

# Razvojni okvir Jasmine

Seminarski rad u okviru kursa  
Verifikacija softvera  
Matematički fakultet

Stefan Stanišić, 1111/2019  
stefan77403@gmail.com

16. januar 2020

## Sažetak

U ovom seminarskom radu obradjen je *Jasmine* razvojni okvir(*eng. framework*) koji se koristi za testiranje *JavaScript* [3] koda. Detaljno su predstavljeni osnovni koncepti i načini korišćenja ovog razvojnog okvira. Takođe naveden je i način instalacije i pokretanje testova uz primere korišćenja razvojnog okvira kako bi čitalac rada mogao da se upozna sa praktičnom upotrebom razvojnog okvira *Jasmine*.

## Sadržaj

<b>1</b>	<b>Uvod</b>	<b>2</b>
<b>2</b>	<b>Instalacija</b>	<b>2</b>
<b>3</b>	<b>Korišćenje radnog okvira</b>	<b>3</b>
3.1	Inicijalizacija . . . . .	3
3.2	Identifikacija i skladištenje test datoteka . . . . .	3
3.3	Sintaksa za pisanje test datoteka . . . . .	4
3.4	Pokretanje test datoteka . . . . .	4
3.5	Praktični primeri . . . . .	5
3.6	Izvršavanje testova - uspeh i neuspeh . . . . .	7
<b>4</b>	<b>Zaključak</b>	<b>7</b>
	<b>Literatura</b>	<b>8</b>

# 1 Uvod

*Jasmine* [2] je razvojni okvir vođen ponašanjem koji omogućuje pisanje testova za testiranje koda napisanog u *JavaScript* programskom jeziku.

Napravljen je tako da ne zavisi od drugih *JavaScript* razvojnih okvira i za njegovo korišćenje nije potreban DOM [1]. Ima jednostavnu i veoma razumljivu sintaksu tako da je relativno jednostavan za korišćenje čak i za programere koji nemaju iskustva sa pisanjem testova. Kako je ovo razvojni okvir namenjen za testiranje koda, *Jasmine* se može koristiti kao alat komandne linije koji automatski pronalazi napisane testove, pokreće ih i izveštava o rezultatima. Postoji i drugi način na koji se može koristiti. To je u okviru veb pregledača. Potrebno je napisati odgovarajući html fajl i okviru njega *JavaScript* kod koji želimo da testiramo. Ovaj razvojni okvir je napravljen po uzoru na neke druge razvojne okvire za testiranje *JavaScript* koda kao što su *ScrewUnit* [8], *JSSpec* [5], *JSPEC* [4] i *RSpec* [7]. *Jasmine* je napravljen tako da poštuje sledeća pravila:

- Nije povezan ni sa jednim pregledačem, razvojnim okvirom ili platformom
- Ima idiomatsku i razumljivu sintaksu
- Treba da radi svuda gde se *JavaScript* može pokrenuti, uključujući pregledače interneta, servere, telefone itd.
- Ne utiče na rad *JavaScript* koda
- Lako se koristi u okviru drugih integrisanih razvojnih okruženja

Takodje jedna od prednosti ovog razvojnog okvira je to što je besplatna za korišćenje i spada u projekte otvorenog koda. Zbog svega gore navedenog *Jasmine* predstavlja jedan od boljih razvojnih okvira za testiranje *JavaScript* koda i aplikacija.

# 2 Instalacija

Kako je *JavaScript* veoma rasprostranjen i jedan od najkorišćenijih programskih jezika i usled postojanja velikog broja različitih biblioteka *JavaScript* programskog jezika, tako postoje i različite instalacije *Jasmine* razvojnog okvira. U ovom radu će biti prikazana instalacija vezana za *NodeJS* [6], kao i način korišćenja preko veb pregledača. Za prvi način najpre potrebno je sa veb sajta *NodeJS* preuzeti najnoviju stabilnu verziju *NodeJS* uz koju će biti preuzeta i najnovija verzija *npm* menadžera paketa. Nakon toga potrebno je pokrenuti instalaciju koja je ista kao i za bilo koji drugi program. Kada je instalacija završena potrebno je u komandnoj liniji ukucati neku od ove dve sledeće naredbe:

- `npm install jasmine`
- `npm install -g jasmine`

Prvom naredbom *Jasmine* razvojni okvir će biti instaliran lokalno u okviru našeg projekta, a drugom naredbom će biti instaliran globalno tako da može svuda da se koristi.

