Comp 336/436 - Markup Languages

Fall Semester 2017 - Week 9

Dr Nick Hayward

XML - XPath details - expressions & patterns

- in XSL - patterns and expressions written using XPath (XML Path language syntax)
- XPath used to select nodes and node sets
  - *specify location paths from XML document*
- XPath includes built-in functions
  - *e.g. mathematics, process strings, conditionals &c.*
  - *many more...*

XML - XPath details - location - nodes

- nodes are the core of working with XPath
  - *use location paths to reference a node or node set*
- location path uses relationships to describe location of node or node set
  - *relative to a given node*
- XPath considers all XML documents as tree structures
- node trees
  - *hierarchical structure of nodes*
  - *XPath uses this hierarchy to navigate*

XML - working example - ancient sites

```
<?xml version="1.0" encoding="UTF-8"?>
<ancient_sites>
    <site>
        <name language="english"></name>
        <location></location>
        <country></country>
        <description></description>
        <culture></culture>
        <history></history>
        <images>
            ...
        </images>
        <notes></notes>
    </site>
</ancient_sites>
```

XML - XPath details - location - node tree - recap

- in the XML DOM, everything in an xml doc is a node, e.g.
  - *entire doc is a document node*
  - *text in xml elements are text nodes*
  - *each attribute is an attribute node*
- root node, or document node, can have any number of child nodes
- to child nodes, the root node is a parent node
- child nodes can have any number of child nodes themselves
- child nodes with the same parent are called sibling nodes
- descendant nodes are a node's child nodes, grandchild nodes &c.
- ancestor nodes are a node's parent node, grandparent nodes
- navigate and find nodes by knowing such node relationships

XML - working example - ancient sites - node tree

```
|- ancient_sites
  |-- site
      |-- name [@language]
        |-- location
        |-- country
        |-- description
        |-- culture
        |-- history
        |-- images
              |-- ...
        |-- notes
```

XML - XPath & XSLT tests - initial XML

Exercise - part 1

- choose a favourite historical or tourist site
  - *e.g. Eiffel Tower in Paris*
- markup in XML relevant and useful details of this site
- add any required metadata, notes, comments, &c.
- add a reference to a representative image of this site

~ 10 minutes

XML - XPath details - location - paths

- two kinds of location paths that we predominantly use with XPath
  - *relative and absolute location paths*
- *relative*
  - *consists of a sequence of location steps separated by `/`*
  - *each step selects a node or node set relative to the current node*
  - *e.g. `parent/child/grandchild`*
- *absolute*
  - *`/` optionally followed by a relative location path*
  - *`/` by itself selects the root node e.g. `/root/child/grandchild`*

XML - XPath details - location - using paths

- two predominant uses
  - *wrapper to find a node's location and then process child nodes*
  - *get a node's value*
- in XPath, there are seven different node types
  - *root nodes (always one)*
  - *element nodes*
  - *text nodes*
  - *attribute nodes*
  - *comment nodes*
  - *processing instruction nodes*
  - *namespace nodes*
- a way to retrieve the value of each node type
- for some nodes
  - *value is part of node*
- for other nodes
  - *value is based on value of descendant nodes*

XML - XPath details - location - current node

- as XSLT processor goes through your style sheet
  - *it works on one node at a time*
- XSLT processor knows the parts of a document to process using
  - *`xsl:template`, `xsl:apply-templates`, and `xsl:for-each` elements*
- node currently being processed is called the *current* node
  - *current node varies as processor traverses node tree*
- *context* node is the start position for an XPath location path address
- refer to current node with shortcut
  - *in a location path simply use `.` (a single period)*

XML - working example - ancient sites - current node

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="ancient-sites.xsl"?>
<ancient_sites>
    <site>
        <name language="english">Great Pyramid</name>
        <location>Giza</location>
        <country>Egypt</country>
        <description>Khufu's pyramid on the Giza plateau outside Cairo</description>
        <culture>Ancient Egypt</culture>
        <history>
      <period>Old Kingdom</period>
      <dynasty>4th</dynasty>
        </history>
        <images>
            ...
        </images>
        <notes>...</notes>
    </site>
</ancient_sites>
```

XML - working example - ancient sites - current node

XSL

```
<!-- first node match -->
<xsl:template match="/">
  <html>
    <body>
      <h3>Ancient Sites</h3>
      <!-- second match -->
      <xsl:apply-templates select="ancient_sites/site">
        <xsl:sort select="location" order="ascending" data-type="string" />
      </xsl:apply-templates>
    </body>
  </html>
</xsl:template>

<!-- third match -->
<xsl:template match="site">
  <xsl:apply-templates select="name[@lang='en']"/>
</xsl:template>

<!-- fourth match -->
<xsl:template match="name[@lang='en']">
  <xsl:value-of select="."/>
</xsl:template>
```

XML - XPath & XSLT tests - current node

Exercise - part 2

- create an XSL stylesheet for your XML document
  - *test the stylesheet work with the XML*
- add matches for root, first parent, &c.
- add sort order for your sites
- check current node throughout stylesheet
  - *i.e. reference current node in XSL*

~ 10 minutes

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

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

XSL

```
<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> </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> </xsl:text>
    <xsl:value-of select="year/@era"/>
  </td>
  <td>
    <xsl:value-of select="./dynasty"/>
    <xsl:text> 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>)</xsl:template>
```

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...)

Demos

XML & XSLT - Part 2

- Ancient Sites - part 1
- Ancient Sites - part 2
- Ancient Sites - part 3
- Ancient Sites - part 4
- Ancient Sites - part 5
- Ancient Sites - part 6

References

- Khufu Pyramid
- Oxygen XSLT Processors