A Few Cool Things About mod_perl 2.0

Geoffrey Young

geoff@modperlcookbook.org

What's New in mod_perl 2.0?

- Everything
- OK, not everything, but...

1.0 Directives

PerlRestartHandler PerlChildInitHandler PerlPostReadRequestHandler PerlInitHandler PerlTransHandler PerlHeaderParserHandler PerlAccessHandler PerlAuthenHandler PerlAuthzHandler PerlTypeHandler PerlFixupHandler PerlHandler PerlLogHandler PerlCleanupHandler PerlChildExitHandler PerlDispatchHandler PerlSetVar PerlAddVar PerlSetEnv PerlPassEnv <Perl> </Perl> PerlFreshRestart PerlModule PerlOpmask PerlRequire PerlScript PerlSendHeader PerlSetupEnv PerlTaintCheck PerlWarn

2.0 Directives

PerlSwitches PerlModule PerlRequire PerlOptions PerlInitHandler
PerlSetVar PerlAddVar PerlSetEnv PerlPassEnv <Perl> </Perl>
PerlSetInputFilter PerlSetOutputFilter PerlLoadModule PerlTrace
PerlInterpStart PerlInterpMax PerlInterpMaxSpare PerlInterpMinSpare
PerlInterpMaxRequests PerlInterpScope PerlProcessConnectionHandler
PerlChildInitHandler PerlChildExitHandler PerlPreConnectionHandler
PerlHeaderParserHandler PerlAccessHandler PerlAuthenHandler
PerlAuthzHandler PerlTypeHandler PerlFixupHandler PerlResponseHandler
PerlLogHandler PerlCleanupHandler PerlInputFilterHandler
PerlOutputFilterHandler PerlPostReadRequestHandler PerlTransHandler
PerlMapToStorageHandler PerlOpenLogsHandler PerlPostConfigHandler

1.0 Classes

```
Apache  Apache::Connection  Apache::Constants  Apache::Constants::Exports
Apache::Debug  Apache::ExtUtils  Apache::FakeRequest  Apache::File
Apache::fork  Apache::httpd_conf  Apache::Include  Apache::Leak
Apache::Log  Apache::ModuleConfig  Apache::MyConfig  Apache::Opcode
Apache::Options  Apache::PerlRun  Apache::PerlRunXS  Apache::PerlSections
Apache::RedirectLogFix  Apache::Registry  Apache::RegistryBB
Apache::RegistryLoader  Apache::RegistryNG  Apache::Resource
Apache::Server  Apache::SIG  Apache::SizeLimit  Apache::src
Apache::StatINC  Apache::Status  Apache::Symbol  Apache::Symdump
Apache::Table  Apache::testold  Apache::URI  Apache::Util  mod_perl
mod_perl_hooks
```

2.0 Classes

Apache2::Access Apache2::Build Apache2::BuildConfig Apache2::CmdParms
Apache2::Command Apache2::compat Apache2::Connection Apache2::Const
Apache2::Directive Apache2::Filter Apache2::FilterRec Apache2::HookRun
Apache2::Log Apache2::Module Apache2::MPM Apache2::ParseSource
Apache2::PerlSections Apache2::PerlSections::Dump Apache2::porting
Apache2::Process Apache2::Reload Apache2::RequestIO Apache2::RequestRec
Apache2::RequestUtil Apache2::Response Apache2::ServerRec Apache2::ServerUtil
Apache2::SourceTables Apache2::Status Apache2::SubProcess
Apache2::SubRequest Apache::Test Apache::Test5005compat Apache::TestBuild
Apache::TestClient Apache::TestCommon Apache::TestCommonPost
Apache::TestConfig Apache::TestConfigC Apache::TestConfigParse
Apache::TestConfigPerl Apache::TestHandler Apache::TestHarness Apache::TestMB
Apache::TestMM Apache::TestPerlDB Apache::TestReport Apache::TestReportPerl
Apache::TestRequest Apache::TestRun Apache::TestRunPerl Apache::TestServer
Apache::TestSmoke Apache::TestSmokePerl Apache::TestSort Apache::TestSSLCA
Apache::TestTrace Apache::TestUtil Apache2::URI Apache2::Util
Apache2::XSLoader APR APR::Base64 APR::Brigade APR::Bucket APR::BucketType
APR::Const APR::Date APR::Error APR::Finfo APR::IpSubnet APR::OS
APR::PerlIO APR::Pool APR::SockAddr APR::Socket APR::String APR::Table
APR::ThreadMutex APR::URI APR::Util APR::UUID APR::XSLoader mod_perl2
ModPerl::BuildMM ModPerl::BuildOptions ModPerl::Code ModPerl::Config
ModPerl::Const ModPerl::CScan ModPerl::FunctionMap ModPerl::Global
ModPerl::Manifest ModPerl::MapUtil ModPerl::MethodLookup ModPerl::MM
ModPerl::ParseSource ModPerl::PerlRun ModPerl::Registry ModPerl::RegistryBB
ModPerl::RegistryCooker ModPerl::RegistryLoader ModPerl::StructureMap
ModPerl::TestReport ModPerl::TestRun ModPerl::TypeMap ModPerl::Util
ModPerl::WrapXS

