

Functions with XSLT

Simon Mahony

From an original document by Susan Hockey

**This document is part of a collection of presentations and exercises on XML. For full details of this and the rest of the collection see the cover sheet at:
<http://ucloer.eprints-hosting.org/id/eprint/19>**

Applying a Template

- Process a set of nodes by whatever templates are appropriate
- Start from the current node
- Recursion
 - Apply all templates within the template
 - Drill down the tree as far as you can go
 - Back up to current node when recursion has finished

Templates

- Stylesheet is a set of templates

```
<xsl:stylesheet>
```

```
<xsl:template></xsl:template>
```

```
<xsl:template></xsl:template>
```

```
</xsl:stylesheet>
```

- Each template consists of two parts:

- a path (how to find the bit you want)

- content (what you want to do with it)

- Use the path to find an element – using XPath language

- The content is what is done to the element as it is transferred to the result document

Current or Context Node

- By default, the node that is specified by the template currently being processed
- Writing the path from the root disregards the current node
- . (full stop/period) refers to the current node

XSLT Nodes

- Node is an individual piece of the XML document
- Root: the document itself - independent of any content
- Element: each element in the XML document
- Attribute: each attribute in the XML document
- Text: text content of an element
- Comment: comment in the XML doc
- Processing instruction: instructions in XML doc

Selecting Attributes

- To select the attributes of a node
- Can output the value of an element and the value of its attributes

/@attributename

`<xsl:value-of select="book/@year"/>`

for

`<book year="1976">Title of book</book>`

would output **1976**

Selecting Attributes Example

```
<p>  
<xsl:value-of select="book"/>  
was published in  
<xsl:value-of select="book/@year"/>  
</p>
```

```
<book year="2003">Eats, Shoots and Leaves</book>
```

would output

```
<p>Eats, Shoots and Leaves was published in  
2003</p>
```

Selecting Attributes by Value

Use 'predicates' to select based on values
Predicate values are enclosed thus: [...]
(ie applying conditions)

```
<xsl:for-each select="memo[@status='keep']">
```

for

```
<memo status="keep">
```

would select all the memos
with the status attribute set to *keep*

Selecting Subsets

- Use predicates (expressions) to test a condition and select a subset of the nodes based on the test
- Often used:
 - to select elements based on particular attribute values
 - to select elements based on their position in their list

Conditionals

The process only happens if a specific condition is found to be true.
(ie tests whether a Boolean condition is true or false)

`<xsl:if test=" " >` tests against the content

If condition found to be true, the processor will execute the instruction contained in the `<xsl:if>` element

`<xsl:for-each select="ingredient">`

`<xsl:if test="fooditem='red wine'">`

do something with red wine

`</xsl:if>`

`</xsl:for-each>`

Choosing Alternatives

```
<xsl:choose ...  
  <xsl:when test ....
```

```
<xsl:for-each select="ingredient">
```

```
<xsl:choose>
```

```
  <xsl:when test="fooditem='red wine'">
```

```
    do something with red wine </xsl:when>
```

```
  <xsl:when test="fooditem='beef dripping'">
```

```
    do something with beef dripping </xsl:when>
```

```
  <xsl:otherwise>
```

```
    what you do when there is no match, i.e with other fooditems
```

```
  </xsl:otherwise>
```

```
</xsl:choose>
```

```
</xsl:for-each>
```

<xsl:choose> element

- <xsl:choose> element is nested immediately inside the template element
- <xsl:when> element and <xsl:otherwise> immediately nested inside <xsl:choose> element
- If the value of <xsl:when> is true then content of <xsl:when> element is output
- If value of <xsl:when> is false then the content of <xsl:otherwise> is output

When to use choose or if?

- **If** is used when there are two possible alternative variables
- **Choose** is used when there are numerous possible alternatives
- Note that the order of `<xsl:when>` is important as the processor will execute instructions in the order given
- Negative test

```
<xsl:when test="not(whateveryoudontwant)">    </xsl:when>
```

Operators for Testing Attribute Values

=	equal to (identical match including whitespace!) – be careful of trailing spaces
!=	not equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

eg: `<xsl:if test="@age <= 21" >`
`<xsl:if>`

Testing a Position

- Select a child of a node by position

`position()=n`

finds position number n

`position()=5`

finds position number 5

`position()=last()`

tests for the last position

Example of Position

```
<xsl:if test="position()=1">  
<p> This is the first one. </p>  
</xsl:if>
```

Outputs the text

```
<p> This is the first one.</p>
```

when the processor is on the first child

Sorting the output elements

Used to specify sort order

- Use `<xsl:sort>`

eg: `<xsl:sort select="fooditem"/>`

- Sorts on the contents of `<fooditem>`

- Alphabetic sort

–ascending default as above.

–descending `<xsl:sort select="." order="descending"/>`

- Numeric sort: add attribute `data-type` to `xsl:sort` and specify `number` as its value. Default is ascending.

`<xsl:sort select="." data-type="number" />`

Working with Images

- Build up an XHTML `` tag
- For XML markup

```
<image imagename="picture1.jpg" caption="picture of something or someone" />
```

Remember you need this defined in your DTD/Schema

Need to create XHTML markup

```

```

Working with Images

Incorrect

```
<img <xsl:value-of select = "image/@imagename" />
/>
```

Working with Images

- Need to access the attributes

```
<img>
```

```
  <xsl:attribute name="src">
```

```
    <xsl:value-of select="image/@imagename"/>
```

```
  </xsl:attribute>
```

```
  <xsl:attribute name="alt">
```

```
    <xsl:value-of select="image/@caption"/>
```

```
  </xsl:attribute>
```

```
</img>
```

Puts **imagename** as the value of the **src** attribute and **caption** as the **alt** attribute of ****

Adding Text

- Be wary of what happens with whitespace
- To add a space between elements use
`<xsl:text> </xsl:text>`
- `<xsl:text>` inserts whatever text is within it into the output document, in this case just a space
- If you want a fixed number of spaces you can also use ` `; its long winded but at least you know exactly how many spaces you have.

Handling Mixed Content

- Need to create more templates to drill down the tree
- Design a stylesheet which consists of small templates, often one for each element