

# DocBook XML Publishing System

A Win32 Installation

**Rijk Ravestein**

**ICT Specialist**

**[www.datraverse.com/technology](http://www.datraverse.com/technology)**

Copyright © 2002 Datraverse B.V.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts.

## Table of Contents

1. Introduction .....	1
2. XMLmind XML Editor M1.3 .....	1
3. DocBook XML DTD 4.1 .....	2
4. DocBook XLS Files .....	2
5. SAXON version 6.5.1 .....	2
6. FOP version 0.20.3 .....	2
7. XFC version 0.9 .....	3
8. References .....	3
9. Open issues .....	3
A. Sample makefile for generating publications. ....	3

## 1. Introduction

The DocBook DTD is broadly viewed as the de-facto standard for writing technical documentation. The XML variant of the proven SGML edition is not officially endorsed yet by OASIS. Nevertheless, all the tools are available in open-source to setup a DocBook XML publishing system so you can generate multiple output formats based on a single XML source. Although, due to some bugs and incomplete implementations, not all output beyond straight HTML can be generated in equal quality, I believe XML is the way to go and is worth to invest in. Inevitably things will grow better. The choice for a Win32 implementation looks a bit peculiar considering I use open source tools. Why not use Linux? I do use Linux as a server platform, but practice is that Windows still is, at least for the majority of my clients, the desktop system of choice. The good news is that all the major tools are Java based. So I only need to use Microsofts nmake.exe for the build process and the Microsoft HTML Help Workshop to compile native windows helpfiles. Ofcourse the DTD and XLS files are portable across operating systems. This article is not a tutorial but a hands-on guide to setup the system. My main purpose is to consolidate my research and (some) development in writing and showing you some DocBook at the same time.

## 2. XMLmind XML Editor M1.3

XMLmind is a free Java based XML editor. Apart from being free, the really nice thing about it is that it understands DTD, which means that it can validate your writings. Even better, at any point in your document it offers you picklists with valid DTD elements you can use. Also, you can edit the attributes of each element. I have been using it for a couple of days now and it has never failed me. XMLmind ships with a slightly modified docbook41.dtd. The modification ensures that spaces in some elements like <programlisting> are preserved so that indented lines will not be trimmed in output. XMLmind has two views. A more technical outline view and a WYSIWYG-like view based on an CSS declaration you put in your XML source. You can edit in both views. XMLmind ships with docbook41.css which gives you a good-enough document view. This view lets you easily paste sourcecode in <programlisting> elements. XMLmind takes care for translating the nasty characters like '<', '>', and '&' to the proper entities. This sounds pretty good. Are there any drawbacks? XMLmind has not been designed to edit big documents. It can edit an article or a chapter (~20-30 pages). It cannot edit a whole book. Download XMLmind from here [[http:// www.xmlmind.com/ xmleditor](http://www.xmlmind.com/xmleditor)].

This is a skeleton for a DocBook article like the one I am writing now.

