TestNG

It is an open source automated testing framework; where **NG** of Test**NG** means **N**ext **G**eneration. TestNG is similar to JUnit but it is much more powerful than JUnit but still it’s inspired by JUnit. It is designed to be better than JUnit, especially when testing integrated classes. Pay special thanks to *Cedric Beust who is the creator of TestNG*.

## **Benefits of TestNG**

There are number of benefits but from Selenium perspective, major advantages of TestNG are :

1. It gives the ability to produce ***HTML Reports*** of execution
2. ***Annotations*** made testers life easy
3. Test cases can be ***Grouped & Prioritized*** more easily
4. ***Parallel*** testing is possible
5. Generates ***Logs***
6. Data ***Parameterization*** is possible

**@BeforeSuite**: The annotated method will be run before all tests in this suite have run.

**@AfterSuite**: The annotated method will be run after all tests in this suite have run.

**@BeforeTest**: The annotated method will be run before any test method belonging to the classes inside the tag is run.

**@AfterTest**: The annotated method will be run after all the test methods belonging to the classes inside the tag have run.

**@BeforeGroups**: The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.

**@AfterGroups**: The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.

**@BeforeClass**: The annotated method will be run before the first test method in the current class is invoked.

**@AfterClass**: The annotated method will be run after all the test methods in the current class have been run.

**@BeforeMethod**: The annotated method will be run before each test method.

**@AfterMethod**: The annotated method will be run after each test method.

**@Test**: The annotated method is a part of a test case.

**@BeforeSuite**

**@BeforeTest**

**@BeforeGroups /BeforeClass**

**@BeforeMethod**

**@Test**

**@AfterMethod**

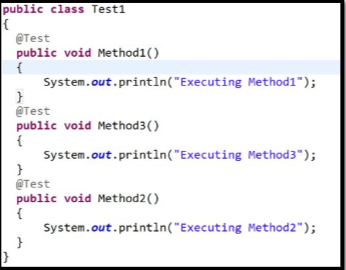
**@AfterGroups/AfterClass**

**@AfterTest**

**@AfterSuite**

TestNG works with Annotations and annotation can be represented by @ symbol

@Test- is this the main annotation from where TestRunner will start execution.  
In other words, you can say @Test in entry point



Output:

Executing Method1

Executing Method2

Executing Method3

**public class Sequencing {**

**@Test**

**public void testCase1() {**

**System.out.println("This is the Test Case 1");**

**}**

**@Test**

**public void testCase2() {**

**System.out.println("This is the Test Case 2");**

**}**

**@BeforeMethod**

**public void beforeMethod() {**

**System.out.println("This will execute before every Method");**

**}**

**@AfterMethod**

**public void afterMethod() {**

**System.out.println("This will execute after every Method");**

**}**

**@BeforeClass**

**public void beforeClass() {**

**System.out.println("This will execute before the Class");**

**}**

**@AfterClass**

**public void afterClass() {**

**System.out.println("This will execute after the Class");**

**}**

**@BeforeTest**

**public void beforeTest() {**

**System.out.println("This will execute before the Test");**

**}**

**@AfterTest**

**public void afterTest() {**

**System.out.println("This will execute after the Test");**

**}**

**@BeforeSuite**

**public void beforeSuite() {**

**System.out.println("This will execute before the Test Suite");**

**}**

**@AfterSuite**

**public void afterSuite() {**

**System.out.println("This will execute after the Test Suite");**

**}**

**}**

**Setting Priorities for TestNG:**

**By default TestNG Methods will be executed in the alphabetical order of the method names.**

**If we want to redefine the order of execution of the methods, then we have to set the priorities for each method.**

**Example:**

**@Test(priority = 1)**

**publicvoidLogin()**

**{**

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

**}**

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

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

**public class MultipleTest {**

**public WebDriver driver;**

**@Test(priority = 0)**

**public void One() {**

**System.out.println("This is the Test Case number One");**

**}**

**@Test(priority = 1)**

**public void Two() {**

**System.out.println("This is the Test Case number Two");**

**}**

**@Test(priority = 2)**

**public void Three() {**

**System.out.println("This is the Test Case number Three");**

**}**

**@Test(priority = 3)**

**public void Four() {**

**System.out.println("This is the Test Case number Four");**

**}**

**}**

‘**Groups**‘ is one more annotation of TestNG which can be used in the execution of multiple tests. Let’s say you have hundred tests of class vehicle and in it ten method of car, ten method of scooter and so on. You probably like to run all the scooter tests together in a batch. And you want all to be in a single test suite. With the help of grouping you can easily overcome this situation.

package automationFramework;

import org.testng.annotations.Test;

public class Grouping {

@Test (groups = { "Car" })

public void Car1() {

System.out.println("Batch Car - Test car 1");

