TestNG

Introduction:-

TestNG is a testing framework created by Cedric Beust.
This is inspired from Junit and Nunit.
In the Test**NG** ng stangs Next Generation.
This is used to cover almost all the categories of testing like unit, end to end,functional,integration etc.

WhyTestNG, why not Junit ?

Before TestNG this was difficult to know about how many test cases have been executed, pass,fail and skip.TestNG gives us this feature to know about how many test cases have been executed, pass,fail and skip.

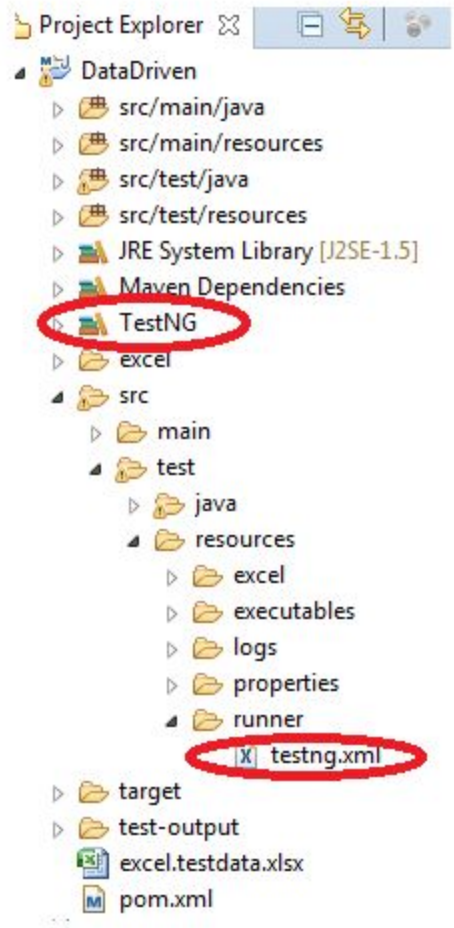

Project Explorer
DataDriven
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [J2SE-1.5]
Maven Dependencies
TestNG
excel
src
main
test
java
resources
excel
executables
logs
properties
runner
testng.xml
target
test-output
excel.testdata.xlsx
pom.xml

fig.1

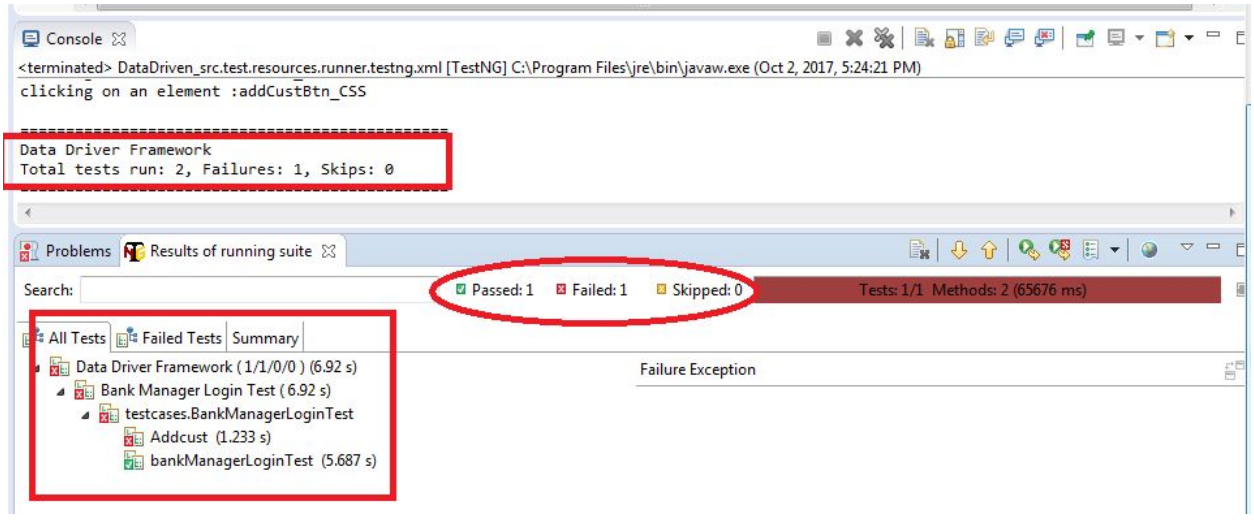

Console
<terminated> DataDriven_src.test.resources.runner.testng.xml [TestNG] C:\Program Files\jre\bin\javaw.exe (Oct 2, 2017, 5:24:21 PM)
clicking on an element :addCustBtn_CSS
Data Driver Framework
Total tests run: 2, Failures: 1, Skips: 0
Problems
Results of running suite
Search:
Passed: 1
Failed: 1
Skipped: 0
Tests: 1/1 Methods: 2 (65676 ms)
All Tests
Failed Tests
Summary
Data Driver Framework (1/1/0/0) (6.92 s)
Bank Manager Login Test (6.92 s)
testcases.BankManagerLoginTest
Addcust (1.233 s)
bankManagerLoginTest (5.687 s)
Failure Exception

Fig.2

TestNG is almost same as Junit but it overcomes the limitations of Junit and provides more functionality than junit for example junit is only used for unit testing while TestNG covers almost all kinds of testing categories.Most of the selenium user prefer TestNG over junit because of its features.This is using widely in the market.

Advantages of TestNG:

1. The prior advantage of TestNG is to provide predefined annotations.
2. It provides a feature to generate the well defined HTML reports.
3. With the help of TestNG we can group our test cases,for example if i have to execute only regression suite so i can group all the regression test cases and with the help of @Test(groups="Regression") statement and will define in the testng.xml inside the <groups></groups> tag.

   E.g :
   ```
   <groups>
       <run>
           <include name="Regression" />
       </run>
       </groups
   ```

4. We can do parallel testing.
5. TestNG uses more Java and OO features.
6. Support for multithreaded testing.

Types of Annotations :

@Test: The annotated method is a part of a test case.

@BeforeSuite: The annotated method will be run before all tests in this suite have run.

@AfterSuite: The annotated method will be run after all tests in this suite have run.

@BeforeTest: The annotated method will be run before any test method belonging to the classes inside the tag is run.

@AfterTest: The annotated method will be run after all the test methods belonging to the classes inside the tag have run.

@BeforeGroups: The list of groups that this configuration method will run before. This method is guaranteed to run 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 run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.

@BeforeClass: The annotated method will be run before the first test method in the current class is invoked.

@AfterClass: The annotated method will be run after all the test methods in the current class have been run.

@BeforeMethod: The annotated method will be run before each test method.

@AfterMethod: The annotated method will be run after each test method.

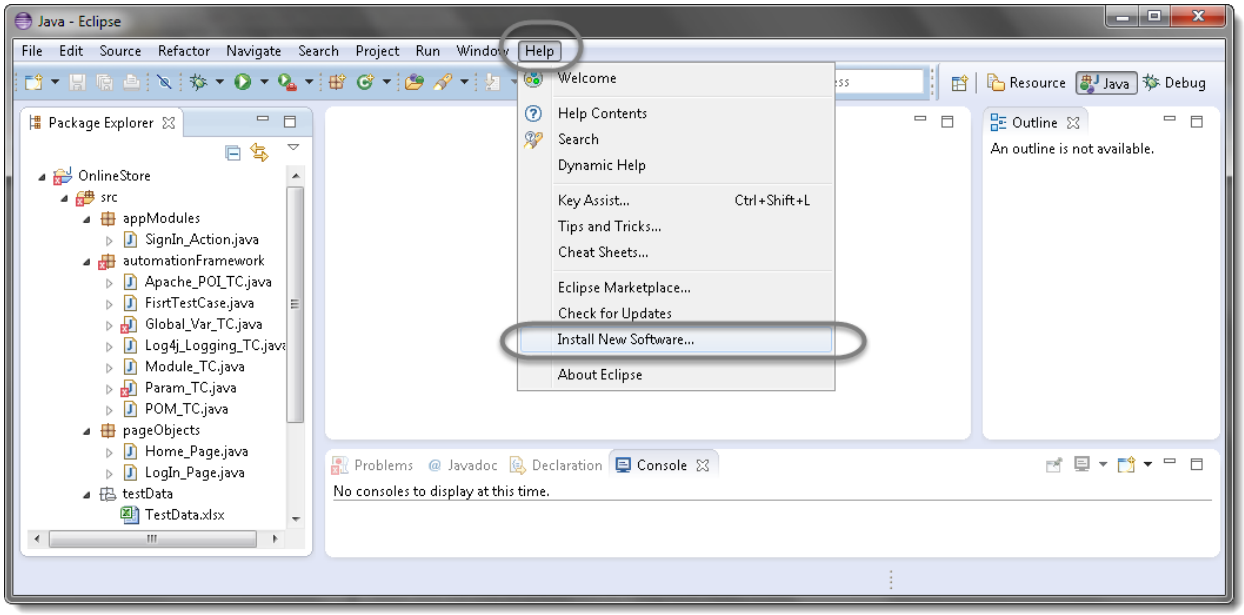
Example of writing test cases with various annotations :-

```java
package testnglearning;

import static org.testng.Assert.fail;

public class TestCase1 {


    @BeforeTest
    public void createDBConn(){
        System.out.println("Create DB Conn:");
    }
    @AfterTest
    public void closeDBConn(){
        System.out.println("Close DB Conn:");
    }

    @BeforeMethod
    public void lauchBrowser(){
        System.out.println("Launching Browser");
    }
    @AfterMethod
    public void closeBrowser(){
        System.out.println("Closing Browser");
    }

    @Test(priority=2,groups="p1")
    public void doLogin() {
        System.out.println("Executing Login Test");
        Assert.fail();
    }

    @Test(priority=1,groups="p1")
    public void doUserReg() {
        System.out.println("Executing User Reg Test");
    }
}
```

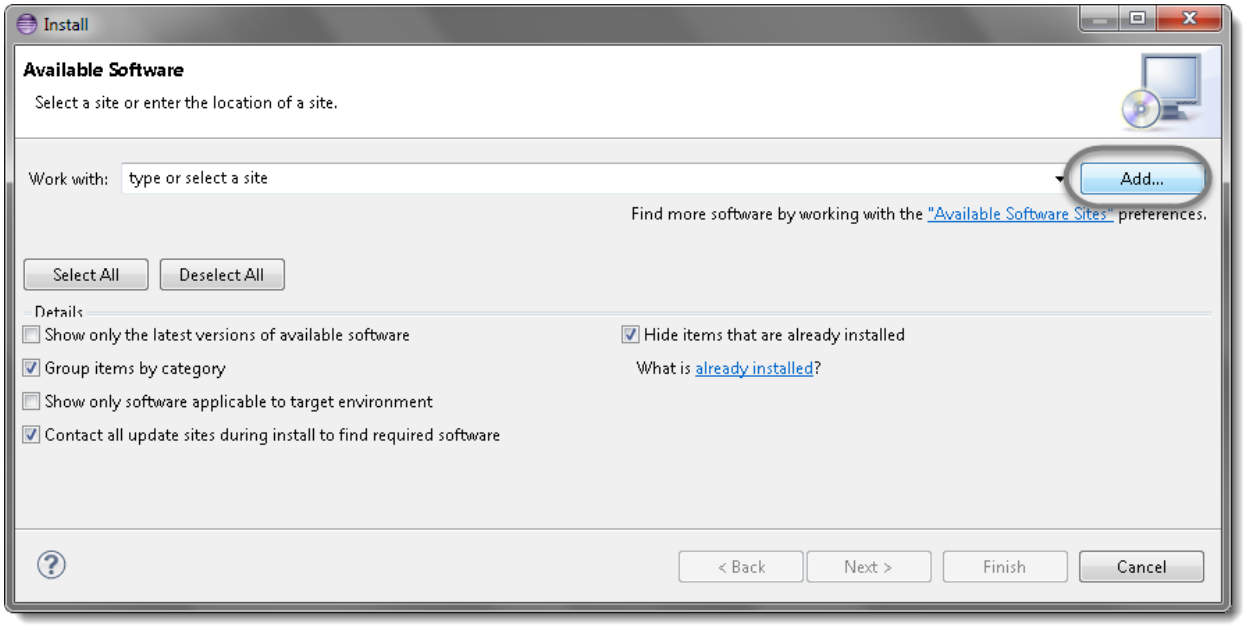
Installing the TestNG :

We simply can copy the dependency from the below url and paste it into the pom.xml file.
http://mvnrepository.com/artifact/org.testng/testng/6.11
Or Download the plug, please follow the below steps .

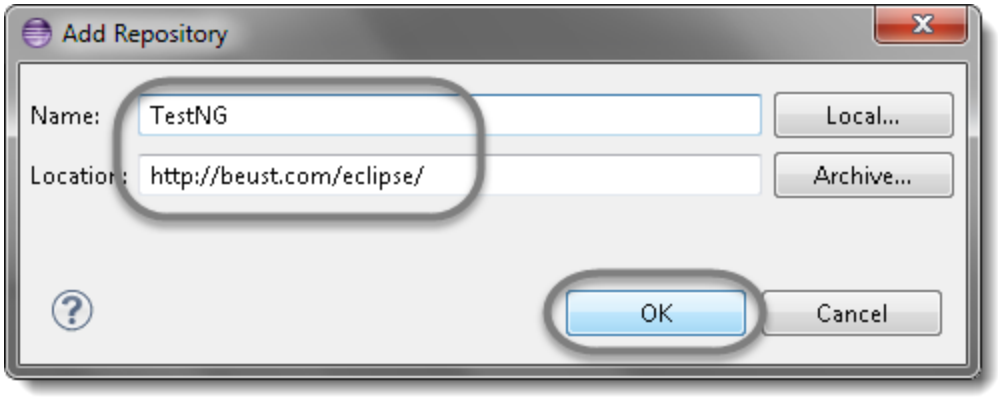
1) Launch the Eclipse IDE and from Help menu, click “Install New Software”.



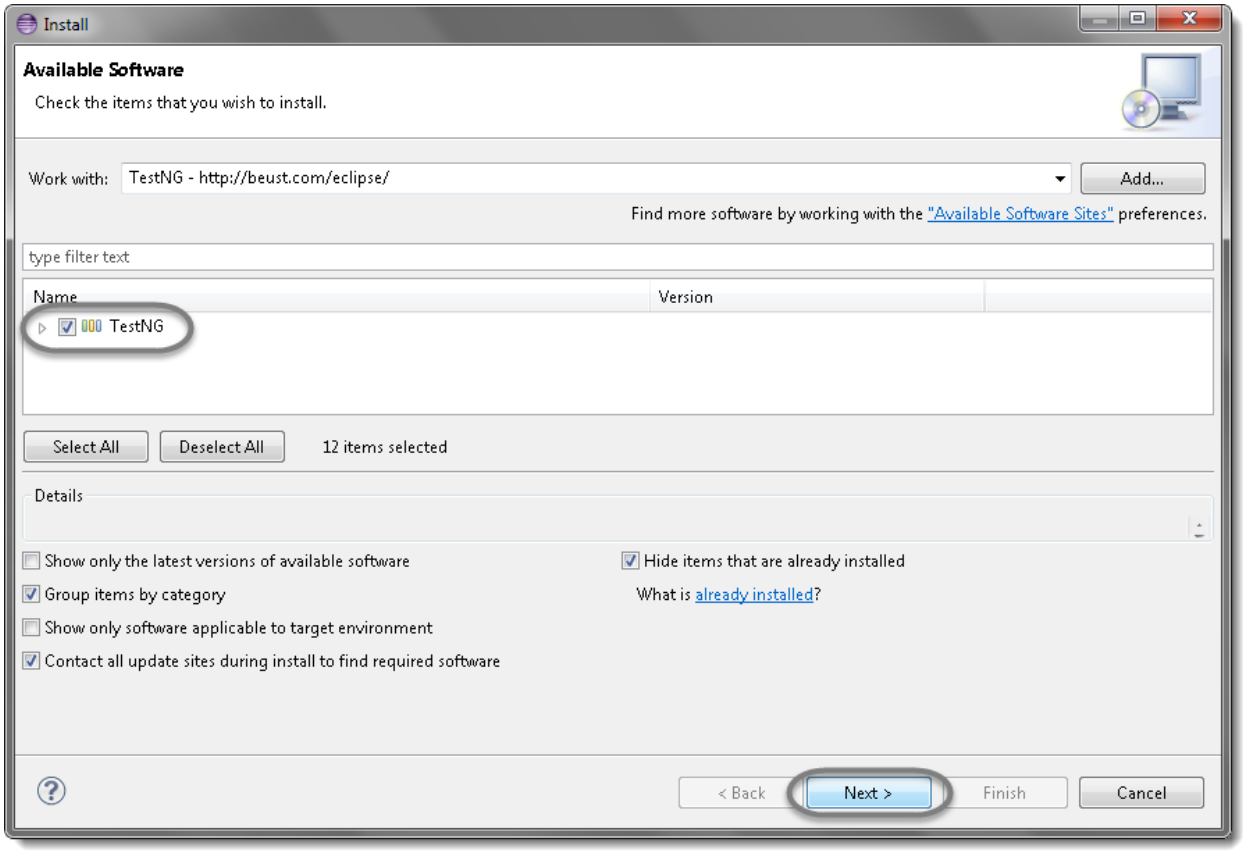
2) You will see a dialog window, click “Add” button.



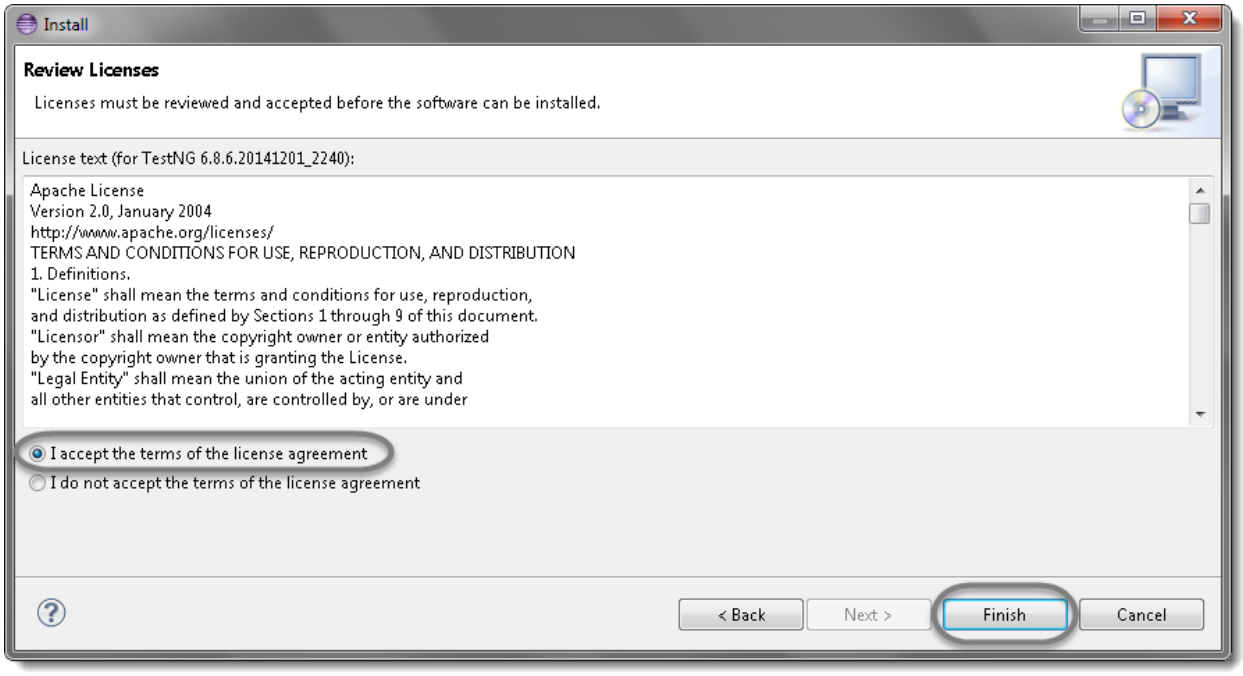
3) Type name as you wish, lets take “TestNG” and type “http://beust.com/eclipse/” as location. Click OK.



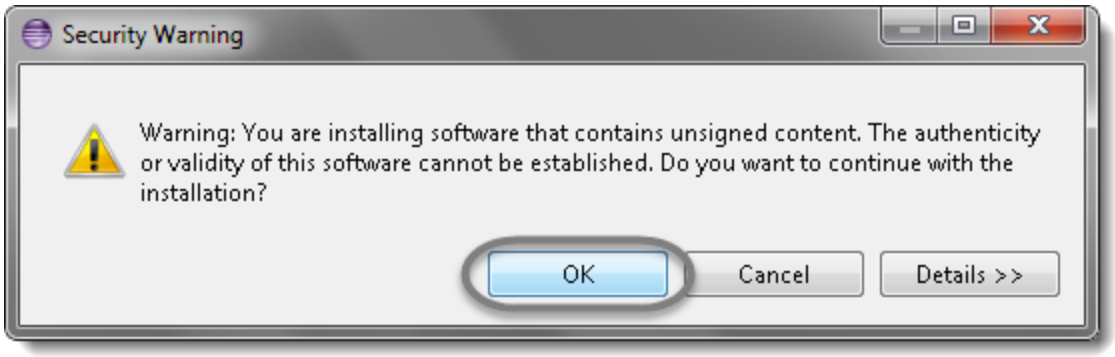
4) You come back to the previous window but this time you must see TestNG option in the available software list. Just Click TestNG and press “Next” button.



5) Click “I accept the terms of the license agreement” then click Finish.

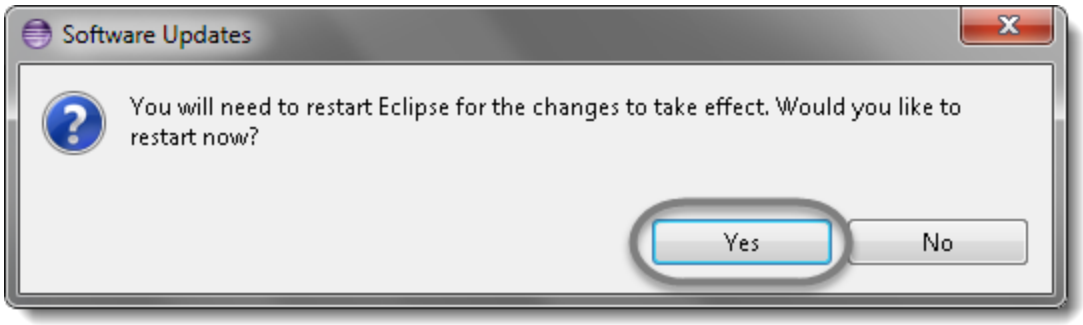


6) You may or may not encounter a Security warning, if in case you do just click OK.



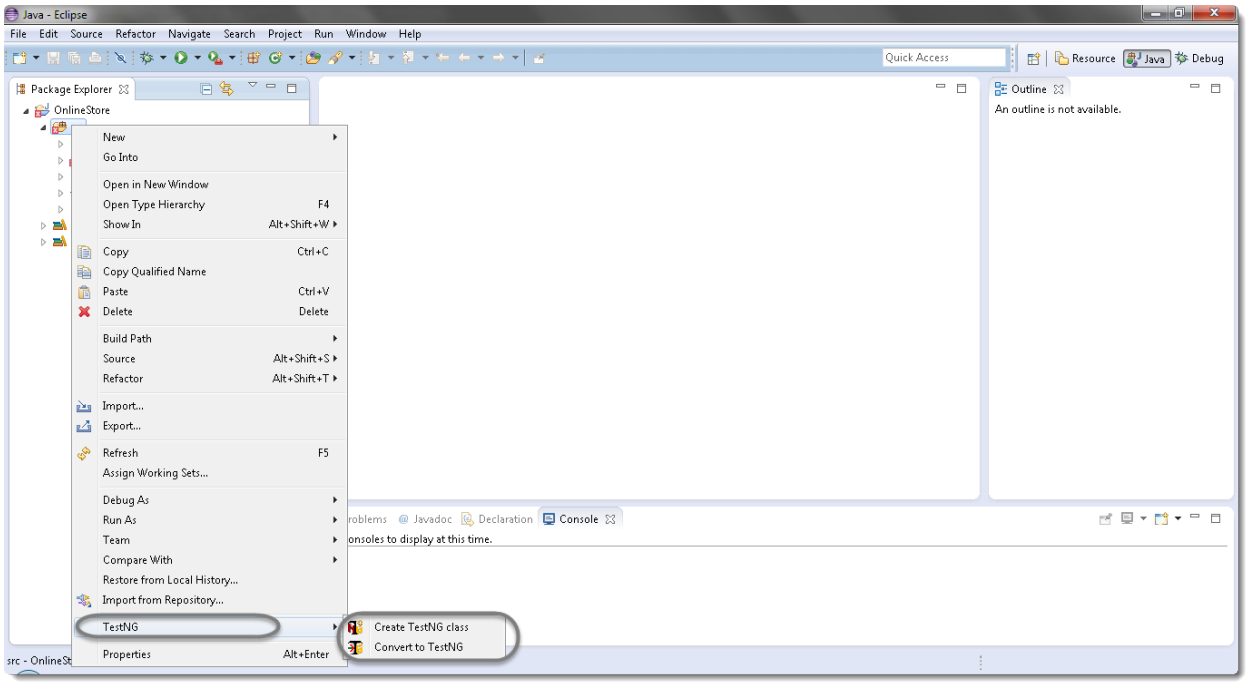
7) Click Next again on the succeeding dialog box until it prompts you to Restart the Eclipse.

8) You are all done now, just Click Yes.



9) Proceed with your workplace.

10) After restart, verify if TestNG was indeed successfully installed. Right click on you project and see if TestNG is displayed in the opened menu.



What is ReportNG :-

This is just a plug in for the TestNG unit-testing framework. The default report is hard to understand also not in the prescribed manner and can't be shown to the managers. so ReportNG overcomes this limitation of testNG.ReportNG provides a simple, colour-coded view of the test results.

Installation Steps for ReportNG:-

1. *Copy the below dependency and paste it into the pom.xml file and save the pom.xml.*

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

<dependency>

<groupId>org.uncommons</groupId>

<artifactId>reportng</artifactId>

<version>1.1.4</version>

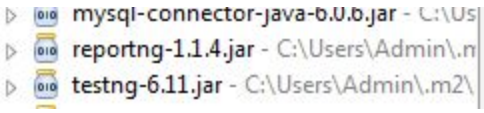
<scope>test</scope>

</dependency>

Or you also can get this dependency from the below link as well.

http://mvnrepository.com/artifact/org.uncommons/reportng/1.1.4

2. After saving the pom.xml you will find the ReportNG jar file as shown in below figure.



To make your reports more interactive,decent and readable you can add below add-ons(dependencies) also.

1.

```
<!-- https://mvnrepository.com/artifact/com.google.inject/guice -->
    <dependency>
        <groupId>com.google.inject</groupId>
        <artifactId>guice</artifactId>
        <version>4.1.0</version>
    </dependency>
```

OR download it from below link.

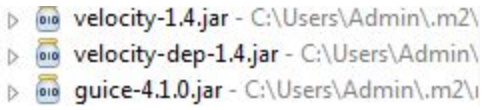
http://mvnrepository.com/artifact/com.google.inject/guice/4.1.0

2.

```
<!-- https://mvnrepository.com/artifact/org.apache.velocity/velocity →
    <dependency>
      <groupId>org.apache.velocity</groupId>
      <artifactId>velocity</artifactId>
      <version>1.7</version>
    </dependency>
```

OR download it from below link.

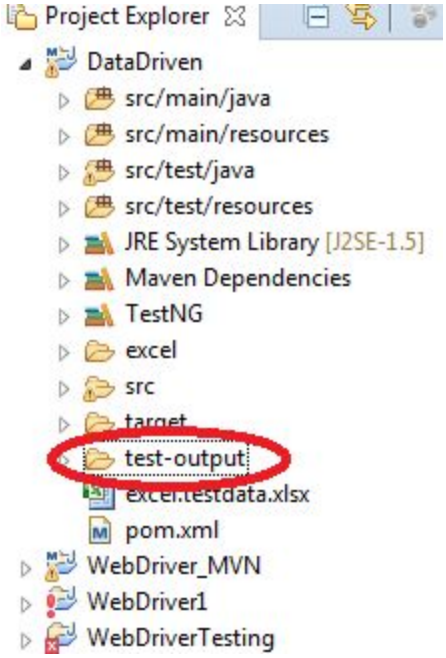
http://mvnrepository.com/artifact/org.apache.velocity/velocity/1.7



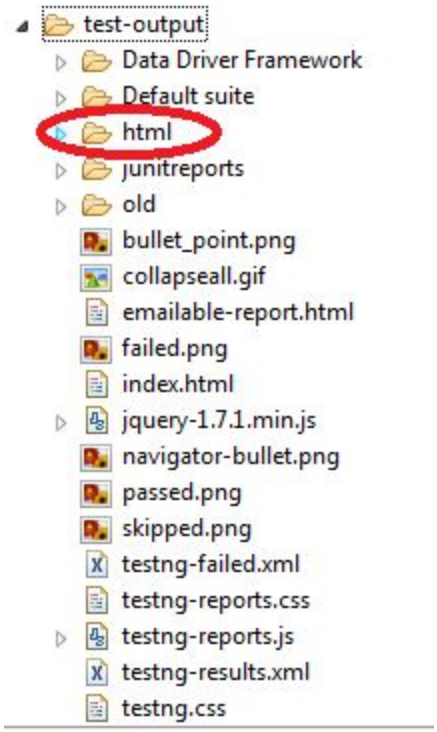
Note:-The current version is 1.1.4, which has been tested with TestNG 6.2 (it should work with any version of TestNG after 5.0, but this has not been tested).

How to generate Test Report :-

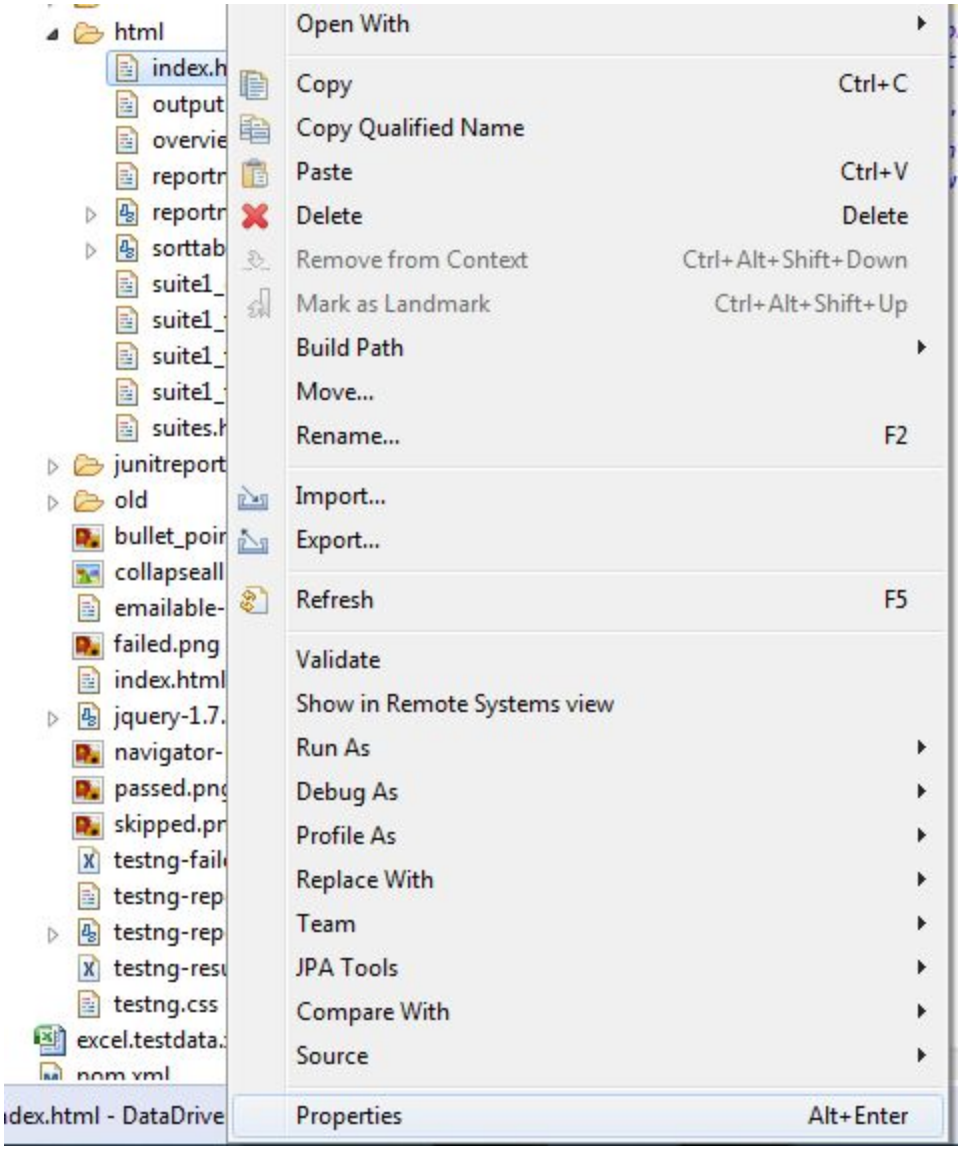
1.Explore the test-output folder .



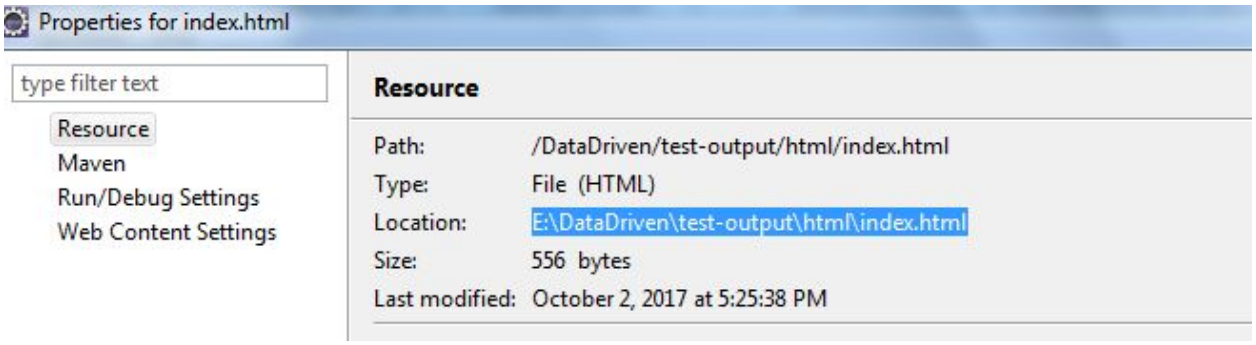
2. Inside the test-output folder you will find html folder .



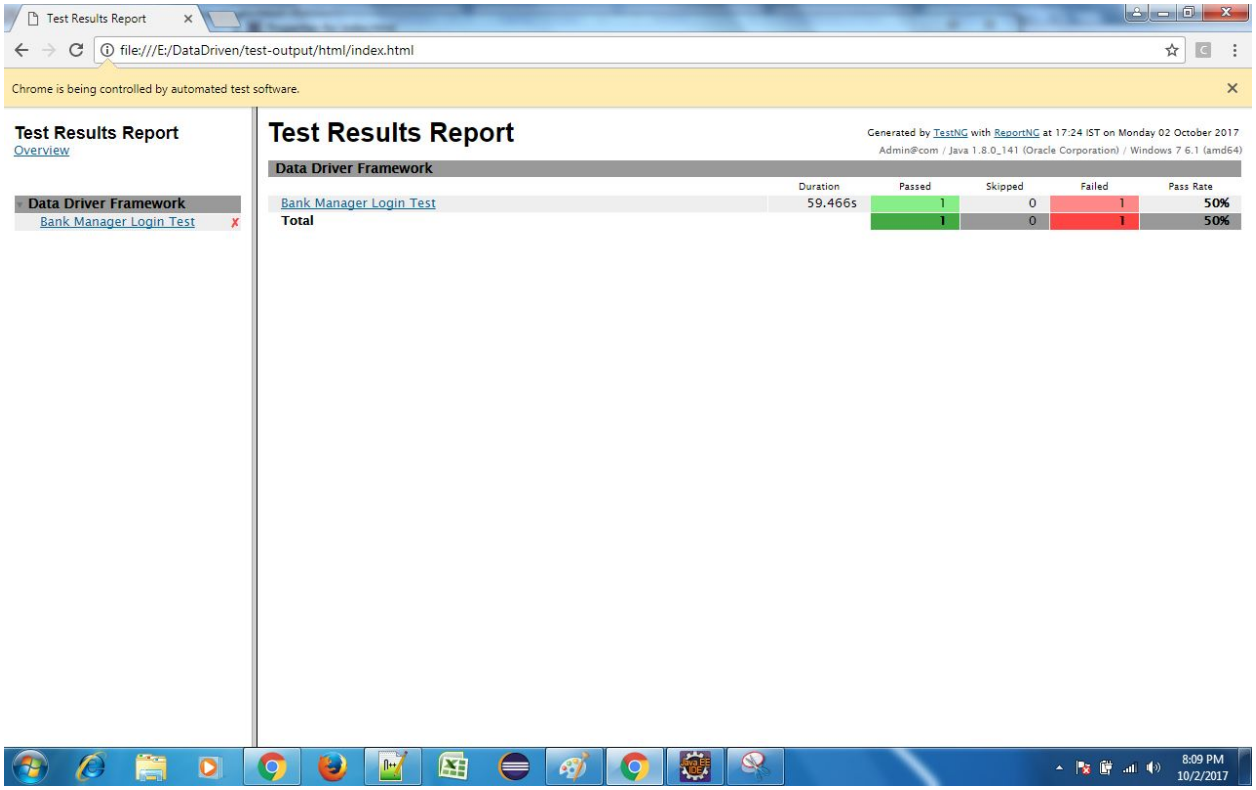
3. Now inside the html folder you have to right click on index.html and select properties.



4. Now copy the given location.



5. And paste it into browser’s url bar and press enter.



6. You will find the report.

You can make this report more understable with the help of surefire-plugin.Please copy the dependency from the below path and paste it into the pom.xml file.

http://mvnrepository.com/artifact/org.apache.maven.plugins/maven-surefire-plugin/2.20.1

What is Asserts in TestNG ?

Assert basically a class of testNG and it works like if else statement but we do cannot use if else in testNG on the place of if else we use the term assertion.

For example:- we have a method to validate the titles.

```
@Test
public void validateTitles(){
        String expected = "Gmail.com" ;
        String actual = "google.com" ;
        if(expected.equals(actual))
        {
```

```
            System.out.println("Pass");
        }else
        {
        System.out.println("fail");
        }
```

Above method will return “fail” but the whole statement has been executed so, in the assertion we use the statement for the above example.

Assert.assertEquals(actual, expected); // this will compare both the values and will return fail

1.Assert.assertTrue()

```
@Test
public void bankManagerLoginTest(){
driver.findElement(By.xpath(OR.getProperty("bmlBtn_CSS"))).sendKeys(value);
Assert.assertTrue(IsElementPresent("addCustBtn_CSS"));
click("addCustBtn_CSS");
}
```

1.Assert.assertFalse()

```
@Test
public void doUserReg() {
System.out.println("Executing User Reg Test");
Assert.fail("User not Registered Successfully");
}
```

Assert true statement fails the test and stop the execution of the test, if the actual output is false. Assert.assertFalse() works opposite of Assert.assertTrue(). It means that if you want your test to continue only if when some certain element is not present on the page. You will use Assert false, so it will fail the test in case of the element present on the page.

3. Assert.assertEquals()

```
@Test
public void test()
{
        String Name = "Rahul Arora";
        System.out.println(" What is your full name");
        Assert.assertEquals("Rahul Arora", Name);
        System.out.println(Name);
 }
```

This will compare the value "Rahul Arora" from the variable Name and will return pass.