

XSLT

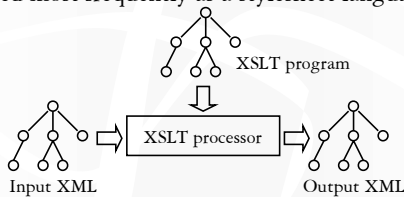
CPS 196.3

Introduction to Database Systems

XSLT

2

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



Actually, output does not need to be in XML in general

XSLT program

3

- ❖ 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

4

- ❖ Element describing transformation rules
 - `<xsl:template>`
- ❖ Elements describing rule execution control
 - `<xsl:apply-template>`
 - `<xsl:call-template>`
- ❖ Elements describing instructions
 - `<xsl:if>`, `<xsl:for-each>`, `<xsl:sort>`, etc.

XSLT example

5

- ❖ 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/XSLT/Transform"
  version="1.0">
<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>`

6

- ```
<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" />` converts the node-set returned by *xpath\_expr* to a string
  - ❖ `<booktitle>` and `</booktitle>` simply get copied to the output for each node match

---

---

---

---

---

---

---

---

## Template in action

7

```
<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>Hull</author>
 <author>Vianu</author>
 <publisher>Addison Wesley</publisher>
 <year>1995</year>
</book>
<book ISBN="ISBN-20" price="40.00">
 <title>A First Course in Databases</title>
 <author>Ulman</author>
 <author>Widom</author>
 <publisher>Prentice-Hall</publisher>
 <year>2002</year>
</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  
Ulman  
Widom  
Prentice-Hall  
2002  
--

---

---

---

---

---

---

---

---

## Removing the extra output

8

### ❖ 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>

9

### ❖ 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="{title}"/>
</xsl:template>
...
```

### ❖ A more general method

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

10

- ❖ Another slightly different example: return (entire) books authored by "Abiteboul"

```
<?xml version="1.0">
<xsl:stylesheet
 xmlns:xsl="http://www.w3.org/1999/XSLT/Transform"
 version="1.0">
<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

11

- ❖ 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:apply-templates>

12

- ❖ 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="." />
</xsl:template>
<xsl:template match="section">

 <xsl:apply-templates select="title" />
 <xsl:apply-templates select="section" />

</xsl:template>
<xsl:apply-templates select="xpath_expr" />
```

(Continue on next slide) applies templates recursively to the node-set returned by *xpath\_expr*

---

---

---

---

---

---

---

---

## Example continued

13

```
<xsl:template match="book">

 <xsl:apply-templates select="title"/>
 <xsl:apply-templates select="section"/>

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

14

- ❖ 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

---

---

---

---

---

---

---

---

---

---

## Whitespace control

15

- ❖ Whitespace is everywhere in XML

```
...
<book ISBN="ISBN-10" price="80.00">
 <title>
 Foundations of Databases
 </title>
 ...
 <xsl:strip-space elements="*" />
 <xsl:template match="text()">
 <xsl:value-of select="normalize-space()" />
 </xsl:template>
```
- “`<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 whitespace
- ❖ To strip leading and trailing whitespace and replace any sequence of whitespace characters by a single space, specify `<xsl:template match="text()"><xsl:value-of select="normalize-space()" /></xsl:template>`

---

---

---

---

---

---

---

---

---

---

## <xsl:for-each>

16

- ❖ `<xsl:for-each select="xpath_expr">`  
*body*  
`</xsl:for-each>`
  - Process *body* for each node in the node-set returned by *xpath\_expr*
- ❖ 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="."/>`  
`</xsl:for-each>`  
... ..  
`</xsl:template>`
  - No need to have separate templates for authors

---

---

---

---

---

---

---

---

## XSLT summary

17

- ❖ 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!

---

---

---

---

---

---

---

---