A Practical Introduction to DocBook

Deb Richardson
Open Source Writers Group

deb@oswg.org

A practical introduction to using DocBook, written with the new user in mind. Over time, this will grow to include a task-based tag reference and many more examples.

1. Copyright and Licensing

The current "master" copy of this document can be obtained through the Open Source Writers Group website at http://www.oswg.org.

2. Introduction

The Open Source Writers Group (OSWG - http://www.oswg.org) uses the DocBook SGML Document Type Definition (DTD) as its standard "source" format for all documentation, including books, articles, papers, FAQs, HOWTOs, mini-HOWTOs, and more.

DocBook was chosen for a number of reasons – it's a very verbose DTD, it covers almost every possible requirement for technical and other forms of documentation, and it is quickly becoming the standard DTD for open-source and open-content information. There are also a number of open-source tools available for processing DocBook instances, some of which work quite well. Additionally, Norman Walsh's modular DocBook stylesheets are freely available, very complete, and under active development. All of these things make DocBook a compelling choice as a standard SGML DTD.

This document is a quick introduction to using DocBook to markup documentation that is intended to be kept in the Open Source Writers Group Concurrent Versions System (CVS). This CVS is freely available for any writer or group of writers who are working on free open-content documentation of any sort. For more information about the CVS system and the OSWG documentation licensing policies, please visit the OSWG website at http://www.oswg.org.

For a far more complete DocBook reference, visit http://www.docbook.org, the home of the electronic versions of Norman Walsh's DocBook: The Definitive Guide (published by O'Reilly http://www.oreilly.com). Better yet, head on over to your local (real or virtual) bookshop and order a printed version. If you're going to delve into the wonderful world of DocBook, you will definitely get your money's worth out of Norman's book.

If you're new to markup languages, I recommend you read at least the first two chapters of DocBook: The Definitive Guide, as these give an overview of HTML, SGML, and XML, as well as an introduction to the major concepts involved in doing structural and semantic markup.

3. Tools

There are two different types of tool related to SGML – the tools you use to create SGML documents, and the tools you use to process SGML documents.

These tools are very different, because they do completely different things.

3.1. Authoring tools

As mentioned in the Introduction, you don't really need any special tools for creating or authoring DocBook documents. In fact, at the moment, your best bet is to use your favourite text editor. DocBook documents must be in plaintext ASCII, not in any special binary format. Thus, you can use vi, emacs, notepad, textpad, or any other text editor that allows you to produce plain ASCII output.

There are some tools being developed that will allow you to create DocBook and other types of SGML documents without having to manually type in all the markup tags. At the moment, however, those tools are not available, or if they are available, they're not all that reliable. For now, it's recommended that you just use a text editor, and add your own markup tags by hand. It's a bit of a bother, but it's not difficult. The end result is definitely worth the time invested.

3.2. Processing tools - DocBook Tools

SGML processing tools are sets of scripts and programs that are used to turn SGML documents into other output formats. It is during this processing that all formatting is added and the final versions of the documents are created.

The OSWG uses a package called "DocBook Tools" to process the DocBook instances that are part of its documentation set. This package includes a set of scripts that simplify the use of a number of different programs, making it very easy to create a variety of different output formats, including HTML, PDF, PS, RTF, DVI, and TEX.

3.2.1. Getting DocBook Tools

You can get all the packages necessary for DocBook Tools at http://sourcware.cygnus.com/docbook-tools/

(http://sourceware.cygnus.com/docbook-tools/).

3.2.2. Installing DocBook Tools

Unfortunately, installing and configuring DocBook Tools is (currently) outside the scope of this document. Refer to the DocBook Tools website for help and documentation. If you have problems, there is a DocBook Tools mailing list called "docbook-tools-discuss" where you can get help from other DocBook Tools users. Subscription information is available on the site.

If you cannot get the help you need on the docbook-tools-discuss list, you can join the oswg-docbook mailing list, and ask there. To subscribe, send a message to the following address with "subscribe" in the subject line of the message:

<oswg-docbook-request@oswg.org>

3.2.3. Using DocBook Tools

Using DocBook Tools is exceedingly simple. When you install the packages, a series of scripts are added to /usr/bin. Each script turns your DocBook document into a different output format. The scripts are called:

db2html - DocBook to HTML

db2dvi - DocBook to DVI

db2pdf - DocBook to PDF

db2ps - DocBook to PS

db2rtf - DocBook to RTF

Using these scripts is very simple, just type the script name on the command line followed by the name of the DocBook file you want to process. For example:

```
$ db2html my-document.sgml
```

If you have any problems with this, you can ask for help on either of the mailing lists mentioned above.

