Announcements (Tue. Oct. 25)
- Homework #3 due in 1½ weeks
- Project milestone #2 due in 2½ weeks

XSLT

- XML-to-XML rule-based transformation language
  - Used most frequently as a stylesheet language
  - An XSLT program is an XML document itself
  - Current version is 2.0; W3C recommendation since January 2007

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 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

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

- <xsl:template match="match_expr"> is the basic XSLT construct
describing a transformation rule
  - `match_expr` is an XPath-like expression specifying which nodes this
    rule applies to
- <xsl:value-of select="xpath_expr"/> evaluates `xpath_expr`
  within the context of the node matching the template, and converts
  the result sequence to a string
- `<booktitle>` and </booktitle> simply get copied to the output
  for each node match

Removing the extra output
- Add the following template:
  <xsl:template match="text()|@*"/>
- 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>
  -  
    <xsl:attribute name="attr">body</xsl:attribute>
    adds an attributed named `attr` with value `body`
    to the parent element in the output

<xsl:copy-of/>
- Another slightly different example: return (entire) books
  authored by "Abiteboul"
  -  
    </xsl:stylesheet
-  
  </xsl:stylesheet>
  -  
- 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

<xs1:template match="title">
  <xsl:value-of select="normalize-space(.)"/>
</xs1:template>
<xsl:template match="section">
  <li>
    <xsl:apply-templates select="title"/>
    <ol><xsl:apply-templates select="section"/></ol>
  </li>
</xs1:template>
(Continue on next slide)

Example continued

<xs1:template match="book">
  <li>
    <xsl:apply-templates select="title"/>
    <ol><xsl:apply-templates select="section"/></ol>
  </li>
</xs1:template>
<xs1:template match="bibliography">
  <html>
    <head><title>Bibliography</title></head>
    <body>
      <ul><xsl:apply-templates select="book"/></ul>
    </body>
  </html>
</xs1:template>

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

A fix using <xs1:if>: replace
<ol><xsl:apply-templates select="section"/></ol>
with
<ol><xsl:apply-templates select="section"/></ol>
</xs1:if>

The body of <xs1:if test="xpath_cond"> is processed only if xpath_cond evaluates to true

Output control
<xs1:output method="html" indent="yes"/>

- Specifies that output
  - Will be HTML
  - Will be indented to make reading easier
- Other possible method values include "text", "xml"
  - For XML output method, set omit-xml-declaration="yes" to suppress "<?xml ...?>" at the beginning of the output

White space control
- White space is everywhere in XML

  <book ISBN="ISBN-10" price="80.00">
    Foundations of Databases
  </book>

- "..." goes into a text node (assuming no DTD)
- "..." goes into another text node
- Specify <xs1: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 <xs1:template match="text()">
  <xsl:value-of select="normalize-space()"/>
</xs1:template>

Another way to render authors as a comma-separated list
<xs1:for-each select="xpath_expr">
  body
  </xs1:for-each>
  - Process body for each node in the node-set returned by xpath_expr
  - Processing context changes to the node being processed
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

```xml
<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

```xml
<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>
  <br/>
</xsl:template>
```

Other useful features

- `<xsl:text>&#10;</xsl:text>` inserts a newline in the output
- `<xsl:message>` for debugging
  - `<xsl:message terminate="yes"/>` exits the program
- `<xsl:variable>` defines a (constant) variable
- `<xsl:function>` defines a function
- `<xsl:key>` defines a key that can be used for lookups

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
- XML Schema: a more sophisticated schema for XML
- 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