

XMLmind XML Editor

User's Guide

V3.0

www.xmlmind.com/xmleditor
xmleditor-support@xmlmind.com
Pixware SARL
Immeuble Capricorne
23 rue Colbert
78180 Montigny Le Bretonneux
France
Phone: +33 (0)1 30 60 07 00
Fax: +33 (0)1 30 96 05 23

December 2, 2005

Contents

1	Install	4
1.1	Installing XXE	4
1.1.1	Requirements	4
1.1.2	Install on Unix	4
1.1.3	Manual install on Windows	5
1.2	Content of the installation directory	5
1.3	Acknowledgments	6
2	Tutorial: basics	7
2.1	Starting XXE	7
2.2	Creating a new document	8
2.3	Inserting elements	10
2.4	Selecting elements	13
2.4.1	The implicit element selection	13
2.4.2	The text selection	13
2.4.3	The node selection	13
2.5	Navigating through elements	16
2.5.1	Using Tab to go from a #text to the other, just like in a form	16
2.5.2	XXE makes a difference between the end of a #text node and the beginning of the #text node next to it	16
2.6	Copy, cut, paste, delete	18
2.6.1	Copy, cut, paste, delete applied to the text selection	18
2.6.2	Copy, cut, paste, delete applied to the node selection	18
2.7	Splitting and joining elements	20
2.7.1	Simple Split and Join	20
2.7.2	Split and Join generalized	20
2.8	Replacing elements	23
2.9	Converting elements	27
2.9.1	Convert applied to the text selection	27
2.9.2	Convert applied to the node selection	27
2.10	Editing element attributes	29
2.10.1	Required attributes in newly created elements	30
2.11	Checking document validity	32
3	Tutorial: creating a modular document	34
3.1	Creating the Copyright.html document	34
3.2	Inserting a reference to Copyright.html into the XHTML page	35
3.3	Extensive use of the "Copy as Reference" command	38

4 Being productive with XXE

42

This document is essentially a tutorial for XMLmind XML Editor (XXE for short). It also contains installation instructions for Unix and information about the content of the distribution.

1 Install

1.1 Installing XXE

1.1.1 Requirements

- Sun or Apple Java[tm] runtime 1.4.1 or above.
- At least 256Mb of memory and a 600MHz CPU.
- 60Mb of free disk space, 120Mb for a self-contained distribution which includes a Java[tm] 1.5 runtime.

XXE is officially supported on Windows NT/2000/XP, on Linux 2.4 and on Mac OS X 10.3/10.4. It is possible to use it on other Java[tm] 1.4.1+ platforms (e.g. Solaris), but without support from XMLmind.

XXE has been tested with:

- Sun Java[tm] runtime 1.4.1+ (up to 1.5.0_05) under Windows NT4, 2000, XP Home, XP Professional (up to SP2), SuSE Linux 9.0 and 9.3.
- Mac OS X 10.3.9 and Java[tm] 1.4.2_09, Mac OS X 10.4.2 and Java[tm] 1.4.2_09.
- Apache HTTP server 2.0.44 + mod_dav with mod_dav_fs and mod_dav_svn (Subversion) back-ends.
- Apache HTTP server 1.3.19 + mod_dav 1.0.2.
- Various Unix FTP servers.
- Windows 2003 IIS HTTP server + its WebDAV extension.
- Windows 2003 FTP server.

1.1.2 Install on Unix

Procedure:

1. Make sure that the Java[tm] bin/ directory is referenced in the \$PATH and, at the same time, check that the Java[tm] runtime in the \$PATH has the right version:

```
$ java -version
java version "1.5.0_04"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_04-b05)
Java HotSpot(TM) Client VM (build 1.5.0_04-b05, mixed mode, sharing)
```

2. Unpack the XXE distribution somewhere.

```
$ cd
$ tar zxvf xxe-std-30.tar.gz
$ ls xxe-std-30
addon/
bin/
demo/
doc/
```

3. XXE is intended to be used directly from the xxe-std-30/ directory. That is, you can add xxe-std-30/bin/ to your \$PATH.

```
$ xxe-std-30/bin/xxe &
```

1.1.3 Manual install on Windows

Manual install on Windows is similar to the install on Unix. XXE must be started using `xxe-std-30\bin\xxe.bat` rather than the `xxe-std-30/bin/xxe` shell script.

1.2 Content of the installation directory

bin/ Contains XXE code (.jar files) and many scripts used to start XXE and its associated utilities.

bin/xxe, xxe.bat Scripts used to start XXE. Use `xxe` on any Unix system. Use `xxe.bat` on Windows.

bin/xxe.exe, xxe.jstart Only when installing XXE on Windows using any of the `*setup*.exe` distributions. `xxe.exe` is XXE launcher and `xxe.jstart` is its (plain text, UTF-8 encoded) associated parameter file.

bin/deploywebstart, deploywebstart.bat Scripts used to generate a Java[tm] Web Start configuration (.jnlp file, signed jars, etc) from a possibly customized XXE distribution. Use `deploywebstart` on any Unix system. Use `deploywebstart.bat` on Windows.

The `deploywebstart` command-line tool is documented in the *XMLmind XML Editor - Configuration and Deployment*.

bin/xsdvalid, dtdvalid, dtdtoxsd, xsdvalid.bat, dtdvalid.bat, dtdtoxsd.bat Scripts used to run `xsdvalid`, `dtdvalid` and `dtdtoxsd`. See *XMLmind Xsdvalid Toolset User's Guide* for more information about these tools.

bin/rngvalid, rngvalid.bat Scripts used to run Jing, James Clark's RELAX NG validator.

bin/rngdoc, rngdoc.bat Scripts used to generate a simple reference manual from a RELAX NG schema.

The generated reference manual is intended to help content authors create instances conforming to a given schema. This manual is not intended to help schema authors document their design.

bin/csscheck, csscheck.bat Scripts used to check the syntax of CSS style sheets written for XXE.

bin/convertdoc, convertdoc.bat Scripts used to run `convertdoc`. This tool allows to execute XXE process commands from the command line, exactly as if these process commands were executed from XXE.

See *XMLmind XML Editor - Commands* for more information about this tool.

bin/*.jar All the (non-system) Java[tm] class libraries needed to run XXE:

- `xxe_app.jar`, `xxe.jar` contains the code of XXE.
`xxe_help.jar` contains the online help of XXE.
`xsc.jar` is XMLmind Spell-Checker product, the best buy for spell-checker written entirely in the Java[tm] language. For more information, please visit <http://www.xmlmind.com/spellchecker/>.
- `jh.jar` is the standard Java[tm] help engine.
`resolver.jar` contains Apache XML Commons Resolver which implements catalog-based entity and URI resolution.
`xp.jar` is James Clark's XML parser. See <http://www.jclark.com/>.
Substantial parts of `xsdregex.jar`, James Clark's XSD to Java Regular Expression Translator, have been directly added to `xxe.jar` (which is why file `xsdregex.jar` is not included in the distribution). Download original package from <http://www.thaiopensource.com/download/>.
Package `com.jclark.xsl.expr` contains the implementation of XPath 1.0 used by XT, James Clark's XSLT engine. A modified version of this package, renamed `com.xmlmind.xmledit.xpath`,

has been directly added to `xxe.jar` (which is why file `xt.jar` is not included in the distribution). Download full XT from <http://www.jclark.com/xml/xt-old.html> or from <http://www.blnz.com/xt/index.html>.

`relaxng.jar` is Jing version 20030619, James Clark's RELAX NG validator, slightly modified for use in XXE. The details of the modifications are found in `relaxng.README`.

`ftp.jar` is Bruce Blackshaw's FTP client. See <http://www.enterprisedt.com/>.

`http.jar` is Ronald Tschalär's HTTP client. See <http://www.innovation.ch/>.

`saxon.jar` is Michael H. Kay's XSLT engine. See <http://saxon.sourceforge.net/>.

`kunststoff.jar` contains the Kunststoff Look&Feel. See <http://www.incors.org/>.

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.

In general, these packages are not the most widely used/most well-known tools of their category. But we have also tested the most widely used/most well-known tools of their category and have found the packages listed above to be smaller, faster and more reliable than the big names. Many, many thanks to the authors of these packages.

- `translatexxe.jar` contains the code of the `translatexxe` command-line utility. This tool can be used to translate the messages (menu labels, error messages, etc) contained in XMLmind XML Editor.

There is no associated script. Start `translatexxe` using `"java -jar translatexxe.jar"`.

bin/icons/ Contains desktop icons for XXE.

bin/mac/ Contains files (e.g. `Info.plist`) used on the Mac to create `XMLmind.app` from a `.zip` or a `.tar.gz` distribution.

addon/ The `addon/` directory is the place where XXE finds its extensions, whatever is the type of this extension: configurations, plug-ins, translations to languages other than English, spell-checker dictionaries.

This `addon/` directory is recursively scanned by XXE at startup time. Therefore, feel free to organize it as you want.

addon/config/ Contains configuration files for the following XML applications: DocBook, Simplified DocBook, Slides, XHTML, XXE Configuration, W3C XML Schema.

The content of a configuration file, which specifies a customization of XXE for a specific XML application, is described in detail in *XMLmind XML Editor - Configuration and Deployment*.

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

addon/translate/ Each `.jar` file found in this directory contains XXE messages translated to a language other than English. `fr.jar` contains French messages, `de.jar` contains German messages, etc.

doc/ Contains XMLmind XML Editor documentation in HTML and PDF (Acrobat) formats.

Note that:

doc/user/ Contains this user's 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 open them in XXE in order to evaluate the editor.

demo/ Contains XML documents that can be opened in XXE to demo some of its features.

1.3 Acknowledgments

On Windows, XMLmind XML Editor installer (i.e. `*setup*.exe`) is built using **Inno Setup** by Jordan Russell's software. XMLmind highly recommends this excellent and free-to-use tool.

2 Tutorial: basics

This tutorial assumes that the reader has a minimal knowledge of XML (that is, the reader knows what is an element, an attribute, etc) and a minimal knowledge of HTML or XHTML (that is, the reader knows that **p** is the tag for a paragraph, that **ul** is a list, etc).

It is recommended that, using XXE, the reader repeats each action described in this tutorial. (On Mac, use the Command key instead of the Control key, except for Ctrl-Tab and Ctrl-Space.)

2.1 Starting XXE

- If you have installed an auto-installable Windows distribution (`xxe-std-30-setup.exe`), XXE can be started by double-clicking on the icon of `xxe.exe` or by using the "XMLmind XML Editor" shortcut added to the **Start** menu.

If you have installed a zipped distribution, XXE can be started by typing `xxe` (or `xxe.bat`) from the command prompt, optionally followed by the name of one or several XML documents.

Example:

```
C> xxe C:\xxe-std-30\doc\user\userguide.xml
```

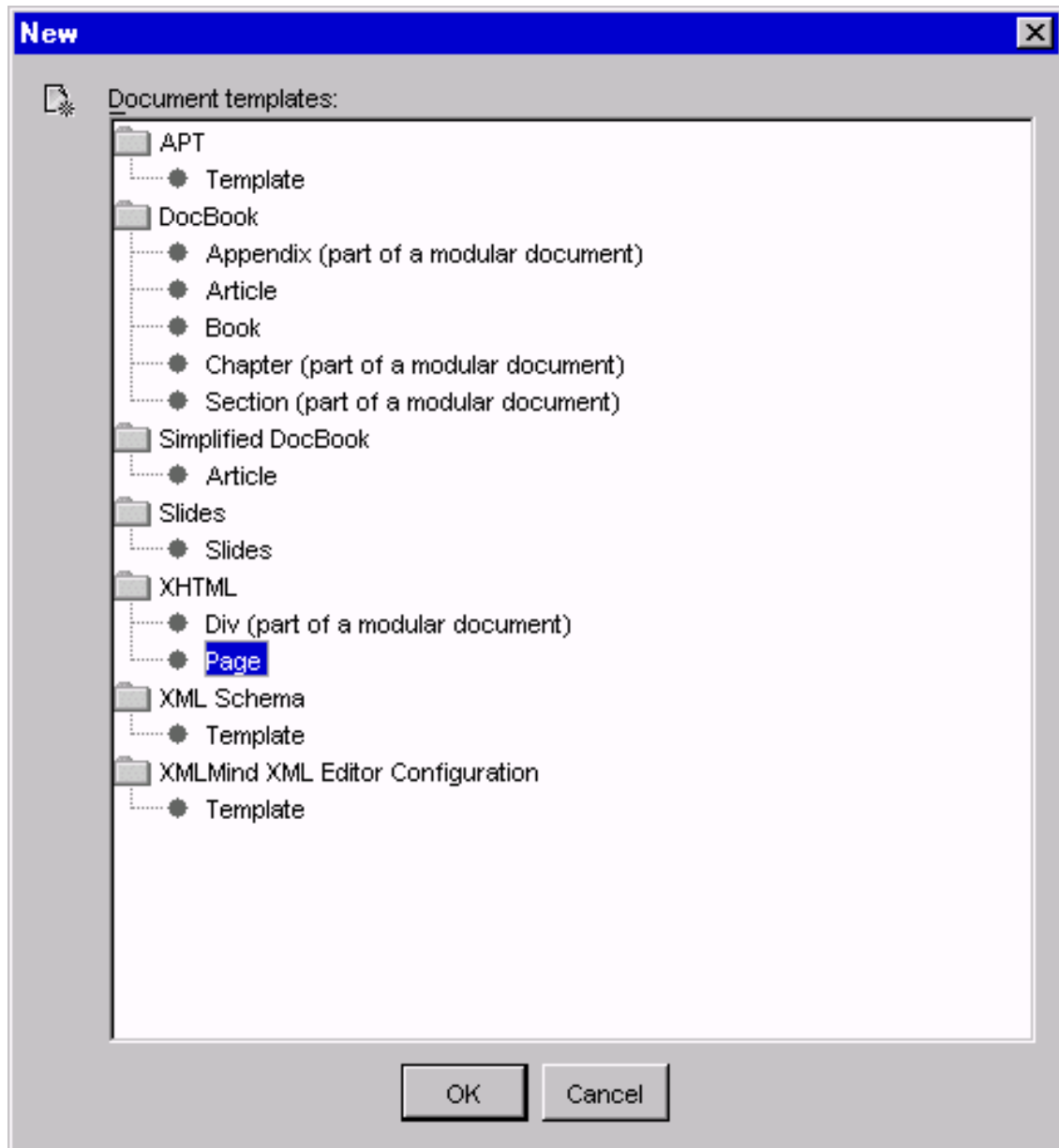
- On Unix, XXE can be started by typing `xxe` from an xterm, optionally followed by the name of one or several XML documents.

