Installing And Using An XML/SGML DocBook Editing Suite

Setting Up A Free XML/SGML DocBook Editing Suite For Windows And Unix/Linux/BSD

by Ashley J.S Mills

Installing And Using An XML/SGML DocBook Editing Suite: Setting Up A Free XML/SGML DocBook Editing Suite For Windows And Unix/Linux/BSD

by Ashley J.S Mills

Table of Contents

Part I. Setting Up Your System To Author DocBook Documents

Chapter 1. Introduction

1. Introduction

DocBook [http://www.docbook.org/] is a DTD (Document Type Definition) defined in either SGML [http://www.w3.org/MarkUp/SGML/] (Standard Generalised Markup Language) or XML (Extensible Markup Language). HTML [http://www.w3.org/MarkUp/] (Hypertext Markup Language) is another DTD defined in SGML. The DTD specifies how certain components of the language should be interpreted for example:

```
<html>
  <head><title>HelloWorld</title></head> (1)
  <body>
    <p>
    <b>Hello World!</b> (2)
    </p>
  </body>
<html>
```

(1) HTML must start with a head element containing the title element.
(2) The text "Hello World" is encased in bold tags, which are allowed within the <p> tag.

DocBook is a DTD particularly suited to technical writing but it is not limited to it. One of the advantages of using DocBook as apposed to using some other text processing system is that DocBook can be converted into many other formats. DocBook's popularity is increasing but it has already been used in many documentation projects such as:

- The FreeBSD Documentation Project [http://www.freebsd.org/docproj/]
- The KDE Documentation Project [http://i18n.kde.org/doc/]
- The GNOME Documentation Project [http://developer.gnome.org/projects/gdp/]
- The Debian Documentation Project [http://www.debian.org/doc/ddp]
- The Darwin Documentation Project [http://www.opensource.apple.com/projects/documentation/]
- The Linux Documentation Project [http://linuxdoc.org/]
- The PHP Documentation Project [http://www.theprojects.org/phpdoc/howto/]

2. SGML vs XML

The DocBook DTD has been defined in both XML and SGML, it is possible to author DocBook documents using either format and produce identical results. SGML is the grand-daddy of all markup languages, XML is a subset of SGML with an intent on being *the* format for use on the Internet. XML attempts to fill the gap between SGML, which can be used for just about anything, and HTML which is severely limited and currently being abused because of this. The idea is that XML will be a universal way to mark up documents so that just-in-time conversion to XHTML can occur and so that information can be exchanged over the Internet in a standard format. For more about the use of XML and the future of the Internet see *XML, Java, and the future of the Web*; http://www.ibiblio.org/pub/sun-info/standards/xml/why/xmlapps.htm

There is much debate about the virtues of XML over SGML and vice-versa. This tutorial will outline both XML DocBook and SGML DocBook. It has been announced that DocBook 5.0 will be XML compliant, this does not mean that one will have to stop using SGML DocBook just that if one wants the latest support and integration with the latest technologies, one will have to make the switch sometime. I suggest that new users go straight to XML and ignore the SGML stuff.

Note

The markup of DocBook with SGML and XML is almost identical, it does not take much effort to switch to authoring from one to the other. For more information about XML and SGML see http://www.ucc.ie/xml/ and http://www.oasis-open.org/cover/general.html respectively, for a comparison see http://www.w3.org/TR/NOTE-sgml-xml.

3. Preliminary Requirements

In order to understand what is going on in this tutorial one should browse this section to get a brief idea of how the components of the DocBook system interact. An SGML/XML document starts with the DOCTYPE declaration, it looks like this:

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

This tells the SGML/XML manipulation tools the DTD in use. Where *name* is the name of the root element in the document, i.e. the tag that encompasses the rest of the document, the tags allowed by a specific DTD are specified therein. FORMALID is replaced with either PUBLIC or SYSTEM identifier or both. PUBLIC identifiers identify the DTD that the document conforms to. SYSTEM identifiers explicitly state the location of the DTD used in the document by means of a URI (Uniform Resource Indicator). PUBLIC and SYSTEM IDs are usually both present in the DOCTYPE declaration. See *Public Identifiers, System Identifiers, and Catalog Files* [http://www.docbook.org/tdg/en/html/ch02.html#s-pid-sid-catalogs]

One could write a DocBook document according to the DocBook DTD and then one could use SGML/XML tools to validate and convert the document to other formats. Generally conversion requires that we apply a stylesheet, stylesheets are written in languages such as DynaText, Panorama, SPICE, JSSS, FOSI,CSS and DSSSL. Stylesheets describe how the source document should be transformed to the respective output formats.

SGML DocBook uses DSSSL [http://www.jclark.com/dsssl/] (Document Style Semantics and Specification Language), XML DocBook uses XSL [http://www.w3.org/Style/XSL/] (Extensible Stylesheet Language) HTML uses CSS [http://www.w3.org/Style/CSS/] (Cascading Style Sheets). The stylesheet defines how certain elements should look when converted to various formats, for instance, one may wish that, when converted to HTML, each sub section of the document should be displayed in Bold 20 point Times New Roman font and that when converted to PostScript each sub section should be displayed in some other font. The stylesheet defines how the converted DocBook document will look.

For more information see the reference page and checkout the DocBook FAQ: http://www.dpawson.co.uk/docbook/

4. About this guide

The intent of this document is to provide a set of generic instructions that should enable the average computer user to set up a system for authoring SGML DocBook and/or XML documents. A key resource for getting going with DocBook is *DocBook: The Definitive guide* by Norman Walsh, http://www.docbook.org/tdg/en/html/docbook.html This is definitely a recommended read.

There are already many guides on the Internet which describe how to set up systems for authoring DocBook in XML/SGML and is is advised that one browses these as well in order to get a more comprehensive view of what is going on and how all the system components are interacting, if it is suggested that you should read something, it is probably worth while to do so, but hopefully this guide should be enough to get you started.

Other tutorials for setting up DocBook or authoring with DocBook:

- http://ourworld.compuserve.com/homepages/hoenicka_markus/ntsgml.html.

  SGML for Windows NT
  Setting up a free SGML/XML editing and publishing system on the Windows platform
  Markus Hoenicka

- Mark Galassi's http://nis-www.lanl.gov/~rosalia/mydocs/docbook-intro.html [http://nis-www.lanl.gov/~rosalia/mydocs/docbook-intro.html]

  Get Going With DocBook
  Notes for Hackers
  Mark Galassi
  Cygnus Solutions

  Copyright 1998 by Mark Galassi

- http://www.tldp.org/LDP/LDP-Author-Guide/

  LDP Author Guide
  Mark F. Komarinski
  Jorge Godoy
  David C. Merrill

- http://www.bureau-cornavin.com/opensource/crash-course/index.html

  Writing Documentation Using DocBook
  A Crash Course
  David Rugge
  Mark Galassi
  Eric Bischoff

More useful web pages can be found in the reference page.

Chapter 2. Installing And Configuring An SGML DocBook Authoring System For Windows

1. Overview Of Required Tools

- OpenJade [http://openjade.sourceforge.net/] ; for validating and converting SGML files.
- TexLive [http://www.tug.org/texlive.html] ; for the JadeTeX and PdfJadeTeX macro packs.
- ISO-Entities [http://www.oasis-open.org/cover/ISOEnts.zip] ; These are required to use the DocBook DTD properly.
- The DocBook DTD [http://www.oasis-open.org/docbook/sgml/4.1/] ; The actual definition of DocBook, used to validate SGML Docbook sources.
- The DocBook DSSSL Stylesheets and Documentation [http://sourceforge.net/projects/docbook/] ; The stylesheets, specifies how to render SGML DocBook source to various output formats.
- GhostScript [http://www.cs.wisc.edu/~ghost/doc/AFPL/get704.htm] ; Used to manipulate PostScript files.
- GhostView [ftp://mirror.cs.wisc.edu/pub/mirrors/ghost/ghostgum/gsv43w32.exe] ; Used to view PostScript files.

2. Setting Up Environment Variables

This installation requires a basic understanding of the target operating system, an understanding of how to change the operating environment etc. This is covered this in documents entitled *Configuring A Windows Working Environment* [../winenvars/winenvarshome.html] and *Configuring A Unix Working Environment* [../unixenvars/unixenvarshome.html].

> **Note**
>
> This installation uses the notation "/path/to/where/you/installed/program" to refer to the location the program was installed at, forward slashes are always used but for Windows systems these should be replaced with backslashes and prefixed with "c:".

3. Install OpenJade

Quoting the website http://www.jclark.com/jade/: “Jade is an implementation of the DSSSL style language”. Jade stands for James' DSSSL Engine. Openjade is the continuation of the work started by James Clark, the "Open" bit distinguishes it from the original Jade written by James. More information about OpenJade can be found here: http://openjade.sourceforge.net/.

OpenJade provides some of the tools required to validate and convert SGML DocBook files to other formats. The first step is to install OpenJade; download a distribution and do what ever is necessary to install the software.

3.1. Windows

Download the latest version of OpenJade from http://sourceforge.net/projects/openjade/, follow the link to OpenJade and then download the most recent distribution that contains the string "bin" this indicates the windows binaries, the file needed should look something like this:

`openjade-1_3_1-2-bin.zip`

> **Note**
>
> The version number may be different, depending on how recently updated this document is.

Unzip the archive to a suitable location.

Append `/path/to/where/you/unzipped/openjade/bin` to the `PATH` environment variable.

3.2. Unix/Linux/BSD

Get the latest *openjade* rpm for your Unix/Linux/BSD distribution from http://rpmfind.net/, install the rpm:

```
rpm -ivh xxx.rpm (for an install)
      rpm -Uvh xxx.rpm (for an upgrade)
```

Here is a page which goes into RPM in more detail: http://www.linux-mandrake.com/en/howtos/mdk-rpm/.

Alternatively, download the source from http://sourceforge.net/projects/openjade/ and build it in the manner specified in the instal-

lation instructions that come with the file. The default installation directory for openjade is (probably) `/usr/local/share/sgml`.

4. Setup `SGML_CATALOG_FILES` environment variable

Catalog files provide the means to map a PUBLIC or SYSTEM identifier in a DOCTYPE declaration to a DTD (Document Type Definition), note that one may use a catalog file to redirect a SYSTEM identifier. One may exploit and include the official PUBLIC and SYSTEM identifiers of the DOCTYPE in the declaration yet have these mapped to a local copy of the DTD, this enables one to represent the DOCTYPE consistently across multiple machines. The environment variable `SGML_CATALOG_FILES` specifies the location of the catalog file or files.

> **Note**
>
> It is not obligatory to define the location of catalog files via environment variables, **openjade** allows, by means of the `-c` option, the user to specify where the catalog files can be found at runtime. This can be useful for overriding catalog files.

An example catalog file entry is shown below:

```
PUBLIC "-//OASIS//DTD DocBook V4.1//EN" "docbook.dtd"
```

Which states that any document that has the PUBLIC ID shown above conforms to the DTD specified by the second parameter. If one does not setup `SGML_CATALOG_FILES` correctly or does not provide a catalog file to use, when using **onsgmls** (used to validate SGML files), one will get an error similar to this:

```
onsgmls:somefile.sgml:1:57:W: cannot generate system identifier for public text "-//OASIS//DTD DocBook
```

Create a new environment variable called `SGML_CATALOG_FILES` in the local environment section. Set it to:

```
./catalog;/path/to/where/you/unzipped/openjade/dsssl/catalog
```

`./catalog` ensures that the working directory is included in the catalog file list so if that one may override a catalog file locally if one needs to.

5. Install The SGML DocBook DTD

In order to use SGML DocBook one needs a copy of the SGML DocBook DTD. This can be obtained from http://www.oasis-open.org/docbook/sgml/4.1/ Download.

Download the zip archive http://www.oasis-open.org/docbook/sgml/4.1/docbk41.zip and unzip it to some suitable location. Append `/path/to/where/you/unzipped/docbk41/docbook.cat` to the `SGML_CATALOG_FILES` environment variable.

6. Install The ISO-Entities

One must install the ISO Entities: http://www.oasis-open.org/cover/ISOEnts.zip. The DocBook catalog file references these files, a normal installation seems to produce an error, this is because the format of the ISO entities is not the same as specified in the DocBook catalog file, for example docbook.cat says:

```
PUBLIC "-ISO 8879:1986//ENTITIES Diacritical Marks//EN" "iso-dia.gml"
```

But this file is actually called ISOdia. In order to fix this I moved these ISO entities and placed them in the same directory as my DocBook DTD and renamed them so they coordinated with the DocBook catalog file, this solved the error messages produced by **onsgmls**. One could alternatively change the entries in the catalog file to point to the actual files on the system.

7. Install The SGML DocBook Stylesheets

To render ones SGML DocBook sources to other file formats one must install stylesheets in order to specify how this rendering is done. Norman Walsh has produced an excellent set of stylesheets to be used with SGML DocBook, these can be obtained from http://sourceforge.net/project/showfiles.php?group_id=21935

Download the latest zip version of the file (currently `docbook-dsssl-1.76.zip`) and unzip it to some suitable directory.

8. Install TeX

8.1. Windows

TeX is a powerful typesetting system created by Donald Knuth of Stanford University. It is used to produce high-quality technical books and papers. TeX is particularly good at handling complex mathematical expressions. There is a macro package for TeX known as **JadeTeX** which provides the ability to convert TeX outputted by other formatting tools from SGML DocBook sources to PDF and PostScript.

TeXLive is an excellent all-in-one TeX package, download the installation program from here: ftp://ftp.dante.de/pub/fptex/current/TeXSetup.exe, execute the program and follow the instructions making sure to install the program in a suitable loaction. Install the *Generic recommend TeXLive scheme* but check the box that says “I want to customize the selected scheme”. Select the package `tex-htmlxml` because this includes the JadeTeX macros. Also select `tex-extrabin`,

`tex-fontbin`, `tex-fontsextra`, `tex-mathextra`, `tex-psfonts` and `tex-psutils`.

Note

During the TeXLive setup, when on the *Root of installation page*, to avoid later problems, only to change the "Root directory" field (the others will update automatically), leave the *Extra TeXMF tree* and *Home TeXMF Tree* alone. In other words, do not check the box that says "You can change the default configuration for the main directories.".

