**What is TestNG?**

TestNG is a testing framework inspired from JUnit and NUnit but introducing some new functionalities that make it more powerful and easier to use, such as:

* **Annotations**.
* Run your tests in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class, etc...).
* Test that your code is multithread safe.
* Flexible test configuration.
* **Support for data-driven testing (with @DataProvider).**
* Support for parameters.
* Powerful execution model (no more TestSuite).
* **Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).**
* Embeds BeanShell for further flexibility.
* Default JDK functions for runtime and logging (no dependencies).
* Dependent methods for application server testing.

**What are the advantages of TestNG?**

1. TestNG provides parallel execution of test methods(testing.xml)

<suite name="Parallel test suite" **parallel="methods"** thread-count="2">

<test name="Regression 1">

<classes>

<class name="com.parallel.TestParallelOne"/>

</classes>

</test>

</suite>

1. It allows to assign priority to test methods
2. It allows grouping of test methods into test groups
3. It has support for parameterizing test cases using @Parameters annotation
4. It allows data driven testing using @DataProvider annotation
5. It has different assertions that helps in checking the expected and actual results

assertEquals

assertNotEquals

assertTrue

assertFalse

assertNull

assertNotNull

1. Detailed (HTML) reports

**What are the annotations available in TestNG?**

@BeforeTest 3  
@AfterTest 3  
@BeforeClass 2  
@AfterClass 2  
@BeforeMethod 1  
@AfterMethod 1  
@BeforeSuite 4  
@AfterSuite 4  
@BeforeGroups  
@AfterGroups  
@Test 0

package softwareTestingMaterial;

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 TestngAnnotation {

// Test Case 1

@Test

public void testCase1() {

System.out.println("in Test Case 1");

}

// Test Case 2

@Test

public void testCase2() {

System.out.println("in Test Case 2");

}

@BeforeMethod

public void beforeMethod() {

System.out.println("in Before Method");

}

@AfterMethod

public void afterMethod() {

System.out.println("in After Method");

}

@BeforeClass

public void beforeClass() {

System.out.println("in Before Class");

}

@AfterClass

public void afterClass() {

System.out.println("in After Class");

}

@BeforeTest

public void beforeTest() {

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

}

@AfterTest

public void afterTest() {

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

}

@BeforeSuite

public void beforeSuite() {

System.out.println("in Before Suite");

}

@AfterSuite

public void afterSuite() {

System.out.println("in After Suite");

}

}

**OUTPUT:**

[TestNG] Running:

in Before Suite

in Before Test

in Before Class

in Before Method

in Test Case 1

in After Method

in Before Method

in Test Case 2

in After Method

in After Class

in After Test

in After Suite

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

Default suite

Total tests run: 2, Failures: 0, Skips: 0

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

**Can you arrange the below testng.xml tags from parent to child?**

<test>

<suite>

<class>

<methods>

<classes>

**ANS**

|  |  |
| --- | --- |
| 1  2  3  4  5 | <suite>  <test>  <classes>  <class>  <methods> |

**How to create and run *testng.xml?***

In TestNG framework, we need to create ***testng.xml*** file to create and handle multiple test classes. We do configure our test run, set test dependency, include or exclude any test, method, class or package and set priority etc in the xml file.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <suite name="softwaretestingmaterial">  <test name="testngTest">  <classes>  <class name="softwareTestingMaterial.STMTestNGClass" />  </classes>  </test>  </suite> |

**Note:** You can choose any name for your Test Suite & Test Name as per your need.

**What is the importance of testng.xml file?**

In a Selenium TestNG project, we use *testng.xml* file to configure the complete test suite in a single file. Some of the features are as follows.

