Using Pandoc with DocBook

Peter Lavin

2014-05-20

Abstract

Updated 2014-06-09

Table of Contents

1. What is Pandoc?

Pandoc is a utility created by John McFarlane used for converting between various document formats. Some of the "from" formats supported are:

- docbook
- textile
- rst
- html
- mediawiki
- haddock
- markdown (various types)
- latex

Some of the "to" formats supported are:

- docbook
- docx
- plain (text)

- markdown (various types)
- html (various types)
- json
- rtf
- odt

The DocBook style sheets do a very good job of transforming DocBook to HTML, PDF, EPUB and other output formats so DocBook users will principally be interested in exporting to other formats and in converting other formats to DocBook. That's what this article is concerned with.

Note: Version 1.12.4.2 of pandoc was used when writing this article.

Included Files

If you have files that are xincluded in a DocBook file, pandoc will not include them. You can solve this problem by first processing your XML file using **xmllint**. Where you have a main file that xincludes other files, use the following command to create a single file: **`xmllint --xinclude main.xml --output out.xml`**.

2. Converting Between Formats

Converting between different formats uses a number of switches. Convert a markdown file to DocBook as follows: **`pandoc -s -f markdown -t docbook in.md -o out.xml`**.

The switches used with the preceding command are:

- `-s` – Create a stand-alone document
- `-f markdown` – The input format is markdown
- `-t docbook` – The output format is DocBook
- `-o out.xml` – Output to a file named `out.xml`

This command creates a standalone DocBook 4.5 file using the default template. If you do not specify the `-s` switch, pandoc outputs a fragment. You can view the template that is used by issuing the command **`pandoc -D docbook`**. As the following output shows, DocBook is output as an `<article>`:

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.5//EN"
                  "http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd">
$endif$
<article>
  <articleinfo>
    <title>$title$</title>
$for(author)$
    <author>
```

```
      $author$
    </author>
$endfor$
$if(date)$
    <date>$date$</date>
$endif$
  </articleinfo>
$for(include-before)$
$include-before$
$endfor$
$body$
$for(include-after)$
$include-after$
$endfor$
</article>
```

If you wish, you can use an alternate template or change the default template. Documentation of template syntax is found at Pandoc User's Guide

The `-t` switch defines the destination format and the `-f` switch defines the source format. The input file, in this case `in.md`, does not require a switch. Specify an output file using the `-o` switch; if you do not use this switch, output is sent to *stdout*.

3. Converting Between DocBook Versions

Pandoc requires DocBook 4.5 for input and it also outputs DocBook 4.5. If you use DocBook 5.x you will need to convert your DocBook files before passing them to **pandoc** and you will also likely want to upgrade the output of **pandoc**. Downgrading to DocBook 4.5 is easily done thanks to the style sheets provided by Thomas Schraitle at Converting DocBook from Version 5 to Version 4. Likewise, upgrading is equally easily done using the upgrade style sheet provided along with the latest style sheets at the DocBook Project.

Install the downgrade style sheets to the directory where you store your customization XSL and you can downgrade DocBook 5 to Docbook 4.5 using xsltproc in the following way:

```
shell> xsltproc --xinclude --nonet --output out.xml path/to/db5to4-withinfo.xsl in.xml
```

The upgrade style sheet comes with all versions of the DocBook 5 XSL files and it is found in the `tools` directory. Use it in the following way:

```
shell> xsltproc --xinclude --nonet --output out.xml path/to/tools/db4-upgrade.xsl in.xml
```

If you prefer you can use the Saxon XSLT processor instead of xsltproc.

4. DocBook to HTML

If you work with DocBook regularly, you will have set up the style sheets for conversion to the various output formats so there is little likelihood that you will want to use pandoc for HTML conversions

of DocBook. However, if you don't have the style sheets readily accessible, pandoc can be a useful. Also, if you wish to use code highlighting, converting DocBook to HTML using pandoc can be a quick solution. For example, to convert a DocBook file to a standalone HTML file with code highlighting in the tango style use: **`pandoc -s --highlight-style=tango -f docbook -t html in.xml -o out.html`**. The highlighting options are: pygments (the default), kate, monochrome, espresso, zenburn, haddock, and tango. For a discussion of the highlighting styles and the languages supported see http://johnmacfarlane.net/highlighting-kate/.

Using DocBook style sheets to create HTML converts all id attributes to anchors. You should be aware that this is *not* the case when you use pandoc. Converting to HTML using pandoc suffers from the following shortcomings:

- `<figure>` tags lose their `<title>`s.
- `<example>` tags lose their `<title>`s.
- The `id` attribute is not converted to an anchor.
- The `linkend` attribute doesn't get converted to the appropriate hyperlink. Even if pandoc created a legitimate cross reference, there would be no anchor to go to since ids are not converted.

As you'll soon see, these failings also apply to transformations to other formats.

5. DocBook to Microsoft Word

Pandoc can't convert Word documents to DocBook but conversion of DocBook to `docx` is supported. The following commands convert a DocBook 5 file to 4.5 and then output a `docx` file.

