

Welcome!

Quality Assurance in PHP Projects

Sebastian Bergmann
<http://sebastian-bergmann.de/>

September 15th 2008

Who I am



- Sebastian Bergmann
- Involved in the PHP Project since 2000
- Developer of PHPUnit
- Author, Consultant, Coach, Trainer

Quality Assurance in PHP Projects

Topics

- PHPUnit
 - Writing and running tests
 - Organizing tests into suites and groups
 - Refactoring, Test-Driven and Behaviour-Driven Development
- Selenium
 - Recording and running tests with Selenium IDE
 - Integrating PHPUnit with Selenium RC

“Hot Topics” in PHP's History

From a Keynote by Derick Rethans



- Make it work
- Make it fast
- Make it scale
- Make it secure

Now that we know how to build applications that "just work", are fast and scalable, as well as secure, the next logical step is to implement processes and use techniques that help assure that the software works correctly throughout the software's lifecycle.

Why test?

- Companies develop more and more enterprise-critical applications with PHP.
- Tests help to make sure that these applications work correctly.

What needs testing?

Web Application

– Backend

- Business Logic
- Reusable Components

– Frontend

- Form Processing, Templates, ...
- “Rich Interfaces” with AJAX, JSON, ...
- Feeds, Web Services, ...

And how do we test it?

Web Application

– Backend

- Functional Testing of the business logic with Unit Tests
- Reusable Components
 - External components should have their own unit tests.

– Frontend

- Acceptance Tests or System Tests that are run “in the browser”
- Testing of feeds, web services, etc. with unit tests
- Compatibility Tests for Operating System / Browser combinations
- Performance Tests and Security Tests

Testing Software

- Component Tests
 - Executable code fragments, so called *Unit Tests*, test the correctness of parts, *units*, of the software (*system under test*)
- System Tests
 - Conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. (Wikipedia)
- Non-Functional Tests
 - Performance, Stability, Security, ...

Testing Software

- Developer Tests
 - Ensure that the code works correctly
- Acceptance Tests
 - Ensure that the code does what the customer wants

Test Tools

- To make (code) testing viable, good tool support is needed.
- This is where a testing framework such as PHPUnit comes into play.
- Requirements
 - Reusable test environment
 - Strict separation of production code and test code
 - Automatic execution of test code
 - Analysis of the result
 - Easy to learn to use and easy to use

Test Tools for PHP / Web

- Unit Tests
 - PHPUnit
 - PHPT
 - SimpleTest
- System Tests
 - Selenium
 - PHPUnit + Selenium RC
- Non-Functional Tests
 - Performance, Load, Stress, Availability, ...
 - ab, httpperf, JMeter, Grinder, OpenSTA, siege, ...
 - Security
 - Chorizo
 - ratproxy

PHPUnit: Website

<http://www.phpunit.de/>

The screenshot shows a Mozilla Firefox browser window with the address bar set to <http://www.phpunit.de/>. The website header features the PHPUnit logo on the left and a search bar on the right. Below the logo is a navigation menu with links for [Login](#), [Help/Guide](#), [About Trac](#), [Register](#), [Forgot your password?](#), and [Preferences](#). A secondary navigation bar includes [Wiki](#), [Timeline](#), [Roadmap](#), [Browse Source](#), [View Tickets](#), and [Search](#). Below this, there are links for [Start Page](#), [Index](#), [History](#), and [Last Change](#).

PHPUnit

In the last decade, [PHP](#) has developed from a niche language for adding dynamic functionality to small websites to a powerful tool making strong inroads into large-scale Web systems. Critical business logic like this needs to work correctly. But how do you ensure that it does? You [test](#) it, of course.

To make code testing viable, good tool support is needed. This is where PHPUnit comes into play. It is a member of the [xUnit family of testing frameworks](#) and provides both a [framework](#) that makes the [writing of tests](#) easy as well as the functionality to easily [run the tests](#) and [analyse their results](#).

Read more about PHPUnit's features [here](#).

Get Started

- [Install](#) PHPUnit.
- Learn how to [write tests](#).
- Learn how to [run tests](#).
- Learn how to group tests into [test suites](#).
- Learn how to set up [phpUnderControl](#) for [continuous integration](#).
- Read the [documentation](#).

News

News can be found on [Planet PHPUnit](#).