8.2. Unix/Linux/BSD

There are a number of schools of thought on which is the best TeX distribution to install. A good one is TeXLive [http://www.tug.org/texlive.html] unfortunately the Unix distribution only comes on CDROM, this is ok, as the CDROM is available free and can be downloaded as an ISO-Image to be burned. An alternative is teTeX whose homepage is http://www.tug.org/teTeX [http://www.tug.org/teTeX/], it is available to download from ftp.

The TeXLive distribution allows the installation of the JadeTex macros as an option during installation. They are in the package `tex-htmlxml`. The packages `tex-extrabin`, `tex-fontbin`, `tex-fontsextra`, `tex-mathextra`, `tex-psfonts` and `tex-psutils` should also be installed.

9. Install JadeTeX

JadeTeX is a set of TeX macros that can convert pseudo-TeX files created by **openjade** to PostScript and PDF formats.

TeXLive comes with JadeTex if it is installed as described above, if the above procedure was not adhered to or a different TeX distribution was installed and it did not come with the JadeTex macros they can be obtained from http://sourceforge.net/projects/jadetex/. Follow the installation guide at http://jadetex.sourceforge.net [http://jadetex.sourceforge.net/].

10. Install Ghostscript and GhostView

10.1. Windows

GhostScript and GhostView are used to view and manipulate PostScript files. Download GhostScript from here http://www.cs.wisc.edu/~ghost/doc/AFPL/get704.htm and GhostView from here ftp://mirror.cs.wisc.edu/pub/mirrors/ghost/ghostgum/gsv43w32.exe Follow the installation instructions, it does not matter where you install these files as we do not execute the program from the command line and we do not use its location for anything. If you want you can setup the `PATH` so that you *can* use these tools from the command line but there should be no need to.

10.2. Unix/Linux/BSD

Download GhostScript from: http://sourceforge.net/projects/ghostscript/ and GhostView from: http://www.cs.wisc.edu/~ghost/doc/AFPL/get704.htm. This resource page provides some more information: http://www.cs.wisc.edu/~ghost/. Source code and RPM installations are available.

11. Install The Custom Style-Sheets

It is useful to add a customisation layer on top of the DSSSL style-sheets so that it is easy to override options specific to a certain type of output without having to rewrite the stylesheets themselves

I have produced a custom stylesheet that provides a customisation layer for the DSSSL stylesheets. Download it here: custom.dsl [files/custom.dsl] and save it in the directory `/path/to/where/you/installed/the/stylesheets/commom/`

For usage instructions see *Chapter 4, Using the custom DSSSL style-sheets*.

Chapter 3. Installing And Configuring An XML DocBook System For Windows

1. Introduction

There are a number of different toolchains that can be used to process XML DocBook documents. For XHTML output, Saxon or xsltproc can be used (there are others like xalan which is not mentioned here). For PDF output there are about three routes, one must first produce a FO intermediate file from the XML DocBook source, Saxon or xsltproc can be used to do this. The FO output can then either be rendered to PDF via the *xmlpdftex* macros or rendered to PDF via FOP. The preferred route is:

```
(XML DocBook Source) - - - (Saxon or xsltproc) - - -> (FO) - - - (FOP) - - -> (PDF)
```

I personally prefer xsltproc but I tend to adjust my preference depending on the current state of the tools (for instance one might have support for a particular feature that the other does not), although generally this is not the case. The installation instructions for the less popular tool chains have been included to afford some redundancy to this file.

Setting up a system for XML Docbook authoring is similar in some respects to the process of setting up a system for SGML DocBook authoring, some of the steps are identical:

1. Install a TeX distribution
2. Install GhostView

This installation requires a basic understanding of the target operating system, an understanding of how to change the operating environment etc. This is covered this in documents entitled *Configuring A Windows Working Environment* [../winenvars/winenvarshome.html] and *Configuring A Unix Working Environment* [../unixenvars/unixenvarshome.html].

Note

This installation uses the notation "/path/to/where/you/installed/program" to refer to the location the program was installed at, forward slashes are always used but for Windows systems these should be replaced with backslashes and prefixed with "c:".

2. Installing the JAVA Runtime Environment

Saxon and FOP are written in Java, this is primarily because Java has a nice Unicode implementation and XML uses Unicode so it is slightly easier to program XML processing tools in Java. Java is a byte compiled language, this means that it is compiled to a sort of Java-machine-language which is then interpreted by the Java runtime environment, this is for portability reasons. At the moment there are no official machine-level compilers for the Java programming language so there are no executables of the tools written in Java that can be run natively, hence it is necessary to install the Java runtime environment in order to execute Java programs.

Download the the Java runtime environment from http://java.sun.com/j2se/downloads.html. Follow the installation instructions.

3. libxml

Within this tutorial the primary purpose for installing the libxml C library will be to gain access the tools that come with it. The tools provide the means to validate and transform XML files. In this tutorial, the program **xmllint** will be used to validate XML DocBook files before processing. The program **xsltproc** can be used to transform XML files. It is a program which uses XSLT.

3.1. Windows

To install *libxml* on a Windows machine one needs to download the Windows binaries and libraries. These can be obtained from http://www.zlatkovic.com/pub/libxml/. Download the following:

- libxml2-2.6.2+.win32.zip [http://www.zlatkovic.com/pub/libxml/libxml2-2.6.2+.win32.zip]
- libxslt-1.1.0.win32.zip [http://www.zlatkovic.com/pub/libxml/libxslt-1.1.0.win32.zip]
- iconv-1.9.1.win32.zip [http://www.zlatkovic.com/pub/libxml/iconv-1.9.1.win32.zip]

Note

The three links shown immediately above may be broken since it is common practice to remove old versions from a download page when they are obsoleted. Goto http://www.zlatkovic.com/pub/libxml/ instead and download the `libxml2...`, `libxslt...`, and `iconv...` files with the highest version numbers. Some older versions are available in the directory `oldreleases` on that server, should one desire them.

It is not necessary to extract the content of these zips entirely, instead the required functionality will be extracted. Create a suitable directory to contain the stuff that is about to be extracted. For example, on my home machine. If I am running a Windows system I have a directory called `c:\tools` which contains all the tools I install. Within `c:\tools` I have a directory called `libxml` that contains the stuff I want from these zips. Create a suitable directory to extract the desired content from the zips into.

Extract the following files from the `libxml` archive into the directory.

- `libxml2.dll`
- `xmllint.exe`

Extract the following files from the `libxslt` archive into the directory.

- `libexslt.dll`
- `libxslt.dll`
- `xsltproc.exe`

Extract the following files from the `iconv` archive into the directory.

- `iconv.dll`
- `iconv.exe`

Append `\directory\you\just\unzipped\everything\to` to the `PATH` environment variable.

You might not use *all* the tools but they are worth having around in case you decide you need them.

3.2. Linux/Unix/BSD

These files are probably already installed on your system, as most modern distributions of these operating systems use XML processing for some of the more popular components. But you may wish to get the latest versions, in which case, goto ftp://xmlsoft.org/ and get the latest *libxml2* and *libxslt*. There are gzipped tars and RPMs available, download whichever you prefer. A list of the latest files at the time of writing is shown below:

```
libxml2-2.4.25.tar.gz
libxml2-2.4.25-1.i386.rpm
libxml2-2.4.25-1.src.rpm
libxslt-1.0.21.tar.gz
libxslt-1.0.21-1.i386.rpm
libxslt-1.0.21-1.src.rpm
```

The ftp directory also contains *devel* versions of the software, this is for people who want to develop with libxml.

4. Install Saxon

1. Download the latest distribution of Saxon from http://saxon.sourceforge.net/ and unzip the zip to a suitable location.
2. Append `/directory/you/unzipped/saxon/saxon.jar` to the `CLASSPATH` environment variable. If integration with FOP is desired, append `/directory/you/unzipped/saxon/saxon-fop.jar` to the `CLASSPATH` environment variable. If integration with JDOM is desired, append `/path/to/where/you/unzipped/saxon/saxon-jdom.jar` to the `CLASSPATH` environment variable.

4.1. Setup Catalog Resolver

A catalog resolver provides a way to use catalog files with Java tools like Saxon, the details of this can be found in the documentation that comes with the installation download.

1. Download the XML Entity and URI Resolvers package from http://wwws.sun.com/software/xml/developers/resolver/. You have to register to download.
2. Unzip the package to some suitable location, add the file `resolver.jar` to your `CLASSPATH` environment variable.
3. Append the directory where the resolver tool was unzipped to the `CLASSPATH` environment variable.

The Resolver is now ready to use with Saxon but by default it uses the AElfred XML parser that comes with Saxon. This has a few problems so it is best to invoke Saxon and specify that xerces should be used instead. Note that this requires that xerces be installed, usually this consists of having the file `xerces.jar` in the `CLASSPATH`. It is likely that if other tools such as FOP have been installed that this is already the case, if it is not, download and install xerces from http://xml.apache.org/xerces-j/.

One invokes Saxon and specifies that xerces should be used as the XML parser like this:

```
java -Djavax.xml.parsers.DocumentBuilderFactory=org.apache.xerces.jaxp.DocumentBuilderFactory
     -Djavax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl
     -Djavax.xml.transform.TransformerFactory=com.icl.saxon.TransformerFactoryImpl
     com.icl.saxon.StyleSheet
```

This must all be input on one line, this is where Ant (http://jakarta.apache.org/ant/) comes in very useful. The resolver uses a file called `CatalogManager.properties` to specify various options, this must be present somewhere on the `CLASSPATH`. By default this file resides in the root directory of the resolver zip extraction, hence the adding of this directory to the `CLASSPATH` in step three of the installtion above. Details on setting up `CatalogManager.properties` can be found in the documentation that comes in the downloaded archive. A simple example is shown below:

```
#CatalogManager.properties
verbosity=1

# Always use semicolons in this list
catalogs=/lib/docbook-xml-4.2/catalog

# Prefer PUBLIC identifiers over SYSTEM identifiers
prefer=public
```

This sets up the resolver to use the catalog file specified.

5. FOP

FOP(Formatting Objects Processor) is used to transform FO files to files of other formats. In this tutorial it is used to transform FO output produced by **xsltproc** into PDF which is a well known format considered by many to be aesthetically pleasing. The Unix and Windows installation paths are very similar, the differences will be mentioned where appropriate.

Download the latest version of the Fop application, from http://ftp.plig.org/pub/apache/dist/xml/fop/. Download the zip or tar with *bin* as a substring of its name to some suitable location.

On Windows, append `/directory/where/you/unzipped/fop/fop.bat` to the `PATH` environment variable.

On Unix, append `/directory/where/you/unzipped/fop/fop.sh` to the `PATH` environment variable.

5.1. Install Jimi

Jimi is needed if you want to use PNG images with FOP, download it from http://java.sun.com/products/jimi/# and open the archive.

On Windows, rename `JimiProClasses.zip` from the archive to `jimi-1.0.jar` and place it in the `/directory/where/you/unzipped/fop/lib` directory.

On Unix, rename `JimiProClasses.zip` from the archive to `JimiProClasses.jar`and place it in the `/directory/where/you/unzipped/fop/lib` directory.

6. DocBook DTD

The DocBook DTD(Document Type Definition) contains rules which specify the structure of a valid DocBook document, for example, the order that elements may appear and valid attributes etc. If one has a document which one claims is written in DocBook, it is not written in DocBook unless it conforms to the DocBook DTD. The DTD is used by tools like **xsltproc** in transforming DocBook documents.

DTD's are especially useful when one wants to validate a document to check that it conforms to the DTD one claims it conforms to. Validation is beneficial because a valid document is less likely to break processing tools (if a valid document does break a processing tool it is likely that the processing tool is broken and not the document). Hence, the DocBook DTD can be used to validate that a purported DocBook document *really is* a DocBook document.

The latest version, at the time of writing, is called *DocBook XML 4.2*, it is distributed from http://www.docbook.org/xml/4.2/index.html.

Download the zipped archive, http://www.docbook.org/xml/4.2/docbook-xml-4.2.zip and unzip it to some suitable location. For example if I was running the Windows operating system I would unzip it to a directory like `c:\lib\docbook-xml-4.2`, this is the same as the name of the zip file. If I was running a Unix, Linux, or BSD operating system I would unzip it to a directory like `/lib/docbook/docbook-xml-4.2`. You might have noticed on the webpage or in the zip, other files apart from DTD files, these are auxiliary files and are necessary.

6.1. Catalog Files

When one writes a DocBook document in XML one inserts a DocType declaration at the top of the document to specify that the document is written in DocBook. This declaration specifies the version of DocBook being used with a *PUBLIC* ID in the declaration, a *SYSTEM* ID in the header specifies where one can find the DTD for DocBook. In the case of DocBook, usually this *SYSTEM* ID points to some location on the OASIS(Organization for the Advancement of Structured Information Standards) website because this is where DocBook's official home is. Some tools used for processing DocBook use the DTD at this location, this is no good when one wants to process a DocBook document on a computer that does not have Internet access or where accessing the Internet is undesirable.

To overcome the necessity to access the Internet to process DocBook documents one can use a *catalog* file. A *catalog* file maps *PUBLIC* or *SYSTEM* IDs to alternate locations. Taking the example from above, one would process the DocBook document that contained the *SYSTEM* ID pointing to the Internet with a tool and specify a catalog file to use. The catalog file would map the *SYSTEM* ID pointing to the Internet to a copy of the DTD on the local system, thus circumventing the need for any Internet access.

The DocBook zip that was just downloaded does actually contain it's own catalog file (`catalog.xml`. This does not seem to provide the desired functionality without modification. Instead of modifying that catalog file, create a new one called `catalog` in the `docbook-xml-4.2` directory. Put the following content into it:

```
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
  <group xml:base="." prefer="public" >
  <public publicId="-//OASIS//DTD DocBook XML V4.2//EN" uri="file:///c:/lib/docbook-xml-4.2/docbookx.d
  </group>
</catalog>
```

This maps the *PUBLIC* ID for DocBook XML V4.2 to a local copy of it's DTD. The example above was taken from a Windows system, modify the value of the *uri* attribute to point to the location of the DTD on your system. On a Unix system this could be `file:///lib/docbook/docbook-xml-4.2/docbookx.dtd`.

