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)
<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='1em'/>
</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
   - interprets the result tree to produce formatted presentation

Transformation & Formatting

XSLT script
XSL Transform
XSL Formatter
Source XML
Result XML
XSL "fo" namespace

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, ...)
  - 245 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 ElemStr 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 (muotoliipiire, piirre)
  - 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="24pt"
  - traits control generation of areas out of formatting objects
  - some traits only available as a result of formatting, e.g., page numbers

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
  - > distances in terms of before/after (for block-leaves-direction), and start/end (for inline-leaves-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)

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

Two area types

- block-areas
  - generated in block-leaves-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-leaves-direction
    - (start ⇒ end; normally left-to-right)
  - characters rendered using fo:character, which generates glyph-areas 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-body, external-graphics, 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:flow**
  - child of a page-sequence
- **fo:region**
  - specifies the creation of page sequences
- **fo:table**
  - for formatting tabular material
- **fo:list-block**
  - to format lists of

Page Regions

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

Some central formatting objects 2/3

- **fo:root**
  - top node of the formatting object tree
  - a wrapper for all the rest
- **fo:flow**
  - child of a page-sequence
- **fo:table**
  - for formatting tabular material
- **fo:list-block**
  - to format lists of

Content objects for pages

- **fo:flow**
  - specifies the creation of page sequences
- **fo:region**
  - for layout
- **fo:table**
  - for formatting tabular material
- **fo:list-block**
  - to format lists of

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:list**
    - to format lists of
      - **fo:list-item**
        - with label
      - **fo:list-item**

Implementations?

- W3C XSL Rec rather recent (10/01, 12/06)
- 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
  - 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 0.94 XSL-FO Compliance

http://xml.apache.org/fop/compliance.html

- 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

<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>

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

- "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:

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

Page Sequences

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

```xml
<fo:page-sequence-master-reference="cover">
  <fo:flow-flow-name="xsl-region-body">
    <fo:block-font-family="Helvetica"
      font-size="10pt" text-align="end">
      Spanish Review Handbook</fo:block>
    <fo:block-font-family="Helvetica"
      font-size="12pt" text-align="end">
      Copyright © 2001 J. David Eisenberg</fo:block>
    <fo:block text-align="end">
      A Catcode Production</fo:block>
  </fo:flow-flow-name>
</fo:page-sequence-master-reference>
```

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 contents of pages:

```
xsl-region-before"
    font-size="10pt" text-align="center">
  Spanish Review Handbook</fo:block>
</fo:static-content>

Example: Content Pages Continue

- Content for page footers:

```
  "any"
  blank-or-not-blank="blank"/"not-blank"
```

Finally, specify the content of the page body:

Example: Content Pages Continue

- Assign a flow of blocks to region-body:

```
  "not-blank"
```

Formatting and rendering this gives ...

Example: Content Pages Formatted

- Watch this space!

```
  "rest" (neither first or last), or "any"

Finally, a page-sequence for content pages

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

```
  "any"
  blank-or-not-blank="blank"/"not-blank"

Finally, specify the content of the page body:

Example: Content Pages Continue

- Assign a flow of blocks to region-body:

```
  "rest" (neither first or last), or "any"
  blank-or-not-blank="blank"/"not-blank"
```

Formatting and rendering this gives ...

Example: Content Pages Formatted

- Watch this space!
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="/"` (next)
  - `page-sequences` with a `flow` for major parts
    (like chapters, or the entire document):

```
<xsl:template match="chapter">
  <fo:flow flow-name="xsl-region-body">
    <xsl:apply-templates/>
  </fo:flow>
</xsl:template>
```

Start of a Typical Stylesheet

```
<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:root>
      <fo:layout-master-set>
        <fo:template-master-name>
          <xsl:apply-templates/>
        </fo:template-master-name>
      </fo:layout-master-set>
      <xsl:apply-templates/>
    </fo:root>
  </xsl:template>
</xsl:stylesheet>
```

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-weight="bold"
    font-family="sans-serif" color="green"
    space-before="6pt" space-after="6pt">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>
```

Examples of mapping content elements

- Formatting in-line emphasis:

```
<xsl:template match="emph">
  <fo:inline font-style="italic">
    <xsl:apply-templates/>
  </fo:inline>
</xsl:template>
```

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 ...