XML Technologies and Applications

Rajshekhar Sunderraman

Department of Computer Science
Georgia State University
Atlanta, GA 30302
raj@cs.gsu.edu

V (b). XML Querying: XSLT

December 2005
Outline

- Introduction
- XML Basics
- XML Structural Constraint Specification
  - Document Type Definitions (DTDs)
  - XML Schema
- XML/Database Mappings
- XML Parsing APIs
  - Simple API for XML (SAX)
  - Document Object Model (DOM)
- XML Querying and Transformation
  - XPath
  - XSLT
  - XQuery
- XML Applications
XSL (eXtensible Stylesheet Language)

- XSL is a high-level functional language used to transform XML documents into various formats (XML, HTML etc.)
- XSL program consists of a set of TEMPLATE rules.
- Each rule consists of a pattern and a template.
  - Pattern (XPath expression) => where clause
    template => construct clause
  - XSL processor starts from the root element and tries to apply a pattern to that node; If it succeeds, it executes the corresponding template.
    - The template, when executed, usually instructs the processor to produce some XML result and to apply the templates
    - Recursively on the node's children.
- An XSL style sheet is a valid XML document
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
    <cd country="UK">
        <title>Dark Side of the Moon</title>
        <artist>Pink Floyd</artist>
        <price>10.90</price>
    </cd>
    <cd country="UK">
        <title>Space Oddity</title>
        <artist>David Bowie</artist>
        <price>9.90</price>
    </cd>
    <cd country="USA">
        <title>Aretha: Lady Soul</title>
        <artist>Aretha Franklin</artist>
        <price>9.90</price>
    </cd>
</catalog>
Applying XSLT Stylesheets to XML Documents

There are three ways of applying an XSLT stylesheet to an XML document:

- Directly applying an XSLT processor to the XML document and the XSLT stylesheet
- Calling an XSLT processor from within a (Java) program
- Adding to the XML document a link to the XSL stylesheets and letting the browser do the transformation
Using an XSL Processor

XML document \rightarrow \text{XSL Processor} \rightarrow \text{Result is either an XML, HTML or text document}

\text{XSL stylesheet}

\text{java org.apache.xalan.xslt.Process}
-\text{IN myXmlFile.xml -XSL myXsltFile.xsl -OUT myOutputFile.html}

Directly applying the Xalan XSL processor
Letting a Browser Perform the Transformation

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="catalog.xsl"?>
<catalog>
    <cd country="UK">
        <title>Dark Side of the Moon</title>
        <artist>Pink Floyd</artist>
        <price>10.90</price>
    </cd>
    ...
</catalog>
```

A link to the stylesheet
The Root of the XSL Document (program)

The Root of the XSL document should be one of the following lines:

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

```xml
<xsl:transform version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

The namespace allows the XSL processor to distinguish between XSL tags and tags of the result document.
How Does XSLT Work?

- An XSL stylesheet is a collection of templates that are applied to source nodes (i.e., nodes of the given XML document).

- Each template has a match attribute that specifies to which source nodes the template can be applied.

- The current source node is processed by applying a template that matches this node.

- Processing always starts at the root (/).
Templates

• A template has the form

  <xsl:template match="pattern">  
    ... Template content ...  
  </xsl:template>

• The content of a template consists of
  - XML elements and text (HTML etc) that are copied to the result
  - XSL elements that are actually instructions

• The pattern syntax is a subset of XPath
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <body>
      <h1>Hello World</h1>
    </body>
  </html>
</xsl:template>

</xsl:stylesheet>
Hello World

Applying a browser to catalog.xml (catalog.xml has a link to catalog.xsl)
The XSL Element `<xsl:apply-templates>`

- Processing starts by applying a template that matches the root (`/`)
  - If the given XSL stylesheet does not have a template that matches the root, then one is inserted by default (“Default Templates”)

- The XSL stylesheet must specify explicitly whether templates should be applied to descendants of the root

- It is done by putting inside a template the instruction:
  ```xml
  <xsl:apply-templates select="xpath"/>
  ```

- Without the select attribute, this instruction processes all the children of the current node
<xsl:template match="cd">
  <h2>A CD!</h2>
</xsl:template>
Default Templates

• XSL provides implicit built-in templates that match every element and text nodes

```xml
<xsl:template match="/ | */">
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="text()">
  <xsl:value-of select="."/>
</xsl:template>
```

• Templates we write always override these built-in templates (when they match)
Interaction of Default and User Defined Templates

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="cd[title='Space Oddity']">
    <html>
        <body>
            <h1>Hello World</h1>
        </body>
    </html>
</xsl:template>

</xsl:stylesheet>
```
The default templates print the text in the leaves of the first and third CD's.

Dark Side of the Moon
Pink Floyd
10.90

<b>Hello World</b>

Aretha: Lady Soul
Aretha Franklin
9.90

This is printed by the template from the previous slide.
The Most Frequently Used Elements of XSL

- `<xsl:value-of select="xpath-expression"/>`
  - This element extracts the value of a node from the nodelist located by `xpath-expression`

- `<xsl:for-each select="xpath-expression"/>`
  - This element loops over all the nodes in the nodelist located by `xpath-expression`

- `<xsl:if test="xpath-expression"/>
  `<xsl:if test="xpath-expression=value"/>
  etc.
  - This element is for conditional processing
The `<xsl:value-of>` Element

```xml
<xsl:value-of select="xpath-expression"/>
```

- The XSL element `<xsl:value-of>` can be used to extract the value of an element that is selected from the source XML document.
- The extracted value is added to the output stream.
- The selected element is located by an XPath expression that appears as the value of the `select` attribute.
Suppose we want to produce the above web page from the XML file.
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <body>
      <h2>A CD Catalog</h2>
      <table border="1">
        <tr bgcolor="yellow">
          <th>Title</th>
          <th>Artist</th>
        </tr>
      </table>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
Note that only the first matched element is retrieved for each `<xsl:value-of>`
The `<xsl:for-each>` Element

