XMLmind XML Editor - DocBook Support

Hussein Shafie, Pixware <xmleditor-support@xmlmind.com>

April 2, 2013

Abstract

This document describes the commands which are specific to DocBook.

Table of Contents

1. The DocBook menu

Table editing commands fully support CALS tables as well as HTML tables. Most table editing commands can be repeated by using Edit → Repeat (**Ctrl**+**A**).

Note that using this table editor, or simply saving a document, or checking a document for validity, guarantees that the `cols` attribute of a `tgroup` is up to date. That is, you may forget about the `cols` attribute, XMLmind XML Editor will always compute it for you.

Menu	Item	Description
Column For a command in this menu to work, click anywhere inside a cell (or explicitly select a cell or an element having a cell ancestor).	Insert Before	Insert a column before column containing specified cell.
	Insert After	Insert a column after column containing specified cell.
	Cut	Cut to the clipboard the column containing specified cell.
	Copy	Copy to the clipboard the column containing specified cell.
	Paste Before	Paste copied or cut column before column containing specified cell.
	Paste After	Paste copied or cut column after column containing specified cell.
	Delete	Delete the column containing specified cell.
	Sort Rows	Sort all the rows of the table according to the string values of the cells of the "selected column". (The "selected column" is the column containing specified cell.) A dialog box is displayed allowing to specify the following sort options: Order Dictionary is the language-specific alphabetical order. Example: (`Charles`, `best`, `Albert`) is sorted as (`Albert`, `best`, `Charles`). Numeric. The string value of a cell is expected to start with a number. Example: (`+15.0%`, `1.50%`, `-20%`) is sorted as (`-20%`, `1.50%`, `+15.0%`).

Menu	Item	Description
		Lexicographic is the order of Unicode characters. Example: (`Charles`, `best`, `Albert`) is sorted as (`Albert`, `Charles`, `best`). Dictionary and Numeric orders will cause this menu item to fail, unless the language of the table can be determined (i.e. lookup for the `lang` attribute). Direction Ascending means: A to Z, low to high. Descending means: Z to A, high to low. Note that: • Header/footer rows (i.e. `thead`) are never sorted. • The contents of row groups (i.e. `tbody`) are sorted separately.
Row For a command in this menu to work, click anywhere inside a cell (or explicitly select a cell or an element having a cell ancestor) or explicitly select a row.	Insert BeforeFor	Insert a row before row containing specified cell. **Note** Note that row editing commands are enabled, not only by implicitly or explicitly selecting a table cell or any of its descendants, but also by explicitly selecting a table row.
	Insert After	Insert a row before row containing specified cell.
	Cut	Cut to the clipboard the row containing specified cell.
	Copy	Copy to the clipboard the row containing specified cell.
	Paste Before	Paste copied or cut row before row containing specified cell.
	Paste After	Paste copied or cut row after row containing specified cell.
	Delete	Delete the row containing specified cell.
Cell For a command in this menu to work, click anywhere inside a cell (or explicitly select a cell or an element having a cell ancestor).	Increment Column Span	Increment the number of columns spanned by specified cell.
	Decrement Column Span	Decrement the number of columns spanned by specified cell.
	Increment Row Span	Increment the number of rows spanned by specified cell.
	Decrement Row Span	Decrement the number of rows spanned by specified cell.

Other commands:

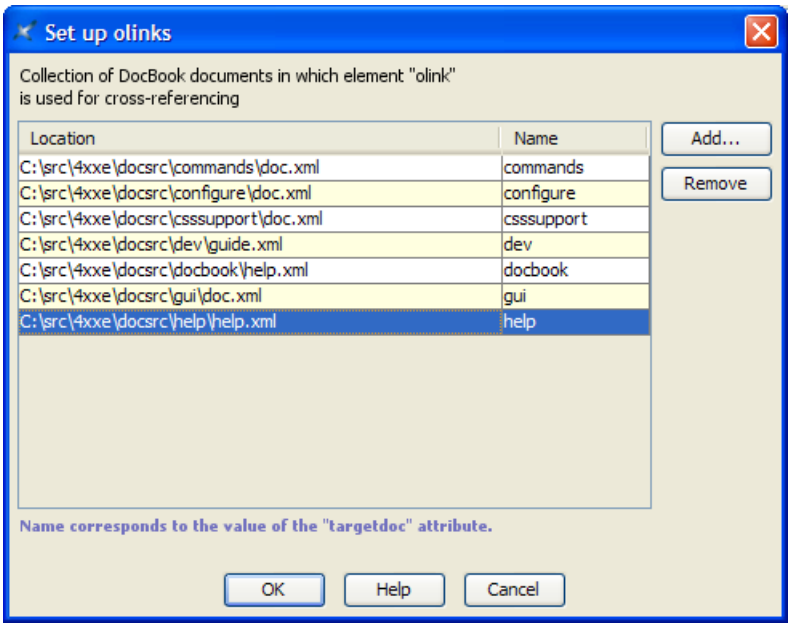
Set up olinks
The `olink` element allows to create links between different documents. Once the `olink` element has been inserted in a document, you have to specify a value for its `targetdoc` attribute and optionally, a value for its `targetptr` attribute. The `targetdoc` attribute contains the symbolic name of the document which is the target of the `olink`. The `targetptr` attribute is the ID of an element found in the target document. More information about the `olink` element and how this element is processed by the DocBook XSL stylesheets in *DocBook XSL: The Complete Guide*, by Bob Stayton.

