TestNG – It is a testing framework which is an abbreviation for Testing New Generation.

It is basically allow the user to control the flow of execution.

Features of TestNG:

1. Execution flow can be decided from user.

2. Pre and post action with respect to test case can be defined.

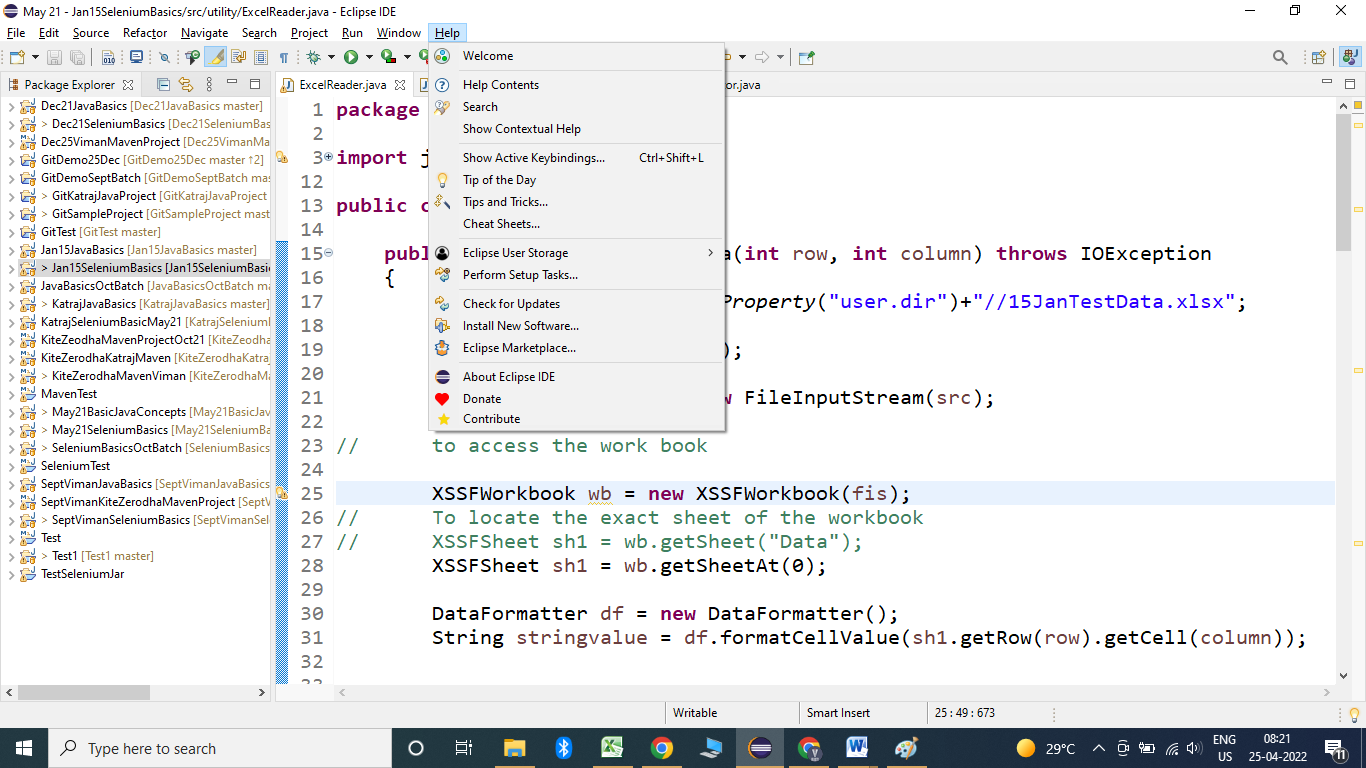
3. Parallel testing can be performed.

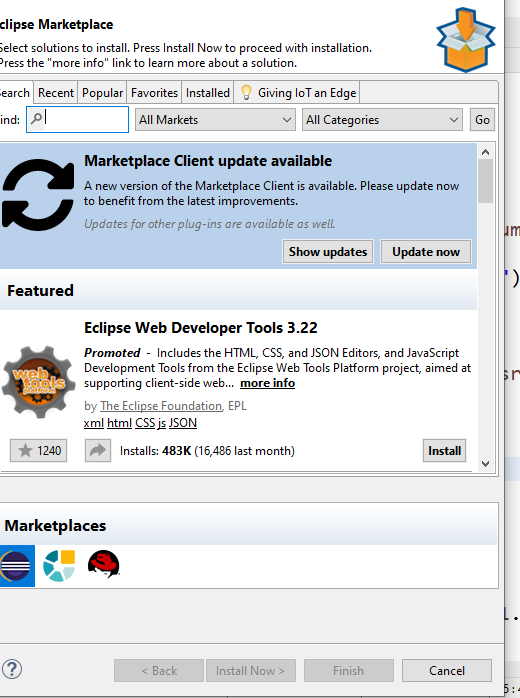
4. Generate the emailable - report based on the execution status.

5. Test case pass / fail / skipped status can be decided using assertions in TestNG.

Installation process of TestNG:

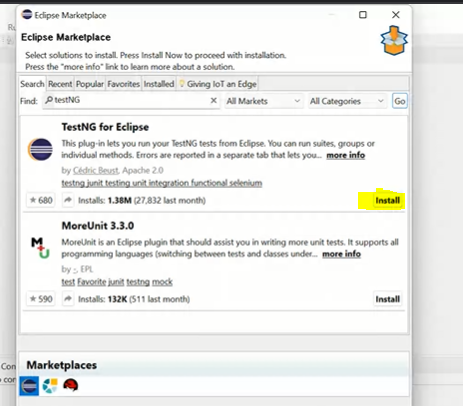
1. Go to help and then –click on Eclipse market place

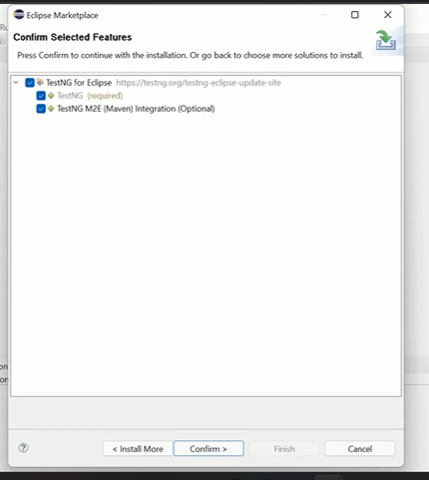




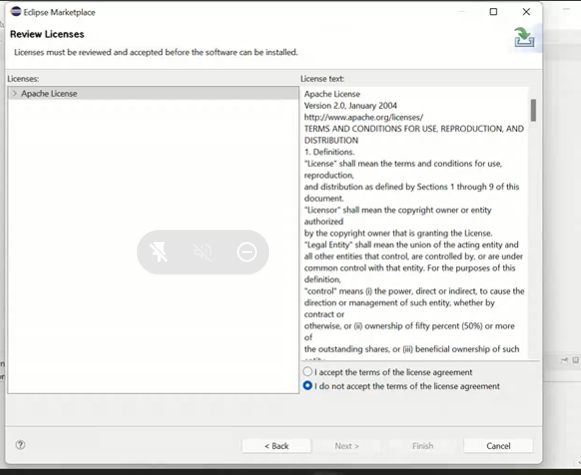
Type TestNG in the text box and click on go.

Click on install:





Click on confirm button



Click on I accept and finish

Execute process of test case:

**public** **class** SingleClassExecution {

@Test

**public** **void** testCase1()

{

System.***out***.println("Test case 1 executed");

}

@Test

**public** **void** testCase2()

{

System.***out***.println("Test case 2 executed");

}

}

To execute Rightclick inside the class and select Run As --- TestNG Test

Output:

Test case 1 executed

Test case 2 executed

PASSED: testCase1

PASSED: testCase2

===============================================

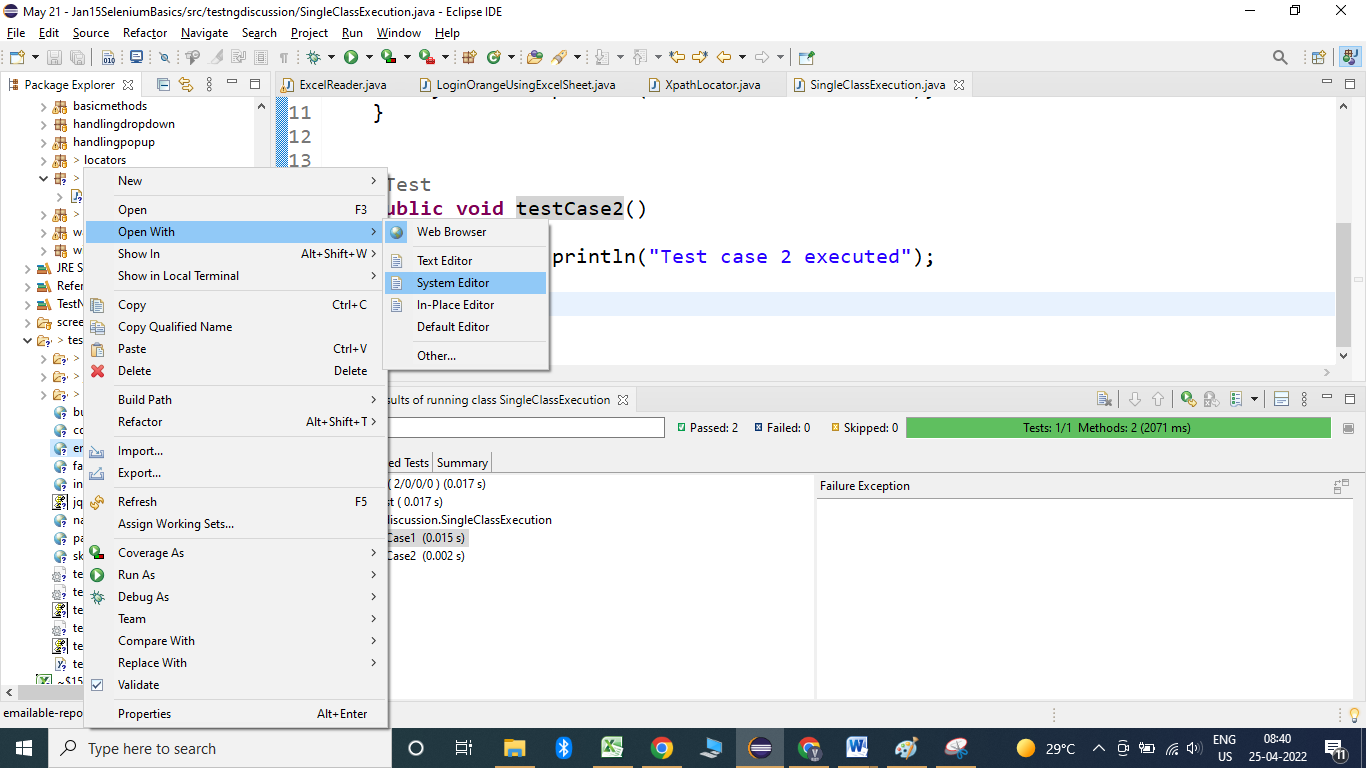
Default test

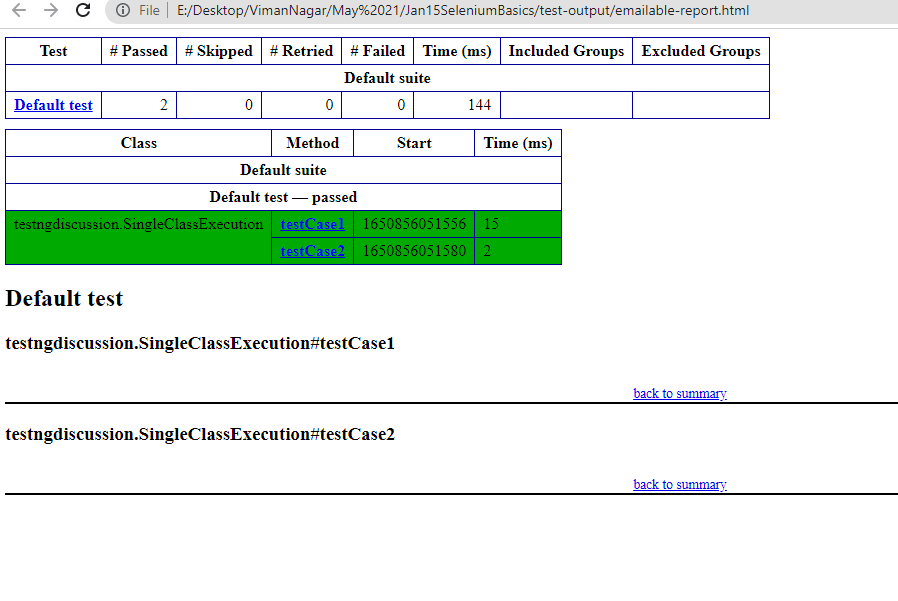
Tests run: 2, Failures: 0, Skips: 0

===============================================

To see the report of the execution:

Refresh the project – go to test output folder – emailable report





Test cases always get execute in the dictionary order.

Example:

**public** **class** SingleClassExecution2 {

@Test

**public** **void** login()

{

System.***out***.println("Login to application");

}

@Test

**public** **void** dashBoard()

{

System.***out***.println("DashBoard Test case");

}

@Test

**public** **void** profile()

{

System.***out***.println("Profile Test case");

}

@Test

**public** **void** home()

{

System.***out***.println("home Test case");

}

@Test

**public** **void** logout()

{

System.***out***.println("logout Test case");

}

}

Output:

DashBoard Test case

home Test case

Login to application

logout Test case

Profile Test case

Keywords in TestNG;

1. **priority**: We can decide the order of execution based on priority keyword.

Example:

**public** **class** SingleClassExecution2 {

@Test(priority = 1)

**public** **void** login()

{

System.***out***.println("Login to application");

}

@Test(priority = 2)

**public** **void** dashBoard()

{

System.***out***.println("DashBoard Test case");

}

@Test(priority = 4)

**public** **void** profile()

{

System.***out***.println("Profile Test case");

}

@Test(priority = 3)

**public** **void** home()

{

System.***out***.println("home Test case");

}

@Test(priority = 5)

**public** **void** logout()

{

System.***out***.println("logout Test case");

}

Output:

Login to application

DashBoard Test case

home Test case

Profile Test case

logout Test case

The priority of a Test case can be:

i. Priority of a test case can be duplicate in that case duplicate priority test case can be executed in dictionary order.

ii. Priority of a test case cannot be in fractions.

@Test(priority = 3.2)

**public** **void** home()

{

System.***out***.println("home Test case");

}

This is **invalid.**

iii. If we don’t define the priority of a test case then it will be considered as 0.

Example:

@Test(priority = 1)

**public** **void** login()

{

System.***out***.println("Login to application");

}

@Test(priority = 2)

**public** **void** dashBoard()

{

System.***out***.println("DashBoard Test case");

}

@Test(priority = 4)

**public** **void** profile()

{

System.***out***.println("Profile Test case");

}

@Test(priority = 3)

**public** **void** home()

{

System.***out***.println("home Test case");

}

@Test

**public** **void** logout()

{

System.***out***.println("logout Test case");

}

Output:

logout Test case

Login to application

DashBoard Test case

home Test case

Profile Test case

iv. It is not necessary to have priority in sequence.

v. Priority of a test case can be –ve as well.

2. InvocationCount: To execute a particular test case multiple times then we have to use invocation count keyword.

Example:

@Test(invocationCount = 5, priority = -1)

**public** **void** home()

{

System.***out***.println("home Test case");

}

@Test

**public** **void** logout()

{

System.***out***.println("logout Test case");

}

@Test(priority = 56)

**public** **void** dashBoard()

{

System.***out***.println("DashBoard Test case");

}

Output:

home Test case

home Test case

home Test case

home Test case

home Test case

logout Test case

DashBoard Test case

Reporter class :

This class’s log method is used to print the messages inside the testng report.

Example:

@Test

**public** **void** logout()

{

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

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

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

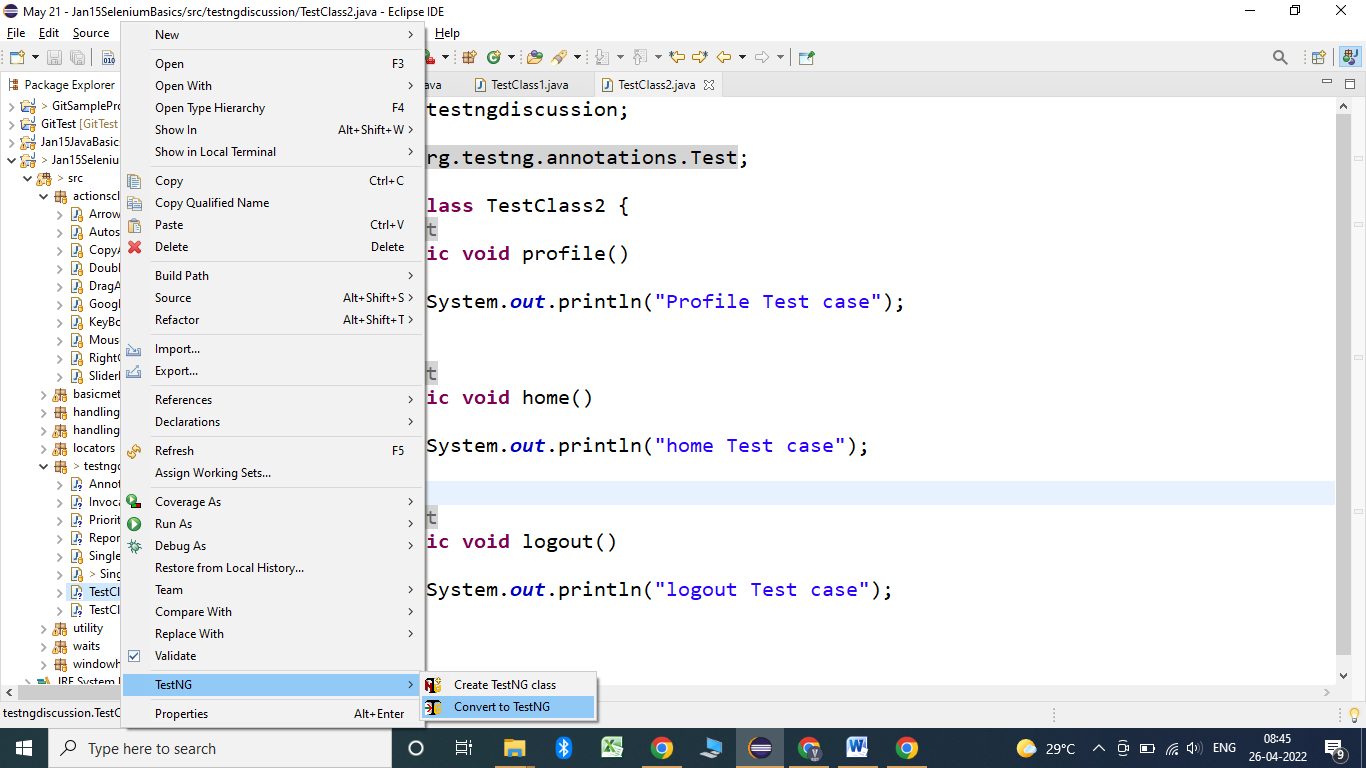
driver.get("https://www.google.com");

