DocBook: Case Studies

Norman Walsh
Sun Microsystems, Inc.

Background

What is DocBook?
A DocBook Document
DocBook History
DocBook's Purpose

What is DocBook?

- **DocBook is an XML vocabulary for writing documentation. It is particularly well-suited to books and papers about computer hardware and software, though it is by no means limited to them.**
- **It has been subset down to something that resembles HTML.**
- **It has been extended to do things as different as websites and presentations, like this one.**

A DocBook Document

```
<book xmlns="http://docbook.org/ns/docbook">
<info>
<title>A Book Title</title>
<author>
  <personname>
    <firstname>John</firstname>
    <surname>Doe</surname>
  </personname>
</author>
</info>
<chapter>
<title>The First Chapter</title>
<para>Some <emphasis>text</emphasis>.</para>
</chapter>
</book>
```

DocBook History

- **DocBook has been actively maintained for more than a decade.**
- **It has always been maintained by a committee of some sort. It is now being developed by an OASIS Technical Committee.**
- **DocBook was an SGML DTD for many years. It is now principally an XML DTD. It will officially be a RELAX NG Grammar *real soon now*.**

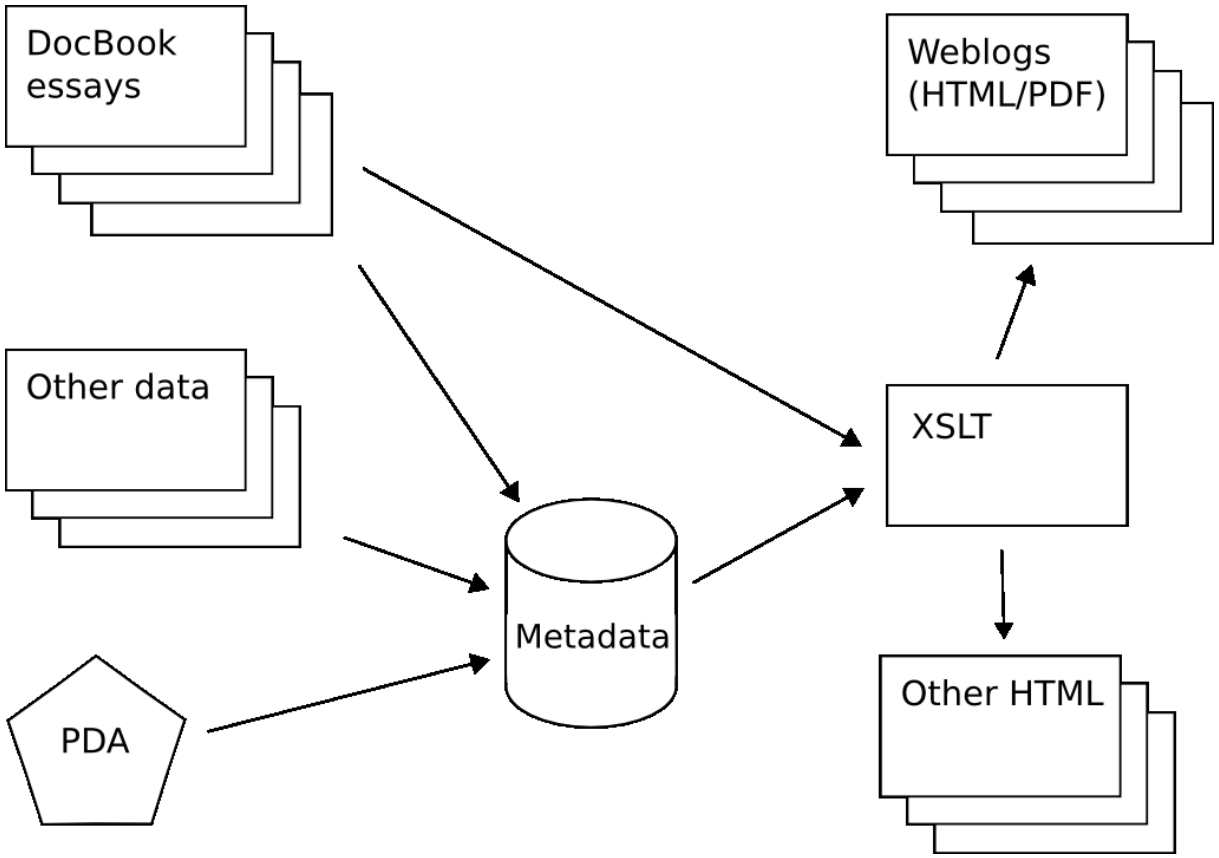
DocBook's Purpose

- **DocBook documents are mostly hand authored. Unlike SOAP envelopes, purchase orders, and XML/RPC invocations, humans write DocBook.**
- **It's mostly read by humans. DocBook documents, aren't usually consumed by unmarshalling processes building object graphs.**
- **DocBook contains a lot of mixed content. Very few elements have "simple content," dates, numbers, etc.**

Case Study: norman.walsh.name

http://norman.walsh.name/
It's all about the metadata

It's all about the metadata



DocBook Customizations

Overview
DocBook 5.0 and XSLT 2.0
A new hierarchy root
HTML markup
HTML markup (2)
HTML markup (3)
Geographic metadata
Geographic metadata (2)
…

Overview

We'll look at schema and stylesheet customizations that support:

- **A new hierarchy root**
- **HTML markup**
- **Geographic metadata**
- **“Database generated” trip itineraries**
- **Inline markup for metadata**
- **Inline markup for personal names**

DocBook 5.0 and XSLT 2.0

The examples in the slides that follow are taken from DocBook V5.0 and from the DocBook XSLT 2.0 stylesheets.

If you're interested in exploring these customizations, you can find everything online:

- **`http://norman.walsh.name/schema/essay.rnc`**
- **`http://norman.walsh.name/style/essay2.html.xsl`**
