DocBook with LyX

B. Guillon

20th September 2001

Contents

Abstract

This document presents how to write DocBook documents with LyX. The main aspects covered here are about the tools involved and their interaction with LyX. By default it is the SGML DocBook format that is supported by LyX, but the XML format can also be used. A small part deals with the XML aspects.

You can dowload this document here:

- The PDF file
- The document sources (LyX) and outputs (XML, HTML, LaTeX, PDF)

Part I

Installation and Use

Chapter 1

Installation

1.1 Install Overview

The use of DocBook through LyX needs:

1. The DocBook, jade, and sgmltools packages be installed.
2. The SGML_CATALOG_FILES variable properly set.
3. LyX reconfigured.

1.1.1 Packages to install

To produce DocBook documents LyX needs some packages to be installed:

- DocBook: the following features must be installed to make DocBook usable:
  - The DocBook DTD, that can be dowloaded from DocBook.
  - The ISO entities, that define some standard SGML entities (e.g. >, <, etc.).
  - The Norman Walsh's DocBook DSSSL modular stylesheets.
- Jade/jadetex
- SGMLtools-lite (version 3.0)

All these features are detailed in the other sections of this chapter. Before downloading each of the packages, check that it is not already installed in your system! For instance, Red Hat provides most of these packages.

1.1.2 Setting SGML_CATALOG_FILES

See section Environment Variables .

1.1.3 LyX Reconfiguration

To reconfigure LyX do as follow:

1. Start LyX
2. Reconfigure LyX (Options->Reconfigure)
3. Exit from LyX
4. Start LyX again, and open a new document (File->New). If the reconfiguration succeded, the following new document layouts should be available in the Layout->Document->Class list::
   - DocBook article (SGML),
   - DocBook book (SGML),
   - DocBook chapter (SGML),
   - DocBook section (SGML).

1.2 DocBook

1.2.1 The DocBook DTD

The DocBook package only provides the template for this kind of document (docbook.dtd). The template includes some description modules (.mod files). The DocBook DTD by itself does not allow to publish a document, because no stylesheet is provided with the package. Untill now the version supported by LyX is V3.1 but it will move to V4.x.

1.2.2 The N. Walsh modular stylesheets

This package provides DSSSL stylesheets by Norman Walsh to publish SGML files at least to tex for jadetex and HTML. All the other stylesheets used suppose that these stylesheets are available, and actually call them.

1.3 Jade/jadetex

Jade compiles an SGML document, and produces one or more files according to the expected format (tex for jadetex, rtf, HTML). The expected format is specified by the stylesheet to use. Jadetex compiles the tex file outputed by jade and produces the related DVI file.

1.4 SGMLtools

SGMLtools interfaces LyX to jade/jadetex, and provides several stylesheets that customize the standard N. Walsh DocBook stylesheets. Figure below shows the relationship between each feature.

1.4.1 SGMLtools and LyX

SGMLtools must be installed to allow LyX to export a DocBook document to one of the available output format (DVI, PostScript, HTML). The SGMLtools version must be greater than 2.0. SGMLTools version 3.0 provides some stylesheets (.dsl) and some scripts written in python.

Le main script, sgmltools, is called by LyX when the document is exported. LyX runs the command:

```
sgmltools -b backend mydoc.sgml
```

Where backend depends on the expected output format:

Output format	Backend
DVI	dvi
PostScript	ps
HTML	html

1.4.2 SGMLtools and Jade

SGMLtools calls jade and jadetex if necessary. So, these tools must be installed (cf. Jade). When no stylesheet is explicitely specified to sgmltools (option -s) the stylesheet used is deduced from the chosen backend. An alias is created according to the backend, and its syntax is ?sglmtools- *backend* ?. The links between the aliases and the stylesheets are defined in the file /etc/sgml/aliases and ?/.aliases if this file exists. At least one of these files must exist and be correctly initialized to make sgmltools work properly.

1.4.3 SGMLtools and DocBook

SGMLtools provides and uses its own stylesheets, according to the selected backend. These stylesheets define some specific characteristics, and call the N. Walsh stylesheets.

