

# Unittesting

- Buzzwords!
- Was sind Unittests?
- PHPUnit im Detail
- DIE Unittests :-)
- Pause
- Testbarer Code in PHP

**SUT**

# SUT

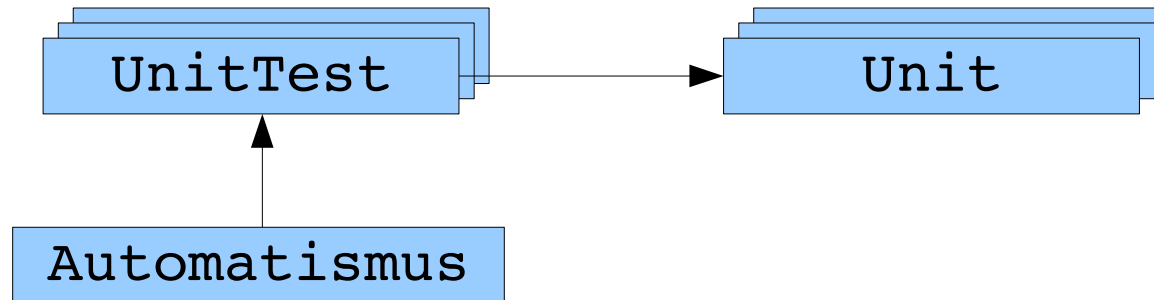
System Under Test /  
Software Under Test

**Fixture(s)**

# Fixture(s)

bekannte und fixe Umgebung

# Was sind Unittests\*?



**VS.**

- `print()`
- `echo()`
- `var_dump()`
- `print_r()`
- ...

\* aka. Modultests, Komponententests

*When ever you are tempted to type something into a print statement or debugger expression, write it as a test instead.*

Martin Fowler




# xUnit Frameworks

- Beck Testing Framework (der Ursprung)
- JUnit (das bekannteste)
- PHPUnit
- CppUnit
- JSUnit
- RUnit
- ...

# PHPUnit im Detail

- `$> phpunit ...`
- Assertions
- Reportgenerierung

```
$> phpunit ...
```

- Konsolen-Tool auf `intra1.rz.kwick.de`
- In `php.ini` `include_path` erweitern:  
`/usr/local/share/php`
- **Achtung!** Nicht aus `pear.php.net` installieren! 

# Assertions\*

- `assertEquals( 'foo', getFoo() )`
- `assertNotEquals( 'bar', getFoo() )`
- `assertTrue( hasFoo() )`
- ...

\* Zusicherungen

# Reportgenerierung

## Das täglich Brot:

```
$> phpunit tests/AllTests.php
```

```
PHPUnit 3.4.8 by Sebastian Bergmann.
```

```
.....I.....S...S..SIS.S.....II.... 60 / 245
.....III.....II.....SS 120 / 245
.....S..... 180 / 245
.....F..... 240 / 245
.....
```

```
Time: 0 seconds, Memory: 17.25Mb
```

```
There was 1 failure:
```

```
1) ResponseTest::testSend
```

```
Failed asserting that two strings are equal.
```

```
--- Expected
```

```
+++ Actual
```

```
@ @
```

```
-A test string foo.
```

```
+A test string.
```

```
FAILURES!
```

```
Tests: 245, Assertions: 1066, Failures: 1, Incomplete: 9, Skipped: 8.
```

# Reportgenerierung

- JUnit-Format
- Test Anything Protocol
- JSON-Format
- Testdox (`--testdox`)
- Code-Coverage (auch Clover)\*

