

RAČUNARSTVO USLUGA I ANALIZA PODATAKA

*"Testing is an infinite process of comparing the invisible to the ambiguous in order to avoid the unthinkable happening to the*

*anonymous."*

*James Bach*

LV4: Testiranje softvera

**Sadržaj**

[1. Uvod 3](#_Toc21569)

[1.1. Što je testiranje? 3](#_Toc21570)

[1.2. Ciljevi testiranja 3](#_Toc21571)

[1.3. Zašto testiramo softver? 4](#_Toc21572)

[1.4. Test proces 6](#_Toc21573)

[1.5. Razine, metode, vrste i alati za testiranje 6](#_Toc21574)

[1.6. Testna dokumentacija 7](#_Toc21575)

[2. Automatsko testiranje 7](#_Toc21576)

[2.1. Katalon Automation Recorder/Katalon Studio 9](#_Toc21577)

[2.2. Selenium WebDriver 11](#_Toc21578)

[2.3. NUnit framework 11](#_Toc21579)

[3. Zadaci 12](#_Toc21580)

[3.1. Zadatak 1 12](#_Toc21581)

[3.2. Zadatak 2 14](#_Toc21582)

[3.3. Zadatak 3 16](#_Toc21583)

[3.4. Zadatak 4 17](#_Toc21584)

[3.5. Dodatni zadaci 17](#_Toc21585)

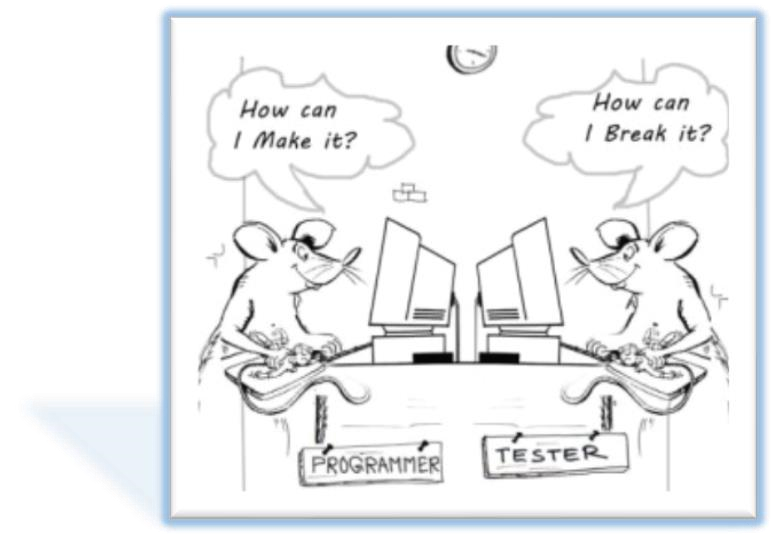
# Uvod

## Što je testiranje?

**Testiranje** je proces ispitivanja softvera i planirani skup aktivnosti koje se kreiraju tokom razvoja projekta kako bi se utvrdilo jesu li u skladu s očekivanim zahtjevima. Navedenim postupkom dobiva se objektivna informacija o njegovoj **kvaliteti**. S poslovne strane, testiranje omogućava procjenu rizika pri implementaciji softverskog rješenja.

Testiranjem se provjerava:

* Zadovoljava li sustav definirane (ugovorene) tehničke **zahtjeve**?
* Radi li kako **korisnici** to očekuju?
* Je li sustav spreman za **implementaciju** (produkciju)?



Slika 1: Developer vs tester

 Testiranje softvera je samo dio cjelokupnog procesa osiguranja kvalitete softvera. Prema istraživanjima tijekom razvoja softvera na testiranje potroši i do **40%** razvojnog vremena[[1]](#footnote-1).

## Ciljevi testiranja

1. **Verifikacija**: *Da li dobro gradimo proizvod?*

Verifikacija je evaluacija softvera kako bismo utvrdili da proizvod u određenoj fazi razvoja zadovoljava zadane uvjete. Započinje na početku razvoja i uključuje revizije, evaluacije dokumenata, planova i specifikacija. Navedenim procesom utvrđujemo da naš proizvod pokriva sve funkcionalnosti prema funkcionalnoj specifikaciji.

1. **Validaciju**: *Da li gradimo dobar proizvod?*

Validacija je proces evaluacije proizvoda za vrijeme ili na kraju razvoja u svrhu dokazivanja da proizvod odgovara definiranim zahtjevima i čini ono što korisnik stvarno želi. Navedenim procesom utvrđujemo da sustav odgovara zadanim zahtjevima i odrađuje sve zadane funkcije.

