



The PHP Company

Zend Framework MVC Applikationen testen

Jan Burkl,
System Engineer, Zend Technologies

.Wer bin ich?

- Jan Burkl
 - jan.burkl@zend.com
- PHP Entwickler seit 2001
 - Projektarbeit
- Bei Zend seit 2006
 - System Engineer
- Zend Certified Engineer
 - PHP 5
 - Zend Framework



Was werden wir besprechen

- Unit Testing Basics
- Basics: funktionales Testen in ZF
- Verschiede “Advanced” ZF Testing Themen

Warum testen?

Vereinfacht Wartung

- Das Testen definiert Erwartungen
- Das Testen beschreibt das Verhalten durch die Applikation
- Das Testen teilt uns mit, wenn Neuerungen den Code an anderer Stelle auseinander brechen lassen

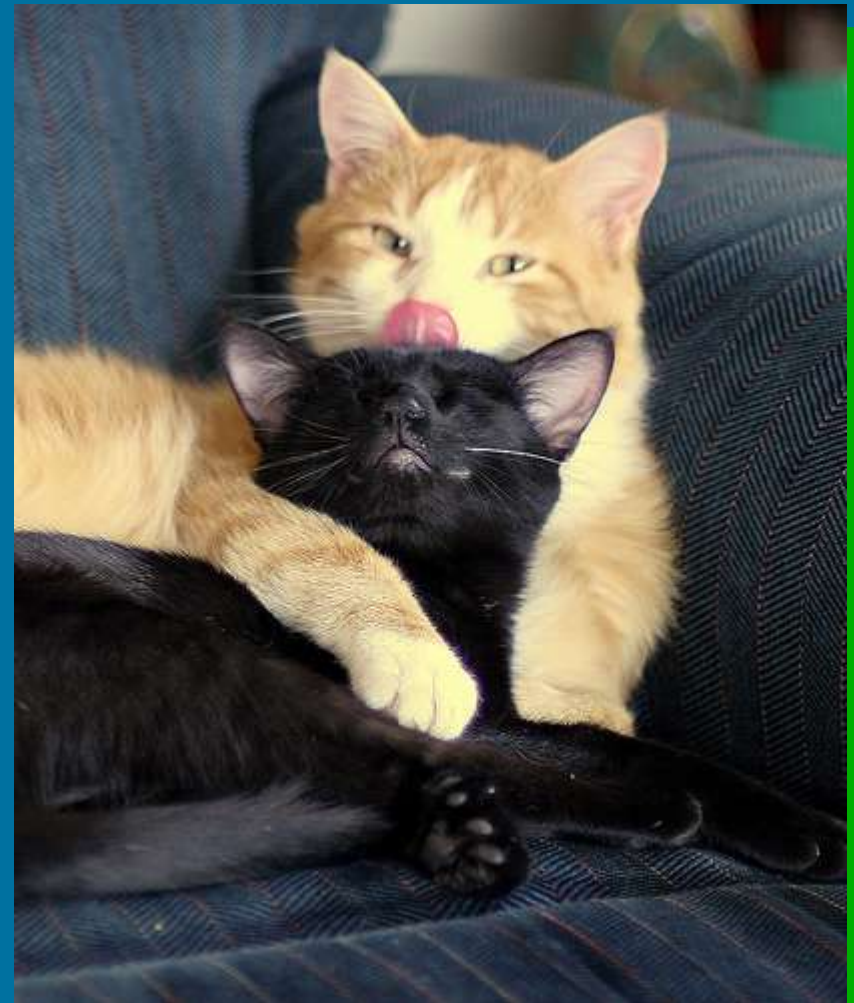
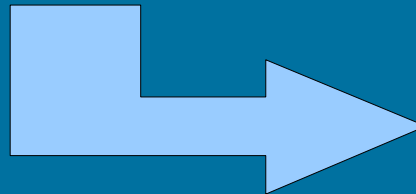
Code Qualität bestimmen

- Code Coverage durch Unit Tests
- Test Methoden dokumentieren das Verhalten, welches durch den Code definiert wird

Psychologischer Vorteil

*Warmes, angenehmes Gefühl,
wenn es grün leuchtet*

```
matthew@zlaptop ~framework/tests  
% phpunit --colors Zend/Dom  
PHPUnit 3.4.2 by Sebastian Bergmann.  
.....  
Time: 1 second  
OK (38 tests, 58 assertions)
```