}

@Test (groups = { "Car" })

public void Car2() {

System.out.println("Batch Car - Test car 2");

}

@Test (groups = { "Scooter" })

public void Scooter1() {

System.out.println("Batch Scooter - Test scooter 1");

}

@Test (groups = { "Scooter" })

public void Scooter2() {

System.out.println("Batch Scooter - Test scooter 2");

}

@Test (groups = { "Car", "Sedan Car" })

public void Sedan1() {

System.out.println("Batch Sedan Car - Test sedan 1");

}

}

**Create a testng xml**

<suite name="Suite">

    <test name="Practice Grouping">

        <groups>

    <run>

<include name="Car" />

    </run>

</groups>

<classes>

    <class name="automationFramework.Grouping" />

</classes>

    </test>

 </suite>

We have just call the group ‘Car’ from the xml and it also executed the test for Sedan Car, as we have mentioned the ‘Car’ as well while declaring the group of Sedan Car.

Clubbing of groups is also possible, take a look at the below xml:

<suite name="Suite">

   <test name="Practice Grouping">

      <groups>

         <define name="All">

   <include name="Car"/>

   <include name="Scooter"/>

</define>

<run>

   <include name="All"/>

</run>

   </groups>

<classes>

      <class name="automationFramework.Grouping" />

</classes>

   </test>

</suite>

You can see that we have created a new Group with the name ‘All’ and include other groups in it.

## **Dependent Test**

Sometimes, you may need to invoke methods in a Test case in a particular order or you want to share some data and state between methods. This kind of dependency is supported by TestNG as it supports the declaration of explicit dependencies between test methods.

TestNG allows you to specify dependencies either with:

* Using attributes *dependsOnMethods* in @Test annotations OR
* Using attributes *dependsOnGroups* in @Test annotations.

package automationFramework;

import org.testng.annotations.Test;

public class Dependent {

@Test (dependsOnMethods = { "OpenBrowser" })

public void SignIn() {

System.out.println("This will execute second (SignIn)");

}

@Test

public void OpenBrowser() {

System.out.println("This will execute first (Open Browser)");

}

@Test (dependsOnMethods = { "SignIn" })

public void LogOut() {

System.out.println("This will execute third (Log Out)");

}

### Disable Selenium Test cases

import org.testng.annotations.Test;

public class TestEnableTC {

@Test

public void testLoginApp(){

System.out.println("User is able to login successfully");

}

@Test(enabled=false)

public void testRegisteruser(){

System.out.println("User is able to register successfully");

}

@Test

public void testLogoutApp(){

System.out.println("User is able to logout successfully");

}

}

# Cross Browser Testing And Parameterization:

Cross browser, testing refers to testing the application in multiple browsers like IE, Chrome, Firefox so that we can test our application effectively.[IE](http://learn-automation.com/challenges-with-ie-browser-in-selenium-webdriver/), [Chrome](http://learn-automation.com/launch-chrome-browser-using-selenium-webdriver/), [Firefox](http://learn-automation.com/use-firefox-selenium-using-geckodriver-selenium-3/) so that we can test our application effectively.

Cross browser, testing is very important concept in Automation because here the actual automation comes into the picture.

Example- Suppose if you have 20 test cases that you have to execute manually, so it is not a big deal right we can execute in 1 day or 2 days. However, if the same test cases you have to execute in five browsers it means 100 test cases then probably you will take one week or more than one week to do the same and it will be quite boring as well.

If you automate these 20 test cases and run them then it will not take more than one or two hour depends on your test case complexity.

package TestNGPackage;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.ie.InternetExplorerDriver;

import org.testng.annotations.Parameters;

import org.testng.annotations.Test;

public class Multi {

@Test

// Here this parameters we will take from testng.xml

@Parameters("Browser")

public void test1(String browser) {

if(browser.equalsIgnoreCase("FF")){

WebDriver driver=new FirefoxDriver();

driver.manage().window().maximize();

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

driver.quit();

}

else if(browser.equalsIgnoreCase("Chrome")){

System.setProperty("webdriver.chrome.driver", "D:\\chromedriver.exe");

WebDriver driver=new ChromeDriver();

driver.manage().window().maximize();

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

driver.quit();

}

}

}

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

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

<suite name=*"Suite"* parallel=*"tests"* thread-count=*"2"*>

<test name=*"Test"*>

<parameter name=*"Browser"* value=*"FF"* />

<classes>

<class name=*"TestNGPackage.Multi"*/>

</classes>

</test>

<test name=*"Test1"*>

<parameter name=*"Browser"* value=*"Chrome"* />

<classes>

<class name=*"TestNGPackage.Multi"*/>

</classes>

</test>

</suite>