

# XMLmind XML Editor Tutorial

---



This tutorial is licensed under a [Creative Commons Attribution 3.0 Unported License](https://creativecommons.org/licenses/by/3.0/). This means that you can freely use it, adapt it and redistribute it.

Download this tutorial in source form (.zip archive containing .html, .png, .swf, etc, files) and in PDF, Office Open XML (.docx, can be opened in Word 2007+) or OpenOffice (.odt, can be opened in OpenOffice.org 2+) formats from <http://www.xmlmind.com/xmleditor/documentation.shtml>.

## Before following this tutorial

Except for the first few ones, this tutorial is organized in largely independent, short, lessons. Almost all lessons contain a short (less than 1mn) screencast. It's strongly recommended to take the time to read the text of the lesson before watching the corresponding screencast.

We'll almost exclusively use [DocBook 5](https://dtd.oasis-open.org/docbook/) examples in these lessons. This does not mean that this tutorial is about DocBook support in XMLmind XML Editor. Everything you'll learn in this tutorial applies to all document types: [DITA](https://dtd.oasis-open.org/dita/), [XHTML](https://dtd.oasis-open.org/xhtml/), etc. We have chosen DocBook mainly because this document type uses self-descriptive names for its elements: `para` for a paragraph, `itemizedlist` of an itemized list, etc. Prior knowledge of DocBook is not required in order to follow this tutorial.

If you find that this tutorial is too comprehensive and don't have enough time to follow it, we nevertheless recommend to read (no screencasts) this [minimal tutorial](#).

### Getting started

Overview of the user interface .....	3
Creating a document .....	9
Basic editing .....	12
Easier editing .....	14
Copy, Cut, Paste and Delete .....	17
Replacing nodes .....	19
Converting elements .....	20
Setting attributes .....	21
Inserting special characters .....	24
Inserting images .....	26
Editing tables .....	27

### Modular documents

Creating a modular document .....	29
Inserting boilerplate content .....	31
Working with a document set .....	33

### Becoming productive

Custom document templates .....	36
Quickly paste selected text using mouse button #2 .....	37
Inserting custom element templates .....	38
101 ways to select nodes .....	39
Quickly adding an attribute .....	42
Drag and drop .....	43
Automating repetitive tasks by recording macros .....	44
Custom keyboard shortcuts .....	47

### Specialized tools

Adding MathML equations .....	49
Easily create DocBook olinks .....	52

Reviewing changes using the Compare tool ..... 55

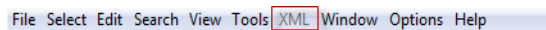
# Overview of the user interface

What's described below is the user interface you'll get “out of the box” after you install XMLmind XML Editor.

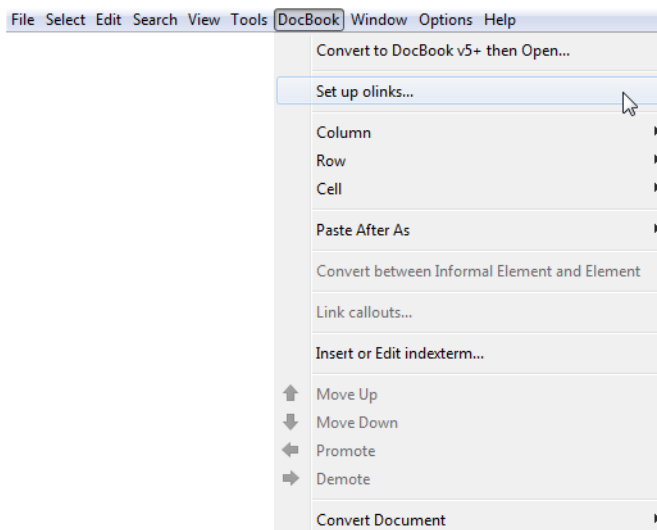
Please remember that XMLmind XML Editor is an XML editor *à la carte*, which is just as powerful as you need it to be. Out of the box, you get an application which is as small and simple as possible. If you are missing a feature, then this feature is probably one mouse click away from you. Depending on the feature, select it in the [Options/Preferences, General/Features preferences panel](#) or download it and install it using [menu item Options/Install Add-ons](#).

From left to right, top to bottom:

## The menu bar:



After a DocBook 5 document is opened:



The **XML** menu is disabled (grayed) when no documents are opened in XMLmind XML Editor (or when there is no specialized menu associated to the type of the document being edited, e.g. DITA ditaval).

The name of this menu changes and the menu is populated with items when you open a document in XMLmind XML Editor or when you switch from a document type to another (e.g. switch from a DITA map to a DITA topic).

## The toolbar:



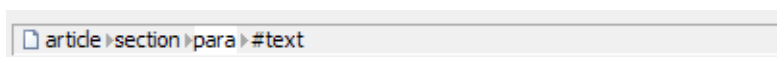
After a DocBook 5 document is opened:





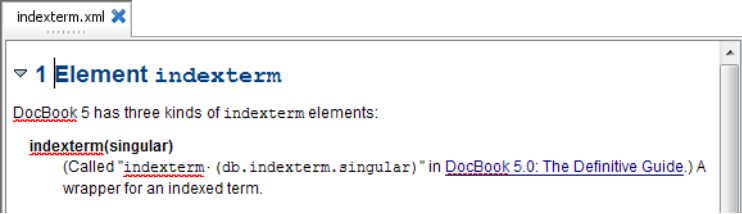
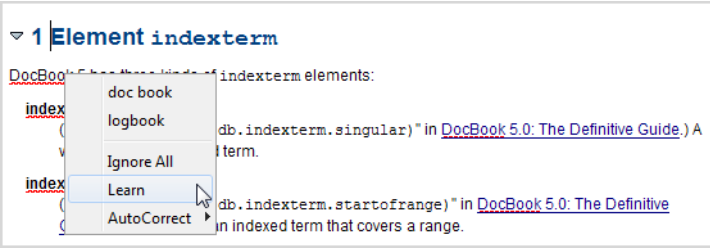
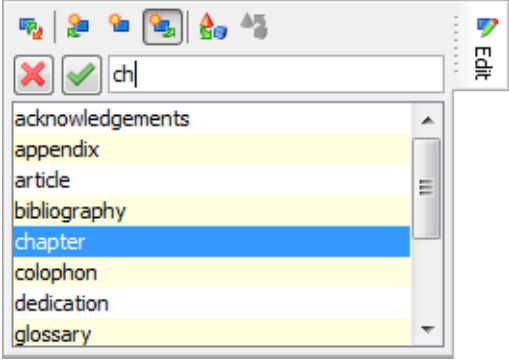
Initially the toolbar contains two groups of buttons: commonly used file commands and generic editing commands not requiring the user to specify an argument.

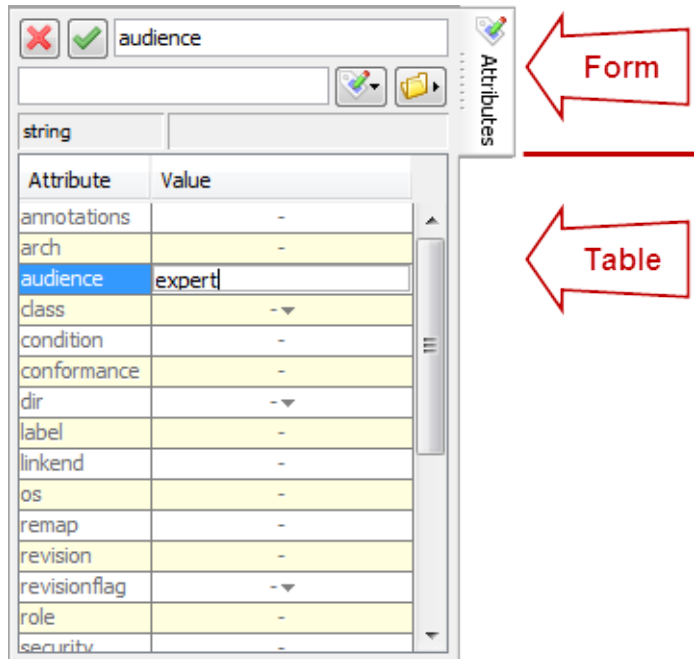
When a document is opened in XMLmind XML Editor, a third group of buttons is added to the toolbar.

## The node path bar:

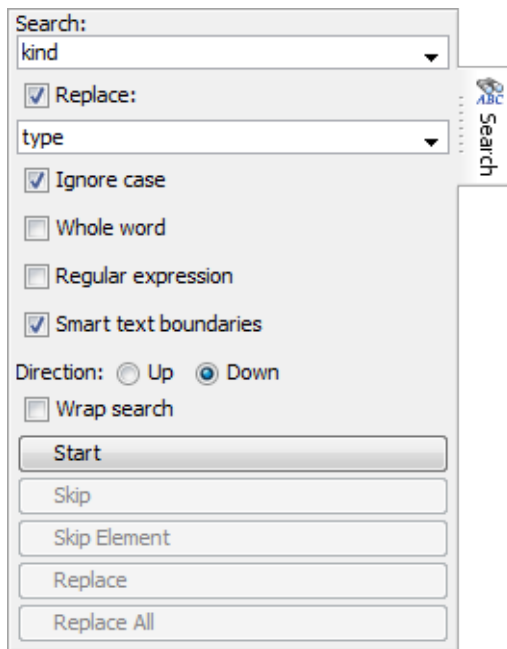


Shows which XML node is selected and which are its ancestor elements. Allows to explicitly select an XML node.

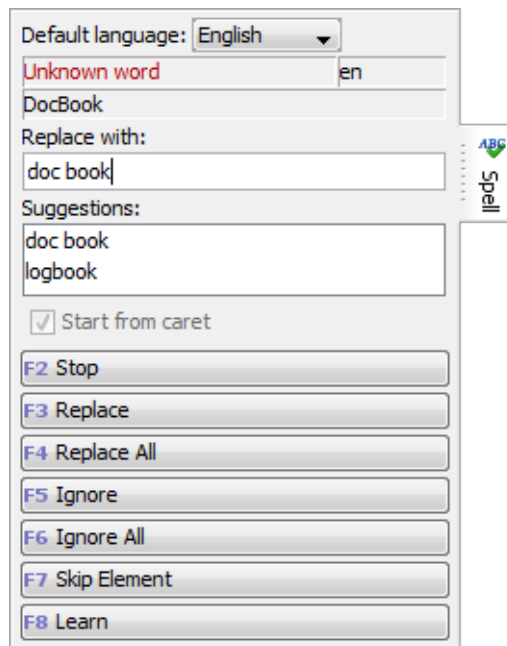
	<p>The node path bar is a key user interface component of XMLmind XML Editor. Without it, you could not author XML using just a tag-less, word processor-like, styled view.</p>
<p>The  <b>Search</b>→<b>Find Element</b> button.</p>	<p>Allows to find an element by its name, one of its attribute, the text it contains or by a combination of these criteria.</p>
<p><b>Buttons allowing to follow link:</b></p> 	
<p><b>Document tab:</b></p> 	<p>A document tab contains up to 5 different, synchronized, views of the same document. By default, a document tab only contains a single view styled using the “normal” <a href="#">CSS</a> stylesheet.</p>
<p><b>Right-click menu:</b></p> 	<p>If you right-click on a misspelled word, this displays a popup menu allowing to correct the spelling error or to ignore an unknown word.</p> <p>If you right-click elsewhere, this displays a popup menu variant of the <b>Edit</b> menu.</p>
<p><b>The Edit tool:</b></p> 	<p>Allows to add, replace, convert and wrap XML nodes.</p>

**The Attributes tool:**

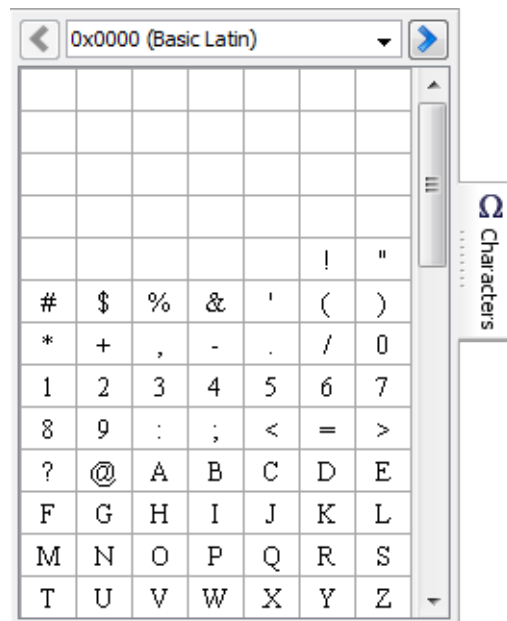
Allows to edit the attributes of the selected element.

**The Search tool:**

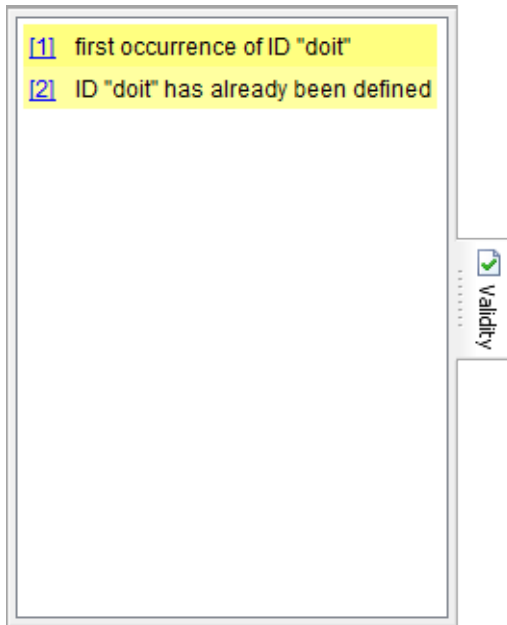
Allows to search and replace text in the document being edited.

**The Spell tool:**

Allows to check the spelling of words in the document being edited.

**The Characters tool:**

Allows to insert special characters.

**The Validity tool:**

Displays error messages when the document being edited is invalid with respect to a DTD or schema.

Clicking on an error number selects the corresponding element in the document view.

**The Validity state button:**

Shows the validity state of the document being edited: ✓ OK, ⚠ semantic warnings, ❌ semantic errors, ⚠ cross-reference errors, ✖ invalid values in some attributes and/or in some elements, ❌ invalid structure of the document.

Clicking this button allows to explicitly check the validity of the document being edited. Normally this is not useful as the validity is automatically checked when you open a document and each time you save it.

**Status message area:**

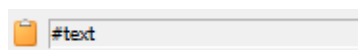
"E:\src\4toxe\demo\docbook5-sample\indexterm.xml" is invalid.

Clicking on ▲ displays a dialog box showing the message log.


ABC toggle used to activate and deactivate the automatic (that is, on-the-fly) spell checker.

Absent in XMLmind XML Editor Personal Edition.

Pressing the **INS/OVR** button (keyboard shortcut: `ESC INS`) allows to switch from Insert Mode to Overwrite Mode.


