O'REILLY® Short Cuts

Get Started Writing in DocBook

Released: April 2008

***DocBook** [**http://docbook.org**] is an OASIS standard for XML that is ideal for writing long, technical documents that have complex structure and cross-references. With its expressive tagging, DocBook provides the equivalent of a high-resolution image that is able to be converted into many alternate formats without losing its fidelity. DocBook can be rendered as PDF for printing, HTML, man pages, audio, or even Braille. This versatility and re-usability makes DocBook 4.4 XML the preferred format for O'Reilly books.*

Contents

Why DocBook?

Why not troff? Why not LaTex?

We've got nothing against LaTex, troff, or any other formatting markup system. But typesetting markup like LaTex is inherently focused on formatting—font size, margins, alignment, etc. We'd rather you spent your time focused on the semantic structure of your book (how sections are divided, which information goes in a sidebar versus a note, and so on), and that's where DocBook shines. In the same way most well-designed web sites separate content from formatting using XHTML and CSS (XHTML for the content, and CSS for the formatting), DocBook lets us abstract formatting decisions away from content decisions.

Particularly as O'Reilly expands its efforts to provide content in multiple formats and at multiple stages of the content's life cycle, cost-effectively generating multiple, distinct output formats from the same source document becomes critical, even though it means losing some degree of granular control over the output. What does that mean for your book? It means that you'll be able to view drafts of your book throughout the authoring process that are formatted much as they will be for print, and it means once your book is finished, it can go live on the Safari online library site almost immediately, rather than needing to first be converted into DocBook from another format, which can take several weeks.

DocBook and Subversion

One of the benefits of working in DocBook is that you can take advantage of O'Reilly's Subversion repositories to maintain your files. Subversion is an open source version control system that keeps track of the changes you make to your book. Throughout the writing process, you can "commit" revised versions of the book files to the repository with a log message, which will be associated with a revision number. Among other features, Subversion allows you to revert back to any revision of your book, as well as run a diff to compare two different versions of a file.

Subversion also supports multiple working copies of the same project, which means you can have multiple authors collaborating and making changes to the same set of files simultaneously, and Subversion provides the functionality to merge, diff, or revert the revisions you make when you commit the files to the repository.

For more information on Subversion and a command reference, see O'Reilly's **Version Control with Subversion** [***http://svnbook.red-bean.com/nightly/en/svn-book.html?***].

Using an O'Reilly Subversion Repository

Once you have spoken with your editor and decided that you will write in DocBook, email the Publishing Services team at O'Reilly at *toolsreq@oreilly.com*; we can create a Subversion (SVN) repository for you on an O'Reilly server that contains *book.xml* and other files to use as a template for getting started. There are several benefits to using an O'Reilly SVN repository:

- It is easier to exchange files with your editor and coauthors for review.
- It is easier for the O'Reilly Digital Content team to help you with any DocBook questions you may have along the way.
- You can trigger PDF builds of your book every time you commit changes.
- The files are stored in a safe and secure location that is backed up regularly.

Once you check out the template files from your SVN repository, you can open the *ch01.xml* file and begin typing your first paragraph. The basic `DOCTYPE` [*http://www.docbook.org/tdg/en/html/ch02.html#d0e2566*] and other metadata will already be in the *book.xml* file for you, so you can concentrate more on writing your book and less on XML markup.

After you complete your first chapter and save it, you can create a new, separate file for chapter 2, modeled on your chapter 1 file. Name your new chapter with O'Reilly's standard naming convention—*ch02.xml*—and then add this chapter to the book file that is in your book directory, as described in the section below (**Organizing Your Files**). As you create new chapters and add them to the book file, you'll build a complete book document that can be published in a variety of formats.

Triggering PDF builds of your book

You can generate a fresh PDF of your book every time you commit changes to the SVN repository. To do so, run the following command:

```
$ svn commit -m'Made some really important changes to Chapter 3.; orm:commitpdf'
```

Then to get the PDF, just run `svn up` on your working copy about 5–10 minutes after committing your files. The PDF will be downloaded to the *pdf* directory of your working copy. If there are any problems in generating the PDF, you'll instead get a *.buildlog* file in the *pdf* directory that lists the errors.

The text `orm:commitpdf` triggers the PDF build, and it can be used with any commit from your working copy, not just the *book.xml* file or a chapter file. So if you would like to generate a fresh PDF without making any changes to your book files, you

can add a separate scratch file to the */current* directory (or a subdirectory) and just make modifications and commit the changes to it with the `orm:commitpdf` string.

Triggering PDF builds of just a chapter

If instead of generating a PDF of the whole book when you commit changes, you'd prefer to generate a PDF of a single chapter, you can use the following commmand:

```
svn commit -m'Committing changes to Chapter 4; orm:chapterpdf @id_for_chapter'
```

Here, the magic commit hook is `orm:chapterpdf` instead of `orm:commitpdf`. You'll also need to include *`@id_for_chapter`*, which is the value of the `@id` attribute of the `<chapter>` in the XML file you want to create a PDF of.

> **Warning**
>
> The text string `orm:chapterpdf` *`@id_for_chapter`* must appear at the *end* of your commit message. If it does not, the commit hook may not work.

The `@id` value of a chapter occurs in the first five lines of the XML file, so a quick way to retrieve the `@id` via the command line is:

```
$ head -n 5 ch04.xml
<?xml version="1.0"?>
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
"http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd">
<chapter id="practice_1_automated_testing">
```

Organizing Your Files

O'Reilly prefers that each chapter is its own separate file and you may find this system much easier to manage than a single, consolidated book file. As you will see in the *book.xml* file that O'Reilly provides for you, it's best to have the book file just contain the book metadata and no actual content. The chapter files' content is referenced and included by **using `XIncludes`** [***http://www.sagehill.net/docbookxsl/ModularDoc.html#UsingXinclude***]. Each chapter is its own full DocBook document with its own `DOCTYPE` declaration, which will make validation easier.

> **Note**
>
> Whether you structure a document with `XIncludes` or not, note that the XML `id` attributes you assign to sections, figures, etc., all need to be unique across the entire book. This is important for cross-referencing and it affects the validity of your XML document.

Will What I See in the XML Editor Mirror the Final PDF?

DocBook markup specifies the structure and semantics of your document, but not the appearance. DocBook isn't stored as WYSIWYG files, so it can display differently in a program like the XMLmind XML Editor (XXE) than it will after rendering to PDF. This not only means that fonts and formatting will display differently, but line breaks may differ as well.

The XML tags that are applied to the elements of your text are used in combination with XSLT stylesheets to render the final appearance of your book files. The XSLT stylesheet is similar to a CSS stylesheet for HTML. It sets up how each element and attribute in your book will look when rendered as a PDF. If you are using XXE, you can apply customizations to to have the display more closely mirror how the final product will look. See the later section **Customizing XXE** for information on downloading and installing these customizations.

The toolchain that transforms your DocBook into its final PDF form is complex, involving layers of XSL-FO stylesheets and an FO to PDF processor, so if there is something that you'd like to see included in our rendering that isn't showing up as you'd expect in your PDF, please contact *toolsreq@oreilly.com* and we will work with you to tweak the stylesheets as needed.

Expressing Code in DocBook

One of the more challenging aspects novices face when working with DocBook is correctly expressing computer code. In general, code blocks should be enclosed inside `programlisting` elements, which may in turn be inside of `example` elements (use an example when you want the code to have a caption you can reference elsewhere in your text). Within a `programlisting`, whitespace will be preserved as is. But you still need to escape any characters that have special meaning in XML, such as `<` and `>`. These obviously come up quite a bit in code, like `x <= 1`.

If you don't want to bother remembering to escape these things, you can use an `XInclude` section (O'Reilly also uses `XIncludes` to organize the book.xml, as described earlier in **Organizing Your Files**). In an `XInclude` section, the code blocks are referenced in `<xi:include>`s and are ignored by the XML parser (they're not dropped or anything, but the parser doesn't try to interpret them as XML):*

* Or you can use a `CDATA` section. In a `CDATA` section, any text between `<![CDATA[` and `]]>` is ignored by the XML parser. If you're using XXE, you can't use `CDATA` sections, but on the other hand, you don't need to worry about escaping any special characters (as XXE takes care of that for you), which is probably the better end of the bargain.

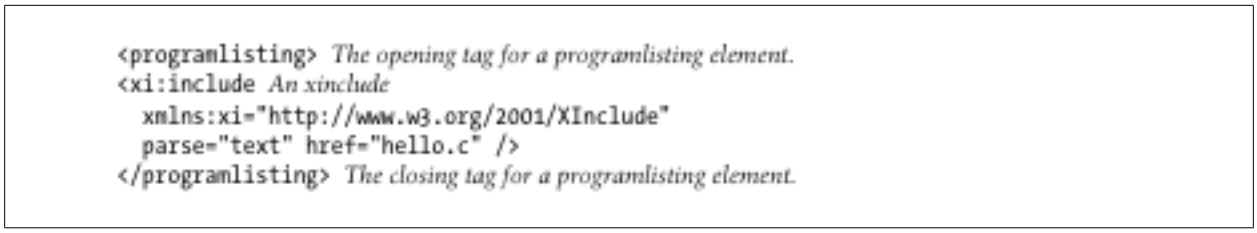

Figure 1. Program listing with line annotations

```
<programlisting>
<xi:include
  xmlns:xi="http://www.w3.org/2001/XInclude"
  parse="text" href="hello.c" />
</programlisting>
```

Annotating Your Code in DocBook

O'Reilly's stylesheets support two different types of DocBook markup for annotating code blocks in `programlisting` elements: line annotations and callouts. Here's some more information on each. For more information, you may want to see ***http://www.sagehill.net/docbookxsl/AnnotateListing.html***

Using line annotations in programlistings

You can add `lineannotation` elements in `programlisting` elements to place annotations adjacent to your code lines. Here's an example of a `programlisting` that contains line annotations:

```
<programlisting> The opening tag for a programlisting element.
<xi:include An xinclude
  xmlns:xi="http://www.w3.org/2001/XInclude"
  parse="text" href="hello.c" />
</programlisting> The closing tag for a programlisting element.
```

O'Reilly's stylesheets will, by default, render the `lineannotation` elements in italic. **Figure 1** shows how the above code example renders using our standard animal book template.

