

WebdriverIO - savremen alat za testiranje

Seminarski rad u okviru kursa
Verifikacija softvera
Matematički fakultet

Nikola Dimić
dimic.nikola@gmail.com

15. januar 2020

Sažetak

Testiranje veb aplikacija može biti jako komplikovano. Podešavanje mnogobrojnih alata, integracija sistema za testiranje sa postojećom aplikacijom kao i naknadno proširivanje novim alatima može predstavljati ozbiljan problem. U ovom radu biće predstavljen fleksibilan, moćan i jednostavan okvir za testiranje koji koristi Node.js i veoma se lako integriše, kako sa drugim alatima, tako i sa samom veb aplikacijom.

Sadržaj

1	Uvod	2
2	Instalacija i konfiguracija	2
2.1	Pokretač testova	2
2.2	Konfiguracija	3
3	Upotreba	4
3.1	Elementi	4
3.2	Objekat pretraživača	4
3.3	Komande	5
4	Primeri testova	6
4.1	Primer upotrebe sa Mocha okvirom	6
4.2	Primer upotrebe sa Cucumber okvirom	7
5	Zaključak	8
	Literatura	8

1 Uvod

Nalik na *Selenium WebDriver* tehnologiju, *WebdriverIO* koristi specifikacije WebDriver protokola koji je preporučen od strane *W3* konzorcijuma, odnosno predstavlja njegovu nadogradnju. [5, 7] Baziran je na paketu *webdriver* za Node.js, koji predstavlja jednostavnu implementaciju WebDriver protokola uz dodatke za upravljanje mobilnim pretraživačima koristeći *Appium*. [4]

Koristeći komande protokola, WebdriverIO kreira nove i efektivnije komande koje omogućavaju korisniku da kontroliše izvršavanje testova nad pretraživačem znatno jednostavnije ali u isto vreme da zadrži njihovu fleksibilnost. Upravo je ova osobina jedna od glavnih odlika WebdriverIO tehnologije, kako dozvoljava korisniku generisanje sopstvenih komandi vrlo intuitivnom konfiguracijom. Ova osobina značajno doprinosi tome da napisan kod bude razumljiv, pregledan i lako promenljiv. Više o ovome biće reči u sekcijama 2.2 i 3.3. [2, 8]

Konfigurisanje i integracija celokupnog sistema za testiranje prilikom razvoja veb aplikacije značajno je brže od povezivanja Selenium tehnologija sa veb aplikacijom. Samu instalaciju i konfigurisanje moguće je izvršiti u par komandi, dok je integrisanje celokupnog sistema lako usled toga da većina aplikacija bar na klijentskoj strani već koristi JavaScript, a vrlo često i Node.js. Ova oblast biće obrađena u okviru sekcije 2.

WebdriverIO dolazi sa pokretačem testova i konfiguracionim fajlom koji je vezan za njega. U kombinaciji sa velikim brojem postojećih šablona koji već imaju napisanu *POM* (eng. *Page Object Model*) strukturu, korisnik je u mogućnosti da već u ranim fazama razvoja softvera pomeri svoj akcenat rada sa konfiguracija na pisanje testova koji će biti kasnije lako nadograđivi.

Komande za Appium, odnosno za testiranje mobilnih aplikacija predstavljaju sastavni deo WebDriverIO tehnologije ali i pored toga WebdriverIO se lako povezuje sa različitim alatima za verifikaciju vrednosti, generisanje izveštaja, ili *CI* (eng. *Continuous Integration*) alatima. [2]

2 Instalacija i konfiguracija

Jednostavnost i brzina pravljenja novog projekta i stukture za testiranje softvera jedna je od odlika ove tehnologije. Da bi se WebDriverIO instalirao i pokrenuo potrebno je prethodno instalirati samo Node.js. Generisanje novog projekta moguće je koristeći sledeće komande:

```
1 mkdir webdriverio -test && cd webdriverio -test
2 npm init -y
```

Listing 1: generisanje novog WebdriverIO projekta

Ukoliko se opcija "-y" izostavi korisnik je u mogućnosti da u toku same instalacije napravi *git* repozitorijum, izmeni ime, opis i druge informacije o samom projektu. U suprotnom biće postavljene podrazumevane opcije. [2]

2.1 Pokretač testova

Jedan od glavnih problema automatskog testiranja je asinhrona priroda veb aplikacija. Događaji i odgovori od strane aplikacije su asinhroni

