

Instrumente de management al testelor Tehnici de automatizare

Alina Pacurar

ThoughtWorks®

Software Testing Life Cycle (STLC)

STLC defineste etapele / fazele prezente în testarea software-ului. Cu toate acestea, STLC variază practic în funcție de următoarele:

1. Capriciile Managementului
2. Software Development Life Cycle

În general, STLC cuprinde următoarele faze:



Test case, Test scenario, Test Plan

In industria software, "scenariile" de test se numesc **test cases**.

Un test case se defineste printr-un set de intrări de testare si un set de iesiri care sunt conforme cu asteptarile.

Scenariile de test reprezinta cerintele de testare grupate în functie de functionalitatea unui modul. Astfel un test scenario contine in general mai multe test cases.

Un **test plan / plan de testare** este un document care descrie domeniul de aplicare, abordarea, obiectivele, resursele, și efortul necesar pentru testare software. Identifică elementele care urmează să fie testate, elemente care nu fi testat, cine va face testarea, tipul de testare aplicat, criterii de admis / respins ale produsului, programul de testare.

Test sessions & reports

Test session - Interval de timp in care se executa o suita de teste pe o versiune a produsului software; pe parcursul unei sesiuni de testare se pot crea test cases noi, verifica defecte, raporta noi probleme/imbunatatiri.

Test session reports

- Zona testata si note detaliate cu privire la tipul de testare efectuat.
- O lista cu toate defectele gasite si intrebari deschise.
- Toate fisierile utilizate sau create pentru a sprijini testarea.
- Mentiuni in ceea ce priveste timpul petrecut pentru executarea testelor vs. timpul petrecut pentru a investiga imbunatatiri ale produsului.
- Grafic cu timpul petrecut pentru: Testarea - crearea si executarea testelor; Investigare / raportare defecte; Configurare sesiune de test sau alte activitati non-testare; Durata sesiunii de test.

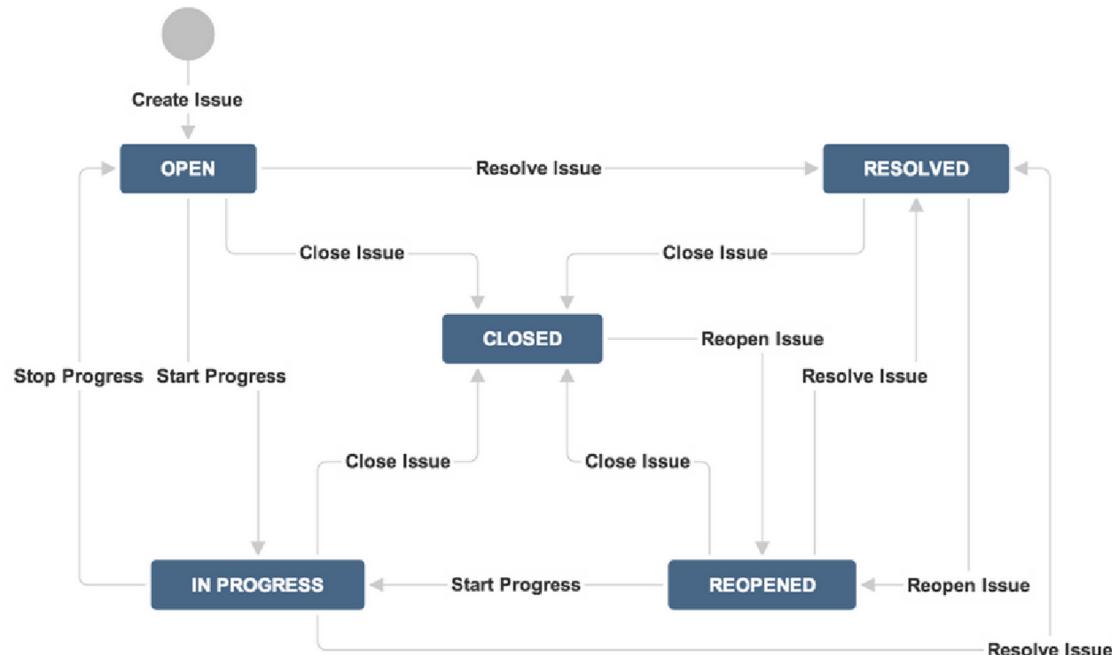
Metrici in testarea software

S.No.	Testing Metric	Data retrieved during test case development & execution
1	No. of Requirements	5
2	Avg. No. of Test cases written per Requirement	20
3	Total no. of Test cases written for all requirements	100
4	Total no. of Test cases Executed	65
5	No. of Test cases Passed	30
6	No. of Test cases Failed	26
7	No. of Test cases Blocked	9
8	No. of Test cases un executed	35
9	Total No. of Defects identified	30
10	Critical Defects count	6
11	High Defects Count	10
12	Medium Defects Count	6
13	Low Defects Count	8