**The Clipboard content tool:**

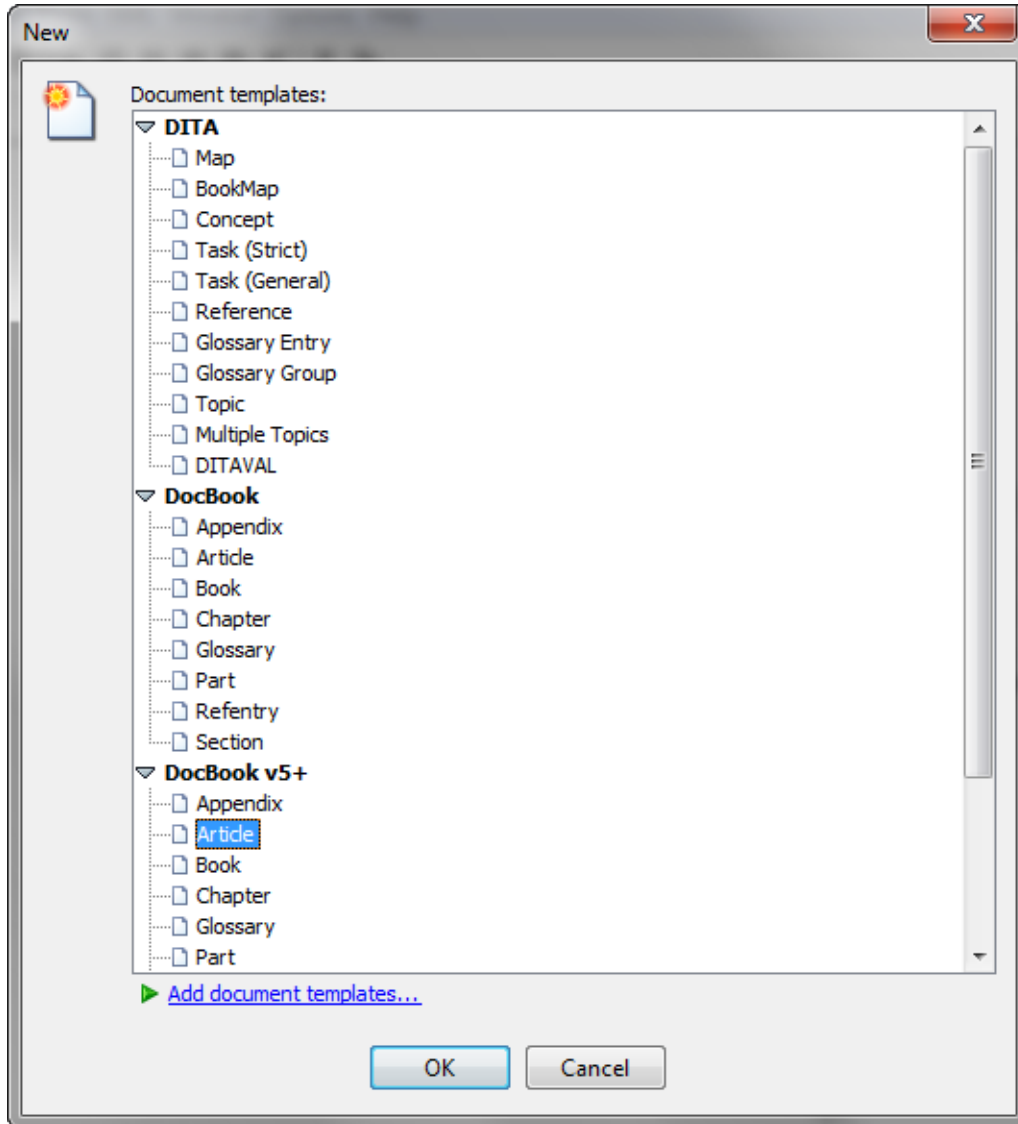
Indicates what has been copied to the system clipboard.

Clicking on  displays a dialog box showing the content of system clipboard.

# Creating a document

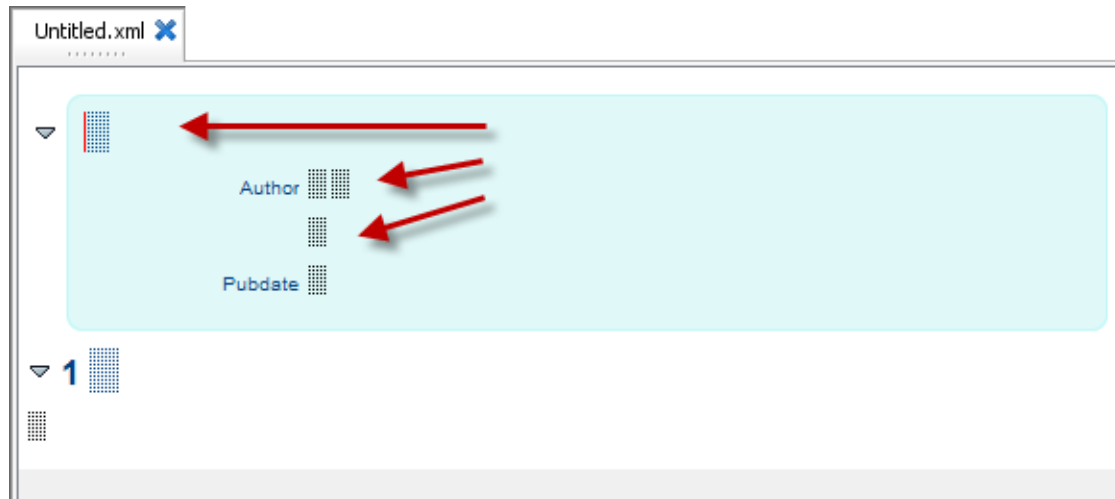
---

Creating a document is done by selecting the customary  **File**→**New** menu item. This displays a dialog box allowing to choose a document template.



Each document template has a name and belongs to a category loosely corresponding to a document type: **DITA** (groups topic, map, bookmap and ditaval), **DocBook** (that is, DocBook v4+), **DocBook v5+** and **XHTML**.

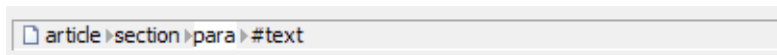
Let's suppose you want to create a DocBook 5 article. Here's what you'll see at first:



The little squares are text placeholders. That is, each little square represents an *empty text node*. When the caret is inside such placeholder, you can directly type some text.

You can move the caret inside a text node, whether empty or not, by clicking on it or by pressing `Tab` or `Shift-Tab`. `Tab` moves the caret to the following text node. `Shift-Tab` moves the caret to the preceding text node.

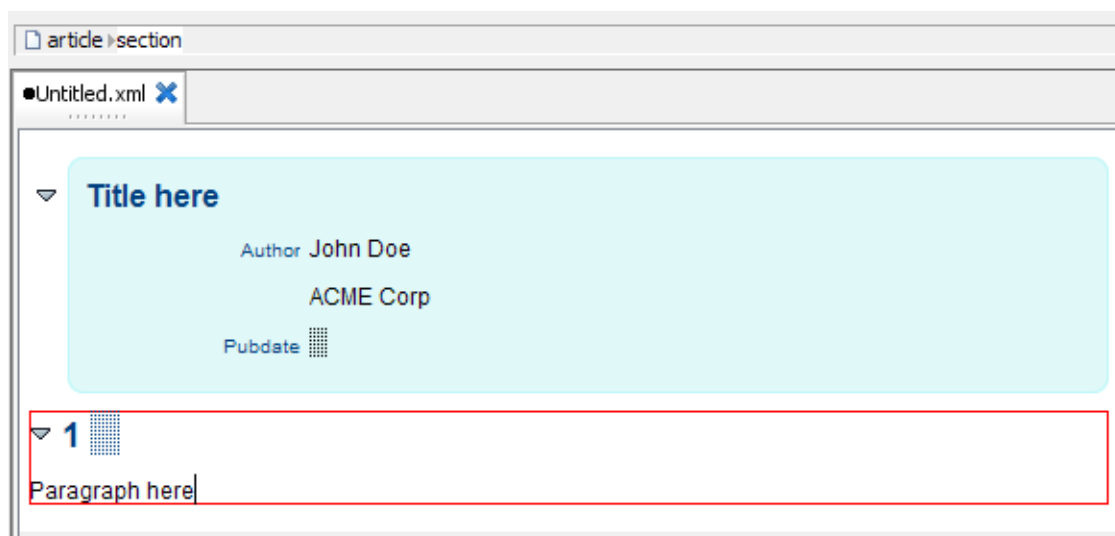
These tag-less placeholders do not convey any meaningful information, so how can you understand what you see? The answer is: look at the *node path bar*. The node path bar shows you which XML node is selected and which are its ancestor elements.



In the above screenshot, the caret is located inside a text node (`#text`) contained in a `para` element, itself contained in a `section` element, itself contained in an `article` element.

Because the caret is contained in some text found in a `para` element, this `para` element is said to be *implicitly selected*. Editing commands such as `Edit→Copy`, `Edit→Cut`, `Edit→Replace`, etc, will all act on this `para` element. This can be surprising because there is no visual clue that something is selected.

The node path bar also allows to explicitly select an XML node. Click in the node path bar on "`#text`" or on the name of an element, for example "`section`", and you'll select the corresponding text node or element. When you do this, you have *explicitly selected* the XML node and in such case, a red box is drawn around it.






## Related productivity tips

▶ [Custom document templates](#)

## Basic editing

---

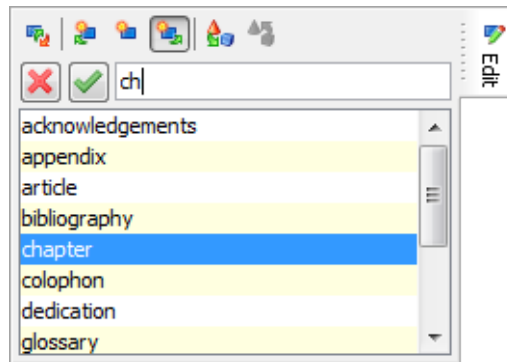
In order to add some content to your XML document, you must:

1. Select an XML node.
2. Use  **Edit**→**Insert Before**,  **Edit**→**Insert** or  **Edit**→**Insert After** in order to add a new element or text node respectively *before*, *into* (that is, at caret position) or *after* the selected node.

You have learned in the [previous lesson](#) that moving the caret into a text node implicitly selects the element containing this text node. You have also learned that clicking in the node path bar on an element name (e.g. "section") or a node name (e.g. "#text") explicitly selects the corresponding node. When a node is explicitly selected, a red box is drawn around it. When an element is implicitly selected, there is no visual clue that something is selected.

In this lesson, you'll learn to use the 3 **Insert** commands. In practice, you'll use **Insert After** most of the time.

The **Insert** commands are found in the **Edit** menu, but are implemented by the **Edit** tool:



In order to use an **Insert** command, you can click on the corresponding button or you can use one of the following keyboard shortcuts (on the Mac, use the Command key, not the Control key):

- **Ctrl-I** (**I** is for **Insert**) invokes command **Insert**, which inserts an element or text node *inside* selected element, at caret position.
- **Ctrl-H** (**I** is for **Insert** and letter **H** is before **I** in the alphabet) invokes command **Insert Before**, which inserts an element or text node *before* selected node. (On the Mac, use **Cmd-B**.)
- **Ctrl-J** (**I** is for **Insert** and letter **J** is after **I** in the alphabet) invokes command **Insert After**, which inserts an element or text node *after* selected node.

The **Edit** tool requires you to specify what you want to insert. An element is specified by its name and a text node is specified as "(text)". In order to let you to do this, the **Edit** tool lists all the elements you may insert given the chosen operation and the currently selected XML node.

A single click suffices to select an item from this list. Alternatively, you can type the name of the element you want to insert. Because the **Edit** tool supports auto-completion, suffice to type the first few letters of an element name and then press **Enter** to confirm your choice. After doing that, the keyboard focus is automatically returned to the document view.

If, when using the **Edit** tool, you don't see the element you want to insert, then it's either because you have chosen the wrong operation (e.g. **Insert** instead of **Insert After**) or because you have not selected the right element.

For example, let's suppose a `para` element is currently selected because the caret is found inside it. Let's suppose you want to add a chapter to your document. If you invoke **Insert After**, the **Edit** tool will not list `chapter`. You have to first select the `chapter` containing the `para`, for example by clicking on "chapter" in the node path bar, and then use **Insert After**.

## The `Ins` keyboard shortcut

In the above screencast, we have used **Insert After** and selected "(text)" from the list to add an empty text node after the `command` element. However, in order to add an empty text node after the `trademark` element, we have just pressed `Ins` (`F1` on the Mac). The `Ins` keyboard shortcut is the one of most useful hotkeys of XMLmind XML Editor. It inserts an empty text node after the selected element and if there is already a text node there, it moves the caret inside this text node.

Other useful keyboard shortcuts are:

- `Shift-Ins` inserts an empty text node before selected element.
- `Ctrl-Ins` inserts the same element after selected element.
- `Ctrl+Shift-Ins` inserts the same element before selected element.

## Easier editing

---

The previous lesson, [Basic editing](#), may leave you with the impression that XMLmind XML Editor (XXE for short) is straightforward, but not very convenient, to use. In fact, XXE has a number of secondary features which makes it convenient to use:

- the text selection,
- convert commands implemented by some toolbar buttons,
- the possibility to repeat a command,
- add-style (as opposed to **Insert After**) commands implemented by some toolbar buttons,
- keyboard shortcuts commonly found in word-processors.

### The text selection

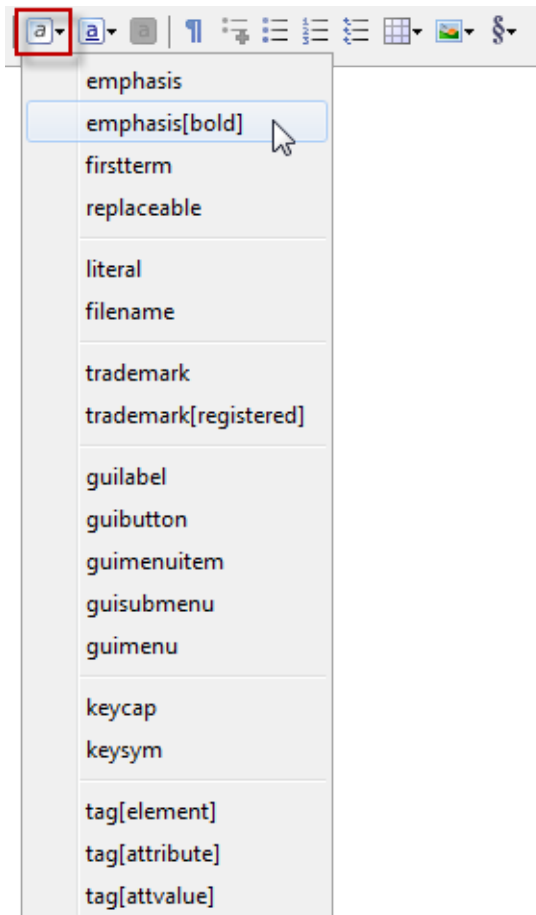
XMLmind XML Editor supports text selection as well as node selection.

There is little to say about text selection because it works as expected in any text editor or word processor: simply click on some text and drag your mouse to the end of selection. Other examples: `Shift-button1` extends the selection to the location clicked upon, `Shift-Down_Arrow` extends the selection to the next line, `Ctrl+Shift-Left_Arrow` extends the selection to the next word.

The text selection is rendered using a light red background.

### Toolbar buttons invoking convert commands


Once you have selected some text by dragging the mouse over it, you can use one of the buttons found in the toolbar to convert this selection to a commonly used element (we'll discuss the general case in [another lesson](#)).




For example, selecting **emphasis[bold]** from the menu depicted in the above screenshot converts the text selection to an `emphasis` element having a `role="bold"` attribute (there is no `bold` element in DocBook v5+).


