04-MAR-2020

-----------

\*\*\* 4. TESTNG FRAMEWORK \*\*\*\*

----------------

TestNG stands for Test New Generation

It is an open source and java based Unit testing tool

It is built-in framework and similar to JUnit

\*\*TestNG is a built-in framework tool, which we can use in Selenium Webdriver to execute our test scripts without main() method in classes using annotations.

Advantages of TestNG

--------------------

--> Default it will create HTML reports

--> Manages test suites(test classes) and test cases execution

--> Dependency testing is possible

--> Using annotation we can control methods execution in a Test case

--> Supports Parameterization using XML file and Data Provider

--> Parallel testing is possible using Grid2

==================================================================================================================================================================

-- Test Suite means Test Classes in Selenium (Eclipse)

-- TestNG is a built-in framework and not User-defined framework.

-- We can't make changes to built-in framework

-- TestNG can be done on Unit testing (White Box testing)

-- This Unit(White-Box) testing is known as JUnit

-- If this unit testing is done on .NET it will be termed as NUnit

-- Unit testing is done on the source code level for each component and not on application level.

==================================================================================================================================================================

Test Data Test Script-1

------------

Keywords XML ---------------------------------------------- JUnit/TestNG --------------------------------------- Test Script-2

------------

HTML Reports Test Script-3

Log Test Script-4

==================================================================================================================================================================

Downloading And Configuration of TestNG to Java Project in Eclipse IDE

Step-1: (TestNG Installation)

Navigation

----------

"Help" menu in Eclipse-->Eclipse Marketplace-->enter TestNG in Search EditBox-->Click on Go-->Click on Install

Note: In "Help" menu --> we can go for Install new Software-- but we need to check for the Eclipse current version which we have or else will face compatibility issues, hence best option is to go for "Eclipse Marketplace".

After installation restart Eclipse

-- If we use Install new software- we need to search for compatible version

-- Instead go for Eclipse Marketplace- By default it will check our local machine eclipse version and install it by default accordingly.

Step 2:

------

Configure TestNG library: (i.e. to make Normal Java Project into TestNG Project)

Navigation

----------

Create Java Project

Right click on Java Project

Build Path

Libraries

Select "Add Library"

Select "Test NG"

Click on "Next"

Click on "Finish"

Note:

Configure WD jars also for "TestNG Project"

\*\* TestNG Annotations:

------------------

Annotations are used to specify sequence of methods execution without using main method in a Class

1. @Test: It is used to specify method under this execution is a test case (i.e. validation script)

For these @Test annotation methods TestNG will create Reports in HTML file and also provide status in terms of Pass/Fail in Console

Note:

Reporter.log();

It is to pass the user message into TestNG html reports and console.

Syntax:

------

Reporter.log (user msg, true/false);

true -- will pass the user message in html reports and also in Console.

false-- it will pass the user message into html reports only

Ex: Create TestNg class with @Test annotation method to pass the user message

into html reports and console

public class DemoClass {

@Test

public void userLogin() {

System.out.println("To validate login functionality");

Reporter.log("Sample script", true);

}

}

====================================================================================================================================================================

-- For reports we will use @Test

-- For validation related methods we will use @Test

-- We can use more than one @Test method annotation

Note:

----

-- In one class we can maintain one or more @Test annotation

-- By default these methods will execute in name alphabetical order

-- Where as based on requirement we can set execution sequence for @Test annotation methods using 'Priority'.\*\*\*

--------

Syntax:

------

@Test (priority=1)

Note:

----

Priority number can start from any number like 0,1,10,20,.....etc

Ex: create TestNg class with 2 @Test annotation methods and set the sequence

of those methods using “priority” option

publicclass DemoClass {

@Test (priority=0)

publicvoid userReg() {

System.out.println("To validate user registration");

}

@Test (priority=1)

public void userLogin() {

System.out.println("To validate login functionality");

}

}

====================================================================================================================================================================

2. @BeforeMethod: Method under this annotation will be executed before each @Test annotation method (i.e. @Test)

-------------

Under @BeforeMethod annotation method will execute depends on number of @Test annotations in that class

3. @AfterMethod: method under this annotation will be executed after each @Test annotation method(i.e. @Test)

------------

Note:

Under @AfterMethod annotation method also will execute depends on number of @Test annotation methods in that class

4. @BeforeClass: method under this annotation will execute prior to or beginning of class execution

------------

5. @AfterClass: method under this annotation will execute after all the methods are executed in a class

-----------

6. @Parameters: to get data from TestNG xml file

7. @DataProvider: It is also used to pass the data to the test case. It always returns two dimensional object array.

Note: Without @Test we cannot execute other annotations, atleast one @Test annotation should be present in the Class.

@BeforeClass

@BeforeMethod

@Test

@ AfterMethod

@AfterClass

---------------------

@BeforeClass

@BeforeMethod

@Test(P:1)

@AfterMethod

@BeforeMethod

@Test(p=2)

@AfterMethod

@AfterClass

====================================================================================================================================================================

Ex: create script to perform execution by taking following scenarios into individual methods

Scenarios:

This program to launch browser

This program to refresh the page after each transaction

This program is to validate user Registration operation

This program is written to validate login functionality

This script to set home page of application

package testng.pack;

import org.testng.Reporter;

import org.testng.annotations.AfterClass;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class DemoTestNg {

@Test (priority= 1)

public void userReg() {

System.out.println("To validate customer Registration");

}

@Test (priority= 2)

public void userLogin() {

System.out.println("To validate login functionality");

}

@BeforeMethod

public void homePg() {

System.out.println("Redirect to home page");

}

@AfterMethod

public void refreshPg() {

System.out.println("To refresh the page after each validation");

}

@BeforeClass

public void setUp() {

System.out.println("To initialize browser");

}

@AfterClass

public void tearDown() {

System.out.println("to close application");

}

}

========================================================================END OF CLASS==================================================================================