upstate.edu

Lesson 6: Navigating an XML Tree

29-37 minutes

Block XML and `$contentRoot`

- One of the uses of formats is to transform block XML into presentable XHTML markups
- When both a block and a format are attached to the same page region, a variable named `$contentRoot` is defined by Velocity for the format to use
- The variable `$contentRoot` is defined for the format attached to a region only if there is a block (but normally not a text block) attached to the same region; otherwise the variable is undefined
- The value associated with the variable, when defined, is a Java object of type `org.jdom.Element`
- This object is the root element (see below) of the XML tree representing the block XML contents
- Before we transform the block XML, we must understand how to access the data stored within the XML tree
- We need to learn to use `org.jdom.Element` methods and the `$_XPathTool` object to get to the data

XML Terminology

- Before working with block XML, let us establish some XML terminology
- Consider the following XML:

```
<books>
    <book id="1">
        <title>Wuthering Heights</title>
        <author>Emily Brontë </author>
    </book>
    <book id="2">
        <title>Pride and Prejudice</title>
        <author>Jane Austen</author>
    </book>
</books>
```

- Each String like `<books>` or `<books>` are called tags in the XML markups
- `<books>` is a start tag, and `</books>` is an end tag
- A start tag and its corresponding end tag and everything in between define an element
- Assuming that this is the entire XML document, there is a root element whose name is `books` and contains everything in the XML markups
- The two `book` elements are said to be children of the `books` element, because the two `book` elements are directly contained inside the `books` element
- The `books` element is the parent of the two `book` elements

- The two `book` elements are said to be siblings because they have the same parent
- The two `book` elements each has two children: a `title` element and an `author` element
- A `title` element like <title>Wuthering Heights</title>, which contains a book title, has a child text node
- The text node, which is a child of `title`, is the node containing the text `Wuthering Heights`
- `id="1"` is an attribute of the first `book` element; `id` is the name of the attribute, and `1` is its value
- In the environment of Java, an XML document is normally represented as a tree
- An element is represented as a node in the tree
- A tree is a set of nodes including the root node and all and only nodes related by the parent-children relationship
- A tree has a unique root node, the root node is the ancestor of all other nodes in the same tree, and all other nodes are the descendants of the root node
- Any part of an XML tree including a node and all and only its descendants is a subtree of the larger tree
- A tree is a subtree of itself
- A node that has no children is a terminal node
- We can also consider a node that contains only a text node a terminal node derivatively

- A terminal node that has no child text node is an empty node
- The parent of the root node is the document object
- A set of unrelated trees without a root node is an XML fragment
- The `$contentRoot` variable, when defined, represents the root node of the XML tree

Navigation in an XML Tree

- To visit a node in an XML tree, we normally start from the root node; i.e., `$contentRoot`
- From the root node, we can visit one, or all, of its children
- If a child node is also a parent node, then we can visit this node's children
- From a node, we can also visit its parent node, unless the node is the root node; in this case, the parent of the root node is the document
- When visiting a node, we can also visit its attributes
- We can keep this visiting routine: from the root, we visit one of its children, and from this point on, we keep visiting a node's child, until we reach a certain node in the tree
- The route from the root node to this specific node is a path in the tree
- A path can start from a node and end with one of its descendant nodes
- A String representation of a path is an XPath expression

- XPath expressions can not only represent paths, but also represent random node-sets of a tree

Displaying Tree Structure

- Let us start with looking at a recursive macro to display the hierarchal structure of an XML tree
- The macro is defined in the library format named `chanw-process-xml`
- To avoid outputting unnecessary whitespace characters, we have to remove these characters from our code
- Here is the macro:

```
#macro( chanwDisplayXmlTree
$node_chanwDisplayXmlTree )
#if( !$node_chanwDisplayXmlTree.Class.Name ||
$node_chanwDisplayXmlTree.Class.Name !=
"org.jdom.Element" )
No node is passed in.
#break
#end
##
## used to keep track of the level of the nodes
#if( !$chanwDisplayXmlTreeCounter )
#set( $chanwDisplayXmlTreeCounter = 0 )
#else
#set( $chanwDisplayXmlTreeCounter =
$chanwDisplayXmlTreeCounter + 1 )
#end
```

```
## for formatting the tree
#set( $chanwSpace = "    " )
## output the name of the node
$node_chanwDisplayXmlTree.Name
#if(
$node_chanwDisplayXmlTree.getChildren().size()
> 0 )
#foreach( $child in
$node_chanwDisplayXmlTree.getChildren() )
#foreach( $num in [ 0 ..
$chanwDisplayXmlTreeCounter ]
)$chanwSpace#end#chanwDisplayXmlTree( $child
)#end
#end
## adjusting the counter of levels
#if( $chanwDisplayXmlTreeCounter > 0 )
#set( $chanwDisplayXmlTreeCounter =
$chanwDisplayXmlTreeCounter - 1 )
#end
#end
```