\* benötigt Xdebug

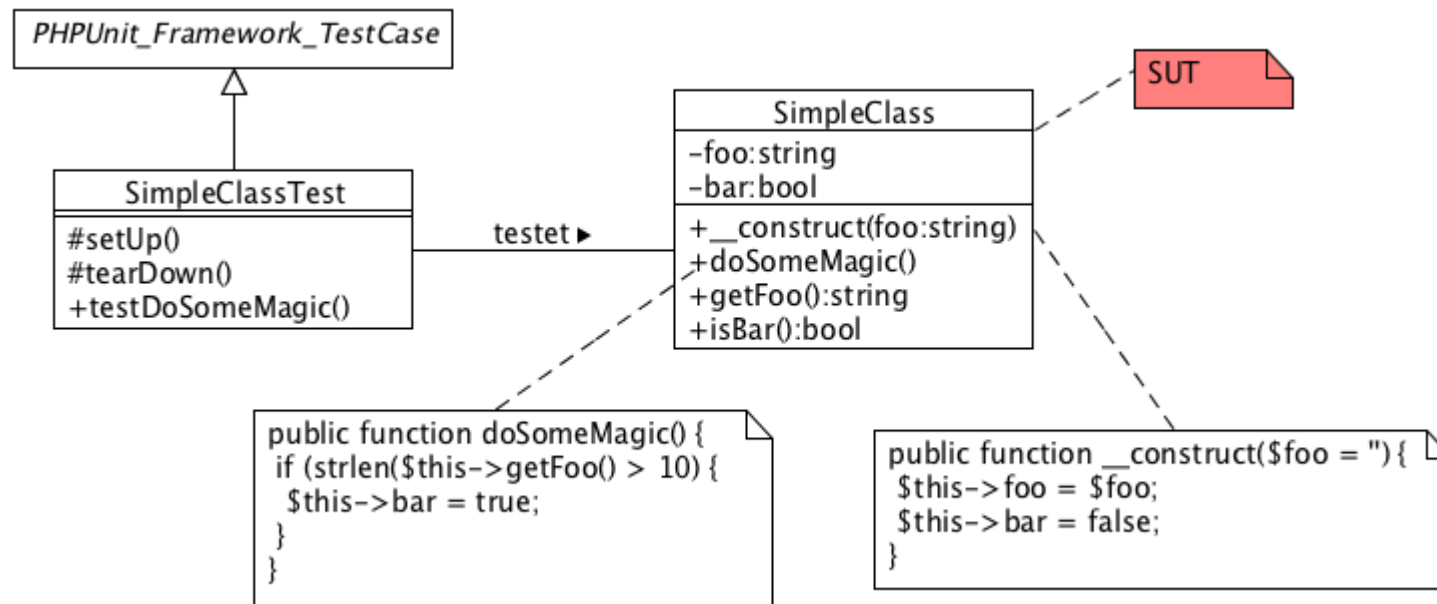
**Die Unittests :-)**

# WTF?

- erzeugt einen definierten Objekt-Graphen
- dieser Graph wird „Stimuliert“
- Verhalten wird mit Erwartung verglichen →  
Zusicherung/Assertion
- Und das immer und immer und immer wieder...



# Ein Test-Case



# Ein Test-Case

```
class SimpleClassTest extends PHPUnit_Framework_TestCase {  
  
    public function testDoSomeMagic() {  
        $fixtureString = 'short';  
        $mySimpleObject = new SimpleClass($fixtureString);  
        $this->assertEquals(fixtureString, $mySimpleObject->getFoo());  
        $this->assertFalse($mySimpleObject->isBar());  
  
        $fixtureString = 'longer than ten characters';  
        $mySimpleObject = new SimpleClass($fixtureString);  
        $this->assertEquals(fixtureString, $mySimpleObject->getFoo());  
        $this->assertTrue($mySimpleObject->isBar());  
    }  
}
```

