

Institutionen för datavetenskap
Department of Computer and Information Science

Master's thesis

**CUSTOMIZATION OF DOCBOOK TO GENERATE
PDF, HTM & CHM**

by


Muhammad Asif

LIU-IDA/LITH-EX-A--09/053--SE

Linköping, 2009



Linköpings universitet

Avdelning, institution Division, department Institutionen för datavetenskap Department of Computer and Information Science	Datum Date <u>2009-10-20</u>	 Linköpings universitet
---	---	---

Språk Language <input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English <input type="checkbox"/> _____	Rapporttyp Report category <input type="checkbox"/> Licentiatavhandling <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input type="checkbox"/> Övrig rapport _____	ISBN <u>LIU-IDA/LITH-EX-A--09/053--SE</u> ISRN <u>LIU-IDA/</u> Serietitel och serienummer ISSN _____ Title of series, numbering
--	--	--

URL för elektronisk version

Titel Title CUSTOMIZATION OF DOCBOOK TO GENERATE PDF, HTM & CHM Författare Author Muhammad Asif
--

Sammanfattning Abstract <p>Software documentation is an important aspect of software projects. Software documentation plays a key role in software development if it is up-to-date and complete. Software documentation should have the synchronization with the software development. One of the problems is duplication; same information is written in different documents and stored in different places with different formats making things complex to manage. By using traditional documentation tools, it's hard to maintain documentation for complex systems and it is time consuming.</p> <p>To overcome these problems, we have used XML Docbook that is a good solution for it. Docbook provides single sourcing technique in which documents are written ideally in one place and can convert it into different other formats from the same location. Actually docbook is based on xml which can be easily edited by most of the programming languages. If there are many developers are writing documentation for their software modules then we don't need to copy and paste all the documents into one document to produce a complete document for the software product. We have to just add the references to all those files that should be present in the final document and then compile it with some processors and it automatically get document contents from all files and put it into one document, so it's easy to handle and maintain software documentation with docbook.</p>

Nyckelord Keywords XML, Docbook, single source, documentation
--

**Intitutionen för datavetenskap
Department of Computer and Information Science**

Master Thesis

**CUSTOMIZATION OF DOCBOOK TO
GENERATE PDF, HTM & CHM**

LIU-IDA/LITH-EX-A--09/053--SE

By

Muhammad Asif

Supervisor: Dr. Rego Granlund (IDA, Lith)
Examiner: Dr. Arne Jönsson (IDA, Lith)

Dedication

I dedicate my work to my loving parents. Without their love, prayers, encouragement and moral support it was difficult to complete the work.

Acknowledgment

This thesis work was performed at Department of Computer and Information Science, IDA at Linköpings University, Sweden.

First of all I am thankful to ALLAH ALMIGHTY for providing me the strength to complete my work successfully.

I am really thankful to my supervisor Dr. Rego Granlund for his continuous guidance, inspiration and fruitful advices to complete my work. Without his guidance, it was difficult to complete my work within given time span. My special thanks to Muhammad Ayaz student of Software Engineering and Management at LITH for his support and helpful comments.

I am equally grateful to my colleagues Rizwan Rashid, Adeel Blouch and Abdul Qudus for their comments and suggestions. And I would like to thanks to all of my friends for their companionship.

Abstract

Software documentation is an important aspect of software projects. Software documentation plays a key role in software development if it is up-to-date and complete. Software documentation should have the synchronization with the software development. One of the problems is duplication; same information is written in different documents and stored in different places with different formats making things complex to manage. By using traditional documentation tools, it's hard to maintain documentation for complex systems and it is time consuming.

To overcome these problems, we have used XML Docbook that is a good solution for it. Docbook provides single sourcing technique in which documents are written ideally in one place and can convert it into different other formats from the same location. Actually docbook is based on xml which can be easily edited by most of the programming languages. If there are many developers are writing documentation for their software modules then we don't need to copy and paste all the documents into one document to produce a complete document for the software product. We have to just add the references to all those files that should be present in the final document and then compile it with some processors and it automatically get document contents from all files and put it into one document, so it's easy to handle and maintain software documentation with docbook.

Key words: XML, Docbook, single source, documentation

ACRONYMS AND ABBREVIATIONS

API	Application Programming Interface, a source code interface provided by computer system or application library.
CHM	Microsoft Compressed HTML Help, a help manual format based on HTML.
CLI	Command Line Interface, non-graphical user interface for the application.
CSS	Cascading Style Sheet, style definition file for HTML.
DTD	Document Type Definition, technique to validate documents written in XML.
DocBook	XML based format designed for technical documentation.
FOP	Formatting Objects Processor, part of the Apache XML Graphics project.
GUI	Graphical User Interface, visual user interface for the application.
HHC	HTML Help Compiler, used to produce CHM documents.
HTML	Hyper Text Markup Language, used for creation of web pages.
HTTP	Hyper Text Transfer Protocol, communication method to transfer for example HTML pages.
OASIS	Organization for the Advancement of Structured Information Standards, a non-profit international consortium that drives the development and adoption of e-business standards.
ODF	Open Document Format standardized documenting format for office applications.
PDF	Portable Document Format widely used printing format developed by Adobe. PS PostScript, a page description and programming language used primary in the electronic publishing.
SGML	Standard Generalized Markup Language. Meta language, predecessor of XML.
SVG	Scalable Vector Graphics, XML based format for two dimensional vector graphics.
SQL	Structured Query Language, the most popular computer language to create, modify, retrieve and manipulate data in the relational database.
TeX	Typesetting system, developed in the beginning of 1980, but still widely used especially in the academic spheres.

TIFF	Tagged Image File Format, popular image format for high color depth images.
TOC	Table of Contents shows the structure and the listing of the main entries in the document.
W3C	World Wide Web Consortium, a group formed by over four hundred organization, which controls and develops common web techniques.
WikiText	Wiki type of website allows easy modification and additions to the content. Term also means text based, both computer and human readable documenting format.
XML	Extensible Markup Language, widely used for information definition developed and controlled by W3C.
XSL	Extensible Stylesheet Language, definition for products that are used to format and interpret XML documents.
XSL-FO	Extensible Stylesheet Language Formatting Objects, markup language for document formatting. Used mainly to generate PDF documents.
XSLT	Extensible Stylesheet Language Transformation, formatting rules especially for XML data.
XSLTProc	Open source XSLT processor available for multiple operating systems.

Table of contents

1	Introduction.....	11
1.1	Objective.....	12
2	Software Documentation	13
2.1	Documentation.....	13
2.1.1	Requirement documentation	13
2.1.2	Technical Documentation.....	13
2.1.3	User documentation.....	14
2.2	Software documentation issues	15
2.3	Documentation format.....	16
3	Docbook Vs Latex	20
3.1	Docbook.....	20
3.2	Why use docbook	20
3.3	Maturity.....	22
3.4	XML/SGML.....	23
3.5	Data Separation	25
3.6	Modularity	26
3.7	Docbook Advantages.....	28
3.7.1	Profiling	28
3.8	Docbook Disadvantages	31
3.9	Latex	31
3.9.1	Features of Latex	32
3.9.2	Basic Layout of Latex	32
3.9.3	What is TeX.....	34
3.9.4	BibTeX.....	34
3.9.5	SliTeX	34
3.10	Advantages and disadvantages of LaTeX/TeX.....	34

3.11	Docbook usage over Latex.....	37
4	Document building & Scripting	38
4.1	Xml Docbook	38
4.2	XSLT Style sheet.....	39
4.3	4.3 CSS (Cascading Style Sheet).....	42
4.4	XSL-FO.....	43
4.5	XSLTPROC	45
4.6	FOP	45
4.7	FOP Limitations.....	47
4.8	HTML Help Compiler	48
4.9	Htmlhelp.hhp.....	48
4.10	Docbook Processing Options.....	48
4.10.1	Display the menu.....	49
4.10.2	Custom buttons	49
4.10.3	Table of contents pan.....	50
5	Implementations	51
5.1	Installation of cygwin	51
5.2	Docbook to html and chm.....	56
5.3	Adding an index.....	59
5.4	How to produce single html file	60
5.5	Tables.....	62
	Chapter 1. On Foo's	63
5.6	Links	64
5.7	Graphics.....	64
5.8	Figures	64
5.9	Special formatting	65
5.10	Plain text formatting	65

5.11	Customizing the style sheets	66
5.12	CSS Support	67
5.13	Custom header and footers.....	69
5.14	Docbook versioning	70
5.15	Docbook to pdf	71
6	Conclusion and Final Work.....	72
6.1	C3Fire Problems	72
6.2	Conclusion	72
6.3	Future Work	73
7	References.....	74

Table of figures

Figure 2-1 B is Transcluded in the document A.....	18
Figure 3-1 Docbook structure.....	21
Figure 3-2 Docbook build process	22
Figure 3-3 XML Document Components	24
Figure 3-4 SGML Document Components.....	25
Figure 3-5 Data and Style separation	26
Figure 3-6 Document references.....	27
Figure 3-7 Source document to other formats process	31
Figure 3-8 LATEX relationship with other formats	35
Figure 3-9 Latex output	36
Figure 4-1 XSLT Processing Model.....	39
Figure 4-2 XSLTPROC Processing.....	45
Figure 4-3 FOP Rendering.....	46
Figure 5-1 Installation directory	52
Figure 5-2 Choose download site	53
Figure 5-3 Select Packages	54
Figure 5-4 HTML Output.....	57
Figure 5-5 CHM Output	59
Figure 5-6 CHM with Index.....	60
Figure 5-7 Single HTML Output	61
Figure 5-8 HTML Header	70

1 Introduction

One of the cornerstones to any quality program is documented processes. Processes are “codified good habits” [Down-94] that “define the sequence of steps performed for a given purpose” [IEEE-610]. By using software documentation in a proper way, we can find that what works best in our organization and where are the faults.

We can make better planning for the new coming projects because with the help of appropriate software documentation we can have the idea that what we have learned in the previous project. So that we can repeat our successes in the incoming projects and stop repeating those actions that leads to problems. In this way we can eliminate the need to “reinvent the wheel” with each new projects by providing a basic architecture to the new project.

Chisholm has pointed out that how-to documents have been closely associated with the use of products [Chisholm, 1988]. Documents cover the gap between products and its potential customers that how to use the products and what features, functionalities contain this particular product. So documents are helpful for the customers to understand and operate the products themselves. Software documentation has very important role in software project. Documents are needed to plan, analyze, design and storing the information for the future usages. Software documentation might help other resource groups to get benefit from our process and save their time.

Normally Microsoft word is used for the software documentation which is easy to use. It works well in small projects to fulfill the basic requirement of documentation but the increasing competition, complexity of systems and accelerating development have made it necessary to look for alternative, possibly more efficient documentation tools, formats and methods. In Microsoft word, the technical writer or anyone who is involve in writing documentation put more concentration on the formatting rather than the document contents. In this thesis we have used XML Docbook to generate the documentation. XML Docbook is a scripting language based on XML and used for writing technical documentation. It provides lot of benefits over traditional software documentation tools. XML Docbook provides single sourcing which means that with one xml docbook source, we can generate lot of other formats according to the requirement and there will be no change effect on the

source document. In this thesis, we will also discuss about how to convert xml docbook to PDF, HTML and CHM formats.

Most of the time of software developer spends on maintenance and for software maintenance two things should be in documentation, first it should be updated and second it should be completed. Actually without documentation it is very difficult to do the software maintenance because by looking into code it's hard to get the idea of a specific module implementation.

1.1 Objective

The objective of this thesis is

- ✚ To use docbook to generate html, pdf and chm formats.
- ✚ Comparison of docbook with latex.
- ✚ Docbook customization and its implementation.
- ✚ Generate the different formats of C3Fire Project documentation.

In this thesis, the documentation is written for a C3Fire project so that it's easy to maintain the future updates in it and to convert it into different target formats. Actually there are many versions of this project for different target audiences so that docbook is used to maintain and generate different versions according to the requirements. C3Fire is an environment that supports training and research in team collaboration. The environment is mainly used in Command, Control and Communication research and in training of team decision making [c3fire.org¹].

¹ <http://c3fire.org/c3fire/home/home.en.shtml>

2 Software Documentation

There are different types of documentation in each phase of software development that is used by different persons in a software firm.

2.1 Documentation

2.1.1 Requirement documentation

Requirement documents are the description about the software that what functionalities and features are performed or will be performed. This documentation used throughout the software development life cycle to communicate that what the software does or shall do. It is also used as an agreement or foundation for agreement that what type of functionalities will be performed by the software. Requirements are produced and consumed by everyone that involved in the production of the software like end users, customers, product managers, project managers, sales, marketing, software architects, usability experts, interaction designers, developers, and testers, to name a few. Thus, requirements documentation has many different purposes. It is difficult to estimate that how much documentation is needed for the software project. Requirement documentation depends on the complexity of product. If the product is very complex then of course more documentation is needed to cover all of its modules and if the product is small then little documentation is enough. Some time we need more formal documentation if the product is very critical and can have negative impact on human life like “Nuclear power systems or Medical software systems”. Requirement documentation is very important when there is need to modify some of the component of the software. Otherwise it’s difficult to trace out that what was the actual behavior of the software. Without proper requirement documentation software changes become more difficult and there for more error prone [wiki²].

2.1.2 Technical Documentation

The term 'technical documentation' refers to different documents with product-related data and information that are used and stored for different purposes. “Different purposes” mean: Product definition and specification, design, manufacturing, quality assurance, product liability, product presentation; description of features, functions and interfaces; intended,

² http://en.wikipedia.org/wiki/Software_documentation

safe and correct use; service and repair of a technical product as well as its safe disposal[transcom.de³].

Technical documentation deals with the programmers during development of software. When software developers develop some complex and big software modules, they need to write technical documents about different modules and functions. These documents contain description of the code but not in a verbose mode, otherwise it is difficult to maintain in future. Normally software products documented by using API Writers. Technical documents are used by the developer when they need to modify some part of the software product otherwise it is difficult and take more time to check out that what is the functionality of a particular code/function. Often, tools such as Doxygen, NDoc, javadoc, EiffelStudio, Sandcastle, ROBODoc, POD, TwinText, or Universal Report can be used to auto-generate the code documents.

Normally software developers write comments about code during the coding phase to understand it easily later on and also when some other developer do the inspection of the code, he or she can easily understand it. The above tools are used to extract these comments from the source code and produce reference manuals in the form of text or html files [wiki⁴].

2.1.3 User documentation

User documents are usually more diverse as compared to the technical documents because it contains each and everything about the products that how to use it and how to troubleshoot it. User documents are written in way that they can easily understand it because all the users are not the technical persons. User documents are also used by the software tester during usability testing. It is very important that user document should be comprehensive and not a confusing. User documents should be up-to-date [wiki⁵].

Some people don't think that incomplete user documentation as a problem because they believe the myth that no one read documentation normally. According to the recent data

³ <http://transcom.de/transcom/en/technische-dokumentation.htm>

⁴ http://en.wikipedia.org/wiki/Software_documentation

⁵ http://en.wikipedia.org/wiki/Software_documentation

from Dataquest, 85% people solve their problem by reading documentation. Many of the people used their manuals before calling to the support. If the user manuals are incomplete, out dated, then the customers will be frustrated and create false expectations about the way the program should work. If the user manual and help is correct and up to date then lot of support calls can be avoided and time is saved. Errors that mislead the customers about the functionalities of the product can lead to repeated, frustrated, support calls and unpleasant views about the company's other products as well. Some time user manual index is incomplete and pointed to the wrong information. The table of contents provides no hint, where to find the correct information and some time the information is incomplete, incomprehensible or spread across to many places in the manual [Cem Kaner, 2000].

