

# XMLmind XML Editor M1.3

## User's Guide

[www.xmlmind.com/xmleditor](http://www.xmlmind.com/xmleditor)  
[xmleditor-info@xmlmind.com](mailto:xmleditor-info@xmlmind.com)  
Pixware SARL  
Immeuble Capricorne  
23 rue Colbert  
78180 Montigny Le Bretonneux  
France  
Phone: (33) 01 30 60 07 00  
Fax: (33) 01 30 96 05 23

August 07, 2002

## Contents

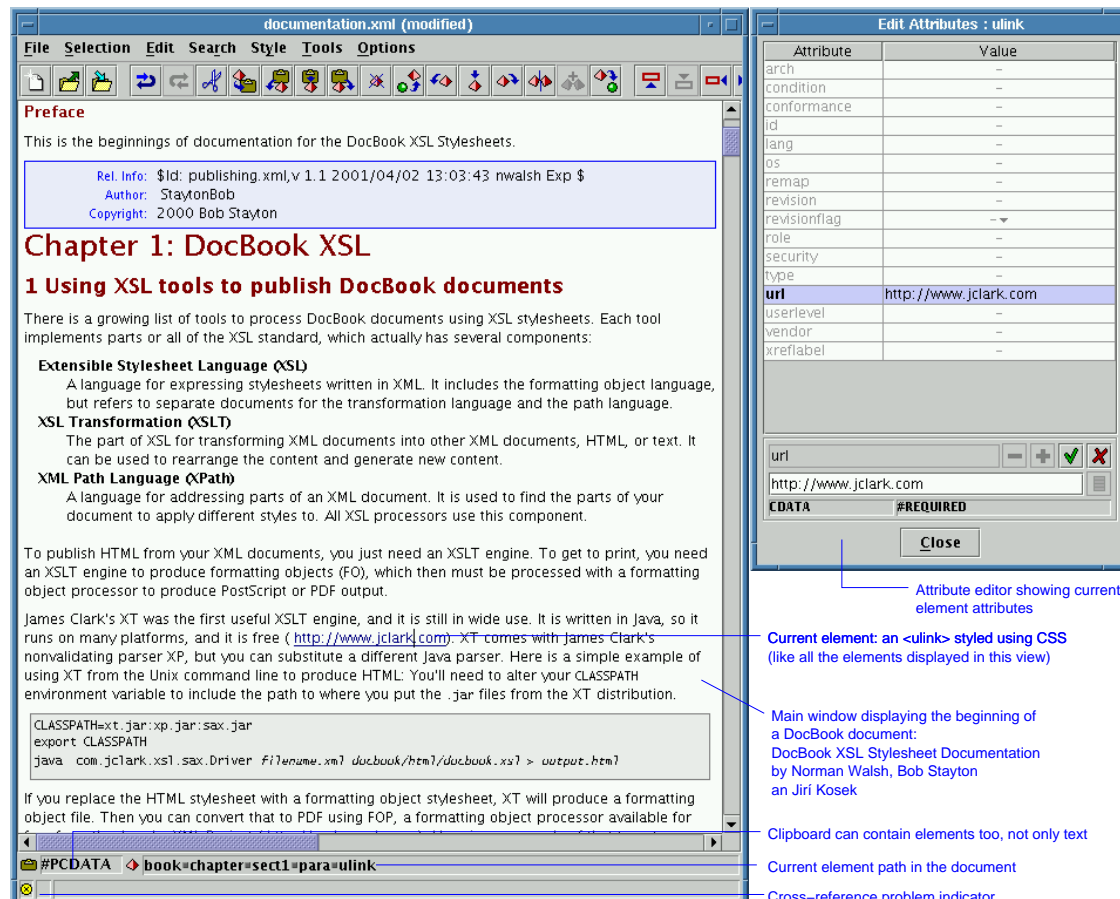
<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	What is XXE? . . . . .	3
<b>2</b>	<b>Features</b>	<b>4</b>
2.1	Features . . . . .	4
2.2	Non features . . . . .	5
<b>3</b>	<b>Install</b>	<b>6</b>
3.1	Installing XXE . . . . .	6
3.1.1	Requirements . . . . .	6
3.1.2	Install on Unix . . . . .	6
3.1.3	Manual install on Windows . . . . .	7
3.2	Content of the distrib/ directory . . . . .	7
3.3	Deploying XXE . . . . .	8
3.3.1	Creating new document templates . . . . .	9
<b>4</b>	<b>Tutorial</b>	<b>11</b>
4.1	Starting XXE . . . . .	11
4.2	Creating a new document . . . . .	12
4.3	Inserting elements . . . . .	14
4.4	Selecting elements . . . . .	17
4.4.1	Element and #PCDATA selection . . . . .	17
4.4.2	Text selection . . . . .	18
4.5	Navigating through elements . . . . .	19
4.6	Copy, cut, paste, delete . . . . .	21
4.6.1	Text selection . . . . .	21
4.6.2	Element and #PCDATA selection . . . . .	22
4.7	Splitting and joining elements . . . . .	24
4.8	Replacing elements . . . . .	27
4.9	Converting elements . . . . .	30
4.9.1	Text selection . . . . .	30
4.9.2	Element and #PCDATA selection . . . . .	31
4.10	Editing element attributes . . . . .	32
4.11	Checking document validity . . . . .	35
<b>5</b>	<b>Command reference</b>	<b>37</b>
5.1	File->Print . . . . .	37
5.2	Search->Replace . . . . .	37
5.3	Tools->CheckSpellingErrors . . . . .	38

5.4	Options->Options . . . . .	40
5.5	Options->Non-XMLFormats . . . . .	49
5.5.1	Using XXE to edit non-XML structured formats . . . . .	49
5.5.2	Registering a format plug-in with XXE . . . . .	49
5.5.3	The Javadoc plug-in . . . . .	52
5.5.4	The APT plug-in . . . . .	55
<b>6</b>	<b>APPENDIX A: CSS2 properties and values supported by XXE</b>	<b>56</b>
6.1	Restrictions . . . . .	56
6.2	Extensions . . . . .	59
<b>7</b>	<b>APPENDIX B: Whitespace processing</b>	<b>65</b>
<b>8</b>	<b>APPENDIX C: Keyboard shortcuts</b>	<b>66</b>
<b>9</b>	<b>APPENDIX D: Enhancements and bug fixes</b>	<b>69</b>
9.1	M1.3 Patch 3 (August 07, 2002) . . . . .	69
9.2	M1.3 Patch 2 (May 22, 2002) . . . . .	69
9.3	M1.3 Patch 1 (March 13, 2002) . . . . .	69
9.4	M1.3 (February 6, 2002) . . . . .	70
9.5	M1.2 Patch 5 (December 19, 2001) . . . . .	71
9.6	M1.2 Patch 4 (November 27, 1001) . . . . .	72
9.7	M1.2 Patch 3 (November 23, 2001) . . . . .	72
9.8	M1.2 Patch 2 (November 7, 2001) . . . . .	72
9.9	M1.2 Patch 1 (October 25, 2001) . . . . .	72
9.10	M1.2 (October 19, 2001) . . . . .	73
9.11	October 1, 2001 . . . . .	74
9.12	September 21, 2001 . . . . .	75
9.13	September 7, 2001 . . . . .	76
9.14	August 17, 2001 . . . . .	76
9.15	August 3, 2001 . . . . .	76

# 1 Introduction

## 1.1 What is XXE?

XMLmind XML Editor (XXE for short) is an XML editor featuring DTD-aware editing commands and a word processor-like view configured using W3C's cascading stylesheets (CSS).



Screen shot of XXE M1

XXE is a commercial product but for now, XXE is only available in *milestone versions*. Milestone 1 is the only version which has been released to the public. Milestone 1 is not a beta version of the future product: it is a prototype which has been created to validate some aspects of the product.

However, this prototype is rock solid and has enough features to allow technical writers to comfortably author XML documents. Today, XXE M1 is routinely used to create DocBook and XHTML documents as well as documents using proprietary DTDs almost as complex as DocBook.

XXE M1 can be used *at no charge* with a very liberal license. See XMLmind Standard XML Editor License.

The XXE product will too have a free edition (the Standard Edition as opposed to the non-free Professional Edition). So XXE M1 users not willing to pay to use an XML editor will not meet a dead end once the product will be released.

## 2 Features

### 2.1 Features

- XXE can be used with or without a CSS style sheet.

If a CSS style sheet has been attached to the document, a word processor-like view may be used to edit the document, otherwise a editable tree view is used.

- A substantial subset of CSS2 is supported, including tables, counters and generated content.
- Using the CSS2 extension **display: tree**, it is possible to mix styled and non-styled elements.
- Using the *replaced content* CSS2 extension, it is possible to embed specialized editors in the styled view. Replaced content is also routinely used to embed images.
- CSS styles contained in **@media print** constructs are automatically applied when printing the document being edited.
- XXE can be used with or without a DTD.

If the XML document has a DTD and is valid (conforms to that DTD), XXE will not suggest to the user any editing commands that would make the document invalid.

Without a DTD, the user is allowed to do whatever he wants, for example he can invent elements or attributes. XXE just ensures that the XML document remains well-formed.

- XXE can load an invalid XML document (that doesn't conform to its DTD).

In such case, after reporting all the validity problems to the user, it puts itself in repair mode (a lenient editing mode) until the document structure is fixed by the user.

#### Notes:

- XXE cannot load a non-well-formed document.
- XXE distinguishes 3 degrees of severity for the invalidity of a document:
  1. Invalid structure (the indicator at the bottom left of the main window is red).
  2. Invalid attributes (the indicator is orange).
  3. Cross reference problems (the indicator is yellow).

XXE enters in repair mode only when structural problems are found.

- XXE will probably have a very nasty behavior if the DTD is itself invalid. The validity of the DTD is never checked by XXE (planned for future product). It must be checked once for all, prior to be used within XXE, with a separate tool such dtdvalid which is part of the Xsdvalid Toolset.
- Multi-level undo/redo.
- Literal and regular expression search and replace.
- Spell checker with dictionaries for several languages.
- Out of the box, XXE has ready to use support for XHTML, DocBook, Simplified DocBook, Slides and Javadoc(TM).
- XXE should run on any platform supporting Java 1.3 and above. XXE is officially supported on Linux, Windows NT/2000/XP and MacOS X.
- XXE M1 distribution includes the full source code of the prototype.

## 2.2 Non features

This M1 release is just a prototype and therefore, some important functionalities have not been implemented. Here are some aspects of XXE M1 which may prevent you from using this version:

- XXE M1 is heavily document oriented. It is not very useful for editing XML data because:
  - It cannot be used to edit comments and processing instructions (planned for future product).
  - It doesn't understand namespaces (planned for future product).
  - It doesn't understand XML-Schemas (planned for future product: a preview of its XML-  
Schema validation engine can be downloaded from <http://www.xmlmind.com/xsdvalid.html>)
  - Element attributes are edited using a secondary window and you are forced to switch back and forth from the main document window to the attribute editor, which is rather tedious.
- The editing commands of XXE M1 are XML element oriented and therefore, pretty low-level for the average user (who may have hard times trying to think in terms of XML elements).
- XXE cannot be used to edit the physical structure of an XML document (entities, CDATA sections, marked sections, etc).

A well-formed XML document making use of these features can of course be loaded into XXE but as soon as the document is loaded, these constructs are expanded and forgotten.

- XXE loads the whole document in memory in order to be able to edit it. Therefore, you need a lot of RAM and a fast CPU if you intend to edit large documents.

## 3 Install

### 3.1 Installing XXE

#### 3.1.1 Requirements

- Sun or IBM Java 1.3 or above.
- At least 128Mb of memory and a 400MHz CPU.
- 20Mb of free disk space for a binary distribution, 25Mb for a source distribution, 50Mb for a self-contained distribution which includes a Java runtime.

*Note that some Java runtimes may be very slow and/or have very severe bugs. So if you experiment extreme slowness or weird behaviors (especially those related to the GUI) with XXE, do not hesitate to upgrade your Java runtime.*

XXE has been tested with:

- Sun Java 1.3.1 and 1.4 under SuSE Linux 7.2.
- Sun Java 1.3.0\_02 under Windows NT SP5.
- Mac Java 1.3.1 under MacOS X 10.1.

#### 3.1.2 Install on Unix

##### Procedure:

1. Unpack the XXE distribution somewhere.

```
$ cd
$ gzip -d -c xxe-m13p2-bin.tgz | tar xvf -
$ ls xxe-m13p2-bin
aptparser.jar
css/
dict/
docs/
dtd/
azcheck.jar
jaxp.jar
regexp.jar
sax2.jar
xp.jar
template/
xxe
xxe.bat
xxe.jar
...
```

2. XXE can be directly used from the xxe-m13p2-bin/ directory.

If for some reasons, you do not want to use XXE from its distribution directory, copy the xxe shell script, \*.jar and the dict/ subdirectory to a directory referenced in your path (example /usr/local/bin).

```
$ su
$ cp ~/xxe-m13p2-bin/xxe /usr/local/bin
$ cp ~/xxe-m13p2-bin/*.jar /usr/local/bin
$ cp -r ~/xxe-m13p2-bin/dict /usr/local/bin
$ chmod a+rx /usr/local/bin/xxe
$ chmod a+r /usr/local/bin/*.jar
$ chmod a+rw /usr/local/bin/dict
$ chmod a+r /usr/local/bin/dict/*
```

### 3.1.3 Manual install on Windows

Manual install on Windows NT is similar to the install on Unix but under Windows you'll have to copy `xxe.bat` rather than the `xxe` shell script.

You'll also have to manually edit this `.bat` file and modify the line:

```
set dist=D:\src\xmledit\distrib
```

by replacing `D:\src\xmledit\distrib` by the name of the actual directory containing the `xxe.bat` and all the `.jar` files.

## 3.2 Content of the distrib/ directory

**docs/** Contains this user guide in HTML and PDF (Acrobat) formats.

The DocBook (`userguide.xml`) and XHTML (`userguide.xhtml`) versions are provided too because it may be useful to load them into XXE in order to evaluate the editor.

**dict/** Contains the dictionaries used by the spell-checker. A dictionary is a Jar file whose name is `LL.jar`, where `LL` is an ISO code for a language.

Dictionaries downloaded from the XMLmind Web site must be copied to this subdirectory.

**dtd/** Contains 3 DTDs (Document Type Definition):

- **docbook/** is the official DocBook XML DTD V4.1.2.

See <http://www.oasis-open.org/docbook/xml/>.

**Modifications:**

- Defined the `xml:space` attribute as:

```
xml:space (default|preserve) #FIXED 'preserve'
```

for the following elements: `address`, `funcsynopsisinfo`, `literallayout`, `programlisting`, `programlistingco`, `screen`, `screenco`, `synopsis`.

- Reordered some definitions in the DTD to have more useful newly inserted element templates.

Element	Old order	New order
table	(graphic+ mediaobject+	(tgroup+ graphic+
informaltable	tgroup+)	mediaobject+)

- **sdocbook/** is Norman Walsh's Simplified DocBook XML DTD V4.1.2.5.

See <http://www.oasis-open.org/docbook/xml/simple/>.

**Modifications:** same as for `docbook`.

- **slides/** is Norman Walsh's Slides XML V2.0 DTD.

See <http://sourceforge.net/projects/docbook/>.

**Modifications:** same as for `docbook`.



- `xhtml/` is the XHTML 1.0 DTD.  
See <http://www.w3.org/TR/xhtml1/>.

**Modifications:**

- Fixed the definition of `xml:space` for element `pre`. The buggy definition was:

<code>xml:space</code>	<code>(preserve)</code>	<code>#FIXED 'preserve'</code>
------------------------	-------------------------	--------------------------------

- Reordered some definitions in the DTD to have more useful newly inserted element templates.

Element	Old order	New order
table	... (tbody+ tr+)	... (tr+ tbody+)
tr	(th td)+	(td th)+

**css/** Contains 2 CSS style sheets:

- `xhtml.css`, a style sheet for XHTML 1.0.
- `docbook.css`, a tentative style sheet for DocBook V4 (simplified or not).

**template/** Contains 4 document templates:

- `html.xhtml` can be used to create an XHTML page.
- `chapter.docb` is a DocBook chapter template.
- `article.docb` is a DocBook article template.
- `article.sdocb` is a simplified DocBook (much faster to load than full DocBook) article template.

**xxe, xxe.bat** Scripts used to start XXE. (Use `xxe` on any Unix system. Use `xxe.bat` on Windows NT.)

**\*.jar** All the (non-system) Java class libraries needed to run XXE:

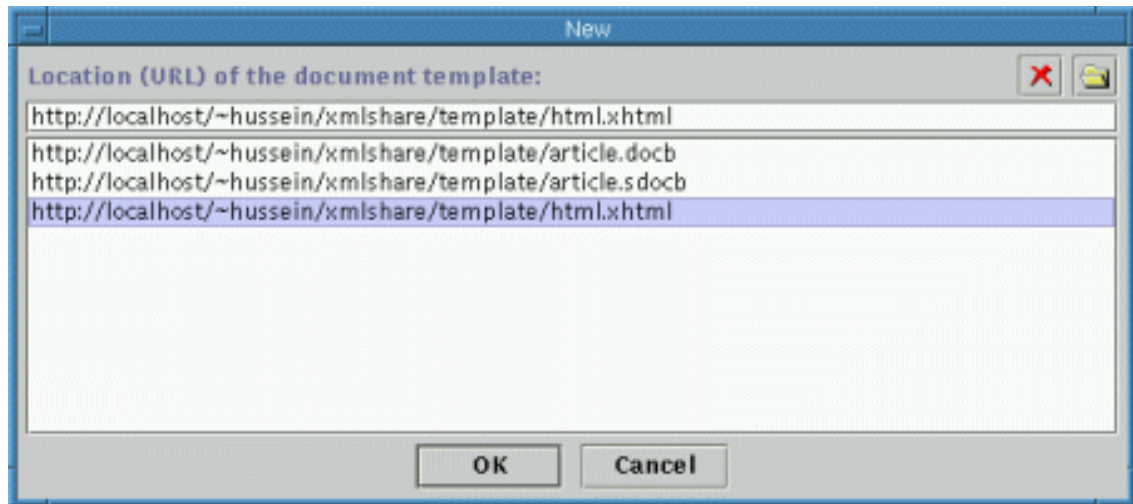
- `xxe.jar`, contains the code of XXE.
- `aptparser.jar`, contains the code of the APT parser.  
This code is required only if you use the APT format plug-in for XXE. (This is our case here at Pixware because all the XXE documentation is written in APT.)
- `jaxp.jar`, `sax2.jar`, `xp.jar` are needed to parse XML.  
These excellent packages have *not* been developed by XMLmind. Copyright information is contained in the corresponding `.LICENSE` file. Read the corresponding `.README` file to have more details about these packages.
- `regex.jar` contains an implementation of regular expressions for Java.  
This excellent package has *not* been developed by XMLmind. Copyright information is contained in the corresponding `.LICENSE` file. Read the corresponding `.README` file to have more details about this package.
- `azcheck.jar` is a beta version of AZcheck, a not-yet-released pure Java spell-checker.  
Copyright information is contained in the corresponding `.LICENSE` file. Read the corresponding `.README` file to have more details about this product.

