Upotreba java skript koda u procesu automatskog testiranja programa

Zoran Škrkar
Student drugog ciklusa studija
Elektrotehnički fakultet Istočno Sarajevo, Bosna i Hercegovina
skrkar.zoran@gmail.com

Sažetak – **U ovome radu detaljno ćemo da opišemo proces povećanja kvaliteta programskih rješenja koje se postiže sa adekvatnim i temeljnim testiranjem sistema. Uz korištenje metoda automatskog testa kao rezultat dobijamo stabilan kod produkcionog aplikativnog rješenja. Danas na tržištu postoji nekoliko vrsta glavnih internet pretraživača i konzistentnost koda zavisi od funkcionalnosti koje rade na jednom pretraživaču, a na nekom drugom možda neće davati očekivani rezultat. Vrlo često se funkcije poput desnog klika ili dvoklika mišem ne mogu izvršiti ukoliko se koriste već gotove seleniumove funkcije. Zbog toga je integracija java skript koda u automatske testove, koji su implementirani u selenium Web drajveru, veoma značajna, ne samo zato što svi internet pretaživači današnjice podržavaju izvršavanje java skript koda na svojim platformama, već i zato što se sve dodatne funkcije mogu izvršiti upravo kroz java skript kod.**

Ključne riječi - Java skript kod; Automatsko testiranje; selenium Web drajver, internet pretraživač.

I. UVOD

Potreba za testiranjem sistema proizilazi iz činjenice da u praksi ne postoji idealan program i da svaki u sebi sadrži neku grešku ili propust. Tako da se može reći da je osnovni razlog testiranja programa pronalaženje potencijalnog problema. Razvojni tim u procesu razvoja koda je sklon greškama, koje su posebno izražene u programima i programskim sistemima, [1].

Automatsko testiranje računarskih programa sve više dobija na značaju i u poslednjih desetak godina ova vrsta testiranja je doživjela svojevrstan preporod. Pojavilo se veliki broj tehnologija i okruženja koje nude određene pristupe kad je automatsko testiranje u pitanju. Jedan od najpopularnijih i najzastupljenijih alata za automatsko testiranje je selenium. Njegova glavna prednost je ta što se veoma lako integriše u sve programske jezike današnjice.

Posebnu masovnu upotrebu selenium je doživio onog momenta kad je u upotrebu pušten selenium Web drajver, [2]. On je omogućio jednostavno pokretanje koda na skoro svim vrstama internet pretraživača, bez potrebe za promjenom koda samog testa. Prvobitna ideja je bila da se pozove Web drajver za konkretan pretraživač i testiranje može da počne. Problemi koji su se tad pojavili zahtjevali su izvjesne izmjene u pristupu samom kodu testa.

U praksi postoji niz situacija u kojima jedan kod ne daje iste rezultate na različitim pretraživačima. Što svakako ne može da se svrsta pod način programiranja ili programersku grešku. Neki Web element jednostavno nije "vidljiv" u određenom pretraživaču, i test pada u momentu pristupa tom elementu. Naročito se problem javlja kad se pristupa elementu koji se dinamički stvara, pa nema stalan selektor. Rješenje ovoga problema pronađeno je u java skript pristupu koji je postao sastavni dio svake Web aplikacije i podržan je od strane bilo kog internet pretraživača. Kao rezultat nastao je interfejs *eng.JavaScriptExecutor*, [3].

Prednost upotrebe java skript koda prilikom testiranja je ta što je svaki elemenat koji je učitan na stranicu dostupan java skript funkcijama. Što nije slučaj u konvencionalnom pristupu, nekad je dovoljno da se izgubi fokus na elementu, prelaskom na drugi elemenat i da taj prethodni više nije dostupan selenium funkcijama. Rezultat je greška zbog isteka vremena i test pada. U tom slučaju potrebno je ponovo uraditi fokus na traženi elemenat, ukoliko postoji veliki broj elemenata, što je najčešće i slučaj, potebno je dosta vremena da se izvrti test, jer prije svakog fokusa radi stabilnosti koda postavlja se pauza u vidu *Thread.Sleep(n);* što usporva izvršavanje koda za zadato vrijeme.

Upotreba java skript koda zaobilazi ove probleme, izvršavanje koda testa se ne prekida i vrlo rijetko je potrebno pauzirati test, jer jednom učitani elementi na stranici uvijek su dostupni java skript funkcijama, i elementima je lako pristupiti.