- I use `#break` to exit the execution of the macro if no node is passed in
- To use the macro, we need code of the following type:

  ```
  #import( "site://_brisk/core/library/velocity
  /chanw/chanw-process-xml" )

  #chanwDisplayXmlTree( $contentRoot )
  ```

- When the `$contentRoot` of a page is passed into this macro, we will see information of the following type:

```
system-index-block
   system-page
      name
      is-published
      last-published-on
      last-published-by
      title
      display-name
      path
      site
      link
      created-by
      created-on
      last-modified-by
      last-modified
      dynamic-metadata
         name
      dynamic-metadata
         name
      dynamic-metadata
         name
      dynamic-metadata
         name
         value
   system-folder
      name
      is-published
      last-published-on
```

```
last-published-by
display-name
path
site
link
created-by
created-on
last-modified-by
last-modified
dynamic-metadata
   name
dynamic-metadata
   name
dynamic-metadata
   name
system-page
   name
   is-published
   last-published-on
   last-published-by
   title
   display-name
   path
   site
   link
   created-by
   created-on
   last-modified-by
   last-modified
```

```
         dynamic-metadata
            name
         dynamic-metadata
            name
         dynamic-metadata
            name
         dynamic-metadata
            name
            value
         system-data-structure
            pre-main-group
               mul-pre-main-chooser
                  path
            main-group
               mul-pre-h1-chooser
                  path
               h1
               mul-post-h1-chooser
                  path
               float-pre-content-blocks-around-
wysiwyg-content
               wysiwyg
                  h2
                  p
                  ul
                     li
                     li
               mul-post-wysiwyg-chooser
                  path
```

```
      post-main-group
         mul-post-main-chooser
            path
      top-group
         mul-top-group-chooser
            path
      bottom-group
         mul-bottom-group-chooser
            path
      admin-group
         master-level-override
            path
         page-level-override
            path
system-folder
   name
   is-published
   last-published-on
   last-published-by
   display-name
   path
   site
   link
   created-by
   created-on
   last-modified-by
   last-modified
   dynamic-metadata
      name
   dynamic-metadata
      name
   dynamic-metadata
      name
system-folder
   name
   is-published
   last-published-on
   last-published-by
   display-name
   path
   site
   link
   created-by
   created-on
   last-modified-by
   last-modified
   dynamic-metadata
      name
   dynamic-metadata
      name
   dynamic-metadata
      name
system-page
   name
   is-published
   last-published-on
   last-published-by
   title
   display-name
   path
   site
   link
   created-by
   created-on
   last-modified-by
   last-modified
   dynamic-metadata
      name
   dynamic-metadata
      name
   dynamic-metadata
      name
   dynamic-metadata
      name
      value
system-folder
name
is-published
last-published-on
last-published-by
display-name
path
site
link
created-by
created-on
last-modified-by
```

```
      last-modified
      dynamic-metadata
         name
      dynamic-metadata
         name
      dynamic-metadata
         name
   system-symlink
      name
      title
      display-name
      path
      site
      created-by
      created-on
      last-modified-by
      last-modified
      dynamic-metadata
         name
      dynamic-metadata
         name
      dynamic-metadata
         name
      link
   calling-page
      system-page
         name
         is-published
         last-published-on
```

```
last-published-by
title
display-name
path
site
link
created-by
created-on
last-modified-by
last-modified
dynamic-metadata
   name
dynamic-metadata
   name
dynamic-metadata
   name
dynamic-metadata
   name
   value
system-data-structure
   pre-main-group
      mul-pre-main-chooser
         path
   main-group
      mul-pre-h1-chooser
         path
      h1
      mul-post-h1-chooser
         path
```

```
               float-pre-content-blocks-around-
wysiwyg-content
               wysiwyg
                  h2
                  p
                  ul
                     li
                     li
               mul-post-wysiwyg-chooser
                  path
            post-main-group
               mul-post-main-chooser
                  path
            top-group
               mul-top-group-chooser
                  path
            bottom-group
               mul-bottom-group-chooser
                  path
            admin-group
               master-level-override
                  path
               page-level-override
                  path
```

