. In our case the element which has references is **ref** and the attribute indicating the referred resource is **location**. We will have to implement a Java extension class for obtaining the referred resources.

Create the class `simple.documentation.framework.ReferencesResolver`. This class must implement the `ro.sync.ecss.extensions.api.AuthorReferenceResolver` interface.

```
import ro.sync.ecss.extensions.api.AuthorReferenceResolver;
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.node.AttrValue;
import ro.sync.ecss.extensions.api.node.AuthorElement;
import ro.sync.ecss.extensions.api.node.AuthorNode;

public class ReferencesResolver
      implements AuthorReferenceResolver {
```

The method `hasReferences` verifies if the handler considers the node to have references. It takes as argument an `AuthorNode` that represents the node which will be verified. The method will return `true` if the node is considered to have references. In our case, to be a reference the node must be an element with the name *ref* and it must have an attribute named *location*.

```
public boolean hasReferences(AuthorNode node) {
  boolean hasReferences = false;
  if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
    AuthorElement element = (AuthorElement) node;
    if ("ref".equals(element.getLocalName())) {
      AttrValue attrValue = element.getAttribute("location");
      hasReferences = attrValue != null;
    }
  }
  return hasReferences;
}
```

The method `getDisplayName` returns the display name of the node that contains the expanded referred content. It takes as argument an `AuthorNode` that represents the node for which the display name is needed. The referred content engine will ask this `AuthorReferenceResolver` implementation what is the display name for each node which is considered a reference. In our case the display name is the value of the *location* attribute from the *ref* element.

```
public String getDisplayName(AuthorNode node) {
  String displayName = "ref-fragment";
  if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
    AuthorElement element = (AuthorElement) node;
    if ("ref".equals(element.getLocalName())) {
      AttrValue attrValue = element.getAttribute("location");
      if (attrValue != null) {
        displayName = attrValue.getValue();
      }
    }
  }
  return displayName;
}
```

The method `resolveReference` resolves the reference of the node and returns a `SAXSource` with the parser and the parser's input source. It takes as arguments an `AuthorNode` that represents the node for which the reference needs resolving, the `systemID` of the node, the `AuthorAccess` with access methods to the Author data model and a SAX `EntityResolver` which resolves resources that are already opened in another editor or resolve resources through the XML catalog. In our implementation we need to resolve the reference relative to the `systemID`, and create a parser and an input source over the resolved reference.

```
public SAXSource resolveReference(
    AuthorNode node,
    String systemID,
    AuthorAccess authorAccess,
    EntityResolver  entityResolver) {
  SAXSource saxSource = null;

  if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
    AuthorElement element = (AuthorElement) node;
    if ("ref".equals(element.getLocalName())) {
      AttrValue attrValue = element.getAttribute("location");
      if (attrValue != null) {
        String attrStringVal = attrValue.getValue();
        try {
          URL absoluteUrl = new URL(new URL(systemID),
              authorAccess.correctURL(attrStringVal));

          InputSource inputSource = entityResolver.resolveEntity(null,
              absoluteUrl.toString());
          if(inputSource == null) {
            inputSource = new InputSource(absoluteUrl.toString());
          }

          XMLReader xmlReader = authorAccess.newNonValidatingXMLReader();
          xmlReader.setEntityResolver(entityResolver);

          saxSource = new SAXSource(xmlReader, inputSource);
        } catch (MalformedURLException e) {
          logger.error(e, e);
        } catch (SAXException e) {
          logger.error(e, e);
        } catch (IOException e) {
          logger.error(e, e);
        }
      }
    }
  }

  return saxSource;
}
```

The method `getReferenceUniqueID`should return an unique identifier for the node reference. The unique identifier is used to avoid resolving the references recursively. It takes as argument an `AuthorNode` that represents the node with the reference. In our implementation the unique identifier is the value of the *location* attribute from the *ref* element.

```
public String getDisplayName(AuthorNode node) {
  String displayName = "ref-fragment";
  if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
    AuthorElement element = (AuthorElement) node;
    if ("ref".equals(element.getLocalName())) {
      AttrValue attrValue = element.getAttribute("location");
      if (attrValue != null) {
        displayName = attrValue.getValue();
      }
    }
  }
  return displayName;
}
```

The method `getReferenceSystemID`should return the `systemID` of the referred content. It takes as arguments an `AuthorNode` that represents the node with the reference and the `AuthorAccess` with access methods to the Author data model. In our implementation we use the value of the *location* attribute from the *ref* element and resolve it relatively to the XML base URL of the node.

```
public String getReferenceSystemID(AuthorNode node,
                                   AuthorAccess authorAccess) {
  String systemID = null;
  if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
    AuthorElement element = (AuthorElement) node;
    if ("ref".equals(element.getLocalName())) {
      AttrValue attrValue = element.getAttribute("location");
      if (attrValue != null) {
        String attrStringVal = attrValue.getValue();
        try {
          URL absoluteUrl = new URL(node.getXMLBaseURL(),
              authorAccess.correctURL(attrStringVal));
          systemID = absoluteUrl.toString();
        } catch (MalformedURLException e) {
          logger.error(e, e);
        }
      }
    }
  }
  return systemID;
}
```

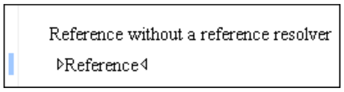
The complete source code of our implementation is found in the Example Files Listings, the Java Files section.

In the listing below, the XML document contains the **ref** element:

```
<ref location="referred.xml">Reference</ref>
```

When no reference resolver is specified, the reference has the following layout:

Figure 7.33. Reference with no specified reference resolver

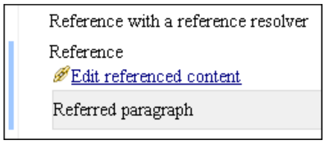


Reference without a reference resolver

▷Reference◁

When the above implementation is configured, the reference has the expected layout:

Figure 7.34. Reference with reference resolver



Reference with a reference resolver

Reference

Edit referenced content

Referred paragraph

Configuring CSS Styles Filter

You can modify the CSS styles for each `ro.sync.ecss.extensions.api.node.AuthorNode` rendered in the Author page using an implementation of `ro.sync.ecss.extensions.api.StylesFilter` You can implement the various callbacks of the interface either by returning the default value given by <oXygen/> or by contributing to the value. The received styles `ro.sync.ecss.css.Styles` can be processed and values can be overwritten with your own. For example you can override the `KEY_BACKGROUND_COLOR` style to return your own implementation of `ro.sync.exml.view.graphics.Color` or override the `KEY_FONT` style to return your own implementation of `ro.sync.exml.view.graphics.Font`.

For instance in our simple document example the filter can change the value of the `KEY_FONT` property for the `table` element:

```
package simple.documentation.framework;

import ro.sync.ecss.css.Styles;
import ro.sync.ecss.extensions.api.StylesFilter;
import ro.sync.ecss.extensions.api.node.AuthorNode;
import ro.sync.exml.view.graphics.Font;

public class SDFStylesFilter implements StylesFilter {

     public Styles filter(Styles styles, AuthorNode authorNode) {
         if (AuthorNode.NODE_TYPE_ELEMENT == authorNode.getType()
           && "table".equals(authorNode.getName())) {
           styles.setProperty(Styles.KEY_FONT, new Font(null, Font.BOLD, 12));
         }
         return styles;
     }
}
```

Configuring a Table Column Width Provider

In our documentation framework the `table` element and the table columns can have specified widths. In order for these widths to be considered by <oXygen/> Author we need to provide the means to determine them. As explained in the Styling the Table Element section which describes the CSS properties needed for defining a table, if we use the table element attribute **width** <oXygen/> can determine the table width automatically. In our example the table has `col` elements with **width** attributes that are not recognized by default. We will need to implement a Java extension class for determining the column widths.

Create the class `simple.documentation.framework.TableColumnWidthProvider`. This class must implement the `ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider` interface.

```
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.AuthorOperationException;
import ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider;
import ro.sync.ecss.extensions.api.WidthRepresentation;
import ro.sync.ecss.extensions.api.node.AuthorElement;

public class TableColumnWidthProvider
      implements AuthorTableColumnWidthProvider {
```

The method `init` is taking as argument the `AuthorElement` that represents the XML `table` element. In our case the column widths are specified in `col` elements from the `table` element. In such cases you must collect the span information by analyzing the `table` element.

```
 public void init(AuthorElement tableElement) {
  this.tableElement = tableElement;
  AuthorElement[] colChildren = tableElement.getElementsByLocalName("customcol");
  if (colChildren != null && colChildren.length > 0) {
   for (int i = 0; i < colChildren.length; i++) {
    AuthorElement colChild = colChildren[i];
    if (i == 0) {
     colsStartOffset = colChild.getStartOffset();
    }
    if (i == colChildren.length - 1) {
     colsEndOffset = colChild.getEndOffset();
    }
    // Determine the 'width' for this col.
    AttrValue colWidthAttribute = colChild.getAttribute("width");
    String colWidth = null;
    if (colWidthAttribute != null) {
     colWidth = colWidthAttribute.getValue();
     // Add WidthRepresentation objects for the columns this 'customcol' specification
     // spans over.
     colWidthSpecs.add(new WidthRepresentation(colWidth, true));
    }
   }
  }
 }
```

The method `isTableAcceptingWidth` should check if the table cells are `td`.

```
public boolean isTableAcceptingWidth(String tableCellsTagName) {
  return "td".equals(tableCellsTagName);
}
```

The method `isTableAndColumnsResizable` should check if the table cells are `td`. This method determines if the table and columns can be resized by dragging with the mouse the edge of a column.

```
public boolean isTableAndColumnsResizable(String tableCellsTagName) {
  return "td".equals(tableCellsTagName);
}
```

The methods `getTableWidth` and `getCellWidth` are used for determining the table width and the column width. The table layout engine will ask this `AuthorTableColumnWidthProvider` implementation what is the table width for each table element and the cell width for each cell element from the table that was marked as cell in the CSS using the property `display:table-cell`. The implementation is simple and just parses the value of the **width** attribute. The methods must return `null` for the tables/cells that do not have a specified width.

```
 public WidthRepresentation getTableWidth(String tableCellsTagName) {
  WidthRepresentation toReturn = null;
  if (tableElement != null && "td".equals(tableCellsTagName)) {
   AttrValue widthAttr = tableElement.getAttribute("width");
   if (widthAttr != null) {
    String width = widthAttr.getValue();
    if (width != null) {
     toReturn = new WidthRepresentation(width, true);
    }
   }
  }
  return toReturn;
 }

 public List<WidthRepresentation> getCellWidth(AuthorElement cellElement, int colNumberSta
   int colSpan) {
  List<WidthRepresentation> toReturn = null;
  int size = colWidthSpecs.size();
  if (size >= colNumberStart && size >= colNumberStart + colSpan) {
   toReturn = new ArrayList<WidthRepresentation>(colSpan);
   for (int i = colNumberStart; i < colNumberStart + colSpan; i ++) {
    // Add the column widths
    toReturn.add(colWidthSpecs.get(i));
   }
  }
  return toReturn;
 }
```

The methods `commitTableWidthModification` and `commitColumnWidthModifications` are used for committing changes made to the width of the table or its columns when using the mouse drag gestures.

```
 public void commitTableWidthModification(AuthorDocumentController authorDocumentControlle
   int newTableWidth, String tableCellsTagName) throws AuthorOperationException {
  if ("td".equals(tableCellsTagName)) {
   if (newTableWidth > 0) {
    if (tableElement != null) {
     String newWidth = String.valueOf(newTableWidth);
     authorDocumentController.setAttribute(
       "width",
       new AttrValue(newWidth),
       tableElement);
    } else {
     throw new AuthorOperationException("Cannot find the element representing the table.")
    }
   }
  }
 }

public void commitColumnWidthModifications(AuthorDocumentController authorDocumentControll
   WidthRepresentation[] colWidths, String tableCellsTagName) throws AuthorOperationExcept
  if ("td".equals(tableCellsTagName)) {
   if (colWidths != null && tableElement != null) {
    if (colsStartOffset >= 0 && colsEndOffset >= 0 && colsStartOffset < colsEndOffset) {
     authorDocumentController.delete(colsStartOffset,
       colsEndOffset);
    }
    String xmlFragment = createXMLFragment(colWidths);
    int offset = -1;
    AuthorElement[] header = tableElement.getElementsByLocalName("header");
    if (header != null && header.length > 0) {
     // Insert the cols elements before the 'header' element
     offset = header[0].getStartOffset();
    }
    if (offset == -1) {
     throw new AuthorOperationException("No valid offset to insert the columns width speci
    }
    authorDocumentController.insertXMLFragment(xmlFragment, offset);
   }
  }
 }

 private String createXMLFragment(WidthRepresentation[] widthRepresentations) {
  StringBuffer fragment = new StringBuffer();
  String ns = tableElement.getNamespace();
  for (int i = 0; i < widthRepresentations.length; i++) {
   WidthRepresentation width = widthRepresentations[i];
   fragment.append("<customcol");
   String strRepresentation = width.getWidthRepresentation();
   if (strRepresentation != null) {
    fragment.append(" width=\"" + width.getWidthRepresentation() + "\"");
   }
   if (ns != null && ns.length() > 0) {
    fragment.append(" xmlns=\"" + ns + "\"");
   }
   fragment.append("/>");
  }
  return fragment.toString();
 }
```

The following three methods are used to determine what type of column width specifications the table column width provider support. In our case all types of specifications are allowed:

```
 public boolean isAcceptingFixedColumnWidths(String tableCellsTagName) {
  return true;
 }

 public boolean isAcceptingPercentageColumnWidths(String tableCellsTagName) {
  return true;
 }

 public boolean isAcceptingProportionalColumnWidths(String tableCellsTagName) {
  return true;
 }
```

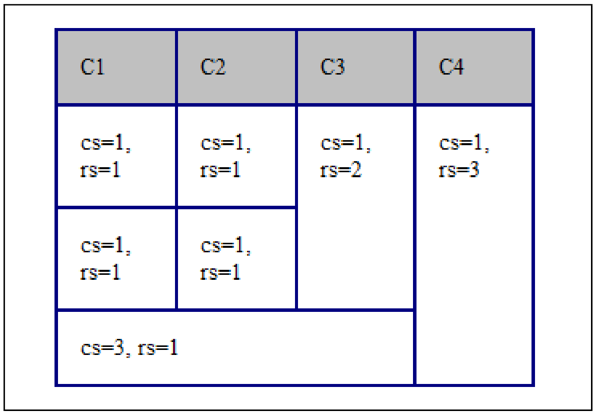
The complete source code of our implementation is found in the Example Files Listings, the Java Files section.

In the listing below, the XML document contains the table element:

```
<table width="300">
    <customcol width="50.0px"/>
    <customcol width="1*"/>
    <customcol width="2*"/>
    <customcol width="20%"/>
    <header>
        <td>C1</td>
        <td>C2</td>
        <td>C3</td>
        <td>C4</td>
    </header>
    <tr>
        <td>cs=1, rs=1</td>
        <td>cs=1, rs=1</td>
        <td row_span="2">cs=1, rs=2</td>
        <td row_span="3">cs=1, rs=3</td>
    </tr>
    <tr>
        <td>cs=1, rs=1</td>
        <td>cs=1, rs=1</td>
    </tr>
    <tr>
        <td column_span="3">cs=3, rs=1</td>
    </tr>
</table>
```

When no table column width provider is specified, the table has the following layout:

Figure 7.35. Table layout when no column width provider is specified



C1	C2	C3	C4

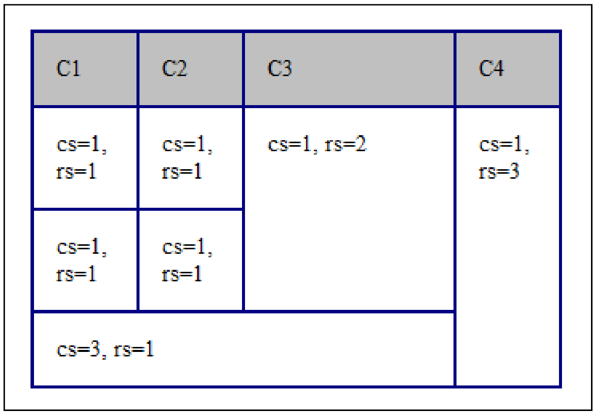
cs=1,
rs=1
	cs=1,
	rs=1
		cs=1,
		rs=2
			cs=1,
			rs=3

cs=1,
rs=1
	cs=1,
	rs=1		

cs=3, rs=1			

When the above implementation is configured, the table has the correct layout:

Figure 7.36. Columns with custom widths



C1	C2	C3	C4

cs=1,
rs=1
	cs=1,
	rs=1
		cs=1, rs=2	cs=1,
			rs=3

cs=1,
rs=1
	cs=1,
	rs=1		

cs=3, rs=1			

Configuring a Table Cell Span Provider

In our documentation framework the `table` element can have cells that span over multiple columns and rows. As explained in the Styling the Table Element section which describes the CSS properties needed for defining a table, we need to indicate <oXygen/> Author a method to determine the cell spanning. If we use the cell element attributes **rowspan** and **colspan** or **rows** and **cols**, <oXygen/> can determine the cell spanning automatically. In our example the `td` element uses the attributes **row_span** and **column_span** that are not recognized by default. We will need to implement a Java extension class for defining the cell spanning.

Create the class `simple.documentation.framework.TableCellSpanProvider`. This class must implement the `ro.sync.ecss.extensions.api.AuthorTableCellSpanProvider` interface.

```
import ro.sync.ecss.extensions.api.AuthorTableCellSpanProvider;
import ro.sync.ecss.extensions.api.node.AttrValue;
import ro.sync.ecss.extensions.api.node.AuthorElement;

public class TableCellSpanProvider
      implements AuthorTableCellSpanProvider {
```

The method init is taking as argument the AuthorElement that represents the XML table element. In our case the cell span is specified for each of the cells so we leave this method empty. However there are cases like the table CALS model when the cell spanning is specified in the table element. In such cases you must collect the span information by analyzing the table element.

```
public void init(AuthorElement table) {
}
```

The method getColSpan is taking as argument the table cell. The table layout engine will ask this AuthorTableSpanSupport implementation what is the column span and the row span for each XML element from the table that was marked as cell in the CSS using the property display:table-cell. The implementation is simple and just parses the value of **column_span** attribute. The method must return null for all the cells that do not change the span specification.

```
public Integer getColSpan(AuthorElement cell) {
  Integer colSpan = null;

  AttrValue attrValue = cell.getAttribute("column_span");
  if(attrValue != null) {
    // The attribute was found.
    String cs = attrValue.getValue();
    if(cs != null) {
      try {
        colSpan = new Integer(cs);
      } catch (NumberFormatException ex) {
        // The attribute value was not a number.
      }
    }
  }
  return colSpan;
}
```

The row span is determined in a similar manner:

```
public Integer getRowSpan(AuthorElement cell) {
  Integer rowSpan = null;

  AttrValue attrValue = cell.getAttribute("row_span");
  if(attrValue != null) {
    // The attribute was found.
    String rs = attrValue.getValue();
    if(rs != null) {
      try {
        rowSpan = new Integer(rs);
      } catch (NumberFormatException ex) {
        // The attribute value was not a number.
      }
    }
  }
  return rowSpan;
}
```

The method hasColumnSpecifications always returns true considering column specifications always available.

```
public boolean hasColumnSpecifications(AuthorElement tableElement) {
  return true;
}
```

The complete source code of our implementation is found in the Example Files Listings, the Java Files section.

In the listing below, the XML document contains the table element:

```
<table>
    <header>
        <td>C1</td>
        <td>C2</td>
        <td>C3</td>
        <td>C4</td>
    </header>
    <tr>
        <td>cs=1, rs=1</td>
        <td column_span="2" row_span="2">cs=2, rs=2</td>
        <td row_span="3">cs=1, rs=3</td>
    </tr>
    <tr>
        <td>cs=1, rs=1</td>
    </tr>
    <tr>
        <td column_span="3">cs=3, rs=1</td>
    </tr>
</table>
```

When no table cell span provider is specified, the table has the following layout:

Figure 7.37. Table layout when no cell span provider is specified

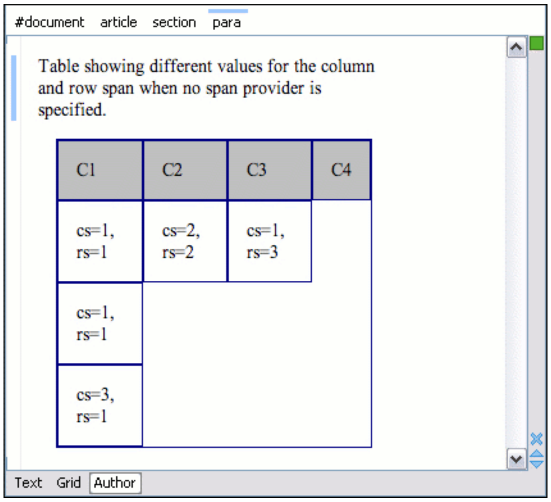


Table showing different values for the column and row span when no span provider is specified.

C1	C2	C3	C4
cs=1, rs=1	cs=2, rs=2	cs=1, rs=3	
cs=1, rs=1			
cs=3, rs=1			

When the above implementation is configured, the table has the correct layout:

Figure 7.38. Cells spanning multiple rows and columns.

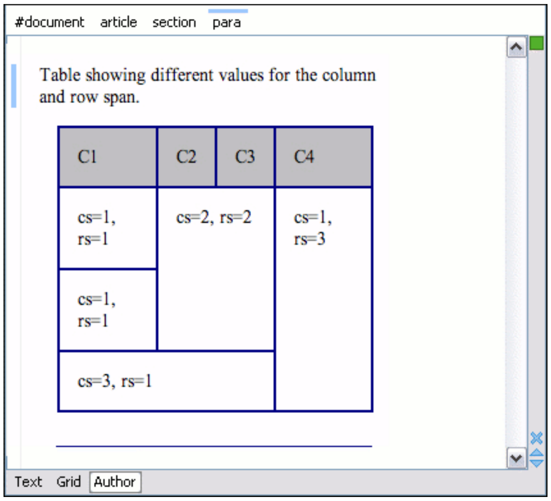


Table showing different values for the column and row span.

C1	C2	C3	C4
cs=1, rs=1	cs=2, rs=2		cs=1, rs=3
cs=1, rs=1			
cs=3, rs=1			

Configuring an Unique Attributes Recognizer

The `ro.sync.ecss.extensions.api.UniqueAttributesRecognizer` interface can be implemented if you want to provide for your framework the following features:

Automatic ID generation

You can automatically generate unique IDs for newly inserted elements. Implementations are already available for the DITA and Docbook frameworks. The following methods can be implemented to accomplish this:

```
/**
 * Assign unique IDs between a start
 * and an end offset in the document
 * @param startOffset Start offset
 * @param endOffset End offset
 */
void assignUniqueIDs(int startOffset, int endOffset);

/**
 * @return true if auto
 */
boolean isAutoIDGenerationActive();
```

Avoiding copying unique attributes when "Split" is called inside an element

You can split the current block element by pressing the "Enter" key and then choosing "Split". This is a very useful way to create new paragraphs. All attributes are by default copied on the new element but if those attributes are IDs you sometimes want to avoid creating validation errors in the editor. Implementing the following method, you can decide whether an attribute should be copied or not during the split:

```
  /**
   * Check if the attribute specified by QName can
   * be considered as a valid attribute to copy
   * when the element is split.
   *
   * @param attrQName The attribute qualified name
   * @param element The element
   * @return true if the attribute should be copied
   * when Split is performed.
   */
  boolean copyAttributeOnSplit(String attrQName,
               AuthorElement element);
```

Tip

The ro.sync.ecss.extensions.commons.id.DefaultUniqueAttributesRecognizer class is an implementation of the interface which can be extended by your customization to provide easy assignation of IDs in your framework. You can also check out the DITA and Docbook implementations of ro.sync.ecss.extensions.api.UniqueAttributesRecognizer to see how they were implemented and connected to the extensions bundle.

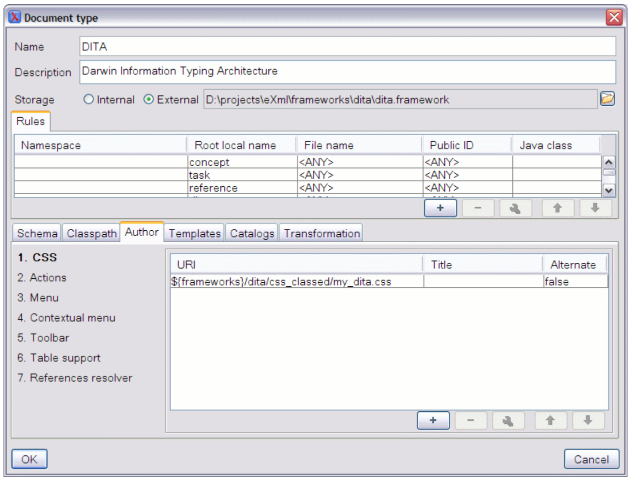
Customizing the default CSS of a document type

The easiest way of customizing the default CSS stylesheet of a document type is to create a new CSS stylesheet in the same folder as the customized one, import the customized CSS stylesheet and set the new stylesheet as the default CSS of the document type. For example let us customize the default CSS for DITA documents by changing the background color of the *task* and *topic* elements to red. First we create a new CSS stylesheet called *my_dita.css* in the folder *${frameworks}/dita/css_classed* where the default stylesheet called *dita.css* is located. *${frameworks}* is the subfolder *frameworks* of the Oxygen XML Editor. The new stylesheet *my_dita.css* contains:

```
@import "dita.css";

task, topic{
    background-color:red;
}
```

To set the new stylesheet as the default CSS stylesheet for DITA documents first open the Document Type Association preferences panel from menu Options → Preferences+Document Type Association Select the DITA document type and start editing it by pressing the Edit button. The user role must be set to *Developer* otherwise a warning is displayed and a duplicate copy of the DITA document type is created and edited. This check makes sure that regular content authors who just edit the content of XML documents do not accidentally modify the document type. In the Author tab of the document type edit dialog change the URI of the default CSS stylesheet from *${frameworks}/dita/css_classed/dita.css* to *${frameworks}/dita/css_classed/my_dita.css*.

Figure 7.39. Set the location of the default CSS stylesheet



Document type

Name: DITA

Description: Darwin Information Typing Architecture

Storage: ○ Internal ● External D:\projects\eXml\frameworks\dita\dita.framework

Rules

Namespace	Root local name	File name	Public ID	Java class
	concept	<ANY>	<ANY>	
	task	<ANY>	<ANY>	
	reference	<ANY>	<ANY>	

URI	Title	Alternate
${frameworks}/dita/css_classed/my_dita.css		false

Press OK in all the dialogs to validate the changes. Now you can start editing DITA documents based on the new CSS stylesheet. You can edit the new CSS stylesheet itself at any time and see the effects on rendering DITA XML documents in the Author mode by running the *Refresh* action available on the Author toolbar and on the DITA menu.

Document type sharing

A document type can be shared between authors in two ways:

- save the document type at global level in the Document Type Association panel and distribute a zip file that includes all the files of the document type (CSS stylesheets, jar files with custom actions, etc). Each user will unzip the zip file in a subdirectory of the ${frameworks} directory and will restart the application for adding the new document type to the list of the Document Type Association panel
- save the document type at project level in the Document Type Association panel and distribute both the Oxygen project file and the files of the document type (CSS stylesheets, jar files with custom actions, etc). Each user will copy the files of the document type in the subdirectory of the ${frameworks} directory that corresponds to the document type and will load the Oxygen project file in the *Project* view.

CSS support in <oXygen/> Author

CSS 2.1 features

Supported selectors

The following CSS level 2.1 selectors are supported by the <oXygen/> Author:

Table 7.3. Supported CSS 2.1 selectors

Expression	Name	Description/Example
*	Universal selector	Matches any element
E	Type selector	Matches any E element (i.e an element with the local name E)
E F	Descendant selector	Matches any F element that is a descendant of an E element.
E > F	Child selectors	Matches any F element that is a child of an element E.
E:first-child	The :first-child pseudo-class	Matches element E when E is the first child of its parent.
E:lang(c)	The :lang() pseudo-class	Matches element of type E if it is in (human) language c (the document language specifies how language is determined).
E + F	Adjacent selector	Matches any F element immediately preceded by a sibling element E.
E[foo]	Attribute selector	Matches any E element with the "foo" attribute set (whatever the value).
E[foo="warning"]	Attribute selector	Matches any E element whose "foo" attribute value is exactly equal to "warning".
E[foo~="warning"]	Attribute selector	Matches any E element whose "foo" attribute value is a list of space-separated values, one of which is exactly equal to "warning".
E[lang|="en"]	Attribute selector	Matches any E element whose "lang" attribute has a hyphen-separated list of values beginning (from the left) with "en".
E:before and E:after	Pseudo elements	The ':before' and ':after' pseudo-elements can be used to insert generated content before or after an element's content.

Unsupported selectors

The following CSS level 2.1 selectors are **not supported** by the <oXygen/> Author:

Table 7.4. Unsupported CSS 2.1 selectors

Expression	Name	Description/Example
E#myid	ID selectors	Matches any E element with ID equal to "myid".
E:link, E:visited	The link pseudo-class	Matches element E if E is the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited).
E:active, E:hover, E:focus	The dynamic pseudo-classes	Matches E during certain user actions.
E:first-line	The :first-line pseudo-class	The :first-line pseudo-element applies special styles to the contents of the first formatted line of a paragraph.
E:first-letter	The :first-letter pseudo-class	The :first-letter pseudo-element must select the first letter of the first line of a block, if it is not preceded by any other content (such as images or inline tables) on its line. The :first-letter pseudo-element may be used for "initial caps" and "drop caps", which are common typographical effects.

Properties Support Table

All the properties belonging to the *aural* and *paged* categories are **not supported** in <oXygen/> Author. The properties from the table below belong to the *visual* category.

Table 7.5. CSS Level 2.1 Properties and their support in <oXygen/> Author

Name	Supported Values	Not Supported Values
'background-attachment'		ALL
'background-color'	<color> | inherit	transparent
'background-image'		ALL
'background-position'		ALL
'background-repeat'		ALL
'background'		ALL
'border-collapse'		ALL
'border-color'	<color> | inherit	transparent
'border-spacing'		ALL
'border-style'	<border-style> | inherit	
'border-top' 'border-right' 'border-bottom' 'border-left'	[<border-width> || <border-style> || 'border-top-color'] | inherit	
'border-top-color' 'border-right-color' 'border-bottom-color' 'border-left-color'	<color> | inherit	transparent
'border-top-style' 'border-right-style' 'border-bottom-style' 'border-left-style'	<border-style> | inherit	
'border-top-width' 'border-right-width' 'border-bottom-width' 'border-left-width'	<border-width> | inherit	
'border-width'	<border-width> | inherit	
'border'	[<border-width> || <border-style> || 'border-top-color'] | inherit	
'bottom'		ALL
'caption-side'		ALL
'clear'		ALL
'clip'		ALL
'color'	<color> | inherit	
'content'	normal | none | [<string> | <uri> | <counter> | attr(<identifier>) | open-quote | close-quote]+ | inherit	no-open-quote | no-close-quote
'counter-increment'	[<identifier> <integer> ?]+ | none | inherit	
'counter-reset'	[<identifier> <integer> ?]+ | none | inherit	
'cursor'		ALL
'direction'	ltr	rtl | inherit
'display'	inline | block | list-item | table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit	run-in | inline-block | inline-table - considered block
'empty-cells'	show | hide | inherit	
'float'		ALL
'font-family'	[[<family-name> | <generic-family>] [, <family-name> | <generic-family>]*] | inherit	

Name	Supported Values	Not Supported Values
'font-size'	<absolute-size> | <relative-size> | <length> | <percentage> | inherit	
'font-style'	normal | italic | oblique | inherit	
'font-variant'		ALL
'font-weight'	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit	
'font'	[['font-style' || 'font-weight']? 'font-size' [/ 'line-height']? 'font-family'] | inherit	'font-variant' 'line-height' caption | icon | menu | message-box | small-caption | status-bar
'height'		ALL
'left'		ALL
'letter-spacing'		ALL
'line-height'	normal | <number> | <length> | <percentage> | inherit	
'list-style-image'		ALL
'list-style-position'		ALL
'list-style-type'	disc | circle | square | decimal | lower-roman | upper-roman | lower-latin | upper-latin | lower-alpha | upper-alpha | none | inherit	lower-greek | armenian | georgian
'list-style'	['list-style-type'] | inherit	'list-style-position' || 'list-style-image'
'margin-right' 'margin-left'	<margin-width> | inherit	
'margin-top' 'margin-bottom'	<margin-width> | inherit	
'margin'	<margin-width> | inherit	
'max-height'		ALL
'max-width'	<length> | <percentage> | none | inherit - supported for block-level and replaced elements, e.g. images, tables, table cells.	
'min-height'		ALL
'min-width'	<length> | <percentage> | inherit - supported for block-level and replaced elements, e.g. images, tables, table cells.	
'outline-color'		ALL
'outline-style'		ALL
'outline-width'		ALL
'outline'		ALL
'overflow'		ALL
'padding-top' 'padding-right' 'padding-bottom' 'padding-left'	<padding-width> | inherit	
'padding'	<padding-width> | inherit	
'position'		ALL

Name	Supported Values	Not Supported Values
'quotes'		ALL
'right'		ALL
'table-layout'	auto	fixed | inherit
'text-align'	left | right | center | inherit	justify
'text-decoration'	none | [underline || overline || line-through] | inherit	blink
'text-indent'		ALL
'text-transform'		ALL
'top'		ALL
'unicode-bidi'		ALL
'vertical-align'	baseline | sub | super | top | text-top | middle | bottom | text-bottom | inherit	<percentage> | <length>
'visibility'	visible | hidden | inherit	collapse
'white-space'	normal | pre | nowrap | pre-wrap | pre-line	
'width'	<length> | <percentage> | auto | inherit - supported for block-level and replaced elements, e.g. images, tables, table cells.	
'word-spacing'		ALL
'z-index'		ALL

<oXygen/> CSS Extensions

Media Type oxygen

The style sheets can specify how a document is to be presented on different media: on the screen, on paper, speech synthesiser, etc. You can specify that some of the features of your CSS stylesheet should be taken into account only in the <oXygen/> Author and ignored in the rest. This can be accomplished by using the media type oxygen.

For instance using the following CSS:

```
b{
 font-weight:bold;
 display:inline;
}

@media oxygen{
 b{
  text-decoration:underline;
 }
}
```

would make a text bold if the document was opened in a web browser who does not recognize `@media oxygen` and bold and underlined in <oXygen/> Author.

You can use this media type to group specific <oXygen/> CSS features and also to hide them when opening the documents with other viewers.

Supported Features from CSS Level 3

Namespace Selectors

In the current CSS 2.1 standard the element selectors are ignoring the namespaces of the elements they are matching. Only the local name of the elements are considered in the selector matching process.

<oXygen/> Author uses a different approach similar to the CSS Level 3 specification. If the element name from the CSS selector is not preceded by a namespace prefix it is considered to match an element with the same local name as the selector value and ANY namespace, otherwise the element must match both the local name and the namespace.

In CSS up to version 2.1 the name tokens from the selectors are matching all elements from ANY namespace that have the same local name. Example:

```
<x:b xmlns:x="ns_x"/>
<y:b xmlns:y="ns_y"/>
```

Are both matched by the rule:

```
b {font-weight:bold}
```

Starting with CSS Level 3 you can create selectors that are namespace aware.

Example 7.5. Defining both prefixed namespaces and the default namespace

Given the namespace declarations:

```
@namespace sync "http://sync.example.org";
@namespace "http://example.com/foo";
```

In a context where the default namespace applies:

sync|A represents the name A in the `http://sync.example.org` namespace.

|B represents the name B that belongs to NO NAMESPACE.

*|C represents the name C in ANY namespace, including NO NAMESPACE.

D represents the name D in the `http://example.com/foo` namespace.

Example 7.6. Defining only prefixed namespaces

Given the namespace declaration:

```
@namespace sync "http://sync.example.org";
```

Then:

sync|A represents the name A in the `http://sync.example.org` namespace.

|B represents the name B that belongs to NO NAMESPACE.

*|C represents the name C in ANY namespace, including NO NAMESPACE.

D represents the name D in ANY namespace, including NO NAMESPACE.

The `attr()` function: Properties Values Collected from the Edited Document.

In CSS Level 2.1 you may collect attribute values and use them as content *only* for the pseudo elements. For instance the :before pseudo-element can be used to insert some content before an element. This is valid in CSS 2.1:

```
title:before{
  content: "Title id=(" attr(id) ")";
}
```

If the `title` element from the XML document is:

```
<title id="title12">My title.</title>
```

Then the title will be displayed as:

```
Title id=(title12) My title.
```

In <oXygen/> Author the use of `attr()` function is not available only for the `content` property but for any other property. This is similar to the CSS Level 3 working draft: http://www.w3.org/TR/2006/WD-css3-values-20060919/#functional. The arguments of the function are:

```
attr(attribute_name, attribute_type, default_value);
```

```
attribute_name ;
attribute_type ;
default_value ;
```

attribute_name
: The name of the attribute. This argument is required.

attribute_type
: The type of the attribute. This argument is optional. If it is missing the type of the argument is considered `string`. This argument indicates what is the meaning of the attribute value and helps to perform conversions of this value. <oXygen/> Author accepts one of the following types:

- color: The value represents a color. The attribute may specify a color in different formats. <oXygen/> Author supports colors specified either by name: `red`, `blue`, `green`, etc. or as an RGB hexadecimal value `#FFEEFF`.
- url: The value is an URL pointing to a media object. <oXygen/> Author supports only images. The attribute value can be a complete URL, or a relative one to the XML document.
- integer: The value must be interpreted as an integer.
- number: The value must be interpreted as a float number.
- length: The value must be interpreted as an integer.
- percentage: The value must be interpreted relative to another value (length, size) expressed in percents.
- em: The value must be interpreted as a size. 1 em is equal to the *font-size* of the relevant font.
- ex: The value must be interpreted as a size. 1 ex is equal to the *height* of the **x** character of the relevant font.

px The value must be interpreted as a size expressed in pixels relative to the viewing device.

mm The value must be interpreted as a size expressed in millimeters.

cm The value must be interpreted as a size expressed in centimeters.

in The value must be interpreted as a size expressed in inches. 1 inch is equal to 2.54 centimeters.

pt The value must be interpreted as a size expressed in points. The points used by CSS2 are equal to 1/72th of an inch.

pc The value must be interpreted as a size expressed in picas. 1 pica is equal to 12 points.

default_value This argument specifies a value that is used by default if the attribute value is missing. This argument is optional.

Example 7.7. Usage samples for the attr() function

Consider the following XML instance:

```
<sample>
    <para bg_color="#AAAAFF">Blue paragraph.</para>
    <para bg_color="red">Red paragraph.</para>
    <para bg_color="red" font_size="2">Red paragraph with large font.</para>
    <para bg_color="#00AA00" font_size="0.8" space="4">
        Green paragraph with small font and margin.</para>
</sample>
```

The `para` elements have `bg_color` attributes with RGB color values like `#AAAAFF`. We can use the `attr()` function to change the elements appearance in the editor based on the value of this attribute:

```
background-color:attr(bg_color, color);
```

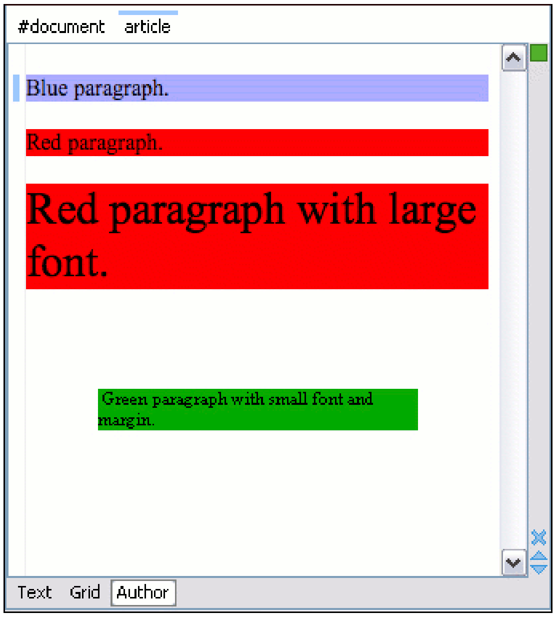
The attribute `font_size` represents the font size in *em* units. We can use this value to change the style of the element:

```
font-size:attr(font_size, em);
```

The complete CSS rule is:

```
para{
 display:block;
 background-color:attr(bg_color, color);
 font-size:attr(font_size, em);
 margin:attr(space, em);
}
```

The document is rendered as:



Blue paragraph.

Red paragraph.

Red paragraph with large font.

Green paragraph with small font and margin.

Additional Custom Selectors

Oxygen Author provides support for selecting additional types of nodes. These custom selectors apply to: *document*, *doctype sections*, *processing-instructions*, *comments*, *CDATA sections*, and *entities*. In order for the custom selectors to work in your CSS files you will have to declare the Author extensions namespace at the beginning of the stylesheet documents:

```
@namespace oxy url('http://www.oxygenxml.com/extensions/author');
```

Example rules:

- *document*

  ```
  oxy|document {
      display:block;
  }
  ```

- *doctype sections*

  ```
  oxy|doctype {
      display:block;
      color:blue;
      background-color:transparent;
  }
  ```

- *processing-instructions*

  ```
  oxy|processing-instruction {
      display:block;
      color:purple;
      background-color:transparent;
  }
  ```

- *comments*

  ```
  oxy|comment {
      display:block;
      color:green;
      background-color:transparent;
  }
  ```

- *CDATA sections*

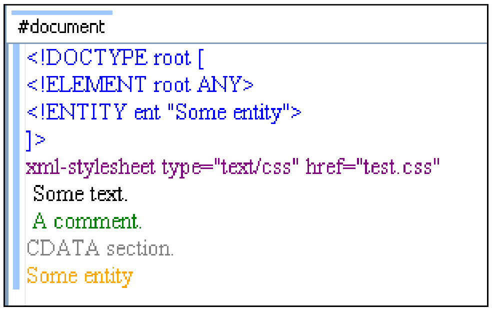
  ```
  oxy|cdata{
      display:block;
      color:gray;
      background-color:transparent;
  }
  ```

- *entities*

  ```
  oxy|entity {
      display:morph;
      editable:false;
      color:orange;
  ```

```
    background-color:transparent;
}
```

A sample document rendered using these rules:



```
#document
<!DOCTYPE root [
<!ELEMENT root ANY>
<!ENTITY ent "Some entity">
]>
xml-stylesheet type="text/css" href="test.css"
 Some text.
 A comment.
CDATA section.
Some entity
```

Additional Properties

Folding elements: `foldable` and `not-foldable-child` properties

<oXygen/> Author allows you to declare some elements to be *foldable* (collapsible). This is especially useful when working with large documents organized in logical blocks, editing a large DocBook article or book for instance.

To define the element whose content can be folded by the user, you must use the property: `foldable:true;`.

When collapsing an element, it is useful to keep some of its content visible, like a short description of the collapsed region. The property `not-foldable-child` is used to identify the child elements that are kept visible. It accepts as value an element name or a list of comma separated element names. If the element is marked as foldable (`foldable:true;`) but it doesn't have the property `not-foldable-child` or none of the specified non-foldable children exists then the element will still be foldable. In this case the element that will be kept visible when folded will be the before pseudo element.

Note

Both `foldable` and `not-foldable-child` are non standard properties and are recognized only by <oXygen/> Author.

Example 7.8. Folding DocBook Elements

All the elements below can have a `title` child element and are considered to be logical sections. We mark them as being foldable leaving the `title` element visible.

```
set,
book,
part,
reference,
chapter,
preface,
article,
sect1,
sect2,
sect3,
sect4,
section,
appendix,
figure,
example,
table {
    foldable:true;
    not-foldable-child: title;
}
```

Link elements

<oXygen/> Author allows you to declare some elements to be *links*. This is especially useful when working with many documents which refer to each other. The links allow for an easy way to get from one document to another. Clicking on the link marker will open the referred resource in an editor.

To define the element which should be considered a link, you must use the property `link` on the before or after pseudo element. The value of the property indicates the location of the linked resource. Since links are usually indicated by the value of an attribute in most cases it will have the a value similar to `attr(href)`

Note

`link` is a non standard property and is recognized only by <oXygen/> Author.

Example 7.9. Docbook Link Elements

All the elements below are defined to be links on the before pseudo element and their value is defined by the value of an attribute.

```
*[href]:before{
    link:attr(href);
    content: "Click " attr(href) " for opening" ;
}

ulink[url]:before{
    link:attr(url);
    content: "Click to open: " attr(url);
}

olink[targetdoc]:before{
    link: attr(targetdoc);
    content: "Click to open: " attr(targetdoc);
}
```

<oXygen/> Custom CSS functions

In <oXygen/> Author there are implemented a few <oXygen/> specific custom CSS functions. Imbricated custom functions are also supported.

Example 7.10. Imbricated functions

The result of the functions below will be the local name of the current node with the first letter capitalized.

```
capitalize(local-name())
```

The `local-name()` function

This function evaluates the local name of the current node. It does not have any arguments

The `name()` function

This function evaluates the qualified name of the current node. It does not have any arguments

The `url()` function

This function evaluates the URL of a location relative to the CSS file location and appends each of the relative path components to the final location.

```
url(location, loc_1, loc_2);(...);
```

```
location ;
loc_1 ;
loc_2 ;
```

location	The location as string. If not absolute, will be solved relative to the CSS file URL.
loc_1 ... loc_n	Relative location path components as string. (optional)

The `base-uri()` function

This function evaluates the base URL in the context of the current node. It does not have any arguments and takes into account the `xml:base` context of the current node. See the XML Base specification [http://www.w3.org/TR/xmlbase/] for more details.

The `parent-url()` function

This function evaluates the parent URL of an URL received as string.

```
parent-url(url);
```

```
url ;
```

url	The url as string.

The `capitalize()` function

This function capitalizes the first letter of the text received as argument.

```
capitalize(text);
```

```
text ;
```

text	The text for which the first letter will be capitalized.

The `uppercase()` function

This function transforms to upper case the text received as argument.

```
uppercase(text);
```

```
text ;
```

text	The text to be capitalized.

The `lowercase()` function

This function transforms to lower case the text received as argument.

```
lowercase(text);
```

```
text ;
```

text	The text to be lower cased.

The `concat()` function

This function concatenates the received string arguments.

```
concat(str_1, str_2);(...);
```

```
str_1 ;
str_2 ;
```

str_1 ... str_n	The string arguments to be concatenated.

The `replace()` function

This function has two signatures:

- `replace(`*`text`*`, `*`target`*`, `*`replacement`*`);`

  `text` `;`
  `target` `;`
  `replacement` `;`

  This function replaces each substring of the text that matches the literal target string with the specified literal replacement string.

text	The text in which the replace will occur.
target	The target string to be replaced.
replacement	The string replacement.

- `replace(`*`text`*`, `*`target`*`, `*`replacement`*`, `*`isRegExp`*`);`

  `text` `;`
  `target` `;`
  `replacement` `;`
  `isRegExp` `;`

  This function replaces each substring of the text that matches the target string with the specified replacement string.

`text`	The text in which the replace will occur.
`target`	The target string to be replaced.
`replacement`	The string replacement.
`isRegExp`	If *true* the target and replacement arguments are considered regular expressions in PERL syntax, if *false* they are considered literal strings.

The `unparsed-entity-uri()` function

This function returns the uri value of an unparsed entity name.

`unparsed-entity-uri(`*`unparsedEntityName`*`);`

`unparsedEntityName` `;`

`unparsedEntityName`	The name of an unparsed entity defined in the DTD.

This function can be useful to display images which are referred with unparsed entity names.

Example 7.11. CSS for displaying the image in Author for an *`imagedata`* with *`entityref`* to an unparsed entity

```
imagedata[entityref]{
content: url(unparsed-entity-uri(attr(entityref)));
}
```

The `attributes()` function

This function concatenates the attributes for an element and returns the serialization.

```
attributes();
```

Example 7.12. attributes()

For the following XML fragment:`<element att1="x" xmlns:a="2" x="&quot;"/>` the `attributes()` function will return `att1="x" xmlns:a="2" x="""`.

Example Files Listings

The Simple Documentation Framework Files

XML Schema files

sdf.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.oxygenxml.com/sample/documentation"
    xmlns:doc="http://www.oxygenxml.com/sample/documentation"
    xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts"
    elementFormDefault="qualified">

    <xs:import
        namespace="http://www.oxygenxml.com/sample/documentation/abstracts"
        schemaLocation="abs.xsd"/>

    <xs:element name="book" type="doc:sectionType"/>
    <xs:element name="article" type="doc:sectionType"/>
    <xs:element name="section" type="doc:sectionType"/>

    <xs:complexType name="sectionType">
        <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element ref="abs:def" minOccurs="0"/>
            <xs:choice>
                <xs:sequence>
                    <xs:element ref="doc:section"
                        maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:choice maxOccurs="unbounded">
                    <xs:element ref="doc:para"/>
                    <xs:element ref="doc:ref"/>
                    <xs:element ref="doc:image"/>
                    <xs:element ref="doc:table"/>
                </xs:choice>
            </xs:choice>
        </xs:sequence>
    </xs:complexType>
```

```
<xs:element name="para" type="doc:paragraphType"/>

<xs:complexType name="paragraphType" mixed="true">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="b"/>
        <xs:element name="i"/>
        <xs:element name="link"/>
    </xs:choice>
</xs:complexType>

<xs:element name="ref">
    <xs:complexType>
        <xs:attribute name="location" type="xs:anyURI"
            use="required"/>
    </xs:complexType>
</xs:element>

<xs:element name="image">
    <xs:complexType>
        <xs:attribute name="href" type="xs:anyURI"
            use="required"/>
    </xs:complexType>
</xs:element>

<xs:element name="table">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="customcol" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="width" type="xs:string"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="header">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="td"
                            maxOccurs="unbounded"
                            type="doc:paragraphType"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="tr" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="td"
                            type="doc:tdType"
                            maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="width" type="xs:string"/>
    </xs:complexType>
    </xs:element>


    <xs:complexType name="tdType">
        <xs:complexContent>
            <xs:extension base="doc:paragraphType">
                <xs:attribute name="row_span"
                    type="xs:integer"/>
                <xs:attribute name="column_span"
                    type="xs:integer"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

abs.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace=
 "http://www.oxygenxml.com/sample/documentation/abstracts">
    <xs:element name="def" type="xs:string"/>
</xs:schema>
```

CSS Files

sdf.css

```
/* Element from another namespace */
@namespace abs "http://www.oxygenxml.com/sample/documentation/abstracts";

abs|def{
    font-family:monospace;
    font-size:smaller;
}
abs|def:before{
    content:"Definition:";
    color:gray;
}

/* Vertical flow */
book,
section,
para,
title,
image,
ref {
    display:block;
}

/* Horizontal flow */
b,i {
    display:inline;
```

```
}

section{
    margin-left:1em;
    margin-top:1em;
}

section{
    foldable:true;
    not-foldable-child: title;
}

link[href]:before{
    display:inline;
    link:attr(href);
    content: "Click to open: " attr(href);
}

/* Title rendering*/
title{
    font-size: 2.4em;
    font-weight:bold;
}

* * title{
    font-size: 2.0em;
}
* * * title{
    font-size: 1.6em;
}
* * * * title{
    font-size: 1.2em;
}

book,
article{
    counter-reset:sect;
}
book > section,
article > section{
    counter-increment:sect;
}
book > section > title:before,
article > section > title:before{
    content: "Section: " counter(sect) " ";
}

/* Inlines rendering*/
b {
    font-weight:bold;
}

i {
    font-style:italic;
```

```
}

/*Table rendering */
table{
    display:table;
    border:1px solid navy;
    margin:1em;
    max-width:1000px;
    min-width:150px;
}

table[width]{
  width:attr(width, length);
}

tr, header{
    display:table-row;
}

header{
    background-color: silver;
    color:inherit
}

td{
  display:table-cell;
  border:1px solid navy;
  padding:1em;
}

image{
    display:block;
    content: attr(href, url);
    margin-left:2em;
}
```

XML Files

sdf_sample.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns="http://www.oxygenxml.com/sample/documentation"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts">
    <title>My Technical Book</title>
    <section>
        <title>XML</title>
        <abs:def>Extensible Markup Language</abs:def>
        <para>In this section of the book I will explain
            different XML applications.</para>
    </section>
    <section>
        <title>Accessing XML data.</title>
        <section>
            <title>XSLT</title>
            <abs:def>Extensible stylesheet language
                transformation (XSLT) is a language for
                transforming XML documents into other XML
                documents.</abs:def>
            <para>A list of XSL elements and what they do..</para>
            <table>
                <header>
                    <td>XSLT Elements</td>
                    <td>Description</td>
                </header>
                <tr>
                    <td>
                        <b>xsl:stylesheet</b>
                    </td>
                    <td>The <i>xsl:stylesheet</i> element is
                        always the top-level element of an
                        XSL stylesheet. The name
                            <i>xsl:transform</i> may be used
                        as a synonym.</td>
                </tr>
                <tr>
                    <td>
                        <b>xsl:template</b>
                    </td>
                    <td>The <i>xsl:template</i> element has
                        an optional mode attribute. If this
                        is present, the template will only
                        be matched when the same mode is
                        used in the invoking
                            <i>xsl:apply-templates</i>
                        element.</td>
                </tr>
                <tr>
                    <td>
                        <b>for-each</b>
                    </td>
                    <td>The xsl:for-each element causes
                        iteration over the nodes selected by
                        a node-set expression.</td>
                </tr>
                <tr>
                    <td column_span="2">End of the list</td>
                </tr>
            </table>
        </section>
        <section>
            <title>XPath</title>
            <abs:def>XPath (XML Path Language) is a terse
                (non-XML) syntax for addressing portions of
                an XML document. </abs:def>
            <para>Some of the XPath functions.</para>
            <table>
```

```
                <header>
                    <td>Function</td>
                    <td>Description</td>
                </header>
                <tr>
                    <td>format-number</td>
                    <td>The <i>format-number</i> function
                        converts its first argument to a
                        string using the format pattern
                        string specified by the second
                        argument and the decimal-format
                        named by the third argument, or the
                        default decimal-format, if there is
                        no third argument</td>
                </tr>
                <tr>
                    <td>current</td>
                    <td>The <i>current</i> function returns
                        a node-set that has the current node
                        as its only member.</td>
                </tr>
                <tr>
                    <td>generate-id</td>
                    <td>The <i>generate-id</i> function
                        returns a string that uniquely
                        identifies the node in the argument
                        node-set that is first in document
                        order.</td>
                </tr>
            </table>
        </section>
    </section>
    <section>
        <title>Documentation frameworks</title>
        <para>One of the most important documentation
            frameworks is Docbook.</para>
        <image
            href="http://www.xmlhack.com/images/docbook.gif"/>
        <para>The other is the topic oriented DITA, promoted
            by OASIS.</para>
        <image
href="http://www.oasis-open.org/images/standards/oasis_standard.jpg"
        />
    </section>
</book>
```

XSL Files

sdf.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
```

```
xpath-default-namespace=
"http://www.oxygenxml.com/sample/documentation">

<xsl:template match="/">
    <html><xsl:apply-templates/></html>
</xsl:template>

<xsl:template match="section">
    <xsl:apply-templates/>
</xsl:template>

<xsl:template match="image">
    <img src="{@href}"/>
</xsl:template>

<xsl:template match="para">
    <p>
        <xsl:apply-templates/>
    </p>
</xsl:template>

<xsl:template match="abs:def"
    xmlns:abs=
    "http://www.oxygenxml.com/sample/documentation/abstracts">
    <p>
        <u><xsl:apply-templates/></u>
    </p>
</xsl:template>

<xsl:template match="title">
    <h1><xsl:apply-templates/></h1>
</xsl:template>

<xsl:template match="b">
    <b><xsl:apply-templates/></b>
</xsl:template>

<xsl:template match="i">
    <i><xsl:apply-templates/></i>
</xsl:template>

<xsl:template match="table">
    <table frame="box" border="1px">
        <xsl:apply-templates/>
    </table>
</xsl:template>

<xsl:template match="header">
    <tr>
        <xsl:apply-templates/>
    </tr>
</xsl:template>

<xsl:template match="tr">
```

```
        <tr>
            <xsl:apply-templates/>
        </tr>
    </xsl:template>

    <xsl:template match="td">
        <td>
            <xsl:apply-templates/>
        </td>
    </xsl:template>

    <xsl:template match="header/header/td">
        <th>
            <xsl:apply-templates/>
        </th>
    </xsl:template>

</xsl:stylesheet>
```

Java Files

InsertImageOperation.java

```
package simple.documentation.framework;

import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.File;
import java.net.MalformedURLException;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.filechooser.FileFilter;

import ro.sync.ecss.extensions.api.ArgumentDescriptor;
import ro.sync.ecss.extensions.api.ArgumentsMap;
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.AuthorOperation;
import ro.sync.ecss.extensions.api.AuthorOperationException;

public class InsertImageOperation implements AuthorOperation {

       //
```

```
// Implementing the Author Operation Interface.
//

/**
 * Performs the operation.
 */
public void doOperation(AuthorAccess authorAccess,
  ArgumentsMap arguments)
  throws IllegalArgumentException,
      AuthorOperationException {

 JFrame oxygenFrame = (JFrame) authorAccess.getParentFrame();
 String href = displayURLDialog(oxygenFrame);
 if (href.length() != 0) {
     // Creates the image XML fragment.
     String imageFragment =
         "<image xmlns='http://www.oxygenxml.com/sample/documentation'" +
                " href='" + href + "'/>";

  // Inserts this fragment at the caret position.
  int caretPosition = authorAccess.getCaretOffset();
  authorAccess.insertXMLFragment(imageFragment, caretPosition);
 }
}

/**
 * Has no arguments.
 *
 * @return null.
 */
public ArgumentDescriptor[] getArguments() {
 return null;
}

/**
 * @return A description of the operation.
 */
public String getDescription() {
 return "Inserts an image element. Asks the" +
" user for a URL reference.";
}

//
// End of interface implementation.
//

//
// Auxiliary methods.
//

/**
 * Displays the URL dialog.
 *
 * @param parentFrame The parent frame for
* the dialog.
 * @return The selected URL string value,
* or the empty string if the user canceled
* the URL selection.
 */
private String displayURLDialog(JFrame parentFrame) {

 final JDialog dlg = new JDialog(parentFrame,
"Enter the value for the href attribute", true);
 JPanel mainContent = new JPanel(new GridBagLayout());

 // The text field.
 GridBagConstraints cstr = new GridBagConstraints();
 cstr.gridx = 0;
 cstr.gridy = 0;
 cstr.weightx = 0;
 cstr.gridwidth = 1;
 cstr.fill = GridBagConstraints.HORIZONTAL;
 mainContent.add(new JLabel("Image URI:"), cstr);

 cstr.gridx = 1;
 cstr.weightx = 1;
 final JTextField urlField = new JTextField();
 urlField.setColumns(15);
 mainContent.add(urlField, cstr);

 // Add the "Browse button."
 cstr.gridx = 2;
 cstr.weightx = 0;
 JButton browseButton = new JButton("Browse");
 browseButton.addActionListener(new ActionListener() {

  /**
   * Shows a file chooser dialog.
   */
  public void actionPerformed(ActionEvent e) {
   JFileChooser fileChooser = new JFileChooser();

   fileChooser.setMultiSelectionEnabled(false);
   // Accepts only the image files.
   fileChooser.setFileFilter(new FileFilter() {
    public String getDescription() {
     return "Image files";
    }

    public boolean accept(File f) {
     String fileName = f.getName();
     return f.isFile() &&
       ( fileName.endsWith(".jpeg")
       || fileName.endsWith(".jpg")
       || fileName.endsWith(".gif")
       || fileName.endsWith(".png")
       || fileName.endsWith(".svg"));
    }
   });
   if (fileChooser.showOpenDialog(dlg)
        == JFileChooser.APPROVE_OPTION) {
    File file = fileChooser.getSelectedFile();
    try {
     // Set the file into the text field.
     urlField.setText(file.toURL().toString());
    } catch (MalformedURLException ex) {
     // This should not happen.
     ex.printStackTrace();
    }
   }
  }
 });
 mainContent.add(browseButton, cstr);

 // Add the "Ok" button to the layout.
 cstr.gridx = 0;
 cstr.gridy = 1;
 cstr.weightx = 0;
 JButton okButton = new JButton("Ok");
 okButton.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
   dlg.setVisible(false);
  }
 });
 mainContent.add(okButton, cstr);
 mainContent.setBorder(
 BorderFactory.createEmptyBorder(10, 5, 10, 5));

 // Add the "Cancel" button to the layout.
 cstr.gridx = 2;
 JButton cancelButton = new JButton("Cancel");
 cancelButton.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
   urlField.setText("");
   dlg.setVisible(false);
  }
 });
 mainContent.add(cancelButton, cstr);

 // When the user closes the dialog
// from the window decoration,
 // assume "Cancel" action.
 dlg.addWindowListener(new WindowAdapter() {
  public void windowClosing(WindowEvent e) {
   urlField.setText("");
  }
 });

 dlg.getContentPane().add(mainContent);
 dlg.pack();
 dlg.setLocationRelativeTo(parentFrame);
 dlg.setVisible(true);
 return urlField.getText();
}

/**
 * Test method.
 *
 * @param args The arguments are ignored.
 */
public static void main(String[] args) {
 InsertImageOperation operation =
      new InsertImageOperation();
 System.out.println("Choosen URL: " +
   operation.displayURLDialog(new JFrame()));
}
}
```

QueryDatabaseOperation.java

```
package simple.documentation.framework;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.Properties;

import ro.sync.ecss.extensions.api.ArgumentDescriptor;
import ro.sync.ecss.extensions.api.ArgumentsMap;
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.AuthorOperation;
import ro.sync.ecss.extensions.api.AuthorOperationException;

public class QueryDatabaseOperation implements AuthorOperation{

  private static String ARG_JDBC_DRIVER ="jdbc_driver";
  private static String ARG_USER ="user";
  private static String ARG_PASSWORD ="password";
  private static String ARG_SQL ="sql";
  private static String ARG_CONNECTION ="connection";

  /**
   * @return The array of arguments the developer must specify when
   * configuring the action.
   */
  public ArgumentDescriptor[] getArguments() {
    ArgumentDescriptor args[] = new ArgumentDescriptor[] {
        new ArgumentDescriptor(
            ARG_JDBC_DRIVER,
            ArgumentDescriptor.TYPE_STRING,
            "The name of the Java class that is the JDBC driver."),
        new ArgumentDescriptor(
            ARG_CONNECTION,
            ArgumentDescriptor.TYPE_STRING,
            "The database URL connection string."),
        new ArgumentDescriptor(
            ARG_USER,
            ArgumentDescriptor.TYPE_STRING,
            "The name of the database user."),
        new ArgumentDescriptor(
            ARG_PASSWORD,
            ArgumentDescriptor.TYPE_STRING,
            "The database password."),
        new ArgumentDescriptor(
            ARG_SQL,
            ArgumentDescriptor.TYPE_STRING,
            "The SQL statement to be executed.")
    };
    return args;
  }

  /**
   * @return The operation description.
   */
  public String getDescription() {
    return "Executes a database query and puts the result in a table.";
  }

  public void doOperation(AuthorAccess authorAccess, ArgumentsMap map)
      throws IllegalArgumentException, AuthorOperationException {

    // Collects the arguments.
    String jdbcDriver =
      (String)map.getArgumentValue(ARG_JDBC_DRIVER);
    String connection =
      (String)map.getArgumentValue(ARG_CONNECTION);
    String user =
      (String)map.getArgumentValue(ARG_USER);
    String password =
      (String)map.getArgumentValue(ARG_PASSWORD);
    String sql =
      (String)map.getArgumentValue(ARG_SQL);

    int caretPosition = authorAccess.getCaretOffset();
    try {
      authorAccess.insertXMLFragment(
          getFragment(jdbcDriver, connection, user, password, sql),
          caretPosition);
    } catch (SQLException e) {
      throw new AuthorOperationException(
          "The operation failed due to the following database error: " +
     e.getMessage(), e);
    } catch (ClassNotFoundException e) {
      throw new AuthorOperationException(
          "The JDBC database driver was not found. Tried to load ' " +
     jdbcDriver + "'", e);
    }
  }
```

```
/**
 * Creates a connection to the database, executes
 * the SQL statement and creates an XML fragment
 * containing a table element that wraps the data
 * from the result set.
 *
 *
 * @param jdbcDriver The class name of the JDBC driver.
 * @param connectionURL The connection URL.
 * @param user The database user.
 * @param password The password.
 * @param sql The SQL statement.
 * @return The string containing the XML fragment.
 *
 * @throws SQLException thrown when there is a
 * problem accessing the database or there are
 * erors in the SQL expression.
 * @throws ClassNotFoundException when the JDBC
 * driver class could not be loaded.
 */
private String getFragment(
      String jdbcDriver,
      String connectionURL,
      String user,
      String password,
      String sql) throws
        SQLException,
        ClassNotFoundException {

    Properties pr = new Properties();
    pr.put("characterEncoding", "UTF8");
    pr.put("useUnicode", "TRUE");
    pr.put("user", user);
    pr.put("password", password);

    // Loads the database driver.
    Class.forName(jdbcDriver);

    // Opens the connection
    Connection connection =
      DriverManager.getConnection(connectionURL, pr);
    java.sql.Statement statement =
      connection.createStatement();
    ResultSet resultSet =
      statement.executeQuery(sql);

    StringBuffer fragmentBuffer = new StringBuffer();
    fragmentBuffer.append(
        "<table xmlns='http://www.oxygenxml.com/sample/documentation'>");

    //
    // Creates the table header.
    //
```

```
      fragmentBuffer.append("<header>");
      ResultSetMetaData metaData = resultSet.getMetaData();
      int columnCount = metaData.getColumnCount();
      for (int i = 1; i <= columnCount; i++) {
          fragmentBuffer.append("<td>");
          fragmentBuffer.append(
              xmlEscape(metaData.getColumnName(i)));
          fragmentBuffer.append("</td>");
        }
      fragmentBuffer.append("</header>");

      //
      // Creates the table content.
      //
      while (resultSet.next()) {
          fragmentBuffer.append("<tr>");
          for (int i = 1; i <= columnCount; i++) {
              fragmentBuffer.append("<td>");
              fragmentBuffer.append(
                  xmlEscape(resultSet.getObject(i)));
              fragmentBuffer.append("</td>");
          }
          fragmentBuffer.append("</tr>");
        }

      fragmentBuffer.append("</table>");

      // Cleanup
      resultSet.close();
      statement.close();
      connection.close();
      return fragmentBuffer.toString();
  }

  /**
   * Some of the values from the database table
   * may contain characters that must be escaped
   * in XML, to ensure the fragment is well formed.
   *
   * @param object The object from the database.
   * @return The escaped string representation.
   */
  private String xmlEscape(Object object) {
    String str = String.valueOf(object);
    return str.
      replaceAll("&", "&amp;").
      replaceAll("<", "&lt;");
  }
}
```

SDFExtensionsBundle.java

```
package simple.documentation.framework;
```

```
import ro.sync.contentcompletion.xml.SchemaManagerFilter;
import ro.sync.ecss.extensions.api.AuthorExtensionStateListener;
import ro.sync.ecss.extensions.api.AuthorReferenceResolver;
import ro.sync.ecss.extensions.api.AuthorTableCellSpanProvider;
import ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider;
import ro.sync.ecss.extensions.api.ExtensionsBundle;
import ro.sync.ecss.extensions.api.StylesFilter;
import ro.sync.ecss.extensions.api.link.ElementLocatorProvider;
import ro.sync.ecss.extensions.commons.DefaultElementLocatorProvider;
import ro.sync.exml.editor.xmleditor.pageauthor.AuthorDnDListener;

public class SDFExtensionsBundle extends ExtensionsBundle {
 public String getDocumentTypeID() {
      return "Simple.Document.Framework.document.type";
 }

 public String getDescription() {
      return "A custom extensions bundle used for the " +
                       "Simple Document Framework document type";
 }

 public AuthorExtensionStateListener createAuthorExtensionStateListener() {
      return new SDFAuthorExtensionStateListener();
 }

 public SchemaManagerFilter createSchemaManagerFilter() {
      return new SDFSchemaManagerFilter();
 }

 public ElementLocatorProvider createElementLocatorProvider() {
      return new DefaultElementLocatorProvider();
 }

 public AuthorDnDListener createAuthorAWTDndListener() {
      return new SDFAuthorDndListener();
 }

 public AuthorReferenceResolver createAuthorReferenceResolver() {
      return new ReferencesResolver();
 }

 public StylesFilter createAuthorStylesFilter() {
      return new SDFStylesFilter();
 }

 public AuthorTableCellSpanProvider createAuthorTableCellSpanProvider() {
      return new TableCellSpanProvider();
 }

 public AuthorTableColumnWidthProvider createAuthorTableColumnWidthProvider() {
      return new TableColumnWidthProvider();
 }
}
```

SDFSchemaManagerFilter.java

```
package simple.documentation.framework;

import java.util.Iterator;
import java.util.List;

import ro.sync.contentcompletion.xml.CIAttribute;
import ro.sync.contentcompletion.xml.CIElement;
import ro.sync.contentcompletion.xml.CIValue;
import ro.sync.contentcompletion.xml.Context;
import ro.sync.contentcompletion.xml.ContextElement;
import ro.sync.contentcompletion.xml.SchemaManagerFilter;
import ro.sync.contentcompletion.xml.WhatAttributesCanGoHereContext;
import ro.sync.contentcompletion.xml.WhatElementsCanGoHereContext;
import ro.sync.contentcompletion.xml.WhatPossibleValuesHasAttributeContext;

public class SDFSchemaManagerFilter implements SchemaManagerFilter {

 @Override
 public List<CIValue> filterAttributeValues(List<CIValue> attributeValues,
       WhatPossibleValuesHasAttributeContext context) {
      return attributeValues;
 }

 @Override
 public List<CIAttribute> filterAttributes(List<CIAttribute> attributes,
       WhatAttributesCanGoHereContext context) {
   // If the element from the current context is the 'table' element add the
   // attribute named 'frame' to the list of default content
      // completion proposals
   ContextElement contextElement = context.getParentElement();
   if ("table".equals(contextElement.getQName())) {
     CIAttribute frameAttribute = new CIAttribute();
     frameAttribute.setName("frame");
     frameAttribute.setRequired(false);
     frameAttribute.setFixed(false);
     frameAttribute.setDefaultValue("void");
     attributes.add(frameAttribute);
   }
   return attributes;
 }

 @Override
 public List<CIValue> filterElementValues(List<CIValue> elementValues,
       Context context) {
      return elementValues;
 }

 @Override
 public List<CIElement> filterElements(List<CIElement> elements,
       WhatElementsCanGoHereContext context) {
```

```
   // If the element from the current context is the 'header'
      // element remove the 'td' element from the list of content
      // completion proposals and add the 'th' element.
   ContextElement contextElement = context.getElementStack().peek();
   if ("header".equals(contextElement.getQName())) {
     for (Iterator<CIElement> iterator = elements.iterator();
                    iterator.hasNext();) {
       CIElement element = iterator.next();
        // Remove the 'td' element
       if ("td".equals(element.getQName())) {
          elements.remove(element);
          break;
        }
      }
     // Insert the 'th' element in the list of content completion proposals
     CIElement thElement = new SDFElement();
     thElement.setName("th");
     elements.add(thElement);
   }
    return elements;
 }

 @Override
 public String getDescription() {
  return null;
 }
}
```

TableCellSpanProvider.java

```
package simple.documentation.framework;

public class TableCellSpanProvider
    implements AuthorTableCellSpanProvider {


  /**
   * Extracts the integer specifing what is the width
   * (in columns)  of the cell
   * representing in the table layout the cell element.
   */
  public Integer getColSpan(AuthorElement cell) {
    Integer colSpan = null;

    AttrValue attrValue = cell.getAttribute("column_span");
    if(attrValue != null) {
      // The attribute was found.
      String cs = attrValue.getValue();
      if(cs != null) {
        try {
          colSpan = new Integer(cs);
        } catch (NumberFormatException ex) {
          // The attribute value was not a number.
        }
      }
    }
    return colSpan;
  }

  /**
   * Extracts the integer specifing what is the
   * height (in rows) of the cell
   * representing in the table layout the cell element.
   */
  public Integer getRowSpan(AuthorElement cell) {
    Integer rowSpan = null;

    AttrValue attrValue = cell.getAttribute("row_span");
    if(attrValue != null) {
      // The attribute was found.
      String rs = attrValue.getValue();
      if(rs != null) {
        try {
          rowSpan = new Integer(rs);
        } catch (NumberFormatException ex) {
          // The attribute value was not a number.
        }
      }
    }
    return rowSpan;
  }

  /**
   * @return true considering the column specifications always available.
   */
  public boolean hasColumnSpecifications(AuthorElement tableElement) {
    return true;
  }

  /**
   * Ignored. We do not extract data from the
   * <code>table</code> element.
   */
  public void init(AuthorElement table) {
  }

  public String getDescription() {
    return
      "Implementation for the Simple Documentation Framework table layout.";
  }
}
```

TableColumnWidthProvider.java

