Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Iľkovičova 2, 842 16, Bratislava 4

Tímový projekt

MOB-UX

Metodika testovania kódu

Vedúci projektu: Ing. Eduard Kuric, PhD.
Názov tímu: MOB-UX
Členovia tímu: Bc. Tomáš Anda
Bc. Dávid Beňo
Bc. Matúš Buzássy
Bc. Martin Nagy
Bc. Patrik Pindéš
Bc. Igor Vereš
Vypracoval: Bc. Patrik Pindéš
Bc. Dávid Beňo
Kontakt: team11fiittp@gmail.com
Akademický rok: 2017/2018, zimný semester

Obsah

1. Účel dokumentu

Tento dokument sa zaoberá metodikou testovania pre jazyky JavaScript a PHP vrámci frameworku CakePHP 3. Táto verzia nezahŕňa v sebe mock-ovanie.

2. Testovanie JS

Testy sú vizualizované na stránke http://localhost:8765/js_tests () . Na tejto stránke sa dá spúšťať testy a vidieť ich výsledky. V niektorých prípadoch sa odporúča otvoriť si aj vývojársku konzolu (F12) a v rámci nej na záložke **Console** je možné vidieť chyby poprípade výpisy z testov robené pomocou metódy **console.log().**

- Template/Pages/js_tests.ctp
  - template stránky zobrazujúcej testy
  - dá sa filtrovať testy podľa modulov
  - nastavenie .js súborov potrebných na testovanie
- webroot/js/tests/tests.js
  - tu sa nacháda kód testov
  - sú tu vzorové testy s popisom

2.1. Moduly

Testy sú usporiadané podľa modulov. Modul má zahŕňať testy týkajúce sa testovania nejakého .js súboru, takže by sa mal volať podľa neho. Rozsah jedného modulu končí začiatkom (definíciou) ďalšieho.

2.2. Testy

Testy pomenujte zmysluplne, podľa toho, čo testujú. V prípade že testujeme kalkulačku, ktorá obsahuje bežné algebraické funkcie (násobenie, delenie, ...), test by sme mohli nazvať “division”.

Testy v rámci modulu môžu mať viacero assertov, na testovanie tej istej funkcie/funckionality s rôznymi vstupmi.

Test sa dá rozdeliť na 3 časti:

1. arrange
   - vykoná sa tu nastavenie testu, vytváranie inštancií tried, pomocných metód, **mock** objektov a metód.
2. act
   - spustenie testu, vykonanie logiky
3. assert

- kontrola výsledkov testu, porovnanie reálneho a očakávaného výsledku.
- pre zjednodušenie je možné vykonávať act v rámci funkcie assert

2.3. Príklad

Kalkulačka definovaná v súbore **calculator.js**

- calculator.js
  - object/funckia **Calculator**
  - metódy **div, mul**, **add**, **sub**

Pridáme *calculator.js* do *js_tests.ctp*.

```
<body>
<div id="qunit"></div>
<div id="qunit-fixture">
<!--can be used to append elements, fire events in tests, as after each test the content resets-->
</div>
<?= $this->Html->script([
        'jquery-3.2.1.min',
        'knockout-3.4.2',
        'qunit/qunit-2.4.1'])
?>
<!-- fetch js.files to test here -->
<?= $this->Html->script([
        'project_controll',
        'calculator'
    ])
?>
<?= $this->Html->script('tests/tests.js')
?>
</body>
</html>
```

Obr. č.1: Pridanie javascript súboru na testovanie

Ukážková štruktúra by mohla vyzerať takto:

```
QUnit.module( "calculator" );
QUnit.test("Addition", function(assert){
    //arrange
    var calc = new Calculator();
    //act
    var resultA = calc.add(5,5);
    var resultB = calc.add(5,-5);
    var resultC = calc.add(0,-5);
    //assert
    assert.equal( resultA, 10, " 5 + 5 = 10" );
    assert.equal( resultB, 0, " 5 - 5 = 0" );
    assert.equal( resultC, -5, " 0 - 5 = -5" );
});
QUnit.test("Division", function(assert){
    //arrange
    var calc = new Calculator();
    //act
    //assert
    assert.equal( calc.div(6,3), 2, " 6 / 3 = 2" );
    assert.equal( calc.div(6,0), 'NaN', " 6 / 0 = NaN , division by zero give NaN (not a number)" );
});
```

Obr. č.2: Štruktúra unit testov

3. Testovanie PHP

Framework CakePHP 3 integruje a používa pre testovanie PHPUnit.

