# Selenium Framework Set up Guideline-:

# System Pre-requisite-:

- Java 8(JDK\_1.8) or Upper version (Note-: Java 8 will be most stable for the framework to work seamlessly)

Browser-:

-Firefox – Recommended V76.0.1(most stable) or Latest V 77 .0.0 will also work

-gecko Driver – v 26.00

-Chrome – Recommended v83

-ChromeDriver-v83

-IDE - Eclipse -Neon , Oxygen ,Java Developer etc.

- Selenium Java version -3.141.59 (need not add jar , already handled through POM.xml)

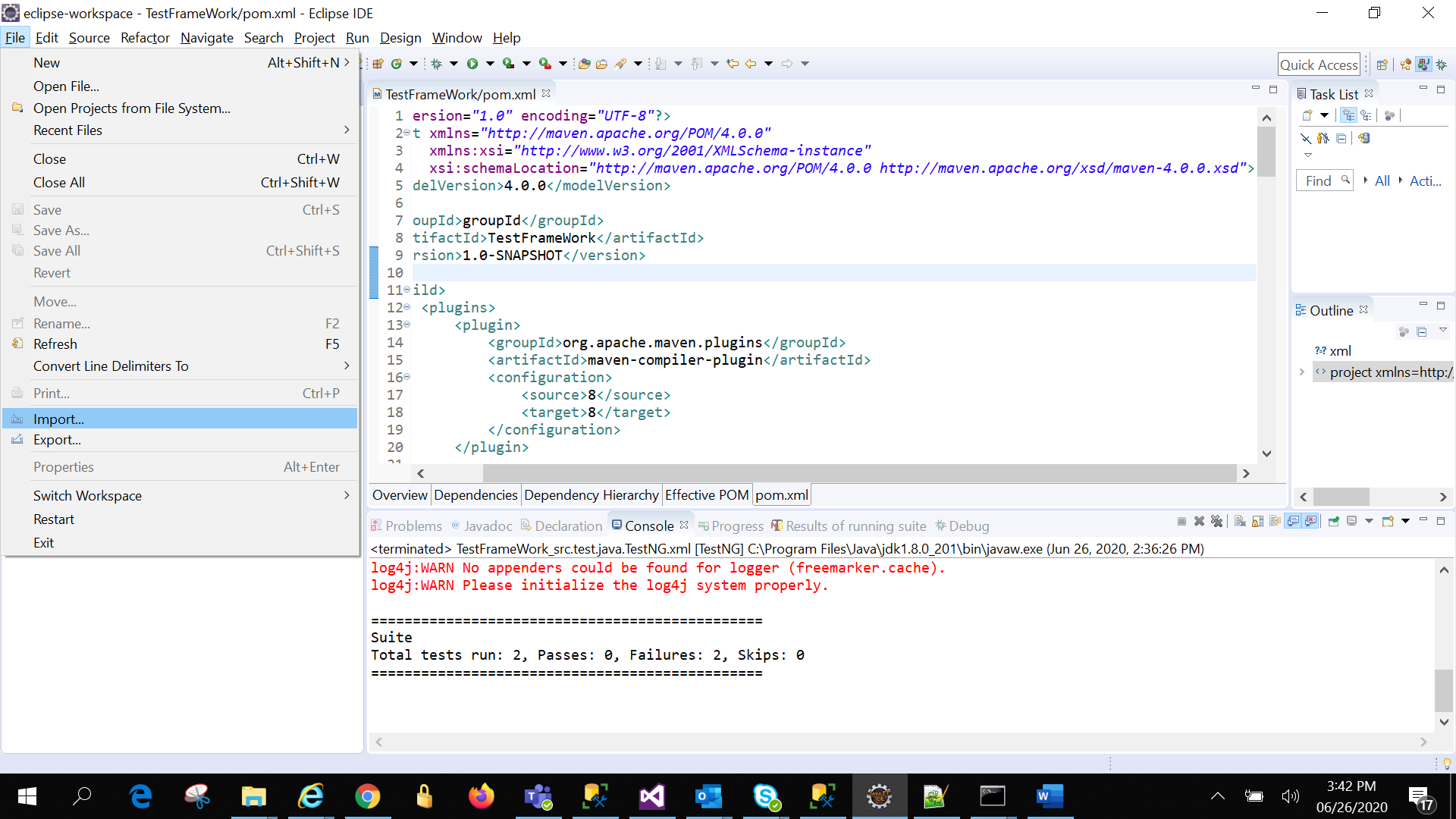
-Selenium Server verion-3.141.59 -(Download to project directory)/.m2 by POM.XML

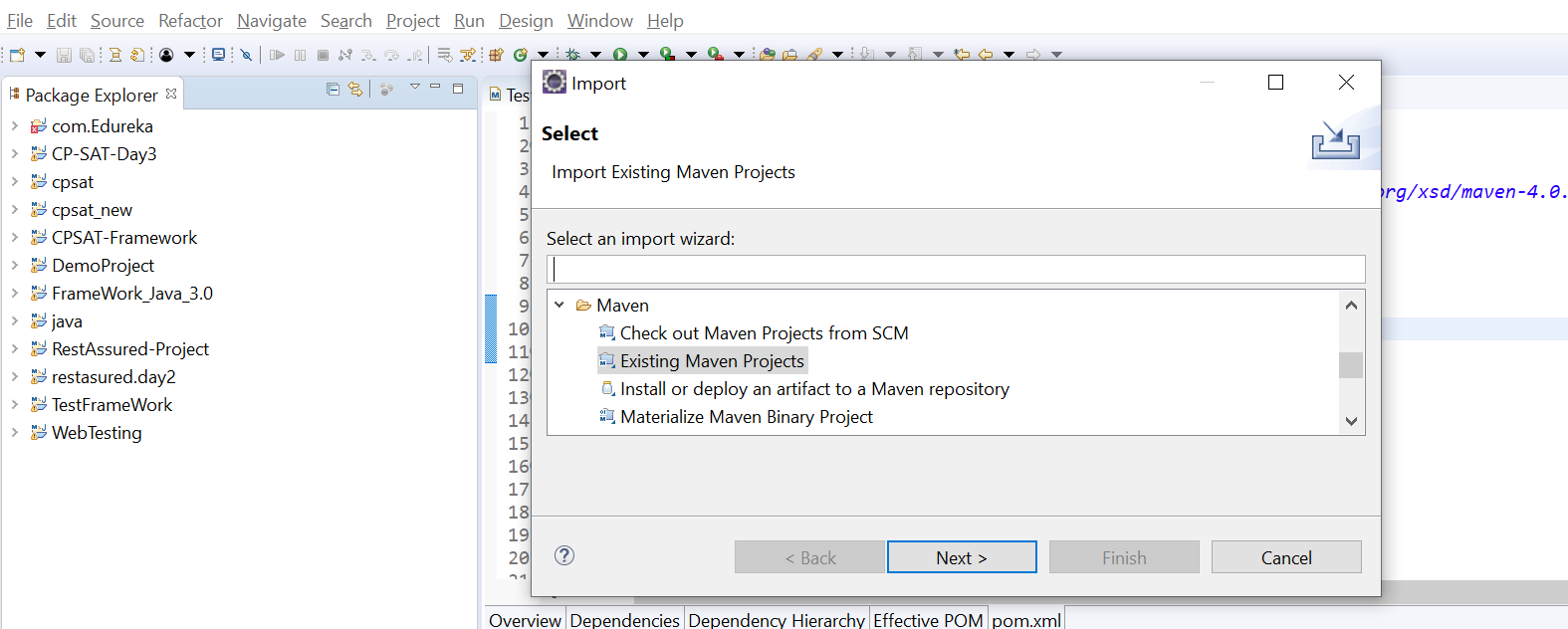
-TestNG V.7.1(added as dependency by POM.xml)

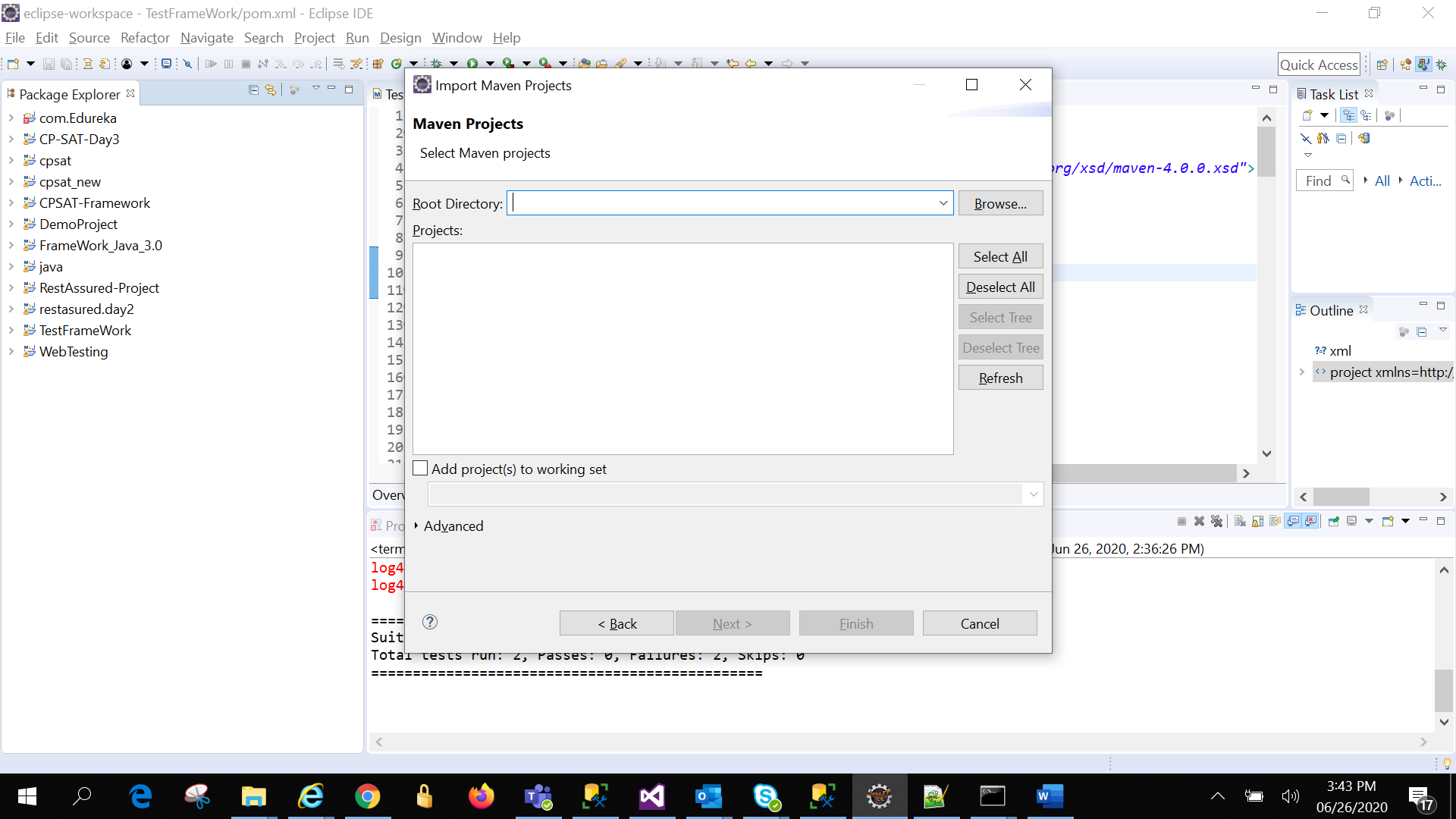
-Maven to be installed into eclipse

# Import below project in eclipse as below-:

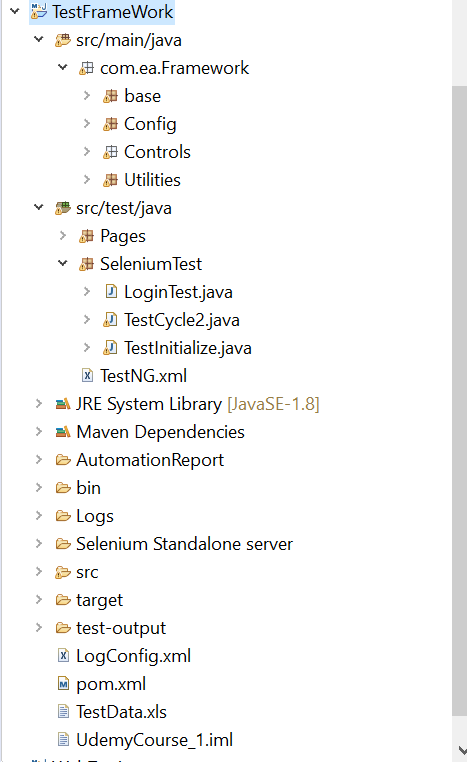
File->Import->Select folder (Maven)->Existing Maven Projects->browse (give the project location to be imported)->Click Next->click Finish







Project will successfully get imported and will look like below in eclipse.

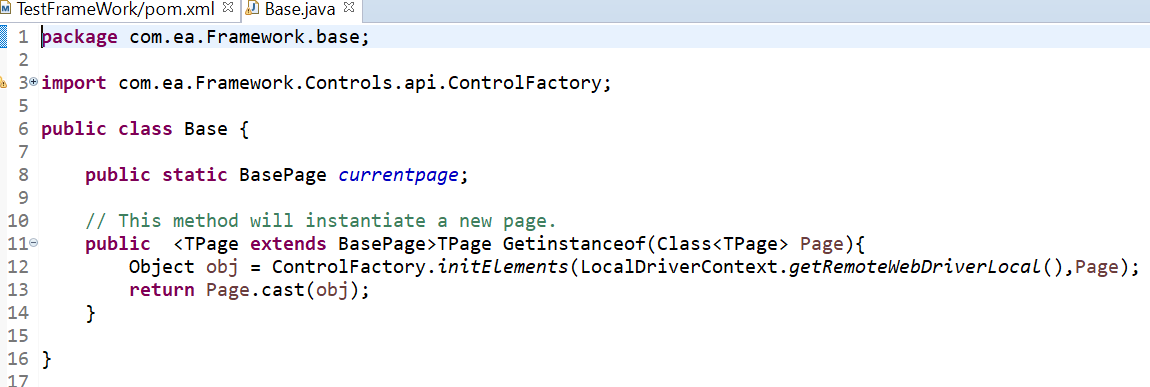


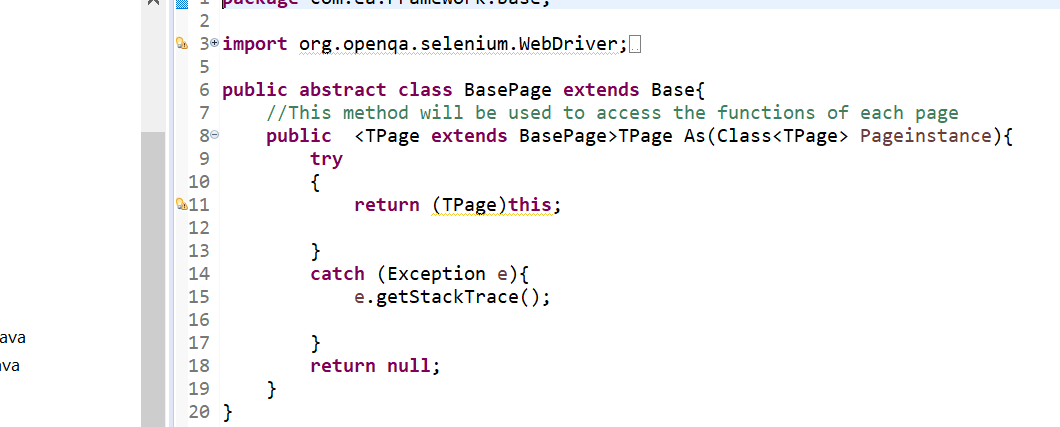
# Frame-Work Internal-:

## Package Base

1. This has Driver instance, Page Factory call, Browser Type (Enuma), Frame-Work Initialization
2. Base and Base Page

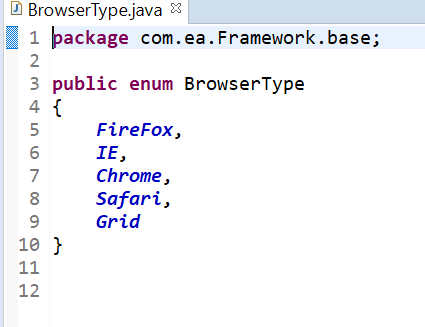
-It has Page factory and Page factory initialization



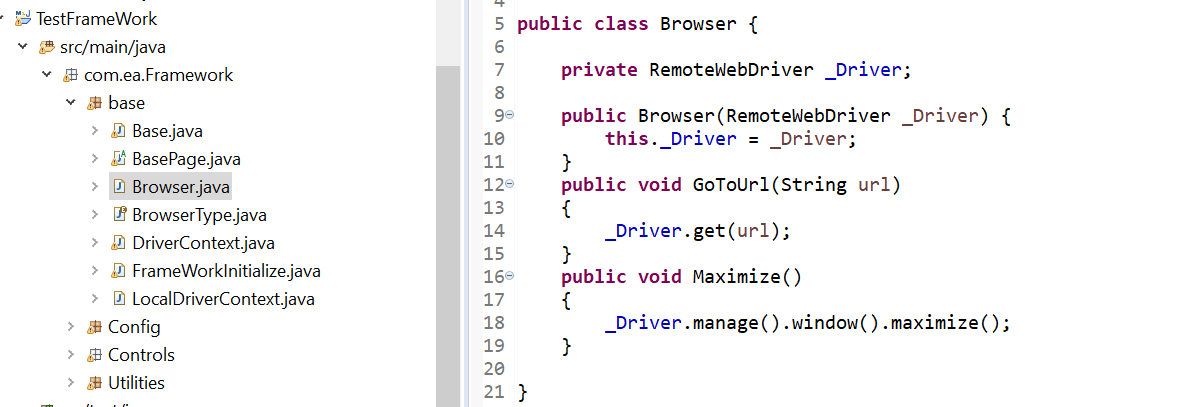


b)Browser-Type and Broswer.java

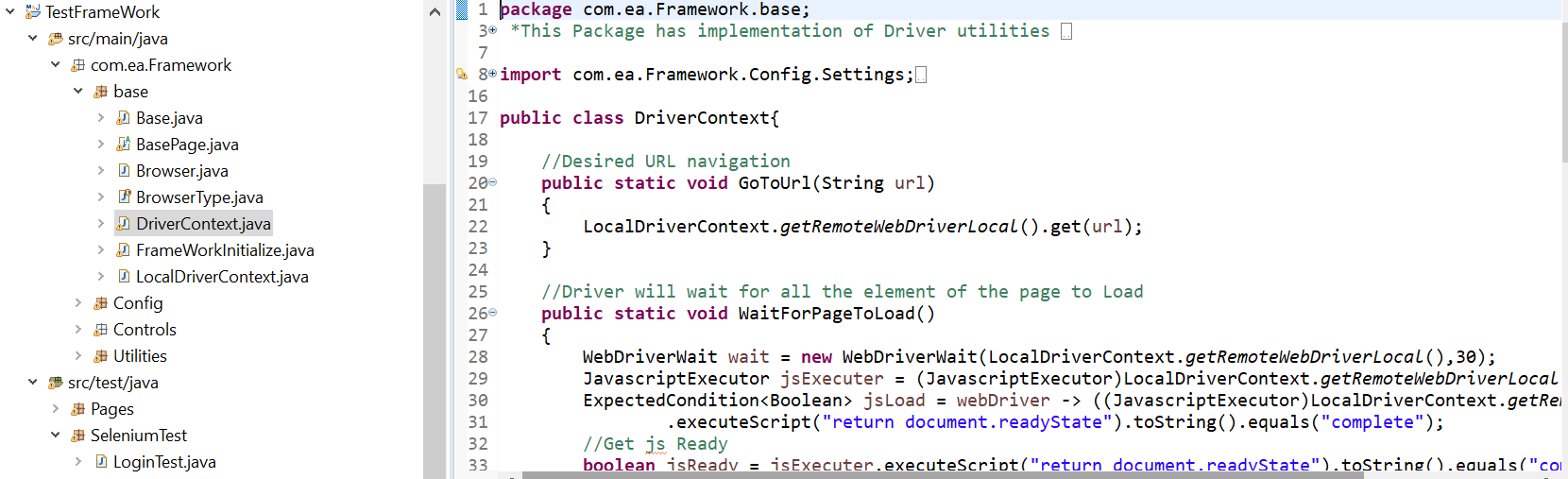
-It has different Browser name and Type is enemy type



Browser.java-It has URL invoke and maximize method



c)DriverContext.java-It has driver methods like Wait,WaitForPageLoad



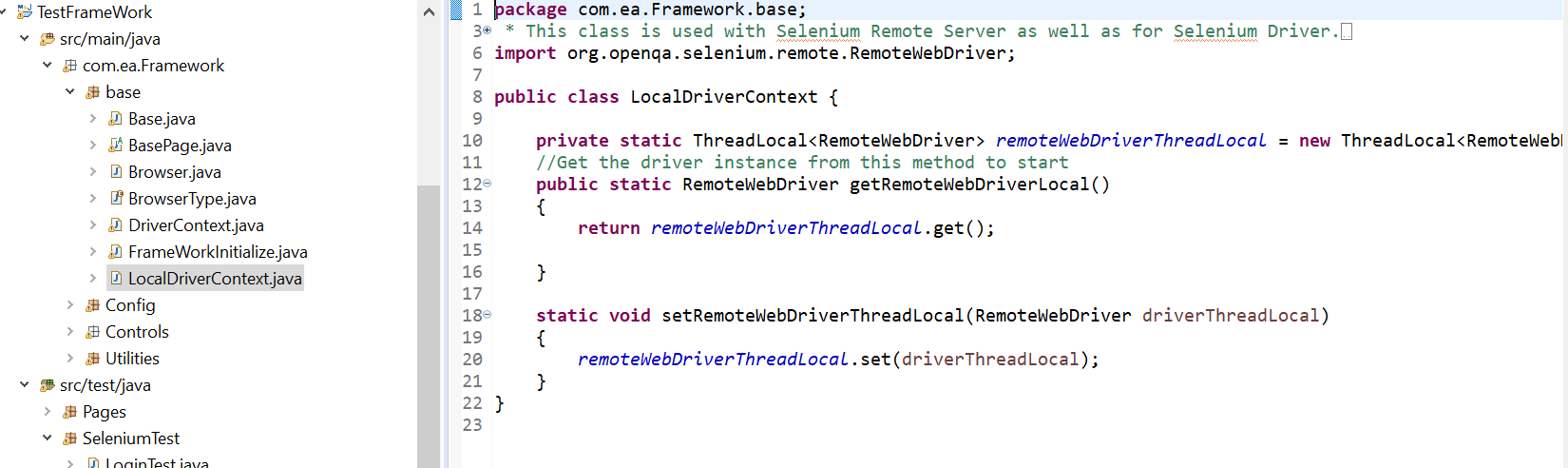
d)FrameWorkInitialize.java-This has Driver initialization for different browser as well as Grid method-:



e)LocalDriverContext.Java-This has methods in getting the driver object which is implemented in above script.

-This will make the driver object to thread safe so that multiple browser also can access the same instance.

-Method will be used both for Selenium Grid and Selenium browser testing

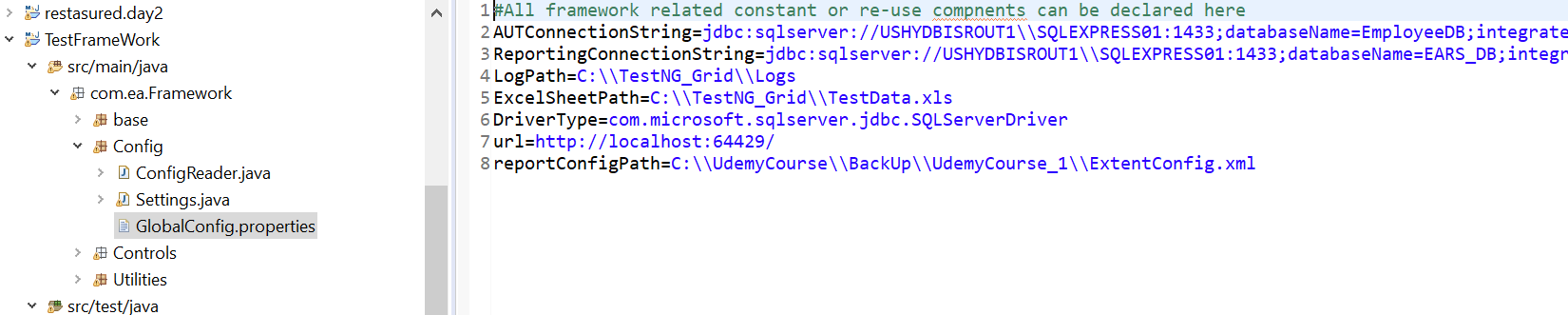


## Package Config-:

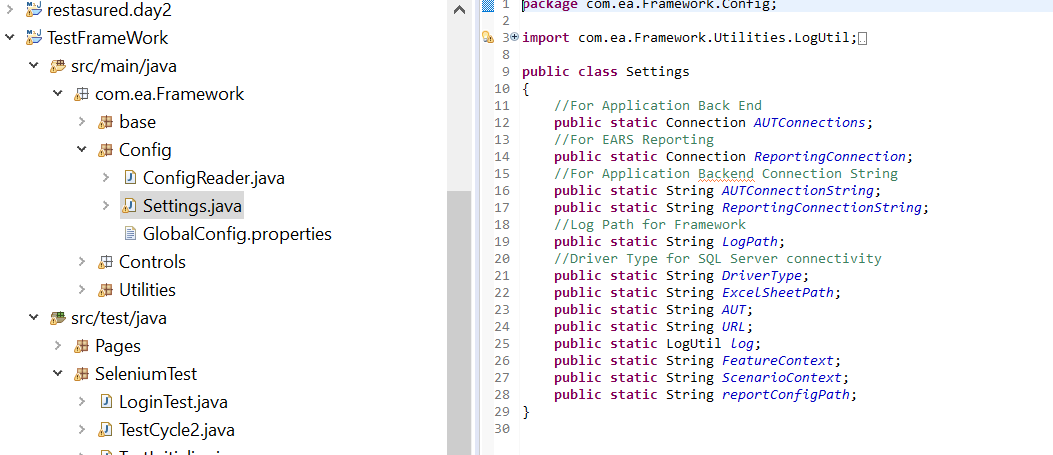
1.This has framework level constant details like Driverpath,broeser name,Log path,Database connection string, excel sheet path, Driver path etc.

2.User can mention as much as Constant with the name and reuse them across the application

a)GlobalConfiguration.properties

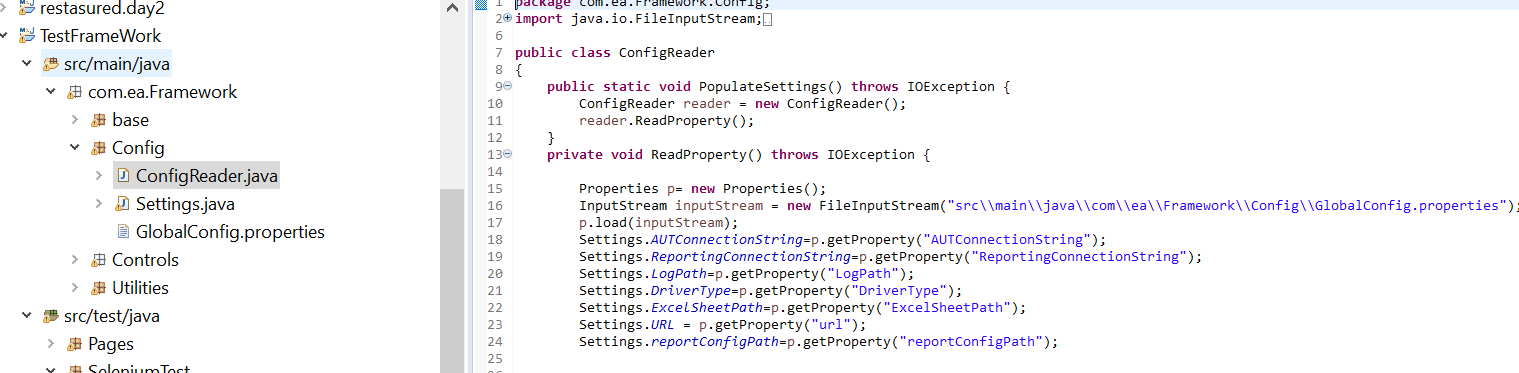


b)Settings. Java-For each of the above constant in this method a variable has been called out



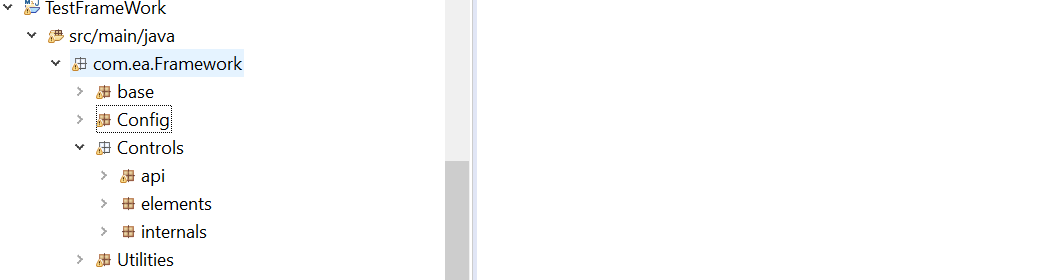
c)ConfigReader.java-Methods to read all the variable where the CONSTANT values are stored.

User has to call each constant by this syntax “Setting.(Constant name) in all the test method



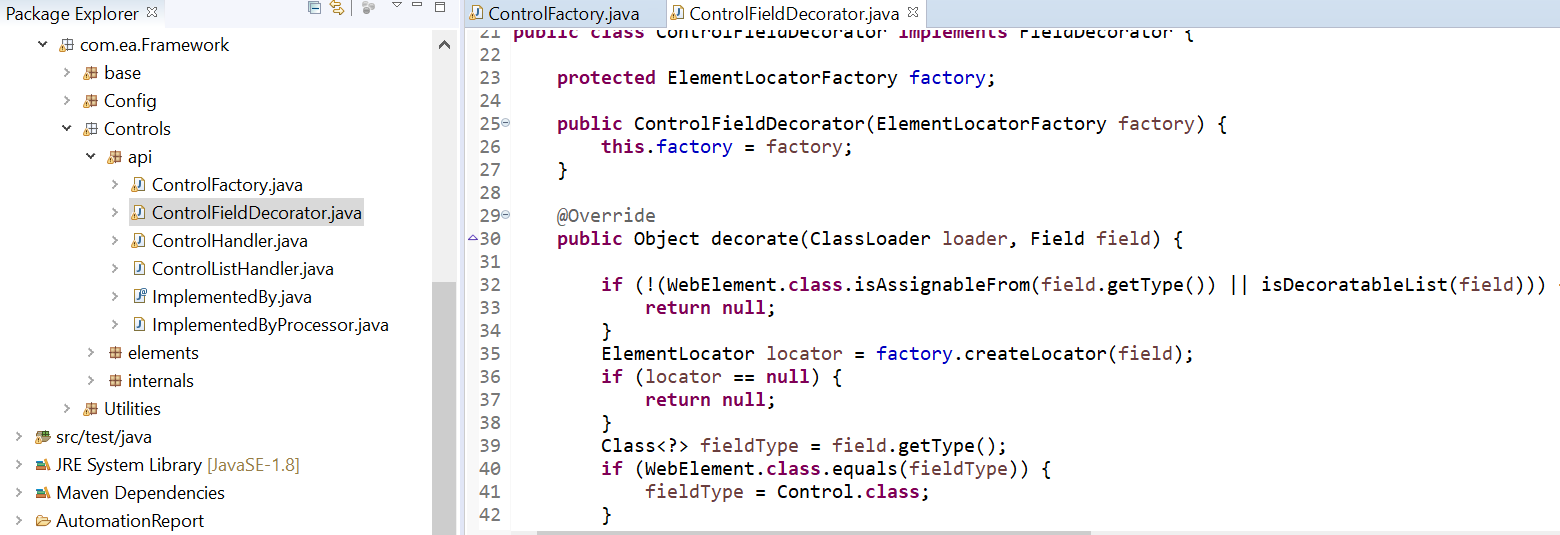
## Package Controls-:

We are trying to achieve different Web-element method for Button , Hyperlink, Textbox etc. so that user can call the method directly from the element (i.e. user doesn’t have to explicitly call out the webelement.Click()



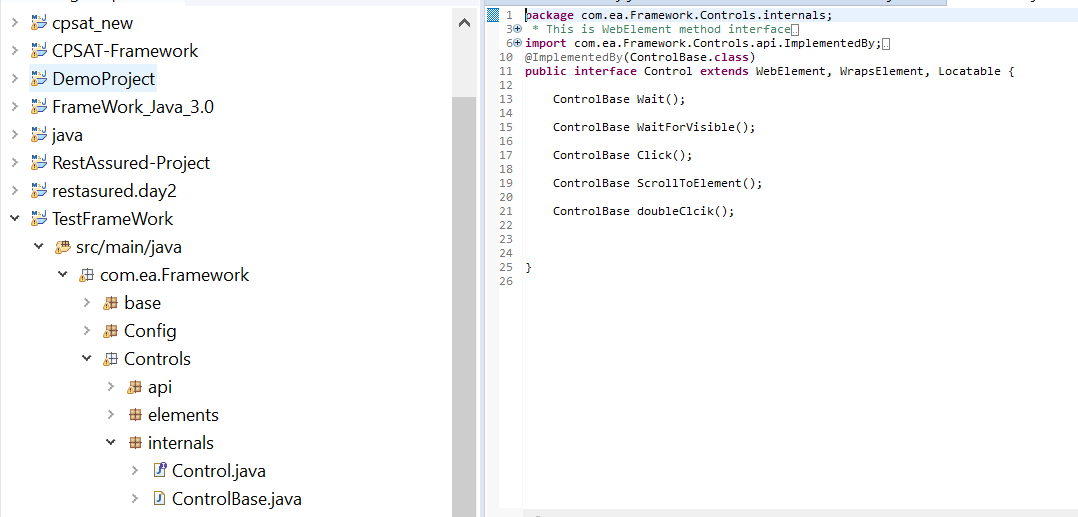
a)-Api-Package has framework internal modification for Page object model so that user an easily perform the weblement action.

Note-:It is strictly recommended not to modify any of this implementation .It may cause framework error



b)Package-Internals

-Control Interface- This interface has designed to be used as a bas for all the type of Web element to access the method in it which is again implemented in ControlBase.java class



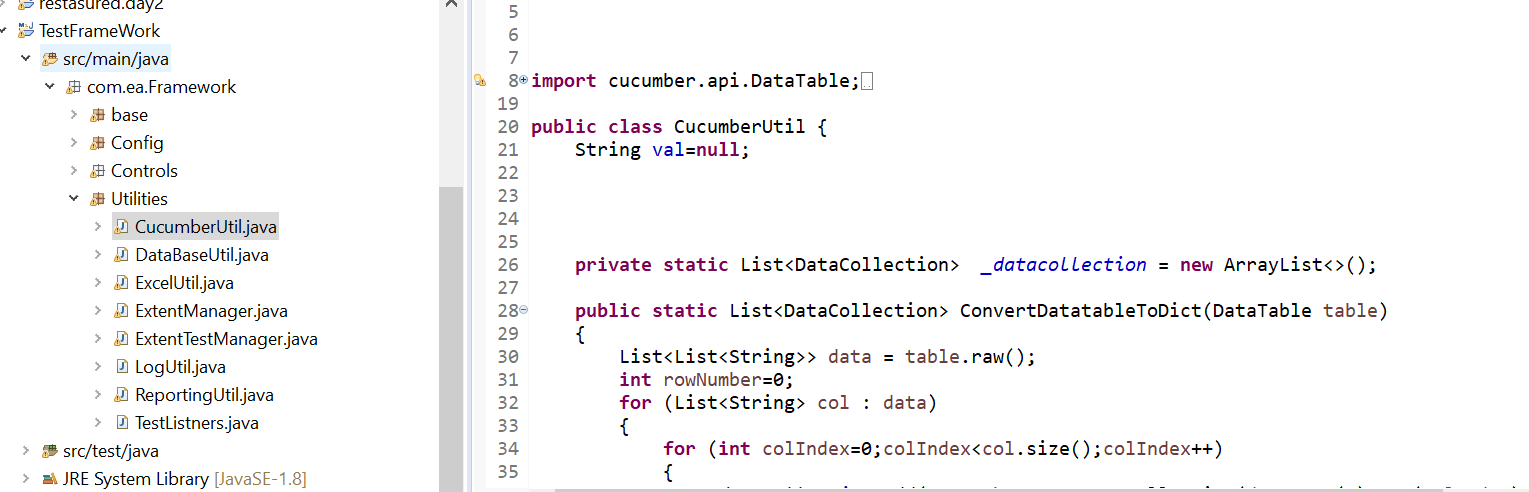
Control is implemented by control base here-: This has all the web-element methods to be used by ‘Button’, Textbox, Hyperlink etc.



## Package-Utilities

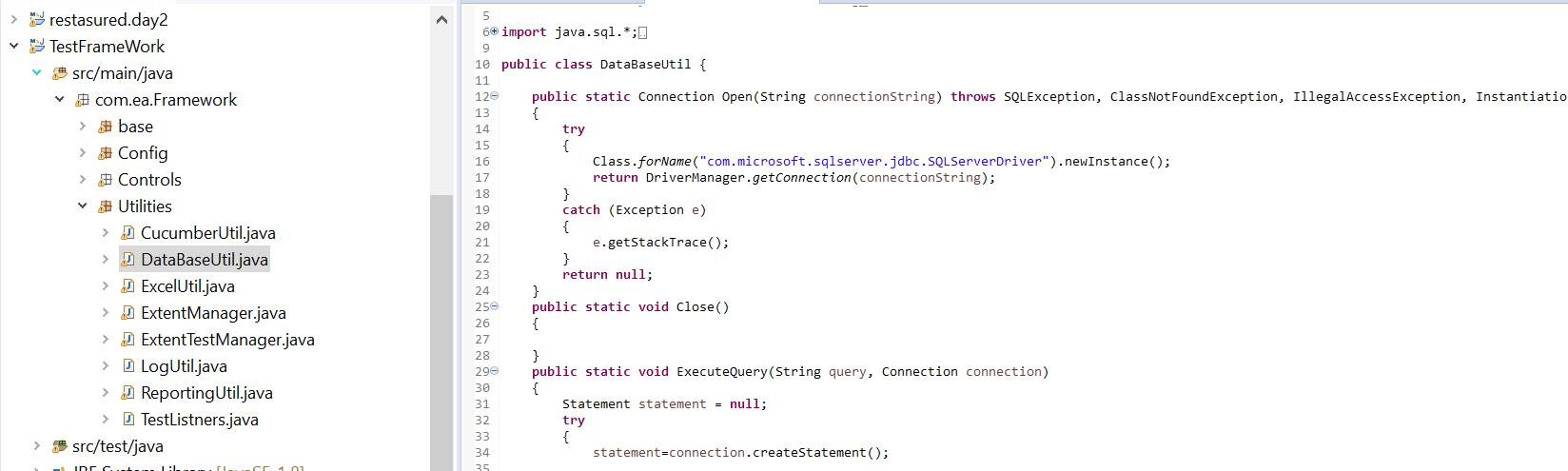
1.Java files for supporting utilities in making the framework more testing friendly.

a)CucumberUtil.Java -It has methods to access feature file and scenario table. This is optional and only used when tester implements Cucumber BDD .

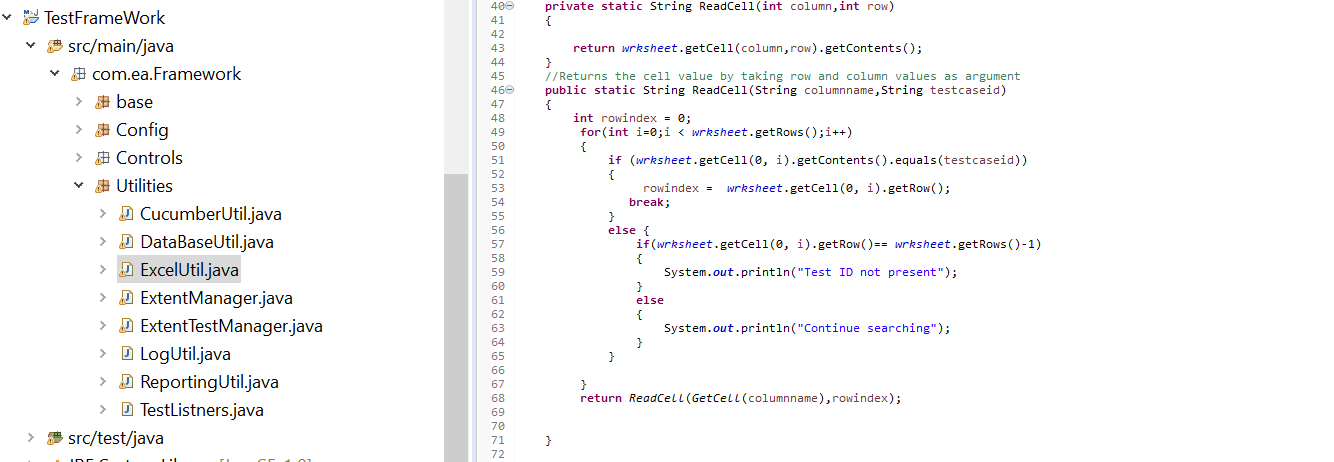


b)DataBaseUtil.Java-It has method for creating a Database connection from connection string as well as inserting and querying from the respective data table.

-Only used for selenium Data Base testing



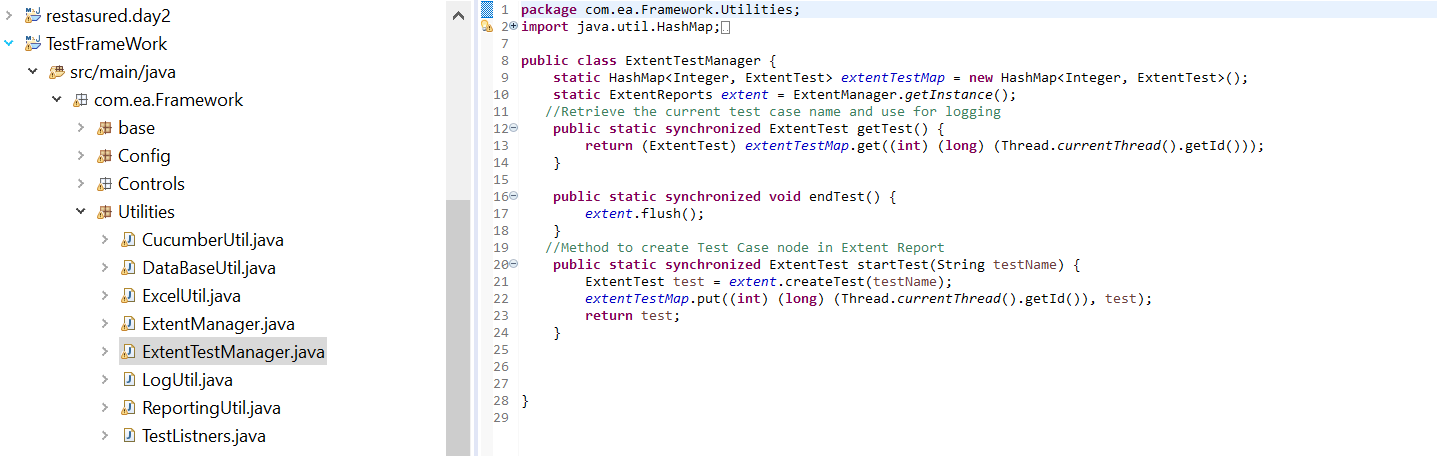
c)ExcelUtil.Java-It has method to fetch data from Test Data sheet by @Param ‘CoulumnName’ and ‘Testcase Name’



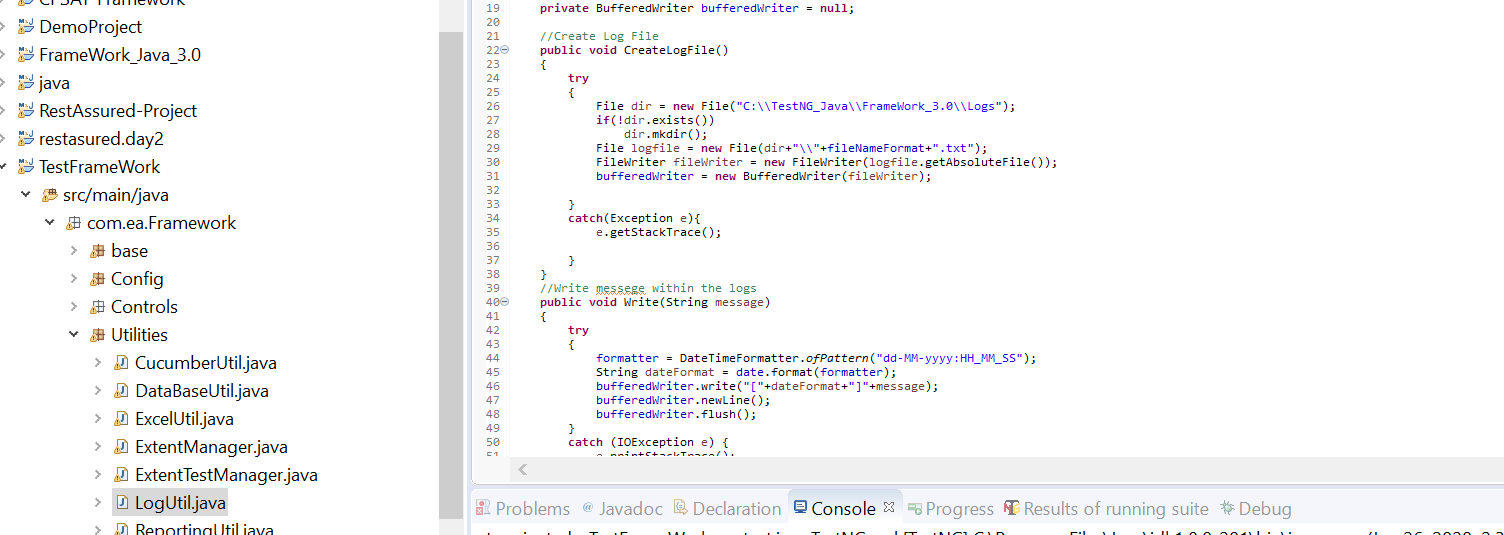
d)ExtentManager.Java-This has method to create extent html file and base for logging the report



e)ExtentTestManager.Java-It has method of getting the testcase name, initiating when test execution starts and test case ends. Basically, it has method which is used in logging in to html report when a test execution starts.

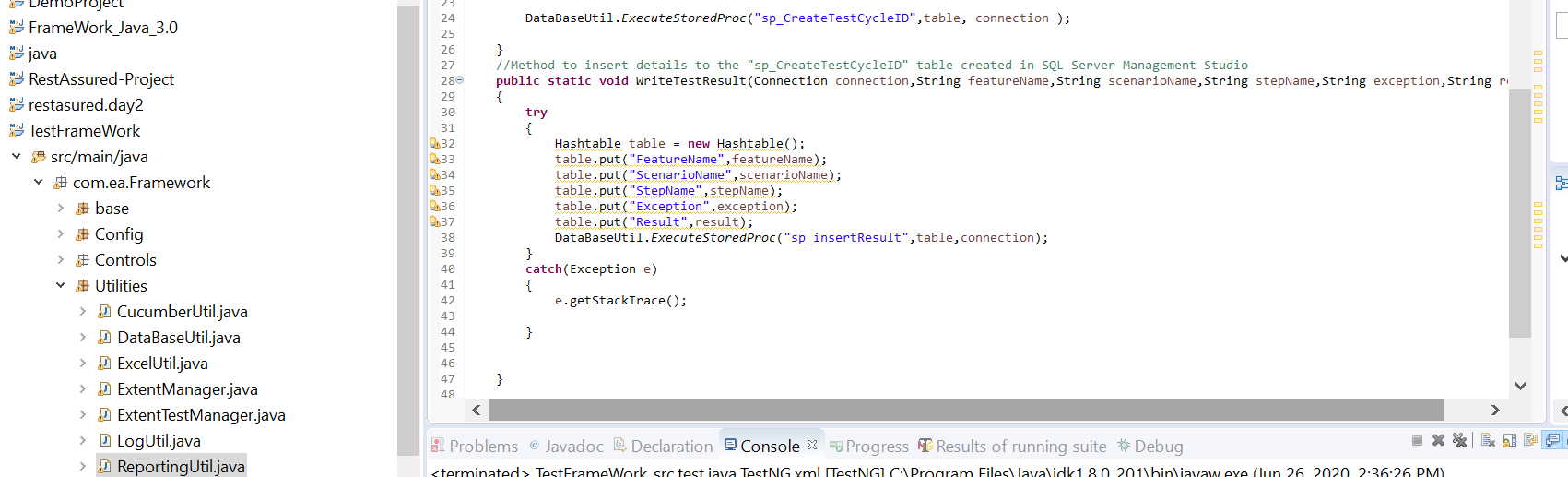


f)LogUtil.java-Helps in logging mechanism to txt file while execution progress .User can implement this by calling **Setting.log.write(String value**); .User can pass the execution details in to this method to save the test log in to a txt file which can be referred later.



g)ReportingUti.java-Helps in creating Test HTML report which runs with the help of internal Data Base. If there is any reporting system which comes with the data base structure in its back end where all the HTML content including the test screen shot saved ,then in those cases we can use the below process.

Note-This implementation is optional to this framework.



h)TestListners.java-This is listener class which work along side with TestNG test script.



# TestNG Scripting and Execution Model-

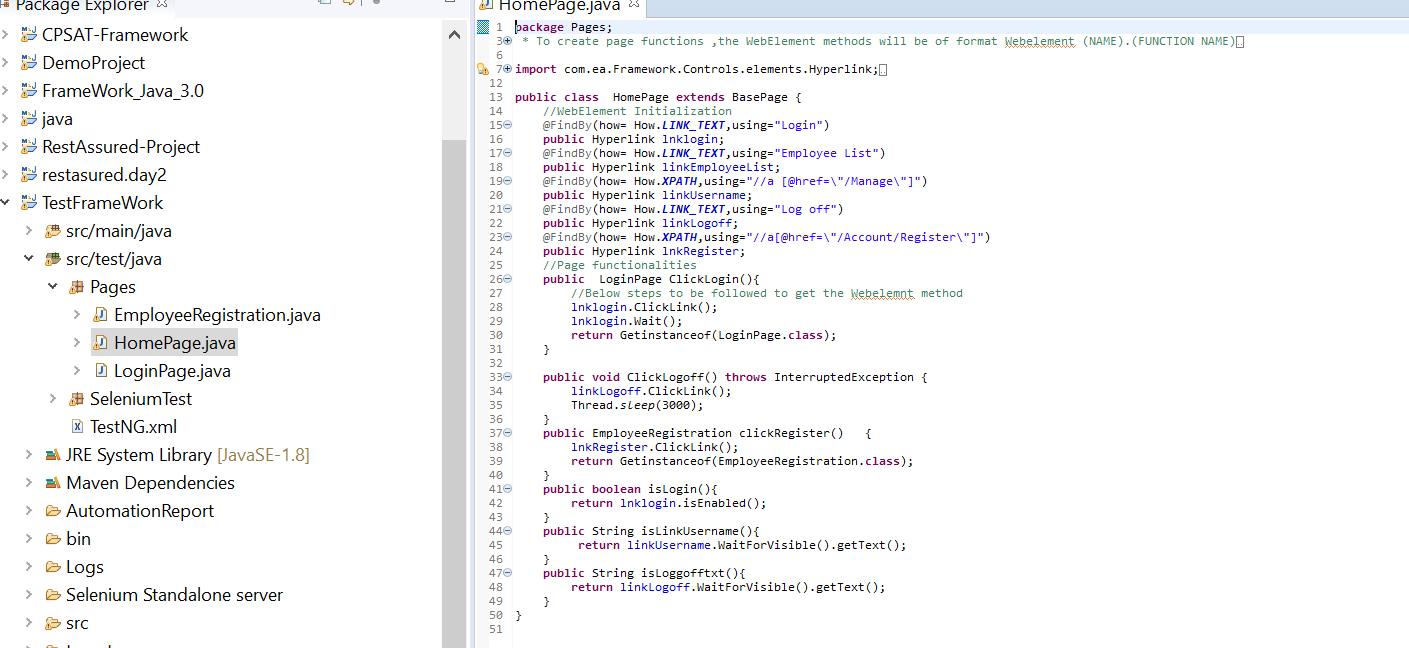
## Page Object-:

1-Create Pages where all the Element identifiers and method exists under

Src/test/java ->Pages->Create {page}.java file and start scripting as below

Mention Textbox if element is ‘Textbox’ ,Button if element is Button, Hyperlink if the element is with anchor tag from the DOM

Basically – all the Element type extends same class i.e. ControlBase.java which has all the element methods are defined.



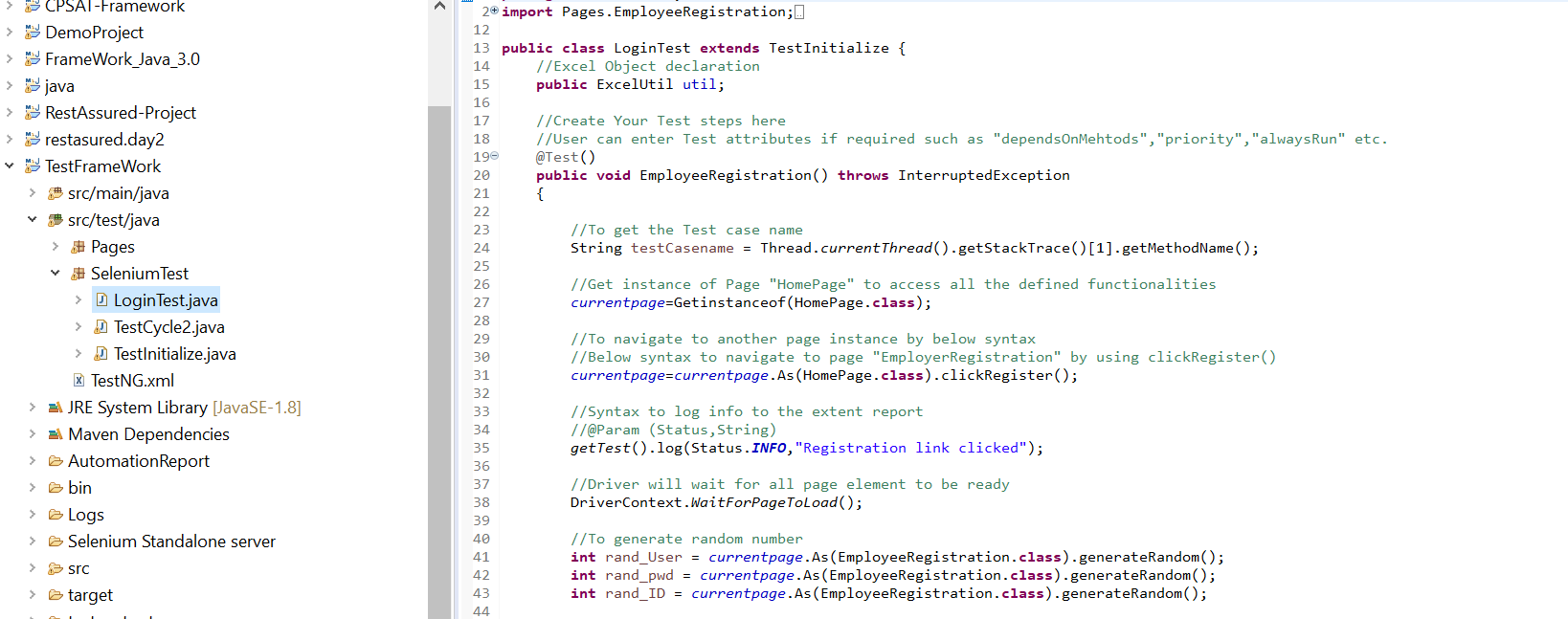
Create as many as pages available to the AUT as Java files and steps as displayed above.

E.g. implemented 3 pages above. ‘LoginPage.Java’ , ‘HomePage.Java’,’EmlyeeRegistration.java’

## Selenium Test-:

1.Test Java file with @Test are mentioned under ‘Selenium Test ‘Package

User can name the file based on there requirement and put the @Test specific to the application



Note-:

‘Current Page- Resistance(Homepage. Class) is optional. This has implemented not to create page instance every time.

User can create page instance object and call out all the method in there.

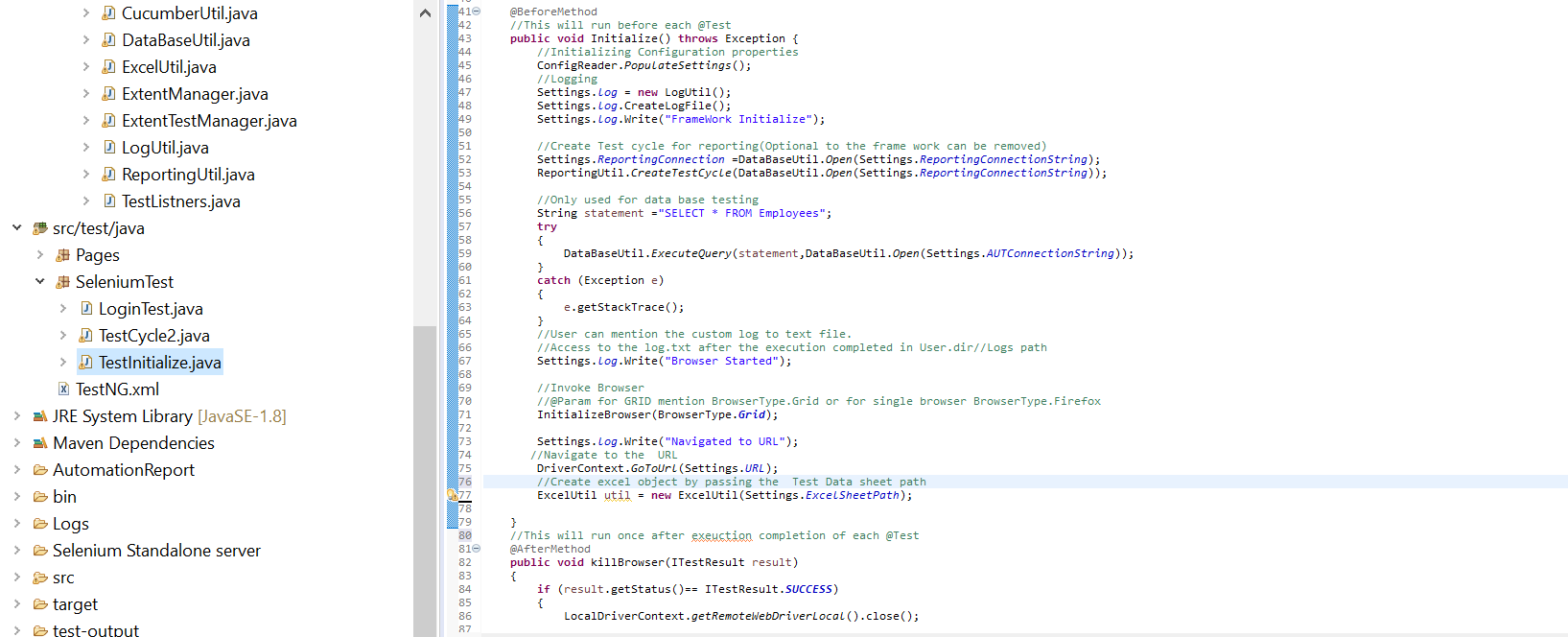
Like - LoginTest ln = new LoginTest()

Ln.clickLogin();

After creating the @Test.Next step is to initiate the selenium web driver and closing the driver to complete the test execution-

For this user can mention on Initialize java file like below.

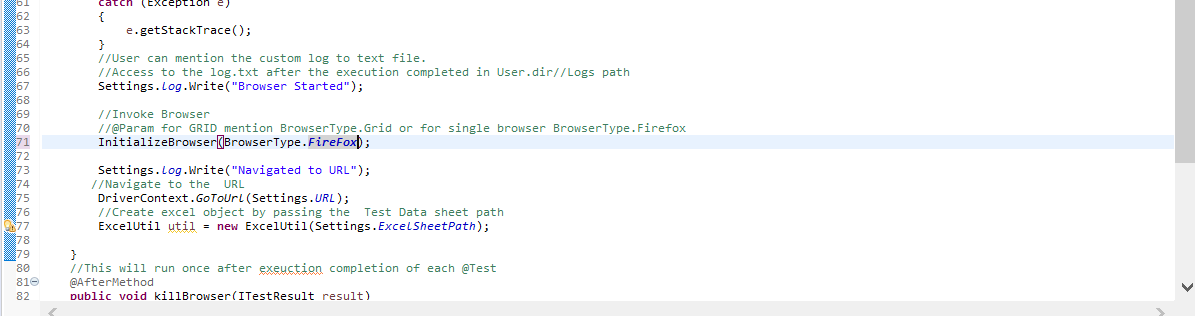
‘TestInitialize.Java’-You can ignore the database util methods written in this .This is optional



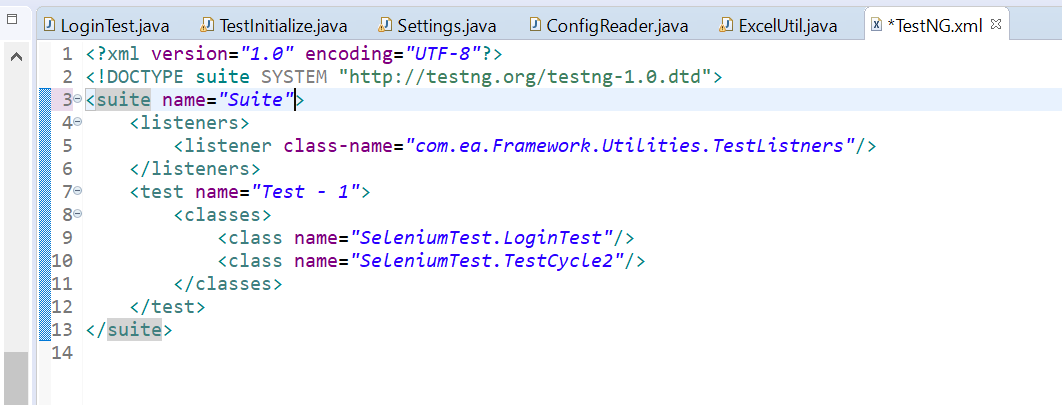
## TestNG.XML

Once the @Test design has been completed. User can follow the below xml and enter the class names where the @Tests are present .

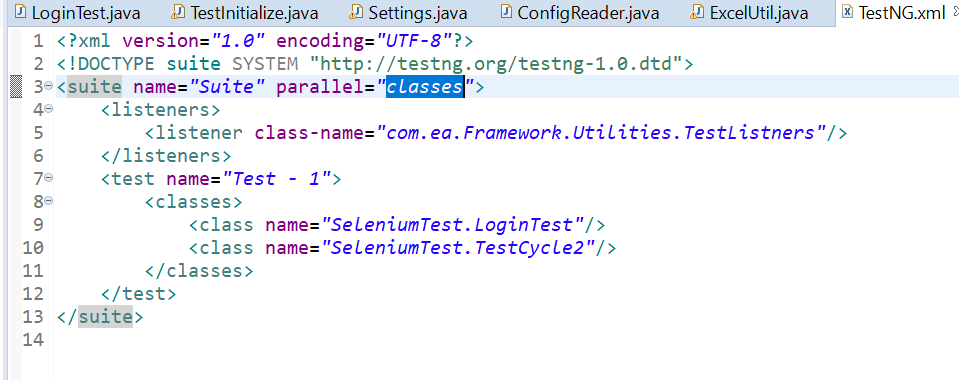
For Single browser before initiating the Execution update the Browser.FirFox in below method which is present in ‘Test Initialize’ java



Then remove the Parallel from the TestNG.xml



For Parallel execution mention attribute name Parallel =’Class’/’Method’

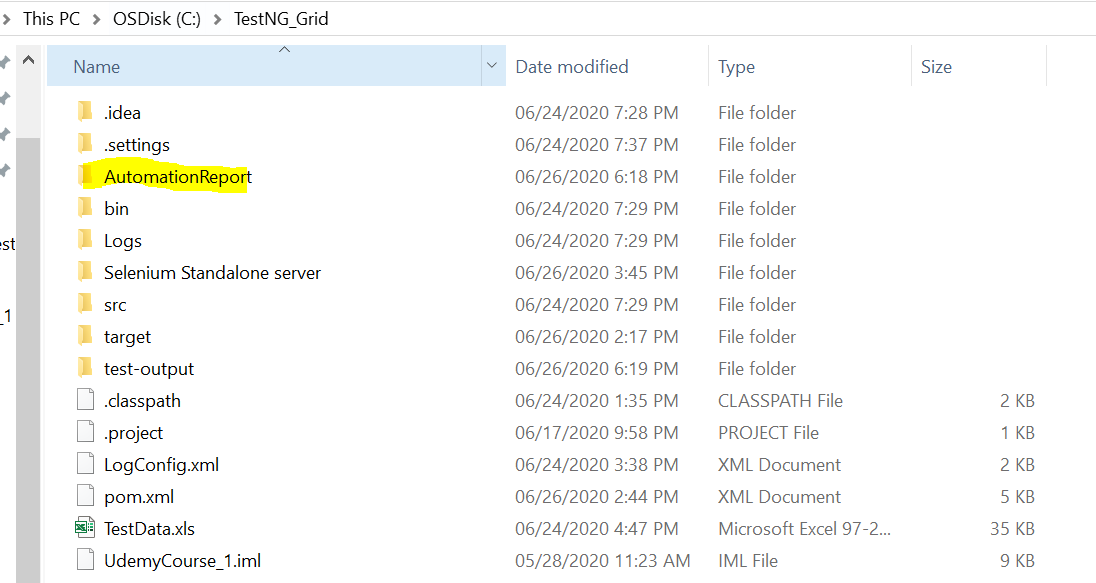


## Reporting

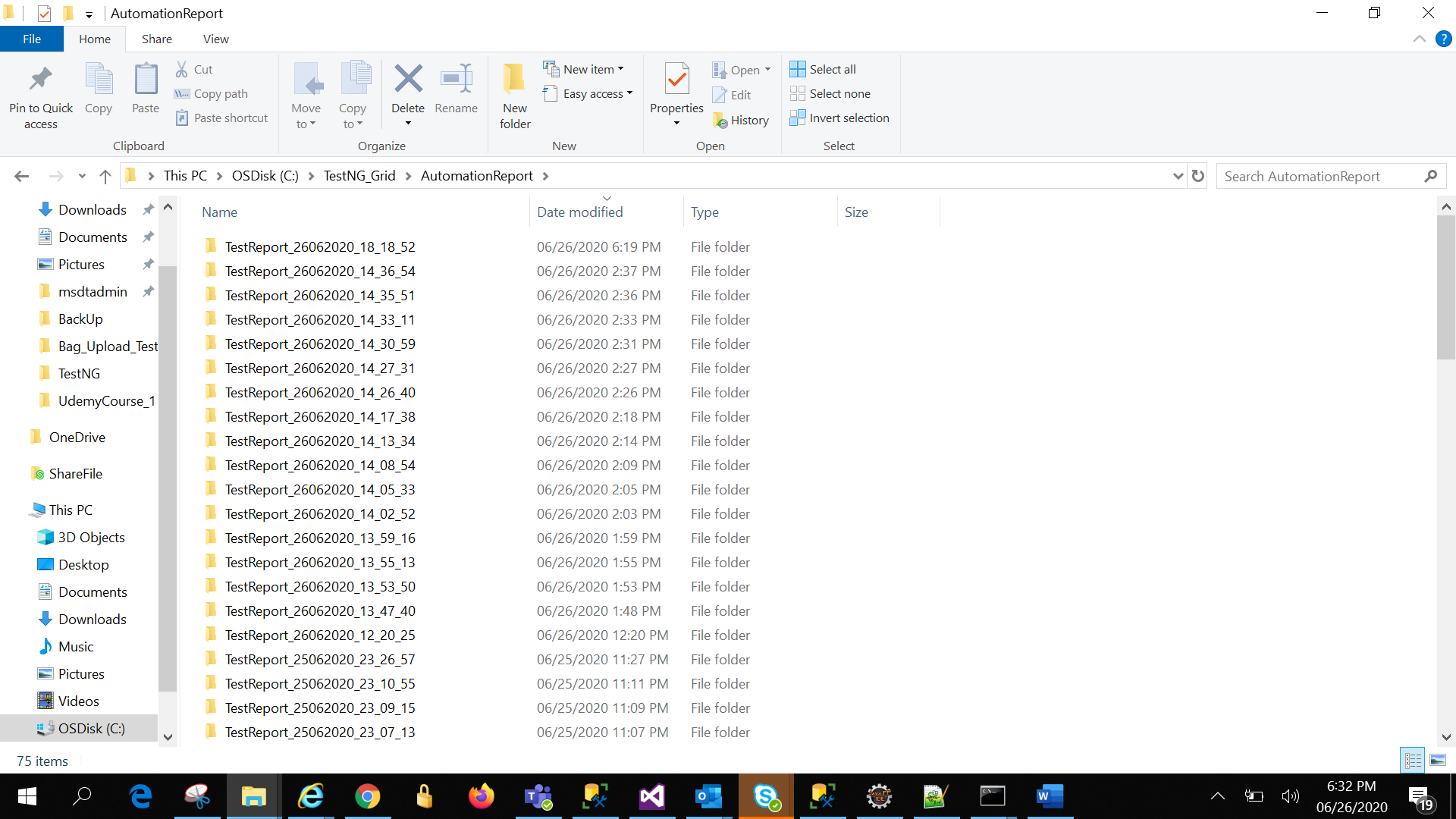
Once Set user can execute by Right Click in eclipse editor->Run As->TestNG

After execution Report are stored with the screen shot if any thig failed in below path

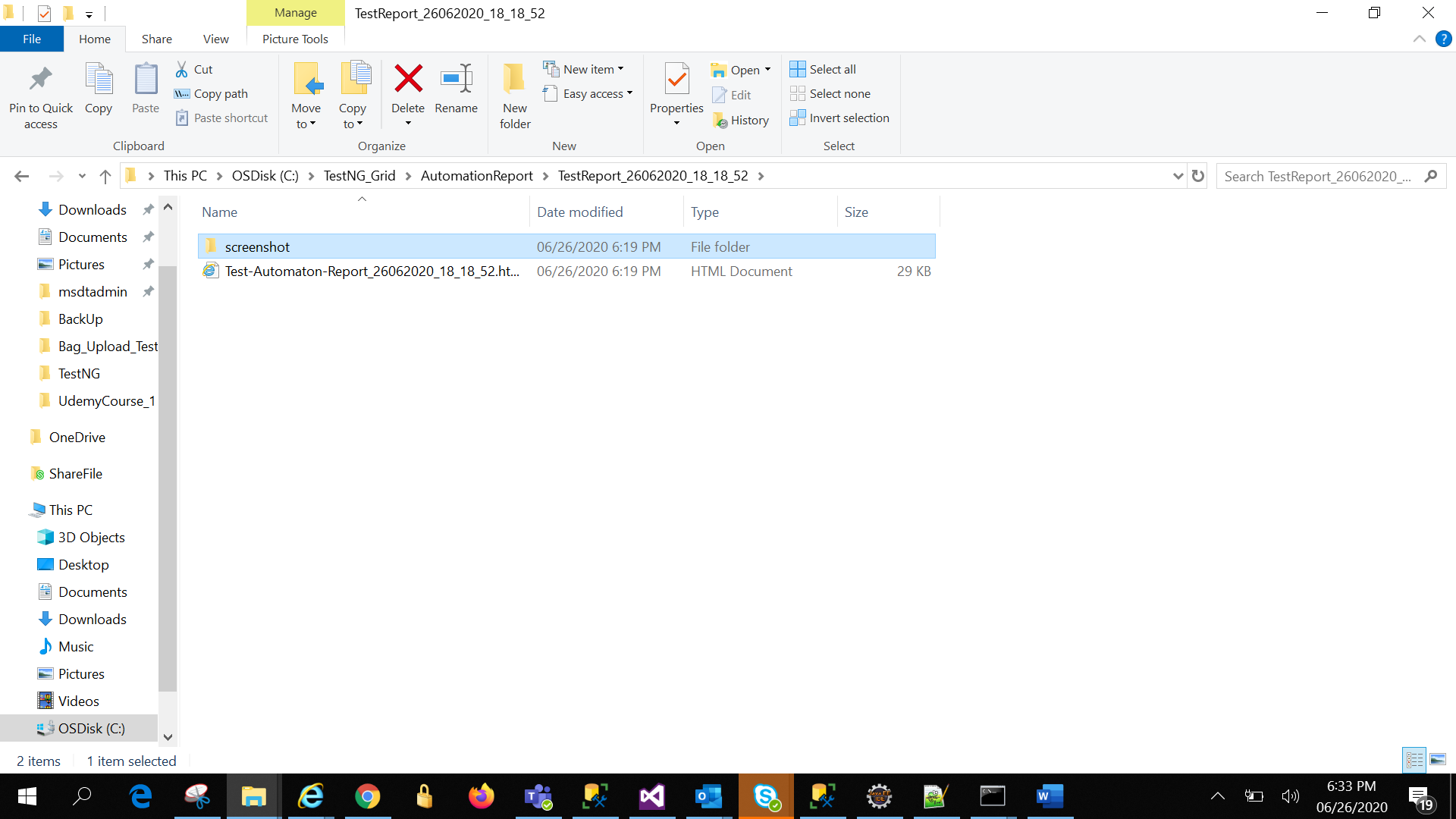
{Project Directory}//Automation Report as below



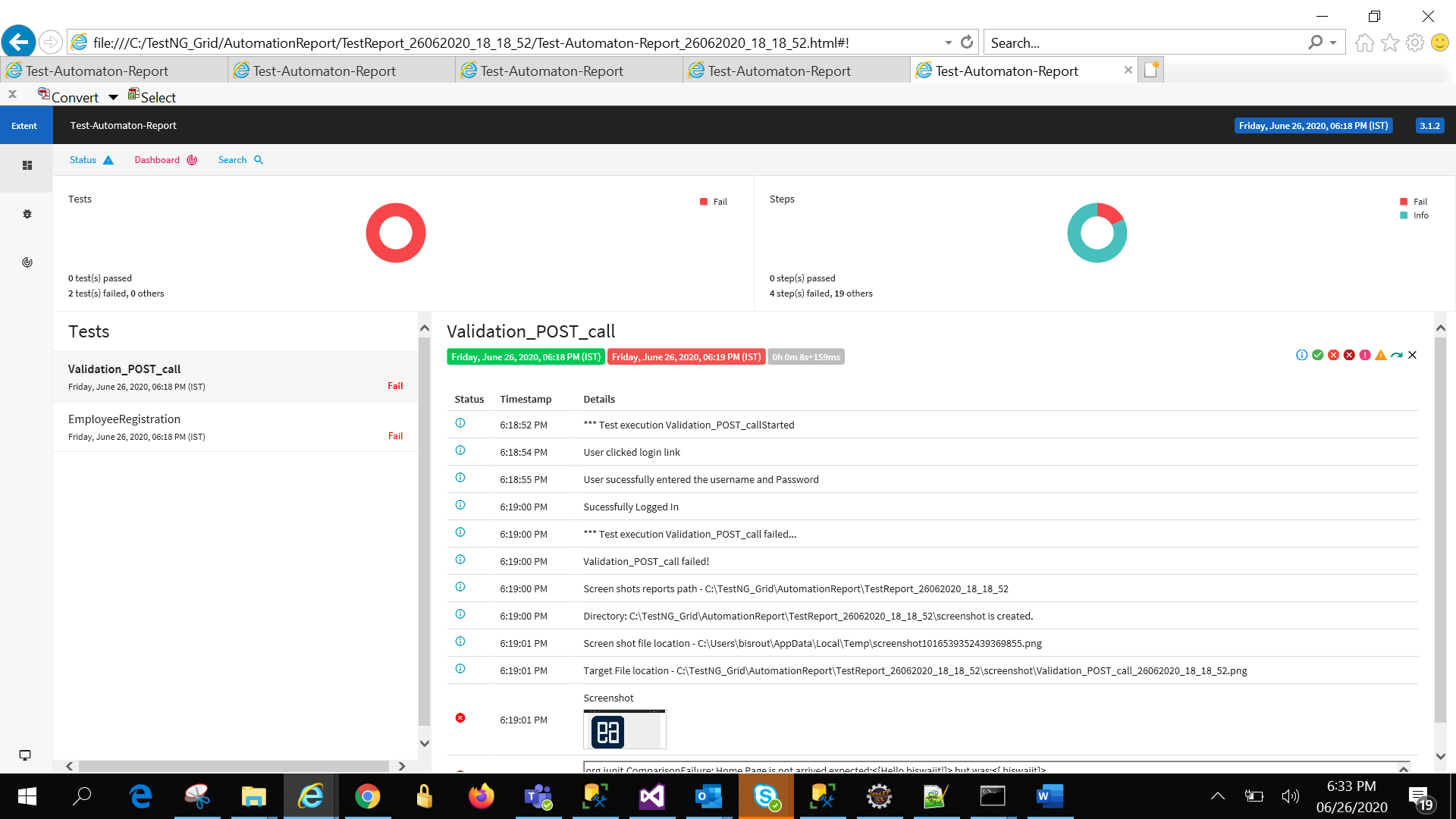
Automation folder looks as below with history of all the execution



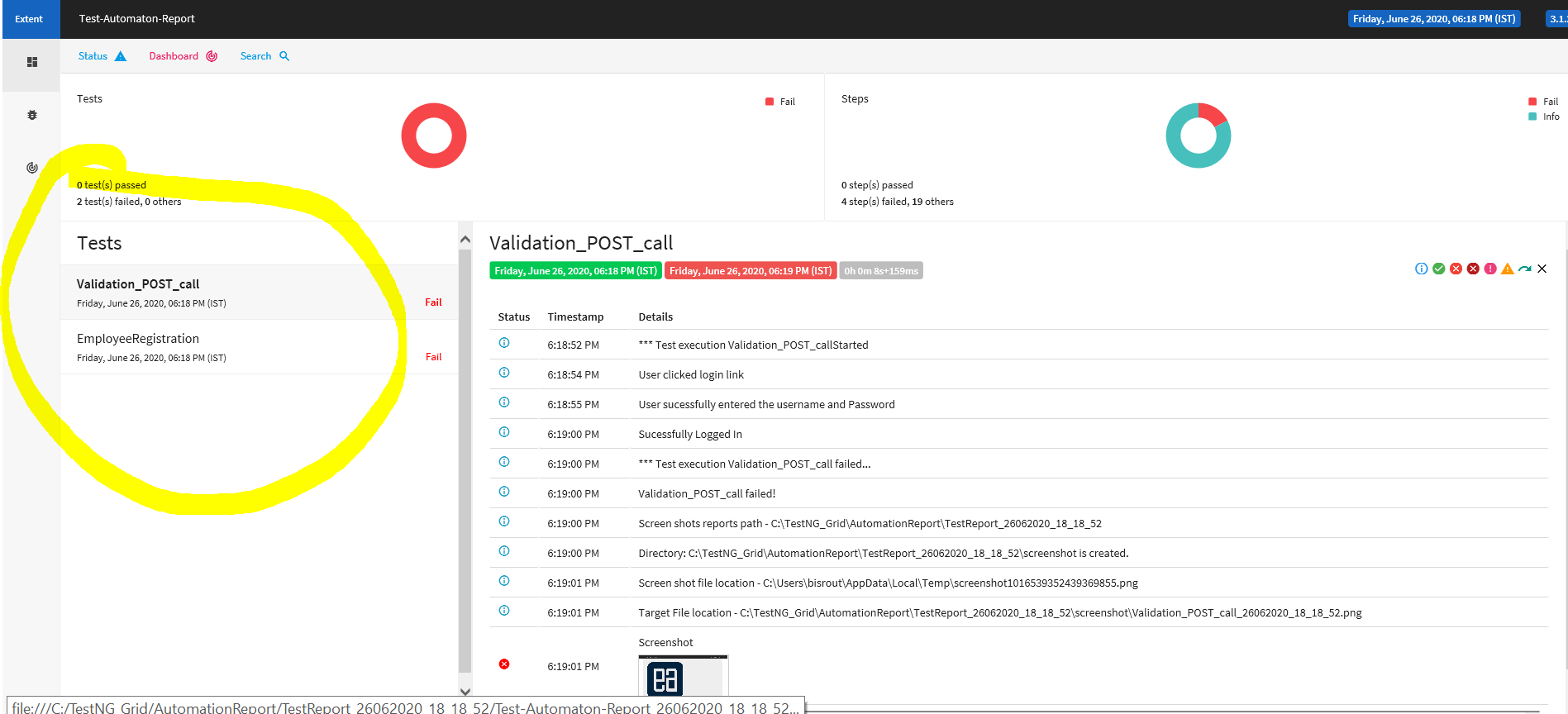
Click the recently executed folder.



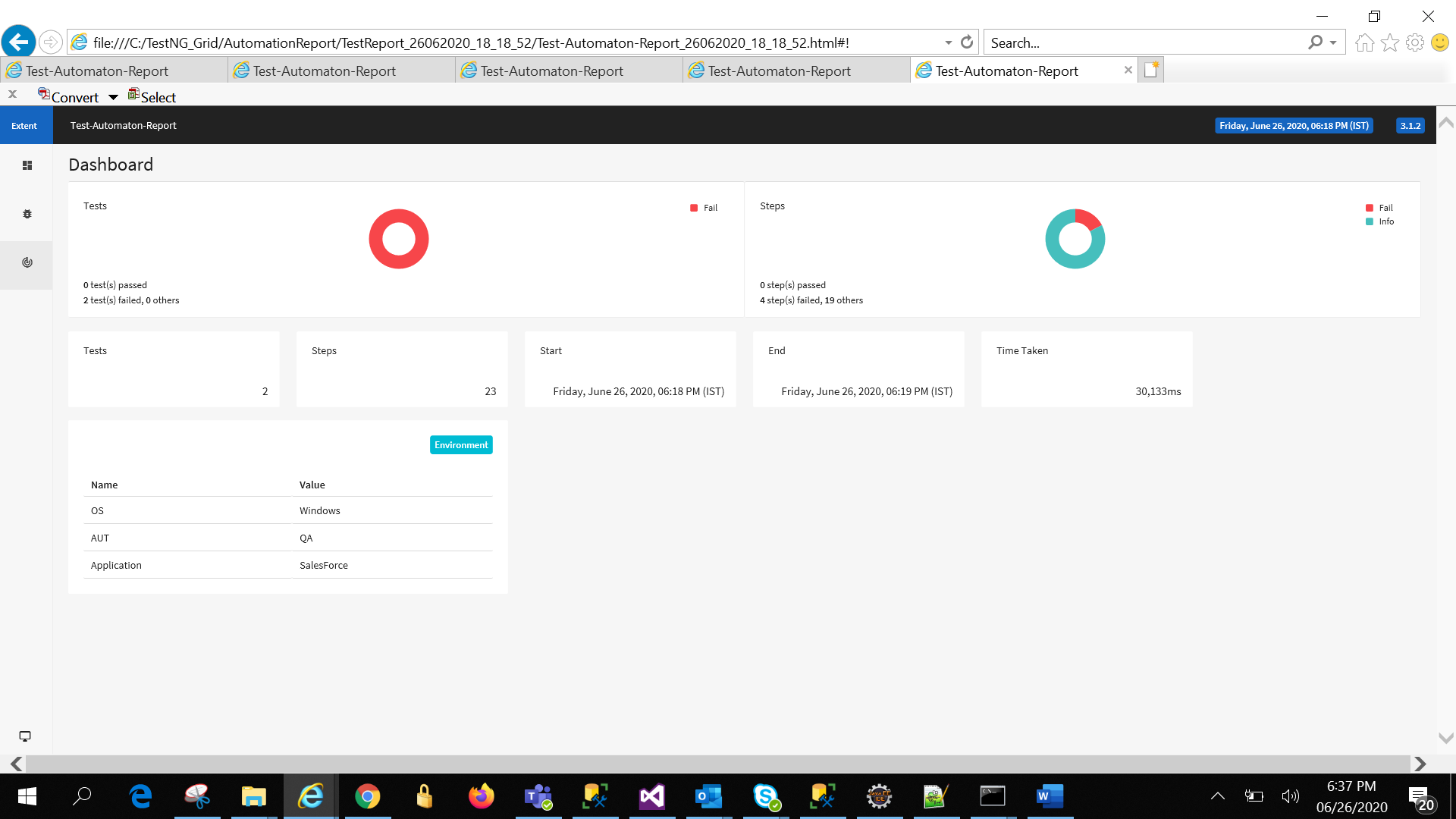
Click Html file.



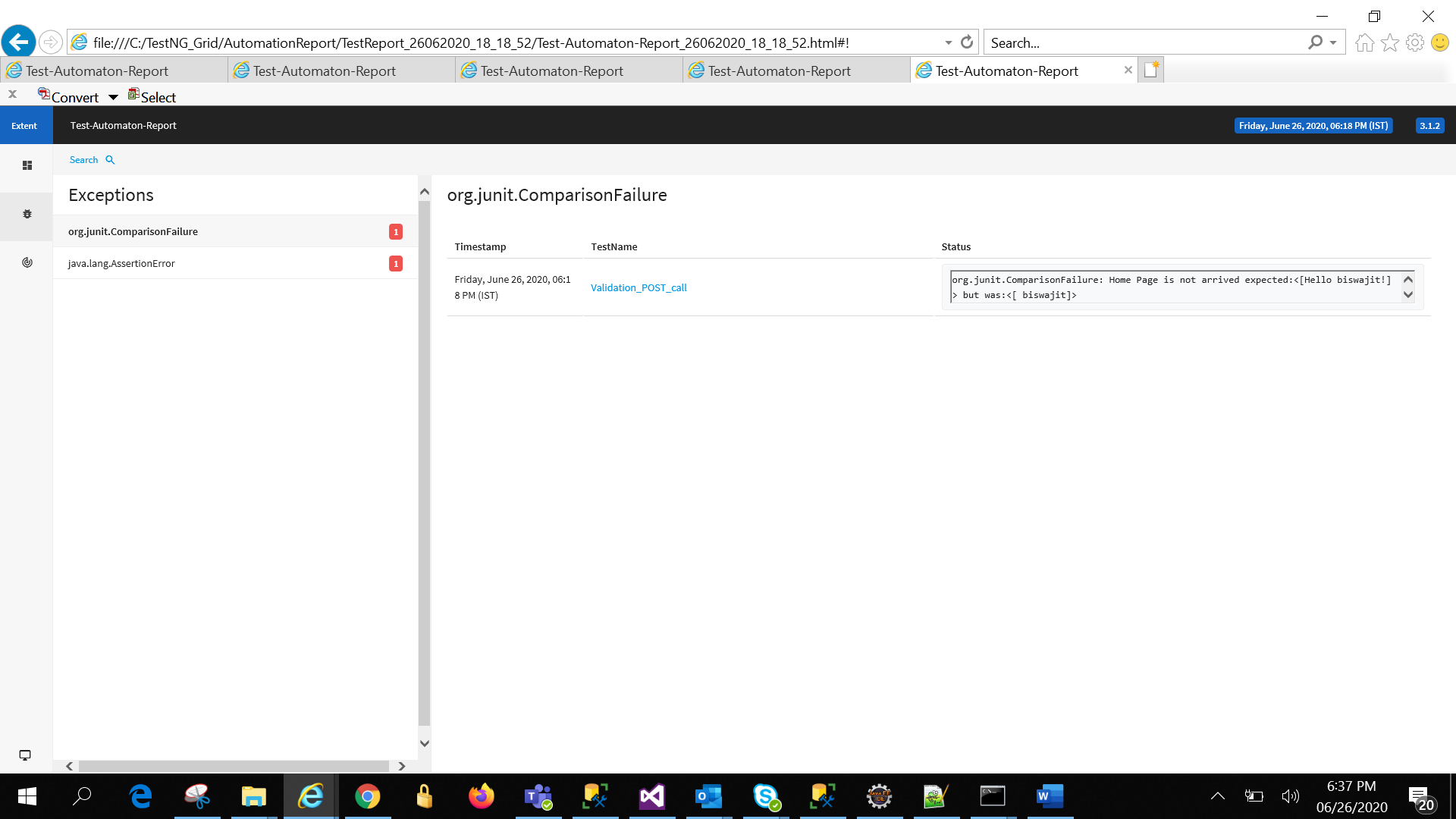
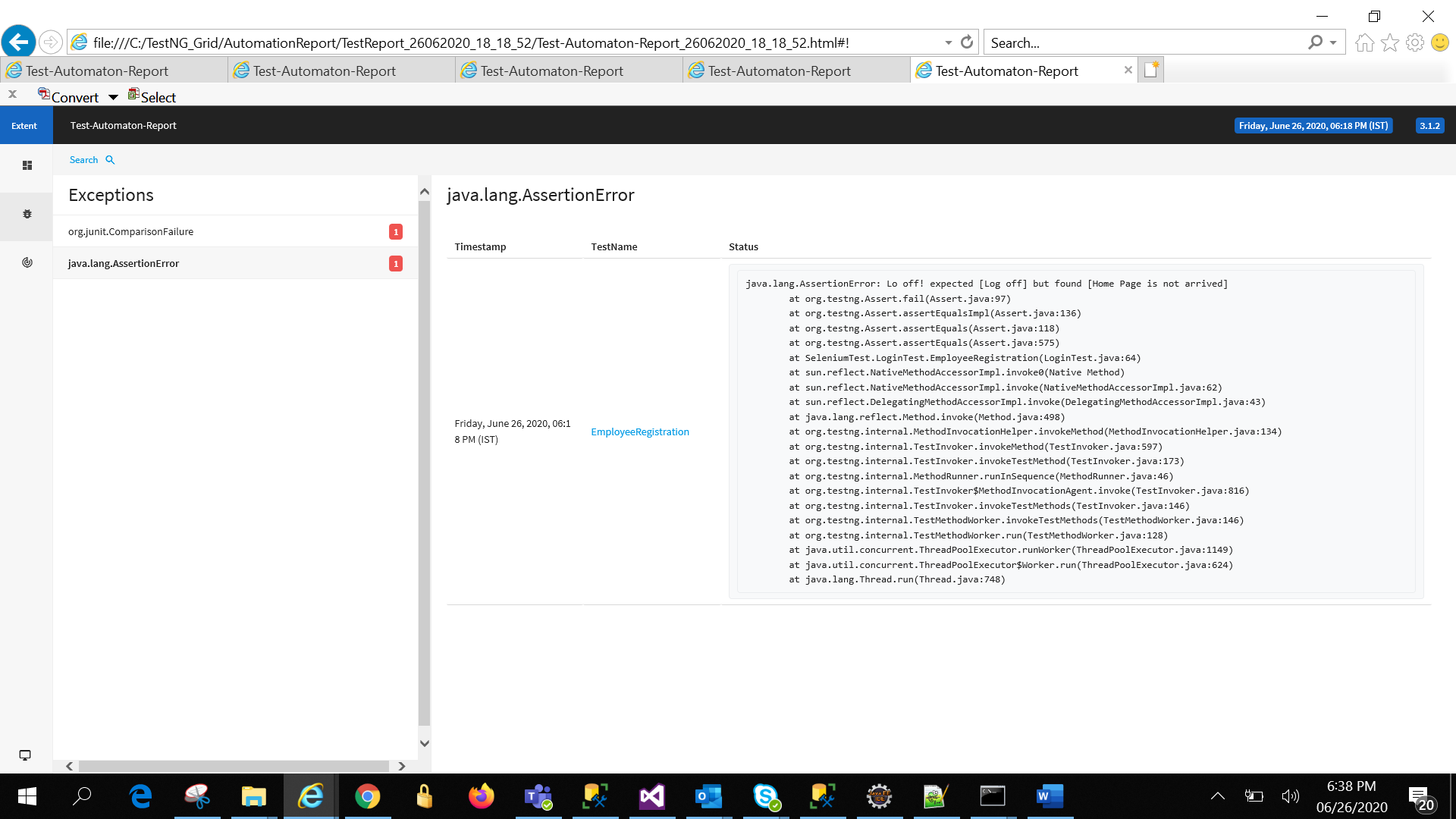
To view test case wise, click the list view in left navigation panel. Screenshot has been attached after failed steps



Dash-Board view



Test case level failure report

## Grid and Multi Machine execution-:

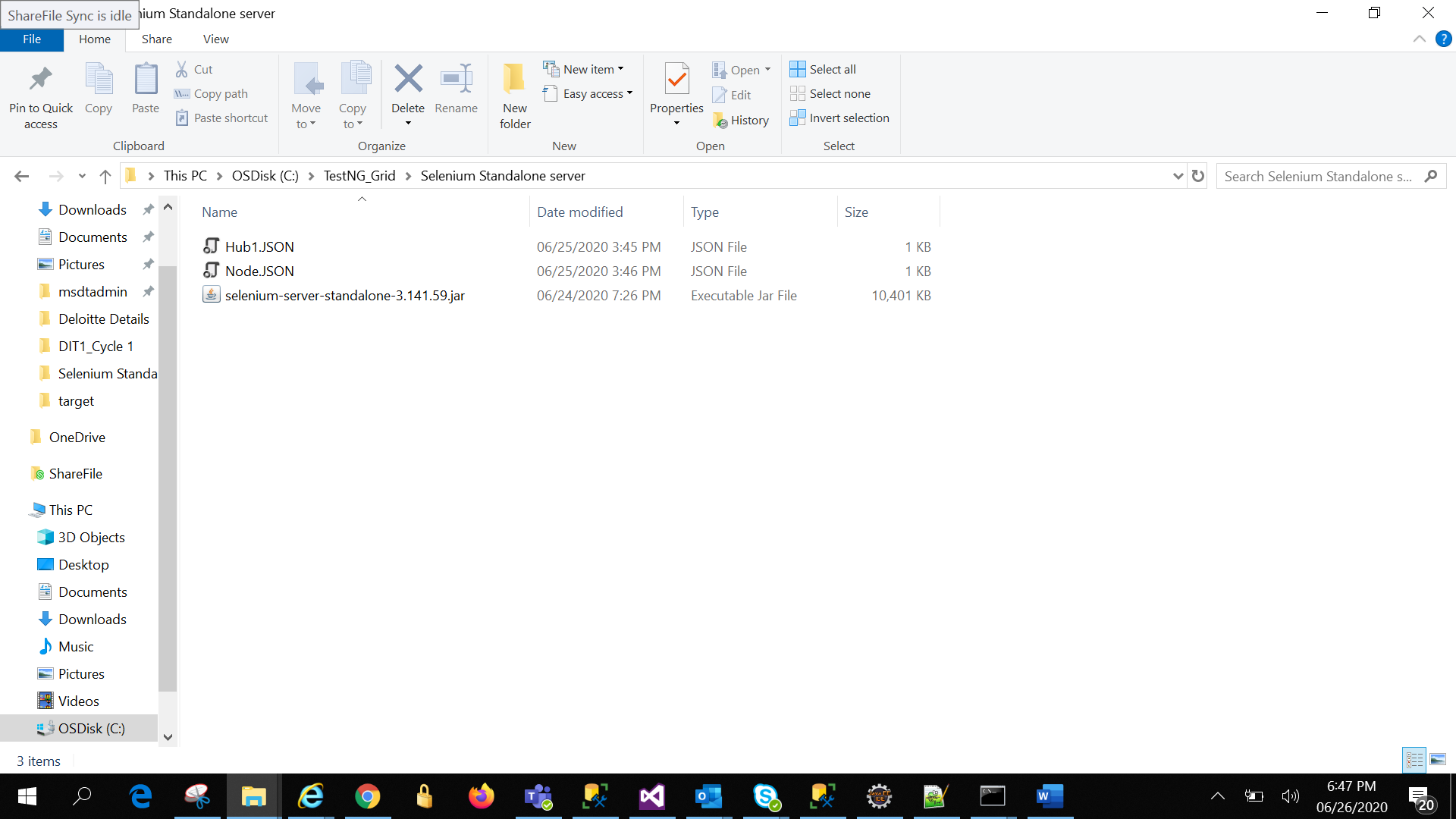
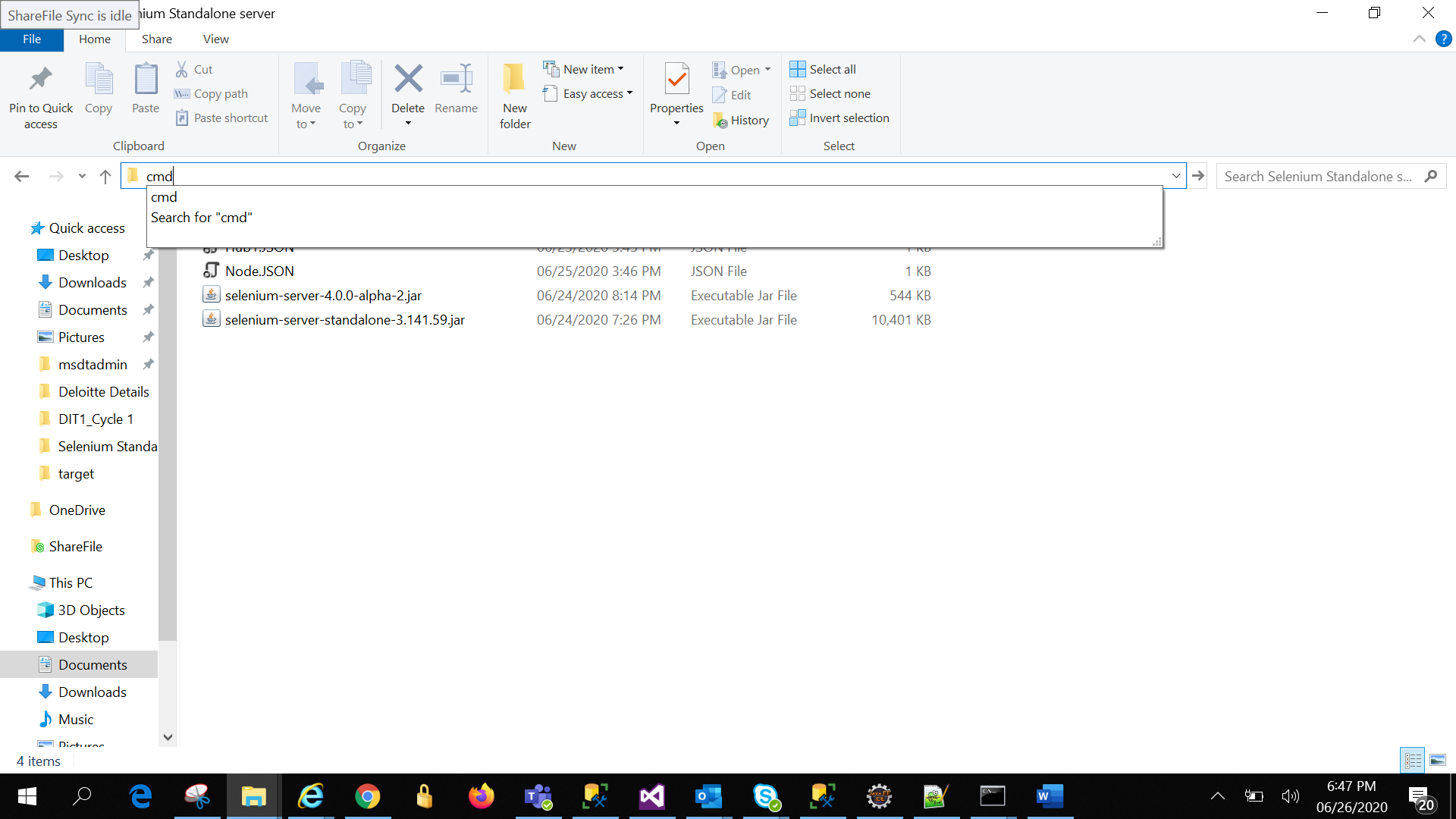
Supporting details – selenium-server-standalone-3.141.59.jar ,Hub config-Hub1.JSON(if required to modify the Hub configuration),Node.JSON(If required to configure node

Steps-:

-Navigate to the folder where the above jar present. Here it is “ProjectDirectories/ Selenium Standalone server”

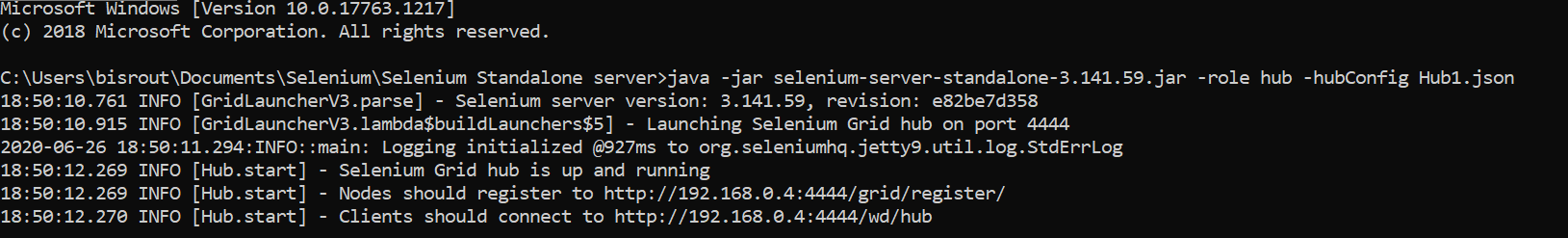
-In folder path edit the path and enter cmd and click enter

-Paste below command to initiate Hub registration

Set Hub with below comand –

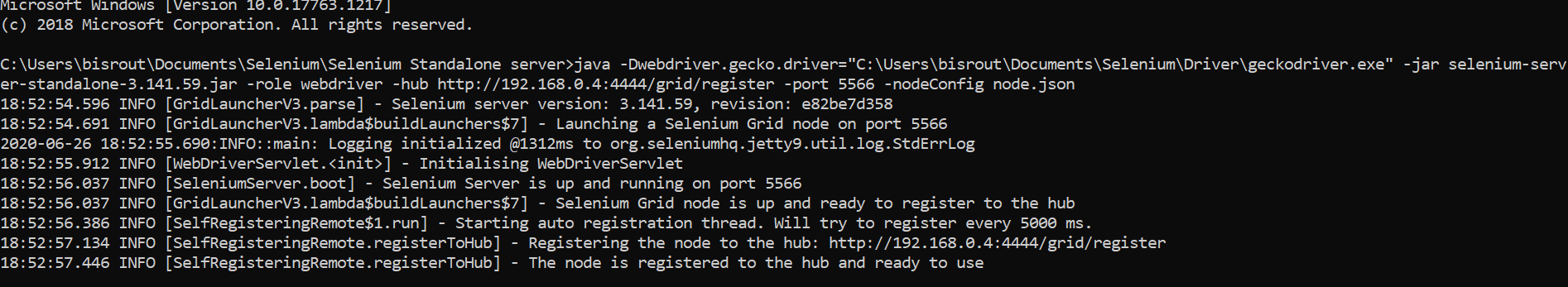
java -jar selenium-server-standalone-3.141.59.jar -role hub -hubConfig Hub1.json-click enter

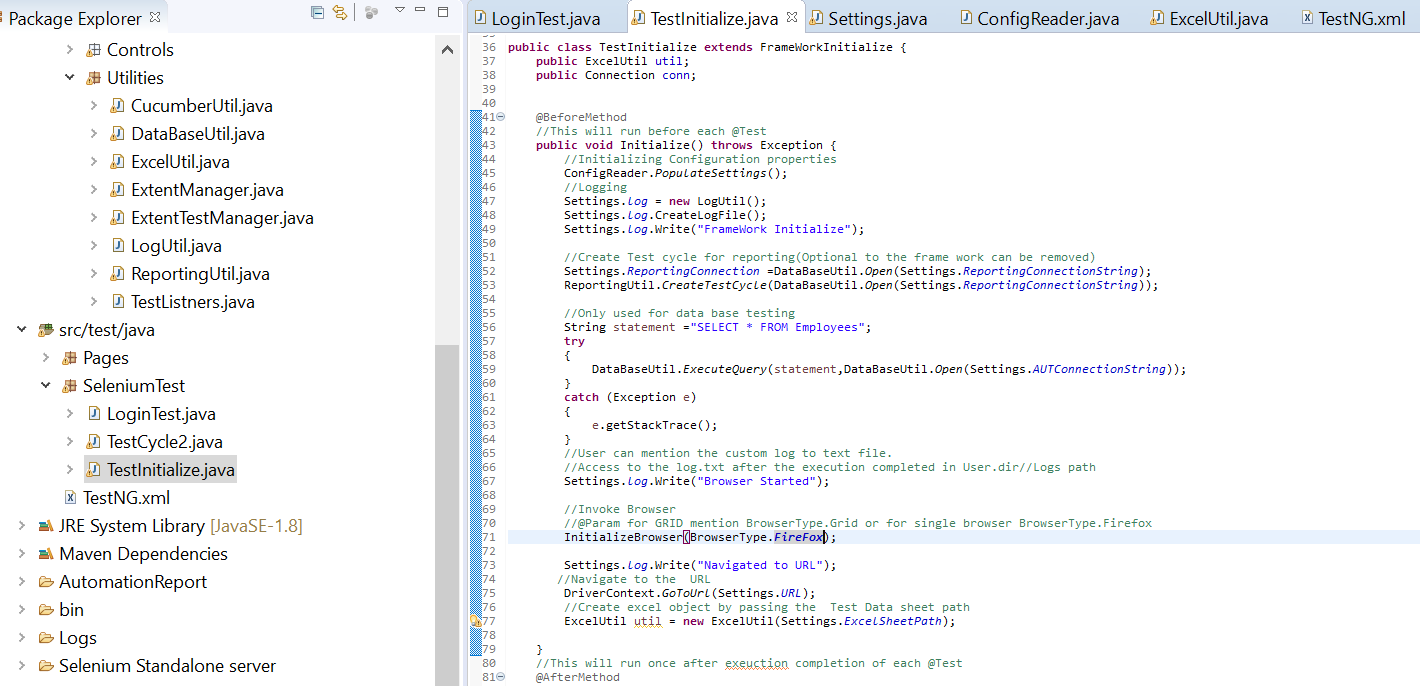


Set register to the hub the same way by pasting below command to the virtual machine where you want to register node .You can choose the local as well

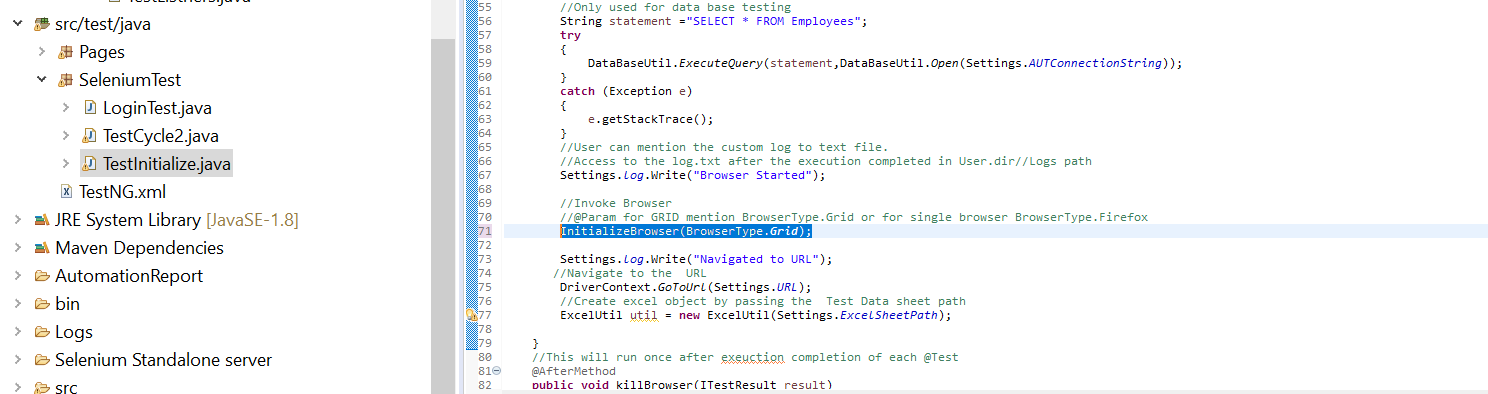
java -Dwebdriver.gecko.driver="C:\Users\bisrout\Documents\Selenium\Driver\geckodriver.exe" -jar selenium-server-standalone-3.141.59.jar -role web driver -hub [http://192.168.0.4:4444/grid/register -port 5566 -nodeConfig node.json](http://192.168.0.4:4444/grid/register%20-port%205566%20-nodeConfig%20node.json)

After doing this system will display node is successfully registered.

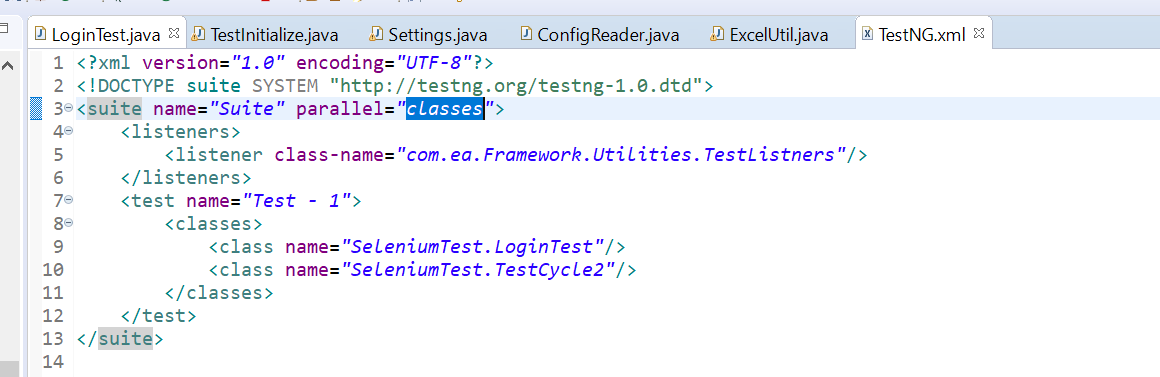


-Next move to ‘TestInitialize.Java’ in the eclipse and updates the 

-Update to BrwserType.FireFox to BrowserType.Grid



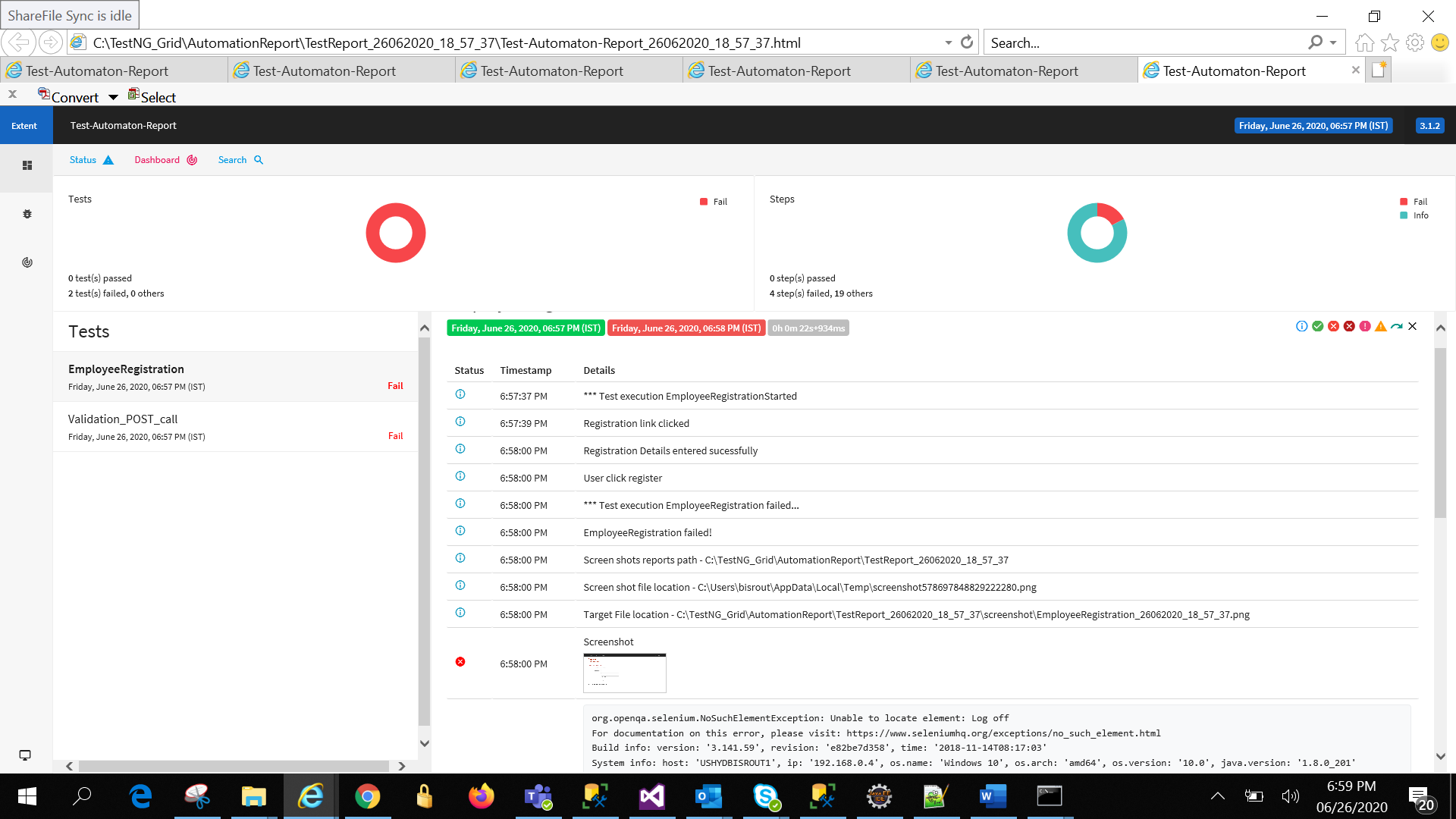
-Next put parallel-‘Class’/’Method’ in to TestNG.xml



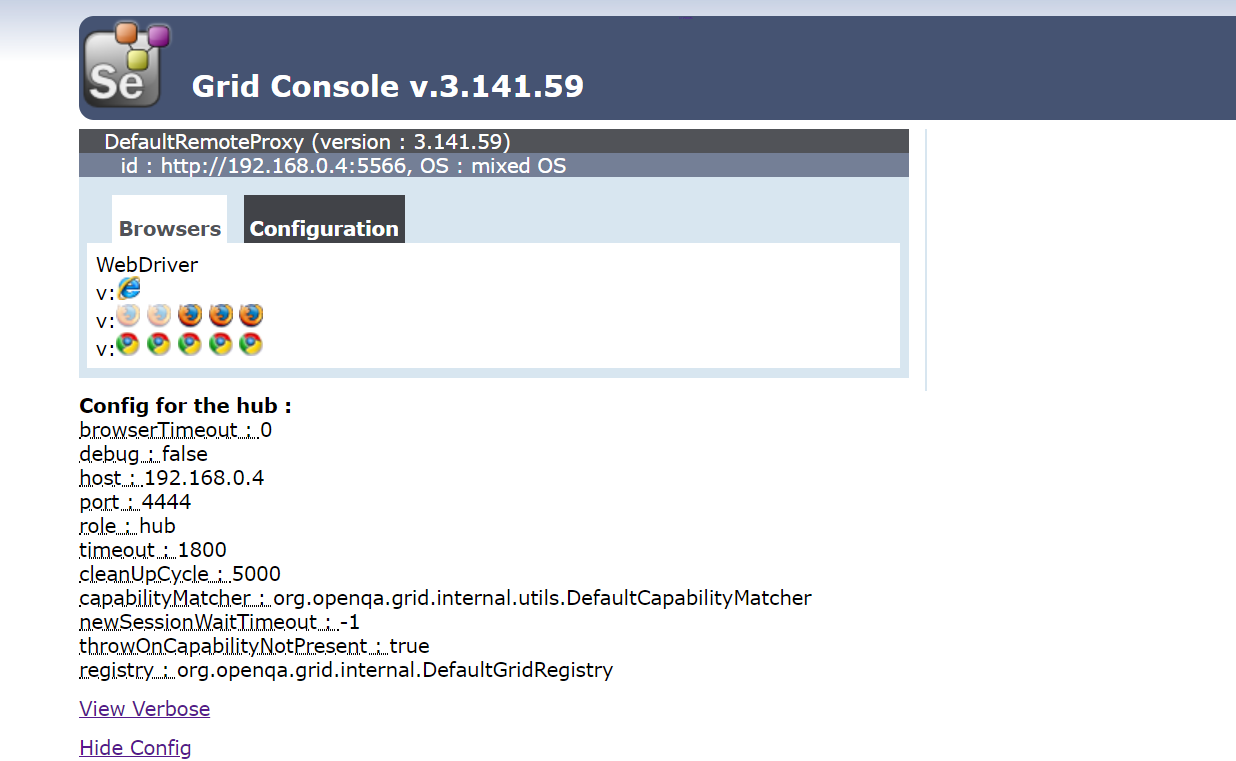
-Click RunAs-TestNG and system will trigger 2 Firefox browsers



Once completed it will generate the html report in the same manner.



-Grid console will be displayed as below by navigating to ‘HTTP://localhost:4444/grid/console’



-User can modify the hub with hub.json as below by modifying the values for the variables.

{

"port": 4444,

"newSessionWaitTimeout": -1,

"servlets" : [],

"withoutServlets": [],

"custom": {},

"capabilityMatcher": "org.openqa.grid.internal.utils.DefaultCapabilityMatcher",

"registry": "org.openqa.grid.internal.DefaultGridRegistry",

"throwOnCapabilityNotPresent": true,

"cleanUpCycle": 5000,

"role": "hub",

"debug": false,

"browserTimeout": 0,

"timeout": 1800

}

-To configure Node then Node.JSON will be as below

{

"capabilities":

[

{

"browserName": "firefox",

"marionette": true,

"maxInstances": 5,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "chrome",

"maxInstances": 5,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "internet explorer",

"platform": "WINDOWS",

"maxInstances": 1,

"seleniumProtocol": "WebDriver"

},

{

"browserName": "safari",

"technologyPreview": false,

"platform": "MAC",

"maxInstances": 1,

"seleniumProtocol": "WebDriver"

}

],

"proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",

"maxSession": 5,

"port": -1,

"register": true,

"registerCycle": 5000,

"hub": "http://localhost:4444",

"nodeStatusCheckTimeout": 5000,

"nodePolling": 5000,

"role": "node",

"unregisterIfStillDownAfter": 60000,

"downPollingLimit": 2,

"debug": false,

"servlets" : [],

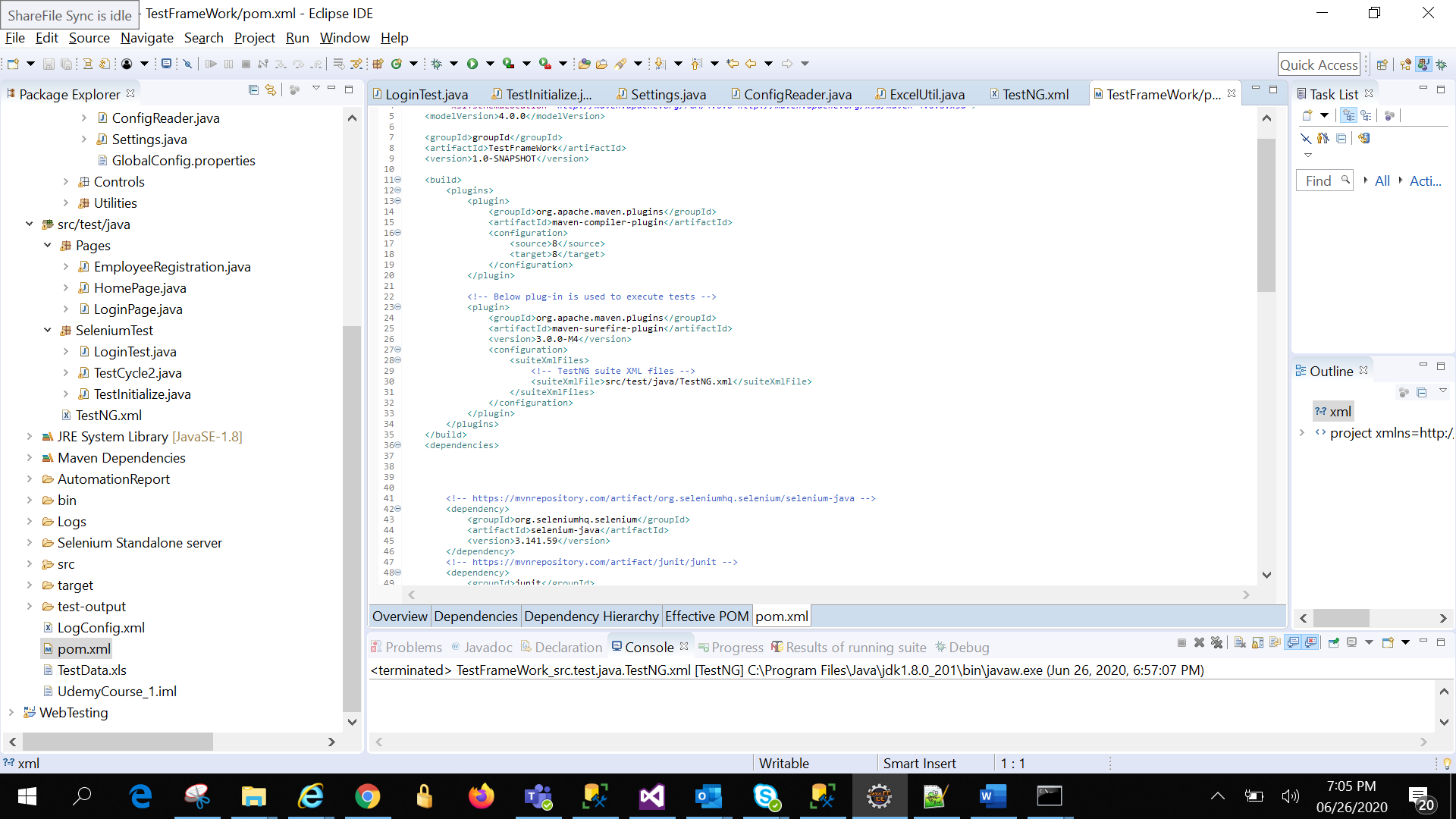
"withoutServlets": [],

"custom": {}

}

## POM.xml

-pom.xml comes with the frame work so user need not to design specifically. Just modify as per the requirement. Once saved it will automatically download all the jars to the //.m2 folder



.m2 folder

