**TestNG**

**Overview of TestNG**

In selenium using java there are two frameworks available

i)**Junit and**

ii)**TestNG** (Next Generation)

**TestNG FrameWork - is a Testing framework**

**TestNG is also used by the software testers to efficiently run the automated test scripts created in Selenium Webdriver. Its full form is the “Testing New Generation” framework.**

Introducing some new functionalities that make it more powerful and easier to use.

is designed for Unit testing and Integration testing later it is used for system testing

TestNG is an open source framework

it is inspired Junit and Nunit (for dot net Nunit)

TestNg is a testing framework designed to simplify a broad range of test

**Advantages of testNg**

**TestNg annotations** are easily

**Prioritise** the test cases

**Grouping test cases**

Generate detailed **test Reports** (html reports)

**Data Driven testing**

Supports for **data parameterization**

TestNG Annotations

TestNg annotations are easy to create test cases

Test Cases can be grouped and prioritised more easily

Supports Parameterization

Supports data driven testing using data providers

Generates Html reports

* It generates the report in a proper format, which includes the following information:
  + Number of test cases executed.
  + Number of test cases passed.
  + Number of test cases failed.
  + Number of test cases skipped

Parallel test execution is possible

Readily supports various tools like build tools and plugins like eclipse ide build tools like ant maven etc

Using TestNg we can create test cases and group tcs and prioritise test cases and execute and generate test reports

\*\*Common configurations for all team members that we use maven

Install TestNG and program

In eclipse go to Help menu

Install new software

Click add

Enter Name as TestNg

Enter Url as "http://beust.com/eclipse"

Select TestNg

accept the agreement and license agreement

Check the installation

GO to Help Menu

Installation

details -It will show details

You can also Uninstall testNg

Select TestNG and Uninstall

Write a testNg test case?

simple manual test case

1.Launch gmail page and verify the title of the page

Test steps

1.Launch browser

2.Navigate to gmail.com

3.Capture the PageTitle and compare with expected

excepted=gmail

Actual after execution

4.Status (after execution only)

\*\* No main method in testNG Programs

TestNg Test Case

1.Create class as Sample

public class Sample{

@Test

public void verifyTitle(){

Webdriver driver =new FirefoxDriver();

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

String pageTitle= driver.getTitle();

Assert.assertEquals(pageTitle,"Gmail")

}

}

Note:

1.Main method is not used for testNg program

2.TestNg program contains only methods that contains @Test Annotations.

3.If we dont write @test annotations then the methods are not going to be executed.

Create Mutliple testcases

public class Sample{

public void testA(){

assert.Assertequals("Gmail",Gmail);

}

public void testB(){

assert.Assertequals("Yahoo",Yahool);

}

Test (priority=1,enabled=false)

public void testC() {

assert.Assertequals("Yahoo",Yahool);

}

TestNG testcases can be executed in alphabetical order but you can change the order of testcases.

If you want to control the test execution process then use priority attribute.

enabled=false means it does not execute the methods

Depends on Methods

@Test (dependsOnMethods ={"login"})

### 

### 

### **TestNg Dependencies**

Sometimes we need to run test methods in a certain order, As we need to run a test method always run after some test method. TestNg provides two ways to accomplish this either with annotation or XML suite.

**Dependencies with annotations.**

dependsOnMethods or dependsOnGroup attributes are used on @Test annotation to accomplish execution of dependent test.

There are two kinds of dependencies:

**Hard dependencies:** All the methods you depend on must have run and succeeded for you to run. If at least one failure occurred in your dependencies, you will not be invoked and marked as a SKIP in the report.

**Soft dependencies:** You will always be run after the methods you depend on, even if some of them have failed. This is useful when you just want to make sure that your test methods are run in a certain order but their success doesn't really depend on the success of others. A soft dependency is obtained by adding "alwaysRun=true" in your @Test annotation.