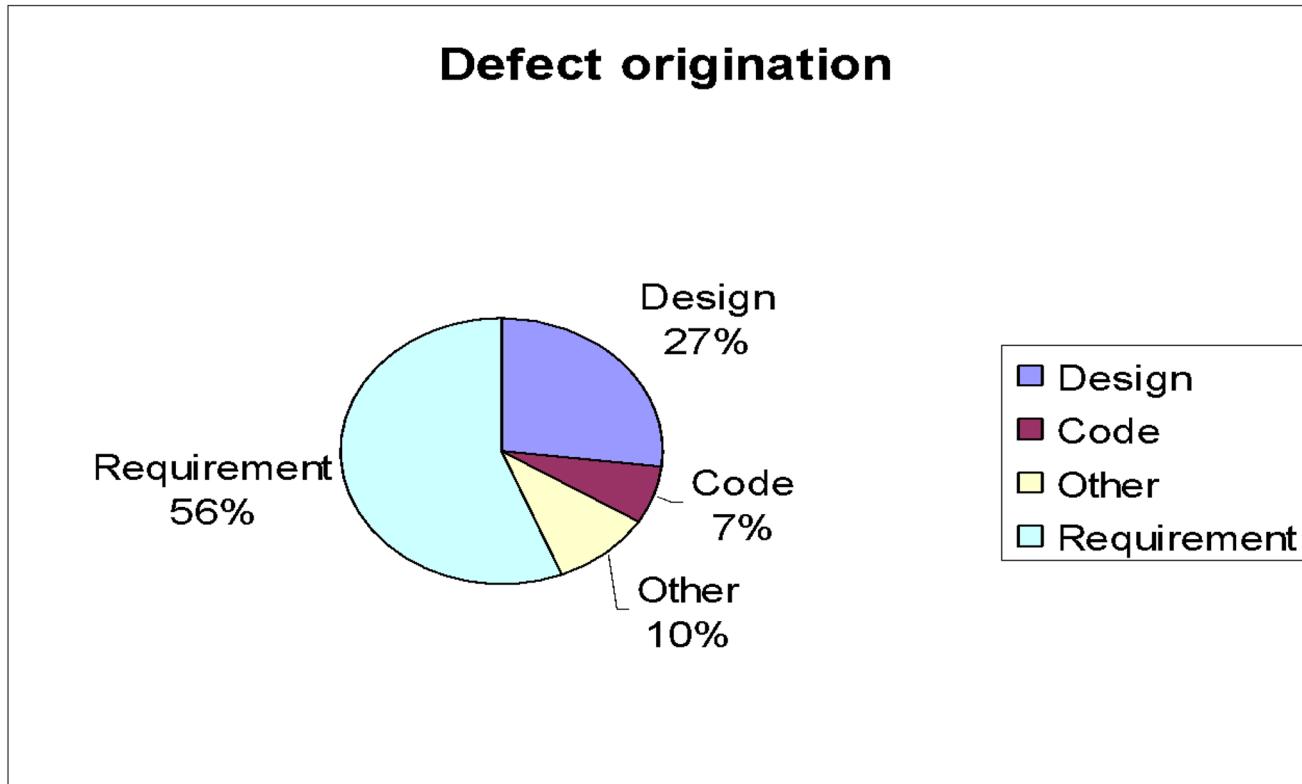
Defect/Bug Management

Defect Lifecycle – se defineste prin etapele pe care le parurge defectul din momentul in care a fost gasit pana cand a fost inchis (rejected/fixed).

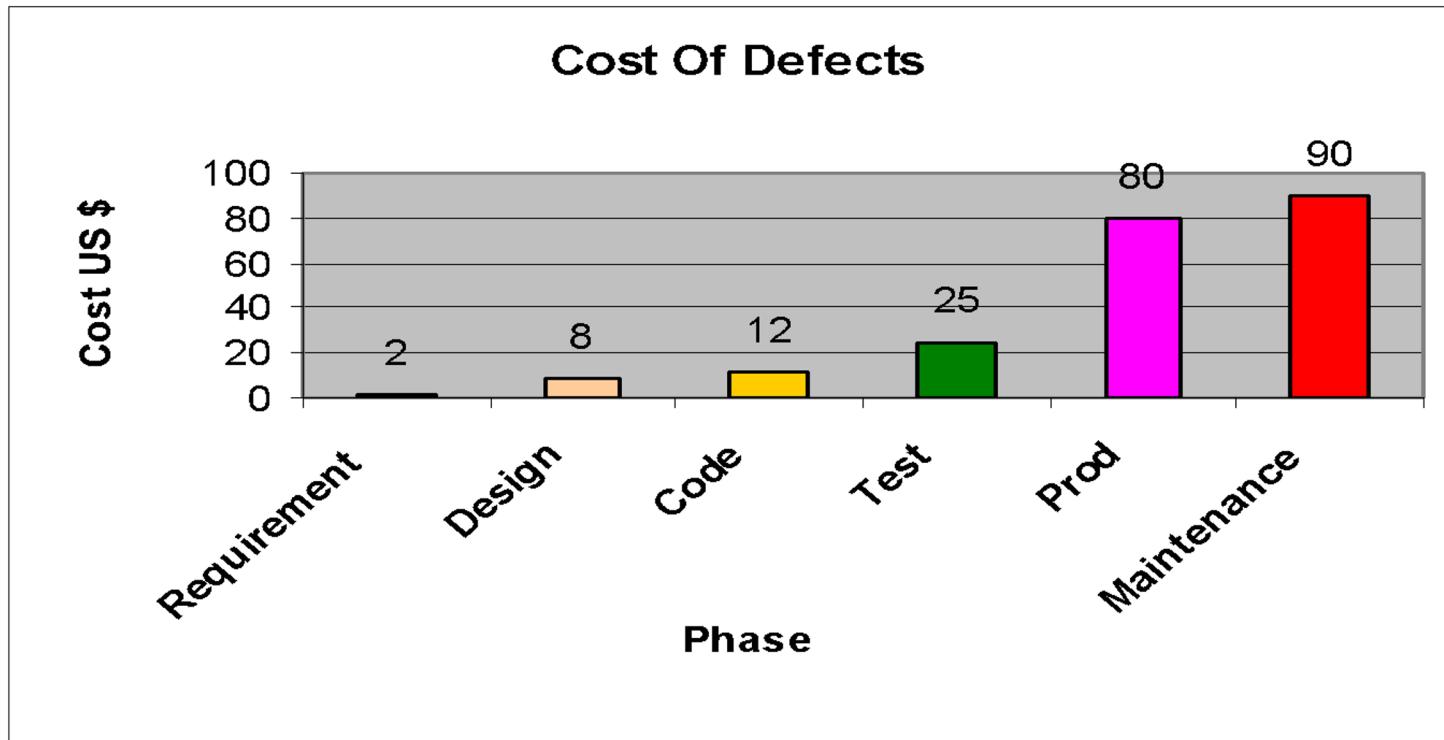
Starile defectelor:



Defects/Issues



Costuri

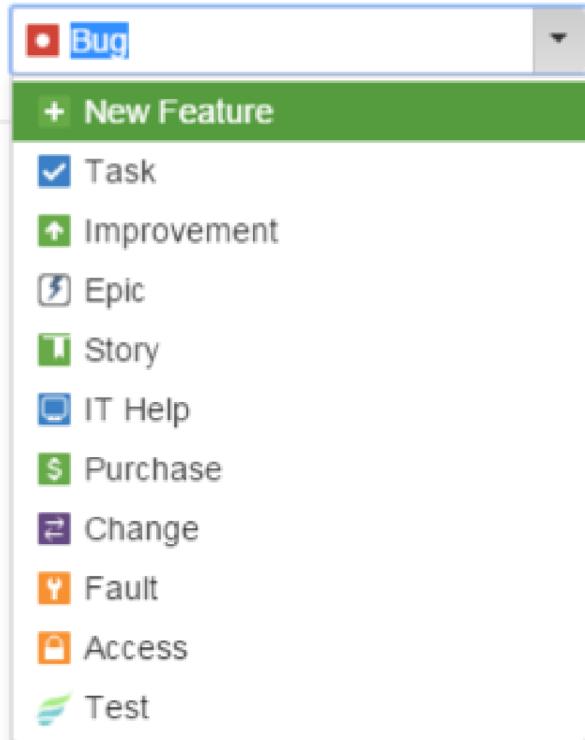


Defect/Bug Management

Varietatea este definita in functie de cerintele proiectului si de specificul acestuia

Cele mai des intalnite tipuri sunt:

- Bug
- Improvement
- new feature
- Task
- Sub-task
- Epic.