# Ein Test-Case

```
class SimpleClassTest extends PHPUnit_Framework_TestCase {
```

einzelner Test-Case

```
public function testDoSomeMagic() {
```

Fixture

SUT

```
    $fixtureString = 'short';  
    $mySimpleObject = new SimpleClass($fixtureString);  
    $this->assertEquals($fixtureString, $mySimpleObject->getFoo());  
    $this->assertFalse($mySimpleObject->isBar());
```

```
    $fixtureString = 'longer than ten characters';  
    $mySimpleObject = new SimpleClass($fixtureString);  
    $this->assertEquals($fixtureString, $mySimpleObject->getFoo());  
    $this->assertTrue($mySimpleObject->isBar());
```

```
}
```

```
}
```

# Der Test-Case im speziellen

- ein Test-Case beginnt immer mit `test...`
- jeder Test-Case läuft isoliert\*
- jeder Test-Case hat die gleichen Vorbedingungen\*
- Reihenfolge sollte keine Rolle spielen\*
- vor jedem Test-Case wird `setUp()` ausgeführt
- nach jedem Test-Case wird `tearDown()` ausgeführt

\* Ausnahmen bestätigen die Regel: `@depends`

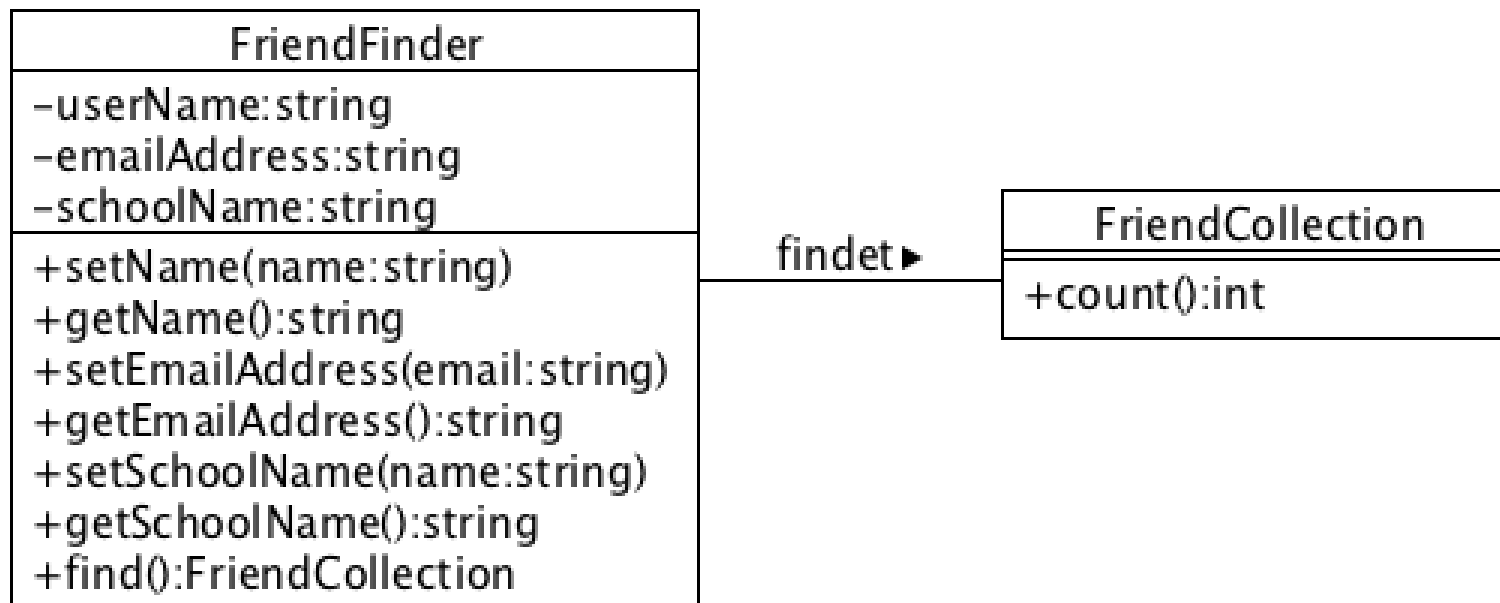
# setUp() und tearDown()

```
class TolleKlasseTest extends PHPUnit_Framework_TestCase {  
  
    private $sharedCache;  
  
    protected function setUp() {  
        $this->sharedCache = new CacheTool('/tmp/unittests');  
    }  
  
    protected function tearDown() {  
        $this->sharedCache->deleteCachedFiles();  
    }  
  
    public function testSomething() {  
        $mySut = new TolleKlasse($this->sharedCache); // DI :-)  
  
        // ...  
    }  
  
    public function testSomethingElse() {  
        $mySut = new TolleKlasse($this->sharedCache);  
  
        // ...  
    }  
  
    // ...  
}
```

