[Базовые методы сравнения 2](#_Toc501723273)

[assertTrue() / assertFalse() 2](#_Toc501723274)

[assertEquals() / assertNotEquals() 2](#_Toc501723275)

[assertGreaterThan() 7](#_Toc501723276)

[assertGreaterThanOrEqual() 7](#_Toc501723277)

[assertLessThan() 8](#_Toc501723278)

[assertLessThanOrEqual() 9](#_Toc501723279)

[assertNull() / assertNotNull() 9](#_Toc501723280)

[assertType() / assertNotType() 10](#_Toc501723281)

[assertSame() / assertNotSame() 10](#_Toc501723282)

[assertRegExp() / assertNotRegExp() 11](#_Toc501723283)

[Методы сравнения массивов 12](#_Toc501723284)

[assertArrayHasKey() / assertArrayNotHasKey() 12](#_Toc501723285)

[assertContains() / assertNotContains() 12](#_Toc501723286)

[assertContainsOnly() / assertNotContainsOnly() 13](#_Toc501723287)

[ООП специфичные методы 14](#_Toc501723288)

[assertClassHasAttribute() / assertClassNotHasAttribute() 14](#_Toc501723289)

[assertClassHasStaticAttribute() / assertClassNotHasStaticAttribute() 14](#_Toc501723290)

[assertAttributeContains() / assertAttributeNotContains() 14](#_Toc501723291)

[assertObjectHasAttribute() / assertObjectNotHasAttribute() 14](#_Toc501723292)

[assertAttributeGreaterThan() 15](#_Toc501723293)

[assertAttributeGreaterThanOrEqual() 15](#_Toc501723294)

[assertAttributeLessThan() 15](#_Toc501723295)

[assertAttributeLessThanOrEqual() 15](#_Toc501723296)

[Методы сравнения файлов 15](#_Toc501723297)

[assertFileEquals() / assertFileNotEquals() 15](#_Toc501723298)

[assertFileExists() / assertFileNotExists() 16](#_Toc501723299)

[assertStringEqualsFile() / assertStringNotEqualsFile() 16](#_Toc501723300)

[Методы сравнения XML 17](#_Toc501723301)

[assertEqualXMLStructure() 17](#_Toc501723302)

[assertXmlFileEqualsXmlFile() / assertXmlFileNotEqualsXmlFile() 19](#_Toc501723303)

[assertXmlStringEqualsXmlFile() / assertXmlStringNotEqualsXmlFile() 20](#_Toc501723304)

[assertXmlStringEqualsXmlString() / assertXmlStringNotEqualsXmlString() 21](#_Toc501723305)

[Разное 22](#_Toc501723306)

[assertTag() 22](#_Toc501723307)

[assertThat() 24](#_Toc501723308)

[Тестирование операций с массивами с использованием PHPUnit 25](#_Toc501723309)

[Использование аннотации @depends для описания зависимостей 26](#_Toc501723310)

[Использование зависимостей между тестами 27](#_Toc501723311)

[Использование провайдера данных, который возвращает массив массивов 28](#_Toc501723312)

[Тестирование исключений 31](#_Toc501723313)

[Тестирование вывода 36](#_Toc501723314)

[Утверждения (Assertions) 37](#_Toc501723315)

[assertClassHasStaticAttribute() 37](#_Toc501723316)

[assertContainsOnlyInstancesOf() 38](#_Toc501723317)

[assertCount() 38](#_Toc501723318)

[assertEmpty() 39](#_Toc501723319)

[assertInstanceOf() 40](#_Toc501723320)

[assertInternalType() 41](#_Toc501723321)

[assertJsonFileEqualsJsonFile() 41](#_Toc501723322)

[assertJsonStringEqualsJsonFile() 42](#_Toc501723323)

[assertJsonStringEqualsJsonString() 43](#_Toc501723324)

[assertObjectHasAttribute() 43](#_Toc501723325)

[assertStringMatchesFormat() 44](#_Toc501723326)

[assertStringMatchesFormatFile() 45](#_Toc501723327)

[assertSelectCount() 46](#_Toc501723328)

[assertSelectEquals() 47](#_Toc501723329)

[assertSelectRegExp() 48](#_Toc501723330)

[assertStringEndsWith() 50](#_Toc501723331)

[assertStringStartsWith() 51](#_Toc501723332)

# Базовые методы сравнения

## assertTrue() / assertFalse()

assertTrue()

assertTrue(bool $condition[, string $message = ''])

Сообщает об ошибке $message если $condition равно логическому FALSE.

**Пример 4.56. Использование assertTrue()**

<?php

class TrueTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertTrue(FALSE);

}

}

?>

**phpunit TrueTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) TrueTest::testFailure

Failed asserting that false is true.

/home/sb/TrueTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertEquals() / assertNotEquals()

assertEquals()

assertEquals(mixed $expected, mixed $actual[, string $message = ''])

Сообщает об ошибке $message если переменные $expected и $actual не равны между собой.

assertNotEquals() утверждает обратное и принимает тот же набор аргументов.

assertAttributeEquals() и assertAttributeNotEquals() это удобные обёртки, которые используют публичные, защищенные, или приватные аттрибуты класса или объекта в качестве области поиска.

**Пример 4.24. Использование assertEquals()**

<?php

class EqualsTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertEquals(1, 0);

}

public function testFailure2()

{

$this->assertEquals('bar', 'baz');

}

public function testFailure3()

{

$this->assertEquals("foo\nbar\nbaz\n", "foo\nbah\nbaz\n");

}

}

?>

**phpunit EqualsTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

FFF

Time: 0 seconds, Memory: 5.25Mb

There were 3 failures:

1) EqualsTest::testFailure

Failed asserting that 0 matches expected 1.

/home/sb/EqualsTest.php:6

2) EqualsTest::testFailure2

Failed asserting that two strings are equal.

--- Expected

+++ Actual

@@ @@

-'bar'

+'baz'

/home/sb/EqualsTest.php:11

3) EqualsTest::testFailure3

Failed asserting that two strings are equal.

--- Expected

+++ Actual

@@ @@

'foo

-bar

+bah

baz

'

/home/sb/EqualsTest.php:16

FAILURES!

Tests: 3, Assertions: 3, Failures: 3.

Более специфические сравнения, которые используются для определённых типов аргументов $expected и $actual смотрите ниже.

assertEquals(float $expected, float $actual[, string $message = '', float $delta = 0])

Сообщает об ошибке $message если два числ а с плавающей запятой $expected и $actual отличаются более чем на $delta друг от друга.

Смотрите " [Что нужно знать про арифметику с плавающей запятой](http://yapro.ru/system/redirect?to=http://habrahabr.ru/post/112953/) " чтобы понять зачем нужна $delta

**Пример 4.25. Использование assertEquals() с числами с плавающей запятой floats**

<?php

class EqualsTest extends PHPUnit\_Framework\_TestCase

{

public function testSuccess()

{

$this->assertEquals(1.0, 1.1, '', 0.2);

}

public function testFailure()

{

$this->assertEquals(1.0, 1.1);

}

}

?>

**phpunit EqualsTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

.F

Time: 0 seconds, Memory: 5.75Mb

There was 1 failure:

1) EqualsTest::testFailure

Failed asserting that 1.1 matches expected 1.0.

/home/sb/EqualsTest.php:11

FAILURES!

Tests: 2, Assertions: 2, Failures: 1.

assertEquals(DOMDocument $expected, DOMDocument $actual[, string $message = ''])

Сообщает об ошибке $message если некомментированная каноническая форма XML документов представленых двумя объектами DOMDocument $expected и $actual не совпадает.

**Пример 4.26. Использование assertEquals() с объектами DOMDocument**

<?php

class EqualsTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$expected = new DOMDocument;

$expected->loadXML('<foo><bar/></foo>');

$actual = new DOMDocument;

$actual->loadXML('<bar><foo/></bar>');

$this->assertEquals($expected, $actual);

}

}

?>

**phpunit EqualsTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) EqualsTest::testFailure

Failed asserting that two DOM documents are equal.

--- Expected

+++ Actual

@@ @@

<?xml version="1.0"?>

-<foo>

- <bar/>

-</foo>

+<bar>

+ <foo/>

+</bar>

/home/sb/EqualsTest.php:12

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

assertEquals(object $expected, object $actual[, string $message = ''])

Сообщает об ошибке $message если два объекта $expected и $actual имеют неодинаковые аттрибуты.

**Пример 4.27. Использование assertEquals() с объектами**

<?php

class EqualsTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$expected = new stdClass;

$expected->foo = 'foo';

$expected->bar = 'bar';

$actual = new stdClass;

$actual->foo = 'bar';

$actual->baz = 'bar';

$this->assertEquals($expected, $actual);

}

}

?>

**phpunit EqualsTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) EqualsTest::testFailure

Failed asserting that two objects are equal.

--- Expected

+++ Actual

@@ @@

stdClass Object (

- 'foo' => 'foo'

- 'bar' => 'bar'

+ 'foo' => 'bar'

+ 'baz' => 'bar'

)

/home/sb/EqualsTest.php:14

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

assertEquals(array $expected, array $actual[, string $message = ''])

Сообщает об ошибке $message если два массива $expected и $actual не равны.