Example:

```
$ xxe &
```


2.2 Creating a new document

Use **File|New** and choose a document template from the following dialog box.

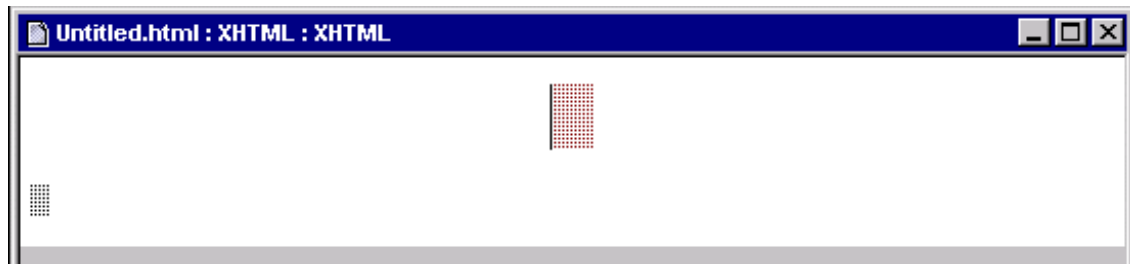


The File/New dialog box

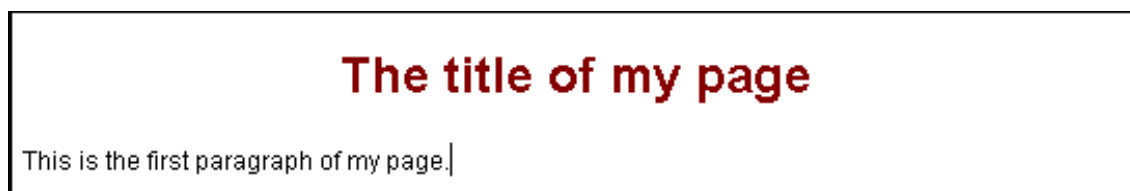
Alternatively, use **File|Open as Template** and choose an existing XML document. A copy of this existing XML document will be emptied to serve as the skeleton of your new document. For example, use this command to select `userguide.xhtml` (User's Guide in XHTML format) or `userguide.xml` (User's Guide in DocBook format) file contained in the `doc/user` subdirectory of your XXE distribution.

In this tutorial, we have chosen to create a XHTML page (strict DTD).

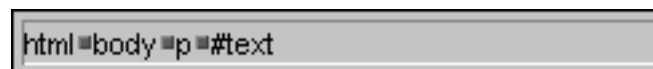
The newly created document (`Untitled.html`) looks like this:



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, just below the tool bar icons, the *node path bar* changes its label.



The node path bar shows where is the caret (also called insertion cursor).

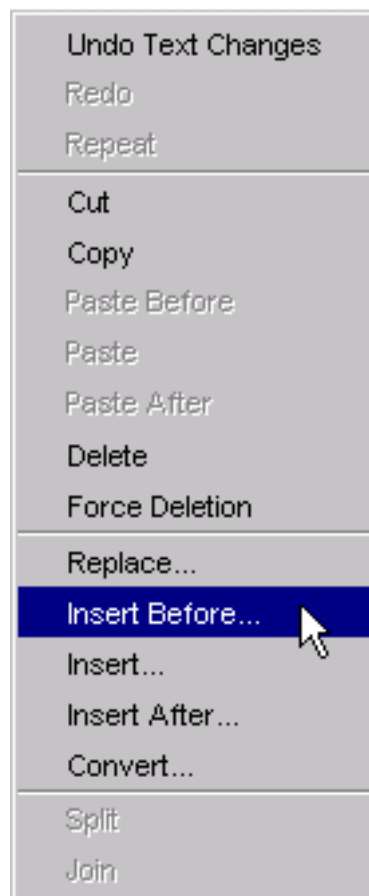
The node path bar does so by displaying the name of the node (anonymous text node **#text**) containing the caret preceded by the name of each of its ancestors elements in the containment hierarchy (element **p** contained in element **body** contained in element **html**).

2.3 Inserting elements

In this section, we'll get familiar with the following commands:

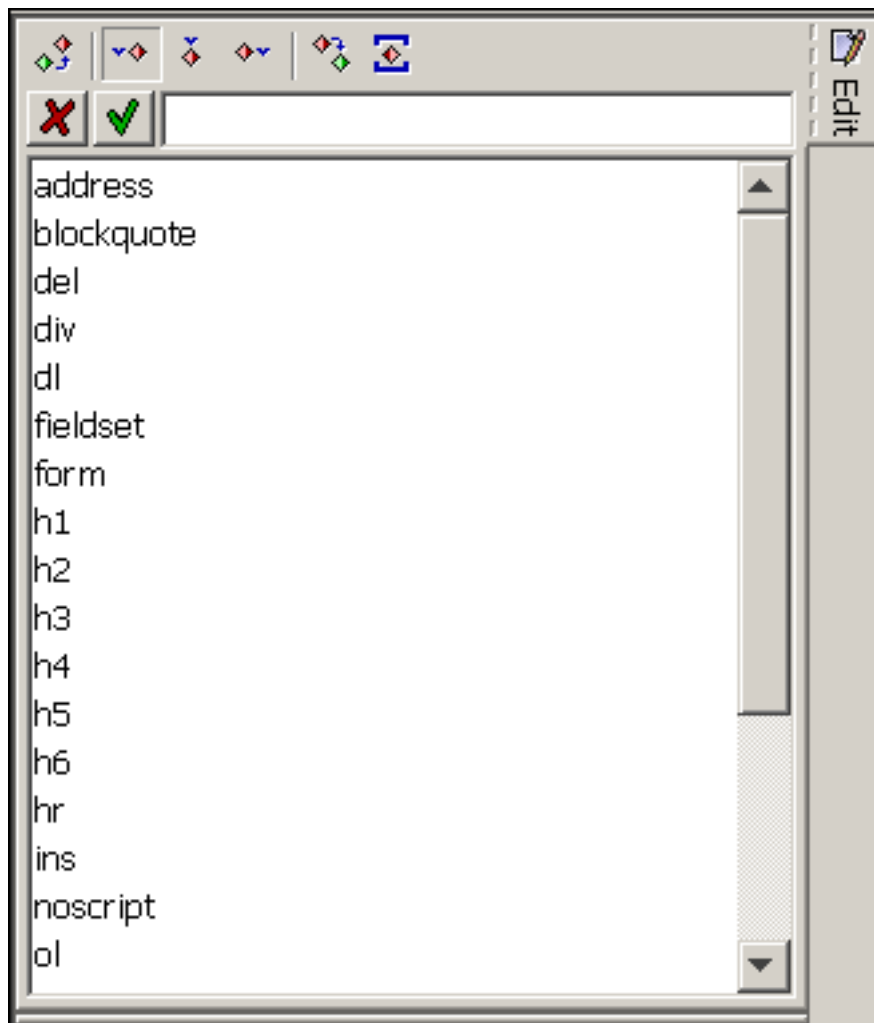
1. Insert before element
2. Insert inside element, at caret location
3. Insert after element

These commands are accessible using the **Edit** menu bar menu, using the **Edit** tool "tab" or using keyboard shortcuts, but in this tutorial we'll limit ourselves to using the **Edit** popup menu. Therefore the notation **Edit|Insert** means "click anywhere in the document view using the right mouse button and select menu item **Insert** in the **Edit** popup menu".



The Edit popup menu

Click anywhere inside the paragraph and execute **Edit|Insert Before**. The keyboard focus is given to the **Edit** tool. This pane now lists all the elements you can insert before a **p** and therefore, is ready to use.



The **Edit** tool listing all elements you can insert before a **p**

Select **h1** from the list (a single click on the list item is sufficient). The newly inserted element has a red border around it. Do not care about that, just type the text of the heading and the red border will go away.

Click inside the **p** before a word and execute **Edit|Insert**, the same element chooser dialog pops up listing all the inline elements you can insert inside a **p**. Select **strong** from the list and type a few words in bold font.

Your document should look like this:



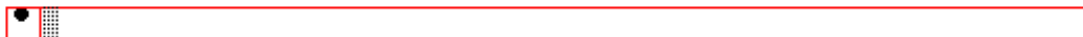
This time, use **Edit|Insert After** and add a **ul** (unordered list) after the **p**. Type the text of the first list item (**li**).

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

The title of my page

This is the title of my first heading

This is the first paragraph of my **XMLmind XML Editor tutorial** page.



Click to get rid of the red border, then using what you have learned, add two more **lis**: one before the first **li** you have created, the second after it.

The title of my page

This is the title of my first heading

This is the first paragraph of my **XMLmind XML Editor tutorial** page.

- List item #1.
- List item #2.
- List item #3.

2.4 Selecting elements

Editing commands are applied to the selection.

XXE supports three types of selection:

- The text selection, found in all text editors and word processors.
- The node selection, which can contain one or several nodes.
- The implicit element selection.

Some commands can be applied to any type of selection, for example: **Edit|Convert**. Other are more restrictive, for example: **Edit|Insert Before** can be applied to the implicit element selection or to an explicit single node selection.

2.4.1 The implicit element selection

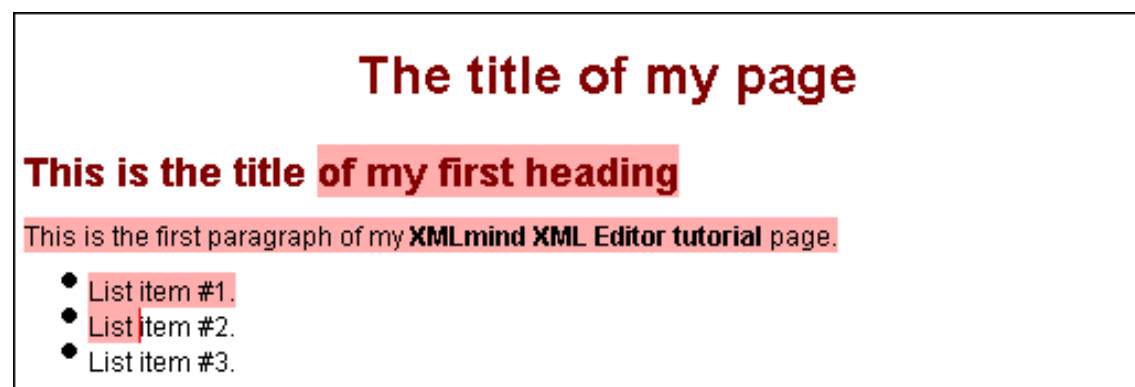
The element containing the caret is implicitly selected. Therefore you can apply commands to it without making any special effort.

This is one of XXE nicest feature, even if it needs to be learned because it has no equivalent in the word processor world.

In the first section of this tutorial, you have already used the implicit element selection to insert an **h1** before the **p** and an **ul** after it.

2.4.2 The text selection

Selecting text in XXE is not different from selecting text in any text editor. Click in the middle of the **h1** and drag the mouse across the document until the middle of the second **li** is reached.



2.4.3 The node selection

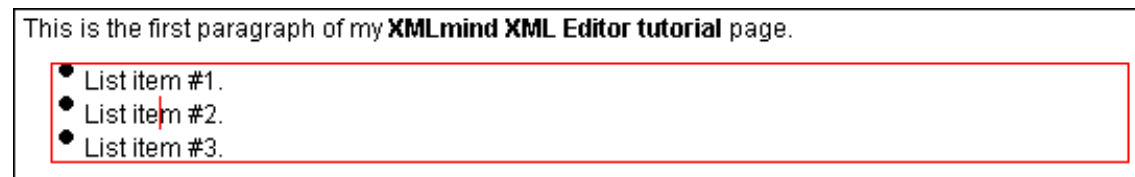
Most general method Now we want to add a **pre** (preformatted text) after the **ul**.

We have learned that clicking inside the a **li** implicitly selects it therefore we know how to add a new **li** after this one but how to select its parent, the **ul**, to add a **pre** after it?

For that we need to use *explicit element selection*. Click inside any **li** and then click on word **ul** inside the node path bar.



This explicitly selects the corresponding node in the document view.



Note that explicitly selected nodes are drawn with a red border around them.

