Announcements (Thu. Oct. 24)

- Homework #3 assigned
  - Due in 1 1/2 weeks
- Project milestone #2 due in 3 weeks
  - Feedback for milestone #1 coming this weekend
- Graded midterm for pick-up outside my office
  - See last set of lecture slides for score distribution

Announcements (Tue. Oct. 29)

- Homework #3 due in one week
- Project milestone #2 due Nov. 14
  - Milestone #1 feedback sent in email
- Data+Journalism talk next Monday before noon
  - "Vibrations" versus Data: What Media Coverage Gets Wrong about Presidential Elections
  - Brendan Nyhan, Professor of Government at Dartmouth College, and co-author of All the President's Spin, a New York Times bestseller

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 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
  ```

Template in action

- Example XML fragment
  ```xml
  <book ISBN="ISBN-10" price="80.00">
    <years>1995</years>
    <title>A First Course in Databases</title>
    <publisher>Addison Wesley</publisher>
    <author>Vianu</author>
    <publisher>Prentice-Hall</publisher>
    <year>1995</year>
  </book>
  ```

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

- Template does not apply; default behavior is to process the node recursively and print all text nodes
  ```xml
  <xsl:template match="text()|@*"/>
  ```

Removing the extra output

- Add the following template:
  ```xml
  <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"`
  ```xml
  <xsl:template match="book[author='Abiteboul']">
    <booktitle>
      <xsl:value-of select="normalize-space(title)"/>
    </booktitle>
  </xsl:template>
  ```

- A more general method
  ```xml
  <xsl:template match="book[author='Abiteboul']">
    <book>
      <xsl:attribute name="title">
        <xsl:value-of select="normalize-space(title)"/>
      </xsl:attribute>
    </book>
  </xsl:template>
  ```

<xsl:copy-of>

- Another slightly different example: return (entire) books authored by "Abiteboul"
  ```xml
  <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
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/author[position()>1]">`<xsl:template>`
- `<xsl:template match="book/author[1]">`<xsl:template>`

- `<xsl:template>` allows precise control of white space in output.

Example continued

- `<xsl:template match="book">`<xsl:template>`
- `<xsl:template match="bibliography">`<xsl:template>`
- `<xsl:template>`<html>`<xsl:template>`

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

<xsl:apply-templates>

- 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.
- `<xsl:template match="book/author[position()>1]">`<xsl:template>`
- `<xsl:template match="book/author[1]">`<xsl:template>`

(Continue on next slide)

<xsl:if>

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

- The body of `<xsl:if test="xpath_cond">` is processed only if `xpath_cond` evaluates to true.

Output control

- `<xsl: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.
- `<text>`<text>`
- `<text>`<text>`
- `<text>`<text>`
- `<text>`<text>`

- "..." goes into a text node (assuming no DTD)
- "..." goes in 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:template>`
- `<xsl:value-of select="normalize-space()"></xsl:value-of>`
- `<xsl:template>`
<xsl:for-each>
  <xsl:for-each select="$xpath_expr">
    <xsl:value-of select="normalize-space(.)"/>
  </xsl:for-each>
</xsl:for-each>

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:call-template name="comma-separated-list">
  - <xsl:text>: </xsl:text>
  - <xsl:value-of select="normalize-space(title)"/>
  - </xsl:call-template>
  - </xsl:template>

Calling templates & passing parameters

- Use the generic template
  - <xsl:template match="book">
  - <xsl:with-param name="things-to-be-formatted" select="$xpath_expr">
  - <xsl:call-template name="comma-separated-list">
  - <xsl:variable>
  - </xsl:call-template>
  - </xsl:template>

Other useful features

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