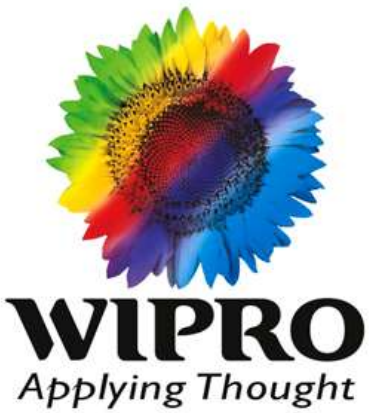


XSLT

Objectives

At the end of this module, you will be able to:

- Describe the use of XSL
- Transform an XML document by using XSLT
- Work with XPATH expressions
- Create XSL Style sheets
- Use XSLT elements
- Sort and filter XML documents

eXtensible Stylesheet Language (XSL)

Introduction to XSL

XSL stands for **Extensible Stylesheet Language**

It is an XML-based style sheet language for XML documents

XSL **describes** how the XML document should be displayed

Introduction to XSL

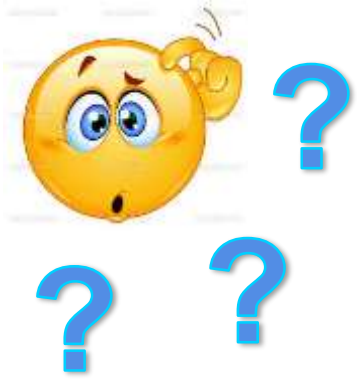
- HTML pages use predefined tags, and these tags are understood by the browsers. For example, <h1> means a heading and means bold and so on.

HTML Code	Output on the browser
Hello	Hello
I am bold	**I am bold**

- With XML, the tags are user-defined and the browsers may not understand the meaning of these tags. For example, a <table> tag could mean an HTML table or maybe a piece of furniture. Because of the nature of XML, there is no standard way to display an XML document.

Heading	Heading
Data	Data
Data	Data
Data	Data

<table>



Introduction to XSL

- In order to display XML documents, it is necessary to have a mechanism to describe how the document should be displayed. One of these mechanisms is Cascading Style Sheets (CSS) and the other one is XSL (eXtensible Stylesheet Language), which is the preferred style sheet language of XML.
- CSS was primarily designed for styling HTML pages. It can also be used to style XML pages where as XSL was designed specifically to style XML pages and is much more sophisticated than CSS

Introduction to XSL

- **The XSL consists of three languages**

XSL Transformations (XSLT): an XML language for transforming XML documents into other types of documents (for eg. HTML)

XML Path Language (XPath): a non-XML language used by XSLT, and also for selecting parts of an XML document

XSL Formatting Objects (XSL-FO): an XML language for specifying visual formatting of an XML document

XSLT

- An important part of XSL
- A W3C Recommendation
- Uses XPath to navigate in XML documents
- Transforms XML documents into any text-based format
  - **HTML, plain text, rich text format (RTF), and Microsoft Word**
  - **The most common transformation is from XML documents to HTML documents**
- Uses two documents for transformation
  - **an XML document containing actual data**
  - **an XSL document (a style sheet with a .xsl extension)**

XSLT

- XSLT has become the most popular part of XSL because it's relatively easy to use and it lets you transform XML documents into other formats, such as HTML or plain text.
- XSLT is popular partly because by using it, you can manipulate the data in XML documents without having to write any software.
- Note that XSLT style sheets are also XML documents.
- XSL is a more general language that lets you format XML in great detail.

Transforming XML by using XSLT

XSLT transformations can happen at three different

places:

- **In the server** - A server program, such as a .NET or JSP can use XSLT to transform XML document and send it to client
- **In the client** - A client program, such as browser can perform XSLT transformations
  - Modern web browsers include an XSLT processor. So if a browser is passed an XML document with an appropriate XSL style sheet then it can transform the document to HTML and display it appropriately.
- **With a separate program** - A number of standalone programs are available to perform XSLT transformations

How does transformation happen?

