xtpxlib/xdocbook module - DocBook 5 xtpxlib-xdoc dialect description

Erik Siegel - Xatapult - 2019-11-07 10:45:53

1 Introduction

The `xtpxlib-xdoc` component (https://github.com/eriksiegel/xtpxlib-xdoc) uses DocBook 5 as its target vocabulary. However, it does not implement the full standard (which is huge!) but only those elements/ attributes that were deemed necessary. This document will explain what is in and what's not.

2 The xtpxlib Docbook 5 dialect

2.1 Supported root elements

Both the `<book>` and the `<article>` root element are supported. A `<book>` root results in paged output (for, as the name implies, a book. The `<article>` root results in something more memo style (like this).

2.2 Document information

Document information: The only document information elements recognized are (any others are ignored):

```
<info>

  <title> ... main title ... </title>
  <subtitle> ... subtitle ...</subtitle>
  <pubdate> ... publication date ... </pubdate>
  <author>
    <personname> ... author name ...</personname>
  </author>

  <orgname> ... organization ... </orgname>

  <mediaobject role="top-logo">
    <!-- Use either role="top-logo" or no role attribute. -->
    <imageobject>
      <imagedata fileref="..." width="...(opt)" height="...(opt)"/>
    </imageobject>
  </mediaobject>

  <mediaobject role="center-page">
    <imageobject>
      <imagedata fileref="..." width="...(opt)" height="...(opt)"/>
    </imageobject>
  </mediaobject>

</info>
```

All elements are optional.

2.3 Chapter/Section structure

- For books, `<preface>`, `<chapter>`, `<appendix>` and `<sect1>` to `<sect3>` are recognized and handled.
- In articles only `<sect1>` to `<sect3>` are allowed.

2.4 Block constructions

the following block level constructions are recognized and handled:

- **Paragraphs**: Normal `<para>` elements recognize the following role attribute values (multiple, whitespace separated, values allowed):

`@role` value	Description
`break`, `smallbreak`	Inserts an empty line, either full or small height. The contents of the `<para>` element is ignored.
`break-before` `break-after`	Adds extra whitespace before or after the paragraph
`header` `keep-with-next`	Keeps this paragraph with the next one together on a page.
`keep-with-previous`	Keeps this paragraph with the previous one together on a page.

Table 2-1

- **Lists**: Both `<itemizedlist>` and `<orderedlist>` are allowed.

- **Tables**: Both `<table>` and `<informaltable>` are allowed. An example of a formal table above. An informal table below.

Example	of
an	informal table

Add `role="nonumber"` to a table to stop it from getting a number:

Blurp	Blorb
Example	of
an	unnumbered table

Unnumbered table

Tables are notoriously difficult in that FOP cannot compute column widths automatically. To amand this (a little bit) add `colspec/@colwidth` information. There is also a mechanism for columns with code (set in a fixed-width font), see "Fixed-width column mechanism" on page 5.

- **Program listings**: For program listings use the `<programlisting>` element
  The easiest way to handle this turned out to put longer program listings in external files and use an `<xi:include parse="text">` construction:
  `<programlisting><xi:include href="ref" parse="text"/></programlisting>`
  Or use a `<![CDATA[` construction around the piece of code.

- **Figures**: Both `<figure>` and `<informalfigure>` are allowed. Width and height can be set on the image data.

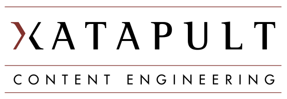

Figure 2-1 - An example of a figure... `(this in fixed width)`

Add `role="nonumber"` to a figure to stop it from getting a number.

- **Bridgeheads**: The `<bridgehead>` element inserts a bridgehead paragraph (bold, underlined and with an empty line before):

<u>This is a bridgehead...</u>

- **Simple lists**: The `<simplelist>` element inserts a simple list:

  An entry
  Another entry...

- **Variable lists**: The `<variablelist>` element inserts a variable list list (also very useful for explaining terms, definitons, etc.):

  The first entry
  The explanation of the first entry!

  The second entry
  The explanation of the second entry!

- **Notes and warnings**: The `<note>` element inserts a note and `<warning>` a warning:

  NOTE:
  This is a note! Nulla ac ex urna. Ut auctor odio quis nulla porta bibendum. Proin hendrerit molestie velit sit amet tristique. Vivamus laoreet ligula leo, vitae placerat ipsum porta sed. Morbi blandit ex mauris, eu volutpat tortor mattis eu. Nam et molestie mi. Aliquam erat volutpat. Aenean a imperdiet lectus. Phasellus condimentum dignissim laoreet.

  WARNING:
  This is a warning! Nulla ac ex urna. Ut auctor odio quis nulla porta bibendum. Proin hendrerit molestie velit sit amet tristique. Vivamus laoreet ligula leo, vitae placerat ipsum porta sed. Morbi blandit ex mauris, eu volutpat tortor mattis eu. Nam et molestie mi. Aliquam erat volutpat. Aenean a imperdiet lectus. Phasellus condimentum dignissim laoreet.

  If you add a `<title>` element, the standard title (NOTE/WARNING) will be replaced by its contents.

