Framework: A well defined structure which has capability to handle all the required tools to execute a project from its own is called Framework.

There are 3 types of framework:

1. Data Driven framework: In which on the basis of data we are driving the test cases.

2. Keyword driven: A particular key word decides the flow of execution.

3. Hybrid: It is the combination of data driven and Keyword driven.

For creation of framework we are using Maven Project as a build automation tool. Reason:

It is to help in downloading the jars after we mention in pom.xml based on which we can download the jars for different version as per the requirement.

Features / Advantage:

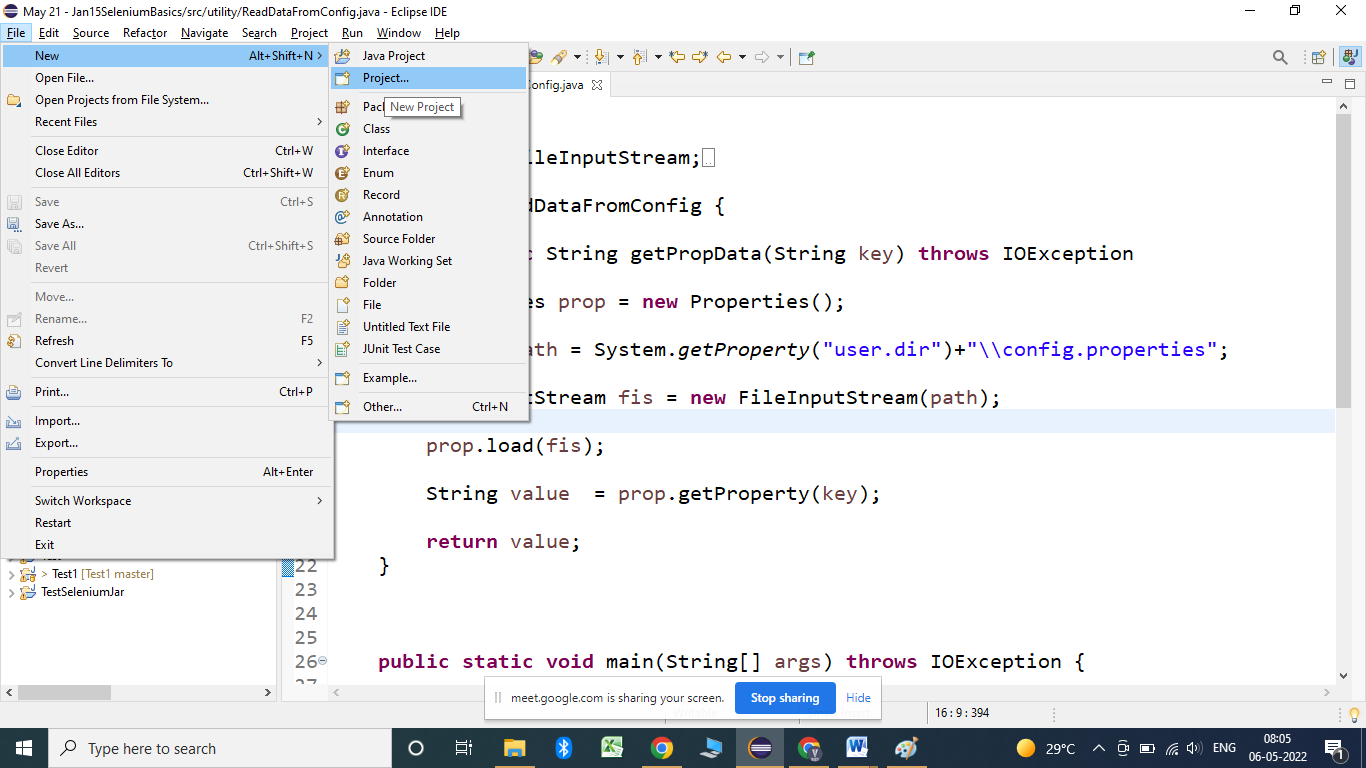
1. Functionality to choose the version for building the project.

2. A well defined structure available so that we can design the framework on the basis of page object model.

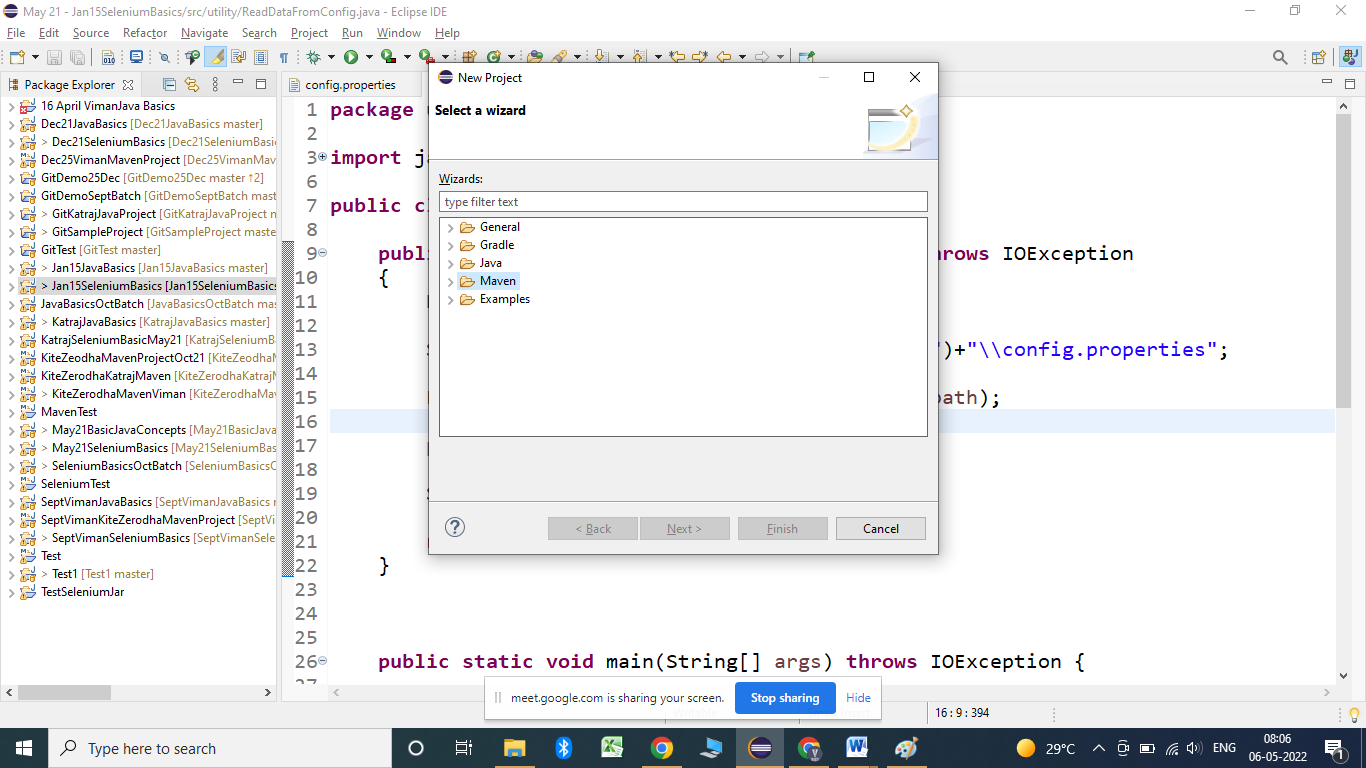
3. Dependency management updation / deletion is very easy.

Steps to create and configure Maven Project:

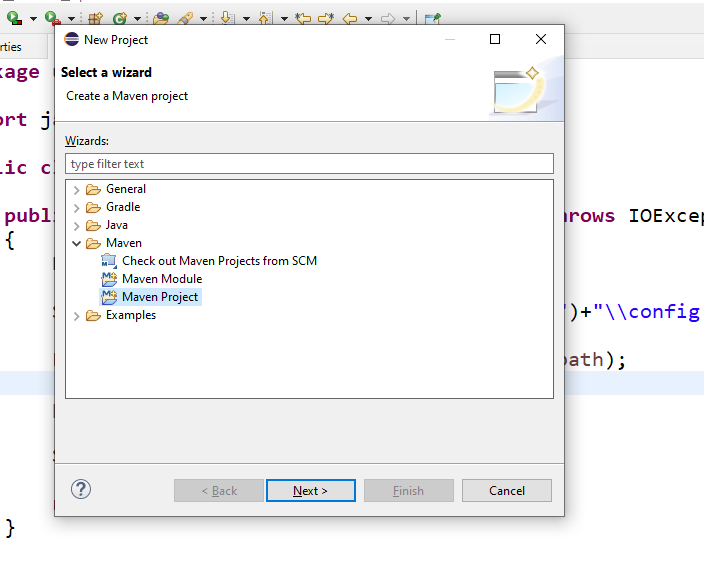
1. Go to File- New –Project



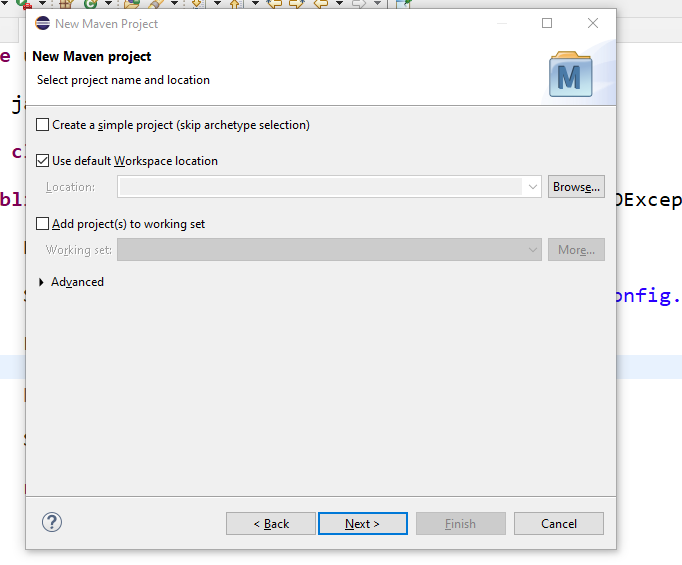
2. Click on Project



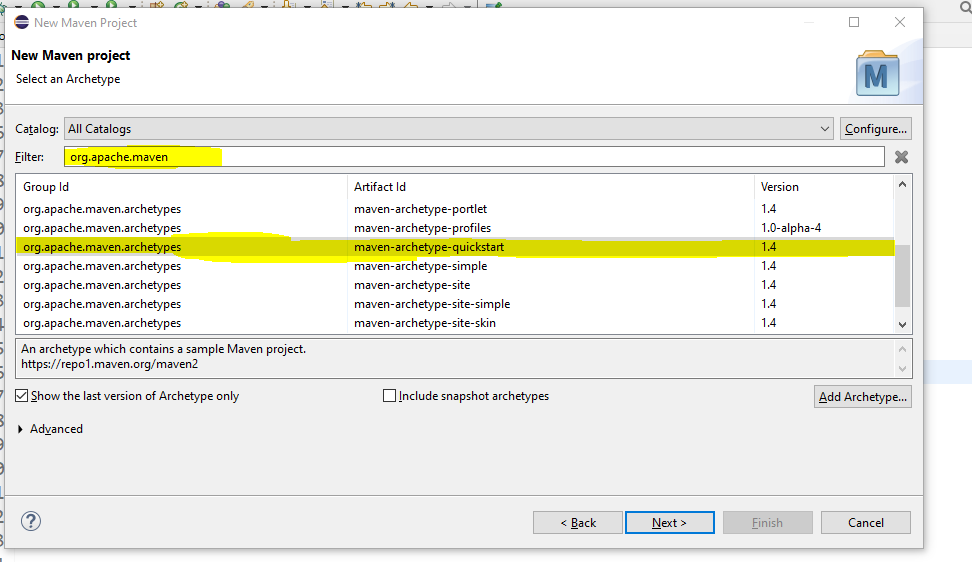
3. Click on Maven and Maven Project:



4. Click on Next button:



5. Click on Next button:



6. Type org.apache.maven and select quickstart option and click on Next button.

7. Enter same group id and artifact id and click on Finish. Then Maven project will get created.

Configuring pom.xml file:

To add the dependency go to following url:

<https://mvnrepository.com/repos/central>

Copy the dependency into the pom.xml file:

<!-- https://mvnrepository.com/artifact/org.testng/testng -->

<dependency>

<groupId>org.testng</groupId>

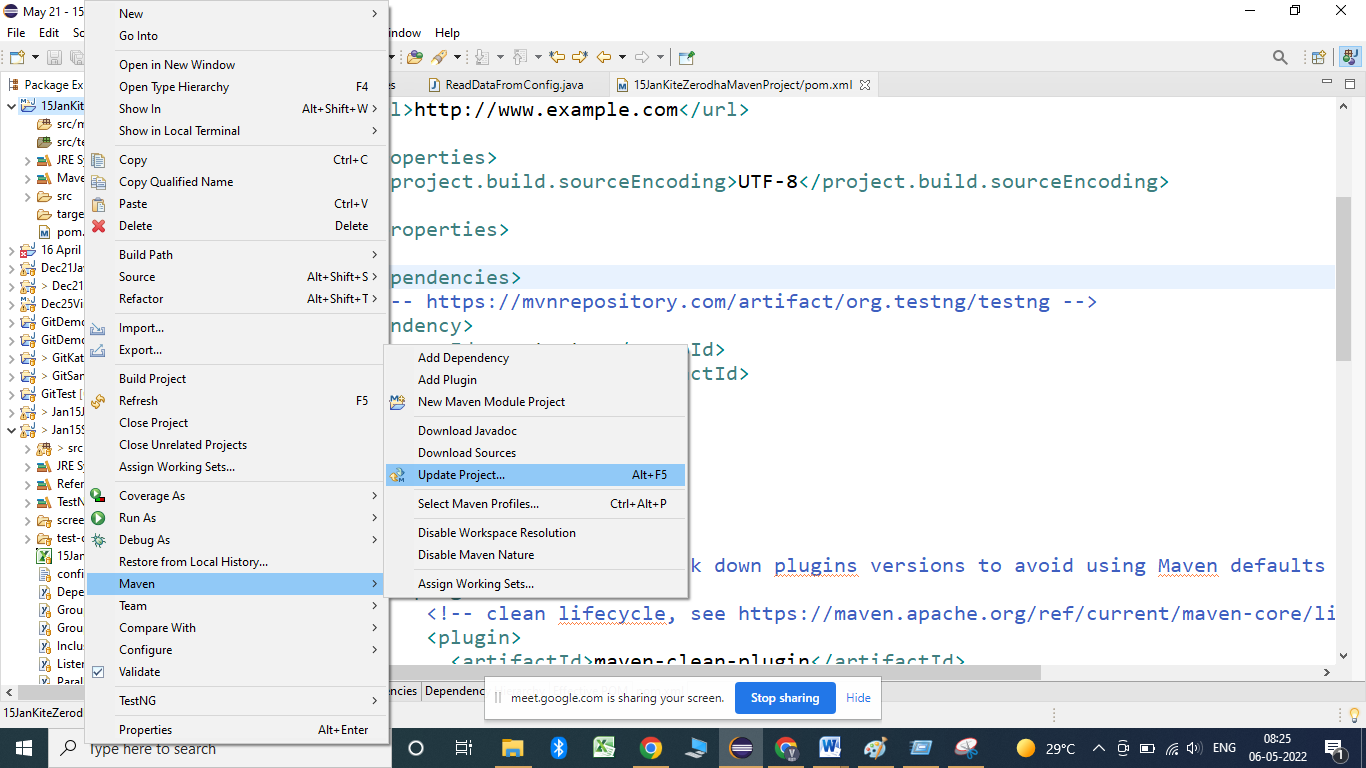
<artifactId>testng</artifactId>

<version>7.5</version>

<scope>test</scope>

</dependency>

After the addition of dependency Update the maven project by:



Add webdriver manager dependency :

<dependency>

<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>4.0.0</version>

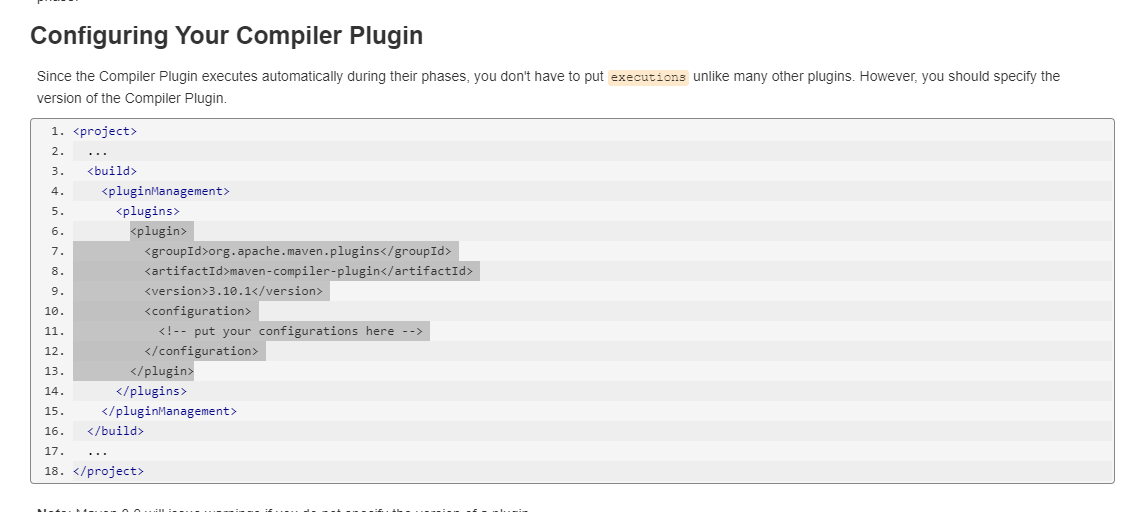
</dependency>

Add maven compilar plugin and Maven surefire plugin:

1. maven-compiler-plugin:

Go to the url:

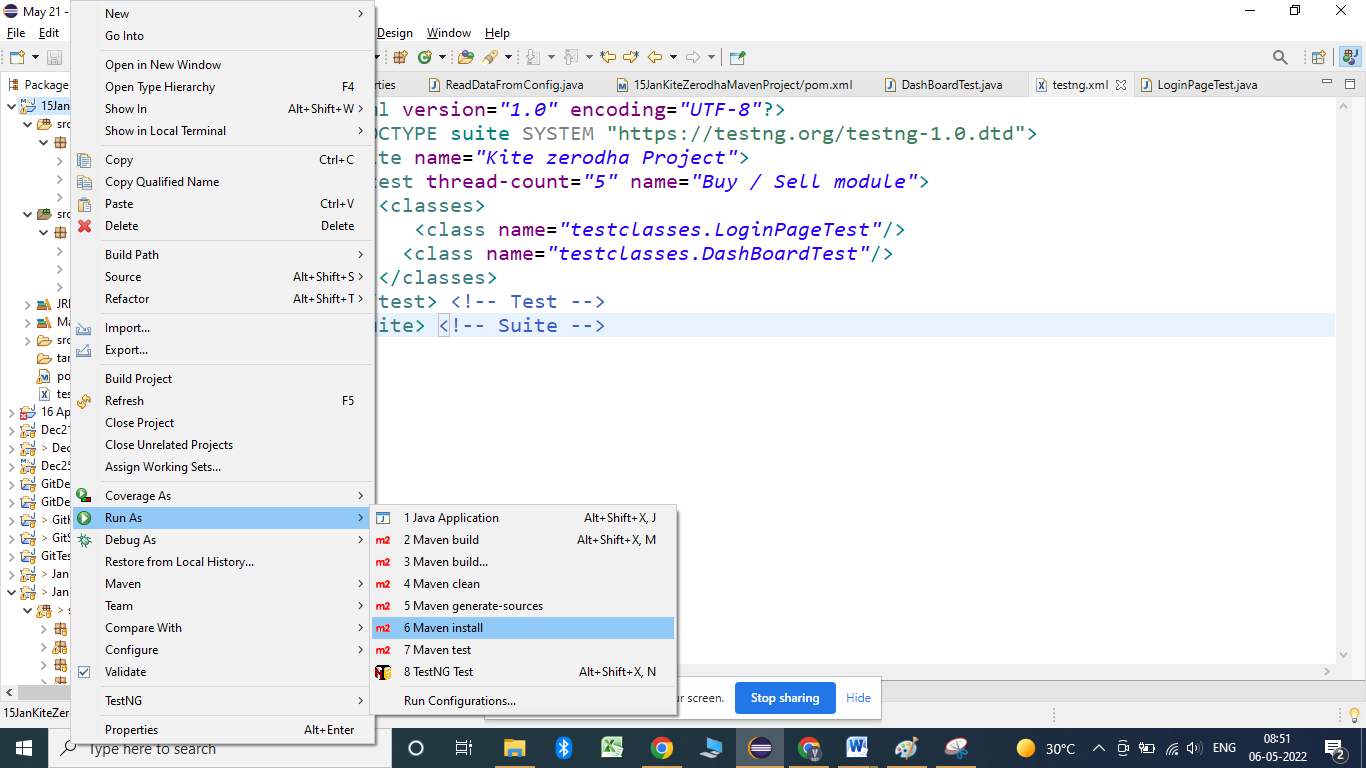
<https://maven.apache.org/plugins/maven-compiler-plugin/usage.html>



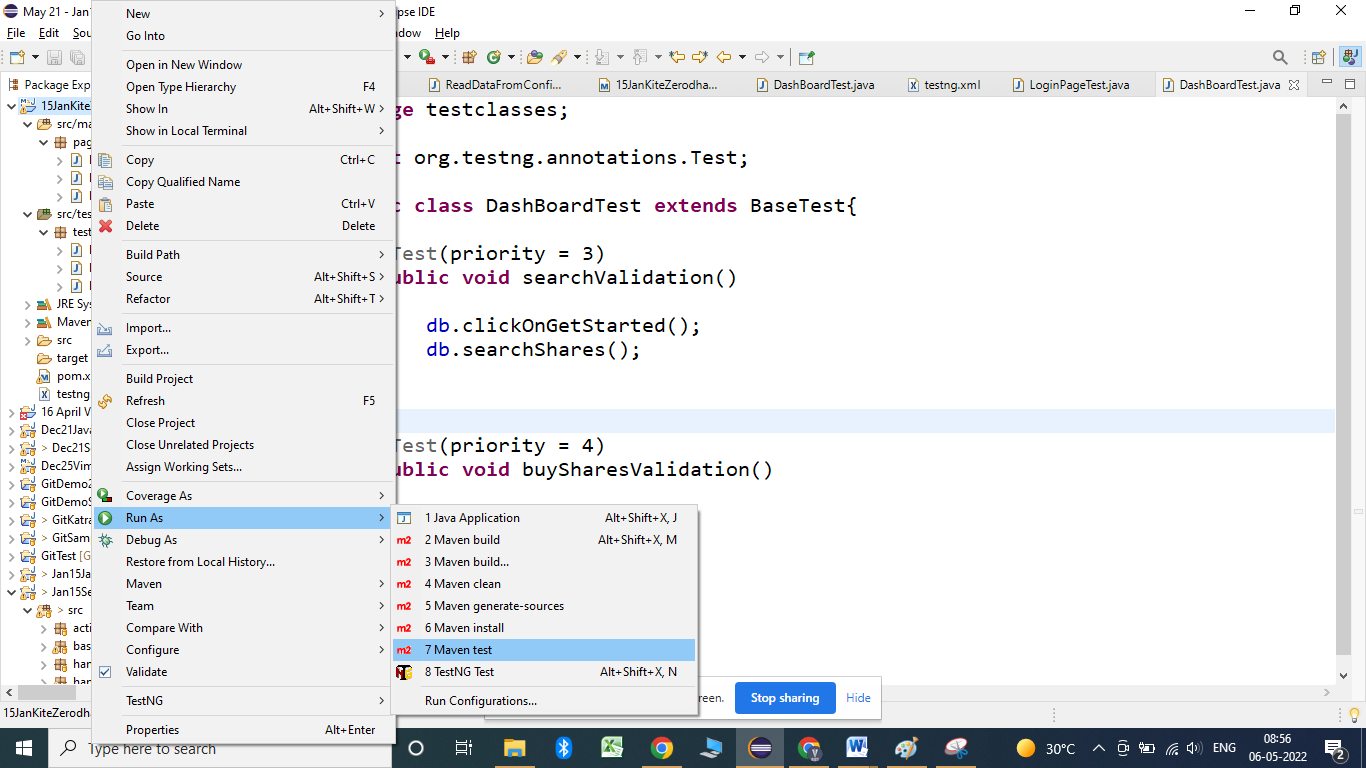
2. Add maven sure fire plugin:

<https://maven.apache.org/surefire/maven-surefire-plugin/usage.html>

To Execute the maven Project for the first time :



Later on we can execute the project by:



To use webdriver manager:

@BeforeSuite

**public** **void** initBrowser() {

// System.setProperty("webdriver.chrome.driver", "E:\\desktop\\Katraj\\15 Jan\\Selenium\\Chromedriver 101\\chromedriver.exe");

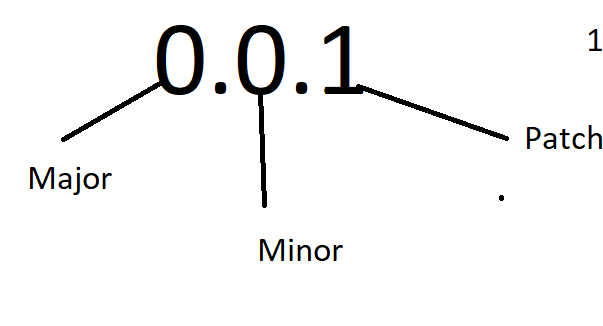
WebDriverManager.*chromedriver*().setup();

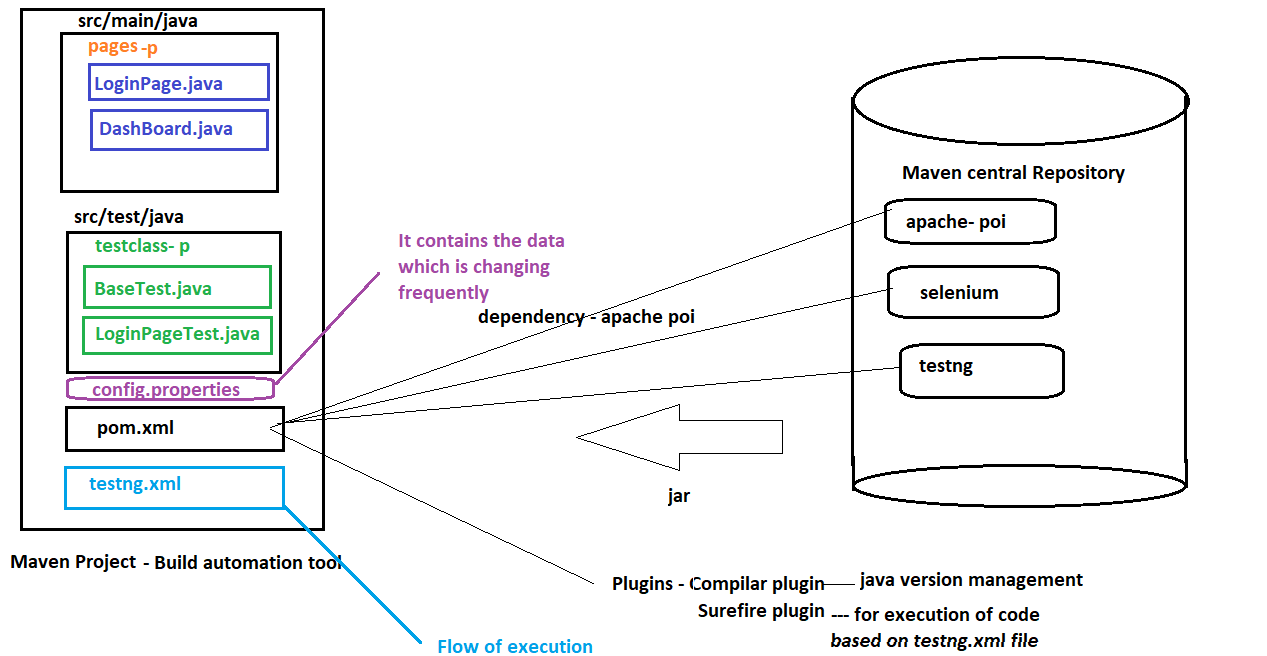
*driver* = **new** ChromeDriver();// chrome browser will get open

*driver*.manage().window().maximize();// to maximize the window

*driver*.get("https://kite.zerodha.com/");

}





Implementing extent reports:

Extent reports can be implemented in 2 steps:

1. A class defining the look and feel of the report.

2. A class defining the listeners for the test cases.

We have to use following dependencies for extent report:

<https://mvnrepository.com/artifact/com.aventstack/extentreports/4.0.9>

<!-- https://mvnrepository.com/artifact/commons-io/commons-io -->

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.6</version>

</dependency>

<!-- https://mvnrepository.com/artifact/com.aventstack/extentreports -->

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>4.0.9</version>

</dependency>

**public** **class** ExtentReportGen {

**static** ExtentReports *extent*;

**public** **static** ExtentReports extentReportGenerator()

{

String path = System.*getProperty*("user.dir")+"//reports//kitezerodhareport.html";

ExtentHtmlReporter reporter = **new** ExtentHtmlReporter(path);

reporter.config().setTheme(Theme.***DARK***);

reporter.config().setReportName("Zerodha Report");

*extent* = **new** ExtentReports();

*extent*.attachReporter(reporter);

*extent*.setSystemInfo("Project name", "Kite Zerodha");

*extent*.setSystemInfo("Environment", "SIT");

*extent*.setSystemInfo("Executed by", "15 Jan Batch");

**return** *extent*;

}

}

**public** **class** ListenerClass **extends** BaseTest **implements** ITestListener

{

**public** **static** ExtentTest *test*;

ExtentReports extent = ExtentReportGen.*extentReportGenerator*();

@Override

**public** **void** onTestStart(ITestResult result) {

System.***out***.println("Test case started :"+result.getName());

*test* = extent.createTest(result.getName());

}

@Override

**public** **void** onTestSuccess(ITestResult result) {

System.***out***.println("Test case passed :"+result.getName());

*test*.log(Status.***PASS***, "Test case is passed");

}

@Override

**public** **void** onTestFailure(ITestResult result) {

System.***out***.println("Test case Failed :"+result.getName());

*test*.fail(result.getThrowable());

**try** {

*test*.addScreenCaptureFromPath(ScreenShot.*captureScreenshotWithPath*(*driver*,result.getName() ), "Captured screenshot");

} **catch** (IOException e) {

System.***out***.println("Exception occured while taking screenshot");

}

}

@Override

**public** **void** onTestSkipped(ITestResult result) {

System.***out***.println("Test case got skipped :"+result.getName());

}

@Override

**public** **void** onTestFailedButWithinSuccessPercentage(ITestResult result) {

}

@Override

**public** **void** onTestFailedWithTimeout(ITestResult result) {

}

@Override

**public** **void** onStart(ITestContext context) {

System.***out***.println("Test started :"+context.getName());

}

@Override

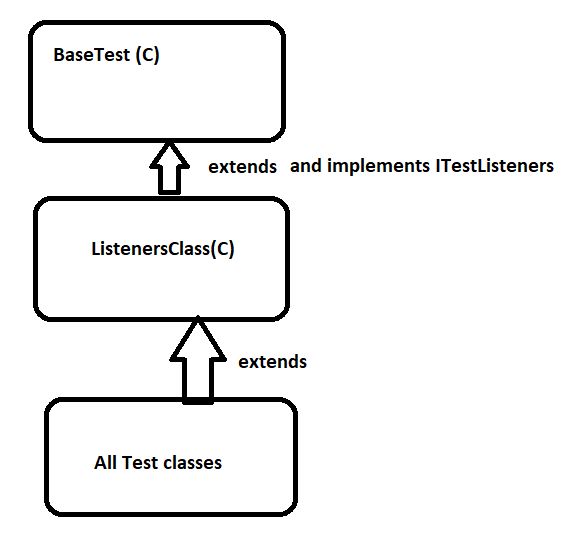
**public** **void** onFinish(ITestContext context) {

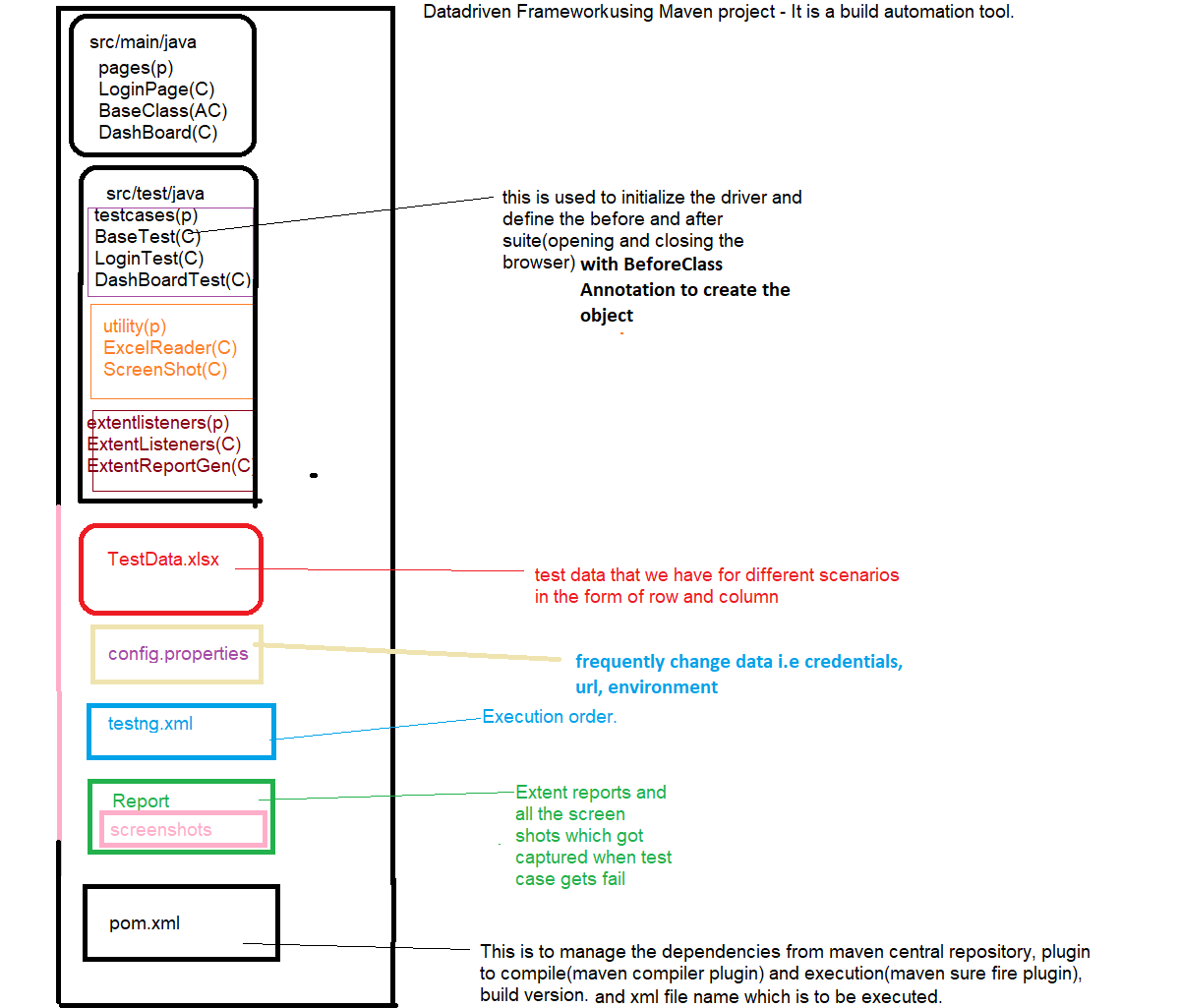
System.***out***.println("Test completed :"+context.getName());

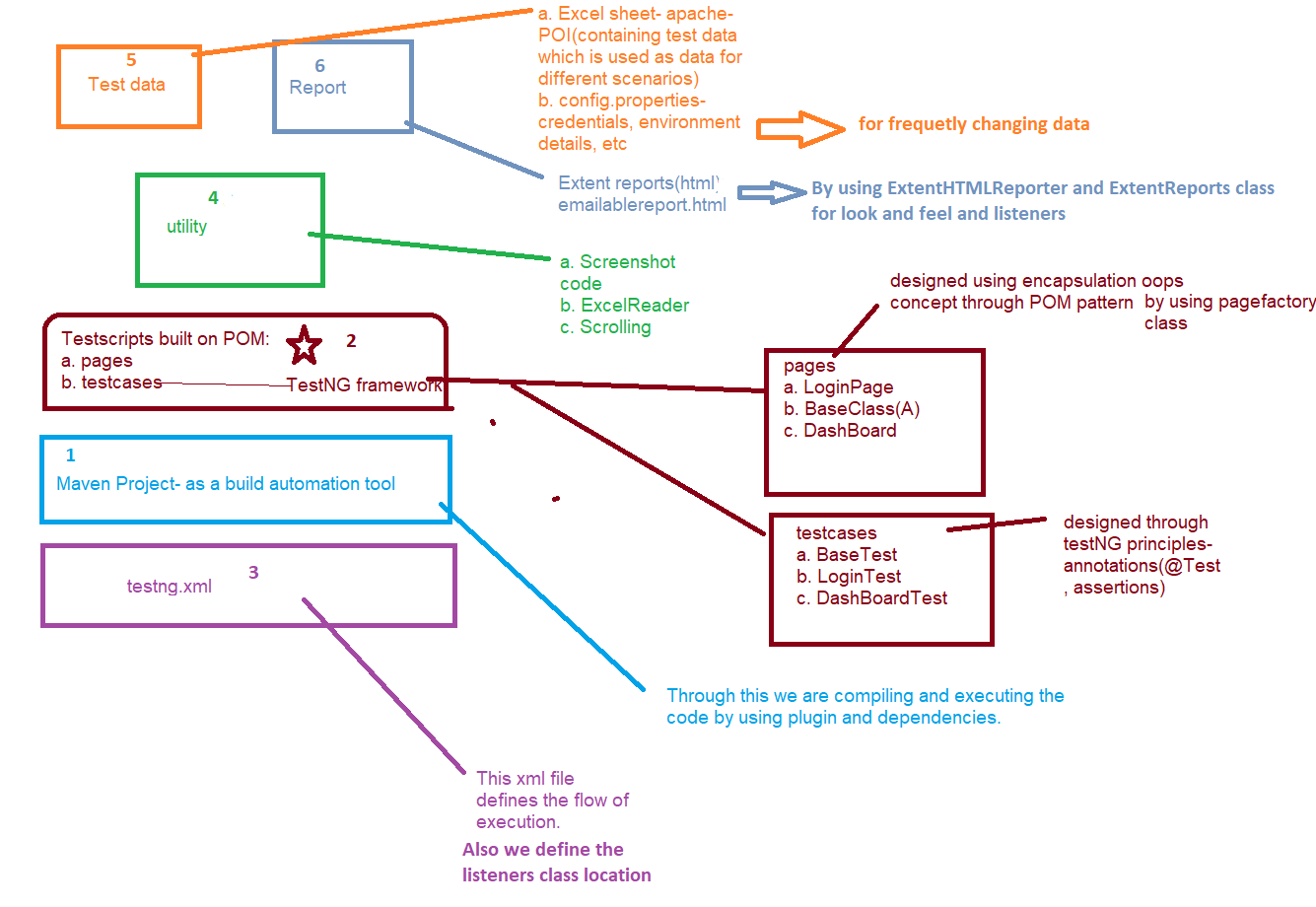
extent.flush();

}

}







Choose of test cases for Automation:

a. Repeatability.

b. Critical test scenarios: Scenarios which highly affects the functionality.

c. End- End behavior.

OOPs concept in Selenium Framework

1. Interface: WebDriver driver = new ChromeDriver();

In this statement WebDriver is nothing but interface in selenium.

TakeScreenShot, JavaScriptExecutor, ITestListener.

1. **UPCASTING**: WebDriver driver = new ChromeDriver();

above statement is nothing but UPCASTING in selenium.

2. **INHERITANCE**

We create a BaseTest in the Framework to initialize WebDriver interface, WebDriver waits, Property files etc.,

We extend the Base Test in Tests Class that is nothing but Inheritance in Selenium Framework.

**POLYMORPHISM**

Combination of overloading and overriding is known as Polymorphism.

Reporter.log, Assert.assertTrue()--- Overloading

Overriding – All the listeners method

3. METHOD OVERLOADING

We use implicit wait in Selenium. Implicit wait is an example of overloading. In Implicit wait we use different time stamps such as SECONDS, MINUTES, HOURS etc.,

A class having multiple methods with same name but different parameters is called Method Overloading.

eg. driver.switchTo().frame(): - String name, int index and WebElement

4**. METHOD OVERRIDING**

Declaring a method in child class which is already present in the parent class is called Method Overriding.

5. **ENCAPSULATION**

All the POM classes in a framework are an example of Encapsulation. In POM classes,

we declare the data members using @FindBy and initialization of data members which are private through this we are achieving datahiding and calling of this private variables into the public method and those public method will get call inside test classes which is Abstraction.

In pagefactory will be done using Constructor to utilize those in methods.

Encapsulation is a mechanism of binding code and data together in a single unit.

Encapsulation is the process of wrapping up code and data together in a single unit. It is used to hide the data of a class from another class.

Encapsulation can be achieved when you declare all variables as private and a public method in a class to get the values of the variable.

6. ABSTRACTION

In Page Object Model design pattern, we write locators (such as id, name, xpath etc.,) in a Page Class.

We utilize these locators in pom class but we can’t see these locators in the tests. Literally we hide the locators from the tests.

Abstraction is the methodology of hiding the implementation of internal details and showing the functionality to the users.

Maven build life cycle

a. Validation: To check whether the names used in the project are correct as per the names in the xml file.

b. Compilation: Code gets compile through maven compilar plugin.

c. package: To prepare a bunch of the content which is to be executed.

d. execution: This is to be done with the help of maven –surefire-plugin.

Maven clean cycle:

a. pre-clean: In this we check whether the current target folder is in use or not.

b. Clean: This will delete the target folder.

c. Post-clean- This will notify the user that build is success.

Commands to execute maven project in command prompt:

mvn test –

mvn install

mvn compile- This is to compile the project

mvn clean – This is to delete the target folder.

Investment Banking(Kite Zeroda) Project Scenarios

1. Quantity field should be by default 1

2. When we click on stock it should show buy ,sell, view chart and other details

26. wishlist--add/remove

3. while selecting order if we select CNC (cash and carry ) order ,we can hold the stock for long term --manual

4. While Selecting order,if we select MIS(margine intraday sqare) it should automatically exit at 3-20 pm.--manual

5. if you try to place order ,but you don't have fund in your account then it should display insufficient fund msg

6. if you forget to logout account,at end of every day it should automatically logged out --manual/automation

7. when we placed the order by selecting market price ,price field should be disabled

8. when we placed limit order price field should be editable

9. when we placed bracket order(BO) then stoploss, target, trailing Stop loss should enable

10. when we placed cover order (co) only Stop loss field enabled

11. price field should accept value in multiple of 0.05 (eg 55.05,55,56.15 etc accepted) (55.06,55.07 not accepted)

12. after placing order ,in orders tab it will show Executed/pending/rejected orders

1. valid 2. invalid

13. in portfolio tab , should show holdings and position of stock. --> mobile

14. after clicking on user id tab it should show customer details

15. in options quantity field should accept in Lott eg ( eg. 25 for banknifty,75 for nifty, depends on stock)

16. while adding fund in your demat account ,after clicking on add fund button , different payment options should enabled

17. when we click on withdraw button , withdrawal amount field and proceed button should enabled.

18. when we click on recent withdrawal option,it should show our withdrawal activities

19. when we click on AMO (after market order) option, we should able to placed order after market timing--manual

20. at time of order placing, BSE Or NSE only one should enabled

21. when we click on chart it should show different formats of chart (eg 1d,5d,1M etc ) d- day,M- month

22. application should show same price of stock as per NSE or bse ,if we open an application in different platforms or machines, mobile etc ---CT

24. if we placed withdrawal request, ammount should be credit next day before 1 pm (if no hollyday) otherwise it should credit next of hollyday--manual

25. while withdrawing funds ,it should show withdrawable ammount.

Modules in Project: 10-15

Profile

Fund

Orders

wishlist

Buy

sell

Position

Holding

Dashboard

Login

Total modules in a project: -- 10--------->10 to 15

webpages in each module:---> 2 ---> 10

Pom / page classes in each module:---> 2 ---> 10

Total pom classes in project:2\*10: 20 -->

1 features= 1 module = TC in each module 50 --> 50 to 200

Test cases in each module: 50 -- 50--150

Test cases in each Test class: 4 3 to 8

Test classes in each module: 12 -- 10 to 40

Test classes in project : 12 \* 10: 120

Test script per day: depends on complexity of Test script (1 to 4) (avg 2)

test script per month (20 working days--> 10 days devp): avg (20 )

src/main/java- All pom classes

src/test/java- All test classes

src/main/java- library files/config files