Rock Your Testing World with `Devel::Cover`

Geoffrey Young

Geoffrey.Young@Ticketmaster.com

Oh, Yeaahh!

- Hopefully, everyone here hath drunk of the testing kool-aid
- Coverage is the next step in your test-fu

Coverage?

- Ok, so you have some tests...
- How much of the code are your tests actually exercising?
- Code coverage measures tested program logic
- `Devel::Cover` adds the magic

Devel::Cover

- On CPAN
- Eerily simple to use
- Complete voodoo inside
- Paul Johnson rocks

Coverage in 2 Easy Steps

```
$ HARNESS_PERL_SWITCHES=-MDevel::Cover make test
$ cover
```

Coverage in 3 Easy Steps

```
$ cover -delete
$ HARNESS_PERL_SWITCHES=-MDevel::Cover make test
$ cover
```

- That's too much work for me

Coverage in 1 Easy Step

- Savvy modules make life even better

```
$ make testcover
```

- `testcover`-aware modules
  - `Module::Build`
  - `Apache-Test`
  - `ExtUtils::MakeMaker::Coverage`

Don't like dependencies?

- do it yourself...

Makefile.PL

```
sub MY::test {

  my $test = shift->MM::test(@_);

  if (eval { require Devel::Cover }) {
    $test .= <<EOF;
testcover ::
        cover -delete
        HARNESS_PERL_SWITCHES=-MDevel::Cover make test
        cover
EOF
  }

  return $test;
}
```

That's It!

- I've just told you everything you need to know to use `Devel::Cover`
- Go forth
- The rest of the talk...
  - understanding `Devel::Cover` data
  - code coverage concepts
  - real-world examples