Drugi način upotrebe kojii ćemo navesti u ovom radu jeste pomoću veb pregledača. Kod ovog načina neophodan je pristup internetu. U *HTML* datoteci potrebno je linkovati *JavaScript* kod koji želimo da koristimo i onda će on automatski biti testiran. Ta *HTML* datoteka treba da sadrži sledeći kod:

```

1000 <!DOCTYPE html>
      <html>
1002 <head>
      <title>Page Title</title>
1004
      <link rel="stylesheet" type="text/css" href="https://cdnjs.
        cloudflare.com/ajax/libs/jasmine/2.7.0/jasmine.css">
1006 <script type="text/javascript" src="https://cdnjs.cloudflare.com/
        ajax/libs/jasmine/2.7.0/jasmine.js"></script>
      <script type="text/javascript" src="https://cdnjs.cloudflare.com/
        ajax/libs/jasmine/2.7.0/jasmine-html.js"></script>
1008 <script type="text/javascript" src="https://cdnjs.cloudflare.com/
        ajax/libs/jasmine/2.7.0/boot.js"></script>
1010
      <script type="text/javascript" src="https://cdnjs.cloudflare.com/
        ajax/libs/require.js/2.3.4/require.js"></script>
1012
      <!-- your js files to be tested -->
1014 <script type="text/javascript" src="./js/first.js"></script>
1016
      <!-- your test files -->
1018 <script type="text/javascript" src="./spec/firstspec.js"></script>
1020
      </head>
1022 <body>
1024
1026 </body>
      </html>

```

Listing 1: Korišćenje razvojnog okvira *Jasmine* pomoću veb pregledača.

Prednosti korišćenja *Jasmine* razvojnog okvira putem veb pregledača je ta što nije potrebna nikakva instalacija, samo je potrebno navedenu html datoteku otvoriti u okviru veb pregledača. Takođe prednost ovog načina upotrebe je to što se testovi izvršavaju odmah čim se otvori stranica u pregledaču i prikazuju se rezultati testova.

## 3 Korišćenje radnog okvira

U prethodnoj celini 2 smo predstavili način instalacije radnog okvira *Jasmine*, a u ovoj ćemo predstaviti njegovu primenu i načine korišćenja.

### 3.1 Inicijalizacija

Pre izvršavanja bilo kakvih testova potrebno je pripremiti projekat nad kojim radimo tako što se prvo u komadnoj liniji pozicioniramo u direktorijumu našeg projekta, a zatim izvršimo naredbu **jasmine init**. Ta naredba će u direktorijumu našeg projekta napraviti jedan novi direktorijum po imenu *spec* i jednu *json* datoteku. U okviru te *json* datoteke navodimo imena datoteka koje želimo da testiramo kao i jos neka podešavanja.

### 3.2 Identifikacija i skladištenje test datoteka

*Jasmine* zahteva da sve datoteke nad kojima se vrši testiranje u imenu moraju da sadrže **spec** identifikator tako da budu oblika **\*.spec.js**. Sve datoteke koje želimo da testiramo potrebno je da sačuvamo u folderu *spec* koji je napravljen prilikom inicijalizacije. 3.1

### 3.3 Sintaksa za pisanje test datoteka

Osnovna metoda za pisanje testova u *Jasmine* razvojnom okviru jeste korišćenjem funkcije **it()**. Funkcija **it()** prima dva argumenta. Prvi je string koji sadrži kratak opis tog pojedinačnog testa, a drugi argument je funkcija u okviru koje se nalazi naš test. U okviru funkcije **it()** pišemo svoje testove kao što pišemo bilo koji drugi *JavaScript* kod. Na raspolaganju su nam različite funkcije koje nam olakšavaju pisanje tih testova. To su sledeće funkcije:

- **expect()** - funkcija koja prima jedan argument koji želimo da testiramo i koja se uvek koristi uz odgovarajuće **Matcher** funkcije
- **Matcher** funkcije - **toBe()**, **toEqual()** i druge - nadovezuju se na **expect()** funkciju i u kombinaciji ove dve funkcije vršimo testiranje
- **beforeEach()** - funkcija koja se poziva pre svake **it()** funkcije u okviru **describe()** bloka
- **afterEach()** - funkcija koja se poziva posle svake **it()** funkcije u okviru **describe()** bloka
- **beforeAll()** - funkcija koja se poziva pre svih **it()** funkcija u okviru **describe()** bloka
- **afterAll()** - funkcija koja se poziva pre svih **it()** funkcija u okviru **describe()** blok

Ukoliko želimo da grupišemo više testova u jednu grupu, to postizemo korišćenjem funkcije **describe()**. I ova funkcija takodje prima dva argumenta. Prvi je string koji predstavlja kratak opis i koji će se nadovezati na opis svakog pojedinačnog testa (konkateniraće se sa prvim argumentom iz funkcije **it()**), a drugi argument je funkcija u okviru koje se nalaze svi testovi. Jedna od prednosti ovog radnog okvira je što se u okviru funkcije **describe()** mogu ugnezditi dodatne **describe()** funkcije ukoliko želimo neke specifičnije testove. U okviru **describe()** funkcije možemo praviti dodatne promenljive koje se zatim mogu koristiti u svim **it()** funkcijama.

### 3.4 Pokretanje test datoteka

Nakon što je izvršena inicijalizacija i nakon što su napisani testovi potrebno ih je izvršiti. To se radi tako što u komandnoj liniji izvršimo naredbu **jasmine** kojom će biti izvršeni svi testovi u direktorijumu našeg projekta. Ukoliko želimo da izvršavamo pojedinačne test datoteke potrebno je da se pozicioniramo u okviru *spec* direktorijuma našeg projekta i zatim izvršimo naredbu **jasmine \*.spec.js**. Neke od dodatnih opcija koje se mogu koristiti uz naredbu **jasmine**:

- **JASMINE\_CONFIG\_PATH**= ova naredba zadaje putanju (relativnu ili apsolutnu) do konfiguracionog fajla
- **-no-color** - isključuje se bojenje dobijenih rezultata testova u komandnoj liniji
- **-filter=** - izvršavaju se samo testovi sa određenim imenom
- **-stop-on-failure=[true|false]** - da li želimo da se završi sa izvršavanjem testova nakon što dodje do prvog neuspeha u testiranju

### 3.5 Praktični primeri

U ovoj sekciji će biti predstavljeni praktični primeri testova napisanih korišćenjem *Jasmine* razvojnog okvira.

**Primer 3.1** *Primer korišćenja zasebne `it()` funkcije.*

```
1000 it('hello test', function(){
      a = true;
1002   expect(a).toBe(true);
      });
```

Listing 2: Zasebna `it()` funkcija.

**Primer 3.2** *Grupisanje više `it()` funkcija pomoću funkcije `describe()`.*

```
1000 describe('hello world', function(){
      var a;
1002   it('hello test', function(){
        a = true;
1004     expect(a).toBe(true);
      });
1006   it('hello test', function(){
        a = true;
1008     expect(a).toBe(true);
1010   });
      });
```

Listing 3: Grupisanje više `it()` funkcija pomoću funkcije `describe()`.

**Primer 3.3** *Testiranje metoda klase uz pomoć korišćenja `expect()` funkcije i različitih `Matcher` funkcija.*

```
1000 class Calculator {
1002   add(a, b) {
        return a + b;
1004   }
1006   subtract(a, b) {
        return a - b;
1008   }
      }
1010 describe('calculate addition', function(){
1012   var calculate = new Calculator();
1014   it('Adding two numbers', function(){
        expect(calculate.add(1, 3)).toBe(4);
1016     expect(calculate.add(1, 3)).toEqual(4);
      });
1018   it('Subtracting two numbers', function(){
1020     expect(calculate.subtract(5, 3)).not.toBe(3);
      });
1022 });
```

Listing 4: Testiranje metoda klase

**Primer 3.4** *Testiranje definisanosti objekta uz pomoć korišćenja `expect()` funkcije i različitih `Matcher` funkcija.*

```

1000 class Calculator {
1002     add(a, b) {
1004         return a + b;
1006     }
1008     subtract(a, b) {
1010         return a - b;
1012     }
1014 }
1016 describe('calculate addition', function(){
1018     var calculate = new Calculator();
1020     it('Should be able to declare the calculator class', function()
1022     {
1024         expect(calculate).toBeDefined();
1026         expect(calculate).not.toBeUndefined();
1028         expect(calculate).not.toBeNull();
1030     })
1032 });

