

2021

CURSO DE PHPUNIT



»» Isabel González Anzano

Curso de PHPUnit para principiantes

Introducción	1
¿Qué son los test unitarios? ¿Qué es PHPUnit?	1
Instalación y Configuración	2
Requisitos de PHPUnit	2
Instalación con Phar, Composer y Global	3
Instalación con Phar	3
Instalación con Composer	4
Instalación Global	4
Phar vs. Composer	5
Instalación global en Ubuntu	5
Fichero XML de configuración	5
Características de PHPUnit	7
Estructura de las pruebas	7
Clase de prueba	7
Clase test	8
setUp() y tearDown()	8
Ejemplo	9
setUpBeforeClass() y tearDownAfterClass()	10
Ejemplo	10
Dependencias	11
Ejemplo @depends	11
Proveedores de datos	12
Ejemplo 1	13
Ejemplo 2	13
Casuísticas especiales	13
Probando excepciones	13
Ejemplo - Clase divide	14
Ejemplo -Test	15
Probando salidas de PHP	15
Ejemplo	15
Anotaciones	17
¿Qué son las anotaciones?	17
Ejemplo	17
Otras anotaciones	21
Aserciones	22
¿Qué son las aserciones?	22
Todas las aserciones están en la clase PHPUnit\Framework\Assert, y pueden ser llamados de dos formas:	22
Arrays	22
Ejemplos	23
Booleanos	23
Clases/Objetos	24
Comparadores	26
Genéricas	26
Otras aserciones	27

Introducción

¿Qué son los test unitarios? ¿Qué es PHPUnit?

Una prueba unitaria se encarga de comprobar el funcionamiento de una unidad de código, y su objetivo es aislar una parte del código para probar que funciona a la perfección.

Normalmente las desarrollan los desarrolladores y QA, y se implementan cuanto antes posible, incluso antes que el propio desarrollo (TDD).

Lo que busca PHPUnit es mejorar la calidad del proyecto y reducir errores, sobre todo de regresión. El coste de solventar un error aumenta en el tiempo, por tanto debemos solucionarlo cuanto antes, y para eso están las pruebas unitarias.

PRUEBAS UNITARIAS

¿Quién? Desarrolladores o QA

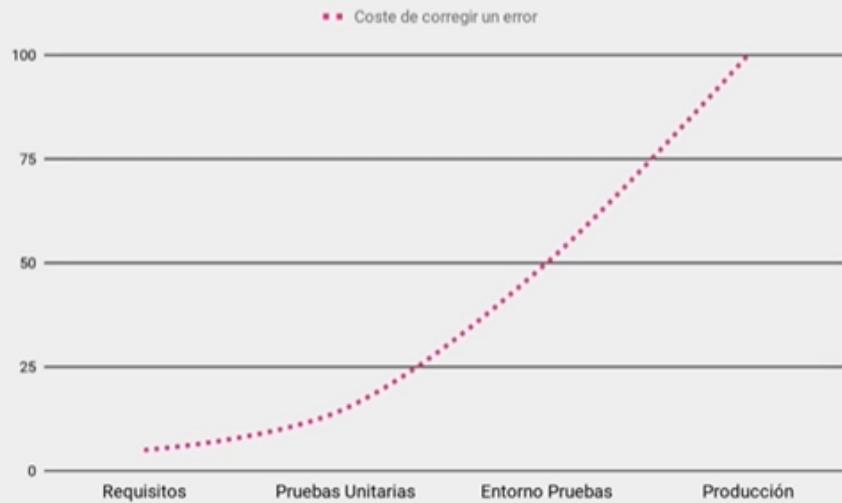
¿Cuándo? Cuanto antes (TDD)

¿Por qué? Ahorro en Tiempo y Costes

BB

Los errores no cometidos no necesitan ser corregidos

PRUEBAS UNITARIAS



Instalación y Configuración

Requisitos de PHPUnit

Se recomienda tener instalada la última versión de PHP siempre para ejecutar PHPUnit. (mínimo 7.2)

Con esta última versión, vienen ya instaladas por defecto las extensiones que son necesarias para PHPUnit: dom, json, pcce, reflection y spl.

Además, si queremos sacar la cobertura de código de las pruebas, necesitaremos las extensiones de Xdebug y Tokenizer.

REQUISITOS

- **Última versión de PHP (>= 7.2)**
- **Extensiones 'dom', 'json', 'pcre', 'reflection' y 'spl' (instaladas por defecto)**
- **Cobertura de código: Xdebug y Tokenizer**

Instalación con Phar, Composer y Global

Instalación con Phar

INSTALACIÓN CON PHAR

- **Todas las dependencias en un sólo fichero**
- **Extensión necesaria: 'phar'**

```
rubenrecacha@recacha:~/Documentos/OpenWebin... x rubenrecacha@recacha:~/Documentos/OpenWebin... x rubenrecacha@recacha:~/Documentos/OpenWebin... x
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/01_Phар$ wget -O phpu... https://ph...ar.phpunit.de/phpunit-7.phar
```

```
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/01_Phар$ ls
phpunit
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/01_Phар$ chmod +x phpunit
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/01_Phар$
```

```
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/01_Phар$ chmod +x phpunit
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/01_Phар$ ./phpunit --version
PHPUnit 7.5.17 by Sebastian Bergmann and contributors.
```

```
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/01_Phар$
```

Instalación con Composer

```
rubenrecacha@recacha:~/Documentos/OpenWebin... rubenrecacha@recacha:~/Documentos/OpenWebin... rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/02_Composer$ composer require --dev phpunit/phpunit ^7

sebastian/global-state suggests installing ext-uopz (*)
phpunit/phpunit suggests installing phpunit/php-invoker (^2.0)
phpunit/phpunit suggests installing ext-soap (*)
Writing lock file
Generating autoload files
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/02_Composer$ ls
composer.json composer.lock vendor
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/02_Composer$ cd vendor/
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/02_Composer/vendor$ ls
autoload.php composer myclabs phpdumper phpunit symfony webmozart
bin doctrine phar-io phpspec sebastian theseer
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/02_Composer/vendor$ █
```

Instalación Global

INSTALACIÓN GLOBAL

- **No recomendada**
- **PHPUnit debe ser gestionado como una dependencia de tu proyecto**

```
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/03_Global$ wget https://phar.phpunit.de/phpunit.phar
--2019-11-30 10:09:14-- https://phar.phpunit.de/phpunit.phar
Resolviendo phar.phpunit.de (phar.phpunit.de)... 188.94.27.25
Conectando con phar.phpunit.de (phar.phpunit.de)[188.94.27.25]:443... conectado.
Petición HTTP enviada, esperando respuesta... 302 Moved Temporarily
Ubicación: https://phar.phpunit.de/phpunit-8.4.3.phar [siguiente]
--2019-11-30 10:09:14-- https://phar.phpunit.de/phpunit-8.4.3.phar
Reutilizando la conexión con phar.phpunit.de:443.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 2881467 (2,7M) [application/octet-stream]
Guardando como: "phpunit.phar"

phpunit.phar          100%[=====] 2,75M  5,65MB/s   en 0,5s

2019-11-30 10:09:15 (5,65 MB/s) - "phpunit.phar" guardado [2881467/2881467]
```

```
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/03_Global$ chmod +x phpunit.phar
rubenrecacha@recacha:~/Documentos/OpenWebinars/Curso/02_Instal_Config/03_Global$ sudo mv phpunit.phar /usr/local/bin/phpunit
```

Phar vs. Composer

Con la instalación mediante el fichero phar, tendremos todas las dependencias necesarias en un solo fichero, aunque es necesario la extensión phar.

