e-PG Pathshala

Subject: Computer Science

Paper: Web Technology

Module: XPath

Module No: CS/WT/14

Quadrant 1 – e-text

Learning Objectives

The last module provides us an understanding about how to access and manipulate an XML DOM tree using JAVA. The module also explains about the concept of parsing XML using Java platform.

The objective of this module is to explain the concept of XPath and its syntax. This module also discusses about how to create XPath expressions. Moreover, we will also learn about XPath predicates, axes, operators and functions.

Introduction

XPath (the XML Path language) is a language for finding information in an XML document. It is a major element in the XSLT standard. Without XPath knowledge you will not be able to create XSLT documents.

XPath uses path expressions to navigate in XML documents and to select nodes or node-sets in an XML document. These path expressions look very much like the expressions used in JavaScript, Java, XML Schema, PHP, Python, C and C++, and lots of other languages.

XPath is a syntax for defining parts of an XML document. It contains a library of standard functions. It is also used in XQuery, XPointer and XLink.

Figure 14.1 shows how XPath, the path language is useful for other XML technologies like XSLT, XQuery, XPointer and XLink.

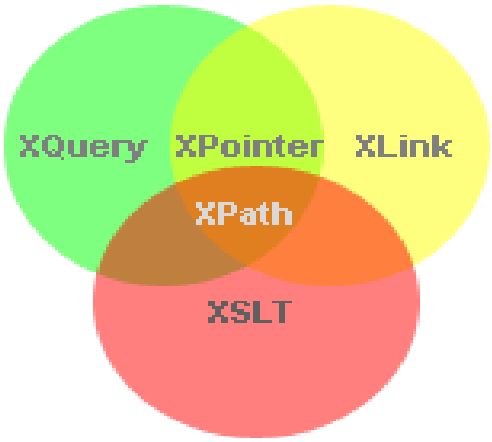

Figure 14.1 XPath used with XSLT, XQuery, XPointer and XLink

XPath Nodes

XPath identifies seven kinds of nodes like Element node, Attribute node, Text node, Namespace node, Processing Instruction node, Comment node and Document node. XML documents are also treated as trees of nodes.