- **Sidebars**: The `<sidebar>` element inserts sidebar section:

  Title of the sidebar
  Contents of the sidebar. Nulla ac ex urna. Ut auctor odio quis nulla porta bibendum. Proin hendrerit molestie velit sit amet tristique. Vivamus laoreet ligula leo, vitae placerat ipsum porta sed. Morbi blandit ex mauris, eu volutpat tortor mattis eu. Nam et molestie mi. Aliquam erat volutpat. Aenean a imperdiet lectus. Phasellus condimentum dignissim laoreet.

- **Examples**: The `<example>` element inserts an example:

  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus elementum diam nec nunc elementum, eget dapibus dui malesuada. Aenean facilisis consequat odio, vitae euismod eros tempor nec. Vestibulum cursus tortor tortor, semper euismod sapien sagittis et.
  Example 2-1 - Example of an example

  Add `role="nonumber"` to an example to stop it from getting a number.

2.5 Inline elements

the following inline elements are recognized and handled:

- **`<emphasis>`**: Sets *emphasis*.
  Use **`role="bold"`** or `role="underline"` to set a specific type of emphasis.

- **`<literal>`** or **`<code>`**: Creates a piece of literal, `mono-spaced` text.

- **`<link>`**: Outputs some link (e.g. a web address). This is either the contents of the element or, if not available, the contents of the `xlink:href` attribute.
  Like this or like this http://www.xatapult.nl.

- **`<inlinemediaobject>`**: Inserts an inline image , like this.

- **`<citation>`**: Inserts a citation between square brackets like this: [CITATION].

- **`<command>`**: Use to indicate an exetuble program or a user provided command, like this: *git checkout origin*

- **`<email>`**: Use to indicate an an email address, like this: *info@xatapult.com*

- **`<filename>`**: Use to indicate an a filename, like this: *blabla.xml*

- **`<replaceable>`**: Use to indicate text to be replaced with user or context supplied values, like this: *`add your own stuff here`*

- **`<keycap>`**: Use to indicate a keyboard physical key, like this: Return

- **`<superscript>`, `<subcript>`**: For super- and subscripts, like this: XX^{super} YY$_{sub}$

- **`<userinput>`**: Use to indicate data entered by the user, like this: **`data entered here`**

- **`<tag>`**: Indicates an object from the XML vocabulary. The `class` attribute signifies what:

`@class` value	Result example(s)
`attribute`	`@attribute` `@class`

`attvalue`	`"attribute value"`
	`"some value for an attribute"`

`emptytag`	`<element/>`
	`<docbook/>`

`endtag`	`</element>`
	`</docbook>`

`pi`	`<?processing-instruction x="y"?>`
`comment`	`<!-- Some comment line... -->`

Anything else defaults to `element`	`<element>`
	`<docbook>`

Table 2-3

- **`<xref>`**: Inserts a cross-reference to the id referenced by `@linkend`
  - Use `role="page-number-only"` to get just a page number.
  - Use `role="simple"` to always get: page #

- Use `role="text"` to only get the (unquoted) text only in cases where a "…" on page … would normally appear.
- Use `role="capitalize"` to force the reference string (for chapters/appendices/pages/figures/tables/…) to start with an upper-case character (so you can be sure a sentence that starts with an `<xref>` always starts with a capital).

Otherwise it depends on what is pointed to:

Target	Result/Examples
To anything that holds an `xreflabel` attribute	"paragraph with xreflabel attribute" on page 1
To a chapter or appendix	chapter # or appendix #
To a section	"Document information" on page 1
To a table (with a number), like this one	table 2-4
To a figure (with a number)	figure 2-1
To an example (with a number)	example 2-1
To anything else	First paragraph: page 1 Unnumbered table: page 2

Table 2-4 - Examples of `<xref>` usage

2.6 Other constructs

- **To-be-done marker**: Start a to-be-done marker with **`[TBD`** and end it with **`]`**. For instance: [TBD this needs to be done...]

2.7 Fixed-width column mechanism

FOP (in the current version, 3Q19) cannot compute the column-widths automatically. It divides the space and you can set a fixed column-width (with `colspec/@colwidth`). For the case that a column contains code stuff (text in a fixed-width font) *and* you want the column-width to be dependent on the text in such a column, there is a (unfortunately a bit complicated) mechanism for this.

The fixed-width column mechanism consists of two parts:

Dynamically compute the column width

This part is optional.

Add a `role` attribute to the `<colspec>` element with, as one of the roles, `code-width-cm:min-max`, where `min` and `max` are (positive) doubles. For instance `<colspec role="code-width-cm:1.2-4">`. `min` and `max` are the minimum and maximum column-widths, expressed in cm.

The PDF conversion will now look in all the contents of this particular column for entries `<code role="code-width-limited">`. Based on the length of these entries it computes an optimal column-width, but always between `min` and `max`.

Output code width-limited

If a table entry contains contents in a `<code role="code-width-limited">` element, it tries to make it fit within the available column-width. If necessary the line is split to prevent overflowing of table cell contents.

This is (currently) not completely fool-proof: if the contents contains whitespace or hyphens, it is assumed to line-break correctly by itself. That, of course, does not guarantee correct results. So it may need a little experimenting before things look right.

A column that contains `<code role="code-width-limited">` contents *must* have a column width set *in cm* (either directly with `<colspec colwidth="…cm">` or by the dynamic mechanism described above).