Docbook In Context

A Context XML Mapping

for Docbook Documents

Simon Pepping

EuroTeX 2003, Brest, 24 June 2003

close

1 Usage

1.1 What is Docbook In ConTEXt?

Two technologies

- Docbook: authoring, structuring
- ConTEXt: layout, rendering

brought together

1.1.1 What is Docbook?

Docbook is an *extensive* DTD for technical literature, books and articles. It is becoming more and more popular for software documentation, e.g. the Linux Documentation Project.

The Docbook DTD files:

docbookx.dtd	The main DTD file
dbhierx.mod	The hierarchical elements: book, article, chapter, etc.
dbpoolx.mod	The pool of other elements
dbcentx.mod	The character entities
dbgenent.mod	The generic entities
dbnotnx.mod	The notations
calstblx.dtd	The CALS table model
soextblx.dtd	The SO Exchange table model (not used)

1.1.2 A short Docbook article

```
<?xml version="1.0" ?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"
                         "docbookx.dtd" []>
<article>

<articleinfo>
<title>DocBook In ConTeXt, ConTeXt XML mapping for DocBook
documents</title>
<authorgroup>
<author>
<firstname>Simon</firstname>
<surname>Pepping</surname>
</author>
<author>
<firstname>Michael</firstname>
<surname>Wiedmann</surname>
</author>
</authorgroup>
</articleinfo>
```

```
<section>
<title>Installation</title>
<para role="first">Change directory to the top directory
of one of the <literal>texmf</literal> trees of your
TeX installation, e.g. <filename>/usr/share/texmf</filename>,
and <command>untar</command> the distribution file
<filename>DocbookInContext.tar.gz</filename>. Then run the command
<command>mktexlsr</command> for that tree,
e.g. <command>mktexlsr /usr/share/texmf</command>.</para>
</section>

<section>
<title>Usage</title>
<programlisting>
\input xtag-docbook
\setupheadertexts[section][pagenumber]
\setupheader[leftwidth=.7\hsize,style=slanted]
</programlisting>
</para>
</section>

</article>
```

For the result see **README**.

close

1.2 How did it start and where is it now?

Start

EuroTeX 2001, Kerkrade: ConTEXt presentations

Context mailing list: ConTEXt XML input

That made me curious enough to dive into it.

Michael Wiedmann was interested and supported me to go on.

Now

How is ConTEXt possible?

Theoretically TEX macro programming is complete. Hans Hagen can turn this theory into practice.

Docbook In ConTEXt works for a number of frequently used elements.

Docbook In ConTEXt has a framework for some fundamental issues.

1.3 Running Docbook In ConTEXt

To run a file `myfile.xml`, create a driver file `myfile.tex`:

```
\input xtag-docbook

\starttext
\processXMLfilegrouped{\jobname.xml}
\stoptext
```

and run it:

```
$ texexec myfile.tex
```

Without a driver file:

```
texexec --xmlfilter=dbk myfile.xml
```

But that would only work if the module were called `xtag-dbk.tex`

1.4 Customizing Docbook In ConTEXt

Docbook In ConTEXt creates a ConTEXt file on the fly. Customize it in the usual way (in the driver file):

```
\input xtag-docbook
\setupindenting[medium]
\setupheadertexts[section][pagenumber]
\setupheader[leftwidth=.7\hsize,style=slanted]
\setuppagenumbering[location=]
\setupitemize[each][packed][before=,after=,indentnext=no]
\setupcombinedlist[content][level=section]
\setuphead[subsection][number=no]

\starttext
\processXMLfilegrouped{\jobname.xml}
\stoptext
```

There are also DIC-specific customization options.

1.4.1 Section blocks

- `\setupXMLDB[pagebreaks=all]`: Default ConTEXt behaviour.
- `\setupXMLDB[pagebreaks=sectionblocks]`: ToC and Index do not start a new page, and they are treated as sections. All other section blocks retain their default ConTEXt behaviour.
- `\setupXMLDB[pagebreaks=none]`: In addition to the `sectionblocks` option, bodymatter, appendices and backmatter do not start a new page.

close

1.4.2 Titles

```
\def\XMLDBarticletitle#1%
   {\startalignment[left]\bfb #1\stopalignment \blank}

\def\XMLDBabstracttitle#1%
   {\blank[big]\midaligned{\bf #1}\blank[medium]}

\def\XMLDBrevhistorytitle#1%
   {\blank[big]\midaligned{\bf #1}\blank[medium]}
```