Opisanim postupcima otkrivamo nedostatke u softveru (greške) i procjenjujemo koliko je sustav upotrebljiv. Softver nikada nije potpuno bez nedostataka, ali mora zadovoljiti zadane zahtjeve da bi ispunio namjenu.



Slika 2: Verifikacija vs. validacija vs. potvrda

## Zašto testiramo softver?

Tvrtke koje razvijaju softver suočavaju se s gubicima upravo zbog velikog broja defekata u softveru isporučenom klijentu. **Štete** pojedinih grešaka u softveru mogu imati velike posljedice s obzirom na to da se računala i softver koriste u kritičnim aplikacijama. Greške mogu prouzrokovati ogromne gubitke.

U kritičnim sustavima greške su uzrokovale padove aviona, ugrožavale svemirske misije, zaustavljale trgovanja na burzama. Softverske greške mogu prouzrokovati katastrofe. Takozvana **Y2K** greška potakla je osnivanje cijele grane konzultanata i stvaranje programskih alata kako moderno društvo ne bi stalo u prvoj sekundi novog tisućljeća. U modernom društvu pouzdanost i kvaliteta softvera imaju presudnu ulogu u procesu razvoja softvera.

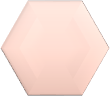
Nesavršenost ljudske prirode pridonosi činjenici da je gotovo nemoguće napisati složeniji program bez ijedne pogreške. Pronalazak problema i njihovo otklanjanje je osnovna zadaća u fazi implementacije softvera.

Glavni zadatak test inženjera jest otkrivanje *bugova* u softveru s ciljem da se otkloni prije isporuke proizvoda klijentu (kupcu) čime je osigurano zadovoljstvo kupca. Također je potrebno identificirati i pratiti uočeni problem do mjesta nastanka (izvora) greške u softveru.



Iscrpno testiranje je

nemoguće

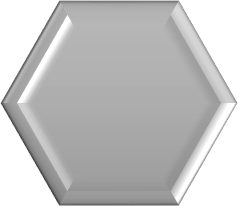
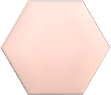


Testiranjem

dokazujemo

postojanost

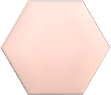
defekata



Testiranje u

ranoj fazi

razvoja



Testiranje

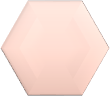
povezanih

defekata (

*Defect*

*clustering*

)



Povezanost

grešaka u

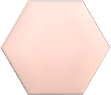
modulima

(

*Pesticide*

*Paradox*

)



Pristup

testiranja prema

vrsti projekta

(

*Context*

*dependent*

)



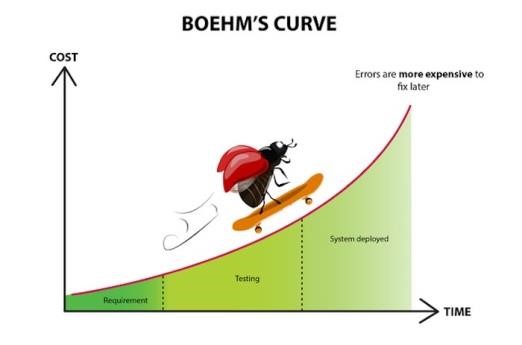
Zabluda

nepostojanja

grešaka

Slika 3: Načela testiranja

Troškovi otklanjanja uočenih grešaka, umjesto sprječavanja njihovog nastanka su veliki i prouzrokuju veliki gubitak u poslovanju i nezadovoljstvo kupca zbog grešaka u softveru. Prije nego što se čeka završetak faze implementacije komponente softvera, pa tek onda testiranje komponente u cilju otkrivanja i otklanjanja grešaka u njima, a koje su nastale u ranijim fazama procesa razvoja, potrebno je preventivno djelovati na nastanak tih grešaka, kako bi se izbjegla kašnjenja i uvećali troškovi njihovog otklanjanja.



Slika 4: Ovisnost troška i faze u kojoj je defekt prijavljen

## Test proces

Aktivnosti koje se primjenjuju tokom faza projekta:

Planiranje testiranja

-

analiza zahtjeva, definiranje testova, metoda, alata

Dizajn testova

-

analiza sustava koji će se testirati;

kreiranje skupa test slučajeva i procedura koje će se koristiti tijekom testiranja sustava

Izvršavanje testova

-

primjena opisanih test slučajeva u

skladu s planom testiranja i dizajnom

Prijava i rješavanje problema

Prateće aktivnosti

:

praćenje statusa testiranja, kontrola kvalitete, unaprjeđenje procesa testiranja

Slika 5: Aktivnosti u procesu testiranja

## Razine, metode, vrste i alati za testiranje

Proces testiranja potrebno je prilagoditi potrebama projekta i sustava koji se testira. Na temelju analize zahtjeva sustava i funkcionalne specifikacije biraju se najprimjenjiviji aspekti testiranja.