- **`http://sourceforge.net/projects/docbook` for the base schemas and stylesheets.**

Share and enjoy!

A new hierarchy root

This one's easy. In the schema, place the new root element in the start pattern:

```
include "http://docbook.org/xml/5.0/rng/docbook.rnc" {
   start = db.essay
}
```

In the stylesheet, place it (exclusively) in the set of root elements.

```
<xsl:param name="root.elements">
  <db:essay/>
</xsl:param>
```

And, naturally, a template for `db:essay`.

HTML markup

The principle target for these essays is HTML. Every now and then, the easiest way to support some HTML feature is simply to insert some HTML:

```
<essay ...>
<info>
...
<html:style xmlns:html="http://www.w3.org/1999/xhtml" type="text/css"
div.classvalue { margin-left: 0.5in; }
</html:style>
</info>
...
</essay>
```

HTML markup (2)

We can allow that by adding patterns to the schema:

```
db.info.elements |=
   html.style

html.style = element html:style {
   attribute type { text },
   attribute media { text }?,
   attribute title { text }?,
   text
}
```

HTML markup (3)

And templates to the stylesheet:

```
<xsl:template match="html:*" priority="200">
  <xsl:element name="{local-name(.)}">
    <xsl:copy-of select="@*"/>
    <xsl:apply-templates/>
  </xsl:element>
</xsl:template>
```

This relies on an `<xsl:apply-templates .../>` elsewhere that processes the HTML elements in the `db:info` (and on the fact that the default namespace is the XHTML namespace).

Geographic metadata

In a metadata-rich vocabulary, one solution is to use the standard geographic metadata namespace:

```
<essay xmlns="http://docbook.org/ns/docbook"
       xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
<info>
<title>norman.walsh.name</title>
...
<geo:lat>42.3382</geo:lat>
<geo:long>-72.45</geo:long>
</info>
...
</essay>
```

Geographic metadata (2)

For essays about a particular place, the Dublin Core coverage element makes more sense:

```
<essay xmlns="http://docbook.org/ns/docbook"
       xmlns:dc='http://purl.org/dc/elements/1.1/'>
<info>
<title>XML Prague, 16-17 June</title>
...
<dc:coverage
  rdf:resource="http://norman.walsh.name/knows/where#cz-prague"/>
</info>
...
</essay>
```

Geographic metadata (3)

Like the HTML example, we do this with new patterns in the grammar:

```
db.info.elements |=
   geoElement
 | dcElement

geoElement = element geo:* { rdfContentModel }
dcElement = element dc:* { rdfContentModel }

rdfContentModel = rdfResource | rdfLiteral

rdfResource = attribute rdf:resource { text }

rdfLiteral  =
   attribute rdf:datatype { text }?,
   text
```