3 Coverage Metrics

- Statement
  - simple execution

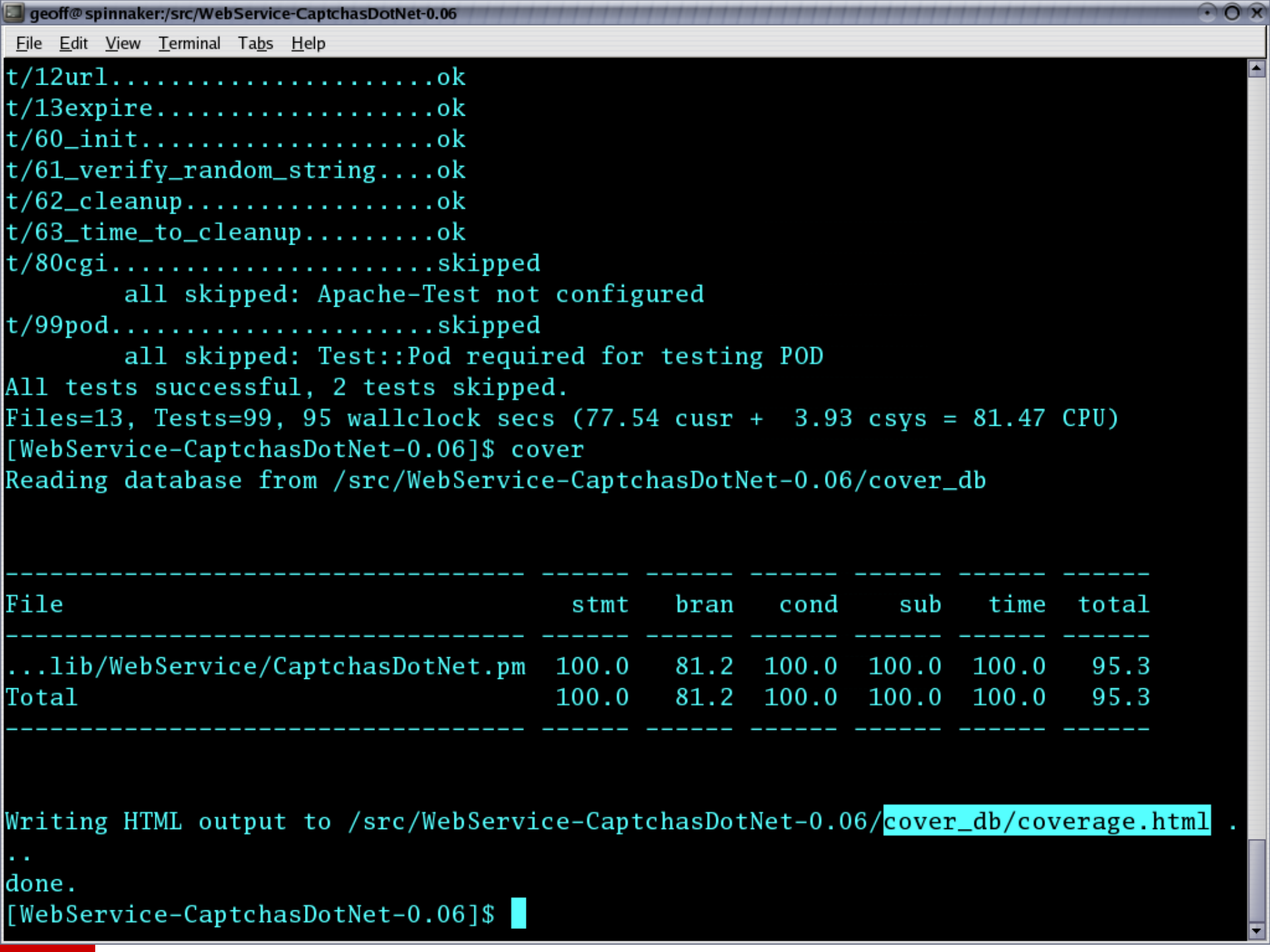
```
print "ok" if $foo || $bar;
```

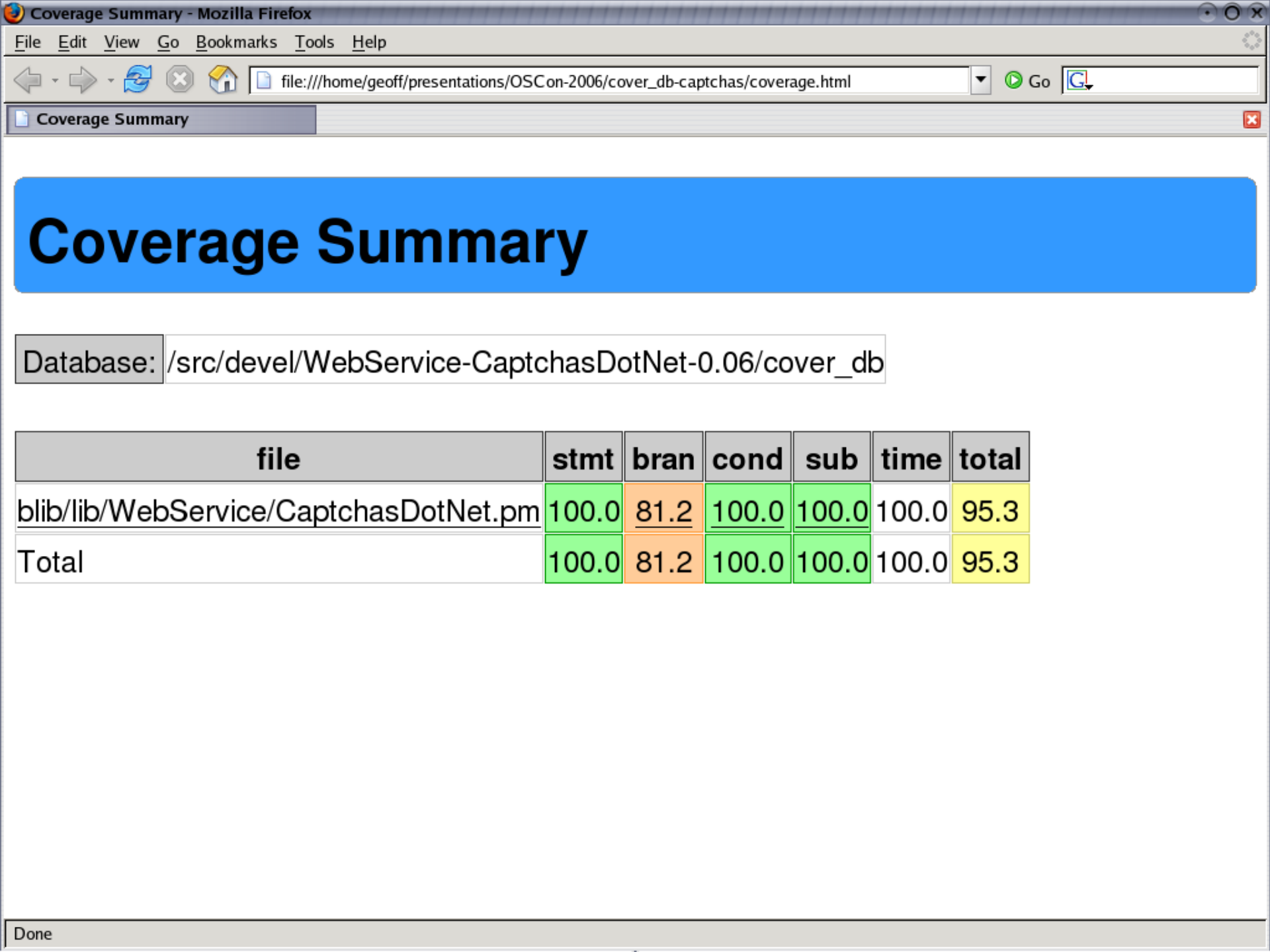
- Branch
  - logical pathways

```
print "ok" if $foo || $bar;
```

- Condition
  - elements of a branch

```
print "ok" if $foo || $bar;
```


geoff@spinnaker:/src/WebService-CaptchasDotNet-0.06
File Edit View Terminal Tabs Help
t/12url.....ok
t/13expire.....ok
t/60_init.....ok
t/61_verify_random_string....ok
t/62_cleanup.....ok
t/63_time_to_cleanup.....ok
t/80cgi.....skipped
all skipped: Apache-Test not configured
t/99pod.....skipped
all skipped: Test::Pod required for testing POD
All tests successful, 2 tests skipped.
Files=13, Tests=99, 95 wallclock secs (77.54 cusr + 3.93 csys = 81.47 CPU)
[WebService-CaptchasDotNet-0.06]$ cover
Reading database from /src/WebService-CaptchasDotNet-0.06/cover_db
File stmt bran cond sub time total
...lib/WebService/CaptchasDotNet.pm 100.0 81.2 100.0 100.0 100.0 95.3
Total 100.0 81.2 100.0 100.0 100.0 95.3
Writing HTML output to /src/WebService-CaptchasDotNet-0.06/cover_db/coverage.html .
..
done.
[WebService-CaptchasDotNet-0.06]$


Coverage Summary - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///home/geoff/presentations/OSCon-2006/cover_db-captchas/coverage.html
Go
Coverage Summary
Coverage Summary
Database: /src/devel/WebService-CaptchasDotNet-0.06/cover_db
file stmt bran cond sub time total
blib/lib/WebService/CaptchasDotNet.pm 100.0 81.2 100.0 100.0 100.0 95.3
Total 100.0 81.2 100.0 100.0 100.0 95.3
Done

File Coverage: blib/lib/WebService/CaptchasDotNet.pm - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

file:///home/geoff/presentations/OSCon-2006/cover_db-captchas/blib-lib-WebService-Captchas

Go

File Coverage: blib/lib/WebServic...

File Coverage

File:	blib/lib/WebService/CaptchasDotNet.pm
Coverage:	95.3%

line	stmt	bran	cond	sub	time	code
1						`package WebService::CaptchasDotNet;`
2						
3	11 11 11			11	548 106 152	`use 5.006;`
4						
5	11 11 11			11	436 87 393	`use strict;`
6	11 11			11	486 98	`use warnings FATAL => qw(all);`

Done

File Coverage: blib/lib/WebService/CaptchasDotNet.pm - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

file:///home/geoff/presentations/OSCon-2006/cover_db-captchas/blib-lib-WebService-Captchas

Go

File Coverage: blib/lib/WebServi...

71						#----------
72						sub verify {
73						
74	18			18	1275	my $self = shift;
75						
76	18				206	my ($input, $random) = @_;
77						
78	18				3023	my $secret = $self->{_secret};
79						
80						# basic sanity checking
81						
82	18	100	100		1084	unless ($secret && $random && $input && $input =~ m/^[a-z]{6}$/) {
83	12	50			152	print STDERR join ' ', 'WebService::CaptchasDotNet - ',
84						"insufficient data for verify()\n"
85						if $DEBUG;
86						
87	12				220	return;
						}

file:///home/geoff/presentations/OSCon-2006/cover_db-captchas/blib-lib-WebService-CaptchasDotNet-pm--subroutine.html#L74