```
shell> xsltproc --xinclude --nonet --output tmp.xml path/to/db5to4-withinfo.xsl in.xml
shell> pandoc -t docx -f docbook -o out.docx tmp.xml
```

Opening `out.docx` in Word reveals the same failings noted in Section 4, “DocBook to HTML”[3]; titles of figures and examples are lost as are `linkend`s.

6. Fixing `<figure>` and `<example>` Tags

Unfortunately pandoc conversion to `docx` or HTML format has the major flaws identified in Section 5, “DocBook to Microsoft Word”[4] and Section 4, “DocBook to HTML”[3] and none of the following workarounds solve these problems:

- Converting to HTML using the DocBook style sheets and then converting to docx using pandoc
- Converting to RTF rather than to docx
- Converting to ODT and then converting this format to docx

The failure to convert an `id` to an anchor looks insurmountable because this is a function of how pandoc is programmed. However, there is an XSLT workaround for the issue with figures and examples.

To recap, when converting to HTML from DocBook, pandoc successfully converts a figure tag to an `<img>` tag but it ignores the title and the same thing happens when converting to `docx`. Consider the following DocBook figure:

```
<figure id="figure">
  <blockinfo>
    <title>The Image Title</title>
  <blockinfo>
  <mediaobject>
    <imageobject>
      <imagedata fileref="images/an_image.png" format="PNG" lang="en"/>
    </imageobject>
    <textobject>
      <phrase lang="en">The Image Phrase</phrase>
    </textobject>
  </mediaobject>
</figure>
```

The figure example has an id attribute and a title nested within a `<blockinfo>` tag. You can use an XSLT transformation to adjust the XML to something that pandoc understands. The following transformation style sheet removes figure and blockinfo tags and converts the figure title to a `<bridgehead>` tag bearing the id previously associated with the figure. It also performs the same transformation on example tags.

fix_figure.xsl

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()"/>
    </xsl:copy>
  </xsl:template>
  <!-- convert figure title to bridgehead, capture id -->
  <xsl:template match="title">
    <xsl:choose>
      <xsl:when test="(name(parent::*) = 'blockinfo')
                  and (name(../..) = 'figure'
                    or name(../..) = 'example')">
        <bridgehead>
          <!-- check that there is an id first! -->
          <xsl:choose>
            <xsl:when test="../../@id">
              <xsl:attribute name="id">
                <!-- id will be grandparent -->
                <xsl:value-of select="../../@id"/>
              </xsl:attribute>
            </xsl:when>
            <xsl:otherwise><!-- do nothing --></xsl:otherwise>
          </xsl:choose>
          <xsl:apply-templates select="@*|node()"/>
        </bridgehead>
      </xsl:when>
      <xsl:when test="name(parent::*) = 'figure'
                  or name(parent::*) = 'example'">
        <bridgehead>
          <xsl:choose>
            <xsl:when test="../@id">
              <xsl:attribute name="id">
                <xsl:value-of select="../@id"/>
```

```
              </xsl:attribute>
            </xsl:when>
            <xsl:otherwise><!-- do nothing --></xsl:otherwise>
          </xsl:choose>
          <xsl:apply-templates select="@*|node()"/>
        </bridgehead>
      </xsl:when>
      <xsl:otherwise>
        <title>
          <xsl:apply-templates select="@*|node()"/>
        </title>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:template match="sectioninfo">
    <xsl:apply-templates/>
  </xsl:template>
  <!-- remove keywordset, authorblurb and all contents, also  blockinfo-->
  <xsl:template match="blockinfo">
    <xsl:apply-templates/>
  </xsl:template>
  <!-- get rid of entirely -->
  <xsl:template match="keywordset"/>
  <xsl:template match="authorblurb"/>
  <!-- remove figure and example -->
  <xsl:template match="figure
                       | example">
    <xsl:apply-templates/>
  </xsl:template>
  <xsl:output method="xml" indent="yes"
    doctype-public="-//OASIS//DTD DocBook XML V4.5//EN"
    doctype-system="http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd"/>
</xsl:stylesheet>
```