### 3.3 Deploying XXE

- The `dtd/`, `css/`, and `template/` directories are intended to be shared. So it is recommended to copy them in a directory which is accessible using an HTTP server.

It is important to do so because all documents saved by XXE contain *absolute* file or http URLs to their DTD and style sheet.

- Add your own DTDs, CSS style sheets, and XML document templates to this shared repository.
- Tell the XXE users to get their document templates from this shared repository using the File->New menu command (in the following figure, the shared repository is `http://localhost/~hussein/xmlshare/`):



*The dialog box displayed by the File->New menu command*

All XXE options and user preferences are persistent (stored in file `~/ .xxe` under Unix and in file `C:\winnt\Profiles\<user>\xxe.ini` under Windows NT) so specifying the location of a document template is done once for all.

### 3.3.1 Creating new document templates

A document template is simply an almost empty document containing:

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type="text/css" href="../css/xhtml.css"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"../dtd/xhtml1/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head><title></title></head>
  <body><p></p></body>
</html>
```

1. The XML declaration (line 1).
2. The xml-stylesheet Processing Instruction (PI) used to associate a style sheet to an XML document. (line 2).

This PI is specified in the W3C recommendation: *Associating Style Sheets with XML documents* (<http://www.w3.org/TR/xml-stylesheet/>).

Using several of these PIs, it is possible to associate alternate style sheets to an XML document, each one having its own descriptive title. Example:

```
<?xml-stylesheet href="bigfonts.css" type="text/css" alternate="yes"
title="Big fonts" ?>
<?xml-stylesheet href="colorful.css" type="text/css" alternate="yes"
title="Important things in red" ?>
<?xml-stylesheet href="normal.css" type="text/css" ?>
```

3. The document type declaration (lines 3-4, note that the root element is specified to be `html`).
4. The skeleton of the root element specified in the document type declaration (line 5 to end).

This skeleton must contain a valid content.

Such template must be created using a text editor. After creating it, validate its content by loading it in XXE as an existing document (i.e. using the File->Open menu command).

## 4 Tutorial

### 4.1 Starting XXE

- On Unix, XXE can be started by typing `xxe` from an xterm, optionally followed by the name of an XML document.

Example:

```
$ xxe &
```

- If you have installed an auto-installable Windows distribution (`setup.exe`), XXE can be started by double-clicking on the icon of `XMLmind_XML_Editor.exe` or by using the XXE shortcut added to the **Start** menu.
- If you have installed a zipped Windows distribution, XXE can be started by typing `xxe` (`xxe.bat`) from the command prompt, optionally followed by the name of an XML document.

Example:

```
C> xxe C:\xxe\docs\userguide.xml
```

#### Tips:

- If you do not want to read this tutorial (definitely not recommended :-)) and just want to start experimenting with XXE, you can load the full user guide (a 50+ pages document with 50+ screen dumps) into XXE.

You'll find it in file

`<istrib>/docs/userguide.xml` (DocBook format)

and in file

`<istrib>/docs/userguide.xhtml` (XHTML format),

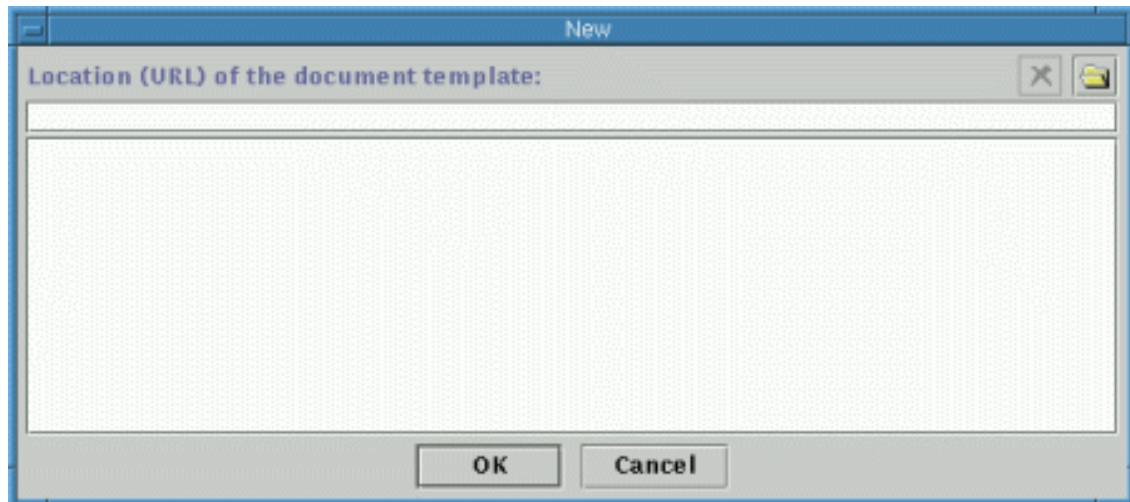
where `<istrib>` is the location of your XXE distribution.

The slowness of loading the XXE user guide comes from the fact that:

1. all the screen dumps are converted to 400x200 "thumbnails" (explained below in this tutorial);
2. the DocBook DTD is huge (that's why the Simplified DocBook XML DTD has been created by Norman Walsh).

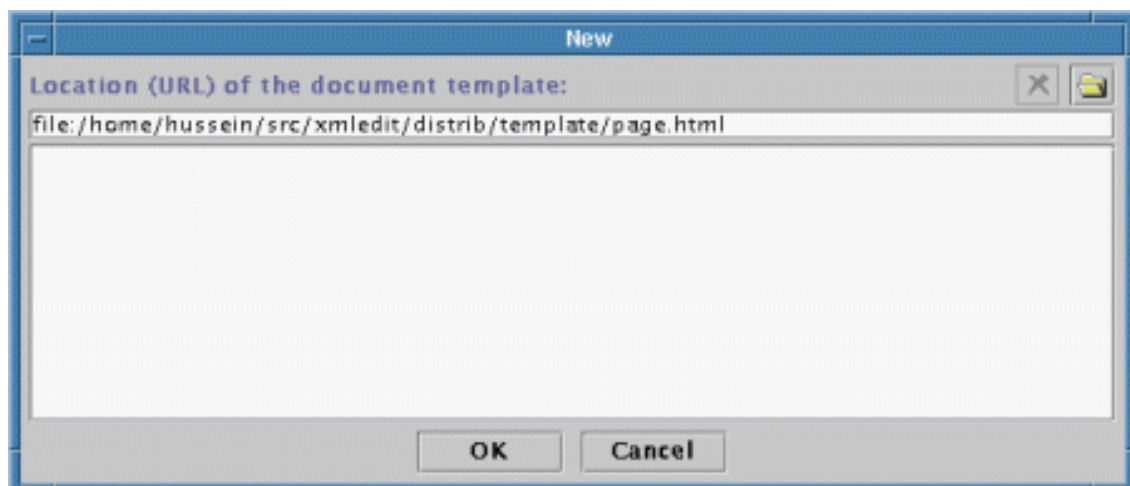
## 4.2 Creating a new document

A new document is simply a copy of an existing document template chosen using the File->New dialog box:



*The File->New dialog box, the first time XXE is started, may contain no document templates*

Click on the icon at the top right of the dialog box and select the `page.html` template contained in the `template` sub-directory of your XXE distribution. (In the following figure, the distribution directory is `/home/hussein/src/xmledit/distrib`):



Note that the `file:` prefix has been automatically prepended to the file name selected using the standard file chooser dialog box. It is also possible (and even recommended) to specify templates hosted by a Web server. In which case, you'll need to enter the full URL of the template including the `http://` prefix.

User preferences are persistent (stored in `~/.xxe` on Unix and in `C:\winnt\Profiles\<user>\xxe.ini` on Windows), so adding templates to the File->New dialog box is done once for all.

### Tips:

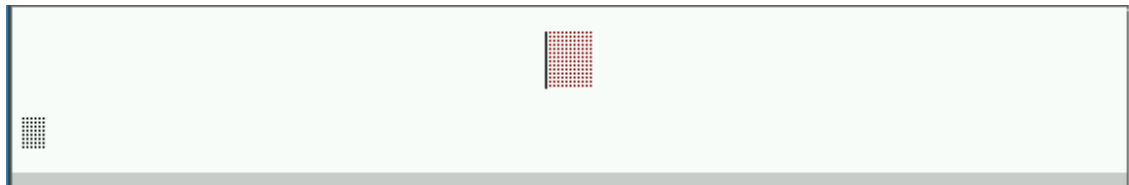
- An alternate way to create a new document is to use the File->OpenAsTemplate menu command.

This command automatically creates a “empty” document by copying the DTD, style sheets and root element (just its attributes, not its content) from the chosen document.

Example: Use this command to select the `userguide.xhtml` file contained in the `docs` subdirectory of your XXE distribution.

### 4.3 Inserting elements

The newly created document (`Untitled.xhtml`) looks like this:



The “blobs” are placeholders for text. Click on the first placeholder and type the title of your XHTML page. Click on the second placeholder and type a few words.



Click again anywhere on the title and then anywhere in the first paragraph. You'll notice that, at the top of the window, some tool bar icons change their state from disabled (grayed) to enabled (colored)



and at the bottom of the window, the *selected element path tool* changes its label.



The path tool shows precisely which element is currently selected, i.e. the document object you can act upon using the Edit menu commands (or their tool bar icon equivalent).

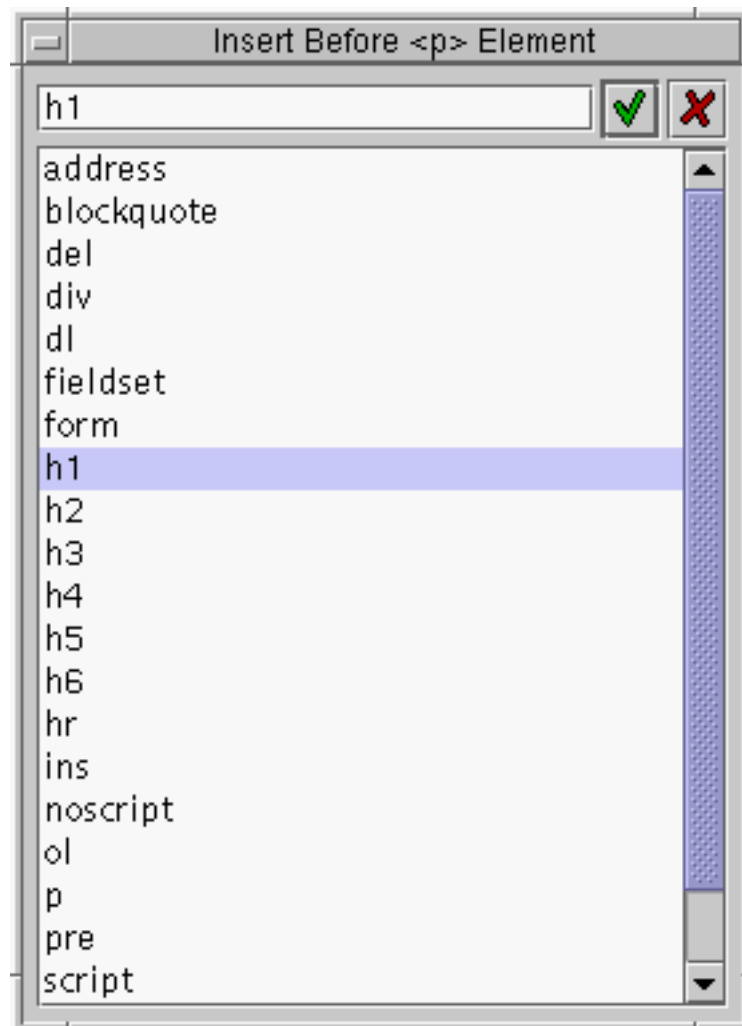
The path tool does so by showing the type name (**p**, **title**, etc) of the selected element and the type name of each of its “ancestors” in the containment hierarchy (**body**, **head**, **html**, etc).

In this section, we'll get familiar with only three Edit menu commands:



1. Insert before selected element
2. Insert inside selected element, at text cursor location
3. Insert after selected element

Click anywhere inside the paragraph and click on the Insert before icon. A dialog box shows up, listing all the element types you can insert before a **p** (the element type name, also called the *tag*, of a paragraph is called **p** in XHTML).



*The tag chooser dialog box*

Select tag **h1** and type the text of the heading.

Click inside the **p** before a word and click on the Insert inside icon, the same tag chooser dialog pops up listing all the inline elements you can insert inside a **p**. Select tag **strong** and type a few words in bold font.

Your document should look like this:





This time, use the Insert after icon and add a **ul** (unordered list) after the **p**. Type the text of the first list item (**li**).



(If you are in the **strong** element -- check it with the path tool -- click in the **p** outside the bold words because inserting an **ul** inside a **p** after a **strong** is not allowed by the XHTML DTD.)

Using what you have learned, add two more **lis**: one before the first **li** you have created, the second after it.

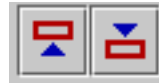


#### Tips:

- The keyboard shortcuts for the three insert commands, Ctrl-H, Ctrl-I, Ctrl-J, are easy to remember: **I** is for **I**nsert, **H** is before **I**, **J** is after **I**.
- The tag chooser dialog has a tag auto-completion feature à la emacs: type the first characters of a tag, *then type a space*, to list all the allowed tags beginning with these characters. Repeat until the tag you intend to insert is completely displayed in the text field at the top of the dialog.
- The Insert key may be used to insert a placeholder for text (called **#PCDATA**, see below) after the selected element (if such insertion is allowed).  
Shift-Insert inserts a placeholder for text before the selected element (if such insertion is allowed).  
Ctrl-Insert inserts after the selected element an element of the same type (if such insertion is allowed).  
Ctrl-Shift-Insert inserts before the selected element an element of the same type (if such insertion is allowed).

## 4.4 Selecting elements

### 4.4.1 Element and #PCDATA selection



*The Select Parent and Select Child tool bar icons*

Clicking on some text moves the text cursor (AKA caret) to the mouse click location and *implicitly* selects the element containing the piece of text.

It is also possible to *explicitly select* elements.

Click on the first **p** outside the **strong** and then click on the Select Parent tool bar icon (or type Ctrl-Up arrow).



Explicitly selected objects are surrounded by a red line.

The path tool indicates that **#PCDATA** is selected. **#PCDATA** is not an element type name, it is used to specify an anonymous piece of text contained in some elements like **p**.

Click on the **strong** element and type Ctrl-Up as long as the selected object changes and observe what is displayed by the path tool: first the **#PCDATA** contained in the **strong** is selected, then the **strong** itself, then the **p**, then the **body** containing the **p** and finally **html** itself.

Click on the Select Child tool bar icon (or type Ctrl-Down arrow) as long as the selected object changes and observe what is displayed by the path tool.

Selecting objects by directly clicking on them is possible too. There are two ways to do so:

1. Click in the path tool on the element type name you want to select.  
Example: try to click in the path tool on the word **body**.
2. Ctrl-click-1 on some text always selects the corresponding **#PCDATA**, which is rarely what you want to do. Ctrl-click-1 on the margin or on anything not editable (such as the bullet of a **li**) surrounding an element generally selects this element.

Example 1: Ctrl-click-1 on the text of the title. Then Ctrl-click-1 on the left or right margin surrounding the text of the title. Notice the difference.

Example 2: Ctrl-click-1 on the text of a **li**. Then Ctrl-click-1 on the bullet of the **li**. Notice the difference.

Note that If you do not click near text (for instance, if you click on an image or on generated content), you don't need to hold the Ctrl key pressed while clicking. In such case, there is no ambiguity and a plain mouse click is sufficient.

#### 4.4.2 Text selection

The customary text selection found in all word processors is also available in XXE.

Click in the middle of the **title** and drag the mouse across the document until the middle of the second **li** is reached.



Note that the path tool displays **html.head.title.#PCDATA**, the place where the text selection begins.

Clicking on some text or using the Escape key cancels the explicit selection whatever its nature.

In summary, you can use 3 different types of selection:

1. The implicit text container element selection.
2. The explicit **#PCDATA** or element selection.
3. The text selection.

All these types of selection differ by the type of editing commands you can apply to them. Only one type of selection is possible at a time.

#### Tips:

- Ctrl-click in the path tool on an element type name selects the corresponding element and inserts after the selected element an element of the same type (if such insertion is allowed).  
Shift-click in the path tool on an element type name selects the corresponding element and inserts before the selected element an element of the same type (if such insertion is allowed).
- The tool bar also contains icons that can be used select the preceeding sibling and the following sibling of the selected element.



The keyboard shortcuts for these commands are Shift-Ctrl-Up and Shift-Ctrl-Down.

- All the mouse and keyboard actions related to the text selection and found in most word processors are also available in XXE.

Example: Using the Shift-Right arrow key extends the text selection to the next character.

All these actions are listed in APPENDIX C: Keyboard shortcuts.

## 4.5 Navigating through elements

The Tab key may be used to move the caret from the current **#PCDATA** to the beginning of the next one. Shift-Tab moves the caret from the current **#PCDATA** to the beginning of the previous one.

Click on the title and use Tab and Shift-Tab to move the caret from one **#PCDATA** to the other.

Inside an element which allows tab characters to be inserted in the text flow just as any ordinary character (example: an XHTML **pre**), you must use Ctrl-Tab and Shift-Ctrl-Tab instead.

The simplest way to move the caret is of course to use the Left or Right arrow keys.

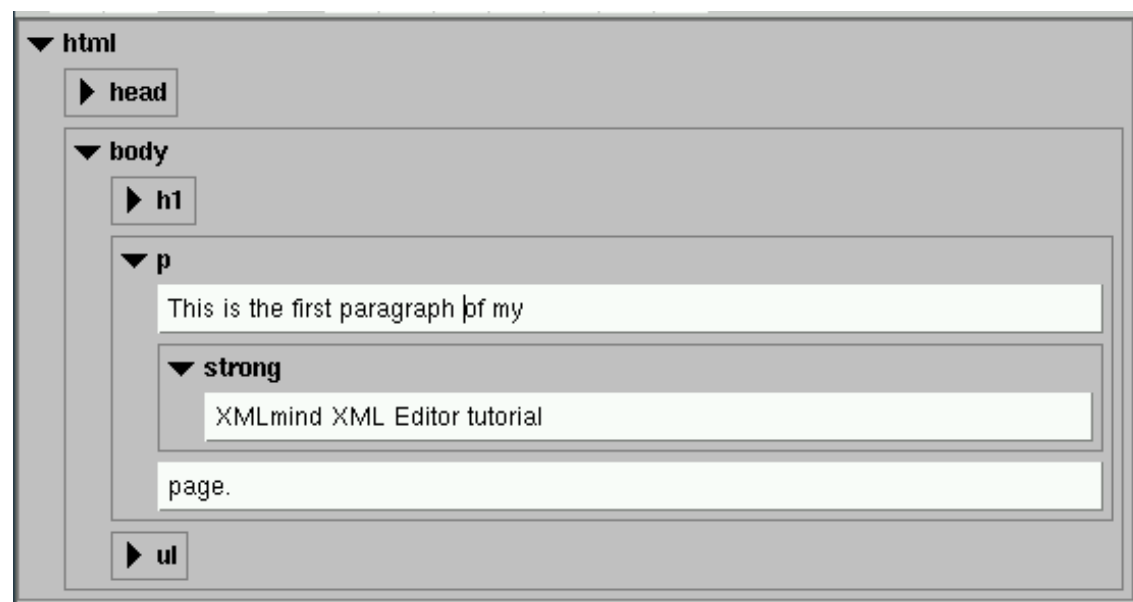
Click on the first **p** to the left of the **strong** and type on the Right key to move the caret in the direction of the **strong**.

Just before reaching the **strong** element, you'll notice that typing on the Right key has caused no perceptible caret movement. Then after this "dead" Right key, the caret seems to move as expected.

Go back to the left using the Left arrow and at the **p/strong** boundary, you'll notice a "dead" Left key then the caret seems to move as expected.

This is not a bug. On the **p/strong** boundary, the caret makes no visible movement but the path tool displays different implicitly selected elements (**p** and **strong**).

Use menu command Style->NoStyleSheet to view the document using a low-level hierarchical view.



*In this figure, for more compactness, the **head**, **h1**, and **ul** elements have been collapsed by clicking on the arrow located at the left of each element type name*

Repeat what you have done with the Right and Left arrow keys and you'll notice that with the tree view, there is a visible caret movement from the end of the **p.#PCDATA** to the beginning of the **strong.#PCDATA** and vice-versa.

Switch back to the styled view using the Style->xhtml command (xhtml is the name of the CSS style sheet used by the document template).

This "dead" key behavior also occurs when using the Del and Backspace keys.

Click on the first **p** to the left of the **strong** and type on the Del key. Notice what happens on the **p/strong** boundary. Then use Edit->Undo (Ctrl-Z) to undo this typing.

Click inside the **strong** and type on the Backspace key. Notice what happens on the **p/strong** boundary. Then use Edit->Undo to undo this typing.

**Tips:**

- The Up and Down arrow keys may be used to move the caret from one line to the other. But because the internals of XXE are not line-based, caret movements may occasionally be surprising (generally inside tables).
- All the keyboard actions related to the caret movement and found in most word processors are also available in XXE.

Example: Using the Ctrl-Right arrow key moves the caret to the beginning of the next word.

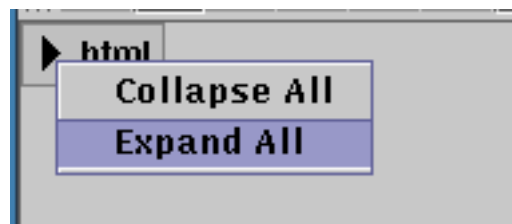
All these actions are listed in APPENDIX C: Keyboard shortcuts.

- If sometimes you do not understand XXE behavior, try to switch to the tree view using the Style->NoStyleSheet menu command. Using this low-level hierarchical view, things tend to be much easier to understand.

Note that everything said in this tutorial also applies to the tree view.

Despite the fact that, for the user, this view looks fundamentally different from the styled view, for XXE, these views are not different at all (because XXE does not work at the view level but at the document elements level).

- In a tree view, Click with the right button of the mouse on the arrow located at the left of each element type name to pop up a contextual menu with ExpandAll and CollapseAll commands.



## 4.6 Copy, cut, paste, delete



The Cut, Copy, Paste before, Paste, Paste after, Delete tool bar icons

### 4.6.1 Text selection

When applied to the text selection, the Edit->Copy menu command copies all the selected characters to the system clipboard. It is then possible to paste these characters in any other application including XXE itself. Select the characters displayed in bold font from the first **p**, copy them to the clipboard using the Copy tool bar icon, click on the **title** and paste the copied characters using the Paste tool bar icon.



After copying the text selection, you'll notice that the *clipboard tool* at the bottom of the XXE windows displays **#PCDATA**: a piece of text is stored in the clipboard.



Edit->Cut and Edit->Delete basically work the same way: they delete all the selected characters from the document. The only difference is that Edit->Cut copies the deleted characters to the clipboard.

Note that these commands also *eagerly delete empty text container elements* contained in a parent text container element.

Click in the middle of the **title** and drag the mouse across the document until the middle of the second **li** is reached. Then click of the Delete tool bar icon.



The **strong** element in the first **p** has been deleted because it was an empty text container element contained in a text container element.

The **h1**, **p**, **li**, while being empty text container elements, have not been deleted because they are contained in a non text container element (**body**, **ul**).

Use Edit->Undo (Ctrl-Z) to undo this command.

#### 4.6.2 Element and #PCDATA selection

The Copy, Cut, Paste, Delete commands can operate on the selected element (whether implicitly or explicitly selected) or **#PCDATA** too.

For this type of selection, two more commands are available: Paste before and Paste after.

Select the first **li**. Use the Cut tool bar icon to move it to the clipboard. Select the last **li**. Use the Paste after tool bar icon to paste the **li** stored in the clipboard after the last **li**.



After copying the first **li**, you'll notice that the clipboard tool at the bottom of the XXE windows displays **li**: a list item element is stored in the clipboard.



**Tips:**

- The Del key may be used to delete the text selection but for the element or **#PCDATA** selection, you'll have to type Ctrl-K (**K** like **K**ill).

(With the element or **#PCDATA** selection, typing on the Del key just deletes the next character.)

- The keyboard shortcuts for the Copy, Cut, Paste commands are those used in most applications: Ctrl-C, Ctrl-X, Ctrl-V.

The keyboard shortcuts for the Paste before and Paste after, Ctrl-U and Ctrl-W, are easy to remember: **U** is before **V** and **W** is after **V**.

- Cutting and pasting elements between two instances of the XXE application is possible.
- When the clipboard is filled by an external application, the new clipboard content does not automatically appear in the clipboard tool at the bottom of the XXE window.

You'll need to manually "refresh" the tool bar and status bar in order to see the new clipboard content. For example, click on another element to implicitly select it. This way, the editing context changes and the tool bar and status bar need to be refreshed.



## 4.7 Splitting and joining elements

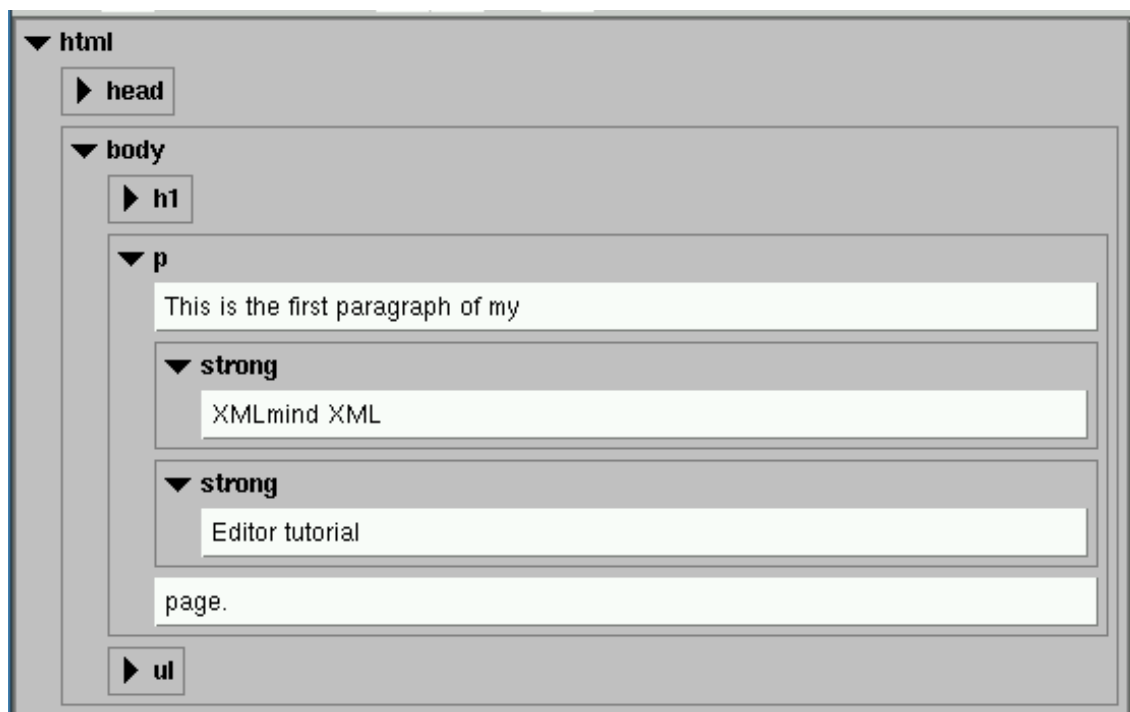


*The Split and Join tool bar icons*

The Edit->Split command may be used to split the selected text container element in two, the split point being specified by the caret position. This command will not operate on any other selected object.

Click in the middle of the **strong** contained in the first **p**. Use the Split tool bar icon to split the **strong** in two. Nothing visible happens.

There is no bug here. Switch to the tree view using Style->NoStyleSheet to see what really happened.



Two adjacent **strong** elements have been created and the boundary between these two elements is not visible using the style view.

Use Edit->Undo (Ctrl-Z) to undo the split and switch back to the styled view using Style->xhtml.

Click in the middle of the **strong** contained in the first **p**. Use Ctrl-Up to explicitly select the **p**.



Type Ctrl-Enter (the keyboard shortcut for Edit->Split) to split the **p** containing the **strong** in two.



The Edit->Join (keyboard shortcut Ctrl-Backspace) is the inverse command of Edit->Split. It concatenates the selected element with the preceding one, if this preceding element is of the same type (i.e. you cannot join a **p** with a **pre**).

When two elements are joined, if the first child element of the selected element and the last child of its preceding element can be joined, they are joined too. And this happens at all nesting levels to really make Edit->Join the inverse command of Edit->Split.

Select the **p** created by the split command described above, for example by clicking on it and then using the selected element path tool.



Type Ctrl-backspace to join this **p** to the previous one, then undo the command by typing Ctrl-Z.

**Tips:**

- If you want to insert a new paragraph after another paragraph, click near the end of the paragraph (the caret is then positioned after the last character of the paragraph), then type Ctrl-Enter.
- When there is no ambiguity, typing Enter (rather than Ctrl-Enter) splits the selected element in two parts.

Example: typing Enter when an XHTML **p** is the selected element, splits the **p** in two parts, while typing Enter when an XHTML **pre** is the selected element, inserts a linefeed character into the **pre**.

## 4.8 Replacing elements

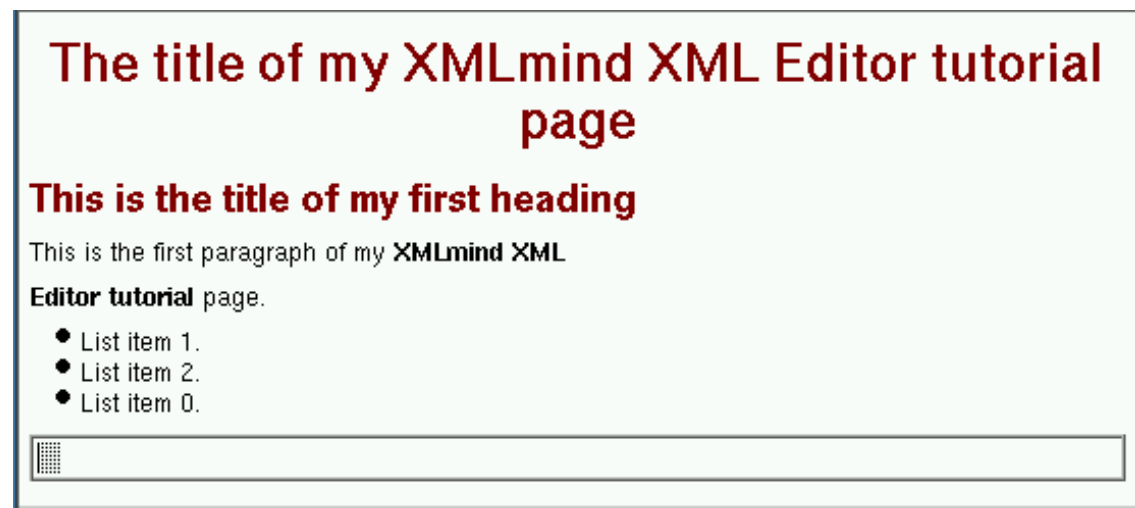


*The Replace tool bar icon*