Defect/issue management tools - Exemple

JIRA - <https://www.atlassian.com/software/jira>

Bugzilla - <https://www.bugzilla.org/>

Mantis

Pivotal tracker

Redmine

Test case management tool

De ce avem nevoie de **Test Case management tool**?

- ne ajuta sa avem date despre calitatea produsului in timp real pe parcursul proiectului;
- sa fim organizati;
- sa putem refolosi teste;
- sa stim in ce directie mergem cu calitatea produsului din timp astfel incat eventuale decizii de corectare sa fie luate la momentul corect.

Test case management tool - Caracteristici

- Gestionarea printr-o interfata web a test case-urilor si a sesiunilor de testare
- Versionarea si prioritizarea testelor
- Importarea/exportarea testelor
- Asocierea testelor cu defecte, cerinte si specificatii, si generarea automata a matricii de trasabilitate a cerintelor
- Planificarea de campanii de testare si definirea atributelor importante in aceste campanii: responsabil, selectie teste care urmeaza sa fie executate din baza de date, pe ce platforme se testeaza etc
- Baza de date pentru rezultatele campaniilor de testare

Test case management tool - Caracteristici

- Suport si integrare cu instrumente de testare automata. De exemplu pentru testarea de performanta se poate programa o suita de teste automate care sa inceapa la o anumita ora si sa intoarca rezultatele direct in baza de date pentru rezultate.
- Definire de metrii de testare si includerea lor in rapoartele de testare
- Integrarea bidirectionala cu un sistem de bug tracking astfel incat atunci cand un test case este failed sa se poate deschide un bug direct din tool-ul de test management si actualiza in sistemul de bug tracking
- Integrarea cu serverul de email pentru trimiterea automata a rapoartelor si a notificarilor

Test case management tool

- Testlink
- Zephyr by Atlassian (add-on to JIRA) https://www.youtube.com/watch?v=oKi_ojyCUA
- QA Complete
- QTest by QASymphony
- Pivotal Tracker

Testarea manuala - pro & cons

PRO

- Nu este necesar niciun tool de automation pentru a testa manual un produs software
- Este cea mai primitiva metoda de testare care ajuta la gasirea bug-urilor
- Orice aplicatie se testeaza prima data manual si abia apoi se trece la testarea automata.
- Testarea manuala este necesara in a verifica fezabilitatea testelor automate
- Testarea manuala este o activitate in care testerul are nevoie de rabdare, creativitate & open mind.
- Testarea manuala se face din prisma utilizatorului final

Testarea manuala - pro & cons

CONS

- Nu toate scenariile de test pot fi executate manual
- Load si performance testing nu pot fi rulate manual
- Nu se potriveste proiectelor mari cu deadline strans.

Testarea automata - pro & cons

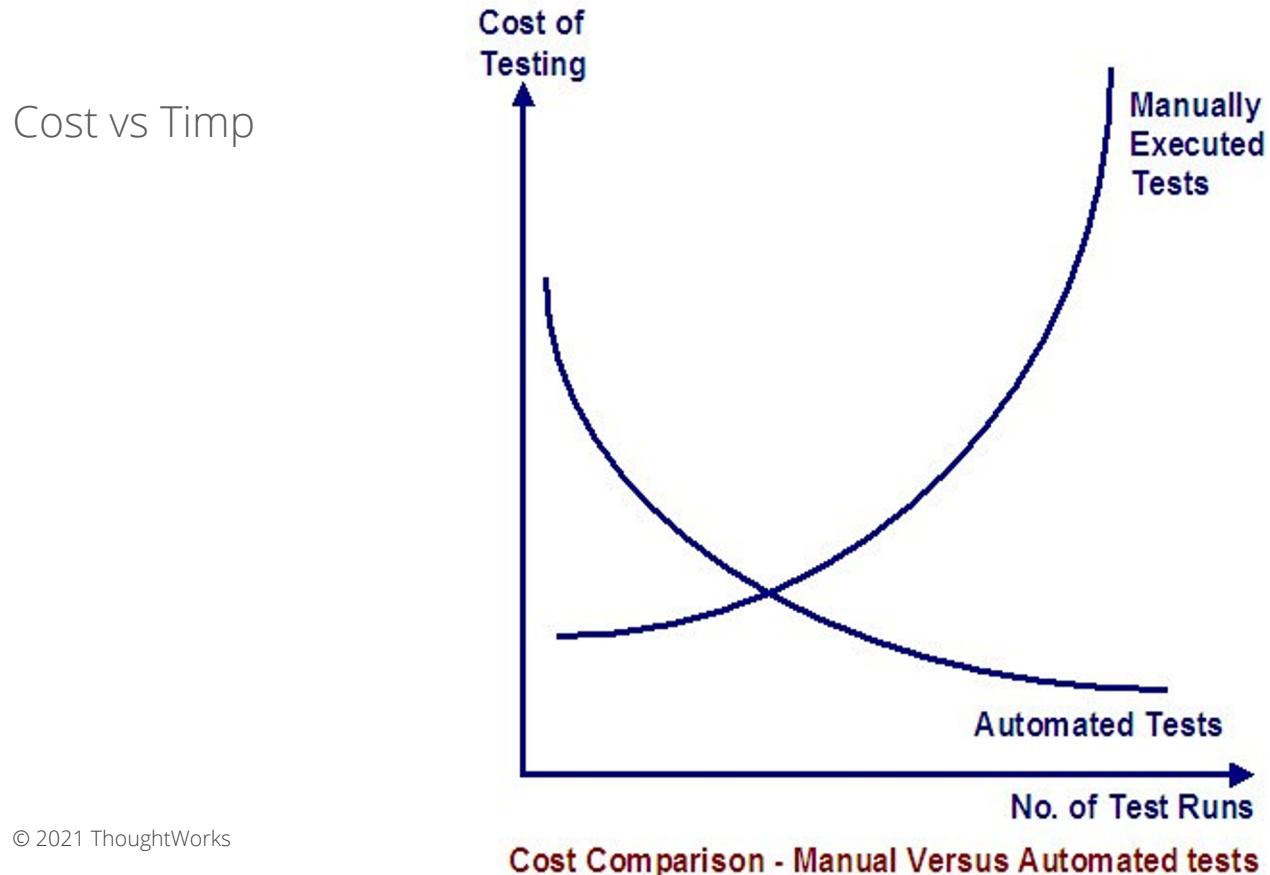
PRO

- Timp scurt de rulare a unui numar mare de teste
- Se evita eroarea umana care apare in procesul de testare manuala

CONS

- Testarea automata necesita investitii substantiale in tool-uri si skills
- Necesita timp indelungat pentru scrierea testelor
- Nu poate inlocui abilitatile intelectuale ale omului in evaluarea UI si UX a unei aplicatii
- Nu poate fi folosita in exploratory testing
- Necesita cunostinte basic de programare

Testarea manuala vs Testarea automata



Automatizarea testarii

Bune practici

- Documentarea codului.
- Identificarea metodelor refolosibile si scrierea lor intr-un fisier separat.
- Follow the language-specific coding conventions.
- Mentinerea datelor de test intr-un fisier separat.
- Rularea testelor la intervale prestabilite de timp.