Slika

6

Razine, metode, vrste i strategije testiranja

:

Ručno

Automatsko

Istraživačko

Regresijsko

Metode



Testiranje

funkcionalnosti



Testiranje

performansi



Testiranje

sigurnosti



Pregled

koda

Vrste

Bijela kutija

Crna kutija

Usporedno

testiranje

Strategije

Jedinično

Integracijsko

Sistemsko

Razine

## Testna dokumentacija

Test plan

Pokazuje strategiju po

kojoj će se testovi

izvršavati da bi se

aplikacija istestirala, koji

će se resursi koristiti.

Sadrže podatke o test

okolini, raspored

testiranja, lista testova,

lista funkcionalnosti koje

se testiraju, rizici.

Test scenarij

Test scenarij nam kaže

koje područje ćemo

testirati.

Testirati se može jedno ili

više područja, svaki

scenarij pokriva jedno

područje.

Test slučaj

Test case sadrži set koraka,

uvjeta i inputa koji se koriste za

izvršavanje tog taska.

Format : Test case ID. Purpose,

Assumptions,

Pre

-

Conditions, Steps,

Expected and Actual Outcome,

Post Conditions

Svaki test case sadrži očekivane

rezultate.

Usporedbom dobivenog i

očekivanog odlučujemo da li je

taj Test case

*prošao*

ili

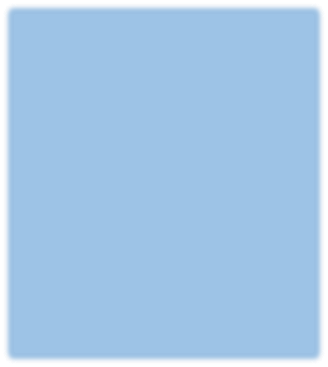
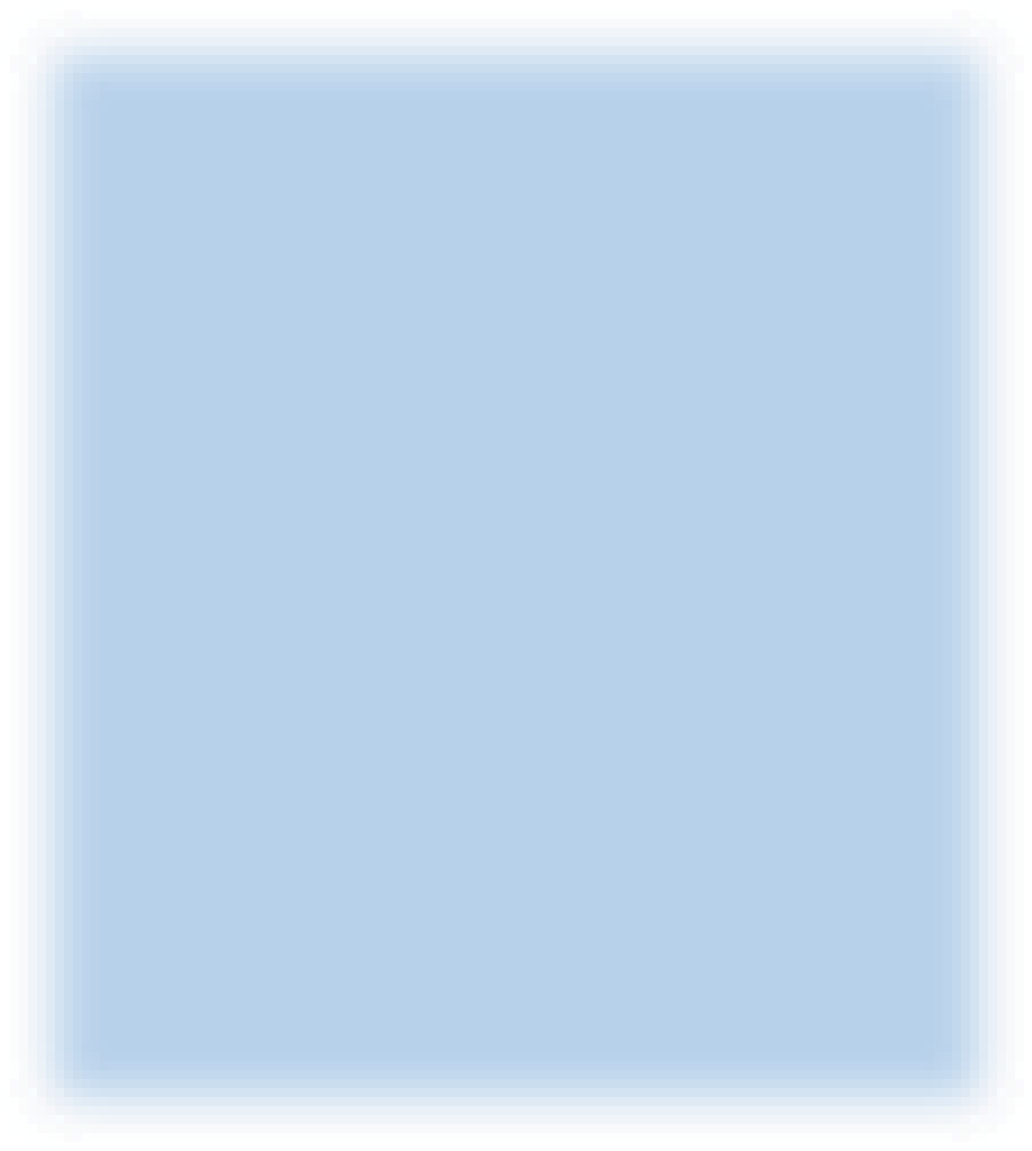
*pao.*

# Automatsko testiranje

Testiranje se najčešće provodi ručno. Problem prilikom ručnog izvođenja većeg broja testova je posebno izražen kada je potrebno ponovno testirati cijeli sustav nakon svake izmjene, što zna rezultirati time da se testiranje provodi brzo i površno. Kako bi se ubrzao i unaprijedio cijeli proces uvodi se automatizacija testiranja (provođenje funkcionalnog i regresijskog testiranja).

Funkcijskim testiranjem provjeravamo radi li implementirana funkcionalnost onako kako se od nje očekuje te se testira ponašanje aplikacije u realnom okruženju, kako bi ju krajnji korisnik koristio. Regresijsko testiranje koristi se za provjeru utjecaja izmjene na ostatak aplikacije.

Većina alata za automatizaciju traži od testera određenu razinu programerskog znanja. Mnogi alati za automatizaciju omogućuju snimanje testnih skripti gdje tester započne snimanje, ručno odradi testni slučaj, a alat snimi sve njegove korake i pretvori ih u programski kod. Na ovaj način opcijom snimanja dobivamo „kostur“ testa koji želimo napisati, a nakon toga manipulacijom i dodavanjem koda izradimo testnu skriptu.



Slika 7: Vjerojatnost pojave bugova u novoj verziji aplikacije -> potreba za automatizacijom regresijskog testiranja

Automatizacija testiranja softvera podrazumijeva korištenje alata koji omogućuju izradu testnih skripti, njihovo izvršavanje te obradu rezultata.

U ovoj vježbi koristit ćemo Firefox/Chrome plugin **Katalon Automation Recorder**.

Katalon Recorder

:

Test skripta (HTML format

)

Export test skripte: C#

(

WebDriver

+

NUnit

format

)

Visual Studio: kreiranje Unit Test Project

NUnit framework

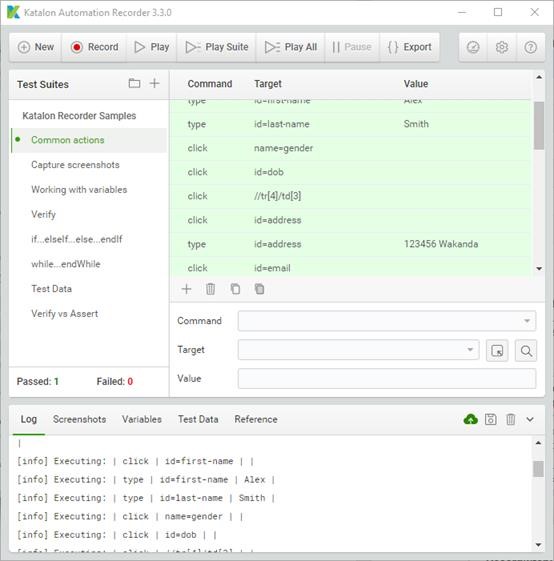
Selenium WebDriver: koristi se kao API za kontrolu browsera

Browser/Web Application (korištenje drivera IE, FF, Chrome)

Slika 8: Automatsko UI testiranje – Katalon Recorder, WebDriver, Nunit, Browser drivers

