

Lesson #15:

XSL / XSLT

XSL and XSLT

- XSL stands for EXtensible Stylesheet Language, and is a style sheet language for XML documents.
- XSLT stands for XSL Transformations. XSLT is used to transform XML documents into other formats, like XHTML.
- The World Wide Web Consortium (W3C) started to develop XSL because there was a need for an XML-based Stylesheet Language.

CSS vs. XSL

- CSS = Style Sheets for HTML
- HTML uses predefined tags, and the meaning of each tag is well understood.
- The `<table>` tag in HTML defines a table, and a browser knows how to display it.
- Adding styles to HTML elements are simple. Telling a browser to display an element in a special font or color, is easy with CSS.

CSS vs. XSL

- XSL = Style Sheets for XML
- XML does not use predefined tags (we can use any tag-names we like), and therefore the meaning of each tag is not well understood.
- A `<table>` tag could mean an HTML table, a piece of furniture, or something else, and a browser does not know how to display it.
- XSL describes how the XML document should be displayed.

XSL in three parts

- XSLT: a language for transforming XML documents
- XPath: a language for navigating in XML documents
- XSL-FO: a language for formatting XML documents

XPath

- Syntax for defining parts of an XML document.
- Uses path expressions to navigate in XML documents.
- Contains a library of standard functions.
- Seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document nodes.
- XML documents are treated as trees of nodes. The topmost element of the tree is called the root element.

XPath

Root element node

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<bookstore>  
  <book>  
    <title lang="en">Harry Potter</title>  
    <author>J K. Rowling</author>  
    <year>2005</year>  
    <price>29.99</price>  
  </book>  
</bookstore>
```

attribute node

element node

Xpath path expressions

- Nodes in XPath can be parents, children, siblings, ancestors and descendants.
- XPath uses path expressions to select nodes or node-sets in an XML document. The node is selected by following a path or steps.

Expression	Description
<i>nodename</i>	Selects all child nodes of the named node
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Xpath path expressions

- Some examples:

Path Expression	Result
bookstore	Selects all the child nodes of the bookstore element
/bookstore	Selects the root element bookstore Note: If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

Xpath predicates

- Predicates are used to find a specific node or a node that contains a specific value.
- Predicates are always embedded in square brackets.

Path Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element. Note: IE5 and later has implemented that [0] should be the first node, but according to the W3C standard it should have been [1]!!
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='eng']	Selects all the title elements that have an attribute named lang with a value of 'eng'
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

Xpath wildcards

- Wildcards are used to select unknown XML elements.

Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

Path Expression	Result
/bookstore/*	Selects all the child nodes of the bookstore element
//*	Selects all elements in the document
//title[@*]	Selects all title elements which have any attribute

Xpath (several paths)

- By using the | operator in an XPath expression you can select several paths.

Path Expression	Result
<code>//book/title //book/price</code>	Selects all the title AND price elements of all book elements
<code>//title //price</code>	Selects all the title AND price elements in the document
<code>/bookstore/book/title //price</code>	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document

Xpath axes

- An axis defines a node-set relative to the current node.
- A location path can be absolute or relative.
- An absolute location path starts with a slash (/) and a relative location path does not. In both cases the location path consists of one or more steps, each separated by a slash.
- A step consists of an axis, a node-test and zero or more predicates.
`axisname::nodetest[predicate]`

Xpath axes examples

- A few examples.

Example	Result
<code>child::book</code>	Selects all book nodes that are children of the current node
<code>attribute::lang</code>	Selects the lang attribute of the current node
<code>child::*</code>	Selects all children of the current node
<code>attribute::*</code>	Selects all attributes of the current node
<code>child::text()</code>	Selects all text child nodes of the current node
<code>child::node()</code>	Selects all child nodes of the current node
<code>descendant::book</code>	Selects all book descendants of the current node
<code>ancestor::book</code>	Selects all book ancestors of the current node
<code>ancestor-or-self::book</code>	Selects all book ancestors of the current node - and the current as well if it is a book node
<code>child::*/*/child::price</code>	Selects all price grandchildren of the current node

Xpath operators

- An XPath expression returns either a node-set, a string, a Boolean, or a number.

Operator	Description	Example	Return value
	Computes two node-sets	//book //cd	Returns a node-set with all book and cd elements
+	Addition	6 + 4	10
-	Subtraction	6 - 4	2
*	Multiplication	6 * 4	24
div	Division	8 div 4	2
=	Equal	price=9.80	true if price is 9.80 false if price is 9.90
!=	Not equal	price!=9.80	true if price is 9.90 false if price is 9.80
<	Less than	price<9.80	true if price is 9.00 false if price is 9.80
<=	Less than or equal to	price<=9.80	true if price is 9.00 false if price is 9.90
>	Greater than	price>9.80	true if price is 9.90 false if price is 9.80
>=	Greater than or equal to	price>=9.80	true if price is 9.90 false if price is 9.70
or	or	price=9.80 or price=9.70	true if price is 9.80 false if price is 9.50
and	and	price>9.00 and price<9.90	true if price is 9.80 false if price is 8.50
mod	Modulus (division remainder)	5 mod 2	1

What is XSLT?

- XSL Transformations
- The most important part of XSL.
- Used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.
- All major browsers support XSLT.

Starting XSLT

- How to transform XML into XHTML using XSLT?
- The root element that declares the document to be an XSL style sheet is `<xsl:stylesheet>` or `<xsl:transform>`

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

Transformation Example

- We want to transform the following XML document ("CdCatalog.xml") into XHTML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<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>
```

Transformation Example

- Then you create an XSL Style Sheet ("cdcatalog.xsl") with a transformation template.
- Add the following line in you xml file:

```
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
```
- The <xsl:template> element defines a template. The match="/" attribute associates the template with the root of the XML source document.

No data copied yet!



Title	Artist
-	-

Transformation Example

- The `<xsl:value-of>` element can be used to extract the value of an XML element and add it to the output stream of the transformation. The value of the `select` attribute is an XPath expression.

```
<xsl:value-of select="Catalog/cd/title"/>  
<xsl:value-of select="Catalog/cd/artist"/>
```

Only one line of data copied

Title	Artist
Empire Burlesque	Bob Dylan

Transformation Example

- The XSL `<xsl:for-each>` element can be used to select every XML element of a specified node-set.
- `<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>`

The output is complete

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr. Hook
Maggie Ma	Rod Stewart
Romanza	Andrea Bocelli

Transformation Example

- 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']">`
- To sort the output, simply add an `<xsl:sort>` element inside the `<xsl:for-each>` element in the XSL file.
- `<xsl:sort select="artist"/>`

Transformation Example

- 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="price > 10">`
- To insert a multiple conditional test against the XML file, add the `<xsl:choose>`, `<xsl:when>`, and `<xsl:otherwise>` elements to the XSL file.

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli

```
<tr>
  <td><xsl:value-of select="title"/></td>
  <xsl:choose>
    <xsl:when test="price > 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>
  </tr>
```

Transformation Example

- The `<xsl:apply-templates>` element applies a template to the current element or to the current element's child nodes.

```
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>

<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:#ff0000">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>

<xsl:template match="artist">
  Artist: <span style="color:#00ff00">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>
```

Title: **Empire Burlesque**
Artist: **Bob Dylan**

Title: **Hide your heart**
Artist: **Bonnie Tyler**

Title: **Greatest Hits**
Artist: **Dolly Parton**

End of lesson