U ovom radu je analizirana integracija java skript koda u kod automatskog testa, sa akcentom na jednostavnim java skript funkcijama, kao što su klik na dugme, dvoklik, desni klik i slično.

II. ZVRŠAVANJE JAVA SKIRPT KODA (JAVASCRIPTEXECUTOR)

Kao što je prethodno navedeno java skript kod je moguće pozvati u kodu automatskog testa korištenjem klasa *JavaScriptExecutor* interfejsa. Ovaj interfejs nudi dva načina izvršavanja skripti, klasični i asinhroni, [4]. Ovdje će biti opisan klasični pristup izvršavanja java skripti, u kojem se java skript kod izvršava u trenutno aktivnom prozoru, bilo

kroz *popup, ifrejm* ili glavni prozor pretraživača. Java skript kod se pokreće unutar bilo koje funkcije koja obavlje neke zadate operacije, sl.1.

Slika 1. Ubacivanje java skript koda (ilustracija)

Kao što se vidi na sl.1, java skript kod koristi se kako bi se obavile neke jednostavne operacije kao što su klik na dugme, dvoklik, ili desni klik.

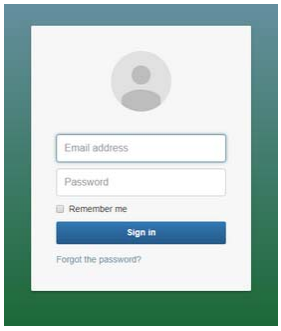
Takođe moguće je obavljati i neke složenije operacije, na primjer prebrojavanja elemenata niza ili kolona tabele, pri čemu se konačne vrijednosti mogu vratit kao povratni argumenti funkcije unutar java skript koda, i to u vidu stringa ili numeričke vrijednosti. Java skript kod se vrlo dobro slaže sa HTML (*eng. Hypertext Markup Language*) strukturom koja je osnov za sve ono što korisnik "vidi" na ekranu pretraživača. Zaključak je da su mogućnosti velike, i da manipulacija elementima stranice pomoću java skript koda dobija jednu potpuno novu dimenziju, jer za java skrit kod ne postoji dio stranice koji nije dostupan za manipulaciju.

Osnovna sintaksa je:

```
IJavaScriptExecutor js;

string JsCode = jsCode;

js = Driver.Instance as IJavaScriptExecutor;

js.ExecuteScript(jsCode);
```

JsCode predstavlja java skript koji treba izvršiti. To može biti java skript funkcija ili neka jednostavna naredba kao što je klik, ili dvoklik i sl. Sintaksa koja je predstavljena je sintaksa C# programskog jezika.

Na sl.2 prikazan je jedan klasični prozor za prijavu korisnika na sistem. Ono što je potrebno kao preduslov jeste da korisnik postoji u bazi korisnika i da se upotrebom svojih pristupnih podataka, u ovom slučaju email-a i šifre, prijavi na sistem.



Slika 2. Prozor za prijavu na sistem

Prvo što je neophodno nakon otvaranja odgovarajuće stranice, jeste da se locijraju elementi koji su potrebni prilikom prijave na sistem. U ovom slučaju to su polja za unos email-a, šifre i dugme za prijavu. U polja za email i šifru unose se odgovarajući podaci, i klikom na dugme koristeći JavaScriptExecutor prijava na sistem je obavljena.

```
jsExecutor.ExecuteScript("arguments[0].click();",loginButton
                                                        );
```

Ukoliko se pristupa direktnim pristupom elementu kroz java skript kod klik se može obaviti na način.

```
js.ExecuteScript("$('#loginButton ').click()");
```

Samo pozivanje funkcije za klik u java skript kodu može da se obavi na više načina, pa u zavisnosti od toga i ovaj dio koda može imati nekoliko verzija. Pored klika na dugme, java skript kod može da se koristi i prilikom upisa vrijednosti u polja za email i šifru, ali taj pristup nije neophodan, jer seleniumove funkcije za upis teksta u polje rade bez problema na svim pretraživačima. Ono što mora da se pomene je to da automatski test pada ukoliko se pristupa elementu koji nije dostupan. U tom slučaju ni java skript kod nije od pomoći. Java skript će da klikne na dugme, ali pošto dugme nije učitano ništa se neće desiti. Nedostupan element može biti rezultat sporosti samog sistema, kao na primjer preopterećenosti servera, loše internet konekcije i sl., ali može biti i problem koji treba da se detektuje. Da se ne bi dešavalo da test pada čekajući da se elemenat pojavi postoji pristup u kojem se "osluškuje" postojanje elementa na tačno definisane vremenske razmake. Najčešće je to razmak od jedne sekunde pri čemu traženje elementa traje jedan minut, ove vrijednosti se mogu i predefinisati.