Automatizarea testarii

Caracteristicile aplicatiilor din prisma testarii automate

Proiectul software supus testării trebuie sa se afle într-o stare relativ stabila, cu alte cuvinte sa se afle mai degraba într-o fază de dezvoltare de noi funcționalități, sau chiar în fază de întreținere.

Initierea testării automate în fazele initiale de dezvoltare, înainte ca proiectul software să fie disponibil unor utilizatori finali, este mai degraba o abordare riscanta, având sanse mari de anulare și rescriere a testelor sau chiar de anulare totală a soluțiilor de testare automate în care s-au investit timp și bani.

Automatizarea testarii - tools

Cum aleg cel mai bun tool de automatizare pentru aplicatia mea?



Selenium, RobotFramework



TestComplete, Soap UI



Built in house tools

Product	 Selenium	 Katalon Studio	 Unified Functional Testing	 TestComplete	 SoapUI
Available since	2004	2015	1998	1999	2005
Application Under Test	Web apps	Web (UI & API), Mobile apps	Web (UI & API), Mobile, Desktop, Packaged apps	Web (UI & API), Mobile, Desktop apps	API/Web service
Pricing	Free	Free	\$\$\$\$	\$\$\$	\$\$
Supported Platforms	Windows Linux OS X	Windows Linux OS X	Windows	Windows	Windows Linux OS X
Scripting languages	Java, C#, Perl, Python, JavaScript, Ruby, PHP	Java/Groovy	VBScript	JavaScript, Python, VBScript, JScript, Delphi, C++ and C#	Groovy
Programming skills	Advanced skills needed to integrate various tools	Not required. Recommended for advanced test scripts	Not required. Recommended for advanced test scripts	Not required. Recommended for advanced test scripts	Not required. Recommended for advanced test scripts
Ease of Installation and Use	Require advanced skills to install and use	Easy to setup and use	Complex in installation. Need training to properly use the tool	Easy to setup. Need training to properly use the tool	Easy to setup and use

Introducere in Framework-ul Selenium

Framework? = guideline (not mandatory rules)

Selenium este un Framework portabil de testare al aplicatiilor web, oferind posibilitatea de a scrie teste automate intr-o multitudine de limbaje (C#, Groovy, Java, Perl, PHP, Python, Ruby si Scala)

- Selenium a aparut in 2004;
- 2007 - Selenium Remote Control (RC)
- 2008 - Selenium Grid
- 2009 - Selenium WebDriver (Selenium 2.0)

Are suport pentru o mare parte din browserele cele mai folosite (Chrome, Firefox, IE, Edge, Safari)

Introducere in Framework-ul Selenium

De ce Selenium?

Este free and open source

- Are un numar mare de utilizatori si o comunitate mare de Help
- Are cross Browser compatibility (Firefox, Chrome, Internet Explorer, Safari etc.)
- Platform compatibility (Windows, Mac OS, Linux etc.)
- Suporta multiple limbaje de programare (Java, C#, Ruby, Python, Pearl etc.)
- Este intr-o continua dezvoltare

Introducere in Framework-ul Selenium

Componente:

- Selenium IDE - record and playback tool. Este distribuit ca un plugin pentru Firefox si Chrome.
- Selenium Remote Control (RC) – este un server care permite utilizatorilor sa scrie teste in limbajul de programare dorit si rularea lor pe o arie mare de browsere.
- Selenium WebDriver - are diverse avantaje fata de RC printre care este capacitatea de a comunica direct cu browserul. Foloseste propria capacitate de a automatiza.
- Selenium Grid - este folosit pentru a distribui concurrent executia testelor pe mai multe platforme si sisteme.

Introducere in Framework-ul Selenium

Limitari:

- Selenium poate fi folosit doar in testarea aplicatiilor web-based
- Aplicatiile mobile nu pot fi testate folosind Selenium
- Captcha si Barcode readers nu pot fi testate folosind Selenium
- Rapoartele pot fi generate folosind third-party tools (TestNG sau JUnit).
- Utilizatorul trebuie sa aiba cunostinte basic de programare.

Introducere in Framework-ul Selenium

Structura framework-ului

- Un folder “src” (source) care contine test scripts.
- Un folder “lib” (library) care contine toate librariile si metodele.
- Un folder “class” care contine toate class files (in-case of java).
- Un folder “log” care contine toate fisierelor cu logurile.
- Un fisier/folder care contine toate web element Ids.
- Un fisier care contine URL, environment si login information. (de ce sa le mentionem intr-un fisier separat? URL, Login, si passwords sunt campuri folosite foarte des si se pot modifica la fel de des. In cazul in care le bagam in codul nostru ar deveni destul de greu de facut update de fiecare data atat timp cat ele nu sunt folosite doar intr-un singur fisier.)

Arhitectura Selenium WebDriver

1. Selenium Client Library - suporta mai multe librarii cum ar fi Java, Ruby, Python... pentru a suporta mai multe limbaje de programare.
2. JSON (JavaScript Object Notation) Wire Protocol over HTTP - este folosit pentru a transfera date intre client si server in cadrul aplicatiilor web. Comunicarea intre servere HTTP se face folosind REST API. Fiecare browser driver are propriul server HTTP.
3. Browser Drivers - fiecare browser are propriul driver. Driverul comunica cu browserul fara a fi nevoie sa cunoasca logica de functionare a browserului. Cand driverul primeste o comanda, atunci acea comanda este executata pe browserul specific iar raspunsul se returneaza sub forma de HTTP response.
4. Browsers - Chrome, Firefox, IE, Safari...

Configurare Selenium WebDriver & Practice

1. New Project
2. Add Selenium JARs to Eclipse Project
 - Right click on the Project, select **Build Path > Configure Build Path...**
 - click on **Libraries** tab
 - Click on **Add External JARs** button. Select all JAR files from the path **..\lib**
3. Setup browser webdrivers



<https://seleniumhq.github.io/selenium/docs/api/java/>

Chrome webdriver - există două metode de configurare.

