**Announcements**
- Homework #3 to be assigned next Monday (October 27)
- Course project milestone 2 due on November 12

---

**XSLT**

CPS 196.3
Introduction to Database Systems

---

**XSLT**

- W3C recommendation
- XML-to-XML rule-based transformation language
- An XSLT program is an XML document itself
- Used most frequently as a stylesheet language

![Diagram of XSLT process]

- Input XML
- XSLT processor
- 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
- 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>`
- Elements generating output
  - `<xsl:value-of>`
  - `<xsl:attribute>`
  - `<xsl:copy-of>`
  - `<xsl:text>`

---

**XSLT example**

- Find titles of books authored by "Abiteboul"

```xml
<?xml version="1.0"?>
<xs1:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
  <xsl:template match="book[author='Abiteboul']">
    <booktitle>
      <xsl:value-of select="title"/>
    </booktitle>
  </xsl:template>
</xs1: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>

- <xsl:template match="*">
  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="*"/>
  evaluates xpath_expr within the context of the node matching the template, and converts
  the result node-set 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>
      <xsl:attribute name="title">
        <xsl:value-of select="title"/>
      </xsl:attribute>
    </book>
  </xsl:template>
- A more general method
  - <xsl:template match="book[author='Abiteboul']">
    <book>
      <xsl:attribute name="attr">
        <xsl:value-of select="body"/>
      </xsl:attribute>
    </book>
    - <xsl:attribute name="attr">
      body</xsl:attribute>
    </xsl:template>
      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
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
    <xsl:stylesheet>
      <xsl:template match="text()|@*"/>
      <xsl:copy-of select="*"/>
    </xsl:template>
  </xsl:stylesheet>
- <xsl:copy-of select="*"/>
  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/title">
      <i><xsl:value-of select="."/></i>
    </xsl:template>
    <xsl:template match="book/author[1]">
      <xsl:value-of select="."/>
    </xsl:template>
    <xsl:template match="book/author[position()>1]">
      <xsl:value-of select="."/>
    </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

Example continued

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

White space control
- White space is everywhere in XML.
- To strip leading and trailing white space and replace any sequence of white space characters by a single space, specify
  \(<\text{xsl:template match="text()">}\n  \<\text{xsl:value-of select="normalize-space()"/>}\n  \<\text{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

A fix using \(<\text{xsl:if}>\): replace
\(<\text{o}l/>\<\text{xsl:apply-templates select="section"/></\text{o}l/>\)
with
\(<\text{o}l/>\<\text{xsl:if test="section">\<\text{o}l/>\</\text{xsl:if>
- The body of \(<\text{xsl:if test="xpath_cond">\) is processed only if \(xpath\_cond\) evaluates to true

Another way to render authors as a comma-separated list
\(<\text{xsl:template name="comma-separated-list">}\n  \<\text{xsl:param name="things-to-be-formatted"/>}\n  \<\text{xsl:for-each select="$things-to-be-formatted">}\n  \<\text{xsl:if test="position()>1">, \</text{xsl:if>\n  \<\text{xsl:value-of select="."/>}\n  \</text:xsl:for-each>\n  \<\text{xsl:for-each>
Calling templates & passing parameters

- Use the generic template
  
  ```xml
  <xsl:template match="book">
    <xsl:value-of select="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>
  ```

- `<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
  - Easily non-terminating, difficult to optimize
    - Cannot replace XQuery
- So many features, so little time! 😊