The processing tools must know where this catalog file is in order to use the functionality it provides. This is achieved via an environment variable called `XML_CATALOG_FILES`, create this environment variable and make it point to the catalog file you just created. You could add similar entries to the catalog file shown above to map other DTDs you desire to use to local copies of their DTD's.

7. XSL StyleSheets

XSL stylesheets dictate how a document written in XML should be transformed using XSLT to a particular output format. In the case of DocBook, Norman Walsh has already written, and regularly maintains some stylesheets for DocBook that provide rules for transformations from an XML DocBook document to the most commonly desirable output formats such as XHTML and PDF. The installation for Unix and Windows machines is the same.

Download the latest stylesheets from http://sourceforge.net/project/showfiles.php?group_id=21935 and unzip the zip or gzipped tar to some suitable location. If I was running a Windows system I would use `c:\lib\docbook-xsl\`, if I was using a Unix system I would use `c:\lib\docbook\docbook-xsl`. The stylesheets are now ready to use.

7.1. Custom StyleSheets

The output produced by the stylesheets mentioned above is reasonable but the stylesheets mentioned above are a standard distribution and as a consequence seem to be designed to cater for the needs of the many, which is sensible, unfortunately. One may modify the stylesheets directly but more often one creates a customisation layer which imports the standard stylesheets and then one overrides specific aspects of the standard stylesheets or adds extra functionality within the customisation layer according to ones tastes. I have created a customisation layer which looks good enough for standard applications and am offering it to download.

This is particularly pertinent if you study at The University Of Birmingham because any documentation created by me there in DocBook uses this customisation layer, all the tutorials I have written conform to these stylesheets. If you your documents to have the same style as the tutorials then use this customisation layer. It is probably worth downloading the customisation layer anyway so you can see how one goes about creating a customisation layer. Here is the zipped customisation layer: custom-stylesheets.zip [files/custom-stylesheets.zip].

Unzip the zip to where you want the customisation layer to be situated, this could be within the stylesheets directory or in separate directory. If you unzip it to the stylesheets directory the customisation layer will unzip into the directories `common`, `fo` and `xhtml`. If you unzip to a separate directory these directories will be created.

Wherever you unzip the zip, it is important to change the references of the imports in the files so that they reflect the state of your system, the files `fo/customfo.xsl`, `xhtml/customxhtml.xsl` and `xhtml/customchunk.xsl` all have references that may need to be modified. For example, the file `fo/customfo.xsl` has the import line:

```
<!- - Import standard fo style-sheet - ->
<xsl:import href="file:///c:/lib/docbook-xsl/fo/docbook.xsl"/>
```

Change this to point to `/where/you/put/the/stylesheets/fo/docbook.xsl`

Similarly, change the entry in `customchunk.xsl` to point to `/where/you/put/the/stylesheets/xhtml/chunk.xsl` and the entry in `customxhtml.xsl` to point to `/where/you/put/the/stylesheets/xhtml/docbook.xsl`. The advantage of unzipping the zip in the same location as the standard stylesheets is that the import links may be relative (the import links can al-

ways be relative assuming the stylesheets are on the same machine, but for clarity if I am using a different directory for the customisation stylesheets I will make the import references absolute).

I have only provided customisations for FO and XHTML. It will become apparent how to use the customisation layer in the section on using the tools later. The provided customisations are listed below:

- `fo/customfo.xsl` - Use this to generate custom FO
- `xhtml/customxhtml.xsl` - Use this to generate custom XHTML (segmented)
- `xhtml/customxchunk.xsl` - Use this to generate custom XHTML (chunked)

More information about customising stylesheets can be found at http://www.sagehill.net/xml/docbookxsl/.

Chapter 4. Using The SGML Tools

1. About this chapter

This chapter provides concrete examples of how to use the tools associated with the processing of SGML DocBook source files.
Conventions used throughout this chapter:

- `in.sgml` is used to represent the input SGML DocBook source file.
- `out.xxx` is used to represent the output of some operation, where xxx is some extension.
- In most of the examples, command sequences are used that are non operating system specific. The notation `/path/to/.../` maybe used as a prefix to the location of certain files, the real directory location should substitute the generic terms and the file separators should be replaced with those used on the target operating system.

Note

When one uses the tools to produce PDF output, if callout or admonition graphics were used, one should first copy the `images` directory into the build directory. If one does not do this, the PDF tools will not be able to find the images and hence problems will occur. The production of HTML output formats does not require this as the images are not actually required to build the output but are referenced from outputted HTML pages, hence the referencing of images is controlled via the output tools and embedded into the HTML page. The location of the callout and admonitions graphics can be set in the stylesheets or via command line parameters.

2. Validating an SGML DocBook Document with onsgmls

It is useful to validate an SGML DocBook document against the DocBook DTD before attempting to process the SGML DocBook source because if the document conforms to the DTD there should be no errors during processing. If there are errors during processing and the document is valid, one can infer that the problem is very unlikely to lie with the document but somewhere else such as the processing tool. **onsgmls** is used to validate SGML DocBook documents.

`onsgmls -s in.sgml`

If no errors occur, **onsgmls** will exit silently (indicative of the `-s` option). If there are errors messages will be output describing where in the input file each error occurred and a short description of the error. An example **onsgmls** error message is shown below:

```
onsgmls:sgmlsys-chapter3.sgml:73:9:E: end tag for "PROGRAMLISTING" omitted, but its
declaration does not permit this.
```

onsgmls error messages take the following form:

```
program-name : input-file : line-number : column-number : error-type : error-description
```

3. Using OpenJade to convert SGML DocBook source to an intermediate TeX format

`openjade -t tex -d /path/to/stylesheets/print/docbook.dsl in.sgml`

This will create the file `in.tex`. The stylesheet to be used for the operation is specified using the `-d` option. In this case, Norman Walsh's stylesheet for print output was chosen. The output type was specified using the `-t` option. It is possible to redirect the output to a file of your choice (by default**openjade** will use the same name as `in.sgml` and create `in.tex`. Redirection is performed in the usual manner that redirection is performed on the target operating system.

4. Using OpenJade to convert SGML DocBook source to (single file)HTML

`openjade -t> sgml -V nochunks -d /path/to/dsssl/html/docbook.dsl in.sgml > out.html`

Notice that the output from the **openjade** was redirected to a desired output file, if this is not done **openjade** outputs to the standard output stream.

5. Using OpenJade to convert SGML DocBook source to (Chunked) HTML

openjade -t sgml -d /path/to/stylesheets/html/docbook.dsl in.sgml

This produces the output file `t1.html`, as well as a number of other auxiliary files. The type of the output is specified as SGML because HTML is defined in SGML. Norman Walsh's HTML output stylesheet was used.y. The HTML produced is segmented, a new HTML page is generated for each each section and an `index.html` file is generated as the entry point into the document.

6. Using openjade to convert SGML DocBook source to RTF

openjade -t -d /path/to/stylesheets/print/docbook.dsl in.sgml

7. Using jadetex to convert (openjade created) TeX to DVI

The TeX files produced by **openjade** cannot be processed using the standard TeX processing tools, the JadeTeX macro pack must be used.

Unix/Linux/BSD users should probably use the following command sequence:

jadetex in.tex

This should work fine. Sometimes one can get errors associated with memory usage when using this command, a recent installation of jadetex should not have this problem. If the **jadetex** command does not exist try:

hugelatex "&jadetex" in.tex

Windows users should use the following command sequence:

jadetex in.tex

Unix users may encounter memory usage errors when using the above commands, some TeX distributions come with the command **hugelatex** which uses larger memory parameters. If **hugelatex** is not installed or the memory parameters are too small, TeX may generate an error like this:

```
! TeX capacity exceeded, sorry [pool size=21903].
```

To fix this, one can alter the memory pool size in the file `texmf.cnf` (situated in the `web2c` directory of the TeX installation). Here is a `texmf.cnf` file from a working installation of TeX and jadetex:

```
main_memory.latex =                1500000
param_size.latex =                 1500
stack_size.latex =                 1500
hash_extra.latex =                 15000
string_vacancies.latex =           45000
pool_free.latex =                  47500
nest_size.latex =                  500
save_size.latex =                  5000
pool_size.latex =                  1250000
max_strings.latex =                55000
font_mem_size.latex=               400000

main_memory.jadetex =              1500000
param_size.jadetex =               1500
stack_size.jadetex =               1500
hash_extra.jadetex =               15000
string_vacancies.jadetex =         45000
pool_free.jadetex =                47500
nest_size.jadetex =                500
save_size.jadetex =                5000
pool_size.jadetex =                1250000
max_strings.jadetex =              55000
font_mem_size.jadetex=             400000


main_memory.pdfjadetex =           2500000
param_size.pdfjadetex =            1500
stack_size.pdfjadetex =            1500
hash_extra.pdfjadetex =            50000
string_vacancies.pdfjadetex =      55000
pool_free.pdfjadetex =             47500
nest_size.pdfjadetex =             500
save_size.pdfjadetex =             5000
pool_size.pdfjadetex =             1250000
max_strings.pdfjadetex =           55000
```

This is how the file came upon installation, the memory allocated is relatively huge, indicating that any modern TeX installation should not have any memory problems. Increase the values in the above file that are indicated as being too small by the error messages produced if necessary.

The command sequence will produce `in.dvi`, `hello.aux` and `hello.log`. It is recommended that this command sequence be executed at least 3 times, this is so that the `in.aux` is correctly updated with page numbers.

8. Using JadeTeX to convert (openjade created)TeX to PDF

Unix users should try using the following command sequence:

```
pdfjadetex in.tex
```

If the above command did not exist, try:

```
%hugelatex "&pdfjadetex" in.tex
```

Windows users should use the following command sequence:

```
pdfjadetex in.tex
```

It is recommended that this command sequence be executed at least 3 times, this is so that the `in.aux` is correctly updated with page numbers. This produces the files `in.aux` and `in.pdf`.

If memory usage error messages were output see the secion on memory problems. Alternatively one can use the command **ps2pdf** which comes with GhostScript to convert files from PostScript to PDF:

```
ps2pdf in.ps
```

9. Using dvipsto convert DVI to PS

```
dvips in.dvi -o out.ps
```

This will produce the PostScript file `out.ps`.

10. Using osx to convert SGML DocBook source to an XML tree

```
osx in.sgml > out.xml
```

This will produce the output file `in.xml`, redirection can be used to redirect the output to a specific file.

There is the possibility that the conversion could generate errors because XML does not allow the use of certain SGML idiosyncrasies. This may be due to use of characters that XML does not allow unless referenced via their Unicode number.

In XML you would have to use a *data entity* instead, where a character is referenced numerically ({) or hexadecimally (&#x12AB). See http://www.w3.org/TR/xml-infoset/#infoitem.character

11. Using the custom DSSSL stylesheets

To use the custom stylesheets one specifies the custom stylesheet as the stylesheet that **openjade** should use. More than one output format is included in the single custom stylesheet provided in the installation section, the `-i` option is used to determine which of the output fromats should be used. The generation of HTML output is shown below:

```
openjade -i output.html -t sgml -d /path/to/stylesheets/common/custom.dsl in.sgml
```

HTML output was specified using the `-i` option with the value *output.html*. The custom stylesheet was specified using the `-d`. The generation of intermediate TeX format is shown below:

```
openjade -i output.print -t tex -d /path/to/docbookdsssl/common/custom.dsl in.sgml
```

HTML was specified using the `-i` with the *output.print* value. The custom stylesheet was specified using the `-d`. RTF can be produced by specifying the RTF backend with the `-t` option.

It is worth taking a look at the custom stylesheets to see how a customisation layer can be created.

Chapter 5. Using the XML Tools

1. About this chapter

This chapter provides concrete examples of how to use the tools associated with the processing of SGML DocBook source files.
Conventions used throughout this chapter:

- `in.sgml` is used to represent the input SGML DocBook source file.
- `out.xxx` is used to represent the output of some operation, where xxx is some extension.
- In most of the examples, command sequences are used that are non operating system specific. The notation `/path/to/.../` maybe used as a prefix to the location of certain files, the real directory location should substitute the generic terms and the file separators should be replaced with those used on the target operating system.

Note

When one uses the tools to produce PDF output, if callout or admonition graphics were used, one should first copy the `images` directory into the build directory. If one does not do this, the PDF tools will not be able to find the images and hence problems will occur. The production of HTML output formats does not require this as the images are not actually required to build the output but are referenced from outputted HTML pages, hence the referencing of images is controlled via the output tools and embedded into the HTML page. The location of the callout and admonitions graphics can be set in the stylesheets or via command line parameters.

2. Using xmllint to validate an XML DocBook document

In order to check the syntactic accordance of a DocBook document with the DocBook DTD one may use **xmllint**.

`xmllint - -valid - -noout in.xml`

The `- -valid` option specifies that **xmllint** should validate the document against the DTD and the `- -noout` option specifies that no output should be produced if there are no errors, hence if the document being validated is valid, **xmllint** will exit silently. If the document is invalid **xmllint** will output an error similar to this:

```
docbook.xml:1: error: Start tag expected, '< not found
?xml version="1.0" encoding='ISO-8859-15'?>
^
```

Which specifies that there is a missing start tag on line one.

Note

The `- -loaddtd` option can be used to specify an external DTD to validate the input document with.

3. Using xsltproc to generate (Single file)XHTML output from an XML Docbook document

`xsltproc file:///path/to/docbook-xsl/xhtml/docbook.xsl in.xml > out.html`

This will produce a single XHTML file according to the XSL stylesheet specifications.

4. Using xsltproc to generate XHTML(Segmented) output from an XML Docbook document

`xsltproc file:///path/to/docbook-xsl/xhtml/chunk.xsl in.xml`

This will produce a set of XHTML files of the document where each section is on a separate HTML page. The layout will accord to the XSL stylesheet specified. The separate files will be given unique names that correspond to the different sections of the book, e.g `index.html`, `ar01s02.html` and `ar01s03.html`.

5. Using xsltproc to generate FO output from an XML Docbook document

`xsltproc file:///path/to/docbook-xsl/fo/docbook.xsl in.xml > out.fo`

This will produce output as an XSL FO(Formatting object), this is an intermediate file type that can be used by other programs to generate other types of output, such as PDF.

