Thakur College of Science and Commerce

Course: Computer Science

Software Testing And Quality Assurance USCS503 Practical E-Journal

Name: **Rohit Vishwakarma**

Div: **B**

RollNo: **432**

CERTIFICATE

This is here to certify that Mr.Rohit Vishwakarma Seat Number 432 of T.Y B.Sc. SEM V Computer Science, has satisfactorily completed the required number of experiments prescribed by the UNIVERSITY OF MUMBAI during the academic year 2021 - 2022.

Date:
Place: Mumbai

Teacher In-Charge Head of Department

External Examiner

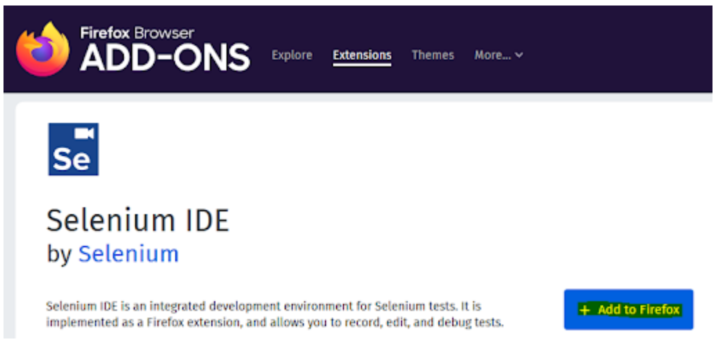
Index

10.	Load testing using JMeter, Android Application testing using Appium Tools, Bugzilla Bug tracking tools	30-33	

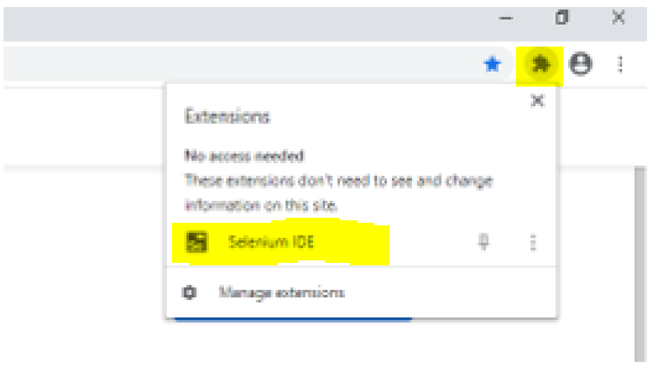
Practical-01

Aim: to Install selenium IDE, write a test Suite containing minimum 4 test cases of different formats.

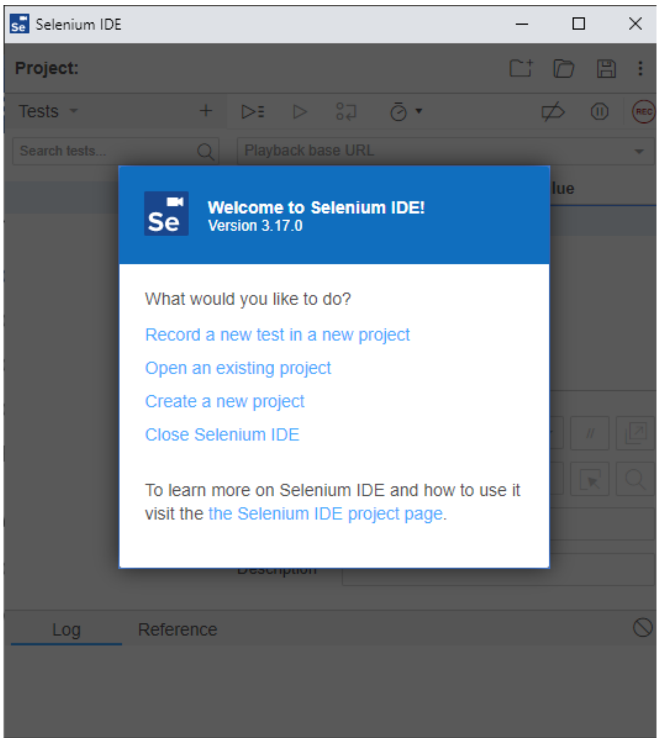
Theory:
Selenium IDE Download and Installation
Before taking off, there is one thing that needs to be in place prior to the installation; Mozilla Firefox. You can download it from here => Mozilla Firefox download
Step #1: Selenium IDE download: Open the browser (Firefox) and enter the URL http://seleniumhq.org/.This would open the official Selenium headquarter website. Navigate to the “Download” page; this page embodies all the latest releases of all the selenium components.
Step #2: Move under the selenium IDE head and click on the link present. This link represents the latest version of the tool in the repository.
Step #3: As soon as we click on the above link, a security alert box would appear so as to safeguard our system against potential risks. As we are downloading the plug-in from the authentic website, thus click on the “Allow” button.
Step #4: Now Firefox downloads the plug-in in the backdrop. As soon as the process completes, the software installation window appears. Now click on the “Install Now” button.
Step #5: After the installation is completed, a pop-up window appears asking to re-start the Firefox. Click on the “Restart Now” button to reflect the Selenium IDE installation.
Step #6: Once the Firefox is booted and started again, we can see selenium IDE indexed under menu bar -> Web Developer -> Selenium IDE.
Step #7: As soon as we open Selenium IDE, the Selenium IDE window appears.



Selenium IDE is an integrated development environment for Selenium tests. It is implemented as a Firefox extension, and allows you to record, edit, and debug tests.



No access needed
These extensions don't need to see and change information on this site.



Welcome to Selenium IDE!
Version 3.17.0

What would you like to do?

Record a new test in a new project
Open an existing project
Create a new project
Close Selenium IDE

To learn more on Selenium IDE and how to use it visit the the Selenium IDE project page.

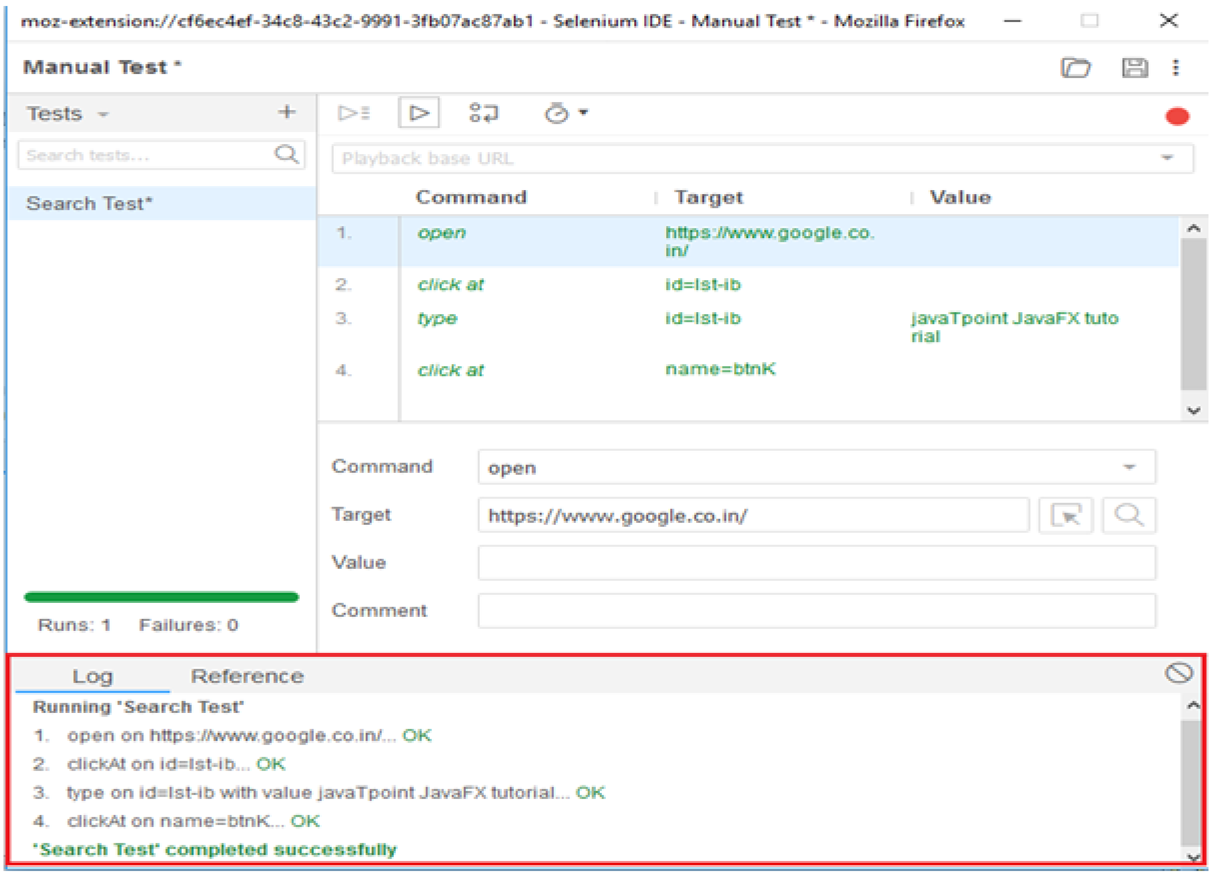
Test scenarios are rather vague and cover a wide range of possibilities. Testing is all about being very specific.
For a Test Scenario: Check Login Functionality there many possible test cases are:
Test Case 1: Check results on entering valid User Id & Password
Test Case 2: Check results on entering Invalid User ID & Password

Test Case 3: Check response when a User ID is Empty & Login Button is pressed, and many more

Types of Test Case

Given below are different types of Test Cases:

1. Functionality Test Case
2. User Interface Test Case
3. Unit Test Case
4. Integration Test Case
5. Performance Test Case
6. Security Test Case


moz-extension://cf6ec4ef-34c8-43c2-9991-3fb07ac87ab1 - Selenium IDE - Manual Test * - Mozilla Firefox
Manual Test *
Tests
Search tests...
Search Test*
Playback base URL
Runs: 1 Failures: 0
Log
Reference

	Command	Target	Value
1.	open	https://www.google.co.in/	
2.	click at	id=lst-ib	
3.	type	id=lst-ib	javaTpoint JavaFX tutorial
4.	click at	name=btnK	

Command: open
Target: https://www.google.co.in/
Value:
Comment:

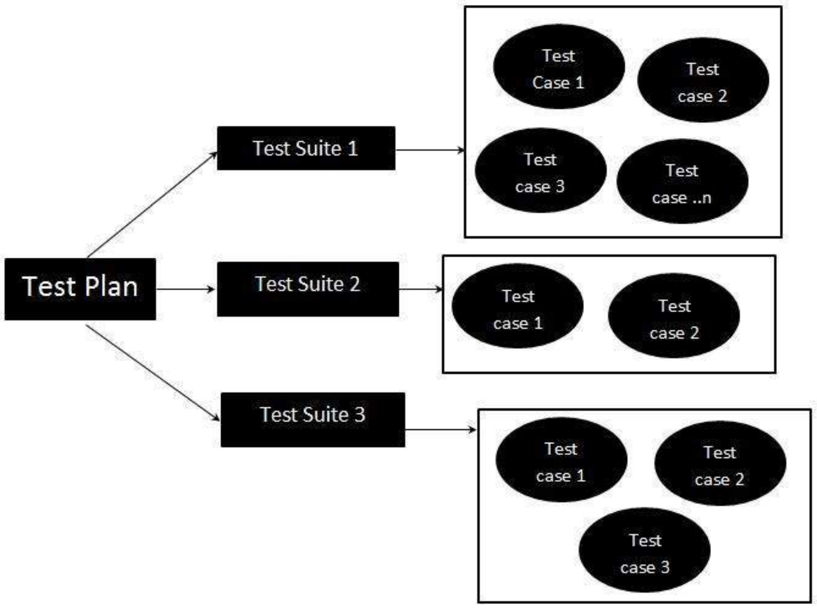
```
Running 'Search Test'
1. open on https://www.google.co.in/... OK
2. clickAt on id=lst-ib... OK
3. type on id=lst-ib with value javaTpoint JavaFX tutorial... OK
4. clickAt on name=btnK... OK
'Search Test' completed successfully
```

Conclusion: Installation of Selenium IDE is completed successfully.

Practical-02

Aim: to conduct test suite of any two websites.

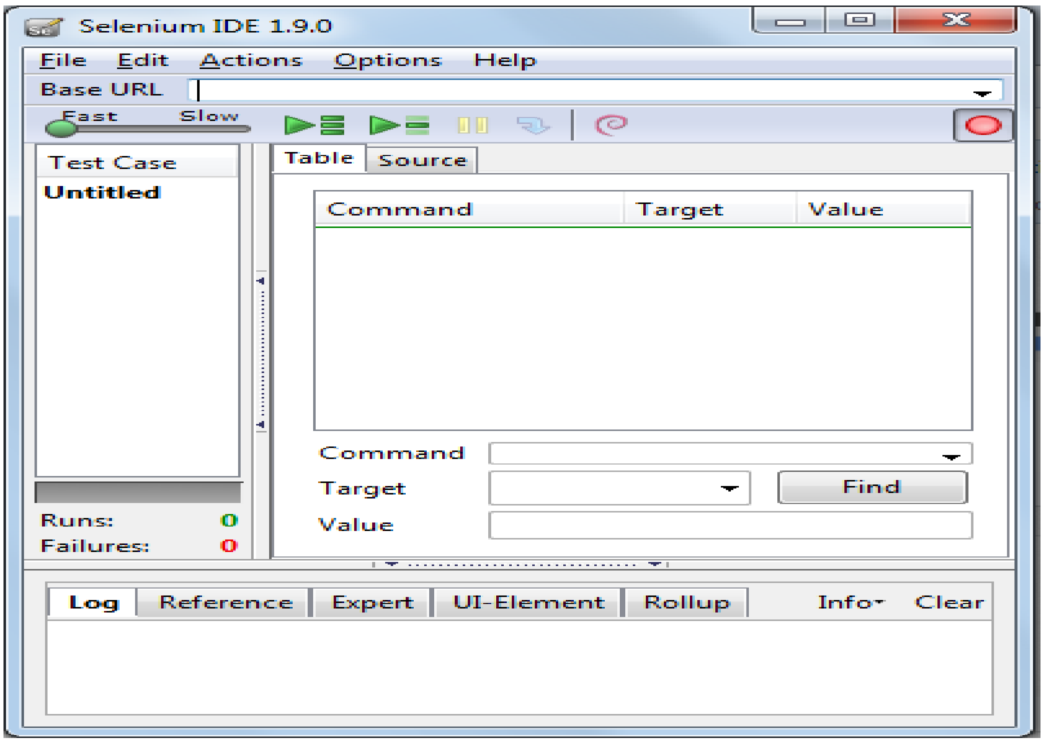
Theory:
Test suite is a container that has a set of tests which helps testers in executing and reporting the test execution status. It can take any of the three states namely Active, InProgress and completed. A Test case can be added to multiple test suites and test plans. After creating a test plan, test suites are created which in turn can have any number of tests. Test suites are created based on the cycle or based on the scope. It can contain any type of tests, viz - functional or Non-Functional.



create your First Selenium Automation Test Script.
Under this test, we will automate the following scenarios:
Invoke firefox browser.
Open URL: www.firefox.com
Click on the firefox Search text box.
Type the value "javapoint tutorials"
Click on the Search button.

Coding:

```
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
public class Firefox_Example{
public static void main(String[] args) {
System.setProperty("webdriver.gecko.driver",Path_of_Firefox_Driver"); //
Setting system properties of FirefoxDriver
WebDriver driver = new FirefoxDriver(); //Creating an object of FirefoxDriver
driver.manage().window().maximize();
driver.manage().deleteAllCookies();
driver.manage().timeouts().pageLoadTimeout(40, TimeUnit.SECONDS);
driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
driver.get("https://www.google.com/");
driver.findElement(By.name("q")).sendKeys("Browserstack Guide"); //name
locator for text box
WebElement searchbutton = driver.findElement(By.name("btnK"));//name
locator for google search
searchbutton.click();
driver.quit();
}
}
```

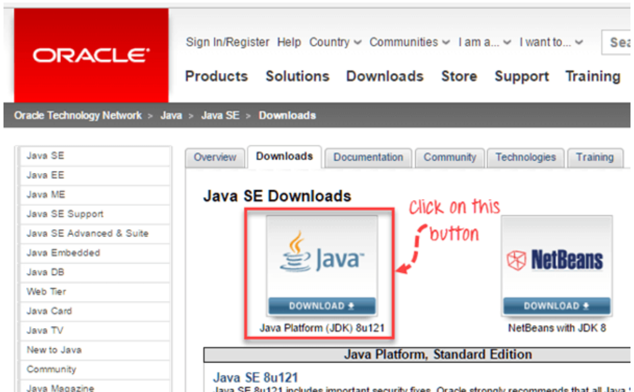


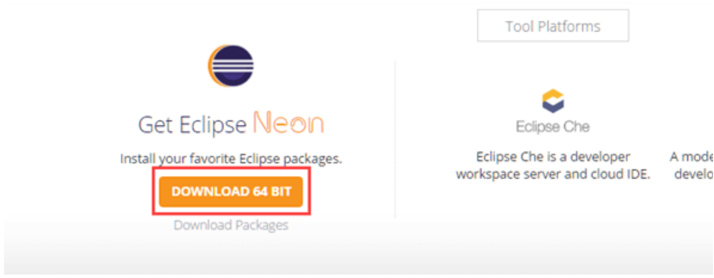
Conclusion: thus, test suite of two websites conducted successfully.

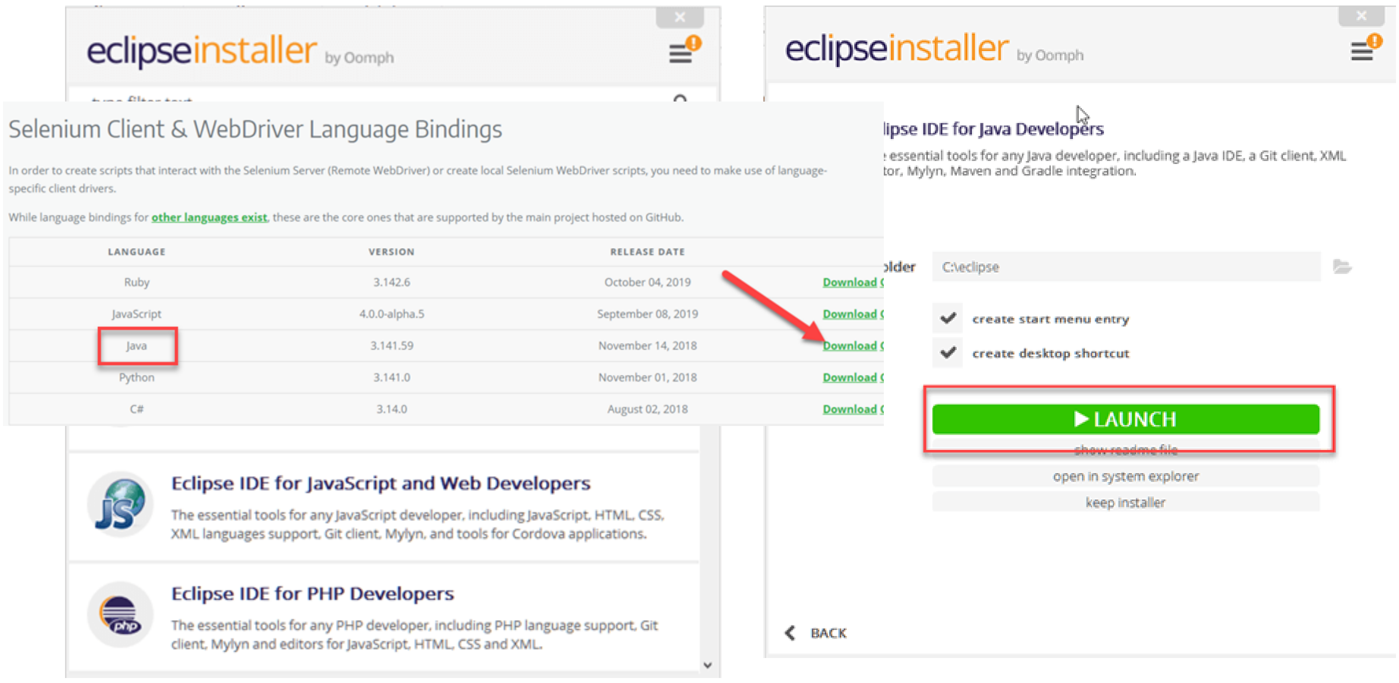
Practical-03

Aim: Install Selenium Server and demonstrate it using a script in java/PHP.

