DocBook

DocBook

http://en.wikibooks.org/wiki/XML_-_Managing_Data_Exchange/DocBook
This Book Is Generated By Wb2PDF
using
RenderX XEP, XML to PDF XSL-FO Formatter

29-04-2011

Table of Contents

DocBook

initially written by Rusen Gul, University of Georgia, 2004

Learning objectives

Upon completion of this chapter, you will be able to

- Learn the basics of DocBook
- Create a DocBook document by using the DocBook XML DTD
- Convert a text document to a DocBook document
- Use XSL Stylesheets to transform a DocBook XML document to multiple formats as HTML, PDF or presentation slides.

Introduction

DocBook is general purpose XML and SGML vocabulary particularly well suited to books, articles, and papers. It has a large, powerful and easy to understand Document Type Definition (DTD), and its main structures correspond to the general concept of what constitutes a book. DocBook is a substantial subject that we can't exactly cover in a few pages. Thus, for the purposes of this chapter, we will talk about creating a simple DocBook document with major elements in the Docbook DTD and the details of publishing the document in order to give you a feel about DocBook. If you would like to study the subject further, we suggest you to have a look at the references provided at the end of the chapter.

What is DocBook?

- DocBook enables you to author and store document content in a presentation-neutral form that captures the logical structure of the content.
- It has an easy-to-understand and widely used DTD. The DocBook tags are applied so that they have a certain "common sense" semantic content, at least to English speakers.
- There are no official versions of the DocBook W3C XML Schema at this time. The DocBook Technical Committee is planning to offer an official Schema in the DocBook V5.0 time frame. The examples provided in this chapter will use the current official DTD.

DTD vs. Schema:

A DTD is the XML Document Type Definition contains or points to markup declarations that provide a grammar for a class of documents. A Schema is a set of shared vocabularies that allow machines to carry out rules made by people. It provides a means for defining the structure, content and semantics of XML documents. In summary, schemas are a richer and more powerful means of describing information than DTDs.

Table 1: Here is a simple xml document

```
<author>
  <firstname>Rusen</firstname>
  <lastname>Gul</lastname>
</author>
```

Table 2: Here is the DTD for this document

```
<!ELEMENT author(firstname, lastname)>
<!ELEMENT firstname(#PCDATA)>
<!ELEMENT lastname(#PCDATA)>
```

Table 3: And here is the SCHEMA

```
<xs:element name="author">
  <xs:complexType>
    <xs:sequence>
```

```
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Output formats for DocBook

XSL (Extensible Style Language) stylesheets can transform DocBook XML into the following formats:

- HTML
- HTML Help (for Windows Help)
- Java Help
- XHTML
- XSL Formatting Objects (FO)
- PDF

DSSSL (Document Style Semantics and Specification Language) stylesheets can transform DocBook SGML into the following formats:

- HTML
- MIF
- RTF
- TeX

A Brief History

DocBook was created around 1991 by HaL Computer Systems and O'Reilly & Associates. It was developed primarily for the purpose of holding the results of troff conversion of UNIX documentation, so that the files could be interchanged. Now it is maintained by OASIS. The official web site for DocBook is http://www.oasis-open.org/docbook/

DocBook is used for:

1. Books for print/trade publication. Many authors are using DocBook for writing books of all kinds, in various print and online formats, worldwide. Some examples include:
   - Processing XML with Java
   - Learning XML
   - DocBook: The Definitive Guide
   - Installing and Using DocBook
   - Dive into Python
2. Articles, theses and dissertations
3. Maintaining websites
4. Producing presentation slides, printed handouts, and whitepapers
5. Documentation for commercial software and hardware

DocBook Tools

DocBook is officially available as a DTD for both XML and SGML. You can download both the latest DocBook XML DTD and DocBook SGML DTD from the official DocBook site at OASIS. The examples provided in this chapter will use DocBook XML DTD. Some experimental DocBook schemas are available at sourceforge.net. DocBook is supported by a number of commercial and open source tools. Easily customizable and extensible "standard" DocBookStylesheets are available from the DocBookOpenRepository along with the other free open source tools. See DocBookTools on the DocBook Wiki for a more complete list of commercial and open source tools.

Other Free Tools:

1. **XSLTProc:** One of the most known and fast processor. Available at http://xmlsoft.org/XSLT/.
2. **Apache FOP:** XSL-FO implementation. Available at http://xmlgraphics.apache.org/fop/.
3. **Xt:** One of original XSLT processors. Less frequently used now. Available at www.jclark.com.

4. **DocBook2x:** Converts DocBook to man and Texinfo pages. Available at Sourceforge.
5. **Refdb:** Creates reference databases and bibliographies from DocBook. Available at Sourceforge.

Commercial Tools:

1. **Arbortext Epic:** Comprehensive suite of both editing and processing tools. Available at Arbortext website.
2. **RenderX XEP:** FO to PDF rendering engine. Available at RenderX website.
3. **Antenna House XSL Formatter:** FO to PDF rendering engine. Available at Antenna Housewebsite.

SGML vs XML

The syntax of SGML and XML DTD is very similar but not identical. The biggest difference between the DocBook DTD for SGML and the one for XML is that the SGML DTD contains SGML exclusions in some content models.

Example: SGML DTD excludes <footnote> as a descendent of <footnote>, because it doesn't make much practical sense to have footnotes within footnotes. XML DTDs can't contain exclusions, so if you're authoring using the DocBook XML DTD, it's possible to produce documents containing some valid-but-not-logical markup like footnotes within footnotes.

Creating a DocBook Document

In order to get started, you will need:

- An XML editor. Download NetBeans IDE if you haven't done so yet.
- The DocBook XML DTD. Although it is optional to use one, DTD's are useful when one wants to validate a document to check that it conforms to the DTD to which one claims it conforms. Hence, the DocBook DTD can be used to validate that a purported DocBook document. DocBook XML 4.2 is the current version of DocBook DTD. Download at the official DocBook website

- The DocBook XSL stylesheets are maintained primarily by Norman Walsh. There are two sets of stylesheets: XSL and DSSSL. Download the latest version XSL 1.65.1 at Source-forge.net
- An XSLT processor (covered in the further sections)

Table 4: A simple DocBook Book, "book.xml"