6. Using xsltproc to generate HTMLhelp output from XML Docbook source

```
xsltproc file:///path/to/docbook-xsl/htmlhelp/htmlhelp.xsl in.xml
```

This will produce a set of HTML files of the document where each section is on a separate HTML page. This will also generate `htmlhelp.hhp` and `toc.hhc`.

7. Using xsltproc to generate Javahelp output from XML Docbook source

```
xsltproc file:///path/to/docbook-xsl/javahelp/javahelp.xsl in.xml
```

This will produce a set of HTML files of the document where each section is on a separate HTML page. This will also generate the JavaHelp helpset file `jhelpset.hs`, the JavaHelp table of contents file `jhelptoc.xml`, the JavaHelp map file `jhelpmap.jhm` and the JavaHelp index file `jhelpidx.xml`.

8. Using Saxon to generate (Single file) HTML output from XML DocBook source

```
java com.icl.saxon.StyleSheet -o out.html in.xml file:///path/to/docbook-xsl/html/docbook.xsl
```

This will produce a single HTML file output according to XSL stylesheet specified.

9. Using Saxon to generate (Single file) XHTML output from XML Doc-Book source

```
java com.icl.saxon.StyleSheet -o out.html in.xml file:///path/to/docbook-xsl/xhtml/docbook.xsl
```

This will produce a single XHTML file output according to XSL stylesheet specified.

10. Using Saxon to generate FO output from XML Docbook source

```
java com.icl.saxon.StyleSheet -o in.xml file:///path/to/docbook-xsl/fo/docbook.xsl
```

This will produce output as an XSL FO(Formatting object), this is an intermediate file type that can be used by other programs to generate other types of output, such as PDF.

11. Using FOP to generate PDF output from XSL FO input

In order to execute this conversion you will need to have generated XSL FO output by using **xsltproc** or some other tool capable of doing so.

```
fop.bat in.fo out.pdf
```

Substituting `fop.sh` for `fop.bat` on Unix derivatives. This will generate a PDF file named according to the name provided as the second argument. FOP will probably generate lots of warnings about un-implemented features whilst generating this output, this is normal and can be ignored.

12. Using pdfxmltex to generate PDF output from XSL FO input

```
pdfxmltex in.fo
```

It is recommended that this command sequence be executed at least 3 times that `in.aux` is correctly updated with page numbers. This will produce the files `in.aux` and `in.pdf`.

13. General Usage

Assume that a file called `test.xml` has been created in XML DocBook. Assume that the stylesheets are located in a directory called `/lib/docbook-xsl/` One would create XHTML output like this:

```
xsltproc file:///lib/docbook-xsl/xhtml/docbook.xsl test.xml > test.html
```

One would create PDF output in two steps, first create the FO output using **xsltproc**:

```
xsltproc file:///lib/docbook-xsl/fo/docbook.xsl test.xml > test.fo
```

Next, process the FO output with FOP to produce the PDF file:

```
fop.bat in.fo out.pdf
```

If you want to use the custom stylesheets you simply modify the stylesheet parameter so that it points to the custom stylesheet you want to use. Assuming an install of the customisation layer mentioned above in the same location as the standard stylesheets one could generate XHTML output that conformed to the custom stylesheet for XHTML like this:

```
xsltproc file:///lib/docbook-xsl/xhtml/customxhtml.xsl test.xml > test.html
```

Similarly, FO output could be produced.

Chapter 6. Using The SGML Tools To Process XML DocBook Documents

1. About This Chapter

It is possible to use the SGML tools to process XML DocBook documents, this can be useful if the XML tools are currently broken or it is preferable to use the SGML tools, for example, **onsgmls** is considered to produce more meaningful error messages than **xmllint**. The rest of this chapter provides some concrete examples.

Conventions used throughout this chapter:

- `in.sgml` is used to represent the input SGML DocBook source file.
- `out.xxx` is used to represent the output of some operation, where xxx is some extension.
- In most of the examples, command sequences are used that are non operating system specific. The notation `/path/to/.../` maybe used as a prefix to the location of certain files, the real directory location should substitute the generic terms and the file separators should be replaced with those used on the target operating system.

When one uses the tools to produce PDF output, if callout or admonition graphics were used, one should first copy the `images` directory into the build directory. If one does not do this, the PDF tools will not be able to find the images and hence problems will occur. The production of HTML output formats does not require this as the images are not actually required to build the output but are referenced from outputted HTML pages, hence the referencing of images is controlled via the output tools and embedded into the HTML page. The location of the callout and admonitions graphics can be set in the stylesheets or via command line parameters.

Note

When one uses SGML tools to process XML documents there may be slight differences in the way that filenames are specified, for example, **xmllint**, and **xsltproc** implements a true URI scheme, more about which can be found at http://www.ietf.org/rfc/rfc2396.txt, an example of a URI is shown below:.

```
file:///path/to/wherever/
```

Most of the SGML tools do not employ this scheme and it is only necessary to specify the filename like this:

```
/path/to/wherever/
```

To process XML files the SGML tools reference a file called `xml.dcl` which is the XML declaration, this is bundled with the OpenJade suite in the `pubtext` directory, it may be a good idea to put this somewhere more convenient.

2. Using onsgsmls to validate XML DocBook documents

```
onsgmls -s /path/to/openjade-1.3.1/pubtext/xml.dcl test.xml
```

If the document is valid the program will exit silently, otherwise errors will be produced. See error descriptions

3. Using openjade to convert XML DocBook to (segmented) HTML

```
openjade -t xml  -d  /path/to/dssslstylesheets/html/docbook.dsl /path/to/openjadedir/pubtext/xml.dcl in
```

The fact an XML file was being processed was specified using the `-t` option and Norman Walsh's DSSSL HTML stylesheet was specified using the `-d`. After this, the XML declaration was passed to **openjade** so that openjade knew how to deal with the XML document it was about to receive which was passed in as the last parameter. This produces the HTML output in chunked format.

4. Using openjade to convert XML DocBook to (single file) HTML

```
openjade -V nochunks -t xml  -d  /path/to/dssslstylesheets/html/docbook.dsl /path/to/openjadedir/pubtex
```

Single file HTML output was selected using the `-V` nochunks option. The fact an XML file was being processed was specified using the `-t` option and Norman Walsh's DSSSL HTML stylesheet was specified using the `-d`. After this, the XML declaration was passed to **openjade** so that openjade knew how to deal with the XML document it was about to receive which was passed in as the last parameter. This produces the HTML output in single file format.

5. Using openjade to convert from XML DocBook to intermediate TeX format

```
openjade -t tex  -d  /path/to/dssslstylesheets/print/docbook.dsl /path/to/openjadedir/pubtext/xml.dcl
```

The TeX input file type was specified using the `-t` option. Norman Walsh's DSSSL print stylesheet was specified using the `-d` op-

tion. After this, the XML declaration was passed to **openjade** so that openjade knew how to deal with the XML document it was about to receive which was passed in as the parameter after. The output `.tex` will have to be processed with special TeX macros in order to produce other output formats.

6. Using openjade to convert from XML DocBook to RTF

```
openjade -t rtf  -d  /path/to/dssslstylesheets/print/docbook.dsl /path/to/openjadedir/pubtext/xml.dcl
```

The RTF input type was specified using the `-t` option and Norman Walsh's DSSSL print stylesheet was specified using the `-d` option. After this, the XML declaration was passed to **openjade** so that openjade knew how to deal with the XML document it was about to receive which was passed in as the parameter after.

7. Using jadetex to convert from (openjade created) intermediate TeX format to DVI

The TeX files produced by **openjade** cannot be processed using the standard TeX commands, instead, the JadeTeX macro pack is used.

Unix users should try using the following command sequence:

```
jadetex in.tex
```

If the above command did not exist, try:

```
hugelatex "&jadetex" in.tex
```

Windows users should use the following command sequence:

```
jadetex in.tex
```

It is recommended that this command sequence be executed at least 3 times, this is so that the `in.aux` is correctly updated with page numbers. This produces the files `in.dvi`, `hello.aux` and `hello.log`.

If memory usage error messages were output see the secion on memory problems.

8. Using JadeTeX to convert (openjade created)TeX to PDF

Unix users should try using the following command sequence:

```
pdfjadetex in.tex
```

If the above command did not exist, try:

```
%hugelatex "&pdfjadetex" in.tex
```

Windows users should use the following command sequence:

```
pdfjadetex in.tex
```

It is recommended that this command sequence be executed at least 3 times, this is so that the `in.aux` is correctly updated with page numbers. This produces the files `in.aux` and `in.pdf`.

If memory usage error messages were output see the secion on memory problems. Alternatively one can use the command **ps2pdf** which comes with GhostScript to convert files from PostScript to PDF:

```
ps2pdf in.ps
```

9. Using dvips to convert from DVI to PS

```
dvips in.dvi -o out.ps
```

This produces the PostScript file `out.ps`.

Part II. Introduction To Authoring DocBook Documents

Chapter 7. Creating An SGML DocBook Document

1. Declaring the document type

When marking up documents in a DTD it is standard practice to include a DOCTYPE declaration so that the processing tools 'know' what DTD the document being processed conforms to. A typical DOCTYPE declaration for a document marked up in DocBook SGML can be seen below:

```
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook V4.1//EN">
```

Notice that the markup is enclosed within:

```
<! ... >
```

These are SGML directives so are enclosed within the special tags *<!* and >. The distinguishing feature is that an exclamation mark follows the '<' which opens the directive section. The SGML directives define various aspects of the SGML document itself. The DOCTYPE declaration as a whole is known as the prologue.

Immediately following the opening of the prologue is the keyword *DOCTYPE* then a word such as *chapter* or *book*, this specifies the root element of the document and is exactly the name of the tag that will be used to open and close the markup after the prologue. Example:

```
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook V4.1//EN">
<chapter><title>The Theory Of Special Relativity</title>
  <para>
    The principle of the physical relativity of all uniform motion...
  </para>
</chapter>
```

Notice that the start and end tags of the document are defined by the root element specified in the DOCTYPE declaration in the prologue of the document.

What follows is the keyword *PUBLIC* this specifies that what follows that is a public identifier, which is described in more detail in the next section, but essentially defines what type of document this is.

2. The use of PUBLIC identifiers in SGML

A public identifier is an indirect way of specifying what DTD a document was created under. (The markup language that was used to write the document). A particular document type is associated with a particular DTD which specifies the particular details of how this type of document should be structured. Hence processing tools can know how to parse and validate your document by examining this DTD.

The processing tools can use the public identifier, along with catalog file which maps the public identifier to a DTD. An example from a catalog file called `docbook.cat` is shown below:

```
 - - DocBook driver file ..................................................... - -

PUBLIC "-//OASIS//DTD DocBook V4.1//EN" "docbook.dtd"
```

The whole of the OpenJade catalog file found in the `dsssl` subdirectory can be seen below:

```
PUBLIC "-//James Clark//DTD DSSSL Flow Object Tree//EN" "fot.dtd"
PUBLIC "ISO/IEC 10179:1996//DTD DSSSL Architecture//EN" "dsssl.dtd"
PUBLIC "-//James Clark//DTD DSSSL Style Sheet//EN" "style-sheet.dtd"
PUBLIC "-//OpenJade//DTD DSSSL Style Sheet//EN" "style-sheet.dtd"
SYSTEM "builtins.dsl" "builtins.dsl"
```

Notice that each public identifier is mapped to a SYSTEM identifier which specifies the location of the actual DTD for the DOCTYPE, in these cases, they happen to be in the same directory as the catalog file but they do not have to. The locations of the files here are not defined by formal URIs because it is not necessary.

A PUBLIC identifier is essentially of the format

```
      prefix//owner-identifier//text-class text-description//language//display version
```

For more information see *Public Identifiers, System Identifiers, and Catalog Files*
http://www.docbook.org/tdg/en/html/ch02.html#s-pid-sid-catalogs.

3. The use of SYSTEM identifiers in SGML

A SYSTEM identifier specifies the exact location of the DTD that the document conforms to. SYSTEM identifiers may follow PUBLIC identifiers as in the example below:

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.1//EN" "docbook.dtd">
```

Or they may occur on their own:

```
<!DOCTYPE book SYSTEM "customdocbook.dtd">
```

The above example specifies that the document conforms to the DTD defined in the file "customdocbook.dtd". More information about SGML declarations can be found here: http://www.oasis-open.org/cover/wlw11.html.

4. An SGML DocBook article template

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V4.1//EN">
<article>
  <articleinfo>
    <title>Your title here</title>

    <author>
      <firstname>Your first name</firstname>
      <surname>Your surname</surname>
      <affiliation>
        <address><email>Your e-mail address</email></address>
      </affiliation>
    </author>

    <copyright>
      <year>1998</year>
      <holder role="mailto:your e-mail address">Your name</holder>
    </copyright>

    <pubdate role="rcs">$Date: 2003/01/08 10:27:39 $</pubdate>

    <releaseinfo>$Id: DocBookSys-Chapter7-SGML-Creating.xml,v 1.1.1.1 2003/01/08 10:27:39 ug55axm Exp $

    <abstract>
      <para>Include an abstract of the article&amp;apos;s contents here.</para>
    </abstract>
  </articleinfo>

  <sect1><title>Section 1</title>
    <para>
      My first article!
    </para>
  </sect1>

  <sect2><title>Section 2</title>
    <para>
      Brilliant!
    </para>
  </sect2>
</article>
```

This is an article template and illustrates some of the features of docbook. The file can be validated using **onsgmls** like this:

onsgmls -s articletemplate.sgml

This should produce no errors and exit silently because the document is structured correctly. Chunked HTML output can be generated from this file like this:

openjade -t sgml -d /path/to/docbook/dbdsssl/html/docbook.dsl articletemplate.sgml

5. An SGML DocBook book template

DocBook books are generally for projects over 25 pages long, they allow the inclusion of chapters and parts, a template for a book is shown below:

```
<!DOCTYPE book PUBLIC '-//OASIS//DTD DocBook V4.1//EN'>
<book>
  <bookinfo>
    <title>Your title here</title>

    <author>
      <firstname>Your first name</firstname>
      <surname>Your surname</surname>
```

```
      <affiliation>
        <address><email>Your e-mail address</email></address>
      </affiliation>
    </author>

    <copyright>
      <year>1998</year>
      <holder role='mailto:your e-mail address'>Your name</holder>
    </copyright>

    <pubdate role='rcs'>$Date: 2003/01/08 10:27:39 $</pubdate>

    <releaseinfo>$Id: DocBookSys-Chapter7-SGML-Creating.xml,v 1.1.1.1 2003/01/08 10:27:39 ug55axm Exp $

    <abstract>
      <para>Include an abstract of the book&amp;apos;s contents here.</para>
    </abstract>
  </bookinfo>

  <part><title>Part1</title>
    <chapter><title>Part 1, Chapter 1</title>
      <sect1><title>Part1, Chapter 1, Section1</title>
        <para>
          Hello there!
        <para>
      </sect1>
    </chapter>

    <chapter><title>Part 1, Chapter 2</title>
      <sect1><title>Part1, Chapter 2, Section 1</title>
        <para>
          Hi there!
        <para>
      </sect1>
    </chapter>
  </part>

  <part><title>Part2</title>
    <chapter><title>Part 2, Chapter 1</title>
      <sect1><title>Part 2, Chapter 1, Section1</title>
        <para>
          GoodDay there!
        <para>
      </sect1>

      <sect1><title>Part2, Chapter1, Section2</title>
        <para>
          GoodNight there!
        <para>
      </sect1>
    </chapter>
  </part>
</book>
```

This file can be validated using **onsgmls** like this:

```
onsgmls -s booktemplate.sgml
```

This should produce no errors and exit silently because the document is structured correctly. RTF output can be generated like this:

```
openjade -t rtf -d /path/to/docbook/dbdsssl/print/docbook.dsl booktemplate.sgml
```

There is a DocBook element quick reference card available from http://www.dpawson.co.uk/docbook/reference.html#d3e60.

