TestNG Interview Questions | TestNG, where NG stands for “next generation” is the next step towards writing test cases.

It was developed by Cedric Beust to overcome the deficiency in JUnit by introducing some more functionalities and features.

Here, we have listed topic wise the best collection of frequently asked TestNG interview questions with the best possible answer for the interview.

The list of top 50+ TestNG interview questions and answers are for freshers and experienced.

Topic-wise TestNG Interview Questions with Answers for Fresher

---

1. What is TestNG?

Ans: TestNG is an open-source automation testing framework based on JUnit by introducing some more functionalities and features. So, TestNG is more powerful than JUnit framework.

It can be integrated with Selenium or any other automation tool to provide multiple features like assertions, reporting, parallel test execution, etc.

2. What are the important features of TestNG?

Ans: TestNG include the following features that are as follows:

1. TestNG uses more object-oriented and java features.

2. TestNG supports multiple types of Before/After annotations to create test cases.

3. It provides different types of assertions that helps in checking actual and expected results.

4. It provides priority and grouping features by which test cases can be grouped and prioritized easily.

5. TestNG internally generates an HTML test report in a readable format.

6. It supports parallel testing, integrated testing, load testing, etc.

7. TestNG allows Data-driven testing using @DataProvider annotations.

8. It supports for parameterizing test cases using @Parameters annotation.

3. What are some advantages of TestNG over JUnit?

Ans: TestNG have the following advantages over JUnit. They are as follows:

1. TestNG generates HTML test report in a readable format.

2. Test methods can be grouped and prioritized more easily that is not possible in the case of JUnit.

3. Annotations of TestNG are easy to understand for creating test cases.

4. In TestNG, we can execute multiple test cases in parallel to reduce execution time.

5. Using TestNG, we can define the dependency of one test method over another.

6. TestNG doesn't need to extend any class.

7. TestNG supports three additional annotations @Before/After Suite, @Before/After Test, and @Before/After Group.

4. Why do we need TestNG in Selenium?

Ans: There are following reasons for integrating TestNG with Selenium.

1. Since Selenium WebDriver does not generate test reports, by integrating TestNg with Selenium, we can easily generate test reports/results in a proper readable format to know how many test cases are passed, failed, and skipped.

2. We can execute selective test cases in TestNG.

3. Using TestNG, we can also execute failed test cases separately.

4. TestNG framework can be integrated easily with other tools like Maven, Jenkins, etc.

5. We can easily organize test cases in a proper way.

5. What are the basic steps required in writing TestNG test?

Ans: There are three basic steps required in writing TestNG test that is as follows:

1. Write down the business logic of the test and annotate it by TestNG annotations.

2. Create testing.xml file and add the information about your test

3. Run TestNG.

6. How to run test script in TestNG?

Ans: We can run test script in TestNG by right-clicking on the TestNG class, click on "Run As" option and then select "TestNG test".

Frequently Asked Questions on TestNG Annotations

7. What are Annotations in TestNG?

Ans: Annotations in TestNG are a set of code that controls how method below them have to be executed i.e, the order of execution of methods below them will be decided by annotations that we give.

8. What are the different annotations available in the TestNG?

Ans: TestNG supports a powerful set of different annotations that run tests in an organized and predictable manner.

a. Precondition annotations

Precondition annotations are those annotations of TestNG that are executed before the execution of test methods The Precondition annotations are @BeforeSuite, @BeforeClass, @BeforeTest, @BeforeMethod.

b. Test annotation

Methods annotated with @Test are called test methods which serve as a unit test.

c. Postcondition annotations

The postcondition annotations are those annotations that are executed after the execution of all the test methods. The postcondition annotation can be @AfterSuite, @AfterClass, @AfterTest, @AfterMethod.

9. What is the sequence of execution of all TestNG annotations?

Ans: The sequence of execution of all annotations in TestNG is as follows:

- @BeforeSuite
- @BeforeTest
- @BeforeClass
- @BeforeMetho
- @Test
- @AfterMethod
- @AfterClass
- @AfterTest
- @AfterSuite

10. What is difference between Suite, Test, and Class?

Ans: Suite: A suite is made of one or more tests.