## Katalon Automation Recorder/Katalon Studio

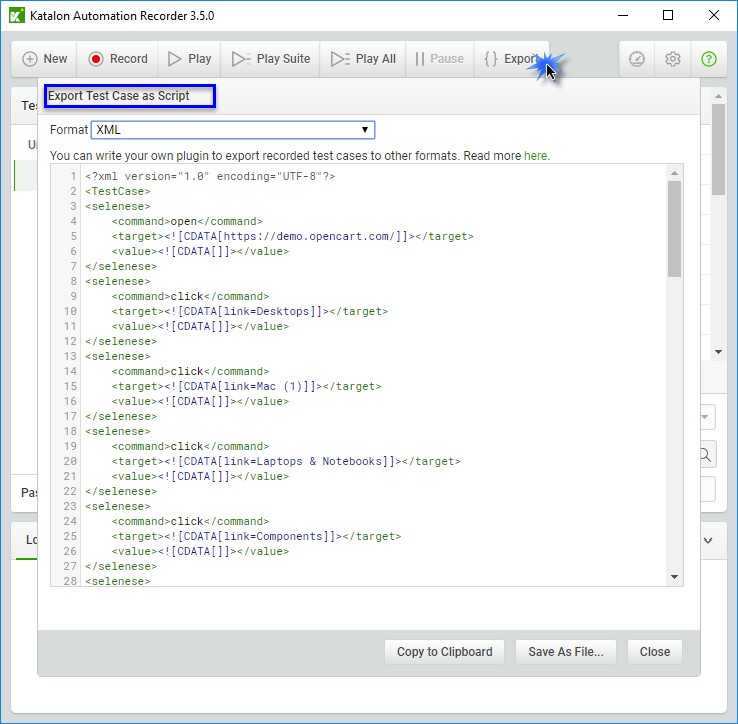
**Katalon Automation Recorder** je alat koji koristimo za kreiranje automatskih test skripti. Preteća ovog alata je Selenium IDE Firefox extension.



Glavne značajke alata:

* Dostupan je kao Firefox/Chrome browser extension ili kao desktop aplikacija Katalon Studio [(usporedba)](https://www.katalon.com/resources-center/blog/katalon-automation-recorder/)
* Snimanje testova putem *record&play* funkcionalnosti
* Snimanje UI testova koji simuliraju rad krajnjeg korisnika
* Koristan za brze korake i učenje automatizacije
* Spremanje i izvoz testa kao skripte u XML, WebDriver (C#, Java, Ruby, Python 2)
* Omogućava samostalno pisanje skripti bez korištenja snimalice ručnim unosom naredbi (sadrži izbornik s objašnjenjima najčešće korištenih Selenium naredbi)
* Jednostavnije regresijsko testiranje prilikom implementacije nove funkcionalnosti • *Scheduliranje* testova

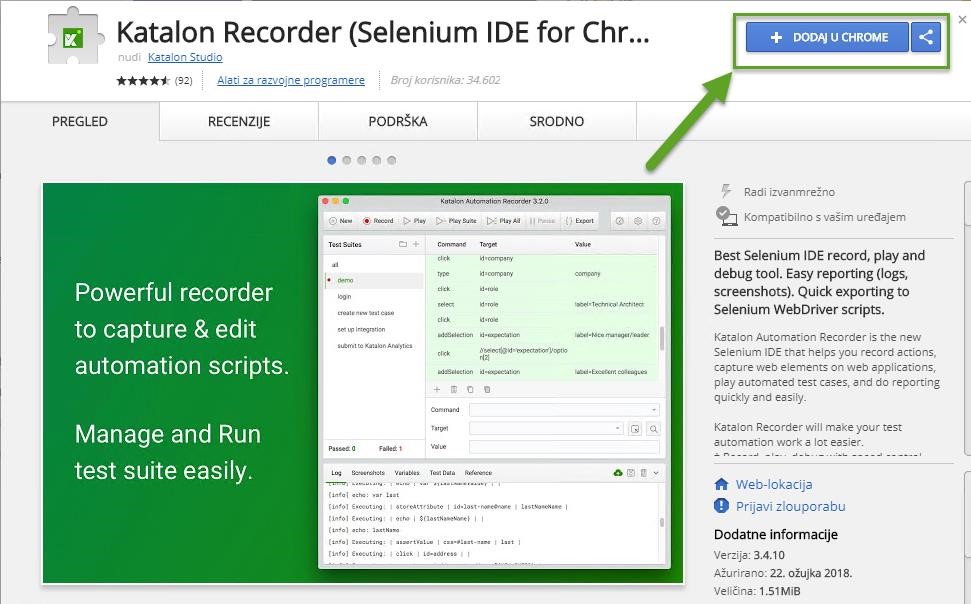
Selenium sve naredbe sprema u obliku HTML tabličnog formata.



