Cheatsheet for test-driven development with TYPO3 CMS

https://github.com/oliverklee/tdd-reader

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

Version 1.5.0, October 20, 2015

License

[1] https://github.com/oliverklee/tdd-reader
[2] http://creativecommons.org/licenses/by-sa/4.0/

Contents

1 File and class naming

1.1 File names

Production code file name	Test file name
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 Class names

Production code class name	Test class name
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 Test class structure

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 Non-extbase extensions

```
// You need to require_once the class-to-test if your extension
// does not make use of ext_autoload.php.
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 Non-TYPO3 PHP projects

```
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 Testing for Exceptions

3.1 Test for the Exception class only (recommended)

```
/**
 * @test
 * @expectedException InvalidArgumentException
 */
```

```
public function createBreadWithNegativeSizeThrowsException() {
  $this->subject->createBread(-1);
}
```

3.2 Test for the exception class, message and (optionally) the code (recommended)

```
/**
 * @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 (not recommended)

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

4 Testing abstract classes

4.1 Using the PHPUnit mock builder (recommended)

This will create an instance of the abstract class with all abstract methods mocked.

```
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 Creating a concrete subclass (recommended)

This is recommended if you need to provide your subclass with some additional or specific behavior.

In `Tests/Unit/Fixtures/`, create a subclass of the abstract class:

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

Then you can include and instantiate the concrete subclass in your unit tests:

```
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 (not recommended)

This is not recommended as this breaks code completion in your IDE.

5 Using the testing framework of the phpunit TYPO3 extension

```
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 Using mock file systems with vfsStream

6.1 Setting it all up

```
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 Using the files

```
/**
 * @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

This list is current for 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 $messa
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 =
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 = T
assertSelectEquals(array $selector, string $content, integer $count, mixed $actual[, string $message = '', b
assertSelectRegExp(array $selector, string $pattern, integer $count, mixed $actual[, string $message = '', b
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 = ''])
```