## Repeating a command

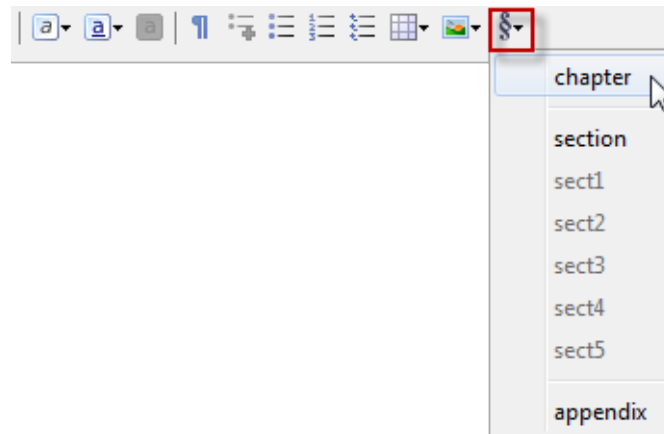
Once you have, for example, converted the text selection to an `emphasis` element having a `role="bold"` using the "Convert to emphasis" toolbar button, you can select another piece of text and convert it too to **emphasis[bold]** simply by pressing `Ctrl-A` (A like A gain).


`Ctrl-A` is the keyboard shortcut for  **Edit**→**Repeat**. Command **Repeat** allows to repeat the last "repeatable command". Most commands which require the user to type something or to select an item from a list have been made repeatable.

## Toolbar buttons invoking add commands

Some buttons of the toolbar allows to convert the selection to commonly used elements. Other buttons allow to add commonly used elements. For example, toolbar button  adds an itemized list.

By add, we mean: *add after the currently selected element at the closest location where this is allowed by the DTD or schema*. For example, if the caret is found in a `para` contained in a `section`, itself contained in a `chapter`. Clicking the  toolbar button and selecting **chapter** will add a new chapter after the current one.



Compare this to what we had to do in [the previous lesson](#) in order to add such chapter using  **Edit→Insert After**.



## Convenient keyboard shortcuts

The easiest way to add another list item after an existing one is to press `Ctrl-Enter` inside the list item.

The easiest way to add another paragraph after an existing one is to press `Enter` at the very end of the paragraph.

In fact, `Enter` splits the paragraph at caret position. That's why if you press `Enter` at the very end of the paragraph, this creates a new, empty, paragraph.




Pressing `Backspace` at the very beginning of a paragraph works as expected: it joins this paragraph to the preceding one. And, of course, pressing `Del` at the very end of a paragraph joins this paragraph to the following one.

Note that `Enter`, `Backspace` and `Del` work this way only inside a paragraph. `Enter` inside a paragraph invokes a specialized variant of more general command  **Edit→Split**. `Backspace` and `Del` respectively at the beginning and at the end of a paragraph invoke specialized variants of more general command  **Edit→Join**.

# Copy, Cut, Paste and Delete

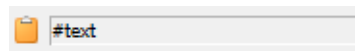
---


## The customary Copy, Cut and Delete commands

The  **Edit→Copy** (`Ctrl-C`),  **Edit→Cut** (`Ctrl-X`) and  **Edit→Delete** (`Backspace`, `Del` and `Ctrl-K` —**K** like **Kill**) all work as expected in any text editor or word processor. Their keyboard shortcuts are the usual ones. These commands operate on both the text selection and the node selection.



However, in XMLmind XML Editor, these commands also have the following specificities:

- A special user interface component found at the bottom/right of the main window indicates what has been copied to the system clipboard. Commands **Copy** and **Cut** of course update this indicator.




(Clicking on  displays a dialog box showing the content of system clipboard.)



- Sometimes the **Cut** and **Delete** commands will refuse to work because doing so would make the document invalid. For example, commands **Cut** and **Delete** cannot be used to delete the `title` child element of a `DocBook` section or `chapter`.
- Commands **Copy**, **Cut** and **Delete** will happily operate on the implicitly selected element, which can be surprising. For example, click inside a `para` to make this `para` implicitly selected. Now simply press `Ctrl-X` to cut this `para`.

In the case of command **Delete**, only the  **Edit→Delete** menu item,  toolbar button and `Ctrl-K` keyboard shortcut operate on the implicitly selected element. As expected, pressing `Backspace` and `Del` only deletes the explicit —highlighted in red— selection.

## Three Paste commands instead of just one

The  **Edit→Paste** (`Ctrl-V`) command works quite normally:

- If there is an explicit text or node selection, **Paste** replaces this selection with the content of the system clipboard.
- If there is no explicit text or node selection, **Paste** inserts the content of the system clipboard at caret position.

The two other **Paste** commands,  **Edit→Paste Before** (`Ctrl-U`; letter **U** is before **V** in the alphabet) and  **Edit→Paste After** (`Ctrl-W`; letter **W** is after **V** in the alphabet), are specific to XMLmind XML Editor. These commands insert the content of the system clipboard respectively before and after the node selection. They do not work if there is a text selection.

These commands are useful to move elements around within the document or across different documents. For example, you can use **Cut** to cut a list item and then use **Paste After** to paste it after the last item of the list.

When the **Paste** command you intend to use is disabled (“grayed”), it's probably because you have chosen the wrong operation (e.g. **Paste** instead of **Paste After**) or because you have not selected the right element. This behavior, which is typical of XMLmind XML Editor, has already been described for the [three Insert commands \(Insert Before, Insert, Insert After\)](#).

## Related productivity tips

- ▶ [Quickly paste selected text using mouse button #2](#)
- ▶ [101 ways to select nodes](#)

## Replacing nodes


---

Generally, when you insert a new element, XMLmind XML Editor automatically creates for you the *valid element having the simplest content*. For example, if you insert a `note` element in a DocBook document, this `note` contains an empty `para`.

In other cases, the simplest valid content is, well, too simple to be useful. In such cases, XMLmind XML Editor is expected to have been configured (by XMLmind engineers or by third-party consultants) to insert the *most commonly used valid content*. For example, if you insert a `figure` element in a DocBook document, this `figure` contains an `imagedata` element.

Now let's suppose you don't want the newly inserted `note` to contain a paragraph. Instead, you want it to contain an `itemizedlist` element. Also, you don't want the newly inserted `figure` to contain an image. You want it to contain a `programlisting` element.


You have learned that you can select the `para` contained in the `note`, insert after it an `itemizedlist`, then delete the unwanted `para`. (In that order, because the DocBook schema, hence XMLmind XML Editor, will not allow a `note` to become empty.)

In fact, the  **Replace** command has been designed to do exactly that, but in a single step. The **Replace** command allows to replace the selected nodes by an element or by a text node (if allowed by the schema, of course). Like all the other generic editing commands, this command is found in in the **Edit** tool, in the **Edit** menu and in the popup menu displayed when you right-click in the document view.

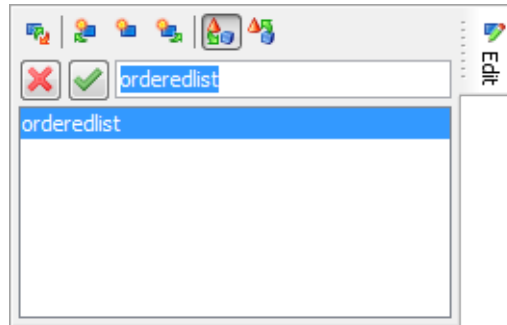
## Related productivity tips

▶ [Inserting custom element templates](#)


## Converting elements

You have learned in a previous lesson, [Easier editing](#), that some toolbar buttons allow to convert the text selection to commonly used elements such as `emphasis`, `literal`, `link`, etc. In fact, these buttons all invoke  **Edit**→**Convert** (keyboard shortcut: `Ctrl-T`, **T** like **Transform**). The **Convert** command may be applied to the node selection as well as to the text selection.

When a single element is selected, **Convert**, to make it simple, allows to change the name of this element. For example, you can select a `note` element and convert it to a `caution` element. You can select an `itemizedlist` and convert it to an `orderedlist`.




Note that the **Convert** command will not let you convert a `note` to an `example` because the content model of a `note` element is not compatible with the content model of an `example` element (the `example` element must begin with a `title` child element).

When a node range is selected, **Convert** allows to wrap this node range into a new element. For example, if you first select a `programlisting` and then use  **Select**→**Extend Selection to Following Sibling** (keyboard shortcut: `Esc Right_Arrow`) to extend the node selection to the `calloutlist` found after it, you can wrap these two elements into an `informalexample`.

### What is a node range?

Some of the commands of XMLmind XML Editor (e.g. **Paste**, **Replace**, **Convert**) can be applied to a *node range*. What we call a node range here is one or more consecutive sibling nodes. A node range may comprise different types of nodes: element, text node, comment, processing-instruction. What counts is that all the nodes have the same parent element and are consecutive.

Now what if you want to wrap a `para` into a `blockquote`? You cannot use **Convert** to do that because this command merely changes the name of an element when a single element is selected. When you need to perform this kind of task, you'll have to use a variant of command **Convert**:  **Edit**→**Convert [wrap]** (keyboard shortcut: `Ctrl+Shift-T`). Unlike plain **Convert**, **Convert [wrap]** always wraps the selection into a new element.

## Related productivity tips

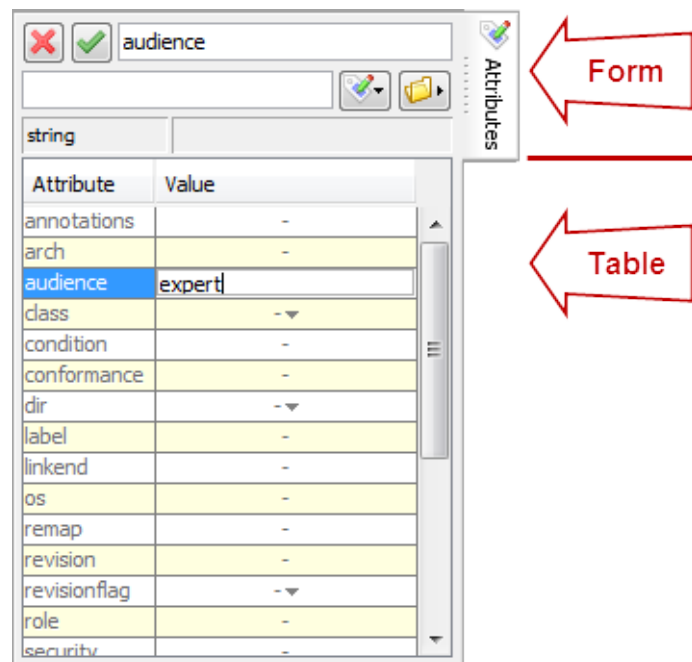
► [101 ways to select nodes](#)

## Setting attributes

The **Attributes** tool allows to edit the attributes of selected element. This tool is disabled (i.e. grayed) when some text or multiple nodes are selected.

The **Attributes** tool comprises two parts:

1. The upper part, a small *form*, which supports auto-completion and specialized attribute editors (specialized dialog boxes).
2. The lower part, a larger *table*, which displays all attributes. This table also allows the user to set attributes on the selected element.



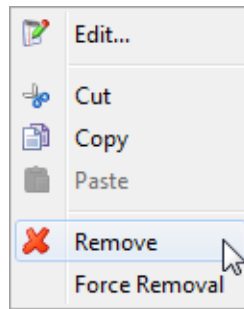
### The attributes table

The most basic way to specify the value of an attribute is to type it in the lower part, the attributes table. For example, let's suppose you are authoring a DocBook `article`. Specifying the `audience` attribute of the root `article` element consists in

- selecting the root element,
- clicking on the `audience` row,
- typing, for example "expert", in the **Value** cell,
- and finally pressing `Enter` to commit the change.


In some cases, for example the `class` attribute of `article`, the **Value** cell contains a drop-down list. In such case, suffice to select an item from this list.


If you want to remove an attribute, do not specify its value as the empty string. Instead, *right-click* on its row in the attributes table. Doing this pops up a menu allowing to perform various actions on the value of the attribute being clicked upon. Among these actions, you'll find **Remove**.



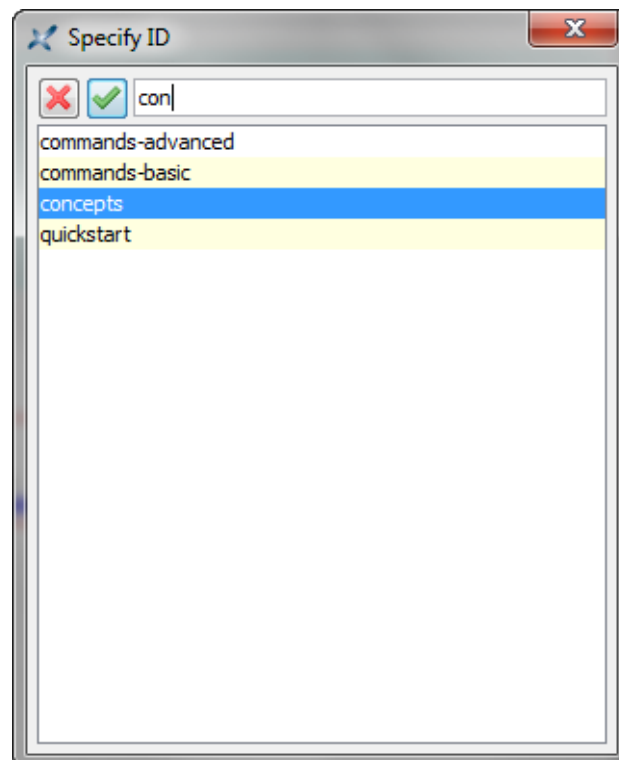
## The attribute form

The upper part, the attribute form, is often used for its specialized attribute editors. For example, let's suppose you have inserted a `figure` in your document. You may want to give an ID to this `figure`:

- First select the `figure`,
- then click on the `xml:id` row in the attributes table,
- and finally click the  **Edit** button.

Doing this pops down a menu, the very same menu which is displayed when you right-click on a row of the attributes table. This menu has an  **Edit** item.

This **Edit** item always displays a specialized dialog box allowing to edit an attribute “more comfortably” than with the attribute form or the attribute table. In the case of the `xml:id` attribute, the specialized dialog box will show you all the existing IDs. This way you'll be able to type an ID which does not already exist.



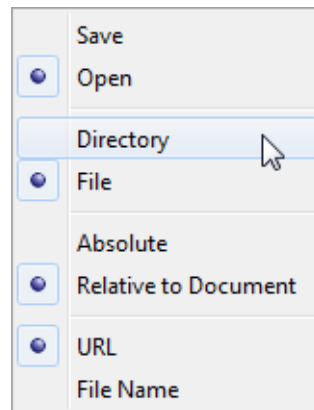
The `figure` you have inserted contains an `imagedata` element. Its required `fileref` attribute allows to specify the graphic file which is the source of the image. In order to specify a value for the `fileref` attribute,

- first click on the image placeholder icon to select `imagedata`,
- then click on the `fileref` row in the attributes table

- and finally click the  "Browse files" button.

Doing this displays a file chooser dialog box (or an URL chooser dialog box, if you have checked **File→Use the URL Chooser** in XMLmind XML Editor Professional Edition).

Note that by default, the "**Browse files**" button specifies the value of the edited attribute as an *URL* which is relative to the location of the document being edited. If you want to specify something else, for example, the absolute filename of a save directory, then you'll have to *right-click* on this button. Doing this pops down a menu allowing to configure the behavior of this button.



## Related productivity tips

- ▶ [Quickly adding an attribute](#)

## Inserting special characters

There are 3 ways to insert in an XML document characters other than letters, digits and common punctuation signs:

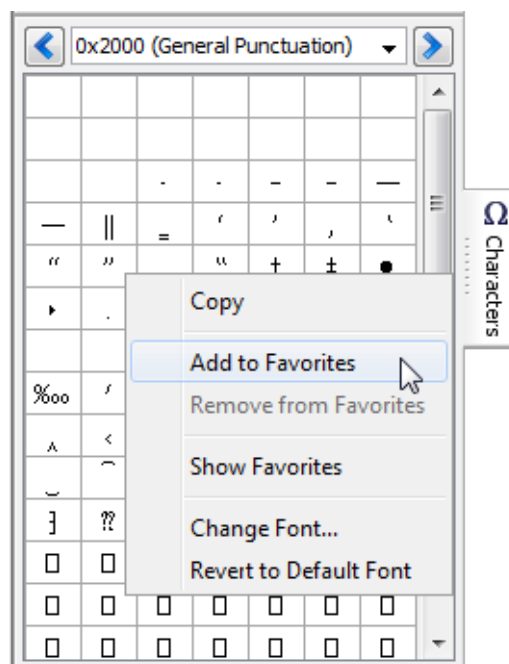
1. Directly type the special character (if this is possible) or copy it from a third-party application (such as Windows **Character Map** utility), then paste it in the XML document.

Note that the non-breaking space (Unicode code point: U+00A0; corresponds to well-known character entity `&nbsp;`), which is very commonly used, comes with its own keyboard shortcut: `Ctrl-Space`. Also notice that XMLmind XML Editor represents the non-breaking space as a small dot a little above the baseline of a text line. That is, this character has been made visible.

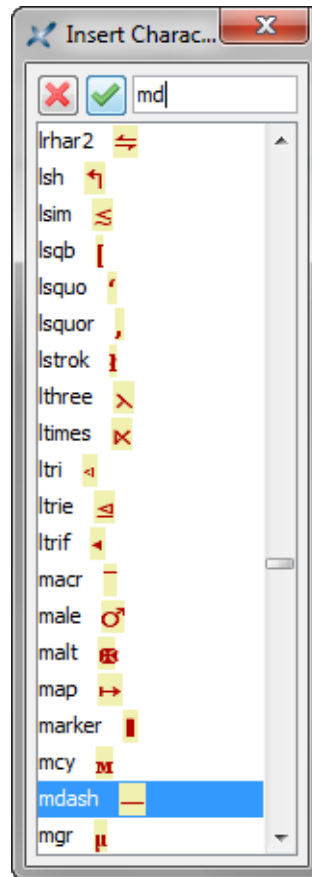
Popular operating systems are Mac OS X and Windows XP.

2. Click on the character displayed in a character palette of the **Characters** tool.

Because searching the same characters over and over in the **Characters** tool can be a daunting task, this tool has a **Favorites** palette. In order to add a character to the **Favorites** palette, right-click on it and then select item "Add to Favorites" in the popup menu.



3. Type `Esc n` (that is, press `Esc` then press `n`; `n` like `name`) to display this dialog box:





Then type the name of a well-known character entity (`mdash`, `rarr`, etc). The text field of this dialog box supports auto-completion, therefore suffice to type the first few characters of the entity name.

The command invoked by the above dialog box inserts the character having specified name. It does not insert a character entity. (XMLmind XML Editor offers no way to work with entities.)

After inserting a character using the above dialog box, it's possible to insert it elsewhere simply by pressing `Ctrl-A` (that is, **Edit**→**Repeat**). `Ctrl-A` (**A** like **Again**) allows to repeat the last “repeatable command”. Most commands which require the user to type something or to select an item from a list have been made repeatable.

## Inserting images

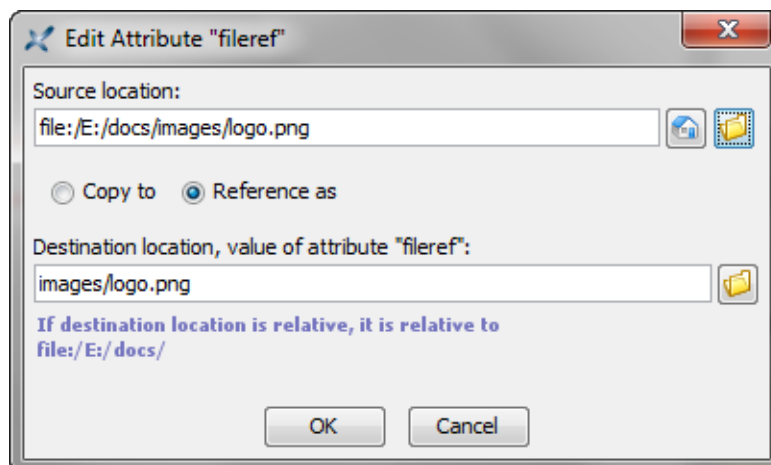
Out of the box, XMLmind XML Editor supports only the following image formats: GIF, PNG, JPEG, BMP and TIFF (TIFF only on the Mac). If you need more formats, then you'll have to install one or more specialized add-ons called "*image toolkit plug-ins*". For example, the add-on called "Apache Batik image toolkit plug-in" adds [SVG](#) support to XMLmind XML Editor. Installing an add-on is done by selecting menu item **Options**→**Install Add-ons**.

The easiest way to insert an image in your document is to use the  **Add image** button found in the toolbar. After doing that, you still have to specify the graphic file which is the source of the image. Generally this is done by first clicking on  the image placeholder icon and then using the **Attributes** tool to specify the graphic file. DocBook example: clicking on the image placeholder selects the corresponding `imagedata` element. Then you have to specify a `fileref` attribute for this element.

Inserting image in a document is a very common task that's why there are more convenient ways to specify the graphic file which is the source of the image:

- Double-click on the image placeholder.
- *Or* drop an image file onto the image placeholder.

In both cases, this displays a specialized dialog box.



The "**Source location**" field allows to specify the graphic file which is the source of the image. The "**Destination location**" field allows to specify how this graphic file is to be referenced in the document. Note that this dialog box requires specifying URLs and not filenames.

For example, let's suppose that your document is found in the `docs/` directory and that all your graphic files are found in the `docs/images/` subdirectory. If you want to insert an image pointing to `docs/images/logo.png`, then simply make sure that the "**Reference as**" radio button is checked.

Now let's suppose you want to insert `C:\tmp\screenshot.png` in your document. You'll probably want to copy this file to `docs/images/` subdirectory and give it a more meaningful name (e.g. `login_dialog_box.png`). This is done by checking "**Copy to**" and specifying "`images/login_dialog_box.png`" in the "**Destination location**" field.

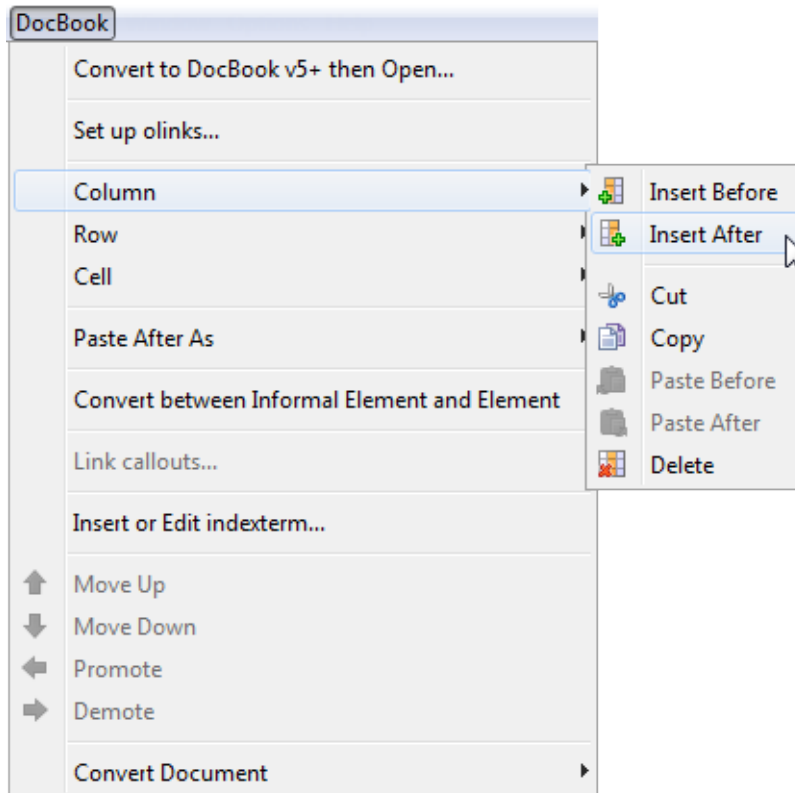
## Related productivity tips

► [Drag and drop](#)

## Editing tables

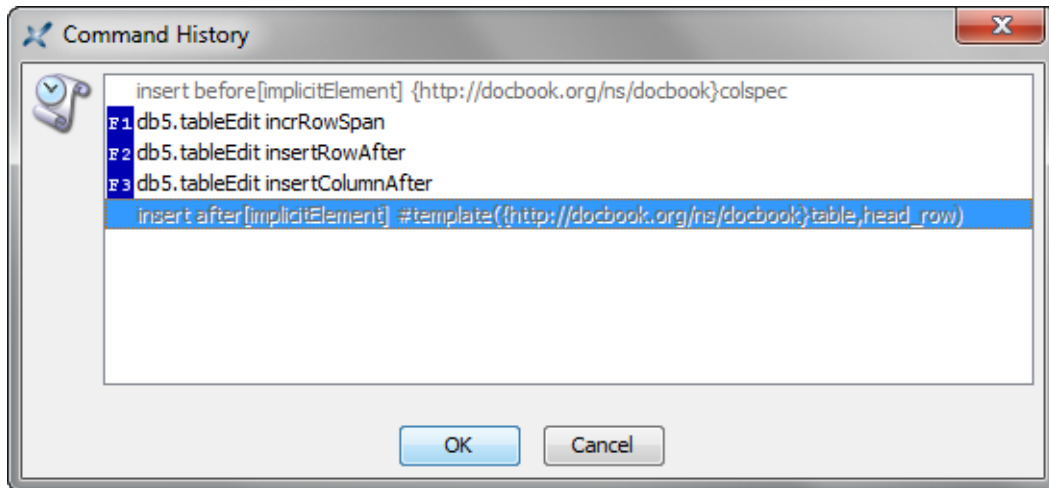
After inserting a table in a document, it's always possible to edit it using the **Edit** tool. For example, you can select a row and use **Insert After** to insert after it another row or you can select a cell and use **Insert Before** to insert before it another cell. However the **Edit** tool does not make it easy working on table columns.

Fortunately, the **DocBook**, **DITA Topic** and **XHTML** menus all have **Column**, **Row** and **Cell** submenus which allow to perform all sorts of operations on table columns, rows and cells. For example, **Column**→**Insert After** adds an empty column after the one containing the caret.



Note that adjusting the presentation of a table still requires you to use the **Edit** tool and the **Attributes** tool. For example, making the third column of a DocBook table twice as wide as the other columns requires you to insert a `colspec` element before the `thead` or `tbody` of the `tgroup` and then specify attributes `colnum="3"` and `colwidth="2*"` for the new `colspec`.

In the above screencast, the user pressed `Ctrl-A` (that is, **Edit**→**Repeat**) twice, one time to repeat **Row**→**Insert After** and another time to repeat **Cell**→**Increment Row Span**. If you don't remember which is the last repeatable command you have invoked, then you may want to use **Edit**→**Command History** (`Ctrl+Shift-A`).



## Related productivity tips

- ▶ [Inserting custom element templates](#)



# Creating a modular document

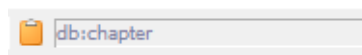
XMLmind XML Editor allows to include in document *B* some content found in document *A*. This feature works by using a special command called **Copy As Reference** and then by using one of the normal **Paste** commands. Unlike the ordinary **Copy/Paste** commands which duplicate content, **Copy As Reference** copies a *reference* to some content found in document *A*. This means document *B* will always be up to date no matter how you'll modify document *A*.

(Internally, XMLmind XML Editor uses standard mechanisms, e.g. [conref](#) for DITA and [XInclude](#) for DocBook, to implement this feature, so you don't have to worry about the portability of the modular documents you'll create.)




This feature is generally used to create modular documents. For example, a modular DocBook book (`Book.xml`) includes several chapters, each `chapter` element being authored in its own file (`Chapter1.xml`, `Chapter2.xml`, etc). This way, different authors may work on different chapters at the same time and also, it's more comfortable to work on a relatively short chapter rather than on a large, monolithic, book.

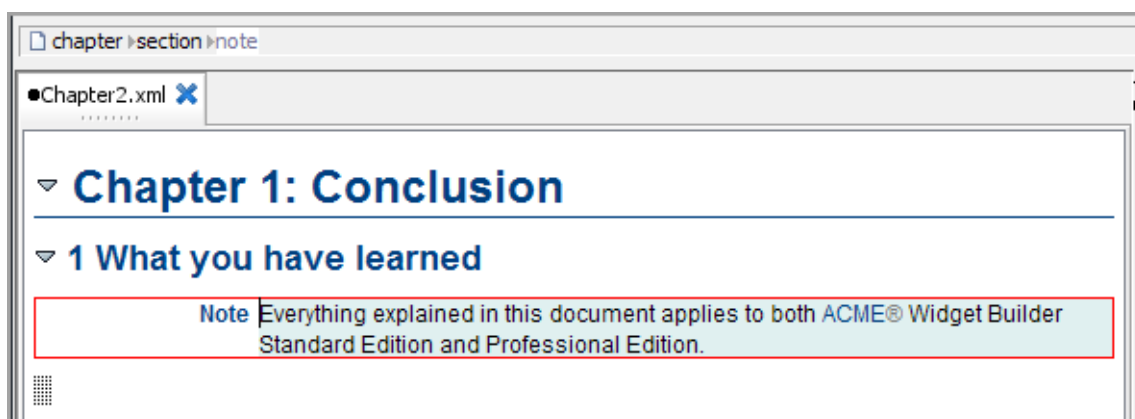
This feature is also used to include “boilerplate” content in a document: company name, product name, copyright statement, etc. More about this in next lesson: [Inserting boilerplate content](#).


 **Edit**→**Reference**→**Copy As Reference** (keyboard shortcut: `Ctrl+Shift-C`) works like ordinary  **Edit**→**Copy** (keyboard shortcut: `Ctrl-C`), except that the node selection must comprise a *single element and this element must have an ID* (or if it does not have an ID, it must be the root element of the document). After you invoke **Copy As Reference**, notice that the **Clipboard** tool displays the name of the referenced element using a dimmed blue/gray color.



Note that it's not possible to **Copy As Reference** some included content. If you want to do that, simply use the normal **Copy** command. The normal **Copy** and **Cut** commands both preserve references when they are used to copy content to the clipboard.

After using  **Edit**→**Paste Before** (keyboard shortcut: `Ctrl-U`),  **Edit**→**Paste** (keyboard shortcut: `Ctrl-V`) or  **Edit**→**Paste After** (keyboard shortcut: `Ctrl-W`) to paste the reference copied using **Copy As Reference**, the included content is rendered using a blue/gray color. This special color means that the included content is read-only. That is, it cannot be edited in place.




In order to edit some included content, you'll have to open the document which actually contains the data. Fortunately, this is done very easily by using menu item (or corresponding toolbar button)  **Edit**→**Reference**→**Edit Referenced Document** (keyboard shortcut: `Ctrl+Shift-E`).

From there, switching back to the included content is done by using menu item (or corresponding toolbar button)

 **Edit**→**Reference**→**Edit Referencing Document** (keyboard shortcut: `Ctrl+Shift-B`).

## Inserting boilerplate content

 The **Include** tool is hidden by default. You need to enable it by checking "**Enable the Include Tool**" in **Options**→**Preferences, Features** section.

We have explained in the previous lesson, [Creating a modular document](#), that **Copy As Reference/Paste** may be used to include in document *B* some content found in document *A*.

This feature is often used to include “boilerplate” content in a document: company name, product name, copyright statement, etc. By working this way, if one day, the value of a boilerplate text changes, you'll not have to manually update all the documents making use of this value.

Such boilerplate content is typically created in a document of its own (e.g. `boilerplate.xml`), so it can be easily shared between different documents and between different authors.

