

XMLmind XML Editor Tutorial

Hussein Shafie

XMLmind Software

35 rue Louis Leblanc,

78120 Rambouillet,

France,

Phone: +33 (0)9 52 80 80 37,

Web: www.xmlmind.com/xmleditor

Email: xmleditor-support@xmlmind.com

May 21, 2025

Table of Contents

Before following this tutorial	ii
Getting started	1
Creating a document	2
Basic editing	5
Easier editing	7
Copy, Cut, Paste and Delete	9
Replacing nodes	11
Converting elements	12
Setting attributes	14
Inserting special characters	17
Inserting images	19
Editing tables	21
Modular documents	23
Creating a modular document	24
Inserting boilerplate content	26
Working with master documents	28
Becoming productive	32
Custom document templates	33
Quickly paste selected text using mouse button #2	34
Inserting custom element templates	35
101 ways to select nodes	37
Quickly adding an attribute	39
Drag and drop	40
Automating repetitive tasks by recording macros	41
Custom keyboard shortcuts	45
Specialized tools	48
Creating a DITA bookmap	49
Adding MathML equations	51
Easily create DocBook olinks	55
Reviewing changes using the Compare tool	59
Conditional processing made easy	62

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^[1]. 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](#) 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](#), [\(X\)HTML5](#), 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.

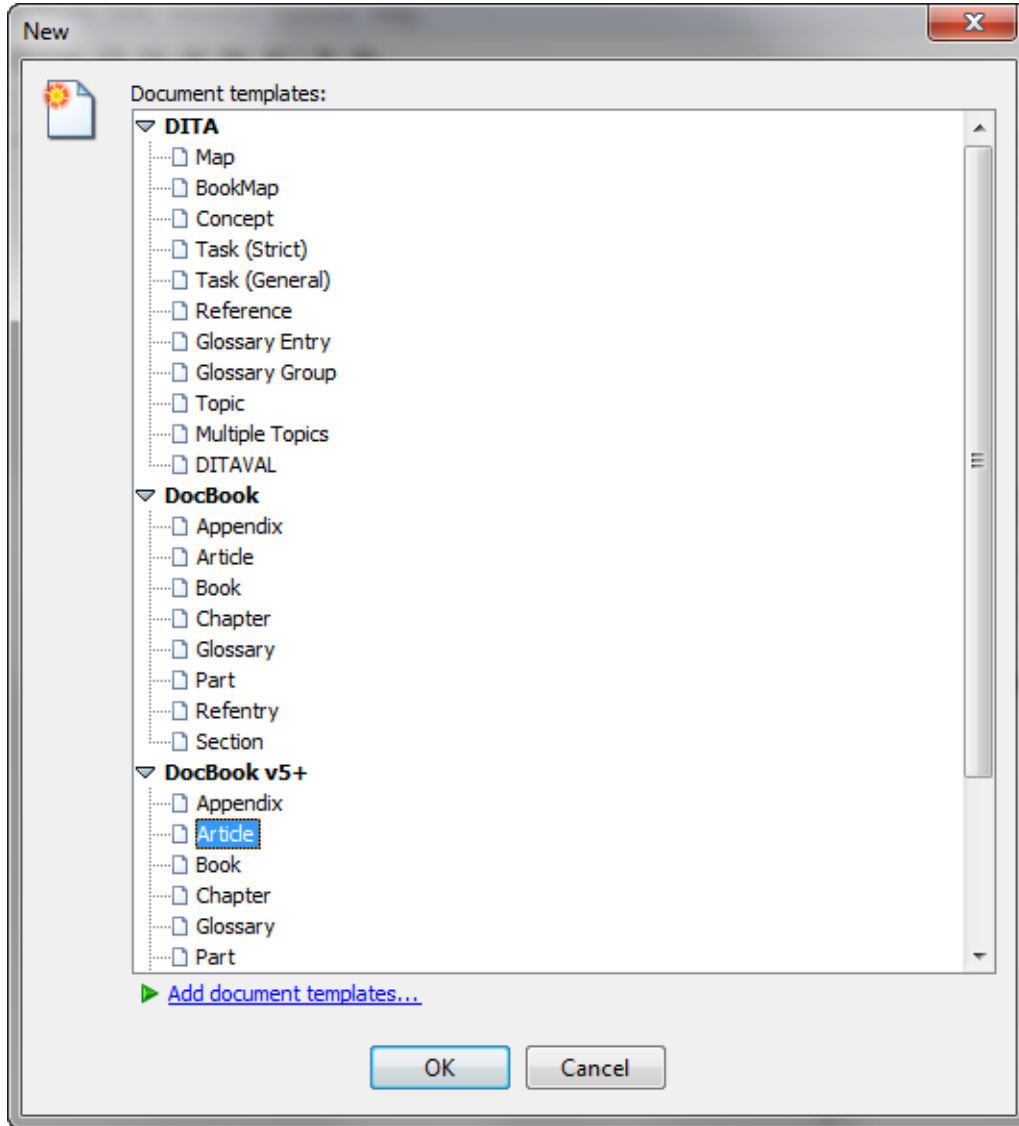
If you find that this tutorial is too comprehensive and don't have enough time to follow it, we nevertheless recommend to read this [minimal tutorial](#) (corresponding [screencast](#)).

^[1]Most screencasts have been created using older versions of XMLmind XML Editor. Tabs and toolbar buttons look quite different now. However what is shown in screencasts still applies to recent versions of the application.

Getting started

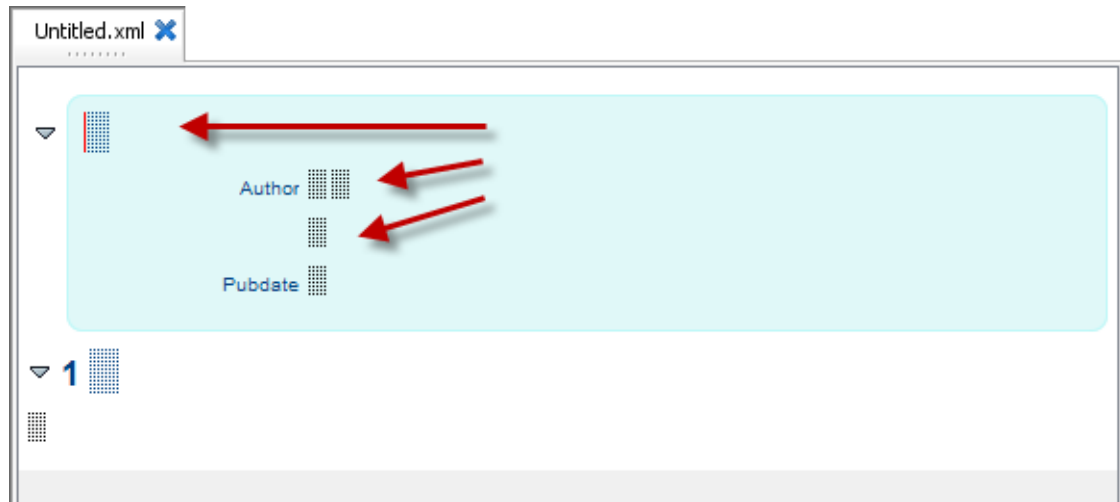
Creating a document

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**.

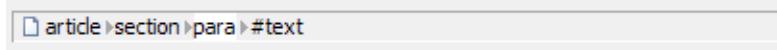
Let's suppose you want to create a DocBook 5 article. Here's what you'll see at first:



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.

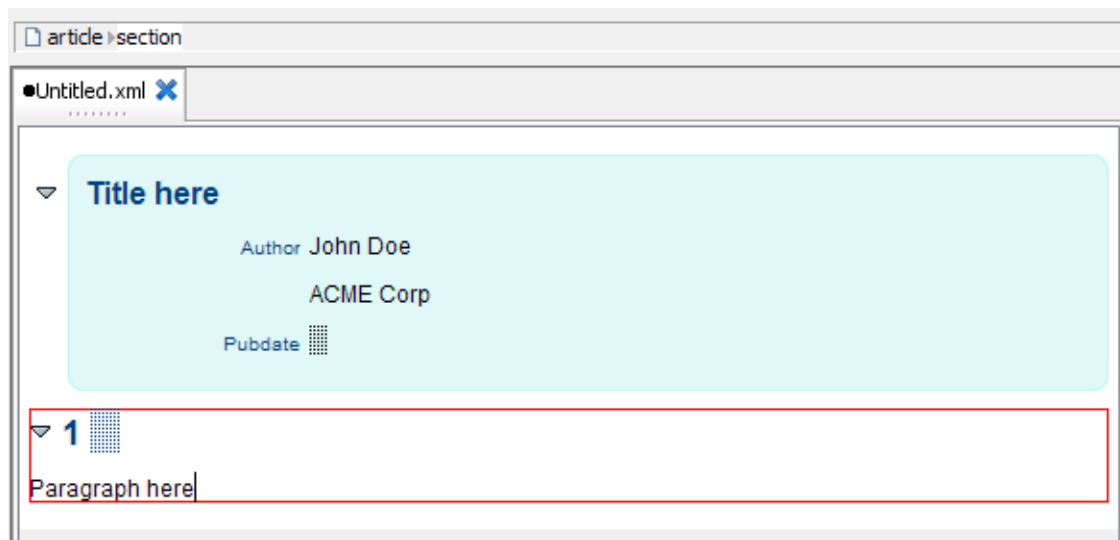
These tag-less placeholders do not convey any meaningful information, so how can you understand what you see? The answer is: look at the *node path bar*. The node path bar shows you which XML node is selected and which are its ancestor elements.



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.






 [Watch the screencast](#)

Related productivity tips

- [Custom document templates](#)

Basic editing

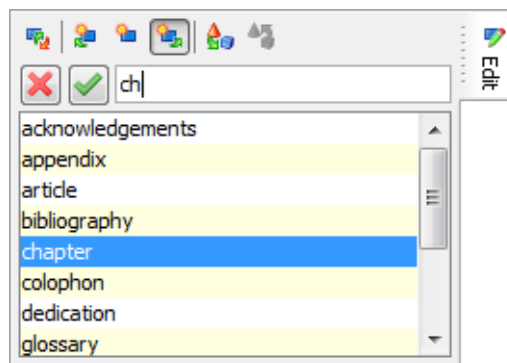
In order to add some content to your XML document, you must:

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, *right here, 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 *just 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 *just after selected node*.

Note that *XMLmind XML Editor does not work like other XML editors*. In other XML editors,

- **Insert** often means "Insert *somewhere* inside selection".
- **Insert Before** often means "Insert *somewhere* before selection".
- **Insert After** often means "Insert *somewhere* after selection".

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**.

 [Watch the screencast](#)

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,
- “text style” toggles found in the toolbar,
- 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.

“Text style” toggles found in the toolbar


The XHTML, DocBook and DITA toolbars all start with a number of “*text style*” toggles. These toggles emulate the behavior of the **Bold**, **Italic**, **Underline**, etc, toggles found in the toolbars of almost all word-processors.




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


Repeating a command

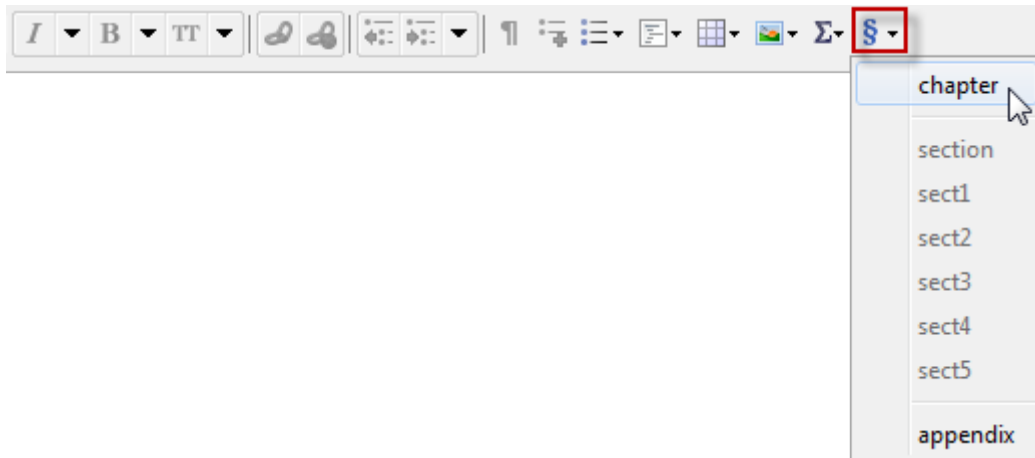
Once you have, for example, converted the text selection to an `emphasis` element having a `role="bold"` using the **emphasis[bold]** toggle, you can select another piece of text and convert it too to **emphasis[bold]** simply by pressing **Ctrl-A** (A like **Again**).