The Edit->Replace command is equivalent to deleting the selected element or #PCDATA and inserting a new element or #PCDATA which replaces the deleted element.

This command is required because it is often not allowed to delete the selected element: doing so would create an invalid document.

Select the **ul**. Insert a **table** after it.

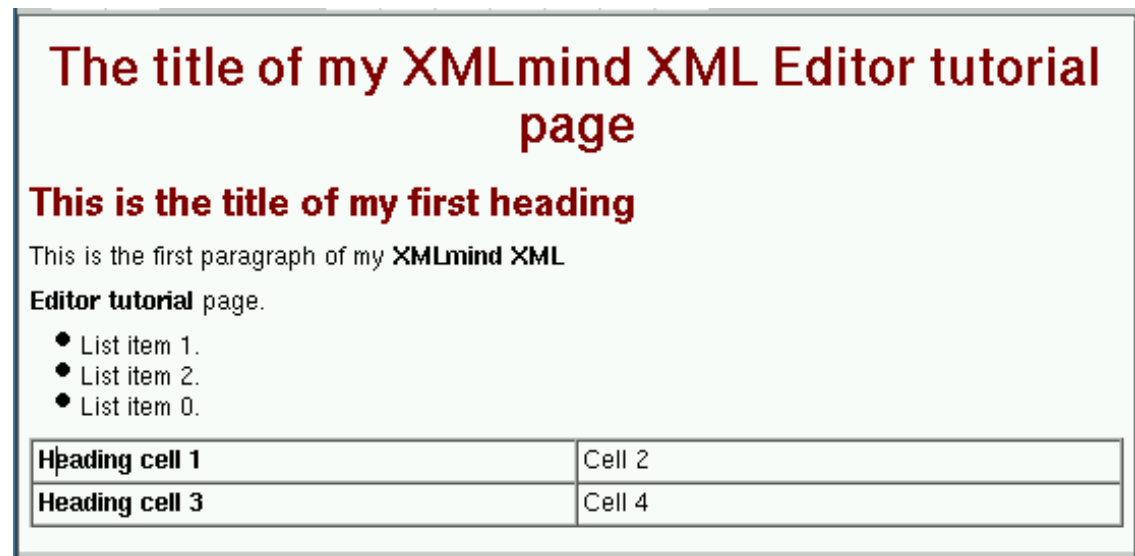


The table is created with a single **tr** (row) containing a single **td** (cell). This is often what you want as a **table** template but not always.

Select the **tr**. Use the Replace tool bar icon to replace it with a **tbody**.



As an exercise, select the **td** contained in the **tr** contained in the **tbody** and replace with a **th** (another type of cell). Then add one more row. Each row must contain a **th** followed by a **td**.

**Tips:**

- Note that beside **tbody**, the tag chooser dialog also suggests to replace a **tr** with another **tr**. This behavior is useful to replace a filled element with an empty one of the same type.

## 4.9 Converting elements



*The Convert tool bar icon*

### 4.9.1 Text selection

When applied to the text selection, the Edit->Convert command

1. deletes all the selected characters (like Edit->Cut or Edit->Delete, it also eagerly deletes empty text container elements contained in a parent text container element);
2. puts all the deleted characters in the chosen element or **#PCDATA** (the target of the conversion);
3. inserts this newly created text container at the location where the text selection starts.

As a simple example, select a non-bold word in the first **p**.

This is the first paragraph of my XMLmind XML

Use the Convert tool bar icon to convert it to **em** (emphasis).

This is the first *paragraph* of my XMLmind XML

As a harder to understand example, select some text from the middle of the **strong** in the second **p** to the end of this **p**.

Editor tutorial page.

Use the Convert tool bar icon to convert the text selection to **strong**. But there is a trick here: because the text selection starts in the **strong**, you must specify **#PCDATA** when prompted by the tag chooser dialog.

Editor tutorial page.

Note that the tag chooser dialog proposes **strong** as a valid choice, because in XHTML, it is possible to have a **strong** contained in a **strong**.

#### 4.9.2 Element and #PCDATA selection

Converting a **#PCDATA** rather than an equivalent text selection is often more convenient.

Select the **#PCDATA** contained in the second cell of the table.

Heading cell 1	Cell 2
Heading cell 3	Cell 4

Type Ctrl-T (the keyboard shortcut of Edit->Convert, **T** is for **T**ranslate) to convert it to **em**.

Heading cell 1	Cell 2
Heading cell 3	Cell 4

Unlike Edit->Replace which creates empty elements, the Edit->Convert command transfers the content of the source element (but not its attributes) to the target converted element.

Select the **ul**.

- List item 1.
- List item 2.
- List item 0.

Type Ctrl-T to convert it to an **ol** (ordered list).

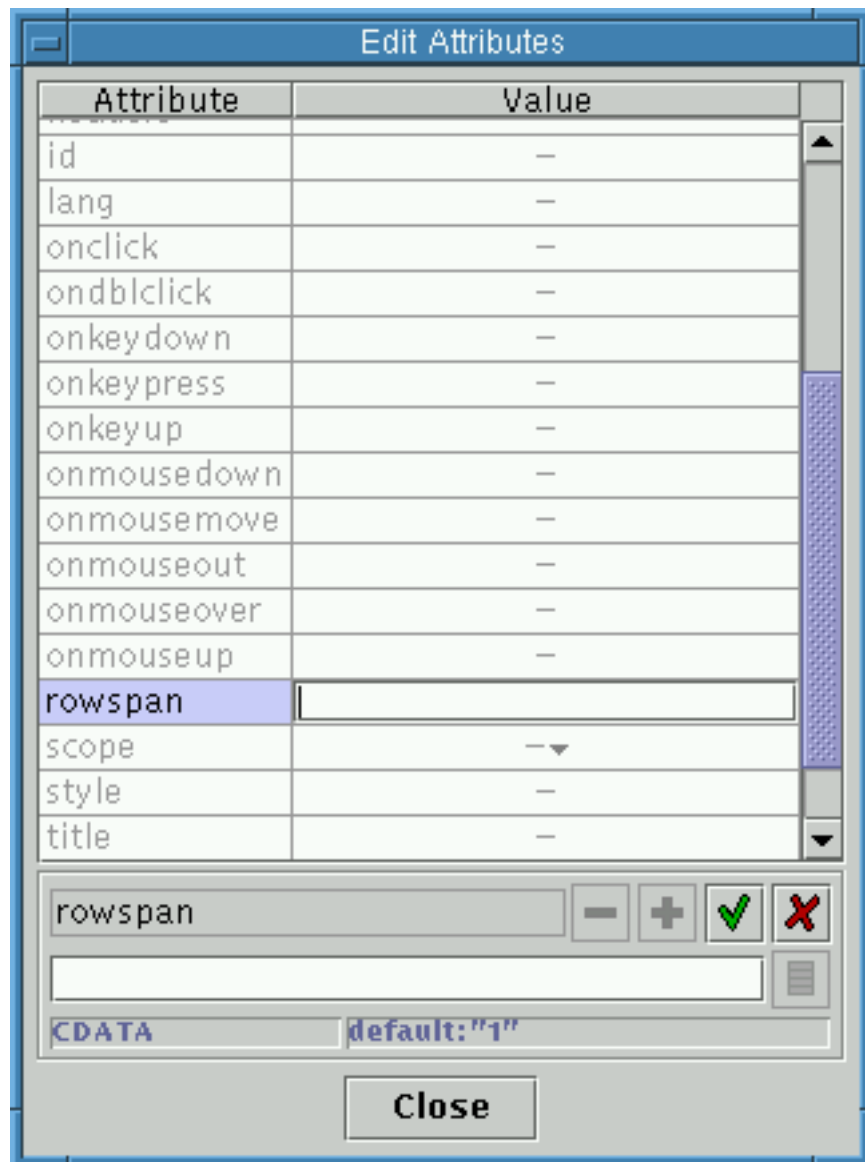
- 1 List item 1.
- 2 List item 2.
- 3 List item 0.

This operation is valid because the **ul** parent, a **body**, accepts **uls** as well as **ols** at this place and because the element content of this **ul** is “compatible” with an **ol**.



#### 4.10 Editing element attributes

Use Tools->EditAttributes (keyboard shortcut Ctrl-E) to edit the attributes of the selected element.



*The Attribute Editor dialog box with the **rowspan** attribute of a **td** being edited*

This dialog box can be quickly described as follows:

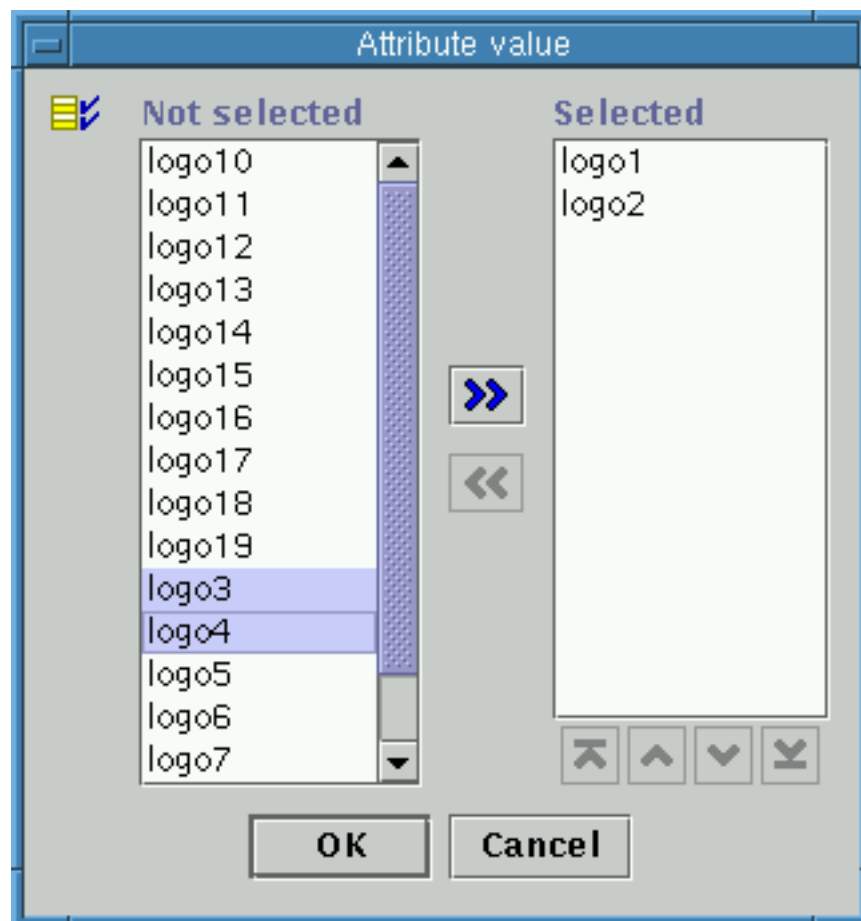
- The dialog box is non modal. You can keep it open all the time. Its content will reflect the attributes of the selected element (whether explicitly or implicitly selected).
- All possible attributes for an element are listed in alphabetical order with their default value (if any). Attributes which have not been set are displayed in gray. Attributes which have been set are displayed in black.
- Required attribute names are displayed using a bold font. Fixed attribute names are displayed using an italic font.

- The field on the right of the attribute name is editable: click on it and type the value of the attribute.  
If the attribute type is enumerated, this field is a menu rather than a text field, so you can directly choose the value of the attribute without having to type.
- Under the attribute name/value table, a small form can be used as an alternate way to edit the value of an attribute.

Practically, you'll never use this form except for the "minus" button which is used to remove an attribute.

This form is mainly useful

- for documents without a DTD where you need to create attributes (i.e. enter their names);
- when the attribute type is complicated (ex. ENTITIES) and you need the assistance of one of the specialized editors popped up when you click on the "list" button at the right of the attribute value text field.



*This dialog box may be used to specify a list of ENTITIES*

Type Ctrl-E to pop up the attribute editor. Select a **td** in the **table**. Set its **align** attribute to **center**, its **rowspan** attribute to 2, its **valign** attribute to **middle**. Add an extra **td** to make the **table** look more balanced.

Heading cell 1	Cell 2	Cell 5
Heading cell 3		Cell 4

It is important to remember that XXE automatically gives a dumb value to required attributes of newly created elements (otherwise the document would be invalid). This means that you have to replace this dumb value by the actual one as soon as the element has been created.

Add a **p** after the **table** and insert an **img** in it.



The **img** has two required attributes **src** and **alt**. XXE has set those attributes to string '?'.

Use the attribute editor to give these attributes an actual value.



(In this figure, we used `file:/usr/local/httpd/icons/apache_pb.gif` for **src** and `Powered by Apache` for **alt**.)

#### Notes about images:

- Using the CSS style sheets included in the XXE distribution, images are systematically displayed as 400x200 “thumbnails”.  
If you don't like this behavior, you'll have to change the style sheets by commenting out the corresponding **content** property.
- XXE can display images in the GIF, JPEG and PNG (JDK1.3) formats. This should not prevent you from specifying images in other formats: these images will be displayed using the standard image placeholder icon.

## 4.11 Checking document validity

You cannot check the validity of a document without an associated DTD. (When editing a document without a DTD, XXE guarantees that what you'll create will be well-formed without making any special effort.)

This section, like most of this tutorial, describes the behavior of XXE when editing documents with an associated DTD.

*Checking document validity is automatically performed each time you save your document.*

Unless the DTD is ill-formed, XXE will never allow editing commands that would make the document *structurally* invalid (a document where some elements have invalid sub-elements or attribute names).

Therefore explicitly checking document validity is rarely needed. You may have to use the Tools->CheckValidity command when:

1. You have loaded an invalid document and you are fixing it. After each editing command, you want to know if it is fixed now.
2. XXE creates elements where the *value* of the attribute is invalid.

This almost always occurs with an element which has an attribute of type IDREF (a reference to another element identified by a unique ID).

It is possible to create this type of element before creating any element identified by a unique ID.

In such case, XXE cannot automatically give to the IDREF attribute a valid value, and for this reason, your document is temporarily invalid.

After explicitly or implicitly checking document validity, its validity status is displayed at the bottom left of the window using the same icon but with three different colors.



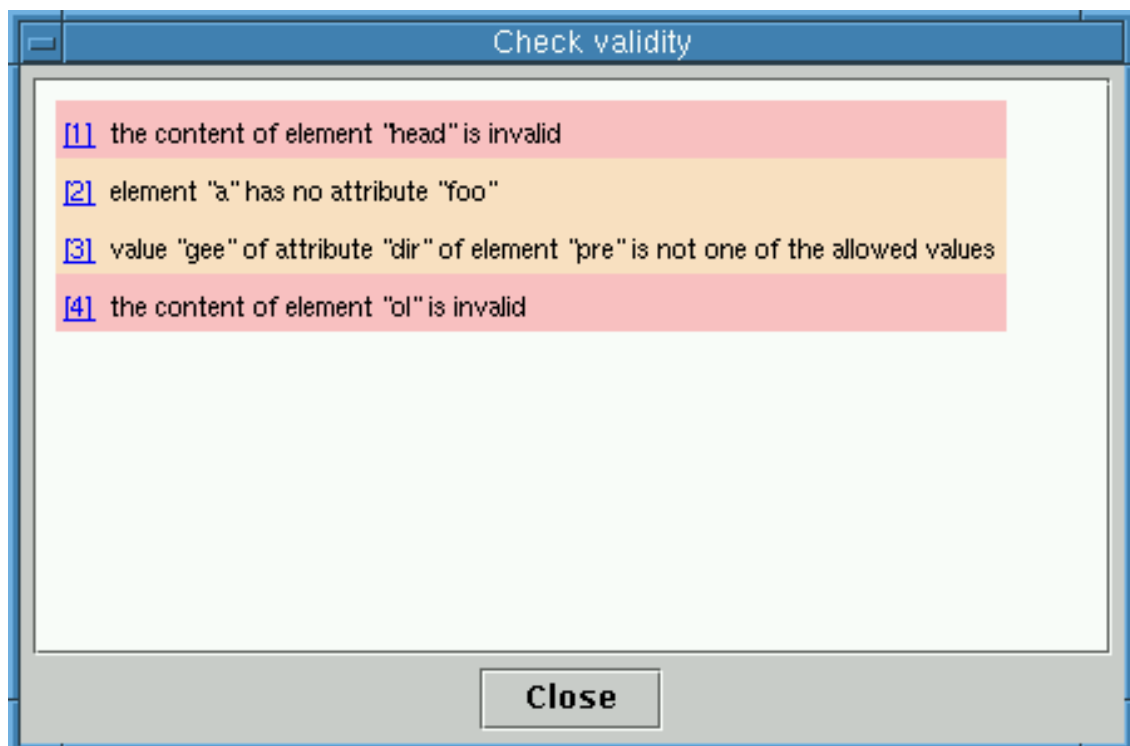
Icon color	Meaning
No icon at all	Document is valid
Yellow	Cross-reference problems (ex. an IDREF attribute contains a reference to a non-existent ID)
Orange	Invalid attribute names or values (other than cross-reference problems)
Red	Invalid element content

When the validity status changes from any color to red, XXE changes its editing mode from normal (strict) to repair (lenient). It will always signal this transition by popping up a dialog box.



In repair mode, in order to fix an invalid document, you are allowed to insert and delete elements more liberally than in the normal, strict mode.

Invoking the Tools->CheckValidity command will pop up a non-modal dialog box listing all validity errors, if such errors are detected.



Clicking on an error number will select the corresponding invalid element and will scroll the XXE window to make it visible.

**Tips:**

- Instead of using the Tools menu, you can click on the validity icon location in the status bar to invoke the Tools->CheckValidity command.

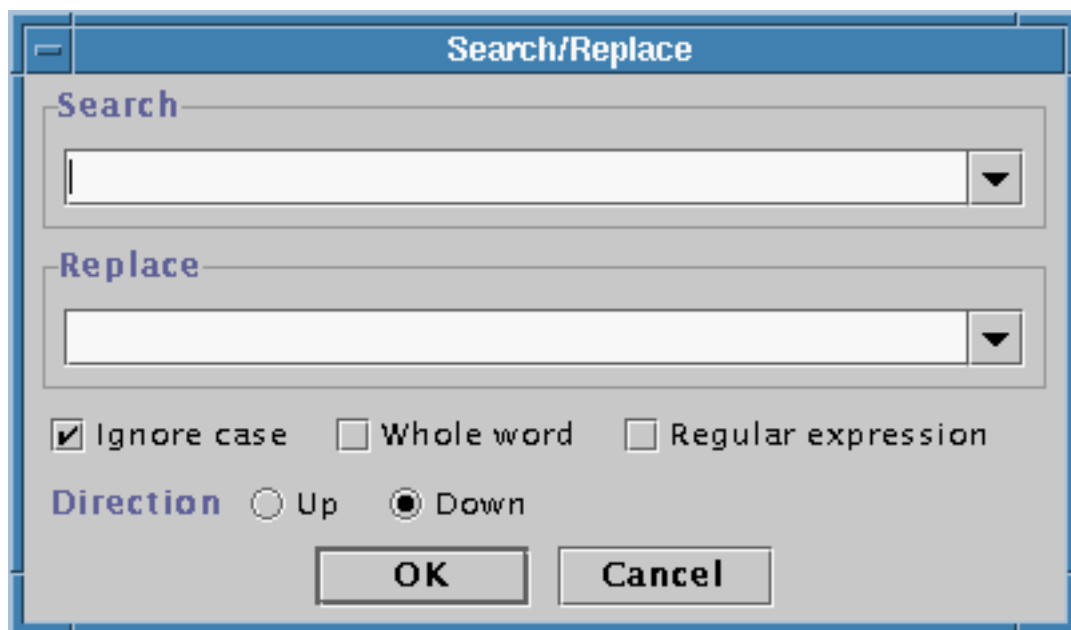
## 5 Command reference

### 5.1 File->Print

Displays the print dialog box in order to print the *explicitly* selected element if any, and the whole document otherwise.

Preferences related to printing may be specified in the Print tab of the Options dialog box.

### 5.2 Search->Replace



*The Search/Replace dialog box*

The document is searched from caret position to the end of the document.

If replacements are to be performed along with the search and an element has been *explicitly* selected, the search/replace starts at the beginning of this element, whatever is the position of the caret, and ends at the end of this element. (A backward search/replace starts at the end of the selected element and ends at its beginning.)

**Ignore case** The search is case-insensitive. Example: "foo" matches both "foo" and "Foo".

**Whole word** The found string must be a word, that is, the found string must be surrounded by white spaces. Example: "foo" matches "foo" but not "foobar".

**Regular expression** The searched string must be a valid regular expression.

In such case, \$1, \$2, ..., \$9 may be used in the replacement string to refer to the substrings matching the parenthesized groups of the regular expression.

\$0 is replaced by the string matching the regular expression in its entirety. \$\$ may be used to quote character '\$'.

Examples:

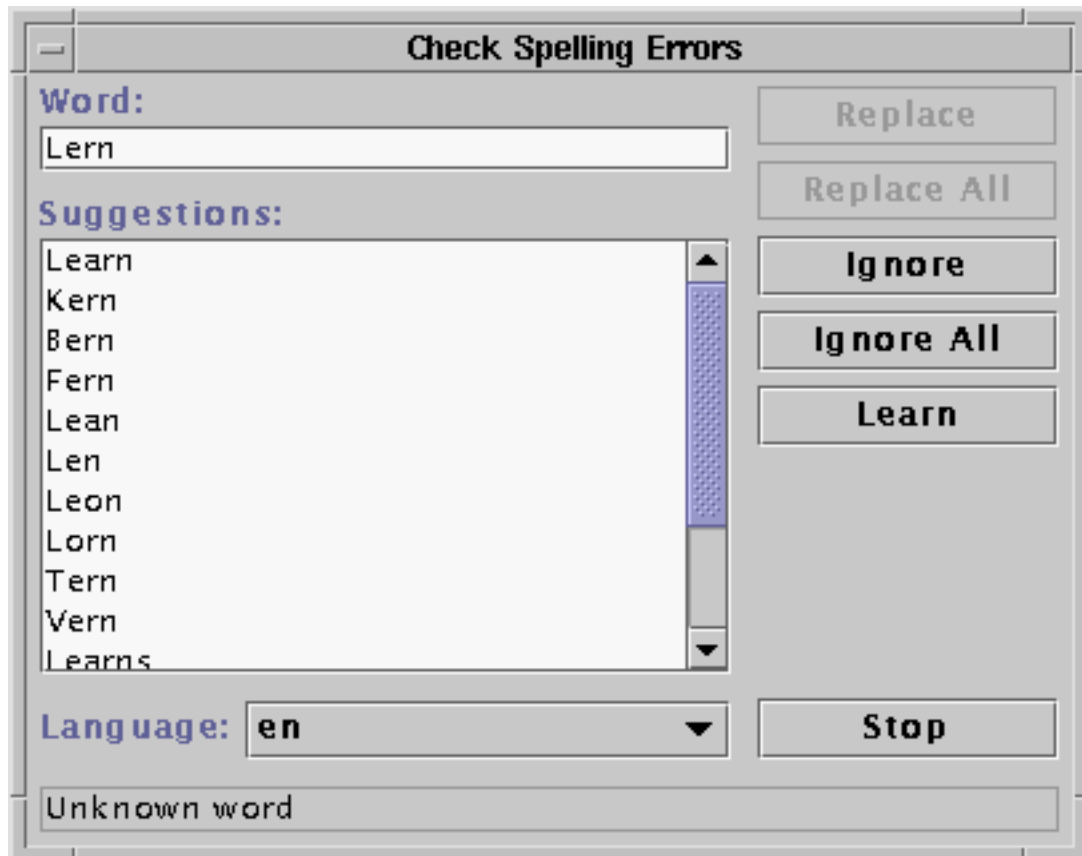
- "f(o+)" matching "foo", replaced by "g\$1", gives "goo".

- "f(o+)" matching "foo", replaced by "\$0bar", gives "foobar".
- "f(o+)" matching "foo", replaced by "g\$1", gives "g\$1".

Backward regular expression search/replace may be slow to the point of being unusable.

Preferences related to searching may be specified in the Search tab of the Options dialog box.

### 5.3 Tools->CheckSpellingErrors



*The Spell Checker dialog box*

Tools->CheckSpellingErrors (keyboard shortcut Ctrl-Shift-S) checks the text of the document from the word containing the caret to the end of the document.

If an element has been explicitly selected, the spell checker starts at the beginning of this element, whatever is the location of the caret, and stops at the end of the element.

Note that if the part of the document to be checked contains no spelling errors the dialog box shown above is not displayed at all. Instead message "Found no spelling errors" is displayed at the bottom of XME window.

**"Word:" text field** This text field displays the word for which the spell checker has found an error.

This text field can also be used to modify this word. After editing the word, typing on the Enter key replaces the original word by the modified one.

**"Suggestions:" list** This list contains replacement words, if any, suggested by the spell checker.

**"Language:" combo box** This combo box may be used to select the language of the dictionary used by the spell checker.

It is possible to switch from a language to another at any time. In such case, the spell checker is automatically restarted with the other dictionary, beginning at the word displayed in the text field.

The last selected language is recorded in the user preference file in order to be automatically chosen in subsequent XXE sessions.

Note that, in this milestone version, the spell checker ignores `xml:lang` and `lang` attributes and that it cannot be configured to automatically skip certain types of elements.

**"Replace" button** Replaces the original word by the content of the text field.

An empty text field may be used to *delete* the original word.

**"Replace All" button** Replaces all occurrences of the original word by the content of the text field.

This button is disabled for errors different from "Unknown word" or "Improperly capitalized word".

**"Ignore" button** Skips the word for which the spell checker has found an error.

**"Ignore All" button** Skips all occurrences of the word for which the spell checker has found an error.

This button is disabled for errors different from "Unknown word" or "Improperly capitalized word".

**"Learn" button** Records the word in the personal dictionary for currently selected language.

This button is disabled for errors different from "Unknown word".

Personal dictionaries are text files named `dict_LL.txt`, where *LL* is an ISO code for a language. These files are created in directory

`~/xxespell`

under Unix and in directory

`C:\winnt\Profiles\<user>\xxespell`

under Windows.

Personal dictionaries are automatically saved when XXE is exited.



## 5.4 Options->Options



*The Options dialog box*

**Guess what spaces** If this toggle is checked, when loading an XML file without a DTD associated to it, whitespace characters are trimmed from elements containing child elements separated by whitespace.

Default: no.

This heuristic is often a good one for XML-data.

**Don't use the style sheet** Check this toggle if you always prefer to edit XML using the tree view rather than the CSS styled view.

Default: no.

Load Save Print Edit Search Style

Encoding: ☒ UTF-8 ☐ ISO-8859-1

Indent ☐ when DTD ☐ when no DTD

Indentation (0-10):

Max. line length (> 0):

**Encoding** UTF-8 is the default encoding of the XML files created by XXE because this encoding is compact and may be used to represent all Unicode characters.

However, if you need to deliver an XML file created by XXE to a person using a text editor to view or modify it, it is nicer to use a human-readable encoding such as ISO-8859-1 (Western character set). In such case, check the ISO-8859-1 toggle.

Default: UTF-8.

Notes:

- If you only use ASCII characters, choosing either encoding makes no difference.
- If you use characters outside the ISO-8859-1 character set, check the UTF-8 toggle or you'll have '?' replacing all non ISO-8859-1 characters in the XML file created by XXE.
- Human-readable encodings other than ISO-8859-1 (Cyrillic=ISO-8859-5, Greek=ISO-8859-7, etc) are not listed in this dialog box because we cannot read back such files in XXE.

**Indent** XML files created by XXE are by default not indented because it is faster and safer (no risk to add superfluous whitespace) to do so.

However, if you need to deliver an XML file created by XXE to a person using a text editor to view or modify it, it is nicer to provide this person with an indented XML file:

**when DTD** Check this toggle to create indented XML files for documents that have a DTD associated with them. It is always safe to do so.

Default: no.

**when no DTD** Check this toggle to create indented XML files for documents without a DTD associated with them. It is not always safe to do so because this option may add superfluous whitespace.

Default: no.

In indented files, elements with an empty content model are saved as (XHTML example) `<br />` rather than `<br />` and empty elements with a non-empty content model are saved as `<p></p>` rather than `<p />`.

This has been recommended by the W3C to help Web browsers (HTML user agents as they say) to correctly display XHTML. For DTDs other than XHTML, this behavior causes no harm. It even improves the readability of the indented XML files created by XXE.

**Indentation** Specifies the number of space characters used to indent a child element relatively to its parent element.

Default: 2.

**Max. line length** Specifies the maximum line length for elements containing text interspersed with child elements.

Default: 78.

This number is only used as a hint: XML files created by XXE may contain lines much longer than this number.

Load Save Print Edit Search Style

Screen resolution (50-150dpi): 100.0

**Header**

Begin:

Middle:

End:

Color:  ☒ Underline

**Footer**

Begin:

Middle:

End:

Color:  ☒ Overline

*Substituted variables: %F=file, %P=page, %I=page number, %C=page count*

**Screen resolution** Specifies the screen resolution in DPI (Dot Per Inch) used when printing. This resolution directly determines the amount of text a printed page can contain.

Default: 100dpi.

**Begin** The page header has 3 areas: begin (the left for left-to-right languages), middle, end (right). This field specifies the text printed at the left of the page header.

Default: empty.

Each area can contain a mix of text and variables:

Variable	Description
%F	File name of the document being edited
%P	Equivalent to "page %I of %C"; see below
%I	Current page number
%C	Total page count

**Middle** Specifies the text printed at the center of the page header.

Default: empty.

**End** Specifies the text printed at the right of the page header.

Default: empty.

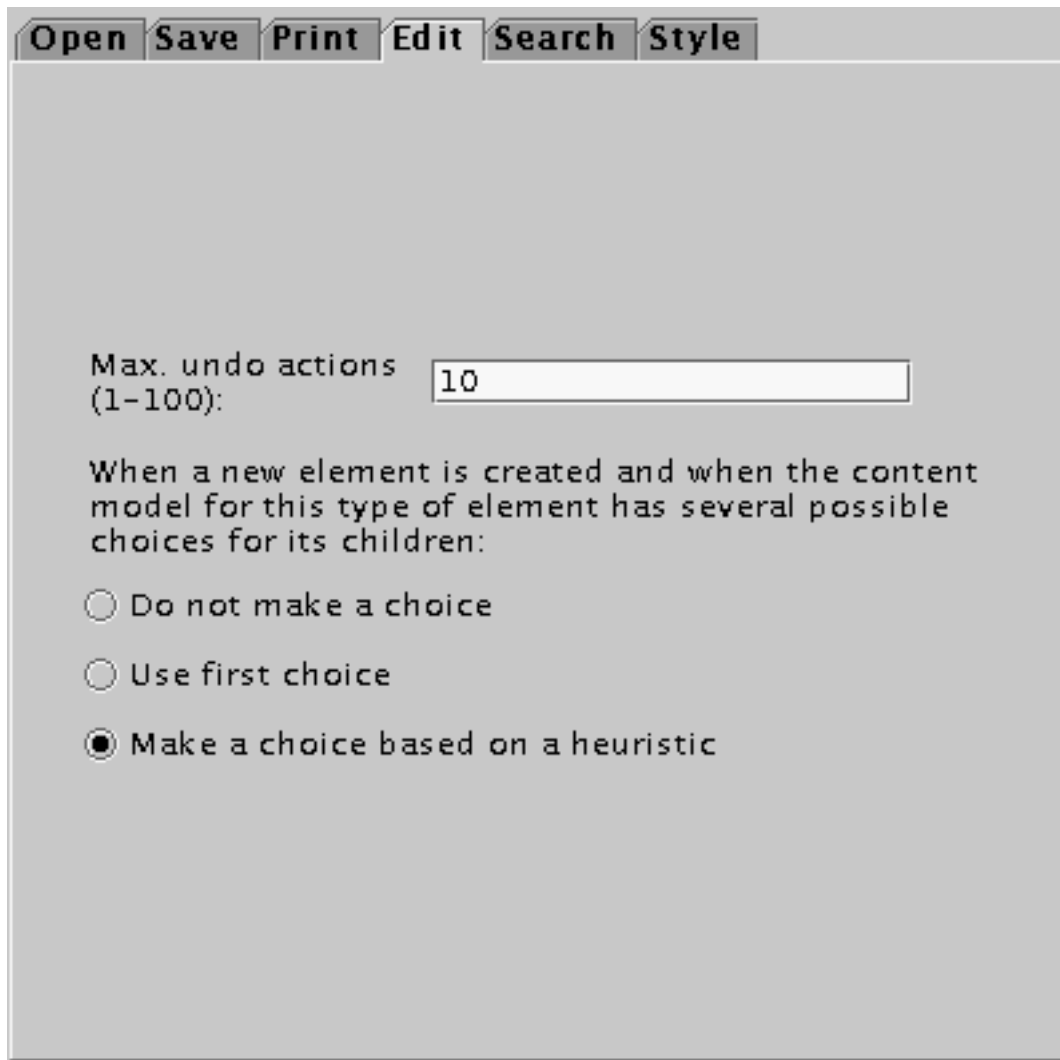
**Color** Specifies the color of the text of the page header.

Default: gray.

Note that the font used for the page header is the default font of the style sheet (see the Style tab below).

**Underline [Overline]** Specifies if a thin line is to be printed below the page header [above the page footer].

Default: yes.



**Max. undo actions** Specifies the maximum number of undo (redo) actions a user will be able to perform. Limited to 100 because a single undo action may consume a great deal of memory.

Default: 10.

**When a new element is created...** XXE in its milestone incarnation has no way to specify a template for a newly created element (example: a table that has a head, two rows and two columns). Instead, XXE recursively creates the new element and its children according to the minimum valid content specified by the DTD.

This option specifies how a new element is recursively created when the content type for this type of element has several possible choices for the child elements.

DocBook example: the new element to be inserted is a figure.