Testen ist nicht... Reloading



Testen ist nicht... var_dump()

```
}
["_POST"] =>
array(1)
  ["date"] =>
string(0) ""
  ["event_date"] =>
string(0) ""
  ["wksp_type"] =>
string(0) ""
  ["wksp_vers"] =>
string(0) ""
  ["wksp_num"] =>
string(7) "F096341"
  ["search"] =>
string(15) "Search (Alt-S)"
  ["/cgi-bin/auth_secure.cgi_L"] =>
string(10) "1188159806"
  ["PHPSESSID"] =>
string(32) "0a6938bflc9a941b5270d2...75600"
}
```

Testen ist... reproduzierbar

```
Terminal
File Edit View Terminal Tabs Help
sb@ubuntu PHPUnit % ll Examples
total 20K
-rw-r--r-- 1 sb sb 1.4K 2008-03-30 17:03 BowlingGame.php
-rw-r--r-- 1 sb sb 1.4K 2008-03-30 17:03 BowlingGameTest.php
-rw-r--r-- 1 sb sb 225 2008-03-30 17:03 Schaltjahr.php
-rw-r--r-- 1 sb sb 386 2008-03-30 17:03 SchaltjahrTest.php
-rw-r--r-- 1 sb sb 1.5K 2008-04-09 09:57 WebTest.php
sb@ubuntu PHPUnit % phpunit Examples
PHPUnit @package_version@ by Sebastian Bergmann.

E.....

Time: 0 seconds

There was 1 error:

1) testTitle(WebTest)
RuntimeException: Could not connect to the Selenium RC server.

FAILURES
Tests: 7, Errors: 1.
sb@ubuntu PHPUnit %
```

Testen ist... automatisierbar

The screenshot shows the phpUnderControl web interface. At the top, the project is identified as 'phpuc-merge-test' and the build as 'More builds'. The interface includes a navigation menu with 'Overview', 'Tests', 'Metrics', 'Coverage', 'Documentation', 'CodeSniffer', and 'PHPUnit PMD'. The 'Tests' section is active, displaying a table of test results for 'Example::PhpUnderControl_Example_MathTest'.

Name	Status	Time(s)
Example::PhpUnderControl_Example_MathTest		0.101
testAddSuccess	Success	0.012
#1 - testAddSuccess - (build: php-5.3.0)	Success	0.006
#2 - testAddSuccess - (build: php-5.2.6)	Success	0.006
#3 - testAddSuccess - (build: php-5.2.5)	Success	0.001
testSubSuccess	Success	0.012
#1 - testSubSuccess - (build: php-5.3.0)	Success	0.006
#2 - testSubSuccess - (build: php-5.2.6)	Success	0.006
#3 - testSubSuccess - (build: php-5.2.5)	Success	0.000
testSubFail	Failure	0.015
#1 - testSubFail - (build: php-5.3.0)	Failure	
#2 - testSubFail - (build: php-5.2.6)	Failure	
#3 - testSubFail - (build: php-5.2.5)	Failure	
testDataProviderOneWIFail	Success	0.051
testDataProviderOneWIFail - (build: php-5.3.0)		0.024
#1 - testDataProviderOneWIFail with data set #0	Success	0.006
#2 - testDataProviderOneWIFail with data set #1	Success	0.006
#3 - testDataProviderOneWIFail with data set #2	Failure	
#4 - testDataProviderOneWIFail with data set #3	Success	0.006
testDataProviderOneWIFail - (build: php-5.2.6)		0.024
#1 - testDataProviderOneWIFail with data set #0	Success	0.006
#2 - testDataProviderOneWIFail with data set #1	Success	0.006
#3 - testDataProviderOneWIFail with data set #2	Failure	
#4 - testDataProviderOneWIFail with data set #3	Success	0.006
testDataProviderOneWIFail - (build: php-5.2.5)		0.002
#1 - testDataProviderOneWIFail with data set #0	Success	0.001
#2 - testDataProviderOneWIFail with data set #1	Success	0.000
#3 - testDataProviderOneWIFail with data set #2	Failure	
#4 - testDataProviderOneWIFail with data set #3	Success	0.001
testFail	Failure	0.011
#1 - testFail - (build: php-5.3.0)	Failure	
#2 - testFail - (build: php-5.2.6)	Failure	
#3 - testFail - (build: php-5.2.5)	Failure	