- **Method 1:** Use `webdriver.chrome.driver` system property

Prin intermediul acestei metode trebuie ca de fiecare dată să adăugăm o linie de cod în testele tale. Copiază calea unde ai dezarchivat chrome webdriver, de exemplu – D:\Drivers\chromedriver.exe. Trebuie adăugat **System.setProperty** având calea unde este salvat driverul.

```
System.setProperty("webdriver.chrome.driver", "D:\\Drivers\\chromedriver.exe");
WebDriver driver = new ChromeDriver();
driver.get("http://www.google.com");
```

Configurare Selenium WebDriver & Practice

- **Method 2:** Set property in Environment Variables

```
WebDriver driver = new ChromeDriver(options);
driver.get("http://www.google.com");
```

Firefox webdriver

Necesar:

- Firefox browser
- Geckodriver (32/64 bit)

Gecko - este un web browser engine folosit in multe dintre aplicatiile dezvoltate de Mozilla

Geckodriver - este un proxy pentru a ajuta clientii compatibili cu W3C WebDriver sa interactioneze cu Gecko-based browsers (FF)

Se folosesc Chrome options pentru a:

- Disable infobars
- Start browser maximized
- Run chrome headless (starting Selenium WebDriver 3.6) - `options.setHeadless(true);`

Configurare Selenium WebDriver & Practice

1. Download Geckodriver (32/64 bit)
2. Configurare:
 - **Metoda 1** - *webdriver.gecko.driver system property*

```
System.setProperty("webdriver.gecko.driver","D:\\Firefox\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.get("http://www.google.com");
```

- **Metoda 2** - Set property in Environment Variables

Configurare Selenium WebDriver & Practice

ChromeOptions & FirefoxOptions

- Run browser maximized
- Headless browser
- Disable infobars

```
ChromeOptions options = new ChromeOptions();
options.addArguments("disable-infobars");
options.addArguments("--start-maximized");

WebDriver driver = new ChromeDriver(options);
driver.get("http://www.google.com");
```

```
ChromeOptions options = new ChromeOptions();
options.setHeadless(true);

//Instantiate Web Driver
WebDriver driver = new ChromeDriver(options);
driver.get("http://www.google.com");
System.out.println("Page title is - " + driver.getTitle());
```

Configurare Selenium WebDriver & Practice

Edge Webdriver

1. Valabil doar incepand cu Windows 10
2. Vezi build-ul sistemului de operare (**Start > Settings > System > About**)
3. Download Microsoft Edge WebDriver <https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/> asigurandu-te ca folosesti versiunea de release corespunzatoare buildului sistemului de operare
4. Instaleaza Microsoft Edge Web Driver - MicrosoftWebDriver.exe este localizat in ProgramFiles> Microsoft Web Driver
5. Configurare:

Metoda 1 - webdriver.edge.driver system property

Metoda 2 - Set property in Environment Variables

```
System.setProperty("webdriver.edge.driver","C:\\Program Files (x86)\\Microsoft Web Driver\\MicrosoftWebDriver.exe");
WebDriver driver = new EdgeDriver();
driver.get("http://www.google.com");
```

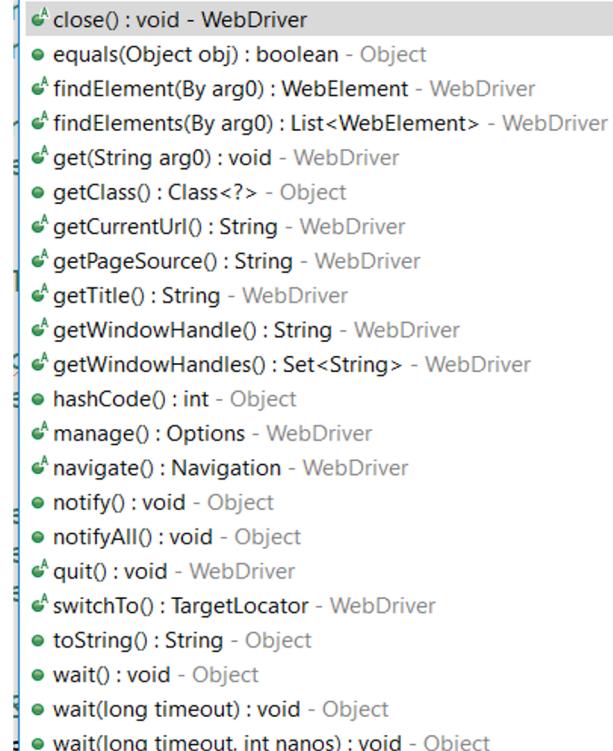
```
WebDriver driver = new EdgeDriver();
driver.get("http://www.google.com");
```

Configurare Selenium WebDriver & Practice

Selenium WebDriver Methods

`driver.quit()` - folosit atunci cand lucrez cu child windows.

`driver.close()` - folosit cand lucrez pe aplicatii single webpage, single browser.



```
close() : void - WebDriver
equals(Object obj) : boolean - Object
findElement(By arg0) : WebElement - WebDriver
findElements(By arg0) : List<WebElement> - WebDriver
get(String arg0) : void - WebDriver
getClass() : Class<?> - Object
getCurrentUrl() : String - WebDriver
getPageSource() : String - WebDriver
getTitle() : String - WebDriver
getWindowHandle() : String - WebDriver
getWindowHandles() : Set<String> - WebDriver
hashCode() : int - Object
manage() : Options - WebDriver
navigate() : Navigation - WebDriver
notify() : void - Object
notifyAll() : void - Object
quit() : void - WebDriver
switchTo() : TargetLocator - WebDriver
toString() : String - Object
wait() : void - Object
wait(long timeout) : void - Object
wait(long timeout, int nanos) : void - Object
```

Selenium WebDriver - Locators

Selenium foloseste ceea ce se numeste “locators” pentru a identifica elementele in pagina cu care trebuie sa interactionezi. Cei mai folositi locatori sunt:

1. ID
2. Class Name
3. Name
4. LinkText
5. Tag Name
6. XPath
7. CSS

Selenium WebDriver - Locators

Exista diverse modalitati de a gasi/compune si verifica **locators** printre elementele HTML ale unei pagini web:

- Firebug/ firepath/ cropath
- Developer tools
- Regular expressions

Selenium scanarea elementelor din pagina incepand cu stanga sus.

Astfel daca exista elemente care au aceeasi valoare, el o sa-l ia pe primul identificat si le ignora pe celelalte din pagina. Trebuie sa ne asiguram ca in identificarea elementelor folosim atributele care sunt unice.

