Spickzettel zu testgetriebener Entwicklung mit TYPO3 CMS

Oliver Klee, `typo3-coding@oliverklee.de`, `@oliklee`

Version 1.4.1, 11. November 2013

Inhaltsverzeichnis

1 Benennung von Dateien und Klassen

1.1 Dateinamen

Dateiname des Produktionscodes	Name der Testdatei
Classes/Domain/Model/Shoe.php	Tests/Unit/Domain/Model/ShoeTest.php
Classes/Service/BaristaService.php	Tests/Unit/Service/BaristaServiceTest.php
pi1/class.tx_frubble_pi1.php	Tests/Unit/pi1/pi1Test.php

1.2 Klassennamen

Name der Klasse im Produktionscode	Name der Testklasse
Tx_Life_Domain_Model_Shoe	Tx_Life_Domain_Model_ShoeTest
Tx_Life_Service_BaristaService	Tx_Life_Service_BaristaServiceTeset
tx_frubble_pi1	tx_frubble_pi1Test

2 Struktur von Testklassen

2.1 Extbase-Extensions

```
class Tx_Articlebase_Domain_Model_ArticleTest extends \Tx_Extbase_Tests_Unit_BaseTestCase {
  /**
   * @var \Tx_Articlebase_Domain_Model_Article
   */
  protected $subject = NULL;

  public function setUp() {
    $this->subject = new Tx_Articlebase_Domain_Model_Article();
    $this->subject->initializeObject();
  }

  public function tearDown() {
    unset($this->subject);
  }
  /**
   * @test
   */
  public function getNameInitiallyReturnsEmptyString() {
    $this->assertSame(
      '',
      $this->subject->getName()
    );
  }

  /**
   * @test
   */
  public function setNameSetsName() {
    $this->subject->setName('foo bar');

    $this->assertSame(
      'foo bar',
      $this->subject->getName()
    );
  }
...
}
```

2.2 Nicht-Extbase-Extensions

```
// Hier die zu testende Klasse einbinden, wenn eure Extension keine
// ext_autoload.php benutzt.
require_once(t3lib_extMgm::extPath('oelib') . 'class.tx_oelib_Attachment.php');

class tx_oelib_AttachmentTest extends \Tx_Phpunit_TestCase {
  /**
   * @var \tx_oelib_Attachment
   */
```

```
protected $subject = NULL;

public function setUp() {
  $this->subject = new tx_oelib_Attachment();
}

public function tearDown() {
  unset($this->subject);
}

/**
 * @test
 */
public function getFileNameInitiallyReturnsAnEmptyString() {
  $this->assertSame(
    '',
    $this->subject->getFileName()
  );
}

/**
 * @test
 */
public function getFileNameWithFileNameSetReturnsFileName() {
  $this->subject->setFileName('test.txt');

  $this->assertSame(
    'test.txt',
    $this->subject->getFileName()
  );
}

/**
 * @test
 */
```

```
  public function setFileNameWithEmptyFileNameThrowsException() {
    $this->setExpectedException('InvalidArgumentException', '$fileName must not be empty.');

    $this->subject->setFileName('');
  }
...
}
```

2.3 Nicht-TYPO3-PHP-Project

```
namespace Books\Domain\Model;

$currentDirectory = dirname(__FILE__);
require_once($currentDirectory . '../../../../../Classes/Domain/Model/Book.php');

class BookTest extends \PHPUnit_Framework_TestCase {
  /**
   * @var Book
   */
  protected $subject = NULL;

  public function setUp() {
    $this->subject = new Book();
  }

  public function tearDown() {
    unset($this->subject);
  }

  /**
   * @test
   */
  public function getTitleInitiallyReturnsEmptyString() {
    $this->assertSame(
      '',
      $this->subject->getTitle()
    );
  }

  /**
   * @test
   */
  public function setTitleSetsTitle() {
    $this->subject->setTitle('foo bar');

    $this->assertSame(
      'foo bar',
      $this->subject->getTitle()
    );
  }
}
```

3 Auf Exceptions testen

3.1 Nur auf die Klasse testen (empfohlen)

```
/**
 * @test
 * @expectedException InvalidArgumentException
 */
public function createBreadWithNegativeSizeThrowsException() {
  $this->subject->createBread(-1);
}
```

