XPath Tutorial and Reference

Better edited version of the tutorial published by W3Schools
Some knowledge of XML is supposed
editor: Jac Gubbels
May 2005, UIL-OTS, University of Utrecht

Introduction

XPath is a language for finding information in an XML document and is used to navigate through elements and attributes in an XML document. XPath uses **path expressions** to select nodes or node-sets in an XML document. These path expressions look very much like the expressions you see when you work with a traditional computer file system. XPath includes over 100 built-in **functions**. There are functions for string values, numeric values, date and time comparison, node and QName manipulation, sequence manipulation, Boolean values, and more. XPath was designed to be used by XSLT, XPointer and other XML parsing software.

XPath nodes

In XPath, there are seven kinds of **nodes**: **element**, **attribute**, **text**, **namespace**, **processing-instruction**, **comment**, and **document (root)** nodes. XML documents are treated as trees of nodes. The root of the tree is called the document node (or root node). Some examples considering the xml document below:

- <bookstore> (document node)
- <author>J K. Rowling</author> (element node)
- lang="en" (attribute node)
- J K. Rowling (atomic value – not a node)
- “en” (atomic value – not a node)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
   <book>
       <title lang="en">Harry Potter</title>
       <author>J K. Rowling</author>
       <year>2005</year>
       <price>29.99</price>
   </book>
</bookstore>
```

Relationships of nodes

Each element and attribute has one **parent**. In the example the book element is the parent of the title, author, year, and price. Element nodes may have zero, one or more **children**. In the example; the title, author, year, and price elements are all children of the book element. **Siblings** are nodes that have the same parent. In the example; the title, author, year, and price elements are all siblings. **Ancestors** are a node's parent, parent's parent, etc. In the example; the ancestors of the title element are the book element and the bookstore element. **Descendants** are a node's children, children's children, etc. In the example; descendants of the bookstore element are the book, title, author, year, and price elements.

XPath Syntax

XPath uses **path expressions** to select nodes or node-sets in an XML document. The node is selected by following a **path** or **steps**. We will use the following XML document in all examples:

```
?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
  <book>
    <title lang="eng">Harry Potter</title>
    <price>29.99</price>
  </book>
  <book>
    <title lang="eng">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore
```

The most useful path expressions in **selecting nodes**:

- *nodename* Selects all child nodes of the node
- **/**: Selects from the root node
- **//**: Selects nodes in the document from the current node that matches the selection no matter where they are
- **.**: Selects the current node
- **..**: Selects the parent of the current node
- **@**: selects attributes

Some examples:

Path Expression	**Result**
bookstore	Selects all the child nodes of the bookstore element
/bookstore	Selects the root element bookstore **Note:** If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that is named lang

Predicates are used to find a specific node or a node that contains a specific value. Predicates are always embedded in square brackets. In the table below we have listed some path expressions with predicates and the result of the expressions:

Path Expression	**Result**
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element

/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='eng']	Selects all the title elements that have an attribute named lang with a value of 'eng'
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00]/title	Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00

XPath wild-cards can be used to select unknown XML elements:

- ***:** Matches any element node
- **@***: Matches any attribute node
- **node()**: Matches any node of any kind

In the table below we have listed some path expressions and the result of the expressions:

Path Expression	Result
/bookstore/*	Selects all the child nodes of the bookstore element
//*	Selects all elements in the document
//title[@*]	Selects all title elements which have any attribute

By using the | operator in an XPath expression you can **select several paths**. In the table below we have listed some path expressions and the result of the expressions:

Path Expression	Result
//book/title | //book/price	Selects all the title AND price elements of all book elements
//title | //price	Selects all the title AND price elements in the document
/bookstore/book/title | //price	Selects all the title elements of the book element of the bookstore element AND all the price elements in the document

XPath Axes

An axis defines a node-set relative to the current node.

AxisName	Result
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself

attribute	Selects all attributes of the current node
child	Selects all children of the current node
descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself
following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
namespace	Selects all namespace nodes of the current node
parent	Selects the parent of the current node
preceding	Selects everything in the document that is before the start tag of the current node
preceding-sibling	Selects all siblings before the current node
self	Selects the current node

A **location path** can be absolute or relative. An **absolute location path** starts with a slash (/) and a r**elative location path** does not. In both cases the location path consists of one or more steps, each separated by a slash:

- An absolute location path: **/step/step/...**
- A relative location path: **step/step/...**

Each step is evaluated against the nodes in the current node-set. A step consists of:

- axis (defines the tree-relationship between the selected nodes and the current node)
- node-test (identyfies a node within an axis)
- zero or more predicates (to further refine the selected node-set)

The syntax for a location step is:

- **axisname::nodetest[predicate]**

Some examples:

Example	**Result**
child::book	Selects all book nodes that are children of the current node
attribute::lang	Selects the lang attribute of the current node
child::*	Selects all children of the current node
attribute::*	Selects all attributes of the current node
child::text()	Selects all text child nodes of the current node
child::node()	Selects all child nodes of the current node
descendant::book	Selects all book descendants of the current node
ancestor::book	Selects all book ancestors of the current node
ancestor-or-self::book	Selects all book ancestors of the current node - and the current as well if it is a book node

child::*/child::price	Selects all price grandchildren of the current node

XPath Operators

XPath expressions returns either a node-set, a string, a Boolean, or a number. Below is a list of the operators that can be used in XPath expressions:

Operator	Description	Example	Return value
|	Computes two node-sets	//book | //cd	Returns a node-set with all book and cd elements
+	Addition	6 + 4	10
-	Subtraction	6 - 4	2
*****	Multiplication	6 * 4	24
div	Division	8 div 4	2
=	Equal	price=9.80	true if price is 9.80 false if price is 9.90
!=	Not equal	price!=9.80	true if price is 9.90 false if price is 9.80
<	Less than	price<9.80	true if price is 9.00 false if price is 9.80
<=	Less than or equal to	price<=9.80	true if price is 9.00 false if price is 9.90
>	Greater than	price>9.80	true if price is 9.90 false if price is 9.80
>=	Greater than or equal to	price>=9.80	true if price is 9.90 false if price is 9.70
or	or	price=9.80 or price=9.70	true if price is 9.80 false if price is 9.50
and	and	price>9.00 and price<9.90	true if price is 9.80 false if price is 8.50
mod	Modulus (division remainder)	5 mod 2	1

Further reference

More examples use the Microsoft XMLDOM object to load the XML document and the selectNodes() function to select nodes from the XML document:

- http://www.w3schools.com/xpath/xpath_examples.asp

Full functions reference:

- http://www.w3schools.com/xpath/xpath_functions.asp