Reporter.*log*("This is logout test case", **true**);// true will print the statement not only inside the console but also in Report

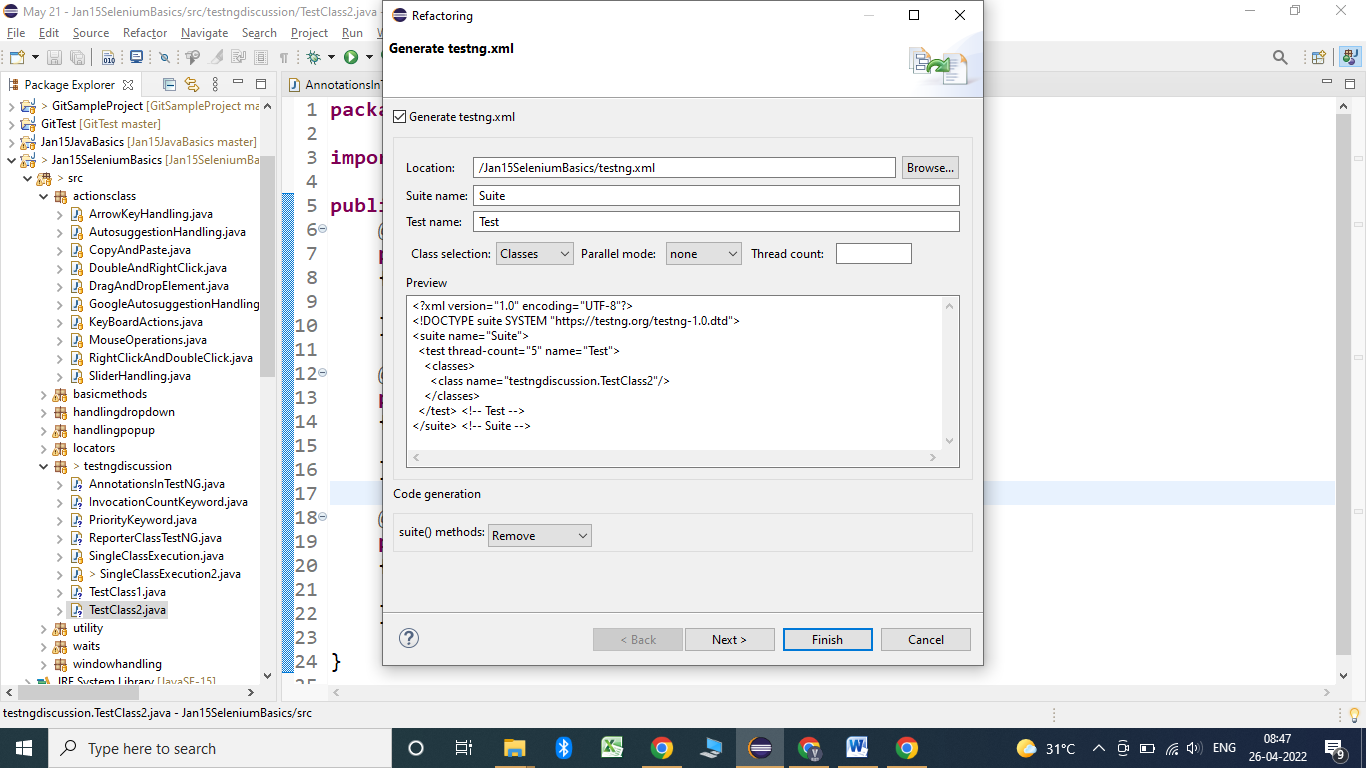
}

Execution of multiple classes by xml file:

1. create the xml file-

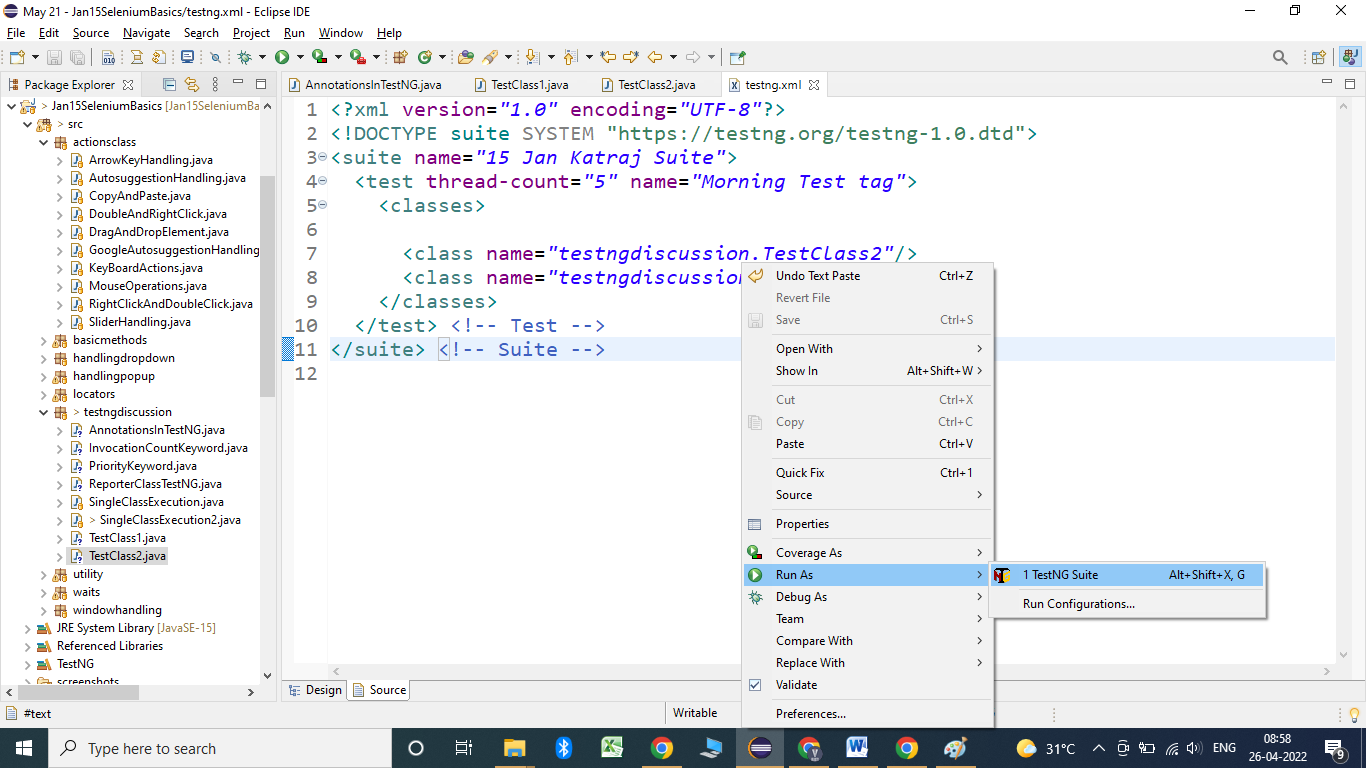


Check the name of file and click on Finish of the below window:



Add the classes which we want to execute in a sequence.

Execute the xml file:



Example:

**public** **class** TestClass1 {

@Test

**public** **void** login()

{

System.***out***.println("Login to application");

}

@Test

**public** **void** dashBoard()

{

System.***out***.println("DashBoard Test case");

}

}

**public** **class** TestClass2 {

@Test

**public** **void** profile()

{

System.***out***.println("Profile Test case");

}

@Test

**public** **void** home()

{

System.***out***.println("home Test case");

}

@Test

**public** **void** logout()

{

System.***out***.println("logout Test case");

}

} <?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"15 Jan Katraj Suite"*>

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

<classes>

<class name=*"testngdiscussion.TestClass2"*/>

<class name=*"testngdiscussion.TestClass1"*/>

</classes>

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

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

Output:

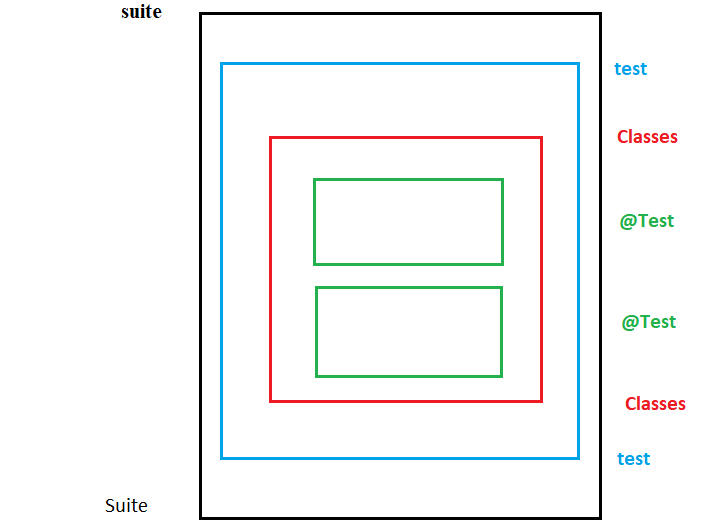
home Test case

logout Test case

Profile Test case

DashBoard Test case

Login to application



Annotations in TestNG;

1. @BeforeMethod: This will execute before the execution of every test case of that class.