Using callouts in programlistings

If you want to have cross-references to specific lines of code outside your `programlistings`—for example, in discussion text following the code—you can use callouts to achieve this effect. To add callouts, you need to add `co` elements to `programlistings`, which add callout markers inline in code. Then you need to create a `calloutlist` element, which includes a set of `callout` elements that include the explanation text. Here's the example from before, reconfigured to use callouts instead of line annotations:

```
<programlisting> ❶
<xi:include ❷
```

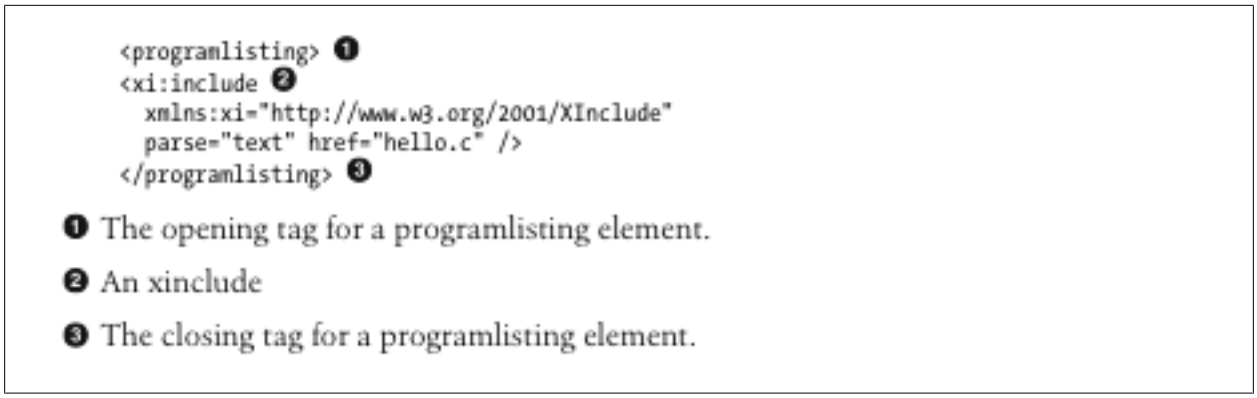

Figure 2. Program listing with callouts

```
  xmlns:xi="http://www.w3.org/2001/XInclude"
  parse="text" href="hello.c" />
</programlisting> ❸
```

❶ The opening tag for a programlisting element.
❷ An xinclude
❸ The closing tag for a programlisting element.

Each `co` element requires a `linkends` attribute that points at the `callout` elements that refer to it, forming a link between the callout marker and the callout text. Similarly, each `callout` element requires an `arearefs` attribute that points at the `co` elements it refers to, forming a link between the callout and the callout marker.

> **Note**
>
> Please *do not* use `areaspec`/`area`/`areaset` elements to specify regions for callouts.

Figure 2 shows how the above code example renders using our standard animal book template.

The callout markers in both the code and the callout list will be rendered as clickable bidirectional cross-references.

For more information on DocBook callout markup, see ***http://www.sagehill.net/docbookxsl/AnnotateListing.html#Callouts***.

> **Note**
>
> If you're avoiding callouts because you'd like them to render in a different way, please let us know. We're interested in suggestions for making callouts more usable for the reader.

Other ways of annotating code

DocBook XML supports adding line numbering to `programlisting` elements, via the `linenumbering` attribute. However, O'Reilly's stylesheet doesn't support rendering of line numbers in this way, because they don't allow for good cross-referencing and can potentially cause problems when code is revised. For example, let's say you used line numbering, and then in the text below you reference something in line 17. But then, later in the process, a tech reviewer notes a bug in lines 14-16, and you decide you don't need those lines anyway, so you delete them. Line 17 then becomes line 14, so you'd have to change that reference in the text below, which is easy to overlook and is unlikely to be caught in production.

If you want to cross-reference code blocks by line number, we recommend using callouts instead.

What Type of XML Editor Should I Use?

You can use whatever tool you like to write in DocBook XML, but O'Reilly recommends and supports XXE; this document specifically references XXE, and much of our documentation is directed toward users of XXE. If you'd like an installer for that to use with our customizations, please let us know. We'll need to know the platform you're running. (More experienced users who write in DocBook extensively may want to consider purchasing the Professional Edition.)

Some of the advantages of using XXE include:

- Validation on the fly
- GUI interface is similar to other popular word-processing programs
- Support from the O'Reilly content team

More details on using XXE can be found in the last section of this document (**Working with XMLmind XML Editor**).

> **Note**
>
> There are, of course, a lot of other editors that are good at handling DocBook. Depending on your background, you may find Emacs' nXML mode useful or **vim with keybindings** [***http://www.pinkjuice.com/howto/vimxml/***]. Some authors have expressed a preference for oXygen (***http://www.oxygenxml.com/***) over XXE, especially given oXygen's built-in SVN support and recent improvements in its WYSIWYG-mode.
>
> A fairly complete list of authoring tools is available on the DocBook wiki: ***http://wiki.docbook.org/topic/DocBookAuthoringTools***. Please note

that non-validating editors will place all of the burden of validation on you, the author. **Validation** [***http://www.sagehill.net/docbookxsl/ToolsSetup.html#Validation***] is not difficult, but invalid documents will not be accepted.

Validating Your XML Files

We require that XML files submitted to Production are valid DocBook 4.4. Many XML editors (like XXE or oXygen) will validate your XML for you on the fly. Another option is to use our PDF build toolchain (see **Triggering PDF builds of your book**) to do a validity check.

Another option for validating on the command line is `xmllint`. `xmllint` is preinstalled on Mac OS X, and is available for a wide range of operating systems as part of the *libxml2* package from xmlsoft (***http://xmlsoft.org/downloads.html***).

To validate your book.xml file using `xmllint`, run the following command:

```
$ xmllint --postvalid --xinclude --noout book.xml
```

If the book files are valid, no output will be produced. Otherwise, validity errors will be printed to standard output.

Validating Locally using an XML Catalog

By default, xmllint validates over the Internet against the DTD located at the URL listed in the XML file's DOCTYPE (***http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd***). But if validating over the Internet is undesirable, you can set up a local copy of the DocBook 4.4 DTD and a local XML catalog for validation purposes.

The easiest way to install a copy of the DocBook 4.4 DTD is via your favorite package manager. It is available via **MacPorts** [***http://trac.macports.org/browser/trunk/dports/textproc/docbook-xml-4.4/Portfile***] and **Cygwin** [***http://cygwin.com/packages/docbook-xml44/docbook-xml44-4.4-1***], among other packaging systems.

Many packaging systems will create the necessary *catalog* file for you, but if you need to write one yourself, you can find more details on doing so at ***http://www.sagehill.net/docbookxsl/WriteCatalog.html*** and a sample catalog file at ***http://www.sagehill.net/docbookxsl/ExampleCatalog.html***.

The default location where `xmllint` will look for the catalog file is */etc/xml/catalog*. Alternatively, you can set the environment variable `XML_CATALOG_FILES` to define another default location for catalog files.

Once you've got the DTD and catalog set up locally, you can validate as follows:

```
$ xmllint --postvalid --xinclude --noout --catalogs book.xml
```

Keep It Simple

"Keep it Simple" sounds a bit silly when referring to something as complex as DocBook, but the point here is that even though DocBook offers over 400 elements, you'll likely need only a fraction of them. For example, you can safely stay away from the `confsponsor`, `inlinemediaobject`, and `seriesvolnums` elements. DocBook is meant to be comprehensive across a universe of technical documentation. We're only dealing with a subset: content meant for expression in an O'Reilly title. Practically speaking, you'll mostly use elements very similar to the standard HTML elements, like `itemizedlist` and `table`.

An article by Keith Fahlgren on XML.com examines the most commonly used DocBook elements in O'Reilly books and may be useful to you: **DocBook in the Wild: A Look at Newer Content** [***http://www.oreillynet.com/xml/blog/2007/05/docbook_elements_in_the_wild_a.html***].

Sample Markup and PDFs

This document is formatted as an article, so it contains some markup not found in a book. While it contains many of the tags that you will use within a book, it might be useful to check out some of the examples of books we have posted.

You can find sample markup to use as a model in your own markup here: ***https://prod.oreilly.com/external/tools/docbook/prod/trunk/samples/*** (username: `guest`; empty password)

The samples include a standard chapter as well as more complex markup, such as reference markup (found in the nutshell directory). There is also an directory for ongoing research and development of new features that authors have requested.

See Also

See the following for more information:

DocBook: The Definitive Guide

http://www.docbook.org/tdg/en/html/docbook.html

DocBook Basics and References

http://www.dpawson.co.uk/docbook/reference.html

DocBook.org

http://www.docbook.org/

XXE Documentation

http://www.xmlmind.com/xmleditor/documentation.shtml

The DocBook DTD

http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd

Working with XMLmind XML Editor

The information in this document is geared toward using XXE as part of O'Reilly's workflow. For more general information about getting started with XXE, a nice starting point is the official XXE documentation found at ***http://www.xmlmind.com/xmleditor/documentation.shtml***. There is also a **useful PDF** [***http://www.xmlmind.com/xmleditor/_distrib/doc/quickrefcard/quickrefcard-Letter.pdf***] on the XXE site. Additionally, you may want to take a look at the XML.com article **Getting Productive with XMLMind** [***http://www.xml.com/pub/a/2007/06/20/getting-productive-with-xmlmind.html***] by O'Reilly authors James Elliot and Marc Loy. Their article was referenced during the creation of this document and is a good resource for beginners.

Using XXE

A program like XXE puts a word-processing face on XML, making it more user-friendly while keeping the markup valid. The Element Bar (see **Figure 3**), or Node Path, at the top of your screen lets you know where you are in the XML structure at any given point (whether you are in a section title, paragraph, list, etc.). Clicking on the element will display a box around that entire element in your document. Conversely, when you click in your document, the Element Bar will show you where you are in the hierarchy of your document.

Figure 3. The Element Bar shows the XML hierarchy

You can also move up and down your hierarchy by using the movement buttons in the upper right corner of the screen.

To see a view of the document structure in tree form, go to View→0 (no style sheet). This is shown in **Figure 4**. If you'd like to see both the DocBook and the tree view,

go to Options→Preferences→Window→Show both tree and styled views, as shown in **Figure 5**.

