05-MAR-2020

-----------

ASSERTION IN SELENIUM TESTNG

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

TestNG Asserts are the most frequently used methods while creating Selenium Scripts

TestNG Asserts help us to verify the condition of the test in the middle of the test run

SYNTAX

------

Assert.assertEqual( actual value, expected value);

\*\*\*[ It is used to compare the values. If it isn't an Assertion Error is thrown in console and

it will terminate current method execution then it will go to another Test case to execute.]

Ex: Assert.assertEqual("Smith","Smith");

**Assert:**Assert command checks whether the given condition is true or false. Let’s say we assert whether the given element is present on the web page or not. If the condition is true then the program control will execute the next test step but if the condition is false, the execution would stop and no further test would be executed.

**Verify:**Verify command also checks whether the given condition is true or false. Irrespective of the condition being true or false, the program execution doesn’t halt i.e. any failure during verification would not stop the execution and all the test steps would be executed.

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

Ex: create TestNg class using Assert class to compare string values

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

package testng.pack;

import org.testng.Assert;

import org.testng.annotations.Test;

public class AssertEx {

@Test (priority= 1)

public void userReg() {

Assert.assertEquals("Smith", "Smith123");

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

}

@Test (priority= 2)

public void userLogin() {

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

}

}

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

Ex: Create TestNg Class tovalidate login functionality in OrangeHRM

project using valid data

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

package testng.pack;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.Reporter;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class OHRMLogin {

WebDriver driver;

@BeforeClass

public void setUp() {

System.setProperty("webdriver.chrome.driver", "./Drivers\\chromedriver.exe");

driver=new ChromeDriver();

driver.get("https://opensource-demo.orangehrmlive.com/");

driver.manage().window().maximize();

}

@Test

public void adminLogin() throws InterruptedException {

WebElement objUserName = driver.findElement(By.id("txtUsername"));

if (objUserName.isDisplayed()) {

Reporter.log("Username element found in Application and Script executing ....",true);

//to perform login operation

driver.findElement(By.id("txtUsername")).sendKeys("Admin");

driver.findElement(By.id("txtPassword")).sendKeys("admin123");

driver.findElement(By.id("btnLogin")).click();

Thread.sleep(5000);

String pgTitle=driver.getTitle();

Assert.assertEquals(pgTitle, "OrangeHRM");

Reporter.log("Successful login operation");

}

else {

Reporter.log("Username Element not found");

}

}

@AfterClass

public void tearDown() {

driver.close();

}

}

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

DEPENDENCY METHODS IN TESTNG CLASS

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

"DependsOnMethods" using this option we can specify current Test case execution based on other Test case method Pass/Fail

Syntax:

------

@Test (priority=2,dependsOnMethods= {"method1", "method2".....}

Ex: Create TestNg class using "dependsOnMethods" option

package testng.pack;

import org.testng.Assert;

import org.testng.annotations.Test;

public class AssertEx {

@Test (priority= 1)

public void userReg() {

Assert.assertEquals("Smith", "Smith123");

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

}

@Test (priority= 2, dependsOnMethods= {"userReg"})

public void userLogin() {

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

}

}

Note: ---> In console we get Skip-1 only when we use dependOnMethod for that particular method

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

\*\* Disable test methods in TestNG class

Sometimes we can disable particular test case execution in a class using "enabled" option

Syntax:

------

@Test (priority=2, enabled= false)

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

Ex: create TestNG class and use "enabled" option to disable 2nd test case

package testng.pack;

import org.testng.Assert;

import org.testng.annotations.Test;

public class AssertEx {

@Test (priority= 1)

public void userReg() {

Assert.assertEquals("Smith", "Smith");

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

}

@Test (priority= 2, enabled= false)

public void userLogin() {

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

}

}

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

\*\*\* I.Q\*\*\* What is the use of .xml file in Test NG?

----> For test suite execution

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

\*\*\* TESTNG SUITE EXECUTION

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

Using TestNG xml file we can execute multiple classes from different packages within the java project and also we can include/exclude some methods from those classes

Create TestNG xml file:

Navigation:

Select Java Project

Right click on mouse

Select "File"

Enter file name with ".xml" extension

----> Click on "Finish"

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

Code in testng .xml file for test suite execution:

Select "Source View" in testng.xml file and write following XML code

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

< suite name = " ">

< test name = " ">

<classes>

<class name=" packagename.classname"/>

<methods>

<include name="method name">

<exclude name="method name">

</methods>

</classes>

</test>

</suite>

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

Note:

One suite can have multiple tests

One test can have multiple classes

One class can have multiple methods

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

Ex: Create TestNg Xml file to execute tests from different classes from different packages

Procedure:

Step 1: Create new package with name as “sampleone”

Step 2: create 2 classes in “sampleone” package with 2 @Test annotation methods

Class-1:

package sampleone;

importorg.testng.annotations.Test;

publicclass Class01 {

@Test

publicvoid methodOne(){

System.out.println("Method-1 from Class01");

}

@Test

publicvoid methodTwo(){

System.out.println("Method-2 from Class01");

}

}

Class-2:

package sampleone;

import org.testng.annotations.Test;

publicclass Class02 {

@Test

public void methodOne(){

System.out.println("Method-1 from Class02");

}

@Test

public void methodTwo(){

System.out.println("Method-2 from Class02");

}

}

Step 3: Create new package with name as “sampletwo”

Step 4: Create class in “sampletwo” package with 2 @Test annotation methods

Class03:

package sampletwo;

import org.testng.annotations.Test;

publicclass Class03 {

@Test

publicvoid methodOne(){

System.out.println("Method-1 from Class03");

}

@Test

publicvoid methodTwo(){

System.out.println("Method-2 from Class03");

}

}

Step 5: create TestNg xml file to execute all the methods from different classes (i.e. Class01, Class02 and Class03 classes)

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

TestNg xml file

<suitename="my Demo testNg suite">

<testname="Sample tests">

<classes>

<classname="sampleone.Class01"/>

<classname="sampleone.Class02"/>

<classname="sampletwo.Class03"/>

</classes>

</test>

</suite>

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Create TestNg xml file to execute some of the methods only

TestNg xml file code

<suitename="my Demo testNg suite">

<testname="Sample tests">

<classes>

<classname="sampleone.Class01"/>

<methods>

<includename="methodOne"/>

<excludename="methodTwo"/>

</methods>

<classname="sampleone.Class02"/>

<classname="sampletwo.Class03"/>

</classes>

</test>

</suite>

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

Note:

@Before Test:

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

This annotation method will execute before test tag in testng .xml file

@After Test:

-----------

This annotation method will execute after test tag closed in testng .xml file

@Before Suite

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

This annotation method will execute first in Test Suite

@After Suite

-----------

This annotation method will execute last in Test Suite

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

Using @Parameters Annotation

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

Using this parameter we can read the values from xml file

SYNTAX

------

@Parameters(("parameter1", "parameter2",......))

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

EX01:

Create TestNg class to validate login functionality in Orange HRM application

by reading the data from Testng xml file

// This program is mentioned in the next Day32 doc.

Types of Console Error Messages:

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

SAXParseException -- <classes> in testing.xml file

java.lang.NoClassDefFoundError:

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