

Writing Tests with Apache-Test Part II

Geoffrey Young

`geoff@modperlcookbook.org`

Apache-Test

- Framework for testing Apache-based application components
- Part of the mod_perl ASF project
- Provides tools to make testing Apache related things simple
- Written in Perl
- Familiar interface
 - `Test.pm`, `Test::More`, `LWP`, etc

Last Year...

- I introduced `Apache-Test` mechanics
- Everyone was impressed
- Some people even started using it
- There's so much I didn't tell you
- But first, let's review...

Testing, Testing... 1, 2, 3

1. Perl `Makefile.PL` `foo`
2. Apache `httpd.conf` `foo`
3. Write the tests

Step 1 - Makefile.PL

```
use Apache::TestMM qw(test clean);
use Apache::TestRunPerl ();

# configure tests based on incoming arguments
Apache::TestMM::filter_args();

# generate the test harness (t/TEST)
Apache::TestRunPerl->generate_script();
```


Step 2 – Apache Foo

- Apache needs a basic configuration to service requests
 - `ServerRoot`
 - `DocumentRoot`
 - `ErrorLog`
 - `Listen`
- Apache-Test "intuits" these and creates its own `httpd.conf`
- You augment this with `extra.conf.in`

Prompt

Enter username and password for "cookbook" at jib

User Name:

Password:

☐ Use Password Manager to remember these values.

OK Cancel

t/conf/extra.conf.in

```
Alias /basic @DocumentRoot@
```

```
<Location /basic>  
    Require valid-user  
    AuthType Basic  
    AuthName myrealm
```

```
    PerlAuthenHandler Apache::BasicAuth  
</Location>
```


BasicAuth.pm




```
package My::AuthenHandler;

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

use Apache2::RequestRec ();
use Apache2::Access ();

sub handler {

    my $r = shift;

    # Get the client-supplied credentials.
    my ($status, $password) = $r->get_basic_auth_pw;

    return $status unless $status == Apache2::Const::OK;

    # Perform some custom user/password validation.
    return Apache2::Const::OK if $r->user eq $password;

    # Whoops, bad credentials.
    $r->note_basic_auth_failure;
    return Apache2::Const::HTTP_UNAUTHORIZED;
}

1;
```


Step 3 – The Tests

- Standard Perl foo

```
plan tests => 3;
```

```
ok ($condition);
```

```
etc...
```


t/01basic.t

```
{  
    my $response = GET '/basic/index.html';  
  
    is ($response->code,  
        401,  
        "no valid password entry");  
}  
  
{  
    my $response = GET '/basic/index.html',  
                      username => 'geoff',  
                      password => 'geoff';  
  
    is ($response->code,  
        200,  
        "geoff:geoff allowed to proceed");  
}
```


Testing, Testing... 1, 2, 3

1. Generate the test harness
2. Configure Apache
3. Write the tests
4. Run the tests


```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
-D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3, 1 wallclock secs ( 1.12 cusr + 0.12 csys = 1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```



```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
-D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3, 1 wallclock secs ( 1.12 cusr + 0.12 csys = 1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```



```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
-D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3, 1 wallclock secs ( 1.12 cusr + 0.12 csys = 1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```



```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
-D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3, 1 wallclock secs ( 1.12 cusr + 0.12 csys = 1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```



```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
-D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3, 1 wallclock secs ( 1.12 cusr + 0.12 csys = 1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```



```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
-D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3, 1 wallclock secs ( 1.12 cusr + 0.12 csys = 1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```



```
[geoff@jib perl-authen-handler]$ make test TEST_VERBOSE=1
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -clean
APACHE_TEST_GROUP= APACHE_TEST_HTTPD= APACHE_TEST_PORT= APACHE_TEST_USE
R= APACHE_TEST_APXS= \
/perl/perl-5.8.7/bin/perl -Iblib/arch -Iblib/lib \
t/TEST -bugreport -verbose=1
/apache/2.0.54/prefork/perl-5.8.7/bin/httpd -d /src/devel/yapc/perl-au
then-handler/t -f /src/devel/yapc/perl-authen-handler/t/conf/httpd.conf
-D APACHE2 -D PERL_USEITHREADS
using Apache/2.0.54 (prefork MPM)

waiting 60 seconds for server to start: ...
waiting 60 seconds for server to start: ok (waited 1 secs)
server jib.modperlcookbook.net:8529 started
t/01basic....1..3
ok 1 - no valid password entry
ok 2 - password mismatch
ok 3 - geoff:geoff allowed to proceed
ok
All tests successful.
Files=1, Tests=3, 1 wallclock secs ( 1.12 cusr + 0.12 csys = 1.24 CP
U)
[warning] server jib.modperlcookbook.net:8529 shutdown
```


Client-Side Tests

- The test file acts as a web browser
- The client is scripted to
 - query the server
 - compare server response to expected results
 - indicate success or failure
- In short, the granularity rests in the client
 - that's where all the calls to `ok()` are

Server-Side Tests

- `Apache-Test` provides a mechanism for achieving granularity in the Apache runtime
- Highly magical (not really)
- Let's see it in action...

