

Selenium – alat za testiranje veb aplikacija

Seminarski rad u okviru kursa
Verifikacija softvera
Matematički fakultet

Dalma Beara
beara.dalma@gmail.com

13. januar 2020

Sažetak

U ovom radu biće opisana arhitektura i način funkcionisanja alata za testiranje veb aplikacija Selenium, motivacija za nastanak i upotrebu, njegove komponente, kao i njihove najvažnije prednosti i nedostaci. Na kraju će biti data dva primera korišćenja kroz programski jezik Python.

Sadržaj

1	Uvod	2
2	Arhitektura	2
2.1	Selenium IDE	2
2.2	Selenium RC	3
2.3	Selenium Grid	4
2.4	Selenium WebDriver	4
2.4.1	POM pristup	5
3	Primeri upotrebe	5
3.1	Instalacija	5
3.2	Broj predmeta na studijama informatike na MATF-u	5
3.3	Prijavljivanje na Facebook	7
3.4	Prebacivanje kontrole u drugi prozor	7
4	Zaključak	9
	Literatura	10

1 Uvod

U današnje vreme okruženi smo milionima veb aplikacija, a nove se pojavljuju gotovo svakodnevno. Međutim, nijedan program, pa tako ni veb aplikacija ili neka njena nova funkcionalnost ne bi trebalo da budu pušteni u rad bez iscrpnog testiranja. Štaviše, testiranje se u toj fazi ne završava. Pošto su ga ranije uglavnom izvodili manuelni testeri, ono bi znalo da traje danima, nedeljama, pa čak i mesecima. Pojavom agilnih metodologija u kojima jedan korak razvoja softvera uglavnom traje od dve do četiri nedelje, manuelni testeri najčešće nisu u mogućnosti da stignu da sa velikom sigurnošću utvrde da li se aplikacija ponaša na očekivan način u svom lokalnom okruženju u van njega. To je jedan od razloga zašto su se pojavili alati za automatizaciju testova. Jedan od njih je Selenium [3].

Alat je otvorenog koda i njegova osnovna namena je da pruža interfejs za pisanje test skripti u Python-u, Javi, PHP-u, Perlu, C-u, NodeJS-u i Ruby-u, ali može se koristiti i u druge svrhe. Ovo su, uz to da podržava testiranje aplikacija u velikom proju veb pregledača (Firefox, Chrome, Opera, Safari itd.) glavni razlozi zašto se danas sve više programera i testera odlučuje da koristi Selenium.

2 Arhitektura

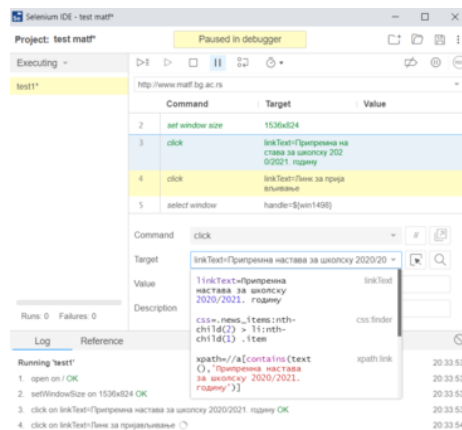
U ovom odeljku biće opisane četiri glavne komponente Seleniuma: Selenium RC, IDE, Grid i WebDriver [1].

2.1 Selenium IDE

Selenium IDE [6] E (Integrated Development Environment) je najjednostavnija od postojećih komponenti, jer za njeno korišćenje nije potrebno dubinsko poznavanje testiranja niti nekog od programskih jezika. Sve što je potrebno da bi se koristila je uključiti dodatak (engl. *add-on*) u pregledač. Do nedavno je IDE dodatak postojao samo za Firefox, a od jula 2019. godine dostupan je i za Chrome. Njegova osnovna funkcija je da snima i pamti sve komande koje izvršavamo za vreme dok je uključen. Kad izvršimo neku akciju, u IDE-u možemo pratiti njene efekte koji su podeljeni u tri kolone [4]:

- **command** – vrsta komande koju smo izvršili, npr. klik
- **target** – padajuća lista svih načina na koje se referencirani element može dohvatiti (HTML identifikator, CSS selektor, različite varijante XPath-a kojima se može dobiti itd.)
- **value** – vrednost vezana za taj element (target) u trenutku snimanja. Ovo može biti, na primer, vrednost za polje korisničkog imena u akciji logovanja na neki sistem. Ova vrednost se može ručno izmeniti pri ponovnom pokretanju snimljenih akcija.