Theory:
Step 1 - Install Java on your computer
Download and install the Java Software Development Kit (JDK)
This JDK version comes bundled with Java Runtime Environment (JRE), so you do not need to download and install the JRE separately.
Once installation is complete, open command prompt and type “java”. If you see the following screen you are good to move to the next step
Step 2 - Install Eclipse IDE
Download latest version of "Eclipse IDE for Java Developers" here. Be sure to choose correctly between Windows 32 Bit and 64 Bit versions
You should be able to download an exe file named "eclipse-inst-win64" for Setup. Double-click on file to Install the Eclipse. A new window will open. Click Eclipse IDE for Java Developers.
After that, a new window will open which click button and change path to "C:\eclipse". Post that Click on Install button. After successful completion of the installation procedure, a window will appear. On that window click on Launch. This will start eclipse neon IDE for you.
Step 3 - Download the Selenium Java Client Driver
You can download the Selenium Java Client Driver here. You will find client drivers for other languages there, but only choose the one for Java.
his download comes as a ZIP file named "selenium-3.14.0.zip". For simplicity, extract the contents of this ZIP file on your C drive so that you would have the directory "C:\selenium-3.14.0\". This directory contains all the JAR files that we would later import on Eclipse.
Step 4 - Configure Eclipse IDE with WebDriver
Launch the "eclipse.exe" file inside the "eclipse" folder that we extracted in step 2. If you followed step 2 correctly, the executable should be located on C:\eclipse\eclipse.exe.
When asked to select for a workspace, just accept the default location.
3. Create a new project through File > New > Java Project. Name the project as "newproject".


ORACLE
Sign In/Register Help Country Communities I am a... I want to...
Products Solutions Downloads Store Support Training
Oracle Technology Network > Java > Java SE > Downloads
Java SE
Java EE
Java ME
Java SE Support
Java SE Advanced & Suite
Java Embedded
Java DB
Web Tier
Java Card
Java TV
New to Java
Community
Java Magazine
Overview Downloads Documentation Community Technologies Training
Java SE Downloads
Click on this button
Java
DOWNLOAD
Java Platform (JDK) 8u121
NetBeans
DOWNLOAD
NetBeans with JDK 8
Java Platform, Standard Edition
Java SE 8u121


Tool Platforms
Get Eclipse Neon
Install your favorite Eclipse packages.
DOWNLOAD 64 BIT
Download Packages
Eclipse Che
Eclipse Che is a developer workspace server and cloud IDE.



Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers.

While language bindings for **other languages exist**, these are the core ones that are supported by the main project hosted on GitHub.

LANGUAGE	VERSION	RELEASE DATE	
Ruby	3.142.6	October 04, 2019	Download
JavaScript	4.0.0-alpha.5	September 08, 2019	Download
Java	3.141.59	November 14, 2018	Download
Python	3.141.0	November 01, 2018	Download
C#	3.14.0	August 02, 2018	Download

Eclipse IDE for JavaScript and Web Developers

The essential tools for any JavaScript developer, including JavaScript, HTML, CSS, XML languages support, Git client, Mylyn, and tools for Cordova applications.

Eclipse IDE for PHP Developers

The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.

lipse IDE for Java Developers

essential tools for any Java developer, including a Java IDE, a Git client, XML tor, Mylyn, Maven and Gradle integration.

older C:\eclipse

✓ create start menu entry

✓ create desktop shortcut

/Run your First Selenium WebDriver script by copy paste below code after executing above five simple steps.

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class Webdriver_class
{
public static void main(String[] args)
{
WebDriver driver = new FirefoxDriver();
driver.get("http://google.com");
System.out.println(driver.getTitle());
driver.close();
}
}
```

Conclusion: thus, Selenium server installation is completed .

Practical-04

Aim: Write and test a program to login a specific web page.

Theory:
1. Create A Selenium WebDriver Instance
Webdriver driver=new ChromeDriver();
In order to launch the website in the desired browser, you need to set the system properties to the path of the driver for the required browser. In this Selenium Java tutorial, we will use Chromedriver for demonstrating Selenium login example with Java. The syntax for the same will be:
System.setProperty(“webdriver.chrome.driver”, “File path for the Exe”);
2. Configure Your Browser If Required
Based on the needs, we can configure the browser. For example, in this Selenium Java tutorial regarding Selenium login with Java, a browser, by default, will be in minimized mode. However, we can set up the browser in the maximize mode. Below is the syntax.
driver.manage().window().maximize();

Other things that you can do for configuring your browser is set up different options like disabling info bars, browser notifications, and adding extensions. You can also use the capabilities class to run your script on various browsers, thereby helping in cross-browser testing.

3. Navigate To The Required URL

Open the browser with the desired URL. All you have to do is write the below syntax and you have your URL open in the desired instantiated browser.

driver.get(“https://www.linkedin.com/login”);

4. Locate The HTML Element

This is the heart of writing a Selenium script. For this to function, you need to have a clear understanding of the different locators used to find the HTML element. You can refer my below articles that talks about the different locators available in Selenium and how to locate the element with different examples:

- ID locator in Selenium WebDriver
- Name Locator in Selenium WebDriver
- TagName Locator in Selenium WebDriver
- CSS Selector in Selenium WebDriver
- XPath in Selenium WebDriver

For example, let's try to locate the email and password field of the login form of LinkedIn

Below is the DOM structure for the email input box:

email input box

You can locate it via ID locator in Selenium WebDriver as below:

driver.findElement(By.id(“username”));

Since this returns a web element, you can store it in web element variable as below:

WebElement username=driver.findElement(By.id(“username”));

The same can be achieved for password and login button field which is Format:

driver.findElement(By.id(“password”)); WebElement password=driver.findElement(By.id(“password”));

driver.findElement(By.xpath(“//button[text()=’Sign in’]”)); WebElement login= driver.findElement(By.xpath(“//button[text()=’Sign in’]”));

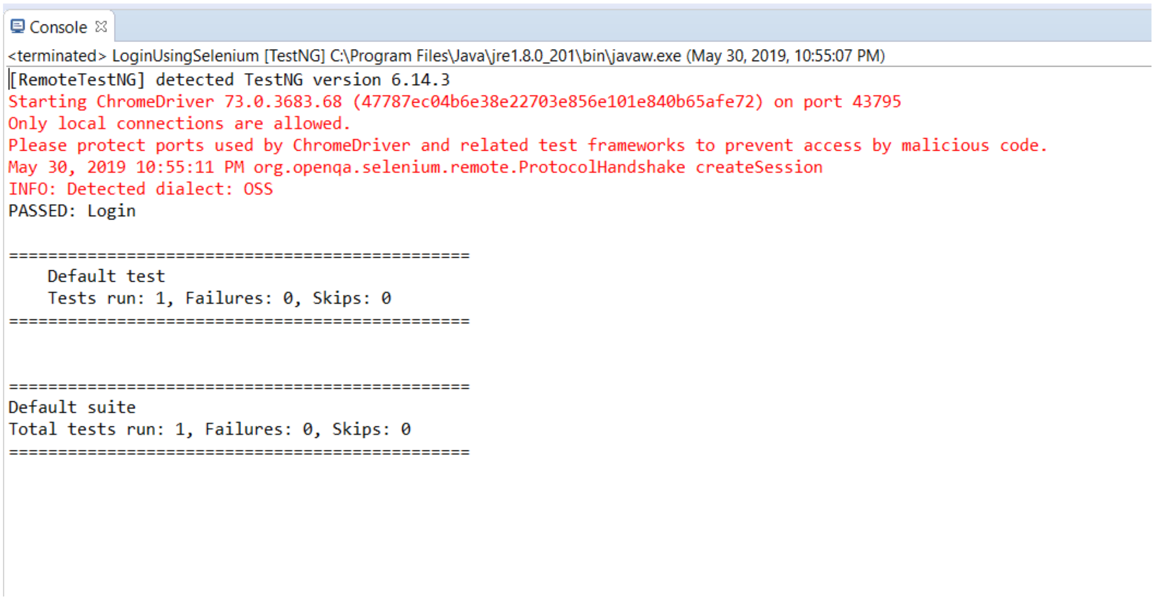
5. Perform Action On The Located HTML Element

Once located, you need to perform the desired action which in our case is sending text to email and password field and clicking on the login button. To

execute this action in Selenium login example with Java, we make use of methods sendKeys and click provided by Selenium as below:
Format:
username.sendKeys(“xyz@gmail.com”);
password.sendKeys(“exampleAboutSelenium123”); login.click();
You just finished writing the most important parts of the script. Now, in this Selenium Java tutorial, you only need to ensure these actions have successfully logged in the user, which comes to our final step of script creation for using Selenium to login with Java.
6. Verify & Validate The Action
In order to validate the results, all you need to do is use an assertion. Assertions are vital for comparing the expected results and the actual results. Almost similar to your test cases, wherein each test case has an actual and expected behavior to it. If it matches, the test case pass, if not, then the test case fails. Assertions do exactly the same. Assertion class are provided by both JUnit and TestNG framework, and you can opt to choose either. The below syntax will help to assert (validate) the outcome from actions by performing Selenium login with Java.
Assert.assertEquals(String actual, String expected);
So, in this case, we will save our actual URL post login into a string value which is:
String actualUrl=” https://www.linkedin.com/feed/”;
And expected URL can be found from the below method:
String expectedUrl= driver.getCurrentUrl();
So your final assertion would become as:
Assert.assertEquals(actualUrl, expectedUrl);
Note: In order to use assertion, you need to use the annotations of TestNG or JUnit "@Test" for assertions to function. In case, right now you don’t want to get into the hassle of going into the framework keywords, you can simply match the string using an "if" statement and print the results in console accordingly, something like below:
Format:
if(actualUrl.equalsIgnoreCase(expectedUrl)) { System.out.println(“Test passed”) } else { System.out.println(“Test failed”) }

Coding:

```
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.Test;
public class LoginUsingSelenium {
@Test public void login() { // TODO Auto-generated method stub
System.setProperty("webdriver.chrome.driver", "path of driver");
WebDriver driver=new ChromeDriver(); driver.manage().window().maximize();
driver.get("https://www.linkedin.com/login");  WebElement
username=driver.findElement(By.id("username"));
 WebElement password=driver.findElement(By.id("password"));
 WebElement login=driver.findElement(By.xpath("//button[text()='Sign in']"));
 username.sendKeys("example@gmail.com"); password.sendKeys("password");
 login.click(); String actualUrl="https://www.linkedin.com/feed/";
 String expectedUrl= driver.getCurrentUrl();
 Assert.assertEquals(expectedUrl,actualUrl); }   }
```



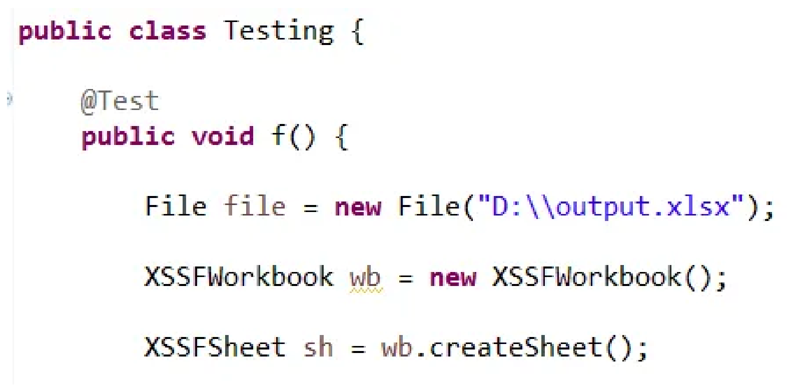
```
Console
<terminated> LoginUsingSelenium [TestNG] C:\Program Files\Java\jre1.8.0_201\bin\javaw.exe (May 30, 2019, 10:55:07 PM)
[RemoteTestNG] detected TestNG version 6.14.3
Starting ChromeDriver 73.0.3683.68 (47787ec04b6e38e22703e856e101e840b65afe72) on port 43795
Only local connections are allowed.
Please protect ports used by ChromeDriver and related test frameworks to prevent access by malicious code.
May 30, 2019 10:55:11 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: OSS
PASSED: Login

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 1, Failures: 0, Skips: 0
===============================================
```

Conclusion: program to login a specific web page tested successfully.

Practical-05

Aim: Write and test a program to update 10 students records into table into Excel file.

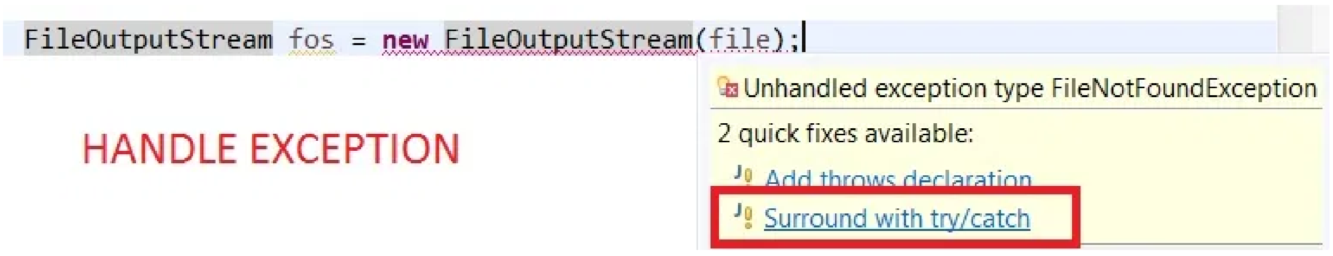
Theory:
Step 1: Specify the file path where you wish the Excel Sheet to show. I am hoping it to show in my D: drive in a file called output. Type the following:
File file = new File("D:\\output.xlsx");
You need to specify the file format too, and hence the suffix of .xlsx.
Step 2: Time to create a Workbook now. We will use the Apache POI class called XSSFWorkbook for that as we had seen in the previous tutorial. Type the following:
XSSFWorkbook wb = new XSSFWorkbook();
You might have to import the following package if not already imported:
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
Step 3: Time to create a sheet. Type the following for creating a sheet:
XSSFSheet sh = wb.createSheet();
You might have to import the following package if not already imported:
import org.apache.poi.xssf.usermodel.XSSFSheet;
You can choose to provide a name for the Sheet by simply typing the name of it in the parameter using double inverted commas like this:
XSSFSheet sh = wb.createSheet("First Sheet");



```
public class Testing {

    @Test
    public void f() {

        File file = new File("D:\\output.xlsx");

        XSSFWorkbook wb = new XSSFWorkbook();

        XSSFSheet sh = wb.createSheet();
```

Step 4: Type the following to enter a String value in the first row and first column (A1) of your excel sheet:
sh.createRow(0).createCell(0).setCellValue("Age");

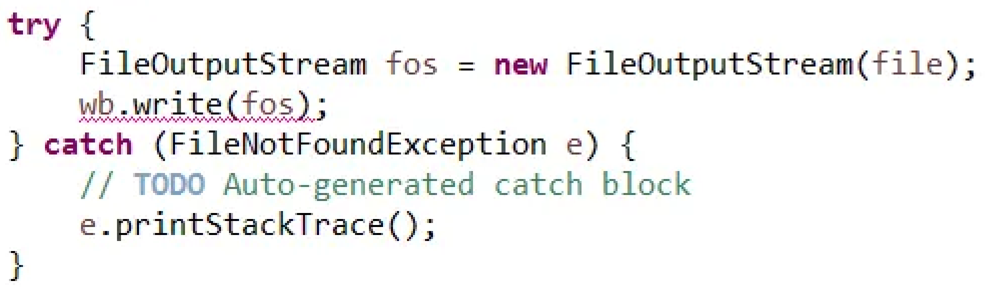
As you can see the parameters of createRow and createCell will be integers where you specify the row and column numbers. We have put 0, 0 in them so it will populate the first cell.
Step 5: Time to use an Output Stream. Type the following:
FileOutputStream fos = new FileOutputStream(file);
You might have to import the following package for your JVM to understand the FileOutputStream Class:
import java.io.FileOutputStream;



```
FileOutputStream fos = new FileOutputStream(file);
```

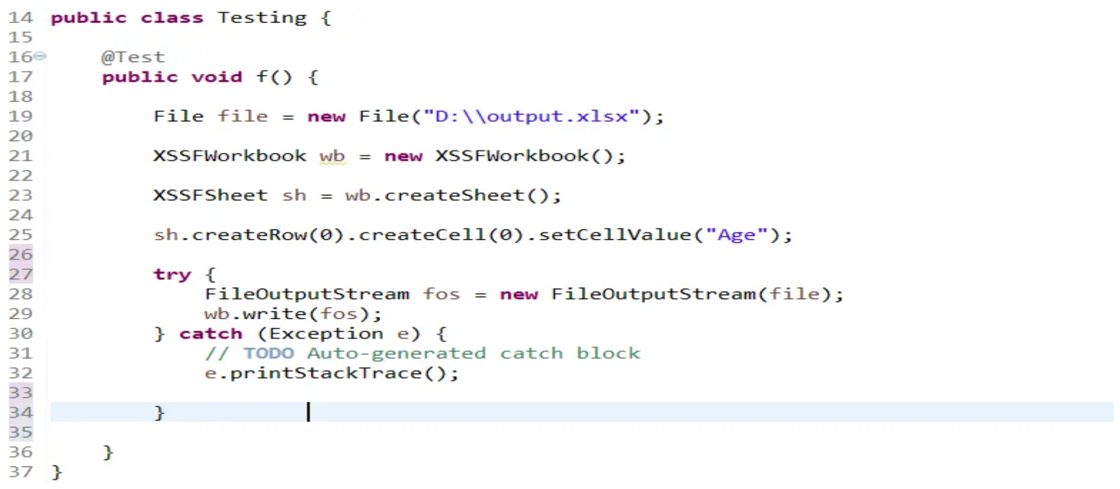
Unhandled exception type FileNotFoundException
2 quick fixes available:
Add throws declaration
Surround with try/catch

Step 6: Now we will use the write method of the XSSFWorkbook instance to actually start writing what we intended to write. Type the following for that in the try block itself:
wb.write(fos);

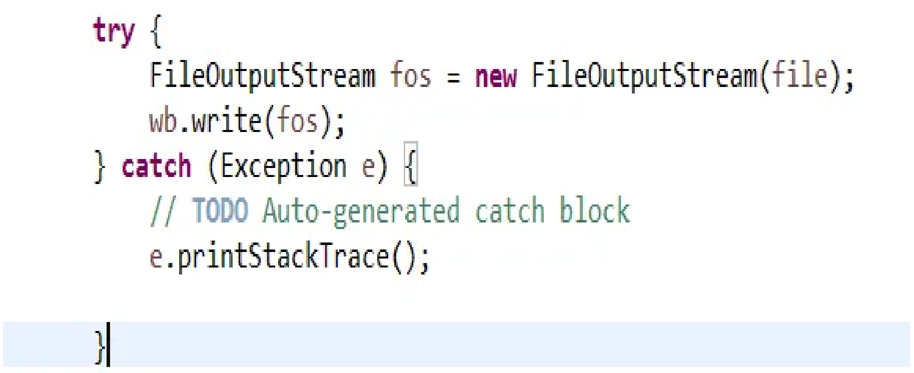


```
try {
    FileOutputStream fos = new FileOutputStream(file);
    wb.write(fos);
} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Step 7: You can either handle that too or simply replace the FileNotFoundException with its superset Exception like this:

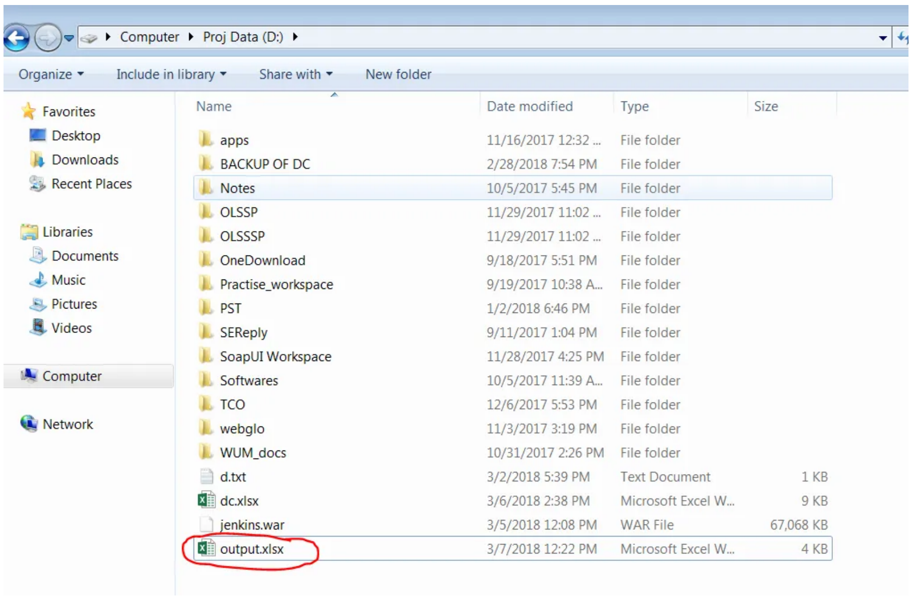


```
public class Testing {

    @Test
    public void f() {

        File file = new File("D:\\output.xlsx");

        XSSFWorkbook wb = new XSSFWorkbook();

        XSSFSheet sh = wb.createSheet();

        sh.createRow(0).createCell(0).setCellValue("Age");

        try {
            FileOutputStream fos = new FileOutputStream(file);
            wb.write(fos);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();

        }

    }
}
```



```
try {
    FileOutputStream fos = new FileOutputStream(file);
    wb.write(fos);
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();

}
```

Step 8: That's the whole code there. Just run the program now.
Step 9: Let's navigate to the D: drive folder to check if the excel file has been created or not:

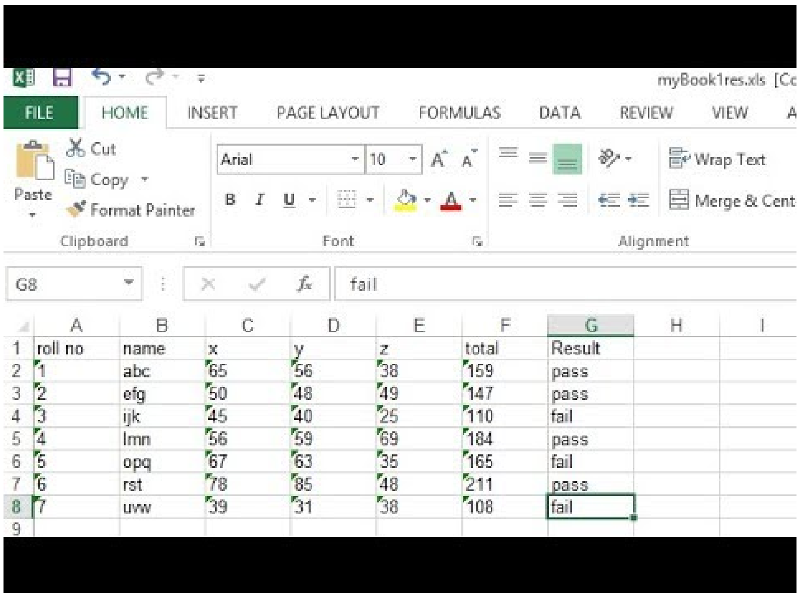


Computer ▸ Proj Data (D:) ▸

Organize ▾ Include in library ▾ Share with ▾ New folder

Favorites: Desktop, Downloads, Recent Places
Libraries: Documents, Music, Pictures, Videos
Computer
Network

Name	Date modified	Type	Size
apps	11/16/2017 12:32 ...	File folder	
BACKUP OF DC	2/28/2018 7:54 PM	File folder	
Notes	10/5/2017 5:45 PM	File folder	
OLSSP	11/29/2017 11:02 ...	File folder	
OLSSSP	11/29/2017 11:02 ...	File folder	
OneDownload	9/18/2017 5:51 PM	File folder	
Practise_workspace	9/19/2017 10:38 A...	File folder	
PST	1/2/2018 6:46 PM	File folder	
SEReply	9/11/2017 1:04 PM	File folder	
SoapUI Workspace	11/28/2017 4:25 PM	File folder	
Softwares	10/5/2017 11:39 A...	File folder	
TCO	12/6/2017 5:53 PM	File folder	
webglo	11/3/2017 3:19 PM	File folder	
WUM_docs	10/31/2017 2:26 PM	File folder	
d.txt	3/2/2018 5:39 PM	Text Document	1 KB
dc.xlsx	3/6/2018 2:38 PM	Microsoft Excel W...	9 KB
jenkins.war	3/5/2018 12:08 PM	WAR File	67,068 KB
output.xlsx	3/7/2018 12:22 PM	Microsoft Excel W...	4 KB



G8 fail

	A	B	C	D	E	F	G
1	roll no	name	x	y	z	total	Result
2	1	abc	65	56	38	159	pass
3	2	efg	50	48	49	147	pass
4	3	ijk	45	40	25	110	fail
5	4	lmn	56	59	69	184	pass
6	5	opq	67	63	35	165	fail
7	6	rst	78	85	48	211	pass
8	7	uvw	39	31	38	108	fail
9							

Conclusion: program to update 10 students records into table into Excel file tested successfully.

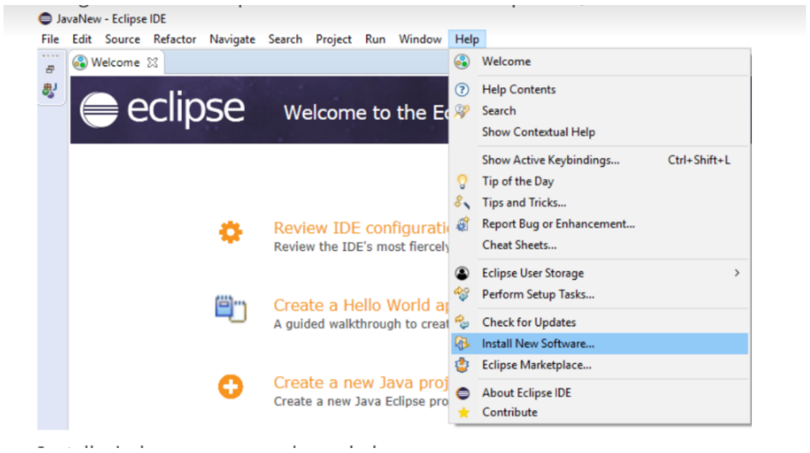
Practical-06

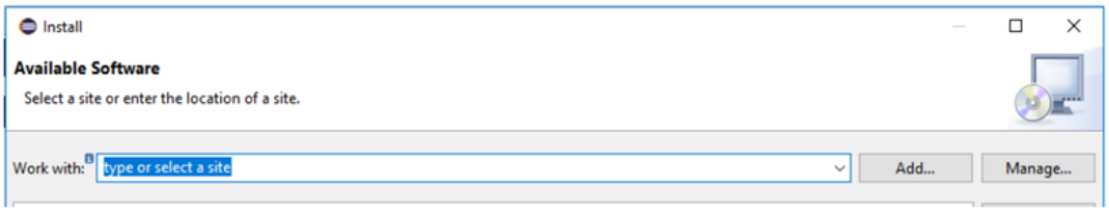
Aim: Steps to install testng plugin in eclipse.

Theory:

- The following steps are performed on latest eclipse version photon
- Navigate to menu Help&gt ; Install New Software in eclipse IDE,

- Install window appears as shown below
- Click on Add button
- In the Add repository window provide following details and click on "Add"
  button. Name: TestNG, url: http://beust.com/eclipse/ ,
- Important Note: url has now changed to https://dl.bintray.com/testng-t
- Eclipse starts searching for the repository team/testng-eclipse-release/
- Once the search is complete, it lists the TestNG repository
- Select TestNG in the list and click on Next button
- Eclipse starts calculating the dependencies and requirements for TestNG plugin
- Once calculation is complete, it lists the items for you to review and select for install
- Select "TestNG" in the above list and click on Next button
- Eclipse requests for your acceptance on license details.
- Accept the license and click on Finish button
- Eclipse starts the installation of the plug-in.
- Review the status by looking at the bottom of eclipse IDE
- There are chances to see some warning messages, you can just go ahead by clicking on "Install Anyway" button
- Once the installation is complete, eclipse requests you to restart the IDE, proceed with the same. This means that installation is successful.



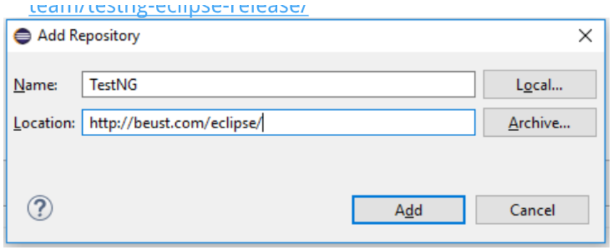

Install

Available Software

Select a site or enter the location of a site.

Work with: type or select a site

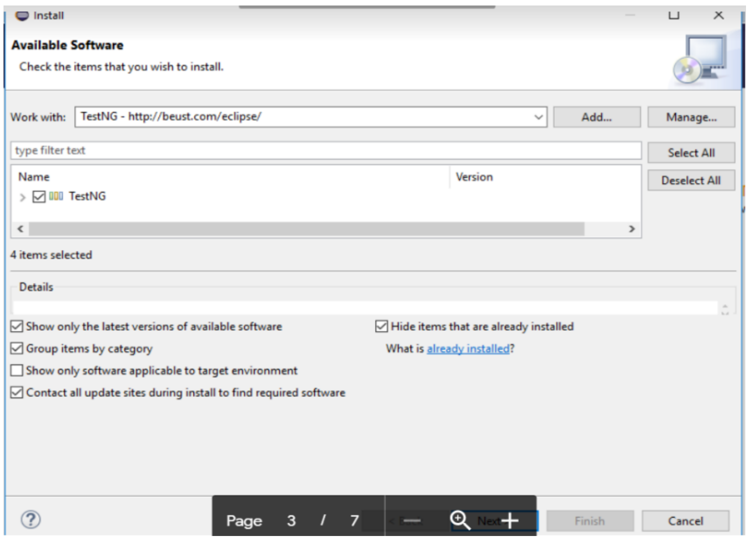
team/testng-eclipse-release/


Add Repository

Name: TestNG

Location: http://beust.com/eclipse/

Eclipse starts searching for the repository


Install

Available Software

Check the items that you wish to install.

Work with: TestNG - http://beust.com/eclipse/

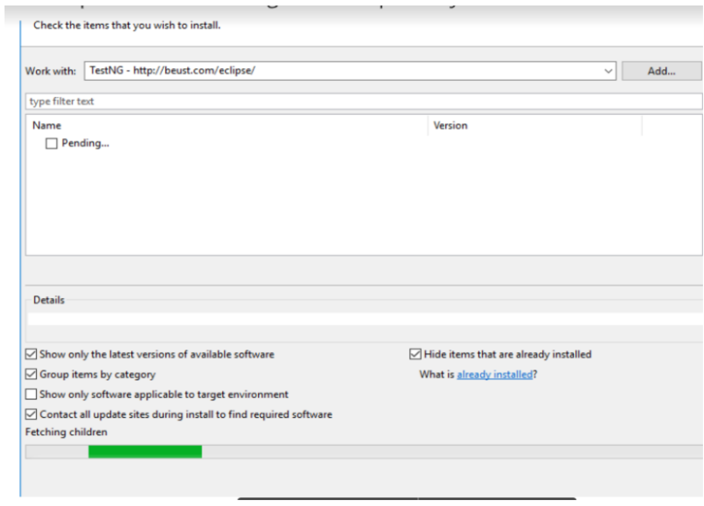
type filter text

Name	Version
☑ TestNG	

4 items selected

Details

☑ Show only the latest versions of available software
☑ Group items by category
☐ Show only software applicable to target environment
☑ Contact all update sites during install to find required software
☑ Hide items that are already installed
What is already installed?


Check the items that you wish to install.

Work with: TestNG - http://beust.com/eclipse/

type filter text

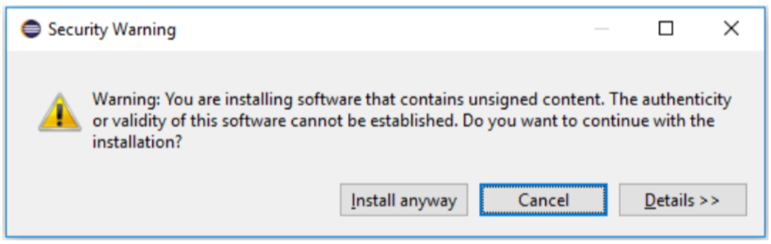
Name	Version
☐ Pending...	

Details

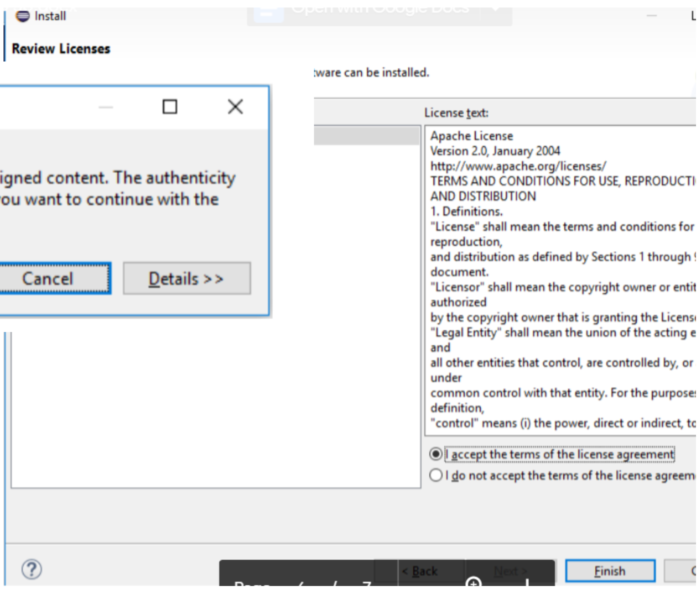
☑ Show only the latest versions of available software
☑ Group items by category
☐ Show only software applicable to target environment
☑ Contact all update sites during install to find required software
☑ Hide items that are already installed
What is already installed?

Fetching children

by clicking on "Install Anyway" button


Security Warning

Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?


Install

Review Licenses

ware can be installed.

License text:

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/
TERMS AND CONDITIONS FOR USE, REPRODUCTI
AND DISTRIBUTION
1. Definitions.
"License" shall mean the terms and conditions for
reproduction,
and distribution as defined by Sections 1 through
document.
"Licensor" shall mean the copyright owner or entit
authorized
by the copyright owner that is granting the Licens
"Legal Entity" shall mean the union of the acting e
and
all other entities that control, are controlled by, or
under
common control with that entity. For the purpose
definition,
"control" means (i) the power, direct or indirect, to

[✓ I accept the terms of the license agreement]
I do not accept the terms of the license agreem

Conclusion: testng plugin installed successfully in ellipse.

Practical-07

Aim: Write and test a program to provide total no of objects present/ available on the page

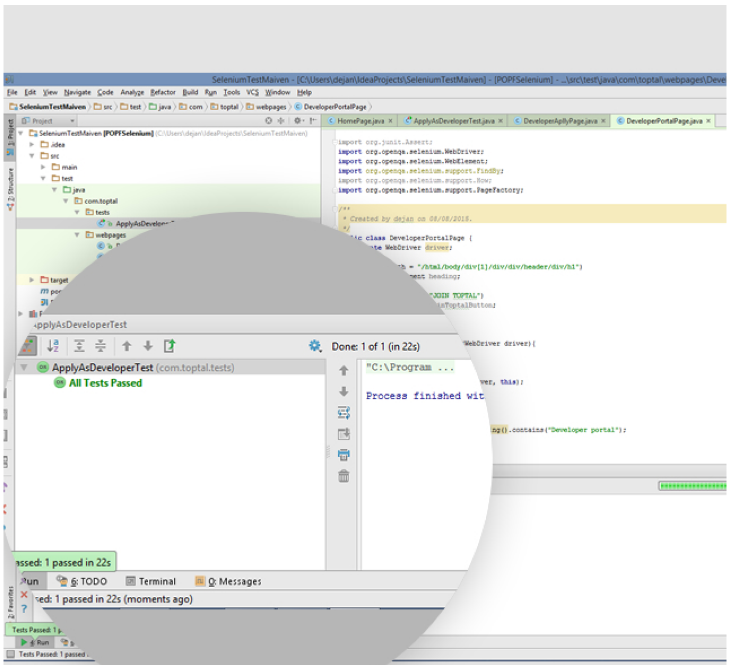
Theory:
As with most software patterns, they really make the most sense once you have actually coded your way into a situation where you realize you need help. Learning a pattern up front can prevent you from making a design mistake, but as in most situations, making a mistake is your very best opportunity to learn. The Page Object Pattern, however, makes perfect sense whether you have an existing suite of functional Selenium tests or you are starting out fresh.
Page object model is an object design pattern in Selenium test automation used to make a repository of Web UI elements. POM improves test readability and reduces code duplication, by acting as an interface for the page under test. In the Page object model, all the webpages have individual page classes. These page classes find the web elements on the page, and contain all the relevant page methods.

```
clickLoginButton();
enterCredentials(user_name,user_password);
checkIfImageIsDisplayed();
clickLoginButton();
enterCredentials(user_name,user_password);
checkIfImageIsDisplayed();
```

So, while performing Selenium test automation, the test case uses the object of this Page class to call the various methods to perform actions on the web page. Since all the web elements related to a page are present at a common location, it becomes easy to implement Selenium Java testing routines, and update them when required.

Coding:

```
import com.thoughtworks.selenium.*;
@SuppressWarnings("deprecation")
public class objectcount extends SeleneseTestCase
{
public void setUp() throws Exception {
setUp("http://www.google.com/","*firefox");
}
public void testloginlogout(){
selenium.setSpeed("1000");
selenium.open("/mail/help/intl/en/logout.html#hl=en");
selenium.waitForPageToLoad("30000");
selenium.windowMaximize();
int num = selenium.getXpathCount("//p").intValue();
System.out.println("The number of option elements present are " +num);
}
}
```



SeleniumTestMaiven - [C:\Users\dejan\IdeaProjects\SeleniumTestMaiven] - [POPFSelenium] - ...\src\test\java\com\toptal\webpages\Deve
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
SeleniumTestMaiven src test java com toptal webpages DeveloperPortalPage
Project
HomePage.java
ApplyAsDeveloperTest.java
DeveloperAplyPage.java
DeveloperPortalPage.java
SeleniumTestMaiven [POPFSelenium] (C:\Users\dejan\IdeaProjects\SeleniumTestMaiven)
.idea
src
main
test
java
com.toptal
tests
webpages
target
6: TODO
Terminal
0: Messages

```
import org.junit.Assert;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.How;
import org.openqa.selenium.support.PageFactory;

/**
 * Created by dejan on 08/08/2016.
 */
```

```
ApplyAsDeveloperTest
Done: 1 of 1 (in 22s)
ApplyAsDeveloperTest (com.toptal.tests)
All Tests Passed
"C:\Program ...
Process finished wit
ssed: 1 passed in 22s
sed: 1 passed in 22s (moments ago)
Tests Passed: 1 passed
```

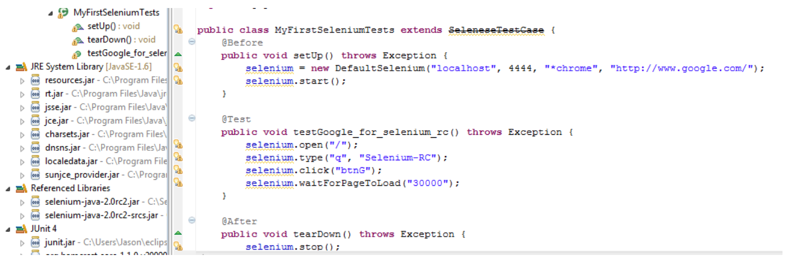
Conclusion: a program to provide total no of present objects tested successfully.

Practical-08

Aim: Write and test a program to get the number of items in a list/ Combo box.