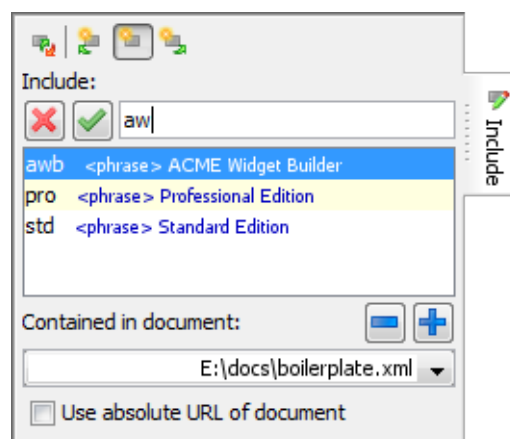
Remember that **Copy As Reference** works only if the node selection is a *single element having an ID*. So what to do if you just want to include some text, for example, the name of your product? In such case, you'll have to wrap this text in an element. For that, choose an element which has no specific semantics. This element is: `phrase` for DocBook, `ph` or `text` for DITA and `span` for XHTML. Example:

```
<phrase xml:id="awb">ACME Widget Builder</phrase>
```



The problem of boilerplate content appears to be solved:

1. Create your `boilerplate.xml` file. This is done once for all.
2. Use **Copy As Reference** (`Ctrl+Shift-C`) to copy a reference from `boilerplate.xml`.
3. Use **Paste** (`Ctrl-V`) to paste this reference in the document you are authoring.




However if you repeat steps 2 and 3 one hundred times a day, you'll quickly find very tedious switching from one document to the other. Fortunately, the **Include** tool has been specially designed to handle the special case of boilerplate content.




Let's suppose you want to include the name of your product in `SampleChapter.xml`, here's what you'll have to do:

1. Use the  button to specify once and for all the filename or URL of the document containing your boilerplate content. In the above screenshot, this document is `boilerplate.xml`.
2. Use `Ctrl+Shift-I` ( **Edit**→**Reference**→**Insert Reference**) to display the **Include** tool.

3. Type the ID of the element for which you want to insert a reference. Typing the first few letters is generally sufficient as the **Include** text field supports auto-completion.
4. Press `Enter` to insert the reference at caret position.

Note that the **Include** tool is not limited to inserting phrase references at the caret position. You can use it to insert all sorts of boilerplate content: paragraphs, tables, admonitions, etc. Therefore it also implements the following commands:  **Edit**→**Reference**→**Replace by Reference**,  **Edit**→**Reference**→**Insert Reference Before**,  **Edit**→**Reference**→**Insert Reference After**.

## Working with a document set

 The **File**→**Document Set** submenu is hidden by default. You need to enable it by checking "**Enable the 'File|Document Set' Submenu**" in **Options**→**Preferences, Features** section. (While at it, you may also want to review the options specified in the **Tools|Document Set** section.)

### What is a document set?

A *document set* is simply a set of related XML documents. These documents are related because they all contribute to the content of the *same deliverable*: PDF file, EPUB file, Web site, etc.

When XMLmind XML Editor knows that some opened documents are members of the same set:

- it will more thoroughly check the links which may exist between these documents,
- it will make it easier creating links between these documents.

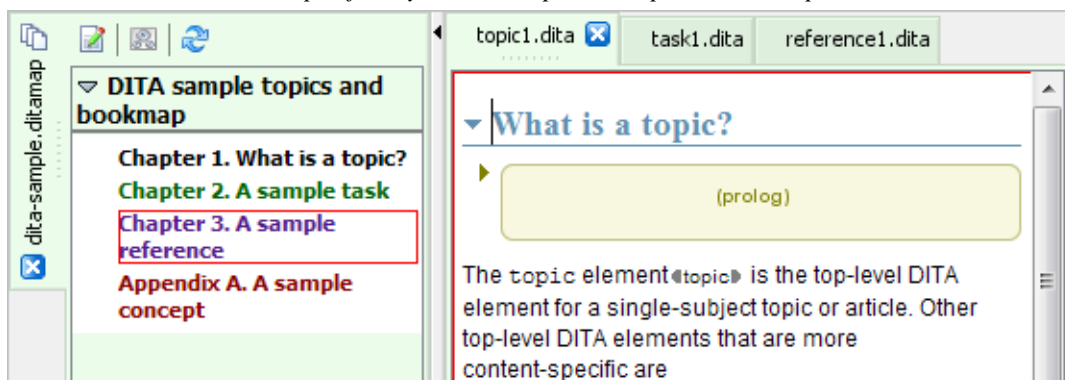
Modular documents, for example a DITA map referencing a number of topics or a modular DocBook book including a number of chapters, implicitly specify a document set. When you author such modular documents, it is strongly recommended to first open their document set in XXE.

### How to open a document set in XXE?

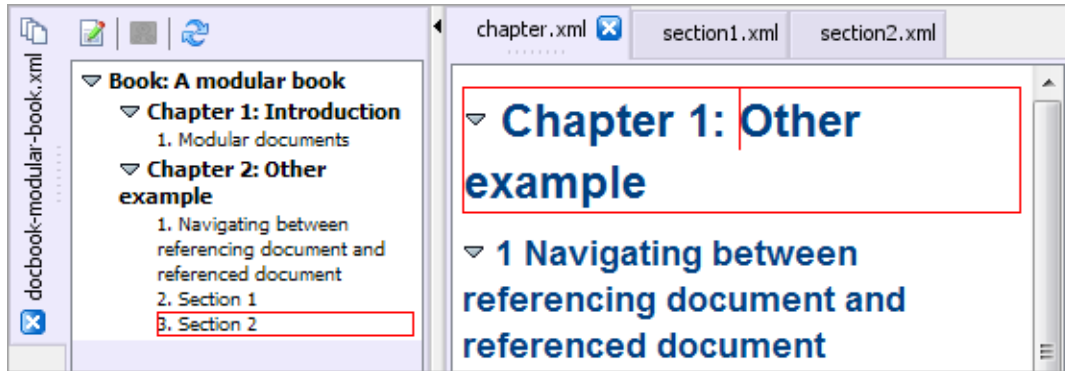
Use **File**→**Document Set**→**Open Document Set** to let XXE know about a given document set. This menu item displays a dialog box allowing to choose the file specifying the document set.

- If the selected file is a DITA map, then the members of the document set are the topics referenced in this map. See [figure](#) below.
- If the selected file is a master DocBook document (for example a modular book including a number of chapters, each chapter being found in its own XML file), then the members of the document set are the DocBook documents included in this master document. See [figure](#) below.
- In all other cases, the content of the selected file is not analyzed by XXE. Instead, the selected file specifies a directory and a filename extension. For example, selecting file `file:/home/john/site/buy.html` specifies the "`file:/home/john/site/*.html`" *glob pattern*. In this example, the members of the document set are all the files found in directory `file:/home/john/site/` having an "html" extension. See [figure](#) below.

*A document set specified by a DITA map and 3 topic members opened in XXE*



A document set specified by a modular DocBook book and 3 chapter or section members opened in XXE



A document set specified by a glob pattern and 3 XHTML page members opened in XXE



## The "Document Set" tool

As you can see it in the 3 above screenshots, a document set is represented by a special tool added to the left of XXE's main window: the "**Document Set**" tool. Each instance of this tool is given a specific, random, background color. This specific background color is also given to the tabs of all the opened documents which are members of the set.

The "**Document Set**" tool is a convenient, interactive, navigational tool: you can double-click on an element to open in XXE the member document corresponding to this element; dragging the selected element will drag the corresponding member document inside or outside XXE; etc. More information in *XMLmind XML Editor - Online Help*, [Using the "Document Set" tool](#).

However the most important part of its job is done behind the scene, because opening a document set in XXE automatically modifies the behavior of [the Validity tool](#) and [the Attributes tool](#):

- The diagnostics issued by the **Validity** tool about cross-reference errors will take into account the fact that the document being checked is a member of a set.
- The target of a link is almost always specified as an attribute value (XHTML example: the `href` attribute of the `a` element). That's why the **Attributes** tool, through its auto-completion feature, will suggest, not only link targets found within the document being edited, but also link targets found in the other members of the set.

## A simple use case

Modular book `mybook.xml` includes 3 chapters. Each chapter is found in its own file: `chapterA.xml`, `chapterB.xml` and `chapterC.xml`.

File `chapterA.xml` contains:

```

<chapter id="chapterA">
  <title>Chapter A</title>

  <para>Link to <link linkend="nowhere">nowhere</link>. Link to <link
  linkend="sectionB1">Section B1</link>.</para>

  <section id="sectionA1">
    <title>Section A1</title>

    <para>TODO.</para>
  </section>

  <section id="sectionA2">
    <title>Section A2</title>

    <para>TODO.</para>
  </section>
</chapter>

```

In the above file, the first `link` element points to a non-existent target and the second `link` points to the first section of `chapterB.xml`.

The user wants to check the links found in `chapterA.xml` and also to add an `xref` element pointing to the first section of `chapterC.xml`. In order to do that, she/he opens `chapterA.xml` in XXE.


Before opening `mybook.xml` as a document set:

- The **Validity** tool reports 2 cross-reference warnings: reference to non-existent ID "nowhere" and reference to non-existent ID "sectionB1".
- When the user inserts an `xref` element and specifies its `linkend` attribute, the **Attributes** tool suggests: `chapterA`, `sectionA1`, `sectionA2`.

After opening `mybook.xml` as a document set:

- The **Validity** tool reports 1 cross-reference warning: reference to non-existent ID "nowhere".
- When the user inserts an `xref` element and specifies its `linkend` attribute, the **Attributes** tool suggests: `chapterA`, `sectionA1`, `sectionA2`, `chapterB`, `sectionB1`, `sectionB2`, `chapterC`, `sectionC1`, `sectionC2`.

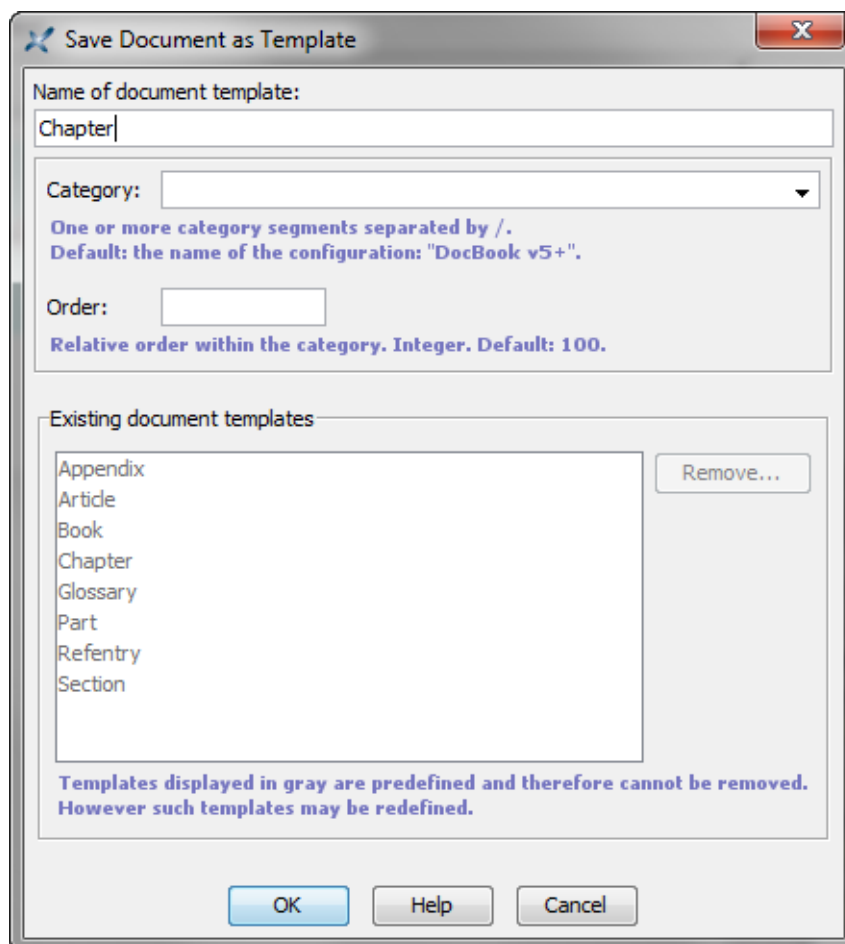
## Custom document templates

 Requires XMLmind XML Editor Professional Edition.

By default, when you create a new DocBook 5 chapter document, the `chapter` root element begins with a `title` child element. Let's suppose this not what you want. You want new chapters to always begin with the more comprehensive `info` element. The `info` element has a `title` child element, but also allows to specify meta-information (e.g. `author`, `pubdate`) about the chapter.

XMLmind XML Editor allows you to save any document as a *named document template*. This custom document template will then be listed by the **File**→**New** dialog box just like the stock document templates.

In order to do this, create or open the document you want to use as a template, make sure that the corresponding file has been saved to disk and then select menu item **Options**→**Customize Configuration**→**Save Document as Template**. Doing this displays a dialog box.




This dialog box allows you to give a name to your custom template. If you give the same name as an existing stock template (e.g. "chapter"), then your custom template will replace the stock one.

Note that you don't need to keep the file which has been used to declare the custom template. XMLmind XML Editor has made a copy of it, so you can safely delete it if you want.


## Quickly paste selected text using mouse button #2

---


 This functionality is disabled by default (because not all persons are at ease clicking with the mouse wheel). To enable it, you need to use **Options**→**Preferences, Edit** section and check "**Clicking with middle button pastes system selection**".

On Linux and on other operating systems using X-Window for their graphical user interfaces, after you have selected some text, for example by dragging your mouse over it, you can paste it elsewhere by just clicking mouse button #2 (also called middle button or mouse wheel). That is, no need to use **Copy** (`Ctrl-C`) / **Paste** (`Ctrl-V`) to paste the selected text. On any X-Window system, this feature, called *the system selection*, is native and works across applications.

This feature is so handy that XMLmind XML Editor implements it, not only on X-Window systems, but also on Windows and on the Mac. However, on Windows and on the Mac, this feature only works within XMLmind's document views.

Note that unlike  **Edit**→**Copy** (`Ctrl-C`) which copies characters as well as XML nodes, selecting text this way just copies *characters* to the system selection. For example, using this feature you can easily copy the content of an XHTML `p` element in order to paste it in a DocBook `para` element. (Using `Ctrl-C` to copy the content of an XHTML `p` and then using `Ctrl-V` to paste it in a DocBook `para` will not work if the `p` element has child elements.)