```
package simple.documentation.framework.extensions;
import java.util.ArrayList;
import java.util.List;
import ro.sync.ecss.extensions.api.AuthorDocumentController;
import ro.sync.ecss.extensions.api.AuthorOperationException;
import ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider;
import ro.sync.ecss.extensions.api.WidthRepresentation;
import ro.sync.ecss.extensions.api.node.AttrValue;
import ro.sync.ecss.extensions.api.node.AuthorElement;

/**
 *
 * Simple Documentation Framework table column width provider.
 *
 */
public class TableColumnWidthProvider implements AuthorTableColumnWidthProvider {

 /**
  * Cols start offset
  */
 private int colsStartOffset;

 /**
  * Cols end offset
  */
 private int colsEndOffset;

 /**
  * Column widths specifications
  */
 private List<WidthRepresentation> colWidthSpecs = new ArrayList<WidthRepresentation>();

 /**
  * The table element
  */
 private AuthorElement tableElement;

 /**
  * @see ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider#commitColumnWidthModif
  */
 public void commitColumnWidthModifications(AuthorDocumentController authorDocumentControl
   WidthRepresentation[] colWidths, String tableCellsTagName) throws AuthorOperationExcept
  if ("td".equals(tableCellsTagName)) {
   if (colWidths != null && tableElement != null) {
    if (colsStartOffset >= 0 && colsEndOffset >= 0 && colsStartOffset < colsEndOffset) {
     authorDocumentController.delete(colsStartOffset,
       colsEndOffset);
    }
    String xmlFragment = createXMLFragment(colWidths);
    int offset = -1;
    AuthorElement[] header = tableElement.getElementsByLocalName("header");
    if (header != null && header.length > 0) {
     // Insert the cols elements before the 'header' element
     offset = header[0].getStartOffset();
    }
    if (offset == -1) {
```

```
    throw new AuthorOperationException("No valid offset to insert the columns width speci
   }
   authorDocumentController.insertXMLFragment(xmlFragment, offset);
  }
 }
}

/**
 * Creates the XML fragment representing the column specifications.
 *
 * @param widthRepresentations
 * @return The XML fragment as a string.
 */
private String createXMLFragment(WidthRepresentation[] widthRepresentations) {
 StringBuffer fragment = new StringBuffer();
 String ns = tableElement.getNamespace();
 for (int i = 0; i < widthRepresentations.length; i++) {
  WidthRepresentation width = widthRepresentations[i];
  fragment.append("<customcol");
  String strRepresentation = width.getWidthRepresentation();
  if (strRepresentation != null) {
   fragment.append(" width=\"" + width.getWidthRepresentation() + "\"");
  }
  if (ns != null && ns.length() > 0) {
   fragment.append(" xmlns=\"" + ns + "\"");
  }
  fragment.append("/>");
 }
 return fragment.toString();
}

/**
 * @see ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider#commitTableWidthModifi
 */
public void commitTableWidthModification(AuthorDocumentController authorDocumentControlle
  int newTableWidth, String tableCellsTagName) throws AuthorOperationException {
 if ("td".equals(tableCellsTagName)) {
  if (newTableWidth > 0) {
   if (tableElement != null) {
    String newWidth = String.valueOf(newTableWidth);

    authorDocumentController.setAttribute(
      "width",
      new AttrValue(newWidth),
      tableElement);
   } else {
    throw new AuthorOperationException("Cannot find the element representing the table.")
   }
  }
 }
}

/**
 * @see ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider#getCellWidth(ro.sync.e
 */
public List<WidthRepresentation> getCellWidth(AuthorElement cellElement, int colNumberSta
  int colSpan) {
 List<WidthRepresentation> toReturn = null;
 int size = colWidthSpecs.size();
 if (size >= colNumberStart && size >= colNumberStart + colSpan) {
  toReturn = new ArrayList<WidthRepresentation>(colSpan);
  for (int i = colNumberStart; i < colNumberStart + colSpan; i ++) {
   // Add the column widths
   toReturn.add(colWidthSpecs.get(i));
  }
 }
 return toReturn;
}

/**
 * @see ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider#getTableWidth(java.lan
 */
public WidthRepresentation getTableWidth(String tableCellsTagName) {
 WidthRepresentation toReturn = null;
 if (tableElement != null && "td".equals(tableCellsTagName)) {
  AttrValue widthAttr = tableElement.getAttribute("width");
  if (widthAttr != null) {
   String width = widthAttr.getValue();
   if (width != null) {
    toReturn = new WidthRepresentation(width, true);
   }
  }
 }
 return toReturn;
}

/**
 * @see ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider#init(ro.sync.ecss.exte
 */
public void init(AuthorElement tableElement) {
 this.tableElement = tableElement;
 AuthorElement[] colChildren = tableElement.getElementsByLocalName("customcol");
 if (colChildren != null && colChildren.length > 0) {
  for (int i = 0; i < colChildren.length; i++) {
   AuthorElement colChild = colChildren[i];
   if (i == 0) {
    colsStartOffset = colChild.getStartOffset();
   }
   if (i == colChildren.length - 1) {
    colsEndOffset = colChild.getEndOffset();
   }
   // Determine the 'width' for this col.
   AttrValue colWidthAttribute = colChild.getAttribute("width");
   String colWidth = null;
   if (colWidthAttribute != null) {
    colWidth = colWidthAttribute.getValue();
    // Add WidthRepresentation objects for the columns this 'customcol' specification
    // spans over.
```

```
     colWidthSpecs.add(new WidthRepresentation(colWidth, true));
    }
   }
  }
 }

 /**
  * @see ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider#isAcceptingFixedColumn
  */
 public boolean isAcceptingFixedColumnWidths(String tableCellsTagName) {
  return true;
 }

 /**
  * @see ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider#isAcceptingPercentageC
  */
 public boolean isAcceptingPercentageColumnWidths(String tableCellsTagName) {
  return true;
 }

 /**
  * @see ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider#isAcceptingProportiona
  */
 public boolean isAcceptingProportionalColumnWidths(String tableCellsTagName) {
  return true;
 }

 /**
  * @see ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider#isTableAcceptingWidth(
  */
 public boolean isTableAcceptingWidth(String tableCellsTagName) {
  return "td".equals(tableCellsTagName);
 }

 /**
  * @see ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider#isTableAndColumnsResiz
  */
 public boolean isTableAndColumnsResizable(String tableCellsTagName) {
  return "td".equals(tableCellsTagName);
 }

 /**
  * @see ro.sync.ecss.extensions.api.Extension#getDescription()
  */
 public String getDescription() {
  return "Implementation for the Simple Documentation Framework table layout.";
 }
}
```

ReferencesResolver.java

```
package simple.documentation.framework;

import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;

import javax.xml.transform.sax.SAXSource;

import org.apache.log4j.Logger;
import org.xml.sax.EntityResolver;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;

import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.AuthorReferenceResolver;
import ro.sync.ecss.extensions.api.node.AttrValue;
import ro.sync.ecss.extensions.api.node.AuthorElement;
import ro.sync.ecss.extensions.api.node.AuthorNode;

/**
 * Resolver for content referred by elements named 'ref' with a
 *    'location' attribute.
 */
public class ReferencesResolver implements AuthorReferenceResolver {

  /**
   * Logger for logging.
   */
  private static Logger logger = Logger.getLogger(
          ReferencesResolver.class.getName());

  /**
   * Verifies if the handler considers the node to have references.
   *
   * @param node The node to be analyzed.
   * @return <code>true</code> if it is has references.
   */
  public boolean hasReferences(AuthorNode node) {
    boolean hasReferences = false;
    if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
      AuthorElement element = (AuthorElement) node;
      if ("ref".equals(element.getLocalName())) {
        AttrValue attrValue = element.getAttribute("location");
        hasReferences = attrValue != null;
      }
    }
    return hasReferences;
  }

  /**
   * Returns the name of the node that contains the expanded referred content.
   *
   * @param node The node that contains references.
   * @return The display name of the node.
   */
  public String getDisplayName(AuthorNode node) {
    String displayName = "ref-fragment";
    if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
      AuthorElement element = (AuthorElement) node;
      if ("ref".equals(element.getLocalName())) {
        AttrValue attrValue = element.getAttribute("location");
        if (attrValue != null) {
          displayName = attrValue.getValue();
        }
      }
    }
    return displayName;
  }

  /**
   * Resolve the references of the node.
   *
   * The returning SAXSource will be used for creating the referred content
   * using the parser and source inside it.
   *
   * @param node        The clone of the node.
   * @param systemID  The system ID of the node with references.
   * @param authorAccess The author access implementation.
   * @param entityResolver The entity resolver that can be used to resolve:
   *
   * <ul>
   *  <li>Resources that are already opened in editor.
   *  For this case the InputSource will contains the editor content.</li>
   *  <li>Resources resolved through XML catalog.</li>
   * </ul>
   *
   * @return The SAX source including the parser and the parser's input source.
   */
  public SAXSource resolveReference(
      AuthorNode node,
      String systemID,
      AuthorAccess authorAccess,
      EntityResolver  entityResolver) {
    SAXSource saxSource = null;

    if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
      AuthorElement element = (AuthorElement) node;
      if ("ref".equals(element.getLocalName())) {
        AttrValue attrValue = element.getAttribute("location");
        if (attrValue != null) {
          String attrStringVal = attrValue.getValue();
          try {
            URL absoluteUrl = new URL(new URL(systemID),
                    authorAccess.correctURL(attrStringVal));

            InputSource inputSource = entityResolver.resolveEntity(null,
                    absoluteUrl.toString());
            if(inputSource == null) {
              inputSource = new InputSource(absoluteUrl.toString());
            }
            XMLReader xmlReader = authorAccess.newNonValidatingXMLReader();
            xmlReader.setEntityResolver(entityResolver);

            saxSource = new SAXSource(xmlReader, inputSource);
          } catch (MalformedURLException e) {
            logger.error(e, e);
          } catch (SAXException e) {
            logger.error(e, e);
          } catch (IOException e) {
            logger.error(e, e);
          }
        }
      }
    }

    return saxSource;
  }

  /**
   * Get an unique identifier for the node reference.
   *
   * The unique identifier is used to avoid resolving the references
   *     recursively.
   *
   * @param node The node that has reference.
   * @return  An unique identifier for the reference node.
   */
  public String getReferenceUniqueID(AuthorNode node) {
    String id = null;
    if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
      AuthorElement element = (AuthorElement) node;
      if ("ref".equals(element.getLocalName())) {
        AttrValue attrValue = element.getAttribute("location");
        if (attrValue != null) {
          id = attrValue.getValue();
        }
      }
    }
    return id;
  }

  /**
   * Return the systemID of the referred content.
   *
   * @param node The reference node.
   * @param authorAccess The author access.
   *
   * @return The systemID of the referred content.
   */
  public String getReferenceSystemID(AuthorNode node,
          AuthorAccess authorAccess) {
    String systemID = null;
    if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
      AuthorElement element = (AuthorElement) node;
      if ("ref".equals(element.getLocalName())) {
        AttrValue attrValue = element.getAttribute("location");
        if (attrValue != null) {
          String attrStringVal = attrValue.getValue();
          try {
            URL absoluteUrl = new URL(node.getXMLBaseURL(),
                authorAccess.correctURL(attrStringVal));
            systemID = absoluteUrl.toString();
          } catch (MalformedURLException e) {
            logger.error(e, e);
          }
        }
      }
    }
    return systemID;
  }

  /**
   * Verifies if the references of the given node must be refreshed
   * when the attribute with the specified name has changed.
   *
   * @param node The node with the references.
   * @param attributeName The name of the changed attribute.
   * @return <code>true</code> if the references must be refreshed.
   */
  public boolean isReferenceChanged(AuthorNode node, String attributeName) {
    return "location".equals(attributeName);
  }

  /**
   * @return The description of the author extension.
   */
  public String getDescription() {
    return "Resolves the 'ref' references";
  }
}
```

CustomRule.java

```
package simple.documentation.framework;

import org.xml.sax.Attributes;

import ro.sync.ecss.extensions.api.DocumentTypeCustomRuleMatcher;

public class CustomRule implements
                DocumentTypeCustomRuleMatcher {
  /**
   * Checks if the root namespace is the one
   * of our documentation framework.
   */
  public boolean matches(
      String systemID,
      String rootNamespace,
      String rootLocalName,
      String doctypePublicID,
      Attributes rootAttributes) {

    return
      "http://www.oxygenxml.com/sample/documentation".equals(rootNamespace);
  }

  public String getDescription() {
    return
  "Checks if the current Document Type Association is matching the document.";
  }
}
```

DefaultElementLocatorProvider.java

```
package ro.sync.ecss.extensions.commons;

import org.apache.log4j.Logger;

import ro.sync.ecss.extensions.api.link.ElementLocator;
import ro.sync.ecss.extensions.api.link.ElementLocatorException;
import ro.sync.ecss.extensions.api.link.ElementLocatorProvider;
import ro.sync.ecss.extensions.api.link.IDTypeVerifier;

/**
 * Default implementation for locating elements based on a given link.
 * Depending on the link structure the following cases are covered:
 * - xinclude element scheme :  element(/1/2) see
 *        http://www.w3.org/TR/2003/REC-xptr-element-20030325/
 * - ID based links : the link represents the value of an attribute of type ID
 */
public class DefaultElementLocatorProvider implements ElementLocatorProvider {
  /** * Logger for logging. */
  private static Logger logger = Logger.getLogger(
          DefaultElementLocatorProvider.class.getName());
  /**
   * @see ro.sync.ecss.extensions.api.link.ElementLocatorProvider#
   *     getElementLocator(ro.sync.ecss.extensions.api.link.IDTypeVerifier,
   *     java.lang.String)
   */
  public ElementLocator getElementLocator(IDTypeVerifier idVerifier,
          String link) {
    ElementLocator elementLocator = null;
    try {
      if(link.startsWith("element(")){
        // xpointer element() scheme
        elementLocator = new XPointerElementLocator(idVerifier, link);
      } else {
        // Locate link element by ID
        elementLocator = new IDElementLocator(idVerifier, link);
      }
    } catch (ElementLocatorException e) {
      logger.warn("Exception when create element locator for link: "
          + link + ". Cause: " + e, e);
    }
    return elementLocator;
  }

  /**
   * @see ro.sync.ecss.extensions.api.Extension#getDescription()
   */
  public String getDescription() {
    return
    "Default implementation for locating elements based on a given link. \n" +
    "The following cases are covered: xinclude element scheme "
        + "and ID based links.";
  }
}
```

XPointerElementLocator.java

```
package ro.sync.ecss.extensions.commons;

import java.util.Stack;
import java.util.StringTokenizer;

import org.apache.log4j.Logger;

import ro.sync.ecss.extensions.api.link.Attr;
import ro.sync.ecss.extensions.api.link.ElementLocator;
import ro.sync.ecss.extensions.api.link.ElementLocatorException;
import ro.sync.ecss.extensions.api.link.IDTypeVerifier;

/**
 * Element locator for links that have the one of the following pattern:
 * <ul>
 *   <li>element(elementID) - locate the element with the same id</li>
 *   <li>element(/1/2/5) - A child sequence appearing alone identifies an
 *   element by means of stepwise navigation, which is directed by a
 *   sequence of integers separated by slashes (/); each integer n locates
 *   the nth child element of the previously located element. </li>
 *   <li>element(elementID/3/4) - A child sequence appearing after an
 *   NCName identifies an element by means of stepwise navigation,
 *   starting from the element located by the given name.</li>
 * </ul>
 *
 */
public class XPointerElementLocator extends ElementLocator {

  /**
   * Logger for logging.
   */
  private static Logger logger = Logger.getLogger(
          XPointerElementLocator.class.getName());

  /**
```

```
   * Verifies if a given attribute is of a type ID.
   */
  private IDTypeVerifier idVerifier;

  /**
   * XPointer path, the path to locate the linked element.
   */
  private String[] xpointerPath;

  /**
   * The stack with indexes in parent of the current iterated elements.
   */
  private Stack currentElementIndexStack = new Stack();

  /**
   * The number of elements in xpointer path.
   */
  private int xpointerPathDepth;

  /**
   * If true then the XPointer path starts with an element ID.
   */
  private boolean startWithElementID = false;

  /**
   * The depth of the current element in document, incremented in startElement.
   */
  private int startElementDepth = 0;

  /**
   * Depth in document in the last endElement event.
   */
  private int endElementDepth = 0;

  /**
   * The index in parent of the previous iterated element. Set in endElement().
   */
  private int lastIndexInParent;

  /**
   * Constructor.
   *
   * @param idVerifier Verifies if an given attribute is of type ID.
   * @param link The link that gives the element position.
   * @throws ElementLocatorException  When the link format is not supported.
   **/
  public XPointerElementLocator(IDTypeVerifier idVerifier, String link)
          throws ElementLocatorException {
    super(link);
    this.idVerifier = idVerifier;

    link = link.substring("element(".length(), link.length() - 1);

    StringTokenizer stringTokenizer = new StringTokenizer(link, "/", false);
    xpointerPath = new String[stringTokenizer.countTokens()];
    int i = 0;
    while (stringTokenizer.hasMoreTokens()) {
      xpointerPath[i] = stringTokenizer.nextToken();
      boolean invalidFormat = false;

      // Empty xpointer component is not supported
      if(xpointerPath[i].length() == 0){
        invalidFormat = true;
      }

      if(i > 0){
        try {
          Integer.parseInt(xpointerPath[i]);
        } catch (NumberFormatException e) {
          invalidFormat = true;
        }
      }

      if(invalidFormat){
        throw new ElementLocatorException(
        "Only the element() scheme is supported when locating XPointer links."
        + "Supported formats: element(elementID), element(/1/2/3),
            element(elemID/2/3/4).");
      }
      i++;
    }

    if(Character.isDigit(xpointerPath[0].charAt(0))){
      // This is the case when xpointer have the following pattern /1/5/7
      xpointerPathDepth = xpointerPath.length;
    } else {
      // This is the case when xpointer starts with an element ID
      xpointerPathDepth = -1;
      startWithElementID  = true;
    }
  }

  /**
   * @see ro.sync.ecss.extensions.api.link.ElementLocator#endElement(
   *     java.lang.String, java.lang.String, java.lang.String)
   */
  public void endElement(String uri, String localName, String name) {
    endElementDepth = startElementDepth;
    startElementDepth --;
    lastIndexInParent = ((Integer)currentElementIndexStack.pop()).intValue();
  }

  /**
   * @see ro.sync.ecss.extensions.api.link.ElementLocator#startElement(
   *    java.lang.String, java.lang.String, java.lang.String,
   *    ro.sync.ecss.extensions.api.link.Attr[])
   */
  public boolean startElement(String uri, String localName,
          String name, Attr[] atts) {
    boolean linkLocated = false;
    // Increase current element document depth
    startElementDepth  ++;

    if (endElementDepth != startElementDepth) {
      // The current element is the first child of the parent
      currentElementIndexStack.push(new Integer(1));
    } else {
      // Another element in the parent element
      currentElementIndexStack.push(new Integer(lastIndexInParent + 1));
    }

    if (startWithElementID) {
      // This the case when xpointer path starts with an element ID.
      String xpointerElement = xpointerPath[0];
      for (int i = 0; i < atts.length; i++) {
        if(xpointerElement.equals(atts[i].getValue())){
          if(idVerifier.hasIDType(
              localName, uri, atts[i].getQName(), atts[i].getNamespace())){
            xpointerPathDepth = startElementDepth + xpointerPath.length - 1;
            break;
          }
        }
      }
    }

    if(xpointerPathDepth == startElementDepth){
      // check if xpointer path matches with the current element path
      linkLocated = true;
      try {
        int xpointerIdx = xpointerPath.length - 1;
        int stackIdx = currentElementIndexStack.size() - 1;
        int stopIdx = startWithElementID ? 1 : 0;
        while (xpointerIdx >= stopIdx && stackIdx >= 0) {
          int xpointerIndex = Integer.parseInt(xpointerPath[xpointerIdx]);
          int currentElementIndex = ((Integer)currentElementIndexStack.
                  get(stackIdx)).intValue();
          if(xpointerIndex != currentElementIndex) {
            linkLocated = false;
            break;
          }

          xpointerIdx--;
          stackIdx--;
        }

      } catch (NumberFormatException e) {
        logger.warn(e,e);
      }
    }
    return linkLocated;
  }
}
```

IDElementLocator.java

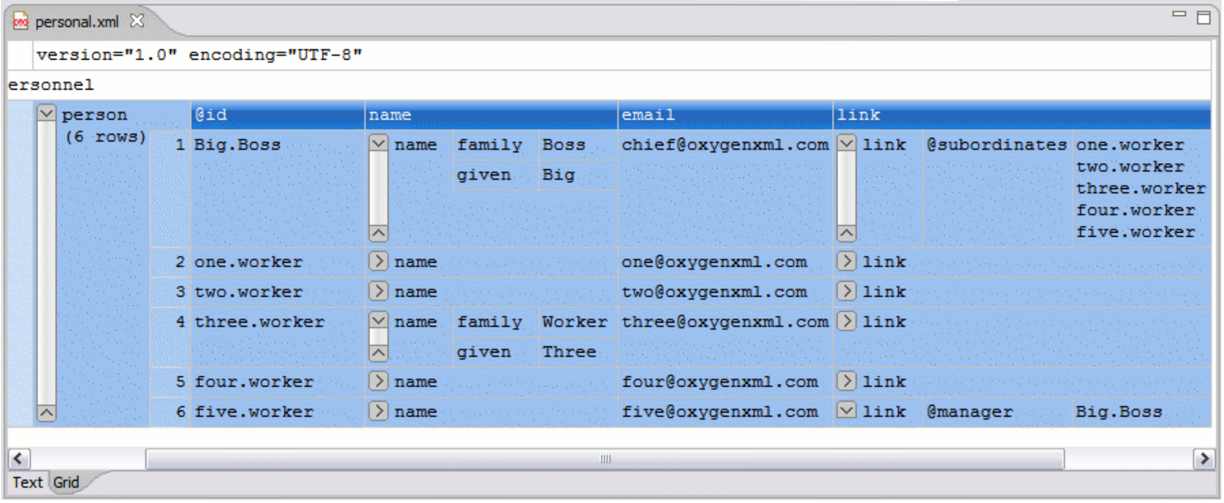
```
package ro.sync.ecss.extensions.commons;

import ro.sync.ecss.extensions.api.link.Attr;
import ro.sync.ecss.extensions.api.link.ElementLocator;
import ro.sync.ecss.extensions.api.link.ExtensionUtil;
import ro.sync.ecss.extensions.api.link.IDTypeVerifier;

/**
 * Implementation of an ElementLocator that treats the link as the value of an
 *     attribute with the type ID.
 */
public class IDElementLocator extends ElementLocator {

  /**
   * Class able to tell if a given attribute is of type ID.
   */
  private IDTypeVerifier idVerifier;

  /**
   * Constructor.
   *
   * @param idVerifier It tells us if an attribute is of type ID.
   * @param link The link used to identify an element.
   */
  public IDElementLocator(IDTypeVerifier idVerifier, String link) {
    super(link);
    this.idVerifier = idVerifier;
  }

  /**
   * @see ro.sync.ecss.extensions.api.link.ElementLocator#endElement(
   *         java.lang.String, java.lang.String, java.lang.String)
   */
  public void endElement(String uri, String localName, String name) {
    // Nothing to do.
  }

  /**
   * @see ro.sync.ecss.extensions.api.link.ElementLocator#startElement(
   *    java.lang.String, java.lang.String, java.lang.String,
   *    ro.sync.ecss.extensions.api.link.Attr[])
   */
  public boolean startElement(String uri, String localName,
          String name, Attr[] atts) {
    boolean elementFound = false;
    for (int i = 0; i < atts.length; i++) {
      if (link.equals(atts[i].getValue())) {
        if("xml:id".equals(atts[i].getQName())) {
          // xml:id attribute
          elementFound = true;
        } else {
          // check if attribute has ID type
          String attrLocalName =
            ExtensionUtil.getLocalName(atts[i].getQName());
          String attrUri = atts[i].getNamespace();
          if (idVerifier.hasIDType(localName, uri, attrLocalName, attrUri)) {
            elementFound = true;
          }
        }
      }
    }

    return elementFound;
  }
}
```

Chapter 8. Grid Editor

Introduction

In the grid editor the XML document is displayed as a structured grid of nested tables in which the text content can be modified by non technical users without editing directly the XML tags. The tables can be expanded and collapsed with a mouse click to show or hide the elements of the document as needed. Also the document structure can be changed easily with drag and drop operations on the grid components. The tables can be zoomed using Ctrl-+ , Ctrl-- , Ctrl-0 or Ctrl-mouse wheel.

Figure 8.1. The Grid Editor



personal.xml

version="1.0" encoding="UTF-8"

ersonnel

person (6 rows)

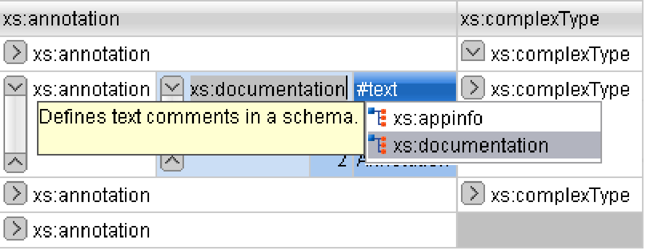
	@id	name	email	link
1	Big.Boss	name: family Boss given Big	chief@oxygenxml.com	link: @subordinates one.worker two.worker three.worker four.worker five.worker
2	one.worker	name	one@oxygenxml.com	link
3	two.worker	name	two@oxygenxml.com	link
4	three.worker	name: family Worker given Three	three@oxygenxml.com	link
5	four.worker	name	four@oxygenxml.com	link
6	five.worker	name	five@oxygenxml.com	link: @manager Big.Boss

You can switch between the text tab and the grid tab of the editor panel with the two buttons *Text* and *Grid* available at the bottom of the editor panel.

If the edited document is associated with a schema (DTD, XML Schema, Relax NG, etc.), the editor offers content completion for the element and attributes names and values. If you choose to insert an element that has required content, it will be inserted automatically including the subtree of needed elements and attributes.

To display the content completion popup you have to start editing, for example by double clicking the cell. When editing, pressing **CTRL SPACE** redisplays the popup.

Figure 8.2. Content Completion in Grid Editor



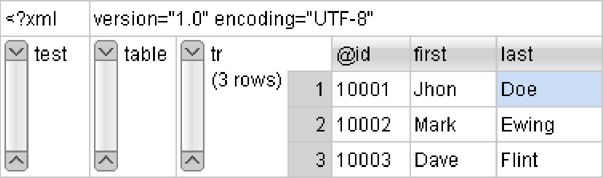
xs:annotation	xs:complexType
xs:annotation	xs:complexType
xs:annotation xs:documentation #text	xs:complexType
xs:annotation	xs:complexType
xs:annotation	

Defines text comments in a schema.

Layouts: Grid and Tree

The grid editor has two modes for the layout. The default one is the "grid" layout. This smart layout of the grid editor detects the recurring elements in the XML document and creates tables having as columns the children (including the attributes) of these elements. In this way it is possible to have tables nested in other tables, reflecting the structure of your document.

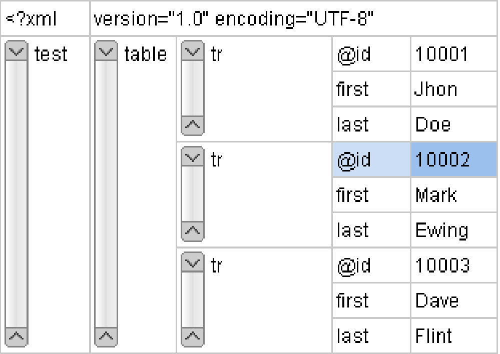
Figure 8.3. Grid Layout



<?xml	version="1.0" encoding="UTF-8"					
test	table	tr (3 rows)		@id	first	last
			1	10001	Jhon	Doe
			2	10002	Mark	Ewing
			3	10003	Dave	Flint

The other layout mode is "tree"-like. This layout does not create any table, it presents the structure of the document directly.

Figure 8.4. Tree Layout



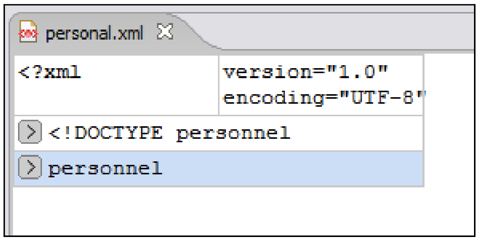
<?xml	version="1.0" encoding="UTF-8"			
test	table	tr	@id	10001
			first	Jhon
			last	Doe
		tr	@id	10002
			first	Mark
			last	Ewing
		tr	@id	10003
			first	Dave
			last	Flint

You can switch between the two modes using the contextual menu: Grid mode/Tree mode

Navigating the grid

When you open a document first in the grid tab, the document is collapsed so that it shows just the root element and its attributes.

Figure 8.5. Initial configuration of grid tab



<?xml	version="1.0" encoding="UTF-8"
<!DOCTYPE personnel	
personnel	

The grid disposition of the node names and values are very similar to a web form or a dialog. The same set of key shortcuts used to select dialog components are used in the grid. For instance moving to the next editable value in a table row is done using the **TAB** key. Moving to the previous cell employs the **SHIFT+TAB** key. Changing a value assumes pressing the **ENTER** key or start typing directly the new value, and, when the editing is finished, pressing **ENTER** again to commit the data into the document.

The arrows and the **PAGE UP/DOWN** keys can be used for navigation. By pressing **SHIFT** while using these keys you can create a selection zone. To add other nodes that are not close to this zone, you can use the mouse and the **CTRL** (**COMMAND** on Mac OS X) key.

The following key combination may be used to scroll the grid:

- **CTRL + UP** Scrolls the grid upwards
- **CTRL + DOWN** Scrolls the grid downwards
- **CTRL + LEFT** Scrolls the grid to the left
- **CTRL + RIGHT** Scrolls the grid to the right

A left arrow sign displayed to the left of the node name indicates that this node has child nodes. You can click this sign to display the children. The expand/collapse actions can be also invoked by pressing the **NumPad + PLUS** and **NumPad + MINUS** keys.

A set of expand/collapse actions can be accessed from the submenu Expand/Collapse of the contextual menu.

Expand All Action

Expands the selection and all its children.

Collapse All Action

Collapses the selection and all its children.

Expand Children Action

Expands all the children of the selection but not the selection.

Collapse Children Action

Collapses all the children of the selection but not the selection.

Collapse Others

Collapses all the siblings of the current selection but not the selection.

Specific Grid Actions

In order to access these actions you can click the column header and choose from the contextual menu the item: Table

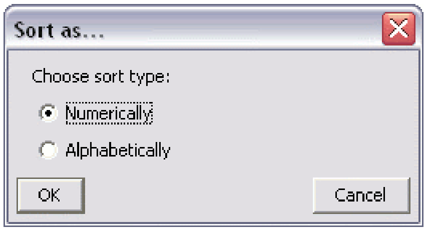
Sorting a Table Column

You can sort the table by a specific column. The sorting can be either ascending or descending.

The icons for this pair of actions are:

The sorting result depends on the data type of the column content and it can be different in case of number or text information. The editor analyses automatically the content and decides what type of sorting to apply. If there is present a mixed set of values in the column, a dialog will be displayed allowing to choose the correct type.

Figure 8.6. Sort Type Dialog



Sort as...

Choose sort type:

● Numerically

○ Alphabetically

Inserting a row to a table

You can add a row by either a copy/paste operation over a row, or directly, by invoking the action from the contextual menu: Table → Insert row

The icon is:

A shorter way of inserting a new row is to move the selection over the row header, and then to press ENTER. The row header is the zone in the left of the row that holds the row number. The inserted row will be below the selection.

Inserting a column into a table

You can insert a column after the selected one, using the action from the contextual menu: Table → Insert column

The icon is:

Clearing the content of a column

You can clear all the cells from a column, using the action from the contextual menu: Table → Clear content

Adding nodes

Using the contextual menu you can add nodes before, after, or as last child of the currently selected node.

The sub-menus containing detailed actions are: Insert beforeInsert afterAppend child

Duplicating nodes

A quicker way of creating new nodes is to duplicate the existing ones.

The action is available in the contextual menu: Duplicate

Refresh layout

When using drag and drop to reorganize the document, the resulted layout may be different from the expected one. For instance, the layout may contain a set of sibling tables that could be joined together. To force the layout to be re-computed you can use the Refresh action 🔄 .

The action is available in the contextual menu: Refresh selected

Start editing a cell value

You can simply press **ENTER** after you have selected the grid cell.

Stop editing a cell value

You can either press **ENTER** when already in cell editing.

To cancel the editing without saving in the document the current changes, you have to press the **ESC** key.

Drag and Drop(DnD) in the Grid Editor

The DnD features of the grid editor make easy the arrangement of the different sections in your XML document.

Using DnD you can:

- Copy or move a set of nodes.
- Change the order of columns in the tables.
- Move the rows from the tables.

These operations are available for single selection and multiple selection.

Note that when dragging the editor paints guide-lines showing accepted locations where the nodes can be dropped.

Nodes can be dragged outside the grid editor and text from other applications can be dropped inside the grid. See Copy and Paste in the Grid Editor for details.

Copy and Paste in the Grid Editor

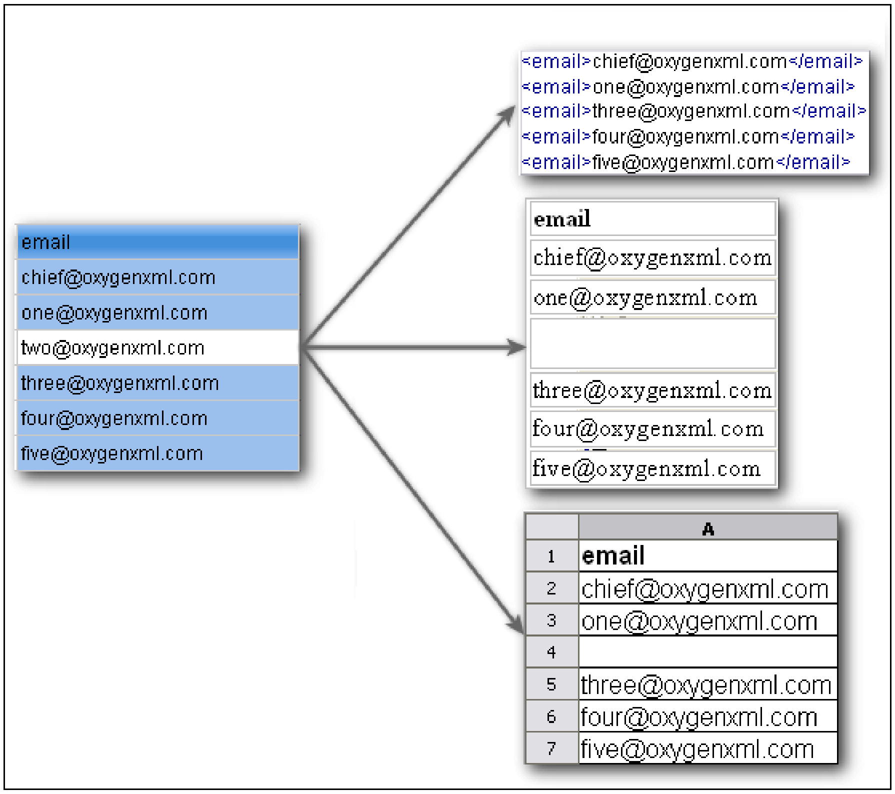
The selection in the grid is a bit complex relative to the selection in a text component. It consists of a current selected cell and additional selected cells. These additional cells are either "hand picked" by the user using the mouse, or are implied by the current selected cell. To be more specific, let's consider you click the name of the column - this becomes the current selected cell; the editor automatically extends the selection so it contains also all the cells from that column. The current selected cell is painted with a color that is different from the rest of the selection.

You can select discontinuous regions of nodes and place them in the clipboard using the copy action. Pasting these nodes may be done in two ways, relative to the current selected cell: by default as brother, just below (after) , or as last child of the selected cell.

The paste as child action is available in the contextual menu: Paste as Child

The copied nodes from the grid can be pasted also into the text editor or other applications. When copying from grid into the text editor or other text based applications the inserted string represents the nodes serialization. The nodes from tables can be copied using HTML or RTF in table format. The resulting cells contain only the concatenated values of the text nodes.

Figure 8.7. Copying from grid to other editors



email
chief@oxygenxml.com
one@oxygenxml.com
two@oxygenxml.com
three@oxygenxml.com
four@oxygenxml.com
five@oxygenxml.com

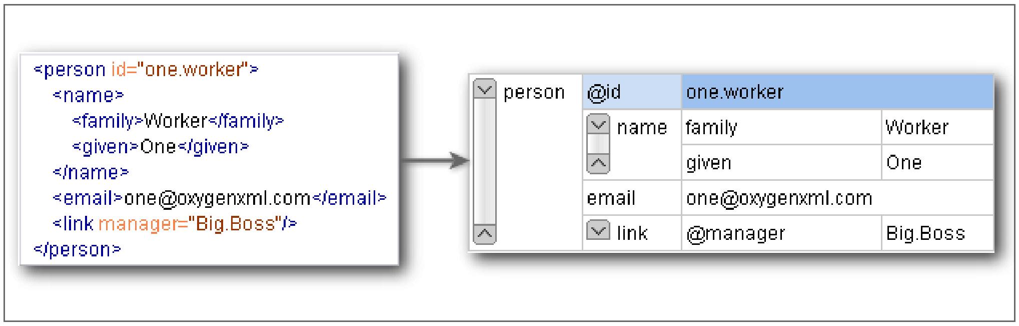
```
<email>chief@oxygenxml.com</email>
<email>one@oxygenxml.com</email>
<email>three@oxygenxml.com</email>
<email>four@oxygenxml.com</email>
<email>five@oxygenxml.com</email>
```

email
chief@oxygenxml.com
one@oxygenxml.com
three@oxygenxml.com
four@oxygenxml.com
five@oxygenxml.com

	A
1	email
2	chief@oxygenxml.com
3	one@oxygenxml.com
4	
5	three@oxygenxml.com
6	four@oxygenxml.com
7	five@oxygenxml.com

In the grid editor you can paste wellformed xml content or tab separated values from other editors. If you paste xml content the result will be the insertion of the nodes obtained by parsing this content.

Figure 8.8. Copying XML data into grid

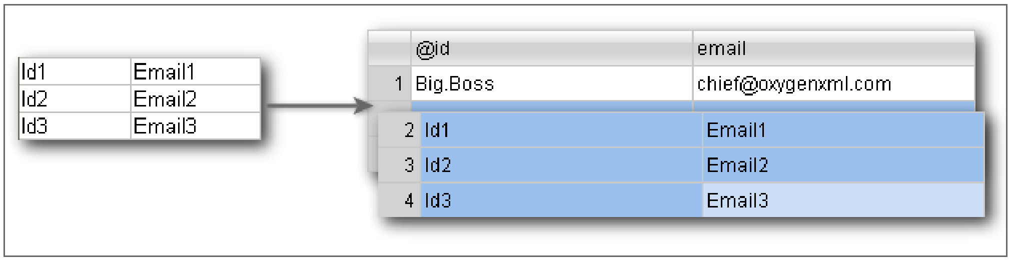


```
<person id="one.worker">
   <name>
      <family>Worker</family>
      <given>One</given>
   </name>
   <email>one@oxygenxml.com</email>
   <link manager="Big.Boss"/>
</person>
```

person	@id	one.worker	
	name	family	Worker
		given	One
	email	one@oxygenxml.com	
	link	@manager	Big.Boss

If the pasted text contains multiple lines of tab separated values it can be considered as a matrix of values. By pasting this matrix of values into the grid editor the result will be a matrix of cells. If the operation is performed inside existing cells the values from these cells will be overwritten and new ones will be created if needed. This is useful for example when trying to transfer data from Excel like editors into grid editor.

Figure 8.9. Copying tab separated values into grid



Id1	Email1
Id2	Email2
Id3	Email3

	@id	email
1	Big.Boss	chief@oxygenxml.com
2	Id1	Email1
3	Id2	Email2
4	Id3	Email3

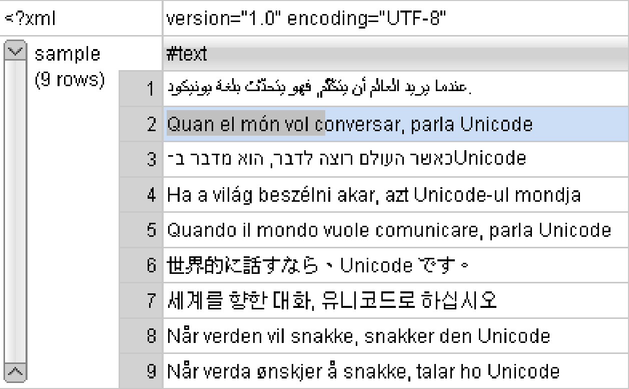
Bidirectional Text Support in the Grid Editor

If you are editing documents employing a different text orientation you can change the way text is rendered and edited in the grid cells.

For this, you can use the shortcut **CTRL SHIFT O** to toggle from the default left to right text orientation to the right to left orientation.

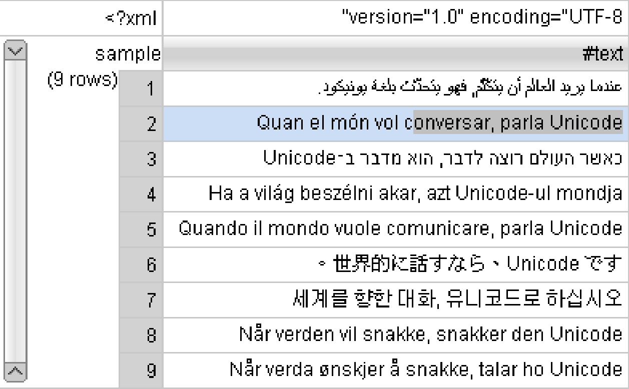
Note that this change applies only to the text from the cells, not to the layout of the grid editor.

Figure 8.10. Default left to right text orientation



<?xml		version="1.0" encoding="UTF-8"
sample (9 rows)		#text
	1	عندما يريد العالم أن يتكلّم، فهو يتحدّث بلغة يونيكود.
	2	Quan el món vol conversar, parla Unicode
	3	כאשר העולם רוצה לדבר, הוא מדבר ב-Unicode
	4	Ha a világ beszélni akar, azt Unicode-ul mondja
	5	Quando il mondo vuole comunicare, parla Unicode
	6	世界的に話すなら、Unicode です。
	7	세계를 향한 대화, 유니코드로 하십시오
	8	Når verden vil snakke, snakker den Unicode
	9	Når verda ønskjer å snakke, talar ho Unicode

Figure 8.11. Right to left text orientation



<?xml		"version="1.0" encoding="UTF-8
sample (9 rows)		#text
	1	عندما يريد العالم أن يتكلّم، فهو يتحدّث بلغة يونيكود.
	2	Quan el món vol conversar, parla Unicode
	3	כאשר העולם רוצה לדבר, הוא מדבר ב-Unicode
	4	Ha a világ beszélni akar, azt Unicode-ul mondja
	5	Quando il mondo vuole comunicare, parla Unicode
	6	。世界的に話すなら、Unicode です
	7	세계를 향한 대화, 유니코드로 하십시오
	8	Når verden vil snakke, snakker den Unicode
	9	Når verda ønskjer å snakke, talar ho Unicode

Chapter 9. Transforming documents

XML is designed to store, carry, and exchange data, not to display data. When we want to view the data we must either have an XML compliant user agent or transform it to a format that can be read by other user agents. This process is known as transformation.

Status messages generated during transformation are displayed in the Console view.

Output formats

Within the current version of <oXygen/> you can transform your XML documents to the following formats without having to exit from the application. For transformation to formats not listed simply install the tool chain required to perform the transformation and process the xml files created with <oXygen/> in accordance with the processor instructions.

PDF
: Adobe Portable Document Format (PDF) is a compact binary file format that can be viewed and printed by anyone, anywhere across a broad range of hardware and software using the free PDF Viewer from Adobe [http://www.adobe.com/products/acrobat/readstep.html].

PS
: PostScript is the leading printing technology from Adobe [http://www.adobe.com:80/products/postscript/main.html] for high-quality, best-in-class printing solutions ranging from desktop devices to the most advanced digital presses, platemakers, and large format image setters in the world. Postscript files can be viewed using viewers such as GhostScript, but are more commonly created as a prepress format.

TXT
: Text files are Plain ASCII Text and can be opened in any text editor or word processor.

XML
: XML stands for eXtensible Markup Language and is a W3C [http://www.w3c.org/XML/] standard markup language, much like HTML, which was designed to describe data. XML tags are not predefined in XML. You must define your own tags. XML uses a Document Type Definition (DTD), an XML Schema or a Relax NG schema to describe the data. XML with a DTD, XML Schema or Relax NG schema is designed to be self-descriptive. XML is not a replacement for HTML. XML and HTML were designed with different goals:

- XML was designed to describe data and to focus on what data is.
- HTML was designed to display data and to focus on how data looks.
- HTML is about displaying information, XML is about describing information.

XHTML
: XHTML stands for eXtensible HyperText Markup Language, a W3C [http://www.w3c.org/MarkUp/] standard. XHTML is aimed to replace HTML. While almost identical to HTML 4.01, XHTML is a stricter and cleaner version of HTML. XHTML is HTML defined as an XML application.

All formatting during a transformation is provided under the control of an Extensible Stylesheet (XSLT). Specifying the appropriate XSLT enables transformation to the above formats and preparation of output files for specific user agent viewing applications, including:

HTML
: HTML stands for Hyper Text Markup Language and is a W3C Standard [http://www.w3c.org/MarkUp/] for the World Wide Web. HTML is a text file containing small markup tags. The markup tags tell the Web browser how to display the page. An HTML file must have an htm or html file extension. An HTML file can be created using a simple text editor.

HTML Help — Microsoft HTML Help [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/vsconHH1Start.asp?frame=true] is the standard help system for the Windows platform. Authors can use HTML Help to create online help for a software application or to create content for a multimedia title or Web site. Developers can use the HTML Help API to program a host application or hook up context-sensitive help to an application.

JavaHelp — JavaHelp software is a full-featured, platform-independent, extensible help system from Sun Microsystems [http://java.sun.com/products/javahelp/index.html] that enables developers and authors to incorporate online help in applets, components, applications, operating systems, and devices. JavaHelp is a free product and the binaries for JavaHelp can be redistributed.

Eclipse Help — Eclipse Help is the help system incorporated in the Eclipse platform [http://www.eclipse.org/] that enables Eclipse plugin developers to incorporate online help in their plugins.

Many other target formats are possible, these are the most popular. The basic condition for transformation to any format is that your source document is well-formed. Always, make sure that the XSL used for the transformation is the right one according to the desired output format and with the input source definition. For example, if you want to transform to HTML format using a DocBook html stylesheet, your source xml document should respect the DocBook DTD.

An XSL stylesheet specifies the presentation of a class of XML documents by describing how an instance of the class is transformed into an output document by using special formatting vocabulary.

XSL consists of three parts:

XSL Transformations (XSLT) — XSLT is a language for transforming XML documents.

XML Path (XPath) Language — XPath is an expression language used by XSLT to access or refer parts of an XML document. (XPath is also used by the XML Linking specification).

XSL Formatting Objects (XSL:FO) — XSL:FO is an XML vocabulary for specifying formatting semantics.

<oXygen/> supports XSLT/XPath version 1.0 using Saxon 6.5.5, Xalan, Xsltproc, MSXML (3.0, 4.0, .NET) and XSLT/XPath 2.0 by using Saxon 9.1.0.7 B, Saxon 9.1.0.7 SA and Saxon.NET. Also the validation is done in function of the stylesheet version.

Transformation scenario

Before transforming the current edited XML document in <oXygen/> you must define a transformation scenario to apply to that document. A scenario is a set of values for various parameters defining a transformation. It is not related to any particular document but to a document type:

Scenarios that apply to XML files — Such a scenario contains the location of an XSLT stylesheet that is applied on the edited XML document and other transform parameters.

Scenarios that apply to XSLT files — Such a scenario contains the location of an XML document that the edited XSLT stylesheet is applied on and other transform parameters.

Scenarios that apply to XQuery files — Such a scenario contains the location of an XML source that the edited XQuery file is applied on and other transform parameters. When the XML source is a native XML database the XML source field of the scenario is empty because the XML data is read with XQuery functions like *document()*. When the XML source is a local XML file the URL of the file is specified in the XML input field of the scenario.

A scenario can be created at document type level or at global level. The scenarios defined at document type level are available only for the documents that match that document type. The global scenarios are available for any document.

In order to apply a transformation scenario one has to press the *Apply Transformation Scenario* button from the *Transformation* toolbar.

Batch transformation

Alternatively, a transform action can be applied on a batch of files from the Project view's contextual menu without having to open the files:

- ▶Apply Transformation Scenario - applies to each selected file the transformation scenario associated to that file. If the currently processed file does not have an associated transformation scenario then a warning is displayed in the *Warnings* view to let the user know about it.
- Transform with... - allows the user to select one transformation scenario to be applied to each one of the currently selected files.

Built-in transformation scenarios

If the Apply Transformation Scenario button from the *Transformation* toolbar is pressed, currently there is no scenario associated with the edited document and the edited document contains a "xml-stylesheet" processing instruction referring to a XSLT stylesheet (commonly used for display in Internet browsers), then <oXygen/> will prompt the user and offer the option to associate the document with a default scenario containing in the *XSL URL* field the URL from the *href* attribute of the processing instruction. This scenario will have the "Use xml-stylesheet declaration" checkbox set by default, will use Saxon as transformation engine, will perform no FO processing and will store the result in a file with the same URL as the edited document except the extension which will be changed to html. The name and path will be preserved because the output file name is specified with the help of two editor variables: ${cfd} and ${cfn}.

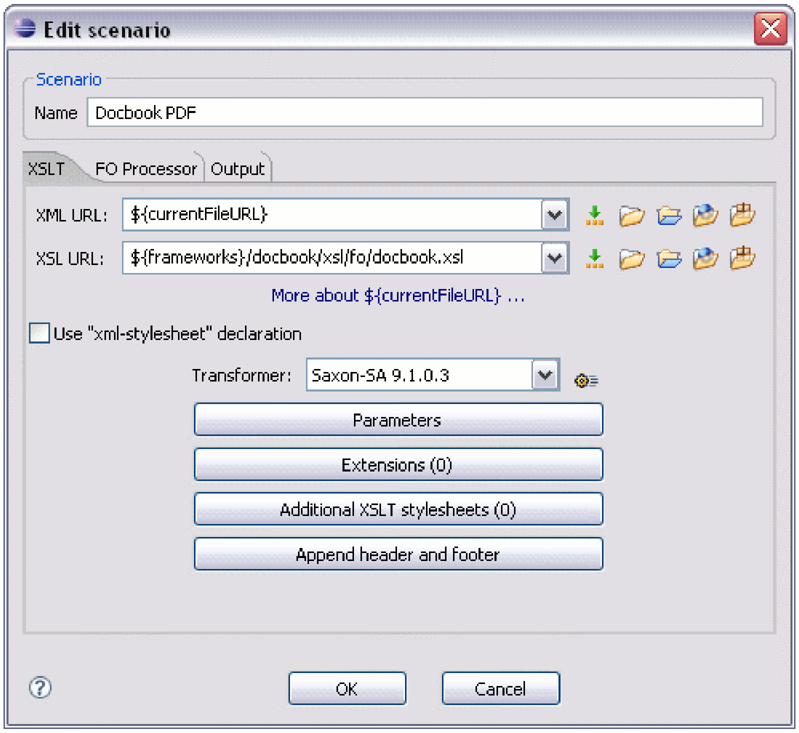
<oXygen/> comes with preconfigured built-in scenarios for usual transformations that enable the user to obtain quickly the desired output: associate one of the built-in scenarios with the current edited document and then apply the scenario with just one click.

Defining a new transformation scenario

The Configure Scenario dialog is used to associate a scenario from the list of all scenarios with the edited document by selecting an entry from the list. The dialog is opened by pressing the Configure Transformation Scenario button on the *Transformation* toolbar of the document view. Once selected the scenario will be applied with only one click on the Apply Transformation button on the same toolbar. Pressing the Apply Transformation button before associating a scenario with the edited document will invoke first the Configure Scenario dialog and then apply the selected scenario.

Open the Configure Transformation dialog by selecting XML → Configure transformation scenario. (**Alt+Shift+T C (Cmd+Alt+T C on Mac OS)**) Complete the dialog as follows:

Figure 9.1. The Configure Transformation Dialog - XSLT Tab



Name: Docbook PDF

XML URL: ${currentFileURL}

XSL URL: ${frameworks}/docbook/xsl/fo/docbook.xsl

More about ${currentFileURL} ...

☐ Use "xml-stylesheet" declaration

Transformer: Saxon-SA 9.1.0.3

XML URL

Specifies an input XML file to be used for the transformation. Please note that this URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, then the editor will try to use the file directly.

Note

If the transformation engine is Saxon 9 and a custom URI resolver is configured for Saxon 9 in Preferences then the XML input of the transformation is passed to that URI resolver.

The following buttons are shown immediately after the input field:

Insert Editor Variables

Opens a pop-up menu allowing to introduce special <oXygen/> editor variables in the XML URL field.

Browse for local file

Opens a local file browser dialog allowing to select a local file name for the text field.

Browse for remote file

Opens a URL browser dialog allowing to select a remote file name for the text field.

Browse for archived file

Opens a zip archive browser dialog allowing to select a file name from a zip archive that will be inserted in the text field.

Open in editor

Opens the file with the path specified in the text field in an editor panel.

XSL URL

Specifies an input XSL file to be used for the transformation. Please note that this URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, then the editor will try to use the file directly.

The above set of browsing buttons are available also for this input.

Use "xml-stylesheet" declaration

Use the stylesheet declared with an "xml-stylesheet" declaration instead of the stylesheet specified in the XSL URL field. By default this checkbox is not selected and the transformation applies the XSLT stylesheet specified in the XSL URL field. If it is checked the scenario applies the stylesheet specified explicitly in the XML document with the xml-stylesheet processing instruction.

Transformer

This combo box contains all the transformer engines available for applying the stylesheet. These are the built-in engines and the external engines defined in the user preferences. If you want to change the default selected engine just select other engine from the drop down list of the combo box. For XQuery/XSLT files only, if no validation scenario is associated, the transformer engine will be used in validation process, if has validation support.

Parameters

Opens the dialog for configuring the XSLT parameters. In this dialog you set any global XSLT parameters of the main stylesheet set in the *XSL URL* field or of the additional stylesheets set with the button *Additional XSLT stylesheets*.

Append header and footer

Opens a dialog for specifying a URL for a header HTML file added at the beginning of the result of an HTML transformation and a URL for a footer HTML file added at the end of the HTML result of the transformation.

Additional XSLT stylesheets

Opens the dialog for adding XSLT stylesheets which are applied on the result of the main stylesheet specified in the XSL URL field. This is useful when a chain of XSLT stylesheets must be applied to the input XML document.

Extensions

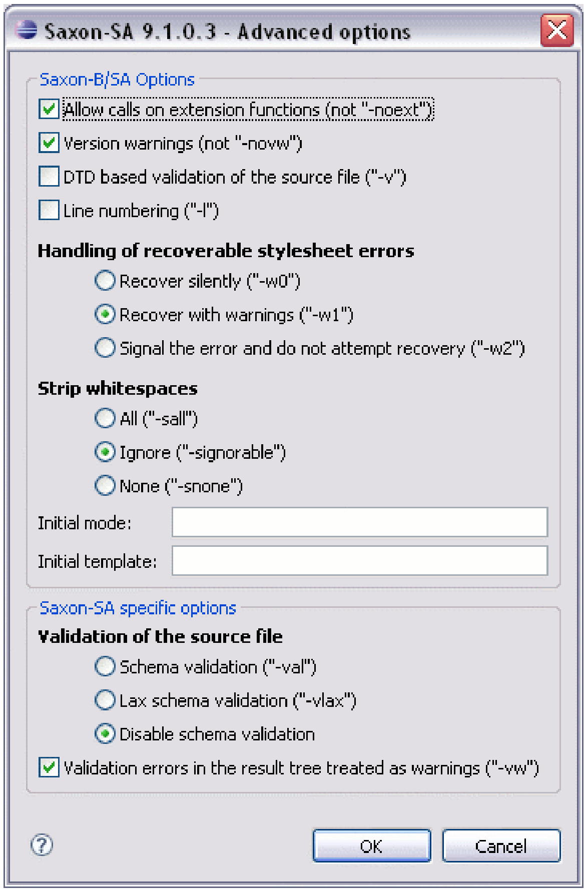
Opens the dialog for configuring the XSLT/XQuery extension jars or classes which define extension Java functions or extension XSLT elements used in the XSLT/XQuery transformation.

Advanced options

Configure advanced options specific for the Saxon B/SA engine. They are the same options as the ones set in the user preferences but they are configured as a specific set of transformation options for each transformation scenario. By default if you do not set a specific value in the transformation scenario each advanced option has the same value as the global option with the same name set in the user preferences.

The advanced options include two options that are not available globally in the user preferences: the initial XSLT template and the initial XSLT mode of the transformation. They are Saxon specific options that allow imposing the name of the first XSLT template that starts the XSLT transformation or the initial mode of transformation.

Figure 9.2. The advanced options of Saxon SA/Saxon B



Saxon-SA 9.1.0.3 - Advanced options

Saxon-B/SA Options

☑ Allow calls on extension functions (not "-noext")

☑ Version warnings (not "-novw")

□ DTD based validation of the source file ("-v")

□ Line numbering ("-l")

Handling of recoverable stylesheet errors

○ Recover silently ("-w0")

● Recover with warnings ("-w1")

○ Signal the error and do not attempt recovery ("-w2")

Strip whitespaces

○ All ("-sall")

● Ignore ("-signorable")

○ None ("-snone")

Initial mode:

Initial template:

Saxon-SA specific options

Validation of the source file

○ Schema validation ("-val")

○ Lax schema validation ("-vlax")

● Disable schema validation

☑ Validation errors in the result tree treated as warnings ("-vw")

The advanced options specific for Saxon SA/Saxon B are:

Allow calls on extension functions
If checked the stylesheet is allowed to call external Java functions.

Version warnings
If checked display a warning when it is applied to an XSLT 1.0 stylesheet.

DTD based validation of the source file
If checked the source XML file is validated against the declared DTD

Line numbering
Include the line number in errors for the

Handling of recoverable stylesheet errors
Policy for handling recoverable errors in the stylesheet: Allows the user to choose how dynamic errors will be handled. Either one of the following options can be selected: recover silently, recover with warnings or signal the error and do not attempt recovery.

Strip whitespaces
Strip whitespaces feature can be one of the three options: All, Ignorable, None.

All
strips all whitespace text nodes from source documents before any further processing, regardless of any xsl:strip-space declarations in the stylesheet, or any xml:space attributes in the source document.

Ignorable

strips all ignorable whitespace text nodes from source documents before any further processing, regardless of any xsl:strip-space declarations in the stylesheet, or any xml:space attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.

None

strips no whitespace before further processing. (However, whitespace will still be stripped if this is specified in the stylesheet using xsl:strip-space).

Initial mode

Specifies to the transformer the initial template mode

Initial template

Specifies the name of the initial template to the transformer. When specified, the XML input URL for the transformation scenario is optional.

Validation of the source file

Available only for Saxon SA.

Schema validation

This mode requires an XML Schema and determines whether source documents should be parsed with schema-validation enabled.

Lax schema validation

This mode determines whether source documents should be parsed with schema-validation enabled if an XML Schema is provided.

Disable schema validation

This determines whether source documents should be parsed with schema-validation disabled.

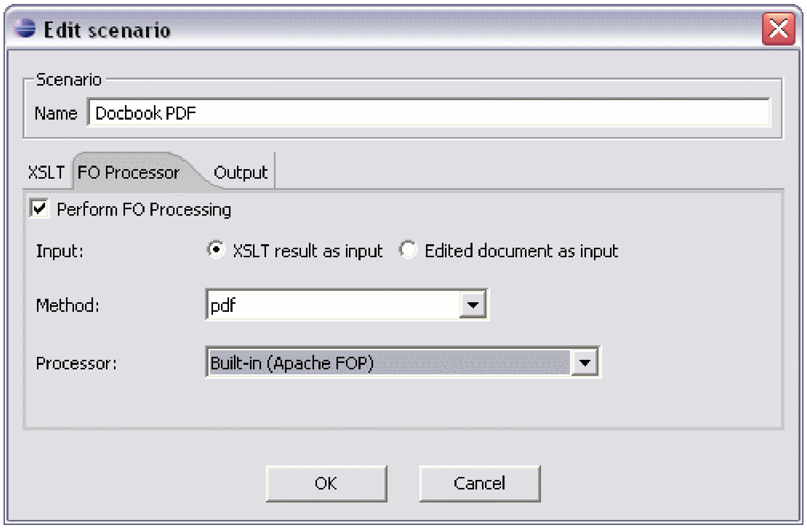
Validation errors in the results tree treated as warnings

Available only for Saxon SA. If checked, all validation errors are treated as warnings, otherwise they are treated as fatal.

When creating a scenario that applies to an XML file, <oXygen/> fills the XML URL with the default variable "${currentFile}". This means the input for the transformation is taken from the currently edited file. You can modify this value to other file path. This is the case of currently editing a section from a large document, and you want the transformation to be performed on the main document, not the section. You can specify in this case either a full absolute path: file:/c:/project/docbook/test.xml or a path relative to one of the editor variables, like the current file directory: `${cfdu}/test.xml`.

When the scenario applies to XSL files, the field XSL URL is containing ${currentFile}. Just like in the XML case, you can specify here the path to a master stylesheet. The path can be configured using the editor variables.

Figure 9.3. The Configure Transformation Dialog - FO Processor Tab



Name: Docbook PDF

☑ Perform FO Processing

Input: ● XSLT result as input ○ Edited document as input

Method: pdf

Processor: Built-in (Apache FOP)

Checkbox *Perform FO Processing* — Enable or disable applying an FO processor (either the built-in Apache FOP engine or an external engine defined in Preferences) during the transformation.

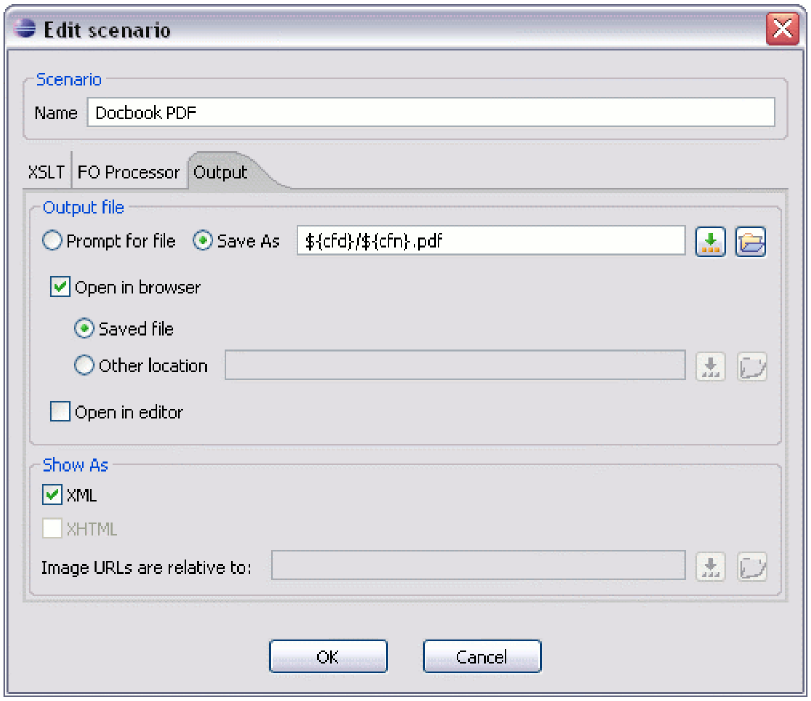
Radio button *XSLT result as input* — The FO processor is applied to the result of the XSLT transformation defined on the XSLT tab of the dialog.

Radio button *Edited document as input* — The FO processor is applied directly to the current edited document.

Combo box *Method* — The output format of the FO processing: PDF, PostScript or plain text.

Combo box *Processor* — The FO processor, which can be the built-in Apache FOP processor or an external processor.

Figure 9.4. The Configure Transformation Dialog - Output Tab



Name: Docbook PDF

○ Prompt for file ● Save As ${cfd}/${cfn}.pdf

☑ Open in browser

● Saved file

○ Other location

□ Open in editor

☑ XML

□ XHTML

Image URLs are relative to:

Radio button *Prompt for file*

At the end of the transformation a file browser dialog will be displayed for specifying the path and name of the file which will store the transformation result.

Text field *Save As*

The path of the file where it will be stored the transformation result. The path can include special <oXygen/> editor variables.

Check box *Open in browser*

If this is checked <oXygen/> will open automatically the transformation result in a browser application specific for the type of that result (HTML/XHTML, PDF, text).

Radio button *Saved file*

When *Open in browser* is selected this button can be selected to specify that <oXygen/> should open automatically at the end of the transformation the file specified in the *Save As* text field.

Radio button *Other location*

When *Open in browser* is selected this button can be used to specify that <oXygen/> should not open the file specified in the *Save As* text field, it should open the file specified in the text field of the *Other location* radio button. The file path can include special <oXygen/> editor variables.

Check box *Open in editor*

When this is checked the transformation result set in the *Save As* field is opened in a new editor panel in <oXygen/> with the appropriate built-in editor type: if the result is an XML file it is opened with the built-in XML editor, if it is an XSL-FO file it is opened with the built-in FO editor, etc.

Check box *Show As XHTML*

It is enabled only when *Open in browser* is disabled. If this is checked <oXygen/> will display the transformation result in a built-in XHTML browser panel at the bottom of the <oXygen/> window.

Important

When transforming very large documents you should be aware that enabling this feature will result in a very long time necessary for rendering the transformation result in the XHTML result viewer panel. This drawback appears due to the built-in Java XHTML browser implementation. In this situations if you wish to see the XHTML result of the transformation you should use an external browser by checking the *Open in browser* checkbox.

Check box *Show As XML*

If this is checked <oXygen/> will display the transformation result in an XML viewer panel at the bottom of the <oXygen/> window with syntax highlight specific for XML documents.

Check box *Show As SVG*

If this is checked <oXygen/> will display the transformation result in a SVG viewer panel at the bottom of the <oXygen/> window by rendering the result as a SVG image.

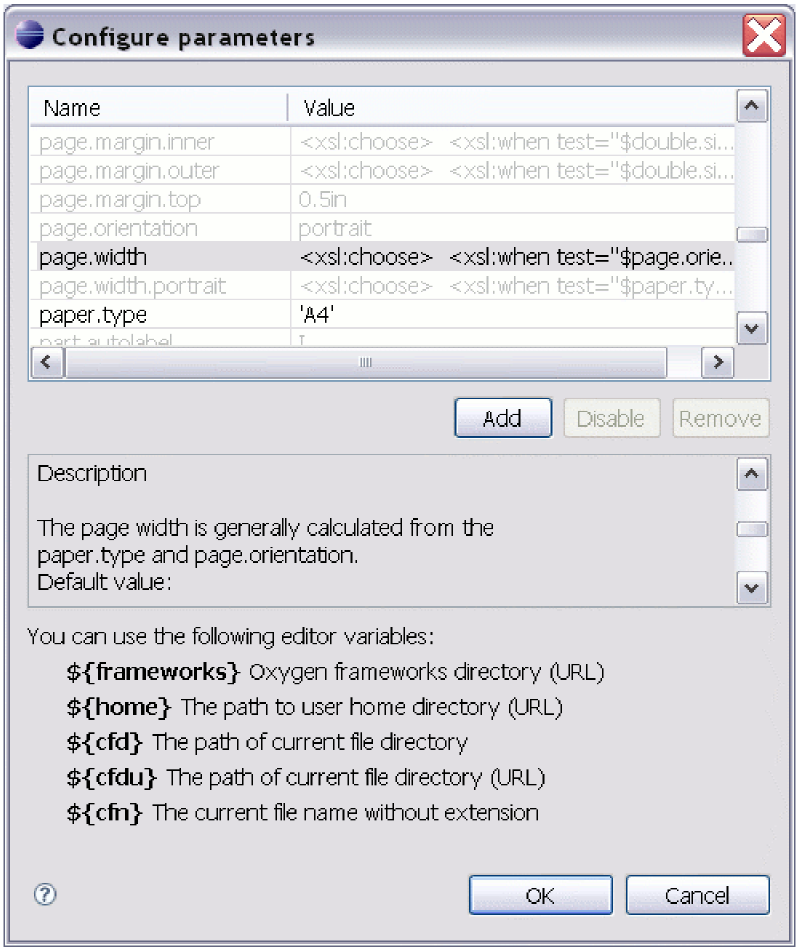
Text field *Image URLs are relative to*

If *Show As XHTML* is checked this text field specifies the path for resolving image paths contained in the transformation result.

XSLT Stylesheet Parameters

The global parameters of the XSLT stylesheet used in the transformation scenario are configured from the dialog available from the *Parameters* button:

Figure 9.5. Configure parameters dialog



Name	Value
page.margin.inner	<xsl:choose> <xsl:when test="$double.si...
page.margin.outer	<xsl:choose> <xsl:when test="$double.si...
page.margin.top	0.5in
page.orientation	portrait
page.width	<xsl:choose> <xsl:when test="$page.orie..
page.width.portrait	<xsl:choose> <xsl:when test="$paper.ty...
paper.type	'A4'
part.autolabel	I

Description

The page width is generally calculated from the paper.type and page.orientation.
Default value:

You can use the following editor variables:

${frameworks} Oxygen frameworks directory (URL)
${home} The path to user home directory (URL)
${cfd} The path of current file directory
${cfdu} The path of current file directory (URL)
${cfn} The current file name without extension

The table presents all the parameters of the XSLT stylesheet and all imported and included stylesheets with their current values. If a parameter value was not edited then the table presents its default value. The bottom panel presents the default value of the parameter selected in the table, a description of the parameter if it is available and the system ID of the stylesheet that declares it.

For setting the value of a parameter declared in the stylesheet in a namespace, for example:

```
<xsl:param name="p:param" xmlns:p="namespace">default</xsl:param>
```

use the following expression in the *Name* column of the *Parameters* dialog:

```
{namespace}param
```

The buttons of the dialog have the following functions:

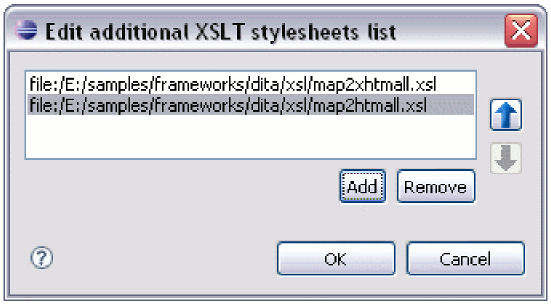
Add   Add a new parameter to the list.

The editor variables displayed at the bottom of the dialog (${frameworks}, ${home}, ${cfd}, etc) can be used in the values of the parameters to make the value independent of the location of the XSLT stylesheet or the XML document.

Additional XSLT Stylesheets

The list of additional XSLT stylesheets can be edited in the dialog opened by the button "Additional XSLT Stylesheets".

Figure 9.6. Edit additional XSL stylesheets list dialog



file:/E:/samples/frameworks/dita/xsl/map2xhtmall.xsl
file:/E:/samples/frameworks/dita/xsl/map2htmall.xsl

Add Adds a stylesheet in the "Additional XSLT stylesheets" list using a file browser dialog , also you can type an editor variable in the file name field of the browser dialog. The name of the stylesheet will be added in the list after the current selection.

New Opens a dialog in which you can type the name of a stylesheet. The name is considered relative to the URL of the current edited XML document. You can use editor variables in the name of the stylesheet. The name of the stylesheet will be added in the list after the current selection.

Remove Deletes the selected stylesheet from the "Additional XSLT stylesheets" list.

Up Move the selected stylesheet up in the list.

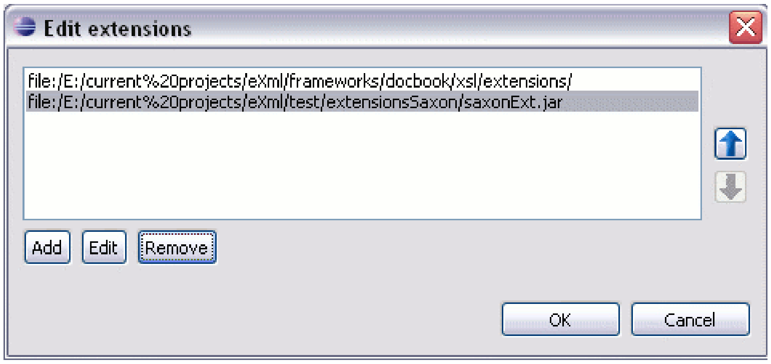
Down Move the selected stylesheet down in the list.

The path specified in the URL text field can include special <oXygen/> editor variables.

XSLT/XQuery Extensions

The edit extensions dialog is used to specify the jars and classes containing extension functions called from the XSLT/XQuery file of the current transformation scenario.

Figure 9.7. The XSLT/XQuery Extension Edit Dialog



file:/E:/current%20projects/eXml/frameworks/docbook/xsl/extensions/
file:/E:/current%20projects/eXml/test/extensionsSaxon/saxonExt.jar

An extension function called from the XSLT or XQuery file of the current transformation scenario will be searched in the specified extensions in the order of the list displayed in the dialog. For changing the order of the items the user must select the item that must be moved to other position in the list and press the ⬆ up and ⬇ down buttons.

Creating a Transformation Scenario

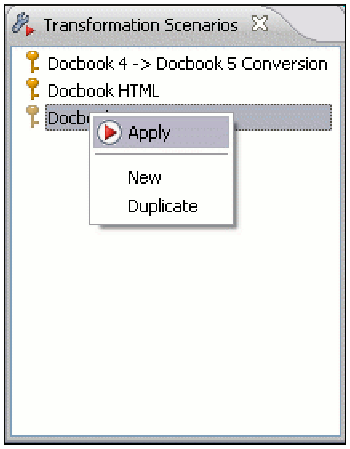
Use the following procedure to create a scenario.

1. Select XML → Configure transformation scenario (**Alt+Shift+T C (Cmd+Alt+T C on Mac OS)**) to open the Configure Transformation dialog.
2. Click the Duplicate Scenario button of the dialog to create a copy of the current scenario.
3. Click in the Name field and type a new name.
4. Click OK or Transform Now to save the scenario.

Transformation Scenarios view

The list of transformation scenarios may be easier to manage for some users as a list presented in a dockable and floating view called *Transformation Scenarios*.

Figure 9.8. The Scenarios view



The actions available on the right click menu allow the same operations as in the dialog Configure Transformation Scenario: creating, editing, executing, duplicating and removing a transformation scenario.

XSL-FO processors

The <oXygen/> installation package is distributed with the Apache FOP [http://xml.apache.org/fop/index.html] (Formatting Objects Processor) for rendering your XML documents to PDF. FOP is a print and output independent formatter driven by XSL Formatting Objects. FOP is implemented as a Java application that reads a formatting object tree and renders the resulting pages to a specified output.

Tip

To include PNG images in the final PDF document you need the JIMI [http://java.sun.com/products/jimi/] or JAI [http://java.sun.com/products/java-media/jai/] libraries. For TIFF images you need the JAI [http://java.sun.com/products/java-media/jai/] library. For PDF images you need the *fop-pdf-images* library [http://www.jeremias-maerki.ch/download/fop/pdf-images/]. These libraries are not bundled with <oXygen/> (JIMI and JAI due to Sun's licensing). Using them is as easy as downloading them and creating a external FO processor based on the built-in FOP libraries and the extension library. The external FO processor created in Preferences will have a command line like:

```
java -cp "${oxygenInstallDir}/lib/xercesImpl.jar:
${oxygenInstallDir}/lib/fop.jar:${oxygenInstallDir}/lib/
avalon-framework-4.2.0.jar:
${oxygenInstallDir}/lib/batik-all-1.7.jar:${oxygenInstallDir}/lib/
commons-io-1.3.1.jar:
${oxygenInstallDir}/lib/xmlgraphics-commons-1.3.1.jar:
${oxygenInstallDir}/lib/commons-logging-1.0.4.jar:
${oxygenInstallDir}/lib/saxon9sa.jar:${oxygenInstallDir}/lib/
saxon9-dom.jar:
${oxygenInstallDir}/lib/xalan.jar:${oxygenInstallDir}/lib/
serializer.jar:
${oxygenInstallDir}/lib/resolver.jar:${oxygenInstallDir}/lib/
fop-pdf-images-1.3.jar:
${oxygenInstallDir}/lib/PDFBox-0.7.3.jar"
org.apache.fop.cli.Main -fo ${fo} -${method} ${out}
```

You need to add to the classpath JimiProClasses.zip for JIMI and jai_core.jar, jai_codec.jar and mlibwrapper_jai.jar for JAI. For the JAI package you also need to include the directory containing the native libraries (mlib_jai.dll and mlib_jai_mmx.dll on Windows) in the PATH system variable.

The MacOS X version of the JAI library can be downloaded from http://www.apple.com/downloads/macosx/apple/java3dandjavaadvancedimagingupdate.html. In order to use it, install the downloaded package.

Other FO processors can be configured in the Preferences -> FO Processors panel.

Add a font to the built-in FOP

If an XML document is transformed to PDF using the built-in Apache FOP processor but it contains some Unicode characters that cannot be rendered by the default PDF fonts then a font that is capable to render these characters must be configured and embedded in the PDF result.

Locate font

First, you have to find out the name of a font that has the glyphs for the special characters you used. One font that covers the majority of characters, including Japanese, Cyrillic and Greek, is Arial Unicode MS. In the following is described how to embed the true type fonts in the output PDF. Embedding the fonts is necessary to ensure your document is portable.

On Windows the fonts are located into the `C:\Windows\Fonts` directory. On Mac they are placed in `/Library/Fonts`. To install a new font on your system is enough to copy it in the Fonts directory.

Generate font metrics file

Generate a FOP font metrics file from the TrueType font file. This example reads the Windows Arial Unicode MS file and generates an `arialuni.xml` font metrics file in the current directory. FOP includes an utility application for this task.

I assume you have opened a terminal or command line console and changed the working directory to the oxygen install directory. The FOP files are stored in the lib subdirectory of the Oxygen install directory.

Create the following script file in the Oxygen installation directory. The relative paths specified in the following script file are relative to the Oxygen installation directory so if you decide to create it in other directory you have to adapt the file paths.

For the Mac OS X: `ttfConvert.sh`

```
#!/bin/sh
export LIB=lib
export CMD="java -cp
"$LIB/fop.jar:$LIB/avalon-framework-4.2.0.jar:$LIB/xercesImpl.jar"
export CMD=$CMD org.apache.fop.fonts.apps.TTFReader"
export FONT_DIR='/Library/Fonts'
$CMD $FONT_DIR/Arialuni.ttf Arialuni.xml
```

For Linux: `ttfConvert.sh`

```
#!/bin/sh
export LIB=lib
export CMD="java -cp
"$LIB/fop.jar:$LIB/avalon-framework-4.2.0.jar:$LIB/xercesImpl.jar"
export CMD=$CMD org.apache.fop.fonts.apps.TTFReader"
export FONT_DIR='/Library/Fonts'
$CMD $FONT_DIR/Arialuni.ttf Arialuni.xml
```

For Windows: `ttfConvert.bat`

```
set LIB=lib
set CMD="java -cp
"%LIB%\fop.jar;%LIB%\avalon-framework-4.2.0.jar;%LIB%\xercesImpl.jar"
set CMD=%CMD% org.apache.fop.fonts.apps.TTFReader"
set FONT_DIR=C:\Windows\Fonts
%CMD% %FONT_DIR%\Arialuni.ttf Arialuni.xml
```

The `FONT_DIR` can be different on your system. Make sure it points to the correct font directory. If java executable is not in the `PATH` you will have to specify the full path for java.

Execute the script. On Linux and Mac OS X you have to use **sh ttfConvert.sh** from the command line.

Note

If Oxygen was installed by an administrator user and now it is used by a standard user who does not have write permission in the Oxygen installation folder (for example on Windows Vista or Linux) then the output location of the font metrics file should be a directory where the user has write permission, for example:

```
%CMD% %FONT_DIR%\Arialuni.ttf C:\temp_dir\Arialuni.xml
```

If the font has bold and italic variants, you will have to convert those also. For this you can modify the script, by adding two more lines:

```
$CMD $FONT_DIR/Arialuni-Bold.ttf Arialuni-Bold.xml
                      $CMD $FONT_DIR/Arialuni-Italic.ttf Arialuni-Italic.xml
```

In our case the font Arial Unicode MS is not having a Bold and Italic variant, so you will leave the script unchanged.

Register font to FOP configuration

Create a file and name it for example `fopConfiguration.xml`.