Download in other formats:
[Plain Text](#)

[Features](#)
[Requirements](#)
[Documentation](#)
[IDE Support](#)
History..
[PHPUnit 3.3](#)
[PHPUnit 3.2](#)
[PHPUnit 3.1](#)
[PHPUnit 3.0](#)
Ideas
[SubversionRepository](#)
[Mailinglists and IRC](#)
[Who Uses PHPUnit](#)
[Consulting and Training](#)
[Articles and Presentations](#)
[Copyright](#)

Powered by [Trac 0.11.1](#)
By [Edgewall Software](#).

PHPUnit is Copyright (C) 2002 - 2008 by [Sebastian Bergmann](#).

Done

PHPUnit: Installation

```
sb@vmware ~ % pear channel-discover pear.phpunit.de
Adding Channel "pear.phpunit.de" succeeded
Discovery of channel "pear.phpunit.de" succeeded

sb@vmware ~ % pear install phpunit/phpunit
downloading PHPUnit-3.3.0.tgz ...
Starting to download PHPUnit-3.3.0.tgz (264,782 bytes)
.....done: 264,782 bytes
install ok: channel://pear.phpunit.de/PHPUnit-3.3.0
```

The Bowling Game Kata

Introduction: Scoring Bowling

The game consists of 10 frames as shown below.

- In each frame the player has two opportunities to knock down 10 pins.
- The score for the frame is the total number of pins knocked down, plus bonuses for strikes and spares.

| | | | | | | | | | | | | | | | | |
|---|----|----|----|----|----|----|----|-----|-----|---|---|---|---|---|---|---|
| 1 | 4 | 4 | 5 | 6 | ▲ | 5 | ▲ | 0 | 1 | 7 | ▲ | 6 | ▲ | 2 | ▲ | 6 |
| 5 | 14 | 29 | 49 | 60 | 61 | 77 | 97 | 117 | 133 | | | | | | | |

The Bowling Game Kata

Introduction: Requirements

- Write a class named `BowlingGame` that has two methods
 - `roll($pins)` is called each time the player rolls a ball.
 - The argument is the number of pins knocked down.
 - `score()` is called only at the very end of the game and returns the total score for that game.

The Bowling Game Kata

Introduction: Design

- A game has 10 frames.
 - A frame has 1 or 2 rolls.
 - The 10th frame has 2 or 3 rolls and is different from all the other frames.
- The `score()` method must iterate over all the frames and calculate all their scores.
 - The score for a spare or a strike depends on the frame's successor.

The Bowling Game Kata

The first test

```
<?php
require_once 'BowlingGame.php';

class BowlingGameTest extends PHPUnit_Framework_TestCase
{

}
}
```

The Bowling Game Kata

The first test

```
<?php
require_once 'BowlingGame.php';

class BowlingGameTest extends PHPUnit_Framework_TestCase
{
    public function testScoreForGutterGameIs0()
    {

    }
}
```

The Bowling Game Kata

The first test

```
<?php
require_once 'BowlingGame.php';

class BowlingGameTest extends PHPUnit_Framework_TestCase
{
    public function testScoreForGutterGameIs0()
    {
        $game = new BowlingGame;

        for ($i = 0; $i < 20; $i++) {
            $game->roll(0);
        }

        $this->assertEquals(0, $game->score());
    }
}
```

The Bowling Game Kata

The first test

```
<?php
class BowlingGame
{
    public function roll($pins)
    {
    }

    public function score()
    {
        return 0;
    }
}
```

The Bowling Game Kata

The first test

```
sb@vmware ~ % phpunit BowlingGameTest  
PHPUnit 3.3.0 by Sebastian Bergmann.
```

```
.
```

```
Time: 0 seconds
```

```
OK (1 test, 1 assertions)
```

The Bowling Game Kata

The first test

```
sb@vmware ~ % phpunit --ansi BowlingGameTest  
PHPUnit 3.3.0 by Sebastian Bergmann.
```

```
.
```

```
Time: 0 seconds
```

```
OK (1 test, 1 assertions)
```

Interlude

Test-Driven Development

- Method of designing software, not just a method of testing software
 - Test
 - What do we want X to do?
 - How do we want to tell X to do it?
 - How will we know when X has done it?
 - Code
 - How does X do it?
- Tests drive the development
 - Tests written before code
 - No code without tests