```
<?xml version="1.0"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<book>
 <bookinfo>
  <title>XML - Managing Data Exchange</title>
  <author>
   <firstname>Rusen</firstname>
   <surname>Gul</surname>
  </author>
 </bookinfo>
 <chapter>
  <title>Introduction</title>
  <sect1>
   <title>First Section</title>
   <para>This is a paragraph.</para>
  </sect1>
  <sect1>...</sect1>
 </chapter>
 <chapter>...</chapter>
 <chapter>...</chapter>
 <chapter>...</chapter>
 <appendix>...</appendix>
 <appendix>...</appendix>
</book>
```

Table 5: A simple DocBook article, "article.xml"

```
<?xml version="1.0"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<article>
 <articleinfo>
  <title>A Simple Approach To DocBook</title>
  <author>
   <firstname>Rusen</firstname>
   <surname>Gul</surname>
  </author>
 </articleinfo>
```

```
 <para>This is the introductory paragraph of my article.</para>
 <sect1>
  <title>First Section</title>
  <para>This is a paragraph in the first section.</para>
 <sect2>
  <title>This is the title for section2.</title>
  <para>This is a paragraph in section2.</para>
 </sect2>
 <sect2>...</sect2>
 <sect2>...</sect2>
 </sect1>
 <sect1>This is a high level section</sect1>
 <sect1>...</sect1>
 <sect1>...</sect1>
</article>
```

Let's examine the details of a typical DocBook document. Standard header to a DocBook XML file is a **DocType declaration**:

Standard header

```
<!DOCTYPE name FORMALID "Owner//Keyword Description//Language">
```

This tells the XML manipulation tools the DTD in use. Name is the name of the root element in the document. **FORMALID** is replaced with either **PUBLIC** or **SYSTEM** identifier or both. **PUBLIC** identifies the DTD to which the document conforms. **SYSTEM** explicitly states the location of the DTD used in the document by means of a **URI** (Uniform Resource Indicator). **PUBLIC** identifiers are optional in XML documents although **SYSTEM** Identifiers are mandatory in the DOCTYPE declaration.

Header example

```
<?xml version="1.0"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
```

Owner: Oasis
Keyword Description: DTD DocBook XML V4.2
Language: EN - English

Caution! If you are not online, you need to change the URL system identifier to path where DTD is installed:

```
<?xml version="1.0"?>
```

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"/usr/share/sgml/docbook/xml-dtd-4.2/docbookx.dtd">
```

Breaking a Document into Physical Portions

Before getting started, here is a useful tip! For the purposes of convenience and performance, you might consider breaking a document into physical chunks and work on each chunk separately. If you have a book that consists of three chapters and two appendixes, you might create a file called *book.xml*, which looks like this:

Table 6: A physically divided book, “dividedbook.xml”

```
<?xml version="1.0"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.2//EN"
[<!ENTITY chap1 SYSTEM "chap1.xml">
<!ENTITY chap2 SYSTEM "chap2.xml">
<!ENTITY chap3 SYSTEM "chap3.xml">
<!ENTITY appa SYSTEM "appa.xml">
<!ENTITY appb SYSTEM "appb.xml">]
<book>
 <title>A Physically Divided Book</title>
 &chap1;
 &chap2;
 &chap3;
 &appa;
 &appb;
</book>
```

You can then write the chapters and appendixes conveniently in separate files. This is why DocBook is well suited to large contents. Note that these separate files do not and must not have document type declarations.

For example, Chapter 1 might begin like this:

```
<chapter id="ch1">
<title>My First Chapter</title>
<para>My first paragraph.</para>…
……
```

Breaking a Document into Logical Portions

Here is a quick reference guide for DocBook Elements: http://www.docbook.org/tdg/en/html/ref-elements.html

There are–literally–hundreds of DocBook elements. This is what makes docBook very powerful. We will try to cover the major ones here and let you review the rest on your own. Firstly, a classification; DocBook Elements can be divided broadly into these categories:

Sets	collection of books
Books	books
Divisions	divide books into parts
Components	divide books or divisions into chapters
Sections	subdivide components
Meta-information Elements	contain information about other elements
Block Elements	occur at paragraph level
Inline Elements	used to mark up running text

Major DocBook Elements

Set: A collection of books

Set is the very top of the DocBook structural hierarchy. There's nothing that contains a *Set*.