Branch Coverage

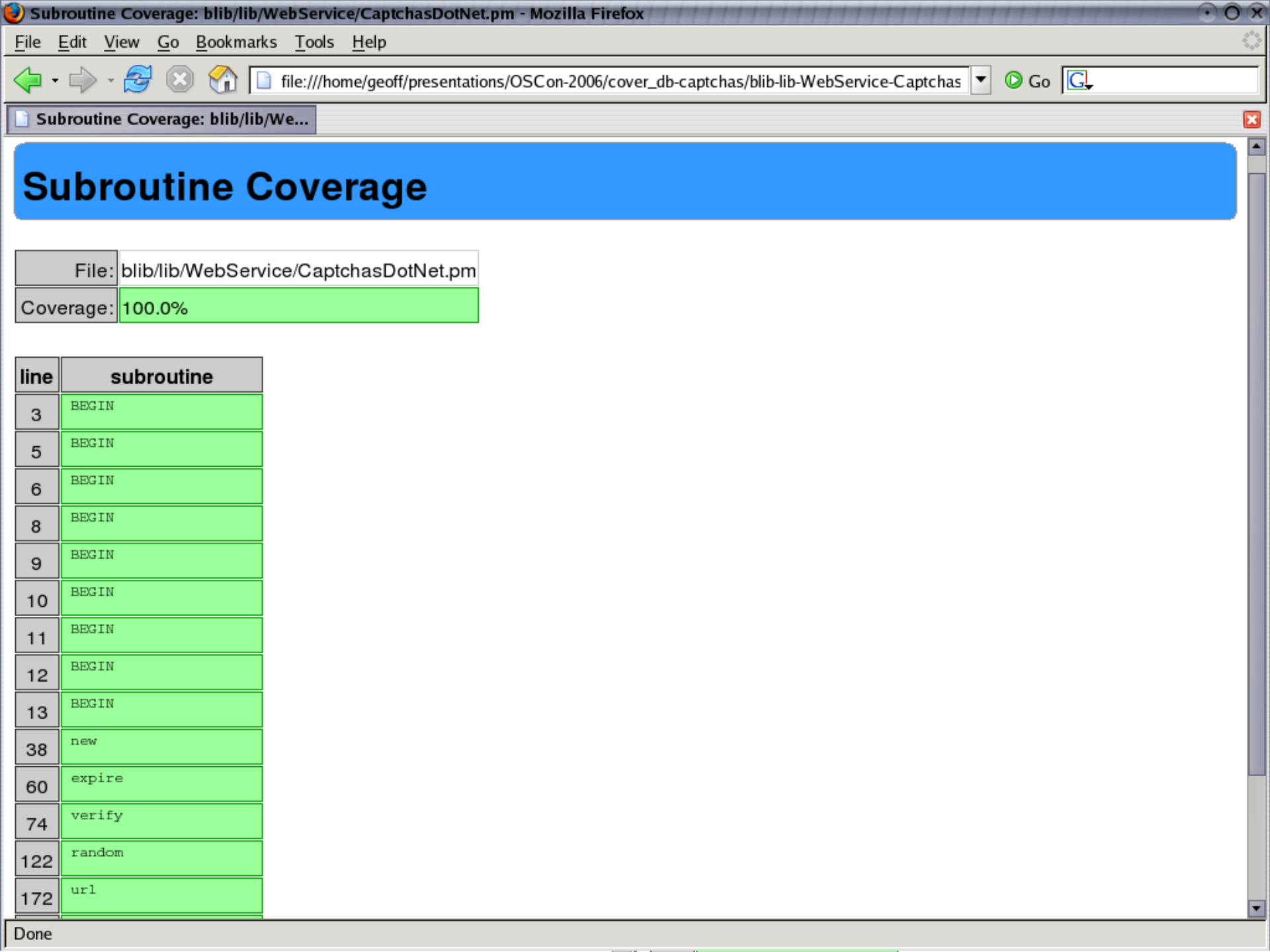
File:	blib/lib/WebService/CaptchasDotNet.pm
Coverage:	81.2%

line	%	coverage		branch
62	100	T	F	`if @_`
82	100	T	F	`unless ($secret and $random and $input and $input =~ /^[a-z]{6}$/)`
83	50	T	F	`if $DEBUG`
93	100	T	F	`unless $file`
105	100	T	F	`if ($input eq $captcha)`
143	100	T	F	`if (-e $file)`
144	50	T	F	`if $DEBUG`
153	100	T	F	`unless ($fh)`
154	50	T	F	`if $DEBUG`
194	100	T	F	`unless -d $tmp`
213	100	T	F	`if $random`
216	100	T	F	`unless ($random)`

Condition Coverage

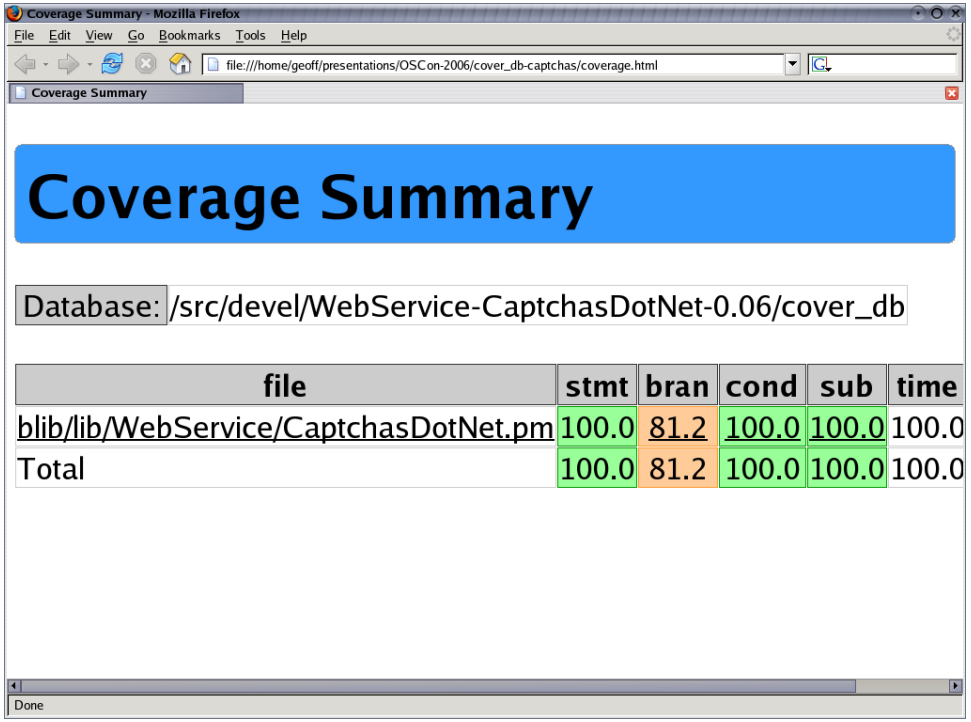
File:	blib/lib/WebService/CaptchasDotNet.pm
Coverage:	100.0%

line	%	coverage	condition
42	100	A dec 0 0 1 1	`$args{'expire'} || 3600`
82	100	A B C D dec 0 X X X 0 1 0 X X 0 1 1 0 X 0 1 1 1 0 0 1 1 1 1 1	`$secret and $random and $input and $input =~ /^[a-z]{6}$/`
293	100	A B dec 0 X 0	`$mtime and $mtime + $$self{'_expire'} < time`


Subroutine Coverage: blib/lib/WebService/CaptchasDotNet.pm - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///home/geoff/presentations/OSCon-2006/cover_db-captchas/blib-lib-WebService-Captchas
Go
Subroutine Coverage: blib/lib/We...
Subroutine Coverage
File: blib/lib/WebService/CaptchasDotNet.pm
Coverage: 100.0%
line subroutine
3 BEGIN
5 BEGIN
6 BEGIN
8 BEGIN
9 BEGIN
10 BEGIN
11 BEGIN
12 BEGIN
13 BEGIN
38 new
60 expire
74 verify
122 random
172 url
Done

PHB

- As soon as pictures are involved, you're pretty much doomed



- "Why isn't it *all* green?"

For the Record...

1. "Wow, we have tests?"
2. "Your initiative is impressive."
3. "Good work."
4. "I didn't know you could even do that"
5. "Take tomorrow off."
6. "Your new code coverage technique is unstoppable."

...*Then*

1. "If it's not too much trouble, sir, after your so obviously selfless toils to improve our understanding of what was before an unnavigable mire of jumbled code, could you be so kind to me, sir, as to explain why, despite your best efforts, the results are less than the perfection you, yourself, aim to achieve, even if we as a company do not."

Translation

- "What do the numbers actually mean?"
- "What should our goals be?"

