Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/02BE42Sogura

ADAPTACIJA I ANALIZA SELENIUM TESTOVA U C# PROGRAMSKOM JEZIKU

ADAPTATION AND ANALYSIS OF SELENIUM TESTS IN C# PROGRAMMING LANGUAGE

Boško Šogura, *Fakultet tehničkih nauka, Novi Sad*

Oblast – SOFTVERSKO INŽENJERSTVO

Kratak sadržaj – *U okviru ovog rada opisan je Selenium radni okvir za implementaciju veb automatskih testova. Manipulisanje i izvršavanje akcija nad elementima veb stranice pomoću WebDriver-a. Upotreba Page Object obrasca kao najbolje prakse prilikom kreiranja test scenarija. Analiza test scenarija u najpopularnijim pretraživačima.*

Ključne reči: *Selenium, WebDriver, Automatski testovi, Page Object obrazac*

Abstract – *The subject of this paper was description of the Selenium framework for implementation Web automated tests. Manipulate and execute actions over Web page elements using the WebDriver. Using the Page Object Model as best practice in creating a test scenario. Analysis of the test scenario in the most popular browsers.*

Keywords: *Selenium, WebDriver, Test automation, Page Object Model*

1. UVOD

Testiranje predstavlja samo jedan dio procesa kontrole kvaliteta softvera. Različite metode se koriste za testiranje različitih slojeva softvera, a ono što je svima zajedničko jeste da se mogu izvoditi manuelno ili automatski. I jedan i drugi način imaju svoje prednosti i mane. U poslednje vrijeme sve se više teži automatizaciji jer se automatizacijom testiranja skraćuje i samo vrijeme testiranja.

2. AUTOMATSKO TESTIRANJE

Softveri sa kritičnom misijom prolaze kroz rigorozne funkcionalne testove, koji su često podržani radnim okvirima (eng. Framework) za automatsko testiranje. Automatizacija ovakvih platformi kao i održavanje kvaliteta softvera kroz verzije su od ključnog značaja za poslovanje. Preduzeća se često suočavaju sa dilemom između balansiranih troškova i upravljanja resursima kako bi se osiguralo da platforme za testiranje pokrivaju sve poslovne scenarije i da isporučeni softver nema grešaka [1]. Implementacijom odgovarajuće platforme za automatsko testiranje, preduzeća mogu značajno povećati brzinu i tačnost procesa testiranja.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac, van. prof.

2.1. Zašto Framework?

Framework definiše način preduzeća da organizuje stvari pomoću "Jedinstvenog Standarda". Prateći ovaj standard, projektni tim postigao bi:

- prikaz automatskih testova bez skripta,
- testove vođene podacima,
- koncizno izvještavanje,
- standardizovano skriptovanje i konzistentnost i
- implementaciju i ponovnu upotrebljivost.

3. SELENIUM

Selenium je prenosni radni okvir za testiranje veb aplikacija. On obezbjeđuje alat za pisanje testova bez potrebe da se uči ili zna test skripting jezik (eng. Slenium IDE). Takođe, pruža test jezik specifičan za domen testiranja (Selenese) u kom se mogu pisati testovi na najpopularnijim programskim jezicima kao što su: C#, Groovy, Java, Perl, PHP, Python, Ruby i Scala. Testovi se mogu pokretati u većini modernih veb pretraživača. Selenium može da se koristi na Windows, Linux i macOS platformama. On je open-source softver, objavljen pod Apache 2.0 licencom, što znači da ga veb programeri mogu preuzeti i koristiti bez naknade [2].

3.1. Selenium korisnički API

Kao alternativa pisanju testova u Selenese-u, testovi se mogu pisati i u različitim programskim jezicima. Ovi testovi potom komuniciraju sa Selenium-om pozivom metoda njegovog korisničkog API-ja. Selenium trenutno nudi korisničke API-je za Javu, C#, Ruby, JavaScript i Python. Sa novom verzijom Selenium-a 2, uveden je novi korisnički API, koji ima WebDriver kao centralnu komponentu. Međutim, stari API, koji koristi Selenium klase, još uvijek je podržan i održava se.

