Frequently Asked TestNG Interview Questions

**Q #1) What is TestNG?**

**Answer:**TestNG is the framework created for executing unit tests in java program by the developers.

TestNG is also used by the software testers to efficiently run the automated test scripts created in Selenium Webdriver. Its full form is “Testing new generation” framework.

TestNG is inspired by “JUnit” which is another framework for unit testing Java programs. In addition to all the features in JUnit, TestNG has its own new features which make it more powerful.

**Q #2) How will you install TestNG in Eclipse?**

**Answer:**

**Follow the below steps to install TestNG on eclipse:**

1. Go to eclipse -> Click on “Help” -> Click on “Install New Software”.
2. Click on the “Add” button, Enter the name(Preferably TestNG) in the “Next” textbox. Enter [this](http://beust.com/eclipse) in “Location” textbox and click on “OK” action button.
3. Check the TestNG checkbox and click on the “Next” action button. The installation will start and the Eclipse will restart after installation.
4. Right click on the project in eclipse -> Select build path -> Configure Build Path.
5. Select the library tab -> Click on Add library button -> Select TestNG-> Click on Next -> Click on Finish and Apply and close.

**Q #3) How to run the TestNG script?**

**Answer:**

**To run the TestNG script:**

Right click on the class in Eclipse, Click on “Run as” and select “TestNG test”.

**OR**

Directly click on the Run button on the toolbar of the Eclipse.

**Q #4) What are the annotations used in TestNG?**

**Answer:**

**There are three sections of annotation in TestNG:**

1. Precondition annotations
2. Test annotation
3. Postcondition annotations

**Precondition annotations:**These are the TestNG annotations which are executed before the test.

@BeforeSuite, @BeforeClass, @BeforeTest, @BeforeMethod are the precondition annotations.

**Test annotation:**This is the annotation which is only mentioned before the test case (Before the method written to execute the test case)

@Test is the test annotation

**Postcondition annotation:**These are the annotations which are executed after the test case. (After the method is written to execute the test case)

@AfterSuite, @AfterClass, @AfterTest, @AfterMethod are the postcondition annotations

**Q #5) What is the sequence of execution of the annotations in TestNG?**

**Answer:**

**The Sequence of execution of the annotations is as follows:**

@BeforeSuite

@BeforeTest

@BeforeClass

@BeforeMethod

@Test

@AfterMethod

@AfterClass

@Aftertest

@AfterSuite

**Q #6) What are the advantages of TestNG?**

**Answer:**

**The advantages of TestNG are as follows:**

1. **TestNG**is an open source framework, hence it is easy to configure.
2. Using **TestNG**we can create the test cases in a systematic way.
3. **TestNG**gives lots of annotations which in turn makes the test case creation easy.
4. Using TestNG, priorities of the tests and the sequence of execution can be defined.
5. Grouping is possible using TestNG.
6. **TestNG**generates HTML reports (Selenium Webdriver cannot generate the test reports alone, **TestNG**helps SW to achieve this).
7. Data parameterization is possible using **TestNG**.
8. In addition to all the functionalities of junit, **TestNG**has its own functionalities, which in turn makes it more powerful.

**Q #7) How to set priorities in TestNG?**

**Answer:** There are always more than one tests or methods in the class. If we do not prioritize these tests or methods, then the methods are selected alphabetically and executed while execution.

If we want to run the tests in the sequence we want, then we need to set the priority along with the @Test annotation.

**This can be done as follows:**

@Test (priority=1), @Test (priority=2)

**Consider the following Example:**

@Test (priority=2)

**public** **void** getText()

{

driver.findElement(By.id(“id”)).getText();

}

@Test(priority=1)

**public** **void** clickelement()

{

driver.findElement(By.id(“id”)).click();

}

In the above example, clickelement() will get executed first as the priority is set to 1.

And getText() will get executed after clickelement() as its priority is set to 2.

**Q #8) How to share the project report using TestNG?**

**Answer:**

**There are a few ways to do so:**

#1) After the execution of the **TestNG**class, there is one tab called “Result of running class “ which is generated next to the console.

We can copy this and share.

#2) After the execution of the **TestNG**class,

* Right click on the project name and refresh
* Click on the “Test-output” folder
* Right click on the “index.html” file and select properties
* Copy the link next to “Location”

We can share this link to see the basic html test report which is generated by **TestN.**

This is the file which gets generated on your machine automatically after the execution of the class using **TestNG**.

**Q #9) How will you define grouping in TestNG?**

**Answer:**We can define grouping in **TestNG**using groups attribute as shown below:

@Test(groups=”title”)

**Q #10) What is a dependency in TestNG?**

**Answer:**There are some methods on which many methods are dependent on.

**For Example,** If we want to test any application, and if the login page of the application is not working then we won't be able to test the rest of the scenarios.

So LoginTest is the method on which many tests are dependent on.

**Hence we will write as follows:**

@Test(dependsOnMethods=”LoginTest”)

Public void homePageLaunched()

{

}

The above code shows that homePageLaunched() method is completely dependent on LoginTest() method.

If LoginTest() is passed, only then the homePageLaunched() method gets executed

**Q #11) What is InvocationCount in TestNG?**

**Answer:**

If we want to execute a test case “n” number of times, then we can use the invocationCount attribute as shown in the below example.