Por el contrario, si elegimos Composer para instalar phpunit, tendremos la ventaja de tener siempre actualizada la librería, y poder instalarla de forma sencilla con los comandos de composer.

Instalación global en Ubuntu

Recordar siempre que este tipo de instalación no es recomendado, ya que cada vez que quisiéramos ejecutar las pruebas en un entorno, tendríamos que instalarlo en dicho entorno.

Por el contrario, PHPUnit siempre debe estar gestionado como una dependencia en el proyecto (idealmente Composer).

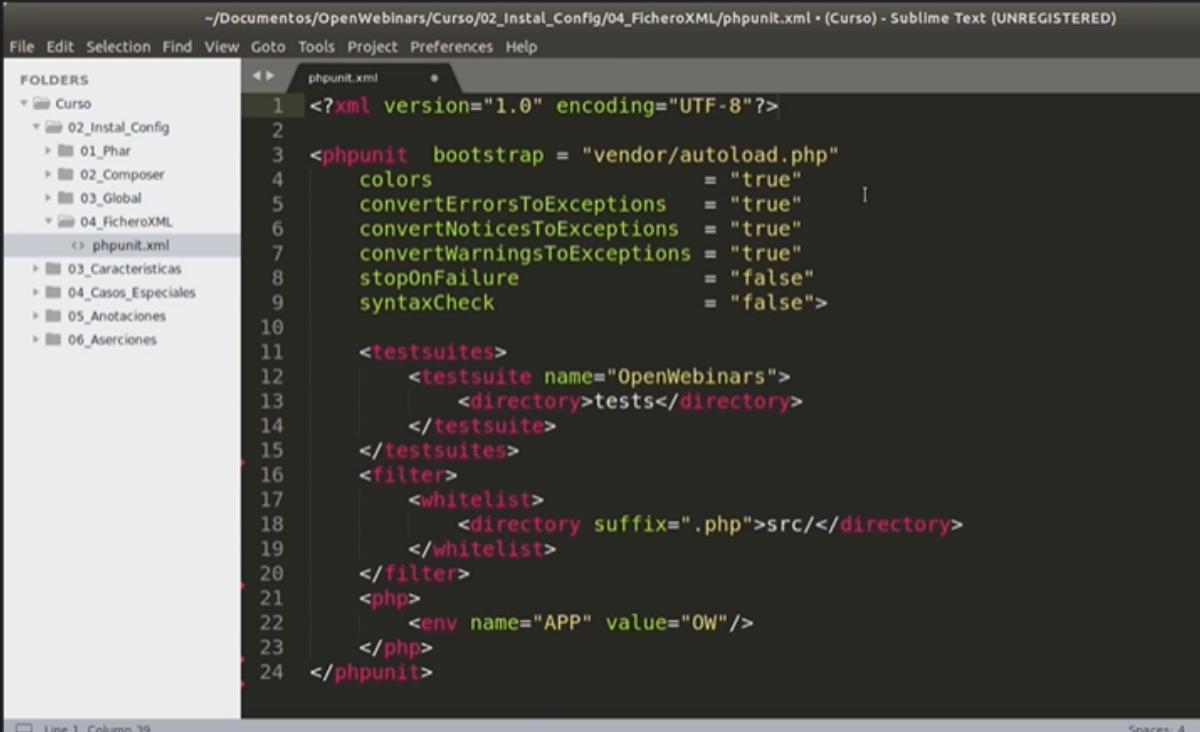
Fichero XML de configuración

Este fichero es uno de los puntos más importantes de PHPUnit, ya que definirá qué pruebas ejecutar, y cómo se organizarán.

Sirve para definir los grupos de pruebas, directorios y ficheros excluidos de las pruebas, filtros de ficheros para la cobertura de código, configuración de los logs o definición de constantes, entre otras.

FICHERO CONFIGURACIÓN XML

- **Definir suites de pruebas**
- **Excluir directorios o ficheros**
- **Filtros para la cobertura de código**
- **Configuración de Log**
- **Definir constantes y variables**



The screenshot shows a Sublime Text editor window with the following details:

- Title Bar:** ~/Documentos/OpenWebinars/Curso/02_Instal_Config/04_FicheroXML/phpunit.xml • (Curso) - Sublime Text (UNREGISTERED)
- Menu Bar:** File Edit Selection Find View Goto Tools Project Preferences Help
- Folders List:** Shows a tree view of the project structure:
 - Curso
 - 02_Instal_Config
 - 01_Phар
 - 02_Composer
 - 03_Global
 - 04_FicheroXML
 - phpunit.xml
 - 05_Características
 - 06_Casos_Especiales
 - 05_Anotaciones
 - 06_Asecciones
- Code Editor:** The main pane displays the XML configuration for PHPUnit. The code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit bootstrap = "vendor/autoload.php"
    colors = "true"
    convertErrorsToExceptions = "true"
    convertNoticesToExceptions = "true"
    convertWarningsToExceptions = "true"
    stopOnFailure = "false"
    syntaxCheck = "false">

    <testsuites>
        <testsuite name="OpenWebinars">
            <directory>tests</directory>
        </testsuite>
    </testsuites>
    <filter>
        <whitelist>
            <directory suffix=".php">src/</directory>
        </whitelist>
    </filter>
    <php>
        <env name="APP" value="OW"/>
    </php>
</phpunit>
```

Line 1, Column 39

Spaces: 4

Características de PHPUnit

Estructura de las pruebas

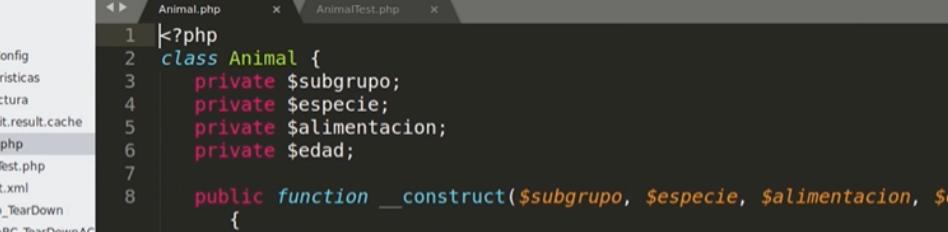
Una clase de prueba se llamará igual que la clase que se quiere probar, y añadiéndole la palabra “Test”. Además, **deberá heredar de la clase PHPUnit\Framework\TestCase**.

Los tests serán públicos y empezarán por la palabra “test*”, o se marcarán con la anotación @test.

ESTRUCTURA DE LAS PRUEBAS

- Las pruebas de la clase 'Coche' van dentro de una clase 'CocheTest'
 - Una clase de prueba debe heredar de 'PHPUnit\Framework\TestCase'
 - Los tests son públicos y empiezan por test* (o marcados con la anotación @test)
 - Dentro de una prueba se encontrarán las aserciones (métodos assert) para comprobar valores reales con valores esperados

Clase de prueba



The screenshot shows the Sublime Text interface with the following details:

- File Path:** ~/Documentos/OpenWebinars/Curso/03_Características/01_Estructura/Animal.php
- File List:** The sidebar shows a tree view of files and folders under "Curso". The "01_Estructura" folder contains "Animal.php" and "AnimalTest.php". Other folders like "02_Instal_Config" and "03_Características" also contain subfolders and files.
- Code Editor:** The main window displays the code for "Animal.php". The code defines a class "Animal" with private properties for subgrupo, especie, alimentacion, and edad. It includes a constructor to initialize these properties and a public method "sonido()" which returns the string "Guau".