Apache::SSLLookup

- `mod_ssl` exposes a few optional functions in C
 - `is_https` # true if https://
 - `ssl_var_lookup` # `SSL_CLIENT_VERIFY`
- **Apache::SSLLookup provides Perl glue**
 - `Apache::SSLLookup->new()`
 - `is_https()`
 - `ssl_lookup()`

What to Test?

- Class
 - compiles
- Constructor
 - defined
 - returns an object of the proper class
 - returns an object with proper attributes
- Methods
 - defined
 - do something useful

Where To Test

- Clearly this API needs a real Apache + mod_perl runtime

```
$ perl -MApache::SSLLookup -e0
```

```
Can't load 'Apache/SSLLookup/SSLLookup.so'  
for module Apache::SSLLookup: undefined  
symbol: MP_debug_level
```

- Apache-Test++
- But even here we have choices

Options

- Client-side test
 - do a bunch of stuff on the server and simply return
 - OK if it went OK
 - 500 if it didn't
 - testing in aggregate
- Server-side test
 - much more granular
 - each test can individually pass or fail
- It's all about where you call `ok()`


```
package TestSSL::01new;

use Apache::Test qw(-withtestmore);

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

sub handler {

    my $r = shift;

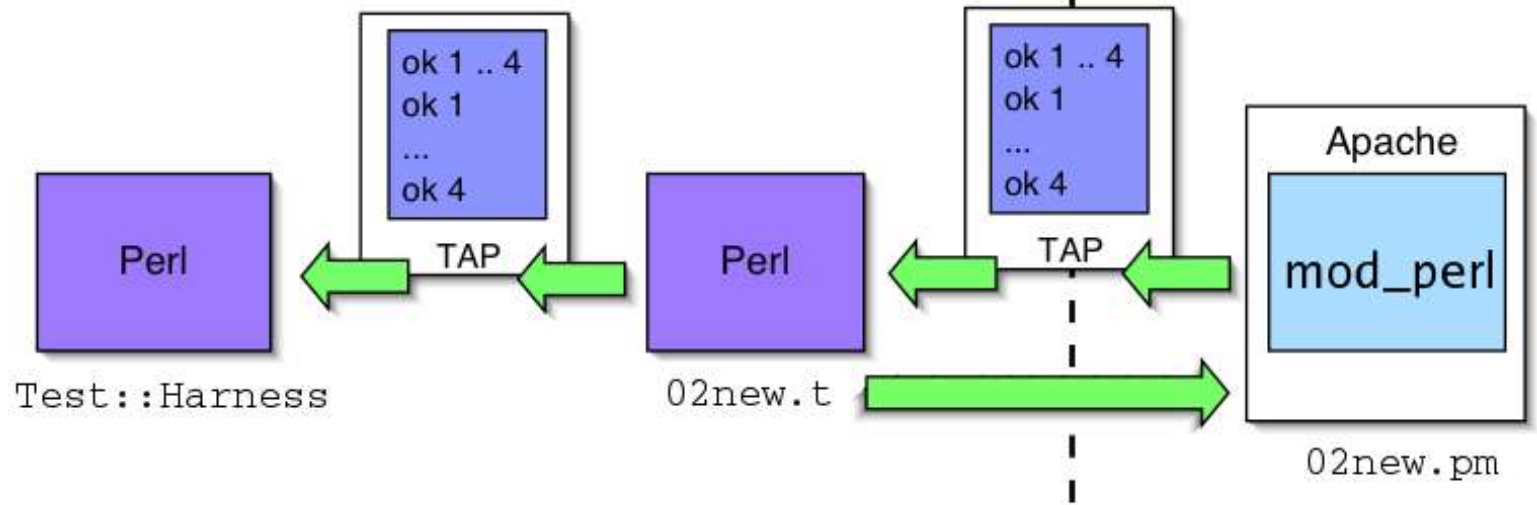
    plan $r, tests => 2;

    {
        use_ok('Apache::SSLLookup');
    }

    {
        can_ok('Apache::SSLLookup', 'new');
    }

    return Apache2::Const::OK
}

1;
```

Magic

- Magical things happen if you follow a specific filesystem pattern
- In our case

