More XSLT

More on XSLT Variables

There is a third, more flexible way of defining xsl variables:

```
<xsl:variable name="numberOfPeople">
    <xsl:choose>
        <xsl:when test="count(/people/person[age>21]) >
1">too many old people</xsl:when>
        <xsl:otherwise><xsl:value-of
select="count(/people/person)"/></xsl:otherwise>
    </xsl:choose>
</xsl:variable>
```

Here the string output from running the code inside the variable tags is what's stored in the variable.

This allows us to use the full range of XSLT elements in producing the content for this variable.

Cumulative Values for XSL Variables

```
<xsl:variable name="agesOfPeople">
    <xsl:for-each select="/people/person">
        <xsl:text> </xsl:text><xsl:value-of select="age"/>
    </xsl:for-each>
</xsl:variable>
```

The `position()` Function

When a sequence of elements is being processed, this function returns the position of the current item within the sequence.

You can use this to (e.g.) not output a comma after the last element in a list:

```
[…]
<xsl:if test="position() &lt; count(/people/person)">
<xsl:text>, </xsl:text></xsl:if>
[…]
```

The `last()` Function

You can shorten this with the `last()` function, which returns the position of the last item in a list:

```
[…]
<xsl:if test="position() &lt; last()"><xsl:text>,
</xsl:text></xsl:if>
[…]
```

Applying Templates to Calculate a Variable's Value

You can use `<xsl:apply-templates>` to compute the value for a variable:

```
<xsl:variable name="agesOfPeople">
    <xsl:apply-templates select="/people/person[age > 21]"/>
</xsl:variable>

[…]

<xsl:template match="person">
    <xsl:value-of select="age"/>
    <xsl:if test="position() &lt; last()"><xsl:text>,
</xsl:text></xsl:if>
</xsl:template>
```

More Sorting

You can sort elements inside `<apply-templates>` as well as in `<for-each>` statements:

```
[…]
<xsl:apply-templates select="/people/person">
    <xsl:sort select="name">
</xsl:apply-templates>
[…]
```