TESTNG :

* What is TestNG?

1. TestNG stands for "**Testing Next Generation**".
2. It is an` automation testing framework used for java programming language developed by Credric beust, and
3. it comes after the inspiration from the JUnit framework.
4. TestNG consists of all the features of JUnit framework
5. But also contains some more additional features that make TestNG more powerful.

**Soft Assertion**: When an assertion fails, TestNG logs the error but continues executing the remaining code in the test.

**Hard Assertion**: When an assertion fails, TestNG immediately stops executing further code in the test and throws an exception.

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

We can also pass the value to test methods at runtime, we can achieve this by sending parameter values through the testng.xml file. We can use the **@Parameter** annotation:

<suite name="Suite">

<test name="It Company">

<parameter name="text" value="javatpoint"/>

<classes>

<**class** name="com.javatpoint.Web"></**class**>

</classes>

</test>

</suite> <!-- Suite -->

* How can we disable the test case from running?

We can disable the test case from running by using the enabled attribute. We can assign the false value to the enabled attribute, in this way we can disable the test case from running.

public **class** Test\_cases

{

@Test(enabled=**false**)

**public** **void** testcase1()

{

System.out.println("testcase1");

}

@Test

**public** **void** testcase2()

{

System.out.println("testcase2");

}

}

* What is the use of @Listener annotation in TestNG?

1. TestNG provides different kinds of listeners which can perform different actions whenever the event is triggered.
2. The most widely used listener in TestNG is ITestListener interface.
3. The ITestListener interface contains methods such as onTestSuccess, onTestfailure, onTestSkipped, etc.

**Following are the scenarios that can be made:**

* If the test case is failed, then what action should be performed by the listener.
* If the test case is passed, then what action should be performed by the listener.
* If the test case is skipped, then what action should be performed by the listener.
* 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.

* What are the advantages of TestNG?

**The following are the advantages of TestNG are:**

* 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
* Multiple test cases can be grouped easily by converting them into a testng.xml file, in which you can set the priority of each test case that determines which test case should be executed first.
* With the help of TestNG, you can execute the multiple test cases on multiple browsers known as cross-browser testing.
* The TestNG framework can be easily integrated with other tools such as Maven. Jenkins, etc.
* Annotations used in a TestNG framework are easily understandable such as @BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest.
* WebDriver does not generate the reports while TestNG generates the reports in a readable format.
* TestNG simplifies the way the test cases are coded.
* We do not have to write the static main method.
* The sequence of actions is maintained by the annotations only.
* TestNG allows you to execute the test cases separately.
* For example, if you have six test cases, then one method is written for each test case. When we run the program, five methods are executed successfully, and the sixth method is failed. To remove the error, we need to run only the sixth method, and this can be possible only through TestNG. Because TestNG generates testng-failed.xml file in the test output folder, we will run only this xml file to execute the failed test case.
* How to run the test script in TestNG?

You can run the test script in TestNG by clicking right click on the TestNG class, click on "Run As" and then select "TestNG test".

* What are the annotations used in the TestNG?

**The following are the annotations used in the TestNG are:**

* **Precondition annotations**  
  Precondition annotations are executed before the execution of test methods The Precondition annotations are @BeforeSuite, @BeforeClass, @BeforeTest, @BeforeMethod.
* **Test annotation**  
  Test annotation is specified before the definition of the test method. It is specified as @Test.
* **Postcondition annotations**  
  The postcondition annotations are executed after the execution of all the test methods. The postcondition annotation can be @AfterSuite, @AfterClass, @AfterTest, @AfterMethod.
* What is the sequence of execution of all the annotations in TestNG?

**The sequence of execution of all the annotations in TestNG is given below:**

* @BeforeSuite
* @BeforeTest
* @BeforeClass
* @BeforeMethod
* @Test
* @AfterSuite
* @AfterTest
* @AfterClass
* @AfterMethod
* How to set the priorities in TestNG?