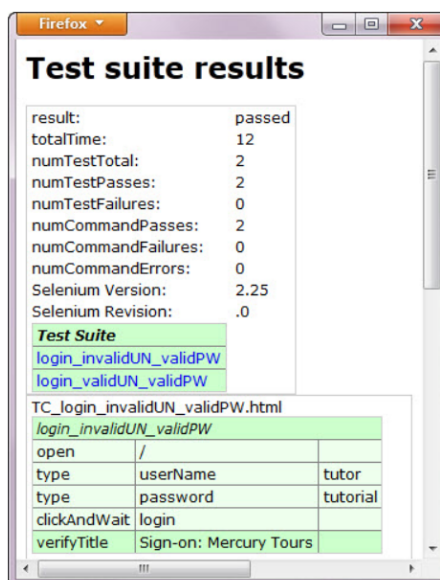
Iako je vrlo jednostavan za korišćenje i omogućava ponovno pokretanje, testiranje pretpostavki (engl. *assertions*), postavljanje tačaka prekida (engl. *breakpoints*) i kontrolisanje brzine izvršavanja testova, IDE ima ozbiljne nedostatke, od kojih su najvažniji ograničenost na dva pregledača, nemogućnost testiranja povezivanja na bazu podataka, odsustvo opcije za snimanje ekrana (engl. *screenshot*) u željenim trenucima, kao i odsustvo opcije za automatsko generisanje izveštaja o dobijenim rezultatima.



Slika 1: Praćenje izvršenih komandi u Selenium IDE-u

2.2 Selenium RC

Selenium RC (Remote Control) se ponaša kao posrednik između komandi napisanih u Seleniumu i veb pregledača. Za inicijaciju komunikacije potrebno je pokrenuti aplikaciju Selenium Remote Control Server. Ona umeće u pregledač program pod nazivom Selenium Core napisan u JavaScript-u u pregledač, i tada počinje slanje naših komandi za testiranje pregledaču. On ih izvršava kao JavaScript komande i javlja svoj odgovor Selenium RC Serveru, koji taj odgovor prima i prikazuje korisniku u formatu čitljive HTML datoteke.



Slika 2: Izveštaj koji nastaje kao rezultat rada Selenium RC-a

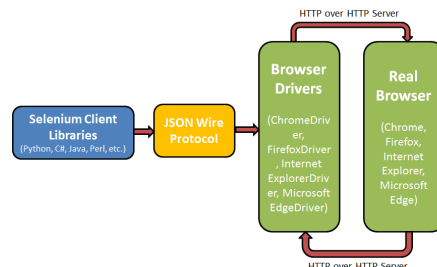
2.3 Selenium Grid

Osnovna namena Selenium Grid-a je izvršavanje testova paralelno u različitim pregledačima, operativnim sistemima i mašinama. Funkcioniše po principu razdelnika (engl. *hub*) – mašine na kojoj se pokreće test i čvorova (engl. *node*) na kojima se on izvršava. Razdelnik predstavlja centralnu tačku koja prima testove a zatim ih delegira čvorovima. Čvorovi su instance Seleniuma koje izvršavaju testove. Mašine na kojima se nalaze čvorovi ne moraju imati nikakve sličnosti. Na primer, instanca na Windows operativnom sistemu moći će da izvršava testove na Internet Explorer pregledaču, dok instanca na nekoj od distribucija Linux-a neće.