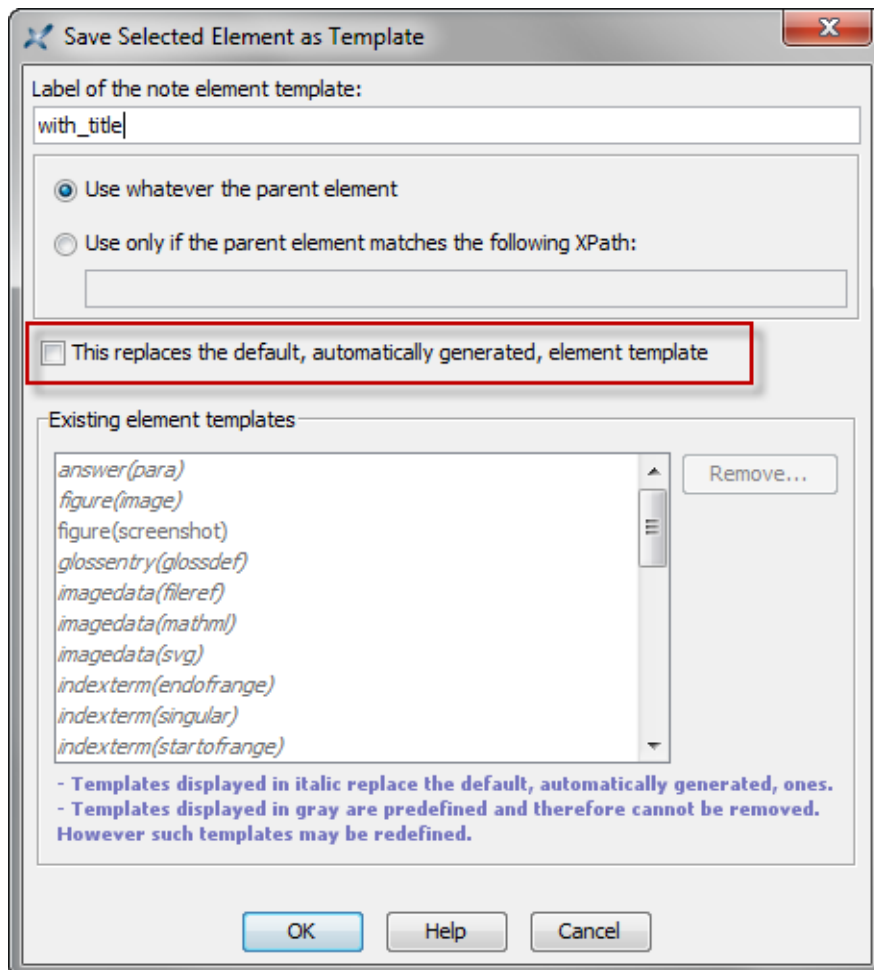
## Inserting custom element templates

 Requires XMLmind XML Editor Professional Edition.

Generally, when you insert a new element, XMLmind XML Editor automatically creates for you the *valid element having the simplest content*. For example, if you insert a `note` element in a DocBook document, this `note` contains an empty `para`.

But what if you often want your notes to begin with a `title` element? Will you have to use **Insert Before** again and again in order to insert by hand a `title` before the `para`? The answer is no. Suffice to do that once, select the customized `note` element and then save it as a *named element template*. This is done by using menu item **Options**→**Customize Configuration**→**Save Selected Element as Template**.

The dialog box displayed by **Save Selected Element as Template** has a checkbox called "**This replaces the default, automatically generated, element template**". If you check it, your custom element template replaces the one created by default by XMLmind XML Editor. That is, if you check it, all the `note` elements you'll insert will begin with a `title`.



Let's suppose you do not want to do that. Let's suppose the name of your custom element template is "with\_title". When you'll use the **Edit** tool in order to insert a `note` element, you'll see two `note` elements listed there: the default template called "note" and your custom template called "note(with\_title)". Notice that named element templates, whether custom ones or stock ones, are displayed using an italic font.

# 101 ways to select nodes

XMLmind XML Editor supports text selection as well as node selection.

There is little to say about text selection because it works as expected in any text editor or word processor. For example, `Shift-button1` extends the selection to the location clicked upon, `Shift-Down_Arrow` extends the selection to the next line, `Ctrl+Shift-Left_Arrow` extends the selection to the next word, etc.


You have already learned in the previous lessons that moving the caret into a text node implicitly selects the element containing this text node. You have also learned that clicking in the node path bar on an element name (e.g. "section") or a node name (e.g. "#text") explicitly selects the corresponding node. When a node is explicitly selected, a red box is drawn around it. When an element is implicitly selected, there is no visual clue that something is selected.

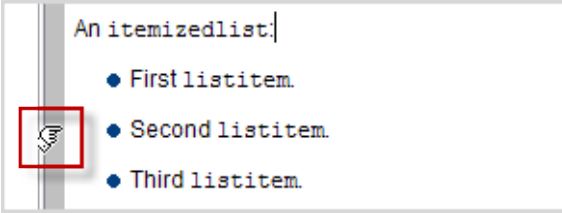
Now there other ways to select nodes, either by using the keyboard or by using the mouse. If you want to be more productive with XMLmind XML Editor, you need to learn some of these other methods.

## Mac users

- Use the `Cmd` key instead of the `Ctrl` key, except for `Ctrl-Tab` (insert tab character) and `Ctrl-Space` (insert non-breaking space character).
- Use `Ctrl-button1` to emulate mouse `button3`.
- Use `Alt-button1` to emulate mouse `button2`.

## Select a node

Using the keyboard	Using the mouse
<ul style="list-style-type: none"> <li>• <code>Ctrl-Up_Arrow</code> selects the parent of selected node. If there is no explicit node selection, this hotkey selects the text node containing the caret.</li> <li>• <code>Ctrl-Down_Arrow</code> selects the last selected child of selected element. That is, pressing repeatedly <code>Ctrl-Up_Arrow</code> and then pressing repeatedly <code>Ctrl-Down_Arrow</code>, works as expected.</li> <li>• <code>Ctrl+Shift-Up_Arrow</code> selects the preceding sibling of selected node.</li> <li>• <code>Ctrl+Shift-Down_Arrow</code> selects the following sibling of selected node.</li> </ul>	<ul style="list-style-type: none"> <li>• <code>Ctrl-button1</code> selects the node clicked upon. Repeating this action without moving the mouse selects its parent, then its grand-parent, etc.</li> <li>• Click on the non-editable ``decor" which has been generated for an element: the bullet of a list item, the number of a section, the image contained in a figure, the border of a table, etc.</li> <li>•  By default, this “interactive gray margin” is absent. To enable it, you need to use <b>Options</b>→<b>Preferences, Edit</b> section and check "<b>Add interactive left margin to the styled view</b>".</li> </ul>

Using the keyboard	Using the mouse
	 <p data-bbox="868 499 1401 719">Move the mouse to the gray margin found at the left of the document view. Here you'll notice that the cursor changes its shape. Move the cursor in front of any kind of “block” (paragraph, list item, row, row group, table) and click once. Mouse clicks in the gray margin select the block which is in front of the click location.</p> <p data-bbox="868 748 1401 967">In the case of the above screenshot, this selects the list item. Click again (not too fast otherwise XMLmind XML Editor will think that you are double-clicking) without moving the mouse and this will select the parent of the list item: the itemized list. Clicking again without moving the mouse would select the parent of the itemized list and so on.</p>

## Extend the node selection

Some of the commands of XMLmind XML Editor (e.g. **Paste**, **Replace**, **Convert**) can be applied to a *node range*. What we call a node range here is one or more consecutive sibling nodes. A node range may comprise different types of nodes: element, text node, comment, processing-instruction. What counts is that all the nodes have the same parent element and are consecutive.

Using the keyboard	Using the mouse
<ul style="list-style-type: none"> <li>• <code>Esc Right_Arrow</code> (means: press <code>Esc</code> then press <code>Right_Arrow</code>) extends the selection to the following sibling of selected node.</li> </ul> <p>Note <code>Esc Right_Arrow</code> (and <code>Esc Left_Arrow</code>) will first select the element containing the caret if there is no explicit node selection, therefore typing <code>Esc Right_Arrow</code> several times is often the quickest way to select a node range.</p> <ul style="list-style-type: none"> <li>• <code>Esc Left_Arrow</code> extends the selection to the preceding sibling of selected node.</li> <li>• <code>Esc Down_Arrow</code> selects all the children of the selected element.</li> </ul>	<ul style="list-style-type: none"> <li>• <code>Ctrl+Shift-button1</code> extends the selection to the node clicked upon.</li> <li>• Clicking in the “interactive gray margin” (see <a href="#">above</a>) while pressing the <code>Shift</code> key extends the node selection to the block which is in front of the click location.</li> </ul>

## Cancel the text or node selection

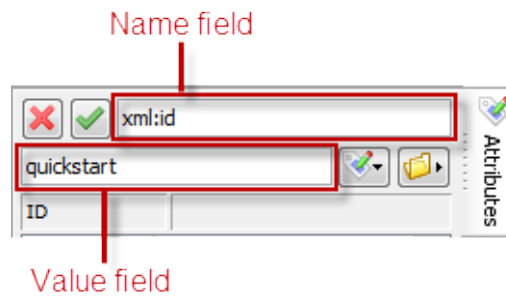
Using the keyboard	Using the mouse
<code>Esc Esc</code> (means: press <code>Esc</code> then press <code>Esc</code> again).	Click inside any text node to cancel the explicit selection.

## Quickly adding an attribute

---

The quickest way to add an attribute to an element is to proceed as follows:

1. Select the element.
2. Press `Ctrl-E` to move the keyboard focus to the **Name** field of the **Attributes** tool.



3. Type the name of the attribute in the **Name** field.

Thanks to the auto-completion feature of this field, typing the first few characters of an attribute name is sufficient to specify it.

4. Press `Enter` or `Tab` to move the keyboard focus to the **Value** field of the **Attributes** tool.
5. Type the value of the attribute in the **Value** field.

In many cases (ID, IDREF, enumeration, etc), this fields supports auto-completion too. However, unlike the **Name** field, the auto-completion feature here allows you to type anything you want, so you may have to press `Space` (which means: auto-complete as much as possible) in order to fully specify the value of the attribute.

6. Press `Enter` to commit the change and return the keyboard focus to the document view.

## Drag and drop

---

After selecting some text or some nodes, it's possible to drag this selection and drop it elsewhere inside or outside XMLmind XML Editor (XXE for short). Just remember that you'll have to press the `Alt` key while dragging the selection. Other than this, dragging works as expected.

Now if you drop some text —whether plain text or well-formed XML— inside XMLmind XML Editor, XXE will first attempt to insert this text at the drop location and if, this insertion is not allowed by the DTD or schema, it will then attempt to insert the text after the element which is found at the drop location.

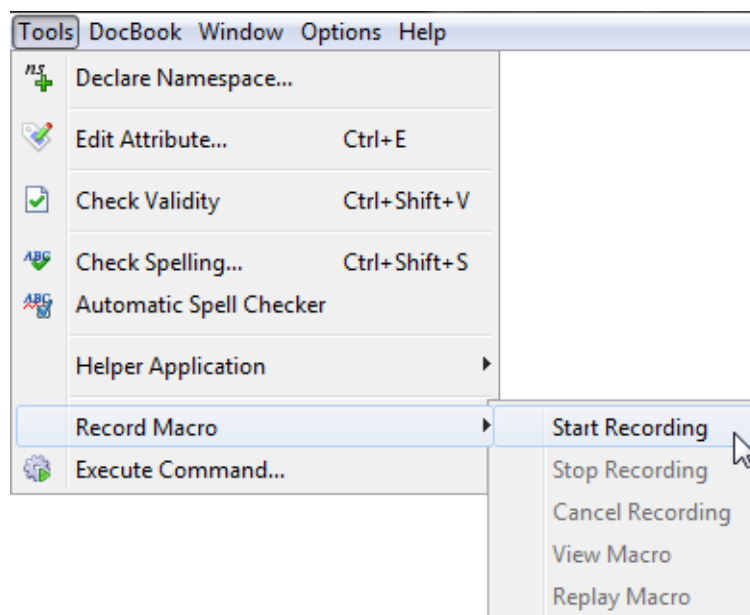
The behavior of XXE will differ if the text you are dropping can be parsed as an URL (e.g. `file:/C:/Program%20Files/XMLmind_XML_Editor/demo/dita-sample.ditamap`) or is the absolute path of an existing file (e.g. `C:\Program Files/XMLmind_XML_Editor\demo\dita-sample.ditamap`):

- If you drop it onto an image placeholder, this will open a dialog box allowing to change the source of the image element. See [Inserting images in your document](#).
- If you drop it onto an external link element (DocBook 5: `link xlink:href="xxx"`, DocBook 4: `ulink` element, DITA topic: `xref` or `link` elements, XHTML: `a href="xxx"` element), this will change the target of the link.
- If you drop it anywhere else, this will attempt to open the corresponding XML document in XXE.

## Automating repetitive tasks by recording macros

Let's suppose that in a poorly-tagged document, you want to replace dozens of `literal` elements containing word "XXE" by the equivalent `abbrev` elements. How would you do that using XMLmind XML Editor, given the fact that this application only has a text search/replace facility?

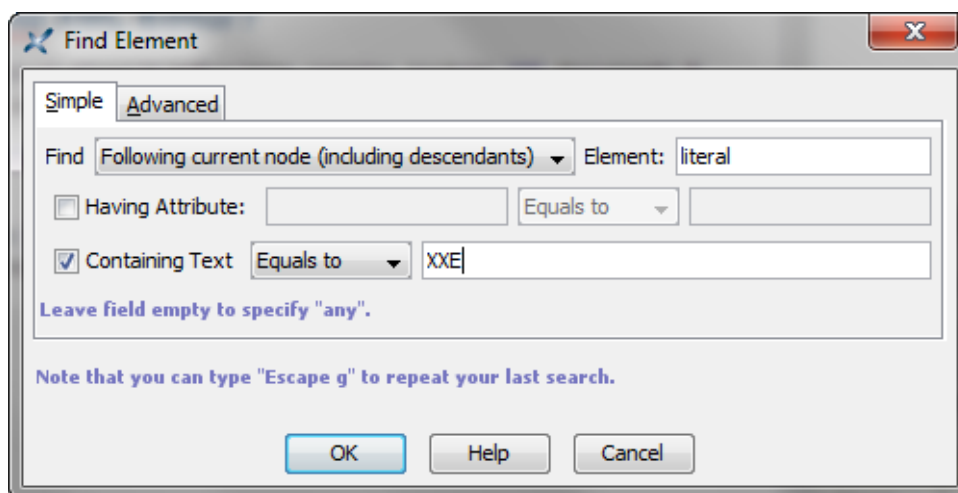
The answer is: perform the editing operation only once as you would do it normally, but make sure to record your actions using the **Tools**→**Record Macro** facility.



After the macro —that is, the sequence of commands you have invoked— has been recorded, it becomes possible to “replay” this macro as many times as needed to.

In the case of the above example, the sequence of commands to be recorded is:

1. Find next occurrence of a `literal` element containing word "XXE" using **Search**→**Find Element** (**Esc f**, **f** like **find**).



2. Use **Edit**→**Convert** (**Ctrl-T**) to convert the `literal` element to an `abbrev` element.

3. Press `Tab` to move the caret outside the `abbrev` element.

Once the macro has been recorded, you'll have to invoke it repeatedly until all the occurrences of `literal` element containing word "XXE" have been replaced by `abbrev` elements. This can be done by pressing `Esc p` (**p** like **play**), which is quicker than selecting menu item **Tools**→**Record Macro**→**Replay Macro**.

While recording a macro, you can use, not only keyboard shortcuts, but also any part of the user interface of XMLmind XML Editor: menu item, toolbar button, right-side pane, etc. This means that you can work almost normally while you are recording macro. There are notable exceptions though. Using any of the following commands will automatically cancel the recording of the macro: **Undo**, **Redo**, **Repeat** and *any command bound to a mouse click* (for example, double-click to select a word).

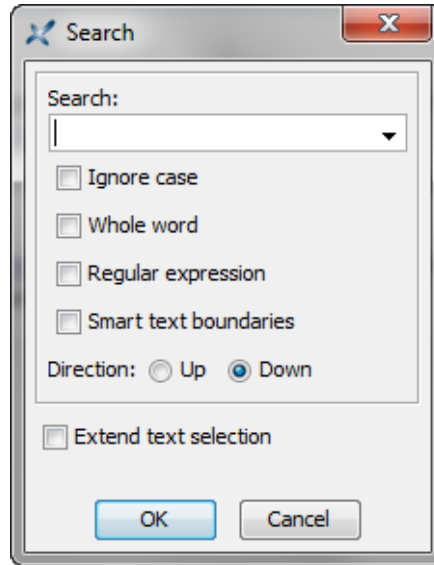
Also note that commands such as **File**→**Save** are simply not recorded because such commands do not contribute in modifying the document.