The Bowling Game Kata

The first test

```
sb@vmware ~ % phpunit --skeleton-class BowlingGameTest  
PHPUnit 3.3.0 by Sebastian Bergmann.
```

```
Wrote skeleton for "BowlingGame" to "BowlingGame.php".
```

```
<?php  
class BowlingGame  
{  
    /**  
     * @todo Implement roll().  
     */  
    public function roll()  
    {  
        // Remove the following line when you implement this method.  
        throw new RuntimeException('Not yet implemented.');    }  
  
    /**  
     * @todo Implement score().  
     */  
    public function score()  
    {  
        // Remove the following line when you implement this method.  
        throw new RuntimeException('Not yet implemented.');    }  
}  
?>
```

The Bowling Game Kata

The second test

```
<?php
require_once 'BowlingGame.php';

class BowlingGameTest extends PHPUnit_Framework_TestCase
{
    // ...

    public function testScoreForAllOnesIs20()
    {
        $game = new BowlingGame;

        for ($i = 0; $i < 20; $i++) {
            $game->roll(1);
        }

        $this->assertEquals(20, $game->score());
    }
}
```

The Bowling Game Kata

The second test

```
sb@vmware ~ % phpunit BowlingGameTest
PHPUnit 3.3.0 by Sebastian Bergmann.
```

```
.F
```

```
Time: 0 seconds
```

```
There was 1 failure:
```

```
1) testScoreForAllOnesIs20(BowlingGameTest)
Failed asserting that <integer:0> matches expected value <integer:20>.
/home/sb/BowlingGameTest.php:25
```

```
FAILURES!
```

```
Tests: 2, Assertions: 2, Failures: 1.
```

The Bowling Game Kata

The second test

```
sb@vmware ~ % phpunit --ansi BowlingGameTest  
PHPUnit 3.3.0 by Sebastian Bergmann.
```

```
.F
```

```
Time: 0 seconds
```

```
There was 1 failure:
```

```
1) testScoreForAllOnesIs20(BowlingGameTest)  
Failed asserting that <integer:0> matches expected value <integer:20>.  
/home/sb/BowlingGameTest.php:25
```

```
FAILURES!
```

```
Tests: 2, Assertions: 2, Failures: 1.
```

The Bowling Game Kata

The second test

```
<?php
require_once 'BowlingGame.php';

class BowlingGameTest extends PHPUnit_Framework_TestCase
{
    protected $game;

    protected function setUp()
    {
        $this->game = new BowlingGame;
    }

    // ...

    public function testScoreForAllOnesIs20()
    {
        for ($i = 0; $i < 20; $i++) {
            $this->game->roll(1);
        }

        $this->assertEquals(20, $this->game->score());
    }

    // ...
}
```

The Bowling Game Kata

The second test

```
<?php
require_once 'BowlingGame.php';

class BowlingGameTest extends PHPUnit_Framework_TestCase
{
    // ...

    protected function rollMany($n, $pins)
    {
        for ($i = 0; $i < $n; $i++) {
            $this->game->roll($pins);
        }
    }

    public function testScoreForGutterGameIs0()
    {
        $this->rollMany(20, 0);
        $this->assertEquals(0, $this->game->score());
    }

    public function testScoreForAllOnesIs20()
    {
        $this->rollMany(20, 1);
        $this->assertEquals(20, $this->game->score());
    }
}
```

The Bowling Game Kata

The third test

```
<?php
require_once 'BowlingGame.php';

class BowlingGameTest extends PHPUnit_Framework_TestCase
{
    // ...

    public function testScoreForOneSpareAnd3Is16()
    {
        $this->game->roll(5);
        $this->game->roll(5);
        $this->game->roll(3);
        $this->rollMany(17, 0);
        $this->assertEquals(16, $this->game->score());
    }
}
```

The Bowling Game Kata

The fourth test

```
<?php
require_once 'BowlingGame.php';

class BowlingGameTest extends PHPUnit_Framework_TestCase
{
    // ...

    public function testScoreForOneStrikeAnd3And4Is24()
    {
        $this->game->roll(10);
        $this->game->roll(3);
        $this->game->roll(4);
        $this->rollMany(17, 0);
        $this->assertEquals(24, $this->game->score());
    }
}
```

The Bowling Game Kata

The fifth test

```
<?php
require_once 'BowlingGame.php';

class BowlingGameTest extends PHPUnit_Framework_TestCase
{
    // ...

    public function testScoreForPerfectGameIs300()
    {
        $this->rollMany(12, 10);
        $this->assertEquals(300, $this->game->score());
    }
}
```

