User Guide

User Guide

Table of Contents

List of Figures

List of Tables

List of Examples

Chapter 1. Writing Documentation - How to create a PDF file using the <oXygen/> XML Editor

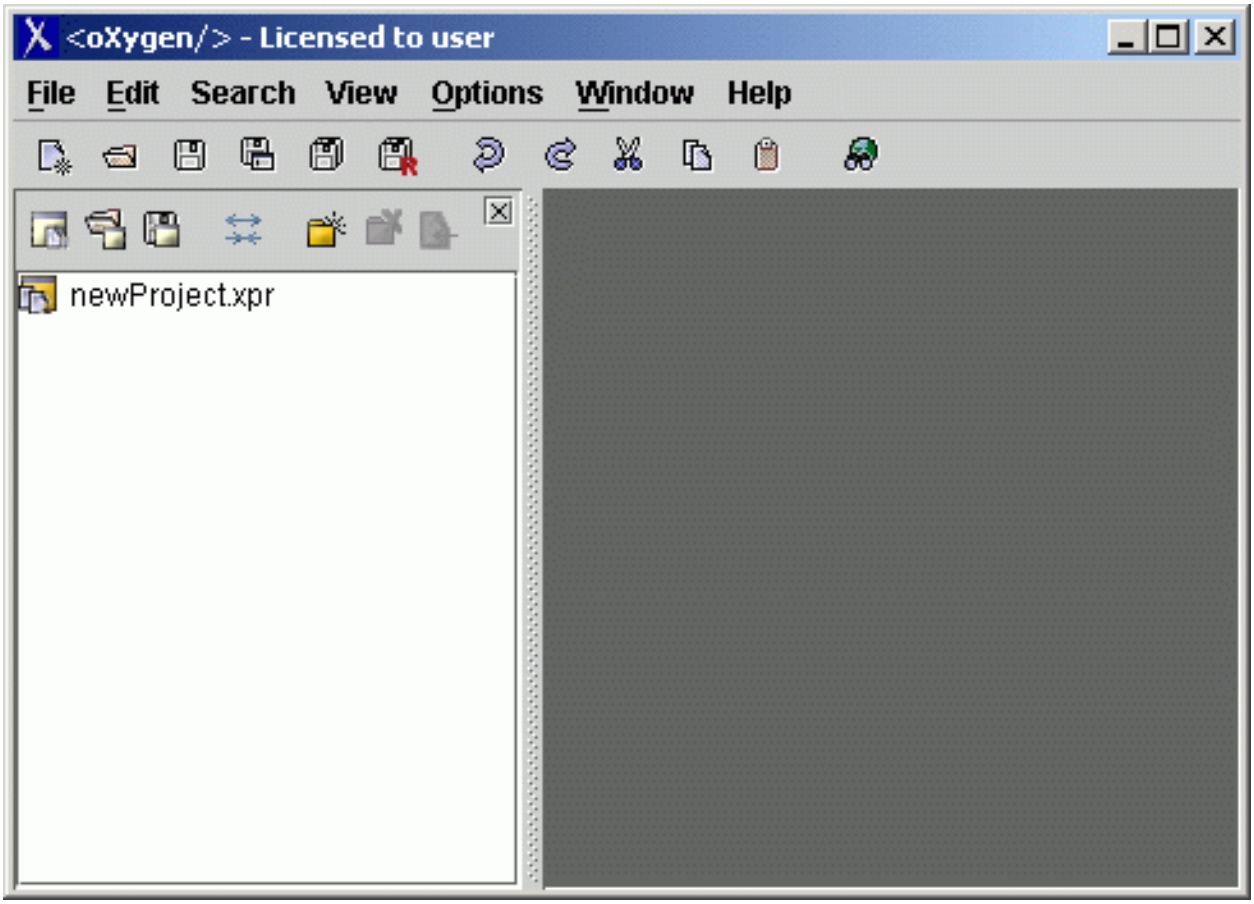
Foreword

This article is intended as an example of creating PDF documents using the <oXygen/> XML editor. In this process will be used the Docbook DTD, the Docbook XSL/FO package and the Apache's FOP. All these are integrated in the installation kit.

Starting the Editor.

Start the XML editor. OnWindows™ you should have a shortcut in Start/Programs/Oxygen.

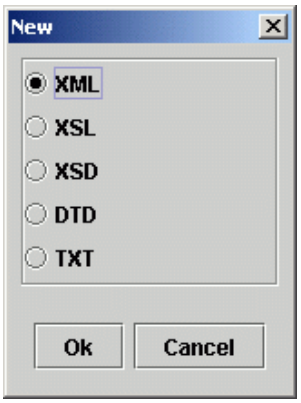
Figure 1.1. Editor GUI



Creating an XML Document Using Docbook.