* testng.xml file allows to include or exclude the execution of test methods and test groups
* It allows to pass parameters to the test cases
* Allows to add group dependencies

Allows to add priorities to the test cases

Allows to configure parallel execution of test cases

<suite name="TestSuite" thread-count="3" parallel="methods" >

<test name="testGuru">

<classes>

<class name="TestGuru99MultipleSession">

</class>

</classes>

</test>

</suite>

* Allows to parameterize the test cases

**How to pass parameter through testng.xml file to a test case?**

|  |  |
| --- | --- |
|  | package softwareTestingMaterial;    import org.testng.annotations.Parameters;  import org.testng.annotations.Test;    public class ParameterizedTest {    *@Test*  *@Parameters*("browser")  public void parameterizedTest(String browser){  if(browser.equals("firefox")){  System.out.println("Open Firefox Driver");  }else if(browser.equals("chrome")){  System.out.println("Open Chrome Driver");  }  }  }  **testng.xml file**  <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >    <suite name="softwaretestingmaterial">  <test name="testngTest">  <parameter name="browser" value="firefox"/>  <classes>  <class name="softwareTestingMaterial.ParameterizedTest" />  </classes>  </test>  </suite>  **Console Output:**  [TestNG] Running:    Open Firefox Driver    ===============================================  softwaretestingmaterial  Total tests run: 1, Failures: 0, Skips: 0  **.java**  @Parameters({"user\_name","password"})  @Test  **public** **void** loginapp()  {  driverget(“appname”);  driver.findElement(By.id(“login”)).sendkeys(user\_name);  driver.findElement(By.id(“password”)).sendkeys(password);  }  **Testing.xml**  <Suite name = *“suitename”*>  <test name =*”testname”*>  <parameter name =*”user\_name”* value=*”user1”*/>  <parameter password =*”password”* value =*”pass1”*/>  <Classes>  <class name =*”passingparameters”*/>  <classes/>  <test/>  <Suite/>  **What is TestNG Assert and list out common TestNG Assertions?**  TestNG Asserts help us to verify the condition of the test in the middle of the test run. Based on the TestNG Assertions, we will consider a successful test only if it is completed the test run without throwing any exception.  **If hard assert fail then it throws an exception and move to the next @Test**  Some of the common assertions supported by TestNG are  Assert.*assertEquals*(driver.getTitle(), actualTitle);(**if fails:** **java.lang.AssertionError: expected [GoogleMail] but found [Gmail - Email from Google]**)   * assertEqual(String actual,String expected, String message) * assertEquals(boolean actual,boolean expected) * assertTrue(condition)( **assertTrue("Verification Failed: The radio button of male status is not selected for the user", driver.findElement(By.id(“gender”)).isSelected());**) * assertTrue(condition, message) * assertFalse(condition) * assertFalse(condition, message)   **Soft Assert in TestNG and Hard Assert in TestNG**  package softwareTestingMaterial;  package softwareTestingMaterial;  import org.testng.Assert;  import org.testng.annotations.Test;  import org.testng.asserts.SoftAssert;  public class SoftAssertion {  @Test  public void softAssert(){  SoftAssert softAssertion= new SoftAssert();  System.out.println("softAssert Method Was Started");  softAssertion.assertTrue(false);  System.out.println("softAssert Method Was Executed");  softAssertion.assertAll();**//throw exception of soft assert and fail it**  }    @Test  public void hardAssert(){  System.out.println("hardAssert Method Was Started");  Assert.assertTrue(false);  System.out.println("hardAssert Method Was Executed");  }  }  **OUTPUT:**  [TestNG] Running:  C:\Users\Administrator\AppData\Local\Temp\testng-eclipse--2097831536\testng-customsuite.xml  hardAssert Method Was Started  softAssert Method Was Started  softAssert Method Was Executed  ===============================================  Default suite  Total tests run: 2, Failures: 1, Skips: 0  =============================================== |
|  |  |

**What is exception test in TestNG?**

TestNG gives an option for tracing the Exception handling of code. You can verify whether a code throws the expected exception or not. The expected exception to validate while running the test case is mentioned using the **expectedExceptions** attribute value along with @Test annotation.

expectedExceptions = {ArithmeticException.class, NullPointerException.class }

**Java Code(which gives exception):**

package softwareTestingMaterial;

import org.testng.annotations.Test;

public class TestNGException {

*@Test*

public void testException() {

System.out.println("SoftwareTestingMaterial.com");

int i = 1 / 0;

