



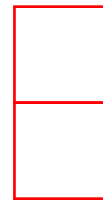
mod_perl handlers

ApacheCon

Rich Bowen - <Rich@CooperMcGregor.com>

- mod_perl handlers
- Introduction
- Introduction - CGI accelerator
- However ...
- A mod_perl handler
- Example mod_perl handler
- Configuring Apache
- Comments
- Writing a Perl module
- Perl OO Tutorial
- Perl OO Tutorial
- Where the modules live
- Installing a mod_perl handler
- Installing a Perl module manually
- Installing a Perl module the right way
- Generating your Makefile.PL
- Makefile.PL
- Auto-generating this stuff
- Configuring Apache
- Actual useful handlers
- Example handler: Form content
- What next?
- Apache::DBI
- connect_on_init
- Things not to do

- Things not to do



ApacheCon : mod_perl handlers

*Copyright © 2002, Cooper McGregor, Inc.
All rights reserved.*

mod_perl handlers - ApacheCon

mod_perl handlers

Rich Bowen -
rbowen@apache.org



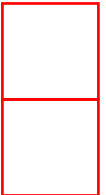
Cooper McGregor -
coopermcgregor.com - Apache
Training

ApacheCon 2002 - Las Vegas

Writing mod_perl handlers

[Index](#)

Forward to [Introduction](#)



ApacheCon : mod_perl handlers - Slide #1 of 26

mod_perl handlers - ApacheCon

Introduction

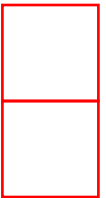
- mod_perl is a CGI accelerator
- Makes CGI programs run many many times faster
- At least, that's what most people use it for
- And it is really good at that

[Index](#)

Back to [mod_perl handlers](#)

Forward to [Introduction - CGI accellerator](#)

ApacheCon : mod_perl handlers - Slide #2 of 26



Introduction - CGI accelerator

```
./ab -n 1000 -c 10 http://localhost/cgi-bin/example1.cgi
```

```
Requests per second:      58.59 [#/sec] (mean)
```

```
...
```

```
./ab -n 1000 -c 10 http://localhost/cgi-perl/example1.cgi
```

```
Requests per second:     108.70 [#/sec] (mean)
```

```
...
```

```
./ab -n 1000 -c 10 http://localhost/perl/example1.cgi
```

```
Requests per second:     213.40 [#/sec] (mean)
```

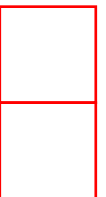
- Pretty impressive

[Index](#)

Back to [Introduction](#)

Forward to [However ...](#)

ApacheCon : mod_perl handlers - Slide #3 of 26



mod_perl handlers - ApacheCon

However ...

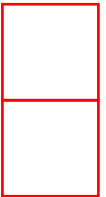
- The real power of mod_perl is that you have access to the Apache API, and can write handlers in Perl
- Influence any part of the Apache process
- Usually used for content generation
- Also often used for authentication, authorization, and access control

[Index](#)

Back to [Introduction - CGI accelerator](#)

Forward to [A mod_perl handler](#)

ApacheCon : mod_perl handlers - Slide #4 of 26



mod_perl handlers - ApacheCon

A mod_perl handler

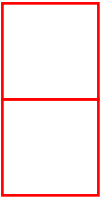
- A mod_perl handler is a Perl module with one method (function)
- The function is (usually) called handler
- Received one argument - an object in the Apache class
- Returns content

[Index](#)

Back to [However ...](#)

Forward to [Example mod_perl handler](#)

ApacheCon : mod_perl handlers - Slide #5 of 26



mod_perl handlers - ApacheCon

Example mod_perl handler

```
package Apache::HandlerTest;
# File is called Apache/HandlerTest.pm
# Path: /usr/lib/perl5/site_perl/5.6.0/Apache/HandlerTest.pm

sub handler {
    my $r = shift; # Apache session object
    $r->content_type('text/html');
    $r->send_http_header;
    $r->print( "Hello, world." );
}

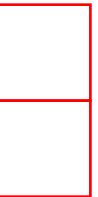
1;
```

[Index](#)

Back to [A mod_perl handler](#)

Forward to [Configuring Apache](#)

ApacheCon : mod_perl handlers - Slide #6 of 26



mod_perl handlers - ApacheCon

Configuring Apache

```
<Location /handlertest>
  SetHandler perl-script
  PerlModule Apache::HandlerTest
  PerlHandler Apache::HandlerTest
</Location>
```

- If your method is not called handler

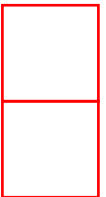
```
PerlHandler Apache::HandlerTest::othermethod
```

[Index](#)

Back to [Example mod_perl handler](#)

Forward to [Comments](#)

ApacheCon : mod_perl handlers - Slide #7 of 26



mod_perl handlers - ApacheCon

Comments

- Uses the c<Apache> object to call Apache API methods `send_http_header` and `print` directly
- Really fast

```
Requests per second:      296.47 [#/sec] (mean)
```

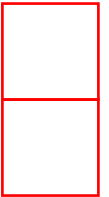
- Note that it will be much faster on a machine that is not as slow as my ancient laptop

[Index](#)

Back to [Configuring Apache](#)

Forward to [Writing a Perl module](#)

ApacheCon : mod_perl handlers - Slide #8 of 26



Writing a Perl module

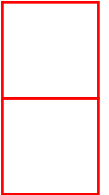
- You need to know a little Perl
- We are specifically writing OO Perl modules
- But OO is really simple in Perl

[Index](#)

Back to [Comments](#)

Forward to [Perl OO Tutorial](#)

ApacheCon : mod_perl handlers - Slide #9 of 26



mod_perl handlers - ApacheCon

Perl OO Tutorial

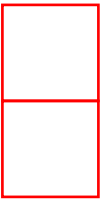
- In Perl, a class is a module
- An object is a bless'ed reference
- A method is a function that gets an object as its first argument

[Index](#)

Back to [Writing a Perl module](#)

Forward to [Perl OO Tutorial](#)

ApacheCon : mod_perl handlers - Slide #10 of 26



Perl OO Tutorial

- A module is a file with a package declaration, which states its namespace
- The first argument to an object method is an object in that namespace

```
my $object = Object::Namespace->new;  
$return = $object->methodname;
```

- And, in the method ...

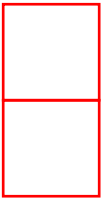
```
sub methodname {  
    my $self = shift;  
    # $self is an object in the Object::Namespace class  
}
```

[Index](#)

Back to [Perl OO Tutorial](#)

Forward to [Where the modules live](#)

ApacheCon : mod_perl handlers - Slide #11 of 26



mod_perl handlers - ApacheCon

Where the modules live

- Perl keeps its modules in its @INC path

```
rbowen@rhiannon:~% perl -le 'print join "\n",@INC;'
/usr/lib/perl5/5.6.0/i686-linux
/usr/lib/perl5/5.6.0
/usr/lib/perl5/site_perl/5.6.0/i686-linux
/usr/lib/perl5/site_perl/5.6.0
/usr/lib/perl5/site_perl
.
```

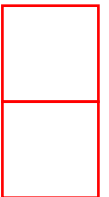
- Those are the directories in which you might find Perl modules
- A Perl module's name is used to find the file path. For example a module called `Apache::HandlerTest` will be located in the file `Apache/HandlerTest.pm`, somewhere in the Perl library path

[Index](#)

Back to [Perl OO Tutorial](#)

Forward to [Installing a mod_perl handler](#)

ApacheCon : mod_perl handlers - Slide #12 of 26



Installing a mod_perl handler

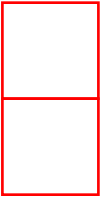
- Two steps:
- Installing the Perl module itself
- Installing the handler in your Apache configuration file

[Index](#)

