**1. INTRODUCING SELENIUM**

**1.1 To Automate or Not to Automate? That is the Question!**

Is automation always advantageous? When should one decide to automate test cases?

It is not always advantageous to automate test cases. There are times when manual testing may be more appropriate. For instance, if the application’s user interface will change considerably in the near future, then any automation would need to be rewritten. Also, sometimes there simply is not enough time to build test automation. For the short term, manual testing may be more effective. If an application has a very tight deadline, there is currently no test automation available, and it’s imperative that the testing get done within that time frame, then manual testing is the best solution.

However, automation has specific advantages for improving the long-term efficiency of a software team’s testing processes. Test automation supports:

* + Frequent regression testing
  + Rapid feedback to developers during the development process
  + Virtually unlimited iterations of test case execution
  + Customized reporting of application defects
  + Support for Agile and extreme development methodologies
  + Disciplined documentation of test cases
  + Finding defects missed by manual testing
  1. **Why should I have to go with selenium**
* Open source
* Easy installation
* Compatibility testing
* Supports parallel execution
* Supports for almost all the operating systems and browsers
* We can implement scripting in any language as we feel
* Logics of the programming languages can be implemented
* Supports OOPs concept

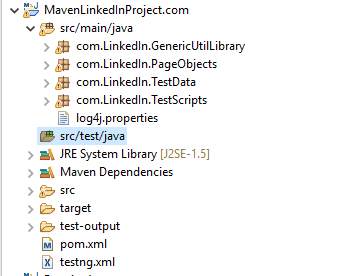
**2. Framework Details**

This section lists and explains about the sample project which has been used for POC and explains the complete framework in details as mentioned below:

**2.1: Maven Project Name:** MavenLinkedInProject.com

This project is created as Maven Project. This project is created with several packages and the complete structure of the project is present as mentioned in the screenshot present in next page. Also, the packages are explained in successive sections.

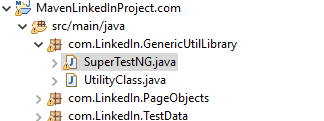
**Screenshot of Project Packages and complete structure:**



**2.2: Details of Utility Package: “com.LinkedIn.GenericUtilLibrary”**

This package will consist of class “SuperTestNG”. This class consists of the steps that are used in every test case as precondition. The package structure and the class details are mentioned below:

**Screenshot of Package:**



Class : SuperTestNG consists of the steps that are re-used in all of the test cases. Some of them are listed below:

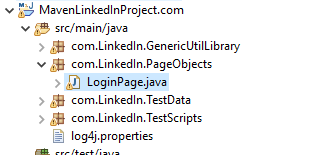
* URL of the application.
* Webdriver initiation : Chrome, IEexplore, Mozilla firefox.
* Steps that are invloved after any methods, like driver to quit, driver to close, printing the logs etc.

Note : Here for example, another class is also created with name “UtilityClass”. This also does the similar steps, however it is completely dependent on user convinient.

**2.2: Details of PageObjects Package: “com.LinkedIn.PageObjects”**

This package will consist of the elements of the pages that are present in application (Repository of the page elements). For each page of an application, the elements of the page can placed in separate class of a page.

Here for the LinkedIn URL, the Login page elements are kept in “LoginPage.Java” class.



The page elements are identified by FindBy annotations and Object identifier. Example of Username page element is shown below for LinkedIn login page:

@FindBy(xpath="//input[@id='username']")

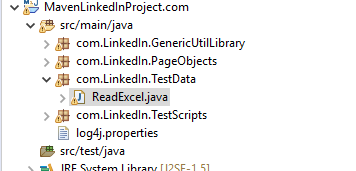
**public** WebElement UserName;

Note : Just like login page webelements, across the application, there will be many pages like Landing page, Notification page, Friends List page. One has to create a separate class for each page and can identify and put the elements as mentioned in above step for usage.

**2.3: Details of Test Data Package: “com.LinkedIn.TestData”**

This package will consist of the steps to read the data from excel sheet by file input stream concepts.

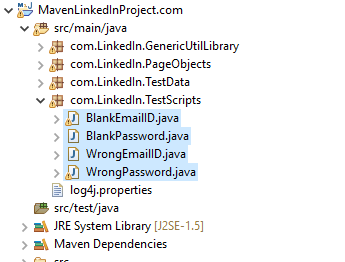
This package needs be imported on the class wherever the test data is needed for testing.



**2.3: Details of Test Scripts Package: “com.LinkedIn.TestScripts”**

This package is the major repository of test cases that are executed against the application under test. The test cases shall be implemented at class level OR at the method level of each class.

Here in this sample project, the test cases are implemented at class level. All the four test cases are run and implemented at class level as shown the below screenshot.



**2.4: POM.xml**

The pom.xml will be created along with the creation of maven project. Here, the dependencies need to be added and maven install step needs to be run to get the dependencies added to the running project.

Sample dependency step is shown below to install the POI dependency for the project. Similarly, the project specific dependency needs to be added.

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>4.0.1</version>

</dependency>

Some of the mandatory dependencies that needs to be added are listed below

* TestNg
* selenium-java
* poi-ooxml

Link for finding the maven dependencies: <https://mvnrepository.com/>

**2.5: TestNG:**

* TestNG is a testing framework that is capable of making Selenium tests easier to understand and of generating reports that are easy to understand.
* The main advantages of TestNG over JUnit are the following.
* Annotations are easier to use and understand.
* Test cases can be grouped more easily.
* TestNG allows us to create parallel tests.
* The Console window in Eclipse generates a text-based result while the TestNG window is more useful because it gives us a graphical output of the test result plus other meaningful details such as:
  + Runtimes of each method.
  + The chronological order by which methods were executed.
* TestNG is capable of generating HTML-based reports.
* Annotations can use parameters just like the usual Java methods.

TestNG shall be run different levels.

1. At package level, it shall be run as mentioned below:

<suite name=*"Suite"*>

<test thread-count=*"5"* name=*"Test"*>

**<packages>**

**<package name=*"com.LinkedIn.TestScripts"*/>**

**</packages>**

</test> <!-- Test -->

</suite> <!-- Suite -->

1. At the class level, it shall be run as mentioned below:

<suite name="Suite">

<test thread-count="5" name="Test">

<classes>

<class name="com.LinkedIn.TestScripts.BlankEmailID"/>

<class name="com.LinkedIn.TestScripts.WrongPassword"/>

<class name="com.LinkedIn.TestScripts.WrongEmailID"/>

<class name="com.LinkedIn.TestScripts.BlankPassword"/>

</classes>

</test> <!-- Test -->

</suite> <!-- Suite -->

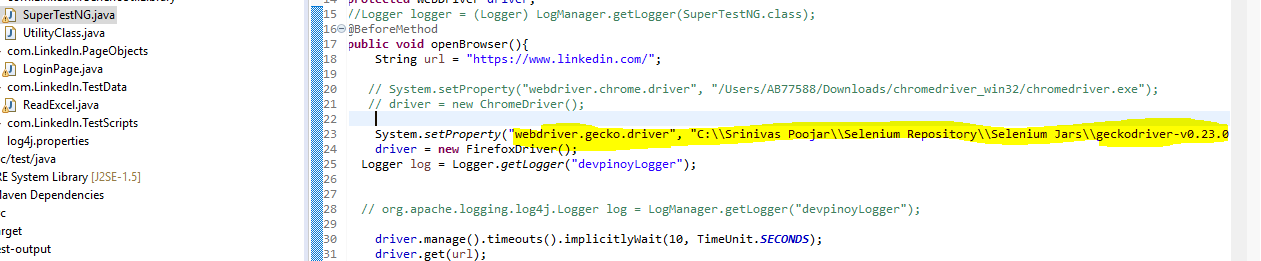
**Installation Steps:**

1. Download the project to your local system



1. Extract the zip file and import same to your eclipse present in your local system.
2. Go to SuperTestNG class and change the system properties to your own system property

Example:



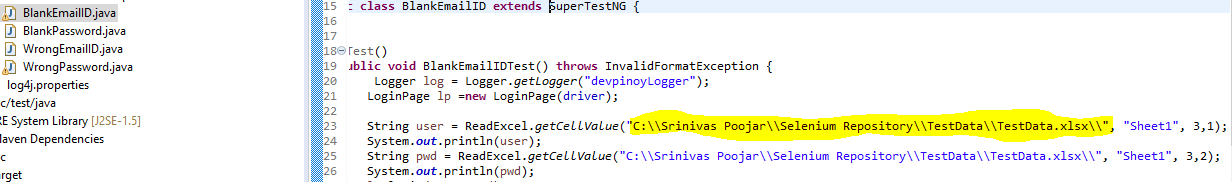
Change this path according to your system path.

1. Download the test data file to your local system:



1. Change the path of the test data file which is downloaded from above step under test cases.

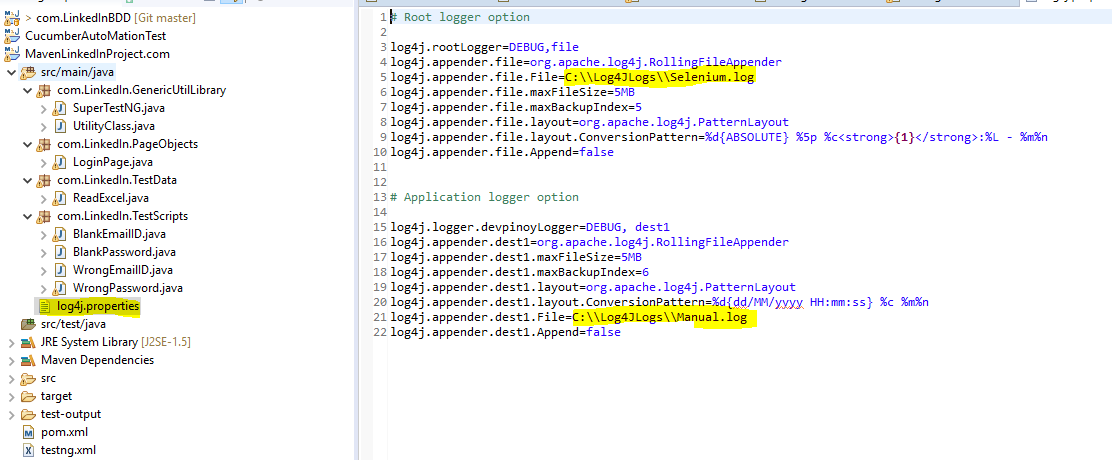
Say here, the BlankEmailID test case has the path of test data file as shown below from my local system:



This path needs to be changed according to the path where the test data sheet is present in your system.

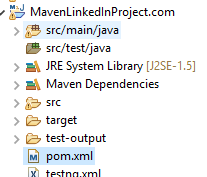
1. For logs to get created, please create an folders and file as mentioned below:

* Under C driver, first create a folder as Log4JLogs
* Under the same folder create two files as Selenium.log and Manual.log



1. Test running step:

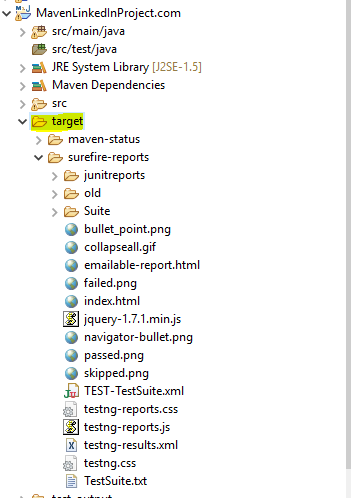
After importing and making the necessary configurations, go to POM.xml file under the project,



Right click and Run As -> Maven test

1. Test Result and Report:

The test result will be present under target folder of the project. Please refresh the project to get the latest test run report.



There will be an e-mailable-report-html file under same target folder. Once after clicking, the report will show as mentioned below:

