1. Introduction

In attempting to put together ip6.progsoc.org, I decided I would use Perl, Postgresql and CGIs. I wrote a set of management scripts which did user registration and tunnel provisioning but had not time to do the web interface.

This is my story.

For Perl, there are a wide variety of module choices. I wanted to pick "best practise" modules that are well documented and widely used. A few reasons for that: - more people to ask - (hopefully) easy to pickup - security issues addressed quickly

2. Modules

2.1. CGI.pm

The grand-daddy of pretty much every CGI module. This should be your starting point if you are looking to use Perl as a CGI language. My initial versions used CGI.pm and hand-generated the HTML within the program.

Using CGI.pm what I ended up with was lots of lines similiar to the below

```perl
print $q->header("text/html");
print $q->start_html(-title => "ProgSoc IPv6 tunnel broker: confirm error");
print $q->h1("Confirm error");
print $q->p("Confirmation code ($md5) already used or doesn’t exist");
print $q->end_html;
```

I thought there had to be a better way.
Templating

Templating is a process where you have a generic document and you customise it for each visitor. There are a number of templating systems: HTML::Template, Template::Toolkit, Mason and Embperl.

Both Mason and Embperl make the HTML document the definitive one. The template HTML document will contain "snippets" of Perl. HTML::Template and Template::Toolkit make the Perl CGI definitive. The CGI replaces "snippets" of special markers with generated HTML.

HTML::Template and Template::Toolkit appear more logical to me, and I only briefly evaluated them before deciding that HTML::Template has less features and, thus, is easier to learn.

Indeed there are. You can choose between a structured CGI::Application method, or rolling your own via/using CGI.pm