- This tree structure will help us a lot when working with the `org.jdom.Element` object `$contentRoot`

Important `org.jdom.Element` Methods and

Properties

- If the variable `$contentRoot` is defined, then it will serve as the starting point of our navigation
- If an `org.jdom.Element` method has a `java.lang.String` parameter, then normally the String is the name of an element; `org.jdom.Element` methods do not accept XPath expressions as arguments
- The `getName()` method and `Name` property return the name of an element
- The `getValue()` method and `Value` property return the text value of an element
- Note that when `getValue()` is called through a node that has child elements, all the text within the subtree rooted at this node will be returned
- The inherited `getParent()` method and `Parent` property return the parent of a element
- The `getChildText( java.lang.String )` method returns the text value of the named child:

  ```
  <system-data-structure>
      <block-type>megablock</block-type>
  </system-data-structure>
  ```

  In this XML, `$contentRoot.getChildText( "block-type" )` return the String `megablock`
- The `getChild( java.lang.String )` method returns the named child element

- The `getChildren()` method and `Children` property return a list containing all children
- The `getChildren( java.lang.String )` returns a list of children bearing the name
- The `getContentSize()` method and `ContentSize` property return the size of the content of the element; if the element contains child nodes, then the size of the content is the number of children of the element; if the element contains text value, then the size of the content is 1 (namely, the text node)
- `getContentSize()` can be used to test empty elements (when the method returns 0)

Selecting Nodes: Introducing `$_XPathTool`

- When working with block XML, we can also use the `$_XPathTool` object
- The type of this object is `org.jdom.xpath.JaxenXPath`
- This class is from an older version of jdom (1.1)
- `org.jdom.xpath.JaxenXPath` extends `org.jdom.xpath.XPath`
- The `org.jdom.xpath.JaxenXPath` API documentation cannot be found online
- The local XPath API documentation
- The two most important methods of `org.jdom.xpath.XPath`, inherited by `org.jdom.xpath.JaxenXPath`:
- `static java.util.List selectNodes(`

`java.lang.Object, java.lang.String )`

- `static java.lang.Object selectSingleNode( java.lang.Object, java.lang.String )`
- We always use these two `static` method to select nodes when using `$_XPathTool`; selected nodes are represented as `org.jdom.Element` objects
- When we consider these two Java methods returning an `org.jdom.Element` object or a list of such objects, we talk about selecting element(s)
- However, if we use XML terminology, then we can talk about selecting a node or a node-set
- In this context, the terms node and element can be treated as interchangeable
- When selecting a node or a node-set, we must have a starting point, and we also need a path to follow, starting from this starting point
- The starting point of selecting node(s) can be a node, or it can be a list of nodes
- The path we want to follow is an XPath expression
- If we start with a node, then the node is called the context node; the XPath expression is evaluated against this context node (start from the context node, and follow the path…)
- The context node, of type `org.jdom.Element`, is the first argument of the two `static` methods, and the XPath expression, a `java.lang.String` object, is the second

argument

- If we start with a list, then the list must be a list of `org.jdom.Element` objects
- Each node in the list will serve as the context node in turn

`org.jdom.xpath.XPath.selectSingleNode`

- In the context of Velocity, this methods takes two parameters
- The first parameter is an object of type `java.lang.Object`, which can be an object of a more specific type of:
- `org.jdom.Element`
- `java.util.ArrayList`
- The second parameter is a `java.lang.String` object, which should be an XPath expression
- If the first parameter is an `org.jdom.Element` object, then this node is the context node against which the XPath expression is evaluated
- This method returns a single `org.jdom.Element` object, representing a node meeting the requirements of the XPath expression, or `null` if no such node exists
- If the first parameter is a `java.util.ArrayList` object, then this list should contain objects of type `org.jdom.Element`, each of which will be treated as the context node in turn
- In this case, every node in the list will serve as the context node in turn, and the execution of the method will stop when the first context node is encountered and the requirements of the XPath

expression are met against this context node, and the method returns the `org.jdom.Element` object; if all nodes in the list have been tried as the context node and no node has been found that meets the requirement of the XPath expression, then the method returns `null`

