

# Lightweight mod\_perl Applications with Apache::Dispatch

Fred Moyer

[fred@redhatpenguin.com](mailto:fred@redhatpenguin.com)

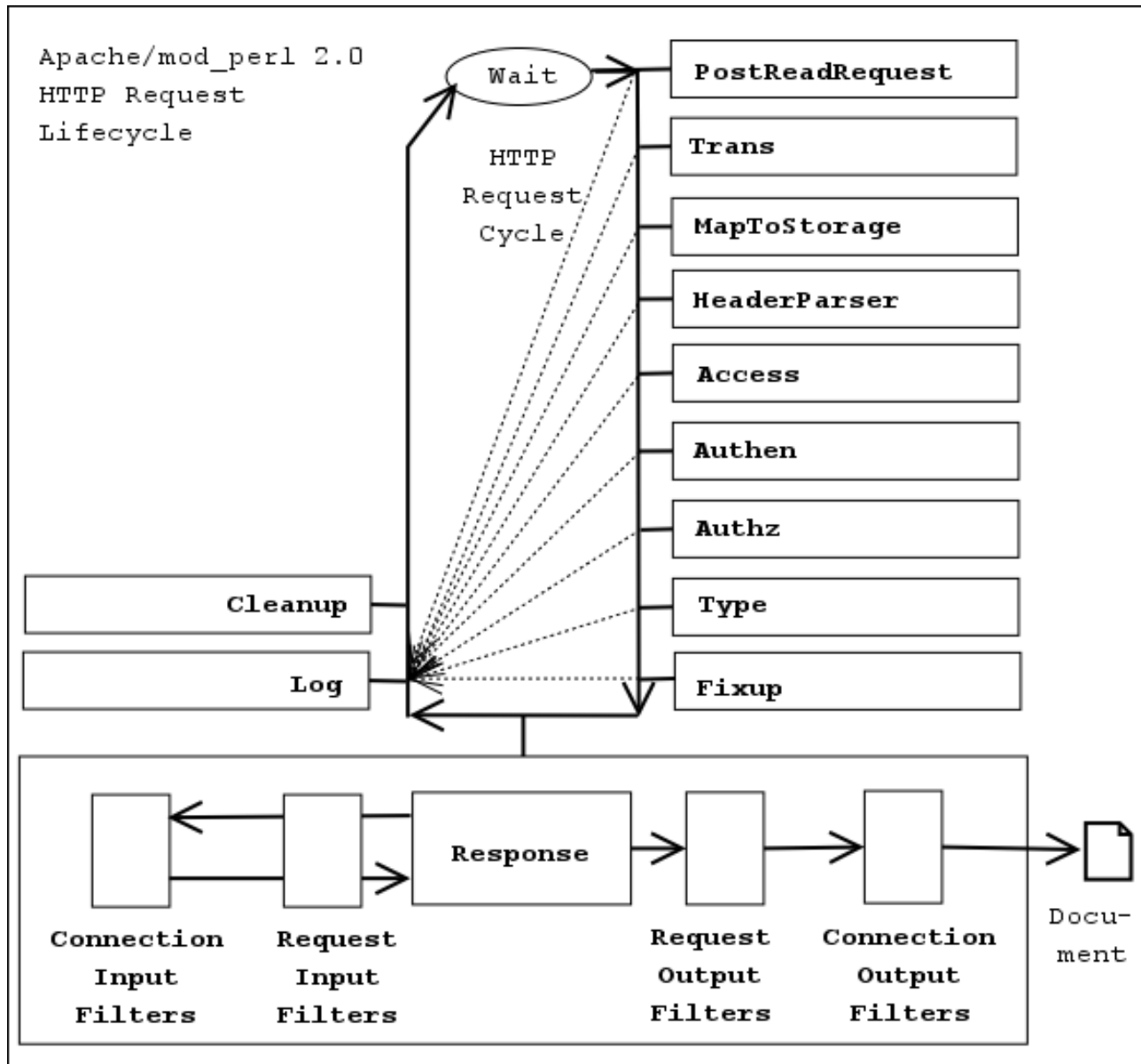
# Apache::Dispatch

- Abstraction layer for mod\_perl applications
- The power of mod\_perl handlers
- Maps URIs to application resources via method handlers
- Painless migration from mod\_perl handlers