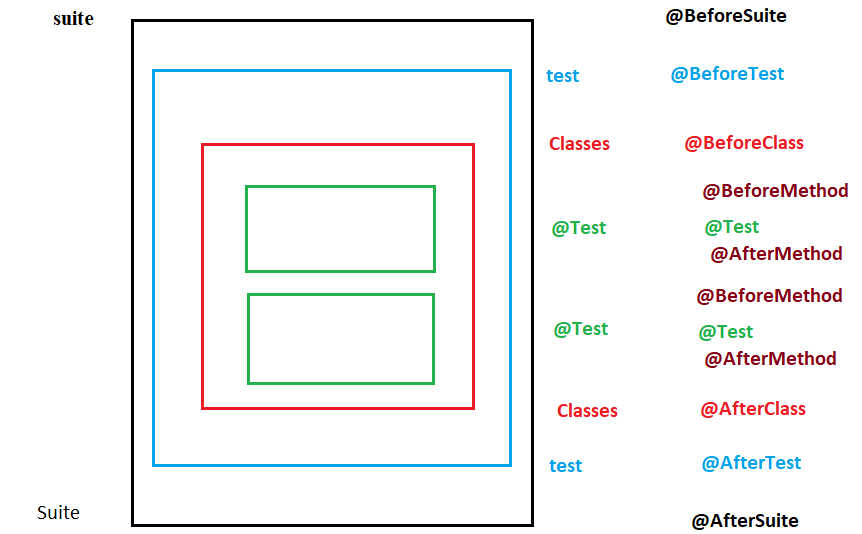
2. @AfterMethod : This will execute after the execution of every test case of that class.

3. @BeforeClass : This will execute before the execution of any thing from a class in which @BeforeClass is defined.

4. @AfterClass : This will execute after the execution of everything from a class in which @AfterClass is defined.

5. @BeforeTest : This will execute before the execution of anything from any class.

6. @AfterTest : This will execute after the execution of everything from all the classes.



Example:

package testngdiscussion;

import org.testng.annotations.AfterClass;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.AfterSuite;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.BeforeSuite;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class AnnotationsInTestNG {

@BeforeSuite

public void beforeSuite()

{

System.out.println("before suite is executing");

}

@AfterTest

public void afterTest()

{

System.out.println("After Test method");

}

@AfterSuite

public void afterSuite()

{

System.out.println("After suite is executing");

}

@BeforeTest

public void beforeTest()

{

System.out.println("Before Test method");

}

@BeforeClass

public void beforeClass()

{

System.out.println("Before class");

}

@AfterClass

public void afterClass()

{

System.out.println("After class");

}

@AfterMethod

public void am()

{

System.out.println("After method");

}

@Test

public void loginTest()

{

System.out.println("Login to application");

}

@Test

public void logoutTest()

{

System.out.println("Logout to application");

}

@BeforeMethod

public void bm()

{

System.out.println("Before method");

}

}

Output:

before suite is executing

Before Test method

Before class

Before method

Login to application

After method

Before method

Logout to application

After method

After class

After Test method

After suite is executing

Actual usage of Annotations:

**public** **class** ActualUsageExampleOfAnnotations {

@BeforeSuite

**public** **void** openBrowser()

{

System.***out***.println("Opening the browser");

}

@AfterTest

**public** **void** navBackToOriginalURL()

{

System.***out***.println("Navigate back to original URL");

}

@AfterSuite

**public** **void** quitBrowser()

{

System.***out***.println("Closing the browser");

}

@BeforeTest

**public** **void** navToURL()

{

System.***out***.println("Navigate to URL");

}

@BeforeClass

**public** **void** loginToApp()

{

System.***out***.println("Login to application");

}

@AfterClass

**public** **void** logoutFromApp()

{

System.***out***.println("Logout from application");

}

@AfterMethod

**public** **void** navBackToHome()

{

System.***out***.println("Navigating back to home page");

}

@Test

**public** **void** testCase1()

{

System.***out***.println("Test case 1");

}

@Test

**public** **void** testCase2()

{

System.***out***.println("Test case 2");

}

@BeforeMethod

**public** **void** navToTestCaseURL()

{

System.***out***.println("Navigating to the test url");

}

}

Output:

Opening the browser

Navigate to URL

Login to application

Navigating to the test url

Test case 1

Navigating back to home page

Navigating to the test url

Test case 2

Navigating back to home page

Logout from application

Navigate back to original URL

Closing the browser

DependsOnMethods keyword:

Through this keyword we can depend the execution of a testcase based on the pass / fail of a particular test case.

If the testcase gets pass then the dependent testcase will execute other wise it will get skipped.

Example:

@Test(priority = 1)

**public** **void** loginTest()

{

System.***out***.println("Login to application");

Assert.*fail*("test case failing deliberately");

}

@Test(dependsOnMethods = "loginTest", priority = 2)

**public** **void** homePage()

{

System.***out***.println("Home page of application");

}

@Test(dependsOnMethods = {"homePage", "loginTest"}, priority = 3)

**public** **void** logout()

{

System.***out***.println("Logout from application");

}

Output:

Default test

Tests run: 3, Failures: 1, Skips: 2

===============================================

===============================================

Default suite

Total tests run: 3, Passes: 0, Failures: 1, Skips: 2

===============================================

Example 2: DependsOnMethods with multiple classes:

**public** **class** DependsOnMethodsKeyword {

@Test(priority = 1)

**public** **void** loginTest()

{

System.***out***.println("Login to application");

Assert.*fail*("test case failing deliberately");

}

@Test(dependsOnMethods = "loginTest", priority = 2)

**public** **void** homePage()

{

System.***out***.println("Home page of application");

}

@Test(dependsOnMethods = {"homePage", "loginTest"}, priority = 3)

**public** **void** logout()

{

System.***out***.println("Logout from application");

}

}

**public** **class** DependsOnMethods2 {

@Test(dependsOnMethods = "testngdiscussion.DependsOnMethodsKeyword.homePage", priority = 4)

**public** **void** profile()

{

System.***out***.println("Profile Test case");

}

@Test(priority = 5)

**public** **void** dashBoard()

{

System.***out***.println("dashboard Test case");

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

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

<classes>

<class name=*"testngdiscussion.DependsOnMethodsKeyword"*/>

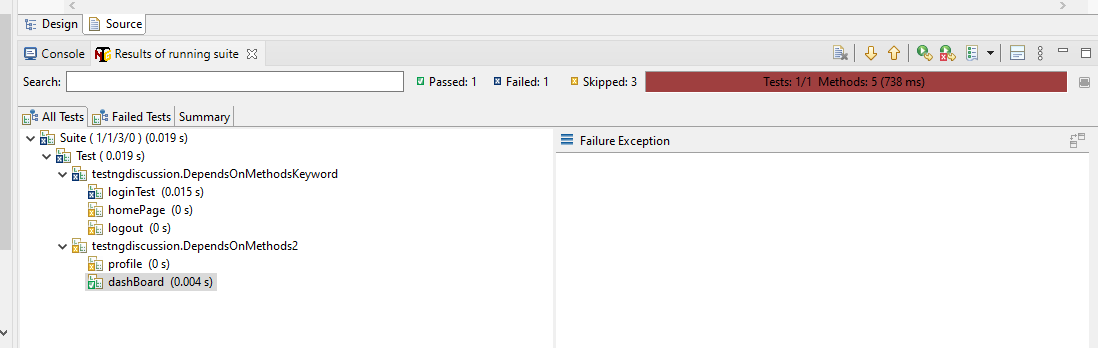
<class name=*"testngdiscussion.DependsOnMethods2"*/>

</classes>

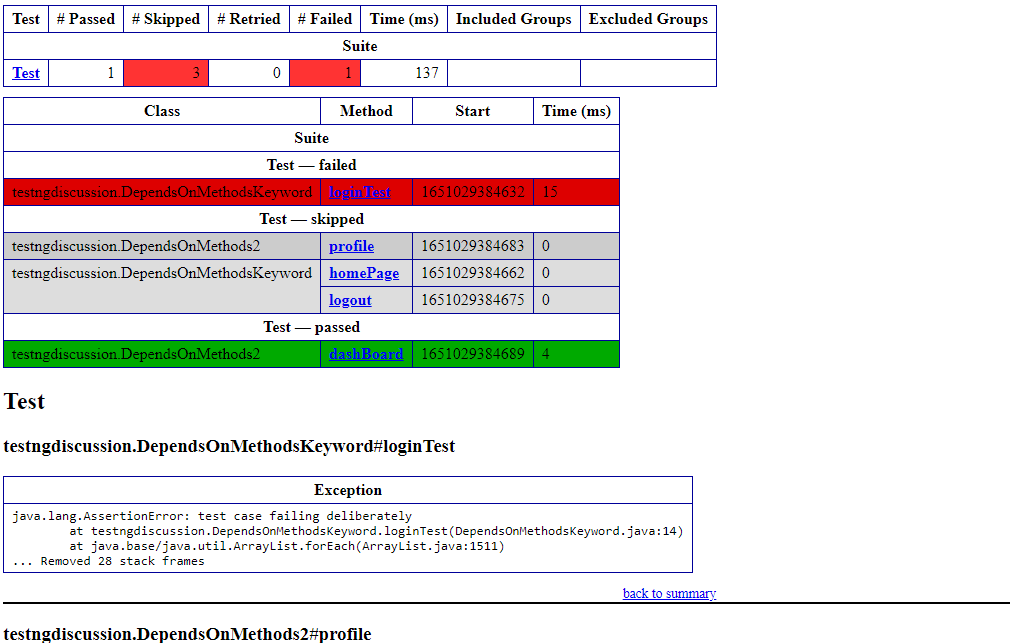
</test> <!-- Test -->

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

Output:



Report:



Inclusion and Exclusion of test case – 1

Enabled keyword: We can use enabled keyword to make a particular test case disable if we write enabled = false then that test case will not consider for execution.

@Test

**public** **void** profile()

{

System.***out***.println("Profile Test case");

}

@Test(enabled = **false**)

**public** **void** login()

{

System.***out***.println("Login Test case");

}

@Test

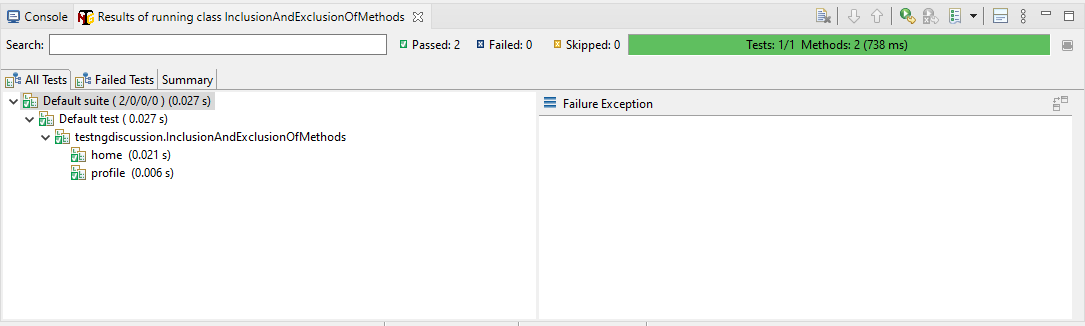
**public** **void** home()

{

System.***out***.println("Home test case");

}

Output:



Include and Exclude a test case from the execution:

To include and exclude the test case from the execution we can mention those inside the xml file with the tagname as - <include> or <exclude>

Example:

**package** testngdiscussion;

**import** org.testng.annotations.Test;

**public** **class** InclusionAndExclusionOfMethods {

@Test

**public** **void** profile()

{

System.***out***.println("Profile Test case");

}

@Test

**public** **void** login()

{

System.***out***.println("Login Test case");

}

@Test

**public** **void** home()

{

System.***out***.println("Home test case");

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

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

<classes>

<class name=*"testngdiscussion.InclusionAndExclusionOfMethods"*>

<methods>

<exclude name=*"profile"*></exclude>

</methods>

</class>

</classes>

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

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

Output:

Home test case

Login Test case

Grouping of Testcase:

We can categorize the testcases based on their groups and on the basis of inclusion / exclusion we can execute a particular group of the test case.

Example:

**package** testngdiscussion;

**import** org.testng.annotations.Test;

**public** **class** GroupingOfTestCase1 {

@Test(groups = "Sanity")

**public** **void** login()

{

System.***out***.println("Login to application");

}

@Test(groups = "Sanity")

**public** **void** dashBoard()

{

System.***out***.println("DashBoard Test case");

}

@Test(groups = "Regression")

**public** **void** profile()

{

System.***out***.println("Profile Test case");

}

@Test(groups = "functional")

**public** **void** home()

{

System.***out***.println("home Test case");

}

@Test(groups = "Sanity")

**public** **void** logout()

{

System.***out***.println("logout Test case");

}

}

**public** **class** GroupingOfTestCase2 {

@Test(groups = "functional")

**public** **void** admin()

{

System.***out***.println("admin application");

}

@Test(groups = "Sanity")

**public** **void** myInfo()

{

System.***out***.println("My info Test case");

}

@Test(groups = "Regression")

**public** **void** buyShares()

{

System.***out***.println("buy share Test case");

}

@Test(groups = "Regression")

**public** **void** sellShares()

{

System.***out***.println("sell shares Test case");

}

@Test(groups = "Sanity")

**public** **void** searchShares()

{

System.***out***.println("search shares Test case");

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

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

<groups>

<run>

<include name=*"Sanity"*></include>

</run>

</groups>

<classes>

<class name=*"testngdiscussion.GroupingOfTestCase1"* />

<class name=*"testngdiscussion.GroupingOfTestCase2"* />

</classes>

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

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

Output from the xml execution:

DashBoard Test case

Login to application

logout Test case

My info Test case

search shares Test case

Testcase with multiple groups:

**public** **class** GroupingOfTestCase1 {

@Test(groups = {"Sanity" , "Regression"})

**public** **void** login()

{

System.***out***.println("Login to application");

}

@Test(groups = "Sanity")

**public** **void** dashBoard()

{

System.***out***.println("DashBoard Test case");

}

@Test(groups = {"Regression", "functional"})

**public** **void** profile()

{

System.***out***.println("Profile Test case");

}

@Test(groups = {"functional", "Sanity", "Regression"})

**public** **void** home()

{

System.***out***.println("home Test case");

}

@Test(groups = "Sanity")

**public** **void** logout()

{

System.***out***.println("logout Test case");

}

}

**package** testngdiscussion;

**import** org.testng.annotations.Test;

**public** **class** GroupingOfTestCase2 {

@Test(groups = "functional")

**public** **void** admin()

{

System.***out***.println("admin application");

}

@Test(groups = "Sanity")

**public** **void** myInfo()

{

System.***out***.println("My info Test case");

}

@Test(groups = "Regression")

**public** **void** buyShares()

{

System.***out***.println("buy share Test case");

}

@Test(groups = "Regression")

**public** **void** sellShares()

{

System.***out***.println("sell shares Test case");

}

@Test(groups = "Sanity")

**public** **void** searchShares()

{

System.***out***.println("search shares Test case");

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

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

<groups>

<run>

<!-- <include name="Sanity"></include> -->

<exclude name=*"Regression"*></exclude>

</run>

</groups>

<classes>

<class name=*"testngdiscussion.GroupingOfTestCase1"* />

<class name=*"testngdiscussion.GroupingOfTestCase2"* />

</classes>

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

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

Output:

DashBoard Test case

logout Test case

admin application

My info Test case

search shares Test case

Note: If we write <**include**> inside the xml then it ONLY include that particular group of test case, but if we write <**exclude**> then it leave that particular group and execute all the other groups along with annotations.

Example:

**public** **class** GroupingAnnotations {

@BeforeSuite

**public** **void** beforeSuite()

{

System.***out***.println("before suite is executing");

}

@AfterTest

**public** **void** afterTest()

{

System.***out***.println("After Test method");

}

@AfterSuite

**public** **void** afterSuite()

{

System.***out***.println("After suite is executing");

}

@BeforeTest

**public** **void** beforeTest()

{

System.***out***.println("Before Test method");

}

@BeforeClass

**public** **void** beforeClass()

{

System.***out***.println("Before class");

}

@AfterClass

**public** **void** afterClass()

{

System.***out***.println("After class");

}

@AfterMethod

**public** **void** am()

{

System.***out***.println("After method");

}

@Test(groups = "Sanity")

**public** **void** loginTest()

{

System.***out***.println("Login to application");

}

@Test

**public** **void** logoutTest()

{

System.***out***.println("Logout to application");

}

@BeforeMethod

**public** **void** bm()

{

System.***out***.println("Before method");

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

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

<groups>

<run>

<exclude name=*"Sanity"*></exclude>

</run>

</groups>

<classes>

<class name=*"testngdiscussion.GroupingAnnotations"*/>

</classes>

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

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

Output:

before suite is executing

Before Test method

Before class

Before method

Logout to application

After method

After class

After Test method

After suite is executing

Annotation with respect to groups : We can execute a particular group of test cases with before group and After group annotations. Before group will execute before the execution of inclusion test case group. After group will execute after the execution of inclusion test case group.

Example:

**package** testngdiscussion;

**import** org.testng.annotations.AfterClass;

**import** org.testng.annotations.AfterGroups;

**import** org.testng.annotations.AfterMethod;

**import** org.testng.annotations.AfterSuite;

**import** org.testng.annotations.AfterTest;

**import** org.testng.annotations.BeforeClass;

**import** org.testng.annotations.BeforeGroups;

**import** org.testng.annotations.BeforeMethod;

**import** org.testng.annotations.BeforeSuite;

**import** org.testng.annotations.BeforeTest;

**import** org.testng.annotations.Test;

**public** **class** GroupingAnnotations {

@BeforeGroups("Sanity")

**public** **void** beforeGroup()

{

System.***out***.println("Before group of sanity");

}

@AfterGroups("Sanity")

**public** **void** afterGroup()

{

System.***out***.println("After group of sanity");

}

@BeforeSuite

**public** **void** beforeSuite()

{

System.***out***.println("before suite is executing");

}

@AfterTest

**public** **void** afterTest()

{

System.***out***.println("After Test method");

}

@AfterSuite

**public** **void** afterSuite()

{

System.***out***.println("After suite is executing");

}

@BeforeTest

**public** **void** beforeTest()

{

System.***out***.println("Before Test method");

}

@BeforeClass

**public** **void** beforeClass()

{

System.***out***.println("Before class");

}

@AfterClass

**public** **void** afterClass()

{

System.***out***.println("After class");

}

@AfterMethod

**public** **void** am()

{

System.***out***.println("After method");

}

@Test(groups = "Sanity")

**public** **void** loginTest()

{

System.***out***.println("Login to application");

}

@Test

**public** **void** logoutTest()

{

System.***out***.println("Logout to application");

}

@BeforeMethod

**public** **void** bm()

{

System.***out***.println("Before method");

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

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

<groups>

<run>

<include name=*"Sanity"*></include>

</run>

</groups>

<classes>

<class name=*"testngdiscussion.GroupingAnnotations"*/>

</classes>

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

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

Output:

Before group of sanity

Login to application

After group of sanity

If we change the group by excluding then

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

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

<groups>

<run>

<exclude name=*"Sanity"*></exclude>

</run>

</groups>

<classes>

<class name=*"testngdiscussion.GroupingAnnotations"*/>

</classes>

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

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

Output:

before suite is executing

Before Test method

Before class

Before method

Logout to application

After method

After class

After Test method

After suite is executing

Assertions in TestNG:

Assertions are the check point which can be defined as a code which compares the actual and expected output based on the condition and decide the status of the test case.

Whenever the assertion gets fail then we will get AssertionError but not the exception.

There are 2 types of assertion:

1. Hard Assert or Assert

2. Soft Assert or verify

1. Hard Assert: Hard assert will not allow the test case to execute further steps if the assertion got failed.

There are few methods in hard assert:

Assert.assertTrue(Boolean condition, String message)

Assert.assertEquals(String actual, String expected)

Assert.assertFalse(Boolean condition, String message)

@Test

**public** **void** verifyUrl()

{

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

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

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

driver.get("https://www.facebook.com/");

String url = driver.getCurrentUrl();

String expectedurl = "facebooke";

**boolean** ispresent = url.contains(expectedurl);

Assert.*assertTrue*(ispresent, "Test case is failed please file a bug");

}

}

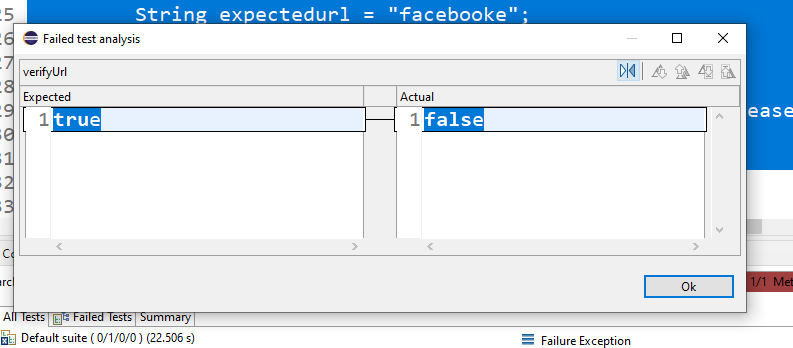
Output:

FAILED: verifyUrl

java.lang.AssertionError: Test case is failed please file a bug expected [true] but found [false]

at org.testng.Assert.fail(Assert.java:99)

at org.testng.Assert.failNotEquals(Assert.java:1037)



Example 2:

@Test

**public** **void** verifyValues()

{

String actualvalue ="Velocity";

String expectedvalue = "velocity";

Assert.*assertFalse*(**false**, expectedvalue);// this is marking the test case as passed.

Assert.*assertEquals*(actualvalue, expectedvalue);// this will mark the test case as failed.

System.***out***.println("After assertion statement");

}

Note : In a test case if any of the assert gets fail then that test case will automatically marked as fail.

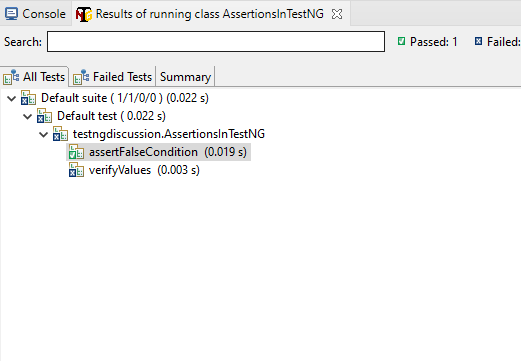
@Test

**public** **void** assertFalseCondition()

{

Assert.*assertFalse*(**false**, "Test case gets fail");

}



2. SoftAssert: Soft Assert will allow the test case to move further even if the assertion got failed and continue to the execution till we call assertAll() method.

If we don’t call assertAll () then it will not mark the status on the basis of assertion. It follow the default behavior for a test case.

Example:

@Test

**public** **void** verifyUrl()

{

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

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

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

driver.get("https://www.facebook.com/");

String url = driver.getCurrentUrl();

String expectedurl = "facebooke";

**boolean** ispresent = url.contains(expectedurl);

SoftAssert sa = **new** SoftAssert();

sa.assertTrue(ispresent, "Test case is failed please file a bug");

System.***out***.println("After assertion statement 1");

System.***out***.println("After assertion statement 2");

sa.assertAll();// at this level we mark the status of test case.

}

}

Output:

After assertion statement 1

After assertion statement 2

FAILED: verifyUrl

java.lang.AssertionError: The following asserts failed:

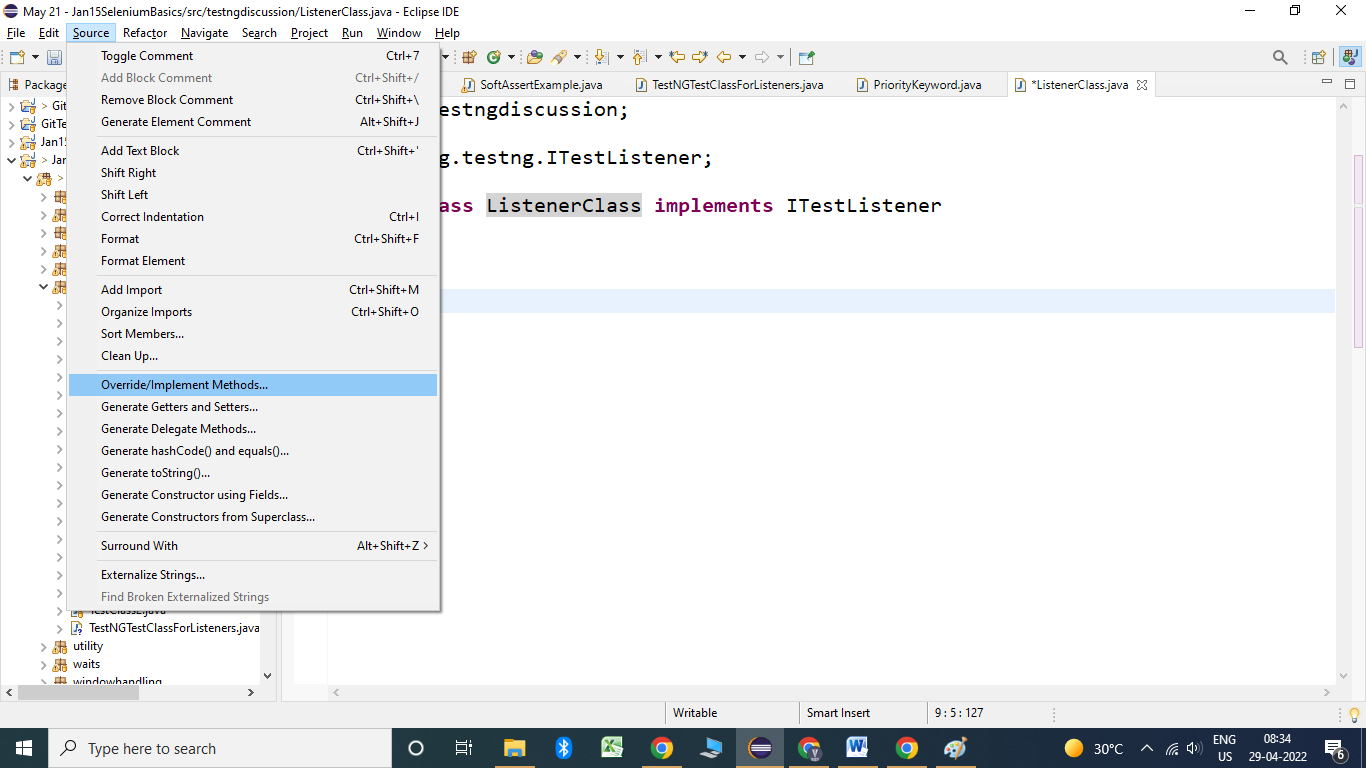
Test case is failed please file a bug expected [true] but found [false]