2.4 Selenium WebDriver

Selenium WebDriver se sastoji iz četiri osnovna dela:

1. klijentska biblioteka – biblioteka za interakciju sa Seleniumom na nekom konkretnom programskom jeziku
2. JSON protokol preko HTTP klijenta – JSON (JavaScript Object Notation) je jedan od uobičajenih formata za razmenu podataka na webu. Ovaj protokol je REST API ¹ koji služi za razmenu podataka sa HTTP serverom pregledača
3. drajveri pregledača – svaki pregledač sadrži svoj specijalizovani drajver (engl. *driver*) koji služi da sa njim komunicira bez ulaženja u njegovu unutrašnju logiku. Sve zahteve koje dobije prosleđuje svom pregledaču, a zatim od njega prima odgovor
4. pregledač



Slika 3: Arhitektura Selenium WebDriver-a

Proces funkcionise na sledeći način: programer piše testove u željenom programskom jeziku koristeći odgovarajuću Selenium biblioteku. Kada se program pokrene, klijentska biblioteka uspostavlja komunikaciju sa Selenium API-jem, koji pomoću JSON protokola šalje napisane komande drajveru pregledača. Drajver korišćenjem HTTP servera šalje pregledaču zahtev, koji ga potom obavlja i šalje odgovor našoj test skripti.

Testiranje korišćenjem WebDriver komponente je brže nego uz pomoć RC-a jer koristi unutrašnju logiku pregledača umesto posrednika, ali on ne generiše nikakav izveštaj obavljenih testova. Takođe, WebDriver ima podršku za HtmlUnit, pregledač bez grafičkog korisničkog interfejsa koji obezbeđuje API za obavljanje svih aktivnosti koje podržavaju

¹REST (engl. *Representational State Transfer*) – stil arhitekture softvera koji definiše skup ograničenja za kreiranje veb servisa

uobičajeni pregledači (dohvatanje stranica, popunjavanje formi, praćenje linkova itd.).

2.4.1 POM pristup

POM (Page Object Model) [2] je obrazac za projektovanje (engl. *design pattern*) koji nalaže da se svaka veb stranica kojoj se pristupa iz aplikacije modeluje odgovarajućom klasom. U toj klasi se definišu elementi veb stranice kao i metode kojima se na njoj vrši neka akcija, obrađuje čekanje i slično. Glavni razlog za uvođenje je to što vremenom količina koda za testiranje postaje prevelika i teška za održavanje. Ukoliko se desi da se recimo promeni identifikator elementa kome pristupamo, tu promenu moramo ispratiti na svakom mestu u kodu gde ga referenciramo, dok je kod POM pristupa potrebno tu promenu izvršiti samo jednom, u klasi gde je registrovan željeni element. Ono što je još jako važno je da je ovakav kod nezavisan od test slučajeva, što znači da se može koristiti za testiranje različitih vrsta. Dakle, možemo koristiti isti kod za primerice funkcionalno i testiranje prihvatljivosti. Takođe, kod postaje čitljiviji i razumljiviji.

3 Primeri upotrebe

U ovom odeljku biće opisan postupak instalacije Seleniuma za Python, a potom data dva primera upotrebe Selenium WebDriver-a u programskom jeziku Python [5].

3.1 Instalacija

Instalacija se vrši zadavanjem sledeće komande kroz komandnu liniju: `pip install -U selenium`, dok se odgovarajuća distribucija Chrome WebDriver-a može preuzeti sa adrese: <https://sites.google.com/a/chromium.org/chromedriver/downloads>.

3.2 Broj predmeta na studijama informatike na MATF-u

Recimo da znamo da student osnovnih studija informatike na MATF-u mora da položi ukupno 41 predmet kako bi diplomirao i želimo da se uverimo da li je na sajtu fakulteta ta informacija ispravno navedena. U nastavku se može videti kod koji pokreće instancu Google Chrome pregledača, odlazi na stranicu predmeta na osnovnim studijama informatike na MATF-u i dohvata informacije o predmetima (naziv, fond časova i ESPB bodove), i na kraju proverava da li je broj predmeta ispravno naveden.