- The XML source document is parsed into an XML source tree
- XPath is used to define templates that *match* parts of the source tree
- XSLT is used to ***transform*** the matched part and put the transformed information into a result tree
- The result tree is output as a result document
- Parts of the source document that are not matched by the template are copied unchanged
- XSLT uses XPath to find information in an XML document. XPath is used to navigate through elements and attributes in XML documents.
- / is the root node of the XML tree that represents the start of the document . That is it does not refer to any specific element.

Understanding XPath

- Consider the XML document:

movies.xml

```
<movieLibrary>
  <movie>
    <title>Mad Max</title>
    <director>George Miller</director>
  </movie>
  <movie>
    <title>Padosan</title>
    <director>Jyoti Swaroop</director>
  </movie>
</movieLibrary>
```

XPath expressions are similar to paths in a computer file system

/ represents the document

/movieLibrary selects root element

/movieLibrary/movie selects *every* movie element

//director selects *every* actor element, wherever it occurs

Understanding XSLT

- **<xsl:for-each select="//movie">**
  - loops through every movie element, everywhere in the document

- **<xsl:value-of select="title"/>**
  - selects *content* of title element at current location

- **<xsl:for-each select="//movie">**
  <xsl:value-of select="title"/>
  </xsl:for-each>
  - selects content of title element for each movie in the XML document

Steps for creating documents

1. Create an XML document movies.xml

2. Create an XSL style sheet file movies.xsl that describes how to select elements from movies.xml and embed them into an HTML page
   a. This is done by intermixing HTML and XSL in movies.xsl file

3. Add the following line to movies.xml file to tell it to connect to the style sheet movies.xsl for formatting information

```
<?xml-stylesheet type="text/xsl" href="movies.xsl"?>
```

Outline of the required XSL file

- An XSLT document has the .xsl extension

```
<?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> ... </html>
    </xsl:template>
</xsl:stylesheet>
```

- Begins with xml declaration followed by <xsl:stylesheet> element
- Contains one or more templates, such as:

  <xsl:template match="/"> ... </xsl:template>

- And ends with </xsl:stylesheet>

XSL Style Sheet - Example

movies.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html"/>
  <xsl:template match="/">
    <html>
      <body>
        <h2>Movie Collection</h2>
        <table border="1">
        <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Director</th></tr>
        <xsl:for-each select="movieLibrary/movie">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="director"/></td></tr>
        </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Output

Movie Collection

Title	Director
Mad Max	George Miller
Padosan	Jyoti Swaroop

Observe here that XSL can rearrange the data; the HTML result can present information in a different order than the XML

XSLT <xsl:template> Element

- An XSL style sheet consists of one or more set of rules called templates
- A template lets you match a node/nodes in the XML document
- It allows to specify what you want to do with the contained data
- The <xsl:template> element is used to build templates
- The match attribute associates a template with an XML element
- It also defines a template for the entire XML document
- Value of match attribute is an XPath expression
- Examples:

  <xsl:template match="/"> …. </xsl:template>

  <xsl:template match=“movieLibrary">

A sample XML document

```
<?xml version="1.0" encoding="ISO8859-1" ?>
<CATALOG>
 <CD>
  <TITLE>Titanic</TITLE>
  <ARTIST>Leonardo di Caprio</ARTIST>
        <PRICE>Rs.300</PRICE>
  <YEAR>1997</YEAR>
 </CD>
 <CD>
  <TITLE>Rab Ne Bana di Jodi</TITLE>
  <ARTIST>Shahrukh Khan</ARTIST>
   <PRICE>Rs.75</PRICE>
  <YEAR>2008</YEAR>
 </CD>
 <CD>
  <TITLE>Ghajini</TITLE>
  <ARTIST>Amir Khan</ARTIST>
   <PRICE>Rs.200</PRICE>
  <YEAR>2008</YEAR>
 </CD>
```

catalog.xml

```
 <CD>
  <TITLE>Slumdog Millionaire</TITLE>
  <ARTIST>Dev Patel</ARTIST>
   <PRICE>Rs.100</PRICE>
  <YEAR>2009</YEAR>
 </CD>
 <CD>
  <TITLE>Mungaru Male</TITLE>
  <ARTIST>Ganesh</ARTIST>
   <PRICE>Rs.175</PRICE>
  <YEAR>2006</YEAR>
 </CD>
 <CD>
  <TITLE>Jab We Met</TITLE>
  <ARTIST>Shahid Kapoor</ARTIST>
  <PRICE>Rs.50</PRICE>
  <YEAR>2007</YEAR>
 </CD>
</CATALOG>
```