- Coverage tells you *nothing* about code
  - specifically, whether it works or not
- Says little about test completeness
  - not what was tested well, just what was tested
- More a measure of test gaps
  - where you need tests
- Coverage is never the whole story

Official Ticketmaster site. Alternative Rock tickets - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.ticketmaster.com/browse?category=60&tm_link=tm_music_b_60&root=10001

Go

Official Ticketmaster site. Alterna...

ticketmaster

Retail Locations | International | Help

HOME | TICKETEXCHANGE | TICKETALERTS | MY ACCOUNT

MUSIC | SPORTS | ARTS & THEATER | FAMILY

Enter Artist, Team, or Venue | Search | United States | see local venues and events | Search the Web | Go

Welcome | Log In

Categories

All Music (7227)
Alternative Rock (1618)
Classical (166)
Comedy (730)
Country and Folk (604)
Dance/Electronic (56)
Hard Rock/Metal (234)
Jazz and Blues (368)
Latin (156)
Miscellaneous (21)
New Age and Spiritual (49)
R&B/Urban Soul (287)
Rap and Hip-Hop (217)
Rock and Pop (2177)
World Music (174)
More Concerts (370)

Refine Your Results

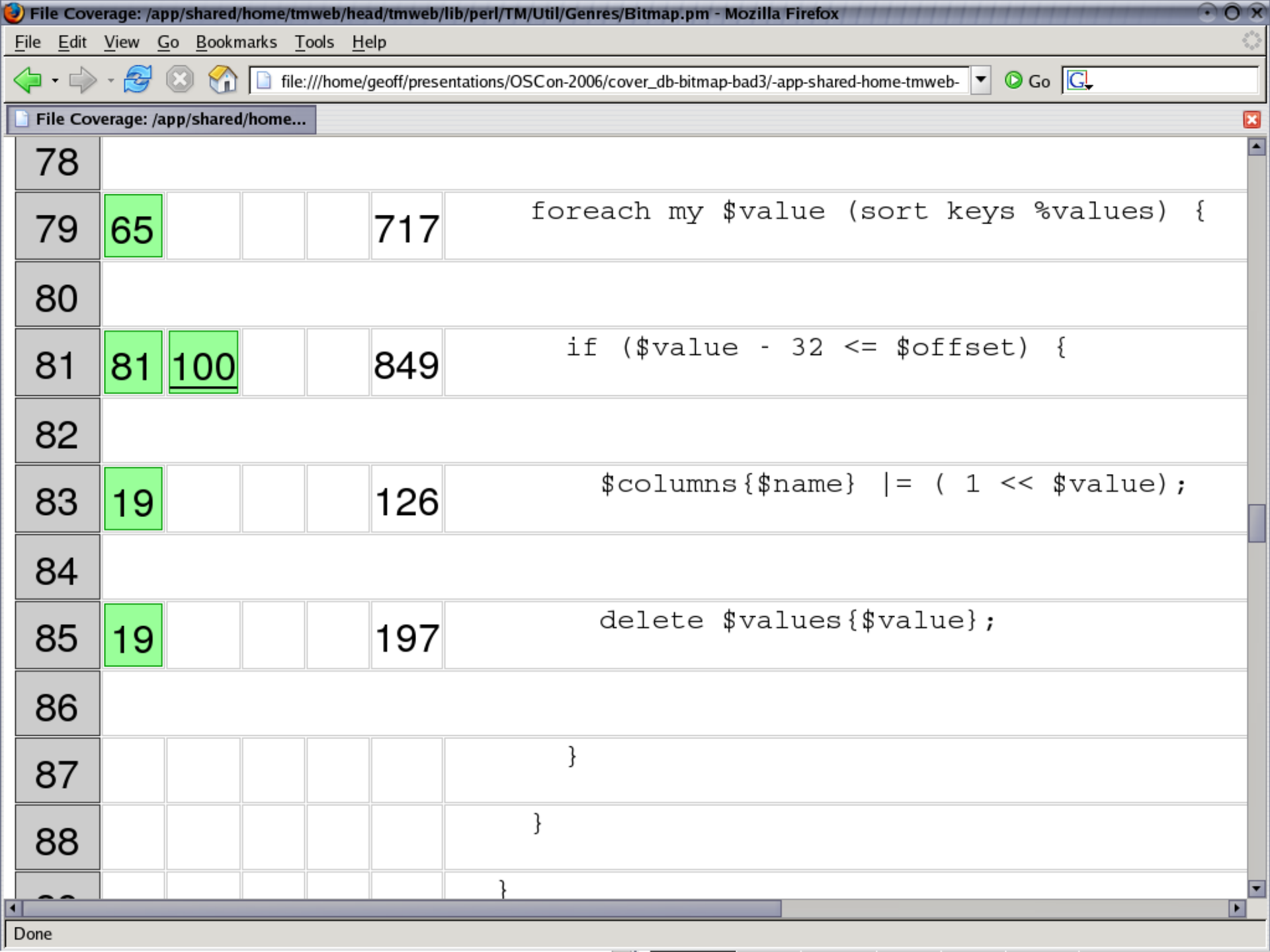
Enter City or Zip Code | Alternative Rock | Next 60 Days | Search

Showing 1 - 30 of 1618 Alternative Rock events found in the United States

1 - 30 of 1618 next

Mon, 07/10/06 05:00 PM	**Chiodos (Acoustic)**	Chain Reaction Anaheim, CA	Find Tickets On Sale Now
Mon, 07/10/06 05:00 PM	**Mindless Self Indulgence**	First Avenue Minneapolis, MN	Find Tickets On Sale Now
Mon, 07/10/06 06:00 PM	**Lucky Boys Confusion**	The Basement Columbus, OH	Find Tickets On Sale Now
Mon, 07/10/06 06:00 PM	**The Moaners Plus Chittlin'**	Beachland Tavern Cleveland, OH	Find Tickets On Sale Now
Mon, 07/10/06 06:30 PM	**Taking Back Sunday / Angels and Airwaves**	The Arena At Gwinnett Center Atlanta, GA	Find Tickets On Sale Now
Mon, 07/10/06 07:00 PM	**Candlebox**	The Blue Note Columbia, MO	Find Tickets On Sale Now

Done


File Coverage: /app/shared/home/tmweb/head/tmweb/lib/perl/TM/Util/Genres/Bitmap.pm - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///home/geoff/presentations/OSCon-2006/cover_db-bitmap-bad3/-app-shared-home-tmweb-
Go
File Coverage: /app/shared/home...
78
79 65 717 foreach my $value (sort keys %values) {
80
81 81 100 849 if ($value - 32 <= $offset) {
82
83 19 126 $columns{$name} |= (1 << $value);
84
85 19 197 delete $values{$value};
86
87 }
88 }
}
Done

Don't Just Test – Test Well

- Don't be led into a false sense of security by coverage reports
- A pretty image won't save you from incomplete or poor tests

What *is* it Good For?

- Paired with good testing practices, code coverage is very powerful
- Coverage shows which parts of your code you still need to test
  - "untested code does not exist"
- Can also uncover unsavory code

Branch Coverage: /app/shared/home/tmweb/geoff/webcore-utc_refactor_2006/lib/perl/TMCS/Model.pm - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

file:///home/geoff/presentations/OSCon-2006/cover_db-model/-app-shared-home-tmweb-geoff-v

Go

Branch Coverage: /app/shared/h...

488	100	T	F	if (not defined &$method)
489	100	T	F	if (0 == $dirty_style) { }
	100	T	F	elsif (1 == $dirty_style) { }
	100	T	F	elsif (2 == $dirty_style) { }
494	100	T	F	if (@_)
497	100	T	F	if ref $lang or not defined $lang
499	50	T	F	if not defined $lang
504	100	T	F	if $$args{'lang'} and $$args{'lang'} eq 'all'
507	100	T	F	if (exists $$args{'lang'} and ref $$args{'lang'} eq 'ARRAY') { }
	100	T	F	elsif (exists $$args{'lang'} and not ref $$args{'lang'} and defined $$
512	100	T	F	unless $lang and ref $lang eq 'ARRAY'
518	100	T	F	unless $lang and ref $lang eq 'ARRAY'
523	100	T	F	if exists $$self{'data'}{$field}{$language}
526	50	T	F	unless defined &$method
533	100	T	F	if (@_)

Done

```
throw TMCS::Exception(text => "...", modify_depth => 1)
  if (ref($lang) || !defined $lang);

throw TMCS::Exception(text => "...", modify_depth => 1)
  if (not defined $lang);
```

```
if(not defined &$method) {

  ...

     *$method = sub {...} unless defined &$method;
}
```

How Much Green is Enough?

- Some people will claim arbitrary goals
  - "... you should aim to provide at least 80% coverage in your code"
- Untested code is untested code
  - "Untested code does not exist"
- 100% coverage is impractical
  - for a number of reasons
- Aim for 100% *understood* coverage

Impractical?

- Yeah, 100% is impractical
  - some say it's sometimes impossible
- Sometimes you don't really care about testing an execution path
  - simple debug statements
  - `$self` in object oriented code
- `Devel::Cover` has its own set of "features"

File Coverage: blib/lib/WebService/CaptchasDotNet.pm - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

file:///home/geoff/presentations/OSCon-2006/cover_db-captchas/blib-lib-WebService-Captchas

Go

File Coverage: blib/lib/WebServi...

71						#----------
72						sub verify {
73						
74	18			18	1275	my $self = shift;
75						
76	18				206	my ($input, $random) = @_;
77						
78	18				3023	my $secret = $self->{_secret};
79						
80						# basic sanity checking
81						
82	18	100	100		1084	unless ($secret && $random && $input && $input =~ m/^[a-z]{6}$/) {
83	12	50			152	print STDERR join ' ', 'WebService::CaptchasDotNet - ',
84						"insufficient data for verify()\n"
85						if $DEBUG;
86						
87	12				220	return;
						}

file:///home/geoff/presentations/OSCon-2006/cover_db-captchas/blib-lib-WebService-CaptchasDotNet-pm--subroutine.html#L74

Condition Coverage: /home/geoff/src/webcore/lib/perl/TMCS/Model.pm - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop file:///src/tm/cover_db/-home-geoff-src-webcore-lib-perl-TMCS-Model-pm--condition.html Print

Condition Coverage: /home/geoff/sr...

	67	A	B	dec	`ref $self || $self`
		0	0	0	
		0	1	1	
		1	X	1	
	67	A	B	dec	`$args{'data'} || $self`
		0	0	0	
		0	1	1	
		1	X	1	
793	67	A	B	dec	`ref $self || $self`
		0	0	0	
		0	1	1	
		1	X	1	
818	67	A	B	dec	`ref $self || $self`
		0	0	0	
		0	1	1	
		1	X	1	
822	100	A	B	dec	`$_[0] && ($_[0] eq 'y' || $_[0] eq 'Y' || $_[0] > 0)`
		0	X	0	
		1	0	0	

Done

File Coverage: blib/lib/My/Bug.pm - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

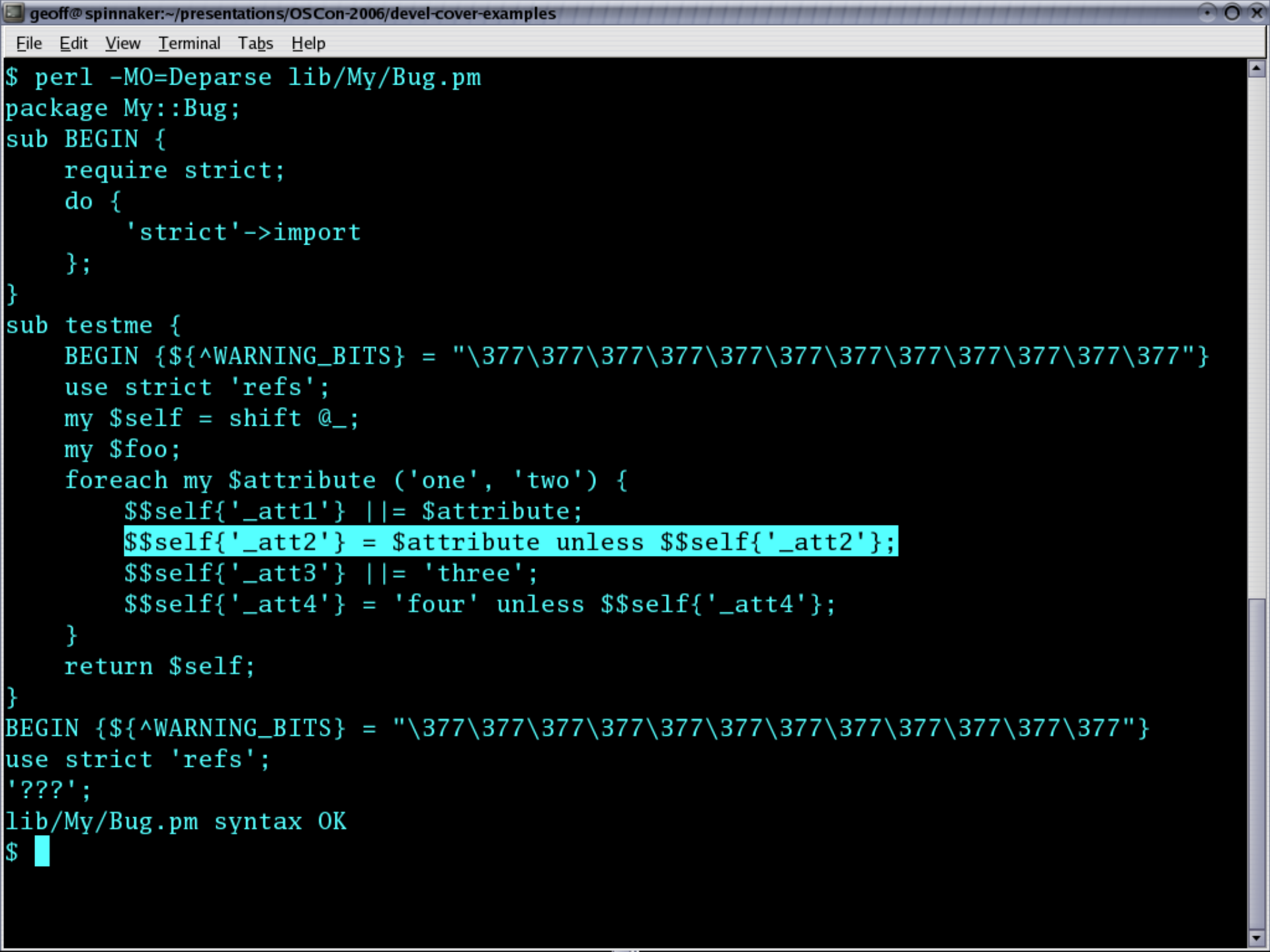
file:///home/geoff/presentations/OSCon-2006/devel-cover-examples/cover_db/blib-lib-My-Bug-p