Theory:
While testing combo box controls, you can use specific properties and methods of the corresponding program object to perform certain actions and obtain data stored in controls. You can call these methods and properties from your keyword tests, as well as from scripts. This topic describes how to work with the needed properties and methods from your scripts. However, when testing a control from your keyword test, you can use the same methods and properties calling them from keyword test operations. For more information, see Keyword Tests Basic Operations.
When working with a combo box control, you may need to determine the total number of items. If the tested application uses standard Win32 combo box controls, you can use various properties and methods provided by the Win32ComboBox object (this object is automatically associated with combo box controls during the test run).

Win32ComboBox provides a set of specific properties and methods that let you work with combo box controls. You can determine the total number of items a combo box contains via the wItemCount property.


MyFirstSeleniumTests
setUp() : void
tearDown() : void
testGoogle_for_seler
JRE System Library [JavaSE-1.6]
resources.jar - C:\Program Files
rt.jar - C:\Program Files\Java\jr
jsse.jar - C:\Program Files\Java\
jce.jar - C:\Program Files\Java\
charsets.jar - C:\Program Files\
dnsns.jar - C:\Program Files\Ja
localedata.jar - C:\Program File
sunjce_provider.jar - C:\Progra
Referenced Libraries
selenium-java-2.0rc2.jar - C:\Se
selenium-java-2.0rc2-srcs.jar -
JUnit 4
junit.jar - C:\Users\Jason\eclips

```
public class MyFirstSeleniumTests extends ~~SeleneseTestCase~~ {
    @Before
    public void setUp() throws Exception {
        selenium = new DefaultSelenium("localhost", 4444, "*chrome", "http://www.google.com/");
        selenium.start();
    }

    @Test
    public void testGoogle_for_selenium_rc() throws Exception {
        selenium.open("/");
        selenium.type("q", "Selenium-RC");
        selenium.click("btnG");
        selenium.waitForPageToLoad("30000");
    }

    @After
    public void tearDown() throws Exception {
        selenium.stop();
```

How to count the number of items in the drop down field using size() using Selenium WebDriver.

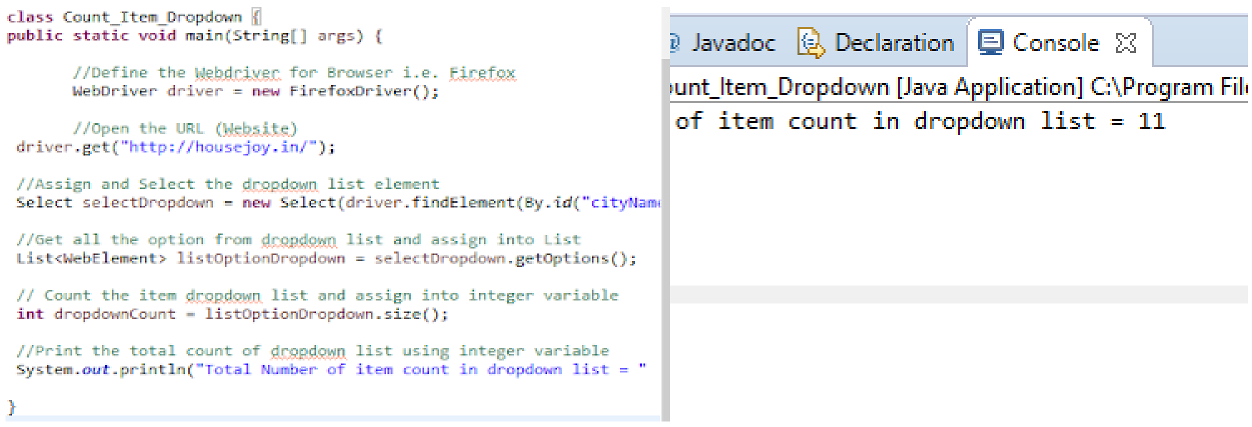
Steps:

1. Define Firefox Browser and open the Firefox Browser
2. Open the URL (Website)
3. Assign and Select the drop-down list element
4. Get all the option from drop-down list and assign into List
5. Count the item drop-down list and assign into integer variable

Coding:

```
package dayOne;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.Select;

public class Count_Item_Dropdown {
      public static void main(String[] args) {
           //Define the Webdriver for Browser i.e. Firefox
               WebDriver driver = new FirefoxDriver();
                //Open the URL (Website)
         driver.get("http://housejoy.in/");
         //Assign and Select the dropdown list element
         Select selectDropdown = new
Select(driver.findElement(By.id("cityName")));
         //Get all the option from dropdown list and assign into List
         List<WebElement> listOptionDropdown = selectDropdown.getOptions();
```

```
          // Count the item dropdown list and assign into integer variable
      int dropdownCount = listOptionDropdown.size();
         //Print the total count of dropdown list using integer variable
      System.out.println("Total Number of item count in dropdown list = "  +
dropdownCount);
           }
}
```



```
class Count_Item_Dropdown {
public static void main(String[] args) {

       //Define the Webdriver for Browser i.e. Firefox
       WebDriver driver = new FirefoxDriver();

       //Open the URL (Website)
 driver.get("http://housejoy.in/");

 //Assign and Select the dropdown list element
 Select selectDropdown = new Select(driver.findElement(By.id("cityNam

 //Get all the option from dropdown list and assign into List
 List<WebElement> listOptionDropdown = selectDropdown.getOptions();

 // Count the item dropdown list and assign into integer variable
 int dropdownCount = listOptionDropdown.size();

 //Print the total count of dropdown list using integer variable
 System.out.println("Total Number of item count in dropdown list = "

}
```

```
unt_Item_Dropdown [Java Application] C:\Program Fil
of item count in dropdown list = 11
```

Conclusion: program to get the number of items in a list/ Combo box tested successfully.

Practical-09

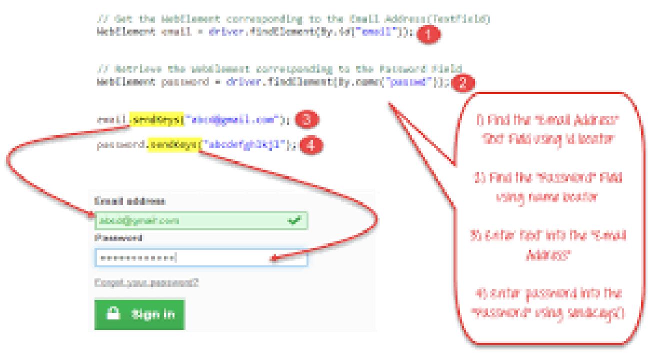
Aim: Write and test a program to get number of links, buttons and fields.

Theory:
Forms are the fundamental web elements to receive information from the website visitors. Web forms have different GUI elements like Text boxes, Password fields, Checkboxes, Radio buttons, dropdowns, file inputs, etc.
We will see how to access these different form elements using Selenium Web Driver with Java. Selenium encapsulates every form element as an object of WebElement. It provides API to find the elements and take action on them like entering text into text boxes, clicking the buttons, etc. We will see the methods that are available to access each form element.
On a webpage, generally, there are multiple links/hyperlinks, and they serve particular purposes. A few links redirect to third-party websites such as Wikipedia and some redirect to the same website. They are a beautiful way to reduce the searching time. For example, If I am reading an article about selenium and says that Selenium WebDriver is used for automation testing, I can link the automation testing word, which would redirect to a detailed article

about automation testing. This will save time for someone who did not know about the automation testing and searched it separately. Therefore, they are important and these broken links in Selenium require testing and verification before publishing a web page.
A working URL will always have an HTTP response code 200, which is a success and valid. While testing these links, it is difficult for a user to manually click and check all the broken links on a webpage. Therefore, we can search for all the links on a web page and check the status of each of the links to validate whether it is a valid link or not. The same is the case with Images, where we can check for the valid and visible image by validating that the images' src link is valid. This article will explain various ways to validate whether certain images and links are valid or are in broken status while browsing through the all/broken links in selenium tests.

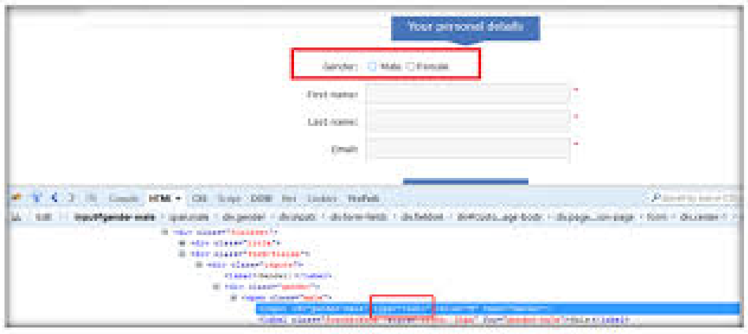
Coding:

```
package pkg_objects;
import com.thoughtworks.selenium.*;
import org.openqa.selenium.server.*;
import org.testng.annotations.*;
@SuppressWarnings(&quot;deprecation&quot;)
public class NewTest {
public Selenium selenium;
public SeleniumServer seleniumserver;
@BeforeClass
public void setUp() throws Exception {
RemoteControlConfiguration rc = new
RemoteControlConfiguration();
seleniumserver = new SeleniumServer(rc);
selenium = new
DefaultSelenium(&quot;localhost&quot;,4444,&quot;*firefox&quot;,&quot;htt
p://&quot;);
seleniumserver.start();
selenium.start();
}
@Test
public void f() {
selenium.open(&quot;http://www.google.co.in/&quot;);
selenium.windowMaximize();
String lc[]=selenium.getAllLinks();
System.out.println(&quot;Total no. of Links: &quot;+lc.length);
String bc[]=selenium.getAllButtons();
System.out.println(&quot;Total no. of Buttons: &quot;+bc.length);
String fc[]=selenium.getAllFields();
System.out.println(&quot;Total no. of Fields: &quot;+fc.length);
}
@AfterClass
public void tearDown() throws Exception{
selenium.stop();
seleniumserver.stop();
}
}
```