Chapter 8. Creating an XML DocBook document

1. Declaring The Document Type

XML documents must declare that they are XML documents like this:

```
<?xml version="1.0" encoding="UTF-8"?>
```

When marking up documents in a DTD it is standard practice to include a DOCTYPE declaration so that the processing tools 'know' what DTD the document being processed conforms to. A typical DOCTYPE declaration for a document marked up in DocBook XML can be seen below:

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

Notice that the markup is enclosed within:

```
<! ... >
```

These are XML directives so are enclosed within the special tags *<!* and >. The distinguishing feature is that an exclamation mark follows the '<' symbol which opens the directive section. The XML directives define various aspects of the XML document.

Immediately following the opening of the prologue is the keyword *DOCTYPE* then a word such as *chapter* or *book*, this specifies the root element of the document and is exactly the name of the tag that will be used to open and close the markup of the document which appears after the prologue. An example is shown below:

```
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
"http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<chapter><title>The Theory Of Special Relativity</title>
  <para>
    The principle of the physical relativity of all uniform motion...
  </para>
</chapter>
```

Notice that the start and end tags of the document are defined by the root element specified in the DOCTYPE declaration in the prologue of the document.

What follows is the keyword *PUBLIC*, this specifies that what follows that is a public identifier, which defines what type of document the document is.

2. The use of PUBLIC and SYSTEM identifiers in XML

A public identifier is an indirect way of specifying what DTD a document was created under. (The markup language that was used to write the document). A particular document type is associated with a particular DTD which specifies the particular details of how this type of document should be structured. Hence processing tools can know how to parse and validate your document by examining this DTD.

The processing tools can use the public identifier, along with catalog file which maps the public identifier to a DTD. An example catalog file is shown below:

```
<catalog xmlns='urn:oasis:names:tc:entity:xmlns:xml:catalog'>
  <group xml:base='/usr/share/xml/' prefer='public' >
       <public
         publicId='-//OASIS//DTD DocBook XML V4.2//EN'
         uri='file:///path/to/docbook/docbook-V4.2/docbookx.dtd'/>
  </group>
</catalog>
```

This catalog file maps the PUBLIC ID ";-//OASIS//DTD DocBook XML V4.2//EN", which might be present in the prologue of a DocBook document, to the file "file:///path/to/docbook/docbook-V4.2/docbookx.dtd".

XML requires that if one uses a PUBLIC ID one also provides a SYSTEM ID. Usually the official URL of the DTD on the Internet is used as the SYSTEM identifier in the actual document, following the PUBLIC identifier. If the processing tools fail to find a mapping from the PUBLIC identifier to a SYSTEM identifier in the catalog file(s) they will fall back to using the SYSTEM identifier specified in the document.

A PUBLIC identifier is essentially of the format

```
prefix//owner-identifier//text-class text-description//language//display version
```

For more information see *Public Identifiers, System Identifiers, and Catalog Files*
http://www.docbook.org/tdg/en/html/ch02.html#s-pid-sid-catalogs
[http://www.docbook.org/tdg/en/html/ch02.html#s-pid-sid-catalogs].

A SYSTEM identifier specifies the exact location of the DTD that the document conforms to. SYSTEM identifiers may follow PUBLIC identifiers as in the example below:

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.1//EN" "docbook.dtd">
```

Or they may occur on their own:

```
<!DOCTYPE book SYSTEM "customdocbook.dtd">
```

The above example specifies that the document conforms to the DTD defined in the file "customdocbook.dtd".

3. Creating an XML DocBook document

For the ultimate reference guide see *DocBook: The Definitive Guide* [http://www.docbook.org/tdg/en/html/docbook.html]. A template for a DocBook *article* is shown below:

> **Note**
>
> The output illustrated in this section was produced using a customisation of the stylesheets hence output on systems not implementing the same customisations may differ.

3.1. An *article* template

```
<?xml version="1.0" encoding='UTF-8'?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
  "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<article>
  <articleinfo>
    <title>Your title here</title>

    <author>
      <firstname>Your first name</firstname>
      <surname>Your surname</surname>
      <affiliation>
        <address><email>Your e-mail address</email></address>
      </affiliation>
    </author>

    <copyright>
      <year>2002</year>
      <holder role="mailto:your e-mail address">Your name</holder>
    </copyright>

    <abstract>
      <para>Include an abstract of the article's contents here.</para>
    </abstract>
  </articleinfo>

  <sect1><title>Section 1</title>
    <para>
      blah blah blah
    </para>
  </sect1>

  <sect1><title>Section 2</title>
    <para>
      blah blah blah
    </para>
  </sect1>
</article>
```

3.2. An example XML DocBook Book

DocBook books are generally for projects over 25 pages long, they allow the inclusion of chapters and parts, a book template is shown below:

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.2//EN"
"http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<book>
  <bookinfo>
    <title>Your title here</title>

    <author>
      <firstname>Your first name</firstname>
```

```
      <surname>Your surname</surname>
      <affiliation>
        <address><email>Your e-mail address</email></address>
      </affiliation>
    </author>

    <copyright>
      <year>1998</year>
      <holder role="mailto:your e-mail address">Your name</holder>
    </copyright>

    <pubdate role="rcs">$Date: 2003/01/08 10:27:39 $</pubdate>

    <releaseinfo>$Id: DocBookSys-Chapter8-XML-Creating.xml,v 1.1.1.1 2003/01/08 10:27:39 ug55axm Exp $

    <abstract>
      <para>Include an abstract of the book&apos;s contents here.</para>
    </abstract>
  </bookinfo>

  <part><title>Part1</title>
    <chapter><title>Part 1, Chapter 1</title>
      <sect1><title>Part1, Chapter 1, Section1</title>
        <para>
          Hello there!
        <para>
      </sect1>
    </chapter>

    <chapter><title>Part 1, Chapter 2</title>
      <sect1><title>Part1, Chapter 2, Section 1</title>
        <para>
          Hi there!
        <para>
      </sect1>
    </chapter>
  </part>

  <part><title>Part2</title>
    <chapter><title>Part 2, Chapter 1</title>
      <sect1><title>Part 2, Chapter 1, Section1</title>
        <para>
          GoodDay there!
        <para>
      </sect1>

      <sect1><title>Part2, Chapter1, Section2</title>
        <para>
          GoodNight there!
        <para>
      </sect1>
    </chapter>
  </part>
</book>
```

This can be validated using **xmllint** like this:

```
xmllint - -valid - -nooutarticletemplatewin.xml
```

This should produce no errors and exit silently because the document is structured correctly. Chunked XHTML output can be generated like this:

```
xsltproc file:///path/to/docbook-xsl/xhtml/chunk.dsl articletemplate.xml
```

There is a DocBook element quick reference card available from http://www.dpawson.co.uk/docbook/reference.html#d3e60.

Chapter 9. DocBook Document Examples

1. Common DocBook Elements

1.1. *<para>*

The reference page for the *para* element can be found here: http://www.docbook.org/tdg/en/html/para.html. *para* is one of the most commonly used elements of all the DocBook elements. *para*'s can contain block elements such as *itemizedlist* and *mediaobject* and can contain almost all inline elements. There is some debate about whether or not it is best to separate block elements from *para* elements, it is probably better to do so however because some processing systems have problems processing block elements within *para* elements. An example of a para element containing some inline elements is shown below:

```
<para>
  <quote>Behold the superfluous. They are always sick. They vomit their gall and call it a ne
  - Friedrich Wilhelm Nietzsche, <citetitle>Twilight of the Idols</citetitle>
</para>
```

Looks like this:

“Behold the superfluous. They are always sick. They vomit their gall and call it a newspaper.” - Friedrich Wilhelm Nietzsche, *Twilight of the Idols*

1.2. *<programlisting>*

The reference page for the *programlisting* element can be found here: http://www.docbook.org/tdg/en/html/programlisting.html. The *programlisting* element is used to display information that should be output verbatim, that is, white space is significant. An example is shown below:

```
  <programlisting>
public class HelloWorld {
   public static void main(String args[]) {
      System.out.println(&amp;quot;Hello World!&amp;quot;);
   }
}
  </programlisting>
```

Is output as:

```
public class HelloWorld {
   public static void main(String args[]) {
      System.out.println("Hello World!");
   }
}
```

Notice the use of (") to represent the (") character, this is know as a character entity and is used to represent a character that is not allowed to be used directly in the document, this is because these characters are used by the XML part of the document for special purposes. These special characters are known as *CDATA* as apposed to *PCDATA*, the latter standing for *Parsed Character DATA*. If one wants to use lots of *CDATA* characters in a document then one can wrap the section in a CDATA section like this:

```
  <programlisting>
    <![CDATA[
      One can get away with using lots of &&& """ ''' <<< >>>
      characters that would normally require being marked up as entities.
    ]]>
  </programlisting>
```

Is displayed as:

```
        One can get away with using lots of &&& """ ''' <<< >>> characters that would normally have to
```

For more information about the available entities see the next section.

1.3. Entities for special characters

The following entities are provided for special characters, they must always be used unless they are used in a section that has been marked as a CDATA section. It is preferred to always use them in preference of CDATA sections however:

Character	Entity
<	&lt;

Character	Entity
>	&gt;
&	&amp;
"	&quot;
'	&apos;

1.4. *<screen>*

The reference page for the *screen* element can be found here: http://www.docbook.org/tdg/en/html/screen.html. Often one wants to illustrate the use of a program or a commandline, the *screen* element is intended to mark content up as text that a user would see on a computer screen. An example is shown below:

```
<screen>
  <userinput><command>java</command> org.apache.fop.apps.Fop <replaceable>in.fo</replaceable>
</screen>
```

Is displayed as:

java org.apache.fop.apps.Fop *in.fo out.pdf*

1.5. *<ulink>*

The reference page for the *ulink* element can be found here: http://www.docbook.org/tdg/en/html/ulink.html. *ulink* is the DocBook equivalent of HTML's "blah blah", an example is shown below:

```
<para>
  <ulink url="http://www.oasis-open.org/committees/docbook/">http://www.oasis-open.org/commit
</para>
```

Displays as:

http://www.oasis-open.org/committees/docbook/

1.6. Lists

1.6.1. *<itemizedlist>*

The reference page for *itemizedlist* is here: http://www.docbook.org/tdg/en/html/itemizedlist.html. Itemized lists are standard bulleted lists and should be used where order of evaluation of the items of the list is not significant, ordered lists should be used where order of evaluation fot he items of the list are significant. An example use of itemized list is shown below:

```
<itemizedlist>
  <listitem><para>Books</para>
    <itemizedlist>
      <listitem><para>Donald E. Knuth - The Art Of Computer Programming</para></listitem>
      <listitem><para>Nils J. Nilsson - Artificial Intelligence: A New Synthesis</para></lis
      <listitem><para>Pure Mathematics 2 - Geoff Mannall, Michael Kenwood</para></listitem>
    </itemizedlist>
  </listitem>

  <listitem><para>Games</para>
    <itemizedlist>
      <listitem><para>Chess</para></listitem>
      <listitem><para>Backgammon</para></listitem>
      <listitem><para>Noughs And Crosses</para></listitem>
    </itemizedlist>
  </listitem>
</itemizedlist>
```

Which looks like this:

- Books
  - Donald E. Knuth - The Art Of Computer Programming
  - Nils J. Nilsson - Artificial Intelligence: A New Synthesis
  - Pure Mathematics 2 - Geoff Mannall, Michael Kenwood

- Games
  - Chess
  - Backgammon
  - Noughs And Crosses

1.6.2. *<orderedlist>*

The reference page for *orderedlist* is here: http://www.docbook.org/tdg/en/html/orderedlist.html. Ordered lists are used to specify a sequence of steps of which the order of evaluation is significant. The general form of an ordered list is like this:

```
<orderedlist>
  <listitem><para>Action A<para></listitem>
  <listitem><para>Action B<para></listitem>
</orderedlist>
```

Which would look like this:

1. Action A
2. Action B

One may also specify the type of enumeration that the list will display, there are five types of enumeration; arabic, loweralpha, lowerroman, upperalpha , upperroman. The type of enumeration is specified via the *numeration* attribute like this:

```
<orderedlist numeration="arabic">
  <listitem>...</listitem>
                    .
                    .
                    .
</orderedlist>
```

The types of enumeration are shown below:
Arabic:

1. arabic
2. arabic
3. arabic

Loweralpha:

a. loweralpha

b. loweralpha

c. loweralpha

Lowerroman:

i. lowerroman

ii. lowerroman

iii. lowerroman

Upperalpha:

A. upperalpha

B. upperalpha

C. upperalpha

Upperroman:

I. upperroman

II. upperroman

III. upperroman

These can be combined to make nested enumeration clearer:

```
<orderedlist numeration="loweralpha">
  <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>
  <listitem>
    <para>Cleanup</para>
    <orderedlist numeration="upperalpha">
      <listitem><para>Throw away scraps</para></listitem>
      <listitem><para>Clean side</para></listitem>
      <listitem><para>Wash hands</para></listitem>
    </orderedlist>
  </listitem>
</orderedlist>
```

Which looks like this:

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

III. Cleanup

   A. Throw away scraps

   B. Clean side

   C. Wash hands

One may also make the enumeration continue at lower nested levels by setting the *continuation* attribute to *continues*:

```
<orderedlist>
  <listitem>
    <para>Do this</para>
    <orderedlist numeration="arabic">
      <listitem><para>And this</para></listitem>
      <listitem><para>And this</para></listitem>
      <listitem><para>And this</para></listitem>
    </orderedlist>
  </listitem>
  <listitem>
    <para>And this</para>
    <orderedlist numeration="arabic" continuation="continues">
      <listitem><para>And this</para></listitem>
      <listitem><para>And this</para></listitem>
      <listitem><para>And this</para></listitem>
    </orderedlist>
  </listitem>
</orderedlist>
```

Which looks like this:

1. Do this

    1. And this
    2. And this
    3. And this

2. And this

    4. And this
    5. And this
    6. And this

Note

Some stylesheets may define that nested lists are of a different numeration by default.

1.7. Some common inline elements

Some common inline elements and their output are shown below:

Example	**Displays as**
<emphasis>Emphasised Text</emphasis>	*Emphasised Text*
<emphasis role="strong">A different type of /emphasis<emphasis>	*A different type of emphasis*
<filename>blahblah.txt</filename>	`blahblah.txt`
<acronym>XML</acronym>	XML
<quote>blahblahblah<quote>	“blahblahblah”

2. Including Images

Images are included in DocBook documents as illustrated below:

```
<figure><title>blah</title>
  <mediaobject>
    <imageobject><imagedata fileref="blah.jpg" format="JPEG"/></imageobject>
    <textobject><phrase>Image description</phrase></textobject>
  </mediaobject>
```

```
</figure>
```

The overall encapsulating element is *figure* the reference page for which can be found at http://www.docbook.org/tdg/en/html/figure.html. The *figure* contains a *mediaobject* element which can occur on it's own too and may contain *audioobject*, *caption*, *imageobject*, *objectinfo*, *textobject* and *videoobject* elements. The reference page *mediaobject* is at http://www.docbook.org/tdg/en/html/mediaobject.html.

imageobject is the type of *mediaobject* used to include an image and it's reference page can be found at http://www.docbook.org/tdg/en/html/imageobject.html. The item within the *imageobject* that handles the image is *imagedata*, it's reference page is at http://www.docbook.org/tdg/en/html/imagedata.html.

The idea behind *mediaobject* is to provide a way to include media in many formats. It becomes the document processors job to decide which of the formats specified in the *mediaobject* to use in the particular output medium chosen. For example the *mediaobject* element may contain a PNG format *imageobject* for HTML output and a TIFF format *imageobject* for print output, there may also be a *textobject* providing a description of the image for an output format that does not have the capability to display images, for example, perhaps the document will be output in an audio format for people with sight problems.

One does not have to encapsulate the *mediaobject* in a *figure* object but doing so allows one to provide a title and be able to have the *figure* listed in a list of figures at the beginning of the document.

imagedata may be of the following formats:

- BMP
- CGM-BINARY
- CGM-CHAR
- CGM-CLEAR
- DITROFF
- DVI
- EPS
- EQN
- FAX
- GIF
- GIF87a
- GIF89a
- IGES
- JPEG
- JPG
- linespecific
- PCX
- PIC
- PNG
- PS
- SGML
- SVG
- TBL
- TEX
- TIFF
- WMF
- WPG

The attribute *format* is thus required along with either *fileref* or *entityref* to reference the image:

```
<mediaobject>
  <imageobject><imagedata fileref="frog.png" format="PNG"/></imageobject>
  <textobject><phrase>A frog</phrase></textobject>
</mediaobject>
```

One could use stylesheets such that, in HTML rendered output, the *phrase* used in the *textobject* would become the alternative text in an image in the HTML. One can use multiple *imageobject*s for different output formats, for instance one may have an *eps* version of the image so that output can be generated with a processing chain that requires the image to be in this form. One could include different image formats for each of the desired output formats.

The *imagedata* element has the useful attributes *align* and *valign*. *align* specifies how the image should be aligned horizontally and can be set to the values; *center*, *left* and *right*. *valign* specifies how the image should be aligned vertically and can be set to the values; *bottom*, *middle* and *top*.

3. Tables

There are two elements used for placing tables inside a DocBook document, *table* and *informaltable*, the only difference between the former and the latter is that the former requires a *title* and the latter does not.

```
<table><title>title</title>          <informaltable>
  .                                        .
  .                            or          .
  .                                        .
```

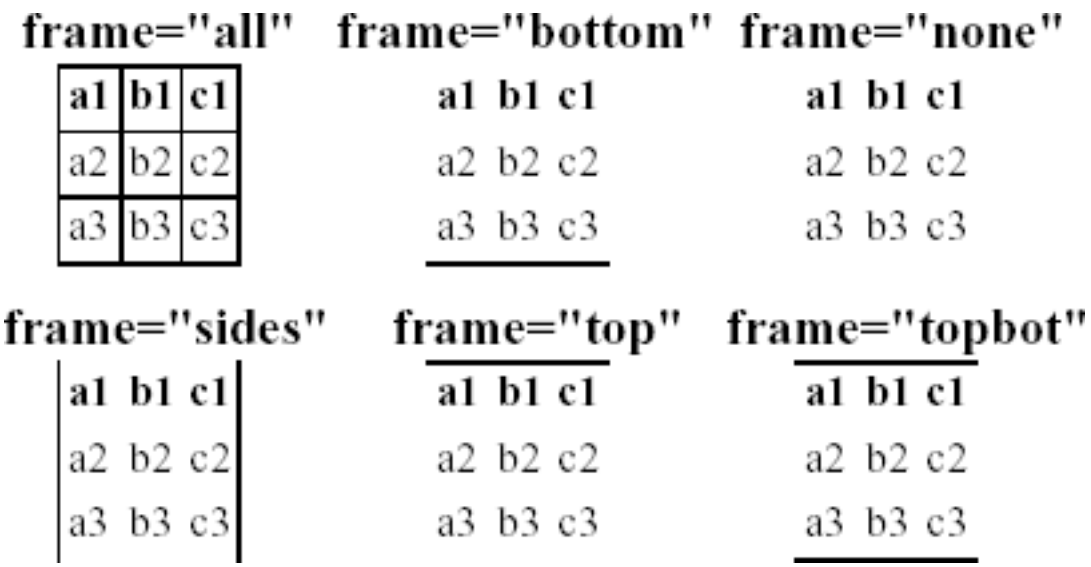
```
</table>                                    </informaltable>
```

The *table* contains an attribute called *frame* which specifies how the table should be framed:

```
<table frame="frametype"><title>frame="frametype"</title>
  <tgroup cols="1">
    <thead>
      <row><entry>a1</entry><entry>b1</entry><entry>c1</entry></row>
    </thead>
    <tbody>
      <row><entry>a2</entry><entry>b2</entry><entry>c2</entry></row>
      <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
  </tgroup>
</table>
```

Where *frametype* is replaced with one of *all*, *bottom*, *none*, *sides*, *top* or *topbot*:

Figure 9.1. Table frame types



The output above is PDF, with HTML all the tables look the same as the one with attribute *all* apart from the one with attribute *none* which has no frame at all. The attributes *colsep* and *rowsep* are used to control whether lines should be drawn between columns and rows respectively:

```
<table colsep="0" rowsep="0"> ... </table>
<table colsep="0" rowsep="1"> ... </table>
<table colsep="1" rowsep="0"> ... </table>
<table colsep="1" rowsep="1"> ... </table>
```

Unfortunately at the time of writing the tools used to convert FO to PDF either did not yet implement this feature or were in a broken state with regards to this feature so no pictorial examples can be provided. Other *table* attributes are discussed at http://www.docbook.org/tdg/en/html/table.html.

The generic layout for a table is as follows:

```
<table><title>title</title>
  <tgroup cols="3">
    <thead>
      <row><entry>blah</entry></row>
    </thead>

    <tbody>
      <row><entry>blah</entry></row>
    </tbody>

    <tfoot>
      <row><entry>blah</entry></row>
    </tfoot>
  </tgroup>
</table>
```

tgroup contains the rest of the table which must contain a *tbody* element which specifies which data is in the body of the table. The *tbody* element may be empty with the table being included in *thead* or *tfoot* but this is not the intention. The reason for the *thead*

and *tfoot* elements is so that different layouts can be applied by the stylesheets for the header and the footer of the table respectively. So usually the first row would be wrapped in a *thead* element. *tgroup* has the mandatory attribute *cols* which specifies the number of columns the table has.

tgroup may also specify alignment of content via the *align* attribute, where *alignment* is either *left*, *center* or *right*:

```
<table frame="all"><title>align="alignment"</title>
  <tgroup cols="3" align="alignment">
    <tbody>
      <row><entry>a2</entry><entry>b2</entry><entry>c2</entry></row>
    </tbody>
  </tgroup>
</table>
```

Figure 9.2. Table alignment types

Table 12. align="left"

a1	b1	c1

Table 13. align="center"

a1	b1	c1

Table 14. align="right"

a1	b1	c1

For more information about the *tgroup* element see http://www.docbook.org/tdg/en/html/tgroup.html.

A *row* consists of a number of *entry* elements which are entered in the sequence they should appear in each table row, for more information about the *row* element see http://www.docbook.org/tdg/en/html/row.html.

The *entry* element has some interesting attributes which allow an entry to span more than one column or row, they are (*namest* & *nameend*) and *morerows* respectively. The *morerow* attribute specifies how many more rows the entry it is applied to should span:

```
<table frame="all"><title><emphasis>morerows</emphasis> example</title>
  <tgroup cols="3">
    <tbody>
      <row><entry morerows="2">a1</entry><entry>b1</entry><entry>c1</entry></row>
      <row><entry>b2</entry><entry>c2</entry></row>
      <row><entry>b3</entry><entry>c3</entry></row>
    </tbody>
  </tgroup>
</table>
```

Figure 9.3. *morerows* example

Table 15. *morerows* example

a1	b1	c1
	b2	c2
	b3	c3

Unfortunately there is no *morecolumns* attribute, instead one has to use *namest* to specify the starting column of the *entry* and *nameend* to specify the ending column of the *entry*. The value applied to this attribute is the name of the columns, columns are named using the *colspec* element, *colspec* elements are inserted inside *tgroup* but before *thead*, *tbody* and *tfoot*:

```
<table frame="all"><title>column spanning</title>
  <tgroup cols="3">
    <colspec colname="col1"/>
    <colspec colname="col2"/>
    <colspec colname="col3"/>
    <tbody>
      <row><entry namest="col1" nameend="col3">a1</entry></row>
      <row><entry>a2</entry><entry>b1</entry><entry>c1</entry></row>
      <row><entry>a3</entry><entry>b2</entry><entry>c2</entry></row>
    </tbody>
  </tgroup>
</table>
```

Figure 9.4. Column spanning example

Table 16. column spanning

a1		
a2	b1	c1
a3	b2	c2

More information about the *entry* element can be found at http://www.docbook.org/tdg/en/html/entry.html. Tables may be nested to a level of one, see http://www.docbook.org/tdg/en/html/entrytbl.html. For the entire source and output pertaining to the examples discussed in this section see Table Examples [files/tableexampleshome.html].

Chapter 10. Auxiliary Tools for DocBook Authoring

1. Editors

One of the most important parts of any DocBook authoring suite is the editing environment since this is where the author spend most of the time in the document production process. It is possible to author DocBook documents in a plain text editor but this does not give the user the benefits as syntax highlighting, auto-indentation and spell-checking. Therefore it is imperative to find an editor that is comfortable and maximises productivity.

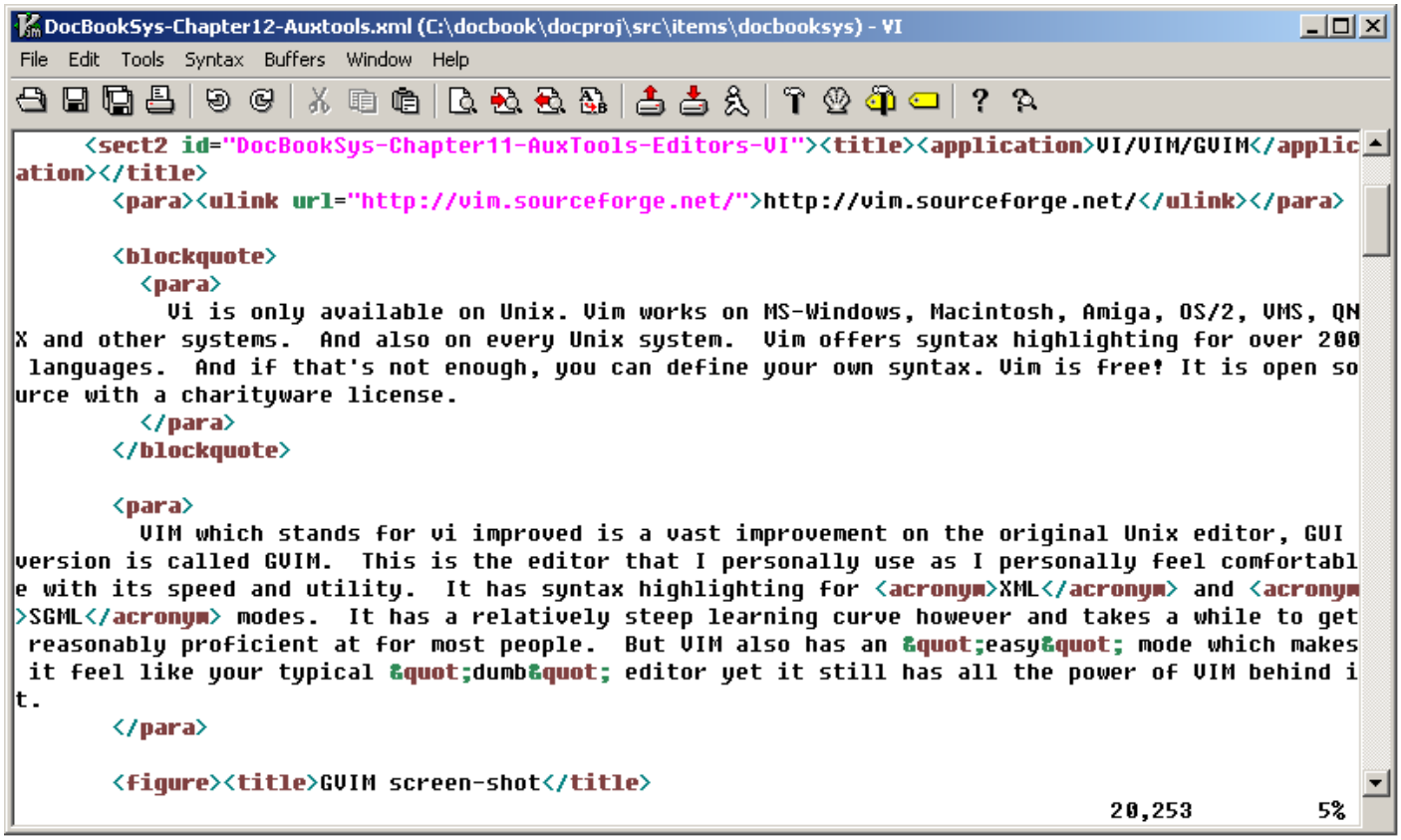
1.1. VI/VIM/GVIM

http://vim.sourceforge.net/

> Vi is only available on Unix. Vim works on MS-Windows, Macintosh, Amiga, OS/2, VMS, QNX and other systems. And also on every Unix system. Vim offers syntax highlighting for over 200 languages. And if that's not enough, you can define your own syntax. Vim is free! It is open source with a charityware license.

