
DocBook Tools: An Overview

by: Scott Nesbitt

This report is copyright ©2006 Scott Nesbitt. All rights reserved.

All commercial software mentioned in this report is copyrighted by, or is a trademark of, its developer.

For more information about my services or to purchase other documents visit <http://www.scottnesbitt.net> or email me at scott@scottnesbitt.net.

Introduction

While DocBook is a powerful language for authoring and publishing documentation, one complaint that is frequently levelled against DocBook is that it is difficult to get the various DocBook tools (collectively known as the *DocBook toolchain*) configured and working together. This is especially true for the toolchain used with DocBook XML. In fact, there are a number of very competent people who have never been able to get the DocBook SGML toolchain to work properly.

However, the DocBook XML toolchain is relatively easy to install and configure. The biggest problem that you will face is choosing the elements of the toolchain to use. The DocBook XML toolchain consists of the following elements:

- DocBook stylesheets
- Editors
- XSLT processors
- FO processors

As you can expect, there are a number of different applications to choose from. This document looks at some of the choices available on the Web, and also contains links to several all-in-one tools and to a number of useful DocBook-related Web sites.

DocBook Stylesheets

You use a set of XSL stylesheets to transform DocBook XML documents to formats that your end users can view. The standard stylesheet distribution enable you to convert files authored in DocBook XML to the following formats:

- HTML, both a single file and individual files broken up at the section level that include navigation between the files
- XHTML, both a single file and individual files broken up at the section level that include navigation between the files
- Microsoft HTML Help – the stylesheets only output the source HTML files and the project files. You must compile the .chm file using a third-party tool
- JavaHelp
- Help for Eclipse
- UNIX man pages

- XSL-FO, which can then be transformed to Postscript, PDF, or RTF

There are also DocBook stylesheets for transforming XML files into slides and Web sites. The stylesheets can also be found at the DocBook Open Repository site.

On top of that, there are third-party stylesheets that enable you to convert DocBook files into other useful formats. One of these is the OperaShow customization layer. This stylesheet transforms a presentation authored with the DocBook slides stylesheets into one that can be viewed using the OperaShow feature of the Opera Web browser.

Technical writers will find David Cramer's `applehelp` stylesheet to be incredibly useful. This stylesheet allows you to create a WebHelp system (a method for delivering online help or documentation in a Web browser) without the need for the expensive RoboHelp authoring tool. For more information on using the `applehelp` stylesheet, see this article.

One other useful stylesheet distribution is DocBook to \LaTeX (`db2\text{\LaTeX}`). `db2\text{\LaTeX}` converts XML source files to a format that can be processed by the \LaTeX document processor and its associated tools.

A Few Words about Parameters and Customization Layers

As you start using DocBook XML and reading various articles, manuals, and how-tos, you will encounter the terms parameter and customization layer. These are useful aspects of DocBook to understand, and the next few paragraphs outline what parameters and customization layers are and why they are useful.

Parameters

Parameters let you control aspects of the documents that you generate from a DocBook source file. There are literally hundreds of DocBook parameters that control everything from the colour and borders of tables to the alignment of text and headings to the location of graphics. The DocBook stylesheet distribution comes with fairly complete documentation for the parameters. Check the `docs/html` and `docs/fo` subdirectories where you installed the stylesheets.

Parameters are a great and simple way to change the look and feel of your documents. You can use parameters with many XSLT processors or in a customization layer.

Parameters consist of a name and a value. The name, obviously, is the name of the parameter. The value determines whether or not the parameter is used. For example, in a customization layer the following parameter ensures that the document is typeset to print on both sides of a page:

```
<xsl:param name="double.sided" select="1"/>
```

How you use parameters with an XSLT process depends on the processor that you are using. See the section XSLT Processors for more information.

Customization Layers

It's generally not a good idea to edit the default DocBook stylesheets in order to customize the look of your output. Why? First, the DocBook stylesheets are regularly updated. Switching over to the new stylesheets means that you have to duplicate your changes – a time-consuming process. Second, there will be times when you want certain documents to have a slightly different look or slightly different features from your other documents. For example, you might not want to include a table of contents or index with a technical article, or you might want an HTML-based online help file to look a bit different from a straight HTML document. And that's where a customization layer comes in.

A customization layer (sometimes called a *driver file*) is simply an XSL stylesheet that overrides certain defaults in one of the main DocBook stylesheets. But it does not work independently of the stylesheet that it is overriding; you still must import the stylesheet.

The customization layer contains parameters and attributes, and the values that you want to use. If, for example, you are creating a document that will be printed you can use a customization layer to change the colour of headings, specify a font family to use with the headings, change the default text alignment, etc.

Using a customization layer, you keep all of your modifications in one file. When you do upgrade your DocBook stylesheets, all you have to do is change the path to the main stylesheet in the customization layer and your modifications still work.