Back to [Where the modules live](#)

Forward to [Installing a Perl module manually](#)

ApacheCon : mod_perl handlers - Slide #13 of 26



Installing a Perl module manually

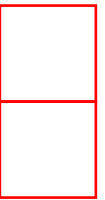
- A module `Foo::Bar` will be located at `Foo/Bar.pm`, with respect to some directory in your Perl library path.
- Typically, these will go in the `site_perl` directory. `site_perl` means modules installed by you at your site
- So, for a module `Apache::Example`, we'll put it at `/usr/lib/perl5/site_perl/5.6.0/Apache/Example.pm` (for example)

[Index](#)

Back to [Installing a mod_perl handler](#)

Forward to [Installing a Perl module the right way](#)

ApacheCon : mod_perl handlers - Slide #14 of 26



Installing a Perl module the right way

- Most Perl modules (ie, those you get off of CPAN) install with:

```
perl Makefile.PL  
make  
make test  
make install
```

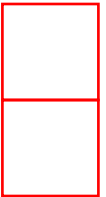
- Yours should too

[Index](#)

Back to [Installing a Perl module manually](#)

Forward to [Generating your Makefile.PL](#)

ApacheCon : mod_perl handlers - Slide #15 of 26



Generating your Makefile.PL

- The contents of Makefile.PL are really simple (or at least they can be)

```
use ExtUtils::MakeMaker;
```

```
WriteMakefile(  
    'NAME'          => 'Apache::Example',  
    'VERSION_FROM' => 'lib/Apache/Example.pm',    # finds $VERSION  
);
```

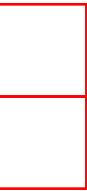
- Yep, that's it
- so, now, what does it mean ...

[Index](#)

Back to [Installing a Perl module the right way](#)

Forward to [Makefile.PL](#)

ApacheCon : mod_perl handlers - Slide #16 of 26



mod_perl handlers - ApacheCon

Makefile.PL

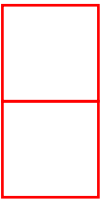
- Assumes that your modules are in a `lib` subdirectory (this is a good standard to follow)
- The `NAME` argument specified the name of the package
- The `VERSION_FROM` argument specifies which file contains the variable `$VERSION`, which will determine the version number put on a distribution
- That's really all there is to it

[Index](#)

Back to [Generating your Makefile.PL](#)

Forward to [Auto-generating this stuff](#)

ApacheCon : mod_perl handlers - Slide #17 of 26



Auto-generating this stuff

- There are a few ways to auto-generate a Perl module package
- Most of them suck
- By far the best is `ExtUtils::ModuleMaker`

```
perl -MExtUtils::ModuleMaker -e 'Quick_Module("Apache::Example");'
```

- Generates all of the necessary files for a Perl module, with installation stuff, documentation, tests, etc.
- Insert demonstration here
- Then you can ...

```
perl Makefile.PL && make && make install
```

- Note that this module is not (yet) a standard part of Perl, and needs to be installed from CPAN before you can use it.
- There's also a utility called `h2xs` which some folks will encourage you to use. Ignore them.

[Index](#)

Back to [Makefile.PL](#)

Forward to [Configuring Apache](#)

ApacheCon : mod_perl handlers - Slide #18 of 26



Configuring Apache

- Next, we need to configure Apache to use our module
- This is done with a <Location> section and a SetHandler

```
<Location /url_goes_here  
    SetHandler perl-script
```

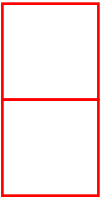
```
    PerlModule Your::Module  
    PerlHandler Your::Module  
</Location>
```

[Index](#)

Back to [Auto-generating this stuff](#)

Forward to [Actual useful handlers](#)

ApacheCon : mod_perl handlers - Slide #19 of 26



mod_perl handlers - ApacheCon

Actual useful handlers

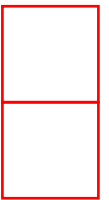
- Handlers that print `Hello World` are not useful
- At a minimum, you need to be able to read in form content.
- Fortunately, mod_perl offers a simple way to do this.

