6 XSL: Extensible Stylesheet Language

- An advanced style language for XML documents:
  1. Language for transforming XML documents: XSLT
  2. XML vocabulary (= markup language) for specifying formatting:
     XSL version 1.0, W3C Rec. (15 October, 2001)
     - written for implementers of XSL processors
- 6.1 Introduction and Overview
- 6.2 XSL Formatting by Example

What is it?

- An XSL style sheet specifies the presentation of a class of XML documents
  - by describing an XSLT transformation of the XML document into an XML document that uses the formatting vocabulary
    - XSL FO: a markup language to describe formatting
  - XSL builds on CSS2 and DSSSL
    - DSSSL an ISO-standardised, but mainly unimplemented SGML style language

Example of XSL syntax

- Formatting paragraph elements (p):
  - NB: An incomplete style sheet!

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="p">
    <fo:block>
      <fo:script>
        This is a paragraph.
      </fo:script>
    </fo:block>
  </xsl:template>
</xsl:stylesheet>
```

6.1 Overview of XSL Formatting

- A style sheet processor accepts an XML document and an XSL style sheet, and produces a formatted presentation
- Two steps:
  1. (XSLT) transformation:
     XML source tree -> result tree
  2. (XSL FO) formatting
     - interpreting the result tree to produce formatted presentation

Transformation & Formatting

- Tree transformation adds information needed to format the result tree
- Formatting semantics expressed using a formatting vocabulary, of:
  - formatting objects (FOs), nodes of the result tree
    - for typographic abstractions like page-sequence, block, in-line text, page reference, ...
    - XSL 1.0 defines 56 formatting object classes
  - formatting properties control the presentation of formatting objects (indents, spacing, fonts, ...) - XSL 1.0 defines 248 formatting properties, many common with CSS2

Areas and Area Tree

- Formatting generates an area tree consisting of nested rectangular areas
  - inline areas (e.g. glyph areas) within line areas
  - lines within block areas
  - blocks within regions of a page
- Rendering causes the area tree to appear on a medium
  - areas printed on a sequence of sheets
    - (or displayed as a single scroll in a browser)
Generating the Area Tree (1/3)

- Formatting a gradual and complex process
- Conceptual process of XSL formatting:
  - (XSL FO) Element and attribute tree
    - target of transformation, source of formatting
    - consists of element, attribute, and text nodes
    - transformed into a …
  - Formatting object tree (→XSL FO Elem&attr tree)
    - consists of formatting objects with properties
    - more detailed: each character its own object

Generating the Area Tree (2/3)

Benefits of XSL

- An extensive model and vocabulary for expressing XML style sheets
- Powerful source selection and manipulation (with XPath/XSLT)
- Pagination and layout model extend existing ones
  - area model a superset of the CSS2 box model
  - » e.g., different writing directions; footnotes, page number refs.
- Support of non-western writing directions
  - distances specified in terms of before, after, start and end, relative to "writing-mode"

 Generating the Area Tree (3/3)

Properties of the formatting object tree refined into traits (muotoilupiirre, piirre)
- e.g., by propagating inherited properties, and computing absolute values for relative properties
  - e.g., properties
    - font-size="12pt", start-indent="2em"
    - become traits
      - font-size="12pt", start-indent="24pt"
  - traits control generation of areas out of formatting objects
  - some traits only available as a result of formatting, e.g., page numbers

XSL Area Model

- Formatting objects generate areas
  - each 0 or more
    - page breaks → additional block areas
    - line breaks → additional line areas
  - Each area tree node (except root) is associated to a rectangular portion of the output medium
  - An area has a content-rectangle
    - portion for child areas
    - optionally surrounded by a border and padding

Content, Padding and Border

- For compatibility also CSS-like margins
  - margin-top, -right, -bottom and -left

Two area types

- block-areas
  - generated in block-progression-direction (normally top-to-bottom)
  - paragraphs and titles normally rendered using \fo:block, which creates block areas
  - line-area a special case: no borders or padding
- inline-areas
  - generated in inline-progression-direction (normally left-to-right)
  - characters rendered using \fo:character, which generates glyph-area inline-areas
  - no child areas, a single glyph image as content

Formatting objects and properties

- XSL 1.0 defines 66 formatting objects …
  - page-sequence, simple-page-master, block, inline, list-block, list-item, list-item-label, list-item-body, external-graphic, basic-link, float, footnote, table, table-row, table-column, …
- 248 properties
  - master-reference, background-color, font-family, font-size, space-before, start-indent, end-indent, text-align, text-indent, …
  - many common with CSS2
Some central formatting objects 1/3

- **fo:root**
  - top node of the formatting object tree
  - a wrapper for all the rest
- **fo:simple-page-master**
  - a template for creating pages
  - specifies the geometry of pages
  - region-body (for page content)
  - region-before (for header),
  - region-after (for footer),
  - region-start and region-end (for left and right sidebar)

Page regions

- A simple page can contain 1-5 regions, specified by child elements of the simple-page-master

Top-level formatting objects

- Slightly simplified:
  - **fo:root**
  - **fo:layout-master-set**
  - **fo:page-sequence+**
  - **fo:region-body**
  - **fo:region-before**
  - **fo:region-after**
  - **fo:region-start**
  - **fo:region-end**

Content objects for pages

- Slightly simplified:
  - **fo:page-sequence+**
  - **fo:static-content+**
  - **fo:flow**
  - Block-level object+
  - Block-level object+

Some block-level objects

- **fo:block**
  - commonly used for paragraphs, titles, ...
  - may contain text, other blocks, or
    - **fo:inline**
      - (to change properties, e.g., font-style of inline text)
  - **fo:table** for formatting tabular material
  - **fo:list-block** to format lists of
    - **fo:list-items** of
      - **fo:list-item-label** and
      - **fo:list-item-body**

Implementations?

- W3C XSL Rec rather recent (10/2001)
- What is the state of implementations?
- Some promising/interesting ones:
  - XEP by RenderX (XSL-FO to PS/PDF formatter),
    - XSL Formatter by Antenna House
  - $0 ... $5000 (evaluation ... server versions; April 2004)
  - Adobe Document Server
  - Passive TeX
    - set of TeX macros to process XSL-FO by Sebastian Rahtz
  - Apache FOP
6.2 XSL-FO by Example

- From J. David Eisenberg: Using XSL Formatting Objects. XML com, January 17, 2001. (acknowledging the loan of some graphics)
- XSL FO instance for a handbook of Spanish
  - NB: XSL FO is not designed to be hand-authored
  - Consider this as a machine-generated result (of an XSLT transformation)
- Overall structure of fo:root: specification of
  - page masters, followed by
  - the content of the pages

Example: Page dimensions and margins