Svaki zapis sadrži:

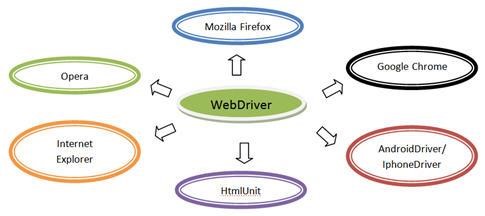
* **Command** – naredba koja će se izvršiti (npr. “open”, “type”, “click”, “verifyText”)
* **Target** – na koji HTML element se odnosi naredba (e.g. textbox, header, table)
* **Value** – koristi se za naredbe koje trebaju imati definirane vrijednosti (npr. Ispisati nešto u textbox)

2.1.1. Instalacija Katalon Recordera (Firefox/Chrome plugin)



## Selenium WebDriver

Selenium WebDriver koristi naredbe snimljene putem Katalon alata i šalje ih pregledniku. Navedeni proces je implementiran putem drivera za određeni preglednik, koji šalje naredbe preglednika, a browser vraća rezultate. WebDriver izravno pokreće instancu preglednika i upravlja njime.

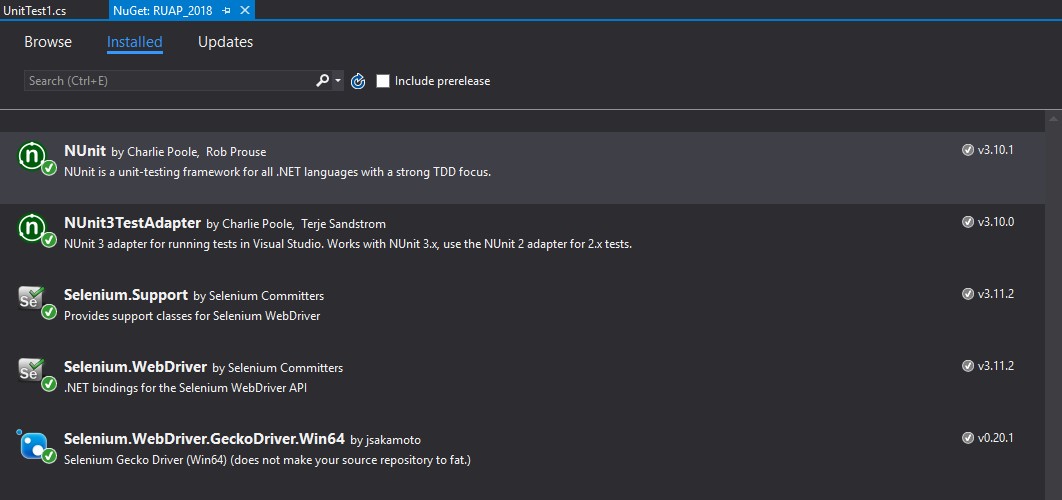


## NUnit framework

NUnit je unit testing framework za Microsoft .NET. Koristi se za unit testiranje. Samostalno ne kreira skripte za testiranje, ali omogućava pokretanje Unit testova unutar Visual Studia.

Značajke:

* Testovi se mogu pokretati kroz konzolu ili putem Test Adapter kroz VS
* Testovi se mogu paralelno izvršavati
* Svaki test se može dodati za jednu ili više kategorija, za selektivno pokretanje



Ključne riječi koje se definiraju unutar testa u Visual Studiu:

**[SetUp]** – pokretanje preglednika (FF, Chrome, IE)

**[TearDown]** – otvaranje web stranice

**[Test] –** zatvaranje preglednika

# Zadaci

Cilj ove vježbe je susresti se s praktičnim primjerom provođenja ručnog testiranja aplikacije na temelju napisanog test scenarija te snimiti skriptu za provođenje automatskog testiranja putem Firefox/Chrome plugina *Katalon Automation Recorder*. Alat omogućava izvoz skripte u nekoliko formata. Koristit ćemo izvoz u C# (WebDriver + NUNit) format za jednostavni import testa prilikom kreiranja Unit Test Projecta.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Manualno testiranje |  | Katalon Recorder |  | Selenium Webdriver |  | GIT |
| •Test analiza i test dizajn  •Definicija testnih slučajeva za demo aplikaciju | •Snimanje testova na temelju testnih slučajeva iz 1.zadatka, korištenjem  Chrome/Firefox plugina  Katalon Automation  Recorder  •export snimljene skripte u format C# (Webdriver  + NUnit) | •Kreiranje Unit Test projekta u Visual Studiu  (potreban VS Professional)  •dodavanje snimljene Katalon skripte  •NuGet paketi: NUnit framework, Test Adapter, Selenium webdriver, Gecko Driver, NUnit Console  Runner  •pokretanje testa u Test Exploreru  •testovi se izvršavaju u Firefox-u korištenjem  Gecko drivera | • kreiranje repozitorija  RUAP\_LV4  •kloniranje na lokalni disk  •commit, push testova |