Test: A test is made of one or more classes.

Class: A class is made of one or more methods.

11. What is the difference between @BeforeClass and @BeforeMethod?

Ans: There are two main differences between @BeforeClass and @BeforeMethod. They are as follows:

1. The method with @BeforeClass will be executed only once before any of the tests in the current class are run whereas, a method annotated with @BeforeMethod will be executed before each method annotated with @Test.

2. @BeforeClass annotation can be used to set up the configuration and initialization which is common to all test methods in the current class. For example, we can set up driver configuration which will be common for all tests in the class.

@BeforeMethod can be used to set that data which is repeating before each @Test annotated method.

12. What is the test method?

Ans: A method annotated with @Test is called test method which serves as a unit test. In the @Test method, we will write the logic of the application which we want to automate.

13. What is the return type of @DataProvider annotation provided by TestNG?

Ans: @DataProvider annotation return an object double array (Object [ ][ ]) as data to the test method.

14. What is the return type of @Factory annotation?

Ans: This annotation returns an array of class objects (Object [ ])

Frequently Asked Interview Questions on TestNG Priority and Groups

15. What is Priority in TestNG?

Ans: Priority is an attribute that tells TestNG which orders have to follow to run test method. It is used to set the order of test cases as ours need.

16. What is the default value for TestNG Priority?

Ans: The default value for TestNG priority is zero.

17. How to set Priority of test cases in TestNG?

Ans: TestNG priority attribute is useful to execute the test cases in order. It has the following general form to set the priority of test cases.
Syntax:

```
@Test(prioirty = 1) // Valid syntax.

@Test(PRIORITY = 1) // Invalid sysntax. IDE will show it as a compilation error.
```

18. How will you execute methods or test cases in TestNG in different order/your order?

Ans: When we have multiple methods or test cases and want to execute in order, TestNG priority attribute is useful to execute multiple test cases in your order. If you have not mentioned test priority, TestNG will assign all @Test a priority as zero (0) and execute them.

19. Define TestNG Groups.

Ans: Groups is an attribute in TestNG that allows us to perform groupings of test methods or test cases. It allows us to declare a set of test methods in a particular named group or multiple groups and run multiple groups of tests at different times or in different places.

For example, if we have 100 test cases of university and 10 test cases of college, and if you want to run all the test cases of university together in a single suite, this can be possible only through the grouping of test cases.

20. How to group multiple test methods in a single group using TestNG?

Ans: We can group multiple test methods by using groups parameter in the @Test annotation. The syntax for grouping multiple test methods in a single group is as follows:
Syntax:

```
@Test(groups = {"GroupName"})

For example:

@Test(groups = {"Chemistry"})
public void atom() {
............
}
@Test(groups = {"Chemistry"})
public void electorn()  {
```

```
................
}
```

Both test methods will execute in one group named Chemistry.

21. How to group multiple test methods in multiple groups?

Ans: The grouping of test methods belonging to multiple groups can be done by providing the group names as an array in the groups attribute of the @Test annotation.

The below syntax lets you add test methods in multiple groups.
Syntax:

```
@Test(groups = { "GroupName1", "GroupName2", "GroupName3" .... })
```

22. How to group multiple test methods with Priority?

Ans: You can also group multiple test methods with priority. The syntax for grouping test methods with priority is as follows:
Syntax:

@Test(groups = {"GroupName"}, priority=0) // The test method annotated with this group will execute first.

@Test(groups = {"GroupName"}, priority=1) // The test method annotated with this group will execute after executing the first group.

23. What is Inclusion & Exclusion Groups in TestNG?

Ans: A group that is included in test execution is called inclusion group. A group which is excluded from test execution is called exclusion group.

