PHP Training

Episode 6

by Andrei Kojusko, 2018

Lesson contents

- Unit testing

PHPUnit

```
> composer require --dev phpunit/phpunit

> ./vendor/bin/phpunit --bootstrap vendor/autoload.php tests/

> ./vendor/bin/phpunit --bootstrap vendor/autoload.php --coverage-html coverage
--whitelist src/ tests/
```

- xdebug

Links

https://xdebug.org

https://phpunit.de/

https://phpunit.readthedocs.io/en/latest/index.html

Testing example - code

```
class Email {
    private $email;

    private function __construct(string $email) {
        $this->ensureIsValidEmail($email);
        $this->email = $email;
    }

    public static function fromString(string $email): self {
        return new self($email);
    }

    public function __toString(): string {
        return $this->email;
    }

    private function ensureIsValidEmail(string $email):
            void {
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            throw new InvalidArgumentException(
                sprintf(
                    '"%s" is not a valid email address',
                    $email
                )
            );
        }
    }
}
```

Testing example - test

```
use PHPUnit\Framework\TestCase;

final class EmailTest extends TestCase {
    public function testCanBeCreatedFromValidEmailAddress(): void {
        $this->assertInstanceOf(
            Email::class,
            Email::fromString('user@example.com')
        );
    }

    public function testCannotBeCreatedFromInvalidEmailAddress(): void {
        $this->expectException(InvalidArgumentException::class);

        Email::fromString('invalid');
    }
}
```

Organizing tests

```
/src
 -- Currency.php
 -- IntlFormatter.php
 -- Money.php
```

```
/tests
-- CurrencyTest.php
-- IntlFormatterTest.php
-- MoneyTest.php
```

- Use same namespaces

Test data providers

```
/**
 * @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]
    ];
}
```

Test exceptions

```php
public function testException()
{
    $this->expectException(InvalidArgumentException::class);
}
```

Test output

```
public function testExpectFooOutput()
{
    $this->expectOutputString('foo');
    print 'foo';
}
```

Test setup / teardown

```
protected static $dbh;

public static function setUpBeforeClass()
{
    self::$dbh = new PDO('sqlite::memory:');
}

public static function tearDownAfterClass()
{
    self::$dbh = null;
}
```

Incomplete / skipped tests

```
$this->markTestIncomplete(
          'This test has not been fully implemented yet.'
        );



$this->markTestSkipped(
              'This test is skipped.'
            );
```