**Listeners in TestNG:**

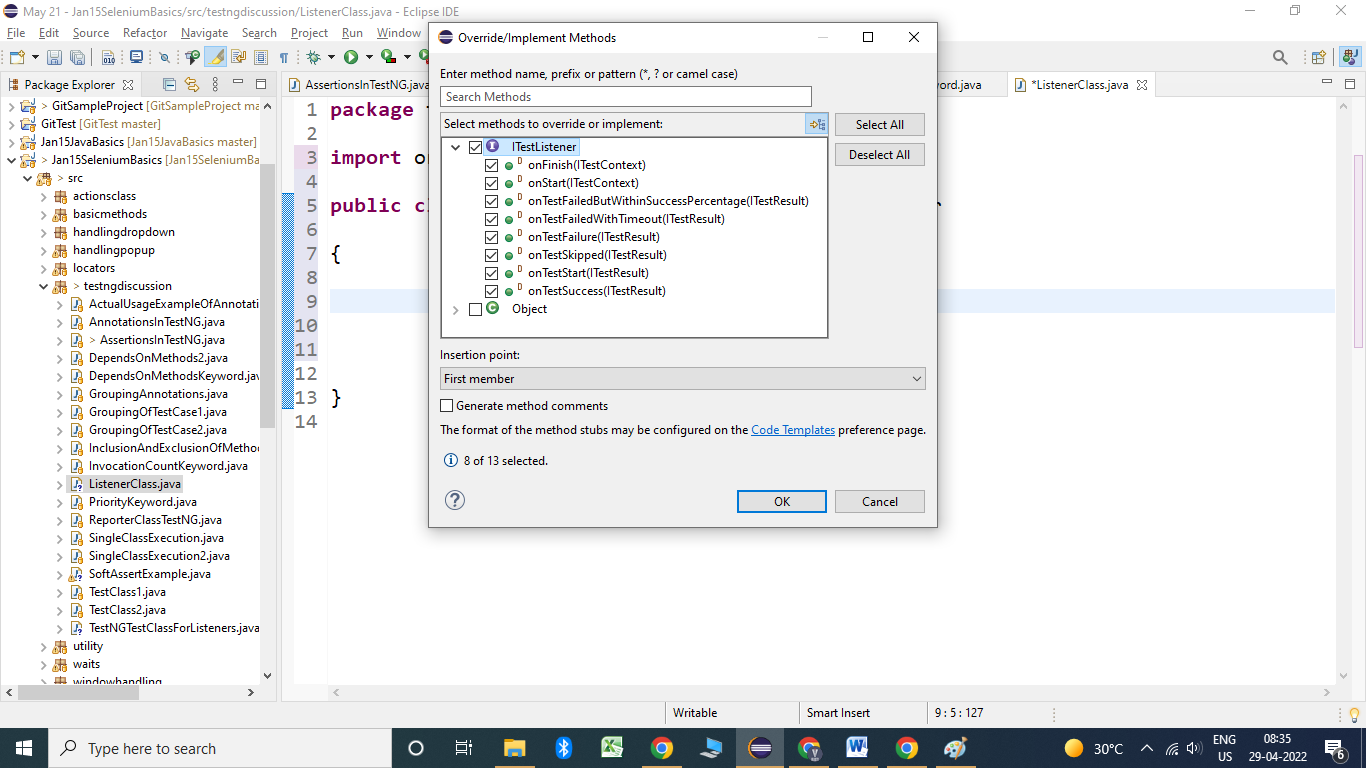
Listener is an entity which listen the events of the test case (statuses of test cases) and perform the action based on the code written inside the particular event specific method (onTestStart(), onTestSuccess() etc) post to the execution of a test case.

To implement the listener we have to use an interface ITestListener.

To implement the ITestListener – go to source – Override/implement methods



Select all the methods in the check box and click on ok



Example”

**public** **class** ListenerClass **implements** ITestListener

{

@Override

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

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

}

@Override

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

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

}

@Override

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

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

}

@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());

}

}

**package** testngdiscussion;

**import** org.testng.Assert;

**import** org.testng.annotations.Test;

**public** **class** TestNGTestClassForListeners {

@Test

**public** **void** login()

{

System.***out***.println("Login to application");

Assert.*assertTrue*(**false**);

}

@Test

**public** **void** dashBoard()

{

System.***out***.println("DashBoard Test case");

}

@Test

**public** **void** profile()

{

System.***out***.println("Profile Test case");

}

@Test(dependsOnMethods = "login")

**public** **void** home()

{

System.***out***.println("home Test case");

}

@Test

**public** **void** logout()

{

System.***out***.println("logout Test case");

Assert.*assertEquals*(**false**, **true**);

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

<listeners>

<listener class-name=*"testngdiscussion.ListenerClass"*></listener>

</listeners>

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

<classes>

<class name=*"testngdiscussion.TestNGTestClassForListeners"*/>

</classes>

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

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

Output:

Test started Listener test

Test case started :login

Login to application

Test case Failed :login

Test case started :dashBoard

DashBoard Test case

Test case passed :dashBoard

Test case started :logout

logout Test case

Test case Failed :logout

Test case started :profile

Profile Test case

Test case passed :profile

Test case started :home

Test case got skipped :home

Test completed Listener test

Listeners with Actual test case:

**public** **class** KiteLogin {

**static** WebDriver *driver*;

@Test(priority = 1)

**public** **void** navToApp()

{

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

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

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

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

}

@Test(priority = 2)

**public** **void** loginToApp()

{

*driver*.manage().timeouts().~~implicitlyWait~~(60, TimeUnit.***SECONDS***);

*driver*.findElement(By.*xpath*("//\*[@id='userid']")).sendKeys("DAA677");

*driver*.findElement(By.*xpath*("//\*[@id='password']")).sendKeys("Velocity@123");

*driver*.findElement(By.*xpath*("//\*[@type='submit']")).click();

*driver*.findElement(By.*xpath*("//\*[@id='pin']")).sendKeys("866918");

*driver*.findElement(By.*xpath*("//\*[@type='submit']")).click();

*driver*.findElement(By.*xpath*("//\*[@type='button']"));

String url = *driver*.getCurrentUrl();

String expected = "dashboard";

**boolean** iscontains = url.contains(expected);

System.***out***.println(iscontains);

Assert.*assertEquals*(iscontains, **false**);

}

**public** **class** ListenerClass **extends** KiteLogin **implements** ITestListener

{

@Override

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

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

}

@Override

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

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

}

@Override

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

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

**try** {

ScreenShot.*captureScreenshot*(*driver*, result.getName());

} **catch** (IOException e) {

e.printStackTrace();

}

}

@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());

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

<listeners>

<listener class-name=*"testngdiscussion.ListenerClass"*></listener>

</listeners>

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

<classes>

<class name=*"testngdiscussion.KiteLogin"*/>

</classes>

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

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

Output:

Test started :Listener test

Test case started :navToApp

Test case passed :navToApp

Test case started :loginToApp

true

Test case Failed :loginToApp

Test completed :Listener test

Parameterization in TestNG:

Whenever we wants to take the data from xml file then we can mention it in xml and read it to the test case in the form of arguments. It can be done by using @Parameters annotations.

Note: Parameter being used by one test case cannot be use for other test case.

Example:

**package** testngdiscussion;

**import** java.util.concurrent.TimeUnit;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** org.testng.Assert;

**import** org.testng.annotations.Parameters;

**import** org.testng.annotations.Test;

**public** **class** ParameterizationOfTestCase {

**static** WebDriver *driver*;

@Test(priority = 1)

**public** **void** navToApp()

{

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

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

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

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

}

@Parameters({"username", "password"})

@Test(priority = 2)

**public** **void** loginToApp(String usr, String pass)

{

*driver*.manage().timeouts().~~implicitlyWait~~(60, TimeUnit.***SECONDS***);

*driver*.findElement(By.*xpath*("//\*[@id='userid']")).sendKeys(usr);

*driver*.findElement(By.*xpath*("//\*[@id='password']")).sendKeys(pass);

*driver*.findElement(By.*xpath*("//\*[@type='submit']")).click();

*driver*.findElement(By.*xpath*("//\*[@id='pin']")).sendKeys("866918");

*driver*.findElement(By.*xpath*("//\*[@type='submit']")).click();

*driver*.findElement(By.*xpath*("//\*[@type='button']"));

String url = *driver*.getCurrentUrl();

String expected = "dashboard";

**boolean** iscontains = url.contains(expected);

System.***out***.println(iscontains);

Assert.*assertEquals*(iscontains, **false**);