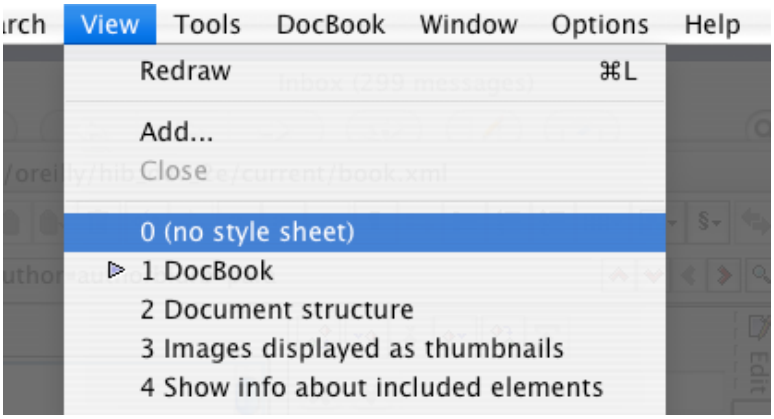

Figure 4. View menu

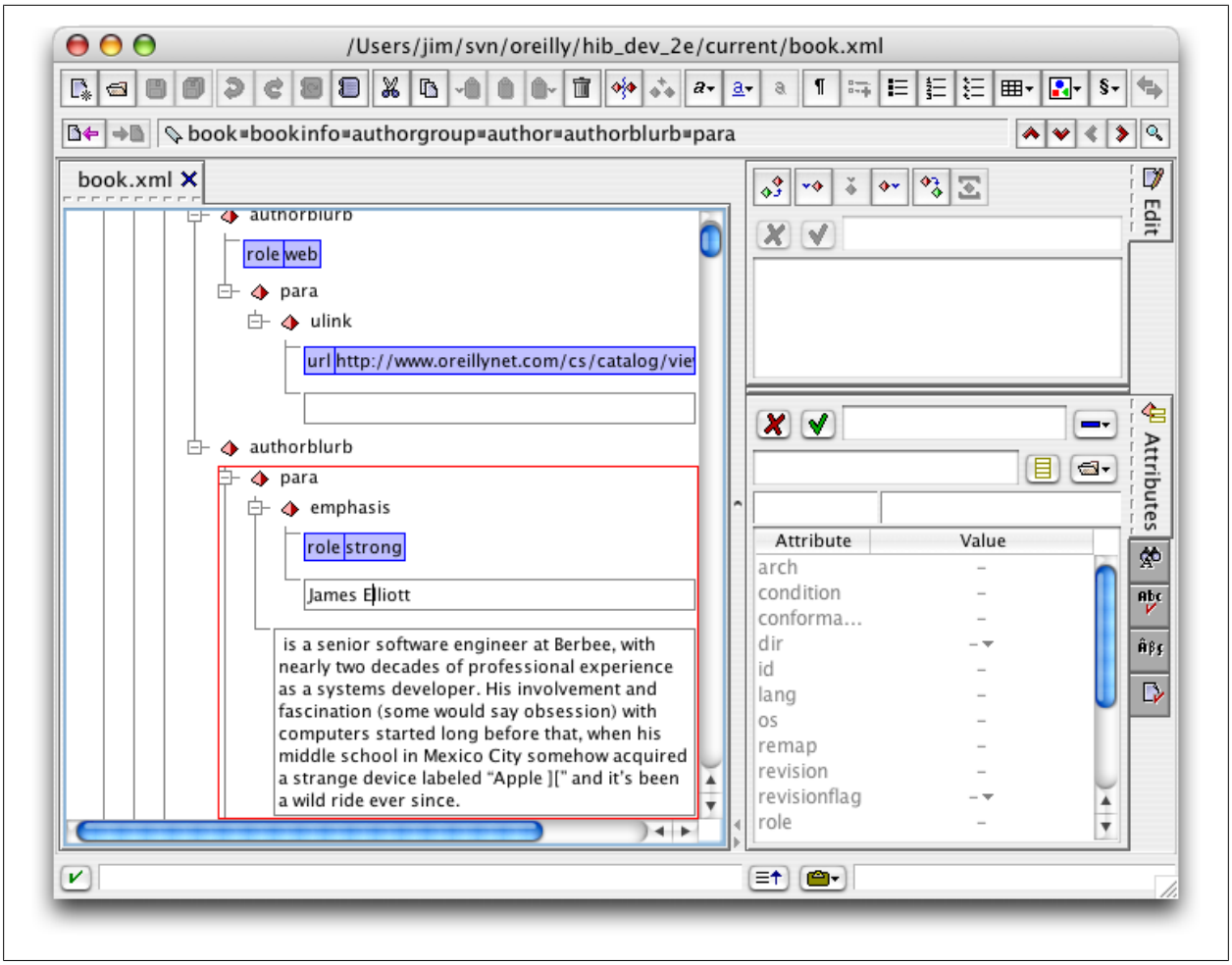

Figure 5. Tree view

> **Note**
>
> When editing the book file itself, note that edits cannot be made to individual chapters there, but must be entered in the individual chapter file. You can open the referenced file by placing your cursor in it and hitting Ctrl-Shift-E. You can also edit the chapter file by placing your cursor in the specific chapter and choosing Edit→Reference→Edit Referencing Document.

Handy Tools

When creating your DocBook manuscript, there are a number of tools you may want to use from within XXE. Here's a list of a few of our favorites:

Search & Replace

When writing, you may want to search or search and replace certain words or phrases. This can be done by clicking the Search tab in the attribute menu (shown in **Figure 6**. You can chose to ignore case or search by part of a word, and you can also choose to skip over an instance or every instance of the word within a certain element.

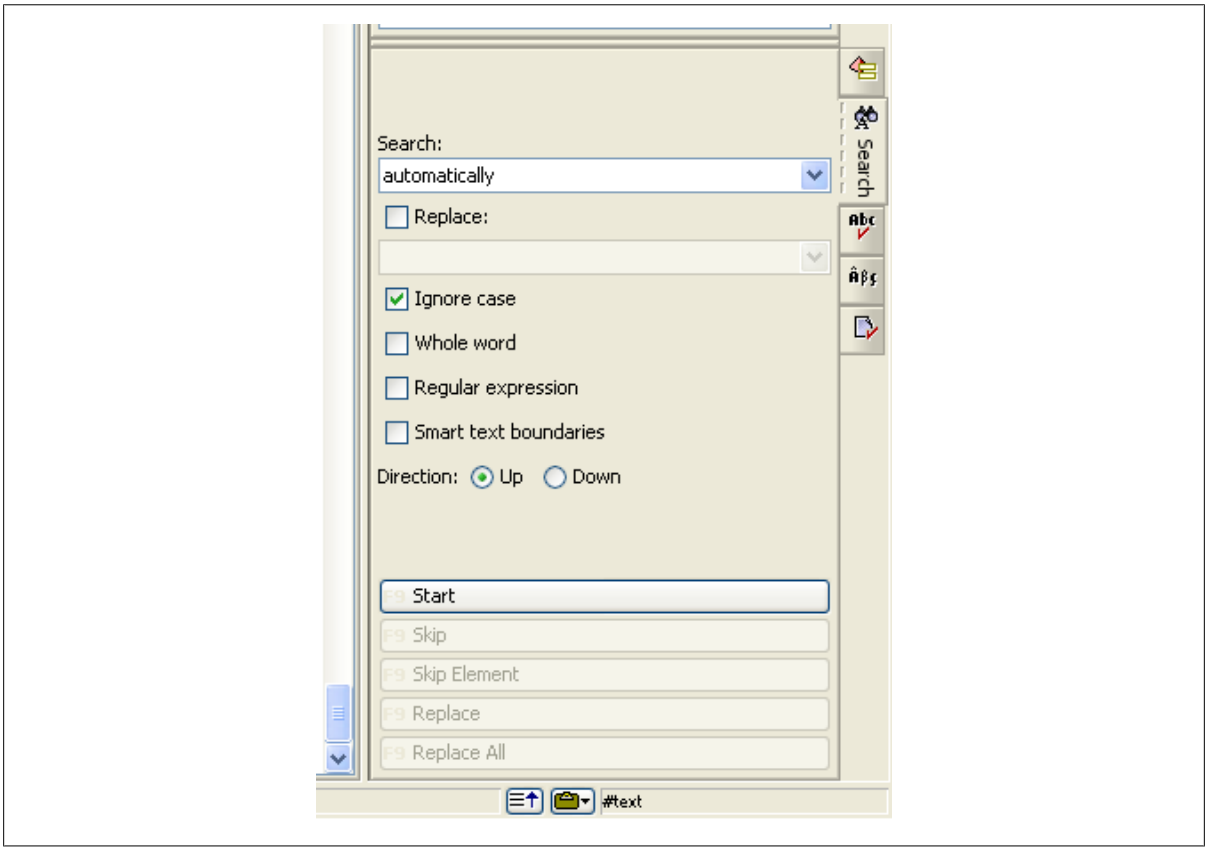

Figure 6. weSearching a document

Spelling

Another tool that comes in handy is the spell check (illustrated in **Figure 7**, also found in the attribute menu. You can also skip elements here, so if you want a word to be capitalized in section headers, but not within text, you can skip the instances in section headers.