Tehnici pentru locatori

1. Locators by ID

`driver.findElement(By.id("elementId"));`

Avantaje & dezavantaje in folosirea ID:

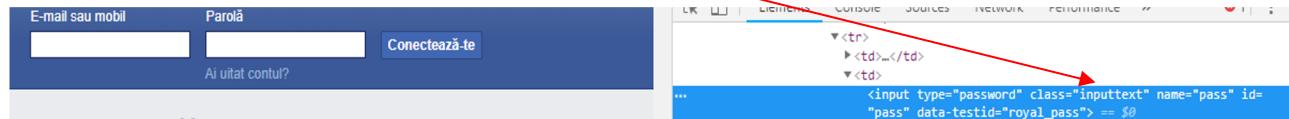
- nu este vizibil in toate paginile web.
- poate fi dinamic si se schimba la reload de pagina (atunci cand ID este alfanumeric), astfel se recomanda folosirea altor locatori in schimbul ID-ului.
- este foarte simplu de detectat in DOM-ul paginii web.

2. Locators by class Name

`driver.findElement(By.className("elementsClass"));` (www.facebook.com) exemplu de atribute care nu sunt unice - vezi class in form de login)

Avantaje & dezavantaje in folosirea Class Name:

- className se poate schimba atunci cand developerul schimba numele clasei;
- Nu toate elementele web au className
- Este foarte simplu de identificat;
- Nu accepta spatii in className;**



Tehnici pentru locatori

3. Locators by Name - foloseste atributul name pentru a identifica elementul web:

- Este de preferat utilizarea lui atunci cand exista o lista de elemente similare;
- Nu este recomandata utilizarea lui in cazul listelor generate dinamic;

driver.findElement(By.name("name")); (Test1.java)

//Search on Google

- driver.findElement(By.name("q")).sendKeys("selenium webdriver");
- driver.findElement(By.name("q")).sendKeys(Keys.ENTER);

3. Locators by LinkText - localizeaza elementul de tip link folosind textul acestuia

- Se foloseste in cazul tag-urilor de tip anchor a;
- Este de preferat in cazul in care se doreste verificarea navigation flows;
- Este nevoie sa-i link text pentru a-l putea folosi;

- **driver.findElement(By.linkText("Click Here")); (Test1.java)**
- **driver.findElement(By.partialLinkText("Click"));**

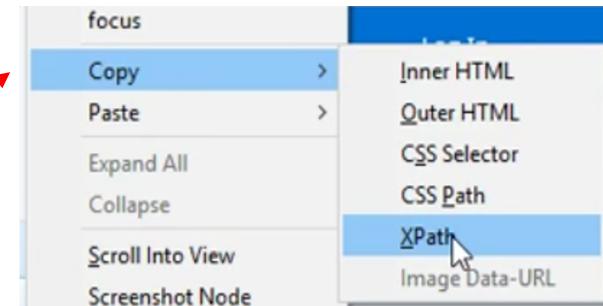
3. Locators by TagName - tag-urile HTML (div, a, input...) sunt folosite pentru localizarea elementelor web:

driver.findElement(By.tagName("a"));

Tehnici pentru locatori

6. Locators by XPath (facebook.java)

- Unul dintre cei mai intalniti locatori in identificarea elementelor web dintr-o pagina.
- Adresa unica pentru fiecare element web;
- Poate fi generat din browser tools - 99% accurate - inspect > right click > copy XPath;
- Customized XPath locators - de recomandat;
- Pentru un singur obiect se poate scrie xpath in n moduri.
- Nu exista nicio regula care sa spuna ca exista doar un singur xpath pentru un obiect
- Xpath poate sa difere de la un browser la altul pentru acelasi obiect;
- A se ignora xpath care incepe cu /html/
- Este de preferat sa se mentioneze tagName in customized xpath;
- Este mai lent decat CSS selectors;



XPATH general formula - //tagName[@attribute='value']

```
driver.findElement(By.xpath("//*[@id='Login']")).click();
```

Se poate construi xpath folosind tehnica [xpath_parent] [xpath child] atunci cand exista mai multe instante pentru acelasi atribut

Construieste Xpath folosind regular expressions:

```
//tagName[contains(@attribute,'value')]
```

```
By.xpath("//div[@id='glsctl00_mainContent_ddl_destinationStation1_CTNR'] //a[@value='MAA']).click();
```

Ex //input[contains(@name,'username')]

Exercitiu → folositi xpath in login form-ul emag

Tehnici pentru locatori

7. Locators by CSS selector (facebook.java)

- Unul dintre cei mai intalniti locatori in identificarea elementelor web dintr-o pagina.
- Este mai rapid decat xPath
- Utilizarea sa creste, pe masura ce paginile web devin din ce în ce mai orientate spre stil.
- Este usor sa definiți un localizator CSS unic, deoarece puteti combina mai multe atribute CSS.
- Poate fi definit in diferite moduri
- Exista cazuri in care CSS nu este identificat in Chrome browser, dar poate fi gasit in toolbar
- Nu este usor sa formezi un selector CSS si necesita o intelegerie mai profunda a CSS / Javascript.

```
driver.findElement(By.cssSelector(".input.identityinput")).sendKeys("test2");
```

CSS general formulas:

```
driver.findElement(By.cssSelector("input.inputtext")).sendKeys("test");
driver.findElement(By.cssSelector("input[name='pass']")).sendKeys("testpass");
driver.findElement(By.cssSelector("[value='Log In']")).click();
```

- tagname[attribute='value']
- tagname#id (sau #id)
- Tagname.classname - tagname poate fi eliminat pentru classname scurt. Daca numele clasei contine spaces acesta se inlocuieste cu .

Construieste CSS selectors folosind regular expressions:

//tagName[attribute*='value']

Ex: input[name*='username']

Doar prin CSS selectors se poate ic

```
driver.findElement(By.cssSelector("#username")).sendKeys("alina");
```

```
driver.findElement(By.cssSelector("input[id='username']")).sendKeys("alina");
driver.findElement(By.cssSelector(".standard_logo")).click();
```

Exercitiu → folositi CSS selectors in login form-ul Emag

Pentru ID se foloseste #
Pentru clase se foloseste .

Tehnici pentru locatori

Modalitati de validare & debug a Xpath si CSS selectors:

- Se pot valida folosind browser console;
- Din cauza protectiei pentru sql injections, este necesar sa rulezi "allow" in consola browserului cand doresti sa validezi un locator
- Returneaza null in cazul in care xpath este invalid, sau error daca CSS este invalid.

