How to render the content of an XML document on a page?

Two mechanisms:

- CSS: Cascading Style Sheets
- XSL (the eXtensible Style sheet Language)
CSS

Definitions:

- CSS: Cascading Style Sheets
- Simple syntax
- It is a rule-based style sheet language.

CSS Rule Syntax:

- The CSS syntax is made up of three parts:
  - a selector: the element/tag you wish to define.
  - A declaration:
    - a property: of the element/tag
    - a value: to be assigned to the property.

- General syntax:

```
selector {
  property: value ;
  another_property : anothervalue;
  ...
}
```
• Example:

```css
p {
    text-align: center;
    color: black;
    font-family: arial
}
```

**Grouping**

• In order to decrease repetitious statements within style sheets

• **Group** selectors and declarations.

• Example: All of the headings in a document could be given identical declarations for all heading in a document:

```css
H1, H2, H3, H4, H5, H6 {
    color:red;
    Text-align:left;
    font-family: sans-serif
}
```

• An HTML example:

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
    <title> Introduction to XML </title>
    <style type="text/css">
        body {color: Blue; margin: 6em ;}
    </style>
</head>
</html>
```
Comments

- Similar to comments in C.

- An example:

/* Comments cannot be nested */
CSS for XML document

Objectives:

• XML relies more on style sheets than HTML.

• Help XML authors display and present their own elements for XML readers.

How to display elements?

• Elements can be displayed
  ▪ inline: do not cause line breaks
  ▪ block: cause line breaks.

• Set style sheet properties for elements.

• Example:
  course_catalog_embedded_css.xml

Style Sheet Properties:

CSS Background
CSS Text
CSS Font
CSS Border
CSS Margin
CSS Padding
CSS List
CSS Dimension
CSS Classification
CSS Positioning
CSS Pseudo-class
CSS Pseudo-element
CSS Media Types
How to display an XML document using a CSS?

- Linked:
  
  ✓ Use PI right after the XML declaration.

  ```xml
  <?xml version="1.0"?>
  <?xml-stylesheet type="text/css" href="course_catalog.css"?>
  ...
  ...
  ```

  ✓ Example: Course_catalog_linked_css.xml

- Embedded

  ```xml
  <?xml version="1.0"?>
  <?xml-stylesheet type="text/css" ?>
  <your_start_tag xmlns:HTML="http://www.w3.org/Profiles/XHTML-transitional">
    <HTML:STYLE>
      ...
    </HTML:STYLE>
    ...
  </your_start_tag>
  ```

  ✓ Example: Course_catalog_embedded_css.xml
**XSL**

**Definitions**

- It is the preferred style sheet language of XML
- It is much more sophisticated than the CSS.
- XSL is a standard recommended by the World Wide Web Consortium.
- Requirement: In order to process an XML document using XSL, you need *an XML parser with an XSL Engine*. Example: Internet Explorer 5.0.

- XSL is constituted of three main components (www.w3.org):
  - A transformation language known as XSLT
  - An expression language for addressing parts of XML documents, known as Xpath
  - A vocabulary of formatting objects with their associated formatting properties, known as XSL-FO.

**XSL Declaration:**

- W3C XSL Recommendation:

  `<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">`
• Internet Explorer method:
  
  ```xsl
  <xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  
  XSL uses Templates

  • XSL uses one or more templates to define how to output XML elements.

  • A match attribute is used to associate the template with an XML element.

  • Since the style sheet is an XML document itself, the document begins with an xml declaration:

    ✓ `<?xml version='1.0'?>`

    ✓ The xsl:stylesheet tag in the second line defines the start of the stylesheet.

    ✓ The xsl:template tag in the third line defines the start of a template:

      ▪ The template attribute match="/" associates (matches) the template to the root (/) of the XML source document.
The rest of the document contains the template itself. In the example below:

```html
<HTML>....</HTML>
```

defines the template into which we want to insert the data from our XML file above.

The last two lines that defines the end of the template and the end of the style sheet.

