Inhaltsverzeichnis

DocBook Publishing

Version 1.02

24. May 2003

Copyright © 2003 Stefan Rinke

This article is available online at Stefan Rinke, Articles

Chapter 1. Introduction

DocBook Publishing describes the framework for publishing with DocBook. It describes how you can easily create a publication or an article in many different formats. You can either read this article online or download (see chapter 12) it in several formats. This framework is based on the work of many people who contributed many parts and made it available to a larger audience as Open Source. Special thanks to Mark Pilgrim who inspired me to write this article because of his publications especially Dive Into Python. I have done nothing else than making it work under Windows and in German with all necessary tools. I use it myself: This article was certainly written with *DocBook Publishing* .

Again: This article is not about DocBook itself (see links to other sites below) but about the description of a setup how to run it in practice. You can also download all necessary tools (also as binaries) from this WebSite; so it should not be a problem to set up your own system enabling you to successfully write documents with DocBook.

1.1. Requirements

It is important to understand that this framework is not a WYSIWYG solution or "Click−clack" application (a Windows application operated by mouse click). Articles are rather written in *soucecode* in DocBook (XML) and then compiled. This might remind you on the good old TEX times where a document was written in the same way.

If you are afraid of using command lines, I can only recommend to use the common Office tools and keep your hands off this approach.

You should be familiar with XML and XSLT basics so that you will not become confused even by the easiest topics. All other information is described in this article so that you will be successful in a very short time.

Chapter 2. Read *DocBook Publishing* online

This article is not completed yet and will be extended by new chapters or user comments. These changes will soon be visible in a Revision history showing the differences between the different versions.

Chapter 3. Software

3.1. Overview

To create all formats, some software packages are required:

- The Build environment is based on Ant , a popular Build tool from the Apache Group . The XML tools and here especially the stylesheet processes rely on *Java* . Thus a Java runtime environment Version 1.3 or higher is required. .
- To create the Word version of an article, *Microsoft Word* is used. This of course only works on Microsoft Windows and is the only tool used here not available for freely.
- The Help File format requires a free Microsoft tool which can be download from Microsoft : hhc.exe the Help compiler
- The PDF format is generated with the tool HTMLDOC . Source code and binary can also be downloaded for free.
- The Plain−Text Format is generated via HTML and the tool w3m, a text browser similar to Lynx.
- Mark realized some support functions in Python, so Python should also be installed on the system.
- If you want to create your DocBook with OpenOffice you naturally need to install OpenOffice.
- The package ooo2sDbk converts OpenOffice documents to DocBook.

3.2. Java

It is recommended to download *Java* directly from Sun. The Runtime 1.3.1_06 [1] for Windows is available at http://java.sun.com/j2se/1.3/download.html . Sun supplies a common setup so that it can easily be installed.

Important: To ensure that the main batch file which starts Ant finds Java, you need to set the environment variable JAVA_HOME. Alternatively it is also possible to enter the Java installation path directly in the batch file.

To check if all settings are right, carry out the following tests in the command line:

```
[C:\ ]$ echo %JAVA_HOME%
 C:\Programme\JavaSoft\jdk1.3.1_06
 [C:\ ]$ %JAVA_HOME%\bin\java
 Usage: java [−options] class [args...]
 (to execute a class)
 or java −jar [−options] jarfile [args...]
 (to execute a jar file)
 ...
```

For defining and transforming with XSLT some additional Java libraries (JARs) are required. As these are the same for all projects, it is recommended to install them in the Lib directory of JAVA_HOME (here `C:\Programme\JavaSoft\jdk1.3.1_06\lib` . Download the libraries here .

3.3. HTMLDOC

HTMLDOC is available from http://www.easysw.com/htmldoc/ as source code and binary, although the functionality of the binary is supposed to be limited. I did not try this out but compiled a new binary.