The Bowling Game Kata

The fifth test

```
sb@vmware ~ % phpunit BowlingGameTest  
PHPUnit 3.3.0 by Sebastian Bergmann.
```

```
.FFFF
```

```
Time: 0 seconds
```

```
There were 4 failures:
```

```
1) testScoreForAllOnesIs20(BowlingGameTest)
```

```
Failed asserting that <integer:0> matches expected value <integer:20>.  
/home/sb/BowlingGameTest.php:67
```

```
2) testScoreForOneSpareAnd3Is16(BowlingGameTest)
```

```
Failed asserting that <integer:0> matches expected value <integer:16>.  
/home/sb/BowlingGameTest.php:75
```

```
3) testScoreForOneStrikeAnd3And4Is24(BowlingGameTest)
```

```
Failed asserting that <integer:0> matches expected value <integer:24>.  
/home/sb/BowlingGameTest.php:84
```

```
4) testScoreForPerfectGameIs300(BowlingGameTest)
```

```
Failed asserting that <integer:0> matches expected value <integer:300>.  
/home/sb/BowlingGameTest.php:90
```

```
FAILURES!
```

```
Tests: 5, Assertions: 5, Failures: 4.
```

PHPUnit

Agile Documentation

```
sb@vmware ~ % phpunit --testdox BowlingGameTest  
PHPUnit 3.3.0 by Sebastian Bergmann.
```

BowlingGame

```
[x] Score for gutter game is 0  
[ ] Score for all ones is 20  
[ ] Score for one spare and 3 is 16  
[ ] Score for one strike and 3 and 4 is 24  
[ ] Score for perfect game is 300
```

The Bowling Game Kata

Final version of the BowlingGame class

```
<?php
class BowlingGame
{
    protected $rolls = array();

    public function roll($pins)
    {
        $this->rolls[] = $pins;
    }

    // ...
}
```

The Bowling Game Kata

Final version of the BowlingGame class

```
<?php
class BowlingGame
{
    // ...

    protected function isSpare($frameIndex)
    {
        return $this->sumOfPinsInFrame($frameIndex) == 10;
    }

    protected function isStrike($frameIndex)
    {
        return $this->rolls[$frameIndex] == 10;
    }

    protected function sumOfPinsInFrame($frameIndex)
    {
        return $this->rolls[$frameIndex] +
            $this->rolls[$frameIndex + 1];
    }
}
```

The Bowling Game Kata

Final version of the BowlingGame class

```
<?php
class BowlingGame
{
    // ...

    protected function spareBonus($frameIndex)
    {
        return $this->rolls[$frameIndex + 2];
    }

    protected function strikeBonus($frameIndex)
    {
        return $this->rolls[$frameIndex + 1] +
            $this->rolls[$frameIndex + 2];
    }
}
```

The Bowling Game Kata

Final version of the BowlingGame class

```
<?php
class BowlingGame
{
    // ...

    public function score()
    {
        $score      = 0;
        $frameIndex = 0;

        for ($frame = 0; $frame < 10; $frame++) {
            if ($this->isStrike($frameIndex)) {
                $score += 10 + $this->strikeBonus($frameIndex);
                $frameIndex++;
            }

            else if ($this->isSpare($frameIndex)) {
                $score += 10 + $this->spareBonus($frameIndex);
                $frameIndex += 2;
            }

            else {
                $score += $this->sumOfPinsInFrame($frameIndex);
                $frameIndex += 2;
            }
        }

        return $score;
    }
}
```

PHPUnit

Agile Documentation

```
sb@vmware ~ % phpunit --testdox BowlingGameTest  
PHPUnit 3.3.0 by Sebastian Bergmann.
```

BowlingGame

```
[x] Score for gutter game is 0  
[x] Score for all ones is 20  
[x] Score for one spare and 3 is 16  
[x] Score for one strike and 3 and 4 is 24  
[x] Score for perfect game is 300
```

PHPUnit

Code Coverage

```
sb@vmware ~ % phpunit --coverage-html report BowlingGameTest  
PHPUnit 3.3.0 by Sebastian Bergmann.
```

```
.....
```

```
Time: 0 seconds
```

```
OK (5 tests, 5 assertions)
```

```
Generating report, this may take a moment.
```

Organizing Test Suites

Application/

– Package/

- Application_Package_Class
Application/Package/Class.php
- ...

– ...

