Ch. 4: XPath Functions

Seoul National University, Internet Database Laboratory

July, 2015

Contents

- Comparing Two Values
- Testing the Position
- Multiplying, Dividing, Adding, Subtracting
- Counting Nodes
- Formatting Numbers
- Rounding Numbers
- Extracting Substrings
- Changing the Case of a String
- Totaling Values
- More XPath Functions

Comparing Two Values [1/2]

- [**first-node-set** *comparer* **second-node-set(value)**]
  - **First-node-set** : path to the first node set that you want to compare
  - *Comparer*
    - = / != / &gt; /&gt;= (greater than or equal to) / &lt; / &lt;= (less than or equal to)
  - **Second-node-set(value)** : a value or a path to the node set that you want to compare with the first-node-set

< xslt >

```
...
<h2>Overview</h2>
    <table border="1"> <tr> <th> Wonder Name</th> <th>Location</th>
        <th>Height</th> </tr>

   <xsl:apply-templates select="ancient_wonders/ wonder[height &gt; 100]">

    <xsl:sort select="height“  order="descending“ data-type="number" />
    </xsl:apply-templates>
    </table>
...
```

Testing the Position

- **position()** = n
  - n : the number that identifies the position of the current node set
- **last()** : returns the last node

< xslt >

```
...
<p>These ancient wonders are

<xsl:for-each select= "ancient_wonders/wonder/name[@language='English']">

    <xsl:value-of select="."/>
    <xsl:choose>

        <xsl:when test= "position()=last()“ >.</xsl:when>
        <xsl:when test= "position()=last()-1“ >, and </xsl:when>
        <xsl:otherwise>, </xsl:otherwise>

    </xsl:choose>
</xsl:for-each>
</p>
...
```

Multiplying, Dividing, Adding, Subtracting [1/2]

- **first operand** *operator* **second operand**
  - **First operand** : numerical constant / a node set
  - *Operator:* * / div / + / -
  - **Second operand** : numerical constant / a node set

< xslt >

```
...
<td valign="top">
    <xsl:choose>
      <xsl:when test="history/year_destroyed != 0">

        <xsl:choose>
           <xsl:when test="history/year_destroyed/@era = 'BC'">
               <xsl:value-of select= "history/year_built - history/year_destroyed“ />
           </xsl:when>
           <xsl:otherwise>
              <xsl:value-of select= "history/year_built + history/year_destroyed - 1“ />
           </xsl:otherwise>
        </xsl:choose>

      </xsl:when>
      <xsl:otherwise>
          <xsl:value-of select = "history/year_built + 2008 - 1“ />
      </xsl:otherwise>
    </xsl:choose>
</td>
...
```

Counting Nodes

- **count(*path*)**
  - *path* : the path to the node to be counted
  - The location *path* referenced can include predicates

< xslt >

```
...
Of these wonders,
<xsl:value-of select=
         "count(ancient_wonders/wonder/history/how_destroyed[. ='earthquake'])" />
were destroyed by earthquake,

<xsl:value-of select="count(//how_destroyed[. = 'fire'])"/>
were destroyed by fire, and
<xsl:value-of select="count(//wonder) - count(//how_destroyed)" />
is still standing.
...
```

Formatting Numbers [1/2]

