NAME

XML::XPathEngine − a re−usable XPath engine for DOM−like trees

DESCRIPTION

This module provides an XPath engine, that can be re-used by other module/classes that implement trees.

In order to use the XPath engine, nodes in the user module need to mimic DOM nodes. The degree of similitude between the user tree and a DOM dictates how much of the XPath features can be used. A module implementing all of the DOM should be able to use this module very easily (you might need to add the cmp method on nodes in order to get ordered result sets).

This code is a more or less direct copy of the XML::XPath module by Matt Sergeant. I only removed the XML processing part to remove the dependency on XML::Parser, applied a couple of patches, renamed a whole lot of methods to make Pod::Coverage happy, and changed the docs.

The article eXtending XML XPath, http://www.xmltwig.com/article/extending_xml_xpath/ should give authors who want to use this module enough background to do so.

Otherwise, my email is below ;−−)

WARNING: while the underlying code is rather solid, this module mostly lacks docs. As they say, ‘‘patches welcome’’...

SYNOPSIS

```
    use XML::XPathEngine;

    my $tree= my_tree->new( ...);
    my $xp = XML::XPathEngine->new();

    my @nodeset = $xp->find('/root/kid/grandkid[1]', $tree); # find all first granki

    package XML::MyTree;

    # needs to provide DOM methods
```

DETAILS

API

XML::XPathEngine will provide the following methods:

new

findnodes ($path, `$context`**)**

Returns a list of nodes found by `$path`, optionally in context `$context`. In scalar context returns an XML::XPathEngine::NodeSet object.

findnodes_as_string ($path, `$context`**)**

Returns the nodes found as a single string. The result is not guaranteed to be valid XML though (it could for example be just text if the query returns attribute values).

findnodes_as_strings ($path, `$context`**)**

Returns the nodes found as a list of strings, one per node found.

findvalue ($path, `$context`**)**

Returns the result as a string (the concatenation of the values of the result nodes).

findvalues($path, `$context`**)**

Returns the values of the result nodes as a list of strings.

exists ($path, `$context`**)**

Returns true if the given path exists.

matches($node, `$path`, `$context`**)**

Returns true if the node matches the path.

find ($path, `$context`**)**

The find function takes an XPath expression (a string) and returns either a XML::XPathEngine::NodeSet object containing the nodes it found (or empty if no nodes matched the path), or one of XML::XPathEngine::Literal (a string), XML::XPathEngine::Number, or XML::XPathEngine::Boolean. It should always return something – and you can use −>*isa()* to find out what it returned. If you need to check how many nodes it found you should check `$nodeset`−>size. See XML::XPathEngine::NodeSet.

getNodeText ($path)

Returns the text string for a particular node. Returns a string, or undef if the node doesn't exist.

set_namespace ($prefix, `$uri`**)**

Sets the namespace prefix mapping to the uri.

Normally in XML::XPathEngine the prefixes in XPath node tests take their context from the current node. This means that foo:bar will always match an element <foo:bar> regardless of the namespace that the prefix foo is mapped to (which might even change within the document, resulting in unexpected results). In order to make prefixes in XPath node tests actually map to a real URI, you need to enable that via a call to the set_namespace method of your XML::XPathEngine object.

clear_namespaces ()

Clears all previously set namespace mappings.

get_namespace ($prefix, `$node`**)**

Returns the uri associated to the prefix for the node (mostly for internal usage)

set_strict_namespaces ($strict)

By default, for historical as well as convenience reasons, XML::XPathEngine has a slightly non-standard way of dealing with the default namespace.

If you search for `//tag` it will return elements `tag`. As far as I understand it, if the document has a default namespace, this should not return anything. You would have to first do a `set_namespace`, and then search using the namespace.

Passing a true value to `set_strict_namespaces` will activate this behaviour, passing a false value will return it to its default behaviour.

set_var ($var. `$val`**)**

Sets an XPath variable (that can be used in queries as `$var`)

get_var ($var)

Returns the value of the XPath variable (mostly for internal usage)

`$XML::XPathEngine::Namespaces`

Set this to 0 if you *don't* want namespace processing to occur. This will make everything a little (tiny) bit faster, but you'll suffer for it, probably.

Node Object Model

Nodes need to provide the same API as nodes in XML::XPath (at least the access API, not the tree manipulation one).

Example

Please see the test files in t/ for examples on how to use XPath.

XPath extension

The module supports the XPath recommendation to the same extend as XML::XPath (that is, rather completely).

It includes a perl-specific extension: direct support for regular expressions.

You can use the usual (in Perl!) `=~` and `!~` operators. Regular expressions are / delimited (no other delimiter is accepted, \ inside regexp must be backslashed), the `imsx` modifiers can be used.

```
  $xp->findnodes( '//@att[.=~ /^v.$/]'); # returns the list of attributes att
                                         # whose value matches ^v.$
```

SEE ALSO

XML::XPath

HTML::TreeBuilder::XPath, XML::Twig::XPath for examples of using this module

Tree::XPathEngine for a similar module for non-XML trees.

<http://www.xmltwig.com/article/extending_xml_xpath/> for background information. The last section of the article summarizes how to reuse XML::XPath. As XML::XPathEngine offers the same API it should help you

AUTHOR

Michel Rodriguez, `<mirod@cpan.org>` Most code comes directly from XML::XPath, by Matt Sergeant.

BUGS

Please report any bugs or feature requests to `bug-tree-xpathengine@rt.cpan.org`, or through the web interface at <http://rt.cpan.org/NoAuth/ReportBug.html?Queue=XML-XPathEngine>. I will be notified, and then you'll automatically be notified of progress on your bug as I make changes.

ACKNOWLEDGEMENTS

COPYRIGHT & LICENSE