3.2. Selenium WebDriver

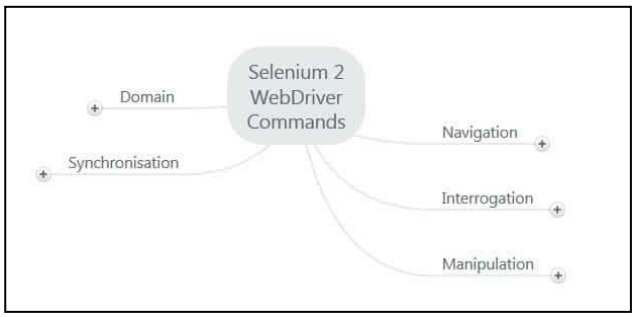
Selenium WebDriver je naslednik Selenium RC-a. On prihvata komande iz Selenese-a ili korisničkog API-ja i šalje ih pretraživaču. Ovo se implementira pomoću posebnog driver-a za pretraživače, koji šalje komande samom pretraživaču i vraća rezultat, odnosno odgovor pretraživača. Većina driver-a za pretraživače zaista pokreće i pristupa aplikaciji pretraživača (kao što su Firefox, Chrome, Internet Explorer, Safari, Edge).

Tu je i HtmlUnit driver, koji omogućava simuliranje pretraživača (nema grafičku reprezentaciju pretraživača). Za razliku od Selenium-a 1, gdje je Selenium server bio neophodan za pokretanje testova, Selenium WebDriver-u nije potreban poseban server za izvršavanje testova. Umjesto toga, WebDriver direktno pokreće instancu pretraživača i kontroliše je.

Međutim, Selenium Grid se može koristiti u kombinaciji sa WebDriver-om za izvršavanje testova na udaljenim sistemima.

Kada je moguće, WebDriver koristi funkcije operativnog sistema a ne JavaScript komande koje su specifične za pretraživač, radi pokretanja istog. Ovaj pristup zaobilazi probleme sa razlikama između komandi operativnog sistema i JavaScript-a, uključujući bezbjednosna ograničenja [3].

Ovo je savršena mapa uma za testere koji koriste WebDriver na projektima automatskog testiranja. Na slici 1. prikazano je osnovnih pet kategorija Selenium WebDriver komandi [4].



Slika 1. *Osnovne kategorije WebDriver komandi*

3.3. Page Object Model obrazac

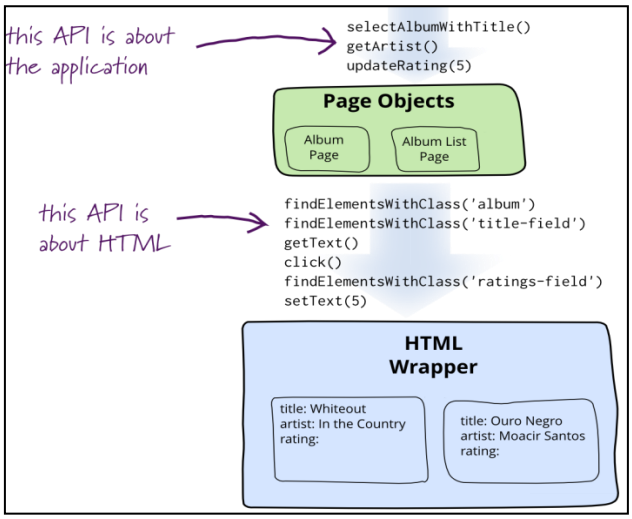
POM je najčešće korišćen dizajn obrazac u kom se svaka veb stranica (ili bar značajne) posmatra kao klasa. Na svakoj od ovih klasa stranica (otuda i Page Object) mogu se definisati njihovi elementi i metode koje su specifične za tu stranicu. Svaka klasa treba da predstavlja stranicu veb aplikacije. Ovaj obrazac predstavlja sloj između test skripti i korisničkog interfejsa i enkapsulira funkcije veb stranice [5].

Page Factory je dodatak POM obrascu koji se koristi za inicijalizaciju veb elemenata koji su definisani na stranici, u klasi koja "preslikava" tu veb stranicu. Elementi se mogu definisati koristeći anotacije pomoću Page Factory obrasca [6].

Kada se pišu testovi koji testiraju određenu veb stranicu, potrebno je da da se koriste elementi i funkcionalnosti te veb stranice.

Međutim, ukoliko takvi testovi direktno manipulišu HTML elementima, oni će biti krhki na promjene u korisničkom interfejsu. POM obrazac omogućava manipulaciju veb elementima u test scenarijima bez kopanja po HTML-u u samim test skriptama.

Slika 2. prikazuje kakve metode bi trebalo da se koriste u test skriptama. Osnovno pravilo POM obrasca jeste da omogući softverskom klijentu da vidi i uradi sve ono što može i čovjek na stvarnim veb stranicama.