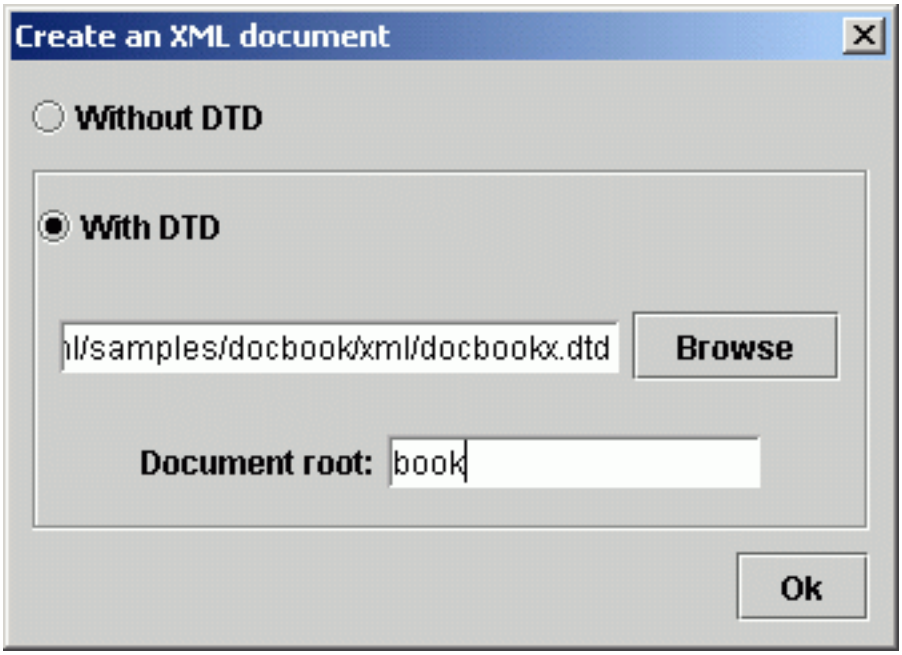
From the "File" menu choose "New". The following window will be displayed, asking you to choose from a list of document types the editor is able to create:

Figure 1.2. File types editor can handle



Choose xml and click on the "Ok" button. You will be asked next to choose the DTD for the document and the root element.

Figure 1.3. Choosing the DTD and the root element



You can browse for the dockbook.dtd file using the "Browse" button. You can find it in the {OxygenInstallDir}/samples/docbook/xml directory, where {OxygenInstallDir} is the install dirrectory of the oXy-<gen/> XML editor. The document root can be "book" or "article". The editor inserts the DOCTYPE declaration and the root elements. The encoding is automatically set to "UTF-8".

Note

The encoding can be changed at any time by editing the XML prolog, for instance, changing to UTF-16:

```
<?xml version="1.0" encoding="UTF-16"?>
```

Editing the XML Document

The editor will scan the DTD file and will initialize the CodeInsight assistant. By pressing the "<" key, is displayed a window containing all the elements that can be inserted at that point in the document. Notice that the window appears also when adding new attributes or when an attribute has default values you can choose from. In the first case the CodeInsight window is triggered by pressing the SPACE key, in the later by pressing ' or " keys.

A simple docbook document can be:

<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE book SYSTEM "file:/C:/projects/eXml/samples/docbook/xml/docbookx.dtd"> <book> <article> <title>A Test Article</title> sec- <tion> <title>A section</title> <para>The section text.</para> </section> </article> </book>

Saving the XML Document

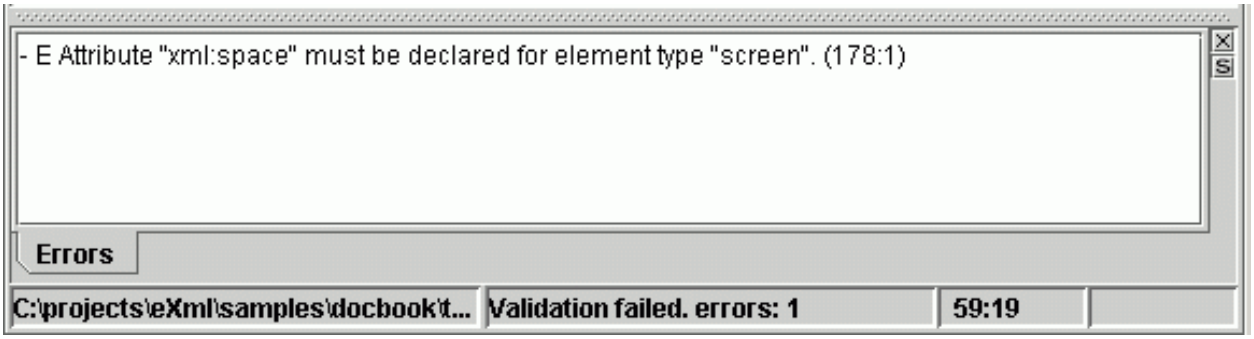
To save the document use the File/Save option. You will be prompted for the file name.

Transforming to PDF

You can either use the sample provided in {OxygenInstallDir}/samples/docbook/tutor/ named "userGuide.xml", or use your own edited file.

Before proceeding to the FOP transformation you need to check the validity of the document. For this, press the "V" button from the left of the view toolbar, or CTRL-SHIFT-V (COMMAND-SHIFT-V on Mac). The validity report will be displayed in the status bar. If there are found some errors, they will be listed.