4. Getting Started

Using the DocBook DTD to mark up your documentation doesn't require any special tools at all. An SGML file (usually having a .sgml or .sgm filename extension) is actually just a plaintext ASCII file that contains text interspersed with SGML "tags". I strongly recommend that you use your favourite text editor (vi, emacs, notepad, what have you) for doing DocBook markup. It's very important that the files are plaintext ASCII, so using a full-blown wordprocessing system is not only overkill, you might end up creating non-ASCII files. Use a text editor, it's still the best and simplest way.

A document that has been marked up with DocBook looks a lot like HTML in a lot of ways. For example, here's a very simple HTML page:

```
<html>
<head>
<title>My HTML Page</title>
</head>
<body bgcolor="#000000">
<h1>This is a heading</h1>
<p>
This is the first paragraph of the HTML page.
</p>
<p>
This is the second paragraph of the HTML page.
</p>
</body>
</html>
```

Markup tags are the things that begin and end with the "<" and ">" brackets. A markup "start" tag, which indicates the beginning of an "element" in the page just has the brackets, while an "end" tag, indicating the end of an element, begins with a "</" combination. Anything contained within a start and end tag of the same type is a complete "element" within the page.

So, what's an "element"? A document element is simply a section of the document that is defined by the tags which contain it. For example, the following is a complete HTML "H1" (heading level 1) element:

```
<h1>This is a Level 1 Heading</h1>
```

Elements can contain certain other elements, as defined by the DTD. For example, an HTML "body" element can contain headings, paragraphs, lists, and a wide variety of others. For example:

```
<body>
<h1>This is a Level 1 Heading</h1>
<p>
Here is a paragraph.
</p>
</body>
```

You will see that both the "h1" and "p" elements are contained by the "body" element.

The similarities between HTML and DocBook SGML pretty much end here, however. HTML is a markup language that is used primarily to define the formatting or appearance of the page when it is displayed in an HTML browser. HTML is also very informal in terms of how the tags fit together and how strict the tag structure must be.

DocBook SGML, however, is a markup language used to define the structure of a document, rather than its formatting. In DocBook there are no tags that you can use to make an element "bold" or "italic". Instead, you mark up the document by defining _what_ is in each element, rather than how each element should look. Here's a very simple DocBook example:

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
<article>

  <artheader>
    <title>This is the Article Title</title>
  </artheader>

  <sect1>
    <title>This is a Level 1 Section Title</title>
    <para>
    This is the first paragraph.
    </para>
    <para>
    This is the second paragraph.
    </para>
  </sect1>

</article>
```

As you can see, there is no formatting information in this document.

The "article" tags tell us that the document is an article. All other elements in the document are contained within this "root" element. The first pair of "title" tags indicate the article title, which is contained in the "artheader" (article header) element. The second set of "title" tags, contained by the "sect1" element, indicate the title of that section.

So, what's that "DOCTYPE" tag at the very beginning? That's the "document type declaration" which is how we define what type of SGML document this is and which DTD we're using for the markup. You don't have to understand the document type declaration, you just have to make sure that it's at the top of all the documents you markup using DocBook. You might have to change the word after "DOCTYPE", depending on what sort of document you're creating: the example is an "article", but you might want to create a "book", a "chapter", a "refentry", or something else.

In terms of the OSWG documentation set, the most common type of document is an "article". Articles include HOWTOs, mini-HOWTOs, magazine articles, papers, and such. An "article" is actually a general class of document that has a certain length and structural complexity. Unless you're doing a manual page ("man" page) or a proper book, chances are that your OSWG document will be an "article".

5. Step by Step Example

Now we'll walk through the process of creating a simple DocBook article. This example includes some of the most commonly used DocBook elements, and is here mostly to show you what a document marked up with DocBook looks like.

If you're looking for a more complete list of DocBook elements and their uses, see the Element Reference, or Norman Walsh's DocBook: The Definitive Guide, available at http://www.docbook.org.

5.1. Step 1: Including the Document Type Declaration

Assuming that you're marking up an "article", the first thing you have to have in your document is the document type declaration:

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
```

5.2. Step 2: Adding a "root" element

After that, you have to define the "root" element, which will contain all the other elements in your document. For an article, the root element is simply "article". So, add that, and its end tag after the document type declaration:

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
<article>
```

```
</article>
```

Everything else you add to your document will be contained within the "article" tags, so when you're done, the last tag in your document should be "</article>".

5.3. Step 3: Adding an article header

Next, you'll want to add some "header" information. Header information includes the article's title, the author's name and email address, the revision history of the document, and lots of other stuff. For now, we'll just add the article title and the author name.

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
<article>

<artheader>
  <title>How I Spent My Summer Vacation</title>
  <author>
    <firstname>Deb</firstname>
    <surname>Richardson</surname>
  </author>
</artheader>

</article>
```