The Attributes tool can help you specify a value for the `targetdoc` attribute by listing[1] all the symbolic names of the target documents. Once the `targetdoc` attribute has been specified, the Attributes tool can help you

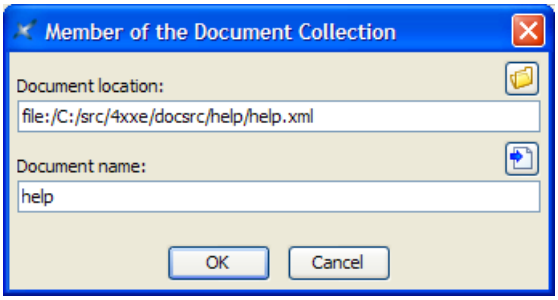
[1]Type a value in the Value field and use auto-completion or use the Edit button which is found at the right of this text field to display a specialized dialog box.

specify a value for the `targetptr` attribute by listing all the IDs found in the target document. However for this facility to work, you first need to declare the collection of DocBook documents in which `olink` is used for cross-referencing.

1. Select DocBook → Set up olinks. This will display the following dialog box:



2. Click Add. This will display this other dialog box.



3. Use the Browse button to specify the URL of a document which is a member of the collection. In the above screenshot, this URL is "`file:/C:/src/5xxe/docsrc/help/help.xml`".

Tip

You can mix DocBook v4+ and DocBook v5+ documents within the same collection.

4. Type the symbolic name of the document in the Document name text field. In the above screenshot, this name is "`help`".

   This name, which cannot contain space characters, corresponds to a possible value for the `targetdoc` attribute. The same symbolic name must also be used in the *target database document*. Example:

```
<!DOCTYPE targetset
  SYSTEM "../../addon/config/docbook/xsl/common/targetdatabase.dtd" [
  ...
  <!ENTITY help SYSTEM "help_html.targets">
  ...
]>
<targetset>
  <sitemap>
    <dir name="doc">
      ...
```

```
      <dir name="help">
        <document targetdoc="help">
          &help;
        </document>
      </dir>
      ...
    </dir>
  </sitemap>
</targetset>
```

Tip

Instead of typing the symbolic name of the document referenced in the Document location text field, it's also possible to click the  button. This button allows to use the ID of the root element (if any) of the document referenced in the Document location text field as a symbolic name.

Using the ID of the root element as the symbolic name of an “olinked document” is a common practice. However, before using this button, make sure that this practice is actually used in your organization.

5. Repeat steps 1 to 4 until you have declared all the members of your document collection.

This setup is done once for all for both the DocBook and DocBook v5+ configurations. However you may add or remove members to/from your document collection at any time.

Paste As

The following entries of this submenu allow to paste the *plain text* copied to the clipboard, typically using a third-party word processor or spreadsheet, as:

- one or more paragraphs,
- OR a `programlisting` element,
- OR one or more list items,
- OR an itemized list,
- OR one or more table rows,
- OR a table.

The last two menu entries assume that each text line specifies a table row and that, within a text line, the contents of the table cells are separated by tab characters.

Tip

If you need to paste the copied text as an ordered list, first paste this text as an itemized list then convert the pasted list to an ordered list using Edit → Convert (**Ctrl**+**T**).

The above menu entries paste elements *after* the implicitly or explicitly selected element.