## Keyboard shortcuts

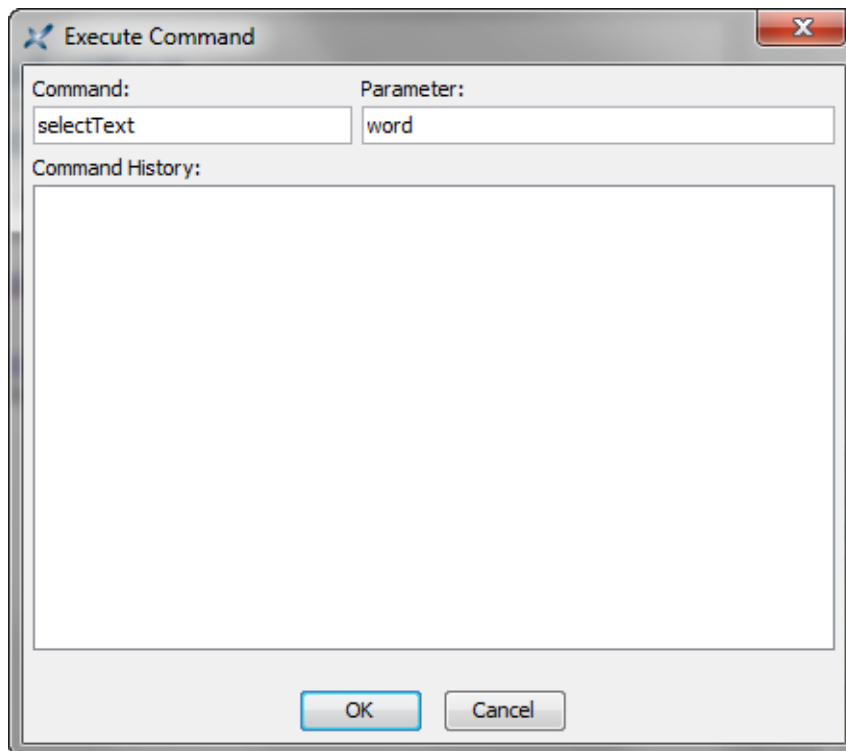
While it's certainly possible to click on menu items, toolbar buttons, etc, while recording a macro, this facility is meant to be used with the keyboard alone.

Keyboard Shortcut	Description
<code>Esc m</code> ( <b>m</b> like <b>macro</b> )	Start/stop recording a macro.
<code>Esc q</code> ( <b>q</b> like <b>quit</b> )	Cancel the recording.
<code>Esc p</code> ( <b>p</b> like <b>play</b> )	Execute ("replay") the macro.
<code>Esc f</code> ( <b>f</b> like <b>find</b> )	Find element.
<code>Esc s</code> ( <b>s</b> like <b>search</b> )	Find some text.  Displays a simple text search dialog box (see <a href="#">below</a> ) which is more convenient to use in the context of macro recording than the <b>Search</b> right-side pane.
<code>Esc x</code> ( <b>x</b> like <b>execute</b> )	Execute a command by specifying its name and its parameter. Example: execute command "selectText" with parameter "word".  The <b>Tools</b> → <b>Execute Command</b> dialog box (see <a href="#">below</a> ) is useful when there is no other way to execute a command, for example, when the command has no keyboard binding or when the command only has a mouse binding. (Remember that using your mouse inside the document view will cancel the recording of the macro.)  The commands which can be executed this way are all documented in <a href="#">XMLmind XML Editor - Commands</a> .

*The simple text search dialog box*



*The **Tools**→**Execute Command** dialog box*



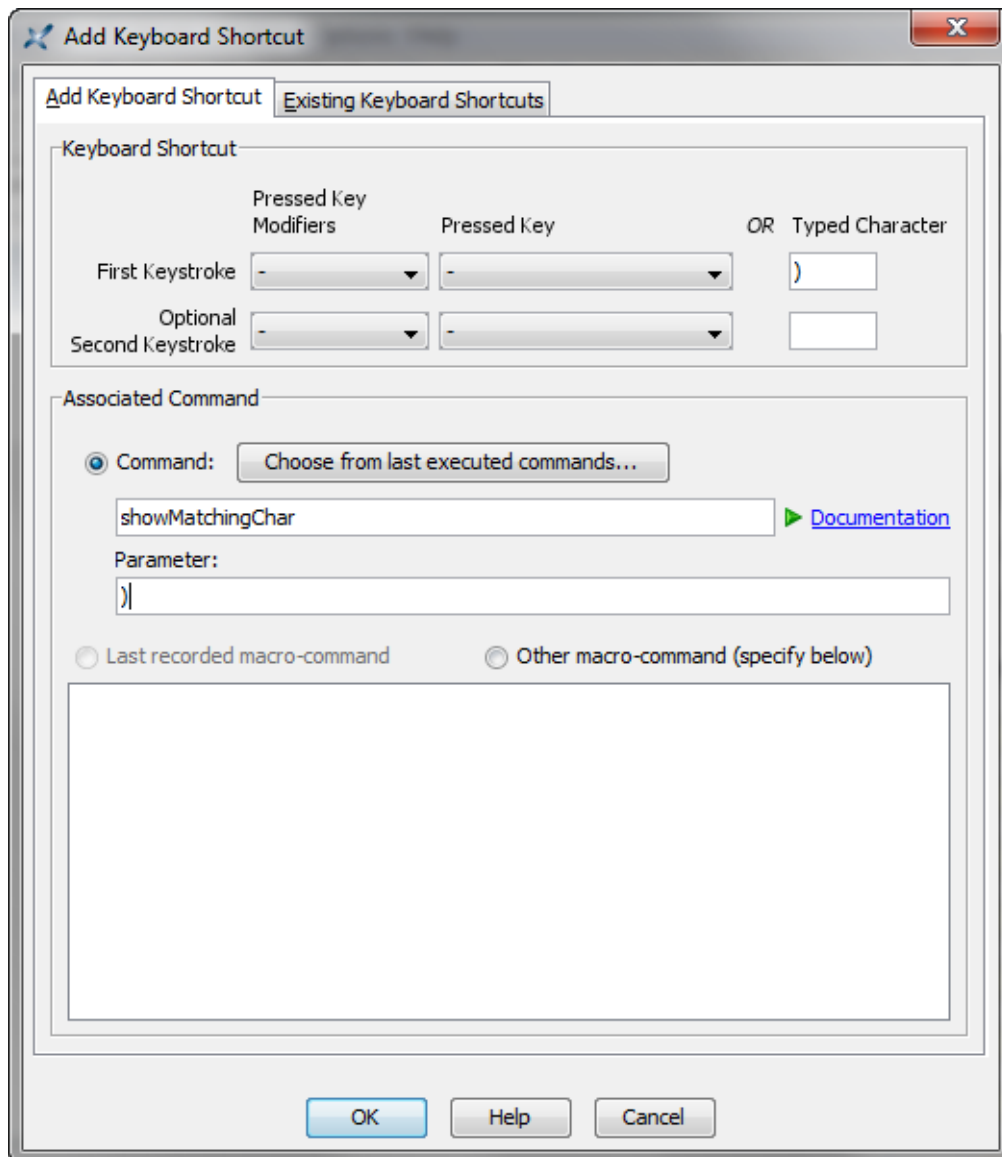
## Related productivity tips

- ▶ [Custom keyboard shortcuts](#)

## Custom keyboard shortcuts

You have learned in this [other lesson](#) that recording a macro allows to automate repetitive tasks. In some cases, you'll want the recorded macro to be remembered across editing sessions and you'll also want to trigger the execution of this macro by pressing a sequence of one or more keystrokes.

This can be done by selecting **Options**→**Customize Configuration**→**Add Keyboard Shortcut**. This menu item displays the following dialog box:

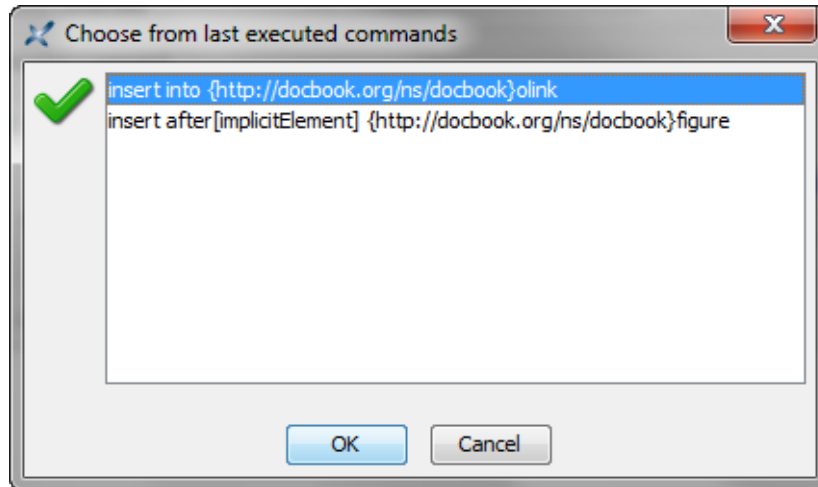


Using this dialog box is simple in its principle:

1. First specify a sequence of one or two keystrokes. A keystroke is specified by the name of a keyboard key (e.g. F7), optionally associated to a modifier key (e.g. Ctrl), or by a typed character (e.g. "}").
2. Then specify the command which is to be executed when the chosen keystrokes have been pressed.

This dialog box has a "**Last recorded macro-command**" option which allows to specify that the command to be executed is the last recorded macro-command.

Another easy to use option consists in binding the chosen keystrokes to one of the last executed commands. The **"Choose from last executed commands"** button displays the list of last executed commands (when an executed command is "repeatable").





More generally, you'll have to specify the name of a command (e.g. "insert") and also, optionally, the parameter—a text string—passed to this command (e.g. "into {http://docbook.org/ns/docbook}olink"). XMLmind XML Editor commands are all documented in [XMLmind XML Editor - Commands](#). This document is directly available from within the dialog box (▶ **Documentation**).


You'll find in the table below, some *very useful* examples of what you can do using this **Add Keyboard Shortcut** facility.

Keystroke	Command	Parameter	Description
- (minus)	insertCharSequence	- mdash	Typing a single "-" inserts a "-" character. Typing two "-" in a row inserts a "-" character (em dash).
` (backquote)	insertCharSequence	` 0x201c	Typing a single "`" inserts a "`" character. Typing two "`" in a row inserts a "``" character (opening quotation mark).
' (quote)	insertCharSequence	' 0x201d	Typing a single "'" inserts a "'" character. Typing two "'" in a row inserts a "''" character (closing quotation mark).
, (comma)	insertCharSequence	, 0x201e	Typing a single "," inserts a "," character. Typing two "," in a row inserts a "„" character (German opening quotation mark).
<	insertCharSequence	< 0x00ab	Typing a "<" single inserts a "<" character. Typing two "<" in a row inserts a "«" character (opening French guillemet).
>	insertCharSequence	> 0x00bb	Typing a ">" single inserts a ">" character. Typing two ">" in a row inserts a "»" character (closing French guillemet).
)	showMatchingChar	)	Inserts a closing parenthesis at caret position then, if the corresponding opening parenthesis is found, highlights this character for half a second. If the opening parenthesis is not found, this command emits an audio beep.
}	showMatchingChar	}	See above.
]	showMatchingChar	]	See above.

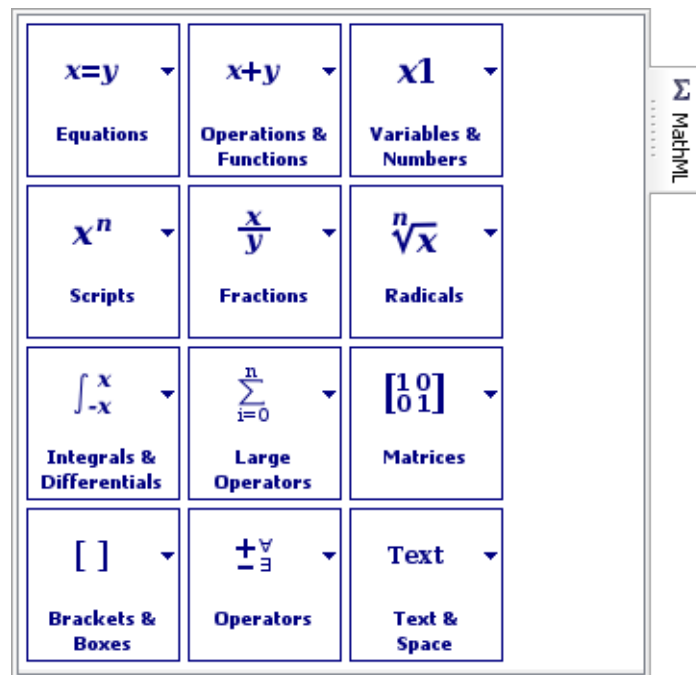
## Adding MathML equations

 Adding [MathML](#) equations to your documents requires installing the add-on called "*MathML support*". Installing an add-on is done by selecting menu item **Options**→**Install Add-ons**.

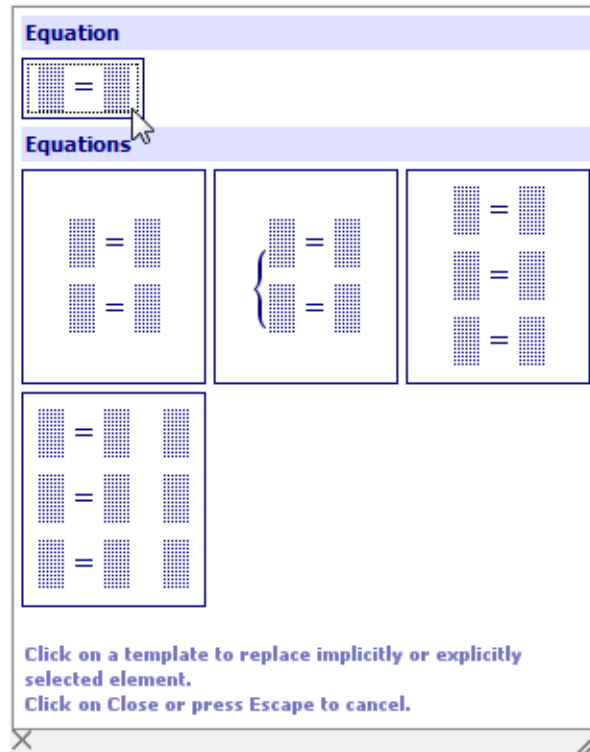
 Requires XMLmind XML Editor Professional Edition.

The  toolbar button allows to add an `m:math` top-level [MathML](#) element to your document. For now, this button is found in the toolbar only when you open a DocBook v5+ document or a DITA topic.

The newly inserted `m:math` element just contains an empty `m:mi` —identifier— element. Using the **MathML** tool, you'll be able to replace this placeholder by complex equations without knowing much about the MathML standard.

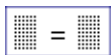


The **MathML** tool contains a number of palettes. Each palette contains a number of MathML templates.



Clicking on a template replaces the node selection by this template. Remember that the element containing the caret is implicitly selected. Therefore, generally, clicking on a template replaces the element containing the caret.

In practice, this means that using the **MathML** tool imposes you to construct your equation using a top-down approach. If you want to write  $x = y + z$ , you cannot write it naturally from left to right. You'll have to first insert the



template, then replace the left placeholder by another template, then replace the right placeholder by yet another template. This top-down approach is indeed slow to use but is also straightforward and easy to remember.

Of course, if you are a MathML expert, you are free to directly write your equations using the **Edit** and the **Attributes** tools. Because XMLmind XML Editor natively supports MathML presentation markup, MathML elements are not treated differently than say, a paragraph or a list item.