}

}

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

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

<suite name="softwaretestingmaterial">

<test name="testngTest">

<classes>

<class name="softwareTestingMaterial.TestNG " />

</classes>

</test>

</suite>

**Console Output:**

[TestNG] Running:

SoftwareTestingMaterial.com

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

Default suite

Total tests run: 1, Failures: 1, Skips: 0

**JAVA CODE(Correct):**

import org.testng.annotations.Test;

public class TestNGException {

*@Test*(**expectedExceptions = ArithmeticException.class**)

public void testException() {

System.out.println("SoftwareTestingMaterial.com");

int i = 1 / 0;

}

}

**OUTPUT:**

[TestNG] Running:

SoftwareTestingMaterial.com

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

Default suite

Total tests run: 1, Failures: 0, Skips: 0

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

**How to set test case priority in TestNG?**

We use priority attribute to the @Test annotations. In case priority is not set then the test scripts execute in alphabetical order.

**-1, -2 …. Are worked in priority**

**JAVA**

// TestNG Interview Questions

package TestNG;

import org.testng.annotations.\*;

public class PriorityTestCase{

@Test(priority=0)

public void testCase1() {

system.out.println("Test Case 1");

}

@Test(priority=1)

public void testCase2() {

system.out.println("Test Case 2");

}

}

**OUTPUT**

Test Case 1

Test Case 2

**What is default priority in testing?**

The **default priority** of test when not specified is integer value 0

**How can we create data driven framework using TestNG?**

By using *@****DataProvider*** annotation, we can create a Data Driven Framework.

**Exmple1:**

@DataProvider(name="getData")

public Object[][] getData(){

//Object [][] data = new Object [rowCount][colCount];

Object [][] data = new Object [2][2];

data [0][0] = "FirstUid";

data [0][1] = "FirstPWD";

data[1][0] = "SecondUid";

data[1][1] = "SecondPWD";

return data;

}

**Example2:**

**public** **class** Data\_Provider{

WebDriver driver;

@BeforeMethod

**public** **void** setup()

{

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

driver = **new** ChromeDriver();

driver.manage().timeouts().pageLoadTimeout(20, TimeUnit.***SECONDS***);

driver.manage().deleteAllCookies();

driver.get("http://pwf-wprevamp.kelltontech.net/personal-loan-application");

}

@DataProvider

**public** Object[][] getLogindata() {

Object data[][]=TestUtil.*getTestData*("Sheet1");

**return** data;

}

@Test(dataProvider="getLogindata")

**public** **void** setData(String firstname, String password, String email)

{

driver.findElement(By.*id*("first\_name")).sendKeys(firstname);

driver.findElement(By.*id*("last\_name")).sendKeys(password);

driver.findElement(By.*id*("email")).sendKeys(email);

}

**Testutil**

**public** **class** TestUtil {

**static** Workbook *book*;

**static** Sheet *sheet*;

**static** String *TESTDATA\_SHEET\_PATH*="D:\\Workspace\\Selenium\_Practice\\src\\main\\java\\testdataexcel\\dataprovider.xlsx";

**public** **static** Object[][] getTestData(String sheetName) {

FileInputStream file = **null**;

**try** {

file = **new** FileInputStream(*TESTDATA\_SHEET\_PATH*);

} **catch** (FileNotFoundException e) {

e.printStackTrace();

}

**try** {

*book* = WorkbookFactory.*create*(file);

} **catch** (InvalidFormatException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

*sheet* = *book*.getSheet(sheetName);

Object[][] data = **new** Object[*sheet*.getLastRowNum()][*sheet*.getRow(0).getLastCellNum()];

// System.out.println(sheet.getLastRowNum() + "--------" +

// sheet.getRow(0).getLastCellNum());

**for** (**int** i = 0; i < *sheet*.getLastRowNum(); i++) {

**for** (**int** k = 0; k < *sheet*.getRow(0).getLastCellNum(); k++) {

data[i][k] = *sheet*.getRow(i + 1).getCell(k).toString();

System.***out***.println(data[i][k]);