`org.jdom.xpath.XPath.selectNodes`

- In the context of Velocity, this methods takes two parameters
- The first parameter is an object of type `java.lang.Object`, which can be an object of a more specific type of:
- `org.jdom.Element`
- `java.util.ArrayList`
- The second parameter is a `java.lang.String` object, which should be an XPath expression
- If the first parameter is an `org.jdom.Element` object, then this node is the context node against which the XPath expression is evaluated
- This method returns a list of `org.jdom.Element` objects, representing a node-set, each of which meeting the requirements of the XPath expression
- If no such nodes are found, then the method returns an empty list
- If the first parameter is a `java.util.ArrayList` object, then this list should contain objects of type `org.jdom.Element`, each of which will be treated as the context node in turn
- In this case, every node in the list will serve as the context node

in turn, and the execution of the method will stop when the first context node is encountered and the requirements of the XPath expression are met against this context node, and the method returns a list of `org.jdom.Element` objects; if all nodes in the list have been tried as the context node and no nodes have been found that meet the requirement of the XPath expression, then the method returns an empty list

- Important:
- `org.jdom.xpath.XPath.selectNodes` always returns a list, which can be empty
- `org.jdom.xpath.XPath.selectSingleNode` returns either `null` or a single `org.jdom.Element` object

More About the Context Node

- The value of `$contentRoot` serves as our initial context node
- Once we select a node, the selected node can become our new context node, against which an XPath expression can be evaluated
- Therefore, a long XPath expression can be broken up into a repetition of setting up a new context node with a much shorter XPath expression
- Example:

```
#set( $path_node =
    $_XPathTool.selectSingleNode(
        $contentRoot,
        "quick-links-group/quick-links-list-
group/quick-links-linkable/path" ) )
```

```
## equivalent to
#set( $qlg = $_XPathTool.selectSingleNode(
    $contentRoot, "quick-links-group" ) )
## a new context node
#set( $qllg = $_XPathTool.selectSingleNode(
    $qlg, "quick-links-list-group" ) )
## yet another new context node
#set( $path_node =
$_XPathTool.selectSingleNode(
    $qllg, "quick-links-linkable/path" ) )
```

Getting the Path of a Node from the Tree

- We saw above how to display an XML tree
- The displayed tree can help us when writing an XPath expression
- Simple XPath expression can be created by copying parts of the tree and making simple adjustments
- For a simple exercise, let us assume that we want to retrieve the `dynamic-metadata` element that has a `value` child
- Here is the relevant part of the tree:

```
system-index-block
   ...
   calling-page
      system-page
         name
         is-published
```

```
last-published-on
last-published-by
title
display-name
path
site
link
created-by
created-on
last-modified-by
last-modified
dynamic-metadata
   name
dynamic-metadata
   name
dynamic-metadata
   name
dynamic-metadata
   name
   value
system-data-structure
```

- Our target is the fourth `dynamic-metadata` element
- We can copy element names from this tree, starting for `calling-page`
- The first part of the path is always the name of a child of the context node
- Each part of the path should be followed by a `/` (slash), except

the last part: `calling-page/system-page/dynamic-metadata`

- If we want to output the text value of the `value` child of the fourth `dynamic-metadata` element, try the following: `$_XPathTool.selectSingleNode( $contentRoot, "calling-page/system-page/dynamic-metadata[4]/value" ).Value`

Selecting a Named Child

```
$contentRoot.getChild( "block-type" )
## or
$_XPathTool.selectSingleNode( $contentRoot,
"block-type" )
```

Selecting All `system-folder` Children

```
$contentRoot.getChildren( "system-folder" )
## or
$_XPathTool.selectNodes( $contentRoot, "system-
folder" )
```

Selecting All `system-folder` Descendants

```
$_XPathTool.selectNodes( $contentRoot,
"//system-folder" )
```

Selecting the Parent

```
$path_node.Parent
## or
```

```
$_XPathTool.selectSingleNode( $path_node,
"parent::node()" )
```

Selecting the Immediate Preceding Sibling

```
$_XPathTool.selectSingleNode( $qll, "preceding-
sibling::node()[last()]" )
```

