XPath

Asst. Prof. Dr. Kanda Runapongsa Saikaew
(krunapon@kku.ac.th)
Dept. of Computer Engineering
Khon Kaen University

Overview

- What is XPath?
- Queries
- The XPath Data Model
- Location Paths
- Expressions
- XPath Engines

What is XPath?

- XPath is a language designed to address specific parts of an XML document
- It was designed to be used by both XSLT and XQuery
  - XSLT: transforms an XML document into any text-based format, such as HTML
  - XQuery: a query language for searching data in XML documents

Queries

- XPath is a declarative language for locating nodes in XML documents
- An XPath location path says which nodes from the document you want
- XPath can be thought of a query language like SQL.
- However, rather than extracting information form a database, it extracts information from an XML document

The XPath Data Model (1/2)

- The XPath data model views a document as a tree of nodes
- An instance of XPath language is called an expression
- A path expression is an expression used for selecting a node set by following a path or steps

The XPath Data Model (2/2)

- The particular tree model XPath divides each XML document into seven kinds of nodes
  - root node
  - element node
  - attribute node
  - text node
  - comment node
  - processing instruction node
  - namespace node

XPath and DOM Data Models (1/4)

- The XPath data model is similar to, but not quite the same as the DOM data model
- The most important differences relate to the names and values of nodes
  - In XPath, only attributes, elements, processing instructions, and namespace nodes have names
  - In XPath, the value of an element node is the concatenation of the values of all its text node descendants, not null as it is DOM

XPath and DOM Data Models (2/4)

- For example, the XPath value of Hello is the string Hello and the XPath value of HelloGoodbye is the string HelloGoodbye
- XPath does not have separate nodes for CDATA sections. CDATA sections are simply merged with their surrounding text

XPath and DOM Data Models (3/4)

- XPath does not include any representation of the document type declaration
- All entity references must be resolved before an XPath data model can be built.
- Once entity references are resolved, they are not reported separately from their contents

XPath and DOM Data Models (4/4)

- In XPath, the element that contains an attribute is the parent of that attribute, although the attribute is not a child of the element
- Each XPath text node always contains the maximum contiguous run of text. No text node is adjacent to any other text node

XPath Expressions

- XPath uses path expressions to identify nodes in an XML document
- These path expressions look very much like the expressions you see when you work with a computer file system

  usr/kanda/xmlws/lectures/xpath

Location Paths (1/2)

- Although there are many different kinds of XPath expressions, the one that's of primary use in Java programs is the **location path**
- A location path selects a set of nodes from an XML document
- Each location path is composed of one or more **location steps**

Location Paths (2/2)

- Each location step has an **axis**, a **node test**, and optionally, one or more **predicates**
- Each location step is evaluated with respect to a particular **context node**
- A double colon (::) separates the axis from the node test, and each predicate is
- Syntax for a location path

  axis::node test[predicates]

The Context Node

- Exactly how the context node for a location step is determined depends on the environment in which the location step appears
- In XSLT the context node is normally the currently matched node in the input document

Example: The Context Node

- Let's pick the root **methodCall** element as the context node
- Then **child::methodName** is a location step that selects a node-set containing the single **methodName** element
- That is, it selects all the children of the context node
- **child::params** returns a node-set containing the single **params** element

Axes

- There are twelve axes along which a location step can move.
- Each selects a different subset of nodes in the document, depending on the context node
- An axis selects the tree relationship between the nodes selected by the location step and the current node

Twelve Axes (1/5)

- child: All child nodes of the context node

(Attributes and namespaces are not considered to be children of the node they belong to)

- descendant: All nodes completely contained inside the context node; that is, all child nodes, plus all children of the child nodes, and so forth

Twelve Axes (2/5)

- descendant-or-self: All descendants of the context node and the context node itself
- parent: The node which most immediately contains the context node
- ancestor: The root node and all element nodes that contain the context node

Twelve Axes (3/5)

- ancestor-or-self
  - All ancestors of the context node and the context node itself
- preceding
  - All non-attribute, non-namespace nodes which come before the context node in document order and which are not ancestors of the context node

Twelve Axes (4/5)

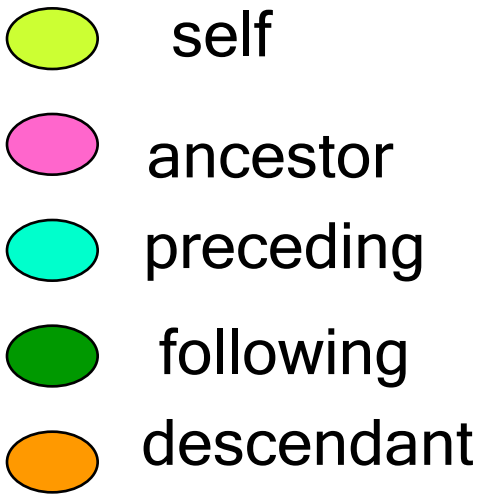
- preceding-sibling
  - All non-attribute, non-namespace nodes which come before the context node in document order and have the same parent node
- following
  - All non-attribute, non-namespace nodes which follow the context node in the document order and which are not descendants of the context node

