

Programowanie zespołowe

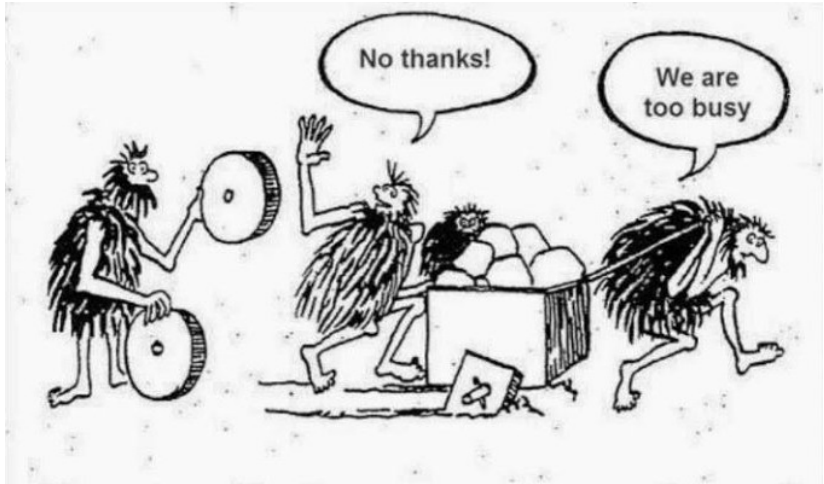
Laboratorium 3 - podstawy testów jednostkowych

mgr inż. Krzysztof Szwarc

krzysztof@szwarc.net.pl

Sosnowiec, 7 marca 2017

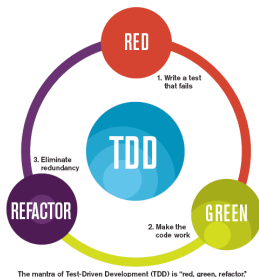
Brak czasu na testy?



<https://media.licdn.com>

TDD (Test-driven development) jest techniką wytwarzania oprogramowania polegającą na powtarzaniu trzech kroków:

- 1 Napisz automatyczny test sprawdzający dodawaną funkcjonalność (aktualnie nie powinien się on udać).
- 2 Zaimplementuj funkcjonalność (test powinien się udać).
- 3 Zrefaktoryzuj kod.



<https://s-media-cache-ak0.pinimg.com>

Testy jednostkowe

Test jednostkowy (unit test) – metoda testowania kodu za pomocą testów weryfikujących poprawność działania pojedynczych elementów (w OOP najczęściej metod). Sprawdzają one czy realizowane jest zamierzone działanie kodu. Zwykle są wykonywane wielokrotnie w trakcie cyklu powstawania oprogramowania, aby sprawdzić czy funkcjonalność działa pomimo zmian w kodzie.

Pokrycie kodu

Pokrycie kodu (code coverage) – procent linii kodu wykonywanych podczas wszystkich testów (bez klamer, deklaracji itp.). Zakłada się, że dobrą praktyką jest pokrycie 70 – 85 % kodu.

Jakie elementy pokryć testami jednostkowymi?

- Testujemy przede wszystkim logikę biznesową.
- Do testowania kontrolerów wykorzystujemy zazwyczaj testy integracyjne.
- Nie testujemy getterów i setterów.

Asercja

Asercja (assertion) – predykat (forma zdaniowa zwracająca wartość logiczną) wskazujący, że w danym miejscu kodu powinna wystąpić wartość **true**, np. wyrażenie `assertEquals(zmienna, 2)` zakłada, że zmienna ma wartość 2 (wtedy test zostanie zakończony sukcesem).

Arrange-Act-Assert

Wzorzec AAA (Arrange-Act-Assert) określa strukturę testu jednostkowego, zapewniając uporządkowanie kodu. Zakłada on następujący podział testu:

- Arrange - dane wejściowe.
- Act - operacje (działanie na metodzie/funkcji/klasie, która zostaje poddana testowi).
- Assert - asercja (sprawdzenie czy wynik jest zgodny z oczekiwanym).

Wybrane frameworki do testowania

- PHPUnit.
- JUnit.

Klasy testowe korzystające z PHPUnit muszą dziedziczyć po klasie `PHPUnit_Framework_TestCase`. Aby sprawdzić czy wartość zwracana przez metodę testowanego obiektu jest równa oczekiwanej stosujemy różne rodzaje asercji.

