

Sr#	Topic	Page #
1	Introduction to Automation testing	1 - 13
2	History Of Selenium	14 - 20
3	Selenium IDE	21 - 35
4	Softwares for RC And Webdriver	36 - 58
5	Basics On RC(Remote Controller)	59 - 63
6	Introduction To Webdriver	64 - 69
7	WebElements	70 - 81
8	Basic Scenarios on Webdriver	82 - 107
9	Firefox Profile	108 - 123
10	Advance Scenarios	124 - 152
11	Wait Statements	153 - 157
12	Actions	158 - 164
13	XPATH & CSS	165 - 177
14	Database Testing	178 - 183
15	Remote Driver	184 - 192
16	Selenium Grid	193 - 201
17	Sikuli	2012 - 206
18	Working With Excel Sheets	207 - 262
19	Working With Unit Testing Frame Works	263 - 328
20	Starting With Frameworks	329 - 338
21	Working With Jenkins	339 - 351

# Introduction To Automation Testing

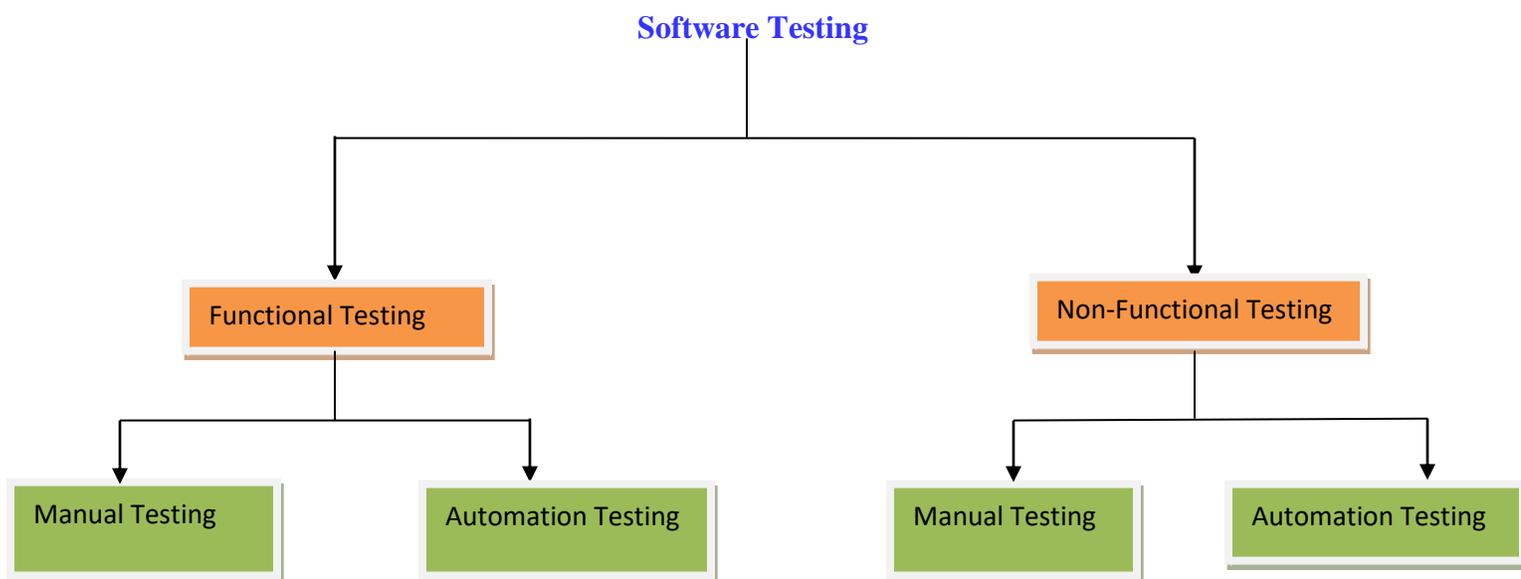
**Software Testing:** it is the process to identify hidden defects in the application with correctness, completeness and with quality of the product.

Software Testing categorized into two types:

- 1) Function Testing.
- 2) Non-Functional Testing.

Testing can be done in two ways:

- a) Manual Testing.
- b) Automation Testing.



**Manual Testing:** In Manual Testing, Testers manually execute test cases without using any automation tools. Manual testing is the most primitive of all testing types and helps find bugs in the software system.

- Any new application must be manually tested before its testing can be automated.
- Manual testing requires more effort
- Manual testing necessary to check automation feasibility.
- Manual testing does not require knowledge of any testing tool.
- One of the software Testing fundamental is **100% Automation is not possible**. This makes manual testing **imperative**.

## Disadvantages of Manual Testing:

- Manual testing requires more time or more resources, some time both
  - Performance testing is impractical in manual testing
  - Less Accuracy
  - Executing same tests again and again time taking process as well as tedious
  - Not Suitable for large scale projects and time bounded projects.
  - Batch testing is not suitable, for each and every test execution Human user interaction is mandatory.
  - Comparing large amount of data is impractical.
  - Manual test case scope is very limited, if it is automated test, scope is unlimited.
- ❖ **Test Automation:** Test Automation involves automating manual execution of test cases for enhancing speed and reliability

**Automation Testing** is nothing but simulation of Manual Actions. Automation testing includes an additional activity of writing test scripts after Test case

When you talk about Software testing or Quality assurance of any product it means we have to deliver our application bug free it means no defect in our application. In order to achieve quality we perform testing in our application it means we perform several test case it can be functional test cases and nonfunctional test cases also.

In order to save our time and resource we can automate manual test cases, which save human effort and time as well.

Scenario – If we need to test same scenario 100 times manually and each time we are checking the same result then better to automate this test case and run it number of times and verify the result.

We have different tools available in the market, which gives us the liberty to automate our application

## When Automation testing should be performed?

Answer- We can perform automation testing in our day to day testing activities it means when we have to execute some test case on regular basics then better to automate that test case

Majorly we perform automation testing for regression testing, smoke testing and sanity testing as well.

Scenario- If you have 50 scenarios that you have to execute on daily basis and you will be testing the same scenario for next couple of releases as well then manually running these test cases will be tedious task so here you can adopt automation testing.

### Benefits of Automated Testing

Benefits of Automated Testing:	
<b>Fast</b>	Automation Scripts run significantly faster than manual test execution.
<b>Reliable</b>	Test performs exactly the same operations each time they are run, in that way eliminating human error.
<b>Reusable</b>	One can reuse tests on different versions of web site or application
<b>Repeatable</b>	Testing same operation using multiple set of data. Eg:DataDriven
<b>Programmable</b>	One can program stylish tests that bring out hidden information.
<b>Comprehensive</b>	Using Automation testing, user can execute series of test cases at a time.

### Disadvantages of Test Automation

- a) 100% test Automation is not possible.
- b) All Types of testing is not possible(Ex: Usability Testing)
- c) Lack of Knowledge
- d) Lack of Debugging skills
- e) Not recommended for short term project
- f) Not recommended for dynamically changing UI designs
- g) Tools may have own their own defects
- h) Tools are technology bounded

### Which Case to Automate?

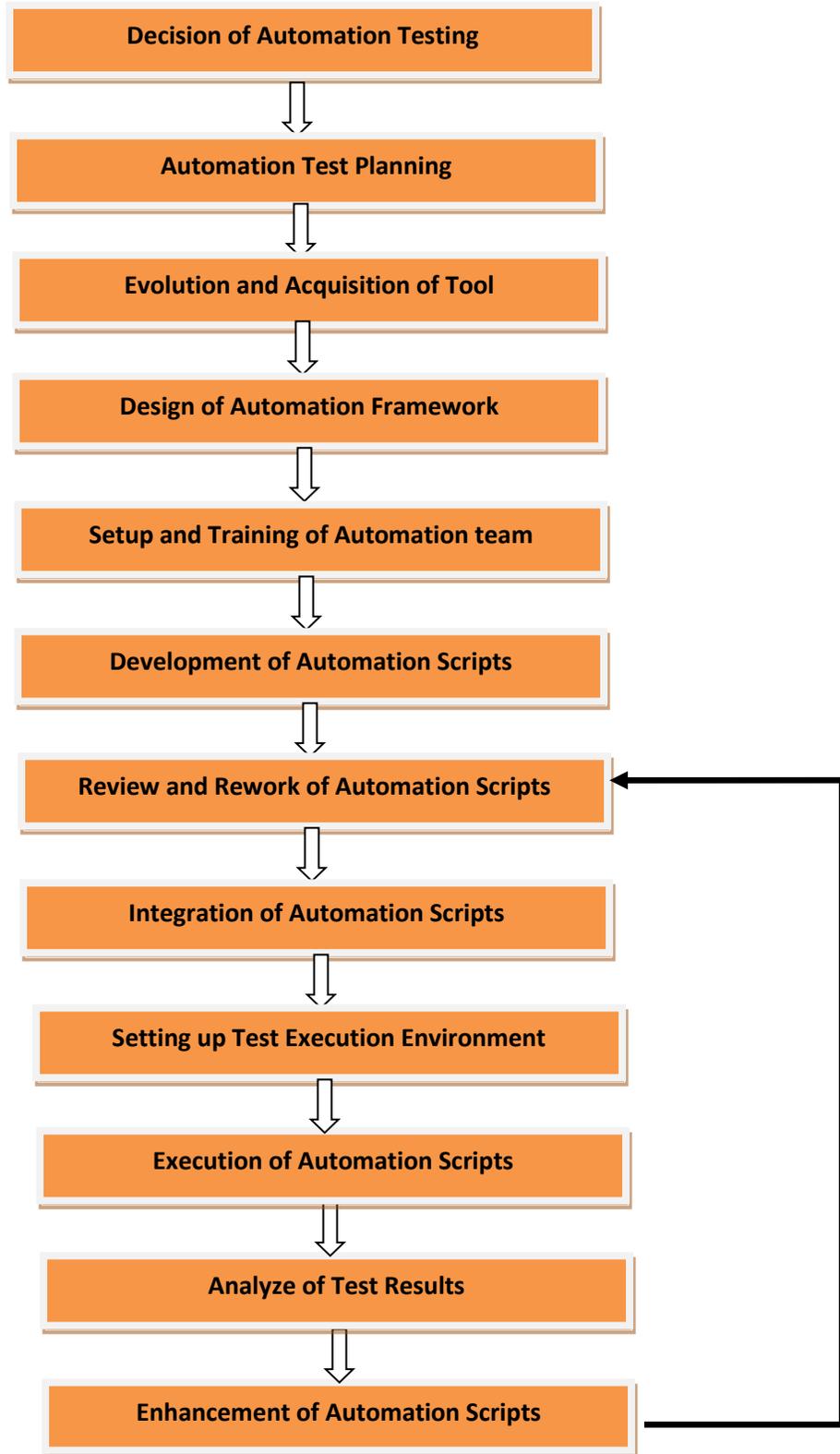
**Test cases to be automated can be selected using following criteria**

- a) High Risk -- Business Critical test cases
- b) Test cases that are executed repeatedly
- c) Test cases are that are very tedious or difficult to perform manually
- d) Test cases which are time consuming

### What type of Test case not to Automate?

- Test Cases that are newly designed and not executed manually at least once
- Test Cases for which the requirements are changing frequently
- Test cases which are executed on ad-hoc basis.

**Automation life cycle:**



- 1. Decision of automation testing:** The first phase of Automation life cycle is to decide if the application under test can be given a nod for go ahead with automation. Usually this decision is taken by the **client** and **test manager**. Lot of factors like return on **investment and future** scope of the application are considered. After all calculations, if return on investment seems to be positive, automation of the under test is approved
- 2. Automation Test planning:** In this phase the planning team which is generally comprised of **Client, Test manager, And Automation lead**; decide on the Automation test planning. This plan involves decisions like how to automate, what to automate and when to automate.
- 3. Evolution and Acquisition of tool :** In General evolution of team which consists of the client, Test manager, and Automation Lead together discuss and evaluate the various tools availability in the market

### How to Choose an Automation Tool?

Selecting the right tool can be a tricky task. Following criterion will help you select the best tool for your requirement-

- Environment Support
- Ease of use
- Testing of Database
- Object identification
- Image Testing
- Error Recovery Testing
- Object Mapping
- Scripting Language Used
- Support for various types of test - including functional, test management, mobile, etc...
- Support for multiple testing frameworks
- Easy to debug the automation software scripts
- Ability to recognize objects in any environment
- Extensive test reports and results
- Minimize training cost of selected tools

**4. Design of Automation Framework:** The automation framework is generally consists of the rules that should be followed in automation. The automation framework design defines how the script should be developed, integrated, executed and maintained.

**5. Setup and Training of Automation team:** Once the tool and framework is decided, the next step is to select team members of the automation team members of the automation team.

**6. Development of Automation scripts:** Automation team to start building scripts based on the training provided on tool, framework and application.

**7. Review & Rework of Automation scripts:** After developing the scripts the next phase involves reviewing the scripts. The factors that are considered while reviewing are if the scripts are following the guidelines defined in the framework. Once the review is done, necessary rework is done on the scripts based on the review comments.

**8. Integration of Automation scripts:** Once the rework is completed all the scripts are integrated together for the execution phase.

**9. Setting up Test Execution environment:** After Integration of scripts the execution environment is set up. This setup includes setting up of the number of systems, setting up of automation tool, setting up of the application under test and setting up the data and environment required for execution of the scripts.

**10. Execution of Automation scripts:** Final scripts (reviewed and reworked) are executed.

**11. Analysis of Test results:** After execution phase the reports generated are analyzed based on the pass/fail status of the scripts.

**12. Enhancement of Automation scripts:** In this phase, scripts might be modified because there might be few changes in the existing functionality of the application.

## Difference between QTP and selenium

### Advantages of Selenium over QTP:

Selenium	QTP
Open source, free to use, and free of charge.	Commercial.
Highly extensible	Limited add-ons
Can run tests across different browsers	Can only run tests in <b>Firefox</b> , <b>Internet Explorer</b> and <b>Chrome</b>
Supports various operating systems	Can only be used in <b>Windows</b>

Supports <b>mobile devices</b>	Supports mobile device using 3 <sup>rd</sup> party software
Can execute tests <b>while the browser is minimized</b>	Needs to have the application under test to be visible on the desktop
Can execute tests <b>in parallel.</b>	Can only execute in parallel but using Quality Center which is again a paid product.

### Advantages of QTP over Selenium:

QTP	Selenium
Can test <b>both web and desktop applications</b>	Can only test web applications
Comes with a <b>built-in object repository</b>	Has no built-in object repository
<b>Automates faster than Selenium</b> because it is a fully featured IDE.	Automates at a slower rate because it does not have a native IDE and only third party IDE can be used for development
Data-driven testing is easier to perform because <b>it has built-in global and local data tables.</b>	Data-driven testing is more cumbersome since you have to rely on the programming language's capabilities for setting values for your test data
<b>Can access controls within the browser</b> (such as the Favorites bar, Address bar, Back and Forward buttons, etc.)	Cannot access elements outside of the web application under test
Provides professional <b>customer support</b>	No official user support is being offered.
Has native capability to <b>export test data</b> into external formats	Has no native capability to export runtime data onto external formats
Parameterization Support is in built	Parameterization can be done via

	programming but is difficult to implement.
Test Reports are generated automatically	No native support to generate test /bug reports.

- ❖ Though clearly, QTP has more advanced capabilities, Selenium outweighs QTP in three main areas:
- ❖ **Cost**(because Selenium is completely free)
- ❖ **Flexibility**(because of a number of programming languages, browsers, and platforms it can support)
- ❖ **Parallel testing**(something that QTP is capable of but only with use of Quality Center)

### Challenges in Automation Testing

1. **Selection of Automation Tool** – Today there are n number of automation tools available in the market and choosing a good automation tool is one of the major challenges that an organization often faces. This is majorly because there are several commercial tools which are expensive, there are open source tools which might not be reliable and there are tools of which an organization may not have sufficient expertise to make optimum use of.
2. **No Defined Process for Executing Automation Project within an organization** – Automation is like a project execution, wherein you start with Requirement, then you design the framework as per your requirements and finally you roll it out for Test Case Execution. Lack of systematic approach and process will make successful automation really tough. As we have processes, guidelines and checklist defined for our development project; we should also have guidelines, processes and checklist available for Automation as well.
3. **Availability of Right Resources** – The right set of resources is a must when you are doing automation. This means the resources should be skilled enough to design and code robust scripting, so that it requires minimum time for debugging during maintenance.
4. **Commitment from Customer or Management** – Automation is time consuming and resource intensive task. Customer or management commitment is required if one wants to get the real benefit of automation. This paper will highlight the approach wherein you can maximize your benefits within reasonable period of time.

# Software Test Tools

Software Test tool is Software used to automate Software Test process.

## I) Functional and Regression Test Tools

### 1) HP UFT (Formerly QTP)

- ❖ UFT - Unified Functional Test Tool from HP is used to automate Functional Regression Test cases of Desktop and Web Applications.
- ❖ It supports MS Windows operating environment only, doesn't support UNIX operating environment.
- ❖ VBScript is used for enhancing tests in UFT.
- ❖ UFT has IDE and Programming interface to create and execute Tests.
- ❖ UFT has an integrated MS Access database engine to support Database Testing.
- ❖ UFT is an Object based Test tool, based on software objects it supports test automation, but for Database testing no front-end object is required.
- ❖ UFT has 2 types of license, one is Seat license another is Concurrent license.
- ❖ UFT can integrate with ALM/QC for Test Management.

### 2) Selenium

- > Selenium is a suite of tools to automate Web browsers, supports only web based applications.
- > Selenium supports Functional and Regression Test automation of Web Applications.
- > It is Open source software, released under Apache 2.0 License. Anybody can download and use with free of cost.
- > Selenium Supports various programming languages(Java, C#, PHP, Perl, Python and Ruby) for enhancing test cases, we can use any one of the supporting language.
- > Selenium supports various browsers (IE, Mozilla Firefox, Google Chrome, Safari etc..)
- > Selenium supports various Operating environments (MS Windows, UNIX and Macintosh etc..)
- > Selenium supports JUnit and TestNG frameworks for grouping test cases and generating test reports

### **3) IBM RFT (Rational Functional Tester)**

- > RFT provides automated testing capabilities for functional, regression, GUI, and data-driven testing.
- > RFT supports a range of applications, such as web-based, .Net, Java, Siebel, SAP, terminal emulator-based applications, PowerBuilder, Ajax, Adobe Flex, Dojo Toolkit, GEF, Adobe PDF documents, zSeries, iSeries, and pSeries.
- > RFT supports Windows and Linux operating environments.
- > Rational Functional Tester's object-oriented technology is used to identify various objects by their internal object properties and not by their screen coordinates.

### **4) SilkTest**

- > SilkTest supports industry standard languages like VB.NET, C# and Java, or you can use Silk Test's own 4Test.
- > SilkTest supports Visual Studio and Eclipse IDEs and is easy to integrate with unit testing and acceptance testing tools.
- > SilkTest supports cross browser testing.
- > SilkTest was originally developed by Segue Software which was acquired by Borland in 2006. Borland was acquired by Micro Focus International in 2009.

### **5) TestComplete**

- > TestComplete is a functional automated testing tool developed by SmartBear.
- > TestComplete supports Python, VBScript, JScript, DelphiScript, C++Script, and C#Script to create tests.
- > TestComplete supports Desktop, Web and Mobile applications test automation.
- > TestComplete supports Distributed Testing, It can run several automated tests across separate workstations or virtual machines

## II) Performance Test Tools

### 1) HP LoadRunner

> HP LoadRunner is Performance Test tool and supports MS Windows and Linux (Load Generator only) operating environments.

> LoadRunner is a Protocol based test tool.

### 2) JMeter

> JMeter is an Open Source testing software.

> JMeter is a Java desktop application with a graphical interface, It can run on any environment that accepts a Java virtual machine, for example – Windows, Linux, Mac, etc.

### 3) SilkPerformer

> SilkPerformer is a software performance testing tool across web, mobile and enterprise applications.

## III) Test Management Tools

### 1) HP ALM

> HP ALM (Application Life Cycle Management Tool) supports various phases of the software development life cycle.

> ALM is a web based tool used to manage the application life cycle right from project planning, requirements gathering, until testing and deployment.

> ALM doesn't have any programming interface, It is only for documentation and generating reports.

> ALM can be used for all types of Software applications.

> HP ALM can be integrated with UFT (formerly QTP) and LoadRunner.

> ALM can be accessed by all Developers, Testers, Business Analysts, Project Managers, and Product Owners.

## **2) Jira**

- > Jira is a project management and issue tracking tool by Atlassian, Inc., and It is platform independent.
- > Jira is written in Java, and It is Open source software.
- > We can access JIRA cloud site via a mobile device. You have to just use the URL of the JIRA cloud site in your mobile web browser.

## **IV) Defect Management Tools**

### **1) Bugzilla**

- > Bugzilla is a Web based defect management tool.
- > Bugzilla is an open source bug tracking system, It is written in Perl and uses MYSQL database.

### **2) Mantis**

- > Mantis defect / issue tracking system is written in PHP, and works on various databases including MySQL, MS SQL, PostgreSQL.

# History of selenium

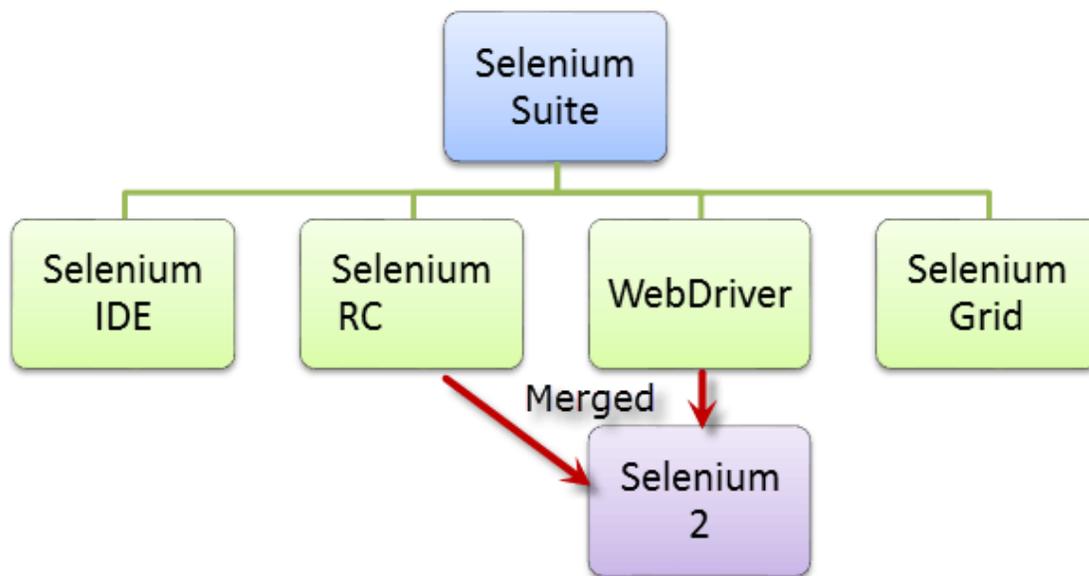
## Introduction to Selenium

### What is selenium?

**Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms.** It is quite similar to HP Quick Test Pro (QTP) only that Selenium focuses on automating **web-based applications**.

Selenium is not just a single tool but a suite of software's, each catering to different testing needs of an organization. **It has four components.**

- Selenium Integrated Development Environment (IDE)
- Selenium Remote Control (RC)
- WebDriver
- Selenium Grid



At the moment, Selenium RC and WebDriver are merged into a single framework to form **Selenium 2**. Selenium 1, by the way, refers to Selenium RC

## History of Selenium

### Who developed Selenium?

Since Selenium is a collection of different tools, it had different developers as well. Below are the key persons who made notable contributions to the Selenium Project

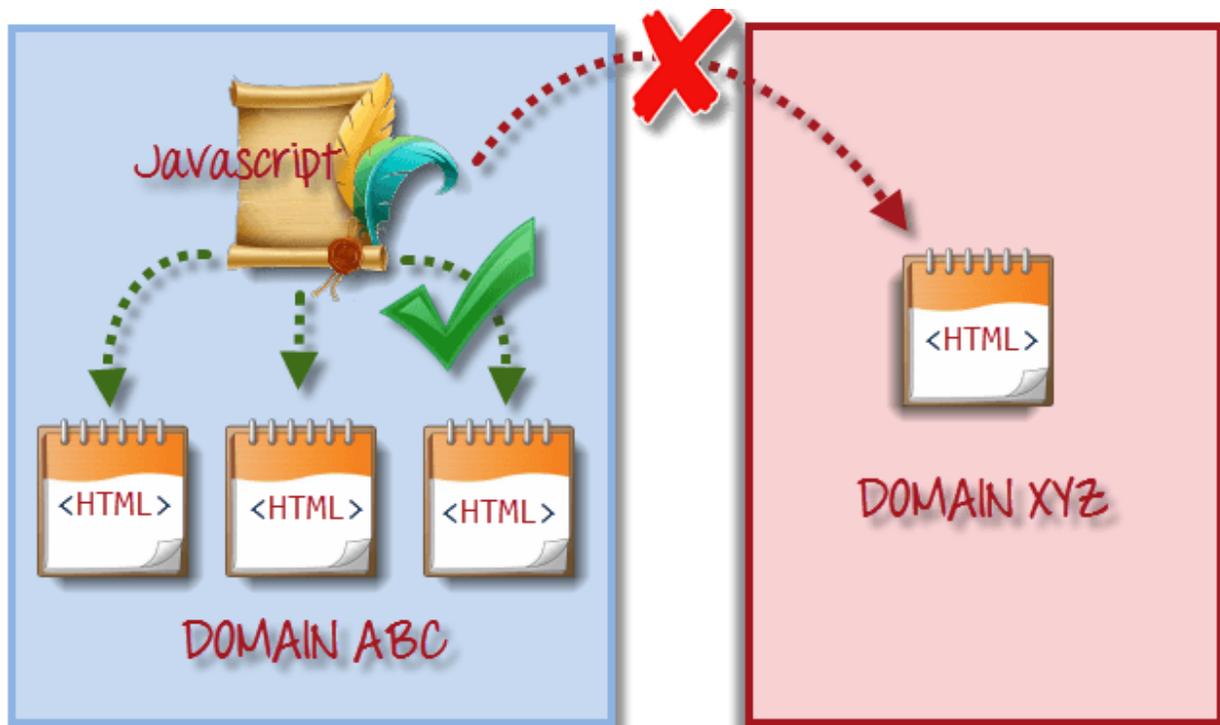
Primarily, Selenium was **created by Jason Huggins in 2004**. An engineer at ThoughtWorks, he was working on a web application that required frequent testing. Having realized that the repetitious manual testing of their application was becoming more and more inefficient, he created a JavaScript program that would automatically control the browser's actions. He named this program as the "**JavaScriptTestRunner**."

Seeing potential in this idea to help automate other web applications, he made JavaScriptRunner open-source which was later re-named as **Selenium Core**.



### The Same Origin Policy Issue:

**Same Origin policy prohibits JavaScript code from accessing elements from a domain that is different from where it was launched.** Example, the HTML code in [www.google.com](http://www.google.com) uses a JavaScript program "randomScript.js". The same origin policy will only allow randomScript.js to access pages within google.com such as [google.com/mail](http://google.com/mail), [google.com/login](http://google.com/login), or [google.com/signup](http://google.com/signup). However, it cannot access pages from different sites such as [yahoo.com/search](http://yahoo.com/search) or [guru99.com](http://guru99.com) because they belong to different domains.



*under same origin policy, a JavaScript program can only access pages on the same domain where it belongs. it cannot access pages from different domains*

This is the reason why prior to Selenium RC, testers needed to install local copies of both Selenium Core (a JavaScript program) and the web server containing the web application being tested so they would belong to the same domain

### **Selenium Features:**

- **OPEN SOURCE (IT IS AVAILABLE FOR FREE OF COST. YOU CAN DOWNLOADED FROM INTERNET)**
- **SUPPORTS WEB BASED APPLICATIONS.**
- **SUPPORTS CROSS BROWSER TESTING. (YOU CAN TEST THE APPLICATION ON DIFFERENT BROWSERS LIKE INTERNET EXPLORER, FIREFOX, CHROME, AND SAFARI.)**
- **SUPPORTS MULTIPLE LANGUAGES(JAVA, PERL, PYTHON, RUBY, PHP, C#)**
- **SUPPORTS MULTIPLE OPERATING SYSTEMS(WINDOWS, LINUX, IOS, ANDROID)**
- **SUPPORTING FOR FUNCTIONAL & REGRESSION TESTING.**
- **SUPPORTS FLASH BASED APPLICATION TESTING.**
- **SUPPORTS AJAX BASED APPLICATION TESTING.**
- **SUPPORTS DATABASE APPLICATION TESTING**
- **SUPPORTS MOBILE BASED APPLICATION TESTING.**

## INTERVIEW QUESTIONS

Q1: what type of applications can automate using Selenium?

Supports Web-based Applications.

Q2: Suppose my Web Based Application developed in .Net, can I automate the application using java or python or Ruby?

Yes. Irrespective of language, you can automate using any of the language which supports Selenium like Java, Perl, python, Ruby, C#.

## Selenium Components

- Selenium IDE
- Selenium R.C (Remote Controller)
- Selenium WebDriver
- Selenium Grid

## Birth of Selenium Remote Control (Selenium RC):

Unfortunately; testers using Selenium Core had to install the whole application under test and the web server on their own local computers because of the restrictions imposed by the **same origin policy**. So another ThoughtWork's engineer, **Paul Hammant**, decided to create a server that will act as an HTTP proxy to "trick" the browser into believing that Selenium Core and the web application being tested come from the same domain. This system became known as the **Selenium Remote Control** or **Selenium 1**.



Paul Hammant

## Birth of Selenium Grid:

Selenium Grid was developed by **Patrick Lightbody** to address the need of minimizing test execution times as much as possible. He initially called the system "**Hosted QA**." It was capable of capturing browser screenshots during significant stages, and also of **sending out Selenium commands to different machines simultaneously**.



### Birth of Selenium IDE:

**Shinya Kasatani** of Japan created **Selenium IDE**, a Firefox extension that can automate the browser through a record-and-playback feature. He came up with this idea to further increase the speed in creating test cases. He donated Selenium IDE to the Selenium Project in **2006**.



### Birth of WebDriver:

Simon Stewart created **WebDriver** circa 2006 when browsers and web applications were becoming more powerful and more restrictive with JavaScript programs like **Selenium Core**. It was the first cross-platform testing framework that could control the browser from the OS level.



### Birth of Selenium 2:

In **2008**, the whole Selenium Team decided to merge WebDriver and Selenium RC to form a more powerful tool called **Selenium 2**, with **WebDriver being the core**. Currently, Selenium RC is still being developed but only in maintenance mode. Most of the Selenium Project's efforts are now focused on Selenium 2.

### So, Why the Name Selenium?:

It came from a joke which Jason cracked one time to his team. Another automated testing framework was popular during Selenium's development, and it was by the company called **Mercury Interactive** (yes, the company who originally made QTP before it was acquired

by HP). Since Selenium is a well-known antidote for Mercury poisoning, Jason suggested that name. His teammates took it, and so that is how we got to call this framework up to the present.



### **What are the technical challenges with selenium?**

As you know Selenium is a free ware open source testing tool. There are many challenges with Selenium.

1. Selenium supports only web based applications.
2. It doesn't support any non web based (Like Win 32, Java Applet, Java Swing, .Net Client Server etc) applications.
3. When you compare selenium with QTP, Silk Test, Test Partner and RFT, there are many challenges in terms of maintainability of the test cases.
4. Since Selenium is a freeware tool, there is no direct support if one is in trouble with the support of applications.
5. There is no object repository concept in Selenium, so maintainability of the objects is very high
6. There are many challenges if one have to interact with Win 32 windows even when you are working with Web based applications.
7. Bitmap comparison is not supported by Selenium.
8. Any reporting related capabilities, you need to depend on third party tools.
9. You need to learn any one of the native language like (.Net, Java, Perl, Python, PHP, Ruby) to work efficiently with the scripting side of selenium.

# Selenium IDE

# Selenium IDE

## Selenium IDE (Integrated Development Environment):

- It is Firefox Add on
- It is prototyping tool
- Selenium Test default format is html

### Features:

- ❖ Record and Playback Test cases
- ❖ Edit Test Cases
- ❖ Execute a Test case
- ❖ Execute a Test Suite
- ❖ Type Test Script using Elements locaters and selenium commands
- ❖ Debugging Test Cases
- ❖ Export Test cases to other programming and scripting languages

### Installation steps:

Launch Firefox and navigate to <http://seleniumhq.org/download/>. Under the **Selenium IDE** section, click on the link that shows the current version number.

#### Selenium IDE

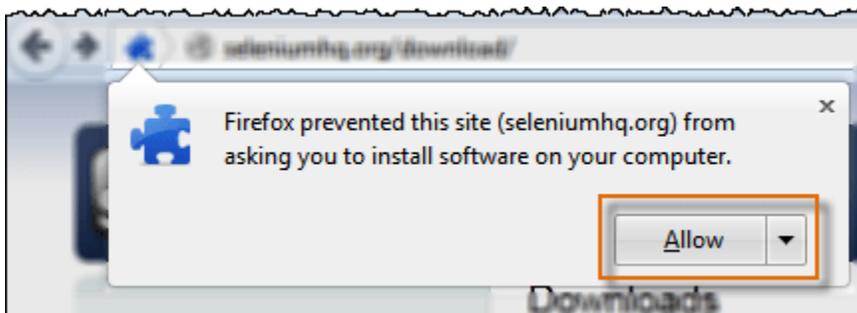
---

Selenium IDE is a Firefox plugin which records and plays back user interactions with the browser. Use this to either create simple scripts or assist in exploratory testing. It can also export Remote Control or WebDriver scripts, though they tend to be somewhat brittle and should be overhauled into some sort of Page Object-y structure for any kind of resiliency.

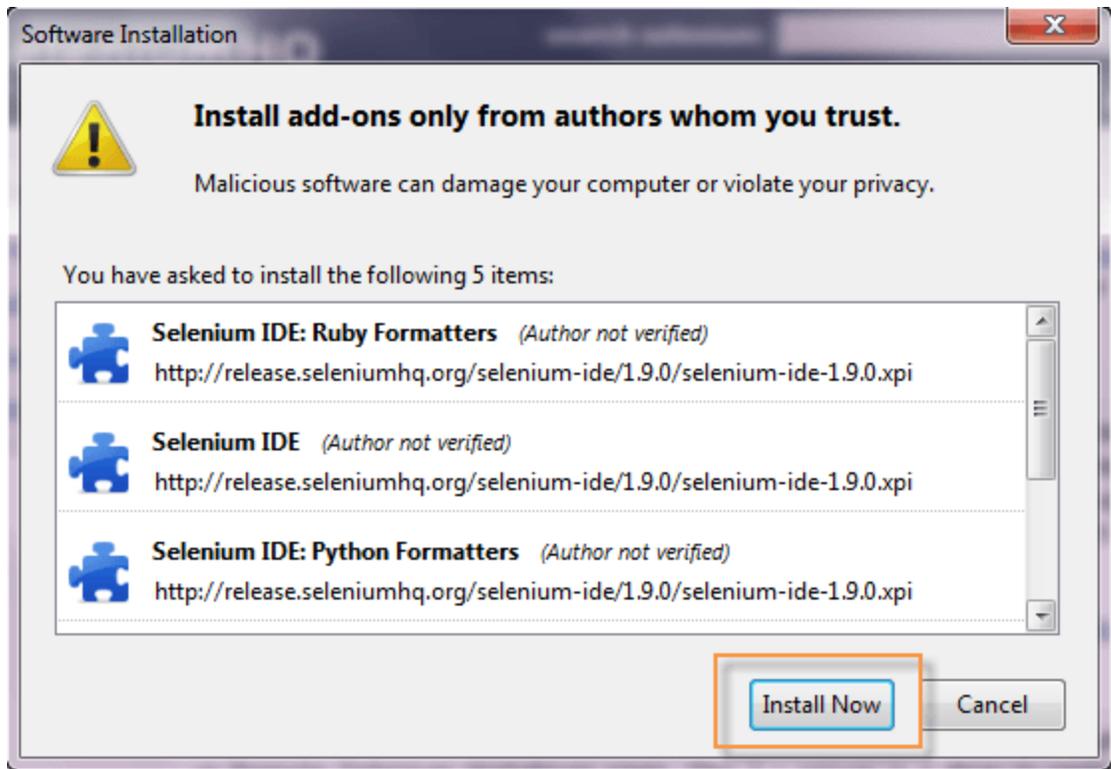
Download latest released version [2.9.0](#) released on 09/Mar/2015 or view the [Release Notes](#) and then [install some plugins](#).

Download previous version [2.8.0](#) released on 29/Sep/2014.

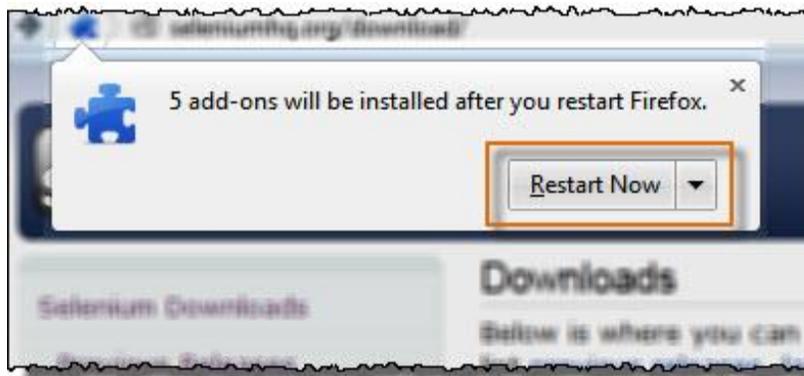
For [security](#), a Firefox notification will pop up. Click on "Allow."



Wait until Firefox completes the download and then click "**Install Now.**"



Wait until installation is completed. In the pop-up window, click "**Restart Now.**"

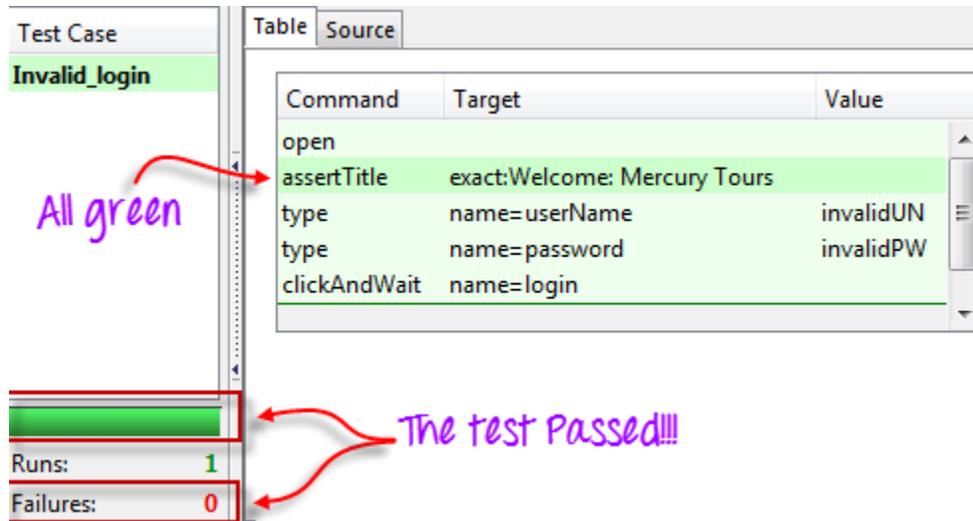


After Firefox has restarted, **launch Selenium IDE** using either of two ways:

- By pressing **Ctrl+Alt+S**
- By clicking on the **Firefox menu button > Web Developer > Selenium IDE**

What are the two modes of views in Selenium IDE?

Either Selenium IDE can be opened as a pop up window or in side bar



In selenium IDE what are the element locators that can be used to locate elements on web page?

In selenium there are mainly 4 locators that are used

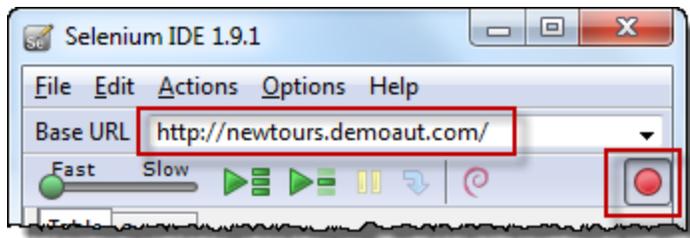
- X-path locators
- Css locators
- Html id
- Html name

### Create script by Recording

Let us now create our first test script in Selenium IDE using the most common method - by recording. Afterwards, we shall execute our script using the playback feature.

#### Step 1

- Launch Firefox and Selenium IDE.
- Type the value for our Base URL: <http://newtours.demoaut.com/>.
- Toggle the Record button on (if it is not yet toggled on by default).

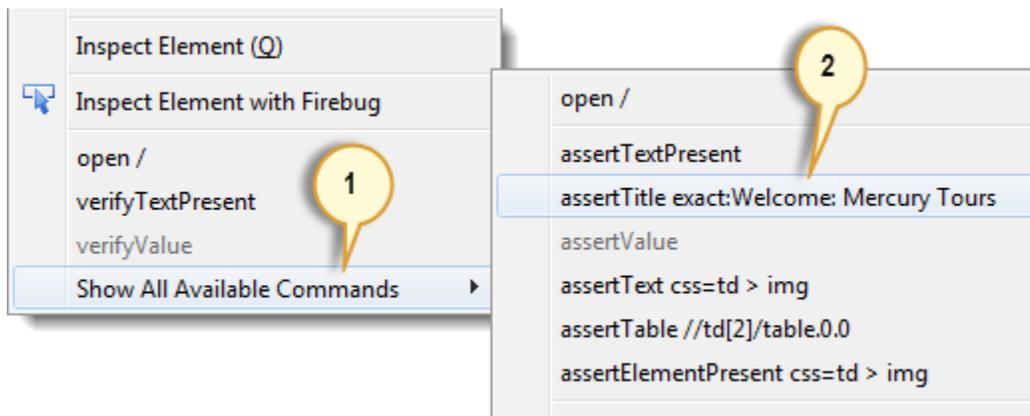


## Step 2

In Firefox, navigate to <http://newtours.demoaut.com/>. Firefox should take you to the page

## Step 3

- Right-click on any blank space within the page, like on the Mercury Tours logo on the upper left corner. This will bring up the Selenium IDE context menu. Note: Do not click on any hyperlinked objects or images
- Select the "Show Available Commands" option.
- Then, select "assertTitle exact:Welcome: Mercury Tours". This is a command that makes sure that the page title is correct.



## Step 4

- In the "User Name" text box of Mercury Tours, type an invalid username, "invalidUN".
- In the "Password" text box, type an invalid password, "invalidPW".

**Find A Flight**

Registered users can **sign-in here** to find the lowest fare on participating airlines.

User Name:

Password:

**Sign-In** →

**Destinations**

Comm...	Target	Value
open	/	
assertTitle	exact>Welcome: Mercury Tours	
type	name=username	invalidUN
type	name=password	invalidPW

*Your Editor should now look like this*

### Step 5

- Click on the "Sign-In" button. Firefox should take you to this page.

**one cool summer** ARUBA

[SIGN-ON](#) [REGISTER](#) [SUPPORT](#) [CONTACT](#)

**SIGN-ON** ↗

Welcome back to Mercury Tours! Enter your user information to access the member-only areas of this site. If you don't have a log-in, please fill out the [registration form](#).

User Name:

Password:

**SUBMIT**

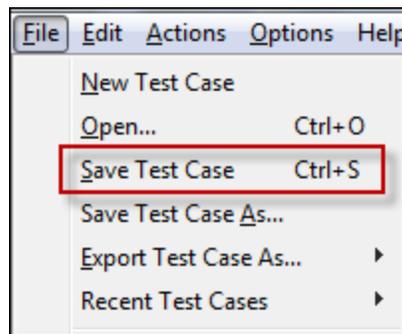
### Step 6

Toggle the record button off to stop recording. Your script should now look like the one shown below.

Command	Target	Value
open		
assertTitle	exact>Welcome: Mercury Tours	
type	name=username	invalidUN
type	name=password	invalidPW
clickAndWait	name=login	

### Step 7

Now that we are done with our test script, we shall save it in a test case. In the File menu, select "Save Test Case". Alternatively, you can simply press Ctrl+S.



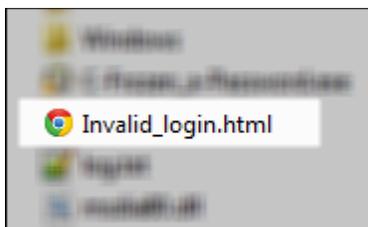
### Step 8

- Choose your desired location, and then name the test case as "Invalid\_login".
- Click the "Save" button.



### Step 9.

Notice that the file was saved as HTML.



## Step 10.

Go back to Selenium IDE and click the Playback button to execute the whole script. Selenium IDE should be able to replicate everything flawlessly.

The screenshot shows the Selenium IDE interface. On the left, the 'Test Case' pane displays 'Invalid\_login' with a green bar and the handwritten text 'All green'. On the right, the 'Table' pane shows a list of commands:

Command	Target	Value
open		
assertTitle	exact>Welcome: Mercury Tours	
type	name=username	invalidUN
type	name=password	invalidPW
clickAndWait	name=login	

At the bottom, the status bar shows 'Runs: 1' and 'Failures: 0', with a green bar above it and the handwritten text 'The test Passed!!!' pointing to the status bar.

## Introduction to Selenium Commands - Selenese

- Selenese commands can have up to a maximum of two parameters: target and value.
- Parameters are not required all the time. It depends on how many the command will need.
- For a complete reference of Selenese commands, click [here](#)

## 3 Types of Commands

<b>Actions</b>	<p>These are commands that directly interact with page elements.</p> <p>Example: the "click" command is an action because you directly interact with the element you are clicking at.</p> <p>The "type" command is also an action because you are putting values into a text box, and the text box shows them to you in return. There is a two-way interaction between you and the text box.</p>
----------------	--

<b>Accessors</b>	<p>They are commands that allow you to store values to a variable.</p> <p>Example: the "storeTitle" command is an accessor because it only "reads" the page title and saves it in a variable. It does not interact with any element on the page.</p>
<b>Assertions</b>	<p>They are commands that verify if a certain condition is met.</p> <p><b>3 Types of Assertions</b></p> <ul style="list-style-type: none"> <li>• <b>Assert.</b> When an "assert" command fails, the test is stopped immediately.</li> <li>• <b>Verify.</b> When a "verify" command fails, Selenium IDE logs this failure and continues with the test execution.</li> <li>• <b>WaitFor.</b> Before proceeding to the next command, "waitFor" commands will first wait for a certain condition to become true. <ul style="list-style-type: none"> <li>○ If the condition becomes true within the waiting period, the step passes.</li> <li>○ If the condition does not become true, the step fails. Failure is logged, and test execution proceeds to the next command.</li> <li>○ By default, timeout value is set to 30 seconds. You can change this in the Selenium IDE Options dialog under the General tab.</li> </ul> </li> </ul>

## Assert vs. Verify:

### ASSERT

test execution  
was halted in  
this part

Command	Target	Value
open		
assertTitle	Welcome: Venus Tours	
type	name=username	invalidUN
type	name=password	invalidPW
clickAndW...	name=login	

Command

Target  Find

Value

Log	Reference	UI-Element	Rollup	Info	Clear
[info]				Executing:  open	
[info]				Executing:  assertTitle   Welcome: Venus Tours	
[error]				Actual value 'Welcome: Mercury Tours' did not match 'Welcome: Venus Tours'	

no further logs were  
displayed after this error  
message, meaning that  
execution indeed stopped

### What is the difference between an assert and a verify with Selenium commands?

Assert: Will fail and abort the current test execution.

Verify: Will fail and continue to run the test execution.

## VERIFY

Execution continued despite the error

The screenshot shows the Selenium IDE interface. The top part is a table of commands:

Command	Target	Value
open		
verifyTitle	Welcome: Venus Tours	
type	name=username	invalidUN
type	name=password	invalidPW
clickAndW...	name=login	

Below the table are input fields for Command, Target, and Value, and a Find button. The bottom part of the screenshot shows the Log window with the following entries:

```
[info] Executing: |verifyTitle | welcome: venus  
Tours | |  
[error] Actual value 'Welcome: Mercury  
Tours' did not match 'Welcome: Venus Tours'  
[info] Executing: |type | name=username |  
invalidUN |  
[info] Executing: |type | name=password |  
invalidPW |
```

Commands after the Failed verify command were still executed

### Advantages of Selenium IDE

- Easily you can generate scripts using Record option.
- No need to identify WebElement properties externally.
- Automatically, it identifies the properties of elements while recording.
- You can record and play **Independent type of scenarios**.
- No need to know any programming knowledge to use this tool.
- Easy to configure from the net in the Firefox Browser.
- It is not only a time saver but also an excellent way of learning scripts syntax.
- We can insert comments in the middle of the script for better understanding and debugging.

### Disadvantages of Selenium IDE

- It supports only Firefox browser.
- You can't record into other languages other than HTML.
- You can't run the scripts in various browsers other than Firefox browser.
- It does not directly support loops and conditions..
- Reading from external files like .txt, .xls is not possible.
- No detailed Results reports.

In the **Advantages of selenium IDE** section, mentioned independent type of scenarios. **What is the meaning of Independent scenario?**

**Example:** I am explaining this concept with **Gmail Application**.

Let us consider there are three scenarios like **Login, Compose a mail, and Logout**.

**Scenario 1: Login Gmail Application.**

For this scenario, required only username & password. If you provide both correct values on these two fields then Login is successful.

**Scenario 2: Compose a Mail.**

If you want to execute this scenario, first you should execute **Login**, if that is successful, then only you can do compose a mail, else it is not possible.

Here **compose a Mail** is **dependent on Login Scenario**. Similarly **logout scenario** also dependant on **Login Scenario**.

### **What are the capabilities of Selenium IDE?**

Selenium IDE (Integrated Development Environment) works similar to commercial tools like QTP, Silk Test and Test Partner etc.

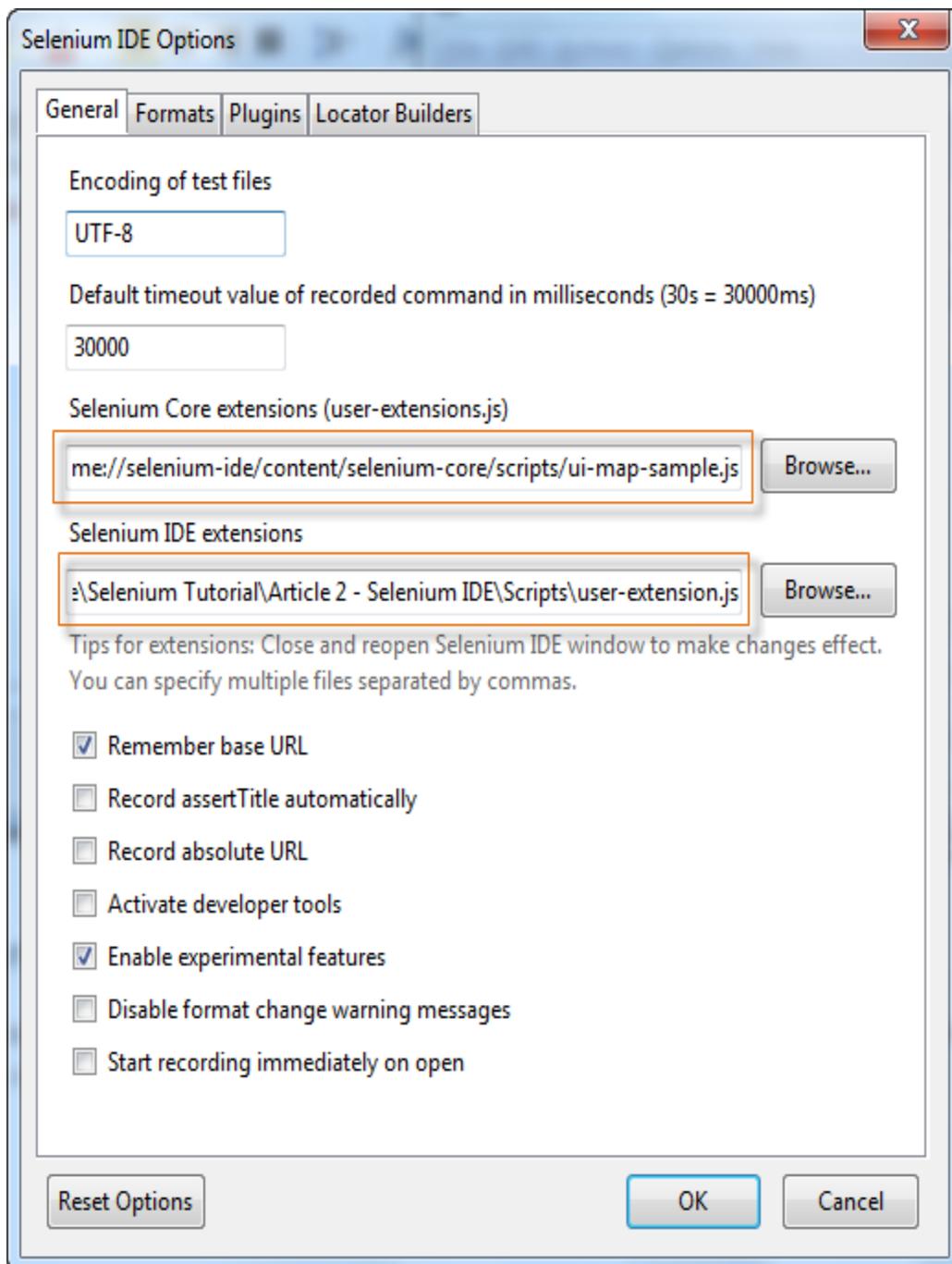
The below points describes well about Selenium IDE.

1. Selenium IDE is a Firefox add-on.
2. Selenium IDE can support recording the clicks, typing, and other actions to make a test cases.
3. Using Selenium IDE, a tester can play back the test cases in the Firefox browser.
4. Selenium IDE supports exporting the test cases and suites to Selenium RC.
5. Debugging of the test cases with step-by-step can be done.
6. Breakpoint insertion is possible.
7. Page abstraction functionality is supported by Selenium IDE.
8. Selenium IDE can support an extensibility capability allowing the use of add-ons or user extensions that expand the functionality of Selenium IDE

### **User Extensions:**

Selenium IDE can support user extensions to provide advanced capabilities. User extensions are in the form of [JavaScript](#) files. You install them by specifying their absolute path in either of these two fields in the Options dialog box.

- Selenium Core extensions (user-extensions.js)
- Selenium IDE extensions



### Sample code for user-extensions.js:

You can find the sample code for user-extensions.js in the below link. It is implemented in the Java script

Link: <https://code.google.com/p/selenium/source/browse/javascript/selenium-core/scripts/user-extensions.js.sample>

## **What are the challenges with Selenium IDE?**

Selenium-IDE does not directly support:

1. conditional statements
2. Iteration or looping
3. Logging and reporting of test results
4. error handling, particularly unexpected errors
5. database testing
6. Test case grouping
7. re-execution of failed tests
8. test case dependency
9. capture screenshots on test failures
10. Results Report generations

## **How to execute a single line command from Selenium IDE?**

Single line command from Selenium IDE can be executed in two ways

1. Right click on the command in Selenium IDE and select "Execute This Command".
2. Select the command in Selenium IDE and press "X" key on the keyboard.

## **How to export the tests from Selenium IDE to Selenium RC in different languages?**

**From selenium IDE, the test cases can be exported into the languages:**

1. .Net,
2. Java,
3. Perl,
4. Python,
5. PHP,
6. Ruby

The below mentioned steps can explain how to export the test cases:

1. Open the test case from Selenium IDE.
2. Select File -> Export Test Case As.

## **How to insert a break point in Selenium IDE?**

Break point can be set in two ways in Selenium IDE:

1. Right click on the command in Selenium IDE and select "Toggle Break Point".
2. Select the command in Selenium IDE and press "B" key on the keyboard.

3. If you want to clear the break point once again press "B" key on the keyboard.
4. You can set multiple break points in Selenium IDE.

#### **How to insert a comment in Selenium IDE?**

Comments in Selenium IDE can be set in two ways.

1. Right click on the command in Selenium IDE and select "Inert New Comment".
2. If you want to comment an existing line, we need to follow the below mentioned steps.
  - a. Select the source tab in IDE.
  - b. Select the line which you want to comment.
  - c. Assume that if you want to comment a open command you need to write like below mentioned code

```
<tr>
<!--
<td>open&l/td>
<td>/node/304/edit&l/td>
<td></td>
-->
</tr>
```

#### **What is the use of context menu in Selenium IDE?**

It allows the user to pick from a list of assertions and verifications for the selected location.

#### **What is a regular expression? How you can use regular expressions in Selenium?**

A regular expression is a special text string used for describing a search pattern. In Selenium IDE regular expression can be used with the keyword- **regexp**: as a prefix to the value and patterns needs to be included for the expected values.

# Softwares For RC And WebDriver

## Software's for RC and WebDriver

### ➤ Steps to Download and Install Java Development Kit (JDK) on Windows:

- ❖ First of all you need to install **JDK** (Java development kit) in your system. So your next question will be “*how to download and install Java*“. [Click here](#) to download Java and install it in your system as per given installation guide over there or follow the below mentioned steps.
- ❖ 1) Visit the [Java downloads page on Oracle's](#) website to find the **JDK environment** download. Scroll down until you find **Java SE Latest Version** and download **JDK**.

Overview Downloads Documentation Community Technologies Training

### Java SE Downloads

Next Releases (Early Access) Embedded Use Previous Releases

 **DOWNLOAD** ↓

Java Platform (JDK) 7u51

 **DOWNLOAD** ↓

JDK 7u51 & NetBeans 7.4

#### Java Platform, Standard Edition

**Java SE 7u51**  
This release includes important security fixes. Oracle strongly recommends that all Java SE 7 users upgrade to this release.  
[Learn more](#) ▶

This release includes important security fixes. Oracle strongly recommends that all Java SE 7 users upgrade to this release.

[Learn more](#) ▶

#### Which Java package do I need?

- **JDK:** (Java Development Kit). For Java Developers. Includes a complete JRE plus tools for developing, debugging, and monitoring Java applications.
- **Server JRE:** (Server Java Runtime Environment) For deploying Java applications on servers. Includes tools for JVM monitoring and tools commonly required for server applications, but does not include browser integration (the Java plug-in), auto-update, nor an installer. [Learn more](#) ▶
- **JRE:** (Java Runtime Environment). Covers most end-users needs. Contains everything required to run Java applications on your system.

#### JDK

[DOWNLOAD](#) ↓

#### JDK 7 Docs

- [Installation Instructions](#)
- [ReadMe](#)
- [Release Notes](#)
- [Oracle License](#)
- [Java SE Products](#)

#### Server JRE

[DOWNLOAD](#) ↓

#### Server JRE 7 Docs

- [Installation Instructions](#)
- [ReadMe](#)
- [Release Notes](#)
- [Oracle License](#)
- [Java SE Products](#)

#### JRE

[DOWNLOAD](#) ↓

#### JRE 7 Docs

- [Installation Instructions](#)
- [ReadMe](#)
- [Release Notes](#)
- [Oracle License](#)
- [Java SE Products](#)

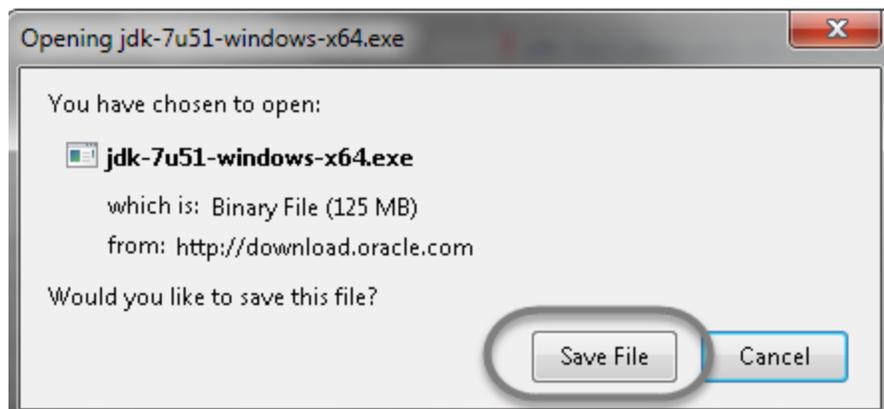
2) Once you have selected download, **accept** the terms of service and choose the correct OS corresponding for the specific JDK. (Windows, Mac, Linux, etc.)

**Java SE Development Kit 7u51**  
 You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

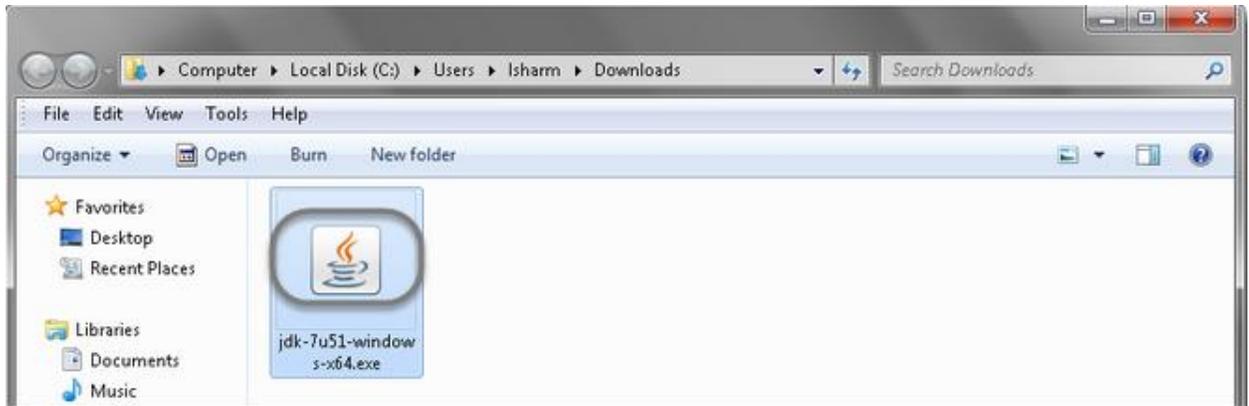
**Accept License Agreement**  Decline License Agreement

Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	67.7 MB	<a href="#">jdk-7u51-linux-arm-vfp-hflt.tar.gz</a>
Linux ARM v6/v7 Soft Float ABI	67.68 MB	<a href="#">jdk-7u51-linux-arm-vfp-sflt.tar.gz</a>
Linux x86	115.65 MB	<a href="#">jdk-7u51-linux-i586.rpm</a>
Linux x86	132.98 MB	<a href="#">jdk-7u51-linux-i586.tar.gz</a>
Linux x64	116.96 MB	<a href="#">jdk-7u51-linux-x64.rpm</a>
Linux x64	131.8 MB	<a href="#">jdk-7u51-linux-x64.tar.gz</a>
Mac OS X x64	179.49 MB	<a href="#">jdk-7u51-macosx-x64.dmg</a>
Solaris x86 (SVR4 package)	140.02 MB	<a href="#">jdk-7u51-solaris-i586.tar.Z</a>
Solaris x86	95.13 MB	<a href="#">jdk-7u51-solaris-i586.tar.gz</a>
Solaris x64 (SVR4 package)	24.53 MB	<a href="#">jdk-7u51-solaris-x64.tar.Z</a>
Solaris x64	16.28 MB	<a href="#">jdk-7u51-solaris-x64.tar.gz</a>
Solaris SPARC (SVR4 package)	139.39 MB	<a href="#">jdk-7u51-solaris-sparc.tar.Z</a>
Solaris SPARC	98.19 MB	<a href="#">jdk-7u51-solaris-sparc.tar.gz</a>
Solaris SPARC 64-bit (SVR4 package)	23.94 MB	<a href="#">jdk-7u51-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	18.33 MB	<a href="#">jdk-7u51-solaris-sparcv9.tar.gz</a>
Windows x86	123.64 MB	<a href="#">jdk-7u51-windows-i586.exe</a>
Windows x64	125.46 MB	<a href="#">jdk-7u51-windows-x64.exe</a>

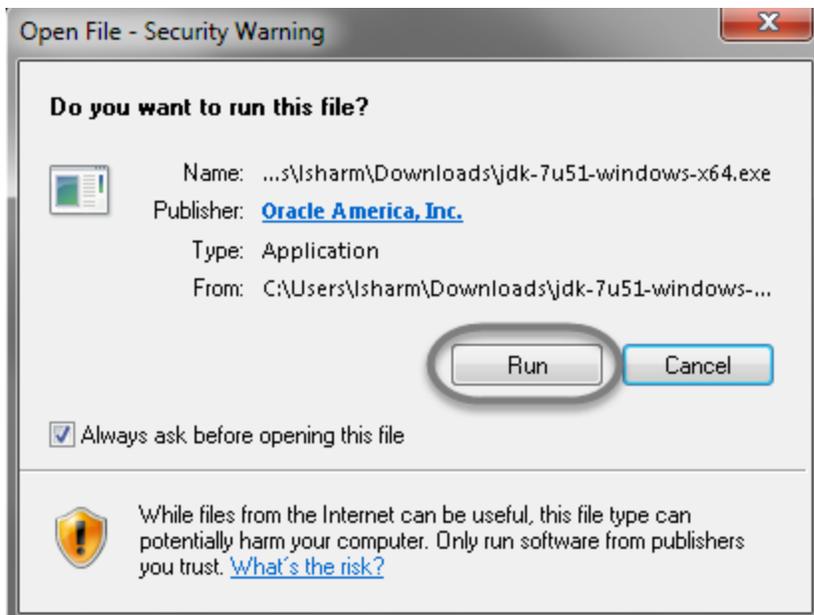
3) Save the `.exe` file to your disk.



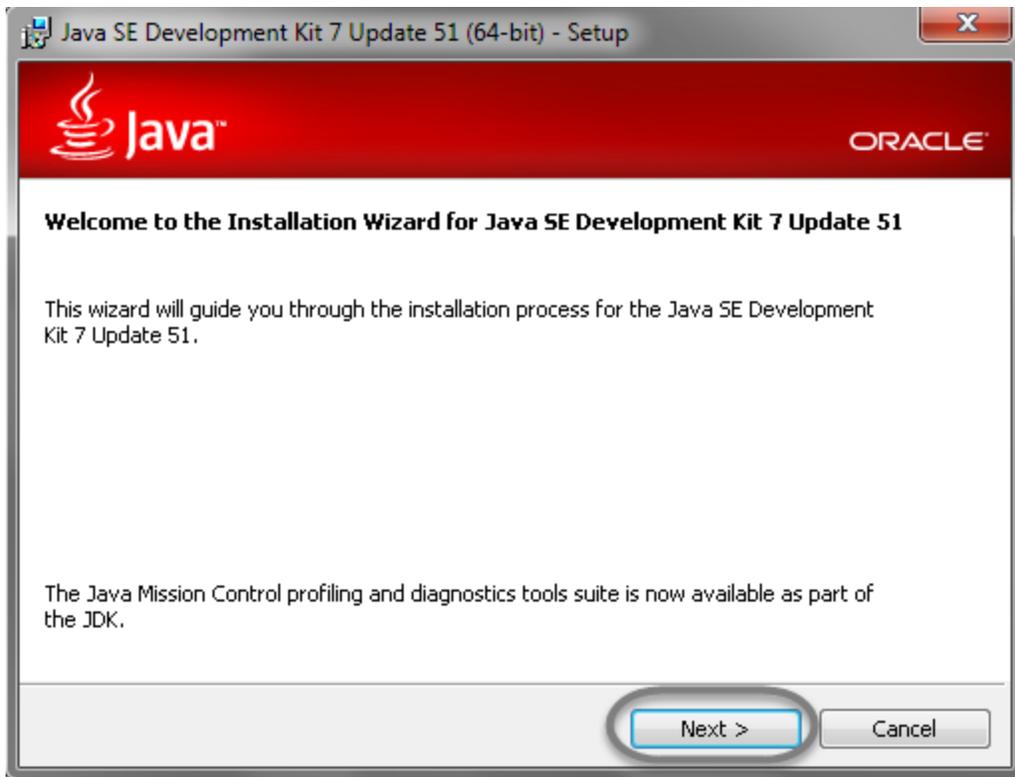
4) Once the download is complete, **double click** the file to begin the installation of JDK.



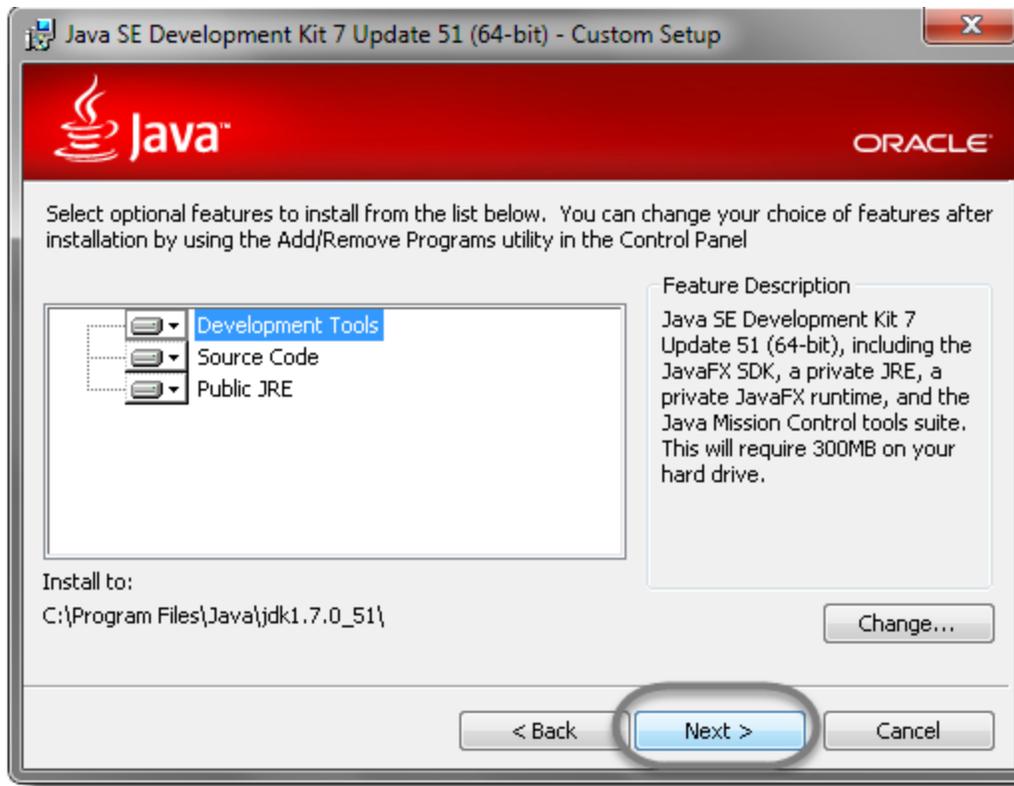
5) To run the installer, click **Run**.



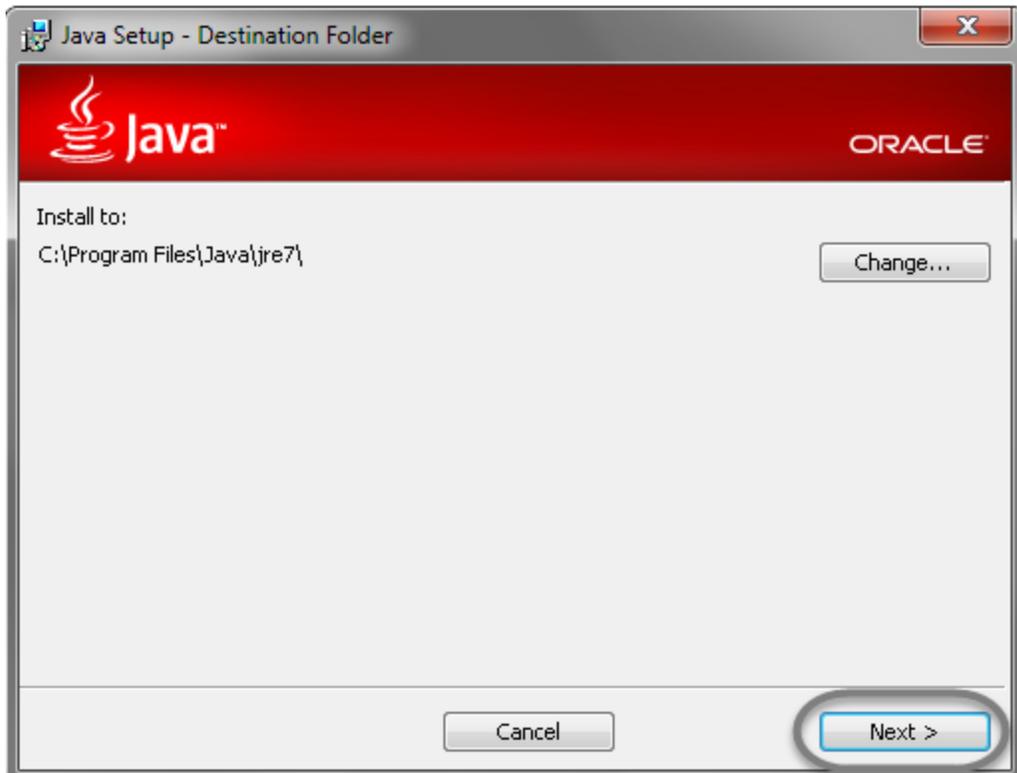
6) The installation process starts. Click the **Next** button to continue the installation.



7) On the next screen you will encounter some options. Just leave these alone and click **Next** unless you know what you are doing.



8) After the initial installation is done, a pop up asking you where your source java files will be. You can choose to change where you want to keep your folder but it's best to stick with what you were given first. Click **Next** to continue.



9) Let the installation finish.



10) A few brief dialogs confirm the last steps of the installation process; click **Close** on the last dialog. This will complete Java installation process.



## Download and Start Eclipse IDE

Download Eclipse for Java Developers, extract and save it in any drive. It is totally free. You can run 'eclipse.exe' directly so you do not need to install Eclipse in your system.

- 1) Go to <http://www.eclipse.org/downloads>.

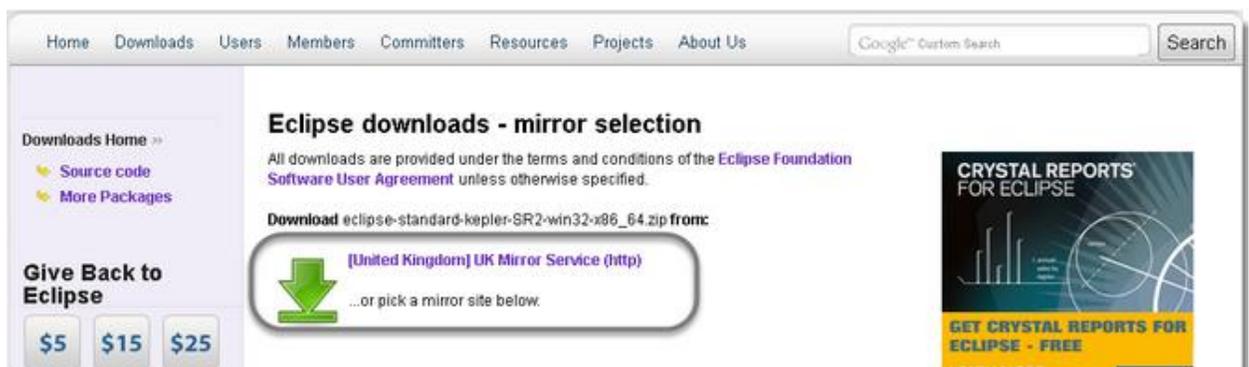


The screenshot shows the Eclipse Downloads website. The navigation bar includes links for Home, Downloads, Users, Members, Committers, Resources, Projects, and About Us. The main heading is "Eclipse Downloads". Below this, there are tabs for "Packages" and "Developer Builds". A dropdown menu is set to "Eclipse Kepler (4.3.2) SR2 Packages for Windows". Two packages are listed:

- Eclipse Standard 4.3.2**, 200 MB, Downloaded 872,492 Times. Description: "The Eclipse Platform, and all the tools needed to develop and debug it: Java and Plug-in Development Tooling, Git and CVS...". Download options: Windows 32 Bit, Windows 64 Bit.
- Eclipse IDE for Java EE Developers**, 250 MB, Downloaded 435,307 Times. Description: "Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn...". Download options: Windows 32 Bit, Windows 64 Bit.

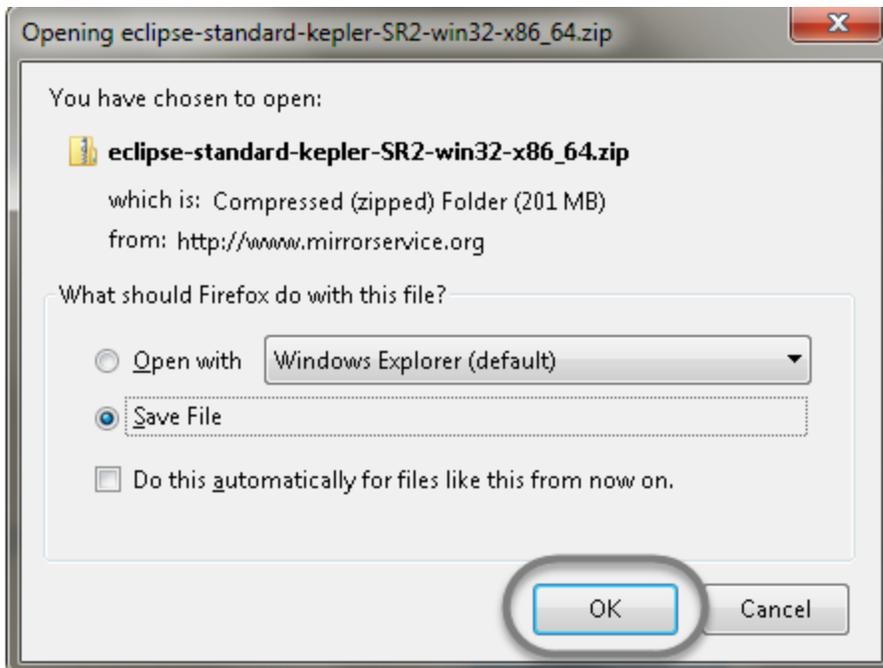
A "Filter Packages" dropdown is also visible.

- 2) For Windows users, you will have to know what type of version of your OS you have. If your computer is a **64 bit** Windows, select Windows 64 and if you have a **32 bit** Windows, select Windows 32 bit.

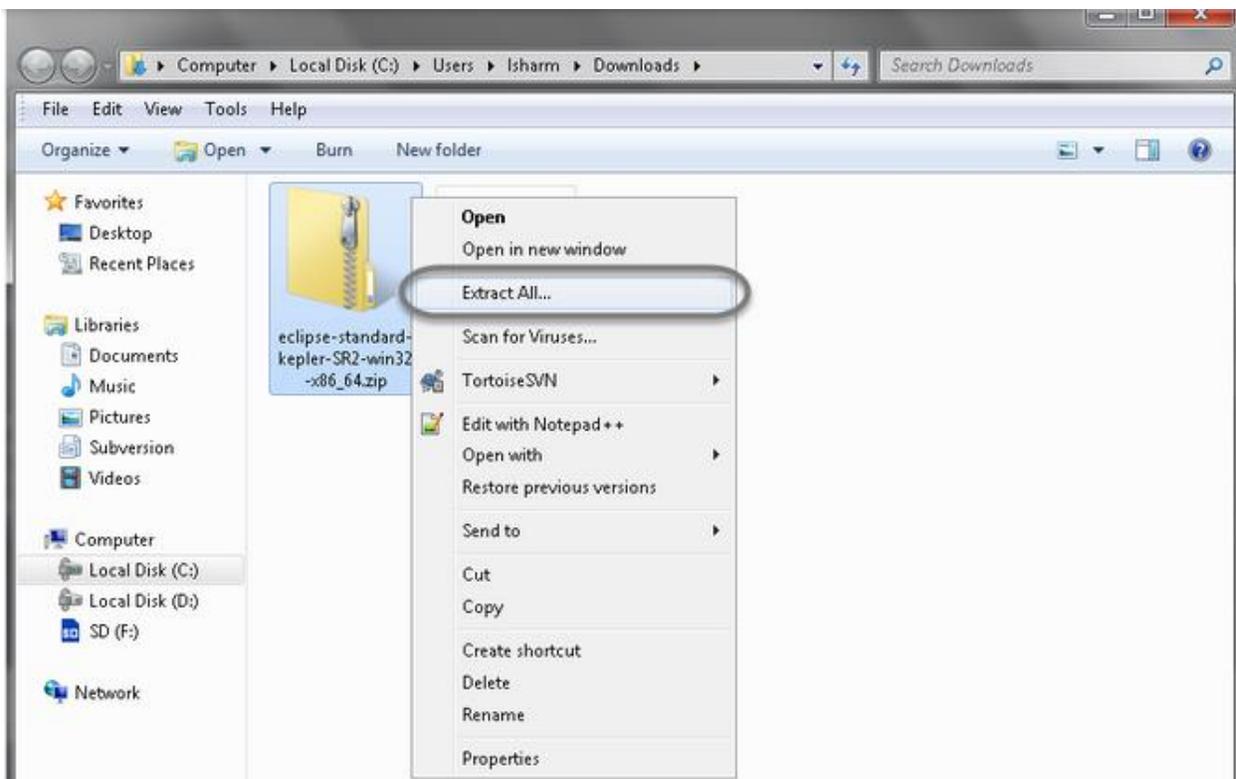


The screenshot shows the "Eclipse downloads - mirror selection" page. It includes a navigation bar with links for Home, Downloads, Users, Members, Committers, Resources, Projects, and About Us. The main heading is "Eclipse downloads - mirror selection". Below this, there is a notice: "All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified." The download link is "Download eclipse-standard-kepler-SR2-win32-x86\_64.zip from: [United Kingdom] UK Mirror Service (http)". A large green download arrow icon is present. Below the icon, it says "...or pick a mirror site below." On the left side, there is a "Give Back to Eclipse" section with buttons for \$5, \$15, and \$25. On the right side, there is an advertisement for "CRYSTAL REPORTS FOR ECLIPSE" with the text "GET CRYSTAL REPORTS FOR ECLIPSE - FREE".

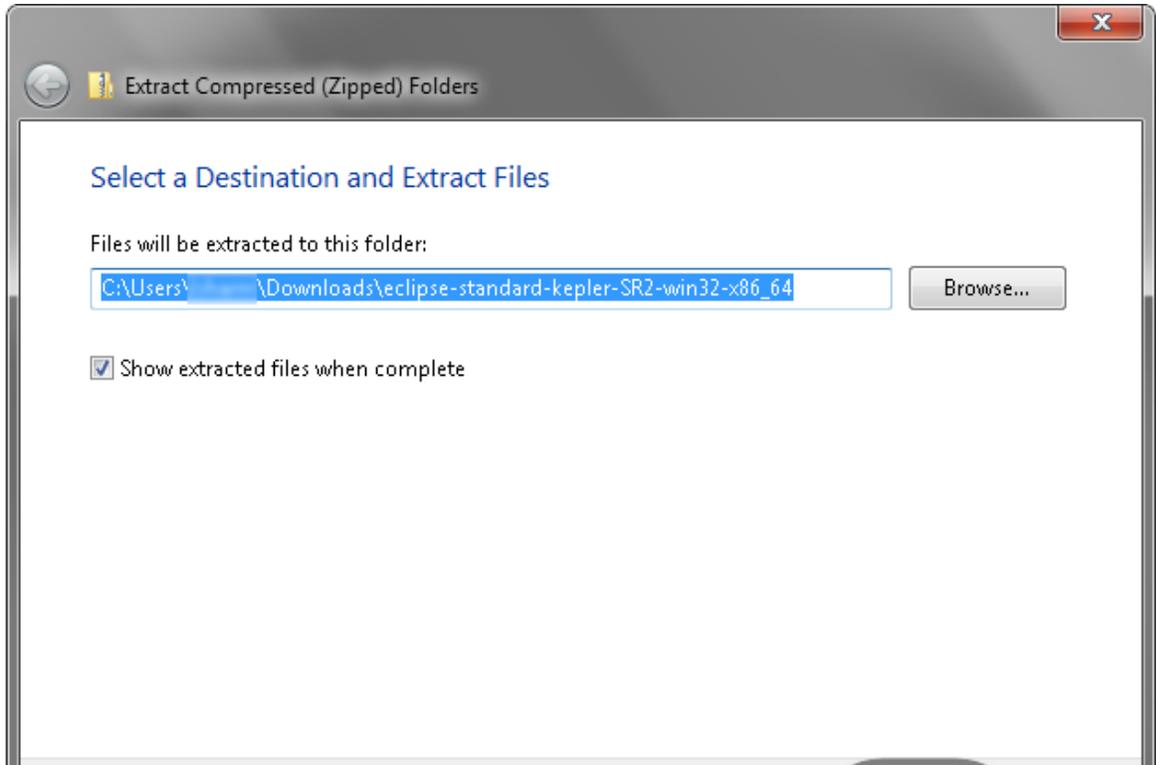
- 3) Save the .zip file to your disk.



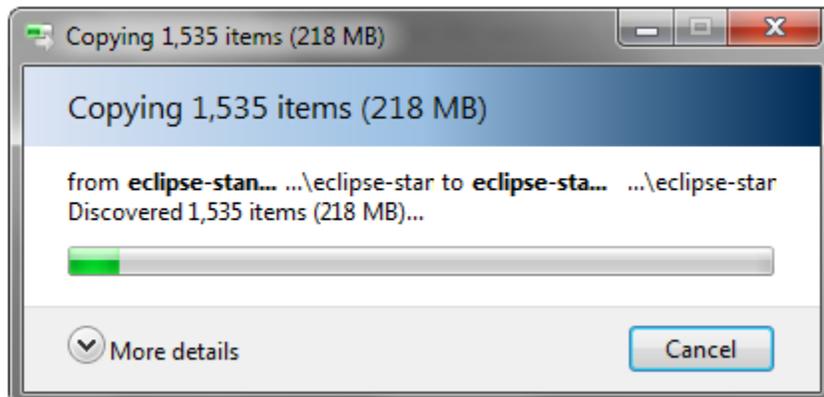
- 4) Once you have downloaded the Eclipse archive you will need to **extract** the zip file, which will create the unzipped Eclipse folder.



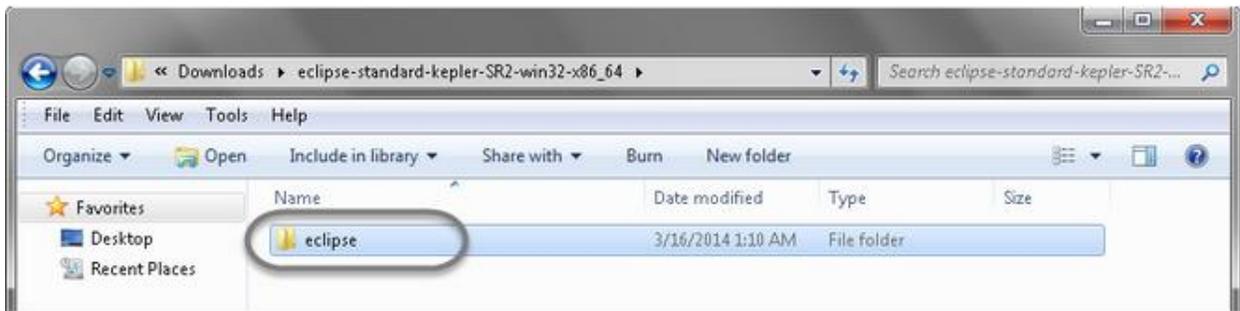
- 5) You may want to **Extract** the archive to the root of C: drive, thus creating the folder “C:eclipse”, or just moved the extracted eclipse folder to the root of C: drive if you extracted it already. I prefer to leave it as it is.



- 6) Let the extraction process finished.

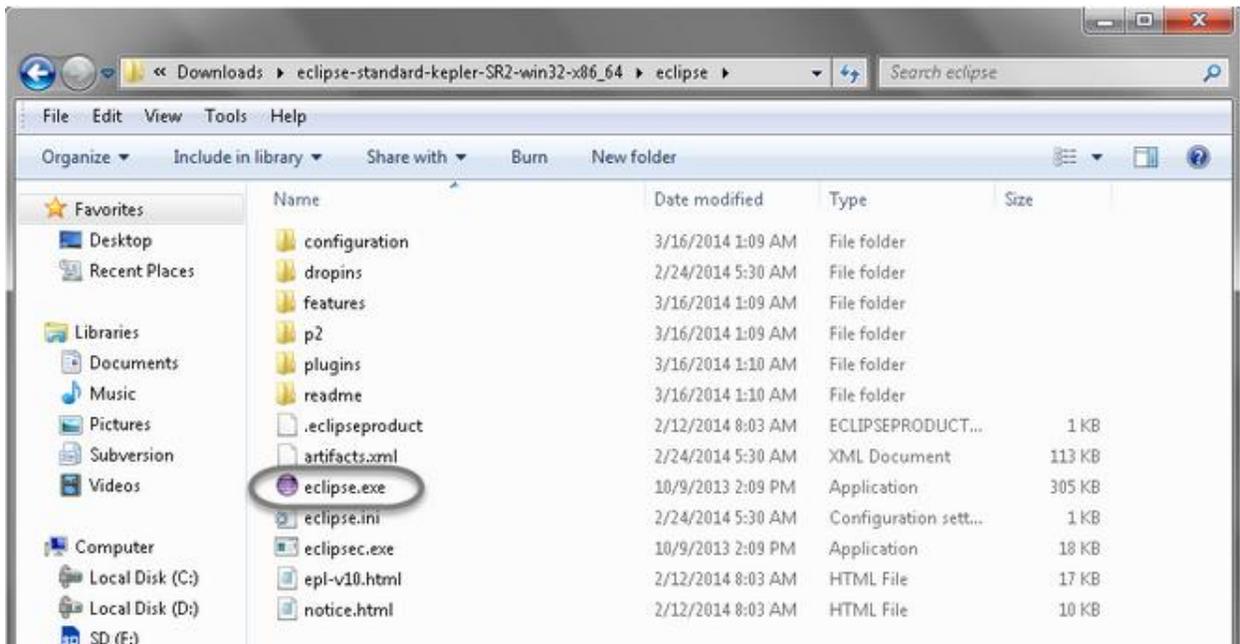


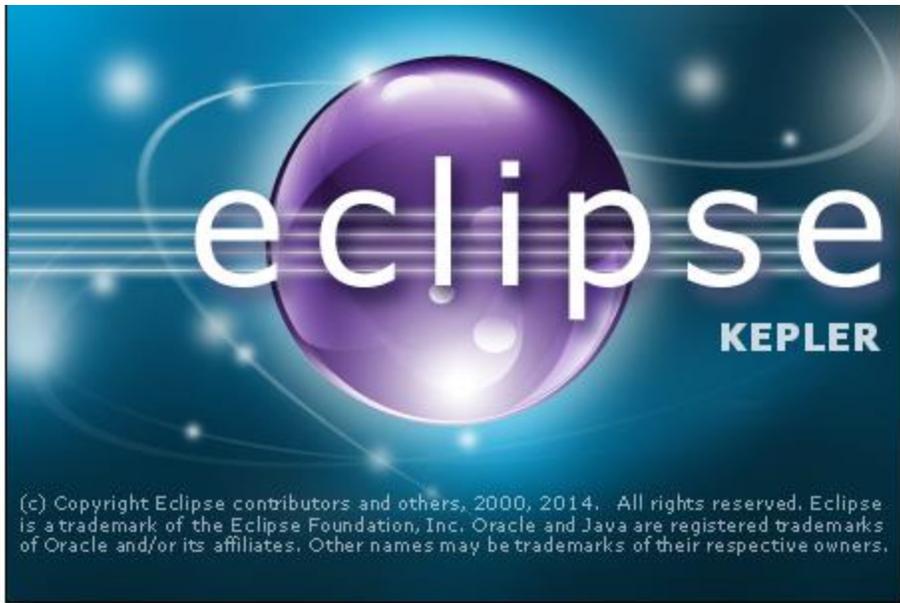
- 7) **Open** the eclipse folder.



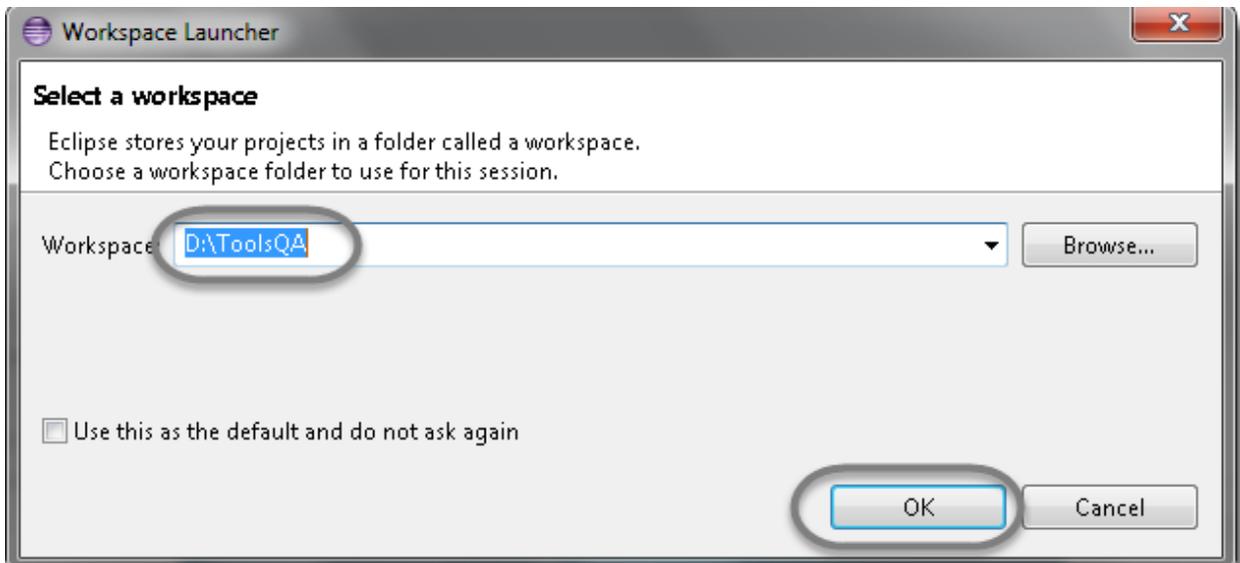
- 8) Since Eclipse IDE does not have any installer, there will be a file inside the Eclipse folder named **eclipse.exe**. You can **double click** on the file to run Eclipse.

**Note:**(This step is not required, but it's strongly recommended.) Right-click the Eclipse Icon and press "Send To" -> "Desktop (Create Shortcut)." Now you will be able to launch Eclipse from your desktop.

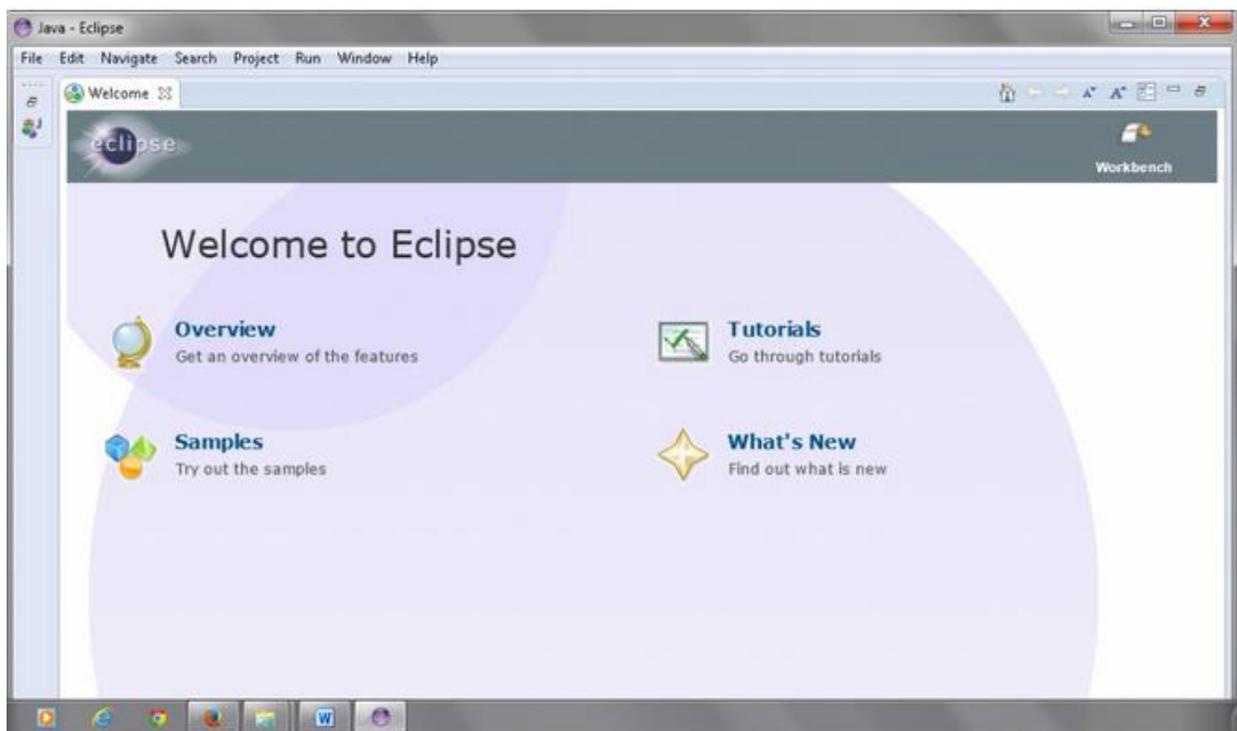




- 9) Create a workspace folder where you will contain all the program files you create. You can choose whatever place you want for your workspace, but it's easiest to just use the default you're given. I like to choose my own workplace location and will place all my Toolsqa tutorial projects under it.



10) You may see the window like this, this is the Welcome window for Eclipse.



## Download WebDriver Java client

Selenium webdriver supports many languages and each language has its own client driver. Here we are configuring selenium 2 with java so we need 'webdriver Java client driver'.

1) [Click here](#) to go on WebDriver Java client driver download page for webdriver download file. On that page click on '**Download**' link of java client driver as shown in the below image.

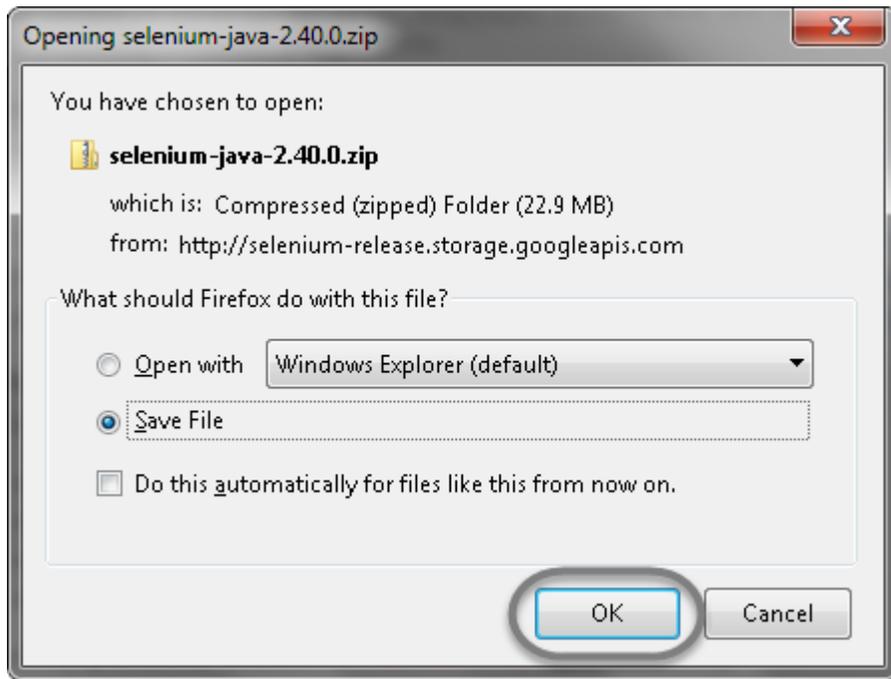
### Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote WebDriver) or create local Selenium WebDriver script you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

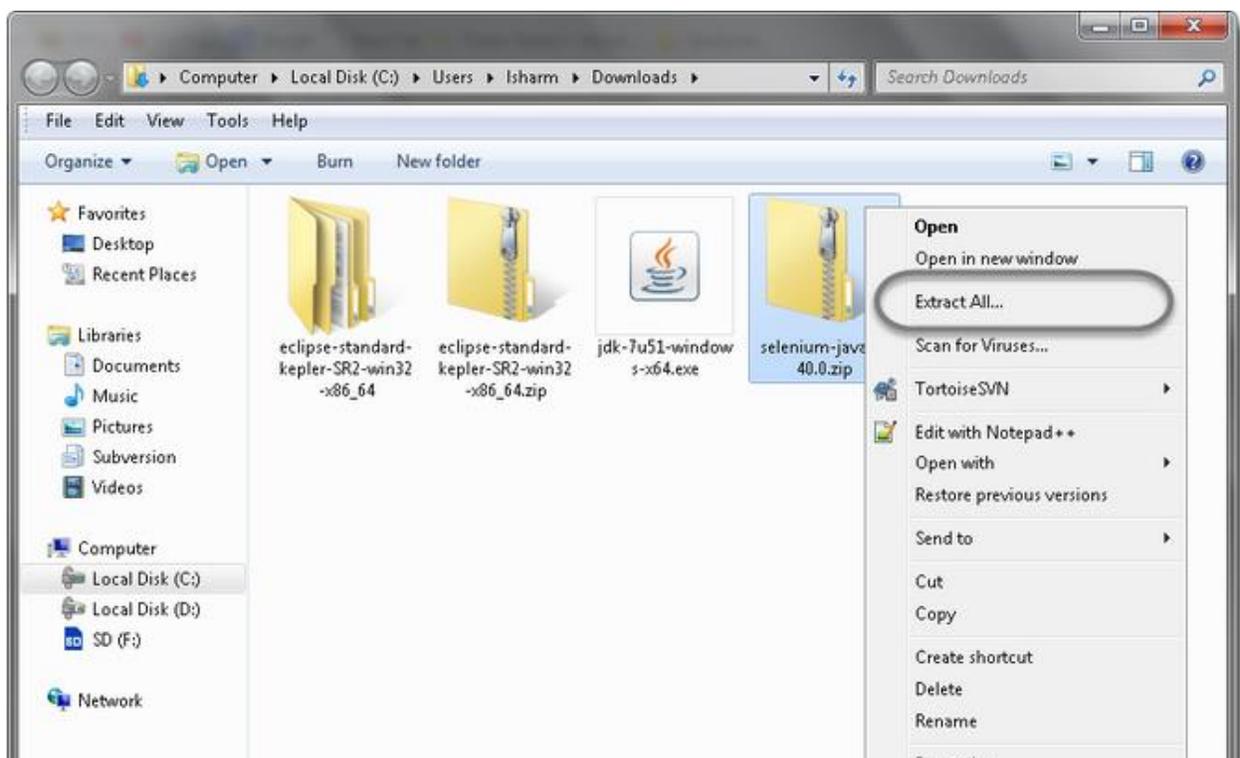
While language bindings for [other languages exist](#), these are the core ones that are supported by the main project hosted on google code.

Language	Client Version	Release Date			
Java	2.40.0	2014-02-19	<a href="#">Download</a>	<a href="#">Change log</a>	<a href="#">Javadoc</a>
C#	2.40.0	2014-02-19	<a href="#">Download</a>	<a href="#">Change log</a>	<a href="#">API docs</a>
Ruby	2.40.0	2014-02-19	<a href="#">Download</a>	<a href="#">Change log</a>	<a href="#">API docs</a>
Python	2.40.0	2014-02-19	<a href="#">Download</a>	<a href="#">Change log</a>	<a href="#">API docs</a>
Javascript (Node)	2.40.0	2014-02-19	<a href="#">Download</a>	<a href="#">Change log</a>	<a href="#">API docs</a>

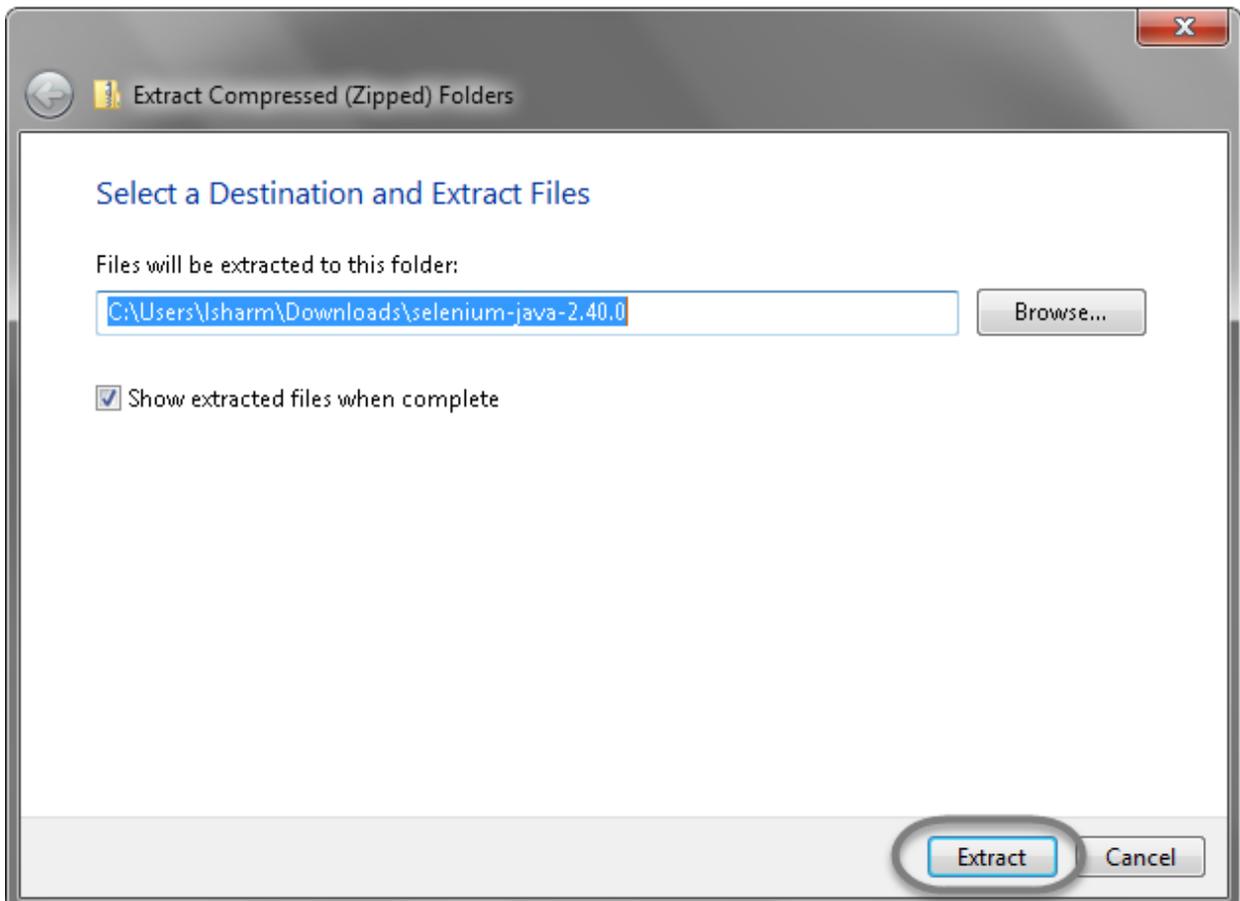
2) Save the zip file to your disk.



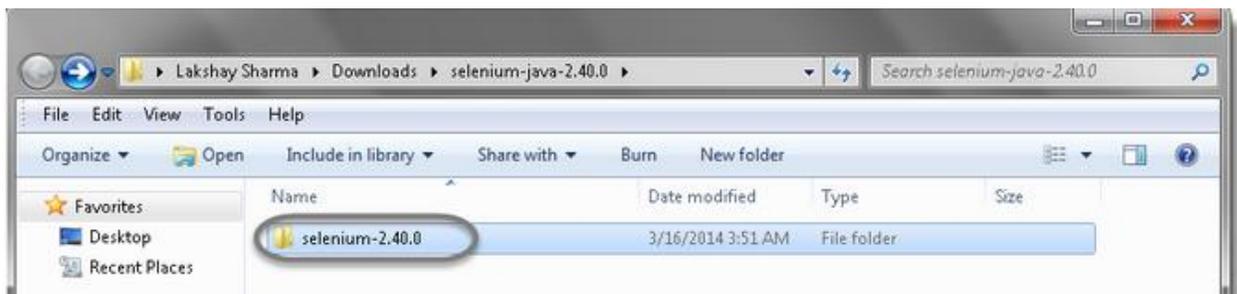
3) Once you have downloaded the archive you will need to **Extract** the zip file, which will create the unzipped Selenium Java folder.



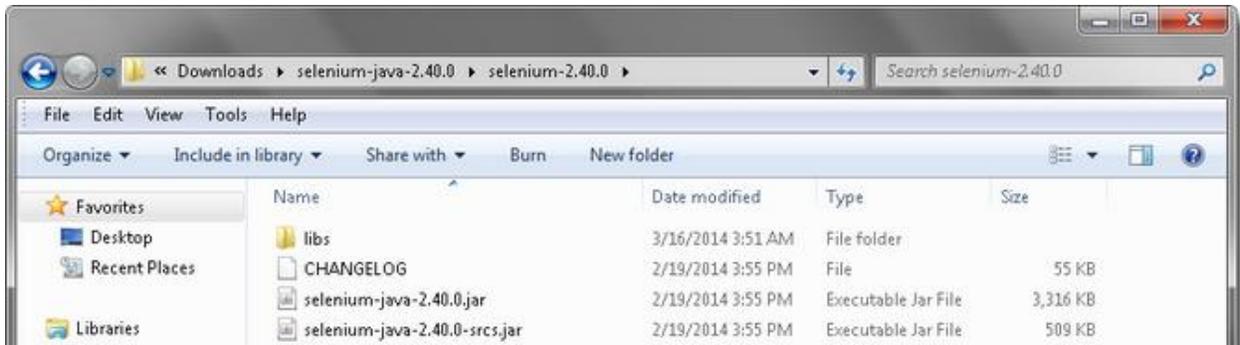
4) You may want to **extract** the archive to the folder where you are maintaining the test ware for your project. I like to extract it at the same location and then move it as per my needs.



5) Once the extraction process is complete, **Open** the Selenium folder.



6) There will be 'libs' folder, 2 jar files and change log in unzipped folder as shown in bellow figure. We will use all these files for configuring webdriver in eclipse.



### ❖ Install Firebug:

- Using Firebug, we can identify the webElements (Eg: textbox, radio button, checkbox, dropdown list..) properties. We are going to see how to identify webElement properties in next chapter.
- Firebug is installed in Firefox Browser.

#### Note:

There are other tools also available to recognize element properties. But Most of them is convenient with Firebug. It is up to you, which tool you want to use.

Other tools:

Internet Explorer: debug bar or press F12

Chrome : press F12

#### Steps for downloading Firebug:

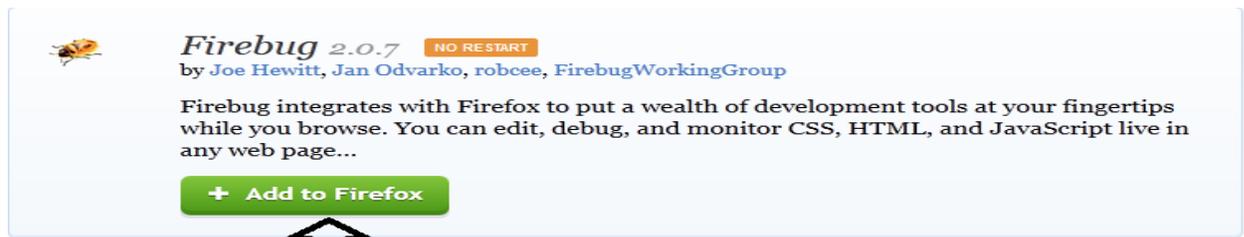
- A) Open Firefox Browser.**
- B) Go to Google search(<https://www.google.co.in>)**
- C) Give the word “Download Firebug” in Google search textbox.**
- D) Click on search button, as shown in the following screenshot.**



- E) Click on below link in the Google results page,as shown in the following screenshot.**

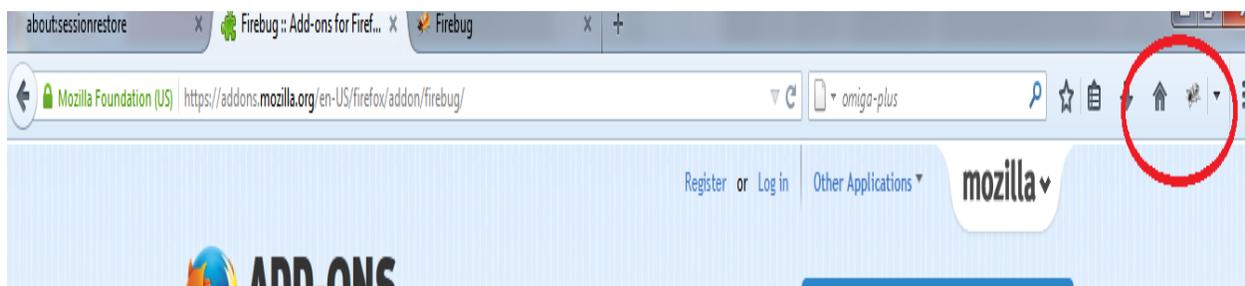
**Firebug :: Add-ons for Firefox - Mozilla Add-ons**  
<https://addons.mozilla.org/en-US/firefox/addon/firebug/>

- F) Click on Add to Firefox**



G) Once it is downloaded and installed in the browser, restart the Firefox Browser.

H) For **confirmation**: Open Firefox Browser, you will see ant symbol in the right side, as shown in the following screenshot.



Press Alt Button in the Keyboard, then you will see Menu Option in the Firefox Browser on the top. Go to Tools → Web Developer → Firebug.

### Installing Firepath:

Using Firepath, we can identify xpath of the webElemnt.we are going to see in the next chapter following topics about xpath:

- ❖ What is XPATH?
- ❖ Different types of xpaths
- ❖ How to use xpath in Script
- ❖ Firepath is installed in Firefox Browser.

#### Steps for downloading Firepath:

- A) Open Firefox Browser.
- B) Go to Google search(<https://www.google.co.in>)
- C) Give the word “Download Firepath” in Google search textbox.
- D) Click on search button, as shown in the following screenshot.



E) Select below link in the Google result set, as shown in the following screenshot

**FirePath :: Add-ons for Firefox - Mozilla Add-ons**  
<https://addons.mozilla.org/en-US/firefox/addon/firepath/>

F) Click on Add to firepath



## FirePath 0.9.7.1

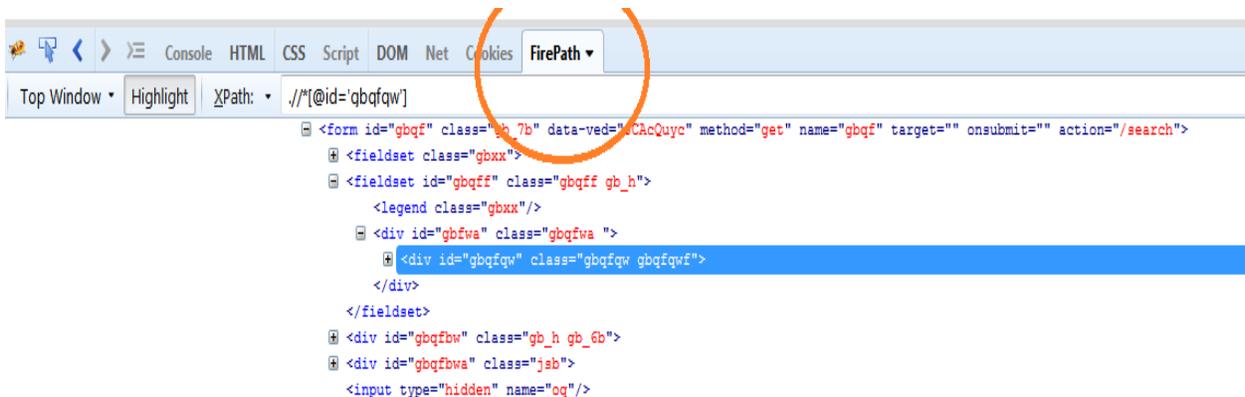
by Pierre Tholence

FirePath is a Firebug extension that adds a development tool to edit, inspect and generate XPath 1.0 expressions, CSS 3 selectors and JQuery selectors (Sizzle selector engine).

+ Add to Firefox

G) Once Downloaded and Installed restart Firefox Browser.

For **confirmation**: Open a Firebug in the Firefox Browser. You will see it in the last.



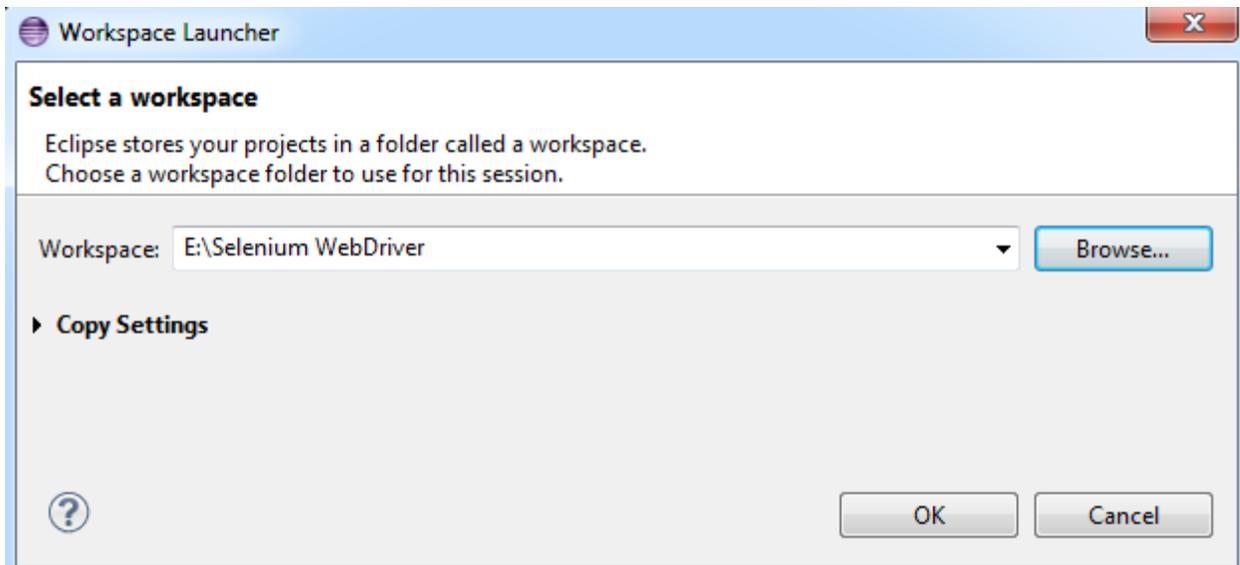
## Set up project a for RC & WebDriver Using Eclipse

Before going to setup project in Eclipse, follow below steps

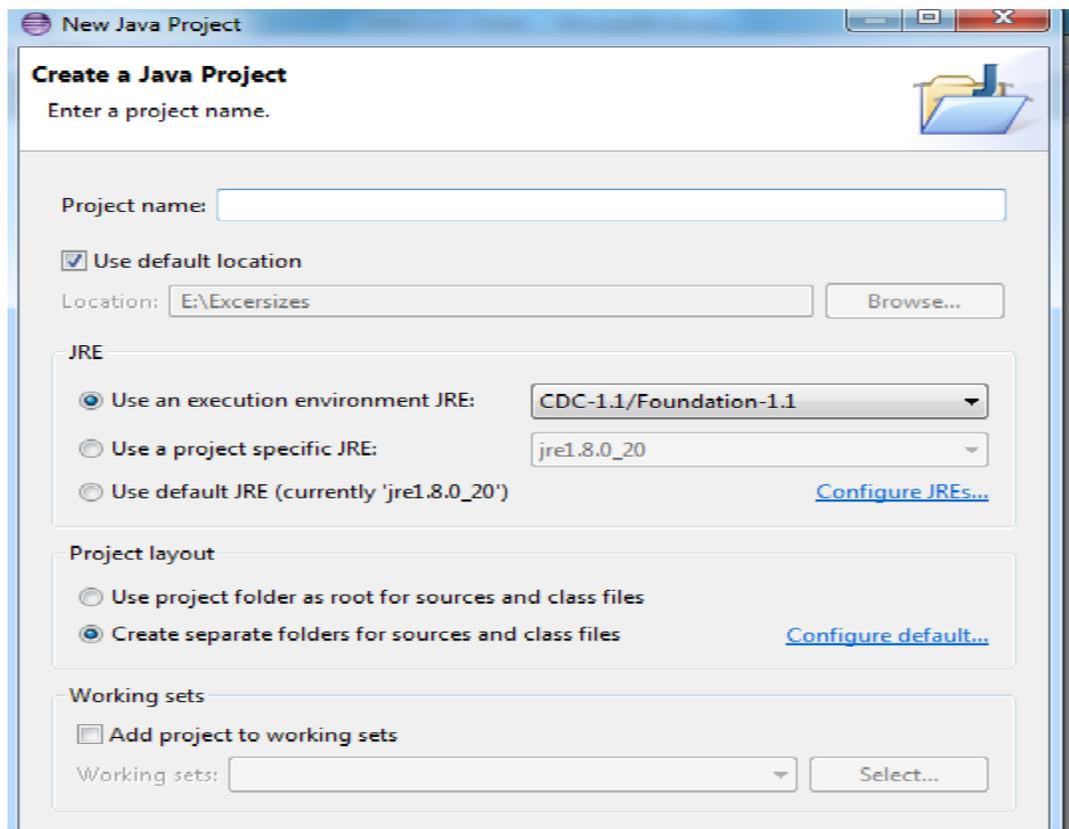
- Create a Folder name : Selenium WebDriver
- Create folder name: libs under Selenium WebDriver.
- Keep Java client.zip files(Java Client Libraries) and selenium-server.jar under libs folder and extract them.

### Project Setup in Eclipse

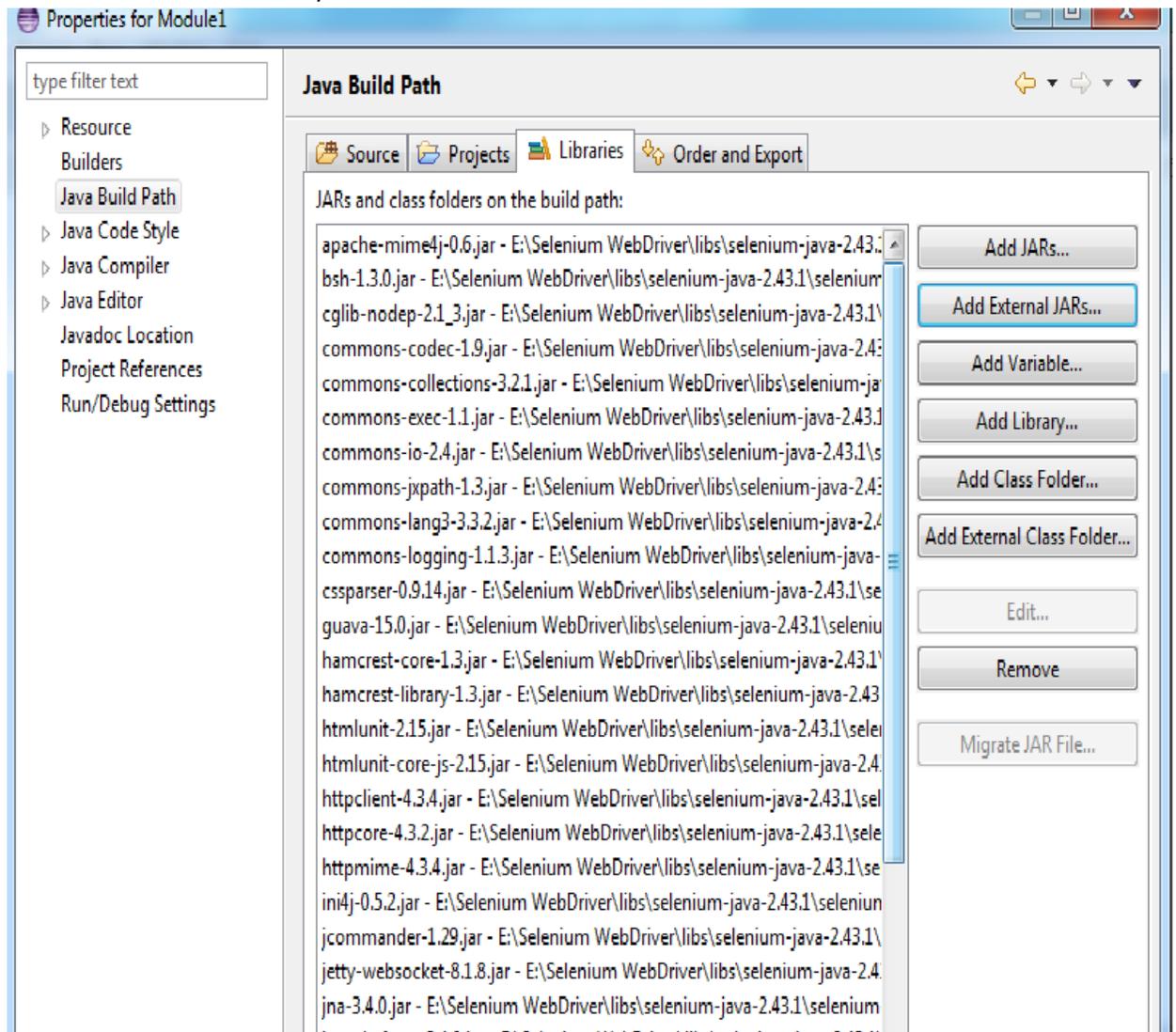
- Click on Eclipse Icon.
- Click on Workbench
- Create a workspace. Workspace means where you are going to keep your project files, as shown in the following screenshot



4. Click Ok.
5. Click on File → New → Project. A new Java project dialog appears, as shown in the following screenshot. Enter the project name of your choice; leave the rest to default, and click Next.



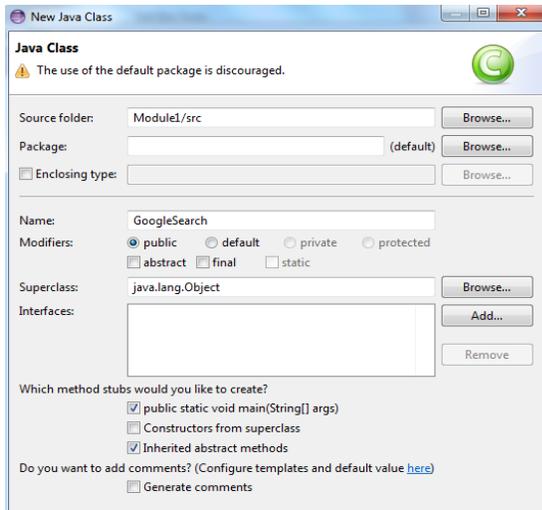
6. Click Finish
7. Right click on project, which you added in the project Explorer → properties→Java Build path(left side)
8. Click on Libraries Tab.
9. Click on Add External Jars. Button and add all the jars available under the libs folder of the Selenium WebDriver directory. Now the Libraries section should look like this:



10. click Ok or Finish.

**Note:** To Create a Java file in Eclipse You need to click on class file.

Navigation: New → Class, as shown in the following screenshot



# Basics on RC(Remote controller)

## Selenium Remote Control (Selenium RC)

### Introduction:

Selenium-RC is the solution for tests that need more than simple browser actions and linear execution. Selenium-RC uses the full power of programming languages to create more complex tests like reading and writing files, querying a database, emailing test results.

You'll want to use Selenium-RC whenever your test requires logic not supported by Selenium-IDE. What logic could this be? For example, Selenium-IDE does not directly support:

- Condition statements
- Iteration
- Logging and reporting of test results
- Error handling, particularly unexpected errors
- Database testing • test case grouping
- Re-execution of failed tests
- Test case dependency
- Screenshot capture of test failures

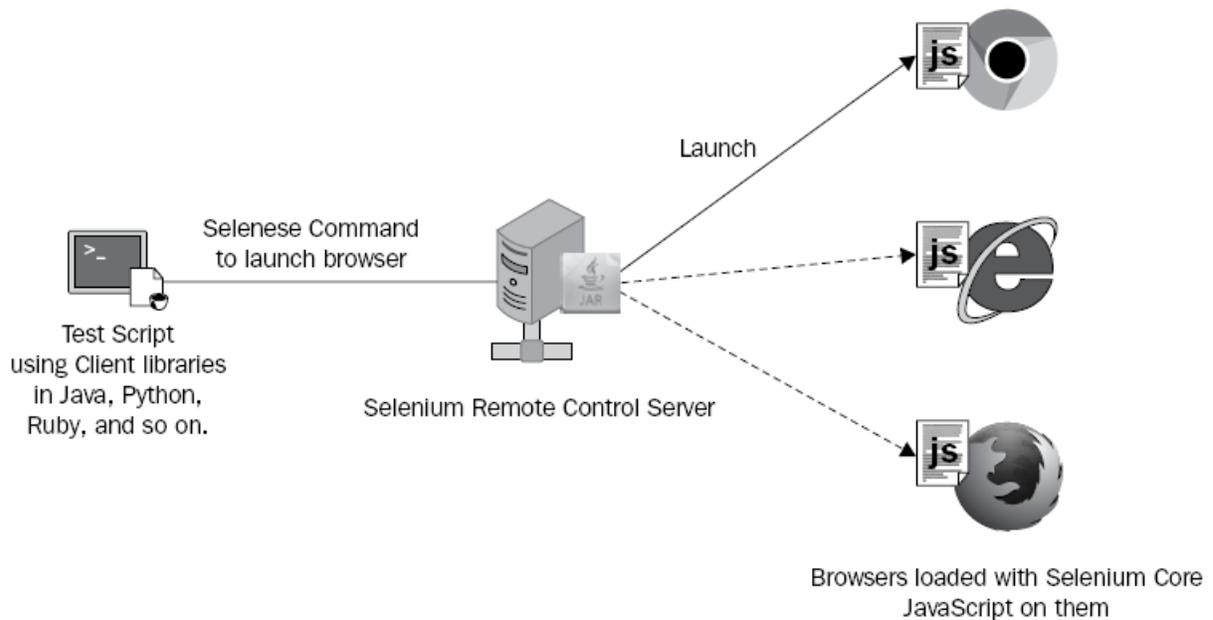
Although these tasks are not supported by Selenium directly, all of them can be achieved by using programming techniques with a language-specific Selenium-RC client library.

### Selenium RC Components:

Selenium-RC components are:

- The Selenium Server which launches and kills browsers, interprets and runs the Selenese commands passed from the test program, and acts as an HTTP proxy, intercepting and verifying HTTP messages passed between the browser and the AUT.
- Client libraries which provide the interface between each programming language and the Selenium-RC Server.

## How selenium –RC Architecture:



- ❖ You first need to launch a separate application called Selenium Remote Control (RC) Server before you can start testing
- ❖ The Selenium RC Server acts as a "middleman" between your Selenium commands and your browser
- ❖ When you begin testing, Selenium RC Server "injects" a JavaScript program called Selenium Core into the browser.
- ❖ Once injected, Selenium Core will start receiving instructions relayed by the RC Server from your test program.
- ❖ When the instructions are received, Selenium Core will execute them as JavaScript commands.
- ❖ The browser will obey the instructions of Selenium Core, and will relay its response to the RC Server.
- ❖ The RC Server will receive the response of the browser and then display the results to you.
- ❖ RC Server will fetch the next instruction from your test script to repeat the whole cycle.

## Sample Program:

```
import com.thoughtworks.selenium.DefaultSelenium;

public class RCBasicProgram {

    public static void main(String[] args) {

        DefaultSelenium selenium = new DefaultSelenium("localhost", 4444,
"*firefox", "http://");
        selenium.start();
        selenium.setSpeed("5000");
        selenium.open("http://google.com");
        selenium.windowMaximize();

        selenium.type("id=lst-ib", "selenium vacancies");
        selenium.click("btnG");
        selenium.close();
        selenium.stop();

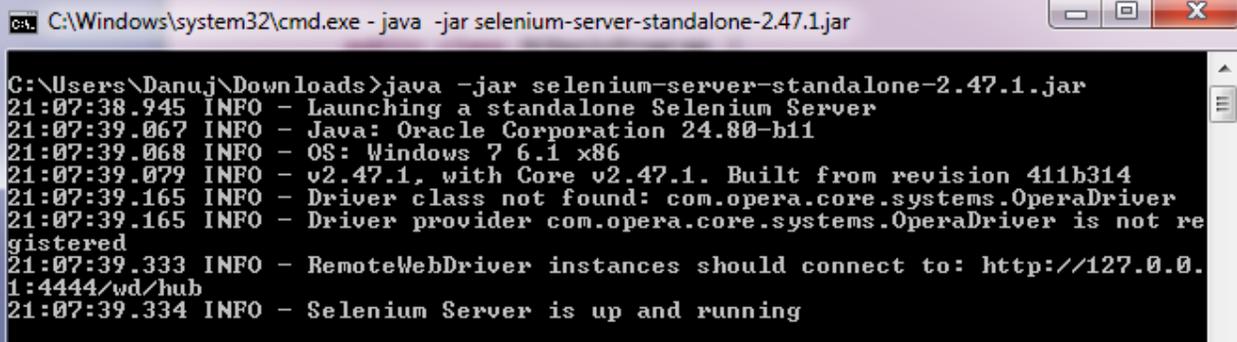
    }

}
```

Before running the program in Selenium-RC, first need to start selenium-server.

- 1) Go to cmd
- 2) Go to path where selenium-server.jar file is located.
- 3) Give the command: java -jar selenium-server.jar

Eg:



```
C:\Windows\system32\cmd.exe - java -jar selenium-server-standalone-2.47.1.jar
C:\Users\Danuj\Downloads>java -jar selenium-server-standalone-2.47.1.jar
21:07:38.945 INFO - Launching a standalone Selenium Server
21:07:39.067 INFO - Java: Oracle Corporation 24.80-b11
21:07:39.068 INFO - OS: Windows 7 6.1 x86
21:07:39.079 INFO - v2.47.1, with Core v2.47.1. Built from revision 411b314
21:07:39.165 INFO - Driver class not found: com.opera.core.systems.OperaDriver
21:07:39.165 INFO - Driver provider com.opera.core.systems.OperaDriver is not re
gistered
21:07:39.333 INFO - RemoteWebDriver instances should connect to: http://127.0.0.
1:4444/wd/hub
21:07:39.334 INFO - Selenium Server is up and running
```

## Program explanation:

### 1. Creating selenium object by using DefaultSelenium class.

```
DefaultSelenium selenium = new DefaultSelenium ("localhost", 4444, "*firefox",  
"http://");
```

DefaultSelenium class is accepting four arguments:

- First Port→In which machine you are working. Eg:localhost
  - Second Port→Port number which selenium is running. By default it is running on 4444.
  - Third Port→Type of Browser
  - Fourth port→protocol
- Selenium.start()→ using this command, starting the session. This is mandatory. Without that, selenium will not start the browser.
  - Selenium.setspeed(5000)→each command is executing for 5 secs
  - Selenium.open(url)→opening the browser.
  - selenium.windowMaximize()→ Maximizing the screen. By default it is in minimize mode.
  - selenium.type("id=lst-ib", "selenium vacancies"): Type command is using for type something in the textbox. Rightnow, giving text is **selenium vacancies**
  - selenium.click("btnG")→click is command to click on button
  - selenium.close()→closing the browser
  - selenium.stop()→closing the session

## output:



# Introduction To WebDriver

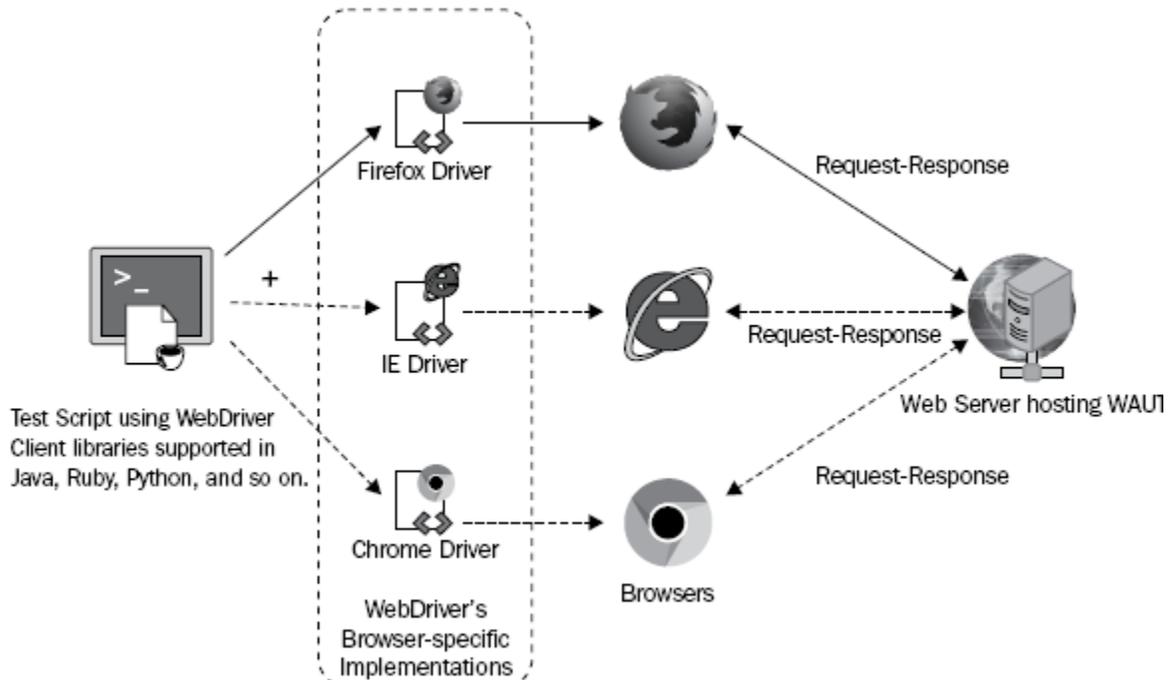
## Introducing WebDriver and WebElements

### Selenium WebDriver or Selenium 2 or WebDriver

WebDriver is a tool for automating testing web applications. It is popularly known as Selenium 2.0. WebDriver interacts directly with the browser without any intermediary, unlike Selenium RC that depends on a server. It is used in the following context:

- ❖ Multi-browser testing including improved functionality for browsers which is not well-supported by Selenium RC (Selenium 1.0).
- ❖ Handling multiple frames, multiple browser windows, popup, and alerts.
- ❖ Complex page navigation.
- ❖ Advanced user navigation such as drag-and-drop.
- ❖ AJAX-based UI elements.
- ❖ WebDriver's architecture is simpler than Selenium RC's.
- ❖ It controls the browser from the OS level

### WebDriver Architecture:



## It works following way:

- ❖ Client libraries are provided in different languages like Java, perl, Python, Ruby,C#,PHP.
- ❖ Using these libraries, user can ask WebDriver to perform certain action on Application(eg: Enter a text in textbox, click on Radio button)
- ❖ By using the client libraries, developers can invoke the browser-specific implementations of WebDriver, such as Firefox Driver, IE Driver, Opera Driver, and so on, to interact with the WAUT (Web based Application Under Test)on the respective browser. These browser-specific implementations of WebDriver will work with the browser natively and execute commands and simulate the actions on the browser.

## Difference between Selenium RC and WebDriver

### Interview Question

Selenium RC(Remote Contol)	Selenium WebDriver
The architecture of Selenium RC is complicated, as the server needs to be up and running before starting a test.	WebDriver's architecture is simpler than Selenium RC, as it controls the browser from the OS level.
Selenium server acts as a middleman between the browser and Selenese commands.	WebDriver interacts directly with the browser and uses the browser's engine to control it.
Selenium RC script execution is slower, since it uses a JavaScript to interact with RC.	WebDriver is faster, as it interacts directly with the browser.
It's a simple and small API.	Complex and a bit large API as compared to RC.
Less object-oriented API.	Comparing with RC, it is object-oriented API.
Cannot test mobile Applications.	Can test iPhone/Android applications.
Selenium Core, just like other JavaScript codes, can access disabled elements.	Selenium Core, just like other JavaScript codes, can access disabled elements.
Selenium RC cannot support the headless HtmlUnit browser. It needs a real, visible browser to operate on.	Web Driver can support the headless HtmlUnit browser.
It can readily support new browser	It cannot readily support new browsers

---

## Limitations of WebDriver:

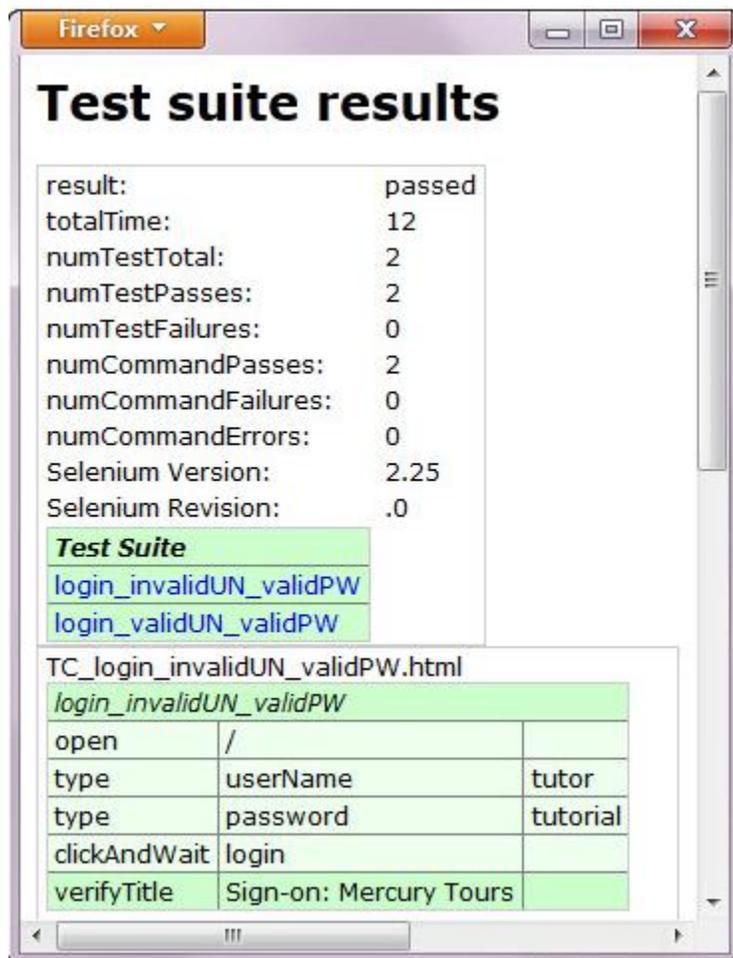
- Webdriver cannot readily support new Browsers

Remember that WebDriver operates on the OS level. Also, remember that different browsers communicate with the OS in different ways. If a new browser comes out, it may have a different process of communicating with the OS as compared to other

browsers. So, **you have to give the WebDriver team quite some time to figure that new process out** before they can implement it on the next WebDriver release.

- Selenium RC has Built-In Test Result Generator

**Selenium RC automatically generates an HTML file of test results.** The format of the report was pre-set by RC itself. Take a look at an example of this report below.



- **WebDriver has no built-in command that automatically generates a Test Results File.** You would have to rely on your IDE's output window, or design the report yourself using the capabilities of your programming language and store it as text, html, etc.

## First Program on WebDriver

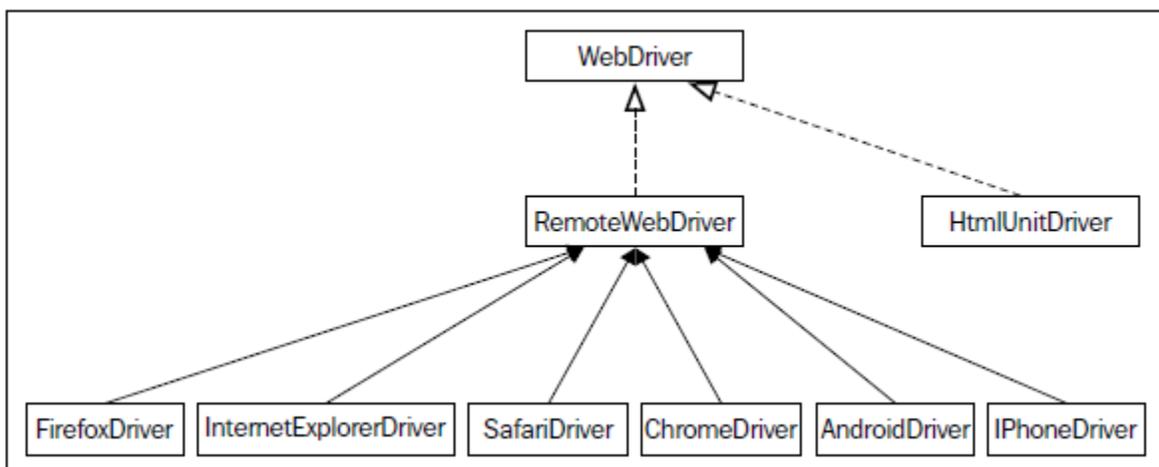
**Scenario:** Launching Google page using firefox browser

```
public class NavigateToAUrl {  
    public static void main(String[] args){  
        WebDriver driver = new FirefoxDriver();  
        driver.get("http://www.google.com");  
    }  
}
```

**Explanation:**

Code: **WebDriver driver = new FirefoxDriver();**

- WebDriver is an interface whose concrete implementation is done in two classes: RemoteWebDriver and HtmlUnitDriver.
- FirefoxDriver is a subclass of the RemoteWebDriver class, which extends the RemoteWebDriver class more specifically for the Firefox browser.
- Similarly, we have the InternetExplorerDriver, ChromeDriver, SafariDriver, AndroidDriver, and iPhoneDriver classes, which are specific implementations for the respective browsers and devices.



**Code:** `driver.get("http://www.google.com")`

In the preceding code, we use one of the methods of the WebDriver interface called the `get()` method to make the browser load the requested web page on it. If the browser, in this case Firefox, is not already opened, it will launch a new browser window.

Now, execute your code by navigating to **Run | Run** or using the *Ctrl + F11* shortcut. A Firefox browser should open and load the Google Search page in your browser.

# WebElements

## Working With WebElements in WebDriver

### Web element:

An Object, which is present or exists in the webpage, is called webElemnt.

Examples: textbox, Checkbox, Combo Box, Dropdown list, Radio button, button, links, tables, etc...

```
<html>
<body>
<form id="loginForm">
<label>Enter Username: </label>
<input type="text" name="Username"/>
<label>Enter Password: </label>
<input type="password" name="Password"/>
<input type="submit"/>
</form>
<a href="forgotPassword.html">Forgot Password ?</a>
</body>
</html>
```

In the preceding HTML code, there are different types of WebElements such as <html>, <body>, <form>, <label>, <input>, and <a>, which together make a web page. Let's analyze the following

### Web Element:

```
<label>Enter Username: </label>
```

Here, <label> is the start tag of the WebElement label. Enter Username: is the text present on the label element. Finally, </label> is the end tag, which indicates the end of WebElement.

### Similarly, take another Web Element:

```
<input type="text" name="Username"/>
```

In the preceding code, type and name are the attributes of the WebElement input with values text and Username, respectively.

## Locating WebElements using WebDriver

Let's start this section by automating the Google Search page, which involves opening the Google Search page, typing the search text in the textbox, and executing the search. The code for that is as follows:

```
public class GoogleSearch {
    public static void main(String[] args){
        WebDriver driver = new FirefoxDriver();
        driver.get("http://www.google.com");
        WebElement searchBox = driver.findElement(By.name("q"));
        searchBox.sendKeys("Packt Publishing");
        searchBox.submit();
    }
}
```

In the preceding code, lines 1 to 4 are same as the example discussed earlier. When you look at line 5, there are three new things that are highlighted as follows:

```
WebElement searchBox = driver.findElement (By.name ("q"));
```

They are the `findElement ()` method, `By.name ()` method, and the `WebElement` interface. The `findElement ()` and `By()` methods instruct `WebDriver` to locate a `WebElement` on a web page, and once found, the `findElement()` method returns the `WebElement` instance of that element. Actions such as click, type, and so on, are performed on a returned `WebElement` using the methods declared in the `WebElement` interface, which will be discussed in detail in the next section.

### The `findElement ()` method

In UI automation, locating an element is the first step before executing any user actions on it. `WebDriver`'s `findElement ()` method is a convenient way to locate an element on the web page

```
WebElement findElement (By by)
```

So, the input parameter for the `findElement ()` method is the `By` instance. The `By` instance is a `WebElement`-locating mechanism. There are eight different ways to locate a `WebElement` on a web page. We will see that when we discuss `By`, shortly.

### The `findElements()` method

If developers think that they may encounter zero or more number of `WebElements` for a given locating mechanism on a web page, they should rather use the `findElements()` method than the `findElement()` method. Because the `findElement()` method throws `NoSuchElementException` in case of zero occurrences of `WebElement` and on the other hand, only the first occurred

WebElement that satisfies the locating mechanism condition though the web page contains multiple WebElements. The method declaration of the findElements() method is as follows:

```
java.util.List<WebElement> findElements(By by)
```

The input parameter is same as the findElement() method, which is an instance of the By class. The difference lies in the return type. Here, if no element is found, an empty list is returned and if there are multiple WebElements present satisfying the locating mechanism, all of them are returned to the caller in a list.

## Using the By locating mechanism

By is the locating mechanism passed to the findElement() method or the findElements() method to fetch the respective WebElement(s) on a web page. There are eight different locating mechanisms; that is, eight different ways to identify an HTML element on a web page. They are located by Name, ID, TagName, Class, LinkText, PartialLinkText, XPath, and CSS.

### The By.name() method

As seen earlier, every element on a web page has many attributes. Name is one among them. For instance, the HTML code for the **Google Search** button will be:

```
<button id="gbqfba" aria-label="Google Search" name="btnK" class="gbqfba"><span id="gbqfsa">Google Search</span></button>
```

```
public class GoogleSearchButtonByName {
    public static void main(String[] args){
        WebDriver driver = new FirefoxDriver();
        driver.get("http://www.google.com");
        WebElement searchBox = driver.findElement(By.name("btnK"));
        searchBox.submit();
    }
}
```

### The By.id() method

On a web page, each element is uniquely identified by an ID, if provided. An ID can be assigned manually by the developer of the web application or, most of the times, left to be dynamically generated by the server where the web application is hosted, and this ID can change over a period of time.

Now, if we consider the same HTML code of the Google Search button:

```
<button id="gbqfba" aria-label="Google Search" name="btnK" class="gbqfba"><span id="gbqfsa">Google Search</span></button>
```

In the preceding code, the id value of this button is gbqfba. This might change by the time you read this book, because this could be a server-generated ID.

Let us see what changes need to be made to our test script to use id instead of name:

```
public class GoogleSearchButtonById {
    public static void main(String[] args){
        WebDriver driver = new FirefoxDriver();
        driver.get("http://www.google.com");
        WebElement searchBox = driver.findElement(By.id("gbqfba"));
        searchBox.submit();
    }
}
```

### **The By.tagName() method:**

Let's see how the code looks like when a search for the number of buttons present on a Google Search page is made.

```
public class GoogleSearchPageByTagName{
    public static void main(String[] args){
        WebDriver driver = new FirefoxDriver();
        driver.get("http://www.google.com");
        List<WebElement> buttons = driver.findElements (By. tagName ("button"));
        System.out.println(buttons.size());
    }
}
```

In the preceding code, we have used the By.tagName locating mechanism and findElements() method, which returns a list of all the buttons available on the page. On line 6, when we printed the size of the list, it returns 3.

If you are wondering how there are three buttons on the Google Search page while only two are visible, the following are all the buttons available on the search page:

```
<button id=gbqfb aria-label="Google Search" class=gbqfb name=btnG><span class=gbqfi></span></button>
```

```
<button id=gbqfba aria-label="Google Search" name=btnK class=gbqfba><span id=gbqfsa>Google Search</span></button>
```

```
<button id=gbqfbb aria-label="I'm Feeling Lucky" name=btnI class=gbqfba onclick="if(this.form.q.value)this.checked=1;else window.top.location='/doodles/'"><span id=gbqfsb>I'm Feeling Lucky</span></button>
```

## The By.className() method

Every HTML element on a web page, generally, is styled by the web page developer or designer. It is not mandatory that each element should be styled, but it is generally followed to make it appealing to the end user.

So, in order to apply styles to an element, they can be declared directly in the element tag or placed in a separate file called the CSS file and can be referenced in the element using the className() method. For instance, a style attribute for a button can be declared in a CSS file as follows:

```
.buttonStyle{  
width: 50px;  
height: 50px;  
border-radius: 50%;  
margin: 0% 2%;  
}
```

Now, this style can be applied on the button element in a web page as follows:

```
<button name="sampleBtnName" id="sampleBtnId" class="buttonStyle">I'm Button</button>
```

So, buttonStyle is used as value for the class attribute of the button element, and it inherits all the styles declared in the CSS file. Now, let's try this on our Google search page. We will try to make WebDriver identify the search box using its class name and type some text into it. First, in order to get the class name of the search box, as we know, we will use Firebug and fetch it. After getting it, change the location mechanism to By.className and specify the class attribute value in it. The code for that is as follows:

```
Public class GoogleSearchByClassName{  
    Public static void main(String[] args) {  
        WebDriver driver = new FirefoxDriver();  
        driver.get("http://www.google.com");  
        WebElement searchBox = driver.findElement(By.className("gbqfif"));  
        searchBox.sendKeys ("Packt Publishing");  
    } }  
}
```

## The `By.linkText()` method:

As the name suggests, the `By.linkText` locating mechanism can only be used to identify the HTML links. The HTML link elements are represented on a web page using the `<a>` tag, abbreviation for the anchor tag. A typical anchor tag looks like this:

```
<a href="/intl/en/about.html">About Google</a>
```

Here, `href` is the link to a different page where your web browser will take you when clicked on the link. So, the preceding HTML code when rendered by the browser looks like this:

This About Google is the link text. So the locating mechanism `By.linkText` uses this text on an anchor tag to identify the `WebElement`. The code for this would look like this:

```
public class GoogleSearchByLinkText{
    public static void main(String[] args){
        WebDriver driver = new FirefoxDriver();
        driver.get("http://www.google.com");
        WebElement aboutLink = driver.findElement(By.linkText("About Google"));
        aboutLink.click();
    }
}
```

Here, the `By.linkText` locating mechanism is used to identify the About Google link.

## The `By.partialLinkText()` method

The `By.partialLinkText` locating mechanism is an extension to the previous one. If you are not sure of the entire link text or want to use only part of the link text, you can use this locating mechanism to identify the link element. So let's modify the previous example to use only partial text on the link, that is, **About**.

```
Public class GoogleSearchByPartialLinkText {
    Public static void main (String [] args) {
        WebDriver driver = new FirefoxDriver();
        driver.get("http://www.google.com");
        WebElement aboutLink = driver.findElement(By. partialLinkText("About"));
        aboutLink.click();
    }
}
```

## The `By.xpath()` method

WebDriver uses **XPath** to identify a `WebElement` on the web page. Before we see how it does that, we will quickly look at the syntax for XPath. XPath is a short name for the XML path. The HTML for our web page is also one form of the XML document. So in order to identify an element on an HTML page, we need to use a specific XPath syntax as follows:

The root element is identified as //

To identify all the div elements, the syntax will be //div

To identify the link tags that are within the div element, the syntax will be //div/a

To identify all the elements with a tag, we use \*. The syntax will be //div/\*

To identify all the div elements that are at three levels down from the root, we can use //\*/\*/div

To identify specific elements, we use attribute values of those elements, such as

//\*/div/a[@id='attrValue'], which will return the anchor element. This element is at third level from root within a div element, and has an id value attrValue

So, we need to pass these kinds of XPath syntaxes to our WebDriver to make it identify our target element. But going through the HTML page figuring out the XPath for each element will be extremely difficult. For this, if you remember, we have installed a Firebug extension named FirePath. This will quickly give you the XPath of the target element that you can use in the WebDriver code. Following is the screenshot of the XPath of the **Google Search** button: If you see the preceding image, the **Google Search** Button is selected and in the **FirePath** tab below the XPath, the value is displayed as `//*[@id='gbqfba']`.

Now, let us see the code example and how WebDriver uses this XPath to identify the element.

```
public class GoogleSearchByXPath{
    public static void main(String[] args){
        WebDriver driver = new FirefoxDriver();
        driver.get("http://www.google.com");
        WebElement searchButton = driver.findElement(By.xpath("//*[@ id='gbqfba']"));
        System.out.println(searchButton.getText());
    }
}
```

In the preceding code, we are using the `By.xpath` locating mechanism and passing the XPath of the `WebElement` to it.

## The `By.cssSelector()` method

The `By.cssSelector()` method is similar to the `By.xpath()` method in its usage but the difference is that it is slightly faster than the `By.xpath` locating mechanism. Following are the commonly used syntaxes to identify elements:

To identify an element using the div element with id #flrs, we use the #flrs syntax

To identify the child anchor element, we use the #flrs > a syntax, which will return the link element

To identify the anchor element with its attribute, we use the #flrs > a[a[href="/intl/en/about.html"]] syntax

Let's try to modify the previous code, which uses the XPath-locating mechanism to use the `cssSelector` mechanism.

```

Public class GoogleSearchByCSSSelector{
    public static void main(String[] args) {
        WebDriver driver = new FirefoxDriver();
        driver.get("http://www.google.com");
        WebElement searchButton = driver.findElement(By.cssSelector("#gbqfba"));
        System.out.println(searchButton.getText());
    }
}

```

The preceding code uses the `By.cssSelector` locating mechanism that uses the css selector ID of the **Google Search** button.

Let's look at a slightly complex example. We will try to identify the **About Google** link on the Google Search page:

```

public class GoogleSearchByCSSSelector{
    public static void main(String[] args) {
        WebDriver driver = new FirefoxDriver();
        driver.get("http://www.google.com");
        WebElement searchButton = driver.findElement(By.cssSelector("#flrs>a[href='/intl/en/about.html']"));
        System.out.println(searchButton.getText());
    }
}

```

## The `getText ()` method

The `getText` action can be taken on all the `WebElements`. It will give the visible text if the element contains any text on it or else will return nothing.

The API syntax for the `getText ()` method is as follows:

```
Java.lang.String getText ()
```

There is no input parameter for the preceding method, but it returns the visible `innerText` string of the `WebElement` if anything is available, else will return an empty string

```

Public class GetText {

    Public static void main (String [] args) {
        WebDriver driver = new FirefoxDriver ();
        driver.get("http://www.google.com");
        WebElement searchButton = driver.findElement
        (By.id("gb_70"));
        System.out.println (searchButton. GetText());
    }
}

```

The `isDisplayed` action verifies if an element is displayed on the web page and can be executed on all the `WebElements`.

The API syntax for the `isDisplayed ()` method is as follows:

```
boolean isDisplayed ()
```

The preceding method returns a Boolean value specifying whether the target element is displayed or not displayed on the web page.

```
Public class IsDisplayed {  
  
    Public static void main (String [] args) {  
        WebDriver driver = new FirefoxDriver ();  
        driver.get ("http://www.google.com");  
        WebElement searchButton = driver.findElement (  
            By.name ("btnK"));  
    }  
}
```

## The `isEnabled()` method

The `isEnabled` action verifies if an element is enabled on the web page and can be executed on all the `WebElements`.

The API syntax for the `isEnabled()` method is as follows:

```
boolean isEnabled()
```

The preceding method returns a Boolean value specifying whether the target element is enabled or not enabled on the web page

```
Public class IsEnabled {  
  
    Public static void main (String [] args) {  
        WebDriver driver = new FirefoxDriver ();  
        driver.get ("http://www.google.com");  
        WebElement searchButton = driver.findElement  
(By.name ("btnK"));  
        System.out.println (searchButton. isEnabled ());  
    }  
}
```

## The `isSelected ()` method

The `isSelected` action verifies if an element is selected right now on the web page and can be executed only on a radio button, options in select, and checkbox `WebElements`. When executed on other elements, it will return `false`.

The API syntax for the `isSelected ()` method is as follows:

```
boolean isSelected()
```

The preceding method returns a `Boolean` value specifying whether the target element is selected or not selected on the web page.

```
public class IsSelected {  
  
    Public static void main (String[] args) {  
        WebDriver driver = new FirefoxDriver ();  
        driver.get ("http://www.google.com");  
        WebElement searchBox = driver.findElement (By.name  
("q"));  
        System.out.println (searchBox.isSelected());  
    }  
}
```

### Interview Question

**Q: How will you interact with webpage, while writing scripts in webDriver?**

1. You need to locate a web element first on the webpage before interacting with it.
2. Locating element can be done using by creating Webdriver instance.
3. Webdriver gives us **findElement**, **findElements** methods.

**Q: what is difference between findElement and findElements Methods?**

**FindElement ():**

- a) 0 matches : throws exception(NoSuchElementException)
- b) 1 match : returns web Element instance
- c) 2 matches: returns only first appear in webpage.

**FindElements ():**

- a) 0 matches: returns empty list.
- b) 1 match: returns list of 1 WebElement instance.
- c) 2 + matches: returns list with all matching web element instances.

# Basic Scenarios on WebDriver

## Accessing Links & Tables using Selenium WebDriver

### Accessing Links

#### Links Matching a Criterion

Links can be accessed using an exact or partial match of their link text. The examples below provide scenarios where multiple matches would exist, and would explain how WebDriver would deal with them.

#### Exact Match

Accessing links using their exact link text is done through the `By.linkText()` method.

However, if there are two links that have the very same link text, this method will only access the first one. Consider the HTML code below

```
<html>
  <head>
    <title>Sample</title>
  </head>
  <body>
    <a href="http://www.google.com">click here</a>
    <br>
    <a href="http://www.fb.com">click here</a>
  </body>
</html>
```



When you try to run the WebDriver code below, you will be accessing the first "click here" link

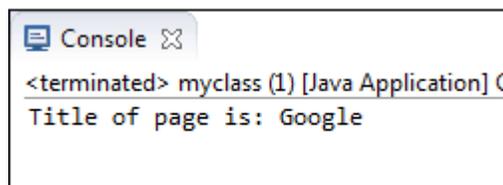
```

public static void main(String[] args) {
    String baseUrl = "file:///D:/newhtml.html";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    driver.findElement(By.linkText("click here")).click();
    System.out.println("Title of page is: " + driver.getTitle());
    driver.quit();
}

```

As a result, you will automatically be taken to Google.



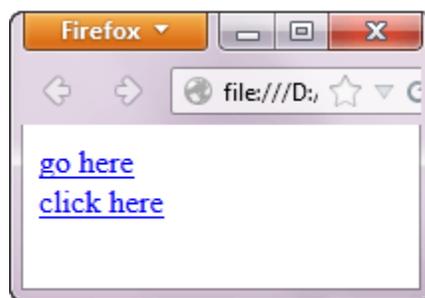
## Partial Match

Accessing links using a portion of their link text is done using the **By.partialLinkText()** method. If you specify a partial link text that has multiple matches, only the first match will be accessed. Consider the HTML code below.

```

<html>
  <head>
    <title>Partial Match</title>
  </head>
  <body>
    <a href="http://www.google.com">go here</a>
    <br>
    <a href="http://www.fb.com">click here</a>
  </body>
</html>

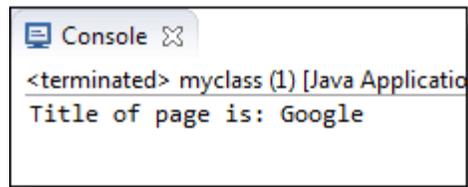
```



When you execute the WebDriver code below, you will still be taken to Google.

```
public static void main(String[] args) {
    String baseUrl = "file:///D:/partial_match.html";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    driver.findElement(By.partialLinkText("here")).click();
    System.out.println("Title of page is: " + driver.getTitle());
    driver.quit();
}
```



Console

```
<terminated> myclass (1) [Java Applicatio
Title of page is: Google
```

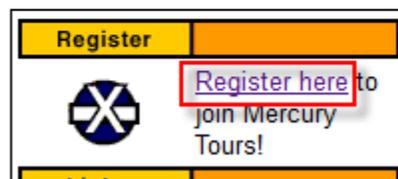
## Case-sensitivity

The parameters for **By.linkText()** and **By.partialLinkText()** are both case-sensitive, meaning that capitalization matters. For example, in Mercury Tours' homepage, there are two links that contain the text "egis" - one is the "REGISTER" link found at the top menu, and the other is the "Register here" link found at the lower right portion of the page.

*The link at the top menu*



*The link at the lower right portion of the page*



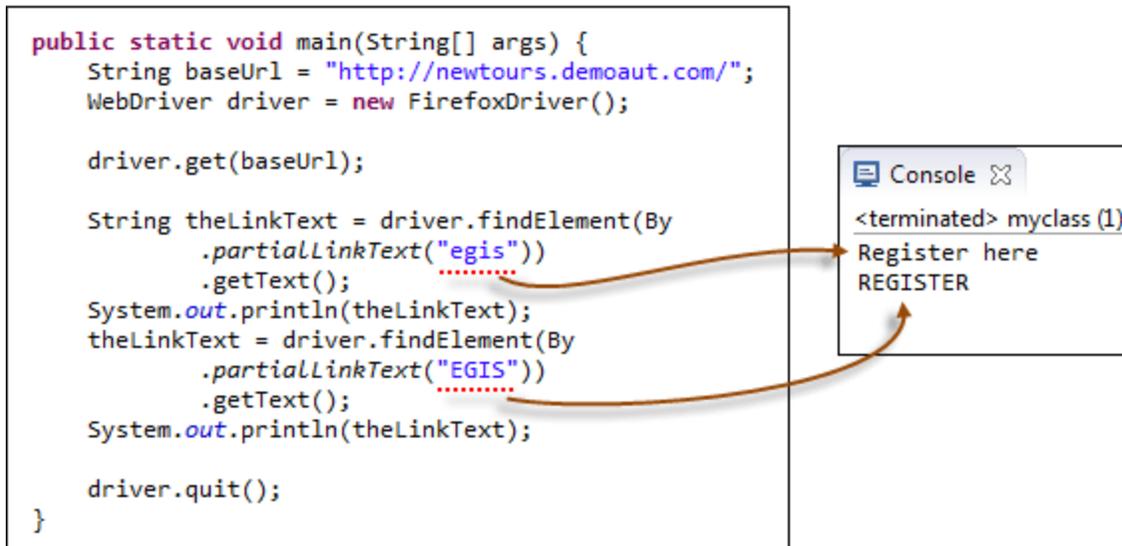
Though both links contain the character sequence "egis", the "By.partialLinkText()" method will access these two links separately depending on the capitalization of the characters. See the sample code below.

```
public static void main(String[] args) {
    String baseUrl = "http://newtours.demoaut.com/";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);

    String theLinkText = driver.findElement(By
        .partialLinkText("egis"))
        .getText();
    System.out.println(theLinkText);
    theLinkText = driver.findElement(By
        .partialLinkText("EGIS"))
        .getText();
    System.out.println(theLinkText);

    driver.quit();
}
```



```
<terminated> myclass (1)
Register here
REGISTER
```

## All Links

One of the common procedures in web [testing](#) is to test if all the links present within the page are working. This can be conveniently done using a combination of the **Java for-each loop** and the **By.tagName("a")** method. The WebDriver code below checks each link from the Mercury Tours homepage to determine those that are working and those that are still under construction.

```
package practice_webdriver;

import java.util.List;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

public class AllLinks {

    public static void main(String[] args) {
        String baseUrl = "http://newtours.demoaut.com/";
        WebDriver driver = new FirefoxDriver();
        String underConsTitle = "Under Construction: Mercury Tours";
        driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

        driver.get(baseUrl);
        List<WebElement> linkElements = driver.findElements(By.tagName("a"));
        String[] linkTexts = new String[linkElements.size()];
```

```

int i = 0;

//extract the link texts of each link element
for (WebElement e : linkElements) {
    linkTexts[i] = e.getText();
    i++;
}

//test each link
for (String t : linkTexts) {
    driver.findElement(By.linkText(t)).click();
    if (driver.getTitle().equals(underConsTitle)) {
        System.out.println("\"" + t + "\""
            + " is under construction.");
    } else {
        System.out.println("\"" + t + "\""
            + " is working.");
    }
    driver.navigate().back();
}
driver.quit();
}
}

```

The output should be similar to the one indicated below.

```

<terminated> myclass (1) [Java Application] C:\Program Files\Java\jre7\bin\j
"Home" is working.
"Flights" is working.
"Hotels" is under construction.
"Car Rentals" is under construction.
"Cruises" is working.
"Destinations" is under construction.
"Vacations" is under construction.
"SIGN-ON" is working.
"REGISTER" is working.
"SUPPORT" is under construction.
"CONTACT" is under construction.
"your destination" is under construction.
"featured vacation destinations" is under construction.
"Register here" is working.
"Business Travel @ About.com" is working.
"Salon Travel" is working.

```

## Links Outside and Inside a Block

The latest HTML5 standard allows the `<a>` tags to be placed inside and outside of block-level tags like `<div>`, `<p>`, or `<h1>`. The `"By.linkText()"` and `"By.partialLinkText()"` methods can access a link located outside and inside these block-level elements. Consider the HTML code below.

```
<body>
  <p>
    <a href="http://www.google.com">Inside a block-level tag.</a>
  </p>

  <br>
  <a href="http://www.fb.com">
    <div>
      <span>Outside a block-level tag.</span>
    </div>
  </a>
</body>
```



The WebDriver code below accesses both of these links using `By.partialLinkText()` method.

```
public static void main(String[] args) {
    String baseUrl = "file:///D:/Links%20Outside%20and%20Inside%20a%20Block.html";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    driver.findElement(By.partialLinkText("Inside")).click();
    System.out.println(driver.getTitle());
    driver.navigate().back();
    driver.findElement(By.partialLinkText("Outside")).click();
    System.out.println(driver.getTitle());
    driver.quit();
}
```

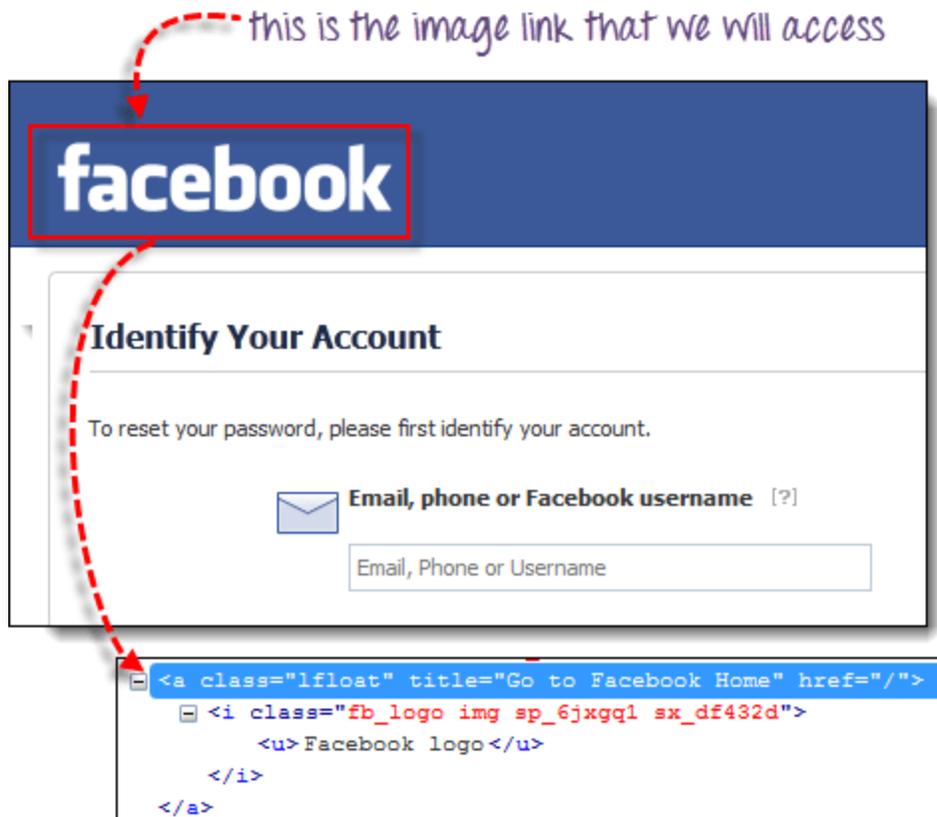
```
Console
<terminated> myclass (1) [Java Application] C:\Program Files\Java\jre7
Google
Welcome to Facebook - Log In, Sign Up or Learn More
```

The output above confirms that both links were accessed successfully because their respective page titles were retrieved correctly.

## Accessing Image Links

Image links are images that act as references to other sites or sections within the same page. Since they are images, we cannot use the `By.linkText()` and `By.partialLinkText()` methods because image links basically have no link texts at all. In this case, we should resort to using either `By.cssSelector` or `By.xpath`. The first method is more preferred because of its simplicity.

In the example below, we will access the "Facebook" logo on the upper left portion of Facebook's Password Recovery page.

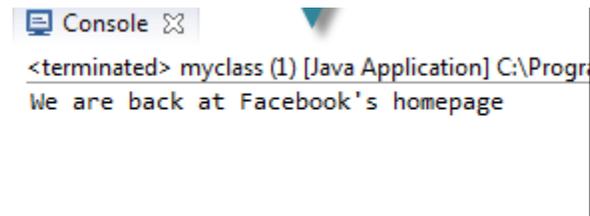


We will use `By.cssSelector` and the element's "title" attribute to access the image link. And then we will verify if we are taken to Facebook's homepage.

```
package practice_webdriver;  
import org.openqa.selenium.*;  
import org.openqa.selenium.firefox.FirefoxDriver;  
  
public class ImageLink {  
  
    public static void main(String[] args) {  
        String baseUrl =  
"https://www.facebook.com/login/identify?ctx=recover";  
        WebDriver driver = new FirefoxDriver();  
  
        driver.get(baseUrl);  
        //click on the "Facebook" logo on the upper left portion  
        driver.findElement(By.cssSelector("a[title=\"Go to Facebook  
Home\"]")).click();  
  
        //verify that we are now back on Facebook's homepage
```

```
        if (driver.getTitle().equals("Welcome to Facebook - Log In, Sign Up  
or Learn More")) {  
            System.out.println("We are back at Facebook's homepage");  
        } else {  
            System.out.println("We are NOT in Facebook's homepage");  
        }  
        driver.quit();  
    }  
}
```

## Result



## Reading a Table

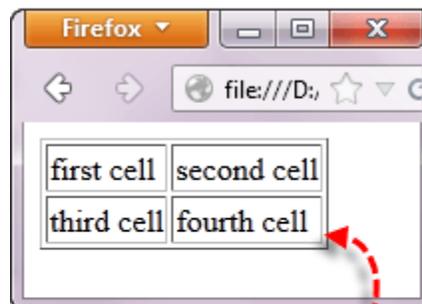
There are times when we need to access elements (usually texts) that are within HTML tables. However, it is very seldom for a web designer to provide an id or name attribute to a certain cell in the table. Therefore, we cannot use the usual methods such as "By.id()", "By.name()", or "By.cssSelector()". In this case, the most reliable option is to access them using the "By.xpath()" method.

## XPath Syntax

Consider the HTML code below.

```
<html>
  <head>
    <title>Sample</title>
  </head>
  <body>
    <table border="1">
      <tbody>
        <tr>
          <td>first cell</td>
          <td>second cell</td>
        </tr>
        <tr>
          <td>third cell</td>
          <td>fourth cell</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

We will use XPath to get the inner text of the cell containing the text "fourth cell".



*we will try to  
access this cell*

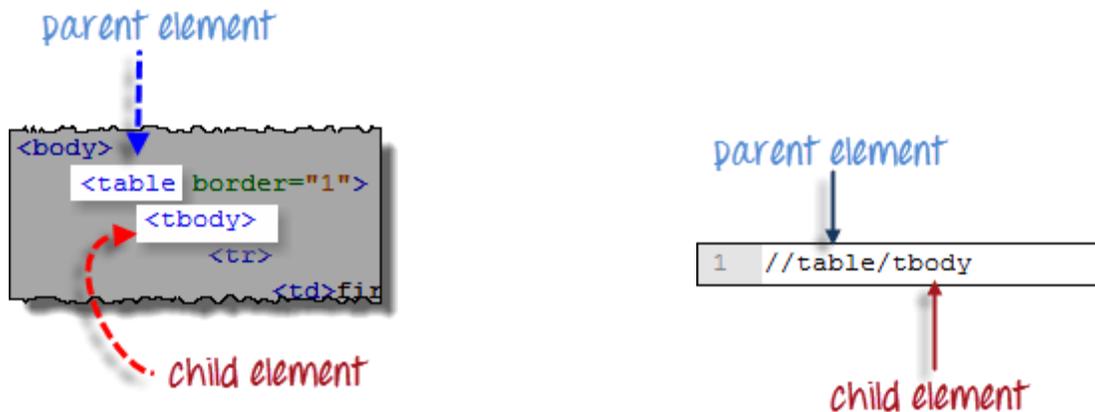
### Step 1 - Set the Parent Element (table)

XPath locators in WebDriver always start with a double forward slash "/" and then followed by the parent element. Since we are dealing with tables, the parent element should always be the <table> tag. The first portion of our XPath locator should therefore start with "//table".

```
//table
```

## Step 2 - Add the child elements

The element immediately under `<table>` is `<tbody>` so we can say that `<tbody>` is the "child" of `<table>`. And also, `<table>` is the "parent" of `<tbody>`. All child elements in XPath are placed to the right of their parent element, separated with one forward slash "/" like the code shown below.



## Step 3 - Add Predicates

The `<tbody>` element contains two `<tr>` tags. We can now say that these two `<tr>` tags are "children" of `<tbody>`. Consequently, we can say that `<tbody>` is the parent of both the `<tr>` elements.

Another thing we can conclude is that the two `<tr>` elements are siblings. Siblings refer to child elements having the same parent.

To get to the `<td>` we wish to access (the one with the text "fourth cell"), we must first access the second `<tr>` and not the first. If we simply write `//table/tbody/tr`, then we will be accessing the first `<tr>` tag.

So, how do we access the second `<tr>` then? The answer to this is to use Predicates.

Predicates are numbers or HTML attributes enclosed in a pair of square brackets "[ ]" that distinguish a child element from its siblings. Since the `<tr>` we need to access is the second one, we shall use `[2]` as the predicate.

```
1 //table/tbody/tr[2]
```

The [2] predicate denotes that we are accessing the 2nd <tr> of the parent <tbody>

If we won't use any predicate, XPath will access the first sibling. Therefore, we can access the first <tr> using either of these XPath codes.

```
//table/tbody/tr
```

this will automatically access the first <tr> because no predicate was used

```
//table/tbody/tr[1]
```

this will access the first <tr> because the predicate [1] explicitly says it

#### Step 4 - Add the Succeeding Child Elements Using the Appropriate Predicates

The next element we need to access is the second <td>. Applying the principles we have learned from steps 2 and 3, we will finalize our XPath code to be like the one shown below.

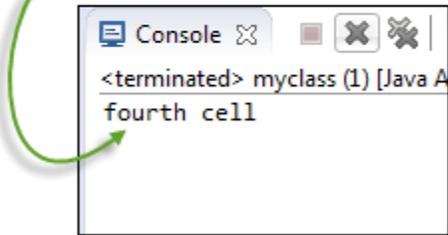
Now that we have the correct XPath locator, we can already access the cell that we wanted to and obtain its inner text using the code below. It assumes that you have saved the HTML code above as "newhtml.html" within your C Drive.

```
public static void main(String[] args) {
    String baseUrl = "file:///C:/newhtml.html";
    WebDriver driver = new FirefoxDriver();

    driver.get(baseUrl);
    String innerText = driver.findElement(
        By.xpath("//table/tbody/tr[2]/td[2]")).getText();
    System.out.println(innerText);
    driver.quit();
}
```

```
//table/tbody/tr[2]/td[2]
```

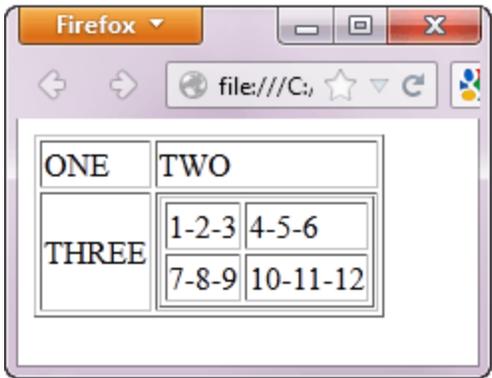
the inner text was  
successfully retrieved  
using xpath



### Accessing Nested Tables

The same principle discussed above applies to nested tables. Nested tables are tables located within another table. An example is shown below.

```
<html>
  <head>
    <title>Sample</title>
  </head>
  <body>
    <!--outer table-->
    <table border="1">
      <tbody>
        <tr>
          <td>ONE</td>
          <td>TWO</td>
        </tr>
        <tr>
          <td>THREE</td>
          <td>
            <!--inner table-->
            <table border="1">
              <tbody>
                <tr>
                  <td>1-2-3</td>
                  <td>4-5-6</td>
                </tr>
                <tr>
                  <td>7-8-9</td>
                  <td>10-11-12</td>
                </tr>
              </tbody>
            </table>
          </td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```



To access the cell with the text "4-5-6" using the "//parent/child" and predicate concepts from the previous section, we should be able to come up with the XPath code below.

The outer table

The inner table

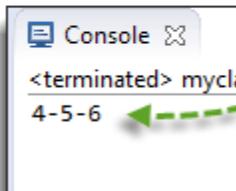
```
//table/tbody/tr[2]/td[2]/table/tbody/tr/td[2]
```

The WebDriver code below should be able to retrieve the inner text of the cell which we are accessing.

```
public static void main(String[] args) {  
    String baseUrl = "file:///C:/newhtml.html";  
    WebDriver driver = new FirefoxDriver();  
  
    driver.get(baseUrl);  
    String innerText = driver.findElement(By  
        .xpath("//table/tbody/tr[2]/td[2]/table/tbody/tr/td[2]"))  
        .getText();  
    System.out.println(innerText);  
    driver.quit();  
}
```

The output below confirms that the inner table was successfully accessed.

The inner text was  
successfully retrieved



### Using Attributes as Predicates:

If the element is written deep within the HTML code such that the number to use for the predicate is very difficult to determine, we can use that element's unique attribute instead. In the example below, the "New York to Chicago" cell is located deep into Mercury Tours homepage's HTML code.

Specials	
Atlanta to Las Vegas	\$398
Boston to San Francisco	\$513
Los Angeles to Chicago	\$168
New York to Chicago	\$198
Phoenix to San Francisco	\$213

```

<body>
  <div>
    <table height="100%" cellspacing="0" cellpadding="0" border="0">
      <tbody>
        <tr>
          <td valign="top" bgcolor="#003366">
            <td valign="top">
              <table cellspacing="0" cellpadding="0" border="0">
                <tbody>
                  <tr>
                  <tr>
                  <tr>
                  <tr>
                </tbody>
                <td>
                  <table cellspacing="0" cellpadding="0" border="0">
                    <tbody>
                      <tr>
                    </tbody>
                    <tr>
                      <td width="14"> </td>
                    </tr>
                    <td>
                      <table width="492" cellspacing="0" cellpadding="0" border="0">
                        <tbody>
                          <tr> </tr>
                          <tr>
                            <td width="273" valign="top">
                              <p>
                                <table width="100%" cellspacing="0" cellpa
                                  <tbody>
                                    <tr>
                                    <tr>
                                    <tr valign="top">
                                  </tbody>
                                <td height="101">
                                  <table width="270" cellspac
                                    <tbody>
                                      <tr bgcolor="#CCCCCC">
                                        <td width="80%">
                                          <font size="2">
                                            Vegas</font>
                                          </td>
                                        <td width="20%">

```

this is the `<table>` that holds the New York to Chicago cell. Notice that it is buried deep into the HTML code, and the number to use as predicate is difficult to determine.

In this case, we can use the table's unique attribute (`width="270"`) as the predicate. Attributes are used as predicates by prefixing them with the `@` symbol. In the example above, the "New York to Chicago" cell is located in the first `<td>` of the fourth `<tr>`, and so our XPath should be as shown below.

Remember that when we put the XPath code in Java, we should use the escape character backward slash "\" for the double quotation marks on both sides of "270" so that the string argument of By.xpath() will not be terminated prematurely.

```
//table[@width="270"]/tbody/tr[4]/td
```

```
By.xpath("//table[@width=\"270\"]/tbody/tr[4]/td"))
```

*use the escape characters here*

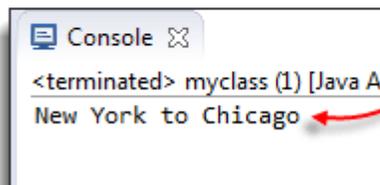
We are now ready to access that cell using the code below.

```
public static void main(String[] args) {
    String baseUrl = "http://newtours.demoaut.com/";
    WebDriver driver = new FirefoxDriver();

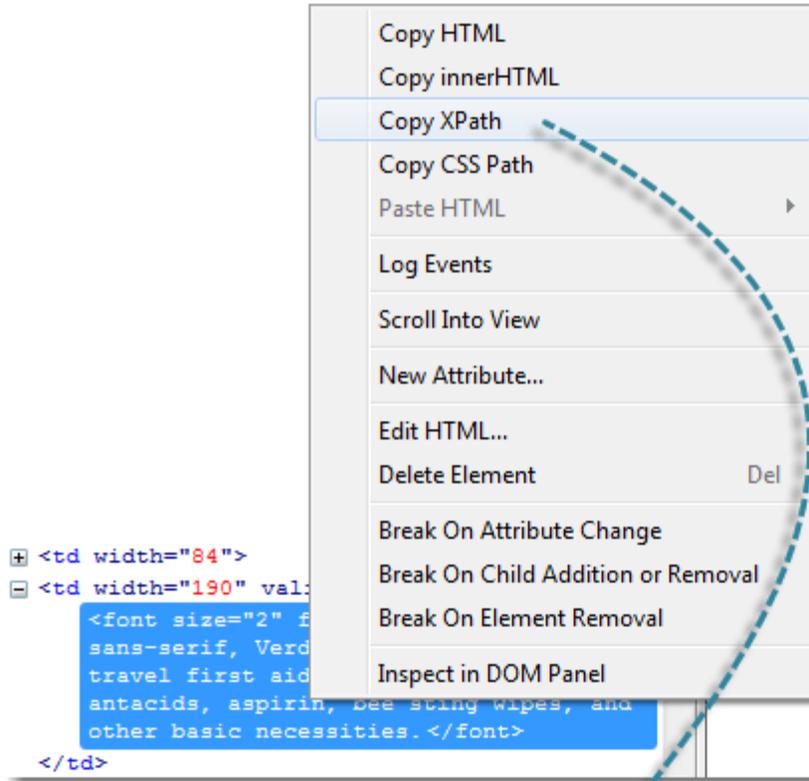
    driver.get(baseUrl);
    String innerText = driver.findElement(By
        .xpath("//table[@width=\"270\"]/tbody/tr[4]/td"))
        .getText();
    System.out.println(innerText);
    driver.quit();
}
```

We are now ready to access that cell using the code below.

*the inner text was  
successfully retrieved.*



```
Console [X]
<terminated> myclass (1) [Java A
New York to Chicago
```

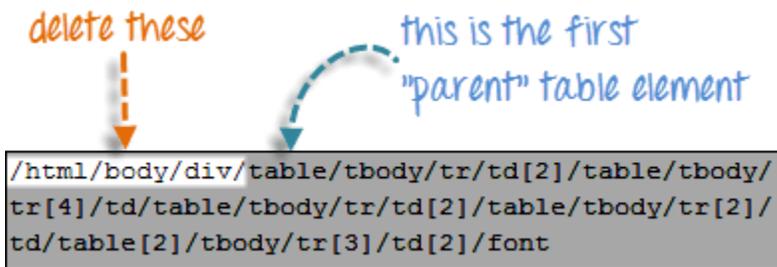


```

/html/body/div/table/tbody/tr/td[2]/table/tbody/
tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[2]/
td/table[2]/tbody/tr[3]/td[2]/font

```

Step 2  
Look for the first "table" parent element and delete everything to the left of it.



Step 3  
Prefix the remaining portion of the code with double forward slash "/" and copy it over to your WebDriver code.

The remaining portion of the code, trimmed and prefixed with "//"

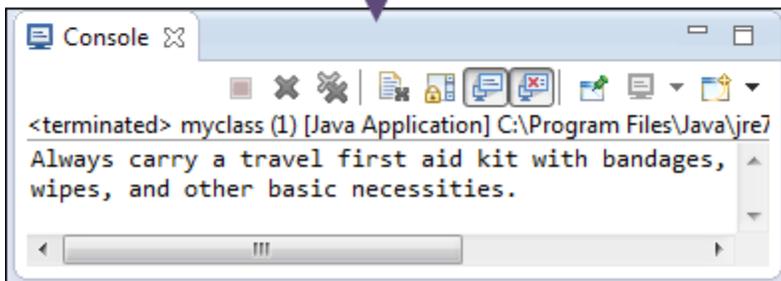
```
//table/tbody/tr/td[2]/table/tbody/tr[4]/td/table/tbody/tr/td[2]/table/tbody/tr[2]/td/table[2]/tbody/tr[3]/td[2]/font
```

```
By.xpath("//table/tbody/tr/td[2]"  
+ "/table/tbody/tr[4]/td"  
+ "/table/tbody/tr/td[2]"  
+ "/table/tbody/tr[2]/td"  
+ "/table[2]/tbody/tr[3]/td[2]/font"))
```

When pasted onto the `By.xpath()` method

The WebDriver code below will be able to successfully retrieve the inner text of the element we are accessing.

```
public static void main(String[] args) {  
    String baseUrl = "http://newtours.demoaut.com/";  
    WebDriver driver = new FirefoxDriver();  
  
    driver.get(baseUrl);  
    String innerText = driver.findElement(  
        By.xpath("//table/tbody/tr/td[2]"  
        + "/table/tbody/tr[4]/td"  
        + "/table/tbody/tr/td[2]"  
        + "/table/tbody/tr[2]/td"  
        + "/table[2]/tbody/tr[3]/td[2]/font"))  
        .getText();  
    System.out.println(innerText);  
    driver.quit();  
}
```



## How to verify Page Title in Selenium WebDriver

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.Test;

Public class verifyTitle {
    WebDriver driver = new FirefoxDriver();
    String title;

    @Test
    public void getTitlePage()
    {
        driver.get("https://www.salesforce.com/login");
        driver.manage().window().maximize();

        title = driver.getTitle();

        System.out.println("Page title is: "+title);

        Assert.assertTrue(title.contains("salesforce.com"));

        driver.close();
    }
}
```

Output of the above code:

Page title is: salesforce.com - Customer Secure Login Page

## Handle Radio Button and Checkbox In Selenium Webdriver

1 HTML For Checkbox

2

3 `<input type="checkbox">`

1 For Radio Button

2

3 `<input type="radio">`



## Radio button and Checkbox in Selenium

When you inspect these elements via firebug and firepath you will get above html type.

The main difference between radio button and checkbox is checkbox you can select multiple but for radio button, only one selection is possible.

In Selenium we have 1 method called `click()` to perform click events.

This `click()` method you can apply with radio button, checkbox, links and sometime with dropdown as well.

```
WebElement ele=driver.findElement(By.id());
ele.click();
```

Before performing click action, sometimes we need to verify some activity as well, take some example

- You need to verify whether radio button or checkbox is enabled.
- You need to verify whether radio button or checkbox is Displayed on UI or not.
- You need to verify whether checkbox and radio button is default selected or not.

These words look quite big while listening but we can easily verify this using some predefined method in Selenium.

```
1 isDisplayed();
2
3 isEnabled();
4
5 isSelected();
```

Therefore, you must be eager now how to use these methods in script so let us see these methods using a single program.

```
import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

public class FacebookDropdown {

    public static void main(String[] args) throws Exception {

        WebDriver driver=new FirefoxDriver();

        driver.manage().window().maximize();

        driver.get("http://www.facebook.com");

        WebElement male_radio_button=driver.findElement(By.id("u_0_e"));

        boolean status=male_radio_button.isDisplayed();

        System.out.println("Male radio button is Displayed >>"+status);

        boolean enabled_status=male_radio_button.isEnabled();

        System.out.println("Male radio button is Enabled >>"+enabled_status);

        boolean selected_status=male_radio_button.isSelected();

        System.out.println("Male radio button is Selected >>"+selected_status);

        male_radio_button.click();
```

```

boolean selected_status_new=male_radio_button.isSelected();

System.out.println("Male radio button is Selected >>" +selected_status_new);

}

}

```

**Explanation-** If you notice above scenario before click Selected status was false but after click status changed to TRUE.

**Check below image for output.**

```

FacebookDropdown [Java Application] C:\Program Files\Java\jdk1.7.0_71\bin\javaw.exe (Apr 23, 2015, 3:29:27 PM)
Male radio button is Displayed >>true
Male radio button is Enabled >>true
Male radio button is Selected >>false
Click on Radio button
Male radio button is Selected >>true

```

## How to handle or select values from dropdown in Selenium webdriver

### Select values from Dropdown in Selenium webdriver

For handling dropdowns Selenium already provides Select class that has some predefined method which help is a lot while working with Dropdown.

Select class can be find in below package.

```
1 org.openqa.selenium.support.ui.Select
```

### Select values Dropdown in Selenium webdriver

Let's discuss some of the method and will see detailed program as well.

- **Select value using Index.**

```
WebElement month_dropdown=driver.findElement(By.id("month"));
```

```
2 Select month=new Select(month_dropdown);
3 month.selectByIndex(4);
```

Explanation- Here selectbyIndex(int) is method which accept integer as argument so depends on index it will select values. If you give index as 4, it will select 5<sup>th</sup> value.

Select value using value attribute

```
1 WebElement month_dropdown=driver.findElement(By.id("month"));
2
3 Select month=new Select(month_dropdown);
4
5 month.selectByValue("5");
```

Explanation- Here selectByValue(String) is a method which accept values it means whatever value you have in your dropdown. Please refer below screenshot to see how to get values from dropdown.

In our example we have taken value as 5 it means it will select May from dropdown.

- **Select value from Visible text**

```
1 WebElement month_dropdown=driver.findElement(By.id("month"));
2
3 Select month=new Select(month_dropdown);
4
5 month.selectByVisibleText("Aug");
```

**Explanation-** We can also select value from text as well. This is very straight forward that whatever text we are passing it simply select that value.

Note- This is case sensitive it means if I pass Aug and dropdown has aug then Selenium will not be able to select value and it will fail your program so make sure Text which you are passing is correct.

In above example I am passing Aug in argument so it will select aug from dropdown.

- **Get Selected option from Dropdown.**

```
1 WebElement month_dropdown=driver.findElement(By.id("month"));
2
3 Select month=new Select(month_dropdown);
4
5 WebElement first_value=month.getFirstSelectedOption();
6
7 String value=first_value.getText();
```

Explanation- Here getFirstSelectedOption() will return the first selected option and will return you the WebElement then you can use getText() method to extract text and validate the same based on your requiremen

- **Get All option from dropdown**

```
1 WebElement month_dropdown=driver.findElement(By.id("month"));
2
3 Select month=new Select(month_dropdown);
4
5 List<WebElement> dropdown=month.getOptions();
6
7 for(int i=0;i<dropdown.size();i++){
8
9 String drop_down_values=dropdown.get(i).getText();
10
11 System.out.println("dropdown values are "+drop_down_values);
12
13 }
```

# Firefox Profile

## What is profile?

Firefox saves your personal information such as bookmarks, passwords, and user preferences in a set of files called your **profile**, which is stored in a separate location from the Firefox program files.

You can create multiple profiles in the firefox browser, each containing a separate set of user information.

Profile manager is the responsibility to create, remove, rename and switch profiles

- **Note: Profile is supporting for Firefox Browser**

### Why do I need New profile

The default Firefox profile is not very automation friendly. When you want to run automation reliably On Firefox browser it is advisable to make a separate profile.

### Some of Uses in real time:

- Download file
- Handling Certificates
- Handling proxy
- Working with grid(You can assign to each Selenium grid 2 node a specific firefox profile)

### Finding Your Profile Folder

The following table shows the typical location of the default profile:

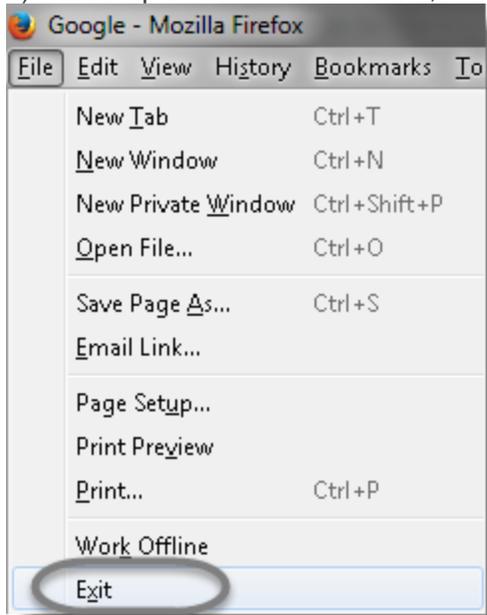
Operating System	Profile Folder Path
Windows XP / 2000 / ..	%AppData%MozillaFirefoxProfilesxxxxxxxx.default
Linux	~/.mozilla/firefox/xxxxxxxx.default/
Mac OS X	~/Library/Application Support/Firefox/Profiles/xxxxxxxx.default/

### Creating a New Profile

Creating a New Firefox profiles and use the same in the Test script involves three steps process. First you need to Start the Profile Manager, second is to Create a New Profile and third is to use the same profile in Test scripts.

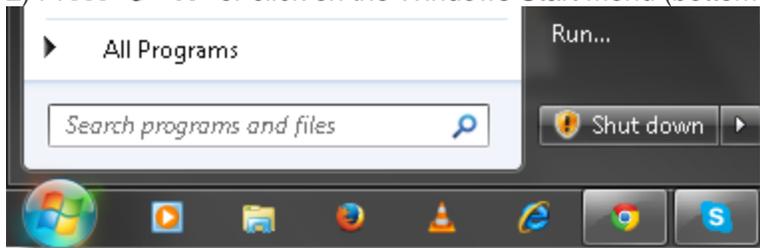
### Step 1: Starting the Profile Manager

1) At the top of the Firefox window, click on the **File** menu and then select **Exit**.

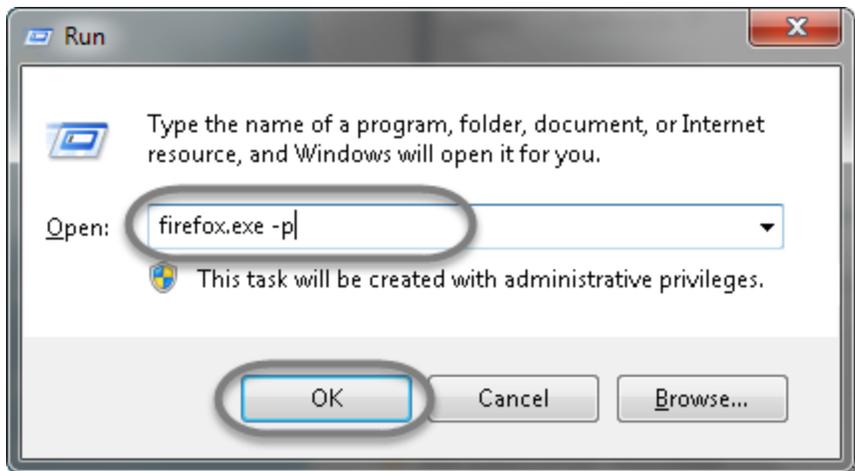


2) Press '**Windows Key + R**' or click on the Windows Start Menu (bottom left button) and then select **Run**.

2) Press '**Windows Key + R**' or click on the Windows Start Menu (bottom left button) and then select **Run**.



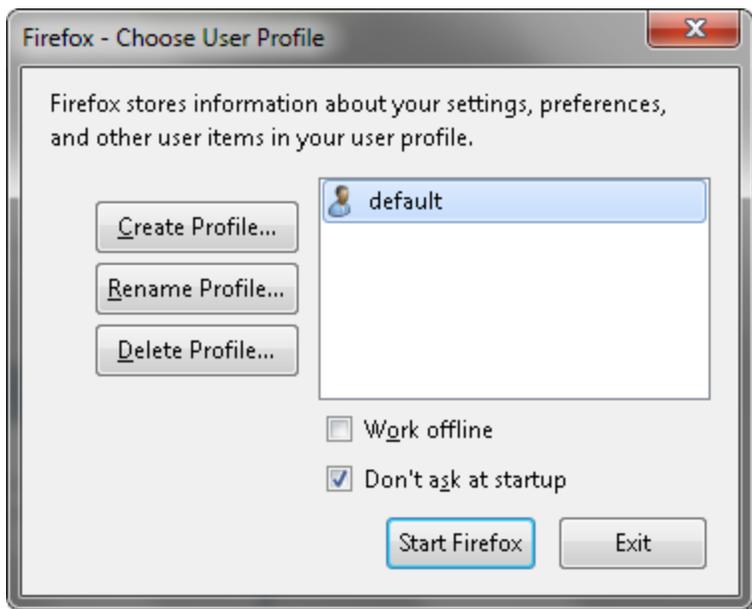
3) In the **Run** dialog box, type in: '`firefox.exe -p`' and then Click **OK**.



**Note:** If the Profile Manager window does not appear, it may be opened in the background. It needs to be closed properly, you may use Ctrl+Alt+Del program to kill it. If it still does not open then you may need to specify the full path of the Firefox program, enclosed in quotes; for example:

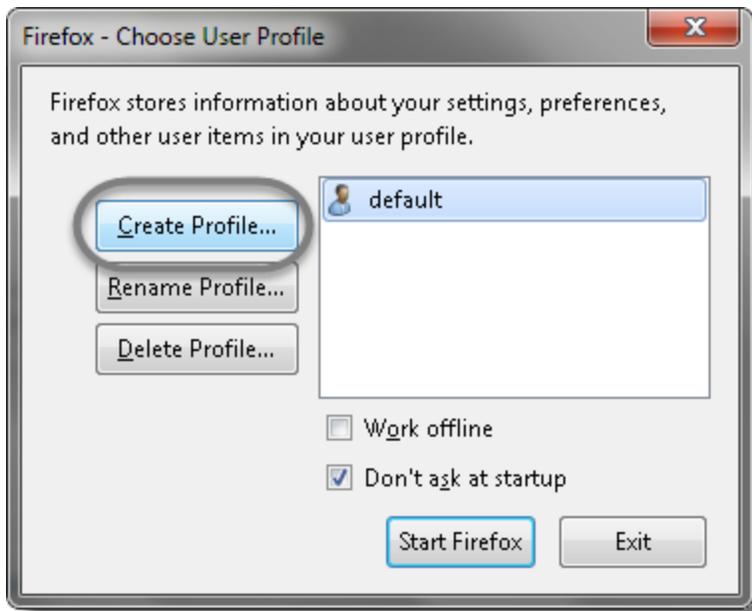
- On **32-bit** Windows: "C:Program Files\Mozilla Firefox\firefox.exe" -p
- On **64-bit** Windows: "C:Program Files (x86)\Mozilla Firefox\firefox.exe" -p

4) The Choose User Profile window will look like this.

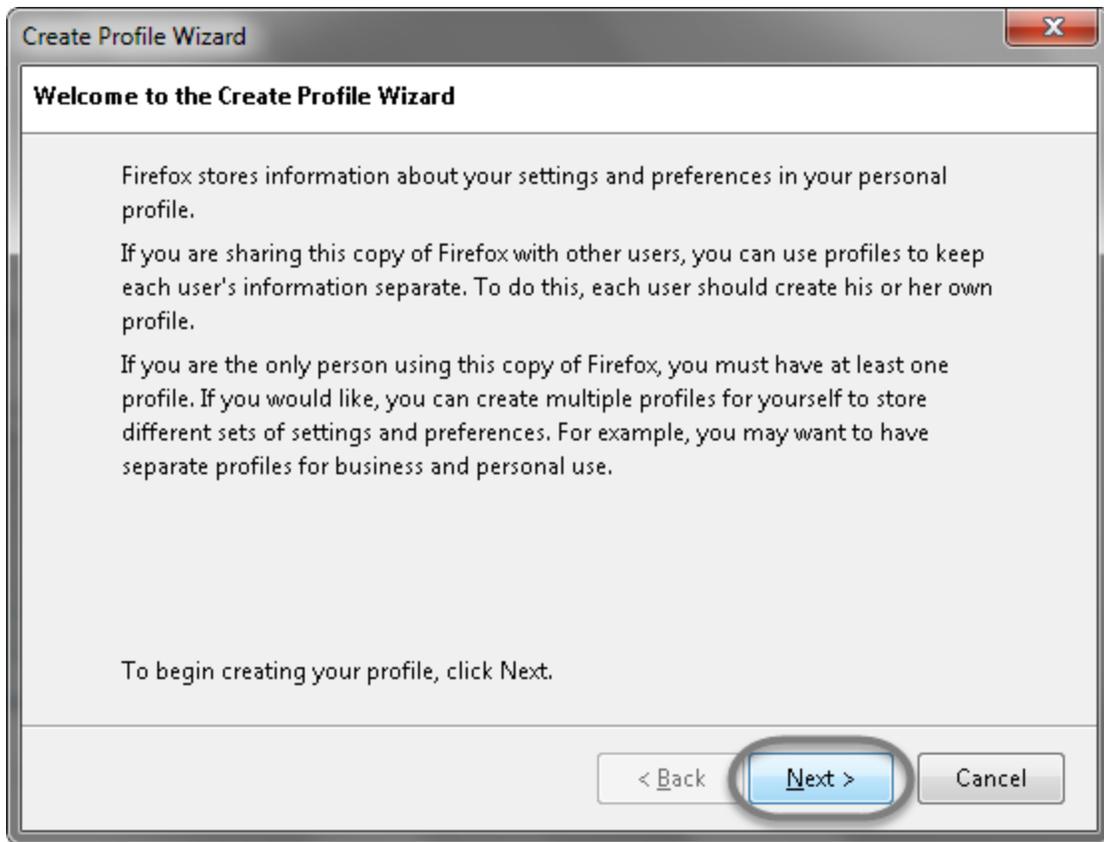


## Step 2: Creating a Profile

1) Click the 'Create Profile...' button on the 'Firefox – Choose User Profile' window that comes up.



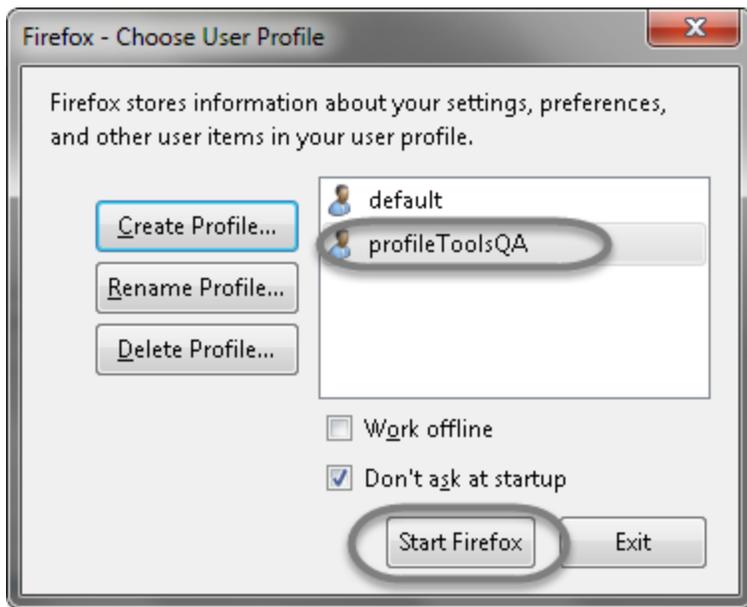
2) Click 'Next >' in the 'Create Profile Wizard' window that comes up.



3) Type in a new name 'profileToolsQA' in the 'Enter new profile name' box and click 'Finish'.



4) 'Choose User Profile' window will display the newly created profile in the list.



5) Click on the 'Start Firefox' box. Firefox will start with the new profile.

**Note:** You will notice that the new Firefox window will not show any of your Bookmarks and Favorite icons.

**Note:** The last selected profile will then start automatically when you next start Firefox and you will need to start the Profile Manager again to switch profiles.

### Step 3: User Custom Profile in Selenium

Once the Automation Profile is created, it needs to be called in the Test scripts. You can now add the below code to your Test scripts to Instantiating the Firefox Driver:

```
ProfilesIni profile = new ProfilesIni();  
FirefoxProfilemyprofile = profile.getProfile("profileToolsQA");  
WebDriver driver = new FirefoxDriver(myprofile);
```

### WebDriver commands

#### Get Command

**Purpose:** This command is use to **open** a new web page in the current browser.

**Command:** driver.get(URL);

**Parameters:** url – The URL to load. It is best to use a fully qualified URL

```
driver.get ("www.google.com") ;
```

#### Get Title Command

**Purpose:** **This command is use to get the title of the current page.**

```
driver.getTitle () ;
```

#### Get Current URL Command

**Purpose:** This command is use to get the **URL** of the page currently loaded in the browser.

```
driver.getCurrentUrl () ;
```

## Get Page Source Command

**Purpose:** This command is use to get the **source** of the last loaded page.

```
driver.getPageSource ();
```

## Close Command

**Purpose:** This command is use to **close** the current window of the browser, if it's the last window it will close the browser.

```
driver.close();
```

## Quit Command

**Purpose:** This command is use to **quit** the browser and all the opened windows in the browser..

```
driver.quit();
```

## Refresh command

**Purpose:** This command is use to **refresh** the current browser.

```
driver.refresh ();
```

## Refresh browser in different ways.

- 1) **Refresh command:** Most commonly used and simple command for refreshing a webpage.

```
driver.get ("https://www.google.com");  
driver.navigate ().refresh ();
```

- 2) **SendKeys command:** Second most commonly used command for refreshing a webpage. As it is using a send keys method, we must use this on any Text box on a webpage.

```
driver.get ("https://www.google.com");  
driver.findElement (By.id ("fdff")).sendKeys ("Keys.F5");
```

- 3) **Get command:** This is a tricky one, as it is using another command as an argument to it. If you look carefully, it is just feeding get command with a page url.

```
driver.get ("https://www.google.com ");  
driver.get (driver.getCurrentUrl ());
```

4) **To command:** This command is again using the same above concept. navigate ( ).to ( ) is feeding with a page url and an argument.

```
driver.get("https://www.google.com");
driver.navigate.to (driver.getCurrentUrl ());
```

**SendKeys command:** This is same SendKeys command but instead of using Key, it is using ASCII code

```
driver.get("https://www.google.com");
driver.findElement (By.id ("fdff")).sendKeys ("\uE035");
```

## Working with Chrome driver & I.E:

### Chromedriver:

- In order to work with chrome driver, we must set system set property.
- Seleniumhq guys treated as Third party browser.
- You need to download chromedriver.exe from  
a) <http://docs.seleniumhq.org/download/>

### Third Party Drivers, Bindings, and Plugins

Selenium can be extended through the use of plugins. Here are a number of plugins maintained by third parties. For more information on how to create your own plugin consult the docs.

Please note that these plugins are not supported, maintained, hosted, or endorsed by the Selenium project. In addition, be advised that the plugins listed below are not necessarily licensed under the Apache License v.2.0. Some of the plugins are available under another free and open source software license; others are only available under a proprietary license. Any questions and their license of distribution need to be raised with their respective developer(s).

### Third Party Browser Drivers **NOT DEVELOPED** by seleniumhq

#### Browser

<a href="#">Chrome</a>	<a href="#">2.14</a>	<a href="#">change log</a>	<a href="#">issue tracker</a>	<a href="#">selenium wiki page</a>
<a href="#">Opera</a>	<a href="#">0.1.0</a>		<a href="#">issue tracker</a>	<a href="#">selenium wiki page</a>
<a href="#">GhostDriver</a>	(PhantomJS)		<a href="#">issue tracker</a>	<a href="#">SeConf talk</a>
<a href="#">Windows Phone</a>			<a href="#">issue tracker</a>	

- b) click on latest version

# ChromeDriver

WebDriver is an open source tool for automated testing of webapps across many browsers. It provides capabilities for navigating to web pages, user input, JavaScript execution, and more. ChromeDriver is a standalone server which implements WebDriver's wire protocol with Chromium. ChromeDriver is available for Chrome on Android and Chrome on Desktop (Linux, Windows and ChromeOS).

## Latest Release: ChromeDriver 2.15

- All versions available in [Downloads](#)



### Sample source code:

```
//optional, if not specified, Webdriver will search your path for chrome driver
System.setProperty ("webdriver.chrome.driver","path to chromedriver.exe")
WebDriver driver = new ChromeDriver ();
```

**InternetExplorerDriver:** like chrome browser, I.E. also treated as Third party browser by seleniumhq guys. So you must set, System set property.

You need to download I.E.exe from

- a) <http://docs.seleniumhq.org/download/>

### The Internet Explorer Driver Server

This is required if you want to make use of the latest and greatest features of the WebDriver InternetExplorerDriver. Please make sure that this is available on your \$PATH (or %PATH% on Windows) in order for the IE Driver to work as expected.

Download version 2.45.0 for (recommended) [32 bit Windows IE](#) or [64 bit Windows IE](#)  
[CHANGELOG](#)

Depends upon your operating system click on above link either 32 bit or 64 bit version of IE.

**Sample source code:**

```
//optional, if not specified, Webdriver will search your path for iE driver  
System.setProperty ("webdriver.IE.driver","path to IEDriver.exe")  
WebDriver driver = new InternetExplorerDriver ();
```

## Headless browser Testing using Selenium HtmlUnitDriver



If you have ever worked on [Firefox](#), [Chrome](#), IE browser in Selenium you might have faced so many issues like xpath is changing everytime when new feature is getting added. Using these browser, some common issue which I faced during my daily automation like Sync issues and speed as well. Some browser takes time to load in my experience I faced performance issues with IE browser.

To increase speed of your test script i.e. performance of your script we can try some testcases using Headless browser Testing using Selenium.

### **What is Headless testing/Headless browser in Automation?**

#### **Advantage and Disadvantage of headless browsers or Why should I use this?**

#### **How to perform Headless testing in Selenium Webdriver using HTMLUnitDriver.**

### **What are the common issues or limitation while working with Headless browsers?**

So let us discuss each topic separately.

### **What is Headless testing/Headless browser in Automation?**

Ans-A browser, which does not have any GUI it means which runs in background. If you run your programs in Firefox, Chrome, IE and different browser then you can see how the browser is behaving but in headless browsers, you cannot .

#### **Advantage and Disadvantage of headless browsers or Why should I use this?**

Ans-One of the most Important advantage is Performance.

1-When we run your test case using headless browsers then you will get result just in seconds, you will see the result in below program.

2-When we have to run test case to create some sample test data or just you have to verify some messages and functionality then you can should try headless browsers.

3- When we have to run test case in remote machine or server, which does not have any browser, but still you have to execute test case then you can try with headless browsers again.

I hope you get the clear picture of this so let us start with some program and output as well.

There are so many headless browser available in market, which do the same like Nodejs etc.

When you build your test case using [Jenkins](#) then also it run in Headless mode.

## Headless browser Testing using Selenium

To implement Headless testing selenium have inbuilt class known as

HtmlUnitDriver like other browsers like FirefoxDriver, ChromeDriver etc.

You can find the official <https://code.google.com/p/selenium/wiki/HtmlUnitDriver>

Before moving to program, you should have setup ready. If you have not done setup then no worry use download and Install

Java, [Eclipse](#), Selenium Webdriver Jars, Latest Release

```
1 package com.bog.htmldemo;
2
3
4 import org.openqa.selenium.By;
5 import org.openqa.selenium.WebDriver;
6 import org.openqa.selenium.WebElement;
7 import org.openqa.selenium.htmlunit.HtmlUnitDriver;
8
9
10
11 public class HtmlDemoProgram1 {
12     public static void main(String[] args) throws InterruptedException {
13
14         // Declaring and initialize HtmlUnitWebDriver
15         WebDriver driver = new HtmlUnitDriver();
16
17
18         // open facebook webpage
19         driver.get("http://www.facebook.com");
20
21
22         // Print the title
23         System.out.println("Title of the page " + driver.getTitle());
24
25
26         // find the username field
27         WebElement username = driver.findElement(By.id("email"));
28
29
30         // enter username
31         username.sendKeys("mukeshotwani.50@gmail.com");
32
33
34         // find the password field
35         WebElement password = driver.findElement(By.id("pass"));
36
37
38         // Click the loginbutton
39         password.sendKeys("pjs@903998");
40
41
```

```

42 // find the Sign up button
43 WebElement Signup_button = driver.findElement(By.id("loginbutton"));
44
45
46 // Click the loginbutton
47 Signup_button.click();
48
    // wait for 5 second to login
    Thread.sleep(5000);

    // You will get new title after login
    System.out.println("After login title is = " + driver.getTitle());

    }}

```

## Output

Title of the page Welcome to Facebook – Log In, Sign Up or Learn More

After login title is = Facebook

In above program you will notice it will take only 3-4 seconds to give you result but if try the same in other browsers it will take 15-20 seconds.

Javascript support in the [HtmlUnitDriver](#)

All the browser which we are using they use separate javascript engine, but HTMLUnitDriver use [Rhino](#) engine so if you will test some java script applications then chances are high you will get diff results.

By default JavaScript is disabled in HTMLUnitDriver so you have to enable it while writing script.

Two ways to enable javascript in HTMLUnitDriver

**First- setJavaScriptEnabled method to true**

```

1 WebDriver driver = new HtmlUnitDriver();
2
3 driver.setJavaScriptEnabled(true);

```

**Second- While initializing browser itself you can enable.**

```

1 HtmlUnitDriver driver = new HtmlUnitDriver(true);

```

# Advance Scearios

## How to verify Error Message in Selenium Webdriver using Assert

Testing is nothing but verification and validation. Mostly in testing we verify title, some error messages , [tooltip](#) messages and so on.

To capture error message or simple text as well we can use predefined method called **getText()**. **getText()** method will simply capture the error message or text and will return you a String then you can store in String variable and you can use in other application or you can verify as well.

```
package demo;

1
2 import org.openqa.selenium.By;
3 import org.openqa.selenium.firefox.FirefoxDriver;
4 import org.testng.Assert;
5 import org.testng.annotations.Test;
6 public class TestNaukri {
7
8
9
10 @Test
11 public void TestError(){
12     // Open browser
13     FirefoxDriver driver=new FirefoxDriver();
14     // maximize browser
15     driver.manage().window().maximize();
16     // Open URL
17     driver.get("http://www.naukri.com/");
18
19
20
21
22 // Click on login button
23 driver.findElement(By.id("p0submit")).click();
24
25
26 // This will capture error message
27 String actual_msg=driver.findElement(By.id("emailId_err")).getText();
28
29
30 // Store message in variable
31 String expect="plz enter valid email";
32
33
34 // Here Assert is a class and assertEquals is a method which will compare two values if// both
35 matches it will run fine but in case if does not match then it will throw an
36 //exception and fail testcases
37 // Verify error message
38 Assert.assertEquals(actual_msg, expect);
39
40 }
}
```

## Output:

```
FAILED: TestError
java.lang.AssertionError: expected [plz enter valid email] but found [Please enter your Email ID]
    at org.testng.Assert.fail(Assert.java:94)
    at org.testng.Assert.failNotEquals(Assert.java:494)
    at org.testng.Assert.assertEquals(Assert.java:123)
    at org.testng.Assert.assertEquals(Assert.java:176)
    at org.testng.Assert.assertEquals(Assert.java:186)
    at demo.TestNaukri.TestError(TestNaukri.java:30)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
    at java.lang.reflect.Method.invoke(Unknown Source)
    at org.testng.internal.MethodInvocationHelper.invokeMethod(MethodInvocationHelper.java:84)
    at org.testng.internal.Invoker.invokeMethod(Invoker.java:714)
    at org.testng.internal.Invoker.invokeTestMethod(Invoker.java:901)
    at org.testng.internal.Invoker.invokeTestMethods(Invoker.java:1231)
    at org.testng.internal.TestMethodWorker.invokeTestMethods(TestMethodWorker.java:127)
    at org.testng.internal.TestMethodWorker.run(TestMethodWorker.java:111)
    at org.testng.TestRunner.privateRun(TestRunner.java:767)
    at org.testng.TestRunner.run(TestRunner.java:617)
    at org.testng.SuiteRunner.runTest(SuiteRunner.java:334)
    at org.testng.SuiteRunner.runSequentially(SuiteRunner.java:329)
    at org.testng.SuiteRunner.privateRun(SuiteRunner.java:291)
    at org.testng.SuiteRunner.run(SuiteRunner.java:240)
```

You can use `getText()` method for capturing simple text messages as well and using above method we can verify the same.

Note- If text does not match then TestNG will throw `AssertionError` and if we do not use Exception handling then it will simply terminate our application/ program.

## Handle Multiple Windows in Selenium Webdriver

In Selenium, we have the feature that we can get the window name of current window.

In Selenium, we have `getWindowName` method that will return current window name in String form. We have `getWindowNames`, which will return `Set<String>` it means set of window name then we can iterate using iterator.

Scenario-1: In some application you will get some pop up that is nothing but separate window take an example of naukri.com. Therefore, in this case we need to close the popup and switch to parent window

```

import org.openqa.selenium.By;
1 import org.openqa.selenium.WebDriver;
2 import org.openqa.selenium.firefox.FirefoxDriver;
3 import org.openqa.selenium.interactions.Actions;
4
5
6 public class TestNaukri {
7     @Test
8     public void TestPopUp() throws InterruptedException{
9         // Open browser
10        WebDriver driver=new FirefoxDriver();
11
12        // Maximize browser
13        driver.manage().window().maximize();
14
15        // Load app
16        driver.get("http://www.naukri.com/");
17
18        // It will return the parent window name as a String
19        String parent=driver.getWindowHandle();
20
21        // This will return the number of windows opened by Webdriver and will return Set of
22        //rings
23        Set<String>s1=driver.getWindowHandles();
24
25        // Now we will iterate using Iterator
26        Iterator<String> I1= s1.iterator();
27
28        while(I1.hasNext()){
29
30            String child_window=I1.next();
31
32            // Here we will compare if parent window is not equal to child window then we will close
33
34            if(!parent.equals(child_window)) {
35                driver.switchTo().window(child_window);
36                System.out.println(driver.switchTo().window(child_window).getTitle());
37                driver.close();
38            }
39        }
40
41        // once all pop up closed now switch to parent window
42        driver.switchTo().window(parent); } }

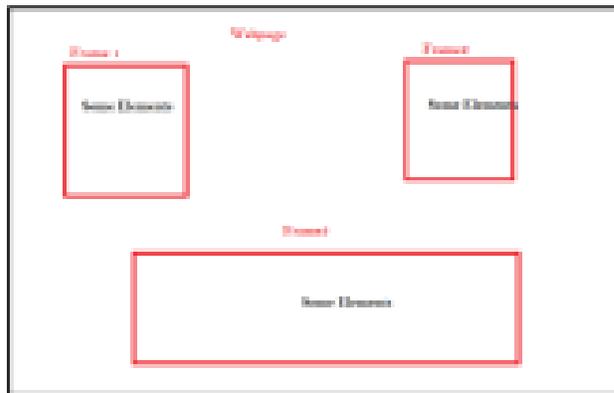
```

We can switch to window using title and content also.

## Handle Frames in Selenium Webdriver and Handle Iframe in Selenium

In some application for better visibility developer use frame concept in webpages.

In this case, if your element exist in frames then we have to switch to frame first then we have to perform our operation.



Handle frames in Selenium

In Selenium we can switch to frames using `switchTo ()` method then perform the action on that element

Syntax-

**`driver.switchTo().frames();`**

Note- You can give frame name or id or index or WebElement .

Refer below snapshot to check different method available to switch frames

Syntax 1-

```
try {  
    driver.switchTo().frame(indexnumber);  
} catch (NoSuchFrameException e) {  
    System.out.println(e.getMessage());  
}
```

We have enclosed our code with try and catch if now frame will not available the this throw exception NoSuchFrameException

Syntax 2-

In this scenario if you know the name of frames in webpage then using name also, you can easily switch

```
try {
    driver.switchTo().frame("framename");
} catch (NoSuchFrameException e){
    System.out.println(e.getMessage());
}
```

Check below screenshot that how we can identify that whether element is inside frame or not

Syntax 3-

In this scenario

```
3
4
5 try {
6     WebElement button=driver.findElement(By.xpath(""));
7     driver.switchTo().frame(button);
8 } catch (NoSuchFrameException e) {
9     System.out.println(e.getMessage());
10 }
11
12
```

Until you are in frames you can not perform any operation so once we are don with frame then switchTo parent window

Syntax-

```
driver.switchTo().defaultContent();
```

## How to Handle alert in Selenium webdriver

In some application while submitting a form or for confirmation a window comes that window known as Alert.

Until you do not handle alert window you cannot perform any action in parent window.

Below is the screenshot of Alert window.



Handle Alert in Selenium

### Handle alert in selenium

To handle alert window we have predefined class known as Alert class.

Alert class methods-

- 1- accept()- Will click on ok button when alert comes.
- 2- dismiss()- Will click on cancel button when alert comes.

Note- Since alert is separate window so before using these methods we have to switch to alert window using switchTo() method

Scenario to –

- Step 1-Open Firefox and open KSRTC website.
- Step 2- Click Search Availability Service button.
- Step 3- Capture the Alert text.

Step 4- Click on OK button.

You can refer [My Youtube](#) Video for the same

Program to Handle Alert in Selenium

```
1 import org.openqa.selenium.By;
2 import org.openqa.selenium.WebDriver;
3 import org.openqa.selenium.firefox.FirefoxDriver;
4 import org.testng.Assert;
5 import org.testng.annotations.Test;
6
7 public class TestAlert {
8
9     @Test
10    public void TestAL() throws InterruptedException{
11
12        // Load Firefox browser
13
14        WebDriver driver=new FirefoxDriver();
15
16        // Open KSRTC website
17
18        driver.get("http://www.ksrtc.in/site/");
19
20        // Maximize the window
21
22        driver.manage().window().maximize();
23
24        // Click on Search Availability Service button
25
26        driver.findElement(By.xpath("//*[ @id='Table_3']/tbody/tr[1]/td[2]/div/a")).click();
27
28        // Switch to alert window and capture the text and print
29
30        System.out.println(driver.switchTo().alert().getText());
31
32        // Pause testcase for 5 second
33
34        Thread.sleep(5000);
35
36        // click on ok button
37
38        driver.switchTo().alert().accept();
```

```
39
40     // Close browser
41
42     driver.quit();
43 }
44 }
```

Sample code to create method

```
1
2
3
4
5
6
7 public static void handleAlert(WebDriver driver){
8
9     if(isAlertPresent(driver)){
10         Alert alert = driver.switchTo().alert();
11         System.out.println(alert.getText());
12         alert.accept();
13     }
14 }
15
16 public static boolean isAlertPresent(WebDriver driver){
17
18     try{
19         driver.switchTo().alert();
20         return true;
21     }catch(NoAlertPresentException ex){
22         return false;
23     }
24 }
25
26
27
28
29
30
31
```

Important point- If alert is not present in window and still we try to switchTo alert window then Selenium will throw NoAlertPresentException which will terminate your program so better you should use exception handle also in your script.

## How to capture Screenshot in Selenium Webdriver

In automation, it is mandatory to take screenshot for verification so that can prove also that your test case has covered certain functionality or not.

Screenshots helps you a lot when your test case will fail so that you can identify what went wrong in your script or in your application

How to capture Screenshot in Selenium webdriver

For taking screenshots Selenium has provide TakesScreenShot interface in this interface you can use getScreenshotAs method which will capture the entire screenshot in form of file than using FileUtils we can copy screenshots from one location to another location

Scenario – Open Google and take screenshot

Lets implement the same

Program for Screenshot in Selenium Webdriver

```
1 import java.io.File;
2
3 import java.io.IOException;
4
5 import org.apache.commons.io.FileUtils;
6
7 import org.openqa.selenium.OutputType;
8
9 import org.openqa.selenium.TakesScreenshot;
10
11 import org.openqa.selenium.WebDriver;
12
13 import org.openqa.selenium.firefox.FirefoxDriver;
14
15 import org.testng.annotations.Test;
16
17 public class ScreenshootGoogle {
18
19     @Test
20     public void TestJavaS1()
```

```

21 {
22 // Open Firefox
23 WebDriver driver=new FirefoxDriver();
24
25 // Maximize the window
26 driver.manage().window().maximize();
27
28 // Pass the url
29 driver.get("http://www.google.com");
30
31 // Take screenshot and store as a file format
32 File src= ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
33 try {
34 // now copy the screenshot to desired location using copyFile //method
35 FileUtils.copyFile(src, new File("C:/selenium/error.png"));
36 }catch (IOException e){
37 System.out.println(e.getMessage());
38 }
39 }
40
41
42
43

```

If you use above code then it will take screenshot and will store in C:/selenium and screenshot name will be error.png

Now consider a scenario where you have to take multiple screenshot then above code will be repetitive so for this we will create a small method which capture screenshots and will call this method from our script.

Example-

```

1 public static void captureScreenShot(WebDriver driver){
2
3 // Take screenshot and store as a file format
4 File src= ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
5 try {
6 // now copy the screenshot to desired location using copyFile method
7 FileUtils.copyFile(src, new File("C:/selenium/"+System.currentTimeMillis()+".png"));
8 }catch (IOException e){
9 System.out.println(e.getMessage());
10 }
11
12 }

```

So this method we can call from our test case and call multiple time. Copy the full code and try to execute in Eclipse

```
package com.bog.htmldemo;
```

```
1 import java.io.File;
2 import java.io.IOException;
3 import org.apache.commons.io.FileUtils;
4 import org.openqa.selenium.OutputType;
5 import org.openqa.selenium.TakesScreenshot;
6 import org.openqa.selenium.WebDriver;
7 import org.openqa.selenium.firefox.FirefoxDriver;
8 import org.testng.annotations.Test;
9
10 public class ScreenshootGoogle {
11
12     @Test
13     public void TestJavaS1(){
14
15         // Open Firefox
16         WebDriver driver=new FirefoxDriver();
17
18         //call method
19         ScreenshootGoogle.captureScreenShot(driver);
20
21         // Maximize the window
22         driver.manage().window().maximize();
23         ScreenshootGoogle.captureScreenShot(driver);
24
25         // Pass the url
26         driver.get("http://www.google.com");
27         ScreenshootGoogle.captureScreenShot(driver);
28     }
29
30     public static void captureScreenShot(WebDriver driver) {
31         // Take screenshot and store as a file format
32         File src=((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
33         try {
34             // now copy the screenshot to desired location using copyFile method
35             FileUtils.copyFile(src, new File("C:/selenium/"+System.currentTimeMillis()+".png"));
36         } catch (IOException e){
37             System.out.println(e.getMessage())
38         }
39     }
}
```

So in above program you can see I called captureScreenshot method three times like this you can create reusable actions in your script and you can use the same

## How to create log file using Log4j in Selenium Webdriver

### What is Log File?

Log file is just simple file, which keep track of the record or event or info when any event happens or any software run. This whole process known as logging. We can create log file as simple log file as well as HTML format.

Note- Any file whose extension is .log will be considered.

### Sample Log file

#### 1- Simple log file in text format

```
2024-07-07 15:55:51,003 - Google -INFO - Browser Opened
2024-07-07 15:55:51,041 - Google -INFO - Implicit wait given
2024-07-07 15:55:59,252 - Google -INFO - url opened
2024-07-07 15:55:59,713 - Google -INFO - keyword type
2024-07-07 22:48:10,748 - testcase1 -INFO - browser opened
```

log file in Selenium

#### 2- Log files in HTML format



Time	Level	Class	Message	Thread
2024-07-07 15:55:51,003	INFO	org.openqa.selenium.WebDriver	Browser Opened	main
2024-07-07 15:55:51,041	INFO	org.openqa.selenium.WebDriver	Implicit wait given	main
2024-07-07 15:55:59,252	INFO	org.openqa.selenium.WebDriver	url opened	main
2024-07-07 15:55:59,713	INFO	org.openqa.selenium.WebDriver	keyword type	main
2024-07-07 22:48:10,748	INFO	org.openqa.selenium.WebDriver	browser opened	main

how to create log files in selenium

### Why it is required in Selenium?

We can create log file for our simple script also so we can track or debug our script easily if anything goes wrong in script. For example, if our script is failing at some point then we can track back what went wrong.

### What is log4J

Log4j is free open source tool given by Apache foundation for creating log files It help us to generate log file in various output target.

## How to create log files in selenium

Step 1- Download log4j jar file

Click on the link to download <http://mirrors.ibiblio.org/pub/mirrors/maven/log4j/jars/log4j-1.2.15.jar>

Step 2- Add log4j to your current project

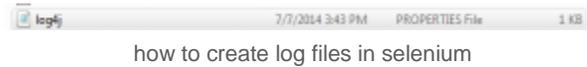
Select your project > Right click > Click on build path> Click Configure build path> Go to library section > Add external jar file > select the log4j > click on save

Step 3- Open notepad and copy the below code and save the file as log4j.properties.

*Note- Please create the log folder inside home directory and while saving use "" to save file*

```
1 // Here we have defined root logger
2 log4j.rootLogger=INFO,CONSOLE,R,HTML,TTCC
3
4 // Here we define the appender
5 log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
6 log4j.appender.R=org.apache.log4j.RollingFileAppender
7 log4j.appender.TTCC=org.apache.log4j.RollingFileAppender
8 log4j.appender.HTML=org.apache.log4j.FileAppender
9
10 // Here we define log file location
11 log4j.appender.R.File=./log/testlog.log
12 log4j.appender.TTCC.File=./log/testlog1.log
13 log4j.appender.HTML.File=./log/application.html
14
15 // Here we define the layout and pattern
16 log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
17 log4j.appender.CONSOLE.layout.ConversionPattern= %5p [%t] (%F:%L)- %m%n
18 log4j.appender.R.layout=org.apache.log4j.PatternLayout
19 log4j.appender.R.layout.ConversionPattern=%d - %c -%p - %m%n
20 log4j.appender.TTCC.layout=org.apache.log4j.TTCCLayout
21 log4j.appender.TTCC.layout.DateFormat=ISO8601
22 log4j.appender.HTML.layout=org.apache.log4j.HTMLLayout
23 log4j.appender.HTML.layout.Title=Application log
24 log4j.appender.HTML.layout.LocationInfo=true
```

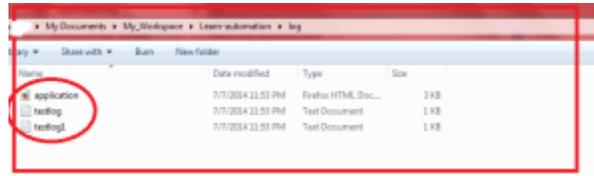
Once you save file it will look like



```
1
2 import java.util.concurrent.TimeUnit;
3 import org.apache.log4j.Logger;
4 import org.apache.log4j.PropertyConfigurator;
5 import org.openqa.selenium.By;
6 import org.openqa.selenium.WebDriver;
7 import org.openqa.selenium.firefox.FirefoxDriver;
8
9
10 public class Google {
11     public static void main(String[] args) {
12
13         // Here we need to create logger instance so we need to pass Class name for
14         //which we want to create log file in my case Google is classname
15         Logger logger=Logger.getLogger("Google");
16
17
18         // configure log4j properties file
19         PropertyConfigurator.configure("Log4j.properties");
20
21
22         // Open browser
23         WebDriver driver = new FirefoxDriver();
24         logger.info("Browser Opened");
25
26         // Set implicit wait
27         driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
28         logger.info("Implicit wait given");
29
30         // Load application
31         driver.get("https://www.google.co.in/");
32         logger.info("Url opened");
33
34         // type Selenium
35         driver.findElement(By.name("q")).sendKeys("Selenium");
36         logger.info("keyword type");
37     }
38 }
39
```

Step 5- Execute your test case and verify the output and log files as well

Finally here is three different log files.



how to create log files in selenium

## What is Listeners and EventFiringWebDriver in Selenium WebDriver

Hello Welcome to Selenium tutorial in this post we will talk about

WebDriverListener,EventFiringWebDriver and WebDriverEventListener in detail.

I know all of you might have heard of Listeners but what exactly Listeners is let us discuss today.In general, terms, Listeners are whom that listen to you and my favorite quotes is “Be a better listener”.



what is listeners in selenium webdriver

If you talk about Webdriver Listener so you should make a note of some classes and interfaces that we will use so will talk about it.

1- WebDriverEventListener – This is an interface, which has some predefined methods so we will implement all of these methods.

2-EventFiringWebDriver- This is a class that actually fire Webdriver event.

## Why we are using Webdriver Listeners

If you talk about Webdriver we are doing some activity like type, click, navigate etc this is all your events which you are performing on your script so we should have activity which actually will keep track of it.

Take an example if you perform click then what should happen before click and after click.

To capture these events we will add listener that will perform this task for us.

## How to implement Listener in our Script

Program for what is listeners in selenium webdriver

```
1 package listerDemo;
2
3 import org.openqa.selenium.By;
4 import org.openqa.selenium.WebDriver;
5 import org.openqa.selenium.WebElement;
6 import org.openqa.selenium.support.events.WebDriverEventListener;
7
8 public class ActivityCapture implements WebDriverEventListener {
9
10 @Override
11 public void afterChangeValueOf(WebElement arg0, WebDriver arg1) {
12 }
13
14 @Override
15 public void afterClickOn(WebElement arg0, WebDriver arg1) {
16     System.out.println("After click "+arg0.toString());
17 }
18
19 @Override
20 public void afterFindBy(By arg0, WebElement arg1, WebDriver arg2) {
21     System.out.println("After FindBy "+arg0.toString());
22 }
23
24 @Override
25 public void afterNavigateBack(WebDriver arg0) {
26     System.out.println("After navigating back "+arg0.toString());
27 }
28
29
30 public void afterNavigateForward(WebDriver arg0) {
31
```

```

32 System.out.println("After navigating forward "+arg0.toString());
33
34 }
35
36 @Override
37 public void afterNavigateTo(String arg0, WebDriver arg1) {
38     System.out.println("After navigating "+arg0.toString());
39     System.out.println("After navigating "+arg1.toString());
40 }
41
42 @Override
43 public void afterScript(String arg0, WebDriver arg1) {
44
45 }
46
47 @Override
48 public void beforeChangeValueOf(WebElement arg0, WebDriver arg1) {
49
50 }
51
52 @Override
53 public void beforeClickOn(WebElement arg0, WebDriver arg1) {
54
55 System.out.println("before click "+arg0.toString());
56
57 }
58
59 @Override
60 public void beforeFindBy(By arg0, WebElement arg1, WebDriver arg2) {
61
62 System.out.println("before FindBY "+arg0.toString());
63
64 }
65
66 @Override
67 public void beforeNavigateBack(WebDriver arg0) {
68
69 System.out.println("Before navigating back "+arg0.toString());
70 }
71
72 @Override
73 public void beforeNavigateForward(WebDriver arg0) {
74 System.out.println("Before navigating Forward "+arg0.toString());
75
76 }
77

```

```

78 @Override
79 public void beforeNavigateTo(String arg0, WebDriver arg1) {
80     System.out.println("Before navigating "+arg0.toString());
81     System.out.println("Before navigating "+arg1.toString());
82 }
83
84 @Override
85 public void beforeScript(String arg0, WebDriver arg1) {
86
87 }
88
89 @Override
90 public void onException(Throwable arg0, WebDriver arg1) {
91
92 System.out.println("Testcase done"+arg0.toString());
93 System.out.println("Testcase done"+arg1.toString());
94 }
95
96 }

```

Let's discuss one of these methods

```

@Override
public void afterClickOn(WebElement arg0, WebDriver arg1) {

System.out.println ("After click "+arg0.toString());

}

```

In above method we are simply printing on console and this method will automatically called once click events done. In same way you have to implement on methods.

*Note- We generally use Listener to generate log events*

**Step 2-** Now create your simple script, create EventFiringWebDriver object, and pass your driver object.

```
EventFiringWebDriver event1=new EventFiringWebDriver(driver);
```

**Step 3-** Create an object of the class who has implemented all the method of WebDriverEventListener so in our case ActivityCapture is a class who has implemented the same.

```
ActivityCapture handle=new ActivityCapture ();
```

**Step 4-** Now register that event using register method and pass the object of ActivityCapture class

```
event1.register(handle);
```

We are done now using event1 object write your script so now let us implement the same

Implementation of Webdriver listener

```
1 package testcases;
2
3 import listerDemo.ActivityCapture;
4
5 import org.openqa.selenium.By;
6 import org.openqa.selenium.WebDriver;
7 import org.openqa.selenium.firefox.FirefoxDriver;
8 import org.openqa.selenium.support.events.EventFiringWebDriver;
9
10 public class ListnerDemo {
11
12 public static void main(String []args){
13
14 System.out.println("Started");
15
16 WebDriver driver=new FirefoxDriver();
17
18 EventFiringWebDriver event1=new EventFiringWebDriver(driver);
19
20 ActivityCapture handle=new ActivityCapture();
21
22 event1.register(handle);
23
24 event1.navigate().to("http://www.facebook.com");
25
26 event1.findElement(By.id("email")).sendKeys("asdsadsa");
27
28 event1.findElement(By.id("loginbutton")).click();
29
30 event1.quit();
31
32 event1.unregister(handle);
33
34 System.out.println("End");
```

```
35 }
36
37 }
```

## Output-

```
Firefox
Before navigating http://www.facebook.com
Before navigating FirefoxDriver Firefox on WINDOWS (9d7f4ac9-9e8d-456a-822e-4dc74bbca3e6)
After navigating http://www.facebook.com
After navigating FirefoxDriver Firefox on WINDOWS (9d7f4ac9-9e8d-456a-822e-4dc74bbca3e6)
Before FindBy(By.id: email)
After FindBy(By.id: email)
Before FindBy(By.id: loginbutton)
After FindBy(By.id: loginbutton)
Before click (|firefoxdriver: firefox on WINDOWS (9d7f4ac9-9e8d-456a-822e-4dc74bbca3e6) -> id: loginbutton)
After click (|firefoxdriver: firefox on WINDOWS (9d7f4ac9-9e8d-456a-822e-4dc74bbca3e6) -> id: loginbutton)
End
```

## How to Handle Untrusted Certificate Selenium Webdriver

- What is Untrusted SSL certificate? Whenever We try to access HTTPS website so many time so will face untrusted SSL certificate issue. This issue comes in all browser like IE, Chrome, Safari, Firefox etc.



### This Connection is Untrusted

You have asked Firefox to connect securely to ██████████ but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

**What Should I Do?**

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

- ▶ **Technical Details**
- ▶ **I Understand the Risks**

2- Why we get this certificate issues often?



**This certificates some in multiple conditions and we should know all of them so that we can rectify them easily.**

- 1- Each secure site has Certificate so it certificate is not valid *up-to date*.
- 2- Certificate has been expired on *date*
- 3 - Certificate is only valid for(site name)
- 4 – The certificate is not trusted because the issuer certificate is unknown due to many reason.

### **Handle Untrusted Certificate Selenium**

**Step 1-**We have to create FirefoxProfile in Selenium.

**Step 2-** We have some predefined method in Selenium called `setAcceptUntrustedCertificates()` which accept Boolean values(true/false)- so we will make it true.

**Step 3-**Open Firefox browser with above created profile.

### Handle untrusted certificate in Firefox

```
1
2
3 import org.openqa.selenium.WebDriver;
4 import org.openqa.selenium.firefox.FirefoxDriver;
5 import org.openqa.selenium.firefox.FirefoxProfile;
6
7 public class SSLCertificate {
8     public static void main(String[] args) {
9
10 //It create firefox profile
11 FirefoxProfile profile=new FirefoxProfile();
12
13 // This will set the true value
14 profile.setAcceptUntrustedCertificates(true);
15
16 // This will open firefox browser using above created profile
17 WebDriver driver=new FirefoxDriver();
18
19 driver.get("pass the url as per your requirement");
20 }
21 }
22
23
```

Since Firefox comes default browser in Selenium so for other browsers like Chrome, IE, Safari we have to use below technique.

### Handle untrusted certificate in Chrome

```
// Create object of DesiredCapabilities class
3 DesiredCapabilities cap=DesiredCapabilities.chrome();
4
5 // Set ACCEPT_SSL_CERTS variable to true
6 cap.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
```

```
7
8 // Set the driver path
9 System.setProperty("webdriver.chrome.driver","Chrome driver path");
10
11 // Open browser with capability
    WebDriver driver=new ChromeDriver(cap);
```

## Handle untrusted certificate in IE

```
1 // Create object of DesiredCapabilities class
2
3 DesiredCapabilities cap=DesiredCapabilities.internetExplorer();
4
5 // Set ACCEPT_SSL_CERTS variable to true
6 cap.setCapability(CapabilityType.ACCEPT_SSL_CERTS, true);
7
8 // Set the driver path
9 System.setProperty("webdriver.ie.driver","IE driver path");
10
11 // Open browser with capability
12 WebDriver driver=newInternetExplorerDriver(cap);
```

## Javascript in Selenium Webdriver and Different usage

### What is JavaScript

JavaScript is the one of the programming language of the Web.

All modern HTML pages are using JavaScript.

For more details, you can refer Wikipedia – [JavaScript](#)

### Use of JavaScript in Selenium

We have used Java in our script and we implemented almost all feature but some features we can't handle using Java so we need scripting language also which can control server side or client side scripting so we will use JavaScript in our Selenium script.



## Implement JavaScript in Selenium

To execute JavaScript in our webdriver script we do not have to write the separate code we have one predefined interface available.

JavascriptExecutor is an Interface which is available in package  
org.openqa.selenium.JavascriptExecutor;

Inside this Interface we have some predefined method called executeScript()- so whatever script you will pass as a String It will be executed by JavascriptExecutor.

Note- This is most common question in interview that how to type in Selenium without using sendKeys method

### Let us implement Javascript in Selenium Webdriver

#### Program 1- How to type in Selenium without using sendKeys() method

```
1 import org.openqa.selenium.JavascriptExecutor;
2 import org.openqa.selenium.firefox.FirefoxDriver;
3
4 public class TestFirefox {
5     public static void main(String[] args) throws InterruptedException {
6         // Open Firefox browser
7         FirefoxDriver driver=new FirefoxDriver();
8
9         // Maximize the window
10        driver.manage().window().maximize();
11
12        // Open application
13        driver.get("enter your application URL");
14
15        // This will execute JavaScript in your script
16        ((JavascriptExecutor)driver).executeScript("document.getElementById('some
17 id').value='mukesh'");
18
19    }
20
21 }
22
```

#### *PROGRAM 2- HOW TO CLICK IN SELENIUM IF BUTTON OR RADIO BUTTON IS DISABLE*

```
1 package seleniumday1;
```

```

2
3 import org.openqa.selenium.JavascriptExecutor;
4 import org.openqa.selenium.firefox.FirefoxDriver;
5
6 public class TestFirefox {
7
8 public static void main(String[] args) throws InterruptedException {
9
10 // Open Firefox browser
11 FirefoxDriver driver=new FirefoxDriver();
12
13 // Maximize the window
14 driver.manage().window().maximize();
15
16 // Open application
17 driver.get("enter your application URL");
18
19 // This will execute JavaScript in your script
20 ((JavascriptExecutor)driver).executeScript("document.getElementById('enter your element
21 id').click()");
22 }
23 }
24

```

### *PROGRAM 3- HOW TO UNCHECK CHECKBOX IN SELENIUM IF CHECKBOX IS DISABLE*

---

```

1 package seleniumday1;
2
3 import org.openqa.selenium.JavascriptExecutor;
4 import org.openqa.selenium.firefox.FirefoxDriver;
5
6 public class TestFirefox {
7
8 public static void main(String[] args) throws InterruptedException {
9
10 // Open Firefox browser
11 FirefoxDriver driver=new FirefoxDriver();
12
13 // Maximize the window
14 driver.manage().window().maximize();
15

```

```

16 // Open application
17 driver.get("enter your application URL");
18
19 // This will execute JavaScript in your script
20 ((JavascriptExecutor)driver).executeScript("document.getElementById('enter element
21 id').checked=false");
22 }
23 }

```

## How to scroll a page in Selenium Webdriver

Hello Welcome to Selenium tutorials, in this post we will see how to scroll a page in Selenium.

Java Script plays very important role in Selenium Webdriver too, earlier also we have seen how to work with JavaScript and [usage of Java Script in Selenium Webdriver](#). We can [highlight elements](#) as well using JavaScript.

Selenium handle scrolling page automatically but if want to scroll page using Selenium then yes we can do easily using Java Script.

**We have method scroll(horizontal, vertical) i.e. scroll(0,400)**

### Scroll page in Selenium Webdriver

To execute Java Script code inside Selenium script we can take the help of JavascriptExecutor

```

1 import org.openqa.selenium.JavascriptExecutor;
2 import org.openqa.selenium.WebDriver;
3 import org.openqa.selenium.firefox.FirefoxDriver;
4
5 public class ScrollTestCase {
6
7     public static void main(String[] args) throws Exception {
8
9         // load browser
10        WebDriver driver=new FirefoxDriver();
11
12        // maximize browser
13        driver.manage().window().maximize();
14
15        // Open Application
16        driver.get("http://jqueryui.com");
17
18        // Wait for 5 second
19        Thread.sleep(5000);

```

```
20
21 // This will scroll page 400 pixel vertical
22 ((JavascriptExecutor)driver).executeScript("scroll(0,400)");
23
24 }
25
26 }
27
28
29
30
31
```

# Waitstatements

## Waiting for WebElements to load

If you have a previous WebUI automation experience, I'm sure you would have come across a situation where your test script couldn't find an element on the webpage as the webpage is still loading. This could happen due to various reasons.

- One classic example is when the application server or webserver is serving the page too slowly due to resource constraints
- The other could be when you are accessing the page on a very slow network.
- The reason could be that the element on the webpage is not loaded by the time your test script tries to find it.

This is where you have to calculate and configure the average wait time your test scripts should wait for WebElements to load on the webpage.

**WebDriver** provides the test script developers a very handy feature to manage wait time.

**Wait time** is the time your driver will wait for the WebElement to load before it **gives up** and **throws NoSuchElementException**.

There are **two ways** by which you can make WebDriver wait for WebElement. They are **implicit wait time and explicit wait time**.

- ❖ Implicit timeouts are common to all the WebElements and has a global timeout period associated to it
- ❖ explicit timeouts can be configured to individual WebElements

## Implicit Wait time

- ❖ **Implicit wait time** is used when you want to configure the **WebDriver's** wait time as a whole for the **application under test**
- ❖ Imagine you have hosted a web application on a **local server and on a remote server**. Obviously, the time to load for a webpage hosted on a local server would be less than the time for the same page hosted on a remote server, due to network latency. Now, if you want to execute your test cases against each of them, you may have to configure the wait time accordingly, such that your test case doesn't end up spending more time waiting for the page or spend far too less time and timeout. To handle these kind of wait time issues, WebDriver gives an option to set the implicit wait time for all of the operations that the driver does using the `manage()` method.

```
public class implicitWaitTime {  
  
    public static void main (String[] args) {  
        WebDriver driver = new FirefoxDriver();  
  
        driver.manage().timeouts().implicitlyWait(10,  
        TimeUnit.SECONDS);  
        driver.get("www.google.com");  
    }  
}
```

## Explicit wait time

### WebDriver Wait

Implicit timeout is generic to all the WebElements of a web page. But, if you have one specific WebElement in your application where you want to wait for a very long time, this approach may not work.

Setting the implicit wait time to the value of this very long time period will delay your entire test suite execution. So you have to make an exception for only a particular case, like this `WebElement`. To handle such scenarios, `WebDriver` has explicit wait time for a `WebElement`.

```
public class ExplicitWait {  
  
    public static void main(String[] args) {  
        WebDriver driver = new FirefoxDriver();  
        driver.get("http://www.google.com");  
  
        WebDriverWait wait = new WebDriverWait(driver,  
20);  
    }  
}
```

## FluentWait

Purpose: Each `FluentWait` instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting, such as `NoSuchElementException` when searching for an element on the page.

```
Public class FluentWait {  
  
    public static void main (String[] args) {  
        WebDriver driver = new FirefoxDriver();  
        driver.get("http://www.commonfloor.com/apartments-for-  
sale");  
  
        new FluentWait<WebDriver>(driver)  
            .withTimeout(10, TimeUnit.SECONDS)  
            .pollingEvery(5, TimeUnit.SECONDS)  
  
            .ignoring(NoSuchElementException.class).until(ExpectedConditions.visi-  
bilityOfElementLocated(By.xpath("//*[@id='listing_loader']/img")));  
    }  
}
```

### **PageLoadTimeout:**

**Purpose:** Sets the amount of time to wait for a page load to complete before throwing an error. If the timeout is negative, page loads can be indefinite.

```
driver.manage ().timeouts ().pageLoadTimeout(100, SECONDS);
```

### **SetScriptTimeout:**

**Purpose:** Sets the amount of time to wait for an asynchronous script to finish execution before throwing an error. If the timeout is negative, then the script will be allowed to run indefinitely.

```
driver.manage ().timeouts ().setScriptTimeout (100, SECONDS);
```

### **Sleep:**

**Purpose:** This is rarely used, as it always force the browser to wait for a specific time. **Thread.Sleep** is never a good idea and that's why Selenium provides wait primitives. If you use them you can specify much higher timeout value which makes tests more reliable without slowing them down as the condition can be evaluated as often as it's required.

```
thread.sleep(1000);
```

## **StaleElementReference Exceptions in Selenium Webdriver**

This Exception occurs when driver is trying to perform action on the element which is no longer exists or not valid.

```
WebElement ele = driver.findElement(By.id("sample"));  
// Before clicking something happened and DOM has changed due to page refresh, or element is
```

removed and re-added  
ele.click();

Now at this point, the element which you're clicking is no longer valid.

so it throws up its hands and gives control to user, who as the test/app author should know exactly what may or may not happen.

In order overcome this, we need to explicitly wait until the DOM is in a state where we are sure that DOM won't change.

# Actions

## Understanding actions, build, and perform

### Learning mouse-based interactions:

There are around eight different mouse actions that can be performed using the Actions class

### The **moveByOffset** Action:

The `moveByOffset()` method is used to move the mouse from its current position to another point on the web page. Developers can specify the X distance and Y distance the mouse has to be moved. When the page is loaded, generally the initial position of a mouse would be (0, 0), unless there is an explicit focus declared by the page.

The API syntax for the `moveByOffset()` method is as follows:

```
public Actions moveByOffset(int xOffset, int yOffset)
```

### Sample code:

```
Actions builder = new Actions (driver);
builder.moveByOffset(three.getLocation().getX()+1,
three.getLocation().getY()+1);
builder.perform();
```

#### The clickAndHold at current location action

The clickAndHold() method is another method of the Actions class that left-clicks on an element and holds it without releasing the left button of the mouse. This method will be useful when executing operations such as drag-and-drop.

#### Sample code:

```
Actions builder = new Actions (driver);
//Move tile3 to the position of tile2
builder.moveByOffset(200, 20)
.clickAndHold()
.moveByOffset(120, 0)
.perform();
```

#### The clickAndHold a WebElement action

If we want to deal with a particular WebElement on the web page, we have to first move the cursor to the appropriate position and then perform the clickAndHold() action.

The API syntax is as follows:

```
public Actions clickAndHold(WebElement onElement)
```

#### Sample code:

```
Actions builder = new Actions(driver);
//Move tile3 to the position of tile2
builder.clickAndHold(webElement)
.moveByOffset(120, 0)
.perform();
```

### The release at current location:

The ultimate action that has to be taken on a held WebElement is to release it so that the element can be dropped or released from the mouse. The release () method is the one that can release the left mouse button on a WebElement.

The API syntax for the release() method is as follows:

**public Actions release()**

#### Sample code:

```
Actions builder = new Actions(driver);
//Move tile3 to the position of tile2
builder.clickAndHold(webElement)
        .moveByOffset(120, 0)
        .release()
        .perform();
```

### The release on another WebElement action

Using this, you can actually release the currently held WebElement in the middle of another WebElement

The API syntax is as follows:

**public Actions release(WebElement onElement)**

```
Actions builder = new Actions(driver);
//Move tile3 to the position of tile2
builder.clickAndHold(three)
        .release(two)
        .perform();
```

**Note:**

Invoking the `release()` or `release(WebElement)` methods without calling the `clickAndHold()` method will result in an undefined behavior.

**The `moveToElement` action**

The `moveToElement()` method is another method of `WebDriver` that helps us to move the mouse cursor to a `WebElement` on the web page.

The API syntax for the `moveToElement ()` method is as follows:

**Public Actions `moveToElement(WebElement toElement)`**

**Sample code:**

```
Actions builder = new Actions(driver);  
//Move tile3 to the position of tile2  
builder.moveToElement(three)  
.clickAndHold()  
.moveByOffset(120,0)  
.perform();
```

**The `dragAndDropBy` Action**

The API syntax for the `dragAndDropBy()` method is as follows:

**public Actions `dragAndDropBy(WebElement source,int xOffset,int yOffset)`**

**sample code:**

```
Actions builder = new Actions (driver);  
builder.dragAndDropBy(dragMe, 300, 200).perform();
```

**The `dragAndDrop` action**

The API syntax for the `dragAndDrop()` method is as follows:

**Public Actions `dragAndDrop(WebElement source,WebElement target)`**

### Sample code:

```
WebElement src = driver.findElement(By.id("draggable"));
WebElement trgt = driver.findElement(By.id("droppable"));
Actions builder = new Actions(driver);
builder.dragAndDrop(src, trgt).perform();
```

### The contextClick on WebElement action

The contextClick() method, also known as right-click, is quite common on many web pages these days. The context is nothing but a menu; a list of items is associated to a WebElement based on the current state of the web page. This context menu can be accessed by a right-click of the mouse on the WebElement. WebDriver provides the developer with an option of emulating that action using the contextClick() method.

### Sample code:

```
Actions builder = new Actions(driver);
builder.contextClick(contextMenu)
.click(driver.findElement(By.name("Item 4")))
.perform();
```

## The getLocation() method

The getLocation action can be executed on all the WebElements. This is used to get the relative position of an element where it is rendered on the web page. This position is calculated relative to the top-left corner of the web page of which the (x, y) coordinates are assumed as (0, 0). This method will be of use if your test script tries to validate the layout of your web page.

The API syntax of the getLocation () method is as follows:

The API syntax of the getLocation() method is as follows:

Point getLocation ()

Sample code:

```
Public class GetLocation {

    Public static void main(String [] args) {
        WebDriver driver = new FirefoxDriver();
        driver.get ("http://www.google.com");
        WebElement searchButton = driver.findElement
        (By.name ("btnK"));
```

## The getSize() method

The getSize action can also be applied on all the visible components of HTML. It will return the width and height of the rendered WebElement.

The API syntax of the getSize () method is as follows:

Dimension getSize()

The preceding method doesn't take any input parameters, and the return type is a class instance named Dimension. This class contains the width and height of the target WebElement.

The following is the code to get the width and height of our favorite Google Search button:

```
Public class GetSize {  
  
    Public static void main (String [] args) {  
        WebDriver driver = new FirefoxDriver ();  
        driver.get ("http://www.google.com");  
        WebElement searchButton = driver.findElement (By.name  
("btnK"));  
        System.out.println (searchButton.getSize ());  
    }  
}
```



# XPATH&CSS

## Locating By CSS Selector:

CSS Selectors are string patterns used to identify an element based on a combination of HTML tag, id, class, and attributes. Locating by CSS Selector is more complicated than ID and Name, but it is the most common locating strategy of advanced Selenium users because it can access even those elements that have no ID or name.

CSS Selectors have many formats, but we will only focus on the most common ones.

- Tag and ID
- Tag and class
- Tag and attribute
- Tag, class, and attribute
- Inner text

## Locating by CSS Selector - Tag and ID:

Syntax	Description
<code>css=tag#id</code>	<ul style="list-style-type: none"><li>tag = the HTML tag of the element being accessed</li><li># = the hash sign. This should always be present when using a CSS Selector with ID</li><li>id = the ID of the element being accessed</li></ul>

Keep in mind that the ID is always preceded by a hash sign (#).

Eg: In Facebook's Email text box in this example. It has id of "email" .

**Step 1.** Navigate to [www.facebook.com](http://www.facebook.com). Using Firebug, examine the "Email or Phone" text box.

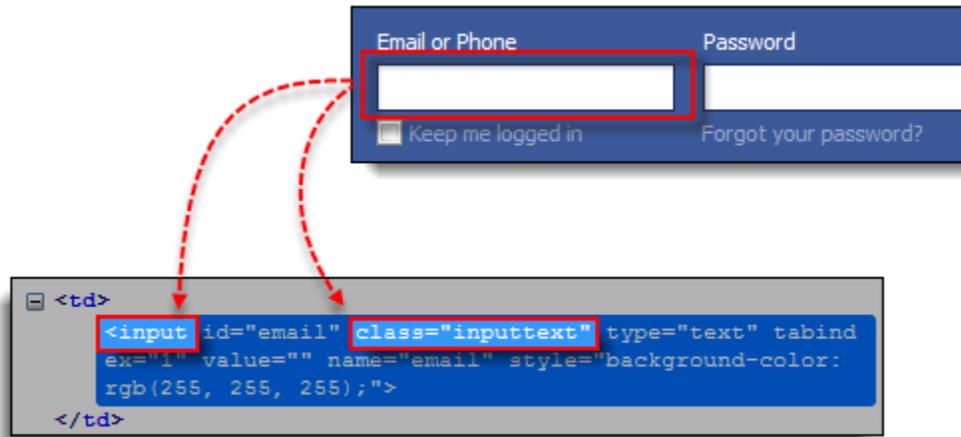
At this point, take note that the HTML tag is "input" and its ID is "email". So our syntax will be "css=input#email".

## Locating by CSS Selector - Tag and Class :

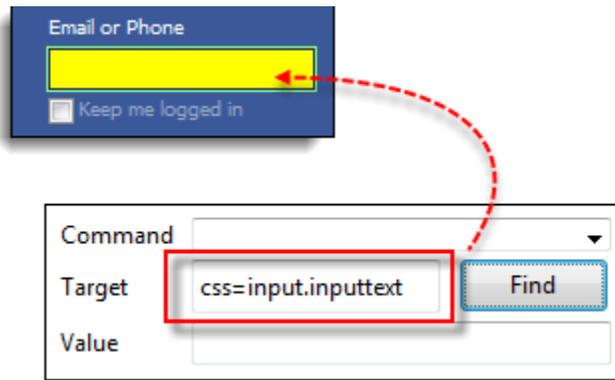
Locating by CSS Selector using an HTML tag and a class name is similar to using a tag and ID, but in this case, a dot (.) is used instead of a hash sign.

Syntax	Description
<code>css=tag.class</code>	<ul style="list-style-type: none"><li>tag = the HTML tag of the element being accessed</li><li>. = the dot sign. This should always be present when using a CSS Selector with class</li><li>class = the class of the element being accessed</li></ul>

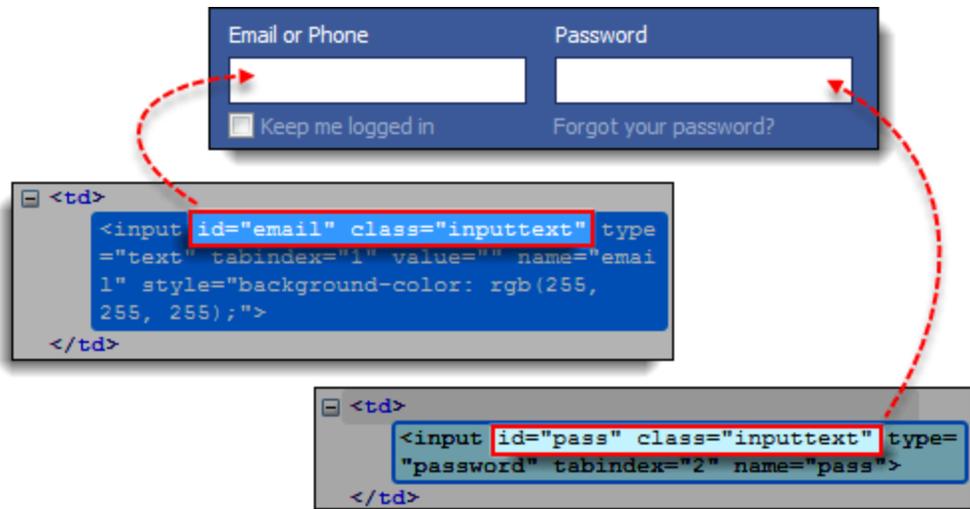
**Step 1.** Navigate to [www.facebook.com](http://www.facebook.com) and use Firebug to inspect the "Email or Phone" text box. Notice that its HTML tag is "input" and its class is "inputtext".



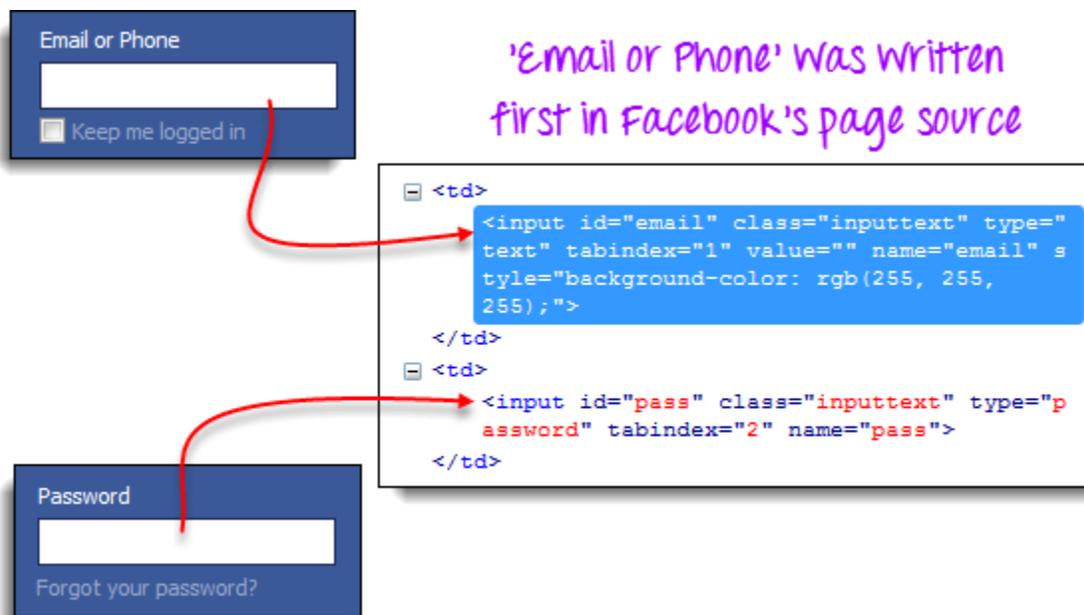
**Step 2.** In Selenium IDE, enter "css=input.inputtext" in the Target box and click Find. Selenium IDE should be able to recognize the Email or Phone text box.



**Take note that when multiple elements have the same HTML tag and name, only the first element in source code will be recognized.** Using Firebug, inspect the Password text box in Facebook and notice that it has the same name as the Email or Phone text box.



The reason why only the Email or Phone text box was highlighted in the previous illustration is that it comes first in Facebook's page source.

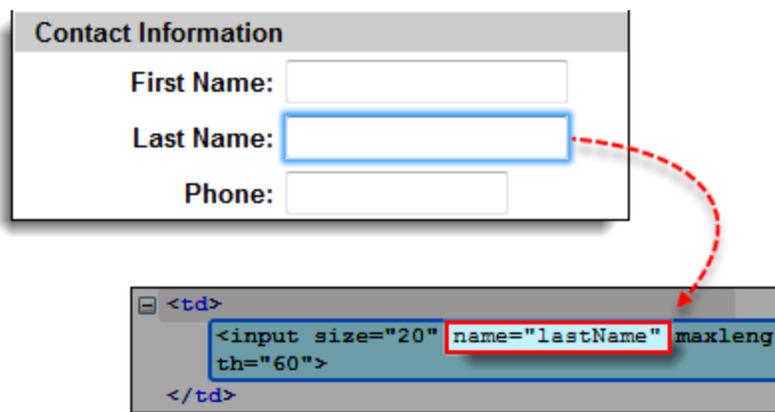


### Locating by CSS Selector - Tag and Attribute

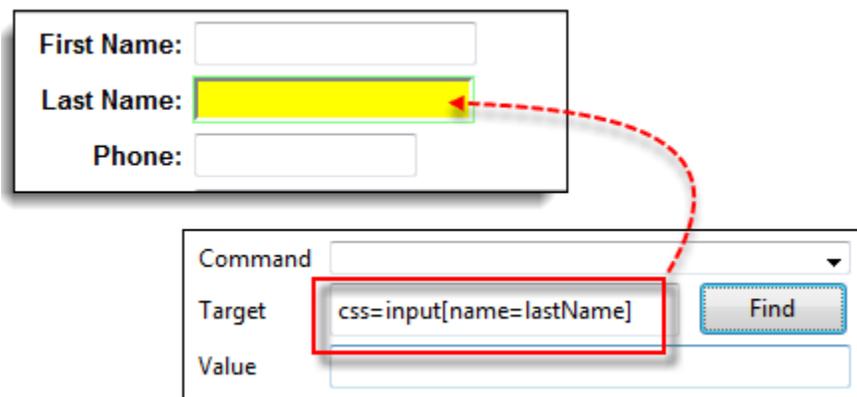
This strategy uses the HTML tag and a specific attribute of the element to be accessed.

Syntax	Description
<code>css=tag[attribute=value]</code>	<ul style="list-style-type: none"> <li>tag = the HTML tag of the element being accessed</li> <li>[ and ] = square brackets within which a specific attribute and its corresponding value will be placed</li> <li>attribute = the attribute to be used. It is advisable to use an attribute that is unique to the element such as a name or ID.</li> <li>value = the corresponding value of the chosen attribute.</li> </ul>

**Step 1.** Navigate to Mercury Tours' Registration page (<http://newtours.demoaut.com/mercuryregister.php>) and inspect the "Last Name" text box. Take note of its HTML tag ("input" in this case) and its name ("lastName").



**Step 2.** In Selenium IDE, enter "css=input[name=lastName]" in the Target box and click Find. Selenium IDE should be able to access the Last Name box successfully.



When multiple elements have the same HTML tag and attribute, only the first one will be recognized. This behavior is similar to locating elements using CSS selectors with the same tag and class.

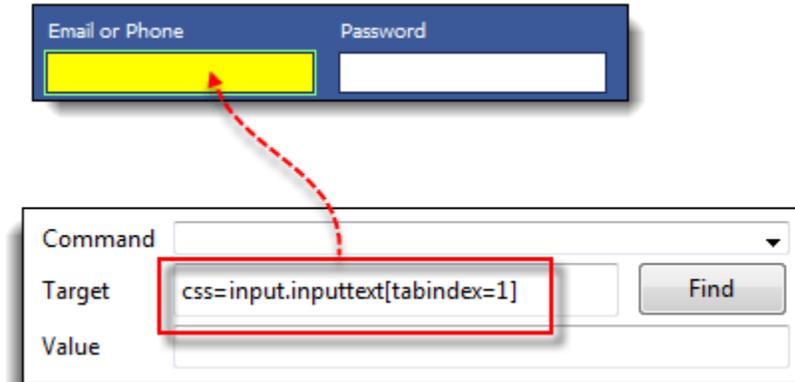
### Locating by CSS Selector - tag, class, and attribute

Syntax	Description
<code>css=tag.class[attribute=value]</code>	<ul style="list-style-type: none"> <li>tag = the HTML tag of the element being accessed</li> <li>. = the dot sign. This should always be present when using a CSS Selector with class</li> <li>class = the class of the element being accessed</li> <li>[ and ] = square brackets within which a specific attribute and its corresponding value will be placed</li> <li>attribute = the attribute to be used. It is advisable to use an attribute that is unique to the element such as a name or ID.</li> <li>value = the corresponding value of the chosen attribute.</li> </ul>

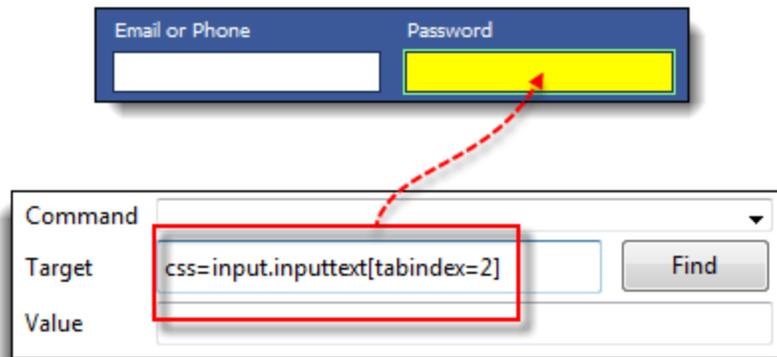
**Step 1.** Navigate to [www.facebook.com](http://www.facebook.com) and use Firebug to inspect the 'Email or Phone' and 'Password' input boxes. Take note of their HTML tag, class, and attributes. For this example, we will select their 'tabindex' attributes.



**Step 2.** We will access the 'Email or Phone' text box first, thus, we will use a tabindex value of 1. Enter "css=input.inputtext[tabindex=1]" in Selenium IDE's Target box and click Find. The 'Email or Phone' input box should be highlighted.



**Step 3.** To access the Password input box, simply replace the value of the tabindex attribute. Enter "css=input.inputtext[tabindex=2]" in the Target box and click on the Find button. Selenium IDE must be able to identify the Password text box successfully.

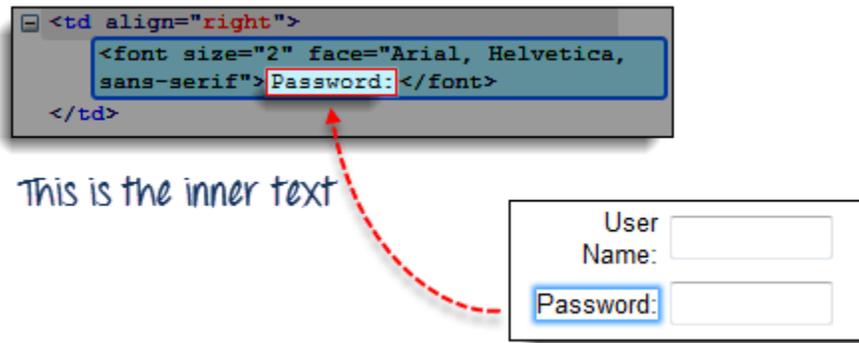


### Locating by CSS Selector - inner text

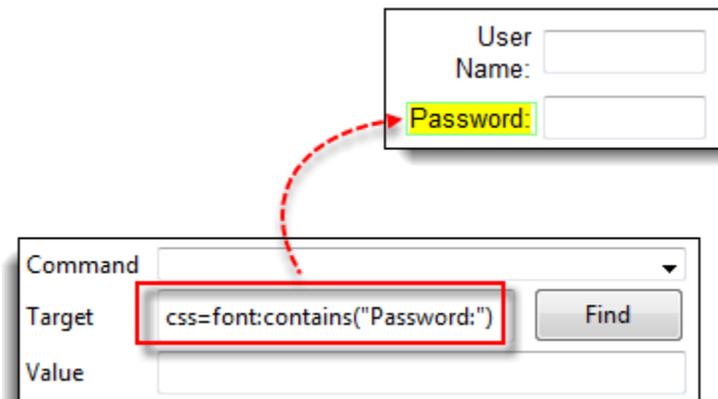
As you may have noticed, HTML labels are seldom given id, name, or class attributes. So, how do we access them? The answer is through the use of their inner texts. **Inner texts are the actual string patterns that the HTML label shows on the page.**

Syntax	Description
css=tag:contains("inner text")	<ul style="list-style-type: none"> <li>tag = the HTML tag of the element being accessed</li> <li>inner text = the inner text of the element</li> </ul>

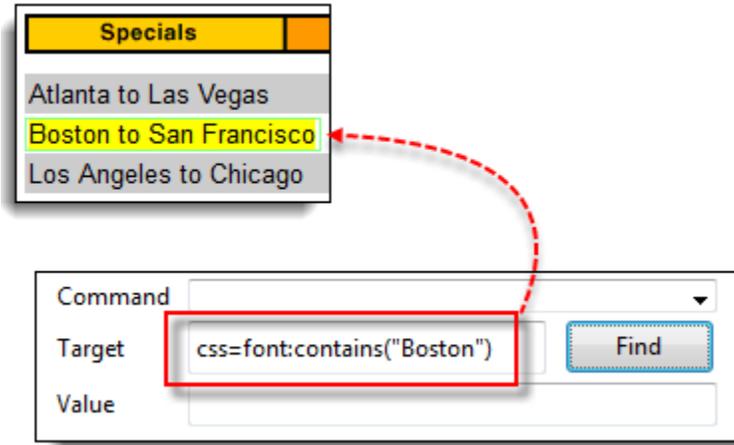
**Step 1.** Navigate to Mercury Tours' homepage (<http://newtours.demoaut.com/>) and use Firebug to investigate the "Password" label. Take note of its HTML tag (which is "font" in this case) and notice that it has no class, id, or name attributes.



**Step 2.** Type `css=font:contains("Password:")` into Selenium IDE's Target box and click Find. Selenium IDE should be able to access the Password label as shown on the image below.



**Step 3.** This time, replace the inner text with "Boston" so that your Target will now become `css=font:contains("Boston")`. Click Find. You should notice that the "Boston to San Francisco" label becomes highlighted. This shows you that Selenium IDE can access a long label even if you just indicated the first word of its inner text.



## Locating by XPath

XPath is the language used when locating XML (Extensible Markup Language) nodes. Since HTML can be thought of as an implementation of XML, we can also use XPath in locating HTML elements.

**Advantage:** It can access almost any element, even those without class, name, or id attributes.

**Disadvantage:** It is the most complicated method of identifying elements because of too many different rules and considerations.

Xpaths are two types.

- 1) **Absolute xpath or full xpath**
- 2) **Partial xpath or relative xpath**

**Absolute xpath:** Absolute xpath or full xpath starts from root level i.e from html.

Eg: `/html/body/div[2]/div/div/footer/section[3]/div/ul/li[3]/a`

**Relative xpath :** firepath will generate relative xpath starts with (`//`). It will generate based on ID. Generally, if the element has not id, then it will provide Absolute xpath from html level onwards.

Eg: `//*[@id='social-media']/ul/li[3]/a`

Absolute xpath is using single slash at the start of the xpath and relative is using double slash.

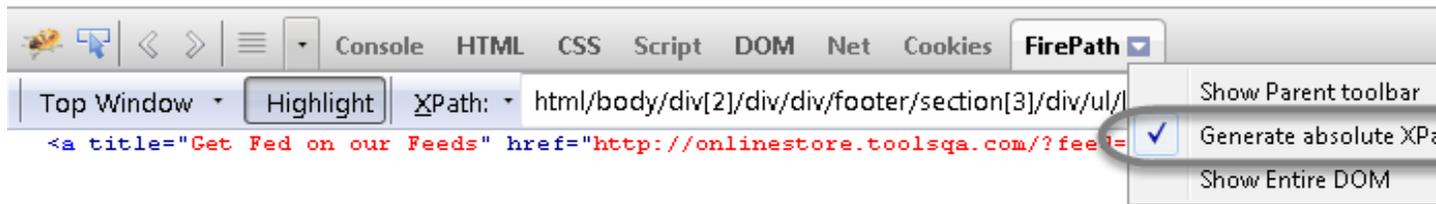
## Difference between single '/' or double '//'

A **single slash** at the start of Xpath instructs XPath engine to look for element starting from **root node**.

A **double slash** at the start of Xpath instructs XPath engine to search look for matching element **anywhere** in the XML document.

## Choosing Relative xpath using FirePath

There is an alternate way to get the relative xpath with help of the **FirePath** tool. Click on the drop down menu on the Firepath button and **Unselect** 'Generate absolute XPath'.



Now click on the same element with the Inspector, the new xpath will look like this:

## HTML Elements

```
<table>
<tbody>
<tr> <td>One </td> <td><input class="chk" name="Chk1" type="checkbox" /></td> <td id="bike">Bike </td>
<tr> <td>Two </td> <td><input class="chk" name="Chk2" type="checkbox" /></td> <td id="car">Car </td>
<tr> <td>Three</td> <td><input class="chk" name="Chk3" type="checkbox" /></td> <td id="bus_23423">Bus </td>
<tr> <td>Four</td> <td><input class="chk" name="Chk4" type="checkbox" /></td> <td id="jeep">Jeep </td>
</tbody>
</table>
```

## Above html in browser

One	<input type="checkbox"/>	Bike
Two	<input type="checkbox"/>	Car
Three	<input type="checkbox"/>	Bus

Four	<input type="checkbox"/>	Jeep
------	--------------------------	------

## Locating elements using Attributes

Attribute	Usage
id	By.xpath("//table[@id='tableId']")
id	By.xpath("//td[@id='car']")
name	By.xpath("//td[@name='Chk3']")

## Locating Rows using index

Row	As a child	As a sub-child
1	By.xpath("//table[@id='tableId']/tbody/tr[1]")	By.xpath("//table[@id='tableId']/tr[1]")
2	By.xpath("//table[@id='tableId']/tbody/tr[2]")	By.xpath("//table[@id='tableId']/tr[2]")
3	By.xpath("//table[@id='tableId']/tbody/tr[3]")	By.xpath("//table[@id='tableId']/tr[3]")
4	By.xpath("//table[@id='tableId']/tbody/tr[4]")	By.xpath("//table[@id='tableId']/tr[4]")

## Locating Rows using functions

Function	As a child
position()	By.xpath("//table[@id='tableId']/tr[position()=1]")
position()	By.xpath("//table[@id='tableId']/tr[position()=3]")
last()	By.xpath("//table[@id='tableId']/tr[last()]")
last()	By.xpath("//table[@id='tableId']/tr[last()-1]")

## Locating Rows using String functions

Function	Usage
text()	By.xpath("//table[@id='tableId']/tr/td[text()='One']")

contains()	By.xpath("//table[@id='tableId']/tr/td[contains(text(),'hre')]")
starts-with()	By.xpath("//table[@id='tableId']/tr/td[start-with(text(),'Fo')]")

## Locating Columns using Xpath Axes

Axes	Usage
child	By.xpath("//table[@id='tableId']/tr/child::td[text()='One']")
parent	By.xpath("//td[@id='car']/parent::tr")
preceding-sibling	By.xpath("//td[contains(@id,'bus')]/preceding-sibling::td/input")
following-sibling	By.xpath("//td[text()='Four']/following-sibling::td[@id='jeep']")

- **Child** : Selects all children of the current node.
- **Parent** : Selects the parent of the current node.
- **preceding-sibling**:Selects all siblings before the current node.
- **following-sibling**:Selects all siblings after the current node.

## Replace XPath with Css selector

If you know css selectors, those are much more easier (and will work fine with IE without any problem ) than xpath even though they can't replace entire xpath.

Consider below HTML tags

```
<div id="firstDiv">This div tag has static id </div>
```

```
<p id="paragraphs_dynamic_1234">This p tag has dynamic id</p>
```

```
<a onclick="doThat()" title="doThatOperation">This a tag has multiple attributes</a>
```

```
<h1 name="heading1" value="headName">This is heading</h1>
```

```
<div>
  <input type="text" id="username" name="username">
  <input type="password" id="password" name="pwd">
</div>
```

```
<h2 id="propertyUserData">This tag is for starts-with example</h2>
```

Tag	XPATH	CSS SELECTOR
Div	"//div[@id='firstDiv']"	"div[id='firstDiv']"
p	"//p[contains(@id,'paragraps_dynamic_')]"	"p[id*='paragraps_dynamic_']"
a	"//a[contains(@onclick,'doThat') and @tile='doThatOperation']"	"a[onclick*='doThat'][tile='doThatOperation']"
H1	"//h1[@name='heading1' and @value='headName']"	"h1[name='heading1'][value='headName']"
input	"//div/input[@id='username']"	"div > input[id='username']"
Input	"//h2[starts-with(@id,'propertyUser')]"	"h2[id^='propertyUser']"

- By seeing above locators we can conclude some points regarding css selectors
- Using css selector we can access particular node or immediate child of any node
- Can combine as many conditions as we want for single node.
- Can achieve starts-with, contains functionality using ^,\* symbols respectively.
- We can't traverse back using css selector.
- We can't go to the siblings also (preceding as well as following)

# DATABASE TESTING

## **Database testing in Selenium with JDBC ODBC using Oracle and MS Access**

As we already know, Selenium does not support Database Testing but partially we can do using JDBC and ODBC. We will connect Java program with database and will fetch data tables and will verify the data based on our requirement

### **Part 1- Connect with MS Access Database**

### **Part 2- Connect with Oracle Database**

Let see how to connect with Database using ODBC Driver

These are the simple steps which we will follow in our program

Step 1- First Load the driver

Step 2-Then Create a connection

Step 3- Then Create a statement

Step 4- Then Execute your SQL query

Step 5- And Store the data in Result set

Step 6- Finally verify whether data (table) is updated or not

Lets implement the same using Java and DB as MS Access

Precondition- Set the DSN (Data Source Name)

### **Step to setup DSN-**

1- Open Control Panel > Then go to Administrative tool > Then Click on ODBC

2- Go to user DSN tab and Click Add button

3- Now Select the respective database which you want to connect in this case I have selected MS Access

4- Specify the name of DSN and Click on Save button

Program 1- Database testing in Selenium using MS Access

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.ResultSet;
4 import java.sql.SQLException;
5 import java.sql.Statement;
6 import org.testng.annotations.Test;
7
8 public class TestDatabase {
9
10
11
12 @Test
13 public void TestVerifyDB(){
14
15 try {
```

```

16
17 // This will load the driver
18 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
19 System.out.println("Driver Loaded");
20
21 // Specify the path of the database
22
23 String dblocation= "C:\\Users\\Desktop\\DB\\FacebookDB1.accdb";
24
25 // This will connect with Database , getConnection taking three argument
26 // first argument Test_Oracle is the dsn which you can change as per your system,
27
28 Connection
29 con=DriverManager.getConnection("jdbc:odbc:AscentAccess;DBQ="+dblocation);
30
31 System.out.println("Connection created");
32
33 // This will create statement
34 Statement smt=con.createStatement();
35
36 System.out.println("Statement created");
37
38 // Now execute the query, here facebook is the table which I have created in DB
39
40 ResultSets= smt.executeQuery("Select * from Facebook");
41
42 System.out.println("Query Executed");
43
44 // Iterate the resultset now
45
46 while(rs.next()){
47 String uname= rs.getString("username");
48 String pass= rs.getString("password");
49 String email= rs.getString("email");
50
51 System.out.println("Uname is "+uname+" password is "+pass+" email id is "+email);
52
53 }
54 }catch (Exception e) {
55 System.out.println(e.getMessage());
56 }
57 }
58 }

```

## Oracle Database connection using JDBC.

Let see how to connect with Database using ODBC Driver

These are the simple steps to follow

Step 1- Load the driver

Step 2-Create a connection

Step 3- Create a statement

Step 4- Execute your query

Step 5- Store the data in Result set

Step 6- Verify whether data (table) is updated or not

Lets implement the same using Java and DB as Oracle

Precondition- Set the DSN (Data Source Name)

### Step to setup DSN-

1- Control Panel > Administrative tool > Click on ODBC

2- Go to user DSN and Click on Add button

3- Select the respective database which you want to connect in this case I selected Oracle 11G

4- Give the name to DSN and Save

### Database testing in Selenium using Oracle

```
1 import java.sql.Connection;
2 import java.sql.DriverManager;
3 import java.sql.ResultSet;
4 import java.sql.SQLException;
5 import java.sql.Statement;
6 import org.testng.annotations.Test;
7
```

```

8 public class TestDatabase {
9
10
11
12 @Test
13 public void TestVerifyDB(){
14
15 try {
16
17 // This will load the driver
18 Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
19 System.out.println("Driver Loaded");
20
21 // This will connect with Database , getConnection taking three argument
22 // first argument Test_Oracle is the dsn which you can change as per your system,
23 // second argument is username and third argument is password
24
25 Connection con=DriverManager.getConnection("jdbc:odbc:Test_Oracle",
26 "system","selenium");
27
28 System.out.println("Connection created");
29
30 // This will create statement
31 Statement smt=con.createStatement();
32
33 System.out.println("Statement created");
34
35 // Now execute the query, here facebook is the table which I have created in DB
36
37 ResultSets= smt.executeQuery("Select * from Facebook");
38
39 System.out.println("Query Executed");
40
41 // Iterate the resultset now
42
43 while(rs.next()){
44
45 String uname= rs.getString("username");
46 String pass= rs.getString("password");
47 String email= rs.getString("email");
48
49 System.out.println("Uname is "+uname+" password is "+pass+" email id is "+email);
50
51 }
52 }
53 catch (Exception e) {

```

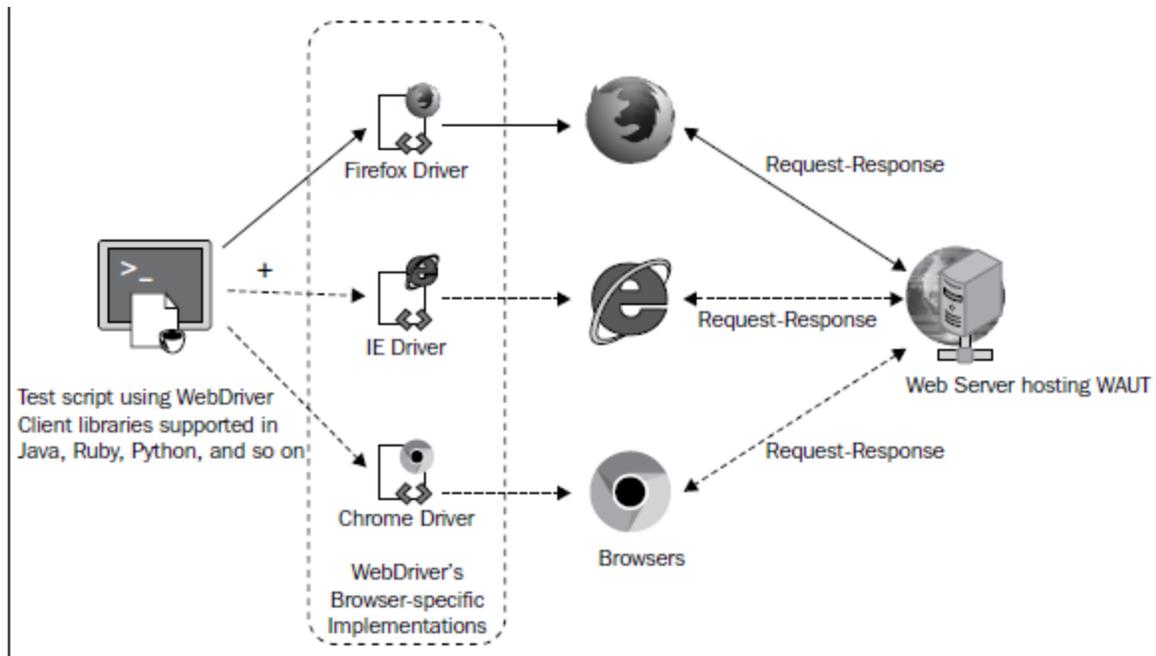
```
54 System.out.println(e.getMessage());
55 }
56
57 }
58
59 }
60
61
```

Remotedriver

## **Introducing RemoteWebDriver**

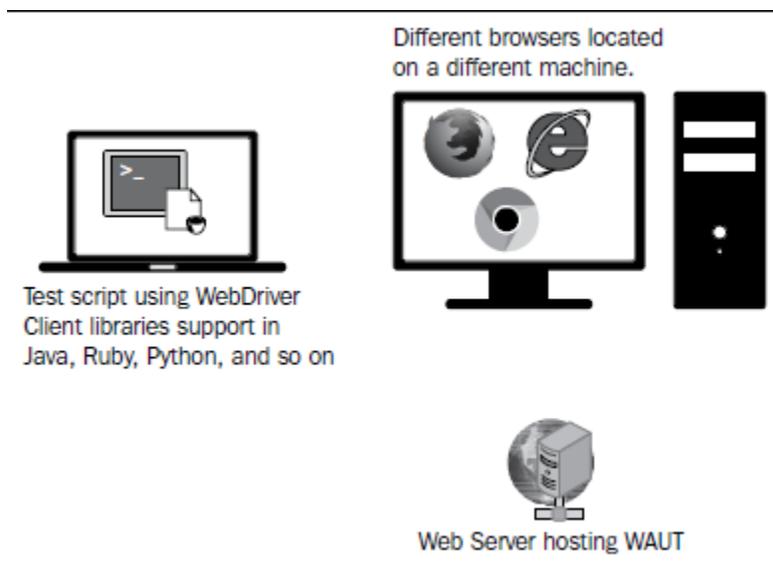
RemoteWebDriver is an implementation class of the WebDriver interface that a test script developer can use to execute their test scripts via the RemoteWebDriver server on a remote machine. There are two parts to RemoteWebDriver: a server and a client. Before we start working with them, let us rewind and see what we have been doing.

The following diagram explains what we have been doing so far.



The test script using WebDriver client libraries, Firefox Driver (or IE Driver or Chrome Driver), and Firefox browser (or IE browser or Chrome browser) are sitting on the same machine. The browser is loading the web application, which may or may not be hosted remotely;

We will discuss different scenarios of test script execution as follows:



The test script is located on a local machine, while the browsers are installed on a remote machine. In this scenario, RemoteWebDriver comes into the picture. As mentioned earlier, there

are two components associated with RemoteWebDriver: the server and the client. Let us start with the RemoteWebDriver server.

## Understanding the RemoteWebDriver server

The RemoteWebDriver server is a component that listens on a port for various requests from a RemoteWebDriver client. Once it receives the requests, it forwards them to any of the following: Firefox Driver, IE Driver, or Chrome Driver, whichever is asked.

## Downloading the server

Let us download the RemoteWebDriver server and start running it. You can download it from <https://code.google.com/p/selenium/downloads/>, but for our purposes, let us download a specific version of it as we are using WebDriver Version 2.45. The specific version can be downloaded from <https://code.google.com/p/selenium/downloads/detail?name=selenium-server-standalone-2.33.0.jar>.

This server JAR should be downloaded to the remote machine on which the browsers are located. Also, make sure the remote machine has Java runtime installed on it.

## Running the server

Open your command-line tool on the remote machine and navigate to the location to which you have downloaded the JAR file. Now, to start the RemoteWebDriver server, execute the following command:

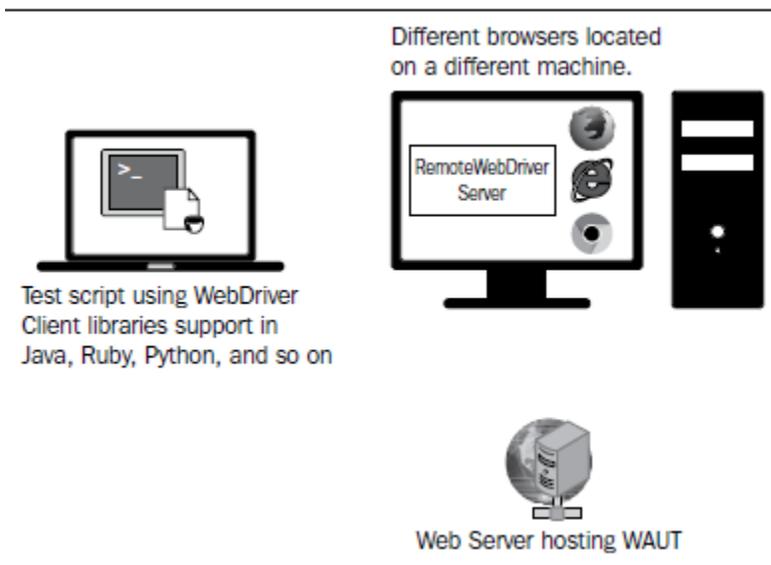
```
java -jar selenium-server-standalone-2.33.0.jar
```

The following screenshot shows what you should see in your console:

The following screenshot shows what you should see in your console:

```
Command Prompt - java -jar selenium-server-standalone-2.33.0.jar - - - - -
C:\>java -jar selenium-server-standalone-2.33.0.jar
Oct 05, 2013 2:40:55 PM org.openqa.grid.selenium.GridLauncher main
INFO: Launching a standalone server
14:41:47.994 INFO - Java: Oracle Corporation 23.21-b01
14:41:47.995 INFO - OS: Windows 8 6.2 x86
14:41:48.016 INFO - v2.33.0, with Core v2.33.0. Built from revision 4e90c97
14:41:48.687 INFO - RemoteWebDriver instances should connect to: http://127.0.0.
1:4444/wd/hub
14:41:48.688 INFO - Version Jetty/5.1.x
14:41:48.689 INFO - Started HttpContext[/selenium-server/driver,/selenium-server
/driver]
14:41:48.690 INFO - Started HttpContext[/selenium-server,/selenium-server]
14:41:48.690 INFO - Started HttpContext[/,/]
14:41:48.935 INFO - Started org.openqa.jetty.jetty.servlet.ServletHandler@123b9c
1
14:41:48.935 INFO - Started HttpContext[/wd,/wd]
14:41:48.958 INFO - Started SocketListener on 0.0.0.0:4444
14:41:48.959 INFO - Started org.openqa.jetty.jetty.Server@136bdda
```

Now, the server has started and is listening on the <remote-machine-ip>:4444 address for remote connections from the RemoteWebDriver client. Now the previously seen image (the second image in the *Introducing RemoteWebDriver* section) will appear as follows:



## Understanding the RemoteWebDriver client

When you execute your tests locally, the WebDriver client libraries talk to your Firefox Driver, IE Driver, or Chrome Driver directly. Now, when you try to execute your tests remotely, the WebDriver client libraries talk to the RemoteWebDriver server and the server talks to either the Firefox Driver, IE Driver, or Chrome Driver, whichever the WebDriver client asks for.

## Converting an existing test script to use RemoteWebDriver server

Let us take a test script that we have executed locally; that is, where the test scripts and the browser were on the same machine:

```
public class ExistingTest {
    public static void main(String... args){
        WebDriver driver = new FirefoxDriver();
    }
}
```

The preceding test script creates an instance of Firefox Driver and launches the Firefox browser. Now let us try to convert this test script to use the RemoteWebDriver server that we have started earlier. Before we do that, let us see the constructor of RemoteWebDriver, which is as follows:

```
RemoteWebDriver(java.net.URL remoteAddress,
    Capabilities desiredCapabilities)
```

The input parameters for the constructor are one of the addresses of the RemoteWebDriver server running on the remote machine and the desired capabilities your test script needs. We will see those desired capabilities shortly.

Now, let's modify the test script to use RemoteWebDriver. Replace `WebDriver driver = new FirefoxDriver();` with the following code:

```
DesiredCapabilities capabilities = new DesiredCapabilities();
RemoteWebDriver remoteWD = null;
try {
    remoteWD = new RemoteWebDriver(new
URL("http://10.172.10.1:4444/wd/hub"),capabilities);
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

We have created a RemoteWebDriver instance that tries to connect to `http://10.172.10.1:4444/wd/hub`, where the RemoteWebDriver server is running and listening for requests. Having done that, we also need to specify which browser your test case should get executed on. This can be done using the DesiredCapabilities instance. So let's ask

RemoteWebDriver to run our test scripts on the Firefox browser. The preceding code will be changed to the following code:

```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setBrowserName("firefox");
RemoteWebDriver remoteWD = null;
try {
    remoteWD = new RemoteWebDriver(new URL("http:// 10.172.10.1:4444/
wd/hub"),capabilities);
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

Now `RemoteWebDriver` will launch the Firefox browser and execute your test case on it. So the modified test case will look as follows:

```
package com.packt.webdriver.chapter7;
import java.net.MalformedURLException;
import java.net.URL;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
public class UsingRemoteWebDriver {
    public static void main(String... args){
        DesiredCapabilities capabilities = new DesiredCapabilities();
        capabilities.setBrowserName("firefox");
        RemoteWebDriver remoteWD = null;
        try {
            remoteWD = new RemoteWebDriver(new URL("http://
10.172.10.1:4444/wd/hub"),capabilities);
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
    }
}
```

## Understanding the RemoteWebDriver client

Now that we have our RemoteWebDriver server up and running, it is time for us to create the RemoteWebDriver client. Fortunately, we don't have to do anything much to create a RemoteWebDriver client. It's nothing but the language-binding client libraries that serve as a RemoteWebDriver client. The client, as it used to when executing tests locally, translates your test script requests to JSON payload and sends them across to the RemoteWebDriver server using the JSON wire protocol.

When you execute your tests locally, the WebDriver client libraries talk to your Firefox Driver, IE Driver, or Chrome Driver directly. Now, when you try to execute your tests remotely, the WebDriver client libraries talk to the RemoteWebDriver server and the server talks to either the Firefox Driver, IE Driver, or Chrome Driver, whichever the WebDriver client asks for.

## Converting an existing test script to use RemoteWebDriver server

Let us take a test script that we have executed locally; that is, where the test scripts and the browser were on the same machine:

```
public class ExistingTest {
    public static void main(String... args){
        WebDriver driver = new FirefoxDriver();
    }
}
```

The preceding test script creates an instance of Firefox Driver and launches the Firefox browser. Now let us try to convert this test script to use the RemoteWebDriver server that we have started earlier. Before we do that, let us see the constructor of RemoteWebDriver, which is as follows:

```
RemoteWebDriver(java.net.URL remoteAddress,
    Capabilities desiredCapabilities)
```

The input parameters for the constructor are one of the addresses of the RemoteWebDriver server running on the remote machine and the desired capabilities your test script needs. We will see those desired capabilities shortly.

Now, let's modify the test script to use RemoteWebDriver. Replace `WebDriver driver = new FirefoxDriver();` with the following code:

```
DesiredCapabilities capabilities = new DesiredCapabilities();
RemoteWebDriver remoteWD = null;
try {
    remoteWD = new RemoteWebDriver(new
URL("http://10.172.10.1:4444/wd/hub"),capabilities);
} catch (MalformedURLException e) {
    e.printStackTrace();
}
```

We have created a RemoteWebDriver instance that tries to connect to `http://10.172.10.1:4444/wd/hub`, where the RemoteWebDriver server is running and listening for requests. Having done that, we also need to specify which browser your test case should get executed on. This can be done using the DesiredCapabilities instance. So let's ask RemoteWebDriver to run our test scripts on the Firefox browser. The preceding code will be changed to the following code:

```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setBrowserName("firefox");
RemoteWebDriver remoteWD = null;
```

```

try {
remoteWD = new RemoteWebDriver(new URL("http:// 10.172.10.1:4444/
wd/hub"),capabilities);
} catch (MalformedURLException e) {
e.printStackTrace();
}

```

Now RemoteWebDriver will launch the Firefox browser and execute your test case on it. So the modified test case will look as follows:

```

package com.packt.webdriver.chapter7;
import java.net.MalformedURLException;
import java.net.URL;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;
public class UsingRemoteWebDriver {
public static void main(String... args){
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setBrowserName("firefox");
RemoteWebDriver remoteWD = null;
try {
remoteWD = new RemoteWebDriver(new URL("http://
10.172.10.1:4444/wd/hub"),capabilities);
} catch (MalformedURLException e) {
e.printStackTrace();
}
}
}
}

```

Now execute this test script from your local machine to establish a connection between the RemoteWebDriver client and the RemoteWebDriver server. The RemoteWebDriver server will launch the Firefox browser. The following is the output you will see in the console where the RemoteWebDriver server is running:

It says that a new session with the desired capabilities is being created, which, after being created, prints the session ID on to the console. At any point in time, you can view all of the sessions that are established with the RemoteWebDriver server by navigating to <http://10.172.10.1:4444/wd/hub>.

It will give the entire list of sessions that the RemoteWebDriver server is currently handling. The screenshot of this is as follows:

This is a very basic portal that lets the test script developer see all of the sessions created with the RemoteWebDriver server and perform some basic operations on it, such as terminating a session, taking a screenshot of a session, loading a script to a session, and seeing all of the desired capabilities of a session. The following screenshot shows all of the default desired capabilities of our current session.

You can see the popup by hovering over the **Capabilities** link, as shown in the following screenshot:

Those are the default desired capabilities that are set implicitly by the server for this session. Now, we have successfully established a connection between our test script, which is using a RemoteWebDriver client on one machine, and the RemoteWebDriver server on another machine.

# Selenium Grid

## **Introduction to Selenium Grid**

# What is Selenium Grid?

**Selenium Grid is a part of the Selenium Suite that specializes on running multiple tests across different browsers, operating systems, and machines in parallel.**

Selenium Grid has 2 versions - the older Grid 1 and the newer Grid 2. We will only focus on Grid 2 because Grid 1 is gradually being deprecated by the Selenium Team.



Selenium Grid uses a hub-node concept where you only run the test on a single machine called a **hub**, but the execution will be done by different machines called **nodes**.

## When to Use Selenium Grid?

You should use Selenium Grid when you want to do either one or both of following:

- **Run your tests against different browsers, operating systems, and machines all at the same time.** This will ensure that the application you are [testing](#) is fully compatible with a wide range of browser-O.S combinations.
- **Save time in the execution of your test suites.** If you set up Selenium Grid to run, say, 4 tests at a time, then you would be able to finish the whole suite around 4 times faster.

# Grid 1.0 Vs Grid 2.0

Following are the main differences between Selenium Grid 1 and 2.

Grid 1	Grid 2
Selenium Grid 1 has its own remote control that is different from the Selenium RC server. They are two different programs.	Selenium Grid 2 is now bundled with the Selenium Server jar file
You need to install and configure Apache Ant first before you can use Grid 1.	You do not need to install Apache Ant in Grid 2.
Can only support Selenium RC commands/scripts.	Can support both Selenium RC and WebDriver scripts.
You can only automate one browser per remote control.	One remote control can automate up to 5 browsers.

## What is a Hub and Node?

### The Hub

- The hub is the central point where you load your tests into.
- There should only be one hub in a grid.
- The hub is launched only on a single machine, say, a computer whose O.S is Windows 7 and whose browser is IE.
- The machine containing the hub is where the tests will be run, but you will see the browser being automated on the node.

### The Nodes

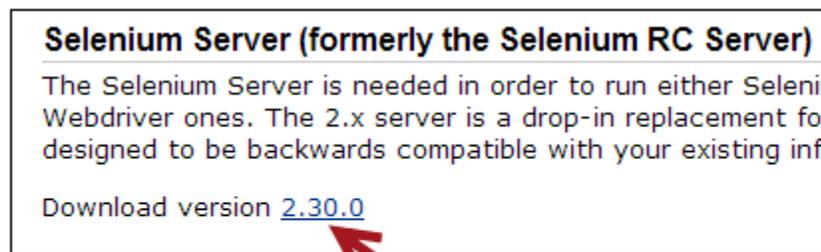
- Nodes are the Selenium instances that will execute the tests that you loaded on the hub.
- There can be one or more nodes in a grid.
- Nodes can be launched on multiple machines with different platforms and browsers.
- The machines running the nodes need not be the same platform as that of the hub.

# How to Install and Use Grid 2.0?

In this section, you will use 2 machines. The first machine will be the system that will run the hub while the other machine will run a node. For simplicity, let us call the machine where the hub runs as "Machine A" while the machine where the node runs will be "Machine B". It is also important to note their IP addresses. Let us say that Machine A has an IP address of 192.168.1.3 while Machine B has an IP of 192.168.1.4.

## Step 1

Download the Selenium Server by [here](#).



click this to start  
download

## Step 2

You can place the Selenium Server .jar file anywhere in your HardDrive. But for the purpose of this tutorial, place it on the C drive of both Machine A and Machine B. After doing this, you are now done installing Selenium Grid. The following steps will launch the hub and the node.

## Step 3

- We are now going to launch a hub. Go to Machine A. Using the command prompt, navigate to the root of Machine A's - C drive, because that is the directory where we placed the Selenium Server.
- On the command prompt, type `java -jar selenium-server-standalone-2.30.0.jar -role hub`
- The hub should successfully be launched. Your command prompt should look similar to the image below

```
C:\Users\... >cd \  
  
C:\>java -jar selenium-server-standalone-2.30.0.jar -role hub  
Feb 23, 2013 4:02:11 AM org.openqa.grid.selenium.GridLauncher main  
INFO: Launching a selenium grid server  
2013-02-23 04:02:12.065:INFO:osjs.Server:jetty-7.x.y-SNAPSHOT  
2013-02-23 04:02:12.113:INFO:osjsh.ContextHandler:started o.s.j.s.ServletContext  
Handler{/,null}  
2013-02-23 04:02:12.126:INFO:osjs.AbstractConnector:Started SocketConnector@0.0.  
0.0:4444  
Feb 23, 2013 4:02:22 AM org.openqa.grid.internal.BaseRemoteProxy <init>  
WARNING: Max instance not specified. Using default = 1 instance
```

#### Step 4

Another way to verify whether the hub is running is by using a browser. Selenium Grid, by default, uses Machine A's port 4444 for its web interface. Simply open up a browser and go to <http://localhost:4444/grid/console>



Also, you can check if Machine B can access the hub's web interface by launching a browser there and going to where "iporhostnameofmachineA" should be the IP address or the hostname of the machine where the hub is running. Since Machine A's IP address is 192.168.1.3, then on the browser on Machine B you should type <http://192.168.1.3:4444/grid/console>

#### Step 5

- Now that the hub is already set up, we are going to launch a node. Go to Machine B and launch a command prompt there.
- Navigate to the root of Drive C and type the code below. We used the IP address 192.168.1.3 because that is where the hub is running. We also used port 5566 though you may choose any free port number you desire.

```
C:\> java -jar selenium-server-standalone-2.30.0.jar -role  
webdriver -hub http://192.168.1.3:4444/grid/register -port 5566
```

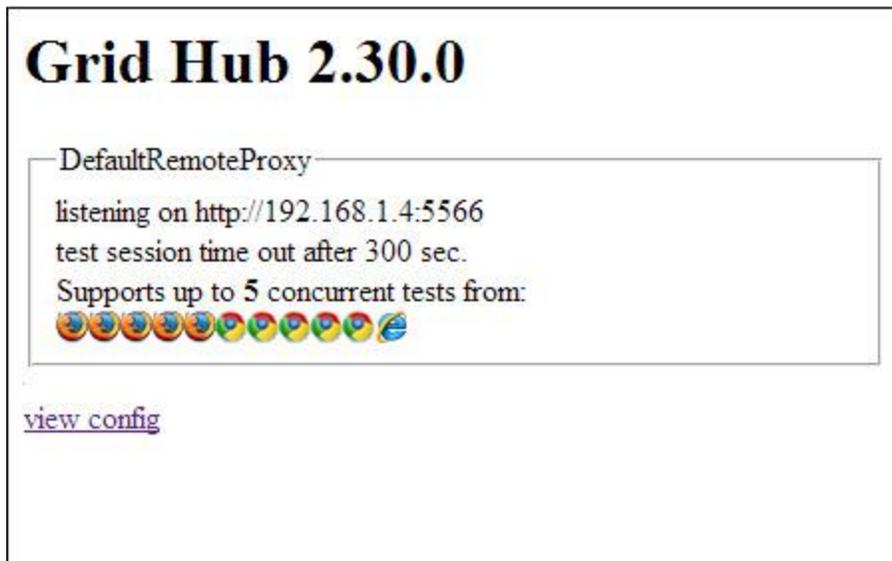
 IP address of the machine  
where the hub is running

- When you press Enter, your command prompt should be similar to the image below.

```
C:\>java -jar selenium-server-standalone-2.30.0.jar -role webdriver -hub http://  
192.168.1.3:4444/grid/register -port 5566  
Feb 23, 2013 4:34:39 PM org.openqa.grid.selenium.GridLauncher main  
INFO: Launching a selenium grid node  
16:34:40.733 INFO - Java: Oracle Corporation 23.7-b01  
16:34:40.733 INFO - OS: Windows XP 5.1 x86  
16:34:40.733 INFO - v2.30.0, with Core v2.30.0. Built from revision dc1ef9c  
16:34:40.874 INFO - RemoteWebDriver instances should connect to: http://127.0.0.  
1:5566/wd/hub  
16:34:40.874 INFO - Version Jetty/5.1.x  
16:34:40.874 INFO - Started HttpContext[/selenium-server/driver,/selenium-server  
/driver1  
16:34:40.889 INFO - Started HttpContext[/selenium-server,/selenium-server]  
16:34:40.889 INFO - Started HttpContext[/,/]  
16:34:40.889 INFO - Started org.openqa.jetty.servlet.ServletHandler@ed3512  
16:34:40.889 INFO - Started HttpContext[/wd,/wd]  
16:34:40.905 INFO - Started SocketListener on 0.0.0.0:5566  
16:34:40.905 INFO - Started org.openqa.jetty.jetty.Server@1f77497  
16:34:40.905 INFO - using the json request : {"class":"org.openqa.grid.common.Re  
gistrationRequest","capabilities":[{"platform":"XP","seleniumProtocol":"WebDrive  
r","browserName":"firefox","maxInstances":5}, {"platform":"XP","seleniumProtocol  
":"WebDriver","browserName":"chrome","maxInstances":5}, {"platform":"WINDOWS","sel  
eniumProtocol":"WebDriver","browserName":"internet explorer","maxInstances":1}],  
"configuration":{"port":5566,"register":true,"host":"192.168.1.4","proxy":"org.o  
penqa.grid.selenium.proxy.DefaultRemoteProxy","maxSession":5,"role":"webdriver",  
"hubHost":"192.168.1.3","registerCycle":5000,"hub":"http://192.168.1.3:4444/grid  
/register","hubPort":4444,"url":"http://192.168.1.4:5566","remoteHost":"http://1  
92.168.1.4:5566"}}  
16:34:40.905 INFO - starting auto register thread. Will try to register every 50  
00 ms.  
16:34:40.905 INFO - Registering the node to hub :http://192.168.1.3:4444/grid/re  
gister  
16:34:46.171 INFO - Executing: org.openqa.selenium.remote.server.handler.Status@  
1e5a771 at URL: /status  
16:34:46.186 INFO - Done: /status
```

## Step 6

Go to the Selenium Grid web interface and refresh the page. You should see something like this.



At this point, you have already configured a simple grid. You are now ready to run a test remotely on Machine B.

## Designing Test Scripts That Can Run on the Grid

To design test scripts that will run on the grid, we need to use **DesiredCapabilities** and the **RemoteWebDriver** objects.

- **DesiredCapabilities** is used to set the type of **browser** and **OS** that we will automate
- **RemoteWebDriver** is used to set which node (or machine) that our test will run against.

To use the **DesiredCapabilities** object, you must first import this package

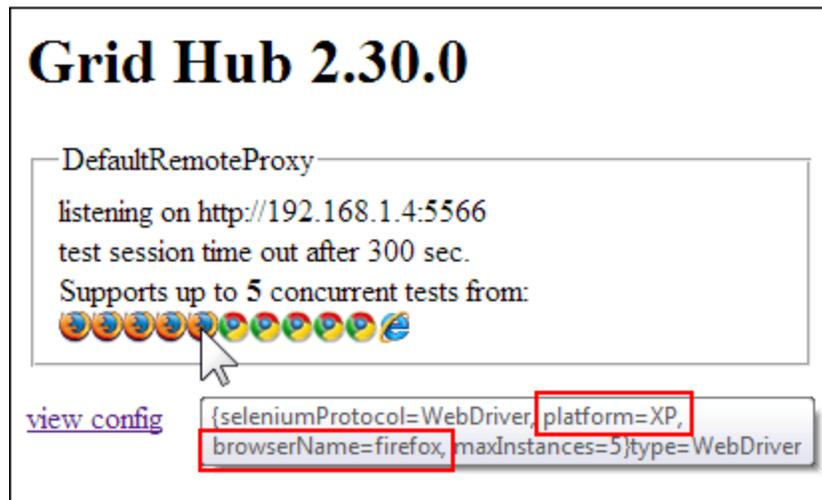
```
import org.openqa.selenium.remote.DesiredCapabilities;
```

To use the **RemoteWebDriver** object, you must import these packages.

```
import java.net.MalformedURLException;  
import java.net.URL;  
import org.openqa.selenium.remote.RemoteWebDriver;
```

## Using the DesiredCapabilities Object

Go to the Grid's web interface and hover on an image of the browser that you want to automate. Take note of the **platform** and the **browserName** shown by the tooltip.



In this case, the platform is "XP" and the browserName is "firefox".

We will use the platform and the browserName in our WebDriver as shown below (of course you need to import the necessary packages first).

```
DesiredCapabilities capability = DesiredCapabilities.firefox();
capability.setBrowserName("firefox");
capability.setPlatform(Platform.XP);
```

## Using the RemoteWebDriver Object

Import the necessary packages for RemoteWebDriver and then pass the DesiredCapabilities object that we created above as a parameter for the RemoteWebDriver object.

*We used RemoteWebDriver and not FirefoxDriver*

```
WebDriver driver;
driver = new RemoteWebDriver(
    new URL("http://192.168.1.4:5566/wd/hub"), capability);
```

*IP address and port on Machine B*

## Running a Sample Test Case on the Grid

Below is a simple WebDriverTestNG code that you can create in Eclipse on Machine A. Once you run it, automation will be performed on Machine B.

```

import org.openqa.selenium.*;
import org.openqa.selenium.remote.DesiredCapabilities;
import java.net.MalformedURLException;
import java.net.URL;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.testng.Assert;
import org.testng.annotations.*;

public class Grid_2 {
    WebDriver driver;
    String baseUrl, nodeURL;

    @BeforeTest
    public void setUp() throws MalformedURLException {
        baseUrl = "http://newtours.demoaut.com/";
        nodeURL = "http://192.168.1.4:5566/wd/hub";
        DesiredCapabilities capability = DesiredCapabilities.firefox();
        capability.setBrowserName("firefox");
        capability.setPlatform(Platform.XP);
        driver = new RemoteWebDriver(new URL(nodeURL), capability);
    }

    @AfterTest
    public void afterTest() {
        driver.quit();
    }

    @Test
    public void simpleTest() {
        driver.get(baseUrl);
        Assert.assertEquals("Welcome: Mercury Tours", driver.getTitle());
    }
}

```

The test should pass.

The screenshot displays the Selenium IDE interface. At the top, a tab labeled 'All suites' is active. Below it, a panel titled 'Default suite' shows the test results. Under the 'Info' section, the file path is 'C:\Users\...AppData\Local\Temp\test1896183272\testng-customsuite.xml', and it lists 1 test, 0 groups, and 1 time. The 'Chronological view' is selected. Under the 'Results' section, it shows '1 method, 1 passed' and 'Passed methods (hide)', with a green checkmark next to 'simpleTest'. An inset window titled 'Methods in chronological order' shows the sequence: 'setUp', 'simpleTest', and 'afterTest'.

Sikuili

## How to Integrate Sikuli with Selenium Webdriver

### Selenium with Sikuli

When I started working with Selenium I used to face so many issue related to XPath and CSS so now we are going to remove all this from the script.



We can remove all these locators from the script now using another tool called [Sikuli](#). Sikuli is another open source tool for automation

We can combine Selenium with Sikuli as well and can perform activity based on your requirement like type, click etc.

### Advantage of [Sikuli](#)–

- 1-It is Open source tool like Selenium.
- 2-We can integrate with Selenium and with other tools as well.
- 3- An Important feature of Sikuli, it can identify object-using images/screenshots it means you can capture images for the script and can perform your operations as well depends on your requirement.

**Note- in UFT this feature is known as Inside Object like Sikuli**

- 4-Flash testing.
- 5-Mobile Testing.
- 6-Windows/Desktop application testing.

We have so many other advantages as well please go through the [official docs](#).

## Sikuli Setup for Selenium

1-Download Sikuli jars from here (My Google Driver)- [Download Jars](#)

2- Add Sikuli jar into your project.

Once Setup is complete then we can start writing selenium/sikuli scripts.

We will use some Sikuli classes that we will get once we will add that jars into the project.

1-Screen – This is a class, which will focus on the screen.

2- Pattern- This is another class, which will focus on images.

## Sample Script-Integrate Sikuli with Selenium Webdriver

This script will perform following activity

- 1- Launch any browser and navigate to [google.com](http://google.com) (using Selenium)
- 2- Click on Gmail button (using Selenium)
- 3- On Gmail enter email, password and will click on login button (using Sikuli code)

```
1 package demoSikuli;
2
3 import org.openqa.selenium.WebDriver;
4
5 import org.openqa.selenium.firefox.FirefoxDriver;
6
7 import org.sikuli.script.FindFailed;
8
9 import org.sikuli.script.Pattern;
10
11 import org.sikuli.script.Screen;
12
13 public class demoSikul {
14
15 public static void main(String[] args) throws FindFailed, InterruptedException {
16
```

```

17 // We have to create Screen class object to access method
18
19 Screen screen = new Screen();
20
21 // Create object of Pattern class and specify the images path
22
23 Pattern image = new Pattern("C:\\gmail.PNG");
24
25 Pattern image1 = new Pattern("C:\\images\\uname.PNG");
26
27 Pattern image2 = new Pattern("C:\\images\\password.PNG");
28
29 Pattern image3 = new Pattern("C:\\images\\click.PNG");
30
31 WebDriver driver=new FirefoxDriver();
32
33 driver.manage().window().maximize();
34
35 driver.get("http://www.google.com");
36
37 screen.wait(image, 10);
38
39 // using screen object we can call click method which will accept image path and will perform
40 //action
41
42 // This will click on gmail image on google home page
43
44 screen.click(image);
45
46 // using screen object we can call type method which will accept image path and content
47 which //we have to type and will perform action.
48
49 // This will type on username field
50
51 screen.type(image1, "mukeshotwani@gmail.com");
52
53 //This will type of password field
54
55 screen.type(image2, "password1");
56
57 // This will click on login button
58
59 screen.click(image3);
60
61 }

```

```
}
```

Below images, I used in the script.

Output-

```
1 [error] ResourceLoaderBasic: checkLibsDir: libs dir is not on system path: C:\setup\libs
2 [action] ResourceLoaderBasic: checkLibsDir: Please wait! Trying to add it to user's path
3 [info] runcmd: reg QUERY HKCU
4 [info] runcmd: reg QUERY HKEY_CURRENT_USER\Environment /v PATH
5 [error] ResourceLoaderBasic: checkLibsDir: Logout and Login again! (Since libs folder is in user's path, but not activated)
6 [error] Terminating SikuliX after a fatal error! Sorry, but it makes no sense to continue!
7 If you do not have any idea about the error cause or solution, run again
8 with a Debug level of 3. You might paste the output to the Q&A board.
```

# Working With Excelsheets

# Working with JXL

## Working with Excel Operations Using JXL

### How to Read Excel file using Java:

Normally, to read a data in excel, first we should have access to workbook, sheet which we want to read as workbook contains multiple sheets and if you want to read a particular cell we need location of a Cell.

In this article, we will discuss how to access workbook, sheet and a Cell using Jxl library. You can also consider [Apache Poi Library](#) to perform read and write operations with excel sheets.

As we know JXL doesn't support Excel 2007 ".xlsx" file format. It only supports the old BIFF (binary) ".xls" format. Where as Apache POI supports both [Excel 2003 - xls](#) and [Excel 2007 - xlsx](#) file formats.

To start with gaining access to Workbook, we should always remember the below command:

```
String FilePath = "d://filepath.xls";  
FileInputStream fs = new FileInputStream(FilePath);  
Workbook wb = Workbook.getWorkbook(fs);
```

Or you can also directly send the file as below

```
Workbook wb = Workbook.getWorkbook(new File("samplefile.xls"));
```

Now to get the access to the particular sheet, we should use the below command:

```
Sheet sh = wb.getSheet(0); //this is to get the access to Sheet1.
```

If you want to get the access to sheet2, you should specify as below:

```
Sheet sh = wb.getSheet(1);
```

You can also get the sheet access by sheet name, you should specify as below:

```
Sheet sh = wb.getSheet("sheet1");
```

Now we will get the content in particular location.

```
String CellGetContent = sh.getCell(0,0).getContents();
```

```
System.out.println(CellGetContent);
```

We can also write it as :

```
System.out.println(sh.getCell(0,0).getContents());
```

There is an other style to get the cell contents as below:

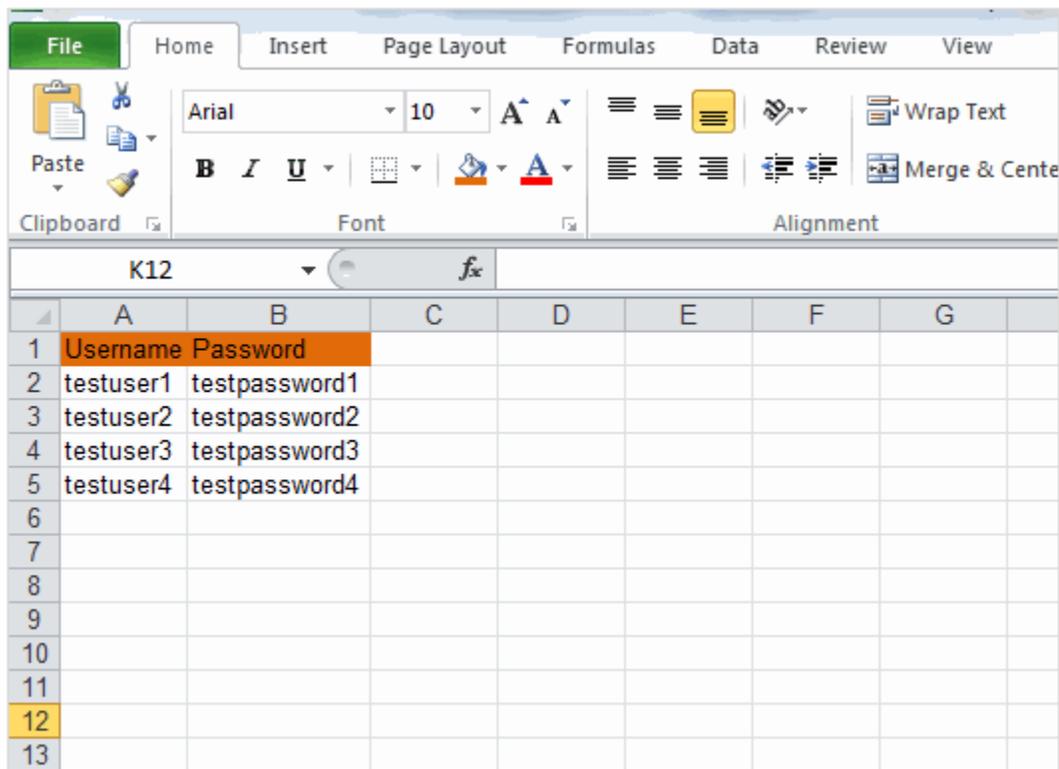
```
Cell Row0Col0 = sheet.getCell(0,0);
```

```
Cell Row1Col1 = sheet.getCell(1,1);
```

```
String FirstRowFirstColumn = Row0Col0.getContents();
```

```
String SecondRowSecondColumn = Row1Col1.getcontents();
```

The below is the input sheet for the example program:



The screenshot shows the Microsoft Excel interface with the 'Home' tab selected. The ribbon includes options for Font (Arial, size 10, bold, italic, underline, color, background color) and Alignment (wrap text, merge & center). The spreadsheet grid shows columns A through G and rows 1 through 13. The first two columns, A and B, contain test data:

	A	B	C	D	E	F	G
1	Username	Password					
2	testuser1	testpassword1					
3	testuser2	testpassword2					
4	testuser3	testpassword3					
5	testuser4	testpassword4					
6							
7							
8							
9							
10							
11							
12							
13							

Please find the below code in which we will read a data from excel sheet and print using for loop

```
Package com.pack;  
import java.io.FileInputStream;  
import java.io.IOException;  
import jxl.Sheet;  
import jxl.Workbook;  
import jxl.read.biff.BiffException;  
publicclass ReadExcelFile {  
publicvoid readExcel() throws BiffException, IOException {  
String FilePath = "D:\\sampledoc.xls";  
FileInputStream fs = new FileInputStream(FilePath);  
Workbook wb = Workbook.getWorkbook(fs); // TO get the access to the sheet  
Sheet sh = wb.getSheet("Sheet1"); // To get the number of rows present in sheet  
int totalNoOfRows = sh.getRows();  
// To get the number of columns present in sheet  
int totalNoOfCols = sh.getColumns();  
for (int row = 0; row < totalNoOfRows; row++) {  
    for (int col = 0; col < totalNoOfCols; col++) {  
System.out.print(sh.getCell(col, row).getContents() + "\t");  
    }  
System.out.println();  
    }  
    }  
publicstaticvoid main(String args[]) throws BiffException, IOException {  
ReadExcelFile DT = new ReadExcelFile();
```

```
DT.readExcel();  
}  
}
```

### How to Set Data into Excel sheet using jxl:

The below example program explains how to write / set data in spreadsheet without any formatting such as fonts etc.

In order to write anything we need to first create a writable workbook as below which creates the workbook object.

```
WritableWorkbook workbook = Workbook.createWorkbook(new File("sampletestfile.xls"));
```

We can also use the existing excel sheet and write the data into excel sheet. But before doing write operations we need to make sure to take the copy of the original file

And then perform the write operations. The below code is the sample code.

```
Workbook workbook = Workbook.getWorkbook(new File("testSampleData.xls"));
```

```
WritableWorkbook workbookCopy= Workbook.createWorkbook(new  
File("testSampleDataCopy.xls"), workbook);
```

Now access the sheet from the workbook which is copied from the original.

```
WritableSheet wSheet = workbookCopy.getSheet(0);
```

As now we are ready with the worksheet to which we need to pass the data. We need to mention the location (Column number and Row number) to write the data.

The below is the sample code to write the data into excel sheet.

```
WritableSheet wshTemp = wwbcopy.getSheet(strSheetName);
```

```
Label label= new Label(iColumnNumber, iRowNumber, strData);
```

```
wshTemp.addCell(label);
```

Please find the below working example program:

```
Import java.io.File;
```

```
import jxl.Workbook;
```

```

import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;
import jxl.write.Label;
import jxl.write.WriteException;

public class dataSheet {

    static Workbook wbook;

    static WritableWorkbook wwbcopy;

    static String ExecutedTestCasesSheet;

    static WritableSheet shSheet;

    public void readExcel() {

        try {

            wbook = Workbook.getWorkbook(new File("path\\testSampleData.xls"));

            wwbcopy = Workbook.createWorkbook(new File("path\\testSampleDataCopy.xls"), wbook);

            shSheet = wwbcopy.getSheet(0);

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

    public void setValueIntoCell(String strSheetName, int iColumnNumber, int iRowNumber, String
strData) throws WriteException {

        WritableSheet wshTemp = wwbcopy.getSheet(strSheetName);

        Label labTemp = new Label(iColumnNumber, iRowNumber, strData);

        try {

            wshTemp.addCell(labTemp);

        } catch (Exception e) {

```

```

e.printStackTrace();
}
}

public void closeFile() {
    try {
        // Closing the writable work book
        wwbCopy.write();
        wwbCopy.close();
        // Closing the original work book
        wbook.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public static void main(String[] args) throws WriteException {
    dataSheet ds = new dataSheet();
    ds.readExcel();
    ds.setValueIntoCell("sheet1", 5, 1, "PASS");
    ds.setValueIntoCell("sheet1", 5, 2, "FAIL");
    ds.setValueIntoCell("sheet1", 5, 3, "PASS");
    ds.closeFile();
}
}

```

## Write Data with formatting information

We can pass the formatting information to Excel by overloading the constructor which takes an additional object containing the cell format information.

Formatting is required when ever you want to distinguish cell content with other cell contents. Generally, when we update the test cases in excel sheet, we keep all "PASS" in green color and Failed in Red color just to highlight the content.

The below syntax and example program helps to write data into excel sheet using formatting styles

```
// Create a cell format for specified font
WritableFont cellFont= new WritableFont(WritableFont.TIMES, 12);
WritableCellFormat cellFormat = new WritableCellFormat(cellFont);
```

and In order to define the font color, we use the below statement.

```
cellFont.setColour(Colour.RED);

// Create the label, specifying content and format
Label lab1 = new Label(iColumnNumber, iRowNumber, strData, cellFormat);
sheetTemp.addCell(lab);
```

We can pass the same format to different cells. We need to share the same object that is created in the above.

```
Label lab2 = new Label(iColumnNumber, iRowNumber, strData, cellFormat);
sheetTemp.addCell(lab);
```

Please find the below example program:

```
import java.io.File;
import jxl.Workbook;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;
import jxl.format.Border;
import jxl.format.BorderLineStyle;
import jxl.format.Colour;
import jxl.write.Label;
import jxl.write.WritableCellFormat;
import jxl.write.WritableFont;
import jxl.write.WriteException;

public class dataSheet {

    static Workbook wbook;
    static WritableWorkbook wwbCopy;
```

```

static String ExecutedTestCasesSheet;
static WritableSheet shSheet;

public void readExcel()
{
    try{
        wbook = Workbook.getWorkbook(new
File("D:\\Dev\\SampleTest\\datasheets\\testSampleData.xls"));
        wwbCopy = Workbook.createWorkbook(new
File("D:\\Dev\\SampleTest\\datasheets\\testSampleDataCopy.xls"), wbook);
        shSheet = wwbCopy.getSheet(0);
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

public void setValueIntoCell(String strSheetName,int iColumnNumber,
int iRowNumber,String strData) throws WriteException
{
    WritableSheet wshTemp = wwbCopy.getSheet(strSheetName);
    WritableFont cellFont = null;
    WritableCellFormat cellFormat = null;

    if(strData.equalsIgnoreCase("PASS"))
    {
        cellFont = new WritableFont(WritableFont.TIMES, 12);
        cellFont.setColour(Colour.GREEN);
        cellFont.setBoldStyle(WritableFont.BOLD);

        cellFormat = new WritableCellFormat(cellFont);
        cellFormat.setBorder(Border.ALL,
BorderLineStyle.THIN);
    }

    else if(strData.equalsIgnoreCase("FAIL"))
    {
        cellFont = new WritableFont(WritableFont.TIMES, 12);
        cellFont.setColour(Colour.RED);
        cellFont.setBoldStyle(WritableFont.BOLD);

        cellFormat = new WritableCellFormat(cellFont);
        cellFormat.setBorder(Border.ALL,
BorderLineStyle.THIN);
    }

    else
    {
        cellFont = new WritableFont(WritableFont.TIMES, 12);
        cellFont.setColour(Colour.BLACK);

        cellFormat = new WritableCellFormat(cellFont);
        cellFormat.setBorder(Border.ALL,
BorderLineStyle.THIN);
        cellFormat.setWrap(true);
    }
}

```

```

    }

    Label labTemp = new Label(iColumnNumber, iRowNumber, strData,
cellFormat);
    try {
        wshTemp.addCell(labTemp);

        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

public void closeFile()
{
    try {
        // Closing the writable work book
        wwbCopy.write();
        wwbCopy.close();

        // Closing the original work book
        wbook.close();
    } catch (Exception e)
    {
        e.printStackTrace();
    }
}

public void main(String[] args) throws WriteException
{
    dataSheet ds = new dataSheet();
    ds.readExcel();
    ds.setValueIntoCell("Sheet1", 5, 1, "PASS");
    ds.setValueIntoCell("Sheet1", 5, 2, "FAIL");
    ds.setValueIntoCell("Sheet1", 5, 3, "N/A");
}
}

```

# working with POI APIS

## Apache POI Tutorial

Apache POI is a popular API that allows programmers to create, modify, and display MS Office files using Java programs. It is an open source library developed and distributed by Apache Software Foundation to design or modify Microsoft Office files using Java program. It contains classes and methods to decode the user input data or a file into MS Office documents.

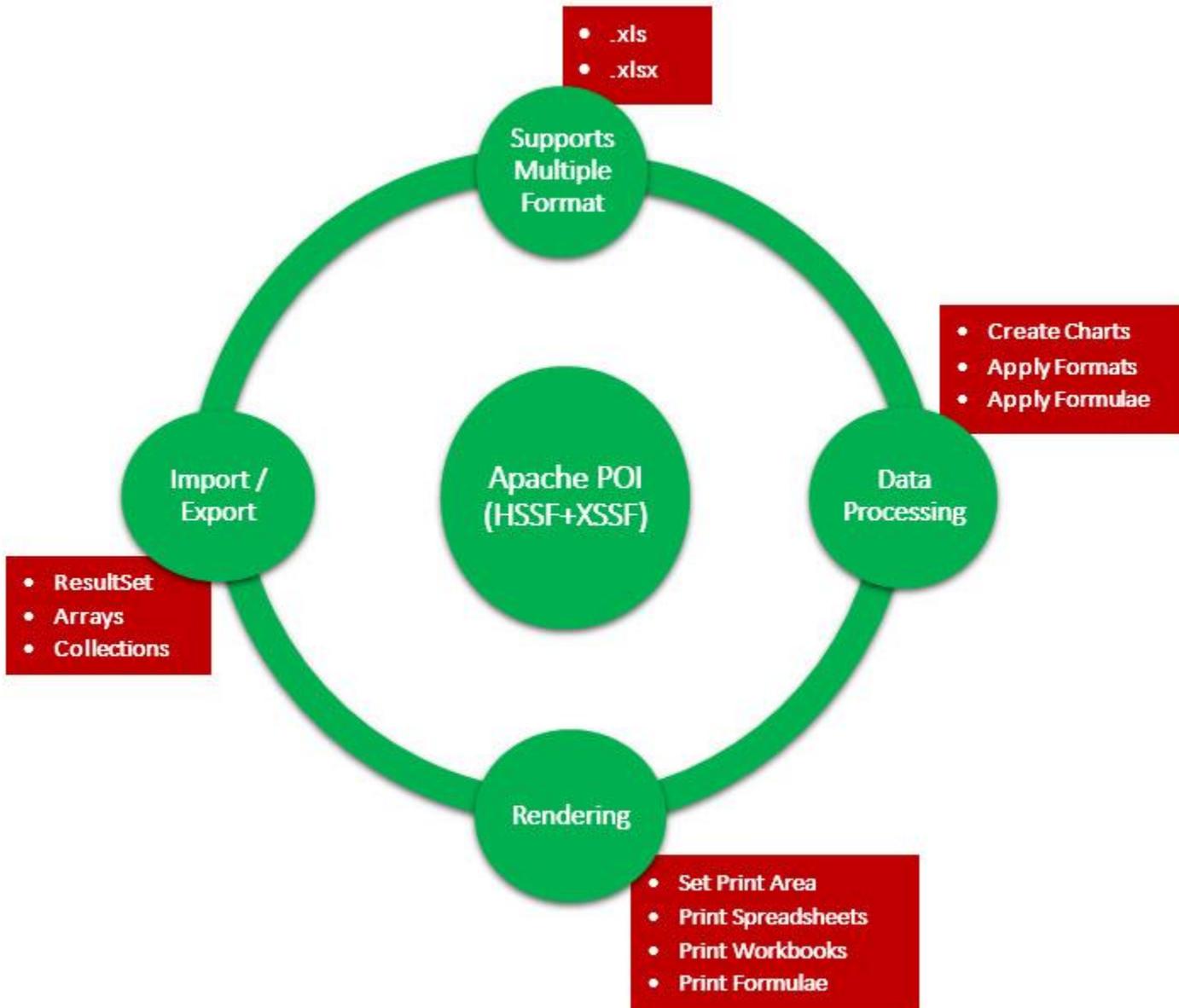
### Components of Apache POI

Apache POI contains classes and methods to work on all OLE2 Compound documents of MS Office. The list of components of this API is given below.

- **POIFS (Poor Obfuscation Implementation File System)** : This component is the basic factor of all other POI elements. It is used to read different files explicitly.
- **HSSF (Horrible Spreadsheet Format)** : It is used to read and write xls format of MS-Excel files.
- **XSSF (XML Spreadsheet Format)** : It is used for xlsx file format of MS-Excel.
- **HPSF (Horrible Property Set Format)** : It is used to extract property sets of the MS-Office files.
- **HWPF (Horrible Word Processor Format)** : It is used to read and write doc extension files of MS-Word.
- **XWPF (XML Word Processor Format)** : It is used to read and write docx extension files of MS-Word.
- **HSLF (Horrible Slide Layout Format)** : It is used for read, create, and edit PowerPoint presentations.
- **HDGF (Horrible DiaGram Format)** : It contains classes and methods for MS-Visio binary files.
- **HPBF (Horrible PuBlisher Format)** : It is used to read and write MS-Publisher files.

## Apache POI

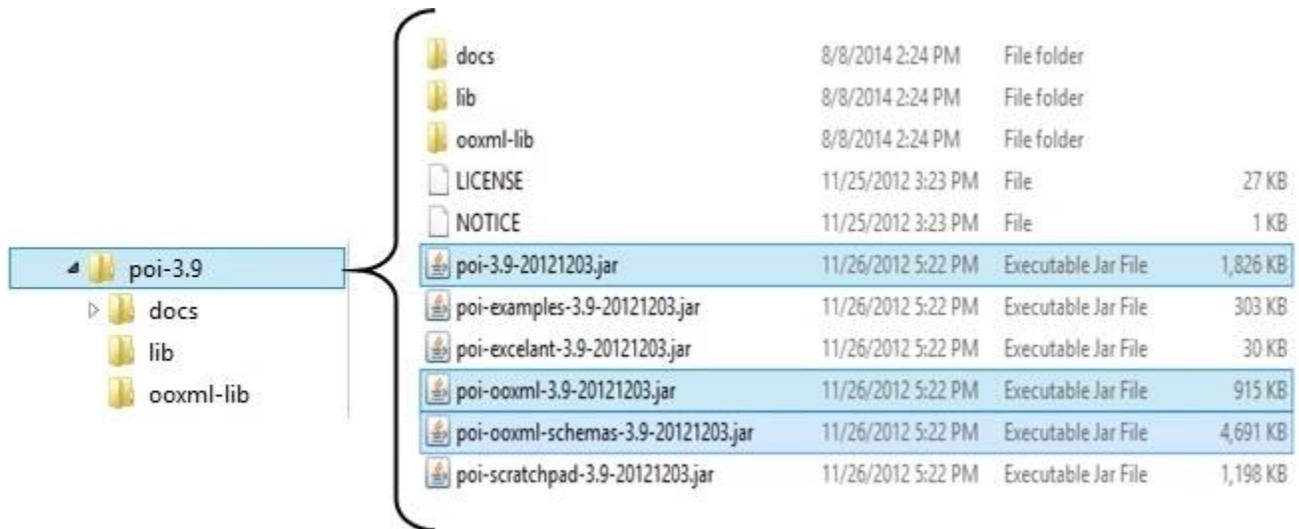
Apache POI is a 100% open source library provided by Apache Software Foundation. Most of the small and medium scale application developers depend heavily on Apache POI (HSSF + XSSF). It supports all the basic features of Excel libraries; however, rendering and text extraction are its main features.



## Install Apache POI Library

Download the latest version of Apache POI from <http://poi.apache.org/download.html> and unzip its contents to a folder from where the required libraries can be linked to your Java program. Let us assume the files are collected in a folder on C drive.

The following images show the directories and the file structure inside the downloaded folder.



## Workbook

This is the super-interface of all classes that create or maintain Excel workbooks. It belongs to the **org.apache.poi.ss.usermodel** package. The two classes that implement this interface are as follows:

- **HSSFWorkbook** : This class has methods to read and write Microsoft Excel files in .xls format. It is compatible with MS-Office versions 97–2003.
- **XSSFWorkbook** : This class has methods to read and write Microsoft Excel and OpenOffice xml files in .xls or .xlsx format. It is compatible with MS-Office versions 2007 or later.

## HSSFWorkbook

It is a high-level class under the **org.apache.poi.hssf.usermodel** package. It implements the **Workbook** interface and is used for Excel files in .xls format. Listed below are some of the methods and constructors under this class.

## Class Constructors

S.No.	Constructor and Description
1	<b>HSSFWorkbook()</b>  Creates a new HSSFWorkbook object from scratch.
2	<b>HSSFWorkbook(DirectoryNode directory, boolean preserveNodes)</b>  Creates a new HSSFWorkbook object inside a specific directory.
3	<b>HSSFWorkbook(DirectoryNode directory, POIFSFileSystem fs, boolean preserveNodes)</b>  Given a POIFSFileSystem object and a specific directory within it, it creates an HSSFWorkbook object to read a specified workbook.
4	<b>HSSFWorkbook(java.io.InputStream s)</b>  Creates a new HSSFWorkbook object using an input stream.
5	<b>HSSFWorkbook(java.io.InputStream s, boolean preserveNodes)</b>  Constructs a POI file system around your input stream.
6	<b>HSSFWorkbook(POIFSFileSystem fs)</b>  Constructs a new HSSFWorkbook object using a POIFSFileSystem object.
7	<b>HSSFWorkbook(POIFSFileSystem fs, boolean preserveNodes)</b>  Given a POIFSFileSystem object, it creates a new HSSFWorkbook object to read a specified workbook.

The frequently used parameters inside these constructors are:

- **directory** : It is the POI filesystem directory to process from.
- **fs** : It is the POI filesystem that contains the workbook stream.
- **preservenodes** : This is an optional parameter that decides whether to preserve other nodes like macros. It consumes a lot of memory as it stores all the POIFileSystem in memory (if set).

**Note** : The HSSFWorkbook class contains a number of methods; however they are compatible with xls format only. In this tutorial, the focus is on the latest version of Excel file formats. Hence, the class methods of HSSFWorkbook are not listed here. If you require these class methods, then refer <https://poi.apache.org/apidocs/org/apache/poi/hssf/usermodel/HSSFWorkbook.html>.

### XSSFWorkbook

It is a class that is used to represent both high and low level Excel file formats. It belongs to the org.apache.xssf.usermodel package and implements the Workbook interface. Listed below are the methods and constructors under this class.

### Class Constructors

S.No.	Constructor and Description
1	<p><b>XSSFWorkbook()</b></p> <p>Creates a new XSSFworkbook object from scratch.</p>
2	<p><b>XSSFWorkbook(java.io.File file)</b></p> <p>Constructs an XSSFWorkbook object from a given file.</p>
3	<p><b>XSSFWorkbook(java.io.InputStream is)</b></p> <p>Constructs an XSSFWorkbook object, by buffering the whole input stream into memory and then opening an OPCPackage object for it.</p>

4	<p><b>XSSFWorkbook(java.lang.String path)</b></p> <p>Constructs an XSSFWorkbook object given the full path of a file.</p>
---	---

### Class Methods

S.No.	Method and Description
1	<p><b>createSheet()</b></p> <p>Creates an XSSFSheet for this workbook, adds it to the sheets, and returns the high level representation.</p>
2	<p><b>createSheet(java.lang.String sheetname)</b></p> <p>Creates a new sheet for this Workbook and returns the high level representation.</p>
3	<p><b>createFont()</b></p> <p>Creates a new font and adds it to the workbook's font table.</p>
4	<p><b>createCellStyle()</b></p> <p>Creates a new XSSFCellStyle and adds it to the workbook's style table.</p>
5	<p><b>createFont()</b></p> <p>Creates a new font and adds it to the workbook's font table.</p>
6	<p><b>setPrintArea(int sheetIndex, int startColumn, int endColumn, int startRow,int endRow)</b></p> <p>Sets the print area of a given sheet as per the specified parameters.</p>

For the remaining methods of this class, refer the complete API document at:<http://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFWorkbook.html>.for the complete list of methods.

## Sheet

Sheet is an interface under the org.apache.poi.ss.usermodel package and it is a super-interface of all classes that create high or low level spreadsheets with specific names. The most common type of spreadsheet is worksheet, which is represented as a grid of cells.

## HSSFSheet

This is a class under the org.apache.poi.hssf.usermodel package. It can create excel spreadsheets and it allows to format the sheet style and sheet data.

## Class Constructors

S.No.	Constructor and Description
1	<b>HSSFSheet(HSSFWorkbook workbook)</b>  Creates new HSSFSheet called by HSSFWorkbook to create a sheet from scratch.
2	<b>HSSFSheet(HSSFWorkbook workbook, InternalSheet sheet)</b>  Creates an HSSFSheet representing the given sheet object.

## XSSFSheet

This is a class which represents high level representation of excel spreadsheet. It is under org.apache.poi.hssf.usermodel package.

## Class Constructors

S.No.	Constructor and Description
1	<b>XSSFSheet()</b>  Creates new XSSFSheet - called by XSSFWorkbook to create a sheet from scratch.

2	<p><b>XSSFSheet(PackagePart part, PackageRelationship rel)</b></p> <p>Creates an XSSFSheet representing the given package part and relationship.</p>
---	--

## Class Methods

S.No.	Methods and Description
1	<p><b>addMergedRegion(CellRangeAddress region)</b></p> <p>Adds a merged region of cells (hence those cells form one).</p>
2	<p><b>autoSizeColumn(int column)</b></p> <p>Adjusts the column width to fit the contents.</p>
3	<p><b>iterator()</b></p> <p>This method is an alias for rowIterator() to allow foreach loops</p>
4	<p><b>addHyperlink(XSSFHyperlink hyperlink)</b></p> <p>Registers a hyperlink in the collection of hyperlinks on this sheet</p>

For the remaining methods of this class, refer the complete API at: <https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFSheet.html>.

## Row

This is an interface under the org.apache.poi.ss.usermodel package. It is used for high-level representation of a row of a spreadsheet. It is a super-interface of all classes that represent rows in POI library.

## XSSFRow

This is a class under the org.apache.poi.xssf.usermodel package. It implements the Row interface, therefore it can create rows in a spreadsheet. Listed below are the methods and constructors under this class.

## Class Methods

S.No.	Description
1	<b>createCell(int columnIndex)</b>  Creates new cells within the row and returns it.
2	<b>setHeight(short height)</b>  Sets the height in short units.

For the remaining methods of this class, follow the given link <https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFRow.html>

## Cell

This is an interface under the org.apache.poi.ss.usermodel package. It is a super-interface of all classes that represent cells in the rows of a spreadsheet.

Cells can take various attributes such as blank, numeric, date, error, etc. Cells should have their own numbers (0 based) before being added to a row.

## XSSFCell

This is a class under the org.apache.poi.xssf.usermodel package. It implements the Cell interface. It is a high-level representation of cells in the rows of a spreadsheet.

## Field Summary

Listed below are some of the fields of the XSSFCell class along with their description.

Cell Type	Description
CELL_TYPE_BLANK	Represents blank cell
CELL_TYPE_BOOLEAN	Represents Boolean cell (true or false)

CELL_TYPE_ERROR	Represents error value on a cell
CELL_TYPE_FORMULA	Represents formula result on a cell
CELL_TYPE_NUMERIC	Represents numeric data on a cell
CELL_TYPE_STRING	Represents string (text) on a cell

### Class Methods

S.No.	Description
1	<p><b>setCellStyle(CellStyle style)</b></p> <p>Sets the style for the cell.</p>
2	<p><b>setCellType(int cellType)</b></p> <p>Sets the type of cells (numeric, formula, or string).</p>
3	<p><b>setCellValue(boolean value)</b></p> <p>Sets a boolean value for the cell.</p>
4	<p><b>setCellValue(java.util.Calendar value)</b></p> <p>Sets a date value for the cell.</p>
5	<p><b>setCellValue(double value)</b></p> <p>Sets a numeric value for the cell.</p>
6	<p><b>setCellValue(java.lang.String str)</b></p>

	Sets a string value for the cell.
7	<b>setHyperlink(Hyperlink hyperlink)</b>  Assigns a hyperlink to this cell.

For the remaining methods and fields of this class, visit the following link:<https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFCell.html>

## XSSFCellStyle

This is a class under the org.apache.poi.xssf.usermodel package. It will provide possible information regarding the format of the content in a cell of a spreadsheet. It also provides options for modifying that format. It implements the CellStyle interface.

## Field Summary

The following table lists a few fields that are inherited from the CellStyle interface.

Field Name	Field Description
ALIGN_CENTER	Center align the cell contents
ALIGN_CENTER_SELECTION	Center-selection horizontal alignment
ALIGN_FILL	Cell fit to the content size
ALIGN_JUSTIFY	Fit cell contents to its width
ALIGN_LEFT	Left align the cell contents
ALIGN_RIGHT	Right align the cell contents
BORDER_DASH_DOT	Cell style with dash and dot

BORDER_DOTTED	Cell style with dotted border
BORDER_DASHED	Cell style with dashed border
BORDER_THICK	Cell style with thick border
BORDER_THIN	Cell style with thin border
VERTICAL_BOTTOM	Align the cell contents vertical bottom
VERTICAL_CENTER	Align the cell contents vertical center
VERTICAL_JUSTIFY	Align and justify the cell contents vertically
VERTICAL_TOP	Top aligned vertical alignment

## Class Constructors

S.No.	Constructor and Description
1	<p><b>XSSFCellStyle(int cellXfld, int cellStyleXfld, StylesTable stylesSource, ThemesTable theme)</b></p> <p>Creates a cell style from the supplied parts</p>
2	<p><b>XSSFCellStyle(StylesTable stylesSource)</b></p> <p>Creates an empty cell Style</p>

## Class Methods

Sets the type of border for the bottom border of the cell

S.No	Method and Description
1	<b>setAlignment(short align)</b>  Sets the type of horizontal alignment for the cell
2	<b>setBorderBottom(short border)</b>
3	<b>setBorderColor(XSSFCellBorder.BorderSide side, XSSFCOLOR color)</b>  Sets the color for the selected border
4	<b>setBorderLeft(Short border)</b>  Sets the type of border for the left border of the cell
5	<b>setBorderRight(short border)</b>  Sets the type of border for the right border of the cell
6	<b>setBorderTop(short border)</b>  Sets the type of border for the top border of the cell
7	<b>setFillBackgroundColor(XSSFCOLOR color)</b>  Sets the background fill color represented as an XSSFCOLOR value.

8	<b>setFillForegroundColor(XSSFColor color)</b> Sets the foreground fill color represented as an XSSFColor value.
9	<b>setFillPattern(short fp)</b> Specifies the cell fill information for pattern and solid color cell fills.
10	<b>setFont(Font font)</b> Sets the font for this style.
11	<b>setRotation(short rotation)</b> Sets the degree of rotation for the text in the cell.
12	<b>setVerticalAlignment(short align)</b> Sets the type of vertical alignment for the cell.

For the remaining methods and fields in this class, go through the following link: <https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFCellStyle.html>

## HSSFColor

This is a class under the org.apache.poi.hssf.util package. It provides different colors as nested classes. Usually these nested classes are represented by using their own indexes. It implements the Color interface.

### Nested classes

All nested classes of this class are static and each class has its index. These nested color classes are used for cell formatting such as cell content, border, foreground, and background. Listed below are some of the nested classes.

S.No.	Class names (colors)
-------	----------------------

1	HSSFColor.AQUA
2	HSSFColor.AUTOMATIC
3	HSSFColor.BLACK
4	HSSFColor.BLUE
5	HSSFColor.BRIGHT_GREEN
6	HSSFColor.BRIGHT_GRAY
7	HSSFColor.CORAL
8	HSSFColor.DARK_BLUE
9	HSSFColor.DARK_GREEN
10	HSSFColor.SKY_BLUE
11	HSSFColor.WHITE
12	HSSFColor.YELLOW

### Class Methods

Only one method of this class is important and that is used to get the index value.

S.No.	Method and Description
1	<b>getIndex()</b>

	This method is used to get the index value of a nested class
--	--

For the remaining methods and nested classes, refer the following link:<https://poi.apache.org/apidocs/org/apache/poi/hssf/util/HSSFColor.html>.

## XSSFColor

This is a class under the org.apache.poi.xssf.usermodel package. It is used to represent color in a spreadsheet. It implements the Color interface. Listed below are some of its methods and constructors.

### Class Constructors

S.No.	Constructor and Description
1	<p><b>XSSFColor()</b></p> <p>Creates a new instance of XSSFColor.</p>
2	<p><b>XSSFColor(byte[] rgb)</b></p> <p>Creates a new instance of XSSFColor using RGB.</p>
3	<p><b>XSSFColor(java.awt.Color clr)</b></p> <p>Creates a new instance of XSSFColor using the Color class from the awt package.</p>

### Class Methods

S.No.	Method and Description
1	<p><b>setAuto(boolean auto)</b></p> <p>Sets a boolean value to indicate that the ctColor is automatic and the system ctColor is dependent.</p>

2	<p><b>setIndexed(int indexed)</b></p> <p>Sets indexed ctColor value as system ctColor.</p>
---	--

For the remaining methods, visit the following link:<https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFColor.html>.

## XSSFFont

This is a class under the org.apache.poi.xssf.usermodel package. It implements the Font interface and therefore it can handle different fonts in a workbook.

### Class Constructor

S.No.	Constructor and Description
1	<p><b>XSSFFont()</b></p> <p>Creates a new XSSFFont instance.</p>

### Class Methods

S.No.	Method and Description
1	<p><b>setBold(boolean bold)</b></p> <p>Sets a Boolean value for the 'bold' attribute.</p>
2	<p><b>setColor(short color)</b></p> <p>Sets the indexed color for the font.</p>
3	<p><b>setColor(XSSFColor color)</b></p> <p>Sets the color for the font in Standard Alpha RGB color value.</p>

4	<p><b>setFontHeight(short height)</b></p> <p>Sets the font height in points.</p>
5	<p><b>setFontName(java.lang.String name)</b></p> <p>Sets the name for the font.</p>
6	<p><b>setItalic(boolean italic)</b></p> <p>Sets a Boolean value for the 'italic' property.</p>

For the remaining methods, go through the following link: <https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFFont.html>.

## XSSFHyperlink

This is a class under the **org.apache.poi.xssf.usermodel** package. It implements the Hyperlink interface. It is used to set a hyperlink to the cell contents of a spreadsheet.

## Fields

The fields of this class are as follows. Here, fields mean the types of hyperlinks used.

Field	Description
LINK_DOCUMENT	Used to link any other document
LINK_EMAIL	Used to link email
LINK_FILE	Used to link any other file in any format
LINK_URL	Used to link a web URL

## Class Methods

S.No.	Method and Description
1	<b>setAddress(java.lang.String address)</b>  Hyperlink address.

For the remaining methods, visit the following link:<https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFHyperlink.html>

## XSSFCreationHelper

This is a class under the **org.apache.poi.xssf.usermodel** package. It implements the CreationHelper interface. It is used as a support class for formula evaluation and setting up hyperlinks.

### Class methods

S.No.	Method and Description
1	<b>createFormulaEvaluator()</b>  Creates an XSSFFormulaEvaluator instance, the object that evaluates formula cells.
2	<b>createHyperlink(int type)</b>  Creates a new XSSFHyperlink.

For the remaining methods, refer the following link:<https://poi.apache.org/apidocs/org/apache/poi/xssf/usermodel/XSSFCreationHelper.html>.

## XSSFPrintSetup

This is a class under the **org.apache.poi.xssf.usermodel** package. It implements the PrintSetup interface. It is used to set print page size, area, options, and settings.

## Class Methods

S.No.	Method and Description
1	<b>setLandscape(boolean ls)</b>  Sets a boolean value to allow or block landscape printing.
2	<b>setLeftToRight(boolean ltor)</b>  Sets whether to go left to right or top down in ordering while printing.
3	<b>setPaperSize(short size)</b>  Sets the paper size.

For the remaining methods, visit the following link:<https://poi.apache.org/apidocs/org/apache/poi/hssf/usermodel/HSSFPrintSetup.htm>

Here the term 'Workbook' means Microsoft Excel file. After completion of this chapter, you will be able to create new Workbooks and open existing Workbooks with your Java program.

## Create Blank Workbook

The following simple program is used to create a blank Microsoft Excel Workbook.

```
import java.io.*;
import org.apache.poi.xssf.usermodel.*;
public class CreateWorkBook
{
    public static void main(String[] args)throws Exception
    {
        //Create Blank workbook
        XSSFWorkbook workbook = new XSSFWorkbook();
        //Create file system using specific name
        FileOutputStream out = new FileOutputStream(
```

```

    new File("createworkbook.xlsx"));

    //write operation workbook using file out object
    workbook.write(out);

    out.close();

    System.out.println("
    createworkbook.xlsx written successfully");
}
}

```

Let us save the above Java code as `CreateWorkBook.java`, and then compile and execute it from the command prompt as follows:

```

$javac CreateWorkBook.java
$java CreateWorkBook

```

If your system environment is configured with the POI library, it will compile and execute to generate the blank Excel file named **createworkbook.xlsx** in your current directory and display the following output in the command prompt.

```

createworkbook.xlsx written successfully

```

## Open Existing Workbook

Use the following code to open an existing workbook.

```

import java.io.*;
import org.apache.poi.xssf.usermodel.*;

public class OpenWorkBook
{
    public static void main(String args[])throws Exception
    {
        File file = new File("openworkbook.xlsx");
        FileInputStream fIP = new FileInputStream(file);

        //Get the workbook instance for XLSX file
        XSSFWorkbook workbook = new XSSFWorkbook(fIP);

        if(file.isFile() && file.exists())
        {

```

```

        System.out.println(
            "openworkbook.xlsx file open successfully.");
    }
    else
    {
        System.out.println(
            "Error to open openworkbook.xlsx file.");
    }
}
}
}

```

Save the above Java code as OpenWorkBook.java, and then compile and execute it from the command prompt as follows:

```

$javac OpenWorkBook.java
$java OpenWorkBook

```

It will compile and execute to generate the following output.

```

openworkbook.xlsx file open successfully.

```

After opening a workbook, you can perform read and write operations on it.

## Create a Spreadsheet

First of all, let us create a spreadsheet using the referenced classes discussed in the earlier chapters. By following the previous chapter, create a workbook first and then we can go on and create a sheet.

The following code snippet is used to create a spreadsheet.

```

//Create Blank workbook
XSSFWorkbook workbook = new XSSFWorkbook();

//Create a blank spreadsheet
XSSFSheet spreadsheet = workbook.createSheet("Sheet Name");

```

## Rows on Spreadsheet

Spreadsheets have a grid layout. The rows and columns are identified with specific names. The columns are identified with alphabets and rows with numbers.

The following code snippet is used to create a row.

```
XSSFRow row = spreadsheet.createRow((short)1);
```

## Write into a Spreadsheet

Let us consider an example of employee data. Here the employee data is given in a tabular form.

Emp Id	Emp Name	Designation
Tp01	Gopal	Technical Manager
TP02	Manisha	Proof Reader
Tp03	Masthan	Technical Writer
Tp04	Satish	Technical Writer
Tp05	Krishna	Technical Writer

The following code is used to write the above data into a spreadsheet.

```
import java.io.File;
import java.io.FileOutputStream;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
```

```

public class Writsheet
{
    public static void main(String[] args) throws Exception
    {
        //Create blank workbook
        XSSFWorkbook workbook = new XSSFWorkbook();
        //Create a blank sheet
        XSSFSheet spreadsheet = workbook.createSheet(
            " Employee Info ");
        //Create row object
        XSSFRow row;
        //This data needs to be written (Object[])
        Map < String, Object[] > empinfo =
            new TreeMap < String, Object[] >();
        empinfo.put( "1", new Object[] {
            "EMP ID", "EMP NAME", "DESIGNATION" });
        empinfo.put( "2", new Object[] {
            "tp01", "Gopal", "Technical Manager" });
        empinfo.put( "3", new Object[] {
            "tp02", "Manisha", "Proof Reader" });
        empinfo.put( "4", new Object[] {
            "tp03", "Masthan", "Technical Writer" });
        empinfo.put( "5", new Object[] {
            "tp04", "Satish", "Technical Writer" });
        empinfo.put( "6", new Object[] {
            "tp05", "Krishna", "Technical Writer" });
        //Iterate over data and write to sheet
        Set < String > keyid = empinfo.keySet();
        int rowid = 0;
        for (String key : keyid)
        {
            row = spreadsheet.createRow(rowid++);
            Object [] objectArr = empinfo.get(key);

```

```

    int cellid = 0;
    for (Object obj : objectArr)
    {
        Cell cell = row.createCell(cellid++);
        cell.setCellValue((String)obj);
    }
}

//Write the workbook in file system
FileOutputStream out = new FileOutputStream(
new File("Writsheet.xlsx"));
workbook.write(out);
out.close();
System.out.println(
"Writsheet.xlsx written successfully" );
}
}

```

Save the above Java code as **Writsheet.java**, and then compile and run it from the command prompt as follows:

```

$javac Writsheet.java
$java Writsheet

```

It will compile and execute to generate an Excel file named **Writsheet.xlsx** in your current directory and you will get the following output in the command prompt.

```

Writsheet.xlsx written successfully

```

The Writsheet.xlsx file looks as follows.

EMP ID	EMP NAME	DESIGNATION
tp01	Gopal	Technical Manager
tp02	Manisha	Proof Reader
tp03	Masthan	Technical Writer
tp04	Satish	Technical Writer
tp05	Krishna	Technical Writer

## Read from a Spreadsheet

Let us consider the above excel file named **Writesheet.xlsx** as input. Observe the following code; it is used for reading the data from a spreadsheet.

```
import java.io.File;
import java.io.FileInputStream;
import java.util.Iterator;
import org.apache.poi.ss.usermodel.Cell;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class Readsheet
{
    static XSSFRow row;
    public static void main(String[] args) throws Exception
    {
```

```

FileInputStream fis = new FileInputStream(
new File("WriteSheet.xlsx"));
XSSFWorkbook workbook = new XSSFWorkbook(fis);
XSSFSheet spreadsheet = workbook.getSheetAt(0);
Iterator < Row > rowIterator = spreadsheet.iterator();
while (rowIterator.hasNext())
{
    row = (XSSFRow) rowIterator.next();
    Iterator < Cell > cellIterator = row.cellIterator();
    while ( cellIterator.hasNext())
    {
        Cell cell = cellIterator.next();
        switch (cell.getCellType())
        {
            case Cell.CELL_TYPE_NUMERIC:
                System.out.print(
                    cell.getNumericCellValue() + " \t\t " );
                break;
            case Cell.CELL_TYPE_STRING:
                System.out.print(
                    cell.getStringCellValue() + " \t\t " );
                break;
        }
    }
    System.out.println();
}
fis.close();
}
}

```

Let us keep the above code in **Readsheet.java** file, and then compile and run it from the command prompt as follows:

```
$javac Readsheet.java
```

```
$java Readsheat
```

If your system environment is configured with the POI library, it will compile and execute to generate the following output in the command prompt.

```
EMP ID EMP NAME DESIGNATION
tp01   Gopal   Technical Manager
tp02   Manisha  Proof Reader
tp03   Masthan  Technical Writer
tp04   Satish   Technical Writer
tp05   Krishna  Technical Writer
```

## Create a Cell

You need to create a row before creating a cell. A row is nothing but a collection of cells.

The following code snippet is used for creating a cell.

```
//create new workbook
XSSFWorkbook workbook = new XSSFWorkbook();

//create spreadsheet with a name
XSSFSheet spreadsheet = workbook.createSheet("new sheet");

//create first row on a created spreadsheet
XSSFRow row = spreadsheet.createRow(0);

//create first cell on created row
XSSFCell cell = row.createCell(0);
```

## Types of Cells

The cell type specifies whether a cell can contain strings, numeric value, or formulas. A string cell cannot hold numeric values and a numeric cell cannot hold strings. Given below are the types of cells, their values, and type syntax.

Type of cell value	Type Syntax
Blank cell value	XSSFCell.CELL_TYPE_BLANK

Boolean cell value	XSSFCell.CELL_TYPE_BOOLEAN
Error cell value	XSSFCell.CELL_TYPE_ERROR
Numeric cell value	XSSFCell.CELL_TYPE_NUMERIC
String cell value	XSSFCell.CELL_TYPE_STRING

The following code is used to create different types of cells in a spreadsheet.

```
import java.io.File;
import java.io.FileOutputStream;
import java.util.Date;
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public class TypesofCells
{
    public static void main(String[] args) throws Exception
    {
        XSSFWorkbook workbook = new XSSFWorkbook();
        XSSFSheet spreadsheet = workbook.createSheet("cell types");
        XSSFRow row = spreadsheet.createRow((short) 2);
        row.createCell(0).setCellValue("Type of Cell");
        row.createCell(1).setCellValue("cell value");
        row = spreadsheet.createRow((short) 3);
        row.createCell(0).setCellValue("set cell type BLANK");
        row.createCell(1);
        row = spreadsheet.createRow((short) 4);
        row.createCell(0).setCellValue("set cell type BOOLEAN");
        row.createCell(1).setCellValue(true);
        row = spreadsheet.createRow((short) 5);
```

```

row.createCell(0).setCellValue("set cell type ERROR");
row.createCell(1).setCellValue(XSSFCell.CELL_TYPE_ERROR );
row = spreadsheet.createRow((short) 6);
row.createCell(0).setCellValue("set cell type date");
row.createCell(1).setCellValue(new Date());
row = spreadsheet.createRow((short) 7);
row.createCell(0).setCellValue("set cell type numeric" );
row.createCell(1).setCellValue(20 );
row = spreadsheet.createRow((short) 8);
row.createCell(0).setCellValue("set cell type string");
row.createCell(1).setCellValue("A String");
FileOutputStream out = new FileOutputStream(
new File("typesofcells.xlsx"));
workbook.write(out);
out.close();
System.out.println(
"typesofcells.xlsx written successfully");
}
}

```

Save the above code in a file named **TypesofCells.java**, compile and execute it from the command prompt as follows.

```

$javac TypesofCells.java
$java TypesofCells

```

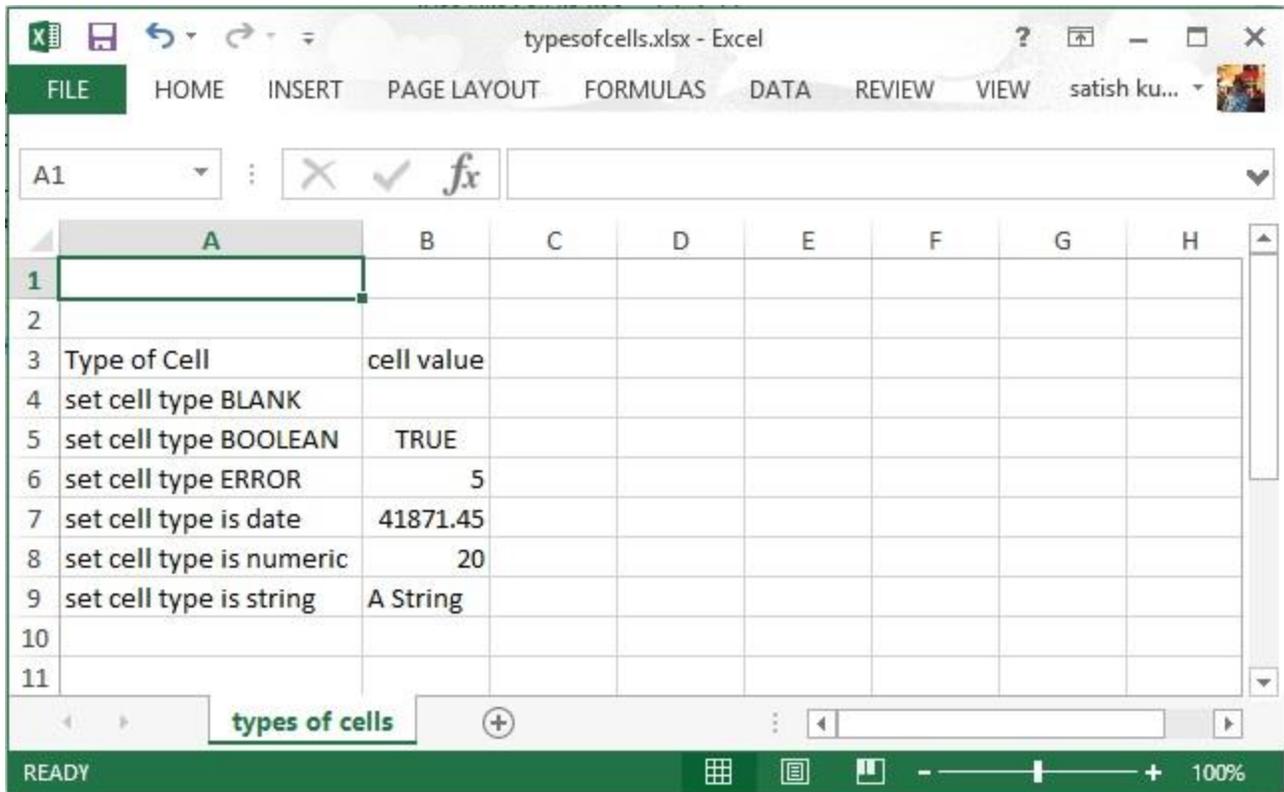
If your system is configured with the POI library, then it will compile and execute to generate an Excel file named **typesofcells.xlsx** in your current directory and display the following output.

```

typesofcells.xlsx written successfully

```

The **typesofcells.xlsx** file looks as follows.



## Cell Styles

Here you can learn how to do cell formatting and apply different styles such as merging adjacent cells, adding borders, setting cell alignment and filling with colors.

The following code is used to apply different styles to cells using Java programming.

```
import java.io.File;
import java.io.FileOutputStream;
import org.apache.poi.hssf.util.HSSFColor;
import org.apache.poi.ss.usermodel.IndexedColors;
import org.apache.poi.ss.util.CellRangeAddress;
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFCellStyle;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public class CellStyle
{
```

```

public static void main(String[] args) throws Exception
{
    XSSFWorkbook workbook = new XSSFWorkbook();
    XSSFSheet spreadsheet = workbook.createSheet("cellstyle");
    XSSFRow row = spreadsheet.createRow((short) 1);
    row.setHeight((short) 800);
    XSSFCell cell = (XSSFCell) row.createCell((short) 1);
    cell.setCellValue("test of merging");

    //MEARGING CELLS

    //this statement for merging cells
    spreadsheet.addMergedRegion(new CellRangeAddress(
        1, //first row (0-based)
        1, //last row (0-based)
        1, //first column (0-based)
        4 //last column (0-based)
    ));

    //CELL Alignment
    row = spreadsheet.createRow(5);
    cell = (XSSFCell) row.createCell(0);
    row.setHeight((short) 800);

    // Top Left alignment
    XSSFCellStyle style1 = workbook.createCellStyle();
    spreadsheet.setColumnWidth(0, 8000);
    style1.setAlignment(XSSFCellStyle.ALIGN_LEFT);
    style1.setVerticalAlignment(XSSFCellStyle.VERTICAL_TOP);
    cell.setCellValue("Top Left");
    cell.setCellStyle(style1);
    row = spreadsheet.createRow(6);
    cell = (XSSFCell) row.createCell(1);
    row.setHeight((short) 800);

    // Center Align Cell Contents
    XSSFCellStyle style2 = workbook.createCellStyle();
    style2.setAlignment(XSSFCellStyle.ALIGN_CENTER);

```

```

style2.setVerticalAlignment(
XSSFCellStyle.VERTICAL_CENTER);
cell.setCellValue("Center Aligned");
cell.setCellStyle(style2);
row = spreadsheet.createRow(7);
cell = (XSSFCell) row.createCell(2);
row.setHeight((short) 800);
// Bottom Right alignment
XSSFCellStyle style3 = workbook.createCellStyle();
style3.setAlignment(XSSFCellStyle.ALIGN_RIGHT);
style3.setVerticalAlignment(
XSSFCellStyle.VERTICAL_BOTTOM);
cell.setCellValue("Bottom Right");
cell.setCellStyle(style3);
row = spreadsheet.createRow(8);
cell = (XSSFCell) row.createCell(3);
// Justified Alignment
XSSFCellStyle style4 = workbook.createCellStyle();
style4.setAlignment(XSSFCellStyle.ALIGN_JUSTIFY);
style4.setVerticalAlignment(
XSSFCellStyle.VERTICAL_JUSTIFY);
cell.setCellValue("Contents are Justified in Alignment");
cell.setCellStyle(style4);
//CELL BORDER
row = spreadsheet.createRow((short) 10);
row.setHeight((short) 800);
cell = (XSSFCell) row.createCell((short) 1);
cell.setCellValue("BORDER");
XSSFCellStyle style5 = workbook.createCellStyle();
style5.setBorderBottom(XSSFCellStyle.BORDER_THICK);
style5.setBottomBorderColor(
IndexedColors.BLUE.getIndex());
style5.setBorderLeft(XSSFCellStyle.BORDER_DOUBLE);

```

```

style5.setLeftBorderColor(
    IndexedColors.GREEN.getIndex());
style5.setBorderRight(XSSFCellStyle.BORDER_HAIR);
style5.setRightBorderColor(
    IndexedColors.RED.getIndex());
style5.setBorderTop(XSSFCellStyle.BIG_SPOTS);
style5.setTopBorderColor(
    IndexedColors.CORAL.getIndex());
cell.setCellStyle(style5);

//Fill Colors
//background color
row = spreadsheet.createRow((short) 10 );
cell = (XSSFCell) row.createCell((short) 1);
XSSFCellStyle style6 = workbook.createCellStyle();
style6.setFillBackgroundColor(
    HSSFColor.LEMON_CHIFFON.index );
style6.setFillPattern(XSSFCellStyle.LESS_DOTS);
style6.setAlignment(XSSFCellStyle.ALIGN_FILL);
spreadsheet.setColumnWidth(1,8000);
cell.setCellValue("FILL BACKGROUND/FILL PATTERN");
cell.setCellStyle(style6);

//Foreground color
row = spreadsheet.createRow((short) 12);
cell = (XSSFCell) row.createCell((short) 1);
XSSFCellStyle style7=workbook.createCellStyle();
style7.setFillForegroundColor(HSSFColor.BLUE.index);
style7.setFillPattern( XSSFCellStyle.LESS_DOTS);
style7.setAlignment(XSSFCellStyle.ALIGN_FILL);
cell.setCellValue("FILL FOREGROUND/FILL PATTERN");
cell.setCellStyle(style7);

FileOutputStream out = new FileOutputStream(
    new File("cellstyle.xlsx"));
workbook.write(out);

```

```
    out.close();  
    System.out.println("cellstyle.xlsx written successfully");  
}  
}
```

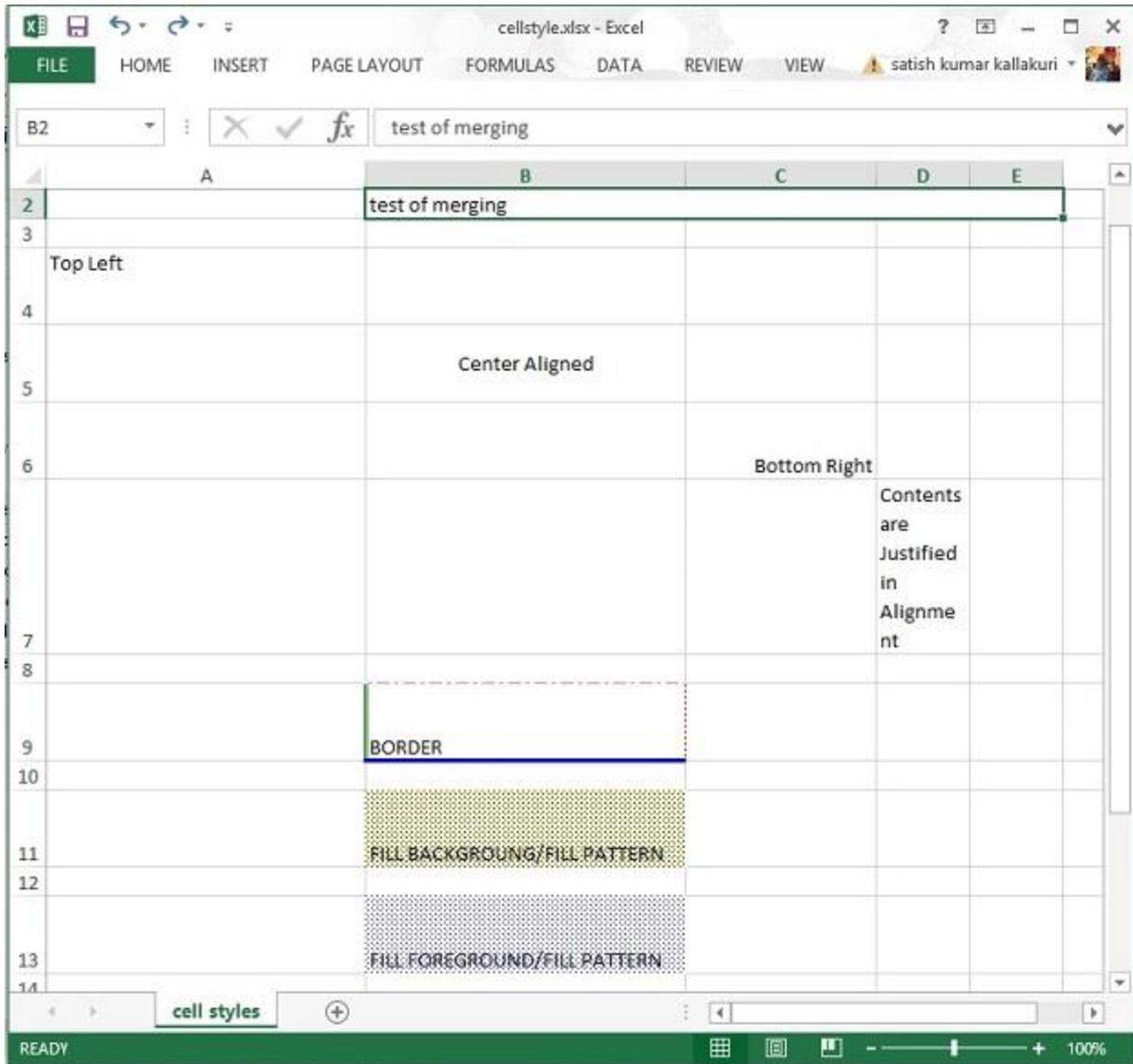
Save the above code in a file named **CellStyle.java**, compile and execute it from the command prompt as follows.

```
$javac CellStyle.java  
$java CellStyle
```

It will generate an Excel file named **cellstyle.xlsx** in your current directory and display the following output.

```
cellstyle.xlsx written successfully
```

The cellstyle.xlsx file looks as follows.



## Fonts and Font Styles

The following code is used to apply a particular font and style to the contents of a cell.

```
import java.io.File;
import java.io.FileOutputStream;
import org.apache.poi.hssf.util.HSSFColor;
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFCellStyle;
import org.apache.poi.xssf.usermodel.XSSFFont;
```

```

import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public class FontStyle
{
    public static void main(String[] args)throws Exception
    {
        XSSFWorkbook workbook = new XSSFWorkbook();
        XSSFSheet spreadsheet = workbook.createSheet("Fontstyle");
        XSSFRow row = spreadsheet.createRow(2);
        //Create a new font and alter it.
        XSSFFont font = workbook.createFont();
        font.setFontHeightInPoints((short) 30);
        font.setFontName("IMPACT");
        font.setItalic(true);
        font.setColor(HSSFColor.BRIGHT_GREEN.index);
        //Set font into style
        XSSFCellStyle style = workbook.createCellStyle();
        style.setFont(font);
        // Create a cell with a value and set style to it.
        XSSFCell cell = row.createCell(1);
        cell.setCellValue("Font Style");
        cell.setCellStyle(style);
        FileOutputStream out = new FileOutputStream(
            new File("fontstyle.xlsx"));
        workbook.write(out);
        out.close();
        System.out.println(
            "fontstyle.xlsx written successfully");
    }
}

```

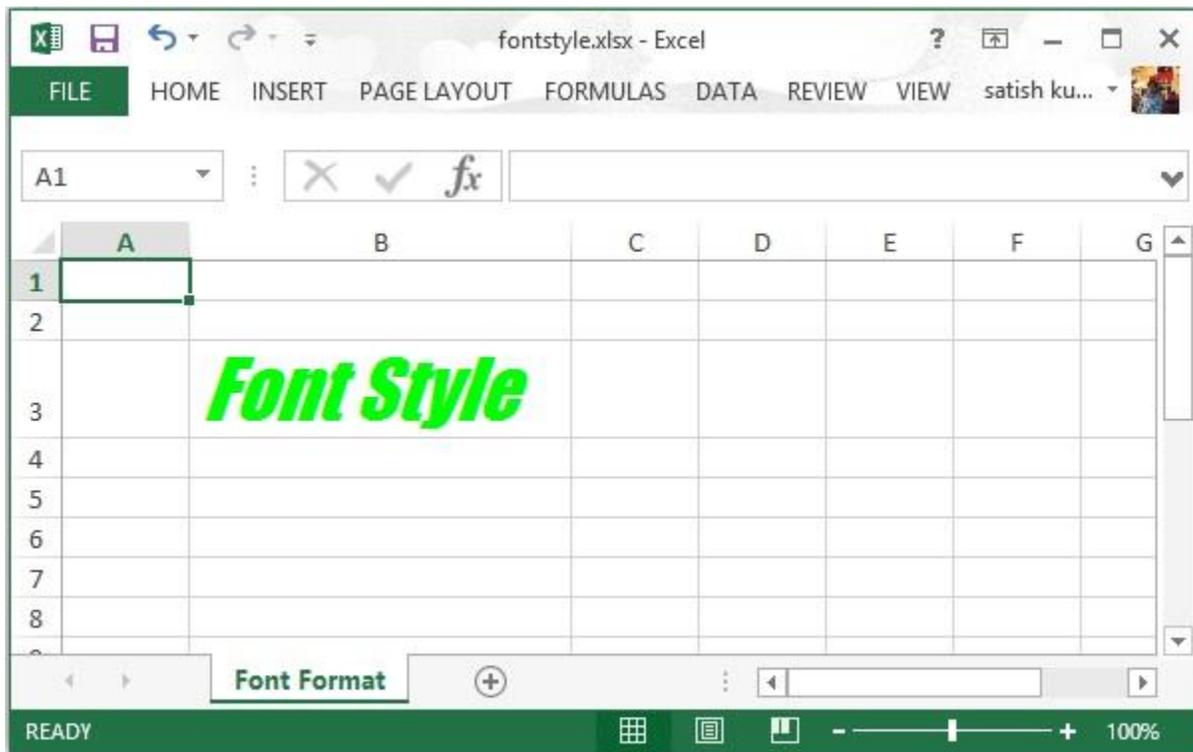
Let us save the above code in a file named **FontStyle.java**. Compile and execute it from the command prompt as follows.

```
$javac FontStyle.java
$java FontStyle
```

It generates an Excel file named **fontstyle.xlsx** in your current directory and display the following output on the command prompt.

```
fontstyle.xlsx written successfully
```

The **fontstyle.xlsx** file looks as follows.



## Text Direction

Here you can learn how to set the text direction in different angles. Usually cell contents are displayed horizontally, from left to right, and at 00 angle; however you can use the following code to rotate the text direction, if required.

```
import java.io.File;
import java.io.FileOutputStream;
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFCellStyle;
import org.apache.poi.xssf.usermodel.XSSFRow;
```

```

import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public class TextDirection
{
    public static void main(String[] args)throws Exception
    {
        XSSFWorkbook workbook = new XSSFWorkbook();
        XSSFSheet spreadsheet = workbook.createSheet(
            "Text direction");
        XSSFRow row = spreadsheet.createRow(2);
        XSSFCellStyle myStyle = workbook.createCellStyle();
        myStyle.setRotation((short) 0);
        XSSFCell cell = row.createCell(1);
        cell.setCellValue("0D angle");
        cell.setCellStyle(myStyle);
        //30 degrees
        myStyle=workbook.createCellStyle();
        myStyle.setRotation((short) 30);
        cell = row.createCell(3);
        cell.setCellValue("30D angle");
        cell.setCellStyle(myStyle);
        //90 degrees
        myStyle=workbook.createCellStyle();
        myStyle.setRotation((short) 90);
        cell = row.createCell(5);
        cell.setCellValue("90D angle");
        cell.setCellStyle(myStyle);
        //120 degrees
        myStyle=workbook.createCellStyle();
        myStyle.setRotation((short) 120);
        cell = row.createCell(7);
        cell.setCellValue("120D angle");
        cell.setCellStyle(myStyle);
    }
}

```

```

//270 degrees
myStyle = workbook.createCellStyle();
myStyle.setRotation((short) 270);
cell = row.createCell(9);
cell.setCellValue("270D angle");
cell.setCellStyle(myStyle);

//360 degrees
myStyle=workbook.createCellStyle();
myStyle.setRotation((short) 360);
cell = row.createCell(12);
cell.setCellValue("360D angle");
cell.setCellStyle(myStyle);

FileOutputStream out = new FileOutputStream(
new File("textdirection.xlsx"));
workbook.write(out);
out.close();
System.out.println(
"textdirection.xlsx written successfully");
}
}

```

Keep the above code in **TextDirectin.java** file, then compile and execute it from the command prompt as follows.

```

$javac TextDirection.java
$java TextDirection

```

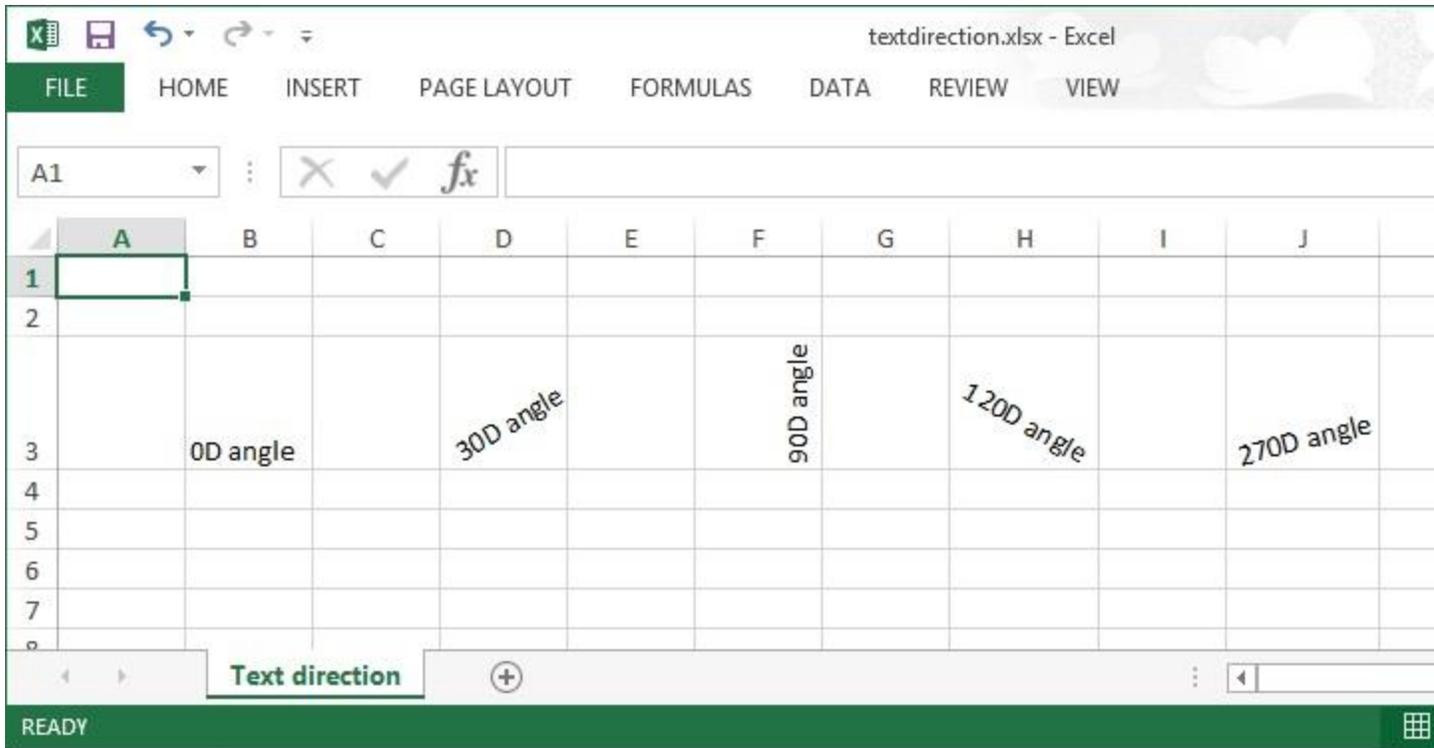
It will compile and execute to generate an Excel file named **textdirection.xlsx** in your current directory and display the following output on the command prompt.

```

textdirection.xlsx written successfully

```

The **textdirection.xlsx** file looks as follows.



## Write into Database

Let us assume the following employee data table called **emp\_tbl** is to be retrieved from the MySQL database **test**.

EMP ID	EMP NAME	DEG	SALARY	DEPT
1201	Gopal	Technical Manager	45000	IT
1202	Manisha	Proof reader	45000	Testing
1203	Masthanvali	Technical Writer	45000	IT
1204	Kiran	Hr Admin	40000	HR
1205	Kranthi	Op Admin	30000	

Use the following code to retrieve data from a database and insert the same into a spreadsheet.

```
import java.io.File;
import java.io.FileOutputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import org.apache.poi.xssf.usermodel.XSSFCell;
import org.apache.poi.xssf.usermodel.XSSFRow;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ExcelDatabase
{
    public static void main(String[] args) throws Exception
    {
        Class.forName("com.mysql.jdbc.Driver");
        Connection connect = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/test" ,
            "root" ,
            "root"
        );
        Statement statement = connect.createStatement();
        ResultSet resultSet = statement
            .executeQuery("select * from emp_tbl");
        XSSFWorkbook workbook = new XSSFWorkbook();
        XSSFSheet spreadsheet = workbook
            .createSheet("employee db");
        XSSFRow row=spreadsheet.createRow(1);
        XSSFCell cell;
        cell=row.createCell(1);
        cell.setCellValue("EMP ID");
        cell=row.createCell(2);
```

```

cell.setCellValue("EMP NAME");
cell=row.createCell(3);
cell.setCellValue("DEG");
cell=row.createCell(4);
cell.setCellValue("SALARY");
cell=row.createCell(5);
cell.setCellValue("DEPT");
int i=2;
while(resultSet.next())
{
    row=spreadsheet.createRow(i);
    cell=row.createCell(1);
    cell.setCellValue(resultSet.getInt("eid"));
    cell=row.createCell(2);
    cell.setCellValue(resultSet.getString("ename"));
    cell=row.createCell(3);
    cell.setCellValue(resultSet.getString("deg"));
    cell=row.createCell(4);
    cell.setCellValue(resultSet.getString("salary"));
    cell=row.createCell(5);
    cell.setCellValue(resultSet.getString("dept"));
    i++;
}
FileOutputStream out = new FileOutputStream(
new File("exceldatabase.xlsx"));
workbook.write(out);
out.close();
System.out.println(
"exceldatabase.xlsx written successfully");
}
}

```

Let us save the above code as **ExcelDatabase.java**. Compile and execute it from the command prompt as follows.

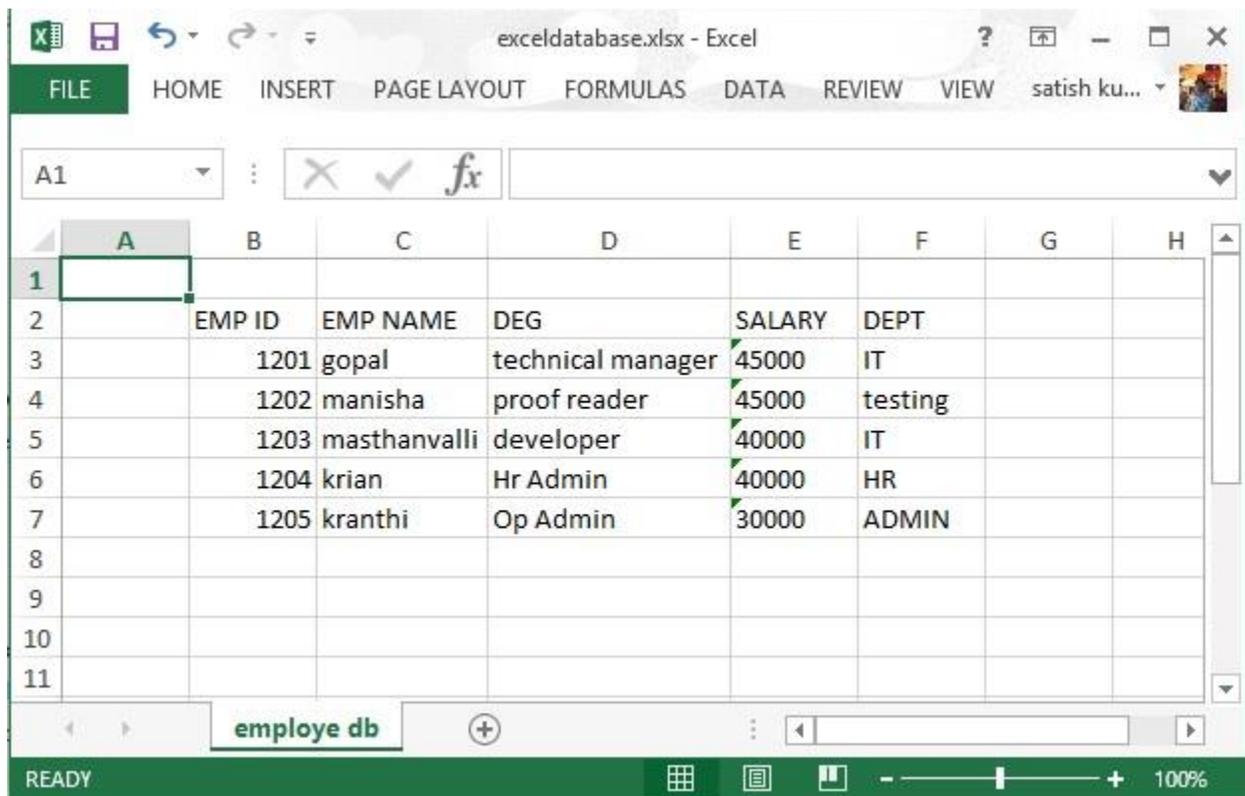
```
$javac ExcelDatabase.java
```

```
$java ExcelDatabase
```

It will generate an Excel file named **exceldatabase.xlsx** in your current directory and display the following output on the command prompt.

```
exceldatabase.xlsx written successfully
```

The **exceldatabase.xlsx** file looks as follows.



The screenshot shows an Excel spreadsheet titled "exceldatabase.xlsx" with a single worksheet named "employe db". The data is organized into a table with the following columns: EMP ID, EMP NAME, DEG, SALARY, and DEPT. The rows contain employee records for five individuals.

	EMP ID	EMP NAME	DEG	SALARY	DEPT
1					
2	1201	gopal	technical manager	45000	IT
3	1202	manisha	proof reader	45000	testing
4	1203	masthanvalli	developer	40000	IT
5	1204	krian	Hr Admin	40000	HR
6	1205	kranthi	Op Admin	30000	ADMIN
7					
8					
9					
10					
11					

# Working With Unit Frameworks

# Working With TestNG

**TestNG**

TestNG is a unit test framework designed for testing needs (developers / Test Engineers). It is inspired from JUnit by adding new functionalities which made TestNG more powerful than other unit test frameworks.

We can use TestNG for the below reasons:

1. To Run test cases in the order which we provide
2. We can run classes without using Main method.
2. To generate reports
3. To read the data from Excel file.

In Eclipse, it is very easy to Install TestNG Software, In the latest version, you will find an option to install software / Market place. Simple search for TestNG which will display a link to install. Click on Install. Installation will get completed in a short time.

### **Some of the TestNG feature as listed below**

- 1- Annotation- It supports multiple annotations at various levels so we will discuss in separate post.
- 2-Support for data-driven testing (with @DataProvider)
- 3-Support for parameters.
- 4- Generate automatic reports
- 5- We can run failed test case only using testng.xml no need to run full test suite in case of failure
- 6- Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).
- 7- Default JDK functions for runtime and logging (no dependencies).
- 8- Dependent methods for application server testing.

The below are the different annotations available in TestNG

**@BeforeSuite:** This annotation method will execute before all tests in this suite

**@AfterSuite:** This annotation method will execute after all tests in this suite

**@BeforeTest:** This annotation method will execute before any test method belonging to the classes inside the Test tag is executed.

**@AfterTest:** This annotation method will execute after all the test methods belonging to the classes inside the Test tag have executed.

**@BeforeGroups:** The list of groups that this configuration method will execute before. This method is guaranteed to execute shortly before the first test method that belongs to any of these groups is invoked.

**@AfterGroups:** The list of groups that this configuration method will execute after. This method is guaranteed to execute shortly after the last test method that belongs to any of these groups is invoked.

**BeforeClass:** This annotation method will execute before the first test method in the current class is invoked.

**@AfterClass:** This annotation method will execute after all the test methods in the current class have been executed.

**@BeforeMethod:** This annotation method will execute before each test method.

**@AfterMethod:**The annotated method will be executed after each test method.

## **How to Install TestNG step by step**

We will follow the below steps to install TestNG in Eclipse IDE .

NOTE: If your using eclipse latest version, please use Eclipse Marketplace option which is very simple, explained at the bottom.

### **There are two ways in installing TestNG in Eclipse**

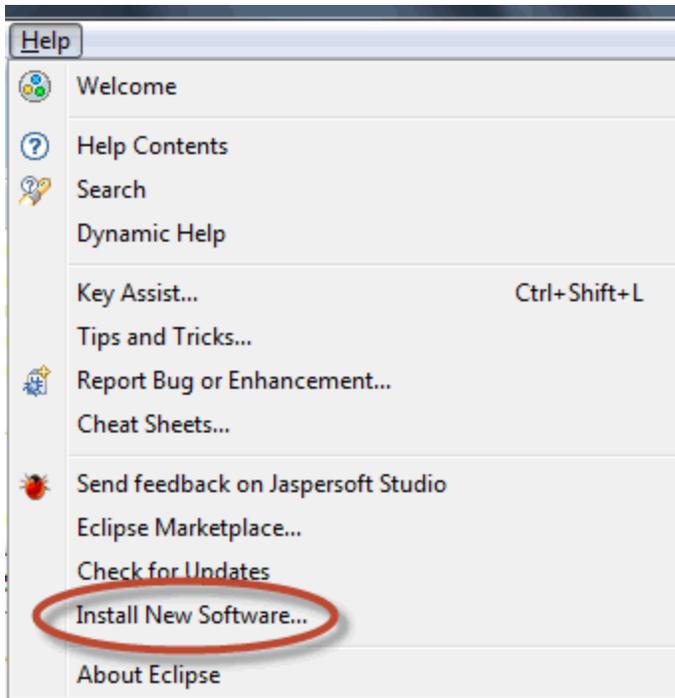
First Way on installing Eclipse is using "Install new software" option.

Second way is using "Eclipse Market Place". - This option will be available in new versions of eclipse.

#### *Steps to Install Eclipse using Install new Software:*

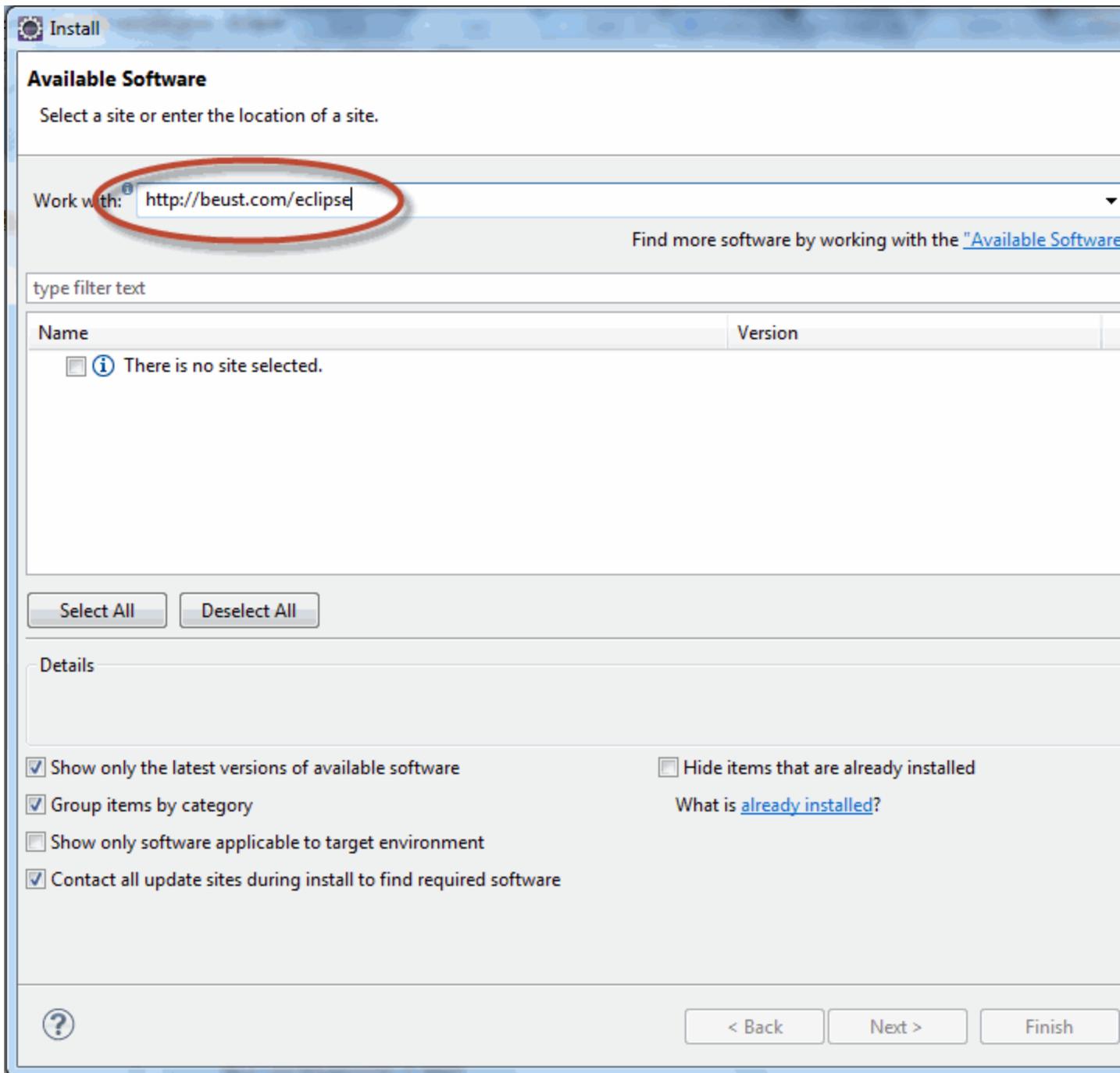
##### **Step 1:**

In Eclipse, on top menu bar, Under Help Menu, Click on "Install new Software" in help window.



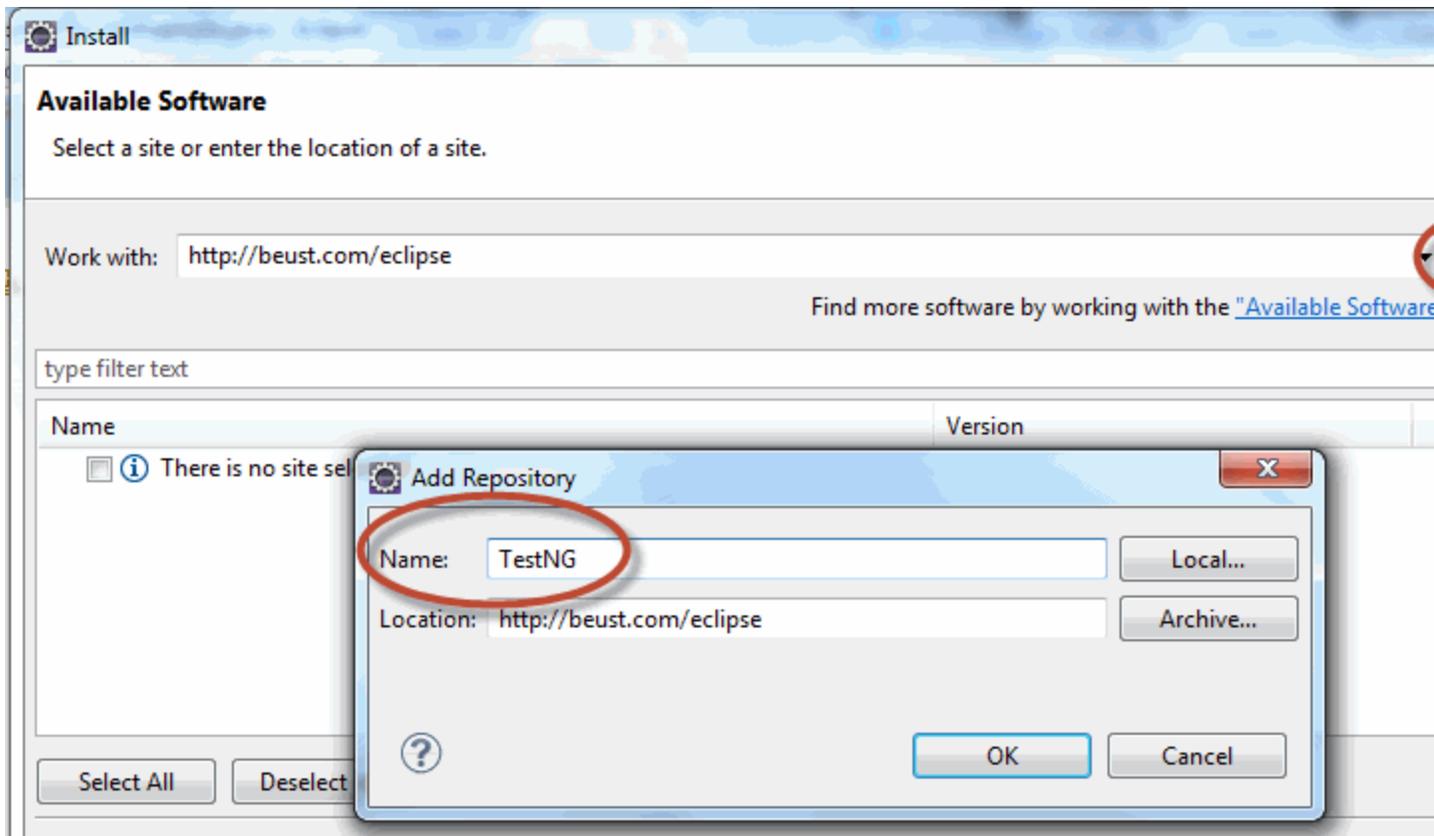
**Step 2:**

Enter the URL (<http://beust.com/eclipse/>) at Work With field and click on "Add" button.



### Step 3:

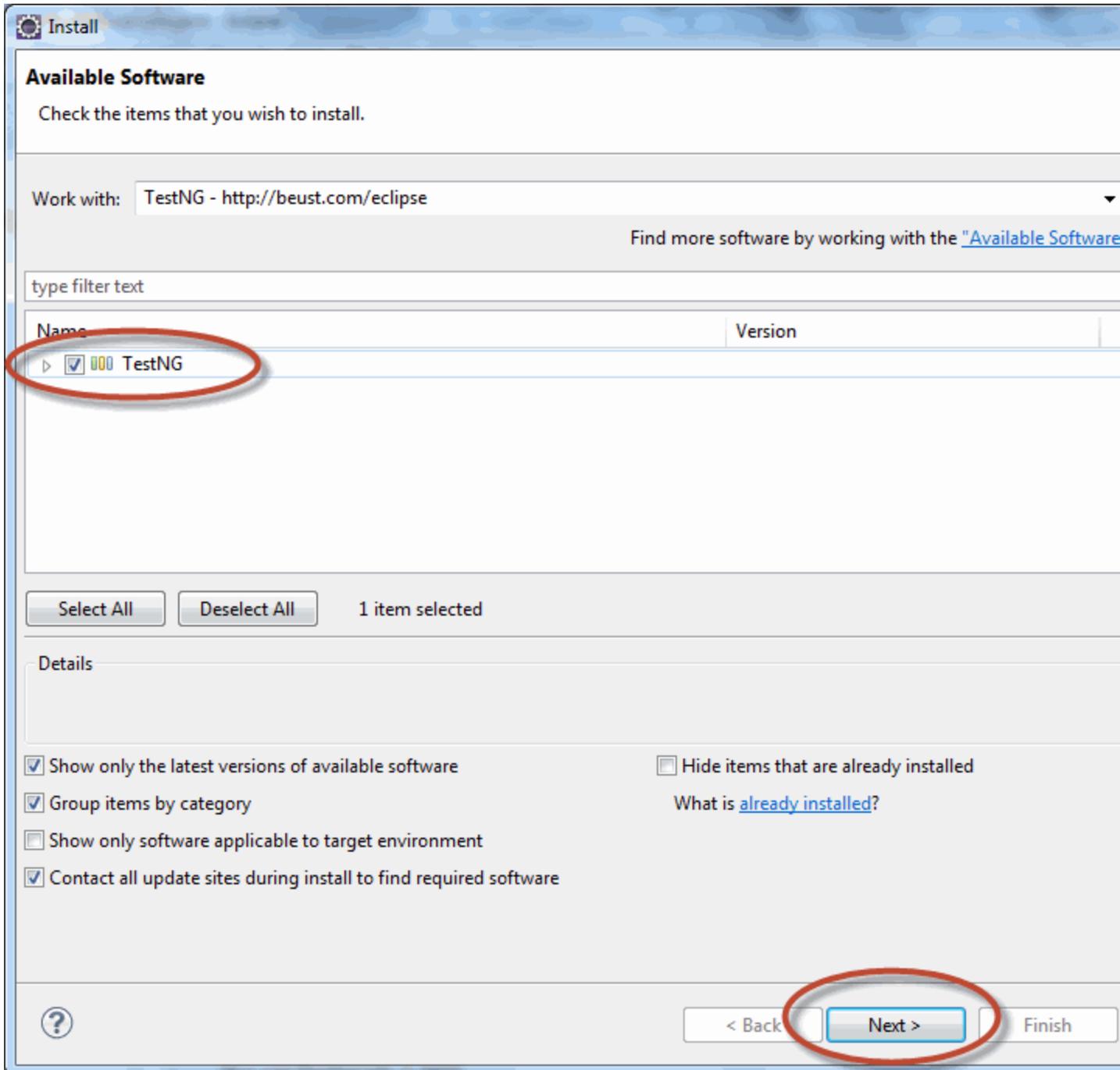
Once you click on "Add", it will display the screen, Enter the Name as "TestNG".



**Step 4:**

After clicking on "OK", it will scan and display the software available with the URL which you have mentioned.

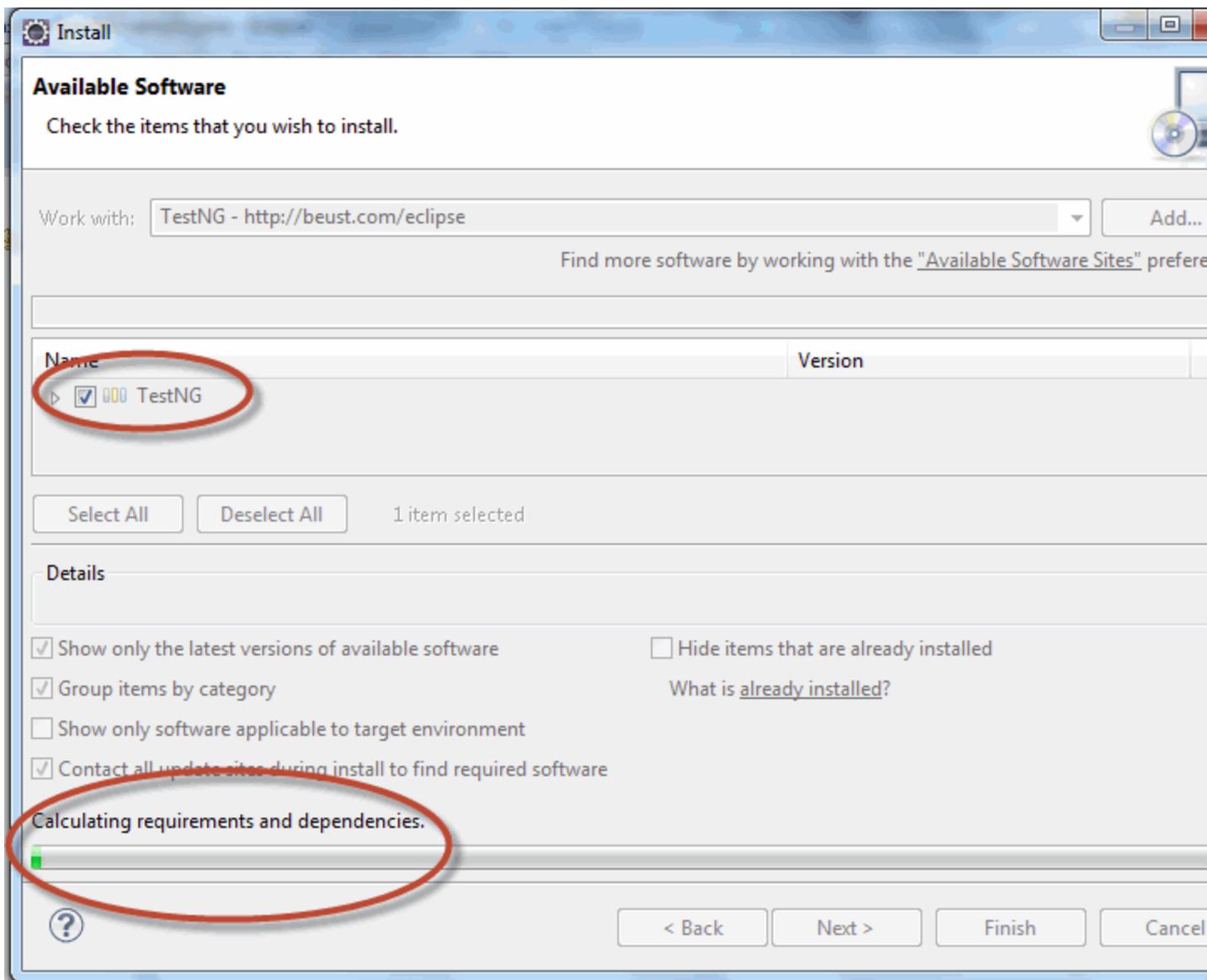
Now select the checkbox at TestNG and Click on "Next" button.



**Step 5:**

It will check for the requirement and dependencies before starting the installation.

If there is any problem with the requirements/dependencies, it will ask you to install them first before continuing with TestNG. Most of the cases it will successfully get installed nothing to worry about it.

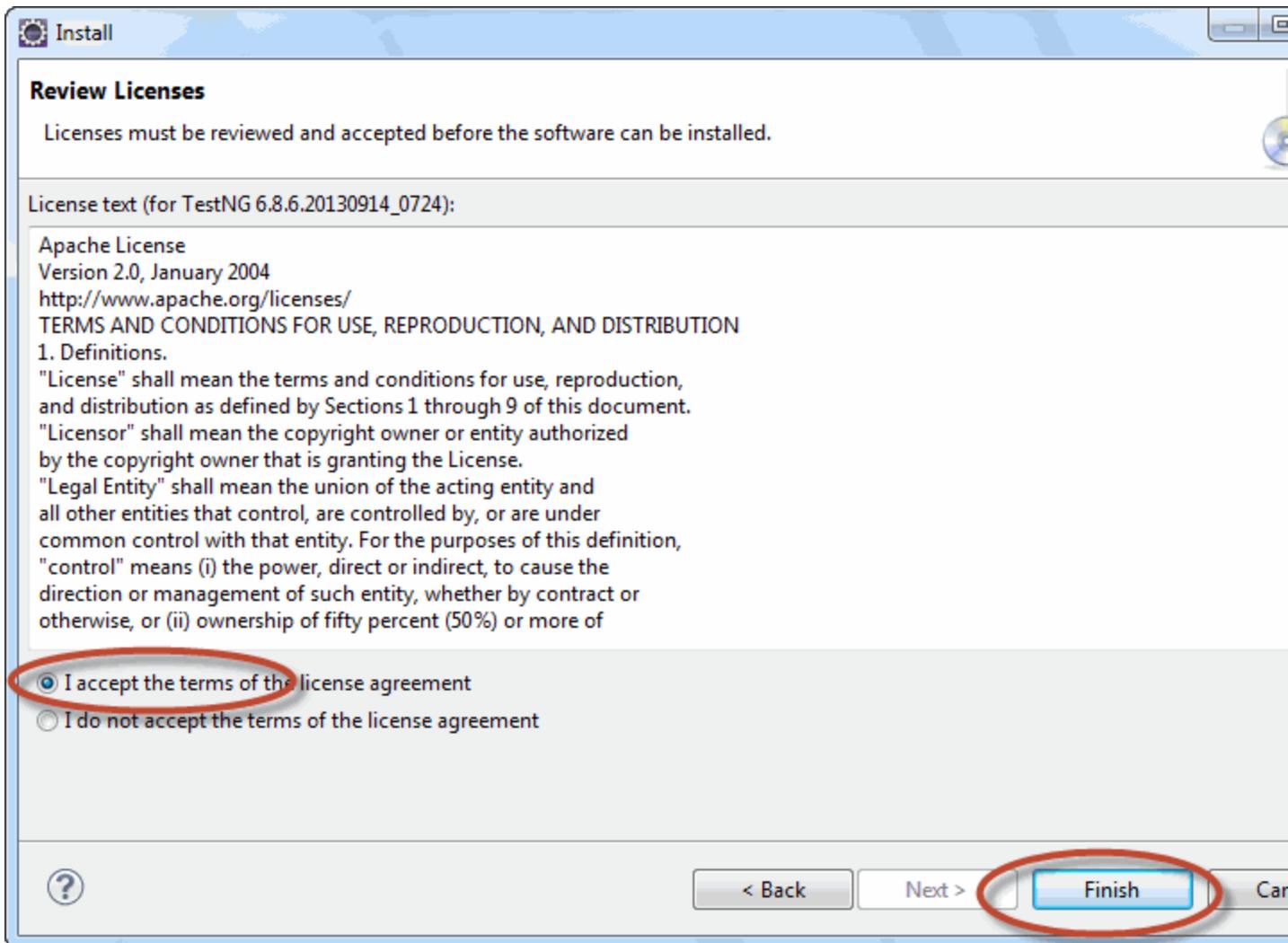


**Step 6:**

Once the above step is done, it will ask you to review the installation details. If your are ready or Ok to install TestNG, click on "Next" to continue.

**Step 7:**

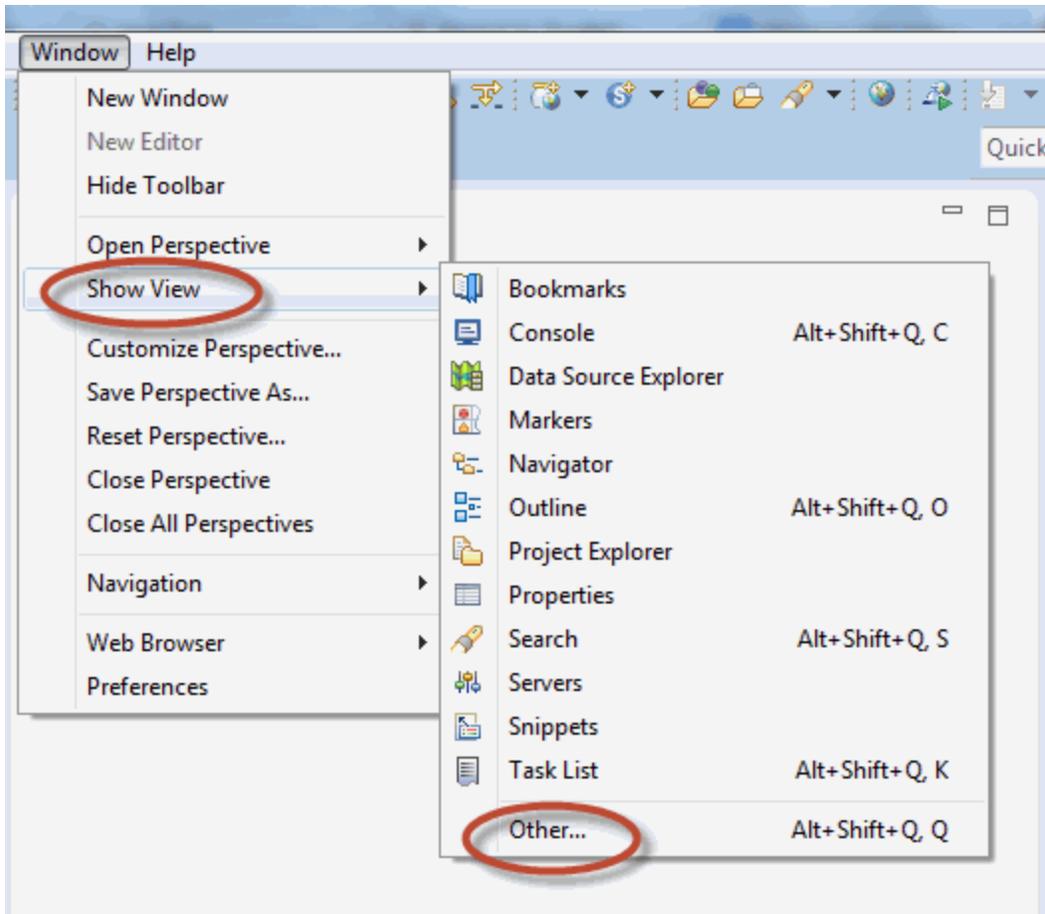
Accept the Terms of the license agreement and Click on "Finish" button.



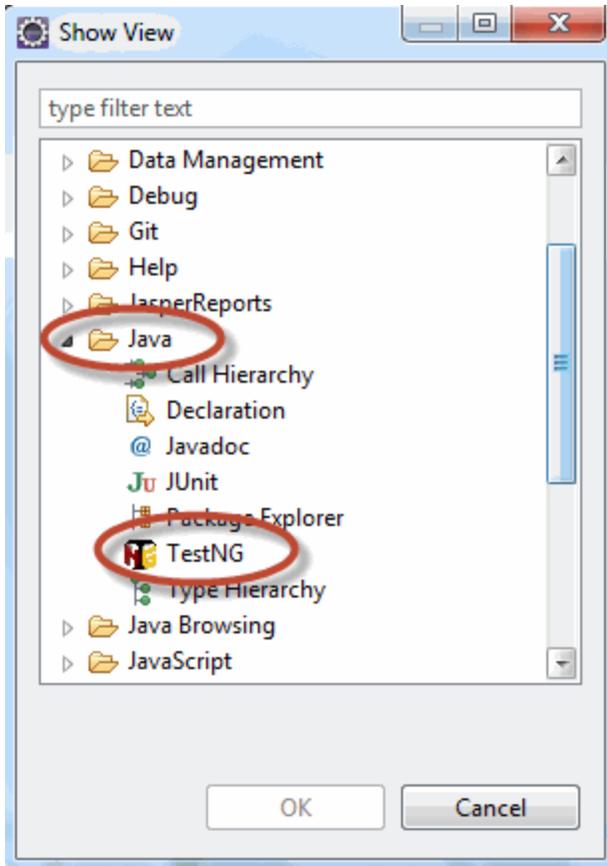
That's it... It will take a few minutes to get installed.

**Finally once the installation is done, you can check if the TestNG is installed properly or Not.**

Go to Windows Menu bar, and Mouse Over on "Show View" and Click on "Other" at the last as in the below screen shot.



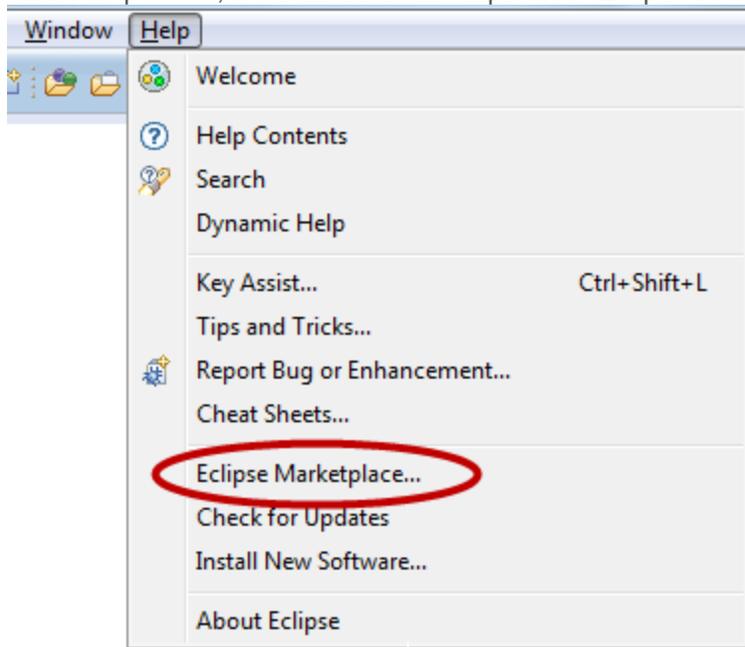
Expand Java folder and see if the TestNg is available as in the below screen shot.



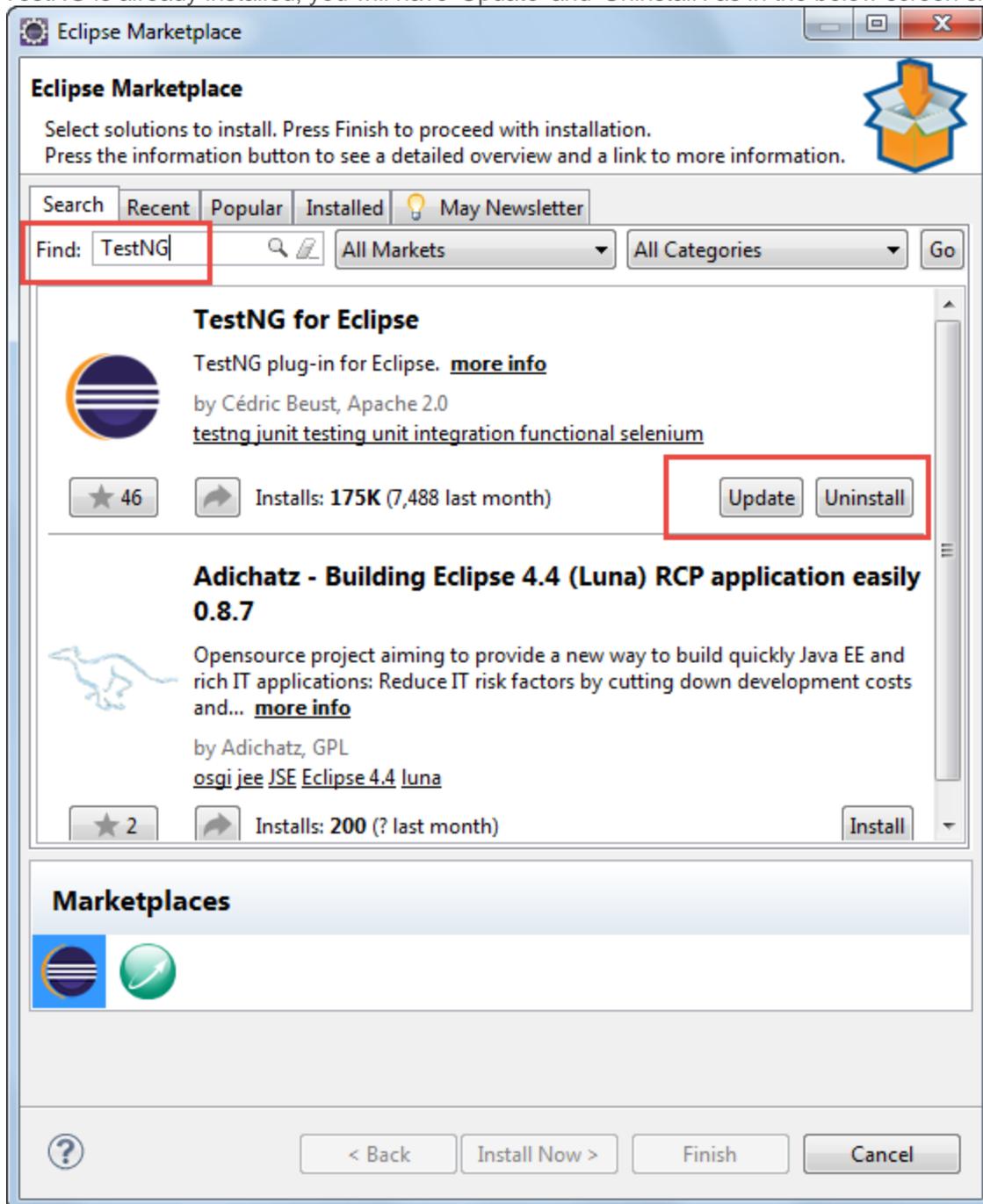
**UPDATE:**

Now eclipse is coming with Marketplace plugin by default which is a rich client solution for installing solutions listed on Eclipse Marketplace directly from an Eclipse. We can easily find third-party plugins that users can add to their Eclipse installation by using search option.

Under help menu, we should have an option as 'Eclipse Marketplace..'. Click on it.



Enter text as 'TestNG' and click on GO to search. You will now see TestNG with install option (If TestNG is already installed, you will have 'Update' and 'Uninstall'. as in the below screen shot)



## Q1-Difference between Java Program and TestNg Script?

When we execute Java program and TestNG script then functionality wise nothing will change because our script is performing the same functionality but using TestNG you will get some additional functionality.

Some benefit of TestNG Script

- 1- For even single test case you will get 3 reports. These reports generated by TestNG
- 2-You can check execution time i.e. How much time testcase has taken
- 3-Parallel execution etc

## Q2- How we can write TestNG Script?

To execute TestNG script we don't have to write separate class. We can use simple java class but here we will not write public static void main(String []args) because we are not going to execute this from jvm.

- TestNG works with Annotations and annotation can be represented by @ symbol
- @Test- is this main annotation from where TestRunner will start execution.  
In other words you can say @Test in entry point
- Step 1- Select your project
- Step 2- Go to src and create a new package and give any name (I gave demo)
- Step 3- Now select above created package and create a new class and give any name (I gave TestScript1)
- Step 4- create a simple java method and before that method use @ Test
- Step 5- Once you completed the script simply right click on above created program and select run as> TestNG Test.

### How to write TestNG Script in Selenium:

```
import org.testng.annotations.Test;

public class TestScript1 {

    @Test

    public void appTest() {

        System.out.println("Hello Selenium");

    }

}
```

## Generate Reports Selenium using TestNG:

Automation without reporting of no use. Reporting plays very important role in Automation as well in Manual testing as well. Report helps us to identify ROI and to prepare [POC](#) as well.

### Q1-Importance of Reporting in Selenium?

- 1- Reports helps you to identify the status of test case (Pass/Fail/Skip).
- 2- Using reports we calculates time taken by each test case that help to calculate ROI(Return of Investment).
- 3- You can share automation reports with your team and clients as well to share the status of testing progress etc.

### Q2- Does selenium support report generation?

Ans- No Selenium will only help you to automate your web application. If you want to generate reports then we can use Third party tools that we can integrate and can generate reports. Here TestNG comes into picture.

### Q3- Do we need to write some script or any additional code for reports?

Ans- No we do not have to write any additional code for report generation. We need to refresh our project and we will get 1 additional folder (default suite) inside that folder we will get all the reports.

One good feature about TestNG for every test case it create 3 different type of reports.

## Generate Reports Selenium using TestNG

Step 1- Create a Simple java class and write some test cases (in my case I have written 3 test case)

```

package demo;

import org.testng.annotations.Test;

public class TestCase1 {

    @Test
    public void TC001(){

        System.out.println("====Test Case 001 executed====");

    }

    @Test
    public void TC002(){

        System.out.println("====Test Case 001 executed====");

    }

    @Test
    public void TC003(){

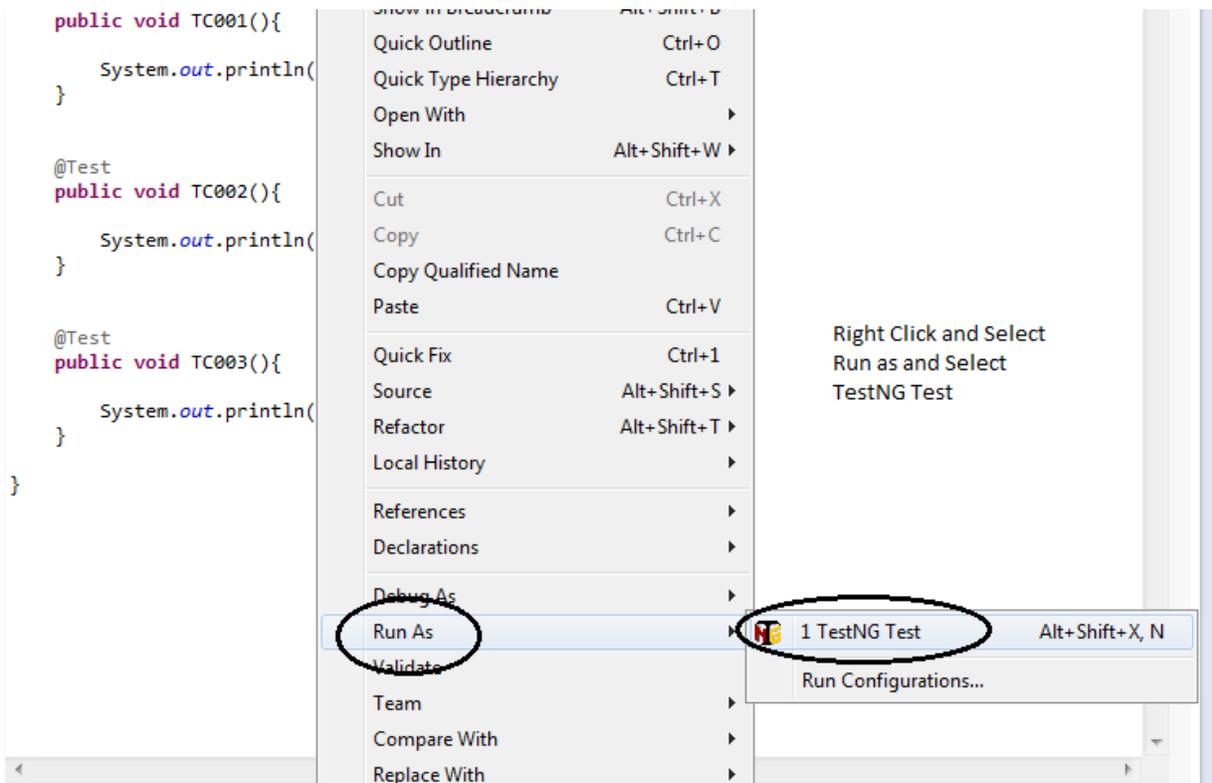
        System.out.println("====Test Case 001 executed====");

    }

}

```

Step 2- To run script simply right click and select TestNG Test.



Step 3- Now testcase will start execution and you will get console notification about the status and wait until the program is getting finished.

```
====Test Case 001 executed====  
====Test Case 001 executed====  
====Test Case 001 executed====  
PASSED: TC001  
PASSED: TC002  
PASSED: TC003  
  
=====
```

Default test  
Tests run: 3, Failures: 0, Skips: 0

```
=====
```

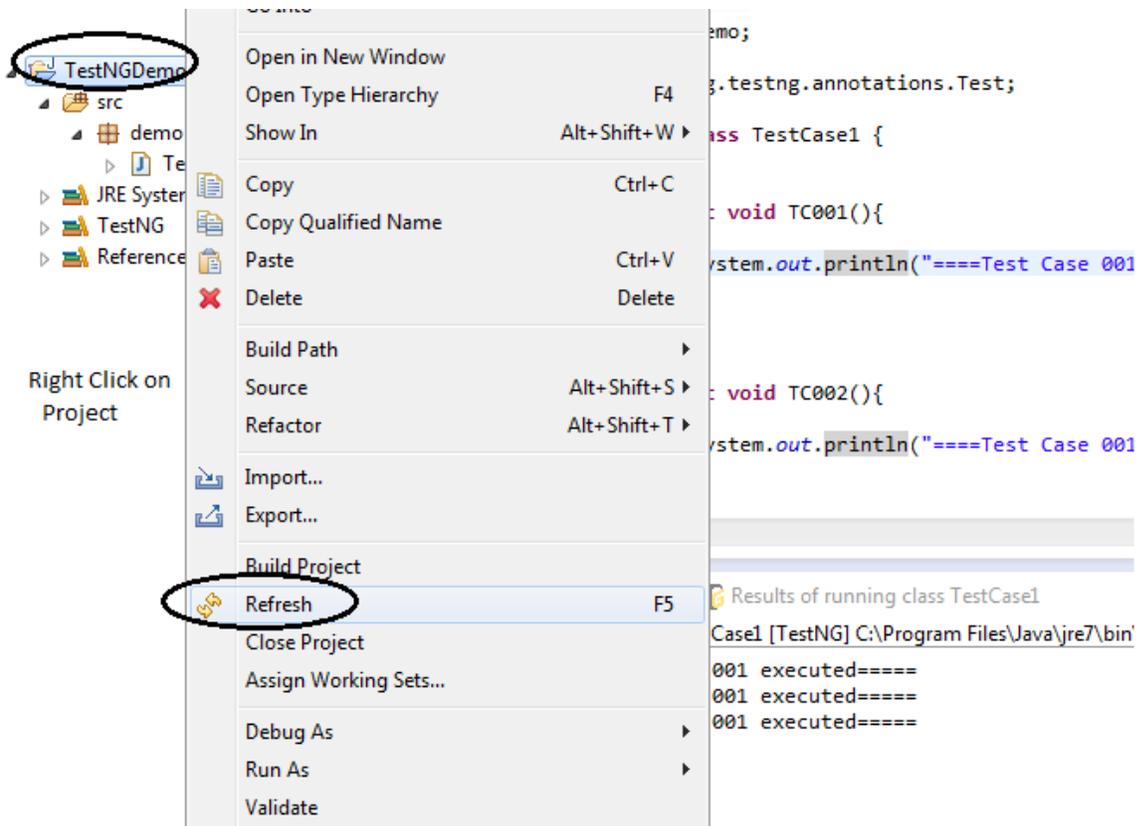
### **Report type 1- Console output.**

Since console output is not useful because you can't share and send to others so we required some HTML reports as well

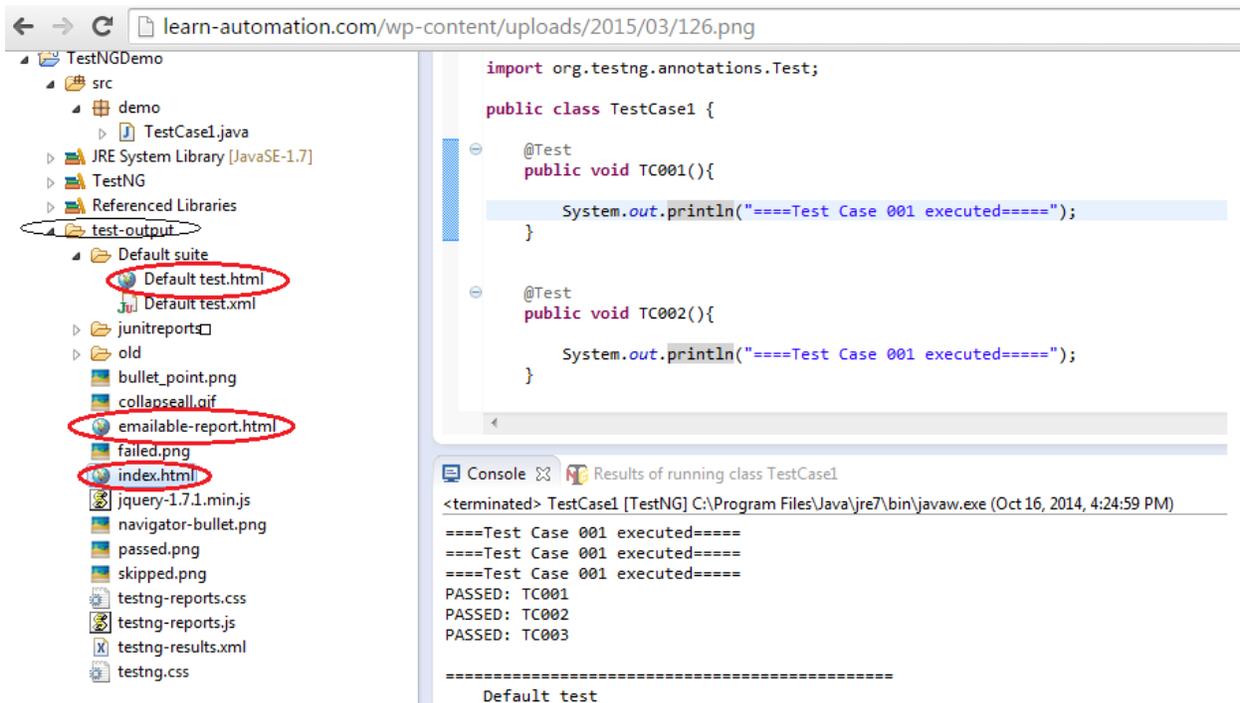
TestNG guys already gave a solution for this so for even single test case you will get HTML reports as well.

Report type 2- HTML report

We have to refresh you project and reports folder will come automatically.



After refreshing you will get below folder ready



Let's open each report and check the content

Note- Open Each report in Web Browser select each report then right click and open in browser

Generate Reports Selenium using TestNG

First open Default test.html

#### Default test

Tests passed/Failed/Skipped:	3/0/0
Started on:	Thu Oct 16 16:25:29 IST 2014
Total time:	0 seconds (167 ms)
Included groups:	
Excluded groups:	

*(Hover the method name to see the test class name)*

PASSED TESTS			
Test method	Exception	Time (seconds)	Instance
TC001 Test class: demo.TestCase1		0	demo.TestCase1@199de59
TC002 Test class: demo.TestCase1		0	demo.TestCase1@199de59
TC003 Test class: demo.TestCase1		0	demo.TestCase1@199de59

Second open emailable-report.html

Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
<b>Default suite</b>						
<a href="#">Default test</a>	3	0	0	167		

Class	Method	Start	Time (ms)
<b>Default suite</b>			
<b>Default test — passed</b>			
demo.TestCase1	<a href="#">TC001</a>	1413456929219	92
	<a href="#">TC002</a>	1413456929318	2
	<a href="#">TC003</a>	1413456929322	2

## Default test

demo.TestCase1#TC001

[back to summary](#)

Third open Index.html

All suites

### Default suite

**Info**

- C:\Users\...AppData\Local\Temp\testng-eclipse--1773637589\testng-customsuite.xml
- 1 test
- 0 groups
- Times
- Reporter output
- Ignored methods
- Chronological view

**Results**

- 3 methods, 3 passed
- Passed methods ([show](#))



This part is info section please explore each option



This is result section where all testcase related info(Status) will come

## Cross Browser Testing using Selenium Webdriver

### What is Cross browser testing?

Cross browser, testing refers to testing the application in multiple browsers like IE, Chrome, Firefox so that we can test our application effectively.

Cross browser, testing is very important concept in Automation because here the actual automation comes into picture.

Example- Suppose if you have 20 test cases that you have to execute manually, so it is not a big deal right we can execute in 1 day or 2 day. However, if the same test cases you have to execute in five browsers it means 100 test cases then probably you will take one week or more than one week to do the same and it will be quite boring as well.

If you automate these 20 test cases and run them then it will not take more than one or two hour depends on your test case complexity.

### Cross Browser Testing using Selenium Webdriver

To achieve this we will use TestNG parameter feature, we will pass parameter from TestNG.xml file, and based on our parameter Selenium will initiate our browsers.

In this scenario we will run the same testcase with two different browser parallel

## Step 1- Write testcase

Package SampleTestcases;

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.ie.InternetExplorerDriver;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class TestCase1 {

    @Test

    // Here this parameters we will take from testng.xml
    @Parameters("Browser")
    public void test1(String browser) {

        if(browser.equalsIgnoreCase("FF")){

            WebDriver driver=new FirefoxDriver();

            driver.manage().window().maximize();

            driver.get("http://www.facebook.com");

            driver.quit();

        }
        else if(browser.equalsIgnoreCase("IE")){

            System.setProperty("webdriver.ie.driver", "./server/IEDriverServer.exe");

            WebDriver driver=new InternetExplorerDriver();

            driver.manage().window().maximize();

            driver.get("http://www.facebook.com");

            driver.quit();

        }
    }
}
```

## Step 2- Create testng.xml and specify

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
```

Here parallel is an attribute which specify the mode of execution and thread-count specify how many browser should open

```
<suite name="Suite" parallel="tests" thread-count="2">
```

```
<test name="Test">
```

```
<parameter name="Browser" value="FF" />
```

```
<classes>
```

```
<class name="SampleTestcases.TestCase1"/>
```

```
</classes>
```

```
</test>
```

```
<test name="Test1">
```

```
<parameter name="Browser" value="IE" />
```

```
<classes>
```

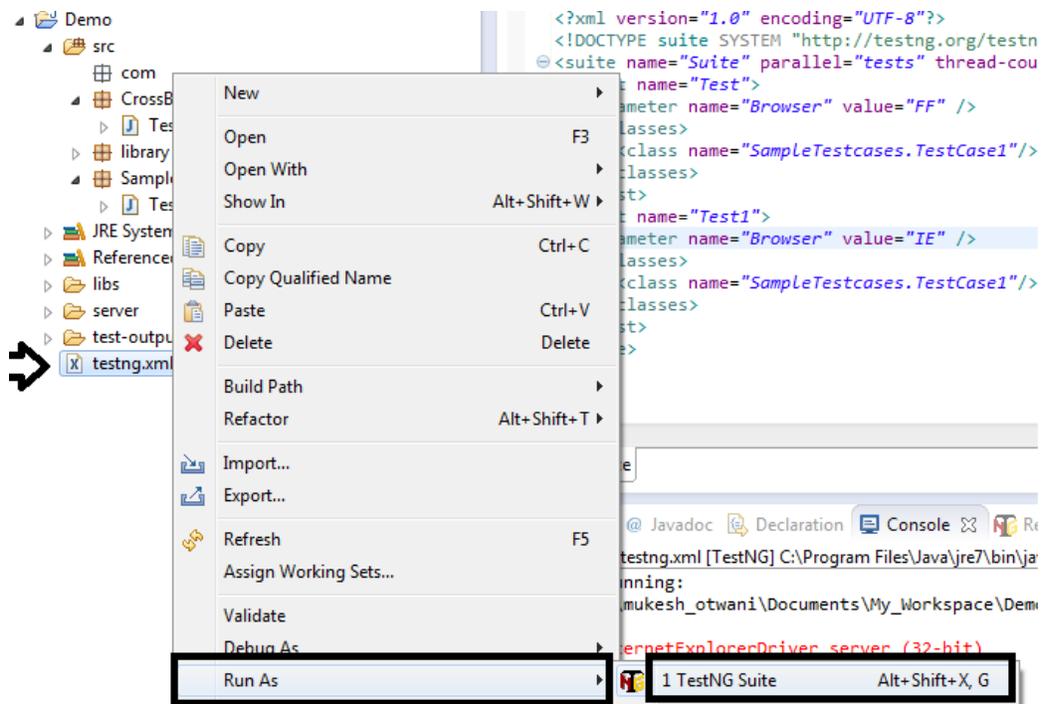
```
<class name="SampleTestcases.TestCase1"/>
```

```
</classes>
```

```
</test>
```

Step 3- Run this xml file refer the below screenshot.

Note- To create testng.xml- Right click on your testcase then go to TestNG then convert to TestNG> It will generate testng.xml then make changes as per above xml file and finish. You will get testng.xml file inside project



Verify the output.

```
Started InternetExplorerDriver server (32-bit)
2.42.0.0
Listening on port 6243
```

```
=====
Suite
Total tests run: 2, Failures: 0, Skips: 0
=====
```

## DataProvider in TestNG

Marks a method as supplying data for a test method. The annotated method must return an `Object[][]` where each `Object[]` can be assigned the parameter list of the test method.

The `@Test` method that wants to receive data from this `DataProvider` needs to use a `dataProvider` name equals to the name of this annotation.

The name of this data provider. If it's not supplied, the name of this data provider will automatically be set to the name of the method.

In the below example we will pass the data from `getData()` method to data provider. We will send 3 rows and 2 columns ie. we will pass three different usernames and passwords.

```
@Test(dataprovider="getData")

Public static void testLogin(String username, String password){

System.Out.println("User Name:"+ username+ " "+"password:" password);

}

@DataProvider

Public Object[][] getData(){

Object[][] data = new Object[3][2];

//1 row

data[0][0] ="username";
data[0][1] ="xxxxx";

data[1][0] ="aaaaa";
data[1][1] ="yyyyy";

data[2][0] ="ppppp";
data[2][1] ="zzzzz";

return data;
}
```

#### [Execute failed test cases using Selenium](#)

Most of the time we have faced this question in interviews that Can we execute only failed test cases in Selenium or can we identify only failed test cases in Selenium and re-run them.

I really love this feature of TestNG that you can run only failed test cases explicitly without any code. This can be easily done by running one simple `testng-failed.xml`.

#### [Execute Failed test cases using Selenium](#)

### **Real time Example**

Take an example that you have one test suite of 100 test cases and once you start execution of test suite there are number of chances that some test cases will fail. Consider 15 test cases are failing out of 100 now you need to check why these test cases are failing so that you can analyze and find out the reason why they have failed.

**Note- Your script can fail due to so many reason some of them are**

1-Some locator has been changed in application because application is getting new feature- so in this case you need to modify your script in other words you have to refine your script.

You can not avoid maintenance of test script you always have to maintain your scripts

2- Either functionality has been broken- in this case you have to raise a defect and assign to respective person.

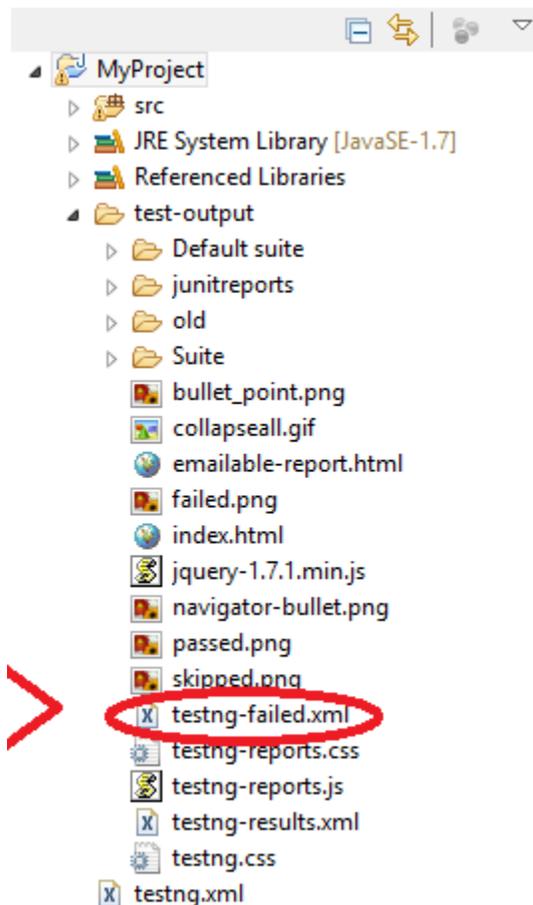
## Execute failed test cases using Selenium

### Steps

1-If your test cases are failing then once all test suite completed then you have to refresh your project . Right click on project > Click on refresh or Select project and press f5.

2-Check test-output folder at last you will get testng-failed.xml

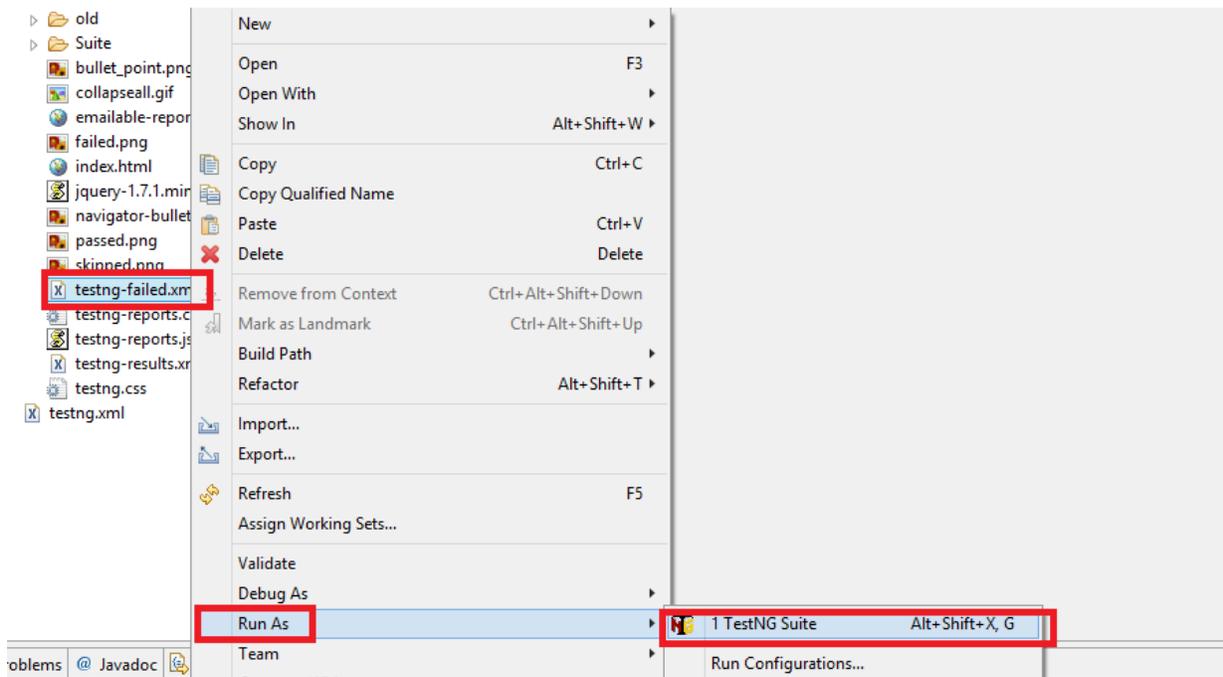
3- Now simply run testng-failed.xml.



## How to run testng-failed.xml

We don't have to perform any other activity once you will get testng-failed.xml double click on this and analyze which test case are failing and why . Then modify your script and run it.

To run above xml simple right click on xml then Select run as then TestNG Suite.



## Generate log in Selenium using TestNG reporter

### Reporter Class in TestNG

Reporter is a separate class in TestNG that is available under org.testng package.

In Selenium you can specify steps, which we are performing so that we can check our output, and in case any issue we can debug at which point our test cases failed.

Let me explain with the help of an example

Scenario 1- We have a script, which has almost 100 steps so if we do not use Reporter or any other reporting feature then we have to check where exactly we are getting issue. We can use Reporter class here and we can check steps which have executed successfully and where our program has stopped.

Before starting below program make sure [Eclipse Selenium and TestNG](#) is installed

## Generate log using TestNG

1 Syntax-Reporter.log("String", boolean);

```
package testngDemo;
2 import org.openqa.selenium.WebDriver;
3 import org.openqa.selenium.firefox.FirefoxDriver;
4 import org.testng.Reporter;
5 import org.testng.annotations.Test;
6
7 public class ReporterDemo {
8
9     @Test
10    public void testReport(){
11
12        WebDriver driver=new FirefoxDriver();
13
14        Reporter.log("Browser Opened");
15
16        driver.manage().window().maximize();
17
18        Reporter.log("Browser Maximized");
19
20        driver.get("http://www.google.com");
21
22        Reporter.log("Application started");
23
24        driver.quit();
25
26        Reporter.log("Application closed");
27    }
28 }
29
30 }
```

Once you run above program check [TestReport](#)

**Default test**

Tests passed/Failed/Skipped:	1/0/0
Started on:	Sun Nov 16 00:47:40 IST 2014
Total time:	39 seconds (39145 ms)
Included groups:	
Excluded groups:	

*(Hover the method name to see the test class name)*

Test method	Exception	Time (seconds)	Instance
<b>testReport</b> Test class: testngDemo ReporterDemo Status: <b>Passed</b> (all methods)		39	testngDem
Browser Opened			
Browser Maximized			
Application started			
Application closed			

 All log which we written on Eclipse

Now if you want to print on console and [html report](#) as well then we have same method with diff argument

if boolean value set to true then values will come on console and html report as well

if boolean value set to false then values will come on html report only.

[How to disable Selenium Testcases using TestNG Feature](#)

## Why to Disable Selenium Test cases?

Before moving forward you should have Eclipse TestNG setup ready if still not configured then please do [Selenium Eclipse setup now](#).

Once our test suite size will increase day by day then chances are high that you don't want to execute some test cases.

Ok so let's consider a scenario that you have written 10 testcases but now you want to execute only 5 testcase because other 5 are working correctly

## How to Disable Selenium Test cases?

In TestNG we can achieve this by simply adding enable attribute and can set value to true/false.

```
@Test(enable=true)
```

If test case enable as false then while running test cases TestRunner simply will ignore this testcase and will only run testcase whose enable is set to true.

**Note- By default @Test is set to enable**

Scenario – I have 3 test case but I want to execute only 2 test case.

Precondition- TestNG should be installed

```
package demotestcase;

5 import org.testng.annotations.Test;
6 public class TestEnableTC {
7
8     @Test
9     public void testLoginApp(){
10
11         System.out.println("User is able to login successfully");
12     }
13
14     @Test(enabled=false)
15     public void testRegisteruser(){
16
17         System.out.println("User is able to register successfully");
18     }
19
20     @Test
21     public void testLogoutApp(){
22
23         System.out.println("User is able to logout successfully");
24     }
25 }
}
```

```
User is able to login successfully ←
User is able to logout successfully
PASSED: testLogin
PASSED: testLogout

=====
Default test
Tests run: 2, Failures: 0, Skips: 0
=====

Default suite
Total tests run: 2, Failures: 0, Skips: 0
=====
```

## How to group test cases in Selenium

Grouping in Automation is quite interesting feature through which you can easily categorized your test cases based on requirement.

For Example- If we have 200 test case out of these some are end to end test cases, some are functional test case and some are regression or smoke test cases so if you don't categorized these test cases these all test case will come under one common category.

## Group Selenium Script using TestNG

For grouping in Selenium, we have to use TestNG feature through which we can create groups and maintain them easily.

We can include and exclude groups through testng.xml file

### How to create group in TestNG

We have predefined syntax for this

Syntax

```
@Test(groups={"groupname"})
```

Example

```
@Test(groups={"smoke"})
```

We can create multiple froup as well

```
@Test(groups={"smoke", "Regression"})
```

After grouping in Selenium we can specify the include and exclude in testng.xml.

<include> – It tells testng.xml that which group we need to execute.

<exclude>- It tells testng.xml which group we have to Skip

Therefore, if you execute your test cases with grouping then we have to modify our testng.xml file and we have to explicitly specify the group.

#### Sample Program

```
package testngDemo;

import org.testng.annotations.Test;

5 public class TestGroupDemo {
16
17
18     @Test(groups={"Smoke"})
19     public void login(){
20
21         System.out.println("Login done");
22         System.out.println("Smoke Scenario passed");
23     }
24
25     @Test(groups={"Regression"})
26     public void register(){
27         System.out.println("Registration done");
28     }
29
30 }
```

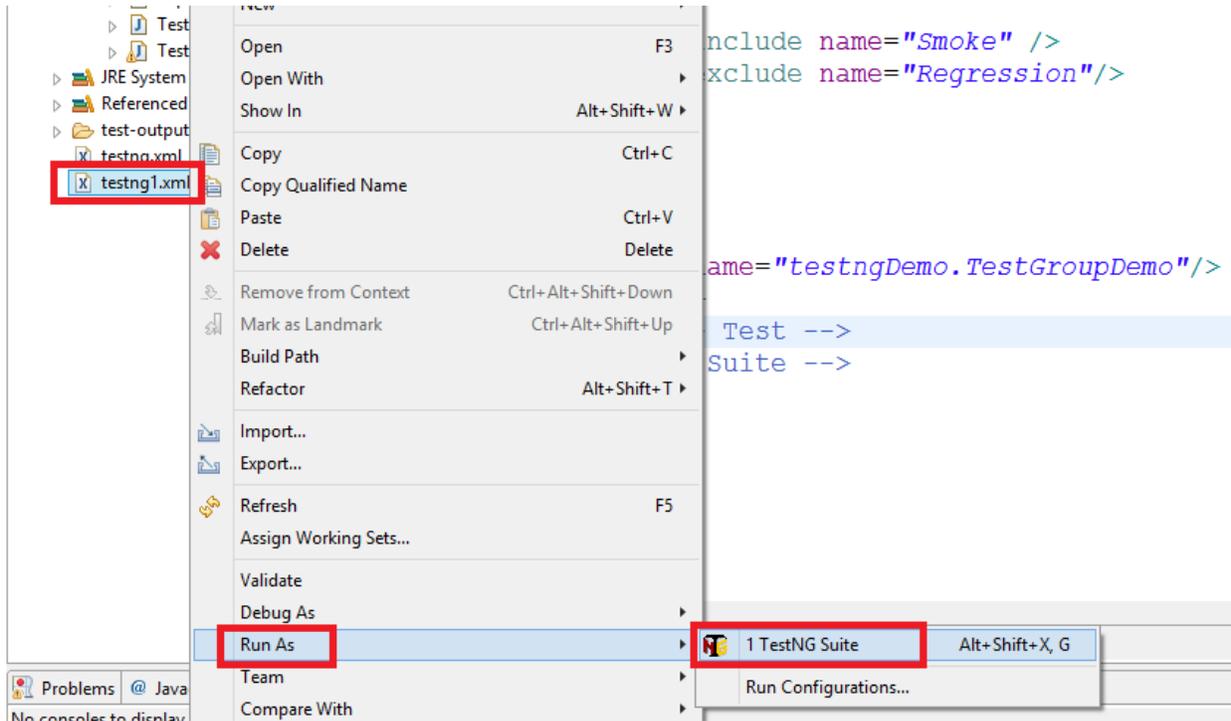
We need to modify the testng.xml

```
2  <?xml version="1.0" encoding="UTF-8"?>
3  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
4  <suite name="Suite" parallel="none">
5  <test name="Test">
6  <groups>
7
8  <run>
9
10     <include name="Smoke" />
11     <exclude name="Regression"/>
12
13 </run>
14
15 </groups>
16 <classes>
17 <class name="testngDemo.TestGroupDemo"/>
18 </classes>
19 </test> <!-- Test -->
20 </suite> <!-- Suite -->
21
```

I created a new tag called group, inside group we have mention include and exclude tag, and we have mention group name, which we have to include and exclude.

Once you run above xml it will run all the test cases, which belongs to Smoke group category.

Let us run this xml and see the output.



### Console output.

```
Login done  
Smoke Scenario passed
```

```
=====+=====  
Suite  
Total tests run: 1, Failures: 0, Skips: 0  
=====
```

## How to create dependency between test cases in Selenium

I am sure you must be confused with Title like How to create a dependency between test cases in Selenium but in this post we will see this through example. In real time, you will come across many situations where you have to create test cases which are dependent on the previous test case. Take an example I have 3 test case. First one which verify login credential and start application  
Second test cases check some validation or do some activity

Third test case simply just for log-out activity.

If my first test case does not work (failed) that it does not make any sense to run second and third test case so in these type of scenarios you can use dependency between test case.

How can I do this?

TestNG already comes with these features so you can use the same with @Test annotations

```
1 Syntax- @Test(dependsOnMethods={"loginTestCase"})
2
3 public void testAccount()
4 {
5
6 // sample code
7
8 }
```

## How to create dependency between test cases in Selenium

It means this test case name testAccount depends on test case whose name is loginTestCase so until loginTestCase will not execute successfully this test case won't execute so if loginTestCase will pass this test case will execute and if loginTestCase will fail then this test case won't execute and runner will skip this test cases and this test case will come under skip test cases.

## Let's implement the same

```
1 package Demo;
2
3 import org.testng.Assert;
4 import org.testng.annotations.Test;
5
6 public class TestMultiple {
7
8     @Test
9     public void testLogin()
10    {
11
12    System.out.println("login done");
13
14    }
15
16    @Test(dependsOnMethods={"testLogin"})
17    public void testAccount()
18    {
19    System.out.println("Account has been created");
20
21    }
22
23    @Test(dependsOnMethods={"testLogin","testAccount"})
24    public void testLogout()
25    {
26    System.out.println("logout");
27
28    }
29
30 }
```

In the above scenario if testLogin will pass then only testAccount will execute and testLogout will only execute of testLogin and testAccount will be executed

But if testLogin will fail due to some error or exception the testAccount will not be executed and this test case will skip and testLogout is dependent on testLogin and testAccount so now this will also come into skipped category.

## Let's implement again

```
1 import org.testng.Assert;
2 import org.testng.annotations.Test;
3
4 public class TestApplication {
5
6     @Test
7     public void testLogin()
8
9     {
10    Assert.assertEquals("Selenium", "Selinium");
```

```

11 System.out.println("login done");
12
13 }
14
15 @Test(dependsOnMethods={"testLogin"})
16 public void testAccount()
17 {
18     System.out.println("Account has been created");
19 }
20
21 }
22
23 @Test(dependsOnMethods={"testLogin", "testAccount"})
24 public void testLogout()
25 {
26     System.out.println("logout");
27 }
28 }
29 }
30 }
31 }

```

#### Output

```

FAILED: testLogin
java.lang.AssertionError: expected [Selnium] but found [Selenium] at
org.testng.Assert.fail(Assert.java:94)
SKIPPED: testAccount
SKIPPED: testLogout

```

```

=====
Default test
Tests run: 3, Failures: 1, Skips: 2
=====
=====
Default suite
Total tests run: 3, Failures: 1, Skips: 2
=====

```

## OUTPUT in HTML

### Default test

Tests passed/Failed/Skipped:	0/1/2
Started on:	Sat Nov 01 15:49:55 IST 2014
Total time:	0 seconds (181 ms)
Included groups:	
Excluded groups:	

## How to execute testng.xml files using Java Program

What is need of this in Selenium?

Let me explain this with a help of an example.

Consider you have 10 test cases in Selenium and you have created a testng.xml to execute all test cases.

Now you create another 20 test cases which belong to the same project but they belong to different module so again we created testng1.xml for the same.

If the same process goes on then at last you must be having couple of xml files which will be having all the test cases.

Now the real problem is if you want to execute all xml in one shot then you cannot do because Eclipse will allow executing only one xml.

In this case we can take help of an TestNG class which allow us to execute multiple xml in one shot. In below example I am having one testng.xml file which I will execute using Java Program.

### How to implement in Selenium Webdriver

```
1 import java.util.ArrayList;
2 import java.util.List;
3
4 import org.testng.TestNG;
5
6 public class RunTestNG {
7
8 public static void main(String[] args) {
9
10 // Create object of TestNG Class
11 TestNG runner=new TestNG();
12
13 // Create a list of String
14 List<String> suitefiles=new ArrayList<String>();
15
16 // Add xml file which you have to execute
17 suitefiles.add("C:\\Users\\Documents\\Blog6March\\dummy16june\\testng.xml");
18
19 // now set xml file for execution
20 runner.setTestSuites(suitefiles);
21
22 // finally execute the runner using run method
23 runner.run();
24 }
25
26 }
```

I above program I have taken one xml file only but if you have mutple files then repeat the same procedure and add files.

### TestNG XML example to execute with package names

In testng.xml file we can specify the specific package name (s) which needs to be executed. In a project there may be many packages, but we want to execute only the selected packages. The below is the example testng.xml which will execute the specific packages.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="example suite 1" verbose="1" >
<test name="Regression suite 1" >
<packages>
<package name="com.first.example" />
</packages>
</test>
</suite>
```

### TestNG XML example to execute Multiple Classes

In testng.xml file we can specify multiple name (s) which needs to be executed.

In a project there may be many classes, but we want to execute only the selected classes.

We can pass class names of multiple packages also. If say suppose, we want to execute two classes in one package and other class from some other package.

The below is the example testng.xml which will execute the class names that are specified.

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="example suite 1" verbose="1" >
<test name="Regression suite 1" >
<classes>
<class name="com.first.example.demoOne"/>
<class name="com.first.example.demoTwo"/>
<class name="com.second.example.demoThree"/>
</classes>
</test>
</suite>
```

## Skip Test in TestNG

Using TestNG, we have multiple ways to Skip a test based on our requirement. We can Skip complete test without executing it or we can Skip a test when a specific condition is not satisfied.

In TestNG, `@Test(enabled=false)` annotation is used to skip a test case if it is not ready to test. We don't need to import any additional statements.

As in JUnit, TestNG will not show you the other test method as Skipped or Ignored. It will not consider that case method at all when the annotation is mentioned as “`@Test(enabled=false)`”

And We can Skip a test by using TestNG [Skip Exception](#) if we want to Skip a particular Test. Syntax:

```
throw new SkipException("message");
```

## Timeout Test in TestNG

When a test method is taking longer time than the specified time (in milliseconds) to execute, then that test method will be terminated and marked as failed, this feature is available in both JUnit 4 and TestNG.

While running test methods there can be cases where certain test methods get struck or may take longer time than to complete the execution than the expected. We need to handle these type of cases by specifying Timeout and proceed to execute further test cases / methods

```
@Test(timeout=1000) // specify time in milliseconds
    public void executetimeOut() throws InterruptedException{
        Thread.sleep(3000);
        // Thread.sleep(500);
    }
```

## TestNG Test Case Priority

In TestNG "Priority" is used to schedule the test cases. When there are multiple test cases, we want to execute test cases in order. Like First we need to execute a test case "Registration" before login.

In order to achive, we use need to add annotation as `@Test(priority=??)`. The default value will be zero for priority.

If you don't mention the priority, it will take all the test cases as "priority=0" and execute.

If we define priority as "priority=", these test cases will get executed only when all the test cases which don't have any priority as the default priority will be set to "priority=0"

### **Include and Exclude Test Methods in TestNG**

TestNg provides an option to include or exclude Groups, Test Methods, Classes and Packages using include and exclude tags by defining in testng.xml.

First we will create an examples to use include and exclude tags for Test Methods in a class.

We will create a Class with three Test Methods. In that we will include two test methods and try to exclude one test method.

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Sample Test Suite" verbose="1" >
<test name="Method Test Cases" >
<classes>
<class name="com.easy.entry.AddTestCase">
<methods>
<include name="addLocationTestCase" />
<include name="addDepartmentTestCase" />
<exclude name="addEmployeeTestCase" />
</methods>
</class>
</classes>
</test>
</suite>
```

# Working With JUNIT

## JUnit introduction

JUnit is an open source testing framework which is used to write and run repeatable automated tests, so that we can be ensured that our code works as expected. JUnit is widely used in industry and can be used as stand alone Java program (from the command line) or within an IDE such as Eclipse.

### JUnit provides:

- Assertions for testing expected results.
- Test features for sharing common test data.
- Test suites for easily organizing and running tests.
- Graphical and textual test runners.

### JUnit is used to test:

- an entire object
- part of an object – a method or some interacting methods
- interaction between several objects

### JUnit Simple Example using Eclipse

In this section we will see a simple [JUnit](#) example. First we will present the class we would like to test:

```
package com.javacodegeeks.junit;  
  
public class Calculate {  
  
    public int sum(int var1, int var2) {  
        System.out.println("Adding values: " + var1 + " + " + var2);  
        return var1 + var2;  
    }  
  
}
```

In the above source code, we can notice that the class has one public method named `sum()`, which gets as inputs two integers, adds them and returns the result. So, we will test this method. For this purpose, we will create another class including methods that will test each one of the methods of the previous class (in this case, we have only one method to be tested). This is the most common way of usage. Of course, if a method is very complex and extended, we can have more than one test methods for this complex

method. The details of creating test cases will be presented in the next sections. Below, there is the code of the class named `CalculateTest.java`, which has the role of our test class:

```
package com.javacodegeeks.junit;

import static org.junit.Assert.*;

import org.junit.Test;

public class CalculateTest {

    Calculate calculation = new Calculate();
    int sum = calculation.sum(2, 5);
    int testSum = 7;

    @Test
    public void testSum() {
        System.out.println("@Test sum(): " + sum + " = " + testSum);
        assertEquals(sum, testSum);
    }

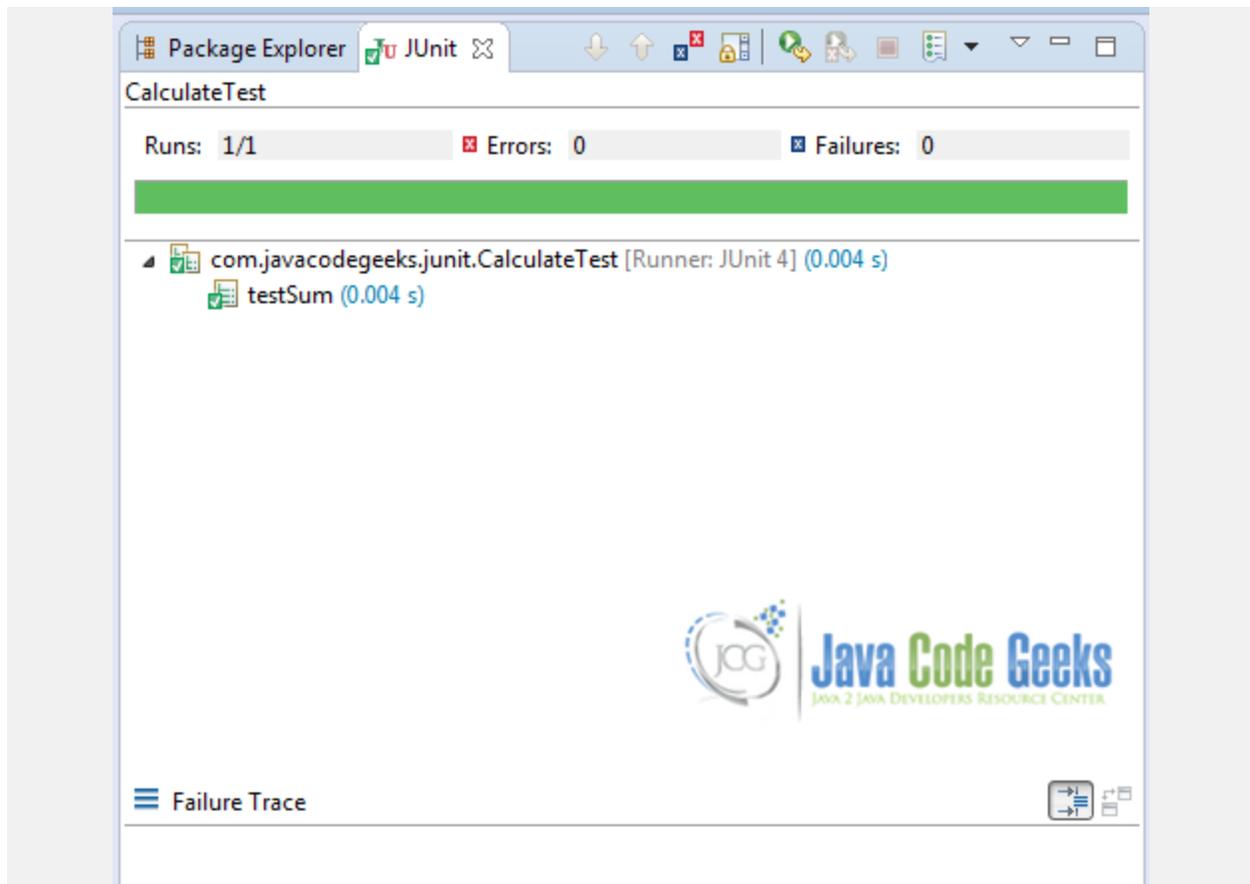
}
```

Let's explain the above code. Firstly, we can see that there is a `@Test` annotation above the `testSum()` method. This annotation indicates that the public void method to which it is attached can be run as a test case. Hence, the `testSum()` method is the method that will test the `sum()` public method. We can also observe a method called `assertEquals(sum, testsum)`. The method `assertEquals ([String message], object expected, object actual)` takes as inputs two objects and asserts that the two objects are equal.

If we run the test class, by right-clicking in the test class and select *Run As -> JUnit Test*, the program output will look like that:

```
Adding values: 2 + 5
@Test sum(): 7 = 7
```

To see the actual result of a JUnit test, Eclipse IDE provides a JUnit window which shows the results of the tests. In this case where the test succeeds, the JUnit window does not show any errors or failures, as we can see in the image below:



Now, if we change this line of code:

```
int testSum = 10;
```

so that the integers to be tested are not equal, the output will be:

```
Adding values: 2 + 5  
@Test sum(): 7 = 10
```

And in the JUnit window, an error will appear and this message will be displayed:

```
java.lang.AssertionError: expected: but was:  
at com.javacodegeeks.junit.CalculateTest.testSum(CalculateTest.java:16)
```

## Unit annotations

In this section we will mention the basic annotations supported in JUnit 4. The table below presents a summary of those annotations:

Annotation	Description
<b>@Test</b> <code>public void method()</code>	The <code>Test</code> annotation indicates that the public void method to which it is attached can be executed as a test case.
<b>@Before</b> <code>public void method()</code>	The <code>Before</code> annotation indicates that this method must be executed before each test in the class so as to execute some preconditions necessary for the test.
<b>@BeforeClass</b> <code>public static void method()</code>	The <code>BeforeClass</code> annotation indicates that the static method to which is attached must be executed once and before all tests in the class. That happens when the test methods share computationally expensive setup (e.g. connect to database).
<b>@After</b> <code>public void method()</code>	The <code>After</code> annotation indicates that this method gets executed after execution of each test in the class (e.g. some variables after execution of every test, delete temporary variables etc)
<b>@AfterClass</b> <code>public static void method()</code>	The <code>AfterClass</code> annotation can be used when a method needs to be executed after execution of all the tests in a JUnit Test Case class so as to clean-up the expensive set-up (e.g disconnect from database). Attention: The method attached with this annotation (similar to <code>BeforeClass</code> ) must be defined as static.
<b>@Ignore</b> <code>public static void method()</code>	The <code>Ignore</code> annotation can be used when you want temporarily disable the execution of a test. Every method that is annotated with <code>@Ignore</code> won't be executed.

Let's see an example of a test class with some of the annotations mentioned above.

```
package com.javacodegeeks.junit;

import static org.junit.Assert.*;
import java.util.*;
import org.junit.*;

public class AnnotationsTest {
```

```

private ArrayList testList;

@BeforeClass
public static void onceExecutedBeforeAll() {
    System.out.println("@BeforeClass: onceExecutedBeforeAll");
}

@Before
public void executedBeforeEach() {
    testList = new ArrayList();
    System.out.println("@Before: executedBeforeEach");
}

@AfterClass
public static void onceExecutedAfterAll() {
    System.out.println("@AfterClass: onceExecutedAfterAll");
}

@After
public void executedAfterEach() {
    testList.clear();
    System.out.println("@After: executedAfterEach");
}

@Test
public void EmptyCollection() {
    assertTrue(testList.isEmpty());
    System.out.println("@Test: EmptyArrayList");
}

@Test
public void OneItemCollection() {
    testList.add("oneItem");
    assertEquals(1, testList.size());
    System.out.println("@Test: OneItemArrayList");
}

@Ignore
public void executionIgnored() {

    System.out.println("@Ignore: This execution is ignored");
}
}

```

If we run the above test, the console output would be the following:

```

@BeforeClass: onceExecutedBeforeAll
@Before: executedBeforeEach
@Test: EmptyArrayList
@After: executedAfterEach
@Before: executedBeforeEach
@Test: OneItemArrayList
@After: executedAfterEach
@AfterClass: onceExecutedAfterAll

```

## JUnit assertions

In this section we will present a number of assertion methods. All those methods are provided by the `Assert` class which extends the `class java.lang.Object` and they are useful for writing tests so as to detect failures. In the table below there is a more detailed explanation of the most commonly used assertion methods.

Assertion	Description
<code>void assertEquals([String message], expected value, actual value)</code>	Asserts that two values are equal. Values might be type of int, byte, char or java.lang.Object. The first argument is an optional message.
<code>void assertTrue([String message], boolean condition)</code>	Asserts that a condition is true.
<code>void assertFalse([String message],boolean condition)</code>	Asserts that a condition is false.
<code>void assertNotNull([String message], java.lang.Object object)</code>	Asserts that an object is not null.
<code>void assertNull([String message], java.lang.Object object)</code>	Asserts that an object is null.
<code>void assertEquals([String message], java.lang.Object expected, java.lang.Object actual)</code>	Asserts that the two objects refer to the same object.
<code>void assertNotSame([String message], java.lang.Object unexpected, java.lang.Object actual)</code>	Asserts that the two objects do not refer to the same object.

```
void assertEquals([String message],
expectedArray, resultArray)
```

Asserts that the array expected and the resulted array are equal.  
Array might be int, long, short, char, byte or java.lang.Object.

Let's see an example of some of the aforementioned assertions.

```
package com.javacodegeeks.junit;

import static org.junit.Assert.*;
import org.junit.Test;

public class AssertionsTest {

    @Test
    public void test() {
        String obj1 = "junit";
        String obj2 = "junit";
        String obj3 = "test";
        String obj4 = "test";
        String obj5 = null;
        int var1 = 1;
        int var2 = 2;
        int[] arithmetic1 = { 1, 2, 3 };
        int[] arithmetic2 = { 1, 2, 3 };

        assertEquals(obj1, obj2);

        assertSame(obj3, obj4);

        assertNotSame(obj2, obj4);

        assertNotNull(obj1);

        assertNull(obj5);

        assertTrue(var1 == var2);

        assertEquals(arithmetic1, arithmetic2);
    }
}
```

In the class above we can see how these assert methods work.

- The `assertEquals()` method will return normally if the two compared objects are equal, otherwise a failure will be displayed in the JUnit window and the test will abort.
- The `assertSame()` and `assertNotSame()` methods tests if two object references point to exactly the same object.
- The `assertNotNull()` and `assertNotNull()` methods test whether a variable is null or not null.

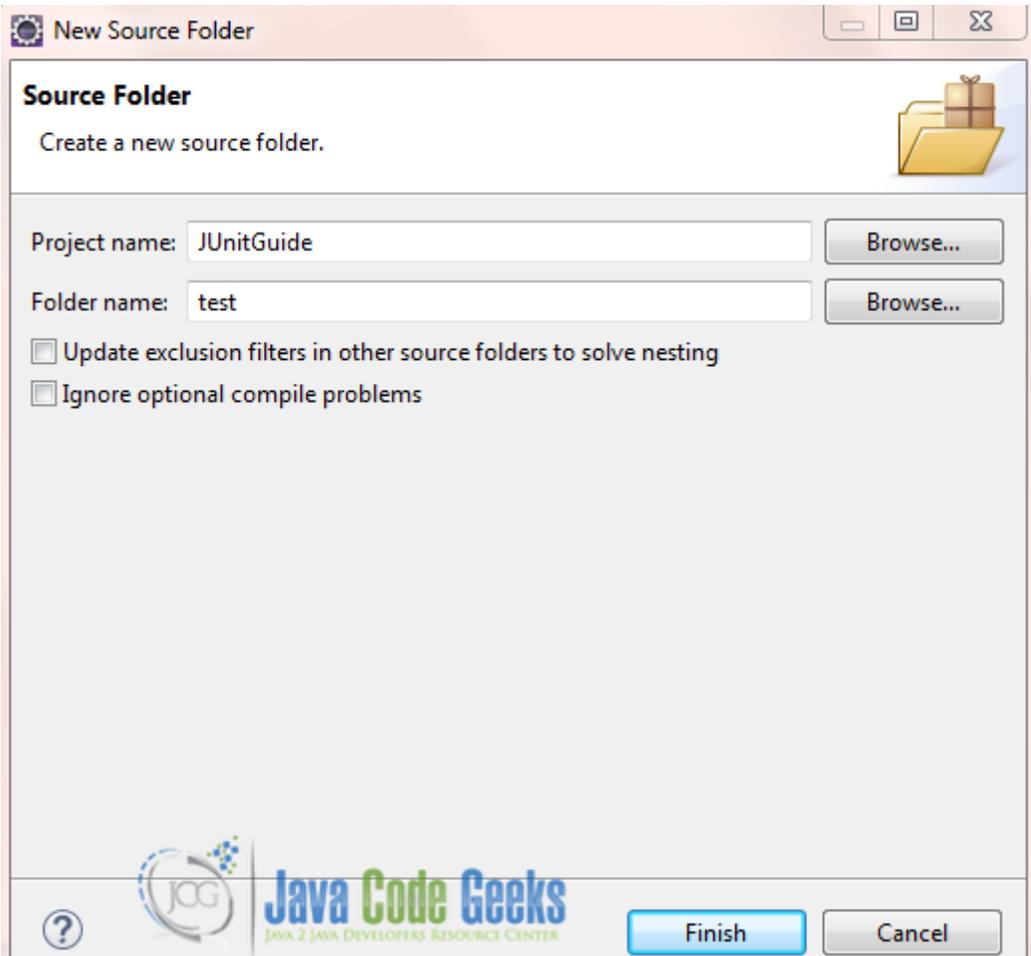
- The `assertTrue()` and `assertFalse()` methods tests if a condition or a variable is true or false.
- The `assertArrayEquals()` will compare the two arrays and if they are equal, the method will proceed without errors. Otherwise, a failure will be displayed in the JUnit window and the test will abort.

## JUnit complete example using Eclipse

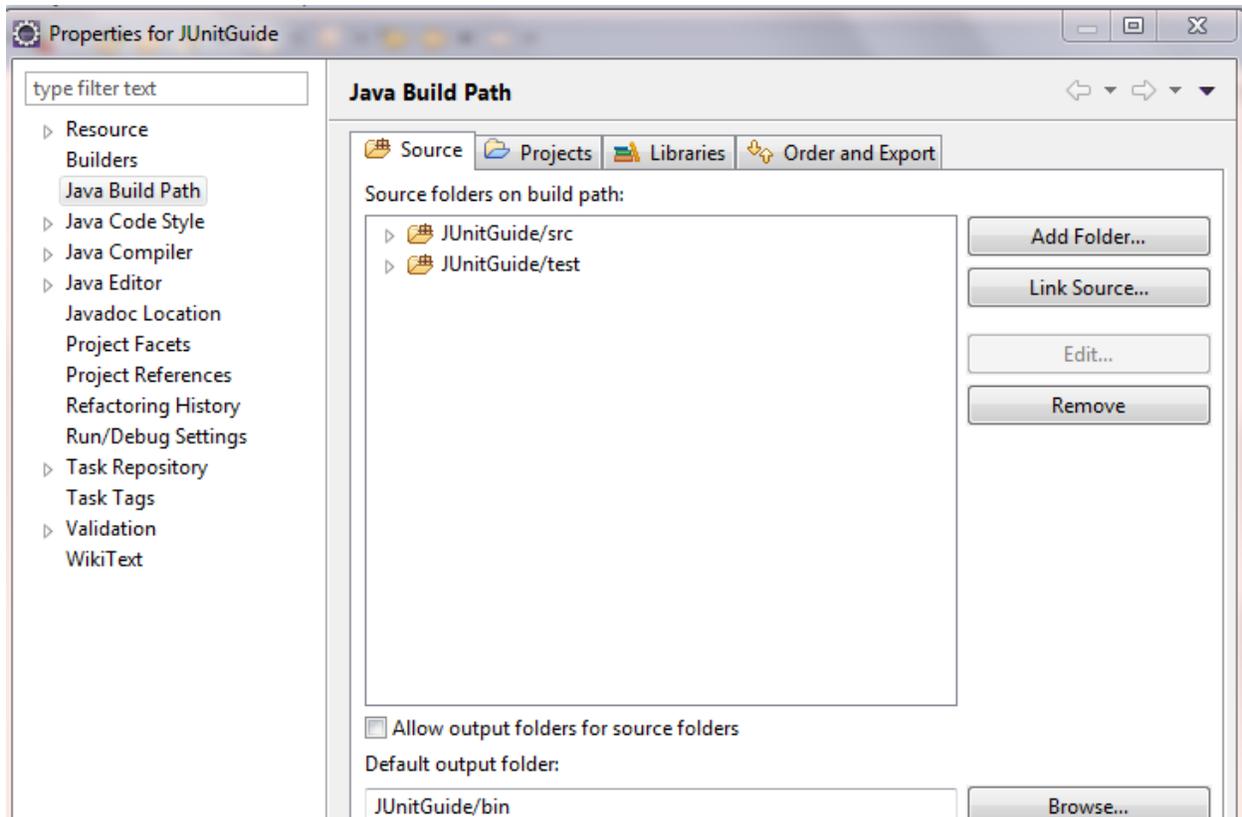
In this section we will show a complete example of using JUnit. We will see in detail how to create and run tests and we will show how to use specific annotations and assertions of JUnit.

### Initial Steps

Let's create a java project named *JUnitGuide*. In the *src* folder, we right-click and select *New -> Package*, so as to create a new package named `com.javacodegeeks.junit` where we will locate the class to be tested. For the test classes, it is considered as good practice to create a new source folder dedicated to tests, so that the classes to be tested and the test classes will be in different source folders. For this purpose, right-click your project, select *New -> Source Folder*, name the new source folder *test* and click *Finish*.



You can easily see that there are two source folders in your project:



You can also create a new package in the newly created test folder, which will be called `com.javacodegeeks.junit`, so that your test classes won't be located to the default package and we are ready to start!

### Create the java class to be tested

Right-click the `src` folder and create a new java class called `FirstDayAtSchool.java`. This will be the class whose public methods will be tested.

```
package com.javacodegeeks.junit;

import java.util.Arrays;

public class FirstDayAtSchool {

    public String[] prepareMyBag() {
        String[] schoolbag = { "Books", "Notebooks", "Pens" };
        System.out.println("My school bag contains: "
            + Arrays.toString(schoolbag));
        return schoolbag;
    }

    public String[] addPencils() {
        String[] schoolbag = { "Books", "Notebooks", "Pens", "Pencils"
    };
};
```

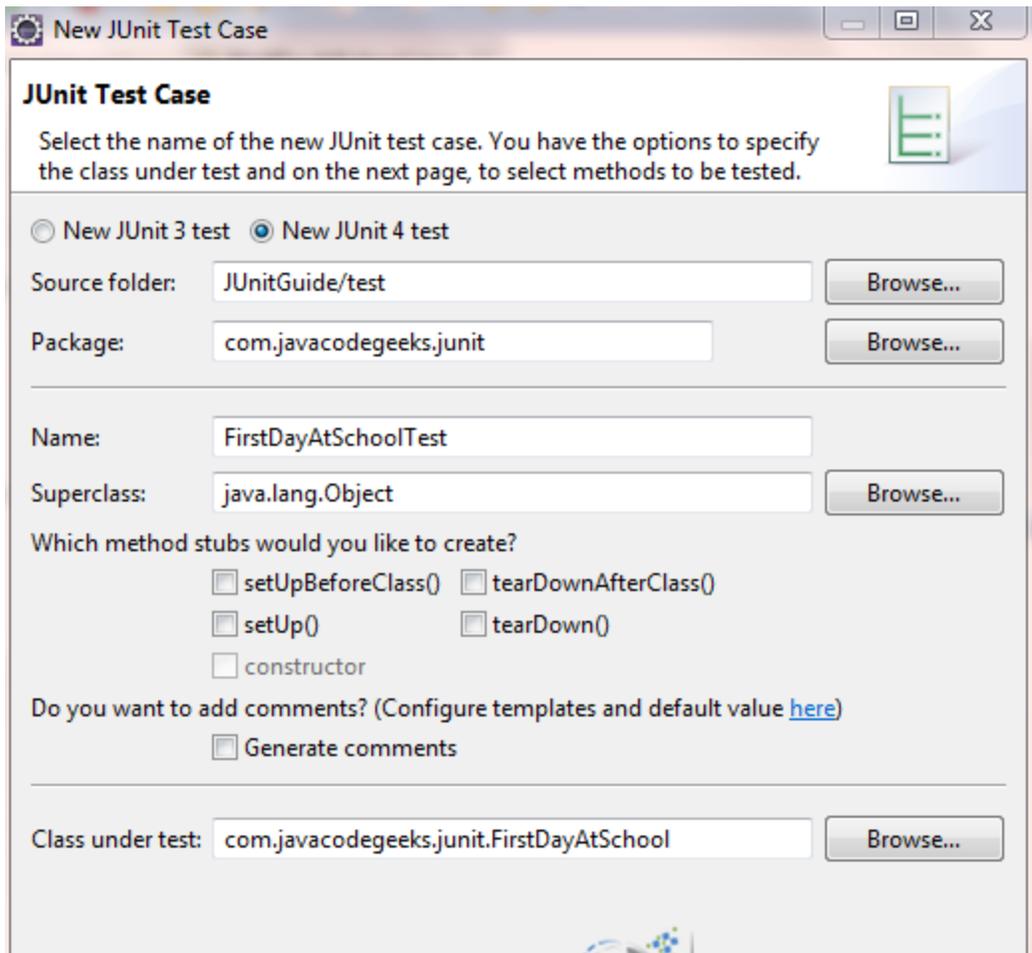
```

        System.out.println("Now my school bag contains: "
            + Arrays.toString(schoolbag));
        return schoolbag;
    }
}

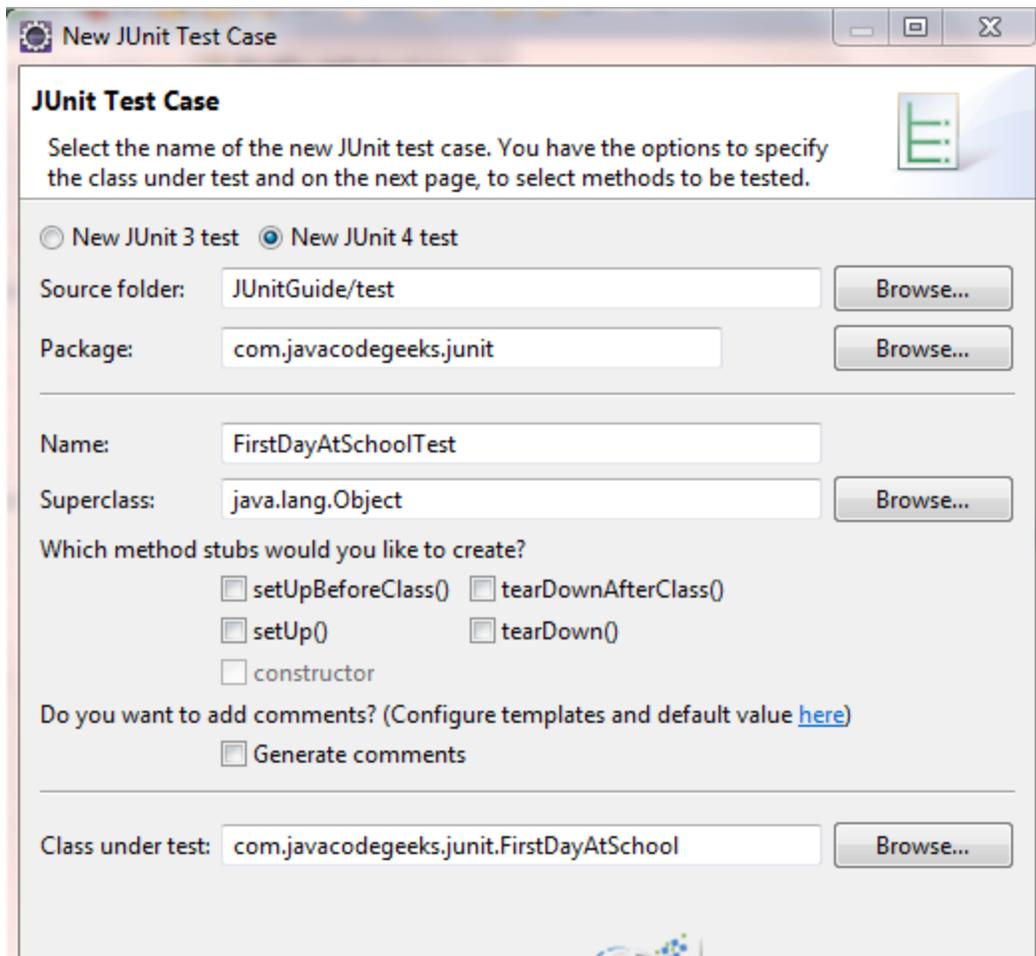
```

### Create and run a JUnit test case

To create a JUnit test case for the existing class `FirstDayAtSchool.java`, right-click on it in the Package Explorer view and select *New* → *JUnit Test Case*. Change the source folder so that the class will be located to *test* source folder and ensure that the flag *New JUnit4 test* is selected.



Then, click *Finish*. If your project does not contain the JUnit library in its classpath, the following message will be displayed so as to add the JUnit library to the classpath:



Below, there is the code of the class named `FirstDayAtSchoolTest.java`, which is our test class:

```
package com.javacodegeeks.junit;

import static org.junit.Assert.*;
import org.junit.Test;

public class FirstDayAtSchoolTest {

    FirstDayAtSchool school = new FirstDayAtSchool();
    String[] bag1 = { "Books", "Notebooks", "Pens" };
    String[] bag2 = { "Books", "Notebooks", "Pens", "Pencils" };

    @Test
    public void testPrepareMyBag() {
        System.out.println("Inside testPrepareMyBag()");
        assertEquals(bag1, school.prepareMyBag());
    }

    @Test
    public void testAddPencils() {
```

```

        System.out.println("Inside testAddPencils()");
        assertEquals(bag2, school.addPencils());
    }
}

```

Now we can run the test case by right-clicking on the test class and select *Run As -> JUnit Test*.

The program output will look like that:

```

Inside testPrepareMyBag()
My school bag contains: [Books, Notebooks, Pens]
Inside testAddPencils()
Now my school bag contains: [Books, Notebooks, Pens, Pencils]

```

and in the JUnit view will be no failures or errors. If we change one of the arrays, so that it contains more than the expected elements:

```
String[] bag2 = { "Books", "Notebooks", "Pens", "Pencils", "Rulers"};
```

and we run again the test class, the JUnit view will contain a failure:

Finished after 0.027 seconds

Runs: 2/2    Errors: 0    Failures: 1

- com.javacodegeeks.junit.FirstDayAtSchoolTest [Runner: JUnit 4] (0.000 s)
  - testPrepareMyBag (0.000 s)
  - testAddPencils (0.000 s)

Failure Trace

```

java.lang.AssertionError: array lengths differed, expected.length=5 actual.length=4
at com.javacodegeeks.junit.FirstDayAtSchoolTest.testAddPencils(FirstDayAtSchoolTest.java:22)

```

Else, if we change again one of the arrays, so that it contains a different element than the expected:

```
String[] bag1 = { "Books", "Notebooks", "Rulers" };
```

and we run again the test class, the JUnit view will contain once again a failure:

Package Explorer JUnit

Finished after 0.044 seconds

Runs: 2/2 Errors: 0 Failures: 1

com.javacodegeeks.junit.FirstDayAtSchoolTest [Runner: JUnit 4] (0.000 s)

- testPrepareMyBag (0.000 s)
- testAddPencils (0.000 s)

Failure Trace

arrays first differed at element [2]; expected:<[Ruler]s> but was:<[Pen]s>

at com.javacodegeeks.junit.FirstDayAtSchoolTest.testPrepareMyBag(FirstDayAtSchoolTest.java:16)

Java Code Geeks  
JAVA 2 JAVA DEVELOPERS RESOURCE CENTER

### Using @Ignore annotation

Let's see in the above example how can we use the `@Ignore` annotation. In the test class `FirstDayAtSchoolTest` we will add the `@Ignore` annotation to the `testAddPencils()` method. In that way, we expect that this testing method will be ignored and won't be executed.

```
package com.javacodegeeks.junit;  
  
import static org.junit.Assert.*;  
  
import org.junit.Ignore;  
import org.junit.Test;
```

```

public class FirstDayAtSchoolTest {

    FirstDayAtSchool school = new FirstDayAtSchool();
    String[] bag1 = { "Books", "Notebooks", "Pens" };
    String[] bag2 = { "Books", "Notebooks", "Pens", "Pencils" };

    @Test
    public void testPrepareMyBag() {
        System.out.println("Inside testPrepareMyBag()");
        assertEquals(bag1, school.prepareMyBag());
    }

    @Ignore
    @Test
    public void testAddPencils() {
        System.out.println("Inside testAddPencils()");
        assertEquals(bag2, school.addPencils());
    }

}

```

Indeed, this is what happens according to the output:

```

Inside testPrepareMyBag()
My school bag contains: [Books, Notebooks, Pens]

```

Now, we will remove the `@Ignore` annotation from the `testAddPencils()` method and we will annotate the whole class instead.

```

package com.javacodegeeks.junit;

import static org.junit.Assert.*;

import org.junit.Ignore;
import org.junit.Test;

@Ignore
public class FirstDayAtSchoolTest {

    FirstDayAtSchool school = new FirstDayAtSchool();
    String[] bag1 = { "Books", "Notebooks", "Pens" };
    String[] bag2 = { "Books", "Notebooks", "Pens", "Pencils" };

    @Test
    public void testPrepareMyBag() {
        System.out.println("Inside testPrepareMyBag()");
        assertEquals(bag1, school.prepareMyBag());
    }

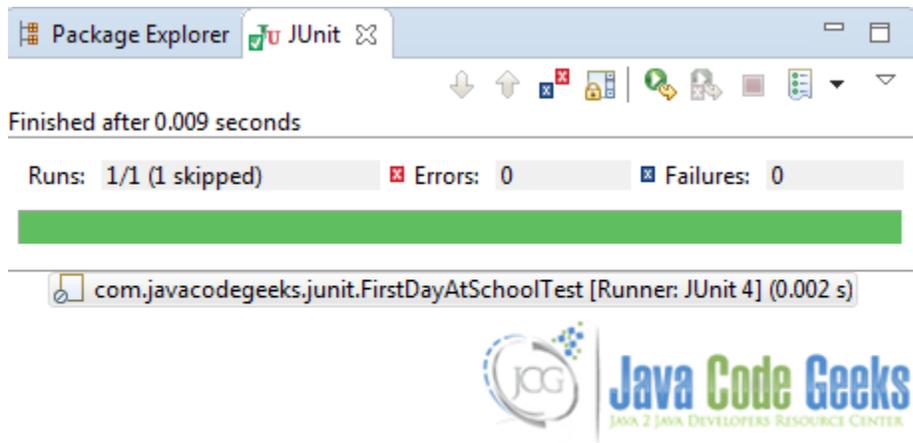
    @Test
    public void testAddPencils() {
        System.out.println("Inside testAddPencils()");
        assertEquals(bag2, school.addPencils());
    }

}

```

```
}
```

The whose test class won't be executed, so no result will be displayed in the console output and in the junit view:



## Creating suite tests

In this section, we will see how to create suite tests. A test suite is a collection of some test cases from different classes that can be run all together using `@RunWith` and `@Suite` annotations. This is very helpful if you have many test classes and you want to run them all together instead of running each test one at a time. When a class is annotated with `@RunWith`, JUnit will invoke the class in which is annotated so as to run the tests, instead of using the runner built into JUnit.

Based on the classes of the previous sections, we can create two test classes. The one class will test the public method `prepareMyBag()` and the other test class will test the method `addPencils()`. Hence, we will eventually have the classes below:

```
package com.javacodegeeks.junit;

import org.junit.Test;
import static org.junit.Assert.*;

public class PrepareMyBagTest {

    FirstDayAtSchool school = new FirstDayAtSchool();

    String[] bag = { "Books", "Notebooks", "Pens" };

    @Test
    public void testPrepareMyBag() {

        System.out.println("Inside testPrepareMyBag()");
        assertEquals(bag, school.prepareMyBag());
    }
}
```

```

    }

}
package com.javacodegeeks.junit;

import org.junit.Test;
import static org.junit.Assert.*;

public class AddPencilsTest {

    FirstDayAtSchool school = new FirstDayAtSchool();

    String[] bag = { "Books", "Notebooks", "Pens", "Pencils" };

    @Test
    public void testAddPencils() {

        System.out.println("Inside testAddPencils()");
        assertEquals(bag, school.addPencils());

    }

}

```

Now we will create a test suite so as to run the above classes together. Right-click the *test* source folder and create a new java class named `SuiteTest.java` with the following code:

```

package com.javacodegeeks.junit;

import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)
@Suite.SuiteClasses({ PrepareMyBagTest.class, AddPencilsTest.class })
public class SuitTest {

}

```

With the `@Suite.SuiteClasses` annotation you can define which test classes will be included in the execution.

So, if you right-click the test suite and select *Run As -> JUnit Test*, the execution of both test classes will take place with the order that has been defined in the `@Suite.SuiteClasses` annotation.

## Creating parameterized tests

In this section we will see how to create parameterized tests. For this purpose, we will use the class mentioned in section 2.1 which provides a public method for adding integers. So, this will be the class to be tested.

But when a test class can be considered as a parameterized test class? Of course, when it fulfills all the following requirements:

- The class is annotated with `@RunWith(Parameterized.class)`.  
As explained in the previous section, `@RunWith` annotation enables JUnit to invoke the class in which is annotated to run the tests, instead of using the runner built into JUnit. `Parameterized` is a runner inside JUnit that will run the same test case with different set of inputs.
- The class has a single constructor that stores the test data.
- The class has a static method that generates and returns test data and is annotated with the `@Parameters` annotation.
- The class has a test, which obviously means that it needs a method annotated with the `@Test` annotation.

Now, we will create a new test class named `CalculateTest.java`, which will follow the guidelines mentioned above. The source code of this class follows.

## Creating parameterized tests

In this section we will see how to create parameterized tests. For this purpose, we will use the class mentioned in section 2.1 which provides a public method for adding integers. So, this will be the class to be tested.

But when a test class can be considered as a parameterized test class? Of course, when it fulfills all the following requirements:

- The class is annotated with `@RunWith(Parameterized.class)`.  
As explained in the previous section, `@RunWith` annotation enables JUnit to invoke the class in which is annotated to run the tests, instead of using the runner built into JUnit. `Parameterized` is a runner inside JUnit that will run the same test case with different set of inputs.
- The class has a single constructor that stores the test data.
- The class has a static method that generates and returns test data and is annotated with the `@Parameters` annotation.
- The class has a test, which obviously means that it needs a method annotated with the `@Test` annotation.

Now, we will create a new test class named `CalculateTest.java`, which will follow the guidelines mentioned above. The source code of this class follows.

```

package com.javacodegeeks.junit;

import static org.junit.Assert.assertEquals;
import java.util.Arrays;
import java.util.Collection;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;

@RunWith(Parameterized.class)
public class CalculateTest {

    private int expected;
    private int first;
    private int second;

    public CalculateTest(int expectedResult, int firstNumber, int
secondNumber) {
        this.expected = expectedResult;
        this.first = firstNumber;
        this.second = secondNumber;
    }

    @Parameters
    public static Collection addedNumbers() {
        return Arrays.asList(new Integer[][] { { 3, 1, 2 }, { 5, 2, 3
},
                { 7, 3, 4 }, { 9, 4, 5 }, });
    }

    @Test
    public void sum() {
        Calculate add = new Calculate();
        System.out.println("Addition with parameters : " + first + "
and "
                + second);
        assertEquals(expected, add.sum(first, second));
    }
}

```

As we can observe in the class above, it fulfills all the above requirements. The method `addedNumbers` annotated with `@Parameters` returns a Collection of Arrays. Each array includes the inputs/output numbers of each test execution. The number of elements in each array must be the same with the number of parameters in the constructor. So, in this specific case, each array includes three elements, two elements that represent the numbers to be added and one element for the result.

If we run the `CalculateTest` test case, the console output will be the following:

```

Addition with parameters : 1 and 2
Adding values: 1 + 2
Addition with parameters : 2 and 3

```

```
Adding values: 2 + 3
Addition with parameters : 3 and 4
Adding values: 3 + 4
Addition with parameters : 4 and 5
Adding values: 4 + 5
```

As we see in the output, the test case is executed four times, which is the number of inputs in the method annotated with `@Parameters` annotation.

## Rules

In this section we present a new feature of JUnit called *Rules* which allows very flexible addition or redefinition of the behavior of each test method in a test class. For this purpose, `@Rule` annotation should be used so as to mark public fields of a test class. Those fields should be of type `MethodRule`, which is an alteration in how a test method is run and reported. Multiple `MethodRules` can be applied to a test method. `MethodRule` interface has a lot of implementations, such as `ErrorCollector` which allows execution of a test to continue after the first problem is found, `ExpectedException` which allows in-test specification of expected exception types and messages, `TestName` which makes the current test name available inside test methods, and many others. Except for those already defined rules, developers can create their own custom rules and use them in their test cases as they wish. Below we present the way we can use one of the existing rules named `TestName` in our own tests. `TestName` is invoked when a test is about to start.

```
package com.javacodegeeks.junit;

import static org.junit.Assert.*;

import org.junit.*;
import org.junit.rules.TestName;

public class NameRuleTest {
    @Rule
    public TestName name = new TestName();

    @Test
    public void testA() {
        System.out.println(name.getMethodName());
        assertEquals("testA", name.getMethodName());
    }

    @Test
    public void testB() {
        System.out.println(name.getMethodName());
        assertEquals("testB", name.getMethodName());
    }
}
```

We can see that the `@Rule` annotation marks the public field `name` which is of type `MethodRule` and specifically, `TestName` type. Then, we can use in our tests this `name` field and find for example the name of the test method, in this specific case.

## Categories

Another new feature of JUnit is called *Categories* and allows you to group certain kinds of tests together and even include or exclude groups (categories). For example, you can separate slow tests from fast tests. To assign a test case or a method to one of those categories the `@Category` annotation is provided. Below there is an example of how we can use this nice feature of JUnit, based on the release notes of [JUnit 4.8](#).

```
public interface FastTests { /* category marker */
}

public interface SlowTests { /* category marker */
}
```

Firstly, we define two categories, `FastTests` and `SlowTests`. A category can be either a class or an interface.

```
public class A {
    @Test
    public void a() {
        fail();
    }

    @Category(SlowTests.class)
    @Test
    public void b() {
    }
}
```

In the above code, we mark the test method `b()` of class `A` with `@Category` annotation so as to indicate that this specific method belongs to category `SlowTests`. So, we are able to mark not only whole classes but also some of their test methods individually.

```
@Category({ SlowTests.class, FastTests.class })
public class B {
    @Test
    public void c() {
    }
}
```

In the above sample of code, we can see that the whole class `B` is annotated with `@Category` annotation. Annotating a test class with `@Category` annotation automatically includes all its test methods in this category. We can also see that a test class or a test method can belong to more than one categories.

```

@RunWith(Categories.class)
@IncludeCategory(SlowTests.class)
@SuiteClasses({ A.class, B.class })
// Note that Categories is a kind of Suite
public class SlowTestSuite {
    // Will run A.b and B.c, but not A.a
}

```

In this sample of code, we notice that there is a suite test named `SlowTestSuite`. Basically, categories are a kind of suite. In this suite, we observe a new annotation called `@IncludeCategory`, indicating which categories will be included in the execution. In this specific case, methods belonging to `SlowTests` category will be executed. Hence, only the test method `b()` of class `A` will be executed as well as the test method `c()` of class `B`, which both belong to `SlowTests` category.

```

@RunWith(Categories.class)
@IncludeCategory(SlowTests.class)
@ExcludeCategory(FastTests.class)
@SuiteClasses({ A.class, B.class })
// Note that Categories is a kind of Suite
public class SlowTestSuite {
    // Will run A.b, but not A.a or B.c
}

```

Finally, we change a little bit the test suite and we add one more new annotation called `@ExcludeCategory`, indicating which categories will be excluded from the execution. In this specific case, only the test method `b()` of class `A` will be executed, as this is the only test method that belongs explicitly to `SlowTests` category. We notice that in both cases, the test method `a()` of class `A` won't be executed as it doesn't belong to any category.

## Run JUnit tests from command line

You can run your JUnit test outside Eclipse, by using the `org.junit.runner.JUnitCore` class. This class provides the `runClasses()` method which allows you to execute one or several test classes. The return type of `runClasses()` method is an object of the type `org.junit.runner.Result`. This object can be used to collect information about the tests. Also, in case there is a failed test, you can use the object `org.junit.runner.notification.Failure` which holds description of the failed tests. The procedure below shows how to run your test outside Eclipse. Create a new Java class named `JUnitRunner.java` with the following code:

```

package com.javacodegeeks.junit;

import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class JunitRunner {

    public static void main(String[] args) {

        Result result = JUnitCore.runClasses(AssertionsTest.class);
        for (Failure fail : result.getFailures()) {
            System.out.println(fail.toString());
        }
        if (result.wasSuccessful()) {
            System.out.println("All tests finished
successfully...");
        }
    }
}

```

```

package com.javacodegeeks.junit;

import org.junit.runner.JUnitCore;
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;

public class JunitRunner {

    public static void main(String[] args) {

        Result result = JUnitCore.runClasses(AssertionsTest.class);
        for (Failure fail : result.getFailures()) {
            System.out.println(fail.toString());
        }
        if (result.wasSuccessful()) {
            System.out.println("All tests finished
successfully...");
        }
    }
}

```

As an example, we choose to run the `AssertionsTest` test class.

- Open command prompt and move down directories so as to find the directory where the two classes are located.
- Compile the Test class and the Runner class.

```

C:\Users\konstantina\eclipse_luna_workspace\JUnitGuide\test\com\javacodegeeks
\junit>javac -classpath "C:\Users\konstantina\Downloads\junit-

```

```
4.11.jar";"C:\Users\konstantina\Downloads\hamcrest-core-1.3.jar";  
AssertionsTest.java JunitRunner.java
```

*s we did in Eclipse, we should also include [library jars](#) of JUnit to our classpath.*

```
C:\Users\konstantina\eclipse_luna_workspace\JUnitGuide\test\com\javacodegeeks  
\junit>java -classpath "C:\Users\konstantina\Downloads\junit-  
4.11.jar";"C:\Users\konstantina\Downloads\hamcrest-core-1.3.jar"; JunitRunner
```

Here is the output:

```
All tests finished successfully...
```

# Starting With Frameworks

## What is Framework?

A framework is considered to be a combination of set protocols, rules, standards and guidelines that can be integrated or followed as a whole so as to leverage the benefits of the scaffolding provided by the Framework.

### Let us see with Real time examples:

I am sure you have attended a seminar / lecture / conference where the participants was asked to observe the following guidelines -

- Participants should occupy their seat 5 minutes before start of lecture
- Bring along a notebook and pen for note taking.
- Read the abstract so you have an idea of what the presentation will be about.
- **Mobile** Phones should be set on silent
- Use the exit gates at opposite end to the speaker should you require to leave in middle of the lecture.
- Questions will be taken at the end of the session

Do you think you can conduct a seminar **WITHOUT** observing these guidelines????

The answer is a big **YES!** Certainly you can conduct a seminar / lecture / conference / demonstration without above guidelines (in fact some of us will not follow them even though there are laid ... :-)

But if the guidelines are followed it will result in beneficial outcome like reduced audience distraction during lecture and increased participant retention and understanding of the subject matter.

### Real time example 2:

We very often use lifts or elevators. There are a few guidelines those are mentioned within the elevator to be followed and taken care off so as to leverage the maximum benefit and prolonged service from the system.

Thus, the users might have noticed the following guidelines:

- Keep a check on the maximum capacity of the elevator and do not get onto an elevator if the maximum capacity has reached.
- Press the alarm button in case of any emergency or trouble.
- Allow the passenger to get off the elevator if any before entering the elevator and stand clear off the doors.

- In case of fire in the building or if there is any haphazard situation, avoid the use of elevator.
- Do not play or jump inside the elevator.
- Do not smoke inside the elevator.
- Call for the help/assistance if door doesn't open or if the elevator doesn't work at all. Do not try to open the doors forcefully.

There can be many more rules or sets of guidelines. Thus, these guidelines if followed, makes the system more beneficial, accessible, scalable and less troubled for the users.

### **What is Test Automation Framework.**

A set of guidelines like coding standards , test-data handling , object repository treatment etc... which when followed during automation scripting produce beneficial outcomes

### **Advantages of Framework:**

- ❖ Ease of scripting
- ❖ Scalability
- ❖ Modularity
- ❖ Understanbility
- ❖ Process defination
- ❖ Reusability
- ❖ Cost
- ❖ Manitanance
- ❖ Easy reporting
- ❖ Minimal manula intervaention
- ❖ Recovery scenario

### **Types of Test Automation Frameworks:**

These frameworks may differ from each other based on their support to different key factors to do automation like reusability, ease of maintenance etc.

**Linear Scripting : Record & Playback:** It is the simplest of all Frameworks and also know as "**Record & Playback**".In this Framework , Tester manually records each step ( Navigation and User Inputs), Inserts Checkpoints ( Validation Steps) in the first round . He then , Plays back the recorded script in the subsequent rounds.

## Advantages :

- Fastest way to generate script
- Automation expertise not required
- Easiest way to learn the features of the Testing Tool

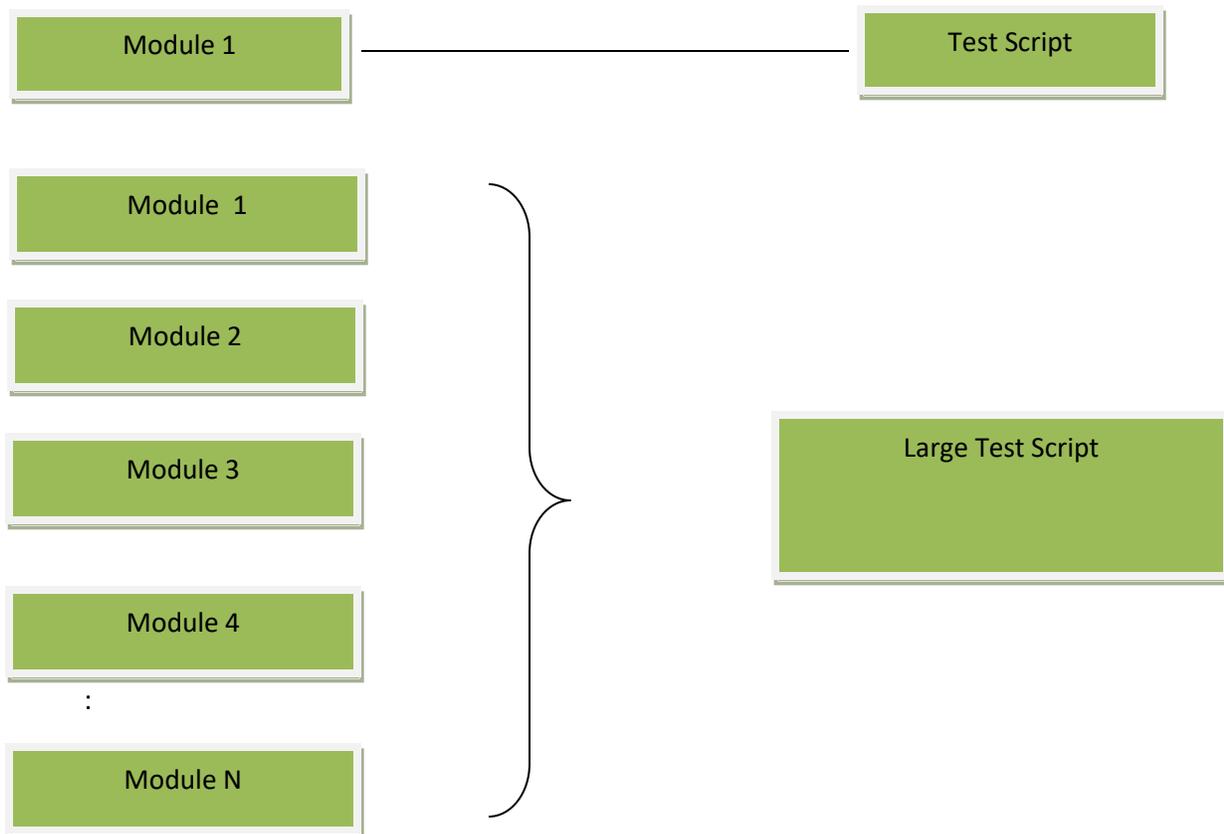
## Disadvantages

- Little reuse of scripts
- Test data is hard coded into the script
- Maintenance Nightmare

## Module Based Testing Framework:

The framework divides the entire "Application Under Test" into number of logical and isolated modules. For each module, we create a separate and independent test script. Thus, when these test scripts taken together builds a larger test script representing more than one modules.

These modules are separated by an abstraction layer in such a way that the changes made in the sections of the application doesn't yields affects on this module.



#### Pros:

1. The framework introduces high level of modularization which leads to easier and cost efficient maintenance.
2. The framework is pretty much scalable
3. If the changes are implemented in one part of the application, only the test script representing that part of the application needs to be fixed leaving all the other parts untouched.

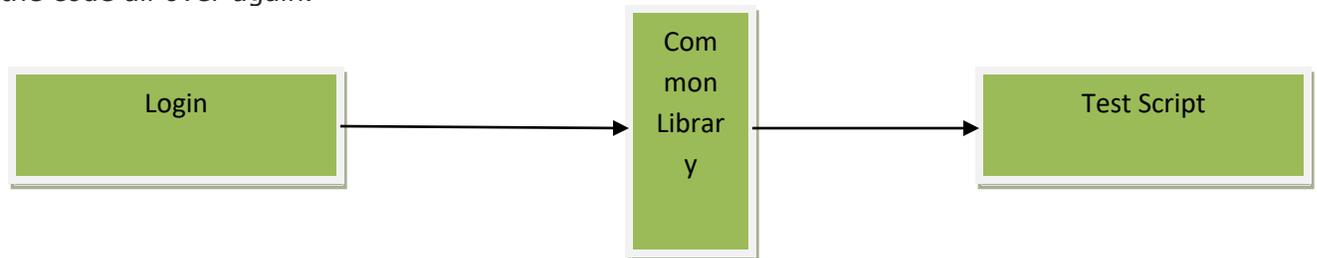
#### Cons:

1. While implementing test scripts for each module separately, we embed the test data (Data with which we are supposed to perform testing) into the test scripts. Thus, whenever we are supposed to test with a different set of test data, it requires the manipulations to be made in the test scripts.

#### Library Architecture Testing Framework

- ❖ It is also known as "**Structured Scripting**" or "**Functional Decomposition**".
- ❖ In this Framework, test scripts are initially recorded by "Record & Playback" method.
- ❖ The Library Architecture Testing Framework is fundamentally and foundationally built on Module Based Testing Framework with some additional advantages.
- ❖ Instead of dividing the application under test into test scripts, we segregate the application into functions
- ❖ Thus we create a common library constituting of common functions for the application under test.
- ❖ Therefore, these libraries can be called within the test scripts whenever required.
- ❖ The basic fundamental behind the framework is to determine the common steps and group them into functions under a library and call those functions in the test scripts whenever required.

**Example:** The login steps can be combined into a function and kept into a library. Thus all the test scripts those require to login the application can call that function instead of writing the code all over again.



### Pros:

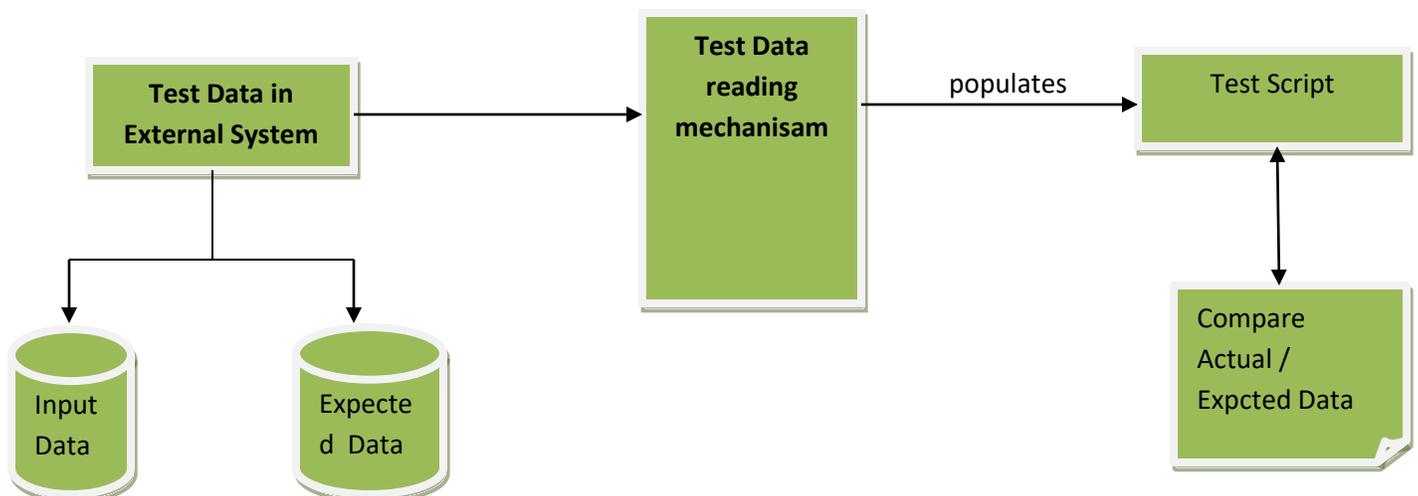
1. Like Module Based Framework, this framework also introduces high level of modularization which leads to easier and cost efficient maintenance and scalability too.
2. As we create common functions that can be efficiently used by the various test scripts across the Framework. Thus, the framework introduces a great degree of re-usability.
3. The automation scripts are less costly to develop due to higher code re-use
4. Easier Script Maintenance

### Cons:

1. Like Module Based Framework, the test data is lodged into the test scripts, thus any change in the test data would require changes in the test script as well.
2. With the introduction of libraries, the framework becomes a little complicated.
3. Technical expertise is necessary to write Scripts using Test Library Framework.
4. More time is needed to plan and prepare test scripts.
5. Test Data is hard coded within the scripts

### Data Driven Testing Framework

- ❖ Testing the same functionality multiples times with the differernt set of input data.
- ❖ Test case logic resides in Test Scripts
- ❖ The Test Data is seperated and kept outside the test scripts.
- ❖ Test data is read from external files(Excel Files, Text Files, CSV files, ODBC sources, DAO Objects, ADO Objects)and are loaded into the variables inside the Test script.
- ❖ Variables are used both for input values and for verification values.



### Pros:

- ❖ Changes to the Test Scripts do not affect the Test Data
- ❖ Test Cases can be executed with multiple Sets of Data
- ❖ A Variety of Test Scenarios can be executed by just varying the Test Data in the External Data File

### Cons:

- ❖ More time is needed to plan and prepare both Test Scripts and Test Data
- ❖ Requires proficiency in a programming language that is being used to develop test scripts.

### Keyword Driven Testing Framework:

- ❖ The Keyword-Driven or Table-Driven framework requires the development of data tables and keywords, **independent of the test automation tool** used to execute them . Tests can be designed with or without the Application. In a keyword-driven test, the functionality of the application-under-test is documented in a table as well as in step-by-step instructions for each test.

There are 3 basis components of a Keyword Driven Framework viz. Keyword , Application Map , Component Function.

### What is a Keyword ?

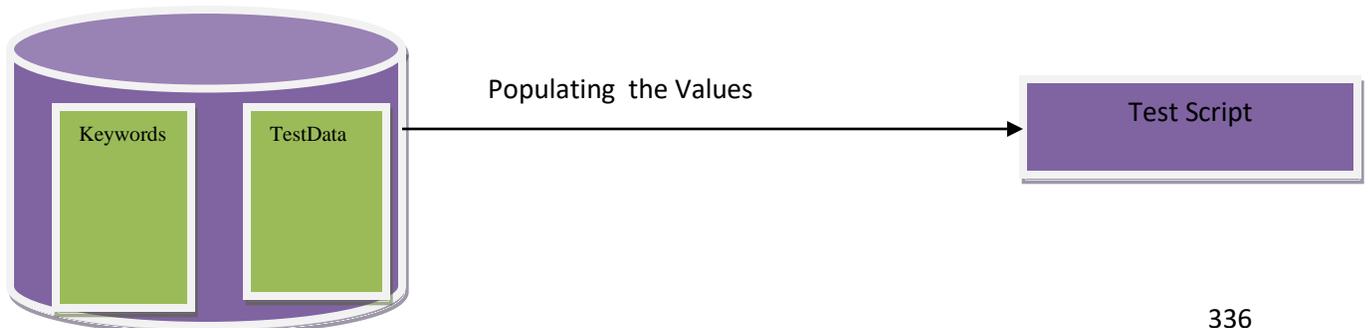
- ❖ Keyword is an Action that can be performed on a GUI Component. Ex . For GUI Component Textbox some Keywords ( Action) would be InputText, VerifyValue, VerifyProperty and so on.

### What is Application Map?

- ❖ An Application Map Provides Named References for GUI Components. Application Maps are nothing but "**Object Repository**"

### What is Component Function?

- ❖ Component Functions are those functions that actively manipulate or interrogate GUI component. An example of a function would be click on web button with all error handling , enter data in a Web Edit with all error handling. Component functions could be application dependent or independent.



### Pros:

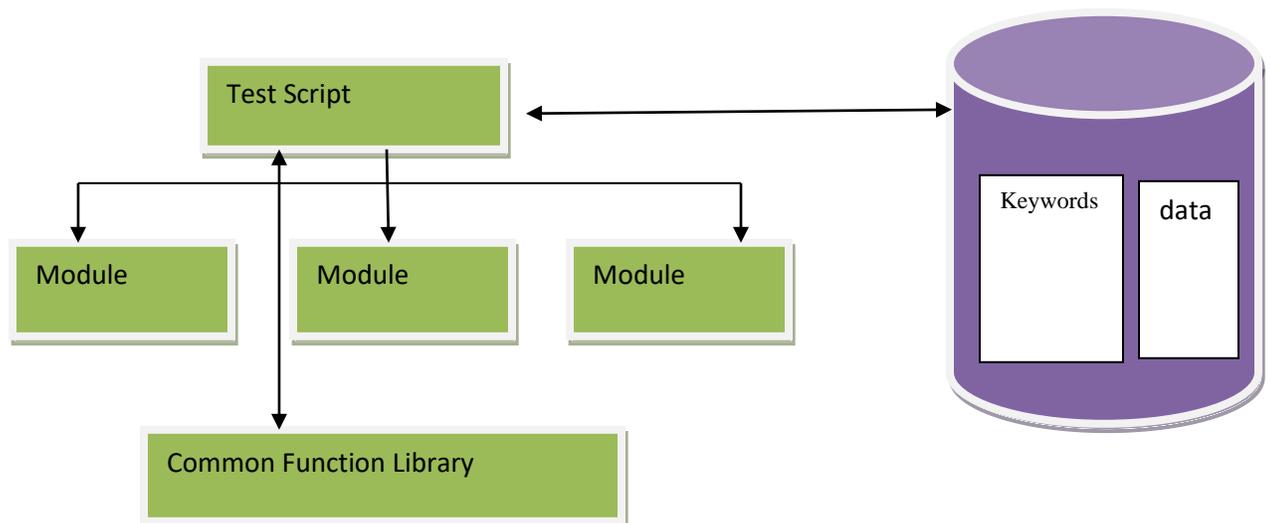
- Provides high code re-usability
- Test Tool Independent
- Independent of Application Under Test, same script works for AUT (with some limitations)
- Tests can be designed with or without AUT
- A single keyword can be used across multiple test scripts.

### Cons:

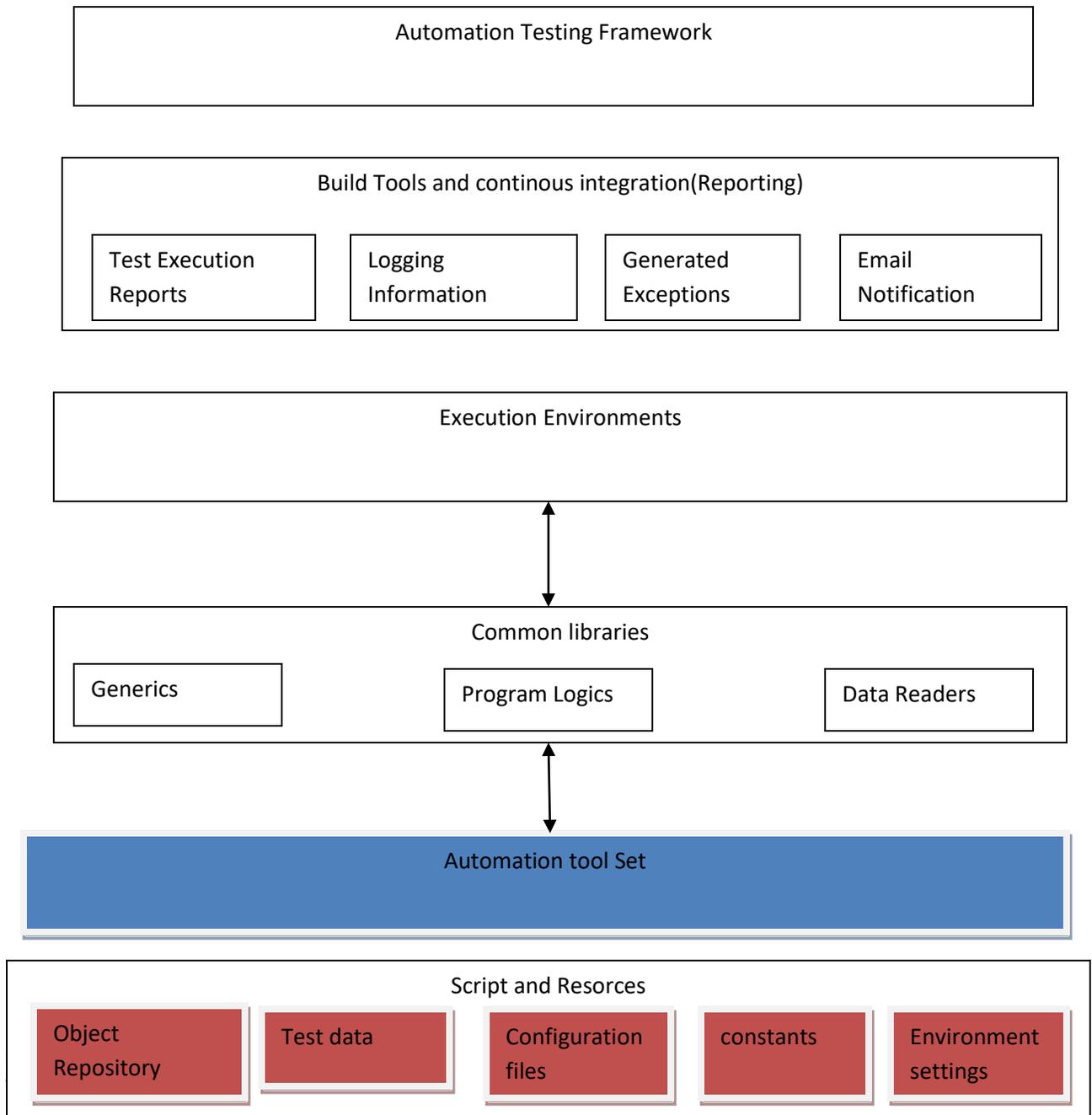
- Initial investment being pretty high, the benefits of this can only be realized if the application is considerably big and the test scripts are to be maintained for quite a few years.
- High Automation expertise is required to create the Keyword Driven Framework.
- The framework becomes complicated gradually as it grows and a number of new keywords are introduced.

### Hybrid Framework:

As the name suggests, the Hybrid Testing Framework is a combination of more than one above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.



Components of Automation Testing Framework:



**Test Data:** The input data with which the scenario would be tested and it can be the expected values with which the actual results would be compared.

**Configuration File/Constants/ Environment Settings:** The file stores the information regarding the application URL, browser specific information etc. It is generally the information that remains static throughout the framework.

**Generics/ Program logics/ Readers:** These are the classes that store the functions which can be commonly used across the entire framework.

**Build tools and Continuous Integration:** These are the tools that aids to the frameworks capabilities to generate test reports, email notifications and logging information.

# Working with Jenkins

## Introduction to Jenkins

### What is Jenkins?

- Jenkins is an open source continuous integration tool which is written in Java
- Jenkins was originally developed as the Hudson project. So if you ever see Hudson then it will be known as Jenkins CI Tool. Hudson's creation started in summer of 2004 at Sun Microsystems. It was first released in java.net in Feb. 2005.

### Jenkins with Selenium

- When your test cases are ready and you want to handover to client or manual testing team so that they can trigger these test cases using single click then Jenkins is the best choice.
- Using Jenkins we can create build (Build – set of Test case combined together) and we can run easily using batch file or Git or build.xml or SVN etc.
- In Jenkins we can schedule the build periodically

Example- You want to run 100 Test case daily at 10 pm then Jenkins will take care of this based on our pattern it will trigger your build

- Email-Notification- Jenkins provide notification mails too once build passed or failed to respective recipients (Depends of configuration)

Before starting if you have be having Eclipse , TestNG and if you are using Excel sheets or csv then these jars should be ready.

## Selenium integration with Jenkins

### Part 1- Download Jenkins

Step 1- Open your web browser and then Navigate to Below URL

<http://jenkins-ci.org> this is the official website of Jenkins

Step 2- Now download Jenkins.war file and save into desktop or any other location depends on your choice

**Meet Jenkins**  
Find out what Jenkins is and get started.

**Use Jenkins**  
See how to get more out of your Jenkins.

### Download Jenkins

Release    Long-Term Support Release

**Java Web Archive (.war)**

**Latest and greatest (1.566)**  
[changelog](#) | [past releases](#) | [RC](#)

upgrading from Hudson?

Or native package

- Windows
- Ubuntu/Debian
- Red Hat Enterprise Linux

Step 3- Once download complete. You will get jenkins.war file that we need to execute

## Part 2-How to configure Jenkins for Selenium

Go to location where Jenkins.war is available. In my case I kept in my project home directory

Name	Date modified	Type	Size
.git	6/1/2014 2:07 AM	File folder	
.settings	6/1/2014 1:49 AM	File folder	
bin	6/4/2014 2:53 AM	File folder	
lib	6/3/2014 12:05 AM	File folder	
src	6/1/2014 1:57 AM	File folder	
testng-xslt	6/3/2014 12:28 AM	File folder	
test-output	6/3/2014 12:18 AM	File folder	
.classpath	6/3/2014 12:15 AM	CLASSPATH File	1 KB
.gitignore	6/1/2014 1:57 AM	GITIGNORE File	1 KB
.project	6/1/2014 1:49 AM	PROJECT File	1 KB
build	11/17/2013 12:53 ...	XML Document	2 KB
jenkins.war	5/13/2014 12:13 AM	WAR File	66,517 KB
testng-results	5/8/2013 9:39 AM	XSL File	58 KB

Step 2- Open Command prompt known as CMD and navigate till project home directory and Start Jenkins server

Start- cmd> Project\_home\_Directory> java -jar jenkins.war

```
java -jar jenkins.war
```

```
Jun 08, 2014 2:21:46 AM jenkins.InitReactorRunner$1 onAttained
INFO: Listed all plugins
Jun 08, 2014 2:21:46 AM jenkins.InitReactorRunner$1 onAttained
INFO: Prepared all plugins
Jun 08, 2014 2:21:46 AM jenkins.InitReactorRunner$1 onAttained
INFO: Started all plugins
Jun 08, 2014 2:21:46 AM jenkins.InitReactorRunner$1 onAttained
INFO: Augmented all extensions
Jun 08, 2014 2:21:52 AM jenkins.InitReactorRunner$1 onAttained
INFO: Loaded all jobs
Jun 08, 2014 2:22:03 AM org.jenkinsci.main.modules.sshd.SSHD start
INFO: Started SSHD at port 30381
Jun 08, 2014 2:22:03 AM jenkins.InitReactorRunner$1 onAttained
INFO: Completed initialization
Jun 08, 2014 2:22:03 AM hudson.TcpSlaveAgentListener <init>
INFO: JNLP slave agent listener started on TCP port 30382
Jun 08, 2014 2:22:04 AM hudson.Main$3 run
INFO: Jenkins is fully up and running
Effective SlaveReactor on master: null
```

### Step 3-

Once Jenkins server is up and running, you will get above success message.

By default Jenkins runs on 8080 port so you can navigate to below url for Jenkins UI

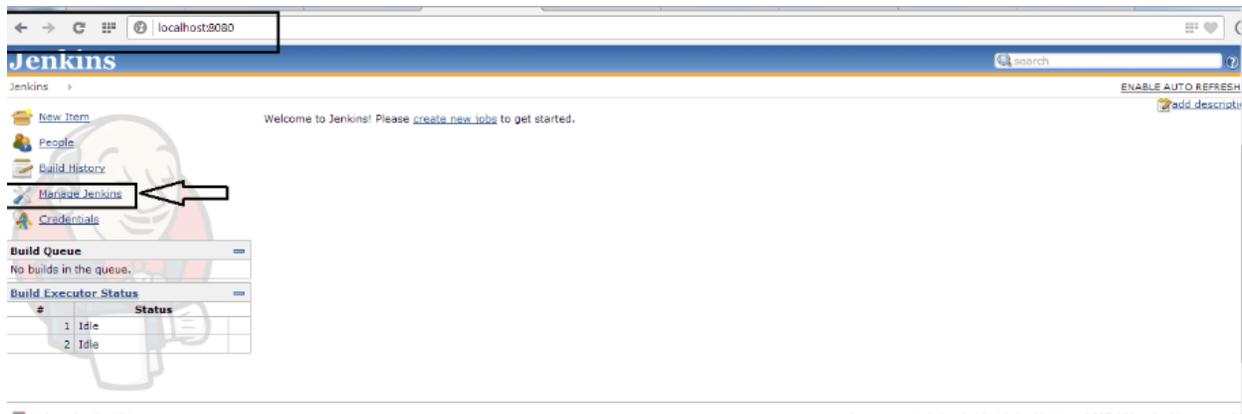
Open any browser and type the url: <http://localhost:8080>

Now Jenkins is up and running so now we have to configure Jenkins so that we can execute our test case via Jenkins.

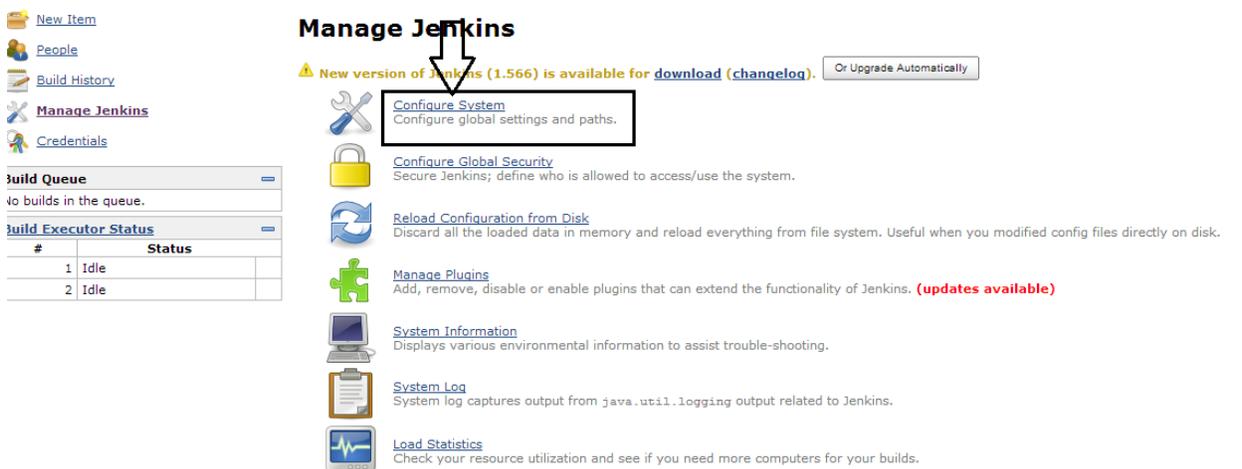
### Step 4-

Once Jenkins is running so we are almost done but before moving to create build we need to configure Jenkins so that Jenkins can identify other tools as well like Java, Maven etc.

Click on > Manage Jenkins



## Click on Configure System



Here we are telling Jenkins that our java is located at this location so we don't have to worry about explicit path.

Navigate to JDK section and Click on Add JDK button

Build History  
Manage Jenkins  
Credentials

**Build Queue**  
No builds in the queue.

**Build Executor Status**

#	Status
1	Idle
2	Idle

# of executors: 2  
Quiet period: 5  
SCM checkout retry count: 0

Restrict project naming

**Global properties**

Environment variables  
 Tool Locations

**Maven Configuration**

Default settings provider: Use default maven settings  
Default global settings provider: Use default maven global settings

**JDK**

JDK installations    
List of JDK installations on this system

Uncheck Install automatically check box so Jenkins will only take java which we have mention above.

**Maven Configuration**

Default settings provider: Use default maven settings  
Default global settings provider: Use default maven global settings

**JDK**

JDK installations

JDK	Name
<input checked="" type="checkbox"/>	<input type="text"/>

**Required**

Install automatically

**Install from java.sun.com**

Version: Java SE Development Kit 8u5

I agree to the Java SE Development Kit License Agreement

**Installing JDK requires Oracle account. Please enter your username/password**

Give the name as JAVA\_HOME and Specify the JDK path

## JDK

JDK installations

JDK Name

JAVA\_HOME

Install automatically

List of JDK installations on this system

## Ant

In Jenkins, we have very good feature that you can configure email notification for user.

This is optional but if you want to configure Email notification then you have to do little setting while configuring Jenkins

Refer below screenshot you can change login details and click on Apply.

E-mail Notification

SMTP server  This is Gmail SMTP server

Default user e-mail suffix  This is suffix will be same for all

Use SMTP Authentication

User Name  This is login details change it accordingly

Password

Use SSL

SMTP Port  This is SMTP port will be same for all

Reply-To Address

Charset

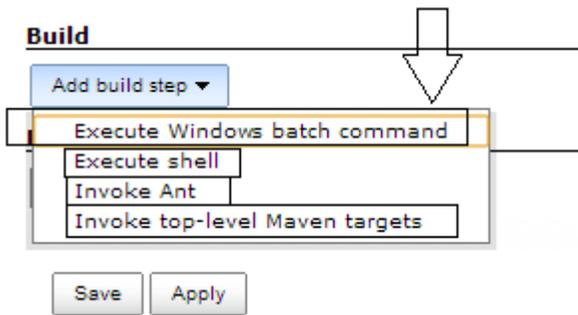
Test configuration by sending test e-mail

Once done click on save and apply.

Congrats, your Jenkins is configured now.

## Part 3- Execute Selenium build using Jenkins

We can execute test-cases in Jenkins using 4 ways refer the below screen-shot



In this post we will execute using Window batch command

### Step 1- Create a batch file first then we will add the same batch file to Jenkins

a-To create batch file we need to set classpath of TestNG so that we can execute testng.xml files  
Our project structure should look like

Name	Date modified	Type	Size
.git	6/8/2014 11:23 PM	File folder	
settings	6/1/2014 1:49 AM	File folder	
bin	6/8/2014 10:34 PM	File folder	
src	6/3/2014 12:05 AM	File folder	
libs	6/8/2014 10:35 PM	File folder	
src	6/8/2014 10:34 PM	File folder	
testng-xsit	6/3/2014 12:28 AM	File folder	
test-output	6/3/2014 12:18 AM	File folder	
.classpath	6/8/2014 10:32 PM	CLASSPATH File	1 KB
.gitignore	6/1/2014 1:57 AM	GITIGNORE File	1 KB
.project	6/1/2014 1:49 AM	PROJECT File	1 KB
build	11/17/2013 12:53 ...	XML Document	2 KB
jenkins.war	5/13/2014 12:13 AM	WAR File	66,517 KB
testng	6/8/2014 11:24 PM	XML Document	1 KB
testng-results	5/8/2013 9:39 AM	XSL File	58 KB

b- Open command prompt and set the classpath-

While setting classpath we will set the path of bin folder and libs folder (inside libs we have all libraries)

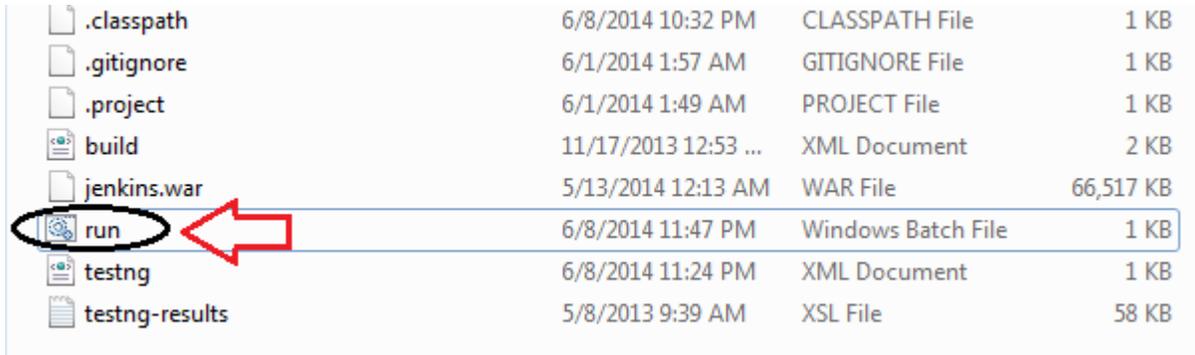
Home directory > set classpath=C:\Users\Learn-automation\bin;C:\Users\Learn-automation\libs\\*;

Note- Please makes the changes as per your system

c- Open notepad and type the below command and save as .bat file –

In my case I have saved as run.bat

```
java -cp bin;libs/* org.testng.TestNG testng.xml
```



File Name	Modified	Type	Size
.classpath	6/8/2014 10:32 PM	CLASSPATH File	1 KB
.gitignore	6/1/2014 1:57 AM	GITIGNORE File	1 KB
.project	6/1/2014 1:49 AM	PROJECT File	1 KB
build	11/17/2013 12:53 ...	XML Document	2 KB
jenkins.war	5/13/2014 12:13 AM	WAR File	66,517 KB
run	6/8/2014 11:47 PM	Windows Batch File	1 KB
testng	6/8/2014 11:24 PM	XML Document	1 KB
testng-results	5/8/2013 9:39 AM	XSL File	58 KB

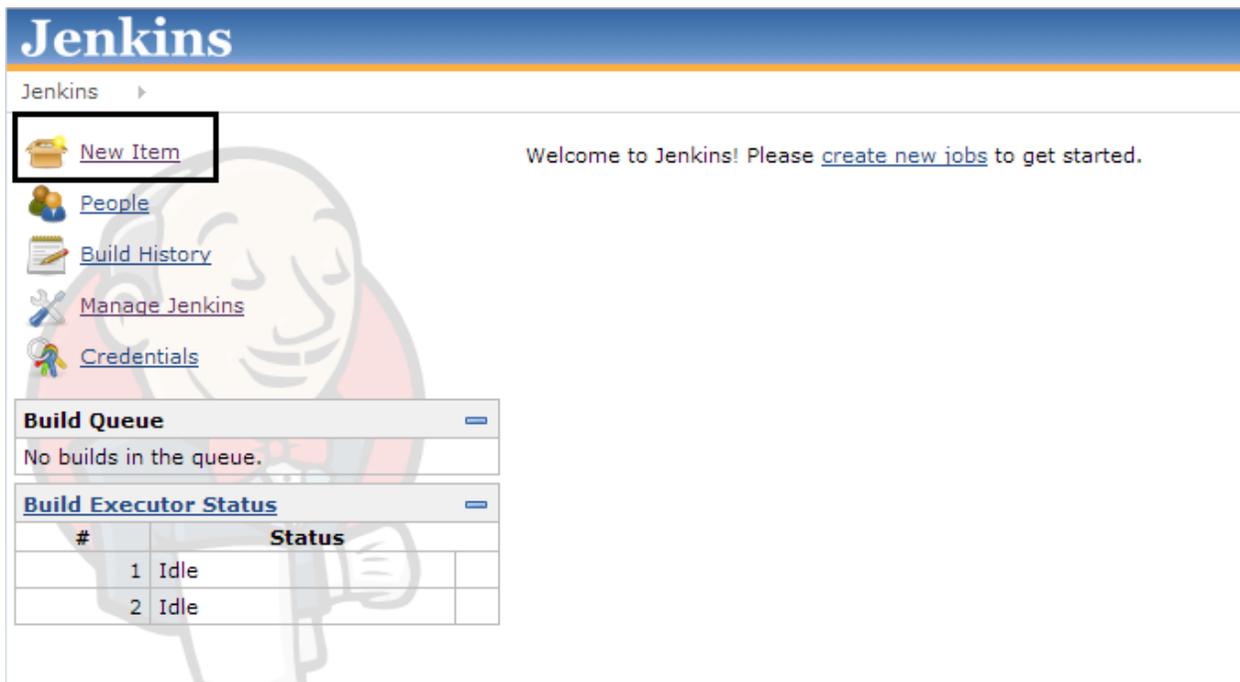
Selenium integration with jenkins

## Step 2-

Create a job in Jenkins which will execute our build

Open Jenkins on browser (type <http://localhost:8080>)

a- Click on new item



Jenkins

Jenkins >

[New Item](#)

Welcome to Jenkins! Please [create new jobs](#) to get started.

[People](#)

[Build History](#)

[Manage Jenkins](#)

[Credentials](#)

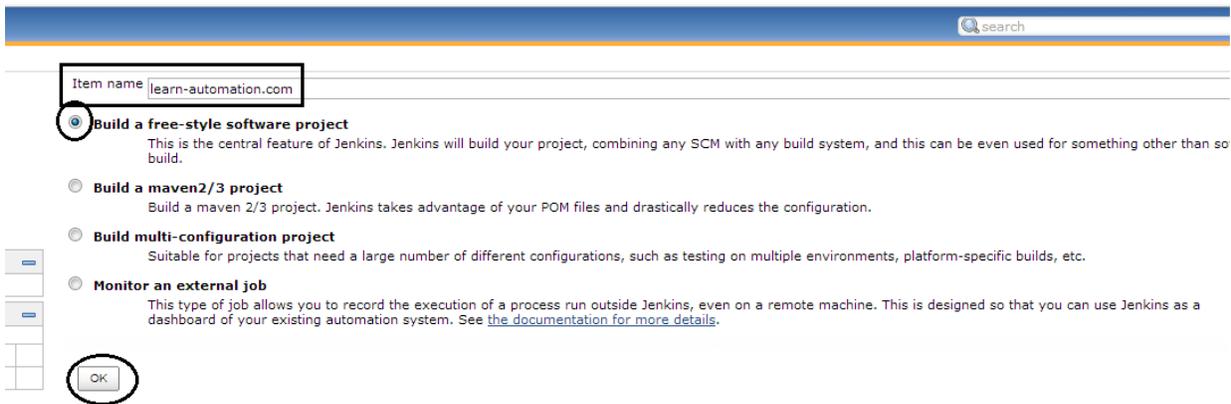
**Build Queue**

No builds in the queue.

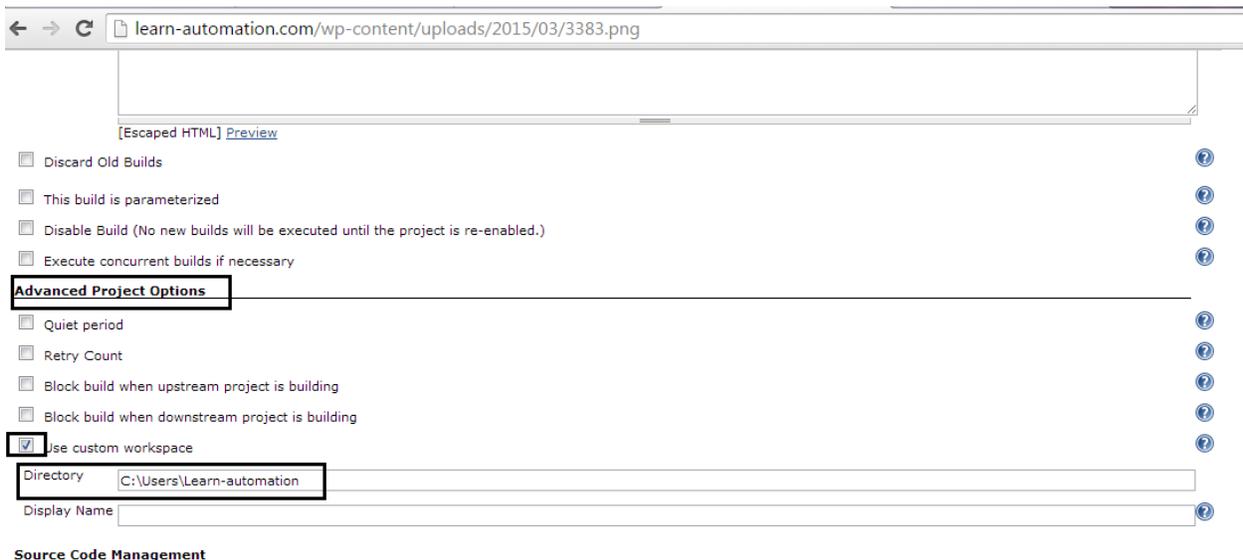
**Build Executor Status**

#	Status
1	Idle
2	Idle

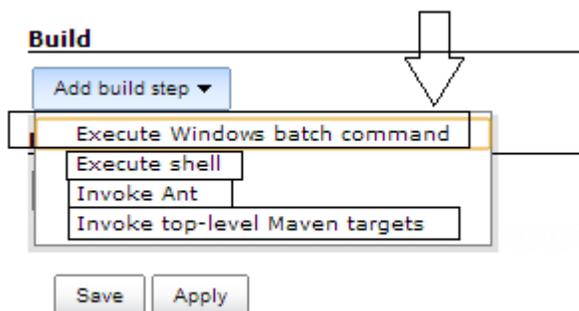
b- Give the Job-Name, select Build a free-style software project and Click on OK button



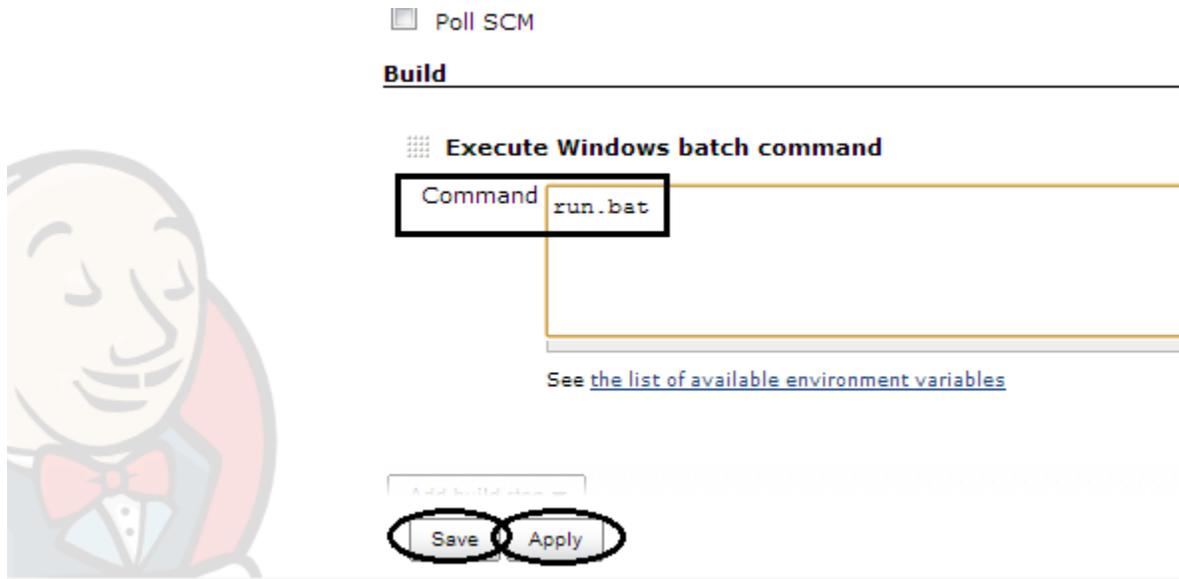
c- Navigate to Advanced Project Options > Check the use custom workspace > in directory we will specify the project home directory



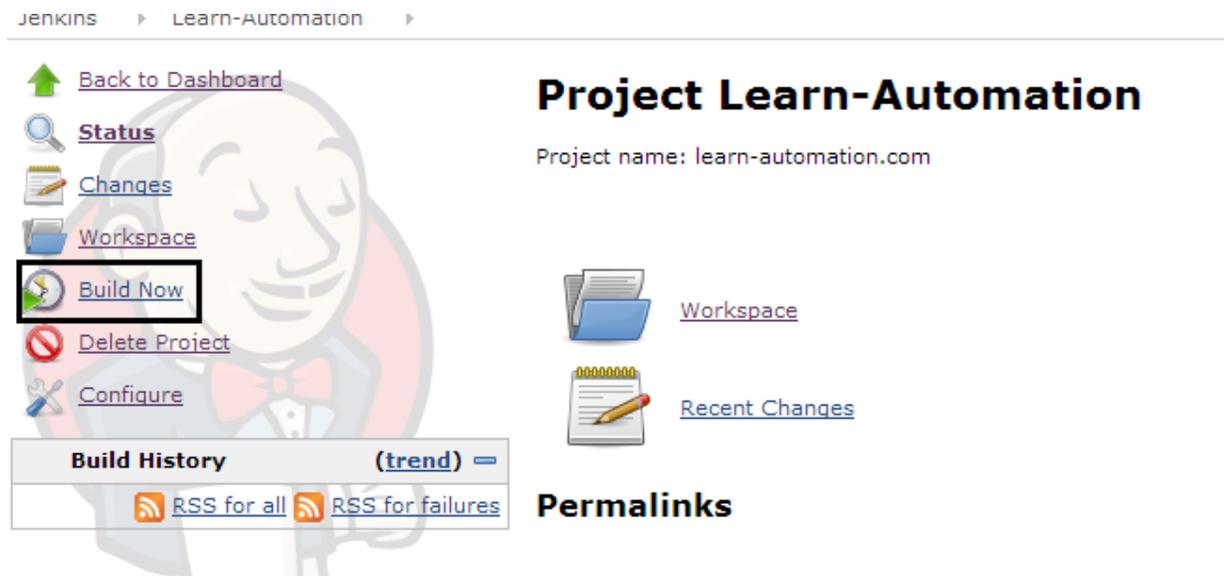
d- Important part now specify the Add Build step > Click on Execute Windows batch command



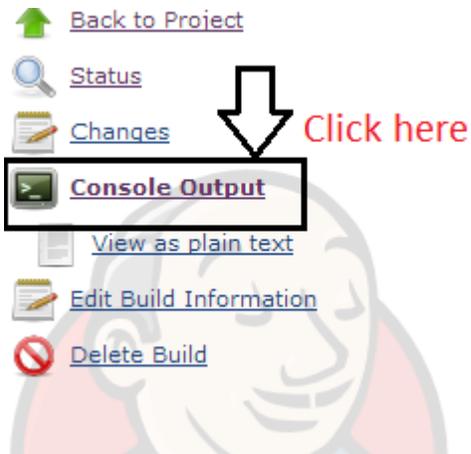
e-In the section please specify the batch file which we created and click on Apply and save



Step 3- you can finally run the Build > Click on Build now option



Step 4- Check Build history and Console output and verify the output



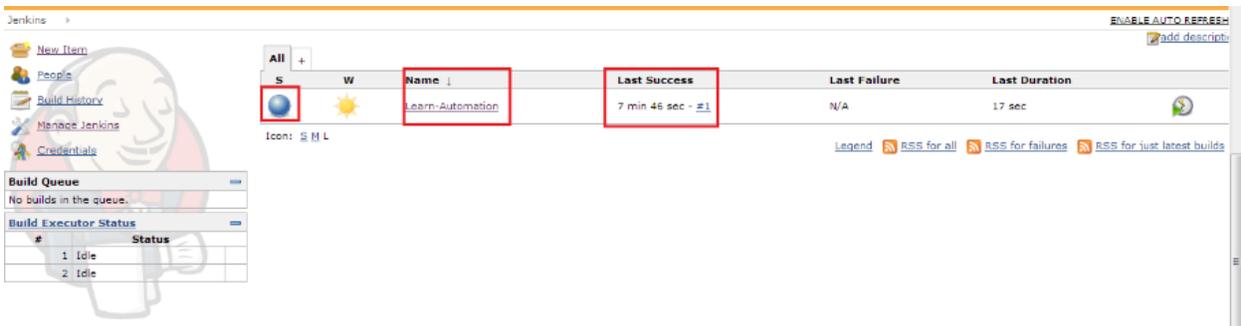
```

Sanity Testcase passed
Smoke Testcase passed
Regression Testcase passed
SIT Testcase passed
UAT Testcase passed
Production Testcase passed

=====
Demo
Total tests run: 6, Failures: 0, Skips: 0
=====

Finished: SUCCESS

```



**Part 4-Schedule your build in Jenkins for periodic execution**

Jenkins comes with very good functionality in which we can schedule jobs which we created. You can schedule build for existing jobs which already created and while creating new project also we can specify the same.

Let's schedule the job. Refer the below screenshot

Step 1-

Open job which we created now and Click on configure > select the check box build periodically



## Step 2-

Specify the time here we need to careful about the syntax

Jenkins works on Cron pattern for more info aboy cron refer cron link <http://en.wikipedia.org/wiki/Cron>

Jenkins will accept 5 parameter lest discuss one by one

\* \* \* \* \*

Here first parameter- specify minute and range will vary from 0-59

Here second parameter- specify hours and range will vary from 0-11

Here third parameter- specify day and range will vary from 0-7 here 0 and 7 will be Sunday

Here fourth parameter- specify month and range will vary from 1-12

Here fifth parameter- specify year so here you can specify \*

Example 1- if you specify 00 22 \* \* \* it means your build will run daily @ 10 PM

Example 2- if you specify 50 \* \* \* \* it means your build will run daily 50 min after every hour

Example 3- if you specify 00 22 1 \* \* it means your build will run every monday @ 10 PM

