**Automation Framework:**

A test automation framework is an integrated system that sets the rules of automation of a specific product. This system integrates the function libraries, test data sources, object details and various reusable modules. These components act as small building blocks which need to be assembled to represent a business process. The framework provides the basis of test automation and simplifies the automation effort.

**Architecture:**



**Maven’s pom.xml** file contains all the dependencies needed for the project. i.e.,All the jars required for the project. It also points to the testng.xml file.

**TestNG’s testng.xml** file contains all test execution configurations along with classes to be executed.

**TestNG Suites** contains collection of classes containing test scripts to be executed.

**Page Objects and Methods** contains individual class files for each page in the application. Each class file contains all the objects present on the webpage and its functional methods/steps mentioned in the test cases. These classes make use of **Maven Dependencies** (jars) downloaded by Maven, retrieve Test Data from **testData.xlsx** Excel file through **Apache POI** tool.

**Utilitites** contains all the generic utility methods required for the entire project. Ex: excel utilities, TestNG Listeners, Reporting utilities, etc.

**TestNG Reports** are generated by TestNG once the test execution of the projetc is completed.

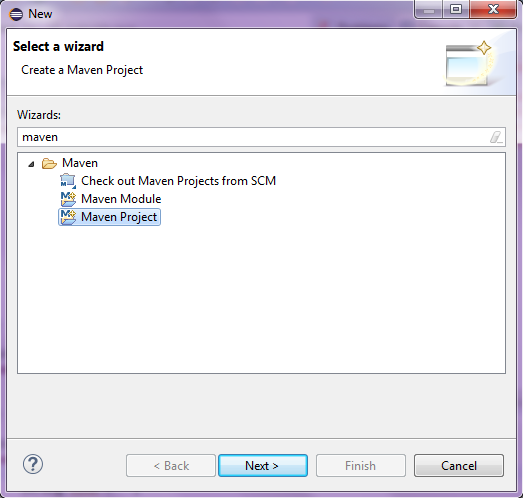
**Pre-requisites for framework creation:**

* Install JAVA and SET Environment variables
* Install an IDE (Eclipse)
* MAVEN PLUGIN from Eclipse Market Place

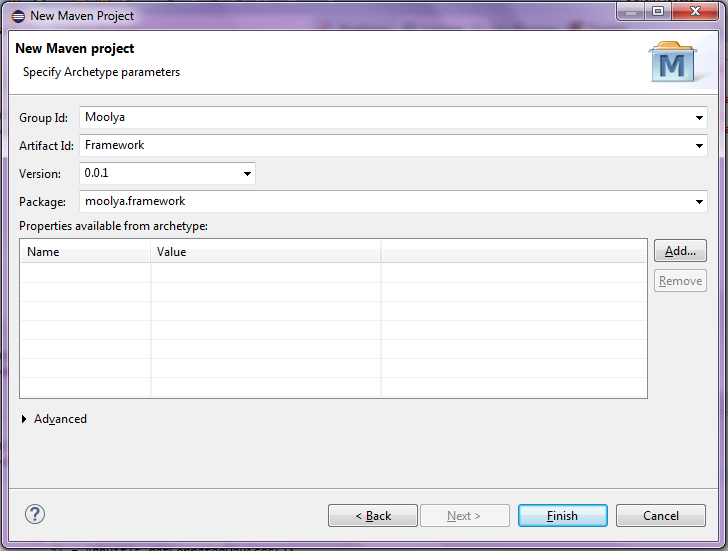
**Steps to Create Automation Framework:**

* Open Eclipse
* Create new Maven Project ----> Click Next -----> Select Workspace loation -----> Click next

------> Select maven-archetype-quickstart Archetype ------> Click Next



* Enter Group-Id, Artifact-Id and Version. Change package name to lowercase. Click Finish



* After creating, project will appear like this. Double Click pom.xml file and select pom.xml view in editor. Erase the below highlighted lines and add these lines.

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.19</version>

<configuration>

<suiteXmlFiles>

<suiteXmlFile>testng.xml</suiteXmlFile>

</suiteXmlFiles>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-server</artifactId>