3.2 Auf Klasse, Nachricht und (optional) Code testen (empfohlen)

```
/**
 * @test
 */
```

```
public function createBreadWithNegativeSizeThrowsException() {
  $this->setExpectedException(
    'InvalidArgumentException',
    '$size must be > 0.',
     1323700434
  );

  $this->subject->createBread(-1);
}

/**
 * @test
 */
public function createBreadWithZeroSizeThrowsException() {
  $this->setExpectedException(
    'InvalidArgumentException',
    '$size must be > 0.'
  );

  $this->subject->createBread(-1);
}
```

3.3 Try/catch (nicht empfohlen)

```
/**
 * @test
 */
public function createBreadWithNegativeSizeThrowsException() {
  try {
    $this->subject->createBread(-1);
    $this->fail('The expected exception has not been thrown.');
  } catch (InvalidArgumentException $exception) {
  }
}
```

4 Abstrakte Klassen testen

4.1 Den PHPUnit-Mock-Builder benutzen (empfohlen)

Dies erzeugt eine Instanz der abtrakten Klassen, wobei alle abstrakten Methoden gemockt werden.

```
class Tx_Coffee_Domain_Model_AbstractBeverageTest {
 /**
  * @var \Tx_Coffee_Domain_Model_AbstractBeverage|\PHPUnit_Framework_MockObject_MockObject
  *
 protected $subject = NULL;

 protected function setUp() {
   $this->subject = $this->getMockForAbstractClass('Tx_Coffee_Domain_Model_AbstractBeverage');
 }
```

4.2 Eine konkrete Unterklasse erstellen (empfohlen)

Dies wird empfohlen, wenn die Subklasse zusätzliches oder spezifisches Verhalten haben soll.

Erzeugt in `Tests/Unit/Fixtures/` eine Unterklasse der abstrakten Klasse:

```
class Tx_Coffee_Domain_Model_TestingBeverage extends \Tx_Coffee_Domain_Model_AbstractBeverage {
  ...
}
```

Dann könnt ihr die konkrete Unterklasse in euren Unit-Tests includen und instanziieren.

```
require_once(t3lib_extMgm::extPath('coffee') . 'Tests/Unit/Fixtures/TestingBeverage.php');

class Tx_Coffee_Domain_Model_AbstractBeverageTest {
 /**
  * @var \Tx_Coffee_Domain_Model_TestingBeverage
  *
 protected $subject = NULL;
 protected function setUp() {
   $this->subject = new Tx_Coffee_Domain_Model_TestingBeverage();
 }
```

4.3 Using eval (nicht empfohlen)

Dies ist nicht empfehlenswert, weil ihr damit die Code-Vervollständigung in eurer IDE kaputtmacht.

5 Das Test-Framework der phpunit-TYPO3-Extension benutzen

```
class tx_oelib_DataMapperTest extends \Tx_Phpunit_TestCase {
  /**
   * @var \Tx_Phpunit_Framework
   */
  protected $testingFramework = NULL;
  /**
   * @var \tx_oelib_DataMapper
   */
  protected $subject = NULL;

  public function setUp() {
    $this->testingFramework = new Tx_Phpunit_Framework('tx_oelib');

    $this->subject = ...
  }

  public function tearDown() {
    $this->testingFramework->cleanUp();

    unset($this->subject, $this->testingFramework);
  }

  /**
   * @test
   */
  public function findWithUidOfExistingRecordReturnsModelDataFromDatabase() {
    $uid = $this->testingFramework->createRecord(
      'tx_oelib_test', array('title' => 'foo')
    );

    $this->assertSame(
      'foo',
      $this->subject->find($uid)->getTitle()
    );
  }
```

6 Gemockte Dateisystem mit vfsStream benutzen

6.1 Einrichten

```
use \org\bovigo\vfs\vfsStream;

/**
 * @var \org\bovigo\vfs\vfsStreamFile
 */
protected $moreStuff;

public function setUp() {
  // This is the same as ::register and ::setRoot.
  $root = vfsStream::setUp('Stuff');
  $this->moreStuff = vfsStream::newDirectory('moreStuff')->at($root);

  $this->subject = new ...
}
```

