NAME

Catalyst::Manual::Deployment::Apache::mod_perl - Deploying Catalyst with mod_perl

mod_perl Deployment

The recommended method of deploying Catalyst applications is FastCGI. In many cases, mod_perl is not the best solution, but we'll list some pros and cons so you can decide for yourself.

Pros

Speed

mod_perl is fast, and your entire app will be loaded in memory within each Apache process.

Shared memory for multiple apps

If you need to run several Catalyst apps on the same server, mod_perl will share the memory for common modules.

Cons

Memory usage

Since your application is fully loaded in memory, every Apache process will be rather large. This means a large Apache process will be tied up while serving static files, large files, or dealing with slow clients. For this reason, it is best to run a two-tiered web architecture with a lightweight frontend server passing dynamic requests to a large backend mod_perl server.

Reloading

Any changes made to the code of your app require a full restart of Apache. Catalyst does not support Apache::Reload or StatINC. This is another good reason to run a frontend web server where you can set up an `ErrorDocument 502` page to report that your app is down for maintenance.

Cannot run multiple versions of the same app

It is not possible to run two different versions of the same application in the same Apache instance because the namespaces will collide.

Cannot run different versions of libraries

If you have two different applications which run on the same machine, and each application needs a different versions of a library, the only way to do this is to have per-vhost perl interpreters (with different library paths). This is entirely possible, but nullifies all the memory sharing benefits that you get from having multiple applications sharing the same interpreter.

Setup

Now that we have that out of the way, let's talk about setting up mod_perl to run a Catalyst app.

2. Install Apache with mod_perl

Both Apache 1.3 and Apache 2 are supported, although Apache 2 is highly recommended. With Apache 2, make sure you are using the prefork MPM and not the worker MPM. The reason for this is that many Perl modules are not thread-safe and may have problems running within the threaded worker environment. Catalyst is thread-safe however, so if you know what you're doing, you may be able to run using worker.

In Debian, the following commands should get you going.

```
    apt-get install apache2-mpm-prefork
    apt-get install libapache2-mod-perl2
```

3. Configure your application

Every Catalyst application will automagically become a mod_perl handler when run within mod_perl. This makes the configuration extremely easy. Here is a basic Apache 2 configuration.

```
    PerlSwitches -I/var/www/MyApp/lib
    PerlModule MyApp

    <Location />
        SetHandler          modperl
        PerlResponseHandler MyApp
    </Location>
```

The most important line here is `PerlModule MyApp`. This causes mod_perl to preload your entire application into shared memory, including all of your controller, model, and view classes and configuration. If you have -Debug mode enabled, you will see the startup output scroll by when you first start Apache.

Also, there have been reports that the block above should instead be (but this has not been confirmed):

```
    <Perl>
        use lib '/var/www/MyApp/lib';
        use MyApp;
    </Perl>

    <Location />
        SetHandler          modperl
        PerlResponseHandler MyApp
    </Location>
```

For an example Apache 1.3 configuration, please see the documentation for Catalyst::Engine::Apache::MP13.

Test It

That's it, your app is now a full-fledged mod_perl application! Try it out by going to http://your.server.com/.

Other Options

Non-root location

You may not always want to run your app at the root of your server or virtual host. In this case, it's a simple change to run at any non-root location of your choice.

```
    <Location /myapp>
        SetHandler          modperl
        PerlResponseHandler MyApp
    </Location>
```

When running this way, it is best to make use of the `uri_for` method in Catalyst for constructing correct links.

Static file handling

Static files can be served directly by Apache for a performance boost.

```
    DocumentRoot /var/www/MyApp/root
    <Location /static>
        SetHandler default-handler
    </Location>
```

This will let all files within root/static be handled directly by Apache. In a two-tiered setup, the frontend server should handle static files. The configuration to do this on the frontend will vary.

Note the path of the application needs to be stated explicitly in the web server configuration for this recipes.

AUTHORS

Catalyst Contributors, see Catalyst.pm

COPYRIGHT

This library is free software. You can redistribute it and/or modify it under the same terms as Perl itself.

syntax highlighting: no syntax highlighting ▼

120193 Uploads, 34929 Distributions
178154 Modules, 12986 Uploaders

hosted by YellowBot