The following entries of this submenu allow to paste the *image* copied to the clipboard as:

- `inlinemediaobject`,
- `mediaobject`,
- `figure`.

Except for menu entry "inlinemediaobject" which replaces the selection or pastes an element at caret position (like Edit → Paste), the other menu entries paste an element *after* the implicitly or explicitly selected element.

Convert between informal element and element

Converts an ``informal element'' to/from a ``formal element'' having a title.

This command currently works for `informaltable`/`table`, `informalfigure`/`figure` and `informalexample`/`example`.

Links callouts

Links a sequence of `callout` elements to the corresponding sequence of `co` or `area` elements (and, of course, also the other way round).

Useful information about callouts is found in *DocBook XSL: The Complete Guide* by Bob Stayton: Program listings, Annotating program listings, Callouts.

In order to use this command, you need to:

1. Create a `programlisting` containing a number of `co` elements. No need to specify the ID or `linkends` attributes for these `co` elements.

   Note that this command also works for any element containing `area` elements rather than `co` elements (e.g. a `programlistingco`).

2. Add a `calloutlist` element somewhere after the `programlisting`. No need to specify the ID or arearefs attributes for the `callout` elements.

   ## Important

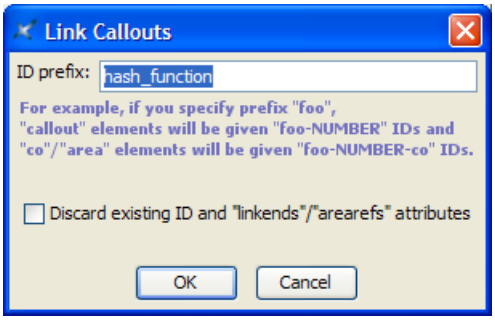
   Make sure to create exactly the same number of `co` and `callout` elements. This is needed because the *N*th `co` element will be linked to the *N*th `callout` element.

3. Explicitly select the node range comprising both the `programlisting` and the `calloutlist` elements.

   In fact, you can select any element containing, at any nesting level, a sequence of `co` elements followed by a sequence of `callout` elements. For example, if your `programlisting` and `calloutlist` elements are contained in a `section` element, you can select just this section.

4. Select DocBook → Link callouts.

5. The following dialog box is displayed:

   Specify a prefix for the IDs which will be automatically generated for the `co` and the `callout` elements. The links (`linkends` and `arearefs` attributes) between the `co` and the `callout` elements of course need to refer to these IDs.

6. Click OK.

Notice that the above dialog box has a "Discard existing ID and linkends/arearefs attributes" checkbox. This checkbox is needed because the "Links callouts" command has been designed to be used, not only on newly created `programlisting` plus `calloutlist` elements, but also on existing, possibly hand-written, possibly complex[2] `programlisting` plus `calloutlist` elements.

[2]For example, containing a `callout` element linked to *several* `co` elements. In such case, the numbering of `co` and `callout` elements done on screen by XMLmind XML Editor will not reflect what you'll get when you'll convert your document to HTML or PDF. However this limitation should not prevent you from specifying such multi-`co` `callout` elements if needed to.

When the `co` and `callout` elements found inside the node selection are found to already have ID attributes, this checkbox is enabled and, by default, unchecked. When this is the case, running this command will affect only the newly created `co` and `callout` elements. All the existing IDs and links will be left unchanged.

Insert or Edit `indexterm`

If the caret is anywhere inside an `indexterm` element or if a single element or node is explicitly selected anywhere inside an `indexterm` element, this menu item displays an `indexterm` editor dialog box [10] allowing to modify this `indexterm` element.

Otherwise, this menu item displays an `indexterm` editor dialog box [10] allowing to create a new `indexterm` element and then to insert it at caret position.

Tip

If some text has been selected, field Term of the dialog box is automatically initialized with the text selection. Therefore the simplest way to create an `indexterm` element is first to select the term in the body of the document, then invoke Insert or Edit indexterm and finally click OK.

 Move Up

Move selected element up, that is, swap it with its preceding sibling node. Requires the element to be explicitly selected.

 Move Down

Move selected element down, that is, swap it with its following sibling node. Requires the element to be explicitly selected.

 Promote

To make it simple, increase the level of selected subsection (e.g. a `sect2` element is converted to a `sect1` element).

Requires a ``subsection'' (`section`, `sect1`, `sect2`, `sect3`, `sect4` or `sect5`) or an element which is contained in the body[3] of the section to be explicitly selected.