**Пример 4.28. Использование assertEquals() с массивами**

**<?php  
class** EqualsTest **extends** PHPUnit\_Framework\_TestCase  
{  
 **public function** testFailure()  
 {  
 $this->assertEquals(**array**('a', 'b', 'c'), **array**('a', 'c', 'd'));  
 }  
}  
**?>**

**phpunit EqualsTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) EqualsTest::testFailure

Failed asserting that two arrays are equal.

--- Expected

+++ Actual

@@ @@

Array (

0 => 'a'

- 1 => 'b'

- 2 => 'c'

+ 1 => 'c'

+ 2 => 'd'

)

/home/sb/EqualsTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

assertFalse()

assertFalse(bool $condition[, string $message = ''])

Сообщает об ошибке $message если $condition равно TRUE.

## assertGreaterThan()

assertGreaterThan()

assertGreaterThan(mixed $expected, mixed $actual[, string $message = ''])

Сообщает об ошибке $message если значение $actual не больше чем $expected.

assertAttributeGreaterThan() это удобные обёртки, которые используют публичные, защищенные, или приватные аттрибуты класса или объекта в качестве актуального значения.

**Пример 4.32. Использование assertGreaterThan()**

**<?php  
class** GreaterThanTest **extends** PHPUnit\_Framework\_TestCase  
{  
 **public function** testFailure()  
 {  
 $this->assertGreaterThan(2, 1);  
 }  
}  
**?>**

**phpunit GreaterThanTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) GreaterThanTest::testFailure

Failed asserting that 1 is greater than 2.

/home/sb/GreaterThanTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertGreaterThanOrEqual()

assertGreaterThanOrEqual()

assertGreaterThanOrEqual(mixed $expected, mixed $actual[, string $message = ''])

Сообщает об ошибке $message если значение $actual меньше значения $expected.

assertAttributeGreaterThanOrEqual() это удобные обёртки, которые используют публичные, защищенные, или приватные аттрибуты класса или объекта в качестве актуального значения.

**Пример 4.33. Использование assertGreaterThanOrEqual()**

<?php

class GreatThanOrEqualTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertGreaterThanOrEqual(2, 1);

}

}

?>

**phpunit GreaterThanOrEqualTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) GreatThanOrEqualTest::testFailure

Failed asserting that 1 is equal to 2 or is greater than 2.

/home/sb/GreaterThanOrEqualTest.php:6

FAILURES!

Tests: 1, Assertions: 2, Failures: 1.

## assertLessThan()

assertLessThan()

assertLessThan(mixed $expected, mixed $actual[, string $message = ''])

Сообщает об ошибке $message если значение $actual больше или равно значению $expected.

assertAttributeLessThan() удобная обёртка, которая позволяет применять assertLessThan к public, protected, или private аттрибута класса или объекта.

**Пример 4.39. Использование assertLessThan()**

<?php

class LessThanTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertLessThan(1, 2);

}

}

?>

**phpunit LessThanTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) LessThanTest::testFailure

Failed asserting that 2 is less than 1.

/home/sb/LessThanTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertLessThanOrEqual()

assertLessThanOrEqual()

assertLessThanOrEqual(mixed $expected, mixed $actual[, string $message = ''])

Сообщает об ошибке $message если значение $actual строго больше чем $expected.

assertAttributeLessThanOrEqual() удобная обёртка, которая позволяет применять assertLessThanOrEqual к public, protected, или private аттрибута класса или объекта.

**Пример 4.40. Использование assertLessThanOrEqual()**

<?php

class LessThanOrEqualTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertLessThanOrEqual(1, 2);

}

}

?>

**phpunit LessThanOrEqualTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) LessThanOrEqualTest::testFailure

Failed asserting that 2 is equal to 1 or is less than 1.

/home/sb/LessThanOrEqualTest.php:6

FAILURES!

Tests: 1, Assertions: 2, Failures: 1.

## assertNull() / assertNotNull()

assertNull()

assertNull(mixed $variable[, string $message = ''])

Сообщает об ошибке $message если $variable не NULL.

assertNotNull() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.41. Использование assertNull()**

<?php

class NullTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertNull('foo');

}

}

?>

**phpunit NotNullTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) NullTest::testFailure

Failed asserting that 'foo' is null.

/home/sb/NotNullTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertType() / assertNotType()

## assertSame() / assertNotSame()

assertSame()

assertSame(mixed $expected, mixed $actual[, string $message = ''])

Сообщает об ошибке $message если две переменных $expected и $actual не совпадают одновременно по типу и значению.

assertNotSame() утверждает обратное и принимает тот же набор аргументов.

assertAttributeSame() и assertAttributeNotSame() удобные обёртки для сравнения public, protected, or private аттрибутов классов или объектов.

**Пример 4.46. Использование assertSame()**

<?php

class SameTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertSame('2204', 2204);

}

}

?>

**phpunit SameTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) SameTest::testFailure

Failed asserting that 2204 is identical to '2204'.

/home/sb/SameTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

assertSame(object $expected, object $actual[, string $message = ''])

Сообщает об ошибке $message если две переменные $expected и $actual не ссылаются на один и тот же объект.

**Пример 4.47. Использование assertSame() with objects**

<?php

class SameTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertSame(new stdClass, new stdClass);

}

}

?>

**phpunit SameTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) SameTest::testFailure

Failed asserting that two variables reference the same object.

/home/sb/SameTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertRegExp() / assertNotRegExp()

assertRegExp()

assertRegExp(string $pattern, string $string[, string $message = ''])

Сообщает об ошибке $message если строка $string не соответствует регулярному выражению $pattern.

assertNotRegExp() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.43. Использование assertRegExp()**

<?php

class RegExpTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertRegExp('/foo/', 'bar');

}

}

?>

**phpunit RegExpTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) RegExpTest::testFailure

Failed asserting that 'bar' matches PCRE pattern "/foo/".

/home/sb/RegExpTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

Методы сравнения массивов

## assertArrayHasKey() / assertArrayNotHasKey()

assertArrayHasKey()

assertArrayHasKey(mixed $key, array $array[, string $message = ''])

Сообщает об ошибке $message если $array не содержит ключа $key.

assertArrayNotHasKey() утверждает обратное и принимает тот же набор аргументов

**Пример 4.14. Использование assertArrayHasKey()**

<?php

class ArrayHasKeyTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertArrayHasKey('foo', array('bar' => 'baz'));

}

}

?>

**phpunit ArrayHasKeyTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ArrayHasKeyTest::testFailure

Failed asserting that an array has the key 'foo'.

/home/sb/ArrayHasKeyTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertContains() / assertNotContains()

assertContains()

assertContains(mixed $needle, Iterator|array $haystack[, string $message = ''])

Сообщает об ошибке $message если $needle не является элементом $haystack.

assertNotContains() утверждает обратное и принимает тот же набор аргументов.

assertAttributeContains() и assertAttributeNotContains() это удобные обёртки, которые используют публичные, защищенные, или приватные аттрибуты класса или объекта в качестве области поиска.

**Пример 4.17. Использование assertContains()**

<?php

class ContainsTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertContains(4, array(1, 2, 3));

}

}

?>

**phpunit ContainsTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ContainsTest::testFailure

Failed asserting that an array contains 4.

/home/sb/ContainsTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

assertContains(string $needle, string $haystack[, string $message = ''])

Сообщает об ошибке $message если $needle не является подстрокой $haystack.

**Пример 4.18. Использование assertContains()**

<?php

class ContainsTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertContains('baz', 'foobar');

}

}

?>

**phpunit ContainsTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ContainsTest::testFailure

Failed asserting that 'foobar' contains "baz".

/home/sb/ContainsTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertContainsOnly() / assertNotContainsOnly()

assertContainsOnly()

assertContainsOnly(string $type, Iterator|array $haystack[, boolean $isNativeType = NULL, string $message = ''])

Сообщает об ошибке $message если $haystack содержит элементы отличные по типу от $type.

$isNativeType флаг который определяет является ли $type встроенным типом PHP или нет.

assertNotContainsOnly() утверждает обратное и принимает тот же набор аргументов.

assertAttributeContainsOnly() и assertAttributeNotContainsOnly() это удобные обёртки, которые используют публичные, защищенные, или приватные аттрибуты класса или объекта в качестве области поиска.

**Пример 4.19. Использование assertContainsOnly()**

<?php

class ContainsOnlyTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertContainsOnly('string', array('1', '2', 3));

}

}

?>

**phpunit ContainsOnlyTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ContainsOnlyTest::testFailure

Failed asserting that Array (

0 => '1'

1 => '2'

2 => 3

) contains only values of type "string".

