Comp 336/436 - Markup Languages

Fall Semester 2017 - Week 10

Dr Nick Hayward

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* `>`, `>=`, `<`, `<=`
  - *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

XML - XPath details - functions - mathematics

- also include simple arithmetic operations with our expressions
- allow us to test for more complicated conditions
  - *or to output calculated values...*
- e.g to multiply, divide, add, or subtract,
  - *add first operand*
    - e.g. numerical constant 12 or a node set
  - *add mathematical operator*
    - `*` (for multiplication)
    - `div` (for division, since `/` is reserved)
    - `+` (for addition)
    - – (for subtraction)
  - add second operand
- multiplication and division are performed before addition and subtraction
  - *e.g. `4+5*3 = 19` and not 27*
  - *use parentheses to override the default, e.g. `(4+5)*3 = 27`*
- modulus operator may also be used
  - *e.g. `20 mod 4 = 0` (since 4 divides evenly into 20)*
  - *but `20 mod 3 = 2` since 20/3 is 6 with a remainder of 2*

XML - working example - ancient sites - mathematics

XML

```
<history>
  <period>New Kingdom</period>
  <dynasty>18</dynasty>
  <year range="start" era="BC">1346</year>
  <year range="end" era="BC">1332</year>
</history>
```

XSL

```
<xsl:choose>
  <xsl:when test="year[@range='end']">
    <xsl:value-of select="year[@range='start'] - year[@range='end']"/>
  </xsl:when>
  <xsl:otherwise>
   <xsl:value-of select="year[@range='start'] + 2017"/>
  </xsl:otherwise>
</xsl:choose>
```

- Demo - Ancient Sites 8

XML - XPath & XSLT tests - functions - mathematics

Exercise - part 9

- update your XSL stylesheet
  - *add an option to calculate a given value from the data in your XML*
    - add new values to XML, if necessary, to perform calculations
  - *add output to rendered document*
- test stylesheet with XML file

~ 10 minutes

XML - XPath details - functions - count nodes

- `count()` function
  - *count total nodes in a given node set*
  - *e.g. count images in an image set...*
- use the `count()` function as follows,
  - *add* `count(`
  - *add the path to the node set count*
  - *add* `)` *to complete the function*

XML - working example - ancient sites - count

XSL

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

- Demo - Ancient Sites 9

XML - XPath & XSLT tests - functions - count

Exercise - part 10

- update your XSL stylesheet
  - *add an option to count nodes in a given node set*
    - add new values to XML, if necessary, to perform count
  - *add output to rendered document*
- test stylesheet with XML file

~ 10 minutes

XML - XPath details - functions - format numbers

- standard arithmetic is performed using *floating point* mathematics
- may return very long numbers for certain calculations
- use `format-number()` function
  - *easily control required output of numbers*
- use the `format-number()` function as follows,
  - *add `format-number(`` *
  - *add expression or number to format*
  - *add `,  '` (a comma, a space, and a single quote)*
  - *add 0 for each digit that should always appear*
  - *add # for each digit that should only appear when* ***not zero***
  - *if necessary, add `.` (a period) to separate integer from fraction parts...*
  - *add `' )` to complete function*

n.b. there are many ways to add further formatting to output numbers...

XML - working example - ancient sites - format numbers

XML

```
<dimensions>
    <width type="average" unit="metre">230.360</width>
    <height type="original" unit="metre">146.59</height>
    <height type="current" unit="metre">138.75</height>
</dimensions>
```

XSL - part 1

```
<xsl:choose>
    <xsl:when test="dimensions">
        <xsl:apply-templates select="dimensions"/>
    </xsl:when>
    <xsl:otherwise>
        <td>N/A</td>
    </xsl:otherwise>
</xsl:choose>
```

XSL - part 2

```
<xsl:template match="dimensions">
  <td>
    <xsl:value-of select="format-number(./width * ./width, '0.#')"/> m<sup>2</sup>
  </td>
</xsl:template>
```