VIM which stands for vi improved is a vast improvement on the original Unix editor, GUI version is called GVIM. This is the editor that I personally use as I personally feel comfortable with its speed and utility. It has syntax highlighting for XML and SGML modes. It has a relatively steep learning curve however and takes a while to get reasonably proficient at for most people. But VIM also has an "easy" mode which makes it feel like your typical "dumb" editor yet it still has all the power of VIM behind it.

Figure 10.1. GVIM screen-shot

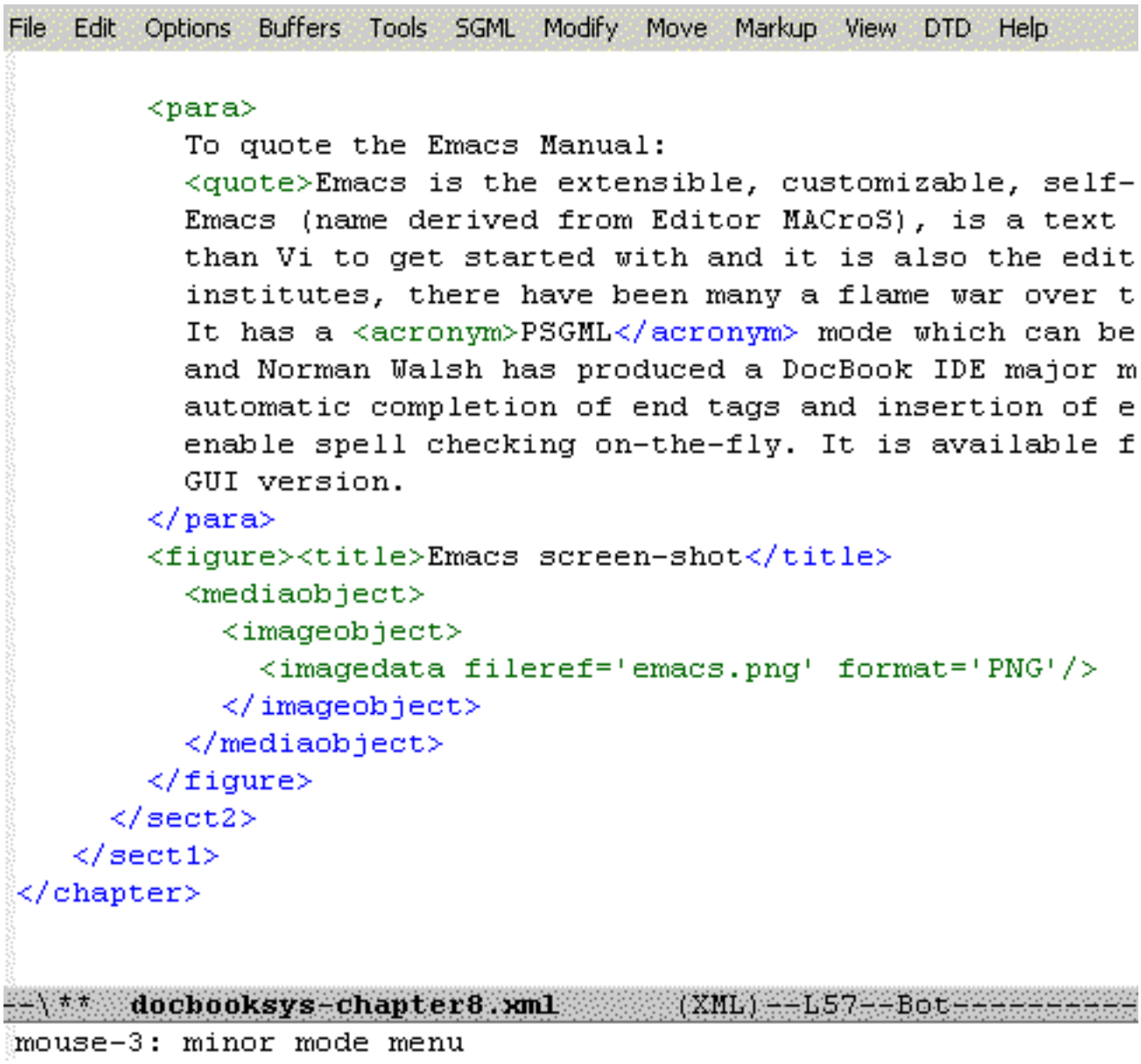


1.2. Emacs

http://www.gnu.org/software/emacs/emacs.html

To quote the Emacs Manual: “Emacs is the extensible, customizable, self-documenting real-time display editor.” Emacs (name derived from Editor MACroS), is a text editor with an emacs lisp core. It is easier than Vi to get started with and it is also the editor of choice for many Universities and educational institutes, there have been many a flame war over the issue of VI vs EMACS. It has a PSGML mode which can be setup to author DocBook with syntax highlighting and Norman Walsh has produced a DocBook IDE major mode which incorporates extensive features such as automatic completion of end tags and insertion of elements. Emacs has an ispell/Aspell plug-in to enable spell checking on-the-fly. It is available for just about any operating system. GUI capable versions are available. Emacs is released under the GPL (its free).

Figure 10.2. Emacs screen-shot



1.3. ThotBook

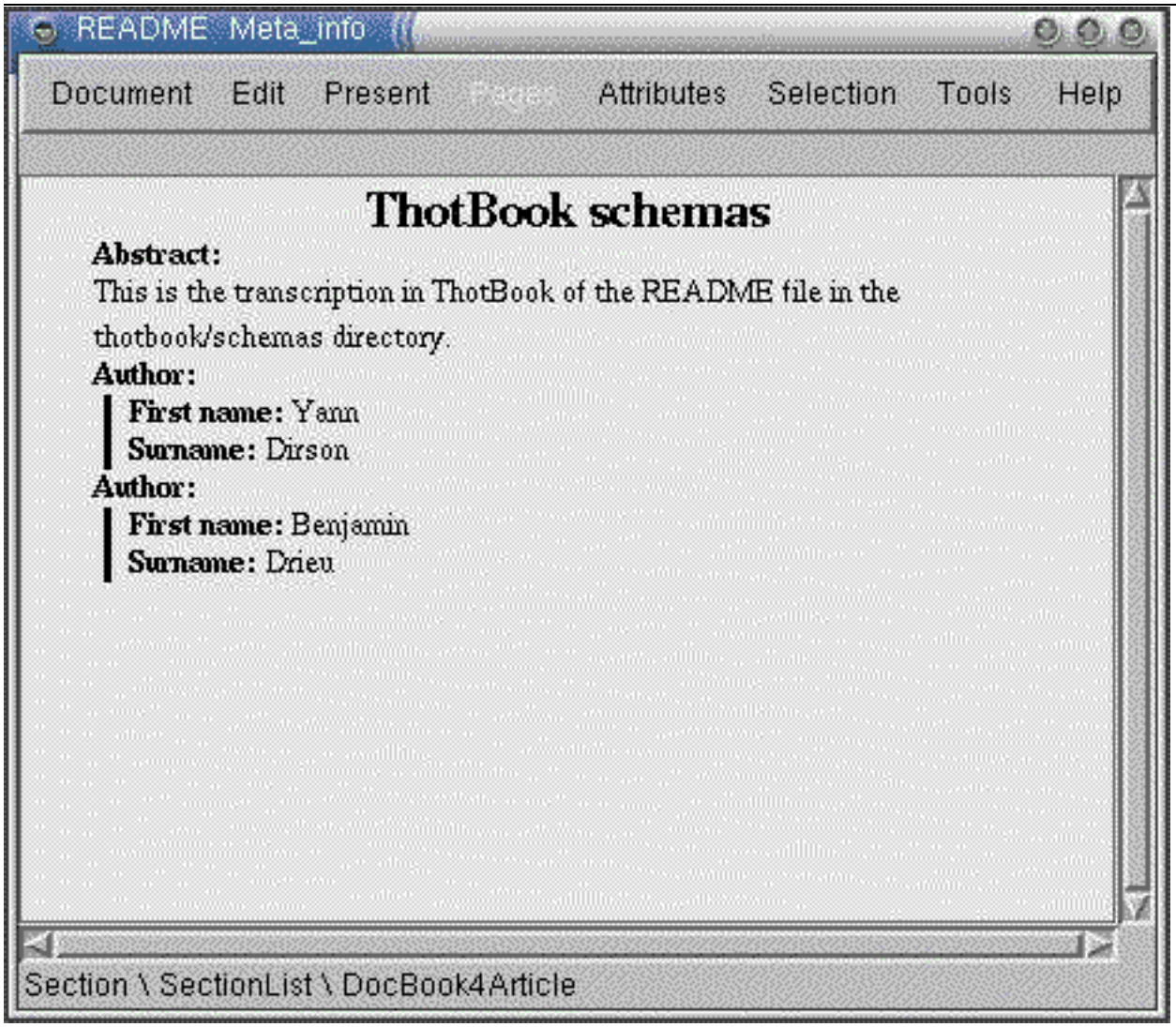
http://www.freesoftware.fsf.org/thotbook/

To quote the FAQ:

> ThotBook is (will be) a visual editor for DocBook, based on ThotLib. ThotBook will allow users to visually edit DocBook documents using a graphical interface and will allow them to import/export their work into structured DocBook SGML/XML. ThotBook will help to produce valid structured DocBook documents without having to learn the DocBook syntax. ThotBook will respect document structure, as opposed to "WYSIWYG" editors, which produce (mostly) unstructured documents.

It is only available via CVS at the moment but should be one to watch for users who like visual interfaces.

Figure 10.3. ThotBook screen-shot



1.4. EpcEdit

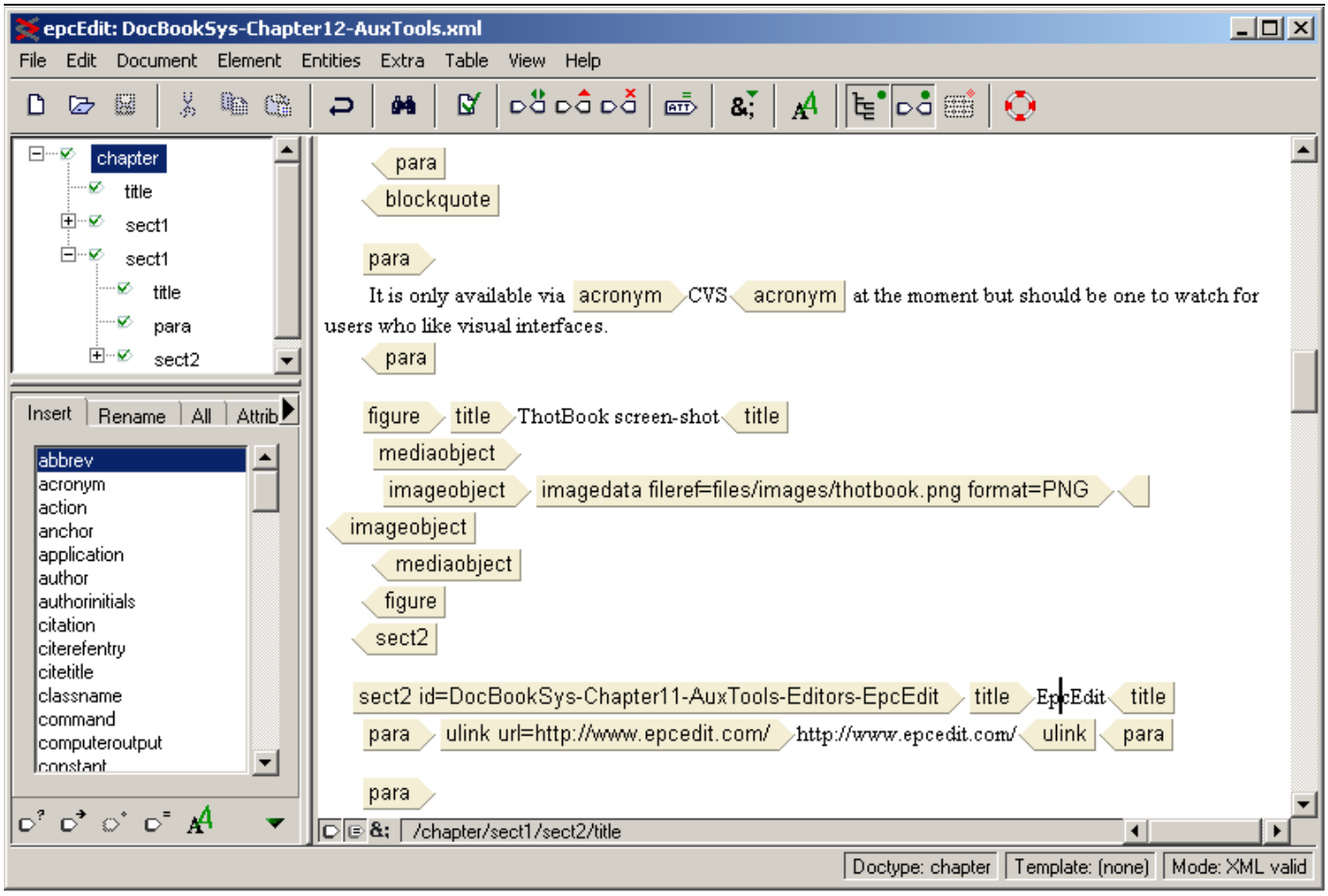
http://www.epcedit.com/

EpcEdit is a commercial GUI editor for Unix and Windows, it has many features such as integrated enhanced validating XML/SGML parser, to quote the website:

> epcEdit provides a structure-sensitive view of XML or SGML documents and allows editing of a document while keeping it in conformance with the document's DTD. An integrated table editor helps in visualizing the layout of tables conforming to the CALS or HTML table models. The hierarchical structure of a document is represented by a tree view that is updated while the document changes.