```
<!ELEMENT figure ((title,titleabbrev?), (literallayout|programlisting|
programlistingco|screen|screenco|screenshot|synopsis|cmdsynopsis|
funcsynopsis|classsynopsis|fieldsynopsis|constructorsynopsis|
destructorsynopsis|methodsynopsis|address|blockquote|graphic|
graphicco|mediaobject|mediaobjectco|informalequation|informalexample|
informalfigure|informaltable|indexterm|beginpage|link|olink|ulink)+)>
```

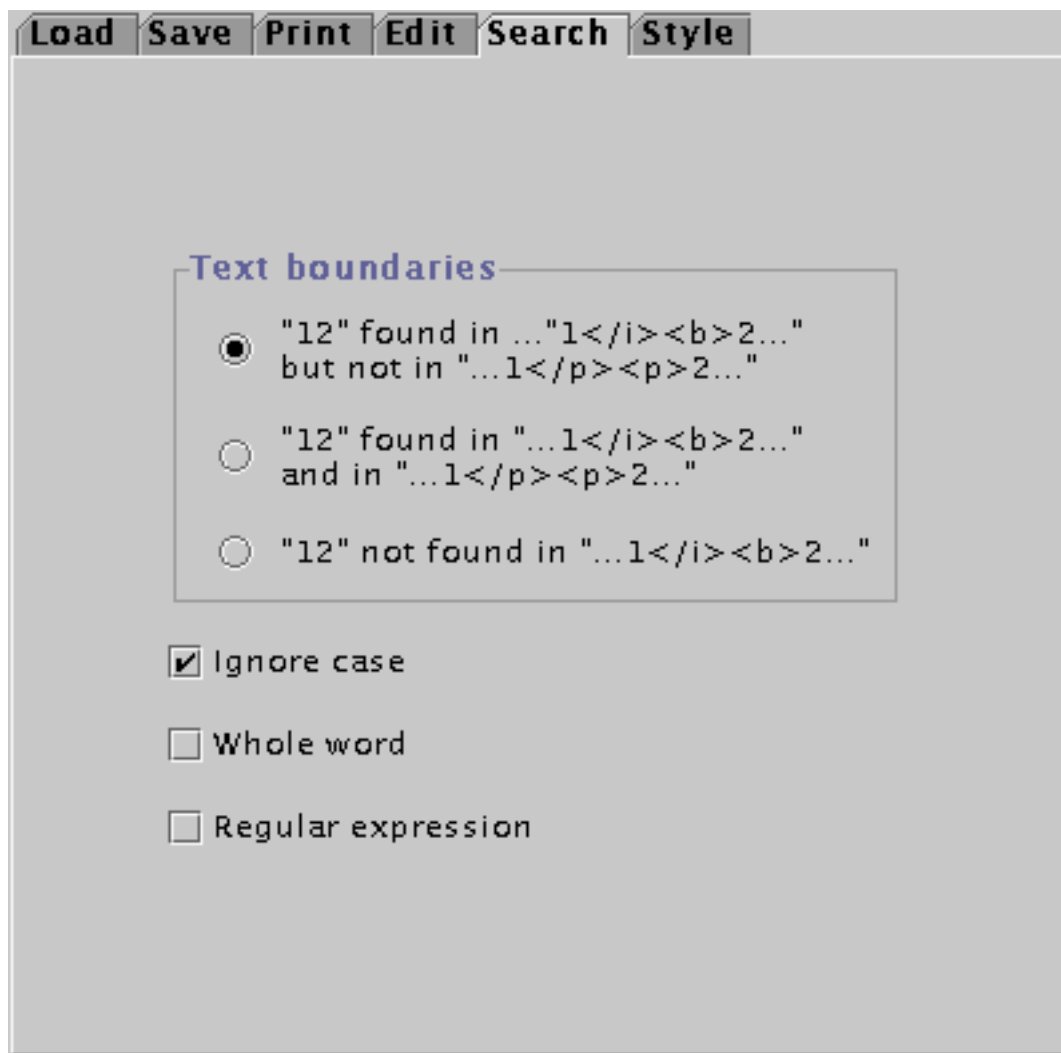
**Do not make a choice** Inserts a figure with a title but no body.

With this option, the newly created element may be invalid. Therefore, use this option if you really know what you are doing.

**Use first choice** Inserts a figure with a title and a literallayout.

**Make choice based a heuristic** Inserts a figure with a title and a link.

The, very questionable, heuristic selects the text container with the shortest name. This is the default value for this option.



**Text boundaries** Searching string "Hello world!" in an XML document is not as obvious as it seems: for example, is "Hello world!" with word "Hello" contained in an emphasis element followed by #PCDATA " world!" supposed to be found by XXE?

This behavior can be specified by choosing one of the 3 following modes (each mode corresponds to a radio button in the Search tab):

1. Smart mode: "Hello world!" is found within "<em>Hello</em> world!" but not within "<p>Hello </p><p>world!</p>".

This mode uses the DTD to recognize logically contiguous text across different types of elements.

A text search behaves as if 2 linefeed characters have been inserted after each contiguous piece of text. It is therefore possible using regular expression "\n\n" to find and visualize these text boundaries in the document.

2. Dumb mode A: there are no boundaries around a #PCDATA whatever the type of the element containing it.
3. Dumb mode B: a #PCDATA is separated from other #PCDATA whatever the type of the element containing it.

Default: mode 1.

**Ignore case** Default value for check box **Ignore case** of the Search/Replace dialog box (see Search->Replace).

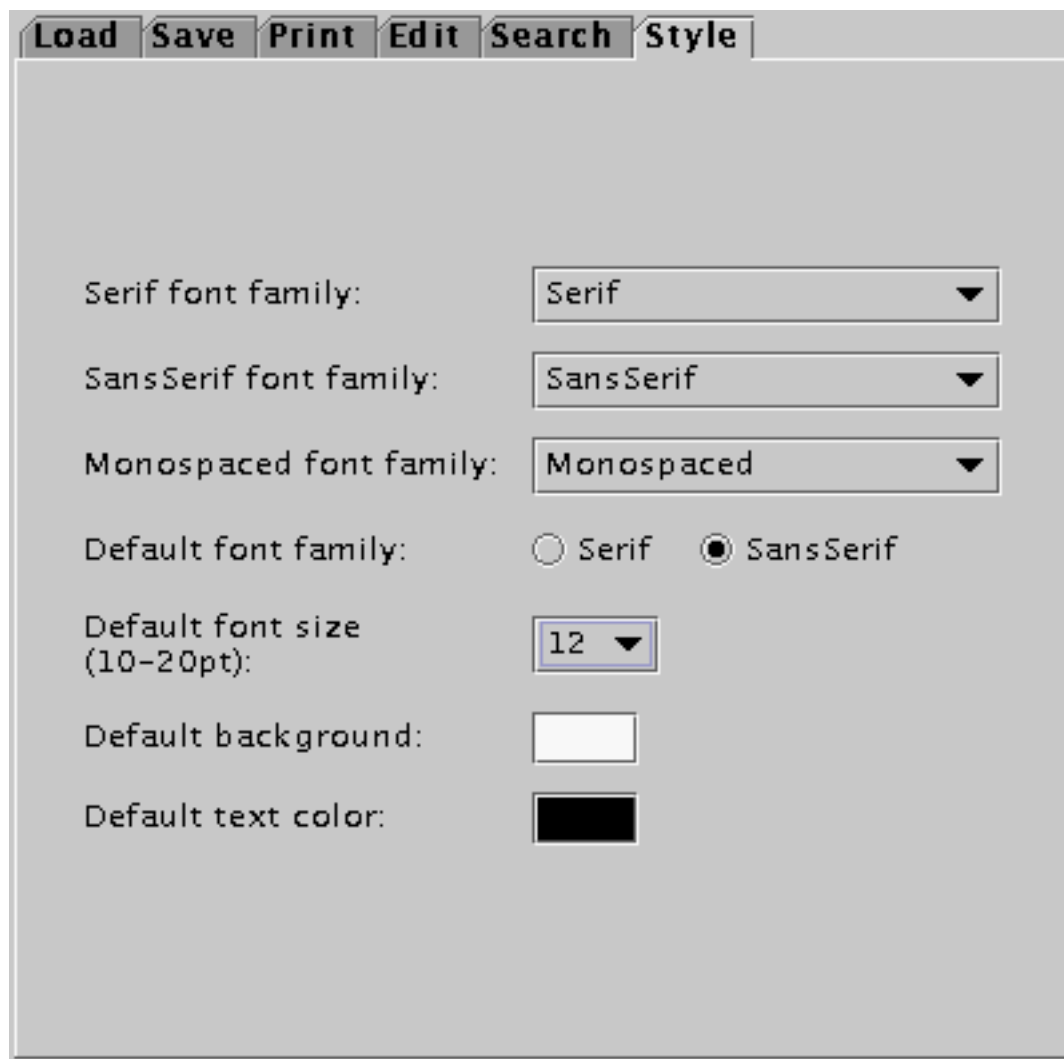
Default: yes.

**Whole word** Default value for check box **Whole word** of the Search/Replace dialog box (see Search->Replace).

Default: no.

**Regular expression** Default value for check box **Regular expression** of the Search/Replace dialog box (see Search->Replace).

Default: no.



These preferences parameterize the CSS style sheet used to visualize the document or to print it.

Setting some of these preferences will have no visible effect if the style sheet author has specified the corresponding properties in the style sheet. For example, if the user preferred background is specified in the Style pane as being light yellow and if the style sheet author has specified the root element background-color as being white, the document will be rendered with a white background.

**Serif font family** Specifies the font family used for property value font-family:serif.

Default: Serif (the Java default serif font family).

**SansSerif font family** Specifies the font family used for property value font-family:sans-serif.

Default: SansSerif (the Java default sans-serif font family).

**Monospaced font family** Specifies the font family used for property value font-family:monospace.

Default: Monospaced (the Java default monospaced font family).

**Default font family** Specifies the default value for property font-family.

Default: SansSerif.

**Default font size** Specifies the default value for property font-size.

Default: 12pt.



**Default background** Specifies the default value for property background-color.

Default: white.

**Default text color** Specifies the default value for property color.

Default: black.

## 5.5 Options->Non-XMLFormats

### 5.5.1 Using XXE to edit non-XML structured formats

XXE can be used to edit non-XML *structured* formats like HTML, Javadoc(TM) comments, APT, `/etc/passwd`, `/etc/printcap`, `http.conf` (the Apache configuration file), etc, if plug-ins are provided for such structured formats.

A format plug-in works by converting the data on the fly from the non-XML format to XML when loading a document and by converting them back to the non-XML format each time the document is saved.

XXE relies on the file name extension to find out which plug-in is needed to load a document. Documents whose file name extension has not been associated with any format plug-in are loaded as XML (unless downloaded from a http server which returns the MIME type of the document).

XXE is bundled with two format plug-ins: one for Javadoc and one for APT. See below for a description of these plug-ins.

### 5.5.2 Registering a format plug-in with XXE

Before being able to load a non-XML document into XXE, the user needs to register the corresponding plug-in with the application.

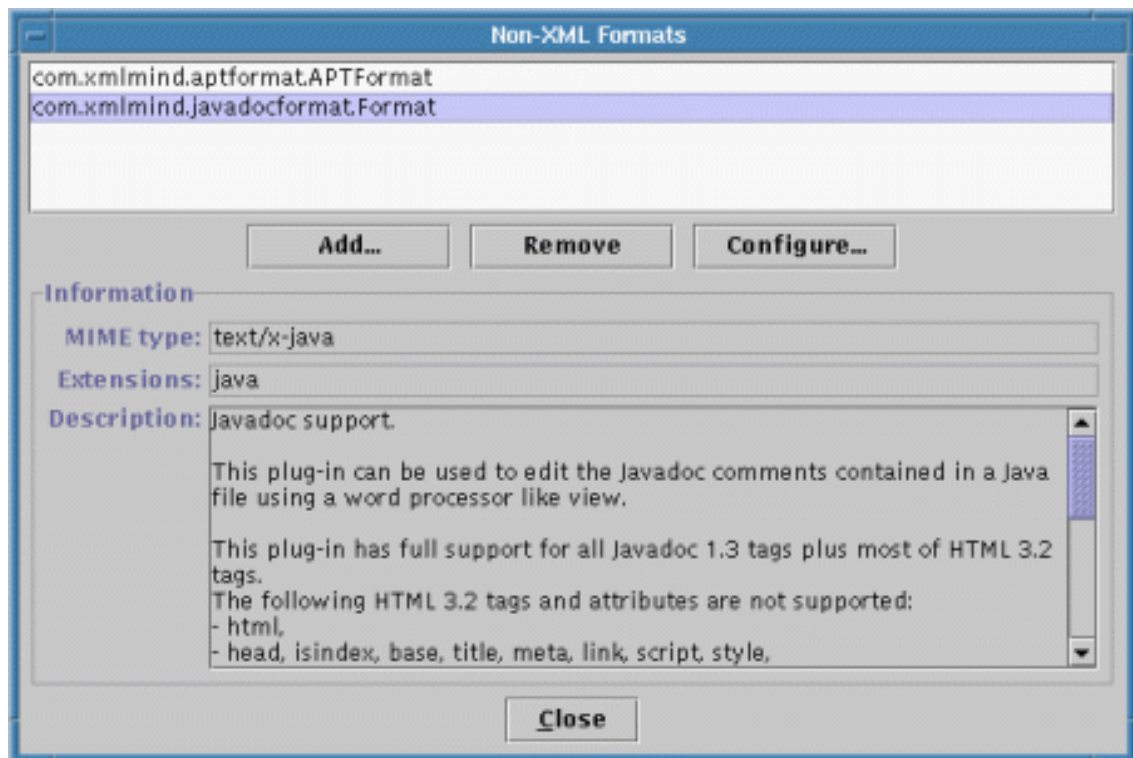
The format plug-ins bundled with XXE are automatically registered the first time the user starts XXE. You may need to manually register these plug-ins before being able to use them if you have an existing XXE user preference file (for example if you have upgraded your XXE distribution). The XXE user preference file is stored in file `~/.xxe` under Unix and in file `C:\winnt\Profiles\<user>\xxe.ini` under Windows NT.

#### Procedure:

1. Exit from XXE.
2. A format plug-in is contained in one or several `.jar` files. Copy these `.jar` files in a place where the Java runtime can find and load them (i.e. "put them in the Java CLASSPATH").
3. Restart XXE.

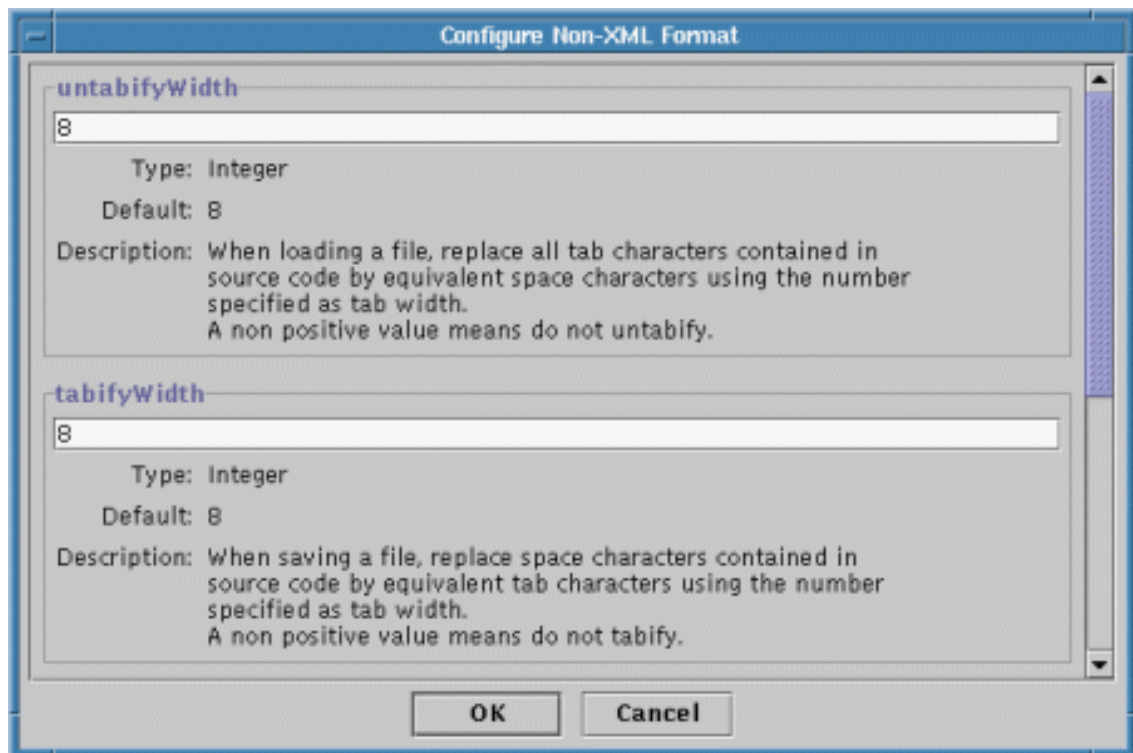
*These first 3 steps are not needed for the bundled plug-ins.*

4. Use menu command Options->Non-XMLFormats. A dialog box is displayed:



*The dialog box used to register a format plug-in with XXE*

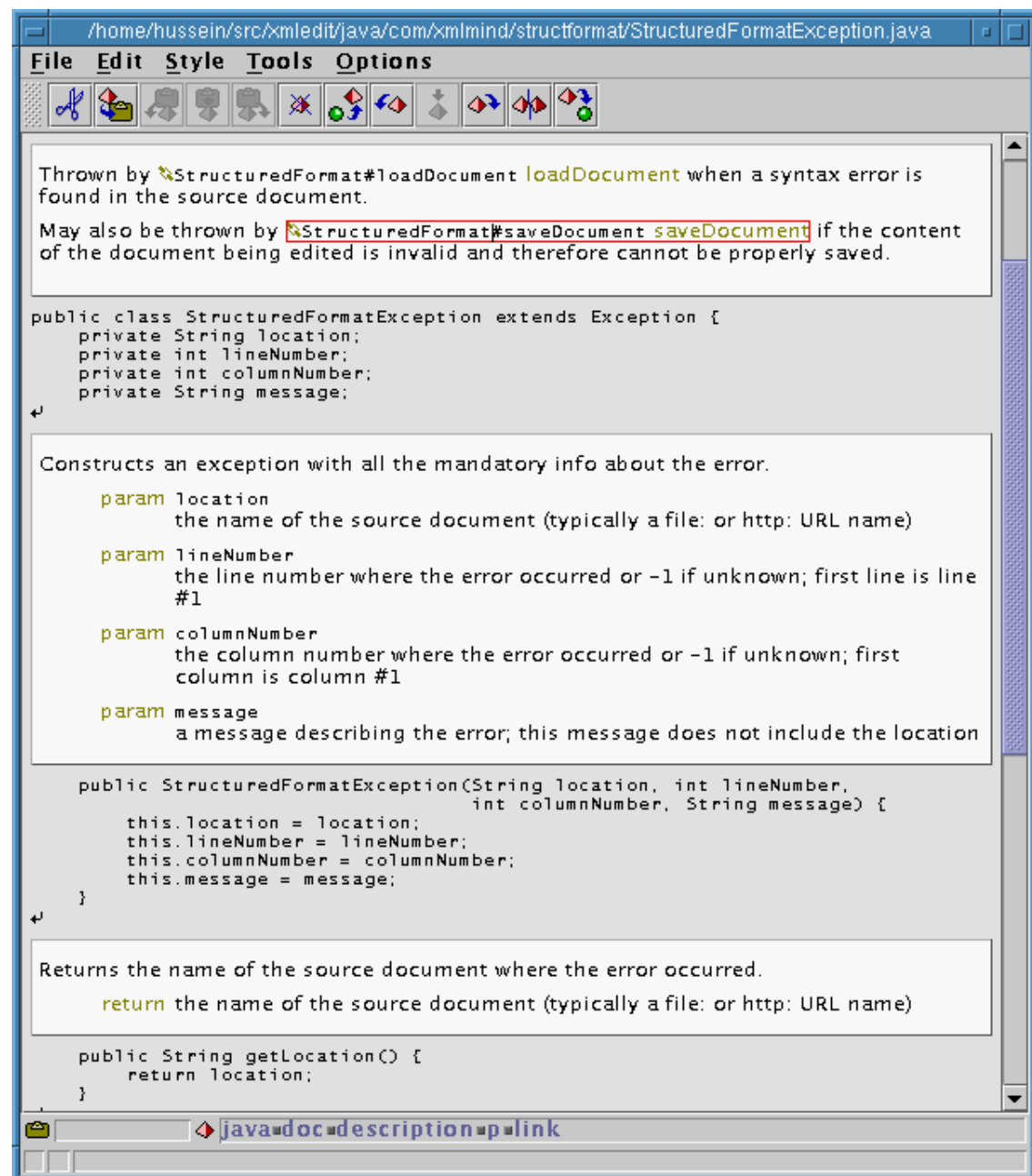
5. Click on the **Add** button. An input dialog box is displayed requesting you to enter the name of the format plug-in.
6. Enter the format plug-in name exactly as specified in its documentation.  
Example: `com.xmlmind.javadocformat.Format` for the Javadoc plug-in.
7. Click in the list on the format plug-in name to display information about it. If the plug-in has parameters, the **Configure** button is enabled, otherwise this button is “grayed”.
8. If possible, click on the **Configure** button to set the parameters of the plug-in. A dialog box is displayed:



*The dialog box used to configure a format plug-in*

Note that the plug-in parameters are documented in this dialog box.

### 5.5.3 The Javadoc plug-in



*A Java file as displayed by XXE*

This plug-in adds Javadoc(TM) support to XXE.

It must be registered as `com.xmlmind.javadocformat.Format`.


























The intended audience for this plug-in is Java programmers and Javadoc writers. With XXE and this plug-in, it becomes possible to edit the Javadoc comments contained in a Java file using a word processor-like view. Writing Javadoc this way is less tedious (no manual formatting of comment lines) and is no longer error-prone (DTD-directed editing).

This plug-in has full support for all Javadoc 1.3 tags plus most of HTML 3.2 tags.

The following HTML 3.2 tags and attributes are *not* supported:

- html,
- head, isindex, base, title, meta, link, script, style,
- body,
- object, applet, param,
- menu, dir,
- map, area, img usemap and ismap attributes,
- form, input, select, option, textarea,
- xmp, listing, plaintext.

The Javadoc 1.3 tags are modeled in XML as follows:

java	
 #PCDATA	Java source code
 formfeed	Ctrl-L inserted in source code
 doc	Javadoc comment block <code>/** */</code>
 <b>description</b> any HTML inline or block +link	The description before the Javadoc tags
 <b>link</b> #PCDATA +attribute <b>label</b>	{@link package.class#member ?label?}
 <b>author</b> any HTML inline or block +link	@author text
 <b>version</b> any HTML inline or block +link	@version text
 <b>param</b>	@param name text
 <b>paramname</b> #PCDATA	
 <b>paramdesc</b> any HTML inline or block +link	
 <b>return</b> any HTML inline or block +link	@return text
 <b>exception</b>	@exception or @throws name text
 <b>exceptionname</b> #PCDATA	
 <b>exceptiondesc</b> any HTML inline or block +link	
 <b>see</b> #PCDATA +attribute <b>label</b>	@see package.class#member ?label?
 <b>seeref</b> #PCDATA	@see "string"
 <b>seehref</b> same content and attributes as HTML <a>	@see <a href="URL#value">label</a>
 <b>since</b> any HTML inline or block +link	@since text
 <b>serial</b> any HTML inline or block +link	@serial text
 <b>serialdata</b> any HTML inline or block +link	@serialData text
 <b>serialfield</b>	@serialField name type text
 <b>fieldname</b> #PCDATA	
 <b>fieldtype</b> #PCDATA	
 <b>fielddesc</b> any HTML inline or block +link	
 <b>deprecated</b> any HTML inline or block +link	@deprecated text

The file name extension required for a “Javadoc document” is `.java`.

Unlike Web browsers, this plug-in is not designed to load broken HTML 3.2. However, this plug-in can help Javadoc writers to easily spot and fix the HTML errors contained in Javadoc comments. See next section for a real world case study.



*the Javadoc plug-in refuses to load Sun's String.java for the above reason*

When XXE refuses to load a Java file, an error dialog is displayed with

- an error message (not always easy to understand due to the layered architecture),
- the line number of the *start* of the Javadoc comment block where the error occurred,
- a column number always equal to 1.

Try to guess what the error message means and fix the problem using your favorite text editor (use [xmlmind-support+xmlmind.com](mailto:xmlmind-support+xmlmind.com) if you are really stuck), then reload the Java file into XXE.

**Warning:** before using this plug-in for the first time, please take the time to configure it properly to make sure that its newline and tab policies are compatible with yours.

**Applying the Javadoc plug-in to Sun JDK 1.3.1 sources** The Javadoc plug-in has been tested on all the Java sources given by Sun for the Linux JDK 1.3.1.

The plug-in has succeeded to load 1750 out of 1877 Java files. It has failed 127 times generally for the following reasons:

- The Javadoc writer adds `<code></code>` tags at places where plain text (`#PCDATA`) is expected.  
Example: putting the name of a `@param` between `<code></code>`.  
Note that it is not useful to do so because Javadoc automatically adds a sensible style to this kind of plain text.  
This is the most important discrepancy between the Javadoc plug-in and Javadoc: Javadoc intelligently discards `<code></code>`, while the Javadoc plug-in stubbornly refuses to load the Java file.
- Typos such as:
  - typos in tag names (examples: `blockquote`, `coder`)
  - forgetting `'>'` at the end of start or end tags

- unknown character entities (example: `&le;`)
  - putting a space between the attribute name, the '=' sign and the attribute value
  - using end tags (example: `</p>`) instead of start tags
  - duplicating attributes (example: `align`)
- The Javadoc writer sometimes forgets what he has learned about HTML.  
Example 1: putting plain text or inline elements such as `<code></code>` directly into a blockquote.  
Example 2: putting a `pre` block between `<code></code>`.
  - '>', '<', '&' not properly escaped. Here, the plug-in tries to do its best but sometimes it misses some of these special characters.

Note that all the errors described above are impossible to do if XXE is used to edit the Javadoc comments.

Having succeeded to load a Java file into XXE does not mean that this file is valid: 45 out the 1750 loaded files were found having validity problems.

Generally these validity problems are minor and are caused by missing attributes or invalid attribute values. It is possible to quickly fix them by using menu command Tools->CheckValidity.

#### 5.5.4 The APT plug-in

This plug-in adds APT (Almost Plain Text) format support to XXE.

It must be registered as `com.xmlmind.aptf format.APTFormat`.

APT is a text format for technical articles which is simple, structured, tag-less, and which is convertible to HTML, XHTML, PDF, RTF, PostScript, SGML or XML DocBook using the Open Source tool called aptconvert. Aptconvert can be downloaded from <http://www.pixware.fr/>.

The file name extension required for an APT document is `.apt` or `.txt`.



## **6 APPENDIX A: CSS2 properties and values supported by XXE**

### **6.1 Restrictions**

The following properties can be inherited whether explicitly (**inherit** keyword) or implicitly (inherited property).

For all properties except line-height where the specified number is inherited (which is the correct behavior), the inherited value is the actual value not the computed value.

Property	Value	Restrictions
background-color	<i>color</i>  transparent  inherit	-
border	<i>width</i> [ <i>style color</i> ?]?  inherit	Order is strictly width, style, color
border-color	<i>side_value</i> { 1,4 }	-
border-bottom-color	<i>color</i>  transparent inherit	-
border-left-color	"	-
border-right-color	"	-
border-top-color	"	-
border-style	none dotted dashed  solid double groove  ridge inset outset	No hidden
border-width	thin thick medium   <i>length</i>  inherit	-
color	<i>color</i>  inherit	-
font	[ <i>style weight</i> ?]? <i>size family</i>  inherit	Order is strictly style then weight
font-family	[[ <i>name generic</i> ] ,]* <i>name generic</i> > <i>name generic</i>  inherit	Anything other than serif sans-serif, monospace is ignored
font-size	medium small large  x-small x-large  xx-small xx-large  smaller larger   <i>length</i>   <i>percentage</i>  inherit	-
font-style	normal italic oblique  inherit	italic and oblique are aliases
font-weight	normal bold inherit	No <i>N00</i> , bolder, lighter
line-height	normal  <i>number</i>  inherit	No <i>length, percentage</i>
margin	<i>side_value</i> { 1,4 }	-
margin-bottom	<i>length</i>  auto inherit	No <i>percentage</i>
margin-left	"	-
margin-right	"	-
margin-top	"	-
padding	<i>side_value</i> { 1,4 }	-
padding-bottom	<i>length</i>  inherit	No <i>percentage</i>
padding-left	"	-
padding-right	"	-
padding-top	"	-
text-align	left right center  inherit	No justify
text-decoration	none underline overline  line-through inherit	No blink
text-indent	<i>length</i>  inherit	No <i>percentage</i>
vertical-align	baseline middle sub  super text-top top  text-bottom bottom  inherit	No <i>length, percentage</i>
white-space	normal pre nowrap  inherit	-

The following properties cannot be inherited whether explicitly (**inherit** keyword) or implicitly (inherited property).

Property	Value	Restrictions
border-spacing	<i>length length?</i>	-
content	<i>string uri attr(X)</i>  open-quote close-quote  no-open-quote  no-close-quote  counter( <i>name</i> )  counter( <i>name</i> , <i>style</i> )  counters( <i>name</i> , <i>separ</i> )  counters( <i>name</i> , <i>separ</i> , <i>style</i> )  disc circle square  see extensions	Open-quote is char. "'", close-quote is char. '"', no-*-quote is ignored See restrictions about counters Counter styles are limited to: decimal, lower-alpha, lower-latin, upper-alpha, upper-latin, lower-roman, upper-roman
display	none inline block  marker table  inline-table  table-row-group  table-header-group  table-footer-group  table-row  table-column-group  table-column table-cell  table-caption  inline-block tree	No list-item (use markers), run-in, compact
height	<i>length auto</i>	No <i>percentage</i>
marker-offset	<i>length auto</i>	No <i>percentage</i>
width	<i>length auto</i>	No <i>percentage</i>

- The CSS box decorations (**border**, **padding**, etc) are not supported for inlined elements. The **background-color** is the only property supported for such elements.
- The border properties, except **border-color**, cannot be specified individually for each side of the box.
- **counter-reset**, **counter-increment** properties are ignored. **counter** and **counters** are supported inside the **content** property but with different semantics: the name of the counter is used to specify the non-formatted value of the counter.

Example 1 (XHTML):