Selecting the Immediate Following Sibling

```
#set( $qll = $_XPathTool.selectSingleNode(
    $qlt, "following-sibling::node()[1]" ) )
```

Selecting a Node with Specific Text Value

```
$_XPathTool.selectSingleNode( $contentRoot,
    "multiple-second[text()='multiple 2.3']" )
```

Selecting a `dynamic-metadata` Field

```
## select dynamic-metadata
$_XPathTool.selectSingleNode(
    $contentRoot,
    "calling-page/system-page/dynamic-
metadata[name='exclude-from-menu']" )

## select the name
$_XPathTool.selectSingleNode(
    $contentRoot,
    "calling-page/system-page/dynamic-metadata
/name[text()='exclude-from-menu']" )
```

```
## select the value
$_XPathTool.selectSingleNode(
    $contentRoot,
    "calling-page/system-page/dynamic-
metadata[name=='exclude-from-menu']/value"
).Value  )
```

Selecting a Node in a Page

```
$_XPathTool.selectSingleNode(
    $contentRoot,
    "calling-page/system-page/system-data-
structure//content-group-size" )
```

Selecting an Index Block attached to a Block Chooser on a Page

```
#set( $index_block =
$_XPathTool.selectSingleNode(
    $contentRoot,
    "calling-page/system-page/system-data-
structure//site-storage//system-index-block" )
)

## go back to the block chooser
#set( $block_chooser =
$index_block.Parent.Parent )

## get the path of the index block
```

```
$block_chooser.getChild( "path" ).Value

## get the site name of the index block
$block_chooser.getChild( "site" ).Value
```

More Examples

```
## select all pages with both is-published and
last-published-on children
#set( $pages = $_XPathTool.selectNodes(
$contentRoot, "//system-page[is-published]
[last-published-on]" ) )

## select pages with start-date child, and
whose path child does not contain the base
#set( $entries =
    $_XPathTool.selectNodes( $contentRoot,
"system-page[start-
date][not(contains(path,'${base}'))]" ) )

## select the page with the current attribute
#set( $page = $_XPathTool.selectSingleNode(
$contentRoot, "//system-page[@current]" ) )

## select all non-index pages, folders, and
symlink within the current folder
#set( $subfolders =
    $_XPathTool.selectNodes( $folder, "system-
page[name!='index'] | system-folder | system-
```

```
symlink" ) )

## select all blocks named setup
#set( $blocks = $_XPathTool.selectNodes(
$contentRoot, "//system-block[name='setup']" )
)

## select all pages and folders with pages in
the base folder
#set( $items =
    $_XPathTool.selectNodes($contentRoot,
"system-page | system-
folder[descendant::system-page]" ) )

## select all Cells, ignoring namespace
prefixes
#set( $ces = $_XPathTool.selectNodes(
$contentRoot, "//*[local-name() = 'Cells']" ) )

## nested predicates
#set ( $jobs =
    $_XPathTool.selectNodes(
        $contentRoot, "system-page[system-data-
structure[job-info[city1='Ash Grove' or
city2='Ash Grove'] and job-category='C.N.A.']]"
) )

## distinct tags, from Erik Gorka
#set( $tags = $_XPathTool.selectNodes(
$pages,"//tag[not(.=preceding::tag)]" ) )

## from Ryan Griffith
## all dynamic metadata whose name has a
certain prefix
#set( $categories = $_XPathTool.selectNodes(
$page, "dynamic-metadata[starts-with(name,
'directory-categories-')]/value" ) )

## select the current page and all its ancestor
folders
#set( $cur_page = $_XPathTool.selectNodes(
$contentRoot, "//system-page[@current]" )[ 0 ]
)
#set( $folders = $_XPathTool.selectNodes(
$cur_page, "ancestor::node()[@id]" ) )
```

XPath Node Test

- Syntax: `axis::node_test[predicate]`
- The name of an element
- The name of an attribute

```
## select the value of the id attribute of the
parent of the current page
$_XPathTool.selectNodes(
$_XPathTool.selectNodes( $contentRoot,
"//system-page[@current]" )[ 0 ].Parent, "@id"
)[ 0 ].Value
```