Go

File Coverage: blib/lib/My/Bug.pm

line	stmt	bran	cond	sub	time	code
12	1				14	`foreach my $attribute (qw(one two)) {`
13						
14	2		67		55	`$self->{_att1} ||= $attribute;`
15						
16	2	100			63	`$self->{_att2} || ($self->{_att2} = $attribute);`
17						
18	2		100		46	`$self->{_att3} ||= 'three';`
19						
20	2	100			53	`$self->{_att4} ||  ($self->{_att4} = 'four');`
21						
22						`}`
23						
24	1				20	`return $self;`
25						`}`

Done


geoff@spinnaker:~/presentations/OSCon-2006/devel-cover-examples
File Edit View Terminal Tabs Help
$ perl -MO=Deparse lib/My/Bug.pm
package My::Bug;
sub BEGIN {
    require strict;
    do {
        'strict'->import
    };
}
sub testme {
    BEGIN {${^WARNING_BITS} = "\377\377\377\377\377\377\377\377\377\377\377\377"}
    use strict 'refs';
    my $self = shift @_;
    my $foo;
    foreach my $attribute ('one', 'two') {
        $$self{'_att1'} ||= $attribute;
        $$self{'_att2'} = $attribute unless $$self{'_att2'};
        $$self{'_att3'} ||= 'three';
        $$self{'_att4'} = 'four' unless $$self{'_att4'};
    }
    return $self;
}
BEGIN {${^WARNING_BITS} = "\377\377\377\377\377\377\377\377\377\377\377\377"}
use strict 'refs';
'???';
lib/My/Bug.pm syntax OK
$

Aesthetically (Un)Pleasing

- Don't let (un)pretty graphs distract you
- 100% coverage is nice, but not always practical

100% Understood Coverage

- Ideally, tests should get as close to 100% as possible
  - don't shoot for 80% or some arbitrary value
- The missing percent should be easily explained
  - non-production code
  - clear API violations
- Colors don't matter

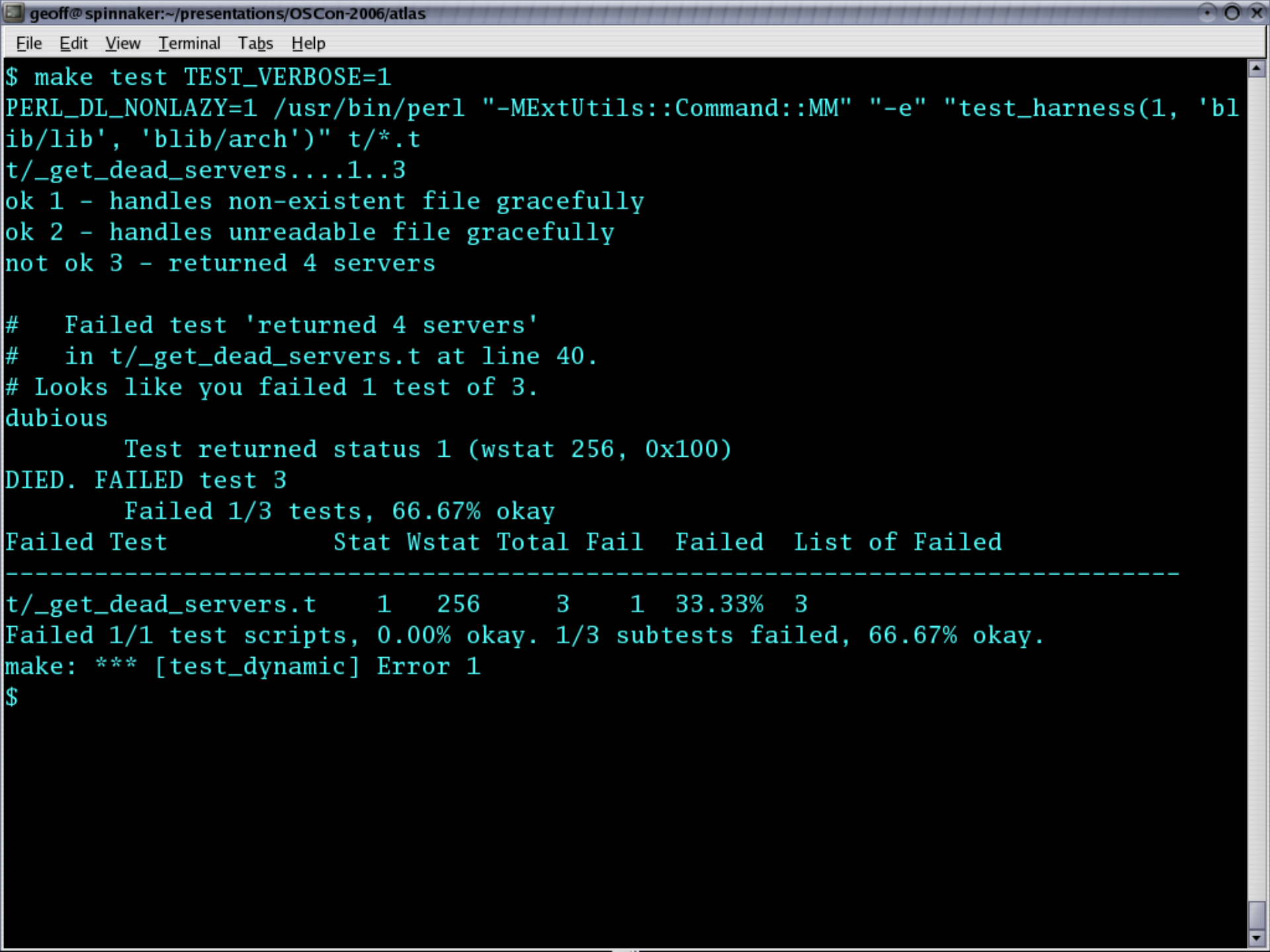
Debugging with Coverage

- `Devel::Cover` adds value to your tests
  - worth the time just for that
- Also a worthy debugging aid

```
my $LAST_CHECK = 0;
my @DEAD_SERVERS = ();

sub _get_dead_servers {

 if ($LAST_CHECK < (my $mtime = (stat($TM::dsfile))[9])) {

      $LAST_CHECK = $mtime;

      my $fh = gensym;
      if (open($fh, $TM::dsfile)) {
          my(@list) = <$fh>;
          close $fh;
          chomp(@list);
          @DEAD_SERVERS = grep { length($_) > 0 } @list;
      } else {
          @DEAD_SERVERS = ();
      }
  }

  return @DEAD_SERVERS;
}
```

My Test Philosophy

- I like to test three things with files
  - No file
  - Unreadable file
  - Readable file
- `httpd` environments lend themselves to unreadable files
  - server starts as `root`
  - requests served by `nobody`


geoff@spinnaker:~/presentations/OSCon-2006/atlas
File Edit View Terminal Tabs Help
$ make test TEST_VERBOSE=1
PERL_DL_NONLAZY=1 /usr/bin/perl "-MExtUtils::Command::MM" "-e" "test_harness(1, 'bl
ib/lib', 'blib/arch')" t/*.t
t/_get_dead_servers....1..3
ok 1 - handles non-existent file gracefully
ok 2 - handles unreadable file gracefully
not ok 3 - returned 4 servers
Failed test 'returned 4 servers'
in t/_get_dead_servers.t at line 40.
Looks like you failed 1 test of 3.
dubious
Test returned status 1 (wstat 256, 0x100)
DIED. FAILED test 3
Failed 1/3 tests, 66.67% okay
Failed Test Stat Wstat Total Fail Failed List of Failed
t/_get_dead_servers.t 1 256 3 1 33.33% 3
Failed 1/1 test scripts, 0.00% okay. 1/3 subtests failed, 66.67% okay.
make: *** [test_dynamic] Error 1
$

What Went Wrong?