Validare XPATH:

```
$x("xpath")      $x("//*[@id='forgot_password_link'])
```

Validare CSS:

```
$(“CSS_Selector”)      $( "#mydomainLink")
```

Exercitiu - valideaza daca mesajul de eroare este afisat sau nu in pagina pe un failed login in login.salesforce.com

- Get URL;
- Type username & password - sendKeys
- Click login button - click()
- Get error message - getText()

```
System.setProperty("webdriver.chrome.driver", "C://work//chromedriver.exe");

WebDriver driver =new ChromeDriver();

driver.get("https://login.salesforce.com/");
driver.findElement(By.id("username")).sendKeys("hello");
driver.findElement(By.name("pw")).sendKeys("123456");
//driver.findElement(By.className("button r4 wide primary")).click(); //Error
driver.findElement(By.xpath("//*[@id='Login']")).click();
System.out.println(driver.findElement(By.cssSelector("div#error.loginError")).getText());
```

Tehnici pentru locatori

Exemplu - facebook login form pentru CSS selector si Xpath

```
public static void main(String[] args) {
    // TODO Auto-generated method stub
    System.setProperty("webdriver.chrome.driver", "C://work//chromedriver.exe");

    WebDriver driver =new ChromeDriver();
    driver.get("http://facebook.com");

    ////tagName[@attribute='value'] - xpath
    driver.findElement(By.xpath("//*[@type='email']")).sendKeys("myown xpath");
    driver.findElement(By.xpath("//input[@id='pass']")).sendKeys("hello");
    driver.findElement(By.xpath("//input[@value='Log In']")).click();*/

    //tagName[v='value'] -CSS
    driver.findElement(By.cssSelector("input[name='email']")).sendKeys("myowncss");
    driver.findElement(By.cssSelector("[value='Log In']")).click();
```

Tehnici pentru locatori

XPath → diferența dintre calea relativă și calea absolută

Definitia XPath → XPath este o tehnica perfecta pentru navigarea prin structura DOM a paginii web. Detectoarele XPath sunt robuste si fiabile. Este o metoda care garanteaza localizarea oricarui element pe pagina utilizand expresia XPath. Dar ar trebui sa fii foarte atent în timp ce formezi un XPath, deoarece nu poate functiona daca exista modificari in aplicatia web.

XPath relativ → XPath-urile relative sunt usor de gestionat deoarece sunt scurte si concise, nu depind de nodul parinte. Poate supravietui schimbarilor din HTML-ul paginii într-o anumita masura. Construirea unui XPath relativ consuma mult timp si este destul de dificila deoarece trebuie sa verifice toate nodurile pentru a forma calea.

XPath absolut → Aceasta porneste de la elementul radacina din pagina web sau dintr-o parte a paginii si merge la identificarea elementului tinta. Pentru a folosi locatori precum XPath este usor cand dai calea elementului direct. Dar XPath s-ar rupe cand structura elementelor se schimba.

Cum identific un obiect pe baza unui text?

Formula generala → `//*[text()='text_value']`

- Nu poate fi folosit in cazul in care textul este lung;
- Este de evitat deoarece textul poate fi schimbat;

XPath: `//section/div/div/div/div/ul/li[2]`

Tehnici pentru locatori

Constructia XPath folosind relatia parinte → copil

- Se foloseste in cazul in care nu exista atribute unice sau statice;
- Se defineste prima data xpath-ul parintelui care are atribut unic, iar traversarea catre copil se face prin /

General formula → //tagName[@attribute='value']/tagname/tagname/tagname...

```
//div[@class='lst-c']/div/div[2]/div/input
```

Constructia XPath folosind relatia dintre nodurile care au acelasi parinte (frati)

Cum navighez intre frati? De exemplu am un XPath definit pentru primul frate - intrebarea este cum ajung la al doilea frate?

- Aceasta tehnica se foloseste in general cand atributele celorlalți frati nu sunt statice

Exercitiu → <https://www.w3schools.com>

Constructia XPath navigand de la copil la parinte

Se compune xpath pentru copil după care se traversează înapoi către parinte, menționând tag-ul parintelui

Doar în XPATH pot să traverseze înapoi în ierarhia DOM a paginii web!!!

```
driver.findElement(By.xpath("//ul[@class='responsive-tabs__list']/li[1]/following-sibling::li[2]")).click();
```

```
driver.findElement(By.xpath(".//*[@id='tablist1-tab2']/parent::ul")).getAttribute("role"));
```

Web UI controls

1. Static dropdowns

- inspect element - tagname <select>;
- Se utilizeaza clasa **Select** din JAVA;
- Acest tip de web element este destul de putin utilizat in ultima perioada;

```
Select s= new Select(driver.findElement(By.id("id_value")));
s.selectByValue("3");
```

Exercitiu - dropdown looping

www.wizzair.com → alege mai multi pasageri pentru care doresti sa cumperi bilete de avion. Returneaza in consola numarul exact de pasageri pe categorii

1. Dynamic dropdowns

- Def - se populeaza doar daca exista conditii initiale
- In general, daca exista 2 elemente cu acelasi xpath, trebuie folosita urmatoarea sintaxa (xpath)/[index]

Exercitiu - dynamic dropdown using indexes - alege doua locatii diferite pentru Oras plecare si Oras destinatie (www.wizzair.com)

Cum as putea sa evit folosirea de indexes in scrierea testelor automate? (P->C)

- Solutia este sa folosesti sintaxa [parent xpath] [child xpath]

Auto suggestive dropdowns: lista de items se restrange in functie de stringul tiparit in text box.

Exercitiu - a se replica fiecare actiune din www.vola.ro - ENTER key, UP/DOWN arrow keys...

```
driver.findElement(By.id("ctl00_mainContent_ddl_originStation1_CTXT")).click();
driver.findElement(By.xpath("//a[@value='BLR']")).click();
Thread.sleep(2000);
driver.findElement(By.xpath("//a[@value='MAA'][2]")).click();
```

Web UI controls

3. Checkboxes

- isSelected - validez daca un checkbox este selectat sau nu;
- Cum returnez cate checkboxes sunt intr-o anumita zona a paginii web? (**findElements(By.cssSelector("input[type='checkbox']"))size()**)

3. Radio buttons

- findElements pentru a afla cate radio buttons sunt pe o pagina (count());
- Cum pot sa afisez in consola atributele unui radio button?
- Cum pot sa fac click pe al doilea radio button fara sa stiu ID sau value?

3. Text buttons

- //button[contains(text(), 'Log in')];

3. Alerts java popups

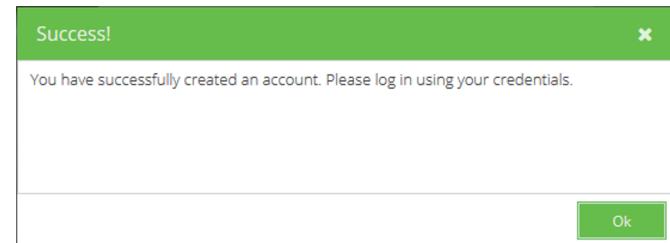
- Nu contine cod HTML, este un JAVA popup. Selenium handles doar HTML web elements. Cum pot sa automatizez asemenea elemente?
- Ex: https://www.jquery-az.com/javascript/demo.php?ex=151.0_1
- Daca popup este HTML code, atunci se folosesc metodele clasice de identificare a elementelor web.
- driver.switchTo().alert().accept()
- driver.switchTo().alert().dismiss()
- Verifica daca textul prezent in popup este cel expected.
- Selecteaza un radio button/checkbox in popup.
- Tipareste text in popup.....