1.4.3 Section titles

Chapter, section, subsection, etc. titles are mapped onto ConTEXt's usual sectioning commands:

```
\def\XMLDBchaptertitle{\chapter[\XMLpar{\XMLparent}{id}{}]}

% chapter or section
\def\XMLDBappendixtitle{\XMLDBmakechapter}

\expandafter\def\csname XMLDBsect1title\endcsname%
   {\section[\XMLpar{\XMLparent}{id}{}]}

% nested sections
\expandafter\def\csname XMLDBsectionhead-1\endcsname%
   {\section[\XMLpar{\XMLparent}{id}{}]}
```

They can be customized in ConTEXt, e.g.

```
\setupcombinedlist[content][level=section]
\setuphead[subsection][number=no]
```

1.4.4 blockquote, epigraph and attribution

```
\setupblockquote[narrower=middle,quote=on,command={--- \it}]
```

- `narrower`. Both `epigraph` and `blockquote` are formatted using ConTEXt's narrower environment. The value of this option is a list of `left`, `right` and `middle` that is passed on to the `\startnarrower` command. See the ConTEXt documentation for `\startnarrower` for the effect of these settings.
- `quote`. The value is `on` or `off`. When on, quotation marks are applied as with ConTEXt's quotation environment.
- `command`. The value is a command or set of commands, which are applied at the start of the `narrower` environment.

1.4.5 Unconstrained attribute values

Example: The `role` attribute of any element.

Preprogrammed actions are not possible, because the possible values are not known.

Insert a hook in the stylesheet for the user's own formatting command.

Stylesheet: `\XMLattributeaction[para][role]`

User: `\defineXMLattributeaction[para][role]action`

Example:

- Customization command:

  ```
  \defineXMLattributeaction[para][role][first]{\bf}
  ```

- In the XML file:

  ```
  <para role="first">
  ```

1.4.6 Example

We add a number of DIC-specific customization options:

```
\input xtag-docbook
\setupindenting[medium]
\setupheadertexts[section][pagenumber]
\setupheader[leftwidth=.7\hsize,style=slanted]
\setuppagenumbering[location=]
\setupitemize[each][packed][before=,after=,indentnext=no]
\setupcombinedlist[content][level=section]
\setuphead[subsection][number=no]

% customizations
\setuphead[section][style=bia,number=no,align=right]
\setupepigraph[narrower={1*right},command=\bi]
\setupattribution[command=---]
\setupXMLDBlists[notoc]
\setupXMLDB[background=off]
\def\XMLDBarticleinfotitle#1%
    {\startalignment[middle]\bib #1\stopalignment\blank[1*big]}
\defineXMLattributeaction[para][role][first]{\bf}
```

For the result see **README2**.

close

1.4.7 More customizations

will follow .... See the file **Customization**.

1.4.8 Writing your own module

```
\input xtag-docbook

\defineXMLenvironment[mediaobject]
   {\XMLDBpushelement\currentXMLelement \XMLDBmayensurebodymatter
    \bgroup
    \defineXMLignore[objectinfo]% processing suppressed
    \defineXMLsave[videoobject]%
    \defineXMLsave[audioobject]%
    \defineXMLsave[textobject]%
    \defineXMLsave[caption]}
   {\expanded
       {\placefigure
            [here,\XMLDBfigurealign]
            [\XMLpar{\XMLparent}{id}{}]
            {\XMLflush{title}}
            {\externalfigure[\XMLDBimagedata]
                 [factor=\XMLDBimagescalefit,%
                  scale=\XMLDBimagescale]}}%
    \egroup
    \XMLDBpopelement}
```

Example: **m-docbook** by Richard Rascher-Friesenhausen.

close

1.5 Other tools for the same task

- XML $\underset{\text{XSLT}}{\longrightarrow}$ FO $\underset{\text{FO processor}}{\longrightarrow}$ type
- XML $\underset{\text{XSLT}}{\longrightarrow}$ ConTEXt file $\underset{\text{ConTEXt}}{\longrightarrow}$ type

1.5.1 Canonical tool

XML $\underset{\text{XSLT}}{\longrightarrow}$ FO $\underset{\text{FO processor}}{\longrightarrow}$ type

XSLT stylesheets for Docbook by Norman Walsh:
large coverage, customization through extensive parametrization

XSLT + FO: one stylesheet, many processors

FO processors: FOP, `xmltex` + `passivetex`

`xmltex`: David Carlisle's XML processor

`passivetex`: Callbacks for FOs (DIC: callbacks for Docbook)

Neither FOP nor `passivetex` run Docbook XSLT stylesheets without errors

1.5.2 Non-orthodox tool

XML $\underset{\text{XSLT}}{\longrightarrow}$ ConTEXt file $\underset{\text{ConTEXt}}{\longrightarrow}$ type

`db2context`

Customizability: Edit the ConTEXt file.