```
1 <?php
2 class Animal {
3     private $subgrupo;
4     private $especie;
5     private $alimentacion;
6     private $edad;
7
8     public function __construct($subgrupo, $especie, $alimentacion, $edad)
9     {
10         $this->subgrupo = $subgrupo;
11         $this->especie = $especie;
12         $this->alimentacion = $alimentacion;
13         $this->edad = $edad;
14     }
15
16     public function sonido() {
17         return 'Guau';
18     }
}
```

Clase test

`assertEquals` comprueba que el primer parámetro es el mismo.

```

~/Documentos/OpenWebinars/Curso/03_Características/01_Estructura/AnimalTest.php (Curso) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  Curso
    02_Instal_Config
    03_Características
      01_Estructura
        .phpunit.result.cache
        Animal.php
        AnimalTest.php
        <> phpunit.xml
    02_SetUp_TearDown
    03_SetUpBC_TearDownAC
    04_Dependencias
    05_Proveedores
    04_Casos_Especiales
    05_Anotaciones
    06_Aserciones
AnimalTest.php
1  <?php
2
3  require_once('Animal.php');
4
5  class AnimalTest extends PHPUnit\Framework\TestCase {
6      public function testSonido() {
7          $animal = new Animal('', '', '', 11);
8          $this->assertEquals($animal->sonido(), 'Guau');
9      }
10 }

```

setUp() y tearDown()

`setUp` nos servirá para configurar variables para las pruebas, y se ejecuta antes de cada método de prueba.

`tearDown` nos servirá para destruir las anteriores variables y se ejecuta después de cada método de prueba.

setUp() y tearDown()

- **setUp()**
 - **Nos ayuda a configurar las pruebas y a compartir variables de configuración y objetos de prueba**
 - **Se ejecuta antes de cada método de prueba**
- **tearDown()**
 - **Se encarga de “limpiar” todo lo que se ha creado en el método setUp()**
 - **Se ejecuta después de cada método de prueba**

Ejemplo

```

~/Documentos/OpenWebinars/Curso/03_Características/02_SetUp_TearDown/ExampleSetUpTearDownTest.php (Curso) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  Curso
    02_Instal_Config
    03_Características
      01_Estructura
      02_SetUp_TearDown
        ExampleSetUpTearDown
          phpunit.xml
        03_SetUpBC_TearDownAC
        04_Dependencias
        05_Proveedores
        04_Casos_Especiales
        05_Anotaciones
        06_Aserciones
1 <?php
2
3 class ExampleSetUpTearDownTest extends PHPUnit\Framework\TestCase {
4
5     protected function setUp() {
6         $this->example = '1';
7         $this->empty_array = array();
8     }
9
10    protected function tearDown() {
11        unset($this->example);
12    }
13
14    public function testEmpty() {
15        $this->example = '2';
16        $this->assertTrue(empty($this->empty_array));
17    }
18
19    public function testEquals() {
20        $this->assertEquals($this->example, '1');
21    }
22 }

```

assertTrue-> comprueba que lo que devuelva sea true.

assertEquals->comprueba que devuelva lo que se le pasa como parámetro.

```

3 class ExampleSetUpTearDownTest extends PHPUnit\Framework\TestCase {
4
5     protected function setUp() {
6         $this->example = '1';
7         $this->empty_array = array();
8     }
9
10    protected function tearDown() {
11        unset($this->example);
12        unset($this->empty_array);
13    }
14
15    public function testEmpty() {
16        $this->example = '2';
17        $this->assertTrue(empty($this->empty_array));
18    }
19
20    public function tes
21        $this->assertEq
22    }
23 }

```

```

.
.
.
Time: 107 ms, Memory: 10.00 MB

OK (2 tests, 2 assertions)
[Finished in 0.2s]

```

setUpBeforeClass() y tearDownAfterClass()

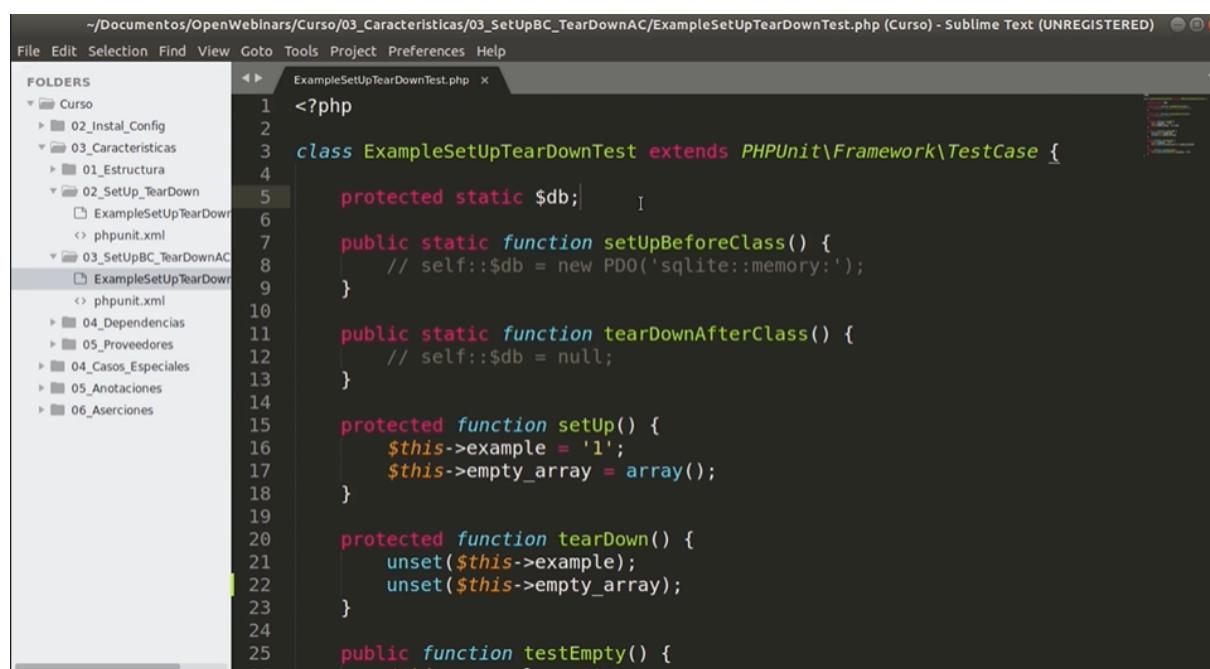
setUpBeforeClass() tiene la misma finalidad que setUp(), pero con la diferencia de que se ejecuta antes del primer test solamente.

De la misma forma, tearDownAfterClass() tiene la misma finalidad que tearDown() pero con la diferencia de que se ejecuta después del último test solamente.

setUpBeforeClass() y tearDownAfterClass()

- **Tienen la misma funcionalidad que setUp() y tearDown(), pero se ejecutan solamente una vez:**
 - **setUpBeforeClass() se ejecuta antes del primer test**
 - **tearDownAfterClass() se ejecuta después del último test**

Ejemplo