1.5 Environment Variables

The environment variable SGML_CATALOG_FILES lists the catalogs where the available stylesheets are. It is used by jade and must refer to all the packages installed (DocBook stuff, jade, sgmltools, etc.). For example SGML_CATALOG_FILES could be set like this:

```
DBK_CAT=/usr/lib/sgml/docbook.cat
CMN_CAT=/usr/lib/sgml/sgml-common.cat
NWS_CAT=/usr/lib/sgml/stylesheets/nwalsh-modular/catalog
JAD_CAT=/usr/doc/jade-1.2.1/dsssl/catalog
JAD_CAT=$JAD_CAT:/usr/doc/jade-1.2.1/unicode/catalog
STL_CAT=/usr/local/share/sgml/stylesheets/sgmltool/sgmltools.cat
export SGML_CATALOG_FILES=$DBK_CAT:$CMN_CAT:$NWS_CAT:$JAD_CAT:$STL_CAT
```

Chapter 2

Use of LyX

2.1 Creating a document

Creating a Docbook document with LyX is exactly the same as for any other kind of document: create a new file and select the Docbook layout you wish (Layout->Document->Class->DocBook {article, book, chaper, section} SGML).

Now, write the document (it should take more time ;-).

2.2 Publishing/exporting a document

The document can be exported to SGML, DVI, PostScript, text and HMTL, by using File->Export as.

2.2.1 DVI, PostScript Format

Publishing to DVI or to PostScript creates in the directory where the source document is a .dvi or .ps file. LyX does the following operations to produce the exported file:

1. The document is translated to a temporary SGML file,
2. A temporary tex file is build by running sgmltools on the SGML temporary file, with tex for backend,
3. The DVI file is build by running jadetex on the temporary tex file,
4. The PostScript file is build by running dvips on the DVI file.

2.2.2 HTML Format

Exporting to HTML creates a directory whose name is the file name to export. This directory contains the output HTML files. There is one HTML file by chapter or section.

2.3 Publishing a document by hand

Publishing by hand allows you to choose stylesheets or backend not supported by default in LyX. Besides, it is a good exercise to learn how the tools work and to check independently from LyX that everything is well configured. To do this you just need to:

- Export as SGML,
- Run sgmltools or jade directly on the outputed SGML file.

The publishing examples in the following sections are given only to show how it works. In fact all of these conversions are possible by using LyX directly.

2.3.1 Use of sgmltools

2.3.1.1 DVI Format

```
sgmltools -b dvi mydoc.sgml
```

2.3.1.2 HTML Format

```
sgmltools -b html mydoc.sgml
```

2.3.2 Use of jade

2.3.2.1 DVI Format

```
jade -t tex -d /usr/lib/sgml/stylesheets/nwalsh-modular/print/docbook.dsl mydoc.sgml
jadetex mydoc.tex
```

2.3.2.2 HTML Format

```
jade -t sgml -d /usr/lib/sgml/stylesheets/nwalsh-modular/html/docbook.dsl mydoc.sgml
```

Part II

Using XML DocBook

Chapter 3

XML Overview

3.1 XML vs SGML

In many ways XML seems to be the future of DocBook: more (free) tools are developped for XML, and it begins to be widely used in the Internet land in contrary of SGML that grows slowly. A possible reason of this situation is that the XSL stylesheet format is easier to understand and to learn than the DSSSL format; thus, stylesheet customization (at least for a beginner) is encouraged with XML.

At the writer level, there is no main differences since the defined elements are the same, even if SGML is more flexible (i.e. the ability to minimize or omit tags). The main point to decide whether using XML instead of SGML is if the DocBook's SGML branch is planned to be obsoleted one day or not.

3.2 XML Maturity

Even if XML seems to be very promising, it is not mature enough, especially for printing documentation. Printed documentation tends to move one more toward DSSSL processors for output, and solid SGML tools chains exist that work as soon as it is installed. For instance the DocBook DTD / DocBook DSSSL / jade / [jadetex, pdfjadetex] chain is provided in the Red Hat distribution and works without big problems.

It seems that there are more XML tools, but the main problem is to make them work together! One step is to configure each of them, the other step is to manage specific configurations that make one work with another. For instance the XSL stylesheets provide some java archives, to add to the CLASSPATH to make the XSL processor used work!

