TestNG

1. TestNG is an open source unit testing framework or unit testing tool where NG stands for Next Generation
2. TestNG is a unit testing TDD (Test Driven Development) framework, which supports java and .net.
3. TestNG was developed as an additional plugin for Eclipse
4. In case of automation, TestNG will be used to develop all the scripts using TestNG annotations and achieve batch execution without any manual interaction.
5. TestNG will be used to handle all framework component & help us to run all the test scripts in batch / parallel / group without any manual intervention.
6. TestNG.xml is main controller of the selenium framework, where we start the execution.
7. TestNG is used by automation engineers because of its features and advantages

Advantage:

- Generate Reports
- Batch Execution of Test Cases
- Group Execution of Test Cases
- Distributed parallel execution of Test Cases
- Perform Parameterization
- Perform Assertion
- Add preconditions and post condition
- Add dependency of test cases
- Prioritize the test cases
- Disabled the test cases.
- Provides different annotations

What is annotation?

Annotation is the java template which is used to give information or metadata to compiler, developer and runtime environment.

Annotation can be used on top of variable, methods or classes (if allowed).

Annotation always start with @symbol

1. @Test
2. @BeforeMethod
3. @AfterMethod
4. @BeforeClass
5. @AfterClass
6. @BeforeTest
7. @AfterTest
8. @BeforeSuite
9. @AfterSuite

10. @parameters
11. @dataProvider
12. @Listener

Installation steps of TestNG:

1. Go to Eclipse Window Click on help option-> Eclipse Marketplace
2. Write TestNG in find edit box and click on go button.
3. Find TestNG for eclipse division and click on install button
4. Click on confirm button and I accept the terms and conditions and click on finish
5. In order to verify the TestNG installation- →Go to windows →Show view →others →expand java folder, TestNG symbol will be present

@Test:

1. It is a core annotation of testing
2. Whenever we execute TestNG class, java Compiler/Interpreter always looks for @Test Annotation method to start the execution.
3. Without @Test , TestNG class will not be executed, @test annotation method act like main method in TestNG
4. In one TestNG class we can have multiple @test methods, but each test method should have @Test annotation before method signature.
5. Annotation method return type should be "void" and access specifier should be public, but method name can be anything (we have provide manual testNameTest).
6. As per the Rule TestNG class Name & TestNG method Name should end With "Test".
7. In one class we can keep multiple @test annotation , but in real time we are going maintain maximum 10 to 15 @test script , because maintenance will be easy.

Priority

Whenever we execute TestNG class, by default all the test methods will be executed based on Alphabetical Order, in order the change the Order of Execution, we go for priority.

DependsOnMethod :

Its help us to check the dependent test case is pass or fail, if dependent test-script get pass, execution will continue If dependent test-script get fail, skip the all other test script which is dependent on first test

Invocation Count:

Same test-script executed multiple times with same test data

@BeforeMethod and @AfterMethod

1. Before method annotations will be executed, before executing each @Test method in a class
2. After method annotation will be executed, after executing each @Test in a class
3. BeforeMethod & AfterMethod will not be executed, without @test annotation method, because they are configuration method

4. In order to implement similar pre- condition for all the test cases like “LOGIN code ” we go for @ BeforeMethod
5. In order to implement similar post-condition for all the test cases like “LOGOUT code ” we go for @ AfterMethod

@BeforeClass & @AfterClass

1. BeforeClass Annotation method will be executed, before Executing first @test in a class
2. AfterClass Annotation method will be executed, after executing all/last test-case within a class
3. BeforeClass & AfterClass annotations will be executed only once in an entire class execution.
4. It will be used to develop global configuration like launch browser, object initialization NOTE: As per the Rule of the Automation, there should not be any dependency between two test cases, every test case should be unique (it means every test should have fresh login & logout code).