```

~/Documentos/OpenWebinars/Curso/03_Características/03_SetUpBC_TearDownAC/ExampleSetUpTearDownTest.php (Curso) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  ▾ Curso
    ▾ 02_Instal_Config
    ▾ 03_Características
      ▾ 01_Estructura
    ▾ 02_SetUp_TearDown
      ▾ ExampleSetUpTearDown
        <> phpunit.xml
    ▾ 03_SetUpBC_TearDownAC
      ▾ ExampleSetUpTearDown
        <> phpunit.xml
    ▾ 04_Dependencias
    ▾ 05_Proveedores
    ▾ 04_Casos_Especiales
    ▾ 05_Anotaciones
    ▾ 06_Asecciones
ExampleSetUpTearDownTest.php
1  <?php
2
3  class ExampleSetUpTearDownTest extends PHPUnit\Framework\TestCase {
4
5      protected static $db;
6
7      public static function setUpBeforeClass() {
8          // self::$db = new PDO('sqlite::memory:');
9      }
10
11     public static function tearDownAfterClass() {
12         // self::$db = null;
13     }
14
15     protected function setUp() {
16         $this->example = '1';
17         $this->empty_array = array();
18     }
19
20     protected function tearDown() {
21         unset($this->example);
22         unset($this->empty_array);
23     }
24
25     public function testEmpty() {
26         // Test logic here
27     }
}

```

Dependencias

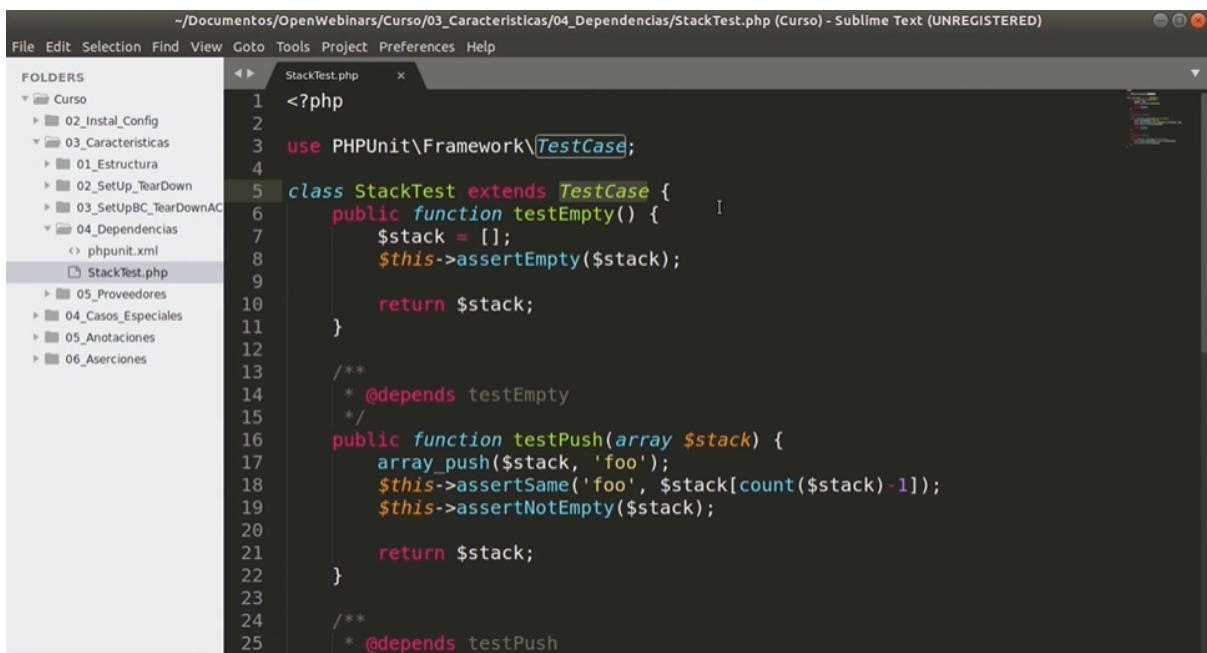
Las dependencias son una buena práctica para indicar que un método de prueba depende de otro (o de otros). **En el caso que falle un método dependiente, la prueba de la que depende no se ejecutará.**

Optimiza el tiempo de ejecución de pruebas.

DEPENDENCIAS

- **Un método de prueba puede depender de otro**
- **PHPUnit omite ejecutar métodos si la prueba de la que depende falla**
- **Mejora la detección de defectos (y optimiza el tiempo de ejecución de pruebas)**
- **Dependencia múltiple es posible**

Ejemplo @depends



```

-/Documentos/OpenWebinars/Curso/03_Características/04_Dependencias/StackTest.php (Curso) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  ▾ Curso
    ▾ 02_Instal_Config
    ▾ 03_Características
      ▾ 01_Estructura
      ▾ 02_SetUp_TearDown
      ▾ 03_SetUpBC_TearDownAC
      ▾ 04_Dependencias
        < phpunit.xml
      ▾ StackTest.php
    ▾ 05_Proveedores
    ▾ 04_Casos_Especiales
    ▾ 05_Anotaciones
    ▾ 06_Aserciones
StackTest.php
  1  <?php
  2
  3  use PHPUnit\Framework\TestCase;
  4
  5  class StackTest extends TestCase {
  6      public function testEmpty() {
  7          $stack = [];
  8          $this->assertEmpty($stack);
  9
 10         return $stack;
 11     }
 12
 13     /**
 14      * @depends testEmpty
 15     */
 16     public function testPush(array $stack) {
 17         array_push($stack, 'foo');
 18         $this->assertSame('foo', $stack[count($stack)-1]);
 19         $this->assertNotEmpty($stack);
 20
 21         return $stack;
 22     }
 23
 24     /**
 25      * @depends testPush
 26     */
 27 }

```

```
    return $stack;
}

/**
 * @depends testEmpty
 */
public function testPush(array $stack) {
    array_push($stack, 'foo');
    $this->assertSame('foo', $stack[count($stack)-1]);
    $this->assertNotEmpty($stack);

    return $stack;
}

/**
 * @depends testPush
 */
public function testPop(array $stack) {
    $this->assertSame('foo', array_pop($stack));
    $this->assertEmpty($stack);
}
```

Proveedores de datos

Otra buena práctica en los test unitarios son los proveedores de datos, los cuales nos facilitarán probar un set de datos.

Estos métodos que proveen datos serán públicos y **se ejecutarán antes de los métodos setUpBeforeClass() y setUp()**

PROVEEDORES

- **Los métodos pueden aceptar argumentos arbitrarios (additionProvider)**
- **Buena práctica**
- **Métodos “public”**
- **Se ejecuta antes de setUpBeforeClass() y de setUp()**

Ejemplo 1

```

<?php
use PHPUnit\Framework\TestCase;
class DataTest extends TestCase {
    /**
     * @dataProvider additionProvider
     */
    public function testAdd($a, $b, $expected) {
        $this->assertSame($expected, $a + $b);
    }
    public function additionProvider() {
        return [
            [0, 0, 0],
            [0, 1, 1],
            [1, 0, 1],
            [1, 1, 3]
        ];
    }
}

```

Ejemplo 2