- It's not immediately obvious
- The old way to debug this might be to add a bunch of print statements
  - or maybe whip out the debugger
- Since you already have the tests in place, `Devel::Cover` can help

File Coverage: blib/lib/TM.pm - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

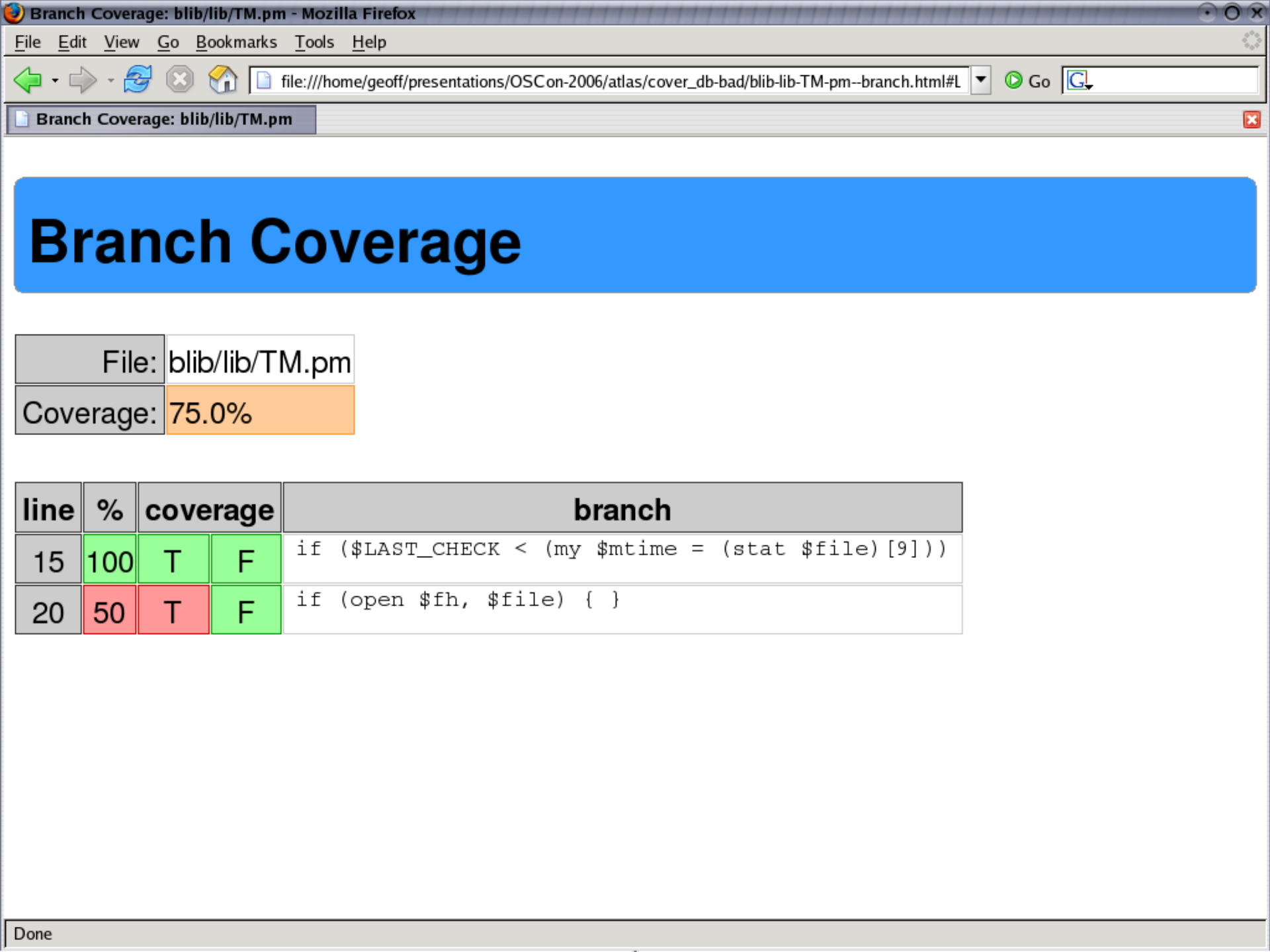
file:///home/geoff/presentations/OSCon-2006/atlas/cover_db-bad/blib-lib-TM-pm.html

Go

File Coverage: blib/lib/TM.pm

line	stmt	bran	cond	sub	time	code
15	3	100		3	188	if ($LAST_CHECK < (my $mtime = (stat($TM::file))[9])) {
16						
17	1				10	$LAST_CHECK = $mtime;
18						
19	1				17	my $fh = gensym;
20	1	50			180	if (open($fh, $TM::file)) {
21	0				0	my(@list) = <$fh>;
22	0				0	close $fh;
23	0				0	chomp(@list);
24	0 0				0 0	@DEAD_SERVERS = grep { length($_) > 0 } @list;
25						} else {
26	1				17	@DEAD_SERVERS = ();
27						}

Done


Branch Coverage: blib/lib/TM.pm - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
file:///home/geoff/presentations/OSCon-2006/atlas/cover_db-bad/blib-lib-TM-pm--branch.html#L
Go
Branch Coverage: blib/lib/TM.pm
Branch Coverage
File: blib/lib/TM.pm
Coverage: 75.0%
line % coverage branch
15 100 T F if ($LAST_CHECK < (my $mtime = (stat $file)[9]))
20 50 T F if (open $fh, $file) { }
Done

Non-Existent File Test

```
local $TM::dbfile = 'deadserver.test';

{
  my @dead_servers = TM->_get_dead_servers();

  is (scalar @dead_servers,
      0,
      'handles non-existent file gracefully');
}
```

Non-Existent File Test

```
local $TM::dbfile = 'deadserver.test';

{
  my @dead_servers = TM->_get_dead_servers();

  is (scalar @dead_servers,
      0,
      'handles non-existent file gracefully');
}
```

