XPath 1.0 Tutorial

28th of May, 2002

Dan Olteanu
XPath 1.0 - W3C Recommendation

- language for addressing and matching parts of an XML document.

- designed to be used standalone, but also by XPointer, XSLT, XQuery.

- uses a compact, non-XML format to facilitate use within URIs and attribute values.

- provides basic facilities for string, number and boolean manipulation.

- supports namespaces.
**XPath 1.0 Data Model**

- can be derived from **XML InfoSet**.
- XML document is viewed as a **tree**, containing different kinds of nodes.
- kinds of nodes: root, element, text, attribute, namespace, processing instruction, comment nodes.
- imposes a document **order** defined on all nodes except attribute and namespace nodes (order of occurrence of element start-tags).
- the root node is the **first** node.
- the namespace nodes are defined to occur **before** the attribute nodes.
- root and element nodes have an **ordered list** of children.
- an element node is the **parent** of the associated **set** of attribute/namespace nodes, the attributes/namespaces are **not children** of the associated element node.
Various Fruits

The next chapters introduce different kinds of fruits, like strawberries or apples.

Strawberries

Strawberige; straw + berie berry; perhaps from the resemblance of the runners of the plant to straws.

A fragrant edible berry, of a delicious taste and commonly of a red colour.

The common American strawberry is Fragaria virginiana, the European is Fragaria vesca.
Tree Representation for the XML fragment example, cf. XPath 1.0 Data Model
XPath Expressions: Syntactic Building Blocks

- **primary** expressions: strings, numbers, booleans, location paths, predicates, function calls, variable reference.

- **complex** expressions: unions, filters, relational expressions.

- basic expression **types**: string, number, boolean, node-set.

- expression **evaluation** done in a context, consisting of:
  - the context **node**.
  - the context **position** and **size**.
  - a set of variable bindings.
  - a function library.
  - a set of namespace declarations in expression scope.

- examples
  - 'Markup-Sprachen und semi-strukturierte Daten'
  - 
  - $(x + y) \times 2 > 10.7$
  - 
  - //lecture[@name = 'Markup' and contains(author,'Bry')]
A path is constructed from steps, which have:

- an **axis**, which specifies the tree relationship between the nodes.
- a **node** test, which specifies the name of the selected nodes.
- zero or more **predicates**, which refine the set of nodes selected by the location step.

Examples:

- `child::para`
- `child::figure[attribute::id="fr_vesca"]`
- `child::*[position()=last()]`

Abbreviated syntax:

- `@name` for `attribute::name`
- `para[1]` for `child::para[position()=1]`
- `./para` for `self::node()/descendant-or-self::node()/child::para`
- `.` for `self::node()`
- `..//para` for `parent::node()/child::para`
We use:

- a graphical test environment for XPath expressions, named XPath Tester.

- Command line for XPath Tester 1.1: `java -jar xpathtester_1_1.jar`.


- Prerequisites:
  - Java Virtual Machine, e.g. jdk 1.2 or higher.
  - a Java-based XPath evaluator, e.g. Xalan, Saxon, XT.
  - a Java-based XML parser, e.g. Xerces, XML4Java, Alfred.
Core Function Library: Node-Set Functions

- `number last()`
  returns the context size from the expression evaluation context.

- `number position()`
  returns the context position from the expression evaluation context.

- `number count(node-set)`
  returns the number of nodes in the argument node-set.

- `node-set id(object)`
  selects elements by their unique ID, as declared in DTD.

- `string name(node-set?), string local-name(node-set?), string namespace-uri(node-set?)`
  returns the expanded/local name/namespace URI of the node in the argument node-set that is first in document order.
Example Break (2) - Testing Node-Set Functions

We use:

- a graphical test environment for XPath expressions, named XPath Tester.
- Command line for XPath Tester 1.1: `java -jar xpathtester_1_1.jar`.
- Prerequisites:
  - Java Virtual Machine, e.g. jdk 1.2 or higher.
  - a Java-based XPath evaluator, e.g. Xalan, Saxon, XT.
  - a Java-based XML parser, e.g. Xerces, XML4Java, Alfred.
Core Function Library: String Functions

- `string string(object?)`
- `string concat(string, string, string*)`
- `boolean start-with(string, string)`
- `boolean contains(string, string)`
- `string substring-before(string, string)`
- `string substring-after(string, string)`
- `string substring(string, number, number?)`
- `number string-length(string?)`
- `string normalize-space(string?)`
- `string translate(string, string, string)`
Example Break (3) - Testing String Functions