2.2 Software documentation issues

Some time simple systems are not necessarily easy to document and complex system do not always require complex documentation. One of the major problems is that technical writers and editors don't have their professional skills to create user manuals.

Software documentation is plagued by various kinds of issues. Despite all the time used to write software documentation, they are often considered of a poor quality, incomplete and outdated. Furthermore, there seems to be a lot of false prejudices and presumptions about documentation writing and usage, but also about the quality and quantity of documentation [GREGORY R. McARTHUR, 1986].

Software engineers rely on software documentation to understand the system, its high level design and implementation details of complex applications. Unfortunately, the documentation of most of the software systems is normally out dated. So the developers usually don't trust on it and focus on the source code. But it's time consuming and error prone process.

One way of producing accurate documentation for the existing system is through reverse engineering. In fact many tools can create documentation, graphical view of software systems and extract the hidden knowledge from the source code. However the truth is that no one knows that what type of documentation is useful. If no one knows what is required, it should come as no surprise that tools that produce this type of documentation are rarely used by real-world software engineers. This situation raises many fundamental questions:

- What types of documentation does a software engineer need? What formats should the documentation take? For example, inline or linked textual commentary? Graphical views? Multimedia?
- Who will produce the document? What is the role of technical professional communication in the process? Who will maintain document when it is produced? [Bill Thomas, 2001]

2.3 Documentation format

Microsoft word is nowadays using for the software documentation by most of the software companies. Due to lot of use of Microsoft word, it also tends to be the tool causing more frustration. Complex applications some time produce multi-volume references causing confusion that which reference manual should be selected and where to locate the information in the manual [Novick David G, 2006].

Documentation format can be categorized into several formats like they can be stored in text file or binary files etc. There are several formats available that is based on xml. Moreover, document contents can be defined by using either structural or semantic information, or alternatively their definition can be based on typesetting rules. Originally, the document format can always be any combination these, too. Open Document format (ODF) is an OASIS standardized documentation format for office applications. The Open Document Format (ODF) is an open XML-based document file format for office applications to be used for documents containing text, spreadsheets, charts, and graphical elements. The file format makes transformations to other formats simple by leveraging and reusing existing standards wherever possible. As an open standard under the stewardship of OASIS, Open Document also creates the possibility for new types of applications and solutions to be developed other than traditional office productivity applications [oasis-open.org]. ODF is comparable with the Microsoft word format and it is not considered very different for the MS word format. Microsoft introduces a new Open XML format for office applications and it shares the same ideology of Open Document format like metadata, style and other resources are split into separate units and finally zipped as a single file. However, while style and data are separated in terms of files, they do not provide full separation as the format contains references to the style definition file. Furthermore, the format also uses less descriptive and non-semantic names for the XML elements, making it quite hard to follow.

Microsoft has also developed MAML (Microsoft Assistance Markup Language) which is xml based and used for “Longhorn” Help. The current help system HTML Help 1.x is using HTML topic files. HTML is a markup language that combines presentational and semantic elements. The most significant aspect of MAML is the shift to a structured authoring model. In MAML, the focus is on contents rather than the formatting and presentation is controlled at rendering time. MAML contain lot of content types, each one specific to a type of document. The MAML content types include: conceptual, FAQ, glossary, procedural, reference, reusable content, task, troubleshooting, and tutorial. Contents authored in MAML can be output into many formats like DHTML, XAML, RTF, and print. There are three levels of run-time transformation: structural, presentational, and rendering.

Example

```
<conceptual>
  <title />
  <content>
    <para />
    ...
  </content>
  <sections>
    <section>
      <title />
      <content>
        <para />
        ...
      </content>
    </section>
    ...
  </sections>
</conceptual>
```

[Help-info.de⁶]

Another documenting tool is wiki. Wiki has introduced a dramatic change in documentation solutions. The term WikiWikiWeb is associated to the web based solution that facilitate the users to add, edit and delete the desired contents of it. It provides a very simple and easy interface to do modifications in the contents of wiki. It is quite easy to learn the scripting language for wiki. There is no commonly accepted standard for wiki text language. The grammar, feature, structure, and keywords depend on particular wiki software that is used for a particular website. Wiki text Markup Language provided a very easy syntax for hyper

⁶ http://www.help-info.de/en/Help_Info_AP_Help/longhorn_maml_example.htm

linking to other web pages within the website but there are also some other way for hyper linking web pages with each other. Many wikis, especially the earlier ones, used Camel Case to mark words that should be automatically linked [en.wikipedia.org⁷].

A simple example of wiki documents is shown in the figure as

= Simple Wiki Document =

== First Chapter ==

This document is really "simple", but complex enough to show how a short example:

```
{{
#!python
from datetime import datetime
#show current date and time
print datetime.now().isoformat()
}}
```

Wiki normally comes with browser based solution with the functionality to create, edit, search and recognize pages. Additionally, the history of changes can be reviewed, comments can be left and existing material can be reused by using the transclusion mechanism [Green Robin, 1997]. Transclusion is the inclusion of part of a document into another document by reference as shown in the figure

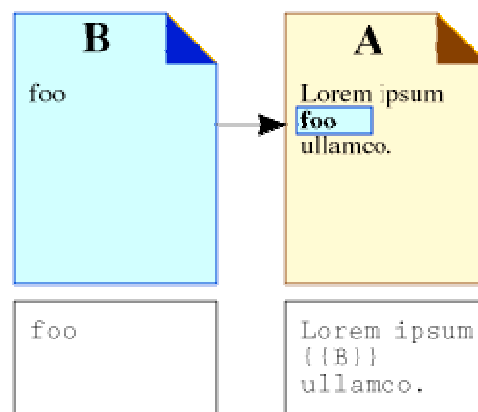


Figure 2-1 B is Transcluded in the document A

Most of the wiki solutions store documentation in relational or file-like databases. There should be the connection to the system to read the wiki documents where as offline

⁷ <http://en.wikipedia.org/wiki/Wikitext>

documents cannot be used to read. Wiki did not meet the high standards for the layout of the deliverable document formats [en.wikipedia.org⁸].

⁸ <http://en.wikipedia.org/wiki/Transclusion>

3 Docbook Vs Latex

3.1 Docbook

Docbook is a general purpose document format being designed, but not limited to computer hardware and software documentation. Docbook uses both xml and sgml. Docbook is standardized and maintain by OASIS. Docbook is a popular format for electronic publishing and features an XML representation. One of the advantages of using Docbook, single-source publishing is arguably the most useful. A Docbook document can be converted into many different formats, such as HTML or PDF, without having to change the source document [ausweb.scu.edu.au⁹].

Docbook is a markup language that is defined by xml or sgml document type definition (DTD). Docbook is a set of tags that define the structure of document. It is much more similar to HTML tags but more useful then plain HTML because it can be converted into several formats. Basically docbook is developed for the documentation of open source projects like Linux [ibm.com¹⁰].

3.2 Why use docbook

The main advantage of docbook is its portability. A document written in Docbook markup can be converted into HTML, PostScript, PDF, RTF, DVI, and plain ASCII text easily and quickly without any expensive tools. Actually docbook and all others tools that are used with docbook to convert it into many formats are free and under open source licenses. Another thing is that docbook documents are written in plain text so that any text editor can be used for it. The author of the docbook doesn't need to take care about the layout and formatting of document. This is main difference between docbook and other word processors that by using Microsoft word, we need to take care about the formatting and contents both at the same time but in docbook the author only concentrate on the contents of the document rather than its formatting. Actually the formatting part is stored in a separate file like CSS which is applied during the rendering of the document [ibm.com¹¹].

⁹ <http://ausweb.scu.edu.au/aw05/papers/edited/ball/poster.html>

¹⁰ <http://www.ibm.com/developerworks/library/l-docbk.html>

¹¹ <http://www.ibm.com/developerworks/library/l-docbk.html>

Using standard docbook tags we can build a complete document using its syntactic structure. The Docbook document is then processed using XSL style sheets so that each tagged Docbook element is transformed to a corresponding element in the target output format. For example each <Para></Para> element in Docbook could be transformed into a <p></p> element in XHTML. Instead of setting the style, color and font for each text, the content of the document is defined. The granularity of the parts depends on the used document format as shown in the figure.

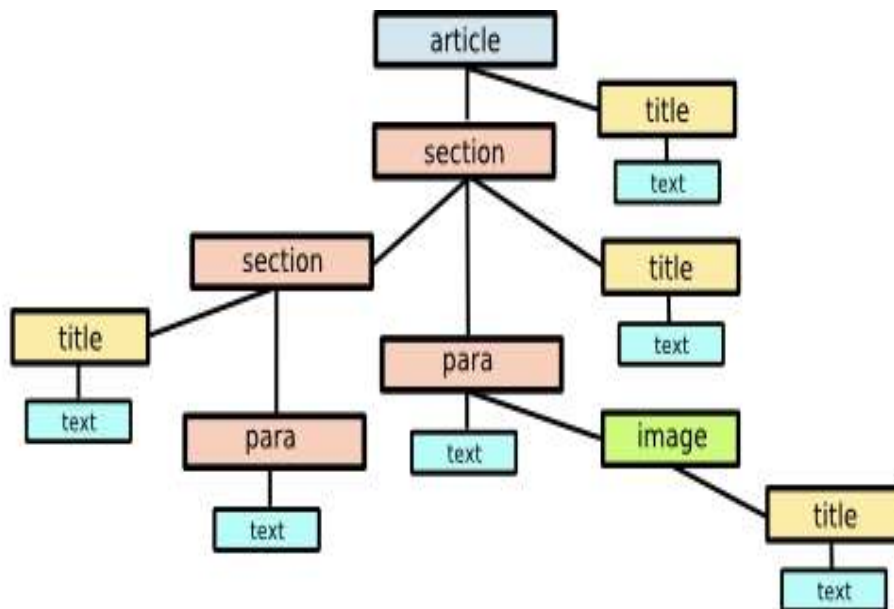


Figure 3-1 Docbook structure

Using different XSL style sheets, we can generate different output formats. For example, we can generate both XHTML and PDF outputs from a single Docbook source. We can also generate multiple versions of XHTML (or PDF) files each with a different style if necessary as shown in the figure.

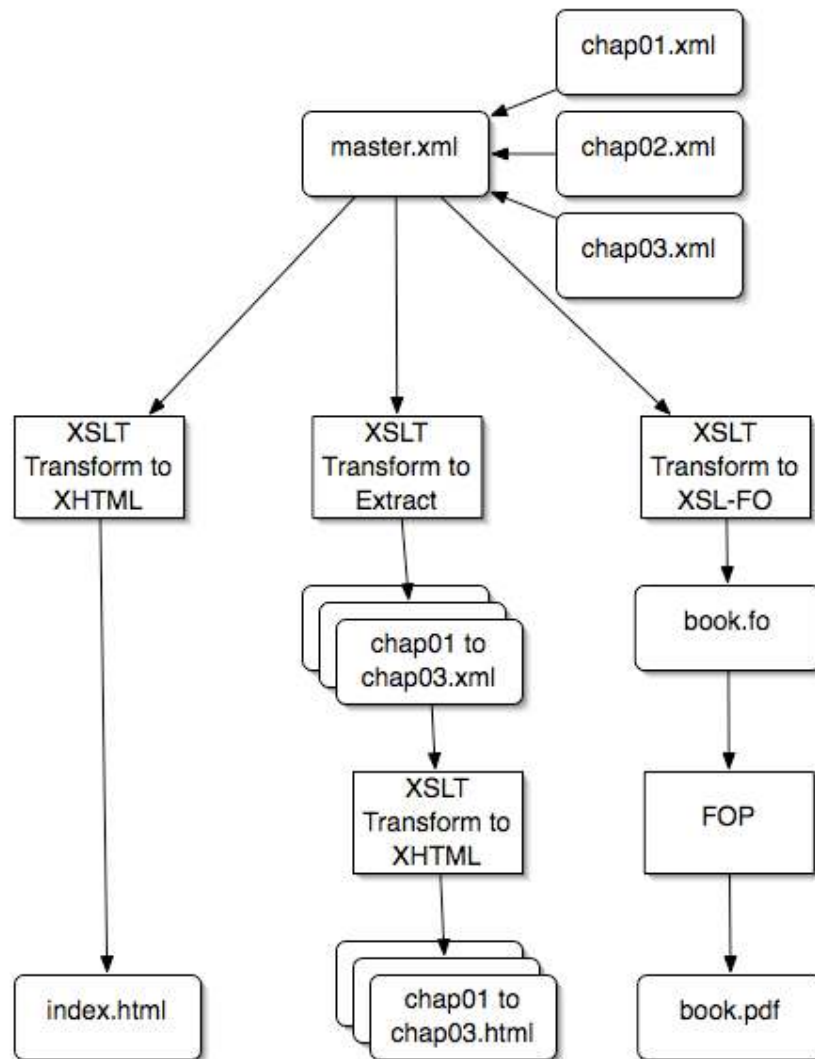


Figure 3-2 Docbook build process

[docs.jboss.org¹²].

3.3 Maturity

Docbook has been developed since 1991 and today it is enough mature and OASIS standardized technical documentation format that is quite widely used in both open source and commercial projects [www.docbook.org]. Actually there is a strong community behind it making it technically strong day by day.

¹² http://docs.jboss.org/docbook/userguide/html_single/

3.4 XML/SGML

The docbook document based on xml or sgml that provide advantages over the other documentation formats. As xml is widely used and there are lot of tools are available to create, edit, validating and querying it. Also the existing and developed techniques can be used in docbook, because it also based on xml.

The XML format itself is an understandable format between human and computer readability. Fortunately, the Docbook definition uses logical element names as show in the figure below

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD Docbook XML V4.4//EN"
"http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd">
<book>
<title>Simple Docbook Document</title>
<bookinfo>
<author>
<surname>Mustonen</surname>
<firstname>Juha</firstname>
<email>juham@ee.oulu.fi</email>
</author>
</bookinfo>
<chapter>
<title>First chapter</title>
<para>This document is really <emphasis>simple</emphasis>, but complex
enough to show how a short example:</para>
<example>
<title>Short example</title>
<programlisting language="python">
from datetime import datetime
#show current date and time
print datetime.now().isoformat()
</programlisting>
</example>
</chapter>
</book>
```

But it cannot be considered as readable as the Wiki format. The compact XML format also goes well with existing software projects.

Xml is subset of sgml. XML is designed for introduce an easy-to-learn way to use SGMLs structure-defining power and to combine it with HTMLs popular features to describe easily

text and graphics in the Internet. XML is a simplified version of SGML; XML was designed to maintain the most useful parts of SGML. Whereas SGML requires that structured documents reference a Document Type Definition (DTD) to be "valid," XML allows for "well-formed" data and can be delivered without a DTD. XML was designed so that SGML can be delivered, as XML, over the Web [irt.org¹³].

As we see from the following figures, structures of XML and SGML do not differ much. This is due the fact that XML is a real subset of SGML. The most important difference is that output specification is not defined by SGML, but it is fixed in XML as shown in the figure below.

XML Document Components

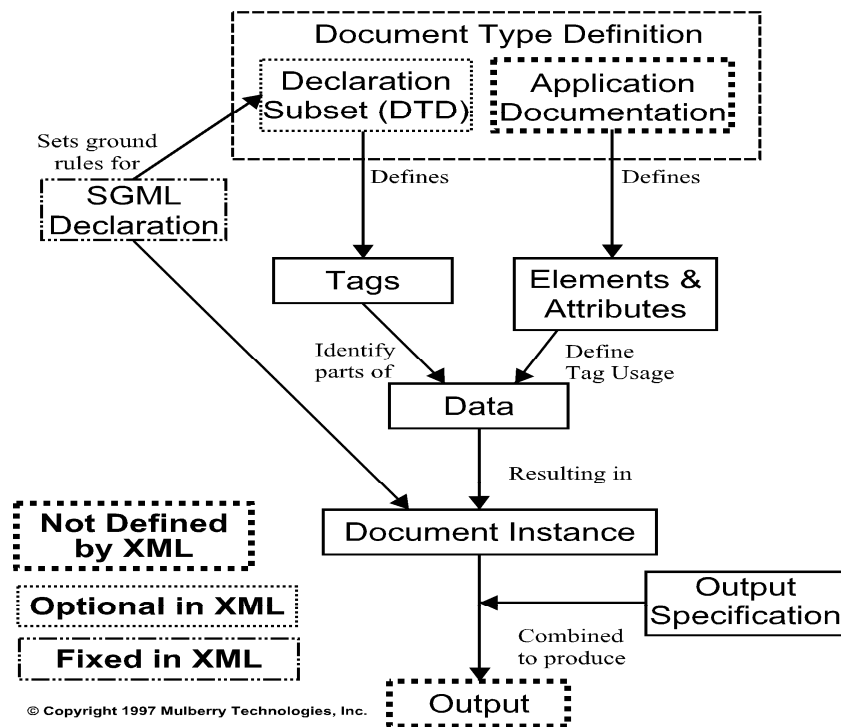


Figure 3-3 XML Document Components

¹³ <http://www.irt.org/script/5206.htm>

SGML Document Components

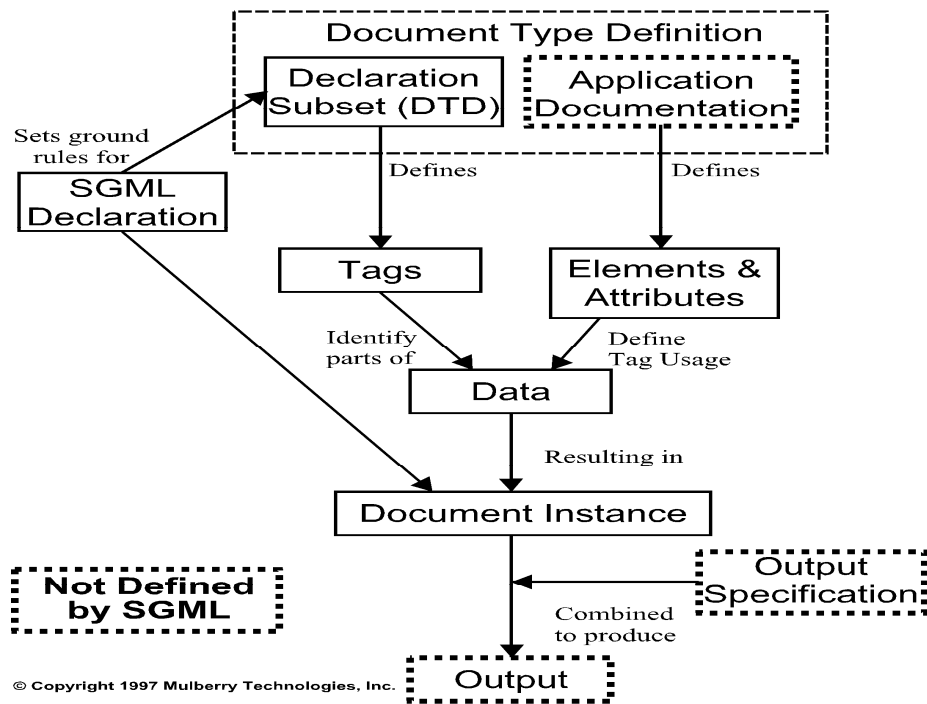


Figure 3-4 SGML Document Components

[students.tut.fi¹⁴]

3.5 Data Separation

There are a lot of differences between docbook and other word processors but the major difference is the data separation in docbook. In docbook content documents are written separately from its presentation. The formatting of the document is stored in CSS document. Whenever the template is changed, it can be applied to all the documents without any manual modifications to the source document. A more practical example is to generate the same document with different layouts, each filling its own specific purpose. Another, yet bigger, advantage is to produce multiple target formats from a single source. In general, Docbook documents are transformed into PDF, CHM and (X) HTML, but also other formats like Man pages, Java Help and WordML are supported. Therefore, the deliverable documents can be easily provided with the software in the format that is most suitable for the reader. The Docbook XSL style sheets are a set of XSLT style sheets for the XML-based Docbook

¹⁴ <http://www.students.tut.fi/~leppane7/leppanen.html>

language. XSLT is used for the transformation of xml document to other xml document. Actually xml document is notable for the presentation of its contents that's why XSLT style sheets are used to convert xml documents into html or xhtml documents for display as web page [wiki¹⁵].

The contents of original documents didn't changed rather than a new document is created based on the contents of original document. It is also used to create printed output. As docbook document is written in xml so XSLT stylesheet is used to convert it into target format. Also during the transformation, various styles, text and image definitions are added to the document, in order to get a more readable and better-looking output. High level design about the style and data separation is shown in the figure below.

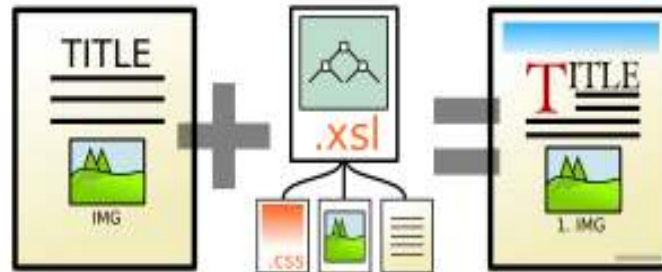


Figure 3-5 Data and Style separation

3.6 Modularity

One of the docbook features is its modularity in which instead of including everything in one document, we can divide it into separate files as shown in the figure below. For example its layout and formatting is kept separately. In the same way images are also kept separately from the original contents files. When we need to modify some of the parts of document then the focus is only on those parts instead of the entire document. In addition, documents can include other documents either partially or completely. The technique thus enables writing reusable document parts and updating them separately. This is also the idea behind the single-sourcing method and therefore it is an appreciated feature in software documentation. Docbook does not provide a self-made technique to include selective parts

¹⁵ <http://en.wikipedia.org/wiki/XSLT>

of another document inside the current one, but uses a standardized XInclude technique to do it [w3.org¹⁶].

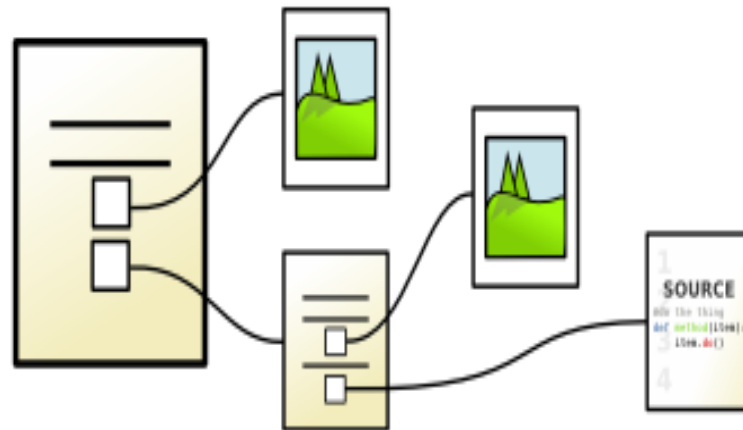


Figure 3-6 Document references

XInclude is generic method to include one document to other document either complete or partially.

For example an XHTML document

```
<?xml version="1.0"?>  
...  
<html xmlns="http://www.w3.org/1999/xhtml"  
  xmlns:xi="http://www.w3.org/2001/XInclude">  
  <head>...</head>  
  <body>  
    ...  
    <p><xi:include href="license.txt" parse="text"/></p>  
  </body>  
</html>
```

Will give

```
<?xml version="1.0"?>  
...  
<html xmlns="http://www.w3.org/1999/xhtml"  
  xmlns:xi="http://www.w3.org/2001/XInclude">  
  <head>...</head>  
  <body>
```






¹⁶ <http://www.w3.org/TR/xinclude/>

```
...
  <p>This document is published under GNU Free Documentation License</p>
</body>
</html>
[en.wikipedia.org17]
```

In this example we have included license.txt file which contain some text and by using XInclude, the text comes in the resulting document.

3.7 Docbook Advantages

There is lot of advantages of docbook over other word processors used for software documentation. Some of the advantages are

-  One source file, multiple outputs (mostly PDF and HTML)
-  Easy change tracking in SVN or a similar versioning system
-  Automatic cross-referencing
-  Automatic index generation
-  Separation of content and design (with XSL) [docbook.theblog.ca¹⁸]

3.7.1 Profiling

There is one useful technique is used in docbook called Profiling. Profiling is an easy way to personalize your contents for several target audience, for different operating systems and for different user groups or levels [kosek.cz¹⁹]. Profiling is a mechanism to describe the conditional text. Conditional text mean, you can specify the text in a single xml document that which text element should be include in the resulting document after docbook processing. This technique is useful when we need to produce different versions of the same document. In this case style sheets are used to include or exclude the marked text to satisfy the condition. So if we want to produce different versions of the same document that include or exclude some text portion then we don't need to make separate document for each version. We will just apply the profiling technique on it that specify that which portion of text should be include of excluded and just process the docbook document to get the desired output.

¹⁷ <http://en.wikipedia.org/wiki/XInclude>

¹⁸ http://docbook.theblog.ca/?page_id=6

¹⁹ <http://www.kosek.cz/xml/dboscon/profiling/frames.html>

This feature is normally used to produce different versions of a document for different audiences. That's where the term profiling comes in. You can create a document profiled for a particular audience. For example, software that runs on different platforms might require different installation instructions for each platform, but might otherwise be the same. You can create one version profiled for Linux customers and another profiled for Windows customers [sagehill.net²⁰].

Part of documents can be assigned to different target audiences:

- ✚ attribute os – target operating system
- ✚ attribute user level – target group of users
- ✚ attribute arch – target hardware architecture
- ✚ other application specific attributes can be used – conformance or role

Sample docbook document with profiling information is shown in the figure below.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD Docbook XML V4.1.2//EN"
    'http://www.oasis-open.org/docbook/xml/4.0/docbookx.dtd'>
<chapter>
<title>How to setup SGML catalogs</title>

<para>Many existing SGML tools are able to map public identifiers to
files on your local file system. Mapping is specified in so called
catalog file. List of catalog files to use is stored in environment
variable <envar>SGML_CATALOG_FILES</envar>.</para>

<Para os="unix">On Unix systems you can set this variable by invoking
command <command>export SGML_CATALOG_FILES=/usr/lib/catalog</command>
on command line. If you want maintain value of the variable between
sessions, place this command into startup file,
e.g. <filename>.profile</filename>.</para>

<para os="win">In Windows NT/2000 you can set environment variable by
issuing command <menuchoice><guimenu>Start</guimenu>
<guisubmenu>Settings</guisubmenu> <guisubmenu>Control
Pannel</guisubmenu>
<guimenuitem>System</guimenuitem></menuchoice>. Then select
<guilabel>Advanced</guilabel> card in the dialog box and click on the
<guibutton>Environment Variables...</guibutton> button. Using the
<guibutton>New</guibutton> button you can add new environment variable
```

²⁰ <http://www.sagehill.net/docbookxsl/Profiling.html>

into your system.</para>

</chapter>

In this example when we processed it, we will get only those contents which have specific parameter name. For example if we use os=UNIX by telling to the xsltproc then the resultant document will only contain document for Unix and if we use os=windows then the resultant document will contain text for windows, so it depends on the situation that what we need to produce. We will discuss more about xsltproc in the next chapter in detail. Docbook documents are normally processed by apply XSLT Style sheets. By applying profiling on docbook document, we need to perform two steps on it. First we have to filter out the contents of the document that which contents should be produced as an output and in the next step we have to process the docbook document by applying XSLT Style sheets as shown in the figure.

In this example you can see that we have a source Docbook document and we want to generate two different documents that will contain some of the different contents from each other. As we know profiling is a two step process, first we have applied profiling for target audience A and then we have applied profiling for target audience B to filter out the desired contents for each audience. Now we have both profiled document for target audience A and target audience B. After this we have applied XSLT Style sheets on each profiled document to generate desired output in different format e.g. to generate HTML, Compiled HTML (CHM) or PDF document as show in the figure below.

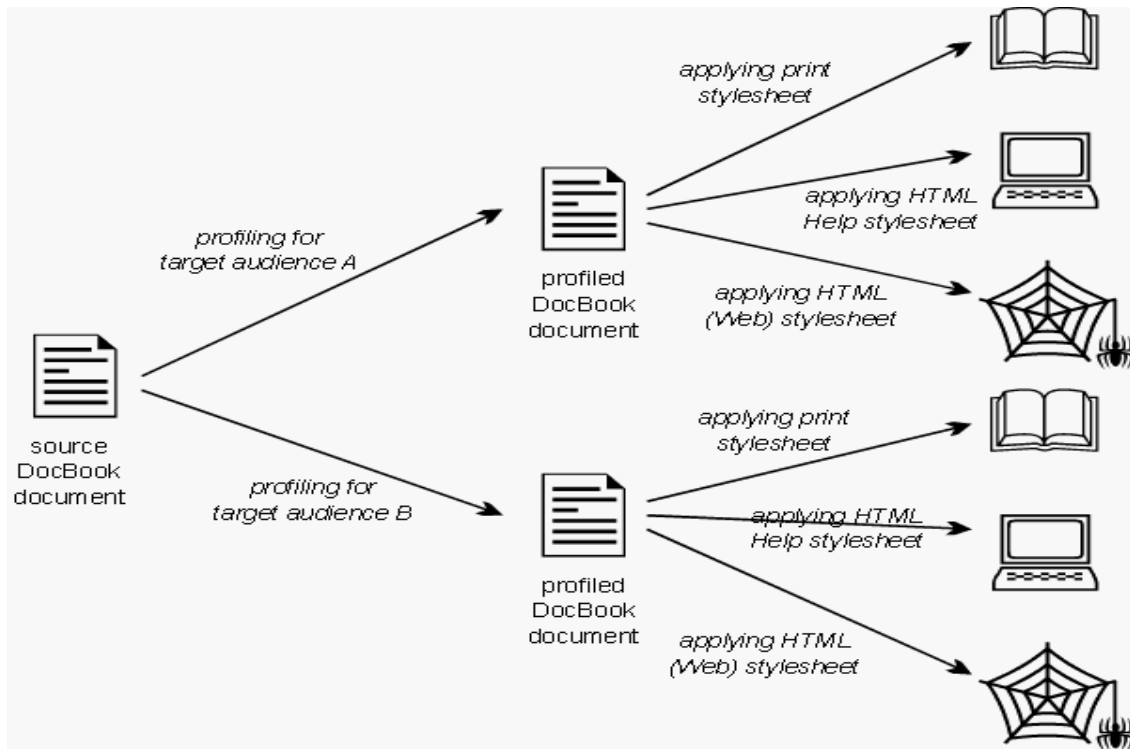


Figure 3-7 Source document to other formats process

[Jirka Kosek, 2001]

3.8 Docbook Disadvantages

The docbook is not without its problems. The setup environment of docbook is very complicated. Setting of environment variable and paths could be complicated for unskilled user. The separation between content and style can be somewhat complex to use, yet it is powerful. Although the style definition needs to be made only once, it is a non-trivial task and the outcome may not always be exactly as wanted. User need not learn tools that are used in docbook and its element and the way to produce final output.

The Formatting Objects processores (e.g. XEP, Antenna House) come mostly under commercial applications too. The development of open-source FO processores (e.g. FOP) is at the beginning. These FO processores are not conducive to formatting of complicated structures.