- **Example:** An XSL file.

```xml
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/"
<html>
<body>
<table border="2" bgcolor="yellow">
<tr>
<th>Title</th>
<th>Location</th>
</tr>
<xsl:for-each select="courses/course">
<tr>
<td><xsl:value-of select="title"/></td>
<td><xsl:value-of select="location"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```
XSL Elements

<xsl:value-of> Element

- It is used to select XML elements into the output stream of the XSL transformation.

- the syntax for the select attribute value is called an XSL Pattern.

- It works like navigating a file system where a forward slash (/) selects subdirectories.

- The result of the transformation will look like this:

<table>
<thead>
<tr>
<th>Title</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part I: Introduction to Oracle 8i</td>
<td>Vienna</td>
</tr>
<tr>
<td>Part II: Introduction to Oracle 8i</td>
<td>Vienna</td>
</tr>
<tr>
<td>Introduction to XML</td>
<td>Tysons</td>
</tr>
<tr>
<td>Introduction to Java</td>
<td>Tysons</td>
</tr>
</tbody>
</table>

<xsl:for-each> Element

- The XSL <xsl:for-each> element can be used to select XML elements into the output stream of the XSL transformation.
The `xsl:for-each` element locates elements in the XML document and repeats a part of the template for each one.

**Sorting: order-by attribute**

- Add this attribute to your `for-each` element in your XSL file like this:

  ```xml
  <xsl:for-each select="courses/course" order-by="+ location">
  ```

- The order-by attribute parameters:
  
  - (+) or (-) sign, to define an ascending or descending sort order
  
  - A sorting element.

- Sort Numbers:
  
  - Order-by=number(PRICE)
Example:

```xml
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <html>
      <body>
        <table border="2" bgcolor="yellow">
          <tr>
            <th>Title</th>
            <th>Location</th>
          </tr>
          <xsl:for-each select="courses/course" order-by = "+ title">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="location"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

✓ Output:

<table>
<thead>
<tr>
<th>Title</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to XML</td>
<td>Tysons</td>
</tr>
<tr>
<td>Introduction to Java</td>
<td>Tysons</td>
</tr>
<tr>
<td>Part I: Introduction to Oracle 8i</td>
<td>Vienna</td>
</tr>
<tr>
<td>Part II: Introduction to Oracle 8i</td>
<td>Vienna</td>
</tr>
</tbody>
</table>
How to filter Information?

• To filter the XML file, simply add a filter to the select attribute in your for-each element in your XSL file like this:

  <xsl:for-each select="courses/course[location='Vienna']">

• XSL filter operators are:

  =     (equal)
  !=    (not equal)
  &lt;   less than
  &gt;   greater than

• Example:

  <?xml version='1.0'?>
  <xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <html>
      <body>
        <table border="2" bgcolor="yellow">
          <tr>
            <th>Title</th>
            <th>Location</th>
            <th>Tuition</th>
          </tr>
          <xsl:for-each select="courses/course[location = 'Vienna']">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="location"/></td>
              <td><xsl:value-of select="tuition"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
<table>
<thead>
<tr>
<th>Title</th>
<th>Location</th>
<th>Tuition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part I: Introduction to Oracle 8i</td>
<td>Vienna</td>
<td>2200</td>
</tr>
<tr>
<td>Part II: Introduction to Oracle 8i</td>
<td>Vienna</td>
<td>1200</td>
</tr>
</tbody>
</table>
XSL IF Element

- To filter information from an XML document.
- Add an xsl:if element to your XSL document.
- Syntax:

  `<xsl:if test="boolean expression"> template elements </xsl:if>`
Example

```xml
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <html>
      <body>
        <table border="2" bgcolor="yellow">
          <tr>
            <th>Title</th>
            <th>Location</th>
            <th>Tuition</th>
          </tr>
          <xsl:for-each select="courses/course">
            <xsl:if match=".@[tuition &gt; 1200]">
              <tr>
                <td><xsl:value-of select="title"/></td>
                <td><xsl:value-of select="location"/></td>
                <td><xsl:value-of select="tuition"/></td>
              </tr>
            </xsl:if>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Output:

```
<table>
<thead>
<tr>
<th>Title</th>
<th>Location</th>
<th>Tuition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part I: Introduction to Oracle 8i</td>
<td>Vienna</td>
<td>2200</td>
</tr>
<tr>
<td>Introduction to Java</td>
<td>Tysons</td>
<td>1300</td>
</tr>
</tbody>
</table>
```
XSL Conditional Choose

- To filter information from an XML document.
- Add an xsl:Choose element to your XSL document.

**Syntax:**

```xml
<xsl:choose>
  <xsl:when test="boolean expression">
    template elements
  </xsl:when>
  <xsl:otherwise>
    template elements
  </xsl:otherwise>
</xsl:choose>
```

**Example**