}

}

**return** data;

}

**How to run a group of test cases using TestNG?**

**Testcase 1**

package softwareTestingMaterial;

import org.testng.annotations.Test;

public class TestCase1 {

*@Test* (groups = { "smokeTest", "functionalTest" })

public void loginTest(){

System.out.println("Logged in successfully");

}

}

**Testcase 2**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | package softwareTestingMaterial;  import org.testng.annotations.Test;  public class TestCase2 {  @Test (groups = { "functionalTest" })  public void composeMail(){  System.out.println("Mail Sent");  }  }  **testng.xml:**  <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >    <suite name="softwaretestingmaterial">  <test name="testngTest">  <groups>  <run>  <include name="smokeTest" />  </run>  </groups>    <classes>  <class name="softwareTestingMaterial.TestCase1" />  <class name="softwareTestingMaterial.TestCase2" />  </classes>  </test>  </suite>  **Console Output:**  [TestNG] Running:    Logged in successfully    ===============================================  softwaretestingmaterial  Total tests run: 1, Failures: 0, Skips: 0  **How to run test cases in parallel using TestNG?**  package softwareTestingMaterial;    import org.openqa.selenium.WebDriver;  import org.openqa.selenium.chrome.ChromeDriver;  import org.openqa.selenium.firefox.FirefoxDriver;  import org.testng.annotations.Test;    public class ParallelTests {    *@Test*  public void getFirefox(){                  //System.setProperty("webdriver.gecko.driver", "geckodriver.exe path");  System.setProperty("webdriver.gecko.driver", "D://Selenium Environment//Drivers//geckodriver.exe");                  System.out.println("GetFirefox Method is running on Thread : " + Thread.currentThread().getId());  WebDriver driver = new FirefoxDriver();  driver.get("http://www.softwaretestingmaterial.com");  driver.close();  }    *@Test*  public void getChorme(){                  //System.setProperty("webdriver.chrome.driver", "chromedriver.exe path");  System.setProperty("webdriver.chrome.driver", "D://Selenium Environment//Drivers//chromedriver.exe");                  System.out.println("GetChrome Method is running on Thread : " + Thread.currentThread().getId());  WebDriver driver = new ChromeDriver();  driver.get("http://www.softwaretestingmaterial.com");  driver.close();  }    }  **Testng.XML**   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  | <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">  <suite name="softwaretestingmaterial" parallel="methods" thread-count="2">  <test name="testngTest">  <classes>  <class name="softwareTestingMaterial.ParallelTests" />  </classes>  </test>  </suite>  Once you run the testng.xml using the above code, you could see both the browsers in action at a time.  **How to exclude a particular test method from a test case execution?**  <class name=*"com.qa.page.test.Business\_LoanDErrorTest"*>  <methods>  < exclude name=*"Test"* />  </methods>  </class>  **How to exclude a particular test group from a test case execution?**  **Testcase1**   |  |  |  |  | | --- | --- | --- | --- | |  | package softwareTestingMaterial;  import org.testng.annotations.Test;  public class TestCase1 {  *@Test* (groups = { "smokeTest", "functionalTest" })  public void loginTest(){  System.out.println("Logged in successfully");  }  }  **Testcase1**   |  |  | | --- | --- | |  | package softwareTestingMaterial;  import org.testng.annotations.Test;  public class TestCase2 {  @Test (groups = { "functionalTest" })  public void composeMail(){  System.out.println("Mail Sent");  }  } |   **Testing.xml** |   <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >    <suite name="softwaretestingmaterial">  <test name="testngTest">    <groups>             <run>  <exclude name="smokeTest"/>              <include name="functionalTest"/>  </run>    </groups>    <classes>  <class name="softwareTestingMaterial.TestCase1" />  <class name="softwareTestingMaterial.TestCase2" />  </classes>  </test>  </suite>  **Console Output:**  [TestNG] Running:    Mail Sent    ===============================================  softwaretestingmaterial  Total tests run: 1, Failures: 0, Skips: 0  ===============================================  **How to create Group of Groups in TestNG?