Alternatively, click inside any **li** and type **Ctrl-Up Arrow** 3 times: first time to select the **#text** node, second time to select the **li**, third time to select the **ul**.

Now that **ul** is selected, use **Edit|Insert After** to add a **pre** after it.

Type 2 lines of text in this **pre**. Do not be worried by the “carriage return icon” always displayed by XXE if the last character of a **#text** node is a newline character.



Directly selecting a node Clicking near some text moves the caret inside the text and no node is explicitly selected.

Instead of simply clicking, try to *Ctrl-click* on the first **p**. A **#text** node is explicitly selected and the node path bar tells you which one.

Selecting a **#text** node is rarely what you want to do. You'll often need to Ctrl-click again without moving the mouse. This will select to parent of the selected **#text**, that is, the **p**. Ctrl-clicking again without moving the mouse would select the parent of the **p** which is a **body** and so on. When doing a series of Ctrl-clicks, always look at the node path bar to know precisely where you are. Also, do not Ctrl-click several times too fast otherwise the editor will think you are double-clicking or triple-clicking and therefore, selecting elements that way would not work.

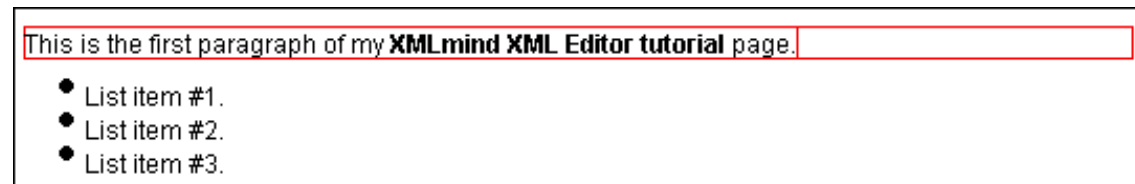
Click on the first **p** to cancel the selection because we are going to study another method to select it. Now try to Ctrl-click, not inside the text itself, but *in the blank space at the right of the text* of first **p**. Notice that one Ctrl-click is sufficient to directly select the whole **p**.

Similarly, you can directly select a node when a non-editable “decor” has been generated for it. For example, simply *click on the bullet* of a **li** to select it. This type of non-editable decor is very common, for example: the image displayed for a figure, the number of a section, the border of a table, a node icon or element name in the tree view, etc.

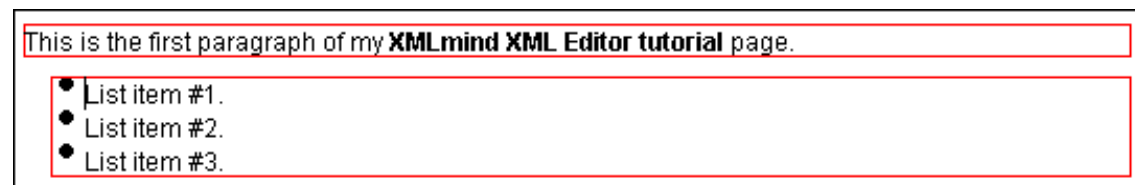
Selecting a node range What if you want to put first **p** and **ul** inside a **blockquote**? The answer is select both of them first and then use **Edit|Convert** (described below in this tutorial).

With XXE it is possible to select a *node range*, that is, adjacent children of the same parent element.

Select first **p** using any of the methods described above.



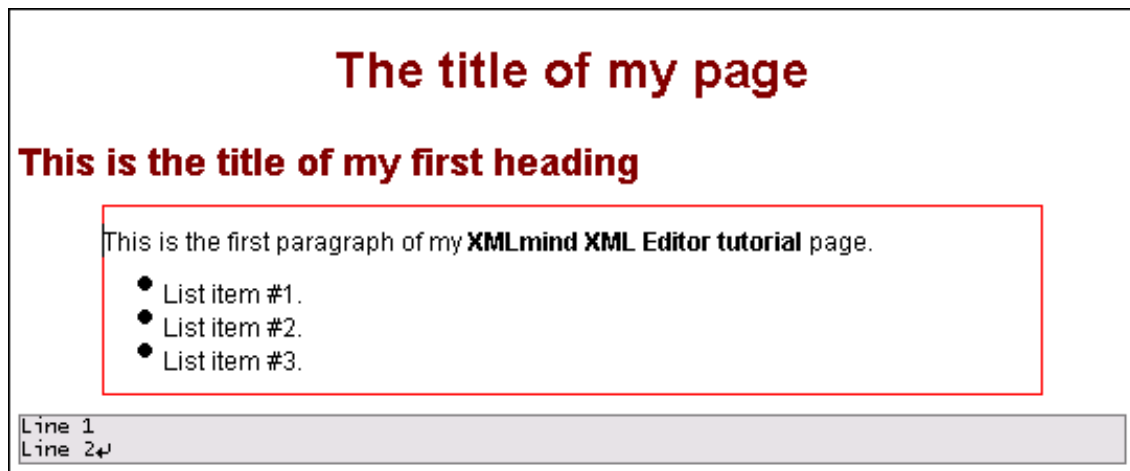
Then *Ctrl+Shift-click* anywhere on the **ul**.



Note that extending the node selection is easy because you do not have to be precise. The reason is that there is no ambiguity about what to select: a **#text** or a **li** is not a sibling of first **p**, only the **ul** is a sibling of first **p**.

Alternatively, you could have typed **Esc** then **Right Arrow** to extend node selection to the **ul**.

Now use **Edit|Convert** and choose **blockquote**.



2.5 Navigating through elements

2.5.1 Using Tab to go from a #text to the other, just like in a form

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

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

What if you really want to insert a Tab character into a **pre** (or any element which allows Tab characters to be inserted in the text flow just as any ordinary character)? The answer is: type Ctrl-Tab instead of Tab.

2.5.2 XXE makes a difference between the end of a #text node and the beginning of the #text node next to it

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

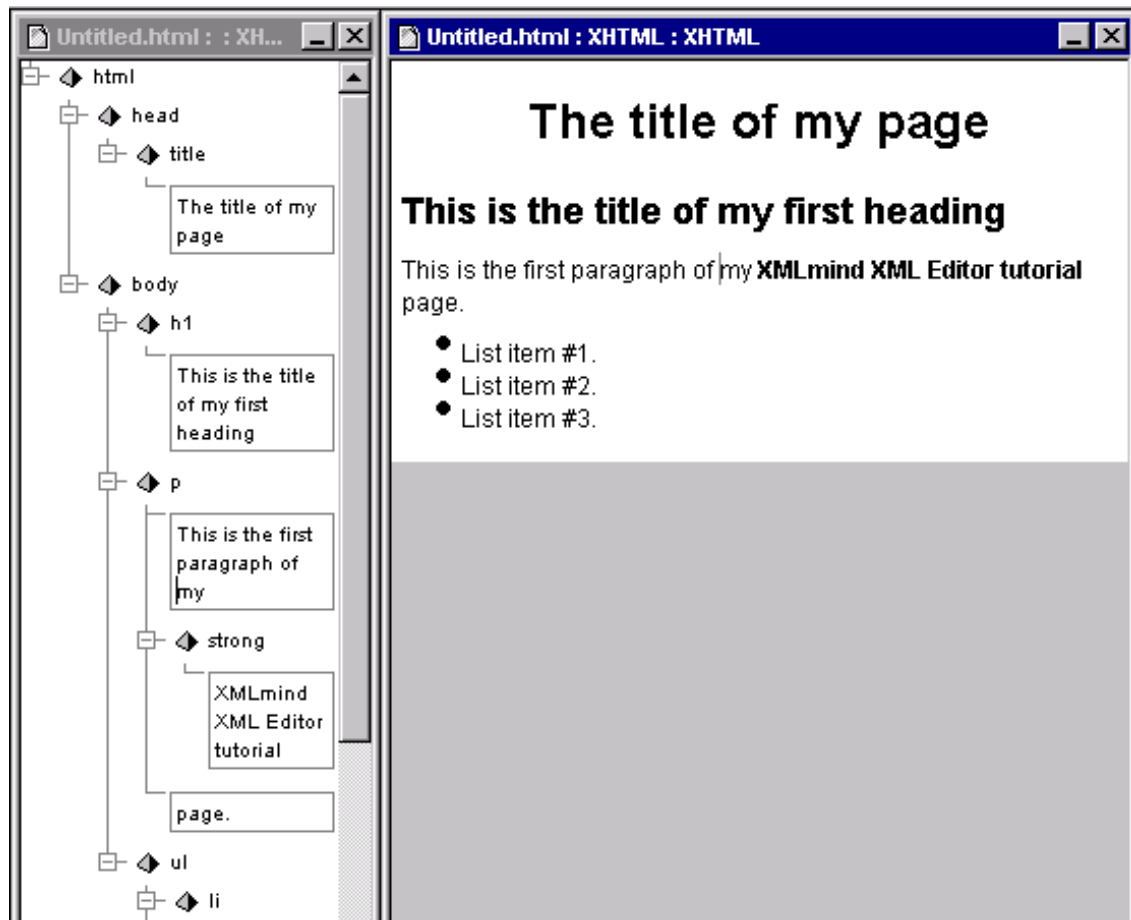
Click on the first **p** to the left of the **strong** and press on the Right key to move the caret in the direction of the **strong**.

Just before reaching the **strong** element, you'll notice that pressing 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.

Note that on the **p/strong** boundary, the caret makes no visible movement but the node path bar displays different paths (**html-body-p-#text** and **html-body-p-strong-#text**).

Use menu command **Style|Both Tree and Styled Views** to view the document using two views: the styled view at right and a low-level, hierarchical view called the tree view, at left.



Both tree and styled Views side by side

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-#text** to the beginning of the **strong-#text** and vice-versa.

Close the tree view (using the standard close button in its title bar) and maximize the styled view using **Window|Stack**.

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 several times. Notice what happens on the **p/strong** boundary. Then use **Edit|Undo** as many times as needed to undo this typing.

Click inside the **strong** and type on the Backspace key several times. Notice what happens on the **p/strong** boundary. Then use **Edit|Undo** as many times as needed to undo this typing.

2.6 Copy, cut, paste, delete

2.6.1 Copy, cut, paste, delete applied to the text selection

The text selection is used not only to specify a range of characters but also a range of nodes and child elements contained in a common ancestor element.

When applied to the text selection, the **Edit|Copy** menu command copies all the characters and nodes in the specified range to the system clipboard. It is then possible to paste these characters and nodes in any other application including XXE itself.

Select the characters displayed in bold font from the first **p**, copy them to the clipboard using **Edit|Copy**, click on the **title** and paste the copied characters using the **Edit|Paste**.



After copying the text selection, you'll notice that, near the *Clipboard Content* button, the status bar displays **#text**: a piece of text is stored in the clipboard.



Note that if you have not been precise when selecting text, the status bar may display **strong** (you have copied the whole **strong** element to the clipboard) or it may display **[2]** (you have copied 2 nodes to the clipboard: the **strong** element and a piece of text next to it).

Edit|Cut and **Edit|Delete** basically work the same: they delete all the characters and nodes in the range specified by text selection. The only difference is that **Edit|Cut** copies the characters and nodes in the specified range to the clipboard.

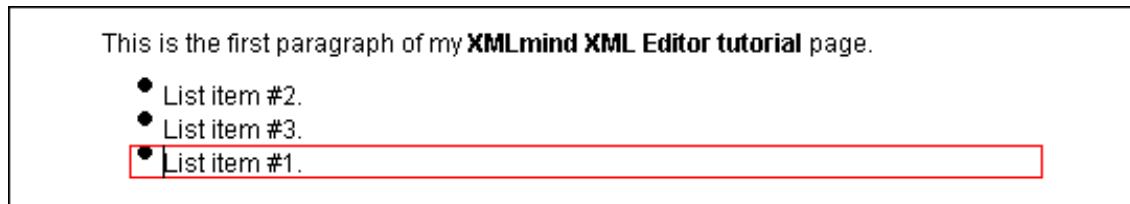
Of course, elements and nodes are deleted only if the DTD or schema constraining the document allows to do so. In this is not the case, selected characters are removed from such elements and nodes and that's it.

2.6.2 Copy, cut, paste, delete applied to the node selection

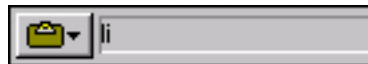
The **Copy**, **Cut**, **Paste**, **Delete** commands can be applied to a node range or to the implicitly selected element.

Two more **Paste** commands are available: **Paste before** and **Paste after**. These commands can only be applied to a single explicitly selected node or to the implicitly selected element.

Select the first **li** (implicit selection is fine for that). Use the **Edit|Cut** to move it to the clipboard. Select the last **li**. Use the **Edit|Paste After** to paste the **li** stored in the clipboard after the last **li**.



After copying the first **li**, you'll notice that the status bar displays **li**: a list item element is stored in the clipboard.



Note that when the clipboard contains a node range, the status bar displays the number of nodes of the range. For example, copy all list items to the clipboard and you'll see:



Pitfall: Once you have copied something to the clipboard from an *external* application, you have to change the “editing context” of XXE if you want to make XXE see what has been copied to the clipboard. For example, click inside another element then click again where you want to paste clipboard content.

2.7 Splitting and joining elements

2.7.1 Simple Split and Join

Click in the middle of the **strong** contained in the first **p**. *Type Enter*. The **p** is split in two parts, each part being a **p**, as expected in any word processor.

