Using XSLT to Fix lyX's Docbook XML

Chad Albers

August 6, 2008

1. lyX and Docbook XML

When I write software documentation, I use the lyX editor. Its simplicity allows me to concentrate on the content of what I write, rather than what my writing will look like (or it's format). I prefer lyX as well because it can export documentation into the Docbook XML format. I can then use Docbook XSL stylesheets to convert it into XHTML or PDF documents. There are a few things to consider when using lyX to create Docbook XML documents.

1.1. lyX's Limitations

- lyX was really designed to edit LaTex documents, not XML documents.
- When lyx's creates a Docbook article using lyX's template mechanism, it actually creates an SGML document, not an XML document. lyX produces the XML version of the document when you export it by selecting the "File" menu "Export" menu item. The only difference between the SGML version that is edited and the XML version that is exported is the Doctype header stanza; the elements in both formats are identical.
- In the Docbook XML specification, only certain markup elements can be nested under other elements. lyX, however, does not restrict were elements are inserted. Since lyX does not enforce the specification, it is quite possible to introduce XML elements that are in the wrong "position" in your Docbook XML document (or, in XML-speak, to create a document that is “invalid.”).
- The Docbook XML markup that lyX produces is version 4.2. The current version is 5.0.
- The suite of Docbook XSL stylesheets managed by Docbook.org were written for version 5.0, although they are backward compatible.

Taking these into consideration, if you use lyX to create Docbook XML documents, the XML format is out of date, possibly contains invalid markup, and may be unsuitable for XSLT transformations into PDF or XHTML documents.

2. Mitigating lyX's Docbook Problems

To work around some of lyX's Docbook XML problems, you could do the following:

- Follow these HOWTOs: here and here. The former deals with Docbook XML and the later deals with Docbook SGML. Both can be helpful.
- Validate the document using xmllint. When it finds errors, you will need to consult the Docbook XML specification to correct the errors manually. If you do not want to maintain multiple versions of your document (the lyX version and the XML version), you will have to go back to your lyX document and figure out why lyX is producing incorrect markup.
- Convert your Docbook XML 4.x to 5.0 by following these instructions. This may prove useful, but I have notneeded it.

Since Docbook XML is a based on XML technologies, I recommend instead leveraging those technologies to solve lyX's problems. The suite of Docbook XSL stylesheets mentioned above can be customized to deal with lyX's quirks. They can also be used to dramatically alter the PDF and XHTML versions of your Docbook documents. I have used Bob Stayton's online book called *DocBook XSL: The Complete Guide* as my reference. The next example applies this technique.

2.1. lyX's <date> element

Here's how to solve a problem with the `<date>` element that appears in lyX's Docbook XML. According to the Docbook specification, the `<date>` element is intended to describe the article's publication date. It is a child element of the `<articleinfo>` parent element. There is another element, `<pubdate>`, that performs the same function. Even though `<date>` is a valid Docbook XML element, the Docbook XSL stylesheets do not recognize the data delimited by this element, and, as a consquence, the data will not be included in the transitional format (XSL-FO) that a XSLT transformation produces (which will, in turn, be converted to a PDF document, using Apache's Formatting Objects Processor, fop). This problem can be fixed by writing a customization layer which manipulates the Docbook XSL stylesheets. The instructions below describe how.

1. Locate where the Docbook XSL stylesheets are stored on your system. In the Debian GNU/Linux distribution, they are located here:

   `/usr/share/xml/docbook/stylesheets/nwalsh`

2. Identify which XSL stylesheets are responsible for transforming Docbook XML to a XSL-FO stylesheet:

   `/usr/share/xml/docbook/stylesheets/nwalsh/fo`

3. Find the stylesheet (and the template) responsible for processing the `<pubdate>` element. I use grep.

   ```
   $> grep pubdate /usr/shar/xml/docbook/stylsheets/nwalsh/fo/*
   ```

   It was located in a file called, "`titlepage.template.xsl`".

   Figure 1.

   ```
   <xsl:apply-templates mode="article.titlepage.recto.auto.mode"
    select="articleinfo/pubdate"/>
   ```

   The template which makes this call is found in the same document.

   Figure 2.