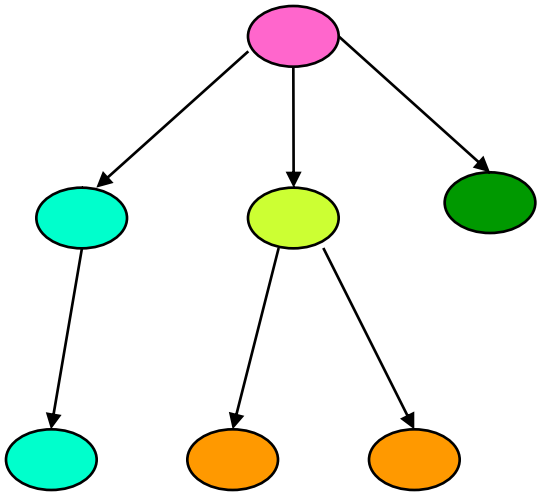
Twelve Axes (5/5)

- following-sibling
  - All non-attribute, non-namespace nodes which follow the context node in document order and have the same parent node
- attribute
  - Attributes of the context node. This axis is empty if the context node is not an element node
- namespace
  - Namespaces in scope of the context node.

Five Axes Cover Everything

- {Ancestor} U {Descendant} U {following} U {preceding} U {self}
  - They do not overlap
  - They together contain all nodes in the document





Node Tests (1/4)

- The axis chooses the direction to move from the context node
- The node test determines what kinds of nodes will be selected along that axis
- Example: child::params
  - child is an axis name
  - params is a note test

Node Tests (2/4)

- name
  - Match any element or attribute with specified name
- *
  - Along the attribute axis the asterisk matches all attribute nodes.
  - Along the namespace axis the asterisk matches all namespace nodes.
  - Along all other axes, this matches all element nodes

Node Tests (3/4)

- prefix:*
  - Match any element or attribute in the namespace mapped to the prefix
- node()
  - Match any node
- text()
  - Match any text node

Node Tests (4/4)

- comment()
  - Match any comment node
- element()
  - Match any element node
- processing-instruction()
  - Match any processing instruction

Predicates

- Each location step can have zero or more predicates that further filter the node-set
- A predicate is an XPath expression in square brackets that is evaluated for each node selected by the location step
- If the predicate is true, then the node is kept in the node-set. Otherwise, it is removed from the node-set

Compound Location Paths

- The forward slash (/) combines location steps into a location path
- The node-set selected by the first step becomes the context node-set for the second step
- The node-set identified by the second step becomes the context node-set for the third step, and so on

Unabbreviated Path Expression Examples (1/2)

- **child::para** selects the para element children of the context node
- **child::*** selects all element children of the context node
- **child::text()** selects all text node children of the context node
- **child::node()** selects all the children of the context nodes

(no attribute nodes are returned)

Unabbreviated Path Expression Examples (2/2)

- **attribute::*** selects all the attributes of the context node
- **parent::node()** selects the parent of the context node.
  - If the context node is an attribute node, this expression returns the element node to which the attribute node is attached
- **descendant::para** selects the para element descendants of the context node

Absolute Location Paths

- Not all location paths require context nodes
- In particular, a location path that begins with a forward slash (/) is an absolute path that starts at the root node of the document
- / selects the root node of the document

Abbreviated Location Paths

- XPath location paths can use the abbreviation in location paths
- The semantics are the same. The syntax is easier to type

Abbreviated Location Paths Example

Abbreviation	Expanded from
Name	child::Name
@Name	attribute::Name
//	/descendant-or-self::node()/
.	self::node()
..	parent::node()

Abbreviated Path Expression Examples

- **child::para** → para
- **child::*** → *
- **child::text()** → text()
- **child::node()** → node()
- **attribute::*** → @*
- **parent::node()** → ..
- **descendant::para** → //para

Combining Location Paths

- Occasionally it's useful to select a node-set that's built from multiple, more or less unrelated parts of an XML document
- We can combine two node-sets into one by using the vertical bar
  - //stk:Price | //stk:Quote

Expressions

- Not all XPath expressions are location paths
- Each XPath 1.0 expression returns one of these four types: string, number, boolean, node-set
- In XPath, a node-set is an unordered collection of nodes from an XML document without any duplicates

Literals (1/2)