– Tests/

- AllTests.php
- Package/
 - AllTests.php
 - Application_Package_ClassTest
Application/Tests/Package/ClassTest.php

Organizing Test Suites

Application/Tests/AllTests.php

```
<?php
require_once 'PHPUnit/Framework.php';

require_once 'Application/Tests/Package/AllTests.php';

class AllTests
{
    public static function suite()
    {
        $suite = new PHPUnit_Framework_TestSuite('Project');
        $suite->addTest(Package_AllTests::suite());

        return $suite;
    }
}
?>
```

Organizing Test Suites

Application/Tests/Package/AllTests.php

```
<?php
require_once 'PHPUnit/Framework.php';

require_once 'Application/Tests/Package/ClassTest.php';

class Package_AllTests
{
    public static function suite()
    {
        $suite = new PHPUnit_Framework_TestSuite('Package');
        $suite->addTestSuite('Package_ClassTest');

        return $suite;
    }
}
?>
```

Organizing Test Suites

Application/Tests/Package/ClassTest.php

```
<?php
require_once 'PHPUnit/Framework.php';

require_once 'Application/Package/ClassTest.php';

class Package_ClassTest extends PHPUnit_Framework_TestCase
{
    public function testSomething()
    {
        // ...
    }
}
?>
```

Organizing Test Suites

Running the tests

- Executing `phpunit AllTests` in the Tests directory will run all tests.
- Executing `phpunit AllTests` in the Tests/Package directory will run only the tests for the `Application_Package_*` classes.
- Executing `phpunit ClassTest` in the Tests/Framework directory will run only the tests for the `Application_Package_Class` class (which are declared in the `Package_ClassTest` class).
- Executing `phpunit --filter testSomething ClassTest` in the Tests/Package directory will run only the test named `testSomething` from the `Package_ClassTest` class.

Organizing Test Suites

The @group annotation

```
<?php
class SomeTest extends PHPUnit_Framework_TestCase
{
    /**
     * @group specification
     */
    public function testSomething()
    {
    }

    /**
     * @group regression
     * @group bug2204
     */
    public function testSomethingElse()
    {
    }
}
```

Organizing Test Suites

The @group annotation

```
sb@vmware ~ % phpunit --list-groups SomeTest
PHPUnit 3.3.0 by Sebastian Bergmann.
```

```
Available test group(s):
```

- bug2204
- regression
- specification

```
sb@vmware ~ % phpunit --group bug2204 SomeTest
PHPUnit 3.3.0 by Sebastian Bergmann.
```

```
.
```

```
Time: 0 seconds
```

```
OK (1 test, 1 assertion)
```

Annotations

@dataProvider

```
<?php
class DataTest extends PHPUnit_Framework_TestCase
{
    /**
     * @dataProvider providerMethod
     */
    public function testAdd($a, $b, $c)
    {
        $this->assertEquals($c, $a + $b);
    }

    public function providerMethod()
    {
        return array(
            array(0, 0, 0),
            array(0, 1, 1),
            array(1, 1, 3),
            array(1, 0, 1)
        );
    }
}
```

Annotations

@dataProvider

```
sb@vmware ~ % phpunit DataTest
PHPUnit 3.3.0 by Sebastian Bergmann.

..F.

Time: 0 seconds

There was 1 failure:

1) testAdd(DataTest) with data (1, 1, 3)
Failed asserting that <integer:2> matches expected
value <integer:3>.
/home/sb/DataTest.php:19

FAILURES!
Tests: 4, Assertions: 4, Failures: 1.
```

Annotations

@expectedException

```
<?php
class ExceptionTest extends PHPUnit_Framework_TestCase
{
    /**
     * @expectedException InvalidArgumentException
     */
    public function testException()
    {
    }
}
```

Annotations

`@expectedException`

```
sb@vmware ~ % phpunit ExceptionTest
PHPUnit 3.3.0 by Sebastian Bergmann.

F

Time: 0 seconds

There was 1 failure:

1) testException(ExceptionTest)
Expected exception InvalidArgumentException

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

Annotations

@test

```
<?php
class Specification extends PHPUnit_Framework_TestCase
{
    /**
     * @test
     */
    public function shouldDoSomething()
    {
    }

    /**
     * @test
     */
    public function shouldDoSomethingElse()
    {
    }
}
```

Annotations

@test

```
sb@vmware ~ % phpunit --testdox Specification  
PHPUnit 3.3.0 by Sebastian Bergmann.
```

```
Specification
```

```
[x] Should do something
```

```
[x] Should do something else
```

Annotations

@assert

```
<?php
class Calculator
{
    /**
     * @assert (1, 2) == 3
     */
    public function add($a, $b)
    {
        return $a + $b;
    }

    /**
     * @assert (2, 1) == 1
     */
    public function sub($a, $b)
    {
        return $a - $b;
    }
}
```

Annotations

@assert

```
sb@vmware ~ % phpunit Calculator
PHPUnit 3.3.0 by Sebastian Bergmann.

