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. How do you run the TestNG script?

Ans: click on the TestNG class -> Run As -> TestNG Test.

---

3. What are the advantages of TestNG?

-> TestNG provides different assertions that help in checking the expected and actual results.

-> It provides parallel execution of test methods.

-> We can define the dependency of one test method over others

-> We can assign priority to test methods

-> We can group multiple test into group

-> It allows data-driven testing using @DataProvider annotation.

-> It has inherent support for reporting.

-> It has support for parameterizing test cases using @Parameters annotation.

-> Automatic report generation features is avialable

-> TestNG can easily integrate with other tools such as Maven, Jenkins

-> It support cross-browser testing.

---

4. What is the difference between a TestNG test and a TestNG test suite?

TestNG test suite is a collection of tests that we can run simultaneously with the help of the TestNG XML file.
On the other side, a TestNG test is a single test case file.

---

5. Define the correct order of tags in the TestNG XML file.

The correct order followed to run the TestNG suite from the XML file is as follows:

```
<suite>
  <test>
    <classes>
   <class>
        <methods>
        <incude>
```

The closing tags don't appear here as it is just for demonstration purposes.

---

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

-> The testng.xml file is used for configuring the whole test suite.

-> In testng.xml file, we can create a test suite

-> We can 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.

---

7. What are different annoation available in TestNG?

-> @BeforeTest
-> @AfterTest
-> @BeforeClass
-> @AfterClass
-> @BeforeMethod
-> @AfterMethod
-> @BeforeSuite
-> @AfterSuite
-> @BeforeGroups
-> @AfterGroups
-> @Test

==========
8. What are the annotations used in TestNG?

Answer: There are three sections of annotation in TestNG:

(i) Precondition annotations: These are the TestNG annotations that are executed before the test.

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

(ii) 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

(iii) Postcondition annotation: These are the annotations that 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. What is the sequence of execution of the annotations in TestNG?

Answer: The Sequence of execution of the annotations is as follows:
1) @BeforeSuite
2) @BeforeTest
3) @BeforeClass
4) @BeforeMethod
5) @Test
6) @AfterMethod
7) @AfterClass
8) @Aftertest
9) @AfterSuite
==========

==========
10. How to set priorities in TestNG?

-> We can set priorities in TestNg by using "priority" attribute of @Test annotation as below

```
@Test (priority=2)
public void getText()
{
   driver.findElement(By.id(“id”)).getText();
}
@Test(priority=1)
public void clickelement()
{
  driver.findElement(By.id(“id”)).click();
}
```

==========

==========

11. How will you define grouping in TestNG?

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

```
@Test(groups=”group_name”)
```

==========

==========

12 . What is a dependency on TestNG?

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.

==========

==========

13. What is InvocationCount in TestNG?

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.

==========

==========

14. What is timeOut in TestNG?

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”.

==========

==========

15. How to handle exceptions in TestNG?

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.

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.
==========

==========

16. How to disable a test in TestNG?

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

@Test(enabled=”false”)

==========

==========

17. 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:

(i) 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.

(ii) 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.

==========

18 .How to pass parameter in the test case through the testng.xml file?

If we want pass parameter in the Test case through TestNG.xml file then we need to use <paramter> tag either at suite level or test level as below

```
 <Suite name = “suitename”>
     <test name =”testname”>
                <parameter name =”user_name” value=”user1”/>
                <parameter password =”password” value =”pass1”/>
        <Classes>
                <class name =”passingparameters”/>
        <classes/>
  <test/>
<Suite/>
```

To accept those parameter value in Test class we need to use @the “@parameters” annotation

```
@Parameters({"user_name","password"})
@Test
public void loginapp(String user_name,String password)
{
        driverget(“appname”);
        driver.findElement(By.id(“login”)).sendkeys(user_name);
        driver.findElement(By.id(“password”)).sendkeys(password);
}
```

==========