24. How to disable a test in TestNG?

Ans: We can also disable tests on an individual basis by using the "enabled" property available on both @Test and @Before/After annotations. The syntax can be like this:
Syntax:

```
@Test(enabled = false)
@Test(groups = {"Cricket Player"}, enabled=false)
```

25. What is Default Group, Partial Groups, and Meta Groups in TestNG?

Ans: Default Group: When an entire class is added to a group, it is called default group. It is a good way of defining a default group for all unit tests within a class.

Partial Groups: When you define groups at the class level and then add groups at the method level, it is called partial groups.

MetaGroups: When groups include other groups, these groups are called metagroups.

Frequently Asked Interview Questions on TestNG @Test Annotation & its Supported Attributes

26. What is @Test Annotation in TestNG?

Ans: @Test annotation is a basic annotation in TestNG that marks a method as a test method. The test method annotated with @Test annotation tells TestNG that this method is a "test" and it should be executed when user runs the class.

27. What are the attributes supported by @Test annotation in TestNG?

Ans: @Test annotation supports lots of attributes that we can use with this annotation. Some of the important attributes supported by Test annotation are alwaysRun, dataProvider, dataProviderClass, dependsOnGroups, dependsOnMethods, enabled, priority, timeOut, etc.

28. Which attribute is used to run test method always?

Ans: The attribute "alwaysRun" supported by @Test annotation is used to run test methods always. It takes a value as true or false. If we set true, this method will always execute even its depending method fails. It has the following general form.
Syntax:

```
@Test(alwaysRun = true)
```

29. Which attribute is used to provide data to test method in Data-driven testing?

Ans: The dataProvider attribute is used to provide data to the test method directly in data-driven testing.
Syntax:

```
@Test(dataProvider = "getData")
```

30. What is dependency in TestNG?

Ans: When we want to execute multiple test cases in a specific order, then we use the concept of dependency in TestNG. There are two types of dependency attributes used in TestNG. They are:

a. **dependsOnMethods:** The attribute dependsOnMethods is used to make the test method depend on a particular method. The test method annotated with @Test and attribute dependsOnMethods will run after executing all those methods on which this test method is dependent.
Syntax:

@Test(dependsOnMethods={"Method1", "Method2". . .})

b. **dependsOnGroups:** This attribute is used to make test methods depend on a particular group. All of the methods of these groups are executed first before this method.
Syntax:

@Test(dependsOnGroups={"GroupA", "GroupB", . . .})

31. How will you skip a particular test case in TestNG?

Ans: We can skip or disable a particular test case using enabled attribute. It can be achieved by setting enabled attribute of the Test annotation to false.
Syntax:

@Test(enabled = false)

32. What is invocation Count in TestNG?

Ans: invocationCount in TestNG is an attribute that is used to execute a method or test in the number of times. It acts as a loop.
For example:

@Test(invocationCount = 5) Hence, this method will execute 5 times.

33. What is timeOut in TestNG?

Ans: timeOut in TestNG is an attribute that is used for a time out test. It specifies a time period (in milliseconds) to wait for a test for complete execution.

FAQs on TestNG XML File

34. What is testng.xml file?

Ans: Testng.xml file is a configuration file (XML file) for TestNG in which we can create test suites, test groups, mark tests for parallel execution, add listeners, and pass parameters to test script. It defines the runtime definition of a test suit.

35. What is the importance or use of testng.xml file?

Ans: There are following importance or usage of the testng.xml file in TestNG. They are as follows:

1. The testng.xml file can be used to control the execution of whole tests from a single file.
2. We can run a set of test cases from a single place.
3. We can pass parameters to test methods or classes.

4. Using testng.xml file, we can perform parallel test execution.
5. We can add the listener.

6. It defines the order of the execution of all the test cases.
7. It allows us to group test cases and can be executed as per requirements.
8. It helps to execute selected test cases.
9. With the help of testng.xml file, we can integrate TestNG framework with Jenkins.