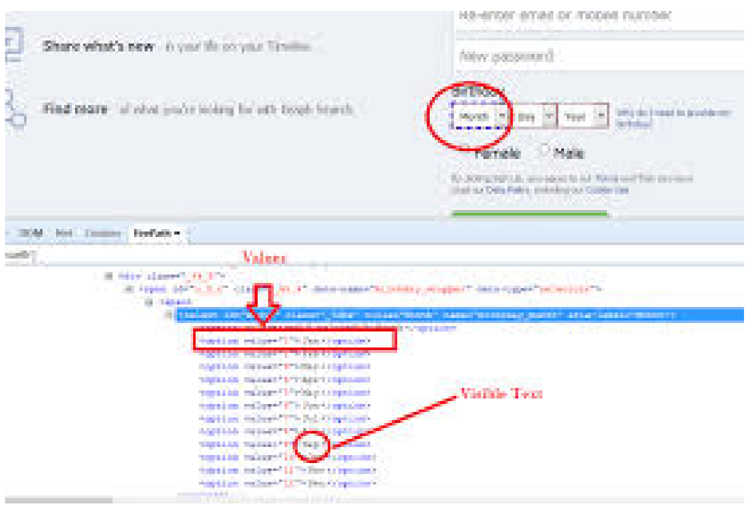


```
// Get the WebElement corresponding to the Email Address(TextField)
WebElement email = driver.findElement(By.id("email")); 1

// Retrieve the WebElement corresponding to the Password Field
WebElement password = driver.findElement(By.name("passwd")); 2

email.sendKeys("abcd@gmail.com"); 3
password.sendKeys("abcdefghlkjl"); 4
```

Email address
abcd@gmail.com
Password
Forgot your password?
Sign in
1) Find the "Email Address" Text Field using id locator
2) Find the "Password" Field using name locator
3) Enter text into the "Email Address"
4) Enter password into the "Password" using sendkeys()



HTML CODE FOR RADIO BUTTON



Re-enter email or mobile number
New password
Share what's new in your life on your Timeline.
Find more of what you're looking for with Graph Search.
Birthday
Month
Day
Year
Female
Male
DOM Net Cookies FirePath
Values
Visible Text

Conclusion: program to get number of links, buttons and fields tested successfully.

Practical-10

Aim: Load testing using JMeter, Android Application testing using Appium Tools, Bugzilla Bug tracking tools.

Theory:
Types of Mobile Testing
There are broadly 2 kinds of testing that take place on mobile devices:
#1. Hardware testing:
The device including the internal processors, internal hardware, screen sizes, resolution, space or memory, camera, radio, Bluetooth, WIFI etc. This is sometimes referred to as, simple "Mobile Testing".
#2. Software or Application testing:
The applications that work on mobile devices and their functionality are tested. It is called the "Mobile Application Testing" to differentiate it from the earlier method. Even in mobile applications, there are few basic differences that are important to understanding:

a) Native apps: A native application is created for use on a platform like mobile and tablets.

b) Mobile web apps are server-side apps to access website/s on mobile using different
browsers like Chrome, Firefox by connecting to a mobile network or wireless network like
WIFI.

c) Hybrid apps are combinations of native app and web app. They run on devices or offline
and are written using web technologies like HTML5 and CSS.

There are few basic differences that set these apart:

Native apps have single platform affinity while mobile web apps have the cross-platform affinity.
Native apps are written in platforms like SDKs while Mobile web apps are written with web technologies like HTML, CSS, asp.net, Java, PHP.
For a native app, installation is required but for mobile web apps, no installation is required.
A native app can be updated from the play store or app store while mobile web apps are centralized updates.
Many native apps don’t require an Internet connection but for mobile web apps, it’s a must.
Native app works faster when compared to mobile web apps.
Native apps are installed from app stores like Google play store or app store where mobile web are websites and are only accessible through the Internet.
The significance of Mobile Application Testing
Testing applications on mobile devices is more challenging than testing web apps on the desktop due to

- Different range of mobile devices with different screen sizes and hardware configurations like a hard keypad, virtual keypad (touch screen) and trackball etc.
  Wide varieties of mobile devices like HTC, Samsung, Apple and Nokia.
- Different mobile operating systems like Android, Symbian, Windows, Blackberry and IOS.
- Different versions of operation system like iOS 5.x, iOS 6.x, BB5.x, BB6.x etc.
- Different mobile network operators like GSM and CDMA.
- Frequent updates – (like Android- 4.2, 4.3, 4.4, iOS-5.x, 6.x) – with each update a new testing cycle is recommended to make sure no application functionality is impacted.

As with any application, Mobile application testing is also very important, as the clientele is usually in millions for a certain product – and a product with bugs is never appreciated. It often results in monetary losses, legal issue, and irreparable brand image damage.
Types of Mobile App Testing:
To address all the above technical aspects, the following types of testing are performed on Mobile applications.

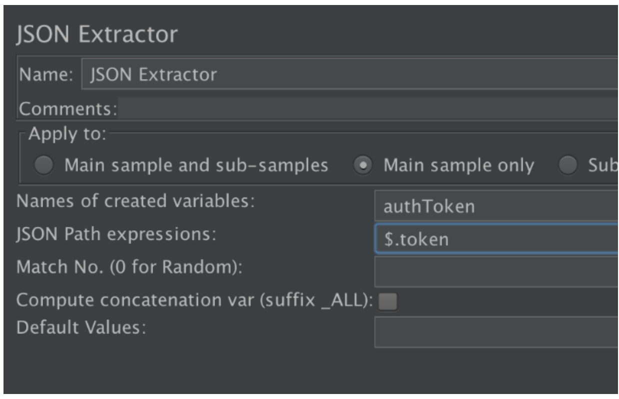
- Usability testing– To make sure that the mobile app is easy to use and provides a satisfactory user experience to the customers

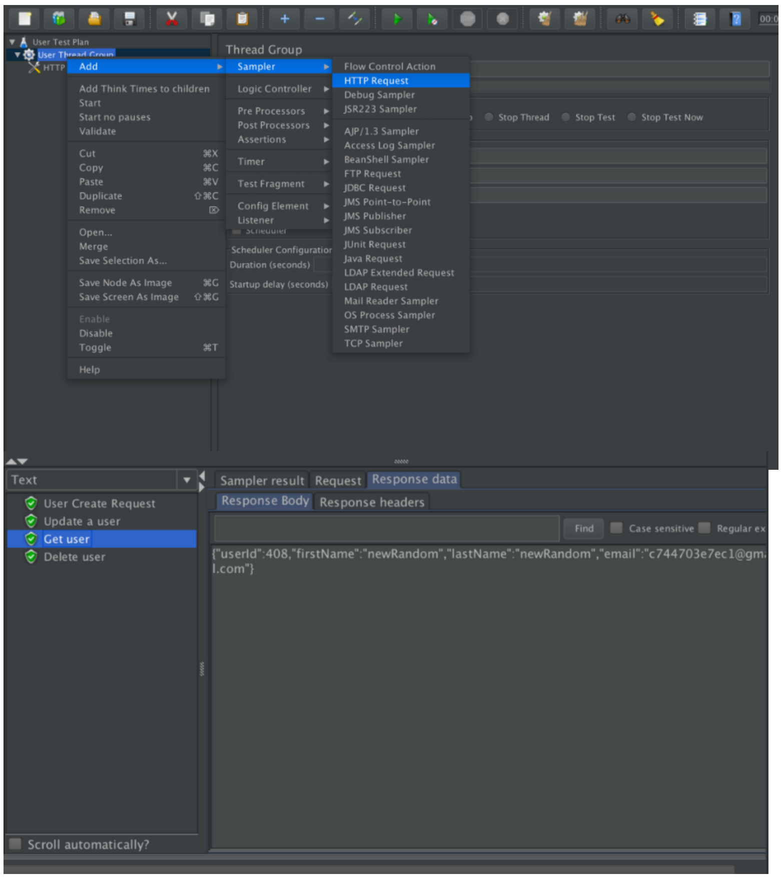
- Compatibility testing– Testing of the application in different mobiles devices, browsers, screen sizes and OS versions according to the requirements.
- Interface testing– Testing of menu options, buttons, bookmarks, history, settings, and navigation flow of the application.
- Services testing– Testing the services of the application online and offline.
- Low-level resource testing-Testing of memory usage, auto-deletion of temporary files, local database growing issues known as low-level resource testing.
- Performance testing– Testing the performance of the application by changing the connection from 2G, 3G to WIFI, sharing the documents, battery consumption, etc.
- Operational testing– Testing of backups and recovery plan if a battery goes down, or data loss while upgrading the application from a store.
- Installation tests– Validation of the application by installing /uninstalling it on the devices.
- Security Testing– Testing an application to validate if the information system protects data or not.

Mobile Application Testing Strategy
The Test strategy should make sure that all the quality and performance guidelines are met. A few pointers in this area:
1) Selection of the devices – Analyze the market and choose the devices that are widely used. (This decision mostly relies on the clients. The client or the app builders consider the popularity factor of certain devices as well as the marketing needs for the application to decide what handsets to use for testing.)
2) Emulators – The use of these is extremely useful in the initial stages of development, as they allow quick and efficient checking of the app. The emulator is a system that runs software from one environment to another environment without changing the software itself. It duplicates the features and works on the real system.
Types of Mobile Emulators

- Device Emulator- provided by device manufacturers
- Browser Emulator- simulates mobile browser environments.
- Operating systems Emulator- Apple provides emulators for iPhones, Microsoft for Windows phones and Google Android phones.

JMeter:
jMeter is an Open Source testing software. It is 100% pure Java application for load and performance testing.
jMeter is designed to cover various categories of tests such as load testing, functional testing, performance testing, regression testing, etc., and it requires JDK 5 or higher.
Load testing:
Load testing is the process of putting the load through (HTTP, HTTPS, WebSocket etc) calls on any software system to determine its behavior under normal and high load conditions. Load test helps identify maximum requests a software system can handle. It helps determine the point of a bottleneck due to which application starts degrading performance. Load testing is effective when we simulate real user scenario. We need to know how our users behave on our application. Try to replicate their behavior using different API calls and generate load through many concurrent requests at our application for an extended period of time to understand application behavior changes.


JSON Extractor
Name: JSON Extractor
Comments:
Apply to:
Main sample and sub-samples [✓ Main sample only] Sub
Names of created variables: authToken
JSON Path expressions: $.token
Match No. (0 for Random):
Compute concatenation var (suffix _ALL):
Default Values:



```
{"userId":408,"firstName":"newRandom","lastName":"newRandom","email":"c744703e7ec1@gm
l.com"}
```

Conclusion: Load testing using JMeter, Android Application testing using Appium Tools, Bugzilla Bug tracking tools Verified.