/home/sb/ContainsOnlyTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

ООП специфичные методы

## assertClassHasAttribute() / assertClassNotHasAttribute()

## assertClassHasStaticAttribute() / assertClassNotHasStaticAttribute()

## assertAttributeContains() / assertAttributeNotContains()

## assertObjectHasAttribute() / assertObjectNotHasAttribute()

assertClassHasAttribute()

assertClassHasAttribute(string $attributeName, string $className[, string $message = ''])

Сообщает об ошибке $message если $className::attributeName не существует.

assertClassNotHasAttribute() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.15. Использование assertClassHasAttribute()**

<?php

class ClassHasAttributeTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertClassHasAttribute('foo', 'stdClass');

}

}

?>

**phpunit ClassHasAttributeTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) ClassHasAttributeTest::testFailure

Failed asserting that class "stdClass" has attribute "foo".

/home/sb/ClassHasAttributeTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertAttributeGreaterThan()

## assertAttributeGreaterThanOrEqual()

## assertAttributeLessThan()

## assertAttributeLessThanOrEqual()

Методы сравнения файлов

## assertFileEquals() / assertFileNotEquals()

assertFileEquals()

assertFileEquals(string $expected, string $actual[, string $message = ''])

Сообщает об ошибке $message если файл $expected не равен по содержимому файлу $actual.

assertFileNotEquals() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.30. Использование assertFileEquals()**

<?php

class FileEqualsTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertFileEquals('/home/sb/expected', '/home/sb/actual');

}

}

?>

**phpunit FileEqualsTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) FileEqualsTest::testFailure

Failed asserting that two strings are equal.

--- Expected

+++ Actual

@@ @@

-'expected

+'actual

'

/home/sb/FileEqualsTest.php:6

FAILURES!

Tests: 1, Assertions: 3, Failures: 1.

## assertFileExists() / assertFileNotExists()

assertFileExists()

assertFileExists(string $filename[, string $message = ''])

Сообщает об ошибке $message если файл $filename не существует.

assertFileNotExists() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.31. Использование assertFileExists()**

<?php

class FileExistsTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertFileExists('/path/to/file');

}

}

?>

**phpunit FileExistsTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) FileExistsTest::testFailure

Failed asserting that file "/path/to/file" exists.

/home/sb/FileExistsTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertStringEqualsFile() / assertStringNotEqualsFile()

assertStringEqualsFile()

assertStringEqualsFile(string $expectedFile, string $actualString[, string $message = ''])

Сообщает об ошибке $message если файл указанный $expectedFile не содержит строки $actualString.

assertStringNotEqualsFile() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.52. Использование assertStringEqualsFile()**

<?php

class StringEqualsFileTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertStringEqualsFile('/home/sb/expected', 'actual');

}

}

?>

**phpunit StringEqualsFileTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) StringEqualsFileTest::testFailure

Failed asserting that two strings are equal.

--- Expected

+++ Actual

@@ @@

-'expected

-'

+'actual'

/home/sb/StringEqualsFileTest.php:6

FAILURES!

Tests: 1, Assertions: 2, Failures: 1.

Методы сравнения XML

## assertEqualXMLStructure()

assertEqualXMLStructure()

assertEqualXMLStructure(DOMElement $expectedElement, DOMElement $actualElement[, boolean $checkAttributes = FALSE, string $message = ''])

Сообщает об ошибке $message если структура XML элемента DOM в $actualElement не совпадает с структурой XML элемента DOM в$expectedElement.

**Пример 4.23. Использование assertEqualXMLStructure()**

<?php

class EqualXMLStructureTest extends PHPUnit\_Framework\_TestCase

{

public function testFailureWithDifferentNodeNames()

{

$expected = new DOMElement('foo');

$actual = new DOMElement('bar');

$this->assertEqualXMLStructure($expected, $actual);

}

public function testFailureWithDifferentNodeAttributes()

{

$expected = new DOMDocument;

$expected->loadXML('<foo bar="true" />');

$actual = new DOMDocument;

$actual->loadXML('<foo/>');

$this->assertEqualXMLStructure(

$expected->firstChild, $actual->firstChild, TRUE

);

}

public function testFailureWithDifferentChildrenCount()

{

$expected = new DOMDocument;

$expected->loadXML('<foo><bar/><bar/><bar/></foo>');

$actual = new DOMDocument;

$actual->loadXML('<foo><bar/></foo>');

$this->assertEqualXMLStructure(

$expected->firstChild, $actual->firstChild

);

}

public function testFailureWithDifferentChildren()

{

$expected = new DOMDocument;

$expected->loadXML('<foo><bar/><bar/><bar/></foo>');

$actual = new DOMDocument;

$actual->loadXML('<foo><baz/><baz/><baz/></foo>');

$this->assertEqualXMLStructure(

$expected->firstChild, $actual->firstChild

);

}

}

?>

**phpunit EqualXMLStructureTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

FFFF

Time: 0 seconds, Memory: 5.75Mb

There were 4 failures:

1) EqualXMLStructureTest::testFailureWithDifferentNodeNames

Failed asserting that two strings are equal.

--- Expected

+++ Actual

@@ @@

-'foo'

+'bar'

/home/sb/EqualXMLStructureTest.php:9

2) EqualXMLStructureTest::testFailureWithDifferentNodeAttributes

Number of attributes on node "foo" does not match

Failed asserting that 0 matches expected 1.

/home/sb/EqualXMLStructureTest.php:22

3) EqualXMLStructureTest::testFailureWithDifferentChildrenCount

Number of child nodes of "foo" differs

Failed asserting that 1 matches expected 3.

/home/sb/EqualXMLStructureTest.php:35

4) EqualXMLStructureTest::testFailureWithDifferentChildren

Failed asserting that two strings are equal.

--- Expected

+++ Actual

@@ @@

-'bar'

+'baz'

/home/sb/EqualXMLStructureTest.php:48

FAILURES!

Tests: 4, Assertions: 8, Failures: 4.

## assertXmlFileEqualsXmlFile() / assertXmlFileNotEqualsXmlFile()

assertXmlFileEqualsXmlFile()

assertXmlFileEqualsXmlFile(string $expectedFile, string $actualFile[, string $message = ''])

Сообщает об ошибке $message если XML документ в файле $actualFile не равен XML документу из файла $expectedFile.

assertXmlFileNotEqualsXmlFile() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.57. Использование assertXmlFileEqualsXmlFile()**

<?php

class XmlFileEqualsXmlFileTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertXmlFileEqualsXmlFile(

'/home/sb/expected.xml', '/home/sb/actual.xml');

}

}

?>

**phpunit XmlFileEqualsXmlFileTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) XmlFileEqualsXmlFileTest::testFailure

Failed asserting that two DOM documents are equal.

--- Expected

+++ Actual

@@ @@

<?xml version="1.0"?>

<foo>

- <bar/>

+ <baz/>

</foo>

/home/sb/XmlFileEqualsXmlFileTest.php:7

FAILURES!

Tests: 1, Assertions: 3, Failures: 1.

## assertXmlStringEqualsXmlFile() / assertXmlStringNotEqualsXmlFile()

assertXmlStringEqualsXmlFile()

assertXmlStringEqualsXmlFile(string $expectedFile, string $actualXml[, string $message = ''])

Сообщает об ошибке $message если XML документ в строке $actualXml не равен XML документу из файла $expectedFile.

assertXmlStringNotEqualsXmlFile() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.58. Использование assertXmlStringEqualsXmlFile()**

<?php

class XmlStringEqualsXmlFileTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertXmlStringEqualsXmlFile(

'/home/sb/expected.xml', '<foo><baz/></foo>');

}

}

?>

**phpunit XmlStringEqualsXmlFileTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) XmlStringEqualsXmlFileTest::testFailure

Failed asserting that two DOM documents are equal.

--- Expected

+++ Actual

@@ @@

<?xml version="1.0"?>

<foo>

- <bar/>

+ <baz/>

</foo>

/home/sb/XmlStringEqualsXmlFileTest.php:7

FAILURES!

Tests: 1, Assertions: 2, Failures: 1.

## assertXmlStringEqualsXmlString() / assertXmlStringNotEqualsXmlString()

assertXmlStringEqualsXmlString()

assertXmlStringEqualsXmlString(string $expectedXml, string $actualXml[, string $message = ''])

Сообщает об ошибке $message если XML документ в строке $actualXml не равен XML документу в строке $expectedXml.

assertXmlStringNotEqualsXmlString() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.59. Использование assertXmlStringEqualsXmlString()**

<?php

class XmlStringEqualsXmlStringTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertXmlStringEqualsXmlString(

'<foo><bar/></foo>', '<foo><baz/></foo>');

}

}

?>

**phpunit XmlStringEqualsXmlStringTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) XmlStringEqualsXmlStringTest::testFailure

Failed asserting that two DOM documents are equal.

--- Expected

+++ Actual

@@ @@

<?xml version="1.0"?>

<foo>

- <bar/>

+ <baz/>

</foo>

/home/sb/XmlStringEqualsXmlStringTest.php:7

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

Разное

## assertTag()

assertTag()

assertTag(array $matcher, string $actual[, string $message = '', boolean $isHtml = TRUE])

Сообщает об ошибке $message если $actual не соответствует спецификации $matcher.