Some children elements: Book, SetIndex, SetInfo, Subtitle, Title, TitleAbbrev, ToC(table of contents).

Reference page: http://www.oreilly.com/catalog/docbook/chapter/book/set.html

Table 7: <set> element, "lordoftherings.xml"

```
<!DOCTYPE set PUBLIC "-//OASIS//DTD DocBook V4.2//EN">
<set>
 <title>Lord of the Rings</title>
 <setinfo>
  <author>J.R. Tolkien</author>
 </setinfo>
 <book><title>The Fellowship of the Ring</title> ... </book>
 <book><title>The Two Towers</title> ... </book>
 <book><title>Return of the King</title> ... </book>
<set>
```

Book: A book

A *Book* is probably the most common top-level element in a document. The DocBook definition of a book is very loose and general. It gives you free rein by not imposing a strict ordering of elements.

Some children elements: Appendix, Article, Bibliography, BookInfo, Chapter, Colophon, Dedication, Glossary, Index, LoT, Part, Preface, Reference, SetIndex, Subtitle, Title, TitleAbbrev, ToC.

Reference page: http://www.oreilly.com/catalog/docbook/chapter/book/book.html

Table 8: <book> element, "xmlbook.xml"

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.2//EN">
<book>
 <title>XML - Managing Data Exchange</title>
 <titleabbrev>XML</titleabbrev>
 <bookinfo>
  <legalnotice><para>No notice is required.</para></legalnotice>
  <author><firstname>Rusen</firstname><surname>Gul</surname></author>
 </bookinfo>
 <dedication>
  <para>This book is dedicated to MIST 7700 class of 2004 at UGA.</para>
 </dedication>
 <preface>
  <title>Forword</title>
  <para>The book aims to fulfill the need for an introductory XML
  textbook. It contains the basics of XML as well as several tools
  using XML.</para>
  </preface>
```

```
 <chapter>
  <title>Introduction</title>
  <para>At least one chapter, reference, part, or article is required.</para>
 </chapter>
 <appendix>
  <title>Optional Appendix</title>
  <para>Appendixes are optional but handy.</para>
 </appendix>
</book>
```

Division: A collection of parts and references (optional)

*Division*s are the first hierarchical level below Book.

Children elements: Part (contain components), Reference (contain RefEntrys)

Components: Chapter-like elements of a Book or Part

These are Preface, Chapter, Appendix, Glossary, Bibliography, and Article. Components generally contain block elements -or sections, and some can contain navigational components and RefEntrys.

Table 9: <Bibliography> element, "references.xml"

```
<!DOCTYPE bibliography PUBLIC "-//OASIS//DTD DocBook 4.2//EN">
<bibliography>
 <title>References</title>
 <bibliomixed>
  <bibliomset relation=article>
   <surname>Watson</surname>
   <firstname>Richard</firstname>.
   <title role=article>Managing Global Communities </title>
  </bibliomset>
  <bibliomset relation=journal>
   <title>The World Wide Web Journal</title>
   <volumenum>2</volumenum>
   <issuenum>1</issuenum>.
   <publishername>O'Reilly & Associates, Inc.</publishername> and
```

```
   <corpname>The World Wide Web Consortium</corpname>.
   <pubdate>Winter, 1996</pubdate>
  </bibliomset>.
 </bibliomixed>
</bibliography>
```

Sections: Several sectioning elements

a. Sect1…Sect5 elements - the most common sectioning elements that can occur in most component-level elements. These numbered section elements must be properly nested (Sect2s can only occur inside Sect1s, Sect3s can only occur inside Sect2s, and so on).

b. Section element - an alternative to numbered sections Sections are recursive, meaning that you can nest them to any depth desired.

c. SimpleSect element - a terminal section that can occur at any level SimpleSect cannot have any other sectioning element nested within it.

d. BridgeHead element - a section title without any containing section

e. RefSect1…RefSect3 elements - numbered section elements in RefEntrys **f. GlossDiv, BiblioDiv, and IndexDiv elements** - do not nest

Please see Table 4 and Table 5 for examples.

Reference page: http://www.oreilly.com/catalog/docbook/chapter/book/section.html

Meta-Information Elements – contain bibliographic information

All of the elements at the section level and above include a wrapper for meta-information about the content. Examples of meta-wrappers: BookInfo, ArticleInfo, ChapterInfo, PrefaceInfo, SetInfo, GlossaryInfo.

Table 10: <bookinfo> element