Figure 1.4. Validation errors

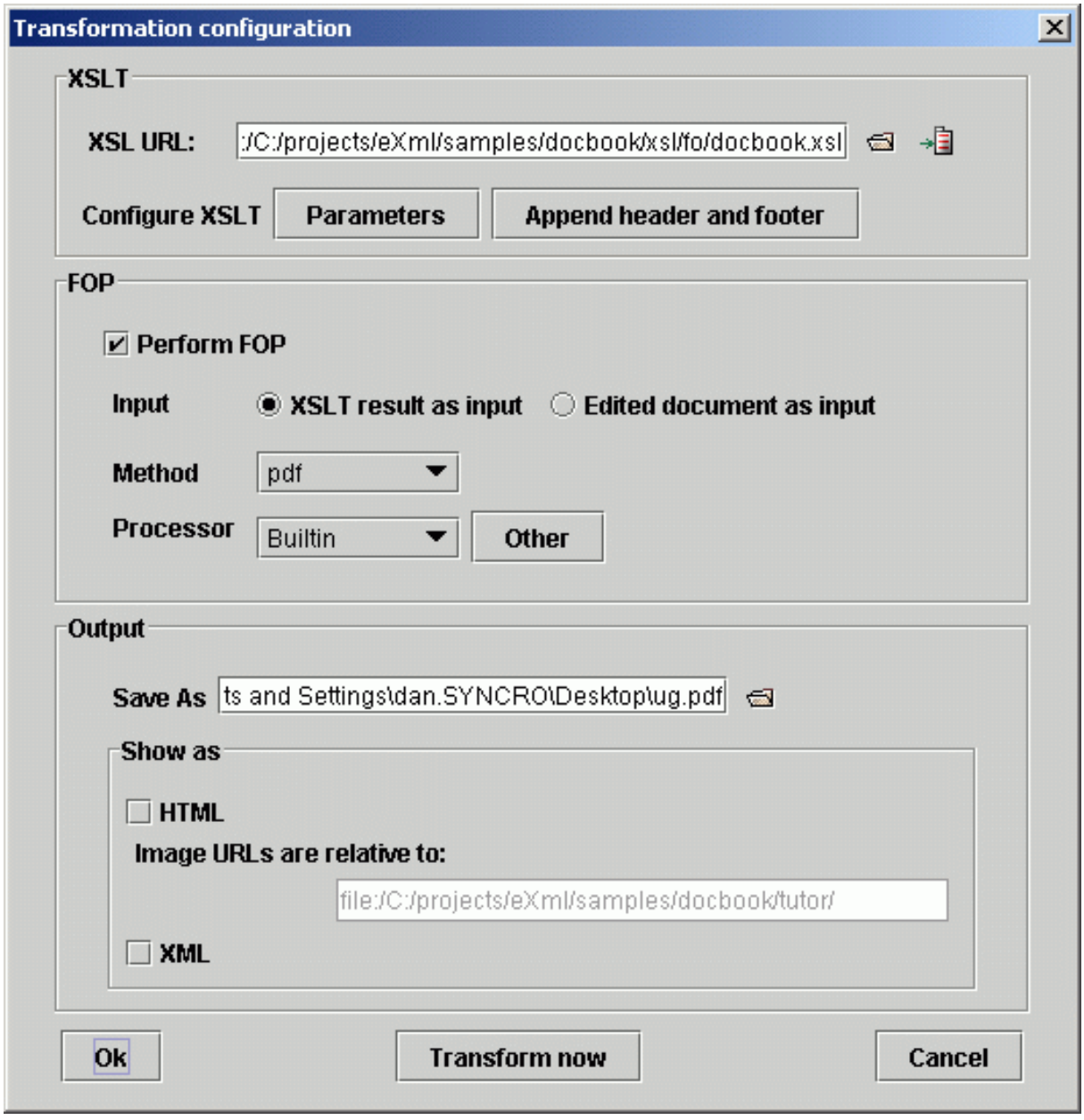


Configuring the XSL/FOP transformation

- Press the "T.." button from the toolbar. This will open the Transformation Configuration window.
- In the XSL URL choose the "docbook.xsl" file. You can find it by browsing to {OxygenInstallDir}/samples/docbook/xsl/fo/. An XML FO document is generated by applying this stylesheet. Next, a FO processor can be run over this result in order to generate output in PFD or PostScript format.

- Make sure that "Perform FOP" checkbox is selected.
- Make sure that "XSLT result as input" radiobutton is selected.
- Let the processor set to "Builtin".
- Let the method set to "pdf".
-

Figure 1.5. Transformation Configuration Dialog



- Press the "Ok" button.

