

Comp 336/436 - Markup Languages

Fall Semester 2018 - Week 7

Dr Nick Hayward

XML - transforming & rendering - XSLT template

- `<template>` is the main element of a XSLT document
 - *may contain many additional directives (elements)*
- a template may be iteratively applied to elements in the XML document
 - *using different tags, e.g. `<xsl:for-each ...>`*
 - e.g.

```
<xsl:template match="chapter">
<xsl:for-each select="paragraph">
<xsl:value-of select=".">
</xsl:for-each>
</xsl:template>
```

- this example applies a template
 - *selects content of each paragraph inside a chapter*

XML - transforming & rendering - XSLT template usage

- one or more sets of rules called templates
- a template contains rules for a specified matched node
- `<xsl:template>` element is used to build templates
- `match` attribute associates a template with an XML element

```
<xsl:template match="/">
```

```
<?xml-stylesheet type="text/xsl" href="christie-full.xsl"?>
```

XML - XSLT working example - Agatha Christie

XSLT - part I

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Collection</h2>
  <table>
    <tr>
      <th>Title</th>
      <th>Author</th>
    </tr>
    <tr>
      <td>.</td>
      <td>.</td>
    </tr>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part I

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:value-of>`

- `<xsl:value-of>`
 - used to extract the value of an XML element
 - and add to transformation output
 - e.g.

```
<xsl:value-of select="catalogue/book/title"/>
<xsl:value-of select="catalogue/book/author"/>
```

- `select` attribute's value contains an XPath expression
- XPath expression navigates to the given position in the XML
- value of an attribute can be found using an expression such as

```
<xsl:value-of select="catalogue/book/title/@id"/>
```

or

```
<xsl:value-of select="catalogue/book/title/attribute::id"/>
```

XML - XSLT working example - Agatha Christie

XSLT - part 2

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Collection</h2>
  <table>
    <tr>
      <th>Title</th>
      <th>Author</th>
    </tr>
    <tr>
      <td><xsl:value-of select="catalogue/book/title"/></td>
      <td><xsl:value-of select="catalogue/book/author"/></td>
    </tr>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 2

XML - XSLT tests - initial XML

Exercise - part 2

- create an initial XSL document
 - *output heading for rendered*
 - *get value of the first two elements per item*
 - *e.g. title & author from book*
- test XSL with XML file
 - *open XML with XSL in browser and check rendering...*

10 minutes...

XML - XSLT working example - Agatha Christie

XSLT - <xsl:for-each>

- *used to select every element from a specified set*

```
<xsl:for-each select="catalogue/book">  
  <xsl:value-of select="title"/>  
  <xsl:value-of select="author"/>  
</xsl:for-each>
```


XML - XSLT working example - Agatha Christie

XSLT - part 3

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Collection</h2>
  <table>
    <tr>
      <th>Title</th>
      <th>Author</th>
    </tr>
    <xsl:for-each select="catalogue/book">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="author"/></td>
      </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 3

XML - XSLT tests - initial XML

Exercise - part 3

- add an extra *book*, *album*, &c. to your XML
- modify XSL
 - add `<xsl:for-each>` for a given parent element
 - e.g. `<xsl:for-each select="catalogue/book">`
 - use `<xsl:value-of>` to get values for the first two elements per item
- test XSL with XML file

10 minutes...

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:for-each>` filtering

```
<xsl:for-each select="catalogue/book[author='Agatha Christie']">
```

- filter XML output using additional operators, e.g.
 - `=` (equal)
 - `!=` (not equal)
 - `<` (less than)
 - `>` (greater than)

```
<xsl:for-each select="catalogue/book[author='Agatha Christie' and year='1941']">
```

XML - XSLT working example - Agatha Christie

XSLT - part 4

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Collection</h2>
  <table>
    <tr>
      <th>Title</th>
      <th>Author</th>
    </tr>
    <xsl:for-each select="catalogue/book[year='1937']">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="author"/></td>
      </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 4

XML - XSLT tests - initial XML

Exercise - part 4

- add filtering to `<xsl:for-each>`
- modify the output to render this filtered result
 - e.g. *an extra column for year*
- test XSL with XML file

10 minutes...