Figure 10.4. EpcEdit screen-shot

ing



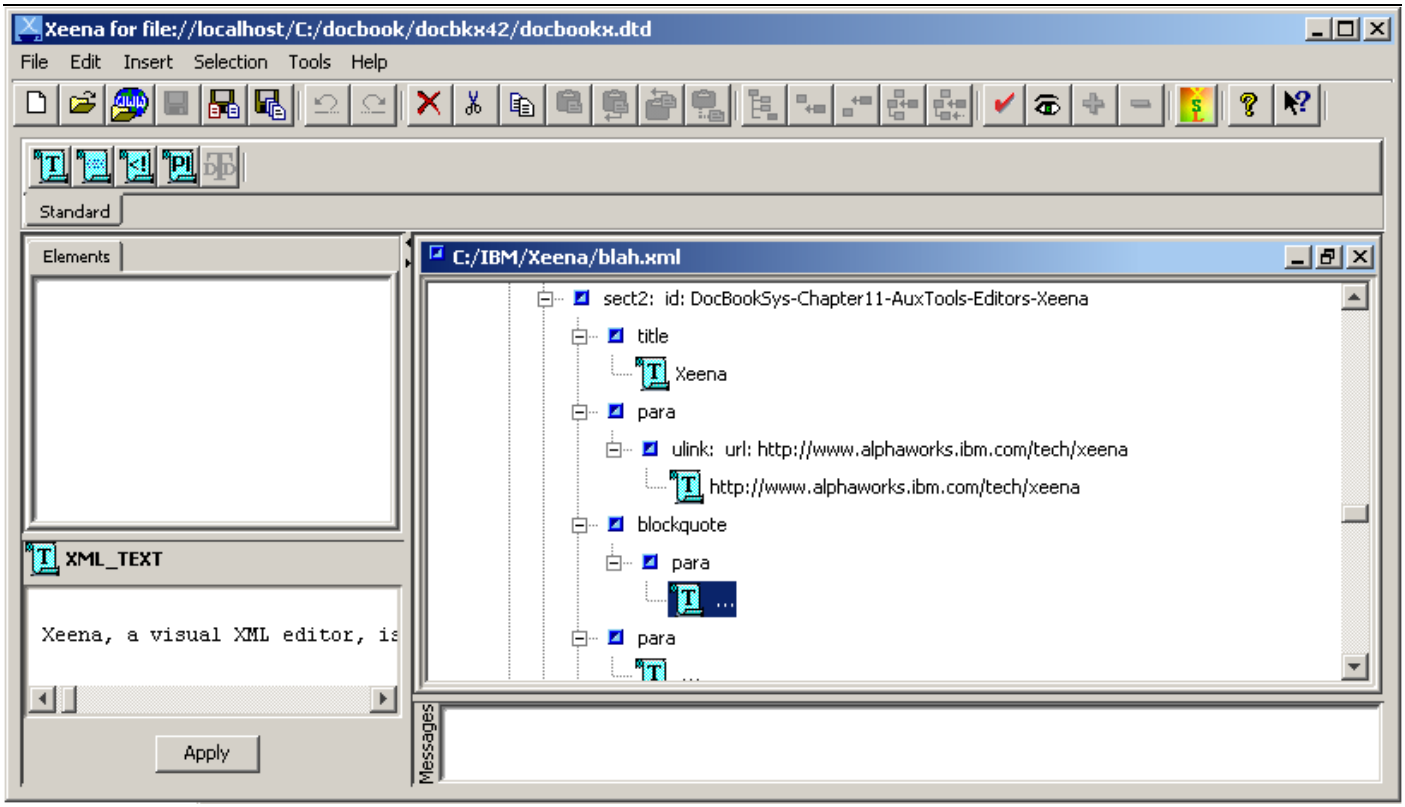
1.5. Xeena

http://www.alphaworks.ibm.com/tech/xeena

> Xeena, a visual XML editor, is a generic Java application for editing valid XML documents derived from any valid DTD. XML files can be created and edited without learning the intricacies of XML. The editor takes as input a given DTD and automatically builds a palette containing the elements defined in the DTD. Any document derived from that DTD by using a visual, tree-directed paradigm can thus be created, edited, or expanded. The visual paradigm requires only a minimal learning curve, because only valid constructs or elements are presented to the user in a context-sensitive palette.

The above quote is according to the website, I installed it and tried it out and it looked OK but I took it no further. As far as licensing goes, it appears that the software is free for "Independent development", but what this means is unknown by me at the moment. It is platform independent.

Figure 10.5. Xeena screen-shot



1.6. The rest

There are quite a few editors available and there is likely to be a lot more in the future. Keep an eye out. Here is a short list of some others, all descriptions are taken from the websites indicated, all rights reserved.

- http://www.morphon.com/xmleditor/index.shtml

  The Morphon XML-Editor is a validating XML-Editor which lets you easily create and modify XML documents. Morphon also provides a CSS editor for use as a styling language. The Editor itself is able to run on the maximum number of platforms as it is written in Java, and uses the Java Foundation Classes (JFC) Libraries.

- http://www.xmlmind.com/xmleditor/

  XMLmind XML Editor (XXE for short) is an XML editor featuring DTD-aware editing commands and a word processor-like view configured using W3C's cascading style-sheets (CSS).

- http://fabrice.bellard.free.fr/qemacs/

  QEmacs (for Quick Emacs) is a very small but powerful UNIX editor....WYSIWYG DocBook mode based on XML/CSS2 renderer.

2. Spell Checking

Since a DocBook document is full of tags it can be quite hard to check the spelling of such a document without having the spell-checker constantly complain. Fortunately there are spell-checkers available out there that can ignore tags.

2.1. Aspell

http://aspell.sourceforge.net/

> Aspell is an Open Source spell checker designed to eventually replace Ispell. Its main feature is that it does a superior job of coming up with suggestions than just about any other spell checker out there for the English language.

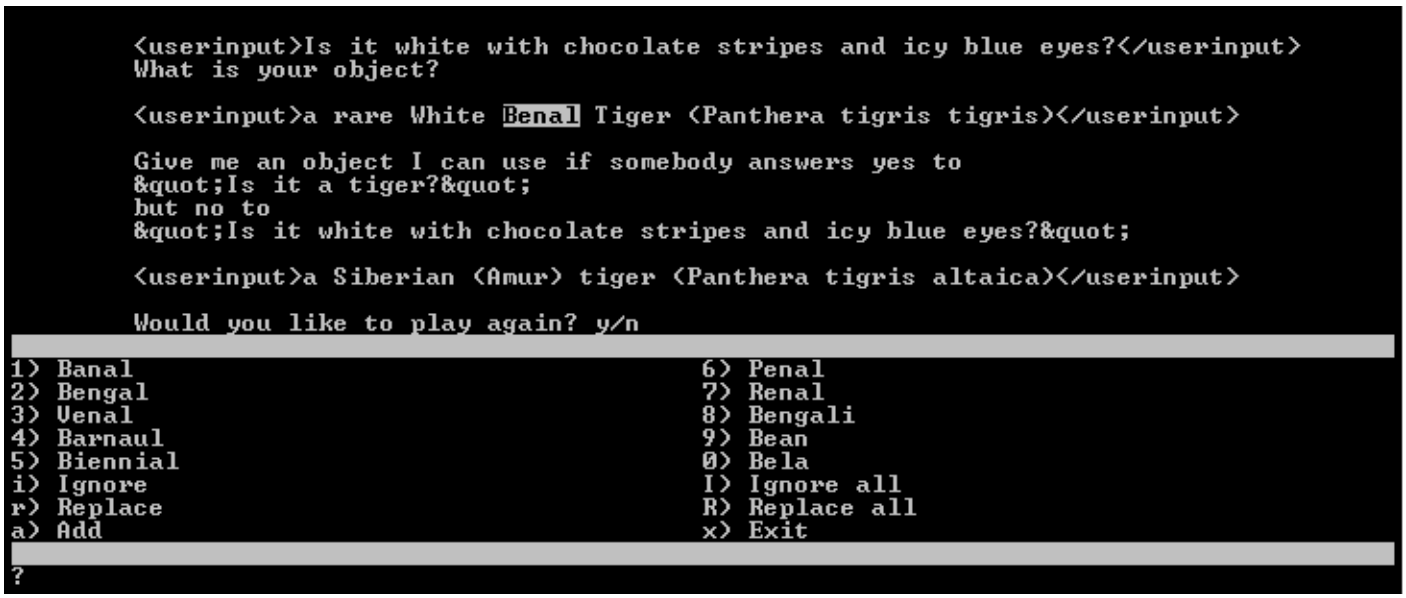
It is available for both Unix and Windows, and while integration into editing environments is possible, it is possible to use it standalone. One can easily spell check a DocBook file with the following command sequence.

```
aspell - -mode=sgml check in.xml/sgml
```

This will bring up a spell checking program with the usual choices of adding to the dictionary, replacing all occurrences of a word and so on. The SGML mode works for both SGML and XML documents. And there are dictionaries for about 16 or so different

languages excluding the standard British or American English language packs.

Figure 10.6. Aspell screen-shot



Chapter 11. References

1. Internet Resources Used

Thanks goto all the people who helped me when I was stuck.

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

  DocBook: The Definitive Guide
  by Norman Walsh and Leonard Muellner
  With contributions from Bob Stayton
  ISBN: 156592-580-7
  Version 2.0.6
  Updated: Wed, 12 Jun 2002

- http://ourworld.compuserve.com/homepages/hoenicka_markus/ntsgml.html
  SGML for NT: A brief tutorial how to set up a free SGML/XML editing and publishing system for Windows
- http://www.dpawson.co.uk/docbook/index.html
  Docbook Frequently Asked Questions
- http://www.bureau-cornavin.com/opensource/crash-course/index.html

  DocBook XML 4.1.2 Quick Start Guide
  Jim Weller
  Sleepless Tech

  jim@sleeplesstech.com

  2001-05-29

- http://nis-www.lanl.gov/~rosalia/mydocs/docbook-intro.html

  Get Going With DocBook
  Notes for Hackers
  Mark Galassi
  Cygnus Solutions

- http://people.freebsd.org/~nik/nwalsh/Customising%20the%20nwalsh%20DSSSL%20stylesheets_files/v3_document.htm
  Customising the nwalsh DSSSL stylesheets nik@FreeBSD.org
- http://old.lwn.net/2000/features/DocBook/

  Exploring SGML DocBook
  April 24, 2000
  Giorgio Zoppi

- http://cyberelk.net/tim/docbook/selfdocbook/selfdocbook.html

  Selfdocbook

Tim Waugh
Red Hat, Inc.

twaugh@redhat.com

Copyright 2000-2 by Tim Waugh

- http://www.oasis-open.org/cover/general.html

The XML Cover Pages
SGML: General Introductions and Overviews
By: Robin Cover

Last modified: June 28, 1999

- http://www.sagehill.net/xml/docbookxsl/index.html

Robert Stayton
<bobs@sagehill.net>

Draft - - 7 July 2002

Copyright 2002 Robert Stayton

- http://www.ibiblio.org/pub/sun-info/standards/xml/why/xmlapps.htm

XML, Java, and the future of the Web
Jon Bosak, Sun Microsystems
Last revised 1997.03.10

- http://www.ibiblio.org/godoy/sgml/docbook/howto/

DocBook HOWTO
Jorge Godoy
Conectiva S.A.
Publishing Department

godoy@conectiva.com

- http://www.w3.org/TR/NOTE-sgml-xml

Comparison of SGML and XML
World Wide Web Consortium Note 15-December-1997

- http://docbook.sourceforge.net/release/dsssl/current/doc/

The Modular DocBook Stylesheets
Norman Walsh
Copyright 1997, 1998, 1999, 2000 by Norman Walsh

- http://docbook.sourceforge.net/release/xsl/current/doc/

DocBook XSL Stylesheet Documentation

Norman Walsh
Bob Stayton

Jiri Kosek

- http://www.tldp.org/LDP/LDP-Author-Guide/

LDP Author Guide
Mark F. Komarinski
mkomarinski@wayga.org

Jorge Godoy
Conectiva S.A.
Publishing Department

godoy@conectiva.com

godoy@metalab.unc.edu

David C. Merrill
dcmerrill@mindspring.com

- http://www.freebsd.org/doc/en_US.ISO8859-1/books/fdp-primer/

FreeBSD Documentation Project Primer for New Contributors
Nik Clayton
nik@FreeBSD.org

- http://www.tldp.org/HOWTO/mini/DocBook-Install/index.html

DocBook Install mini-HOWTO
Robert B Easter
reaster@reaster.com

- http://sources.redhat.com/ml/docbook-apps/
  The docbook-apps mailing list archives