PHPUnit

—

Wojtek

1.How the tasks of the pill have been organized

1. I read about PHPUnit and Composer
2. Install composer and PHPUnit
3. Create project folder
4. Create utilTest.php file
5. Check if PHPUnit is working
6. Create util.php with class and method
7. Create method in phpTest.php with assertion.
8. Tests

2.Explain the knowledge learned

1. What is PHPUnit:
   - PHPUnit is a unit testing framework for the PHP language.
2. What is Composer:
   - Composer is a package manager for PHP language.
3. What is TDD(Test Driven Development):
   TDD is a software development process.It consists of repeating several steps many times.
   - Add a test
   - Run all tests and see if the new test fails
   - Write the code
   - Run tests
   - Refactor code
   - Repeat

3. What difficulties have arisen during the pill

The most difficult part was configure framework and start test.

4. Explains the following features and functionalities of PHPUnit:

1. What is the phpunit.xml file used for?
   - phpunit.xml is file for configuration framework we can change attributes for example :

The stopOnError Attribute

Possible values: true or false (default: false)

This attribute configures whether the test suite execution should be stopped after the first test finished with status “error”.

2. What are its main parameters and what are they used for?

- Is default configuration that we can change depend on requirements.

3. How to set the color option to be enabled by default ?

- **The colors Attribute**
- Possible values: true or false (default: false)
- This attribute configures whether colors are used in PHPUnit's output.
- Setting this attribute to true is equivalent to using the --colors=auto CLI option.
- Setting this attribute to false is equivalent to using the --colors=never CLI option.

4. How to define the path in which the tests are stored (so that it is not necessary to write it by parameter when executing PHPUnit).

```
<testsuites>
        <testsuite name="unit">
            <directory>tests</directory>
        </testsuite>
```

5. What verification methods and assertions are the most used for:

- Verify numerical values

```
<?php
    use PHPUnit\Framework\TestCase;
    class NumericTest extends TestCase
    {
        public function testFailure()
        {
            $this->assertIsNumeric(null);
        }
    }
```

- Verify text strings

```
<?php
use PHPUnit\Framework\TestCase;

class StringTest extends TestCase
{
    public function testFailure()
    {
        $this->assertIsString(null);
    }
}
```

- Work with arrays

```
<?php
use PHPUnit\Framework\TestCase;
class ArrayTest extends TestCase
{
    public function testFailure()
    {
        $this->assertIsArray(null);
    }
}
```

- Work with Classes

```
<?php
use PHPUnit\Framework\TestCase;
class ClassHasAttributeTest extends TestCase
{
public function testFailure()
    {
        $this->assertClassHasAttribute('foo', stdClass::class);
    }
}
```

- Work with directories and files

```
<?php
use PHPUnit\Framework\TestCase;
class DirectoryExistsTest extends TestCase
{
    public function testFailure()
    {
        $this->assertDirectoryExists('/path/to/directory');
    }
}
```

- Work with JSON

```
<?php
use PHPUnit\Framework\TestCase;
class JsonFileEqualsJsonFileTest extends TestCase
{
    public function testFailure()
    {
        $this->assertJsonFileEqualsJsonFile(
            'path/to/fixture/file', 'path/to/actual/file');
    }
}
?>
```