36. How to pass parameters in test cases through testng.xml file?

Ans: We can pass the parameter value to test methods at runtime through the testng.xml file. For this, we can use @Parameter annotation:
Syntax:

```
@Parameter("param-name");
```

37. What is thread-count in TestNG?

Ans: The thread-count in TestNG is an attribute that is used to run the maximum number of threads for each suite if the parallel mode is enabled (otherwise ignore it).

For example, thread-count = “2”: It will run your tests with two threads.

38. What is verbose in TestNG?

Ans: verbose is an attribute in TestNG which is mostly used when reporting a bug or when trying to diagnose the execution of test run.

39. How to enable or disable test cases using testng.xml file?

Ans: We can enable or disable test cases by using <include> and <exclude> tags in the testng.xml file.

FA Interview Questions on TestNG Parameterization and Parallel Testing

40. What is Parameterization in TestNG?

Ans: Parameterization is a feature provided by TestNG which allows the users to pass parameter values to test methods as arguments.

Using this feature, we can run the same test over and over again with different values. The parameterization test is supported by using Parameter and DataProvider annotations of TestNG.

41. How many ways by which we can pass parameter values to test methods?

Ans: There are mainly two ways through which we can directly pass parameter values to test methods.

1. Through testng.xml file.
2. Through DataProviders

42. Which annotation is used to pass parameter values to test method from the testng.xml file?

Ans: Parameter annotation.

43. What is DataProvider?

Ans: DataProvider feature provided by TestNG is the second way of passing parameters to test methods. It allows the users to write data-driven tests in which we can run multiple times the same test method with different sets of test data.

TestNG provides a @DataProvider annotation to use the DataProvider feature in tests. This annotation is declared with a test method in the class.

@DataProvider annotation takes an attribute “name” called dataProvider in the Test annotation. It has the following general syntax/form to use.
Syntax:

```
@DataProvider(name = "myTestData")
```

44. What is the return type of DataProvider?

Ans: DataProvider returns a two dimensional an array of object (Object [ ][ ]) to the test method.

45. How to Call DataProvider from another Class?

Ans: In the test method, we will add one more attribute “dataProviderClass” in @Test annotation. The syntax is like this:
Syntax:

```
@Test(dataProvider = "getData", dataProviderClass = DataProviderClass.class) // Here,
```
DataProviderClass is the name of class.

46. What are the advantages of TestNG DataProvider feature?

Ans: The advantages of data provider in TestNG are as follows:
1. TestNG Data provider helps to pass parameter values directly to the test method.

2. It allows users to write data-driven test where we can run multiple times the same test method with different sets of test data.

47. What is Parallel Testing in TestNG?

Ans: Parallel Testing is a testing technique in which multiple tests are executed simultaneously in different threads to reduce execution time.

48. How to run Parallel Tests, Classes, & Methods in Selenium using TestNG XML File?

Ans: In Selenium, we can run our test methods/classes/tests in parallel by using the "parallel" attribute for Test Suite in the testng.xml file.

FAQs on TestNG Assertion

49. What is Assertion in TestNG?

Ans: An assertion is basically blocks of code that are placed in the test cases to check or verify whether the test has failed or passed. A test will be considered as successful ONLY if test is completed without throwing any exception. i.e, If the conditions are not true (do not satisfy), it will stop the further execution of the test and marks it as failing.

50. Why do we use Assertion in Selenium?

Ans: An assertion in selenium is used to verify the result or TestNG reports. It is mainly used to compare the actual result of an application with the expected result.

51. What are the types of assertions in TestNG?

Ans: There are two types of assertions present in the TestNG framework. They are as follows:

1. Hard Assertion
2. Soft Assertion

Hard Assertion in TestNG

Hard assertion is a type of assertion that throws an AssertException when the assertion condition does not satisfy and the test case is immediately marked as failed.

An example of hard assertion methods are assertEquals, assertNotEquals, assertTrue, assertFalse, assertNull, assertNotNull.

Soft Assertion in TestNG

Soft assertion is a type of assertion that continues with test execution even if the assertion condition is not met. That means soft assertions do not throw an exception when the assert statement (assertion condition) fails.