}

}

Xml:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

<parameter name=*"username"* value=*"DAA677"*></parameter>

<parameter name=*"password"* value=*"Velocity@123"*></parameter>

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

<classes>

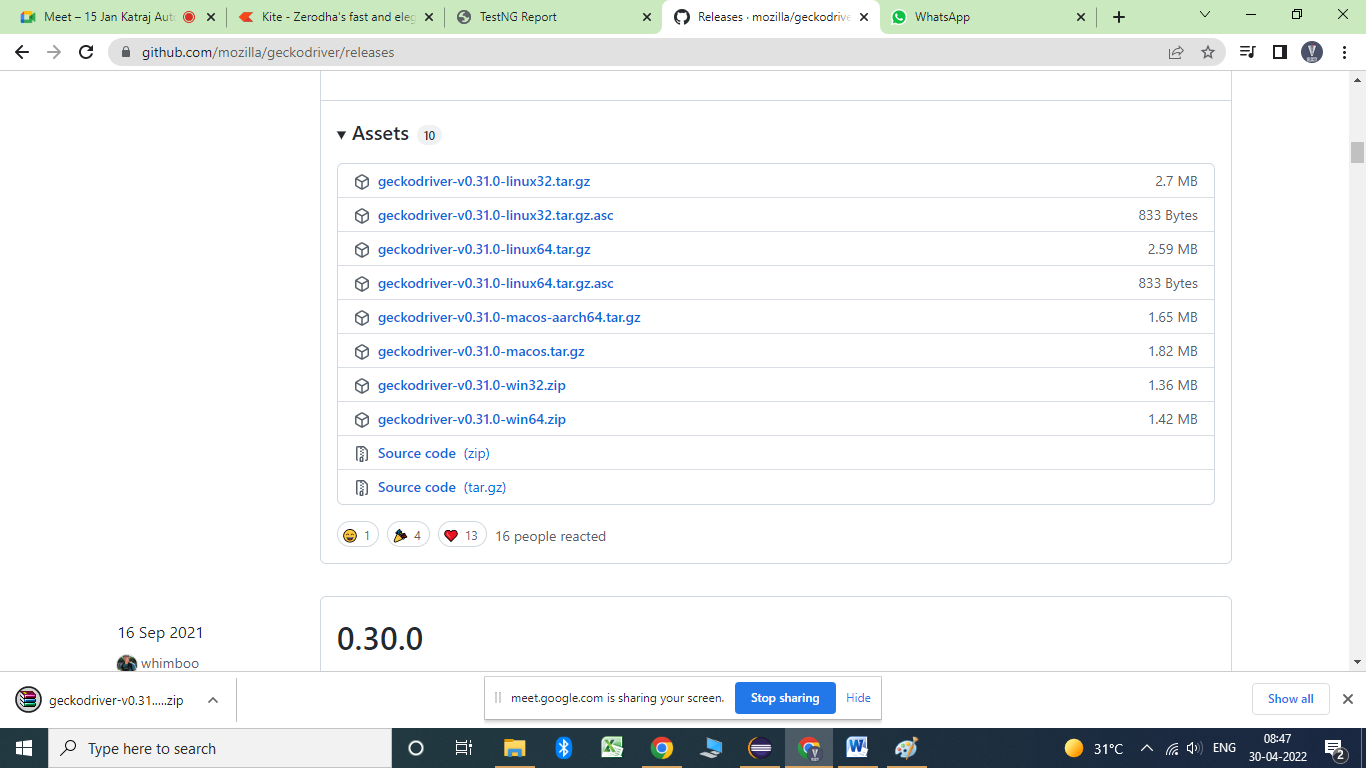
<class name=*"testngdiscussion.ParameterizationOfTestCase"*/>

</classes>

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

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

Gecko driver download: <https://github.com/mozilla/geckodriver/releases>



Parallel execution in TestNG:

We can execute Methods, Classes and <Test> in parallel with the help of TestNG. For that purpose we have to use parallel keyword inside the xml file and then on the basis of keyword it will execute them in parallel.

1. Test case parallel execution:

@Test

**public** **void** googleTest() {

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

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

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

driver.get("https://www.google.com");

}

@Test

**public** **void** facebookTest() {

System.*setProperty*("webdriver.gecko.driver", "E:\\desktop\\Katraj\\15 Jan\\Selenium\\chromedriver.exe");

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

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

driver.get("https://www.facebook.com");

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

<test thread-count=*"10"* name=*"Test"* parallel = *"methods"*>

<classes>

<class name=*"testngdiscussion.ParallelTestCaseExecution"*/>

</classes>

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

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

Assignment:

1 test case --- Chrome and Firefox execution in Parallel

Parallel class execution:

**public** **class** ParallelTestClass1 {

@Test

**public** **void** login()

{

System.***out***.println("Login to application");

}

@Test

**public** **void** dashBoard()

{

System.***out***.println("DashBoard Test case");

}

}

**public** **class** ParallelTestClass2 {

@Test

**public** **void** profile()

{

System.***out***.println("Profile test case");

}

@Test

**public** **void** logout()

{

System.***out***.println("logout Test case");

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"*>

<test thread-count=*"10"* name=*"Test"* parallel = *"classes"*>

<classes>

<class name=*"testngdiscussion.ParallelTestClass1"*/>

<class name=*"testngdiscussion.ParallelTestClass2"*/>

</classes>

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

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

**Test Tag execution in Parallel mode:**

**public** **class** ParallelTestClass1 {

@Test

**public** **void** login()

{

System.***out***.println("Login to application");

}

@Test

**public** **void** dashBoard()

{

System.***out***.println("DashBoard Test case");

}

}

**public** **class** ParallelTestClass2 {

@Test

**public** **void** profile()

{

System.***out***.println("Profile test case");

}

@Test

**public** **void** logout()

{

System.***out***.println("logout Test case");

}

}

<suite name=*"Suite"* parallel =*"tests"*>

<test thread-count=*"5"* name=*"Set 1"*>

<classes>

<class name=*"testngdiscussion.ParallelTestClass1"*/>

</classes>

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

<test thread-count=*"5"* name=*"Set 2"*>

<classes>

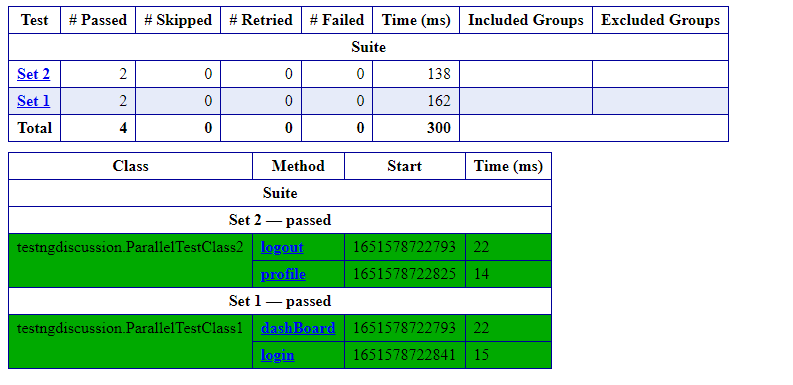
<class name=*"testngdiscussion.ParallelTestClass2"*/>

</classes>

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

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

**Output:**



Scenario- 1 testcase – Execution at different urls at the same time (parallel mode)

**public** **class** ParallelTestTagExecutionActual {

WebDriver driver;

@Parameters("url")

@Test(priority = 1)

**public** **void** navToApp(String pageurl)

{

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

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

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

driver.get(pageurl);

}<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Suite"* parallel= *"tests"*>

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

<parameter name = *"url"* value = *"https://www.google.com"*></parameter>

<classes>

<class name=*"testngdiscussion.ParallelTestTagExecutionActual"*/>

</classes>

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

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

<parameter name = *"url"* value = *"https://kite.zerodha.com/"*></parameter>

<classes>

<class name=*"testngdiscussion.ParallelTestTagExecutionActual"*/>

</classes>

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

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

Failed test case execution: We can execute the failed test case in the previous execution cycle by going to the test output folder and execute the testng-failed.xml file which contains all the data specific to the failed test cases then it will execute only the failed test cases.

Example:

**public** **class** FailedTestCaseExecution {

@Test

**public** **void** login()

{

System.***out***.println("Login to application");

Assert.*assertEquals*(**false**, **true**);

}

@Test

**public** **void** dashBoard()

{

System.***out***.println("DashBoard Test case");

}

@Test

**public** **void** profile()

{

System.***out***.println("profile to application");

}

@Test

**public** **void** logout()

{

System.***out***.println("logout Test case");

}

}

After the execution Login test case will get fail then testng-failed.xml contains the data:

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">

<suite name=*"Failed suite [Failed suite [Failed suite [Default suite]]]"* guice-stage=*"DEVELOPMENT"*>

<test thread-count=*"5"* name=*"Default test(failed)(failed)(failed)"*>

<classes>

<class name=*"testngdiscussion.FailedTestCaseExecution"*>

<methods>

<include name=*"login"*/>

</methods>

</class> <!-- testngdiscussion.FailedTestCaseExecution -->

</classes>

</test> <!-- Default test(failed)(failed)(failed) -->

</suite> <!-- Failed suite [Failed suite [Failed suite [Default suite]]] -->

Once we execute it, this will execute only the failed test case.

Note: To execute the skipped test case we have to execute the same testng-failed.xml file then it will execute only failed test cases and those which got skipped.