Methods and Tools for Management Information Systems

Lecture 7

18. Januar 2010
**XSLT**

- XSLT (Extensible Stylesheet Language Transformations) is a standardised, declarative transformation language to describe and control the transformation of XML documents into other formats, such as HTML, XHTML or text via an XSLT processor.

- The XSLT processor converts a source XML document into a result document using an XSL stylesheet document (a sort of template) ⇒ the XSL document is written in XML too.
Transformation

XML
Source

Transformation

XSLT
Stylesheet

XSLT-Processor

Result
XSLT Processors

- MS IE 6.0 or higher
  - Includes one
- Oxygen XML editor and debugger for Eclipse
  - http://www.oxygenxml.com/
- Apache Cocoon (including Xalan XSLT processor)
  - http://cocoon.apache.org/
- PHP / Sablotron XSLT support
  - http://at.php.net/xslt
- Saxon XSLT / XQuery processor
  - http://saxon.sourceforge.net/
Resources

- XSLT reference
  - http://www.w3schools.com/xsl/xsl_w3celementref.asp

- Specifications
  - http://www.w3.org/Style/XSL/
  - http://www.w3.org/TR/xslt20/

- Literature
Flexibility

- The content of the XML document can be re-structured
- Elements, attributes, processing directives, namespaces and comments can be accessed
- The data can be filtered or sorted
- Variables and loops can be used
- CSS formatting is available when transforming to HTML / XHTML
- Because of many functions XSL is a very complex language
Integrating XML and XSL

- Remember the processing directive used for integrating CSS files into XML documents:

  ```xml
  <?xml-stylesheet type="text/css" href="filename"?>
  ```

- For XSL files a similar processing directive is set within the XML file:

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

- When declaring several XSL stylesheets only the first one is used
- When declaring XSL and CSS stylesheets only the XSL stylesheet is used
The components of an XML document are then represented in XSLT by a tree structure (similar to the DOM’s node structure); the whole XML document has its equivalent in the XSLT root node.
**XPath**

- For navigation within the tree and accessing the included information XPath (XML Path Language) is used.
- XPath is a path description language for XML documents and a derived subset of XQuery.
  - [http://www.w3.org/TR/xpath](http://www.w3.org/TR/xpath)
  - [http://www.w3.org/TR/xquery](http://www.w3.org/TR/xquery)
- XSLT 1.0 only works with XPath 1.0; XSLT 2.0 only works with XPath 2.0.
- XPath is mainly used within XSLT.
**XSL Stylesheet Structure (I)**

- **File structure**

- ```xml
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <!-- (corresponds to the XSLT root node) -->
  ... content ...
</xsl:template>
</xsl:stylesheet>
```
**XSL Stylesheet Structure (II)**

- Content structure (transforming to HTML / XHTML)
  - `<html>
    `<head>
        
        ....
    `
    `<body`
    `<span optional_CSS_instructions>optional_content`
    `<xsl:value-of select="element_name"/>
    `
    
    ....
    
    </body`
    `</html>`
Example – XML Document

XML document

- <library>
  - <book>
    - <author>Mark Twain</author>
    - <title>Huckleberry Finn</title>
    - <pages>334</pages>
  - ....
- </library>
Example – XSL Stylesheet

- XSL document
</span><br />
  - `<span>Title: <xsl:value-of select="library/book/title"/>
</span>

- The element path must start with the template match node (unless the XSLT processor already moved to another node (the so-called context node) while processing the stylesheet)
Loops (I)

- To show all elements of a data set
  - `<xsl:for-each select="library/book" >
    <span>Author: <xsl:value-of select="author"/>
    </span><br />
    <span>Title: <xsl:value-of select="title"/>
    </span><br />
    <span>Pages: <xsl:value-of select="pages"/>
    </span>
  </xsl:for-each>

- Now the actual node (context node) is `library/book`, selected in the `xsl:for-each-statement`, therefore only the path from that node onwards has to be specified.
**Loops (II)**

Loops can also be created by defining more than one template in a stylesheet

```
<xsl:template match="/">
  <body>
    <xsl:apply-templates select="library/book"/>
  </body>
</xsl:template>
<xsl:template match="book">
  <span>Author: <xsl:value-of select="author"/></span><br />
  <span>Title: <xsl:value-of select="title"/></span>
</xsl:template>
```

Author: <xsl:value-of select="author"/>
<br />
Title: <xsl:value-of select="title"/>
### Select- & Match-Terms (I)

<table>
<thead>
<tr>
<th>Path</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>the element with the specified name</td>
<td>book</td>
</tr>
<tr>
<td>/ (within a path)</td>
<td>separates the levels of a path</td>
<td>book/title</td>
</tr>
<tr>
<td>/ (at the beginning of a path)</td>
<td>the XSLT root node</td>
<td>/library</td>
</tr>
</tbody>
</table>
**Select- & Match-Terms (II)**

