

Spickzettel zu testgetriebener Entwicklung mit PHPUnit

Oliver Klee | typo3-coding@oliverklee.de | @oliklee
<https://github.com/oliverklee/tdd-reader>

Version 2.0.2, 6. Mai 2017, für PHP ≥ 5.6 und TYPO3 CMS 7.6

Lizenz

Dieser Reader ist unter einer *Creative-Commons*-Lizenz lizenziert, und zwar konkret unter der *Namensnennung-Weitergabe unter gleichen Bedingungen 4.0 (CC BY-SA 4.0)*. Das bedeutet, dass ihr den Reader unter diesen Bedingungen für euch kostenlos verbreiten, bearbeiten und nutzen könnt (auch kommerziell):

Namensnennung. Ihr müsst den Namen des Autors (Oliver Klee) nennen. Wenn ihr dabei zusätzlich auch noch die Quelle¹ nennt, wäre das nett. Und wenn ihr mir zusätzlich eine Freude machen möchtet, sagt mir per E-Mail Bescheid.

Weitergabe unter gleichen Bedingungen. Wenn ihr diesen Inhalt bearbeitet oder in anderer Weise umgestaltet, verändert oder als Grundlage für einen anderen Inhalt verwendet, dann dürft ihr den neu entstandenen Inhalt nur unter Verwendung identischer Lizenzbedingungen weitergeben.

Lizenz nennen. Wenn ihr den Reader weiter verbreitet, müsst ihr dabei auch die Lizenzbedingungen nennen oder beifügen.

Die ausführliche Version dieser Lizenz findet ihr online.²

¹<https://github.com/oliverklee/tdd-reader>

²<http://creativecommons.org/licenses/by-sa/4.0/>

Inhaltsverzeichnis

1	Ein Wort zu TYPO3 CMS 8.7	3
2	Benennung von Dateien und Klassen	3
2.1	Dateinamen	3
2.2	Klassennamen	3
3	Struktur von Testklassen	4
3.1	Extbase-Extensions	4
3.2	Nicht-Extbase-Extensions	4
3.3	Nicht-TYPO3-PHP-Project mit Composer	5
3.3.1	composer.json	5
3.3.2	Testcase	6
4	Tests ausführen	8
4.1	Nicht-TYPO3-Projekte	8
4.1.1	Auf der Kommandozeile	8
4.1.2	In PhpStorm	8
4.2	TYPO3-Extensions	8
4.2.1	In PhpStorm	8
4.2.2	Mit dem Backend-Modul der PHPUnit-Extension	11
5	Mocks	12
5.1	Warum mocken?	12
5.2	Tools für Mocks	12
6	Auf Exceptions testen	13
6.1	Nur auf die Klasse testen	13
6.2	Auf Klasse, Nachricht und Code testen	13
7	Abstrakte Klassen testen	14
7.1	Den PHPUnit-Mock-Builder benutzen	14
7.2	Eine konkrete Unterklasse erstellen	14
8	Das Test-Framework der PHPUnit-TYPO3-Extension benutzen	15
9	Gemockte Dateisystem mit vfsStream benutzen	16
9.1	Lauffähige Beispiele	16
9.2	Einrichten	16
9.3	Die Dateien benutzen	16
10	PHPUnit-Assertions	18

1 Ein Wort zu TYPO3 CMS 8.7

Die TYPO3-spezifischen Teile dieses Handouts passen zu TYPO3 CMS 7.6. Für die Version 8.7 gibt es ein paar Unterschiede:

- Benutzt das TYPO3-Testing-Framework³ anstelle des Testing-Frameworks des PHPUnit-Extension.
- Die Test-Basisklasse kommt jetzt aus dem TYPO3-Testing-Framework und nicht mehr aus dem TYPO3-Core.
- Da TYPO3 CMS 8.7 kein PHP 5.6 mehr unterstützt, könnt ihr PHPUnit 6 benutzen (das PHP 7 erfordert).