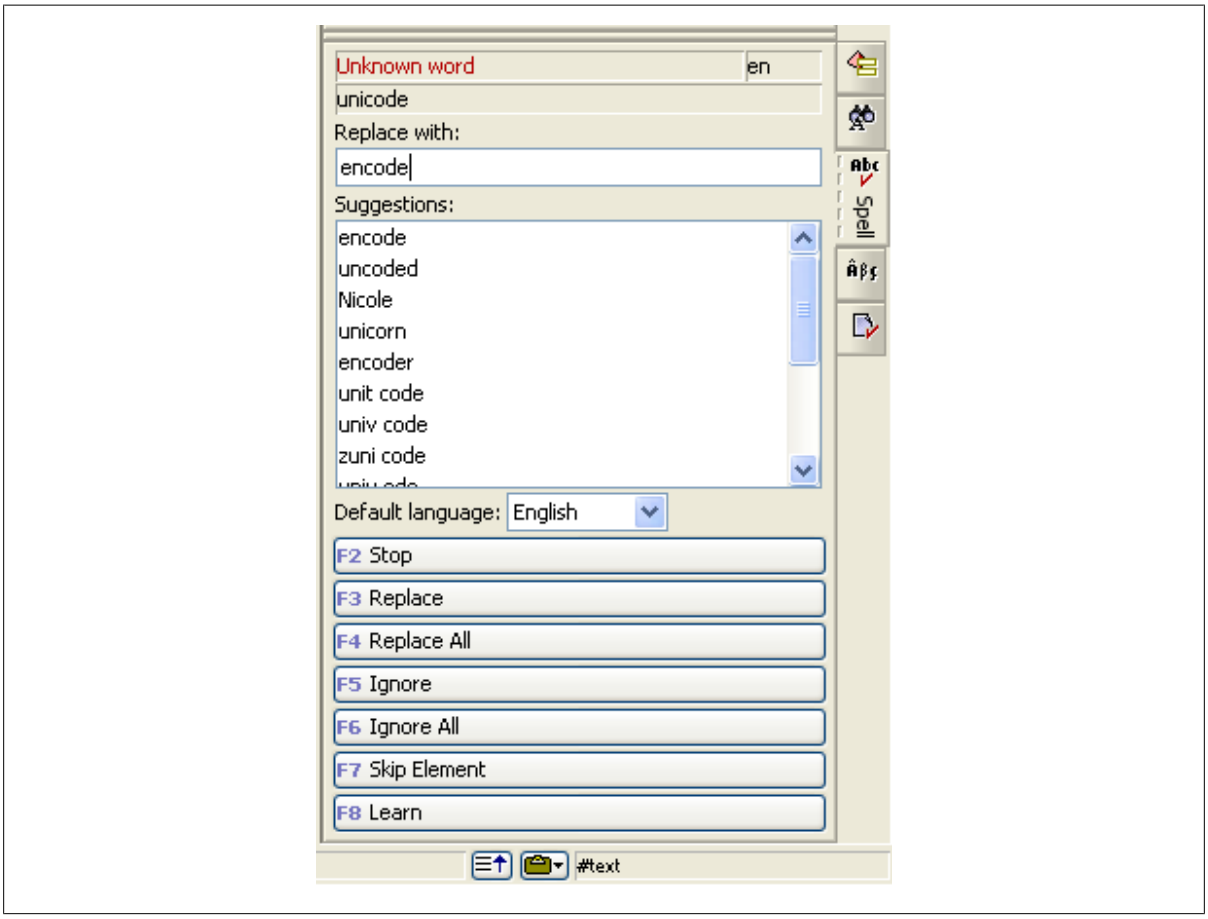

Figure 7. Spell check

Special Characters

If you find that you need to insert a character that isn't a standard keyboard character, you can choose it from the Characters tab, displayed in **Figure 8**. This character palette will also display the Unicode number for a character if you hover over the symbol. If you can't find a symbol, but know the Unicode number, you can type in the Unicode number to bring up the appropriate symbol.

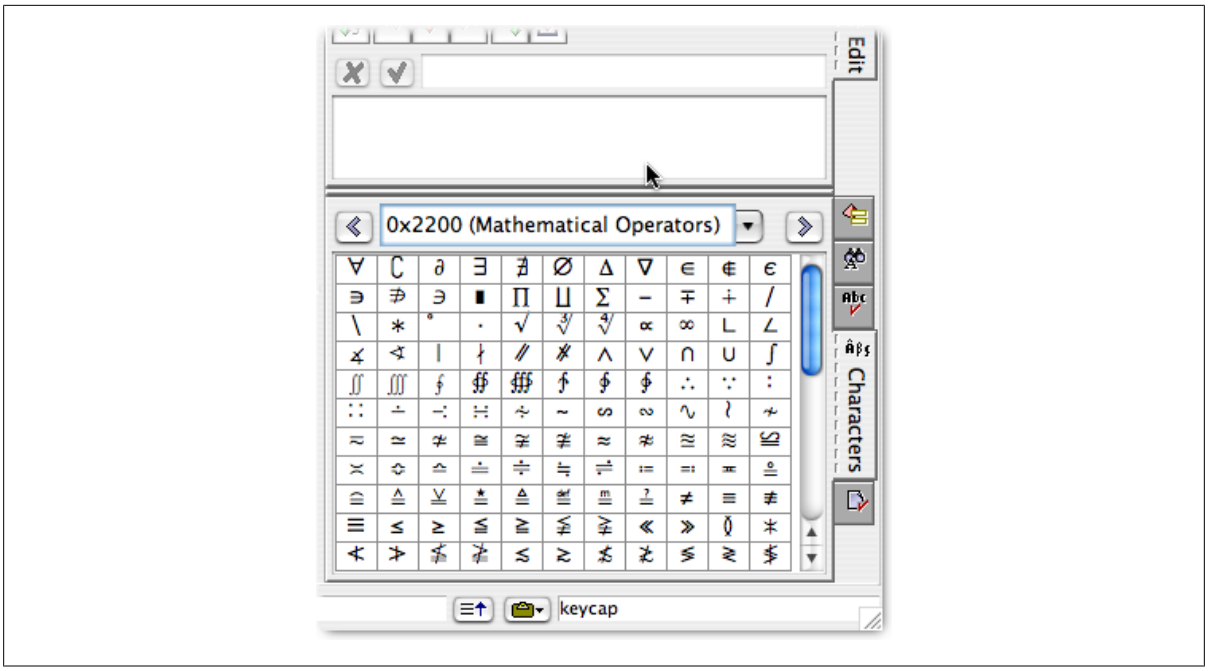

Figure 8. Character palette

Validation

Though tools like XXE are checking the validity of your document as you go, you can also chose the Validity tab to display any errors that might have cropped up in the document.

> **Note**
>
> You can ignore errors about broken cross-references (like those shown in **Figure 9**) if the reference is to something in a different chapter file. These types of errors will disappear once you look at the book file with all chapters included.

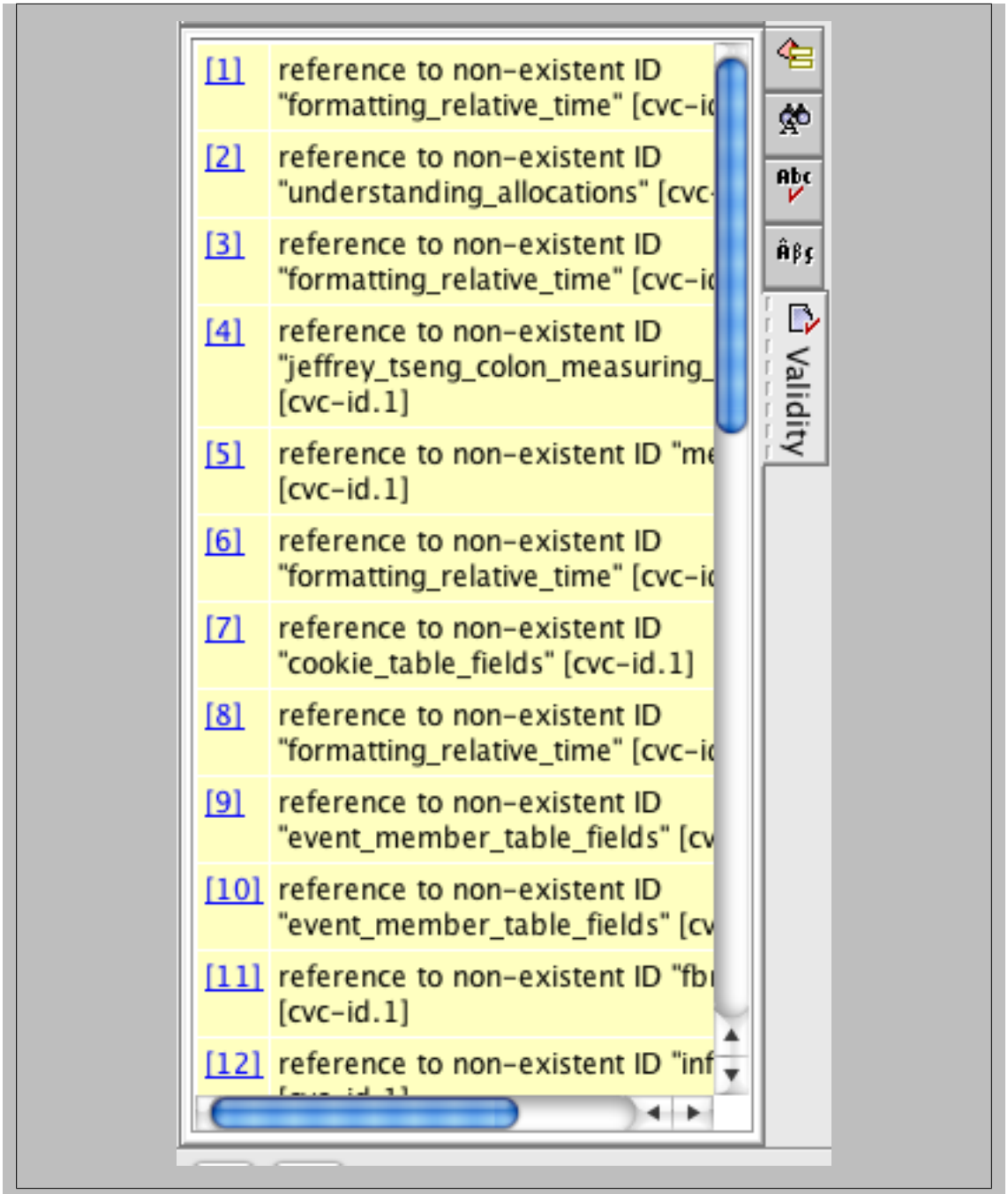

Figure 9. Validation tab with broken cross-reference errors

Note to self…

If you would like to put in a comment as a reminder to yourself or someone else who will be working with your files, you can choose Edit→Comment→Insert Comment (as shown in **Figure 10**. A comment will be inserted with a yellow background to help it stand out. See **Figure 11** for an example. You don't need

to worry about removing comments, since the stylesheets know not to display them in final copy.

Another option is entering <remark>s in the text. These are displayed in a different font and color so as to stand out.