3.1. Inštalácia

PHPUnit sa môže nainštalovať dvoma spôsobmi:

- **Composer**

  php composer.phar require --dev phpunit/phpunit:"^5.7|^6.0"

- **PHAR súbor**

  https://phpunit.de/#download

Pre náš projekt odporúčam použiť spôsob inštalácie číslo 1. Po nainštalovaní composer vytvorí priečinok vendor, kde sú uložené všetky dependecies k phpunit.

Nastavenie databázy:

Pre účely testovania odporúčam mať vytvorenú samostatnú testovaciu databázu. Ju následne nastaviť v súbore **config/app.php**

```
'Datasources' => [
    'test' => [
        'datasource' => 'Cake\Database\Driver\Postgres',
        'persistent' => false,
        'host' => 'dbhost',
        'username' => 'dblogin',
        'password' => 'dbpassword',
        'database' => 'test_database'
    ],
],
```

Spustenie phpunit:

vendor/bin/phpunit

3.2. Testy

Testy sú uložené v priečinsku **tests**. Pre spustenie všetkých testov, stačí spustiť skript phpunit. Ak je potrebné spustiť nejaký test samostatne, zadajte k skriptu phpunit ako argument cestu k danému testu.

Konvencie:

1. PHP skripty, ktoré obsahujú testy, sa nachádzajú v priečinku **tests/TestCase/**
2. Názvy testovacích súborov sa končia príponou Test - [nazov testu]Test.php
3. Triedy obsahujúce testy by mali dediť z Cake \TestSuite\TestCase, Cake\TestSuite\IntegrationTestCase alebo \PHPUnit\Framework\TestCase.
4. Názov triedy zodpovedá názvu súboru v ktorom je implementovaná
5. Názov metódy, ktorá obsahuje test, záčína slovom **test**

3.3. Príklad

Uvádzam príklad jednoduchého testu, ktorý zodpovedá konvenciám:

```
namespace App\Test\TestCase\View\Helper;

use App\View\Helper\ProgressHelper;
use Cake\TestSuite\TestCase;
use Cake\View\View;

class ProgressHelperTest extends TestCase
{
   public function setUp()
   {
        parent::setUp();
        $View = new View();
        $this->Progress = new ProgressHelper($View);
   }

   public function testBar()
   {
        $result = $this->Progress->bar(90);
   $this->assertContains('width: 90%', $result);
   $this->assertContains('progress-bar', $result);

   $result = $this->Progress->bar(33.3333333);
   $this->assertContains('width: 33%', $result);
   }
}
```

4. Dodatočné materiály

4.1. Arrange/act/assert

http://www.c-sharpcorner.com/UploadFile/dacca2/fundamental-of-unit-testing-understand-aaa-in-unit-testing/

4.2. Qunit testing principles

http://qunitjs.com/cookbook/

4.3. Intro into Unit testing

Ako sa má písať, refaktorovať kód aby sa dal dobre testovať.

Príklad na ukážku metódu vhodnú na unit testovanie.

http://qunitjs.com/intro/

4.4. Async testing

https://api.qunitjs.com/assert/async

4.5. Ďalšie pre JS

http://www.hoonzis.com/unit-testing-knockout-applications/

http://www.breck-mckye.com/blog/2015/02/testing-knockout-dot-js-web-applications/

http://www.breck-mckye.com/blog/2015/03/testing-knockout-custom-bindings/

4.6 Ďalšie pre PHP

Pre datailnejšie naštudovanie phpunit testov, odporúčame túto stránku:

https://book.cakephp.org/3.0/en/development/testing.html