`t/response/TestSSL/02new.pm`

automagically generates

`t/ssl/02new.t`

and an entry in

`t/conf/httpd.conf`

t/ssl/02new.t

```
# WARNING: this file is generated, do not edit
# 01: Apache/TestConfig.pm:898
# 02: /Apache/TestConfig.pm:916
# 03: Apache/TestConfigPerl.pm:138
# 04: Apache/TestConfigPerl.pm:553
# 05: Apache/TestConfig.pm:584
# 06: Apache/TestConfig.pm:599
# 07: Apache/TestConfig.pm:1536
# 08: Apache/TestRun.pm:501
# 09: Apache/TestRunPerl.pm:80
# 10: Apache/TestRun.pm:720
# 11: Apache/TestRun.pm:720
# 12: t/TEST:28
```

```
use Apache::TestRequest 'GET_BODY_ASSERT';
print GET_BODY_ASSERT "/TestSSL__02new";
```


t/conf/httpd.conf

```
<Location /TestSSL__02new>  
    SetHandler modperl  
    PerlResponseHandler TestSSL::02new  
</Location>
```


t/conf/httpd.conf

```
<Location /TestSSL__02new>  
    SetHandler modperl  
    PerlResponseHandler TestSSL::02new  
</Location>
```



```

sub handler {

    plan shift, tests => 7;

    use_ok('Apache::SSLLookup');

    can_ok('Apache::SSLLookup', 'new');

    eval { my $r = Apache::SSLLookup->new() };

    like ($@, qr/Usage:/, 'new() requires arguments');

    eval { my $r = Apache::SSLLookup->new({}) };

    like ($@, qr/invoked by a 'unknown' object with no 'r' key/,
          'new() requires an object');

    eval { my $r = Apache::SSLLookup->new(bless {}, 'foo') };

    like ($@, qr/invoked by a 'foo' object with no 'r' key/,
          'new() requires an Apache2::RequestRec object');

    my $r = Apache::SSLLookup->new($r);

    isa_ok($r, 'Apache::SSLLookup');
    isa_ok($r, 'Apache2::RequestRec');

    return Apache2::Const::OK;
}

```



```

sub handler {

    plan shift, tests => 7;

    use_ok('Apache::SSLLookup');

    can_ok('Apache::SSLLookup', 'new');

    eval { my $r = Apache::SSLLookup->new() };

    like ($@, qr/Usage:/, 'new() requires arguments');

    eval { my $r = Apache::SSLLookup->new({}) };

    like ($@, qr/invoked by a 'unknown' object with no 'r' key/,
          'new() requires an object');

    eval { my $r = Apache::SSLLookup->new(bless {}, 'foo') };

    like ($@, qr/invoked by a 'foo' object with no 'r' key/,
          'new() requires an Apache2::RequestRec object');

    my $r = Apache::SSLLookup->new($r);

    isa_ok($r, 'Apache::SSLLookup');
    isa_ok($r, 'Apache2::RequestRec');

    return Apache2::Const::OK;
}

```



```
sub handler {  
    my $r = shift;  
  
    plan $r, tests => 3;  
  
    {  
        use_ok('Apache::SSLLookup');  
    }  
  
    {  
        can_ok('Apache::SSLLookup', 'is_https');  
    }  
  
    {  
        $r = Apache::SSLLookup->new($r);  
  
        ok(defined $r->is_https,  
            'is https returned a defined value');  
    }  
  
    return Apache2::Const::OK;  
}
```


More Magic

- Take this code

```
package My::NeedsToBePreLoaded;

BEGIN {
    Apache->push_handlers(
        PerlChildInitHandler => sub { ... }
    );
}
```

- We need PerlModule or the PerlChildInitHandler won't run
—or startup.pl, of course

The Solution

```
package TestMy::NeedsToBePreloaded::01compile;

use Apache::Test qw(-withtestmore);

sub handler {

    plan shift, tests => 1;

    use_ok (My::NeedsToBePreloaded);

    return 0;
}
1;

__DATA__
# make sure that the PerlChildInitHandler
# invokes the BEGIN block
PerlModule My::NeedsToBePreloaded
```


t/conf/httpd.conf

```
# make sure that the PerlChildInitHandler
# invokes the BEGIN block
PerlModule My::NeedsToBePreloaded

<Location /TestMy__NeedsToBePreloaded__01compile>
    SetHandler perl-script
    PerlHandler TestMy::NeedsToBePreloaded::01compile
</Location>
```


____DATA____

- Stuff Apache foo into ____DATA____
- Apache-Test converts to `httpd.conf`
 - with substitutions, like `@DocumentRoot@`
- Localized to generated `<Location>`
 - `PerlModule`, `Alias`, etc are special
- Use `<Base></Base>` to force top-level configuration