```

Listing 5: Testiranje definisanosti objekta

**Primer 3.5** *Korišćenje funkcija `beforeAll()`, `beforeEach()`, `afterAll()` i `afterEach()`*

```

1000 describe('Description', function(){
1002     beforeAll(function(){
1004         console.log('Running the beforeAll hook');
1006     });
1008     beforeEach(function(){
1010         console.log('Running the beforeEach hook');
1012     });
1014     afterAll(function(){
1016         console.log('Running the afterAll hook');
1018     });
1020     afterEach(function(){
1022         console.log('Running the afterEach hook');
1024     });
1026     it('test_1', function(){
1028         console.log('Running test 01');
1030     });
1032     it('test_2', function(){
1034         console.log('Running test 01');
1036     });
1038     it('test_3', function(){
1040         console.log('Running test 01');
1042     });
1044 });

```

Listing 6: Testiranje `beforeAll()`, `beforeEach()`, `afterAll()` i `afterEach()` funkcija

### 3.6 Izvršavanje testova - uspeh i neuspeh

Nakon što smo pokrenuli test možemo dobiti dva različita rezultata - da su svi testovi uspešno izvršeni ili da nisu. Ukoliko je test uspešno izvršen dobićemo poruku o tome koliko testova je izvršeno i da nijedan nije pao, kao i za koliko vremena su izvršeni ti testovi.

```
Randomized with seed 01132
Started
..

2 specs, 0 failures
Finished in 0.009 seconds
Randomized with seed 01132 (jasmine --random=true --seed=01132)
```

Listing 7: Izveštaj o uspešno izvršenom testu

Ukoliko neki test nije uspešno izvršen dobićemo poruku o tome koja sadrži opis testa koji nije prošao, razlog zbog kojeg nije prošao kao i mesto u kodu gde je došlo do greške. Takodje će biti ispisan broj testova koji su izvršeni, broj onih koji nisu prošli kao i vreme izvršavanja.

```
Randomized with seed 72356
Started
.false
F

Failures:
1) hello world hello test
   Message:
     Expected false to be true.
   Stack:
     Error: Expected false to be true.
       at <Jasmine>
       at UserContext.<anonymous> (C:\Users\PC1\Desktop\
Verifikacija-sofтвера\spec\proba.spec.js:15:13)
       at <Jasmine>

2 specs, 1 failure
Finished in 0.01 seconds
Randomized with seed 72356 (jasmine --random=true --seed=72356)
```

Listing 8: Izveštaj o neuspešno izvršenom testu

## 4 Zaključak

*Jasmine* je veoma koristan radni okvir za testiranje *JavaScript* koda. Zbog svoje jednostavne i razumljive sintakse kao i zbog principa kojima su autori vodjeni prilikom njegovog kreiranja (navedenih u poglavlju 1) danas je jedan najkorišćenijih razvojnih okvira za testiranje *JavaScript* koda. Ono što ga izdvaja u odnosu na druge je jednostavnost i to što mogu relativno lako da je koriste i oni koji nemaju velikog iskustva sa testiranjem koda. Sintaksa joj je kao i sintaksa samog *JavaScript* jezika što predstavlja veliku prednost. Takodje dobra strana je i to što pokazuje sintaksne greške u kodu. Dobro je i to što može da se koristi za pisanje kako malih i jednostavnih testova, tako i za pisanje velikih testova za testiranje cele aplikacije, a da ne utiče na performanse.

## Literatura

- [1] Dom. <https://www.javascript.com/>.
- [2] Jasmine. <https://jasmine.github.io/index.html>.
- [3] Javascript. <https://www.javascript.com/>.
- [4] Jspec. <https://javalite.io/jspec>.
- [5] Jsspec. <https://github.com/prayerslayer/js.spec>.
- [6] Nodejs. <https://nodejs.org/en/>.
- [7] Rspec. <https://rspec.info/>.
- [8] Screwunit. <https://content.pivotal.io/blog/screw-unit-a-new-js-testing-framework-version-0-1>.