50% More... Free!

- mod_perl 1.0
  - 30 directives
  - 40 classes
  - 208 methods
- mod_perl 2.0
  - 40 directives
  - 109 classes
  - 413 methods

PerlTypeHandler

- Ever written a `PerlTypeHandler`?
- Of course not!
- mod_mime has a stranglehold on the request in Apache 1.3
  - return `OK` and `SetHandler` doesn't work
  - return `DECLINED` and mod_mime clobbers the `Content-Type`
- No more

A PerlTypeHandler

```
package My::TypeHandler;

use Apache2::RequestRec ();
use Apache2::Const -compile => qw(OK);

use strict;

sub handler {

  my $r = shift;

  $r->content_type('text/foo')
    if $r->filename =~ m!\.foo$!;

  return Apache2::Const::OK;
}

1;
```

Who Cares?

- You *still* won't ever write a `PerlTypeHandler`
- Just one of the ways that mod_perl 2.0 (and Apache 2.0) are different
- Different is (generally) better

Apache2::Const::OK ?

- mod_perl 2.0 made some major changes just prior to the release of 2.0.0
  - aka the great namespace war of 2004
- What's done is done
- Here's what you need to know...

Migrating to Apache2::

- All Apache:: modules now live in the Apache2:: namespace
  - Apache2::Util
  - Apache2::RequestRec
  - Apache2::Const
- Apache-Test is the exception
  - Apache::Test
  - Apache::TestRequest

Migrating to `Apache2::`

- Constants are fully qualified to their package
  - `Apache2::Const::OK`
  - `APR::Const::SUCCESS`
- You don't *need* to use `-compile`

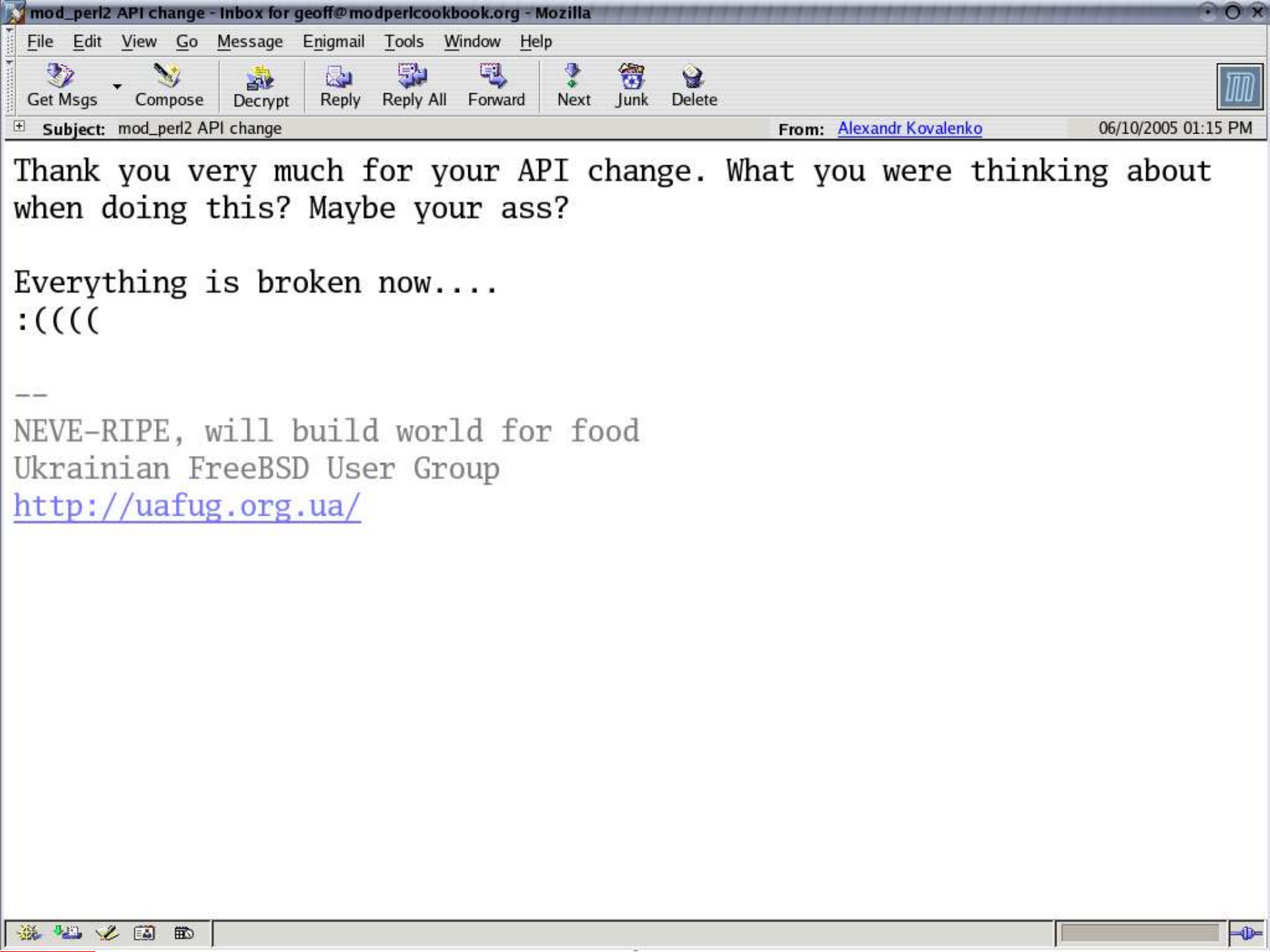
```
use Apache2::Const qw(OK);
...
OK
```

