TestNG Interview Questions

1. What is TestNG?

1. TestNG is a Testing framework that overcomes the limitations of another popular testing framework called JUnit.
2. The "NG" means "Next Generation". Most Selenium users use this more than Junit because of its advantages. There are so many features of TestNG.

2. What are the advantages of TestNG?

1. TestNG provides parallel execution of test methods
2. It allows to define dependency of one test method over other method
3. It allows to assign priority to test methods
4. It allows grouping of test methods into test groups
5. It has support for parameterizing test cases using @Parameters annotation
6. It allows data driven testing using @Data Provider annotation
7. It has different assertions that helps in checking the expected and actual results
8. Detailed (HTML) reports

3. What are the annotations available in TestNG?

@BeforeTest, @AfterTest, @BeforeClass, @AfterClass, @BeforeMethod, @AfterMethod, @BeforeSuite, @AfterSuite, @BeforeGroups, @AfterGroups, @Test.

4. Can you arrange the below testng.xml tags from parent to child?

<test>
<suite>
<class>
<methods>
<classes>

Correct order:

<suite>
<test>
<classes>
<class>
<methods>

5. How to create and run *testng.xml?*

1. We can create testng.xml based on existing TestNG class, we use option in eclipse convert to TestNG
2. We need to create testng.xml file to create and handle multiple test classes.
3. We do configure our test run, set test dependency, include, or exclude any test, method, class, or package and set priority etc in the xml file.
4. Once testng.xml created we use option run as TestNG Suite or we can execute from Maven.

6. What is the importance of testng.xml file?

In a Selenium TestNG project, we use testng.xml file to configure the complete test suite in a single file. Some of the features are as follows.

1. testng.xml file allows to include or exclude the execution of test methods and test groups
2. It allows to pass parameters to the test cases
3. Allows to add group dependencies
4. Allows to add priorities to the test cases
5. Allows to configure parallel execution of test cases
6. Allows to parameterize the test cases

7. How to pass parameter through testng.xml file to a test case?

For Example, we have TestNG class ParameterizedTest as shown below

Syntax:

```
public class ParameterizedTest {
    @Test
    @Parameters("browser")
    public void invokeBrowser(String browser) {
        if (browser.equals("firefox")) {
            System.out.println("Open Firefox Driver");
        } else if (browser.equals("chrome")) {
            System.out.println("Open Chrome Driver");
        }
    }
}
```

Now we will provide value to for invokeBrowser method from testng.xml, because it contains annotation @Parameters("browser") in ParameterizedTest class.

Syntax:

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="softwaretestingmaterial">
    <test name="testngTest">
        <parameter name="browser" value="firefox"/>
        <classes>
            <class name="com.atos.wlp.ParameterizedTest" />
        </classes>
    </test>
</suite>
```

8. What is TestNG Assert and list out common TestNG Assertions?

TestNG Asserts help us to verify the condition of the test in the middle of the test run.

Based on the TestNG Assertions, we will consider a successful test only if it is completed the test run without throwing any exception.

Some of the common assertions:

1. assertEquals (String actual, String expected)
2. assertEquals (String actual, String expected, String message)
3. assertEquals (boolean actual, boolean expected)
4. assertTrue (condition)
5. assertTrue (condition, message)
6. assertFalse (condition)
7. assertFalse (condition, message)

9. What is Soft Assert in TestNG?

1. Soft Assert collects errors during @Test.
2. Soft Assert does not throw an exception when an assert fails and would continue with the next step after the assert statement.
3. If there is any exception and you want to throw it then you need to use assertAll() method as a last statement in the @Test and test suite again continue with next @Test as it is.

10. What is Hard Assert in TestNG?

Hard Assert throws an Assert Exception immediately when an assert statement fails and test suite continues with next @Test

11. What is exception test in TestNG?

1. TestNG gives an option for tracing the Exception handling of code.
2. You can verify whether a code throws the expected exception or not.
3. The expected exception to validate while running the test case is mentioned using the expectedExceptions attribute value along with @Test annotation.

Syntax:

```
@Test(expectedExceptions = ArithmeticException.class)
public void testException() {
    System.out.println("SoftwareTestingMaterial.com");
    int i = 1 / 0;
}
```

12. How to set test case priority in TestNG?

We use priority attribute to the @Test annotations. In case priority is not set then the test scripts execute in alphabetical order.

Syntax:

```
public class Demo {
    @Test(priority=0)
    public void bookingTicket() {
        System.out.println("Test Case 1");
    }

    @Test(priority=1)
    public void additonalInformation() {
        System.out.println("Test Case 2");
    }
}
```