3.9 Latex

Latex is a document preparation system for the TEX typesetting program. We can produce publication-quality output with great accuracy and consistency. LATEX works on any

computer and produces industry-standard PS or PDF documents. It is available both in free (open-source) and commercial implementations. LATEX can be used for any kind of document, but it is especially suited to those with complex structure, repetitive formatting, mathematics¹, technical stability, and dimensional accuracy [tug.ctan.org²¹].

Latex is not a word processor. Latex encourages authors to more concentrate on the contents of document rather than its appearance and format. Latex is faster for producing documentation, but lacks the diverse transformation capabilities offered by Docbook XSL.

3.9.1 Features of Latex

Latex consists of a rich set of built-in-commands. As Latex support fully programming features that make complicated macros to easily define. Latex macros do take care of formatting decision for the author and one can use the default layout of Latex. If the default layout is not suitable for you then you can customize the layout of the document according to you your choice but while doing this some of the default setting will not be changed like

- ✚ Footnotes and marginal notes are automatically located on the page.
- ✚ Latex will automatically number sections and equations in a document.
- ✚ Latex makes it easy to control the actual width and format of columns in tables and to set paragraph entries in columns.

Latex is output device independent. The output of the Latex is device-independent (DVI) in a standard and well documented format. Filter programs then covert this file format to other required formats. Latex works the same way on all the system and produce the same output regardless of the type of the system. DVI files are interchangeable between the systems.

3.9.2 Basic Layout of Latex

Latex files consist of text of the particular document and commands. Everything is free-format and Latex doesn't care about spaces that how many spaces are in between words. It reads the input as a byte stream, looking for commands or blocks of text (words), separated by blanks, new-lines, tabs, or special symbols. Commands or text do not have to begin in a specific column or be on a line by themselves.

²¹ <http://tug.ctan.org/tex-archive/info/latex-veryshortguide/veryshortguide.pdf>

The syntax of the command starts with the backslash `\` followed by an alphabetic of arbitrary length, for example

```
\make
```

The backslash and the command name do not appear in the target output document because they are interpreted by Latex. Almost all Latex commands are to be written in lowercase letters only.

Some commands required parameters to set the margin or heading name etc, for example

```
\section{text of the heading}
```

Some commands have optional parameter; if you don't provide the parameter by yourself then default value is used for example

```
\document style[11pt]{...}
```

A few commands do not have alphabetic names, but rather a single non-alphabetic character after the backslash, for example:

This command `//` forces the start of a new line in the output

This command `\%` puts a percent sign in the output (`%` by itself has a special meaning).

The following "reserved" symbols are interpreted as special command names or arguments and do not appear in the output. You can get them typeset in your output file with special Latex commands in your input file.

```
# $ % & ~ _ ^ \ { }
```

Some of the command will always present in the Latex document like

```
\documentstyle{stylename}
```

```
\begin{document}
```

```
\end{document}
```

3.9.3 What is TeX

TeX is a low level markup and programming language to produce documentation precisely and consistently. It's a programming language as it uses if else structure to make calculations with it while compiling the document with TeX compiler.

3.9.4 BibTeX

Separate program works with Latex to produce formatted bibliographies and reference lists.

3.9.5 SlitEX

Separate program works with LaTeX to format text for slides or overhead transparencies [pangea.stanford.edu²²].

3.10 Advantages and disadvantages of LaTeX/TeX

Since Latex comprises a group of TeX commands, Latex document processing is essentially programming. You create a text file in Latex markup. The Latex macro reads this to produce the final document.

Clearly this has disadvantages in comparison with a WYSIWYG (What You See Is What You Get) program such as Openoffice.org Writer or Microsoft Word:

- ✚ You can't see the final result straight away.
- ✚ You need to know the necessary commands for Latex markup.
- ✚ It can sometimes be difficult to obtain a certain 'look'.

On the other hand, there are certain advantages to the markup language approach:

- ✚ The layout, fonts, tables and so on are consistent throughout.
- ✚ Mathematical formulae can be easily typeset.
- ✚ Indexes, footnotes and references are generated easily.
- ✚ You are forced to correctly structure your documents.

Latex document is a plain text file contains the contents of the documents and additional markup tags. We cannot see the final output of unfinished document because we need to compile all the document files with Latex or TeX macros.

²² <http://pangea.stanford.edu/computerinfo/unix/formatting/features.html>

Note that Latex is a collection of macros for Tex. So if we compile the TeX document with Latex compiler it will work perfectly but if we try to compile Latex document with TeX compiler then it will produce a lot of warning and errors. Latex natively supports DVI and PDF, but by using other software you can easily create PostScript, PNG, JPG, etc.

When Latex was developed, on that time the only format for Latex was DVI. After this pdf support was added with the name of pdf latex. So we can create pdf from both pdf latex and dvi2pdfm but the out is pretty good with pdf latex as compare to dvi2pdfm. Actually DVI is an old format and it also do not support hyperlinks in the document but the pdf latex support it perfectly.

The following diagram shows the relationships between the (La)TeX source code and all the formats you can create from it:

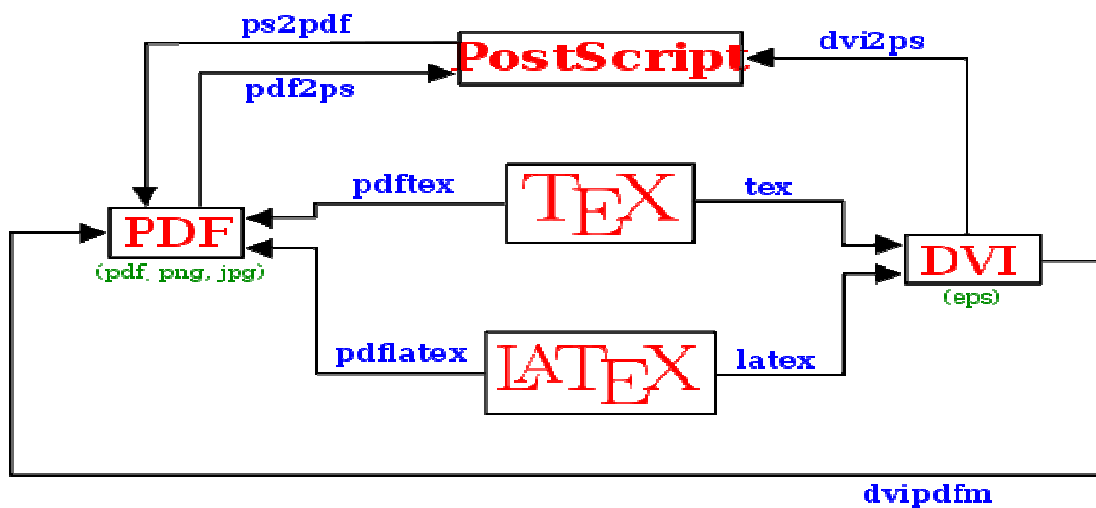


Figure 3-8 LATEX relationship with other formats

In this figure the red text denotes the file formats, blue text shows the commands to produce different file format outputs and the green text represents the image formats that are supported.

As we can see different paths in this diagram to get the desired output, some are shortest and some are longer to get the same output. If we use the longer path then the quality of target output will be decrease because each format conversion loses some pixel

values/information and if we use the shortest path then we can get the better quality of the documents [en.wikibooks.org²³].

A simple Latex template is shown in the figure below

```
% Example Latex document for GP111 - note % sign indicates a comment
\documentstyle[11pt]{article}
% Default margins are too wide all the way around. I reset them here
\setlength{\topmargin}{-.5in}
\setlength{\textheight}{9in}
\setlength{\oddsidemargin}{.125in}
\setlength{\textwidth}{6.25in}
\begin{document}
\title{LaTeX Typesetting By Example}
\author{Phil Farrell\
Stanford University School of Earth Sciences}
\renewcommand{\today}{November 2, 1994}
\maketitle
This article demonstrates a basic set of Latex formatting commands.
Compare the typeset output side-by-side with the input document.
\end{document}
```

The output of this document will show as below

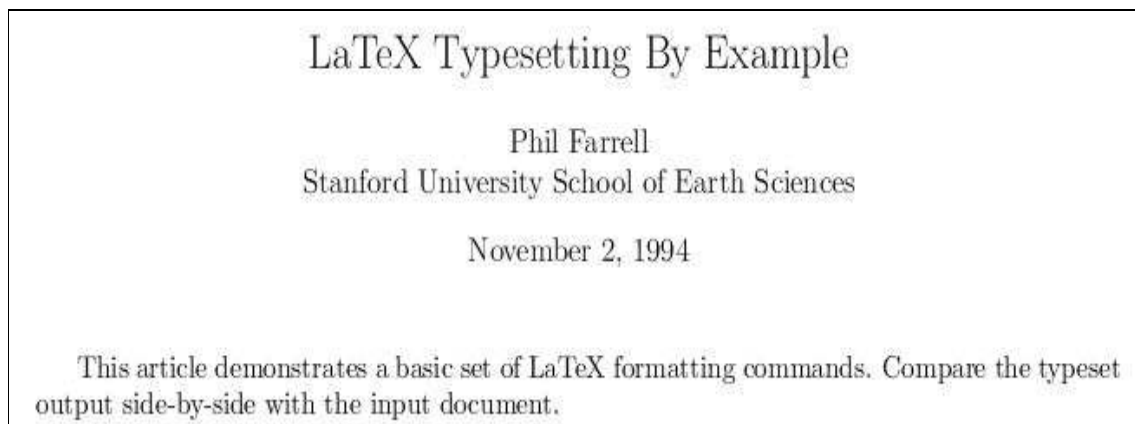


Figure 3-9 Latex output

²³ <http://en.wikibooks.org/wiki/LaTeX/Introduction>

3.11 Docbook usage over Latex

It's difficult to produce good html by using Latex. The standard tool that is used to produce html from latex is latex2html which produce atrocious and unnavigable html.

It's easier for those who are unexperienced with either latex or docbook to use xml docbook because if anyone have knowledge about html then it is very easy to understand xml. For example chapter tag in docbook will always begin with `<chapter>` and end with `</chapter>`.

XML has some built-in advantages. First, it makes it easy to check to see if the document is "well-formed".

The book structure is very simple; including external files via entities is pretty simple.

4 Document building & Scripting

Docbook is a very cool XML-based syntax that allows you to author documentation in a single format, and then run it through various processors to create your final documentation output. In this chapter we will show that how to process documentation using docbook help processor and then creating customizations for producing plain html for offline and online usage. We will also show that how to produce “compiled html (chm)” and pdf format from xml docbook source.

The number of technologies that will be used with xml docbook to produce documentation will be described below.

4.1 Xml Docbook

Docbook is a semantic markup language that is used for writing technical documentation. As a semantic language, Docbook enables its users to create document content in a presentation-neutral form that captures the logical structure of the content; that content can then be published in a variety of formats, including HTML, XHTML, EPUB, PDF, man pages and HTML Help, without requiring users to make any changes to the source [en.wiki.org²⁴].

We will write our documentation for C3Fire project in xml docbook format. The syntax of xml docbook is show as below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
"http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd">
<book>
<title>Simple DocBook Document</title>
<bookinfo>
<author>
<surname>Mustonen</surname>
<firstname>Juha</firstname>
<email>juham@ee.oulu.fi</email>
</author>
</bookinfo>
<chapter>
<title>First chapter</title>
<para>This document is really <emphasis>simple</emphasis>, but complex
```

²⁴ <http://en.wikipedia.org/wiki/DocBook>

enough to show how a short example:</para>

<example>

<title>Short example</title>

<programlisting language="python">

```
from datetime import datetime
```

```
#show current date and time
```

```
print(datetime.now().isoformat())
```

</programlisting>

</example>

</chapter>

</book>

4.2 XSLT Style sheet

XSL Transformation (XSLT) is a declarative xml based language for the transformation of xml document to other xml documents. Actually we cannot display the xml directly as web page or some other format. So that we have to transform it into HTML or XHTML so that it can be display as a web page or some other formats. When we transform xml to other format then the original document did not change rather than a new document based on the existing document is created. To convert xml to other formats, we need some processors according to our requirement but here we will use xsltproc processor that will be discussed later on.

The XSLT processing model is shown in the figure below.

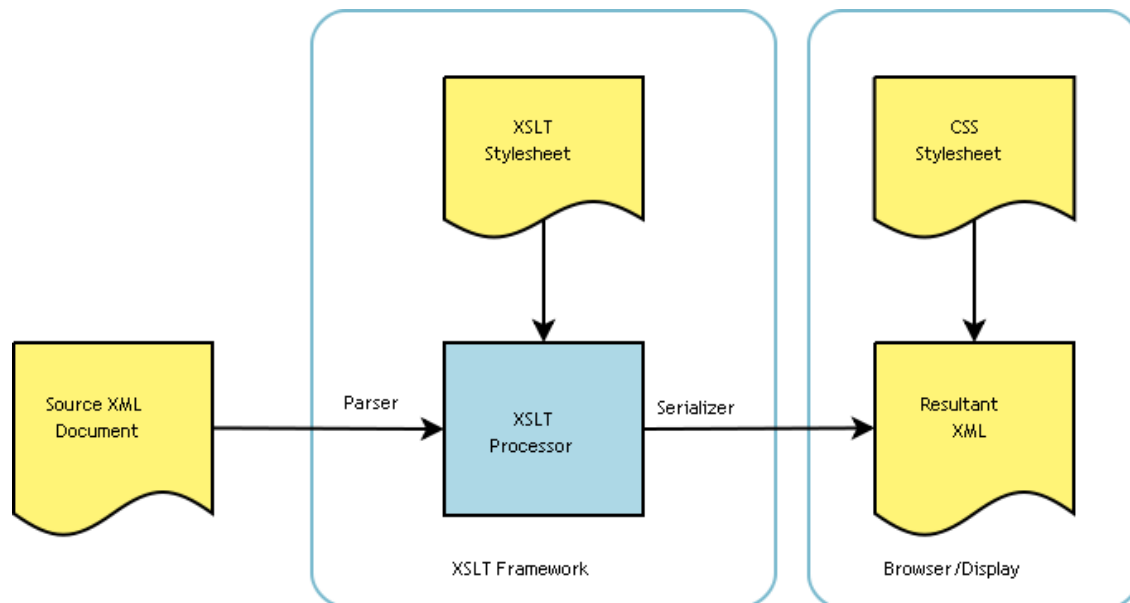


Figure 4-1 XSLT Processing Model

As we need to apply XSLT style sheet on xml document that's why XSLT Processor takes two inputs, one as a source xml document and another is XSLT style sheet. The XSLT style sheet contains contain template rules: instructions and other directives that guide the processor in the production of the output document.

The simple source xml is written as below

```
<?xml version="1.0" ?>
<persons>
  <person username="JS1">
    <name>John</name>
    <family-name>Smith</family-name>
  </person>
  <person username="MI1">
    <name>Morka</name>
    <family-name>Ismincius</family-name>
  </person>
</persons>
```

And now applying XSLT style sheet template on it to transform it into other xml document

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" indent="yes"/>

  <xsl:template match="/persons">
    <root>
      <xsl:apply-templates select="person"/>
    </root>
  </xsl:template>

  <xsl:template match="person">
    <name username="{@username}">
      <xsl:value-of select="name" />
    </name>
  </xsl:template>

</xsl:stylesheet>
```

²⁵ http://services.exeter.ac.uk/cmit/modules/meaningful_markup/webct/ch-xslt-intro.html