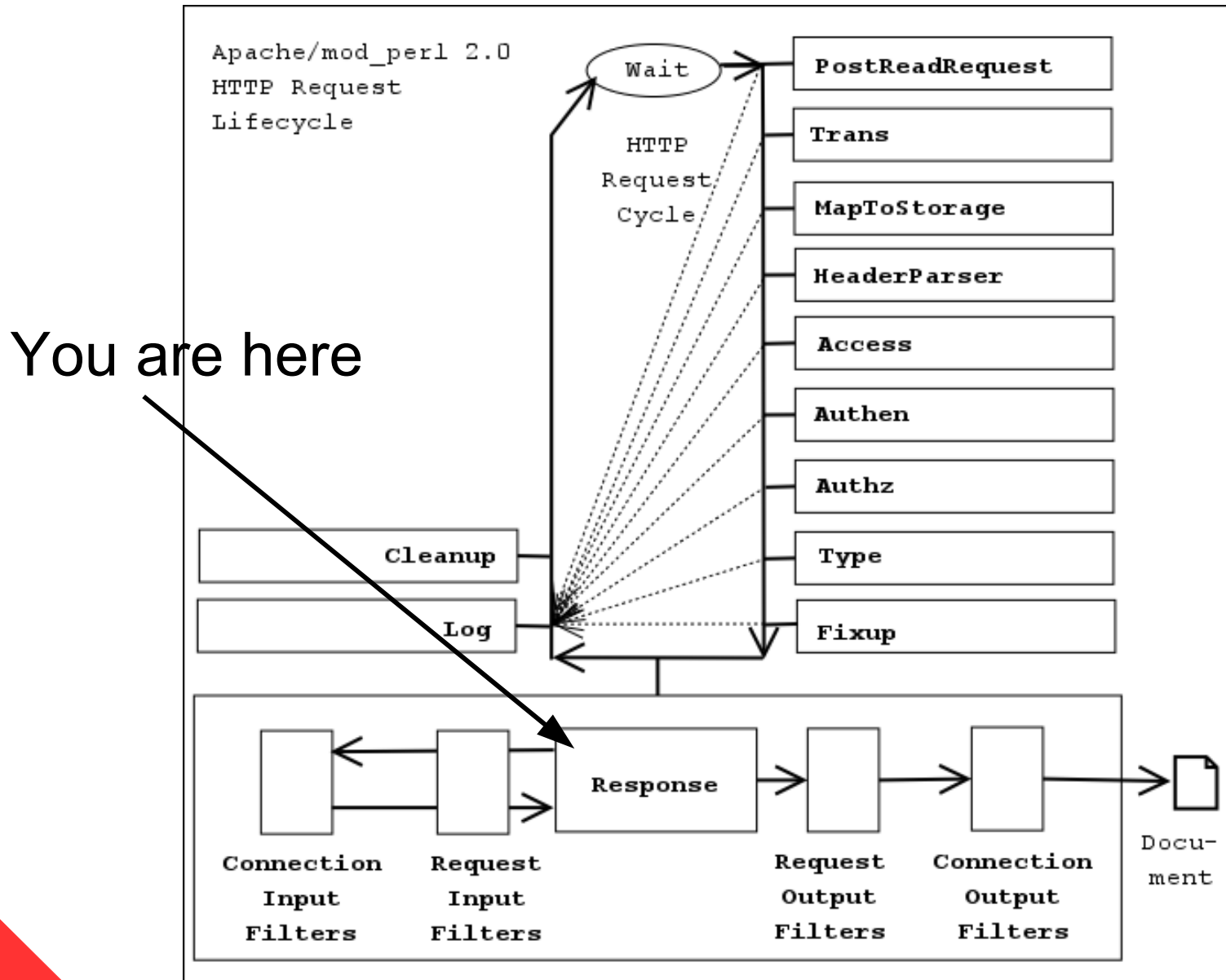
# Not another Framework

- There are plenty of those already
  - OpenInteract
  - Catalyst
  - Jifty
  - Axkit
  - Woodstock
  - ...
- Like CGI::Application in the sense that it's a library, but only for mod\_perl
- **MVC**