Slika 2. *Primjer Page Object Modela*

4. OPIS PROJEKTNOG RJEŠENJA

4.1. ICodeFactory sajt

ICodeFactory je kompanija koja izvršava proces razvoja softvera od ideje do proizvodnje [7]. Na njihovom oficijalnom sajtu www.ICodeFactory.com mogu se pronaći osnovne informacije o firmi, usluge koje pružaju, tehnologije i projekti na kojima rade, najnovije vijesti vezane za poslovanje firme, konkurse za otvorene pozicije u firmi kao i mnoge druge informacije.

Sajt takođe pruža nekoliko administratorskih funkcionalnosti, ali je prvo potrebno prijaviti se na stranici za prijavljivanje. Funkcionalnosti koje prijavljeni korisnik može da izvrši (zavisno od njegovih prava) su:

- pregled, unos, izmjene i brisanje aktuelnih vijesti koje se prikazuju u sekciji "News",
- pregled, unos, izmjene i brisanje trenutno otvorenih pozicija koje se prikazuju u sekciji "Career",
- pregled, unos, izmjene i brisanje drugih aktivnih korisnika,
- pregled, kreiranje, izmjene i brisanje aktivacionih kodova za razne servise,
- pregled log fajlova i
- pregled, čuvanje i skidanje dokumenata.

4.2. Test scenario

Test scenario predstavlja niz akcija koje zajedno testiraju određenu funkcionalnost. U daljem nastavku ovog poglavlja biće analiziran scenario za dodavanje nove vijesti na ICodeFactory sajtu. Koraci ovog scenarija su:

1. Otvaranje početne stranice sajta i klik na poruku o kolačićima.
2. Klik na dugme za prijavljivanje i prelazak na stranicu za prijavljivanje.
3. Unos korisničkog imena i lozinke u odgovarajuća polja i klik na dugme za prijavljivanje.
4. Navigacija na administratorsku stranicu za pregled vijesti kao prijavljen korisnik.

5. Klik na dugme za dodavanje nove vijesti i prelazak na stranicu za dodavanje vijesti.

6. Unos potrebnih informacija za kreiranje vijesti i klik na dugme za dodavanje.

7. Provjera očekivanih i dobijenih rezultata nakon dodavanja vijesti.

4.3. Inicijalizacija testa

Ovaj dio testa je zajednički za sve test scenarije. U ovom koraku se radi inicijalizacija baze, kreiranje WebDriver objekta, pronalazak i klik na poruku o kolačićima.

4.4. Prelazak na stranicu za prijavljivanje

NewsRelatedScenarios klasa sadrži test scenarije koji testiraju funkcionalnosti vezane za vijesti sajta. Slika 3. prikazuje metodu za inicijalizaciju spomenute klase. Nakon inicijalizacije testa, vrši se kreiranje objekta početne stranice (eng. HomePage).

Sledeći korak testa jeste prelazak na stranicu za prijavljivanje.

To se izvršava pozivom NavigateToLoginPage metode nad futer objektom početne stranice. Metoda vraća objekat stranice za prijavljivanje čime se završava ovaj korak testa.

```
[TestFixture]
-- references | Bosko Sogura, 2 days ago | 1 author, 1 change
class NewsRelatedScenarios : BaseScenario
{
    private ApplicationStructure.AdminPages.NewsPage newsPage;

    [SetUp]
    -- references | Bosko Sogura, 2 days ago | 1 author, 1 change
    public override void Initialize()
    {
        base.Initialize();

        // Login for all these tests as content editor and navigate to admin news page:
        var homePage = new HomePage(Driver);
        var loginPage = homePage.Footer.NavigateToLoginPage();
        homePage = loginPage.LoginAsContentEditor();
        newsPage = homePage.MainMenu.NavigateToAdminNewsPage();
    }
```

Slika 3. *Prikaz Initialize metode klase koja sadrži scenarije o vijestima*

NavigateToLoginPage metoda locira dugme za prelazak na stranicu za prijavljivanje, izvršava klik nad dugmetom i zatim vraća kreiranu instancu stranice za prijavljivanje.

4.5. Prijavljivanje korisnika

U ovom koraku se unose korisničke informacije u odgovarajuća polja na stranici za prijavljivanje. Na slici 3. u metodi za inicijalizaciju, može se vidjeti metoda LoginAsContentEditor koja se poziva nad objektom stranice za prijavljivanje. Na slici 4. je prikazana metoda za prijavljivanje korisnika sa pravima menadžera sadržaja.

Metoda prihvata opcione parametre koji su korisničko ime i lozinka.

