**TestNG**

Is an automation testing framework in which NG stands for "Next Generation". TestNG is inspired from JUnit which uses the annotations @. Used by QA’s for test-driven development.

TestNG is designed to cover all categories of tests: **Unit**, **Integration**, **functional**, **end to end** etc.

[ Note: Annotations are nothing but a piece of instructions for the compiler that we apply for Classes, Methods or Variables in Java code.]

Advantages of TestNG:

* Manages test suites and test cases.
* Helps in prioritizing of tests.
* Helps in Grouping of tests.
* Parallel execution.
* Reporting.
* Can be easily integrated with tools like, Maven, Jenkins etc.

[ Note: WebDriver has no native mechanism for generating reports. TestNG can generate the report in a readable format.]

**Setup TestNG Environment in Intellij.**

Step 1:

IntelliJ > New project:> java project:

Step 2:

Add TestNG dependency (if it is maven project) Or

Download jar file (it is java project) and configure project> project structure> modules> dependencies> select add and choose the TestNG jar.

**TestNG XML file:**

In TestNG Xml file play important role, using testing.xml file:

* we can execute number test cases as suites.
* We create number of suites and execute the parallelly.
* We can also pass parameters, etc

**How to create a testing.xml file.**

First add new plugin by setting>plugins add > create testing xml.

And add the <class> tag with testClassName should be add <classes> tag.

<suite> < test> <classes> <class> </classes></test></suite>

Then right click testing.xml file run with xml file.

To generate TestNG report:

We need to add 2 TestNG Listeners

Select project go to > edit configurations> under Listeners options add below.

1. EmailableReporter ( org.testng)
2. FailedReporter (org.testng)

**Running test using testing.xml file:**

Then right click on testing.xml file run with xml file. We can see reports are generated passed and failed.

**Annotations in TestNG:** annotations are used to controls the execution of test methods.

**@Test**: becomes the- part of a test.

**@BeforeMethod**: allows a method to run before executing any of the @test methods/multiple times

**@AfterMethod**: allows to take off after all of the @test finished execution/multiple times

**@BeforeClass**: gets executed once before the first @test method of a current class/once

**@AfterClass**: gets executed once after finishing all @test methods in of a current class/once

**@BeforeTest**: run before any test method belonging to the classes inside the <test> tag in testing.xml.

**@AfterTest**: Halts from execution till all the test methods finish their execution in<test> tag.

**@BeforeSuite**: any such method will get called before any suites runs from the test.

**@AfterSuite**: any such method will get called before any suites runs from the test.

**@Factory**: use to execute any specific group of test cases with different levels. returns <Object []>.

**@BeforeGroups**: it sets up the method to run before first @test method belongs to a group involved.

**@AfterGroups**: it sets up methods to run after execution belongs to group involved.

**@Parameters: used for passing the parameters to the test methods.**

**@DataProvider: it marks a method as data source of the test. Returns value as <Object [] []>.**

**@Listeners:** use them with the test classes for the logging function.

[ Note: Package itself is TestSuite]

**Prioritizing Tests and Disabling Tests in TestNG:**

[ By default, methods annotated by @Test are executed alphabetically.]

[ If you want the methods to be executed in a different order, use the parameter.]

"priority". **Parameters are keywords that modify the annotation's function**.

**@Test( priority = 1)**

If we want to ignore the test case, we have to use enabled = false as parameter.

**@Test(enabled=false)**

**Dependency Tests and AlwaysRun property in TestNG:**

If we have any Test methods depends upon another method to be executed, then we have to pass

**@Test(dependsOnMethod={“test1”}**

If we want to forcefully execute the Test method, we have to pass parameter AlwaysRun=ture.

**@Test(dependsOnMethod={“tes1” , AlwaysRun=true}**

**Grouping Tests in TestNG:**

**Grouping of the Test cases is very important at the time of execution. Use parameter groups.**

**@Test(groups={“sanity”) and configure in testing.xml file by using <groups> tag and <run>tag after <test> tag.**

**Example of testing.xml file:**

**<suite><test><classes> <groups> <run>< include name=”sanity”/> </run></groups>**

**<class name=”TestClassname”/>**

**</classes></test></suite>**

**[ note: we can also use exclude group by using <exclude > tag in testing.xml].**

**Assertions in TestNG:** assertion are verification point, Assert class provides assetTrue() , assertFalse(), assertEquals().

Assert.assertTrue() – returns boolean value based upon the condition.

Assert.assertFalse()- returns Boolean – opposite of Assert.assertFalse().

Assert.assertEquals() – check for expected and actual value.

**Parameters in TestNG:**

Parameterization is plays important role in automation testing, we do not have to hard code the values, we can use TestNG parameters. By using testing.xml file we can pass parameters to test methods.

**@Parameter(username, password)** after @Test and define test method as required parameterized.

And configure the testing.xml as below.

Select the test class and create testing.xml file for the class file amend as below and then run xml file.

**<suite>**

**<parameter name=”nameofParameter” value=”valueofParameter”/ >**

**<parameter name=”nameofParameter” value=”valueofParameter”/ >**

**<test><classes>**

**<class name=”TestClassname”/>**

**</classes></test></suite>**

[Note: we can pass at different levels class level test level or suite level.]

**DataProvider in TestNG:**

It is similar to Parameters, we can use DataProvider are used to data driven development and we can pass multiple combinations of data to test methods.

A method generates and provides the data to another method. The data provider method returns an array of Object (object [] [] ).

@DataProvider(name=”logindata”) - simple define before the method, it is just method not test method. To use that data we need to use @Test(dataProvider =”dataproviderMethodName”).

**Parallel Tests in TestNG:**

* We can Test methods in parallel.
* We can Test classes in parallel.
* We can Test Suite in parallel.

In order to execute parallelly, we have to configure testing.xml file.

In the <suite> tag level we have to add parallel=”methods” thread-count = “numberOfThreads”>

Example:

<suite name=”suiteaname” Parallel=”methods” thread-count=”4”>

[Note: to execute to class change the parallel as “classes”]

**Listeners in TestNG:**

Basically, while automating test cases, after executing the test methods, **based on the results/status** we want **to perform some post actions**. We can perform post actions and TestNG had provided Listeners concept.

Listeners interface contains different methods, and these should be integrated with test cases. They will be automatically trigger.

[ Note: To perform we have implements ITestListerner to test class and pass as parameters in test methods.

The popular Listeners the most popular is **ITestListener**.

Contains methods

onStart(ITestConext arg)

onFinish(ITestContext args0) and etc.

To use Listener, we have to use @Listeners(listeners.CustomListerners.class) at test class level. And configure in testing.xml by adding a <listener> tag in suite level.

Example

<listeners>

<listener class-name=”listener.CustomListerners”>

</listeners>