

Extending XML::XPath

Michel Rodriguez

I have always admired Matt Sergeant's `XML::XPath` module. `XML::XPath` is a pretty complete implementation of the W3C's XPath recommendation. You see, I am the author of `XML::Twig`, which implements, in a way that can only be described as dirty, a subset of XPath. Matt's XPath engine is written properly: it uses the BNF of the language as the specification, its regexp-based tokenizer feeds an analyzer that builds an internal form of the expression which is then applied to an XML document (or fragment). This is much cleaner, and more powerful, than the simple regexp-based, all-in-one analyzer that I use in `XML::Twig`. Incidentally, this sad state of affairs is mostly due the fact that I never set out to write a full XPath engine, but it just grew as users, including me, asked that I support more and more features of the language.

When I decided to add a complete XPath engine to `XML::Twig`, good software engineering principle told me that I should have a look at `XML::XPath`, and try to re-use it as much as possible.

This article shows how I integrated the `XML::XPath` engine into `XML::Twig` to (at last!) give it full XPath support, then how I got carried away and added XPath support to `XML::DOM` too. I also give you a blueprint on how to add XPath support to modules that manage trees. Maybe more important, the method I used can be used in other cases to wrestle re-use out of code that might not have been designed for it.

XPath

XPATH (<http://www.w3c.org/XPATH/>) is a W3C standard for the query of XML documents. XPath expressions can be as simple as `/doc/title`, which selects all elements `title` right within the root of the document (the `doc` element), and as complex as `//item/title[following-sibling::description[starts_with(string(), "2002")]]`, which selects all `title` elements within `item` elements, which are followed by a `description` element that starts with the string 2002.

`XML::XPath` is a module that builds a tree representing an XML document, and then offers methods to select nodes in the tree using XPath, to modify the tree and to output it (or part of it). It is a stable module, quite popular, although `XML::LibXML` is probably a better choice if you can install it: it is faster and more powerful, if often more unstable.

`XML::Twig` is a Perl module that does "*XML, The Perl Way*". It offers a comprehensive API on XML documents, and includes a (very!) crude XPath engine.

Another module on CPAN, `Class::XPath`, is designed as a general purpose XPath engine. Its documentation lists clearly the methods to implement to add XPath support to any code that manages trees. I chose not to use it though, because it's XPath engine is quite limited, even less powerful than `XML::Twig`'s.

First Contact

The first step was to have a look at `XML::XPath`, figure out how it worked, and see how I could use it's XPath engine.