Only required is the htmldoc.exe and a few font files which as a default are expected by the binary in `C:\Program Files\htmldoc` . This ZIP file contains all files (Version 1.8.21). Make sure that the binary htmldoc.exe is found in the PATH. This is done by copying it to the Windows directory. If you do not like this, create a new directory for such tools (I use `C:\Programme\bin`) and adapt the path so that the directory is included. Test this by entering

htmldoc in the prompt. The following should be displayed:

```
[C:\ ]$ htmldoc
 ERROR: No HTML files!
HTMLDOC Version 1.8.21 Copyright 1997-2002 Easy Software Products, All Rights Reserved.
This software is governed by the GNU General Public License, Version 2, and is based in part on the w

Usage:
 htmldoc [options] filename1.html [ ... filenameN.html ]

Options:

 --batch filename.book
 ...
```

3.4. W3M

The tool w3m for creating plain text is available for download at http://www.w3m.org/ . The only thing you need for this framework is the w3m.exe binary and `CGYWIN-DLL` [2] . You can download it here directly. As described above, copy the EXE and DLL to a directory included in the path. The command line should look as follows:

```
[C:\ ]$ w3m
 version w3m/0.1.9
usage: w3m [options] [URL or filename]
options:

-t tab set tab width
 ...
```

3.5. HTML Help Workshop

The Microsoft Help compiler is directly available form Microsoft clicking this URL . The file HTMLHELP.EXE Version 1.32 is the required file and started after the download. The setup installs the *HTML Help Workshop* usually using the path C:\Programme\HTML Help Workshop . Again, if you want to run this directly, the directory must be included in the path. As an alternative you can also copy the file hhc.exe to the tool directory. The command line will look as follows:

```
[C:\ ]$ hhc
 Usage: hhc <filename>
 where <filename> = an HTML Help project file
 Example: hhc myfile.hhp
```

3.6. Python

Python is available from http://www.python.org/ or download ActivePython from ActiveState . As said before: Python needs to be included in the path after installation. Both distributions usually do this automatically. Test directly in the prompt:

```
[C:\ ]$ python
 ActivePython 2.2.1 Build 222 (ActiveState Corp.) based on
Python 2.2.1 (#34, Apr 15 2002, 09:51:39) [MSC 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

To close the interactive mode, press CTRL-Z

Important: To generate the "real" Word document, the win32com extension needs to be installed− Test in the prompt with:

```
>>> import win32com
>>> print win32com
 <module 'win32com' from 'C:\Python22\Lib\site-packages\win32com\__init__.pyc'>
```

3.7. The Framework itself

Now you nearly have all the tools you need, only missing is the first project. You can download this file from the section Download and generate all pages and output format with **`make all`** .

Nevertheless, it is easier to start with the Hello World project which is described in the next chapter.

3.8. OpenOffice−Writer

This is an optional component as you can use any editor to create a DocBook source code file. Especially for the beginner it might easier to start with OpenOffice. The advantage of the OpenOffice Writer is that you can edit your text WYSIWYG without seeing the XML code. This advantage might also be a disadvantage as you will not have complete control over all aspects of DocBook Publishing. OpenOffice Version 1.0.1 can be downloaded from http://www.openoffice.org/ .

3.9. OOo2sDbk

To generate DocBook with OpenOffice a package is required which converts the OpenOffice format to DocBook. This is for example the package OOo2sDbk dar. See chapter 6 DocBook with OpenOffice for more information.

The package can easily be installed. Only extract the zip file ooo2sdbk.zip to the directory of the Python Site package. The module is then commonly available. Download the package from this Website as it contains extended functionality compared to the original.

Test respectively:

```
>>> import ooo2sdbk
>>> print ooo2sdbk
 <module 'ooo2sdbk' from 'C:\Python22\Lib\site-packages\ooo2sdbk\__init__.pyc'>
```

---

[1] You can also use the Version 1.4.1 alternatively

[2] An environment which easily ports many Unix programs and tools to Windows (see http://www.cygwin.com/)

Chapter 4. Hello World−Project

The standard directory structure is as follows:

- The Common directory: contains all libraries and tools required for the project. The contents is always the same and independent of the project contents.

  Note: Note

  This is not fully right as the layout of the documents with XSL style sheet is defined here.
- The Build directory: contains the batch file for calling Ant: make.bat (or their UNIX variant make.sh), the file `build.xml`, which controls the building and the directory with the source documents: `xml`.

- The XML directory: contains all XML files which make up the actual document or book. For Hello World one file is sufficient.

The build file for ant looks like follows:

```
<?xml version="1.0" encoding="utf−8"?>
<!DOCTYPE project [
         <!ENTITY common "../common">
         <!ENTITY buildcommon SYSTEM "../common/build_common.xml">
]>
<project name="HelloWorld" default="html" basedir=".">(1)
         <property name="lang" value="de"/>
         <target name="postprocess"/>
         <property name="project" value="HelloWorld"/>
         <property name="javahome" value="..."/>
         <property name="fileversion" value="1.0"/>(2)
&buildcommon;
</project>
```

This file only determines the project name and which document types will be generated as a standard.

(1) Defines project name and standard target (here html).

(2) Defines the project properties. Here the project name which is required as a variable in the shared control file for Ant `build_common.xml`.

Only missing is the contents; here the file HelloWorld.xml which is saved in the XML directory.

For the simplest HelloWorld this is only:

```
<?xml version="1.0" encoding="utf−8"?>
<!DOCTYPE article PUBLIC "−//OASIS//DTD DocBook XML V4.1.2//EN"
"../../common/docbook/dtd/docbookx.dtd">
<article lang="de">
 <articleinfo>
     <title>Hello World</title>
     <author>
       <firstname>Stefan</firstname>
       <surname>Rinke</surname>
     </author>
 </articleinfo>
 <section>
     <title>Hello World</title>
     <para>That's all.</para>
 </section>
</article>
```

Now we have everything we need for the first test. After having installed all tools, it should not be a problem to structure your fist project. If the following is returned:

```
make all
 BUILD SUCCESSFUL
Total time: 11 seconds
```

the dist directory under build should now contain all formats of the Hello World document created just now.

Chapter 5. Short Tutorial

5.1. Book or Article

The following explanations follow mainly the reference book of Norman Walsh and Leonard Muellner (see here).

Typically a book is a larger bit of work than an article. Its structure comprises meta information (<bookinfo>), which describe the author, the title and the copyright information. It might also have a Preface Vorwort and several chapters which are divided into paragraphs. In addition it is possible to have appendixes, literatur reference list, a glossary and an index.

An article is much shorter as it does not contain as much parts as a book and is usually only structured into individual chapters. But it does also contain meta information (<articleinfo>) and can have appendixes, a literature reference list, glossary and index.

5.2. A simple DocBook book

The following XML section shows a typical structure of a DocBook book.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
"../../common/docbook/dtd/docbookx.dtd">
<book lang="de">
 <title>Mein erstes DocBook Buch</title>
 <bookinfo>(1)
     <author>
       <honorific>Dipl.-Ing</honorific>
       <firstname>Stefan</firstname>
       <surname>Rinke</surname>
     </author>
 </bookinfo>
 <preface>(2)
     <title>Vorwort</title>
     <para>In einem Vorwort stehen Dinge über das Buch an sich.</para>
 </preface>
 <chapter>(3)
     <title>Einführung</title>
     <para>Dies ist ein einfacher Artikel
       <indexterm>(4)
         <primary>Artikel</primary>
         <secondary>DocBook-Artikel</secondary>
       </indexterm>.
     Das ist nur ein Beispielabsatz.</para>
 </chapter>
 <appendix>(5)
     <title>Anhang 1</title>
     <para>Das ist ein Anhang.</para>
 </appendix>
 <glossary>(6)
     <glossdiv>
       <title>Ein Glossar</title>
       <glossentry>
         <glossterm id="DocBook">DocBook</glossterm>
         <glosssee>DocBook ist ein Standard zur Erstellung...</glosssee>
       </glossentry>
     </glossdiv>
 </glossary>
```

```
 <index/>(7)
</book>
```

(1) The Bookinfo section is optional although you might want to be named as the author.

(2) The Preface is also optional. But if the book becomes very large, the preface will have about this strucutre.

(3) Here follow one or several chapters which start with a title <title> and are even more structured by paragraphs <para>.

(4) With this an index entry is created. The secondary entry is optional.

(5) Following tihs you will have one or more appendixes.

(6) If you want to, you can also add a glossary.

(7) This tag marks the position where the completed index is to be inserted.

To see how such a book might look in HTML[3], click hier.