- XPath defines literal forms for strings and numbers
- Numbers have more or less the same form as double literals in Java
- XPath does not recognize scientific notation such as 5.5E-10

Literals (2/2)

- XPath string literals are enclosed in single or double quotes
- For example, “red” and ‘red’ are different representations for the same string literal containing the word *red*
- There are no boolean or node-set literals
- However, the true() and false() functions sometimes substitute for the lack of boolean literals

Operators (1/2)

- XPath provides the following operators for basic floating point arithmetic:
  - + addition
  - - subtraction
  - * multiplication
  - div division
  - mod taking the remainder

Operators (2/2)

- < less than
- > greater than
- <= less than or equal to
- >= greater than or equal to
- = boolean equals (not an assignment statement)
- != not equal to
- or Boolean or
- and Boolean and

Functions

- XPath defines a number of useful functions that operate on and return the four fundamental XPath data types
  - Some of these take variable numbers of arguments
  - None of these functions modify their arguments

Node-set Functions

- number last()
  - Returns the number of nodes in the context node list
- number position()
  - Returns the position of the context node list. The first node has position 1, not 0
- number count(node-set)
  - Returns the number of nodes in the argument

Boolean Functions (1/2)

- boolean boolean(object)
  - Converts the argument to a boolean in a mostly sensible way.
  - NaN and 0 are false. All other numbers are true.
  - Empty strings are false. All other strings are true.
  - Empty node-sets are false. All other node-sets are true.

Boolean Functions (2/2)

- boolean not(boolean)
- boolean true()
- boolean false()
- boolean lang(string)
  - This function returns true if the context node is written in the language specified by the argument

String Functions (1/4)

- string string(object?)
  - This function returns the string-value of the argument. If the argument is a node-set, then it returns the string-value of the first node in the set
- string concat(string, string, string …)
  - This function returns a string containing the concatenation of all its arguments

String Functions (2/4)

- boolean starts-with(string, string)
  - This function returns true if the first string starts with the second string. Otherwise it returns false
- boolean contains(string, string)
  - This function returns true if the first string contains the second string. Otherwise it returns false

String Functions (3/4)

- string substring(string, number, number?)
  - This returns the substring of the first argument
    - Beginning at the second argument
    - Continuing for the number of characters specified by the third argument
    - Or until the end of the string if the third argument is omitted

String Functions (4/4)

- number string-length(string?)
  - Returns the number of Unicode characters in the string
- string normalize-space(string?)
  - This function strips all leading and trailing white-space from its argument

Number Functions (1/2)

- number number(object?)
  - This function converts its argument to a number in a reasonable way
- number sum(node-set)
  - Each node in the node-set is converted to a number, and then those numbers are added together

Number Functions (2/2)

- number floor(number)
  - Returns the largest integer less than or equal to the argument
- number ceiling(number)
  - Returns the smallest integer grater than equal to the argument
- number round(number)
  - Returns the integer nearest to the argument

Path Expressions with Functions/Predicates (1/2)

- **child::para[position() = 1]** selects the first **para** child of the context node
- **child::para[position() = last()]** selects the last **para** child of the context node
- **child::chapter[child::title]** selects the **chapter** children of the context node that have one or more **title** children

Path Expressions with Functions/Predicates (2/2)

- **para[1]** selects the first **para** child of the context node
- **para[@type="warning"][5]** selects the fifth **para** child of the context node that a **type** attribute value **warning**
- **para[5][@type="warning"]** selects the fifth **para** child of the context node if that child has a **type** attribute with value **warning**

Comments

- Comments may be used to provide informative annotation for an expression
- Comments are lexical constructs only, and do not affect expression processing
- Comments are strings, delimited by the symbols (: and :)
- An example of a comment

  (: Houston, we have a problem :)

XPath Engines

- XPath Explorer (XPE) is a GUI application that lets you interactively experiment with XPath
- Given an XPath expression and URL (to an HTML or XML document), it displays matching nodes and their values
- This makes it easy to play with and debug your XPath expression
- For more Info
  - http://sourceforge.net/projects/xpe

Summary (1/2)

- XPath is a straightforward declarative language for selecting particular subsets of nodes from an XML document
- XPath location paths are composed of one or more location steps
- Each location step has an axis and a node test, and may have one or more predicates

Summary (2/2)

- Each location step is evaluated with respect to the context nodes determined by the previous step in the path
- The axis determines in which direction you move from the context node
- The node test determines which nodes are selected along the axis
- The predicate decides which of the selected nodes are retained in the set

References

- **XML Path Language (XPath) Version 1.0** http://www.w3.org/TR/xpath
- W3Schools XPath Tutorial

  http://www.w3schools.com/xpath/default.asp
- Zvon XPath Tutorial

http://www.zvon.org/xxl/XPathTutorial/General/examples.html