

mod_perl and Other Environments

Bringing mod_perl Applications to Other
Perl Environments

Gunther Birznieks

gunther@eXtropia.com

Overview

- Open Discussion
 - Feel Free to Ask Questions Anytime...
- Primary Motivation and Background
 - Migrating an application written for mod_perl to other environments or vice versa
 - Come from an environment where we write applications to run everywhere Perl does
- Purpose: To provide an overview of the various quirks in other persistent environments
- Covers mod_perl, PerlEx, Velocigen, Speedy::CGI, FastCGI

The Main Issues

- Persistent Perl Interpreters have different quirks in their architecture.
- This may affect different things:
 - Caching of resource (eg Apache::DBI)
 - BEGIN/END Block
 - Initialization and Clean up of resources
 - Application architecture (how interpreters are chosen)

Mod_perl

- Mod_perl v1.x will be the base of our comparison.
- Reasons:
 - Open Source
 - Easy to dissect
 - Supports multiple application models

Mod_perl

- Mod_perl is not purely an application accelerator.
 - It is an adapter to Apache using Perl.
- This allows it to satisfy many different application models.
 - We will cover three:
 - Apache::Registry
 - Apache::PerlRun
 - Apache handler

Mod_perl

- Architecture
 - Perl embedded in Apache
 - Apache == MultiProcessing not MultiThreaded
 - This means 1 separate Perl process for each Apache
 - Everything is a handler() (a hook into the Apache API)

Mod_perl Architecture

- Perl/CGI normally runs under *mod_cgi*
- *Apache::Registry* and *Apache::PerlRun* are handlers also
- *Apache::Registry* takes scripts and wraps them inside a subroutine cached in memory which is called everytime the script is called
- *Apache::PerlRun* is the same. Except the code and data for the script is cleared out afterwards. Module code and data is still cached.
- We'll focus on *Apache::Registry* as *Apache::PerlRun* is for unclean scripts.

Mod_perl

- Resource Caching
 - The classic – Database Connections
 - Apache::DBI
 - Caches connections based on connect parameters: username, password, database, error level params
 - Pings database connection to see if still valid (setPingTimeout can adjust)

Mod_perl

- BEGIN/END Blocks
 - BEGIN only executes once upon first pull of script into memory
 - An additional time on restart for parent process if PerlFreshRestart is on, and pulled into parent process with `Apache::RegistryLoader`
 - END blocks called once upon shutdown except in Registry scripts (then called each time)

Mod_perl

- Initialization/Cleanup
 - PerlModule and PerlRequire (startup.pl)
 - Apache->server->register_cleanup()

PerIEx

- Activestate's PerIEx product for NT
- Very similar to Apache::Registry
- Architecture
 - A pool of Perl interpreters shared by a multi-threaded web server (eg IIS)

PerlEx

- BEGIN/END blocks
 - BEGIN is basically same as mod_perl
 - END is different from mod_perl Registry
 - Executes only when script is reloaded or when PerlEx DLL unloaded.
- Resource caching
 - ActiveState recommends connecting in a BEGIN block using a global. ☹️
 - This is a bad practice for caching connections
 - Just look at Apache::DBI design

PerlEx

- Resource Caching
 - Apache::DBI would be a better alternative
 - Unfortunately, there are Apache::* functions
 - However, these can be stripped out
 - Or use Velocigen trick (discussed later)
- Initialization and Cleanup
 - Cleanup handled by END block
 - Initialization
 - Handled by Preload directive for scripts and libraries

Speedy::CGI

- Architecture
 - Standalone Perl Daemon
 - First time the program is called it gets loaded in a daemon that forever listens to requests
 - Similar to FastCGI
 - But easier – no need for changes to code other than to handle persistent Perl

Speedy::CGI

- Resource Caching
 - SpeedyCGI Docs recommendation similar to ActiveState.
 - But better...acknowledges problems with DB connections getting lost.

```
use vars qw($dbh);  
unless (defined($dbh) && &db_connection_ok($dbh)) {  
    $dbh = &open_db_connection;  
}
```

- Also can modify Apache::DBI
 - But no startup script hook means that you still have to modify the script to do DB caching

Speedy::CGI

- BEGIN/END Blocks
 - BEGIN at start of script (and reload)
 - END after FIRST run of script only
- Initialization/Cleanup
 - No specific initialization
 - `$sp->register_cleanup()` end of request
 - Replace END with this
 - `$sp->add_shutdown_handler()` end of perl interpreter

Velocigen

- Architecture
 - Similar to Servlets
 - Plug-in in server is lightweight and opens a socket to stand-alone Velocigen interpreter process pool
 - Runs scripts similar to Apache::PerlRun
 - Not Apache::Registry

Velocigen

- Resource Caching
 - Comes with Apache::DBI!
 - And other Apache::* modules
 - So that Apache-specific directives just silently fail

Velocigen

- BEGIN/END Blocks
 - Not documented
 - In practice BEGIN seems to operate same as the others (run once, then run again on a change to the script)
 - For equivalent of BEGIN/END hooks are added
 - start() and end()
 - Equivalent of BEGIN/END
- Initialization/Cleanup
 - None documented...see start() and end()

FastCGI

- Architecture
 - Separate Perl Daemon running
 - Web Server connects via FastCGI protocol handler
 - Scripts always require modification.
 - Even w/CGI.pm wrapper
 - Normally use FCGI and have to run inside an `accept()` loop kind of like writing a server

FastCGI

- Resource Caching
 - Not really applicable in the same way
 - The script runs in a perpetual loop that the programmer has to deal with
 - However, Apache::DBI methods should work similar to discussed previously

FastCGI

- BEGIN/END Blocks
 - FastCGI daemon is a normal Perl script so BEGIN and END run before and after the accept() request loop
- Initialization/Cleanup
 - You perform your own based on the top and bottom of the request loop

Persistence Summary

	Mod_perl	PerlEx	Speedy::CGI	Velocigen	FastCGI
Architecture	Perl embedded in Apache, close Apache API ties	Pool of Interpreters embedded in Web Servers	Separate Perl Daemon connected from web server or command-line	Separate Perl Processes connected from Web Servers	Single Process running in a perpetual accept() loop using FastCGI protocol
Resource Caching	Apache::DBI	BEGIN/END block logic	BEGIN/END block logic	Apache::DBI	Custom
BEGIN/END	END at end of request	END at end of interpreter	END at end of 1 st run, replace with cleanup	start(), end()	Custom
Init/Cleanup	PerlModule & PerlRequire, register_cleanup()	Preload directive, END block	No Init, Shutdown handler	NA	Custom

Open Discussion

Questions?

Updates?