```
<!DOCTYPE bookinfo PUBLIC "-//OASIS//DTD DocBook V4.2//EN">
<bookinfo>
```

```
 <title>XML - Managing Data Exchange</title>
 <authorgroup>
  <author>
   <firstname>Richard</firstname>
   <surname>Watson</surname>
  </author>
  <author>
   <firstname>Hendrik</firstname>
   <surname>Fischer</surname>
  </author>
  <author>
   <firstname>Rusen</firstname>
   <surname>Gul</surname>
   <affiliation>
    <orgname>University of Georgia</orgname>
   </affiliation>
  </author>
 </authorgroup>
 <edition>Introduction to XML - Version 1.0 </edition>
 <pubdate>1997</pubdate>
 <copyright>
  <year>1999</year>
  <year>2000</year>
  <year>2001</year>
  <year>2002</year>
  <year>2003</year>
  <holder> O'Reilly & Associates, Inc. </holder>
  </copyright>
  <legalnotice>
    <para>Permission to use, copy, modify and distribute the DocBook
    DTD and its accompanying documentation for any purpose and without
    fee is hereby granted in perpetuity, provided that the above
    copyright notice and this paragraph appear in all copies.
    </para>
   </legalnotice>
</bookinfo>
```

Block vs. Inline Elements

There are two classes of paragraph-level elements: *block* and *inline*.

Block elements are usually presented with a paragraph break before and after them. Most can contain other block elements, and many can contain character data and inline elements. Examples of block elements are: Paragraphs, lists, sidebars, tables, and block quotations.

Inline elements are generally represented without any obvious breaks. The most common distinguishing mark of inline elements is a font change, but inline elements may present no visual distinction at all. Inline elements contain character data and possibly other inline elements, but they never contain block elements. They are used to mark up data. Some examples are: cross references, filenames, commands, options, subscripts and superscripts, and glossary terms.

Block Elements - paragraph-level elements

The block elements occur immediately below the component and sectioning elements.

Lists

CalloutList	A list of marks, frequently numbered and typically on a graphic or verbatim environment and their descriptions.
GlossList	A list of glossary terms and their definitions
ItemizedList	An unordered (bulleted) list
OrderedList	A numbered list
SegmentedList	A repeating set of named items. For example, a list of states and their capitals might be represented as a SegmentedList.
SimpleList	An unadorned list of items
VariableList	A list of terms and definitions or descriptions

Table 11: <segmentedlist> element, "statecapital.xml"

```
<!DOCTYPE para PUBLIC "-//OASIS//DTD DocBook V4.2//EN">
<para>The capitals of the states of the United States of America are:
 <segmentedlist>
  <title>State Capitals</title>
  <segtitle>State</segtitle>
  <segtitle>Capital</segtitle>
  <seglistitem>
   <seg>Georgia</seg>
   <seg>Atlanta</seg>
  </seglistitem>
  <seglistitem>
```

```
   <seg>Alaska</seg>
   <seg>Anchorage</seg>
  </seglistitem>
  <seglistitem>
   <seg>Arkansas</seg>
   <seg>Little Rock</seg>
  </seglistitem>
 </segmentedlist>
</para>
```

Table 12: "statecapital.xml" output

The capitals of the states of the United States of America are:

State Capitals

State: Georgia

Capital: Atlanta

State: Alaska

Capital: Anchorage

State: Arkansas

Capital: Little Rock

Table 13: <orderedlist> element, "mashpotatoe.xml"

```
<!DOCTYPE para PUBLIC "-//OASIS//DTD DocBook V4.2//EN">
<para>
 <orderedlist numeration="upperroman">
  <listitem>
   <para>Preparation</para>
   <orderedlist numeration="upperalpha">
    <listitem><para>Chop tomatoes</para>
    </listitem>
    <listitem><para>Peel onions</para>
    </listitem>
    <listitem><para>Mash potatoes</para>
    </listitem>
   </orderedlist>
  </listitem>
  <listitem>
   <para>Cooking</para>
```

```
   <orderedlist numeration="upperalpha">
   <listitem><para>Boil water</para>
   </listitem>
   <listitem><para>Put tomatoes and onions in </para></listitem>
   <listitem><para>Blanch for 5 minutes</para>
   </listitem>
    </orderedlist>
  </listitem>
 </orderedlist>
</para>
```

Table 14: "mashpotatoe.xml" output

I.Preparation
A.Chop tomatoes
B.Peel onions
C.Mash potatoes
II.Cooking
A.Boil water
B.Put tomatoes and onions in
C.Blanch for 5 minutes

Admonitions

There are five types of *admonition*s: Caution, Important, Note, Tip, and Warning.

Table 15: <caution> element, "caution.xml"

```
<!DOCTYPE caution PUBLIC "-//OASIS//DTD DocBook V4.2//EN">
<caution>
 <title>This is a caution</title>
 <para>Be careful while opening the box!</para>
</caution>
```

Line-specific environments

Line-specific environments preserve whitespace and line breaks.

Address	A real-world address, generally a postal address

LiteralLayout	A block of text in which line breaks and white space are to be reproduced faithfully
ProgramListing	A literal listing of all or part of a program
Screen	Text that a user sees or might see on a computer screen
ScreenShot	A representation of what the user sees or might see on a computer screen
Synopsis	A general-purpose element for representing the syntax of commands or functions

Table 16: <literallayout> element, "If_by_Kipling.xml"

```
<!DOCTYPE blockquote PUBLIC "-//OASIS//DTD DocBook V4.2//EN">
<blockquote>
 <attribution>Rudyard Kipling,
  <citetitle>If</citetitle>
 </attribution>
 <literallayout>
     If you can force your heart and nerve and sinew
To serve your turn long after they are gone,
And so hold on when is nothing in you
Except the Will
which says to them:
     Hold on!
 </literallayout>
</blockquote>
```

Common block-level elements

Common block-level elements include Examples, figures, and tables. The distinction between formal and informal elements is that formal elements have titles while informal ones do not.

Example, InformalExample

Table 17: <example> element

