NAME

XML::LibXML::Pattern – XML::LibXML::Pattern – interface to libxml2 XPath patterns

SYNOPSIS

```
use XML::LibXML;
my $pattern = XML::LibXML::Pattern->new('/x:html/x:body//x:div', { 'x' => 'http://
# test a match on an XML::LibXML::Node $node

if ($pattern->matchesNode($node)) { ... }

# or on an XML::LibXML::Reader

if ($reader->matchesPattern($pattern)) { ... }

# or skip reading all nodes that do not match

print $reader->nodePath while $reader->nextPatternMatch($pattern);

$pattern = XML::LibXML::Pattern->new( pattern, { prefix => namespace_URI, ... } );
$bool = $pattern->matchesNode($node);
```

DESCRIPTION

This is a perl interface to libxml2's pattern matching support *http://xmlsoft.org/html/libxml-pattern.html*. This feature requires recent versions of libxml2.

Patterns are a small subset of XPath language, which is limited to (disjunctions of) location paths involving the child and descendant axes in abbreviated form as described by the extended BNF given below:

```
Selector ::=     Path ( '|' Path )*
Path     ::=     ('.//' | '//' | '/' )? Step ( '/' Step )*
Step     ::=     '.' | NameTest
NameTest ::=     QName | '*' | NCName ':' '*'
```

For readability, whitespace may be used in selector XPath expressions even though not explicitly allowed by the grammar: whitespace may be freely added within patterns before or after any token, where

```
token     ::=     '.' | '/' | '//' | '|' | NameTest
```

Note that no predicates or attribute tests are allowed.

Patterns are particularly useful for stream parsing provided via the `XML::LibXML::Reader` interface.

new()

```
$pattern = XML::LibXML::Pattern->new( pattern, { prefix => namespace_URI, ... }
```

The constructor of a pattern takes a pattern expression (as described by the BNF grammar above) and an optional HASH reference mapping prefixes to namespace URIs. The method returns a compiled pattern object.

Note that if the document has a default namespace, it must still be given an prefix in order to be matched (as demanded by the XPath 1.0 specification). For example, to match an element `<a xmlns="http://foo.bar"</a>`, one should use a pattern like this:

```
$pattern = XML::LibXML::Pattern->new( 'foo:a', { foo => 'http://foo.bar' });
```

matchesNode($node)

```
$bool = $pattern->matchesNode($node);
```

Given an XML::LibXML::Node object, returns a true value if the node is matched by the compiled pattern expression.

SEE ALSO

XML::LibXML::Reader for other methods involving compiled patterns.

AUTHORS

Matt Sergeant, Christian Glahn, Petr Pajas

VERSION

2.0134

COPYRIGHT

LICENSE