The format of new xml document will be

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <name username="JS1">John</name>
  <name username="MI1">Morka</name>
</root>
```

To transform XML to XHTML, first we need XSLT document as shown below

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns="http://www.w3.org/1999/xhtml">

  <xsl:output method="xml" indent="yes" encoding="UTF-8"/>

  <xsl:template match="/persons">
    <html>
      <head> <title>Testing XML Example</title> </head>
      <body>
        <h1>Persons</h1>
        <ul>
          <xsl:apply-templates select="person">
            <xsl:sort select="family-name" />
          </xsl:apply-templates>
        </ul>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="person">
    <li>
      <xsl:value-of select="family-name"/><xsl:text>, </xsl:text>
      <xsl:value-of select="name"/>
    </li>
  </xsl:template>

</xsl:stylesheet>
```

After transforming it into XHTML

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

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head> <title>Testing XML Example</title> </head>
  <body>
    <h1>Persons</h1>
    <ul>
      <li>Ismincius, Morka</li>
      <li>Smith, John</li>
    </ul>
  </body>
</html>
[en.wikipedia.org26]
```

4.3 4.3 CSS (Cascading Style Sheet)

Cascading style sheet is a mechanism to add style like color, font and spacing etc to the web documents to make them more attractive.

CSS preliminary separate the html content document from its presentation like color, font size and layout etc and gives more control to manage it. It enables multiple pages to share formatting style and give consistency among all the pages. So if we want to change the style of some html tags then we don't need to go on each particular tag to change its style, we will just do some change in the CSS document and all the pages that contain that particular tag will update their formatting according to CSS. So it's easy to control the formatting style of multiple pages by doing less effort.

For example, here you can see the html document

```
<html>
<head>
<link rel="stylesheet"
type="text/css" href="test.css" />
</head>

<body>

<h1>This header is 36 pt</h1>
<h2>This header is blue</h2>

<p>This paragraph has a left
margin of 50 pixels</p>

</body>
```

²⁶ <http://en.wikipedia.org/wiki/DocBook>

```
</html>
```

Here you can see the test.css template for the above html document

```
body {background-color: yellow}  
h1 {font-size: 36pt}  
h2 {color: blue}  
p {margin-left: 50px}
```

We can see that test.css contain body, h1, h2 and p tag with different attribute values. If we see the html document above, there is a link tag that contains the reference to the “test.css” style sheet. So the body color in html document will be yellow, the h2 header size will always be 36 pt either we use it in single html page or multiple and same for the other tags. If we want to change the appearance and layout of any tag, we will just do a smaller changing in CSS document.

4.4 XSL-FO

XSL-FO stands for Extensible Style sheet Language Formatting Objects. It is xml based and a formatting language. XSL-Fo is a markup language for XML document formatting which is most often used to generate PDFs. XSLT is a language for transforming xml documents and XSL-FO is a language for formatting xml documents.

Styling is both about transforming and formatting information. When the World Wide Web Consortium (W3C) made their first XSL Working Draft, it contained the language syntax for both transforming and formatting XML documents.

Later, the Working Group at W3C split the original draft into separate Recommendations [w3schools.com²⁷].

The general idea behind XSL-FO is not to write document in FO (formatting Object) but in XML. After writing required document in xml format then we need to use some xslt processor for example in our case we are using xsltproc, to convert it into XSL-FO format. Once the XSL-FO document is generated then we need FO processors to convert it into

²⁷ http://www.w3schools.com/xslfo/xslfo_intro.asp

readable, printable or both. The most common output of XSL-FO is a PDF file or as PS, but some FO processors can output to other formats like RTF files [en.wikipedia.org²⁸]

XSL-FO documents normally stored in files with .fo or .fob extensions. Each XSL-FO Page contains a number of Regions:

- ✚ region-body (the body of the page)
- ✚ region-before (the header of the page)
- ✚ region-after (the footer of the page)
- ✚ region-start (the left sidebar)
- ✚ region-end (the right sidebar)
- ✚ XSL-FO Regions contain Block areas.

A simple template of XSL-FO is shown in the figure below.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
  <fo:simple-page-master master-name="A4">
    <fo:region-body />
  </fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="A4">
  <fo:flow flow-name="xsl-region-body">
    <fo:block>Hello W3Schools</fo:block>
  </fo:flow>
</fo:page-sequence>
</fo:root>[w3schools.com29]
```

The output of this template will be simple showing the text “Hello W3Schools”.

²⁸ http://en.wikipedia.org/wiki/XSL_Formatting_Objects

²⁹ http://www.w3schools.com/xslfo/xslfo_intro.asp

4.5 XSLTPROC

XSLTPROC is a command line tool for applying XSLT style sheets on XML documents. It is a part of libxslt, the XSLT C library for GNOME. While it was developed as part of the GNOME project, it can operate independently of the GNOME desktop.

xsltproc is invoked from the command line with the name of the style sheet to be used followed by the name of the file or files to which the style sheet is to be applied. By default, output is to stdout. We can specify a file for output using the -o option. [linuxcommand.org].

We can use xsltproc to generate html pages, fo objects and compiled html pages from xml documents.

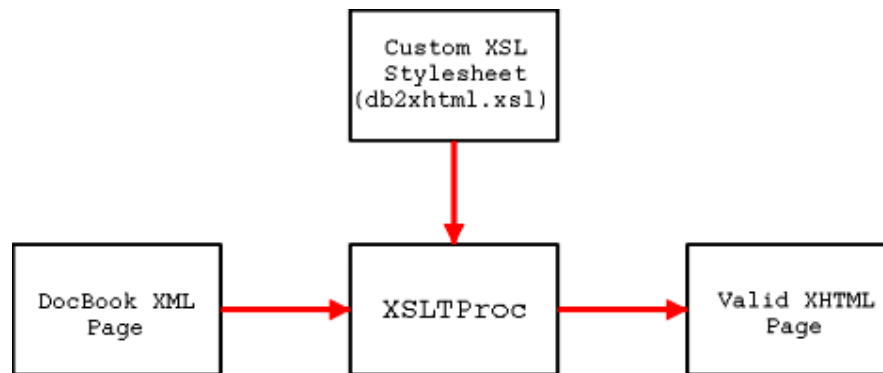



Figure 4-2 XSLTPROC Processing

[mirrors.bieringer.de³¹]

4.6 FOP

FOP is open source software under Apache Software License and abbreviated as Formatting Object Processor. FOP is a java application that converts XSL-FO files to pdf or other printable formats.

Apache FOP supports embedding a number of image formats in the XSL-FO (through the <fo:external-graphic> element). These include:

 SVG

³⁰ http://linuxcommand.org/man_pages/xsltproc1.html

³¹ <http://mirrors.bieringer.de/www.deepspace6.net/contribute/ds6-architecture.html>

- 🚦 PNG
- 🚦 Bitmap BMP
- 🚦 PostScript (as EPS)
- 🚦 JPEG
- 🚦 Some TIFF formats.

Apache FOP does not implement the <fo:float> element. External graphics objects are thus limited to being drawn inline or in a block with no wrapped text.

Apache FOP supports the following output formats:

- 🚦 PDF (best output support)
- 🚦 ASCII text file facsimile
- 🚦 PostScript
- 🚦 Direct printer output (PCL)
- 🚦 AFP
- 🚦 RTF
- 🚦 Java2D/AWT for display, printing, and page rendering to PNG and TIFF

In progress:

- 🚦 MIF
- 🚦 SVG [en.wikipedia.com³²]

The current release of FOP is 0.95 and the primary output target is pdf.

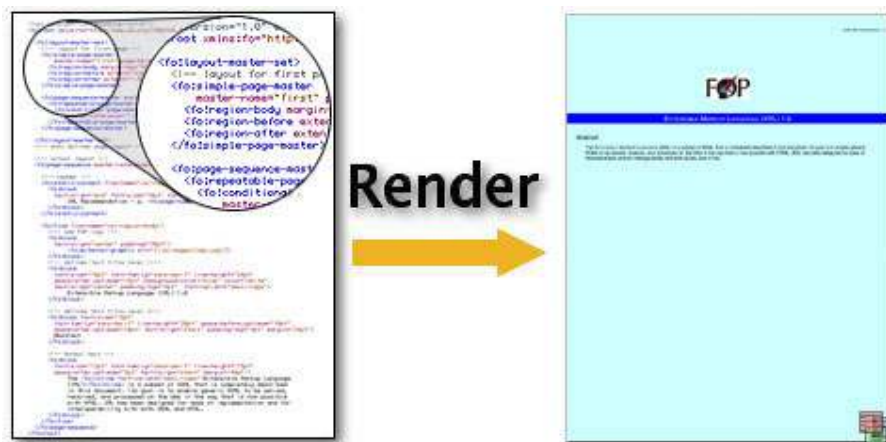


Figure 4-3 FOP Rendering

[xmlgraphics.apache.org³³]

³² http://en.wikipedia.org/wiki/Formatting_Objects_Processor

4.7 FOP Limitations

Development of FOP is in under process and it is not enough mature to work accordingly. As it works but have some limitations.

- ✚ dropped Text if inline FOs fall near page boundaries
- ✚ dropped lines if inline images and other elements cause page breaks
- ✚ fo:inline is basically ineffective, can only be used as property holder like fo:wrapper
- ✚ fo:character lacks basically all features you'd want to use it for
- ✚ no proper vertical alignment in lines, most of the values aren't recognized
- ✚ no handling of different font sizes within the same line
- ✚ linefeed-treatment not implemented
- ✚ whitespace-treatment not implemented
- ✚ improper line breaking
- ✚ improper hyphenation
- ✚ no implementation for the advanced hyphenation controls
- ✚ no reparenting for markers after retrieval
- ✚ retrieve-marker-position only really works to first-starting-within-page and occasionally for
- ✚ last-ending-within-page, everything else is pure coincidence
- ✚ retrieval of wrong markers in case page rendering is deferred
- ✚ footnotes are not broken correctly
- ✚ footnotes don't mix well with multi-column layout
- ✚ leaders may be misaligned (even after the recent fixes)
- ✚ conditional spaces and borders are not implemented
- ✚ margins are not properly implemented
- ✚ forced page breaking is not properly implemented
- ✚ the space resolution algorithm mandated by the spec is not
- ✚ Implemented
- ✚ collapsed table borders are not properly implemented [osdir.com³⁴]

³³ <http://xmlgraphics.apache.org/fop/>

³⁴ <http://osdir.com/ml/text.xml.fop.devel/2003-03/msg00111.html>

4.8 HTML Help Compiler




Microsoft HTML Help is a standard help system for windows platform. Authors can use Html Help to generate online help for software applications and contents for multimedia title or website etc. Developers can use the HTML Help API to program a host application or hook up context-sensitive help to an application. As an information delivery system, HTML Help is suited for a wide range of applications, including training guides, interactive books, and electronic newsletters, as well as help for software applications.

HTML Help offers some diverse advantages over standard HTML, such as the capability to employ a combined table of contents and index and the use of keywords for advanced hyper linking capability. The HTML Help compiler (part of the HTML Help Workshop) makes it possible to compress HTML, graphic, and other files into a relatively small compiled help (.chm) file, which can then be distributed with a software application, or downloaded from the Web. [msdn.microsoft.com³⁵]

4.9 Htmlhelp.hhp

When we process an xml docbook document with htmlhelp.xsl by using xsltproc, then the output of this process is a collection of HTML files and some non-HTML files. The HTML files are chunked HTML files with the navigational headers and footers removed. In fact, you can use all of the stylesheet parameters and customizations you would normally use when generating chunked HTML.

The non-html files are

-  Htmlhelp.hhp: this file is used for producing compiled html document (.chm) with help of Html Help Compiler.
-  Toc.hhc: this file is used to produce table of contents in a document.
-  Index.hhk: tis file is used to produce indexing in a document.

4.10 Docbook Processing Options

We can customize the htmlhelp.xsl style sheet to display different options in a final output document. With style sheet parameter we can control

³⁵ [http://msdn.microsoft.com/en-us/library/ms670169\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms670169(VS.85).aspx)

- ✚ The help window title, size and position.
- ✚ Whether the help menu appears.
- ✚ Which standard toolbar buttons are displayed
- ✚ Adding custom toolbar buttons.

4.10.1 Display the menu

If the value of `htmlhelp.show.menu` is set to 1 then the help application will have the standard menu at the top otherwise there will no menu displayed.

We can select which toolbar buttons are displayed in our Help application. Each parameter controls one button. Set its value to 1 to display the button, or to zero to hide it. The following table lists the button parameters.

Standard button name	Parameter
Hide/Show	<i>htmlhelp.button.hideshow</i>
Back	<i>htmlhelp.button.back</i>
Forward	<i>htmlhelp.button.forward</i>
Stop	<i>htmlhelp.button.stop</i>
Refresh	<i>htmlhelp.button.refresh</i>
Home	<i>htmlhelp.button.home</i>
Options	<i>htmlhelp.button.options</i>
Print	<i>htmlhelp.button.print</i>
Locate	<i>htmlhelp.button.locate</i>
Next	<i>htmlhelp.button.next</i>
Previous	<i>htmlhelp.button.previous</i>
Zoom	<i>htmlhelp.button.zoom</i>

Table 4.1 Parameter values

4.10.2 Custom buttons

We can add custom button in a help application that link to the external links. These buttons are called jump buttons, and each one has three parameters: to display the button, to label the button, and to identify the link for the button. The following table lists the parameters that control the custom buttons.

Custom button	Parameters	Description
Custom button 1	<i>htmlhelp.button.jump1</i>	When set to 1, display this button.

Custom button	Parameters	Description
	<i>htmlhelp.button.jump1.title</i>	Specify the text to show below the button.
	<i>htmlhelp.button.jump1.url</i>	Jump to this URL when pressed.
Custom button 2	<i>htmlhelp.button.jump2</i>	When set to 1, display this button.
	<i>htmlhelp.button.jump2.title</i>	Specify the text to show below the button.
	<i>htmlhelp.button.jump2.url</i>	Jump to this URL when pressed.

Table 4.2 Parameter values

4.10.3 Table of contents pan

We can customize the various aspects of table of contents window pane that appears to the left of the Help text. Some of them are described below.

Htmlhelp.hhc.width: Specifies the width of TOC (table of content).

Htmlhelp.hhc.section.depth: Specifies how many levels of nested sections to include in the TOC pane. Set to 5 by default, which means all section levels are included.

Htmlhelp.show.favorites: If set to 1, then a Favorites tab is added to the top of the TOC pane. The Favorites pane lets the reader save bookmarks into the Help file. The default is zero.

Htmlhelp.show.advanced.search: By enabling this feature, help application will have more advanced search options.

Htmlhelp.hhc.binary: If set to 1 (the default), it compiles the TOC into a binary form to improve performance. This setting also enables the Next and Previous buttons [[sagehill.net](http://www.sagehill.net)³⁶].

³⁶ <http://www.sagehill.net/docbookxsl/HtmlHelp.html>

5 Implementations

In this chapter we will discuss that how to setup and configure docbook environment to produce documentation. We will focus on that how

- ✚ To convert xml docbook to html
- ✚ To convert xml docbook to pdf
- ✚ To convert docbook to chm

Cygwin tool is used to configure docbook environment. Cygwin is a Linux-like environment for Windows. It consists of two parts:

- ✚ A DLL (cygwin1.dll) which acts as a Linux API emulation layer providing substantial Linux API functionality.
- ✚ A collection of tools which provide Linux look and feel.

Cygwin is not a way to run native Linux apps on Windows. You have to rebuild your application from source if you want it to run on Windows. Cygwin is free and open source tool that provides the command line interface for Microsoft Windows. Some of the tools included (in no particular order) are ssh, cvs, gcc, make, touch, scp, more, less, cat, bash, perl, python, and many, many others [cygwin.com³⁷].