..

Time: 0 seconds

OK (2 tests, 2 assertions)
```

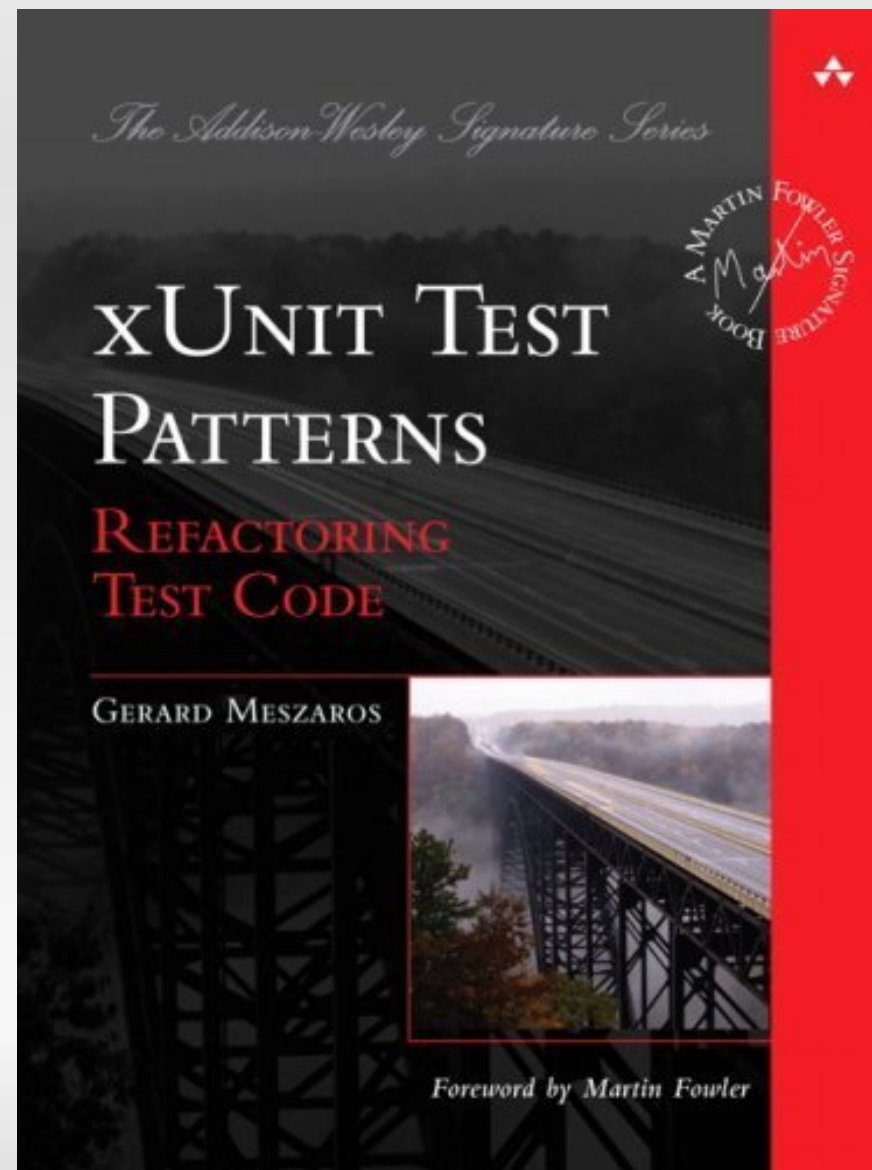
Test Doubles

- *How can we verify logic independently when code it depends on is unusable?*
- *How can we avoid slow tests?*
- **We replace a component on which the SUT depends with a “test-specific equivalent”.**

Test Doubles

Terminology

- Dummy
 - Not the real object
- Fake
 - Usable for testing but not for real job
- Stub
 - Fake that returns canned data
- Spy
 - Stub that records called methods, etc.
- Mock
 - Spy with expectations



Test Doubles

Stubs