```
<fop version="1.0">
  <base>file:/C:/path/to/FOP/font/metrics/files/</base>
  <source-resolution>72</source-resolution>
  <target-resolution>72</target-resolution>
  <default-page-settings height="11in" width="8.26in"/>

  <renderers>
    <renderer mime="application/pdf">
      <filterList>
        <value>flate</value>
      </filterList>
      <fonts>
          <font metrics-url="Arialuni.xml" kerning="yes"
                embed-url="file:/Library/Fonts/Arialuni.ttf">
              <font-triplet name="Arialuni" style="normal"
                    weight="normal"/>
          </font>
      </fonts>
    </renderer>
  </renderers>
</fop>
```

The *embed-url* attribute points to the TTF file to be embedded. You have to specify it using the URL convention. The *metrics-url* attribute points to the font metrics file with a path relative to the *base* element. The triplet refers to the unique combination of name, weight, and style (italic) for each variation of the font. In our case is just one triplet, but if the font had variants, we would have to specify one for each variant. Here is an hypothetic example for the Arial Unicode if it had italic and bold variants:

```
<fop version="1.0">
  ...
    <fonts>
        <font metrics-url="Arialuni.xml" kerning="yes"
              embed-url="file:/Library/Fonts/Arialuni.ttf">
            <font-triplet name="Arialuni" style="normal"
                  weight="normal"/>
        </font>
        <font metrics-url="Arialuni-Bold.xml" kerning="yes"
              embed-url="file:/Library/Fonts/Arialuni-Bold.ttf">
            <font-triplet name="Arialuni" style="normal"
                  weight="bold"/>
        </font>
        <font metrics-url="Arialuni-Italic.xml" kerning="yes"
              embed-url="file:/Library/Fonts/Arialuni-Italic.ttf">
            <font-triplet name="Arialuni" style="italic"
                  weight="normal"/>
        </font>
    </fonts>
  ...
</fop>
```

More details about the FOP configuration file are available on http://xmlgraphics.apache.org/fop/0.93/configuration.htmlthe FOP website.

Set FOP configuration file in Oxygen

Go to menu Options → Preferences → XML → XSLT / FO / XQuery → FO Processors

Click the browse button near *Configuration file for the built-in FOP* text field and locate the `fopConfiguration.xml` file.

Click on the OK button to accept the changes.

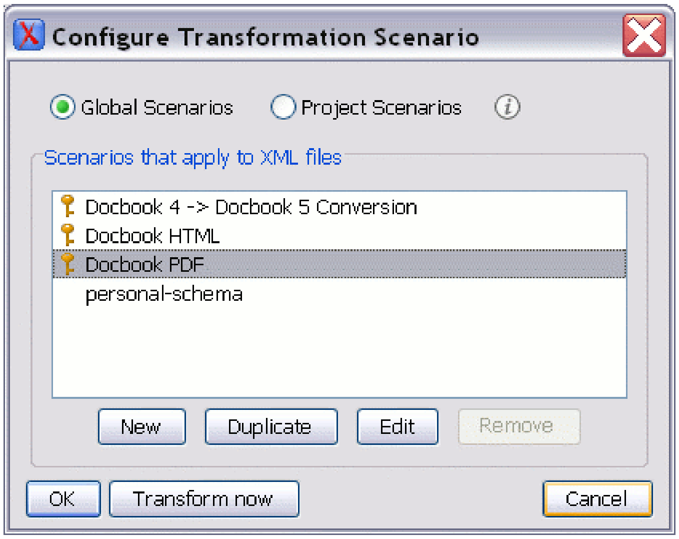
Add new font to FO output

You can do this by changing the stylesheet parameters.

DocBook Stylesheets

Create a transformation scenario that makes use of the `docbook.xsl` file from the `[oXygen-install-dir]/frameworks/docbook/xsl/fo` directory. You must do this in the *Configure Transformation Scenario* dialog.

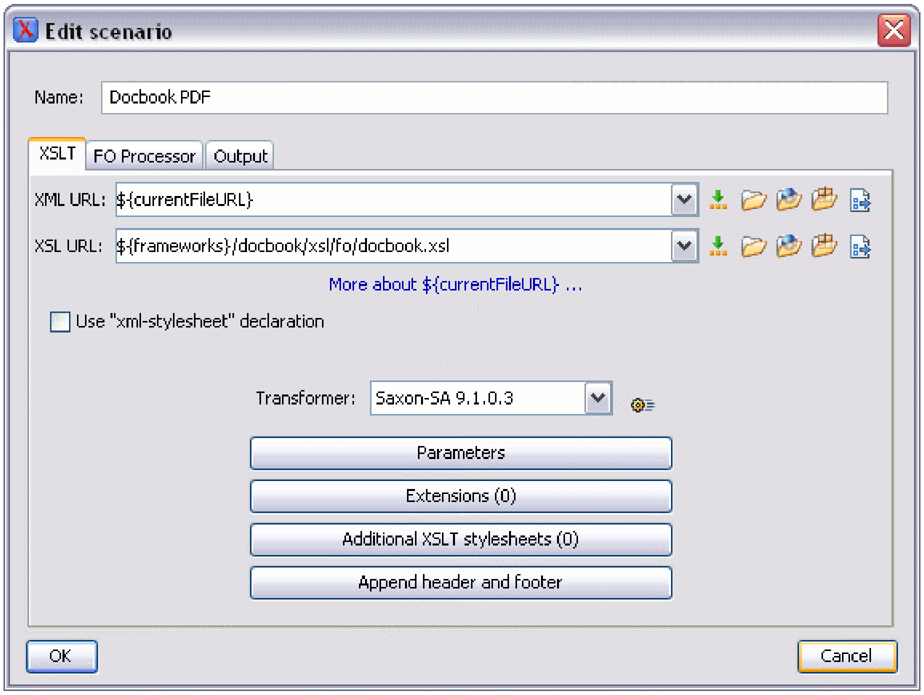
Figure 9.9. The Configure Transformation Scenario dialog



Also you can use the predefined *Docbook PDF* scenario which is based on this Docbook stylesheet. Run a test transformation to make sure the PDF is generated. The Unicode characters are not yet displayed correctly. We have to specify to the stylesheet to generate FO output that uses the font *Arialuni*.

Click on the *Parameters* button in the transformation scenario edit dialog.

Figure 9.10. The Edit Scenario dialog



Edit scenario

Name: Docbook PDF

XML URL: ${currentFileURL}

XSL URL: ${frameworks}/docbook/xsl/fo/docbook.xsl

More about ${currentFileURL} ...

□ Use "xml-stylesheet" declaration

Transformer: Saxon-SA 9.1.0.3

Enter the following parameters indicating the font for the body text and for the titles:

Table 9.1. XSL FO Parameters

Name	Value
body.font.family	Arialuni
title.font.family	Arialuni

TEI Stylesheets

Create a transformation scenario that makes use of the `tei.xsl` file from the `[oXygen-install-dir]/frameworks/tei/xsl/fo` directory. Also you can use the predefined *TEI PDF* scenario which is based on this XSLT stylesheet. Run a test transformation to make sure the PDF is generated. Just like for the Docbook, we have to specify to the stylesheet to generate FO output that uses the font *Arialuni*.

Click on the *Parameters* button of the transformation scenario edit dialog. Enter the following parameters indicating the font for the body text and for other sections:

Table 9.2. XSL FO Parameters

Name	Value
bodyFont	Arialuni
sansFont	Arialuni

Make the transformation again. The characters are now displayed correctly.

DITA-OT Stylesheets

For setting a font to the Apache FOP processor in the transformation of a DITA map with an IDIOM FOP transformation there are two files that must be modified :

- `font-mappings.xml` - available in folder `${frameworks}/dita/DITA-OT/demo/fo/cfg/fo`: the *font-face* element included in each element *physical-font* having the *attribute char-set="default"* must contain the name of the font (*Arialuni* in our example) instead of the default value
- *fop.xconf* - available in folder `${frameworks}/dita/DITA-OT/demo/fo/fop/conf`: an element *font* must be inserted in the element *fonts* which is inside the element *renderer* having the attribute *mime="application/pdf"* as in the above `fopConfiguration.xml` file, for example:

```
<renderer mime="application/pdf">
  . . .
   <fonts>
       <font metrics-url="Arialuni.xml" kerning="yes"
           embed-url="file:/Library/Fonts/Arialuni.ttf">
           <font-triplet name="Arialuni" style="normal"
               weight="normal"/>
       </font>
   </fonts>
  . . .
</renderer>
```

Common transformations

The following examples use the DocBook XSL Stylesheets to illustrate how to configure <oXygen/> for transformation to the various target formats.

Note

<oXygen/> comes with the latest versions of the DocBook and TEI frameworks including special XSLT stylesheets for DocBook and TEI documents. DocBook XSL extensions for the Saxon and Xalan processors are included in the `frameworks/docbook/xsl/extensions` directory.

The following steps are common to all the example procedures below.

1. Set the editor focus to the document to be transformed.
2. Select XML → Configure transformation scenario (**Alt+Shift+T C (Cmd+Alt+T C on Mac OS)**) to open the Configure Transformation dialog.

3. If you want to edit an existing scenario select that scenario in the list and press the *Edit* button. If you want to create a new scenario press the *New* button. If you want to create a new scenario based on an existing scenario select the scenario in the list and press the *Duplicate* button.

4. Select the XSLT tab.

5. Click the "Browse for an input XSL file button". The Open dialog is displayed.

Note

During transformations the Editor Status Bar will show "Transformation - in progress". The transformation is successfully complete when the message "XSL transformation successful" displays. If the transform fails the message "XSL transformation failed" is displayed as an error message in the Messages Panel. The user can stop the transformation process, if the transformer offers such support, by pressing the "Stop transformation" button. In this case the message displayed in the status bar will be "Transformation stopped by user". For the specific case of an XQuery transformation, if you chose an NXD transformer, pressing the "Stop transformation" button will have no effect, as NXD transformers offer no such support.

PDF Output

1. Change directory to **[oxygen]/frameworks/docbook/xsl/fo/**.

2. Select `docbook.xsl`, click Open. The dialog closes.

3. Select the FOP tab.

4. Check the Perform FOP option. The remaining options are enabled.

5. Select the following options:

   a. XSLT result as input.

   b. PDF as method.

   c. Built-in(Apache FOP) as processor.

6. Select the Output tab.

7. In the Save As field enter the output file name relative to the current directory (`YourFileName.pdf`) or the path and output file name (`C:\FileDirectory\YourFileName.pdf`).

8. Optionally, uncheck the XHTML and XML check boxes in the Show As group.

9. Click Transform Now. The transformation is started.

PS Output

1. Change directory to **[oxygen]/frameworks/docbook/xsl/fo/**.

2. Select `docbook.xsl`, click Open. The dialog closes.

3. Select the FOP tab.

4. Check the Perform FOP option. The remaining options are enabled.

5. Select the following options:

   a. XSLT result as input.

   b. PS as method.

   c. Built-in(Apache FOP) as processor.

6. Select the Output tab.

7. In the Save As field enter the output file name relative to the current directory (`YourFileName.ps`) or the path and output file name (`C:\FileDirectory\YourFileName.ps`).

8. Optionally, uncheck the XHTML and XML check boxes in the Show As group.

9. Click Transform Now. The transformation is started.

TXT Output

1. Change directory to **[oxygen]/frameworks/docbook/xsl/fo/**.

2. Select `docbook.xsl`, click Open. The dialog closes.

3. Select the FOP tab.

4. Check the Perform FOP option. The remaining options are enabled.

5. Select the following options:

   a. XSLT result as input.

   b. TXT as method.

   c. Built-in(Apache FOP) as processor.

6. Select the Output tab.

7. In the Save As field enter the output file name relative to the current directory (`YourFileName.txt`) or the path and output file name (`C:\FileDirectory\YourFileName.txt`).

8. Optionally, uncheck the XHTML and XML check boxes in the Show As group.

9. Click Transform Now. The transformation is started.

HTML Output

1. Change directory to **[oxygen]/frameworks/docbook/xsl/html/**.

2. Select `docbook.xsl`, click Open. The dialog closes.

3. Select the FOP tab.

4. Uncheck the Perform FOP option. The FOP options are disabled.

5. Select the Output tab.

6. In the Save As field enter the output file name relative to the current directory (`YourFileName.html`) or the path and output file name (`C:\FileDirectory\YourFileName.html`).

    a. If your pictures are not located relative to the out location, check the XHTML check box in the Show As group.

    b. Specify the path to the folder or URL where the pictures are located

7. Click Transform Now. The transformation is started.

HTML Help Output

1. Change directory to **[oxygen]/frameworks/docbook/xsl/htmlhelp/**.
2. Select `htmlhelp.xsl`, click Open. The dialog closes.
3. Set the XSLT parameter base.dir, it identifies the output directory. (If not specified, the output directory is system dependent.) Also set the manifest.in.base.dir to 1 in order to have the project files copied in output as well.
4. Select the FOP tab.
5. Uncheck the Perform FOP option. The FOP options are disabled.
6. Click Transform Now. The transformation is started.
7. At the end of the transformation you should find the html, hhp and hhc files in the *base.dir* directory.
8. Download Microsoft's HTML Help Workshop and install it.
9. Apply the HTML Help compiler called `hhc.exe` on the html, hhp and hhc files in the `base.dir` directory.

Java Help Output

1. Change directory to **[oxygen]/frameworks/docbook/xsl/javahelp/**.
2. Select `javahelp.xsl`, click Open. The dialog closes.
3. Set the XSLT parameter base.dir, it identifies the output directory. (If not specified, the output directory is system dependent.)
4. Select the FOP tab.
5. Uncheck the Perform FOP option. The FOP options are disabled.
6. Click Transform Now. The transformation is started.

XHTML Output

1. Change directory to **[oxygen]/frameworks/docbook/xsl/xhtml/**.
2. Select `docbook.xsl`, click Open. The dialog closes.
3. Select the FOP tab.
4. Uncheck the Perform FOP option. The FOP options are disabled.

5. Select the Output tab.

6. In the Save As field enter the output file name relative to the current directory (`YourFileName.html`) or the path and output file name (`C:\FileDirectory\YourFileName.html`).

   a. If your pictures are not located relative to the out location, check the XHTML check box in the Show As group.

   b. Specify the path to the folder or URL where the pictures are located

7. Click Transform Now. The transformation is started.

Supported XSLT processors

The <oXygen/> distribution comes with the following XSLT processors:

Xalan 2.7.1

Xalan-Java http://xml.apache.org/xalan-j/ is an XSLT processor for transforming XML documents into HTML, text, or other XML document types. It implements XSL Transformations (XSLT) Version 1.0 and XML Path Language (XPath) Version 1.0.

Saxon 6.5.5

Saxon 6.5.5 [http://saxon.sourceforge.net/saxon6.5.5/] is an XSLT processor, which implements the Version 1.0 XSLT and XPath with a number of powerful extensions. This version of Saxon also includes many of the new features that were first defined in the XSLT 1.1 working draft, but for conformance and portability reasons these are not available if the stylesheet header specifies version="1.0".

Saxon 9.1.0.7 B

Saxon-B http://saxon.sf.net/ implements the "basic" conformance level for XSLT 2.0 and XQuery. The term basic XSLT 2.0 processor is defined in the draft XSLT 2.0 specifications: it is a conformance level that requires support for all features of the language other than those that involve schema processing.

Saxon 9.1.0.7 SA

Saxon 9SA http://www.saxonica.com/ is the schema-aware edition of Saxon 9 and it is one of the built-in processors of <oXygen/>. Saxon SA includes an XML Schema processor, and schema-aware XSLT, XQuery, and XPath processors.

The validation in schema aware transformations is done according to the W3C XML Schema 1.0 specification or according to the W3C XML Schema 1.1 one. This can be configured in Preferences.

Besides the above list <oXygen/> supports the following processors:

Xsltproc (libxslt)

Libxslt http://xmlsoft.org/XSLT/ is the XSLT C library developed for the Gnome project. Libxslt is based on libxml2 the XML C library developed for the Gnome project. It also implements most of the EXSLT set of processor-portable extensions functions and some of Saxon's evaluate and expressions extensions. The libxml2 version included in <oXygen/> is 2.6.32 and the libxslt version is 1.1.23

<oXygen/> uses Libxslt through its command line tool (Xsltproc). The XSLT processor is included into the distribution kit of the stand-alone version for Windows and Mac OS X. Because there are differences between different Linux distributions, on Linux you must install *Libxslt* on your machine as a separate application and set the PATH variable to contain the *Xsltproc* executable.

If you do not have the Libxslt library already installed, you should copy the following files from <oXygen/> stand-alone installation directory to root of the com.oxygenxml.editor_10.3.0 plugin

- Windows: `xsltproc.exe`; `zlib1.dll`,`libxslt.dll`,`libxml2.dll`, `libexslt.dll`,`iconv.dll`
- Linux: `xsltproc`,`libexslt.so.0`, `libxslt.so.1`,`libxsml2.so.2`
- Mac OSX: `xsltproc.mac`, `libexslt`, `libxslt`, `libxml`

The Xsltproc processor can be configured from the XSLTPROC options page.

Note

Known problem: file paths containing spaces are not handled correctly in the LIBXML processor. For example the built-in XML catalog files of the predefined document types (DocBook, TEI, DITA, etc) are not handled by LIBXML if <oXygen/> is installed in the default location on Windows (C:\Program Files) because the built-in XML catalog files are stored in the *frameworks* subdirectory of the installation directory which in this case contains at least a space character.

MSXML 3.0/4.0

MSXML 3.0/4.0 http://msdn.microsoft.com/xml/ is available only on Windows 2000, Windows NT and Windows XP platforms. It can be used for transformationand validation of XSLT stylesheets .

<oXygen/> use the Microsoft XML parser through its command line tool msxsl.exe [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnxml/html/msxsl.asp]

Because msxsl.exe is only a wrapper, Microsoft Core XML Services (MSXML) must be installed on the computer otherwise you get an corresponding warning. You can get the latest Microsoft XML parser from Microsoft web-site http://www.microsoft.com/downloads/details.aspx?FamilyId=3144B72B-B4F2-46DA-B4B6-C5D7485F2B42&displaylang=en [http://www.microsoft.com/downloads/details.aspx?FamilyId=3144B72B-B4F2-46DA-B4B6-C5D7485F2B42&displaylang=en]

MSXML .NET

MSXML .NET http://msdn.microsoft.com/xml/ is available only on Windows NT4, Windows 2000 and Windows XP platforms. It can be used for transformationand validation of XSLT stylesheets .

<oXygen/> performs XSLT transformations and validations using .NET Framework's XSLT implementation (System.Xml.Xsl.XslTransform class) through the nxslt [http://www.tkachenko.com/dotnet/nxslt.html] command line utility. The nxslt version included in <oXygen/> is 1.6.

You should have the .NET Framework version 1.0 already installed on your system otherwise you get this warning: MSXML.NET requires .NET Framework version 1.0 to be installed. Exit code: 128

You can get the .NET Framework version 1.0 from Microsoft web-site http://www.microsoft.com/downloads/details.aspx?FamilyID=d7158dee-a83f-

4e21-b05a-009d06457787&displaylang=en [http://www.microsoft.com/downloads/details.aspx?FamilyID=d7158dee-a83f-4e21-b05a-009d06457787&displaylang=en]

.NET 1.0

A transformer based on the System.Xml 1.0 library available in the .NET 1.0 and .NET 1.1 frameworks from Microsoft (http://msdn.microsoft.com/xml/). It is available only on Windows.

You should have the .NET Framework version 1.0 or 1.1 already installed on your system otherwise you get this warning: MSXML.NET requires .NET Framework version 1.0 to be installed. Exit code: 128

You can get the .NET Framework version 1.0 from Microsoft web-site http://www.microsoft.com/downloads/details.aspx?FamilyID=d7158dee-a83f-4e21-b05a-009d06457787&displaylang=en [http://www.microsoft.com/downloads/details.aspx?FamilyID=d7158dee-a83f-4e21-b05a-009d06457787&displaylang=en]

.NET 2.0

A transformer based on the System.Xml 2.0 library available in the .NET 2.0 framework from Microsoft (http://msdn.microsoft.com/xml/). It is available only on Windows.

You should have the .NET Framework version 2.0 already installed on your system otherwise you get this warning: MSXML.NET requires .NET Framework version 2.0 to be installed. Exit code: 128

You can get the .NET Framework version 2.0 from Microsoft web-site http://www.microsoft.com/downloads/details.aspx?FamilyID=9655156b-356b-4a2c-857c-e62f50ae9a55&DisplayLang=en [http://www.microsoft.com/downloads/details.aspx?FamilyID=9655156b-356b-4a2c-857c-e62f50ae9a55&DisplayLang=en]

Saxon.NET

Saxon.NET http://weblog.saxondotnet.org/ is the port of Saxon 9B XSLT processor to the .NET platform and it is available on a Mozilla Public License 1.0 (MPL) from the Mozilla [http://www.mozilla.org/MPL/MPL-1.0.html] site.

In order to use it you have to unzip in the <oXygen/> install folder the Saxon.NET distribution which you can download from http://saxon.sourceforge.net/ [http://www.saxondotnet.org/saxon.net/downloads/Saxon.NET-1.0-RC1.zip].

You should have the .NET Framework version 1.1 already installed on your system otherwise you get this warning: Saxon.NET requires .NET Framework 1.1 to be installed.

You can get the .NET Framework version 1.1 from Microsoft web-site http://www.microsoft.com/downloads/ThankYou.aspx?familyId=262d25e3-f589-4842-8157-034d1e7cf3a3&displayLang=en [http://www.microsoft.com/downloads/ThankYou.aspx?familyId=262d25e3-f589-4842-8157-034d1e7cf3a3&displayLang=en]

Intel® XML Software Suite

Intel® XML Software Suite http://www.intel.com/software/xml/ is a suite of runtime libraries for Linux and Windows operating systems. It is available on a commercial license from Intel®.

After installing the Intel® XML Software Suite you will have to configure it in the <oXygen/> preferences.

Note

There is no integrated XML Catalog support for MSXML 3.0/4.0 and .NET processors.

Configuring custom XSLT processors

One can configure other XSLT transformation engines than the ones which come with the <oXygen/> distribution. Such an external engine can be used for XSLT transformations within <oXygen/>, in the Editor perspective, and is available in the list of engines in the dialog for editing transformation scenarios.However it cannot be used in the XSLT Debugger perspective.

The output messages of a custom processor are displayed in an output view at the bottom of the <oXygen/> window. If an output message follows the format of an <oXygen/> linked message then a click on the message in the output view highlights the location of the message in an editor panel containing the file referred in the message.

Configuring the XSLT processor extensions paths

The Xalan and Saxon processors support the use of extension elements and extension functions. Unlike a literal result element, which the stylesheet simply transfers to the result tree, an extension element performs an action. The extension is usually used because the xslt stylesheet fails in providing adequate functions to the user for accomplishing a more complex task.

Samples on how to use extensions can be found at:

- for Xalan - http://xml.apache.org/xalan-j/extensions.html
- for Saxon 6.5.5 - http://saxon.sourceforge.net/saxon6.5.5/extensions.html
- for Saxon 9.1.0.7 - http://www.saxonica.com/documentation/extensions/intro.html

In order to set an XSLT processor extension (a directory or a jar file), you have to use the Extensions button of the scenario edit dialog. The old way of setting an extension (using the parameter -Dcom.oxygenxml.additional.classpath) was deprecated and you should use the extension mechanism of the XSLT transformation scenario.

Chapter 10. Querying documents

Running XPath expressions

What is XPath

XPath is a language for addressing specific parts of an XML document. XPath, like the Document Object Model (DOM), models an XML document as a tree of nodes. An XPath expression is a mechanism for navigating through and selecting nodes from the XML document. An XPath expression is in a way analogous to a Structured Query Language (SQL) query used to select records from a database.

XPath models an XML document as a tree of nodes. There are different types of nodes, including element nodes, attribute nodes and text nodes. XPath defines a way to compute a string-value for each type of node.

XPath defines a library of standard functions for working with strings, numbers and Boolean expressions.

Examples:

child: : * Select all children of the root node.

.//name Select all elements having the name "name", descendants of the current node.

/catalog/cd[price>10.80]Selects all the cd elements that have a price element with a value larger than 10.80

To find out more about XPath, the following URL is recommended: http://www.w3.org/TR/xpath

<oXygen/>'s XPath console

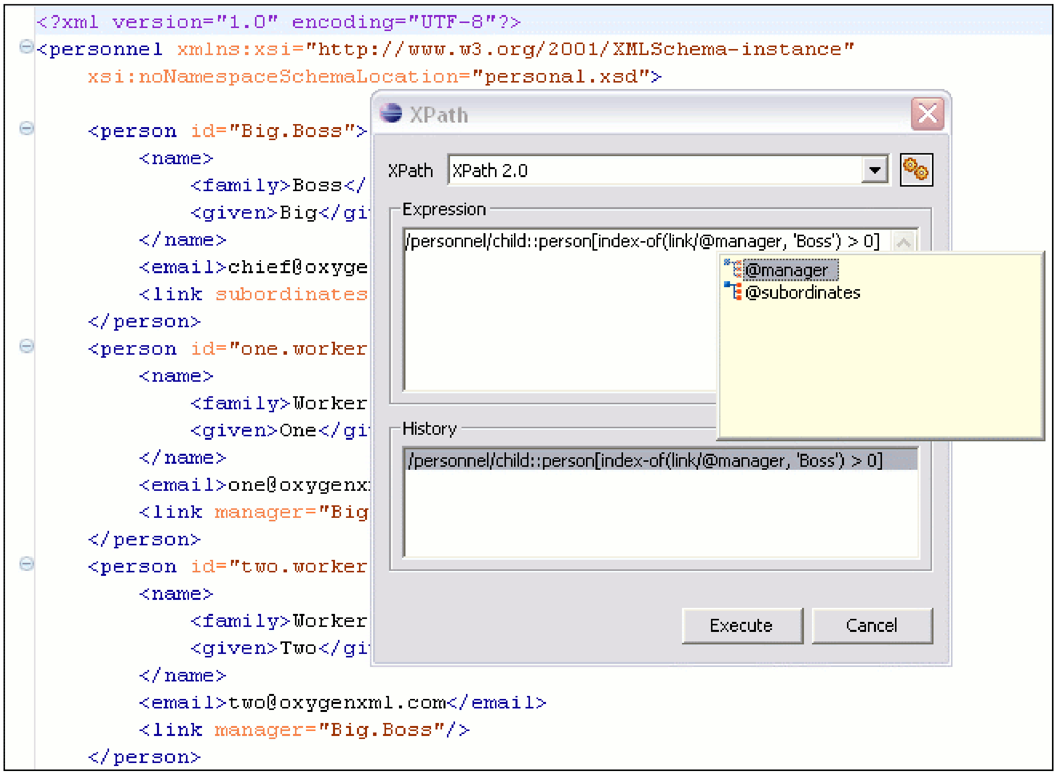
To use XPath effectively requires at least an understanding of the XPath Core Function Library [http://www.w3.org/TR/xpath#corelib]. If you have this knowledge the <oXygen/> XPath expression field part of the current editor toolbar can be used to aid you in XML document development.

In <oXygen/> a XPath 1.0 or XPath 2.0 expression is typed and executed on the current document from the menu XML → XPath (**Ctrl+Shift+X (Cmd+Shift+X on Mac OS)**) or from the toolbar button //* . Both XPath 2.0 basic and XPath 2.0 schema aware expressions can be executed in the XPath console.

The content completion assistant that helps in entering XPath expressions in attributes of XSLT stylesheets elements is also available in the XPath console and offers always proposals dependent of the current context of the cursor inside the edited document. The set of XPath functions proposed by the assistant depends on the XPath version selected from the drop-down menu of the XPath button (1.0 or 2.0).

In the following example the cursor is on a *person* element and the content completion assistant offers all the child elements of the *person* element and all XPath 2.0 functions:

Figure 10.1. Content Completion in the XPath console



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

    <person id="Big.Boss">
        <name>
            <family>Boss</
            <given>Big</gi
        </name>
        <email>chief@oxyge
        <link subordinates
    </person>
    <person id="one.worker
        <name>
            <family>Worker
            <given>One</gi
        </name>
        <email>one@oxygenx
        <link manager="Big
    </person>
    <person id="two.worker
        <name>
            <family>Worker
            <given>Two</gi
        </name>
        <email>two@oxygenxml.com</email>
        <link manager="Big.Boss"/>
    </person>
```

XPath: XPath 2.0

Expression:

```
/personnel/child::person[index-of(link/@manager, 'Boss') > 0]
```

History:

```
/personnel/child::person[index-of(link/@manager, 'Boss') > 0]
```

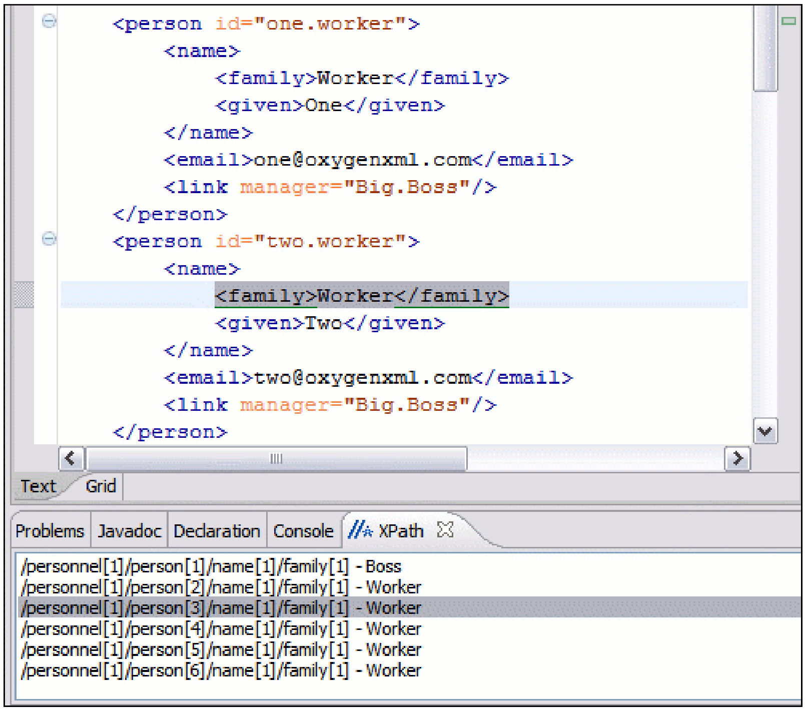
The evaluation of the XPath expression tries to resolve the locations of documents referred in the expression through the XML catalogs which are configured in Preferences and the current XInclude preferences, for example when evaluating the *collection(`URIofCollection`)* function (XPath 2.0). If you need to resolve the references from the files returned by the *collection()* function with an XML catalog set up in the <oXygen/> preferences you have to specify in the query which is the parameter of the *collection()* function the name of the class of the XML catalog enabled parser for parsing these collection files. The class name is `ro.sync.xml.parser.CatalogEnabledXMLReader` and you specify it like this:

```
let $docs := collection(iri-to-uri(
    "file:///D:/temp/test/XQuery-catalog/mydocsdir?recurse=yes;select=*.xml;
    parser=ro.sync.xml.parser.CatalogEnabledXMLReader"))
```

The results of an XPath query are returned in the Message Panel. Clicking a record in the result list highlights the nodes within the text editor panel with a character level precision. Results are returned in a format that is a valid XPath expression:

- [FileName.xml] /node[value]/node[value]/node[value] -

Figure 10.2. XPath results highlighted in editor panel with character precision

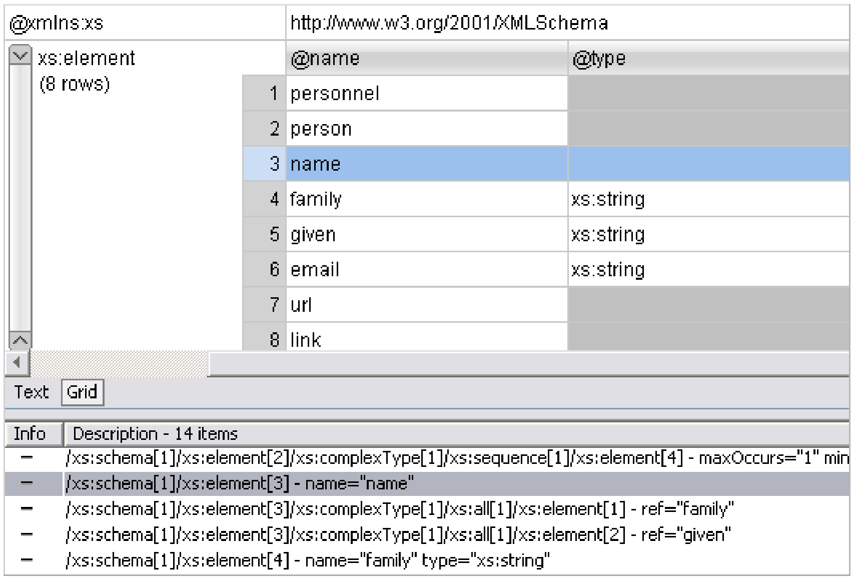


```
<person id="one.worker">
    <name>
        <family>Worker</family>
        <given>One</given>
    </name>
    <email>one@oxygenxml.com</email>
    <link manager="Big.Boss"/>
</person>
<person id="two.worker">
    <name>
        <family>Worker</family>
        <given>Two</given>
    </name>
    <email>two@oxygenxml.com</email>
    <link manager="Big.Boss"/>
</person>
```

```
/personnel[1]/person[1]/name[1]/family[1] - Boss
/personnel[1]/person[2]/name[1]/family[1] - Worker
/personnel[1]/person[3]/name[1]/family[1] - Worker
/personnel[1]/person[4]/name[1]/family[1] - Worker
/personnel[1]/person[5]/name[1]/family[1] - Worker
/personnel[1]/person[6]/name[1]/family[1] - Worker
```

When using the grid editor, clicking a result record will highlight the entire node.

Figure 10.3. XPath results highlighted in the Grid Editor



@xmlns:xs http://www.w3.org/2001/XMLSchema

xs:element (8 rows)

	@name	@type
1	personnel	
2	person	
3	name	
4	family	xs:string
5	given	xs:string
6	email	xs:string
7	url	
8	link	

Info	Description - 14 items
–	/xs:schema[1]/xs:element[2]/xs:complexType[1]/xs:sequence[1]/xs:element[4] - maxOccurs="1" min
–	/xs:schema[1]/xs:element[3] - name="name"
–	/xs:schema[1]/xs:element[3]/xs:complexType[1]/xs:all[1]/xs:element[1] - ref="family"
–	/xs:schema[1]/xs:element[3]/xs:complexType[1]/xs:all[1]/xs:element[2] - ref="given"
–	/xs:schema[1]/xs:element[4] - name="family" type="xs:string"

Note

XPath 2.0 basic queries are executed using Saxon 9 B engine. XPath 2.0 schema aware queries are executed using Saxon 9 SA engine.

The popup menu of the history list of the XPath dialog contains the action *Remove* for removing the selected expression from the history list.

Example 10.1. XPath Utilization with DocBook DTD

Our example is taken from a DocBook book based on the DocBook XML DTD. The book contains a number of chapters. DocBook defines that chapters as have a <chapter> start tag and matching </chapter> end tag to close the element. To return all the chapter nodes of the book enter **//chapter** into the XPath expression field, then **Enter**. This will return all the chapter nodes of the DocBook book, in the Message Panel. If your book has six chapters, their will be six records in the result list. Each record when clicked will locate and highlight the chapter and all sibling nodes contained between the start and end tags of the chapter.

If we used XPath to query for all example nodes contained in the section 2 node of a DocBook XML document we would use the following XPath expression **//chapter/sect1/sect2/example**. If an example node is found in any section 2 node, a result will be returned to the message panel. For each occurrence of the element node a record will be created in the result list.

In our example an XPath query on the file `oxygen.xml` determined that:

```
- [oxygen.xml] /chapter[1]/sect1[3]/sect2[7]/example[1]
```

Which means:

In the file `oxygen.xml`, first chapter, third section level 1, seventh section level 2, the example node found is the first in the section.

Note

If your project is comprised of a main file with ENTITY references to other files, you can use XPath to return all the name elements of a certain type by querying the main file. The result list will query all referenced files.

Important

If the document defines a default namespace then <oXygen/> will bind this namespace to the first free prefix from the list: default, default1, default2, etc. For example if the document defines the default namespace *xmlns="something"* and the prefix *default* is not associated with a namespace then you can match tags without prefix in a XPath expression typed in the XPath console by using the prefix *default*. For example to find all the *level* elements when the root element defines a default namespace you should execute in the XPath console the expression:

```
//default:level
```

To define default mappings between prefixes that can be used in the XPath console and namespace URIs go to the XPath Options user preferences panel and enter the mappings in the *Default prefix-namespace mappings* table. The same preferences panel allows also the configuration of the default namespace used in XPath 2.0 expressions entered into the XPath toolbar and the creation of different results panels for XPath queries executed on different XML documents.

To apply a XPath expression relative to the element on which the caret is positioned use the action XML editor contextual menu → XML Document → Copy XPath (**Ctrl+Shift+.**) (also available on the context menu of the main editor panel) to copy the XPath expression of the element to the clipboard and the Paste action of the contextual menu of the XPath console to paste this expression in the console. Then add your relative expression and execute the resulting complete expression.

The popup menu available on right click in the *Expression* panel of the XPath expressions dialog offers the usual edit actions (Cut, Copy, Paste, Select All)

On Windows the context menu can be displayed with the mouse on a right click or with the keyboard by pressing the special context menu key available on Windows keyboards.

Working with XQuery

What is XQuery

XQuery is the query language for XML and is officially defined by a W3C Recommendation document [http://www.w3.org/TR/xquery/]. The many benefits of XQuery include:

- XQuery allows you to work in one common model no matter what type of data you're working with: relational, XML, or object data.
- XQuery is ideal for queries that must represent results as XML, to query XML stored inside or outside the database, and to span relational and XML sources.
- XQuery allows you to create many different types of XML representations of the same data.
- XQuery allows you to query both relational sources and XML sources, and create one XML result.

Syntax Highlight and Content Completion

To create a new XQuery document select File → New (**Ctrl+N**) and when the New Document dialog appears select XQuery entry.

Once you created the new document <oXygen/> provides syntax highlight for keywords and all known XQuery functions and operators. Also for these there is available a content completion component that can be activated by pressing Ctrl+Space keys. The functions and operators are presented together with a comment about parameters and functionality. For some supported database engines like eXist and Berkeley DB, the content completion lists contain the XQuery functions implemented by that engine if the XQuery file has an associated transformation scenario which use one of the specified engine or the XQuery file has no associated scenario but the validation is make with one of these engines (a validation engine is specified in *XML / XSLT - FO / XQuery* Preferences page). This helps you to insert in your queries only calls to the functions implemented by the target database engine.

The extension functions built in the Saxon product are available on Content Completion if one of the following conditions are true:

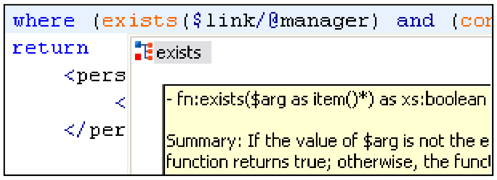
- if the edited file has a transformation scenario associated that use as transformation engine Saxon 9.1.0.7 B or Saxon 9.1.0.7 SA
- if the edited file has a validation scenario associated that use as validation engine Saxon 9.1.0.7 B or Saxon 9.1.0.7 SA
- if the validation engine specified in Options is Saxon 9.1.0.7 B or Saxon 9.1.0.7 SA.

If the Saxon namespace (http://saxon.sf.net [http://saxon.sf.net/]) is mapped to a prefix this prefix is used when the functions are presented, otherwise the default prefix for the saxon namespace (*saxon*) is used.

If you want to use a function from a namespace mapped to a prefix, just type that prefix and the Content Completion will display all the XQuery functions from that namespace. The XQuery functions from default namespace offered by content completion are prefixed if the default namespace is mapped to a prefix, otherwise is displayed just the name of this.

The content completion popup window presents all the variables and functions from both the edited XQuery file and its imports.

Figure 10.4. XQuery Content Completion



```
where (exists($link/@manager) and (cor
return
       <pers
           <
       </per
```

exists

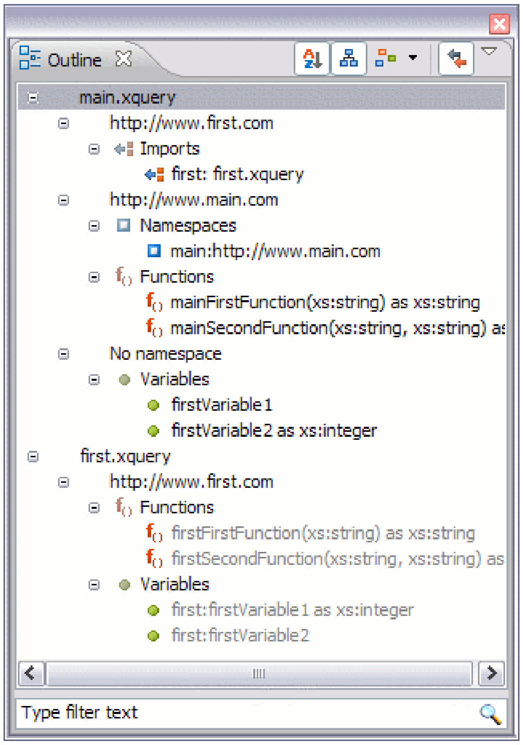
- fn:exists($arg as item()*) as xs:boolean

Summary: If the value of $arg is not the e
function returns true; otherwise, the funcl

XQuery Outline View

The XQuery document structure is presented in the *XQuery Outline* view. The outliner presents the list of all the components (namespaces, imports, variables and functions) from both the edited XQuery file and its imports. It allows a quick access to a component by knowing its name. It is opened from Window → Show View → Other → oXygen → Outline.

Figure 10.5. XQuery Outline View



```
main.xquery
    http://www.first.com
        Imports
            first: first.xquery
    http://www.main.com
        Namespaces
            main:http://www.main.com
        Functions
            mainFirstFunction(xs:string) as xs:string
            mainSecondFunction(xs:string, xs:string) as
    No namespace
        Variables
            firstVariable1
            firstVariable2 as xs:integer
first.xquery
    http://www.first.com
        Functions
            firstFirstFunction(xs:string) as xs:string
            firstSecondFunction(xs:string, xs:string) as
        Variables
            first:firstVariable1 as xs:integer
            first:firstVariable2
```

To easily navigate in the document, the XQuery Outline View provide four options:

Sort Allows you to sort alphabetically the xquery components.

Show imported/included Show also the imported/included components.

Grouping Options Allows you to group the components by location, namespace and type. When grouping by namespace, the main XQuery module namespace is the first presented in the outline view.

Selection update on caret move Allows a synchronization between Outline View and source document. The selection in the outline view can be synchronized with the caret's moves or the changes in the XQuery editor. Selecting one of the components from the outline view also selects the corresponding item in the source document.

If you know the component name, you can search it in the outline view by typing his name in the filter text field from the bottom of the view or directly on the tree structure. When you type the component name in the filter text field you can switch to the tree structure using the arrow keys of the keyboard, Enter, Tab, Shift-Tab. To switch from tree structure to the filter text field you can use Tab, .

Tip

The search filter is case insensitive. The following wildcards are accepted:

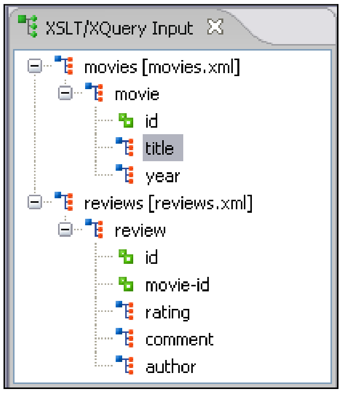
- ***** - any string
- **?** - any character
- **,** -patterns separator

If no wildcards are specified, the string to search will be searched as a partial match (similar to ***textToFind***).

The Query Input View

A node can be dragged and dropped in the editor area for quickly inserting *doc()* or other XQuery expressions.

Figure 10.6. XQuery input view



For example for the following XML documents

```
<movies>
<movie id="1">
<title>The Green Mile</title>
<year>1999</year>
</movie>
<movie id="2">
<title>Taxi Driver</title>
<year>1976</year>
```

```
</movie>
</movies>
```

and

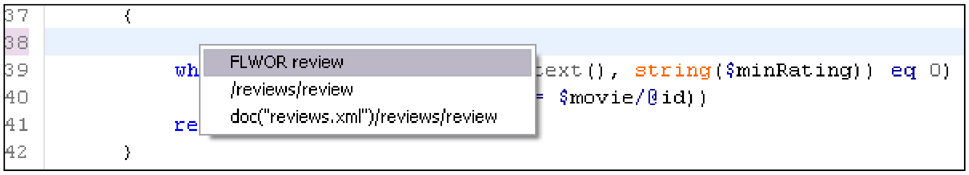
```
<reviews>
<review id="100" movie-id="1">
<rating>5</rating>
<comment>It is made after a great Stephen King book.
</comment>
<author>Paul</author>
</review>
<review id="101" movie-id="1">
<rating>3</rating>
<comment>Tom Hanks does a really nice acting.</comment>
<author>Beatrice</author>
</review>
<review id="104" movie-id="2">
<rating>4</rating>
<comment>Robert De Niro is my favorite actor.</comment>
<author>Maria</author>
</review>
</reviews>
```

and the following XQuery

```
let $review := doc("reviews.xml")
for $movie in doc("movies.xml")/movies/movie
let $movie-id := $movie/@id
return
<movie id="{$movie/@id}">
{$movie/title}
{$movie/year}
<maxRating>
{

}
</maxRating>
</movie>
```

if you drag the *rating* element and drop between the braces a popup menu will be displayed.

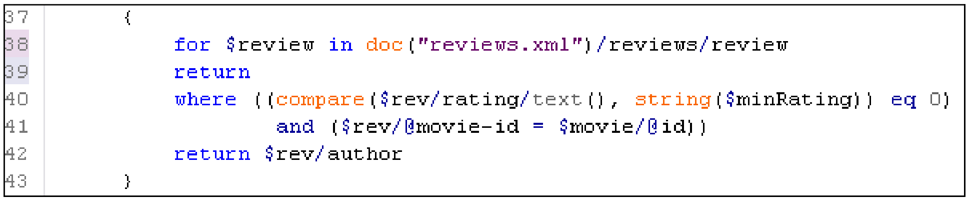
Figure 10.7. XQuery Input drag and drop popup menu



```
        {

                wh                                    ext(), string($minRating)) eq 0)
                                                      = $movie/@id))
                re
        }
```

Select for example *FLWOR rating* and the result document will be:

Figure 10.8. XQuery Input drag and drop result

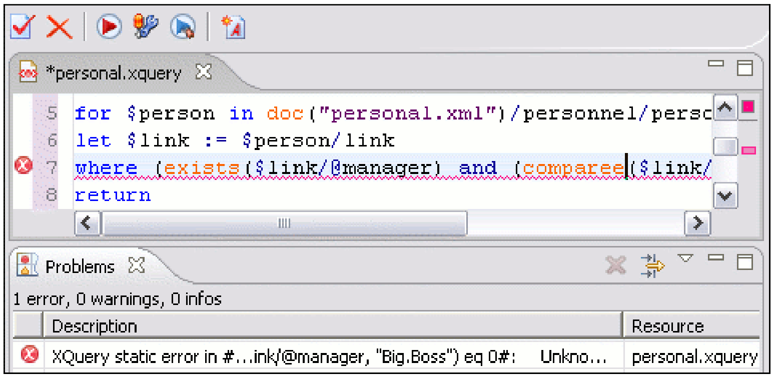


```
        {
            for $review in doc("reviews.xml")/reviews/review
            return
            where ((compare($rev/rating/text(), string($minRating)) eq 0)
                    and ($rev/@movie-id = $movie/@id))
            return $rev/author
        }
```

XQuery Validation

With <oXygen/> you can validate your documents before using them in your transformation scenarios. The validation uses the Saxon 9.1.0.7 B processor or the 9.1.0.7 SA, IBM DB2, eXist, Software AG Tamino, Berkeley DB XML or Documentum xDb (X-Hive/DB) if you installed them. This is in conformance with the XQuery Working Draft http://www.w3.org/TR/xquery/. The processor is used in two cases: validation of the expression and execution. Although the execution implies a validation, it is faster to syntactically check the expression without executing it. The errors that occurred in the document are presented in the messages view at the bottom of editor window, with a full description message. As with all error messages, if you click on one entry, the line where the error appeared is highlighted.

Figure 10.9. XQuery Validation



```
for $person in doc("personal.xml")/personnel/persc
let $link := $person/link
where (exists($link/@manager) and (comparee($link/
return
```

1 error, 0 warnings, 0 infos

	Description	Resource
⊗	XQuery static error in #...ink/@manager, "Big.Boss") eq 0#: Unkno...	personal.xquery

Please note that if you choose a processor that doesn't support XQuery validation you will receive a warning when trying to validate.

Note

If there is no transformation scenario associated with the current document, the validation will be performed using the processor or connection specified in the *XML / XSLT - FO / XQuery* Preferences page. Otherwise, the XQuery document will be validated using the Transformer from the associated scenario.

Other XQuery editing actions

The XQuery editor type offers a reduced version of the popup menu available in the XML editor type, that means only the folding actions,the edit actions a part of the source actions (only the actions *To lower case*, *To upper case*, *Capitalize lines*) and the open actions: *Open file at Caret*, *Open file at Caret in System Application*.

Transforming XML Documents Using XQuery

XQueries are very similar to the XSL stylesheets in the sense they both are capable of transforming an XML input into another format. You can define transformation scenarios that specify the input URL, the preview mode, XML or XHTML. The result can be saved and opened in the associated application. You can even run a FO processor on the output of an XQuery. The transformation scenarios may be shared between many XQuery files, are exported at the same time with the XSLT scenarios and can be managed in the dialog *Configure Transformation Scenario* or in the *Scenarios* view. The transformation performed can be based on the XML document specified in the Input field, or, if this field is empty, the documents referred from the query expression are used instead. The parameters of XQuery transforms must be set in the *Parameters* dialog. Parameters that are in a namespace must be specified using the qualified name, for example a *param* parameter in the *http://www.oxygenxml.com/ns* namespace must be set with the name *{http://www.oxygenxml.com/ns}param*.

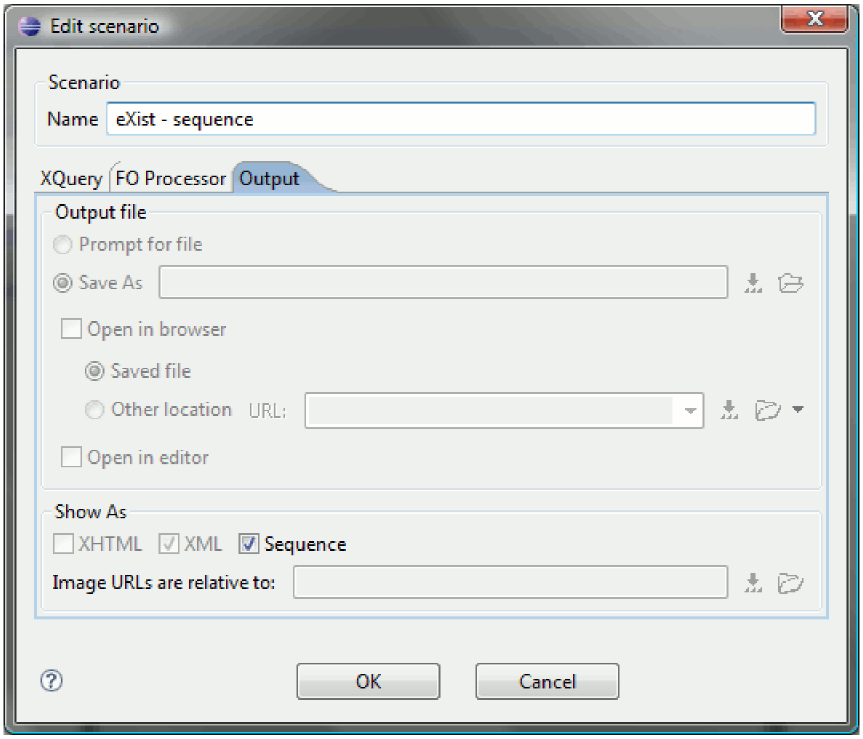
The transformation uses the processor Saxon 9.1.0.7 B, Saxon 9.1.0.7 SA or a database connection (details can be found in the Working with Databases chapter - in the XQuery transformation section).

Display result in Sequence view

The result of an XQuery executed on a database can be very large and sometimes only a part of the full result is needed. For avoiding the long time necessary for fetching the full result the `<Sequence>` option of the XQuery transformation scenario should be used. This option fetches only the first chunk of the result and the user decides if he wants to fetch the next chunk after looking at the first chunk in the `<Sequence>` result view. The size of a chunk can be set with a user option.

The *Sequence* option of the XQuery scenario must be selected in the *Output* tab of the scenario dialog.

Figure 10.10. The "Edit scenario" dialog / "Output" tab



Edit scenario

Scenario

Name: eXist - sequence

Output file

○ Prompt for file

● Save As

□ Open in browser

● Saved file

○ Other location URL:

□ Open in editor

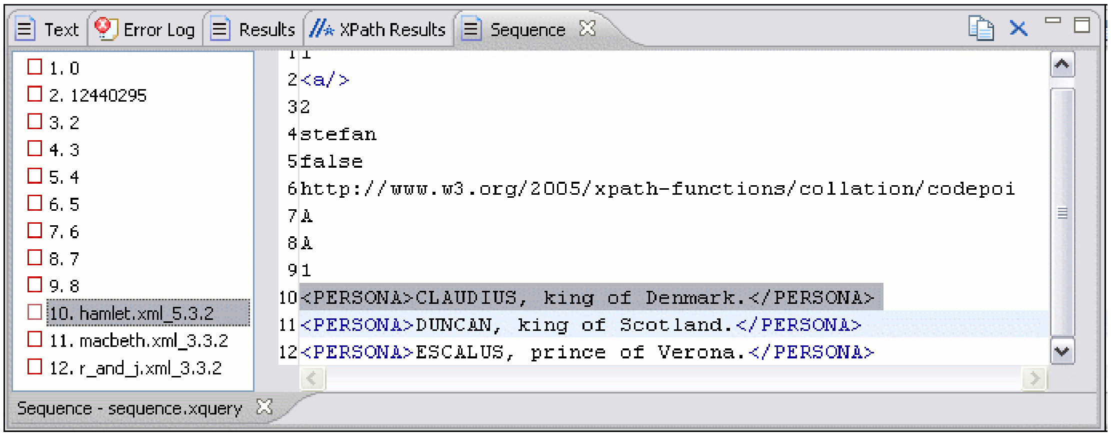
Show As

□ XHTML ☑ XML ☑ Sequence

Image URLs are relative to:

A chunk of the XQuery transformation result is displayed in the `<Sequence>` view.

Figure 10.11. The XQuery transformation result displayed in "Sequence" view



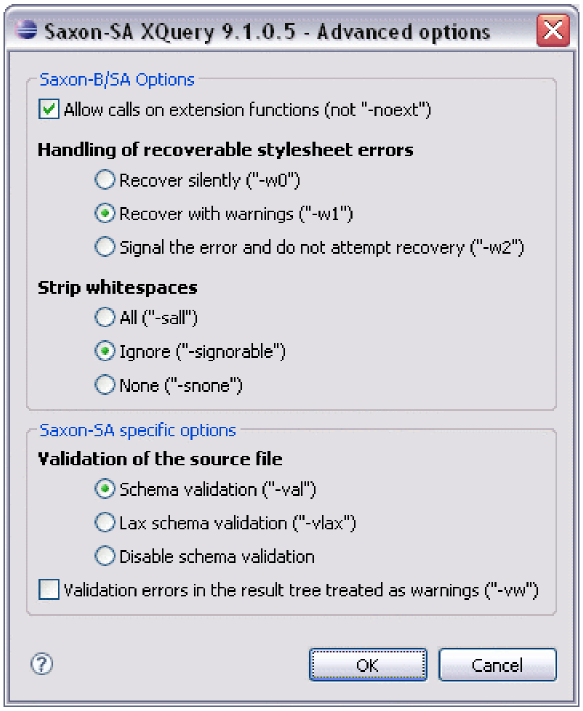
```
□ 1. 0
□ 2. 12440295
□ 3. 2
□ 4. 3
□ 5. 4
□ 6. 5
□ 7. 6
□ 8. 7
□ 9. 8
□ 10. hamlet.xml_5.3.2
□ 11. macbeth.xml_3.3.2
□ 12. r_and_j.xml_3.3.2
```

```
 1 1
 2 <a/>
 3 2
 4 stefan
 5 false
 6 http://www.w3.org/2005/xpath-functions/collation/codepoi
 7 A
 8 A
 9 1
10 <PERSONA>CLAUDIUS, king of Denmark.</PERSONA>
11 <PERSONA>DUNCAN, king of Scotland.</PERSONA>
12 <PERSONA>ESCALUS, prince of Verona.</PERSONA>
```

Advanced Saxon B/SA transform options

The XQuery transformation scenario allows configuring advanced options specific for the Saxon B/SA engine. They are the same options as the ones set in the user preferences but they are configured as a specific set of transformation options for each transformation scenario. The default values of the options in the transformation scenario are the values set in the user preferences.

Figure 10.12. The advanced options of Saxon SA/Saxon B



Saxon-SA XQuery 9.1.0.5 - Advanced options

Saxon-B/SA Options

☑ Allow calls on extension functions (not "-noext")

Handling of recoverable stylesheet errors

○ Recover silently ("-w0")
● Recover with warnings ("-w1")
○ Signal the error and do not attempt recovery ("-w2")

Strip whitespaces

○ All ("-sall")
● Ignore ("-signorable")
○ None ("-snone")

Saxon-SA specific options

Validation of the source file

● Schema validation ("-val")
○ Lax schema validation ("-vlax")
○ Disable schema validation

□ Validation errors in the result tree treated as warnings ("-vw")

The advanced options specific for Saxon SA/Saxon B are:

Allow calls on extension functions   If checked call external Java functions is allowed.

Handling of recoverable stylesheet errors

Policy for handling recoverable errors in the stylesheet: Allows the user to choose how dynamic errors will be handled. Either one of the following options can be selected: recover silently, recover with warnings or signal the error and do not attempt recovery.

Strip whitespaces

Strip whitespaces feature can be one of the three options: All, Ignorable, None.

All

strips all whitespace text nodes from source documents before any further processing, regardless of any xsl:strip-space declarations in the stylesheet, or any xml:space attributes in the source document.

Ignorable

strips all ignorable whitespace text nodes from source documents before any further processing, regardless of any xsl:strip-space declarations in the stylesheet, or any xml:space attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.

None

strips no whitespace before further processing. (However, whitespace will still be stripped if this is specified in the stylesheet using xsl:strip-space).

Validation of the source file

Available only for Saxon SA.

Schema validation

This mode requires an XML Schema and determines whether source documents should be parsed with schema-validation enabled.

Lax schema validation

This mode determines whether source documents should be parsed with schema-validation enabled if an XML Schema is provided.

Disable schema validation

This determines whether source documents should be parsed with schema-validation disabled.

Validation errors in the results tree treated as warnings

Available only for Saxon SA. If checked, all validation errors are treated as warnings, otherwise they are treated as fatal.

Updating XML documents using XQuery

Using the bundled Saxon 9.1.0.7 SA XSLT processor <oXygen/> now offers support for XQuery Update 1.0. The XQuery Update Facility provides expressions that can be used to make persistent changes to instances of the XQuery 1.0 and XPath 2.0 Data Model. Thus, besides querying XML documents, you can modify them using the various insert/delete/modify/create methods available in the XQuery Update 1.0 [http://www.w3.org/TR/xquery-update-10/#introduction] standard.

Just choose Saxon 9.1.0.7 SA as a transformer in the scenario associated with XQuery files containing update statements and <oXygen/> will notify you if the update was successful.

Example 10.2. Using XQuery Update to modify a tag name in an XML file

```
rename node doc("books.xml")//publisher[1]//book[1] as "firstBook"
```

Chapter 11. Debugging XSLT stylesheets and XQuery documents

Overview

The Debugger perspectives enables you to test and debug XSLT 1.0 /2.0 stylesheets and XQuery 1.0 documents. The interface presents simultaneous views of the source XML document, the XSLT/XQuery document and the result document. As you go step by step through the XSLT/XQuery document the corresponding output is generated step by step, and the corresponding position in the XML file is highlighted for each step. At the same time, special views in the interface provide various types of debugging information and events useful for understanding the transformation process.

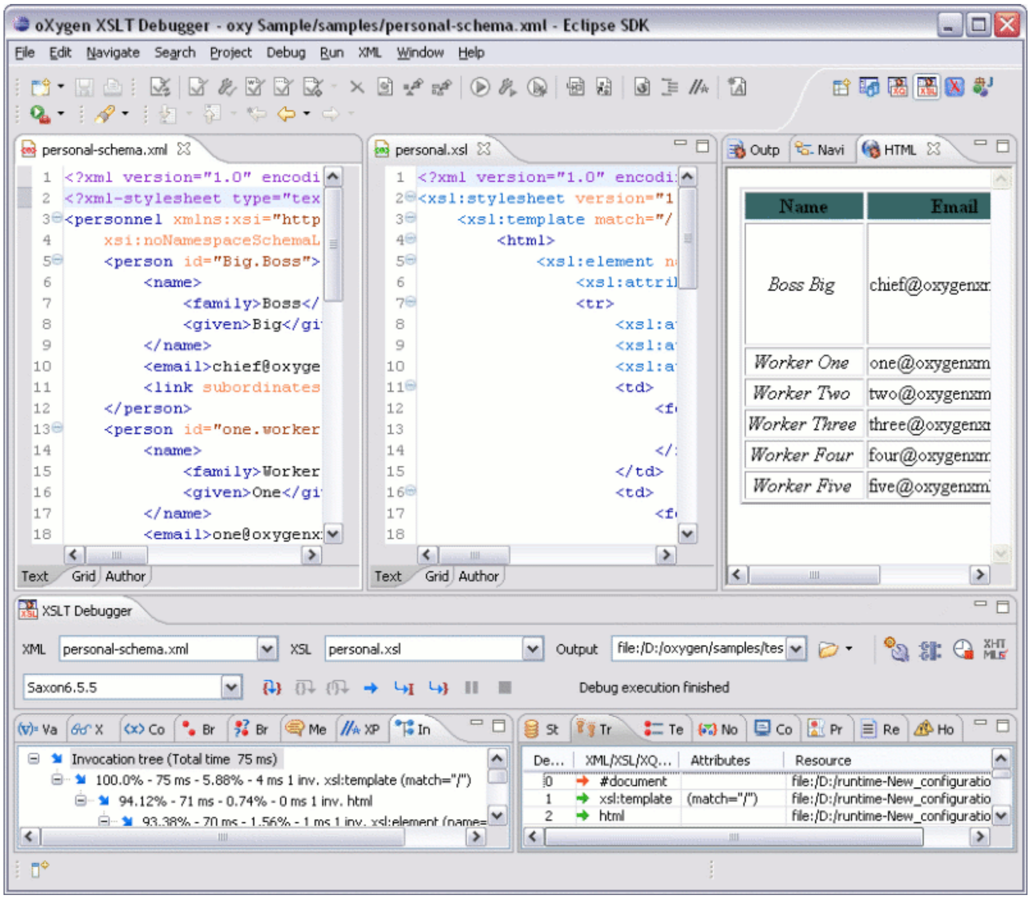
The user benefits of a rich set of features for testing and solving XSLT/XQuery problems:

- Support for XSLT 1.0 stylesheets (through the Saxon 6.5.5 and Xalan XSLT engines) , XSLT 2.0 stylesheets (through the Saxon 9.1.0.7 B XSLT engine and the Saxon 9.1.0.7 SA one) and XQuery 1.0 (through the Saxon 9.1.0.7 B XQuery engine and the Saxon 9.1.0.7 SA one).
- Stepping capabilities: step in, step over, step out, run, run to cursor, run to end, pause, stop.
- Back mapping between every piece of output and instruction element /source context who generate it .
- Breakpoints on both source and XSLT/XQuery documents.
- Call stack view on both source and XSLT/XQuery documents.
- Trace history on both source and XSLT/XQuery documents.
- Support for XPath expression evaluation during debugging.
- Step into imported/included stylesheets as well as included source entities.
- Available templates and hits count.
- Variables view.
- Dynamic output generation.

Layout

The Debugger perspective interface looks like below. This interface is comprised of 4 panes as follows:

Figure 11.1. Debugger Mode Interface



oXygen XSLT Debugger - oxy Sample/samples/personal-schema.xml - Eclipse SDK
File Edit Navigate Search Project Debug Run XML Window Help
personal-schema.xml
personal.xsl
Outp
Navi
HTML
Text Grid Author
XSLT Debugger
Va X Co Br Br Me XP In
St Tr Te No Co Pr Re Ho

```
<?xml version="1.0" encodi
<?xml-stylesheet type="tex
<personnel xmlns:xsi="http
    xsi:noNamespaceSchemaL
    <person id="Big.Boss">
        <name>
            <family>Boss</
            <given>Big</gi
        </name>
        <email>chief@oxyge
        <link subordinates
    </person>
    <person id="one.worker
        <name>
            <family>Worker
            <given>One</gi
        </name>
        <email>one@oxygenx
```

```
<?xml version="1.0" encodi
<xsl:stylesheet version="1
    <xsl:template match="/
        <html>
            <xsl:element n
                <xsl:attri
                <tr>
                    <xsl:a
                    <xsl:a
                    <xsl:a
                    <td>
                        <f

                        </
                    </td>
                    <td>
                        <f
```

Name	Email
Boss Big	chief@oxygenxm
Worker One	one@oxygenxm
Worker Two	two@oxygenxm
Worker Three	three@oxygenx
Worker Four	four@oxygenxm
Worker Five	five@oxygenxm

XML: personal-schema.xml XSL: personal.xsl Output: file:/D:/oxygen/samples/tes

Saxon6.5.5 Debug execution finished

```
Invocation tree (Total time  75 ms)
  100.0% - 75 ms - 5.88% - 4 ms 1 inv. xsl:template (match="/")
    94.12% - 71 ms - 0.74% - 0 ms 1 inv. html
      93.38% - 70 ms - 1.56% - 1 ms 1 inv. xsl:element (name=
```

De...	XML/XSL/XQ...	Attributes	Resource
0	#document		file:/D:/runtime-New_configuratio
1	xsl:template	(match="/")	file:/D:/runtime-New_configuratio
2	html		file:/D:/runtime-New_configuratio

Source document view (XML)
Displays and allows editing of data or document oriented XML files (documents).