3.3 XML and LyX

3.3.1 Lyx-1.1.6*

In release 1.1.6*, LyX does not directly support the DocBook XML format. Only DocBook SGML V3.1 is supported. However, the elements used are fully compliant to XML, and the SGML produced by LyX is well formed for XML.

In fact few things are needed to:

- make LyX export as XML,
- import XML files into LyX.

3.3.1.1 Exporting as XML

Since LyX uses an XML compliant output format, exporting as XML only means changing the output file header. For instance, in the main file of an SGML document (that is, not a file included in another one) the following SGML header:

```
<!doctype article public "-//OASIS//DTD DocBook V3.1//EN"
[...]>
```

should be changed to:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE article SYSTEM "file:///(full path to)/docbook/docbkx412/docbookx.dtd"
[...]>
```

3.3.1.2 Importing XML files

To import XML file into LyX, you can use the DB2LyX set of stylesheets, that translate any XML file to a LyX-1.1.6 file compliant format.

3.3.2 LyX-1.2.0

LyX version 1.2.0 is expected to be able to export as XML DocBook, and even import an XML DocBook document. However, it does not mean that LyX has changed its format to XML! In fact it is only that the processes described in the previous section will be integrated in this release.

Chapter 4

Dependencies

Writing or importing XML DocBook documents in LyX needs the following installed:

- the XML DocBook DTD 4.1.2,
- the Norman Walsh DocBook XSL stylesheets,
- an XSLT.

4.1 Compliances

One of the problems encountered in using XML tools is the compliances between each other. According to some experiments, we can say that:

- The DocBook XSL stylesheets version 1.29 don't work with XT.
- To use XT, use the stylesheets version 1.34 or later.
- To chunk with XT you need to use the xtchunk.xsl stylesheet instead of chunk.xsl

4.2 XSL Translator choice

Until now the best translators seem to be:

1. xsltproc (Gnome libxml2/libxslt),
2. Xalan Java Version 2,
3. Saxon 6.2.2,
4. XT/XP.

Chapter 5

XSL Translators Use

This chapter gives standard intall procedure examples for the most usual translators: XT, Xalan, xsltproc and Saxon. Besides, it tries to list the particularities in using each of them.

5.1 XT/XP

5.1.1 Installing XT/XP

XT is a translator that seems a bit lighter than Xalan. However some features are missing. Anyway, HTML transformation works without problem provided that the N. Walsh XSL stylesheets used are compliant to this tool. Here is an install procedure example:

1. First, check that java is installed.
2. Download the XT distribution.
3. Download an XML parser that supports SAX, let's say XP.
4. Unzip de XT/XP packages:

```
% mkdir /usr/local/xt /usr/local/xp
% unzip xt.zip -d /usr/local/xt
% unzip xp.zip -d /usr/local/xp
```

5. Put the XT libraries (xt.jar and sax.jar) and the XP library (xp.jar) in your CLASSPATH:

```
% CLASSPATH=$CLASSPATH:/usr/local/xt/xt.jar:/usr/local/xt/sax.jar
% CLASSPATH=$CLASSPATH:/usr/local/xp/xp.jar
% export CLASSPATH
```

5.1.2 Publishing in HTML with XT

5.1.2.1 Creating a single HTML file

To create a single HTML file use the `html/docbook.xsl` stylesheet:

```
java com.jclark.xsl.sax.Driver mydoc.xml (path)/docbook/htm/docbook.xsl > mydoc.html
```

5.1.2.2 Chunking with XT

Chunking with XT is possible by using the dedicated `html/xtchunk.xsl` stylesheet. The stylesheet creates a file for each part, chapter and section named automatically with their identifier.