Migrating to `Apache2::`

- A few APIs are entirely different
- Old
  - `Apache->request()`
  - `Apache->server()`
- New
  - `Apache2::RequestUtil->request()`
  - `Apache2::ServerUtil->server()`

Coworker Observation

- "You can practically migrate your code in minutes using `sed`"
  - I'd use perl, of course
- While true, it only applies to 1.99 -> 2.0 migrations
- For mod_perl 1.0 migrations...
  - see prior "why mod_perl 2.0 sucks" talk


mod_perl2 API change - Inbox for geoff@modperlcookbook.org - Mozilla
File Edit View Go Message Enigmail Tools Window Help
Get Msgs Compose Decrypt Reply Reply All Forward Next Junk Delete
Subject: mod_perl2 API change
From: Alexandr Kovalenko
06/10/2005 01:15 PM
Thank you very much for your API change. What you were thinking about when doing this? Maybe your ass?
Everything is broken now....
:((((
--
NEVE-RIPE, will build world for food
Ukrainian FreeBSD User Group
http://uafug.org.ua/

Apache Directives

- Over 340 directives are supported by the standard Apache 2.0 distribution
- Less that 90 are from core Apache
  - `core.c`
  - `http_core.c`
  - `mpm_common.c`
- All the rest are from C extension modules

Core Apache is Small

```
<IfModule mod_perl.c>
  PerlFixupHandler My::Fixup
</IfModule>

<IfModule mod_alias.c>
  Alias /perl-bin /usr/local/apache/perl-bin
</IfModule>

<IfModule mod_env.c>
  PassEnv ORACLE_HOME
</IfModule>
```

Perl Module Configuration

- Most people just use what is available

```
PerlSetVar Widget 1
PerlSetVar Fidget 0
```

- Wouldn't this be cooler?

```
Widget On
Fidget Off
```

Directive Handlers

- As with all things, mod_perl provides an API for creating our own Apache directives
- API in 1.0 was intimidating
  - which is why nobody ever used it
- 2.0 directive handler API is pure Perl
  - you'll love it

```
package My::Directive;

use Apache2::Module ();

use Apache2::Const -compile => qw(FLAG RSRC_CONF);

my @directives = (
  { name         => 'Widget',
    req_override => Apache2::Const::RSRC_CONF,
    args_how     => Apache2::Const::FLAG,
  },
);

Apache2::Module::add(__PACKAGE__, \@directives);

sub Widget {

  my ($cfg, $parms, $arg) = @_;

  $cfg->{widget} = $arg;
}
```

