Processing of structured documents

Spring 2003, Part 7
Helena Ahonen-Myka

XML Path Language (XPath)

- The ability to navigate through XML documents is needed in many applications of XML
  - querying of XML documents
  - creation of hypertext links to objects that do not have unique identifiers
  - formatting of document components for presentation

XML Path Language (XPath)

- XPath provides
  - common syntax and semantics to address parts of an XML document
  - basic facilities for manipulation of strings, numbers and booleans
- XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs and XML attribute values

XML Path Language (XPath)

- Use e.g. as a pattern in XSLT:

```
<xsl:template match="chapter/title">
…
</xsl:template>
```


XML Path Language (XPath)

- XPath operates on an XML document as a tree
- every element in an XML document has a specific and unique contextual location
  - any element in the document can be identified by the steps it would take to reach it, either from the root element, or from some other fixed starting location

Expressions

- The primary syntactic construct in XPath is the expression
- an expression is evaluated to yield an object, which has one of the following types
  - node-set (unordered)
  - boolean (true or false)
  - number
  - string
  - (in XPath 2.0 also more types)

Location paths

- relative location paths
  - a path that starts from an existing location
  - sequence of one or more location steps separated by /
  - steps are composed from left to right
  - the initial step selects a set of nodes relative to the context node
  - each node in this set is used as a context node for the following step
  - the sets of nodes identified by that step are unioned together
  - e.g. child:name/child:fname

7

Location paths

- An absolute location path consists of / optionally followed by a relative location path
- A / by itself selects the root node of the document
- if / is followed by a relative path, then the location path selects the set of nodes that would be selected by the relative location path relative to the root node

8

Location steps

- A location step has three parts
  - an axis: the tree relationship between the nodes selected by the location step and the context node
  - a node test: the node type and name of the nodes selected by the location step
  - zero or more predicates, which use arbitrary expressions to further refine the set of nodes selected by the location step
- syntax:
  - axis::node-test[expr][expr]...
  - e.g. child::para[position()=1]

9

Location steps

- The node-set selected by the location step is the node-set that results from
  - generating an initial node-set from the axis and node-test, and then
  - filtering that node-set by each of the predicates in turn
- the initial node-set consists of the nodes
  - having the relationship to the context node specified by the axis, and
  - having the node type and name specified by the node test

10

Axes

- child
- descendant
- parent
- ancestor
- following-sibling
  - empty, if the context node is an attribute node or namespace node
- preceding-sibling
  - empty, if the context node is an attribute node or namespace node

11

Axes

- following
  - all nodes in the same document as the context node that are after the context node in document order, excluding any descendants and excluding attribute nodes and namespace nodes
- preceding
  - all nodes in the same document as the context node that are before the context node in document order, excluding any ancestors and excluding attribute nodes and namespace nodes

12

Axes

- attribute
  - attribute nodes of the context node
  - empty unless the context node is an element
- namespace
  - namespace nodes of the context node
  - empty unless the context node is an element
- self
  - the context node itself
- descendant-or-self, ancestor-or-self

13

Axes

- The ancestor, descendant, following, preceding and self axes partition a document (ignoring attribute and namespace nodes): they do not overlap and together they contain all the nodes in the document

14

Node tests

- Every axis has a principal node type
  - for the attribute axis: attribute
  - for the namespace axis: name space
  - for other axes: element
- a node test
  - both name and type have to match
  - child::para
    - selects the para element children of the context node
    - if the context node has no para children, it will select an empty set of nodes

15

Node tests

- node test node() represents any node
- node tests text(), comment(), and processing-instruction() represent any object of these specific types
- a node test * is true for any node of the principal node type
  - child::*
    - selects all element children of the context node
  - attribute::*
    - selects all attributes of the context node

16

Abbreviated syntax

- child:: -> can be omitted from a location step; child is the default axis
  - child::div/child::para -> div/para
- attribute:: -> @
  - child::para[attribute::type="warning"] -> para[@type="warning"]
- /descendant-or-self::node()/ -> //
  - //para selects any para element in the document
  - div//para selects all para descendants of div children (of the context node)

17

Abbreviated syntax

- self::node() -> . (fullstop)
  - .//para selects all para descendant elements of the context node
- parent::node() -> ..
  - ../title selects the title children of the parent of the context node

18

Predicates

- An axis is either a forward axis or a reverse axis
  - forward axis: an axis that only ever contains the context node or nodes that are after the context node in document order
  - reverse axis: an axis that only ever contains the context node or nodes that are before the context node in document order

19

Predicates

- the proximity position of a member of a node-set with respect to an axis:
  - the position of the node in the node-set ordered in
    - document order if the axis is a forward axis
    - reverse order if the axis is a reverse axis
  - the first position is 1
- a predicate filters a node-set to produce a new node-set
  - for each node in the node-set, the predicate expression is evaluated with that node as the context node and with the proximity position of the node in the node-set

20

Predicates

- If the predicate expression evaluates to true for that node, the node is included in the new node-set
- the result of the evaluation is converted to a boolean
  - if the result is a number, the result is true if the number is equal to the context position
  - otherwise, the result will be converted as if by a call to the function **boolean** (see below)
  - e.g. para[3] equals para[position()=3]

21

Predicates

- Contained element tests
  - the name of an element can appear in a predicate filter -> represents an element that must be present as a child
  - note[title]
    - a note element is only selected if it directly contains a title element
  - note[title="first note"]
    - true, if the content of the element is 'first note'
  - note[id("123")]

22

Predicates

- Attribute tests
  - para[@type='secret']
    - every 'para' element with a 'type' attribute value of 'secret'

23

Expressions

- boolean operators: or, and
- comparisons: =, !=, <=, <, >=, >
  - in XML documents: < has to be converted to <
- numeric operators: +, -, *, div, mod

24

Core functions

- some functions
  - number last()
  - number position()
  - number count(node-set)
  - node-set id(object)
    - e.g. id("foo") selects the element with unique ID foo
  - boolean starts-with(string, string)
    - returns true if the first string starts with the second
  - boolean contains(string, string)
    - returns true if the first string contains the second

25

Examples

- para selects the para element children of the context node
- * selects all element children
- text() selects all text node children
- @name selects the name attribute
- @* selects all the attributes
- para[1] selects the first para child
- para[last()] selects the last para child
- */para selects all para grandchildren
- /doc/chapter[5]/section[2] selects the second section of the fifth chapter of the doc (root)

26

Examples

- chapter//para selects the para element descendants of the chapter element children
- //para selects all the para descendants of the document root and thus selects all the para elements in the same document as the context node
- //olist/item selects all the item elements in the same document as the context node that have an olist parent
- . selects the context node
- .//para selects the para element descendants
- .. selects the parent
- ../@lang selects the lang attribute of the parent

27

Examples

- para[@type="warning"] selects all para children of the context node that have a type attribute with value warning
- para[@type="warning"][5] selects the fifth para child of the context node that has a type attribute with 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

28

Examples

- chapter[title="Introduction"] selects the chapter children of the context node that have one or more title children with string-value equal to Introduction
- chapter[title] selects the chapter children of the context node that have one or more title children
- employee[@secretary and @assistant] selects all the employee children of the context node that have both a secretary attribute and an assistant attribute

29