XSL/XQuery document view (XSL/XQuery)
Displays and allows editing of XSL files(stylesheets) or XQuery documents.

Output document view
Displays the transformed output that results from the input of a selected document (XML) and selected stylesheet (XSL) or XQuery document to the transformer. The result of transformation is dynamically written as the transformation is processed.

Control view
The control view provides functionality for configuration and control of debugging operations. It also provides a series of Information views types. This pane is comprised of two parts:

- Control Toolbar
- Information views

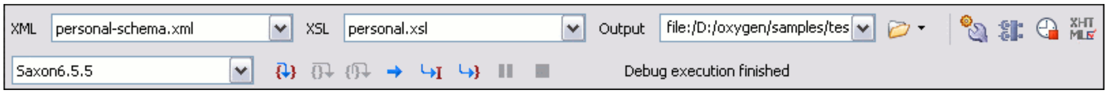
XML documents and XSL stylesheets or XQuery documents that were opened in Editor perspective are automatically sorted into the first two panes. When multiple files of each type are opened, the individual documents and stylesheets are separated using the familiar tab management system of the Editor perspective. Selecting a tab brings the document or stylesheet into focus and enables editing without toggling back to the Editor perspective.

During debugging the current execution node is highlighted on both document (XML) and XSL/XQuery views.

Control Toolbar

The toolbar contains all actions needed in order to configure and control the debug process. Items are described below from left to right as they appear in the toolbar.

Figure 11.2. Control Toolbar



XML personal-schema.xml XSL personal.xsl Output file:/D:/oxygen/samples/tes

Saxon6.5.5 Debug execution finished

XML source selector

The selection represents the source document to be used as input by the transformation engine. The selection list is filled-in with all opened files (the XML ones being emphasized). This gives you the possibility to use other file types as source. In case of XQuery debugging session this selection field can be set to default value NONE, as usually XQuery documents do not require an input source.

XSL/XQuery selector

The selection represents the stylesheet or XQuery document to be used by the transformation engine. The selection list is filled-in with all opened files (the XSL/XQuery ones being emphasized).

Output selector

The selection represents the output file specified in the associated transformation scenario.

XSLT/XQuery engine selector

Lists the available XSLT/XQuery processors

(Saxon and Xalan Java - see specifications for XSLT or Saxon9B for XQuery.)

XSLT/XQuery parameters

XSLT/XQuery parameters to be used by the transformation.

Edit extensions

Add and remove the Java classes and jars used as XSLT extensions.

Enable profiling

Enable/Disable current transformation profiling.

Enable XHTML output

Enable or disable rendering of output to the XHTML Output document View during the transformation process. For performance issues, it is advisable to disable XHTML output for large jobs. Also, the XHTML area is only able to render XHTML documents. In order to view the output result of other formats, such as HTML, save the Text output area to a file and use the required external browser for viewing.

When starting a debug session from the editor perspective using the Debug Scenario action, the state of this toolbar button reflects the state of the "Show as XHTML" output option from the scenario.

Step into (F7)

Starts the debugging process and runs until the next stylesheet node (next step in transformation).

Step over (F8 (Alt+F7 on Mac OS))

Executes the current stylesheet node (including its sub-elements) and goes to next node in document order (usually the next sibling of the current node).

Step out (Shift + F7)

Steps out to the parent node (equivalent to the Step over (F8 (Alt+F7 on Mac OS)) on the parent).

Run

Starts the debugging process and runs until the first breakpoint is encountered or until the end of transformation occurs, if no breakpoints are encountered (see the section called “Breakpoints View”).

Run to cursor (Ctrl + F5)

Starts the debugging process and runs until one of the following conditions occur: the line of cursor is reached, a valid breakpoint is reached or end of execution.

Run to end (Alt + F5)

Runs the transformation until the end, without taking into account any enabled breakpoints that might be set.

Pause (Shift + F6)

Interrupts the current transformation. This is useful for long transformations (DocBook for instance) when you want to find out what point the transformation has reached. The transformation can be resumed after.

Stop (F6)

Ends the transformation process.

Information views

The information view is comprised of two panes that are used to display various types of information used to understand the transformation process. For each information type there is a corresponding tab. While running a transformation, relevant events are displayed in the various information views. This enables the developer to obtain a clear view of the transformation progress. Using the Debug controls developers can easily isolate parts of stylesheet therefore they may be understood and modified. The information types include (for a more detailed discussion on each information type see Viewing processing information):

Left side information views

- Context Node View
- XWatch View
- Breakpoints View
- Break Conditions View
- Messages View (XSLT only)
- Variables View

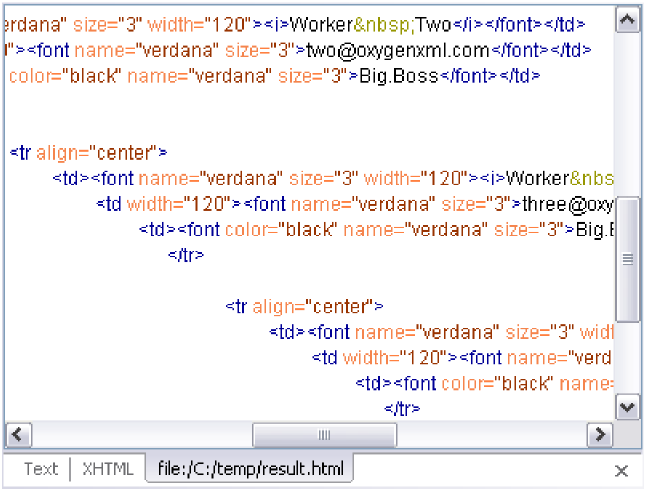
Right side information views

- Stack View
- Trace View
- Templates View (XSLT only)
- Nodeset View

Multiple output documents in XSLT 2.0

For XSLT 2.0 stylesheets that store the output in more than one file by using the *xsl:result-document* instruction the content of the file created in this way is displayed dynamically while the transformation is running in an output view. There is one view for each *xsl:result-document* instruction so that the output of different instructions is not mixed but is presented in different views.

Figure 11.3. Multiple output documents in XSLT 2.0



```
erdana" size="3" width="120"><i>Worker&nbsp;Two</i></font></td>
l"><font name="verdana" size="3">two@oxygenxml.com</font></td>
 color="black" name="verdana" size="3">Big.Boss</font></td>



<tr align="center">
      <td><font name="verdana" size="3" width="120"><i>Worker&nbs
            <td width="120"><font name="verdana" size="3">three@oxy
                  <td><font color="black" name="verdana" size="3">Big.E
                        </tr>


                              <tr align="center">
                                    <td><font name="verdana" size="3" widt
                                          <td width="120"><font name="verd
                                                <td><font color="black" name=
                                                      </tr>
```

Working with the XSLT/XQuery Debugger

The following topics are present about how to follow XSLT/XQuery processing and detect errors in your stylesheets or XQuery documents:

- Steps in a typical debug process
- Using breakpoints
- Viewing processing information
- Determining what XSL/XQuery expression generated particular output

Steps in a typical debug process

To debug a stylesheet or XQuery document follow the procedure:

1. Open the source XML document and the XSLT/XQuery document.
2. If you are in the <oXygen/> XML perspective switch to the <oXygen/> XSLT Debugger perspective in case of XSLT debugging or to the <oXygen/> XQuery Debugger in case of XQuery debugging with one of the actions (here explained for XSLT):

   - Window → Open Perspective → Other ...+<oXygen/> XSLT Debugger
   - The toolbar button Debug scenario . This action initializes the Debugger perspective with the parameters of the transformation scenario. Any modification applied to the scenario parameters (the transformer engine, the

XSLT parameters, the transformer extensions, etc) will be saved back in the scenario when exiting from the Debugger perspective.

3. Select the source XML document in the XML source selector of the Control toolbar In case of XQuery debugging if your XQuery document has no implicit source set the source selector value to NONE.

4. Select the XSL/XQuery document in the XSL/XQuery selector of the Control toolbar.

5. Set XSLT/XQuery parameters from the button available on the Control toolbar.

6. Set one or more breakpoints.

7. Step through the stylesheet using the buttons available on the Control toolbar: Step into, Step over, Step out, Run, Run to cursor, Run to end, Pause, Stop

8. Examine the information in the Information Views to find the bug in the transformation process.

Note

Initially only the two available Saxon XSLT/XQuery Processors are active in the Debugger perspective. If you select Xalan XSLT Processor an warning message is shown requiring Xalan version 2.7.1. To set Xalan 2.7.1 you need to copy `xalan.jar` and `serializerOxygen.jar` from `[oxygen]/lib` and put it to the endorsed folder from your JRE/JDK used for running Eclipse (you can find it in Help → About Eclipse Platform+Configuration Details`java.endorsed.dirs` entry) and restart Eclipse.

Using breakpoints

The <oXygen/> XSLT/XQuery Debugger allows you to interrupt XSLT/XQuery processing to gather information about variables and processor execution at particular points.

Inserting breakpoints

To insert a breakpoint:

1. In the XML source document or the XSLT/XQuery document that you want to set a breakpoint, place your cursor on the line where you want the breakpoint to be. You can set breakpoints on XML source only for XSLT debugging sessions.

2. Click with the mouse the left side stripe of the editor window on the line where you want the breakpoint to be.

Note

If the start tag of the element you want to set a breakpoint is spanning on multiple rows, then you have to place the breakpoint on the line containing the end of the start tag. In the following example if you try to place a breakpoint on the call-template line, the editor will show an error dialog, explaining that you must place the breakpoint at the end of the start tag. This means you have to place the breakpoint on the line containing the text: ">" , just after the "name" attribute.

```
<xsl:template match="chapter">
    <xsl:call-template
            name="title"
     >
```

```
          </xsl:call-template>
     </xsl:template>
```

Removing breakpoints

To remove a breakpoint:

- Click with the mouse the left side stripe of the editor window on the line with the breakpoint or select Edit → Breakpoints → Remove all

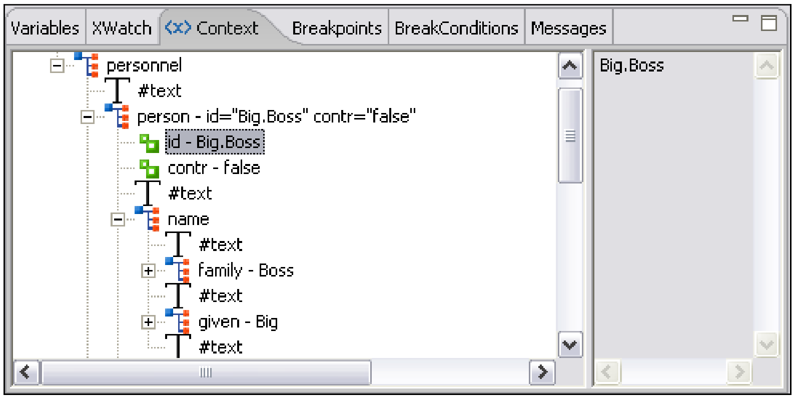
Viewing processing information

Detailed information about the debugger status are provided using the information views.

Context node view

The context node is valid only for XSLT debugging session and is a source node corresponding to the XSL expression being evaluated. It is also called the context of execution. The context node implicitly changes as the processor hits various steps (at the point where XPath expressions are evaluated). This node has the same value as evaluating '.' (dot) XPath expression on XWatch View. The value of the context node is presented as a tree in the view.

Figure 11.4. The Context node view

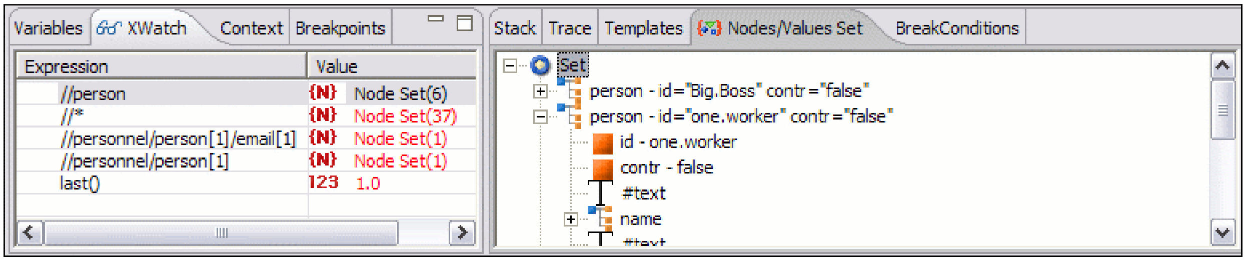


The context node is presented in a tree-like fashion. Nodes from a defined namespace bound to a prefix are displayed using the qualified name. If the namespace is not bound to a prefix the namespace URI will be presented before the node name. The value of the selected attribute or node is shown in the right side panel.

XPath watch view

Shows XPath expressions to be evaluated during debugging. Expressions are evaluated dynamically as the processor changes its source context.

Figure 11.5. The XPath watch view



Expression	Value
//person	{N} Node Set(6)
//*	{N} Node Set(37)
//personnel/person[1]/email[1]	{N} Node Set(1)
//personnel/person[1]	{N} Node Set(1)
last()	123 1.0

```
Set
  person - id="Big.Boss" contr="false"
  person - id="one.worker" contr="false"
    id - one.worker
    contr - false
    #text
    name
    #text
```

Table 11.1. XWatch details

Column	Description
Expression	XPath expression to be evaluated (should be XPath 1.0 or 2.0 compliant).
Value	Result of XPath expression evaluation. Value has a type (see Possible Values in the section the section called “Variables View”). For *Node Set* results the number of nodes in the set is shown in parenthesis.

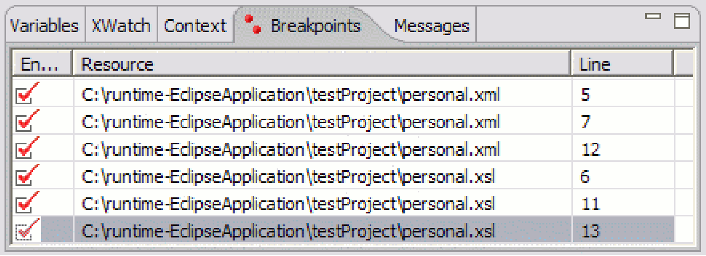
Remarks

- Expressions referring to variables names are not evaluated. In case of an XPath error, you get an Error line.
- The expression list is not deleted at the end of transformation (it is preserved during sessions).
- To insert a new expression click the last line on the expression column and enter it or right click and select the *Add* action. Press enter on cell to add and evaluate.
- To delete an expression click on its Expression column and delete its content or right click and select the *Remove* action. Press enter on cell to commit changes.
- If the expression result type is a Node Set you can click on it (Value column) and you will see on the right side its value. (see Nodeset View).
- Copy, Add, Remove and Remove All actions are offered in every row's contextual menu.

Breakpoints View

Lists all breakpoints set on opened documents. Once you set a breakpoint it is automatically added in this list. Breakpoints can be set on XSL/XQuery documents and in XML documents for XSLT debugging sessions.

Figure 11.6. The Breakpoints View



En...	Resource	Line
☑	C:\runtime-EclipseApplication\testProject\personal.xml	5
☑	C:\runtime-EclipseApplication\testProject\personal.xml	7
☑	C:\runtime-EclipseApplication\testProject\personal.xml	12
☑	C:\runtime-EclipseApplication\testProject\personal.xsl	6
☑	C:\runtime-EclipseApplication\testProject\personal.xsl	11
☑	C:\runtime-EclipseApplication\testProject\personal.xsl	13

Table 11.2. Breakpoints details

Column	Description
Enabled	If checked, the current condition is evaluated and taken into account.
Resource	Resource file where the breakpoint is set.
Line	Line number inside resource where the breakpoint is set.

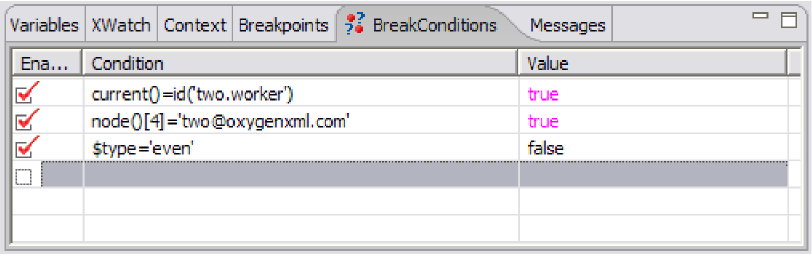
Valid Breakpoint

- Not all set breakpoints are valid. For example if the breakpoint is set on one empty or commented line or the line is not reached by the processor (no template to match it, line containing only an end tag), that breakpoint is invalid.
- Clicking a record highlights the breakpoint line into the document.

Break conditions view

Lists all defined break conditions. Unlike breakpoints, break conditions are not associated with a document, but they represent XPath expressions evaluated in the current debugger context. In order to be processed their evaluation result should be a boolean value.

Figure 11.7. The Break conditions view



Ena...	Condition	Value
☑	current()=id('two.worker')	true
☑	node()[4]='two@oxygenxml.com'	true
☑	$type='even'	false
□		

Table 11.3. Break conditions details

Column	Description
Enabled	If checked, the current condition is evaluated and taken into account.
Condition	XSLT/XQuery expression to be evaluated during debugging. The expression will be evaluated at every debug step.
Value	Boolean result of the evaluated condition or error message if the condition expression cannot be evaluated.

When the Debugger hits an active break condition it pauses the execution of the transformation and places a small marker on the left side of the line where the break condition occurred. The tooltip of the marker explains the cause of the pause. To disable further pauses when the same condition occurs you have to uncheck the *Enabled* column of the corresponding line in the *Break conditions* view.

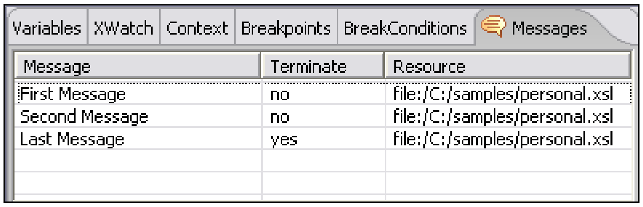
Important

- The contextual menu on table has the Add, Remove, Remove All, Enable All, Disable All options.

Messages View

`<xsl:message>` instructions are one way to signal special situations encountered during transformation as well as a raw way of doing the debugging. This view is available only for XSLT debugging sessions and shows all `<xsl:message>` calls executed by the XSLT processor during transformation.

Figure 11.8. The Messages View



Message	Terminate	Resource
First Message	no	file:/C:/samples/personal.xsl
Second Message	no	file:/C:/samples/personal.xsl
Last Message	yes	file:/C:/samples/personal.xsl

Table 11.4. Messages details

Column	Description
Message	Message content.
Terminate	Signals if processor will terminate the transformation or not once it encounters the message (true/false respectively)
Resource	Resource file where `<xsl:message>` instruction is defined.

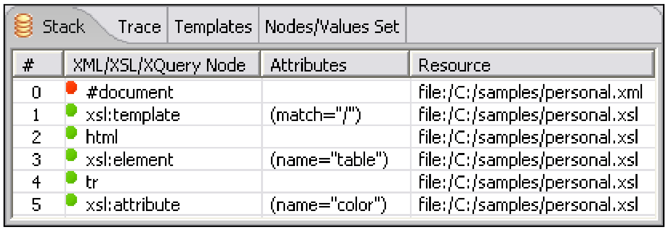
Remarks

- Clicking a record from the table highlights the `<xsl:message>` declaration line.
- Message table values can be sorted by clicking the corresponding column header (ascending, descending, no sort)

Stack View

Shows the current execution stack of both source and XSL/XQuery nodes. During transformation two stacks are managed: one of source nodes being processed and the other for XSL/XQuery nodes being processed. <oXygen/> shows both node types into one common stack. The source (XML) nodes are preceded by a red color icon while XSL/XQuery nodes are preceded by a green color icon. The advantage of this approach is that you can always see the source scope on which a XSL/XQuery instruction is executed (the last red color node on the stack). The stack is oriented upside down.

Figure 11.9. The Stack View



#	XML/XSL/XQuery Node	Attributes	Resource
0	🔴 #document		file:/C:/samples/personal.xml
1	🟢 xsl:template	(match="/")	file:/C:/samples/personal.xsl
2	🟢 html		file:/C:/samples/personal.xsl
3	🟢 xsl:element	(name="table")	file:/C:/samples/personal.xsl
4	🟢 tr		file:/C:/samples/personal.xsl
5	🟢 xsl:attribute	(name="color")	file:/C:/samples/personal.xsl

Table 11.5. Stack details

Column	Description
#	Order number, represents the depth of the node (0 is the stack base).
XML/XSL/XQuery Node	Node from source or stylesheet document currently being processed. One particular stack node is the document root, noted as #document.
Attributes	Attributes of the node (list of `id="value "` pairs).
Resource	Resource file where the node is located.

Remarks

- Clicking a record from the stack highlights that node's location inside resource.
- Using Saxon, the stylesheet elements are qualified with XSL proxy, while on Xalan you only see their names. (example `<xsl:template>` on Saxon and template on Xalan).
- Only Saxon processor shows element attributes.
- Xalan processor shows the "built-in" rules.

Trace history view

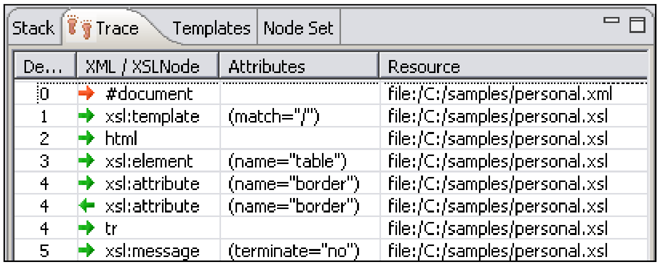
Usually the XSLT/XQuery processors signal the following events during transformation:

- → entering a source (XML) node.
- ← leaving a source (XML) node.
- → entering a XSL/XQuery node.
- ← leaving a XSL/XQuery node.

The trace history catches all these events, so you can see how the process evolved. The red icon lines denote source nodes while the green icon lines denote XSL/XQuery nodes.

It is possible to save the element trace in a structured XML document. It is available on the context menu of the view. In this way you have the possibility to compare the trace results from different debug sessions.

Figure 11.10. The Trace History View



De...	XML / XSLNode	Attributes	Resource
0	→ #document		file:/C:/samples/personal.xml
1	→ xsl:template	(match="/")	file:/C:/samples/personal.xsl
2	→ html		file:/C:/samples/personal.xsl
3	→ xsl:element	(name="table")	file:/C:/samples/personal.xsl
4	→ xsl:attribute	(name="border")	file:/C:/samples/personal.xsl
4	← xsl:attribute	(name="border")	file:/C:/samples/personal.xsl
4	→ tr		file:/C:/samples/personal.xsl
5	→ xsl:message	(terminate="no")	file:/C:/samples/personal.xsl

Table 11.6. Trace History details

Column	Description
Depth	Starts from 0 and represents the level of overlapping for that node. This is similar with the # order number from stack at the moment the node was processed.
XML/XSL/XQuery Node	Represents the node from the processed source or stylesheet document. One particular node is the document root, noted as #document. Every node has an arrow in front of it representing what action was performed on it (entering or leaving).
Attributes	Attributes of the node (list of `id="value "` pairs).
Resource	Resource file where the node is located.

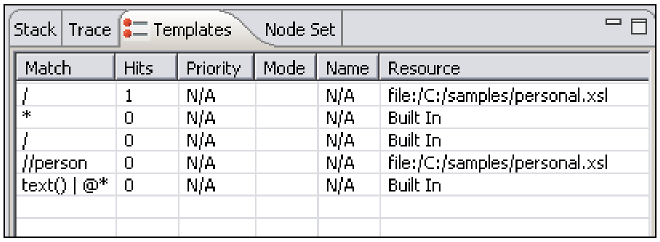
Remarks

- Clicking a record highlights that node's location inside the resource.
- Only Saxon processor shows element attributes.
- Xalan processor shows the "built-in" rules.

Templates view

The `<xsl:template>` is the basic element for stylesheets transformation. This view is only available during XSLT debugging sessions and shows all `<xsl:template>` instructions used by the transformation. By seeing the number of hits for each of the templates you get an idea of the stylesheet coverage by template rules with respect to the input source.

Figure 11.11. The Templates view



Match	Hits	Priority	Mode	Name	Resource
/	1	N/A		N/A	file:/C:/samples/personal.xsl
*	0	N/A		N/A	Built In
/	0	N/A		N/A	Built In
//person	0	N/A		N/A	file:/C:/samples/personal.xsl
text() | @*	0	N/A		N/A	Built In

Table 11.7. Templates details

Column	Description
Match	Match attribute of the `<xsl:template>`.
Hits	Number of hits for the `<xsl:template>`. Shows how many times the XSLT processor used this particular template.
Priority	Template priority as established by XSLT processor.
Mode	Mode attribute of the `<xsl:template>`.
Name	Name attribute of the `<xsl:template>`.
Resource	Resource file where template is located.

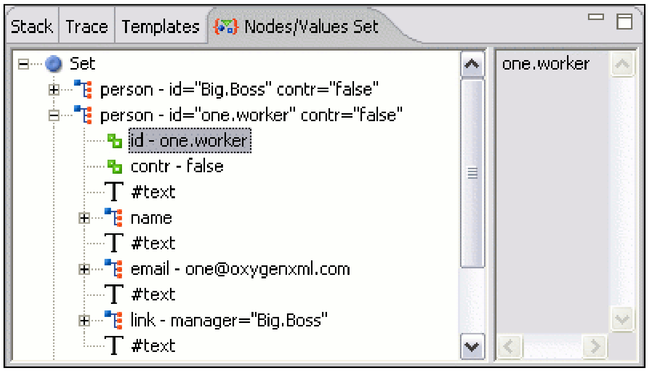
Remarks

- Clicking a record highlights that template definition inside resource.
- Saxon only shows the applied templates having at least one hit from the processor. Xalan shows all defined templates, with or without hits.
- Template table values can be sorted by clicking the corresponding column header (ascending, descending, no sort)
- Xalan shows the "built-in" rules.

Node set view

This view is always used in relation with Variables View and XWatch View and shows a nodeset value in a tree form. Once you click a variable having as value a nodeset or tree fragment or an XPath expression evaluated to a nodeset in the above views the node set view gets updated with the respective value.

Figure 11.12. The Node Set view



```
⊟ Set
  ⊞ person - id="Big.Boss" contr="false"
  ⊟ person - id="one.worker" contr="false"
      id - one.worker
      contr - false
      T #text
    ⊞ name
      T #text
    ⊞ email - one@oxygenxml.com
      T #text
    ⊞ link - manager="Big.Boss"
      T #text
```

```
one.worker
```

The nodes/values set is presented in a tree-like fashion. Nodes from a defined namespace bound to a prefix are displayed using the qualified name. If the namespace is not bound to a prefix the namespace URI will be presented before the node name. The value of the selected attribute or node are shown in the right side panel.

Remarks

- In case of longer values for Value/Attributes column content, the interface shows three suspension points (...) at the end. A more detailed value is available as tooltip.
- Clicking a record highlights the location of that node into the source or stylesheet view.

Variables View

During transformation variables and parameters play an important role.

<oXygen/> uses the following icons to differentiate variables/parameters:

- V{} Global variable.
- {V} Local variable.

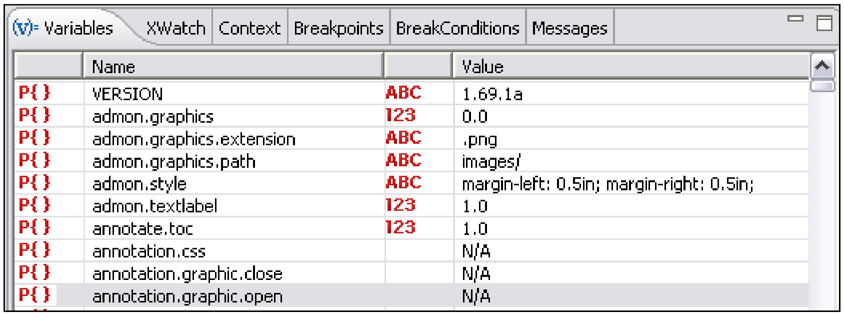
- P{ } Global parameter.
- {P} Local parameter.

The values types of a variable are marked by icons explained below:

Possible Values

- 1/0 Boolean.
- ABC String.
- 123 Numeric.
- {N} Node set.
- {...} Tree fragment.
- 7 Date. (XSLT 2.0 only)
- □ Object.
- ? Any.

Figure 11.13. The Variables View



	Name		Value
P{ }	VERSION	ABC	1.69.1a
P{ }	admon.graphics	123	0.0
P{ }	admon.graphics.extension	ABC	.png
P{ }	admon.graphics.path	ABC	images/
P{ }	admon.style	ABC	margin-left: 0.5in; margin-right: 0.5in;
P{ }	admon.textlabel	123	1.0
P{ }	annotate.toc	123	1.0
P{ }	annotation.css		N/A
P{ }	annotation.graphic.close		N/A
P{ }	annotation.graphic.open		N/A

Table 11.8. Variables details

Column	Description
Name	Name of the variable/parameter.
Value	Current value for the variable/parameter.

Remarks

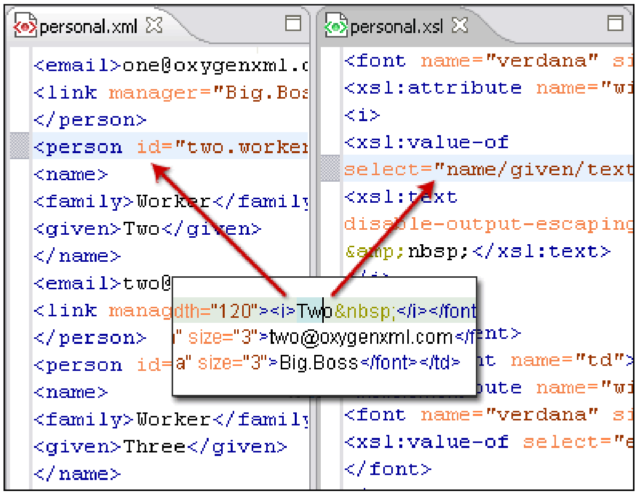
- Clicking a record highlights the variable definition line.
- Variable values could differ depending on the transformation engine used or stylesheet version set.
- If the value of the variable is a node-set or a tree-fragment, clicking on it causes the Node set view to be shown with corresponding set of values.
- Variable table values can be sorted by clicking the corresponding column header (ascending, descending, no sort)

Determining what XSL/XQuery expression generated particular output

In order to quickly spot the XSL templates or XQuery expressions with problems it is important to know what XSL template in the XSL stylesheet or XQuery expression in the XQuery document and what element in the source XML document generated a specified area in the output. Some of the debugging capabilities, for example "Step in" can be used for this purpose. Using "Step in" you can see how output is generated and link it with the XSL/XQuery element being executed in the current source context. However, this can become difficult on complex stylesheets or XQuery documents that generates a large output.

Output to source mapping is a powerful feature that makes this output to source mapping persistent that is you can click on the text from the Output document view and the editor will select the XML source context and the XSL/XQuery element that generated the text.

Figure 11.14. Output to Source Mapping



```
<email>one@oxygenxml.c
<link manager="Big.Bos
</person>
<person id="two.worker
<name>
<family>Worker</family
<given>Two</given>
</name>
<email>two@
<link manag
</person>
<person id=
<name>
<family>Worker</family
<given>Three</given>
</name>
```

```
<font name="verdana" si
<xsl:attribute name="wi
<i>
<xsl:value-of
select="name/given/text
<xsl:text
disable-output-escaping
&amp;nbsp;</xsl:text>
         nt>
         nt name="td">
         bute name="wi
<font name="verdana" si
<xsl:value-of select="e
</font>
```

```
dth="120"><i>Two&nbsp;</i></font
" size="3">two@oxygenxml.com</f
a" size="3">Big.Boss</font></td>
```

1. If you are in the <oXygen/> XML perspective switch to the <oXygen/> XSLT Debugger or <oXygen/> XQuery Debugger perspective with one of the actions (here explained for XSLT):

   - Window → Open Perspective → Other ...+<oXygen/> XSLT Debugger
   - The toolbar button Debug scenario . This action initializes the Debugger perspective with the parameters of the transformation scenario. Any modification applied to the scenario parameters (the transformer engine, the XSLT parameters, the transformer extensions, etc) will be saved back in the scenario when exiting from the Debugger perspective.

2. Select the source XML document in the XML source selector of the Control toolbar. In case of XQuery debugging without an implicit source choose the NONE value.
3. Select the XSL/XQuery document in the XSL/XQuery selector of the Control toolbar
4. Select the XSLT/XQuery engine in the XSLT/XQuery engine selector of the Control toolbar
5. Set XSLT/XQuery parameters from the button available on the Control toolbar
6. Apply the stylesheet or XQuery transformation using the button Run to end available on the Control toolbar:

7. Inspect the mapping by clicking a section of the output from the Output view of the <oXygen/> XSLT Debugger or <oXygen/> XQuery Debugger perspectives to have the XSL/XQuery element and the source context highlighted.

Chapter 12. Profiling XSLT stylesheets and XQuery documents

Overview

Whether you are trying to identify a performance issue that is causing your production XSLT/XQuery transformation to not meet customer expectations or you are trying to proactively identify issues prior to deploying your XSLT/XQuery transformation, using the XSLT/XQuery profiler feature is essential to helping you save time and ultimately ensure a better performing, more scalable XSLT/XQuery transformation.

The XSLT/XQuery profiling feature can use any available XSLT/XQuery processors that could be used for debugging and it is available from the editor debugging perspective.

Enabling/disabling the profiler is controlled by the Profiler button from the debugger control toolbar. The XSLT/XQuery profiler is off by default. This option is not available during a debugger session so you should set it before starting the transformation.

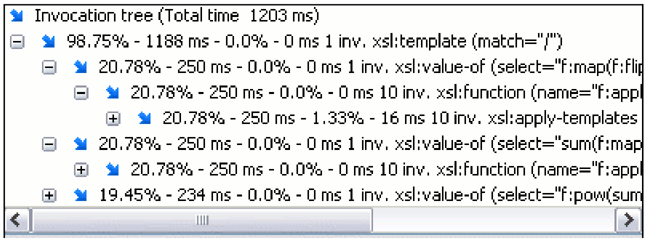
Viewing profiling information

Detailed profiling information for the current transformation is provided using the information views:

Invocation tree view

The invocation tree view shows a top-down call tree representing how XSLT instructions or XQuery expressions are processed.

Figure 12.1. Invocation tree view



```
Invocation tree (Total time  1203 ms)
⊟ 98.75% - 1188 ms - 0.0% - 0 ms 1 inv. xsl:template (match="/")
   ⊟ 20.78% - 250 ms - 0.0% - 0 ms 1 inv. xsl:value-of (select="f:map(f:flip
      ⊟ 20.78% - 250 ms - 0.0% - 0 ms 10 inv. xsl:function (name="f:appl
         ⊞ 20.78% - 250 ms - 1.33% - 16 ms 10 inv. xsl:apply-templates
   ⊟ 20.78% - 250 ms - 0.0% - 0 ms 1 inv. xsl:value-of (select="sum(f:map
      ⊞ 20.78% - 250 ms - 0.0% - 0 ms 10 inv. xsl:function (name="f:appl
   ⊞ 19.45% - 234 ms - 0.0% - 0 ms 1 inv. xsl:value-of (select="f:pow(sum
```

The entries in the invocation tree have different meanings which are indicated by the displayed icons:

- ↘ This points to a call whose inherent time is insignificant compared to its call tree time.

- ⏱ This points to a call whose inherent time is significant compared to its call tree time. (greater than 1/3rd of its call tree time).

Every entry in the invocation tree has textual information attached which depends on the XSLT/XQuery profiler settings

- a percentage number of total time which is calculated with respect to either the root of the tree or the calling instruction;

- a total time measurement in ms or µs. This is the total execution time that includes calls into other instructions;

- a percentage number of inherent time which is calculated with respect to either the root of the tree or the calling instruction;
- an inherent time measurement in ms or µs. This is the inherent execution time of the instruction;
- an invocation count which shows how often the instruction has been invoked on this path;
- an instruction name which contains also the attributes description.

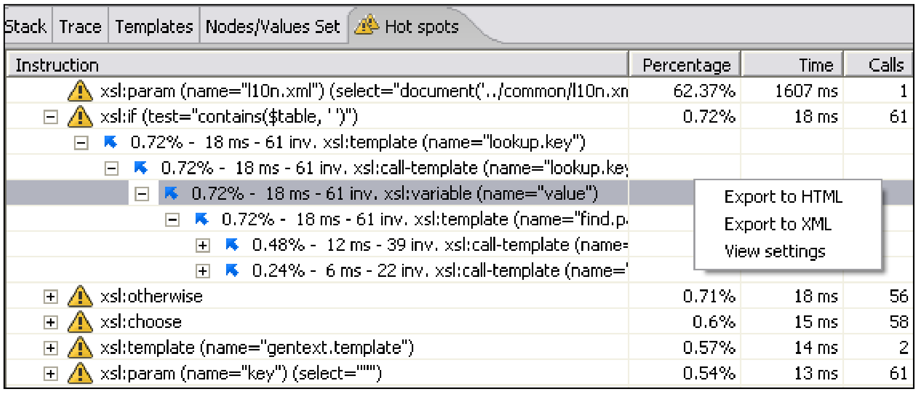
Note

All nodes having their call tree time less than the one specified in the XSLT/XQuery profiler settings are cumulated and shown as *Others* node.

Hotspots View

The hotspots view shows a list of all instruction calls which lie above the threshold defined in the XSLT/XQuery profiler settings .

Figure 12.2. Hotspots View



Instruction	Percentage	Time	Calls
⚠ xsl:param (name="l10n.xml") (select="document('../common/l10n.xn	62.37%	1607 ms	1
⊟ ⚠ xsl:if (test="contains($table, ' ')")	0.72%	18 ms	61
⊟ 0.72% - 18 ms - 61 inv. xsl:template (name="lookup.key")			
⊟ 0.72% - 18 ms - 61 inv. xsl:call-template (name="lookup.key			
⊟ 0.72% - 18 ms - 61 inv. xsl:variable (name="value")			
⊟ 0.72% - 18 ms - 61 inv. xsl:template (name="find.p.			
⊞ 0.48% - 12 ms - 39 inv. xsl:call-template (name=			
⊞ 0.24% - 6 ms - 22 inv. xsl:call-template (name='			
⊞ ⚠ xsl:otherwise	0.71%	18 ms	56
⊞ ⚠ xsl:choose	0.6%	15 ms	58
⊞ ⚠ xsl:template (name="gentext.template")	0.57%	14 ms	2
⊞ ⚠ xsl:param (name="key") (select="''")	0.54%	13 ms	61

By opening a hotspot instruction entry, the tree of back-traces leading to that instruction call are calculated and shown.

Every hotspot is described in several columns:

- the instruction name;
- the inherent time in ms or µs of how much time has been spent in the hotspot together with a bar whose length is proportional to this value. All calls into this instruction are summed up regardless of the particular call sequence;
- the invocation count of the hotspot.

If you click on the ⚠ handle on the left side of a hotspot, a tree of back-traces will be shown.

Every entry in the backtrace tree has textual information attached to it which depends on the XSLT/XQuery profiler settings .

- a percentage number which is calculated with respect either to the total time or the called instruction;
- a time measured in ms or µs of how much time has been contributed to the parent hotspot on this path;
- an invocation count which shows how often the hotspot has been invoked on this path;

Note

This is not the number of invocations of this instruction.

- an instruction name which contains also its attributes.

Working with XSLT/XQuery profiler

Profiling activity is linked with Debugging activity, so the first step in order to profile is to switch to debugging perspective and follow the corresponding procedure (see Working with XSLT Debugger).

Immediately after turning the profiler on two new information views are added to the current debugger information views (Invocation tree view on left side, Hotspots view on right side). Profiling data is available only when the transformation ends successfully.

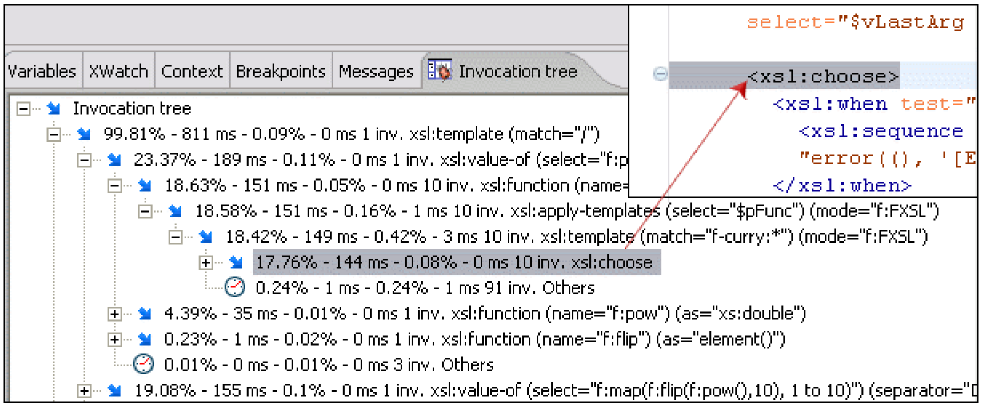
Note

Breakpoints/step capabilities may influence the result of profiling so their usage should be restricted to minimum.

Looking to right side (Hotspots view), you can immediately spot the time the processor spent in each instruction. As instruction usually calls other instructions the used time of the called instruction is extracted from the duration time of the caller (the hotspot only presents the inherent time of the instruction).

Looking at left side (Invocation tree view), you can examine how style instructions are processed. This result view is also named call-tree, as it represents the order of style processing. The profiling result shows the duration time for each of the style-instruction including the time needed for its called children.

Figure 12.3. Source backmapping



```
Invocation tree
  99.81% - 811 ms - 0.09% - 0 ms 1 inv. xsl:template (match="/")
    23.37% - 189 ms - 0.11% - 0 ms 1 inv. xsl:value-of (select="f:p
      18.63% - 151 ms - 0.05% - 0 ms 10 inv. xsl:function (name=
        18.58% - 151 ms - 0.16% - 1 ms 10 inv. xsl:apply-templates (select="$pFunc") (mode="f:FXSL")
          18.42% - 149 ms - 0.42% - 3 ms 10 inv. xsl:template (match="f-curry:*") (mode="f:FXSL")
            17.76% - 144 ms - 0.08% - 0 ms 10 inv. xsl:choose
            0.24% - 1 ms - 0.24% - 1 ms 91 inv. Others
      4.39% - 35 ms - 0.01% - 0 ms 1 inv. xsl:function (name="f:pow") (as="xs:double")
      0.23% - 1 ms - 0.02% - 0 ms 1 inv. xsl:function (name="f:flip") (as="element()")
      0.01% - 0 ms - 0.01% - 0 ms 3 inv. Others
    19.08% - 155 ms - 0.1% - 0 ms 1 inv. xsl:value-of (select="f:map(f:flip(f:pow(),10), 1 to 10)") (separator="
```

```
            select="$vLastArg

        <xsl:choose>
          <xsl:when test="
            <xsl:sequence
            "error((), '[E
          </xsl:when>
```

In any of the above views you can use the backmapping feature in order to find the XSLT stylesheet or XQuery expression definition. Clicking on the selected item cause <oXygen/> to highlight the XSLT stylesheet or XQuery expression source line where the instruction is defined.

When navigating through the trees by opening instruction calls, <oXygen/> automatically expands instructions which are only called by one other instruction themselves.

The profiling data can be saved into XML and HTML format. On any view you should right click , use the pop-up menu and select the corresponding choice. Basically saving HTML means saving XML and applying an XSLT stylesheet to render the report as XML. These stylesheets are also available on distribution (see the subdirectory `frame-`

`works/profiler/` of the <oXygen/> installation directory) so you can make your own report based on the profiling raw data.

If you like to change the XSLT/XQuery profiler settings you should right click on view, use the pop-up menu and choose the corresponding "View settings" entry.

Caution

Profiling exhaustive transformation may run into an OutOfMemoryException due to the large amount of information being collected. If this is the case you can close unused projects when running the profiling or use high values for Java VM options -Xms and -Xmx. If this does not help you can shorten your source xml file and try again.

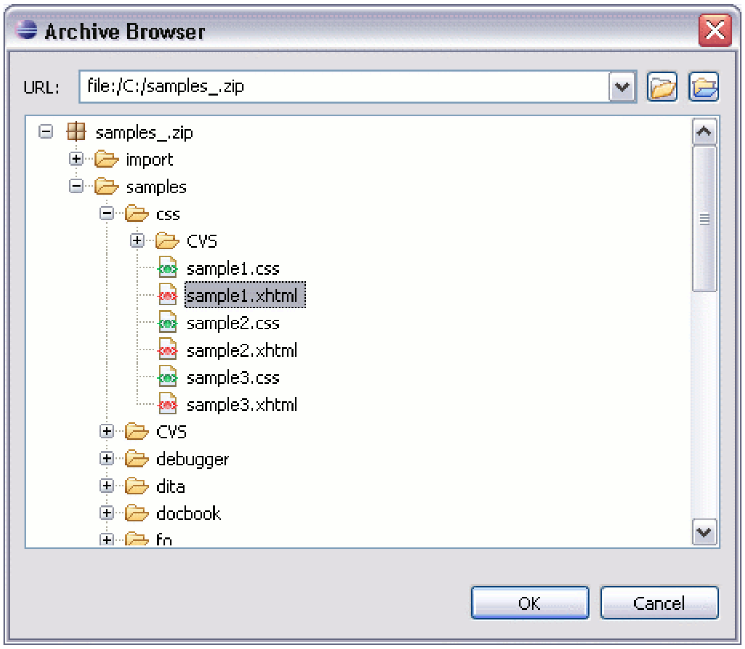
Chapter 13. Working with Archives

<oXygen/> offers the means to manipulate files directly from ZIP type archives. By manipulation one should understand opening and saving files directly in archives, browsing and modifying archive structures. The archive support is available for all ZIP-type archives, for JAR and ODF formats and for IDML files which are also based on the ZIP archive format. This means that you can modify, transform, validate files directly from OOXML or ODF packages.

Using files directly from archives

Now you can transform, validate and perform many other operations on files directly from an archive. When selecting an URL for a specific operation like transformation or validation you can click the Browse for archived file button to navigate and choose the file from a certain archive.

Figure 13.1. Browsing for a file in an archive



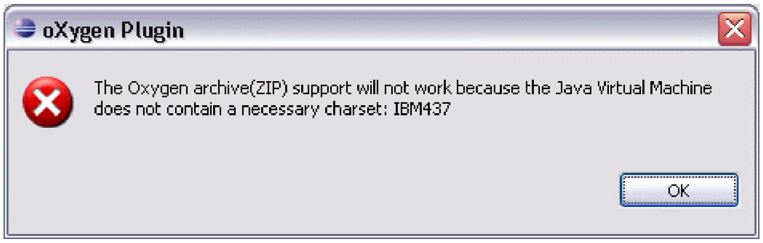
Browsing and modifying archives' structure

You can navigate archives directly in the Archives Browser either by opening them from the Navigator or by using the integration with the Eclipse File System.

For the EFS (Eclipse File System) integration you must right click the archive in the Navigator and choose *Expand Zip Archive*. All the standard Eclipse Navigator actions are available on the mounted archive. If you decide to close the archive you can use the *Collapse ZIP Archive* action located in the contextual menu for the expanded archive. Any file opened from the archive expanded in the EFS will be closed when the archive in unmounted.

Warning

The ZIP support needs the IBM437 character set to be properly installed in the Java Runtime Environment in order to be able to navigate/open ZIP archives. If you encounter the following error message when expanding



oXygen Plugin

The Oxygen archive(ZIP) support will not work because the Java Virtual Machine does not contain a necessary charset: IBM437

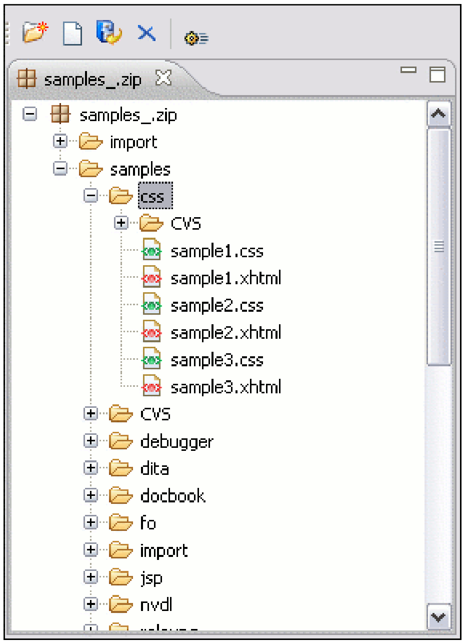
ZIP archives then the JRE used to run Eclipse does not have the character set library properly installed.

If you open an archive as an Eclipse editor, the archive will be unmounted when the editor is closed.

Important

If a file extension is not known by <oXygen/> as a supported archive type you can add it from the Archive preferences page .

Figure 13.2. Browsing an archive



The following operations are available on the Archive Browser's toolbar:

New folder... Create a new folder as child of the selected folder in the browsed archived.

New file... Create a new file as child of the selected folder in the browsed archived.

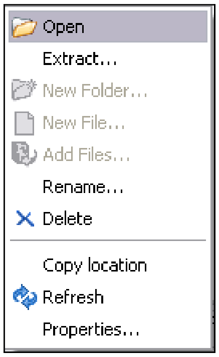
Add files... Add some already existing files as children of the selected folder in the browsed archived.

Delete Delete the selected resource in the browsed archived.

Archive Options... Open the Archive preferences page.

The following additional operations are available from the Archive Browser's contextual menu:

Figure 13.3. Contextual menu



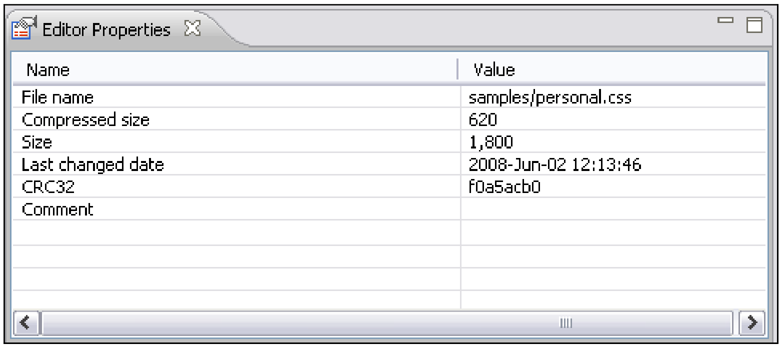
Open — Open a resource from the archive in the editor.

Copy location — Copy the URL location of the selected resource.

Refresh — Refresh the selected resource.

Properties... — View properties for the selected resource.

Figure 13.4. Archive resource properties



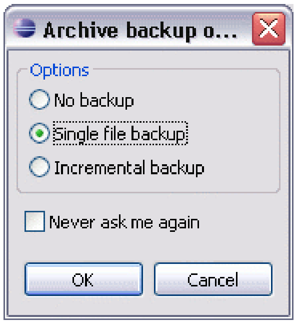
Name	Value
File name	samples/personal.css
Compressed size	620
Size	1,800
Last changed date	2008-Jun-02 12:13:46
CRC32	f0a5acb0
Comment	

Editing files from archives

You can open in <oXygen/> and edit files directly from an archive.

When saving the archived file you will be prompted with some backup operations which can be performed to ensure that your archive data will not be corrupted.

Figure 13.5. Archive Backup options



Archive backup o...

Options

○ No backup

● Single file backup

○ Incremental backup

□ Never ask me again

You have the following backup before save options :

No backup

Perform no backup of the archive before save. This means that the file will be saved directly in the archive without any additional precautions.

Single file backup

Before any operation which modifies the archive is performed, the archive contents will be duplicated. The duplicate file name will be `originalArchiveFileName.bak` and will be saved in the same directory.

Incremental backup

Before each operation which modifies the archive is performed, the archive contents will be duplicated. The duplicate file names will be `originalArchiveFileName.bak#dupNo` and the files will be saved in the same directory.

Never ask me again

Check this if you do not want to be notified again to backup. The last backup option you chose will always be used as the default one.

You can re-enable the dialog pop-up from the Archive preferences page.

Chapter 14. Working with Databases

XML is a storage and interchange format for structured data and it is supported by all major database systems. <oXygen/> offers the means of managing the interaction with some of the widely used databases, both relational ones and Native XML Databases. By interaction, one should understand browsing, querying, SQL execution support, content editing, importing from databases, generating XML Schema from database structure.

Relational Database Support

Relational databases use a relational model and are based on tables linked by a common key. <oXygen/> offers support for the following relational databases: IBM DB2, JDBC-ODBC Bridge, MySQL, Microsoft SQL Server, Oracle 11g like browsing the tables of these types of database in the *Database Explorer* view, executing SQL queries against them, calling stored procedures with input and output parameters.

In the following sections one can find the tools that <oXygen/> offers for working with relational databases and a description on how to configure a relational data source, a connection to a data source and also the views where connections can be browsed and results are displayed.

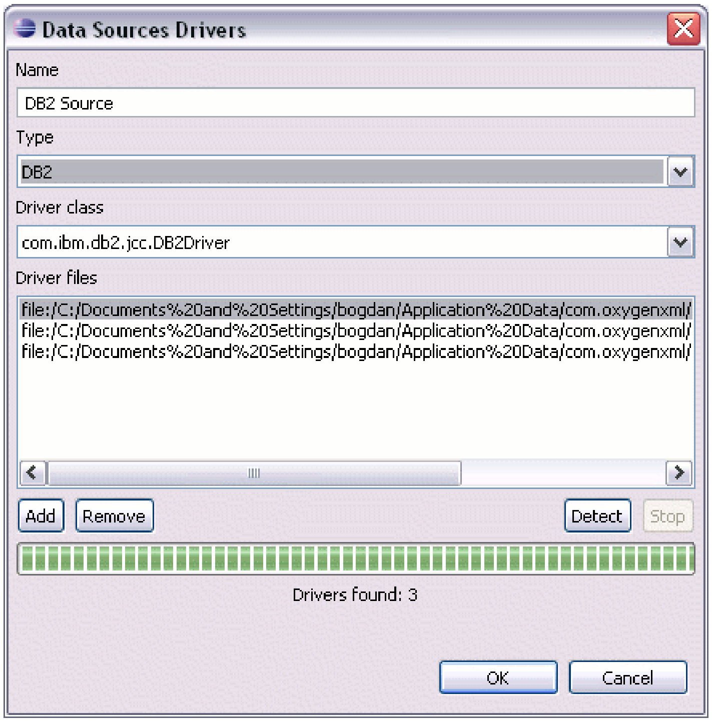
Configuring Database Data Sources

How to configure an IBM DB2 Data Source

1. Go to *Preferences* -> *Data Sources*. In the Data Sources panel click the *New* button.

2. Enter a unique name for this data source and select *DB2* from the driver type combo box.

Figure 14.1. Data Source Drivers Configuration Dialog



Data Sources Drivers

Name: DB2 Source

Type: DB2

Driver class: com.ibm.db2.jcc.DB2Driver

Driver files:

```
file:/C:/Documents%20and%20Settings/bogdan/Application%20Data/com.oxygenxml/
file:/C:/Documents%20and%20Settings/bogdan/Application%20Data/com.oxygenxml/
file:/C:/Documents%20and%20Settings/bogdan/Application%20Data/com.oxygenxml/
```

Drivers found: 3

Press the Add button to add the following IBM DB2 specific files:

- db2jcc.jar
- db2jcc_license_cisuz.jar
- db2jcc_license_cu.jar

In the Download links for database drivers section there are listed the URLs from where to download the drivers necessary for accessing IBM DB2 databases in <oXygen/>.

You can manually manage the Driver Files using *Add*, *Remove*, *Detect* and *Stop*(detection) buttons.

3. Select the most suited *Driver class*.
4. Click *OK* to finish the data source configuration.

How to configure a Generic JDBC Data Source

<oXygen/>'s default configuration already contains a generic JDBC data source called *JDBC-ODBC Bridge*.

1. Go to *Preferences -> Data Sources*. In the Data Sources panel click the *New* button.
2. Enter a unique name for this data source and select *Generic JDBC* from the driver type combo box.

   Click the *Add* button and find the driver file on your file system.

   You can manage the Driver Files using *Add*, *Remove*, *Detect* and *Stop*(detection) buttons.
3. Select the most suited *Driver class*.
4. Click *OK* to finish the data source configuration.

How to configure a Microsoft SQL Server Data Source

1. Go to *Preferences -> Data Sources*. In the Data Sources panel click the *New* button.
2. Enter a unique name for this data source and select *SQLServer* from the driver type combo box.
3. Press the Add button to add the following Microsoft SQL Server specific files:

   - `sqljdbc.jar`

   In the Download links for database drivers section there are listed the URLs from where to download the drivers necessary for accessing Microsoft SQL Server databases in <oXygen/>.

   You can manage the Driver Files using *Add*, *Remove*, *Detect* and *Stop*(detection) buttons.
4. Select the most suited *Driver class*.
5. Click *OK* to finish the data source configuration.

How to configure a MySQL Data Source

<oXygen/>'s default configuration already contains a generic JDBC data source called *MySQL*.

1. Go to *Preferences -> Data Sources*. In the Data Sources panel click the *New* button.

2. Enter a unique name for this data source and select *Generic JDBC* from the driver type combo box.

   Press the Add button to add the following MySQL specific files:

   - mysql-com.jar

   You can manage the Driver Files using *Add*, *Remove*, *Detect* and *Stop*(detection) buttons.

3. Select the most suited *Driver class*.

4. Click *OK* to finish the data source configuration.

How to configure an Oracle 11g Data Source

1. Go to *Preferences -> Data Sources*. In the Data Sources panel click the *New* button.

2. Enter a unique name for this data source and select *Oracle* from the driver type combo box.

   Press the Add button to add the following Oracle 10.2 specific files:

   - ojdbc5.jar

   In the Download links for database drivers section there are listed the URLs from where to download the drivers necessary for accessing Oracle 11g databases in <oXygen/>.

   You can manage the Driver Files using *Add*, *Remove*, *Detect* and *Stop*(detection) buttons.

3. Select the most suited *Driver class*.

4. Click *OK* to finish the data source configuration.

How to configure a PostgreSQL 8.3 Data Source

1. Go to *Preferences -> Data Sources*. In the Data Sources panel click the *New* button.

2. Enter a unique name for this data source and select `Postgres` from the driver type combo box.

   Press the Add button to add the following `Postgres` 8.3 specific files:

   - `postgresql-8.3-603.jdbc3.jar`

   In the Download links for database drivers section there are listed the URLs from where to download the drivers necessary for accessing PostgreSQL databases in <oXygen/>.

   You can manage the Driver Files using *Add*, *Remove*, *Detect* and *Stop*(detection) buttons.

3. Select the *org.postgresql.Driver* class in the *Driver class* combo box.

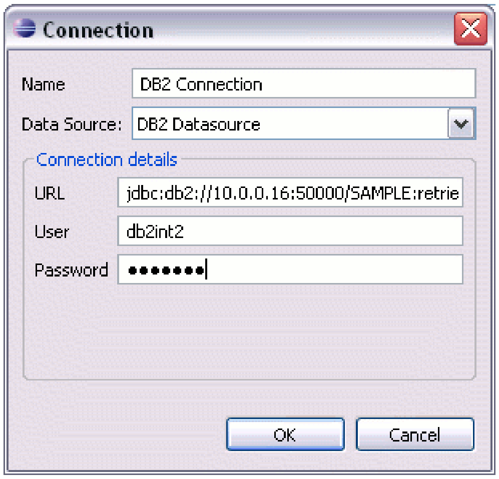
4. Click *OK* to finish the data source configuration.

Configuring Database Connections

This section presents a set of procedures describing how to configure connections that use relational data sources.

How to Configure an IBM DB2 Connection

Figure 14.2. The Connection Configuration Dialog



Name: DB2 Connection
Data Source: DB2 Datasource
Connection details
URL: jdbc:db2://10.0.0.16:50000/SAMPLE:retrie
User: db2int2
Password: •••••••

1. Go to *Preferences -> Data Sources*. In the Connections panel click the *New* button.
2. Enter a unique name for this connection and select one of the previously configured DB2 data sources from the Data Source combo box.
3. Fill-in the Connection Details:

   URL — URL to the installed IBM DB2 engine.

   User — User name to access the IBM DB2 database engine.

   Password — Password to access the IBM DB2 engine.

4. Click *OK*.

How to Configure a JDBC-ODBC Connection

1. Go to *Preferences -> Data Sources*. In the Connections panel click the *New* button.
2. Enter a unique name for this connection and select one of the previously configured Generic JDBC data sources from the Data Source combo box.
3. Fill-in the Connection Details:

   URL — URL to the configured ODBC source.

   User — User name to access the configured ODBC source.

   Password — Password to access the configured ODBC source.

4. Click *OK*.

How to Configure a Microsoft SQL Server Connection

1. Go to *Preferences -> Data Sources*. In the Connections panel click the *New* button.

2. Enter a unique name for this connection and select one of the previously configured SQLServer data sources from the Data Source combo box.

3. Fill-in the Connection Details:

   URL URL to the installed SQLServer engine.

   User User name to access the SQLServer database engine.

   Password Password to access the SQLServer engine.

4. Click *OK*.

How to Configure a MySQL Connection

1. Go to *Preferences -> Data Sources*. In the Connections panel click the *New* button.

2. Enter a unique name for this connection and select one of the previously configured MySQL data sources from the Data Source combo box.

3. Fill-in the Connection Details:

   URL URL to the installed MySQL engine.

   User User name to access the MySQL database engine.

   Password Password to access the MySQL engine.

4. Click *OK*.

How to Configure an Oracle 11g Connection

1. Go to *Preferences -> Data Sources*. In the Connections panel click the *New* button.

2. Enter a unique name for this connection and select one of the previously configured Oracle data sources from the Data Source combo box.

3. Fill-in the Connection Details:

   URL URL to the installed Oracle engine.

   User User name to access the Oracle database engine.

   Password Password to access the Oracle engine.

4. Click *OK*.

Note

Registering,unregistering or updating a schema might involve dropping/creating types. For schema-based XMLType tables or columns in schemas, you need privileges like

- CREATE ANY TABLE
- CREATE ANY INDEX
- SELECT ANY TABLE
- UPDATE ANY TABLE
- INSERT ANY TABLE
- DELETE ANY TABLE
- DROP ANY TABLE
- ALTER ANY TABLE
- DROP ANY INDEX

To avoid granting these privileges to the schema owner, Oracle recommends that the operations requiring these privileges be performed by a DBA if there are XML schema-based XMLType table or columns in other users' database schemas.

How to Configure a PostgreSQL 8.3 Connection

1. Go to *Preferences -> Data Sources*. In the Connections panel click the *New* button.
2. Enter a unique name for this connection and select one of the previously configured PostgreSQL data sources from the Data Source combo box.
3. Fill-in the Connection Details:

   URL URL to the installed PostgreSQL engine.

   User User name to access the PostgreSQL database engine.

   Password Password to access the PostgreSQL engine.

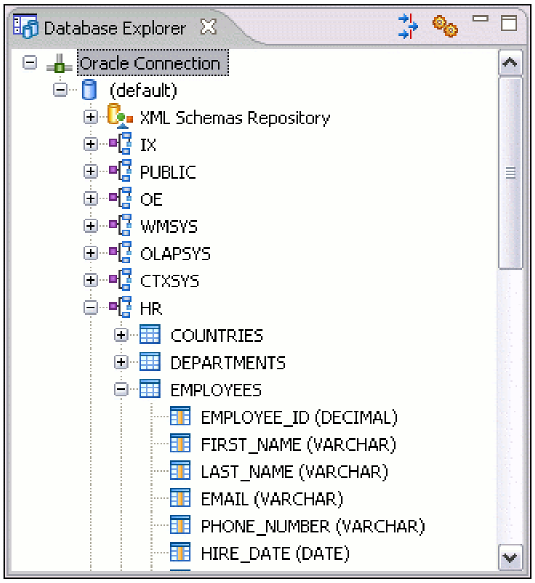
4. Click *OK*.

Resource Management

Database Explorer View

This view presents in a tree-like fashion the database connections configured in *Preferences -> Data Sources*. You can connect to a database simply by expanding the connection node. The database structure can be expanded up to column level. <oXygen/> supports multiple simultaneous database connections and the connections tree provides an easy way to browse them.

Figure 14.3. Database Explorer View



```
Oracle Connection
  (default)
    XML Schemas Repository
    IX
    PUBLIC
    OE
    WMSYS
    OLAPSYS
    CTXSYS
    HR
      COUNTRIES
      DEPARTMENTS
      EMPLOYEES
        EMPLOYEE_ID (DECIMAL)
        FIRST_NAME (VARCHAR)
        LAST_NAME (VARCHAR)
        EMAIL (VARCHAR)
        PHONE_NUMBER (VARCHAR)
        HIRE_DATE (DATE)
```

The following objects are displayed by the Database Explorer view:

- Connection
- Catalog
- XML Schema Repository
- XML Schema Component
- Schema
- Table
- System Table
- Table Column

The following actions are available in the view's toolbar:

- The Filters button opens the *Data Sources / Table Filters*Preferences page, allowing you to decide which table types will be displayed in the *Database Explorer* view.

- The Configure Database Sources button opens the *Data Sources* preferences page where you can configure both data sources and connections.

Below you can find a description of the contextual menu actions available on the Database Explorer levels. Please note that you can also open an XML schema component in the editor by double-clicking it. To view the content of a table in the Table Explorer view double-click one of its fields.

Actions available at connection level

- Refresh - performs a refresh of the selected node's subtree.
- Configure Database Sources - opens the *Data Sources*preferences page where you can configure both data sources and connections.

Actions available at catalog level

- Refresh - performs a refresh of the selected node's subtree.

Actions available at schema level

- Refresh - performs a refresh of the selected node's subtree.

Actions available at table level

- Refresh - performs a refresh of the selected node's subtree.
- Edit - opens the selected table in the *Table Explorer* View.
- Export to XML - opens the Export Criteria dialog (a thorough description of this dialog can be found in the Import from database chapter) .

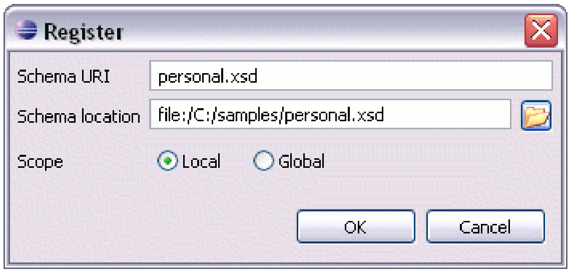
XML Schema Repository level

For relational databases that support XML schema repository (XSR) in their database catalogs, the actions available at this level are presented in the following sections.

Oracle's XML Schema Repository Level

- Refresh - performs a refresh of the selected node's subtree.
- Register - Opens a dialog for adding a new schema file in the DB XML repository.

Figure 14.4. Register Dialog



Schema URI: personal.xsd

Schema location: file:/C:/samples/personal.xsd

Scope: ● Local ○ Global

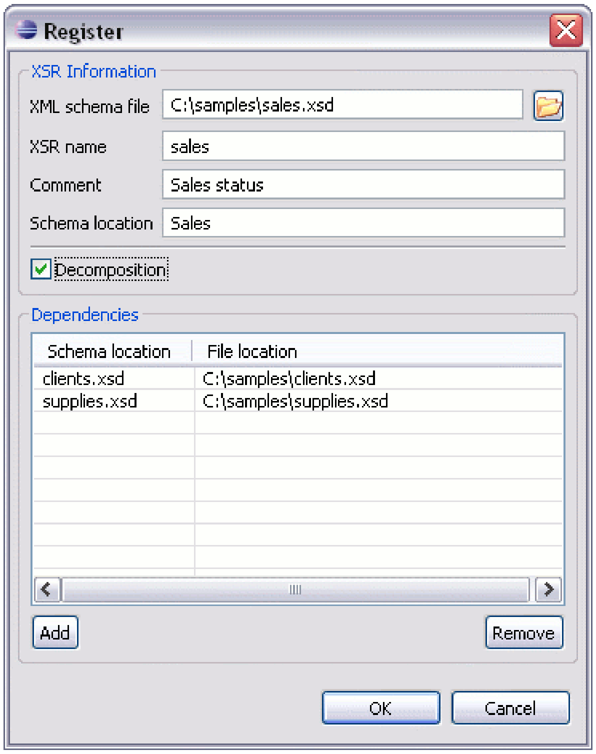
To add an XML Schema, enter the schema URI and location on your file system. Local scope means that the schema will be visible only to the user who registers it. Global scope means that the schema is public.

IBM DB2's XML Schema Repository Level

- Refresh - performs a refresh of the selected node's subtree.

- Register - opens a dialog for adding a new schema file in the XML Schema repository.

Figure 14.5. Register Dialog



XML schema file: C:\samples\sales.xsd

XSR name: sales

Comment: Sales status

Schema location: Sales

☑ Decomposition

Schema location	File location
clients.xsd	C:\samples\clients.xsd
supplies.xsd	C:\samples\supplies.xsd

The XSR Information section of the above figure contains the following fields:

- *XML schema file* - location on your file system.
- *XSR name* - schema name.
- *Comment* - short comment (optional).
- *Schema location* - primary schema name (optional).

Decomposition means that parts of the XML documents are stored into relational tables. Which parts map to which tables and columns is specified into the schema annotations.

Schema dependencies management is done by using the *Add* and *Remove* buttons.

Actions available at schema level:

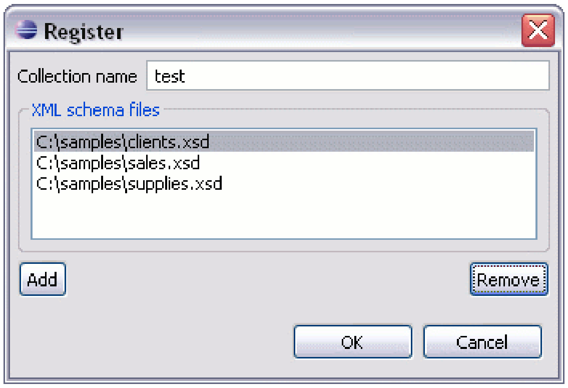
- Refresh - performs a refresh of the selected node (and it's subtree).
- Unregister - removes the selected schema from the XML Schema Repository.
- View - opens the selected schema in <oXygen/>.

Microsoft SQL Server's XML Schema Repository Level

- Refresh - performs a refresh of the selected node's subtree.

- Register - Opens a dialog for adding a new schema file in the DB XML repository.

Figure 14.6. Register Dialog



Collection name: test

C:\samples\clients.xsd
C:\samples\sales.xsd
C:\samples\supplies.xsd

To register a new schema, enter a collection name and the necessary schema files in the above dialog. XML Schema files management is done by using the *Add* and *Remove* buttons.

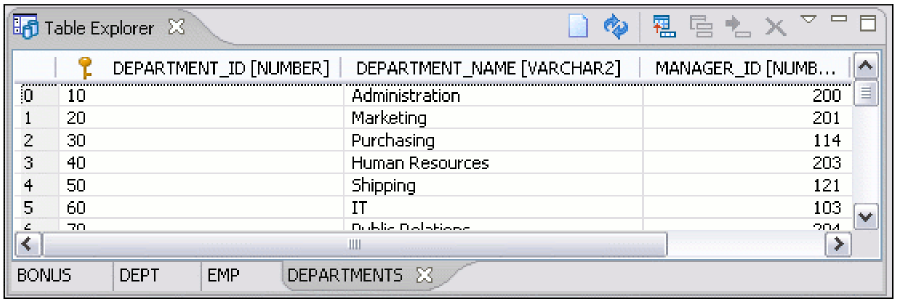
Actions available at schema level:

- Refresh - performs a refresh of the selected node (and it's subtree).
- Add - adds a new schema to the XML Schema files.
- Unregister - removes the selected schema from the XML Schema Repository.
- View - opens the selected schema in <oXygen/>.

Table Explorer View

Every table from the Database Explorer can be displayed and edited by pressing the *Edit* button from the contextual menu or by double-clicking one of its fields. To modify a cell's content, double click it and start typing. When editing is finished, <oXygen/> will try to update the database with the new cell content.