$matcher - это ассоциативный массив который описывает критерии соответствия для допущения:

id: Значение которому должен быть равен аттрибут id элемента.

tag: Ожидаемый тип элемента.

attributes: Аттрибуты элемента должны совпадать с соответствующими значениями в ассоциативном массиве $attributes.

content: Текстовое содержимое элемента должно совпадать со значением этого ключа массива.

parent: Родитель элемента должен совпадать с описаным в ассоциативном масиве $parent.

child: Хотя бы один прямой потомок элемента должен соответствовать критериям ассоциативного массива$child.

ancestor: Хотя бы один из предков элемента должен соответствовать критериям из массива $ancestor.

descendant: Хотя бы один из потомков (включая непрямых) должен соответствовать критериям из массива $descendant.

children: Ассоциативный массив количества прямых потомков

count: Количество потомков должно быть равно этому числу.

less\_than: Количество потомков должно быть меньше этого числа.

greater\_than: Количество потомков должно быть больше этого числа.

only: Еще один ассоциативный массив состоящий из ключей соответствующих потомкам и только соответствующие потомки будут учтены.

assertNotTag() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.54. Использование assertTag()**

<?php

// Массив который описывает элемент с id="my\_id".

$matcher = array('id' => 'my\_id');

// Массив описывающий наличие тега "span".

$matcher = array('tag' => 'span');

// Массив описывающий наличие тега "span" содержащего текст

// "Hello World".

$matcher = array('tag' => 'span', 'content' => 'Hello World');

// Массив описывающий наличие тега "span" содержащего текст

// соответствующий регулярному выражению

$matcher = array('tag' => 'span', 'content' => '/Try P(HP|ython)/');

// Массив описывающий наличие тега "span" с аттрибутом class равным "list".

$matcher = array(

'tag' => 'span',

'attributes' => array('class' => 'list')

);

// Массив описывающий наличие тега "span" непосредственно внутри тега "div".

$matcher = array(

'tag' => 'span',

'parent' => array('tag' => 'div')

);

// Массив описывающий наличие тега "span" где-то внутри тега "table".

$matcher = array(

'tag' => 'span',

'ancestor' => array('tag' => 'table')

);

// Массив описывающий наличие тега "span" с хотя бы одним "em" среди прямых потомков.

$matcher = array(

'tag' => 'span',

'child' => array('tag' => 'em')

);

// Массив описывающий наличие тега "span" с хотя бы одним "strong" где-то среди потомков включая непрямых.

$matcher = array(

'tag' => 'span',

'descendant' => array('tag' => 'strong')

);

// Массив описывающий наличие тега "span" содержащего "em" тегов

// в качестве прямых потомков.

$matcher = array(

'tag' => 'span',

'children' => array(

'less\_than' => 11,

'greater\_than' => 4,

'only' => array('tag' => 'em')

)

);

// Массив описывающий наличие тега "div", среди предков котрого есть "ul" а тег "li"

// (с классом "enum") является его прямым предком, содержащий где-то среди потомков тег "span" который содержит

// тег с id="my\_test" и текстом "Hello World".

$matcher = array(

'tag' => 'div',

'ancestor' => array('tag' => 'ul'),

'parent' => array(

'tag' => 'li',

'attributes' => array('class' => 'enum')

),

'descendant' => array(

'tag' => 'span',

'child' => array(

'id' => 'my\_test',

'content' => 'Hello World'

)

)

);

// Использование assertTag() для роверки наличия элемента $matcher в переменной $html.

$this->assertTag($matcher, $html);

// Использование assertTag() для роверки наличия элемента $matcher в переменной $xml.

$this->assertTag($matcher, $xml, '', FALSE);

?>

## assertThat()

assertThat()

Более сложные утверждения могут быть сформулированы с использованием классов PHPUnit\_Framework\_Constraint. Они могут быть вычислены с использованием метода assertThat(). [Пример 4.55, «Использование assertThat()»](#writing-tests-for-phpunit.assertions.as) показывает как с помощью условий logicalNot() и equalTo() можно выразить то же утверждение что и с помощью assertNotEquals().

assertThat(mixed $value, PHPUnit\_Framework\_Constraint $constraint[, $message = ''])

Сообщает об ошибке $message если значение $value не соответствует условию $constraint.

**Пример 4.55. Использование assertThat()**

<?php

class BiscuitTest extends PHPUnit\_Framework\_TestCase

{

public function testEquals()

{

$theBiscuit = new Biscuit('Ginger');

$myBiscuit = new Biscuit('Ginger');

$this->assertThat(

$theBiscuit,

$this->logicalNot(

$this->equalTo($myBiscuit)

)

);

}

}

?>

# Тестирование операций с массивами с использованием PHPUnit

Пример «Тестирование операций с массивами с использованием PHPUnit» показывает, как можно писать тесты на PHPUnit для операций с массивами в PHP. Этот пример представляет базовые соглашения и шаги для написания тестов с помощью PHPUnit:

Тесты для класса Class содержатся в классе ClassTest.

ClassTest унаследован (чаще всего) от PHPUnit\_Framework\_TestCase.

Тесты — это публичные методы с именами test\*.

Кроме этого, можно использовать аннотацию @test в док-блоке чтобы пометить метод как тест.

Внутри тестовых методов для проверки того что реальные данные соответствуют ожидаемым используются методы-утверждения (assertion methods) такие как assertEquals().

**Пример Тестирование операций с массивами с использованием PHPUnit**

<?*php***class** StackTest **extends** PHPUnit\_Framework\_TestCase  
{  
 **public function** testPushAndPop()  
 {  
 $stack = **array**();  
 $this->assertEquals(0, count($stack));  
  
 array\_push($stack, 'foo');  
 $this->assertEquals('foo', $stack[count($stack)-1]);  
 $this->assertEquals(1, count($stack));  
  
 $this->assertEquals('foo', array\_pop($stack));  
 $this->assertEquals(0, count($stack));  
 }  
}  
**?>**

Всякий раз, когда вы испытываете искушение напечатать что-то внутри оператора print или отладочного выражения, напишите это как тест.

--Мартин Фаулер

Межтестовые зависимости

Модульные тесты главным образом в виде практики которая помогает разработчикам идентифицировать и исправить баги, реорганизовывать (refactor) код и служит в качестве документации для модуля тестируемой программы. Для того чтобы достичь этих преимуществ, модульные тесты, в идеале, должны охватывать все возможные пути исполнения программы. Один модульный тест обычно покрывает один конкретный путь в одной функции или методе. Однако тестовый метод это не обязательно инкапсулированная, независимая сущность. Часто существуют неявные зависимости между тестовыми методами, скрытые в сценарии реализации теста.

--Adrian Kuhn et. al.

PHPUnit поддерживает описание явных зависимостей между тестовыми методами. Эти зависимости не определяют порядок, в котором тесты будут выполняться, но они позволяют передавать тестовые данные (фикстуры), созданные поставщиком, (producer) потребителям (consumers), которые от него зависят.

Поставщик - это тестовый метод, который предоставляет свой тестируемый модуль в качестве возвращаемого значения.

Потребитель - это тестовый метод, который зависит от одного или более поставщиков и их возвращаемых значений.

# Использование аннотации @depends для описания зависимостей

Пример «Использование аннотации @depends для описания зависимостей» показывает, как использовать аннотацию @depends для описания зависимостей между тестовыми методами.

**Пример Использование аннотации @depends для описания зависимостей**

<?*php***class** StackTest **extends** PHPUnit\_Framework\_TestCase  
{  
 **public function** testEmpty()  
 {  
 $stack = **array**();  
 $this->assertEmpty($stack);  
  
 **return** $stack;  
 }  
  
 */\*\*  
 \** ***@depends*** *testEmpty  
 \*/* **public function** testPush(**array** $stack)  
 {  
 array\_push($stack, 'foo');  
 $this->assertEquals('foo', $stack[count($stack)-1]);  
 $this->assertNotEmpty($stack);  
  
 **return** $stack;  
 }  
  
 */\*\*  
 \** ***@depends*** *testPush  
 \*/* **public function** testPop(**array** $stack)  
 {  
 $this->assertEquals('foo', array\_pop($stack));  
 $this->assertEmpty($stack);  
 }  
}  
**?>**

В приведённом выше примере первый тест, testEmpty(), создаёт новый массив и утверждает, что массив пустой. Затем тест возвращает фикстуру в виде результата. Второй тест, testPush(), зависит от testEmpty() и получает результат от теста-зависимости в качестве аргумента. Наконец, testPop() зависит от testPush().

# Использование зависимостей между тестами

Для того чтобы находить дефекты быстро, нам нужно сконцентрировать внимание на соответствующих тестах. Для этого PHPUnit пропускает исполнение тестов, когда зависимость провалилась. Это улучшает локализацию дефектов за счёт использования зависимостей между тестами как показано в Пример «Использование зависимостей между тестами».