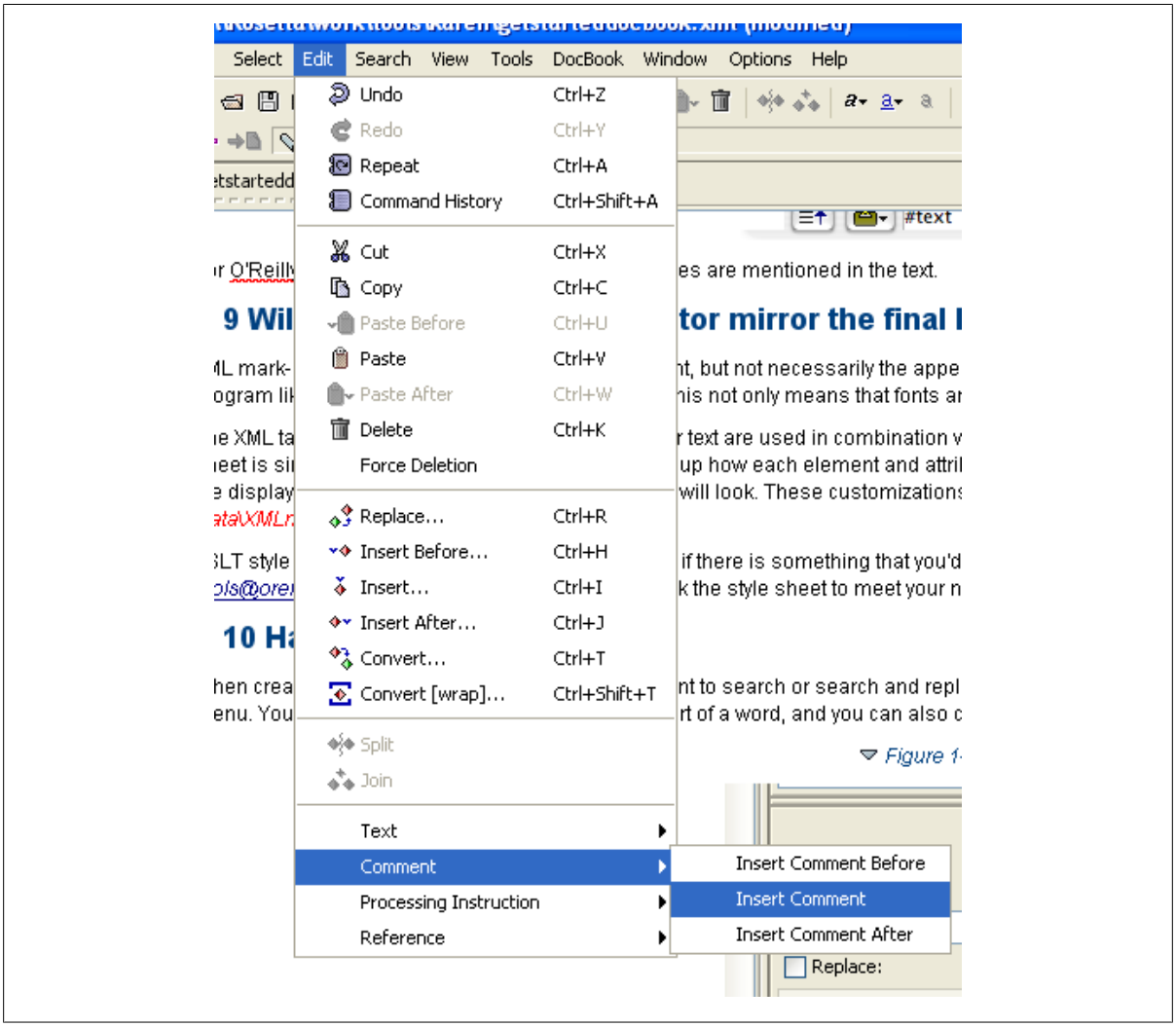

Figure 10. Inserting a comment

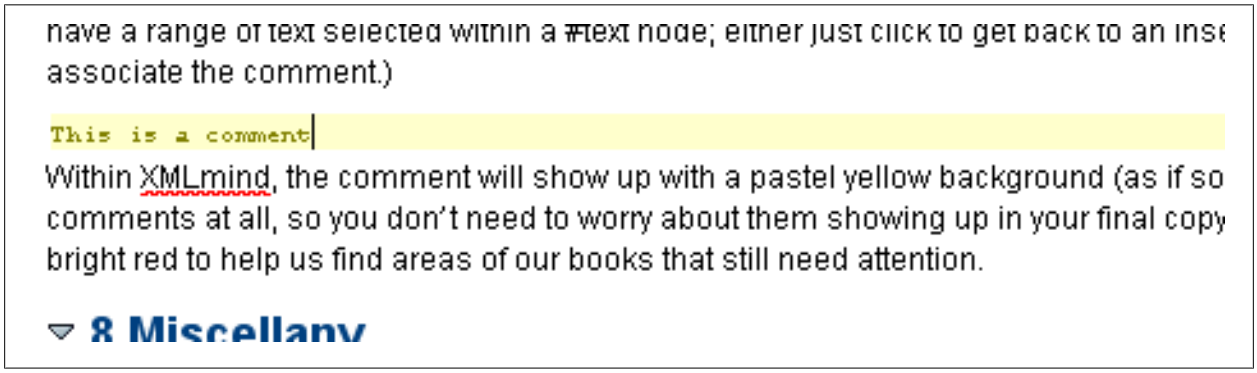

Figure 11. Comment within a document

New sections

When adding sections to your document, choose the `section` tag as shown in **Figure 12**. If the new section comes above or below where you want it to in the hierarchy, you can simply use DocBook→Promote and DocBook→Demote to move it to the appropriate level. The buttons used are shown in **Figure 13**. By promoting sections, you can move a section that is currently a second-level, or B-head to be a first-level, or A-Head. Demoting will move an A-head down to a B-head.

The Move Up and Move Down functions can be used to move elements past each other as well, though they're mostly used to move sections around.

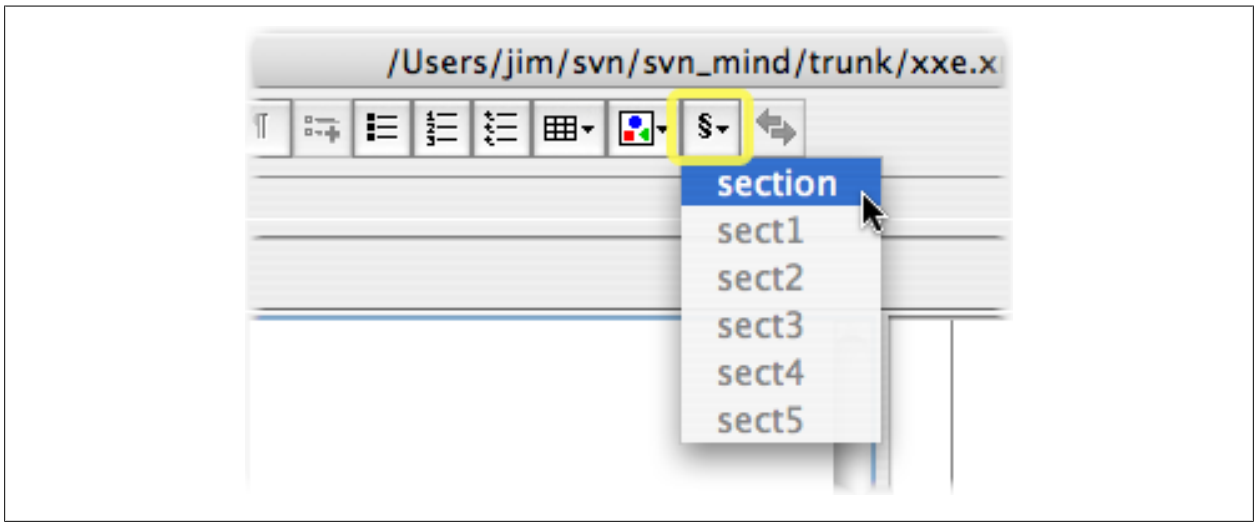

Figure 12. Adding a new section

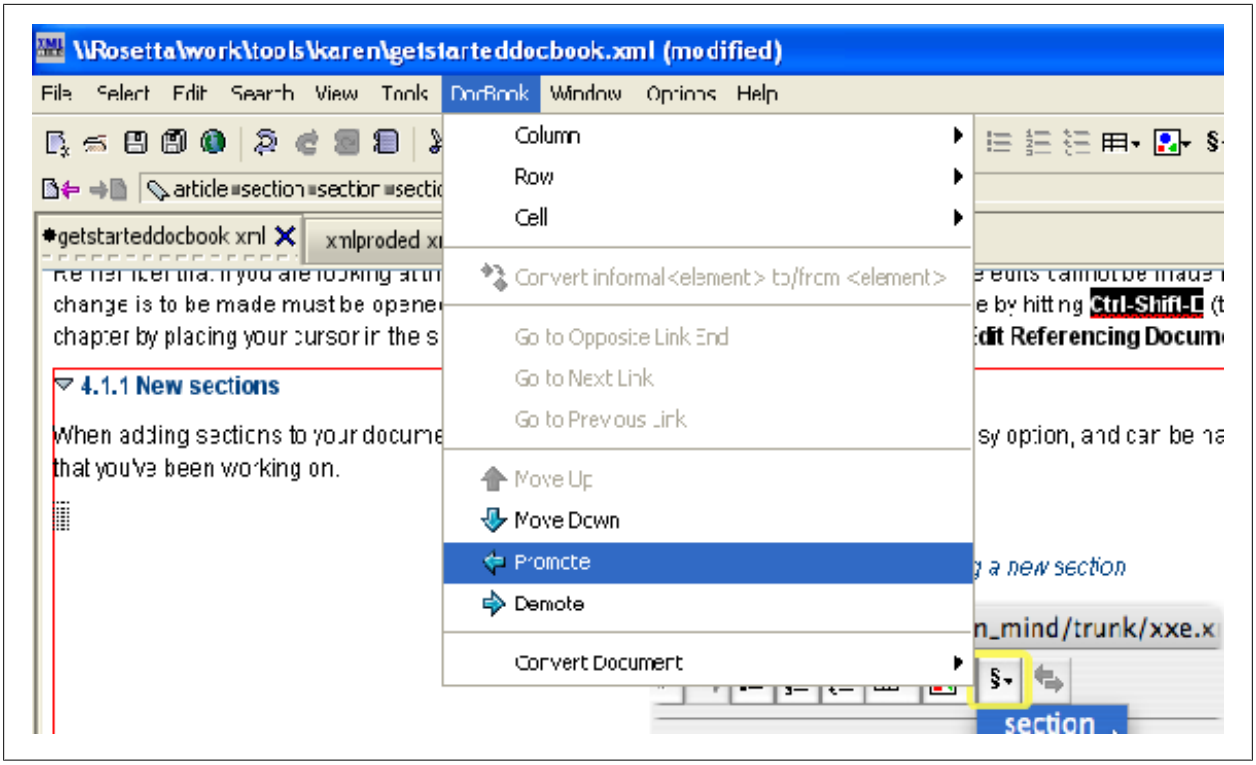

Figure 13. Promoting and demoting sections

When you create new sections, your sections will automatically renumber in XXE according to where the new section has been entered. The same applies to figure and table numbering.

