CENTER FOR TEXTUAL STUDIES AND
DIGITAL HUMANITIES

DIGH 400 - Introduction to Digital Humanities Research

Fall Semester 2014

Week 11

Today's Class

- TEI Exercise Week 9
- Arnolfini Portrait
- www.digital-humanities.com
- SciFi Authors - Jules Verne work
- Overview of final conceptual project design
- Overview of 2nd presentations

TEI Exercise - Sample Solutions

- Sample Solutions

Arnolfini Portrait

- Sample Image
- Working with image and transcription
- TEI examples
  - Facsimile | Facsimile examples
    - Surface | Surface examples
      - Zone | Zone examples

Digital Humanities site

- Digital Humanities site updates?

Verne Digital Corpus

- Many updates
  - Timelines spreadsheet
  - Current editions including 1900 Second Edition of “An Antarctic Mystery” or "The Sphinx of the Ice Realm"
  - Anything else?

Conceptual Project

conceptual project design (40%)

- choose your own preferred material, text, work (you'll need to be able to justify your selection)

- helps visualise project management and development

- beneficial for future development and preparation of grant proposals

- does NOT require actual project development, simply conceptual planning and design

NEH guidelines overview

Project Presentation 2

25th November 2014 - ~10 mins per paper & questions from the class

- conference style short paper
- describe the work or material you have selected for the conceptual project

For example:

- briefly outline and describe the material
- describe the project and proposed final outcomes
- any possible research output
- any similar works, publications, or projects
- how the project will contribute to DH research and development...

XSL

Overview

- consists of three parts
    - XSLT: transforms XML documents
    - XPath: navigates XML documents
    - XSL-FO: formats XML documents

XSLT

- XSL Transformations
- considered most important part of XSL
- transforms XML into another XML document, or another type of document
- transforms each XML elements into an (X)HTML element for browser viewing
- uses XPath for XML navigation
- W3C recommendation
- manipulate and transform an XML document
    - add or remove elements
    - rearrange and sort elements...

XSLT - Browser Support (W3 list)

Mozilla Firefox
Firefox supports XML, XSLT, and XPath from version 3.

Internet Explorer
Internet Explorer supports XML, XSLT, and XPath from version 6.
Internet Explorer 5 is NOT compatible with the official W3C XSL Recommendation.

Google Chrome
Chrome supports XML, XSLT, and XPath from version 1.

Opera
Opera supports XML, XSLT, and XPath from version 9. Opera 8 supports only XML + CSS.

Apple Safari
Safari supports XML and XSLT from version 3.

XSLT - Process Overview

- select the XML document you want to transform into XHTML
- create an XSL style sheet with a transformation template
- link the XSL style sheet to the XML document
- XSLT compliant browser will transform XML into XHTML

Agatha Christie Example

Initial Basic XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalogue>
  <book>
    <title>Evil Under the Sun</title>
    <author>Agatha Christie</author>
    <country>UK</country>
    <publisher>Collins Crime Club</publisher>
    <price>7</price>
    <year>1941</year>
  </book>
.
.
</catalogue>
```

XSLT - <xsl:template>

- 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"?>

DIGH 400 - Introduction to Digital Humanities Research

```
<?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>
```

Example

XSLT - <xsl:value-of>

- used to extract the value of an XML element and add to transformation output

<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

- the 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"/>

DIGH 400 - Introduction to Digital Humanities Research

```
<?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>
```

Example

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>
```

DIGH 400 - Introduction to Digital Humanities Research

```
<?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>
```

Example

XSLT - <xsl:for-each> filtering

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

- filter XML output using additional operators such as

= (equal)
!= (not equal)
< (less than)
> (greater than)

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

DIGH 400 - Introduction to Digital Humanities Research

```
<?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>
```

Example 1 - Year =1937 Example 2

DIGH 400 - Introduction to Digital Humanities Research

```
<?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&lt;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>
```

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 (NB: for numbers this will always be the first number regardless of natural value - eg: 1, 10, 100, 2, 3, 31, 4...)

DIGH 400 - Introduction to Digital Humanities Research

```
<?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>
```

Example

XSLT - <xsl:if>

- conditional test within the template for certain conditions in the XML

<xsl:if test="year > 1929">

</xsl:if>

- value of 'test' attribute contains the expression to be tested

DIGH 400 - Introduction to Digital Humanities Research

```
<?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 &gt; 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>
```

[Example]

XSLT - <xsl:choose>

- we can use <xsl:choose> with <xsl:when> or <xsl:otherwise> to test multiple conditions

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

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalogue>
  <book>
    <title>Evil Under the Sun</title>
    <author>Agatha Christie</author>
    <country>UK</country>
    <publisher>Collins Crime Club</publisher>
    <price>7 shillings and sixpence</price>
    <year>1941</year>
  </book>
.
.
</catalogue>
```

DIGH 400 - Introduction to Digital Humanities Research

```
<?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 &gt; 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>
```

Example

XSLT - <xsl:apply-templates>

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

DIGH 400 - Introduction to Digital Humanities Research

```
<?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>
```

Example

```
<xsl:template match="title">
 <xsl:choose>
       <xsl:when test="../price&lt;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>
```

Example

XSLT - client side

- XML and XSL can be transformed with a browser
- javascript can also be used to perform the transformation
    - allows browser-specific testing
    - can apply different style sheets as needed
- sample javascript process may include
    - load defined XML and XSL files
    - test current browser type
    - perform specified functions relative to browser type
    - output styled document to specified container
- javascript caveat: will not work in a browser lacking XML parser
- XML parsers in PHP such as SAX parser

Example

XSLT - server side

- transform the XML to XHTML on the server
- often known as server-side processing
- provides a cross-browser solution
- many different languages include support for XSLT including PHP
- transform offline and upload to web server
- integrated development environment (IDE) for editing and transformation

Information on TEI stylesheets can be found at

http://www.tei-c.org/Tools/Stylesheets/