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

The other buttons of the toolbar allow to add commonly used elements. For example, toolbar button  allows to add a list item whatever the kind of this 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 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.

A handy alternative is to press **Ctrl-Enter** while the caret is anywhere inside a 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 not only inside paragraphs, but also inside *list items* and a few other kinds of “block” (e.g. DocBook v5+ bridgehead element). Pressing **Enter** inside a block 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.

 [Watch the screencast](#)

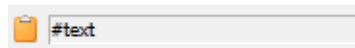
Copy, Cut, Paste and Delete


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**).



Watch the screencast

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.




Watch the screencast

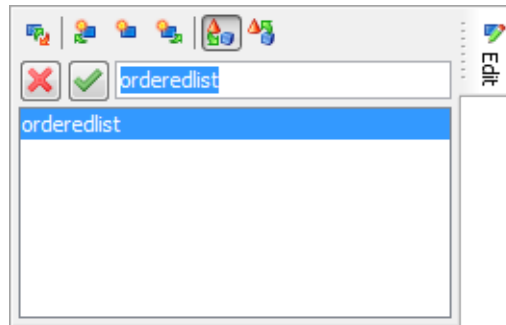
Related productivity tips

► [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 commands based on  **Edit**→**Convert** (keyboard shortcut: **Ctrl-T**, T like **T**ransform). 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.

 [Watch the screencast](#)

Related productivity tips

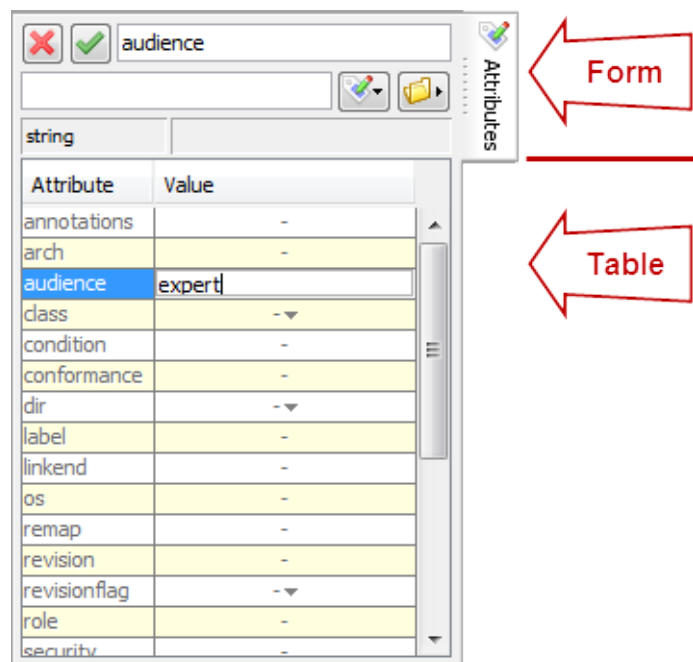
- ▶ [101 ways to select nodes](#)

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.



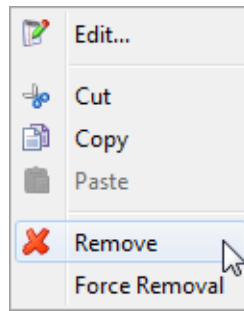
The attributes table

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**.



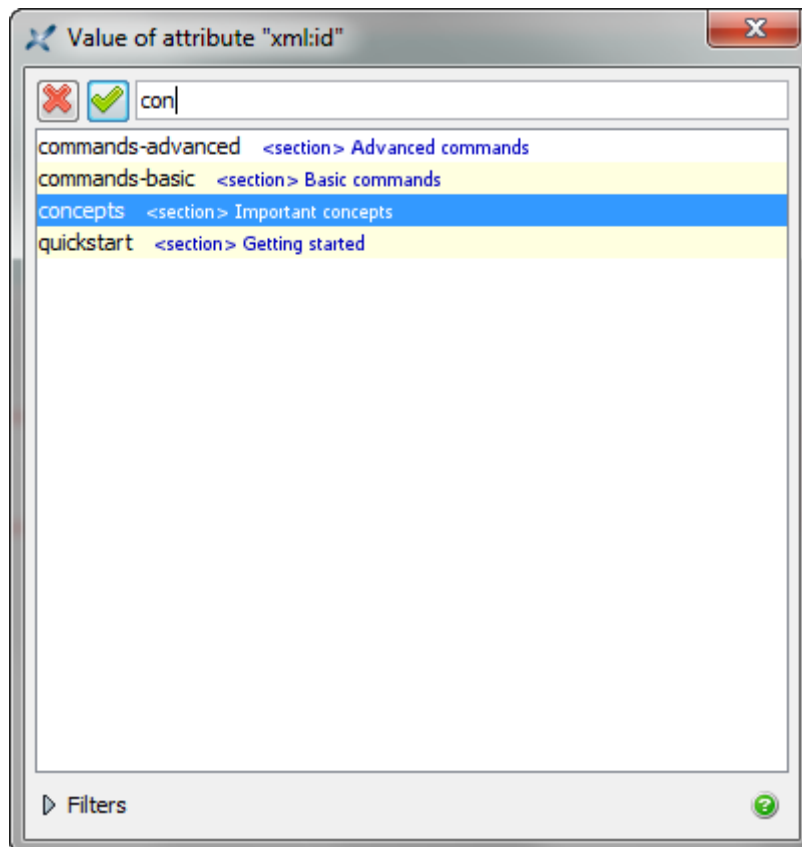
The attribute form

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  "Choose file" button.

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

 [Watch the screencast](#)

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:

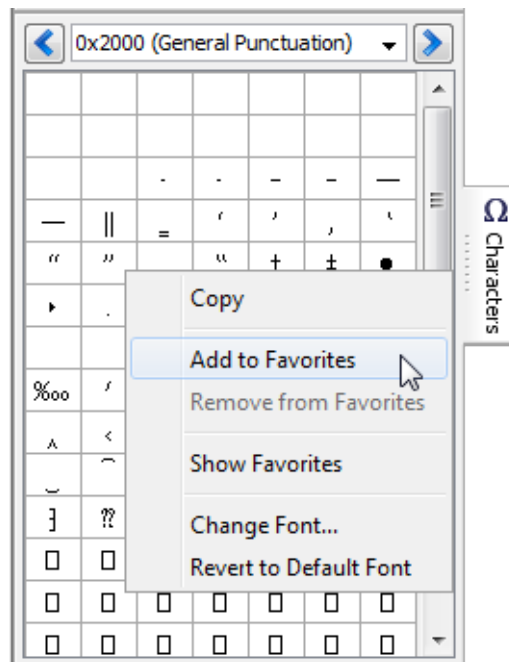
1. Directly type the special character (if this is possible) or copy it from a third-party application (such as Windows **Character Map** utility), then paste it in the XML document.

Note that the non-breaking space (Unicode code point: U+00A0; corresponds to well-known character entity ` `), 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.

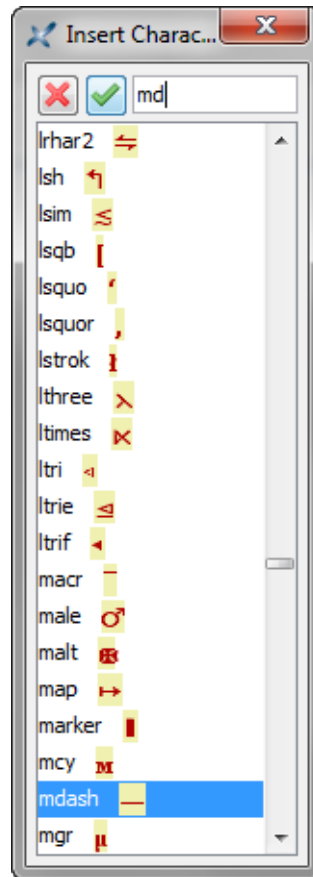
Popular operating systems are Mac OS-X and Windows XP.

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 **A**gain) 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.



[Watch the screencast](#)

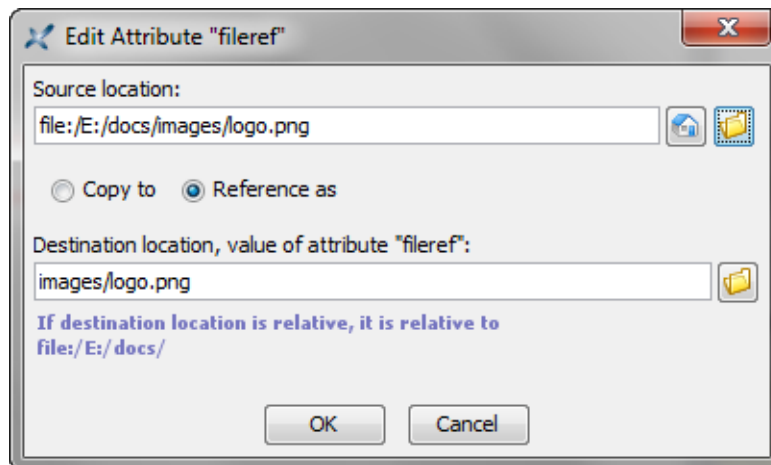
Inserting images

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 equivalently, right-click on the image placeholder, then select "**Set Image...**" from the contextual menu.)
- 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.

 [Watch the screencast](#)

Related productivity tips

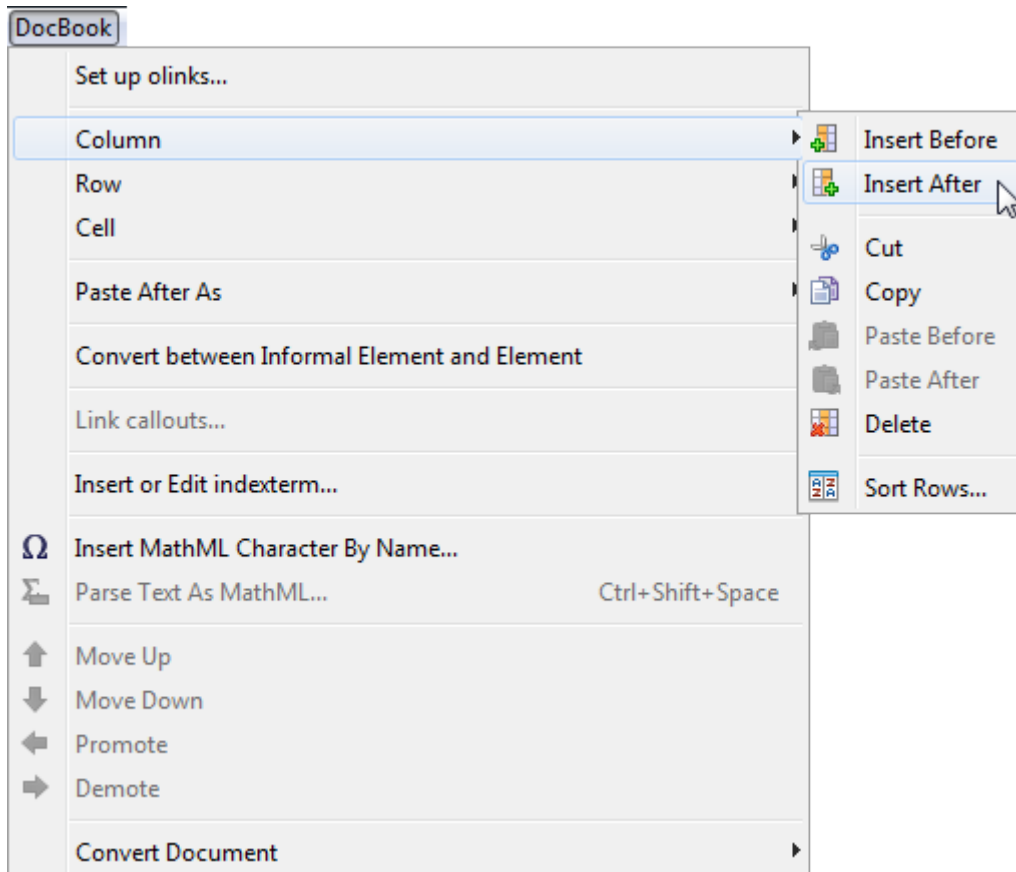
► Drag and drop

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.