   ```
   <xsl:template name="article.titlepage.recto">
   ```

4. Create a XSL stylesheet using your favorite text editor that contains a customization layer which, first, overrides the "`<xsl:template name="article.titlepage.recto">`" behavior and, second, contains a template that understands the `<date>` element.

5. To create the XSL stylesheet, start a new document using your favorite text editor and paste this line at the beginning of the document...

Figure 3.

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="1.0">
```

...and this line at the end...

Figure 4.

```
<xsl:stylesheet>
```

...everything you add to the stylesheet in the next steps should be added between these two lines.

6. Import the URI reference to the Docbook XSL stylesheets by adding this line (change the href to match the location of your Docbook XSL stylesheets):

   Figure 5.

   ```
   <xsl:import href="/usr/share/xml/docbook/stylesheet/nwalsh/fo/docbook.xsl"/
   >
   ```

7. In your text editor, open up the file called "`titlepage.template.xsl`" that you found in the previous step.

8. Copy the entire `<template>` stanza that calls the `<pubdate>` element. From...

   Figure 6.

   ```
   <xsl:template name="article.titlepage.recto">
   ```

   ...to...

   Figure 7.

   ```
   <xsl:template>
   ```

9. Paste this into your XSL stylesheet.

10. Inside the template that you just pasted into your XSL stylesheet, add a line to call a template specifically associated with the `<date>` (remember that it is nested inside an `<articleinfo>` element):

    Figure 8.

    ```
    <xsl:apply-templates mode="article.titlepage.recto.auto.mode"
     select="articleinfo/date"/>
    ```

11. Add a template in your XSL stylesheet that responds to the `<xsl:apply-templates>` call above:

Figure 9.

```
<xsl:template match="articleinfo/date"
 mode="article.titlepage.recto.auto.mode">
</xsl:template>
```

12. Decide how you would like the `<date>` data formatted using XSL-FO (fo) elements. Since it is the same data as a `<pubdate>` element, I looked for how the `<pubdate>` element was handled. Fortunately, it was in the `titlepage.template.xsl` file.

13. Cut and paste the `fo` element from "articleinfo/pubdate" template into your "articleinfo/date" template .

Figure 10.

```
<fo:block xmlns:fo="http://www.w3.org/1999/XSL/Format" xsl:use-attribute-
sets="set.titlepage.recto.style">
     <xsl:apply-templates select="." mode="set.titlepage.recto.mode"/>
</fo:block>
```

14. Now save your custom XSL stylesheet to, say, `my-customized-docbook-stylesheet.xsl`.

Using this XSL stylesheet, you can now produce a XSL-FO stylesheet that contains the data of a `<date>` element. I use `xsltproc`, (but xalan or saxon will work, too) to apply the custom stylesheet to my Docbook XML document.

```
$> xsltproc my-customized-docbook-stylesheet.xsl lyX-produced-
docbookXML-file.xml
```

An XSL-FO document (fo output) will be sent to the standard output. I save this output to a file by appending the following to the command line above: "`> output.fo`". Once you have have the XSL-FO document produced by these commands, you can now run the file through Apache's fop to produce your PDF document.

```
$> fop -fo output.fo -pdf output.pdf
```

3. Tutorial Documents, Samples, and Stylesheets

Following the method above, here is a link to a pdf version of this tutorial. Here is a link to a customized stylesheet that produced this sample. It recognizes the `<date>` element that lyX produces. The stylesheet also contains a few more tricks. It recognizes the `<command>` element (inserted using lyX's ERT function) and surrounds a command line with a grey box. It also handles source code samples that have been inserted into your lyX document as "Figures" and marked as "Code". You can see how these tricks were accomplished by viewing the original lyX document that this document is based on.

In addition, I am also releasing a customized Docbook XSL stylesheet that will produce XHTML version that is a near duplicate of the PDF version. The link is here.

All documents produced in this tutorial can be downloaded from here. A gpg signature for this file is located here. (My public key is on my website: http://www.neomantic.com.)

4. Documentation License

This tutorial is released under the GNU Free Documentation License Version 1.2.

The full text of this license is found in the file called “`fdl.txt`” released with tarred, gzipped file of this documentation.

5. Contact

Please direct questions or requests for more information to `<chad@neomantic.com>`. Corrections, suggestions, bug reports, and patches are welcome as well.