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type="text/css" href="file:/D:/Tools/xxe-ml3-bin/css/docbook41.css" ?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
    "file:/D:/Tools/xml/docbook-x-xxe/docbookx.dtd">
<article>
</article>
```

### 3. DocBook XML DTD 4.1

Although the DTD is shipped with XMLmind you can download the original DocBook XML DTD 4.1 from OASIS [<http://www.oasis-open.org/docbook>].

### 4. DocBook XLS Files

Now that you can deliver wellformed DocBook XML with XMLmind, sit back and relax. We will put tools to work so you can generate the output you like. Basically these tools need the XML and a set of DocBook XLS files for each target document format. Download the DocBook XSL stylesheets from here [<http://sourceforge.net/projects/docbook>].

### 5. SAXON version 6.5.1

SAXON is a Java command-line program that provides a set of services for converting XML data into other formats. Download from here [<http://saxon.sourceforge.net/>]. SAXON is able to convert your XML into:

- HTML

A single HTML page or a multipage so-called chunked HTML version where you can navigate from page to page.

- JavaHelp
- Microsoft Windows Help

Among the generated files is a file called `htmlhelp.hpp` which is input for the *Microsoft HTML Help Compiler*. Using the generated files this compiler can on his turn generate a Windows™ .CHM helpfile.

- XHTML

A single XHTML page or a multipage so-called chunked XHTML version where you can navigate from page to page.

- Formatting Objects (FO)

An intermediate format which can be picked up by other processors to generate PDF and RTF formats.

*If you reference any graphics be sure to include them in the target directory of the generated output as well.*

See the sample makefile for technical details on calling SAXON.

### 6. FOP version 0.20.3

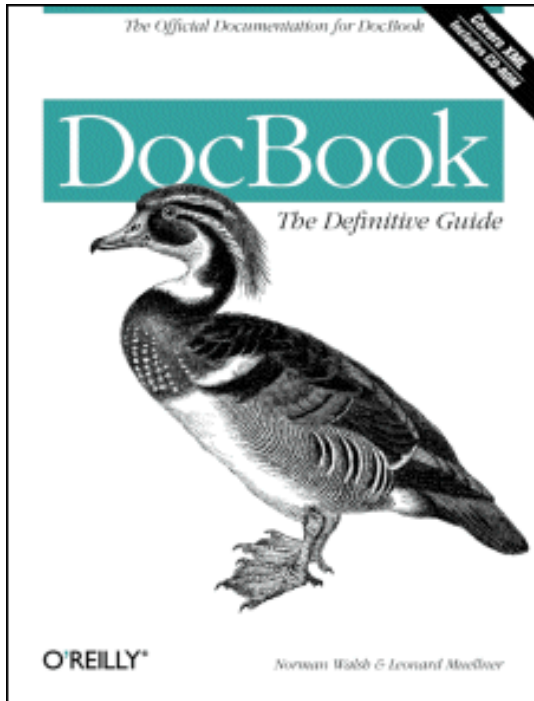
FOP is part of Apache's XML project and is the world's first print formatter driven by XSL formatting objects (FO) and the world's first output independent formatter. It is a Java application that reads a formatting object tree (as e.g. generated by SAXON) and then renders the resulting pages to a specified output. Output formats

currently supported are PDF, PCL, PS, SVG, XML (area tree representation), Print, AWT, MIF and TXT. Since the primary output target is PDF we will use FOP for PDF formatting. Download FOP from here [http://xml.apache.org/fop]. See the sample makefile for technical details on calling FOP.

## 7. XFC version 0.9

XMLmind FO Converter (XFC for short) is an XSL-FO to RTF converter. It takes an XSL-FO source file as input and converts it to RTF. XFC preserves the structure of the original document (e.g. a list-block element is converted to an actual RTF list) and most of the presentation information (font attributes, indentation, etc). The output RTF conforms to version 1.6 of the specification. Download from here [http://www.xmlmind.com/fo-converter]. See the sample makefile for technical details on calling XFC. Though XFC is deemed usable in its current state, it is indeed subject to a number of limitations. Some of these limitations are expected to be withdrawn in future releases.

## 8. References



DocBook: The Definitive Guide. An online [http://docbook.org/tdg/en/] version is available.

## 9. Open issues

There are some issues which needs to be solved.

- How to generate an index?

## Sample makefile for generating publications.

```
#####  
# File      : xml.mak  
# Function  : Produce DocBook based documentation output  
# Dependencies : NMAKE.EXE (Microsoft)
```

```

#
#####
#-----
# Set the name of the main XML source
#-----
SOURCE=docbook_win32_xml_setup

#-----
# Configure the home of the tools...
#-----
XFC_HOME=d:\tools\xml\xfc-09
SAXON_HOME=d:\tools\xml\saxon6_5_1
FOP_HOME=d:\tools\xml\fop-0.20.3
JHHOME=c:\jhl.1
JAVAJAR=$(JAVA_HOME)\bin\jar

#-----
# Configure the home of HTML Help Workshop in
# PATH environment variable
#-----
PATH=$(PATH);C:\Program Files\HTML Help Workshop"

#-----
# Configure location of stylesheets
#-----
XSL_HOME=d:\tools\xml\docbook-xsl