```
<!DOCTYPE example PUBLIC "-//OASIS//DTD DocBook V4.2//EN">
<example>
 <title>Sample code</title>
 <programlisting>print "Hello, world!"</programlisting>
</example>
```

Figure, InformalFigure

Table 18: <figure> element

```
<!DOCTYPE figure PUBLIC "-//OASIS//DTD DocBook V4.2//EN">
<figure>
 <title>Revenues for Q1</title>
 <mediaobject>
  <imageobject>
   <imagedata fileref="q1revenue.jpg" format="JPG"/>
  </imageobject>
 </mediaobject>
</figure>
```

Table, InformalTable

Table 19: <table> element

```
<!DOCTYPE table PUBLIC "-//OASIS//DTD DocBook V4.2//EN">
<table frame="frametype">
 <title>frame="frametype"</title>
 <tgroup cols="1">
  <thead>
   <row>
    <entry>row 1, cell 1</entry>
    <entry>row 1, cell 2</entry>
    <entry>row 1, cell 3</entry>
   </row>
  </thead>
  <tbody>
   <row>
    <entry>row 2, cell 1</entry>
    <entry>row 2, cell 2</entry>
    <entry>row 3, cell 3</entry>
   </row>
  </tbody>
 </tgroup>
</table>
```

Paragraphs

Paragraphs are Para, SimPara (simple paragraphs may not contain other block-level elements), and FormalPara (formal paragraphs have titles). Paragraphs are the most commonly used high-level elements that can contain block elements such as itemizedlist and Mediaobject and can contain almost all inline elements.

Reference page: http://www.docbook.org/tdg/en/html/para.html

Table 20: <para> element, "Nietzsche.xml"

```
<!DOCTYPE para PUBLIC "-//OASIS//DTD DocBook V4.2//EN"> <para>
 <quote>Behold the superfluous. They are always sick. They vomit their gall and call it a newspa-
per.</quote>
 -Friedrich Wilhelm Nietzsche,
 <citetitle>Twilight of the Idols</citetitle>
</para>
```

Equations

Equation and InformalEquation (without titles)

Table 21: <informalequation> element inside a <para> element