This is the first paragraph of my XMLmind
XML Editor tutorial page.

Now *type Backspace* at the beginning of the second part. The two parts are joined to form our original single paragraph.

Typing Del at the end of the first part would have given the same result.

This is also a handy method for inserting elements. Type *Enter* at the end of the **p**.

This is the first paragraph of my XMLmind XML Editor tutorial page.
■

Type some text in the newly created paragraph. Type *Enter* at the beginning of newly created **p**. This creates another **p** before it.

This is the first paragraph of my XMLmind XML Editor tutorial page.
■
New paragraph here.

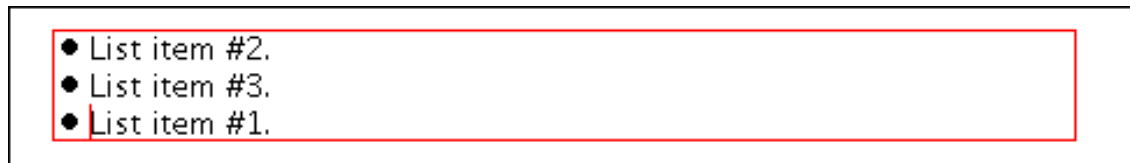
Use **Edit|Undo** three times to undo the creation of last two paragraphs.

2.7.2 Split and Join generalized

The Split and Join commands bound to *Enter*, *Backspace* or *Del* keystrokes are very handy but they can only be applied to paragraphs.

What if we want to split the list in two parts in order to insert a paragraph before the second part?

Move the caret at the beginning of the last list item and explicitly select the whole list using the node path bar.



Now execute **Edit|Split**. This command splits the explicitly selected element at caret position, which gives us two adjacent lists.



The generalized form of Join is the inverse command of Split. Command Join joins explicitly selected element to its *preceding* sibling, an element of same type.

Note that the generalized form of Split and Join are the only commands that cannot be applied to the implicitly selected element.

Explicitly select the second **ul** and execute **Edit|Join**. The two adjacent lists are now joined to form our original single list.

Undo last Join because we really want to split the list in two parts.

- Add a **p** before the second **ul**, type some text in it.
- Copy text "List item #1" (as usual, by dragging the mouse and by pressing Ctrl-C) and paste it (by pressing Ctrl-V) in the empty last **li** of first **ul**.
- Add another **li** before the only **li** of second **ul**, for example by clicking in the **li**, using **Edit|Copy** and then using **Edit|Paste Before**.
- Change all the text of the two list items of the second **ul**, for example by selecting "List item #1" as usual and by typing a few words.

Our XHTML document now looks like this.

The title of my XMLmind XML Editor tutorial page

This is the title of my first heading

This is the first paragraph of my **XMLmind XML Editor tutorial** page.

- List item #2.
- List item #3.
- List item #1.

A paragraph before second list.

- Another list item.
- Result of splitting List item #1.

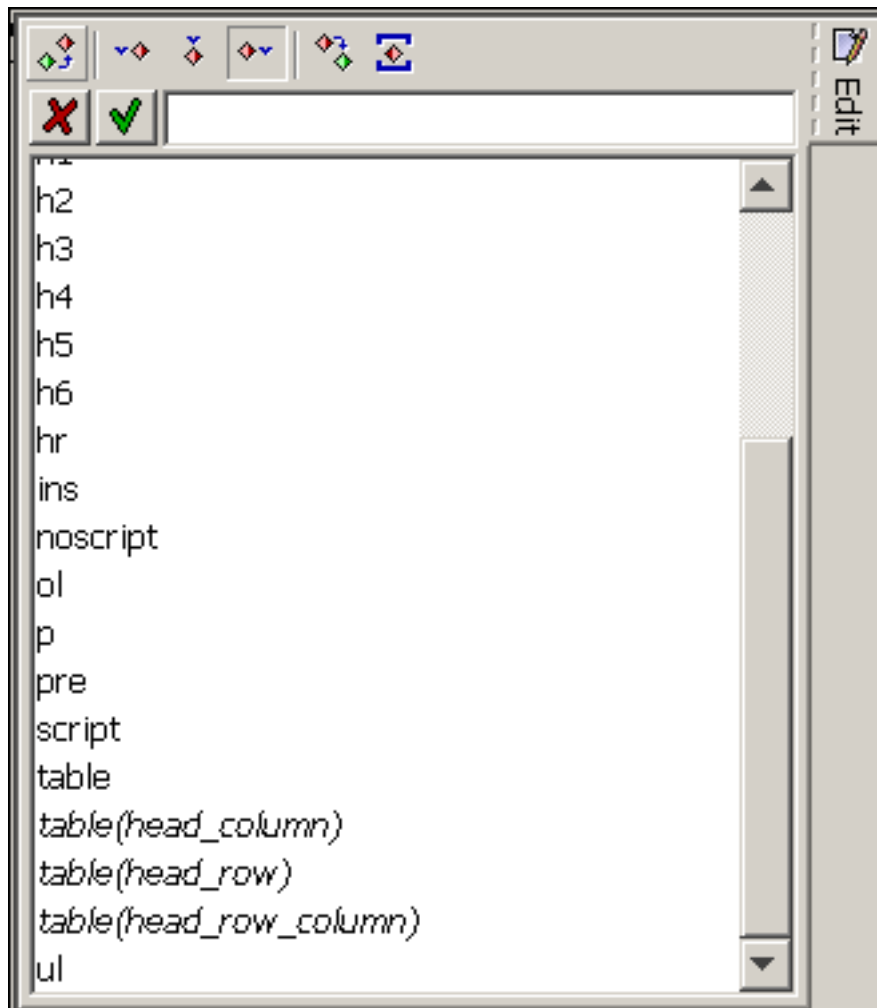
Line 1
Line 2↵

2.8 Replacing elements

The **Edit|Replace** command is equivalent to deleting the node selection or implicitly selected element and inserting a new element or **#text** which replaces the deleted nodes.

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

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



table(head_column), ..., **table(head_row_column)** are preconfigured *table templates*. Most of the time, you'll choose one of these but for this exercise, we'll choose **table** which is the simplest table possible in XHTML.

The table is created with a single **tr** (row) containing a single **th** (a "heading" cell where text is displayed using a bold font). This is often what you want for the skeleton of **table** but not always.

The title of my XMLmind XML Editor tutorial page

This is the title of my first heading

This is the first paragraph of my **XMLmind XML Editor tutorial** page.

- List item #2.
- List item #3.
- List item #1.

A paragraph before second list.

- Another list item.
- Result of splitting List item #1.

Line 1

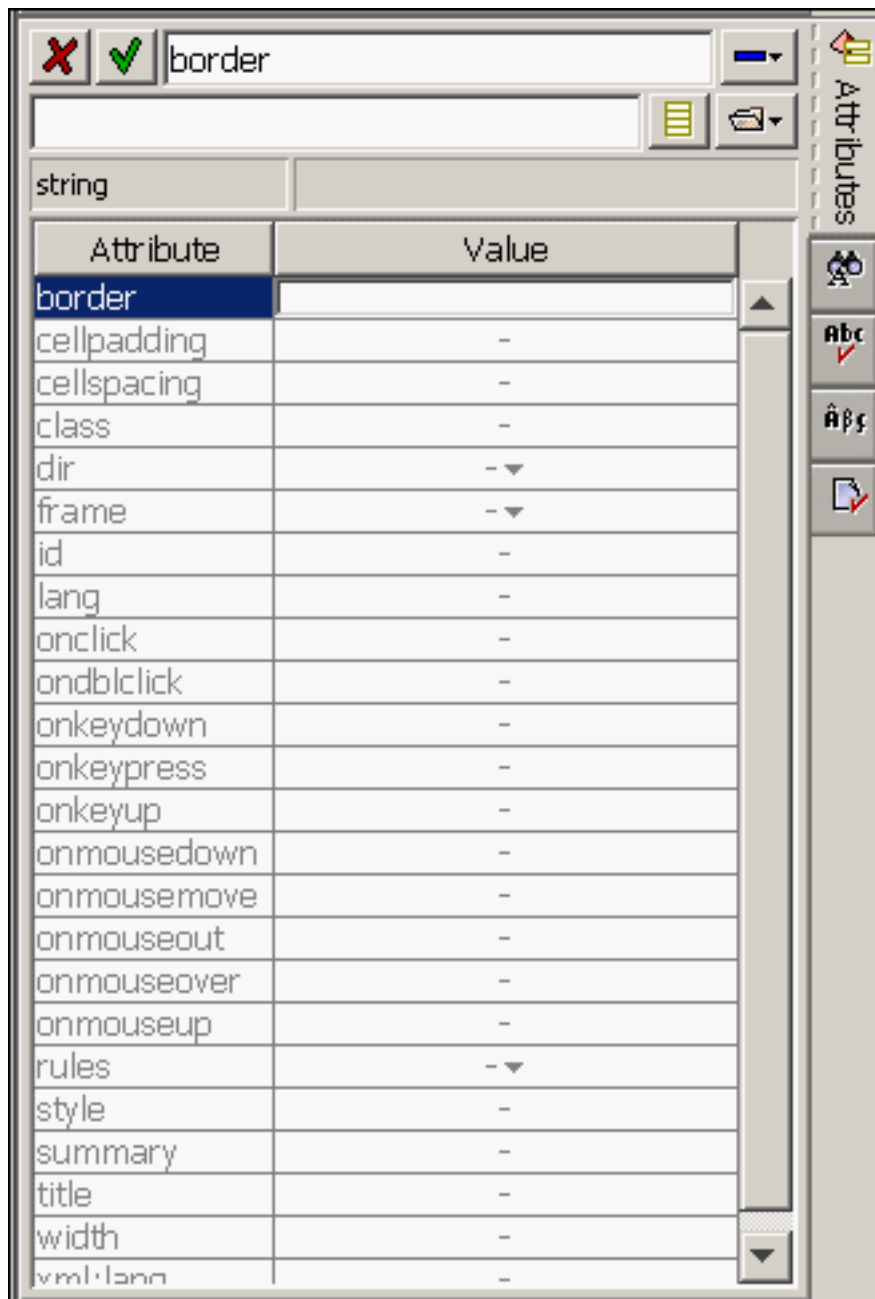
Line 2

--	--

This very simple table has no borders, even if the footprint of a cell is displayed using a very light gray. This would make the screenshots of this tutorial hard to read. Therefore we'll immediately add a border to this table.

Select the **table** (using the node path bar) if it is not already the case. The **Attributes** tool should be already displayed, otherwise click on the corresponding tab.

First row of the attribute table is for attribute **border**. Click inside the attribute value cell at the right of the attribute name cell containing **border**, type **1** and press **Enter**.



Now the table has a nice black border. (We'll learn how to use the **Attributes** tool later in this tutorial.)

Exercise:

1. Select the **th** contained in the **tr** (implicit selection is fine for that). Using **Edit|Replace**, replace it with a **td** (a plain cell).
2. Select this **td**. Use **Edit|Insert After** to add another **td** after it.
3. Using the node bar path, select the **tr** containing the two cells. Use **Edit|Copy** to copy it to clipboard.
4. Use **Edit|Paste After** to paste a copy of the **tr** after currently selected **tr**.
5. Type some text in each cell.

The title of my XMLmind XML Editor tutorial page

This is the title of my first heading

This is the first paragraph of my **XMLmind XML Editor tutorial** page.

- List item #2.
- List item #3.
- List item #1.

A paragraph before second list.

- Another list item.
- Result of splitting List item #1.

Line 1
Line 2

Cell 1,1	Cell 1,2
Cell 2,1	Cell 2,2

2.9 Converting elements

2.9.1 Convert applied to the text selection

Select a non-bold word in the first **p**.

This is the first **paragraph** of my **XMLmind XML Editor tutorial** page.

Use the **Edit|Convert** to convert it to **em** (emphasis).

This is the first *paragraph* of my **XMLmind XML Editor tutorial** page.

Remember that text selection is used not only to specify a range of characters but also a range of nodes and child elements contained in a common ancestor element. This feature is very useful when doing a conversion. For example: select text from word "paragraph" to word "XMLmind".

This is the first *paragraph* of my **XMLmind** XML Editor tutorial page.

Use the **Edit|Convert** to convert it to **button**.

This is the first **paragraph** of my **XMLmind** XML Editor tutorial page.

Use **Edit|Undo** to undo the last conversion.

2.9.2 Convert applied to the node selection

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

Select the **#text** node contained in the first cell of the table. Note that explicit node selection is needed to do that: implicit element selection selects the **td** not the **#text** inside it.

Cell 1,1	Cell 1,2
Cell 2,1	Cell 2,2

Used **Edit|Convert** to convert it to **em**.

Cell 1,1	Cell 1,2
Cell 2,1	Cell 2,2

Unlike **Edit|Replace** which creates an *empty* new element, **Edit|Convert** transfers the content of the selection to the new element which is the result of the conversion.

More precisely, in the case of the node selection:

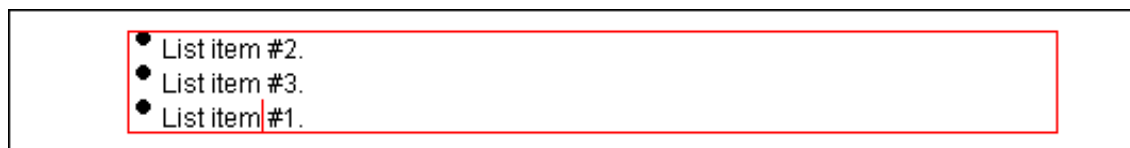
- When several nodes or a single non-element node are selected, all these nodes are given a new parent element which is the result of the conversion.