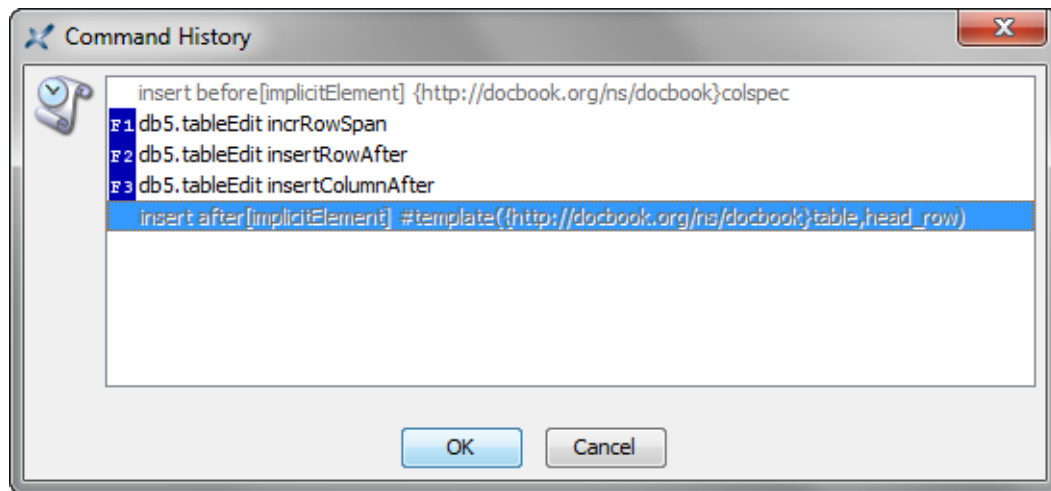


Note that precisely adjusting the presentation of a table may require you to use the **Edit** tool and the **Attributes** tool. For example, making the third column of a DocBook table twice as wide as the other columns requires you to insert a `colspec` element before the `thead` or `tbody` of the `tgroup` and then specify attributes `colnum="3"` and `colwidth="2*"` for the new `colspec`. However, when a precise column width is not required, suffice to *drag the column separator* as you would do it in any word-processor.

 [Watch the screencast](#)

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

Modular documents



Creating a modular document

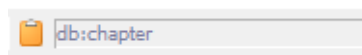
XMLmind XML Editor allows to include in document *B* some content found in document *A*. This feature works by using a special command called **Copy As Reference** and then by using one of the normal **Paste** commands. Unlike the ordinary **Copy/Paste** commands which duplicate content, **Copy As Reference** 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.

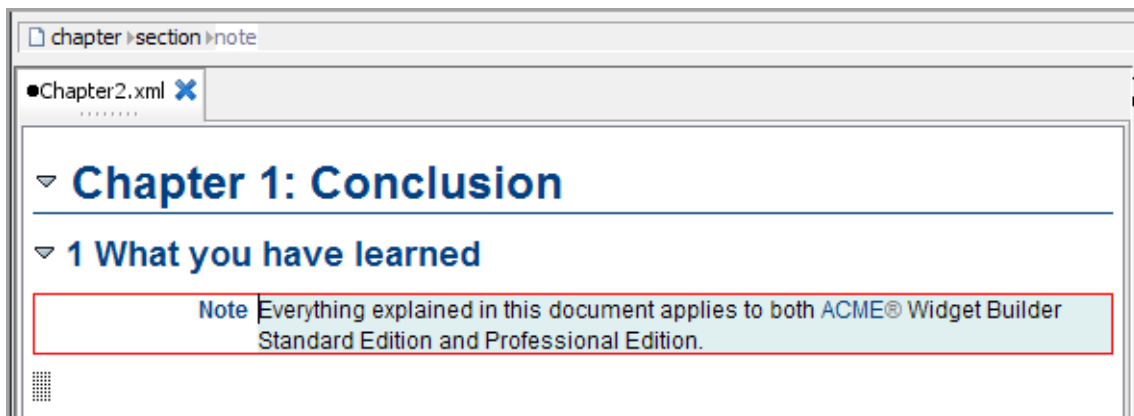
This feature is also used to include “boilerplate” content in a document: company name, product name, copyright statement, etc. More about this in next lesson: [Inserting boilerplate content](#).

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


From there, switching back to the included content is done by using menu item .

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



Watch the screencast

Inserting boilerplate content

 The **Include** tool is hidden by default. To display it, you need to use **Options→Preferences, General|Features** section and check "**Enable the Include tool**".

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.

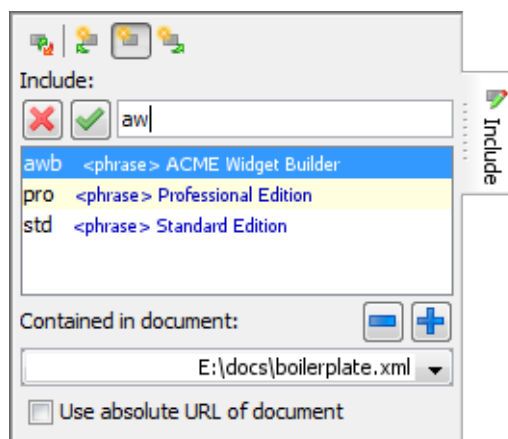
Remember that **Copy As Reference** works only if the node selection is a *single element having an ID*. So what to do if you just want to include some text, for example, the name of your product? In such case, you'll have to wrap this text in an element. For that, choose an element which has no specific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



[Watch the screencast](#)

Working with master documents

What is a master document?

The following documents may be used as *master documents*: DITA map, DocBook 5.1 [assembly](#), DocBook modular documents (for example, a book including chapters by the means of [XInclude](#)).

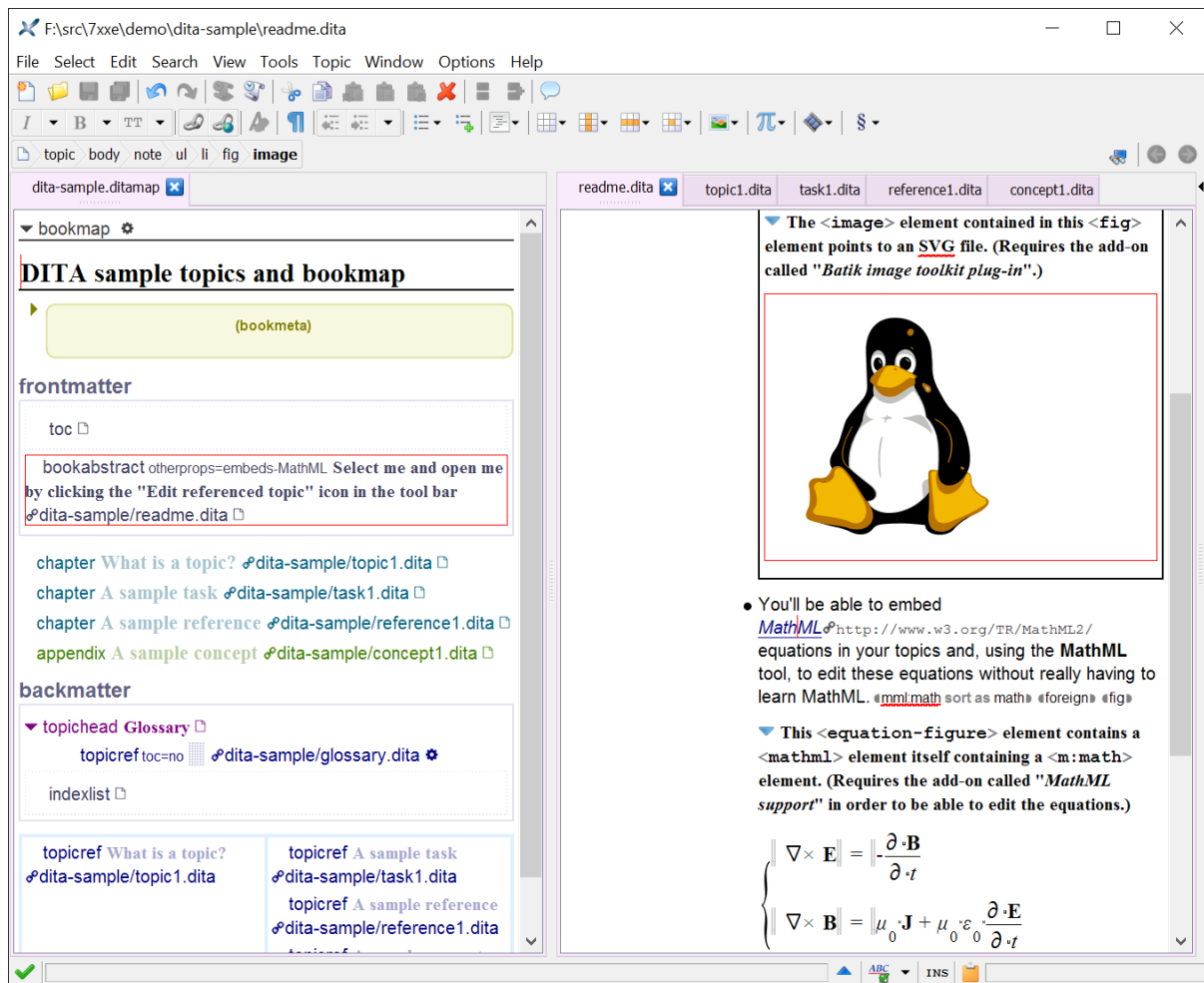
In order to use a document as a master document, simply open this document and check

Tools→Use as Master Document. This is done once for all as this choice is is persistent across editing sessions.

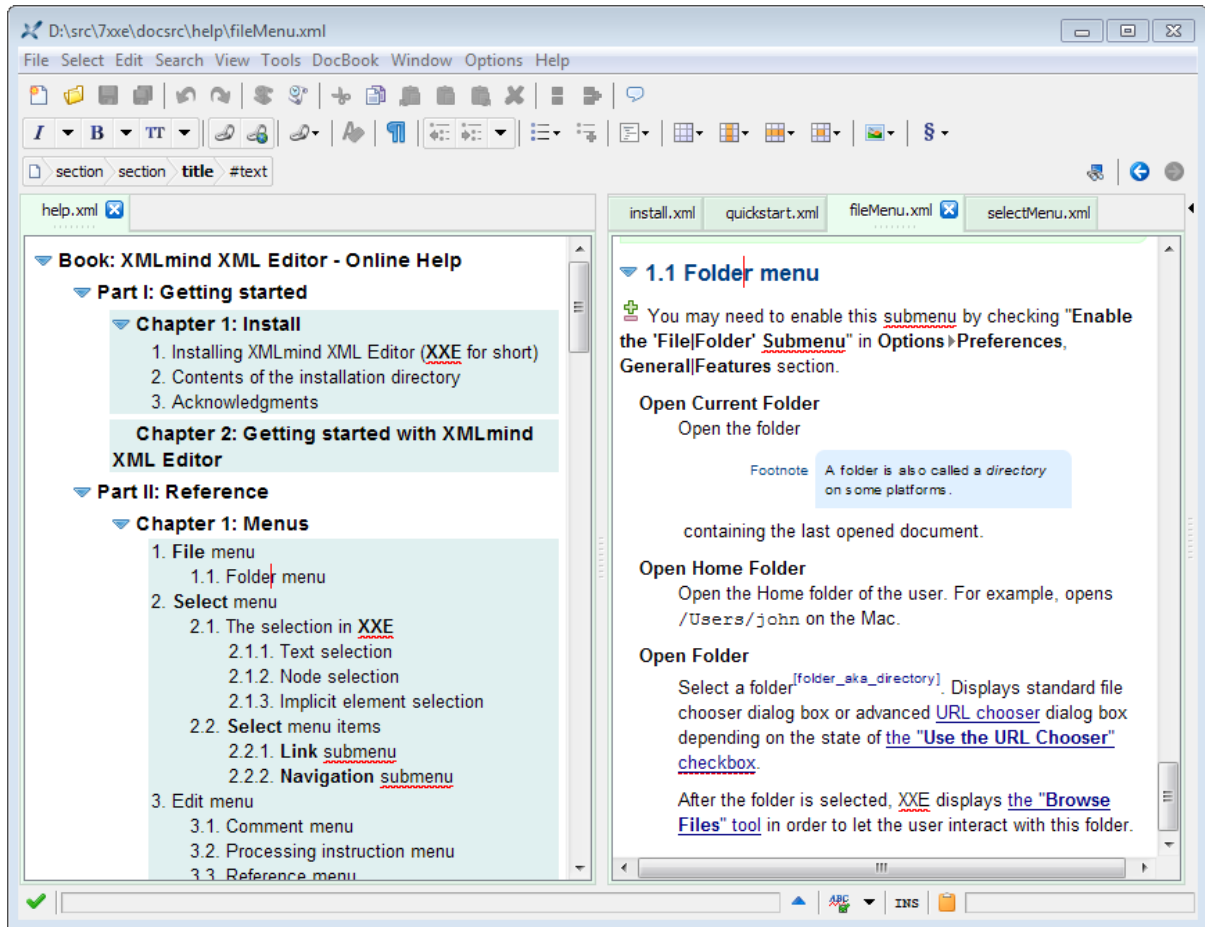
Declaring a document as being a master document creates a *document set* containing the master document and its module document. This document set is automatically updated when the master document is saved to disk.

