Birkbeck
UNIVERSITY OF LONDON

# XML

Session 8

XSLT part 2

# XSLT variables

- A variable is a container of information that has a name. Whenever the variable is references in the stylesheet, the reference is replaced by the value of the variable.
- Variables are declared using the **variable** element.

  <xsl:variable name="colour">red</xsl:variable>
- Variables are identified by a leading '$' symbol
  - <h1> the colour is <xsl:value-of select='$colour'/></h1>
- Note that variables are not 'variable'. Their value cannot be changed. It can be temporarily overridden by another variable with the same name, defined within a sub-structure of the stylesheet.
  - 'global' variables defined within the stylesheet element can be replaced within a particular template element.
  - The same variable can have different values within each iteration of a for-each element loop.

# Named templates

- When the same formatting is required in a number of places, it is possible to simply reuse the same template. Instead of a match attribute, a name attribute is used to give the template a unique identifier.

```
<xsl:template name="createHeader">
   <hr />
   <h2> **** <xsl:apply-templates /> **** </h2>
   <hr/>
</xsl:template>
```

# Named templates

- Elsewhere, within other templates, the call-template element is used to activate the named template, also using a name attribute to identify the template required.

```
<xsl:template match="title>
   <xsl:call-template name="createHeader" />
</xsl:template>
<xsl:template match="head">
   <xsl:call-template name="createHeader" />
</xsl:template>
```

# Template parameters

- The named template mechanism is more useful when the action performed by the named template can be modified, by passing parameters to it that override default values.
- The **param** element defines a variable

<param name="SecurityLevel">0</param>

- The default value can be overridden by a parameter value. Parameters can be passed to a named template by use of the **with-param** element.

<with-param name="SecurityLevel">3</with-param>

# Template parameters

- The with-param element is placed within the call-template element.

```
<xsl:call-template name="createHeader">
   <xsl:with-param name="prefix">%%%</xsl:with-param>
</xsl:call-template>
<xsl:template name="createHeader">
   <xsl:param name="prefix">****</xsl:param>
   <hr/>
   <h2><xsl:value-of select="$prefix" />
   <hr/>
</xsl:template>
```

- Parameters can be passed the same way with apply-templates.

# White space

- When processing the source document, an XSLT processor creates a tree of nodes, including nodes for each text string within and between the markup tags.
  - Text nodes that contain only whitespace will by default be discarded.
- To preserve space in an element use **preserve-space**:

<xsl:preserve-space elements="pre" />

- You can use the wildcard character '*' to include all elements.
  - You can then override the instruction for a specific element using the **strip-space** xslt element.

<xsl:preserve-space elements="*" />

<xsl:strip-space elements="title para" />

# Adding extra whitespace

- When the XSLT processor reads the stylesheet, ignorable whitespace is stripped from all elements, except from within the **text** element. This element is primarily used to ensure that whitespace is not removed.
- Unicode symbols can be used to add special whitespace characters:
  - tab: **#x9** ; newline: **#xA** ; carriage return: **#xD**

```
<xsl:value-of select="@security" />
<xsl:text> </xsl:text>
<xsl:apply-templates />
```

# Output

- An XSLT transformation is typically required to write out a new document with meaningful tags (e.g. XML or HTML).
- By default, the content of each element in the source document is identified and processed, and the text content is output, but no formatting is applied to this text. The element tags from the source document are also discarded during this process.
- The most common way to write out XML elements is simply to insert the appropriate output element tags into the templates.

```
<xsl:template match="para">
    <p><xsl:value-of select="." /></p>
</xsl:template>
```

# Output modes

- You can use XSLT to create simple text files (e.g. CSV, TSV, SQL).
- You can use the output element to specify the method.

<xsl:output method="text" />

- You can also output HTML (not XHTML).

<xsl:output method="html" />

# Indentation

- The indent attribute can be used on the output element to ensure that the output data file contain indents to reveal the hierarchical structure of the document.

<xsl:output indent="yes" />

- The default value is no.

# XML/XHTML output headers

- Using the output element you can specify whether the xml declaration at the beginning of the generated file will be generated.

<xsl:output method="xml" omit-xml-declaration="yes" />

- You can also specify the DTD of the document using the output element.

<xsl:output doctype-public='...' doctype-system='...' />

- HTML 5 does not have a doctype. However W3 has provided a DOCTYPE legacy string so that XSLT engines can produce valid HTML 5.

<xsl:output doctype-system="about:legacy-compat" />