<table>
<thead>
<tr>
<th>Path</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>//</td>
<td>recursion; the following term means all subordinated elements on any level</td>
<td>library//author (means all authors in the library)</td>
</tr>
<tr>
<td>. (select only)</td>
<td>the actual context node</td>
<td>&lt;xsl:value-of select=&quot;.&quot; /&gt; (returns the context node)</td>
</tr>
<tr>
<td>.. (select only)</td>
<td>the superordinated node to a context node</td>
<td>..//author (each author-element on the same level as</td>
</tr>
</tbody>
</table>
## Select-& Match-Terms (II)

<table>
<thead>
<tr>
<th>Path</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>each element</td>
<td>book/* (each element subordinated to book)</td>
</tr>
<tr>
<td>@name</td>
<td>the attribute with the specified name</td>
<td>book/@available (each attribute with the specified name belonging to a book element)</td>
</tr>
<tr>
<td></td>
<td>combines more than one path in a path</td>
<td>*</td>
</tr>
</tbody>
</table>
Functions

■ A function is an XSLT module that performs a task and then returns a value
  ■ `<xsl:value-of select="sum(library/books/pages)"/>

■ If at least one of the corresponding nodes returns a value not being a number the function returns ”NaN” meaning ”not a number”

■ For a list of all available functions see:
  ■ http://www.w3.org/TR/xquery-operators/
Filtering (I)

- A filter defines a condition to narrow down the number of selected nodes
  - `<xsl:for-each select="library/book[author='Mark Twain']">`
    - `<span>Title: <xsl:value-of select="title"/>`<br>
      - `<span>`</span>`
    - `<xsl:for-each>`

- What about this one?
  - `<xsl:apply-templates select="library/book[author='Mark Twain']"/>
    - `<xsl:template match="book">`
      - `<span>Title: <xsl:value-of select="title"/>`<br>
        - `<span>`</span>`
    - `<xsl:template>`
Filtering (II)

- And what about this one?
  
  - <xsl:apply-templates select="library/book"/>
  
    ... 
    
    <xsl:template match="book[author='Mark Twain']">
    
    <span>Title: <xsl:value-of select="title"/></span>
    
    </xsl:template>
### Operators To Compare

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>=</code></td>
<td>is</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>is not</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>smaller than</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>smaller than or same as</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>bigger than</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>bigger than or same as</td>
</tr>
</tbody>
</table>

- Remember: `&lt;` and `&lt;=` must be used as `<` is not a valid character within an attribute’s value.
Special Filterings

- If there is more than one subordinated element with the same name
  - `<xsl:for-each select="catalog/trousers[colour[2] = 'blue']">`

- If all subordinated elements of a specific element are to be selected
  - `<xsl:for-each select="library/book[5]/*">`

- A set of subelements may only be selected if it includes a certain subelement
  - `<xsl:for-each select="library/book[pages]">`
Sorting

- Controls the order of the nodes
  
  ```
  <xsl:for-each select="library/book">
    <xsl:sort select="author" data-type="text" order="ascending"/>
    <xsl:sort select="title" data-type="text" order="ascending"/>
    <span>Author (alphabetically): <xsl:value-of select="author"/></span>
    <span>Title: <xsl:value-of select="title"/></span>
  </xsl:for-each>
  
  Values for data-type: text and number
  
  Values for order: ascending and descending
Accessing Attributes

- Displaying a specific attribute of an element
  - `<xsl:value-of select="element/@attribute_name"/>`
- Displaying all attributes of an element
  - `<xsl:value-of select="element/@*"/>`
- Filtering using an attribute (without its value)
  - `<xsl:for-each select="element[@attribute_name]">`
- Filtering using an attribute (with its value)
  - `<xsl:for-each select="element[@attribute_name='value']">`
Conditions (1)

- If-condition

- `<xsl:for-each select="library/book">
  
  <span>
    <xsl:value-of select="title"/>
    <xsl:if test="@available='no'">Not available!</xsl:if>
  </span>
  
  <br />
</xsl:for-each>`
Conditions (II)

- Choose-condition

```xml
<xsl:for-each select="library/book">
  <span>
    <xsl:choose>
      <xsl:when test="pages &lt;=300">*</xsl:when>
      <xsl:when test="pages &lt;=500">**</xsl:when>
      <xsl:otherwise>***</xsl:otherwise>
    </xsl:choose>
    <xsl:value-of select="title"/>
  </span>
  <br />
</xsl:for-each>
```