```
<!DOCTYPE para PUBLIC "-//OASIS//DTD DocBook V3.1//EN"> <para>
  The equation
  <informalequation>
   <alt>e^(pi*i) + 1 = 0</alt>
   <graphic fileref="figures/epi10"></graphic>
  </informalequation>
is delightful because it joins together five of the most important mathematical constants.
</para>
```

Graphics

InlineGraphic, MediaObject, InlineMediaObject

These elements may contain video, audio, image, and text data. A single media object can contain several alternative forms from which the presentation system can select the most appropriate object.

Please see Table 18 for example.

Inline Elements – used to mark up running text

In published documents, inline elements often cause a font change or other small change, but they do not cause line or paragraph breaks.

Abbrev	An abbreviation, especially one followed by a period.
Acronym	An often pronounceable word made from the initial (or selected) letters of a name or phrase.
Emphasis	Emphasized text.
Footnote	A footnote. The location of the Footnote element identifies the location of the first reference to the footnote. Additional references to the same footnote can be inserted with FootnoteRef.
Phrase	A span of text.
Quote	An inline quotation.
Trademark	A trademark.
Citation	An inline bibliographic reference to another published work.
GlossTerm	A glossary term.
Link	A hypertext link.
ULink	A link that addresses its target by means of a URL (Uniform Resource Locator).
XRef	A cross reference to another part of the document.

ForeignPhrase	A word or phrase in a language other than the primary language of the document.
ComputerOutput	Data, generally text, displayed or presented by a computer.
Markup	A string of formatting markup in text that is to be represented literally.
Replaceable	Content that may or must be replaced by the user.
UserInput	Data entered by the user.
Literal	Inline text that is some literal value.
Command	The name of an executable program or other software command.
MsgText	The actual text of a message component in a message set.
Optional	Optional information.
Email	An email address.
Database	The name of a database, or part of a database.
Filename	The name of a file.
Token	A unit of information.
Type	The classification of a value.
Application	The name of a software program.

Entities for Special Characters

The following entities are provided for special characters:

Character	Entity
<	& lt;
>	& gt;
&	& amp;
"	& quot;
'	'

Publishing a DocBook Document

DSSSL vs XSL Stylesheets:

Document Style Semantics and Specification Language (DSSSL) is a stylesheet language for both print and online rendering. It is mainly intended to work with SGML.

Extensible Stylesheet Language (XSL) is a language for expressing stylesheets written in XML. It includes the formatting object language, but refers to separate documents for the transformation language and the path language. In this chapter, we will use XSL Stylesheets because they're more powerful, you are already familiar with them, and they are intended to work with XML.

Step 1: Get the Standard StyleSheets

DocBook strictly separates the content and appearance of a document. A DocBook document only explains the semantics of the document, not its formatting or appearance. In order to publish your DocBook Document, you will need to use a set of DSSSL or XSL Stylesheets describing the formatting and an XSL processor.

If you're thinking that it would be a lot of work to write your own XSL stylesheets, you're right. The good news is that you don't need to. There are a large number of freely available standard XSL stylesheets for DocBook maintained primarily by Norman Walsh.

Make sure that you download the latest version of these stylesheets at Sourceforge.net - Stylesheets Repository. The stylesheet distribution consists of a collection of modular XSL files that are assembled into several complete XSL stylesheets. There is a stylesheet for generating a single HTML file, and one for generating multiple smaller HTML files from a single DocBook document. There are stylesheets for print output, XHTML output, HTML Help output, and JavaHelp output. Since there are XSL processors for all major computer types, you can use DocBook on Unix, Linux, Windows, and Macintosh computers. By using these default stylesheets installed on your system, it is quite easy to create customized stylesheets. But don't forget to note that the common approach to customize the stylesheets is creating a customization layer rather than editing them directly.

Step 2: Download an XSLT processor

To publish HTML from your XML documents, you will need an XSLT engine. To print, you need an XSLT engine to produce formatting objects (FO), which then must be processed with an FO engine to produce PostScript or PDF output. A variety of XSLT engines are available. Here's a list of some free/open-source ones you might consider. Note that xsltproc and Saxon are currently the only recommended XSLT engines for use with DocBook.

XSLT Engines

1. **Xsltproc:** A free processor written in C, available as part of the open source libxml2 library from the Gnome development project. It is considered the fastest of the processors, and is highly conformant to the specification. Download at http://xmlsoft.org/XSLT/
2. **Saxon:** A free processor written in Java, that can be run on any operating system with a modern Java interpreter. It uses the Aelfred XML parser internally, which has some bugs, so many people substitute the Xerces parser. Download at http://saxon.sourceforge.net/
3. **Xalan:** Xalan is part of the Apache XML Project. It has versions written in both Java and C++, both of them free. The Java version is highly portable and more fully developed. Generally Xalan is used with the Xerces XML parser (Java or C++), also available from the Apache XML Project. Download at http://xml.apache.org/

Your choice of an XSLT engine may depend a lot on the environment in which you'll be running the engine. Many DocBook users who need or want a non-Java application use xsltproc. It's very fast and the developers respond very quickly to bug reports and questions. But one current limitation xsltproc has is that it doesn't yet support Norm Walsh's DocBook-specific XSLT extension functions.

Saxon is the most popular one for use in a Java environment. It also supports Norm Walsh's DocBook-specific XSLT extension functions.

NetBeans IDE, the XML editor we've been using for other chapters' exercises, has a built-in XSLT processor using the XALAN parser by default. NetBeans IDE not only lets you validate your XML documents but also does XSL transformations right there in the IDE. It will work for this chapter's purpose well enough. It doesn't provide any XSLT debugging though, so you might want to get yourself a decent XSL IDE (e.g.,XML Spy or Xcelerator) for serious XSLT work.

FO Engines

For generating print/PDF output from FO files, there are two free/open-source FO engines:

1. **PassiveTeX:** available at http://www.tei-c.org.uk/Software/passivetex/index.xml.ID=body.1_div.1. Download and installation information at http://www.tei-c.org.uk/Software/passivetex/index.xml.ID=body.1_div.3.
2. **FOP:** a Java-based processor from the Apache XML Project, available at http://xml.apache.org/fop/

Step 3: Customize the XSL stylesheets

Output to HTML

- The main strength of standard stylesheets is that they are easily customizable.
- Parameters found in params.xsl
- Call your customization layer instead of the standard stylesheet

Table 22: A customized XSL stylesheet, "myxsl1.xsl"

```
<?xml version="1.0"?>
  <!-- Customization layer -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <!-- Use 'chunk.xsl' in line below to chunk files. -->
<xsl:import href="/usr/share/sgml/docbook/docbook-xsl-1.51.1/html/docbook.xsl"/>
<xsl:param name="chapter.autolabel" select="1"/>
<xsl:param name="section.autolabel" select="1"/>
<xsl:param name="section.label.includes.component.label" select="1"
doc:type="boolean"/>
  <!-- Insert more parameters here. -->
</xsl:stylesheet>
```

- Beyond setting parameters, you can modify XSLT "templates" to override default behavior
- You need at least a minimal knowledge of XSLT

Table 23: A customized XSL stylesheet, "myxsl2.xsl"

```
<xsl:template match="emphasis">
<xsl:choose>
<xsl:when test="(@role='strong') or (@role='bold')">
  <xsl:call-template name="inline.boldseq"/>
</xsl:when>
<xsl:otherwise>
  <xsl:call-template name="inline.italicseq"/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>
```

Output to PDF

It generally requires a two-stage process:

1. Generation of FO from XML
2. Generation of PDF from FO

Table 24: An XSL stylesheet to generate FO, "myxsl3.xsl"

```
xsltproc -o sample.fo  $DB/fo/docbook.xsl sample.xml
fop.sh -fo sample.fo -pdf sample.pdf
```

Extensions

a. Slides Doctype

- Creation of presentation slides from DocBook XML
- You can create HTML (with or without frames) and FO
- Uses DocBook elements within a specific hierarchical framework
- Downloadable from DocBook Open Repository at SourceForge

Table 25: "slides.xml"