# HTTP Request Lifecycle



# Handles Response Phase



# Start with URIs

- /hello
- /hello/world
- /hello/world/oslo
- /hello/oslo
- hello/oslo/mongers

# Map URIs to Method Handlers

- Method handler passes package name as first argument, \$r as second argument
- /hello => My::Hello->dispatch\_index
- /hello/world => My::Hello->dispatch\_world
- /hello/world/oslo => My::Hello::World->dispatch\_oslo
- /hello/oslo => My::Oslo->dispatch\_index
- /hello/oslo/mongers => My::Oslo->dispatch\_mongers
- dispatch\_index only works at the top level

# Apache::Test is your Friend

- You do write your tests first, don't you? ;)
- my/app/t/01hello.t =>

```
use strict;
use warnings FATAL => 'all';
use Apache::Test;
use Apache::TestRequest;
plan tests => 2;
my $res = GET '/hello';
ok($res->code == 200);
ok($res->content =~ m/Hello/i);
```

# Configuration

my/app/t/conf/extra.last.conf.in =>

```
PerlLoadModule      Apache2::Dispatch
PerlLoadModule      My::Hello
PerlLoadModule      My::Hello::World
DispatchRequire     On
DispatchUpperCase  On

<Location /hello>
  SetHandler         modperl
  DispatchPrefix     My::Hello
  PerlResponseHandler Apache2::Dispatch
</Location>
```

# Method Handlers

my/app/lib/My/Hello.pm =>

```
package My::Hello;
use strict;
use Apache2::Const -compile => qw( OK );
use Apache2::RequestIO;
use Apache2::RequestRec;

sub dispatch_index {
    my $class = shift;
    my $r = shift;
    $r->content_type('text/plain');
    $r->print("Hello!");
    return Apache2::Const::OK;
}
```

# Apache::Test

my/app/Makefile.PL =>

```
#!/perl
use strict;
use warnings FATAL => 'all';
use Apache::TestMM;
Apache::TestMM->import(qw(test clean));
Apache::TestMM::filter_args();
Apache::TestMM::generate_script('t/TEST');
use ModPerl::MM;
ModPerl::MM::WriteMakefile(
    %standard_make_params );
```

# Apache::Test

my/app/t/TEST.PL =>

```
#!/perl
```

```
use strict;
```

```
use warnings FATAL => 'all';
```

```
use Apache::TestRunPerl();
```

```
Apache::TestRunPerl->new->run(@ARGV);
```

# Hello!

- `perl Makefile.PL`
- `make`
- `make test`
- debug if errors
- `make install`

Include the `extra.last.conf.in` section shown earlier to your `httpd.conf`, restart `httpd`, and visit `http://localhost/hello`

# Test me, Test me

Easily test with different httpd/mod\_perl builds

- export APACHE\_TEST\_HTTPD = /path/to/test\_httpd/bin/httpd
- export APACHE\_TEST\_APXS = /path/to/test\_httpd/bin/apxs

Test without having to 'make test' each time

- export APACHE\_TEST\_LIVE\_DEV = /path/to/my/app/lib
- ./t/TEST -start-httpd
- ./t/TEST t/01hello.t

# Review

- Created a basic test driven Apache::Dispatch based hello web service
- Wrote a test, a config file, and a method handler
- Reviewed Apache::Test basics

# The Extras

- Pre-Dispatch method handler
- Post-Dispatch method handler
- Custom Error method handler
- Inheritance
- Autoloading

