# Top 25 TestNG interview Questions for QA Automation /SDET

**Ques.1. What is TestNG?**
Ans. TestNG(NG for Next Generation) is a testing framework that can be integrated with selenium or any other automation tool to provide multiple capabilities like assertions, reporting, parallel test execution, etc.

**Ques.2. What are some advantages of TestNG?**
Ans. Following are the advantages of TestNG-

1. TestNG provides different assertions that help in checking the expected and actual results.
2. It provides parallel execution of test methods.
3. We can define the dependency of one test method over others in TestNG.
4. We can assign priority to test methods in Selenium.
5. It allows the grouping of test methods into test groups.
6. It allows data-driven testing using @DataProvider annotation.
7. It has inherent support for reporting.
8. It has support for parameterizing test cases using @Parameters annotation.

**Ques.3. How is TestNG different from Selenium WebDriver?**
Ans. Selenium is an automation tool using which we can automate web-based applications. In order to add testing capabilities in the Test Automation suites – Selenium is clubbed with TestNG. With TestNG, we can have different features in our automation suite like different types of assertions, reporting, parallel execution, parameterization, etc.
In short, for doing Automation Testing, Selenium helps with 'automation' and TestNG helps with 'testing' capabilities.

**Ques.4. What is the use of the testng.xml file?**
Ans. The testng.xml file is used for configuring the whole test suite. In the testng.xml file, we can create a test suite, create test groups, mark tests for parallel execution, add listeners, and pass parameters to test scripts. We can also use this testng.xml file for triggering the test suite from the command prompt/terminal or Jenkins.

**Ques.5. How can we group test cases like separate test cases for the Sanity suite, Regression suite, etc?**
Ans. Using the **groups** attribute in TestNG, we can assign the test methods to different groups.

//Test method belonging to sanity suite only

```
@Test(groups = {"sanitySuite"})
public void testMethod1() {
  //Test logic
}

//Test method belonging to both sanity and regression suite
@Test(groups = {"sanitySuite", "regressionSuite"})
public void testMethod2() {
  //Test logic
}
```

**Ques.6. How can we exclude a Test method from getting executed via the testng.xml file?**
Ans. Using the **exclude** tag in the testng.xml file, we can exclude a particular test method from getting executed.

```
<suite name="Test Suite" verbose="1">
  <test name="TestName">
    <classes>
      <class name="TestClassName">
        <methods>
          <exclude name="testMethodToBeExcluded"/>
        </methods>
      </class>
    </classes>
  </test>
</suite>
```

**Ques.7. What are some commonly used TestNG annotations?**
Ans. The commonly used TestNG annotations are-

- @Test – @Test annotation marks a method as a Test method.
- @BeforeSuite – The annotated method will run only once before all tests in this suite have run.
- @AfterSuite – The annotated method will run only once after all tests in this suite have run.

- @BeforeClass – The annotated method will run only once before the first test method in the current class is invoked.
- @AfterClass – The annotated method will run only once after all the test methods in the current class have been run.
- @BeforeTest – The annotated method will run before any test method belonging to the classes inside the <test> tag is run.
- @AfterTest – The annotated method will run after all the test methods belonging to the classes inside the <test> tag have run.
- **@**BeforeMethod – The annotated method will run before each test method marked by @Test annotation.
- **@**AfterMethod – The annotated method will run after each test method marked by @Test annotation.
- @DataProvider – The @DataProvider annotation is used to pass test data to the test method. The test method will run as per the number of rows of data passed via the data provider method.

**Ques.8. What is the order of execution of the test method based on the different annotations?**
Ans. The test methods in TestNG follow the Suite->Test->Class->Method sequence combined with the Before annotations->Test annotations->After annotations sequence. So, the order of execution is-
@BeforeSuite
@BeforeTest
@BeforeClass
@BeforeMethod
@Test
@AfterMethod
@AfterClass
@AfterTest
@AfterSuite

**Ques.9. What are some common assertions provided by TestNG?**
Ans. Some of the common assertions provided by testNG are-