```
<!DOCTYPE slides SYSTEM "/usr/share/sgml/docbook/xsl-slides-1.1/slides.dtd">
<slides>
 <slidesinfo>
```

```
 <title>A Simple Approach to DocBook</title>
</slidesinfo>
<foil>
 <title>My first slide</title>
 <itemizedlist>
  <listitem><para>...</para></listitem>
  <listitem><para>...</para></listitem>
  <listitem><para>...</para></listitem>
 </itemizedlist>
</foil>
<foil>
 <title>My second slide</title>
 <para>... </para>
</foil>
</slides>
```

b. Website Doctype

- Creation of web sites from a collection of DocBook XML files
- Uses most DocBook elements within specific framework. It has separate files that control page navigation and hierarchy.
- Downloadable from DocBook Open Repository at SourceForge.

Table 26: "website.xml"

```
<!DOCTYPE webpage SYSTEM "../website.dtd" [
<!NOTATION XML SYSTEM "xml">
<!ENTITY test1a SYSTEM "test1a.xml" NDATA XML>
<!ENTITY test3 SYSTEM "test3.xml" NDATA XML>
<!ENTITY about.xml SYSTEM "about.xml" NDATA XML>]>
<webpage id="home">
```

```
<config param="desc" value="The Test Home Page"/>
<config param="rcsdate" value="$Date: 2001/11/08 20:44:20 $"/>
<config param="footer" value="about.html" altval="About..."/>
<head>
<title>Welcome to Website</title>
<summary>Introduction</summary>
<keywords>Rusen Gul, XSL, XML, DocBook, Website</keywords>
</head>
<para> This website demonstrates the DocBook.</para>
<webtoc/>
<section>
<title>What is a Website?</title>
<para>A website is a collection of pages organized, for the purposes
of navigation, into one or more hierarchies. In Website, each page
is a separate XML document authored according to the Website DTD,
a customization of <ulink url="http://www.oasis-open.org/doc-
book/">DocBook</ulink>.</para>
</section>
</webpage>
```

Why use DocBook?

This certainly looks like too much work, doesn't it? You're not wrong. Why do we bother to use DocBook then?

- It is portable! A document written in DocBook markup can be converted into HTML, PostScript, PDF, RTF, DVI, plain ASCII text easily and quickly without any expensive tools.
- It is flexible! It enables output to multiple formats, including HTML, PDF, Slides, and many others.
- It separates the content from format! DocBook is only concerned with the structure of a document. It frees the author from worrying about the formatting and layout of a document.
- It is easy to understand! Most of DocBook elements are self explanatory.
- It can handle large quantities of content. You can physically divide the document into different files and work on them separately and conveniently.

- It is free! There a lot of freely available open source tools used to work with DocBook.

DocBook is well suited to any collection of technical documentation that is regularly maintained and published. Multiple authors can contribute to a single document, and their content can easily be merged because all the authors are using a highly structured, standard markup language. Just one little point to keep in mind; because the formatting for DocBook documents is strictly accomplished by stylesheets, DocBook is not well matched to highly designed layout-driven content like magazines.

Setting up a DocBook system will certainly take some time and effort. The payoff will be an efficient, flexible, and inexpensive publishing system that is iterative and that can grow with your needs. Therefore, it is worth the effort!

DocBook Filters - Reading and Writing DocBook XML Using OpenOffice.org

The goal of the project is to use OpenOffice.org as a WYSIWYG editor of XML content to edit structured documents using styles. When exported, these styles are then transformed to XML tags. This section shows you how to enable and use DocBook filters. Below are some links to stylesheets that can be download to use the latest transformations.

Enabling the DocBook XSLT's in OpenOffice.org 1.1 Beta 2/RC

There are three different ways to enable the DocBook filters.

1. Download the DocBook XSLT Stylesheets and OpenOffice.org Style Template
    - Using this method will make certain that the most recent stylesheets and OpenOffice.org style template will be used for import and export. It is required to download the following to import, export and modify DocBook documents in OpenOffice.org:
    - The relevant XSLT stylesheets for the XML transformations (All available here)
    - An OpenOffice.org style template that contains custom styles corresponding to DocBook tags (Available here)

The most recent stylesheets support the import and export of DocBook documents with article or chapter as the top-level tag. The different stylesheets required for each of these operation are listed below:

- Stylesheets required for import Article docbooktosoffheadings.xsl
- Stylesheets required for import Chapter docbooktosoffheadings.xsl
- Stylesheets required for export Article sofftodocbookheadings_article.xsl
- Stylesheets required for export Chapter sofftodocbookheadings_chapter.xsl

OpenOffice.org Template required for Article and Chapter documents:

- DocBookTemplate.stw

Creating a new DocBook filter

- Go to Tools -> XML Filter Settings...
- Set Filter Name and Name of File Type to DocBook (Chapter)
- Go to the Transformation tab
- Set DocType to <chapter>
- For XSLT for Export browse to the chapter export stylesheet (docbooktosoffheadings.xsl).
- For XSLT for Import browse to the chapter import stylesheet (sofftodocbookheadings_chapter.xsl).
- For Template for Import browse to the style template (DocBookTemplate.stw).
- Click OK and close the XSLT Filter Setting dialog