Splitting and merging sections

If you need to split a section in two, select the paragraph that you want to start the new section by choosing the `para` element and then choose DocBook→Promote to create a new section with that paragraph.

Merging two sections can be a bit more difficult. If you are just moving a few paragraphs into another section, you can cut and paste the text from one section to the other and delete the second section title. If you have a more complex section to move, use the steps below:

1. Make sure the two sections you want to join are adjacent to each other, with the section heading that you want to keep positioned first.
2. Select the first `section` from the element bar.
3. Now Select→Select All Children (boxes will surround all the individual elements that make up that section).
4. Choose Edit→Copy (Cut isn't an option for a complex section).

5. Select the `title` element of the second section you want to join.
6. Choose Edit→Paste and paste section one into the `title` element of the first `section`. This will replace the second section's title with the first section's title, and insert all the other elements of the first section at the beginning of the second section.
7. Now go back and delete the first section.

Using elements correctly

For XML to be valid it must not only be well-formed (which means that all the tags match), but also must have all the tags in the proper hierarchy according to the associated DTD (in our case, the DocBook 4.4 DTD). The tag at the top of the hierarchy is called the root element. For a book, `<book>` would be your root element, which would contain `<part>` or `<chapter>` children, for example. Tags like `<chapter>` must be nested within `<book>`, and a `<sect3>` cannot be directly nested within a `<sect1>`—it would have to be within a `<sect2>`. Reversed tags or improper nesting will return invalid DocBook files.

We'd also prefer that you don't put any block elements within `<para>`s. Some of the block elements that we'd like to avoid in `<para>`s are the following:

```
<blockquote>
<calloutlist>
<caution>
<computeroutput>
<example>
<figure>
<glosslist>
<important>
<informalequation>
<informalexample>
<informalfigure>
<informaltable>
<itemizedlist>
<literallayout>
<mediaobject>
<note>
<orderedlist>
<programlisting>
<screen>
<screenshot>
```

```
<segmentedlist>
<simplelist>
<table>
<tip>
<variablelist>
<warning>
```

XXE will show you what element you are using in the element bar at the top of your screen. You can apply block elements to a large chunk of text (`<programlisting>`). You can also apply inline elements that apply to just a word or phrase within a block element (`<emphasis>`). To make a word or phrase emphasized (typically *italic*), highlight the text and click the Convert button. In the box below the Convert button, start typing the inline element you wish to apply, in this case `<emphasis>`. The most common elements that you will need are included in an XXE toolbar, so you won't need to go searching for them.

The Element Bar's default description of the text in paragraph is `#text`. However, you can select the `#text` element and Convert to something that better suits your needs, like `literal`.

If you tag something and realize that you need to retag it as something else, click on the element in your Element Bar to box the word or phrase. Then click the Convert button and choose the correct tag from below. The convert button is shown in **Figure 14**. You can also use the formatting buttons in the toolbar to convert text to *emphasis*, links, or plain text. This is often faster that using the Convert button.

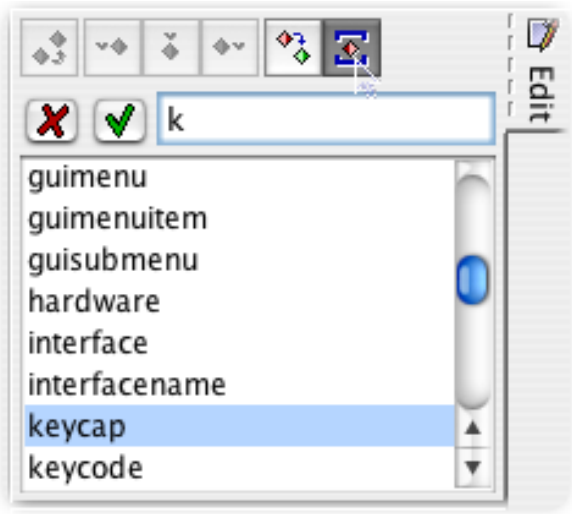

Figure 14. Using the Convert button

If you don't see the element you want as an option for your highlighted text, it probably means it isn't a valid option. Try placing your cursor at a different level in the hierarchy, or moving the element you want to change up or down. Only the elements that are valid for your location in the hierarchy are displayed as options.

Similarly, if you've cut text from one spot and are having trouble pasting it, make sure that you are pasting into the same element. Try moving up or down the hierarchy. You can also select the Paste, Paste Before..., and Paste After...buttons from the tool bar.

> **Note**
>
> Some markup isn't automatically available as an inline element. There is no default tag for making text bold in DocBook. To work around this, you would tag the word as `<emphasis>` and then give it an attribute—in this case, you would assign an attribute `role="bold"`.

To move a piece of text up or down, chose the element from the Element Bar and then go to DocBook→Move Up or Docbook→Move Down. To move an element up or down in the hierarchy, chose the element in the Element Bar and select DocBook→Promote or DocBook→Demote.

If you are including URLs in your document, it's best to designate it as a `ulink`. This will allow a PDF or online version of your document to have active links to the URL and it will provide more consistent display of URLs in the printed text.

Make sure if you tag something as a `ulink` that you enter the `url` in the attribute menu. Select the ??? and enter the URL where you want the link to direct readers. If you don't enter the address as an URL attribute, you will get a red message that says "Empty <ulink>! Needs url>>".†

For a bulleted list, you can choose Add itemizedlist from the toolbar to insert the first bullet point. You can also add numbered lists (Add orderedlist) and variable, or term-definition, lists (Add variablelist).

Tables

There are two options for tables: formal and informal. A formal table will have a table number and caption, making it better suited for cross-referencing. An informal table lacks the table number and caption and is more suited for a little table interspersed with normal text that won't need to be referenced later. The easiest way to add a table to a DocBook document is by choosing the Add Table button

† ...assuming you have the ORM customizations installed (see **Customizing XXE**).

in the toolbar (shown in **Figure 15**). The most common type of table you will probably use is `table(head_row)`—that is, a table with the row at the top serving as the header.

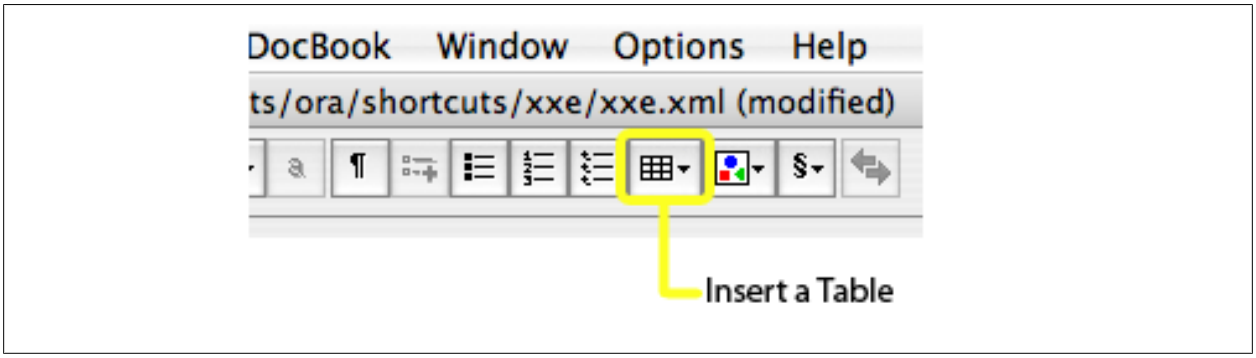

Figure 15. Inserting a table

If you need to add rows or columns to your table (the default is three rows by two columns), go to DocBook→Column→Insert Before or After or DocBook→Row→Insert Before or After.

For a formal table, make sure you select the `table` element and apply an `id` attribute, as illustrated in **Figure 16**. This is a unique identifier that will be used later when you add cross-references to it.

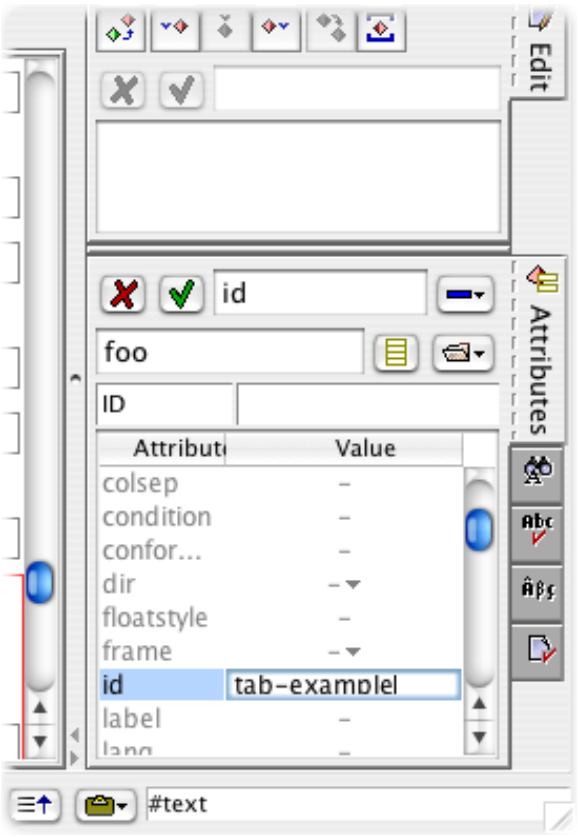

Figure 16. Applying an id attribute to a table

> **Warning**
>
> Make sure that you press Enter/Return or the green check mark after putting in an attribute or else it will disappear when you move on.