što se najčešće rešava obradom korišćenjem *Promise* struktura. WebDriverIO ima integrisani pokretač testova pomoću koga je pisanje asinhronih komandi moguće uraditi na sinhron način i tako korisnik ne mora brinuti o tome da li će se naredba početi da pre nego što se prethodna završi, niti pisati *Promise* naredbe kako bi obradio asinhrono događaje na ispravan način. Instalacija pokretača testova izvršava se sledećom naredbom:

```
1 npm i --save-dev @wdio/cli
```

Listing 2: instalacija pokretača testova

Pokretač testova takođe omogućava korišćenje različitih okvira za rad poput *Jasmine*, uz širok izbor alata za generisanje izveštaja kao što su *JUnit* ili *Allure*. Korišćenje pokretača testova nije obavezno u sklopu WebDriverIO specifikacije ali se preporučuje, kako omogućava razne pogodnosti kao što je globalan pristup WebDriver objektu o čemu će biti detaljnije reči u sekciji 3.2. [2, 3]

2.2 Konfiguracija

WebdriverIO odlikuje se mogućnošću da se kroz jedan konfiguracioni fajl mogu povezati različiti alati za testiranje, definisati pravila za izvršavanje, kao i akcije koje se izvršavaju pre ili posle test scenarija, kolekcije testova ili pojedinačnog testa. Konfiguracioni fajl definiše se na sledeći način:

```
1 ./node_modules/.bin/wdio config -y
```

Listing 3: generisanje konfiguracionog fajla

Povezivanje alata za verifikaciju kao što je *Chai* i testiranje kao što je *Mocha* izvršava se kroz *config* objekat u sklopu opcije *framework*. Na sličan način mogu se definisati konfiguracione promenljive kojima se može pristupiti u sklopu samih testova pa u slučaju da se testira više različitih aplikacija istovremeno ili na različitim uređajima, na osnovu tih vrednosti se pokreću testovi. [1, 2]

U narednom primeru prikazan je deo konfiguracionog fajla koji omogućava da se nakon svakog izvršenog testa čuva slika ekrana, ukoliko je došlo do greške prilikom izvršavanja testa.

```
1   afterTest: function (test) {  
2       if (test.error) {  
3           browser.takeScreenshot();  
4       }  
5   }
```

Listing 4: kod koji se izvršava nakon svakog testa

Kako bi testovi koji se izvršavaju koristili napredne opcije JavaScript jezika u okviru *ES6* standarda potrebno je podesiti i odgovarajući kompajler koji će kompilirati testove na odgovarajući način. To se može podesiti pomoću *Babel* podešavanja izvršavanjem naredbe prikazane na slici 5 a zatim konfiguracijom generisanog fajla *babel.config.js*

```

1 npm install --save-dev @babel/core @babel/cli @babel/
  preset-env @babel/register

```

Listing 5: podešavanje Babel kompajlera

Okviri i alati se povezuju na sličan način, pa je tako i slučaj sa okvirom *Cucumber* koji sa svojom specifičnom *Gherkin* sintaksom predstavlja odličan alat za razvoj vođen testovima. Više o integraciji sa ovim alatom i upotrebom biće reči u sekciji 4.2. [2]

3 Upotreba

Centralnu tačku automatizacije testova predstavlja elemente veb aplikacije i komande pomoću kojih se nad tim elementima izvršava zadata akcija. WebDriverIO podržava širok dijapazon kako komandi, tako i selektora (eng. *selectors*), pomoću kojih se elementima može pristupiti, pa će u ovom radu biti predstavljene samo neke od njih.

3.1 Elementi

Postoji veliki broj načina da se elementu pristupi pomoću selektora. U tom smislu ovaj okvir ne gubi na svojoj ekspresivnosti u odnosu na *Seleinium WebDriver* a s druge strane olakšava upotrebu selektora uvođenjem jednostavnijih komandi za to, kao što su \$ ili \$\$\$. U narednom primeru biće prikazani različiti načini korišćenja selektora

```

1 // Pristupanje elementu preko css selektora
2 const elemCss = $('h2 a')
3 // Pristupanje elementu preko teksta nekog linka
4 const elemLink = $('=Verifikacija softvera')
5 // Pristupanje elementu preko dela teksta linka
6 const elemPart = $('*=softvera')
7 // Pristupanje elementu preko XPath selektora
8 const elemXPath = $('//body/p/h1')
9 // Pristupanje elementu sa odredjenim tekstom
10 const elemText = $('h1=Dobrodosli')
11 // Pristupanje elementu sa pomocu javascript funkcije
12 const elemFunc = elem.$(function () { return this.
  nextSibling.nextSibling })