2 Programming

close

2.1 Programming Docbook In ConTEXt

2.1.1 ConTEXt and XML

A typical mapping instruction:

```
\defineXMLenvironment[element]{start action}{stop action}.
```

Accessing the attribute values of the element:

```
\doifXMLvar{entry}{align}%
   {\expanded{\setupTABLE[align=\XMLvar{entry}{align}{}]}}
```

Timing of expansion requires attention: `\expanded`

2.1.2 It is easy, is it not?

A simple mapping:

```
\defineXMLenvironment[subtitle]
   {\startalignment[middle]\bfb}
   {\stopalignment\blank[2*big]}
```

A slightly less simple mapping. Generate the correct separators and pay attention to the spaces:

```
\defineXMLenvironment[firstname]{\XMLDBseparator}{\XMLDBdospaces}
\defineXMLenvironment[surname]{\XMLDBseparator}{\XMLDBdospaces}
```

2.2 Encoding and language

Declaration of encoding:

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

Reading the encoding:

```
\defineXMLprocessor[xml] \setencoding
\def\setencoding#1{\dogetXMLarguments{xml}#1>
    \setevalue{\??xmldbenc}{\XMLvar{xml}{encoding}{utf}}}
```

Declaration of language:

```
<article lang="de">
```

Reading the language:

```
\XMLDBstartdocument{\XMLpar{\currentXMLelement}{lang}{en}}
```

Using it all:

```
\def\XMLDBstartdocument#1{%
    \expanded{\enableregime[\getvalue{\??xmldbenc}]}
    \mainlanguage[#1]%
    \disableXML\readfile{xtag-docbook-literals-#1}{}{}\enableXML
}
```

2.2.1 The literal strings

```
\def\XMLDBAbstract{Zusammenfassung}
\def\XMLDBabstract{Zusammenfassung}
\def\XMLDBAnswer{A:}
\def\XMLDBanswer{A:}
\def\XMLDBGlossSeeAlso{Siehe Auch}
\def\XMLDBGlossSeealso{Siehe auch}
\def\XMLDBglossseealso{siehe auch}
```

Compare the **english** and **german** versions of the same text.

2.3 Features for each element

2.3.1 Context stack

```
\defineXMLenvironment[xxx]
   {\XMLDBpushelement{\currentXMLelement}}
   {\XMLDBpopelement}
```

Access to the stack:

- `\XMLDBcurrentelement`: The current element's name.
- `\XMLancestor#1`: The name of the ancestor at level `#1` The current element is at level 0.
- `\XMLparent`: The name of the current element's parent.
- `\the\XMLdepth`: The depth of the context stack.
- `\doifXMLdepth#1`: Execute the following instruction if the context stack has a certain depth.
- `\XMLDBprintcontext`: Print the context stack in the log file (mainly for debugging purposes).

  Example

  XMLcontext : article, section, section, variablelist, para, itemizedlist

2.3.2 Ignorable white space

```
<author>
   <firstname>Simon</firstname>
   <surname>Pepping</surname>
</author>

\processcommacommand[articleinfo,authorgroup,author,affiliation]%
    \defineXMLDBstripspace

\defineXMLenvironment[xxx]
   {\XMLDBpushelement{\currentXMLelement} \XMLDBdospaces}
   {\XMLDBpopelement \XMLDBdospaces}
```

- Ignore spaces in element `xxx` if applicable
- Ignore spaces in the parent if applicable

```
<para><!-- Do not ignore spaces at start -->2nd
description.<indexterm><!-- Ignore spaces at start -->
  <primary>Some term</primary>
</indexterm><!-- Do not ignore spaces at end --> More
text.</para><!-- Ignore? spaces at end -->
```

2.3.3 Every element

```
\defineXMLenvironment[xxx][id=\undefined]
   {\XMLDBpushelement\currentXMLelement
    \XMLDBseparator \XMLDBdospaces}
   {\XMLDBpopelement \XMLDBdospaces}
```

- Clear out the `id` attribute
- Push the element on the context stack
- Place the separator, if any
- Ignore spaces if applicable
- Pop the element from the context stack
- Ignore spaces if applicable

2.4 Which element is next?

2.4.1 Is there a title?

Abstract may but need not have a title. If it does not have a title, I want to print a default title ‘Abstract’. Similarly for Preface.

```
<abstract>
<!-- optional title -->
```

⇐ How do I know whether I am past a possible title?

```
<!-- formalpara or para or simpara -->
</abstract>
```

2.4.1.1 Implementations

- Option 1:
  - Let element `title` store its value in a macro.
  - Redefine `para`, `formalpara`, `simpara` to typeset the title or the default title. Then reset to original mapping.

