NGA

https://www.nextgenerationautomation.com

TestNG Interview Questions and Answers

Prepared By:

Next Generation Automation Academy

Website: https://www.nextgenerationautomation.com

Question: What is TestNG?
Answer:
TestNG is a testing framework designed to simplify a broad range of testing needs, from unit testing to integration testing.

Question. What are the annotations available in TestNG?
Answer:
@BeforeTest
@AfterTest
@BeforeClass
@AfterClass
@BeforeMethod
@AfterMethod
@BeforeSuite
@AfterSuite
@BeforeGroups
@AfterGroups
@Test

Question: What is TestNG Assert and list out some common Assertions supported by TestNG?
Answer:
TestNG Asserts help us to verify the condition of the test in the middle of the test run. Based on the TestNG Assertions, we will consider a successful test only if it is completed the test run without throwing any exception.

Some of the common assertions supported by TestNG are

assertEqual(String actual,String expected)
assertEqual(String actual,String expected, String message)
assertEquals(boolean actual,boolean expected)
assertTrue(condition)
assertTrue(condition, message)
assertFalse(condition)
assertFalse(condition, message)

Different TestNG Asserts Statements:

a) Assert.assertEqual(String actual,String expected) :
Asserts that two Strings are equal. If they are not, an AssertionError is thrown.

Parameters:
actual – the actual value
expected – the expected value

b) Assert.assertEqual(String actual,String expected, String message) :
Asserts that two Strings are equal. If they are not, an AssertionError, with the given message, is thrown.
Parameters:
actual – the actual value
expected – the expected value
message – the assertion error message

c) Assert.assertEquals(boolean actual,boolean expected) :
Asserts that two booleans are equal. If they are not, an AssertionError is thrown.
Parameters:
actual – the actual value
expected – the expected value

d) Assert.assertTrue(condition) :
Asserts that a condition is true. If it isn't, an AssertionError is thrown.
Parameters:
condition – the condition to evaluate

e) Assert.assertTrue(condition, message) :
Asserts that a condition is true. If it isn't, an AssertionError, with the given message, is thrown.
Parameters:
condition – the condition to evaluate
message – the assertion error message

f) Assert.assertFalse(condition) :
Asserts that a condition is false. If it isn't, an AssertionError is thrown.
Parameters:
condition – the condition to evaluate

g) Assert.assertFalse(condition, message) :
Asserts that a condition is false. If it isn't, an AssertionError, with the given message, is thrown.
Parameters:
condition – the condition to evaluate
message – the assertion error message

Question: How to set test case priority in TestNG?
Answer:
We use priority attribute to the @Test annotations.

In case priority is not set then the test scripts execute in alphabetical order.

```
1.  package nextGenerationAutomationLearnTestNG;
2.
3.  import org.testng.annotations.*;
4.
5.  public class PriorityTestCase {
6.
7.  @Test(priority=0)
8.  public void testCase1() {
9.      System.out.println("Test Case 1");
10. }
11.
12. @Test(priority=1)
13. public void testCase2() {
14.   System.out.println("Test Case 2");
15.   }
16. }
17.
18. Output:
19. Test Case 1
20. Test Case 2
```

Question: What is Parameterized testing in TestNG?
Answer:
Parameterized tests allow developers to run the same test over and over again using different values.

There are two ways to set these parameters:
With testng.xml
With Data Providers
Let's see passing parameters with testng.xml:

With this technique, we could define the parameters in the testng.xml file and then reference those parameters in the source files.

Create a java test class, say, ParameterizedTest.java

Add test method parameterizedTest() to your test class. This method takes a string as input parameter

Add the annotation @Parameters("browser") to this method. The parameter would be passed a value from testng.xml, which we will see in the next step.

```
1.  package nextGenerationAutomationLearnTestNG;
2.
3.  import org.testng.annotations.Parameters;
4.  import org.testng.annotations.Test;
5.
6.  public class ParameterizedTest {
7.
8.  @Test
9.  @Parameters("browser")
10. public void parameterizedTest(String browser){
11.     if(browser.equals("firefox")){
12.       System.out.println("Open Firefox Driver");
13.     }
14.     else if(browser.equals("chrome")){
15.       System.out.println("Open Chrome Driver");
16.     }
17.   }
18. }
```

Testng.xml

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
3.  <suite name="nextGenerationAutomation">
4.    <test name="testngTest">
5.      <parameter name="browser" value="firefox"/>
6.      <classes>
7.        <class name="nextGenerationAutomation.ParameterizedTest" />
8.      </classes>
9.    </test>
10. </suite>
```

Console Output:

[TestNG] Running:
Open Firefox Driver
==========
nextGenerationAutomationLearnTestNG
Total tests run: 1, Failures: 0, Skips: 0
==========

TestNG will automatically try to convert the value specified in testng.xml to the type of your parameter. Here are the types supported:

String
int/Integer
boolean/Boolean
byte/Byte

char/Character
double/Double
float/Float
long/Long
short/Short

Question. How can we create a data-driven framework using TestNG?
Answer:
By using @DataProvider annotation, We can create a Data Driven Testing Framework.

```
1.  package nextGenerationAutomationLearnTestNG;
2.
3.  import org.testng.annotations.DataProvider;
4.  import org.testng.annotations.Test;
5.
6.  public class DataProviderClass {
7.
8.    // This method takes data as input parameters. The attribute dataprovider is mapped to "get-
    Data"
9.    @Test (dataProvider="getData")
10.   // Number of columns should match the number of input parameters
11.   public void loginTest(String Uid, String Pwd){
12.     System.out.println("UserName is "+ Uid);
13.     System.out.println("Password is "+ Pwd);
14.   }
15.
16.   //If the name is not supplied, the data provider's name automatically de-
    faults to the method's name.
17.   //A data provider returns an array of objects.
18.   @DataProvider(name="getData")
19.   public Object[][] getData(){
20.     //Object [][] data = new Object [rowCount][colCount];
21.     Object [][] data = new Object [2][2];
22.     data [0][0] = "FirstUid";
23.     data [0][1] = "FirstPWD";
24.     data[1][0] = "SecondUid";
25.     data[1][1] = "SecondPWD";
26.     return data;
27.   }
28. }
29.
30. Console Output:
31. UserName is FirstUid
32. Password is FirstPWD
33. UserName is SecondUid
34. Password is SecondPWD
35. ===============================================
36. Total tests run: 2, Failures: 0, Skips: 0
```

Question: What are TestNG Groups?
Answer:
TestNG allows you to perform sophisticated groupings of test methods. Not only can you declare that methods belong to groups, but you can also specify groups that contain other groups. Then TestNG can be invoked and asked to include a certain set of groups (or regular expressions) while excluding another set. This gives you maximum flexibility in how you partition your tests and doesn't require you to recompile anything if you want to run two different sets of tests back to back.

Groups are specified in your testng.xml file and can be found either under the <test> or <suite> tag. Groups specified in the <suite> tag apply to all the <test> tags underneath.

Script – Test Case 1:

```
1.  package nextGenerationAutomationLearnTestNG;
2.
3.  import org.testng.annotations.Test;
4.
5.  public class TestCase1 {
6.    @Test (groups = { "smokeTest" })
7.    public void loginTest(){
8.      System.out.println("Logged in successfully");
9.    }
10. }
```

Script – Test Case 2:

```
1.  package nextGenerationAutomationLearnTestNG;
2.
3.  import org.testng.annotations.Test;
4.
5.  public class TestCase2 {
6.    @Test (groups = { "functionalTest" })
7.    public void composeMail(){
8.    System.out.println("Mail Sent");
9.    }
10. }
```

testng.xml:

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
3.  <suite name="nextGenerationAutomation">
4.    <test name="testngTest">
```

```
    <groups>
      <run>
        <include name="smokeTest" />
      </run>
    </groups>
   <classes>
     <class name="nextGenerationAutomation.TestCase1" />
     <class name="nextGenerationAutomation.TestCase2" />
   </classes>
  </test>
</suite>

Console Output:
[TestNG] Running:
Logged in successfully
===============================================
nextGenerationAutomationLearnTestNG
Total tests run: 1, Failures: 0, Skips: 0
===============================================
```

Group of Groups in TestNG Groups:

Groups can also include other groups. These groups are called MetaGroups. For example, you might want to define a group all that includes smokeTest and functionalTest. Let's modify our testng.xml file as follows:

testng.xml – Group of Groups:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="nextGenerationAutomation">
  <test name="testngTest">
    <groups>
      <define name="all">
        <include name="smokeTest"/>
        <include name="functionalTest"/>
      </define>
      <run>
        <include name="all" />
      </run>
    </groups>
    <classes>
        <class name="nextGenerationAutomation.TestCase1" />
        <class name="nextGenerationAutomation.TestCase2" />
     </classes>
  </test>
</suite>

Console Output:
[TestNG] Running:
C:\Users\NextGenerationAutomation\Desktop\TestNGProject\testng.xml
Logged in successfully
Mail Sent
===============================================
nextGenerationAutomation
Total tests run: 2, Failures: 0, Skips: 0
===============================================
```

Groups Exclusion:
TestNG allows you to include groups as well as exclude them. You can ignore a group by using the <exclude> tag as shown below:

For example, it is quite usual to have tests that temporarily break because of a recent change, and you don't have time to fix the breakage yet. However, you do want to have clean runs of your functional tests, so you need to deactivate these tests but keep in mind they will need to be reactivated.

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
3.  <suite name="nextGenerationAutomation">
4.    <test name="testngTest">
5.      <groups>
6.        <run>
7.           <exclude name="smokeTest"/>
8.           <include name="functionalTest"/>
9.        </run>
10.     </groups>
11.     <classes>
12.       <class name="nextGenerationAutomation.TestCase1" />
13.       <class name="nextGenerationAutomation.TestCase2" />
14.     </classes>
15.   </test>
16. </suite>
17.
18. Console Output:
19. [TestNG] Running:
20. Mail Sent
21. ===============================================
22. nextGenerationAutomationLearnTestNG
23. Total tests run: 1, Failures: 0, Skips: 0
24. ===============================================
```

You can also disable tests on an individual basis by using the "enabled" property available on both @Test and @Before/After annotations.

Question: What are TestNG Listeners – Selenium WebDriver
Answer:
Listeners "listen" to the event defined in the selenium script and behave accordingly. The main purpose of using listeners is to create logs. There are many types of listeners such as WebDriver Listeners and TestNG Listeners.

Let's see how to implement TestNG Listeners.

Step 1: Create a Class "ListenerTestNG" to implement ITestListener methods
In the example we are implementing 3 methods of ITestListerner:
onTestFailure, onTestSkipped, onTestSuccess.

```
1.  package nextGenerationAutomationLearnTestNG;
2.
3.  import org.testng.ITestContext;
4.  import org.testng.ITestListener;
5.  import org.testng.ITestResult;
6.
7.  public class ListenerTestNG implements ITestListener
8.  {
9.    @Override
10.   public void onFinish(ITestContext Result)
11.   { }
12.
13.   @Override
14.   public void onStart(ITestContext Result)
15.   { }
16.
17.   @Override
18.   public void onTestFailedButWithinSuccessPercentage(ITestResult Result)
19.   { }
20.
21.   // When Test case get failed, this method is called.
22.   @Override
23.   public void onTestFailure(ITestResult Result)
24.   {
25.     System.out.println("The name of the testcase failed is :"+Result.getName());
26.   }
27.
28.   // When Test case get Skipped, this method is called.
29.   @Override
30.   public void onTestSkipped(ITestResult Result)
31.   {
32.     System.out.println("The name of the testcase Skipped is :"+Result.getName());
33.   }
34.
35.   @Override
36.   public void onTestStart(ITestResult Result)
37.   { }
38.
39.   // When Test case get passed, this method is called.
40.   @Override
41.   public void onTestSuccess(ITestResult Result)
42.   {
43.     System.out.println("The name of the testcase passed is :"+Result.getName());
44.   }
45. }
```

Step 2: Add the listeners annotation (@Listeners) in the Class "ListenerTestNGTestCase"
The complete "ListenerTestNGTestCase" class after adding Listener annotation is mentioned below:

```
1.  package nextGenerationAutomationLearnTestNG;
2.
3.  import org.openqa.selenium.WebDriver;
4.  import org.openqa.selenium.firefox.FirefoxDriver;
5.  import org.testng.SkipException;
6.  import org.testng.annotations.Listeners;
7.  import org.testng.annotations.Test;
8.
```

```
@Listeners(listeners.ListenerTestNG.class)
public class ListenerTestNGTestCase {
    WebDriver driver= new FirefoxDriver();

    // Test to pass as to verify listeners .
    @Test(priority=1)
    public void TestToPass()
    {
        System.out.println("This method to pass test");
        driver.get("https://www.nextgenerationautomation.com/go-europe-model");
        driver.getTitle();
        driver.quit();
    }

    //Used skip exception to skip the test
    @Test(priority=2)
    public void TestToSkip ()
    {
        System.out.println("This method to skip test");
        throw new SkipException("Skipping - This is not ready for testing ");
    }

    // In the above method, we have already closed the browser. So we couldnot get the title here. It is to forcefully fail the test
    @Test(priority=3)
    public void TestToFail()
    {
        driver.getTitle();
        System.out.println("This method to test fail");
    }
}
```

Step 3: Execute the "ListenerTestNGTestCase" class. Methods in class "ListenerTestNG" are called automatically according to the behavior of methods annotated as @Test.

Step 4: Verify the Output in the console. You could find the logs in the console.
If you want to use listeners in multiple classes.
Add the below lines of code in the TestNG.xml file

<listeners>
<listener class-name="listeners.listenerTestNG"/>
</listeners>

Final testng.xml file will be like this:

```
<xml version="1.0" encoding="UFT-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Suite">
  <listeners>
    <listener class-name="listeners.listenerTestNG"/>
  </listeners>
  <test name="Test">
    <classes>
```

```
9.        <class name="listeners.ListenerTestNGTestCase">
10.     </classes>
11.   </test>
12. </suite>
```

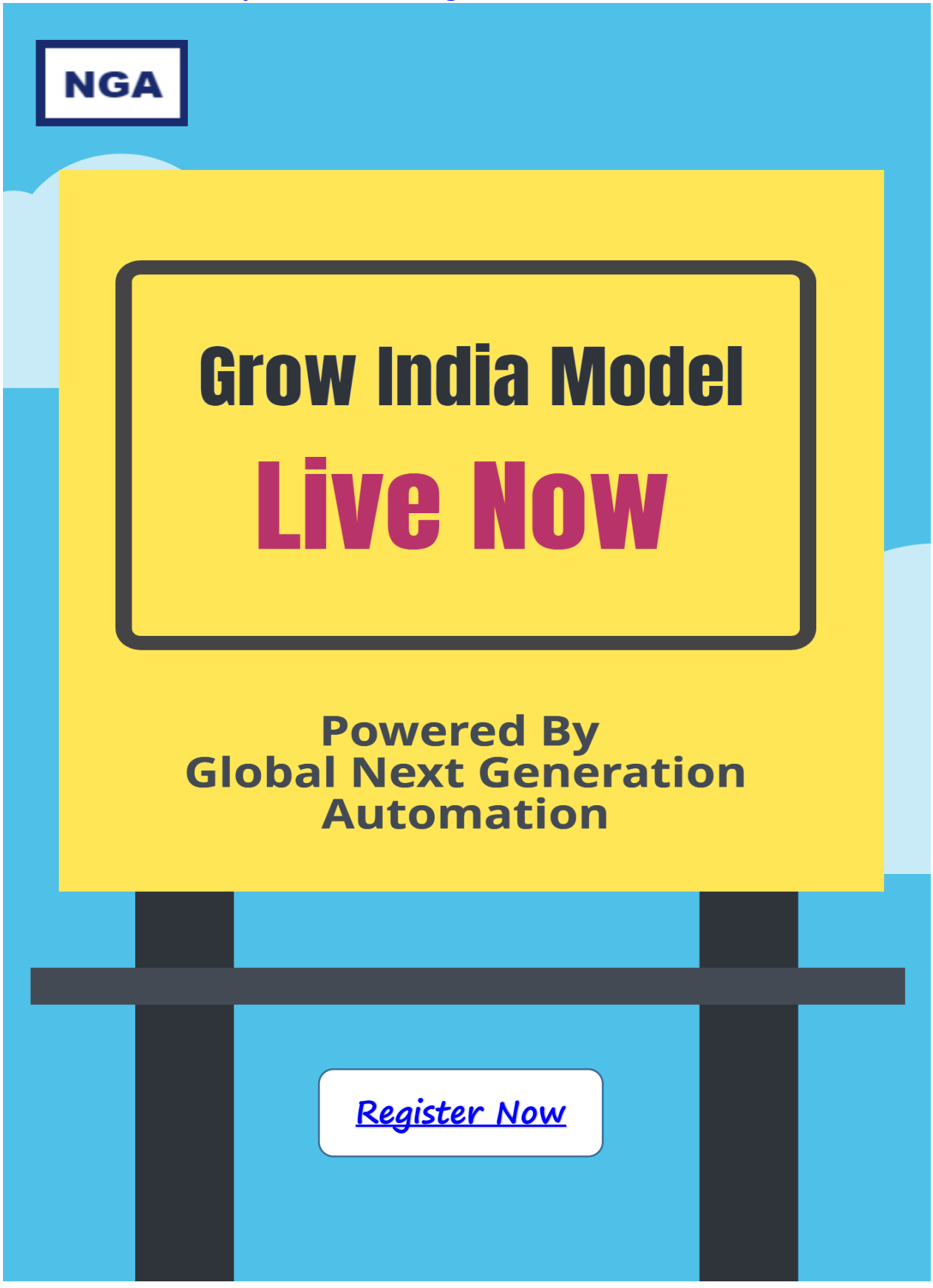
Execute it by right clicking on testng.xml and run as TestNG Suite

#NGAutomation

Building better for tomorrow

NGA

https://www.nextgenerationautomation.com



Thanks!!

Next Generation Automation Academy