Automating Document Assembly in DocBook

Norman Walsh
Lead Engineer
Mark Logic Corporation
13 March 2010

A brief aside

http://www.w3.org/TR/xproc/

W3C Proposed Recommendation

XProc: An XML Pipeline Language

W3C Proposed Recommendation 9 March 2010

This Version:
http://www.w3.org/TR/2010/PR-xproc-20100309/
Latest Version:
http://www.w3.org/TR/xproc/
Previous versions:
http://www.w3.org/TR/2010/WD-xproc-20100105/
http://www.w3.org/TR/2009/CR-xproc-20090528/
http://www.w3.org/TR/2008/CR-xproc-20081126/
Editors:
Norman Walsh, Mark Logic Corporation <norman.walsh@marklogic.com>
Alex Milowski, Invited expert <alex@milowski.org>
Henry S. Thompson, University of Edinburgh <ht@inf.ed.ac.uk>

This document is also available in these non-normative formats: XML, Revision markup

Agenda

- Background
- Topic-based documentation with DocBook
- What is document assembly?
- DocBook assembly documents
- Interpreting assemblies

Background

- "Traditional" (or narrative) authoring

  A narrative is a story that is created in a constructive format (as a work of writing, …) that describes a sequence of fictional or non-fictional events. It derives from the Latin verb narrare, which means "to recount" and is related to the adjective gnarus, meaning "knowing" or "skilled"… [B]ut can also be used to refer to the sequence of events described in a narrative.

  —*Wikipedia*

- Topic-based Authoring

  Topic-based authoring is a modular content creation approach (popular in the technical publications and documentation arenas) that supports XML content reuse, content management, and makes the dynamic assembly of personalized information possible.

  —*Wikipedia*

A few myths

- DocBook is only be used for books
- DocBook content can't be reused
- DocBook can't support a topic-based writing methodology
- Topic-based authoring produces better documentation

Topic-based markup in DocBook

- <refentry>
- <article>
- <section>
- Coming soon: <topic>

What about assembly?

Consider:

- Four products on three platforms
- with 250 topics
- of which, about 200 are shared between any two products

Assembly goals

- Combine a collection of resources
- Using an (arbitrary) order and nesting suitable for the kind of output needed
- To produce a *valid DocBook* document
  - Or a valid document in some appropriate customization

In particular: transforming the assembled document into the actual output format (PDF, eBook, web pages, help system) is the responsibility of another process.

Assembly contents

- Identify the resources
- Combine them into structures (i.e. products)
- Describe additional relationships
- Perhaps identify transformations

N.B. This presentation is based on internal drafts. They don't represent any sort of consensus position from the DocBok Technical Committee. Subject to change without notice. YMMV. etc.

Identify resources


```
<resources xml:base="/some/base/">
  <resource xml:id="simple" fileref="simple.xml"/>

  <resource xml:id="topicA" fileref="topicA.xml"
            xpointer="element(A)"/>

  <resource xml:id="topicB" fileref="topicA.xml"
            xpointer="element(B)"/>

  <resource xml:id=”toc”>
    <toc/>
    <toc role=”procedures”/>
  </resource>
</resources>
```

The resources


```
simple => <article>...

topicA => <chapter>...<section xml:id="A">...

topicB => <book>...<topic xml:id="B">....

toc    => <toc/>, <toc role="procedures"/>
```

Combine into structures


```
<structure xml:id="user-guide">
  <output renderas="book"/>
  <info>
    <title>Widget User Guide</title>
  </info>
  <module resourceref="toc"/>
  <module>
    <output renderas="chapter"/>
    <info>
      <title>Chapter Title</title>
    </info>
    <module resourceref="topicA"/>
    <module resourceref="topicB"/>
  </module>
</structure>
```

Combined result


```
<book xml:id="user-guide">
  <info>
    <title>Widget User Guide</title>
  </info>
  <toc/>
  <toc role="procedures"/>
  <chapter>
    <info>
      <title>Chapter Title</title>
    </info>
    <section xml:id="A">...</section>
    <topic xml:id="B">...</topic>
  </module>
</structure>
```

Combine with more care


```
<structure xml:id="user-guide">
  <output renderas="book"/>
  <info>
    <title>Widget User Guide</title>
  </info>
  <module resourceref="toc"/>
  <module>
    <output renderas="chapter"/>
    <info>
      <title>Chapter Title</title>
    </info>
    <module resourceref="topicA">
      <output renderas="section"/>
    </module>
    <module resourceref="topicB">
      <output renderas="section"/>
    </module>
```

Additional relationships


```
<relationship type="seealso">
  <instance resourceref="tut1"/>
  <instance resourceref="tut2"/>
  <instance resourceref="task1"/>
</relationship>

<relationship type="path">
  <info>
    <title>New User Introduction</title>
  </info>
  <instance resourceref="over1"/>
  <instance resourceref="over2"/>
  <instance resourceref="task3"/>
  <instance resourceref="cleanup"/>
</relationship>
```

Identify Transformations


```
<transformations>
  <transform name="dita2docbook" type="text/xsl"
             fileref="dita2db.xsl"/>
  <transform name="tutorial" type="text/xsl"
             fileref="db2tutorial.xsl"/>
  <transform name="art2pi" type="text/xsl"
             fileref="art2pi.xsl"/>
  <transform name="office"
             type="application/xproc+xml"
             fileref="office2db.xpl"/>
  <transform name="office" type="text/xsl"
             fileref="extractoffice.xsl"/>
</transformations>
```

Transformations


```
<resource xml:id="overview"
          fileref="dita/over.xml"
          transform="dita2docbook"/>

…

<module resourceref="overview">
  <transform name="art2pi"/>
</module>
```

Interpreting assemblies

- An assembly declares the structure of a document
- Interpreting an assembly requires processing a set of resources (documents or document fragments)
- A structure is the concatenation of a set of modules, each of which may have been transformed
- A module is the concatenation of a set of resources or other modules, each of which may have been transformed
- In other words, an assembly describes a pipeline of operations that must be performed to build the assembled document

Interpreting assemblies (2)

- Use XSLT to transform an assembly structure into an XProc pipeline.
- Run the XProc pipeline to build the assembly.