3. Selenium webdriver form methods

Exercitiu practic - registration form gmail.com

3. Calendar

Exemplu - www.tarom.ro Cum validez daca un web element este enabled sau disabled?



Handling AJAX / mouse interactions

- Cum pot sa fac mouse over pe un element folosind Selenium?
- Cum pot sa am keyboard interactions in Selenium?

API-ul Advanced User Interactions al Selenium WebDriver ne permite sa efectuam operatiuni de la evenimentele de la tastatura si evenimentele simple ale mouse-ului la evenimente complexe, cum ar fi drag & drop, select text.....

Aduce un plus in simularea actiunilor utilizatorului.

Exemplu: www.amazon.com (actions.java)

```
//click pe search bar -> scriu un string cu majuscule  
a.moveToElement(driver.findElement(By.id("twotabsearchtextbox"))).click().keyDown(Keys.SHIFT).sendKeys("test").build().perform();
```

Handling multiple windows - child windows

Cum reușesc sa manipulez child windows?

Selenium identifica doar elementele din parent window. Pentru a identifica elementele din child window

GetWindowHandles(); -> Set data structure

switchTo().window(args)

Exemplu → multipleWindows.java

```
//handle multiple windows
Set<String>ids=driver.getWindowHandles();

//iterate multiple windows in set
Iterator<String> it=ids.iterator();
String parentid = it.next();
String childid=it.next();
driver.switchTo().window(childid);
System.out.println(driver.getTitle());

//switch to parent window
driver.switchTo().window(parentid);
System.out.println(driver.getTitle());|
```

Handling frames

Frames - componentă gazduite de paginile web. Are propriul container și este independent de codul HTML al paginii web.

Selenium nu poate identifica aceste elemente direct.

Exemplu → <http://jqueryui.com/droppable/>

Pasi:

1. driver.switchTo().frame(webElement) Identificare frame în pagina bazându-mă pe findElement
2. După identificarea frame-ului putem să folosim elementele din interiorul lui. Recomandarea este de a folosi clasa Actions.

```
//switch to frames
driver.switchTo().frame(driver.findElement(By.cssSelector("iframe[class='demo-frame']")));
driver.findElement(By.id("draggable")).click();

WebElement source = driver.findElement(By.id("draggable"));
WebElement target = driver.findElement(By.id("droppable"));

//drag & drop
Actions a = new Actions(driver);
a.dragAndDrop(source, target).build().perform();
```

driver.switchTo().defaultContent(); → spunem driverului să iasă din frame. Switch către default content

Capture screenshot and email results

Captarea pozelor de ecran semnifica enorm de mult in testarea automata atunci cand se verifica rezultatele testelor.

getScreenshotAs() method - face screenshot la intreaga pagina

```
- driver.get("http://www.google.com");
-     File src=((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
- try {
-
-         FileUtils.copyFile(src, new File("alo.png"));
-
-     }
-     catch (IOException e) {
-         e.printStackTrace();
-     }
-         driver.get("http://www.google.com");
-         WebElement ele = driver.findElement(By.name("q"));

// Get entire page screenshot
File screenshot = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
BufferedImage fullImg = ImageIO.read(screenshot);

// Get the location of element on the page
Point point = ele.getLocation();

→ // Get width and height of the element
int eleWidth = ele.getSize().getWidth();
int eleHeight = ele.getSize().getHeight();

// Crop the entire page screenshot to get only element screenshot
BufferedImage eleScreenshot = fullImg.getSubimage(point.getX(), point.getY(), eleWidth, eleHeight);
ImageIO.write(eleScreenshot, "png", screenshot);

// Copy the element screenshot to disk
File screenshotLocation = new File("C:\\\\Screenshot\\\\images\\\\GoogleLogo_screenshot.png");
FileUtils.copyFile(screenshot, screenshotLocation);
```

Capture screenshot and email results

Send email

Java mail jar - download si install

Daca proiectul este cu MAVEN atunci trebuie adaugate dependencies

SendEmail se apeleaza folosind @AfterSuite daca folosesc TestNG framework.

Se prefera folosirea interfetelor Listeners pentru a reduce liniile de cod scrise in captarea unui ecran sau trimitera de emails. Se mapeaza extrem de bine pe un framework de automation si reporting cum ar fi TestNG pe care se poate implementa interfața [ITestListner](#)

1- **onFinish**; 2- **onStart**; 3- **onTestFailureButWithSuccessPercentage**

4- **onTestFailure**; 5- **onTestSkipped**; 6- **onTestStart**

7- **onTestSuccess**

```
<!-- https://mvnrepository.com/artifact/javax.mail/mail -->
<dependency>
    <groupId>javax.mail</groupId>
    <artifactId>mail</artifactId>
    <version>1.4</version>
</dependency>
```

Synchronization in Selenium

Sincronizarea in Selenium → astept pana elementele din pagina sunt incarcate.

- **Implicit Wait** - asteapta pentru n secunde inainte de a arunca o exceptie. Se defineste la nivel global. In situatia in care elementele din pagina s-au incarcat mai repede decat cele n secunde definite in Wait, testuliese din Implicit Wait si continua cu urmatorul pas. Asculta DOM-ul web browserului. Setarea implicita este 0.
- **Explicit Wait** - are ca target un element /scenariu. Logica de sincronizare se bazeaza pe conditia din test. Il spune WebDriverului sa astepte Expected Conditions sau depasirea timpului maxim inainte de a arunca "ElementNotVisibleException". Ajuta la asteptarea elementelor Ajax care se incarca dinamic.

Combinatia dintre Implicit si explicit wait este o solutie ideală in sincronizarea cu Selenium

Exemplu - <https://www.expedia.com/>

- **Thread.Sleep** - face parte din JAVA, nu este parte din Selenium, nu asculta DOM-ul. Pune pauza testului. Nu este recomandat deoarece intarzie executia testelor fara motiv in anumite scenarii.
- **Fluent Wait** - comunica web driverului sa astepte o conditie precum si frecventa cu care vrem sa verificam starea inainte de a arunca exceptia "ElementNotVisibleException"

Feedback & Learnings & Questions

Knowledge is the
Most Beautiful of
Awards

