TestNG – Testing Framework

TestNG – Testing New Generation framework(unit test framework) or also called as

TDD Framework – Test Driven Development -- this help to define the test cases along with the development in agile methodology

Purpose : To design test cases in a systematic way.

* Open Source
* Available in the form of jar files
* Also called as java unit testing framework i.e it is used only for java. Therefore, JUnit and TestNG are used only for java. Each language has got separate framework for testing like PHP has got PHP Unit, for .NET and C# M Unit is available.

Advantages :

* To design the test cases systematicallly.
* To generate very good HTML reports.
* It gives lot of different Annotations.
* We can define the priorities or the sequence of the test cases.
* Dependency
* Grouping
* Data Provider feature

Installation of TestNG

* Download TestNG plugin for eclipse from google.
* Copy this url(<http://beust.com/eclipse/>) to the help tab of the eclipse Help 🡪 Install New Software 🡪Work with
* Select TestNG and click Next
* Three options will be displayed. Click on Next
* Accept the license agreement and click on Finish
* It will start installing TestNG.
* Add the jars files of TestNG by using build path. If the TestNG is installed properly then go to build path -> Configure build path -> Libraries -> Add Library -> Select TestNG -> Apply & Close

Another way to install TestNG without downloading it.

* First create a Maven project
* Add dependency for the TestNG in the pom.xml file of the Maven project from the (<http://testng.org/doc/download.html>) google.
* Similarly add dependency for the selenium in the pom.xml of the Maven project from the (<https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java/3.14.0>) google.
* Both testing and selenium jars are added to the project(builded) by adding the dependencies to the pom.xml. No need of external jars to be added for selenium and TestNG
* You need to install the TestNG by the above procedure.

TestNG class doesn’t require any main method. It will automatically execute the class.

Annotations in TestNG(annotations must start with ‘@’)

Each annotation must be associated with a method.

Pre-Conditions – starting with @Before

* @BeforeSuite – First preference
* @BeforeClass - Third preference
* @BeforeMethod - Fourth preference – This is executed before the test cases
* @BeforeTest - Second preference

Test Cases – starting with @Test

* @Test - All the test cases which are executed

Post Conditions - starting with @After

* @AfterMethod
* @AterTest
* @AfterClass
* @AfterSuite

@BeforeMethod and the @AfterMethod are run before and after each @Test. Each time these two methods are run before and after each test.

Always give ‘Test’ in the name of the class which you need to test Eg : “GoogleTest”

If you want to know whether an element is displayed or not you can use .isDisplayed() method (Eg : boolean b = driver.findElement(By.xpath(“”)).isDisplayed().This method returns a boolean value either true or false.

The order of the tests is not defined it picks randomly if we don’t mention priority to the test cases. But before and after annotated methods will be executed before and after each test. To follow the particular order of test cases you can use keyword “priority” Eg : @Test(priority = 1) Now based on the priority the test cases will be chosen.

To implement TestNG framework, the following steps must be followed

* First use the @BeforeMethod annotation which generally consists of pre requisites i.e system.setproperty or driver initialization.
* At the end @AfterMethod must be used.
* You can write any number of testCases and annotate them with @Test.
* If you want to know the number of test cases, you need to count all the tests with are annotated with @Test
* If you refresh the project (Right click on the project name and click refresh). Once the project is refreshed a new folder is generated in the project with the name ‘test-output’. It consists of the output of the test cases after execeution.

**How do you execute test cases in a sequencing manner or how do you define the sequencing of test cases?** – Interview question

To execute the test cases in a sequence manner we have a keyword “priority”. Based on the priority given the test cases executes sequentially. E.g : @Test(priority = 1)

**Test-output folder & How to generate the HTML report (index.html)**

Once all the test cases are written and executed, right click on the project and refresh the whole project not the page. We observe that a folder is generated automatically with the name ‘test-output’ which consists of number of files.

It contains the output of your test results after the execution.

To get the HTML report of your output follow the below steps.

* Once the test-output is generated after refreshing the project, right click on the index.html file in that test-output folder and click ‘Properties’.
* In the Resource tab of Properties you can find the location where the test reports are generated.
* Copy that location and open it in google chrome.
* The HTML report of your tests is generated which can be forwarded to manager if need for proof.

The left panel of your test report consists of various links which gives info about the passed and failed test cases. The chronological view gives info the order of the execution of the test cases. Times gives info about the time utilized for execution of each test case individually.

* TestNG is preferred as it generates html reports (index.html) which is present in test-output folder (this is obtained when we refresh the whole project after executing the test.)

**Grouping of testCases:**

Grouping of the test cases can be done by using ‘groups’.

Suppose I have 100 test cases and out of them 20 are related to Search then you can group them by using groups=’Search’. Similarly if there are 10 test cases on registration then we can group them by using groups=’Register’ etc. E.g : @Test(priority=1,groups=’Search’)

So after the execution check the html report as mentioned in above paragraph. One the html report is opened on the left side panel you will be able to see total groups which gives info about the test cases that are grouped.

**TestNg Features(class : TestNgFeatures)**

**Dependency of the Test Cases:**

Some test cases are dependent on other. Suppose, consider login test and home page test, if the login fails then there is no point of execution of home page, it is waste of time. In order to save the time we need to stop the execution of the test cases which are dependent on login test.

To achieve this we need to define the dependency of the test cases.

* We know that the home page test is totally dependent on the login test so use ‘dependsOnMethods’ keyword for the home page test to mention that it depends on the login test.
* E.g : @Test(dependsOnMethods = “LoginTest”)

public void HomePageTest () {

}

This implies if the LoginTest() is passed then only it considers or executes the HomePageTest(). If the LoginTest() fails the HomePageTest () is not considered or not executed or ignored or skipped automatically to save the time.

**Invocation Count :**

If you want to execute any test case(method) for multiple times then we can use invocation count keyword .If you annotate the test case with @Test(invocationCount=10) it implies that the test case is executed ten times immediately.