- If a subsection is selected, this subsection becomes a sibling of its parent section. Example: `sect2` element having `id="C"` is ``promoted'':

```
<sect1 id="A">...
  <sect2 id="B">...
  <sect2 id="C">...
  <sect2 id="D">...
```

This results in:

```
<sect1 id="A">...
  <sect2 id="B">...
<sect1 id="C">...
  <sect2 id="D">...
```

- If another type of child element is selected, this element is wrapped in a newly created section which becomes a sibling of its parent section. Example: `para` element having `id="C"` is ``promoted'':

```
<sect1 id="A">...
  <para id="B">...
  <para id="C">...
  <sect2 id="D">...
```

This results in:

```
<sect1 id="A">...
  <para id="B">...
```

[3] That is, it is not possible to ``promote'' the *title* of a section.

```
<sect1>...
  <para id="C">...
  <sect2 id="D">...
```

Demote

To make it simple, decrease the level of selected section (e.g. a `sect1` element is converted to a `sect2` element).

Requires a ``section'' (`chapter`, `appendix`, `section`, `sect1`, `sect2`, `sect3` or `sect4`) or an element which is contained in the body[4] of the section to be explicitly selected.

- If a section is selected and if this section is preceded by a section of the same type, this section becomes a subsection of its preceding sibling. Example: `sect1` element having `id="C"` is ``demoted'':

```
<sect1 id="A">...
  <para id="B">...
<sect1 id="C">...
  <para id="D">...
```

This results in:

```
<sect1 id="A">...
  <para id="B">...
  <sect2 id="C">...
    <para id="D">...
```

- If a section is selected and if this section is *not* preceded by a section of the same type, a new section is created and selected section becomes a subsection of this new section. Example: `sect2` element having `id="C"` is ``demoted'':

```
<sect1 id="A">...
  <para id="B">...
  <sect2 id="C">...
    <para id="D">...
```

This results in:

```
<sect1 id="A">...
  <para id="B">...
  <sect2>...
    <sect3 id="C">...
      <para id="D">...
```

- If another type of child element is selected, this element and all the other ``body elements'' which follow it are wrapped in a newly created subsection. Example: `para` element having `id="C"` is ``demoted'':

```
<sect1 id="A">...
  <para id="B">...
  <para id="C">...
  <para id="D">...
  <sect2 id="E">...
```

This results in:

```
<sect1 id="A">...
  <para id="B">...
  <sect2>...
    <para id="C">...
    <para id="D">...
  <sect2 id="E">...
```

[4] That is, it is not possible to ``demote'' the *title* of a section.

1.1. Convert Document sub-menu

Using the profiling stylesheets

Profiling, or conditional text, means that you can create a single XML document with some elements marked as conditional. More information in DocBook XSL: The Complete Guide.

If you need to use the profiling XSLT stylesheets rather than the regular ones, use Options → Customize Configuration → Customize Document Conversion Stylesheets and select the corresponding stylesheet.

Convert to HTML, Convert to HTML [one page]
Converts the document being edited to multi page or single page HTML.

Generating XHTML rather than HTML

If you prefer to generate XHTML rather than HTML, use Options → Customize Configuration → Customize Document Conversion Stylesheets and select the corresponding stylesheet.

Convert to Web Help
Converts the document being edited to Web Help.

Convert to HTML Help
Converts the document being edited to a `.chm` file. This command is disabled on platforms other than Windows.

For this command to work, the HTML Help compiler, `hhc.exe`, must have been declared as the helper application associated to files having a "`hhp`" extension. This can be specified by using the Preferences dialog box, Helper Applications section.

Convert to Java Help
Converts the document being edited to a `.jar` file for use by the Java™ Help system.

For this command to work, the Java™ Help indexer, `jhindexer`, must have been declared as the helper application associated to files having a "`application/x-java-help-index`" MIME type. This can be specified by using the Preferences dialog box, Helper Applications section.

Convert to Eclipse Help
Converts the document being edited to Eclipse Help.

The `eclipse.plugin.name`, `eclipse.plugin.id`, `eclipse.plugin.provider` XSL style sheet parameters must have been specified using the Options → Customize Configuration → Change Document Conversion Parameters facility.

All HTML files as well as Eclipse's `plugin.xml` and `toc.xml` are generated in the same directory:
1. This directory must be a subdirectory of the Eclipse `plugins/` directory.
2. The name of this directory must be identical to the value of the `eclipse.plugin.id` XSL style sheet parameter.

Convert to EPUB
Converts the document being edited to EPUB.

Convert to RTF (Word 2000+)
Converts the document being edited to RTF (Rich Text Format) using XMLmind FO Converter (see http://www.xmlmind.com/foconverter/). The document generated by this command can be edited and printed using Microsoft® Word 2000 and above.

Convert to WordprocessingML (Word 2003+).
Converts the document being edited to WordprocessingML using XMLmind FO Converter. The document generated by this command can be edited and printed using Microsoft® Word 2003 and above.

Convert to Office Open XML (Word 2007+)
: Converts the document being edited to Office Open XML (`.docx` file) using XMLmind FO Converter. The document generated by this command can be edited and printed using Microsoft® Word 2007 and above.

Convert to OpenDocument (OpenOffice.org 2+)
: Converts the document being edited to OpenDocument (`.odt` file) using XMLmind FO Converter. The document generated by this command can be edited and printed using OpenOffice.org 2.

Print PostScript
: Converts the document being edited to PostScript® using RenderX XEP (see http://www.renderx.com/), if its plug-in has been installed, and Apache FOP otherwise (see http://xmlgraphics.apache.org/fop/), and then, sends the generated file to the chosen printer.

Convert to PDF
: Converts the document being edited to PDF (Adobe® Portable Document Format, also known as Acrobat®) using RenderX XEP (see http://www.renderx.com/), if its plug-in has been installed, and Apache FOP otherwise (see http://xmlgraphics.apache.org/fop/).

All the above Convert commands display the URL chooser dialog box rather than the standard file chooser dialog box.

For all Convert commands except for the "Convert to HTML" command, you must specify the URL (Uniform Resource Locator) of a save file. The "Convert to HTML" command creates multiple HTML pages with a first page called `index.html`, therefore you need to specify the URL of a save directory.

Note that these commands can create directories on the fly, if needed to. For example, if you specify `http://www.acme.com/docs/report43/mydoc.html` as the URL of the save file and if directory `report43/` does not exist, this directory will be created during command execution.

Syntax highlighting

You can automatically colorize the source code contained in `programlisting` elements. This feature, commonly called *syntax highlighting*, has been implemented using an open source software component called "XSLT syntax highlighting".

If you want to turn on syntax highlighting in a DocBook document:

1. Add attribute `language` to element `programlisting`. The value of attribute `language` must be any of: `c`, `cpp`, `csharp`, `delphi`, `ini`, `java`, `javascript`, `m2`, `perl`, `php`, `python`, `ruby`, `tcl`, `xml`.
2. Specify XSLT stylesheet parameter highlight.source=1 using Options → Customize Configuration → Change Document Conversion Parameters. Do this for each output format you want to generate.

If you want to customize syntax highlighting for an HTML-based output format (XHTML, EPUB, etc), redefine any of the following CSS styles: `.hl-keyword`, `.hl-string`, `.hl-number`, `.hl-comment`, `.hl-doccomment`, `.hl-directive`, `.hl-annotation`, `.hl-tag`, `.hl-attribute`, `.hl-value`, `.hl-doctype`. Example:

```
.hl-keyword {
    font-weight: bold;
    color: #602060;
}
```

This can be done from within XXE using Options → Customize Configuration → Customize Document Conversion Stylesheets.

