6 XSL: Extensible Stylesheet Language

- An advanced style language for XML documents:
  1. Language for transforming XML documents: XSLT
  2. XML vocabulary for specifying formatting: XSL 1.0, W3C Rec. 10/01; XSL 1.1 Rec. 12/06
     » written for implementers of XSL processors
- 6.1 Introduction and Overview
- 6.2 XSL Formatting by Example

Example of XSL Syntax

- Formatting paragraphs (as part of a complete style sheet)

```xml
<xsl:stylesheet version='1.0' xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                   xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:template match="para[position() > 1]">
    <fo:block text-indent=""/>
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
</xsl:stylesheet>
```

Transformation & 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 ➔ interprets the result tree to produce formatted presentation

Basis of formatting

- Transformation adds info 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 FO classes (XSL 1.1 adds 25)
  - formatting properties control the presentation of formatting objects (indents, spacing, fonts, ...)
    » 248 formatting properties: many common with CSS2

Formatting

- 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 and space-before.optimum
    -> final rendering is not unique

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)

- Properties of the formatting object tree refined into traits (muotollipans, piire)
  - e.g., by propagating inherited properties, and computing absolute values for relative properties
    - e.g. properties
      - font-size="12pt", text-indent="2em"
    - become traits
      - font-size="12pt", text-indent="2em"
  - 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)

- 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

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

Benefits and Usage of XSL

- Powerful selection and manipulation (~XSLT)
  - Extended pagination and layout capabilities
    - area model a superset of the CSS2 box model
    - e.g. different writing directions: footnotes, page number refs.
  - Support of non-western writing directions
    - for distances in terms of before/after (for block-progression-direction), and start/end (for inline-progression-direction)
  - “XSL most useful to produce tailored, paginated documents on a server, in response to individual customer requests” (S. Deach, XSL 1.0 co-author, Adobe)

Content, Padding and Border

- For CSS-compatibility also margin-top, -right, -bottom and -left
- space-start/end not supported by FOP

Two area types

- block-areas
  - generated in block-progression-direction (before -> after, 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 (start -> end, 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 56 formatting objects …
  - page-sequence, single-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 249 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
  - A layout master set connects to a region with a matching region-name
    - the contents is distributed to that region of pages

Top-level formatting objects

- Slightly simplified:
  - `.fo:root`
    - contents of pages
  - `.fo:layout-master-set` + `.fo:page-sequence`
    - `.fo:flow` 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`

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

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-item`
        - `.fo:list-item-label`
        - `.fo:list-item-body`

Content objects for pages

- Slightly simplified:
  - `.fo:page-sequence`
    - `.fo:static-content`
      - (repeated on every page)
  - `.fo:flow`
    - (distributed to pages)

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-item`
        - `.fo:list-item-label`
        - `.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:page-sequence master-reference="page">
    <fo:block>World</fo:block>
    <fo:flow flow-name="xsl-region-body">
      <fo:block>Hello World</fo:block>
    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

Implementations?

- W3C XSL Rec rather recent (10/01, 12/07)
- 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 ... $5,000 (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 PS/SVG/TXT/...)
  - processor
  - Not complete, but implements a useful subset of XSL 1.0

FOP 0.93 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>39</td>
<td>6</td>
<td>14</td>
<td>59</td>
</tr>
<tr>
<td>objects</td>
<td>(66%)</td>
<td>(10%)</td>
<td>(24%)</td>
<td>59</td>
</tr>
<tr>
<td>(incl. 3 XSL 1.1 bookmark FOs)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 16 of 18 core functions implemented

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

```xml
<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>
```

- plus similar simple-page-masters
  - with master-name="rightPage" (identical)
  - master-name="leftPage" (left and right margins switched)

Intended layout of pages

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

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

Example: Region dimensions

```xml
<fo:simple-page-master master-name="cover"
  dimensions and margins as above … />
</fo:simple-page-master>
```

- And "rightPage" similarly … -->
- NB: body uses all space inside page margins
- margins of region-body have to accommodate other regions!

Layout of Page Regions
Example: Page Sequences

- Next: masters for sequences of content pages, using the defined simple-page-masters
  - repeatedly alternate masters for left and right pages:
    <fo:page-sequence-master-master-name="contents">...
    <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>

SDPL 2007 6: XSL Formatting 32

Example: Contents of the Cover Page

- Formatting the first page-sequence gives...

Example: Content Pages

- Finally, a page-sequence for content pages
  - with static-content for the header and footer, and a flow for contents of pages:
    <fo:page-sequence master-reference="contents" initial-page-number="2">...
    <fo:block font-family="Helvetica" font-size="10pt" text-align="end">...
    </fo:block>
    </fo:page-sequence>

Example: Content Pages Continue

- Content for page footers:
  <fo:block flow-name="xsl-region-after">...
  </fo:block>

- Finally, specify the content of the page body:

Example: Cover Page Formatted

- Other attributes of conditional-page-master-reference to select the page master to be used:
  - page-positions="first"
  - or "last", or "first" (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

Example: Content Pages Formatted

- Assign a flow of blocks to region-body:
  <fo:block flow-name="xsl-region-body">...
  </fo:block>

- Formatting and rendering this gives...

Watch this space!

Finally, specify the content of the page body:
Using FOs in Practise

- XSL FO instances should not be created manually.
- Instead, use XSLT rules to create formatting objects
  - fo:root with layout masters for match="/"
  - page-sequences with a flow for major parts (like chapters, or the entire document):
    <xsl:template match="chapter">
      <fo:page-sequence master-reference="...">
        <fo:flow flow-name="xml-region-body">
          <xsl:applytemplates/>
        </fo:flow>
      </fo:page-sequence>
    </xsl:template>

Examples of mapping content elements

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

- More 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...