**Пример Использование зависимостей между тестами**

<?*php***class** DependencyFailureTest **extends** PHPUnit\_Framework\_TestCase  
{  
 **public function** testOne()  
 {  
 $this->assertTrue(**FALSE**);  
 }  
  
 */\*\*  
 \** ***@depends*** *testOne  
 \*/* **public function** testTwo()  
 {  
 }  
}  
**?>**

**phpunit --verbose DependencyFailureTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

FS

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) DependencyFailureTest::testOne

Failed asserting that false is true.

/home/sb/DependencyFailureTest.php:6

There was 1 skipped test:

1) DependencyFailureTest::testTwo

This test depends on "DependencyFailureTest::testOne" to pass.

FAILURES!

Tests: 1, Assertions: 1, Failures: 1, Skipped: 1.

# Использование провайдера данных, который возвращает массив массивов

У теста может быть несколько аннотаций @depends. PHPUnit не изменяет порядок, в котором будут выполняться тесты, поэтому вы должны убедиться, что все зависимости теста удовлетворены до его выполнения.

Провайдеры данных

Тестовый метод может принимать произвольные аргументы. Эти аргументы должны быть предоставлены методом - провайдером данных (provider() в Пример «Использование провайдера данных, который возвращает массив массивов »). Метод, который будет использован в качестве провайдера данных, обозначается с помощью аннотации @dataProvider.

Метод провайдера данных должен быть public и возвращать либо массив массивов, либо объект реализующий интерфейс Iterator, который возвращает массив при каждой итерации. Для каждого массива, являющегося частью коллекции будет вызван тестовый метод с элементами массива в качестве параметров

**Пример 4.4. Использование провайдера данных, который возвращает массив массивов**

<?php

class DataTest extends PHPUnit\_Framework\_TestCase

{

/\*\*

\* @dataProvider provider

\*/

public function testAdd($a, $b, $c)

{

$this->assertEquals($c, $a + $b);

}

public function provider()

{

return array(

array(0, 0, 0),

array(0, 1, 1),

array(1, 0, 1),

array(1, 1, 3)

);

}

}

?>

**phpunit DataTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

...F

Time: 0 seconds, Memory: 5.75Mb

There was 1 failure:

1) DataTest::testAdd with data set #3 (1, 1, 3)

Failed asserting that 2 matches expected 3.

/home/sb/DataTest.php:9

FAILURES!

Tests: 4, Assertions: 4, Failures: 1.

**Пример 4.5. Использование провайдера данных, который возвращает объект-итератор**

<?php

require 'CsvFileIterator.php';

class DataTest extends PHPUnit\_Framework\_TestCase

{

/\*\*

\* @dataProvider provider

\*/

public function testAdd($a, $b, $c)

{

$this->assertEquals($c, $a + $b);

}

public function provider()

{

return new CsvFileIterator('data.csv');

}

}

?>

**phpunit DataTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

...F

Time: 0 seconds, Memory: 5.75Mb

There was 1 failure:

1) DataTest::testAdd with data set #3 ('1', '1', '3')

Failed asserting that 2 matches expected '3'.

/home/sb/DataTest.php:11

FAILURES!

Tests: 4, Assertions: 4, Failures: 1.

**Пример 4.6. Класс CsvFileIterator**

<?php

class CsvFileIterator implements Iterator {

protected $file;

protected $key = 0;

protected $current;

public function \_\_construct($file) {

$this->file = fopen($file, 'r');

}

public function \_\_destruct() {

fclose($this->file);

}

public function rewind() {

rewind($this->file);

$this->current = fgetcsv($this->file);

$this->key = 0;

}

public function valid() {

return !feof($this->file);

}

public function key() {

return $this->key;

}

public function current() {

return $this->current;

}

public function next() {

$this->current = fgetcsv($this->file);

$this->key++;

}

}

?>

Примечание

Если тест получает данные как от @dataProvider, так и от теста (или тестов) от которых он зависит (@depends), аргументы от провайдера будут переданы в тест первыми, а аргументы от @depends добавлены в конец списка аргументов.

Примечание

Если один тест зависит от другого теста, который использует провайдер, тест-потребитель выполнится если тест-поставщик успешно пройдёт хотя бы для одного набора параметров из провайдера. Результат выполнения теста-поставщика, который использует провайдера, не может быть передан тесту-потребителю.

Примечание

Все функции-провайдеры выполняются до того, как впервые будет вызван метод setUp. Поэтому, вы не сможете получить доступ к переменным, которые вы определили в нём изнутри провайдера.

# Тестирование исключений

[Пример 4.7, «Использование аннотации @expectedException»](#writing-tests-for-phpunit.exceptions.ex) показывает как c помощью аннотации @expectedException проверить было ли вызвано исключение в тестируемом коде.

**Пример 4.7. Использование аннотации @expectedException**

<?php

class ExceptionTest extends PHPUnit\_Framework\_TestCase

{

/\*\*

\* @expectedException InvalidArgumentException

\*/

public function testException()

{

}

}

?>

**phpunit ExceptionTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) ExceptionTest::testException

Expected exception InvalidArgumentException

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

Дополнительно вы можете использовать @expectedExceptionMessage и @expectedExceptionCode в сочетании с @expectedException для проверки сообщений и кодов исключений как показано в [Пример 4.8, « Использование аннотаций @expectedExceptionMessage и @expectedExceptionCode »](#writing-tests-for-phpunit.exceptions.ex).

**Пример 4.8. Использование аннотаций @expectedExceptionMessage и @expectedExceptionCode**

<?php

class ExceptionTest extends PHPUnit\_Framework\_TestCase

{

/\*\*

\* @expectedException InvalidArgumentException

\* @expectedExceptionMessage Right Message

\*/

public function testExceptionHasRightMessage()

{

throw new InvalidArgumentException('Some Message', 10);

}

/\*\*

\* @expectedException InvalidArgumentException

\* @expectedExceptionCode 20

\*/

public function testExceptionHasRightCode()

{

throw new InvalidArgumentException('Some Message', 10);

}

}

?>

**phpunit ExceptionTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

FF

Time: 0 seconds, Memory: 3.00Mb

There were 2 failures:

1) ExceptionTest::testExceptionHasRightMessage

Failed asserting that exception message 'Some Message' contains 'Right Message'.

2) ExceptionTest::testExceptionHasRightCode

Failed asserting that expected exception code 20 is equal to 10.

FAILURES!

Tests: 2, Assertions: 4, Failures: 2.

[«@expectedExceptionMessage»](http://yapro.ru/web-master/php/appendixes.annotations.html#appendixes.annotations.expectedExceptionMessage) и [«@expectedExceptionCode»](http://yapro.ru/web-master/php/appendixes.annotations.html#appendixes.annotations.expectedExceptionCode) приводят другие примеры использования @expectedExceptionMessage и @expectedExceptionCode соответственно.

В качестве альтернативы, можно использовать метод setExpectedException(), чтобы задать ожидаемое исключение [Пример 4.9, «Ожидаем исключение которое будет вызвано тестовым кодом»](#writing-tests-for-phpunit.exceptions.ex).

**Пример 4.9. Ожидаем исключение которое будет вызвано тестовым кодом**

<?php

class ExceptionTest extends PHPUnit\_Framework\_TestCase

{

public function testException()

{

$this->setExpectedException('InvalidArgumentException');

}

public function testExceptionHasRightMessage()

{

$this->setExpectedException(

'InvalidArgumentException', 'Right Message'

);

throw new InvalidArgumentException('Some Message', 10);

}

public function testExceptionHasRightCode()

{

$this->setExpectedException(

'InvalidArgumentException', 'Right Message', 20

);

throw new InvalidArgumentException('The Right Message', 10);

}

}?>

**phpunit ExceptionTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

FFF

Time: 0 seconds, Memory: 3.00Mb

There were 3 failures:

1) ExceptionTest::testException

Expected exception InvalidArgumentException

2) ExceptionTest::testExceptionHasRightMessage

Failed asserting that exception message 'Some Message' contains 'Right Message'.

3) ExceptionTest::testExceptionHasRightCode

Failed asserting that expected exception code 20 is equal to 10.

FAILURES!

Tests: 3, Assertions: 6, Failures: 3.

[Таблица 4.1, «Методы тестирования исключений»](#writing-tests-for-phpunit.exceptions.ta) представляет методы для проверки исключений

**Таблица 4.1. Методы тестирования исключений**

**Метод**

**Назначение**

void setExpectedException(string $exceptionName[, string $exceptionMessage = '', integer $exceptionCode = NULL])

Устанавливает тип ($exceptionName) ,сообщение ($exceptionMessage), и код ($exceptionCode.) ожидаемого исключения.

String getExpectedException()

Возвращает тип ожидаемого исключения.

Также для тестирования исключения вы можете использовать подход который иллюстрирует пример [Пример 4.10, «Альтернативный подход к тестированию исключений»](#writing-tests-for-phpunit.exceptions.ex)

**Пример 4.10. Альтернативный подход к тестированию исключений**

<?php

class ExceptionTest extends PHPUnit\_Framework\_TestCase {

public function testException() {

try {

// ... Код, который вызывает исключение ...