Notice in the screenshots below that **XXE** gives the same distinctive color to the tabs containing the master document and its module documents. This makes it easier for the user to spot the members of a document set.

A DITA map (master document) and topics (module documents) side by side



Modular DocBook book styled using the "Table of contents" stylesheet (master document) and its chapters (module documents) side by side



Why check "Use as Master Document"?

By grouping a master document and its module documents, you inform XMLmind XML Editor (XXE) that all the module documents referenced or included, directly or indirectly, by the master document are related. When XXE knows that some of the opened documents are related:

- it will more thoroughly check the cross-references which may exist between these documents;
- it will make it easier creating cross-references between these documents;
- it will make it easier following cross-references between these documents;
- if a DITA map contains [key definitions](#), then this map acts not only as a cross-reference creation/validation context for its topics, but it also acts as a key space;
- if the **"Easy Profiling"** add-on has been installed, then the conditional processing profile selected for the master document is automatically shared by all module documents;
- if the document view area is split in two parts, the module documents opened from a master document will appear at the opposite of this master document. This allows to use the view of the master document as a rudimentary navigation pane.

Use case explaining the enhanced cross-reference creation/validation provided by a master document

DITA map mymap.ditamap references 3 topics: topicA.dita, topicB.dita and topicC.dita.

File topicA.dita contains:

```
<topic id="topicA">
  <title>Topic A</title>

  <body>
    <section id="sectionA1"><title>Section A1</title><p>TODO</p></section>

    <section id="sectionA2"><title>Section A2</title><p>TODO</p></section>
  </body>

  <related-links>
    <link href="nowhere.dita">
      <linktext>Link to nowhere</linktext>
    </link>

    <link href="topicB.dita#topicB/sectionB1">
      <linktext>Link to Section B1</linktext>
    </link>
  </related-links>
</topic>
```

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

The user wants to check the links found in topicA.dita and also to add an xref element pointing to the first section of topicC.dita. In order to do that, she/he opens topicA.dita in **XXE**.

Before using mymap.ditamap as a master document:

- The **Validity** tool does not report any error, even for `<link href="nowhere.dita">`.
- When the user inserts an xref element and specifies its href attribute, **XXE** suggests just: `#topicA`, `#topicA/sectionA1`, `#topicA/sectionA2`.

After using mymap.ditamap as a master document:

- The **Validity** tool now reports 1 cross-reference warning: "nowhere.dita" does not belong to this document set.
- When the user inserts an xref element and specifies its href attribute, **XXE** now suggests: `#topicA`, `#topicA/sectionA1`, `#topicA/sectionA2`, `topicB.dita`, `topicB.dita#topicB/sectionB1`, `topicB.dita#topicB/sectionB2`, `topicC.dita`, `topicC.dita#topicC/sectionC1`, `topicC.dita#topicC/sectionC2`.



Watch the screencast

Related tutorials

- Conditional processing made easy

Becoming productive

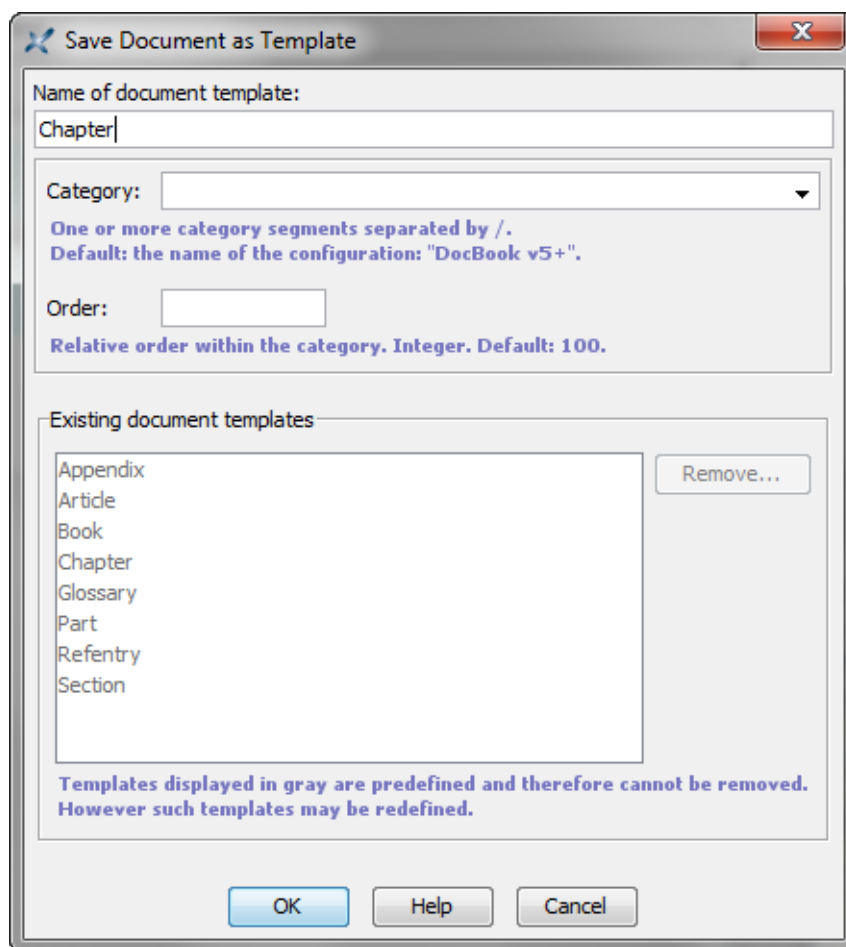
Custom document templates

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.


 [Watch the screencast](#)

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.)

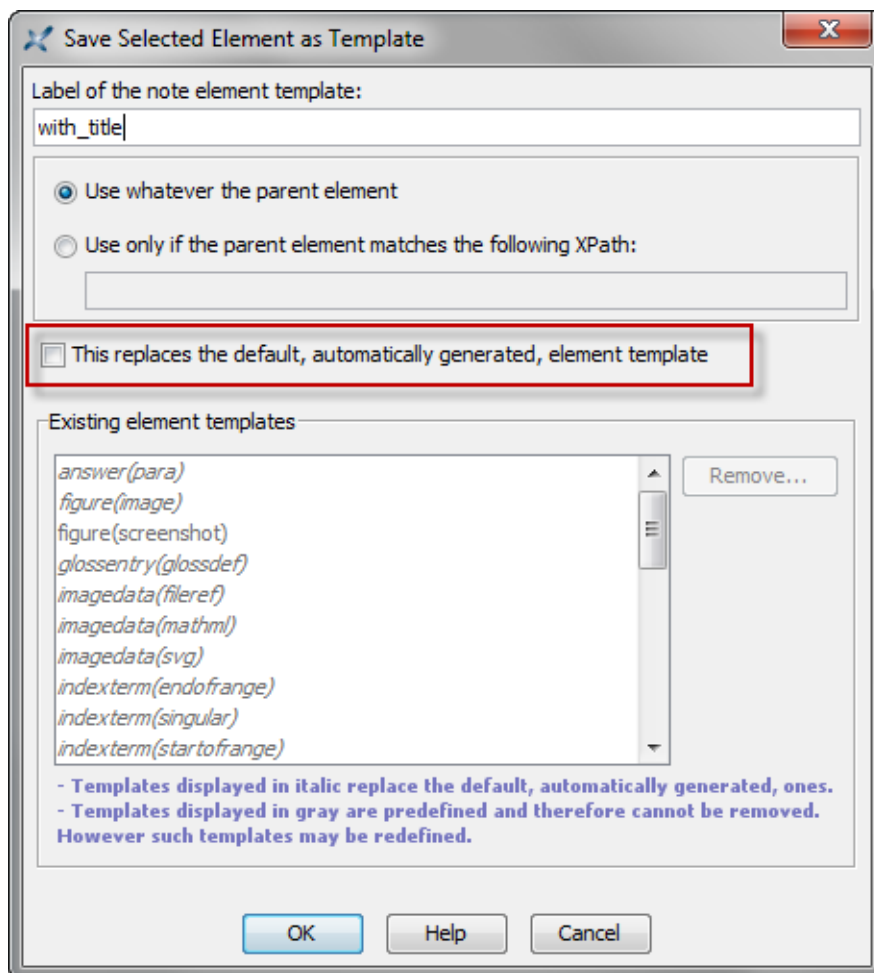
 [Watch the screencast](#)

Inserting custom element templates

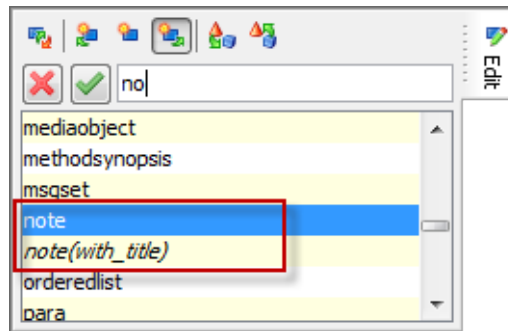
Generally, when you insert a new element, XMLmind XML Editor automatically creates for you the *valid element having the simplest content*. For example, if you insert a `note` element in a DocBook document, this `note` contains an empty `para`.

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.



 Watch the screencast

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 text 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.

However, there are 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"> • Ctrl-Up_Arrow selects the parent of selected node. <p>If there is no explicit node selection, this hotkey selects the text node containing the caret.</p> <ul style="list-style-type: none"> • Ctrl-Down_Arrow selects the last selected child of selected element. <p>That is, pressing repeatedly Ctrl-Up_Arrow and then pressing repeatedly Ctrl-Down_Arrow, works as expected.</p> <ul style="list-style-type: none"> • Ctrl+Shift-Up_Arrow selects the preceding sibling of selected node. • Ctrl+Shift-Down_Arrow selects the following sibling of selected node. 	<ul style="list-style-type: none"> • Ctrl-button1 selects the node clicked upon. <p>Repeating this action without moving the mouse selects its parent, then its grand-parent, etc.</p> <ul style="list-style-type: none"> • 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.

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"> • Esc Right_Arrow (means: press Esc then press Right_Arrow) extends the selection to the following sibling of selected node. <p>Note Esc Right_Arrow (and Esc Left_Arrow) will first select the element containing the 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.</p> <ul style="list-style-type: none"> • Esc Left_Arrow extends the selection to the preceding sibling of selected node. • Esc Down_Arrow selects all the children of the selected element. 	<ul style="list-style-type: none"> • Shift-button1 extends the node selection, if any, to the node clicked upon.

Cancel the text or node selection

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

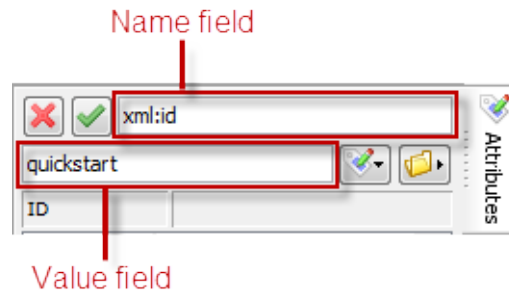


Watch the screencast

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.

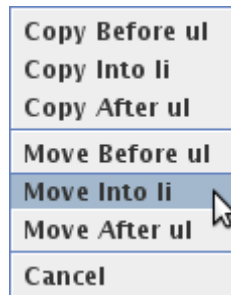
 [Watch the screencast](#)

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 display a popup menu allowing copy or move this text before, into or after the drop location.

The popup menu displayed at the end of a drop action



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`):

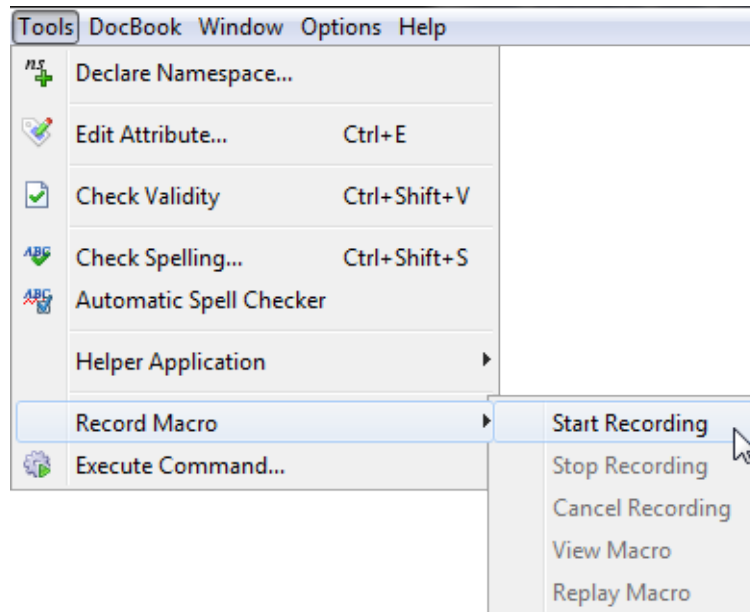
- If you drop it onto an image placeholder, this will open a dialog box allowing to change the source of the image element. See [Inserting images in your document](#).
- If you drop it onto an external link element (DocBook 5: `link xlink:href="xxx"`, DocBook 4: `ulink` element, DITA topic: `xref` or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

 [Watch the screencast](#)

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?

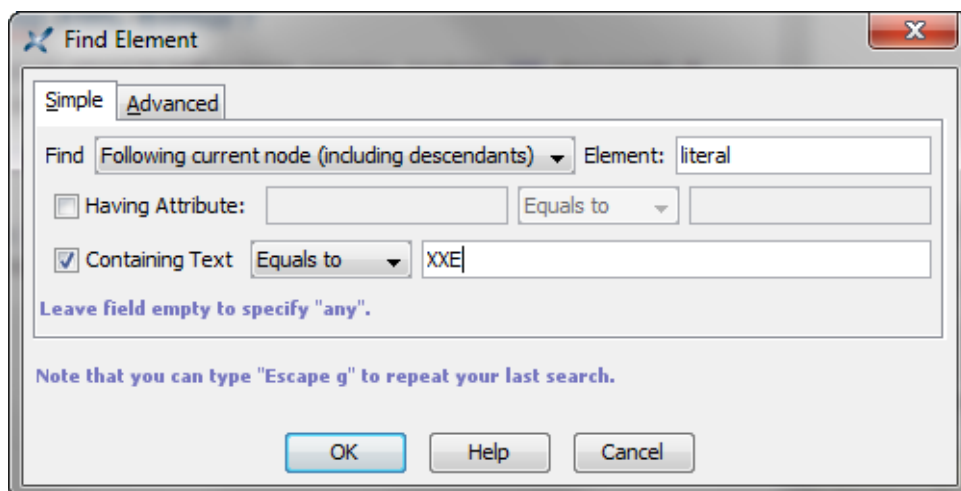
The answer is: perform the editing operation only once as you would do it normally, but make sure to record your actions using the **Tools**→**Record Macro** facility.




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.

 [Watch the screencast](#)

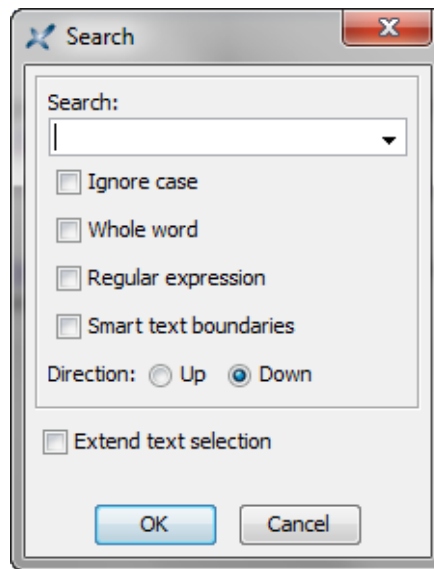
Keyboard shortcuts

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.

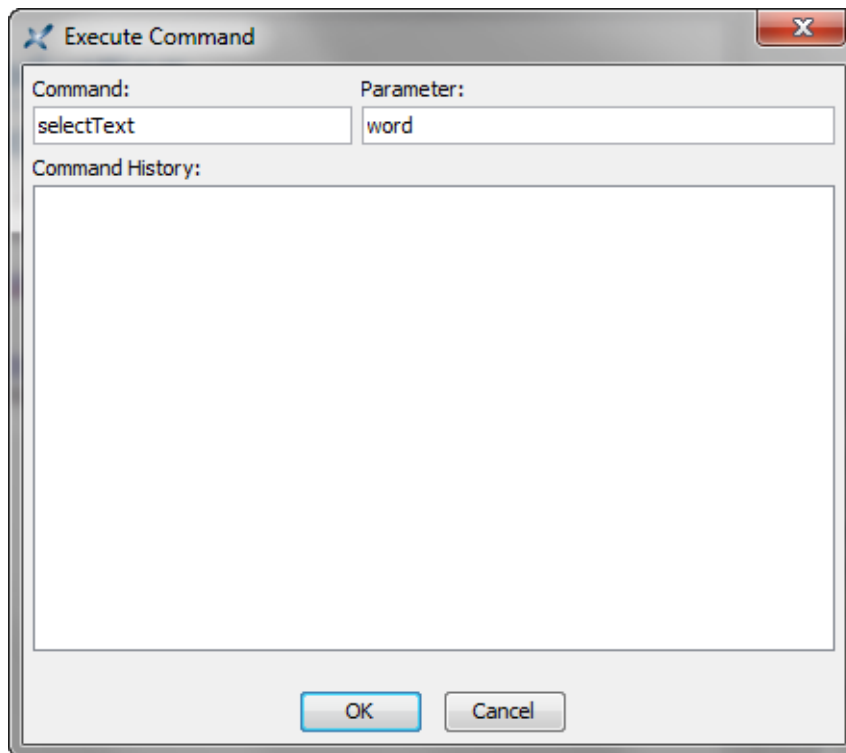
Keyboard Shortcut	Description
Esc m (m like macro)	Start/stop recording a macro.
Esc q (q like quit)	Cancel the recording.
Esc p (p like play)	Execute ("replay") the macro.
Esc f (f like find)	Find element.
Esc s (s like search)	Find some text. Displays a simple text search dialog box (see below) which is more convenient to use in the context of macro recording than the Search right-side pane.
Esc x (x like execute)	Execute a command by specifying its name and its parameter. Example: execute command "selectText" with parameter "word". The Tools → Execute Command dialog box (see below) 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

Keyboard Shortcut	Description
	<p>your mouse inside the document view will cancel the recording of the macro.)</p> <p>The commands which can be executed this way are all documented in XMLmind XML Editor - Commands.</p>

The simple text search dialog box



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



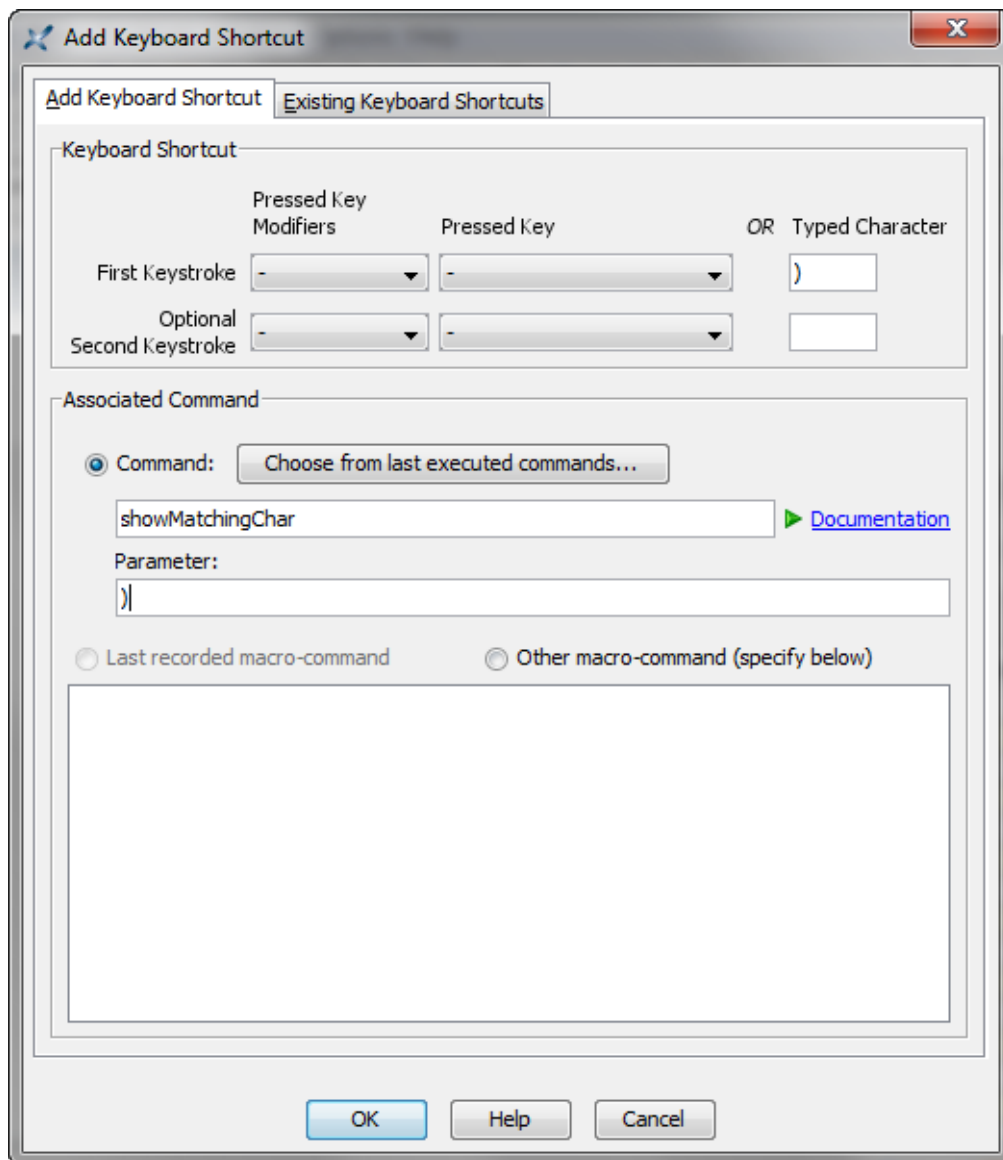
Related productivity tips

► 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:

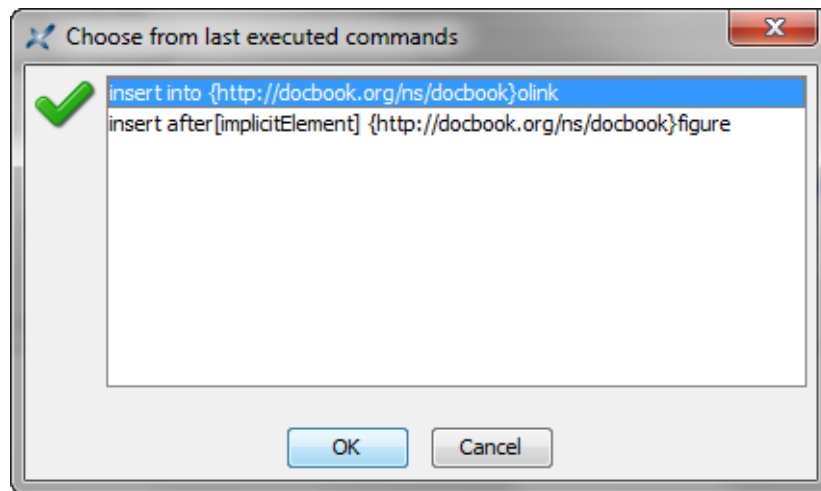


Using this dialog box is simple in its principle:

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).

Keystroke	Command	Parameter	Description
	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.

[Watch the screencast](#)

Specialized tools






Creating a DITA bookmark


A DITA map is basically a sequence of possibly nested `topicref` elements. As suggested by its name, a `topicref` is simply a reference (or a “pointer”) to a topic stored in a separate file of its own.

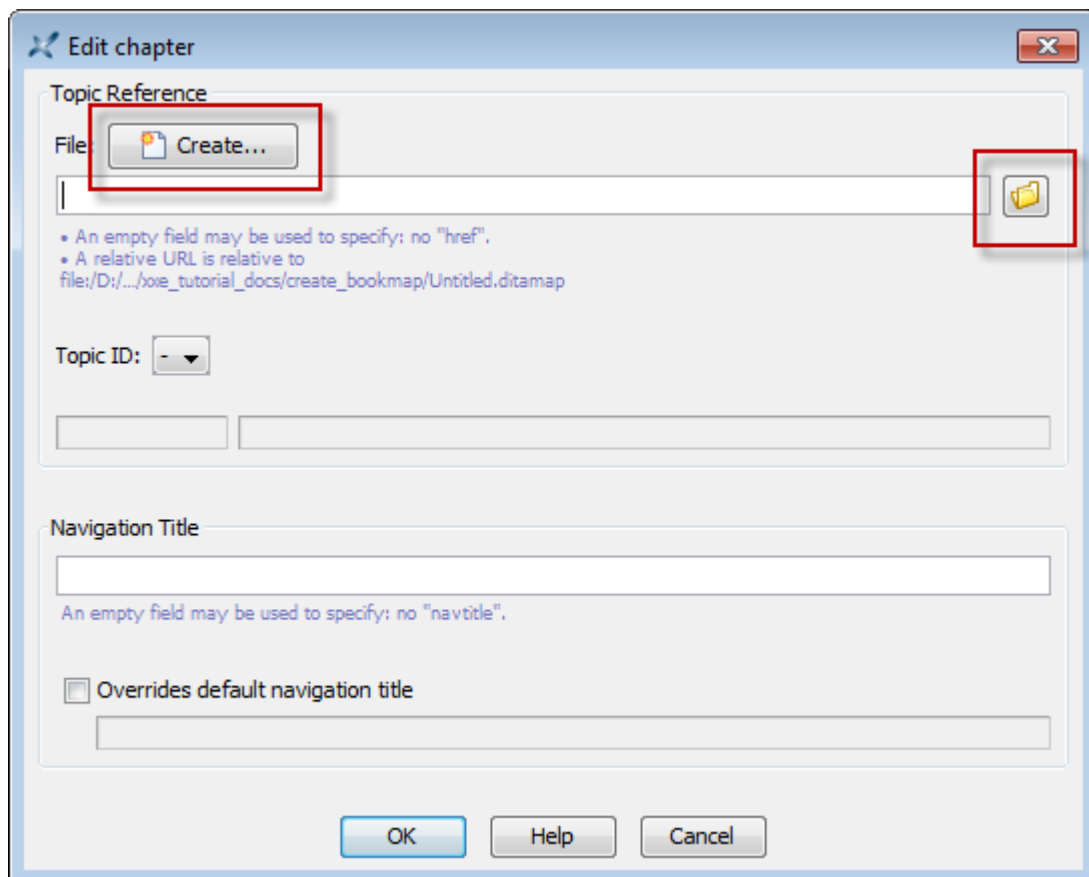
A DITA *bookmark* is a useful variant of a map in which some specialized `topicref` elements: `part`, `chapter`, `appendix`, etc, may be used, not only to reference a topic, but also to give it a distinctive role.



In this lesson, we'll get acquainted with the buttons of the specialized toolbar which is automatically displayed when you open a DITA map or bookmark in XMLmind XML Editor (**XXE** for short).

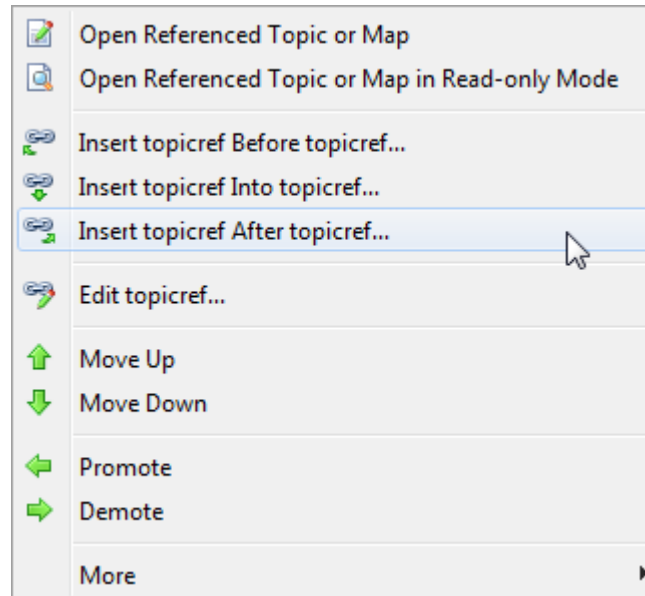






- We'll first see how the  **"Edit Topicref"** button may be used to modify a `chapter` `topicref`.
- Clicking the  **"Open Referenced Topic or Map"** button or more simply, double-clicking a `topicref`, may be used to open the corresponding topic in **XXE**.
- Buttons  **"Insert Topicref Before"**,  **"Insert Topicref Into"** and  **"Insert Topicref After"** may be used to add `topicrefs` to a map.

Note that these buttons, just like the  **"Edit Topicref"** button, display a dialog box allowing to create on the fly the topic to be referenced or to choose an existing topic file.



- Buttons  **Promote** and  **Demote** may be used to respectively un-nest and nest topicrefs. For example, selecting a `topicref` nested inside a chapter then clicking **Promote** will un-nest this `topicref` and automatically convert it to a chapter.
- All the aforementioned tools are found not only in the specialized toolbar, but also in a popup menu displayed when you right-click a `topicref` or drag and drop a file onto a `topicref`.



- Opening a map in **XXE** does not display a specialized pane but simply displays an ordinary document view styled using a specific CSS file. This means that, in addition to the specialized toolbar, any stock tool (**Edit** tool, **Attributes** tool, etc) may be used to edit a DITA map. Examples:
 - Select appendix and click  **Delete** (or press `Del`) to delete it.
 - Select frontmatter/booklist and use  **"Insert After"** (or press `Ctrl-J`) to add a `topicref` pointing to an introductory topic after the `toc` element.
- Finally, we'll see how to use the  **"Add reltable"** button to add a `reltable` to a DITA map and how to use the  **"Reltable row"** button to remove a `relrow` from this `reltable`.

 [Watch the screencast](#)

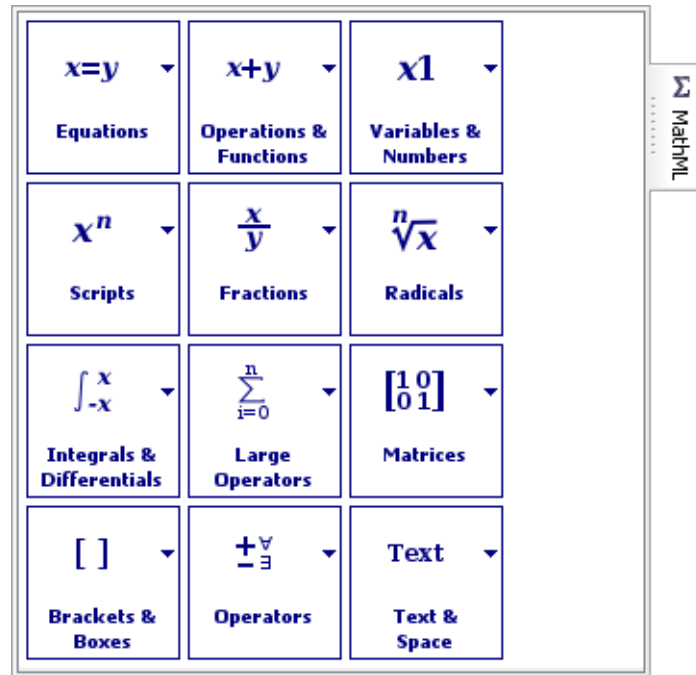
Related productivity tips

- Custom document templates

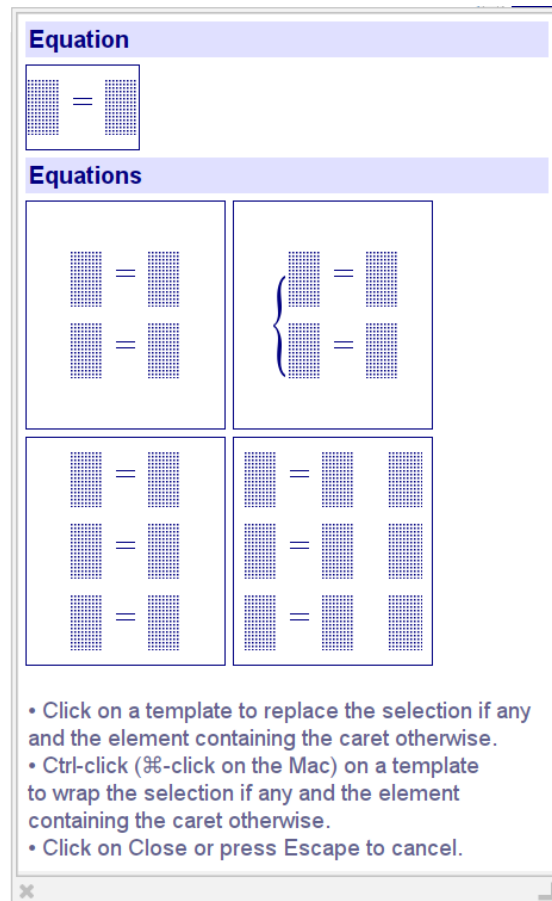
Adding MathML equations

The Σ 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, a DITA topic or an XHTML 5 page.

The newly inserted `m:math` element contains just an empty `m:mi` (math 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.

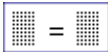



Clicking on a template replaces the explicit or implicit selection (whether it's a text or node selection) by this template. Remember that when there is no explicit text or node selection, the element containing the caret is said to be “implicitly selected”. Therefore, generally, clicking on a template 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

$$\frac{1}{x} = \frac{1}{y}$$

approach. If you want to write $\frac{1}{x} = \frac{1}{y}$, you cannot write that naturally, from left to right. You'll have to first

insert the  template, then replace the left placeholder by a fraction template , then replace the right placeholder by another fraction template and so on. This top-down approach may seem tedious and slow, but it is also easy to remember and straightforward to use.

Fortunately, in some cases, there are ways to workaround this top-down approach:

- It's possible to type simple expressions in `m:mi` (math identifier) and `m:mn` elements (math number) and then invoke menu item  **"Parse Text As MathML"** (keyboard shortcut **Ctrl+Shift-SPACE**) in order to parse the expression as MathML. For example, if you can type "E = m*c^2" and then press **Ctrl+Shift-SPACE** to let XMLmind XML Editor do all the hard work for you.
- If you **Ctrl-click** (**Cmd-click** on the Mac) on a template, this will *wrap*, rather than replace, the explicit or implicit selection (whether it's a text or node selection) in the MathML template clicked upon.

This being said, if you know MathML, you are free to write your equations directly 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.

Let's give it a try

$$F = \frac{G * m_1 * m_2}{d^2}$$

Suppose that you need to insert the following equation in a DocBook 5 document:

1. Replace the initial empty `m:mi` (math identifier) element by clicking on the simplest equation template.
2. Type "F" in the left placeholder.
3. Replace the right placeholder by clicking on the fraction template.
4. Type "G*m_1*m_2" in the numerator of the fraction; press **Ctrl+Shift-SPACE** to invoke **"Parse Text As MathML"**; Click in the popup (or press **Down** then **Enter**, like for any popup menu) to accept what's suggested in the popup.

Notice that character "*" is translated by **"Parse Text As MathML"** to what's looks like a small x. This very common MathML operator is called `InvisibleTimes`. This operator is peculiar because normally it's invisible. However for easier editing, XMLmind XML Editor has made it visible.

5. Type "d^2" in the denominator of the fraction; press **Ctrl+Shift-SPACE** to invoke **"Parse Text As MathML"**; press **Down** then **Enter** to accept what's suggested in the popup.

Most computer keyboards will allow you to directly type "d²" in the empty denominator of the fraction. This empty denominator is, like all placeholders, an `m:mi` element. Don't do that because "d²" is clearly not an identifier. In MathML, "d²" is represented by:

```
<m:msup>
  <m:mi>d</m:mi>
  <m:mn>2</m:mn>
</m:msup>
```

and this is exactly what will be suggested by **"Parse Text As MathML"** if you type "d^2" in the denominator of the fraction.



Watch the screencast

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 [
  <!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>
```

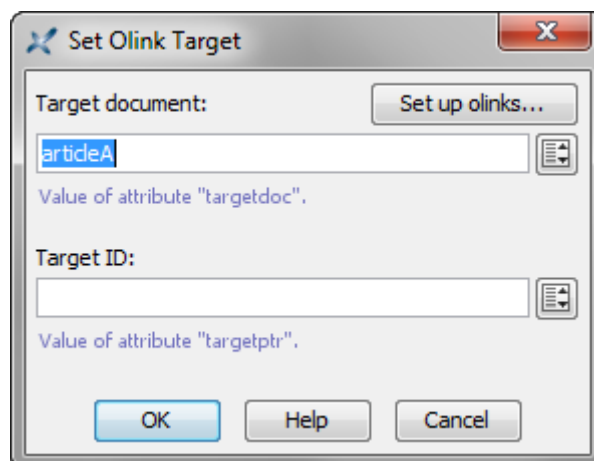
It is customary to use the ID of the root element of a document as its symbolic name.


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.

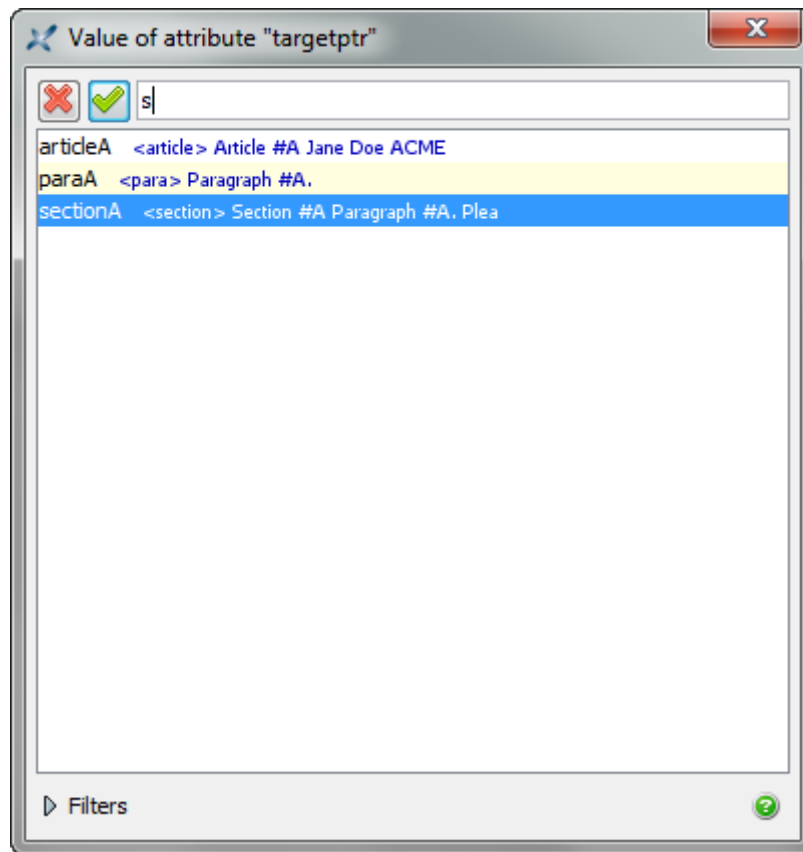
In the following section, we'll see how XMLmind XML Editor can help you *create* olink elements. However it is important to understand that XMLmind XML Editor (**XXE** for short) cannot help you in putting these olink elements into use when converting your DocBook document to HTML, PDF, etc. For example, **XXE** cannot assist you in creating the sitemap file, in populating it with link targets, etc. All these tasks must be performed "by hand", outside **XXE**.

How XMLmind XML Editor can help you

1. Use **DocBook**→**Set up olinks** to declare all your DocBook documents (XMLmind XML Editor cannot read your sitemap file). This is done once for all. See how [below](#).
2. Use the **Edit** tool to insert an empty olink element at caret position or to convert selected text to an olink element.
3. Right-click anywhere inside the olink element. Doing this displays a contextual popup menu having a "**Set Link Target**" entry.
4. Selecting this "**Set Link Target**" entry displays the following dialog box:



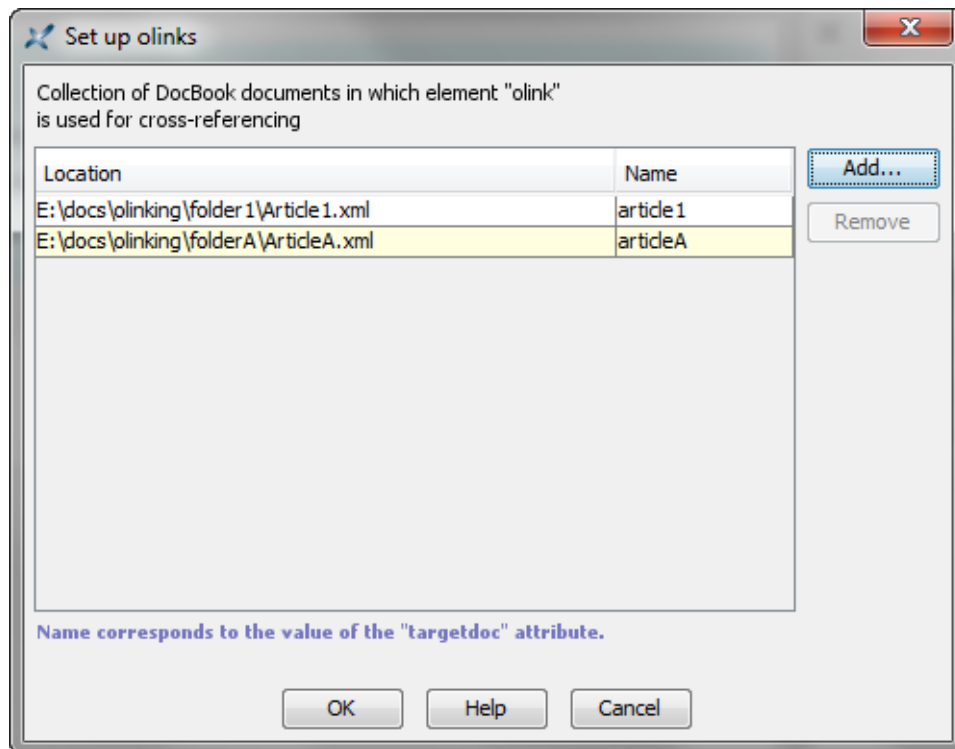
Both text fields in the above dialog box support autocompletion. Moreover, clicking  displays a value chooser dialog box which may be more convenient to use:



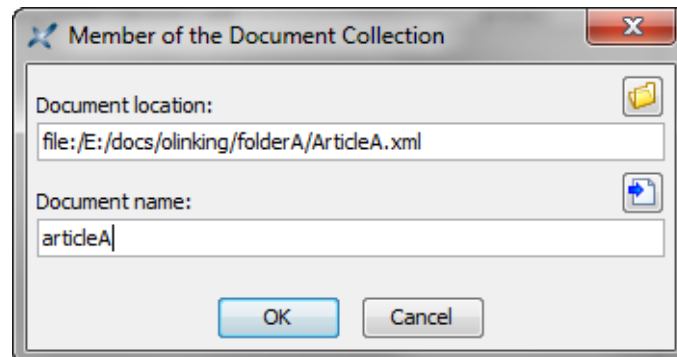
Alternatively, you may use the **Attributes** tool to specify the `targetdoc` attribute and optionally, the `targetptr` attribute of an `olink` element. If, and only if, **DocBook**→**Set up olinks** has first been used to declare all your DocBook documents, then you'll be able to use the autocompletion facility of the **Attributes** tool when you'll specify the values of the `targetdoc` and `targetptr` attributes.


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.

 [Watch the screencast](#)

Reviewing changes using the Compare tool

Why use the Compare tool?

The **Compare** tool allows to compare two revisions of the same initial document.

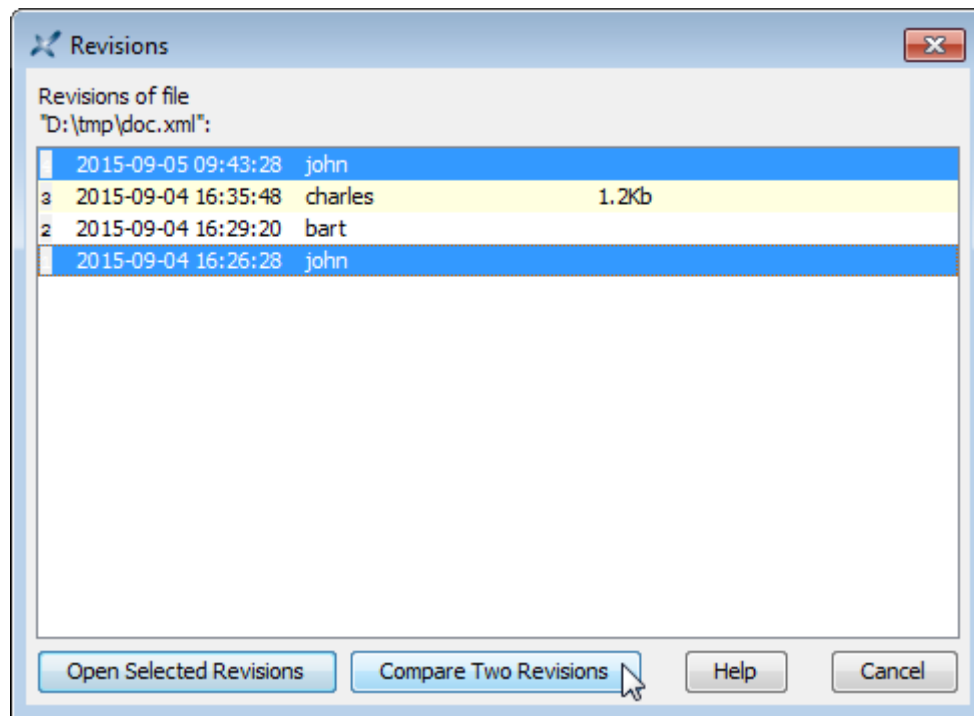
Unlike generic XML comparison tools, you must explicitly turn on an option in the initial document if you want to be able to compare two revisions using the **Compare** tool. This is done by selecting menu item **Tools→Revisions→Enable the Comparison of Revisions** or **Tools→Revisions→Store All Revisions in the Document**.

Option "**Enable the Comparison of Revisions**" requires you to keep each revision in its own XML file. In practice, this option is useful only when your documents are stored in a CMS, a document repository, on a versioning file system or more simply, if you often archive copies of the XML sources of your documents.

Option "**Store All Revisions in the Document**" instructs XMLmind XML Editor to start storing all the revisions of the document being edited in the XML file containing this document. Therefore this option is more handy when it comes to reviewing the changes made to your document.

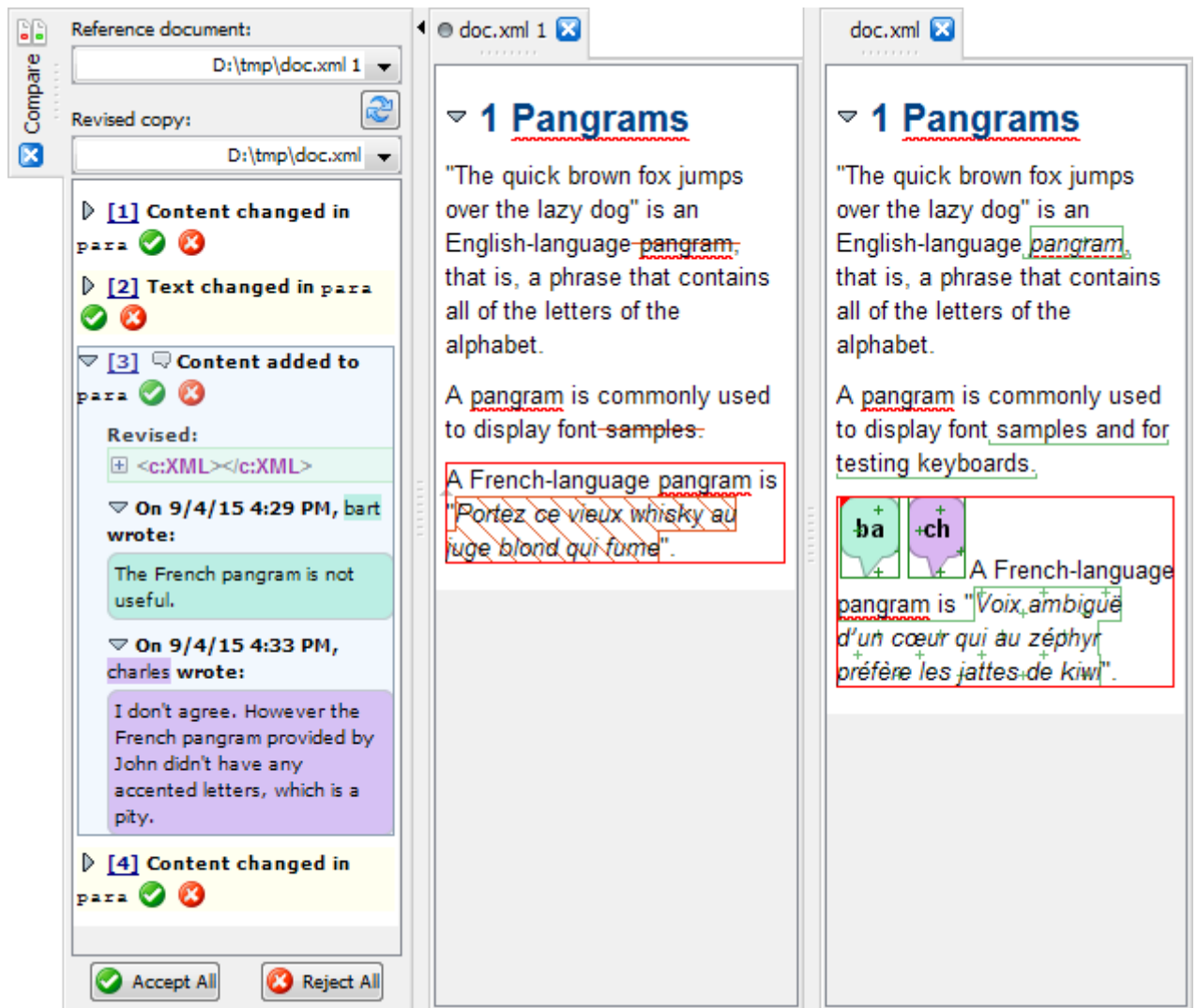
If all revisions are stored in the XML file, then you'll be able to use open any of these revisions using **Tools→Revisions→Open Revision**.

The "Open Revision" dialog box, here allowing to compare the document being edited (revision #4; latest one) to the initial revision (revision #1)



The comparison of revisions is done using the **Compare** tool. 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 1 (initial revision) and doc.xml (latest revision), 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 turns on option "Store All Revisions in the Document" in `doc.xml` using **Tools**→**Revisions**→**Store All Revisions in the Document**, saves the document to disk and then sends a copy to Bart.

Bart modifies `doc.xml`. He also adds a *remark* using **Tools**→**Remark**→**Insert or Edit Remark** explaining why he deleted the last paragraph. This results in creating revision #2 of `doc.xml`.

Bart sends his modified copy to Charles, who does not agree with some of the changes made by Bart. This results in creating revision #3 of `doc.xml`.

Finally, John receives a `doc.xml` file containing the changes made by Bart and then by Charles.

Out of curiosity, John uses **Tools**→**Revisions**→**Open Revision** to take a look at what Bart has done, that is, revision #2 of `doc.xml`.

After closing revision #2, John uses **Tools→Revisions→Open Revision** one more time. This time, he wants to compare his original work (revision #1; the initial revision) to the document being edited (revision #4; latest one) which contains all the changes made by Bart and Charles.

John accepts all the changes except those contained in the last paragraph, closes revision #1 and the **Compare** tool, turns off option "**Store All Revisions in the Document**" and saves `doc.xml` to disk.



[Watch the screencast](#)

Conditional processing made easy

What is conditional processing?

Conditional processing, also called *profiling*, allows you to use the same XML source document to create several variants of the deliverables (PDF, RTF, EPUB, etc). For example, your document is a manual for a product which has 3 editions: "Lite", "Professional", "Ultimate", and you want to create a different set of deliverables for each edition.

Implementing conditional processing requires you to set special attributes (called *conditional processing attributes* or *profiling attributes*) on the elements you want to include or exclude from the deliverable depending on the variant you want to generate.

