Getting started with PSGI

PSGI Plack plackup

Prev Next

Building a really good web applications needs a lot of high-level knowledge, but knowing the basic building blocks can help. In the last couple of years a new standard emerged in the Perl world for doing that, called PSGI.

Let's see how can you use these low level building blocks to start building a simple web application.

Higher level abstraction

Before looking at actual PSGI code, let's emphasize. If you would like to build a modern web application in Perl, you'd probably start with one of the web frameworks that provide a higher level of abstraction than PSGI. Frameworks such as Dancer, Mojolicious, or Catalyst. In fact they all use PSGI in order to communicate with the web server. Similarly, if you'd like to build a "CGI application", you'd be probably better off writing a PSGI compliant web page and then configure CGI in-front of the PSGI-based application.

The Prerequisites

In order to be able to run the example, first you need to have Plackinstalled.

I won't go into details how to do that, in general you should be able to type `cpan Plack`, or `cpanm Plack`.

If you are using Windows, then the DWIM Perl for Windows distribution already includes Plack.

First script

Create a Perl script called first.psgi - it is just a regular perl script, with a final statement returning a subroutine:

```
#!/usr/bin/perl
use strict;
use warnings;

my $app = sub {
  return [
    '200',
    [ 'Content-Type' => 'text/html' ],
        [ 42 ],
    ];
};
```

That's the whole script.

In the console run `plackup first.psgi`. It will tell you something like this:

```
HTTP::Server::PSGI: Accepting connections at http://0:5000/
```

Take your browser now and point it to http://localhost:5000/

You should see **42** in your browser.

Add some dynamic data

Then change the file to have time() instead of 42:

```
#!/usr/bin/perl
use strict;
use warnings;

my $app = sub {
    return [
        '200',
        [ 'Content-Type' => 'text/html' ],
        [ time() ],
    ];
};
```

Stop the server using Ctrl-C and then start it again.

If you reload the browser you will see a timestamp there. A 10-digit long number.

If you reload the page several times, you will see how that timestamp changes. With that we managed to write our first dynamic web application using PSGI.

If you'd like to have a more human readable timestamp, replace the `time()` call by `scalar localtime`. Stop, restart the script. Reload the browser.

Explain PSGI

A couple of things. The extension .psgi is not important. It is just a convention. You could call your file any name.

The anonymous subroutine we created returns and array reference with 3 values. The first is the HTTP status code. In our case it is 200 meaning success.

The second is itself an array reference containing key-value pairs of the header. There are certain rules for what kind of keys we can supply there, but in case we set the status to be 200, we must provide the at least the `Content-Type`.

The third part is again an array reference with the body of the response. This can be split into several parts or provided as a single element in that array reference.

That's about the basics of writing a "web application" using PSGI.

Enjoy playing with it.