Figure 14.7. The Table Explorer View



	DEPARTMENT_ID [NUMBER]	DEPARTMENT_NAME [VARCHAR2]	MANAGER_ID [NUMB...
0	10	Administration	200
1	20	Marketing	201
2	30	Purchasing	114
3	40	Human Resources	203
4	50	Shipping	121
5	60	IT	103
6	70	Public Relations	204

You can sort the content of a table by one of its columns by clicking on its (column) header.

Note the following:

- The first column is an index (does not belong to the table structure).

- Every column header contains the field name and its data type.
- The primary key columns are marked with this symbol: 🔑 .
- Multiple tables are presented in a tabbed manner

For performance issues, you can set the maximum number of cells that will be displayed in the Table Explorer view (the "Limit the number of cells" field from the Data Sources Preferences page). If a table having more cells than the value set in <oXygen/>'s options is displayed in the Table Explorer view, a warning dialog will inform you that the table is only partially shown.

Note

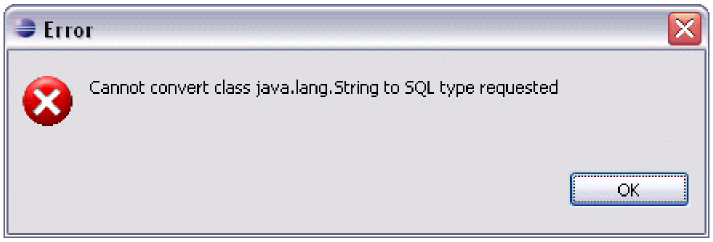
A custom validator cannot be applied on files loaded through an <oXygen/> custom protocol plugin developed independently and added to <oXygen/> after installation. This applies also on columns of type XML.

You will be notified if the value you have entered in a cell is not valid (and thus it cannot be updated).

- If the content of the edited cell does not belong to the data type of the column, an Information dialog will appear, notifying you that the value you have inserted cannot be converted to the SQL type of that field.

  For example, in the above figure *DEPARTMENT_ID* contains *NUMBER* values. If a character or string was inserted, you would get the following message:

Figure 14.8. Cell containing an invalid value.

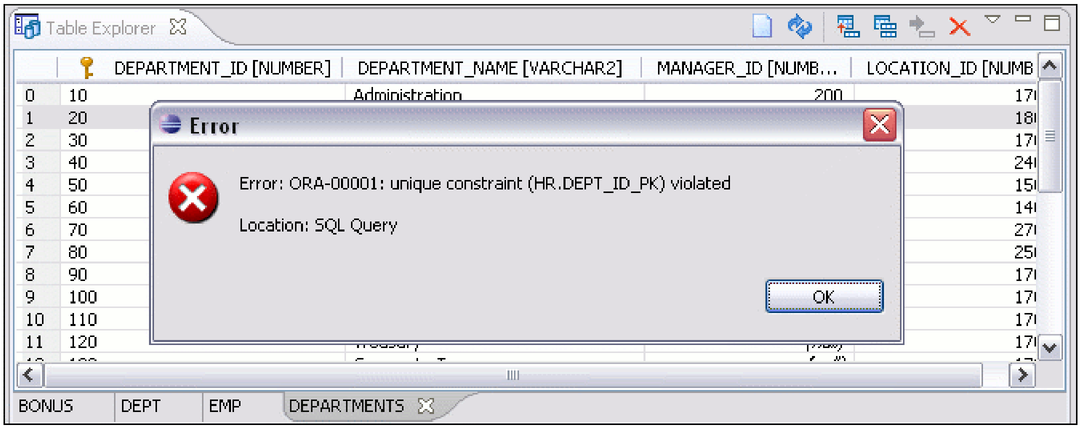


Cannot convert class java.lang.String to SQL type requested

- If the constraints of the database are not met (like primary key constraints for example), an Information dialog will appear, notifying you of the reason the database has not been updated.

  For example, if you'd try to set the primary key *DEPARTMENT_ID* for the second record in the table to 10 also, you would get the following message:

Figure 14.9. Duplicate entry for primary key



	DEPARTMENT_ID [NUMBER]	DEPARTMENT_NAME [VARCHAR2]	MANAGER_ID [NUMB...	LOCATION_ID [NUMB
0	10	Administration	200	17
1	20			18
2	30			17
3	40			24
4	50			15
5	60			14
6	70			27
7	80			25
8	90			17
9	100			17
10	110			17
11	120			17

Error

Error: ORA-00001: unique constraint (HR.DEPT_ID_PK) violated

Location: SQL Query

OK

The usual edit actions (Cut, Copy, Paste, Select All, Undo, Redo) are available in the popup menu of the edited cell

The contextual menu available on every cell has the following actions:

- Set NULL - sets the content of the cell to (null). This action is disabled for columns that cannot be null.
- Insert row - inserts an empty row in the table.
- Duplicate row - makes a copy of the selected row and adds it in the Table Explorer view. You should note that the new row will not be inserted in the database table until all conflicts are solved.
- Commit row - commits the selected row.
- Delete row - deletes the selected row.
- Copy - copies the content of the cell.
- Paste - performs paste in the selected cell

Some of the above actions are also available on the Table Explorer toolbar:

- Export to XML - opens the Export Criteria dialog (a thorough description of this dialog can be found in the Import from database chapter) .
- Refresh - performs a refresh of the selected node's subtree.
- Insert row - inserts an empty row in the table.
- Duplicate row - makes a copy of the selected row and adds it in the Table Explorer view. You should note that the new row will not be inserted in the database table until all conflicts are solved.
- Commit row - commits the selected row.
- Delete row - deletes the selected row.

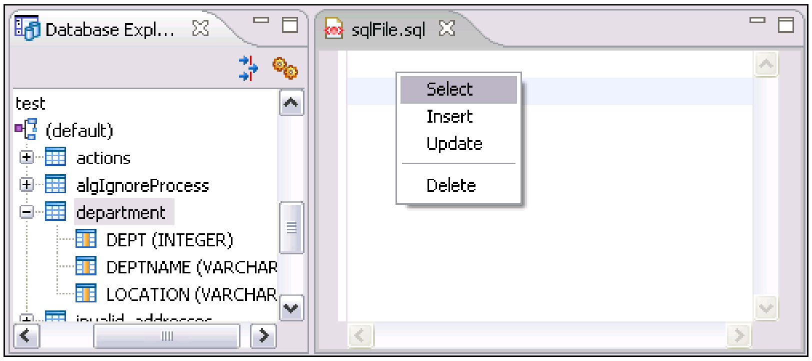
SQL Execution Support

<oXygen/>'s support for writing SQL statements includes syntax highlight, folding and drag&drop(DND) from the Database Explorer View. It also includes transformation scenarios for executing the statements and the results are displayed in the Table Explorer View.

Drag and Drop from Database Explorer

Configure a database connection as it was shown previously in this chapter and browse to the table you will use in your statement and drag it into the editor (where a sql file is open).

Figure 14.10. SQL statement editing with DND



Next, select the type of statement from the popup menu that appears in the sql editor. Depending on your choice, one of the following statements will be inserted into the document:

- SELECT `*field1*`,`*field2*`, ... FROM `*catalog*`.`*table*` (for this example: `` SELECT `DEPT`,`DEPTNAME`,`LOCATION` FROM `test`.`department` ``)

- UPDATE `*catalog*`. `*table*` SET `*field1*`=, `*field2*`=,.... (for this example: `` UPDATE `test`.`department` SET `DEPT`=, `DEPTNAME`=, `LOCATION`= ``)

- INSERT INTO`*catalog*`. `*table*` (`*field1*`,`*field2*`, ...) VALUES (, ,) (for this example: `` INSERT INTO `test`.`department` (`DEPT`,`DEPTNAME`,`LOCATION`) VALUES (, , ) ``)

- DELETE FROM `*catalog*`. `*table*` (for this example: `` DELETE FROM `test`.`department` ``)

DND is available both on the table and on its fields. Click on the column and drag it into the editor. The same popup menu as above will appear. Depending on your choice, one of the following statements will be inserted into the document:

- SELECT `*field*` FROM `*catalog*`. `*table*` (for this example: `` SELECT `DEPT` FROM `test`.`department` ``)

- UPDATE `*catalog*`. `*table*` SET `*field*`= (for this example: `` UPDATE `test`.`department` SET `DEPT`= ``)

- INSERT INTO`*catalog*`. `*table*` (`*field1*`) VALUES () (for this example: `` INSERT INTO `test`.`department` (`DEPT`) VALUES () ``)

- DELETE FROM `*catalog*`. `*table*` (for this example: `` DELETE FROM `test`.`department` WHERE `DEPT`= ``)

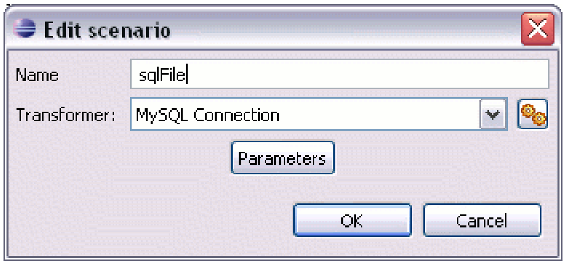
SQL Validation

Currently, SQL validation support is offered for IBM DB2. Please note that if you choose a connection that doesn't support SQL validation you will receive a warning when trying to validate. The SQL document will be validated using the connection from the associated transformation scenario.

Executing SQL Statements

First configure a transformation scenario. Click on the Configure Transformation Scenario button from the *Transformation* toolbar. The dialog that appears contains the list of existing scenarios that apply to SQL documents. To configure a new scenario, click the *New* button.

Figure 14.11. New SQL scenario dialog



Edit scenario

Name: sqlFile

Transformer: MySQL Connection

Enter a name for the scenario and choose one of the available database connections. To configure a new connection click on Configure Database Sources .

Place holders(?) for parameters are supported by <oXygen/>. For the following example `` SELECT * FROM `test`.`department` where DEPT = ? or DEPTNAME = ? `` two parameters can be configured for the transformation scenario. To do this, in the previous dialog click the Parameters button and add a new parameter for each placeholder. When the sql statement will be executed, the first placeholder will be replaced with the value set for the first parameter in the scenario, the second placeholder will be replaced by the second parameter value and so on.

The result of a SQL transformation will be displayed in the *Table Explorer* view.

To view a more complex value returned by the SQL query that cannot be displayed entirely in the query result table at the bottom of the Eclipse window, for example an XMLTYPE value or a CLOB value, you have to right click on that cell, select the action *Copy cell* from the popup menu for copying the value in the clipboard and paste the value where you need it, for example an opened XQuery editor panel of Eclipse .

Importing from Databases

This feature is explained in detail in the Import from database section of Importing Data chapter.

Creating XML Schema from Databases

This feature is explained in detail in the Convert table structure to XML section of Importing Data chapter.

Native XML Database (NXD) Support

Native XML databases have an XML-based internal model and their fundamental unit of storage is XML. <oXygen/> offers support for: Berkeley DB XML, eXist, MarkLogic, Software AG Tamino, Raining Data TigerLogic, Documentum xDb (X-Hive/DB) and Oracle XML DB.

Configuring Database Data Sources

This section presents a set of procedures describing how to configure NXD data sources.

How to configure a Berkeley DB XML datasource

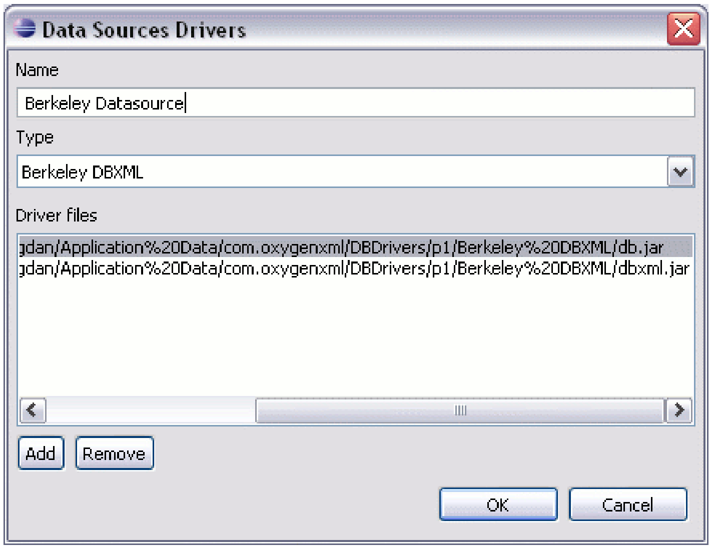
The latest instructions on how to configure Berkeley DB XML support in <oXygen/> can be found on our website [http://www.oxygenxml.com/doc/ug-oxygenEclipse/native-xml-database-support.html#configure-berkeley-datasource].

<oXygen/> supports Berkeley DB XML versions 2.3.10, 2.4.13 & 2.4.16. The following directory definitions shall apply:

- OXY_DIR - <oXygen/> installation root directory. (for example on Windows C:\Program Files\Oxygen 10.3)
- DBXML_DIR - Berkeley DB XML database root directory. (for example on Windows C:\Program Files\Sleepycat Software\Berkeley DB XML *<version>*)
- DBXML_LIBRARY_DIR (usually on Mac and Unix is DBXML_DIR/lib and on Windows is DBXML_DIR/bin)

1. Go to *Preferences* -> *Data Sources*. In the Data Sources panel click the *New* button.
2. Enter a unique name for this data source and select *Berkeley DBXML* from the driver type combo box.

   Figure 14.12. Data Source Drivers Configuration Dialog

   Data Sources Drivers

   Name

   Berkeley Datasource

   Type

   Berkeley DBXML

   Driver files

   ```
   jdan/Application%20Data/com.oxygenxml/DBDrivers/p1/Berkeley%20DBXML/db.jar
   jdan/Application%20Data/com.oxygenxml/DBDrivers/p1/Berkeley%20DBXML/dbxml.jar
   ```

3. Press the Add button to add the following Berkeley DB specific files:
   - db.jar (check for it into DBXML_DIR/lib or DBXML_DIR/jar)

- dbxml.jar (check for it into DBXML_DIR/lib or DBXML_DIR/jar)

4. Click *OK* to finish the data source configuration.

How to configure an eXist datasource

The latest instructions on how to configure eXist support in <oXygen/> can be found on our website [http://www.oxygenxml.com/doc/ug-oxygenEclipse/native-xml-database-support.html#configure-exist-datasource].

The eXist database server versions supported by <oXygen/> are 1.0, 1.1, 1.2.2, 1.2.4, 1.2.5 and 1.3.

1. Go to *Preferences* -> *Data Sources*. In the Data Sources panel click the *New* button.
2. Enter a unique name for this data source and select *eXist* from the driver type combo box.
3. Press the Add button to add the following eXist specific files which are located in the eXist installation root directory:
   - exist.jar
   - lib/core/xmldb.jar
   - lib/core/xmlrpc-client-3.1.1.jar
   - lib/core/xmlrpc-common-3.1.1.jar
   - lib/core/ws-commons-util-1.0.2.jar
4. Click *OK* to finish the data source configuration.

How to configure a MarkLogic datasource

The latest instructions on how to configure MarkLogic support in <oXygen/> can be found on our website [http://www.oxygenxml.com/doc/ug-oxygenEclipse/native-xml-database-support.html#configure-marklogic-datasource].

1. Go to *Preferences* -> *Data Sources*. In the Data Sources panel click the *New* button.
2. Enter a unique name for this data source and select *MarkLogic* from the driver type combo box.
3. Add the following MarkLogic specific file:
   - `xcc.jar`

   In the Download links for database drivers section there are listed the URLs from where to download the drivers necessary for accessing MarkLogic databases in <oXygen/>.
4. Click *OK* to finish the data source configuration.

How to configure a Software AG Tamino datasource

The latest instructions on how to configure Software AG Tamino support in <oXygen/> can be found on our website [http://www.oxygenxml.com/doc/ug-oxygenEclipse/native-xml-database-support.html#configure-tamino-datasource].

1. Go to *Preferences* -> *Data Sources*. In the Data Sources panel click the *New* button.

2. Enter a unique name for this data source and select *Tamino* from the driver type combo box.

3. Using the *Add* button add the following jar files available in the `SDK\TaminoAPI4J\lib` subdirectory of the Tamino 4.4.1 database install directory:

   - `TaminoAPI4J.jar`
   - `TaminoAPI4J-l10n.jar`
   - `TaminoJCA`.jar

   Note

   You must use the jar files from the version 4.4.1 of the Tamino database.

4. Click *OK* to finish the data source configuration.

How to configure a Raining Data TigerLogic datasource

The latest instructions on how to configure TigerLogic support in <oXygen/> can be found on our website [http://www.oxygenxml.com/doc/ug-oxygenEclipse/native-xml-database-support.html#configure-tigerlogic-datasource].

1. Go to *Preferences* -> *Data Sources*. In the Data Sources panel click the *New* button.

2. Enter a unique name for this data source and select *TigerLogic* from the driver type combo box.

3. Add the following TigerLogic specific files (found in the TigerLogic JDK lib directory from the server side):

   - `connector.jar`
   - `jca-connector.jar`
   - `tlapi.jar`
   - `tlerror.jar`
   - `utility.jar`
   - `xmlparser.jar`
   - `xmltypes.jar`

4. Click *OK* to finish the data source configuration.

How to configure a Documentum xDb (X-Hive/DB) datasource

The latest instructions on how to configure Documentum xDb (X-Hive/DB) support in <oXygen/> can be found on our website [http://www.oxygenxml.com/doc/ug-oxygenEclipse/native-xml-database-support.html#configure-xhive-datasource].

1. Go to *Preferences* -> *Data Sources*. In the Data Sources panel click the *New* button.

2. Enter a unique name for this data source and select *Documentum xDb (X-Hive/DB)* from the driver type combo box.

3. Add the following Documentum xDb (X-Hive/DB) specific files (found in the Documentum xDb (X-Hive/DB) lib directory from the server side):

   - `antlr-runtime-3.0.1.jar`
   - `icu4j.jar`
   - `xhive.jar`

   If you like to use a bootstrap file when connecting to the database you need to additionally add the following JAR files found in the same place:

   - `fop.jar`
   - `jsr173_api.jar`
   - `lucene.jar`
   - `mx4j.jar`
   - `serializer.jar`
   - `w3c.jar`
   - `xalan.jar`
   - `xbean.jar`
   - `xercesImpl.jar`
   - `xml-apis.jar`

4. Click *OK* to finish the data source configuration.

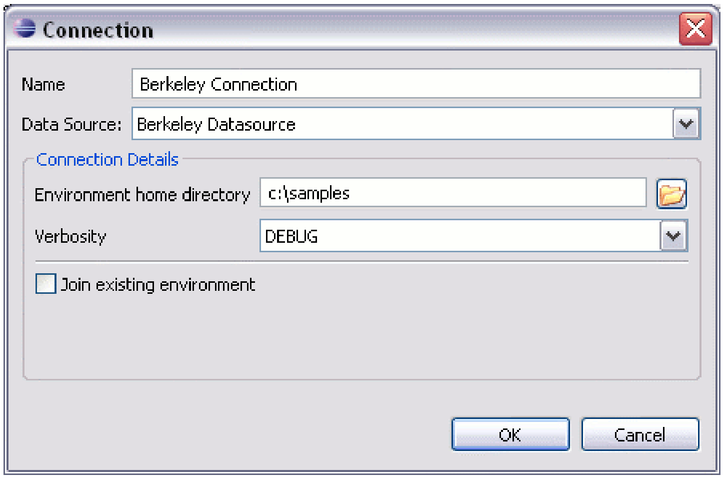
Configuring Database Connections

This section presents a set of procedures describing how to configure connections that use Native XML Database data sources.

How to configure a Berkeley DB XML Connection

<oXygen/> supports Berkeley DB XML versions 2.3.10, 2.4.13 & 2.4.16.

Figure 14.13. The Connection Configuration Dialog



Name: Berkeley Connection

Data Source: Berkeley Datasource

Environment home directory: c:\samples

Verbosity: DEBUG

□ Join existing environment

1. Go to *Preferences* -> *Data Sources*. In the Connections panel click the *New* button.

2. Enter a unique name for this connection and select one of the previously configured Berkeley data sources from the Data Source combo box.

3. Fill-in the Connection Details:

   Environment home directory — Path to the Berkeley DB XML's home directory.

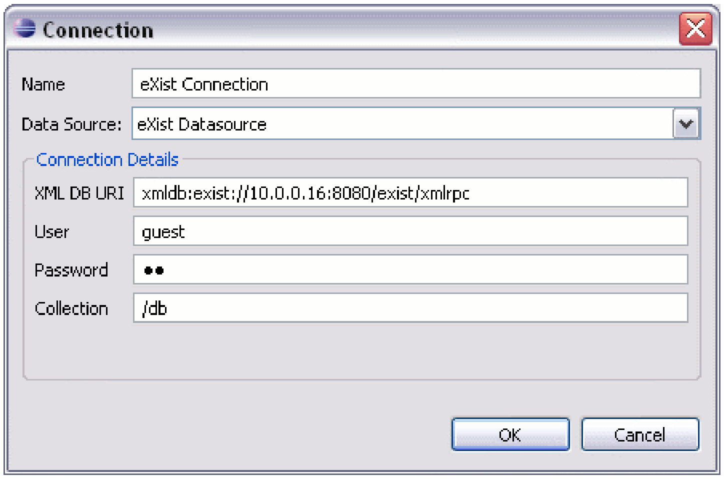
   Verbosity — The user can choose between four levels of verbosity: DEBUG, INFO, WARNING, ERROR.

   Join existing environment — If checked, an attempt will be made to join an existing environment in the specified home directory and all the original environment settings will be preserved. If that fails, you should consider reconfiguring the connection with this option unchecked.

4. Click *OK*.

How to configure an eXist Connection

Figure 14.14. The Connection Configuration Dialog



Name: eXist Connection
Data Source: eXist Datasource
XML DB URI: xmldb:exist://10.0.0.16:8080/exist/xmlrpc
User: guest
Password: ••
Collection: /db

1. Go to *Preferences* -> *Data Sources*. In the Connections panel click the *New* button.

2. Enter a unique name for this connection and select one of the previously configured eXist data sources from the Data Source combo box.

3. Fill-in the Connection Details

   XML DB URI — URI to the installed eXist engine.

   User — User name to access the eXist database engine.

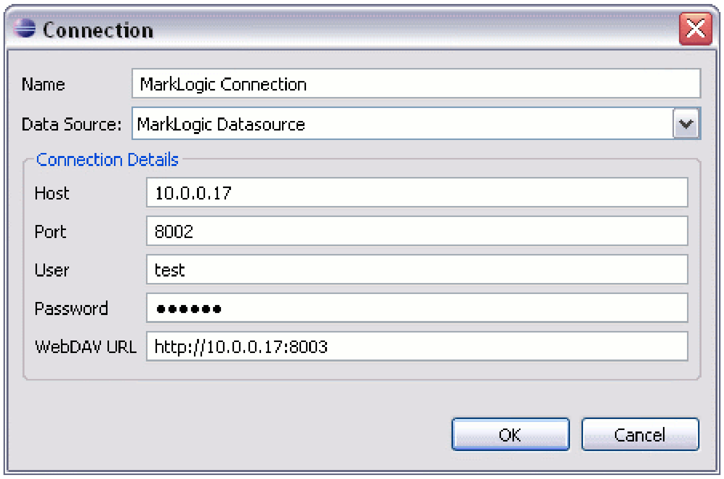
   Password — Password to access the eXist database engine.

   Collection — eXist organizes all documents in hierarchical collections. Collections are like directories. They are used to group related documents together. This text field allows the user to set the default collection name.

4. Click *OK*.

How to configure a MarkLogic Connection

Figure 14.15. The Connection Configuration Dialog



Name: MarkLogic Connection

Data Source: MarkLogic Datasource

Host: 10.0.0.17

Port: 8002

User: test

Password: ••••••

WebDAV URL: http://10.0.0.17:8003

1. Go to *Preferences* -> *Data Sources*. In the Connections panel click the *New* button.

2. Enter a unique name for this connection and select one of the previously configured MarkLogic data sources from the Data Source combo box.

3. Fill-in the Connection Details:

   XDBC Host — The host name or ip address of the installed MarkLogic engine.

   Port — The port number of the MarkLogic engine.

   User — User name to access the MarkLogic engine.

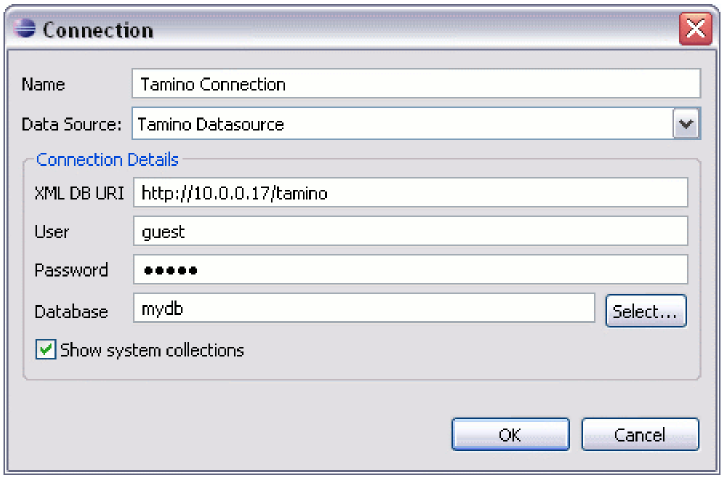
   Password — Password to access the MarkLogic engine.

   WebDAV URL — The url used for browsing the MarkLogic database in the Database Explorer view. (optional)

4. Click *OK*.

How to configure a Software AG Tamino Connection

Figure 14.16. The Connection Configuration Dialog



Name: Tamino Connection

Data Source: Tamino Datasource

XML DB URI: http://10.0.0.17/tamino

User: guest

Password: ●●●●●

Database: mydb

☑ Show system collections

1. Go to *Preferences -> Data Sources*. In the Connections panel click the *New* button.

2. Enter a unique name for this connection and select one of the previously configured Tamino data sources from the Data Source combo box.

3. Fill-in the Connection Details:

   XML DB URI
   URI to the installed Tamino engine

   User
   User name to access the Tamino database engine

   Password
   Password to access the Tamino database engine

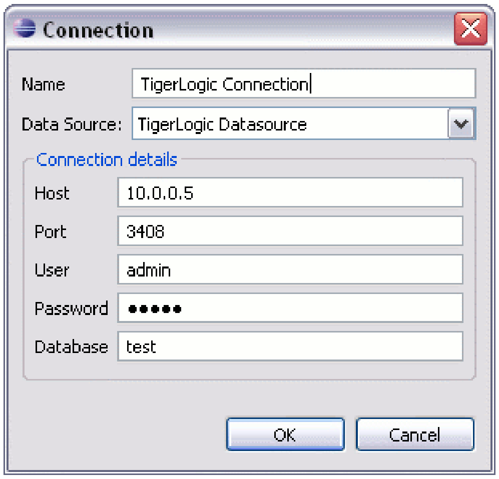
   Database
   The name of the database to access from the Tamino database engine. Choose the Select button to display all databases on the specified server in an additional dialog box. You can then choose the desired database. This feature works only with databases that have been created starting with version 4.2.1. In all other cases, a message appears saying that a list of databases is not available.

   Show system collections
   Check this if you want to see the Tamino system collections in the Database Explorer.

4. Click *OK*.

How to configure a Raining Data TigerLogic Connection

Figure 14.17. The Connection Configuration Dialog



Name TigerLogic Connection
Data Source: TigerLogic Datasource
Connection details
Host 10.0.0.5
Port 3408
User admin
Password •••••
Database test

1. Go to *Preferences -> Data Sources*. In the Connections panel click the *New* button.

2. Enter a unique name for this connection and select one of the previously configured TigerLogic data sources from the Data Source combo box.

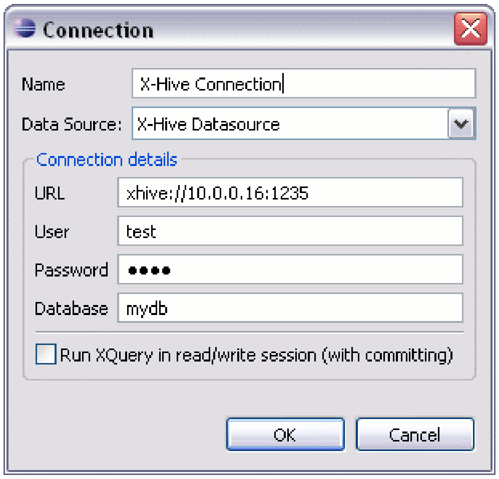
3. Fill-in the Connection Details:

Host	The host name or ip address of the installed TigerLogic engine.
Port	The port number of the TigerLogic engine.
User	User name to access the TigerLogic engine.
Password	Password to access the TigerLogic engine.
Database	The name of the database to access from the TigerLogic engine.

4. Click *OK*.

How to configure an Documentum xDb (X-Hive/DB) Connection

Figure 14.18. The Configure Connection Dialog



Name: X-Hive Connection

Data Source: X-Hive Datasource

URL: xhive://10.0.0.16:1235

User: test

Password: ●●●●

Database: mydb

□ Run XQuery in read/write session (with committing)

1. Go to *Preferences* -> *Data Sources*. In the Connections panel click the *New* button.

2. Enter a unique name for this connection and select one of the previously configured Documentum xDb (X-Hive/DB) data sources from the Data Source combo box.

3. Fill-in the Connection Details:

   URL — The URL property for Documentum xDb (X-Hive/DB) connection. It is also called xhive.bootstrap and specifies the location of the Documentum xDb (X-Hive/DB) federation.

   This property can be used in two different ways:

   If the property is a URL of the form xhive://host:port, the Documentum xDb (X-Hive/DB) connection will attempt to connect to an Documentum xDb (X-Hive/DB) server running behind the specified TCP/IP port.

   If the property is the complete (or relative) path to an `XhiveDatabase.bootstrap` file, an Documentum xDb (X-Hive/DB) server will be started in the current JVM. Depending on the application, this can be much faster than using a remote server because the communication overhead is avoided. However, only one JVM can run an Documentum xDb (X-Hive/DB) server for a specific federation at the same time.

   For the second case (using a bootstrap file to connect) you need to add additional JAR files when you configure the data source.

   User — User name to access the Documentum xDb (X-Hive/DB) database engine.

   Password — Password to access the Documentum xDb (X-Hive/DB) database engine.

   Database — The name of the database to access from the Documentum xDb (X-Hive/DB) database engine.

Run XQuery in read/write session (with committing)	If checked the Documentum xDb (X-Hive/DB) session ends with a commit, otherwise it ends with a rollback.

4. Click *OK*.

Resource Management

Database Explorer View

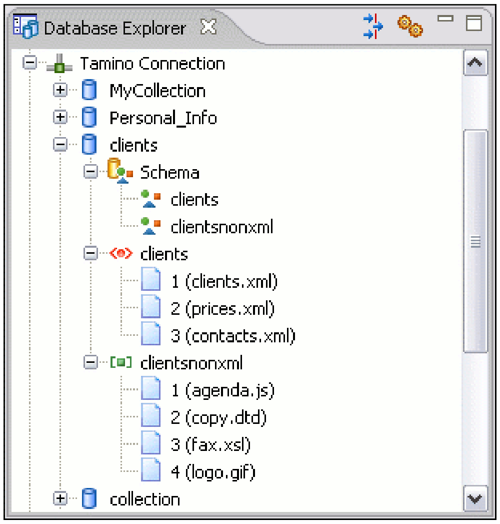
This view presents in a tree-like fashion the database connections configured in *Preferences* -> *Data Sources*. You can connect to a database simply by expanding the connection node. The database structure can be expanded up to column level. <oXygen/> supports multiple simultaneous database connections and the connections tree provides an easy way to browse them.

Some of the basic components employed by the XML:DB API are collections and resources, and they appear in the tree sorted in alphabetical order.

A collection is a hierarchical container for resources and further sub-collections.

There are two types of resources: XML resource and non XML resource . An *XML resource* represents an xml document or a document fragment, selected by a previously executed XPath query.

Figure 14.19. The Database Explorer View



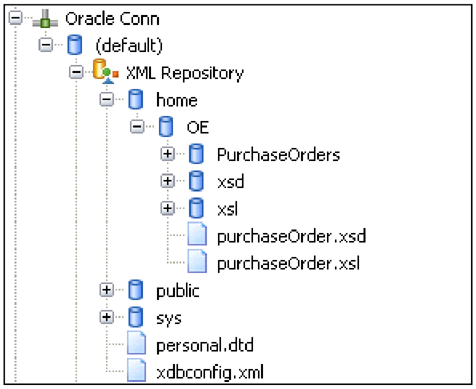
Below you can find a description of the contextual menu actions available on the Database Explorer levels (explained for each connection). Please note that you can open in the editor a resource or a schema component by double-clicking it.

Oracle XML DB Browser

Oracle XML DB is a feature of the Oracle Database. It provides a high-performance, native XML storage and retrieval technology.

<oXygen/> allows the user to browse the native Oracle XML Repository and perform various operations on the resources in the repository.

Figure 14.20. Browsing the Oracle XML DB Repository



Actions available at XML Repository level

- Refresh - performs a refresh of the XML Repository.
- Add container - add a new child container to the XML Repository
- Add resource - adds a new resource to the XML Repository.

Actions available at container level

- Refresh - performs a refresh of the selected container.
- Add container - add a new child container to the current one
- Add resource - adds a new resource to the folder.
- Delete - delete the current container.
- Properties - shows various properties of the current container.

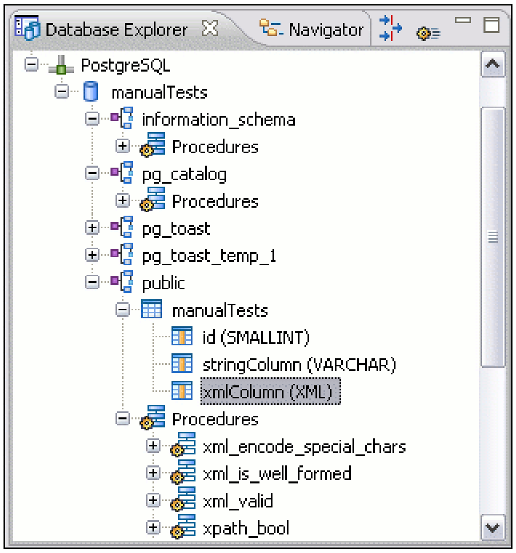
Actions available at resource level

- Refresh - performs a refresh of the selected resource.
- Open - opens the selected resource in the editor.
- Rename - rename the current resource.
- Move - move the current resource to a new container.
- Delete - delete the current resource.
- Copy location - allows you to copy to clipboard an application specific URL for the resource which can then be used for various actions like opening or transforming the resources.
- Properties - shows various properties of the current resource.

PostgreSQL connection

<oXygen/> allows the user to browse the structure of the PostgreSQL database in the *Database Explorer* view and open the tables in the *Table Explorer* view.

Figure 14.21. Browsing a PostgreSQL repository



Actions available at container level

- Refresh - performs a refresh of the selected container.

Actions available at resource level

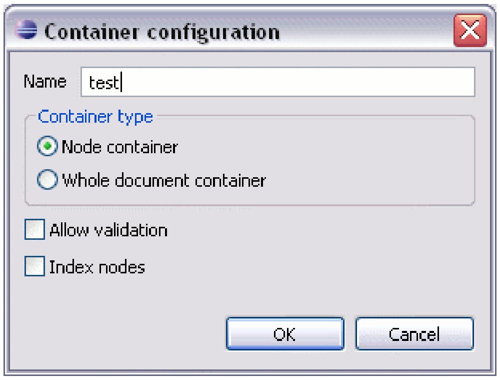
- Refresh - performs a refresh of the selected database table.
- Edit - opens the selected database table in the *Table Explorer* view.
- Export to XML ... - export the content of the selected database table as an XML file using the dialog from importing data from a database.

Berkeley DB XML Connection

Actions available at connection level

- Refresh - performs a refresh of the selected node's subtree.
- Configure Database Sources - opens the *Data Sources* preferences page where you can configure both data sources and connections.
- Add container - allows adding a new container.

Figure 14.22. Add Container Dialog



Container configuration

Name: test

Container type

● Node container

○ Whole document container

□ Allow validation

□ Index nodes

Name

The name of the new container.

Container type

At creation time, every container must have a type defined for it. This container type identifies how XML documents are stored in the container. As such, the container type can only be determined at container creation time; you cannot change it on subsequent container opens.

Containers can have one of the following types specified for them:

Node container

Xml documents are stored as individual nodes in the container. That is, each record in the underlying database contains a single leaf node, its attributes and attribute values if any, and its text nodes, if any. BDB XML also keeps the information it needs to reassemble the document from the individual nodes stored in the underlying databases. This is the default, and preferred, container type.

Whole document container

The container contains entire documents; the documents are stored without any manipulation of line breaks or whitespace.

Allow validation

If checked it causes documents to be validated when they are loaded into the container. The default behavior is to not validate documents.

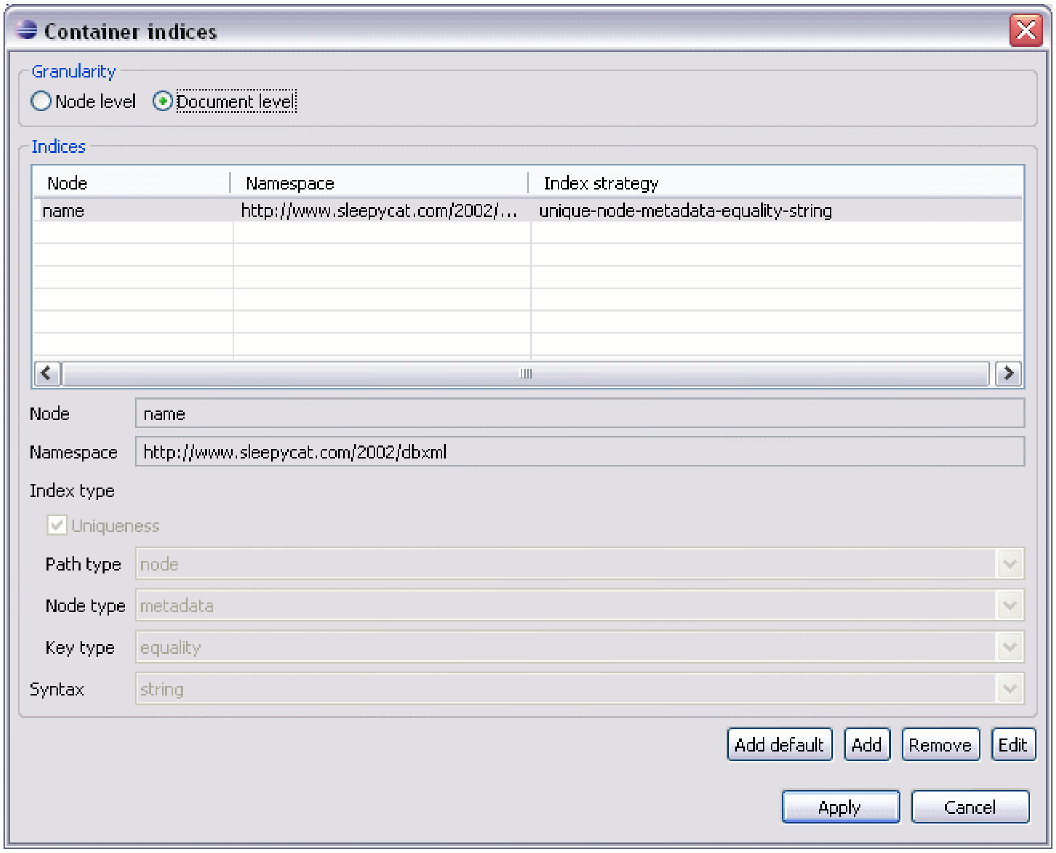
Index nodes

If checked it causes indices for the container to return nodes rather than documents. The default is to index at the document level. This property has no meaning if the container type is whole document container.

Actions available at container level

- Refresh - performs a refresh of the selected node's subtree.
- Add Resource - adds a new XML resource to the selected container.
- Rename - allows you to specify a new name for the selected container.
- Delete - removes the selected container from the database tree.
- Edit indices - allows you to edit the indices for the selected container.

Figure 14.23. Container indices



Container indices

Granularity

○ Node level ● Document level

Indices

Node	Namespace	Index strategy
name	http://www.sleepycat.com/2002/...	unique-node-metadata-equality-string

Node: name

Namespace: http://www.sleepycat.com/2002/dbxml

Index type

☑ Uniqueness

Path type: node

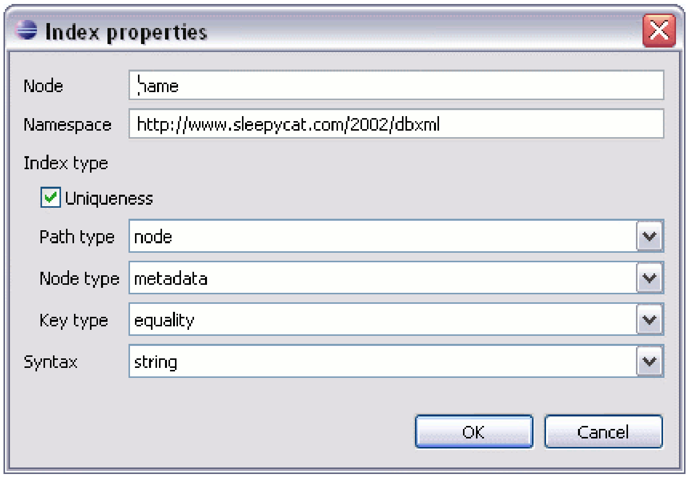
Node type: metadata

Key type: equality

Syntax: string

- Specifying the granularity:
  - Document granularity is good for retrieving large documents
  - Node granularity is good for retrieving nodes from within documents
- Adding/editing indices:

Figure 14.24. Adding/editing indices



Index properties

Node: name

Namespace: http://www.sleepycat.com/2002/dbxml

Index type

☑ Uniqueness

Path type: node

Node type: metadata

Key type: equality

Syntax: string

- Node - the node name
- Namespace - the index namespace
- Index strategy:
    - Index type:
        - Uniqueness - indicates whether the indexed value must be unique within the container
        - Path type:
            - node - indicates that you want to index a single node in the path
            - edge - indicates that you want to index the portion of the path where two nodes meet
        - Node type:
            - element - an element node in the document content
            - attribute - an attribute node in the document content
            - metadata - a node found only in a document's metadata content.
        - Key type:
            - equality - improves the performances of tests that look for nodes with a specific value
            - presence - improves the performances of tests that look for the existence of a node regardless of its value
            - substring - improves the performance of tests that look for a node whose value contains a given substring
    - Syntax types - the syntax describes what sort of data the index will contain and is mostly used to determine how indexed values are compared

Actions available at resource level

- Refresh - performs a refresh of the selected resource.
- Open - opens the selected resource in the editor.
- Rename - allows you to change the name of the selected resource.
- Move - allows you to move the selected resource in a different container in the database tree.
- Delete - removes the selected resource from the container.
- Copy location - allows you to copy to clipboard an application specific URL for the resource which can then be used for various actions like opening or transforming the resources.

eXist Connection

Actions available at connection level

- Refresh - performs a refresh of the selected node's subtree.

- Configure Database Sources - opens the *Data Sources* preferences page where you can configure both data sources and connections.

Actions available at container level

- Refresh - performs a refresh of the selected node's subtree.
- Add Resource - adds a new XML resource to the selected container.
- Add Container - creates a new collection in the selected one.
- Delete - removes the selected collection.
- Rename - allows you to change the name of the selected collection.
- Move - allows you to move the selected collection in a different location in the database tree.

Actions available at resource level

- Refresh - performs a refresh of the selected resource.
- Open - opens the selected resource in the editor.
- Rename - allows you to change the name of the selected resource.
- Move - allows you to move the selected resource in a different collection in the database tree.
- Delete - removes the selected resource from the collection.
- Copy location - allows you to copy to clipboard an application specific URL for the resource which can then be used for various actions like opening or transforming the resources.
- Properties - allows the user to view various useful properties associated with the resource.
- Save As - allows you to save the name of the selected binary resource as a file on disk.

MarkLogic Connection

Resource management for MarkLogic database ca be done through WebDAV. For this the WebDAV url must be configured in the MarkLogic connection. The actions that can be performed on MarkLogic resources through WebDAV are the same used for a WebDAV connection (see more about this in WebDAV Connection section).

Note

The interaction with the database is also made using XQuery (more on this topic can be found in the XQuery section) .

Software AG Tamino Connection

Actions available at connection level

- Refresh - performs a refresh of the selected node's subtree.

- Configure Database Sources - opens the *Data Sources* preferences page where you can configure both data sources and connections.
- Add container - allows you to create a new collection in the database.

Actions available at collection level

For every new Tamino collection, you can specify if a schema is *required*, *optional* or *prohibited*. The following actions are available:

- Refresh - performs a refresh of the selected node's subtree.
- Filter ... - An XQuery expression can be specified for filtering the nodes displayed in the selected Tamino container. It is only possible to specify one predicate. In the XQuery syntax a predicate is enclosed in square brackets. The square brackets, however, must not be specified in the dialog box displayed by this action. Only the predicate must be specified and it will be applied on the selected doctype. For example:

  ```
  name/surname between 'B', 'C'
  ```

- Insert XML instance - allows you to load a new XML document.
- Insert non XML instance - allows you to load a non XML document.
- Modify Collection Properties - allows you to change the schema usage for the selected collection to optional. This action is available on collections with required and prohibited schema usage.
- Define schema - allows you to add a new schema in the Schema Repository. This action is available on collections with optional and required schema usage.
- Delete - removes the selected collection. If it is a Tamino doctype then the action removes all the XML instances contained in the doctype.
- Set default - Sets this collection as the default collection for running queries with the input() function.

Actions available at schema level

- Refresh - performs a refresh of the selected schema.
- Open - opens the selected schema in the editor. There are supported schema changes that preserve the validity relative to the existent instances.
- Delete - removes the selected schema from the Schema Repository.

Actions available at resource level

- Refresh - performs a refresh of the selected resource.
- Open - opens the selected resource in the editor.
- Rename - allows you to change the name of the selected resource.
- Delete - removes the selected resource.

- Copy location - allows you to copy to clipboard an application specific URL for the resource which can then be used for various actions like opening or transforming the resources.
- Properties - allows the user to view various useful properties associated with the resource.
- Save As - allows you to save the name of the selected binary resource as a file on disk.

Validation of an XML resource stored in a Tamino database is done against the schema associated with the resource in the database.

Note

<oXygen/> also displays the contents of the WebDAV enabled collection `ino:dav`. The actions that can be performed on Tamino resources through WebDAV are the same used for a WebDAV connection (see more about this in WebDAV Connection section).

Raining Data TigerLogic Connection

Note

Resource management is unavailable (no browsing support is offered). The interaction with the database is made using XQuery (more on this topic can be found in the XQuery section) .

Documentum xDb (X-Hive/DB) Connection

Actions available at connection level

- Refresh - performs a refresh of the selected node's subtree.
- Configure Database Sources - opens the *Data Sources* preferences page where you can configure both data sources and connections.
- Add library - allows you to add a new library.
- Insert XML Instance - allows you to add a new xml resource directly into the database root. See Documentum xDb (X-Hive/DB) Parser Configuration for more details.
- Insert non XML Instance - allows you to add a new non xml resource directly into the database root.
- Properties - displays the connection properties.

Actions available at catalog level

- Refresh - performs a refresh of the selected catalog.
- Add AS models - allows you to add a new abstract schema model to the selected catalog.
- Set default schema - allows you to set a default DTD to be used for parsing. It is not possible to set a default XML Schema.
- Clear default schema - allows you to clear the default DTD. The action is available only if there is a DTD set as default.
- Properties - displays the catalog properties.

Actions available at schema resource level

- Refresh - performs a refresh of the selected schema resource.
- Open - opens the selected schema resource in the editor.
- Rename - allows you to change the name of the selected schema resource.
- Save As - allows you to save the selected schema resource as a file on disk.
- Delete - removes the selected schema resource from the catalog
- Copy location - allows you to copy to clipboard the URL of the selected schema resource.
- Set default schema - allows you to set the selected DTD to be used as default for parsing. The action is available only for DTD.
- Clear default schema - allows you to unset the selected DTD. The action is available only if the selected DTD is the current default to be used for parsing.

Actions available at library level

- Refresh - performs a refresh of the selected library.
- Add library - adds a new library as child of the selected library.
- Add local catalog - adds a catalog to the selected library. By default, only the root-library has a catalog, and all models would be stored there.
- Insert XML Instance - allows you to add a new xml resource to the selected library. See Documentum xDb (X-Hive/DB) Parser Configuration for more details.
- Insert non XML Instance - allows you to add a new non xml resource to the selected library.
- Rename - allows you to specify a new name for the selected library.
- Move - allows you to move the selected library to a different one.
- Delete - removes the selected library.
- Properties - displays the library properties.

Actions available at resource level

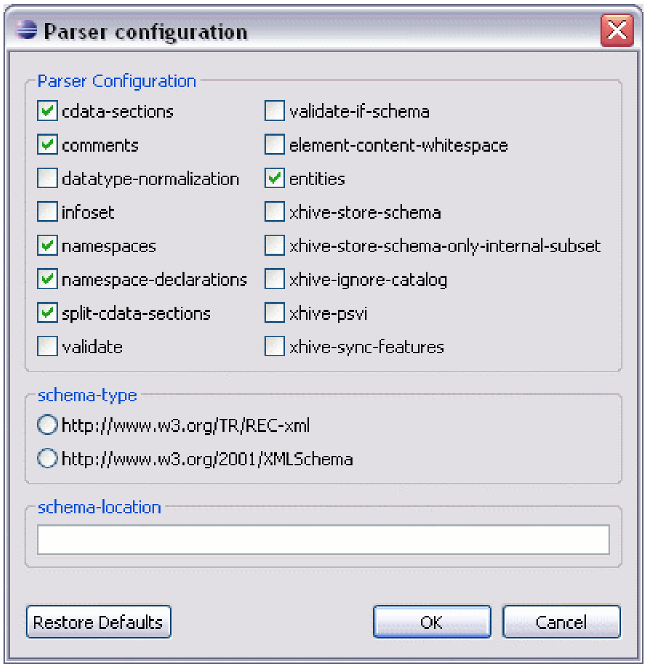
- Refresh - performs a refresh of the selected resource.
- Open - opens the selected resource in the editor.
- Rename - allows you to change the name of the selected resource.
- Move - allows you to move the selected resource in a different library in the database tree.
- Save As - allows you to save the selected binary resource as a file on disk.
- Delete - removes the selected resource from the library.

- Copy location - allows you to copy to clipboard the URL of the selected resource.
- Add AS model - allows you to add an XML schema to the selected XML resource.
- Set AS model - allows you to set an active AS model for the selected XML resource.
- Clear AS model - allows you to clear the active AS model of the selected XML resource.
- Properties - displays the resource properties. Available only for XML resources.

Validation of an XML resource stored in an Documentum xDb (X-Hive/DB) database is done against the schema associated with the resource in the database.

Documentum xDb (X-Hive/DB) parser configuration for adding XML instances

Figure 14.25. Parser configuration



Parser Configuration

☑ cdata-sections	☐ validate-if-schema
☑ comments	☐ element-content-whitespace
☐ datatype-normalization	☑ entities
☐ infoset	☐ xhive-store-schema
☑ namespaces	☐ xhive-store-schema-only-internal-subset
☑ namespace-declarations	☐ xhive-ignore-catalog
☑ split-cdata-sections	☐ xhive-psvi
☐ validate	☐ xhive-sync-features

schema-type

○ http://www.w3.org/TR/REC-xml

○ http://www.w3.org/2001/XMLSchema

schema-location

- DOM Level 3 parser configuration parameters. More about each parameter can be found here: DOM Level 3 Configuration [http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#DOMConfiguration]
- Documentum xDb (X-Hive/DB) specific parser parameters (for more information please consult the Documentum xDb (X-Hive/DB) manual):
  - xhive-store-schema - During validated parsing, the corresponding DTD's or XML schemas are or are not stored in the catalog.
  - xhive-store-schema-only-internal-subset - Store only the internal subset of the document (not any external subset). Modifier for xhive-store-schema (only has a function when that parameter is set to true, and when DTDs are involved). Use this option if you only want to store the internal subset of the document (not the external subset).
  - xhive-ignore-catalog - During validated parsing, the corresponding DTD's and XML schemas in the catalog are ignored.

- `xhive-psvi` - Store psvi information on elements and attributes. Documents parsed with this feature turned on, give access to psvi information and enable support of data types by XQuery queries.
- xhive-sync-features - Convenience setting. With this setting turned on, parameter settings of XhiveDocumentIf are synchronized with the parameter settings of LSParser. Note that parameter settings "xhive-psvi" and "schema-location" are always synchronized.

XQuery and Databases

XQuery is a native XML query language and it can be used to query XML views of relational data to create XML results. It provides the mechanism to efficiently and easily extract information from Native XML Databases (NXD) and relational data as well. The following database systems offer XQuery support:

- *Native XML Databases:*
  - Berkeley DB XML
  - eXist
  - MarkLogic (validation support not available)
  - Software AG Tamino
  - Raining Data TigerLogic (validation support not available)
  - Documentum xDb (X-Hive/DB)
- *Relational Databases:*
  - IBM DB2
  - Microsoft SQL Server (validation support not available)
  - Oracle (validation support not available)

Drag and Drop from Database Explorer

You can use <oXygen/>'s DND support when you are querying relational databases. Configure the relational data source and the database connection (as it was previously shown in this chapter), browse the connection up to table or column level and drag it in the editor (where an xquery file is open). An XPath expression of the selection will be inserted in the xquery document (at caret position).

XQuery validation

Please note that if you choose a processor that doesn't support XQuery validation you will receive a warning when trying to validate.

Note

If there is no transformation scenario associated with the current document, the validation will be performed using the processor or connection specified in the *XML / XSLT - FO / XQuery* Preferences page. Otherwise, the xquery document will be validated using the Transformer from the associated scenario.

XQuery transformation

Data is stored in relational databases but often it is required that data is extracted and transformed as XML when interfacing to other components and services Also, it is an XPath-based querying language supported by most NXD vendors. XQuery is designed to retrieve and interpret XML data from any source, whether it is a database or a document.

To perform a query database you will first need to configure a data source and a connection (details can be found in the Relational Database Support and Native XML Database Support sections).

Next, configure a transformation scenario and associate it with your XQuery document:

1. Open the Configure Transformation Scenario dialog.
2. Click the *New* button.
3. In the *Edit Scenario* dialog insert the scenario's name. Then, from the list of available *Transformer*s choose the database connection you need. Configure any other parameters if necessary.
4. Click *OK* to finish editing the scenario.

For an XQuery transformation the output tab has an option called *Sequence* which allows you to execute an XQuery in lazy mode. The amount of data extract from the database is control from option Size limit on Sequence view. If you choose *Perform FO Processing* in the *FO Processor* tab, *Sequence* option is ignored.

Once the scenario is associated with the XQuery file, depending on the target database engine the query can include calls to specific XQuery functions implemented by that engine. For example for the eXist and Berkeley DB engine the content completion assistant lists the functions supported by that database engine. This is useful for inserting in the query only calls to the supported functions (standard XQuery functions or extension ones).

To query the database, apply the transformation scenario associated with your XQuery document. To view a more complex value returned by the query that cannot be displayed entirely in the XQuery query result table at the bottom of the Eclipse window, for example an XMLTYPE value or a CLOB value, you have to right click on that cell, select the action *Copy cell* from the popup menu for copying the value in the clipboard and paste the value where you need it, for example an opened XQuery editor panel of Eclipse .

XQuery database debugging

XQuery debugging is currently supported only for the MarkLogic database engine.

To start a debug session against the MarkLogic engine you will first need to configure a MarkLogic datasource and a MarkLogic connection. Also you have to make sure that the debugging support is enabled in the MarkLogic server that will be accessed from <oXygen/>. On the server side debugging must be activated both in the XDBC server and in the section *Task Server* of the server control console (the switch *debug allow*) otherwise the error DBG-TASKDEBUGALLOW is reported by the MarkLogic server.

The MarkLogic XQuery debugger integrates seamlessly into the XQuery Debugger perspective. If you already have a MarkLogic scenario configured for the XQuery file you can choose directly to debug the scenario. If not, you just have to switch to the XQuery Debugger perspective, open the XQuery file in the editor and select the MarkLogic connection in the XQuery engine selector from the debug control toolbar. For general information about how a debugging session is started and controlled see the working with the debugger section.

When debugging queries which import modules the recommended steps are as follows:

- After starting the debugging session 'Step in' repeatedly until reaching the desired modules

- Add each of the modules to the project for easy access
- Set breakpoints in the modules as needed
- Debug the query as you see fit
- When starting a new debugging session make sure that the modules which you will debug are already opened in the editor. This is necessary so that the breakpoints in modules will be considered. Also make sure there are no other opened modules which are not involved in the current debugging session

Peculiarities and limitations of the MarkLogic debugger integration:

- Debugging support is available only for MarkLogic server versions 3.2 or newer.
- For MarkLogic server versions 4.0 or newer there are three XQuery syntaxes which are supported: '0.9-ml' (inherited from MarkLogic 3.2), '1.0-ml' and '1.0'
- All the debugging steps are executed by the MarkLogic server and the results or possible errors of each step are presented by the local debugger user interface.
- All declared variables are presented as strings.
- No support for Output to Source Mapping.
- No support for evaluating break conditions.
- No support for showing the trace.
- Breakpoints can be set in the imported modules but they are only active if the modules are opened in the editor at the time of debugging.
- Break conditions are not supported hence the Break Conditions view is disabled in the XQuery Debugger perspective.
- The modules can only be opened in the editor during the debugging session by stepping in repeatedly until reaching the module.
- There should not be any breakpoints set in modules from the same server which are not involved in the current debugging session.
- No support for profiling when an XQuery transformation is executed in the debugger.

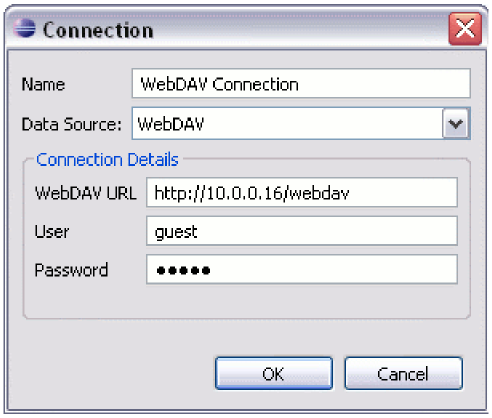
WebDAV Connection

This section presents the procedure used to configure a WebDAV connection in the Database Explorer.

<oXygen/>'s default configuration already contains a WebDAV data source called *WebDAV*.

How to Configure a WebDAV Connection

Figure 14.26. The WebDAV Connection Configuration Dialog



Name WebDAV Connection

Data Source: WebDAV

WebDAV URL http://10.0.0.16/webdav

User guest

Password •••••

1. Go to *Preferences* -> *Data Sources*. In the Connections panel click the *New* button.

2. Enter a unique name for this connection and select one of the *WebDAV* data source from the Data Source combo box.

3. Fill-in the Connection Details:

   WebDAV URL — URL to the WebDAV repository.

   User — User name to access the WebDAV repository.

   Password — Password to access the WebDAV repository.

4. Click *OK*.

WebDAV connection actions

Actions available at connection level

- Refresh - performs a refresh of the connection.
- Configure Database Sources - opens the *Data Sources* preferences page where you can configure both data sources and connections.
- Add container - allows you to create a new folder.
- Add Resource - allows you to add a new file on the server.

Actions available at folder level

- Refresh - performs a refresh of the selected node's subtree.
- Add container - allows you to create a new folder.

- Add Resource - allows you to add a new file on the server in the current folder.
- Rename - allows you to change the name of the selected folder.
- Move - allows you to move the selected folder in a different location in the tree.
- Delete - removes the selected folder.

Actions available at file level

- Refresh - performs a refresh of the selected node.
- Open - allows you to open the selected file in the editor.
- Rename - allows you to change the name of the selected file.
- Move - allows you to move the selected file in a different location in the tree.
- Delete - removes the selected file.
- Copy location - allows you to copy to clipboard an application specific URL for the resource which can then be used for various actions like opening or transforming the resources.

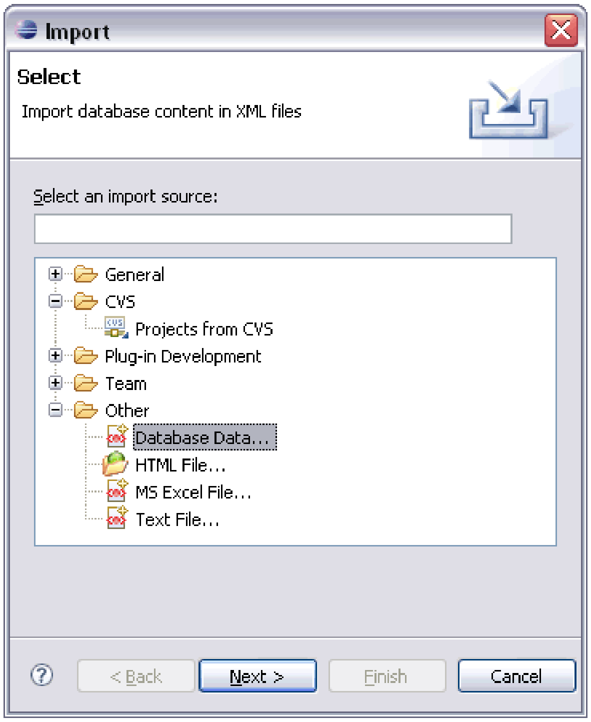
Chapter 15. Importing data

Introduction

XML was designed to describe data. Computer systems and databases contain data in incompatible formats and one of the most time-consuming activities has been to exchange data between these systems. Converting the data to XML can greatly reduce complexity and create data that can be read by many different types of applications.

This is why <oXygen/> now offers you support for importing text files, MS Excel files, Database Data and HTML files into XML documents, that can be further converted into other formats using the Transform features.

Figure 15.1. The Import wizards of <oXygen/> plugin



Import

Select

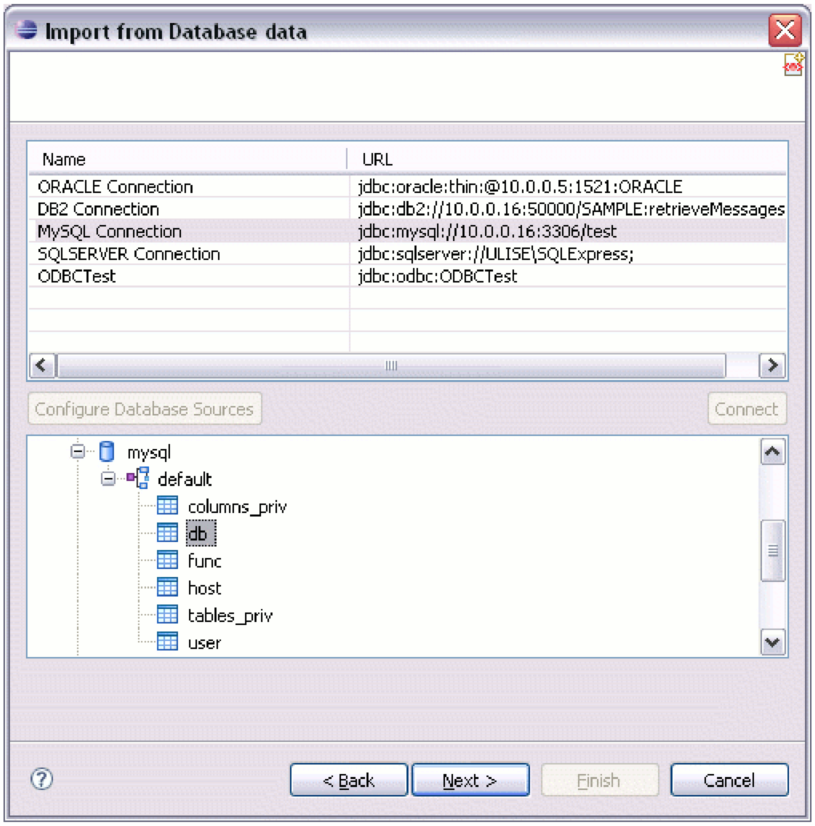
Import database content in XML files

Import from database

Import table content as XML document

To import from a database, select File → Import → Database data Next, in the "Import from Database data" wizard choose the connection you want to use. You can edit, delete or add a new data source and connection: click on the "Configure Database Sources" button and the "Preferences" dialog will open at Data Sources section. Click Connect.

Figure 15.2. Import from Database data Wizard

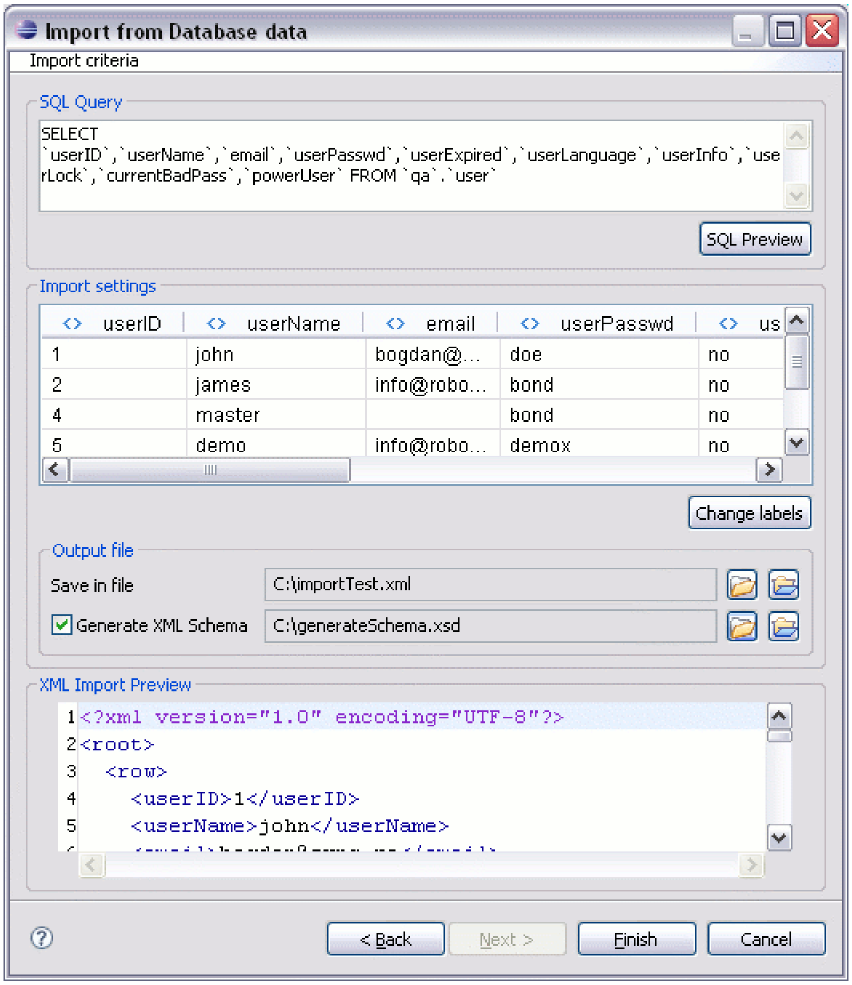


Name	URL
ORACLE Connection	jdbc:oracle:thin:@10.0.0.5:1521:ORACLE
DB2 Connection	jdbc:db2://10.0.0.16:50000/SAMPLE:retrieveMessages
MySQL Connection	jdbc:mysql://10.0.0.16:3306/test
SQLSERVER Connection	jdbc:sqlserver://ULISE\SQLExpress;
ODBCTest	jdbc:odbc:ODBCTest

From the catalogs list click on a schema and choose the required table. Click Ok.

The "Import criteria" Dialog will open next, showing a default Query string like "select * from table" in SQL Query. You can click the "SQL Preview" button to see the input data displayed in a tabular form and the XML Import Preview containing an example of what the generated XML will look like. The SQL Query message is editable. You can specify which fields should be taken into consideration.

Figure 15.3. Import from Database Criteria Dialog



```
SELECT
`userID`,`userName`,`email`,`userPasswd`,`userExpired`,`userLanguage`,`userInfo`,`use
rLock`,`currentBadPass`,`powerUser` FROM `qa`.`user`
```

<> userID	<> userName	<> email	<> userPasswd	<> us
1	john	bogdan@...	doe	no
2	james	info@robo...	bond	no
4	master		bond	no
5	demo	info@robo...	demox	no

Save in file: C:\importTest.xml

☑ Generate XML Schema: C:\generateSchema.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <row>
    <userID>1</userID>
    <userName>john</userName>
```

If you edit the query string so that the query does a join of two or more tables and selects columns with the same name from different tables you should use an alias for the columns like in the following example. That will avoid a confusion of two columns mapped to the same name in the result document of the importing operation.