6.2 Die Dateien benutzen

```
/**
 * @test
 */
public function checkFileWithPathOfExistingNonEmptyFileReturnsTrue() {
  $file = vfsStream::newFile('test.php')->at($this->moreStuff);
  $file->withContent('Hello world!');

  $this->assertTrue(
    $this->subject->checkFile(vfsStream::url('Stuff/moreStuff/test.php'))
  );
}
```

7 PHPUnit-Assertions

Diese Liste ist aktuell für PHPUnit 3.7.x.

```
assertArray[Not]HasKey(mixed $key, array $array[, string $message = ''])
assertClass[Not]HasAttribute(string $attributeName, string $className[, string $message = ''])
assertClass[Not]HasStaticAttribute(string $attributeName, string $className[, string $message = ''])
assert[Not]Contains(mixed $needle, Iterator|array $haystack[, string $message = ''])
assert[Not]ContainsOnly(string $type, Iterator|array $haystack[, boolean $isNativeType = NULL, string $message = ''])
assertContainsOnlyInstancesOf(string $classname, Traversable|array $haystack[, string $message = ''])
assert[Not]Count($expectedCount, $haystack[, string $message = ''])
assert[Not]Empty(mixed $actual[, string $message = ''])
assertEqualXMLStructure(DOMElement $expectedElement, DOMElement $actualElement[, boolean $checkAttributes = FALSE, string $message = ''])
assert[Not]Equals(mixed $expected, mixed $actual[, string $message = ''])
assertFalse(bool $condition[, string $message = ''])
assertFile[Not]Equals(string $expected, string $actual[, string $message = ''])
assertFile[Not]Exists(string $filename[, string $message = ''])
assertGreaterThan(mixed $expected, mixed $actual[, string $message = ''])
assertGreaterThanOrEqual(mixed $expected, mixed $actual[, string $message = ''])
assert[Not]InstanceOf($expected, $actual[, $message = ''])
```

```
assert[Not]InternalType($expected, $actual[, $message = ''])
assertJsonFileEqualsJsonFile(mixed $expectedFile, mixed $actualFile[, string $message = ''])
assertJsonStringEqualsJsonFile(mixed $expectedFile, mixed $actualJson[, string $message = ''])
assertJsonStringEqualsJsonString(mixed $expectedJson, mixed $actualJson[, string $message = ''])
assertLessThan(mixed $expected, mixed $actual[, string $message = ''])
assertLessThanOrEqual(mixed $expected, mixed $actual[, string $message = ''])
assert[Not]Null(mixed $variable[, string $message = ''])
assertObject[Not]HasAttribute(string $attributeName, object $object[, string $message = ''])
assert[Not]RegExp(string $pattern, string $string[, string $message = ''])
assertString[Not]MatchesFormat(string $format, string $string[, string $message = ''])
assertString[Not]MatchesFormatFile(string $formatFile, string $string[, string $message = ''])
assert[Not]Same(mixed $expected, mixed $actual[, string $message = ''])
assertSelectCount(array $selector, integer $count, mixed $actual[, string $message = '', boolean $isHtml = TRUE])
assertSelectEquals(array $selector, string $content, integer $count, mixed $actual[, string $message = '', boolean $isHtml = TRUE])
assertSelectRegExp(array $selector, string $pattern, integer $count, mixed $actual[, string $message = '', boolean $isHtml = TRUE])
assertStringEnds[Not]With(string $suffix, string $string[, string $message = ''])
assertString[Not]EqualsFile(string $expectedFile, string $actualString[, string $message = ''])
assertStringStarts[Not]With(string $prefix, string $string[, string $message = ''])
assertTag(array $matcher, string $actual[, string $message = '', boolean $isHtml = TRUE])
assertThat(mixed $value, PHPUnit_Framework_Constraint $constraint[, $message = ''])
assertTrue(bool $condition[, string $message = ''])
assertXmlFile[Not]EqualsXmlFile(string $expectedFile, string $actualFile[, string $message = ''])
assertXmlString[Not]EqualsXmlFile(string $expectedFile, string $actualXml[, string $message = ''])
assertXmlString[Not]EqualsXmlString(string $expectedXml, string $actualXml[, string $message = ''])
```