1. assertEquals(String actual, String expected, String message) and other overloaded data types in parameter
2. assertNotEquals(double data1, double data2, String message) and other overloaded data types in parameter
3. assertFalse(boolean condition, String message)
4. assertTrue(boolean condition, String message)
5. assertNotNull(Object object)
6. fail(boolean condition, String message)

7. true(String message)

**Ques.10. How can we disable or prevent a test case from running?**
Ans. By setting the “enabled” attribute as false, we can disable a test method from running.

```
//In case of a test method
@Test(enabled = false)
public void testMethod1() {
  //Test logic
}

//In case of test method belonging to a group
@Test(groups = {"NegativeTests"}, enabled = false)
public void testMethod2() {
  //Test logic
}
```

**Ques.11. How can we make one test method dependent on others using TestNG?**
Ans. Using dependsOnMethods parameter inside @Test annotation in TestNG we can make one test method run only after the successful execution of the dependent test method.

```
@Test(dependsOnMethods = { "preTests" })
```

**Ques.12. How can we set the priority of test cases in TestNG?**
Ans. We can define the priority of test cases using the “priority” parameter in the @Test annotation. The tests with lower priority values will get executed first. Example-

```
@Test(priority=1)
```

**Ques.13. What is the default priority of test cases in TestNG?**
Ans. The default priority of a test when not specified is integer value 0. So, if we have one test case with priority 1 and one without any priority then the test without any priority value will get executed first (as the default value will be 0 and tests with lower priority are executed first).

**Ques.14. How can we run a Test method multiple times in a loop(without using any data provider)?**

Ans. Using invocationCount parameter and setting its value to an integer value, makes the test method runs n number of times in a loop.

```
@Test(invocationCount = 10)
public void invocationCountTest(){
  //Test logic
}
```

**Ques.15. What is threadPoolSize? How can we use it?**
Ans. The **threadPoolSize** attribute specifies the number of threads to be assigned to the test method. This is used in conjunction with **invocationCount** attribute. The number of threads will get divided by the number of iterations of the test method specified in the invocationCount attribute.

```
@Test(threadPoolSize = 5, invocationCount = 10)
public void threadPoolTest(){
  //Test logic
}
```

**Ques.16. What is the difference between soft assertion and hard assertion in TestNG?**
Ans. This is one of the most frequently asked TestNG interview questions. Soft assertions (SoftAssert) allow us to have multiple assertions within a test method, even when an assertion fails the test method continues with the remaining test execution. The result of all the assertions can be collated at the end using softAssert.assertAll() method.

```
@Test
public void softAssertionTest(){
  SoftAssert softAssert= new SoftAssert();

  //Assertion failing
  softAssert.fail();
  System.out.println("Failing");
  //Assertion passing
  softAssert.assertEquals(1, 1);
  System.out.println("Passing");
  //Collates test results and marks them pass or fail
  softAssert.assertAll();
}
```

Here, even though the first assertion fails still the test will continue with execution and print the message below the second assertion.
Hard assertions on the other hand are the usual assertions provided by TestNG. In case of hard assertion in case of any failure, the test execution stops, preventing the execution of any further steps within the test method.

**Ques.17. How to fail a testNG test if it doesn't get executed within a specified time?**
Ans. We can use the **timeOut** attribute of @Test annotation. The value assigned to this timeOut attribute will act as an upper bound. If the test doesn't get executed within this time frame then it will fail with timeout exception.

```
@Test(timeOut = 1000)

public void timeOutTest() throws InterruptedException {

  //Sleep for 2sec so that test will fail

  Thread.sleep(2000);

  System.out.println("Will throw Timeout exception!");

}
```

**Ques.18. How can we skip a test case conditionally?**
Ans. Using SkipException, we can conditionally skip a test case. On throwing the skipException, the test method is marked as skipped in the test execution report and any statement after throwing the exception will not get executed.

```
@Test

public void testMethod(){

 if(conditionToCheckForSkippingTest)

 throw new SkipException("Skipping the test");

  //test logic

}
```

**Ques.19. How can we make sure a test method runs even if the test methods or groups on which it depends fail or get skipped?**
Ans. Using the "alwaysRun" attribute of @Test annotation, we can make sure the test method will run even if the test methods or groups on which it depends fail or get skipped.

```
@Test

public void parentTest() {

  Assert.fail("Failed test");

}
```

```
@Test(dependsOnMethods={"parentTest"}, alwaysRun=true)

public void dependentTest() {

  System.out.println("Running even if parent test failed");

}
```

Here, even though the **parentTest** failed, the **dependentTest** will not get skipped instead it will get executed because of “alwaysRun=true”. In case, we remove the “alwaysRun=true” attribute from @Test then the report will show one failure and one skipped test, without trying to run the **dependentTest** method.