Sorting with XSL

- Use xsl:sort and order attribute for sorting XML data

- For example:

```
<xsl:for-each select="CATALOG/CD" >
        <xsl:sort select="ARTIST" order="descending"/>
```

sorts elements in descending order of ARTIST values

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>Artist</th>
      </tr>
      <xsl:for-each select="CATALOG/CD" >
         <xsl:sort select="ARTIST" order="descending"/>
      <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>
```

Filtering Output

- You can filter output by introducing filtering parameter to the select value
- For example:

```
<xsl:for-each select="CATALOG/CD[PRICE>150]">
```

  This will select only those CDs where PRICE is greater than 150

- Compound filtering parameters is also possible
- For example:

```
<xsl:for-each select="CATALOG/CD[YEAR>2005 and
  PRICE>150]">
```

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>Artist</th>
      </tr>
      <xsl:for-each select="CATALOG/CD[YEAR>2005 and PRICE>150]">
      <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>
```

if condition

- Use *if* condition for matching an element's value and obtain selective display based on the result of if condition

```
<xsl:for-each select="CATALOG/CD">
 <xsl:if test="PRICE > 160'">
  <tr>
        <td><xsl:value-of select="TITLE"/></td>
      <td><xsl:value-of select="ARTIST"/></td>
   </tr>
  </xsl:if>
</xsl:for-each>
```

- Here, the values of TITLE and ARTIST are displayed based on result of xsl:if match

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>Artist</th></tr>
```

```
    <xsl:for-each select="CATALOG/CD">
     <xsl:if match="PRICE > 160">
       <tr>
      <td><xsl:value-of select="TITLE"/></td>
    <td><xsl:value-of select="ARTIST"/></td>
       </tr>
     </xsl:if>
    </xsl:for-each>
   </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Choose, When and Otherwise

- The xsl:choose ... xsl:when ... xsl:otherwise construct is XML's equivalent of Java's switch ... case ... default statement

```
<xsl:choose>
    <xsl:when test="PRICE >100">
        <td bgcolor="#ff0000"><xsl:value-of select="ARTIST"/></td>
    </xsl:when>
     <xsl:otherwise>
        <td><xsl:value-of select="ARTIST"/></td>
    </xsl:otherwise>
</xsl:choose>
```

- This code checks whether PRICE is greater than 150 and if true displays ARTIST with a red background. Otherwise, background color remains unchanged

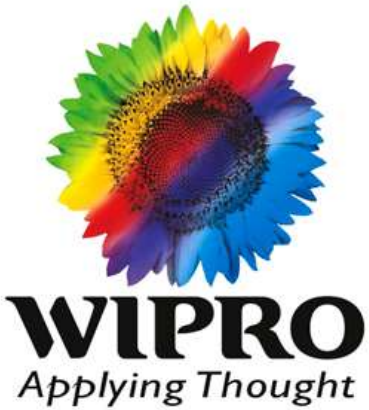
Example

```
<?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>Artist</th>
      </tr>
      <xsl:for-each
select="CATALOG/CD">
       <tr>
        <td><xsl:value-of
select="TITLE"/></td>
        <xsl:choose>
          <xsl:when test="PRICE > 150">
           <td bgcolor="#ff0000"><xsl:value-of
select="ARTIST"/></td>
          </xsl:when>
          <xsl:otherwise>
           <td><xsl:value-of
select="ARTIST"/></td>
          </xsl:otherwise>
        </xsl:choose>
       </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Summary

In this module, you were able to

- Describe the use of XSL
- Transform an XML document by using XSLT
- Work with XPATH expressions
- Create XSL Style sheets
- Use XSLT elements
- Sort and filter XML documents



Thank You