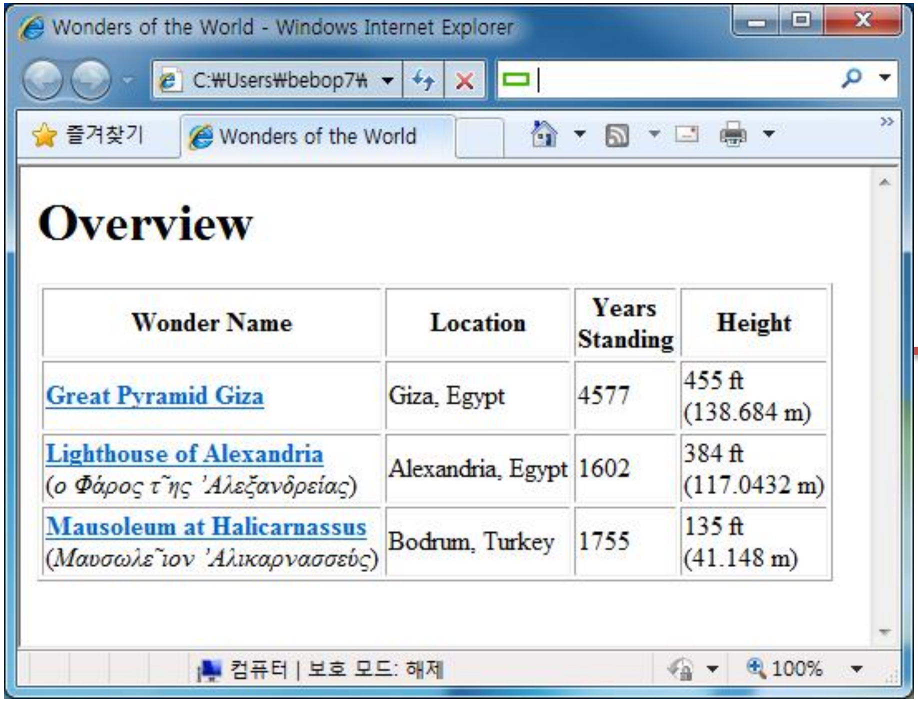
- **format-number(*expression*, '*format*')**
  - *expression* : contains the number to be formatted
  - *format*
    - **0** : a digit that should always appear
    - **#** : a digit that should only appear when not zero
    - **.**(a period) : separate the integer and fraction
    - **,**(a comma) : separate groups of digits in integer

< xslt >

```
...
<xsl:when test="height != 0">
   <xsl:value-of select= "height"/> feet<br />
(<xsl:value-of select= "height * 0.3048"/> m)
...
```

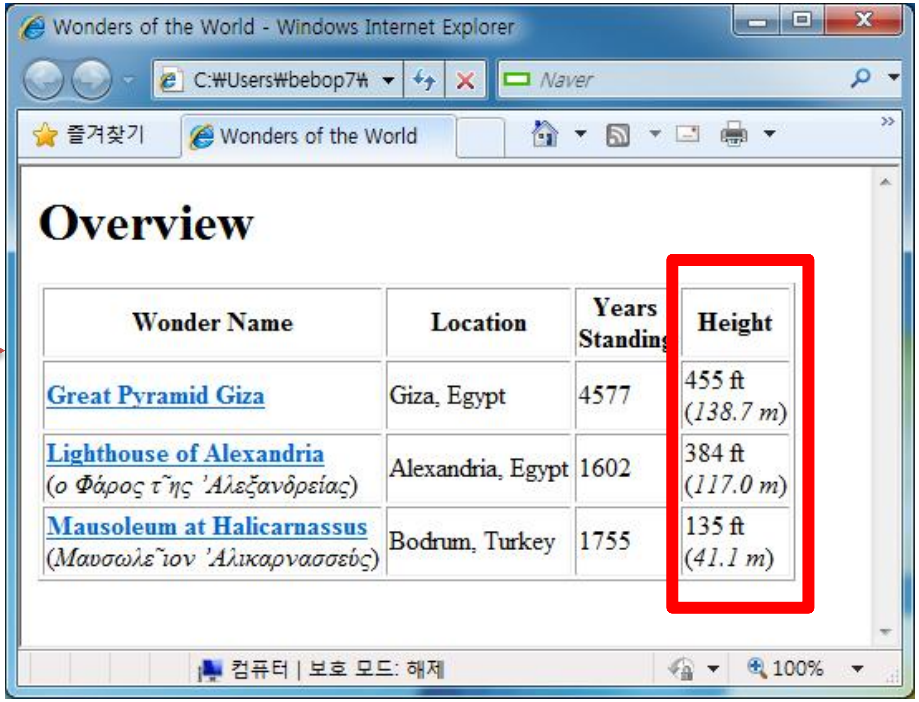
```
...
"(<em><xsl:value-of select=
   "format-number(height * 0.3048, '##0.0')"/> m
   </em>)"/> m)
...
```

Formatting Numbers [2/2]



Overview

Wonder Name	Location	Years Standing	Height
Great Pyramid Giza	Giza, Egypt	4577	455 ft (138.684 m)
Lighthouse of Alexandria (ο Φάρος τ῀ης 'Αλεξανδρείας)	Alexandria, Egypt	1602	384 ft (117.0432 m)
Mausoleum at Halicarnassus (Μαυσωλε῀ιον 'Αλικαρνασσεύς)	Bodrum, Turkey	1755	135 ft (41.148 m)