Pick Me!

```
<Location /foo>
    Widget bleep
</Location>
```

```
Syntax error on line 45 of httpd.conf:
Invalid command 'Widget', perhaps mis-spelled
or defined by a module not included in the
server configuration
```

Where?

```
<Location /foo>
    Widget bleep
</Location>

Syntax error on line 45 of httpd.conf:
Widget not allowed here
```

Data Validation

```
Widget bleep
```

```
Syntax error on line 45 of httpd.conf:
Widget must be On or Off
```

Power to the People

- With just a (very) few lines of code we have allowed Apache to validate
  - what our directive is called
  - where it is allowed to live
  - what arguments it is allowed to have
- Now we need to
  - glean the arguments from the config
  - use them in our `Perl*Handler`

```
package My::Directive;

use Apache2::Module ();

use Apache2::Const -compile => qw(FLAG RSRC_CONF);

my @directives = (
  { name         => 'Widget',
    req_override => Apache2::Const::RSRC_CONF,
    args_how     => Apache2::Const::FLAG,
  },
);

Apache2::Module::add(__PACKAGE__, \@directives);

sub Widget {

  my ($cfg, $parms, $arg) = @_;

  $cfg->{widget} = $arg;
}
```

```
package My::Directive;

use Apache2::RequestRec ();
use Apache2::ServerRec ();

sub handler {

  my $r = shift;

  my $cfg = Apache2::Module::get_config(__PACKAGE__,
                                        $r->server,
                                        $r->per_dir_config);

  my $widget = $cfg->{widget};

  ...
}
```

per_dir_config()

- Um... `$r->per_dir_config`?

Total Access

- 1.0 remains incomplete
- 2.0 offers complete API access
  - Apache structure accessors and mutators
  - Apache functions
  - Apache phases
- There is even `$r->assbackwards()`

Output Filters

- New in Apache 2.0
- Allow you to post-process content *after* the content phase has run
- mod_perl has been able to filter content for years
  - limited to mod_perl generated content
    - mod_perl can do lots, like CGI and SSI
- Output filters let you filter *everything*
  - no matter who generates the content

```
package My::Filter;

use Apache2::Filter ();

use Apache2::Const qw(OK);

sub handler {

  my $f = shift;

  while ($f->read(my $buffer, 1024)) {

    # do something with $buffer

    $f->print($buffer);
  }

  return OK;
}

1;
```