We have already seen two examples of this behavior. First one is when we “wrapped” a **p** and a **ul** into a **blockquote**. Second one is just above.

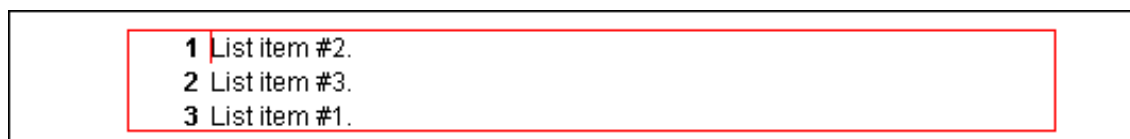
- When a single element is selected, all its children (but not its attributes) are transferred to the result of the conversion.

What follows is an example of this second behavior. Here we want to “morph” single selected element which is an **ul** to an **ol**.

Select the first **ul**.



Used **Edit|Convert** to convert it to an **ol** (ordered list).



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**.

2.10 Editing element attributes

Use **Tools|Edit Attribute** (keyboard shortcut Ctrl-E) to edit the attributes of the selected element. This action displays and gives the keyboard focus to the **Attributes** tool. Alternatively, if this pane is already displayed, you can click inside it and use it right away.

Attribute	Value
axis	-
char	-
charoff	-
class	-
colspan	1
dir	- ▼
headers	-
id	-
lang	-
onclick	-
ondblclick	-
onkeydown	-
onkeypress	-
onkeyup	-
onmousedown	-
onmousemove	-
onmouseout	-
onmouseover	-
onmouseup	-
rowspan	
scope	- ▼
style	-
title	-
valign	- ▼

*The **Attributes** tool with the **rowspan** attribute of a **td** being edited*

There are two methods for adding or changing the attributes of the (explicitly or implicitly) selected element:

1. Using the attribute form (the upper side of the **Attributes** tool). This should be the method of choice for persons who prefer to use the keyboard.

- Using the attribute table (the lower side of the **Attributes** tool). This should be the method of choice for persons who prefer to use the mouse.

The “minus” button of the attribute form can be used to remove an attribute. Removing an attribute directly from the attribute table is possible too: simply right-click on the attribute row and use the displayed popup menu.

The content of the attribute table can be described as follows:

- All attributes set for the selected element are displayed by the table.
- All possible attributes for the selected element, *even those not set*, are also displayed by the table. Attributes which have not been set are displayed in gray. Attributes which have been set are displayed in black.
- Attributes are listed sorted in alphabetical order.
- The names of required attributes are displayed using a bold font.
- The names of fixed attributes are displayed using an italic font.

Exercise:

- Select second **td** in the **table**. Set its **align** attribute to **center**.
- Set its **rowspan** attribute to 2.
- Set its **valign** attribute to **middle**.
- Add an extra **td** after it to make the **table** look more balanced.

Cell 1,1	Cell 1,2	Extra cell that makes table look more balanced.
Cell 2,1		Cell 2,2

2.10.1 Required attributes in newly created elements

It is important to remember that, by default, XXE automatically gives a placeholder value ("???") to required attributes of newly created elements. This means that you have to replace this placeholder value by the actual one as soon as the element has been created.

Add a **p** after the **table** using **Edit|Insert After** and insert an **img** in it using **Edit|Insert**.



The **img** has two required attributes **src** and **alt**. XXE has set those attributes to string "???". Use the **Attributes** tool to give these attributes an actual value.

The title of my XMLmind XML Editor tutorial page

This is the title of my first heading

This is the first *paragraph* of my **XMLmind XML Editor tutorial** page.

1. List item #2.
2. List item #3.
3. List item #1.


A paragraph before second list.

- Another list item.
- Result of splitting List item #1.

Line 1

Line 2

Cell 1,1	Cell 1,2	Extra cell that makes table look more balanced.
Cell 2,1		Cell 2,2



For this **img**, we used `XXE_install_dir/doc/user/tutorial/xxe.gif` and "XMLmind logo" for **alt** (directory `XXE_install_dir/doc/user/tutorial/`, generally `C:\Program Files\XMLmind_XML_Editor\doc\user\tutorial\`, contains all the files used in this tutorial).

The useless **#text** node has been removed by explicitly selecting it and using **Edit|Delete**.

2.11 Checking document validity

You cannot check the validity of a document without an associated DTD, W3C XML Schema or RELAX NG schema. When editing a document not constrained by a grammar, 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 constrained by a grammar.

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

Unless you use **Edit|Force Deletion**, XXE will never allow editing commands that would make the document *structurally* invalid (a document where some elements have invalid child elements or attribute names).

Therefore explicitly checking document validity is rarely needed. You may have to use the **Tools|Check Validity** command when:

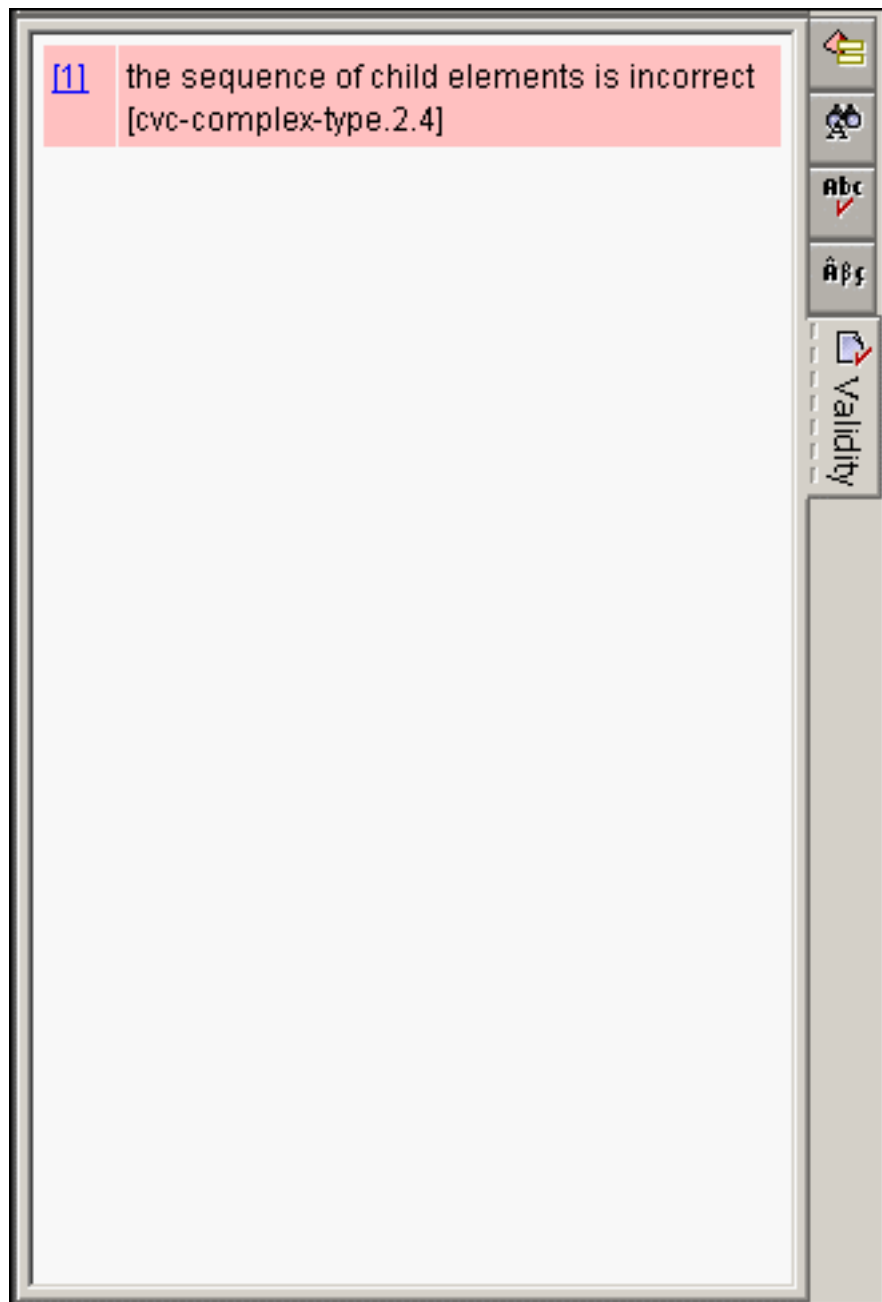
1. You have loaded an invalid document and you are fixing it. After each editing command, you want to know if it is fixed now.
2. XXE creates elements where the *value* of required attributes, if any, is invalid: unless configured differently, the required attributes are given "???" as a placeholder value.

After explicitly or implicitly checking document validity, its validity status is displayed at the left of the status bar.



After forcing the deletion of all the items of a list, "Untitled.html" is structurally invalid.

The **Validity** tool “tab” displays validity error messages if any.



Validity error message displayed after forcing the deletion of all the items of a list.

The color of the message reflects the severity of the error. Clicking on the number of an error message selects the element where the validity error was found.

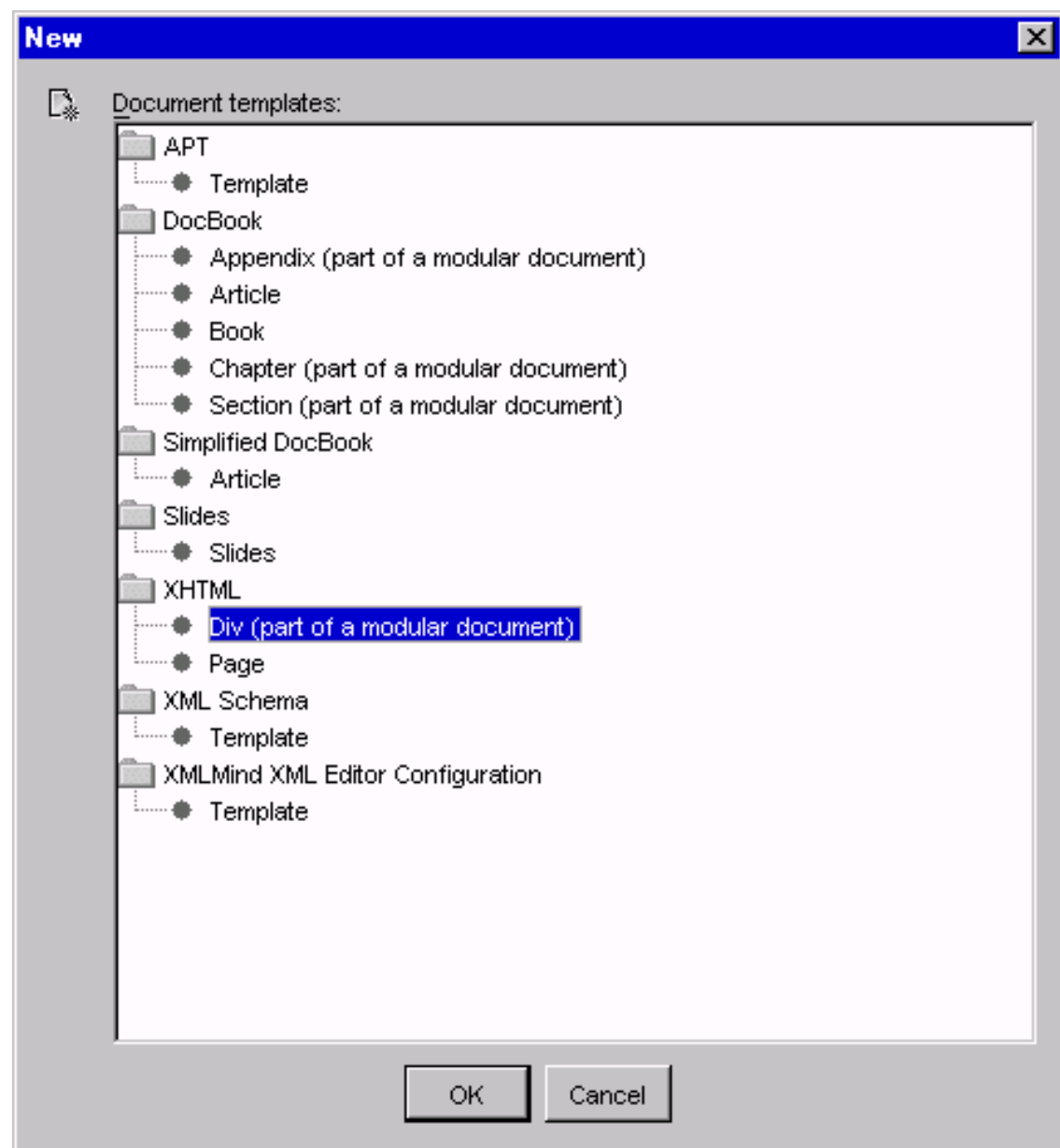
3 Tutorial: creating a modular document

A *modular document* is a document which not only has its own content but which also includes elements contained in other documents.

Let's say you want to add the same copyright information at the bottom of all the XHTML pages you create. The first step is to author the copyright information in a separate document we'll call `Copyright.html`. The second step is to include this copyright information at the bottom of the XHTML page created during this tutorial.