```
<?php
require_once 'PHPUnit/Framework.php';

class StubTest extends PHPUnit_Framework_TestCase
{
    public function testStub()
    {

    }
}
?>
```

Test Doubles

Stubs

```
<?php
require_once 'PHPUnit/Framework.php';

class StubTest extends PHPUnit_Framework_TestCase
{
    public function testStub()
    {
        $stub = $this->getMock('SomeClass');

    }
}
?>
```

Test Doubles

Stubs: returnValue()

```
<?php
require_once 'PHPUnit/Framework.php';

class StubTest extends PHPUnit_Framework_TestCase
{
    public function testStub()
    {
        $stub = $this->getMock('SomeClass');
        $stub->expects($this->any())
            ->method('doSomething')
            ->will($this->returnValue('foo'));
    }
}
?>
```

Test Doubles

Stubs: returnValue()

```
<?php
require_once 'PHPUnit/Framework.php';

class StubTest extends PHPUnit_Framework_TestCase
{
    public function testStub()
    {
        $stub = $this->getMock('SomeClass');
        $stub->expects($this->any())
            ->method('doSomething')
            ->will($this->returnValue('foo'));

        // Calling $stub->doSomething() will now return
        // 'foo'.
    }
}
?>
```

Test Doubles

Stubs: returnArgument()

```
<?php
class StubTest extends PHPUnit_Framework_TestCase
{
    public function testReturnArgumentStub()
    {
        $stub = $this->getMock(
            'SomeClass', array('doSomething')
        );

        $stub->expects($this->any())
            ->method('doSomething')
            ->will($this->returnArgument(0));

        // $stub->doSomething('foo') returns 'foo'
        // $stub->doSomething('bar') returns 'bar'
    }
}
```

Test Doubles

Stubs: returnCallback()

```
<?php
class StubTest extends PHPUnit_Framework_TestCase
{
    public function testReturnCallbackStub()
    {
        $stub = $this->getMock(
            'SomeClass', array('doSomething')
        );

        $stub->expects($this->any())
            ->method('doSomething')
            ->will($this->returnCallback('callback'));

        // $stub->doSomething() returns callback(...)
    }
}

function callback() {
    $args = func_get_args();
    // ...
}
```

Test Doubles

Stubs: `throwException()`

```
<?php
class StubTest extends PHPUnit_Framework_TestCase
{
    public function testThrowExceptionStub()
    {
        $stub = $this->getMock(
            'SomeClass', array('doSomething')
        );

        $stub->expects($this->any())
            ->method('doSomething')
            ->will($this->throwException(new Exception));

        // $stub->doSomething() throws Exception
    }
}
```

Test Doubles

Mock Objects

```
<?php
require_once 'PHPUnit/Framework.php' ;

class ObserverTest extends PHPUnit_Framework_TestCase {
    public function testUpdateIsCalledOnce() {

    }
}
?>
```

Test Doubles

Mock Objects

```
<?php
require_once 'PHPUnit/Framework.php';

class ObserverTest extends PHPUnit_Framework_TestCase {
    public function testUpdateIsCalledOnce() {
        $observer = $this->getMock(
            'Observer', array('update')
        );
    }
}
?>
```

Test Doubles

Mock Objects

```
<?php
require_once 'PHPUnit/Framework.php';

class ObserverTest extends PHPUnit_Framework_TestCase {
    public function testUpdateIsCalledOnce() {
        $observer = $this->getMock(
            'Observer', array('update')
        );

        $observer->expects($this->once())
            ->method('update')
            ->with($this->equalTo('something'));

    }
}
?>
```

Test Doubles

Mock Objects

```
<?php
require_once 'PHPUnit/Framework.php';

class ObserverTest extends PHPUnit_Framework_TestCase {
    public function testUpdateIsCalledOnce() {
        $observer = $this->getMock(
            'Observer', array('update')
        );

        $observer->expects($this->once())
            ->method('update')
            ->with($this->equalTo('something'));

