XSLT

Introduction

- XSL (eXtensible Stylesheet Language) is a styling language for XML.
- XSLT stands for XSL Transformations.
- **XSLT** is a language for transforming XML documents.
- **XPath** is a language for navigating in XML documents.
- **XQuery** is a language for querying XML documents.

XSLT – Transformation

- The root element that declares the document to be an XSL style sheet is <xsl:stylesheet> or <xsl:transform>.

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

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

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <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>
</xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
</catalog>
```

XSLT <xsl:template> Element

- An XSL style sheet consists of one or more set of rules that are called templates.
- A template contains rules to apply when a specified node is matched.
- The <xsl:template> element is used to build templates.
- The **match** attribute is used to associate a template with an XML element. The match attribute can also be used to define a template for the entire XML document. The value of the match attribute is an XPath expression (i.e. match="/" defines the whole document).

<xsl:value-of> Element

- The <xsl:value-of> element is used to extract the value of a selected node.

```
<tr>
    <td><xsl:value-of select="catalog/cd/title"/></td>
    <td><xsl:value-of select="catalog/cd/artist"/></td>
  </tr>
```

- The **select** attribute, in the example above, contains an XPath expression. An XPath expression works like navigating a file system; a forward slash (/) selects subdirectories.

<xsl:for-each> Element

- The <xsl:for-each> element allows you to do looping in XSLT.

```
<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>
```

Filtering the Output

- We can also filter the output from the XML file by adding a criterion to the select attribute in the <xsl:for-each> element.

```
<xsl:for-each select="catalog/cd[artist='Bob Dylan']">
```

- Legal filter operators are:
  - = (equal)
  - != (not equal)
  - < less than
  - > greater than

<xsl:sort> Element

- The <xsl:sort> element is used to sort the output.
- To sort the output, simply add an <xsl:sort> element inside the <xsl:for-each> element in the XSL file

```
<xsl:for-each select="catalog/cd">
    <xsl:sort select="artist"/>
```

- The **select** attribute indicates what XML element to sort on.

<xsl:if> Element

- The <xsl:if> element is used to put a conditional test against the content of the XML file.
- To put a conditional if test against the content of the XML file, add an <xsl:if> element to the XSL document.

```
<xsl:if test="expression">
  ...some output if the expression is true...
</xsl:if>
```

<xsl:if> Element

- To add a conditional test, add the <xsl:if> element inside the <xsl:for-each> element in the XSL file

```
<xsl:if test="price &gt; 10">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
        <td><xsl:value-of select="price"/></td>
      </tr>
    </xsl:if>
```

- The value of the required **test** attribute contains the expression to be evaluated.

<xsl:choose> Element

- The <xsl:choose> element is used in conjunction with <xsl:when> and <xsl:otherwise> to express multiple conditional tests.

```
<xsl:choose>
  <xsl:when test="expression">
    ... some output ...
  </xsl:when>
  <xsl:otherwise>
    ... some output ....
  </xsl:otherwise>
</xsl:choose>
```

- To insert a multiple conditional test against the XML file, add the <xsl:choose>, <xsl:when>, and <xsl:otherwise> elements to the XSL file

<xsl:choose> Element

```
<xsl:choose>
    <xsl:when test="price &gt; 10">
      <td bgcolor="#ff00ff">
      <xsl:value-of select="artist"/></td>
    </xsl:when>
    <xsl:otherwise>
      <td><xsl:value-of select="artist"/></td>
    </xsl:otherwise>
  </xsl:choose>
```

<xsl:choose> Element

```
<xsl:choose>
    <xsl:when test="price &gt; 10">
      <td bgcolor="#ff00ff">
      <xsl:value-of select="artist"/></td>
    </xsl:when>
    <xsl:when test="price &gt; 9">
      <td bgcolor="#cccccc">
      <xsl:value-of select="artist"/></td>
    </xsl:when>
    <xsl:otherwise>
      <td><xsl:value-of select="artist"/></td>
    </xsl:otherwise>
  </xsl:choose>
```