Uključujemo potrebnu biblioteku, Selenium WebDriver za Chrome:

```
1000 from selenium.webdriver import Chrome
```

Listing 1: potrebne biblioteke

Zadajemo putanju do mesta na sistemu gde smo sačuvali WebDriver i pravimo instancu Chrome pregledača pomoću njega:

```

1000 webdriver = "path/to/driver"
      driver = Chrome(webdriver)

```

Listing 2: instanciranje pregledača

Specifikujemo URL željene stranice i dohvatamo njen sadržaj:

```

1000 url = "http://www.matf.bg.ac.rs/m/180/osnovne-informatika"
      /"
      driver.get(url)

```

Listing 3: dohvaćanje sadržaja stranice

Sada treba da dohvatimo tabele predmeta za četiri godine studija. To ćemo uraditi pomoću css selektora:

```

1000 tables = driver.find_elements_by_css_selector("table"
      [:4]
      subject_data = []

```

Listing 4: dohvaćanje tabele

Zatim uzimamo samo tekst iz redova tabele (pomoću css selektora tr), i dodajemo ga našoj listi predmeta ako ustanovimo da je red s podacima o predmetu:

```

1000 for table in tables:
      table_rows = table.find_elements_by_css_selector("tr"
1002 )
      for row in table_rows:
          row_text = row.text
1004         if any(['Семестар' in row_text,
                  'Предмет' in row_text,
1006                 'Укупно' in row_text
                  ]):
1008             continue
          else:
1010             subject_data.append(row_text)

```

Listing 5: dohvaćanje i parsiranje podataka iz redova tabele

Pišemo test da je broj predmeta onoliki koliki smo pretpostavili da jeste. Ukoliko nije, na standardni izlaz će biti ispisana poruka o grešci:

```

1000 assert len(subject_data) == 41, "Unexpected number of
      subjects!"

```

Listing 6: pretpostavka da je broj predmeta ispravno naveden

Na kraju zatvaramo otvoreni drajver:

```
1000 driver.close()
```

Listing 7: zatvaranje drajvera

3.3 Prijavljivanje na Facebook

Naredni primer ilustruje korišćenje Selenium-a u svrhu provere ispravnosti prijavljivanja (logovanja) na Facebook. Početni deo (uključivanje biblioteke, instanciranje drajvera i pregledača na stranicu na kojoj izvodimo prijavljivanje, <http://facebook.com/login>), u ovom slučaju preskačemo jer je isto kao i u prethodnom primeru.

Dohvatamo polja za e-mail i šifru korisnika po njihovim HTML identifikatorima i popunjavamo ih svojim podacima.

```
1000 email = driver.find_element_by_id("email")
      email.send_keys("test.selenium.matf@gmail.com")
1002
      password = driver.find_element_by_id("pass")
1004 password.send_keys("testpwd")
```

Listing 8: dohvatanje i popunjavanje polja za e-mail i šifru korisnika

Na isti način dohvatamo i dugme za slanje forme za prijavljivanje, a potom zadajemo komandu za klik na to dugme.

```
1000 login_btn = driver.find_element_by_id("loginbutton")
      login_btn.click()
```

Listing 9: dohvatanje dugmeta za potvrdu prijavljivanja i klik na njega
Zatvaranje resursa preskačemo.

3.4 Prebacivanje kontrole u drugi prozor

Kada pristupamo nekom veb sajtu, često dolazimo u situaciju da se klikom na neko dugme ili vezu (engl. *link*) tražena stranica otvori u potpuno novom prozoru. Kada se ovo desi, Selenium kontrolu zadržava na prozoru pozivaocu, a mi ćemo neretko želeti da se ona preusmeri na novootvoreni prozor. U nastavku je kod koji to radi za nas. ovog puta poslužićemo se i bibliotekom *unittest*. Takođe, ovaj put ćemo se poslužiti objektno-orijentisanim pristupom.