```
select s.subcat_id,
      s.nr as s_nr,
      s.name,
      q.q_id,
      q.nr as q_nr,
      q.q_text
  from faq.subcategory s,
          faq.question q
  where  ...
```

SQL Preview

Displays the labels that will be used in the XML document and its preview. Import setting: If the "SQL Preview" button is pressed, it shows the labels that will be used in the XML document and the first 5 lines from the database. All data items in the input will be converted by default to element content, but this can be over-ridden by clicking on the individual column headers. Clicking once on a column header (ex Heading0) will cause the data from this column to be used as attribute values on the row elements. Clicking a second time - the column's data will be ignored when generating the XML file. You can cycle through these three options by continuing to click on the column header. If the data column will be converted to element content, the header will contain the "<>" symbol. If the data column will be converted to attribute content, the header will contain the "=" symbol, and if it will be skipped, the header will contain "x".

Change labels

This button opens a new dialog, allowing you to edit the names of the root and row elements, change the XML name and the conversion criterion.

The XML names can be edited by double-clicking on the desired item and entering the required label. The conversion criterion can also be modified by selecting from the drop-down list ELEMENT, ATTRIBUTE or SKIPPED.

Save in file

If checked, the new XML document will be saved at the specified path.

Note

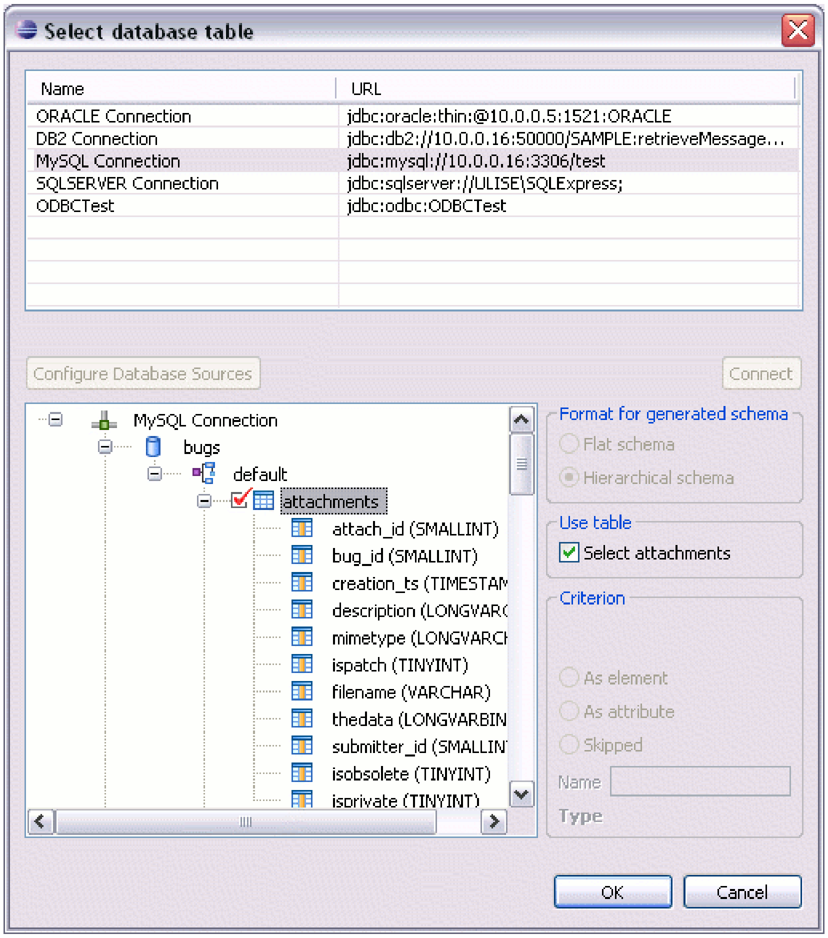
If only Open in editor is checked, the newly created document will be opened in the editor, but as an unsaved file.

Generate XML Schema

Allows you to specify the path of the generated XML Schema file.

Convert table structure to XML Schema

Figure 15.4. Select database table Dialog



Name	URL
ORACLE Connection	jdbc:oracle:thin:@10.0.0.5:1521:ORACLE
DB2 Connection	jdbc:db2://10.0.0.16:50000/SAMPLE:retrieveMessage...
MySQL Connection	jdbc:mysql://10.0.0.16:3306/test
SQLSERVER Connection	jdbc:sqlserver://ULISE\SQLExpress;
ODBCTest	jdbc:odbc:ODBCTest

MySQL Connection
bugs
default
☑ attachments
attach_id (SMALLINT)
bug_id (SMALLINT)
creation_ts (TIMESTAM
description (LONGVARC
mimetype (LONGVARCH
ispatch (TINYINT)
filename (VARCHAR)
thedata (LONGVARBIN
submitter_id (SMALLIN
isobsolete (TINYINT)
isprivate (TINYINT)

Format for generated schema
○ Flat schema
● Hierarchical schema

Use table
☑ Select attachments

Criterion
○ As element
○ As attribute
○ Skipped
Name
Type

Next, in the "Select database table" choose the connection you want to use.

Note

Only connections configured on relational data sources can be used to import to XML or to generate schemas.

You can edit, delete or add a new data source and connection: click on the "Configure Database Sources" button and the "Preferences" dialog will open at the Data Sources section. Click Connect.

Format

Enables you to choose a format for the structure.

- Flat - Generates an XML Schema according to the ISO-ANSI Working draft (Part 14: XML Related Specifications SQL/XML).
- Hierarchical - Represents the database structure as a tree hierarchy taking into account the relationship between tables.

Enable attachments

If checked, the database table is selected for conversion.

Criterion

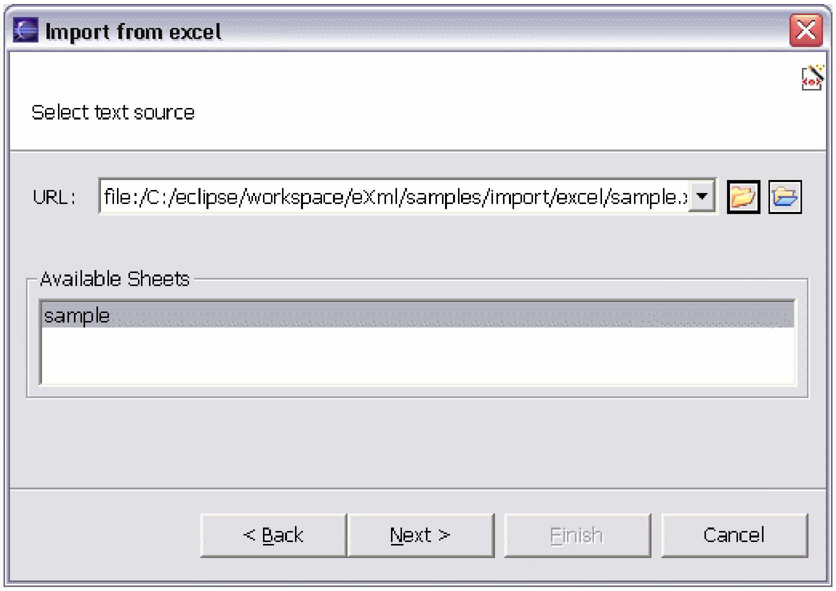
The Criterion options allow the user to specify the name of the selected database column and also how it should be converted into XML. The following options are available:

- Element: When checked the selected column will be converted into an XML element.
- Attribute: If checked the selected column will be converted into an XML attribute.
- Skipped: Is to be selected if the intention is to skip that column from being imported.
- Name: Allows you to specify the name of the column to be imported. Implicitly <oXygen/> suggests an import name that is according to SQL/XML Specification.
- Type: Displays the data type of the imported column.

Import from MS Excel files

<oXygen/> can also import MS (Microsoft) Excel files into XML format documents. To do this, select File → Import... → MS Excel files... In the "Select Excel Sheet" dialog provide the URL of the Excel document, choose one of the available sheets and click Ok.

Figure 15.5. Select Excel Sheet



Import from excel

Select text source

URL: file:/C:/eclipse/workspace/eXml/samples/import/excel/sample.x

Available Sheets

sample

The input data is displayed next in the "Import Criteria" Dialog in a tabular form and the XML Import Preview contains an example of what the generated XML will look like.

The Import Criteria Dialog has a similar behaviour with the one shown in case of "Import from text files".

Note

Please note that Excel sheets saved with versions later that Excel 2002 may not be handled correctly by the Import operation.

Import from HTML files

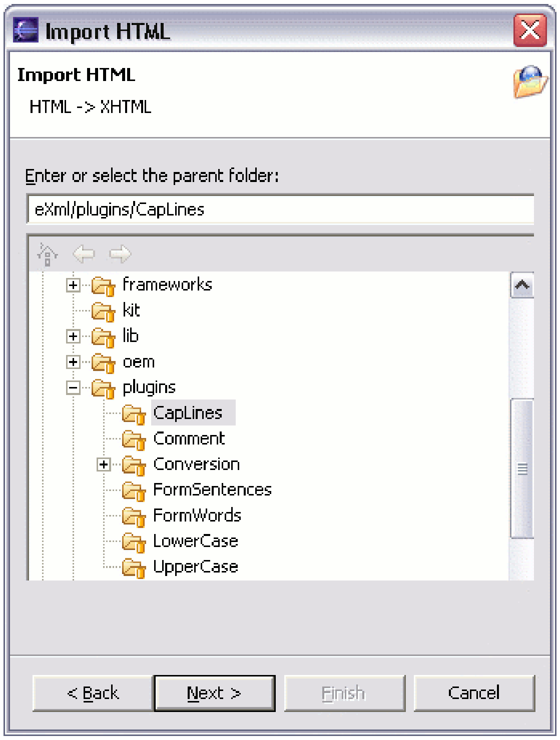
Another format that can be imported in an XML document is HTML.

Procedure 15.1. Import from HTML

1. Select File → Import

2. Select HTML file in the list and click the Next button.

   Figure 15.6. Import HTML file - step 1



Import HTML

Import HTML

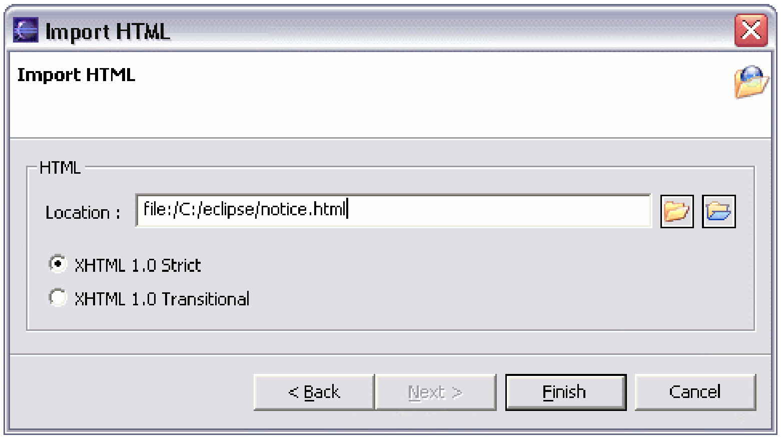
HTML -> XHTML

Enter or select the parent folder:

eXml/plugins/CapLines

3. Type a name for the new document and click the Next button.

Figure 15.7. Import HTML - step 2



Import HTML

Import HTML

HTML

Location : file:/C:/eclipse/notice.html

● XHTML 1.0 Strict

○ XHTML 1.0 Transitional

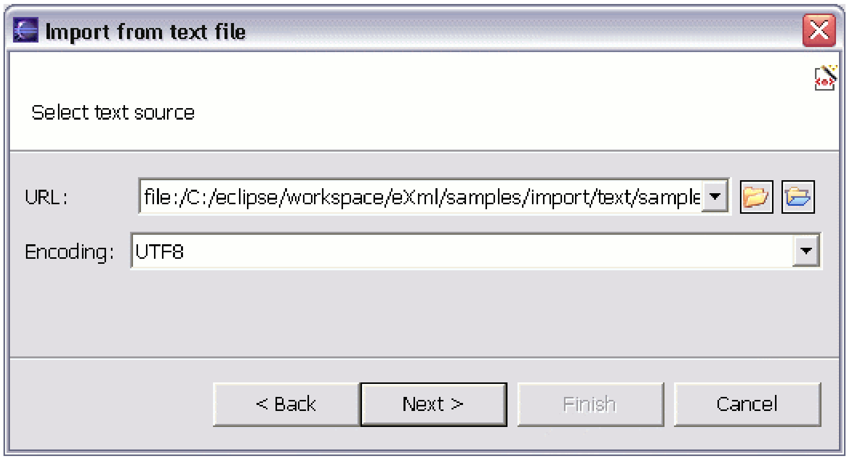
4. Complete the HTML document name and click the OK button.

The resulted document will be an XHTML file containing a DOCTYPE declaration referring to the XHTML DTD definition on the Web and the parsed content of the imported file as XHTML Transitional or Strict depending on what radio button the user chose when performing the import operation.

Import from text files

To import from a text file you'll have to select File → Import... → Text File In the "Select text file" dialog choose the URL and the encoding to be used and click OK.

Figure 15.8. Select text file Dialog



Import from text file

Select text source

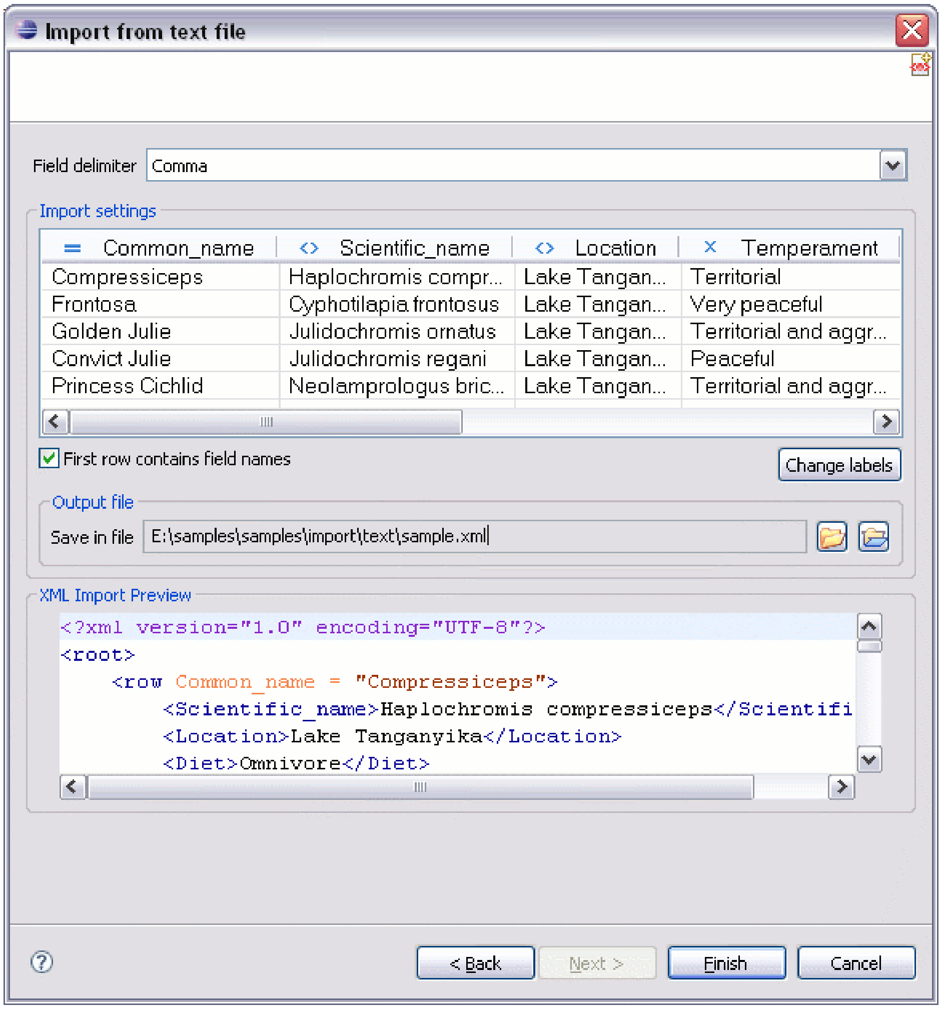
URL: file:/C:/eclipse/workspace/eXml/samples/import/text/sample

Encoding: UTF8

- *URL*: Specifies the location of the text file to be imported.
- *Encoding*: Specifies the encoding

Next, in the "Import Criteria" Dialog select the field delimiter for the import settings. The input data is displayed here in a tabular form and the XML Import Preview contains an example of what the generated XML will look like.

Figure 15.9. Import Text Criteria Dialog



Field delimiter: Comma

Import settings

= Common_name	<> Scientific_name	<> Location	× Temperament
Compressiceps	Haplochromis compr...	Lake Tangan...	Territorial
Frontosa	Cyphotilapia frontosus	Lake Tangan...	Very peaceful
Golden Julie	Julidochromis ornatus	Lake Tangan...	Territorial and aggr...
Convict Julie	Julidochromis regani	Lake Tangan...	Peaceful
Princess Cichlid	Neolamprologus bric...	Lake Tangan...	Territorial and aggr...

☑ First row contains field names

Change labels

Output file

Save in file: E:\samples\samples\import\text\sample.xml

XML Import Preview

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <row Common_name = "Compressiceps">
        <Scientific_name>Haplochromis compressiceps</Scientifi
        <Location>Lake Tanganyika</Location>
        <Diet>Omnivore</Diet>
```

The above table shows the labels that will be used in the XML document and the first 5 lines from the text file in a tabular form. All data items in the input will be converted by default to element content, but this can be over-ridden by clicking on the individual column headers. Clicking once on a column header will cause the data from this column to be used as attribute values on the row elements. Clicking a second time - the column's data will be ignored when generating the XML file. You can cycle through these three options by continuing to click on the column header. If the data column will be converted to element content, the header will contain the "<>" symbol. If the data column will be converted to attribute content, the header will contain the "=" symbol, and if it will be skipped, the header will contain "x".

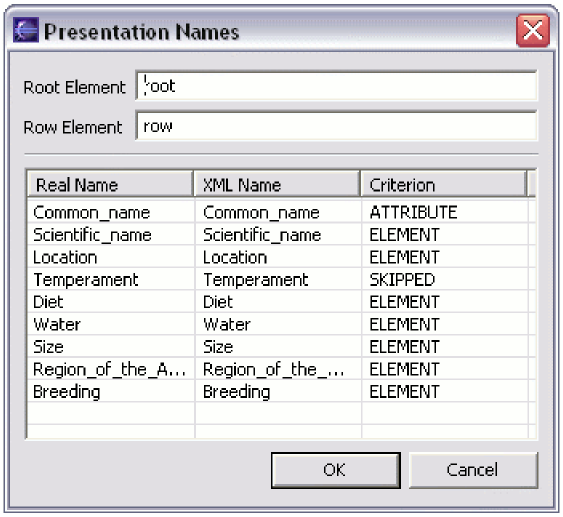
First row contains field names

If the option is checked, you'll notice that the table has moved up; the default column headers are replaced (where there is information) by the content of the first row. In other words, the first row is interpreted as containing the field names. The changes are also visible in the preview of the XML document. To return to default (where the first row is interpreted as not containing field names), simply uncheck the option.

Change labels

If the above option is set, the first row of the input file contains presentation names and these will be used as tokens in the created XML files, otherwise some generic heading names will be used. This button opens a new dialog, allowing you to edit the names of the root and row elements, change the XML name and the conversion criterion.

Figure 15.10. Presentation Names



Root Element: root

Row Element: row

Real Name	XML Name	Criterion
Common_name	Common_name	ATTRIBUTE
Scientific_name	Scientific_name	ELEMENT
Location	Location	ELEMENT
Temperament	Temperament	SKIPPED
Diet	Diet	ELEMENT
Water	Water	ELEMENT
Size	Size	ELEMENT
Region_of_the_A...	Region_of_the_...	ELEMENT
Breeding	Breeding	ELEMENT

The XML names can be edited by double-clicking on the desired item and entering the required label. The conversion criterion can also be modified by selecting from the drop-down list ELEMENT, ATTRIBUTE or SKIPPED.

Output file

Allows you to select the output XML file.

Chapter 16. Composing Web Service calls

Overview

Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information.

The WSDL files contain information about the published services, like the name, the message types and the bindings. The editor is offering a way to edit the WSDL files that is similar to editing XML, the content completion and validation being driven by a mix of the WSDL and SOAP schemas. <oXygen/> supports WSDL version 1.1 and 2.0 and SOAP versions 1.1 and 1.2. That means that in the location where a SOAP extension can be inserted the content completion assistant offers elements from both SOAP 1.1 and SOAP 1.2. Validation of SOAP requests is executed first against a SOAP 1.1 schema and after that against a SOAP 1.2 schema. In addition to validation against the XSD schemas the WSDL file is also analysed during validation so that more element reference specific problems can be detected.

Note

For WSDL 2.0 only content completion and validation are supported. That means if the namespace of the WSDL file is *http://www.w3.org/ns/wsdl* the content completion and validation work with a WSDL 2.0 schema but a SOAP request cannot be obtained and edited correctly yet in the *WSDL SOAP Analyser* view starting from a WSDL 2.0 file.

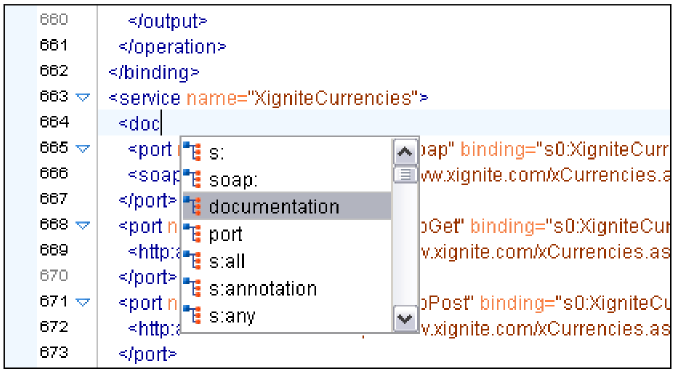
After you edit and validate your Web service descriptor against a mix of the XML Schemas for WSDL and SOAP it is very easy to check if the defined SOAP messages are accepted by the remote Web Services server using <oXygen/>'s WSDL SOAP Analyser integrated tool.

Composing a SOAP request

To design, compose, and test Web service calls in <oXygen/> follow the procedure:

1. Create a new document or open an existing document of type WSDL.
2. Design the Web Service descriptor in the WSDL editor pane where the content completion is driven by a mix of the WSDL and SOAP schemas. You do not need to specify the schema location for the WSDL standard namespaces because <oXygen/> comes with these schemas and uses them by default to assist the user in editing Web Service descriptors.

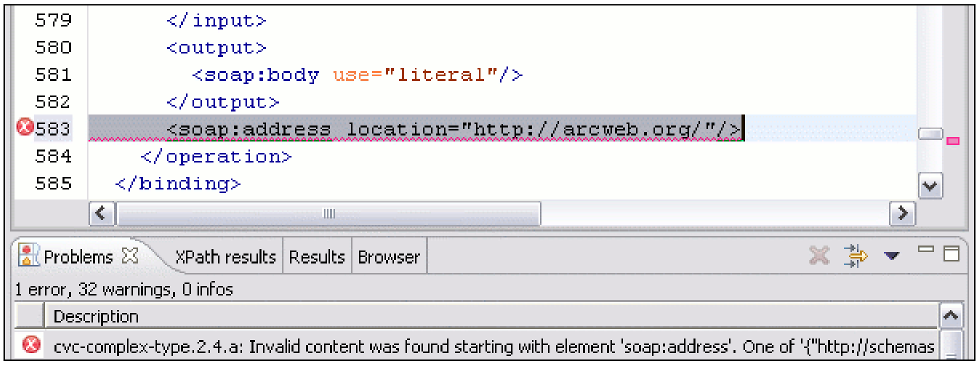
Figure 16.1. Content completion for WSDL documents



```
  </output>
 </operation>
</binding>
<service name="XigniteCurrencies">
 <doc
  <port                ap" binding="s0:XigniteCurr
  <soap                ww.xignite.com/xCurrencies.a
 </port>
 <port n               Get" binding="s0:XigniteCu
  <http:               v.xignite.com/xCurrencies.as
 </port>
 <port n               Post" binding="s0:XigniteCu
  <http:               v.xignite.com/xCurrencies.as
 </port>
```

3. While editing the Web-Services descriptors check their conformance to the WSDL and SOAP schemas. In the following example you can see how the errors are reported.

Figure 16.2. Validating a WSDL file



```
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
      <soap:address location="http://arcweb.org/"/>
    </operation>
  </binding>
```

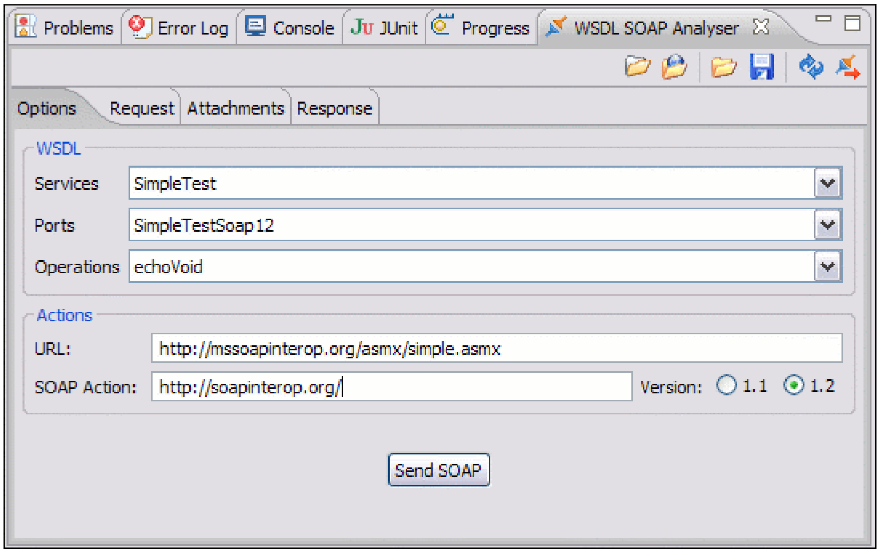
1 error, 32 warnings, 0 infos

Description

cvc-complex-type.2.4.a: Invalid content was found starting with element 'soap:address'. One of '{"http://schemas

4. Check if the defined messages are accepted by the Web Services server. <oXygen/> is providing two ways of testing, one for the currently edited WSDL file and other for the remote WSDL files that are published on a web server.For the currently edited WSDL file open the WSDL SOAP Analyser tool by pressing the toolbar button WSDL SOAP Analyser or use the menu item WSDL → WSDL SOAP Analyser or from the Project view contextual menu select

Figure 16.3. WSDL SOAP Analyser



Services: SimpleTest

Ports: SimpleTestSoap12

Operations: echoVoid

URL: http://mssoapinterop.org/asmx/simple.asmx

SOAP Action: http://soapinterop.org/

Version: ○ 1.1 ● 1.2

It contains a SOAP analyser and sender for Web Services Description Language file types.The analyser fields are:

- Services. The list of services defined by the WSDL file.
- Ports. The ports for the selected service.
- Operations. The list of available operations for the selected service.
- Action URL. Shows the script that serves the operation.
- SOAP Action. Identifies the action performed by the script.
- Version: 1.1 or 1.2. The SOAP version is selected automatically depending on the selected port.
- Request Editor. It allows you to compose the web service request. When an action is selected, <oXygen/> tries to generate as much content as possible for the SOAP request. The envelope of the SOAP request has the correct namespace for the selected SOAP version, that is *http://schemas.xmlsoap.org/soap/envelope/* for SOAP 1.1 or *http://www.w3.org/2003/05/soap-envelope* for SOAP 1.2. Usually you just have to change few values in order for the request to be valid. The content completion is available for this editor and is driven by the schema that defines the type of the current message. While selecting different operations <oXygen/> will remember the modified request for each one. You can press the "Regenerate" button in order to overwrite your modifications for the current request with the initial generated content. The editor has visual line wrap so that all content is visible without scrolling.
- Attachments List. You can define a list of file's URLs to be attached to the request.
- Response Area. Initially it displays an auto generated server sample response so you can have an idea about how the response will look like. After pressing the *Send* button it will present the message received from the server in response to the Web Service request. It may show also error messages. In case the response message contains attachments, <oXygen/> will prompt you to save them, then will try to open them with the associated system application. The response area has visual line wrap so that all content is visible without scrolling.
- Errors List. There may be situations in which the WSDL file is respecting the WSDL XML Schema, but it fails to be valid for example in the case of a message that is defined by means of an element that is not found in the

types section of the WSDL. In such a case, the errors will be listed here. This list is presented only when there are errors.

- Send Button. Executes the request. A status dialog is shown when <oXygen/> is connecting to the server.

The testing of a WSDL file is straight-forward, you just have to click on the WSDL analysis button, then select the service, the port and the operation. The editor will generate the skeleton for the SOAP request. You can edit the request, eventually attach files to it and send it to the server. Watch the server response in the response area. For testing remote WSDL files see the next section.

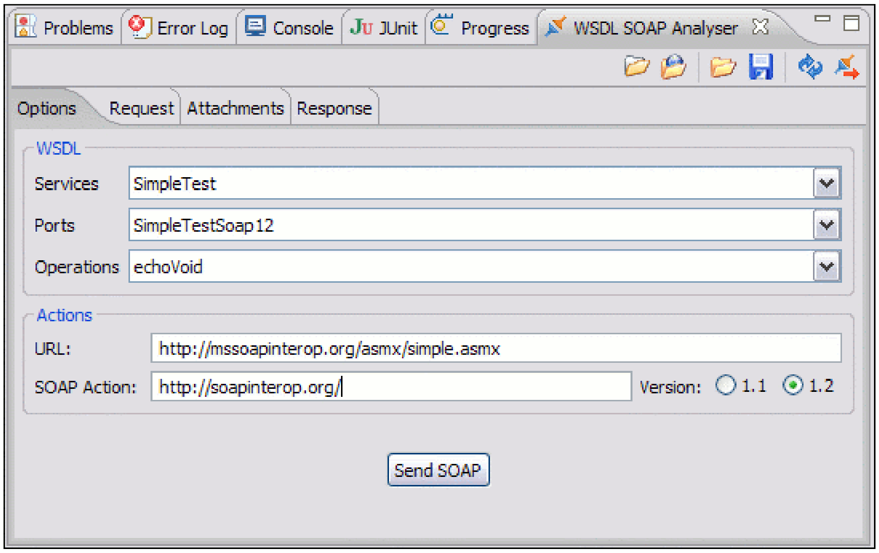
5. Once defined, a request derived from a Web Service descriptor can be saved with the Save button to a Web Service SOAP Call(WSSC) file for later reuse. In this way you will save time in configuring the URLs and parameters.

6. You can open the result of a Web Service call in an editing view. In this way you can save it or process it further.

Testing remote WSDL files

To open and test a remote WSDL file use the menu item Window → Show View → Other+oXygen+WSDL SOAP Analyser ...

Figure 16.4. WSDL File Opener



Services: SimpleTest

Ports: SimpleTestSoap12

Operations: echoVoid

URL: http://mssoapinterop.org/asmx/simple.asmx

SOAP Action: http://soapinterop.org/

Version: ○ 1.1 ● 1.2

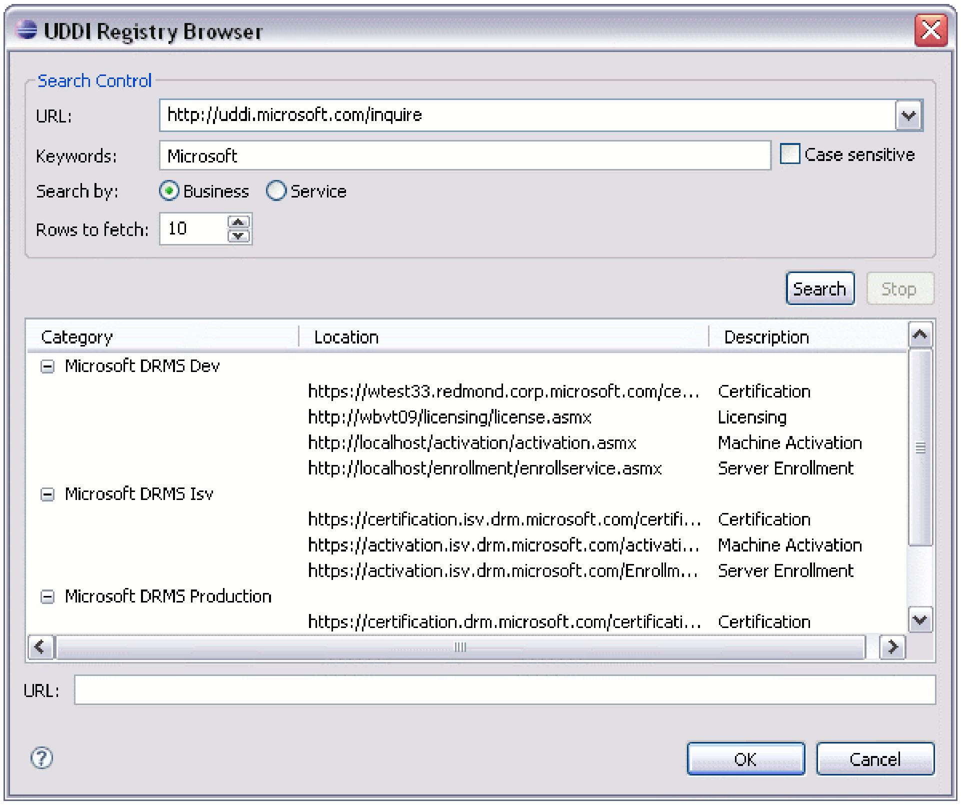
press the *Choose WSDL* button and enter the URL of the remote WSDL file by typing or by browsing the local file system, a remote file system or even a UDDI Registry. Pressing OK will open the WSDL SOAP Analyser tool.

In the Saved SOAP Request tab you can open directly a previously saved Web Service SOAP Call(WSSC) file thus skipping the analysis phase.

The UDDI Registry browser

Pressing the button opens the UDDI Registry Browser dialog.

Figure 16.5. UDDI Registry Browser dialog



URL: http://uddi.microsoft.com/inquire

Keywords: Microsoft

□ Case sensitive

Search by: ● Business ○ Service

Rows to fetch: 10

Category	Location	Description
⊟ Microsoft DRMS Dev		
	https://wtest33.redmond.corp.microsoft.com/ce...	Certification
	http://wbvt09/licensing/license.asmx	Licensing
	http://localhost/activation/activation.asmx	Machine Activation
	http://localhost/enrollment/enrollservice.asmx	Server Enrollment
⊟ Microsoft DRMS Isv		
	https://certification.isv.drm.microsoft.com/certifi...	Certification
	https://activation.isv.drm.microsoft.com/activati...	Machine Activation
	https://activation.isv.drm.microsoft.com/Enrollm...	Server Enrollment
⊟ Microsoft DRMS Production		
	https://certification.drm.microsoft.com/certificati...	Certification

URL:

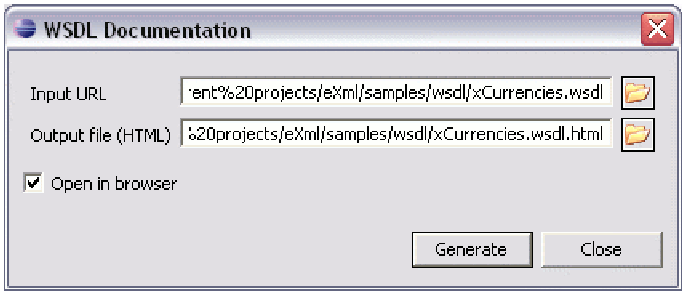
- In the URL combo box type the URL of an UDDI registry or choose one list.
- In the Keywords field enter the string you want to be used when searching the selected UDDI registry for available Web services.
- Optionally, you may change:
  - Rows to fetch - The maximum number of rows to be displayed in the result list.
  - Search by - you can choose to search whether by company or by provided service.
  - Case sensitive - When checked, the search will take into account the Keywords' case.
- Click the Search button. WSDL's that matched the search criteria are added in the result list.
- Select a WSDL from the list and click OK. The UDDI Registry Browser dialog is closed and you are returned to the WSDL File Opener dialog.

Generate WSDL documentation

To generate documentation for a WSDL document use the action XML Tools → Generate Documentation → WSDL Documentation.

The WSDL documentation dialog can be also opened from the Navigator contextual menu: Generate WSDL Documentation

Figure 16.6. WSDL Documentation dialog



Input URL: ·ent%20projects/eXml/samples/wsdl/xCurrencies.wsdl

Output file (HTML): %20projects/eXml/samples/wsdl/xCurrencies.wsdl.html

☑ Open in browser

- In the Input URL field type the URL of the file or click on the browse button and select it from the file system.
- In the Output file(HTML) field you will have to enter the path and the filename where the documentation will be generated.
- If you want the result to be opened in a browser, select the corresponding checkbox.
- Click the Generate button and the documentation for the WSDL file will be generated.

Chapter 17. Digital signature

Overview

Digital signatures are widely used as security tokens, not just in XML.

A digital signature provides a mechanism for assuring integrity of data, the authentication of its signer, and the nonrepudiation of the entire signature to an external party.

- a digital signature must provide a way to verify that the data has not been modified or replaced to ensure integrity.
- the signature must provide a way to establish the identity of the data's signer for authentication.
- the signature must provide the ability for the data's integrity and authentication to be provable to a third party for nonrepudiation.

A public key system is used to create the digital signature and it's also used for verification. The signature binds the signer to the document because digitally signing a document requires the originator to create a hash of the message and then encrypt that hash value with his own private key. Only the originator has that private key and he is the only one can encrypt the hash so that it can be unencrypted using his public key. The recipient, upon receiving both the message and the encrypted hash value, can decrypt the hash value, knowing the originator's public key. The recipient must also try to generate the hash value of the message and compare the newly generated hash value with the unencrypted hash value received from the originator. If the hash values are identical, it proves that the originator created the message, because only the actual originator could encrypt the hash value correctly.

XML Signatures can be applied to any digital content (data object), including XML (see W3C Recommendation, XML-Signature Syntax and Processing [http://www.w3.org/TR/xmldsig-core/]). An XML Signature may be applied to the content of one or more resources.

- Enveloped or enveloping signatures are over data within the same XML document as the signature.
- Detached signatures are over data external to the signature element; the signature is "detached" from the content it signs. This definition typically applies to separate data objects, but it also includes the instance where the Signature and data object reside within the same XML document but are sibling elements.

The XML Signature is a method of associating a key with referenced data; it does not normatively specify how keys are associated with persons or institutions, nor the meaning of the data being referenced and signed.

The original data is not actually signed; instead, the signature is applied to the output of a chain of canonicalization and transformation algorithms, which are applied to the data in a designated sequence. This system provides the flexibility to accommodate whatever "normalization" or desired preprocessing of the data that might be required or desired before subjecting it to being signed.

To canonicalize something means to put it in a standard format that everyone generally uses. Because the signature is dependent on the content it is signing, a signature produced from a not canonicalized document could possibly be different from one produced from a canonicalized document. The canonical form of an XML document is physical representation of the document produced by the method described in this specification. The term canonical XML refers to XML that is in canonical form. The XML canonicalization method is the algorithm defined by this specification that generates the canonical form of a given XML document or document subset. The term XML canonicalization refers to the process of applying the XML canonicalization method to an XML document or document subset. XML canonicalization is designed to be useful to applications that require the ability to test whether the information content of a document or document subset has been changed. This is done by comparing the canonical form of the original document before application processing with the canonical form of the document result of the application processing.

A digital signature over the canonical form of an XML document or document subset would allows the signature digest calculations to be oblivious to changes in the original document's physical representation. During signature generation, the digest is computed over the canonical form of the document. The document is then transferred to the relying party, which validates the signature by reading the document and computing a digest of the canonical form of the received document. The equivalence of the digests computed by the signing and relying parties (and hence the equivalence of the canonical forms over which they were computed) ensures that the information content of the document has not been altered since it was signed.

The following canonicalization algorithms are used in <oXygen/>: Canonical XML (or Inclusive XML Canonicalization)(XMLC14N [http://www.w3.org/TR/2001/REC-xml-c14n-20010315]) and Exclusive XML Canonicalization(EXCC14N [http://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/]). The first is used for XML where the context doesn't change while the second was designed for canonicalization where the context might change.

Inclusive Canonicalization copies all the declarations, even if they are defined outside of the scope of the signature. In this way all the declarations you might use will be unambiguously specified. A problem appears when the signed XML is moved into another XML document which has other declarations because the Inclusive Canonicalization will copy then and the signature will be invalid.

Exclusive Canonicalization finds out what namespaces you are actually using (the ones that are a part of the XML syntax) and just copies those. It does not look into attribute values or element content, so the namespace declarations required to process these are not copied.

This type of canonicalization is useful when you have a signed XML document that you wish to insert into other XML documents and it will insure the signature verifies correctly every time, so it is required when you need self-signed structures that support placement within different XML contexts.

Inclusive Canonicalization is useful when it is less likely that the signed data will be inserted in other XML document and it's the safer method from the security perspective because it requires no knowledge of the data that are to be secured in order to safely sign them.

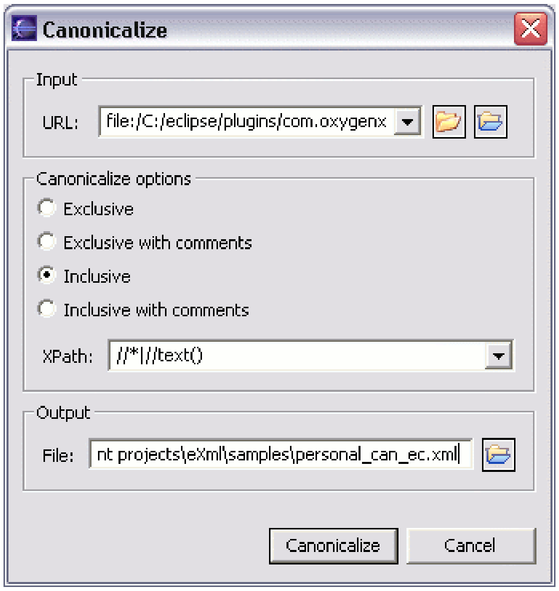
The canonicalization method can specify whether or not comments should be included in the canonical form output by the XML canonicalization method. If a canonical form contains comments corresponding to the comment nodes in the input node-set, the result is called canonical XML with comments. In an uncommented canonical form comments are removed, including delimiter for comments outside document element.

These three operations: Digital Signing, Canonicalization and Verification of the signature are available from the Tools menu or from the Editor contextual menu->Source.

Canonicalizing files

The user can select the canonicalization algorithm to be used for his document from the following dialog displayed by the action *Canonicalize* available from editor panel context menu+Source

Figure 17.1. Canonicalization settings dialog



URL: file:/C:/eclipse/plugins/com.oxygenx

○ Exclusive
○ Exclusive with comments
● Inclusive
○ Inclusive with comments

XPath: //*|//text()

File: nt projects\eXml\samples\personal_can_ec.xml

URL
Specifies the location of the input URL

Exclusive
If selected, the exclusive (uncommented) canonicalization method is used.

Exclusive with comments
If selected, the exclusive with comments canonicalization method is used.

Inclusive
If selected, the inclusive (uncommented) canonicalization method is used.

Inclusive with comments
If selected, the inclusive with comments canonicalization method is used.

XPath
The XPath expression provides the fragments of the XML document to be signed.

Output
Specifies the output file path where the signed XML document will be saved.

Open in editor
If checked, the output file will be opened in the editor.

Certificates

A certificate is a digitally signed statement from the issuer (an individual, an organization, a website or a firm), saying that the public key (and some other information) of some other entity has a particular value. When data is digitally signed, the signature can be verified to check the data integrity and authenticity. Integrity means that the data has not been modified. Authenticity means the data comes indeed from the entity that claims to have created and signed it. Certificates are kept in special repositories called Keystores.

A Keystore is an encrypted file that contains private keys and certificates. All keystore entries (key and trusted certificate entries) are accessed via unique aliases. An alias must be assigned for every new entry of either a key or certificate as a reference for that entity. No Keystore can store an entity if it's "alias" already exists in that Keystore and no KeyStore can store trusted certificates generated with keys in it's KeyStore.

In <oXygen/> there are provided two types of keystores: Java Key Store (JKS) and Public-Key Cryptography Standards version 12 (PKCS-12). A keystore file is protected by a password. In a PKCS 12 keystore you should not store a certificate without alias together with other certificates, with or without alias, as in such a case the certificate without alias cannot be extracted from the keystore.

To set the options for a certificate or to validate it, go to Options → Preferences → Certificates .

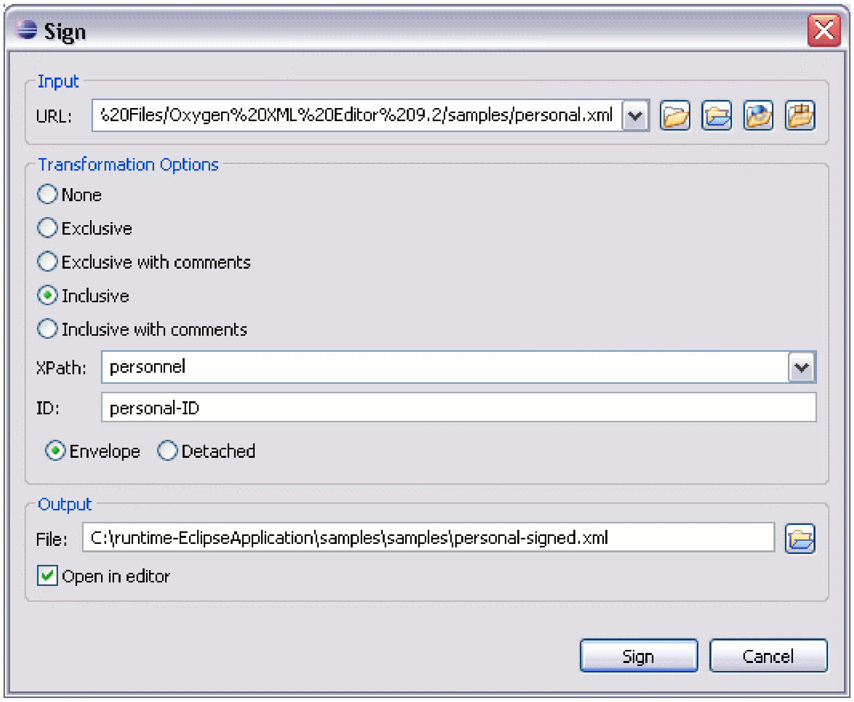
Note

A certificate without alias stored in a PKCS 12 keystore together with other certificates, with or without alias, cannot be always extracted correctly from the keystore due to the missing alias. Such a certificate should be the only certificate of a PKCS 12 keystore.

Signing files

The user can select the type of signature to be used for his document from the following dialog displayed by the action *Sign* available from editor panel context menu+Source

Figure 17.2. Signature settings dialog



Sign

Input

URL: %20Files/Oxygen%20XML%20Editor%209.2/samples/personal.xml

Transformation Options

○ None
○ Exclusive
○ Exclusive with comments
● Inclusive
○ Inclusive with comments

XPath: personnel

ID: personal-ID

● Envelope ○ Detached

Output

File: C:\runtime-EclipseApplication\samples\samples\personal-signed.xml

☑ Open in editor

URL
Specifies the location of the input URL

None
If selected, no canonicalization algorithm is used.

Exclusive
If selected, the exclusive (uncommented) canonicalization method is used.

Exclusive with comments
If selected, the exclusive with comments canonicalization method is used.

Inclusive
If selected, the inclusive (uncommented) canonicalization method is used.

Inclusive with comments
If selected, the inclusive with comments canonicalization method is used.

XPath
The XPath expression provides the fragments of the XML document to be signed.

ID
Provides ID of the XML element to be signed.

Envelope — If selected, the enveloping signature is used.

Detached — If selected, the detached signature is used.

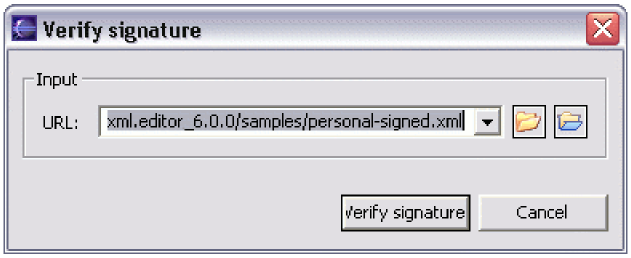
Output — Specifies the output file path where the signed XML document will be saved.

Open in editor — If checked, the output file will be opened in the editor.

Verifying the signature

The user can select a file to verify its signature in the following dialog displayed by the action *Verify Signature* available from editor panel context menu+Source

Figure 17.3. Verifying signature dialog



URL: xml.editor_6.0.0/samples/personal-signed.xml

URL Specifies the location of the document for which to verify the signature.

If the signature is valid, a dialog displaying the name of the signer will be opened. If not, an error message will show details about the problem.

Chapter 18. Text editor specific actions

<oXygen/> provides user actions common in any text editor:

Finding and replacing text in the current file

The Find All Elements dialog

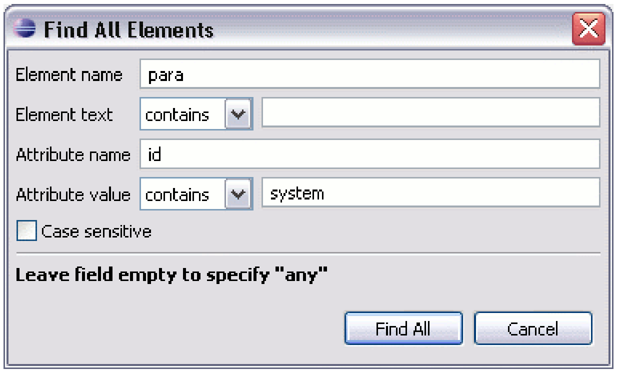
The Find All Elements dialog opened with the menu entry Edit → Find All Elements... assists you in defining "search for XML elements" operations on the current document. As a result, the dialog can perform the following:

- Find all the elements with a specified name
- Find all the elements which contain a specified string in their text
- Find all the elements which have a specified attribute
- Find all the elements which have an attribute with a specified value

All these search criteria can be combined to fine filter your results.

The results of all the operations in the Find All Elements dialog will be presented as a list in the Message Panel.

Figure 18.1. Find All Elements Dialog



Find All Elements

Element name: para

Element text: contains

Attribute name: id

Attribute value: contains system

□ Case sensitive

Leave field empty to specify "any"

The dialog fields are described as follows:

Element name
: The target element name to search for. Only the elements with this exact name are returned. For any element name just leave the field empty.

Element text
: The target element text to search for. The combo box beside this field allows you to specify that you are looking for an exact or partial match of the element text. For any element text, select *contains* in the combo box and leave the field empty.

If you leave the field empty but select *equals* in the combo box, only elements with no text will be found.

Attribute name
: The name of the attribute which needs to be present in the elements. Only the elements which have an attribute with this name will be returned. For any/no attribute name just leave the field empty.

Attribute value

The attribute value The combo box beside this field allows you to specify that you are looking for an exact or partial match of the attribute value. For any/no attribute value select *contains* in the combo box and leave the field empty.

If you leave the field empty but select *equals* in the combo box, only elements that have at least an attribute with an empty value will be found.

Case sensitive

When this option is checked, operations are case sensitive.

VI editor actions

The Text mode editor implements many actions known from the VI text editor:

Ctrl+Delete (Meta+Delete on Mac)	Delete next word
Ctrl+Backspace (Meta+Backspace on Mac)	Delete previous word
Ctrl+W (Meta+W on Mac)	Cut previous word
Ctrl+K (Meta+K on Mac)	Cut to end of line

Chapter 19. Configuring the application

Importing/Exporting Global Options

In the <oXygen/> preferences page (Window -> Preferences -> oXygen) you can find the Import/Export preferences buttons which allow you to move your global preferences in XML format from one computer to another.

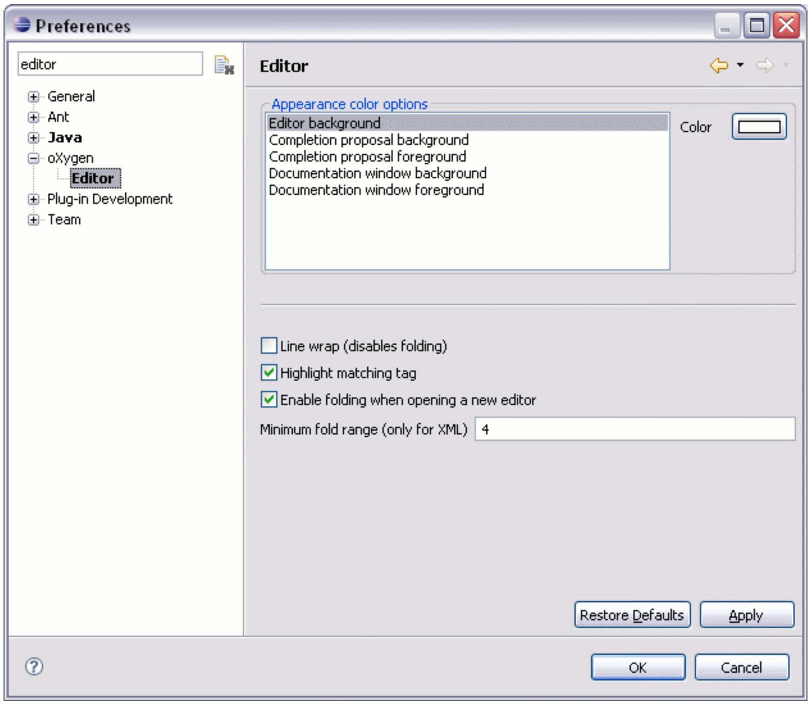
Preferences

Once the application is installed you can use the Preferences dialog accessed from Options → Preferences to customize the application for your requirements and network environment.

You can always revert modifications to their default values by using the Restore Defaults button, available in each preference page.

A restricted version of the Preferences dialog is available at any time in editors of the <oXygen/> plugin by right-clicking in the editor panel and selecting *Preferences*:

Figure 19.1. Eclipse Preferences dialog - restricted version



□ Line wrap (disables folding)

☑ Highlight matching tag

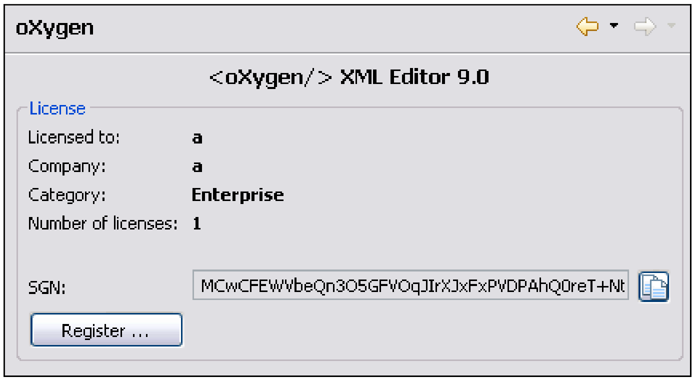
☑ Enable folding when opening a new editor

Minimum fold range (only for XML) 4

Global

The Global preferences panel is opened from menu Window → Preferences → oXygen

Figure 19.2. The Global preferences panel



<oXygen/> XML Editor 9.0

License

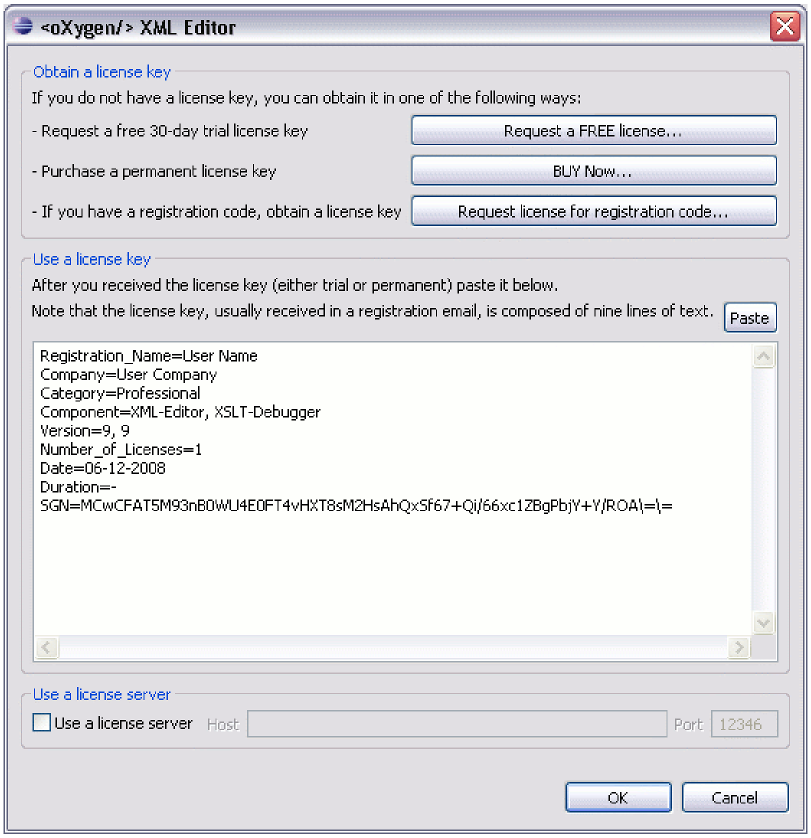
Licensed to: a
Company: a
Category: Enterprise
Number of licenses: 1

SGN: MCwCFEWVbeQn3O5GFVOqJIrXJxFxPVDPAhQ0reT+Nt

License

This panel presents the data of the license key which enables the <oXygen/> plugin: registration name, category and number of purchased licenses, encrypted signature of the license key. Clicking on the Register button opens the <oXygen/> XML Editor License dialog that allows you to insert a new license key:

Figure 19.3. <oXygen/> XML Editor License dialog



If you do not have a license key, you can obtain it in one of the following ways:

- Request a free 30-day trial license key
- Purchase a permanent license key
- If you have a registration code, obtain a license key

After you received the license key (either trial or permanent) paste it below.

Note that the license key, usually received in a registration email, is composed of nine lines of text.

```
Registration_Name=User Name
Company=User Company
Category=Professional
Component=XML-Editor, XSLT-Debugger
Version=9, 9
Number_of_Licenses=1
Date=06-12-2008
Duration=-
SGN=MCwCFAT5M93nB0WU4E0FT4vHXT8sM2HsAhQxSf67+Qi/66xc1ZBgPbjY+Y/ROA\=\=
```

☐ Use a license server Host Port 12346

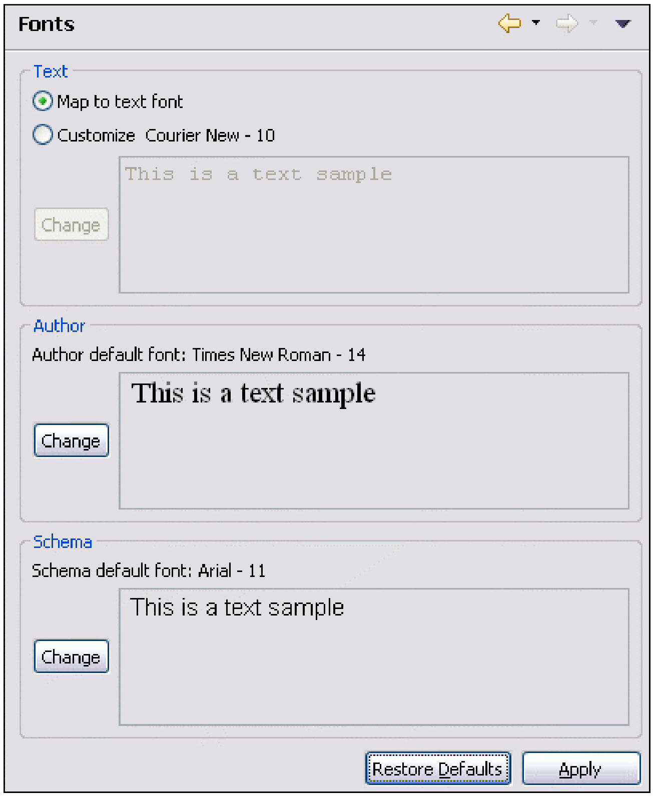
Show hidden files and directories

Show system hidden files and folders in the file and directory browsers. This setting is not available on Mac OS X.

Fonts

The Fonts preferences panel is opened from menu Window → Preferences → oXygen+Fonts

Figure 19.4. The Fonts preferences panel



Fonts

Text

● Map to text font

○ Customize Courier New - 10

This is a text sample

Author

Author default font: Times New Roman - 14

This is a text sample

Schema

Schema default font: Arial - 11

This is a text sample

Text Use this section to customize the font used in text based editors. There are two options:

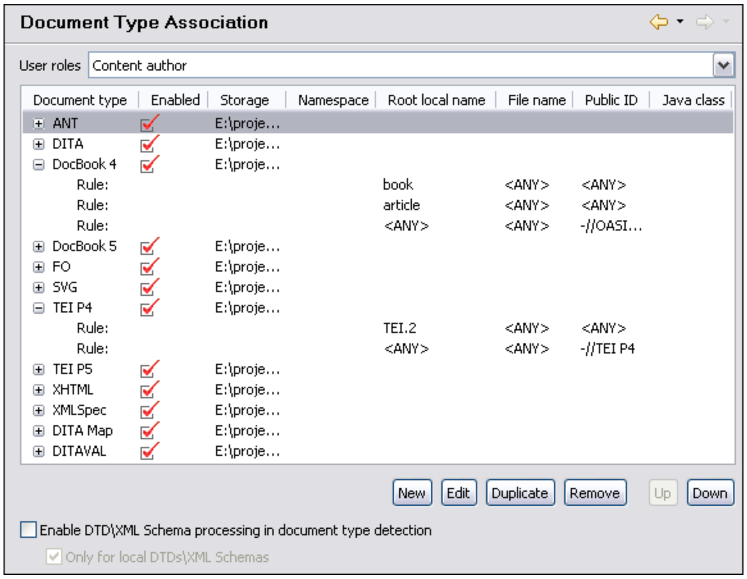
- *Map to text font* - the font used is the same as the one set in General / Appearance / Colors and Fonts for the basic text editor.
- *Customize* - allows you to choose a specific font.

Author Allows you to specify a font to be used in the Author mode.

Document Type Association

The Document Type Association preferences panel is opened from menu Window → Preferences → oXygen+Document Type Association

Figure 19.5. Document Type Association preferences panel



Document Type Association

User roles: Content author

Document type	Enabled	Storage	Namespace	Root local name	File name	Public ID	Java class
⊞ ANT	☑	E:\proje...					
⊞ DITA	☑	E:\proje...					
⊟ DocBook 4	☑	E:\proje...					
Rule:				book	<ANY>	<ANY>	
Rule:				article	<ANY>	<ANY>	
Rule:				<ANY>	<ANY>	-//OASI...	
⊞ DocBook 5	☑	E:\proje...					
⊞ FO	☑	E:\proje...					
⊞ SVG	☑	E:\proje...					
⊟ TEI P4	☑	E:\proje...					
Rule:				TEI.2	<ANY>	<ANY>	
Rule:				<ANY>	<ANY>	-//TEI P4	
⊞ TEI P5	☑	E:\proje...					
⊞ XHTML	☑	E:\proje...					
⊞ XMLSpec	☑	E:\proje...					
⊞ DITA Map	☑	E:\proje...					
⊞ DITAVAL	☑	E:\proje...					

☐ Enable DTD\XML Schema processing in document type detection

☑ Only for local DTDs\XML Schemas

User roles

You can select between two user roles *Content author* and *Developer*. When the selected role is *Content author* you can modify only the properties of the Document Type Associations stored in the user preferences. The externally stored associations cannot be modified and you will have to duplicate them in order to further customize these associations. The *Developer* user can change any document type association.

Document types table

The table presents the currently defined document type associations. The columns are:

Document type

Contains the name of the document type.

Enabled

When checked the corresponding document type association is enabled, it is analyzed when trying to determine the type of a document opened in <oXygen/>.

Storage

Presents the location where the document type association is stored.

When expanding a Document Type Association its defined rules are presented. A rule is described by:

Namespace

Specifies the namespace of the root element from the association rules set (any by default). If you want to apply the rule only when the root element is in no namespace you must leave this field empty (remove the *ANY_VALUE* string).

Root local name

Specifies the local name of the root element (any by default).

File name

Specifies the name of the file (any by default).

Public ID

Represents the Public ID of the matched document.

Java class

Presents the name of the class which will be used to determine if a document matches the rule.

New

Opens a new dialog allowing you to add a new association.

Edit

Opens a new dialog allowing you to edit an existing association.

Delete

Deletes one of the existing association.

Up

Moves the selected association one level up (the order is important because the first document type association in the list that can be associated with the document will be used).

Down

Moves the selected association one level down.

Enable DTD/XML Schema processing in document type detection

When this is enabled the matching process will also examine the DTD/XML Schema associated with the document. For example the fixed attributes declared in the DTD for the root element will be analyzed also if this is specified in the association rules.

Example 19.1. Enabling DTD Processing for DITA customizations

If you are writing DITA customizations you should enable this checkbox. DITA Topics and Maps are also matched by looking for the DITAArchVersion attribute in the root element. If the DTD is not processed on detection then this attribute specified as default in the DTD will not be detected on the root element and the DITA customization will not be correctly matched.

Only for local DTDs/XML Schemas

When the previous feature is enabled you can choose to process only the local DTDs/XML Schemas.

Note

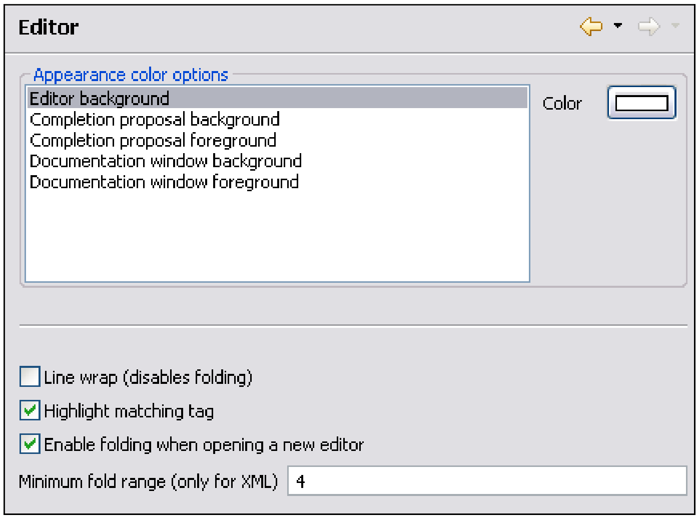
The *Reset Defaults* button that is available in all *Preferences* panels has no effect for document types with external storage.

Editor

The Editor preferences panel is opened from menu Window → Preferences → oXygen+Editor

Use these options to configure the visual aspect of the text editor. The same options panel is available in the restricted version of the *Preferences* dialog.

Figure 19.6. The Editor preferences panel



Editor

Appearance color options

- Editor background
- Completion proposal background
- Completion proposal foreground
- Documentation window background
- Documentation window foreground

Color

☐ Line wrap (disables folding)

☑ Highlight matching tag

☑ Enable folding when opening a new editor

Minimum fold range (only for XML) 4

Editor background color

Use this option to set the background color of the editor and also of the Diff Files editors.

Completion proposal background

Use this option to set the background color for the content completion window.

Completion proposal foreground

Use this option to set the foreground color for the content completion window.

Documentation window background

Use this option to set the background color for the window containing documentation for the content completion elements.

Documentation window foreground

Use this option to set the foreground color for the window containing documentation for the content completion elements.

Line Wrap (disables folding)

This option will do a soft wrap of long lines, that is automatically wrap lines in edited documents. When this option is checked line folding will be disabled.

Highlight matching tag

This option enables highlight for the tag matching the one on which the caret is situated.

Enable folding when opening a new editor

If checked, it enables folding when a new editor is opened.

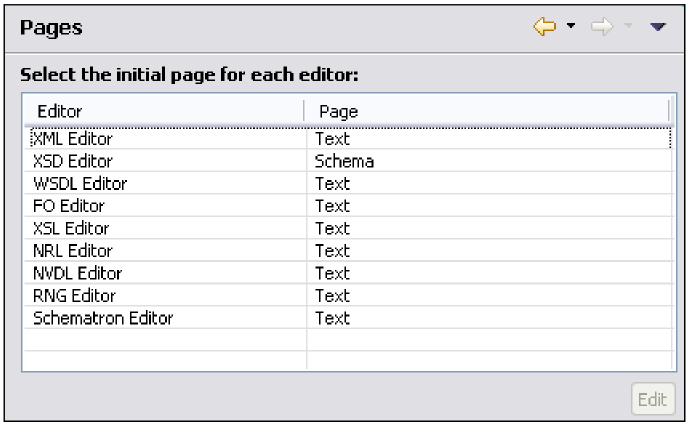
Minimum fold range (only for XML)

If "Enable folding when opening a new editor" is checked, you can specify the minimum number of lines for folding. If you modify this value, you'll notice the changes when you open/reopen the editor.

Pages

The Pages preferences panel is opened from menu Window → Preferences → oXygen → Editor → Pages and allows you to select the initial page for an editor. The mode in which a file was edited in the previous session is saved and will be used when the application is restarted and the file reopened.

Figure 19.7. The <oXygen/> Pages preferences panel



Pages

Select the initial page for each editor:

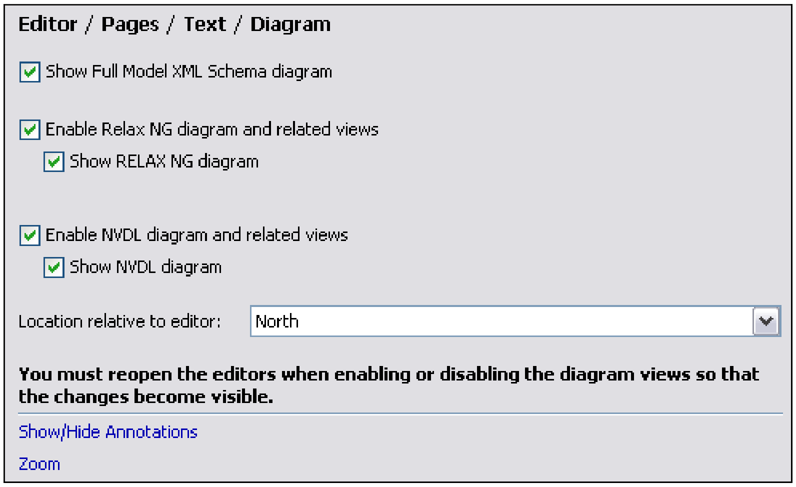
Editor	Page
XML Editor	Text
XSD Editor	Schema
WSDL Editor	Text
FO Editor	Text
XSL Editor	Text
NRL Editor	Text
NVDL Editor	Text
RNG Editor	Text
Schematron Editor	Text

Text/Diagram

If operation is slow for very large schemas disabling the schema diagram view will improve the speed of navigation through the edited schema.

The Diagram preferences panel is opened from menu Window → Preferences → oXygen+Editor+Diagram

Figure 19.8. Schema diagram preferences panel



Editor / Pages / Text / Diagram

☑ Show Full Model XML Schema diagram

☑ Enable Relax NG diagram and related views

☑ Show RELAX NG diagram

☑ Enable NVDL diagram and related views

☑ Show NVDL diagram

Location relative to editor: North

You must reopen the editors when enabling or disabling the diagram views so that the changes become visible.

Show Full Model XML Schema diagram
: If this option is enabled a schema diagram will be available in the Text page for XML Schemas.

Enable Relax NG diagram and related views
: If this option is disabled the schema diagram for Relax NG will not be generated and displayed, also the related views that present the schema components are not populated with data. In case you need the related views to be active, you can let this option checked and un check the following one.

Show Relax NG diagram
: If this option is disabled the schema diagram for Relax NG schemas will not be generated and displayed.

Enable NVDL diagram and related views
: If this option is disabled the schema diagram for NVDL will not be generated and displayed, also the related views that present the schema components are

not populated with data. In case you need the related views to be active, you can let this option checked and un check the following one.

Show NVDL diagram

If this option is disabled the schema diagram for NVDL schemas will not be generated and displayed.

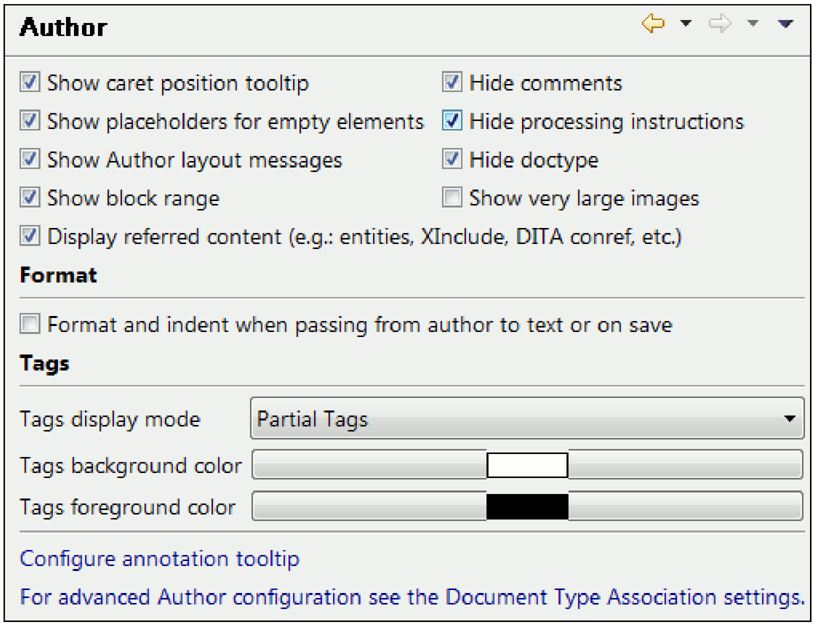
Location relative to editor

The location of the diagram panel in the editing area can be: North, East, South, West. For example North means that the diagram panel takes the North part of the editing area and the text editor panel takes the rest of the editing area.

Author

The Author preferences panel is opened from menu Window → Preferences → oXygen+Editor+Author

Figure 19.9. The <oXygen/> Author preferences panel



Author

☑ Show caret position tooltip
☑ Show placeholders for empty elements
☑ Show Author layout messages
☑ Show block range
☑ Hide comments
☑ Hide processing instructions
☑ Hide doctype
☐ Show very large images
☑ Display referred content (e.g.: entities, XInclude, DITA conref, etc.)

Format

☐ Format and indent when passing from author to text or on save

Tags

Tags display mode: Partial Tags
Tags background color: □
Tags foreground color: ■

Configure annotation tooltip

For advanced Author configuration see the Document Type Association settings.

Show caret position info

If checked, the position information tooltip will be displayed. More information about the position information tooltip can be found in the section Position information tooltip. The documentation tooltip can be disabled from the Content Completion Annotations preferences panel.

Show placeholders for empty elements

When checked, placeholders will be displayed for empty elements to make them clearly visible.

Show Author layout messages

If checked, all errors reported during layout creation will be presented in the *Errors* view.

Show block range

If checked, a block range indicator will be shown in a stripe located in the left side of the editor.

Hide comments

When checked, comments from the documents edited in Author mode will be hidden.

Hide processing instructions

When checked, processing instructions from the documents edited in Author mode will be hidden.

Hide doctype

When checked, doctype sections from the documents edited in Author mode will be hidden.

Show very large images

If unchecked, images larger than 6 megapixels(24MB uncompressed) will not be loaded and displayed in Author mode. Please be aware that this option is unchecked by default because of the large amounts of application memory that images of high resolution can occupy. As a result, an OutOfMemory error could occur which would practically make <oXygen/> unusable without a restart of the entire application.

Display referred content (e.g.: entities, XInclude, DITA conref, etc.)

When checked, the references(entities, XInclude, DITA conref, etc) will also display the content of the resources they refer.

Format and indent when passing from author to text or on save

The content of the document is formatted by applying the Format and Indent action on every switch from the author editor to the text editor of the same document.

Tags display mode

Default display mode for element tags presented in Author mode. You can choose between *Full Tags with Attributes*, *Full Tags*, *Block Tags*, *Inline Tags*, *Partial Tags* and *No Tags*.

Tags background color

Allows you to configure the author tags background color.

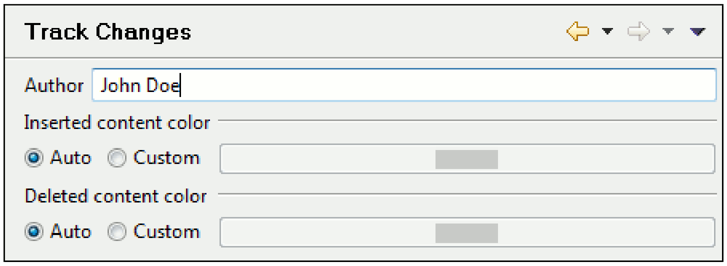
Tags foreground color

Allows you to configure the author tags foreground color.

Track Changes

The Author Track Changes preferences panel is opened from menu Window → Preferences → oXygen+Editor+Author+Track Changes

Figure 19.10. The <oXygen/> Track Changes preferences panel



Track Changes

Author: John Doe

Inserted content color

● Auto ○ Custom

Deleted content color

● Auto ○ Custom

Author

The name of the user who performs the changes when Change Tracking is active for a given editor. This information will be associated with each performed change.

Inserted content color

Auto — Automatically assign colors for the insert changes based on the Author name.

Custom — Use a custom color for all insert changes, regardless of the Author name.

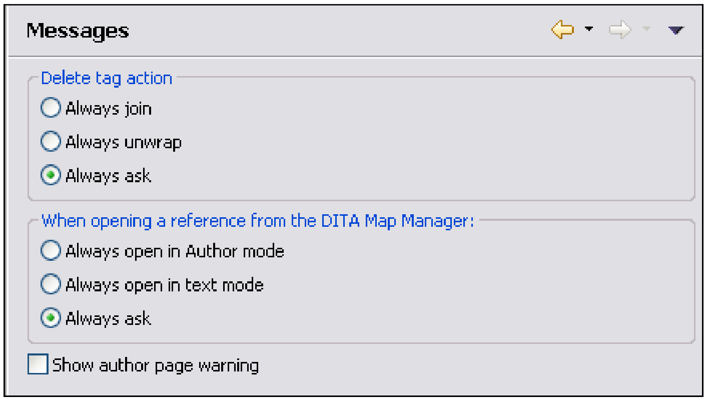
Deleted content color

Auto — Automatically assign colors for the delete changes based on the Author name.

Custom Use a custom color for all delete changes, regardless of the Author name.

Messages

Figure 19.11. The Author's Messages preferences panel



Messages

Delete tag action

○ Always join

○ Always unwrap

● Always ask

When opening a reference from the DITA Map Manager:

○ Always open in Author mode

○ Always open in text mode

● Always ask

□ Show author page warning

Delete tag action

Specifies the default behavior when you delete the start or end marker of an element. You can choose between:

- Always ask
- Always join
- Always unwrap

You can read more about this in Editing the XML markup section.

When opening a map

Specified the default behavior when try to open a map. You can choose between:

- Always open in the DITA Map Manager
- Always open as XML
- Always ask

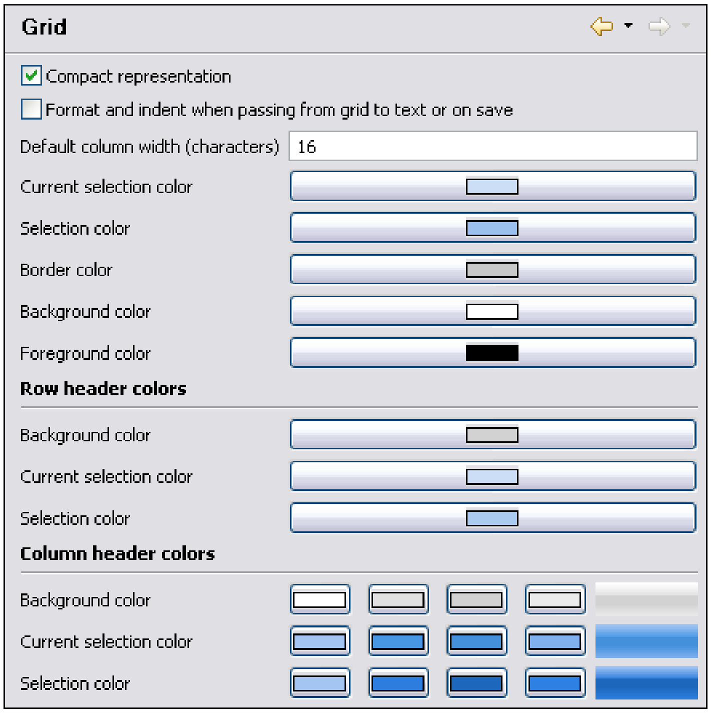
Show author page warning

When checked, a warning dialog will be displayed when switching to Author mode. The warning reminds you that the whitespaces from the text content are evaluated according to the value of the CSS *white-space* property associated to the enclosing elements.

Grid

The Grid preferences panel is opened from menu Window → Preferences → oXygen+Editor+Grid

Figure 19.12. The Grid editor preferences panel



Grid

☑ Compact representation

□ Format and indent when passing from grid to text or on save

Default column width (characters) 16

Current selection color

Selection color

Border color

Background color

Foreground color

Row header colors

Background color

Current selection color

Selection color

Column header colors

Background color

Current selection color

Selection color

Compact representation

If checked a child element is displayed at the same height level with the parent element. If unchecked a child elements is presented nested with one level in the parent container, that is lower than the parent with one row.

Format and indent when passing from grid to text or on save

The content of the document is formatted by applying the Format and Indent action on every switch from the grid editor to the text editor of the same document.

Default column width (characters)

The default width in characters of a table column of the grid. A column can hold an element name and its text content, an attribute name and its value. If the total width of the grid structure is too large you can resize any column with the mouse but the change is not persistent. To make it persistent set the new column width in this user option. the

Current selection color

The background color used in the focused selected cell of the grid to make it different in the set of selected cells. For example when an entire row is selected only one cell of the row is the focused selected one.

Selection color

The background color used in the selected cells of the grid except the focused selected cell which uses a different background color.

Border color

The color used for the lines that separate the grid cells.

Background color

The background color of grid cells that are not selected.

Foreground color

The color of the text used for the element names, text content of elements, attribute names and attribute values.

Row header colors - Background color

The background color of row headers that are not selected.

Row header colors - Current selection color

The background color of the row header that is currently selected and has the focus.

Row header colors - Selection color

The background color of the row header that is currently selected and does not have the focus.

Column header colors - Background color

The background color of column headers that are not selected.

Column header colors - Current selection color

The background color of the column header that is currently selected and has the focus.

Column header colors - Selection color

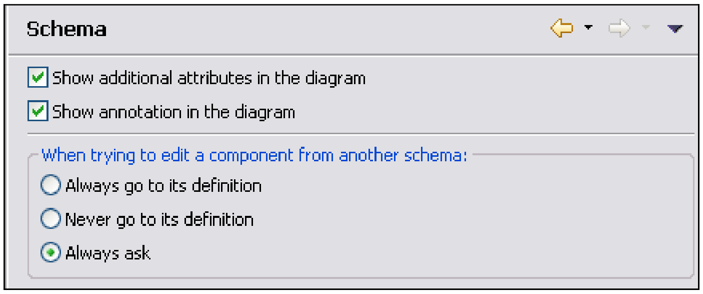
The background color of the column header that is currently selected and does not have the focus.

The column headers are painted with two color gradients, one for the upper 1/3 part of the header and the other for the lower 2/3 part. The start and end colors of the first gradient are set with the first two color buttons. The start and end colors of the second gradient are set with the last two color buttons.

Schema

The XML Schema editor preferences panel is opened from menu Window → Preferences → oXygen+Editor+Schema

Figure 19.13. The XML Schema editor preferences panel



Schema

☑ Show additional attributes in the diagram

☑ Show annotation in the diagram

When trying to edit a component from another schema:

○ Always go to its definition

○ Never go to its definition

● Always ask

Show additional attributes in the diagram

If checked the component symbols of the XML Schema diagram will include also element properties like the name of the referred element (in case of a reference symbol), the base type, etc.

Show annotation in the diagram

The content of *xs:documentation* elements is displayed only if this option is checked.

When trying to edit components from another schema

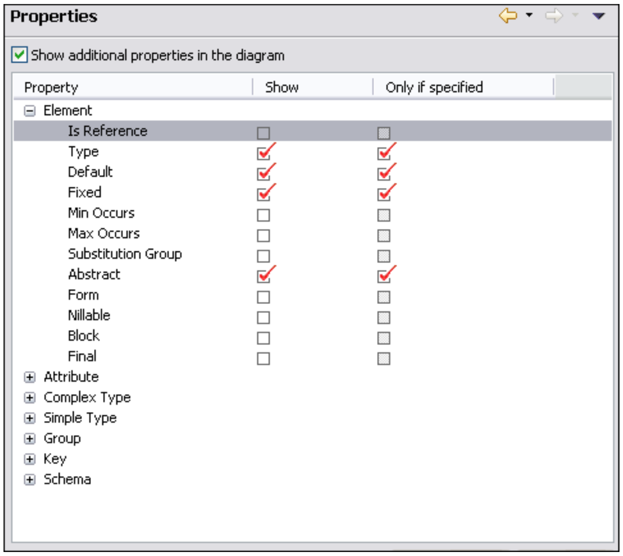
Specifies the default behavior when you try to edit a component from another schema. You can choose between:

- Always to its definition
- Never go to its definition
- Always ask

Properties

You can decide to show additional properties for components in the diagram and customize the properties to be displayed for each schema component.

Figure 19.14. The Schema Properties preferences panel



Properties

☑ Show additional properties in the diagram

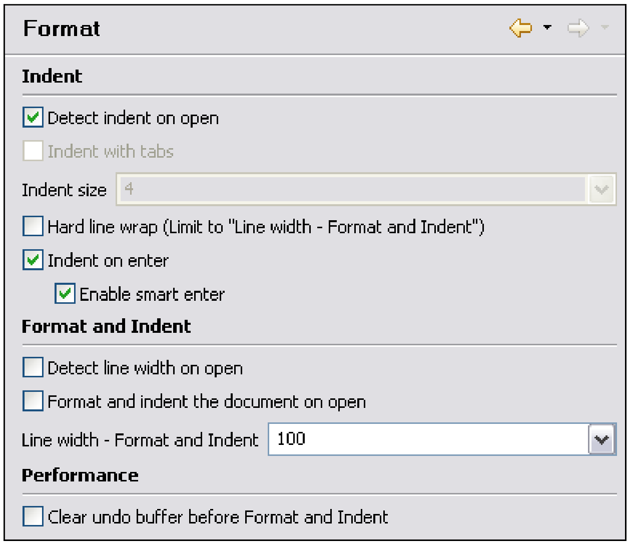
Property	Show	Only if specified
⊟ Element		
Is Reference	□	□
Type	☑	☑
Default	☑	☑
Fixed	☑	☑
Min Occurs	□	□
Max Occurs	□	□
Substitution Group	□	□
Abstract	☑	☑
Form	□	□
Nillable	□	□
Block	□	□
Final	□	□
⊞ Attribute		
⊞ Complex Type		
⊞ Simple Type		
⊞ Group		
⊞ Key		
⊞ Schema		

For a component's properties you can decide if you want to display them only when having a specified value or all the time.

Format

The Format preferences panel is opened from menu Window → Preferences → oXygen+Editor+Format

Figure 19.15. The Format preferences panel



Format

Indent

☑ Detect indent on open

□ Indent with tabs

Indent size: 4

□ Hard line wrap (Limit to "Line width - Format and Indent")

☑ Indent on enter

☑ Enable smart enter

Format and Indent

□ Detect line width on open

□ Format and indent the document on open

Line width - Format and Indent: 100

Performance

□ Clear undo buffer before Format and Indent

Detect indent on open

The editor tries to detect the indent settings of the opened XML document. In this way you can correctly format (pretty-print) files that were created with different settings, without changing your options. More than that you can activate the advanced option for detecting the maximum line width to be used for formatting and hard wrap. These features were designed to minimize the differences created by the pretty print operation when working with a versioning system, like CVS for example.

Indent with tabs

When checked enables 'Indent with tabs' to set the indent to a tab unit. When unchecked, 'Indent with tabs' is disabled and the indent will measure as many spaces as needed in order to go to the next tab stop position. The maximum number of space characters is defined by the 'Indent size' option.

Indent size

Sets the number of spaces or the tab size that will equal a single indent. The Indent can be spaces or a tab, select the preference using the Indent With Tabs option. If set to 4 one tab will equal 4 white spaces or 1 tab with size of 4 characters depending on which option was set in the Indent With Tabs option.

Hard line wrap

This feature saves time when writing a reach text XML document. You can set a limit for the length of the lines in your document. When this limit is exceeded the editor will insert a new line before the word that breaks the limit, and indent the next line. This will minimize the need of reformatting the document.

Indent on Enter

If checked, it indents the new line introduced when pressing Enter.

Enable Smart Enter

If checked, it inserts a new indented line between start and end tag.

Detect line width on open

If checked, it detects the line width automatically when the document is opened.

Format and indent the document on open

When checked, the *Format and indent the document on open* operation will format and indent the document before opening it in the editor panel.

Line width - Format and Indent

Defines the point at which the "Format and Indent" (Pretty-Print) function will perform hard line wrapping. So if set to 100 Pretty-Print will wrap lines at the 100th space inclusive of white spaces, tags and elements.

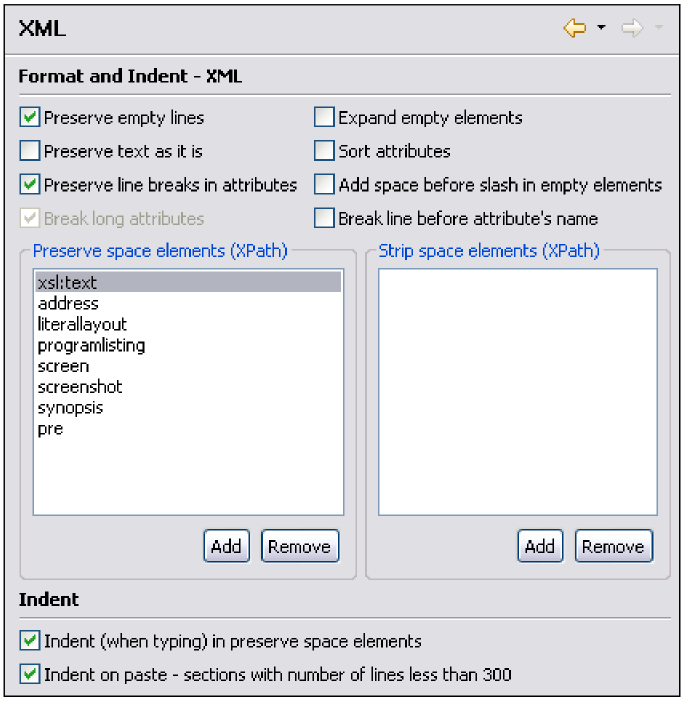
Clear undo buffer before Format and Indent

If checked, the undo buffer is cleared. The undo action can now only undo the Format and Indent action

XML

The XML Format preferences panel is opened from menu Window → Preferences → oXygen+Editor+Format+XML

Figure 19.16. The XML format preferences panel



XML

Format and Indent - XML

☑ Preserve empty lines
□ Expand empty elements
□ Preserve text as it is
□ Sort attributes
☑ Preserve line breaks in attributes
□ Add space before slash in empty elements
☑ Break long attributes
□ Break line before attribute's name

Preserve space elements (XPath)

xsl:text
address
literallayout
programlisting
screen
screenshot
synopsis
pre

Strip space elements (XPath)

Indent

☑ Indent (when typing) in preserve space elements
☑ Indent on paste - sections with number of lines less than 300

Preserve empty lines

When checked the *Format and Indent* operation will preserve all empty lines found in the document on which the pretty-print operation os applied.

Preserve text as it is

If checked, the "Format and Indent" (Pretty-Print) function will preserve text nodes as they are without removing or adding any whitespace.

Preserve line breaks in attributes

If checked, the "Format and Indent" (Pretty-Print) function will preserve the line breaks found in attributes. When this option is checked, *Break long lines* option will be disabled.

Break long attributes

If checked, the "Format and Indent" (Pretty-Print) function will break long attributes.

Expand empty elements

When checked the *Format and Indent* operation will output empty elements with a separate closing tag, ex. <a atr1="v1"></a>. When not checked the same operation will represent an empty element in a more compact form: <a atr1="v1"/>

Sort attributes

When checked the *Format and Indent* operation will sort the attributes of an element alphabetically. When not checked the same operation will leave them in the same order as before applying the operation.

Add space before slash in empty elements

When checked the *Format and Indent* operation will add a space before the closing slash of an empty element, for instance an empty *br* will appear as *<br />*.

Break line before attribute's name

If checked, the "Format and Indent" (Pretty-Print) function will break the line before the attribute's name.

Preserve space elements (XPath)

This list contains simplified XPath expressions for the names of the elements for which the contained white spaces like blanks, tabs and newlines are preserved by the *Format and Indent* operation exactly as before applying the operation. The allowed XPath expressions are of one of the form:

- author
- //listing
- /chapter/abstract/title
- //xs:documentation

The namespace prefixes like *xs* in the previous example are treated as part of the element name without taking into account its binding to a namespace.

Strip space elements (XPath)

This list contains the names of the elements for which contiguous white spaces like blanks, tabs and newlines are merged by the *Format and Indent* operation into one blank.

Indent (when typing) in preserve space elements

If checked, automatic tags indentation while editing will take place for all elements including the ones that are excluded from Pretty Print (default behaviour). When unchecked, indentation while editing will not take place in elements that have the 'xml:space' attribute set on 'preserve' or are in the list of Preserve Space Elements.

Indent on paste

Indent paste text corresponding to the indent settings set by the user. This is useful for keeping the indent style of text copied from other document.

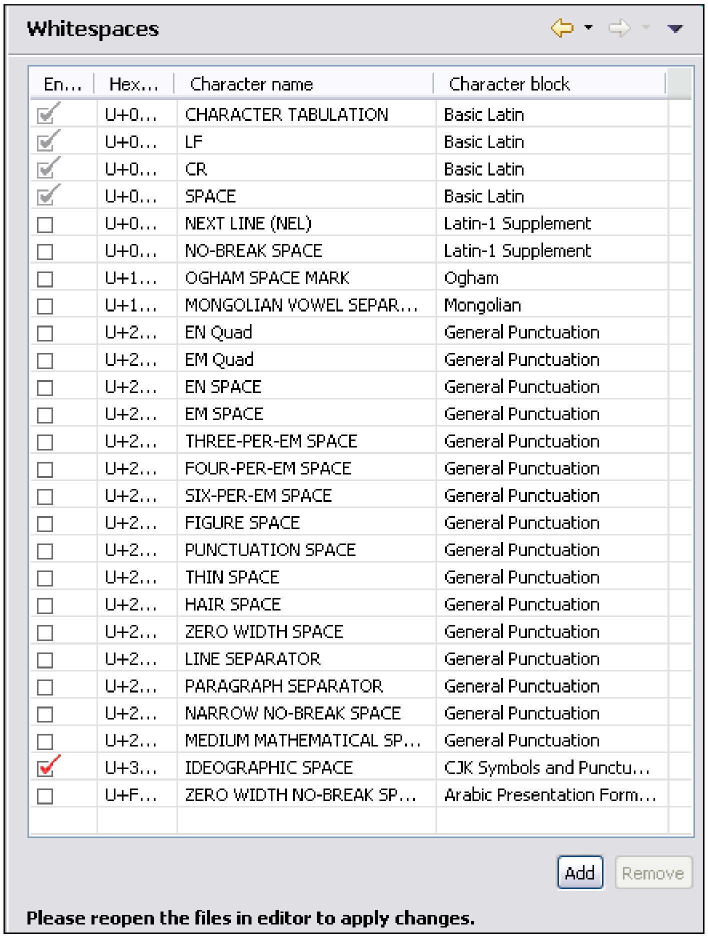
Whitespaces

This panel displays the special whitespace characters of Unicode. Any character that is checked in this panel is considered whitespace that can be normalized in an XML document. The whitespaces are normalized when the action *Format and Indent* is applied or when you switch from Text mode to Author mode or from Author mode to Text mode.

The characters with the codes 9, 10, 13 and 32 are always in the group of whitespace characters that must be normalized so they are always enabled in this panel.

The list of whitespace characters can be improved with additional characters. Any character added to the list is considered whitespace.

Figure 19.17. The Whitespaces preferences panel

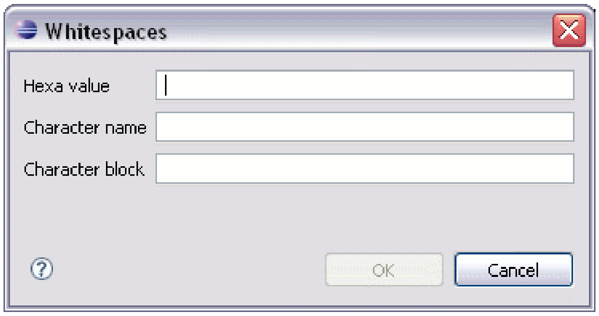


En…	Hex…	Character name	Character block
☑	U+0…	CHARACTER TABULATION	Basic Latin
☑	U+0…	LF	Basic Latin
☑	U+0…	CR	Basic Latin
☑	U+0…	SPACE	Basic Latin
☐	U+0…	NEXT LINE (NEL)	Latin-1 Supplement
☐	U+0…	NO-BREAK SPACE	Latin-1 Supplement
☐	U+1…	OGHAM SPACE MARK	Ogham
☐	U+1…	MONGOLIAN VOWEL SEPAR…	Mongolian
☐	U+2…	EN Quad	General Punctuation
☐	U+2…	EM Quad	General Punctuation
☐	U+2…	EN SPACE	General Punctuation
☐	U+2…	EM SPACE	General Punctuation
☐	U+2…	THREE-PER-EM SPACE	General Punctuation
☐	U+2…	FOUR-PER-EM SPACE	General Punctuation
☐	U+2…	SIX-PER-EM SPACE	General Punctuation
☐	U+2…	FIGURE SPACE	General Punctuation
☐	U+2…	PUNCTUATION SPACE	General Punctuation
☐	U+2…	THIN SPACE	General Punctuation
☐	U+2…	HAIR SPACE	General Punctuation
☐	U+2…	ZERO WIDTH SPACE	General Punctuation
☐	U+2…	LINE SEPARATOR	General Punctuation
☐	U+2…	PARAGRAPH SEPARATOR	General Punctuation
☐	U+2…	NARROW NO-BREAK SPACE	General Punctuation
☐	U+2…	MEDIUM MATHEMATICAL SP…	General Punctuation
☑	U+3…	IDEOGRAPHIC SPACE	CJK Symbols and Punctu…
☐	U+F…	ZERO WIDTH NO-BREAK SP…	Arabic Presentation Form…

Please reopen the files in editor to apply changes.

The Whitespaces dialog is used to add a new character as whitespace by specifying the hexa value, the name and the character block.

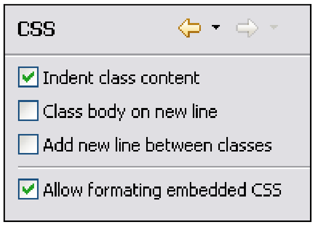
Figure 19.18. The add whitespace dialog



CSS

The CSS Format preferences panel is opened from menu Window → Preferences → oXygen+Editor+Format+CSS

Figure 19.19. The CSS format preferences panel



CSS

☑ Indent class content

□ Class body on new line

□ Add new line between classes

☑ Allow formating embedded CSS

Indent class content

If checked, the class content is indented during a "Format and Indent" (Pretty-Print) operation.

Class body on new line

If checked, the class body (including the curly brackets) are placed on a new line after a Pretty-Print operation.

Add new line between classes

If checked, an empty line is added between two classes after a Pretty-Print operation is performed.

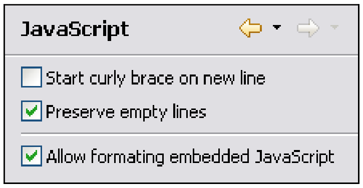
Allow formatting embedded CSS

If checked, the CSS content embedded in XML will be formated when the XML content is formated.

JavaScript

The JavaScript Format preferences panel is opened from menu Window → Preferences → oXygen+Editor+Format+JavaScript

Figure 19.20. The JavaScript Format preferences panel



JavaScript

□ Start curly brace on new line

☑ Preserve empty lines

☑ Allow formating embedded JavaScript

Start curly brace on new line

If true, opening curly braces will start on a new line.

Preserve empty lines

If true, empty lines in the JavaScript code will be preserved.

Allow formatting embedded JavaScript

If checked, the JavaScript content embedded in XML will be formated when the XML content is formated.

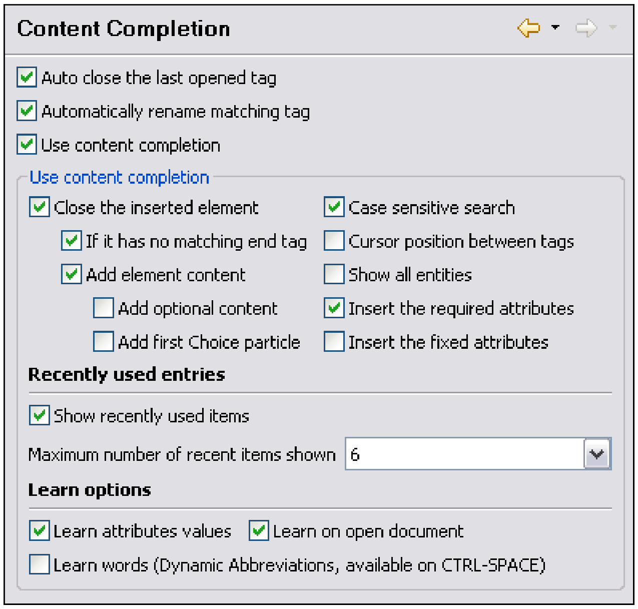
Content Completion

The Content Completion feature enables inline syntax lookup and Auto Completion of mark-up elements and attributes to streamline mark-up and reduce errors while editing.

These settings define the operating mode of the content assistant.

The Content Completion preferences panel is opened from menu Window → Preferences → oXygen+Editor+Content Completion

Figure 19.21. The Content Completion preferences panel



Content Completion

☑ Auto close the last opened tag

☑ Automatically rename matching tag

☑ Use content completion

Use content completion

☑ Close the inserted element

☑ If it has no matching end tag

☑ Add element content

□ Add optional content

□ Add first Choice particle

☑ Case sensitive search

□ Cursor position between tags

□ Show all entities

☑ Insert the required attributes

□ Insert the fixed attributes

Recently used entries

☑ Show recently used items

Maximum number of recent items shown 6

Learn options

☑ Learn attributes values

☑ Learn on open document

□ Learn words (Dynamic Abbreviations, available on CTRL-SPACE)

Auto close the last opened tag

If the Use Content Completion option is not checked and if this option is checked, <oXygen/> will close the last opened tag when </ is typed.

Automatically rename matching tag

If checked, <oXygen/> will automatically rename the matching end tag when the start tag is modified in the editor.

Use Content Completion

When unchecked, all Content Completion features are disabled.

Close the inserted element

When inserting elements from the Content Completion assistant, both start and end tags are inserted.

If it has no matching tag

When checked, the end tag of the inserted element will be automatically added only if it is not already present in the document.

Add element content

When checked, <oXygen/> will insert automatically the required elements from the DTD or XML Schema. This option is applied also in the Author mode of the XML editor.

Add optional content

When checked, <oXygen/> will insert automatically the optional elements from the DTD or XML Schema. This option is applied also in the Author mode of the XML editor.

Add first Choice particle

When checked, <oXygen/> will insert automatically the first Choice particle from the DTD or XML Schema. This option is applied also in the Author mode of the XML editor.

Case sensitive search

When it is checked the search in the content completion window when you type a character is case sensitive ('a' and 'A' are different characters). This option is applied also in the Author mode of the XML editor.

Cursor position between tags

When checked, <oXygen/> will set the cursor automatically between tags. Even if the auto-inserted elements have attributes that are not required, the position of cursor can be forced between tags.

Show all entities

When checked, <oXygen/> will display a list with all the internal and external entities declared in the current document when the user types the start character of an entity reference (i.e. &).

Insert the required attributes

When checked, <oXygen/> will insert automatically the required attributes from the DTD or XML Schema for an element inserted with the help of the Content Completion assistant. This option is applied also in the Author mode of the XML editor.

Insert the fixed attributes

When checked, <oXygen/> will insert automatically any *FIXED* attributes from the DTD or XML Schema for an element inserted with the help of the Content Completion assistant. This option is applied also in the Author mode of the XML editor.

Show recently used items

When checked, <oXygen/> will remember the last inserted items from the Content Completion window. The number of items to be remembered is limited by *Maximum number of recent items shown* combo box. These most frequently used items are displayed on the top of Content Completion window and their icon is decorated with a small red square. This option is applied also in the Author mode of the XML editor.

Maximum number of recent items shown

Limits the number of recently used items presented at the top of the content completion window. This option is applied also in the Author mode of the XML editor.

Learn attributes values

When checked, <oXygen/> will display a list with all attributes values learned from the current document. This option is applied also in the Author mode of the XML editor.

Learn on open document

When checked, <oXygen/> will automatically learn the document structure when the document is opened. This option is applied also in the Author mode of the XML editor.

Learn words (Dynamic Abbreviations, available on CTRL+SPACE)

When checked, <oXygen/> will automatically learn the typed words and will make them available in a Content Completion fashion by pressing CTRL+SPACE.

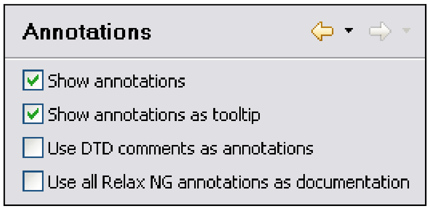
Note

In order to be learned, the words need to be separated by space characters.

Annotations

The Annotations preferences panel is opened from menu Window → Preferences → oXygen+Editor+Content Completion+Annotations

Figure 19.22. The Content Completion Annotations preferences panel



Annotations

☑ Show annotations

☑ Show annotations as tooltip

□ Use DTD comments as annotations

□ Use all Relax NG annotations as documentation

Show annotations

When checked, <oXygen/> will display the annotations that are present in the used schema for the current element, attribute or attribute value. This option is applied also in the Author mode of the XML editor.

Show annotations as tooltip

If checked, it shows the annotations of elements and attributes as tooltips. This option is applied also in the Author mode of the XML editor.

Use DTD comments as annotation

When checked, <oXygen/> will use all DTD comments as annotation.

Use all Relax NG annotations as documentation

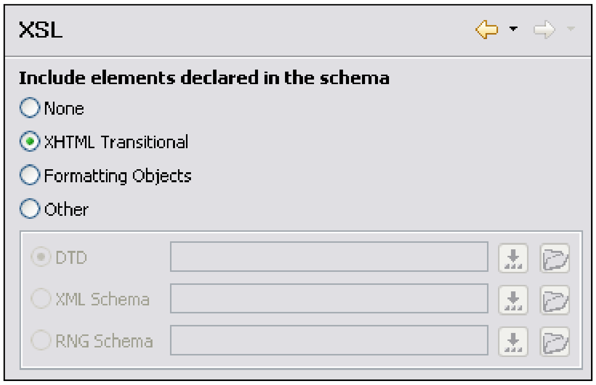
When checked any element that is not from the Relax NG namespace, that is "http://relaxng.org/ns/structure/1.0" will be considered annotation and will be displayed in the annotation window next to the content completion window and in the Model View. When unchecked only elements from the Relax NG annotations namespace, that is "http://relaxng.org/ns/compatibility/annotations/1.0" will be considered annotation.

XSL

These settings define what elements are suggested by the content assistant in addition to the XSL ones.

The XSL preferences panel is opened from menu Window → Preferences → oXygen+Editor+Content Completion+XSL

Figure 19.23. The Content Completion XSL preferences panel



XSL

Include elements declared in the schema

○ None

● XHTML Transitional

○ Formatting Objects

○ Other

● DTD

○ XML Schema

○ RNG Schema

None

The Content Completion will offer only the XSL information.

XHTML transitional

Includes XHTML Transitional elements as substitutes for xsl:element.

Formating objects

Includes Formating Objects elements as substitutes for xsl:element.

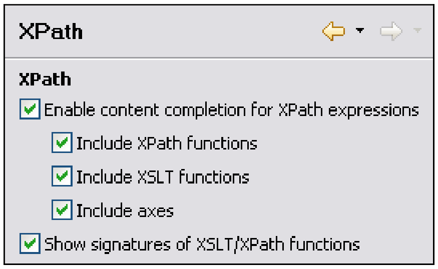
Other

Includes elements from a DTD file, a XML Schema file or a RNG Schema file specified from a URL as substitutes for xsl:element.

XPath

The XPath preferences panel is opened from menu Window → Preferences → oXygen+Editor+Content Completion+XPath

Figure 19.24. The Content Completion XPath preferences panel



XPath

XPath

☑ Enable content completion for XPath expressions

- ☑ Include XPath functions
- ☑ Include XSLT functions
- ☑ Include axes

☑ Show signatures of XSLT/XPath functions

Enable content completion for XPath expressions

Disables and enables content completion in XPath expressions entered in the XSL attributes *match*, *select* and *test* and also in the XPath toolbar.

Options are available to allow the user to include XPath functions, XSLT functions or axes in the content completion suggestion list.

The XPath section controls if the functions, axes are presented in the content completion list when editing XPath expressions.

Show signatures of XSLT/XPath functions

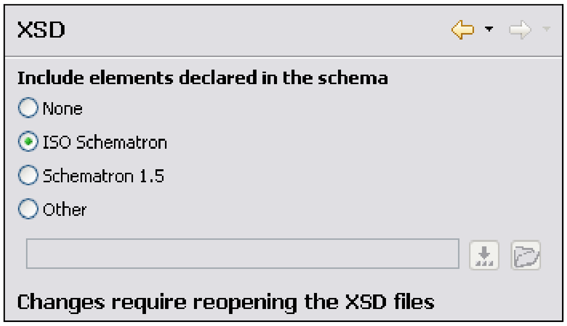
If checked, the editor will indicate in a tooltip helper the signature of the XPath function located at the caret position. See the XPath Tooltip Helper section for more information.

XSD

These settings define what elements are suggested by the content assistant, in addition to the ones from the XML Schema schema, inside the *xs:annotation/xs:appinfo* elements of an XML Schema.

The XSD preferences panel is opened from menu Window → Preferences → oXygen+Editor+Content Completion+XSD

Figure 19.25. The Content Completion XSD preferences panel



XSD

Include elements declared in the schema

○ None

● ISO Schematron

○ Schematron 1.5

○ Other

Changes require reopening the XSD files

None

The Content Completion will offer only the XML Schema schema information.

ISO Schematron

Includes ISO Schematron elements in xs:appinfo.

Schematron 1.5

Includes Schematron 1.5 elements in xs:appinfo.

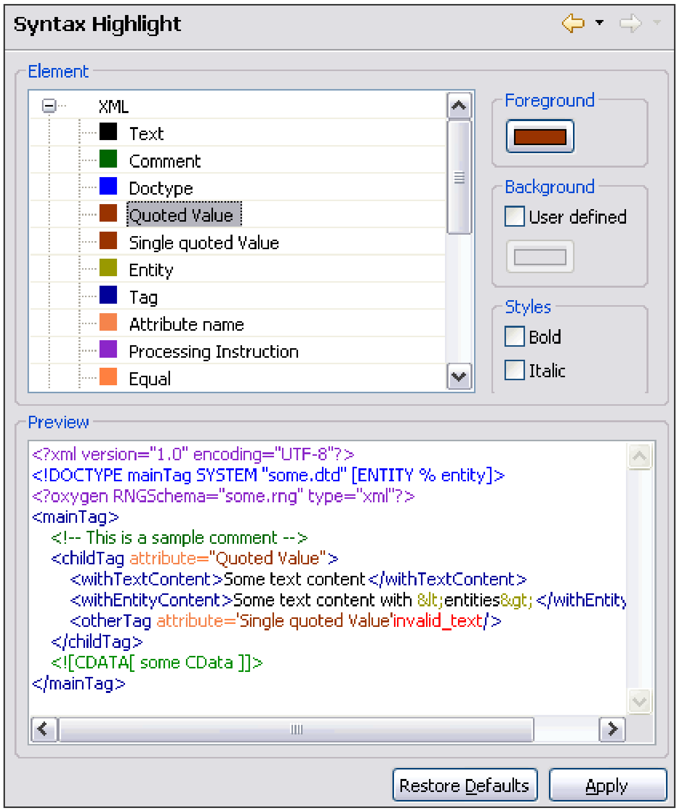
Other	Includes in xs:appinfo elements from an XML Schema specified from a URL.

Syntax Highlight

<oXygen/> supports Syntax Highlight for XML, DTD, Relax NG (XML and Compact Syntax), Java, JavaScript, PHP,CSS, XQuery, C++, C, Perl, Properties, SQL, Shell and Batch documents. While <oXygen/> provides a default color configuration for highlighting the tokens, you may choose to customize, as required, using the Colors dialog.

The Syntax Highlight preferences panel is opened from menu Window → Preferences → oXygen+Editor+Syntax Highlight

Figure 19.26. The Colors preferences panel



XML
- ■ Text
- 🟩 Comment
- 🟦 Doctype
- 🟫 Quoted Value
- 🟫 Single quoted Value
- 🟨 Entity
- 🟦 Tag
- 🟧 Attribute name
- 🟪 Processing Instruction
- 🟧 Equal

□ User defined

□ Bold

□ Italic

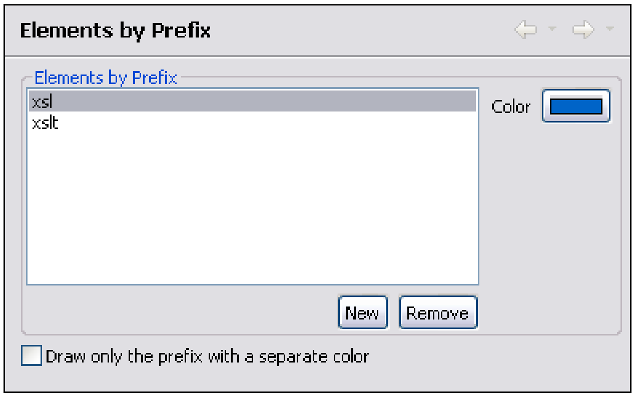
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mainTag SYSTEM "some.dtd" [ENTITY % entity]>
<?oxygen RNGSchema="some.rng" type="xml"?>
<mainTag>
  <!-- This is a sample comment -->
  <childTag attribute="Quoted Value">
    <withTextContent>Some text content</withTextContent>
    <withEntityContent>Some text content with &lt;entities&gt; </withEntity
    <otherTag attribute='Single quoted Value'invalid_text/>
  </childTag>
  <![CDATA[ some CData ]]>
</mainTag>
```

Open the Syntax Highlight panel by selecting Window->Preferences->oXygen->Syntax Highlight and choose one of the supported Document Types. Each document type contains a set of Tokens. When a Document Type node is expanded, the associated tokens are listed. Selecting a token displays the current color properties and enables you to modify them. You can also select a token by clicking directly in the preview area on that type of token.

Syntax Highlight / Elements by Prefix

The Syntax Highlight preferences panel is opened from menu Window → Preferences → oXygen+Editor+Syntax Highlight+Elements by Prefix

Figure 19.27. The Elements by Prefix preferences panel



Elements by Prefix

Elements by Prefix

- xsl
- xslt

Color

□ Draw only the prefix with a separate color

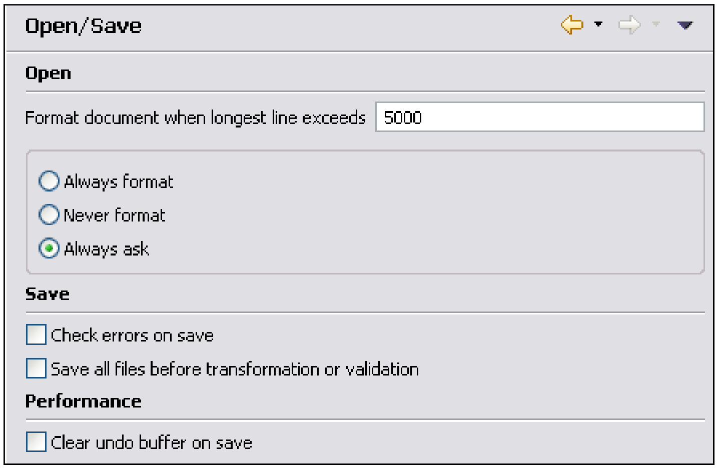
One row of the table contains the association between a namespace prefix and the color used to mark start tags and end tags in that prefix. Note that the marking mechanism does not look at the namespace bound to that prefix. If the prefix is bound to different namespaces in different XML elements of the same file all the tags with the prefix will be marked with the same color.

One can choose that only the prefix to be displayed in the chosen color by checking the *Draw only the prefix with a separate color* option.

Open/Save

The Open/Save preferences panel is opened from menu Window → Preferences → oXygen+Editor+Open/Save

Figure 19.28. The Open/Save preferences panel



Open/Save

Open

Format document when longest line exceeds: 5000

○ Always format
○ Never format
● Always ask

Save

□ Check errors on save
□ Save all files before transformation or validation

Performance

□ Clear undo buffer on save

Format document when long lines exceeds

Specifies the default behavior when the longest line of a document exceeds the specified limit. You can choose between:

- Always format
- Never format
- Always ask

Check well-formedness on save — If selected the <oXygen/> plugin will perform a well-formed check every time the user saves a document.

Save all files before transformation or validation — Save all opened files before validating or transforming an XML document. In this way the dependencies are resolved, for example when modifying both the XML document and its XML Schema.

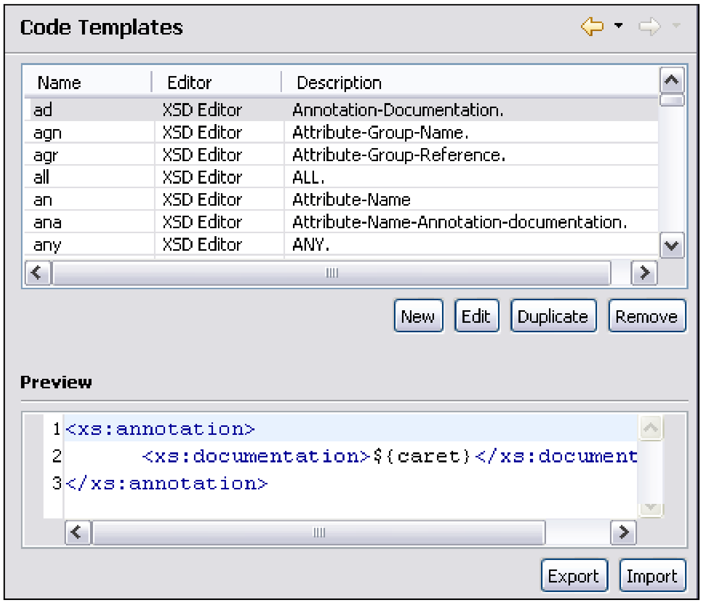
Clear undo buffer on save — If checked, the undo action has no effect after you've saved your document. You can only undo the modifications made after you've saved it.

Code Templates

Code templates are small document fragments that can be reused in other editing sessions. <oXygen/> comes with a large set of ready-to use templates for XSL and XML Schema. You can even share your code templates with your colleagues using the Export and Import functions. To obtain the template list you have use the Content Completion on request shortcut key (usually CTRL-SPACE).

The Code Templates preferences panel is opened from menu Window → Preferences → oXygen+Editor+Templates+Code Templates

Figure 19.29. The Code Templates preferences panel



Name	Editor	Description
ad	XSD Editor	Annotation-Documentation.
agn	XSD Editor	Attribute-Group-Name.
agr	XSD Editor	Attribute-Group-Reference.
all	XSD Editor	ALL.
an	XSD Editor	Attribute-Name
ana	XSD Editor	Attribute-Name-Annotation-documentation.
any	XSD Editor	ANY.

```
<xs:annotation>
      <xs:documentation>${caret}</xs:document
</xs:annotation>
```

New — Define a new code template.

You can define a code template for a specific type of editor or for all editor types.

Edit — Edit the selected code template.

Duplicate — Duplicate the selected code template.

Delete — Delete the selected code template.

Import — Import a file with code templates.

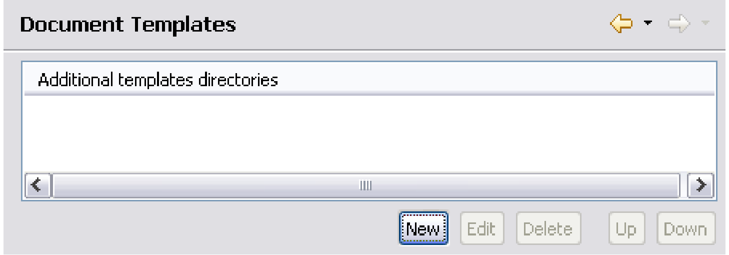
Export — Export a file with code templates.

Document Templates

The user can add template files in the `templates` folder of the <oXygen/> install directory. Directories to be scanned for additional templates can also be specified in the Document Templates option page.

The Document Templates preferences panel is opened from menu Window → Preferences → oXygen+Editor+Templates+Document Templates

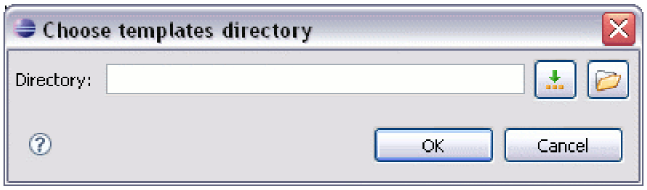
Figure 19.30. Document Templates preferences panel



Document Templates

Additional templates directories

Figure 19.31. Document Templates input dialog



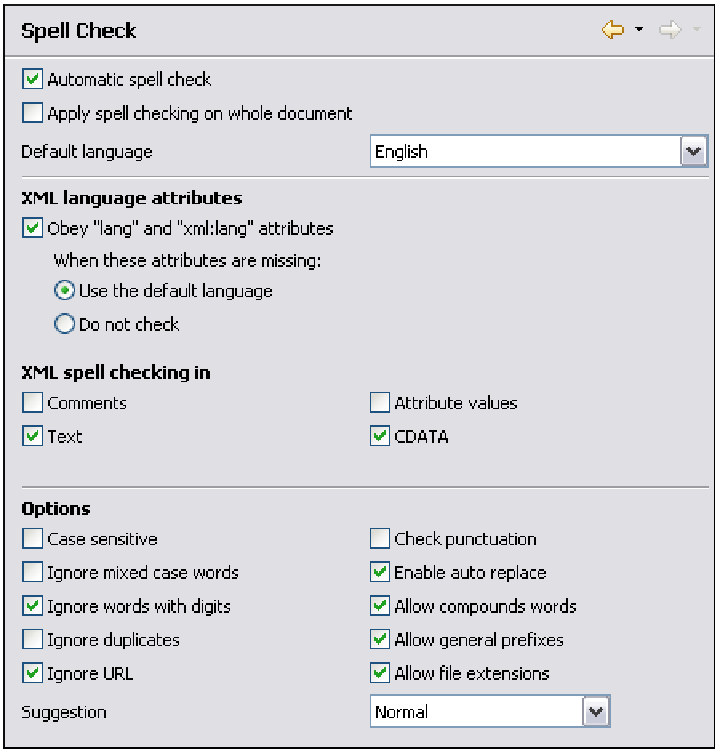
Choose templates directory

Directory:

Spell Check

The Spell Check preferences panel is opened from menu Window → Preferences → oXygen+Editor+Spell Check

Figure 19.32. Spell check preferences panel



Spell Check

☑ Automatic spell check

□ Apply spell checking on whole document

Default language: English

XML language attributes

☑ Obey "lang" and "xml:lang" attributes

When these attributes are missing:

● Use the default language

○ Do not check

XML spell checking in

□ Comments

☑ Text

□ Attribute values

☑ CDATA

Options

□ Case sensitive

□ Ignore mixed case words

☑ Ignore words with digits

□ Ignore duplicates

☑ Ignore URL

□ Check punctuation

☑ Enable auto replace

☑ Allow compounds words

☑ Allow general prefixes

☑ Allow file extensions

Suggestion: Normal

Automatic Spell Check
: When checked, the spell checker is activated. Spell errors will be highlighted as you type.

Apply spell checking on whole document
: When checked, a spell check action will be performed on entire document, highlighting all encountered errors.

Note

On large documents, spell checking the entire content may take a lot of time.

Default language
: The default language combo allows you to choose the language used by default.

XML language attributes
: Options in this subsection control the way the attributes *lang* and *xml:lang* change the language used for check spelling.

- **Obey "lang" and "xml:lang" attributes** - when checked the value of these attributes is used as check spelling language inside the content of the element where they are present.
- **When these attributes are missing** the language used is controlled by the two radio buttons. The two options are to *Use the default language* or *Do not check* the spelling.

XML spell checking in
: These options allow the user to specify if the spell checker will be enabled inside Comments, Attribute values, Text and CDATA sections.

Case sensitive
When checked, operations ignore capitalization errors.

Ignore mixed case words
When checked, operations do not check words containing case mixing (e.g. "SpellChecker").

Ignore words with digits
When checked, the Spell Checker does not check words containing digits (e.g. "b2b").

Ignore Duplicates
When checked, the Spell Checker does not signal two successive identical words as an error.

Ignore URL
When checked, ignores words looking like URL or file names (e.g. "www.oxygenxml.com" or "c:\boot.ini") .

Check punctuation
When checked, punctuation checking is enabled: misplaced white space and wrong sequences, like a dot following a comma, are detected.

Enable auto replace
Enables the "Replace Always" feature.

Allow compounds words
When checked, all words formed by concatenating two legal words with an hyphen are accepted. If the language allows it, two words concatenated without hyphen are also accepted.

Allow general prefixes
When checked, a word formed by concatenating a registered prefix and a legal word is accepted. For example if "mini-" is a registered prefix, accepts "mini-computer".

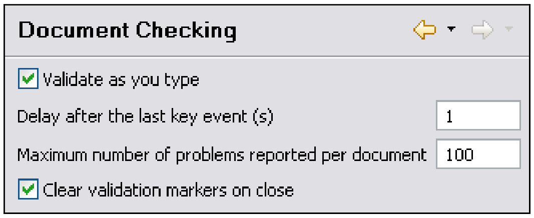
Allow file extensions
When checked, accepts any word ending with registered file extensions (e.g. "myfile.txt", "index.html" etc.).

Suggestion
This option indicates the type of spell checker accuracy, which may be: "Favour speed over quality", "Normal" and "Favour quality over speed".

Document Checking

The Document Checking preferences panel is opened from menu Window → Preferences → oXygen+Editor+Document Checking

Figure 19.33. Document Checking preferences panel



Document Checking

☑ Validate as you type

Delay after the last key event (s) 1

Maximum number of problems reported per document 100

☑ Clear validation markers on close

Validate as you type
Validation of edited document is executed as the document is modified by editing in <oXygen/>.

Delay after the last key event (s)
The period of keyboard inactivity which starts a new validation (in seconds).

Maximum number of errors reported per document
If there are many validation errors the process of marking them in the document is long. You should limit the maximum number of reported errors with this setting to keep the time for error marking short

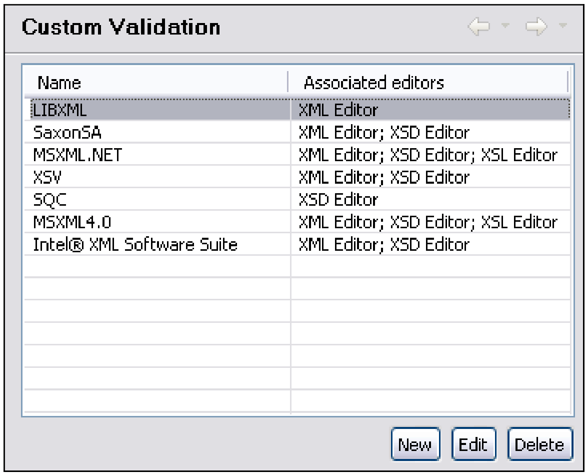
Clear validation markers on close

When a document edited with the <oXygen/> plugin is closed all the error markers added in the Problems view for the validation errors obtained for that document are removed.

Custom Validation

The Custom Validation preferences panel is opened from menu Window → Preferences → oXygen+Editor+Custom Validation

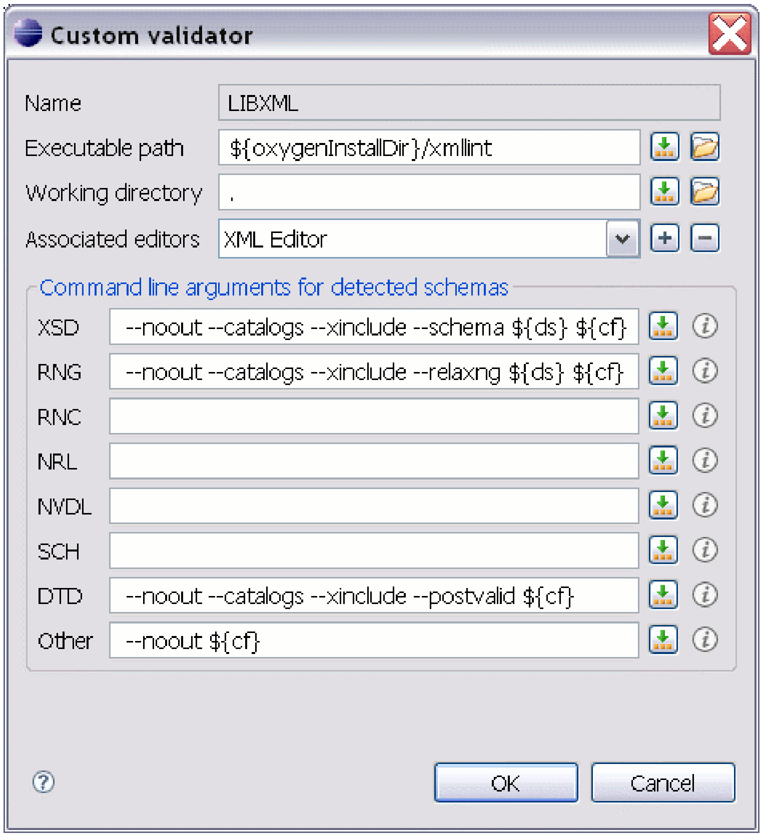
Figure 19.34. Custom Validation preferences panel



Name	Associated editors
LIBXML	XML Editor
SaxonSA	XML Editor; XSD Editor
MSXML.NET	XML Editor; XSD Editor; XSL Editor
XSV	XML Editor; XSD Editor
SQC	XSD Editor
MSXML4.0	XML Editor; XSD Editor; XSL Editor
Intel® XML Software Suite	XML Editor; XSD Editor

If you want to add a new custom validation tool or edit the properties of an exiting one you can use the Custom Validator dialog displayed by pressing New or Edit buttons.

Figure 19.35. Custom validator dialog



Name	LIBXML
Executable path	${oxygenInstallDir}/xmllint
Working directory	.
Associated editors	XML Editor

Command line arguments for detected schemas

XSD	--noout --catalogs --xinclude --schema ${ds} ${cf}
RNG	--noout --catalogs --xinclude --relaxng ${ds} ${cf}
RNC	
NRL	
NVDL	
SCH	
DTD	--noout --catalogs --xinclude --postvalid ${cf}
Other	--noout ${cf}

Name

The name of the custom validation tool displayed in the External Validation toolbar

Executable path

The path to the executable file of the external validation tool. You can insert here editor variables like `${homeDir}`, `${pd}`, etc.

Working directory

The working directory of the external validation tool. The following editor variables can be used:

`${homeDir}`	The path to user home directory
`${pd}`	Project directory
`${oxygenInstallDir}`	<oXygen/> installation directory

Associated editors

The editors which can perform validation with the external tool.

Command line arguments for detected schemas

Command line arguments used to validate the current edited file against different types of schema (W3C XML Schema, Relax NG full syntax, Relax NG compact syntax, Namespace Routing Language, Schematron, DTD, other schema type). The arguments can include any custom switch (like -rng) and the editor variables:

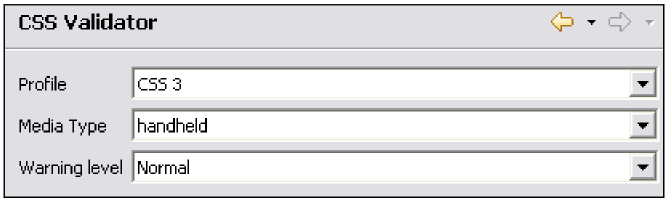
`${cf}`	The path of the currently edited file
`${cfu}`	Path of current file (URL)

`${ds}`	The path of detected schema file
`${dsu}`	The path of detected schema file (URL)

CSS Validator

The CSS Validator preferences panel is opened from menu Window → Preferences → oXygen+CSS Validator

Figure 19.36. CSS Validator preferences panel



CSS Validator

Profile: CSS 3

Media Type: handheld

Warning level: Normal

Profile — Choose one of the available validation profiles: CSS 1, CSS 2, CSS 2.1, CSS 3, SVG, SVG Basic, SVG Tiny, Mobile, TV Profile, ATSC TV Profile

Media Type — Choose one of the available mediums: all, aural, braille, embossed, handheld, print, projection, screen

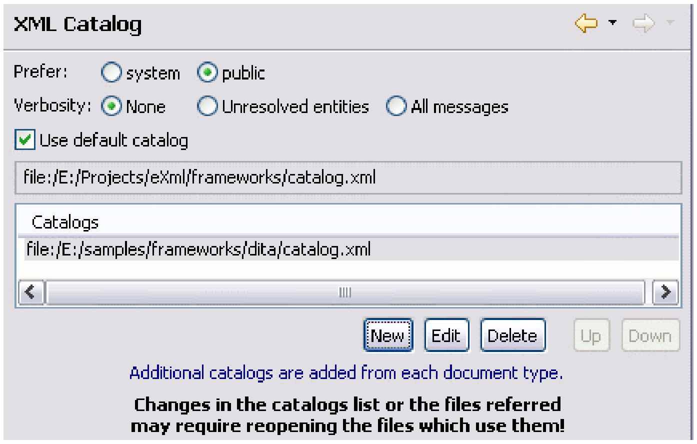
Warning Level — Set the minimum severity level for reported validation warnings. It is one of: all, normal, most important, no warnings.

XML

XML Catalog

The XML Catalog preferences panel is opened from menu Window → Preferences → oXygen+XML+XML Catalog

Figure 19.37. The XML Catalog preferences panel



XML Catalog

Prefer: ○ system ● public

Verbosity: ● None ○ Unresolved entities ○ All messages

☑ Use default catalog

file:/E:/Projects/eXml/frameworks/catalog.xml

Catalogs

file:/E:/samples/frameworks/dita/catalog.xml

Additional catalogs are added from each document type.

Changes in the catalogs list or the files referred may require reopening the files which use them!

The Prefer option is used to specify if <oXygen/> will try to resolve first the PUBLIC or SYSTEM reference using the specified XML catalogs. If a PUBLIC reference is not mapped in any of the catalogs then a SYSTEM reference is looked up.

When using catalogs it is sometimes useful to see what catalog files are parsed, if they are valid or not, and what identifiers are resolved by the catalogs. The Verbosity option selects the detail level of such messages of the catalog resolver that will be displayed in the *Catalogs* view at the bottom of the window:

None
: No message is displayed by the catalog resolver when it tries to resolve a URI reference with the XML catalogs set in the application.

Unresolved entities
: Only the messages that track the failed attempts to resolve URI references are displayed.

All messages
: The messages of both failed attempts and successful ones are displayed.

If the Use default catalog option is checked the first XML catalog which <oXygen/> will use to resolve system IDs at document validation and URI references at document transformation will be a default built-in catalog which maps such references to the built-in local copies of the local DocBook and TEI frameworks and the schemas for XHTML, SVG and JSP documents.

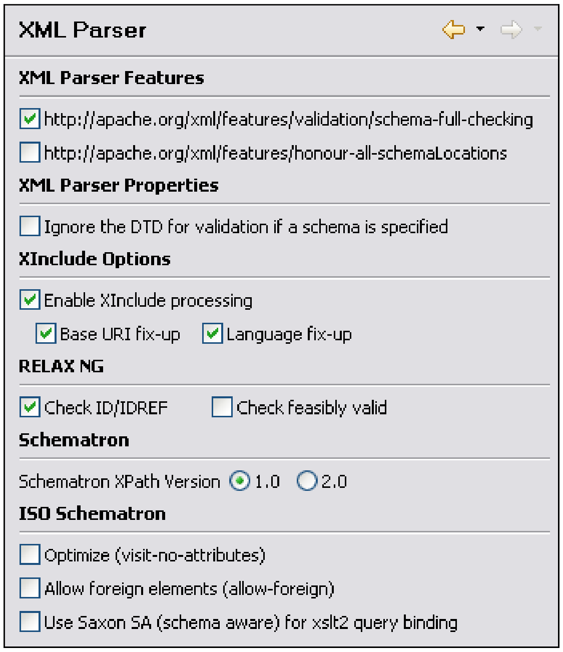
You can also add or configure catalogs for each of the defined document types from Document Type Association preferences page.

When you add/delete or edit an XML catalog to/from the list you must sometimes reopen the current edited files which use the modified catalog so that the changes take full effect.

XML Parser

The XML Parser preferences panel is opened from menu Window → Preferences → oXygen+XML+XML Parser

Figure 19.38. The XML Parser preferences panel



XML Parser

XML Parser Features

☑ http://apache.org/xml/features/validation/schema-full-checking
□ http://apache.org/xml/features/honour-all-schemaLocations

XML Parser Properties

□ Ignore the DTD for validation if a schema is specified

XInclude Options

☑ Enable XInclude processing
☑ Base URI fix-up ☑ Language fix-up

RELAX NG

☑ Check ID/IDREF □ Check feasibly valid

Schematron

Schematron XPath Version ● 1.0 ○ 2.0

ISO Schematron

□ Optimize (visit-no-attributes)
□ Allow foreign elements (allow-foreign)
□ Use Saxon SA (schema aware) for xslt2 query binding

http://apache.org/xml/features/validation/schema-full-checking
: This option sets the 'schema-full-checking' feature to true.

http://apache.org/xml/features/honour-all-schema-location
: This option sets the 'honour-all-schema-location' feature to true. This means all the schemas that are imported for a specific namespace are used to compose the

validation model. If this is false, only the first schema import is taken into account.

Ignore the DTD for validation if a schema is specified

This option forces validation against a referred schema (XML Schema, Relax NG schema, Schematron schema) even if the document includes also a DTD declaration. It is useful when the DTD declaration is used to declare entities and the schema reference is used for validation.

Enable XInclude processing

Enable XInclude processing - if checked the XInclude support in <oXygen/> is turned on.

Base URI fix-up

[Xerces XML Parser documentation:] According to the specification for XInclude, processors must add an xml:base attribute to elements included from locations with a different base URI. Without these attributes, the resulting infoset information would be incorrect.

Unfortunately, these attributes make XInclude processing not transparent to Schema validation.

One solution to this is to modify your schema to allow xml:base attributes to appear on elements that might be included from different base URIs.

If the addition of xml:base and/or xml:lang is undesired by your application, you can disable base URI fix-up.

Language fix-up

[Xerces XML Parser documentation:]The processor will preserve language information on a top-level included element by adding an xml:lang attribute if its include parent has a different [language] property.

If the addition of xml:lang is undesired by your application, you can disable the Language fix-up.

Check ID/IDREF

Checks the ID/IDREF matches when the Relax NG document is validated.

Check feasibly valid

Checks the Relax NG to be feasibly valid when this document is validated.

Schematron XPath Version

1.0 - Allows XSLT 1.0 expressions for Schematron 1.5 assertion tests.

2.0 - Allows XSLT 2.0 expressions for Schematron 1.5 assertion tests.

Optimize (visit-no-attributes)

If your ISO Schematron assertion tests do not contain the attributes axis you should check this option for faster ISO Schematron validation.

Allow foreign elements (allow-foreign)

Enable support for allow-foreign on ISO Schematron. Used to pass non-Schematron elements to the generated stylesheet.

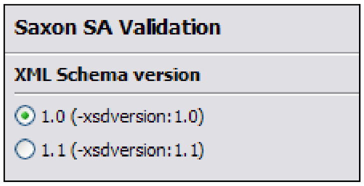
Use Saxon SA (schema aware) for xslt2 query binding

If checked, Saxon SA will be used for xslt2 query binding.

Saxon SA Validation

The Saxon SA Validation preferences panel is opened from menu Window → Preferences → oXygen+XML+XML Parser+Saxon SA Validation

Figure 19.39. The Saxon SA preferences panel



Saxon SA Validation

XML Schema version

● 1.0 (-xsdversion:1.0)

○ 1.1 (-xsdversion:1.1)

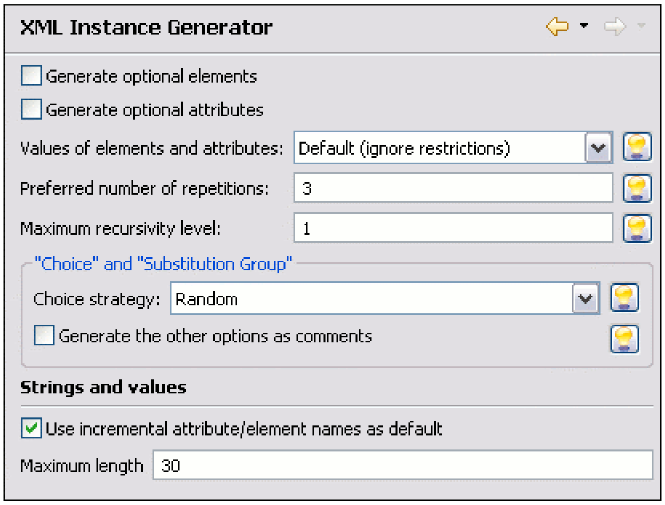
XML Schema version 1.0
: The validation of XML Schema schemas is done according to the W3C XML Schema 1.0 specification.

XML Schema version 1.1
: The validation of XML Schema schemas is done according to the W3C XML Schema 1.1 specification.

XML Instances Generator

The XML Instances Generator preferences panel is opened from menu Window → Preferences → oXygen+XML+XML Instances Generator

Figure 19.40. The XML Instances Generator preferences panel



XML Instance Generator

□ Generate optional elements

□ Generate optional attributes

Values of elements and attributes: Default (ignore restrictions)

Preferred number of repetitions: 3

Maximum recursivity level: 1

"Choice" and "Substitution Group"

Choice strategy: Random

□ Generate the other options as comments

Strings and values

☑ Use incremental attribute/element names as default

Maximum length 30

Generate optional elements
: If checked the elements declared optional in the schema will be generated in the XML instance

Generate optional attributes
: If checked the attributes declared optional in the schema will be generated in the XML instance

Values of elements and attributes
: Specifies what values are generated in elements and attributes of the XML instance. It can have one of the values: None (no values for elements and attributes), Default (the value is like the element name or attribute name), Random (a random value).

Preferred number of repetitions
: The number of repetitions for an element that has a big value of the maxOccurs attribute.

Maximum recursivity level
: For recursive type definitions this parameter specifies the number of levels of recursive elements inserted in the parent element with the same name.

Choice strategy — For choice element models specifies what choice will be generated in the XML instance. It can be First (the first choice is generated) or Random (a random choice is generated).

Generate the other options as comments — If checked the other options of the choice element model which are not selected will be generated inside a comment in the XML instance.

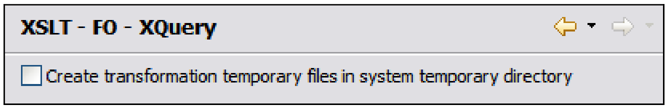
Use incremental attribute/element names as default — If checked the value of an element/attribute is like the name of that element/attribute. For example the values of a elements are a1, a2, a3, etc. If not checked the value is the name of the type of that element /attribute, for example string, decimal, etc.

Maximum length — The maximum length of string values generated for elements and attributes.

XSLT/FO/XQuery

The XSLT/FO/XQuery preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery

Figure 19.41. The XSLT/FO/XQuery preferences panel



XSLT - FO - XQuery

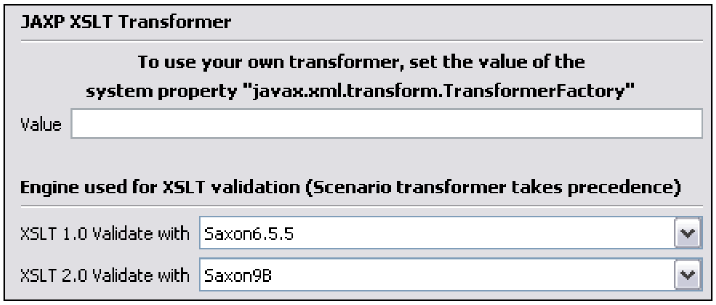
□ Create transformation temporary files in system temporary directory

Check the option *Create transformation temporary files in system temporary directory* when creating transformation temporary files in the same folder as the source of the transformation breaks the transformation, for example the transformation processes all the files located in the same folder as the source of the transformation, which will include the temporary files, which you probably do not want.

XSLT

The XSLT preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+XSLT

Figure 19.42. The XSLT preferences panel



JAXP XSLT Transformer

To use your own transformer, set the value of the system property "javax.xml.transform.TransformerFactory"

Value

Engine used for XSLT validation (Scenario transformer takes precedence)

XSLT 1.0 Validate with: Saxon6.5.5

XSLT 2.0 Validate with: Saxon9B

If you want to use an XSLT transformer different than the ones that ship with <oXygen/> namely Apache Xalan and Saxon all you have to do is to specify the name of the transformer's factory class which <oXygen/> will set as the value of the Java property "javax.xml.transform.TransformerFactory". To perform an XSLT transformation with Saxon 7 for instance you have to place the Saxon 7 jar file in the <oXygen/> libraries directory (the *lib* subdirectory of the installation directory), set "net.sf.saxon.TransformerFactoryImpl" as the property value and select JAXP as the XSLT processor in the transformation scenario associated to the transformed XML document.

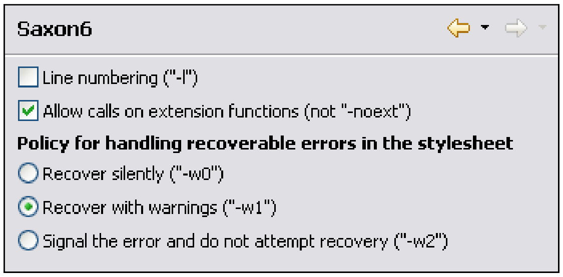
Value — Allows the user to enter the name of the transformer factory Java class.

XSLT 1.0 Validate with	Allows the user to set the XSLT Engine used for validation of XSL 1.0 documents.
XSLT 2.0 Validate with	Allows the user to set the XSLT Engine used for validation of XSL 2.0 documents.

Saxon6

The Saxon 6 preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+XSLT+Saxon+Saxon 6

Figure 19.43. The Saxon 6 XSLT preferences panel



Saxon6

□ Line numbering ("-l")

☑ Allow calls on extension functions (not "-noext")

Policy for handling recoverable errors in the stylesheet

○ Recover silently ("-w0")

● Recover with warnings ("-w1")

○ Signal the error and do not attempt recovery ("-w2")

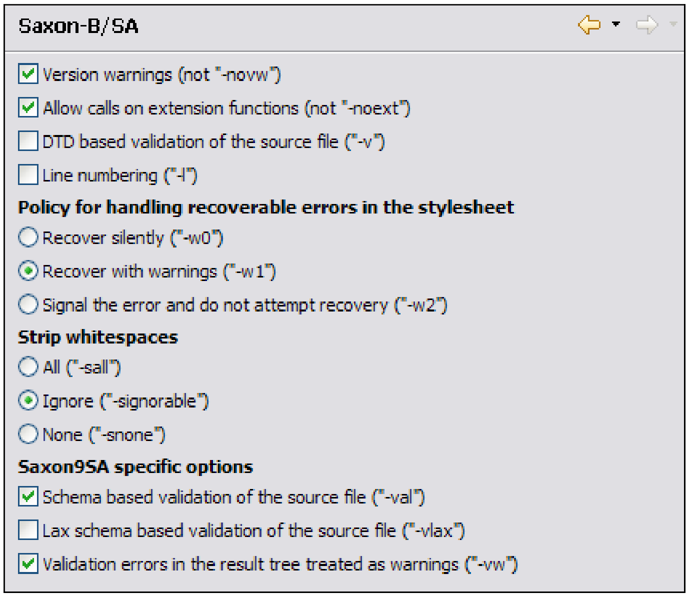
- Line numbering: If checked line numbers are maintained for the source document.
- Allow calls on extension functions: If checked external functions called is allowed. Not checking this is recommended in an environment where untrusted stylesheets may be executed. Also disables user-defined extension elements, together with the writing of multiple output files, all of which carry similar security risks.
- Policy for handling recoverable errors in the stylesheet: Allows the user to choose how dynamic errors will be handled. Either one of the following options can be selected: recover silently, recover with warnings or signal the error and do not attempt recovery.

Saxon-B/SA

The Saxon-B/SA preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+XSLT+Saxon+Saxon-B/SA

The XSLT options which can be configured for the Saxon 9 transformer (both the Basic and the Schema Aware versions) are:

Figure 19.44. The Saxon-B/SA XSLT preferences panel



Saxon-B/SA

☑ Version warnings (not "-novw")

☑ Allow calls on extension functions (not "-noext")

□ DTD based validation of the source file ("-v")

□ Line numbering ("-l")

Policy for handling recoverable errors in the stylesheet

○ Recover silently ("-w0")

● Recover with warnings ("-w1")

○ Signal the error and do not attempt recovery ("-w2")

Strip whitespaces

○ All ("-sall")

● Ignore ("-signorable")

○ None ("-snone")

Saxon9SA specific options

☑ Schema based validation of the source file ("-val")

□ Lax schema based validation of the source file ("-vlax")

☑ Validation errors in the result tree treated as warnings ("-vw")

- Version warnings: If checked a warning will be output when running an XSLT 2.0 processor against an XSLT 1.0 stylesheet. The XSLT specification requires this to be done by default.
- Allow calls on extension functions: If checked external functions called is allowed. Not checking this is recommended in an environment where untrusted stylesheets may be executed. Also disables user-defined extension elements, together with the writing of multiple output files, all of which carry similar security risks.
- DTD based validation of the source file: If checked XML source documents are validated against their DTD.
- Line numbering: If checked line numbers are maintained for the source document.
- Policy for handling recoverable errors in the stylesheet: Allows the user to choose how dynamic errors will be handled. Either one of the following options can be selected: recover silently, recover with warnings or signal the error and do not attempt recovery.
- Strip whitespaces feature can be one of the three options: All, Ignorable, None.

  All — strips all whitespace text nodes from source documents before any further processing, regardless of any xsl:strip-space declarations in the stylesheet, or any xml:space attributes in the source document.

  Ignorable — strips all ignorable whitespace text nodes from source documents before any further processing, regardless of any xsl:strip-space declarations in the stylesheet, or any xml:space attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.

  None — strips no whitespace before further processing. (However, whitespace will still be stripped if this is specified in the stylesheet using xsl:strip-space).

Saxon9SA specific options

- Schema based validation of the source file: This determines whether source documents should be parsed with schema-validation enabled.

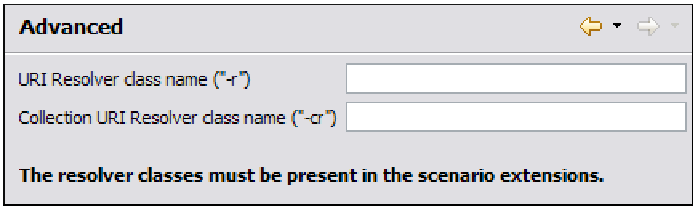
- Lax schema based validation of the source file: This determines whether source documents should be parsed with schema-validation enabled.
- Validation errors in the result tree treated as warnings: If checked, all validation errors are treated as warnings, otherwise they are treated as fatal.

Saxon-B/SA Advanced options

The Saxon-B/SA Advanced preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+XSLT+Saxon+Saxon-B/SA+Advanced

The advanced XSLT options which can be configured for the Saxon 9 transformer (both the Basic and the Schema Aware versions) are:

Figure 19.45. The Saxon-B/SA XSLT Advanced preferences panel



Advanced

URI Resolver class name ("-r")

Collection URI Resolver class name ("-cr")

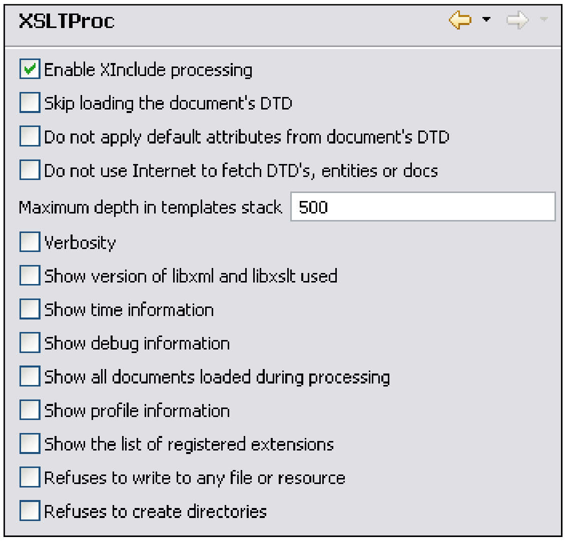
The resolver classes must be present in the scenario extensions.

- URI Resolver class name: Allows the user to specify a custom implementation for the URI resolver used by the XSLT Saxon 9 transformer ("-r" option when run from the command line). The class name must be fully specified and the corresponding jar or class extension must be configured from the dialog for configuring the XSLT extension for the particular scenario
- Collection URI Resolver class name: Allows the user to specify a custom implementation for the Collection URI resolver used by the XSLT Saxon 9 transformer ("-cr" option when run from the command line). The class name must be fully specified and the corresponding jar or class extension must be configured from the dialog for configuring the XSLT extension for the particular scenario

XSLTProc

The XSLTProc preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+XSLT+XSLTProc

Figure 19.46. The XSLTProc preferences panel



XSLTProc

☑ Enable XInclude processing

□ Skip loading the document's DTD

□ Do not apply default attributes from document's DTD

□ Do not use Internet to fetch DTD's, entities or docs

Maximum depth in templates stack 500

□ Verbosity

□ Show version of libxml and libxslt used

□ Show time information

□ Show debug information

□ Show all documents loaded during processing

□ Show profile information

□ Show the list of registered extensions

□ Refuses to write to any file or resource

□ Refuses to create directories

The options of the XSLTProc processor are the same as the ones available in the command line for the XSLTProc processor:

Enable XInclude processing
If checked XInclude references will be resolved when XSLTProc is used as transformer in the transformation scenario.

Skip loading the document's DTD
If checked the DTD specified in the DOCTYPE declaration will not be loaded.

Do not apply default attributes from document's DTD
If checked the default attributes declared in the DTD and not specified in the document are not included in the transformed document.

Do not use Internet to fetch DTD's, entities or docs
If checked the remote references to DTD's and entities are not followed.

Maximum depth in templates stack
If the limit of maximum templates is reached the transformation ends with an error.

Verbosity
If checked the transformation will output detailed status messages about the transformation process in the Warnings view.

Show version of libxml and libxslt used
If checked <oXygen/> will display in the Warnings view the version of the libxml and libxslt libraries invoked by XSLTProc.

Show time information
If checked the Warnings view will display the time necessary for running the transformation.

Show debug information
If checked the Warnings view will display debug information about what templates are matched, parameter values, etc.

Show all documents loaded during processing
If checked <oXygen/> will display in the Warnings view the URL of all the files loaded during transformation.

Show profile information
If checked <oXygen/> will display in the Warnings view a table with all the matched templates, and for each template: the match XPath expression, template name, number of template modes, number of calls, execution time.

Show the list of registered extensions

If checked <oXygen/> will display in the Warnings view a list with all the registered extension functions, extension elements and extension modules.

Refuses to write to any file or resource

If checked the XSLTProc processor will not write any part of the transformation result to an external file on disk. If such an operation is requested by the processed XSLT stylesheet the transformation ends with a runtime error.

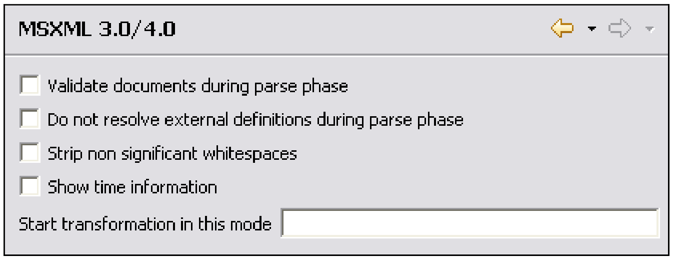
Refuses to create directories

If checked the XSLTProc processor will not create any directory during the transformation process. If such an operation is requested by the processed XSLT stylesheet the transformation ends with a runtime error.

MSXML

The MSXML preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+XSLT+MSXML

Figure 19.47. The MSXML preferences panel



MSXML 3.0/4.0

□ Validate documents during parse phase

□ Do not resolve external definitions during parse phase

□ Strip non significant whitespaces

□ Show time information

Start transformation in this mode

The options of the MSXML 3.0 and 4.0 processors are the same as the ones available in the command line for the MSXML processors: [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnxml/html/msxsl.asp]

Validate documents during parse phase

If checked and either the source or style sheet document has a DTD or schema against which its content should be checked, validation is performed.

Do not resolve external definitions during parse phase

By default, MSXSL instructs the parser to resolve external definitions such as document type definition (DTD), external subsets or external entity references when parsing the source and style sheet documents. If this option is checked the resolution is disabled.

Strip non-significant whitespaces

If checked strip non-significant white space from the input XML document during the load phase. Enabling this option can lower memory usage and improve transformation performance while, in most cases, creating equivalent output.

Show time information

If checked the relative speed of various transformation steps can be measured: time to load, parse, and build the input document; time to load, parse, and build the style sheet document; time to compile the style sheet in preparation for the transformation; time to execute the style sheet.

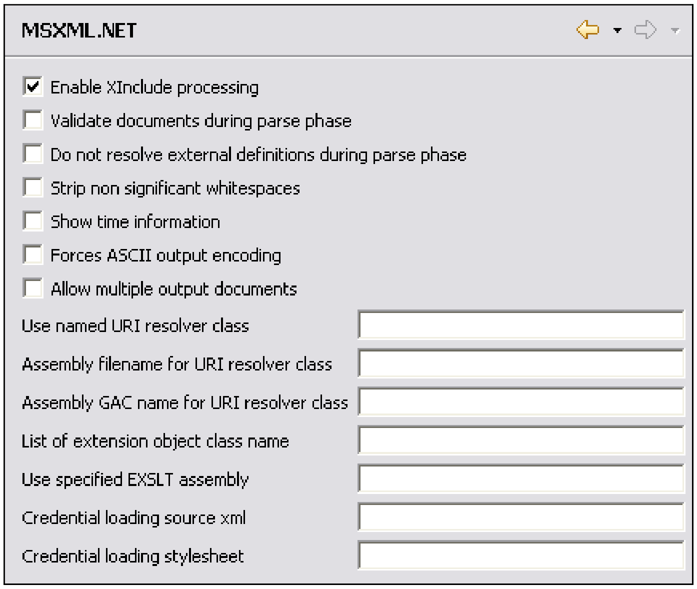
Start transformation in this mode

Although style sheet execution usually begins in the empty mode, this default may be changed by specifying another mode. Changing the start mode allows execution to jump directly to an alternate group of templates.

MSXML.NET

The MSXML.NET preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+XSLT+MSXML.NET

Figure 19.48. The MSXML.NET preferences panel



MSXML.NET

☑ Enable XInclude processing

☐ Validate documents during parse phase

☐ Do not resolve external definitions during parse phase

☐ Strip non significant whitespaces

☐ Show time information

☐ Forces ASCII output encoding

☐ Allow multiple output documents

Use named URI resolver class

Assembly filename for URI resolver class

Assembly GAC name for URI resolver class

List of extension object class name

Use specified EXSLT assembly

Credential loading source xml

Credential loading stylesheet

The options of the MSXML.NET processor are the same as the ones available in the command line for the MSXML.NET processor: [http://www.xmllab.net/Products/nxslt/tabid/62/Default.aspx]

Enable XInclude processing

If checked XInclude references will be resolved when MSXML.NET is used as transformer in the transformation scenario.

Validate documents during parse phase

If checked and either the source or style sheet document has a DTD or schema against which its content should be checked, validation is performed.

Do not resolve external definitions during parse phase

By default MSXML.NET resolves external definitions such as DTD external subsets or external entity references when parsing source XML document and stylesheet document. Using this option you can disable this behaviour. (Note, that it may affect also the validation process.)

Strip non-significant whitespaces

If checked strip non-significant white space from the input XML document during the load phase. Enabling this option can lower memory usage and improve transformation performance while, in most cases, creating equivalent output.

Show time information

If checked the relative speed of various transformation steps can be measured: time to load, parse, and build the input document; time to load, parse, and build the style sheet document; time to compile the style sheet in preparation for the transformation; time to execute the style sheet.

Forces ASCII output encoding

There is a known problem with .NET 1.X XSLT processor (System.Xml.Xsl.XslTransform class) - it doesn't support escaping of characters as XML character references when they cannot be represented in the output encoding. That means that when you output a character that cannot be represented in output encoding, it will be outputted as '?'. Usually this happens when output encoding is set to ASCII. With this option checked the output is forced to be ASCII encoded and all non-ASCII characters get escaped as XML character references (&#nnnn; form).

Allow multiple output documents

This option allows to create multiple result documents using the `exsl:document` extension element. [http://www.exslt.org/exsl/elements/document/index.html]

Use named URI resolver class

This option allows to specify a custom URI resolver class to resolve URI references in xsl:import/xsl:include instructions (during XSLT stylesheet loading phase) and in document() function (during XSL transformation phase).

Assembly file name for URI resolver class

The previous option specifies partially or fully qualified URI resolver class name, e.g. `Acme.Resolvers.CacheResolver`. Such name requires additional assembly specification using this option or the next option, but fully qualified class name (which always includes an assembly specifier) is all-sufficient. See MSDN for more info about fully qualified class names. [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconspecifyingfullyqualifiedtypenames.asp] This option specifies a file name of the assembly, where the specified resolver class can be found.

Assembly GAC name for URI resolver class

This option specifies partially or fully qualified name of the assembly in the global assembly cache [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconglobalassemblycache.asp] (GAC), where the specified resolver class can be found. See MSDN for more info about partial assembly names. [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconpartialassemblyreferences.asp] Also see the previous option.

List of extension object class names

This option allows to specify extension object [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconxsltargumentlistforstylesheetparametersextensionobjects.asp] classes, whose public methods then can be used as extension functions in an XSLT stylesheet. It is a comma-separated list of namespace-qualified extension object class names. Each class name must be bound to a namespace URI using prefixes as when providing XSLT parameters. [http://www.xmllab.net/Products/nxslt/tabid/62/Default.aspx#parameters]

Use specified EXSLT assembly

MSXML.NET supports rich library of the EXSLT [http://www.exslt.org/] and EXSLT.NET [http://www.xmllab.net/exslt] extension functions via embedded or plugged in EXSLT.NET [http://workspaces.gotdotnet.com/exslt] library. EXSLT support is enabled by default and cannot be disabled in this version. If you want to use an external EXSLT.NET implementation instead of a built-in one use this option.

Credential loading source xml

This option allows to specify user credentials to be used when loading XML source documents. The credentials should be provided in the "username:password@domain" format (all parts are optional).

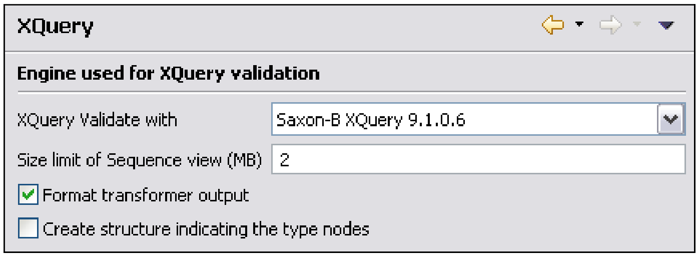
Credential loading stylesheet

This option allows to specify user credentials to be used when loading XSLT stylesheet documents. The credentials should be provided in the "username:password@domain" format (all parts are optional).

XQuery

The XQuery preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+XQuery

Figure 19.49. The XQuery preferences panel



XQuery

Engine used for XQuery validation

XQuery Validate with: Saxon-B XQuery 9.1.0.6

Size limit of Sequence view (MB): 2

☑ Format transformer output

☐ Create structure indicating the type nodes

XQuery validate with

Allows you to select the processor to validate the XQuery. In case you are validating an XQuery file that has an associated validation scenario , <oXygen/> uses the processor specified in the scenario. If no validation scenario is associated, but the file has an associated transformation scenario, the processor specified in the scenario will be used. If the processor does not support validation or if no scenario is associated, then the value from this combo box will be used as validation processor

Size limit of Sequence view (MB)

When the result of an XQuery transformation is set in the transformation scenario as sequence the size of one chunk of the result that is fetched from the database in one step is set in this option.

Size limit of Sequence view (MB)

The limit of the data extract from a database when execute a XQuery in lazy mode. If this limit is exceed you can extract more data from the database by click on "More result available" node from the Sequence view.

Format transformer output

When checked the transformer's output is formatted and indented (pretty printed). Option is ignored if in the transformation scenario you choose *Sequence*(lazy extract data from a database).

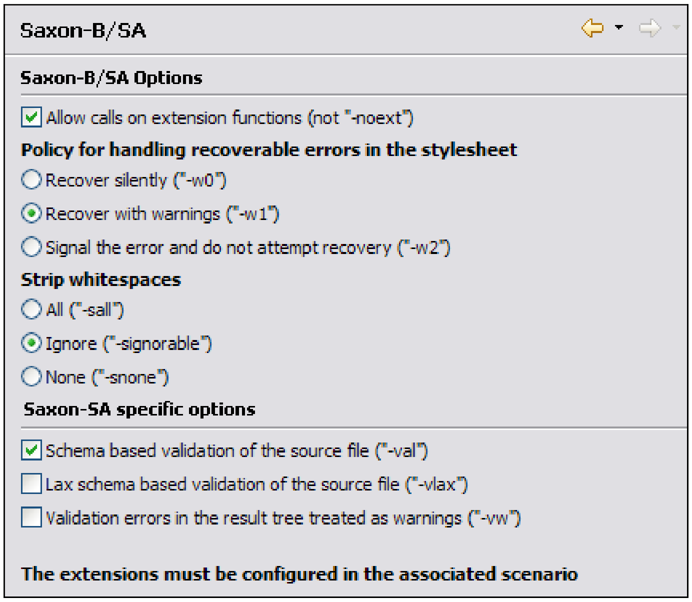
Create structure indicating the type nodes

If checked, <oXygen/> takes the results of a query and creates an XML document containing copies of all items in the sequence, suitably wrapped. Option is ignored if in the transformation scenario you choose *Sequence*(lazy extract data from a database).

Saxon-B/SA

The XQuery/Saxon-B/SA preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+XQuery+Saxon-B/SA

Figure 19.50. The Saxon XQuery preferences panel



Saxon-B/SA

Saxon-B/SA Options

☑ Allow calls on extension functions (not "-noext")

Policy for handling recoverable errors in the stylesheet

○ Recover silently ("-w0")

● Recover with warnings ("-w1")

○ Signal the error and do not attempt recovery ("-w2")

Strip whitespaces

○ All ("-sall")

● Ignore ("-signorable")

○ None ("-snone")

Saxon-SA specific options

☑ Schema based validation of the source file ("-val")

□ Lax schema based validation of the source file ("-vlax")

□ Validation errors in the result tree treated as warnings ("-vw")

The extensions must be configured in the associated scenario

Saxon9 options:

Allow calls on extension functions

If checked external functions called is allowed. Not checking this is recommended in an environment where untrusted stylesheets may be executed. Also disables user-defined extension elements, together with the writing of multiple output files, all of which carry similar security risks.

Policy for handling recoverable errors in the stylesheet

Allows the user to choose how dynamic errors will be handled. Either one of the following options can be selected: recover silently, recover with warnings or signal the error and do not attempt recovery.

Strip whitespaces

Can have one of the three values: All, Ignore, None. *All* - strips all whitespace text nodes from source documents before any further processing, regardless of any xml:space attributes in the source document. *Ignore* - strips all ignorable whitespace text nodes from source documents before any further processing, regardless of any xml:space attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content. *None* - strips no whitespace before further processing.

Saxon9SA specific options:

Schema based validation of the source

This determines whether source documents should be parsed with schema-validation enabled.

Lax schema based validation of the source

This determines whether source documents should be parsed with schema-validation enabled.

Validation errors in the result tree treated as warnings

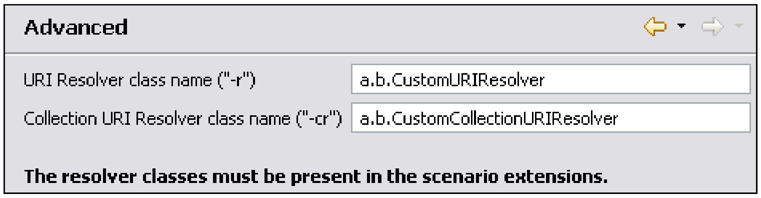
If checked, all validation errors are treated as warnings, otherwise they are treated as fatal.

Saxon-B/SA Advanced options

The XQuery/Saxon-B/SA Advanced preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+XQuery+Saxon-B/SA+Advanced

The advanced XQuery options which can be configured for the Saxon 9 transformer (both the Basic and the Schema Aware versions) are:

Figure 19.51. The Saxon-B/SA XQuery Advanced preferences panel



Advanced

URI Resolver class name ("-r") a.b.CustomURIResolver

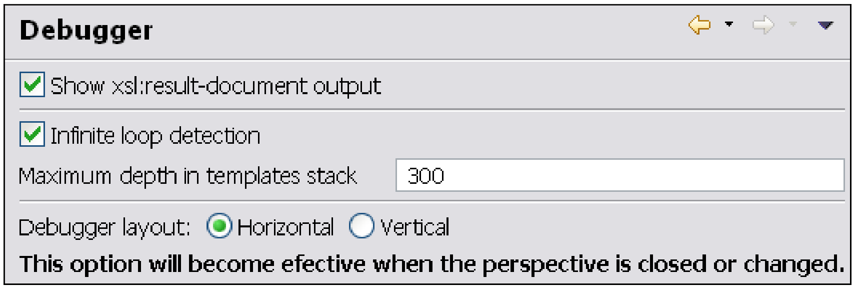
Collection URI Resolver class name ("-cr") a.b.CustomCollectionURIResolver

The resolver classes must be present in the scenario extensions.

- URI Resolver class name: Allows the user to specify a custom implementation for the URI resolver used by the XQuery Saxon 9 transformer ("-r" option when run from the command line). The class name must be fully specified and the corresponding jar or class extension must be configured from the dialog for configuring the XQuery extension for the particular scenario
- Collection URI Resolver class name: Allows the user to specify a custom implementation for the Collection URI resolver used by the XQuery Saxon 9 transformer ("-cr" option when run from the command line). The class name must be fully specified and the corresponding jar or class extension must be configured from the dialog for configuring the XQuery extension for the particular scenario

Debugger

Figure 19.52. The Debugger preferences panel



Debugger

☑ Show xsl:result-document output

☑ Infinite loop detection

Maximum depth in templates stack 300

Debugger layout: ● Horizontal ○ Vertical

This option will become efective when the perspective is closed or changed.

The following settings are available:

Show xsl:result-document output
: If checked, the debugger presents the output of xsl: result-document instruction into the debugger output view.

Infinite loop detection
: Set this option to receive notifications when an infinite loop occurs during transformation.

Maximum depth in templates stack
: How many templates (`<xsl:templates>`) instructions can appear on the current stack. This setting is used by the infinite loop detection.

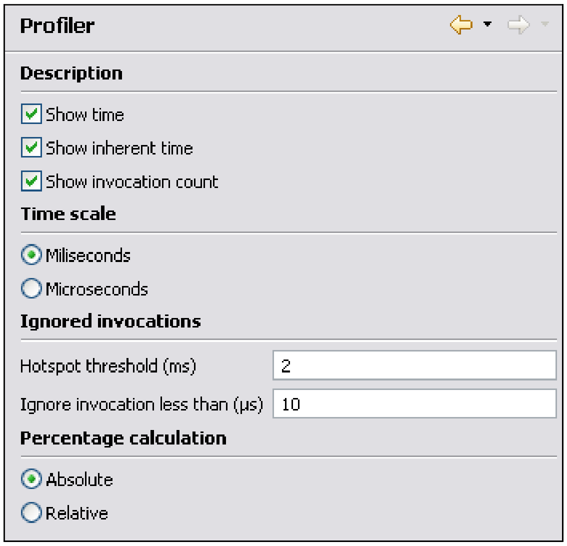
Debugger layout
: A horizontal layout means that the stack of XML editors takes the left half of the editing area and the stack of XSL editors takes the right one. A vertical layout

means that the stack of XML editors takes the upper half of the editing area and the stack of XSL editors takes the lower one.

Profiler

This section explains the settings available for XSLT Profiler mode. To display settings select Window → Preferences → oXygen → XML → XSLT/FO/XQuery+Profiler (see the section called “Debugger”).

Figure 19.53. The Profiler preferences panel



Profiler

Description

☑ Show time

☑ Show inherent time

☑ Show invocation count

Time scale

● Miliseconds

○ Microseconds

Ignored invocations

Hotspot threshold (ms) 2

Ignore invocation less than (µs) 10

Percentage calculation

● Absolute

○ Relative

The following settings are available:

Show time — Show the total time that was spent in the node.

Show inherent time — Show the inherent time that was spent in the node. The inherent time is defined as the total time of a node minus the time of its child nodes.

Show invocation count — Show how many times the node was called in this particular call sequence.

Time scale — The time scale options determine the unit of time measurement, which may be milliseconds (ms) or microseconds (µs).

Hotspot threshold — The threshold below which hot spots are ignored is entered in milliseconds (ms).

Ignore invocation less than — The threshold below which invocations are ignored is entered in microseconds (µs).

Percentage calculation — The percentage base determines against what time span percentages are calculated.

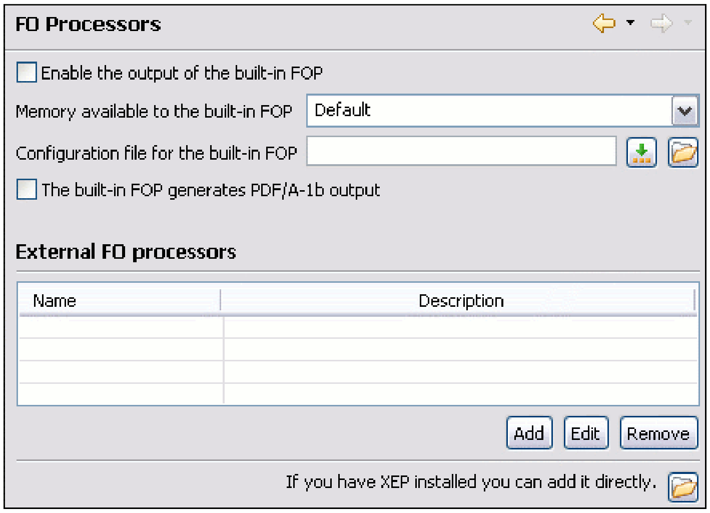
- Absolute: Percentage values show the contribution to the total time.
- Relative: Percentage values show the contribution to the calling node.

FO Processors

Besides the built-in formatting objects processor (Apache FOP) the user can use other external processors. <oXygen/> has implemented an easy way to add RenderX XEP as external FO processor if the user has the XEP installed.

The FO Processors preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+FO Processors

Figure 19.54. The FO Processors preferences panel



FO Processors

□ Enable the output of the built-in FOP

Memory available to the built-in FOP: Default

Configuration file for the built-in FOP:

□ The built-in FOP generates PDF/A-1b output

External FO processors

Name	Description

If you have XEP installed you can add it directly.

Enable the output of the built-in FOP

When checked all FOP output will be displayed in a results pane at the bottom of the editor window including warning messages about FO instructions not supported by FOP.

Memory available to the built-in FOP

If your FOP transformations fail with an "Out of Memory" error select from this combo box a larger value for the amount of memory reserved for FOP transformations.

Configuration file for the built-in FOP

You should specify here the path to a FOP configuration file, necessary for example to render to PDF using a special true type font a document containing Unicode content.

The built-in FOP generates PDF/A-1b output

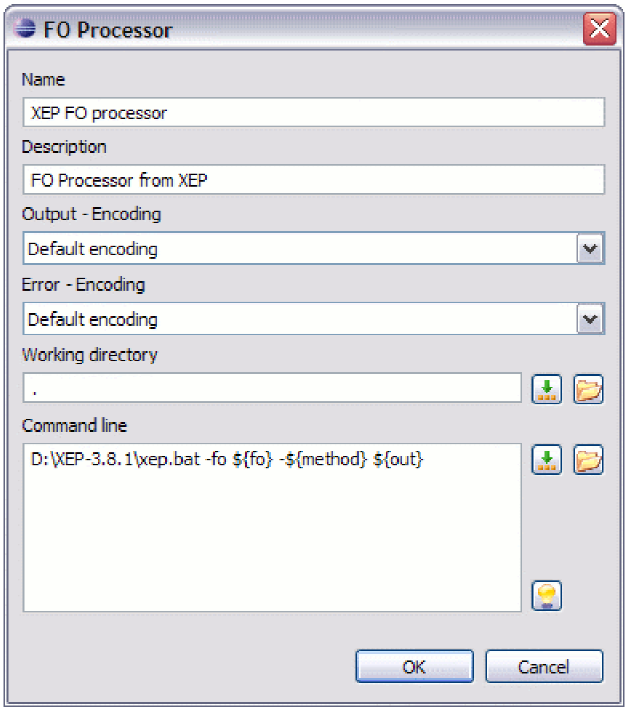
When selected PDF/A-1b output is generated.

> **Note**
>
> All fonts have to be embedded, even the implicit ones. More information about configuring metrics files for the embedded fonts can be found in Add a font to the built-in FOP.

> **Note**
>
> You cannot use the `<filterList>` key in the configuration file. FOP will generate the following error: *The Filter key is prohibited when PDF/A-1 is active*.

The users can configure the external processors for use with <oXygen/> in the following dialog.

Figure 19.55. The external FO processor configuration dialog



Name

XEP FO processor

Description

FO Processor from XEP

Output - Encoding

Default encoding

Error - Encoding

Default encoding

Working directory

.

Command line