```
<fo:layout-master-set>
  <fo:simple-page-master master-name="cover"
    page-height="12cm"
    page-width="12cm"
    margin-top="0.5cm"
    margin-bottom="0.5cm"
    margin-left="1cm"
    margin-right="0.5cm"/>
</fo:simple-page-master>
</fo:layout-master-set>
```

Let’s refine the page masters with regions

Example: Region dimensions

```
<fo:simple-page-master master-name="cover"
  page-height="12cm"
  page-width="12cm"
  ...>
  <fo:region-body margin-top="3cm"/>
</fo:simple-page-master>
```

Layout of Page Regions

- A simple page can contain 1-5 regions, specified by child elements of the simple-page-master

<table>
<thead>
<tr>
<th></th>
<th>Left-hand Content Pages</th>
<th>Right-hand Content Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example: Page regions

- A simple page can contain 1-5 regions, specified by child elements of the simple-page-master

<table>
<thead>
<tr>
<th></th>
<th>Left-hand Content Pages</th>
<th>Right-hand Content Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cover</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page 5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: Page Sequences

- Next: masters for sequences of content pages, using the defined simple-page-master:
  - repeatedly alternate masters for left and right pages:

```xml
<fo:page-sequence-master master-name="contents">
  <fo:repeatable-page-master-alternatives>
    <fo:conditional-page-master-reference master-reference="leftPage" odd-or-even="even"/>
    <fo:conditional-page-master-reference master-reference="rightPage" odd-or-even="odd"/>
  </fo:repeatable-page-master-alternatives>
</fo:page-sequence-master>
```

Example: Cover Page Formatted

- Formatting the first page-sequence gives ...

```xml
<fo:block font-family="Helvetica" font-size="18pt" text-align="end">
  Spanish Review Handbook
</fo:block>
```

Example: Content Pages

- Finally, a page-sequence for content pages
  - with static-content for the header and footer,
  - and a flow for page bodies:

```xml
<fo:page-sequence master-reference="contents">
  <fo:block font-family="Helvetica" font-size="18pt" text-align="end">
    Spanish Review Handbook
  </fo:block>
</fo:page-sequence>
```

Example: Content Pages Continue

- Content for page footers:

```xml
<fo:static-content flow-name="xsl-region-after">
  Copyright © 2001 J. David Eisenberg
</fo:static-content>
```

Example: Content Pages Formatted

- Assign a flow of blocks to region-body:

```xml
<fo:block font-family="Helvetica" font-size="14pt">
  Watch this space!
</fo:block>
```

- Formatting and rendering this gives ...

```
Copyright © 2001 J. David Eisenberg
```

Example: Content Pages Continue

- Formatting and rendering this gives ...

```
Spanish Review Handbook
```

Example: Content Pages Formatted

- Other attributes of conditional-page-master-reference to select the page master to be used:

  - page-position="first"
  - or "last", or "next" (neither first or last), or "any"
  - blank-or-not-blank="blank"/"not-blank"

- Next: Specifying the sequences of content pages
  - by naming masters to be used,
  - and connecting content flows to regions
Today, XSL is most useful if you need to produce a W3C Recommendation page allows fine arbitrary transformations of input documents currently used mainly for producing PDF browser support being expected …

Examples of mapping content elements

Formatting in-line emphasis:
  \<xsl:template match="emph"\>
  <fo:inline font-style="italic">\</fo:inline>\</xsl:template>\

More examples in the exercises

Summary

- XSL is a powerful (and complex) style language for XML documents
  - allows arbitrary transformations of input documents
  - allows fine-tuned specification of formatted representation
- It is a standard!
  - well, almost: a W3C Recommendation
  - emerging implementations seem promising
  - currently used mainly for producing PDF
  - browser support being expected …

Expert Views on XSL

- “What is XSL-FO and When Should I Use It” in Seybold Report, 2(17) (Dec. 02) by S. Deach, an XSL 1.0 co-author and computer scientist at Adobe:
  - “XSL-FO is now in the ‘early-adopter’ phase”
  - “It is expected that a wide variety of authoring tools become available […] I expect a significant adoption […] over a three-to-five year time frame”
  - “XSL-FO is best […] generating content-driven documents in response to individual customer requests”
  - “Today, XSL is most useful if you need to produce customer-tailored, paginated documents on a server.”