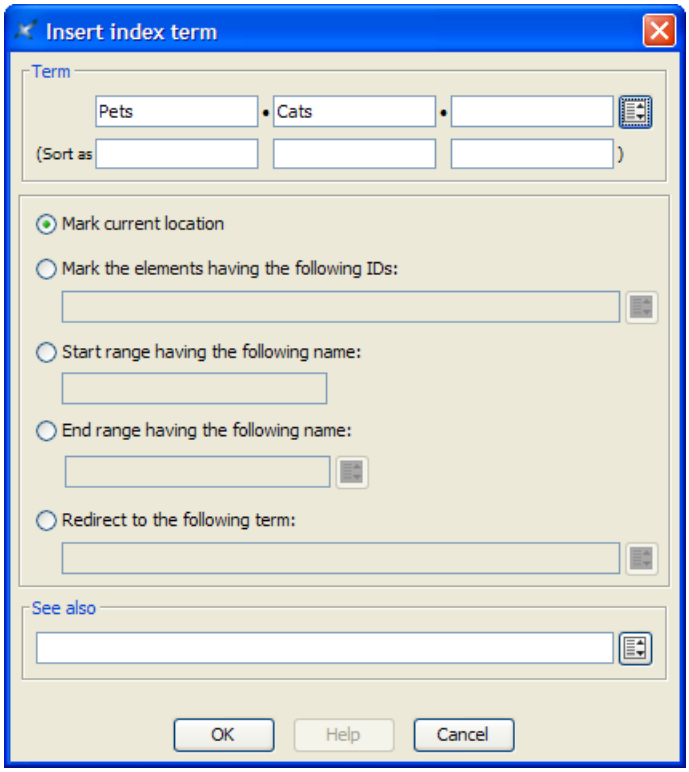
If you want to customize syntax highlighting for an XSL-FO-based output format (PDF, RTF, etc), redefine any of the following `attribute-sets`: `hl-keyword`, `hl-string`, `hl-number`, `hl-comment`, `hl-doccomment`, `hl-directive`, `hl-annotation`, `hl-tag`, `hl-attribute`, `hl-value`, `hl-doctype`. Example:

```
<xsl:attribute-set name="hl-keyword" use-attribute-sets="hl-style">
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <xsl:attribute name="color">#602060</xsl:attribute>
</xsl:attribute-set>
```

This can be done from within XXE using Options → Customize Configuration → Customize Document Conversion Stylesheets.

1.2. Using the `indexterm` editor

This dialog box, displayed by menu item DocBook → Insert or Edit indexterm [6], allows to edit the selected `indexterm` element if any, or to create a new `indexterm` element and then insert it at caret position otherwise.



We'll explain with examples how to use the `indexterm` editor.

- If you want to get this kind of entry in your back of the book index:

```
P
Pet 12
```

  specify Term=`Pet`.

- Back of the book index:

```
P
Pet
    Cat 26
```

  specify Term=`Pet`, Term #2=`Cat`.

- Back of the book index:

```
P
"+" 54
```

  specify Term=`"+"`, Sort as=`plus`. Without this Sort as specification, the index entry corresponding to `"+"` would have been found in the **Symbols** category:

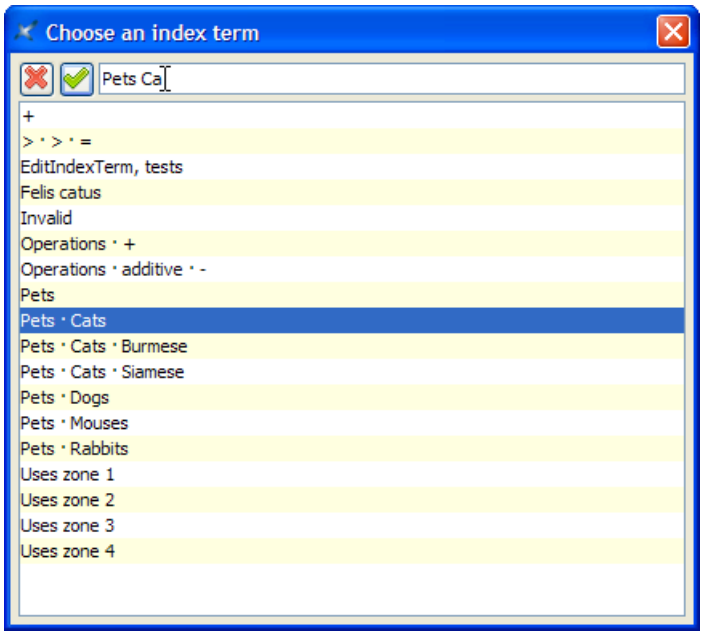
```
Symbols
"*" 53, 78
"+" 54
"-" 55, 91-95
...
```

- Back of the book index:

```
D
Domesticated animals 34 See also Pet
```

  specify Term=`Domesticated animals`, See also=`Pet`.

Note that the content of the See also field must refer to an existing index entry. That's why instead of typing "`Pet`", you can select this index entry by using the dialog box displayed by the Pick from list button found at the right of the See also row.



The above dialog box supports autocompletion. Note that if, for example, you want specify compound term "`Pet Cat Siamese`", you must type a space character between each simple term.

- Back of the book index:

```
I
IT See Information Technology
```

specify Term=`IT`, select "Redirect to the following term" then specify Redirect=`Information Technology`. (In the above example, notice that `IT` has no associated page number.)