**package com.test;**

**import org.junit.Assert;**

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

**public class DependenciesTest {**

**@Test**

**public void testMethod1() {**

**System.out.println("Test Method1 executed");**

**}**

**@Test(dependsOnMethods={"testMethod1"})**

**public void testMethod2() {**

**System.out.println("Test Method2 executed");**

**Assert.assertTrue(false);**

**}**

**@Test(dependsOnMethods={"testMethod2"})**

**public void testMethod3() {**

**System.out.println("Test Method3 executed");**

**}**

**@Test(dependsOnMethods={"testMethod3"},alwaysRun = true)**

**public void testMethod4() {**

**System.out.println("Test Method4 always executed");**

**}**

**}**

When above class executed first test method "testMethod1" executed after that "testMethod2" executed and "testMethod3" not executed due to "testMethod2" failure at last "testMethod4" executed as it marked as "alwaysRun = true"

**Output:**

**Test Method1 executed**

**Test Method2 executed**

**Test Method4 always executed**

**PASSED: testMethod1**

**PASSED: testMethod4**

**FAILED: testMethod2**

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

**Answer: There are three sections of annotation in TestNG:**

(i) **Precondition annotations**: These are the TestNG annotations that are executed before the test.

@BeforeSuite, @BeforeClass, @BeforeTest, @BeforeMethod are the precondition annotations.

(ii) **Test annotation**: This is the annotation which is only mentioned before the test case (Before the method written to execute the test case)

@Test is the test annotation

(iii) **Postcondition annotation**: These are the annotations that are executed after the test case. (After the method is written to execute the test case)

@AfterSuite, @AfterClass, @AfterTest, @AfterMethod are the postcondition annotations

**What is the sequence of execution of the annotations in TestNG?**

**Answer: The Sequence of execution of the annotations is as follows:**

**@BeforeSuite**

**@BeforeTest**

**@BeforeClass**

**@BeforeMethod**

**@Test**

**@AfterMethod**

**@AfterClass**

**@Aftertest**

**@AfterSuite**

**What are the advantages of TestNG?**

**Answer: The advantages of TestNG are as follows:**

* It is an **open-source framework**, hence it is **easy to configure.**
* Using TestNG we can systematically create the test cases.
* It gives lots of **annotations** which in turn makes the **test case creation easy.**
* Using TestNG, **priorities** of the tests and the **sequence of execution** can be defined.
* **Grouping** is possible using TestNG.
* It generates **HTML reports** (Selenium Webdriver cannot generate the test reports alone, it helps SW to achieve this).
* **Data parameterization** is possible using TestNG.

**How to share the project report using TestNG?**

**Answer: There are a few ways to do so:**

**(i)** After the execution of the TestNG class, there is one tab called “Result of running class “ which is generated next to the console.

We can copy this and share it.

**(ii)** After the execution of the TestNG class,

* Right-click on the project name and refresh
* Click on the “Test-output” folder
* Right-click on the “index.html” file and select properties
* Copy the link next to “Location”

We can share this link to see the basic HTML test report which is generated by TestNG.

This is the file that gets generated on your machine automatically after the execution of the class using TestNG.

**What is InvocationCount in TestNG?**

Answer: If we want to execute a test case “**n” number of times**, then we can use the invocationCount attribute as shown in the below example.

Example:

@Test(invocationCount=8)

Public void print()

{

}

**What is timeOut in TestNG?**

**Answer:** If any method in the script takes a long time to execute, then we can terminate that method using “timeout” in TestNG.

@Test(timeout = 5000)

In this case, the method will get terminated in 5000 ms (5 seconds) and the test case is marked as “Failed”.

**What is the need to create a testng.xml file?**

**Answer:** When we test a project using Selenium Webdriver, it has a lot of classes on it. We cannot choose these classes one by one and put them for automation. Hence we need to create a suite so that all the classes run in a single test suite.

We can achieve this by creating a testing.xml file.

**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 testing.xml)

Then, Click on finish.

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>