**Ques.20. How can we pass the parameter to the** test the **script using TestNG?**
Ans. Using @Parameter annotation and the ‘parameter’ tag in testng.xml we can pass parameters to test scripts.
Sample testng.xml –

```
<suite name="sampleTestSuite">

  <test name="sampleTest">

    <parameter name="sampleParamName" value="sampleParamValue"/>

    <classes>

      <class name="TestFile" />

    </classes>

  </test>

</suite>
```

Sample test script-

```
public class TestFile {

  @Test

  @Parameters("sampleParamName")

  public void parameterTest(String paramValue) {

    System.out.println("Value of sampleParamName is - " + sampleParamName);

  }
```

**Ques.21. How can we create data-driven framework using TestNG?**
Ans. Using @DataProvider we can create a data-driven framework in which data is passed to the associated test method and multiple iterations of the test run for the different test data values passed from the @DataProvider method. The method annotated with @DataProvider annotation return a 2D array of object.

```
//Data provider returning 2D array of 3*2 matrix
```

```
@DataProvider(name = "dataProvider1")
  public Object[][] dataProviderMethod1() {
    return new Object[][] {{"kuldeep","rana"}, {"k1","r1"},{"k2","r2"}};
  }
  //This method is bound to the above data provider returning 2D array of 3*2 matrix
  //The test case will run 3 times with different set of values
  @Test(dataProvider = "dataProvider1")
  public void sampleTest(String s1, String s2) {
    System.out.println(s1 + " " + s2);
  }
```

**Ques.22. What is the use of @Listener annotation in TestNG?**

Ans. TestNG provides us with different kinds of listeners using which we can perform some action in case an event gets triggered. Usually, testNG listeners are used for configuring reports and logging. One of the most widely used listeners in testNG is the ITestListener interface. It has methods like onTestSuccess, onTestFailure, onTestSkipped, etc. We need to implement this interface by creating a listener class of our own. After that using the @Listener annotation we can use specify that for a particular test class, *our customized listener class should be used.*

```
@Listeners(PackageName.CustomizedListenerClassName.class)
public class TestClass {
   WebDriver driver= new FirefoxDriver();@Test
   public void testMethod(){
   //test logic
   }
}
```

Read more from here.

**Ques.23. What is the use of @Factory annotation in TestNG?**

Ans. @Factory annotation helps in the dynamic execution of test cases. Using @Factory annotation we can pass parameters to the whole test class at run time. The parameters passed can be used by one or more test methods of that class.
For example – there are two classes TestClass and the TestFactory class. Because of the @Factory annotation, the test methods in class TestClass will run twice with the data “k1” and “k2”

```
public class TestClass{
   private String str;
```

```
    //Constructor
    public TestClass(String str) {
        this.str = str;
    }

    @Test
    public void TestMethod() {
        System.out.println(str);
    }
}

public class TestFactory{
    //The test methods in class TestClass will run twice with data "k1" and "k2"
    @Factory
    public Object[] factoryMethod() {
        return new Object[] { new TestClass("K1"), new TestClass("k2") };
    }
}
```

**Ques.24. What is the difference between @Factory and @DataProvider annotation?**
Ans. @Factory method creates instances of test class and runs all the test methods in that class with a different set of data.
Whereas, @DataProvider is bound to individual test methods and runs the specific methods multiple times.

**Ques.25. How can we run test cases in parallel using TestNG?**
Ans. In order to run the tests in parallel just add these two key-value pairs inside the suite tag of the testng.xml file-

- parallel=”{methods/tests/classes}”
- thread-count=”{number of threads you want to run simultaneously}”.

```
<suite name="TestSuite1" parallel="methods" thread-count="5">
```