5.1 Installation of cygwin

1. Download and run "setup.exe" from <http://www.cygwin.com/setup.exe>
2. Accept all defaults, clicking "Next" until you get to "Choose a installation directory"

³⁷ <http://www.cygwin.com/>

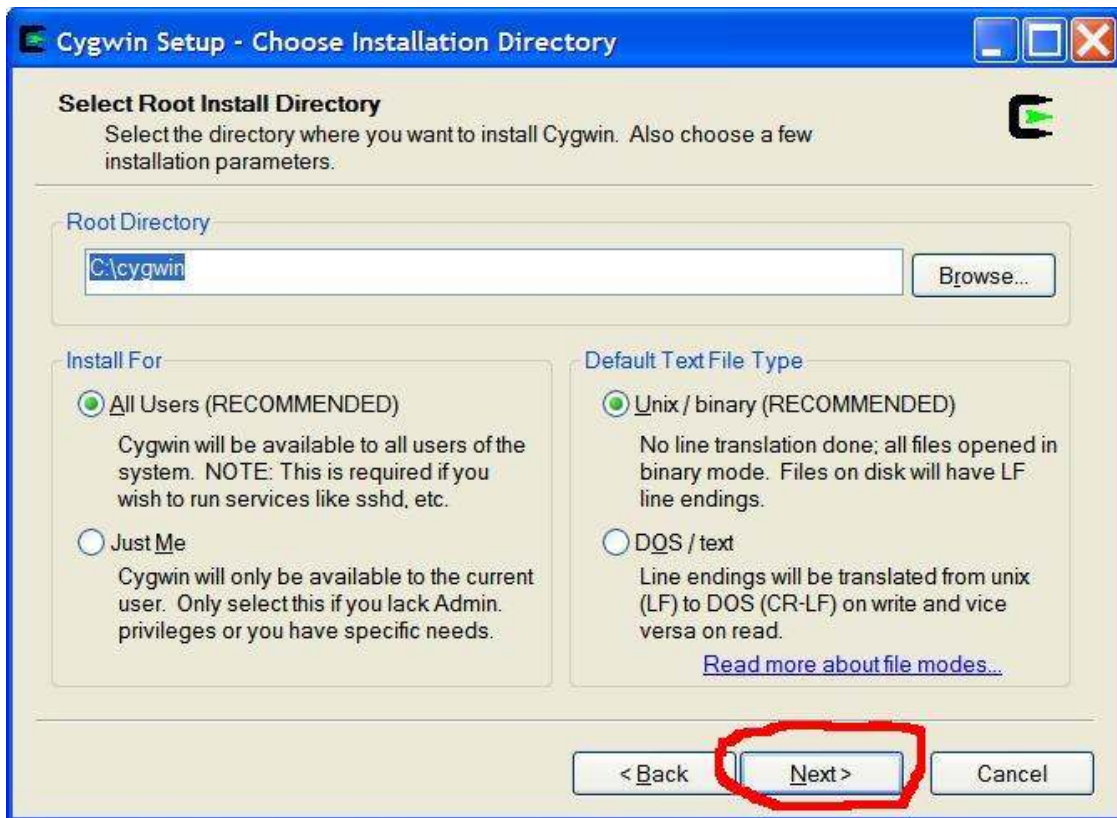


Figure 5-1 Installation directory

In this window you have to select DOS/text choice for default text file type.

3. Accept all defaults, clicking "Next" until you get to "Choose a installation directory"

4. Choose a download site. "http://mirror.mcs.anl.gov" is a good choice.

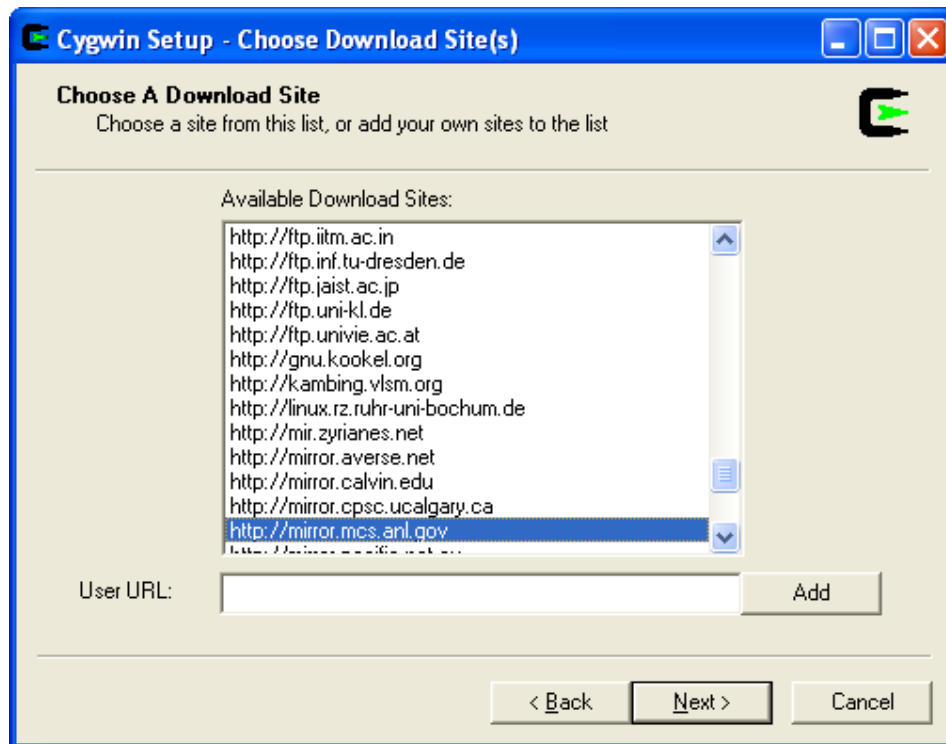


Figure 5-2 Choose download site

Click next and now the installer grabs a list of available packages, and displays it in this rather clumsy way:

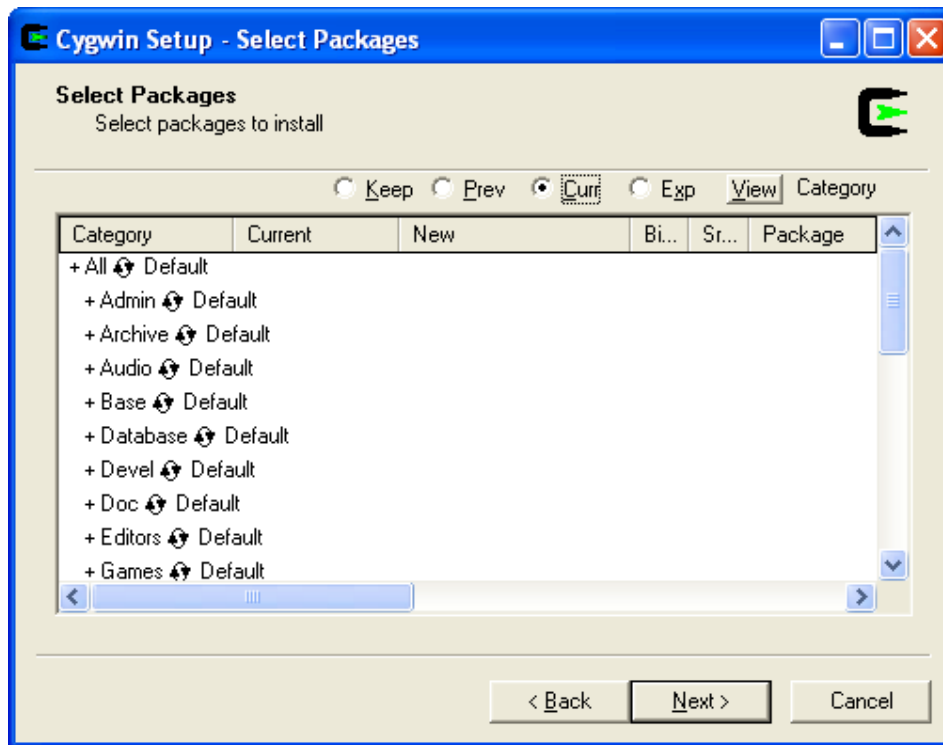


Figure 5-3 Select Packages

Click on plus sign to change the value to install all and click next. This will install everything you need to run xml docbook.

Now open up your bash command line and type (you can do this via Start > Programs > Cygwin > Cygwin Bash Shell):

After installing cygwin, now it's time to test cygwin that it is working fine or not. Open the cygwin base shell and write

```
Xsltproc -version
```

You should get message like

```
$ xsltproc -version
```

```
Using libxml 20423, libxslt 10013 and libexslt 705
```

```
xsltproc was compiled against libxml 20417, libxslt 10013 and libexslt 705
```

```
libxslt 10013 was compiled against libxml 20417
```

```
libexslt 705 was compiled against libxml 20417
```


Next you need to download docbook xsl style sheet. If you don't find it under C:\cygwin\usr\share\docbook-xsl then you can download it from sourceforge.net and unzip it at C:\cygwin\usr\share\.

Now we can write simple test program. Create a simple.xml and write

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

<book>
  <title>Simple Book</title>
  <titleabbrev>Simple</titleabbrev>

  <preface><title>Introduction</title>
  <para>
    Hello! Here's an introduction!
  </para>
</preface>

  <chapter><title>On Foo's</title>
  <para>
    Stuff about Foo's goes here.
  </para>
</chapter>

  <chapter><title>On Bars's</title>
  <para>
    Stuff about Bars's goes here.
  </para>
</chapter>
</book>
```

In this example you can see lot of things like dtd, book, title, preface, Para, and chapter etc. DTD is used for the verification of xml docbook. Don't forget to add <?xml version="1.0"?> on top of each xml docbook file otherwise you will get lot of errors about xml parsing. The basic building blocks that are needed to organize various sections in a docbook are book, chapter and section etc.

Chapters can nest under a book, and sections can nest under a chapter or another section. Paragraphs are wrapped with the <Para> tags and can occur most anywhere you want. You can include section tag under section tag for subsections etc.

5.2 Docbook to html and chm

First of all put simple.xml under C:\cygwin\home\

Use xsltproc to generate html files by writing this command

```
xsltproc --nonet /usr/share/docbook-xsl/htmlhelp/htmlhelp.xsl simple.xml
```

This will produce

```
$ xsltproc --nonet /usr/share/docbook-xsl/htmlhelp/htmlhelp.xsl simple.xml
Attempt to load network entity http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd
Writing pr01.html for preface
Writing ch01.html for chapter
Writing ch02.html for chapter
Writing index.html for book
Writing htmlhelp.hhp
Writing toc.hhc
```

You can see the different files have been created under C:\cygwin\home\

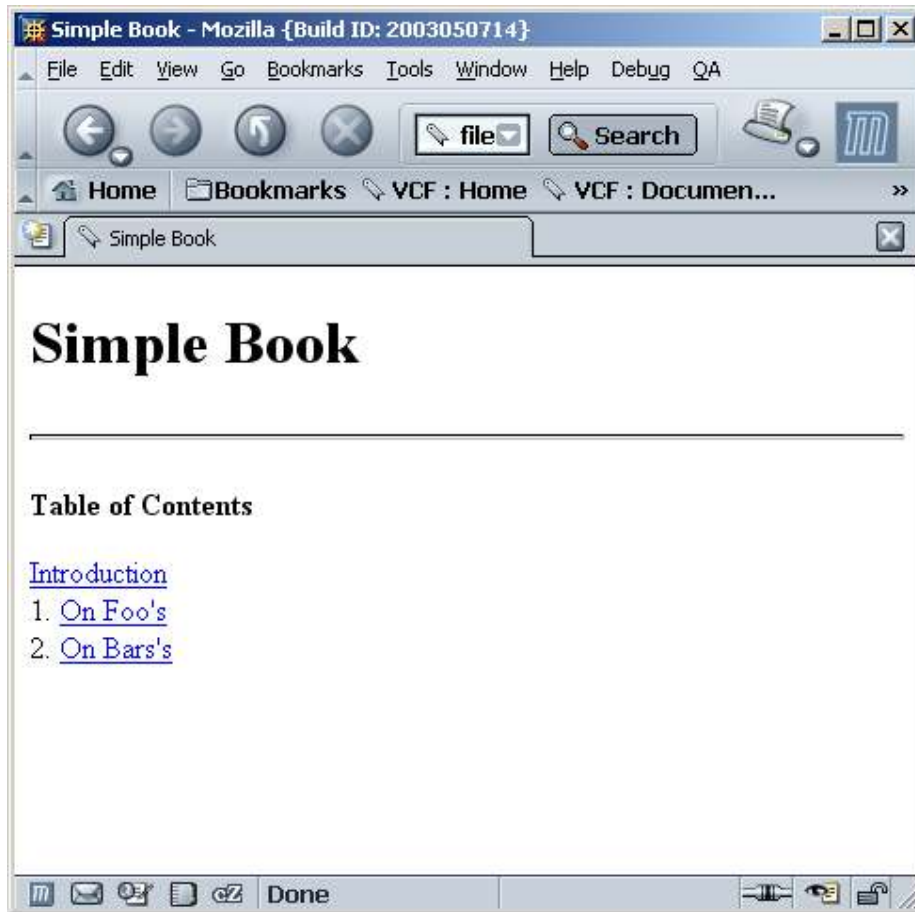


Figure 5-4 HTML Output

The `-nonet` option tells to the `xsltproc` that not to connect via the network to verify the xml docbook with dtd.

As you can see that we have added two chapters in the same xml file, we can break up it into multiple files for easy handling. For example we make two separate chapter files, `chap1.xml`

```
<chapter><title>On Foo's</title>
<para>
  Stuff about Foo's goes here.
</para>
</chapter>
```

And `chap2.xml`

```
<chapter><title>On Bars's</title>
<para>
  Stuff about Bars's goes here.
```

```
</para>  
</chapter>
```

Now we have to modify simple.xml

```
<?xml version="1.0"?>  
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"  
  "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd" [  
  <!ENTITY chap1 SYSTEM "chap1.xml">  
  <!ENTITY chap2 SYSTEM "chap2.xml">  
>
```

The use of the <!ENTITY> tags creates entities named chap1 and chap2. Using them automatically includes their contents in to the simple.xml file, so be careful to not put the <?xml?> preprocessor tags in the included files (chap1.xml and chap2.xml respectively). We can verify that all this works by running xsltproc again:

```
xsltproc -nonet /usr/share/docbook-xsl/htmlhelp/htmlhelp.xsl simple.xml
```

Again we will get the same output as above.

Now we can generate the chm. It's really easy to produce it, just call the command line html help compiler and pass the htmlhelp.hhp file that is already generated by executing xsltproc.

You can see out chm output below

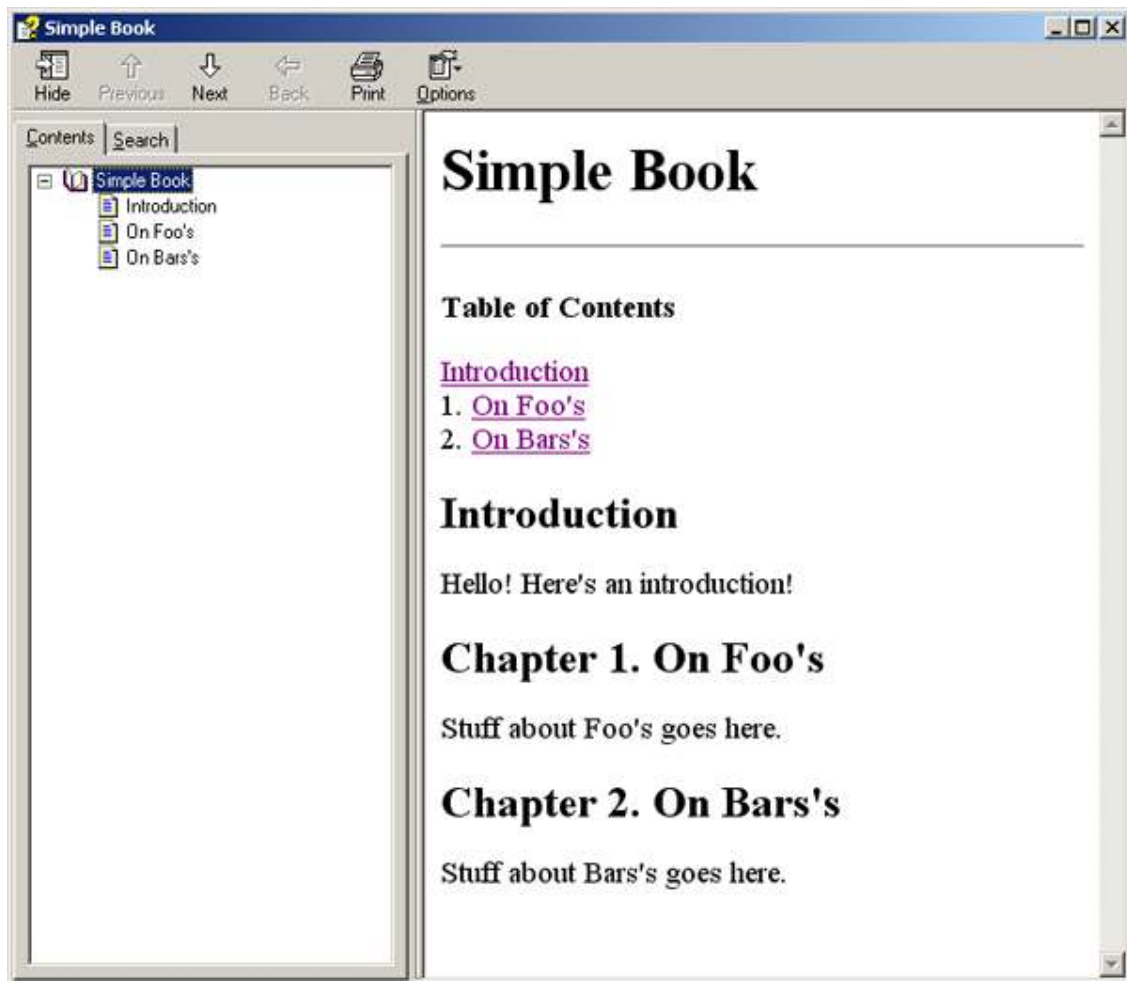


Figure 5-5 CHM Output

5.3 Adding an index

You can see above the chm output figure, there is no indexing support, we can add it by using `<indexterm>` in both chap1.xml

```
<chapter><title>On Foo's</title>
<para><indexterm><primary>About Foo's</primary></indexterm>
Stuff about Foo's goes here.
</para>
</chapter>
```

And chap2.xml

```
<chapter><title>On Bars's</title>
<para><indexterm><primary>About Bar's</primary></indexterm>
Stuff about Bars's goes here.
```

```
</para>  
</chapter>
```

Again compile simple.xml with xsltproc to generate htmlhelp.hhp and then run html help compiler hhc htmlhelp.hhp to generate chm. This time you can see the indexing support in it as shown below.

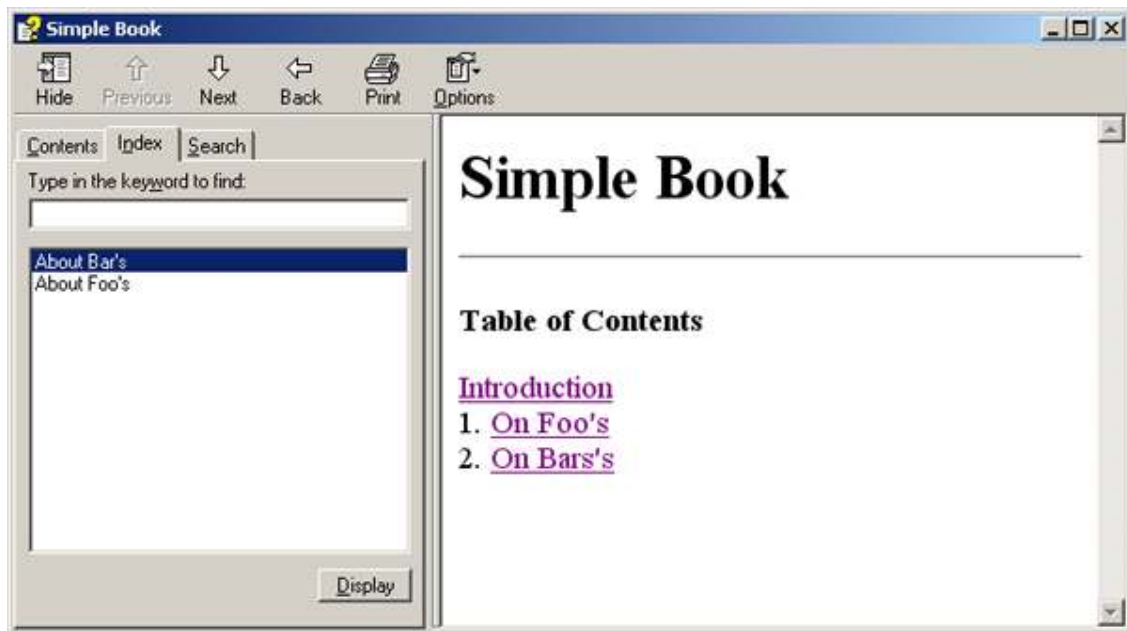


Figure 5-6 CHM with Index

5.4 How to produce single html file

When we compile an xml docbook document with the help of xsltproc then we get multiple html files. One of the features of docbook is to create single html file to write an article. To achieve this, we can use the same xml file that is simple.xml.

By using xml docbook we can use single documentation source to generate multiple outputs but this time we will use different XSL Style sheet to produce single html file as shown below.

```
xsltproc -nonet /usr/share/docbook-xsl/htmlhelp/docbook.xsl simple.xml
```

By using this style sheet, we will simply dump the output to stdout format but we can dump it into html format.

```
xsltproc -nonet /usr/share/docbook-xsl/htmlhelp/htmlhelp.xsl simple.xml >simple.html
```

Now we can see the output in a single html file

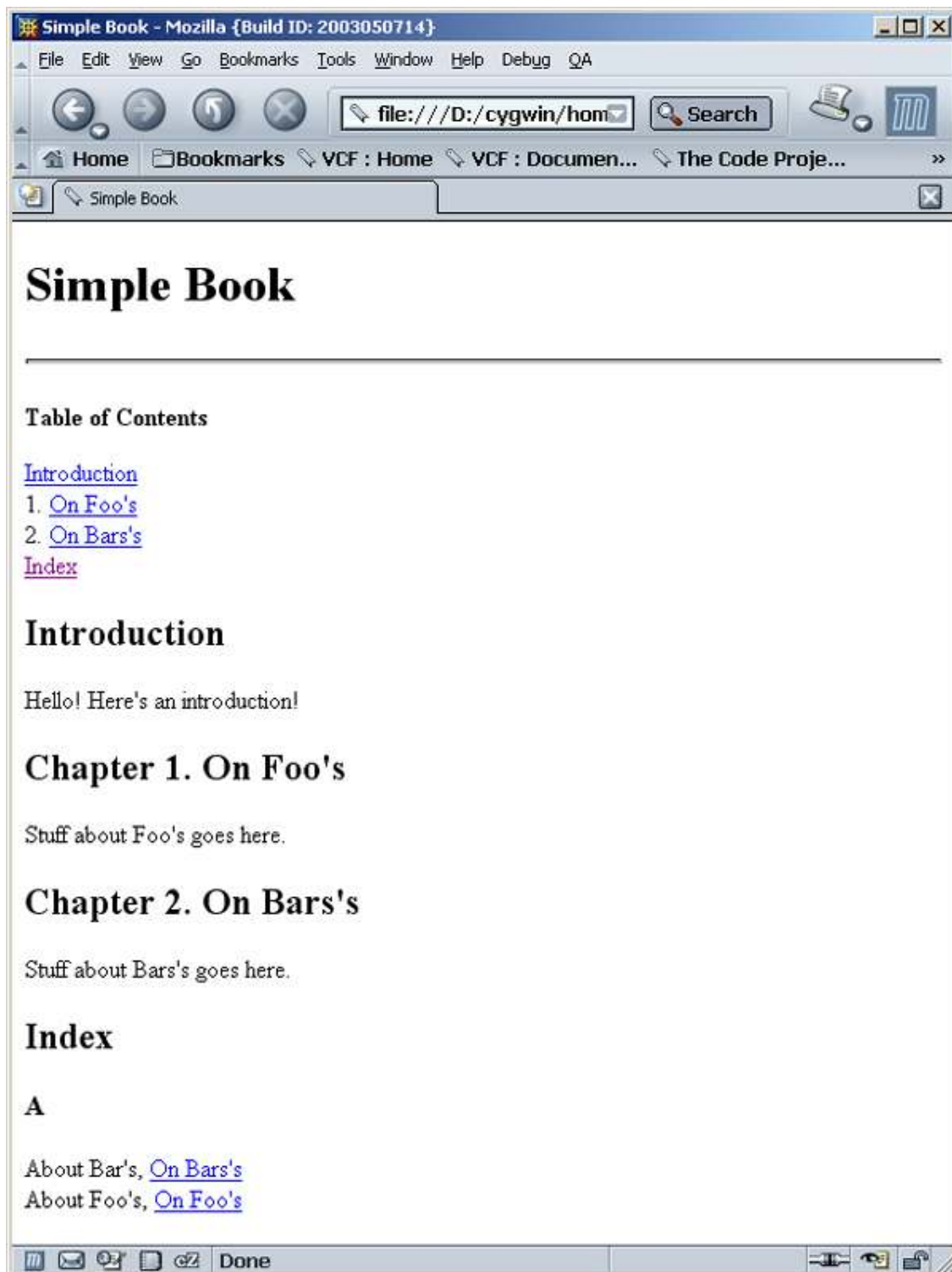


Figure 5-7 Single HTML Output

With Docbook we can describe various pieces of information about the documentation, such as the author, legal notices, book version, and copyright(s) notice. We start this by adding a `<bookinfo>` tag.

By adding these tags in `simple.xml` we will have

```
<book>
<bookinfo>
  <legalnotice>
    <para>
      Here a short legal notice: You agree that all your base
      belongs to me!
    </para>
  </legalnotice>
  <author>
    <firstname>Bob</firstname>
    <surname>Grey</surname>
  </author>
  <copyright>
    <year>2003</year>
    <year>2021</year>
    <holder>Pennywise the Clown</holder>
  </copyright>
</bookinfo>
<title>Simple Book</title>

<!-- ... -->

</book>
```

5.5 Tables

Table is very important in any documentation and docbook fully support it.

```
<chapter><title>On Foo's</title>
<para><indexterm><primary>About Foo's</primary></indexterm>
  Stuff about Foo's goes here.
</para>
<para>And now for some data in a table:
  <table frame="none" pgwide="1">
    <tgroup cols="3" align="left" colsep="1" rowsep="1">
      <thead>
        <row>
          <entry>Column 1</entry>
          <entry>Column 2</entry>
```



```

    <entry>Column 3</entry>
  </row>
</thead>
<tbody>
  <row>
    <entry>Heres</entry>
    <entry>A</entry>
    <entry>Row entry!</entry>
  </row>
</tbody>
</tgroup>
</table>
</para>
</chapter>

```

<tgroup> tag is very important otherwise table will not rendered properly. After processing it the output will be

Chapter 1. On Foo's

Stuff about Foo's goes here.

And now for some data in a table:

Table 1.1.

Column 1	Column 2	Column 3
Heres	A	Row entry!

You have to specify the number of rows and columns before execution otherwise it will produce errors.

If under <thead> tag, we specify that one row will contain 3 columns and then under <tbody> we specify that one row will contain more than 3 columns then you will get error about it, so be careful when using table in it.

Docbook automatically add number to the tables for example if the chapter 4 contains 3 tables then it will be numbered like Table 4.1, Table 4.2 and Table 4.3 etc. We can turnoff this feature by using <informaltable> tag then we will get simple table with no numbering.

5.6 Links

Docbook fully support links to other things. <ulink> tag is used to link to external URL's.

```
<chapter><title>On Bars's</title>
<para><indexterm><primary>About Bar's</primary></indexterm>
Stuff about Bars's goes here.
</para>
<para>To learn more about the wonderful world of Bar's look
<ulink url="http://www.google.com/search?q=Bars">
here
</ulink>
</para>
</chapter>
```

The url attribute is used to specify a link to load.

5.7 Graphics

Graphics are very important in documentation. Docbook provide <graphic> tag to add graphics in a document

```
<chapter><title>On Bars's</title>
<para><indexterm><primary>About Bar's</primary></indexterm>
Stuff about Bars's goes here.
</para>
<para>To learn more about the wonderful world of Bar's look
<ulink url="http://www.google.com/search?q=Bars">
here
</ulink>
</para>
<para>
Don't forget: Graphics are important!
<graphic fileref="smiley.bmp"></graphic>
</para>
</chapter>
```

5.8 Figures

Figures are also very important in documentation. We can add automatically number figured just like table numbers by using <figure> tag and in the same way we can use without numbered figures by using <informalfigure>

```
<chapter><title>On Bars's</title>
```

```

<para><indexterm><primary>About Bar's</primary></indexterm>
Stuff about Bars's goes here.
</para>
<para>To learn more about the wonderful world of Bar's look
  <ulink url="http://www.google.com/search?q=Bars">
  here
  </ulink>
</para>
<para>
<figure><title>Hierarchic organisation</title>
<mediaobject>
  <imageobject>
    <imagedata fileref="C:\cygwin\home\Asif\C3Fire\gfx\hierarchic.gif"/>
  </imageobject>
</mediaobject>
</figure>

</para>
</chapter>

```

5.9 Special formatting

<emphasis> tag is used for special formatting like

```

<para>
Not only are Foo's important to proper software development, but they are
critical to understanding the synergistic relationship between Neo
  <emphasis>and</emphasis> Trinity.
</para>

```

It will produce output like

Not only are Foo's important to proper software development, but they are critical to understanding the synergistic relationship between Neo *and* Trinity.

5.10 Plain text formatting

Some time we need to write text as it is as we get it from some command line source. For this purpose docbook provide <programlisting> tag as show below

```

<para>
  Here's an example of a code listing:
  <programlisting>
  int foo = 12 * 23;

```

```
multiply_endlessly( foo );  
  
</programlisting>  
</para>
```

Some time we have to use some special characters in documentation for example if we use < or > character in xml docbook document and compile it with xsltproc then there will be errors about parsing. For this purpose docbook provide <CDATA> tag. Xsltproc ignore all the character in between <CDATA> tag as shown below.

```
<para>  
Here's an example of a code listing:  
<programlisting>  
<![CDATA[  
int foo = 12 * 23;  
std::vector<int> vec;  
vec.push_back( foo );  
  
]]>  
</programlisting>  
</para>
```

5.11 Customizing the style sheets

We can customize the XSL style sheets to control header, footer and CSS etc. For this purpose first of all we make a simple.xml style sheet that will be inherited from htmlhelp.xml.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"  
version="1.0">  
<xsl:import href="/usr/share/docbook-xsl/htmlhelp/htmlhelp.xml "/>  
</xsl:stylesheet>
```

The import tag is used to tell the process to include or import the URL that is referred by the href attribute. Now we can run xsltproc like

```
xsltproc --nonet simple.xml simple.xml
```

By using simple.xml, we have replaced the default html help style sheet. To generate legal info as a separate html page we have to include

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
  <xsl:import href="/usr/share/docbook-xsl/htmlhelp/htmlhelp.xsl "/>
  <xsl:param name="generate.legalnotice.link" select="1"/>
</xsl:stylesheet>
```

By using `<xsl:param>` tag we can specify a parameter's value. In this case we have specified that `generate.legalnotice.link` value is 1 or true.

For the addition of next and back navigation links at the bottom of each page, we need to include parameter

```
<xsl:param name="suppress.navigation" select="0"/>
```

Docbook provide the facility of using standard graphics for different tags. For example when we use some specific tag then docbook automatically load graphics according to that tag to represent it with graphic sign. For using standard image, a specific image directory is needed otherwise broken link will be shown on the page. If you don't want to use standard graphics then don't add this parameter in xsl style sheet.