  Not very generic; in `legalnotice`, `preface` other elements need to be redefined.

- Option 2:
  - Output abstract in `\vbox`.
  - Let element `title` store its value in a macro.
  - Typeset title or default title.
  - `\unvbox` the abstract.

  A `\vbox` spoils vertical spacing.

- Option3 :
  - Save the abstract.
  - Scan the text of the abstract for the word `<title`.
  - If it does not occur, typeset the default title.
  - Typeset the abstract.

  Saving text makes it impossible to redefine `\catcode`s.

Option 3 is currently used.

2.4.2 The title comes later

`chapter`, `section`, `figure`, `table` do have a required title. But the title comes later:

```
<section>
  <title>The title comes later</title>
    <para><code>chapter</code>, etc.
```

XML and TEX have a different approach to titles and the parts they belong to.

In XML they are separated, in TEX they are combined in one command.

This is the difference between a structuring language and an authoring language.

2.4.3 Sectioning

A ConTEXt document consists of frontmatter, bodymatter, appendices and backmatter, which are called section blocks.

A Docbook document does not have such parts.

The first element that cannot be in frontmatter, starts bodymatter.

A Docbook `book`'s bodymatter starts with the first `part`, `chapter`, `article` or `reference`.

A Docbook `article`'s bodymatter starts with the first `calloutlist`, `glosslist`, `itemizedlist`, `orderedlist`, `segmentedlist`, `simplelist`, `variablelist`, `caution`, etc. (56 elements).

All these elements execute `\mayensurebodymatter`:

If they are at nesting depth 2, and we are still in the front matter, close front matter and open body matter.

Similarly for the other section blocks.

This is one case where TEX grouping runs counter to the XML tree structure: the start of a node closes a TEX group. It makes the name of the current element (`\currentXMLelement`) and its attribute values disappear.

2.5 Specific elements

2.5.1 Tables

Docbook uses the CALS table model. ConTEXt uses its TABLE environment, also called natural tables. Both are rather similar.

There are three main complications.

- The `frame` attribute of the CALS table has no equivalent in ConTEXt.
- Multiple `tgroup` elements, each with their own number of columns, and their own alignment and frame settings.
- Each `tgroup` may have its own `thead` and `tfoot` elements, with their own alignment and frame settings.

Solution:

The `table` element generates a ConTEXt table, i.e. the table float, using the `\placetable` command.

Each `tgroup` element generates its own TABLE environment, i.e. the actual table.

The table is not openend by the start tag of the `table`, because at that moment the title is not yet known.

The TABLE is not ended by the end tag of the `tgroup`, because we do not know if it is the last `tgroup`, which has the bottom frame.

2.5.1.1 Example table

A table with three `tgroups`:

	1	2	3	4
	A	B	C	D
EEEE		F	G	H
	I	J	K	L
	M	N	O	P
	1	2	3	4

1		2	3
A	B		C
E	FFFF		G
I	J		K
M	N		O

1	2		3	4
A	B	C		D
E	F	GGGG		H
I	J	K		L
M	N	O		P
1	2		3	4

2.5.2 Revision history

Revision History

Revision	Date	Remark
0.1	27 December 2002	First draft for MAPS
0.2	31 January 2003	Final version for MAPS
0.3	31 March 2003	Presentation for DANTE meeting
0.4	20 June 2003	Presentation for EuroTEX2003

Of the five possible columns `revnumber`, `date`, `authorinitials`, `revdescription`, `revremark` only those are printed which have data.

This is achieved by processing the revision history twice.

- Save the revision history.
- Define the elements such that the revisions are counted and the used columns are registered.
- First pass.
- Redefine the elements such that the table is typeset, with the columns used.
- Second pass.

Reprocessing is a powerful feature of TEX macro processing. It is used often in ConTEXt. It takes some time before one has a good grasp of this pattern.

