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

Example of XSL syntax

- Formatting paragraph elements (p):
  
  ```xml
  <xsl:stylesheet
      version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:fo="http://www.w3.org/1999/XSL/Format">

  <!-- settings for the first line: -->
  <fo:block font-variant="small-caps"/>

  <fo:block>
      <xsl:apply-templates/>
  </fo:block>

  </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

Basis of 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, ...) within the result tree
  - XSL 1.0 defines 248 formatting properties; many common with CSS2

Areas and Area Tree

- Formatting-object tree interpreted to produce the representation
- Each FO specifies a part of pagination, layout and styling applied to its content
- Properties control the formatting of a FO
  - some directly, e.g., color
  - some through constraints, e.g., space-before, minimum
  - the final rendering is not uniquely defined by XSL

- 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)

- Properties of the formatting object tree refined into traits (mutoloupirre, 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

Generating the Area Tree (3/3)

- Benefits of XSL
  - Rich model and vocabulary for XML stylesheets
  - Powerful selection and manipulation (→ XSLT)
  - Pagination and layout extend existing ones
    - area model a superset of the CSS2 box model
  - Support of non-western writing directions
    - > distances expressed in terms of before/after (for block-progression-direction), and start/end (for inline-progression-direction)

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) 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 for: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 for:character, which generates glyph-area inline-areas
  - no child areas, a single glyph image as content

Formatting objects and properties

- XSL 1.0 defines 56 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, ...
  - and 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**
  - model of 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:simple-page-master | fo:page-sequence-master)+
  - **fo:flow**
    - child of a page-sequence
    - Attribute flow-name connects to a region with a matching region-name
      - the contents is distributed to that region of pages
    - NB: There are no 'page'-formatting objects
      - pages created by the formatter

Content objects for pages

- Slightly simplified:
  - **fo:page-sequence**
    - (repeated on every page)
  - **fo:flow**
    - (distributed to pages)

Some central formatting objects 2/3

- **fo:page-sequence**
  - specifies the creation of page sequences
  - possibly different page-sequence (and page-sequence-master) for, say, each chapter
- **fo:flow**
  - child of a page-sequence
  - Attribute flow-name connects to a region with a matching region-name
    - the contents is distributed to that region of pages
- **NB**: There are no 'page'-formatting objects
  - pages created by the formatter

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**
    - **fo:list-item-label** and
    - **fo:list-item-body**

"Hello world" result tree as an XSL document

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="page">
      <fo:region-body/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:flow flow-name="xsl-region-body">
    Hello World!
  </fo:flow>
</fo:root>
```

Implementations?

- W3C XSL Rec rather recent (10/2001)
- What is the state of implementations?
- Some promising/interesting ones (2004):
  - 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
Apache FOP

- FOP (Formatting Objects Processor) by J. Tauber
  - "fop: a man who pays too much attention to his appearance"
  - donated to XML Apache project (http://xml.apache.org/fop/)
  - open-source freeware
  - Java-based XML/XSL-FO to PDF (or MIF/PCL/TXT/...)
  - processor
- Not complete, but implements a useful subset of XSL 1.0

SDPL 2005 Notes 6: XSL Formatting 25

FOP 0.20.5 XSL-FO-Compliance


<table>
<thead>
<tr>
<th>Implemented</th>
<th>fully</th>
<th>partially</th>
<th>no</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>formatting</td>
<td>36 (64%)</td>
<td>17 (30%)</td>
<td>3 (5%)</td>
<td>56</td>
</tr>
<tr>
<td>objects</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>formatting</td>
<td>95 (38%)</td>
<td>110 (44%)</td>
<td>43 (17%)</td>
<td>248</td>
</tr>
<tr>
<td>properties</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>non-aural</td>
<td>95 (41%)</td>
<td>110 (48%)</td>
<td>25 (11%)</td>
<td>230</td>
</tr>
</tbody>
</table>

SDPL 2005 Notes 6: XSL Formatting 26

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

SDPL 2005 Notes 6: XSL Formatting 27

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>
```

SDPL 2005 Notes 6: XSL Formatting 28

Intended layout of pages

Page regions

- A simple page can contain 1-5 regions, specified by child elements of the simple-page-master
- Let's refine the page masters with regions

SDPL 2005 Notes 6: XSL Formatting 29

Example: Region dimensions

```
<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>
```

SDPL 2005 Notes 6: XSL Formatting 30

Layout of Page Regions

SDPL 2005 Notes 6: XSL Formatting 31

Example: Region dimensions

```
<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>
```

SDPL 2005 Notes 6: XSL Formatting 32
Example: Page Sequences

Next: masters for sequences of content pages, using the defined simple-page-masters:

- repeatedly alternate masters for left and right pages:

```xml
<fo:page-sequence-master master-name="contents">
  <fo:repeatable-page-master-alternatives/>
</fo:page-sequence-master>
```

**Example: Contents of the Cover Page**

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

**Example: Content Pages Continue**

Finally, a page-sequence for content pages

- with static-content for the header and footer, and a flow for contents of pages:

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

**Example: Content Pages Continue**

Assign a flow of blocks to region-body:

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

Formatting and rendering this gives...

Page Sequences

- Other attributes of conditional-page-master-reference to select the page master to be used:
  - page-position="first, last, first-last, 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
Using FOs in Practise

- XSL FO instances should not be created manually.
- Instead, use XSLT style rules to create formatting objects:
  - `<xsl:root with layout masters for match="/*`
  - `<page-sequences with a flow for major parts (like chapters, or the entire document):


Examples of mapping content elements

- Formatting in-line emphasis:
  - `<fo:inline font-weight="bold">`
  - `<xsl:template />

- More examples in the exercises

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 [...] in 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."