The example below is a very simple customization layer for outputting HTML documents.

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
  <xsl:import href="docbook-xsl-1.62.4/html/docbook.xsl"/>
  <xsl:param name="html.stylesheet" select="'docbook.css'"/>
  <xsl:param name="shade.verbatim" select="1"/>
  <xsl:param name="generate.toc" select="0"/>
</xsl:stylesheet>
```

Editors

Since DocBook files are XML, you can edit them in any text editor. A number of people use editors like Vim, JEdit or Nedit to craft DocBook files. However, many of

these editors aren't always the best choices. Most of them are generic editors, designed for viewing/modifying system files or programming. They generally don't have much – if any – support for XML and specifically DocBook authoring.

However, there are several text and XML editors that are powerful tools for authoring in DocBook. Not only do they have extensive XML editing features, they are also highly configurable and, in some cases, run on multiple operating systems. Many of them also share features, like built-in XML parsers that check the validity of your documents.

Emacs and XEmacs

Emacs and its graphical counterpart XEmacs are very popular text editors. In fact, XEmacs is my editor of choice when authoring documents using DocBook and \LaTeX . As with any other text editor, you can use Emacs to edit DocBook source files. However, Emacs has an interesting extension called psgmlx which turns Emacs into an XML editor. psgmlx will validate your XML files and enables you to insert tags by using a keystroke combination or by right-clicking in XEmacs.

But what is especially useful about using Emacs or XEmacs is that you can call an application to process a DocBook file from within the editor. You simply type `Esc-!`, followed by the name of the tool or script and the name of the file to process. Press Enter and, if all goes well, instant conversion.

XXE

XXE is short for XMLMind XML Editor. Unlike the other editors discussed in this section, XXE is a WYSIWYG tool, more or less. What appears in the interface is a good approximation of how your rendered document will look.

While not exclusively a DocBook editor, XXE has been developed with extensive support for DocBook. In fact, the editor has a menu called DocBook which, as you can guess, contains DocBook-specific commands. You can even convert DocBook files to HTML, RTF, Postscript and PDF.

There are two versions of XXE: a Standard edition, which is free; and a Professional edition, which costs \$220 for a single license. The biggest differences between the two editions of XXE are that the Professional edition supports a wider number of XML schemas, and you can convert documents to RTF, Postscript, and PDF from within the editor. With the Standard edition, you can only convert DocBook files to HTML.

Morphon XML-Editor

Once a commercial product, Morphon was released as free software in 2003. When using Morphon as an authoring tool, you can right click anywhere to insert tags. You can also change views – you can show the XML tags in your document in several ways, or work without tags. The last option is not a WYSIWYG view, although you can use a Cascading Style Sheet file to format the text in the browser.

You can also download a plugin that enables you to output a PDF file directly from within the editor. The plugin is highly configurable – you can specify the stylesheet and XSL-FO processor that you want to use – and is useful for producing interim drafts or even the final version of your document.

On the downside, Morphon can be quite slow. On a new computer with 512 MB or more of memory, this is not too much of a problem. On older hardware, you may have to use another editor.

epcEdit

epcEdit is a multi-platform editor that can handle both XML and SGML. It is a commercial product, and it packs a number of features that make it useful for generic XML and SGML editing, as well as editing DocBook documents. It has a number of useful features, which are listed at the editor's Web site.

The biggest drawback of epcEdit is its price: 89 euros for personal or academic use, and 299 euros for commercial use.

oXygen XML Editor

Another commercial editor, oXygen/ is one of the more fully-featured DocBook editing applications available. Not only does it come with the DocBook stylesheets and DTDs, the FOP processor is also bundled with it. This enables you to output Postscript and PDF files right from the editor. oXygen also allows you to configure other processors as plug-ins.

At \$74, oXygen isn't overly expensive. But there are a number of cheaper or free alternatives available.

Vex

Vex is a rather interesting and unique XML editor. It's based on Eclipse, a large and powerful programming environment that is designed to support just about any development task. Vex is just a modified version of the editing component of Eclipse.

When you edit a DocBook file with Vex, you're not directly editing the code. Instead, you're working in a pseudo-WYSIWYG view (along the lines of XXE). Vex uses a Cascading Stylesheet to format an XML file within the editor. Note that the way a document looks in Vex won't be the same as it looks when you compile it.

While Vex is an all-purpose XML editor, it has a number of DocBook-specific features. The ones that you will use most often are:

- A right-click menu that allows you to apply DocBook markup to paragraphs or elements in a paragraph.
- An outline that enables you to quickly navigate to the various sections and subsections within a document.
- The ability to add external tools to Vex. You can run the tools that transform DocBook to more usable formats from within Vex.

On top of that, Vex forces you to work with *projects*, not individual documents. In Vex, a project is a container for the DocBook source files that make up your document as well as any related files, such as graphics. This feature makes Vex great for authoring and maintaining a large document that is made up of several individual chapters. However, it can be a bit cumbersome when you are dealing with a single file.

Quanta

Quanta is a popular text-based Linux HTML editor. But Quanta is also a great tool for editing XML documents, including DocBook files. In fact, the developers of Quanta have added some DocBook-specific features to the editor, including:

- Syntax highlighting
- A DocBook toolbar
- A tag editor
- Auto completion of DocBook entities

For more information, including screenshots, visit the [Quanta development site](#).

TextPad

TextPad is a popular shareware text editor for Windows. Thanks to several of its features and its overall extensibility, TextPad has become an especially popular tool for

Web developers and programmers. With some simple configuration, TextPad can become a serviceable DocBook XML editing environment.

For information on how to use TextPad as a DocBook XML editor, see this article.

AbiWord and OpenOffice.org

The AbiWord word processor and the word processor component of the OpenOffice.org office suite can save files as DocBook XML. However, the DocBook support of both applications is at best fair. In fact, both only seem to support a subset of the DocBook tags. Using either application, conversions seem to work moderately well with smaller documents. But with larger files many things can go wrong. For example, section tags are stripped out, bullet and number lists lose their formatting, and non-DocBook markup is thrown into the mix.

Until all the problems are worked out, it is a good idea to create and maintain your DocBook documents in a different editor.

XSLT Processors

XSLT processors work in conjunction with DocBook (and other XML) stylesheets to transform your DocBook source files into other formats. These formats include XSL-FO, HTML, and other variants of XML. In the case of DocBook, you can also use an XSLT processor to convert you documents to HTML Help, JavaHelp, slides, a Web site, L^AT_EX, or any other format for which stylesheets exist.

There are many XSLT processors available. I tend to use ones that are written in Java, mainly because they are more portable and are consistent across various operating systems. As well, all of the XSLT processors that I mention here are command line tools. While you can type the string of commands, options, and parameters each time that you want to transform a DocBook file, this gets tedious. It's more convenient to encapsulate the commands, options, and parameters into shell scripts or batch files.

Saxon

Saxon is a well-known and mature XSLT processor. Saxon is fast and very robust, and I've found that it is one of the easiest XSLT processors to use. In fact, Saxon was the first application of this type that I ever used and even now is my main XSLT processor on both Linux and Windows.

To convert a DocBook file to XHTML, you would type something like the following at the command line:

```
java net.sf.saxon.Transform
/<path>/document.xml
</<path>/xhtml/docbook.xsl
```

Where path is the directory path to your document and to the appropriate DocBook stylesheet, respectively.

You can also specify DocBook parameters on the command line. You place after the declaration of the XSL stylesheet. For example, you can add the following parameters to apply a Cascading Style Sheet to an XHTML file that you create from a DocBook document:

```
html.stylesheet.type=text/css
html.stylesheet=webhelp.css
```

Xalan

Xalan is the XSLT processor for the Apache XML Project. Like Saxon, it's a mature and well-regarded XSLT processor that can handle most DocBook transformation tasks quickly and efficiently.

To convert a DocBook file to an XSL-FO file, you would type something like the following at the command line:

```
java org.apache.xalan.xslt.Process -in
/<path>/document.xml -xsl
/<path>/fo/docbook.xsl
```

Where path is the directory path to your document and to the appropriate DocBook stylesheet, respectively.

You can also specify DocBook parameters on the command line. You type the parameter names and values after the declaration of the XSL stylesheet. The parameter names and values must be preceded by the -param option. For example, you can add the following parameter to suppress the generation of a table of contents:

```
-param generate.toc 0
```

xsltproc

xsltproc is the XML library for the Gnome desktop environment. Unlike many other XSLT processors xsltproc is written in C and not Java. It is a powerful tool, but it is

also a bit cumbersome to get running. You need to compile, install and configure not only the xsltproc executable, but also a set of XML libraries. Note that you can get pre-compiled binaries and libraries for Windows.

To convert a DocBook file to an XSL-FO file, you would type something like the following at the command line:

```
xsltproc -o myFile.fo
/<path>/fo/docbook.xsl myFile.xml
```

Where path is the directory path to the appropriate DocBook stylesheet and to your document, respectively.

You can also specify DocBook parameters on the command line. Unlike Saxon and Xalan, when using xsltproc you type the parameter names and values immediately after the name of the executable. The parameter names and values must be preceded by the `--stringparam` option. For example, you can add the following parameter to generate an index at the end of your document:

```
--stringparam generate.index 1
```

XSL-FO Processors

XSL-FO is short for XSL Formatting Objects, which is a form of XML that enables you to convert your DocBook (and other XML) documents to a format that can be viewed and printed. The conversion is done using a tool called an FO processor. Depending on the FO processor that you are using, you can convert an XSL-FO file to the following formats:

- PDF
- Postscript
- PCL (a format native to Hewlett-Packard printers)
- SVG (Scalable Vector Graphics, an XML-based graphics format)
- AWT (Abstract Window Toolkit, which provides graphical user interfaces for Java programs)
- MIF (for import into FrameMaker)
- Plain text

There are a number of XSL-FO processors available; you can find a fairly complete list [here](#).

FOP

FOP is part of the Apache XML Project and was one of the first XSL-FO processors available. FOP is a free Java application that works on the command line. You feed FOP the name of the file that you want to convert, an option to specify the type of output, and the name of the output file. For example, on Linux/UNIX you would type something like the following to convert an XSL-FO file to PDF:

```
fop.sh myFile.fo -pdf myFile.pdf
```

While a powerful and useful tool, FOP still does not implement all of the XSL-FO standard. You can view a table of FOP's compliance with the standard here. Therefore, FOP may not be the best tool for producing high-quality output.

XEP

XEP from RenderX is a commercial product that converts XSL-FO files to Postscript, PDF, or HTML. One interesting feature of XEP it can transform an XML file to Postscript, PDF, or HTML without using an XSLT processor.

To convert an FO file to PDF using XEP on Linux/UNIX, you would type something like the following:

```
java com.renderx.xep.XSLDriver -fo  
myFile.fo -pdf myFile.pdf
```

The XEP distribution also comes with a shell script and a batch file that encapsulate the calls to the XEP processor.

XEP support the XSL-FO standard far more fully than FOP. it is also highly configurable. However, a license for XEP costs \$299 and up. If you can afford it, XEP is a very useful tool. If you can't afford or justify the price, you might be safer with FOP.

Graphical Tools

While the command line is an incredibly powerful and flexible interface (read this article if you don't believe me), it can also be a pain to use. Typing long strings of commands and options, or remembering the names and syntax of scripts can sometimes be painful. Sometimes you just want to point and click. Luckily, there are at least two

When you want to convert a DocBook file to a printable output format like Postscript or PDF, you do it in two steps: use an XSLT processor to convert the file to XSL-FO,

and then use an FO processor to generate the output. But the graphical tools discussed in this section can go directly from XML to Postscript or PDF.

XFC

XFC is short for XMLmind FO Converter. Written in Java, XFC is quite simple to use. You just specify the document you want to transform, select the output format you want, select where you want the file to go, and then click the Convert button in XFC. If all goes well (and it should, unless your DocBook file has errors), you will have a usable output.

XFC has two features that set it apart from other converters. The first is that it is highly configurable. You can define literally dozens of transformations – I currently have about 40 defined. You can do this because XFC supports DocBook parameters. Therefore, you can have one transformation that generates a PDF with a table of contents and index, another without, etc. The other feature is that XFC can output RTF files for later import into a word processor or desktop publishing application. I cannot think of any other tool that does this.

For more information on using XFC, see [this article](#).

DocMan

DocMan is short for the DocBook Toolchain Manager. Everything is available with a click. However, DocMan is not as flexible as XFC. It currently only supports the following six transformations: HTML, HTML split into multiple files, XHTML, XHTML broken into multiple files, PDF, and HTML Help.

Some of DocMan's features aren't implemented yet, but it is a useful tool for quickly getting up and running with DocBook and not having to worry about installing stylesheets and the individual components of the DocBook XML toolchain.

Useful DocBook Web Sites

Here are links to some useful Web sites containing information about DocBook.

- [DocBook.org](#), which is the central repository of all DocBook knowledge on the Web.
- [DocBook Wiki](#), where you can find useful information, links, and more.
- [Software Documentation Weblog](#), a great blog that has a number of DocBook-related postings.

- [Writing Documentation Using DocBook](#), an older though still useful tutorial on authoring with DocBook.
- [DocBookmarks](#), a set of useful DocBook-related links.
- [DocBook Frequently Asked Questions](#), which is a comprehensive FAQ about authoring with DocBook and using the DocBook stylesheets.
- [How to Avoid Learning XML](#), a presentation on how to use DocBook to meet all of your documentation needs.
- [Writing Technical Documentation with DocBook](#). A slide presentation on using DocBook for creating documentation.
- [DocBook XML Resource at CERN](#). This is a document on how the CERN particle physics laboratory uses DocBook.
- The online edition of [DocBook XSL: The Complete Guide](#), which is the best reference available for the DocBook XSL stylesheets.
- [Antenna House XSL Formatter](#), a powerful but very expensive piece of software for transforming XML files to formats that humans can read.
- [Getting Started With DocBook on OpenOffice](#) is a guide to editing DocBook files using the OpenOffice.org suite.