To create a DocBook Article filter, the above steps can be repeated with article replacing chapter

1. Download the DocBook XSLT Jar Packages for Article or Chapter

This method is more convenient, however there is no guarantee that the most recent stylesheets and OpenOffice.org template will be used.

1. Download the DocBook UNO component for Article only

The DocBook UNO component adds filter support for the retention of unresolved XML entities.

- Download the DocBookFilter
- Unzip it to the <OOo install Dir>/
- Run pkgchk in the <OOo install Dir>/program dir

- The DocBook Article filter will now import DocBook unresolved entities as OpenOffice.org set variables

How to Import a DocBook document

A DocBook article or chapter document can now be opened using the File -> Open dialog.

- Go to File -> Open...
- Browse to the DocBook document.
- Click OK

The DocBook XSLT filter should automatically determine the root element of the document and import it with the matching XSLT filter. Alternatively, it is possible to browse manually to the desired DocBook filter in the File Type combo-box in the File -> Open dialog.

How to Export a DocBook document

The DocBook document can also be exported using the File -> Save As dialog.

- Go to File -> Save As...
- Browse to the location where the document is to be saved
- Click Save

Again, the DocBook XSLT filter should automatically determine the file type and export with the matching XSLT filter. Alternatively, it is possible to browse manually to the desired DocBook filter in the File Type combo-box in the File -> Save As dialog.

Using OpenOffice.org Headings and Styles for different DocBook tags

Using OpenOffice.org styles to represent DocBook tags The style template supplies all of the custom styles that are currently supported. Once a DocBook document has been imported to Open-Office.org, the available DocBook specific styles can be viewed using the Stylist. On import, each of the supported DocBook tags will be mapped to formatted OpenOffice.org content. Similarly, to

modify the imported DocBook document, OpenOffice.org text styles can be used to represent the DocBook tags marking-up the text. NOTE: A new DocBook document can be created in OpenOffice.org by opening the DocBookTemplate.stw. The document can then be saved as a DocBook document, and the new content will be represented as DocBook mark-up. How to create new DocBook content:

- Press F11 to display the Stylist
- Select Custom Styles in the Stylist combo-box
- Click the Character Styles icon (second from left on the Stylist)
- Double-click the SubScript style
- Enter text in the OpenOffice.org document
- On exporting as DocBook, the text formatted as the SubScript custom style will be marked-up with the DocBook tag <subscript>

How to create DocBook sections: Initially the DocBook project used OpenOffice.org sections to enforce the nesting of DocBook sections. Feedback has shown that authors wish to use the common word processing styles such as Heading1, Heading2, etc. The following instructions describe how to create a <sect1> that contains a <sect2>

- Press F11 to display the Stylist
- Select All Styles in the Stylist combo-box
- Click the Paragraph Styles icon (first in the left on the Stylist)
- Double-click the Heading 1 style
- Enter the text to be the <sect1> title
- All the text below this heading will now be the content of the DocBook <sect1>
- Enter other DocBook styles, tables, etc.
- Enter other DocBook styles, tables, etc. to be included in <sect1>

1. Double-click the Heading 2 style
2. Enter the text to be the <sect2> title
3. All the text below this heading will now be the content of the DocBook <sect2>
4. Enter other DocBook styles, tables, etc. to be included in <sect2>

- This nesting of DocBook sect's using OpenOffice headings can go as far as <sect4> / Heading 4

Navigating through the document: If you wish to see how DocBook sections are nested as OpenOffice.org headings, use the F5 key to Display the Navigator window. Expand the headings tag, to display the layout of the headings within the document. You can skip to the start of a given DocBook section/OpenOffice.org heading, by double-clicking on it.

Exercises

1. Download the latest Docbook DTD at http://www.oasis-open.org/docbook/xml/. Convert the Learning Objectives part of Chapter 2 of this textbook into a DocBook document. Check that it is well-formed and valid.
2. Download the DocBook XSL Stylesheets at Sourceforge.net. Transform the DocBook document you created in the first exercise into an HTML file by using the *docbook.xsl* stylesheet in the *HTML* folder.

References and Useful Links

1. Docbook Official Website
2. DocBook: The Definitive Guide, by Norman Walsh and Leonard Muellner, published by O'Reilly & Associates, October 1999 http://www.docbook.org/
3. Docbook Wiki
4. Sourceforge - Docbook Open Repository
5. Installing and Using DocBook - Copyright 2002, The University Of Birmingham
6. Using the DocBook XSL Stylesheets - http://www.sagehill.net/docbookxsl/index.html
7. Setting Up A Free XML/SGML DocBook Editing Suite For Windows And Unix
8. http://lists.oasis-open.org/archives/docbook-apps/
9. http://www.dulug.duke.edu/~mark/docbookmarks/
10. http://www.linuxdoc.org/LDP/LDP-Author-Guide/
11. http://www.nwalsh.com/docs/
12. http://www.e-smith.org/docs/docprocess.html
13. http://www.lodestar2.com/people/dyork/talks/docbook/

14. DocBook mailing list: mailto:docbook@lists.oasis-open.org
15. http://xml.openoffice.org/xmerge/docbook/