# Was gehört in einen Unit-Test?

- Prüfung auf korrektes Verhalten (happy path)
- Prüfung auf Verhalten bei Randbedingungen!
  - Verhalten bei Fehlbenutzung
  - Verhalten bei falschen Parametern (z.B. Division durch 0)
  - Verhalten bei Ausnahmebedingungen/werfen von Exceptions (z.B. Service nicht erreichbar, Permissions etc.)
- öffentliche Schnittstelle
- einfache Getter und Setter nicht unbedingt
  - Werden oft implizit getestet

# Implizites Testen



# Implizites Testen

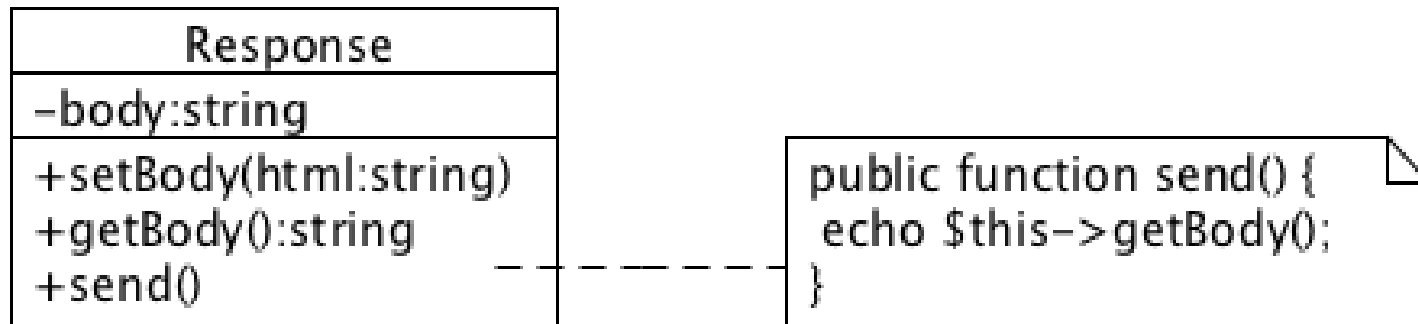
```
class FriendFinderTest extends PHPUnit_Framework_TestCase {  
  
    public function testSetName() {  
        // ...  
    }  
  
    public function testGetName() {  
        // ...  
    }  
  
    public function testSetEmailAddress() {  
        // ...  
    }  
  
    public function testGetEmailAddress() {  
        // ...  
    }  
  
    public function testSetSchoolName() {  
        // ...  
    }  
  
    // ...  
}
```