1. If we do not prioritize the test methods, then the test methods are selected alphabetically and executed.
2. If we want the test methods to be executed in the sequence we want, then we need to provide the priority along with the @Test annotation.

public **class** Test\_methods

{

@Test(priority=2)

**public** **void** test1()

{

System.out.println("Test1");

}

@Test(priority=1)

**public** **void** test2()

{

System.out.print("Test2");

}

* Define grouping in TestNG?

1. The group is an attribute in TestNG that allows you to execute the multiple test cases.
2. For example, if we have 100 test cases of it-department and 10 test cases of hr-department, and if you want to run all the test cases of it-department together in a single suite, this can be possible only through the grouping.

public **class** Test\_methods

{

@Test(groups="it\_department")

**public** **void** java()

{

System.out.println("I am a java developer");

}

@Test(groups="it\_department")

**public** **void** dot\_net()

{

System.out.println("I am a .Net developer");

}

@Test(groups="it\_department")

**public** **void** tester()

{

System.out.println("I am a software tester");

}

@Test (groups="hr")

**public** **void** hr()

{

System.out.print("I am hr");

}

}

**testng.xml**

1. ?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3. <suite name="Suite">
4. <test name="It Company">
5. <groups>
6. <run>
7. <include name="it\_department"/>
8. </run>
9. </groups>
10. <classes>
11. <**class** name="com.javatpoint.Test\_methods"></**class**>
12. </classes>
13. </test>
14. </suite> <!-- Suite -->

* What is dependency in TestNG?

When we want to run the test cases in a specific order, then we use the concept of dependency in TestNG.

**Two types of dependency attributes used in TestNG:**

1. **dependsOnMethods**  
   The dependsOnMethods attribute tells the TestNG on which methods this test will be dependent on, so that those methods will be executed before this test method.

public **class** Login

{

 @Test

**public** **void** login()

 {

     System.out.println("Login page");

 }

 @Test(dependsOnMethods="login")

**public** **void** home()

 {

     System.out.println("Home page");

 }

}

1. **dependsOnGroups**  
   It is similar to the dependsOnMethods attribute. It allows the test methods to depend on the group of test methods. It executes the group of test methods before the dependent test method.

public **class** Test\_cases

{

 @Test(groups="test")

**public** **void** testcase1()

 {

     System.out.println("testcase1");

 }

 @Test(groups="test")

**public** **void** testcase2()

 {

     System.out.println("testcase2");

 }

 @Test(dependsOnGroups="test")

**public** **void** testcase3()

 {

     System.out.println("testcase3");

 }

}

* What is timeOut in TestNG?

1. While running test cases, there can be a case when some test cases take much more time than expected. In such a case, we can mark the test case as a failed test case by using timeOut.
2. TimeOut in TestNG allows you to configure the time period to wait for a test to get completely executed. It can be configured in two levels:

**At the suit level:** It will be available to all the test methods.

**At each method level:** It will be available to a particular test method.

1. The timeOut attribute can be specified as shown below:
2. **@Test( timeOut = 700**)
3. The above @Test annotation tells that the test method will be given 700 ms to complete its execution otherwise it will be marked as a failed test case.

* What is invocationCount in TestNG?

An invocationCount in TestNG is the number of times that we want to execute the same test.

**package** com.javatpoint;

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

**public** **class** Test\_cases

{

 @Test(invocationCount=5)

**public** **void** testcase1()

 {

     System.out.println("testcase1");

 }

}

* What is the importance of testng.xml file?

**The testng.xml file is important because of the following reasons:**

* It defines the order of the execution of all the test cases.
* It allows you to group the test cases and can be executed as per the requirements.
* It executes the selected test cases.
* In TestNG, listeners can be implemented at the suite level.
* It allows you to integrate the TestNG framework with tools such as Jenkins.
* What is the difference between @Factory and @DataProvider annotation?

1. **@DataProvider:** It is annotation used by TestNG to execute the test method multiple numbers of times based on the data provided by the Data Provider.
2. **@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.