Kod za provjeru dostupnosti elementa je:

```
IsElementPresent(By.Id(“inputEmail”)));

IsElementPresent(By.Id(“inputPassword”)));

IsElementPresent(By.Id(“loginButton”)));
```

Ovaj dio koda se najčešće ubacuje u for petlju, gdje se svake sekunde provjerava istinitost iskaza, tako što se nakon svake iteracije kod zaustavi jednu sekundu pomoću,

```
Thread.Sleep(1000);
```

Ukoliko je tvrdnja istinita, dakle elemenat je učitan i dostupan, izlazi se iz for petlje naredbom

```
break;
```

zatim se nastavlja dalje sa koracima testa. Ukoliko se dođe do kraja vremenskog intervala, opet se izlazi iz for petlje, na isti način, ali ovog puta zbog greške jer elemenat nije učitan.
U tom slučaju test će da padne i greška se proslijedi na email ili se upiše u neki fajl kako bi se naknadno ručno provjerilo o čemu se radi. Prikazani primjer odnosi se na operaciju klik. Vrlo često se javlja potreba dvoklika na dugme ili desnog

klika zbog otvaranja dodatnih opcija, kao i opcija prelaska mišem preko nekog elementa, radi pojave dodatnih opcija.

U konkretnom slučaju naredba za dvoklik bi izgledao ovako,

```
js.ExecuteScript("$('#loginButton ').dblclick()");
```

Desni klik,

```
js.ExecuteScript("$('#loginButton ') .trigger("contextmenu")"
```

A prelazak mišem preko elementa radi se na sledeći način.

```
js.ExecuteScript("$('#loginButton ') .trigger("mouse over")"
```

Pored ovih jednostavnih naredbi koje se mogu izvesti pomoću java skripta, moguće je izvesti i niz drugih opcija, kao i funkcija koje testeri kreiraju po potrebi u java skript kodu. Na taj način vrlo jednostavno mogu da dobiju neke podatke sa stranice kojim inače ne bi mogli da manipulišu.

Kod kojim se pokreće bilo koja java skripta iz fajla je:

```
string jsCode = File.ReadAllText(jsFile);

string values = (string)js.ExecuteScript(jsCode);
```

Pročita se java skripta i u sledećem koraku se kod izvrši. Prethodno se vrši kastovanje u tip string radi lakše obrade povratnih podataka.

Kao što se vidi gore u dva reda koda može da se izvrši bilo koja java skripta unutar testa koji je na primjer pisan u C# programskom jeziku.

III. FAKTOR ANALIZE KVALITETA

Trendovi razvoja računarskih sistema danas imaju potrebu da se sve što se radi uradi i *upload* na Web ili na oblak sisteme. Ta potreba za posljedicu ima potrebu generisanja što kvalitetnijih programskih rješenja, što takođe utiče i na način testiranja gotovih računarskih sistema. Iz prethodnih primjera datih u poglavlju 2. se vidjelo na koji način se java skript kod integriše u C# kod, taj pristup je u velikoj mjeri olakšao stvari prilikom testiranja. Pošto je na primjer desni klik, kojim se otvaraju nove opcije na ciljanom elementu, bio moguć samo u teoriji. Praksa je pokazala da je desni klik vrlo često bio gotovo nepremostiva prepreka za testere koji kreiraju automatske testove.

Sistemi koji su sve više u upotrebi su oni čiji se sadržaj na Web učitava dinamički, dakle konstantno se mjenjaju selektori elemenata. Korisnicima se daje širok spektar dodatnih opcija da sistem na neki način naprave ličnim. Mogu da mijenjaju položaj menija, boje, kao i da potpuno ukidaju neke elemente na stranici. Sve nabrojano predstavlja dodatni izazov za testerske timove. Poseban problem sa svim nabrojanim mogu da imaju timovi koji rade na razvoju automatksih testova, jer će sistemi postati sve teži za obradu i naravno sve veći.

Takođe je veliki broj sistema sa velikom količinom podataka koje je potrebno obraditi, a naravno prethodno i testirati. Najveći izazovi prilikom testiranja velikih sistema nije nedostatak prostora ili svođenje sistema na jedan server, već testiranje vjerodostojnosti podataka, kao na koji način obezbjediti automatsko regresijsko testiranje. Iz razloga što testeri gube vrijeme na ručno testiranje velikih sistema, pristupa se uvođenju novih inovativnih metoda u testiranje velikih sistema, kao i široka primjena uvođenja automatskih testova, [5].