```
ol > li:before {
  display: marker;
  content: counter(n, decimal);
  font-weight: bold;
}
```

In the previous example, the counter name is *n* (single letter 'n', any letter is OK) which specifies that the counter value is the rank of *li* within its parent element (an *ol*).

Example 2 (DocBook):

```
sect3 > title:before {
  content: counter(nnn-) " ";
}
```

In the previous example, the counter name is `nnn-` (3 letters followed by a dash) which specifies that the counter segmented value must be built as follows:

1. Skip (dash means skip) the rank of `title` within its parent element (a `sect3`).
  2. Prepend (any letter means use) the rank of `title` parent (a `sect3`) within its parent (a `sect2`).
  3. Prepend the rank of `title` grand-parent (a `sect2`) within its parent (a `sect1`).
  4. Prepend the rank of `title` grand-grand-parent (a `sect1`) within its parent (an `article` or a `chapter`).
- The style of an element may be specified using a selector containing the **:first-child** pseudo-class but the style of such element is not recomputed when sibling elements are inserted or deleted.  
The workaround is to manually “refresh” the view of the parent of such element using the Ctrl-L keyboard shortcut.
  - The **!important** specifier is ignored.

## 6.2 Extensions

- Two properties **column-span** and **row-span** have been added to specify the column and row span of elements with a **table-cell display**. The value for these properties is a strictly positive integer number. The initial value is 1. These properties are not inherited.
- **eval(value, ..., value)** may be used to specify a dynamically evaluated property value anywhere a static property value is allowed.

A dynamic property value is evaluated just before building the view corresponding to the subject of the selector:

1. The *value* arguments are converted to strings and concatenated together.
2. The resulting string expression is passed to a script language interpreter which evaluates it in the context of the subject of the selector.
3. The result of the evaluation is a string which is parsed as a property value.

XXE M1.3 has no built-in script language interpreter yet but even without, say a JavaScript interpreter, this feature is useful because it allows to use **attr()** property values outside **content** properties.

Example, XHTML `th` and `td` **table-cell** formatting heavily relies on this feature:

```
td, th {
  display: table-cell;
  text-align: eval(attr(align));
  vertical-align: eval(attr(valign));
  row-span: eval(attr(rowspan));
  column-span: eval(attr(colspan));
  border: 1 inset gray;
  padding: 2;
}
```

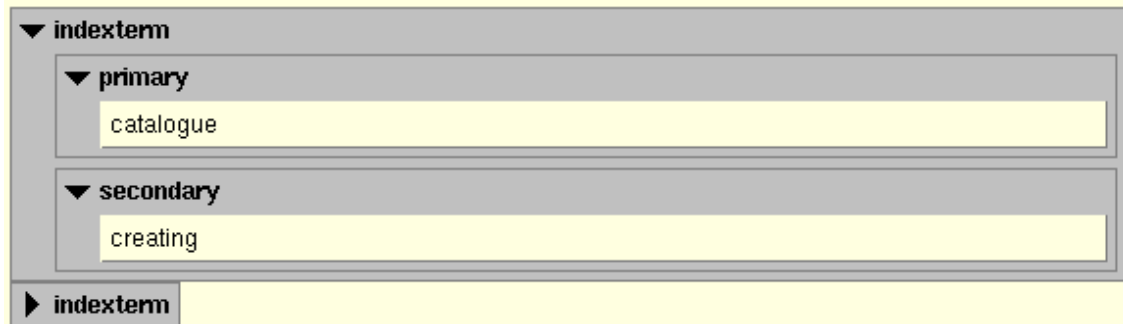
When a script language interpreter will be integrated in XXE (expected in M2), this feature will allow the editor to fully support DocBook and XHTML tables by extracting cell property values from table meta-information tags such as `colspec`, `spanspec`, `colgroup`, `column`, etc.

- **display: inline-block** may be used to specify a rectangular block that participates in an inline formatting context. (This is similar to **inline-table**.)

- **display: tree** may be used to mix styled element views and non-styled (tree-like) element views. This is particularly useful for meta-information (such as DocBook's bookinfo, sectioninfo, indexterm, etc) for which a sensible style is hard to come up with.

Example (DocBook):

### 3 Creating and modifying catalogues



A catalogue is a text file containing the translation rules of the public identifier to system's files.

- In addition to what is specified by CSS2, the **:before** and **:after** pseudo-elements allow values of the **display** property as follows:
  - If the subject of the selector is a **table** element, allowed values are **none**, **block**, **table-row-group** and **table-row**. If the value of **display** has any other value, the pseudo-element will behave as if the value was **block**.
  - If the subject of the selector is a **table-row-group** element, allowed values are **none** and **table-row**. If the value of **display** has any other value, the pseudo-element will behave as if the value was **table-row**.
  - If the subject of the selector is a **table-row** element, allowed values are **none** and **table-cell**. If the value of **display** has any other value, the pseudo-element will behave as if the value was **table-cell**.

These extensions are supported to add generated column and row headers to arbitrary XML data displayed as a table.

For example, with these styles, the select, optgroup and option XHTML elements are displayed as a table with automatically generated column and row headers:

```

select {
  display: table;
  border: 1 solid black;
  padding: 2;
  border-spacing: 2;
  background-color: silver;
}
select:before {
  content: row(cell("Category", width, 20ex), cell("Choice #1"),
               cell("Choice #2"), cell("Choice #3"),
               font-weight, bold, color, olive,
               padding-top, 2, padding-right, 2,
               padding-bottom, 2, padding-left, 2,
               border-width, 1, border-style, solid);
  display: table-row-group;
}
optgroup {
  display: table-row;
}
optgroup:before {
  content: attr(label);
}
option {
  display: table-cell;
  border: 1 solid black;
  padding: 2;
  background-color: white;
}

```

XHTML source:

```

<p><select>
<optgroup label="Language">
<option>Java</option>
<option>C++</option>
<option>Perl</option>
</optgroup>
<optgroup label="Editor">
<option>Emacs</option>
<option>Vi</option>
<option>UltraEdit</option>
</optgroup>
<optgroup label="OS">
<option>Linux</option>
<option>Windows</option>
<option>Solaris</option>
</optgroup>
</select></p>

```

Rendered as:

Category	Choice #1	Choice #2	Choice #3
Language	Java	C++	Perl
Editor	Emacs	Vi	UltraEdit
OS	Linux	Windows	Solaris

- **content** may be used for any element and not only for **:before** and **:after** pseudo-elements. When used for an actual element, it replaces what is normally displayed for this element.
- **content** may contain **icon**(*name*).

Name must be one of the following identifiers: **disc**, **circle**, **square**, **diamond**, **invisible** (1x1 transparent image), **right-target**, **left-target**, **filled-square**, **hollow-diamond**, **left-half-disc**, **right-half-disc**, **up**, **down**, **left**, **right**, **pop-up**, **pop-down**, **pop-left**, **pop-right**, **left-link**, **right-link**.

- **content** may contain **image**(*source*, *width*, *height*, **smooth|default**).

**source** Mandatory.

URL or path of an image file. Only GIF, JPEG, PNG files will be displayed by XXE but this must not prevent you from using other formats if your backend processor supports them.

A relative URL or path is relative to the location of the document being edited and not to the current working directory.

**width, height** Optional.

Dimension of the image in pixels. A length may optionally be followed by a standard CSS unit such as **px**, **in**, **cm**, **mm**, **pt**, **pc**, **em**, **ex**.

A negative length is interpreted as a maximum size. This is useful to display images as thumbnails.

Use **attr**(*name*) to get the desired dimension from the value of attribute named *name*.

**smooth|default** Optional.

The name of the algorithm used to change the image size: **smooth** means high-quality/slow and **default** means low-quality/fast.

Example (XHTML):

```
img {
  content: image(attr(src), attr(width), attr(height));
}
```

- **content** may contain **paragraph**(*content*, ..., *content*). This construct is used to force the creation of an anonymous paragraph where the generated/replaced content is added.

Example 1, this content will not wrap at word boundaries even if the white-space property specifies to do so:

```
content: "Very long line of generated/replaced text content \
that will *not* wrap.";
white-space: normal;
```

Example 2, this content will wrap at word boundaries:

```
content: paragraph("Very long line of generated/replaced text content \
that will wrap because the paragraph construct is used.");
white-space: normal;
```

This construct is also useful to specify complex generated/replaced cell content. See below.

- **content** may contain **rows**(*row\_spec*, ..., *row\_spec*, ...), **row**(*cell\_spec*, ..., *cell\_spec*, ...), or **cell**(*cell\_content*, ...). These constructs are used to specify tabular content.

Syntax:

- *rows\_spec* = **rows**(*row\_spec*, ..., *row\_spec*, *prop\_name*, *prop\_value*, ..., *prop\_name*, *prop\_value*)
- *row\_spec* = **row**(*cell\_spec*, ..., *cell\_spec*, *prop\_name*, *prop\_value*, ..., *prop\_name*, *prop\_value*)
- *cell\_spec* = **cell**(*cell\_content*, *prop\_name*, *prop\_value*, ..., *prop\_name*, *prop\_value*)
- *cell\_content* = string|**attr**(*name*)|**paragraph**(*content*, ..., *content*)

Example:

```
content: row(cell("Category", width, 20ex), cell("Choice #1"),
            cell("Choice #2"), cell("Choice #3"),
            font-weight, bold, color, olive,
            padding-top, 2, padding-right, 2,
            padding-bottom, 2, padding-left, 2,
            border-width, 1, border-style, solid);
```

*prop\_name*, *prop\_value* pairs may be used to specify the properties of a *cell*. Specifying such pairs at the row level is equivalent to specifying them for each cell contained in the row. Specifying such pairs at the rows level allows even more factoring.

Note that shorthand properties cannot be used there. Example: **padding-top**, **padding-left**, **padding-bottom**, **padding-right** must be used rather than the single shorthand property **padding**.

- **content** may contain **component**(*className*, *param*, ..., *param*).

*className* is the name of a Java class which implements the interface `com.xmlmind.xmledit.styledview.ComponentFactory`.

`com.xmlmind.xmledit.styledview.AttrCheckBoxFactory` is for now the only class that implements this interface. This class can be used to embed a boolean attribute editor in the styled view.

Example (APT):

```
verbatim:after {
  content: component("com.xmlmind.xmledit.css.AttrCheckBoxFactory",
                    box, no, yes);

  display: block;
  color: olive;
  font: .83em sans-serif;
  text-align: center;
  margin-left: auto;
  margin-right: auto;
}
```

which is rendered as:

Element <verbatim>