**   |  |  | | --- | --- | |  | <groups>     <define name="all">  <include name="smokeTest"/>  <include name="functionalTest"/>     </define>     <run>           <include name="all" />     </run>  </groups>  **How to disable/skip/ignore a test case in TestNG ?**  @Test(enabled = false)  **How to skip a @Test method from execution in TestNG?**  package softwareTestingMaterial;    import org.testng.annotations.Test;  import org.testng.SkipException;    public class SkipTestCase {    @Test  public void aSkipTest(){  String a ="Skip Test";  if(a.equals("Skip Test")){  **throw new SkipException("Skipping - This is not ready for testing ");**  }else{  System.out.println("I am in else condition");  }  System.out.println("I am out of the if else condition");  }    @Test  public void nonSkipTest(){  System.out.println("No need to skip this test");  }    }  [TestNG] Running:    No need to skip this test  PASSED: nonSkipTest  SKIPPED: aSkipTest    ===============================================  Default suite  Total tests run: 2, Failures: 0, Skips: 1  =============================================== |   **How TestNG allows to state dependencies?**  package softwareTestingMaterial;    import org.testng.annotations.Test;    public class DependsOnMethodsTestCase {    *@Test***(dependsOnMethods = {"testCase2"}**)  public void testCase1(){  System.out.println("Test Case 1");  }  *@Test*  public void testCase2(){  System.out.println("Test Case 2");  }    }  **//if testcase2 is fail then testcase1 failed too because it depends on testcase2**  [TestNG] Running:    Test Case 2  Test Case 1    ===============================================  softwaretestingmaterial  Total tests run: 2, Failures: 0, Skips: 0  ===============================================  **Way 2:**  package softwareTestingMaterial;    import org.testng.annotations.Test;    public class DependsOnMethodsTestCase {    *@Test*(groups = {"FirstGroup"})  public void testCase1(){  System.out.println("Test Case 1");  }  *@Test*(groups = {"SecondGroup"})  public void testCase2(){  System.out.println("Test Case 2");  }    }  <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >    <suite name="softwaretestingmaterial">  <test name="testngTest">    <groups>                      <dependencies>                            <group name="FirstGroup**" depends-on="SecondGroup"**></group>                      </dependencies>                  </groups>    <classes>  <class name="softwareTestingMaterial.DependsOnMethodsTestCase" />  </classes>    </test>  </suite> | |  |  |   **Junit vs Testng**  **(**<https://www.youtube.com/watch?v=OWXWQE4pL_0>**)**    **What are the different ways to produce reports for TestNG results?**  **Listeners** implement the interface *org.testng.ITestListener* and are notified in real time of when a test starts, passes, fails, etc…  <https://www.youtube.com/watch?v=-a5aHd0gsGw> Mention what are the types of [Listeners in TestNG](https://www.guru99.com/listeners-selenium-webdriver.html)? The types of Listeners in TestNG are,   * IAnnotationTransformer * IAnnotationTransformer2 * IConfigurable * IConfigurationListener * IExecutionListener * IHookable * IInvokedMethodListener * IInvokedMethodListener2 * IMethodInterceptor * IReporter * ISuiteListener * ITestListener   **How to write regular expression In testng.xml file to search @Test methods containing “smoke” keyword.**  Regular expression to find @Test methods containing keyword “smoke” is as mentioned below.  <methods>       <include name=".\*smoke.\*"/>  </methods>  **What is the time unit we specify in test suites and test cases?**  We specify the time unit in test suites and test cases is in *milliseconds*.  **List out various ways in which TestNG can be invoked?**  TestNG can be invoked in the following ways   * Using Eclipse IDE * Using ant build tool * From the command line * Using IntelliJ’s IDEA |