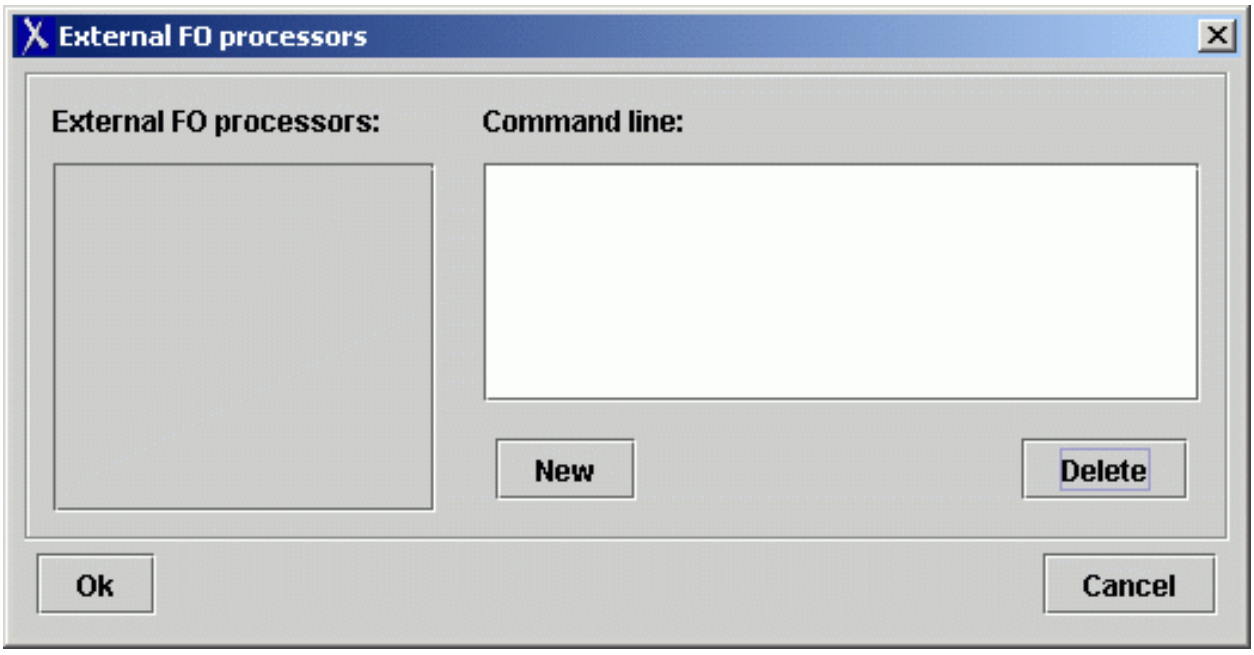
Press the "T" (Transform) button from the toolbar or use CTRL-SHIFT-T (COMMAND-SHIFT-T on Mac) shortcut. This way the XSLT and FO processing is started. In the status bar is presented the current state. Because this transformation consists of two stages: an XSL transform of our document and the docbook.xsl, resulting the FO document, and then the FO to PDF transformation, it is possible to be first displayed the "XSL transformation successfull" message and then "FO transformation successfull" when the FO processor ends. You can inspect the generated PDF file using theAcrobat Reader™.

Configuring an external FO Processor, or other post processor.

The builtin FO Processor is based on the Apache project, and is under development. Problems may occure, depending on the complexity of the intermediate FO file.

To avoid this kind of difficulties it is possible to configure one or more external FO processors. In other situations, you may need to run other processors, not necessarly FOP on the XSLT result or over the edited document. The button "Other" from the Transformation Configuration window gives us access to the list of external processors.

Figure 1.6. The external FOP configuration dialog



By default, the list is empty. By pressing the "New" button we can create a new FOP entry. The entry consists of:

- A name. This can be any name, but no whitespaces are allowed in it.
- A command line template. The command line is specifying the FOP executable. Try not to use envirnoment variables in its expression, since those will not be expanded. There are three macros that allow to indicate the FO method (the macro "$1") - this being replaced by the editor with "pdf", "ps" or "txt"; the FO input file (the macro "$1")- this can be the edited file or the result of an XSLT; the file in which the result is stored (the macro $3).

Note

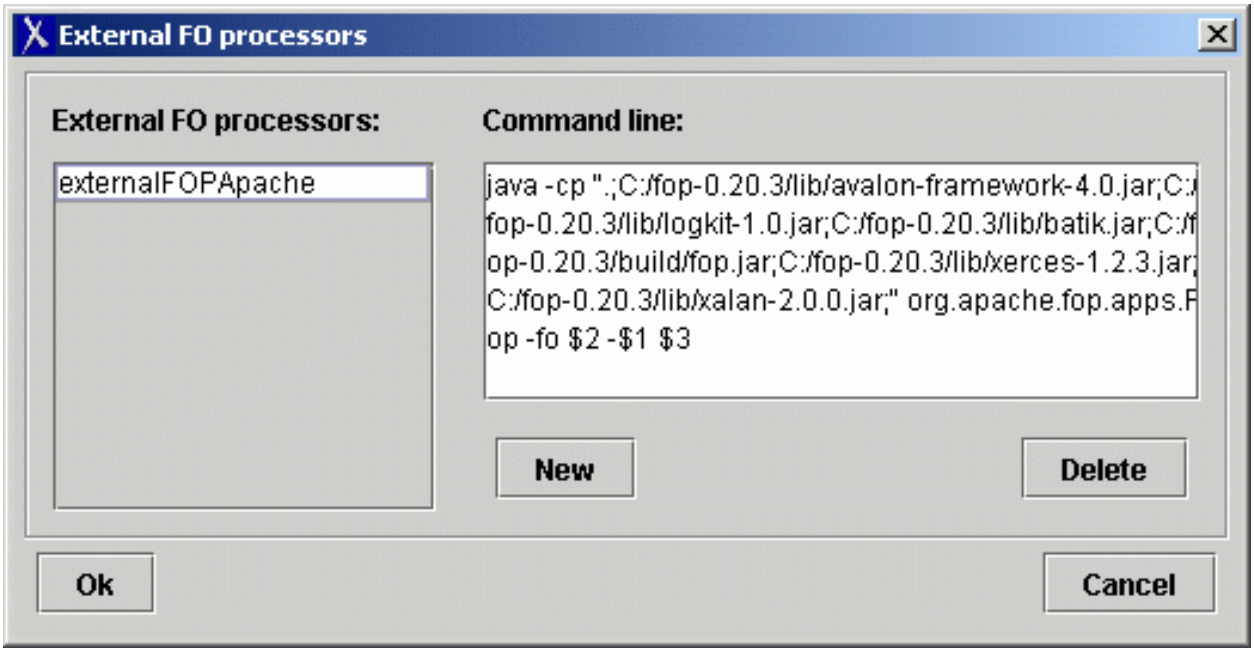
Your editor may take as argument just one or two of the possible parameters. It is not required to use all "$1", "$2" ,"$3" macros in the command line.