Inovativne metode se svakako odnose i na što kvalitetnije ubacivanje java skript koda u sistem testiranja. Najbolji način za ubacivanje java skript koda je taj da se sve skripte napišu u posebnim fajlovima, koji se kasnije samo pozivaju na izvršenje. Prije nego da se radi svaka funkcija posebno u kodu, to je odlika lošeg stila programiranja.

Naredni korak, koji je već uveliko počeo, je prelazak sistema na oblak (*eng Cloud*) platforme. Prednost oblak platformi je što je sve dostupno na jednom mjestu. Potrebna je samo internet konekcija i eventulano pristupni podaci za konkretan nalog. Tako da se i testiranje, posebno automatsko, sve više premješta u oblak tehnologiju. U ovom slučaju što se tiče java skript koda neće biti problema, jer je u pitanju Web okruženje koje podržava java skript u potpunosti. Ono što se može očekivati u procesu automatskog testiranja je samo nadogradnja već postojećih interfejsa i klasa koje isti upotrebljava.

Jedna od tehnologija koja će takođe da nastavi svoj razvoj je svakako mobilna tehnologija. Razvoj mobilnih aplikacija predstavlja ključni faktor za QA (eng. Quality Analyst) razvoj. Životni vijek mobilnih aplikacija je relativno kompaktan, kad se poredi u odnosu na Web aplikacije. Ovo zajedno sa velikim brojem operativnih sistema koji se koriste na mobilnm uređajima, kao i velikim brojem dimenzija ekrana mobilnih uređaja, predstavlja pravi izazov za testne timove, [6]. Svakako oblast u kojoj java skript može da se upotrebi u praksi bez problema. Kod faktora analize kvaliteta treba uzeti u obzir činjenicu da se u budućnosti predviđa povećanje troškova grešaka, krajnji korisnici će postati netolerantni prema iznenadnim pucanjem aplikacija, kao i nepredviđenim funckionalnim problemima, [7].

IV. ZAKLJUČAK

Testiranje Web aplikacija je veoma kompleksan proces i zahtjeva ozbiljan pristup, tim prije što korisnik ima mogućnost da aplikaciju pokrene u bilo kom dostupnom Web pretraživaču koristeći proizvoljnu verziju istog. Upravo ova činjenica je prava noćna mora za razvojne timove, jer po pravilu najvše vremena se upravo i troši prilikom prilagođavanja koda za različite verzije internet pretraživača. Iz ovoga se izvlači zaključak da se veliki broj defekata upravo i javlja na onim djelovim koda koji se ne ponašaju isto na svim pretraživačima, koje je naknadno potrebno prilagođavati. Tako na primjer postoje posebni css fajlovi koji se učitavaju samo u internet eksploreru (*eng* Internet Explorer).

Iz svega navedenog zaključuje se i da automatski testovi nailaze na slične probleme, jer jedan isti kod može da radi na jednom pretraživaču ali na nekom drugom ne. U tom slučaju u pomoć može da pritekne java skript kod, pomoću kojeg se u velikoj mjeri mogu izvršiti svi potrebni koraci.

Takođe zbog svega što java skript predstavlja u svijetu Web platformi sve više sistema se okreće upravo ovim tehnologijama kad je u pitanju automatsko testiranje Web sistema.

LITERATURA

[1] Software Testing: A Craftsman's Approach, Third Edition (Hardcover) by Paul C. Jorgensen

[2] http://www.seleniumhq.org

[3] http://seleniumhq.github.io/selenium/docs/api/java/

[4] https://www.guru99.com

[5] softwaretestingnews.co.uk

[6] assets.kpmg.com

[7] intland.com

ABSTRACT

In this paper, we will detail the process of increasing the quality of software that is achieved with adequate and basic system testing. With the use of an automatic test, we get stable in the production application solution. Today, there are several types of major Internet browsers on the market, and the consistency of the code depends on the functionality of one browser and may not give the expected result to anyone else. Very often, functions such as right-click or double-click can not be performed if selenium functions are already in use. Therefore, the integration of java script into automated tests, which are implemented in the selenum web driver, is very significant, not only because all internet subscribers today support the execution of java code scripts on their platforms, but because all additional functions can be performed right through the java script code.

USE THE JAVA SCIRPT CODE IN THE PROCESS OF AUTOMATIC TESTING OF THE PROGRAM

Zoran Škrkar