Ukoliko se ne navedu parametri, korisničko ime i lozinka se uzimaju iz app.config fajla gdje su definisani različiti korisnici, sa različitim pravima, odnosno grupama kojim pripadaju.

Zatim se poziva metoda LoginUser kojoj se kao parametri proslijeđuju korisničko ime i lozinka.

```
5 references | Bosko Sogura, 3 days ago | 1 author, 1 change
public HomePage LoginAsContentEditor(string username = null, string password = null)
{
    if (username == null || password == null)
    {
        username = UserStore.Editor.Username;
        password = UserStore.Editor.Password;
    }

    return LoginUser(username, password);
}
```

Slika 4. *Prikaz metode za prijavljivanje korisnika kao menadžera sadržaja*

4.6. Navigacija na administratorsku stranicu za pregled vijesti

Poslednja komanda u Iitialize metodi, koja je prikazana na slici 3. jeste navigacija na administratorsku stranicu za pregled vijesti (eng. NewsPage). U predhodnom koraku testa, nakon prijave korisnika, vraćena je instanca početne stranice sajta. U ovom koraku, pozivamo metodu NavigateToAdminNewsPage nad objektom menija koji se nalazi u klasi početne stranice.

4.7. Navigacija na administratorsku stranicu za dodavanje vijesti

Nakon izvršavanja koda Initialize metode, prelazi se na glavnu metodu testa. Na slici 5. je prikazana Home_LoginAndAddNews_CheckNewsTitleInTable metoda, koja predstavlja glavni dio test scenarija koji se opisuje. Na početku ove metode se deklarišu osnovne informacije kao što su naslov vijesti i broj postojećih vijesti do sada. Ove informacije se koriste i na kraju za samu verifikaciju, odnosno provjeru da li je test izvršio očekivane promjene. Nakon definisanja informacija za verifikaciju, poziva se metoda za prelazak na administratorsku stranicu za dodavanje vijesti (eng. AddNewsPage).

```
[Test]
0 references | Bosko Sogura, 3 days ago | 1 author, 1 change
public void Home_LoginAndAddNews_CheckNewsTitleInTable()
{
    // arrange:
    var testTitle = "test title";

    var oldNumberOfNews = newsPage.NewsCount;
    var addNewsPage = newsPage.NavigateToAddNews();

    // act:
    newsPage = addNewsPage.InsertTitle(testTitle)
                          .InsertDescription("Some description")
                          .InsertContent("insert text")
                          .TogglePublish()
                          .SelectPublishDate(DateTime.Now.AddDays(-5))
                          .SaveNews();

    // assert:
    Assert.AreEqual(oldNumberOfNews + 1, newsPage.NewsCount);
    Assert.AreEqual(testTitle, newsPage.GetNewsTitle(3));
}
```

Slika 5. *Prikaz glavnog dijela testa za dodavanje nove vijesti*

NavigateToAddNews metoda vrši lociranje odgovarajuće stavke menija pomoću identifikatora, pa zatim izvršava akciju klika nad njim. Nakon klika očekuje se prelazak na stranicu za dodavanje vijesti. Ovaj korak se završava kreiranjem objekta administratorske stranice za dodavanje vijesti.

4.8. Unos potrebnih informacija i kreiranje vijesti

Ovo je najvažniji korak test scenarija. Takođe, korak u kom najčešće padaju testovi. U ovom koraku se unose potrebne informacije za kreiranje nove vijesti u odgovarajuća polja i zatim se pokušava snimanje unijete vijesti. Na slici 5. se vidi nekoliko poziva različitih metoda nad objektom stranice za dodavanje vijesti, koje vrše simulaciju unošenja određenih informacija u polja za unos. InsertTitle metoda vrši lociranje polja za unos u koji se upisuje naslov vijesti. InsertDescription metoda se koristi za dodavanje opisa vijesti, itd.

4.9. Provjera dobijenih i očekivanih rezultata pri završetku testa

Nakon unosa vijesti, vrši se provjera da li je ona stvarno unijeta. Prvo se provjerava da li se broj vijesti na stranici za pregled uvećao za jedan u odnosu na stari broj prije dodavanja. Druga provjera je da li maloprije unijeti naslov vijesti odgovara naslovu u tabeli gdje se izvršilo dodavanje. Ukoliko obje provjere vrate pozitivne rezultate, smatra se da je test uspješno izvršen.