        $subject = new Subject;
        $subject->attach($observer);
        $subject->doSomething();
    }
}
?>
```

Selenium

- Selenium
 - Test web applications in a web browser
 - Browser Compatibility Testing
 - System Functional Testing
 - Runs in the browser
- Selenium IDE
 - IDE for Selenium tests
 - Extension for Firefox
 - Record, execute, edit, debug tests in the browser

Selenium

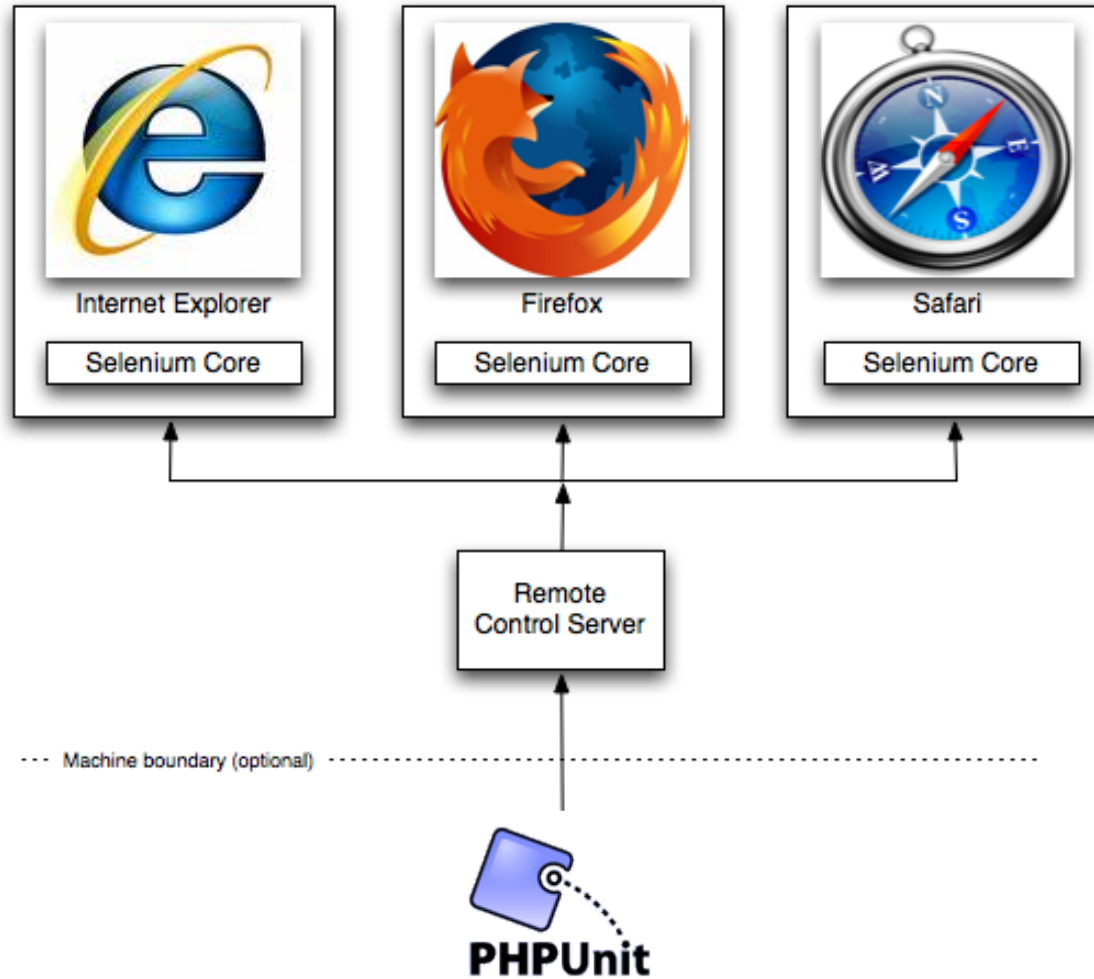
Selenium RC

- Selenium RC
 - Automated execution of Selenium tests
 - Tests can be specified in any language
 - PHP Bindings: PEAR Testing_Selenium
 - PHPUnit natively speaks the Selenium RC protocol
 - One test can be executed on multiple OS / Browser combinations

Selenium

Selenium RC

Windows, Linux, or Mac (as appropriate)...



The End

- Thank you for your interest!
- These slides will be available shortly on <http://sebastian-bergmann.de/talks/>.

License

This presentation material is published under the Attribution-Share Alike 3.0 Unported license.

You are free:

- ✓ **to Share** – to copy, distribute and transmit the work.
- ✓ **to Remix** – to adapt the work.

Under the following conditions:

- **Attribution.** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.

For any reuse or distribution, you must make clear to others the license terms of this work.

Any of the above conditions can be waived if you get permission from the copyright holder.

Nothing in this license impairs or restricts the author's moral rights.