Announcements (October 24)

- Homework #3 due next Tuesday
- Project milestone #2 due November 9

XSLT

- XML-to-XML rule-based transformation language
- Used most frequently as a stylesheet language
- An XSLT program is an XML document itself
- Version 1.0: W3C recommendation
- Version 2.0: still a candidate recommendation, developed in conjunction with XPath 2.0

XSLT processor

Input XML  ➔  XSLT program  ➔  Output XML

Actually, output does not need to be in XML in general.
XSLT program

- An XSLT program is an XML document containing
  - Elements in the `<xsl:` namespace
  - Elements in user namespace
- The result of evaluating an XSLT program on an input XML document = the XSLT document where each `<xsl:` element has been replaced with the result of its evaluation
- Basic ideas
  - Templates specify how to transform matching input nodes
  - Structural recursion applies templates to input trees recursively
- Uses XPath as a sub-language

XSLT elements

- Element describing transformation rules
  - `<xsl:template>`
- Elements describing rule execution control
  - `<xsl:apply-templates>`
  - `<xsl:call-template>`
- Elements describing instructions
  - `<xsl:if>`, `<xsl:for-each>`, `<xsl:sort>`, etc.
- Elements generating output

XSLT example

- Find titles of books authored by "Abiteboul"

```xml
<?xml version="1.0"?>
<xsl:stylesheet
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 version="2.0">
 <xsl:template match="book[author='Abiteboul']">
   <booktitle>
     <xsl:value-of select="title"/>
   </booktitle>
 </xsl:template>
</xsl:stylesheet>
```

- Not quite; we will see why later
<xsl:template>
  <xsl:template match="book[author='Abiteboul']">
    <booktitle>
      <xsl:value-of select="title"/>
    </booktitle>
  </xsl:template>

  <xsl:template match="book[author='Abiteboul']">
    <booktitle>
      <xsl:value-of select="title"/>
    </booktitle>
  </xsl:template>
</xsl:template>

Template in action

Example XML fragment

Example XML fragment

Removing the extra output

Add the following template:

This template matches all text and attributes

XPath features

Body of the rule is empty, so all text and attributes become empty string

This rule effectively filters out things not matched by the other rule
Again, find titles of books authored by “Abiteboul”;
but make the output look like <book title="booktitle"/>

A more general method

Another slightly different example: return (entire) books authored by “Abiteboul”

Formatting XML into HTML

Example templates to

* Render a book title in italics in HTML
* Render the authors as a comma-separated list
* Allows precise control of white space in output
Example: generate a table of contents
- Display books in an HTML unordered list
- For each book, first display its title, and then display its sections in an HTML ordered list
- For each section, first display its title, and then display its subsections in an HTML ordered list

Example continued

One problem remains
- Even if a book or a section has no sections, we will still generate an empty \(<ol/>\) element

A fix using \(<xsl:if>\): replace \(<ol/>\)<xsl:apply-templates select="section"/>\(</ol>\) with
\(<xsl:if test="section"/>\)<ol><xsl:apply-templates select="section"/>\(</ol>\)</xsl:if>

The body of \(<xsl:if test="xpath_cond"/>\) is processed only if \(xpath\_cond\) evaluates to true
White space control

- White space is everywhere in XML.
  
```xml
<book ISBN="ISBN-10" price="80.00">
  
  <title>
    Foundations of Databases
  </title>

  "Alaa" goes into a text node
  "Foundations of Databases Alaa" goes into another text node

- Specify `<xsl:strip-space elements="*"/>` to remove text nodes (under any element) containing only white space.

- To strip leading and trailing white space and replace any sequence of white space characters by a single space, specify `<xsl:template match="text()">
  <xsl:value-of select="normalize-space(.)"/>
</xsl:template>`

<xsl:for-each>

- `<xsl:for-each select="$xpath_expr">
  
  body

  - Process `body` for each node in the node-set returned by `$xpath_expr`
  - Processing context changes to the node being processed

- Another way to render authors as a comma-separated list

  `<xsl:template match="book">
  
  <xsl:for-each select="author">
    <xsl:if test="position()>1">, </xsl:if>
    <xsl:value-of select="normalize-space(.)"/>
  </xsl:for-each>

  </xsl:template>`

Named templates with parameters

- Define a generic template for rendering a list of things as a comma-separated list

  Cannot use `match` because we do not know in advance the things to render

  `<xsl:template name="comma-separated-list">
  
  <xsl:param name="things-to-be-formatted"/>

  <xsl:for-each select="$things-to-be-formatted">
    <xsl:if test="position()>1">, </xsl:if>
    <xsl:value-of select="normalize-space(.)"/>
  </xsl:for-each>

  </xsl:template>`
Calling templates & passing parameters

- Use the generic template
  <xsl:template match="book">
    <xsl:value-of select="normalize-space(title)"/>
    <xsl:text>: </xsl:text>
    <xsl:call-template name="comma-separated-list">
      <xsl:with-param name="things-to-be-formatted"
        select="author"/>
    </xsl:call-template>
  </xsl:template>

- <xsl:with-param name="para_name" select="xpath_expr"/>
  <xsl:call-template>
    invokes the named template without changing the context
  </xsl:call-template>

XSLT summary

- Used often as a stylesheet language, but can be considered a query language too
  - Grouping in XSLT 2.0 (<xsl:for-each-group>)
  - Very expressive, with full recursion
    - Cannot be replaced by XQuery?
      - Well, XQuery supports user-defined functions, which can be recursive
    - Easily non-terminating, difficult to optimize
      - Cannot replace XQuery
  - So many features, so little time! ☺

Review

- XML: tree (or graph)-structured data
- DTD: simple schema for XML
  - Well-formed XML: syntactically correct
  - Valid XML: well-formed and conforms to a DTD
- XPath: path expression language for XML
  - An XPath expression selects a list of nodes in an XML document
  - Used in other languages
- XQuery: SQL-like query language for XML
  - FLWOR expression, quantified expression, aggregation, etc.
- XSLT: stylesheet language for XML, in XML
  - Transforms input XML by applying template rules recursively on the structure of input XML