```xml
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
<body>
<table border="2" bgcolor="yellow">
<tr>
<th>Title</th>
<th>Location</th>
<th>Tuition</th>
</tr>
<xsl:for-each select="courses/course">
<xsl:choose>
  <xsl:when test=".[tuition &gt; 1300]">
    <tr bgcolor="blue">
  </xsl:when>
</xsl:choose>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```
<table>
<thead>
<tr>
<th>Title</th>
<th>Location</th>
<th>Tuition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part I: Introduction to Oracle 8i</td>
<td>Vienna</td>
<td>2200</td>
</tr>
<tr>
<td>Part II: Introduction to Oracle 8i</td>
<td>Vienna</td>
<td>1200</td>
</tr>
<tr>
<td>Introduction to XML</td>
<td>Tysons</td>
<td>500</td>
</tr>
<tr>
<td>Introduction to Java</td>
<td>Tysons</td>
<td>1300</td>
</tr>
</tbody>
</table>

Output:
## Internet Explorer 5.0 XSLT Elements:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Description</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>xsl:apply-templates</td>
<td>Applies a template to the current order-by=&quot;+</td>
<td>-pattern&quot; select=&quot;pattern&quot;</td>
</tr>
<tr>
<td>xsl:attribute</td>
<td>Adds a new attribute to the current output element.</td>
<td>name=&quot;attribute-name&quot;</td>
</tr>
<tr>
<td>xsl:cdata</td>
<td>Adds a new CDATA section to the output.</td>
<td></td>
</tr>
<tr>
<td>xsl:choose</td>
<td>Provides a selection mechanism based on conditions.</td>
<td></td>
</tr>
<tr>
<td>xsl:comment</td>
<td>Adds a comment node to the output.</td>
<td></td>
</tr>
<tr>
<td>xsl:copy</td>
<td>Copies the current node to the output.</td>
<td></td>
</tr>
<tr>
<td>xsl:define-template-set</td>
<td>Defines a new set of templates</td>
<td></td>
</tr>
<tr>
<td>xsl:element</td>
<td>Adds a new element node to the output.</td>
<td>name=&quot;name&quot;</td>
</tr>
<tr>
<td>xsl:entity-ref</td>
<td>Adds a new entity reference node to the output.</td>
<td>name=&quot;name&quot;</td>
</tr>
<tr>
<td>xsl:eval</td>
<td>Provides an evaluation mechanism to evaluate output content.</td>
<td>language=&quot;language&quot;</td>
</tr>
<tr>
<td>xsl:for-each</td>
<td>Provides a mechanism to create a loop in the output stream.</td>
<td>select=&quot;pattern&quot; order-by=&quot;-</td>
</tr>
<tr>
<td>xsl:if</td>
<td>Provides a conditional branch mechanism based on a condition.</td>
<td>match=&quot;pattern&quot;</td>
</tr>
<tr>
<td>xsl:node-name</td>
<td>Adds the name of the current node to the output.</td>
<td></td>
</tr>
<tr>
<td>xsl:otherwise</td>
<td>Part of the choose mechanism (see xsl:choose).</td>
<td></td>
</tr>
<tr>
<td>xsl:pi</td>
<td>Adds a processing instruction to the output.</td>
<td>name=&quot;name&quot;</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
<td>Attributes</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>xsl:script</td>
<td>Defines a script area inside a template.</td>
<td>language=&quot;language&quot;</td>
</tr>
<tr>
<td>xsl:stylesheet</td>
<td>Defines the root element of the style sheet.</td>
<td>xmlns:xml=&quot;namespace&quot; language=&quot;language&quot;</td>
</tr>
<tr>
<td>xsl:template</td>
<td>Defines a template.</td>
<td>match=&quot;pattern&quot; language=&quot;language&quot;</td>
</tr>
<tr>
<td>xsl:value-of</td>
<td>Defines a node to insert into the output.</td>
<td>select=&quot;pattern&quot;</td>
</tr>
<tr>
<td>xsl:when</td>
<td>Part of the choose mechanism (see xsl:choose)</td>
<td>test=&quot;expression&quot;</td>
</tr>
</tbody>
</table>
CSS or XSL?

- CSS can be used for both HTML and XML documents
- CSS is easy to learn and implement
- You cannot sort the content of the XML file before displaying it.
- XSL is more complex than CSS
- XSL is more powerful than CSS: allow rearrangement of elements in a page.