PHP Testing Frameworks

- PHPT
 - ▶ Genutzt von PHP, und einigen PEAR und unabhängigen Librarys
- SimpleTest
 - ▶ JUnit-Style Testing Framework
- PHPUnit
 - ▶ JUnit-Style Testing Framework
 - ▶ *De facto* Industrie-Standard

Testing Basics

Unit Tests schreiben

- Test Klasse erstellen
- Eine oder mehr Methoden erstellen, die das Verhalten beschreiben
 - Das Verhalten in natürlicher Sprache beschreiben
- Code schreiben, der das Verhalten erstellt
 - Code schreiben, der die API nutzt
- Behauptungen aufstellen, die auf die Erwartungen hindeuten

Test Klasse erstellen

- Normalerweise wird die Klasse nach dem Unit Under Test benannt

```
class EntryTest
    extends PHPUnit_Framework_TestCase
{
}
```

Methode zum Beschreiben des Verhaltens

- Präfix mit “test”

```
class EntryTest
    extends PHPUnit_Framework_TestCase
{
    public function testMaySetTimestampWithString()
    {
    }
}
```

Das Verhalten selbst coden

```
class EntryTest
    extends PHPUnit_Framework_TestCase
{
    public function testMaySetTimestampWithString()
    {
        $string = 'Fri, 7 May 2010 09:26:03 -0700';
        $ts     = strtotime($string);
        $this->entry->setTimestamp($string);
        $setValue = $this->entry->getTimestamp();
    }
}
```

Annahmen von Erwartungen

```
class EntryTest
    extends PHPUnit_Framework_TestCase
{
    public function testMaySetTimestampWithString()
    {
        $string = 'Fri, 7 May 2010 09:26:03 -0700';
        $ts     = strtotime($string);
        $this->entry->setTimestamp($string);
        $setValue = $this->entry->getTimestamp();
        $this->assertSame($ts, $setValue);
    }
}
```

- Data Provider

Run the Tests

- Fehler?
 - ▶ Überprüfen der Tests und Assertions auf potentielle Tippfehler oder falscher Benutzung
 - ▶ Den Unit Under Test nach Fehlern überprüfen
 - ▶ Korrekturen machen und Tests neu starten
- Erfolg?
 - ▶ Weiter zum nächsten Verhalten oder Feature!

Ein wenig Testing Terminologie

Test Scaffolding

- Sicherstellen, dass das Environment frei von Annahmen/Voraussetzungen ist
- Abhängigkeiten vor dem Testen initialisieren
- Normalerweise in der “setUp()” Methode

Test Doubles

- **Stubs**

Ein Objekt mit einem anderen ersetzen, so dass das System unterhalb des Tests weitermachen kann.

- ▶ *Behaviour & Status Verifizierung*

- **Mock Objects**

Ein Objekt mit einem anderen ersetzen und dafür Erwartungen definieren.

- ▶ *Behaviour Verifizierung*

- *<http://martinfowler.com/articles/mocksArentStubs.html>*

Zusätzliche Test Typen

- **Conditional Testing**
Nur testen, wenn bestimmte Umgebungsparameter zutreffen
- **Funktions- und Integrations-Tests**
Testen, dass das System als Ganzes sich wie gewünscht verhält; Testen, dass die Units wie erwartet interagieren

Quasi-Functional Testing im Zend Framework

Überblick

- Das PHPUnit Environment aufsetzen
- Einen Test-Case basierend auf einem ControllerTestCase erstellen
- Bootstrap der Applikation
- Erstellen und Dispatchen eines Requests
- Assertions auf der Response durchführen

Das PHPUnit Environment

- Verzeichnisstruktur

```
tests
|-- application
|   `-- controllers
|-- Bootstrap.php
|-- library
|   `-- Custom
`-- phpunit.xml

4 directories, 2 files
```

Das PHPUnit Environment

- phpunit.xml

```
<phpunit bootstrap="./Bootstrap.php">
  <testsuite name="Test Suite">
    <directory>./</directory>
  </testsuite>
  <filter>
    <whitelist>
      <directory
        suffix=".php">../library/</directory>
      <directory
        suffix=".php">../application/</directory>
      <exclude>
        <directory
          suffix=".phtml">../application/</directory>
        </exclude>
      </whitelist>
    </filter>
  </phpunit>
```