}

catch (InvalidArgumentException $expected) {

return;

}

$this->fail('Ожидаемое исключение не было вызвано.');

}

}

?>

[Пример 4.10, «Альтернативный подход к тестированию исключений»](#writing-tests-for-phpunit.exceptions.ex) демонстрирует, что если код от которого мы ожидали вызов исключения не вызовет его, последующий вызов метода fail() прервёт тест и сообщит о проблеме в тесте. Если ожидаемое исключение было вызвано, блок catch будет выполнен и тест завершится успешно.

Тестирование ошибок PHP

По умолчанию, PHPUnit преобразует ошибки, предупреждения и замечания (PHP Error, PHP Warning, PHP Notice соответственно), которые возникают во время выполнения теста в исключения. Используя эти исключения вы можете, например, что тест вызовет ошибку PHP, как это показывает [Пример 4.11, «Ожидание ошибки PHP с использованием @expectedException»](#writing-tests-for-phpunit.exceptions.ex).

**Пример 4.11. Ожидание ошибки PHP с использованием @expectedException**

<?php

class ExpectedErrorTest extends PHPUnit\_Framework\_TestCase

{

/\*\*

\* @expectedException PHPUnit\_Framework\_Error

\*/

public function testFailingInclude()

{

include 'not\_existing\_file.php';

}

}

?>

**phpunit ExpectedErrorTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

.

Time: 0 seconds, Memory: 5.25Mb

OK (1 test, 1 assertion)

PHPUnit\_Framework\_Error\_Notice и PHPUnit\_Framework\_Error\_Warning представляют замечания и предупреждения PHP соответственно.

Примечание

Вы должны быть как можно более конкретны когда теститруете исключения. Проверка слишком общих классов может приветси к нежелательным побочным эффектам. Поэтому проверка на встроенный класс Exception при помощи @expectedException или setExpectedException() больше не разрешена.

Когда вы тестируете код, основаный на функциях, которые вызывают ошибки, например fopen иногда может оказаться полезным подавление ошибок во время тестирования. Это позволит вам проверять возвращаемые значения подавляя замечания, которые иначе привели бы phpunit к PHPUnit\_Framework\_Error\_Notice.

**Пример 4.12. Testing return values of code that uses PHP Errors**

<?php

class ErrorSuppressionTest extends PHPUnit\_Framework\_TestCase

{

public function testFileWriting() {

$writer = new FileWriter;

$this->assertFalse(@$writer->write('/is-not-writeable/file', 'stuff'));

}

}

class FileWriter

{

public function write($file, $content) {

$file = fopen($file, 'w');

if($file == false) {

return false;

}

// ...

}

}

?>

**phpunit ErrorSuppressionTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

.

Time: 1 seconds, Memory: 5.25Mb

OK (1 test, 1 assertion)

Без подавления ошибок мы бы получили fopen(/is-not-writeable/file): failed to open stream: No such file or directory.

# Тестирование вывода

Иногда вам нужно проверить что выполнение метода, например, даёт нам ожидаемый вывод (через echo или print) Класс PHPUnit\_Framework\_TestCase использует [Буферизацию Вывода](http://yapro.ru/system/redirect?to=http://www.php.net/manual/en/ref.outcontrol.php) для того чтобы предоставить такую возможность.

[Пример 4.13, «Тестирование вывода функции или метода»](#writing-tests-for-phpunit.output.exampl) демонстрирует как использовать метод expectOutputString() для того чтобы задать ожидаемый вывод. Если это вывод не будет сгенерирован тест отмечен как проваленый.

**Пример 4.13. Тестирование вывода функции или метода**

<?php

class OutputTest extends PHPUnit\_Framework\_TestCase

{

public function testExpectFooActualFoo()

{

$this->expectOutputString('foo');

print 'foo';

}

public function testExpectBarActualBaz()

{

$this->expectOutputString('bar');

print 'baz';

}

}

?>

**phpunit OutputTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

.F

Time: 0 seconds, Memory: 5.75Mb

There was 1 failure:

1) OutputTest::testExpectBarActualBaz

Failed asserting that two strings are equal.

--- Expected

+++ Actual

@@ @@

-'bar'

+'baz'

FAILURES!

Tests: 2, Assertions: 2, Failures: 1.

[Таблица 4.2, «Методы для тестирования вывода»](#writing-tests-for-phpunit.output.tables) представлены в таблице

**Таблица 4.2. Методы для тестирования вывода**

**Метод**

**Назначение**

void expectOutputRegex(string $regularExpression)

Задаёт регулярное выражение $regularExpression которому должен соответствовать вывод

void expectOutputString(string $expectedString)

Устанавливает ожидаемое значение $expectedString которому должен быть равен вывод.

bool setOutputCallback(callable $callback)

Устанавливает callback-функцию которая может быть использована, например, для нормализации вывода

Примечание

Обратите внимание, что PHPUnit проглатывает весь вывод который генерируется во время выполнения теста. В строгом режиме (strict mode) тест, кторый генерирует вывод провалится

# Утверждения (Assertions)

Эта секция перечисляет различные доступные методы-утверждения

## assertClassHasStaticAttribute()

assertClassHasStaticAttribute(string $attributeName, string $className[, string $message = ''])

Сообщает об ошибке $message если статический аттрибут $className::attributeName не существет.

assertClassNotHasStaticAttribute() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.16. Использование assertClassHasStaticAttribute()**

<?php

class ClassHasStaticAttributeTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertClassHasStaticAttribute('foo', 'stdClass');

}

}

?>

**phpunit ClassHasStaticAttributeTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) ClassHasStaticAttributeTest::testFailure

Failed asserting that class "stdClass" has static attribute "foo".

/home/sb/ClassHasStaticAttributeTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertContainsOnlyInstancesOf()

assertContainsOnlyInstancesOf(string $classname, Traversable|array $haystack[, string $message = ''])

Сообщает об ошибке $message если $haystack содержит что-то кроме экземпляров класса $classname.

**Пример 4.20. Использование assertContainsOnlyInstancesOf()**

<?php

class ContainsOnlyInstancesOfTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertContainsOnlyInstancesOf('Foo', array(new Foo(), new Bar(), new Foo()));

}

}

?>

**phpunit ContainsOnlyInstancesOfTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ContainsOnlyInstancesOfTest::testFailure

Failed asserting that Array ([0]=> Bar Object(...)) is an instance of class "Foo".

/home/sb/ContainsOnlyInstancesOfTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertCount()

assertCount($expectedCount, $haystack[, string $message = ''])

Сообщает об ошибке $message если количество элементов в $haystack не равно $expectedCount.

assertNotCount() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.21. Использование assertCount()**

<?php

class CountTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertCount(0, array('foo'));

}

}

?>

**phpunit CountTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) CountTest::testFailure

Failed asserting that actual size 1 matches expected size 0.

/home/sb/CountTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertEmpty()

assertEmpty(mixed $actual[, string $message = ''])

Сообщает об ошибке $message если $actual не пустое.

assertNotEmpty() утверждает обратное и принимает тот же набор аргументов.

assertAttributeEmpty() и assertAttributeNotEmpty() это удобные обёртки, которые используют публичные, защищенные, или приватные аттрибуты класса или объекта в качестве области поиска.

**Пример 4.22. Использование assertEmpty()**

<?php

class EmptyTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertEmpty(array('foo'));

}

}

?>

**phpunit EmptyTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) EmptyTest::testFailure

Failed asserting that an array is empty.

/home/sb/EmptyTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

**Пример 4.29. Использование assertFalse()**

<?php

class FalseTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertFalse(TRUE);

}

}

?>

**phpunit FalseTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) FalseTest::testFailure

Failed asserting that true is false.

/home/sb/FalseTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertInstanceOf()

assertInstanceOf($expected, $actual[, $message = ''])

Сообщает об ошибке $message если $actual не является экземпляром $expected.

assertNotInstanceOf() утверждает обратное и принимает тот же набор аргументов.

assertAttributeInstanceOf() и assertAttributeNotInstanceOf() это удобные обёртки, которые можно использовать с публичными, защищенными, или приватными аттрибутом класса или объекта.

**Пример 4.34. Использование assertInstanceOf()**

<?php

class InstanceOfTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertInstanceOf('RuntimeException', new Exception);

}

}

?>

**phpunit InstanceOfTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) InstanceOfTest::testFailure

Failed asserting that Exception Object (...) is an instance of class "RuntimeException".

/home/sb/InstanceOfTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertInternalType()

assertInternalType($expected, $actual[, $message = ''])

Сообщает об ошибке $message если $actual не является переменной типа $expected.

assertNotInternalType() утверждает обратное и принимает тот же набор аргументов.

assertAttributeInternalType() и assertAttributeNotInternalType() это удобные обёртки, которые можно использовать с публичным, защищенным, или приватным аттрибутом класса или объекта.

**Пример 4.35. Использование assertInternalType()**

<?php

class InternalTypeTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertInternalType('string', 42);

}

}

?>

**phpunit InternalTypeTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) InternalTypeTest::testFailure

Failed asserting that 42 is of type "string".