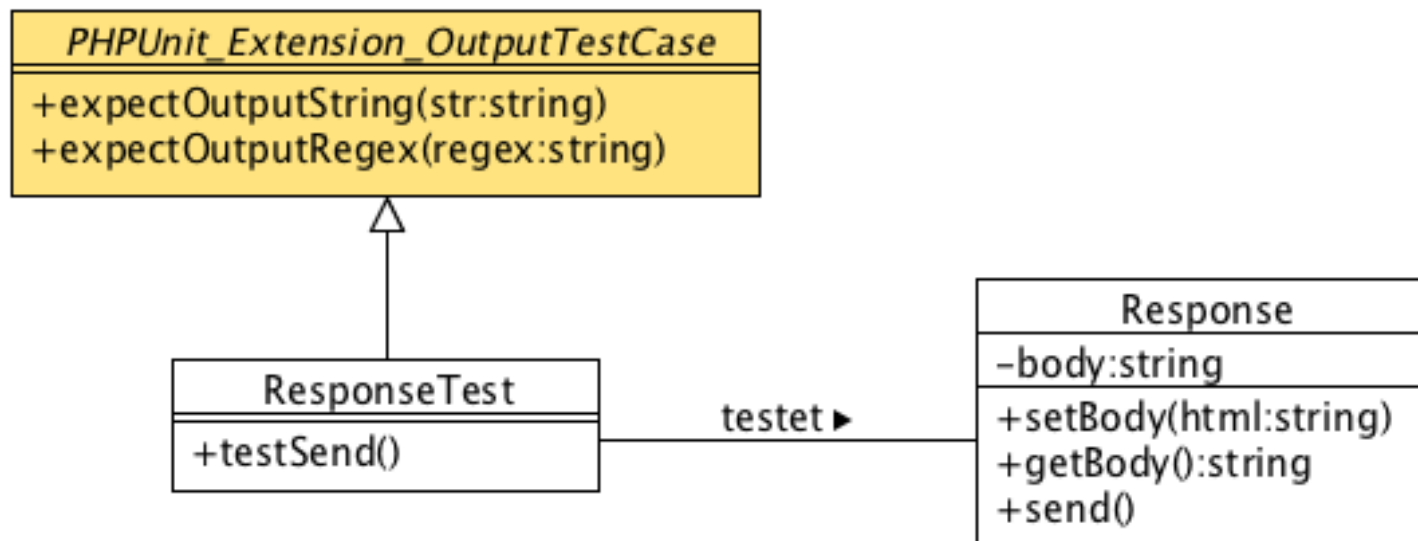
# Implizites Testen

```
class FriendFinderTest extends PHPUnit_Framework_TestCase {  
  
    public function testFindByName() {  
        $foundCount = 3; // Fixture  
        $name       = 'Hans Dampf'; // Fixture  
        $finder     = new FriendFinder(); //SUT  
        $finder->setName($name);  
  
        $this->assertEquals($name, $finder->getName());  
        $this->assertEquals($foundCount, $finder->find()->count());  
    }  
  
    // ...  
}
```

# Wie prüfe ich Ausgaben auf stdout?



# Wie prüfe ich Ausgaben auf stdout?



# Wie prüfe ich Ausgaben auf stdout?

```
class ResponseTest extends PHPUnit_Extension_OutputTestCase {  
  
    public function testSend() {  
        $fixture = '<html><body>foo</body></html>';  
        $response = new Response();  
        $response->setBody($fixture);  
  
        $this->assertEquals($fixture, $response->getBody());  
        $this->expectOutputString($fixture);  
        $response->send();  
    }  
  
    // ...  
}
```

# Wie prüfe ich Exceptions?

| XmlReader                     |       |
|-------------------------------|-------|
| -fileName:string              |       |
| +setFileName(fileName:string) |       |
| +load()                       | ----- |

```
public function load() {  
    $fileContent = file_getcontents($this->fileName);  
  
    if (false === $fileContent) {  
        $message = 'Can not read file: ' . $this->fileName;  
        throw new RuntimeException($message);  
    }  
    ...  
}
```

# Wie prüfe ich Exceptions?

```
class XmlReaderTest extends PHPUnit_Framework_TestCase {  
  
    public function testThrowExceptionOnFileDoesNotExists() {  
        $reader = new XmlReader();  
        $reader->setFileName('/file/does/not/exists');  
  
        $this->setExpectedException('RuntimeException');  
        $reader->load();  
    }  
  
    // ...  
}
```