Das PHPUnit Environment

- Bootstrap.php

```
$rootPath = realpath(dirname(__FILE__) . '/../..');  
if (!defined('APPLICATION_PATH')) {  
    define('APPLICATION_PATH',  
          $rootPath . '/application');  
}  
if (!defined('APPLICATION_ENV')) {  
    define('APPLICATION_ENV', 'testing');  
}  
set_include_path(implode(PATH_SEPARATOR, array(  
    '.',  
    $rootPath . '/library',  
    get_include_path(),  
)));  
require_once 'Zend/Loader/Autoloader.php';  
$loader = Zend_Loader_Autoloader::getInstance();  
$loader->registerNamespace('Custom_');
```

Test Case Klasse erstellen

- Extend Zend_Test_PHPUnit_ControllerTestCase

```
class ExampleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
}
```

Bootstrap

- Zend_Application Instanz erstellen und in setUp() referenzieren

```
class ExampleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    public function setUp()
    {
        $this->bootstrap = new Zend_Application(
            APPLICATION_ENV,
            APPLICATION_PATH
            . '/configs/application.ini'
        );
        parent::setUp();
    }
}
```

Erstellen und Dispatchen eines Requests

- Einfache Methode: Dispatchen einer “URL”

```
class ExampleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    // ...
    public function testStaticPageHasGoodStructure()
    {
        $this->dispatch('/example/page');
        // ...
    }
}
```

Erstellen und Dispatchen eines Requests

- Fortgeschritten: Vor dem Dispatchen das Request Objekt anpassen

```
class ExampleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    // ...
    public function testXhrRequestReturnsJson()
    {
        $this->getRequest()
            ->setHeader('X-Requested-With',
                'XMLHttpRequest')
            ->setQuery('format', 'json');
        $this->dispatch('/example/xhr-endpoint');
        // ...
    }
}
```

Assertions erstellen

- Typische Assertions für:
 - ▶ Struktur des Response Markup
Entweder *CSS Selektoren* oder *XPath Assertions* benutzen.
 - ▶ HTTP Response Headers und Status Code
 - ▶ Request und/oder Response Objekt Teile

CSS Selektor Assertions

- `assertQuery($path, $message = "")`
- `assertQueryContentContains($path, $match, $message = "")`
- `assertQueryContentRegex($path, $pattern, $message = "")`
- `assertQueryCount($path, $count, $message = "")`
- `assertQueryCountMin($path, $count, $message = "")`
- `assertQueryCountMax($path, $count, $message = "")`
- **Jede Methode hat auch eine “Not” Variante**

XPath Selektor Assertions

- `assertXPath($path, $message = "")`
- `assertXPathContentContains($path, $match, $message = "")`
- `assertXPathContentRegex($path, $pattern, $message = "")`
- `assertXPathCount($path, $count, $message = "")`
- `assertXPathCountMin($path, $count, $message = "")`
- `assertXPathCountMax($path, $count, $message = "")`
- **Jede Methode hat auch eine “Not” Variante**

Redirect Assertions

- `assertRedirect($message = "")`
- `assertRedirectTo($url, $message = "")`
- `assertRedirectRegex($pattern, $message = "")`
- **Jede Methode hat auch eine “Not” Variante**

Response Assertions

- `assertResponseCode($code, $message = "")`
- `assertHeader($header, $message = "")`
- `assertHeaderContains($header, $match, $message = "")`
- `assertHeaderRegex($header, $pattern, $message = "")`
- **Jede Methode hat auch eine “Not” Variante**

Request Assertions

- `assertModule($module, $message = "")`
- `assertController($controller, $message = "")`
- `assertAction($action, $message = "")`
- `assertRoute($route, $message = "")`
- **Jede Methode hat auch eine “Not” Variante**

Assertion Beispiele

```
public function testSomeStaticPageHasGoodStructure ()
{
    $this->dispatch ('/example/page');
    $this->assertResponseCode (200);
    $this->assertQuery ('div#content p');
    $this->assertQueryCount ('div#sidebar ul li', 3);
}
```

