XSL Formatting Objects

R. Alexander Milowski

milowski@sims.berkeley.edu

What is XSL?

- XSL is a formatting language.
- It is a vocabulary for describing the formatting (i.e. layout) of a document via "Formatting Objects".
- You use XSLT to transform and XML document into XSL Formatting Objects to get rendered output.

Formatting Objects?

- A formatting object is a specification "object" that generates a sequence of areas.
- An area is a typically a rectangular object placed on a page.
- For example, a paragraph generates a rectangular area in which the sentences of the paragraph are set.
- Also, a character generates a rectangular glyph which represents the character.

Enough Already! Where's the...

- Just think of XSL as a vocabulary you can use to create nice PDFs.
- In theory, you can use XSL in a browser...
- The nice thing about XSL is that you can:
  - Use XSLT to re-arrange the output.
  - Specify page sequences for different page layouts.
  - Control the page layout (body, headers, footers, etc.).
  - There is an expression language that lets you calculate expressions against the rendered tree.

XSL by Example

- Let's format these slides into PDF.
- Each slide will become a page.
- The content is basically XHTML: p, pre, ol, ul, li, etc.

Getting Started

- Setup the XSLT transformation:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
                xmlns:fo="http://www.w3.org/1999/XSL/Format"
                xmlns:s="urn:publicid:IDN+mathdoc.org:schema:slides:2004:1.0:us"
                xmlns:md="urn:publicid:IDN+mathdoc.org:schema:common:2004:1.0:us"
                xmlns:h="http://www.w3.org/1999/xhtml"
                >
<xsl:output method="xml"/>

</xsl:stylesheet>
```

- The namespace for the output FO vocabulary is http://www.w3.org/1999/XSL/Format
- We'll have to run the result through an XSL aware formatter (e.g. FOP).

Page Masters and Page Sequences

- A sequence of pages follows a page master.
- A page master defines the basic layout of a page and the regions on the page.
- There is a "simple page master" built-in with the following regions:
  - region-before - The header of the page.
  - region-after - The footer of the page.
  - region-start - The "left margin" region.
  - region-end - The "right margin" region.
  - region-body - The "right margin" region.
- We'll "flow" content into these regions.

Defining the Page Sequence

- Let's define the page in the template for [s:]slides:

```
<xsl:template match="s:slides">
<fo:root font-size="18pt">
   <fo:layout-master-set>
      <fo:simple-page-master master-name="slide" page-height="8.5in" page-width="11in">
         <fo:region-body margin-top="0.5in" margin-bottom="0.5in"
                         margin-left="0.75in" margin-right="0.75in"/>
      </fo:simple-page-master>
   </fo:layout-master-set>
   <fo:page-sequence master-reference="slide">
      <fo:title><xsl:value-of select="s:title/text()"/></fo:title>
      <fo:flow flow-name="xsl-region-body">
         <xsl:apply-templates/>
      </fo:flow>
   </fo:page-sequence>
</fo:root>
</xsl:template>
```

- The contents of the document will flow into the region-body of the pages.

The Title Slide

- The title slide will be output for the [s:]authors element:

```
<xsl:template match="s:slides/s:authors">
<fo:block break-after="page" text-align="center" font-weight="bold">
<fo:block font-size="150%"><xsl:value-of select="preceding-sibling::s:title"/></fo:block>
<xsl:apply-templates/>
</fo:block>
</xsl:template>
```

- The [fo:]block element acts a lot like XHTML's p element.

Each Slide

- We'll contain each slide in a [fo:]block:

```
<xsl:template match="s:slides/s:slide">
<fo:block>
  <xsl:if test="following-sibling::s:slide">
    <xsl:attribute name="break-after">page</xsl:attribute>
  </xsl:if>
  <xsl:apply-templates/>
  </fo:block>
</xsl:template>
```

- [fo:]block can contain other [fo:]block much like XHTML's div can contain p elements.
- The break-after attribute tells the page to break after the slide.
- We only want a page break between slides.

Blocks and Inlines

- Just like CSS, XSL has block and inline formatting objects.
- Blocks can contain other blocks.
- Inlines "stack" in the writing direction (e.g. possibly vertical-down in Japanese, left-to-right in English, right-to-left in Hebrew, etc.).
- The difference is that XSL is fully internationalized and allows mixing of languages and writing directions.

Properties

- Properties control how the areas are generated and placed.
- For example, font properties control the font used by inline areas.
- Properties are specifed via attributes.
- Some property values can be inherited by children and descendants.
- Many of the CSS properties are available "verbatim".
- Properties values can also be expressions: body-start()+3pt

Cleaning up the Title Slide

- We'd like things centered (text-align property).
- We need some spacing between things.
- The font size, bold, etc. should be changed.
- One way to do this:

```
<xsl:template match="s:slides/s:authors">
<fo:block break-after="page" text-align="center" font-weight="bold">
   <fo:block font-size="175%" space-before="60pt" space-after="30pt">
      <xsl:value-of select="preceding-sibling::s:title"/>
   </fo:block>
   <xsl:apply-templates/>
</fo:block>
</xsl:template>

<xsl:template match="md:author/md:name">
<fo:block font-size="140%">
<xsl:attribute name="space-before">
<xsl:choose>
<xsl:when test="preceding-sibling::md:author">
15pt
</xsl:when>
<xsl:otherwise>30pt</xsl:otherwise>
</xsl:choose>
</xsl:attribute>
<xsl:apply-templates/>
</fo:block>
</xsl:template>

<xsl:template match="md:author/md:email">
<fo:block space-before="8pt"><xsl:apply-templates/></fo:block>
</xsl:template>
```

The Slide Titles

- The title should "stand out" and be centered.
- One way to do this:

```
<xsl:template match="s:slide/s:title">
<fo:block
   font-weight="bold" text-align="center" font-size="150%"
   space-after="30pt"
>
<xsl:apply-templates/>
</fo:block>
</xsl:template>
```

XHTML Paragraphs

- Paragraphs should be offset from the rest of the text.
- The left-indent is relative to the current placement, so we'll use an expression for 'start-indent'.
- One way to do this:

```
<xsl:template match="h:p">
<fo:block space-before="3pt" space-after="3pt"
          start-indent="body-start()+3pt">
  <xsl:apply-templates/>
</fo:block>
</xsl:template>
```

XHTML Lists

- Numbers and bullets must be generated by XSLT.
- The [fo:]list-block, [fo:]list-item, [fo:]list-item-label, and [fo:]list-item-body will generate list areas.
- One way to do this:

```
<xsl:template match="h:ol|h:ul">
<fo:list-block margin-top="5pt">
<xsl:apply-templates/>
</fo:list-block>
</xsl:template>

<xsl:template match="h:ul/h:li">
<fo:list-item margin-top="10pt">
   <fo:list-item-label end-indent="label-end()">
      <fo:block><fo:inline font-family="Symbol">&#x2022;</fo:inline></fo:block>
   </fo:list-item-label>
   <fo:list-item-body start-indent="body-start()">
      <xsl:apply-templates/>
   </fo:list-item-body>
</fo:list-item>
</xsl:template>

<xsl:template match="h:ol/h:li">
<fo:list-item margin-top="10pt">
   <fo:list-item-label end-indent="label-end()">
      <fo:block><xsl:number count="h:li" level="single" format="1."/></fo:block>
   </fo:list-item-label>
   <fo:list-item-body start-indent="body-start()">
      <xsl:apply-templates/>
   </fo:list-item-body>
</fo:list-item>
</xsl:template>
```

Preformatted Text

- We want the XHTML pre elements to have a border, background, and for the formatting to remain.
- Certain properties of [fo:]block enable this.
- One way to do this:

```
<xsl:template match="h:pre">
<fo:block
   font-family="monospace"
   font-size="12pt"
   white-space-treatment="preserve"
   white-space-collapse="false"
   linefeed-treatment="preserve"
   wrap-option="no-wrap"
   background-color="rgb(220,220,220)"
   border-color="gray"
   border-width="1pt"
   border-style="solid"
   padding="5pt"
   space-before="5pt"
   space-after="5pt"
>
<xsl:apply-templates/>
</fo:block>
</xsl:template>
```

Running FOP

- There is a program from apache called FOP that will generate PDF, etc. from XSL.
- You can run this from the command-line as:

```
fop.sh -xml lecture.xml -xsl slides2fo.xsl -pdf lecture.pdf
```

  or to generate from XSL directly:

```
fop.sh lecture.fo lecture.pdf
```

- The result is (fo) (pdf) as produced by slides2fo.xsl .