XML CODE

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>                                    (ROOT ELEMENT NODE)
 <book>
        <title lang="en">Harry Potter</title>   (ATTRIBUTE NODE)
        <author>J K. Rowling</author>           (ELEMENT NODE)
        <year>2005</year>
        <price>29.99</price>
 </book>
</bookstore>
```

Relationship of Nodes

PARENT NODE

Each element and attribute has *one parent*. In the above example the <book> element is the parent of the <title>, <author>, <year>, and <price>.

CHILDREN NODE

Element nodes may *have zero, one or more children*. The <title>, <author>, <year>, and <price> elements are all children of the <book> element.

SIBLING NODE

Nodes that *have the same parent*. The <title>, <author>, <year>, and <price> elements are all siblings

ANCESTOR NODE

A node's parent, parent's parent, etc. The ancestors of the <title> element are the <book> element and the <bookstore> element.

DESCENDANT NODE

A node's children, children's children, etc. In the above example, descendants of the <bookstore> element are the <book>, <title>, <author>, <year>, and <price> elements.

XPath Syntax

Table 14.1: For Selecting Nodes

EXPRESSION	DESCRIPTION
nodename	Selects all nodes with the name "*nodename*"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

XPath Syntax for Path Expression

XPath uses a path expression to select node or list of nodes from an XML document.

Table 14.2: Path Expression

PATH EXPRESSION	DESCRIPTION

bookstore	Selects all nodes with the name "bookstore"
/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 are named lang

XPath Syntax for Predicates

Predicate refers to XPath expression written in square brackets. Predicates are used to find a specific node or a node that contains a specific value. Table 14.3 shows how to give path expressions with predicates.

Table 14.3: Path Expression and Predicates

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

Table 14.4 shows how to give path expressions with predicates along with attributes.

Table 14.4: Path Expression and Predicates Used with Attributes

PATH EXPRESSION	RESULT
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang='en']	Selects all the title elements that have a "lang" attribute with a value of "en"
/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

Wildcard in XPath

Selecting Unknown Nodes

XPath wildcards can be used to select unknown XML nodes. XPath defines following wildcards on nodes to be used with XPath expressions. Table 14.5 shows the different wildcards used and its description.

Table 14.5 Wildcards

Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

Selecting Several Paths

By using the **'|'** operator in an XPath expression you can select several paths. In the Table 14.6 below we have listed some path expressions that can be given with several paths and the result of the expressions.

Table 14.6 Selecting Several Paths

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

Axes are named so because they refers to axis on which elements are lying relative to an element. Following is the list of various Axes values given in Table 14.6 and Table 14.7.

Table 14.6 XPath Axes

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

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

Table 14.7 XPath Axes

AxisName	Result
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 all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes
preceding-sibling	Selects all siblings before the current node
Self	Selects the current node

Location Path Expression

A location path can be absolute or relative. An absolute location path starts with a slash (/) and a relative 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 will be given as: /step/step/...

A relative location path will be given as: step/step/...

Each step is evaluated against the nodes in the current node-set.

Location Path Expression Steps & Syntax

Location path expression consists of following steps:

1. An axis that defines the tree-relationship between the selected nodes and the current node.
2. A node-test that identifies a node within an axis or specifies the name of the node to be selected.
3. Zero or more predicates to further refine the selected node-set.

The syntax for a location step is:

axisname::nodetest[predicate]

Location Path Expression Examples

An XPath expression returns either a node-set, a string, a Boolean, or a number.

Table 14.8 Examples of Location Path Expression

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 element children of the current node
attribute::*	Selects all attributes of the current node
child::text()	Selects all text node children of the current node
child::node()	Selects all children 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

Table 14.8 shows a list of the operators that can be used in XPath expressions.

Table 14.8 XPath Operators

Operator	Description	Example
|	Computes two node-sets	//book | //cd
+	Addition	6 + 4
-	Subtraction	6 – 4
*	Multiplication	6 * 4
Div	Division	8 div 4

=	Equal	price=9.80
!=	Not equal	price!=9.80
<	Less than	price<9.80
<=	Less than or equal to	price<=9.80
>	Greater than	price>9.80
>=	Greater than or equal to	price>=9.80
Or	Or	price=9.80 or price=9.70
And	and	price>9.00 and price<9.90
Mod	Modulus (division remainder)	5 mod 2

XPath Expressions

XPath query string to retrieve information from XML documents is called expression.

XPath expressions are classified as,

- ✓ Sequence Expressions
- ✓ Range Expressions
- ✓ Filter Expressions
- ✓ Arithmetic Expression
- ✓ Comparison Expression
- ✓ Logical Expression
- ✓ For Expression
- ✓ Conditional Expression
- ✓ Quantified Expression

XPath Functions

The XPath language provides a core library of functions that deal with Node set functions for working with node-sets, either the implicit current node set or one passed as a parameter.

String functions are provided for working with strings and include type coercions.

Boolean functions are provided for working with Booleans, including type coercions.

Number functions are provided for working with numbers, including type coercions.

XPath Function Example

Some examples using XPath functions are:

To determine the number of articles written by Mr. Jones:

count(/journal/article[author/last="Jones"])

To find all authors whose last name begins with Mc:

/journal/article[starts-with(author/last,"Mc")]

XPath Node Set Function

Below is the list of some of the XPath Node set functions.

number last() - Returns the index of the last item of the current node set.

Example - /journal/article[last()]

number position() - Returns the index of the current item in the current node set.

Example - /journal/article[position()<3]

number count(node-set) - Returns the number of items in the argument node set.

Example - count(/journal/article)

node-set id(object) - Returns the elements with the ID specified.

Example - id("article.1")/author/last

XPath String Function

Below is the list of some of the XPath string functions.

string string(object?) - Converts an object (possibly the current context node) to its string value.

Example - /journal/article/author[string()='Jones']

string concat(string, string, string*) - Concatenates two or more strings together.

Example - concat(author/salutation, ' ', author/last)

string starts-with(string, string) - Determines if the first argument starts with the second argument string.

Example - /journal/article[starts-with(title, 'Advanced')]

string ends-with(string, string) - Determines if the first argument ends with the second argument string.

Example - /journal/article[ends-with(title, 'Advanced')]

string contains(string, string) - Determines if the first argument contains the second argument string.

Example - /journal/article[contains(title, 'XPath')]

string substring-before(string, string) - Retrieves the substring of the first argument that occurs before the first occurrence of the second argument string.

Example - substring-before(/journal/article[1]/date, '/')

string substring-after(string, string) - Retrieves the substring of the first argument that occurs after the first occurrence of the second argument string.

Example - substring-after(/journal/article[1]/date, '/')

string substring(string, number, number?) - Retrieves the substring of the first argument starting at the index of the second number argument, for the length of the optional third argument.

Example - substring('Jones', 3)

number string-length(string?) - Determines the length of a string, or the current context node coerced to a string.

Example - /journal/article[string-length(author/last) > 9]

string normalize-space(string?) - Retrieves the string argument or context node with all space normalized, trimming white space from the ends and compressing consecutive white space elements to a single space.

Example - normalize-space(/journal/article[1]/content)

XPath Boolean Function

Below is the list of some of the XPath Boolean functions.

Boolean boolean(object) - Converts the argument to a Boolean value.

Example - boolean(/journal/article/author/last[.='Jones'])

Boolean not(boolean) - Negates the boolean value.

Example - not(/journal/article/author/last[.='Jones'])

Boolean boolean-from-string(string) - Returns true if the required parameter string is true, 1, or Yes. In all other conditions, false is returned.

Example - boolean-from-string(../pay_entire_amount)

Boolean true() - The Boolean value is true.

Boolean false() - The Boolean value is false.

object if(boolean,object,object) - Evaluates the first parameter as a Boolean, returning the second parameter when true, otherwise the third parameter.

Example – if(/journal/article/author/last[.='Jones'],'Match found','No match')

XPath Number Function

Below is the list of some of the XPath Number functions.

number number(object?) - Converts the argument or context node to a number value.

Example - /journal[number(year)=2003]

number sum(node-set) - Sums the node set value.

Example - sum(/journal/article/author/age)

number floor(number) - Returns the largest integer that is not greater than the number argument.

Example - floor(100.5)=100

number round(number) - Rounds the number argument.

Example - ceiling(100.3)=100

Summary

This module explains about XPath and how to define XPath expressions to navigate in XML documents.

It also explores about XPath syntax, Location path expression ,Absolute/Relative path, Predicate, Wildcard, Axes and XPath operators and functions.