XPath

Introduction

XPath is a language for addressing parts of XML documents. It has been designed within the World Wide Web Consortium (W3C) and has a status of a *W3C Recommendation* (Version 1.0, a de-facto standard).

Complete XPath Recommendation can be found here:

http://www.w3.org/TR/xpath

XPath Essentials

XPath is designed to *address well-formed parts* of XML documents. It is designed to be used in URIs (Universal Resource Identifiers), and therefore, it has a non-XML syntax.

XPath gives two ways of expressing paths: *full syntax* and *abbreviated syntax.* In *full syntax*, an XPath *expression* consists of a sequence of *location steps* separated by a "/" (slash). The basic anatomy of an XPath expression, and of a location step is shown below:

```
XPath ::= [/] LocationStep [/ LocationStep]*
LocationStep ::= AxisName '::'  [NodeTest] '[' [Predicate]']'
```

Here:

AxisName: is the name of the type of a "one-step" traversal through the XML (DOM) tree.

NodeTest: describes the names of the XML elements of interest on current step.

Predicate: specifies additional *selection conditions* on the nodes for the next step.

Axes

XPath expressions describe the *traversal* of the XML tree. An XPath *axis* is one step in the traversal. A *context node* is a node in the XML (DOM) tree that is current on the path.

	Axis	Meaning
1	`ancestor`	proceed to the ancestors of context node(s)
2	`ancestor-or-self`	`ancestor` or `self`
3	`attribute`	proceed to the attributes of context node(s)
4	`child`	proceed to children of context node(s)
5	`descendant`	proceed to descendants of context node in the DOM tree
6	`descendant-or-self`	`descendent` or `self`
7	`following`	proceed to nodes that follow context node in XML document order
8	`following-sibling`	proceed to siblings that follow context node in XML document order
9	`parent`	proceed to the parent of context node(s)
10	`preceding`	proceed to nodes that preceed context node in XML document order
11	`preceding-sibling`	proceed to siblings that follow context node in XML document order
12	`self`	stay at context node

Node Tests

Each axis in XPath tree has a principal type:

- `attribute` axis has type `attribute`;
- all other axes have type `element`;

```
NodeTest ::= '*' | Name | NodeType'()'
```

`*` - matches all nodes.

`Name` - matches all nodes with given name.

`NodeType()` - matches all nodes of a specific type.

Types:

Type	Explanation
`node()`	matches all nodes
`element()`	matches element nodes
`attribute()`	matches attribute nodes
`text()`	matches text nodes
`comment()`	matches comment nodes

Predicates

Predicates specify extra conditions on selection of nodes during the current traversal step. Predicates are built around *core functions*.

Functions:

Type	FunctionName	Explanation
number	`last()`	size of the context
number	`position()`	position of the context (in the list of siblings)
number	`count()`	number of children of the node
node-set	`id()`	value of the ID attribute
string	`string()`	converts argument into an string
boolean	`contains(.,.)`	rue if first argument contains the second
string	`substring(string,number,number)`	selects the substring
number	`string-length(string)`	returns the length of the string
number	`floor`,`ceiling`,`round`	standard meaning

Note: for more functions, consult the W3C Recommendation.

Predicates can be built using standard expression techniques, comparison operators (`>`,`<`, `=`, `!=`, $\geq$, $\leq$), logical operators (`and`, `or`, `not`) and simple arithmetics operators ($+$, $-$, $*$, `div`, `mod`).

Relative Path vs. Absolute Path

XPath expressions are either *relative* or *absolute*.

relative: Expression states that the traversal starts at current node.
Syntax: `LocationStep [/ LocationStep]*`

absolute: Expression states that the traversal starts at the root of the XML document.
Syntax: `/LocationStep [/ LocationStep]*`

Document Order

Document order is defined on all the nodes in the document. It is the order in which the first character of the XML representation of each node occurs in the XML representation of the document. Thus, the root node will be the first node. Element nodes occur before their children. Thus, **document order** orders element nodes in order of the occurrence of their start-tag in the XML.

Examples

Notation: *context node*: node of the XML tree, currently being "observed" by an XPath expression.

XPath Expression	Meaning
`/self::node()/child::element()`	Find all children of the root that are element nodes.
`self::car`	Select current (context) node if it is `car`.
`/descendant::name/child::first`	Find all `first` elements, that are children of `name` elements.
`parent::node()/parent::node()`	Find the "grandparent" of the context node
`/descendant::name/following-sibling::address`	Find all `address` elements that have as prior siblings `name` elements.
`/descendant::attribute()`	Find all attribute nodes.
`preceeding::car`	Find all `car` element preceeding (in document order) the context node.
`descendant-or-self::element()`	Find all element nodes in the subtree of the context node (including itself)
`child::car[position()=3]`	Select the third `car` child of the context node
`child::*[self::car or self::person]`	Select all `person` and `car` children of the context node
`child::name[child::number="2"]`	Select `name` children of context node, if it has at least one `number` child with content "2"
`/descendant::car/self::*[attribute::maker="US"]`	Find all `car` nodes that have an attribute `maker` with value "US"

Semantics

The semantics of XPath nodes is given in terms of *node sets*. A *node set* is any collection of nodes in the DOM tree. Given a node set and an XPath location step, a new node set is determined according to the semantics of the axes, nodetests and predicates (see descriptions above) by applying the location step to each node from the node set individually and taking the union of all "reached" nodes.

Absolute XPath expressions start at the root node (i.e., input context node set is a set containing a single node, the root).

Relative XPath expressions assume that there is a context node set and operate on it.

Abbreviated Syntax

For simplicity some axes, node tests and predicates can be abbreviated.

Abbreviated XPath Expression	XPath Expression
`car`	`child::car`
`*`	`child::element()`
`text()`	`child::text()`
`@maker`	`attribute::maker`
`@*`	`attribute::*`
`x//y`	`child::x/descendant::y`
`.`	`self::node()`
`..`	`parent::node()`
`../@maker`	`parent::node()/attribute::maker`
`car[5]`	`child::car[position()=5]`
`car[@maker="US"]`	`child::car/self::node()[maker="US"]`
`car[milage]`	`child::car/self::node()[child::milage]`

More complex examples:

`//car/../../*[5]`: Select all nodes, that are a fifth child of a "grandparent of a `car` node.

`/*/*/*[@*]`: Select all "great grandchildren" of the root that have attributes.

`..//`: select all descendants of the parent of the context node.

`../*`: select all element siblings of the context node.