5.3. An artikel

A simple article looks accordingly:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
"../../common/docbook/dtd/docbookx.dtd">
<article lang="de">
 <title>Mein erster Artikel</title>
 <articleinfo>
     <author>
       <honorific>Dipl.-Ing</honorific>
       <firstname>Stefan</firstname>
       <surname>Rinke</surname>
     </author>
 </articleinfo>
 <para>Dies ist ein einfacher Artikel. Dieser Ausdruck
     <indexterm>
       <primary>Ausdruck</primary>
     </indexterm>
         soll im Index erscheinen.</para>
 <appendix>
     <title>Anhang 1</title>
     <para>Das ist ein Anhang.</para>
 </appendix>
 <glossary>
     <glossdiv>
       <title>Ein Glossar</title>
       <glossentry>
         <glossterm id="DocBook">DocBook</glossterm>
         <glosssee>DocBook ist ein Standard zur Erstellung...</glosssee>
       </glossentry>
     </glossdiv>
 </glossary>
 <index/>
</article>
```

To see the result online, click here.

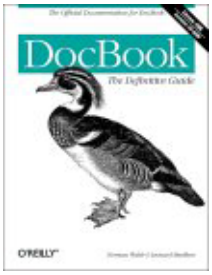
5.4. Online Resources

More deeper knowledge on DocBook can be found in the following online resources.

5.4.1. DocBook: The Definitive Guide

The Standard: DocBook: The Definitive Guide which can be read online and offers information from scratch to reference material.

Figure 5.1. DocBook: The definitive Guide



5.4.2. Tutorial by Stephan Wiesner

Stephan Wiesner has written an excellent tutorial. It is also available online.

5.4.3. Online Tutorial von Lars Trieloff

An excellent tutorial by Lars Trieloff can be found under http://trieloff.net/doctutorial/build/index.html.

---

[3] this depends on the defined style sheets

Chapter 6. DocBook publishing with OpenOffice

A very easy possibility to work with docbook is to use OpenOffice as an editor. OpenOffice uses xml as natural file format, therefore it is quite obvious to think about a conversion from OpenOffice writer files to docbook xml. Indeed this is relatively easy. The following steps are necessary:

- Unpack the OpenOffice file
- Join the individual XML files to a large, flat XML file
- Swap out embedded pictures
- Transform the large XML file to DocBook using a xslt stylesheet

To make it clearer, what exactly is going on, the procedure is to be described briefly.

OpenOffice stores its documents in files with the extension swx. These files are actually ZIP−archives[4], containing serveral XML−files. One can extract and rearrange the individual files very easily.

In order to be able to transform via XSLT at the end, one single large XML file is merged, whereby the different part remain separate by XSLT namespaces.

Pictures are embedded by OpenOffice directly into the XML file and stored as base64 in the XML file. Unfortunately the usual DocBook stylesheet are not able to use these embedded pictures, but rather expect them as external files.

All these steps are settled by a small Python module of Eric Bellot. It can downloaded here and is part of the whole framework either.

The following transformation is accomplished with a special stylesheet using an arbitrary XSLT−processor (here saxon). The stylesheet of Eric Bellot was improved for this framework and there are serveral variants, depending upon whether one would like to write a book or an article .

6.1. Interaction between OpenOffice and DocBook

OpenOffice uses paragraph formats, character formats and frame formats for formatting documents. The are named and linked to certain layout rules. The connection between DocBook and OpenOffice is made exactly over these predefined formats. If for instance a sentence contains a `filename`, then it is formatted in OpenOffice with the character format filename and converted later into `<filename>...</filename>` .

It is the same with paragraph format for enumeration or numbered lists: one can use them in OpenOffice as usual and they will be converted in the corrosponding DocBook tags automatically. To make this always work right, one should use a document template or an empty standard document. Such a document is included in the framework.

The most corrosponding formats / tags are intuitive, a more detailed description is on Eric Bellot's website.

6.2. More functions

6.2.1. Book or Article

As already mentioned there are two different stylesheets or a controllable one, to generate alternatively a book or an article . That is defined by the build−property doc−type in the control file build.xml .

6.2.2. CalloutList

The be able to produce so−called CalloutLists , an additional enumerating format CalloutList was taken up .

References are converted to cross references. It is important, that the names of the bookmarks begins with co. . If these conventions are kept, the stylesheet automatically generates the right `<co id= co.test >` and `<callout arearefs= co.test >`. How that functions exactly, one can see best in this document.

6.2.3. Insert a note

If one inserts a note with OpenOffice, this is transferred simply literally to the DocBook document. Thereby one can generate arbitrary DocBook tags. However the exact control of the position of the tag is not possibly, so that structure tags, that are bound to a certain content model sometimes cannot be inserted in this way. What is possible without any problems are e.g. Index entries.

---

[4] One can open it actually e.g. simply with WinZip

Chapter 7. Download DocBook Publishing

- PDF
- HTML
- Microsoft Word
- Windows Help
- Plaintext
- HTML (eine Datei)
- XML (DocBook)
- OpenOffice
- Build Framework (for users of the framework)