We use:

- a graphical test environment for XPath expressions, named XPath Tester.
- Command line for XPath Tester 1.1: `java -jar xpathtester_1_1.jar`.
- Prerequisites:
  - Java Virtual Machine, e.g. jdk 1.2 or higher.
  - a Java-based XPath evaluator, e.g. Xalan, Saxon, XT.
  - a Java-based XML parser, e.g. Xerces, XML4Java, Alfred.
Core Function Library: Boolean & Number Functions

Boolean Functions

- boolean boolean(object)
- boolean not(boolean)
- boolean true()
- boolean false()

Number Functions

- number number(object?)
- number sum(node-set?)
- number floor(number)
- number ceiling(number)
- number round(number)
Example Break (4) - Testing Boolean & Number Functions

We use:

- a graphical test environment for XPath expressions, named XPath Tester.
- Command line for XPath Tester 1.1: java -jar xpathtester_1_1.jar.
- downloadable from 5Sight:
- Prerequisites:
  - Java Virtual Machine, e.g. jdk 1.2 or higher.
  - a Java-based XPath evaluator, e.g. Xalan, Saxon, XT.
  - a Java-based XML parser, e.g. Xerces, XML4Java, Alfred.
More Examples (1)

- Select the *figure* elements without attributes:
  \[//\text{figure}[\text{not}(@*)]\]

- Select the *chapters* having three *paragraphs*:
  \[//\text{chapter}[\text{count}(./\text{para}) = 3]\]

- Select the first *paragraph* of each *chapter*:
  \[//\text{chapter}//\text{para}[1]\]

- Select the first *paragraph* of all *chapters*:
  \[(//\text{chapter}//\text{para})[1]\]

- Select the *figures* with an attribute *caption* 'Fragaria virginiana' from the second *chapter*:
  \[//\text{chapter}[2]//\text{figure}[\text{caption} = \text{'}Fragaria virginiana\text{'}]\]

- Select the *figures* in *chapters* 2 through 5:
  \[//\text{chapter}[\text{position()} \geq 2 \text{ and position()} \leq 5]//\text{figure}\]

- Select *captions* of *figures* that are referenced by *figref* attributes of *fruit* elements in the first *chapter*:
  \[\text{id}(//\text{chapter}[1]//\text{fruit}@\text{figref})[\text{self::figure}]\text{/caption}\]
More Examples (2)

- Select *chapters* in which the word 'Strawberry' is mentioned in at least one *paragraph*:
  
  \[
  \text{//chapter[//para[contains(.,'Strawberry')]]}
  \]

- Select *chapters* in which the word 'Strawberry' is mentioned in every *paragraph*:
  
  \[
  \text{//chapter[count(./para) = count(./para[contains(.,'Strawberry')]) and ./para]}
  \]
  
  OR
  
  \[
  \text{//chapter[not(./para[not(contains(.,'Strawberry')]])] and ./para]
  \]

- List the *names* of the second-level *managers* of all *employees* whose *rating* is 'Good':
  
  \[
  \text{id(id(/emp[rating = "Good"]/@mgr)[self::emp]/@mgr)[self::emp]/name}
  \]

- List all distinct *employees* from a *company*:
  
  \[
  \text{(//company//employee)[not(.=preceding::employee)]}
  \]

- Prepare a critical sequence report consisting of all elements that occur between the first and second *incision*:
  
  \[
  \text{(//incision[2]/preceding::*)[count( . | (//incision[1]/following::*)) = count (//incision[1]/following::*))}
  \]
What’s coming with XPath 2.0?

- support for XML Schema primitive datatypes.
- explicit For Any and For All quantifiers (some and every).
- FR construct from XQuery FLoWeR expression (for-return).
- extended set of aggregation functions (e.g. min, max, avg ...).
- conditional expressions (if-then-else).
- node-set intersection and difference functions (intersect, except).
- string matching using regular expressions.
Useful links

- W3C XPath Recommendation:
  http://www.w3.org/TR/xpath.

- XPath Tutorial from Zvon:

- XPath Tester from 5Sight:

- XPath Tester from PhPXXML:

- XPath Tutorial from Resin:

- XPath Implementation from Xalan: