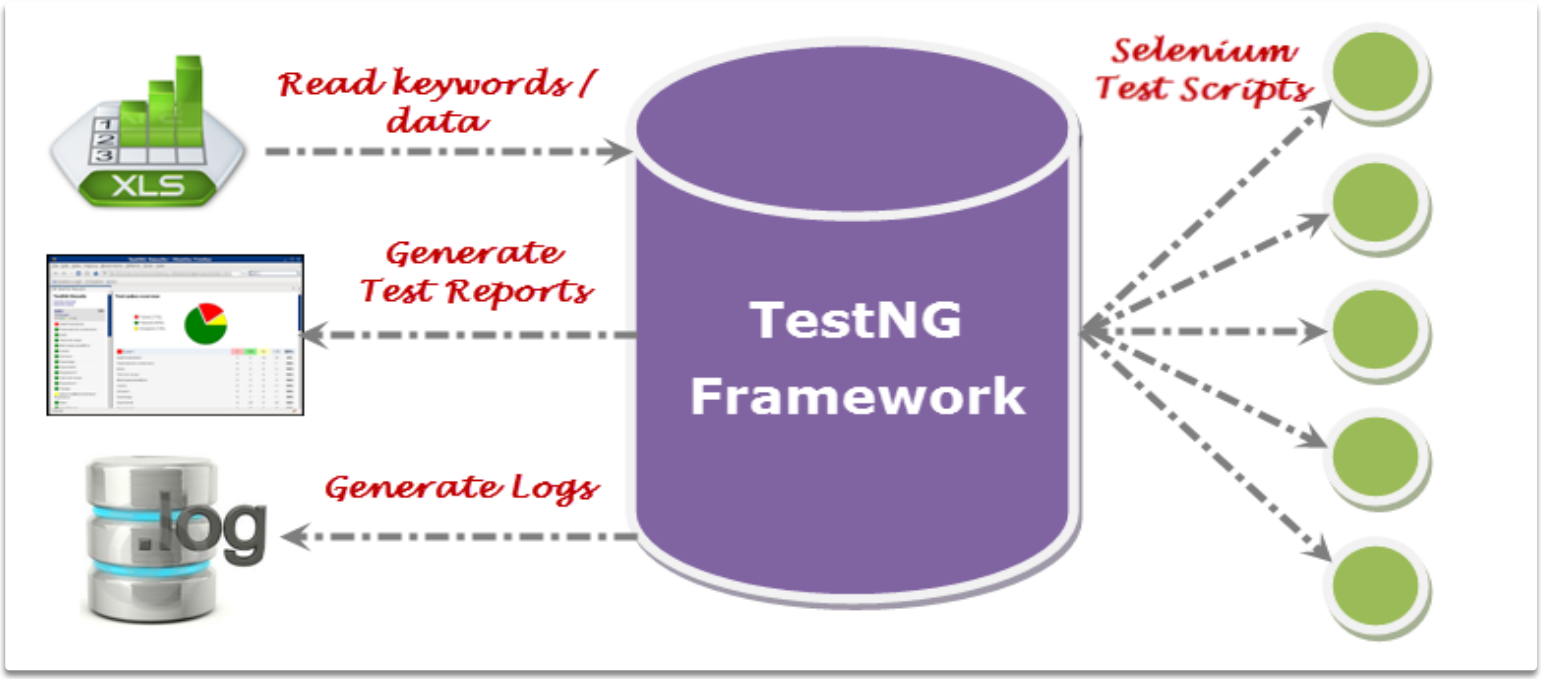
Test NG Framework Introduction for Selenium beginners

Why Test NG

- To Run test cases in the order which we provide (Priority , alphabetical etc.)
- We can run classes without using Main method and using annotations
- To generate reports
- We can Parameterize Tests
- Group tests and execute only specific group
- Implement logging using TestNG Listeners
- We can do the parallel execution

Why Test NG


XLS
Read keywords / data
Generate Test Reports
log
Generate Logs
TestNG Framework
Selenium Test Scripts

Demo - Test NG Down load and Install

Demo –Imp TestNG Annotations

ANNOTATION	DESCRIPTION
@BeforeSuite	The annotated method will be executed before any tests declared inside a TestNG suite.
@AfterSuite	The annotated method will be executed after any tests declared inside a TestNG suite.
@BeforeTest	The annotated methods will be executed before each test section declared inside a TestNG suite.
@AfterTest	The annotated methods will be executed after each test section declared inside a TestNG suite.
@BeforeClass	BeforeClass annotated method is executed before any of the test method of a test class.
@AfterClass	AfterClass annotated method is executed after the execution of every test methods of a test class are executed.
@BeforeMethod	These annotated methods are executed before the execution of each test method.
@AfterMethod	These annotated methods are executed after the execution of each test method.
@Test	Marks a class or a method as a test method. If used at class level, all the public methods of a class will be considered as a test method.

Demo- Selenium + TestNG

Soft Assertion and Hard Assertions

Hard Assertions

❖A Hard Assertion is type of assertion that throws an exception immediately when an assert statement fails and continues with the next test in the test suite

❖After the suite completes execution, the particular test is marked as passed instead of a FAIL.

Assertions

❖***assertEqual(String actual,String expected)*** :- It takes two string arguments and checks whether both are equal, if not it will fail the test.

❖***assertEqual(String actual,String expected, String message)*** :- It takes three string arguments and checks whether both are equal, if not it will fail the test and throws the message which we provide.

❖***assertEquals(boolean actual,boolean expected)*** **:-** It takes two Boolean arguments and checks whether both are equal, if not it will fail the test.

❖***assertEquals(java.util.Collection actual, java.util.Collection expected, java.lang.String message)*** **:-** Takes two collection objects and verifies both collections contain the same elements and with the same order. if not it will fail the test with the given message.

Hard Assertions

❖***Assert.assertTrue(condition)*** **:-** It takes one boolean arguments and checks that a condition is true, If it isn't, an AssertionError is thrown.

❖***Assert.assertTrue(condition, message)*** **:-** It takes one boolean argument and String message. It Asserts that a condition is true. If it isn't, an AssertionError, with the given message, is thrown.

❖***Assert.assertFalse(condition)*** **:-** It takes one boolean arguments and checks that a condition is false, If it isn't, an AssertionError is thrown.

❖***Assert.assertFalse(condition, message)*** **:-** It takes one boolean argument and String message. It Asserts that a condition is false. If it isn't, an AssertionError, with the given message, is thrown.

Soft Assertions

❖To deal with the disadvantage of Hard Assertions, customized error handler provided by TestNG is called **Soft Assertion**

❖Soft Assertions are the type of assertions that do not throw an exception when an assertion fails and would continue with the next step after assert statement

❖This is usually used when our test requires multiple assertions to be executed and the user want all of the assertions/codes to be executed before failing/skipping the tests.

❖In order to achieve the desired result we need to call the **assertAll()** method at the end of the test which will collate all the exceptions thrown and fail the test if necessary.

```
SoftAssert softAssert = new SoftAssert();

@Test
public void testCasetwo() {

System.out.println("*** test case two started ***");
softAssert.assertEquals("Hello", "Hello", "First soft assert failed - testCasetwo");
System.out.println("Soft assert success....");
softAssert.assertTrue("Hello".equals("hello"), "Second soft assert failed - testCasetwo");
softAssert.assertTrue("Welcome".equals("welcomeeee"), "Third soft assert failed - testCasetwo");
System.out.println("*** test case two executed successfully ***");
//softAssert.assertAll();
}

@Test
public void testCaseThree() {

System.out.println("*** test case three started ***");
softAssert.assertEquals("Hello", "Hello", "First soft assert failed - testCaseThree");
System.out.println("Soft assert success....");
softAssert.assertTrue("Hello".equals("hello"), "Second soft assert failed - testCaseThree");
softAssert.assertTrue("Welcome".equals("welcomeeee"), "Third soft assert failed - testCaseThree");
System.out.println("*** test case three executed successfully ***");
//softAssert.assertAll();
}
```

Data Providers

❖Marks a method as supplying data for a test method.
❖@Test method that wants to receive data from this DataProvider needs to use a dataProvider name equals to the name of this annotation.
❖The name of this data provider will automatically be set to the name of the method.
❖The annotated method must return an Object[][] where each Object[] can be assigned the parameter list of the test method.

```
@Test(dataProvider = "getData")
public void setData(String username, String password) {
System.out.println("you have provided username as::" + username);
System.out.println("you have provided password as::" + password);
}

@DataProvider
public Object[][] getData() {

Object[][] data = new Object[3][2];

// 1st row
data[0][0] = "sampleuser1";
data[0][1] = "abcdef";

// 2nd row
data[1][0] = "testuser2";
data[1][1] = "zxcvb";

// 3rd row
data[2][0] = "guestuser3";
data[2][1] = "pass123";

return data;
}
```

Test Case Priority

```
@Test
public void registerAccount()
{
System.out.println("First register your account");
}
@Test(priority=2)
public void sendEmail()
{
System.out.println("Send email after login");
}
@Test(priority=1)
public void login()
{
System.out.println("Login to the account after registration");
}
```