<version>2.43.0</version>

</dependency>

<dependency>

<groupId>io.appium</groupId>

<artifactId>java-client</artifactId>

<version>2.1.0</version>

</dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.8.8</version>

</dependency>

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.2.4</version>

</dependency>

<dependency>

<groupId>org.json</groupId>

<artifactId>json</artifactId>

<version>20141113</version>

</dependency>

<dependency>

<groupId>org.apache.httpcomponents</groupId>

<artifactId>httpclient</artifactId>

<version>4.3.4</version>

</dependency>

<dependency>

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

<artifactId>poi</artifactId>

<version>3.7</version>

</dependency>

<dependency>

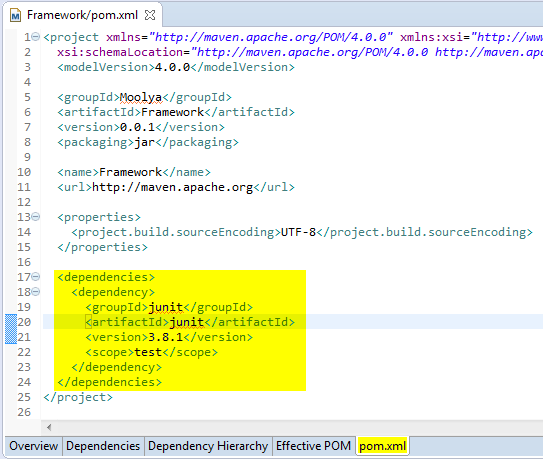
<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

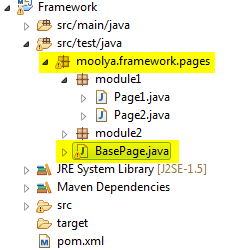
<version>3.14</version>

</dependency>

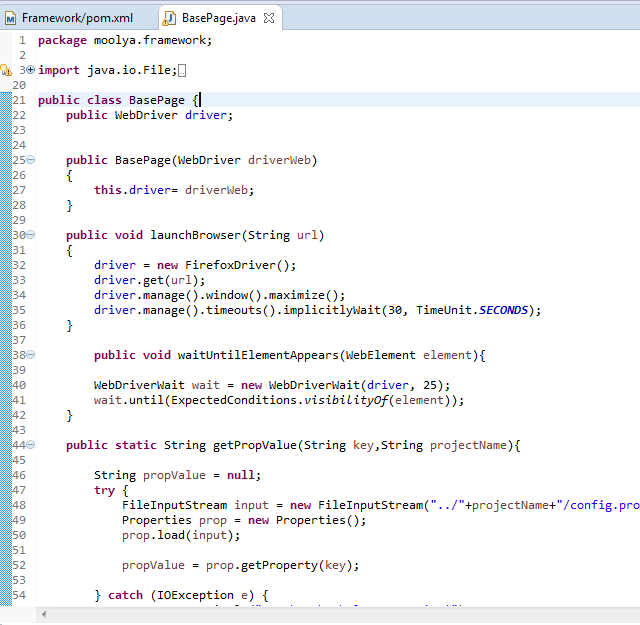
</dependencies>



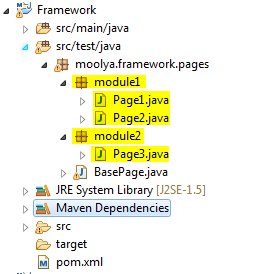
* Save the project and wait until all jars are downloaded from maven online reoository and building workspace is done.
* Whole project scripting will be done under src/test/java package. Create packages pages under that package and BasePage.java class under pages package.



* Create WebDriver variable and constructor in Basepage.java. Add common methods of webdriver like launch(), wait(), selectDropdown(), etc methods in BasePage.java.



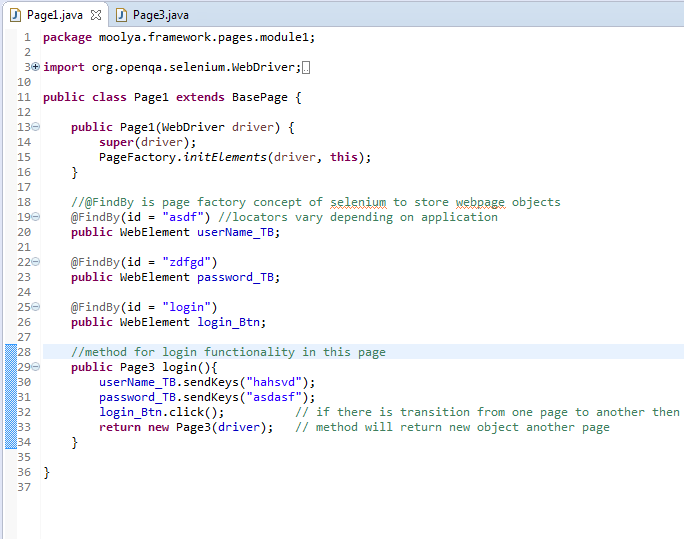
* Create packages for modules under src/test/java and classes for its respective pages under it as below.

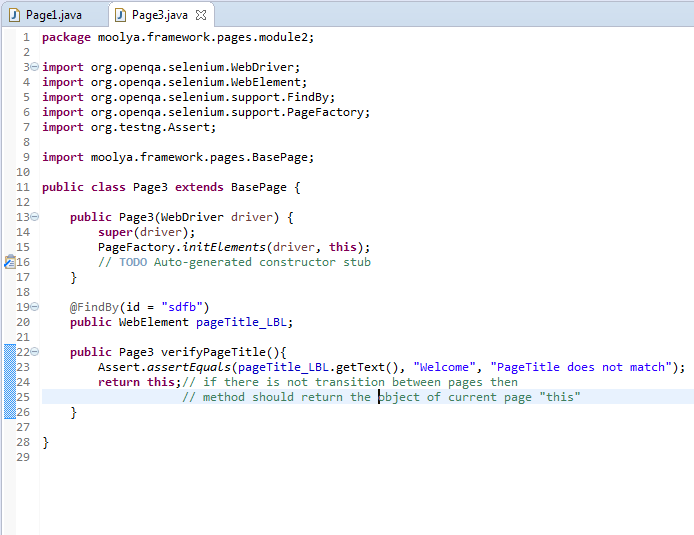


* Each class created for individual pages must extend BasePage
* Create a constructor which calls the BasePage constructor and add this line

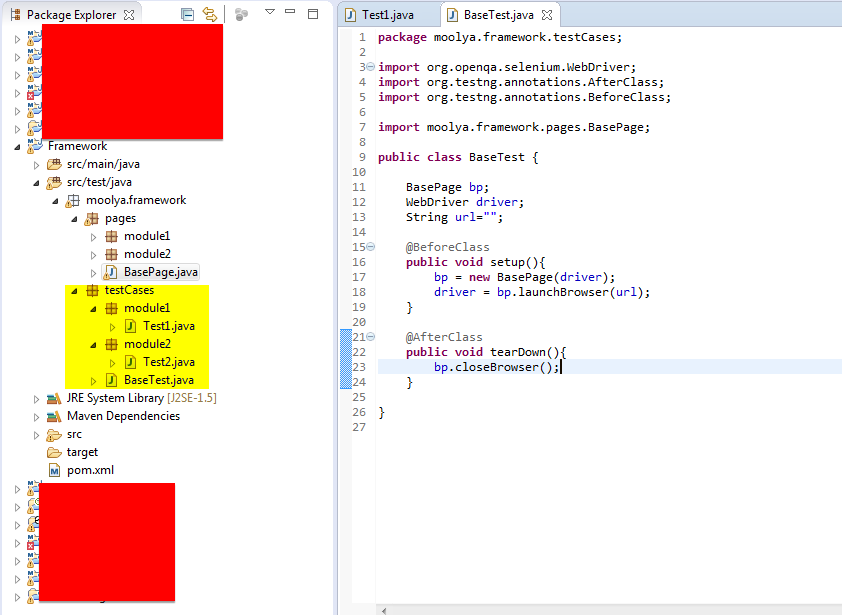
**PageFactory.initElements(driver, this);**

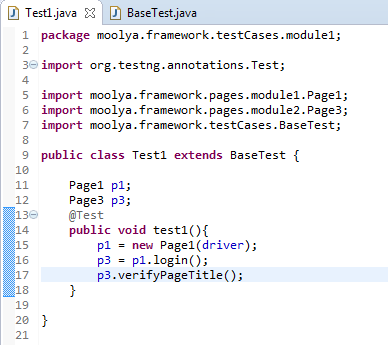
* Objects related to that particular page must be declared
* Methods related to the page’s functionality must also be defined



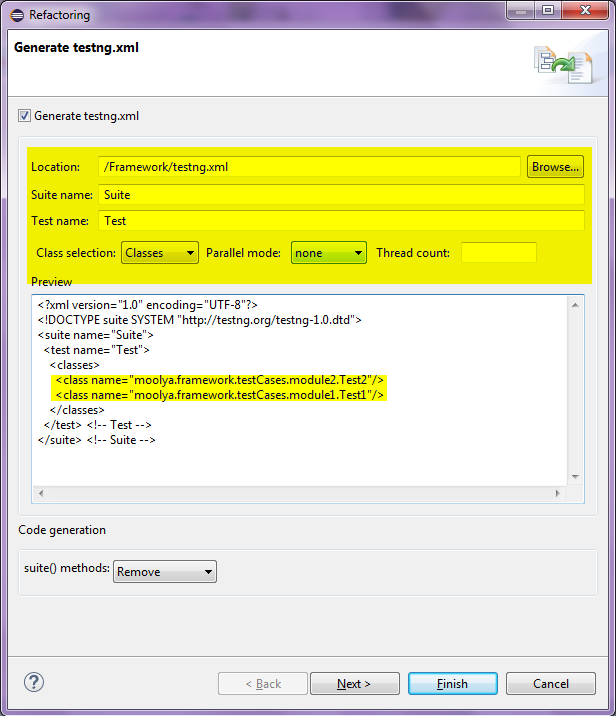


* Create package testCase under src/test/java and create a class BaseTest.java
* Add all the pre-condition and post-condition methods which are common to all the testCases using testNG annotions (BeforeClass and AfterClass)
* Create packages for modules and classes for testcases related to those modules
* Each testCase class must extends BaseTest class and test case are written in a methods using testNG’s @Test annotation.
* Object of particular page class must be created and method of that page class must be called inside the test method according to the steps written in test cases.
* If there is any transition from one page to another then calling of one page’s methods must be assigned to another page variable since the method returns new object of another page.

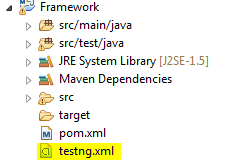


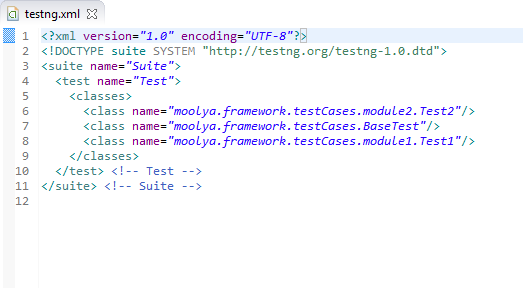


* Right Click on project -----> TestNG -----> Click Convert to TestNG
* Configure highlighted areas
* Make sure all the test classes are included inside class tag
* Click finish

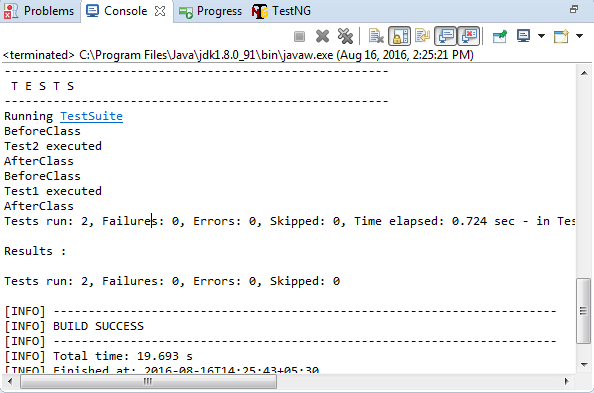


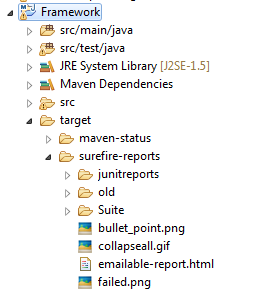
* After creating you can see testng.xml file and its contents as below

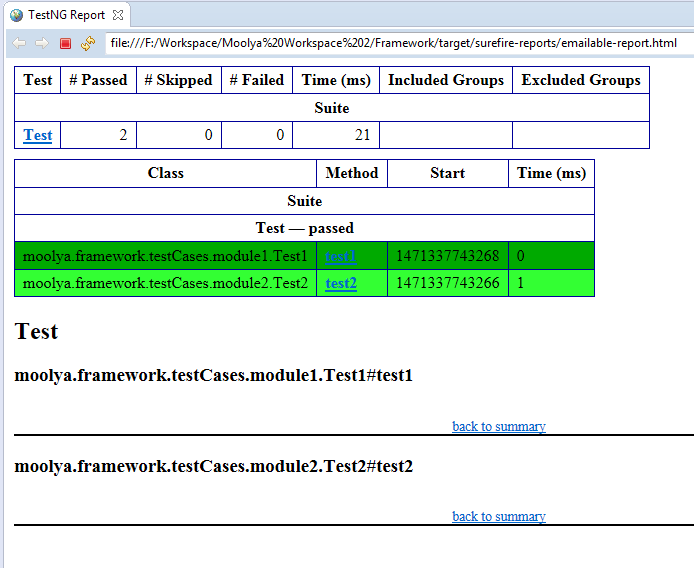




* If there is any addition or deletion of test classes then update the class tag in testng.xml
* Finally, Right click pom.xml -----> Run as ------> Maven test. Wait for execution to complete.
* After execution is complete, you can see three testNG report views: console output, testNG heirarchical report and testNG html reports in surefire-reports directory under project folder/target folder
* It generates emailable-report.html which can be attached to email.







**Note:**

* Various Utilities can be used depending on requirements of project
* For ex, if you want to read test data from excel then you need to write custom method to for that using Apache POI tool
* If you want to genrate custome reports then you need to use reporter utilities. etc