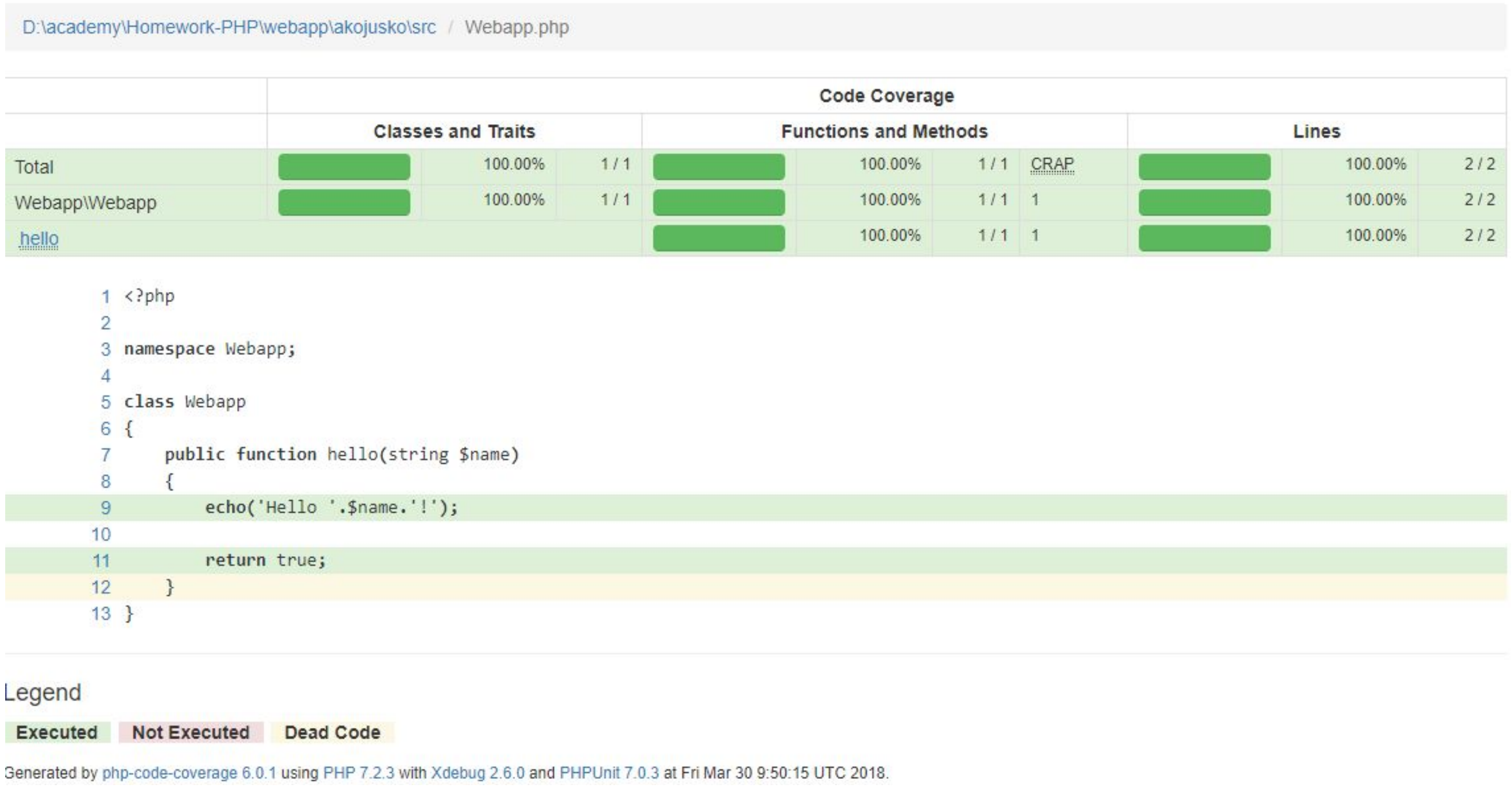
Testing doubles

```
 public function testStub()
 {
     $stub = $this->createMock(SomeClass::class);

     $stub->method('doSomething')
              ->willReturn('foo');

     $service = new Service($stub);

     $this->assertSame(
             'foo',
             $service->someActionWhichCallsSomeClass()
     );
}
```

Testing results

```
PHPUnit 7.0.0 by Sebastian Bergmann and contributors.

...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 is identical to 3.

/home/sb/DataTest.php:9

FAILURES!
Tests: 4, Assertions: 4, Failures: 1.
```

Code coverage



D:\academy\Homework-PHP\webapp\akojusko\src / Webapp.php

	Code Coverage									
	Classes and Traits			Functions and Methods				Lines		
Total	🟩	100.00%	1 / 1	🟩	100.00%	1 / 1	CRAP	🟩	100.00%	2 / 2
Webapp\Webapp	🟩	100.00%	1 / 1	🟩	100.00%	1 / 1	1	🟩	100.00%	2 / 2
hello				🟩	100.00%	1 / 1	1	🟩	100.00%	2 / 2

```php
<?php

namespace Webapp;

class Webapp
{
    public function hello(string $name)
    {
        echo('Hello '.$name.'!');

        return true;
    }
}
```

Legend

Executed Not Executed Dead Code

Generated by php-code-coverage 6.0.1 using PHP 7.2.3 with Xdebug 2.6.0 and PHPUnit 7.0.3 at Fri Mar 30 9:50:15 UTC 2018.

Homework

Implement unit tests, get coverage at least 80%, preferably 100%.