4.10. Uporedna analiza

Tabela 1. *Uporedna analiza testova u Chrome-u, Firefox-u i Internet Explorer-u*

Pretraživač	Prosječno vrijeme testa [s]	Postotak uspješnosti [%]
Chrome	21,86	95
Firefox	31,64	80
Internet Explorer	41,93	100

Analiza je izvršena tako što se test scenario, koji je opisan u poglavlju 4.2. pokrenuo 20 puta u svakom od navedenih pretraživača. Zatim se izračunalo prosječno vrijeme testa i postotak uspješnosti izvršavanja u svakom od pretraživača.

Iz tabele 1. može da se zaključi da Chrome pretraživač ima najkraće prosječno vrijeme izvršavanja testova. Takođe, uspješnost ovih testova je veoma visoka (95%), što ga stavlja na prvo mjesto ove analize. Mozilla Firefox ima duže prosječno vrijeme izvršavanja testova od Chrome-a za čitavih 10 sekundi. Sa uspješnošću testova od 80%, što znači da će svako peto izvršavanje testa da bude neuspješno. Firefox predstavlja najslabiji pretraživač od svih analiziranih kada se uzme u obzir trajanje testova i uspješnost istih. Internet Explorer ima prosječno vrijeme izvršavanja testova za 20 sekundi duže od Chrome-a, i 10 sekundi duže od Firefox-a. Ono što ipak stavlja IE na drugo mjesto ove analize, jeste činjenica da se testovi izvršavaju sa uspješnošću od 100%.

Potrebno je spomenuti i da za inicijalizaciju baze treba u prosjeku 9 sekundi po testu. Što znači da se prosječna vrijemena testova mogu dodatno skratiti ukoliko se primjene druge metode za inicijalizaciju baze. Prosječno vrijeme testa se može smanjiti još više upotrebom kolačića da bi se izbjegao dio vezan za prijavljivanja korisnika u svakom test scenariju.

5. ZAKLJUČAK

U ovom radu opisan je problem osiguravanja kvaliteta veb aplikacije i njegovo rješenje uvođenjem automatskog testiranja. Automatski testovi su poželjni u svakoj aplikaciji jer testiraju funkcionalnosti sa istim podacima u istom okruženju svaki put. Tako da se, nakon svake izmjene na aplikaciji koja se nalazi u produkciji, mogu pokrenuti testovi koji testiraju sve funkcionalnosti aplikacije. Tako se može lako utvrditi bilo kakva greška nastala sa poslednjim izmjenama.

Možda i najvažnija osobina opisanog projekta za testiranje jeste lako prebacivanje između instanci pretraživača u kojima se test scenariji izvode. To se radi izmjenom odgovarajuće promjenljive app.config fajla. Na taj način se pokriva veliki dio različitih produkcionih okruženja (pretraživača) u kojima korisnici mogu da koriste aplikaciju.

Moguća poboljšanja ovog projekta za testiranje bi bila upotreba kolačića kao mehanizma za prijavljivanje korisnika. Pomoću kolačića, mogu se direktno u pretraživač unijeti podaci o korisnicu, tako da bi veb aplikacija smatrala da je korisnik već prijavljen i dozvolila mu pristup određenim funkcionalnostima. Takođe, upotreba snimaka stanja baze prije i posle testa i zatim vraćanje na stanje prije testa. Trenutno se koristi inicijalizacija baze iznova prije svakog testa, što znači da se čitava baza obriše i opet popuni odgovarajućim podacima. Upotreba ovih mehanizama bi dodatno smanjila vrijeme izvršavanja test scenarija.

6. LITERATURA

[1] http://toolsqa.com/selenium-webdriver/automation-framework-introduction/ (pristupljeno u septembru 2018.)

[2] https://www.seleniumhq.org/ (pristupljeno u septembru 2018.)

[3] "The Architecture of Open Source Applications: Selenium WebDriver". *www.aosabook.org*. (pristupljeno u septembru 2018.)

[4] https://www.swtestacademy.com/selenium-webdriver-api/ (pristupljeno u oktobru 2018.)

[5] http://www.assertselenium.com/automation-design-practices/page-object-pattern/ (pristupljeno u oktobru 2018.)

[6] https://www.guru99.com/page-object-model-pom-page-factory-in-selenium-ultimate-guide.html (pristupljeno u oktobru 2018.)

[7] https://www.icodefactory.com/about (pristupljeno u oktobru 2018.)

Kratka biografija:

Boško Šogura rođen je u Bijeljini 1993. god. Master rad na Fakultetu tehničkih nauka iz oblasti Softverskog inženjerstva – Testiranje softvera odbranio je 2018.god.
kontakt: bolesogura93@gmail.com