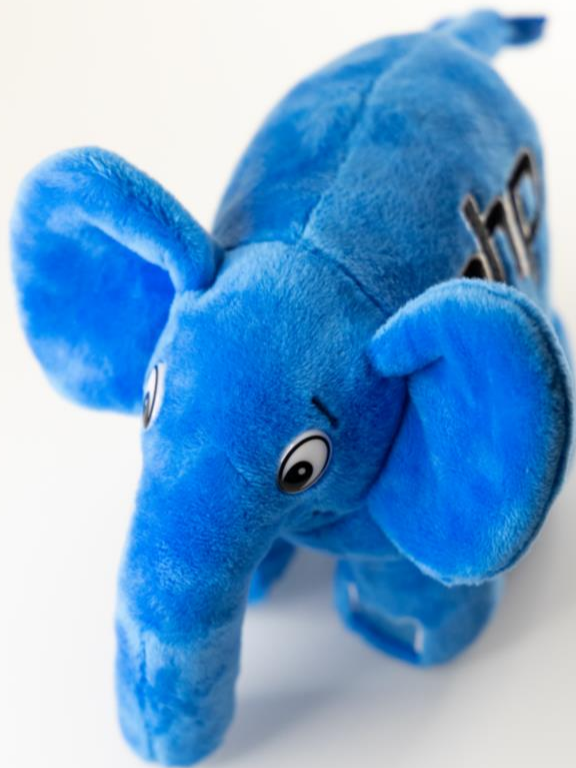


PHPUnit

Disclaimer: вы смотрите просто запись лекции,
это НЕ специально подготовленный видеокурс!



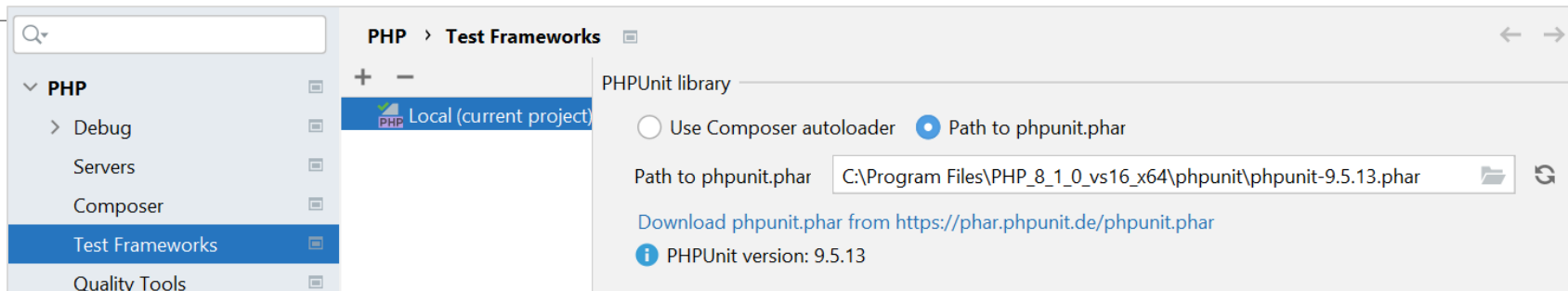
PHPUnit is a programmer-oriented testing framework for PHP. It is an instance of the xUnit architecture for unit testing frameworks.

Disclaimer! This course is not about automated testing (and unit testing in particular), so here we'll see just the main idea and usage samples.

Setup and configuration

The easiest way to start PHPUnit is to integrate it in PhpStorm, so:

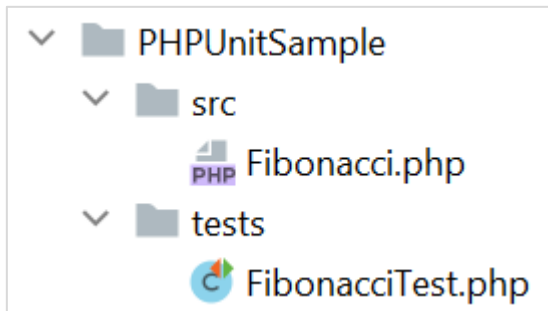
1. Go to <https://phpunit.de>.
2. Download the latest version (phar file).
3. In PhpStorm press Ctrl+Alt+S.
4. Add local PHPUnit phar:



Project structure

Usually two directories are required:

- src – for the code itself;
- tests – for tests.



And see the sample in a couple of seconds...

The code under test, just a simple class

Yes, PHPUnit is capable of much more, but the most common case is to test classes, methods, interactions and so on. So, here's our class:

```
<?php

namespace EPAM\Training\PHP\Samples;

use PHPUnit\Util\Test;

class Fibonacci
{
    // Yes, this code has errors :)

    public function getFibonacci(int $n): int
    {
        if ($n == 0) {
            return 0;
        } else
            if ($n == 1) {
                return 1;
            } else {
                return ($this->getFibonacci($n - 1) + $this->getFibonacci($n - 2));
            }
    }
}
```

The test code

Then we have to create a class with tests. For simplicity and convenience we'll use autoloading here:

```
<?php
namespace KRM\Testing\PHPUnit;

use PHPUnit\Framework\TestCase;

spl_autoload_register(function ($class) {
    $prefix = 'KRM\Testing\PHPUnit';
    $baseDir = __DIR__ . '/../src/';

    $len = strlen($prefix);
    if (strlen($prefix, $class, $len) !== 0) {
        return;
    }

    $relativeClass = substr($class, $len);
    $file = str_replace('\\', '/', $relativeClass) . '.php';

    if (file_exists($file)) {
        require $file;
    }
});

class FibonacciTest extends TestCase
{
    private static Fibonacci $fibonacci;

    public static function setUpBeforeClass(): void
    {
        self::$fibonacci = new Fibonacci();
    }

    public function positiveValuesTest(): void
    {
        $cases = [
            [0, 'zero failed'],
            [1, 'single situation failed'],
            [2, 'two situations failed'],
            [3, 'three situations failed'],
            [4, 'four situations failed'],
            [5, 'five situations failed'],
            [6, 'six situations failed'],
            [7, 'seven situations failed'],
            [8, 'eight situations failed'],
            [9, 'nine situations failed'],
            [10, 'ten situations failed'],
            [11, 'eleven situations failed'],
            [12, 'twelve situations failed'],
            [13, 'thirteen situations failed'],
            [14, 'fourteen situations failed'],
            [15, 'fifteen situations failed'],
            [16, 'sixteen situations failed'],
            [17, 'seventeen situations failed'],
            [18, 'eighteen situations failed'],
            [19, 'nineteen situations failed'],
            [20, 'twenty situations failed'],
            [21, 'twenty one situations failed'],
            [22, 'twenty two situations failed'],
            [23, 'twenty three situations failed'],
            [24, 'twenty four situations failed'],
            [25, 'twenty five situations failed'],
            [26, 'twenty six situations failed'],
            [27, 'twenty seven situations failed'],
            [28, 'twenty eight situations failed'],
            [29, 'twenty nine situations failed'],
            [30, 'thirty situations failed'],
            [31, 'thirty one situations failed'],
            [32, 'thirty two situations failed'],
            [33, 'thirty three situations failed'],
            [34, 'thirty four situations failed'],
            [35, 'thirty five situations failed'],
            [36, 'thirty six situations failed'],
            [37, 'thirty seven situations failed'],
            [38, 'thirty eight situations failed'],
            [39, 'thirty nine situations failed'],
            [40, 'forty situations failed'],
            [41, 'forty one situations failed'],
            [42, 'forty two situations failed'],
            [43, 'forty three situations failed'],
            [44, 'forty four situations failed'],
            [45, 'forty five situations failed'],
            [46, 'forty six situations failed'],
            [47, 'forty seven situations failed'],
            [48, 'forty eight situations failed'],
            [49, 'forty nine situations failed'],
            [50, 'fifty situations failed'],
            [51, 'fifty one situations failed'],
            [52, 'fifty two situations failed'],
            [53, 'fifty three situations failed'],
            [54, 'fifty four situations failed'],
            [55, 'fifty five situations failed'],
            [56, 'fifty six situations failed'],
            [57, 'fifty seven situations failed'],
            [58, 'fifty eight situations failed'],
            [59, 'fifty nine situations failed'],
            [60, 'sixty situations failed'],
            [61, 'sixty one situations failed'],
            [62, 'sixty two situations failed'],
            [63, 'sixty three situations failed'],
            [64, 'sixty four situations failed'],
            [65, 'sixty five situations failed'],
            [66, 'sixty six situations failed'],
            [67, 'sixty seven situations failed'],
            [68, 'sixty eight situations failed'],
            [69, 'sixty nine situations failed'],
            [70, 'seventy situations failed'],
            [71, 'seventy one situations failed'],
            [72, 'seventy two situations failed'],
            [73, 'seventy three situations failed'],
            [74, 'seventy four situations failed'],
            [75, 'seventy five situations failed'],
            [76, 'seventy six situations failed'],
            [77, 'seventy seven situations failed'],
            [78, 'seventy eight situations failed'],
            [79, 'seventy nine situations failed'],
            [80, 'eighty situations failed'],
            [81, 'eighty one situations failed'],
            [82, 'eighty two situations failed'],
            [83, 'eighty three situations failed'],
            [84, 'eighty four situations failed'],
            [85, 'eighty five situations failed'],
            [86, 'eighty six situations failed'],
            [87, 'eighty seven situations failed'],
            [88, 'eighty eight situations failed'],
            [89, 'eighty nine situations failed'],
            [90, 'ninety situations failed'],
            [91, 'ninety one situations failed'],
            [92, 'ninety two situations failed'],
            [93, 'ninety three situations failed'],
            [94, 'ninety four situations failed'],
            [95, 'ninety five situations failed'],
            [96, 'ninety six situations failed'],
            [97, 'ninety seven situations failed'],
            [98, 'ninety eight situations failed'],
            [99, 'ninety nine situations failed'],
            [100, 'one hundred situations failed'],
        ];

        foreach ($cases as $case) {
            $this->assertEquals($case[0], self::$fibonacci->fibonacci($case[1]));
        }
    }

    public function negativeValuesTest(): void
    {
        $cases = [
            [0, 'zero input'],
            [1, 'one input'],
            [2, 'two input'],
            [3, 'three input'],
            [4, 'four input'],
            [5, 'five input'],
            [6, 'six input'],
            [7, 'seven input'],
            [8, 'eight input'],
            [9, 'nine input'],
            [10, 'ten input'],
            [11, 'eleven input'],
            [12, 'twelve input'],
            [13, 'thirteen input'],
            [14, 'fourteen input'],
            [15, 'fifteen input'],
            [16, 'sixteen input'],
            [17, 'seventeen input'],
            [18, 'eighteen input'],
            [19, 'nineteen input'],
            [20, 'twenty input'],
            [21, 'twenty one input'],
            [22, 'twenty two input'],
            [23, 'twenty three input'],
            [24, 'twenty four input'],
            [25, 'twenty five input'],
            [26, 'twenty six input'],
            [27, 'twenty seven input'],
            [28, 'twenty eight input'],
            [29, 'twenty nine input'],
            [30, 'thirty input'],
            [31, 'thirty one input'],
            [32, 'thirty two input'],
            [33, 'thirty three input'],
            [34, 'thirty four input'],
            [35, 'thirty five input'],
            [36, 'thirty six input'],
            [37, 'thirty seven input'],
            [38, 'thirty eight input'],
            [39, 'thirty nine input'],
            [40, 'forty input'],
            [41, 'forty one input'],
            [42, 'forty two input'],
            [43, 'forty three input'],
            [44, 'forty four input'],
            [45, 'forty five input'],
            [46, 'forty six input'],
            [47, 'forty seven input'],
            [48, 'forty eight input'],
            [49, 'forty nine input'],
            [50, 'fifty input'],
            [51, 'fifty one input'],
            [52, 'fifty two input'],
            [53, 'fifty three input'],
            [54, 'fifty four input'],
            [55, 'fifty five input'],
            [56, 'fifty six input'],
            [57, 'fifty seven input'],
            [58, 'fifty eight input'],
            [59, 'fifty nine input'],
            [60, 'sixty input'],
            [61, 'sixty one input'],
            [62, 'sixty two input'],
            [63, 'sixty three input'],
            [64, 'sixty four input'],
            [65, 'sixty five input'],
            [66, 'sixty six input'],
            [67, 'sixty seven input'],
            [68, 'sixty eight input'],
            [69, 'sixty nine input'],
            [70, 'seventy input'],
            [71, 'seventy one input'],
            [72, 'seventy two input'],
            [73, 'seventy three input'],
            [74, 'seventy four input'],
            [75, 'seventy five input'],
            [76, 'seventy six input'],
            [77, 'seventy seven input'],
            [78, 'seventy eight input'],
            [79, 'seventy nine input'],
            [80, 'eighty input'],
            [81, 'eighty one input'],
            [82, 'eighty two input'],
            [83, 'eighty three input'],
            [84, 'eighty four input'],
            [85, 'eighty five input'],
            [86, 'eighty six input'],
            [87, 'eighty seven input'],
            [88, 'eighty eight input'],
            [89, 'eighty nine input'],
            [90, 'ninety input'],
            [91, 'ninety one input'],
            [92, 'ninety two input'],
            [93, 'ninety three input'],
            [94, 'ninety four input'],
            [95, 'ninety five input'],
            [96, 'ninety six input'],
            [97, 'ninety seven input'],
            [98, 'ninety eight input'],
            [99, 'ninety nine input'],
            [100, 'one hundred input'],
        ];

        foreach ($cases as $case) {
            $this->assertEquals($case[0], self::$fibonacci->fibonacci($case[1]));
        }
    }
}
```

Let's see some important parts in details...