If the transformation in the section called “fix_figure.xsl”[5] is applied prior to using pandoc, figure and example titles will be preserved in the output regardless of the type of output. Converting DocBook 5 to `docx` (or HTML) then requires the following steps:

```
shell> xsltproc --xinclude --nonet --output tmp.xml path/to/db5to4-withinfo.xsl in.xml
shell> xsltproc --xinclude --nonet --output tmp.xml path/to/fix_figure.xsl tmp.xml
shell> pandoc -t docx -f docbook -o out.docx tmp.xml
```

The additional step is shown in bold. Note that the input file for this step is the output file of the first step.

Not Definitive

No definitive tests were conducted to determine exactly how pandoc handles all DocBook tags. There may well be other tags that are not accurately transformed by pandoc. However, if titles are unsupported with other tags, fix_figure.xsl can easily be adjusted to accommodate them.

There are a few issues with pandoc's transformation of DocBook files (and it would be especially nice if ids were converted to anchors) but no conversion between different markup languages will be 100% "correct" all of the time.

Thanks to John McFarlane for creating this tool.

7. Addendum

Since writing this article I have discovered the following issues with conversion of DocBook to a Word file:

- The titles of `<sidebar>` tags are not output.

  Solution: Adapt the fix_figure.xsl transformation style sheet to process `sidebar` tags in the same way as `figure` and `example` tags.

- The `<userinput>` tag is treated as a verbatim, block tag.

  Solution: Adapt the fix_figure.xsl transformation style sheet to convert `userinput` tags to `emphasis` tags.

- The content of `<remark>` tags is output.

  Solution: Adapt the fix_figure.xsl transformation style sheet to ignore `remark` tags.

The diff file of the changes is as follows:

```
       <xsl:apply-templates select="@* | node()"/>
     </xsl:copy>
   </xsl:template>
-  <!-- convert figure title to bridgehead, capture id -->
+  <!-- convert figure title to bridgehead, capture id.
+   Do this for figures, examples and sidebars. -->
   <xsl:template match="title">
     <xsl:choose>
       <xsl:when
         test="(name(parent::*) = 'blockinfo')
                   and (name(../..) = 'figure'
-                    or name(../..) = 'example')">
+                    or name(../..) = 'example'
+                    or name(../..) = 'sidebar')">
         <bridgehead>
           <!-- check that there is an id first! -->
           <xsl:choose>
@@ -27,7 +29,8 @@
       </xsl:when>
       <xsl:when
         test="name(parent::*) = 'figure'
-           or name(parent::*) = 'example'">
+           or name(parent::*) = 'example'
+           or name(parent::*) = 'sidebar'">
         <bridgehead>
           <xsl:choose>
             <xsl:when test="../@id">
@@ -50,16 +53,25 @@
   <xsl:template match="sectioninfo">
     <xsl:apply-templates/>
   </xsl:template>
-  <!-- remove keywordset, authorblurb and all contents, also  blockinfo-->
+  <!-- remove keywordset, authorblurb, remark and all contents, also  blockinfo-->
   <xsl:template match="blockinfo">
     <xsl:apply-templates/>
   </xsl:template>
   <!-- get rid of entirely -->
   <xsl:template match="keywordset"/>
```

```
   <xsl:template match="authorblurb"/>
+  <xsl:template match="remark"/>
+  <!-- convert userinput to emphasis -->
+  <xsl:template match="userinput">
+    <emphasis>
+      <xsl:apply-templates/>
+    </emphasis>
+  </xsl:template>
   <!-- remove figure and example -->
-  <xsl:template match="figure
-                       | example">
+  <xsl:template
+    match="figure
+         | example
+         | sidebar">
     <xsl:apply-templates/>
   </xsl:template>
   <xsl:output method="xml" indent="yes"
```

You can determine which DocBook tags have and which haven't been implemented by looking at Text-Pandoc-Readers-DocBook.

8. About the Author

Peter Lavin is a technical writer who has been published in a number of print and online magazines. He is the author of Object Oriented PHP, published by No Starch Press and a contributor to PHP Hacks by O'Reilly Media.