```
public interface ComponentFactory {
  Component createComponent(ElementModel element,
    Style style, StyleValue[] parameters,
    DocumentView docView,
    boolean[] stretch);
}
```

☒ box

Embedded boolean attribute editor used to edit the box attribute of the <verbatim> element





## 7 APPENDIX B: Whitespace processing

Whitespace processing is performed once at document load time:

- A Mac or DOS line break (`'\r'` or `"\r\n"`) is normalized to a single newline character (`'\n'`).
- If an element has the `xml:space` attribute set to `preserve`, all the whitespace characters it contains are preserved.

Note that unlike other attributes, the `xml:space` and `xml:lang` attributes of an element may be “inherited” from an ancestor element.

- Otherwise, the default behavior of XXE is to replace consecutive whitespace characters (`' '`, `'\t'`, `'\n'`, `'\r'`) by a single space character (`' '`).
- If a document has a DTD and if a *mix* element (an element described in the DTD as being able to contain text interspersed with child elements) is contained in a *structure* element (an element described in the DTD as only containing child elements), the space characters at the beginning and at the end of the mix, if any, are trimmed.

This behavior applies to:

- elements with a mixed content model,
  - elements with ANY as their content model,
  - elements which are referenced but not declared in the DTD.
- If a document has no DTD and if the “guess what spaces are not useful” option has been checked in the Options dialog box (see Options->Options), whitespace characters are trimmed from elements containing child elements separated by whitespace.

The `xml:space` attribute is very important for XXE because it is the only way to skip the compression and trimming of whitespace characters. *Do not forget to add support for this standard XML attribute to the DTDs you'll define.*

## 8 APPENDIX C: Keyboard shortcuts

Keyboard shortcuts invoking menu commands (ex. Ctrl-X that invokes Edit->Cut) are not listed here.

Mouse click	Description
Press-1	move caret and begin text selection; If there is no text close to the area clicked upon, select the element or #PCDATA found there
Drag-1	extend text selection
Release-1	end text selection
Shift-click-1	extend text selection to the mouse click location
Ctrl-click-1	select element or #PCDATA clicked upon
Double-click-1	select word (1)
Triple-click-1	select the whole #PCDATA

Key stroke	Description
Up	move caret to previous line (2)
Shift-Up	extend text selection to previous line
Ctrl-Up	select enclosing #PCDATA if any; select parent element otherwise
Ctrl-Shift-Up	select the preceeding sibling of the selected element
Down	move caret to next line
Shift-Down	extend text selection to next line
Ctrl-Down	select previously selected child element if any; if no child element was previously selected, select first child if any; cancel element selection otherwise
Ctrl-Shift-Down	select the following sibling of the selected element
Left	move caret to previous char.
Shift-Left	extend text selection to previous char.
Ctrl-Left	move caret to previous word
Ctrl-Shift-Left	extend text selection to previous word
Right	move caret to next char.
Shift-Right	extend text selection to next char.
Ctrl-Right	move caret to next word
Ctrl-Shift-Right	extend text selection to next word
Tab	insert a tab char. if the view of the current element accepts such char. (ex. if the value of <b>white-space</b> CSS property is <b>pre</b> ); move caret to next #PCDATA otherwise
Ctrl-Tab	move caret to next #PCDATA
Shift-Tab	move caret to previous #PCDATA
Ctrl-Shift-Tab	
Home	move caret to the beginning of #PCDATA
Shift-Home	extend text selection to beginning of #PCDATA
End	move caret to the end of #PCDATA
Shift-End	extend text selection to the end of #PCDATA
BackSpace	delete previous char.
Delete	delete next char.
Escape	cancel text or element selection
Enter	insert a newline char. if the view of the current element accepts such char. (ex. if the value of <b>white-space</b> CSS property is <b>pre</b> ); split the selected element in two parts (if allowed) otherwise
Ctrl-Space	insert a non breaking space char. (&nbsp;)
Insert	insert after the selected element an empty #PCDATA (if allowed)
Shift-Insert	insert before the selected element an empty #PCDATA (if allowed)
Ctrl-Insert	insert after the selected element an element of the same type (if allowed)
Ctrl-Shift-Insert	insert before the selected element an element of the same type (if allowed)

**Notes:**

1. On Mac, keyboard shortcuts use the Command key rather than the Control key.
2. Words are simply sequences of characters separated by white spaces.
3. XXE has no concept of lines. For example, for XXE, the “previous line” is simply the view of a #PCDATA which precedes (in the order of the document) the #PCDATA containing the caret. This “previous line” view is chosen to be the nearest one above the view of the #PCDATA containing the caret.

## 9 APPENDIX D: Enhancements and bug fixes

### 9.1 M1.3 Patch 3 (August 07, 2002)

Bug fixes:

- Workaround a Java 1.4 bug related to menu mnemonics. Example: saving a file with "Alt-F S" inserted an "F" in the document. This bug does not seem to happen with Java 1.3.1.

Enhancements:

- Slovak localization by Zdenko Podobný.
- German localization by Volker Hess.

### 9.2 M1.3 Patch 2 (May 22, 2002)

M1.3 Patch 2 has been released mainly because our installer built using ZeroG InstallAnywhere (an excellent product that we recommend) seemed not to be compatible with Java 1.4.

While at it, we have also qualified XXE under Java 1.4.

Bug fixes:

- Plain text "<em>word</em>" when copied to clipboard, was pasted as an **em** element.

Enhancements:

- Added support for Norman Walsh's Slides XML V2.0 DTD (a CSS style sheet and a template).
- Split Brian Lalonde's docbook41.css in two parts: sdocbook.css and docbook.css (which imports sdocbook.css).

This modularization was needed in order to cleanly implement slides.css.

XMLmind's original docbook.css (i.e. pre-Brian Lalonde's) has been removed from the distribution.

- On Mac, keyboard shortcuts use the Command key rather than the Control key. There are still *several problems related to mouse and keyboard events* on the Mac:
  - Ctrl-Click-1 does not work.
  - On a French keyboard, typing Cmd-Z sends a Cmd-W event to XXE and Cmd-Q sends a Cmd-A.
  - The swapped Cmd-Z/Cmd-W only works one time.
- The AZcheck spell-checker engine bundled with XXE has been upgraded to its latest version. The status of the dictionaries needed by AZcheck has been clarified (see Web site).

### 9.3 M1.3 Patch 1 (March 13, 2002)

Enhancements:

- Added support for the following encodings (both when loading a document and when saving it -- see updated Save tab in Options->Options dialog box):

- Windows-1252 (or Cp1252) Windows Latin1, West Europe.
- ISO-8859-15 Latin9, *updates Latin1 with symbols such as the Euro*.
- ISO-8859-2 Latin2, Central and Eastern Europe.
- ISO-8859-3 Latin3, Esperanto, Maltese, Turkish.
- ISO-8859-4 Latin4, Estonian, Latvian, Lithuanian, Greenlandic, Lappish.
- ISO-8859-5 Cyrillic.
- ISO-8859-7 Greek.
- ISO-8859-9 Latin5, Turkish.
- ISO-8859-13 Latin7, Baltic.

Do not forget to explicitly add this kind of declaration `<?xml version='1.0' encoding='ISO-8859-15'?>` at the beginning of your document templates because such encodings cannot be automatically detected.

We've only tested ISO-8859-15. If you have problems with the other newly supported encodings or if you need more encodings, tell us.

- XXE localized to Spanish by Jesus Redrado of Universidad de Navarra.
- `docbook41.css` enhancements by Brian Lalonde.
- Improved the look of the Tools->CheckValidity non-modal dialog box.

Bug fixes:

- In some cases, XXE used the same auto-generated value for ID attributes of newly created elements.

## 9.4 M1.3 (February 6, 2002)

Milestone 1.3.

Enhancements:

- Did a very simple integration of a beta version of AZcheck, a not-yet-released pure Java spell-checker product.
  - The English dictionary is the only one bundled with XXE, but several other dictionaries can be downloaded from the download page of XMLmind.  
Downloaded dictionaries must be copied to the `dict` subdirectory of the directory containing `xxe.jar`. *Make sure that downloaded dictionaries are saved as files with extension .jar.*
  - Like for search/replace and for printing, it is possible to check a subset of the document by *explicitly* selecting the element to be checked.  
Like for search/replace and for printing, restricting the command to the text selection is not possible.
  - The keyboard shortcut of Tools->CheckSpellingErrors is Ctrl-Shift-S (**S** like **S**pell).
- Added a Selection menu and corresponding buttons to the tool bar.
- Added File->New, File->Open, File->Save, Edit->Undo, Edit->Redo buttons to the tool bar.
- Added an new Edit->ApplyLastConversion command which is very handy when, for example, you want to convert several pieces of plain text to XHTML **strong** or **em**.  
Its keyboard shortcut is Ctrl-A (**A** like **A**gain). Previously, this was the keyboard shortcut of Edit->SelectAll but this command is not very useful.

- Slightly improved the indented output by adding some open lines at safe places.
- Added a new option to the Edit tab of the Options dialog box which could be useful for users who create their own DTDs. This option specifies what to do when a newly created element is inserted in the document and when the content model for this type of element has several possible choices for the child elements.
- A fairly complete DocBook style sheet named `docbook41.css` has been contributed by Brian Lalonde (Brian Lalonde's email address can be found at the beginning of `docbook41.css`). This style sheet becomes the default one for DocBook.

The previous one named `docbook.css`, is still available but will no longer be maintained.

If you want to use this new style sheet, edit your existing DocBook documents with a text editor and replace:

```
<?xml-stylesheet type="text/css" href="../css/docbook.css" ?>
```

by:

```
<?xml-stylesheet type="text/css" href="../css/docbook41.css" ?>
<?xml-stylesheet type="text/css" alternate="yes" title="Previous docbook stylesheet"
href="../css/docbook.css" ?>
```

Of course, replace `../css` by the actual URL of the directory containing your style sheet.

Bug fixes:

- `xxe.bat` does not launch XXE if installed in a directory like `C:\Program Files\xxe`, because the directory name contains whitespace characters.
- The detection of the charset of a CSS style sheet based on the BOM (Byte Order Mark) found at the beginning of the file was incorrect.

## 9.5 M1.2 Patch 5 (December 19, 2001)

*Official support for MacOS X.* However, for now, the MacOS X version of XXE is less usable than the other Unix and Windows versions for the following reasons:

- Java bug: whatever the paper size the user chooses, the margins and overall size of the printed output are incorrect.
- Java bug: printing cannot be canceled.
- Java bug: on a French keyboard, typing Ctrl-Z sends a Ctrl-W event to XXE and Ctrl-Q sends a Ctrl-A.
- Java bug: the standard shortcut Command-Q found on the Mac menu bar does not invoke the windowClosing handler registered by the application. Therefore, if a user types Command-Q instead of Ctrl-Q, the state of XXE will not be saved in `~/xxe`.
- Mac specificity: even when using a multi-button mouse, clicking with the right button will not pop-up the contextual menu (such as the "Expand All/Collapse All" menu of a tree view component). The user must use Command-Click instead.
- Mac specificity: some Macs, such as iBooks, have no Insert and Delete keys, therefore some of the most useful XXE shortcuts are not available.



Bug fix: if the style of the document element (that is the root element) depends on one of its attributes, this style was not properly updated each time this attribute was changed.

Enhancement: added a `defaultEncoding` parameter to both the APT and Javadoc format plug-ins. This parameter specifies the encoding to be used when loading and saving a file if no encoding has been specified by the connection. This encoding must be a Java encoding name (example `ISO8859_1`) and not a IANA charset name (example `ISO-8859-1`).

## 9.6 M1.2 Patch 4 (November 27, 2001)

Bug fix: fixes a bug related to valid but “strangely written” element content models found in DTDs such as RSS 0.91. This bug is described below in subsection October 1, 2001.

Note that even with this bug fix, XXE still doesn't check the validity of the DTD: DTD writers really need to check their DTDs using a third party tool such as James Clark's SP (in XML mode).

Users that just write Docbook and XHTML documents don't need to upgrade.

## 9.7 M1.2 Patch 3 (November 23, 2001)

Bug fix: in a CSS, a reference to the attribute *A* of the element which is the subject of the selector is specified using construct **attr**(*A*).

CSS being case insensitive, all CSS identifiers are converted to lower-case character strings (*except in selectors*).

Therefore, if you specify attribute name *A* as an identifier, *A* is automatically converted to lower-case, which may be a problem because XML is case-sensitive.

In order to specify non-lower-case attribute names, one must specify *A* as a quoted CSS string rather than a CSS identifier. Example: specify `attr( "HRef" )` rather than `attr( HRef )`.

Unfortunately, a bug prevented XXE from accepting construct **attr**(*string*).

## 9.8 M1.2 Patch 2 (November 7, 2001)

Workaround a bug which prevents IBM Java from displaying menu labels and other resources.

## 9.9 M1.2 Patch 1 (October 25, 2001)

Bug fixes:

- Dialogs appeared in iconic state the second time they were used under the control of certain window managers such as kwin, KDE2 window manager, and fvwm2. (No such problem under Windows or when using kwm, KDE1 window manager, or mwm.)

We have found a workaround for these Java(?) / window manager(?) bugs. Things are not perfect (there are still focus, window location and window size problems for fvwm2) but at least XXE becomes usable.

- The search facility tended to select found text from the beginning of the element to the beginning of the next one which is not as convenient as selecting found text from the beginning of the element to its end.

Bug workarounds:

- Under Windows, multi-font text location and size problems whether on screen or when printing can be solved by switching from Java generic fonts (Serif, SansSerif, Monospaced) to Windows native TrueType fonts (Arial, Verdana, etc).

This can be done by using the Style tab of the new Options->Options dialog box.

## 9.10 M1.2 (October 19, 2001)

### Milestone 1.2.

The new printing feature contained in this release requires at least a Java runtime 1.3. Therefore, the Java 1 platform is no longer supported. *Do not upgrade if you need to keep working with a Java runtime 1.1.*

#### Enhancements:

- Added a literal and regular expression search and replace facility.
- The document being edited can now be printed. It is also possible to just print the selected element (for example: the selected section).

A `@media print` section has been added to all the CSS style sheets bundled with XXE to slightly customize them for printing.

Note that this is just a quick and dirty printing facility: CSS constructs which are specific to paged media, such as `@page` or `page-break-before`, are ignored. High-quality printing can be achieved by using third party XSL tools.

- Using the new Options dialog box, it is now possible to specify the user preferences in terms of font face, font size, text color and background color.  
Of course, these preferences will have no effect of the word processor-like view if the corresponding style properties have been “hardwired” into the CSS style sheet. (This is not the case with all the style sheets bundled with XXE.)

- Edit->Join, the inverse command of Edit->Split, concatenates the selected element with the preceeding one, if this preceeding element is of the same type.
- Selecting an element automatically moves the caret inside it, unless the caret is already inside one of the descendants of this element.
- Ctrl-Shift-Up arrow selects the preceeding sibling of the selected element. Ctrl-Shift-Down arrow selects the following sibling of the selected element.

- Better user feedback during lengthy operations such as document loading or printing.

During certain “safe” steps, it is now possible to cancel these lengthy operations by clicking on the spinning beach ball icon or on the status message located at the bottom of XXE main window.

- Undoing/redoin an action moves the caret inside the modified element. If the modified element contains no text, it is selected instead.
- Added command Tools->RefreshSelectedElement (shortcut Ctrl-L) that rebuilds the view of the selected element.
- The algorithm used to scale images can now be specified in the style sheet by using *scaling*, a new optional parameter of the **image**(*source, width, height, scaling*) construct.

Parameter *scaling* has two possible values: **smooth** which means high-quality/slow and **default** which means low-quality/fast.

See APPENDIX A: CSS2 properties and values supported by XXE.

**Bug fixes:**

- Using Tools->EditAttributes disabled (without “graying out” the tool bar icons or Edit menu entries) most editing commands until you clicked elsewhere in the document view to change the selected element.
- Typed characters from range \u00A1 to \u00A7 (examples: £, §) were ignored.
- Images displayed as replaced content for some elements were cached. If you changed the image data without changing its file name (example: you update screenshot12.jpeg) and you typed Ctrl-L (refresh selected element), the old image was still displayed.
- Undoing/redoing attribute changes didn't properly move the caret when the caret was inside the element being edited.
- Pasting some text (or inserting a text file) containing control characters caused XXE to silently create a non-well formed XML document.
- The Tools->CheckValidity window crashed after the following sequence of actions: Tools->CheckValidity displays some validity errors in its window, these errors are fixed by the user, Tools->CheckValidity displays a blank window, user uses Edit->Undo to revert to the invalid state, Tools->CheckValidity is used again to redisplay the validity errors.
- An element with content model ANY can contain #PCDATA or any declared element type. Previously, this type of element was not allowed to contain #PCDATA.
- For consistency, element types that have not been declared in the DTD (using <!ELEMENT>) referenced in the DTD (for example, referenced in the content model of other element types or having an <!ATTLIST> declaration) are now treated as first class element types.

Therefore the trimming of white space in a “mix” contained in a “structure” now applies to:

- elements with a mixed content model,
- elements with ANY as their content model,
- elements which are referenced but not declared in the DTD.

See APPENDIX B: Whitespace processing.

- In a CSS style sheet, @page, @fontface and @media (other than those of type screen or print) were not properly skipped.
- The XHTML style sheet has been cleaned-up to only support the xhtml1-strict DTD.

## **9.11 October 1, 2001**

- Bug fix: *important performance problems* related to the interaction between the document view and the scroll bars. These performance problems were not present in Milestone 1. These problems are important enough to justify an upgrade from the original XXE M1.1.
- Bug fix: embedded Java components were not properly moved in the document view when the document is modified by the user.
- Bug fix: in the non-styled (tree) view, some borders were drawn with a raised relief instead of a much nicer sunken relief.

- Bug that *cannot be fixed without a major rewrite*: some DTDs such as RSS 0.91 are not supported by XXE because they contain strangely written, but valid, content models for some elements. Example:

```
<!ELEMENT channel (title | description | link | language | item+ |
                    rating? | image? | textinput? | copyright? |
                    pubDate? | lastBuildDate? | docs? | managingEditor? |
                    webMaster? | skipHours? | skipDays?)*>
```

The author of this DTD wants in fact to express the following content model:

```
<!ELEMENT channel (title & description & link & language & item+ &
                    rating? & image? & textinput? & copyright? &
                    pubDate? & lastBuildDate? & docs? & managingEditor? &
                    webMaster? & skipHours? & skipDays?)*>
```

where non-XML operator ' & ' means that the particles can occur in any order.

The workaround is to rewrite the content models that are rejected by XXE. What follows is equivalent in terms of validation to the example above:

```
<!ELEMENT channel (title | description | link | language | item |
                    rating | image | textinput | copyright |
                    pubDate | lastBuildDate | docs | managingEditor |
                    webMaster | skipHours | skipDays)*>
```

## 9.12 September 21, 2001

### Milestone 1.1

- The Javadoc plug-in adds Javadoc(TM) support to XXE. With XXE, you can now edit the Javadoc comments contained in a Java file using a word processor-like view.
- Bug fix: under certain conditions, when loading a document with a DTD, trimming the white space of an element with mixed content resulted in useless empty #PCDATA.
- Bug fix: moving the caret into a newly created collapsed TreeView caused an ArrayIndexOutOfBoundsException.
- Bug fix: under certain conditions, when the tag auto-completion feature was used, the tag list contained in the tag chooser dialog box was not properly scrolled.
- The new menu command Edit->InsertFile may be used to paste the content of a file (XML or plain text) before, into or after the selected element.

If after parsing the content of the file and after analyzing what can be done in the context of the selected element, XXE finds that such insertion is not allowed, XXE will simply beep to signal that the command has not been completed.

- The caret blinks when the document view has the keyboard focus.
- When a document is scrolled, the caret is moved so as to try to keep it always visible.
- When you switch from a style sheet to another (or to No style sheet) using the Style menu, all the document marks are preserved (caret, text selection, element or #PCDATA selection). Previously, these marks were cleared.
- When there is no ambiguity, typing Enter splits the selected element in two parts.

Example: typing Enter when an XHTML **p** is the selected element, splits the **p** in two parts, while typing Enter when an XHTML **pre** is the selected element, inserts a linefeed character into the **pre**.

The Edit->Split menu command (shortcut Ctrl-Enter) is still the standard way to split the selected element in two parts.

- The arrow buttons of the scrollbars now scroll the document view by a reasonable amount. Previously the scroll increment was 1 pixel.
- The algorithm used to scale images has been changed from high-quality/slow to low-quality/fast. The size of the “thumbnails” used in all the CSS style sheets bundled with XXE has been changed from 200x200 max. to 400x200 max.

### 9.13 September 7, 2001

- Bug fix: a build error may prevent XXE from starting when using Sun Java 1.3.1.
- When saving a file as indented XML, the indentation can be set to 0 to achieve a not uncommon formatting style. Previously, the minimum value for indentation was 1. (See Options dialog box.)
- An element displayed as a tree, displays its attributes (if any) as well as its child elements (if any). Previously, the attributes were not displayed.

Note that the attributes are still not editable in the document view: you must use the Attribute Editor dialog box.



*An Ant Example: most Ant elements have just attributes and no child elements.*

### 9.14 August 17, 2001

- Bug fix: the *indented* XML file generated for a document that contains empty elements with a non-empty content model was not well-formed.

### 9.15 August 3, 2001

Milestone 1.