```

Listing 6: načini pristupanja elementu

Korišćenjem uniformnih oznaka za sve vrste pristupa elementu kod postaje jednostavniji za razumevanje i održavanje. [2]

3.2 Objekat pretraživača

Ukoliko se prilikom izvršavanja testova koristi WebdriverIO pokretač testova, WebDriver objektu pretraživača je moguće pristupiti preko globalne promenljive. Pozivanjem različitih komandi moguće je dobiti informacije o trenutnoj sesiji kao i druge korisne informacije o pretraživaču u datom trenutku u toku izvršavanja testa.

Konfiguracionim promenljivama o kojima je bilo reči u sekciji 2.2 moguće je pristupiti preko objekta pretraživača što je prikazano u sledećem primeru:

```

1 // Fajl: wdio.conf.js
2 exports.config = {
3   // ...
4   region: 'Serbia',
5 }
6
7 // Fajl: test.js
8 console.log(browser.config.region) // output: "Serbia"

```

Listing 7: pristup WebDriver objektu

Kako se veliki broj veb aplikacija koristi i na mobilnom telefonu, gde se elementi veb aplikacije često razlikuju posebno korisne mogu biti logičke promenljive koje nose informaciju na kom se uređaju određeni test izvršava. Tako se pozivanjem nekih od sledećih od sledećih komandi mogu u odnosu na uređaj o kom se radi primeniti različite akcije. [2]

```

1 // Ukoliko se test pokrene na Android uređaju
2 console.log(driver.isMobile) // outputs: true
3 console.log(driver.isIOS) // outputs: false
4 console.log(driver.isAndroid) // outputs: true

```

Listing 8: upotreba logičkih promenljivih pretraživača

3.3 Komande

WebDriverIO podržava veliki broj komandi za izvršavanje akcija kako nad elementima tako i nad samim pretraživačem kroz WebDriver objekat pretraživača. Kroz objekat pretraživača moguće je kontrolisati sesiju, kolačiće, prozor pretraživača, čekati na određen uslov i čak sinhrono izvršavati asinhronu pozive. Neke od komandi nad objektom pretraživača prikazane su kroz sledeći primer.

```

1 // Postavljanje kolacica
2 browser.setCookies([
3   {name: 'cookie1', value: 'verifikacija'},
4   {name: 'cookie2', value: 'softvera'}
5 ])
6
7 // Navigacija do sajta
8 browser.url("www.verifikacijasoftware.matf.bg.ac.rs/")

```

Listing 9: komande nad objektom pretraživača

Uprkos velikom broju komandi nad elementom pretraživača, najveći broj komandi izvršava se nad samim elementima veb aplikacije. Ove komande enkapsuliraju sve metode koje koristi Selenium Webdriver, pojednostavljujući ih i dodavajući nove komande koje koriste *Appium* za testiranje mobilnih uređaja. Neke od komandi prikazane su sledećim primerom:

```

1 const element = $('#idelementa')
2
3 // Navigacija do elementa
4 elem.scrollToView();
5
6 // Klik na element
7 element.click()

```

```

8
9 // Prikupljanje teksta elementa
10 let text = button.getText()
11
12 // Provera da li je element vidljiv na stranici
13 let boolean = elem.isDisplayed();

```

Listing 10: komande nad elementima

Komande koje posebno mogu biti od koristi jesu korisnički definisane komande. Definisanjem korisničkih komandi u okviru konfiguracionog fajla, mogu se konstruisati specifične akcije koje se sastoje od kombinacije postojećih akcija. Na ovaj način se kompleksije akcije koje se ponavljaju u testovima mogu grupisati pa kod postaje čitljiviji i fleksibilniji. U kodu prikazanom na slici 11 prikazan je primer definisanja korisničkih komandi. [2]

```

1 // Fajl: wdio.conf.js
2 browser.addCommand("waitAndClick", function () {
3     this.waitForDisplayed()
4     this.click()
5 }, true)
6
7 // Fajl: test.js
8 const element = $('#idelementa')
9 element.waitAndClick()

```

Listing 11: korisnički definisane komande

4 Primeri testova

WebdriverIO koristi postojeće okvire za pisanje testova, kao što su *Jasmine*, *Mocha* ili *Cucumber*. Ovo omogućava da se poznata sintaksa ovih okvira koristi u kombinaciji sa efektivnim osobinama WebdriverIO okvira.