**What is the use of @Test(invocationCount=x)?**

The *invocationcount* attribute tells how many times TestNG should run a test method

@Test(invocationCount = 10)

public void testCase1(){

}

Here testcase1 will run for 10 times

**What does the test timeout mean in TestNG?**

The maximum number of milliseconds a test case should take.

@Test(threadPoolSize = 3, invocationCount = 10,  timeOut = 10000)

public void testCase1(){

In this example, the function testCase1 will be invoked ten times from three different threads. Additionally, a time-out of ten seconds guarantees that none of the threads will block on this thread forever.

What is the use of @Factory annotation?

The @Factory annotation is useful when we want to run multiple test cases through a single test class. It is mainly used for the dynamic execution of test cases.

**testcase1.java**

**package** com.javatpoint;

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

**public** **class** Testcase1

{

@Test

**public** **void** test1()

{

System.out.println("testcase 1");

}

}

**testcase2.java**

**package** com.javatpoint;

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

**public** **class** Testcase2

{

@Test

**public** **void** test1()

{

System.out.println("testcase 2");

}

}

**Factory.java**

**import** org.testng.annotations.Factory;

**public** **class** Factory1

{

@Factory

**public** Object[] getTestClasses()

{

Object tests[]=**new** Object[2];

tests[0]=**new** Testcase1();

tests[1]=**new** Testcase2();

**return** tests;

}

}

### What is the difference between @Factory and @DataProvider annotation?

**@DataProvider:** It is annotation used by TestNG to execute the test method multiple numbers of times based on the data provided by the DataProvider.

**@Factory:** It is annotation used by the TestNG to execute the test methods present in the same test class using different instances of the respective class.

**How to create an XML file in TestNG?**

**Answer:**Go to the src folder -> click on file ->enter the name of the file(mostly written testng.xml)

Click on finish.

Now we have a blank xml file. Here, we have to mention the project name and the classes to be executed along with the package name as shown below.

<Suite name = *"Testing project"*>

<test name = *"testing feature 1"*>

<classes>

<class name = *"packagename.name of class1"*/>

<class name = *"packagename.name of class1"*/>

<class name = *"packagename.name of class1"*/>

<class name = *"packagename.name of class1"*/>

</classes>

</test>

</Suite>

To run this file, we have to go to testng.xml in the package explorer right click and run as -> Testng suite

**How do you control the execution of your test cases/test classes?**

<classes>

<class name=*"com.qa.page.test.Professional\_keydErrorTest"*>

<methods>

<include name=*"Test"* />

</methods>

</class>

<class name=*"com.qa.page.test.Professional\_loandErrorTest"*>

<methods>

<include name=*"Test"* />

</methods>

</class>

</classes

**Maven commands to execute**

### mvn test

This command is used to run the test cases of the project

**What is the current version of maven used?**

Maven 3.6.3(as of April 2020 )

What is listners in testing?

It allow us to customize logs or reports of testing

ITestListner : It is a interface in testng which contains unimplemented methods so we can implement ITestListner and modify all the unimplemented methods accordingly

TESTNG.XML

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

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

<suite name=*"Suite"*>

<listeners>

<listener class-name=*"ListnersDemo.Listners"* />

</listeners>

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

<classes>

<class name=*"ListnersDemo.mylistnertestcase"* />

</classes>

</test>

</suite>

**LISTENERS.java**

package ListnersDemo;

public class Listners implements ITestListener {

public void onTestStart(ITestResult result) {

System.out.println("testcase are started and details are" + result.getName());

}

public void onTestSuccess(ITestResult result) {

System.out.println("testcase are success and details are" + result.getName());

}

public void onTestFailure(ITestResult result) {

System.out.println("testcase are fail and details are" + result.getName());

}

public void onTestSkipped(ITestResult result) {

System.out.println("testcase are skipped and details are" + result.getName());

}

public void onTestFailedButWithinSuccessPercentage(ITestResult result) {

System.out.println("testcase are FailedButWithinSuccessPercentage and details are" + result.getName());

}

public void onStart(ITestContext context) {

System.out.println("testcase are start contex and details are" + context.getName());

}

public void onFinish(ITestContext context) {

System.out.println("testcase are finish and details are" + context.getName());

}

}

Mylistnertestcase.jave

**package** ListnersDemo;

//@Listeners(ListnersDemo.Listners.class)

**public** **class** mylistnertestcase {

@Test

**public** **void** google() {

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

WebDriver driver = **new** ChromeDriver();

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

System.***out***.println("Title is : " + driver.getTitle());

driver.quit();

}

}