```

class Data2Test extends TestCase {
    /**
     * @dataProvider additionProvider
     */
    public function testAdd($a, $b, $expected) {
        $this->assertSame($expected, $a + $b);
    }
    public function additionProvider() {
        return [
            'adding zeros' => [0, 0, 0],
            'zero plus one' => [0, 1, 1],
            'one plus zero' => [1, 0, 1],
            'one plus one' => [1, 2, 3]
        ];
    }
}

```

Casuísticas especiales

Probando excepciones

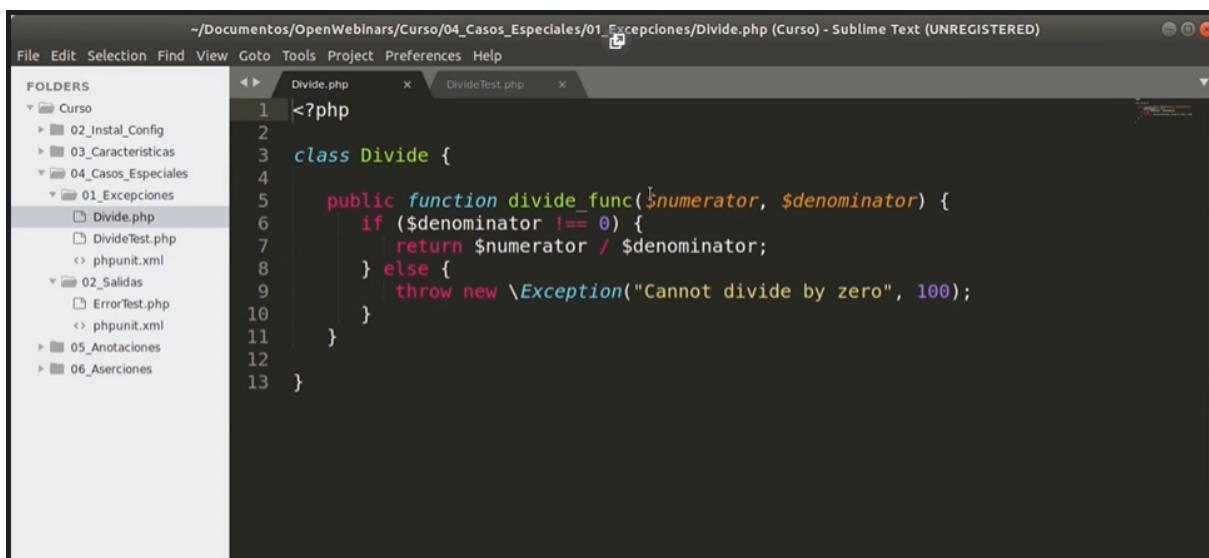
Es posible probar excepciones de PHP. El objetivo es probar si las excepciones se lanzan o no.

También podemos probar si el código de la excepción o el mensaje que devuelve es el correcto.

Pruebas Especiales - Excepciones

- La finalidad es probar (mediante el método **expectException**) si una excepción es lanzada por el código que se está probando
- Otras posibilidades:
 - **expectExceptionCode**
 - **expectExceptionMessage**
 - **expectExceptionMessageRegExp**

Ejemplo - Clase divide



The screenshot shows a Sublime Text window with two tabs: 'Divide.php' and 'DivideTest.php'. The 'Divide.php' tab contains the following PHP code:

```
<?php
class Divide {
    public function divide_func($numerator, $denominator) {
        if ($denominator !== 0) {
            return $numerator / $denominator;
        } else {
            throw new \Exception("Cannot divide by zero", 100);
        }
    }
}
```

Ejemplo -Test

```

~/Documentos/OpenWebinars/Curso/04_Casos_Especiales/01_Excepciones/DivideTest.php (Curso) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS Divide.php DivideTest.php
1  <?php
2
3  use PHPUnit\Framework\TestCase;
4  require_once('Divide.php');
5
6  class DivideTest extends TestCase {
7
8      protected function setUp() {
9          $this->divideObj = new Divide();
10     }
11
12     protected function tearDown() {
13         unset($this->divideObj);
14     }
15
16     public function test_divide() {
17         $this->assertSame(2,$this->divideObj->divide_func(4,2));
18
19         $this->expectException("Exception");
20         $this->expectExceptionCode(100);
21         $this->expectExceptionMessage("Cannot divide by zero");
22         $this->expectExceptionMessageRegExp('/divide by zero$/');
23
24         $this->divideObj->divide_func(4,0);
25         $this->assertSame(0,1); // NO EJECUTADO
26     }

```

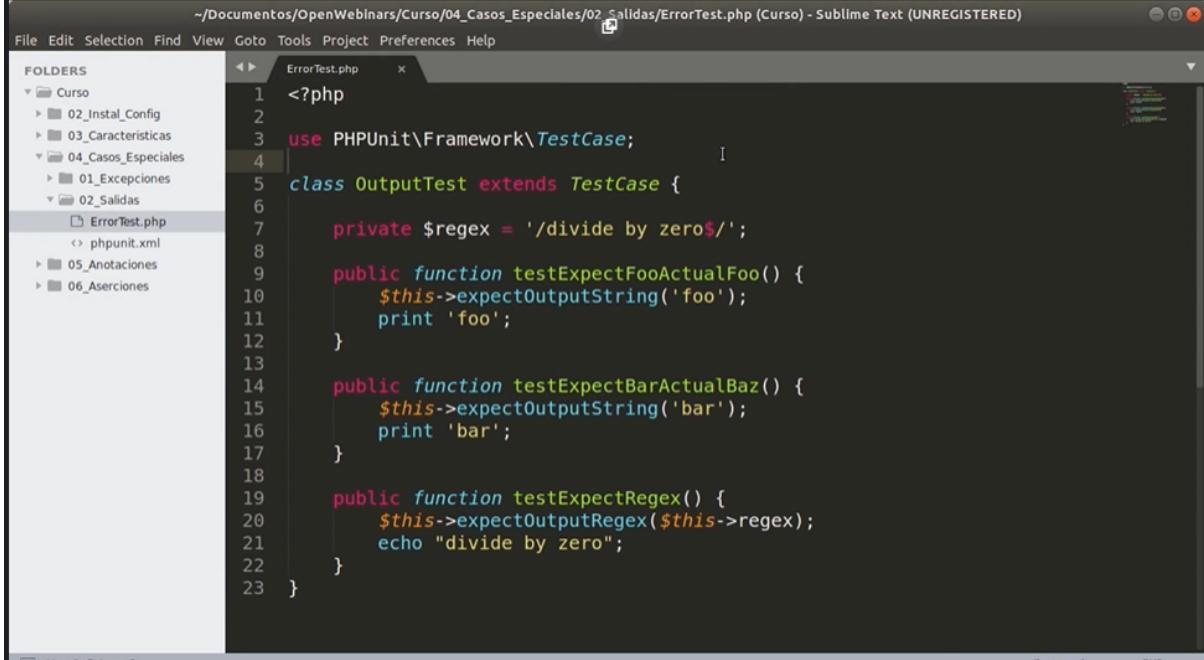
Probando salidas de PHP

También es posible probar una salida de PHP, como por ejemplo cuando una función realiza un “echo”. El objetivo es probar que la salida es la correcta.

Pruebas Especiales - Salidas en PHP

- La finalidad es probar si un método genera la salida prevista o no
- Posibilidades:
 - **expectOutputString**
 - **expectOutputRegex**
- Utilidades:
 - **setOutputCallback**
 - **getActualOutput**

Ejemplo



The screenshot shows a Sublime Text window with the following details:

- Title Bar:** ~/Documentos/OpenWebinars/Curso/04_Casos_Especiales/02_Salidas/ErrorTest.php (Curso) - Sublime Text (UNREGISTERED)
- Menu Bar:** File Edit Selection Find View Goto Tools Project Preferences Help
- Left Panel (Folders):** Shows a tree view of project folders:
 - Curso
 - 02_Instal_Config
 - 03_Caracteristicas
 - 04_Casos_Especiales
 - 01_Excepciones
 - 02_Salidas
 - ErrorTest.php
 - 05_Anotaciones
 - 06_Asecciones
- Central Panel (Code Editor):** Displays the content of `ErrorTest.php`. The code uses PHPUnit's `TestCase` and `OutputTestCase` to test output strings.
- Status Bar:** Line 4, Column 1 | Spaces: 4 | PHP

Anotaciones

¿Qué son las anotaciones?

Las anotaciones nos permitirán documentar métodos y dotar de funcionalidad de ejecución o filtrado.

Anotaciones

- **Existen múltiples anotaciones soportadas por PHPUnit que podemos utilizar para documentar nuestros métodos, e incluso proporcionar funcionalidad de filtrado, ejecución, etc.**

Anotaciones Principales

**@author, @after, @before, @covers,
@dataProvider, @depends,
@expectedException, @group, @test,
@testWith, @uses**

Ejemplo