4.1 Primer upotrebe sa Mocha okvirom

WebdriverIO po inicijalnim podešavanjima radi sa *Mocha* okvirom dok se *Jasmine* može podesiti jednostavnom promenom *framework* promenljive unutar konfiguracionog fajla i instalacijom samog okvira sledećom komandom. [1, 2]

```

1 npm install @wdio/jasmine-framework --save-dev

```

Listing 12: instalacija Jasmine okvira

Oba okvira odlikuju se *describe/it* sintaksom pa primer izgleda testa može se videti na slici 13. [1, 3]

```

1  \\ test.js
2  const assert = require('assert')
3
4  describe('Ispravno prikazan naslov', () => {
5    it('Should have the correct title', () => {
6      browser.url('http://www.verifikijasoftvera.matf
7      .bg.ac.rs/');
8      let naslovSelector = $("#naslov");
9      let naslovText = naslovSelector.getText().trim();
10     assert.strictEqual(naslovText, 'VERIFIKACIJA
11     SOFTVERA');
12   })
13 })

```

Listing 13: primer testa sa Mocha okvirom

4.2 Primer upotrebe sa Cucumber okvirom

Cucumber okvir predstavlja izuzetno popularan alat prilikom korišćenja metodologija razvoja vođenim testiranjem. Odlikuje se specifičnom *Gherkin* sintaksom koja je lako razumljiva i ljudima koji se ne bave programiranjem. Povezivanje ovog okvira sa WebdriverIO projektom izvršava se instalacijom paketa koristeći komandu na slici 4.2 a zatim menjanjem konfiguracionog fajla. Unutar njega mogu se i definisati opcije za Cucumber korišćenjem *cucumberOpts* promeljive. [2, 6]

```

1  npm install @wdio/cucumber-framework --save-dev

```

Listing 14: integracija Cucumber okvira

Primer testa napisanog u ovom okviru prikazan je na slici 15.

```

1  \\ steps.js
2  Given(/^I am on site homepage$/i, function () {
3    browser.url('http://www.verifikijasoftvera.matf.bg.
4    ac.rs');
5  });
6  When(/^I navigate to course information/i, function () {
7    let information = $("#obavestjenja");
8    information.click();
9  });
10
11  Then(/^Information no. (\d) should be (\w+)$/i, function
12    (infoNum, infoTitle) {
13    let allInfoTitles = $("#obavestjenja .obavestjenje h4"
14    );
15    let selectedInfoTitle = allInfoTitles[infoNum - 1];
16    assert.strictEqual(naslovText, infoTitle);
17  });
18
19  \\ firstInfo.feature
20  Given I am on site homepage
21  When I navigate to course information
22  Then Information no. 1 should be Teorijski seminarski rad

```

Listing 15: *Cucumber* test

5 Zaključak

Kroz ovaj rad su prikazane funkcionalnosti okvira WebdriverIO, kao i konkretni primeri integracije sa drugim alatima za testiranje. Osobine kao što su fleksibilnost, laka integracija, jednostavnost komandi i brz način za konfiguraciju celokupnog projekta, jeste ono što odvajava ovaj okvir od njemu sličnih.

Iako i dalje bez velike podrške i iscrpne dokumentacije ovaj okvir predstavlja tehnologiju koja u kombinaciji sa drugim alatima, pretenduje da postane standard u polju testiranja veb aplikacija, posebno onih sa serverskim delom napisanim u Node.js tehnologiji.

Literatura

- [1] Open JS Foundation. Mocha, 2020. on-line at: <https://mochajs.org/>.
- [2] OpenJS Foundation. WebDriverIO, 2019. on-line at: <https://webdriver.io/>.
- [3] Jasmine. Jasmine. on-line at: https://jasmine.github.io/pages/getting_started.html/.
- [4] Node.js. npmjs. on-line at: <https://www.npmjs.com/package/webdriver>.
- [5] Selenium. Selenium Webdriver, 2019. on-line at: <https://selenium.dev/projects/>.
- [6] SmartBear Software. Cucumber, 2019. on-line at: <https://cucumber.io/>.
- [7] W3C. W3C Webdriver, 2016. on-line at: <https://www.w3.org/TR/webdriver/>.
- [8] WebDriverJS. WebDriver.js, 2016. on-line at: <http://www.webdriverjs.com/webdriverio/>.