XSL_HTML=$(XSL_HOME)\html\docbook.xsl
XSL_XHTML=$(XSL_HOME)\xhtml\docbook.xsl
XSL_HTML_HELP=$(XSL_HOME)\htmlhelp\htmlhelp.xsl
XSL_JAVA_HELP=$(XSL_HOME)\javahelp\javahelp.xsl
XSL_HTML_CHUNK=$(XSL_HOME)\html\chunk.xsl
XSL_XHTML_CHUNK=$(XSL_HOME)\xhtml\chunk.xsl
XSL_FO=$(XSL_HOME)\fo\docbook.xsl

#-----
# Configure XSL parameters for HTML generation
#-----
XSL_HTML_PARM_1=shade.verbatim="1" section.autolabel="1"
XSL_HTML_PARM_2=callout.graphics.path="images/callouts/gif/"
XSL_HTML_PARM_3=callout.graphics="1" callout.graphics.extension=".gif"
XSL_HTML_PARAMS=$(XSL_HTML_PARM_1) $(XSL_HTML_PARM_2) $(XSL_HTML_PARM_3)

#-----
# Configure XSL parameters for PDF generation
#-----
XSL_FO_PARM_01=paper.type="A4"
XSL_FO_PARM_02=draft.watermark.image=" "
XSL_FO_PARM_03=ulink.show="1" ulink.footnotes="0" ulink.hyphenate=" "
XSL_FO_PARM_04=section.autolabel="1"
XSL_FO_PARM_05=generate.index="1"
XSL_FO_PARM_06=page.margin.top="0.5in"
XSL_FO_PARM_07=body.margin.top="0.5in"
XSL_FO_PARM_08=page.margin.bottom="0.5in"
XSL_FO_PARM_09=body.margin.bottom="0.5in"

# fop.extensions="1" enables bookmarks:
# not to be used in RTF generation
# due to bug in XFC
XSL_FO_PARM_10=fop.extensions="1"

XSL_FO_PARAMS=$(XSL_HTML_PARAMS) $(XSL_FO_PARM_01)
XSL_FO_PARAMS=$(XSL_FO_PARAMS) $(XSL_FO_PARM_02)
XSL_FO_PARAMS=$(XSL_FO_PARAMS) $(XSL_FO_PARM_03)
XSL_FO_PARAMS=$(XSL_FO_PARAMS) $(XSL_FO_PARM_04)
XSL_FO_PARAMS=$(XSL_FO_PARAMS) $(XSL_FO_PARM_05)
XSL_FO_PARAMS=$(XSL_FO_PARAMS) $(XSL_FO_PARM_06)
XSL_FO_PARAMS=$(XSL_FO_PARAMS) $(XSL_FO_PARM_07)
XSL_FO_PARAMS=$(XSL_FO_PARAMS) $(XSL_FO_PARM_08)
XSL_FO_PARAMS=$(XSL_FO_PARAMS) $(XSL_FO_PARM_09)

XSL_RTF_PARAMS=$(XSL_FO_PARAMS)
XSL_PDF_PARAMS=$(XSL_FO_PARAMS) $(XSL_FO_PARM_10)

#-----
# All code to follow need not to be changed if you

```

```

# adhere to the following directory structure:
#
# <your DocBook project directory>
#   |
#   +----- ~temp      (intermediate build products)
#   |
#   +----- source     (DocBook XML sources)
#   |   |
#   |   +----- img
#   |
#   +----- target     (generated publications)
#       |
#       +----- html
#       |
#       +----- html-chunk
#       |
#       +----- htmlhelp
#       |
#       +----- javahelp
#       |
#       +----- pdf
#       |
#       +----- rtf
#       |
#       +----- xhtml
#       |
#       +----- xhtml-chunk
#
#-----
#
#-----
# Directory structure
#-----
S=.
PATH2ROOT=..
PATH2TARGET=..\target
T_XML_TEMP=$(PATH2ROOT)\~temp
T_XML_HTML=$(PATH2TARGET)\html
T_XML_XHTML=$(PATH2TARGET)\xhtml
T_XML_PDF=$(PATH2TARGET)\pdf
T_XML_RTF=$(PATH2TARGET)\rtf
T_XML_HTML_CHUNK=$(PATH2TARGET)\html-chunk
T_XML_XHTML_CHUNK=$(PATH2TARGET)\xhtml-chunk
T_XML_HTML_HELP=$(PATH2TARGET)\htmlhelp
T_XML_JAVA_HELP=$(PATH2TARGET)\javahelp