3.1 Creating the Copyright.html document

Use **File|New** and this time choose **XHTML|Div (part of a modular document)** as the document template.

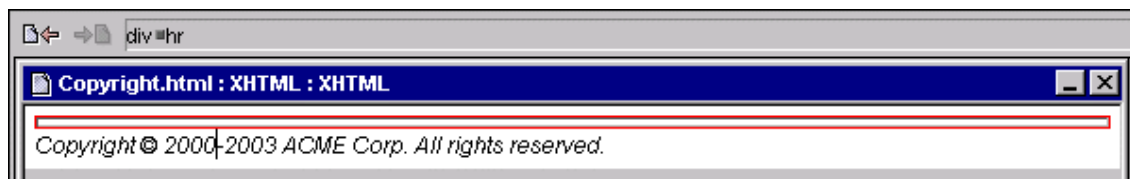


This template is a **div** (a generic container) containing a **p**. Using **Edit|Replace**, replace the **p** by an **address** element and then type the copyright information. (The copyright sign © has been inserted using the **Characters** tool.)



Using **Edit|Insert Before**, insert a **hr** (horizontal rule) before the **address** to visually separate the copyright information from the body of the XHTML page.

Using **File|Save As** (or **File|Save** which behaves like **File|Save As** for a new document), save this document as `Copyright.html` in a place where it can be shared by all the XHTML pages you'll create.



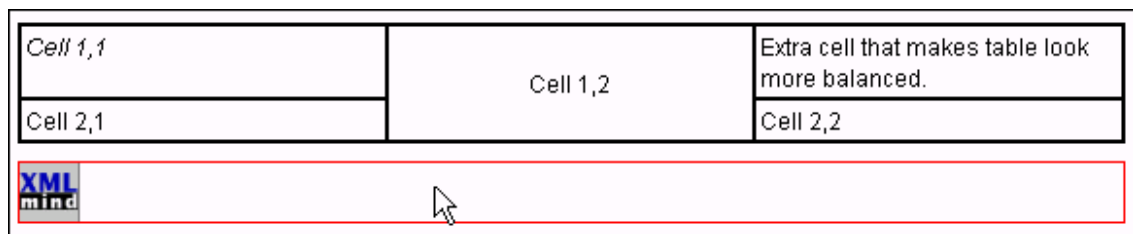
3.2 Inserting a reference to Copyright.html into the XHTML page

The procedure to do this is very simple. It is similar to copying an element in a document and pasting it in another document:

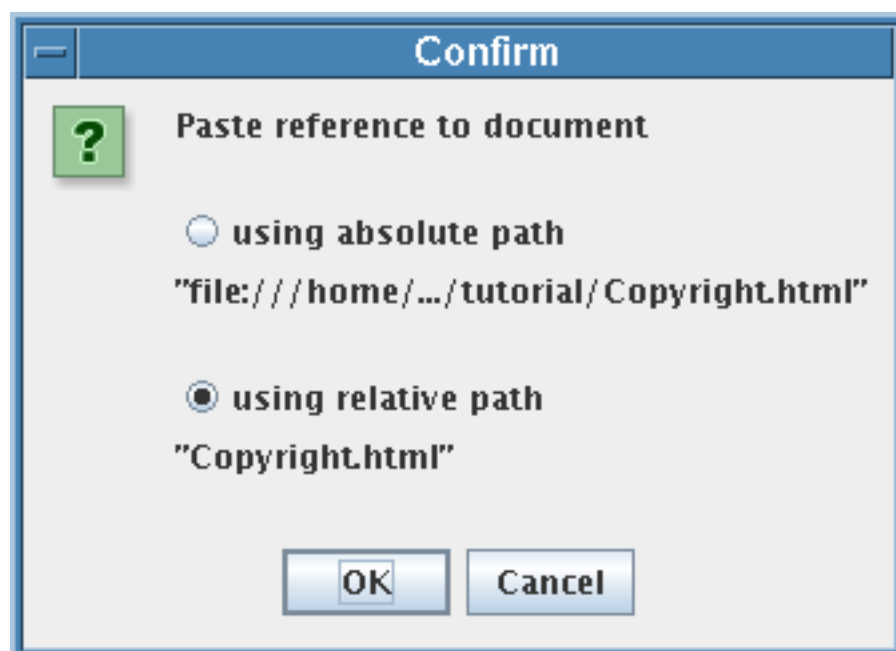
1. Open in XXE the document containing the element you want to reference. If the document containing the element you want to reference is already opened in XXE, simply display its window.
2. Explicitly select the element you want to reference.
3. Use menu bar menu **Edit** (not popup menu **Edit**), select sub-menu **Document Reference** and choose entry **Copy as Reference** (shortcut **Ctrl+Shift-C**). Unlike the usual **Copy** command which copies XML data to the clipboard, this special command copies to the clipboard a *reference* to an element (that is, a *pointer* to the element).
4. Now switch to the window containing the modular document.
5. Select the element where you want to insert the reference.
6. Use one of the standard **Paste Before**, **Paste** or **Paste After** commands to insert the reference into the modular document. As always, the **Paste Before**, **Paste** or **Paste After** commands are enabled only if the DTD or schema constraining the modular document allows it.

Now let's apply this procedure to our example:

1. The window containing `Copyright.html`, the document we want to reference, is already displayed. There is nothing to do in step #1.
2. We want to reference the whole `Copyright.html`, therefore we need to select its root element. Click on word **div** in the node path bar to do this.
3. Use **Ctrl+Shift-C** to copy to the clipboard a reference to the **div**. Notice that at the bottom of XXE window, near the "View Clipboard Content" icon, word **div** is displayed using a dimmed color. This means that the clipboard contains a reference rather than ordinary data.
4. Switch to the window containing our XHTML page. XXE being a multi-document editor, the XHTML page created during this tutorial is still loaded in the editor but it is hidden behind `Copyright.html`. Use **Window|Untitled.html : XHTML : XHTML** to switch back to our XHTML page.
5. Select last **p** at the bottom of the page, for example by simply clicking at the right of the XMLmind logo. (There are many quick ways to select nodes using the keyboard or the mouse: they are all explained in next chapter of this user's guide: being productive with XXE.)

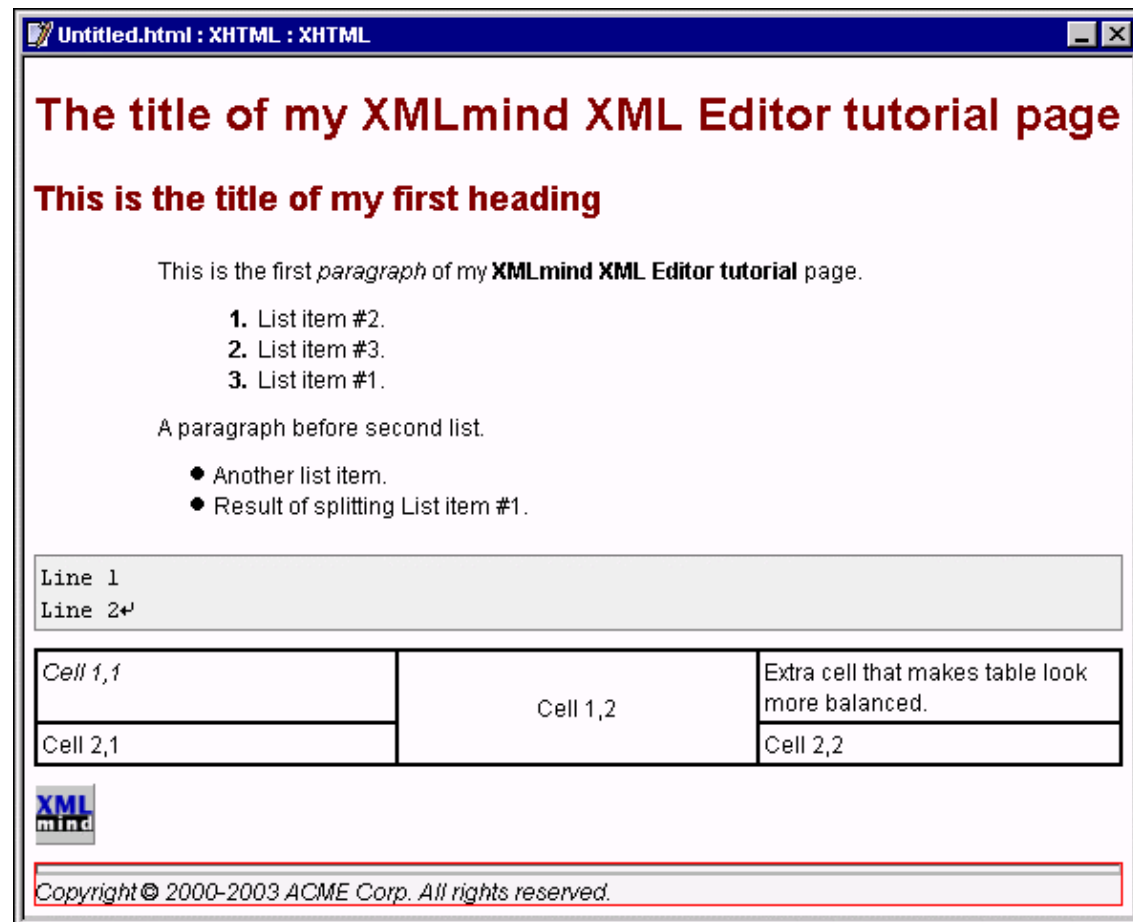


6. Use command **Paste After** (shortcut **Ctrl-W**) to paste the reference to the **div** after selected **p**. A dialog box is displayed to mainly inform you that you are pasting a reference to an external element and not ordinary data.



Click on OK.

The XHTML page now looks like this.



Notice that the copyright information inserted at the bottom of the page is displayed with a light gray background. This is used to indicate that this part of the document has been included from an other document and that, consequently, *it cannot be edited in this window*.

Click anywhere inside the copyright information and try to type some text: no characters are inserted. (You can still delete, cut or replace the whole copyright information: it is the **div** which is not editable not its parent **body**.)

Also notice that the node path bar displays non-editable nodes using a dimmed color.



At the left of the node path bar, the button with a right arrow switches to the window of the included document. Try it and you'll be able to edit `Copyright.html`.

Once having `Copyright.html` in front of you, use the button at the left of the node path bar showing a left arrow to go back to the XHTML page created during this tutorial.

3.3 Extensive use of the "Copy as Reference" command

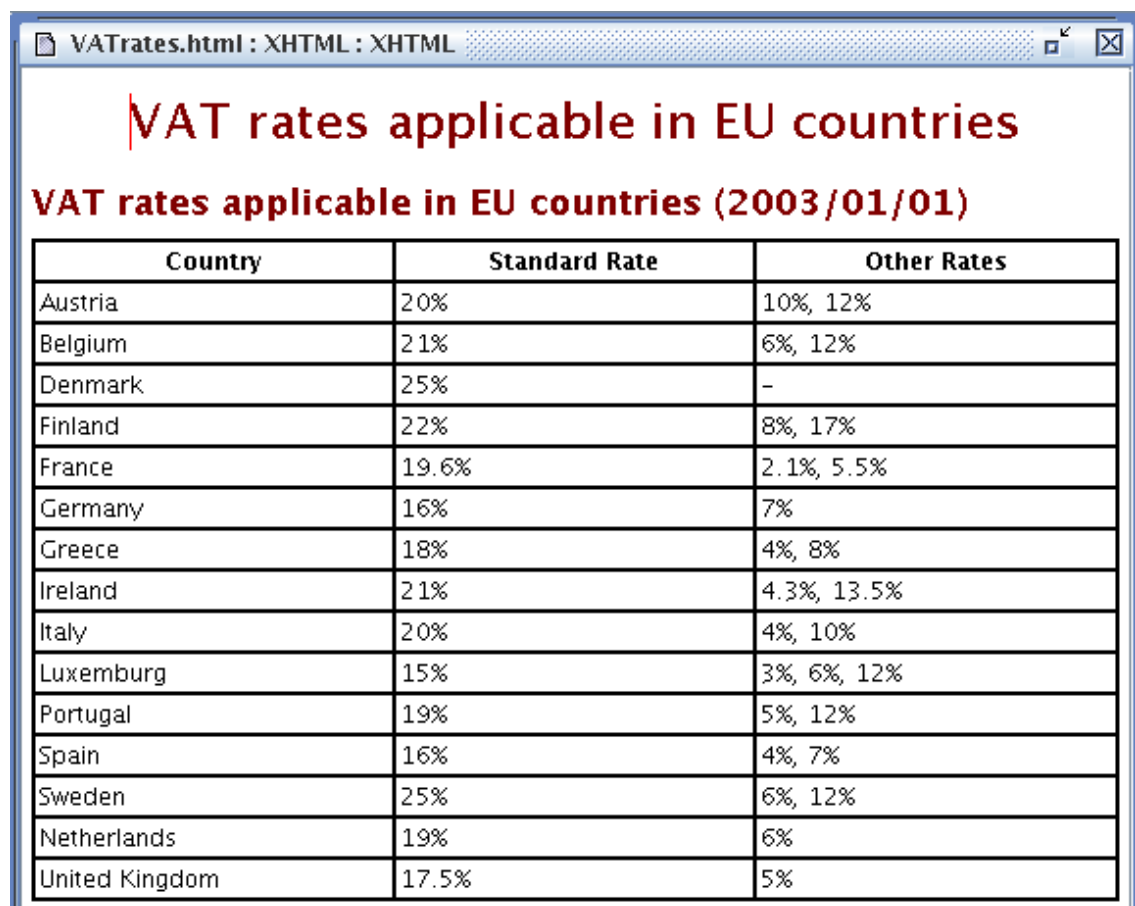
Let's suppose you need to write an article about taxes but you don't want to type directly in your document the values of the different VAT rates used in European countries, because you know that these VAT rates are about to change. Let's say that you already have an XML document detailing these VAT rates.