# Wie prüfe ich Exceptions?

```
class XmlReaderTest extends PHPUnit_Framework_TestCase {

    public function testThrowExceptionOnFileDoesNotExists() {
        $fileName          = '/file/does/not/exists';
        $expectedMessage = 'Can not read file: ' . $fileName;

        $reader = new XmlReader();
        $reader->setFileName($fileName);

        try {
            $reader->load();
            $this->fail('On loading non existing file a
                        RuntimeException should be thrown!');
        } catch (RuntimeException $e) {
            $this->assertEquals($expectedMessage, $e->getMessage());
            // ...
        }

    }

    // ...
}
```

# Tests organisieren

```
lib/  
  FriendFinder.php  
  Response.php  
  XmlReader.php
```



# Tests organisieren

lib/  
  FriendFinder.php  
  Response.php  
  XmlReader.php



tests/  
  AllTests.php  
  FriendFinderTest.php  
  ResponseTest.php  
  XmlReaderTest.php

# Tests organisieren

lib/  
FriendFinder.php  
Response.php  
XmlReader.php

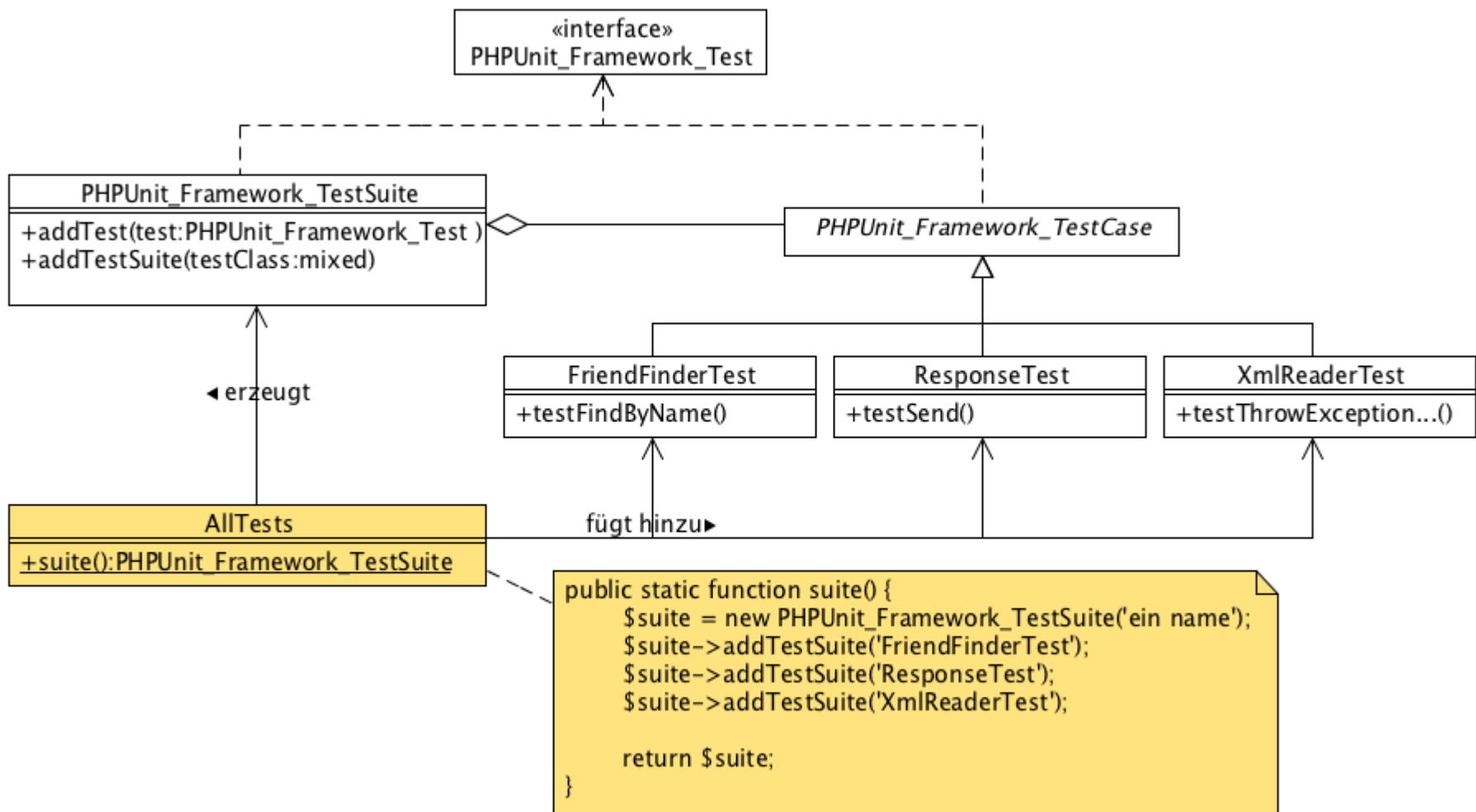


tests/  
AllTests.php  
FriendFinderTest.php  
ResponseTest.php  
XmlReaderTest.php



```
$> phpunit tests/AllTests.php
PHPUnit 3.4.8 by Sebastian Bergmann.
.I...II..S...IS.. 17 / 17
Time: 0 seconds, Memory: 7.5Mb
OK, but incomplete or skipped tests!
Tests: 17, Assertions: 56, Incomplete: 4, Skipped: 2.
```

# Tests organisieren



# Tests organisieren

```