5.4. Step 4: Adding a first level sectioin

So, now that we have some header information added, lets actually add some content in a level 1 section with a title:

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
<article>
```

```
<artheader>
   <title>How I Spent My Summer Vacation</title>
   <author>
      <firstname>Deb</firstname>
      <surname>Richardson</surname>
   </author>
</artheader>

<sect1>
<title>The Month of May</title>
<para>
My summer vacation started on May 14th, right after graduation.  Being a
new, unemployed, and highly-debt-ridden university graduate, I decided
that I should probably get a job.  I spent the vast majority of May
padding my resume with all sorts of impressive-sounding yet terribly
obscure skills and accomplishments, and sending those out to any company
whose name appeared in the phone book.
</para>
</sect1>

</article>
```

I think you're probably getting the hang of it at this point.

5.5. Step 5: Adding subsections and sub-subsections

Adding subsections and sub-subsections is also simple. A second level section starts and ends with "sect2" tags, third level with "sect3", and so on. The standard DocBook DTD only supports up to a level 5 section, so there are no "sect6", "sect7" or higher tags. Unless you're doing a massively long and professional book, however, you probably won't need more than 5 section levels.

Each section level should be contained within a higher level, so level 2 sections should be contained by level 1 sections, and so on. Here's an example:

```
<sect1>
```

```
<title>The Month of May</title>
<para>
My summer vacation started on May 14th, right after graduation.  Being a
new, unemployed, and highly-debt-ridden university graduate, I decided
that I should probably get a job.  I spent the vast majority of May
padding my resume with all sorts of impressive-sounding yet terribly
obscure skills and accomplishments, and sending those out to any company
whose name appeared in the phone book.
</para>

<sect2>
<title>My Resume</title>
<para>
My resume was 4 pages long, and mostly contained notes about my
education and employment history, which was sparse, at best.
</para>

<sect3>
<title>Employment History</title>
<para>
Most of the jobs I've ever had have been in the food service industry.
Like most aspiring writers, I spent most of my formative years wiping
tables and grumbling about lousy tips.
</para>
</sect3>

</sect2>

</sect1>
```

As you can see, this example ends with three end tags – each section level is contained or "nested" within the higher level.

6. Element Reference

The following is a short and incomplete guide to the most commonly used DocBook elements. I'm adding to this list as I find I need the elements, so expect this section to grow and evolve over time. Keep in mind that I'm pretty new to this whole DocBook thing, too, and I cannot guarantee that I'm using all of the elements as intended. In other words: everything in this book may be wrong ;>

6.1. Article header elements

Any DocBook "article" should include an article header. The article header contains information such as the article title, author, revision history, abstract, and so forth.

6.1.1. Article header

artheader: the "artheader" element contains all of the other elements that make up the article header.

6.1.2. Article title

title: the "title" element contained in the article header defines the article title. So, if you're writing an article called 101 Stupid Ferret Tricks, your article header would look like this:

```
<article>
  <artheader>
    <title>101 Stupid Ferret Tricks</title>
  </artheader>

...

</article>
```

6.1.3. Author information

author is the element which contains all of your author-related information, including first name, surname, email address, organizational affiliation, etc. Each of these pieces of information is marked up separately within the "author" element, which might seem like a bit of a pain, but it's worth it. The more comprehensive the markup, the more we can do with the information when processing the document.

All that aside, here's an example of an article header that contains an author's first name, surname, organizational affiliation, and email address. Usually all you need is the name and email address, and there's other stuff you can add. For now, we'll keep it simple:

```
<article>
  <artheader>

    ...

    <author>
      <firstname>Deb</firstname>
      <surname>Richardson</surname>
      <affiliation>
        <orgname>Open Source Writers Group</orgname>
        <address>
          <email>deb@oswg.org</email>
        </address>
      </affiliation>
    </author>

    ...

  </artheader>

...

</article>
```

6.1.4. Revision information

Most technical documentation goes through a series of revisions as the writer improves and updates the document. In order that readers know which version of a document s/he is reading, technical documentation often includes a revision number and other information. In a DocBook article, the revision information is part of the article header, and is contained in the revhistory element. Here's an example:

```
<article>
  <artheader>

    ...