```
<xsl:param name="admin.graphics" select="1"/>
<xsl:param name="admin.graphics.path">gfx/</xsl:param>
```

You can see that `gfx` is a directory that contains images to be used with the specific docbook tags and you have to put that directory on the same level of other xml files.

This will be the output of above tags and you can see a standard image with "Note". Actually note is a docbook tag to write something as a note.



Note

Not only can Docbook do graphics, but it can handle notes as well! Isn't that just cool?

5.12 CSS Support

Docbook fully support CSS (Cascading Style Sheets) to control the layout of the contents. Here we will describe how to add CSS support in docbook. First of all we have to design the

CSS that will be applied on the contents of the docbook that how it should be layout and for its contents presentation. You can see the simple.css below

```
a, body, div, table, td
{
  font-family: Verdana, Geneva, Arial, Helvetica, sans-serif;
}
body
{
  background-color: #DDDDDD;
  color: #000000;
  margin: 0px 0px 0px 0px;
  padding : 0px 5px 5px 5px;
  border: 1px solid #000000;
}
```

We can set different style, color, font etc with the help of CSS. You see that background color, margin, body, div color and font etc. after completing the CSS template , now it's time to add it in XSL Style sheet.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
  <xsl:import href="D:/docbook-xsl-1.60.1/htmlhelp/htmlhelp.xsl"/>
  <xsl:param name="generate.legalnotice.link" select="1"/>
  <xsl:param name="suppress.navigation" select="0"/>
  <xsl:param name="admon.graphics" select="1"/>
  <xsl:param name="admon.graphics.path">gfx/</xsl:param>
  <xsl:param name="html.stylesheet" select="simple.css"/>
</xsl:stylesheet>
```

This will instruct the docbook to use CSS and apply its effect on html.

We can do lot of modification using CSS, for example adding header and footer etc. We can control the name of .chm file by add the parameter in a XSL style sheet. Previously the default name of chm file is generated but now the chm file name will be simple.chm every time.

```
<xsl:param name="htmlhelp.chm" select="simple.chm"/>
```

Next, we'll control how deep various sections should be shown. Each time you nest a <section> tag inside another <section> tag it causes a new level of numbering (i.e. 1.1, and its first child 1.1.1). The top of each page can display a certain amount of the pages sections and sub sections in a TOC style set of links. By adjusting the style sheet, we can control how deep this goes.

```
<xsl:param name="toc.section.depth" select="4"/>
```

5.13 Custom header and footers

We can add custom header and footer in all pages by using <xsl:template> tag. We can use html tag in it. For header we have to add the below tag in XSLT style sheet.

```
<xsl:template name="user.header.navigation">
  <hr>
  <p>Documentation by ACME Data Inc. No Coyote's allowed.</p>
  <hr>
</xsl:template>
```

If we compile this code with xsltproc, we will get lot of error about mismatch tag about valid xml because xsltproc was accepting xml tag instead of html tag

```
$ xsltproc --nonet simple.xsl simple.xml
simple.xsl:22: error: Opening and ending tag mismatch: hr and xsl:template
  </xsl:template>
    ^
simple.xsl:23: error: Opening and ending tag mismatch: hr and xsl:stylesheet
</xsl:stylesheet>
    ^
simple.xsl:23: error: Premature end of data in tag xsl:template
</xsl:stylesheet>
    ^
simple.xsl:23: error: Premature end of data in tag xsl:stylesheet
</xsl:stylesheet>
    ^
cannot parse simple.xsl
```

But we can use all the html ending tags with back slash, for example <hr></hr> which is even acceptable to all browsers.

Adding custom footer is same as header as shown below

```
<xsl:template name="user.header.navigation">
  <hr></hr>
  <p>Documentation by ACME Data Inc. No Coyote's allowed.</p>
  <hr></hr>
</xsl:template>

<xsl:template name="user.footer.navigation">
  <hr></hr>
  <p>The Road Runner was here - All Wrongs Reserved.</p>
  <hr></hr>
</xsl:template>
```

Header can be shown in the figure like

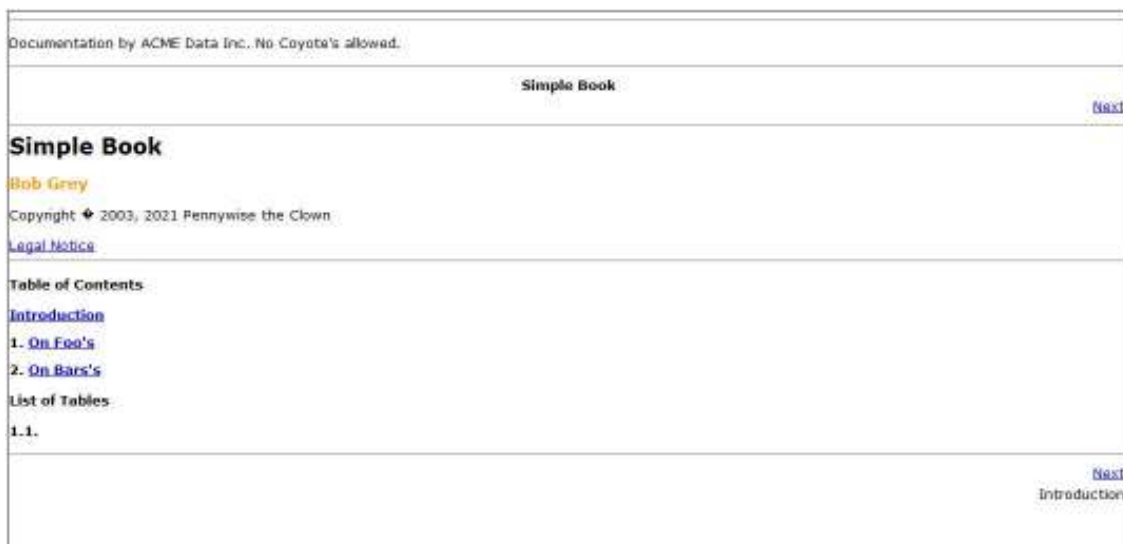


Figure 5-8 HTML Header

5.14 Docbook versioning

We can use profiling for docbook versioning but there is another simple method for making different versions of same document. First of adjust the xml docbook files for a new version that what chapters, sections you want to include in it and save all the xml files.

In our case, we will add "Version BOOK_VERSION" in a simple.xml. The text "BOOK_VERSION" will be used as a place holder for the actual version that we will store in a text file.

First of all we will write some text 1.2.1 as a version number and store it in a file named version.txt. Now we will define one variable to get the value from the version.txt

```
$ bk_ver=`cat version.txt`
```

After this we will use “SED”, a UNIX program that perform find and replace operation. So it will find BOOK_VERSION text in all the documents and replace the version number that is written in version.txt by using bk_ver variable as shown below.

```
sed "s?BOOK_VERSION?$bk_ver?g" simple.xml > simple.xml.tmp
```

As you can see that we are dumping the simple.xml into new tmp file. After this just remove the tmp extension and rename it with valid xml extension. Now this new file contains the version number in all places where we have written BOOK_VERSION text. Just run this new xml file with xsltproc and now we will get new documentation version.

5.15 Docbook to pdf

Docbook to pdf is a two step process, first we have to convert xml file to FO (Formatting Object) by using xsltproc and then we have to use FOP to convert FO file to finished pdf.

Here is the complete process that how to do. First of all convert simple.xml to simple.fo by using

```
xsltproc -nonet /usr/share/docbook-xsl/fo/docbook.xsl simple.xml > simple.fo
```

After this we have to use FO processor to convert simple.fo to finished pdf. If FOP exists under docbook-xsl directory then we will write the command like

```
/usr/share/docbook-xsl/fop/fop simple.fo -pdf simple.pdf
```

This will produce pdf from xml docbook file.

6 Conclusion and Final Work

6.1 C3Fire Problems

During the conversion of C3Fire Project documentation into different formats, we have found some problems in it.

When we use tables in XML Docbook to convert it into html and chm then it works fine but when we try to convert it into pdf using FOP then it produces some errors about table columns. Actually we have to specify the table columns before the rendering because the FOP is immature and cannot handle table without pre-defined number of columns in a table. If we don't specify number of columns in the case of html or chm conversion then it works as expected.

Another problem is when we convert xml docbook to chm. When we use "graphic" or "figure" tag then we should use the path like
fileref=file:///C:/cygwin/home/Asif/gfx/FireSimulation-BurnOut-2.gif Otherwise no figure will be displayed in chm. We should not use the path like
fileref="C:\cygwin\home\Asif\C3Fire\image-2\simulation-1.gif".

6.2 Conclusion

We have tested and setup an environment for generating documentation by using XML Docbook. We have concluded that there is lot of advantages to use docbook over other traditional software documentation tools. Docbook provides single sourcing technique, to covert document into several other formats from the centralized location with minimum effort. We don't need to buy costly commercial tools for generating many other formats. Docbook provides easy handling of documentation versioning system by using profiling technique. We have concluded that by using docbook, the author don't need to concentrate on formatting of document because formatting and style thing is stored in separate files which is applied during the compilation of document. So the author feels comfortable to concentrate only on the actual document contents. When the writing thing is finished then we can apply different stylesheet to get the desired formatted output from the single source and the original document source is not affected by applying different styles and formats on it. Otherwise if we use some other traditional software documentation tools like Microsoft word, the author need to concentrate more on formatting rather than the document

contents that's why it's time consuming and maintenance is not easy. Docbook provides modularity which mean that content files are separate from the style sheets, pictures etc. so the maintenance and editing is very easy in it.

6.3 Future Work

In the future work, we can enhance the software documentation by using docbook. Actually it's bit hard to maintain the environment for docbook to produce software documentation. And the docbook is based on xml, so the author of the docbook should be good in xml as well to write documentation properly and the document conversion is also an error prone process for example if we write an xml document that is needed to convert into html and pdf then the xml document should be written in a format that can be easily converted into desired output. Actually FOP is not enough mature to process the same xml document that is written for generating pdf and html. As we have discussed the table formatting problem with FOP.

So we need to make an application that can generate xml docbook scripting language automatically to make easiness for the author who is not good in xml etc or it can make documentation writing process speedily. We can include the functionality of converting the source document into different formats by using simple button click. In this way we can handle different xml files and directories easily. We can accept xml and style sheet documents from external sources and by using the brows option in an application can save the those files in appropriate directories automatically to make the work speedy.

7 References

Bill Thomas, Scott Tilley. Documentation for software engineers: what is needed to aid system understanding?

Chisholm, Richard M. 1988. Improving the Management of Technical Writers: Creating a Context for Useable Documentation. In Stephen Doheny-Farina (ed.), Effective Documentation: What We Have Learned From Research. Cambridge, Massachusetts: MIT Press.

Cem Kaner, Ph.D., J.D. & David Pels, B.A. April, 2000. Improving User Documentation and Customer Care

Down-94: Alex Down, Michael Coleman, Peter Absolon, Risk Management for Software Projects, McGraw-Hill Book Company, London 1994.

Green Robin (1997) A Web-based Documentation Review Tool. Proceedings of the 15th annual international conference on Computer documentation SIGDOC '97, ACM Press,

IEEE-610: IEEE Standards Software Engineering, Volume 1, IEEE Standard Glossary of Software Engineering Terminology, IEEE Std. 610-1990, The Institute of Electrical and Electronics Engineers, 1999, ISBN 0-7381-1559-2.

Jirka Kosek (2001), Profiling Docbook documents. An easy way to personalize your content for several target audiences

Mc Arthur Gregory R. (1986) If Writers Can't Program and Programmers Can't Write, Who's Writing User Documentation?. ACM Press

Novick David G. & Ward Karen (2006) Documentation usability: What users say they want in documentation. Proceedings of the 24th annual conference on Design of communication SIGDOC '06, ACM Press,

W3C (15.11.2006) XML Inclusions (XInclude). [URL:http://www.w3.org/TR/xinclude/](http://www.w3.org/TR/xinclude/)

Walsh Norman & Muellner Leonard (2000) Docbook: The Definitive Guide. URL: <http://www.docbook.org/tdg/en/html/> O'Reilly & Associates,

http://en.wikipedia.org/wiki/Software_documentation (Access date 2009-08-29)

<http://transcom.de/transcom/en/technische-dokumentation.htm> (Access date 2009-08-29)

http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office (Access date 2009-08-30)

http://www.help-info.de/en/Help_Info_AP_Help/longhorn_maml_example.htm (Access date 2009-08-30)

<http://en.wikipedia.org/wiki/Wikitext> (Access date 2009-08-30)

<http://en.wikipedia.org/wiki/Transclusion> (Access date 2009-08-30)

<http://ausweb.scu.edu.au/aw05/papers/edited/ball/poster.html> (Access date 2009-09-03)

<http://www.ibm.com/developerworks/library/l-docbk.html> (Access date 2009-09-03)

http://docs.jboss.org/docbook/userguide/html_single/ (Access date 2009-09-03)

<http://www.irt.org/script/5206.htm> (Access date 2009-09-04)

<http://www.students.tut.fi/~leppane7/leppanen.html> (Access date 2009-09-04)

<http://en.wikipedia.org/wiki/XSLT> (Access date 2009-09-04)

<http://www.w3.org/TR/xinclude/>

<http://en.wikipedia.org/wiki/XInclude> (Access date 2009-09-05)

http://docbook.theblog.ca/?page_id=6 (Access date 2009-09-06)

<http://www.sagehill.net/docbookxsl/Profiling.html> (Access date 2009-09-06)

<http://www.kosek.cz/xml/dboscon/profiling/frames.html> (Access date 2009-09-07)

<http://tug.ctan.org/tex-archive/info/latex-veryshortguide/veryshortguide.pdf> (Access date 2009-09-08)

<http://en.wikibooks.org/wiki/LaTeX/Introduction> (Access date 2009-09-08)

<http://pangea.stanford.edu/computerinfo/unix/formatting/features.html> (Access date 2009-09-09)

<http://en.wikipedia.org/wiki/DocBook> (Access date 2009-09-22)

http://services.exeter.ac.uk/cmit/modules/meaningful_markup/webct/ch-xslt-intro.html
(Access date 2009-09-22)

http://linuxcommand.org/man_pages/xsltproc1.html (Access date 2009-09-25)

<http://mirrors.bieringer.de/www.deepspace6.net/contribute/ds6-architecture.html> (Access date 2009-09-25)

http://www.w3schools.com/xslfo/xslfo_intro.asp (Access date 2009-09-25)

http://en.wikipedia.org/wiki/XSL_Formatting_Objects (Access date 2009-09-26)

http://en.wikipedia.org/wiki/Formatting_Objects_Processor (Access date 2009-09-26)

<http://xmlgraphics.apache.org/fop/> (Access date 2009-09-26)

<http://osdir.com/ml/text.xml.fop.devel/2003-03/msg00111.html> (Access date 2009-09-26)

[http://msdn.microsoft.com/en-us/library/ms670169\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms670169(VS.85).aspx) (Access date 2009-09-26)

<http://www.sagehill.net/docbookxsl/HtmlHelp.html> (Access date 2009-09-27)

<http://www.cygwin.com/> (Access date 2009-09-28)

<http://c3fire.org/c3fire/home/home.en.shtml> (Access date 2009-09-29)