Przegląd najważniejszych rodzajów asercji w PHPUnit

Rodzaj	Opis	Przykład
<code>assertTrue()</code>	Sprawdzamy czy zmienna ma wartość true	<code>\$this->assertTrue(2==2);</code>
<code>assertEquals()</code>	Sprawdzamy czy oba parametry są sobie równe	<code>\$this->assertEquals(2,2);</code>
<code>assertContains()</code>	Sprawdzamy czy pierwszy parametr jest elementem tablicy (drugi parametr)	<code>\$this->assertContains(2,\$tablica);</code>
<code>assertInstanceOf()</code>	Sprawdzamy czy zmienna jest obiektem danej klasy	<code>\$this->assertInstanceOf('DateTime', \$date);</code>

Przykład - plik CalculatorClass

```
class CalculatorClass {  
    public function calculateSum($x, $y){  
        return $x+$y;  
    }  
}
```

Przykład - plik CalculatorClassTest

```
require_once 'CalculatorClass.php';

class CalculatorClassTest extends
    PHPUnit_Framework_TestCase {
    public function calculateSumTest(){
        // Arrange
        $calculatorClass = new CalculatorClass();
        // Act
        $sum = $calculatorClass->calculateSum(2,
            3);
        // Assert
        $this->assertEquals($sum, 7);
    }
}
```

Przykład w PhpStorm

The screenshot displays the PhpStorm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The project structure on the left shows a folder named 'zajecia' containing subfolders 'app' and 'External Libraries'. The 'app' folder contains files: 'CalculatorClass.php', 'CalculatorClassTest.php', 'Controller.php', 'k.php', 'config.php', 'ConfigClass.php', and 'index.php'. The main editor window shows the content of 'CalculatorClassTest.php':

```
<?php
require_once 'CalculatorClass.php';

class CalculatorClassTest extends PHPUnit_Framework_TestCase
{
    public function test()
    {
        $calculatorClass = new CalculatorClass();
        $sum = $calculatorClass->calculateSum(2, 3);
        $this->assertEquals($sum, 7);
    }
}
```

Below the editor, the 'Run' window is open, showing the execution of 'CalculatorClassTest'. The output indicates that one test failed:

```
1 test failed - 10ms
Test Results
  CalculatorClassTest
    test
      Failed asserting that 7 matches expected 5.
      Expected :5
      Actual   :7
      <Click to see difference>
```

At the bottom of the Run window, it states: 'Tests Failed: 0 passed, 1 failed (moments ago)'. The status bar at the bottom right shows '1.1 CRLF: UTF-8'.

Przykład 2 w PhpStorm

The screenshot displays the PhpStorm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The breadcrumb shows the project path: `zajecia > app > CalculatorClassTest.php`. The left sidebar shows a project tree with files like `CalculatorClass.php`, `CalculatorClassTest.php`, `Controller.php`, `k.php`, `config.php`, `ConfigClass.php`, and `index.php`.

The main editor window shows the following PHP code in `CalculatorClassTest.php`:

```
<?php
require_once 'CalculatorClass.php';

class CalculatorClassTest extends PHPUnit_Framework_TestCase
{
    public function test()
    {
        $calculatorClass = new CalculatorClass();
        $sum = $calculatorClass->calculateSum(2, 3);
        $this->assertTrue($sum);
        $this->assertEquals($sum, 7);
    }
}
```

The bottom panel shows the Run window for `CalculatorClassTest`. It indicates that 1 test failed in 110ms. The test results are as follows:

Test Name	Duration	Status	Message
test	110ms	Failed	Failed asserting that 5 is true. D:\xampp\htdocs\zajecia\app\CalculatorClassTest.php:10

Additional information in the Run window: Time: 1 second, Memory: 2.75Mb. The status bar at the bottom shows: Tests Failed: 0 passed, 1 failed (moments ago). The encoding is UTF-8.

