TestNG Interview Questions:

1. What is TestNG?

TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing to integration testing.

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 @DataProvider 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?

1 <test>

2 <suite>

3 <class>

4 <methods>

5 <classes>

The correct order of the TestNG tags are as follows

1 <suite>

2 <test>

3 <classes>

4 <class>

5 <methods>

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

In TestNG framework, we need to create ***testng.xml*** file to create and handle multiple test classes. 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.

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.

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

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

We could define the parameters in the *testng.xml* file and then reference those parameters in the source files.

Create a java test class, say, *ParameterizedTest.*`java` and add a test method say *parameterizedTest()* to the test class. This method takes a string as input parameter. Add the annotation @Parameters("browser") to this method.

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

The parameter would be passed a value from testng.xml, which we will see in the next step.

We could set the parameter using the below syntax in the *testng.xml file.*

```
<parameter name="browser" value="firefox"/>
```

Here, name attribute represents the parameter name and value represents the value of that parameter.

Practical Example

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 supported by TestNG are

- assertEqual(String actual,String expected)
- assertEqual(String actual,String expected, String message)
- assertEquals(boolean actual,boolean expected)
- assertTrue(condition)
- assertTrue(condition, message)
- assertFalse(condition)
- assertFalse(condition, message)

9. What is Soft Assert in TestNG?

Soft Assert collects errors during `@Test`. Soft Assert does not throw an exception when an assert fails and would continue with the next step after the assert statement.

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 *AssertException* immediately when an assert statement fails and test suite continues with next `@Test`

Practical Example

11. What is exception test in TestNG?

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

Practical Example

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.

```
// TestNG Interview Questions
package TestNG;
import org.testng.annotations.*;
public class PriorityTestCase{
 @Test(priority=0)
 public void testCase1() {
 system.out.println("Test Case 1");
 }
 @Test(priority=1)
 public void testCase2() {
 system.out.println("Test Case 2");
 }
}
```

Output:

```
Test Case 1
Test Case 2
```

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:

- using *testng.xml* – Practical Example
- using *Data Providers* – Practical Example

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

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

```
// TestNG Interview Questions
@DataProvider(name="getData")
 public Object[][] getData(){
 //Object [][] data = new Object [rowCount][colCount];
 Object [][] data = new Object [2][2];

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

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

 return data;

 }
```

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

TestNG allows you to perform sophisticated groupings of test methods. Not only can you declare that methods belong to groups, but you can also specify groups that contain other groups. Then TestNG can be invoked and asked to include a certain set of groups (or regular expressions) while excluding another set. This gives you maximum flexibility in how you partition your tests and doesn't require you to recompile anything if you want to run two different sets of tests back to back.

Groups are specified in your `testng.xml` file and can be found either under the `<test>` or `<suite>` tag. Groups specified in the `<suite>` tag apply to all the `<test>`tags underneath.

```
@Test (groups = { "smokeTest", "functionalTest" })

public void loginTest(){

System.out.println("Logged in successfully");

}
```

16. How to create Group of Groups in TestNG?

Groups can also include other groups. These groups are called *MetaGroups*. For example, you might want to define a group *all* that includes *smokeTest* and *functionalTest*. Let's modify our testng.xml file as follows:

```
<groups>

   <define name="all">

 <include name="smokeTest"/>
```

```
<include name="functionalTest"/>
 </define>
 <run>
      <include name="all" />
 </run>
</groups>
```

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 – All the test cases inside <test> tag of testng.xml file will run parallel
classes – All the test cases inside a java class will run parallel
methods – All the methods with *@Test* annotation will execute parallel
instances – Test cases in same instance will execute parallel but two methods of two different instances will run in different thread.

```
<suite name="softwaretestingmaterial" parallel="methods">
```

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

By adding the exclude tag in the *testng.xml*

```
<classes>
 <class name="TestCaseName">
   <methods>
     <exclude name="TestMethodNameToExclude"/>
   </methods>
 </class>
```

```
</classes>
```

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

By adding the exclude tag in the *testng.xml*

```
<groups>
   <run>
 <exclude name="TestGroupNameToExclude"/>
   </run>
</groups>
```

20. How to disable a test case in TestNG ?

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

```
@Test(enabled = false)
```

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

By using *throw new SkipException()*

Once *SkipException()* thrown, remaining part of that test method will not be executed and control will goes directly to next test method execution.

```
throw new SkipException("Skipping - This is not ready for testing ");
```

22. How to Ignore a test case in TestNG?

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

```
@Test(enabled = false)
```

23. How TestNG allows to state dependencies?

TestNG allows two ways to declare the dependencies.
Using attributes dependsOnMethods in @Test annotations Using attributes dependsOnGroups in @Test annotations

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?

TestNG listeners are used to configure reports and logging. One of the most widely used listeners in testNG is *ITestListener* interface. It has methods like *onTestStart, onTestSuccess*, *onTestFailure*, *onTestSkipped* etc. 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.

Practical Example

26. How to write regular expression In testng.xml file to search @Test methods containing "smoke" keyword.
Regular expression to find @Test methods containing keyword "smoke" is as mentioned below.

```
<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

- Using Eclipse IDE
- Using ant build tool
- From the command line
- Using IntelliJ's IDEA

29. How To Run TestNG Using Command Prompt?

Run the TestNG using command prompt

Open command prompt and use the below code

```
C:\Users\Admin\Desktop\STMSeleniumTutorial\workspace\SoftwareTestingMaterial



set
classpath=C:\Users\Admin\Desktop\STMSeleniumTutorial\workspace\SoftwareTestingMateri
al\bin;C:\Users\Admin\Desktop\STMSeleniumTutorial\workspace\SoftwareTestingMaterial\li
b\*


java org.testng.TestNG

C:\Users\Admin\Desktop\STMSeleniumTutorial\workspace\SoftwareTestingMaterial\testng.x

ml
```

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

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

```
@Test(invocationCount = 10)

public void testCase1(){
```

In this example, the method `testCase1` will be invoked ten times

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

```
@Test(threadPoolSize = 3, <code class="plain">invocationCount = </code><code
class="value">10</code>) public void testCase1(){
```

In this example, the method `testCase1` will be invoked from three different threads

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(){
```

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.