# Pre-Dispatch Method Handler

- DispatchExtras Pre
- Used for setup common to dispatch\_\* methods
- Similar functionality in most frameworks
- Best suited for code re-use - DRY

```
sub pre_dispatch {  
  my ($class, $r) = @_  
  my %data = My::Model::Hello->data;  
  # stash the data in the request for use later  
  $r->pnotes( hello_common => \%data );  
}
```

# Post-Dispatch Method Handler

- DispatchExtras Post
- Teardown for package dispatch\_\* methods
- As with pre\_dispatch, similar to most frameworks, main function is DRY
- Still part of the response phase
- Runs after the dispatch\_\* handler, even if the response was not OK

```
sub post_dispatch {  
  my ($class, $r) = @_;  
  $r->log->info("Request to " .  
    $r->connection->remote_ip . " was served");  
}
```

# Custom Error Method Handler

- DispatchExtras Error
- Handles error in dispatch\_\* methods
- Ignores errors in Pre/Post handlers

```
sub error_dispatch {  
    my ($class, $r, $payload, $src) = @_;  
    $r->log->error("Oops!: $@");  
    if ($src == Apache2::Const::NOT_FOUND) {  
        $r->print(  
            $self->process_tmpl('not_found.tmpl'));  
    }  
    return Apache2::Const::OK;  
}
```

# Inheritance

- DispatchISA My::BaseHandler
- Useful for defining methods across different packages
- For example, define one error\_dispatch method and share it among other classes
- Suggested that it be used with DispatchAUTOLOAD Off (default)

# Autoloading

- DispatchAUTOLOAD On
- Autoloaded methods must be declared
- Read the camel book 3<sup>rd</sup> ed pp326-329
- Read the examples in t/lib

```
our $AUTOLOAD;  
sub dispatch_autoloaded_method;  
sub AUTOLOAD {  
    my ($class, $r) = @_;  
    $r->log->info("Method $AUTOLOAD called");  
    # ....  
}
```

# Filtering

- Use mod\_perl2 filtering API for mp2
- Apache::Dispatch has built-in filtering for mp1

httpd.conf =>

DispatchFilter On

PerlResponseHandler Apache::Dispatch My::Filter

```
package My::Filter;
```

```
sub handler {
```

```
    my $r = shift;
```

```
    $r->send_http_header();
```

```
    $r = $r->filter_register;
```

```
    my $fh = $r->filter_input;
```

```
    while (<$fh>) {
```

```
        # modify $_;
```

```
        print;
```

```
    }
```

```
}
```

# Lightweight

- <http://chamas.com/bench>
  - 658.7 req/sec for Apache::Dispatch
  - 856.0 req/sec for mod\_perl handler
- Geoffrey Young's benchmarks
  - <Location> and method handler – 14.34 req/sec
  - A::D with DispatchISA On – 13.78 req/sec
- See Todo for the details
- One Perl module, not zillions

# Dependencies

- None unless you want to do testing
- In which case you'll need
  - Apache::Test
  - LWP::UserAgent
- Compare this to other libraries/frameworks which use x number of modules to get started

# Credits

- Geoffrey Young – Author, mod\_perl and testing guru, helping me release the latest version soonish
- Thomas Klausner – Maintenance work and material for this talk
- The mod\_perl community

# References

- [1] - <http://search.cpan.org/perldoc?Apache::Dispatch>
- [2] - mod\_perl Developers Cookbook  
<http://www.modperlcookbook.org>
- [3] - Josh Chamas' Hello World Benchmark  
<http://chamas.com/bench/>
- [4] - Thomas Klausner's YAPC 2003 Presentation  
<http://domm.zsi.at/talks/yapc2003>
- [5] - Apache::Test Part 2 Presentation  
<http://modperlcookbook.org/~geoff/apache>
- [6] - <http://sourceforge.net/projects/apache-dispatch>

# Slides

These slides are freely available at

<http://www.redhotpenguin.com/talks>

Try out the latest version of Apache::Dispatch at

<http://sourceforge.net/projects/apache-dispatch>

Coming soon to a CPAN mirror near you

**Thank you NPW 2006!**