Przykład 3 w PhpStorm

The screenshot displays the PhpStorm IDE interface. The top toolbar includes menus for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The breadcrumb path is 'Project > app > CalculatorClassTest.php'. The left sidebar shows a project tree with folders 'zajecia' and 'app', and files 'CalculatorClass.php', 'CalculatorClassTest.php', 'Controller.php', 'k.php', 'config.php', 'ConfigClass.php', 'index.php', and 'External Libraries'. The main editor window shows the code for 'CalculatorClassTest.php' with the following content:

```
<?php
require_once 'CalculatorClass.php';

class CalculatorClassTest extends PHPUnit_Framework_TestCase
{
    public function test()
    {
        $calculatorClass = new CalculatorClass();
        $sum = $calculatorClass->calculateSum(2, 3);
        $this->assertEquals($sum, 5);
    }
}
```

The 'Run' window at the bottom shows the execution of 'CalculatorClassTest'. The status bar indicates '1 test passed - 10ms'. The output text is:

```
Dr:\xampp\php\php.exe C:/Users/pc/AppData/Local/Temp/Ide-phpunit.php --no-configuration CalculatorClassTest D:\xampp\htdocs\zajecia\app\CalculatorC
Testing started at 22:52 ...
PHPUnit 3.7.21 by Sebastian Bergmann.

Time: 0 seconds, Memory: 2.00Mb

OK (1 test, 1 assertion)
```

The status bar at the bottom of the Run window shows 'Tests Passed: 1 passed (moments ago)' and '1:1 CRLF: UTF-8'.

Przegląd możliwych testów w JUnit

Oznaczenie	Opis
@Test	Oznaczenie metody jako test
@Test(expected=Exception.class)	Oznaczenie metody jako test, która powinna zwrócić wyjątek Exception
@Test(timeout=200)	Oznaczenie metody jako test, która powinna się wykonać w < 200 ms
@Before	Metoda wykonywana przed każdym testem
@After	Metoda wykonywana po każdym teście
@BeforeClass	Metoda wykonywana raz przed wszystkimi testami
@AfterClass	Metoda wykonywana raz po wszystkich testach

Przegląd najważniejszych rodzajów asercji w JUnit

Rodzaj	Opis	Przykład
<code>fail()</code>	Zawsze zróci false	<code>fail();</code>
<code>assertTrue()</code>	Sprawdzamy czy parametr ma wartość true	<code>assertTrue(zmienna);</code>
<code>assertEquals()</code>	Sprawdzamy czy oba parametry są sobie równe	<code>assertEquals(2,2);</code>
<code>assertNull()</code>	Sprawdzamy czy obiekt ma wartość null	<code>assertNull(obiekt);</code>
<code>assertSame()</code>	Sprawdzamy czy oba parametry są identyczne	<code>assertSame(2,2);</code>

Przykład w NetBeansie

The screenshot displays the NetBeans IDE interface. The main editor window shows the source code for the `ZadaniaIT` class, which is a JUnit test class. The code includes annotations for `@Before`, `@After`, and `@Test`, along with methods `setUp()`, `tearDown()`, and `testMain()`. The `testMain()` method uses `assertEquals(2, 2)` to verify a simple assertion.

```
20 public class ZadaniaIT {
32
33     @Before
34     public void setUp() {
35     }
36
37     @After
38     public void tearDown() {
39     }
40
41     /**
42     * Test of main method, of class Zadania.
43     */
44     @Test
45     public void testMain() {
46         assertEquals(2, 2);
47     }
48
49
50 }
```

The `Test Results` window at the bottom shows the execution of the test. It indicates that all tests passed (100.00%) and provides a detailed log of the test execution, including the time taken (0.045 s) and the output of the `main` method.

```
Test Results
-----
Zadania.ZadaniaIT
-----
Tests passed: 100.00-%
The test passed. (0,045 s)
main
5
5
```

- 1 Skonfiguruj dowolny Framework testowy dla wybranego IDE.
- 2 Napisz program kalkulator realizujący operacje dodawania, odejmowania, mnożenia, dzielenia, potęgowania i pierwiastkowania. Następnie pokryj wszystkie metody testami jednostkowymi.
- 3 Napisz i pokryj testami jednostkowymi następujące metody: tworzącą dwuwymiarową tablicę o losowej wielkości (z zakresu $[1,10]$) zawierającą różną liczbę wierszy i kolumn, wypełniającą ją losowymi wartościami z przedziału $[0,10]$ (za wyjątkiem elementów znajdujących się na przekątnej tablicy - mają one mieć wartość 1) oraz zwracającą największą wartość znajdującą się w tablicy.

- 4 Pobierz program ze strony <http://szwarc.net.pl/program.zip> i napisz do niego testy jednostkowe.

Dziękuję za uwagę