/home/sb/InternalTypeTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertJsonFileEqualsJsonFile()

assertJsonFileEqualsJsonFile(mixed $expectedFile, mixed $actualFile[, string $message = ''])

Сообщает об ошибке $message если значение $actualFile совпадает с $expectedFile.

**Пример 4.36. Использование assertJsonFileEqualsJsonFile()**

<?php

class JsonFileEqualsJsonFileTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertJsonFileEqualsJsonFile(

'path/to/fixture/file', 'path/to/actual/file');

}

}

?>

**phpunit JsonFileEqualsJsonFileTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) JsonFileEqualsJsonFile::testFailure

Failed asserting that '{"Mascott":"Tux"}' matches JSON string "["Mascott", "Tux", "OS", "Linux"]".

/home/sb/JsonFileEqualsJsonFileTest.php:5

FAILURES!

Tests: 1, Assertions: 3, Failures: 1.

## assertJsonStringEqualsJsonFile()

assertJsonStringEqualsJsonFile(mixed $expectedFile, mixed $actualJson[, string $message = ''])

Сообщает об ошибке $message если строка $actualJson не совпадает с содержимым $expectedFile.

**Пример 4.37. Использование assertJsonStringEqualsJsonFile()**

<?php

class JsonStringEqualsJsonFileTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertJsonStringEqualsJsonFile(

'path/to/fixture/file', json\_encode(array("Mascott" => "ux"));

}

}

?>

**phpunit JsonStringEqualsJsonFileTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) JsonStringEqualsJsonFile::testFailure

Failed asserting that '{"Mascott":"ux"}' matches JSON string "{"Mascott":"Tux"}".

/home/sb/JsonStringEqualsJsonFileTest.php:5

FAILURES!

Tests: 1, Assertions: 3, Failures: 1.

## assertJsonStringEqualsJsonString()

assertJsonStringEqualsJsonString(mixed $expectedJson, mixed $actualJson[, string $message = ''])

Сообщает об ошибке $message если десериализованая строка $actualJson не равна десериализованой строке $expectedJson.

**Пример 4.38. Использование assertJsonStringEqualsJsonString()**

<?php

class JsonStringEqualsJsonStringTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertJsonStringEqualsJsonString(

json\_encode(array("Mascott" => "Tux"), json\_encode(array("Mascott" => "ux"));

}

}

?>

**phpunit JsonStringEqualsJsonStringTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) JsonStringEqualsJsonStringTest::testFailure

Failed asserting that two objects are equal.

--- Expected

+++ Actual

@@ @@

stdClass Object (

- 'Mascott' => 'Tux'

+ 'Mascott' => 'ux'

)

/home/sb/JsonStringEqualsJsonStringTest.php:5

FAILURES!

Tests: 1, Assertions: 3, Failures: 1.

## assertObjectHasAttribute()

assertObjectHasAttribute(string $attributeName, object $object[, string $message = ''])

Сообщает об ошибке $message если $object->attributeName не существует.

assertObjectNotHasAttribute() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.42. Использование assertObjectHasAttribute()**

<?php

class ObjectHasAttributeTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertObjectHasAttribute('foo', new stdClass);

}

}

?>

**phpunit ObjectHasAttributeTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) ObjectHasAttributeTest::testFailure

Failed asserting that object of class "stdClass" has attribute "foo".

/home/sb/ObjectHasAttributeTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertStringMatchesFormat()

assertStringMatchesFormat(string $format, string $string[, string $message = ''])

Сообщает об ошибке $message если строка $string не соответствует формату $format

assertStringNotMatchesFormat() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.44. Использование assertStringMatchesFormat()**

<?php

class StringMatchesFormatTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertStringMatchesFormat('%i', 'foo');

}

}

?>

**phpunit StringMatchesFormatTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) StringMatchesFormatTest::testFailure

Failed asserting that 'foo' matches PCRE pattern "/^[+-]?\d+$/s".

/home/sb/StringMatchesFormatTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

Строка формата может содержать следующие шаблоны:

%e: Обозначает разделитель папок, например / для Linux.

%s: Один или более любых символов (включая пробелы) кроме символа конца строки.

%S: Любое количество (ноль или больше) любых символов (включая пробелы) кроме символа конца строки.

%a: Один или более любых символов включая пробелы и символ конца строки.

%A: Любое количество (ноль или больше) любых символов, включая пробелы и символ конца строки.

%w: Любое количество (ноль или больше) пробельных символов.

%i: Целое число со знаком, например+3142, -3142.

%d: Целое число без знака, например123456.

%x: Один или более One or more шестнадцатиричных символов. Т.е. символов 0-9, a-f, A-F.

%f: Число с плавающей запятой, например: 3.142, -3.142, 3.142E-10, 3.142e+10.

%c: Один любой символ.

## assertStringMatchesFormatFile()

assertStringMatchesFormatFile(string $formatFile, string $string[, string $message = ''])

Сообщает об ошибке $message если строка $string не удовлетворяет формату из $formatFile.

assertStringNotMatchesFormatFile() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.45. Использование assertStringMatchesFormatFile()**

<?php

class StringMatchesFormatFileTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertStringMatchesFormatFile('/path/to/expected.txt', 'foo');

}

}

?>

**phpunit StringMatchesFormatFileTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) StringMatchesFormatFileTest::testFailure

Failed asserting that 'foo' matches PCRE pattern "/^[+-]?\d+

$/s".

/home/sb/StringMatchesFormatFileTest.php:6

FAILURES!

Tests: 1, Assertions: 2, Failures: 1.

## assertSelectCount()

assertSelectCount(array $selector, integer $count, mixed $actual[, string $message = '', boolean $isHtml = TRUE])

Сообщает об ошибке $message если CSS селектор $selector выбирает из DOMNode $actual отличное от $count количество элементов.

$count может быть одного из перечисленных типов:

boolean: Проверяет простое наличие (TRUE) или отсутствие (FALSE) элементов соответствующих селектору.

integer: Проверяет количество элементов.

array: Проверяет что количество элементов находится в заданном диапазоне заданном ключами массива в виде <, >, <=, и >=.

**Пример 4.48. Использование assertSelectCount()**

<?php

class SelectCountTest extends PHPUnit\_Framework\_TestCase

{

protected function setUp()

{

$this->xml = new DomDocument;

$this->xml->loadXML('<foo><bar/><bar/><bar/></foo>');

}

public function testAbsenceFailure()

{

$this->assertSelectCount('foo bar', FALSE, $this->xml);

}

public function testPresenceFailure()

{

$this->assertSelectCount('foo baz', TRUE, $this->xml);

}

public function testExactCountFailure()

{

$this->assertSelectCount('foo bar', 5, $this->xml);

}

public function testRangeFailure()

{

$this->assertSelectCount('foo bar', array('>'=>6, '<'=>8), $this->xml);

}

}

?>

**phpunit SelectCountTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

FFFF

Time: 0 seconds, Memory: 5.50Mb

There were 4 failures:

1) SelectCountTest::testAbsenceFailure

Failed asserting that true is false.

/home/sb/SelectCountTest.php:12

2) SelectCountTest::testPresenceFailure

Failed asserting that false is true.

/home/sb/SelectCountTest.php:17

3) SelectCountTest::testExactCountFailure

Failed asserting that 3 matches expected 5.

/home/sb/SelectCountTest.php:22

4) SelectCountTest::testRangeFailure

Failed asserting that false is true.

/home/sb/SelectCountTest.php:27

FAILURES!

Tests: 4, Assertions: 4, Failures: 4.

## assertSelectEquals()

assertSelectEquals(array $selector, string $content, integer $count, mixed $actual[, string $message = '', boolean $isHtml = TRUE])

Сообщает об ошибке $message если CSS селектор $selector выбирает из DOMNode $actual отличное от $count количество элементов со значением $content.

$count может быть одного из перечисленных типов:

boolean: Проверяет простое наличие (TRUE) или отсутствие (FALSE) элементов соответствующих селектору.

integer: Проверяет количество элементов.

array: Проверяет что количество элементов находится в заданном диапазоне заданном ключами массива в виде <, >, <=, и >=.

**Пример 4.49. Использование assertSelectEquals()**

<?php

class SelectEqualsTest extends PHPUnit\_Framework\_TestCase

{

protected function setUp()

{

$this->xml = new DomDocument;

$this->xml->loadXML('<foo><bar>Baz</bar><bar>Baz</bar></foo>');

}

public function testAbsenceFailure()

{

$this->assertSelectEquals('foo bar', 'Baz', FALSE, $this->xml);

}

public function testPresenceFailure()

{

$this->assertSelectEquals('foo bar', 'Bat', TRUE, $this->xml);

}

public function testExactCountFailure()

{

$this->assertSelectEquals('foo bar', 'Baz', 5, $this->xml);

}

public function testRangeFailure()

{

$this->assertSelectEquals('foo bar', 'Baz', array('>'=>6, '<'=>8), $this->xml);

}

}

?>

**phpunit SelectEqualsTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

FFFF

Time: 0 seconds, Memory: 5.50Mb

There were 4 failures:

1) SelectEqualsTest::testAbsenceFailure

Failed asserting that true is false.