Test Case Depends On

```
@Test
public void methodOne() {
System.out.println("Helloo.. Im in method adding numbers");
}

@Test
public void MethodTwo() {
System.out.println("Helloo.. Im in method divided by zero");
int e = 1 / 0;
}

@Test(dependsOnMethods = { "MethodTwo" })
public void methodThree() {
System.out.println("Helloo.. Im in method skip");
}
```

Test NG XML – Execute Test Case using package Names

```java
package com.first.example;

import org.testng.annotations.Test;

public class demoOne {

	@Test

	public void firstTestCase()

	{

		System.out.println("im in first test case from demoOne Class");

	}

	@Test

	public void secondTestCase()

	{

		System.out.println("im in second test case from demoOne Class");

	}

}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>

<suite name="example suite 1" verbose="1" >

  <test name="Regression suite 1" >

    <packages>

      <package name="com.first.example" />

    </packages>

  </test>

</suite>
```

Test NG XML – Grouping Test Cases

```
@Test(groups="Regression")
public void testCaseOne()
{
System.out.println("Im in testCaseOne - And in Regression Group");
}
@Test(groups="Regression")
public void testCaseTwo(){
System.out.println("Im in testCaseTwo - And in Regression Group");
}
@Test(groups="Smoke Test")
public void testCaseThree(){
System.out.println("Im in testCaseThree - And in Smoke Test Group");
}
@Test(groups="Regression")
public void testCaseFour(){
System.out.println("Im in testCaseFour - And in Regression Group");
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="Sample Suite">
  <test name="testing">
        <groups>
       <run>
         <include name="Regression"/>
       </run>
     </groups>
     <classes>
        <class name="com.example.group.groupExamples" />
     </classes>
  </test>
</suite>
```

Test NG XML – Handle Exception

```
import org.testng.annotations.Test;
public class TestNGExamples {

        @Test(expectedExceptions=ArithmeticException.class)
        public void dividedByZeroExample1(){
                int e = 1/0;
        }

        @Test
        public void dividedByZeroExample2(){
                int e = 1/0;
        }
}
```

Test NG XML – Include or exclude test cases

```java
public class AddTestCase {

	@Test
	public void addLocationTestCase() {
		System.out.println("Im in add location test case");
	}

	@Test
	public void addDepartmentTestCase() {
		System.out.println("Im in add department test case");
	}

	@Test
	public void addEmployeeTestCase() {
		System.out.println("Im in add employee test case");
	}
}
```

```xml
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="Sample Test Suite" verbose="1" >
  <test name="Method Test Cases" >
    <classes>
      <class name="com.easy.entry.AddTestCase">
       <methods>
       <include name="addLocationTestCase" />
       <include name="addDepartmentTestCase" />
       <exclude name="addEmployeeTestCase" />
      </methods>
       </class>
    </classes>
  </test>
</suite>
```

Test NG XML – Parameterization

```java
public class TestParameters {


	@Parameters({ "browser" })
	@Test
	public void testCaseOne(String browser) {
		System.out.println("browser passed as :- " + browser);
	}


	@Parameters({ "username", "password" })
	@Test
	public void testCaseTwo(String username, String password) {
		System.out.println("Parameter for User Name passed as :- " + username);
		System.out.println("Parameter for Password passed as :- " + password);
	}
}
```

```xml
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name="Parameterization Test Suite">

	<test name="Testing Parameterization">

	<parameter name="browser" value="Firefox"/>

	<parameter name="username" value="testuser"/>

	<parameter name="password" value="testpassword"/>

		<classes>

			<class name="com.parameterization.TestParameters" />

		</classes>

	</test>

</suite>
```

Test NG XML – Parallel Methods

```
public class TestParallelOne {


	@Test
	public void testCaseOne() {
		//Printing Id of the thread on using which test method got executed
		System.out.println("Test Case One with Thread Id:- "
				+ Thread.currentThread().getId());
	}


	@Test
	public void testCaseTwo() {
		////Printing Id of the thread on using which test method got executed
		System.out.println("Test Case two with Thread Id:- "
				+ Thread.currentThread().getId());
	}
}
```

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name="Parallel test suite" parallel="methods" thread-count="2">

  <test name="Regression 1">

    <classes>

      <class name="com.parallel.TestParallelOne"/>

    </classes>

  </test>

</suite>
```

Test NG XML – Parallel Classes

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Parallel test suite" parallel="classes" thread-count="2">
  <test name="Test 1">
    <classes>
      <class name="com.parallel.TestParallelClassOne"/>
      <class name="com.parallel.TestParallelClassTwo"/>
    </classes>
  </test>
</suite>
```