[Index](#)

Back to [Configuring Apache](#)

Forward to [Example handler: Form content](#)

ApacheCon : mod_perl handlers - Slide #20 of 26



mod_perl handlers - ApacheCon

Example handler: Form content

```
sub handler {  
    my $r = shift;
```

```
    if ($r->method eq 'POST') {  
        %form = $r->content;  
    } else {  
        %form = $r->args;  
    }  
}
```

```
$r->send_http_header( 'text/html' );  
$r->print( "Name = " . $form{name} );  
}
```

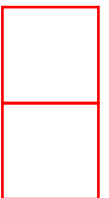
- The method method determines the HTTP method used (GET or POST in this case)
- Form content is retrieved from different places, depending on the method.
- Return value is a hash, which you can then use for all of your input needs.

[Index](#)

Back to [Actual useful handlers](#)

Forward to [What next?](#)

ApacheCon : mod_perl handlers - Slide #21 of 26



mod_perl handlers - ApacheCon

What next?

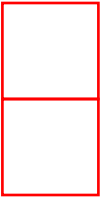
- Well, if you've been doing CGI, stuff, pretty much the rest of it is the same as you're used to
- Read in form content
- Do something with it
- Print output
- A few other useful things to know ...

[Index](#)

Back to [Example handler: Form content](#)

Forward to [Apache::DBI](#)

ApacheCon : mod_perl handlers - Slide #22 of 26



mod_perl handlers - ApacheCon

Apache::DBI

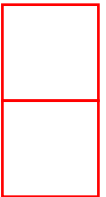
- The Perl module DBI facilitates connections to your favorite database.
- Apache::DBI hijacks DBI and makes your database connections persistent.
- Even if you explicitly `disconnect`, it retains your connection
- See also the `connect_on_init` method

[Index](#)

Back to [What next?](#)

Forward to [connect_on_init](#)

ApacheCon : mod_perl handlers - Slide #23 of 26



mod_perl handlers - ApacheCon

connect_on_init

- In `httpd.conf`:

```
PerlRequire /path/to/apache_preload.pl
```

- Then, in `apache_preload.pl`:

```
use Apache::DBI ();  
use DBI ();
```

```
Apache::DBI->connect_on_init (  
    $database, $DBI_username, $DBI_password,  
    { AutoCommit => 1, }  
);
```

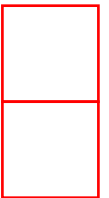
- Note that `Apache::DBI` should get loaded before `DBI`
- `DBI` will create the connection each time a new child process is created
- Then, in your code, you just use `DBI` as you would ordinarily

[Index](#)

Back to [Apache::DBI](#)

Forward to [Things not to do](#)

ApacheCon : mod_perl handlers - Slide #24 of 26



mod_perl handlers - ApacheCon

Things not to do

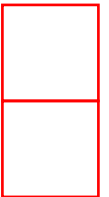
- Don't `exit`
- The `exit` command terminates the Perl process. This is a Bad Thing under `mod_perl`
- You are left with an Apache child with no embedded Perl process
- `mod_perl` gets confused, and may return whatever content it already has in the buffer from last time
- This is potentially a huge problem, which is easily solved
- So don't `exit`

[Index](#)

Back to [connect_on_init](#)

Forward to [Things not to do](#)

ApacheCon : mod_perl handlers - Slide #25 of 26



mod_perl handlers - ApacheCon

Things not to do

- Don't use global variables
- Of course, you should not do that anyway
- Global variables are really global. Across connections. One user can see variables set by another user. This is a Bad Thing
- `use strict` and `use warnings` everywhere. This will remove the temptation to use magical or global variables
- Use `my` religiously on all variables. (Again, you should be doing this anyway.)

[Index](#)

Back to [Things not to do](#)

ApacheCon : mod_perl handlers - Slide #26 of 26