Like See Also, the content of the Redirect field must refer to an existing index entry. Unlike See Also, a Redirect entry is merely a redirection to an actual index entry.

- Back of the book index:

```
O
Operation
    Additive
        "+" 87-90
```

1. Insert a first `indexterm` element at the beginning the range (this will give us page number 87).

   In order to do that, use DocBook → Insert or Edit indexterm and specify Term=`Operation`, Term #2=`Additive`, Term #3=`"+"`, Sort as #3=`plus`.

   Then check "Start range having the following name" and give your `indexterm` element an ID by specifying "`plus_reference`" in the Start range field.

2. Insert another `indexterm` element at the end the range (this will give us page number 90).

   In order to do that, use DocBook → Insert or Edit indexterm, check "End range having the following name" and specify the same ID, "`plus_reference`", in the End range field. All the other fields must be left blank.

   Note that instead of typing "`plus_reference`" in the End range field, you can select this ID by using the dialog box displayed by the Pick from list button found at the right of the End range field.

- Normally, that is, when "Mark current location" is selected, an `indexterm` element contributes to the back of the book index with its own page number. However, in some cases, it may be convenient to insert an `indexterm` at some place (typically in `chapterinfo`, `sectioninfo`, etc, elements) and specify that this `indexterm` corresponds to the page numbers of one or more other elements.

  Example: let's suppose that the `indexterm` element is contained in a `sectioninfo` element and that the chapter about dogs has "`ch.dogs`" its ID. Back of the book index:

```
P
Pet
    Dog 22
```

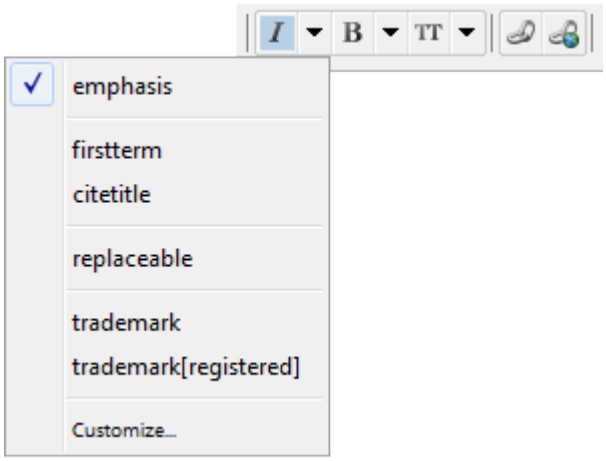
  specify Term=`Pet`, Term #2=`Dog`, select "Mark the elements having the following IDs" then specify Element IDs=`ch.dogs`.

  Note that instead of typing "`ch.dogs`" in the Element IDs field, you can select one or more IDs by using the dialog box displayed by the Pick from list button found at the right of the Element IDs field.

2. The DocBook tool bar

The DocBook tool bar starts with a number of “text style” toggles. These toggles emulate the behavior of the Bold, Italic, Underline, etc, toggles found in the tool bars of almost all word-processors. More information about text style toggles in About “text style” toggles in *XMLmind XML Editor - Online Help*.

Figure 1. Toggles found at the beginning of the DocBook tool bar



In the above screenshot, the caret is inside an `emphasis` element and the user clicked the arrow button next to a “italic text style” toggle.

I Toggle emphasis
: “Toggle” element `emphasis`. Next to this toggle is found an arrow button displaying a menu containing additional checkboxes for the following elements: `firstterm`, `citetitle`, `replaceable`, `trademark`, `trademark[registered]`.

B Toggle emphasis[bold]
: “Toggle” element `emphasis[bold]`. Next to this toggle is found an arrow button displaying a menu containing additional checkboxes for the following elements: `abbrev`, `guilabel`, `guibutton`, `guimenuitem`, `guisubmenu`, `guimenu`, `keycap`, `keysym`.

TT Toggle literal
: “Toggle” element `literal`. Next to this toggle is found an arrow button displaying a menu containing additional checkboxes for the following elements: `filename`, `sgmltag[element]`, `sgmltag[attribute]`, `sgmltag[attvalue]`.

 Toggle link
“Toggle” element `link`.

 Toggle ulink
“Toggle” element `ulink`.

 Add para
Add a `para` element after node selection or after caret at a location where it is valid to do so.

 Add list item
Add a `listitem` or `varlistitem` element after current list item. For this command to work, suffice to click anywhere inside an `itemizedlist`, `orderedlist` or `variablelist` element.

