6 XSL: Extensible Stylesheet Language

■ An advanced style language for XML documents:
  1. Language for transforming XML documents: XSLT
  2. XML vocabulary (of formatting objects) for specifying formatting semantics:
XSL version 1.0. W3C Rec. (15 October, 2001)
written for implementers of XSL processors
■ 6.1 Introduction and Overview
■ 6.2 Using XSL Formatting Objects

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 a standardised but mainly unimplemented SGML style language

Example of XSL syntax

■ Formatting paragraph elements (p):
  – NB: An incomplete style sheet!
<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="/" >
    <fo:block>
      <!-- settings for the first line: -->
      <fo:initial-property-set
        font-variant="small-caps"/>
      <xsl:apply-templates/>
    </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

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, inline text, page reference,
    x XSL 1.0 defines 56 formatting object classes
  – formatting properties control the presentation of formatting objects (indents, spacing, fonts, ...) 
    x XSL 1.0 defines 246 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
  - rendered form not uniquely defined by XSL

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 (a XSL FO Element&Attr tree)
  - consists of formatting objects with properties
  - more detailed: each character its own object

Generating the Area Tree (2/3)

- The last part of formatting describes the generation of a tree of geometric areas. These areas are positioned on a sequence of one or more pages.

Generating the Area Tree (3/3)

- Properties of the formatting object tree refined into traits
  - 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="2em"
  - traits control generation of areas out of formatting objects
  - some traits only available as a result of formatting, e.g., page numbers

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-language directions
  - distances specified in terms of before, after, start and end, relative to "writing-mode"
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

Two area types

- block-areas
  - generated in block-progression-direction
    - paragraphs and titles normally rendered using forblock, 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 forcharacter, which generates glyph-area inline-areas
  - no child areas, a single glyph image as content

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
  - used as a template for creating pages
  - specifies the geometry of pages
    - region-body (for page content)
    - region-before, region-after, region-start and region-end
    - (for header, footer, and left and right sidebar)

Content, Padding and Border

- space-before
- space-start
- padding-start
- space-end
- padding-end
- margin-start
- margin-end
- margin-top
- margin-right
- margin-bottom
- margin-left

For compatibility also CSS-like margins

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 246 properties
  - master-reference, background-color, font-family, font-size, space-before, end-indent, text-align, text-indent, ...
  - many common with CSS2

Page regions

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

Page
Region
Region
Region
Region
Region
Region
Top-level formatting objects

- Slightly simplified:
  - `fo:root`
  - `fo:layout-master-set`
  - `fo:page-sequence`

  - `(fo:region-body | fo:region-sequence-master)`
  - `fo:region-start`
  - `fo:region-after`
  - `fo:region-before`

  - Specify masters for page sequences by referring to `simple-page-master`

  - `contents of page`

Content objects for pages

- Slightly simplified:
  - `fo:page-sequence`
  - `fo:static-content`
  - `fo:flow`

  - Block-level object
  - Block-level object

‘Hello World’ result tree as an XSL document

```xml
<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>Hello World</fo:block>
  </fo:page-sequence>
</fo:root>
```

Some central formatting objects 2/3

- `fo:page-sequence`
  - Specifies the creation of page sequences
  - Possibly different page-sequence (and page-sequence-master) for each page

- `fo:flow`
  - Child objects of page-sequence
  - Flows connected to regions of page-master
  - Content of flows distributed to regions of pages

- NB: No ‘page’-formatting objects
  - Pages created by the formatter

Some central formatting objects 3/3

- `fo:block`
  - Commonly used for paragraphs, titles, ...
  - May contain text, other blocks, or
    - `fo:flow`
      - 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` and `fo:list-item-body`

Implementations?