```
C:\Users\a589240\AppData\Local\Temp\testng-eclips

Test Case 1
Test Case 2
PASSED: bookingTicket
PASSED: additonalInformation

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================
```

13. What is Parameterized testing in TestNG?

Parameterized tests allow developers to run the same test over and over again using different values. There are two ways to set these parameters:

1. using testng.xml
2. using Data Providers

14. How can we create data driven framework using TestNG?

By using @DataProvider annotation, we can create a Data Driven Framework.

Syntax:

```
public class DataProviderClass {

    // This method takes data as input parameters. The attribute dataprovider is mapped to "getData"
    @Test (dataProvider="getData")
    // Number of columns should match the number of input parameters
    public void loginTest(String Uid, String Pwd, int age){
        System.out.println("UserName is "+ Uid);
        System.out.println("Password is "+ Pwd);
        System.out.println("age is "+ age);
    }

    //If the name is not supplied, the data provider's name automatically defaults to the method's name.
    //A data provider returns an array of objects.
    @DataProvider(name="getData")
    public Object[][] getData(){
    //Object [][] data = new Object [rowCount][colCount means number of parameters values we have to pass value];
        Object [][] data = new Object [2][3];

        data [0][0] = "FirstUid";
        data [0][1] = "FirstPWD";
        data [0][2] = 20;

        data [1][0] = "SecondUid";
        data [1][1] = "SecondPWD";
        data [1][2] = 23;

        return data;
    }
```

15. How to run a group of test cases using TestNG?

We can add parameter (groups = { "smokeTest", "functionalTest" }) to @test annotation, then we will try to execute from testng.xml

Syntax: First creating groups in Test Case

```
public class GropsOneTest {
    @Test(groups = { "SmokeTest", "FunctionalTest" })
    public void loginTest() {
        System.out.println("Logged in successfully");
    }

    @Test(groups = { "FunctionalTest" })
    public void checkAcountInfo() {
        System.out.println("AcountInfo is successfully fetched");
    }

    @Test(groups = { "Regression", "FunctionalTest" })
    public void GetAcountBalance() {
        System.out.println("AcountBalance is $200");
    }
}
```

Syntax: configure the testng.xml now

```
<suite name="Suite" parallel="none">
    <test name="Test">
        <groups>
            <run>
                <include name="SmokeTest" />
                <include name="Regression" />
            </run>
        </groups>
        <classes>
            <class name="com.atos.wlp.GropsOneTest" />
            <class name="com.atos.wlp.GropsTwoTest" />
            <class name="com.atos.wlp.GropsThreeTest" />
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

16. How to create Group of Groups in TestNG?

Groups can also include other groups. These groups are called Metagroups.

Syntax:

```
<suite name="Suite" parallel="none">
    <test name="Test">
        <groups>
            <define name="all">
                <include name="SmokeTest" />
                <include name="FunctionalTest" />
            </define>
            <run>
                <include name="all" />
            </run>
        </groups>
        <classes>
            <class name="com.atos.wlp.GropsOneTest" />
            <class name="com.atos.wlp.GropsTwoTest" />
            <class name="com.atos.wlp.GropsThreeTest" />
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

17. How to run test cases in parallel using TestNG?

we can use “parallel” attribute in testng.xml to accomplish parallel test execution in TestNG

The parallel attribute of suite tag can accept four values: tests, classes, methods & instances.

Syntax:

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Suite" parallel="methods" thread-count="3">
    <test name="Test">
        <classes>
            <class name="com.atos.wlp.GropsOneTest"/>
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

18. How to exclude a particular test method from a test case execution?

By adding the exclude tag inside class>methods tag in the testng.xml

Syntax:

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Suite" parallel="none">
    <test name="Test">
        <classes>
            <class name="com.atos.wlp.GropsOneTest">
                <methods>
                    <exclude name="checkAcountInfo" />
                </methods>
            </class>
        </classes>
    </test> <!-- Test -->
```

19. How to exclude a particular test group from a test case execution?

By adding the exclude tag inside groups>run in the testng.xml

Syntax:

```
<suite name="Suite1" parallel="none">
    <test name="Test">
        <groups>
            <run>
                <exclude name="Regression" />
            </run>
        </groups>
        <classes>
            <class name="com.atos.wlp.GropsOneTest">
            </class>
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

20. How to disable a test case in TestNG ?

To disable the testcase we use the parameter enabled = false to the @Test annotation.

Syntax: @Test(enabled = false)

21. How to skip a @Test method from execution in TestNG?

1. By using throw new SkipException()
2. Once SkipException() thrown, remaining part of that test method will not be executed and control will go directly to next test method execution.

Syntax:

```
public class SkipTest {
    @Test //skiped test while RUN
    public void aSkipTest() {
        String a = "Skip Test";
        if (a.equals("Skip Test")) {
            throw new SkipException("Skipping - This is not ready for testing ");
        } else {
            System.out.println("I am in else condition");
        }
        System.out.println("I am out of the if else condition");
    }

    @Test //executed the tests while RUN
    public void nonSkipTest() {
        System.out.println("No need to skip this test");
    }
```

22. How to Ignore a test case in TestNG?

To ignore the testcase we use the parameter enabled = false to the @Test annotation.

Syntax: @Test(enabled = false)

23. How TestNG allows to state dependencies?

TestNG allows two ways to declare the dependencies.

1. Using attributes dependsOnMethods in @Test annotations.
2. Using attributes dependsOnGroups in @Test annotations

Syntax: dependsOnMethods

```
public class DependsOnMethods {
    @Test(dependsOnMethods = { "testTwo" })
    public void testOne() {
        System.out.println("executed Successfully TestOne ");
    }

    @Test
    public void testTwo() {
        System.out.println("executed Successfully TestTwo ");
    }
```

```
executed Successfully TestTwo
executed Successfully TestOne
PASSED: testTwo
PASSED: testOne

===============================
    Default test
    Tests run: 2, Failures: 0, Sk
===============================
```

Syntax: dependsOnGroups

Step01:

```
public class GroupsEx {
    @Test(groups = { "FirstGroup" })
    public void testCase1() {
        System.out.println("Test Case 1");
    }

    @Test(groups = { "SecondGroup" })
    public void testCase2() {
        System.out.println("Test Case 2");
    }
}
```

Step02:

```
<suite name="softwaretestingmaterial">
    <test name="testngTest">
        <groups>
            <dependencies>
                <group name="FirstGroup" depends-on="SecondGroup"></group>
            </dependencies>
        </groups>
        <classes>
            <class name="com.atos.wlp.GroupsEx" />
        </classes>
    </test>
```

24. What are the different ways to produce reports for TestNG results?

TestNG offers two ways to produce a report.

Listeners implement the interface org.testng.ITestListener and are notified in real time of when a test starts, passes, fails, etc…

Reporters implement the interface org.testng.IReporter and are notified when all the suites have been run by TestNG. The IReporter instance receives a list of objects that describe the entire test run.

25. What is the use of @Listener annotation in TestNG?

1. TestNG listeners are used to configure reports and logging.
2. One of the most widely used listeners in TestNG is ITestListener interface. It has methods like onTestStart, onTestSuccess, onTestFailure, onTestSkipped etc.
3. We should implement this interface creating a listener class of our own. Next, we should add the listeners annotation (@Listeners) in the Class which was created.

26. How to write regular expression in testng.xml file to search @Test methods containing “smoke” keyword?

Syntax:

```
<methods>
    <include name=".*smoke.*" />
</methods>
```

27. What is the time unit we specify in test suites and test cases?

We specify the time unit in test suites and test cases is in milliseconds.

28. List out various ways in which TestNG can be invoked?

TestNG can be invoked in the following ways

1. Using (Eclipse, IntelliJ's, NetBeans) any IDE
2. Using Maven build tool
3. From the command line

29. How to Run TestNG Using Command Prompt?

C:\Users\Admin\ws\projecName
set classpath= C:\Users\Admin\ws\projecName \bin; C:\Users\Admin\ws\projecName\lib*
java org.testng.TestNG C:\Users\Admin\ws\projecName\testng.xml

30. What is the use of @Test(invocationCount=x)?

The invocationcount attribute tells how many times TestNG should run a test method

Syntax:

```
public class InvocationCountEx {
  @Test(invocationCount= 6) //this test executed 6 times
  public void MethodOne() {
      System.out.println(" invoked method");
  }
}
```

31. What is the use of @Test(threadPoolSize=x)?

The threadPoolSize attribute tells to form a thread pool to run the test method through multiple threads.

Note: This attribute is ignored if invocationCount is not specified

32. What does the test timeout mean in TestNG?

The maximum number of milliseconds a test case should take.

```
@Test(threadPoolSize = 3, invocationCount = 10, timeOut = 10000)
public void testCase1() {
    System.out.println(" invoked method");
}
```

In this example, the function testCase1 will be invoked ten times from three different threads. Additionally, a time-out of ten seconds guarantees that none of the threads will block on this thread forever.

33. What are @Factory and @DataProvider annotation?

@Factory: A factory will execute all the test methods present inside a test class using a separate instance of the respective class with different set of data.

@DataProvider: A test method that uses DataProvider will be executed the specific methods multiple number of times based on the data provided by the DataProvider.

The test method will be executed using the same instance of the test class to which the test method belongs.