 Add itemizedlist
Add an `itemizedlist` element after node selection or after caret at a location where it is valid to do so.

 Add orderedlist
Add an `orderedlist` element after node selection or after caret at a location where it is valid to do so.

 Add variablelist
Add an `variablelist` element after node selection or after caret at a location where it is valid to do so.

 Add programlisting
Displays a menu which allows to add several styles of elements containing preformatted text after node selection or after caret at a location where it is valid to do so.

 Add table
Displays a menu which allows to add several styles of tables after node selection or after caret at a location where it is valid to do so.

 Add image
Displays a menu which allows to add several styles of figures and images after node selection or after caret at a location where it is valid to do so.

 Add section
Displays a menu which allows to add chapter or section elements after node selection or after caret at a location where it is valid to do so.

3. Custom bindings

Keystroke	Action
F2 Up	Same as menu item Move Up [6].
F2 Down	Same as menu item Move Down [6].
F2 Left	Same as menu item Promote [6].
F2 Right	Same as menu item Demote [7].
Enter	Insert a newline character if possible. Otherwise, if caret is at the beginning of a paragraph, list item or a few other kinds of block, insert same block before. Otherwise, if caret is at the end of a block, insert same block after. Otherwise, split block.
Del	Delete selection if any. Otherwise, if caret is at the end of a paragraph, list item or a few other kinds of block,

Keystroke	Action
	join with following block. Otherwise, delete character following caret.
BackSpace	Delete selection if any. Otherwise, if caret is at the beginning of a paragraph, list item or a few other kinds of block, join with preceding block. Otherwise, delete character preceding caret.
Ctrl+**Enter**	Add same block after the paragraph, list item or a few other kinds of block which is the ancestor of selected node.
Shift+**Ctrl**+**Enter**	Add same block before the paragraph, list item or a few other kinds of block which is the ancestor of selected node.
Application Event	**Action**

Drop an object.	• On a `ulink` element, change the value of the `url` attribute to the dropped string.
	• On an `image` element, change the value of the `fileref` attribute to the dropped string.
	• Elsewhere
	  • If the object being dropped represents an URL or an absolute filename, open the corresponding document in XMLmind XML Editor.
	  • Otherwise, paste the dropped text or XML at or after the drop location.

Drag one of the “handles” displayed around an image. (The “handles” are displayed after clicking on the image.)	Resize the image, but always preserve its aspect ratio. Pressing Ctrl (Cmd on the Mac) while dragging the handle allows to distort the image.
Drag a separator found between two table columns.	Resize the table column. More precisely this gives an appropriate proportional width (e.g. `<colspec colwidth="3*">`) to *all* table columns.

4. Table rendering

The following attributes are either completely ignored or partially supported. All other attributes are supported.

Attribute	Support
`table` (or `informaltable`) `orient`	Ignored.
`table` (or `informaltable`) `pgwide`	Ignored.
`colspec colwidth`	All forms including "`2*`" or "`3*+1pc`" are supported. Coefficients of "`*`" are always converted to integers. Examples: "`2.5*`" is equivalent to "`2*`". "`3.95*+0.5in`" is equivalent to "`4*+0.5in`". A column must contain at least one cell with a column span equal to 1 for the `colwidth` attribute to have an effect.
`entry rotate`	Ignored.
`align`	Values `justify` and `char` are rendered like `left`.
`char`	Ignored. See `align`.

Attribute	Support
charoff	Ignored. See align.

4.1. HTML tables

DocBook supports HTML tables as well as CALS tables (that is, ``traditional" DocBook tables) starting from version 4.3. Therefore XMLmind XML Editor also supports both table models. See Appendix A, *Table rendering* in *XMLmind XML Editor - XHTML Support* for details.

The only limitation is that mixing both HTML and CALS content models in the same table or informaltable is *absolutely not supported* by table rendering code and by table editing commands, even if this is allowed according to the DTD V4.3.

Example 1: an informaltable contains tr child elements. In such case, the informaltable is an HTML table. Setting attribute frame to topbot on this informaltable will have absolutely no visual effect.

Example 2: a table has a child tgroup element which itself contains a tbody with row/entry descendants. In such case, the table is a CALS table. Adding a thead having tr/td descendants before the tbody of the tgroup would lead to catastrophic results. Fortunately, the DocBook configuration of XMLmind XML Editor makes it hard to do this unintentionally.