2 Benennung von Dateien und Klassen

2.1 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

2.2 Klassennamen

Name der Klasse im Produktionscode	Name der Testklasse
Shoes\Shop\Domain\Model\Shoe	Shoes\Shop\Tests\Unit\Domain\Model\ShoeTest
Shoes\Shop\Service\BaristaService	Shoes\Shop\Tests\Unit\Service\BaristaServiceTest

³<https://github.com/TYPO3/testing-framework>

3 Struktur von Testklassen

3.1 Extbase-Extensions

Es gibt auf GitHub dazu auch ein Beispielprojekt (das *Tea-Example*):
https://github.com/oliverklee/ext_tea

```
1 namespace \OliverKlee\Shop\Tests\Unit\Domain\Model;
2
3 use OliverKlee\Shop\Domain\Model\Article;
4
5 class ArticleTest extends \TYPO3\CMS\Core\Tests\UnitTestCase {
6     /**
7      * @var Article;
8      */
9     protected $subject = null;
10
11     protected function setUp()
12     {
13         $this->subject = new Article;
14         $this->subject->initializeObject();
15     }
16
17     /**
18      * @test
19      */
20     public function getNameInitiallyReturnsEmptyString()
21     {
22         self::assertSame('', $this->subject->getName());
23     }
24
25     /**
26      * @test
27      */
28     public function setNameSetsName()
29     {
30         $name = 'foo bar';
31
32         $this->subject->setName($name);
33
34         self::assertSame($name, $this->subject->getName());
35     }
36
37     // ...
38 }
```

3.2 Nicht-Extbase-Extensions

```
1 class AttachmentTest extends \Tx_Phpunit_TestCase {
2     /**
3      * @var \Tx_Delibe_Attachment
4      */
5 }
```

```

5     protected $subject = null;
6
7     protected function setUp()
8     {
9         $this->subject = new \Tx_Oelib_Attachment();
10    }
11
12    /**
13     * @test
14     */
15    public function getFileNameInitiallyReturnsAnEmptyString()
16    {
17        self::assertSame('', $this->subject->getFileName());
18    }
19
20    /**
21     * @test
22     */
23    public function getFileNameWithFileNameSetReturnsFileName()
24    {
25        $fileName = 'test.txt';
26
27        $this->subject->setFileName($fileName);
28
29        self::assertSame($fileName, $this->subject->getFileName());
30    }
31
32    /**
33     * @test
34     * @expectedException \UnexpectedValueException
35     */
36    public function setFileNameWithEmptyFileNameThrowsException()
37    {
38        $this->subject->setFileName('');
39    }
40
41    // ...
42 }

```

3.3 Nicht-TYPO3-PHP-Project mit Composer

Es gibt auf GitHub dazu auch ein leeres Startprojekt:

<https://github.com/oliverklee/tdd-seed>

3.3.1 composer.json

Diese composer.json installiert PHPUnit und vfsStream für **PHP bis 5.6**:

```

1 {
2     "require-dev": {
3         "phpunit/phpunit": "^5.7.19",
4         "mikey179/vfsStream": "^1.6.4"

```

```

5     },
6     "autoload": {
7         "psr-4": {
8             "...":
9         }
10    },
11    "autoload-dev": {
12        "psr-4": {
13            "...":
14        }
15    }
16 }

```

Diese composer.json installiert PHPUnit und vfsStream für **PHP ab 6:**

```

1 {
2     "require-dev": {
3         "phpunit/phpunit": "^6.0.13",
4         "mikey179/vfsStream": "^1.6.4"
5     },
6     "autoload": {
7         "psr-4": {
8             "...":
9         }
10    },
11    "autoload-dev": {
12        "psr-4": {
13            "...":
14        }
15    }
16 }

```

3.3.2 Testcase

```

1 namespace OliverKlee\Books\Tests\Unit\Domain\Model;
2
3 use OliverKlee\Books\Domain\Model;
4
5 class BookTest extends \PHPUnit_Framework_TestCase {
6     /**
7      * @var Book
8      */
9     protected $subject = null;
10
11     protected function setUp()
12     {
13         $this->subject = new Book();
14     }
15
16     /**
17      * @test

```

```
18      */
19      public function getTitleInitiallyReturnsEmptyString()
20      {
21          self::assertSame('', $this->subject->getTitle());
22      }
23
24      /**
25       * @test
26       */
27      public function setTitleSetsTitle()
28      {
29          $title = 'foo bar';
30
31          $this->subject->setTitle($title);
32
33          self::assertSame($title, $this->subject->getTitle());
34      }
35  }
```

4 Tests ausführen

4.1 Nicht-TYPO3-Projekte

4.1.1 Auf der Kommandozeile

```
1 vendor/bin/phpunit Test/
```

4.1.2 In PhpStorm

1. Settings > Languages & Frameworks > PHP > PHPUnit
2. PHPUnit library > Use Composer autoloader
3. PHPUnit library > Path to script: `vendor/autoload`
4. OK
5. auf den Order **Tests/** rechtsklicken (oder einen anderen Ordner oder eine Testdatei)
6. Run 'Tests'

4.2 TYPO3-Extensions

4.2.1 In PhpStorm

Für eine existierende TYPO3-Installation im Composer-Modus: Bei diesem Ansatz werden alle installierten Extensions geladen, sodass ihr auch die Features der PHPUnit-Extension nutzen könnt.

1. Settings > Languages & Frameworks > PHP > PHPUnit
2. PHPUnit library > Use Composer autoloader
3. PHPUnit library > Path to script: `vendor/autoload` aus dem Document-Root der TYPO3-Installation
4. Test runner > Default configuration file: `typo3/sysex/core/Build/UnitTests.xml` innerhalb des Document-Roots der TYPO3-Installation
5. OK
6. Run > Edit Configurations
7. Defaults > PHPUnit
8. Command Line > Environment variables
9. zwei Variablen hinzufügen:
 - `TYPO3_CONTENT` = `Development`
 - `TYPO3_PATH_WEB` = der absolute Pfad zum Document-Root der TYPO3-Installation (ohne den Slash am Ende)
10. auf den Order **Tests/** rechtsklicken (oder einen anderen Ordner oder eine Testdatei)
11. Run 'Tests'

Für eine existierende TYPO3-Installation im Klassik-Modus (Nicht-Composer-Modus): In diesem Fall werdet ihr keine Klassen aus anderen Extensions autoloade können, d. h., ihr werdet auch keine Features der PHPUnit-Extension nutzen können (und auch nicht von anderen Extension-Abhängigkeiten).

1. Wenn ihr den TYPO3-Source per git statt als TAR-Paket heruntergeladen habt, braucht ihr ein `composer install` im TYPO3-Source-Verzeichnis.
2. Settings > Languages & Frameworks > PHP > PHPUnit
3. PHPUnit library > Use Composer autoloader
4. PHPUnit library > Path to script: `vendor/autoload` innerhalb des TYPO3-Source
5. Test runner > Default configuration file: `typo3/sysext/core/Build/UnitTests.xml` innerhalb des TYPO3-Document-Roots
6. OK
7. Run > Edit Configurations
8. Defaults > PHPUnit
9. Command Line > Environment variables
10. zwei Variablen hinzufügen:
 - `TYP03_CONTENT` = `Development`
 - `TYP03_PATH_WEB` = der absolute Pfad zum Document-Root der TYPO3-Installation (ohne den Slash am Ende)
11. auf den Order `Tests/` rechtsklicken (oder einen anderen Ordner oder eine Testdatei)
12. Run 'Tests'

Ohne eine existierende TYPO3-Installation Dieser Ansatz benutzt das TYPO3-Extension-Skelett⁴ von Helmut Hummel und Nicole Cordes.

Fügt die folgenden Abschnitte zur `composer.json` eurer Extension hinzu:

```
1  "require": {
2      "typo3/cms": "~7.6.0"
3  },
4  "require-dev": {
5      "namelesscoder/typo3-repository-client": "^1.2",
6      "nimut/testing-framework": "^1.0",
7      "mikey179/vfsStream": "^1.4",
8      "phpunit/phpunit": "^4.7 || ^5.0"
9  },
10 "config": {
11     "vendor-dir": ".Build/vendor",
12     "bin-dir": ".Build/bin"
13 },
14 "scripts": {
15     "post-autoload-dump": [
16         "mkdir -p .Build/Web/typo3conf/ext/",
17         "[ -L .Build/Web/typo3conf/ext/tea ] || ln -snvf ../../../../.Build/Web/typo3conf/ext/tea"
18     ]
19 }
```

⁴https://github.com/helhum/ext_scaffold

```

19 },
20 "extra": {
21     "typo3/cms": {
22         "cms-package-dir": "${vendor-dir}/typo3/cms",
23         "web-dir": ".Build/Web"
24     }
25 }

```

Ersetzt dabei `tea` in Zeile 18 durch den Schlüssel eurer Extension.

Wenn ihr außerdem noch andere, im TER verfügbarer Extensions in den Tests nutzen möchtet (z. B. die PHPUnit-Extension), fügt noch diese Abschnitte zu eurer `composer.json` hinzu (bzw. mergt sie):

```

1 "repositories": [
2 {
3     "type": "composer",
4     "url": "https://composer.typo3.org/"
5 }
6 ],
7 "require-dev": {
8     "typo3-ter/phpunit": "*"
9 }

```

Führt danach diese Schritte in PhpStorm aus:

1. Settings > Languages & Frameworks > PHP > PHPUnit
2. PHPUnit library > Use Composer autoloader
3. PHPUnit library > Path to script:
 - auf den Button *Show hidden files and directories* klicken `.Build/vendor/autoload.php` im Verzeichnis der Extension
4. Test runner > Default configuration file: `typo3/sysex/core/Build/UnitTests.xml` im TYPO3-Source-Verzeichnis
5. OK
6. Run > Edit Configurations
7. Defaults > PHPUnit
8. Command Line > Environment variables
9. zwei Variablen hinzufügen:
 - `TYPO3_CONTENT = Development`
 - `TYPO3_PATH_WEB` = der absolute Pfad zu `.Build/Web` im Verzeichnis der Extension (ohne den Slash am Ende)
10. auf den Order `Tests/` rechtsklicken (oder einen anderen Ordner oder eine Testdatei)
11. Run 'Tests'

4.2.2 Mit dem Backend-Modul der PHPUnit-Extension

Dieser Ansatz funktioniert sowohl für TYPO3-Installationen im Composer-Modus als auch im Klassik-Modus.

Dabei werden alle installierten Extensions geladen (inklusive der PHPUnit-Extension), sodass ihr die Features der PHPUnit-Extension nutzen könnt.

Allerdings werden die Tests dann im Kontext der aktuellen Backend-Session ausgeführt, was die Tests sehr zehrbrechlich macht. Für die meisten Unit-Tests von Extensions funktioniert das, für funktionale Tests wird es hingegen meistens nicht gut funktionieren.

1. Admin > PHPUnit

5 Mocks

5.1 Warum mocken?

- um eine Methode „auszuschalten“ (damit sie nicht in die DB schreibt, kein Cruise-Missile abschießt etc.) und `null` zurückzugeben
- um einer Methode einen bestimmten Rückgabewert zu geben oder sie eine Exception werfen zu lassen
- um zu testen, dass eine Methode auf eine bestimmte Art und Weise aufgerufen wird

5.2 Tools für Mocks

Prophecy: Das empfohlene, einfach benutzbare, aktuelle Mocking-Framework. Allerdings kann es keine partiellen Mocks erzeugen.⁵

PHPUnit mocks: Die alte Art, Mocks zu erzeugen. Mocking ist damit etwas unhandlich, aber dafür kann es auch partielle Mocks erzeugen.⁶

Mockery: Auch sehr elegant.⁷

⁵Prophecy-Cheatsheet:

<https://github.com/oliverklee/tdd-reader/blob/master/AdditionalDocuments/prophecy-cheatsheet.pdf>

⁶PHPUnit-Mocking-Cheatsheet:

<https://github.com/oliverklee/tdd-reader/blob/master/AdditionalDocuments/mocking-cheatsheet.pdf>

⁷<https://github.com/mockery/mockery>

6 Auf Exceptions testen

6.1 Nur auf die Klasse testen

```
1  /**
2   * @test
3   * @expectedException \UnexpectedValueException
4   */
5  public function createBreadWithNegativeSizeThrowsException()
6  {
7      $this->subject->createBread(-1);
8  }
```

6.2 Auf Klasse, Nachricht und Code testen

```
1  /**
2   * @test
3   * @expectedException \UnexpectedValueException
4   * @expectedExceptionMessage size must be > 0.
5   * @expectedExceptionCode 1323700434
6   */
7  public function createBreadWithNegativeSizeThrowsException()
8  {
9      $this->subject->createBread(-1);
10 }
```

7 Abstrakte Klassen testen

7.1 Den PHPUnit-Mock-Builder benutzen

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

```
1 namespace OliverKlee\Coffee\Tests\Unit\Domain\Model;
2
3 use OliverKlee\Coffee\Domain\Model\AbstractBeverage;
4
5 class AbstractBeverageTest {
6     /**
7      * @var AbstractBeverage|\PHPUnit\Framework\MockObject\MockObject
8      */
9     protected $subject = null;
10
11     protected function setUp()
12     {
13         $this->subject = $this->getMockForAbstractClass(
14             AbstractBeverage::class
15         );
16     }
17 }
```

7.2 Eine konkrete Unterklasse erstellen

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

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

```
1 namespace OliverKlee\Coffee\Tests\Unit\Domain\Model\Fixtures;
2
3 class TestingBeverage extends \OliverKlee\Coffee\Domain\Model\AbstractBeverage {
4     // ...
5 }
```

Dann könnt ihr die konkrete Unterklasse in euren Unit-Tests nutzen und instanziierten.

```
1 use OliverKlee\Coffee\Tests\Unit\Domain\Model\Fixtures\TestingBeverage;
2
3 class AbstractBeverageTest {
4     /**
5      * @var TestingBeverage
6      */
7     protected $subject = null;
8
9     protected function setUp()
10     {
11         $this->subject = new TestingBeverage();
12     }
13 }
```

8 Das Test-Framework der PHPUnit-TYPO3-Extension benutzen

```
1 class DataMapperTest extends \Tx_Phpunit_TestCase {
2     /**
3      * @var \Tx_Phpunit_Framework
4      */
5     protected $testingFramework = null;
6
7     protected $subject = null;
8
9     protected function setUp()
10    {
11        $this->testingFramework = new \Tx_Phpunit_Framework('tx_oelib');
12
13        $this->subject = new ...;
14    }
15
16    protected function tearDown()
17    {
18        $this->testingFramework->cleanUp();
19    }
20
21    /**
22     * @test
23     */
24    public function findWithUidOfExistingRecordReturnsModelDataFromDatabase()
25    {
26        $title = 'foo';
27        $uid = $this->testingFramework->createRecord(
28            'tx_oelib_test', ['title' => $title]
29        );
30
31        self::assertSame($title, $this->subject->find($uid)->getTitle());
32    }
```

9 Gemockte Dateisystem mit vfsStream benutzen

9.1 Lauffähige Beispiele

Die funktionalen Tests zur FileUtility-Klasse im Tea-Beispiel zeigen, wie Tests mit vfsStream aussehen können.

9.2 Einrichten

```
1 use org\bovigo\vfs\vfsStream;
2 use org\bovigo\vfs\vfsStreamDirectory;
3
4 /**
5  * @var |org\bovigo\vfs\vfsStreamFile
6  */
7 protected $moreStuff;
8
9 protected function setUp()
10 {
11     // This is the same as ::register and ::setRoot.
12     $this->root = vfsStream::setup('home');
13     $this->targetFilePath = vfsStream::url('home/target.txt');
14
15     $this->subject = new ...
16 }
```

9.3 Die Dateien benutzen

```
1 /**
2  * @test
3  */
4 public function concatenateWithOneEmptySourceFileCreatesEmptyTargetFile()
5 {
6     // This is one way to create a file with contents, using PHP's file functions.
7     $sourceFileName = vfsStream::url('home/source.txt');
8     // Just calling vfsStream::url does not create the file yet.
9     // We need to write into it to create it.
10    file_put_contents($sourceFileName, '');
11
12    $this->subject->concatenate($this->targetFilePath, [$sourceFileName]);
13
14    self::assertSame('', file_get_contents($this->targetFilePath));
15 }
16
17 /**
18  * @test
19  */
20 public function concatenateWithOneFileCopiesContentsFromSourceFileToTargetFile()
21 {
22     // This is vfsStream's way of creating a file with contents.
23     $contents = 'Hello world!';
24     $sourceFileName = vfsStream::url('home/source.txt');
25     vfsStream::newFile('source.txt')->at($this->root)->setContent($contents);
26 }
```

```
27     $this->subject->concatenate($this->targetFilePath, [$sourceFileName]);
28
29     self::assertSame($contents, file_get_contents($this->targetFilePath));
30 }
```

10 PHPUnit-Assertions

Diese Liste ist aktuell für PHPUnit 5.7.x.

```
assertArrayHasKey()
assertClassHasAttribute()
assertArraySubset()
assertClassHasStaticAttribute()
assertContains()
assertContainsOnly()
assertContainsOnlyInstancesOf()
assertCount()
assertDirectoryExists()
assertDirectoryIsReadable()
assertDirectoryIsWritable()
assertEmpty()
assertEqualXMLStructure()
assertEquals()
assertFalse()
assertFileEquals()
assertFileExists()
assertFileIsReadable()
assertFileIsWritable()
assertGreaterThan()
assertGreaterThanOrEqual()
assertInfinite()
assertInstanceOf()
assertInternalType()
assertIsReadable()
assertIsWritable()
assertJsonFileEqualsJsonFile()
assertJsonStringEqualsJsonFile()
assertJsonStringEqualsJsonString()
assertLessThan()
assertLessThanOrEqual()
assertNan()
assertNull()
assertObjectHasAttribute()
assertRegExp()
assertStringMatchesFormat()
assertStringMatchesFormatFile()
assertSame()
assertStringEndsWith()
assertStringEqualsFile()
assertStringStartsWith()
assertThat()
assertTrue()
assertXmlFileEqualsXmlFile()
assertXmlStringEqualsXmlFile()
assertXmlStringEqualsXmlString()
```