The placeholders contained in all MathML templates always are `m:mi` —identifier— elements. Let's suppose you

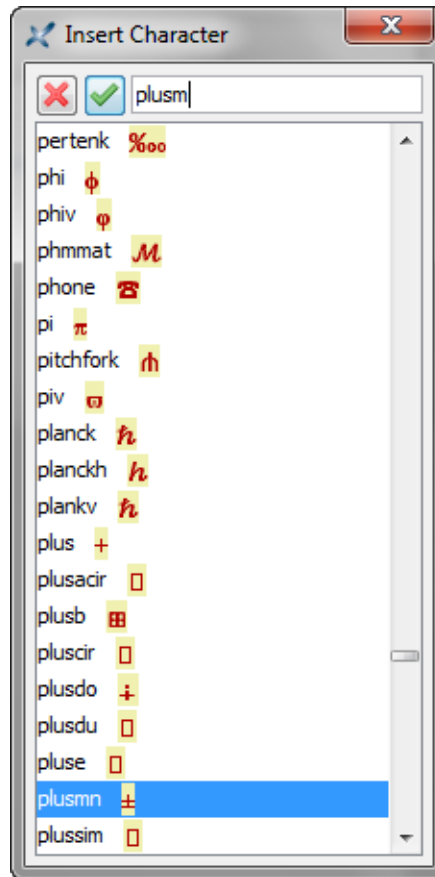


have inserted the `m:mi` template because you want to write  $A/3.14$ . The top placeholder being an `m:mi`, simply type "A" in it. But you'll first replace the bottom `m:mi` placeholder by a `m:mn` —number— element before typing "3.14" in it.

Now let's suppose you want to write  $-A/3.14$ . You cannot directly type "-A" in the top placeholder because "-" is an operator. You'll first replace the top `m:mi` by a template containing an `m:mo` —operator— element followed by a `m:mi`.

*The distinction between `m:mi` (identifier), `m:mn` (number) and `m:mo` (operator) is the only point you have to remember about MathML in order to write correct equations using the **MathML** tool.*

What if you need an operator (e.g. "±") not found in any template of the **MathML** tool? In such case, click inside the `m:mo` you want to populate and use **DocBook** and DITA **Topic** menu item  **"Insert MathML Character by Name"**.



One of the most commonly used operators is called `InvisibleTimes`. For example, it's found between  $m$  and  $c$  in the famous formula:  $E = mc^2$ . This operator is peculiar because normally it's invisible. However for easier editing, XMLmind XML Editor has made this operator visible:  $E = m \cdot c^2$ .

# Easily create DocBook olinks

---

## A refresher about olinks

The DocBook `olink` element allows to create a link between two different documents. For example, it allows to create a link in document `folder1/Article1.xml` pointing to the section having `"sectionA"` as its ID which is found in document `folderA/ArticleA.xml`.

The `olink` element used to represent the aforementioned link is:

```
<olink targetdoc="articleA" targetptr="sectionA"></olink>
```

- Notice that the target document is specified by a symbolic name, "articleA", rather than by its URL, "folderA/ArticleA.xml".
- The value of the `targetptr` attribute is simply the ID of the target element as contained in the target document. This attribute is optional. Without it, the `olink` implicitly points to the root element of the target document.
- An `olink` can optionally contain some text. When this text is absent, the [DocBook XSL stylesheets](#) automatically use the content of the `title` child element of the target element.

Where to find the symbolic name of each document involved in linking? The DocBook XSL stylesheets requires such documents to be declared in a special XML file called a *sitemap*. The sitemap file describes the directory structure of your HTML or PDF output tree. You must declare the symbolic names of your documents in this file. Example:

```
<!DOCTYPE targetset
  SYSTEM "file:/opt/xxe/addon/config/docbook/xsl/common/targetdatabase.dtd" [
  <!ENTITY article1 SYSTEM "folder1/target.db">
  <!ENTITY articleA SYSTEM "folderA/target.db">
]>
<targetset>
  <sitemap>
    <dir name="docs">
      <dir name="folder1">
        <document targetdoc="article1">
          &article1;
        </document>
      </dir>

      <dir name="folderA">
        <document targetdoc="articleA">
          &articleA;
        </document>
      </dir>
    </dir>
  </sitemap>
</targetset>
```


It is customary to use the ID of the root element of a document as its symbolic name.

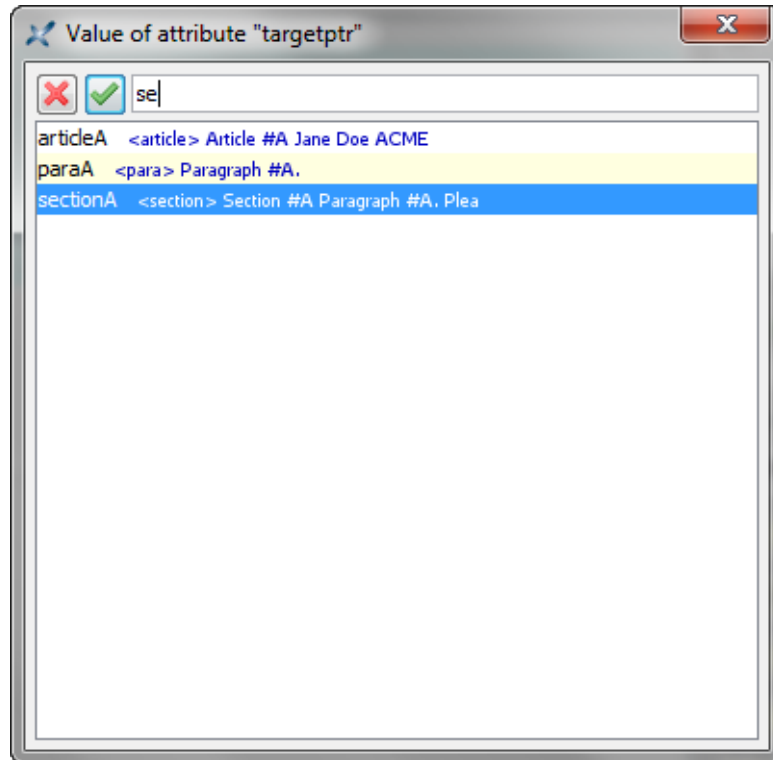
Processing a set of olinked documents using the DocBook XSL stylesheets is not a simple task and we'll not attempt to describe it in this tutorial. For more information please refer to [DocBook XSL: The Complete Guide](#) by Bob Stayton.

## How XMLmind XML Editor can help you

XMLmind XML Editor does not offer any special authoring tool which could help you in inserting `olink` elements in your document. You'll have to use the **Edit** tool to insert an `olink` element at caret position. Then you'll have to use the **Attributes** tool to specify its `targetdoc` attribute and optionally, its `targetptr` attribute.

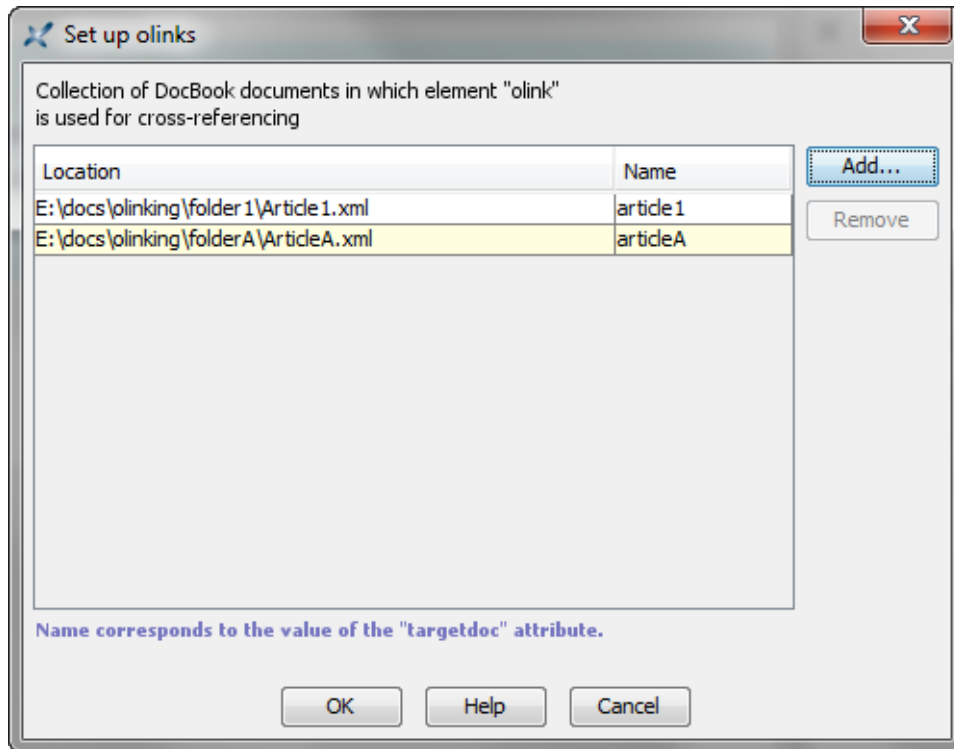
Obviously, specifying the values of the `targetdoc` and `targetptr` attributes without any assistance is tedious and error-prone. That's why, if and only if you first use **DocBook**→**Set up olinks** to declare all your DocBook documents (XMLmind XML Editor cannot read your sitemap file):

1. You'll be able to use the autocompletion facility of the **Attributes** tool when you'll specify the values of the `targetdoc` and `targetptr` attributes.
2. The  **Edit** button which is next to the attribute value field will display chooser dialog boxes specialized in `olink` attributes.

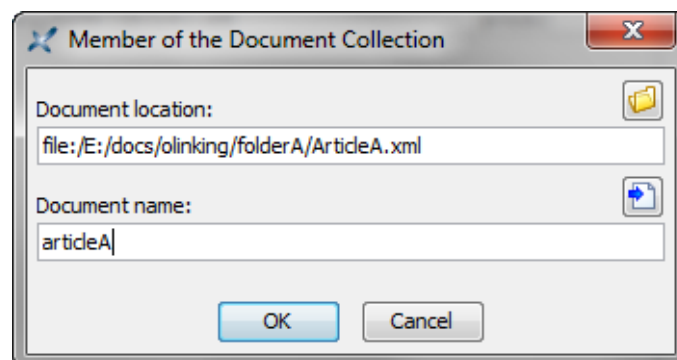



## Set up olinks

The **DocBook**→**Set up olinks** menu item displays the following dialog box:




Basically this dialog box allows to create a list of DocBook 4 and/or DocBook 5 documents., each document being specified by its URL (e.g. "file:/E:/docs/olinking/folderA/ArticleA.xml") and its symbolic name (e.g. "articleA"). This list is specified once for all.



Make sure to specify the same symbolic name as the one found in your sitemap. If, by convention, your organization always uses the ID of the root element of a document as its symbolic name, then suffice to click the  button.

## Reviewing changes using the Compare tool

 The **Tools**→**Changes** submenu is hidden by default. You need to enable it by checking "**Enable the 'Tools|Changes' Submenu**" in **Options**→**Preferences, Features** section.

 Requires XMLmind XML Editor Professional Edition.

### Why use the Compare tool?

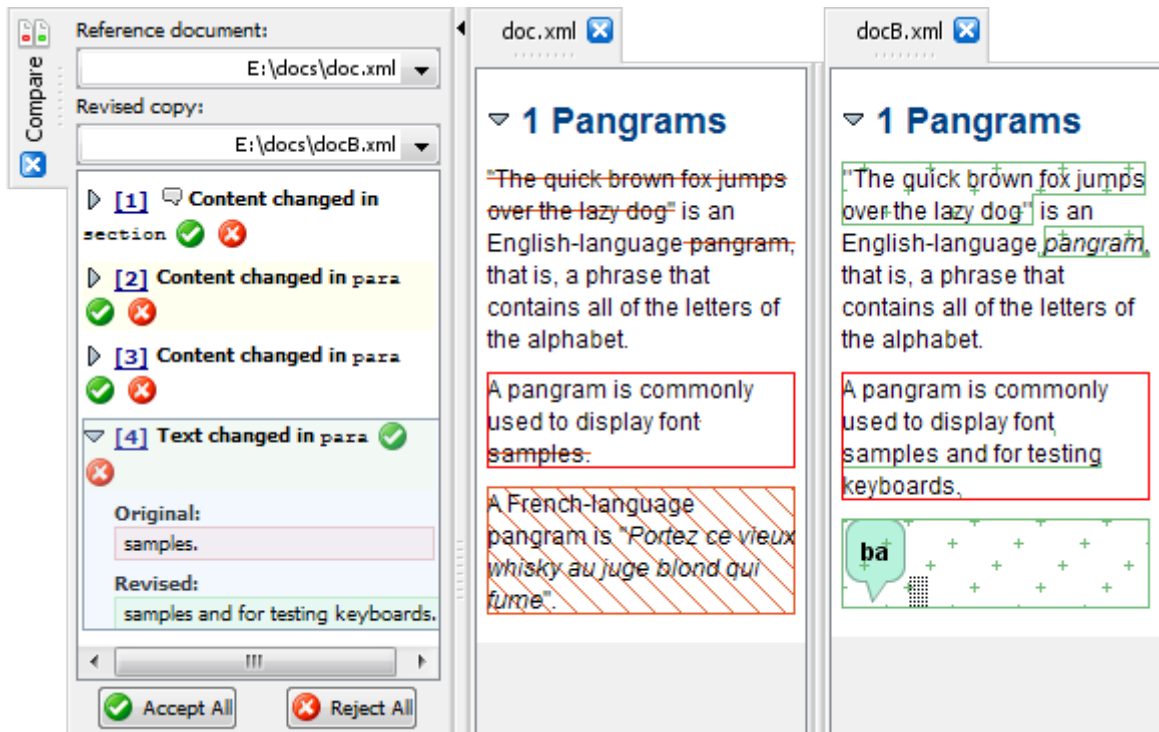
The [Compare tool](#) allows to compare two revisions of the same initial document.

Unlike generic XML comparison tools,

- you must explicitly activate the change detection in the initial document if you want to be able to compare two revisions using the **Compare** tool. This activation is made by selecting menu item **Tools**→**Changes**→**Activate Change Detection**.
- Moreover adding or deleting elements using a text or XML editor other than XXE will cause the change detection to be automatically deactivated the next time you'll reopen the document in XXE. Any other kind of change, for example text or attribute changes, poses no problem.

After selecting menu item **Tools**→**Changes**→**Compare Revisions** in order to display it, the **Compare** tool will show you the differences existing between two revisions. It also allows to accept or reject some or all the changes.

The **Compare** tool showing the differences existing between `doc.xml` (the initial document) and `docB.xml` (a revision of `doc.xml`), displayed side by side



## A simple use case

This tool is typically used by an author after her/his draft document has been reviewed and possibly modified by other authors. For example, let's suppose that John has finished writing `doc.xml`.

John activates the change detection in `doc.xml` using **Tools→Changes→Activate Change Detection**, saves the document to disk and then sends a copy to Bart.

Bart modifies his copy. He also adds a *remark* using **Tools→Remark→Insert or Edit Remark** explaining why he deleted the last paragraph. This results in creating `docB.xml`, a revision of `doc.xml`.

Note that Bart didn't even notice that the change detection has been activated in the copy of `doc.xml` he has received. The use of this facility should be completely transparent to the user of XXE, even in terms of perceived performances.

After the review of the document by Bart is finished, John receives `docB.xml`.

John first compares `doc.xml` to `docB.xml`, accepts all the changes except the commented deletion made by Bart, then saves `docB.xml` to disk.

Finally John renames the “approved” `docB.xml` to `doc.xml`.