

# XSLT

CPS 116

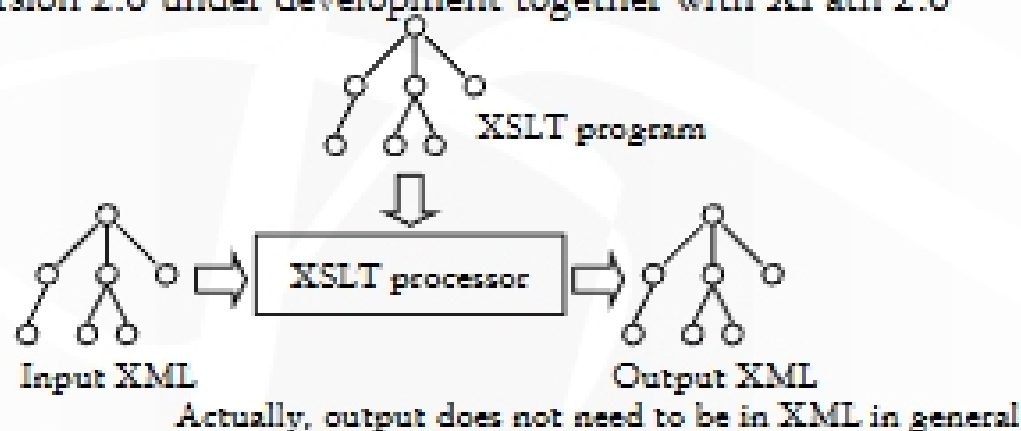
Introduction to Database Systems

## 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 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
  - `<xsl:value-of>`, `<xsl:attribute>`, `<xsl:copy-of>`, `<xsl:text>`, etc.

## XSLT example

- ❖ Find titles of books authored by "Abiteboul"
- ```
<?xml version="1.0"?> Standard header of an XSLT document
<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>

7

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

## Template in action

8

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

### ❖ Example XML fragment

```
<book ISBN="ISBN-10" price="80.00">
  <title>Foundations of Databases</title>
  <author>Abiteboul</author>
  <author>Null</author>
  <author>Vianu</author>
  <publisher>Addison Wesley</publisher>
  <year>1998</year>
  <section"></section">
</book>
<book ISBN="ISBN-20" price="40.00">
  <title>A First Course in Databases</title>
  <author>Ullman</author>
  <author>Widom</author>
  <publisher>Prentice-Hall</publisher>
  <year>2002</year>
  <section"></section">
</book>
```

### Template applies

```
<booktitle>
  Foundations of Databases
</booktitle>
```

### Template does not apply;

default behavior is to process the node recursively and print out all

### text nodes

```
A First Course in Databases
Ullman
Widom
Prentice-Hall
2002
--
```

## Removing the extra output

9

- ❖ 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>

10

- ❖ 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">val</xsl:attribute>` adds an attributed named `attr` with value `val` to the parent element in the output

## <xsl:copy-of>

11

- ❖ 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>
```

- ❖ `<xsl:copy-of select="xpath_expr"/>` copies the entire contents (including tag structures) of the node-set returned by `xpath_expr` to the output

## Formatting XML into HTML

12

- ❖ 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:text>` allows precise control of white space in output

## <xsl:apply-templates>

13

### ❖ 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="title">
  <xsl:value-of select="normalize-space(.)"/>
</xsl:template>
<xsl:template match="section">
  <i>
    <xsl:apply-templates select="title"/>
    <ol><xsl:apply-templates select="section"/></ol>
  </i>
</xsl:template>
<xsl:apply-templates select="xpath_expr"/>
  applies templates recursively to the node-set
  returned by xpath_expr
(Continue on next slide)
```

## Example continued

14

```
<xsl:template match="book">
  <i>
    <xsl:apply-templates select="title"/>
    <ol><xsl:apply-templates select="section"/></ol>
  </i>
</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

## <xsl:if>

15

### ❖ 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

16

### ❖ White space is everywhere in XML

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

- " " goes into a text node
  - " Foundations of Databases " 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>

17

### ❖ <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

18

### ❖ 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>
```