XPath

Overview

XPath is a language that allows to navigate and find the nodes within an XML document by a variety of criteria, exploiting the tree like representation of the document itself.

Syntax

The most common kind of expression is the **location path**, which consists of **location steps**.
An XPath expression can be performed with respect to a **context node**, the node the XPath processor is currently looking at (which is obviously going to change as long as the XPath processor evaluates the query). Passing a document to the processor, the root node is the initial context node.
Each location step has three components:

Components	Description
Axis	It indicates the direction in which the XML document tree can be navigated: more specifically a list of nodes in relation to the context node.
Node test	It allows to control specific node names or more general expression, selecting only some particular nodes from the tree. For instance, if there's an XML document in which has been defined the prefix of the namespace `pkm`, `//pkm:attack` will find all the nodes `attack` of that namespace
Predicate	General expressions written in square brackets that can be used to filter a node-set according to some condition

Syntax Types

- *Abbreviated*: it's compact and allows XPath to be written and read easily using intuitive characters and constructs
- *Full*: it's more verbose, but allows for more options to be specified
- Example: Find all the team's pokémons
  - Full syntax: `/child::Team/child::Pokemon`
  - Abbreviated: `/Team/Pokemon`

General Syntax

Every location step presents the following syntax:

```
axisname::node_test[predicate]
```

Available Axis

Full Syntax	Abbreviated
child	<child_name>
attribute	@<attr_name>
descendant	N/A
descendant-or-self::node()/	//
parent	..
ancestor	N/A
ancestor-or-self	N/A
following	N/A
preceding	N/A
following-sibling	N/A
preceding-sibling	N/A
self	.
namespace	N/A

Available Node tests (in addition to the node names and wildcards)

Node test formats	Description
comment()	Finds a comment node (e.g. <!--xx-->)
text()	Finds a text node (e.g. BULBASAUR in <species>BULBASAUR</species>)
processing-instruction()	Finds XML processing instructions (e.g. <?php echo "pikachu";?>)
node()	Finds a node of any type

Operators

Operator	Description
/, //, [...]	Used in path expressions
|	Union of two node-set
and, or, not()	Boolean operators
+, -, *, div, mod	Arithmetic operators
=, !=, <, >, <=, >=	Comparing operators

Common functions

Function	Description
`position()`	returns a number representing the position of this node in the sequence of nodes currently being processed
`count(node-set)`	returns the number of nodes in the node-set supplied as its argument.
`string(object?)`	converts any of the four XPath data types into a string according to built-in rules.
`concat(string1,...,stringN)`	concatenates two or more strings
`starts-with(s1, s2)`	returns true if s1 starts with s2
`contains(s1, s2)`	returns true if s1 contains s2
`substring(string, start, length?)`	e.g. `substring("pikachu", 1, 4) = "pika"`
`substring-before(s1, s2)`	e.g. `substring-before("pika-ch-u, "-") = "pika"`
`substring-after(s1, s2)`	e.g. `substring-after("pika-ch-u, "-") = "ch-u"`
`string-length(string?)`	returns number of characters in string
`normalize-space(string?)`	all leading and trailing whitespace is removed and any sequences of whitespace characters are replaced by a single space
`not(boolean)`	negates any boolean expression
`true`	evaluates to true
`false`	evaluates to false
`sum(node-set)`	converts the string values of all the nodes found by the XPath argument into numbers, then returns the sum of these nodes

References

If you're interested in more examples take a look at the xPath documentation here https://www.w3.org/TR/2017/REC-xpath-31-20170321/