/home/sb/SelectEqualsTest.php:12

2) SelectEqualsTest::testPresenceFailure

Failed asserting that false is true.

/home/sb/SelectEqualsTest.php:17

3) SelectEqualsTest::testExactCountFailure

Failed asserting that 2 matches expected 5.

/home/sb/SelectEqualsTest.php:22

4) SelectEqualsTest::testRangeFailure

Failed asserting that false is true.

/home/sb/SelectEqualsTest.php:27

FAILURES!

Tests: 4, Assertions: 4, Failures: 4.

## assertSelectRegExp()

assertSelectRegExp(array $selector, string $pattern, integer $count, mixed $actual[, string $message = '', boolean $isHtml = TRUE])

Сообщает об ошибке $message если CSS селектор $selector выбирает из DOMNode $actual отличное от $count количество элементов со значением которое соответствуе регулярному выражению $pattern.

$count может быть одного из перечисленных типов:

boolean: Проверяет простое наличие (TRUE) или отсутствие (FALSE) элементов соответствующих селектору.

integer: Проверяет количество элементов.

array: Проверяет что количество элементов находится в заданном диапазоне заданном ключами массива в виде <, >, <=, и >=.

**Пример 4.50. Использование assertSelectRegExp()**

<?php

class SelectRegExpTest extends PHPUnit\_Framework\_TestCase

{

protected function setUp()

{

$this->xml = new DomDocument;

$this->xml->loadXML('<foo><bar>Baz</bar><bar>Baz</bar></foo>');

}

public function testAbsenceFailure()

{

$this->assertSelectRegExp('foo bar', '/Ba.\*/', FALSE, $this->xml);

}

public function testPresenceFailure()

{

$this->assertSelectRegExp('foo bar', '/B[oe]z]/', TRUE, $this->xml);

}

public function testExactCountFailure()

{

$this->assertSelectRegExp('foo bar', '/Ba.\*/', 5, $this->xml);

}

public function testRangeFailure()

{

$this->assertSelectRegExp('foo bar', '/Ba.\*/', array('>'=>6, '<'=>8), $this->xml);

}

}

?>

**phpunit SelectRegExpTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

FFFF

Time: 0 seconds, Memory: 5.50Mb

There were 4 failures:

1) SelectRegExpTest::testAbsenceFailure

Failed asserting that true is false.

/home/sb/SelectRegExpTest.php:12

2) SelectRegExpTest::testPresenceFailure

Failed asserting that false is true.

/home/sb/SelectRegExpTest.php:17

3) SelectRegExpTest::testExactCountFailure

Failed asserting that 2 matches expected 5.

/home/sb/SelectRegExpTest.php:22

4) SelectRegExpTest::testRangeFailure

Failed asserting that false is true.

/home/sb/SelectRegExpTest.php:27

FAILURES!

Tests: 4, Assertions: 4, Failures: 4.

## assertStringEndsWith()

assertStringEndsWith(string $suffix, string $string[, string $message = ''])

Сообщает об ошибке $message если строка $string не заканчивается строкой $suffix.

assertStringEndsNotWith() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.51. Использование assertStringEndsWith()**

<?php

class StringEndsWithTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertStringEndsWith('suffix', 'foo');

}

}

?>

**phpunit StringEndsWithTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 1 second, Memory: 5.00Mb

There was 1 failure:

1) StringEndsWithTest::testFailure

Failed asserting that 'foo' ends with "suffix".

/home/sb/StringEndsWithTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

## assertStringStartsWith()

assertStringStartsWith(string $prefix, string $string[, string $message = ''])

Сообщает об ошибке $message если строка $string не начинается строкой $prefix.

assertStringStartsNotWith() утверждает обратное и принимает тот же набор аргументов.

**Пример 4.53. Использование assertStringStartsWith()**

<?php

class StringStartsWithTest extends PHPUnit\_Framework\_TestCase

{

public function testFailure()

{

$this->assertStringStartsWith('prefix', 'foo');

}

}

?>

**phpunit StringStartsWithTest**

PHPUnit 3.7.0 by Sebastian Bergmann.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) StringStartsWithTest::testFailure

Failed asserting that 'foo' starts with "prefix".

/home/sb/StringStartsWithTest.php:6

FAILURES!

Tests: 1, Assertions: 1, Failures: 1.

[Таблица 4.3, «Условия»](#writing-tests-for-phpunit.assertions.as) показывает набор доступных под-классов PHPUnit\_Framework\_Constraint.

**Таблица 4.3. Условия**

**Условие**

**Значение**

attribute(PHPUnit\_Framework\_Constraint $constraint, $attributeName)

Условие которое применяет другое условие к класса или объекта.

PHPUnit\_Framework\_Constraint\_IsAnything anything()

Этому условию удовлетворяет любое входное значение.

PHPUnit\_Framework\_Constraint\_ArrayHasKey arrayHasKey(mixed $key)

Этому условию соответствует массив содержит ключ $key.

PHPUnit\_Framework\_Constraint\_TraversableContains contains(mixed $value)

Этому условию соответствует массив или объект предоставляющий интерфейс Iterator который содержит некоторое значение $value.

PHPUnit\_Framework\_Constraint\_TraversableContainsOnly containsOnly(string $type)

Этому условию соответствует массив или объект предоставляющий интерфейс Iterator содержащий только значения определённого типа.

PHPUnit\_Framework\_Constraint\_TraversableContainsOnly containsOnlyInstancesOf(string $classname)

Этому условию соответствует массив или объект предоставляющий интерфейс Iterator содержащий только экземпляры класса определённого класса.

PHPUnit\_Framework\_Constraint\_IsEqual equalTo($value, $delta = 0, $maxDepth = 10)

Проверяет на равенство с заданной погрешностью и глубиной вложенности.

PHPUnit\_Framework\_Constraint\_Attribute attributeEqualTo($attributeName, $value, $delta = 0, $maxDepth = 10)

Проверяет на равенство аттрибута сласса или объекта заданному значению.

PHPUnit\_Framework\_Constraint\_FileExists fileExists()

Проверяет что файл с таким именем существует.

PHPUnit\_Framework\_Constraint\_GreaterThan greaterThan(mixed $value)

Этому условию соответствует значение большее чем некое заданное $value.

PHPUnit\_Framework\_Constraint\_Or greaterThanOrEqual(mixed $value)

Этому условию соответствует значение большее либо равное заданному $value.

PHPUnit\_Framework\_Constraint\_ClassHasAttribute classHasAttribute(string $attributeName)

Этому условию соответствует класс содержащий заданный аттрибут.

PHPUnit\_Framework\_Constraint\_ClassHasStaticAttribute classHasStaticAttribute(string $attributeName)

Этому условию соответствует класс содержащий заданный статический аттрибут.

PHPUnit\_Framework\_Constraint\_ObjectHasAttribute hasAttribute(string $attributeName)

Этому условию соответствует объект содержащий заданный аттрибут.

PHPUnit\_Framework\_Constraint\_IsIdentical identicalTo(mixed $value)

Этому условию соответствуют полностью идентичные значения.

PHPUnit\_Framework\_Constraint\_IsFalse isFalse()

Этому условию соответствует переменная когда она равна логическому FALSE.

PHPUnit\_Framework\_Constraint\_IsInstanceOf isInstanceOf(string $className)

Этому условию соответствует объект являющийся экземпляром заданного класса.

PHPUnit\_Framework\_Constraint\_IsNull isNull()

Этому условию соответствует переменная когда она равна NULL.

PHPUnit\_Framework\_Constraint\_IsTrue isTrue()

Этому условию соответствует переменная когда она равна логическому TRUE.

PHPUnit\_Framework\_Constraint\_IsType isType(string $type)

Этому условию соответствует переменная когда она принадлежит к заданному типу.

PHPUnit\_Framework\_Constraint\_LessThan lessThan(mixed $value)

Этому условию соответствует значение меньшее чем некое заданное $value.

PHPUnit\_Framework\_Constraint\_Or lessThanOrEqual(mixed $value)

Этому условию соответствует значение меньшее либо равное заданному $value.

logicalAnd()

Логическое И.

logicalNot(PHPUnit\_Framework\_Constraint $constraint)

Логическое НЕ.

logicalOr()

Логическое ИЛИ.

logicalXor()

Логическое взаимоисключающее ИЛИ.

PHPUnit\_Framework\_Constraint\_PCREMatch matchesRegularExpression(string $pattern)

Проверяет соответствие строки заданному регулярному выражению.

PHPUnit\_Framework\_Constraint\_StringContains stringContains(string $string, bool $case)

Проверяет что заданная подстрока содержится в строке.

PHPUnit\_Framework\_Constraint\_StringEndsWith stringEndsWith(string $suffix)

Проверяет что строка оканчивается на заданный суфикс.

PHPUnit\_Framework\_Constraint\_StringStartsWith stringStartsWith(string $prefix)

Проверяет что строка начинается с заданного префикса.