Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 22.4.5 Manual Pages on command 'XML::NamespaceSupport.3pm'

$ man XML::NamespaceSupport.3pm

XML::NamespaceSupport(3pm) User Contributed Perl Documentation XML::NamespaceSupport(3pm)

NAME

XML::NamespaceSupport - A simple generic namespace processor

VERSION

version 1.12

SYNOPSIS

```
use XML::NamespaceSupport;
my $nsup = XML::NamespaceSupport->new;

# add a new empty context
$nsup->push_context;
# declare a few prefixes
$nsup->declare_prefix($prefix1, $uri1);
$nsup->declare_prefix($prefix2, $uri2);
# the same shorter
$nsup->declare_prefixes($prefix1 => $uri1, $prefix2 => $uri2);

# get a single prefix for a URI (randomly)
$prefix = $nsup->get_prefix($uri);
```

```
# get all prefixes for a URI (probably better)
@prefixes = $nsup->get_prefixes($uri);
# get all prefixes in scope
@prefixes = $nsup->get_prefixes();
# get all prefixes that were declared for the current scope
@prefixes = $nsup->get_declared_prefixes;
# get a URI for a given prefix
$uri = $nsup->get_uri($prefix);

# get info on a qname (java-ish way, it's a bit weird)
($ns_uri, $local_name, $qname) = $nsup->process_name($qname, $is_attr);
# the same, more perlish
($ns_uri, $prefix, $local_name) = $nsup->process_element_name($qname);
($ns_uri, $prefix, $local_name) = $nsup->process_attribute_name($qname);

# remove the current context
$nsup->pop_context;

# reset the object for reuse in another document
$nsup->reset;

# a simple helper to process Clarkian Notation
my ($ns, $lname) = $nsup->parse_jclark_notation('{http://foo}bar');
# or (given that it doesn't care about the object
my ($ns, $lname) = XML::NamespaceSupport->parse_jclark_notation('{http://foo}bar');
```

DESCRIPTION

This module offers a simple to process namespaced XML names (unames) from within any application that may need them. It also helps maintain a prefix to namespace URI map, and provides a number of basic checks.

The model for this module is SAX2's NamespaceSupport class, readable at http://www.saxproject.org/namespaces.html It adds a few perlisations where we

thought it appropriate.

NAME

XML::NamespaceSupport - a simple generic namespace support class

METHODS

? XML::NamespaceSupport->new(\%options)

A simple constructor.

The options are "xmlns", "fatal_errors", and "auto_prefix"

If "xmlns" is turned on (it is off by default) the mapping from the xmlns prefix to the URI defined for it in DOM level 2 is added to the list of predefined mappings (which normally only contains the xml prefix mapping).

If "fatal_errors" is turned off (it is on by default) a number of validity errors will simply be flagged as failures, instead of die()ing.

If "auto_prefix" is turned on (it is off by default) when one provides a prefix of "undef" to "declare_prefix" it will generate a random prefix mapped to that namespace. Otherwise an undef prefix will trigger a warning (you should probably know what you're doing if you turn this option on).

If "xmlns_11" us turned off, it becomes illegal to undeclare namespace prefixes. It is on by default. This behaviour is compliant with Namespaces in XML 1.1, turning it off reverts you to version 1.0.

? $nsup->push_context

Adds a new empty context to the stack. You can then populate it with new prefixes defined at this level.

? $nsup->pop_context

Removes the topmost context in the stack and reverts to the previous one. It will die() if you try to pop more than you have pushed.

? $nsup->declare_prefix($prefix, $uri)

Declares a mapping of $prefix to $uri, at the current level.

Note that with "auto_prefix" turned on, if you declare a prefix mapping in which $prefix is undef(), you will get an automatic prefix selected for you. If it is off you will get a warning.

This is useful when you deal with code that hasn't kept prefixes around and need to reserialize the nodes. It also means that if you want to set the default namespace (i.e. with an empty prefix) you must use the empty string instead of undef. This behaviour is consistent with the SAX 2.0 specification.

? $nsup->declare_prefixes(%prefixes2uris)

Declares a mapping of several prefixes to URIs, at the current level.

? $nsup->get_prefix($uri)

Returns a prefix given a URI. Note that as several prefixes may be mapped to the same URI, it returns an arbitrary one. It'll return undef on failure.

? $nsup->get_prefixes($uri)

Returns an array of prefixes given a URI. It'll return all the prefixes if the uri is undef.

? $nsup->get_declared_prefixes

Returns an array of all the prefixes that have been declared within this context, ie those that were declared on the last element, not those that were declared above and are simply in scope.

Note that at least one context must be added to the stack via "push_context" before this method can be called.

? $nsup->get_uri($prefix)

Returns a URI for a given prefix. Returns undef on failure.

? $nsup->process_name($qname, $is_attr)

Given a qualified name and a boolean indicating whether this is an attribute or another type of name (those are differently affected by default namespaces), it returns a namespace URI, local name, qualified name tuple. I know that that is a rather abnormal list to return, but it is so for compatibility with the Java spec. See below for more Perlish alternatives.

If the prefix is not declared, or if the name is not valid, it'll either die or return undef depending on the current setting of "fatal_errors".

? $nsup->undeclare_prefix($prefix);

Removes a namespace prefix from the current context. This function may be used in SAX's end_prefix_mapping when there is fear that a namespace declaration might be available outside their scope (which shouldn't normally happen, but you never know ;)). This may be needed in order to properly support Namespace 1.1.

? $nsup->process_element_name($qname)

Given a qualified name, it returns a namespace URI, prefix, and local name tuple. This method applies to element names.

If the prefix is not declared, or if the name is not valid, it'll either die or return undef depending on the current setting of "fatal_errors".

? $nsup->process_attribute_name($qname)

Given a qualified name, it returns a namespace URI, prefix, and local name tuple. This method applies to attribute names.

If the prefix is not declared, or if the name is not valid, it'll either die or return undef depending on the current setting of "fatal_errors".

? $nsup->reset

Resets the object so that it can be reused on another document.

All methods of the interface have an alias that is the name used in the original Java specification. You can use either name interchangeably. Here is the mapping:

Java name	Perl name
pushContext	push_context
popContext	pop_context
declarePrefix	declare_prefix
declarePrefixes	declare_prefixes
getPrefix	get_prefix
getPrefixes	get_prefixes
getDeclaredPrefixes	get_declared_prefixes
getURI	get_uri
processName	process_name
processElementName	process_element_name

processAttributeName process_attribute_name

parseJClarkNotation parse_jclark_notation

undeclarePrefix undeclare_prefix

VARIABLES

Two global variables are made available to you. They used to be constants but simple scalars are easier to use in a number of contexts. They are not exported but can easily be accessed from any package, or copied into it.

? $NS_XMLNS

The namespace for xmlns prefixes, http://www.w3.org/2000/xmlns/.

? $NS_XML

The namespace for xml prefixes, http://www.w3.org/XML/1998/namespace.

TODO

- add more tests
- optimise here and there

SEE ALSO

XML::Parser::PerlSAX

AUTHORS

? Robin Berjon <robin@knowscape.com>

? Chris Prather <chris@prather.org>

COPYRIGHT AND LICENSE

terms as the Perl 5 programming language system itself.

CONTRIBUTORS

- Chris Prather <cprather@hdpublishing.com>
- David Steinbrunner <dsteinbrunner@pobox.com>
- Paul Cochrane <paul@liekut.de>
- Paulo Custodio <pauloscustodio@gmail.com>