```
D:\XEP-3.8.1\xep.bat -fo ${fo} -${method} ${out}
```

Name — The name that will be displayed in the list of available FOP processors on the FOP tab of the Transforming Configuration dialog.

Description — The description of the FO processor displayed in the Preferences->FO Processors option.

Output Encoding — The encoding used for the output stream of the FO processor which will be displayed in a results panel at the bottom of the <oXygen/> window.

Error Encoding — The encoding used for the error stream of the FO processor which will be displayed in a results panel at the bottom of the <oXygen/> window.

Working directory — The directory in which the intermediate and final results of the processing will be stored. Here you can use one of the following editor variables:

`${homeDir}`	The path to user home directory.
`${cfd}`	The path of current file directory. If the current file is not a local file the directory will be the user's Desktop directory.
`${pd}`	The project directory.
`${oxygenInstallDir}`	The <oXygen/> installation directory.

Command line — The command line that will start the FO processor, specific to each processor. Here you can use one of the following editor variables:

${method}	The FOP transformation method (pdf, ps, txt).
${fo}	The input FO file.

${out} The output file.

${pd} The project directory.

${frameworksDir} The path of the `frameworks` subdirectory of the <oXygen/> install directory.

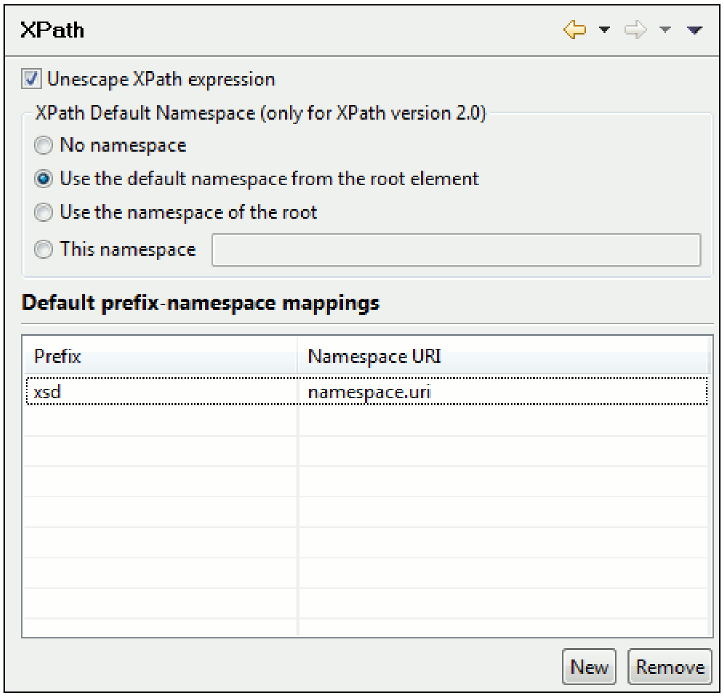
${oxygenInstallDir} The <oXygen/> installation directory.

${ps} The separator which can be used on different operating systems between libraries specified in the class path.

XPath

The XPath preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+XPath

Figure 19.56. The XPath preferences panel



XPath

☑ Unescape XPath expression

XPath Default Namespace (only for XPath version 2.0)

○ No namespace

● Use the default namespace from the root element

○ Use the namespace of the root

○ This namespace

Default prefix-namespace mappings

Prefix	Namespace URI
xsd	namespace.uri

Unescape XPath expression When checked, unescapes the entities found in the XPath expression. For example the expression

```
//varlistentry[starts-with(@os,'&#x73;')]
```

is equivalent with

```
//varlistentry[starts-with(@os,'s')]
```

.

No namespace If checked <oXygen/> will consider unprefixed element names in XPath expressions evaluated in the XPath console as belonging to no namespace.

Use the default namespace from the root element
: If checked <oXygen/> will consider unprefixed element names in XPath expressions evaluated in the XPath console as belonging to the default namespace declared on the root element of the document.

Use the namespace of the root
: If checked <oXygen/> will consider unprefixed element names in XPath expressions evaluated in the XPath console as belonging to the same namespace as the root element of the document.

This namespace
: The user has the possibility to enter here the namespace of the unprefixed elements used in the XPath console

Default prefix-namespace mappings
: Associates prefixes to namespaces. These mappings are useful when applying an XPath in XPath console and you don't have to define these mappings for each document separately.

  The New button creates an editable prefix-namespace mapping.

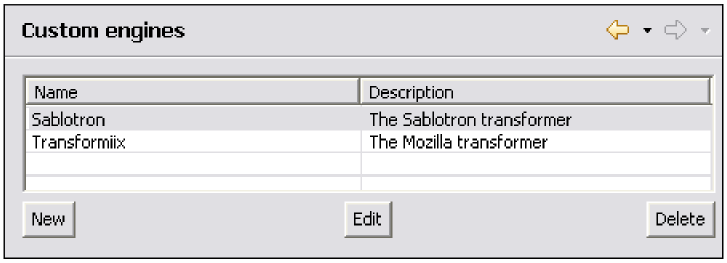
  The Remove button deletes the selected mapping.

Custom Engines

One can configure transformation engines other than the ones which come with the <oXygen/> distribution. Such an external engine can be used for XSLT / XQuery transformations within <oXygen/>, in the Editor perspective, and is available in the list of engines in the dialog for editing transformation scenarios.However it cannot be used in the Debugger perspective.

The Custom Engines preferences panel is opened from menu Window → Preferences → oXygen+XML+XSLT/FO/XQuery+Custom Engines

Figure 19.57. Configuration of custom transformation engines

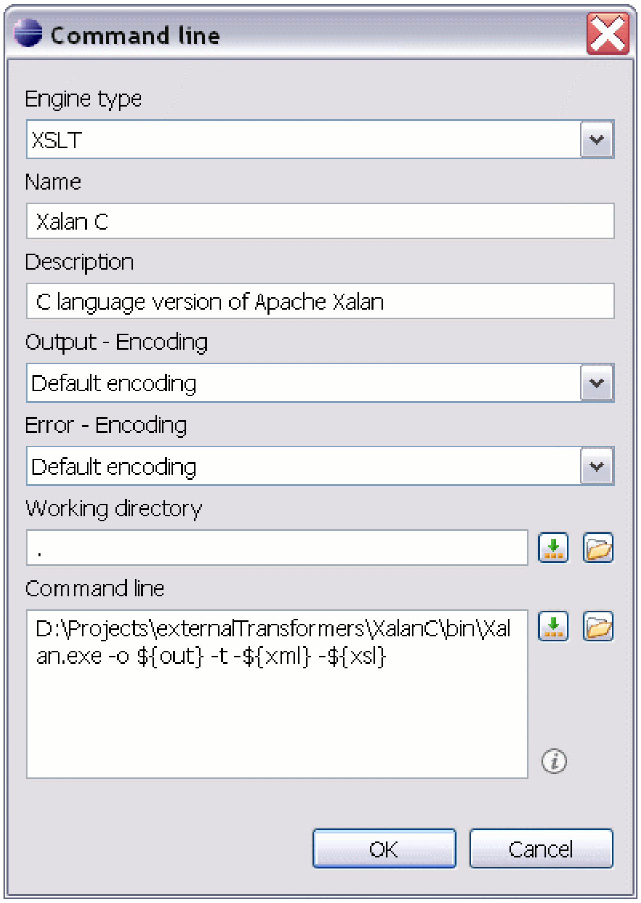


Custom engines

Name	Description
Sablotron	The Sablotron transformer
Transformiix	The Mozilla transformer

The following parameters can be configured for a custom engine:

Figure 19.58. Parameters of a custom transformation engine



Engine type: XSLT

Name: Xalan C

Description: C language version of Apache Xalan

Output - Encoding: Default encoding

Error - Encoding: Default encoding

Working directory: .

Command line:

```
D:\Projects\externalTransformers\XalanC\bin\Xalan.exe -o ${out} -t -${xml} -${xsl}
```

Engine type

Combo box allowing you to choose the transformer type. There are two options: XSLT engines and XQuery engines.

Name

The name of the transformer displayed in the dialog for editing transformation scenarios

Description

Text description of the transformer

Output Encoding

The encoding of the characters sent to the output stream of the transformer

Error Encoding

The encoding of the characters sent to the error stream of the transformer

Working directory

The start directory of the transformer executable program. The following editor variables are available for making the path to the working directory independent of the input XML file:

- `${homeDir}` - the user home directory in the operating system
- `${cfd}` - the path to the directory of the current file
- `${pd}` - the path to the directory of the current project
- `${oxygenInstallDir}` - the <oXygen/> install directory

Command line

The command line that must be executed by <oXygen/> to perform a transformation with the engine. The following editor variables are available for making the items of the command line (the transformer executable, the input files) independent of the input XML file:

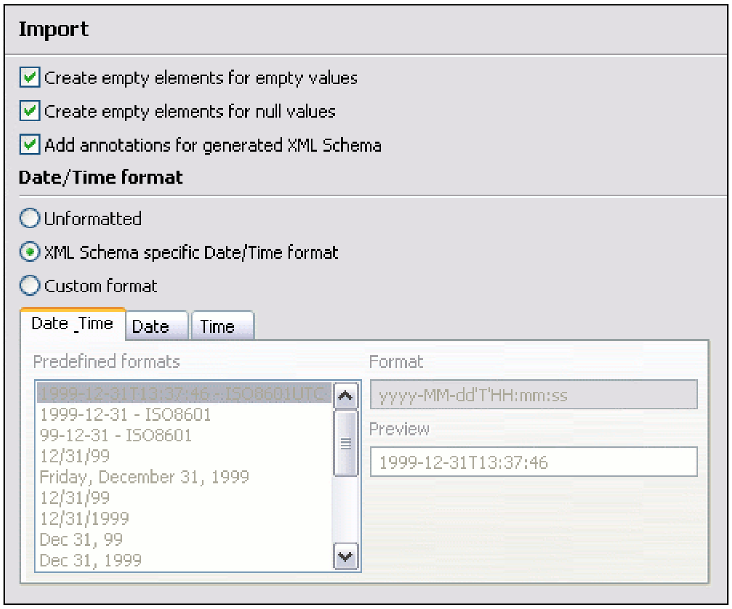
- `${xml}` - the XML input document as a file path
- `${xmlu}` - the XML input document as a URL
- `${xsl}` - the XSL / XQuery input document as a file path
- `${xslu}` - the XSL / XQuery input document as a URL
- `${out}` - the output document as a file path
- `${outu}` - the output document as a URL
- `${ps}` - the separator which can be used on different operating systems between libraries specified in the class path.

Import

The Import preferences panel is opened from menu Window → Preferences → oXygen+XML+Import

Here it is configured how empty values and null values are handled when they are encountered in an import operation.

Figure 19.59. The XML Import preferences panel



Import

☑ Create empty elements for empty values

☑ Create empty elements for null values

☑ Add annotations for generated XML Schema

Date/Time format

○ Unformatted

● XML Schema specific Date/Time format

○ Custom format

Date _Time | Date | Time

Predefined formats

1999-12-31T13:37:46 - ISO8601UTC
1999-12-31 - ISO8601
99-12-31 - ISO8601
12/31/99
Friday, December 31, 1999
12/31/99
12/31/1999
Dec 31, 99
Dec 31, 1999

Format: yyyy-MM-dd'T'HH:mm:ss

Preview: 1999-12-31T13:37:46

Create empty elements for empty values

If this option is enabled an empty value from a database column or from a text file will be imported as an empty element.

Create empty elements for null values

If this option is enabled a null value from a database column will be imported as an empty element.

Add annotations for generated XML Schema

If checked, the generated XML Schema will contain an annotation for each of the imported table's columns. The documentation inside the annotation tag will contain the remarks of the database columns (if available) and also information about the conversion between the column type and the generated XML Schema type.

Date/Time format

The section *Date/Time format* specifies the format used for importing date and time values from Excel spreadsheets or database tables and in the generated XML schemas.

Unformatted

If this option is selected the date and time formats specific to the database will be used for import. When importing data from Excel a string representation of date or time values will be used. The type used in the generated XML Schema will be xs:string.

XML Schema date format

If this option is checked, the XML Schema specific format ISO8601 will be used for imported date/time data (yyyy-MM-dd'T'HH:mm:ss for `datetime`, yyyy-MM-dd for `date` and HH:mm:ss for `time`). The types used in the generated XML Schema will be `xs:datetime`, `xs:date` and `xs:time`.

Custom format

If this is selected, the user can define a custom format for date/time values or choose from the predefined formats. A preview of the values is presented when a format is used. The type used in the generated XML Schema is xs:string.

Date/Time Patterns

Table 19.1. Pattern letters

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am/pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am/pm (0-11)	Number	0
h	Hour in am/pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800

Pattern letters are usually repeated, as their number determines the exact presentation:

- *Text*: If the number of pattern letters is 4 or more, the full form is used; otherwise a short or abbreviated form is used if available.
- *Number*: the number of pattern letters is the minimum number of digits, and shorter numbers are zero-padded to this amount.
- *Year*: If the number of pattern letters is 2, the year is truncated to 2 digits; otherwise it is interpreted as a number.
- *Month*: If the number of pattern letters is 3 or more, the month is interpreted as text; otherwise, it is interpreted as a number.
- *General time zone*: Time zones are interpreted as text if they have names. For time zones representing a GMT offset value, the following syntax is used:

GMTOffsetTimeZone:	GMT Sign Hours : Minutes
Sign:	one of + -
Hours:	Digit - Digit Digit
Minutes:	Digit Digit
Digit:	one of 0 1 2 3 4 5 6 7 8 9

  Hours must be between 0 and 23, and Minutes must be between 00 and 59. The format is locale independent and digits must be taken from the Basic Latin block of the Unicode standard.
- *RFC 822 time zone*: The RFC 822 4-digit time zone format is used:

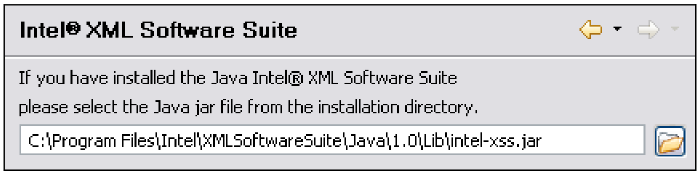
RFC822TimeZone:	Sign TwoDigitHours Minutes
TwoDigitHours:	Digit Digit

  TwoDigitHours must be between 00 and 23.

Intel® XML Software Suite

The *Intel® XML Software Suite* preferences panel is opened from menu Window → Preferences → oXygen+XML+Intel® XML Software Suite

Figure 19.60. The Intel® XML Software Suite preferences panel



Intel® XML Software Suite

If you have installed the Java Intel® XML Software Suite
please select the Java jar file from the installation directory.

C:\Program Files\Intel\XMLSoftwareSuite\Java\1.0\Lib\intel-xss.jar

After installing the *Intel® XML Software Suite* you will have to browse its installation directory from the preferences panel in <oXygen/> and choose the *intel-xss.jar* jar file.

You will also need to update your **PATH** environment variable for Windows or the **LD_LIBRARY_PATH** environment variable for Linux/Mac to include the path to the system libraries(.dll/.so) distributed in the *Intel® XML Software Suite*. For details please consult the *Intel® XML Software Suite* manual.

After updating the environment variables you will need to restart <oXygen/> so that they take effect.

Note

Depending on the version of Intel® XML Software Suite that you are installing, the environment variables may already be updated at installation time.

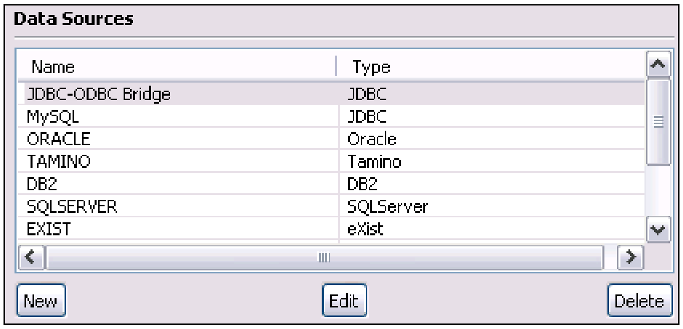
Data Sources

The Data Sources preferences panel is opened from menu Window → Preferences → oXygen+Data Sources

Configuration of Data Sources

Here you can configure data sources and connections to relational databases as well as native XML databases. You can check the list of drivers (http://www.oxygenxml.com/database_drivers.html) available for the major database servers.

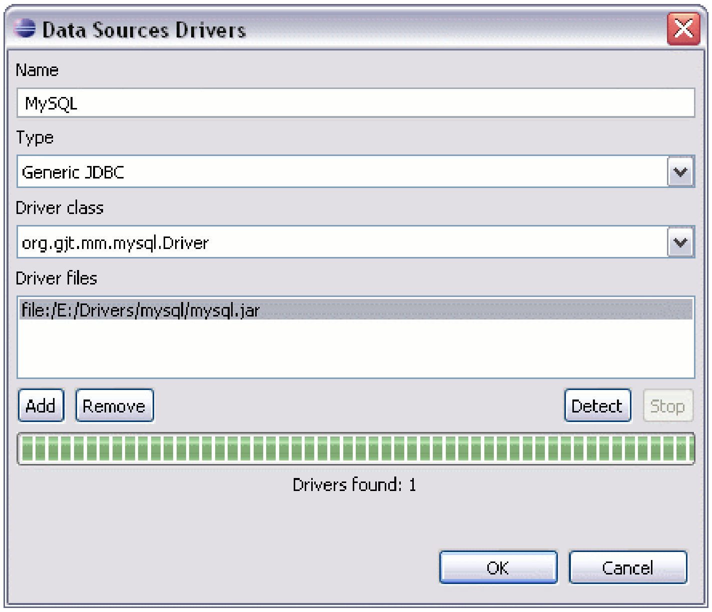
Figure 19.61. The Data Sources preferences panel



Name	Type
JDBC-ODBC Bridge	JDBC
MySQL	JDBC
ORACLE	Oracle
TAMINO	Tamino
DB2	DB2
SQLSERVER	SQLServer
EXIST	eXist

New Opens the Data Sources Drivers dialog, allowing you to configure a new driver.

Figure 19.62. The Data Sources Drivers dialog



Name: MySQL

Type: Generic JDBC

Driver class: org.gjt.mm.mysql.Driver

Driver files: file:/E:/Drivers/mysql/mysql.jar

Drivers found: 1

Name — Allows you to name the new data source driver.

Type — Select data source type from the supported driver types.

Help — Open the User Manual at the list of the sections where the configuration of supported data sources is explained and the URLs for downloading the database drivers are specified.

Driver Class — Provide the Driver Class for the data source driver

Add — Adds the driver class library.

Remove — Removes driver class library from the list.

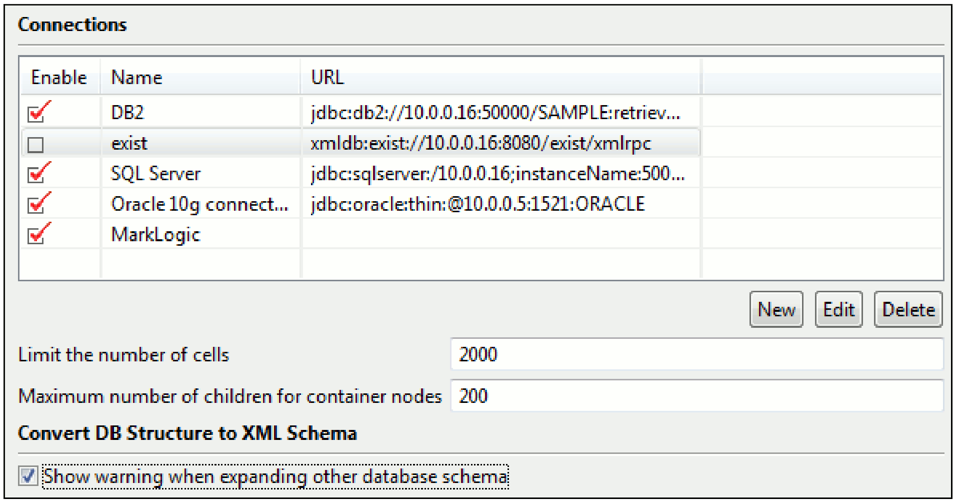
Detect — Detects driver candidates.

Stop — Stops the detection of the driver candidates.

Edit — Opens the Data Sources Drivers dialog, allowing you to edit the selected driver. See above the specifications for the Data Sources Drivers dialog (in order to edit a data source , there must be no connections using that data source driver).

Delete — Deletes the selected Data Source Driver (in order to delete a data source , there must be no connections using that data source driver).

Figure 19.63. The Connections preferences panel



Connections

Enable	Name	URL
✓	DB2	jdbc:db2://10.0.0.16:50000/SAMPLE:retriev...
□	exist	xmldb:exist://10.0.0.16:8080/exist/xmlrpc
✓	SQL Server	jdbc:sqlserver:/10.0.0.16;instanceName:500...
✓	Oracle 10g connect...	jdbc:oracle:thin:@10.0.0.5:1521:ORACLE
✓	MarkLogic	

Limit the number of cells: 2000

Maximum number of children for container nodes: 200

Convert DB Structure to XML Schema

☑ Show warning when expanding other database schema

Note

Checked connections will be visible in the *Database Explorer View*.

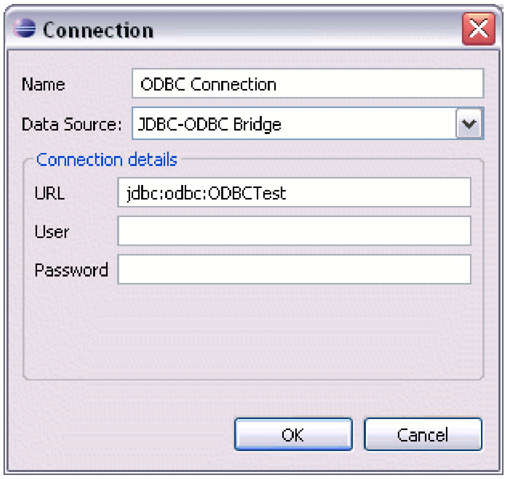
For performance issues, you can set the maximum number of cells that will be displayed in the Table Explorer view. Leave the field *Limit the number of cells* empty if you want the entire content of the table to be displayed. By default this field is set to 2,000. If a table having more cells than the value set here is displayed in the Table Explorer view, a warning dialog will inform you that the table is only partially shown.

In Oracle XML and Tamino databases a container can hold millions of resources. If the node corresponding to such a container in the Database Explorer view would display all the contained resources at the same time the performance of the view would be very slow. To prevent such a situation only a limited number of the contained resources is displayed as child nodes of the container node. Navigation to other contained resources from the same container is enabled by the *Up* and *Down* buttons of the Database Explorer view. This limited number is set in the option *Maximum number of children for container nodes*. The default value is 200 nodes.

The Show warning when expanding other database schema in the section Convert DB Structure to XML Schema controls if a warning message will be displayed when expanding another database schema and there are tables selected in the current expanded one. This applies for the dialog Select database table when invoking Convert DB Structure to XML Schema action.

New   Opens the Connection dialog.

Figure 19.64. The Connection dialog



Name: ODBC Connection
Data Source: JDBC-ODBC Bridge
URL: jdbc:odbc:ODBCTest
User:
Password:

Name   Allows you to name the new connection.

Data Source   Select data source defined in the Data Source Drivers dialog.

Depending upon the selected Data Source, you can set some of the following parameters in the *Connection details* area:

URL: The URL used to connect.

User: Provide the database user .

Password: Provide the database password.

Host: Provide the host address.

Port: Provide a port to connect.

XML DB URI: Provide the database URI to connect.

Database: Provide the initial database.

Collection: Select one of the available collections for the specified data source.

Environment home directory: Specify the home directory for a Berkeley database.

Verbosity: Set the verbosity level for a Berkeley database.

Edit Opens the Connection dialog, allowing you to edit the selected connection. See above the specifications for the Connection dialog.

Delete Deletes the selected connection.

Download links for database drivers

You can find below the locations where you have to go to get the drivers necessary for accessing databases in <oXygen/>

Berkeley DB XML database Copy the jar files from the Berkeley database install directory to the <oXygen/> install directory as described in the procedure for configuring a Berkeley DB data source.

IBM DB2 Pure XML database Go to the IBM website: http://www-306.ibm.com/software/data/db2/express/download.html [http://www-306.ibm.com/software/data/db2/express/download.html], in the *DB2 Clients and Development Tools* category select the *DB2 Driver for JDBC and SQLJ* download link, fill the download form and download the zip file. Unzip the zip file and use the db2jcc.jar and db2jcc_license_cu.jar files in <oXygen/> for configuring a DB2 data source.

eXist database Copy the jar files from the eXist database install directory to the <oXygen/> install directory as described in the procedure for configuring an eXist data source.

MarkLogic database Download Java and .NET XCC distributions (XCC Connectivity Packages) from http://xqzone.marklogic.com/download/#XCC. Details about configuring a MarkLogic data source are here.

Microsoft SQL Server 2005 / 2008 database Both SQL Server 2005 and SQL Server 2008 are supported. Download the SQL Server 2005 JDBC driver called `sqljdbc.jar` from the Microsoft website: http://www.microsoft.com/downloads/details.aspx?familyid=C47053EB-3B64-4794-950D-81E1EC91C1BA&displaylang=en and use it for configuring an SQL Server data source. Download the SQL Server 2008 JDBC driver called `sqljdbc4.jar` from the Microsoft website: http://www.microsoft.com/downloads/details.aspx?familyid=f914793a-6fb4-475f-9537-b8fcb776befd&displaylang=en and use it for configuring an SQL Server data source.

Oracle 11g database Download the Oracle 11g JDBC driver called ojdbc5.jar from the Oracle website: http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/jdbc_111060.html [http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/jdbc_111060.html] and use it for configuring an Oracle data source.

PostgreSQL 8.3 database Download the PostgreSQL 8.3 JDBC driver called postgresql-8.3-603.jdbc3.jar from the PostgreSQL website: http://jdbc.postgresql.org/download.html and use it for configuring a PostgreSQL data source.

RainingData TigerLogic XDMS database Copy the jar files from the TigerLogic JDK lib directory from the server side to the <oXygen/> install directory as described in the procedure for configuring a TigerLogic data source.

SoftwareAG Tamino database Copy the jar files from the SDK\TaminoAPI4J\lib subdirectory of the Tamino database install directory to the <oXygen/> install directory as described in the procedure for configuring a Tamino data source.

Documentum xDb (X-Hive/DB) XML database

Copy the jar files from the Documentum xDb (X-Hive/DB) database install directory to the <oXygen/> install directory as described in the procedure for configuring an XHive data source.

MySQL database

A MySQL driver file is included in the Oxygen kit. The installer creates the file mysql.jar in the folder `[Oxygen-install-folder]/lib`. When creating a new data source select the type *Generic JDBC* and add the file `[Oxygen-install-folder]/lib/mysql.jar` in *Driver files*. If you want to connect to a MySQL 5 server you may need the latest driver from the MySQL website: http://dev.mysql.com/downloads/connector/j/5.1.html

Table Filters

The Table Filters preferences panel is opened from menu Window → Preferences → oXygen+Data Sources+Table Filters

Here you can choose which of the table types will be displayed in the *Database Explorer* view.

Figure 19.65. Table Filters Preferences Page

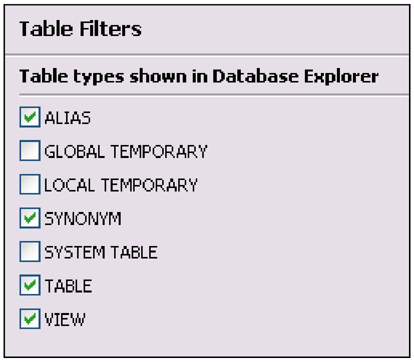


Table Filters

Table types shown in Database Explorer

☑ ALIAS
☐ GLOBAL TEMPORARY
☐ LOCAL TEMPORARY
☑ SYNONYM
☐ SYSTEM TABLE
☑ TABLE
☑ VIEW

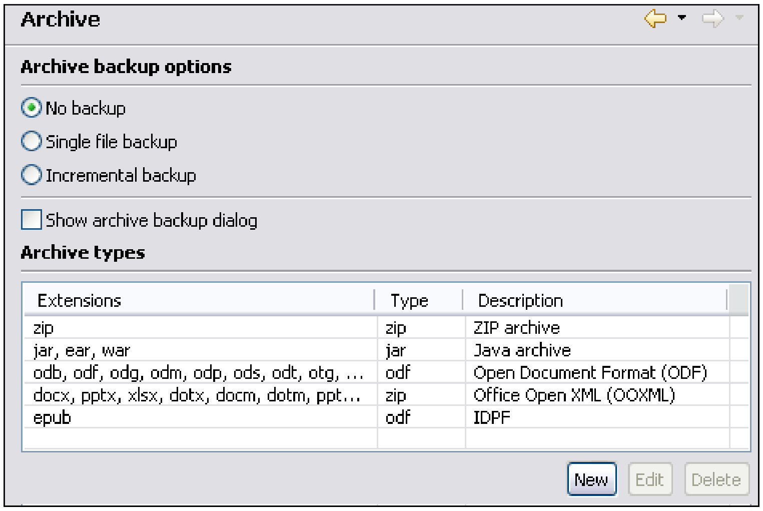
Note

Table types filtering depends on the driver implementation.

Archive

The Archive preferences panel is opened from menu Window → Preferences → oXygen+Archive

Figure 19.66. The Archive preferences panel



Archive

Archive backup options

● No backup

○ Single file backup

○ Incremental backup

□ Show archive backup dialog

Archive types

Extensions	Type	Description
zip	zip	ZIP archive
jar, ear, war	jar	Java archive
odb, odf, odg, odm, odp, ods, odt, otg, ...	odf	Open Document Format (ODF)
docx, pptx, xlsx, dotx, docm, dotm, ppt...	zip	Office Open XML (OOXML)
epub	odf	IDPF

The following options are available in the Archive preferences page:

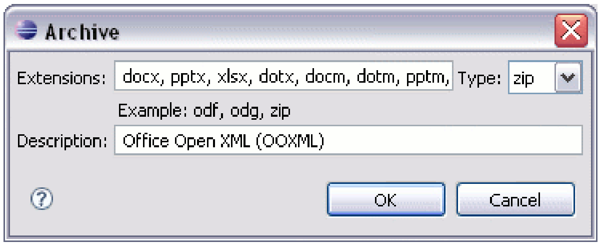
The following archive backup options are considered default options for backup in the Archive Backup dialog.

- No backup — Perform no backup of the archive before save. This means that the file will be saved directly in the archive without any additional precautions.
- Single file backup — Before any operation which modifies the archive is performed, the archive contents will be duplicated. The duplicate file name will be `originalArchiveFileName.bak` and will be saved in the same directory.
- Incremental backup — Before each operation which modifies the archive is performed, the archive contents will be duplicated. The duplicate file names will be `originalArchiveFileName.bak#dupNo` and the files will be saved in the same directory.

Show archive backup dialog — Check this if you want to be notified for backup when modifying in archives. The last backup option you chose will always be used as the default one.

Archive types table — This table contains all known archive extensions mapped to known archive formats. You can edit the table to modify existing mappings or add your own extensions to the list of known archive extensions.

Figure 19.67. Edit the Archive extension mappings



Archive

Extensions: docx, pptx, xlsx, dotx, docm, dotm, pptm, Type: zip

Example: odf, odg, zip

Description: Office Open XML (OOXML)

You can map a list of extensions to an archive type supported in <oXygen/>.

Important

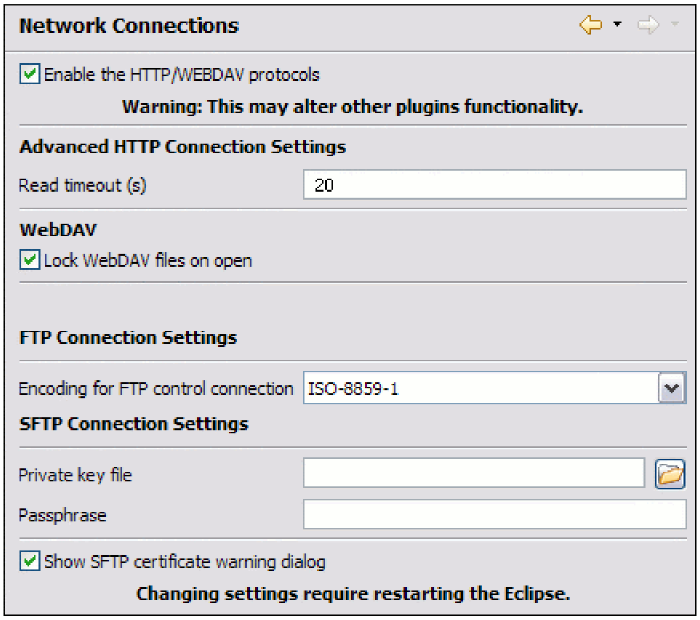
You have to restart <oXygen/> after removing an extension from the table in order for that extension to not be recognised anymore as an archive extension.

Network Connections

Some networks use Proxy servers to provide Internet Services to LAN Clients. Clients behind the Proxy may therefore, only connect to the Internet via the Proxy Service. The Proxy Configuration dialog enables this configuration. If you are not sure whether your computer is required to use a Proxy server to connect to the Internet or the values required by the Proxy Configuration dialog, please consult your Network Administrator.

Open the Network Connections panel by selecting Window → Preferences → oXygen+Network Configuration.

Figure 19.68. The Network Connections preferences panel



Network Connections

☑ Enable the HTTP/WEBDAV protocols

Warning: This may alter other plugins functionality.

Advanced HTTP Connection Settings

Read timeout (s): 20

WebDAV

☑ Lock WebDAV files on open

FTP Connection Settings

Encoding for FTP control connection: ISO-8859-1

SFTP Connection Settings

Private key file:

Passphrase:

☑ Show SFTP certificate warning dialog

Changing settings require restarting the Eclipse.

Complete the dialog as follows:

Enable the HTTP/WEBDAV protocols

When checked Http/WebDAV proxy and proxy settings are enabled. The host, port, username and password that the <oXygen/> plugin uses are the ones set in the general *Network Connections* settings of Eclipse.

Important

This may affect other plugins functionality.

Lock WebDAV files on open

If checked the files opened through WebDAV are locked on the server so that they cannot be edited by other users while the lock placed by the current user still exists.

Encoding for FTP control connection — The encoding used to communicate with FTP servers. It is one of ISO-8859-1 and UTF-8. If the server supports the UTF-8 encoding <oXygen/> will use it for communication. Otherwise it will use ISO-8859-1.

Private key file — The path to the file containing the private key used for the private key method of authentication of the secure FTP (SFTP) protocol. The user/password method of authentication has precedence if it is used in the Open URL dialog.

Passphrase — The passphrase used for the private key method of authentication of the secure FTP (SFTP) protocol. The user/password method of authentication has precedence if it is used in the Open URL dialog.

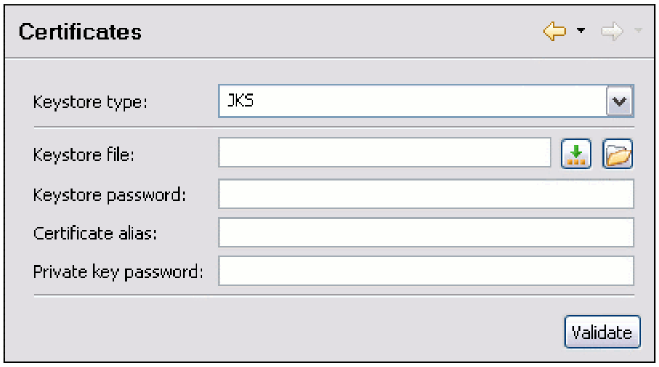
Show SFTP certificate warning dialog — If checked a warning dialog will be shown each time when the authenticity of the host cannot be established.

Certificates

In <oXygen/> there are provided two types of Keystores: Java KeyStore (JKS) and Public-Key Cryptography Standards version 12 (PKCS-12). A keystore file is protected by a password.

The Certificates preferences panel is opened from menu Window → Preferences → oXygen+Certificates

Figure 19.69. The Certificates preferences panel



Certificates

Keystore type: JKS

Keystore file:

Keystore password:

Certificate alias:

Private key password:

Keystore type — Represents the type of keystore to be used.

Keystore file — Represents the location of the file to be imported.

Keystore password — The password which is used to protect the privacy of the stored keys.

Certificate alias — The alias to be used to store the key entry (the certificate and /or the private key) inside the keystore.

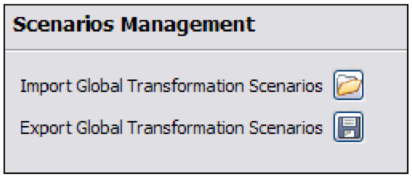
Private key password — It is only necessary in case of JKS keystore. It represents the certificate's private key password.

Validate — Verifies the entries from the fields; assures that the certificate is valid.

Scenarios Management

The Scenarios Management preferences panel is opened from menu Window → Preferences → oXygen+Scenarios Management

Figure 19.70. The Scenarios Management preferences panel



Scenarios Management

Import Global Transformation Scenarios

Export Global Transformation Scenarios

Import transformation scenarios

Allows you to import all transformation scenarios from a scenarios properties file. Their names will appear in the "Configure Transformation Scenario" dialog followed by "(import)". This way there are no scenarios' names conflicts.

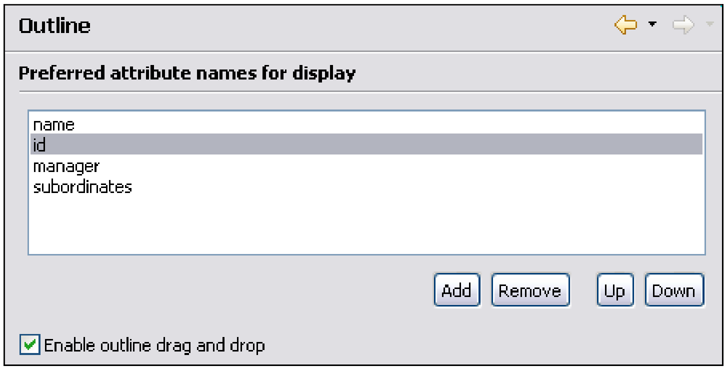
Export transformation scenarios

Allows you to export all transformation scenarios available in the "Configure Transformation Scenario" dialog.

Outline

The Outline preferences panel is opened from menu Window → Preferences → oXygen+Outline

Figure 19.71. The Outline preferences panel



Outline

Preferred attribute names for display

name
id
manager
subordinates

☑ Enable outline drag and drop

Preferred attribute names for display

The attribute names which should be preferred when displaying element's attributes in the outline view. If there is no preferred attribute name specified the first attribute of an element is displayed in the outline view.

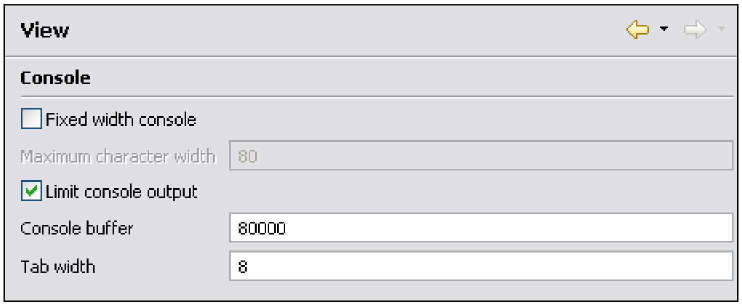
Enable outline drag and drop

When drag and drop is disabled for the tree displayed by the outline view there is no possibility to accidentally change the structure of the document.

View

The View preferences panel is opened from menu Window → Preferences → oXygen+View

Figure 19.72. The View preferences panel



View

Console

□ Fixed width console

Maximum character width: 80

☑ Limit console output

Console buffer: 80000

Tab width: 8

Fixed width console

When checked, a line in the Console view will be hard wrapped after Maximum character width characters.

Limit console output

When checked the content of the Console view will be limited to a configurable number of characters.

Console buffer - specifies the maximum number of characters that can be written at some point in the Console view.

Tab width - specifies the number of spaces used for depicting a tab.

Automatically importing the preferences from the other distribution

If you want to use the settings from “standalone” in the Eclipse plugin just delete the file with the Eclipse plugin settings `[user-home-dir]/Application Data/com.oxygenxml/oxyOptionsEc10.3.xml` on Windows / `[user-home-dir]/.com.oxygenxml/oxyOptionsEc10.3.xml` on Linux, start Eclipse and the “standalone” settings will be automatically imported in Eclipse. The same for importing the Eclipse plugin settings in “standalone”: delete the file `[user-home-dir]/com.oxygenxml/oxyOptionsSa10.3.xml`, start the <oXygen/> “standalone” distribution and the Eclipse settings will be automatically imported.

Reset Global Options

To reset all custom user settings of the application that are stored in a local file (not in the project), to the installation defaults go to: Window+Preferences → oXygen+Reset Global Options The list of transformation scenarios will be reset to the default scenarios.

Scenarios Management

You can import, export and reset the scenarios stored in the global options.

- The action Window → Preferences+oXygen / Scenarios Management+ Import Global Transformation Scenarios loads a properties file with scenarios.

- The action Window → Preferences+oXygen / Scenarios Management+ Export Global Transformation scenarios stores all the scenarios in a separate properties file.

The option to Export Transformation Scenarios is used to store all the scenarios in a separate file , a properties file. In this file will also be saved the associations between document URLs and scenarios. The saved URLs are absolute. You can load the saved scenarios using Import Transformation Scenarios option. All the imported scenarios will have added to the name the word 'import'.

Note

The scenarios are exported/imported from/in the global options, not from the project options. So be aware that the list of scenarios kept at the project level are not affected.

Editor variables

An editor variable is a shorthand notation for a file path or directory path. It is used in the definition of a command (the input URL of a transformation, the output file path of a transformation, the command line of an external tool, etc.) to make the command generic. When the same command is applied the notation is expanded so that the same command has different effects depending on the actual value of the notation.

The following editor variables can be used in <oXygen/> commands:

`${frameworks}` the path of the `frameworks` subdirectory of the <oXygen/> install directory as URL

`${frameworksDir}` the path of the `frameworks` subdirectory of the <oXygen/> install directory

`${home}` the path of the user home directory as URL

`${homeDir}` the path of the user home directory

`${cfdu}` current file directory url - the path of the current edited document up to the name of the parent directory as URL

`${cfd}` current file directory - the path of the current edited document up to the name of the parent directory

`${cfn}` current file name - the name of the current edited document without extension and parent directory

`${cf}` current file - the absolute file path of the current edited document

`${currentFileURL}` current file as URL - the absolute file path of the current edited document as URL

`${ps}` Path Separator - The separator which can be used on different operating systems between libraries specified in the class path.

`${timeStamp}` Time Stamp - The current Unix time on the computer which can be used to save transformation results in different output files on each transform.

Chapter 20. Common problems

20.1. When I run a transformation in the XSLT Debugger perspective it is very slow. Can I increase the speed?

Disable rendering of output to the XHTML Output view during the transformation process if the transformation produces HTML or XHTML output. In order to view the output result run the transformation in the Editor perspective with the option "Open in browser" or run it in the Debugger perspective, save the Text output area to a file and use an external browser for viewing.

20.2. Before installing Oxygen XML Editor/Author I had no problems viewing XML files in Internet Explorer but now Internet Explorer opens an XML file in Oxygen XML Editor/Author. How can I view XML files in Internet Explorer again?

XML files are opened in Oxygen because Internet Explorer uses the Windows file associations for opening files and you associated XML files with Oxygen XML Editor/Author in the installer panel called File Associations. This installer panel displays a warning above the XML file association that XML files will not be viewed correctly in Internet Explorer if you associate them with Oxygen XML Editor/Author.

For viewing XML files in Internet Explorer again you have to associate XML files with IE by right-clicking on an XML file in Windows Explorer, selecting Open With -> Choose Program, selecting IE in the list of applications and checking the checkbox "Always use the selected program". Also you have to run the following command from a command line:

```
wscript revert.vbs
```

where revert.vbs is a text file with the following content:

```
function revert()
  Set objShell = CreateObject("WScript.Shell")
  objShell.RegWrite "HKCR\.xml\", "xmlfile", "REG_SZ"
  objShell.RegWrite "HKCR\.xml\Content Type", "text/xml", "REG_SZ"
end function

revert()
```

20.3. I associated the `.ext` extension with <oXygen/> in Eclipse. Why does an `.ext` file opened with the Oxygen XML Editor not have syntax highlight?

Associating an extension with <oXygen/> in Eclipse 3.4+ requires three steps:

1. Associate the `.ext` extension with the Oxygen XML Editor: go to Windows -> Preferences -> General -> Editors -> File Associations, add `*.ext` to the list of file types, select `*.ext` in the list by clicking on it, add *Oxygen XML Editor* to the list of *Associated editors* and make it the default editor.

2. Associate the `.ext` extension with the Oxygen XML content type: go to Windows -> Preferences -> General -> Content Types and for the Text -> XML -> oXygen XML content type add `*.ext` to the *File associations* list.

3. Press the *OK* button of the Eclipse preferences dialog.

When a *.ext file is opened the icon of the editor and the syntax highlight should be the same as for XML files opened with the Oxygen XML Editor.

Index