Overview

Wonder Name	Location	Years Standing	Height
Great Pyramid Giza	Giza, Egypt	4577	455 ft (*138.7 m*)
Lighthouse of Alexandria (ο Φάρος τ῀ης 'Αλεξανδρείας)	Alexandria, Egypt	1602	384 ft (*117.0 m*)
Mausoleum at Halicarnassus (Μαυσωλε῀ιον 'Αλικαρνασσεύς)	Bodrum, Turkey	1755	135 ft (*41.1 m*)

Rounding Numbers

- **ceiling(***expression***)** / **floor(***expression***)** / **round(***expression***)**
  - *expression* : contains the number to be formatted
- When using the **format-number()**, if there are any decimal places lost, XSLT will automatically round the result

< xslt >

```
...
<img>
    <xsl:attribute name="src">
        <xsl:value-of select="./@file"/>
    </xsl:attribute>
    <xsl:attribute name="width">
        <xsl:value-of select="ceiling(./@w div 2)"/>
    </xsl:attribute>
    <xsl:attribute name="height">
        <xsl:value-of select="ceiling(./@h div 2)"/>
    </xsl:attribute>
</img>
...
```

- Result

< xml >

```
<main_image
file="artemis.jpg" w="528" h="349"/>
```

→

< html >

```
…
<p align="center">
<img src="artemis.jpg"
width="264" height="175"/>
…
```

Extracting Substrings

- **substring-after(*expression*, '*c*')**
  substring-before(*expression*, '*c*')
  - *expression* : contains the number to be formatted
  - *c* : the character after or before the substring to be extracted

- Tip
  - **substring(*s*, *f*, *n*)**
    - *s* : expression
    - *f* : the position of the first character that you want to extract
    - *n* : the total number of characters

< xslt >

```
...
<h2>Overview</h2>
<table border="1"><tr>
    <th>Wonder Name</th>
    <th>City</th>
    <th>Country</th>
    <th>Years<br />Standing</th>
    <th>Height</th></tr>
...
<td valign="top">
    <xsl:value-of select="
        substring-before(location, ',')"/>
</td>
<td valign="top">
    <xsl:value-of select="
        substring-after(location, ',')"/>
</td>
...
```

Changing the Case of a String

- Example from lower case to upper case
  - **translate**(*expression*, ‘*abcdefghijklmnopqrstuvwxyz*’, ‘*ABCDEFGHIJKLMNOPQRSTUVWXYZ*’)
    - *expression* : contains the number to be formatted
    - ‘*abcdefghijklmnopqrstuvwxyz*’ : lower case alphabets
    - ‘*ABCDEFGHIJKLMNOPQRSTUVWXYZ*’ : upper case alphabets

< xslt >

```
...
<p align="center">
<strong><xsl:value-of select="
translate(../name[@language='English‘],'abcdefghijklmnopqrstuvwxyz'
,'ABCDEFGHIJKLMNOPQRSTUVWXYZ')"/>
</strong><br />

<xsl:apply-templates select="../main_image"/>
</p>
...
```

Totaling Values

- **sum(***path***)**
  - *path* : the path to the node set whose nodes should be totaled

< xslt >

```
...
<tr>
<td valign="top" align="right"
   colspan="4">Average Height: </td>
<td valign="top">
   <xsl:value-of select="
   format-number(
      sum(/ancient_wonders/wonder/height)
      div
      count(/ancient_wonders/wonder/height[.!=0]),
   '##0.0')" />
ft</td>
</tr>
...
```

More XPath Functions

- Node Functions
  - **name(***node-set***)** : returns the first node in the specified *node-set*
  - **id(***id-str***)** : returns all the elements that an ID equal to *id-str*
- String Functions
  - **contains(***str1***,** *str2***)** : returns true if *str1* contains *str2*
  - **string-length(***str1***)** : returns the number of characters in *str1*
  - **normalize-space(***str1***)** : returns *str1* with
    - all leading and trailing white space removed
    - Sequences of white space replaced with a single space
- Boolean Functions
  - **not(***expression***)** : returns true if expression evaluates to false
  - **| (a vertical bar)** : used to combine two node sets into one