(In this tutorial, creating the VAT rates document from scratch would be tedious that's why you'll find VATrates.html in `XXE_install_dir/doc/user/tutorial/`. You'll also find in this directory: Untitled.html, the XHTML page we are trying to create, Copyright.html and xxe.gif. That is, all the files used in this tutorial.)

Add a new **p** at the end of our XHTML page and type in it "The VAT rate of France is higher than the VAT rate of Germany".

Now we need to use VATrates.html to insert the numerical values of the VAT rates in our new paragraph.

1. Open VATrates.html in XXE.



VAT rates applicable in EU countries

VAT rates applicable in EU countries (2003/01/01)

Country	Standard Rate	Other Rates
Austria	20%	10%, 12%
Belgium	21%	6%, 12%
Denmark	25%	-
Finland	22%	8%, 17%
France	19.6%	2.1%, 5.5%
Germany	16%	7%
Greece	18%	4%, 8%
Ireland	21%	4.3%, 13.5%
Italy	20%	4%, 10%
Luxembourg	15%	3%, 6%, 12%
Portugal	19%	5%, 12%
Spain	16%	4%, 7%
Sweden	25%	6%, 12%
Netherlands	19%	6%
United Kingdom	17.5%	5%

2. Click inside the cell which contains the VAT rate of France to implicitly select it, then use **Ctrl+Shift-C** to copy it as a reference.
3. Switch to the window displaying our XHTML page: Untitled.html.
4. Move the caret after "The VAT rate of France".

5. Paste the copied reference using **Ctrl-V**.

Do the same for the VAT rate of Germany.

The XHTML page now looks like this.

The title of my XMLmind XML Editor tutorial page

This is the title of my first heading

This is the first *paragraph* of my XMLmind XML Editor tutorial page.


1. List item #2.
2. List item #3.
3. List item #1.

A paragraph before second list.

- Another list item.
- Result of splitting List item #1.

Line 1
 Line 2

Cell 1,1	Cell 1,2	Extra cell that makes table look more balanced.
Cell 2,1		Cell 2,2



The VAT rate of France 19.6% is higher than the VAT rate of Germany 16%.

Copyright © 2000–2003 ACME Corp. All rights reserved.

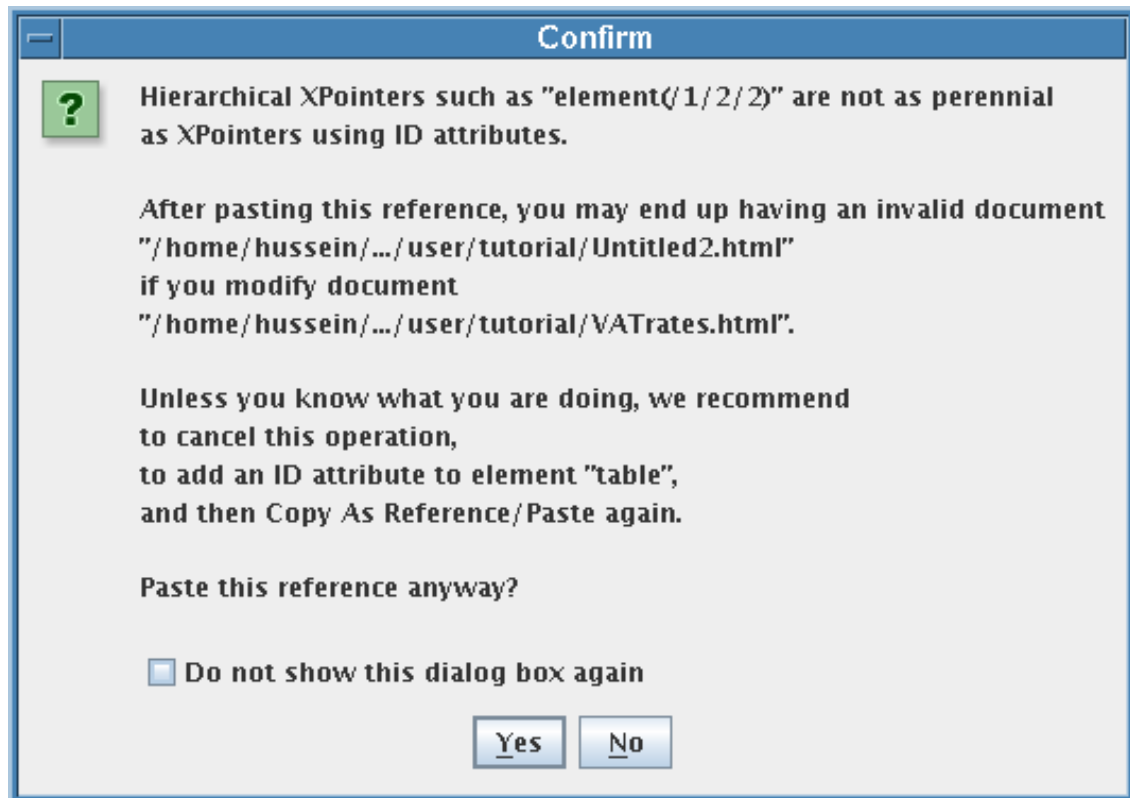
This has worked smoothly because:

- a. Each table cell of VATrates.html contains a single **span** element rather than plain text. A **span** element can be inserted almost anywhere in an XHTML document.
- b. Each **span** of second column is identified using an **id** attribute: austria_vat, belgium_vat, etc.

Now let's try to insert the whole VAT table in our XHTML page:

1. Switch to the window of VATrates.html.
2. Select the whole table, for example by clicking on word **table** in the node path bar.
3. Use **Ctrl+Shift-C** to copy the table as a reference.
4. Switch to the window displaying our XHTML page.
5. Select last added **p**.
6. Paste copied reference to the VAT table after selected **p**.

After the usual confirmation dialog box, this verbose dialog box is displayed.



Click on No. What does this mean?

Command **Copy as Reference** allows to copy as a reference *any* element, including those not having an ID.

Unless the copied reference is a reference to the root element of a document, this means that the copied reference is based on the position of the element as a child of its parent element, on the position of the parent element as a child of the grand-parent element and so on.

This is what we call a *hierarchical XPointer*. In the above example, it is `"VATrates.html#element(/1/2/2)"`.

Pasting a reference based on a hierarchical XPointer is *not safe* because if you modify the referenced document, your referencing document may point to an element other than the one for which a reference has been originally pasted.

This is why XXE will warn you each time you'll try to paste a reference based on a hierarchical XPointer.

Note that you'll never get the above dialog box when you'll paste a reference to the root element of a document (example: `"Copyright.html"`) or when you'll paste a reference which exclusively uses an ID to locate the element (example: `"VATrates.html#france_vat"`) because these references are considered to be safe.

Now type this sentence at the end of last added **p**: "Note that the VAT rate of France is lower than the VAT rate of Italy." and insert the value of the VAT rate of Italy as explained above.

Inserting the VAT rate of France in second sentence is easier because we have already pasted it in our XHTML page. In such case, there is no need to switch to the window displaying `VATrates.html`:

1. Select the **span** previously pasted in our XHTML page which contains the VAT rate of France. For example, implicitly select it by clicking inside it.

2. **Ctrl+Shift-C** to copy it as a reference.
3. Click in second sentence after “Note that the VAT rate of France”.
4. Paste the copied reference using **Ctrl-V**.

The XHTML page finally looks like this.

The title of my XMLmind XML Editor tutorial page

This is the title of my first heading


This is the first *paragraph* of my XMLmind XML Editor tutorial page.

1. List item #2.
2. List item #3.
3. List item #1.

A paragraph before second list.

- Another list item.
- Result of splitting List item #1.

Line 1		
Line 2		
Cell 1,1	Cell 1,2	Extra cell that makes table look more balanced.
Cell 2,1		Cell 2,2



The VAT rate of France 19.6% is higher than the VAT rate of Germany 16%. Note that the VAT rate of France 19.6% is lower than the VAT rate of Italy 20%.

Copyright © 2000–2003 ACME Corp. All rights reserved.

If you save this XHTML page or if you click on the **Check Validity** icon found at the bottom left of XEX window, you'll find that XEX pretends that the XHTML page is invalid: ID "france_vat" already defined (in fact, the same element is included several times from "file:/opt/xxe/doc/user/tutorial/cvc-id.2]).

You can safely ignore this warning. It is reported when a reference to an external element has been pasted several times in the same modular document. In the case of our example, it is the reference to the external **span** element containing the VAT rate of France which has been inserted two times.

4 Being productive with XXE

Tips:

- *Do not use the tree view.* You can do everything efficiently in the styled view. If this is not the case then you have missed something in the tutorial.
(The tree view has been created to be able to open and edit documents for which a style sheet has not yet been written.)
- Use the *Insert* key to insert a **#text** node after explicitly selected node or implicitly selected element. This is often handy in the following situation: you have typed some text in a paragraph then inserted a **strong** for which you have typed text.



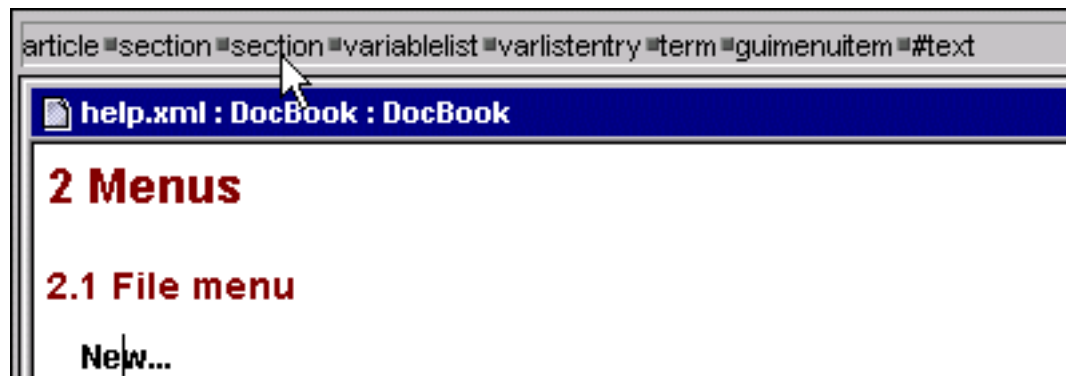
Then how to quickly continue typing plain text after the **strong**? The answer is: use the Insert key of your keypad.



Also note that *Shift-Insert* inserts a **#text** node *before* explicitly selected node or implicitly selected element.

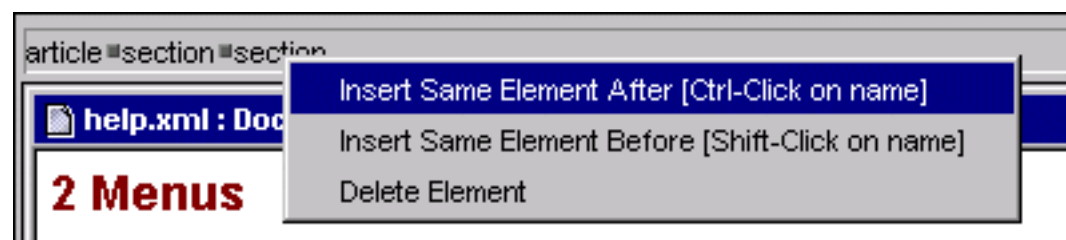
(On the Mac, there is no **Insert** key, so use the **F1** function key instead.)

- Use *Ctrl-Insert* to insert after explicitly or implicitly selected element, an element of the same type.
Example: you want to quickly add a section after current section. Select the section using the node path bar and type Ctrl-Insert.
Also note that *Ctrl+Shift-Insert* inserts *before* explicitly or implicitly selected element, an element of the same type.
- Even quicker than previous tip, *Ctrl-click in the node path bar* on the name of the element you want to duplicate.