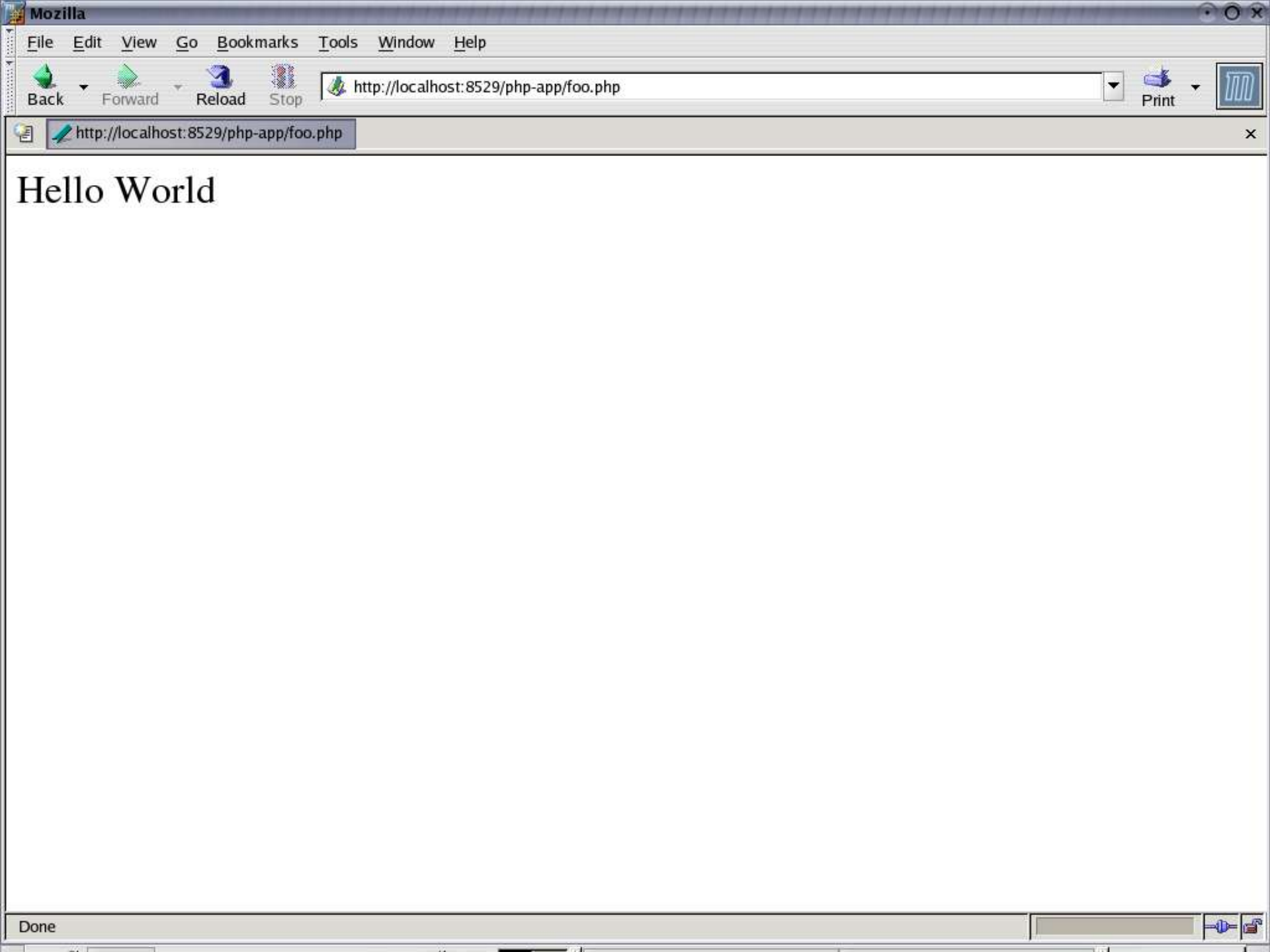
httpd.conf

```
PerlOutputFilterHandler My::Filter
```

Fun with PHP

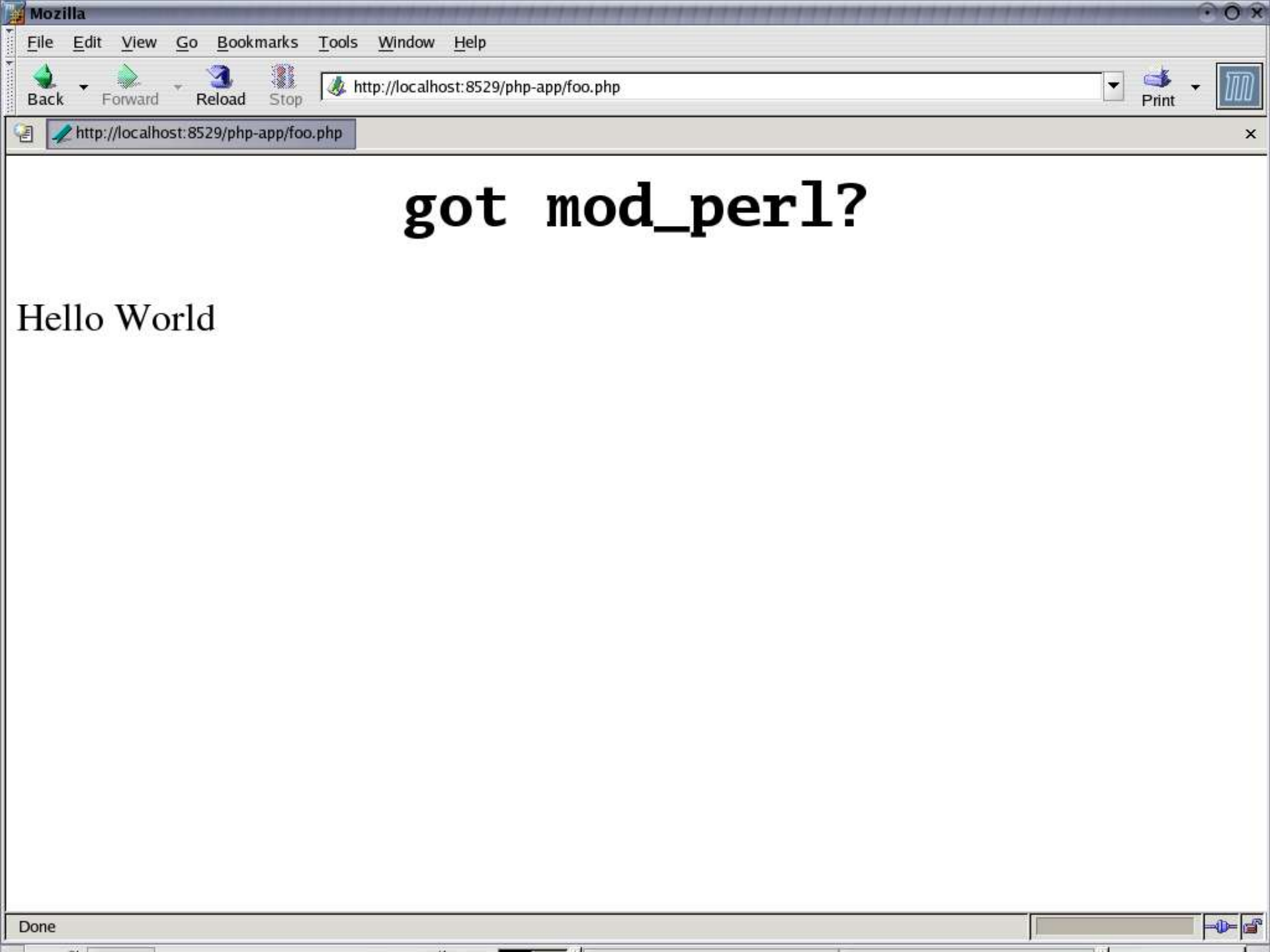
- That filter wasn't all that interesting
- Let's mess with PHP

```
package Apache::Hijack;

use Apache2::Filter ();
use Apache2::RequestRec ();

use Apache2::Const -compile => qw(OK DECLINED);

use strict;

sub handler {

  my $f = shift;
  my $r = $f->r;

  return Apache2::Const::DECLINED
    unless $r->handler eq 'php-script' or
           $r->handler eq 'application/x-httpd-php';

  while ($f->read(my $buffer, 1024)) {
    $buffer =~ s!(<body>)!$1<h1>got mod_perl?</h1>!i;

    $f->print($buffer);
  }

  return Apache2::Const::OK;
}
1;
```


Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8529/php-app/foo.php
Print
http://localhost:8529/php-app/foo.php
Hello World
Done

httpd.conf

```
PerlOutputFilterHandler Apache::Hijack
```


Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8529/php-app/foo.php
Print
http://localhost:8529/php-app/foo.php
got mod_perl?
Hello World
Done

Stacked Perl Handlers

- mod_perl supports the idea of stacked handlers

```
PerlFixupHandler My::One My::Two
```

- Idea borrowed from Apache
- mod_perl 1.0 didn't get it quite right

Stacked C Handlers

- In Apache, how the module list is traversed depends on the phase
- Some phases run until the handler list is exhausted
  - fixups
  - loggers
- Some phases terminate on the first `OK`
  - translation
  - authentication

Problems in 1.0

- Calling mechanism in mod_perl 1.0 did not allow for early phase termination via `OK`
- Worse yet, you were misled by the documentation

It Does Matter

- Given

```
PerlAuthenHandler My::All My::Admins
```

- mod_perl 1.0 will call `My::Admins` even if `My::All` succeeds
- Users passed by the first will be failed overall
- 2.0 handler stacks behave properly
  - or at least exactly like stuff in C

Getting it Right

- The following request phases (finally) end on the first handler to return `OK`
  - `PerlTransHandler`
  - `PerlMapToStorageHandler`
  - `PerlAuthenHandler`
  - `PerlAuthzHandler`
  - `PerlTypeHandler`
  - `PerlResponseHandler`

Lost in Translation

- Um... `PerlMapToStorageHandler`?

Apache-Test

- Single best part of mod_perl 2.0
- Framework for testing Apache-based applications
  - mod_perl handlers
  - CGI scripts
  - PHP
  - SOAP servers
  - Apache 1.3 or 2.0

The `test` Target

- With `Apache-Test`, `make test` will
  - configure Apache
  - start Apache
  - execute the tests
  - issue the report
  - stop Apache
- All you need to do is write the tests
  - and get `Apache-Test` working

Slides

- These slides freely available at some long URL you will never remember…

`http://www.modperlcookbook.org/~geoff/slides/YAPC/2005/`

- Linked to from my homepage

`http://www.modperlcookbook.org/~geoff`