Announcements (October 25)

- Homework #3 due next Tuesday
- Project milestone #2 due Nov. 10
- My office hours today are cancelled
  - Moved to Wednesday 2-3pm instead

XSLT

- XML-to-XML rule-based transformation language
- An XSLT program is an XML document itself
- Used most frequently as a stylesheet language
- Version 1.0 a W3C recommendation
- Version 2.0 under development together with XPath 2.0

XSLT processor

Input XML  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="match_expr">
    <xsl:value-of select="xpath_expr"/>
  </xsl:template>

  <xsl:value-of select="booktitle"/>
  </xsl:template>

  ™
</xsl:template>

Template in action

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

Example XML fragment

Template applies

Template does not apply; default behavior is to process the node recursively and print out all text nodes

Removing the extra output

Add the following template:

This template matches all text and attributes

XPath features

- text() is a node test that matches any text node
- @* matches any attribute
- | means "or" in XPath

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
<xsl:attribute>
- Again, find titles of books authored by “Abiteboul”; but make the output look like <book title="booktitle"/>
  <xsl:template match="book[author='Abiteboul']">
    <book title="normalize-space(title)"/>
  </xsl:template>
- A more general method
  <xsl:template match="book[author='Abiteboul']">
    <book>
      <xsl:attribute name="title">
        <xsl:value-of select="normalize-space(title)"/>
      </xsl:attribute>
    </book>
  </xsl:template>
- adds an attributed named attr with value body to the parent element in the output
</xsl:attribute>

<xsl:copy-of>
- Another slightly different example: return (entire) books authored by “Abiteboul”
  <?xml version="1.0">
  <xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="2.0">
    <xsl:template match="text()|@*"/>
    <xsl:template match="book[author='Abiteboul']">
      <xsl:copy-of select="."/>
    </xsl:template>
  </xsl:stylesheet>
- copies the entire contents (including tag structures) of the node-set returned by xpath_expr to the output
</xsl:copy-of>

Formatting XML into HTML
- Example templates to
  - Render a book title in italics in HTML
  - Render the authors as a comma-separated list
    <xsl:template match="book/title">
      <i><xsl:value-of select="normalize-space(.)"></i>
    </xsl:template>
    <xsl:template match="book/author[1]">
      <xsl:value-of select="normalize-space(.)"/>
    </xsl:template>
    <xsl:template match="book/author[position()>1]">
      <xsl:text>, </xsl:text>
      <xsl:value-of select="normalize-space(.)"/>
    </xsl:template>
    <xsl:template>
      <xsl:text> 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

```xml
<xsl:template match="title">
  <xsl:value-of select="normalize-space(.)"/>
</xsl:template>
<xsl:template match="section">
  <li>
    <xsl:apply-templates select="title"/>
    <ol><xsl:apply-templates select="section"/></ol>
  </li>
</xsl:template>
```

(Continue on next slide)

Example continued

```xml
<xsl:template match="book">
  <li>
    <xsl:apply-templates select="title"/>
    <ol><xsl:apply-templates select="section"/></ol>
  </li>
</xsl:template>
<xsl:template match="bibliography">
  <html>
    <head><title>Bibliography</title></head>
    <body>
      <ul><xsl:apply-templates select="book"/></ul>
    </body>
  </html>
</xsl:template>
```

One problem remains

- Even if a book or a section has no sections, we will still generate an empty `<ol></ol>` element.

A fix using `<xsl:if>`: replace

```xml
<ol><xsl:apply-templates select="section"/></ol>
```

with

```xml
<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.
  <book ISBN="ISBN-10" price="80.00">...
  ...<title>
  ..."<tt>FoundationsofDatabases</tt>...
  ...
- "<tt>" goes into a text node.
- "<tt>" 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
  <xsl:for-each>
  - 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:
  <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"` evaluates `xpath_expr` and passes its result as the value of the parameter `para_name`
- `<xsl:call-template>` invokes the named template without changing the context

XSLT summary

- Used often as a stylesheet language, but can be considered a query language too
  - Very expressive, with full recursion
    - Cannot be replaced by XQuery?
      - Well, XQuery actually support user-defined functions, which can be recursive
  - Easily non-terminating, difficult to optimize
    - Cannot replace XQuery
  - Features like dynamic scoping really help in text processing
- 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