lib/  
  Decorator.php  
  Decorator/  
    Iphone.php  
    Json.php  
  Request.php  
  Response.php  
  Session.php  
  Session/  
    Interface.php  
    SaveHandler.php  
  ...
```



```
tests/  
  AllTests.php  
  DecoratorTest.php  
  Decorator/  
    AllTests.php  
    IphoneTest.php  
    JsonTest.php  
  fixtures/  
    Decorator/  
      ...  
    Session/  
      ...  
    ...  
  RequestTest.php  
  ResponseTest.php  
  SessionTest.php  
  Session/  
    AllTests.php  
    SaveHandlerTest.php  
  ...  
  TestConfiguration.php  
  TestHelper.php
```

`/repos/kwick/tests`

# Mehr Code zum spicken

- <http://framework.zend.com/svn/framework/>
- <http://svn.ez.no/svn/ezcomponents/>
- <http://github.com/sebastianbergmann/phpunit/>
- <http://github.com/Weltraumschaf/VClasses/>

# Tipps & Tricks

- Failure vs. Error
  - Errors oft einfacher zu fixen
  - Bei vielen Failures und Errors erst alle Errors fixen
- erst einzelne Test-Cases laufen lassen
  - mit `$this->markTestIncomplete()` ausblenden
  - erst `MyClassTest.php` dann `AllTests.php`

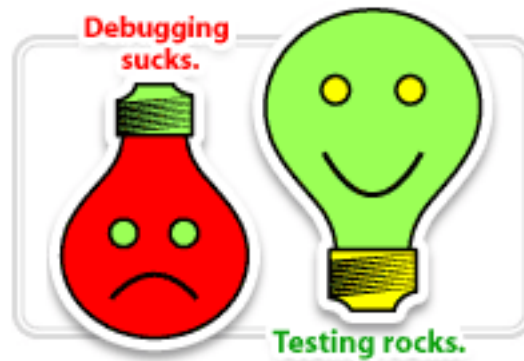
# RTFM

- [www.phpunit.de](http://www.phpunit.de)
- [googletesting.blogspot.com](http://googletesting.blogspot.com)
- Refactoring [Martin Fowler]



# Upcoming

- Self-Shunting
- Stubbing
- Mocking
- Database-Testing
- Selenium-Testing



**If it ain't broke, you're not trying hard enough!**