#-----
# Win32 Tooling: HTML Help Workshop
#-----
HHC=hhc.exe

#-----
# Java tooling
#-----
HSVIEWER=$(JHHOME)\demos\bin\hsviewer1_1

#-----
# Program macro's
#-----
XFC_CLASSPATH=$(XFC_HOME)\xfc.jar;$(XFC_HOME)\jaxp.jar
XFC_CLASSPATH=$(XFC_CLASSPATH);$(XFC_HOME)\sax.jar;$(XFC_HOME)\xp.jar
XFC=java -cp $(XFC_CLASSPATH) com.xmlmind.fo.converter.Driver

SAXON=java -jar $(SAXON_HOME)\saxon.jar

FOP_CLASSPATH=$(FOP_HOME)\build\fop.jar;$(FOP_HOME)\lib\batik.jar
FOP_CLASSPATH=$(FOP_CLASSPATH);$(FOP_HOME)\lib\xalan-2.0.0.jar
FOP_CLASSPATH=$(FOP_CLASSPATH);$(FOP_HOME)\lib\xerces-1.2.3.jar
FOP_CLASSPATH=$(FOP_CLASSPATH);$(FOP_HOME)\lib\avalon-framework-4.0.jar
FOP_CLASSPATH=$(FOP_CLASSPATH);$(FOP_HOME)\lib\logkit-1.0.jar;
FOP_CLASSPATH=$(FOP_CLASSPATH);$(FOP_HOME)\lib\jimi-1.0.jar
FOP=java -cp $(FOP_CLASSPATH) org.apache.fop.apps.Fop

#-----
# rules
#-----
.SUFFIXES: .xml .html

```

```

.SUFFIXES: .xml .fo
.SUFFIXES: .fo-pdf .pdf
.SUFFIXES: .fo-rtf .rtf
.SUFFIXES: .xml .xhtml
.SUFFIXES: .xml .chm
.SUFFIXES: .xml .hs

#-----
# Products
#-----
all: html xhtml pdf rtf \
    html-chunk xhtml-chunk \
    htmlhelp javahelp

html: $(T_XML_HTML)\$(SOURCE).html
xhtml: $(T_XML_XHTML)\$(SOURCE).xhtml
pdf: $(T_XML_TEMP)\$(SOURCE).fo-pdf $(T_XML_PDF)\$(SOURCE).pdf
rtf: $(T_XML_TEMP)\$(SOURCE).fo-rtf $(T_XML_RTF)\$(SOURCE).rtf
htmlhelp: $(T_XML_HTML_HELP)\$(SOURCE).chm
html-chunk: $(T_XML_HTML_CHUNK)\index.html
xhtml-chunk: $(T_XML_XHTML_CHUNK)\index.html
javahelp: $(T_XML_JAVA_HELP)\$(SOURCE).jar