Slika 9: Workflow vježbe

## Zadatak 1.

|  |
| --- |
| **Zadatak 1.** |
| Napisati test scenarij s listom test slučajeva za korištenje demo aplikacije <http://demo.opencart.com/>il[i http://demowebshop.tricentis.com/.](http://demowebshop.tricentis.com/)  Svaki test slučaj treba sadržavati detaljno opisane korake na temelju kojega će se snimiti test putem Katalon Recorder alata (zadatak 2.) Test scenarij može sadržavati npr.:   * Registracija novog korisnika (validacija podataka) * Prijava korisnika * Ažuriranje podataka korisničkog računa * Shopping chart * Estimate shipping • Provjera navigacije * Itd. |

|  |
| --- |
| **Primjer 1.** |
| |  |  | | --- | --- | | ID Testa: | 2 | | Područje funkcionalnosti: | Modul za izlistavanje moblinih uređaja | | Cilj: | Pronaći listu mobilnih uređaja i sortirati ih po cijeni od veće prema manjoj | | Rezultat testnog scenarija: | 4 testnih točki: 6 Pass, 0 Fail |        |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Test seq.**  **ID** | **Poduzeta akcija** | **Preduvjet** | **Očekivani**  **rezultati** | **Pass/Fail** | **Napomena**  **(u slučaju**  **„Fail“)** | | 1. | Upisati u preglednik link: <http://demo.opencart.com/> |  | Prilikom upisanog link-a korisniku se otvara stranica. | P |  | | 2. | U izborniku odabrati Phones and PDAs | Korisnik treba moći vidjeti listu kategorija na lijevoj strani po kategorijama. | Prilikom pritiska na Phones and PDAs otvara se prikaz svih uređaja koji su dostupni. | P |  | | 3. | Odabrati prikazivanje broja liste. | Pronaći padajući izbornik za odabir broja prikaza uređaja. | Prilikom odabira na broj izlistava se prikaz definiran odabirom broja. | P |  | | 4. | Sortiranje po cijeni | Pronaći padajući izbornik za soritranje. | Sortiranje proizvoda po cijeni od veće prema manjoj. | P |  | |

 **Pisanje test slučajeva:**

Checklist:<http://www.softwaretestinghelp.com/sample-test-cases-testing-web-desktop-applications/>

Primjer TC: <http://www.softwaretestinghelp.com/test-case-template-examples/>

## Zadatak 2.

|  |
| --- |
| **Zadatak 2.** |
| 1.. Pokrenuti Firefox ili Chrome browser i instalirati Katalon Recorder browser plugin.   1. Pokrenuti Katalon Automation Recorder. 2. Za Base URL postaviti <http://demo.opencart.com/> 3. Snimiti test putem Record za test slučajeve iz zadatka 1. 4. Nakon snimanja svih koraka reproducirati skriptu (**Play**). 5. Izvesti test u formatu C# (WebDriver+Nunit) (**Export** opcija). |

|  |
| --- |
| **Primjer 2.** |
| namespace SeleniumTests  {  [TestFixture] public class UntitledTestCase  {  private IWebDriver driver; private StringBuilder verificationErrors; private string baseURL; private bool acceptNextAlert = true;    [SetUp] public void SetupTest()  {  driver = new FirefoxDriver(); baseURL = [https://demo.opencart.com/;](https://demo.opencart.com/) verificationErrors = new StringBuilder();  }    [TearDown]  public void TeardownTest()  { try  {  driver.Quit();  }  catch (Exception)  {  // Ignore errors if unable to close the browser  }  Assert.AreEqual("", verificationErrors.ToString());  }    [Test]  public void TheUntitledTestCaseTest()  { |