- The wildcard `*`
  ```
  ## select all attributes of the parent of the
  current page
  $_XPathTool.selectNodes(
  $_XPathTool.selectNodes( $contentRoot,
  "//system-page[@current]" )[ 0 ].Parent, "@*" )
  ```
- `comment()`
- `node()`
- `processing-instruction()`
- `text()`
  ```
  ## select all elements that do not contain a
  child 3text node
  $_XPathTool.selectNodes( $contentRoot,
  "//*[not(text())]" )
  ```

XPath Axes

- `ancestor`
  ```
  ## select all ancestor folder of the current
  page
  $_XPathTool.selectNodes(
  $_XPathTool.selectNodes( $contentRoot,
  "//system-page[@current]" )[ 0 ],
  "ancestor::system-folder[@id]" )
  ```
- `ancestor-or-self`
  ```
  ## select the context node (a folder) and all
  its ancestor folders
  $_XPathTool.selectNodes(
  ```

```
$_XPathTool.selectNodes( $contentRoot,
"//system-page[@current]" )[ 0 ].Parent,
"ancestor-or-self::system-folder[@id]" )
```

- attribute
```
## select all attributes of the current page
$_XPathTool.selectNodes(
$_XPathTool.selectNodes( $contentRoot,
"//system-page[@current]" )[ 0 ],
"attribute::node()" )
```
- child
```
## select all children (is-published, last-
published-on, etc.) of the current page
$_XPathTool.selectNodes(
$_XPathTool.selectNodes( $contentRoot,
"//system-page[@current]" )[ 0 ],
"child::node()" )
```
- descendant
```
## select all descendants of the current page
$_XPathTool.selectNodes(
$_XPathTool.selectNodes( $contentRoot,
"//system-page[@current]" )[ 0 ],
"descendant::node()" )
```
- descendant-or-self
```
## select the current page and all its
descendants
$_XPathTool.selectNodes(
$_XPathTool.selectNodes( $contentRoot,
"//system-page[@current]" )[ 0 ], "descendant-
or-self::node()" )
```

- following
```
## select all following elements of the current
page
$_XPathTool.selectNodes(
$_XPathTool.selectNodes( $contentRoot,
"//system-page[@current]" )[ 0 ],
"following::node()" ).size()
```
- following-sibling
```
## select all following siblings of the current
page
$_XPathTool.selectNodes(
$_XPathTool.selectNodes( $contentRoot,
"//system-page[@current]" )[ 0 ], "following-
sibling::node()" )
```
- namespace
- parent
```
## select the parent of the current page
$_XPathTool.selectNodes(
$_XPathTool.selectNodes( $contentRoot,
"//system-page[@current]" )[ 0 ],
"parent::node()" )[ 0 ]
```
- preceding
```
## select all preceding elements of the current
page
$_XPathTool.selectNodes(
$_XPathTool.selectNodes( $contentRoot,
"//system-page[@current]" )[ 0 ],
"preceding::node()" )
```

- `preceding-sibling`
  ```
  ## select all preceding siblings of the current
  page$_XPathTool.selectNodes(
  $_XPathTool.selectNodes( $contentRoot,
  "//system-page[@current]" )[ 0 ], "preceding-
  sibling::node()" )
  ```
- `self`
  ```
  ## select the current page itself
  $_XPathTool.selectNodes(
  $_XPathTool.selectNodes( $contentRoot,
  "//system-page[@current]" )[ 0 ],
  "self::node()" )
  ```

XPath Node Set Functions

- `name()`
  ```
  ## select the name of the context node
  $_XPathTool.selectNodes( $contentRoot, "name()"
  )[ 0 ]
  ```
- `last()`
  ```
  ## select the id's of last folders at every
  level
  $_XPathTool.selectNodes( $contentRoot,
  "//system-folder[last()]/@id" )
  ```
- `position()`

```
## select the id's of first folders at every level
$_XPathTool.selectNodes( $contentRoot,
"//system-folder[position()=1]/@id" )
```

- `count()`
  ```
  ## select the number of all child elements that have a name child element
  $_XPathTool.selectNodes( $contentRoot,
  "count(node()/name)" )[ 0 ]
  ```
- `id()`: does not work in Cascade

XPath Operators

- `=`
- `!=`
- `<`
- `>`
- `<=`
- `>=`
- `|`
- `+`
- `-`
- `*`
- `div`
- `or`

- `and`
- `mod`

XPath Boolean Functions

- `boolean( object )`
- `false()`
- `not( boolean )`
- `true()`
- `lang( String )`