The test code: using data providers

Data providers are methods that provide data for multiple test runs.

```
public function positiveDataProvider(): array
{
    return [
        [0, 0, "Zero failed."],
        [1, 1, "Simple situation failed."],
        [2, 1, "'Backward' part failed."],
        [3, 2, "'Backward' part failed."],
        [4, 3, "'Backward' part failed."],
        [5, 5, "'Flat' part failed."],
        [6, 8, "'Onward' part failed."],
        [-1, 1, "Simple negative value situation failed."],
        [-2, -1, "Simple negative value situation failed."],
        [-3, 2, "Normal negative value situation failed."]
    ];
}
```

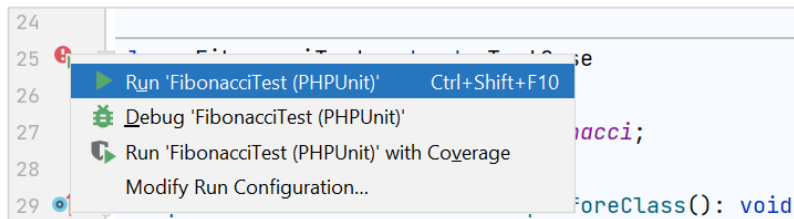
You may either “hardcode” data here, or use external sources (databases, files). Generally, each line of this array is a set of parameters for one test run.

This test will be run 10 times.

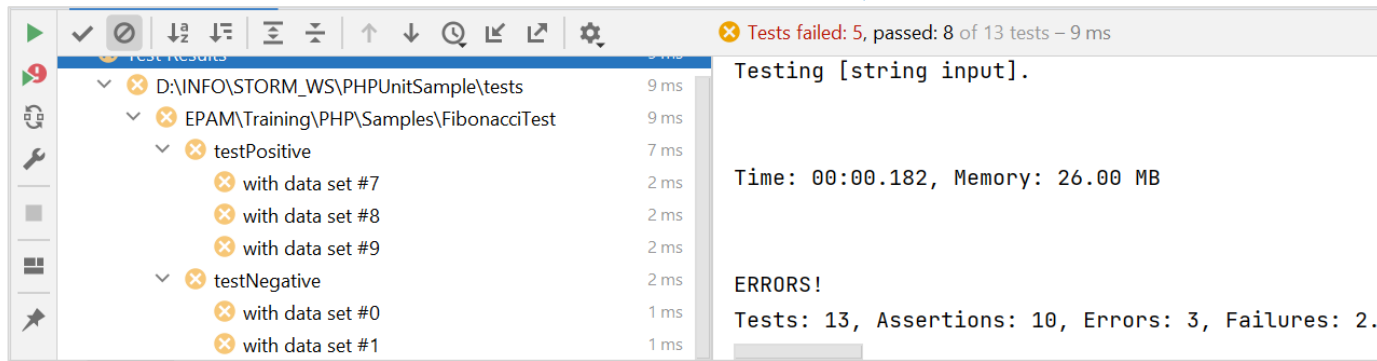
```
/**
 * @dataProvider positiveDataProvider
 */
public function testPositive(int $n, int $expected, string $message)
{
    $tmpResult = self::$fibonacci->getFibonacci($n);
    self::assertEquals($expected, $tmpResult, $message);
}
```

Run tests

Now we only have to “run” the class with tests and see the result:



If our code has no errors, all test would have passed successfully.



So, what's the idea?

Unit tests...

should always pass.

should be separated from the application code.

should be independent and simple.

may be created by developers.

may be created before code under test creation.

So, what's the idea?

The idea is to create enough different tests that will analyze your code behavior in most likely (all possible?) conditions and situations.

Don't wait till something happens to your code in production environment. Make it happen now with unit tests and be sure that your code behaves as expected.

So, what's the idea?

Unit tests...

improve architecture quality.

encourage to write simple classes / methods.

encourage changes in the code.

simplify modules integration.

help code documenting.

minimize dependencies.

are based on unified principles for many programming languages.

So, what's the idea?

Unit tests test...

methods (even void ones).

classes.

classes interaction.

complex getters/setters.

constructors.

exceptions / errors.

external dependencies.

And finally...

1. Read the documentation: <https://phpunit.readthedocs.io>
2. Read a good book: <https://www.manning.com/books/unit-testing>
3. Start using unit tests and see how your code improves.

PHPUnit

Disclaimer: вы смотрите просто запись лекции,
это НЕ специально подготовленный видеокурс!