```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  ▶ Curso
    ▶ 02_Instal_Config
    ▶ 03_Caracteristicas
    ▶ 04_Casos_Especiales
    ▶ 05_Anotaciones
      □ AnotationsTest.php
    □ Divide.php
    □ phpunit.xml
  ▶ 06_Asecciones

AnotationsTest.php x Divide.php x
1 <?php
2
3 require_once('Divide.php');
4
5 class AnotationsTest extends PHPUnit\Framework\TestCase {
6
7     protected function setUp() {
8         $this->divideObj = new Divide();
9     }
10
11     protected function tearDown() {
12         unset($this->divideObj);
13     }
14
15 /**
16 * @after
17 * @group grupo_1
18 */
19 public function test_after() {
20     $this->assertSame(1, 1);
21 }
22
23 /**
24 * @author Rubén
25 * @group grupo_1
26
27 /**
28 * @covers DivideTest::divide_func
29 * @uses DivideTest
30 * @expectedException Exception
31 */
32 public function test_divide() {
33     $this->assertSame(2,$this->divideObj->divide_func(4,2));
34
35     $this->expectException("Exception");
36     $this->expectExceptionCode(100);
37     $this->expectExceptionMessage("Cannot divide by zero");
38     $this->expectExceptionMessageRegExp('/divide by zero$/');
39
40     $this->divideObj->divide_func(4,0);
41     $this->assertSame(0,1); // NO EJECUTADO
42 }
```

```

    /**
     * @dataProvider additionProvider
     */
    public function testAdd($a, $b, $expected) {
        $this->assertSame($expected, $a + $b);
    }

    public function additionProvider() {
        return [
            [0, 0, 0],
            [0, 1, 1],
            [1, 0, 1],
            [1, 2, 3]
        ];
    }
}

```

```

/**
 * @depends testEmpty
 */
public function testPush(array $stack) {
    array_push($stack, 'foo');
    $this->assertSame('foo', $stack[count($stack)-1]);
    $this->assertNotEmpty($stack);

    return $stack;
}

```

```

/**
 * @covers DivideTest::divide_func
 * @uses DivideTest
 * @expectedException Exception
 */
public function test_divide() {
    $this->assertSame(2, $this->divideObj->divide_func(4,2));

    $this->expectException("Exception");
    $this->expectExceptionCode(100);
    $this->expectExceptionMessage("Cannot divide by zero");
    $this->expectExceptionMessageRegExp('/divide by zero$/');

    $this->divideObj->divide_func(4,0);
    $this->assertSame(0,1); // NO EJECUTADO
}

```

```
// --group grupo_1

/**
 * @author Rubén
 * @group grupo_1
 */
public function test_author() {
    $this->assertSame(1, 1);
}
```

```
/**
 * @before
 * @group grupo_1
 */
```

```
/**
 * @test
 */
public function same() {
    $this->assertEquals(1, 0);
}
```

```
/**
 * @param string $input
 * @param int $expectedLength
 *
 * @testWith ["test", 4]
 *          ["longer-string", 13]
 */
public function testStringLength(string $input, int $expectedLength)
{
    $this->assertSame($expectedLength, strlen($input));
}
```

```
/**
 * @covers Divide::divide_func
 * @uses Divide
 * @expectedException Exception
 */
public function test_divide() {
    $this->assertSame(2,$this->divideObj->divide_func(4,2));

    $this->expectException("Exception");
    $this->expectExceptionCode(100);
    $this->expectExceptionMessage("Cannot divide by zero");
    $this->expectExceptionMessageRegExp('/divide by zero$/');

    $this->divideObj->divide_func(4,0);
    $this->assertSame(0,1); // NO EJECUTADO
}
```

@author - autor de la prueba.
@after - esa prueba debe ejecutarse después de todas las pruebas unitarias.
@before - esa prueba debe ejecutarse antes de todas las pruebas unitarias.
@covers - indica que clase o función cubre esa prueba unitaria. (útil para los informes de cobertura de código)
@dataProvider - Ese método de prueba va a utilizar un proveedor de datos.
@depends - Esta función depende de otra.
@expectedException - Este método de prueba espera que lance una excepción del tipo indicado.
@group - todos los métodos de prueba pertenecen a un mismo grupo. Desde la consola de comandos utilizando “ --group nombreDelGrupo” para ejecutar los test de ese grupo únicamente.
@test - Cuando no comenzamos el nombre del test con la palabra test, es otra forma de indicar que la función es un test.
@testwith - Similar al proveedor de datos, pero no vienen en una función sino que se lo indicamos en la documentación del propio test.

@uses - Nos indica qué clase está utilizando.

Otras anotaciones

Otras Anotaciones

@afterClass, @backupGlobal, @backupStaticAttributes,
@beforeClass, @codeCoverageIgnore, @coversDefaultClass,
@coversNothing, @doesNotPerformAssertions,
@expectedExceptionCode, @expectedExceptionMessage,
@expectedExceptionMessageRegExp, @group, @large, @medium,
@preserveGlobalState, @requires, @runTestsInSeparateProcesses,
@runInSeparateProcess, @small, @ticket

<https://phpunit.readthedocs.io/es/latest/annotations.html>

Aserciones

¿Qué son las aserciones?

Todas las aserciones están en la clase `PHPUnit\Framework\Assert`, y pueden ser llamados de dos formas:

- `$this->assertTrue()`
- `self::assertTrue()`

Aserciones

- **Todas las aserciones están en la clase `PHPUnit\Framework\Assert`.**
- **`PHPUnit\Framework\TestCase` hereda de esta clase, por lo que todos los métodos de aserciones (que son estáticos) pueden ser llamados desde cualquier usando `$this->assertTrue()` o `self::assertTrue`**

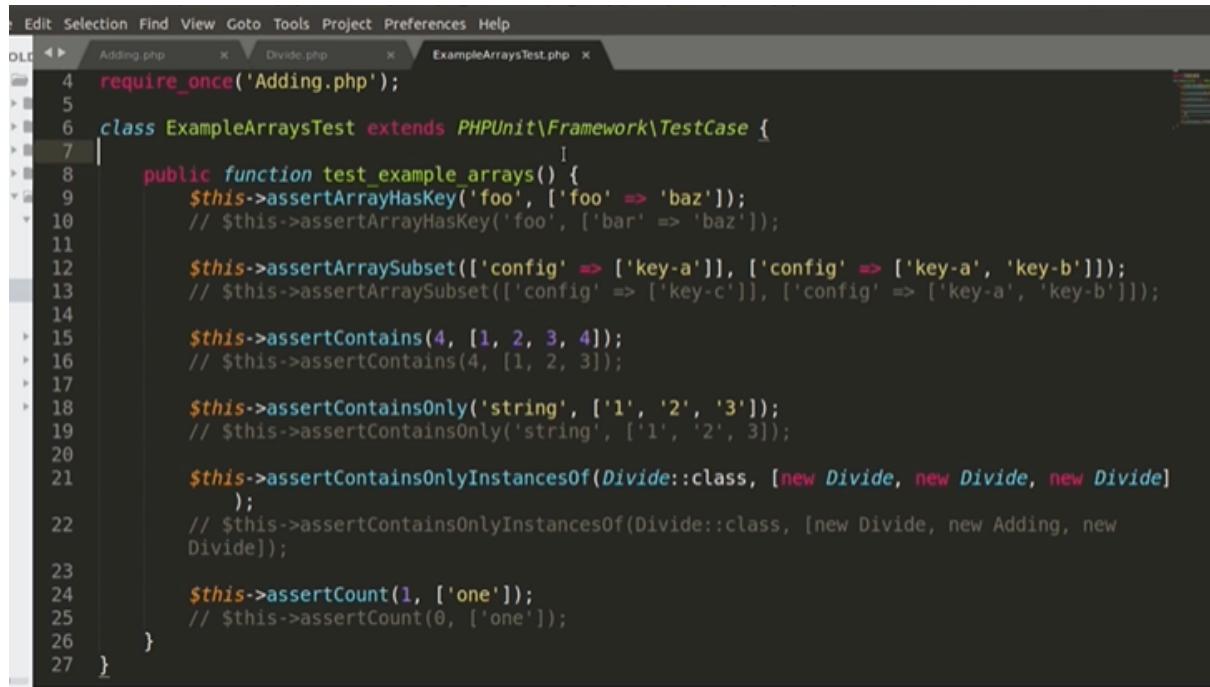
Arrays

Aserciones - Arrays

- assertArrayHasKey()**
- assertArraySubset()**
- assertContains()**
- assertContainsOnly()**
- assertContainsOnlyInstancesOf()**
- assertCount()**

Ejemplos

Con las clases Divide y Adding (clase vacía)