**Example:**

@Test(invocationCount=8)

Public void print()

{

}

In the above example, the print() method will get executed 8 times.

**Q #12) What is timeOut in TestNG?**

**Answer:**

If any method in the script takes a long time to execute, then we can terminate that method using “timeout” in **TestNG**.

@Test(timeout = 5000)

In this case, the method will get terminated in 5000 ms (5 seconds) and the test case is marked as “Failed”.

**Q #13) How to handle exceptions in TestNG?**

**Answer:**If there are some methods from which we expect some exceptions, then we can mention the exception in @Test annotation so that the test case does not fail.

**Example:** If a method is expected to have “numberFormatException” exception, then the test case will fail because of this exception if no try catch block is specified.

But we can do it in TestNG by using “expectedException” attribute as follows.

@Test(expectedException=numberFormatException.class)

Then the test case will run without failing.

**Q #14) What are the common TestNG assertions?**

**Answer:**

**Common TestNG assertions include:**

#1) Assert.assetEquals(String actual, String expected);

* It accepts two strings.
* If both the strings are equal, the test case executes successfully otherwise the test case fails.

#2) Assert.assertEquals(String actual, String expected, String message)

* It accepts two strings.
* If both the strings are equal, the test case executes successfully otherwise the test case fails.
* The message is printed if the test case fails.

#3) Assert.assertEquals(boolean actual, boolean expected)

* It accepts two boolean values.
* If both the boolean values are equal, the test case executes successfully otherwise the test case fails.

#4) Assert.assertTrue(<condition(t/f)>)

* It accepts a boolean value.
* The assertion passes if the condition is True, else an assertion error is displayed.

#5) Assert.assertFalse(<condition(t/f)>)

* It accepts a boolean value.
* The assertion passes if the condition is False, else an assertion error is displayed.

#6) Assert.assertTrue(<condition(t/f)>,message)

* It accepts a boolean value.
* The assertion passes if the condition is True, else an assertion error is displayed with the mentioned message.

#7) Assert.assertFalse(<condition(t/f)>,message)

* It accepts a boolean value.
* The assertion passes if the condition is False, else an assertion error is displayed with the mentioned message.

**Q #15) How to disable a test in TestNG?**

**Answer:**

**To disable a test in TestNG, we have to use the “enabled” attribute as follows:**

@Test(enabled=”false”)

**Q #16) What are the types of asserts in TestNG?**

**Answer:**

To validate the results (pass/fail), we have to use the assertion.

**There are two types of assert in TestNG**

* Hard assert
* Soft assert

**Hard Assert:**

Hard Assert is the normal assert which is used to do validations in the TestNG class.

We have to use Assert class for hard assert as follows:

Assert.assertEquals(actual value, expected value);

If the hard assert fails, then none of the code gets executed after the assert statement.

**Soft Assert:**

If we want to continue the test execution even after the assert statement fails, then we have to use soft assert.

To create a soft assert, we have to create an object of a “softAssert” class as follows

softAssert sassert = new softAssert();  
sassert.assertAll();

So now if the test case fails, the execution is not terminated when we use soft assert.

**Q #17) How to pass parameter in the test case through the testng.xml file?**

**Answer:**If we have a class in which a login method is defined, then we can pass the login parameters to this login method from the testing.xml file

**We will have to use the “@parameters” annotation as follows:**

@Parameters({"user\_name","password"})

@Test

**public** **void** loginapp()

{

driverget(“appname”);

driver.findElement(By.id(“login”)).sendkeys(user\_name);

driver.findElement(By.id(“password”)).sendkeys(password);

}

**Now go to the testng.xml file and enter the parameters there as follows:**

<Suite name = *“suitename”*>

<test name =*”testname”*>

<parameter name =*”user\_name”* value=*”user1”*/>

<parameter password =*”password”* value =*”pass1”*/>

<Classes>

<class name =*”passingparameters”*/>

<classes/>

<test/>

<Suite/>

**Q #18) What is the need to create a testng.xml file?**

**Answer:**When we test a project using Selenium Webdriver, it has a lot of classes in it. We cannot choose these classes one by one and put them for automation. Hence we need to create a suite so that all the classes run in a single test suite.

We can achieve this by creating a testing.xml file.

**Q #19) How to create an XML file in TestNG?**

**Answer:**Go to the src folder -> click on file ->enter the name of the file(mostly written testing.xml)

Click on finish.

Now we have a blank xml file. Here, we have to mention the project name and the classes to be executed along with the package name as shown below.

<Suite name = *"Testing project"*>

<test name = *"testing feature 1"*>

<classes>

<class name = *"packagename.name of class1"*/>

<class name = *"packagename.name of class1"*/>

<class name = *"packagename.name of class1"*/>

<class name = *"packagename.name of class1"*/>

</classes>

</test>

</Suite>

To run this file, we have to go to testng.xml in the package explorer right click and run as -> Testng suite

**Q #20) How to throw a SKIP Exception in TestNG?**

**Answer:**

If we want to SKIP any Test using testing, then we have to use the SKIP exception in TestNG.

**It is written as follows:**

**public** **void** skipExc()

{

System.out.println("SKIP me");

**throw** **new** skipException(“Skipping skipExc”);

}

}