EOP

- Apache-Test is an Equal Opportunity Platform
- You can now use Apache-Test to test PHP too!
 - server-side
 - client-side

t/response/TestAPI/01php.php

```
<?php
```

```
include 'test-more.php';
```

```
plan(7);
```

```
ok(true, 'ok() pass');
```

```
ok(false, 'ok() fail');
```

```
is('foo', 'foo', 'is() pass');
```

```
isnt('foo', 'bar', 'isnt() pass');
```

```
like('foo', '/oo/', 'like() pass');
```

```
unlike('foo', '/oo/', 'unlike() fail');
```

```
diag('diag()');
```

```
?>
```


t/response/TestAPI/01php.php

```
<?php
```

```
include 'test-more.php';
```

```
plan(7);
```

```
ok(true, 'ok() pass');
```

```
ok(false, 'ok() fail');
```

```
is('foo', 'foo', 'is() pass');
```

```
isnt('foo', 'bar', 'isnt() pass');
```

```
like('foo', '/oo/', 'like() pass');
```

```
unlike('foo', '/oo/', 'unlike() fail');
```

```
diag('diag()');
```

```
?>
```



```
t/api/01php....1..7
```

```
ok 1 - ok() pass
```

```
not ok 2 - ok() fail
```

```
# Failed test (t/response/TestAPI/01php.php at line 8)
```

```
ok 3 - is() pass
```

```
ok 4 - isnt() pass
```

```
ok 5 - like() pass
```

```
not ok 6 - unlike() fail
```

```
# Failed test (t/response/TestAPI/01php.php at line 16)
```

```
# 'foo'
```

```
# matches '/oo/'
```

```
ok 7 - cmp_ok() pass
```

```
# diag()
```

```
# Looks like you failed 2 tests of 7.
```

```
FAILED tests 2, 6
```

```
Failed 2/7 tests, 71.43% okay
```

```
Failed Test Stat Wstat Total Fail Failed List of Failed
```

```
-----
```

```
t/api/01php.t 7 2 28.57% 2 6
```

```
Failed 1/1 test scripts, 0.00% okay. 2/7 subtests failed, 71.43% okay.
```

```
[warning] server jib.modperlcookbook.net:8529 shutdown
```

```
[ error] error running tests (please examine t/logs/error_log)
```

```
make: *** [run_tests] Error 1
```



```
t/api/01php....1..7
```

```
ok 1 - ok() pass
```

```
not ok 2 - ok() fail
```

```
# Failed test (t/response/TestAPI/01php.php at line 8)
```

```
ok 3 - is() pass
```

```
ok 4 - isnt() pass
```

```
ok 5 - like() pass
```

```
not ok 6 - unlike() fail
```

```
# Failed test (t/response/TestAPI/01php.php at line 16)
```

```
# 'foo'
```

```
# matches '/oo/'
```

```
ok 7 - cmp_ok() pass
```

```
# diag()
```

```
# Looks like you failed 2 tests of 7.
```

```
FAILED tests 2, 6
```

```
Failed 2/7 tests, 71.43% okay
```

```
Failed Test Stat Wstat Total Fail Failed List of Failed
```

```
-----  
-----
```

```
t/api/01php.t 7 2 28.57% 2 6
```

```
Failed 1/1 test scripts, 0.00% okay. 2/7 subtests failed, 71.43% okay.
```

```
[warning] server jib.modperlcookbook.net:8529 shutdown
```

```
[ error] error running tests (please examine t/logs/error_log)
```

```
make: *** [run_tests] Error 1
```



```
t/api/01php....1..7
```

```
ok 1 - ok() pass
```

```
not ok 2 - ok() fail
```

```
# Failed test (t/response/TestAPI/01php.php at line 8)
```

```
ok 3 - is() pass
```

```
ok 4 - isnt() pass
```

```
ok 5 - like() pass
```

```
not ok 6 - unlike() fail
```

```
# Failed test (t/response/TestAPI/01php.php at line 16)
```

```
# 'foo'
```

```
# matches '/oo/'
```

```
ok 7 - cmp_ok() pass
```

```
# diag()
```

```
# Looks like you failed 2 tests of 7.
```

```
FAILED tests 2, 6
```

```
Failed 2/7 tests, 71.43% okay
```

```
Failed Test Stat Wstat Total Fail Failed List of Failed
```

```
-----  
-----
```

```
t/api/01php.t 7 2 28.57% 2 6
```

```
Failed 1/1 test scripts, 0.00% okay. 2/7 subtests failed, 71.43% okay.
```

```
[warning] server jib.modperlcookbook.net:8529 shutdown
```

```
[ error] error running tests (please examine t/logs/error_log)
```

```
make: *** [run_tests] Error 1
```


Where is Apache-Test?

- mod_perl 2.0
- CPAN

More Information

- `perl.com`
 - <http://www.perl.com/pub/a/2003/05/22/testing.html>
- Apache-Test **tutorial**
 - <http://perl.apache.org/docs/general/testing/testing.html>
- Apache-Test **manpages**
- *mod_perl Developer's Cookbook*
 - <http://www.modperlcookbook.org/>
- All the tests in the `perl-framework` part of the `httpd-test` **project**

Slides

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

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

- Linked to from my homepage

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