```
java com.jclark.xsl.sax.Driver mydoc.xml (path)/docbook/htm/xtchunk.xsl > mydoc.html
```

5.2 Xalan J2

Xalan Java 1.x should not be used anymore, since the current XSL stylesheets cannot work with these versions anymore. I suggest to use Xalan Java 2.0.1, that works fine on my machine.

5.2.1 Installing Xalan J2

1. First, check that java is installed.
2. Download the package from the Xalan page. You don't need to download an XML parser since Xalan provides the Xerces java archive.
3. Unzip the package:

   ```
   % cd /usr/local/xslt/
   % tar xvzf xalan-j_2_*_*.tar.gz
   ```

4. Put the Xalan java archives in your CLASSPATH:

   ```
   % CLASSPATH=$CLASSPATH:/usr/local/xslt/xalan-j_2_*_*/bin/xalan.jar
   % CLASSPATH=$CLASSPATH:/usr/local/xslt/xalan-j_2_*_*/bin/xerces.jar
   % export CLASSPATH
   ```

5. Some extensions are sometimes required by the stylesheets (e.g, for tables). Add the DocBook XSL Xalan2 extension archive in your CLASSPATH:

   ```
   % CLASSPATH=$CLASSPATH:/(path)/docbook/extensions/xalan2.jar
   ```

5.2.2 Publishing in HTML with Xalan

5.2.2.1 Particularity

Absolute paths for XSL stylesheets must be prefixed by `file://` . For instance, using the HTML stylesheet with Xalan looks like:

```
java org.apache.xalan.xslt.Process -in mydoc.xml -xsl file:///path/to/docbook/html/docbo
```

5.2.2.2 Creating a single HTML file

To create a single HTML file use the `html/docbook.xsl` stylesheet:

```
java org.apache.xalan.xslt.Process -in mydoc.xml -xsl file:///(path)/html/docbook.xsl -
out mydoc.html
```

5.2.2.3 Chunking with Xalan

You can chunk by using the `html/chunk.xsl` stylesheet. The stylesheet creates a file for each part, chapter and section named automatically with their identifier.

```
java org.apache.xalan.xslt.Process -in mydoc.xml -xsl file:///(path)/html/chunk.xsl
```

5.3 xsltproc – The Gnome Toolkit

xsltproc is the XSLT program example provided with the libxslt library from Gnome. The libxslt/libxml2 libraries development is a Gnome project, written entirely in C (not in C++ nor in Java). This default XSLT is good enough to publish the XML documents, but one can always create its own program linked to the Gnome libraries. The big advantage of these libraries are:

- It is really fast,
- No java virtual machine is needed,
- It seems closer to the XML/XSL specifications.

5.3.1 Installing the Gnome libraries

Here is an install procedure example, for those installing the packages from the gzipped sources:

1. Download the libxml2 library from the XML page.
2. Unpack the package:

```
% tar xvfz libxml2-<version>.tar.gz
```

3. Configure and build the binaries:

```
% cd libxml2-<version>
% ./configure
% make
```

4. Install the package:

```
% make install
```

5. Download the libxslt library from the XSLT page.
6. Unpack the package:

```
% tar xvfz libxslt-<version>.tar.gz
```

7. Configure and build the binaries:

```
% cd libxslt-<version>
% ./configure
% make
```

8. Install the package:

```
% make install
```

Once done, xsltproc is available, and the libxml2/libxslt libaries are installed.

5.3.2 Publishing in HTML with xsltproc

5.3.2.1 Catalog resolution

Libxml2 supports catalog resolution, and thus xsltproc supports this feature too. To use you just need to:

1. Define a catalog that gives the XML DocBook DTD path. For example the catalog would look like this:

```
 -- Catalog to find the DTD DocBook XML file--

PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
    "file:///path/to/docbook/docbkx412/docbookx.dtd"
```

2. Add this catalog location in the SGML_CATALOG_FILES variable:

```
export SGML_CATALOG_FILES=$SGML_CATALOG_FILES:/path/of/the/catalog
```

3. Call xsltproc with the option –catalogs.

5.3.2.2 Creating a single HTML file

To create a single HTML file use the `html/docbook.xsl` stylesheet:

```
xsltproc --catalogs /(path)/html/docbook.xsl mydoc.xml > mydoc.html
```

5.3.2.3 Chunking with xsltproc

You can chunk by using the `html/chunk.xsl` stylesheet. The stylesheet creates a file for each part, chapter and section named automatically with their identifier.

```
xsltproc --catalogs /(path)/html/chunk.xsl mydoc.xml
```

5.4 Saxon

TDB.