By default the editor provides a configuration sample, in which the name is "externalFOP" and the command line template is: "sample: fop -method $1 -fo $2 -out $3".

In the following we will configure the template to use FOP Apache as external FOP. You can download it fromhttp://xml.apache.org/. We will consider that you have extracted the bundle/ \ in: C:\fop-0.20.3. The command line template will be in this case:

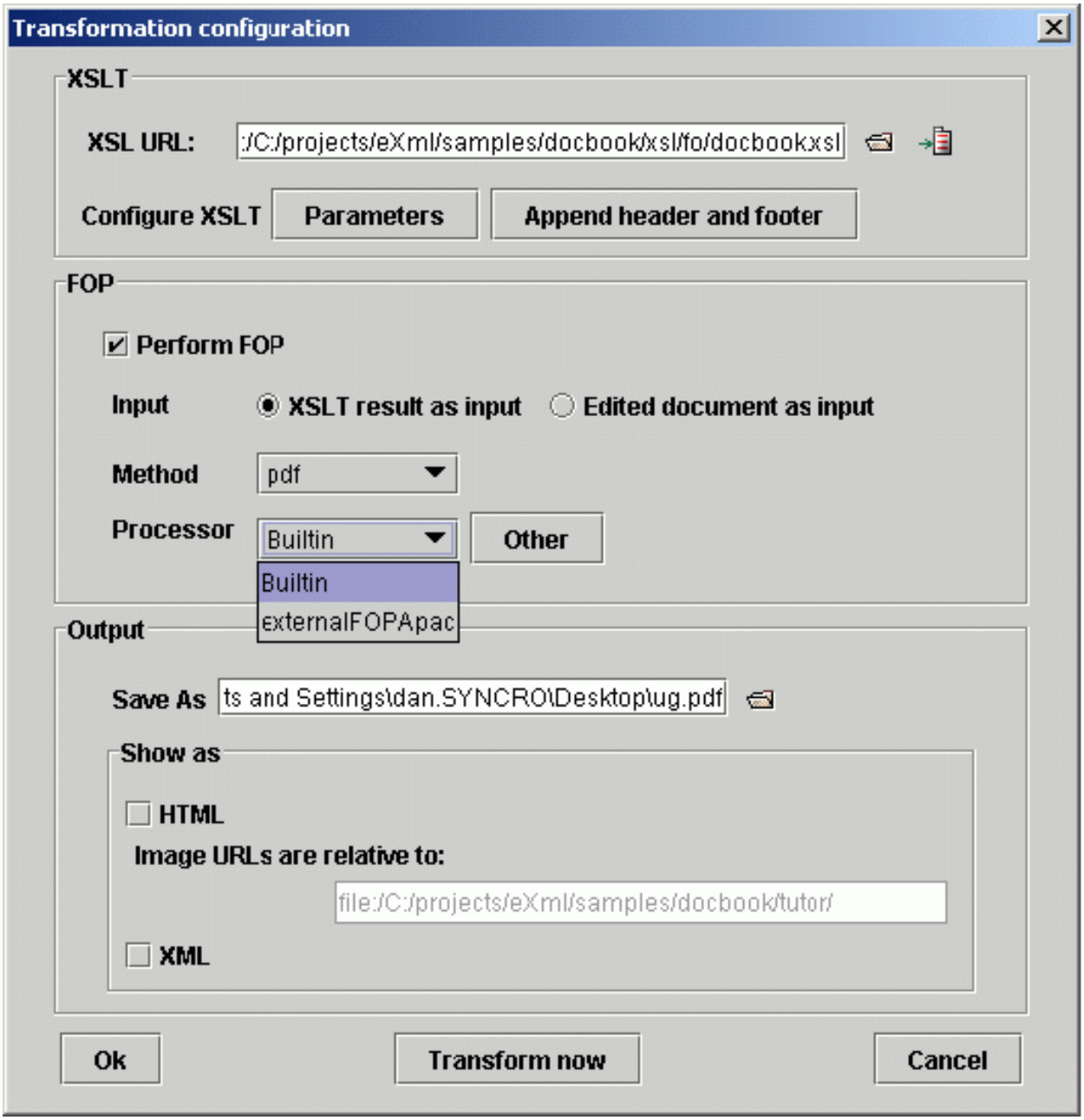
java -cp ".;C:/fop-0.20.3/lib/avalon-framework-4.0.jar;C:/fop-0.20.3/lib/logkit-1.0.jar;C:/fop-0.20.3/lib/batik.jar;C:/fop-0.20.3/build/fop.jar;C:/fop-0.20.3/lib/xercesImpl.jar;C:/fop-0.20.3/lib/xmlParserAPIs.jar;" org.apache.fop.apps.Fop -fo $2 -$1 $3

Figure 1.7. Adding an external FOP



Press "Ok" and return to the Transformation Configuration Dialog. Now you will see listed in the processor combo box bellow the "Builtin" entry the name of the external FOP we configured "externalFOPApache".