```

1 Edit Selection Find View Goto Tools Project Preferences Help
2 Adding.php Divide.php ExampleArraysTest.php
3
4 require_once('Adding.php');
5
6 class ExampleArraysTest extends PHPUnit\Framework\TestCase {
7     public function test_example_arrays() {
8         $this->assertArrayHasKey('foo', ['foo' => 'baz']);
9         // $this->assertArrayHasKey('foo', ['bar' => 'baz']);
10
11        $this->assertArraySubset(['config' => ['key-a']], ['config' => ['key-a', 'key-b']]);
12        // $this->assertArraySubset(['config' => ['key-c']], ['config' => ['key-a', 'key-b']]);
13
14        $this->assertContains(4, [1, 2, 3, 4]);
15        // $this->assertContains(4, [1, 2, 3]);
16
17        $this->assertContainsOnly('string', ['1', '2', '3']);
18        // $this->assertContainsOnly('string', ['1', '2', 3]);
19
20        $this->assertContainsOnlyInstancesOf(Divide::class, [new Divide, new Divide, new Divide]);
21        // $this->assertContainsOnlyInstancesOf(Divide::class, [new Divide, new Adding, new Divide]);
22
23        $this->assertCount(1, ['one']);
24        // $this->assertCount(0, ['one']);
25    }
26 }
27

```

assertArrayHasKey -> Comprueba si la clave existe en el array que se le pasa.

assertArraySubset -> comprueba que un array pertenezca a otro array.

assertContains-> si el array del segundo param, contiene el valor del primer param.

assertContainsOnly->si el segundo param(arr) contiene sólo valores del tipo del primer param.

assertContainsOnlyInstancesOf->si el arr como 2º param. sólo contiene instancias de una determinada clase.

assertCount->comprueba que un array contiene un numero determinado de elementos.

Booleanos

Aserciones - Booleanos

assertTrue()

assertFalse()

The screenshot shows a Sublime Text window with the following details:

- File Path: ~/Documentos/OpenWebinars/Curso/06_Aserciones/02_Booleanos/ExampleBooleansTest.php (Curso) - Sublime Text (UNREGISTERED)
- File Menu: File Edit Selection Find Goto Tools Project Preferences Help
- Code Editor Content:

```
<?php
class ExampleBooleansTest extends PHPUnit\Framework\TestCase {
    public function test_example_booleans() {
        $this->assertTrue(true);
        // $this->assertTrue(false);

        $this->assertFalse(false);
        // $this->assertFalse(true);
    }
}
```
- Project Explorer (Left):
 - FOLDERS
 - Curso
 - 02_Instal_Config
 - 03_Caracteristicas
 - 04_Casos_Especial
 - 05_Anotaciones
 - 06_Aserciones
 - 01_Arrays
 - 02_Booleanos
 - ExampleBooleansTest.php
 - 03_Clases
 - 04_Comparadores
 - 05_Genericas

Clases/Objetos

Aserciones - Clases / Objetos

assertInstanceOf()

assertClassHasAttribute()

assertClassHasStaticAttribute()

assertObjectHasAttribute()

The screenshot shows two code editors in Sublime Text. The top editor contains the file `ExampleObjectsClassesTest.php`, which includes code for testing the `Adding` class using PHPUnit assertions. The bottom editor contains the file `Adding.php`, which defines the `Adding` class with attributes `$att` and `$static_att`.

```

~/Documentos/OpenWebinars/Curso/06_Aserciones/03_Classes/ExampleObjectsClassesTest.php (Curso) - Sublime Text (UNREGISTERED)
File Edit Selection Find Goto Tools Project Preferences Help
FOLDERS
  <?php
  require_once('Adding.php');

  class ExampleObjectsClassesTest extends PHPUnit\Framework\TestCase {
    public function test_example_objects() {
      $this->assertInstanceOf(Exception::class, new Exception);
      // $this->assertInstanceOf(RuntimeException::class, new Exception);

      $this->assertClassHasAttribute('att', Adding::class);
      // $this->assertClassHasAttribute('test', Adding::class);

      $this->assertClassHasStaticAttribute('static_att', Adding::class);
      // $this->assertClassHasStaticAttribute('att', Adding::class);

      $this->assertObjectHasAttribute('static_att', new Adding());
      // $this->assertObjectHasAttribute('test', new Adding());
    }
  }