#-----
{$(S)}.xml{$(T_XML_HTML)}.html:
    @echo Generating $(@B).html ...
    @$(SAXON) -o $(T_XML_HTML)\$(@B).html $(S)\$(@B).xml $(XSL_HTML) $(XSL_HTML_PARMS)
#-----
{$(S)}.xml{$(T_XML_XHTML)}.xhtml:
    @echo Generating $(@B).xhtml ...
    @$(SAXON) -o $(T_XML_XHTML)\$(@B).xhtml $(S)\$(@B).xml $(XSL_XHTML)
#-----
{$(S)}.xml{$(T_XML_TEMP)}.fo-pdf:
    @echo Generating $(@B).fo-pdf ...
    @$(SAXON) -o $(T_XML_TEMP)\$(@B).fo-pdf $(S)\$(@B).xml $(XSL_FO) $(XSL_PDF_PARMS)
#-----
{$(S)}.xml{$(T_XML_TEMP)}.fo-rtf:
    @echo Generating $(@B).fo-rtf ...
    @$(SAXON) -o $(T_XML_TEMP)\$(@B).fo-rtf $(S)\$(@B).xml $(XSL_FO) $(XSL_RTF_PARMS)
#-----
{$(T_XML_TEMP)}.fo-pdf{$(T_XML_PDF)}.pdf:
    @echo Generating $(@B).pdf ...
    @copy $(T_XML_TEMP)\$(@B).fo-pdf
    @$(FOP) -q -fo $(@B).fo-pdf -pdf $(T_XML_PDF)\$(@B).pdf
    @del $(@B).fo-pdf
#-----
{$(T_XML_TEMP)}.fo-rtf{$(T_XML_RTF)}.rtf:
    @echo Generating $(@B).rtf ...
    @copy $(T_XML_TEMP)\$(@B).fo-rtf
    $(XFC) $(@B).fo-rtf $(T_XML_RTF)\$(@B).rtf
    @del $(@B).fo-rtf
#-----
$(T_XML_HTML_CHUNK)\index.html: $(S)\$(SOURCE).xml
    @echo Generating html chunks ...
    @$(SAXON) $(S)\$(SOURCE).xml $(XSL_HTML_CHUNK) $(XSL_HTML_PARMS)
    @move *.html $(T_XML_HTML_CHUNK)
#-----
$(T_XML_XHTML_CHUNK)\index.html: $(S)\$(SOURCE).xml
    @echo Generating xhtml chunks ...
    @$(SAXON) $(S)\$(SOURCE).xml $(XSL_XHTML_CHUNK)
    @move *.html $(T_XML_XHTML_CHUNK)
#-----
{$(S)}.xml{$(T_XML_HTML_HELP)}.chm:
    @echo Generating $(@B).chm ...
    @$(SAXON) $(S)\$(@B).xml $(XSL_HTML_HELP)
    @-l $(HHC) htmlhelp.hhp
    @move *.html $(T_XML_TEMP)
    @move *.hhp $(T_XML_TEMP)
    @move *.hhc $(T_XML_TEMP)
    @move *.chm $(T_XML_HTML_HELP)\$(SOURCE).chm
#-----
$(T_XML_JAVA_HELP)\$(SOURCE).jar: $(S)\$(SOURCE).xml
    @echo Generating JavaHelp ...
    @$(SAXON) $(S)\$(SOURCE).xml $(XSL_JAVA_HELP)
    @$(JAVAJAR) cf $(T_XML_JAVA_HELP)\$(SOURCE).jar jhelp*. * .html img\*.

```

```

@if exist *.html del *.html /Q
@if exist jhelp*. * del jhelp*. * /Q

#-----
saxon:
    $(SAXON) -?
hhc:
    $(HHC) -?
fop:
    $(FOP) -?
xfc:
    $(XFC) -help
v-javahelp: javahelp
    $(HSVIEWER) jhelpset.hs $(T_XML_JAVA_HELP)\$(SOURCE).jar
showjar:
    $(JAVAJAR) tf $(T_XML_JAVA_HELP)\$(SOURCE).jar
v-pdf: pdf
    @start /B $(T_XML_PDF)\$(SOURCE).pdf

#-----
clean:
    @if exist $(T_XML_TEMP)\*.h* del $(T_XML_TEMP)\*.h*
    @if exist $(T_XML_TEMP)\*.fo del $(T_XML_TEMP)\*.fo

    @if exist $(T_XML_HTML)\*.h* del $(T_XML_HTML)\*.h*
    @if exist $(T_XML_XHTML)\*.x* del $(T_XML_XHTML)\*.x*
    @if exist $(T_XML_PDF)\*.pdf del $(T_XML_PDF)\*.pdf

    @if exist $(T_XML_HTML_CHUNK)\*.h* del $(T_XML_HTML_CHUNK)\*.h*
    @if exist $(T_XML_XHTML_CHUNK)\*.h* del $(T_XML_XHTML_CHUNK)\*.h*
    @if exist $(T_XML_HTML_HELP)\*.c* del $(T_XML_HTML_HELP)\*.c*

    @if exist $(T_XML_JAVA_HELP)\*.h* del $(T_XML_JAVA_HELP)\*.h*
    @if exist $(T_XML_JAVA_HELP)\*.j* del $(T_XML_JAVA_HELP)\*.j*
    @if exist $(T_XML_JAVA_HELP)\*.x* del $(T_XML_JAVA_HELP)\*.x*

#-----
# end of makefile

```