Príloha A: Testovanie v jazyku javascript - JASMINE

1 Úvod

Čo je JASMIN ?

JASMIN je správaním riadený (behaviour-driven) rámec pre testovanie kódu v javascripte. Je vhodný pre vývoj riadený testami (test-driven development) a má syntax podobnú nástroju RSpec pre testovanie jazyka Ruby. Celý rámec JASMINE je možné stiahnuť na adrese: *https://github.com/pivotal/jasmine/releases*.

2 Základná štruktúra

Predstavme si funkciu v javascripte, ktorá vracia jednoduchý reťazec "Team Focus." :

```
function focusName() {

      return "Team Focus.";
}
```

Ak by sme pre takúto funkciu chceli otestovať jej správnosť test v JASMINE by mohol vyzerať nasledovne:

```
describe("Focus Team Name", function() {
      it("return Team Focus Name", function() {
            expect(focusName()).toEqual("Team Focus.");
      });
});
```

Časť *describe* predstavuje nejaký komponent aplikácie, v našom prípade je to funkcia *focusName()*. "Focus Team Name" je len reťazec a nepredstavuje žiadny kód.

Časť *it* predstavuje špecifikáciu, je to vlastne funkcia, ktorá hovorí čo má daný komponent robiť [1].V rámci časti *describe* môžeme pridať viacero špecifikácií. Časť *expect* predstavuje nejaké očakávanie, ktoré pracuje s hodnotou v zátvorke (v našom príklade návratová hodnota funkcie) a túto hodnotu následne nejakým spôsobom konfrontuje s funkciou nazývanou *matcher* [2] (v našom prípade *.toEqual*) .

3 Matcher

Ako už bolo spomenuté funkcia *matcher* vykonáva boolean porovnávanie medzi aktuálnou hodnotou a očakávanou hodnotou [2]. To znamená, že pred každú takúto funkciu môžeme pridať *.not* ako negáciu (napríklad *.not.toEqual()*) .

Dôležitou črtou týchto funkcií je, že existuje veľa predpísaných funkcií *matcher* ale taktiež si môžeme definovať aj vlastné.

Základnými príkladmi predpísaných funkcií *matcher* sú:

- .toEqual() - porovnanie či sú dva objekty rovnaké.
- .toContain() - porovnanie či jeden objekt obsahuje druhý objekt (reťazce, polia, ...) .
- .toBe() - porovnanie či jeden objekt je ten istý ako druhý objekt.
- .toBeTruthy() - porovnanie či je niečo *true*.
- .toBeFalsy() - porovnanie či je niečo *false*.
- .toBeDefined() - porovnanie či je objekt definovaný.
- .toBeUndefined() - porovnanie či nie je objekt definovaný.
- .toBeNull() - porovnanie či je objekt NULL.
- .toBeNaN() - porovnanie či je objekt NaN (Not a Number).
- .toBeGreaterThan() - porovnanie či je objekt väčší ako druhý objekt.
- .toBeLessThan() - porovnanie či je objekt menši ako druhý objekt.
- .toBeCloseTo() - porovnanie či je číslo blízke inému číslu na stanovený počet desatiných miest.

Vlastné funkcie *matcher* [1]:

```
beforeEach(function() {
     this.addMatchers({
          toBeLarge: function() {
               this.message = function() {
                 return "Expected " + this.actual + " to be
               large";
               };
               return this.actual > 100;
          }
     });
});
```

4 Ďalšie funkcie a črty

Rámec JASMINE má aj veľa ďalších užitočných čŕt, ktoré sa dajú efektívne využiť pri testovaní kódu vzhľadom na špecifikáciu.

4.1 Before and After

Klauzula *beforeEach* sa využíva ak chceme vykonať nejaký kód pred každou špecifikáciou (*it*). Na druhej strane klauzula *afterEach* sa využíva ak chceme nejaký kód vykonať po vykonaní každej špecifikácie.

Príklady [1]:

```
beforeEach(function() {
     employee = new Employee;
});
```

```
afterEach(function() {
     calculator.reset();
});
```

4.2 Zahniezdenie

Jednotlivé bloky *describe(...* môžeme do seba jednoducho zahniezdiť a vytvárať tak rôzne podskupiny. Príklad [2]:

```
describe("A            spec",            function()            {
     var foo;
     describe("nested inside a second describe", function() {
          .   .   .
     });
});
```

4.3 Vynechávanie špecifikácií

Ak z nejakého dôvodu nechceme niektoré špecifikácie testovať, tak ich väčšinou zakomentujeme v kóde. JASMINE však má na tieto prípady vlatné riešenie. Stačí pridať pred každú sekciu *it* alebo *describe* písmeno "x" (napr. *xit* alebo *xdescribe*). Pri testovaní sa táto sekcia automaticky preskočí.

Príkaz *return;* zase preruší testovanie a ďalej už test nebude pokračovať [1]:

```
describe("I'm only going to run SOME of these", function() {
     it("will run this spec", function() {});
     it("will run this spec", function() {});
     return; // This will stop the function
     it("will not run this spec", function() {});
     it("will not run this spec", function() {});
});
```

5 Špióni

Špióni (spies) sú špeciálne črty rámca JASMINE, ktoré dokážu sledovať volania funkcií a aj ich atribúty. Existujú iba v rámci *describe* a *it* blokov a zrušia sa pokaždej ukočenej špecifikácii.

Obsahujú špeciálne funkcie *matcher* ako napríklad:

- .toHaveBeenCalled
- .toHaveBeenCalledWith
- and.callThrough
- and.returnValue
- and.callFake
- and.throwError
- and.stub

Samotného špióna môžeme vytvoriť nasledovne [1]:

```
var distionary = new Distionary;
spyOn(dictionary, "hello");
expect(dictionary.hello).toHaveBeenCalled();
```

V tomto prípade špión nahradí funkciu *hello* triedy *Dictionary*. A následne bude volanie tejto funkcie sledované.

V rámci tohto dokumentu som spísal základné informácie o testovaní javascript jazyka prostredníctvom rámcu JASMINE. Využil som pri tom dva zdroje, kde môžete nájsť viacero ďalších informácií.

6 Zdroje

[1] http://jasmine.github.io/2.0/introduction.html
[2] http://it-ebooks.info/book/2085/