XPath Number Functions

- `ceiling( number )`
- `floor( number )`
- `number( object )`
- `round( number )`
- `sum( node-set )`

XPath String Functions

- `String( object )`
- `concat( String, String )`
- `contains( String, String )`
- `starts-with( String, String )`
- `subString-before( String, String )`

- `subString-after( String, String )`
- `subString( String, number, number )`
- `String-length( String )`
- `normalize-space( String )`
- `translate( String, String, String )`

Important Points About XPath Expressions

- For comparison of equality, use =, not ==
- Node position starts with 1, not 0

Library Macros to Display Path Information

- At Upstate, we use the data definition dd_page.xml for pages

- The library format named chanw-process-xml can be used to display path information of any XML
- For example, a page is processed by `#chanwProcessSystemDataStructure` and `#chanwDisplayMapKeys` is invoked to display the result:

```
#import( "site://_brisk/core/library/velocity
/chanw/chanw-process-xml" )

#set( $page = $_XPathTool.selectSingleNode(
$contentRoot, "calling-page/system-page" ) )
```

```
## skip processing the wysiwyg node
#chanwProcessSystemDataStructure( $page [
"wysiwyg" ] )
#chanwDisplayMapKeys( $chanwXmlXPathTextMap
true )
```

And here is the path information being displayed:

```
name
is-published
last-published-on
last-published-by
title
display-name
path
site
link
created-by
created-on
last-modified-by
last-modified
dynamic-metadata[1]
dynamic-metadata[1]/name
dynamic-metadata[2]
dynamic-metadata[2]/name
dynamic-metadata[3]
dynamic-metadata[3]/name
dynamic-metadata[4]
dynamic-metadata[4]/name
dynamic-metadata[4]/value
```

```
system-data-structure
system-data-structure/pre-main-group
system-data-structure/pre-main-group/mul-pre-
main-chooser
system-data-structure/pre-main-group/mul-pre-
main-chooser/path
system-data-structure/main-group
system-data-structure/main-group/mul-pre-h1-
chooser
system-data-structure/main-group/mul-pre-h1-
chooser/path
system-data-structure/main-group/h1
system-data-structure/main-group/mul-post-h1-
chooser
system-data-structure/main-group/mul-post-h1-
chooser/path
system-data-structure/main-group/float-pre-
content-blocks-around-wysiwyg-content
system-data-structure/main-group/wysiwyg
system-data-structure/main-group/mul-post-
wysiwyg-chooser
system-data-structure/main-group/mul-post-
wysiwyg-chooser/path
system-data-structure/post-main-group
system-data-structure/post-main-group/mul-post-
main-chooser
system-data-structure/post-main-group/mul-post-
main-chooser/path
system-data-structure/top-group
```

```
system-data-structure/top-group/mul-top-group-
chooser
system-data-structure/top-group/mul-top-group-
chooser/path
system-data-structure/bottom-group
system-data-structure/bottom-group/mul-bottom-
group-chooser
system-data-structure/bottom-group/mul-bottom-
group-chooser/path
system-data-structure/admin-group
system-data-structure/admin-group/master-level-
override
system-data-structure/admin-group/master-level-
override/path
system-data-structure/admin-group/page-level-
override
system-data-structure/admin-group/page-level-
override/path
```

- Note that:
- The context node is `calling-page/system-page`
- Every line is a valid XPath expression, from the context node, that can be used to access a node on the page

`selectNodes` vs. `selectSingleNode`

- `selectNodes` always returns a list
- `selectSingleNode` returns an `org.jdom.Element` object or `null`

- There is a subtle difference between these two methods
- When `selectSingleNode` is called and if no node is selected, the method returns `null`; when it does, we are not warned
- On the other hand, when `selectNodes` is called and if no node is selected, the method returns an empty list
- If we use code like `$_XPathTool.selectNodes( $node, "expression" )[ 0 ]`, Velocity will throw an exception if the list is empty
- This means that if there is a typo in the XPath expression, we will not be warned if we use `$_XPathTool.selectSingleNode( $node, "expression" )`, but the code `$_XPathTool.selectNodes( $node, "expression" )[ 0 ]` will throw an exception
- That's the reason why I recommend consistent uses of `$_XPathTool.selectNodes( $node, "expression" )[ 0 ]` to avoid possible bugs altogether