2.5.3 Program listing

Statement: `programlisting` is `verbatim`:

```
<programlisting>
   #include "string.h"

   void *memset    (void *s, int c, size_t n);
</programlisting>
```

Not quite, it does enable XML markup:

```
<programlisting>
   #include &lt;string.h>

   void *memset    (void *s, int c, size_t n);
</programlisting>
```

2.5.4 CDATA

Statement: `CDATA` is `verbatim`:

```
<![CDATA[
   #include <string.h>

   void *memset(void *s, int c, size_t n);
]]>
```

Not quite, it just disables XML markup:

```
<para><literal>#include <![CDATA[<string.h>]]></literal>
includes a system header file.</para>
```

Conclusion:

- `programlisting` indicates line oriented layout,
- `CDATA` disables XML markup.

2.5.5 Line oriented layout

ConTEXt's line oriented layout macros use line scanning. The line after `</programlisting>` is scanned with the wrong `\catcode`s. That could produce extra linebreaks:

```
<para>The line <programlisting>
   #include "string.h"
</programlisting>includes
a system header file.</para>
```

`programlisting` uses some simple macros to enable line oriented layout.

```
\def\obeyedline{\strut\par}
\def\obeyedspace{\strut\space}
```

Active `^^M` characters and `\struts` take care of line oriented layout, preserving spaces at the start of the line.

2.5.6 Hyperlinks, URLs and external documents

Two types of links:

- external documents, i.e. local PDF documents, ConTEXt's `\useexternaldocument`;
- web documents and non-PDF local documents, ConTEXt's `\useURL`.

Requires analysis of the given link:

- Is there a scheme (e.g. `http`)?
- If not, or if the scheme is `file`, it is a local file.
- If it is a local file, is it a PDF file? If so, use `\useexternaldocument`.
- If it is not a PDF file, if the URI is relative, make it complete.
- If it is not a local file, or if it is not a PDF file, use `\useURL`.

2.5.6.1 Examples of `ulink` URLs

Local PDF files:

```
<ulink url="file://localhost/DIC/SAX-doc.pdf">SAX-doc.pdf</ulink>
<ulink url="/DIC/SAX-doc.pdf">SAX-doc.pdf</ulink>
<ulink url="SAX-doc.pdf">SAX-doc.pdf</ulink>
```

Scheme `http`:

```
<ulink url="http://www.hobby.nl/DIC/SAX-doc.html">SAX-doc.html</ulink>
<ulink url="http://localhost/DIC/SAX-doc.html">SAX-doc.html</ulink>
```

Local non-PDF files (scheme `file`):

```
<ulink url="/DIC/SAX-doc.html">SAX-doc.html</ulink>
<ulink url="SAX-doc.html">SAX-doc.html</ulink>
```

Problem: Are these (abbreviated) URLs or local files?

```
<ulink url="www.dante.de">DANTE</ulink>
<ulink url="dante.html">DANTE program</ulink>
```

User may choose with `\XMLDBcheckabbrURItrue` or `\XMLDBcheckabbrURIfalse`.

3 Next, Where, Who

close

3.1 Future plans

- Docbook In ConTEXt should be integrated in the ConTEXt distribution.
- Docbook is a complex DTD. Presenting Docbook documents is therefore a complicated task. Currently there are three efforts to do so:
  - Docbook XSLT stylesheets
  - Docbook in ConTEXt
  - Docbook to ConTEXt via XSLT

  Why so many efforts to present Docbook? Large user communities can support multiple solutions to the same problem. It depends on the user community, not only on me.

  Docbook in ConTEXt could be further developed

  - as a ConTEXt module, or
  - as an Open Source project on Sourceforge.
- Other useful efforts:
  - TEX as a FO processor (SR's passivetex)
  - Unicode enabled TEX
  - Extensible TEX

  I want to spend more attention to these efforts.

3.2 Availability

Docbook In ConTEXt is available separately from the ConTEXt distribution, from my web site `http://www.hobby.nl/~scaprea/context`.

Michael Wiedmann's web page with Docbook tools `http://www.miwie.org/db-context/index.html` has a link to the Docbook In ConTEXt files.

3.3 Acknowledgement

Michael Wiedmann contributed the mappings for several elements, a.o. `ulink`, `table` and `mediaobject`.

He also contributed the implementation of string literal files, and the string literals for English and German.

Giuseppe Bilotta contributed the string literals file for Italian.

Pablo Rodriguez contributed the string literals file for Spanish.

Richard Rascher-Friesenhausen contributed the mappings for several elements, and came up with the idea of a customized module.

He also contributed a well-organized framework for the documentation of the Docbook In ConTEXt, which I intend to apply.

And of course, nothing of this would have been possible without Hans Hagen's ConTEXt. ConTEXt is the framework upon which Docbook In ConTEXt runs and a rich source of examples of excellent TEX macro programming.