- W3C XSL Recommendation rather recent
- What is the state of implementations?
- Some promising/interesting ones:
  - XEP by RenderX
  - Java-based XSL-FO to PDF/PS formatter
  - Commercial (~ $5000, April 2001): evaluation version free
  - Passive TeX
  - Set of TeX macro to process XSL-FO by Sebastian Rantz
  - Apache FOP
Apache FOP

- FOP (Formatting Objects Processor) by J. Tauber
  "for 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 MiFi/ClUTXT/...)
  processor
- Implements a useful subset of XSL 1.0 Rec;
  Version 0.20.3:
  - 41 formatting objects (out of 56)
  - 111 formatting properties (out of 246, or 228 w/o aural)

6.2 An XSL-FO Example

- From J. David Eisenberg: Using XSL Formatting Objects
  XML.com, January 17, 2001, (acknowledging the loan of
  some graphics)
- XSL FOs for a version of a handbook of Spanish
  - Tedious to manually mark-up document instances with
    XSL formatting objects; Think of this as the result of an
    XSLT transformation
- Overall structure of for: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="10cm"
    page-width="15cm"
    margin-top="0.5cm"
    margin-bottom="0.5cm"
    margin-left="1cm"
    margin-right="1.5cm">
    </fo:simple-page-master>
  </fo:layout-master-set>
```

---

Intended layout of pages

![Intended layout of pages]

Example: Region dimensions

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

<fo:simple-page-master master-name="leftPage" ... >
  <fo:region-before extent="1.5cm"/>
  <fo:region-body
    margin-top="1.5cm" margin-bottom="1.5cm"/>
</fo:simple-page-master>
```

---

Page regions

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

---

![Page regions]

- Let us refine the page masters with regions

---

Example: Region dimensions

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

<fo:simple-page-master master-name="leftPage" ... >
  <fo:region-before extent="1.5cm"/>
  <fo:region-body
    margin-top="1.5cm" margin-bottom="1.5cm"/>
</fo:simple-page-master>
```

---

- 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:

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

Page Sequences

- Other attributes of conditional-page-master-reference to select the page master to be used:
  - page-position="first";
    - or "last" or "next" (either 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: Contents of the Cover Page

```xml
<for:page-sequence master-reference="cover">
  <for:block font-family="Helvetica" font-size="12pt" text-align="end">
    Spanish Review Handbook
    Copyright © 2003 J. David Eisenhower
  </for:block>
  <for:block font-family="serif" font-size="12pt" text-align="end">
    A Catcode Production
  </for:block>
</for:page-sequence>
```

Example: Cover Page Formatted

- 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 page bodies:

```xml
<for:page-sequence master-reference="contents">
  <!-- Content of page header: -->
  <for:static-content flow-name="xsl-region-before">
    <for:block font-family="serif" font-size="12pt" text-align="center">
      Spanish Review Handbook
    </for:block>
  </for:static-content>
</for:page-sequence>
```
Example: Content Pages Continue

- Content for page footers:
  <!-- static-content is repeated on every page -->
  <xsl:template match="xsl-region-after">
    <fo:block font-family="Helvetica"
      font-size="1pt" text-align="center">
      Page #25. 
    </fo:block>
  </xsl:template>
- Finally, specify the content of page body:
  <xsl:template match="xsl-region-body">
    <fo:block>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

Example: Content Pages Continue

- Assign a flow of blocks to region-body:
  <xsl:template match="xsl-region-body">
    <fo:block font-size="14pt">
      Watch this space.
    </fo:block>
  </xsl:template>
- Normally all content of, say, a chapter would come here:
  <xsl:template match="xsl-page-sequence">
    <fo:block>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>
- Formatting and rendering this gives ...

Example: Content Pages Formatted

Using FOs in Practise

- No one should write XSL FO document instances by hand.
- Instead, use XSLT style rules to create formatting objects
  - <xsl:template match="xsl-region-body">
      <fo:block font-size="14pt">
        Watch this space.
      </fo:block>
    </xsl:template>
- page-sequence with a flow for major parts (like chapters, or the entire document):
  <xsl:template match="xsl-region-body">
    <fo:block font-size="14pt">
      Watch this space.
    </fo:block>
  </xsl:template>

Mapping content elements

- content elements would be mapped to blocks, inlines, list-blocks, tables, ... as appropriate
- For example, headers:
  <xsl:template match="header">
    <fo:block font-size="14pt" font-family="sans-serif" weight="bold">
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

Examples of mapping content elements

- Formatting in-line emphasis:
  <xsl:template match="emphasis">
    <fo:inline font-weight="bold" style="italic">
      <xsl:apply-templates/>
    </fo:inline>
  </xsl:template>
- More examples in the exercises

XML 1999  Note 4 XSL 31
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...