For example, let's suppose your document contains 3 paragraphs: first paragraph has no profiling attributes, second paragraph has profiling attribute `product="Lite"` and third paragraph has `product="Professional"`. If you generate a variant of the deliverable for the "Lite" edition, then this variant will contain the first and second paragraph, but not the third one.

Generating a variant of the deliverable for the "Lite" edition implies applying a *profile* where `product="Lite"` to your document. A profile may be seen a named combination of profiling attribute values. In our example, let's call this profile "lite". The "lite" profile comprises a single profiling attribute value `product="Lite"`.

Conditional processing is available for DITA documents and for DocBook documents. See *Darwin Information Typing Architecture (DITA) Version 1.2, Conditional processing (profiling)*. See *DocBook XSL: The Complete Guide, Profiling (conditional text)* by Bob Stayton.

Why install and use the "Easy Profiling" add-on?

In a nutshell, without the help of the **"Easy Profiling"** add-on, implementing conditional processing in your DITA or DocBook documents is tedious and error-prone.


Moreover, the **"Easy Profiling"** add-on allows to assign *screen styles* to elements having profiling attribute values and/or to elements which are to be included or excluded from the deliverable depending on the selected profile.


How to put the "Easy Profiling" add-on into use?

We'll use a simple DITA bookmap, `user_guide.ditamap`, referencing just two topics, `export_pdf.dita` and `advanced_features.dita`, to explain this process.

First step: specify which profiles you are going to use in your document

This is done using **File**→**New** and then selecting the document template called **"Conditional Processing Profiles"** found in the category of the document you are authoring (**"DITA"** in the case of our example).



Untitled.profiles 

▼ **Conditional processing profiles** 

What conditional processing attributes do you use in your documents?

(Click + to add an element. Click - to remove an element.)


▼ Attribute name ??? - +

Allow value   - +


☒ Allow multiple values separated by whitespace

What combinations of conditional processing attributes (profiles) do you use in your documents?

▼ Profile ID ??? - +

Description 

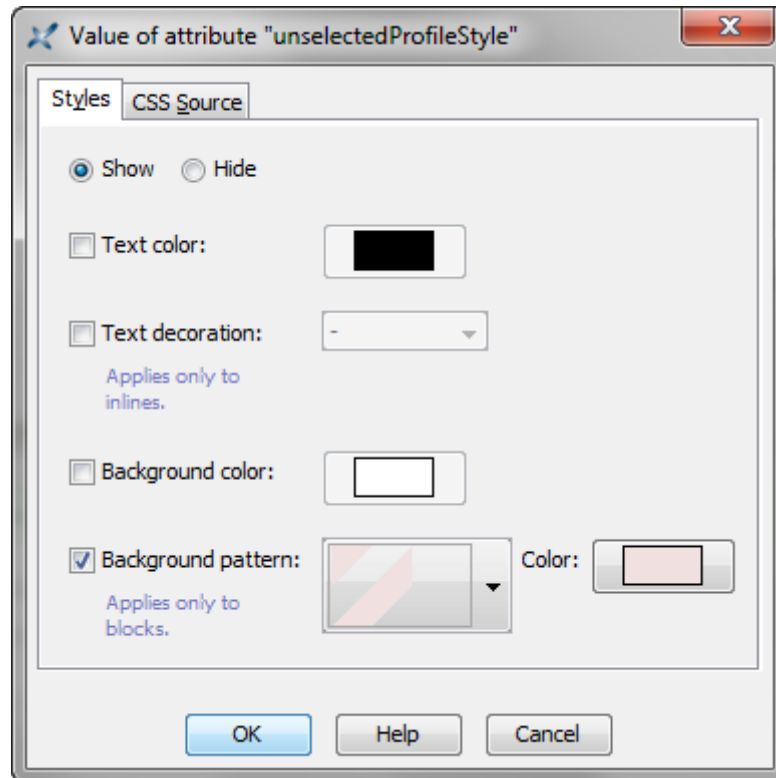
▼ Attribute name ??? - +

Has value  - +

1. First specify which profiling attributes you are going to use in your document (example: attribute "product"). For each of these attributes, specify all the allowed values (example: attribute values "Lite", "Professional" and "Ultimate").
2. Then specify one or more profiles, that is, give a name (examples: "lite", "pro", "ultimate") to a combination of profiling attribute values (example: profile "lite" comprises product="Lite").
3. While at it, give a distinct screen style (CSS style properties) to the "unselected profiles". That is, give a distinct style to the elements which will be excluded from the deliverable given the selected profile.

This is done by clicking the  button which is next to the **"Conditional processing profiles"** document title and then choosing **"Style of Unselected Profile"** from the popup menu.

In the dialog box below, excluded elements are given a pink hatched background.

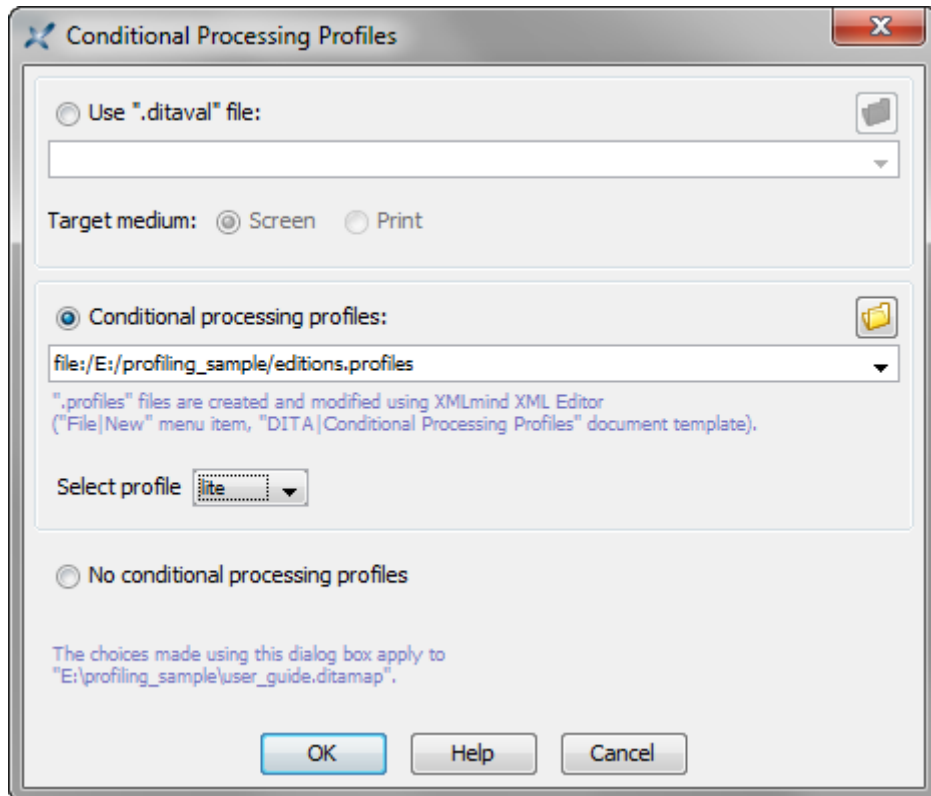


4. Finally save this special XML document to a file having a ".profiles" extension.

Second step: associate a profile to your document

This is done by using **BookMap**→**Conditional Processing**→**Select Profile**.

In the dialog box below, the "editions.profiles" file created above is associated to DITA bookmap user_guide.ditamap.



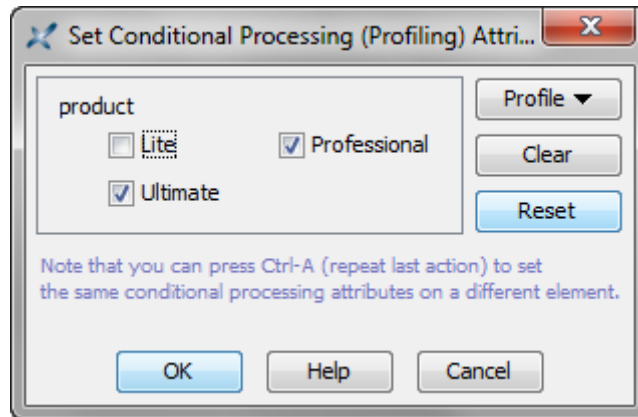
Associate a profile to the DITA map and not to the individual topics referenced by this map, then make sure to have checked **Tools→Use as Master Document** for this map. When this the case, the topics referenced by a map automatically “inherit” the profile of this map.

The same tip applies to DocBook. Associate a profile to the modular book and not to the individual chapters included in this book, then make sure to have checked **Tools→Use as Master Document** for this modular book.

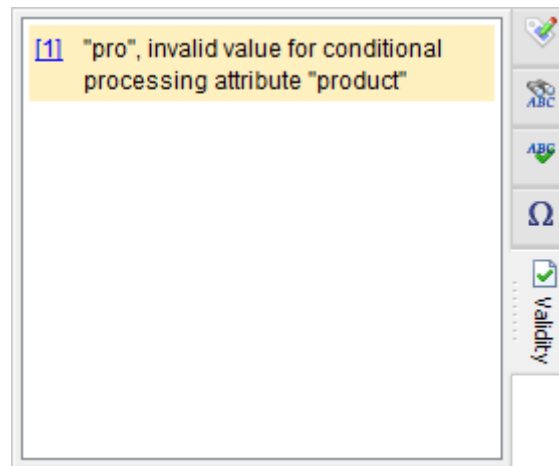
Third step: set profiling attributes on some elements

The easiest way to set profiling attributes on some elements is to use **Bookmap→Conditional Processing→Set Profiling Attributes**.

In the dialog box below, `topicref href="advanced_features.dita"` found in DITA bookmap `user_guide.ditamap` is given profiling attribute value `product="Professional Ultimate"` (which means that this `topicref` applies to both the "Professional" and "Ultimate" editions or the product being documented).



While less convenient, it's also certainly possible to use the **Attributes** tool to set profiling attributes on some elements. Note that when a ".profiles" file has been associated to the document being edited, undeclared profiling attribute values are reported as semantic errors by the **Validity** tool.



Fourth step: generate the deliverable corresponding to selected profile

Once a profile has been selected using **BookMap**→**Conditional Processing**→**Select Profile**, suffice to use one of the items of menu **BookMap**→**Convert Document** to generate the corresponding variant of the deliverable. For example, use **BookMap**→**Convert Document**→**Convert to PDF** to generate "user_guide_lite.pdf" then select profile "ultimate" and use **BookMap**→**Convert Document**→**Convert to PDF** again to generate "user_guide_ultimate.pdf".

 [Watch the screencast](#)

Related tutorials

- [Working with master documents](#)