XML - XSLT working example - Agatha Christie

XSLT - part 5

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Collection</h2>
  <table>
    <tr>
      <th>Title</th>
      <th>Author</th>
      <th>Year</th>
    </tr>
    <xsl:for-each select="catalogue/book[year!='1937' and year<1942]">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="author"/></td>
        <td><xsl:value-of select="year"/></td>
      </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 5

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:sort>`

- sort returned data from XML

```
<xsl:sort select="title"/>
```

- we can then add the sort filter to the for-each option

```
<xsl:for-each select="catalogue/book">  
<xsl:sort select="title"/>
```

- default sort order is ascending

XML - XSLT working example - Agatha Christie

XSLT - part 6

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Collection</h2>
  <table>
    <tr>
      <th>Title</th>
      <th>Author</th>
    </tr>
    <xsl:for-each select="catalogue/book">
      <xsl:sort select="title"/>
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="author"/></td>
      </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 6

XML - XSLT tests - initial XML

Exercise - part 5

- add a sort option to `<xsl:for-each>`
- modify the output to render this sorted result
 - *add a note to rendered output for chosen sort order...*
- test XSL with XML file

10 minutes...

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:if>`

- conditional test within the template
 - *for certain conditions and requirements in the XML*

```
<xsl:if test="year > 1929">  
  ...  
</xsl:if>
```

- value of `test` attribute contains expression to be tested
 - *year elements with values greater than 1929*
 - *i.e. `year > 1929`*

XML - XSLT working example - Agatha Christie

XSLT - part 7

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Collection</h2>
  <table>
    <tr>
      <th>Title</th>
      <th>Author</th>
      <th>Year</th>
    </tr>
    <xsl:for-each select="catalogue/book">
      <xsl:sort select="title"/>
      <xsl:if test="year > 1937">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="author"/></td>
          <td><xsl:value-of select="year"/></td>
        </tr>
      </xsl:if>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 7

XML - XSLT tests - initial XML

Exercise - part 6

- add a conditional option to `<xsl:for-each>`
- modify the output to render this conditional result
 - *add a note to rendered output for chosen conditional...*
- test XSL with XML file

10 minutes...

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:choose>`

```
<xsl:choose>
  <xsl:when test="expression">
    ... some output ...
  </xsl:when>
  <xsl:otherwise>
    ... some output ....
  </xsl:otherwise>
</xsl:choose>
```

- use `<xsl:choose>` with `<xsl:when>` or `<xsl:otherwise>`
 - *test multiple conditions in XML*

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:choose>`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>Collection</h2>
  <table>
    <tr>
      <th>Title</th>
      <th>Author</th>
    </tr>
    <xsl:for-each select="catalogue/book">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <xsl:choose>
          <xsl:when test="year > 1941">
            <td class="post">After: <xsl:value-of select="author"/></td>
          </xsl:when>
          <xsl:otherwise>
            <td class="pre">Before: <xsl:value-of select="author"/></td>
          </xsl:otherwise>
        </xsl:choose>
      </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 8

XML - XSLT tests - initial XML

Exercise - part 7

- modify existing conditional option
 - use `<xsl:choose>` with `<xsl:when>` and `<xsl:otherwise>`
- modify the output to render this conditional result
 - add a note to rendered output for chosen conditional...
- test XSL with XML file

10 minutes...

XML - XSLT working example - Agatha Christie

XSLT - <xsl:apply-templates>

- add a template to a current element or its child nodes
- use a `select` attribute
 - *only process child element specified in the value*
- use a `sort` attribute
 - *specify order of child node processing*

XML - XSLT working example - Agatha Christie

XSLT - template and match

- then use standard `<xsl:template>`
 - *add `match` attribute to specify required element*
 - *add further XSL options to modify elements &c.*

e.g.

```
<xsl:template match="title">
  <xsl:choose>
    <xsl:when test="../price<10">
      <span style="color:#ff00ff">
        <xsl:value-of select="."/>
      </span>
    </xsl:when>
    <xsl:otherwise>
      <span>Price too high: <xsl:value-of select="."/></span>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
```

- Agatha Christie - XSLT - part 9

XML - XSLT working example - Agatha Christie

XSLT - `<xsl:apply-templates>`

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

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

<xsl:template match="book">
  <p>
    <xsl:apply-templates select="title"/>
    <xsl:apply-templates select="author"/>
  </p>
</xsl:template>

<xsl:template match="title">
  Title: <span class="title"><xsl:value-of select="."/></span>
  <br />
</xsl:template>

<xsl:template match="author">
  Author: <span class="author"><xsl:value-of select="."/></span>
  <br />
</xsl:template>

</xsl:stylesheet>
```

- Agatha Christie - XSLT - part 10

XML - XSLT tests - initial XML

Exercise - part 8

- modify existing XSL
- abstract XSL to use `<xsl:template>` and `<xsl:apply-templates>`
- test XSL with XML file

10 minutes...

References

- XMLNS - FOAF spec
- W3C - GRDDL
- W3C - OWL
- W3C - RDF
- W3C - SPARQL
- Xalan Project