Comp 336/436 - Markup Languages

Fall Semester 2018 - Week 9

Dr Nick Hayward

DEV Week assessment

Course total = 25%

- project outline and introduction
- developed using a chosen markup language
- consider and apply metadata schemes and semantic organisation for chosen domain
- current working examples - what does and does not work...
- demo and project report
  - *due on Wednesday 31st October 2018 @ 4.15pm*
- anonymous peer review
  - *similar to user comments and feedback*
  - *chance to respond to feedback before final project*

DEV Week Demo

DEV week assessment will include the following:

- brief presentation or demonstration of current project work
  - *~ 5 to 10 minutes per group*
  - *analysis of work conducted so far*
  - *e.g. during semester & DEV week*
  - *presentation and demonstration*
  - *outline current state of application/project*
  - *show prototypes, designs, outlines &c.*
  - *explain what works & does not work*
    - i.e. outline what has been completed to date...
  - ...

XML - XPath details - location - select children

- use a shortcut to refer to child nodes
- instead of writing the location path from the root node
  - *reference child nodes using their name, e.g.*

```
<xsl:template match="history">
...
<xsl:value-of select="dynasty"/>
```

- `dynasty` matches a child of the `history` element
- also use standard paths to get grandchild &c.
- use `*` to select all the current node's children
- `xsl:text` element used to add literal text to output
  - *can't contain other elements*
  - *often used to add special characters, e.g. &, >*
  - *can be used to control white space...*

XML - working example - ancient sites - select children

XML

```xml
<history>
  <period>New Kingdom</period>
  <dynasty>19th</dynasty>
  <year era="BC">c. 1264</year>
</history>
```

XSL

```xml
<xsl:template match="site">
  <tr>
    <xsl:apply-templates select="name[@language='english']"/>
    <xsl:apply-templates select="history"/>
  </tr>
</xsl:template>
...
<xsl:template match="history">
  <td>
    <xsl:value-of select="year"/>
    <xsl:text>&#160;</xsl:text>
    <xsl:value-of select="year/@era"/>
  </td>
</xsl:template>
```

- Demo - Ancient Sites 3

XML - XPath & XSLT tests - select children

Exercise - part 3

- update your XSL stylesheet
  - *match required current node for parent*
  - *add template for matching child elements*
  - *combine values and text for output*
- test stylesheet with XML file

~ 10 minutes

XML - XPath details - location - select parent or siblings

- if relationship between current node and required node is clear
  - *e.g. between element nodes*
- select parent node
  - *add `..` - select current node's parent*
- select a node's siblings
  - *locate node's parent*
  - *add `/sibling` - where sibling is name of required node*
  - *add `/niece` - where niece is name of child of sibling*
  - *&c. for grandniece...*
- repeat as necessary to access multiple hierarchies...
- also get attributes from these nodes
  - e.g. `../@attribute`
- also use wildcard option within a location path
  - e.g. `../*`

XML - working example - ancient sites - select parent or siblings

XSL

```
<xsl:template match="history">
  <td>
    <xsl:value-of select="year"/>
    <xsl:text>&#160;</xsl:text>
    <xsl:value-of select="year/@era"/>
  </td>
  <td>
    <xsl:value-of select="./dynasty"/>
    <xsl:text>&#160;dynasty</xsl:text>
  </td>
</xsl:template>
```

- Demo - Ancient Sites 4

XML - XPath & XSLT tests - select parent or siblings

Exercise - part 4

- update your XSL stylesheet
  - *use current node in XSL*
  - *get value for a parent or sibling*
  - *combine values and text for output*
- test stylesheet with XML file

~ 10 minutes

XML - XPath details - location - select attributes

- `@` to specify returning an attribute
- to select a node's attributes specify the following
  - *location path to the node*
  - *add `/@` to indicate values from attributes required*
  - *add attribute name to get specific attribute on current node*
  - *or add `*` to select all attributes on current node*
- `@` sometimes referred to as *attribute axis*
- in XPath - axis is a set of nodes relative to current node
- in addition to attribute axis - 12 other axes defined in XPath, e.g.
  - *ancestor, ancestor-or-self, child, descendant, descendant-or-self, following*
  - *following-sibling, namespace, parent, preceding, preceding-sibling, and self*
- each axes specifies a *direction* relative to current node
  - *represents the corresponding node set*
  - *each axis may also be represented by a shortcut*

XML - working example - ancient sites - select attributes

XSL

```
<xsl:apply-templates select="links/overview[@type='general']"/>
...
<xsl:template match="links/overview[@type='general']">
  <td>
    <a>
      <xsl:attribute name="href">
        <xsl:value-of select="./@url"/>
      </xsl:attribute>
      <xsl:value-of select="."/>
    </a>
  </td>
</xsl:template>
```

- Demo - Ancient Sites 5

XML - XPath & XSLT tests - select attributes

Exercise - part 5

- update your XSL stylesheet
  - *select a node in your XML file*
  - *get attribute value to select another attribute value on current node*
  - *combine values and text for output*
- test stylesheet with XML file