ExampleObjectsClassesTest.php x Adding.php x
  1  <?php
  2
  3  class Adding {
  4    public $att;
  5    public static $static_att;
  6  }

```

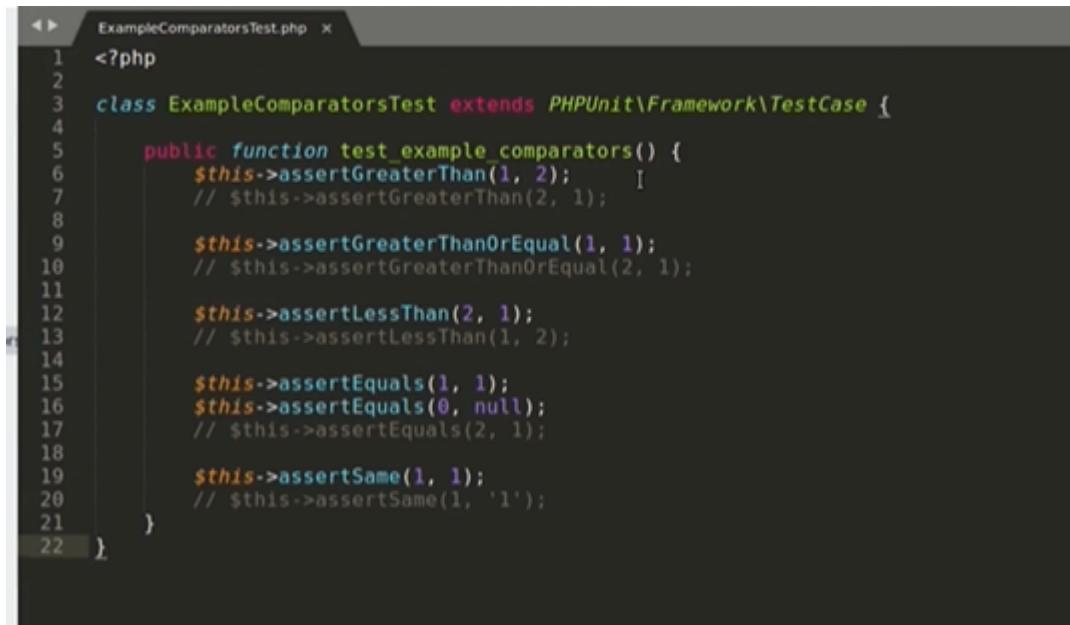
`assertInstanceOf`-> comprueba que el objeto que le pasamos es una instancia de la clase pasada como 2º param.

`assertClassHasAttribute` ->comprueba que la clase pasada como 2º att. tiene un atributo que se llama como el primer param.

`assertClassHasStaticAttribute`-> igual pero con un atributo estático.

`assertObjectHasAttribute`-> Un objeto de una clase tiene un atributo o no.

Comparadores

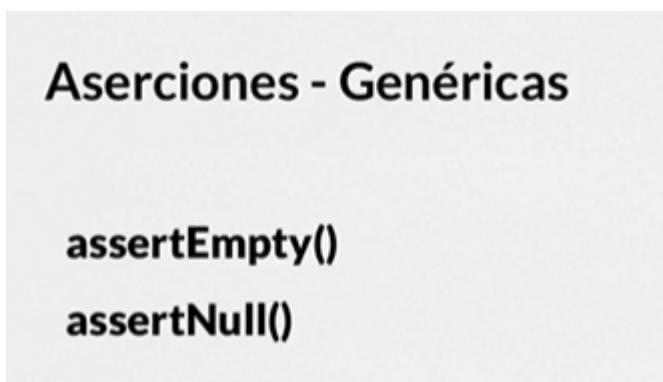
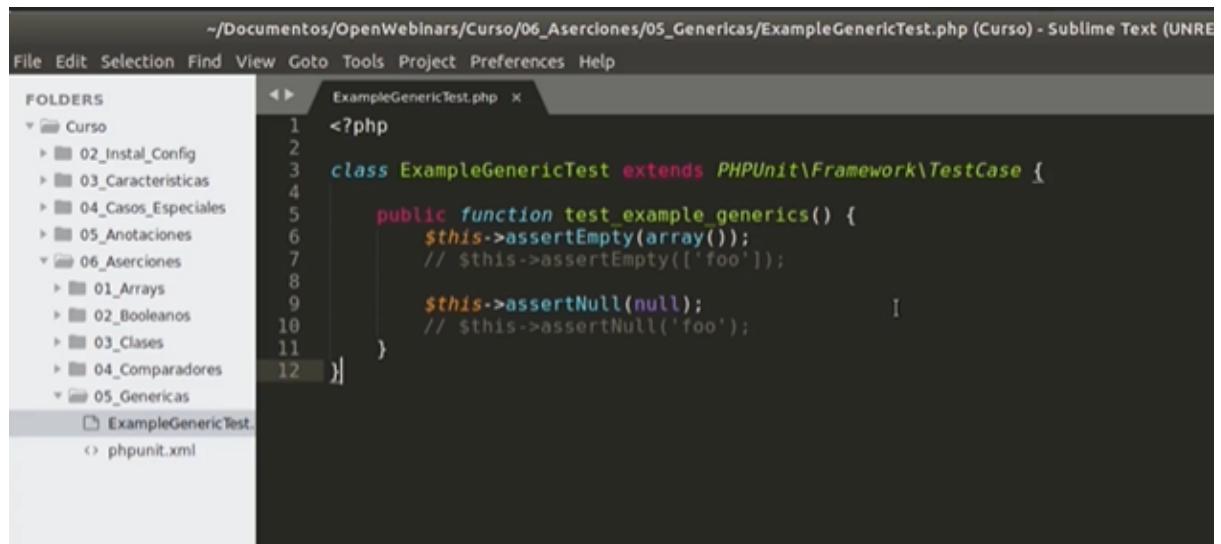


```

1 <?php
2
3 class ExampleComparatorsTest extends PHPUnit\Framework\TestCase {
4
5     public function test_example_comparators() {
6         $this->assertGreaterThan(1, 2);           I
7         // $this->assertGreaterThan(2, 1);
8
9         $this->assertGreaterThanOrEqual(1, 1);    I
10        // $this->assertGreaterThanOrEqual(2, 1);
11
12         $this->assertLessThan(2, 1);             I
13        // $this->assertLessThan(1, 2);
14
15         $this->assertEquals(1, 1);               I
16         $this->assertEquals(0, null);            I
17        // $this->assertEquals(2, 1);
18
19         $this->assertSame(1, 1);                I
20        // $this->assertSame(1, '1');
21     }
22 }

```

Genéricas

```

~/Documentos/OpenWebinars/Curso/06_Aserciones/05_Genericas/ExampleGenericTest.php (Curso) - Sublime Text (UNRE)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  ▾ Curso
    ▾ 02_Instal_Config
    ▾ 03_Características
    ▾ 04_Casos_Especiales
    ▾ 05_Anotaciones
  ▾ 06_Aserciones
    ▾ 01_Arrays
    ▾ 02_Booleanos
    ▾ 03_Clases
    ▾ 04_Comparadores
  ▾ 05_Genericas
    ▾ ExampleGenericTest
      ▷ phpunit.xml

```

```

<?php
class ExampleGenericTest extends PHPUnit\Framework\TestCase {
    public function test_example_generics() {
        $this->assertEmpty(array());
        // $this->assertEmpty(['foo']);
        $this->assertNull(null);
        // $this->assertNull('foo');
    }
}

```

Otras aserciones

Otras Aserciones

- **Ficheros**
 - **assertFileEquals()**
 - **assertFileExists()**
- **Strings**
 - **assertStringMatchesFormat()**
 - **assertStringMatchesFormatFile()**
 - **assertStringEndsWith()**
 - **assertStringEqualsFile()**
 - **assertStringStartsWith()**
 - **assertRegExp()**

Otras Aserciones

- **JSON**
 - **assertJsonFileEqualsJsonFile()**
 - **assertJsonStringEqualsJsonFile()**
 - **assertJsonStringEqualsJsonString()**
- **XML**
 - **isEqualXMLStructure()**
 - **assertXmlFileEqualsXmlFile()**
 - **assertXmlStringEqualsXmlFile()**
 - **assertXmlStringEqualsXmlString()**

<https://phpunit.readthedocs.io/es/latest/assertions.html>

Otras fuentes <https://phpunit.readthedocs.io/es/latest/writing-tests-for-phpunit.html>