- Demo - Ancient Sites 10

XML - XPath & XSLT tests - functions - format numbers

Exercise - part 11

- update your XSL stylesheet
  - *add an option to format some numbers in your XML*
    - add new values to XML, if necessary
    - or format the result of a calculation...
  - *add output to rendered document*
- test stylesheet with XML file

~ 10 minutes

XML - XPath details - functions - round numbers

- three XPath functions for rounding numbers
  - *follows similar usage pattern to `format-number()`*
- round to the nearest integer using the `round()` function
- always round up with the `ceiling()` function
- always round down using the the `floor()` function
- use the round functions as follows,
  - *add `ceiling(`, `floor(`, or `round(` depending on requirements*
  - *add expression or number to format*
  - *add `)` to complete function*

XML - working example - ancient sites - round numbers

XSL

```
<xsl:template match="dimensions">
  <td>
    approx. <xsl:value-of select="ceiling(./width * ./width)"/> m<sup>2</sup>
  </td>
</xsl:template>
```

- Demo - Ancient Sites 11

XML - XPath & XSLT tests - functions - round numbers

Exercise - part 11

- update your XSL stylesheet
  - *add an option to format some numbers in your XML*
    - add new values to XML, perform calculation, round result...
  - *add output to rendered document*
- test stylesheet with XML file

~ 10 minutes

XML - XPath details - functions - extract substrings

- useful option to extract substrings from XML content
- extract substrings for processing, rendering, computed values...
- extract substrings before or after a particular character
  - *use* `substring-before()` *or* `substring-after()`
- to use substring functions before and after,
  - *add either* `substring-before(` *or* `substring-after(`
  - *choice of function depends on required part of source string*
  - *add expression containing the source string*
  - *add* `)` *to close the function*
- also possible to extract specific substring within source string
  - *e.g. start at character 3 in the source string, extract 5 characters*
- to use specific substring function, `substring(s,f,n)`
  - *add* `substring(`
  - *add expression for source string,* `s`
  - *add position of the first character for the substring,* `f`
  - *add total number of characters to extract,* `n`

XML - working example - ancient sites - extract substrings

XML

```
<notes>
  <note type="intro">
    ... add lots of text ...
  </note>
</notes>
```

XSL

```
<xsl:template match="notes/note[@type='intro']">
  <td>
    <xsl:value-of select="substring(.,1,75)"/>
    ...
  </td>
</xsl:template>
```

- Demo - Ancient Sites 12

XML - XPath & XSLT tests - functions - extract substrings

Exercise - part 12

- update your XSL stylesheet
  - *add an option to extract a substring from a string value in your XML*
    - add new values to XML, if necessary
  - *add output to rendered document*
- test stylesheet with XML file

~ 10 minutes

XML - XPath details - functions - modify case

- whilst processing and rendering text
  - *useful to change letters from upper-case to lower-case*
  - *and vice-versa...*
- use the following pattern to capitalise characters
  - *add* `translate(`
  - *add expression containing source string*
  - *Next, add* `, abcdefghijklmnopqrstuvwxyz`
    - (a comma, a space, and string containing letters to change)
  - *add* `, ABCDEFGHIJKLMNOPQRSTUVWXYZ`
    - (a comma, a space, and string containing letters to replace)
  - *add* `)` *to complete the function*
- process function will also work either way

XML - working example - ancient sites - modify case

XML

```
...
<overview type="general" url="...">wikipedia</overview>
...
```

XSL

```
...
<xsl:value-of select="translate(., 'w', 'W')"/>
...
```

- Demo - Ancient Sites 13

XML - XPath & XSLT tests - functions - modify case

Exercise - part 13

- update your XSL stylesheet
  - *add an option to modify the case of a string value in your XML*
    - add new values to XML, if necessary
  - *add output to rendered document*
- test stylesheet with XML file

~ 10 minutes

Demos

XML & XSLT - Part 2 - Functions

- Ancient Sites - comparison - part 7
- Ancient Sites - mathematics - 8
- Ancient Sites - count - part 9
- Ancient Sites - format numbers - 10
- Ancient Sites - round numbers -part 11
- Ancient Sites - substrings - part 12
- Ancient Sites - modify case - part 13

References

- XPath Version 1.0 functions
- XPath Version 2 functions