Q: What is the significance of <testng.xml> file?

Answer: In a Selenium TestNG project, we use <testng.xml> file to configure the complete test suite into a single file. This file makes it easy to group all the test suites and their parameters in one file. It also gives the ability to pull out subsets of your tests or split several runtime configurations. Few of the tasks which we can group in the <testng.xml> file are as follows.

1- Can configure test suite comprising of multiple test cases to run from a single place.
2- Can include or exclude test methods test execution.
3- Can mark a group to include or exclude.
4- Can pass parameters in test cases.
5- Can add group dependencies.
6- Can configure parallel test execution.
7- Can add listeners.

Q: How to pass parameter through <testng.xml> file to a test case?

Answer: You can set the parameter using the below syntax in the <testng.xml> file.

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

Here, name attribute represents the parameter name and value signifies the value of that parameter. Then we can use that parameter in the selenium WebDriver software automation test case using the bellow syntax.

```
@Parameters ({"browser"})
```

Q: How to exclude a @Test method from a test case with two @Test methods? Is it possible?

Answer: Yes, you need to add @Test method in the exclude tag of <testng.xml> file as mentioned below.

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Test Exclusion Suite">
      <test name="Exclusion Test" >
            <classes>
                  <class name="Your Test Class Name">
                        <methods>
                              <exclude name="Your Test Method Name To
                        Exclude"/>
                        </methods>
                  </class>
            </classes>
      </test>
</suite>
```

Q: How to skip a @Test method from execution?

Answer: You can use the below syntax inside @Test method to skip a test case from test execution.

```
throw new SkipException("Test Check_Checkbox Is Skipped");
```

It will throw skip exception and @Test method will be ignored immediately from execution.

Q: Can you arrange the below <testng.xml> tags from parent to child?

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

Answer: The <testng.xml> file will have the following structure.

- The parent tag in the <testng.xml> file is the <suite> tag.
- <suite> tag can include one or more <test> tags.
- <test> tag can include the <classes> tag.
- <classes> tag can include one or more <class> tags.
- <class> tag wraps the <methods> tag where we define the test methods to include or exclude.

Hence, the correct order of the TestNG tags would be.

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

Q: How to define the priority of @Test method? Also, mention its usage?

Answer: In your Selenium WebDriver project, you can set priority for TestNG @Test annotated methods as shown in the following example.

```
@Test(priority=0)
```

Using priority, you can manage @Test method execution sequence as per your requirement. That means @Test method with priority = 0 will run 1st and @Test method with priority = 1 will execute 2nd and so on.

Q: Can you specify any 6 assertions of TestNG to be used in a Selenium WebDriver software testing tool.

Answer: There are multiple assertions available In TestNG but generally we use the following assertions in our test cases.

```
1- assertEquals
2- assertNotEquals
3- assertTrue
4- assertFalse
```

```
5- assertNull
6- assertNotNull
```

Q: Why soft assertion is used in Selenium WebDriver and TestNG automation project?

Answer: TestNG soft assertion allows to continue the test execution even if the assertion is failed. That means once the soft assertion fails, remaining part of the <@Test> method is executed and the assertion failure is reported at the end of the <@Test> method.

Q: How to apply regular expression in <testng.xml> file to find @Test methods containing "product" keyword?

Answer: Refer below example, here we've used a regular expression to find @Test methods containing keyword "product".

```
<methods>
<include name=".*product.*"/>
</methods>
```

Q: What are the time unit we specify in test cases and test suites? Minutes? Seconds? Milliseconds? or hours? Give Example.

Answer: Time unit we specify at @Test method level and test suite level which normally set in milliseconds unit.

Q: List out the benefits of TestNG over Junit?

Answer: TestNG framework has following benefits over JUnit.

1- TestNG annotations are more logical and easier to understand.
2- Unlike JUnit, TestNG does not require to declare @BeforeClass and @AfterClass.
3- There is no method name constraint in Selenium TestNG framework.
4- TestNG supports three additional setups:
    4.1- @Before/AfterSuite,
    4.2- @Before/AfterTest, and
    4.3- @Before/AfterGroup.
5- In Selenium TestNG projects, there is no need to extend any class.
7- In TestNG, it is possible to run Selenium test cases in parallel.
8- TestNG supports grouping of test cases which is not possible in JUnit.
9- Based on the group, TestNG allows you to execute the test cases.
10- TestNG permits you to determine the dependent test cases. Every test case is autonomous to other test cases.

Q: What are the basic steps for drafting TestNG test cases?

Answer: Following are the most common steps for writing TestNG test cases.

1- Write down the business logic of your test.
2- Add appropriate TestNG annotations in your code.
3- In <build.xml> or <testing.xml>, add the information about your test.
4- Run your TestNG project.

Q: List out different ways to run TestNG?

Answer: You can run TestNG in the following ways.

1- Start directly from the Eclipse IDE, or
2- Run using the IntelliJ's IDEA IDE.
3- Run with ant build tool.
4- Launch from the command line.

Q: In TestNG how can you disable a test?

Answer: To disable the test case, you can use the following annotation.

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

Q: Explain what does the test timeout mean in TestNG?

Answer: The timeout test in TestNG is nothing but the time allotted to perform unit testing. If the unit test fails to finish in that specific time limit, TestNG will abandon further testing and mark it as a failed.

Q: What is exception test and why is it used for?

Answer: TestNG provides an option for tracing the Exception handling of code. You can verify whether a code throws the desired exception or not. The **expectedExceptions** parameter is availed along with @Test annotation. Please refer the below example for clarity.

```
package com.techbeamers.testng.examples.exception;

import org.testng.annotations.Test;

public class TestRuntime

{

      @Test(expectedExceptions = { IOException.class })

      public void exceptionTestOne() throws Exception

      {

            throw new IOException();

      }

      @Test(expectedExceptions = { IOException.class,
      NullPointerException.class })


      public void exceptionTestTwo() throws Exception
```

```
        {
                throw new Exception();
        }
}
```

Q: Explain what is parametric testing?

Answer: Parameterized testing lets the programmer re-run the same test with different values. TestNG allows you to pass parameters directly to the test methods using the two ways as given below.

1- With testing.xml.
2- With Data Providers.

Q: Explain what does @Test(invocationCount=?) and (threadPoolSize=?) indicates?

Answer:

1- @Test (threadPoolSize=?): The threadPoolSize attribute directs TestNG to create a thread pool to run the test method through multiple threads. With thread pool, the running time of the test method reduces considerably.
2- @Test(invocationCount=?): The invocation count refers to the no. of times TestNG should run the test method.

Q: What are the different ways to produce reports for TestNG results?

Answer: TestNG offers following two ways to produce a report.

Listeners: For a listener class, the class has to implement the org.testng./TestListener interface. TestNG notifies these classes at runtime when the test enters into any of the below states. e.g. Test begins, finishes, skips, passes or fails.

Reporters: For a reporting class to implement, the class has to implement an org.testng/Reporter interface. When the whole suite run ends, these classes are called. When called, the object consisting the information of the whole test run is delivered to this class.

Q: How does TestNG allow you to state dependencies?

Answer: TestNG supports two ways to declare the dependencies.

1- Using attributes dependsOnMethods in @Test annotations.
2- Using attributes dependsOnGroups in @Test annotations.