Assertion Beispiele

```
public function testXhrRequestReturnsJson()  
{  
    // ...  
    $this->assertNotRedirect();  
    $this->assertHeaderContains(  
        'Content-Type', 'application/json');  
}
```

Advanced Topics

Order: Real World Use Cases

Testing Models und Resources

- **Das Problem:**
Diese Klassen können mit dem Standard-Autoloader nicht geladen werden
- **Die Lösung:**
Zend_Application benutzen, um während der setUp() Phase die “modules” Ressource zu bootstrappen

Testing Models und Resources

```
class Blog_Model_EntryTest
    extends PHPUnit_Framework_TestCase
{
    public function setUp()
    {
        $this->bootstrap = new Zend_Application(
            APPLICATION_ENV,
            APPLICATION_PATH
            . '/configs/application.ini'
        );
        $this->bootstrap->bootstrap('modules');

        $this->model = new Blog_Model_Entry();
    }
}
```

Testing Actions mit Authentifizierung

- **Das Problem:**
Einige Actions benötigen einen authentifizierten User - wie kann man das emulieren?
- **Die Lösung:**
Manuell gegen Zend_Auth authentifizieren bevor der Dispatcher aufgerufen wird

Test User authentifizieren

```
class ExampleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    // ...
    public function loginUser($user)
    {
        $params = array('user' => $user);
        $adapter = new Custom_Auth_TestAdapter(
            $params);
        $auth = Zend_Auth::getInstance();
        $auth->authenticate($adapter);
        $this->assertTrue($auth->hasIdentity());
    }
}
```

Authentifizierung und Dispatchen

```
class ExampleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    // ...
    public function testAdminUserCanAccessAdmin()
    {
        $this->loginUser('admin');
        $this->dispatch('/example/admin');
        $this->assertQuery('div#content.admin');
    }
}
```

Von Actions abhängige Seiten testen

- **Das Problem:**

Einige Actions hängen von anderen ab, z.B. das Erstellen einer Seite mit Content Highlighting, das auf einem Suchstring basiert

- **Die Lösung:**

Zweimal dispatchen, den Request und die Response zwischen den Aufrufen resetten

Von Actions abhängige Seiten testen

```
class ExampleControllerTest
    extends Zend_Test_PHPUnit_ControllerTestCase
{
    // ...
    public function testHighlightedTextAfterSearch()
    {
        $this->getRequest()->setQuery(
            'search', 'foobar');
        $this->dispatch('/search');

        $this->resetRequest();
        $this->resetResponse();

        $this->dispatch('/example/page');
        $this->assertQueryContains(
            'span.highlight', 'foobar');
    }
}
```

Schlussfolgerung

Immer Testen!

- Unit Tests der Models, Service Layers, etc.
- Funktions- und Akzeptanztests der Workflows, Seitenstruktur, etc.

Zend Framework Training & Certification

- **Zend Framework: Fundamentals**

Dieser Kurs vermittelt Wissen über ZF mit der Einführung des MVC-Entwurfsmusters (Model-View-Controller), um sicherzustellen, dass Sie die Best Practices bei der PHP-Entwicklung lernen.

- **Nächste Kurse:**

- ▶ 20. - 30. September (englisch)
- ▶ 2. - 12. November (deutsch)

Zend Framework Training & Certification

- **Zend Framework: Advanced**
Der Fortgeschrittenen-Kurs soll Zend Framework Entwicklern näherbringen, wie Best Practices beim Aufbau und der Konfiguration von Anwendungen für Skalierbarkeit, Interaktivität und hoher Performance angewendet werden.
- **Nächster Kurs**
 - ▶ 7. - 17. September (englisch)

Zend Framework Training & Certification

- **Test Prep: Zend Framework Certification**

Der ZF Zertifizierungskurs bereitet erfahrene ZF Entwickler auf die Herausforderung der Zertifizierungsprüfung vor. Bei bestandener Prüfung erhält der Entwickler den Status des Zend Certified Engineer in Zend Framework (ZCE-ZF).

- **Nächste Kurse**

- ▶ 6. - 24. September (deutsch)
- ▶ 13. September - 1. Oktober (englisch)

- **Freier Study Guide**

<http://www.zend.com/en/download/173>

ZendCon 2010!

- 1. bis 4. November 2010
- <http://www.zendcon.com/>

Danke schön!
jan@zend.com