Geographic metadata

And rules to the stylesheet:

```
<xsl:template match="db:essay">
   ...
   <xsl:choose>
     <xsl:when test="db:info/geo:lat">
       <meta name="ICBM">
         <xsl:attribute name="content">
           <xsl:value-of select="db:info/geo:lat"/>
           <xsl:text>,</xsl:text>
           <xsl:value-of select="db:info/geo:long"/>
         </xsl:attribute>
       </meta>
     </xsl:when>
     ...
```

“Database generated” trip itineraries

It starts with my PDA:

Trip data

- PDA event (flights, speaking engagements, etc) data becomes XML then RDF
- PDA contact (hotel, venue, etc.) data becomes XML then RDF
- Augmented with geographic metadata about countries, cities, and airports.
- Stirred and shaken into a great big bag

Trip itineraries

Itineraries are extracted from this data and incorporated into an essay:

```
<essay ...>
<info>
<title>XML Prague, 16-17 June</title> ...
</info>
<para xml:id='p1'>I'm presenting at the XML Prague conference.</para>
<trip xmlns="http://nwalsh.com/rdf/itinerary#"
      startDate="2007-06-14T11:00:00-04:00"
      endDate="2007-06-19T17:40:00-04:00"
      trip="06-14-xmlprague">
   <itinerary>
      <leg class="flight">
         <startDate>2007-06-14T06:30:00-04:00</startDate>
         <endDate>2007-06-14T07:45:00-04:00</endDate>
         <description>BDL-JFK/Delta 6189</description> ...
      </leg> ...
</trip>
```

Integrating trip itineraries

Add the itinerary markup to the informal blocks, so that it can appear in an essay:

```
db.informal.blocks |= tripItinerary

tripItinerary = element it:trip {
   attribute startDate { xsd:date | xsd:dateTime },
   attribute endDate { xsd:date | xsd:dateTime },
   attribute trip { text }?,
   attribute nomap { text }?,
   anyElement*
}
```

With appropriate stylesheet modifications too.

Inline markup for metadata

- **Metadata is only useful if it exists.**
- **It is more likely to exist if the barrier to entry is as low as possible.**
- **One useful set of metadata identifies what an essay is about.**
- **I use Wikipedia URIs to identify topics for cross-referencing.**

```
<para xml:id='p1'>Last week, ...
the JAXP team released
<link xlink:href="https://jaxp.dev.java.net/1.4/">version
1.4.2</link> of <wikipedia>JAXP</wikipedia>.</para>
```

Metadata markup

```
db.markup.inlines |= db.wikipedia

db.wikipedia = element db:wikipedia {
   db.common.attributes,
   attribute page { text }?,
   text
}
```

Inline markup for personal names

- **This is for another sort of index.**
- **It could be used for more interesting linking.**

Personal names

The standard way to markup a personal name is verbose:

```
<personname><firstname>Norman</firstname>
<surname>Walsh</surname></personname>
```

Borrowing form FOAF, I've adopted two more abbreviated mechanisms:

```
<foaf:name>Norman Walsh</foaf:name>
```

and

```
<foaf:nick>ndw</foaf:name>
```

Personal name markup

```
db.bibliography.inlines |= foafName | foafNick

foafName = element foaf:name {
   text
}

foafNick = element foaf:nick {
   text
}
```

Styling personal names

```
<xsl:template match="foaf:name">
  <xsl:variable name="allFriends" select="key('foaf:names', ., $allro

  <xsl:choose>
    <xsl:when test="count($allFriends) != 1">
      ...
    </xsl:when>
    <xsl:otherwise>
      <xsl:choose>
 <xsl:when test="$allFriends/foaf:weblog"> ... </xsl:when>
 <xsl:otherwise>
   <span class="foafName">
     <xsl:value-of select="$allFriends/foaf:firstName"/>
   </span>
 </xsl:otherwise>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

Q&A

This set of slides will be online soon; other useful pointers:

- **`http://norman.walsh.name/Makefile`**
- **`http://norman.walsh.name/schema/essay.rnc`**
- **`http://norman.walsh.name/style/essay2.html.xsl`**
- **`http://sourceforge.net/projects/docbook` for the base schemas and stylesheets.**

Share and enjoy!