Figure 1.8. Transformation Configuration Dialog



Fill the rest of the form fields. Now you can perform the transformation. The output of the external FOP will be captured and presnted in a tab named "Processor". If the processor does not generate output, this tab will not be shown. Bellow is an output sample:

Table 1.1. Output

[INFO]: FOP 0.20.3 [INFO]: building formatting object tree

```
[ERROR]: Error in background-image property value
'http://docbook.sourceforge.net/release/images/draft.png':
org.apache.fop.fo.expr.PropertyException: illegal character
[WARN]: property - "background-attachment" is not implemented yet.
[WARN]: property - "background-repeat" is not implemented yet.
[WARN]: property - "background-position-horizontal" is not implemented yet.
[WARN]: property - "background-position-vertical" is not implemented yet.
[WARN]: property - "last-line-end-indent" is not implemented yet.
[WARN]: property - "last-line-end-indent" is not implemented yet.
[WARN]: property - "last-line-end-indent" is not implemented yet.
[WARN]: property - "last-line-end-indent" is not implemented yet.
[WARN]: property - "last-line-end-indent" is not implemented yet.
[WARN]: property - "last-line-end-indent" is not implemented yet.
[WARN]: property - "last-line-end-indent" is not implemented yet.
[INFO]: [1]
[INFO]: [1]
[INFO]: [2]
[INFO]: [3]
[INFO]: [4]
[INFO]: [5]
[INFO]: [6]
[INFO]: [7]
[INFO]: Parsing of document complete, stopping renderer
```

Chapter 2. Working with Large Documents

Foreword

Explains how to deal with large documents - splitting.

The problem

Let's consider the case of documenting a large project. It is likely that there are 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 will not be affected ?

Fortunately, XML provides a solution for this. It can be created a master document, with references to other documents, containing the documentation 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.

- 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 ceratin 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.

Using the project panel

When you have a large number of files to edit and organize, you may use the project support.

Note

The operations can be accessed using the toolbar buttons.

Creating a project.

Choose File / New Project to create a new project. Make sure the project panel is visible by checking the View / Show Project item. (A check mark shold be displayed in the menu.)

Creating project folders.

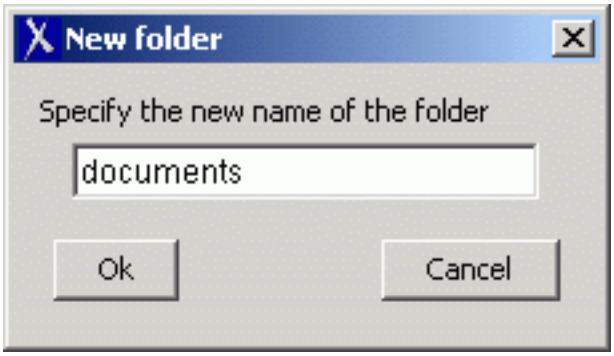
We can organiza the project as a collection of folders. These are logical folders, they do not have any connection

with directories on the disk. Right click on the icon of the project, in the project panel. A popup menu will be shown.

Figure 2.1. Project panel popup menu

Choose the first option, "New folder". Enter a name of the folder.

Figure 2.2. Project panel new folder dialog

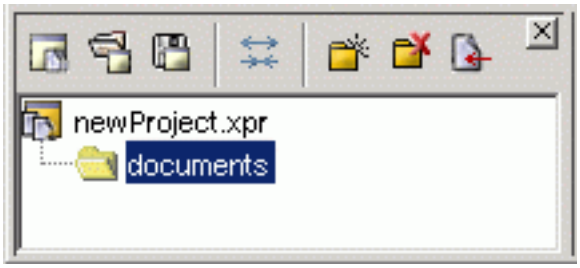


Adding files to a project.

To add one or more files to the newly created folder, right click on it, and choose "Add file".

A shortcut for adding the edited file to the selected folder is to press the right-most button from the project panel toolbar.

Figure 2.3. Project panel toolbar



Removing files or project folders .

Right click on the item you want to remove. Choose ythe remove option.

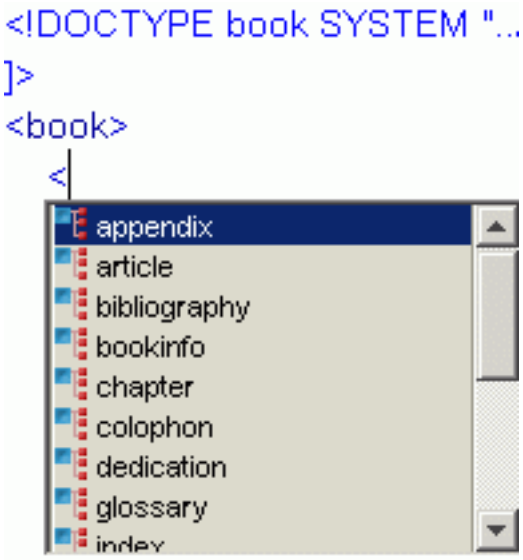
Imposing a DTD for the CodeInsight.

As explained above, when splitting a large document, only one (the main) will contain the Document Type Definition (the DTD) and will include the others. The included sections cannot define again the DTD because the main document will not be valid.

Important

The editor is creating the CodeInsight lists in function of the specified DTD and the current context (the position in the editor). If you change the DTD you can observe that the list of tags to be inserted is changing.

Figure 2.4. CodeInsight driven by a Docbook DTD



To offer CodeInsight on the included files, you can specify a DTD or XML Schema to be used when the documents do not specify one.

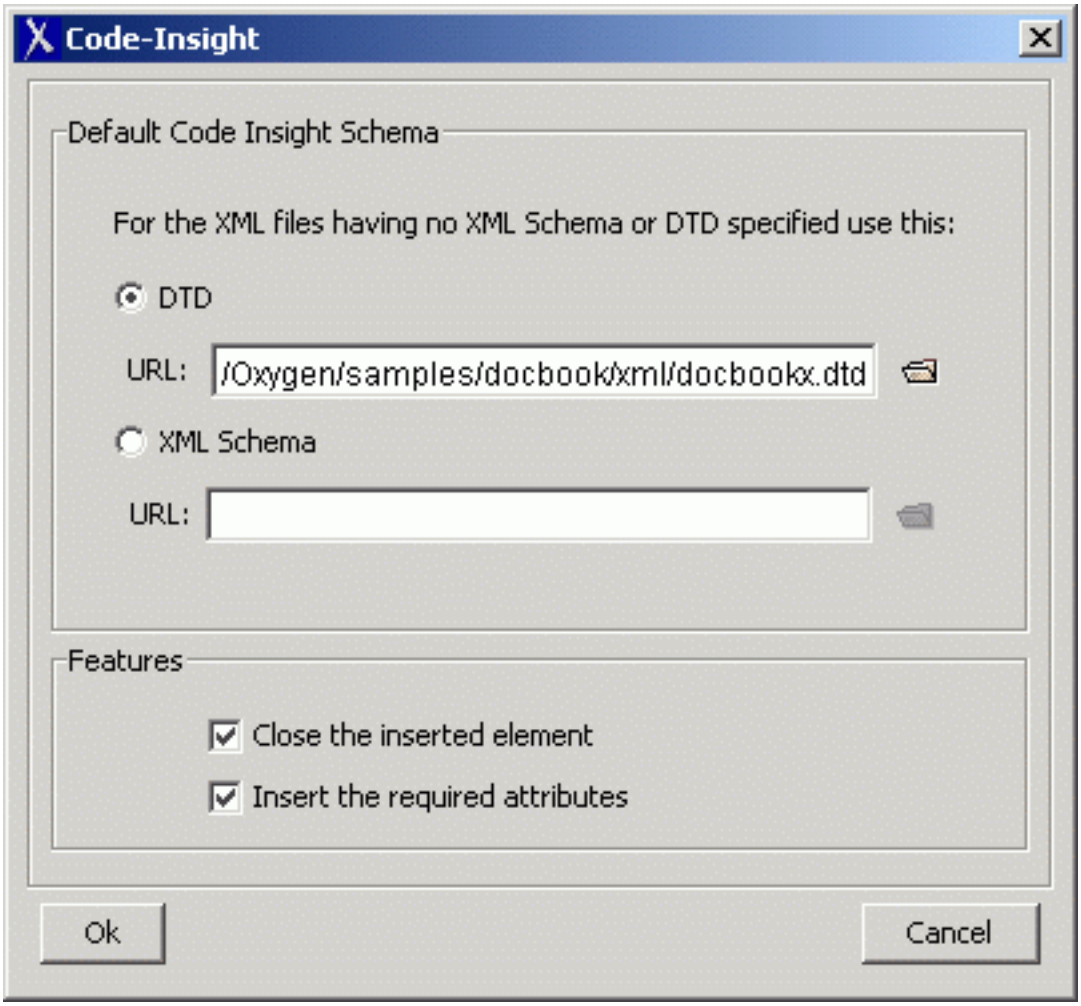
Changing the default DTD

From the Options menu, choose Code-Insight.

The displayed dialog has two sections. One for selecting the DTD or XML Schema, and other for controlling the automatic tag completion and attributes insertion.

If you are creating documentation with Docbook then is a good choice to set the docbookx.dtd file.