    <revhistory>

      <revision>
        <revnumber>0.1</revnumber>
        <date>19 June 1999</date>
        <authorinitials>dr - deb@oswg.org</authorinitials>
        <revremarks>
          First draft, incomplete
        </revremarks>
      </revision>

      <revision>
        <revnumber>0.2</revnumber>
        <date>20 Sept 1999</date>
        <authorinitials>dr</authorinitials>
        <revremarks>
          Updated section 14, added section 11.5
        </revremarks>
      </revision>

    </revhistory>

    ...
```

```
  </artheader>

...

</article>
```

6.1.5. Article abstract

Many people write abstracts for their articles, which are short summaries, usually no more than a paragraph long, that tell the reader just what the article is about. In a DocBook article, the abstract is part of the article header, contained in an abstract element.

```
<article>
  <artheader>

  ...

    <abstract>
      <para>
        This article describes the steps that are performed to boot
        the Linux kernel. While this kind of information is not
        relevant to the system's functionality, it's interesting to
        see how the different architectures bring the system up.
      </para>
    </abstract>

  ...

  </artheader>

...

</article>
```

6.2. Marking up code, filenames, commands, and the like

If you're doing computer-related documentation or writing, chances are that you're going to have to use code samples, filenames, commands, and other computer-centric things in your document. DocBook, being designed for doing technical documentation, provides tags for marking each of these up as a specific type of element.

6.2.1. Code samples

When you want to include a multiple-line code example in your document, you should use the programlisting tag. Here's an example that's part of a level 1 section:

```
<sect1>
<para>
Here's an example:
</para>

<programlisting>
if [ $# -eq 1 ]
then
  if [ ! -r $1 ]
  then
    echo Cannot read \"$1\".  Exiting. >&2
    exit 1
  fi
fi
</programlisting>

</sect1>
```

The programlisting tag is usually processed so the output will be rendered with all whitespace and formatting preserved, done in a fixed-width font, such as courier.

If you want to include an example of SGML, XML, or HTML in your document, you have to use a special bit of markup that tells the parser to ignore any tags it contains. This special bit of markup is a CDATA marker, the beginning of which appears directly after the programlisting start tag, and which ends just before the programlisting end tag:

```
<programlisting>
<![CDATA[

<html>
<head>
<title>Web Page Title</title>
</head>

<body bgcolor="#ffffff">
Web Page Content
</body>
</html>

]]>
</programlisting>
```

Note that the beginning of the CDATA marker is "<![CDATA[", and the end of the CDATA marker is "]]>". Any markup contained between the beginning and end bits of a CDATA marker will be ignored by the processing tools and treated as if it were just part of the regular text.

6.2.2. Filenames and directories

Appropriately enough, in DocBook you markup filenames with a filename tag. You mark up directories with the same tag, only by adding class="directory" as an attribute. For example:

```
<para>
When you install DocBook Tools a number of files are added to
<filename class="directory">/usr/bin</filename>, including
```

<filename>db2html</filename>, and <filename>db2pdf</filename>.
</para>

6.2.3. Commands

When you want to mark up a command or command line, use the intuitively-named command tag. For example:

<para>
When you want to generate HTML output from your DocBook instance, use
the <command>db2html</command> command from the command line.
</para>

Often, commands include more than one part, some of which, in documentation, are actually "replaceable", meaning that they indicate something that can be replaced or modified by the user who is typing the command. For example:

<para>
When you want to generate HTML output from your DocBook instance, type
the following on the command line: <command>db2html
<replaceable>my-filename.sgml</replaceable></command>.
</para>

You can also mark up command options or flags separately as well.

<para>
If you want to generate HTML from your DocBook instance using a
different set of stylesheets, you can use the "-d" option on the command
line:
</para>

<programlisting>
<command>db2html <option>-d</option> </command>
</programlisting>

To Be Continued

7. Wrap-up & Further References

So, that's the very quick introduction to how DocBook works. This document will expand over time to include a reference of various tags and what you should use them for, etc. For now, I recommend that you take a look at some of the *.sgml files that are available through the OSWG website – with most things, the easiest way to learn DocBook is by example. You can find links to the OSWG *.sgml files here: http://www.oswg.org/oswg-nightly (http://www.oswg.org/oswg-nightly/).

I also recommend that you take the time to look at Norman Walsh's Docbook: The Definitive Guide, the online version of which is available through http://www.docbook.org.

If you have any questions or need further help, I invite you to subscribe to the OSWG docbook discussion list. Subscribe by sending mail with subscribe in the subject line to:

<oswg-docbook-request@oswg.org>

8. Other DocBook Resources

8.1. Online Resources

TBD – list of online DocBook and SGML resources

8.2. Offline Resources

TBD – list of offline DocBook and SGML resources