Batch Execution

1. Collection of multiple test scripts together is called a batch, to execute multiple test scripts through the xml in a single click is called batch execution.
2. In order to achieve batch execution, we go for Testng.xml configuration file
3. TestNG xml file always start with suite xml tag followed by and
4. In one xml file we can invoke N-number of testng classes, but all the classes should be present within a project.

How to create testNG xml file automatically through eclipse?

- Select all the testNG classes or packages -> right click-> select➔ testNG ➔and click on “Convert to testing” and➔ click on finish.
- Automatically you will get the testng.xml file with in the project
- In order to edit xml File➔ double click on testing.xml➔ click on “Source”

Grouping Execution:

1. Collection of similar test scripts across the testing classes is called grouping Execution
2. In order achieve grouping execution , each & every test script should have group name, group name will be written along with annotation
3. In grouping execution, all configure annotation should have group name , other wise those annotation will not participate in grouping execution like

@BeforeSuite @BeforeClass , @BeforeMethode etc EG :

```
@BeforeSuite(groups = {"smokeTest","regressionTest"})
public void configBS() {
     System.out.println("======Execute BeforeSuite============");
}
```

Smoke Test:

```
@Test(groups={"smokeTest"})
public void createCustomerTest(){
        System.out.println("execute createCustomerTest");
}


@Test(groups={"regressionTest")
public void modifyCustomerTest(){
        System.out.println("execute modifyCustomerTest");
}
```

➔ In order to invoke grouping execution should declare Group Key in testing.xml file & group keep should be declared before <test>, after <suite> tag

Smoke Test:

```
<suite name="Suite">
        <groups>
                <run>
                        <include name="smokeTest"/>
                </run>
        </groups>
<test name="Test">
        <classes>
                <class name="pac1.ProjectAndCustomerTest"/>
                <class name="pac2.ReportTest"/>
</classes>
</test>
</suite>
```

➔We can invoke multiple group-key in one XML File

```
<suite name="Suite">
        <groups>
                <run>
                        <include name="smokeTest"/>
                        <include name="regressionTest"/>
                </run>
        </groups>
```

```
<test name="Test">
    <classes>
        <class name="pac1.ProjectAndCustomerTest"/>
        <class name="pac2.ReportTest"/>
    </classes>
</test>
```

Parallel Execution:

i. **Distributed Parallel execution:**

α. Distribute the test case across the multiple test runner & execute each <test>/Test runner in parallel is called Distributed Parallel execution

β. we reduce the suite execution time, so that we can get the result early

χ. in order the archive parallel execution we should enable parallel=”tests” & thread-count=5 in

d. <suite> , then create multiple test runner & distribute the testcase

ε. maximum thread count is 5

ϕ. Thread count should be same as number of <test> testRunner

ii. **Cross browser parallel execution / Browser Compatibility testing**

α. Execute same set of testcase in different Browser parallel is called cross browser testing

β. To achieve cross browser testing we should we <parameter> in XML file & @parameters annotation inside the testScript

1. **<parameter > is used to specify the browser data for each <test>**

EG :

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
    <suite name="Suite" parallel= "tests" thread-count="2">
        <test name="Test-Runner-Chrome">
            <parameter name="BROWSER" value="chrome"/>
            <classes>
                <class name="com.vtiger.comcast.organizationtest.CreateOrganization"/>
                <class name="com.vtiger.comcast.organizationtest.SearchOrgTest"/>
            </classes>
        </test> <!-- Test -->
        <test name="Test-Runner-Firefox">
            <parameter name="BROWSER" value="firefox"/>
            <classes>
                <class name="com.vtiger.comcast.organizationtest.CreateOrganization"/>
                <class name="com.vtiger.comcast.organizationtest.SearchOrgTest"/>
            </classes>
        </test> <!-- Test -->
    </suite> <!-- Suite -->
```

2. **@parameters annotation will be used to receive the data (Eg : browser , platform etc) from the XML file to test Scripts [Like BaseClass]**