Kao i do sada, importujemo potrebne biblioteke. Za *unittest* nije potrebna nikava dodatna instalacija.

```
1000 from selenium.webdriver import Chrome
      import unittest
```

Listing 10: uključivanje potrebnih resursa

Klasu `WindowTest` definišemo tako da nasledi klasu `TestCase` iz biblioteke `unittest`. Potom inicijalizujemo drajver za Chrome na standardni način:

```
1000 class WindowTest(unittest.TestCase):
      def setUp(self):
1002         webdriver = "path/to/driver"
            self.driver = Chrome(webdriver)
```

Listing 11: definicija klase i inicijalizacija drajvera

Započnimo definiciju funkcije `test_windows`. Odlazimo na veb stranicu koja nam je zgodna za testiranje ove funkcionalnosti jer u sebi sadrži vezu čijim se praćenjem otvara drugi prozor. Kad je stranica dohvaćena, u promenljivu `window_before` upisujemo sadržaj nultog, odnosno trenutnog prozora u kom se nalazimo. Njegov naziv pamtimo u promenljivoj `parent_window`.

```
1000     def test_windows(self):
            driver = self.driver
1002         driver.get("http://www.quackit.com/html/codes/
html_popup_window_code.cfm")
            window_before = driver.window_handles[0]
1004         parent_window_title = driver.title
```

Listing 12: odlazak na stranicu i čuvanje naslova roditeljskog prozora

Pozicioniramo se unutar okvira unutar kojeg se nalazi veza koju želimo da pratimo. Zatim klikćemo na tekst veze koja otvara novi, dete prozor. Sadržaj deteta prozora pamti promenljiva `window_after`. Zatim se pozicioniramo u novonastali prozor i čuvamo njegov naslov u promenljivoj `child_window_title`.

```
1000         driver.switch_to.frame(driver.
find_element_by_name('result1'))
            driver.find_element_by_link_text('Open a popup
window').click()
1002         window_after = driver.window_handles[1]
            driver.switch_to.window(window_after)
1004         child_window_title = driver.title
```

Listing 13: prelazak u novi prozor pamćenje njegovog naslova

Najpre pravimo pretpostavku da su naslovi roditeljskog i deteta prozora različiti, a potom se vraćamo na roditeljski prozor i testiramo jednakost njegovog naslova i trenutnog naslova drajvera, koji, ako je sve prošlo kako treba, prolazi taj test. Ovdje se definicija funkcije *test_windows* završava.

```
1000 self.assertEqual(parent_window_title ,
1001                   child_window_title)
1002     driver.switch_to.window(window_before)
1003     self.assertEqual(parent_window_title , driver.
1004                       title)
```

Listing 14: testiranje (ne)jednakosti naslova prozora

Za kraj, standardno, treba dodati funkciju za zatvaranje resursa, u našem slučaju drajvera.

```
1000 def close_resources(self):
1001     self.driver.close()
```

Listing 15: zatvaranje resursa

Funkcije izvršavamo uz podršku biblioteke unittest:

```
1000 if __name__ == '__main__':
1001     unittest.main()
```

Listing 16: poziv funkcije *main*

4 Zaključak

U ovom radu opisana je namena alata za testiranje Selenium, njegove komponente i arhitektura, a na kraju su dati primeri upotrebe. Iz navedenih svojstava može se videti zašto je Selenium u današnje vreme jedan od najpopularnijih i najkorišćenijih alata za testiranje veb aplikacija. Ono što ga izdvaja u odnosu na konkurenciju su dostupnost korišćenja u mnogobrojnim programskim jezicima, fleksibilnost, podrška za sve rasprostranjene operativne sisteme i veb pregledače, kao i jednostavnost upotrebe. Takođe vrlo primamljiva karakteristika je to što je alat otvorenog koda.

Literatura

- [1] Components of selenium. on-line at: <https://www.educba.com/components-of-selenium/>.
- [2] Page object model (pom) page factory: Selenium web-driver tutorial. on-line at: <https://www.guru99.com/page-object-model-pom-page-factory-in-selenium-ultimate-guide.html>.
- [3] The selenium browser automation project. on-line at: <https://selenium.dev/documentation/en/>.
- [4] What is command, target and value fields in selenium ide? on-line at: <https://seleniumonlinetrainingexpert.wordpress.com/2012/11/25/what-is-command-target-and-value-fields-in-selenium-ide/>.
- [5] Baiju Muthukadan. Selenium with python. on-line at: <https://selenium-python.readthedocs.io/>.
- [6] Jash Unadkat. Getting started with selenium ide, 2019. on-line at: <https://www.browserstack.com/guide/what-is-selenium-ide>.