```xml
<xsl:for-each select="xpath-expression"/>
```

- The `<xsl:for-each>` element loops over all the nodes in the nodelist of the XPath expression that appears as the value of the `select` attribute.

- The value of each node can be extracted by an `<xsl:value-of>` element.
Example of `<xsl:foreach>` Element

A CD Catalog

<table>
<thead>
<tr>
<th>Title</th>
<th>Artist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dark Side of the Moon</td>
<td>Pink Floyd</td>
</tr>
<tr>
<td>Space Oddity</td>
<td>David Bowie</td>
</tr>
<tr>
<td>Aretha: Lady Soul</td>
<td>Aretha Franklin</td>
</tr>
</tbody>
</table>

All the values are selected

Suppose we want to produce the above web page from the XML file.
As in the previous example
<xsl:for-each select="catalog/cd">
  <tr>
    <td><xsl:value-of select="title"/></td>
    <td><xsl:value-of select="artist"/></td>
  </tr>
</xsl:for-each>
</table>
</body>
</html>

Note that all the /catalog/cd elements are retrieved
Consider the following change in the select attribute:

```xml
<xsl:for-each
    select="catalog/cd[price<10]">
    <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
    </tr>
</xsl:for-each>
</table>
</body>
</html>
```

Only elements that satisfy /catalog/cd[price<10] are retrieved.
The `<xsl:sort>` Element

Suppose we want to produce the above web page from the XML file.
The `<xsl:sort>` Element

- The `<xsl:sort>` element is used to sort the list of nodes that are looped over by the `<xsl:for-each>` element.

- Thus, the `<xsl:sort>` must appear inside the `<xsl:for-each>` element.

- The looping is done in sorted order.
Example of `<xsl:sort>` Element

A CD Catalog

<table>
<thead>
<tr>
<th>Title</th>
<th>Artist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aretha: Lady Soul</td>
<td>Aretha Franklin</td>
</tr>
<tr>
<td>Space Oddity</td>
<td>David Bowie</td>
</tr>
<tr>
<td>Dark Side of the Moon</td>
<td>Pink Floyd</td>
</tr>
</tbody>
</table>

Sorted by the name of the artist
The /catalog/cd elements are sorted according to the value of the artist element.
The `<xsl:if>` Element

- The `<xsl:if>` element is used for conditional processing.

- The condition appears as the value of the `test` attribute, for example:
  ```xml
  <xsl:if test="price > 10">
    some output ...
  </xsl:if>
  ```

- The elements inside the `<xsl:if>` element are processed if the condition is true. Processing the inside elements means:
  - Copying them into the output stream if they are not XSL elements, and
  - Evaluating them if they are XSL elements.

- If the value of the test attribute is just an XPath expression (i.e., without any comparison), then the test is satisfied if the nodelist of this XPath expression is not empty.
Example of `<xsl:if>` Element

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
    <body>
      <h2>A CD Catalog</h2>
      <table border="1">
        <tr bgcolor="yellow">
          <th>Title</th>
          <th>Artist</th>
        </tr>
        <tr>
          <td>Title</td>
          <td>Artist</td>
        </tr>
        <tr>
          <td>Title</td>
          <td>Artist</td>
        </tr>
      </table>
    </body>
  </html>
</xsl:template>
```

As in the previous examples
Only /catalog/cd with price>10 are retrieved
# A CD Catalog

<table>
<thead>
<tr>
<th>Title</th>
<th>Artist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dark Side of the Moon</td>
<td>Pink Floyd</td>
</tr>
</tbody>
</table>
Applying Templates Recursively

• The following example shows how to apply templates recursively

• Generally, it is possible (but not in this example) that more than one template matches the current source node

• The specification (www.w3.org/TR/xslt) describes (Section 5.5) which template should be chosen for application
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/" match="/">
  <html>
    <body>
      <h2>A CD Catalog</h2>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
<xsl:template match="cd">
  <p>
    <xsl:apply-templates select="title"/>
    <xsl:apply-templates select="artist"/>
  </p>
</xsl:template>

<xsl:template match="title">
  Title: <span style="color:red"><xsl:value-of select="."/></span>
  <br/>
</xsl:template>
<xsl:template match="artist">
   Artist: <span style="color:green">
   <xsl:value-of select="." />
   </span>
   <br />
</xsl:template>

</xsl:stylesheet>
A CD Catalog

Title: Dark Side of the Moon
Artist: Pink Floyd

Title: Space Oddity
Artist: David Bowie

Title: Aretha: Lady Soul
Artist: Aretha Franklin
Is Recursive Application of Templates Really Needed?

- The output of the previous example can also be generated by an XSL stylesheet that uses only one template that matches the root (and does not use the element `<xsl:apply-templates>`)  

- However, some tasks can only be done by applying templates recursively  
  - This typically happens when the structure of the source XML document is not known
Recursive Template Application

- Suppose that we want to write an XSL stylesheet that generates an exact copy of the source XML document
  - It is rather easy to do it when the structure of the source XML document is known
- Can we write an XSL stylesheet that does it for every possible XML document?
  - Yes!
<?xml version="1.0"?>
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:template match="*">
    <xsl:element name="{name(.)}">
      <xsl:for-each select="@*">
        <xsl:attribute name="{name(.)}">
          <xsl:value-of select="."/>
        </xsl:attribute>
      </xsl:for-each>
      <xsl:apply-templates/>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
Summary

• XSLT is a high-level transformation language

• Create core output once in XML format (using Servlets, JSP, etc.)

• Use XSLT to transform the core output as needed