Unreadable File Test

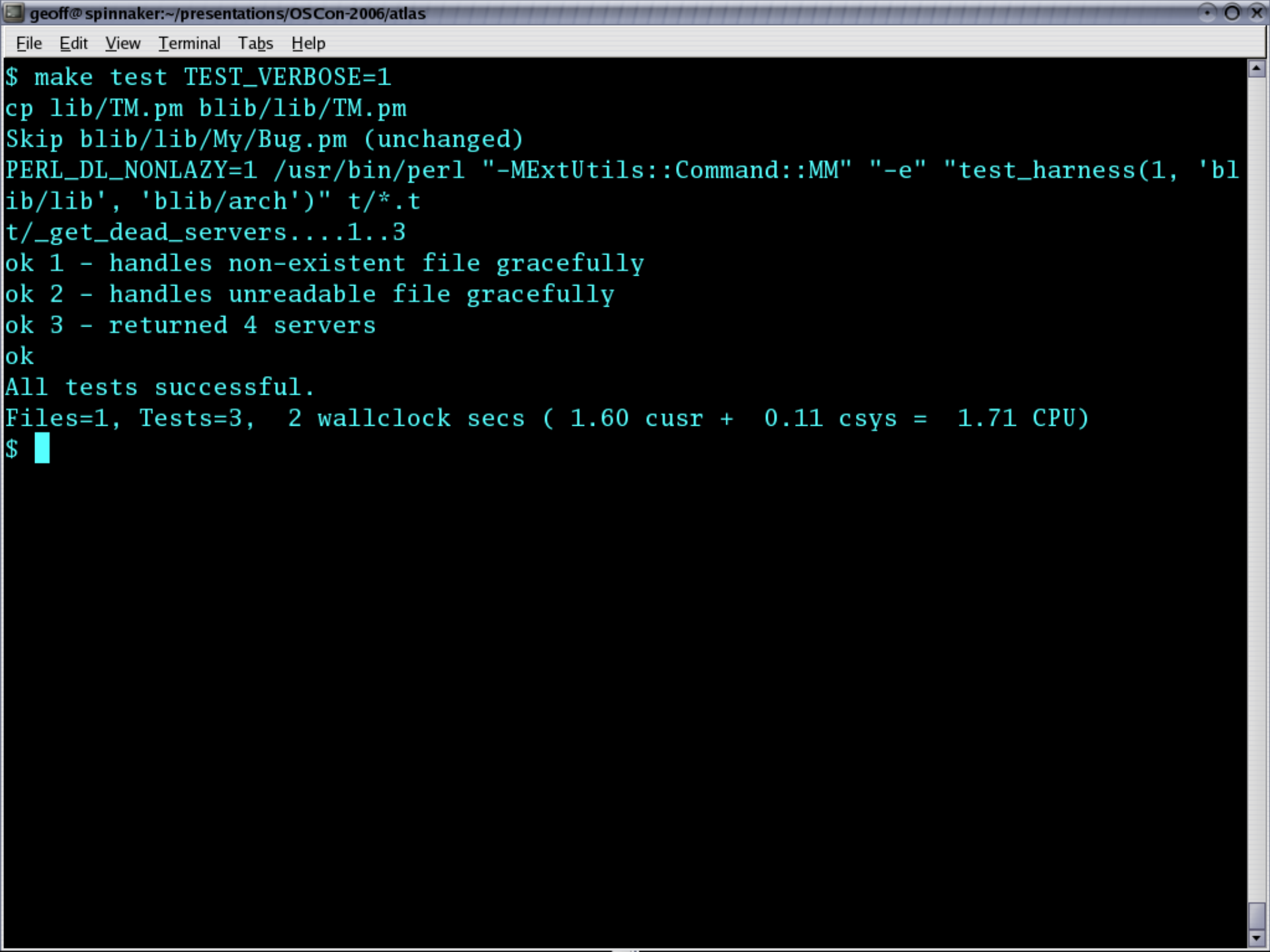
```
my @servers = qw(test1 test2 test3 test4);

[ stuff that creates and populates the file ]

{
  chmod 0111, $TM::dbfile;

  my @dead_servers = TM->_get_dead_servers();

  is (scalar @dead_servers,
      0,
      'handles unreadable file gracefully');
}
```

Readable File Test

```
my @servers = qw(test1 test2 test3 test4);

{
  chmod 0644, $TM::file;

  my @dead_servers = TM->_get_dead_servers();

  ok (eq_array(\@dead_servers, \@servers),
      'returned 4 servers');
}
```

Why Does It Fail?

- `chmod` does not update the `mtime`
- `$LAST_CHECK` represents a *failed* check
- Now you get to decide whether this is a bug or a feature
  - is the test or the code wrong?
- I decided it is a bug
  - you want to be able to `chmod` a file without restarting the server


geoff@spinnaker:~/presentations/OSCon-2006/atlas
File Edit View Terminal Tabs Help
$ make test TEST_VERBOSE=1
cp lib/TM.pm blib/lib/TM.pm
Skip blib/lib/My/Bug.pm (unchanged)
PERL_DL_NONLAZY=1 /usr/bin/perl "-MExtUtils::Command::MM" "-e" "test_harness(1, 'bl
ib/lib', 'blib/arch')" t/*.t
t/_get_dead_servers....1..3
ok 1 - handles non-existent file gracefully
ok 2 - handles unreadable file gracefully
ok 3 - returned 4 servers
ok
All tests successful.
Files=1, Tests=3, 2 wallclock secs (1.60 cusr + 0.11 csys = 1.71 CPU)
$

File Coverage: blib/lib/TM.pm - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

file:///home/geoff/presentations/OSCon-2006/atlas/cover_db-good/blib-lib-TM-pm.html

File Coverage: blib/lib/TM.pm

line	stmt	bran	cond	sub	time	code
15	3	100		3	204	if ($LAST_CHECK < (my $mtime = (stat($TM::file))[9])) {
16						
17	2				20	$LAST_CHECK = $mtime;
18						
19	2				31	my $fh = gensym;
20	2	100			372	if (open($fh, $TM::file)) {
21	1				136	my(@list) = <$fh>;
22	1				44	close $fh;
23	1				16	chomp(@list);
24	1 5				13 58	@DEAD_SERVERS = grep { length($_) > 0 } @list;
25						} else {
26	1				12	@DEAD_SERVERS = ();
27	1				17	undef $LAST_CHECK;

Done

got core?

- `Devel::Cover` mostly "just works"
- But when it doesn't it pretty much just dumps core
- Figuring out why is generally an exercise best left to Paul
- XP practices ease the pain
  - code, test, code, test

Boom!

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

BEGIN {
    foreach my $atom (*year, *month, *day) {



        *{$atom} = sub($$) { ... };
    }
}
```

Devel::Cover "Compliant"

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

BEGIN {
    foreach my $atom (qw(year month day)) {

        no strict qw(refs);

        *{$atom} = sub($$) { ... };
    }
}
```

ETOOLITTLETIME

- `Devel::Cover` options
  - `-MDevel::Cover=db,$(TOPDIR)/cover_db`
  - `-MDevel::Cover=+ignore,.t$,`
- Coverage for XS code
  - `$ man gcov2perl`
  - `$ cover -test`
- `httpd` integration
  - `Apache-Test` magic

Resources

- `Devel::Cover` sources

  `http://search.cpan.org/dist/Devel-Cover/`

- Code coverage articles

  `http://en.wikipedia.org/wiki/Code_coverage`

- Kathy Sierra's "girl code" blog entry

  `http://xrl.us/girlcode`

- These slides

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