Figure 2.5. CodeInsight configuration dialog

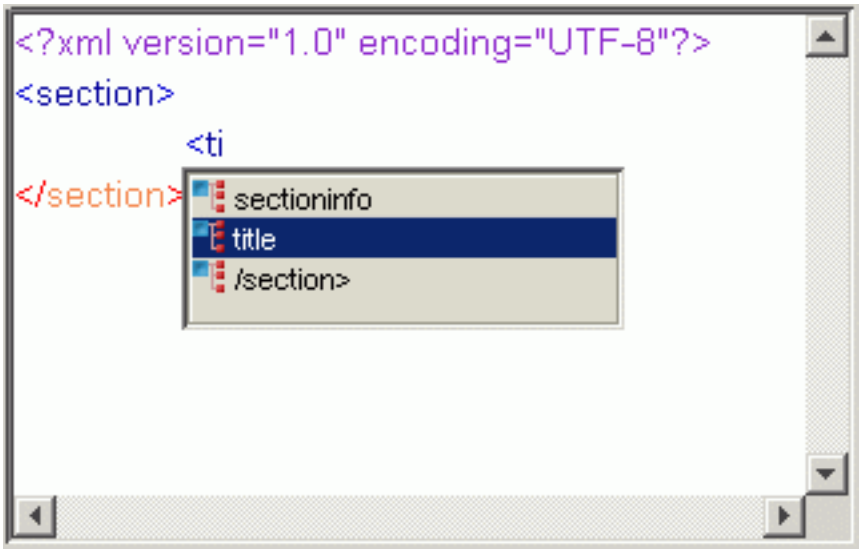


Creating a included file - a section.

Select File / New. Choose the XML type, but with no DTD.

Make sure that in the Code-Insight option you have choosed the correct DTD. Now you can type in the edited 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 insertable element names.

Figure 2.6. Code insight list over a document with no DTD



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.

Chapter 3. Using XPath

XPath references

The XPath standard can be read from http://www.w3.org/TR/xpath

A very good XPath tutorial is found at http://www.zvon.org/xxl/XPathTutorial/General/examples.html Has a lot of examples ordered by their complexity. You can try them with the editor.

Running an XPath expression

When editing XML, XSL, XSD documents the toolbar contains the XPath combo box. Here you can enter the XPath expression. Press the ENTER key to see the results.

Examples

- "/" - selects the document node
- "//title" - selects all the elements having the name "title"
- "//graphic/@align='left'" - selects all the elements having the name "graphic" and the attribute "align" equal to "left"

Applying XPath when editing XML, XSD documents

The resulted list of nodes is presented in a tab located at the bottom of the editor. When clicking on an item from the list, the text section corresponding to the selected node will be highlighted. Doubleclicking or pressing ENTER over the selection will move the cursor at the begining of the highlight.

When the query returned no results, a message box will indicate this.

Important

The XPath expressions can contain namespace prefixes, as they are defined in the document.

Example 3.1. Document defining a namespace. (A piece of XML Schema)

<?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'> <xs:element name="personnel">

A possible expression is: "//xs:element".

If you have a mix of elements with namespace proxy and elements without namespace proxy, you will have to prefix the last ones with the ":" string.

Example 3.2. Document defining namespaces without proxies.

```
<?xml version="1.0" encoding="UTF-8"?> <xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'> xmlns='http://oxygen.sync.ro'> <xs:element name="personnel"> <a>aaa</a>
```

In order to select all the "a" elements, the XPath expression is: "//:a".

Applying XPath when editing XSL documents

In this case the XPath expression is applied over the associated XML document, not on the edited XSL. This is useful when building the template "match" expressions, offering the possibility to test first the expression. You can associate an XML document to a stylesheet by using the "Transformation Configuration Dialog", the "XML URL" field. This is displayed when pressing the "T.." button from the toolbar.

The results are presented in a separate dialog.

Chapter 4. Questions and answers

4.1.

Question:

I open a file. I click on Options | Code Insight and established a DTD to use with the file.

I type

```
<?xml version="1.0" encoding="UTF-8"?>
```

type "<" nothing happens.

I type my known top level tag, then I type "<" it shows me the next level or a close tag. It appears to work. But when I hit Validate and it complains at the top level tag! [Message:E cvc-elt.1: Cannot find the declaration of element 'doXm'. SystemID:null, Line:2, Column:1]

Sorry but I don't really see what it is I'm supposed to do to get this editor to associate this new file with a particular DTD so that I can Validate.

Answer:

If you use File/New, and then choose to create an XML file, you will be asked to select a DTD and a root element for the new document.

A document like this one will be created:

```
<?xml               version="1.0"               encoding="UTF-8"?>
        <!DOCTYPE root SYSTEM "path to your dtd"> <root> </root>
```

If you type "<" between the tags you will get the list of possible elements that can be inserted. The editor automatically detects the used DTD and changes the CodeInsight model accordingly. You do not have to use the Options/CodeInsight feature for this.

Note that if the "root" element is declared in the DTD, the error you got will not appear any more.

Note

The menu Options/CodeInsight is used for selecting a DTD for CodeInsight when editing a document that cannot have a DOCTYPE declaration. This is the case of editing pieces of a large document that are included later in a larger document. (The larger document can be validated, and not its components)

Note

In order to check that a document is valid, you have to specify the DTD or XML Schema for it, using the DOCTYPE.

Example 4.1. Well formed, but not valid, since the root was not declared

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

Example 4.2. This can be validated, because the parser can search for the "root" in the DTD.

```
<!DOCTYPE root SYSTEM
       "some_dtd_declaring_the_root"> <root></root>
```