driver.Navigate().GoToUrl("https://demo.opencart.com/"); driver.FindElement(By.LinkText("Desktops")).Click(); driver.FindElement(By.LinkText("Mac (1)")).Click(); driver.FindElement(By.LinkText("Laptops & Notebooks")).Click(); driver.FindElement(By.LinkText("Components")).Click(); driver.FindElement(By.LinkText("Tablets")).Click(); driver.FindElement(By.LinkText("Components")).Click(); driver.FindElement(By.LinkText("Monitors (2)")).Click(); driver.FindElement(By.XPath("//div[@id='top-links']/ul/li[2]/a/span")).Click(); driver.FindElement(By.LinkText("Register")).Click(); driver.FindElement(By.Id("input-firstname")).Click(); driver.FindElement(By.Id("input-firstname")).Clear(); driver.FindElement(By.Id("input-firstname")).SendKeys("Test"); driver.FindElement(By.Id("input-lastname")).Clear(); driver.FindElement(By.Id("input-lastname")).SendKeys("User1"); driver.FindElement(By.Id("input-email")).Clear(); driver.FindElement(By.Id("input-email")).SendKeys("test.user1@test.com"); driver.FindElement(By.Id("input-telephone")).Clear(); driver.FindElement(By.Id("input-telephone")).SendKeys("0145872"); driver.FindElement(By.Id("input-password")).Clear(); driver.FindElement(By.Id("input-password")).SendKeys("test123"); driver.FindElement(By.Id("input-confirm")).Clear(); driver.FindElement(By.Id("input-confirm")).SendKeys("test123"); driver.FindElement(By.Name("agree")).Click(); driver.FindElement(By.XPath("//input[@value='Continue']")).Click(); driver.FindElement(By.LinkText("Continue")).Click(); driver.FindElement(By.XPath("//div[@id='top-links']/ul/li[2]/a/span")).Click(); driver.FindElement(By.LinkText("Logout")).Click();

}

 Napomene:

* Katalon automation Recorder Quickstart: [https://www.katalon.com/resourcescenter/blog/katalon-automation-recorder/](https://www.katalon.com/resources-center/blog/katalon-automation-recorder/)
* Instalacija Katalon Automation Recorder:
* Firefox: [https://addons.mozilla.org/en-US/firefox/addon/katalon-automationrecord/?src=search](https://addons.mozilla.org/en-US/firefox/addon/katalon-automation-record/?src=search)
* Chrome: [https://chrome.google.com/webstore/detail/katalon-recorderselenium/ljdobmomdgdljniojadhoplhkpialdid](https://chrome.google.com/webstore/detail/katalon-recorder-selenium/ljdobmomdgdljniojadhoplhkpialdid)

## Zadatak 3.

|  |
| --- |
| **Zadatak 3.** |
| 1. Pokrenuti VS i kreirati novi projekt: **Visual C#**  **Unit Test Project** s nazivom RUAP\_LV4. 2. Putem **NuGet Packages** dodati:    * NUnit framework (3.10.1)    * Selenium WebDriver (3.11.2)    * Selenium Support(3.11.2)    * Nunit3 Test Adapter (3.10)    * Selenium WebDriver – Gecko Driver (0.20.1) – **izvođenje testa u Firefox-u** 3. Dodati test koji je prethodno eksportiran u format C# (WebDriver+Nunit) u UnitTest class. 4. Provjeriti da klasa sadrži potrebne metode:   [SetUp]  [Test]  [TearDown]     1. Dopuniti test (npr. Deklaracija random varijabli.) 2. Uključiti Test Explorer: Test-> Windows-> Test Explorer 3. Testovi su vidljivi nakon ***Build >> Build Solution***. 4. Pokrenuti testove: Run tests |

 Napomene za izvođenje zadatka 3

• Detaljni koraci su dostupni na:

[http://toolsqa.com/selenium-webdriver/c-sharp/how-to-write-selenium-test-using-nunitframework/](http://toolsqa.com/selenium-webdriver/c-sharp/how-to-write-selenium-test-using-nunit-framework/)

## Zadatak 4.

|  |
| --- |
| **Zadatak 4.** |
| 1. Na GitHub-u kreirajte javni repozitorij koji se zove RUAP-LV4 i pri tome dodajte VisualStudio .gitignore 2. Klonirajte repozitorij na lokalni disk koristeći Git command line. 3. Kreirajte branch development i prebacite se na njega. 4. Dodajte projekt koji ste kreirali za snimanje NUnit testa. 5. Napravite commit koji u naslovu ima naziv testa. 6. Pushajte sve iz lokalnog development brancha na server (origin). |

## Dodatni zadaci

|  |
| --- |
| **Zadatak 5.** |
| 1. Složiti jedan test suite s više test slučajeva (odvojeno ključnom riječi [Test]) i izvršiti sve testove. 2. Izvršavanje testova u drugim browserima (Google Chrome driver, Internet Explorer driver). |

1. Izvor: ISTQB – Certified tester, Advanced level syllabus, 2007. [↑](#footnote-ref-1)