~ 10 minutes

XML - XPath details - location - conditional selection

- create boolean expressions called *predicates*
  - *test a condition*
  - *use results of test to select specific subset of node set...*
- predicates can
  - *compare values, test existence, perform mathematics...*
- to conditionally select nodes
  - *create location path to node that contains desired subset*
  - *add [*
  - *add expression to define required subset*
  - *add]*

XML - XPath details - location - conditional selection - predicates

- predicates not only for comparisons
  - *e.g. we could use* `[@language]`
  - *selects all current node's elements with language attribute*
- also use multiple predicates to narrow search, e.g.

```
name[@language='English'][position() = last()]
```

- also add attribute selector after predicate - if required
- example XSL usage

```
<xsl:template match= "name[@language!='english']"> (<em><xsl:value-of select="."/> </em>)<
```

XML - working example - ancient sites - conditional selection

XSL

```
<xsl:apply-templates select="images"/>
...
<xsl:template match="images">
  <td>
    <xsl:value-of select="image[@type='jpg'][position() = last()]"/>
  </td>
</xsl:template>
```

- Demo - Ancient Sites 6

XML - XPath & XSLT tests - conditional selection

Exercise - part 6

- update your XSL stylesheet
  - *apply template for new parent node*
  - *add template for child node*
  - *conditionally select from child nodes using attributes*
  - *combine values and text for output*
- test stylesheet with XML file

~ 10 minutes

XML - XPath details - location - absolute paths

- create absolute location paths
  - *do not rely on the current node*
- to create an absolute location path
  - *add / - indicate starting at root node of XML document*
  - *add `root` - use root element name of your XML document*
  - *add / - down one level in XML document's tree hierarchy*
  - *add `container` - identify name of element on next level containing required element*
  - *repeat traversal to reach required depth in tree structure*
  - *add any predicates, select the node's attributes &c.*
- at any point in the location path
  - *we may also use `*` - specify all the elements at that level*

XML - XPath details - location - select all descendants

- `//` - useful to select all descendants of a particular node
- use it in either *absolute* or *relative* location path
- example usage includes
  - *all descendants of root node,*
    - `//`
  - *all descendants of current node*
    - `.//`
  - *all descendants of any node*
    - locate required node
    - `//`
  - *some descendants of any node*
    - locate required node
    - `//`
    - add name of required descendant elements
  - *output matching elements whose element name matches*
    - `//element_name` (add name of required element...)

XML - XPath details - functions - intro

- with XPath functions
  - *apply additional logic to node sets*
  - *useful option to return only the data you need...*
- e.g. perform one or more operations on a string
  - *operation performed before it is output*
  - *quickly and efficiently modify the final result*
- official specifications for XPath Version 1.0 functions
  - *https://www.w3.org/TR/xpath/#corelib*

XML - XPath details - functions - comparison

- comparison is often a common test on location paths
  - *e.g. one value greater than another...*
- use a standard conditional pattern, e.g.
  - *set path to first node set for comparison*
  - *add =, or `!=`*
  - *or add `&gt;`, `&gt;=`, `&lt;`, `&lt;=`*
  - *add value or path to a node set for comparison*
- these options can be used with `xsl:template` and `xsl:apply-templates` processing
- also use with condition testing
  - *e.g. `xsl:if` and `xsl:when`*
- use `and` operator to test a series of multiple conditions
- use `or` operator to test at least one in a series of multiple conditions

XML - working example - ancient sites - comparison

XSL

```
<xsl:apply-templates select="ancient_sites/site[./history/year &lt; 1571]">
  <xsl:sort select="year" order="descending" data-type="number" />
</xsl:apply-templates>
```

- Demo - Ancient Sites 7

XML - XPath & XSLT tests - functions - comparison

Exercise - part 7

- update your XSL stylesheet
  - *apply template for a specific node selection*
  - *add comparison against a given element for the current node*
  - *add custom sort order for output*
- test stylesheet with XML file

~ 10 minutes

XML - XPath details - functions - test position

- might also choose to select a specific node in the node set
- e.g. first, second, or even the last
- to test a node's position
  - *add `position() = n` (n = position of node in current node set)*
- also get last node in a particular node set
  - *add `last()` to get the last node*
- shortcut can be used
  - *e.g. `site[1]` would return the first `site` node*
- use this shortcut in template processing
- can't use shortcut with `xsl:if` or `xsl:when`
- can't use shortcut in `xsl:value-of` instruction

XML - XPath & XSLT tests - functions - test position

Exercise - part 8

- update your XSL stylesheet
  - *add an option to get first and last node values for a given node set*
  - *use functions to test position with a conditional statement*
    - e.g. `xsl:when`
  - *add output to rendered document*
- test stylesheet with XML file

~ 10 minutes

Demos

- XML & XSLT
  - *Ancient Sites - part 4*
  - *Ancient Sites - part 5*
  - *Ancient Sites - part 6*
- XML & XSLT - functions
  - *Ancient Sites - comparison - part 7*

References

- Oxygen XSLT Processors
- W3C - GRDDL
- W3C - OWL
- W3C - RDF
- W3C - SPARQL
- W3C - XML well formed
- Xalan Project