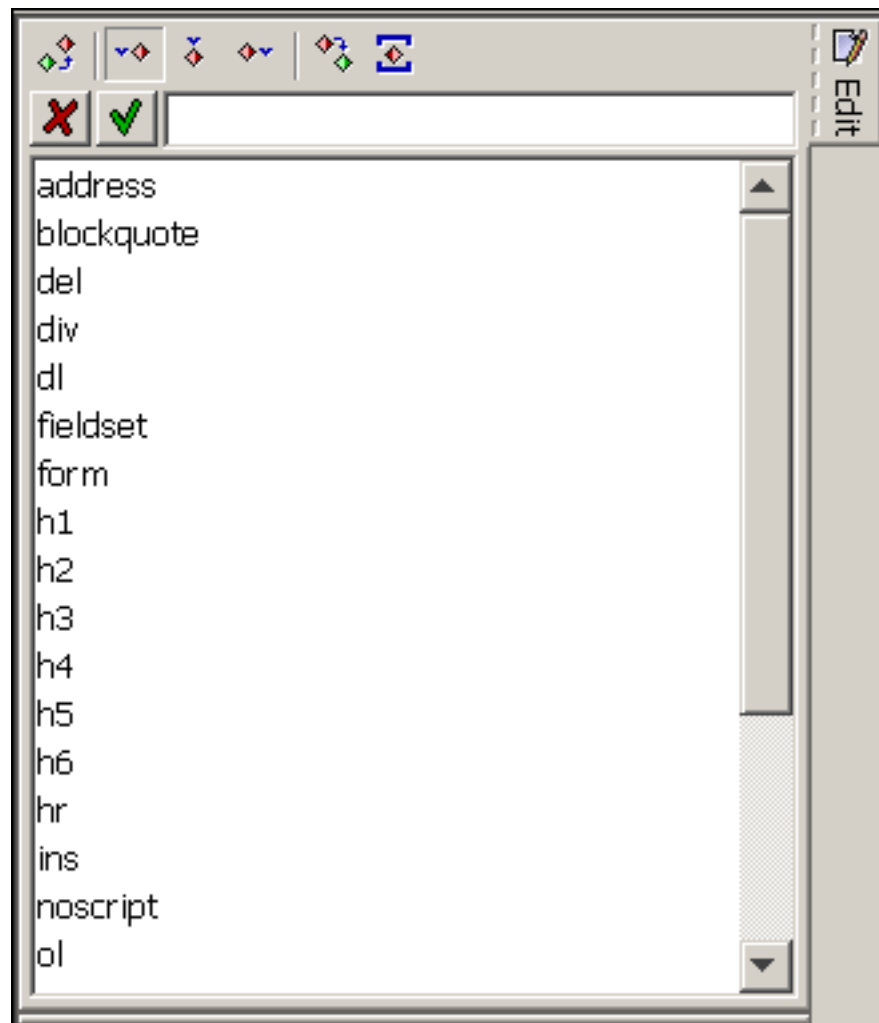


Also note that if you *Shift-click* in the node path bar on the name of the element, this selects this element and then creates a new element of same type *before* this element.

Remembering these two tips is easy, the node path bar has a contextual menu which is displayed when you click using the right mouse button.



- Most XXE configurations (XHTML, DocBook, Slides, etc) bind keystrokes *Ctrl-Enter* and *Ctrl+Shift-Enter* to the following actions:
 - Ctrl-Enter pressed anywhere inside a paragraph or a list item (i.e. any commonly used, repeatable, element) inserts a new paragraph or a list item after it.
 - Ctrl+Shift-Enter pressed anywhere inside a paragraph or a list item inserts a new paragraph or a list item before it.
- When choosing an element using this pane, type the element name (or "**(text)**" to specify a text node) in the text field instead of choosing it in the element list.



This text field supports *auto-completion*. User preferences related to auto-completion can be changed using the **Options** dialog box, **Edit** section.

- When adding an attribute to an element or when changing the value of an attribute, use the attribute form rather than the attribute table:

Attribute	Value
border	
cellpadding	-
cellspacing	-
class	-
dir	-▼
frame	-▼
id	-
lang	-
onclick	-
ondblclick	-
onkeydown	-
onkeypress	-
onkeyup	-
onmousedown	-
onmousemove	-
onmouseout	-
onmouseover	-
onmouseup	-
rules	-▼
style	-
summary	-
title	-
width	-
vml:lang	-

1. Type the name of the attribute in the name field (first field of the form).
2. Press **Enter** to move to the value field (second field of the form).
3. Type the value of the attribute in the value field.
4. Press **Enter** to commit the change and to give the keyboard focus back to the document view.

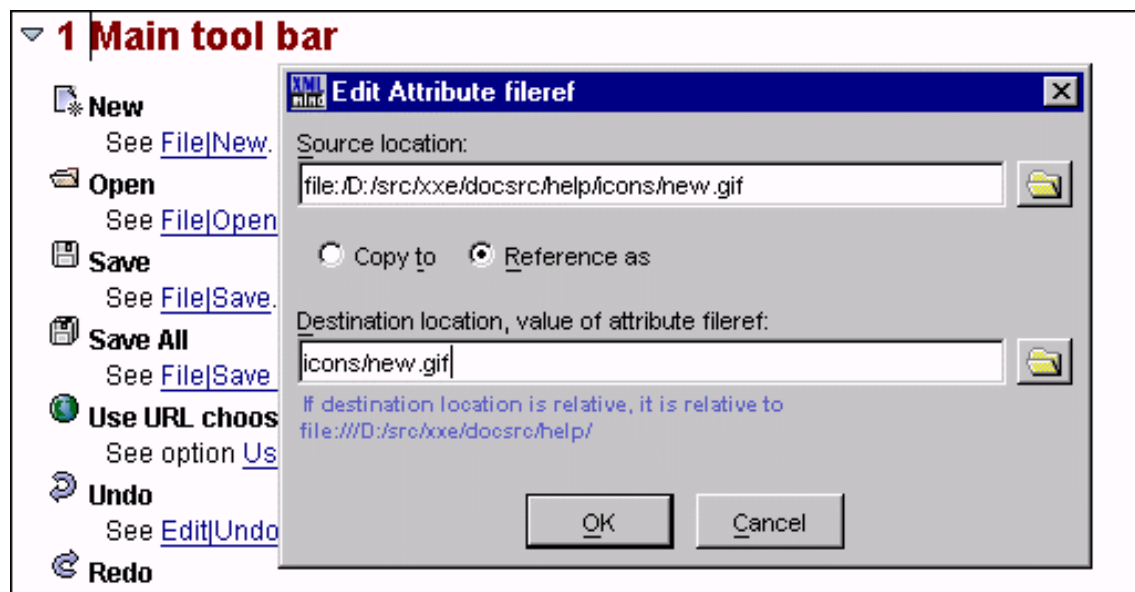
Both the name and value fields support *auto-completion*. However auto-completion in the value field only works for attributes having the following types: any enumerated type, ID, IDREF, IDREFS. This auto-completion feature can be configured using the **Options** dialog box, **Edit** section.

- When an image is displayed in a styled document view, the simplest way to change it is to drag and drop a file on it.

When a file is dropped on an image, a pre-filled, specialized dialog box is displayed to let the user specify exactly what he wants to do with the image file.

For example, if the image file is referenced in attribute `fileref` of element `imagedata` (like in screenshot below), the user is given the choice between

- copying the image to the document directory and then updating the reference to the image file in the attribute,
- OR just updating the reference to the image file in the attribute.



The same dialog box can be opened without having to drop a file on an image displayed in the document view: simply double-click on the image.

This functionality has been implemented mainly to allow users to upload images to the remote site when they edit documents located on a FTP or WebDAV server, but it is also handy when working on the local file system.

- Use the mouse to quickly copy and paste text:
 - Selecting text automatically “copies it as system selection” on platforms supporting system selection (X-Window) and automatically copies it to an internal clipboard on other platforms.
 - Clicking with mouse button #2 (middle button or mouse wheel) pastes the content of system selection on platforms supporting system selection and pastes the content of an internal clipboard on other platforms.

Note that unlike **Edit|Copy** which copies characters as well as nodes, selecting text this way just copies *characters* to the system selection.

This functionality is disabled by default. To enable it, please use the **Options** dialog box, **Edit** section and check "**Clicking with middle button pastes system selection**".

- Apply commands to a node range whenever possible.
Use keyboard to select a node range:

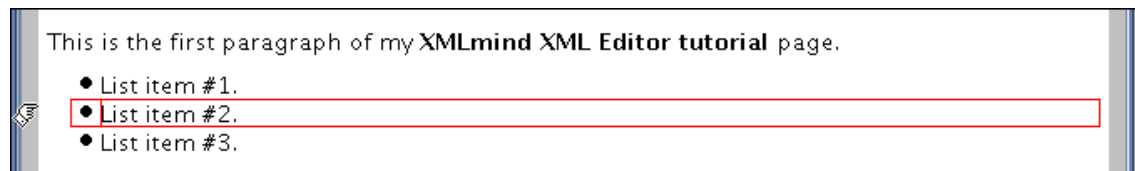
- *Esc Down Arrow* (that is: type *Esc*, then type *Down Arrow*) selects all child nodes of implicitly or explicitly selected element.
- *Esc Right Arrow* extends node selection to following sibling.
- *Esc Left Arrow* extends node selection to preceding sibling.

Note *Esc Right Arrow* (and *Esc Left Arrow*) will first select element containing caret if there is no explicit node selection, therefore typing *Esc Right Arrow* several times is often the quickest way to select a node range.

- The easiest way to select a paragraph or a table row is to use the option which adds *interactive “gray margins”* at the left and at the right of the document view.

By default these interactive margins are absent. To enable them, you need to use **Options|Options, Edit** section and check the **"Add interactive margins to styled views"** toggle. After doing this, you'll also need to reload any opened document.

Example, using interactive margins to select an **ul** in an XHTML document:



Move the mouse in the gray margin found at the left and at the right of the document view. Here you'll notice that the cursor changes its shape. Move the cursor in front of any list item and click once. Mouse clicks in the left or in the right margin selects the “block” (paragraph, row, row group, table) which is in front of the click. In this case, this selects the **li**. Click again without moving the mouse and this will select the parent of the **li**: the **ul**. Clicking again without moving the mouse would select the parent of the **ul**: the **body**, and so on. (Do not click several times too fast otherwise the editor will think you are double-clicking or triple-clicking and therefore, selecting elements that way would not work.)

Note: Because this way of selecting blocks is so easy to use, it tends to “cannibalize” the other ways of selecting nodes, especially the implicit element selection. Therefore we do not recommend turning on this option.

- Use the following *keyboard shortcuts*:

Command	Keyboard shortcut	Trick to remember the keyboard shortcut
Undo	Ctrl-Z	Standard shortcut
Redo	Ctrl-Y	Standard shortcut
Repeat	Ctrl-A	A like A gain
Cut	Ctrl-X	Standard shortcut
Copy	Ctrl-C	Standard shortcut
Paste	Ctrl-V	Standard shortcut
Paste Before	Ctrl-U	Ctrl-V means Paste and U is before V
Paste After	Ctrl-W	Ctrl-V means Paste and W is after V
Delete	Ctrl-K	K like K ill
Replace	Ctrl-R	R like R eplace
Insert	Ctrl-I	I like I nsert
Insert Before	Ctrl-H (Cmd-B on Mac)	Ctrl-I means Insert and H is before I
Insert After	Ctrl-J	Ctrl-I means Insert and J is after I
Convert	Ctrl-T	T like T ransform
Wrap	Ctrl-Shift-T	T like T ransform (variant of Convert)
Split	Esc Enter	A paragraph-specific form of Split is often bound to the Enter keystroke
Join	Esc Backspace	A paragraph-specific form of Join is often bound to the Backspace keystroke
Search	Ctrl-F	Standard shortcut F like F ind
Replace	Ctrl-M	M like M odify
Find Next	Ctrl-G	Standard shortcut

On Mac (where the use of a 2-button mouse with a scroll wheel is *very well supported and highly recommended*):

- Use the Command key instead of the Control key, except for Ctrl-Tab and Ctrl-Space.
 - Use Control button1 to emulate mouse button 3.
 - Use Option button1 to emulate mouse button2.
- Copy `XXE_install_dir/doc/configure/samples2/customize.xxe` to `XXE_user_preferences_dir/addon/` to be able to use the following keyboard shortcuts:

Keyboard shortcut	Command
Esc l (means type Esc then type l)	Converts selected text to lower case. If no text is selected, operates on word containing caret.
Esc u	Converts selected text to upper case. If no text is selected, operates on word containing caret.
Esc c (means type Esc then type c)	Changes the first letter of all words found in selected text to upper case. If no text is selected, operates on word containing caret.
Esc d	Deletes characters from caret to end of word.
Esc f, Esc b	Identical to "Ctrl-Right" and "Ctrl-Left" except that these bindings are more handy to use when mixed with "Esc d", "Esc c", "Esc u", etc.
Esc t	Swaps the character before the caret with the character after the caret. Useful if you are a bit dyslexic.
Esc !	Inserts before caret the output of an external command.
Esc x	Executes an XXE command by specifying its name and its parameter. Useful for commands which don't have a key binding.
Esc m	Starts recording a macro-command, if a macro-command is not already being recorded. Otherwise, stops recording the macro.
Esc M (means type Esc then type <i>Shift-m</i>)	Cancels the recording of a macro-command.
Esc p	Replays the recorded macro-command.
Esc s	Displays a simple "Find Text" dialog box which is especially useful when recording macro-commands.
) ,] or }	Highlights matching (, [or { . Beeps if matching character is not found.
Esc C (means type Esc then type <i>Shift-c</i>)	Inserts before caret a character by specifying its entity name (example: nbsp).
Esc /	Collapses nearest collapsible view if it is expanded and expands nearest collapsible view if it is collapsed.
Esc +	Expands nearest collapsible view and then, recursively expands all its collapsible descendant views.
Esc -	Collapses nearest collapsible view and then, recursively collapses all its collapsible descendant views.
Esc l	Is equivalent to Esc - followed by Esc /. Very useful just after you open a large document to see its outline.

XXE_user_preferences_dir is:

– \$HOME/.xxe/ on Unix,

-
- `%SystemDrive%\Documents and Settings\%USERNAME%\Application Data\XMLmind\XMLeditor\`on Windows 2000/XP,
 - `%SystemDrive%\winnt\Profiles\%USERNAME%\Application Data\XMLmind\XMLeditor\`on Windows NT.