Figures

Figures are similar to tables in that you can have formal or informal figures. For formal figures, make sure to include a caption. There is no need to number the figure since the O'Reilly stylesheet will take care of that later. O'Reilly books rarely have informal figures.

> **Note**
>
> For instructions and guidelines for producing the figures for your book, please see the "**O'Reilly Media Illustration Guidelines** [***https://prod.oreilly.com/external/illustrations/illustrations_guidelines.html***]." A **PDF is also available** [***https://prod.oreilly.com/external/illustrations/illustrations_guidelines.pdf***].

Please place the image files for your book in the *figs/incoming* directory, which is located in the same directory as your *book.xml* file. This will be required down the road by the PDF processing software.

Then, to add an image, you can simply click the Add Image button in the toolbar and choose which type of image you are inserting. This is illustrated in **Figure 15**. For standard figures, choose "figure" from the drop-down menu.

Screenshots insert a spot for the image plus a blue line called `screeninfo`. We do not render the `<screeninfo>` element, so please do not add any text to it that you want diplayed in the book.

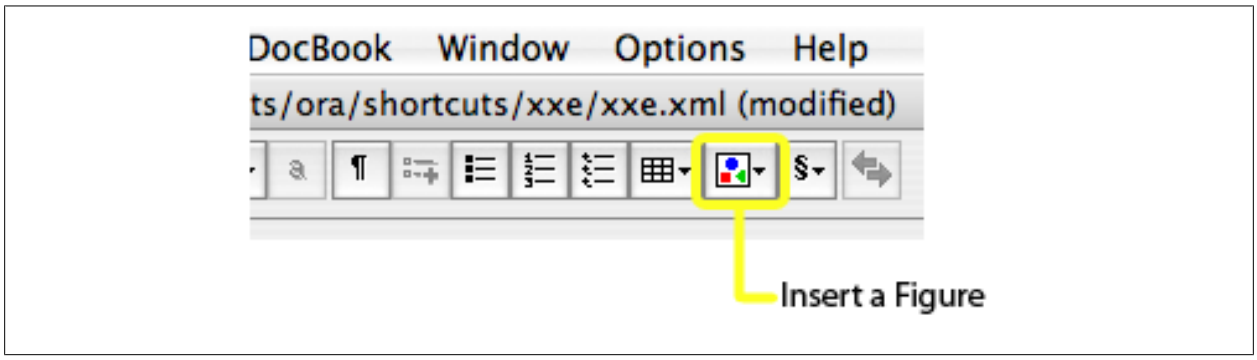

Figure 17. Inserting a figure

When the space for the image is selected, you will notice that in the attribute menu, `fileref` has ??? entered. By clicking the folder at the top of the attribute menu, you can browse for the file location of your image (see **Figure 18**). You can also manually enter this file path next to `fileref` in the menu below or double-click on the image placeholder to bring up a `fileref` window, as shown in **Figure 19**.

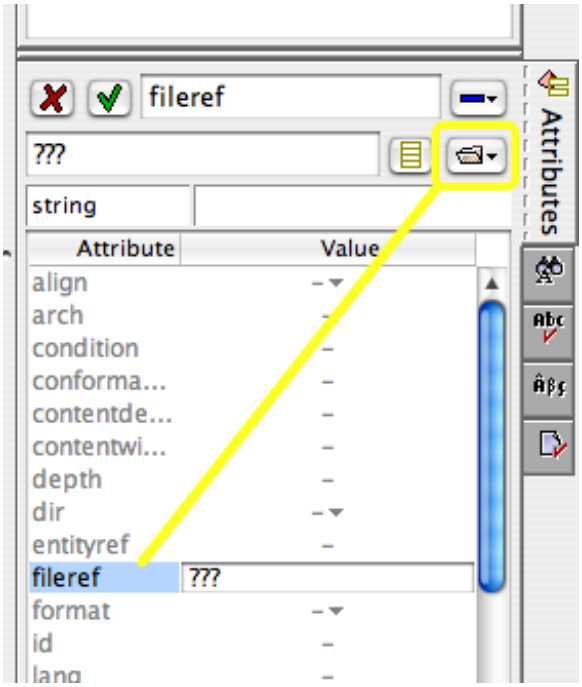

Figure 18. Sourcing the figure file

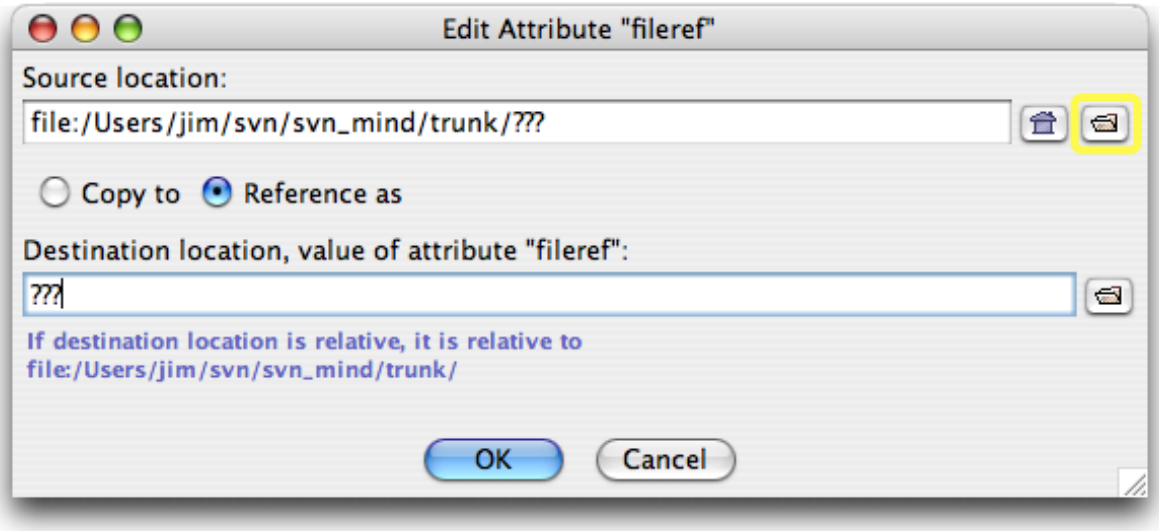

Figure 19. Picking a file source location

You'll also want to make sure you enter an `id` so that the figure can be cross-referenced later.

> **Note**
> XXE is able to render PNGs but not PDFs—just like web pages.

Scaling images

When your book goes into Production, Illustration will handle processing the images you submit, including scaling them to the appropriate size. However, if you'd like to scale your images during the manuscript phase for the purposes of generating draft PDF documents, you can do so using the `width` attribute of the `imagedata` element, which scales the image proportionally to the width value supplied. For example, to set a width of 4.8 inches (maximum width for Animal Guide books; see our **Illustration Guidelines** [***https://prod.oreilly.com/external/illustrations/illustrations_guidelines.pdf***]) for an image, you'd add the `width` attribute shown in **Figure 20**.

Table 1 contains a list of maximum widths you can use to scale images to fit your book's template.

Table 1. Maximum widths for different book templates

Book Series	Maximum width (in inches)
Animal Guide	4.8in
Nutshell (and other books with 6x9 inches trim size)	4.3in
Pocket Reference	2.8in

Establishing cross-references

While writing your document, you might want to put placeholder text (like `()` for example) where the cross-reference will later be inserted. When you are ready to insert the actual cross-reference, place your cursor inside the parens, where you want the `xref` to go.

Click the Insert button and pick the `xref` element from the list of available elements in the Edit tool. Then in the attributes menu, next to `linkend` you will see ???. Click on the question marks and then click on the attribute list at the top of the attribute menu. This will give you a list of `id`s you can cross-reference. **Figure 21** and **Figure 22** illustrate this.

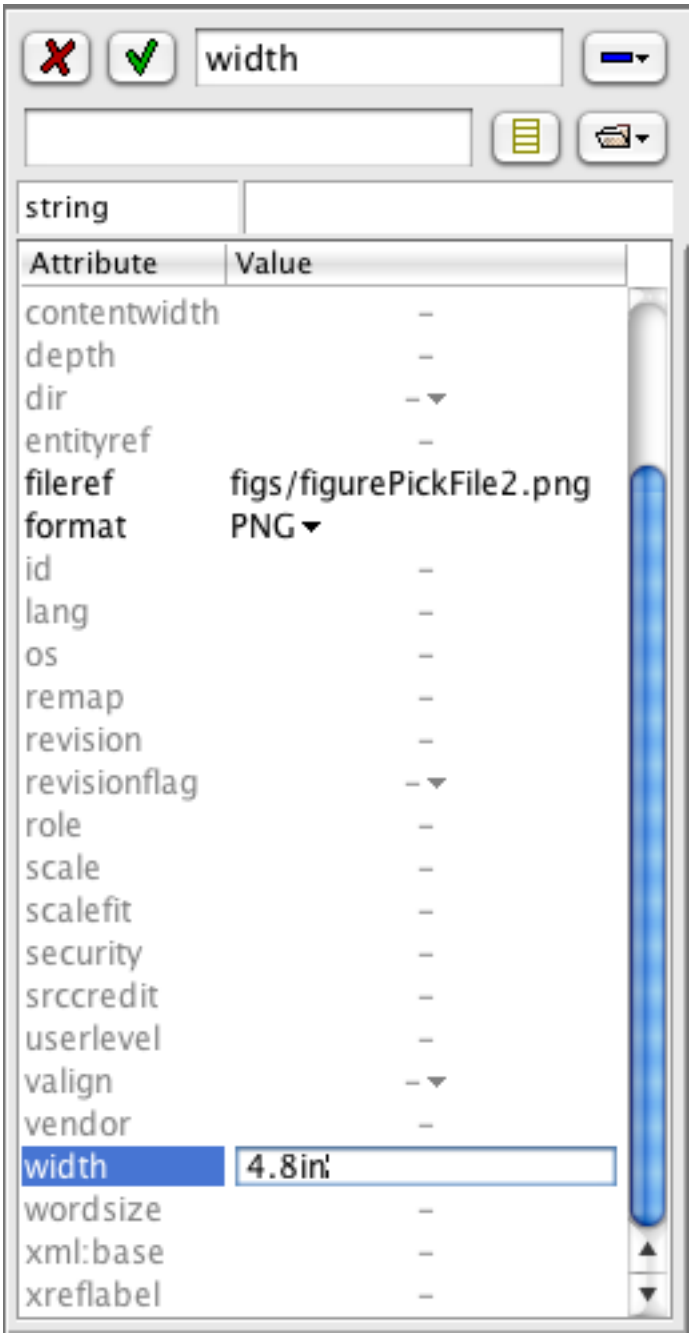

Figure 20. Adding a width attribute

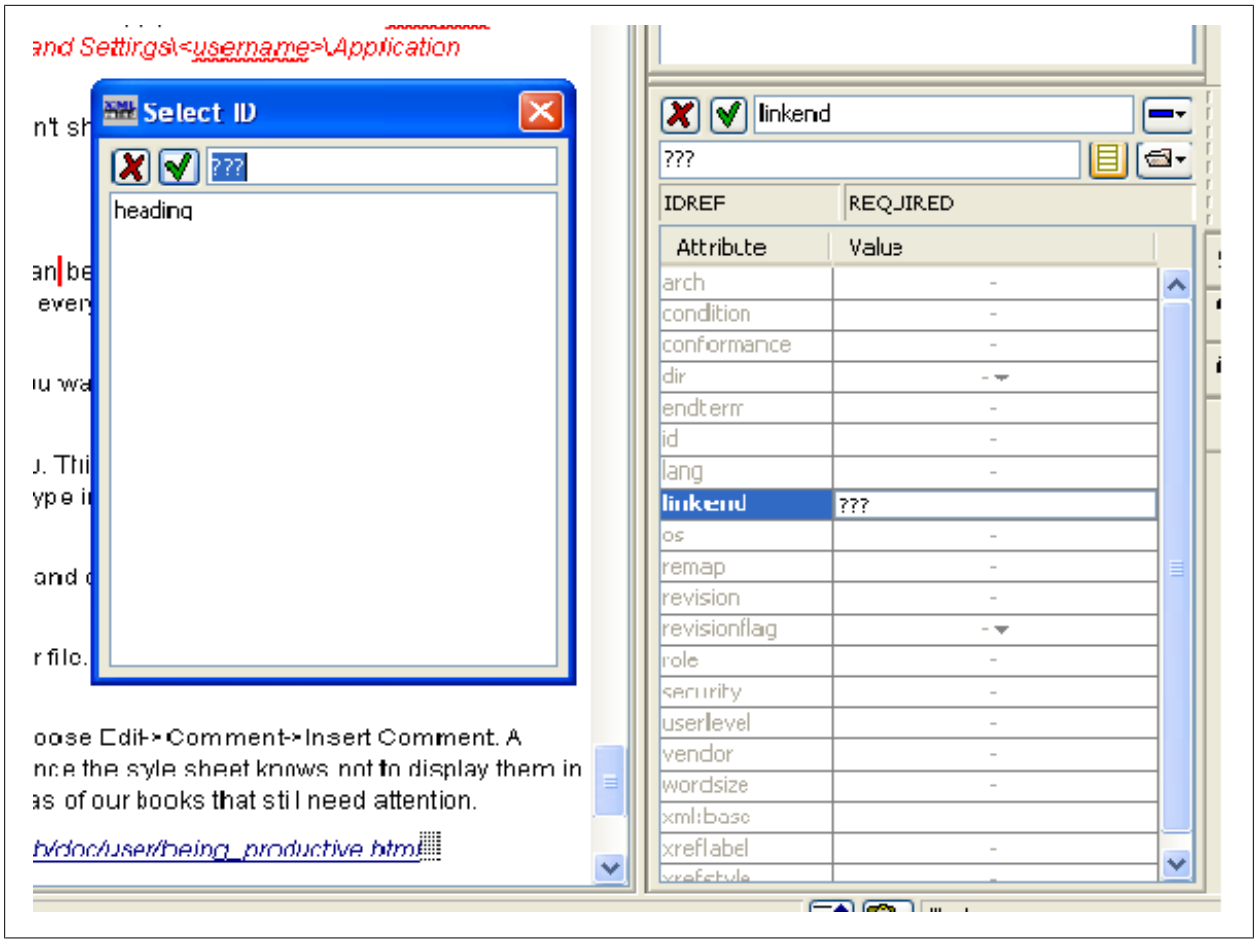

Figure 21. linkend

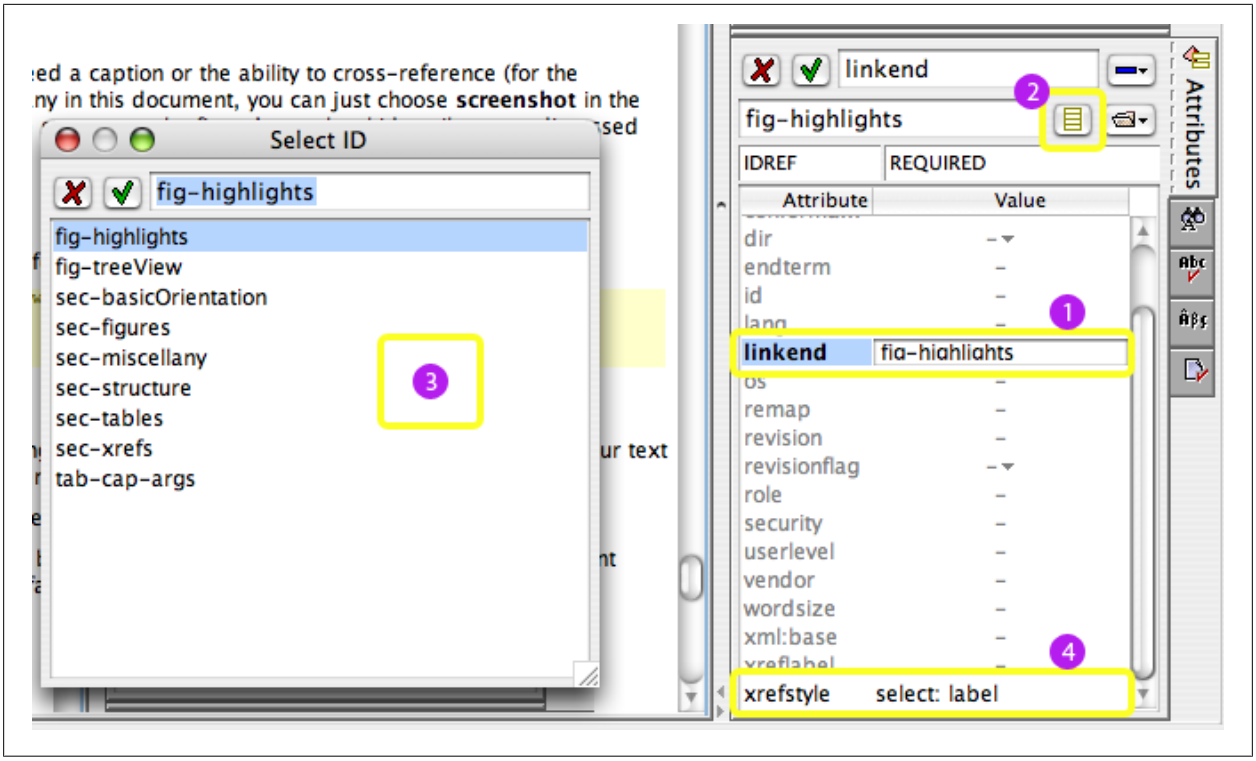

Figure 22. Select an xrefstyle

Note

If you need to reference a web page, select the text that you want to make a hyperlink and then convert it by using the Convert to link button in the toolbar. The location of this button is shown in **Figure 23**. Then in the attribute menu, enter the `url` where the `ulink` should take readers (see **Figure 24**). This will produce a live link in the PDF or HTML versions of the document.

Figure 23. Convert to ulink

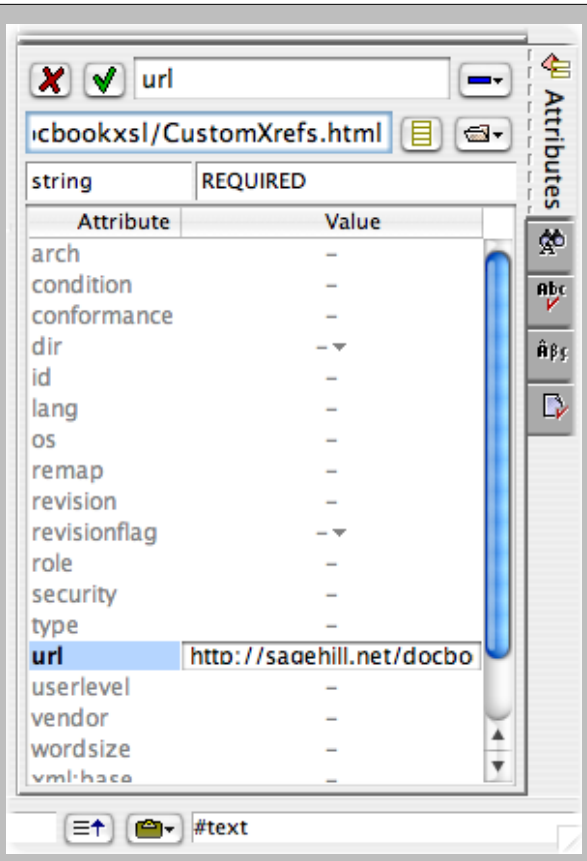

Figure 24. Entering the URL

For O'Reilly books, make sure that all formal figures, tables, and examples are explicitly cross-referenced in the text.

Customizing XXE

You can run some customizations that tweak XXE to display your DocBook documents more closely to their PDF (and eventually printed) form, you can find a CSS file customized to O'Reilly fonts, headings, and item numbering at ***https://prod.oreilly.com/external/tools/xxe/customizations/xmlmind_custom.zip*** (username: `guest`; empty password).

Unzip the *xmlmind_custom.zip* file to *~/.xxe/addon/* if on Linux or Mac (and Unix, more generally). If on Windows XP, unzip to *C:\Documents and Settings\<user name>\Application Data\XMLmind\XMLEditor\addon*. More detailed instructions for installing customizations are found on the XXE web site: ***http://www.xmlmind.com/xmleditor/addons.shtml#manual_install***.